MITIGATING PHISHING THREATS

Yunjia Wang

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews

2023

Identifiers to use to cite or link to this thesis:

DOI: https://doi.org/10.17630/sta/411
http://hdl.handle.net/10023/27439

Mitigating Phishing Threats

Yunjia Wang

This thesis is submitted in partial fulfilment for the degree of

Doctor of Philosophy (PhD)

at the University of St Andrews

December 2022

ABSTRACT

Due to the rapid development of the Internet, modern daily behaviour has become more efficient and convenient. The Internet has become an indispensable element in our daily life, providing significant resources to people whether for play, work or education. In addition, with the increased universality of mobile devices, a magnitude of services is at our fingertips, the efficiency of our life or work has improved. However, the negative side of this is the increase in cybercrimes, with large losses for both individuals and enterprises.

Phishing is currently defined as a criminal mechanism employing both social engineering and technical subterfuge to gather any useful information such as user personal data or financial account credentials. Phishing threats have been in existence for many years, since the establishment of the Internet, and they have continuously evolved and increased in application. So far, phishing attacks have accounted for a large proportion of all malicious attacks, and they are a globally growing threat with an increasing frequency of known attacks. Phishing attacks are a major current cyber threat as they are always cheap to produce and easy to deploy, in particular, due to the development of E-commerce, either to an individual user or organization. For the individual, sensitive credentials are always of interest to phishers due to the development of E-commerce. For an enterprise, a successful phishing attack, such as a subdomain takeover attack, may affect their organization's reputation as well as cause financial loss.

Currently, most security vendors have been using different approaches to prevent phishing attacks. However, these solutions cannot keep up with the constant updating of phishing websites. In this thesis, web phishing attack types are classified into three different categories, from the shallower to the deeper. They are General Phishing Attack, Advanced Phishing Attack and Subdomain Takeover Attacks. The purpose of this thesis is to present an effective mitigation to defend against these phishing threats. From the shallower approach to a deeper, more complex approach, according to our defined categories of phishing threats, the specific mitigations and contributions are presented.

ACKNOWLEDGMENTS

A PhD is a four-year research, based on one specific topic, from the shallower to the deeper. In order to obtain a valuable contribution, we put much effort on research day by day. During this period, I have encountered many difficulties, both in academia and in life. However, with the help of my supervisor, family and friends, I have finally overcome these obstacles.

First and foremost, I would like to thank both my parents for their continued mental and financial support for me.

Secondly, I am sincerely grateful for my supervisor, Dr. Ishbel Duncan for her valuable and unforgettable support throughout my research. The interaction between her and me has been a truly insightful experience both academically and personally, with each meeting bringing me closer to my goal for this research. I am extremely appreciated for her help.

Thirdly, I owe thank to my friends who have provided insightful advice and support during my research, including Dr Xu Zhu, Dr Xuefeng Liu, Mr Minghao Zhong, Ms Yaoyuan Lu and Mr Hanhui Liao.

Then, I am much appreciated NSFOCUS Technologies Group Co., Ltd for providing me an internship opportunity for completing this research in a better way. In particular, I am very thankful to Mr Tiejun Wu for his intellectual help and critical comments.

Lastly, I would like to thank China Scholarship Council and University of St Andrews who support and fund my research.

FUNDING

Yunjia Wang was supported by the China Scholarship Council and University of St Andrews Scholarship.

DECLARATION

The following is an agreed request by candidate and supervisor regarding the publication of this thesis:

Printed copy

No embargo on print copy.

Electronic copy

No embargo on electronic copy.

Date	29/11/2022	Signature of candidate
Date	29/11/2022	Signature of supervisor

Underpinning Research Data or Digital Outputs

Candidate's declaration

I, Yunjia Wang, hereby certify that no requirements to deposit original research data or digital outputs apply to this thesis and that, where appropriate, secondary data used have been referenced in the full text of my thesis.

Date	29/11/2021	Signature of candidate

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Background

Due to the rapid development of the Internet, modern daily behaviour has become more efficient and convenient. In particular, with the emergence of e-commerce, through online banking and cloud services, a variety of data, such as personal information and enterprise sensitive data, are transferred through multiple networks [1]. The Internet has become an indispensable element in our daily life, providing significant resources to people whether for play, work or education. In addition, with the increased universality of mobile devices, a magnitude of services are at our fingertips, improving the efficiency of our social life or work [2]. Meanwhile, as shown by existing data from Juniper Research, cyber-crimes have significantly increased, with large losses for both individuals and enterprises [3]. In 2019, the global loss was estimated at 2 trillion dollars, and at least 65% of US organisations experienced a successful phishing attack [4].

Phishing attacks have accounted for a large proportion of all malicious attacks as they are always cheap to produce and easy to deploy. They are a globally growing threat with a continually increasing attack frequency [5]. In the UK, the official Cyber Security Breaches Survey 2020 showed the degree of cyber security threats has not been diminished, but has evolved and become more frequent. The most common type of cyber-attacks that UK organisations have experienced are phishing attacks where either staff receive fraudulent emails or are redirected to illegitimate websites. In comparison to other types of cyber-attacks, such as virus, malware and ransomware, phishing threats are the largest proportion [6]. Furthermore, from the detailed statistics, the cyber threats that UK organisations have experienced over four years, from 2017 to 2020, have undergone a significant evolution. The phishing attacks continually increased from 72% up to 86%. Other threats have had a relative reduction, from 33% to 16% and from 17% to 8% respectively [6].

The Federal Bureau of Investigation (FBI) in the USA revealed that phishing was the most common type of cybercrime in 2020, and that the incidents were almost double the frequency of the 2019 incidents. From their annual report on the comparison of top crime types in 2020, phishing was considered a primary weapon in cyber-attacks, with the largest proportion at 56% [7], as shown in Figure 1 below.

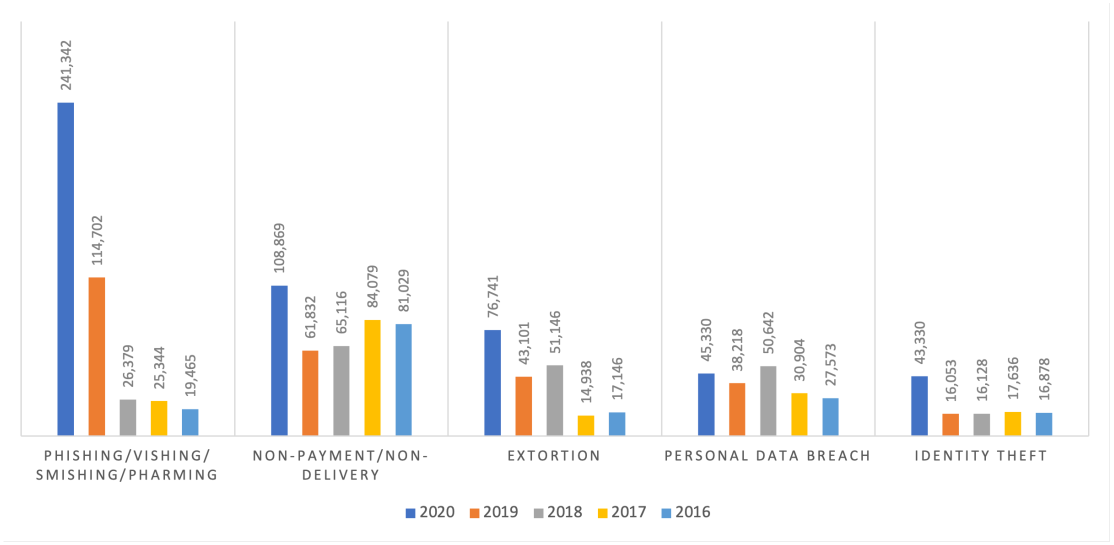


FIGURE 1: The FBI comparison for the top five reported crime types since 2016 to 2020 **[7]**.

Furthermore, the degree of phishing attack growth has been prominent since 2020, as the rapid spread of COVID-19 resulted in companies creating remote workforces and operating through cloud-based platforms [4]. Tessian Research published statistics showing that about 47% of employees have indicated that distraction (from work) is one of the reasons for falling for a phishing scam, especially while working from home [8]. In April 2020, 18 million daily malware and phishing emails related to COVID-19 were blocked by Google alone [9]. Due to the COVID-19 pandemic, AL-Qahtani and Cresci noted that phishing was the dominant type of attack at 69.2% [10].

Phishing threats have been in existence for many years, since before the establishment of the Internet, and they have continuously evolved and increased in application. Phishing attacks are a major current cyber threat, in particular, due to the development of E-commerce [11] [12]. The first time that the phishing term and concept appeared in the public perspective can be traced back to the 1990s through America Online (AOL). A group of attackers (hackers), also known as "phishers" (in this thesis, the attacker who implements phishing attack will be described with the term of phisher), called users themselves purporting to be AOL staff in online communities. They attempted to collect the user AOL login credentials and personal information [13].

Subsequently, phishers attention shifted to online payment systems. In June 2001, the first known phishing attack happened on an eCommerce website, the E-Gold website. This campaign planted a critical seed for further phishing attacks and evolution, even though it was unsuccessful [14]. The first successful phishing attack on payment systems took place in late 2003 [14]; phishers registered several domains that were similar to legitimate websites, such as eBay and PayPal. The spoofed emails were sent to PayPal users through email worm programs, wherein the victims were redirected to an illegitimate PayPal website and asked to update their credit card details and other personal information. By the beginning of 2004, phishers had started to conduct phishing attacks on banking sites and their customers. Between May 2004 and May 2005, phishing attacks resulted in around 1.2 million users suffering financial losses in the USA, totalling $929 million [14].

So far, phishing is evolving into a profitable business, especially, for online criminals. According to estimates, nearly 3.4 billion phishing e-mails have been delivered for launching phishing attacks per day in 2022 [15]. Phishing is currently defined by the Anti-Phishing Working Group (APWG) as a criminal mechanism employing both social engineering and technical subterfuge to gather the user’s personal identity data and financial account credentials [16], such as private user information or bank information etc. As the data summarised from Verizon shows [17], the top five types of data that attract phishers to compromise under a phishing campaign are:

- Credentials (passwords, usernames, pin numbers)
- Personal data (name, address, email address)
- Internal data (sales projections, product roadmaps)
- Medical (treatment information, insurance claims)
- Banking information (account numbers, credit card information)

1.2. The Classification of Phishing Attacks

Although the exact classification (type) of phishing attacks is different between academic papers and technical reports, we summarised various types of phishing attack from [18] [19] [20] [21] [22] [23]. These types and descriptions are shown below in Table 1.

Phishing Type	Description
Email-phishing	Also named “deception phishing”, which is one of the most well-known phishing types. Phishers impersonate a legitimate person or organization and send mass emails to victims, leveraging social engineering tactics. The content of these suspicious emails usually asks the victims to type in information or go to malicious websites for either stealing their credentials or installing malicious code on the victim’s device.
Web phishing	Also named “HTTP(s) phishing”, “phishing redirection”, “link manipulation”, “website spoofing”. Phishers design a fake website that has a high visual similarity to a well-known website, leading the victim to access this suspicious website for gathering their credential information via a social engineering approach.
Spear phishing	This uses email to transmit phishing information for stealing victim credential information. Compared to email phishing, spear phishing targets more sophisticated and targeted individuals, rather than sending out mass emails to thousands of victims. This type of phishing is often more personalized in order to make the victim trust that they have a relationship with the sender.
Whaling	Also named “CEO fraud”. An even more targeted type of phishing which typically targets a CEO, CFO or any high-level executive within an industry or a specific organization. The Phisher always impersonates a trusted role with a high-pressure situation to hook their victim, such as the company allegedly facing legal consequences, or transferring money, or for reviewing a document and entering critical data.

SMS phishing	Also named “smishing”. This leverages text messages rather than email to carry out a phishing attack. A common SMS phishing delivers a message to victims through SMS that contains a clickable link. The victims will be redirected to a suspicious website once they click this link. In this suspicious website, the victims are required to either type in their credential information or install malicious application.
Voice phishing	Also known as vishing, this is similar to smishing in that a phone call is used as the vehicle for launching a phishing attack instead of a text message. A vishing call often relays an automated voice message from what is meant to seem like a legitimate institution, such as a bank or a government entity.
Search engine phishing	Also known as “SEO poisoning” or “SEO Trojans”. The Phisher designs a fake website and it becomes indexed on legitimate search engines. Once the victims clicked this link, the victims will be redirected to the phisher’s website, and required to type in their credential information.
Pharming	A combination words of “phishing” and “farming”, this is a more technical attack and often hard to identify, and is also known as “DNS hijacking”. A Phisher exploits the mechanics of Internet browsing to redirect a victim to a designed malicious website with incorrect IP addresses for gathering their credential information by targeting DNS (Domain Name System) servers.
Evil twin phishing	In this attack, phishers set up a fake Wi-Fi hotspot that looks legitimate and lures victims to a phishing site when they connect to it. Once the victim lands on this malicious website, they may be asked to enter their personal data. Alternatively, a phisher can leverage a man-in-the-middle attack for intercepting the victim’s credentials or sensitive information transferred across the connection under this attack.
Social media phishing	The Phisher launches a phishing attack under the vehicle of social networking sites, such as Facebook, to gather a victim’s sensitive data or to lure them into clicking a link to access a phishing website.

Malware, keylogger and web trojan	These three attacks use malicious variants of applications, wherein the phishers, always using phishing emails, transmit these malicious applications to either ask a victim to download and install an app directly or redirect a victim to a phishing website to download and install apps or cookies, for achieving their goal of obtaining a victim's credential information.

TABLE 1: Phishing types and their behaviours **[18] [19] [20] [21] [22] [23]**.

From the table above, the common feature of these phishing types is that the phishers require to camouflage themselves and convince their victims that they are interacting with official (or legitimate) organisations. In order to achieve phishers' goal, phishers may consider using various ways to interact with their victims, such as through messages, email or telephone calls. There is no accurate data online to reveal the quantification of each kind of phishing attack, because the boundary of classification among the attacks is fuzzy. For example, if a phisher launches a phishing attack via email for a specific individual only, this activity belongs to both spear phishing and email phishing. Moreover, if this email attaches a suspicious link that redirects victim to a phishing website, this phishing attack could also be classified as web phishing. Therefore, current phishing attacks often combine multiple approaches. However, most phishing attacks are directed at a specific person or organization, called spear phishing attacks. A report from CISCO 2021 Cybersecurity Threat Trends revealed that spear phishing is the most common type of phishing attack, comprising 65% of all phishing attacks [24].

Email is a common vector to launch a phishing attack [23] [25]. In the Verizon 2022 Data Breach Investigations Report [26], it was disclosed that around 96% of phishing attacks start by email. The most common phishing attack is the combination of email phishing and web phishing, which is when a phisher sends an email to transmit content that attaches a phishing link that redirects victims to a "well-known" website (a fake website, or suspicious site built by phisher) [25]. According to the latest prevalence report of phishing websites from Google Safe Browsing [27], the previously leading category of unsafe websites has been replaced by phishing sites since 2016. In 2021, the prevalence of phishing sites is approximately 75 times the number of malware sites, as shown in Figure 2 below.

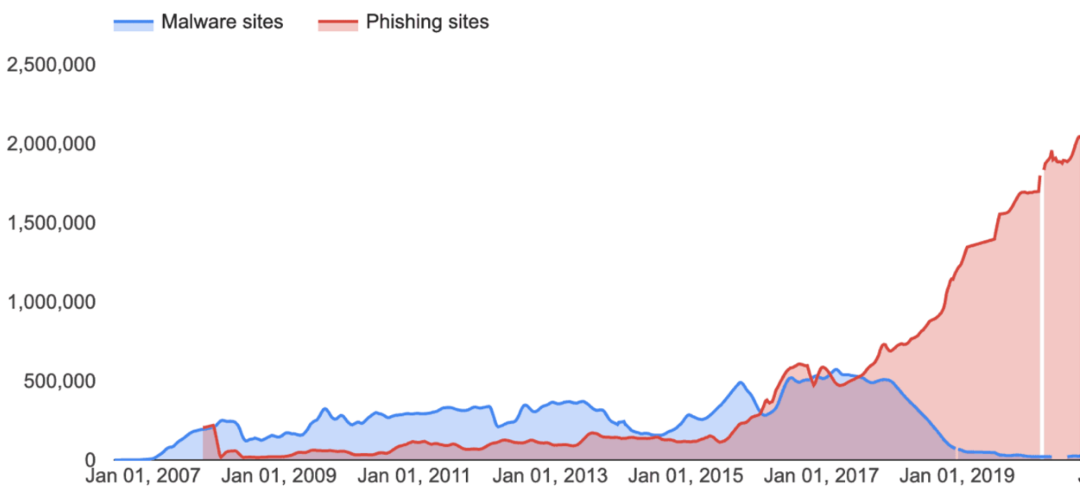


FIGURE 2: The comparison between Malware sites and Phishing sites since 2007 **[27]**.

Therefore, in order to analyse and overcome these changing and increasing trends, phishing threats are worth researching to determine better and complete solutions.

1.3. Research Direction and Goal

Most of phishing research mitigates phishing attacks in technical way focus on two aspects. Some research papers discuss preventing phishing from the transmission medium, such as how to identify the received email is a phishing email, as most phishing attacks start by email. Some research papers focus on the verification of web phishing, such as how to confirm the accessed website is a phishing website, because most phishers exploit a malicious website to illegally gather the victim data. Further details are given and reviewed in a later chapter (see the Literature Review Chapter).

Given the rise of phishing websites noted above, in our research we initially focussed on the verification of web phishing, where a camouflaged and suspicious (phishing) website is required to satisfy the following criteria:

1. A Phisher impersonates a well-known website by camouflaging the whole or part of the target (official) site.
2. The camouflaged website has a high visual similarity to its target.

This camouflaged website may be sent via various social engineering ways, such as via message, email and social applications (Facebook, WhatsApp). We do not pay attention to how this camouflaged website is transmitted; we only focus on the consequences to victims when they receive this malicious link and access this suspicious website. Also, after the victims access this camouflaged website, there are many potential scenarios, for example, the victims are required to type in their sensitive information (such as username, password and bank information), or the victims are asked to install an application. All of these scenarios are the objective in our research, we consider presenting an approach to identifying these phishing attacks immediately after the victim accesses camouflaged websites, and before they conduct any action, such as typing in sensitive information or installing any application.

Furthermore, web phishing attacks are complicated; features that are used to identify the phishing website are various due to the different attack types, which means a common mitigation may not be suitable for all web phishing attacks. Thus, in our research, web phishing attacks were further categorised into three types according to attack sophistication and features, and we present specific mitigations for each.

We called these types general phishing attacks, advanced phishing attacks and subdomain takeover attacks. Under general phishing attacks, the phishing website may have the identical content to the related official (legitimate) website, although the URL address and SSL certificate are different to the official target website. Currently, many users are sensible about suspicious activity (abnormality) from suspicious URL addresses or obvious warning information from browsers [28] [29] [30]. However, in some phishing attacks, we call these advanced phishing attacks, these abnormalities will be eliminated through implementing some technical ways, for example, if the victim is under a hijacking attack, either through a DNS (Domain Name System) hijacking attack or by an ISP (Internet Service Provider) hijacking attack. Here, both the URL address and the website content are the same as the related official site. In January 2019, the UK National Cyber Security Centre (NCSC) issued an alert regarding an ongoing large-scale global DNS attack, which affected government and commercial organisations worldwide [31] [32]. Moreover, there is another type of phishing attack, termed the subdomain takeover attack. A Subdomain Takeover has the highest concealment amongst these three kinds of attack, as even the SSL certificate is the same as the corresponding official site. Recent research elsewhere revealed that the subdomain takeover issue is a serious and widespread security risk, although this attack has received much less attention than other kind of attacks in the web security literature [33] [34] [35]. A successful subdomain takeover attack has a higher threat level as a controllable subdomain shares the same SSL certificate with its parent website. Squarcina et al [35] examined 50K most popular domains, including CNN and Cisco, to quantify the threats posed by sub domain attacks and identified 887 vulnerabilities. Moreover, from the HackerOne report [36], the threat of a subdomain takeover is not rare, but it is complex and consequently has a higher bonus for the security researcher through online challenges. For example, Patrik Hudal found two subdomain takeover issues on Starbucks on June 26 2018 and May 23 2019 respectively [37] [38]. The subdomain *mydailydev.starbucks.com* and *svcgatewayus.starbucks.com* were available to register by anyone, and he was awarded $4000 by Starbucks. Subsequently, Parzel Zenker submitted another report to Starbucks and was awarded a $2000 bonus on August 29 2019, as the subdomain *datacafe-cert.starbucks.com* had a CNAME record pointing to an unclaimed Azure webservice [39].

The features of these three web phishing attacks can be summarised in Table 2 below, where "High" means the original and the attack site can have high visual similarity, "Different" means that they are overtly distinguishable, "Same" means that they are unequivocally the same.

The Difference between Phishing webpage and Official webpage		***Web Content***	***URL Address***	***SSL Certificate***
General Phishing		High	Different	Different
Advanced Phishing	*DNS hijacking*	High	High	Different
	ISP hijacking	High	High	Different
Subdomain Takeover		High	High	Same

TABLE 2: Summaries of three phishing attacks.

Phishers may impersonate a well-known organisation with a higher concealment for achieving their final goal, either illegally obtaining victim's sensitive data or stealing a victim's asset. In these three web phishing attacks, from the general phishing attacks to subdomain takeover attacks, both attack sophistication and concealment degree are increased, and the user's security awareness and the possibility of user identification to these suspicious activities are reduced. In general phishing attacks, a user can identify suspicious activity from the URL and SSL certificate; In an advanced phishing attack, an SSL certificate can still be used as a criterion for recognizing suspicious activity. However, in a subdomain takeover attack, a user cannot identify this suspicious activity due to the high visual similarity of web content and an identical SSL certificate. In the latter two types of attacks, these can be considered as phishing attacks being launched through technical means (such as exploiting network features and configuration flaws). The result is that the detection is more challenging.

Therefore, in our research, we classified web phishing attacks into three different categories, from the shallower to the deeper, from the General Phishing Attacks, through Advanced Phishing Attacks to the deeper Subdomain Takeover Attacks. We are the first to classify web phishing attacks this way, which is also one of our key contributions. In further chapters of this thesis, phishing threats from these three categories are discussed and addressed.

1.4. Motivation of Research

Undeniably, there is much research about the presentation and evaluation of phishing threats as these risks have not been eliminated or reduced. In the first quarter of 2020, phishing accounted for 1 in every 4200 emails [4]. Meanwhile, other potential risks are derived. For example, Spear Phishing is the number one infection vector employed by 71% of organized groups according to the Internet Security Threat Report from Symantec [40]. Also, most Advanced Persistent Threat (APT) groups have used spear phishing as the initial infection vector [40]. Furthermore, with the increased usage of QR codes, QR phishing presents a new threat [41]. For phishing emails and QR phishing, the malicious URL are often made invisible, as most unwary users do not check the accessed address [42]. A survey from Wandera in 2018 shows a new phishing site was created every 20 seconds [43].

There are various techniques that can be deployed to launch a phishing attack. With the increase in technical approaches, phishing attacks have become more sophisticated and have caused significant impacts on both individuals and organizations [27]. For individual victims, phishing is a technique used by phishers to redirect users into an illegitimate website, to either steal user's data or sensitive credentials or trick a user into installing (executing) a malicious attachment. Most of these kinds of phishing attacks result in the loss of both property and privacy for individuals. For the organizations, phishing attacks may have an effect on both the organization and their staff. Staff may face the same consequences as individuals; thus we may put staff in the category of individuals. However, the effect on the organization may also be extremely serious. A successful duplicate phishing website can result in their users questioning both the organization's security and professionalism, so not only is there potential corporate and user loss, but also organizational reputations are being challenged.

Phishing attacks have caused heavy economic losses, 90% of data breaches occur due to phishing [24]. As the report revealed from Proofpoint [44], phishing attacks now cost large organizations almost $15 million annually, or more than $1,500 per employee. Lloyds of London [45] placed the cost of cyber-attacks in the region of $120 billion with phishing attacks taking nearly three weeks to resolve. Also, the most recent projections performed showed the average loss in phishing is $14.8 million in 2021, which is more than triple in 2015. The FBI 2020 report disclosed that spear phishing attacks cost US business more than $1.8 billion in 2020, up from $1.7 billion in 2019 [7]. According to IBM's 2021 Cost of a Data Breach Report, Phishing is the most expensive cause of data breaches, a breach caused by phishing costs businesses an average of $4.65 million [46]. In the second quarter of 2022, the Anti-Phishing Working Group (APWG) identified over 1.09 million phishing attacks, the worst quarter ever, with an average business email request of over $100,000 [47].

Problematically, present research and existing solutions are not very effective at mitigating phishing attacks. Many phishing detection approaches focus on the prevention of general phishing attacks, but there may not be good enough to detect advanced phishing attacks and subdomain takeover attacks. According to our defined categories, there are several limitations in the current solutions, which are discussed in detail in Chapter 3. In the prevention of General Phishing Attacks, most solutions use a blacklist or a content-based (such as, machine learning) approach [48]. The use of blacklists has a higher accuracy, but they cannot be used to defend against a zero-day phishing attack as sites are not detected and registered in real-time [43]. The accuracy of machine learning is subject to the integrity and complexity of the data set and training features, and the false positive of predicted results is a considerable factor, as some machine learning based approaches always result in a higher false predicted result [49]. In the mitigation of Advanced Phishing Attacks, there are many tools that can detect the user network status to isolate DNS hijacking, but a phishing website is difficult to be actively detected through existing anti phishing solutions if this phishing website is under a DNS hijacking attack. Also, ISP hijacking is quite hard to identify and recognize as the network traffic appears benign as it comes from a legitimate source (More details see Chapter 5). Furthermore, hackers have already focused their attention on mobile devices as users have increasingly shifted towards using mobile devices for daily tasks [50]. Mobile application users are more vulnerable to phishing attacks due to the limited resources on mobile devices [51]. In the detection of a Subdomain Takeover Attack, most current tools are used to query the relevant subdomain about the target website, but, according to our recent research [52], there is currently not an automatic approach or tool (with a higher accuracy or a faster speed) to discover the subdomain which has the potential takeover risk.

1.5. Research Questions & Hypotheses & Publications

The purpose of this thesis is to present a better and effective mitigation to defend against web phishing threats. Firstly, according to both attack sophistication and features, web phishing attacks was further classified into general phishing attacks, advanced phishing attacks and subdomain takeover attacks, which is one of our contributions. Next, from the shallower approach to a deeper, more complex approach, according to our defined categories of phishing threats, the research questions, hypotheses and publications for each part are summarised as follows:

For General Phishing Attacks:

In this research, we present a novel method to protect against phishing attacks. By using image recognition (OCR) technology, phishing attacks can be distinguished by reading the logos on a website and comparing them against the related legitimate site. An easy to implement prototype demonstrated a high accuracy of detection in the experimental trials.

The following research questions were considered:

1) What is an effective solution to prevent a phishing attack?
2) What is the limitation of the current phishing prevention approaches?
3) How can we overcome the limitation of mobile devices when we aim to prevent phishing attacks?
4) What is the difference regarding security mechanisms against phishing attacks between the Android and IOS Systems on mobile devices?

We presented the first hypothesis in this part when investigating effective solutions:

*H1: "**An OCR approach can be implemented efficiently to detect phishing attacks without using machine learning to predict and category the results**"*

A paper on this topic, entitled "**A Novel Method to Prevent Phishing by using OCR Technology**", was accepted and published in the ***IEEE 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*** [2].

For Advanced Phishing Attacks:

In this research, we present a novel approach to determine the security of the user network against ISP hijacking attacks through monitoring the consistency and differentiation of the associated Document Object Model (DOM) tree between the targeted website and a referenced website. We then successfully used the previous OCR method to examine the phishing website if the victim is under a DNS hijacking attack. For hijacking attacks, we also provided the relevant prototype implementation scheme to determine its applicability on both desktop and mobile platforms.

The following research questions were considered:

1) What is an effective solution to mitigate hijacking attacks?
2) How can we overcome the limitation of mobile devices when we deploy our mitigation?
3) How can we compare the SSL certificate of the related legitimate website against the phishing website if the phishing website is under a DNS hijacking attack?
4) How can we compute the consistency and differentiation of DOM trees between the accessed website and the official website?

We presented two following hypotheses to DNS hijacking attack and ISP hijacking attack separately in this part:

*H2: "**A phishing website that is under a DNS hijacking attack can be detected by implementing OCR and examining the associated SSL certificate**"*

*H3: "**ISP hijacking attacks can be identified by observing the consistency and differentiation of DOM trees**"*

A paper, entitled "**A Cost-Effective OCR Implementation to Prevent Phishing on Mobile Platforms**", was accepted and published in the ***IEEE 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*** [30].

A paper, entitled "**ISP Hijacking Protection via Detecting the Consistency and Differentiation of DOM Tree**", is pending on ***IEEE 2023 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*** [1].

For Subdomain Takeover Attacks:

In this research, we investigate how to detect a potential subdomain takeover attack by an automatic approach (in this study, we focus on the CNAME threat), and we also present a method to generate a valuable dictionary for querying subdomains by using machine learning.

The following research questions were considered:

1) What is the best tool for discovering subdomains?

2) What is the limitation of the current querying subdomain approach?

3) How to generate a valuable dictionary (used in a Brute Force approach to discover subdomains) based on machine learning?

4) How to perform an automatic approach to discover subdomain takeover risks?

In this part, we derived two hypotheses to further our investigation of subdomains and automatic detection as below:

*H4: "**Subdomains can be predicted using machine learning to cover the shortage of non-dictionary word domains in the existing dictionary**"*

*H5: "**The risky subdomain can be discovered by an automatic approach without any human efforts**"*

A paper, entitled "**An Empirical Study: Automated Subdomain Takeover Threat Detection**", was accepted and published in the ***IEEE 2021 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*** [52].

1.6. Research Plan

This research began with planning an itinerary of work over the allocated time period and the relevant research and deliverables are summarised in the following Gantt Chart**,** as shown in Figure 3 below:

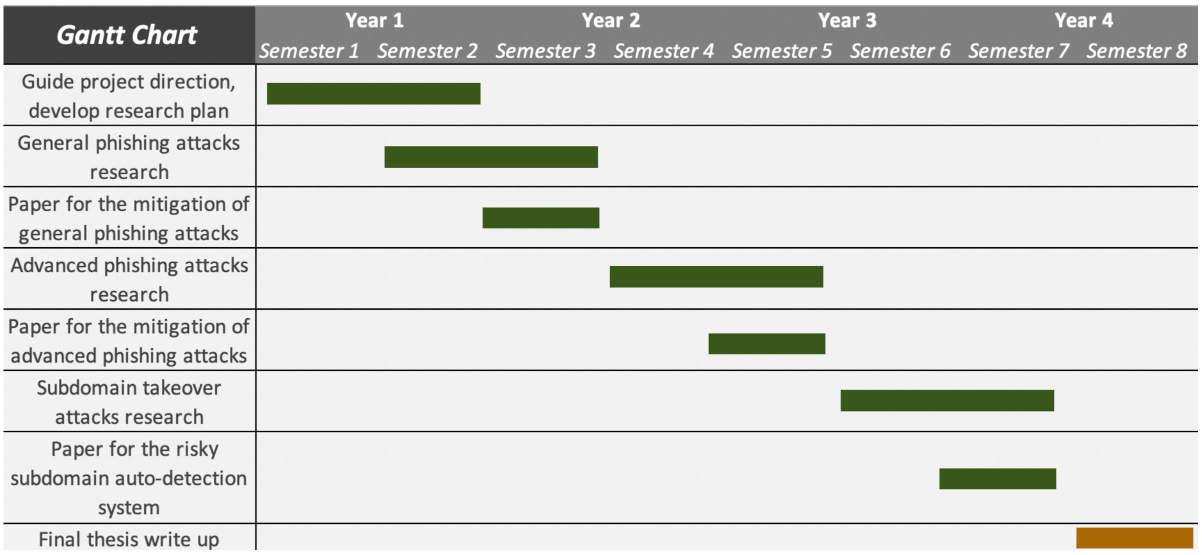


FIGURE 3: A Gantt Chart showing the research schedule of this project.

1.7. Thesis Structure

The remainder of this PhD thesis is structured as follows: In **Chapter Two**, the relevant terms used in this thesis are explained in a glossary to enable the reader to understand the terminology used. In **Chapter Three**, the existing and most up to date literature on the general field of phishing are reviewed and presented to show how the work presented here follows on from and extends existing work through automation. The next three chapters demonstrate the empirical research undertaken to advance the field of automated phishing detection in increasing attack complexity. **Chapter Four** through to **Chapter Six** demonstrated experiments undertaken on general phishing attacks, then advanced phishing attacks and, finally, subdomain takeover attacks. The three different, common types of attack are discussed specifically and in relation to the design, implementation and evaluation of the undertaken empirical work. Next, a discussion and analysis of this empirical and complex research is conducted in **Chapter Seven**. Finally, the conclusion and future work chapter complete this thesis in **Chapter Eight**.

2. TERMINOLOGY

This chapter outlines some commonly used terms in the thesis.

Malicious Payload

Typically, the term payload refers to the load carried by a vehicle, such as the passengers in a train. In computing, the payload always points to the transferred data [53]. However, in cyber security, the term malicious payload refers to the component in a malicious attack, which is adopted to execute a malicious activity to harm the target [54]. For example, in a buffer overflow attack, the malicious payload points to the malicious code which results in this buffer overflow attack.

General Data Protection Regulation (GDPR)

The General Data Protection Regulation (also called as GDPR) is a regulation in EU law on data protection and privacy in the European Union (EU) and the European Economic Area (EEA) [55]. Even the transfer of person data outside the EU and EEA areas is also covered by this policy. The full details of this policy can be seen in [56].

HTTP

HTTP stands for Hypertext Transfer Protocol, which is an application layer protocol for distributed collaborative, hypermedia information systems [57]. HTTP is a client-server protocol that allows the fetching of resources, such as HTML (Hyper Text Markup Language) documents. The communication between client and server is an exchange of the individual messages. The message sent from the client side (usually a Web browser), is called a request, and the message sent from the server side, is called a response [58]. Under an HTTP connection, the transferred data is unencrypted plain text. In a web browser, the URL bar displays a "Not Secure" symbol if the connection is based on HTTP, as shown in Figure 4 below.

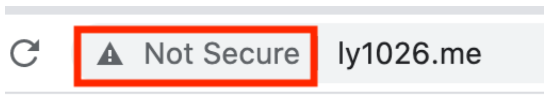


FIGURE 4: The URL bar displays a "Not Secure" symbol if the connection is based on HTTP.

HTTPS

The term HTTPS is an updated version of HTTP, and stands for the Hypertext Transfer Protocol Secure. It is used to overcome the issue of unencrypted plain text using Transport Layer Security (TLS) during the communication over a computer network. Therefore, this protocol is also referred as HTTP over TLS or HTTP over SSL between a web browser and a website [59] [60]. Under the HTTPS connection, the transferred data is encrypted, and any hijacked data through a Man in the Middle (MITM) attack is unreadable. In a web browser, the URL bar displays a "lock" symbol if the connection is based on HTTPS, as shown in Figure 5 below.

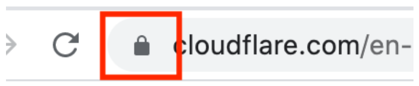


FIGURE 5: The URL bar displays a "lock" symbol if the connection is based on HTTPS.

URL

A URL (Uniform Resource Locator) is a unique identifier used to index an expected resource on the Internet, it is also referred to as a web address [61]. A URL consists of various parts, some parts are mandatory, others are optional, such as scheme, domain name, port, path and parameters. The structure of URL is illustrated in Figure 6 below.

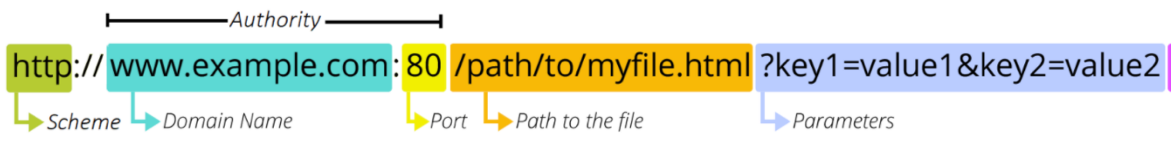

Authority
http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2
Scheme
Domain Name
Port
Path to the file
Parameters

FIGURE 6: The illustration of URL component **[62]**.

Short URL

The shortening of a URL is a technique that is used to make URLs shorter and simpler on the World Wide Web [63]. For example, the target website, available at: *https://www.st-andrews.ac.uk*, is also accessible by a shortURL; the shortened result is shown in Figure 7 below. More literature detail about short URLs please see Chapter 2. Various short URL generation tools online are, such as shorturl[1], tinyurl[2] , and bitly[3].

[1] shorturl: online tool for generating short URL, available at: *https://www.shorturl.at/*.
[2] tinyurl: online tool for generating short URL, available at: *https://tinyurl.com/app*.
[3] bitly: online tool for generating shot URL, available at: *https://bitly.com/*.

Long URL: https://www.st-andrews.ac.uk

FIGURE 7: The shortened URL result of *https://www.st-andrews.ac.uk* put through the online short URL tool *https://www.shorturl.at/*.

MITM Attacks

The man-in-the-middle attack is often referred to as MITM. In computer security, this attack always happens in the communication between two parties, resulting in each party believing that they are directly communicating with each other [64]. In fact, the traffic from each party will go through this attacker before arriving at the destination, and an illustration is shown in Figure 8 below. Attacks adopt MITM attacks to steal credentials or personal information [65]. In this thesis, the implementation of several experiments uses this technique to achieve the expected purpose, and we adopt software called mitmproxy, which is developed in python, to execute the data interruption. More experimental details can be seen in further chapters.

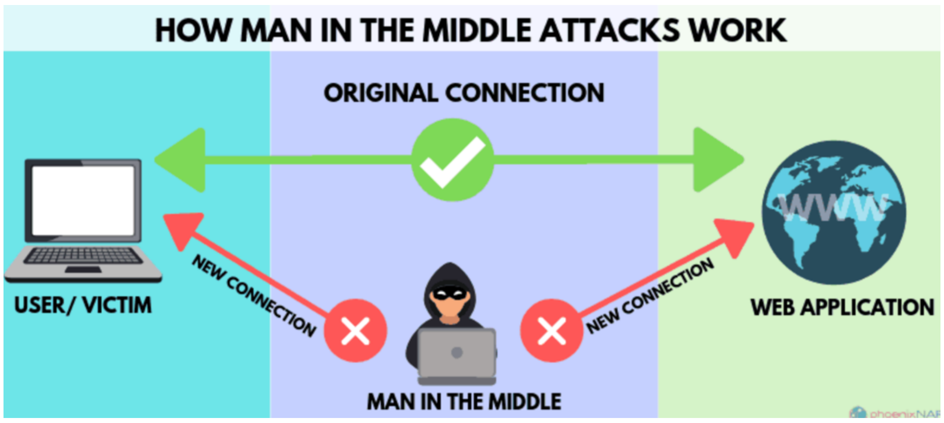


FIGURE 8: The illustration of how a Man in the Middle Attack works **[66]**.

DOM Tree

When a webpage is loaded, the HTML file will be parsed. A Document Object Model (DOM) of this page will be generated by the browser. The DOM model is constructed as a tree of Objects [67], and an example of a DOM tree is shown in Figure 9 below.

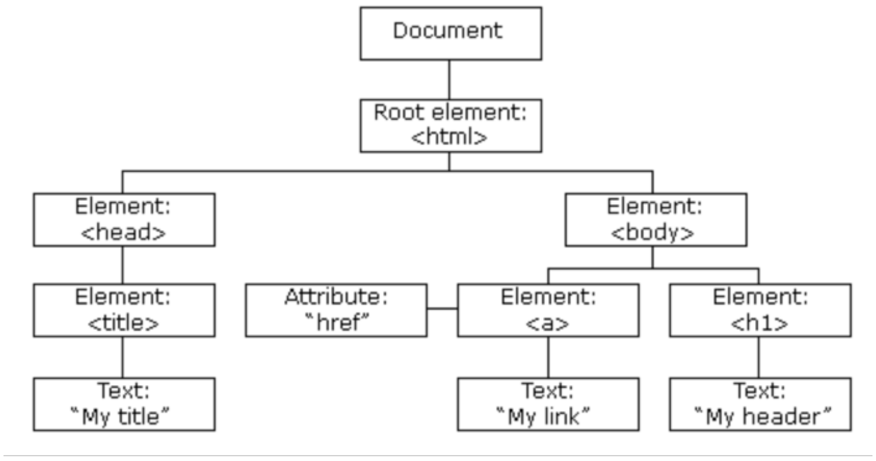


FIGURE 9: An example of the HTML DOM tree.

A Record

An A record maps a domain name to an (IPv4) IP address that works for computer hosting the domain. An A record uses a domain name to locate the IP address of a computer connected to the internet [68]. For example, in order to access the St Andrews website, the user types *www.st-andrews.ac.uk*. In their server, an A record is configured that maps to an IP address 138.251.7.84. This means that a request from the user's browser to *www.st-andrews.ac.uk* is directed to the targeted server with IP address 138.251.7.84.

Thus, the A record stands for IP address and this is the most fundamental type of a DNS record [69]. In their DNS server, the configuration of an A record is shown in Table 3 below:

st.andrews.ac.uk	record type	value
@	A	138.251.7.84

TABLE 3: An example of A record configuration in its server.

CNAME Record

A Canonical Name (CNAME) Record is used in the Domain Name System (DNS) to create an alias from one domain name to another domain name [70]. A common example is to use CNAME to alias *example.com* to *www.example.com* [71]. For instance, the accessed results are the same between *www.st-andrews.ac.uk* and *st-andrews.ac.uk*, because a CNAME record is configured in the St Andrews server. The configuration of a CNAME record in the St Andrews server is as shown in Table 4 below:

st.andrews.ac.uk	record type	value
@	CNAME	*www.st-andrews.ac.uk*

TABLE 4: An example of CNAME record configuration in its server.

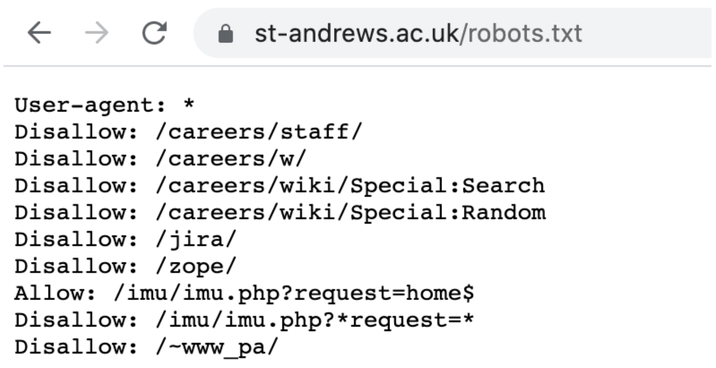
Therefore, the A record points a domain name to a specific IP address, and the CNAME record aliases a domain name to another domain name instead of to an IP address [72].

DNS Zone Transfer

DNS (Domain Name System) is like an Internet phonebook; it is used to resolve the target website from human-readable domain names into a computer-readable IP address. The system involves authoritative DNS servers and DNS caches, where, the former provides the relevant DNS mapping information and the latter is responsible for storing that information temporarily for client lookups [73]. DNS zone transfer uses the AXFR protocol to replicate DNS records across DNS servers, copying DNS information from one DNS server to other DNS servers [74].

Robots.txt

A robots.txt file is a text file without HTML markup code on the website and this file is located on the web server the same as any other files on the website [75]. Moreover, a robots.txt file is responsible for telling search engine crawlers which URLs are accessible on this website through the crawler. Typically, this is adopted to avoid overloading a website with requests [76]. For example, the path of a robots.txt file in the *st-andrews.ac.uk* website is shown in Figure 10 below:


st-andrews.ac.uk/robots.txt

```
User-agent: *
Disallow: /careers/staff/
Disallow: /careers/w/
Disallow: /careers/wiki/Special:Search
Disallow: /careers/wiki/Special:Random
Disallow: /jira/
Disallow: /zope/
Allow: /imu/imu.php?request=home$
Disallow: /imu/imu.php?*request=*
Disallow: /~www_pa/
```

FIGURE 10: The accessed result of robots.txt in the University of St Andrews official website.

3. A LITERATURE REVIEW ON PHISHING

In this chapter, the relevant literature about the prevention and identification of phishing attacks were thoroughly researched and reviewed. We started investigating the workflow of a phishing attack before focussing on the technical vectors of how victims fall prey to an attack. We then present a section on current phishing attack threats. Finally, existing research about phishing attacks and different prevention approaches were reviewed.

3.1. The Workflow of Phishing Attacks

The most common approach for launching a phishing attack is to send a phishing email, almost 90% of cyber-attacks are initiated via phishing emails [77]. The phishing email often attaches an external link that redirects victims to a spoofed website and requires victim to enter their personal information [25]. The workflow of a phishing attack was illustrated by Jain and Gupta [23], as shown in Figure 11 below. In this case, the phishing website is an illegal clone of a legitimate website and has a high visual similarity. These phishing sites always involve some input fields, such as a text box, to require victims to type in their sensitive information. The information is sent to the phisher once the victims submit their details. The complete procedure about how phisher launch a phishing attack is divided into six steps.

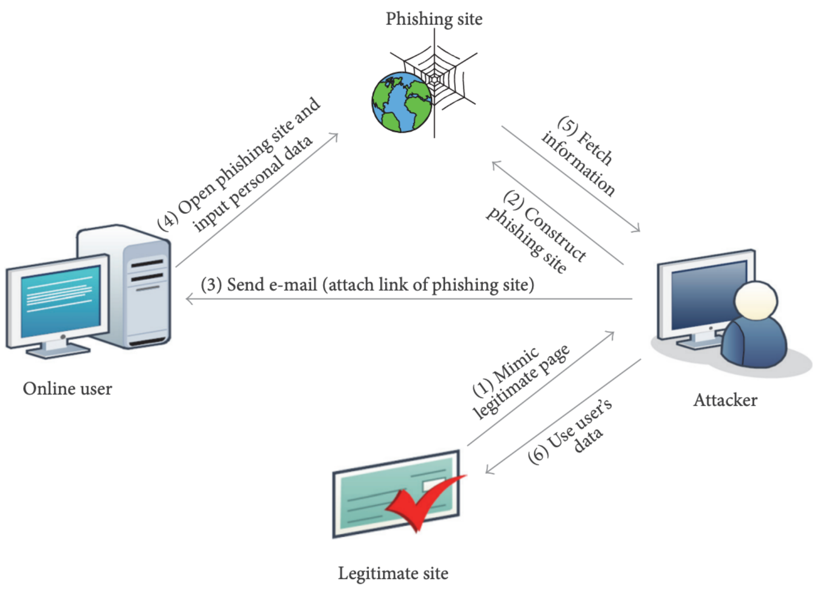

Phishing site
(4) Open phishing site and input personal data
(5) Fetch information
(2) Construct phishing site
(3) Send e-mail (attach link of phishing site)
Online user
Attacker
(1) Mimic legitimate page
(6) Use user's data
Legitimate site

FIGURE 11: The illustration about how phisher launch a phishing attack **[23]**.

Step 1: Phishers gather and duplicate the data from a well-known legitimate (targeted) website.

Step 2: Phishers develop their own phishing website according to the collected data from step 1 above.

Step 3: Phishers send this malicious website to multiple victims via email, usually attached with a link within the email text.

Step 4: Victims click the malicious link within the email, access the phishing website, input and submit their personal information.

Step 5: The victims' data is sent to the phisher from the phishing website.

Step 6: In the end, phishers access the targeted website using the victim's identification information.

3.2. How Victims Fall for a Phishing Attack

Phishing has been studied since the early 2000s, but the problem has not been totally solved as phishers constantly evolve their attack approach [77]. A phishing website is relatively easy to identify through observing the URL path, and a careful and experienced user is now knowledgeable about these suspicious websites [28] [29] [30].

A phisher fools the victims fall into a phishing attack by impersonating a visually similar website, via the following tactics [23]:

1. **Visual Appearance**. The phishing website has a high visual similarity to the legitimate website. This is because phishers duplicate the HTML code from the legitimate website to develop their phishing website.
2. **Address Bar**. Phishers cover the URL address by using a script or image, resulting in the victim believing they are on the correct website.
3. **Embedded Objects**. Phishers exploit embedded objects, such as images, scripts etc, to hide their malicious textual content and HTML code.
4. **Favicon Similarity**. Favicon is an image icon associated with a particular website. Phishers may duplicate the favicon of a target website resulting in the victim trusting that they are on the right website.

An experiment conducted by Dhamija et al, of Harvard University and the University of California Berkeley, considered whether individuals could identify a phishing website [78]. They revealed that around 90% of participants were unable to identify phishing websites, and even an experienced user could also be fooled due to the high visual appearance of a phishing website. Additionally, almost 23% participants did not examine the URL bar when they accessed the website in this experiment.

Also, an individual lacking security awareness is the primary reason that they fall into a phishing attack [77]. Volkamer et al summarised five main reasons resulting in users falling into a phishing attack [79]:

- Users lack detailed knowledge about the (true) URL.
- Users do not know which web page is to be trusted.
- Users do not check the complete URL address due to page redirection or hidden URLs.
- Users do not have much time to consult URLs, or they accidentally enter some web pages
- Users cannot determine phishing web pages from the legitimate ones.

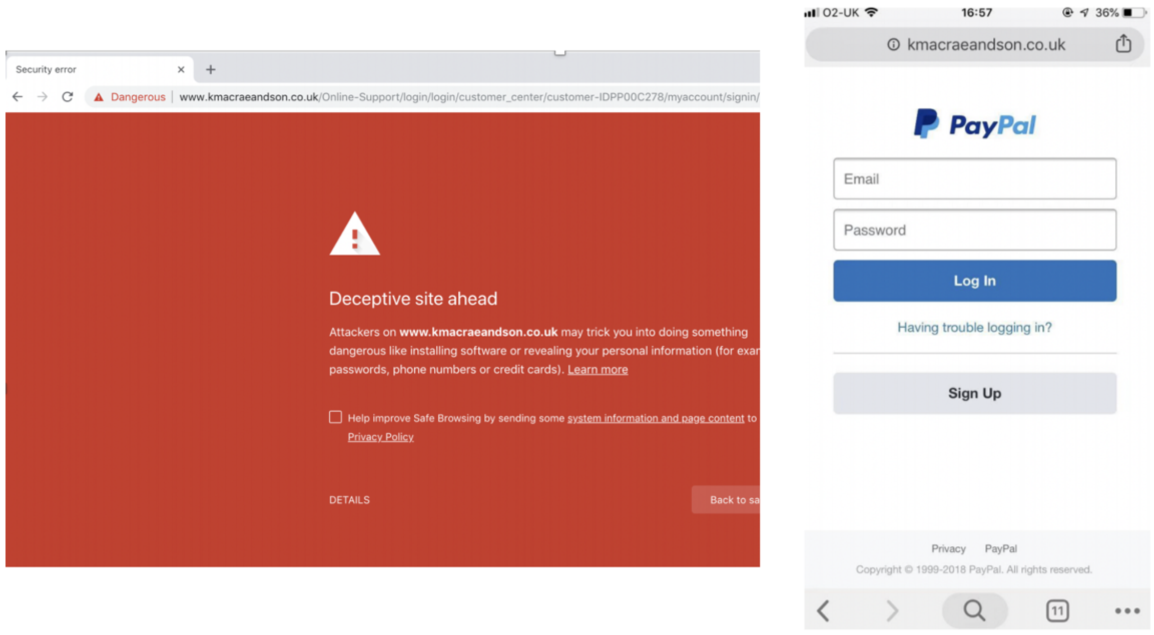
3.3. Present Threats on Phishing

Currently phishing attacks are more sophisticated as phishers evolve their attack approach through various techniques. For example, a general phishing attack can be relatively easy to identify through observing an URL path, and many users are now knowledgeable about this kind of attack from suspicious URL addresses or obvious warning information from browsers [28] [29] [30]. However, these kinds of malicious URL address can be hidden due to various advanced technologies, such as the use of a shortened URL or QR code.

Shortened URLs were created to reduce the length of URL addresses as much as possible, to make them tidy and easy to quickly type in a web address [80]. Shortened URLs have now been used for well over a decade, the first-time a shortener service (such as TinyURL) appeared in public can be traced back to 2002 [18]. Subsequently, shortened URLs have grown in popularity with the emergence of Twitter in 2006 [80]. At that time, Twitter limited a user post to 140 characters. If the users wanted to attach a link in their post, most of the characters would have been taken up by this URL. So, shortened URLs were used as a work-around. TinyURL was the first choice of Twitter for URL shortening until Bitly came along [80]. Shortened URLs have many advantages, but they still face a large security challenge. For the users, it is difficult to determine where the web browser will actually take them. Most criminals exploit this approach to direct victims to access their phishing site or to download their malicious software [81]. There are many examples of this, such as in 2010, malware was spread by using shortened URLs to spam instant messaging services. In 2012, a skype worm spread quickly through the use of shortened URLs [18]. Currently the usage of shortened URLs is rare, as they have been mostly replaced by QR codes.

In Krombholz et al [82], the emerging technology of QR codes was shown to be a phishing attack vector as they are cheap to produce and easy to deploy. Either the phisher replaces the entire QR code or the phisher modifies a few pixels of a QR code to be used as the attack vector. These encoded malicious QR codes redirect the user to a phishing scam. Kharraz et al [83] revealed that the majority of malicious QR codes redirects the victim to a phishing website or an exploited website for stealing their sensitive data, rather than to download malware. As shown by Sharma [84], the first malicious case using QR codes was detected in September 2011.

Furthermore, the prevention of phishing attacks on mobile platforms is more complicated than expected. Along with the same problems as desktop computers face, mobiles still face additional challenges. According to our summary, the majority of phishing links come from phishing emails, and mobile platforms do not support secure identification [85]. The mobile user is unable to know whether the accessed URL address is safe, and in particular if the user lacks enough security awareness. For example, Google Chrome provides much better security mechanisms against phishing attacks then other web browsers [48]. It prints a warning page that shows that the accessed URL contains a potential risk to the users on their browser if this URL is malicious, but the Google Chrome browser does not provide the same security level on mobile platforms. As shown in Figure 12 below, a suspicious PayPal website was accessed with the Google Chrome browser on both mobile and desktop platforms, and the accessed results are different. The Google Chrome browser on mobile platforms ignores the potential risk and assesses the connection as benign. We performed a comparative experiment to test browser responses and the Google Chrome browser on IOS mobile systems is relatively poor, with no phishing attack or a risky connection URL being detected. In addition, the user may still fall victim to a scam even if they check the accessed URL. For example, some phishers exploit the JavaScript *scrollTo()* function to hide the original URL bar on the page, which is replaced by a fake address bar within the page, and which is below the real address bar to confuse the users [86]. Although this problem has been fixed recently, it still exists in older versions of the platform.


Security error

Dangerous | www.kmacraeandson.co.uk/Online-Support/login/login/customer_center/customer-IDPP00C278/myaccount/signin/

Deceptive site ahead

Attackers on **www.kmacraeandson.co.uk** may trick you into doing something dangerous like installing software or revealing your personal information (for exar passwords, phone numbers or credit cards). Learn more

☐ Help improve Safe Browsing by sending some system information and page content to Privacy Policy

DETAILS

Back to sa

O2-UK 16:57 36%

kmacraeandson.co.uk

PayPal

Email

Password

Log In

Having trouble logging in?

Sign Up

Privacy PayPal

Copyright © 1999-2018 PayPal. All rights reserved.

FIGURE 12: The comparison of accessing a malicious PayPal link with Google Chrome on both desktop platforms (left) and mobile platform (right) **[30]**.

Secondly, there are many phishing risks from QR codes on mobile platforms. As a QR code is encoded, it not only is unreadable to humans, but it also requires a relevant QR reader to parse the encoded information. Thus, a potential risk such as a buffer overflow or command injection, is possible though executing the manipulated QR code [87] [88]. Moreover, QR scanners, which are used to parse the encode QR code, have had reported issues with security which is not as good as expected. Krombholz et al [89] showed that the majority of QR code scanners are unable to detect the security of a QR code, either the URL or the website content. Yao and Shin [90] conducted a comparative experiment; 31 QR scanner applications were compared, with only two apps having a security warning feature, albeit with a higher ratio of false negative errors. Even more to the point, two better open-source APIs (Google Safe Browsing and PhishTank) were recommended by the authors to improve the accuracy of phishing attack prevention, but the static limitation has not yet been overcome.

3.4. Existing Research on Mitigating Phishing Attacks

The relevant current prevention and detection strategies are presented next. Mark [91] believed the detection of phishing attacks is a challenging problem, as this attack exploits the weaknesses in human characteristics, rather than network flaws. Researchers have focused on two strategies to mitigate phishing attacks; either improving the performance of phishing detection technology or developing human education [92]. Generally, phishing detection technology has focused on identifying malicious websites, and there are two main approaches adopted to mitigate phishing attacks: a list-based and a content based approach [48] [93] [94]. A list-based scheme involves two kinds of list, a whitelist and a blacklist [29]. This scheme is a static approach wherein the target URL is compared to the phishing lists before accessing the URL. In a whitelist approach, the legitimate domains are stored in the list, and any matched result shows the target website is a legitimate website, and therefore the user can access this website safely. In a blacklist approach, the malicious domains are collected in the list, and any matched result shows the target website is likely a phishing website, and a warning notification will be sent to the user on their browser. A content-based scheme detects phishing attacks through extracting features from the target URL [95]. Various machine learning approaches are derived from this approach, such as using lexical and host-based analysis to categorize the features. For example, in some research [51] [82] [96], the textual properties in the URL are used for lexical analysis. For host-based analysis, the server properties (IP address, registration information etc) are investigated from open sources, such as WHOIS[4].

Nevertheless, both approaches are still not good enough to mitigate all phishing attacks due to their limitations. In the list-based approach, especially in the blacklist scheme, a zero-day

[4] WHOIS: Finds information on any domain name or website, available at: *https://who.is/*.

phishing attack cannot be detected as the blacklist-based scheme is not dynamic [93], and these lists require to be constantly updated. A site may still steal user credentials if this is a newly created phishing website not already included in the blacklist. Three well-known phishing blacklists are operated by Microsoft, Google and PhishTank [49]. Moreover, a blacklist approach usually needs enormous human effort to manually verify suspicious websites as phishing, and thus the reported phishing websites are slow, as human verification requires much time. From the posted statistics by Sheng et al [97], an estimated 47% - 83% of phishing URLs appeared on blacklists about 12 hours after they were launched by a phisher.

In the content-based approach, the accuracy of the analysis is subject to the integrity of extracted features from the target website. For instance, gathering server related information, such as IP address, registration information etc, may become difficult due to possible restrictions in the future. WHOIS has continually sparked controversy as it generates too much private information. In May 2018, the BBC stated that most of this private information have been wiped from WHOIS in order to comply with the EU's General Data Protection Regulation (GDPR) legislation [98]. Many feature-based algorithms and approaches have been developed to automatically detect phishing websites. However, these approaches are prone to false positives (incorrectly verifying phishing sites as a legitimate website) as well as false negatives (incorrectly verifying legitimate sites as phishing website) [49] [99] [100].

3.4.1. Relevant reviews on the list-based approach

Currently, various studies are based on these two approaches. In [99], Liu et al believed a promising solution is to improve the detection performance through merging manually verified blacklists with computational techniques, not only to improve the detection accuracy, but to also reduce the time consumption for verifying attacks. This solution would benefit the sites that support phishing blacklists services, such as PhishTank. The combination technique was considered from four major approaches; making the problem invisible to end users, improving the design of user interfaces, improve end-users training and leveraging the crowds. They designed a system similar to crowdsourcing to identify phishing websites. The system extracted unverified URLs from PhishTank and filtered the URLs which were not on a whitelist.

They deployed relevant algorithms (DBSCAN and shingling) to cluster similar pages in order to increase further detection efficiency. These clusters were submitted to Amazon's Mechanical Turk for verification by participants. In the end, the URL was identified to be phishing or general according to the vote weight from each participant. Their system was faster than other existing blacklists. Also their solution can be easily adopted by any existing manually-verified blacklist such as those offered by Google, Microsoft and PhishTank.

Xiang et al [101] revealed that a large number of current phishing websites are created with tools, such as toolkits, which result in a high content similarity. Thus, they proposed a hierarchical blacklist enhanced method to identify phishing attacks through exploiting existing backlists and applying fuzzy matching techniques in a probabilistic fashion. in their approach, an n-grams technique was deployed on manually verified URL blacklists to analyse phishing webpage content, and a single near-duplicated phishing webpage was identified in a probabilistic fashion via shingling technique [102]. Moreover, in order to reduce the false positives, they used a filter module, which further determined the legitimacy of a potential phishing website via information retrieval techniques within query search engines.

The list-based approach does not index the target URL only. Geng et al [103] believed that most current websites consist of multiple resources, such as CSS, JS and image. However, these legitimate (brand) websites usually get resource content from another domain due to the browser limitations of maximum concurrent connections to the same domain. For example, the legitimate website PayPal gets its CSS, JS and image files resources from *paypalobjects.com*. So mining the resources request relation is an effective method to prevent emerging phishing websites. Geng et al proposed a novel approach that is based on mining brand resource request relations to detect phishing attacks. Not only is their approach efficient and easy to implement, but it is also an effective complement to existing methods. In their solution, the target URL and all queried domains will be checked from both a blacklist and whitelist.

In a PhD thesis from the Royal Holloway University, Bell [104] believed that Twitter is an attractive target for cybercriminals as there are more than 500 million daily tweets from over 330 million active users. Thus Bell focused on the protection of twitter users and explored how well protected Twitter users are from phishing and malware attacks. The investigation began from three popular phishing and malware blacklists, Google Safe Browsing (GSB), PhishTank (PT) and OpenPhish (OP). This study focused on the measurements of empirical data, such as blacklist delay times, number of URLs appearing in blacklists, number of users exposed to blacklisted URLs, etc. In this study, over 182 million URL-containing public tweets gathered from Twitter's Stream API were analysed. The experiment findings revealed that as time increased, fewer URLs remained in blacklists. The OP blacklist removed a significant volume of URLs from its dataset after a duration of 5 and 7 days, and no URLs were listed in OP for more than 21 days. Moreover, most malicious URLs were detected by the GSB blacklist within 6 hours of being tweeted, but there were still a lot of suspicious URLs that required at least 20 days to index in GSB.

A research paper from the RWTH Aachen University, Drury, Lux, and Meyer [105] conducted an investigation about the life cycle of phishing attacks by analyzing the creation process of phishing websites. Four dating methods were deployed to determine dates of interest in the creation process of phishing, 1) crt.sh (it is a certificate transparency log monitor that includes past certificates for a given domain name); 2) direct download of server certificate; 3) WHOIS; and 4) carbon (it is an open source tool that is implemented to check all images on a website, including external resources and return the earliest date from any image). According to their findings, the certificates and WHOIS information might be used to detect phishing websites up to several days earlier, which is subject to the type of phishing campaign though. Other types of campaigns, such as launching attacks using hosting providers or the largest clusters, might still be detectable with predictive blocklists. Moreover, resource sharing among malicious phishing website seems to be common behaviour, and it might be exploited to cluster attacks. So, the knowns malicious patterns of phishing domains could be used to predict and block the additional websites in the phishing campaigns for several months after the occurrence of the first time.

3.4.2. Relevant reviews on the content-based approach

In [29], Sahingoz, et al, developed a learning-based phishing detection system, as they thought detection of phishing threats is a simple classification issue using a machine learning method. In their approach, the training data includes many features, which are related to both phishing and legitimate website classes. Forty different NLP based features were selected, such as raw word count, brand check for domain, longest word length, shortest word length, known TLD etc. Also, seven different algorithms (Decision Tree, Adaboost, K-star, KNN[n=3], Random Forest, SMO and Naïve Bayes) were conducted during the training process in order to select the optimal algorithm which has the highest accuracy. According to the findings of this approach, the Random Forest algorithm gave the best performance with the highest detection accuracy rate among these seven algorithms for the detection of phishing websites. However, in order to satisfy the situation closest to a real-world scenario for phishing detection, subsequent extensive research had to be conducted by Serrano et al.

In [28], Serrano et al, presented a representative machine learning based approach to identify phishing websites, named as a Phishing Index Login URL (PILU-60K). In this research, the experimental dataset has been improved to close the real-world scenario. Phishing datasets usually contain a landing page similar to a legitimate and well-known website, but most of them are without the login forms. Also, most phishing websites can be identified through observing the HTTP protocol. The APWG [106] revealed that phishing websites using the HTTPS protocol have undergone a significant increase, from around 10% in 2017, up to 84% in 2020 [107]. For that reason, the performance of detecting phishing using old HTTP training data will be affected with the improvement of phisher strategies. Thus, in this work, a dataset with URLs of both index pages and login pages was collected. In total 60K URLs, with 20K samples each of legitimate index page URLs, legitimate login page URLs and phishing page URLs. The phishing URLs were gathered from PhishTank, between November 2019 and January 2020.

In [108], Zuhair, Selamat and Sallehs considered that most of the current machine learning based phishing detections have several disadvantages in leveraging features that are continually exploited by the phishers. They investigated the previous deployed features for phishing detection and categorized them into three parts; webpage content features, URL features and online features. Also, the specific features in each part were summarised. For example, in the webpage content features, there are images, logos, WHOIS data and DOM tree components etc. Under URL features, there are IP addresses, hostname lengths and dots count etc. Under online features, there are certificate states, website ranking in search engine results, URL address history and domain history etc. Therefore, they presented a hybrid features based machine learning approach to identify phishing attacks. Two different features categories were designed that involved fifty-eight features in total. The first feature category included forty-eight features that are mostly cross-site scripting and embedded objects, the second feature category contained ten features that are extracted URL features. These features were trained and tested using the SVM classifier to evaluate their prediction susceptibility with respect to their classification performance against the phishing website.

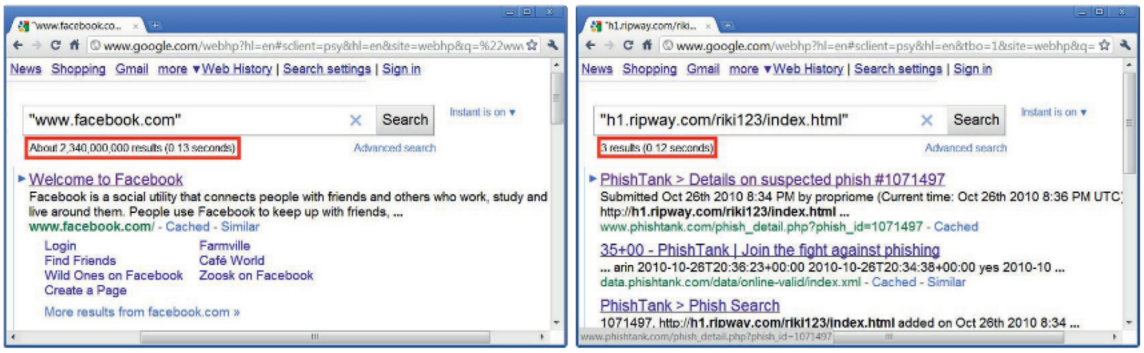
Xiang and Hong [109] proposed a novel hybrid detection approach based on both information extraction and information retrieval techniques. Two components are involved in their method. The identity-based component was implemented to recognize the identity of a web page using an information retrieval technique. Subsequently, the keywords-retrieval component utilizes an information retrieval algorithm exploiting the power of search engines to identify phishing, and named entity recognition (NER) algorithms are used to reduce false positives. In the end, they deployed a domain whitelist and a login form detector to filter good web pages.

In a paper from New Mexico Tech, Basnet, Sung and Liu [110] proposed a rule-based approach to detect phishing websites. Fifteen rules were proposed according to various factors, such as Search Engine-based Rules, Red Flagged Keyword-based Rule, Obfuscation-based Rules, Blacklist-based Rule, Reputation-based Rule and Content-based Rules, A rule is usually written in the form of *IF* conditions *THEN* actions. For example, from the search engine-based factor, IF a webpage's URL is not present in all search engines‟ indexes, THEN the webpage is potentially phishing. These rules were then used as features in Decision Tree and Logistic Regression learning algorithms to identify the phishing webpages. In comparison, we studied logos and subdomains as a primarily efficient way of detecting phishing without recourse to decision trees.

Subsequently, Basnet, Sung and Liu published another paper on the mitigation of phishing attacks [111], they evaluated the correlation-based and wrapper-type feature selection techniques. They applied 177 initial features, of which 38 features were extracted from content-based data, and the rest features are extracted from URL-based data. The experiments showed that wrapper-based technique improved classifier accuracies significantly compared to the correlation-based technique, although the implementation process was slower.

In a PhD thesis from University of Ottawa, Cui [112] researched the phishing attacks from a new perspective, not only proposed methods to mitigate phishing attacks, but only focus on exploring the causes and motivation behind the attacks. The proposed detection mechanism consists of two categories, the client-side of phishing detection and the server-side of phishing detection. they proposed a clustering-based approach for the client-side of phishing detection, as the experiments shown that a common behaviour of phishers is to repeatedly launch their attacks with using different domain names and hosts, their clustering method can effectively detect these minor modifications. For the server-side of phishing detection, Cui proposed a machine learning classifier that is able to identify the specific patterns of outgoing traffic that is phishing.

Huh and Kim [113] proposed a novel heuristic-based approach to identify phishing attacks by detecting the search results returned from popular web search engines, such as Google, Bing and Yahoo. Different to other machine learning based approaches, their approach results in detection by analysing the “number of results” returned and the “ranking” of the website from the popular web search engines, rather than other web and URL features. According to the findings of their experiment, there are significant differences about the web reputation (number of results and ranking) between a legitimate website and a phishing website. For example, a comparison between the number of results from a legitimate website and a phishing website is shown in Figure 13 below. Moreover, they conducted an experiment to observe the reputation changes on Google between a newly legitimate website and a phishing website. They found that the number of results for legitimate websites increased quickly, and the ranking is in the top position since the second day. In contrast, the phishing websites had a smaller result and were never ranked. They experimented with four well-known classification algorithms (Linear Discriminant Analysis, Naïve Bayesian, K-Nearest Neighbour, and Support Vector Machine) to demonstrate the effectiveness of their approach, the K-Nearest Neighbour algorithm performed best.


News Shopping Gmail more ▾Web History | Search settings | Sign in
Search
Instant is on ▾
Advanced search

"www.facebook.com"

About 2,340,000,000 results (0.13 seconds)

Welcome to Facebook
Facebook is a social utility that connects people with friends and others who work, study and live around them. People use Facebook to keep up with friends, ...
www.facebook.com/ - Cached - Similar
Login | Farmville
Find Friends | Café World
Wild Ones on Facebook | Zoosk on Facebook
Create a Page
More results from facebook.com »

"h1.ripway.com/riki123/index.html"

3 results (0.12 seconds)

PhishTank > Details on suspected phish #1071497
Submitted Oct 26th 2010 8:34 PM by propriome (Current time: Oct 26th 2010 8:36 PM UTC)
http://h1.ripway.com/riki123/index.html ...
www.phishtank.com/phish_detail.php?phish_id=1071497 - Cached

35+00 - PhishTank | Join the fight against phishing
... arin 2010-10-26T20:36:23+00:00 2010-10-26T20:34:38+00:00 yes 2010-10 ...
data.phishtank.com/data/online-valid/index.xml - Cached - Similar

PhishTank > Phish Search
1071497, http://h1.ripway.com/riki123/index.html added on Oct 26th 2010 8:34 ...

(a) Facebook (b) A phishing website

FIGURE 13: A comparison about the number of results between a legitimate website and a phishing website **[113]**.

In a Master thesis from University of Colombo, Botejue [114] explored the performance of various machine learning algorithms on phishing detection issues. Eight different algorithms were examined, there are as XG Boost Classifier, Logistic Regression, Random Forest Classifiers, Decision Tree Classifier, K Neighbours Classifier, Multilayer Perceptron (MLPs): Deep Learning, Autoencoder Neural Network and Support Vector Machines. According to the experimental findings, XG Boost algorithm provides the best classification performance compared to other algorithms.

Phishers always attempt to impersonate well-known legitimate websites and the architecture of a webpage and its impersonation is expected to be similar. Thus, there are various phishing detection approaches based on the comparison of DOM trees between the suspicious website and a legitimate website [23]. For example, in [115], Rosiello et al proposed a DOM tree-based approach to identify the phishing activity by detecting the reuse of the same information on multiple websites. If a user reuses the same information, such as same username, password etc, on multiple websites, their system will generate a warning and conduct a further detection. The system compares the structure of the DOM tree of the first website where the data was initially entered and the accessing website where the data is reused. If the DOM tree between these two websites is found to be similar, the accessing websites is identified as a phishing attack. A list of previously accessed websites and the entered information needs to be pre stored and maintained for comparison with the current website information.

Xiang et al [116] proposed a comprehensive approach that exploit the DOM, search engines and third-party services with machine learning to detect the phishing attacks. In their approach, in order to achieve runtime speedup and reduce human effort, two mechanisms were designed to filter the input webpages during the testing process. Firstly, the hash-based filter was used to find the web pages that have no matching results with existing phishing attacks via comparing the similarity. Secondly, the login form filter was deployed to detect whether the analysed web pages used a login form in the HTML. In the training process, various features are used; there are URL-based features (including domain, IP address, Number of dots in URL, Suspicious URL, Number of sensitive words in URL and TLD), HTML-based features (including Bad forms, Bad action fields, Nonmatching URLs and brand name) and web-based features (involving age of domain, page in search results, PageRank, etc). They implemented six learning algorithms to evaluate the effectiveness of their feature set, including Support Vector Machines (SVM), Logistic Regression (LR), Bayesian Network (BN), J48 Decision Tree, Random Forest (RF) and Adaboost. According to their findings, Bayesian Networks returned the best result.

Subsequently, Xiang [49] proposed a cascaded approach to classifying phishing websites, via implementing different types of features into multiple stages of a single cascade. The architecture of this approach is shown in Figure 14 below. Each stage consists of a machine learning classifier using a different subset of features. Thirteen features are selected in total, from URL-based, HTML-based and Web-based separately. A majority of legitimate websites will be classified correctly and quickly in the first stage as the lightweight features (fast features, such as URL-based features, HTML-based features) are used in early stages of the cascade. Most phishing websites entered the final stage for the classification, where the slowest features (web-based features, such as age of domain, page in top search results) are extracted. Thus, his approach has a better runtime performance.

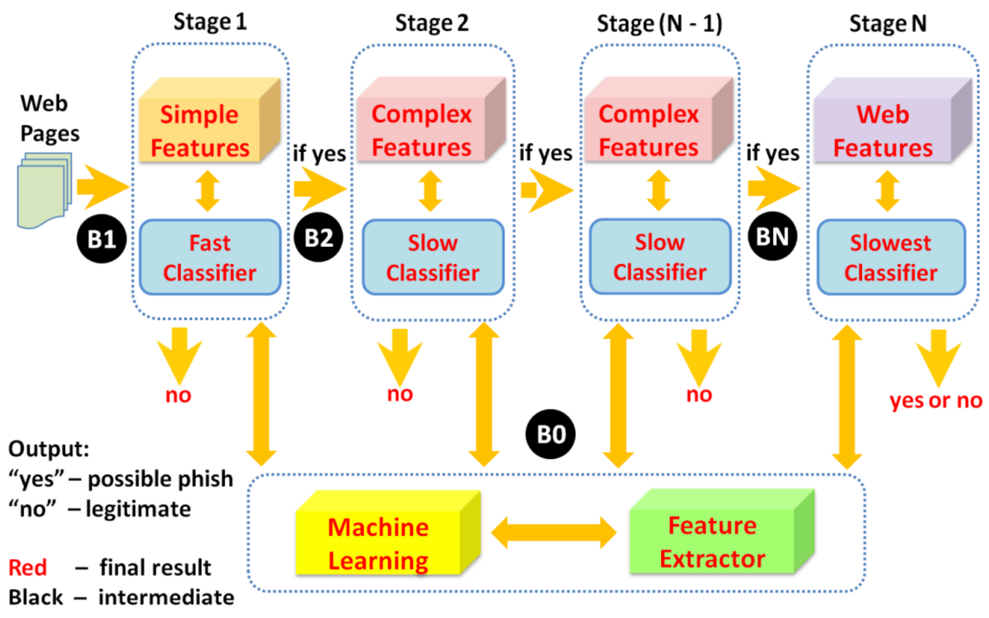


FIGURE 14: The architecture diagram of the cascaded learning framework for phish detection **[49]**.

Moreover, some researchers consider the detection of phishing web pages as clustering problems [117]. For example, in [118], Feng, Zhang and Qiao proposed a novel DOM based approach to detect phishing attacks from the perspective of clustering via comparing the structure of web pages, as they thought the clustering of web pages would be different between a benign website and its related phishing website. An example illustration is shown in Figure 15 below. In order to better represent the overall web page and semantic information in each HTML tag, the DOM tree is represented as a vector form using the Doc2Vec model. In their approach, the web page is parsed to enable the construction of the DOM tree and only the tag name (node name) is gathered, so that the attributes and content in each tag are discarded. This obtained DOM tree is converted to a vector form using Doc2Vec mode, and the semantic similarity in web pages is measured by the distance between different DOM vectors. Finally, clustering is performed on the similarity, and the category of the web page to be tested is determined by labelling the clustered classes. According to their findings, the experimental accuracy of this kind of clustering method is not as good as the classification method, but this unsupervised learning method is more in line with the human cognitive model.

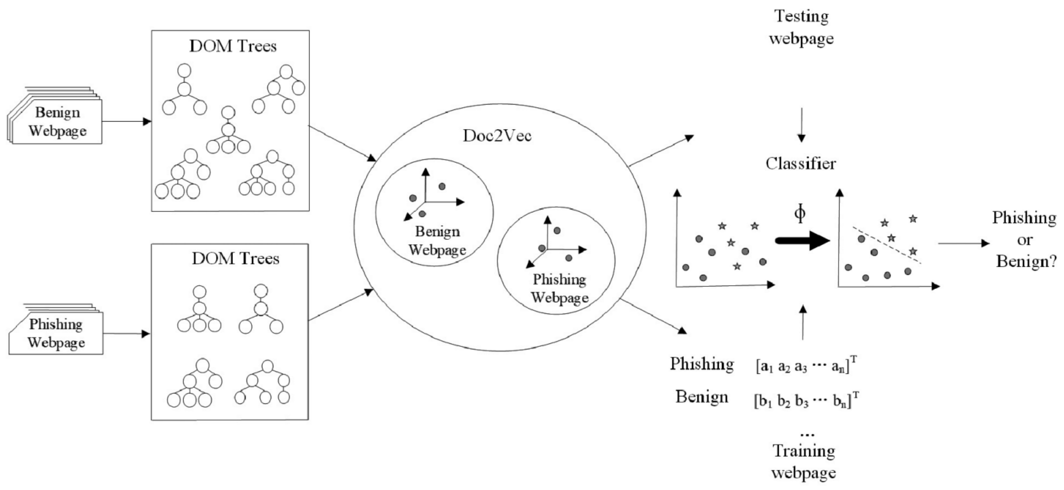


FIGURE 15: A illustration about the clustering comparison of web pages between a benign website and a phishing website **[118]**.

Dunlop, Groat and Shelly [119] presented an approach that named “GoldPhish” using OCR (Optical Character Recognition) to identify phishing activity as the trusted organization have easily recognizable brand logos on their websites. In their designed approach, GoldPhish takes a screen capture of the page that the user is visiting. This screenshot is converted from a Bitmap image into a TIFF image and is saved for OCR processing. In OCR process, GoldPhish focuses on the top portion of the screen capture as the logo is typically located at the top of the page, and a list of text entries is produced for submission to the search engine. According to the recognized text, GoldPhish implements the Google Search API and observes the first four results, as a legitimate website will generally come up within the first four results in a Google search due to its high PageRank. The accessed website is identified as a legitimate website if it can be indexed in the returns. Otherwise the accessed website is considered to be a phishing website as a phishing website that has been online for a short amount of time will have a low page ranking. Most phishing websites are active for a few days only, so are unable to attain a high ranking. This approach is efficient, but it has a higher requirement due to the screen capture. If the image resolution is either too high or too low, the recognized results from the OCR process are valueless. In their method, the resolution of screen capture is required to be 1200 x 400 pixels.

Aung and Yamana [120] believed that phishers tend to create phishing URLs with NAN characters, such as 1) extra unnecessary dots; 2) "//" to redirect the user to another domain;

3) "-" to mimic a legitimate website domain; and 4) unnecessary symbols. Also, for some phishing websites, phishers may hide information on the page's content, which results in the detection being difficult through using a content-based approach. So, Aung and Yamana proposed a new feature called the entropy of non-alphanumeric (NAN) characters for URL-based phishing detection. For example, the number of "." as feature Fx, the number of "-" as feature Fy, and the number of "@" as feature Fz. They then use a vector [Fx, Fy, Fz] as the final feature for a URL. The entropy of all NAN characters is feature Fe, where Fe = entropy (frequent probability distribution of NAN characters). This solution indicates the phishing website through checking the URL only, without accessing the content of website.

Nathezhtha, Sangeetha and Vaidehi [121] proposed a web crawler based approach to detect phishing attacks, which is called WC-PAD (Web Crawler based Phishing Attack Detector). In this solution, three phases were involved; a DNS blacklist, a web crawler based approach and a Heuristic based approach. In the web crawler approach, various parameters were extracted through the web crawler, such as page content, URL content and interconnected links. Then these parameters would be delivered to conduct further heuristic analysis. Some conditions were declared to classify if the target URL is a phishing website or not. For example, computing PageRank and AlexaReputation through the count of total visits, pages per visit, average visit duration and bounce rate, plus computing the occurrence of symbol "@","-" and ".".

Niakanlahiji, Chi and Al-Shaer [122] proposed a scalable feature-rich machine learning approach to detect phishing websites, called "PhishMon". Different to other machine learning based phishing detection approaches, they selected fifteen novel features that can be computed efficiently, using features that do not require interaction with any third-party services, such as search engines and WHOIS servers, thus decreasing the decision-making time. The features were extracted from four aspects, HTTP responses, SSL certificates, HTML document and JavaScript file respectively. Their solution achieves a high degree of accuracy in identifying phishing websites as the selected features capture various characteristics of legitimate web applications as well as their underlying web infrastructures.

The most common approach for launching a phishing attack is to send a phishing email [77]. A convincing email developed with social engineering can lure victims into trusting it [123]. A phishing email can be imagined as the source that launches a phishing campaign, thus some researchers put effort on the detection of phishing email [123]. Jakobsson [124] summarised an investigation of the ability of users to distinguish phishing emails, through spelling and design as important identifications. A phishing email can be identified from either poor spelling or grammar or an unconvincing logo and content in the email. However, Blythe et al [123] revealed that many current phishing emails contain no obvious spelling or grammar mistakes. At least, 38% were spelled correctly and 68% made themselves look convincing by copy pasting from the legitimate source. Phishing emails are becoming more convincing with better spelling, and therefore more difficult to spot.

A research poster from the University of Washington introduced work on simulations of phishing emails. Dupuis and Smith [125] believed a realistic looking phishing email mimics what individuals would typically receive from an organization or other entity for which they have a prior relationship. However, the problem is that simulation of this kind of phishing email means using name and logos associated with the entity, which presents a legal challenge. In their investigation, they launched a real phishing email campaign with the IRB (Institutional Review Board) approval and 146 participants were sent the email. 11% of the participants clicked the phishing link from the phishing email.

Nicholson et al [92] presented work using Signal Detection Theory to evaluate phishing email issues on the effects of both sender saliency and receiver saliency. They found that users are easily identifying phishing emails when their attention is focused on the sender's details, such as name and original email address, and the received time.

Many Chief Information Security Officers (CISOs) who responsible for phishing awareness training are often concerned when the phishing success rates are higher than expected within their organization. Greene [126] has shown that the successful phishing email will vary based on its context. In particular, when the premise of a phishing email aligns with a user's work context, it is much more challenging for users to detect a phish. Moreover, it results in some people questioning the efficacy of phishing awareness training. Steves et al [127] proposed a

Phish Scale that determine the difficulty rating for phishing training exercises. This work categorized the reason for people falling for a phishing campaign, and the actual phishing success rate is computed through providing various cues (such as spelling and grammar irregularities, attachment type, URL hyperlinking etc) during the training. This can help CISOs explain the associated reason of successful phishing campaign.

Gutierrez [128] presented a machine learning based approach to efficiently identify and purge phishing emails. Current phishing email detection mechanisms involve three categories; rules-based, classifiers and manual effort. In the rules-based category, the malicious phishing emails can be idented by filtering with the generated rules that include various keywords, syntax checks and sender address etc. In classifiers, the phishing emails are detected by using statistics and pattern matching. In the manual effort approach, detecting phishing emails requires end users to detect the deceptive content within emails. In their approach, the phishing emails detection were not only based on the surface level text, but also from linguistic patterns and subterfuges. A rich feature set was selected, which consists of five "high level" of features: (i) frequently known phishing words and their synonyms, (ii) relevant words associated with the tier-one university[5] sites used in experiment, (iii) frequently occurring words from the phishing corpus and their synonyms, (iv) proper noun organization names and (v) structural features in the email. According to their findings, these features were effective.

Ripa, Islam and Arifuzzaman [129] proposed a combination approach to detect phishing URLs, phishing email and phishing websites. Firstly, in phishing URL detection, they deployed the XGBoost algorithm to detect phishing URLs. Three features were considered in this phase; create age (in months), expiry age (in months) and update age (in days). In the phishing email detection, Naive Bayes Classifier were implemented to categorize the email into non phishing mail or spam by text categorizing. Five features were selected in this phase; 1) the total number of unique words; 2) the total number of words in spam; 3) the total number of words in ham; 4) the count of each word in spam; and 5) the count of each word in ham. In phishing website detection, five algorithms (Random Forest Classifier model, SVC, Logistic Regression,

[5] Tier-one university: This is the top university in US, it always refers as TOP 50 of US university ranking.

KNN and Decision Tree Algorithm), were deployed to train their data, and 30 features were extracted from address bar, abnormal, HTML, JavaScript and domain. In their findings, random forest classifier method provided the best performance compared to other approaches.

3.4.3. Relevant reviews on social engineering attacks and victims

Additionally, there are many extensive studies about social engineering attacks and their victims, as phishing attacks are considered to be the most complicated and incessant type of socially engineered attack [130], which targets based on both the psychological and physical perspective [131]. In order to obtain the physical access permission of an information system, the social engineer usually exploits a psychological approach to gain the trust of an employee and perform their deceit [132].

In [133] Yasin et al proposed a framework for social engineering attacks, which described the existing social engineering attack scenarios from the social roles played by victims and phishers, the potential vulnerabilities of the victims, the principles exploited by phishers, and the storyline used by phishers. This framework consists of five phases: Gather, Medium, Scenario, Persuasion and Result. In the Gather phase, the social engineering phishers gather the victim information, such as account information, email, friends' information etc, using social networking, and generate a storyline for the further phases. In the Medium phase, according to the information gathered, the phishers contact the victims via the most intense way, such as by calling, or via email. In the Scenario phase, a key part in social engineering attacks, the phishers need to impersonate and make up a story to lure and convince the target victim using the information collected. In the Persuasion phase, the phishers try to take advantage of psychological weakness, such as feelings and emotions, to decrease human security mechanisms, making victims perform the actions that are desired by the phishers. In the Result phase, the phishers achieve their goals. Usually, the goals require victims to transfer money, or share sensitive information, or install a malicious application. An illustration is shown in Figure 16 below.

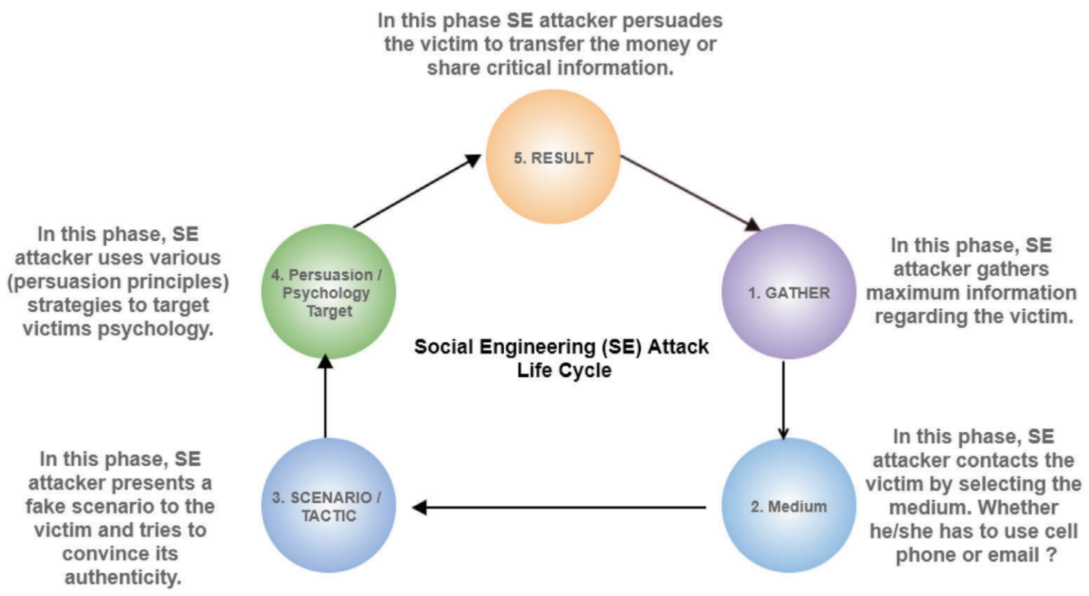

In this phase SE attacker persuades the victim to transfer the money or share critical information.
5. RESULT
1. GATHER
In this phase, SE attacker gathers maximum information regarding the victim.
2. Medium
In this phase, SE attacker contacts the victim by selecting the medium. Whether he/she has to use cell phone or email ?
3. SCENARIO / TACTIC
In this phase, SE attacker presents a fake scenario to the victim and tries to convince its authenticity.
4. Persuasion / Psychology Target
In this phase, SE attacker uses various (persuasion principles) strategies to target victims psychology.
Social Engineering (SE) Attack Life Cycle

FIGURE 16: A illustration of social engineering attack life cycle **[133]**.

Phishing attacks are dangerous as the victims are often unaware of being exploited [134]. Kelley et al revealed most victims always ignore the website warning that the accessed website is not secure if this website is frequently accessed in the history [135]. In [136] Sarno, Lewis and Neider conducted research about the victims of phishing attacks, to determine if there are age-related differences in the victims of phishing attacks. According to their findings, compared to younger adults, older adults are more vulnerable to phishing attacks. Although the older adults are more cautious while classifying phishing emails, this extra caution may only reduce the classification speed rather than improve the classification accuracy. Older adults are more likely to click on a link from an email, and this decision is subject to whether the content of the email is interesting. Also, they found that both older adults and younger adults have a poor accuracy in identifying phishing emails in their experiment.

The susceptibility of older adults in phishing attacks is a critical and growing public concern [137]. Three main reasons are summarised; 1) older adults have accumulated financial assets over a life, which they have increasingly managed online; 2) Many older adults occupy powerful positions in finances and politics and are confronted with a variety of decision-making situations in cyberspace; 3) the sensitivity of deception declines with age [138]. Lin et al [137] conducted an investigation on the effect of Internet user age and email content and life domains on spear-phishing susceptibility. In their experiment, the participants who included 100 young and 58 older users, received daily simulated phishing emails over 21 days. Lin et al found that 43% users fell for the simulated phishing emails, where older women demonstrated the highest susceptibility. Moreover, the susceptibility in young users declined, but in older users it remained stable. The effectiveness of a phishing attack is different, and varies with different age groups; young users were most susceptible to scarcity, and older adults were most susceptible to reciprocation.

Social engineering attacks have become a serious threat, and it is an effective approach to attack information systems within an organization [139]. Hove [140] believed a reasonable strategy within an organization is to enhance the protection of data and mitigate social engineering attacks. Also, three major mitigation metrics have been confirmed from successful attack strategies: (a) IT risk management strategies, such as systems backup, isolate the network etc, (b) employee training strategies, and (c) cyberattack contingency plans, such as the use of multifactor authentication and installing reputable security software etc. Hove also [140] believed the development of a successful strategy could detect, minimize and respond to business and system risks, identify the vulnerabilities of their information assets, and assess the security degree of the organization through the identification and evaluation of the industry and organizational risks.

Astakhova and Medvedev [141] believed that anthropogenic methods of defence against social engineering attacks are important, but the effectiveness is not yet good enough. They developed a resilience scanner which can not only determine the resistance of an organization's employees to social engineering attacks, but also provides a training system with complex feedback. This application involves three functionalities: accessing the employees' vulnerability to social engineering attacks through the distribution of probabilities for their possible psychological types; training employees to counter social engineering attacks; and monitoring the level of employee information security awareness.

Heartfield and Loukas [142] proposed a human-as-a-security-sensor (HaaSS) framework, which can be used to actively improve the existing technical detection systems to confirm and highlight the extent of the threat through the combination of user detected threats and technical defence system flagged threats. This framework is formulated by a set of three core processes, involving detection, classification and response. Moreover, this framework was implemented in the form of Cogni-Sense, which is designed to encourage participants to actively detect and report the relevant semantic social engineering attacks. According to the findings, during the experimental period of 45 days, human sensors performed a better detection result than technical security system, with the missed detection rate below 10%.

With the emergence of cryptocurrency, various social engineering attacks have taken place on cryptocurrency users. In [25], Weber et al summarised several popular cases (Red Pulse, Blockchain.Info, Bee Token and Fake Twitter Account) of cryptocurrency frauds using social engineering attacks. In each case, the target victims either transferred their cryptocurrency asset or disclosed their credentials due to phishers impersonation, resulting in financial losses for the victims. In the Red Pulse case, phishers created a fake twitter account to lure the victims to access their designed phishing website, and asked victims enter their private key for "earning" a bonus. In the Blockchain.Info case, phishers created an ad from Google AdWords, this ad was shown to victims once they searched for key words, such as "blockchain" and "bitcoin wallet", on Google. This ad came attached with a phishing link, which redirected a victim to the phishing website for stealing the victim's credentials. In Bee Token, the phishers stole the victim's data from the outsourced KYC (Know you customer) procedure and the newsletter module, they then sent a phishing email to these victims to lure them into transferring cryptocurrency. In the Fake Twitter Account case, the phisher created a fake twitter account, using information copied from a famous investor including his name and profile. This fake twitter message lured its followers to transfer cryptocurrency to the phisher's wallet.

In Iswarya and Preetha [143], an analysis about social engineering attacks in online banking was researched. They designed a descriptive experiment to evaluate human psychological ploys under this kind of scenario. The psychological features were identified as curiosity, empathy, excitement, fear, and greed [144]. According to their findings, Iswarya and Preetha believed the most vulnerable medium used to launch a social engineering attack by phisher is (phishing) email. The psychological features of social engineering people are most vulnerable to are greed, excitement and curiosity. Moreover, they found the age group between eighteen to twenty-five years are more vulnerable to the fear trait. However, over twenty-five-year-olds are not affected by the fear trait. The greed trait is evident across all ages and curiosity is also a vulnerability across all genders.

Lansley et al [145] presented an approach to identify conversation dialog either online or offline as a potential social engineering attack, or not. In this approach, they converted the dialog into a dataset for classification and they scored three aspects; Link score, Spelling score and Intent score. Various classification techniques have been used in each implementation, such as Case-based Reasoning, Fuzzy Logic, Decision Trees and Deep Neural Networks. In the link score, URL features were selected from the attacked URL link in a dialog, they are URL ranking, URL length, URL dot count[6], URL IP, URL links in count and URL word number. In the spelling score, the dialog content was parsed and checked for grammatical errors. In the intent score, the conversation text was compared to a blacklist word, which involved forty-eight security policy style words, such as credentials, passwords, database etc. The results of each score are weighted by importance, and the system made a final decision as to whether a social engineering attack is taking place.

3.4.4. Relevant reviews on security platforms and tools

There are some studies focussed on security platforms and tools that mitigate phishing attacks. For example, Stafford [77] summarised and evaluated prevention approaches from existing cybersecurity platforms for combatting phishing threats. These are least privilege, effectiveness of filters and lists in detecting phishing email, and utilizing AI and machine learning to detect cyber threats. Moreover, Stafford believed least privilege does not help an individual identify and detect a phishing email, but it limits the degree of information exposure, as the phishers have limited permission to access the data. Tools such as filters, blacklists and whitelists contribute to detecting phishing emails, but they require human input

[6] URL dot count: The count of dot (.) in the URL, for example, if the URL is *www.st-andrews.ac.uk*, the URL dot count is 3, the first dot is behind of www, the second dot is behind of st-andrews, and the third dot is behind of ac.

and update to their databases and rules. Therefore, AI and machine learning perform better for detecting phishing threats with continuing threat evolvement.

Tian [146], developed a python tool named “SquatPhish” and released on Github. This tool supports a machine learning based approach to detect a general phishing attack. The experimental features were selected mainly from three aspects; Firstly, features include a brand name and sign-in keywords in the HTML source code. Secondly, in the HTML structure, features include submission forms and their attributes. Thirdly, in any Image text, the content of an image is extracted using OCR from the image. Also, Tian used NLP to filter and discard nonsense words. Finally, the author implemented a Random Forest classification algorithm to combine all the properties to predict the result.

3.4.5. Relevant reviews on two factor authentication

Along with the above studies, some researchers believed two factor authentication could be deployed to mitigate phishing attacks. Two factor authentication (also named 2FA) is a security mechanism which implements two vectors for security authentication, and it is considered more secure than the traditionally implemented one factor authentication system. Three commonly recognized authentication factors are: what you know (e.g. passwords), what you have (e.g. tokens), and what you are (biometrics) [147]. Over the last few years, 2FA is implemented by most popular websites, such Google, Microsoft, Facebook, and Twitter [148]. 2FA helps users to protect their identity and accounts; an extra authentication request is sent to a user on a different vector, to check and confirm extra identification. For mitigating phishing attacks, 2FA can protect the user’s account, even if the phishers collect the user’s identity, the phisher still cannot access the target website as the user’s identification has to be checked through two vectors [149]. Essentially, the user’s account cannot be logged into by others without the user’s agreement.

However, 2FA alone will not prevent all phishing attacks from succeeding. There are various approaches which can bypass the 2FA mechanism. For example, in a runtime phishing attack, if a victim is directed to a phishing page and enters their credentials, the phisher can then use those details in real-time to log into the legitimate site [150]. An illustration of this scenario is shown in Figure 17 below. A 2FA code request is sent to a victim (in step 4), this victim then types in the code into the phishing website (in step 6). Subsequently, this phisher uses this code to log into the legitimate site as the user's identification (in step 7 & 8). In the end, this phisher may send anything back to victim making them truest the accessed website is legitimate website so that reduce their security awareness (in step 9).

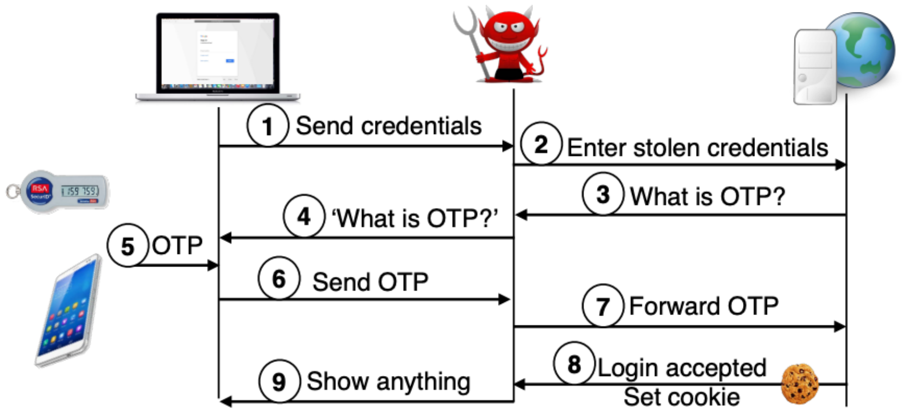


FIGURE 17: The workflow of runtime phishing attack **[151]**.

In [151], Ulqinaku, Lain and Capkun presented a scheme that overcame the challenge of phishing attacks prevention by 2FA, which is called 2FA-PP. In their scheme, 2FA-PP relies on direct communication between the smartphone and the browser through the Web Bluetooth interface. An illustration is shown in Figure 18 below. A server delivers the encrypted code to the user browser; the browser then requires the decryption key through the phone app; the decrypted result is sent back to phone app for the verification; the communication channel between the phone app and the browser is through Bluetooth. If the verified result is correct, the app will approve the authentication request to the server for login.

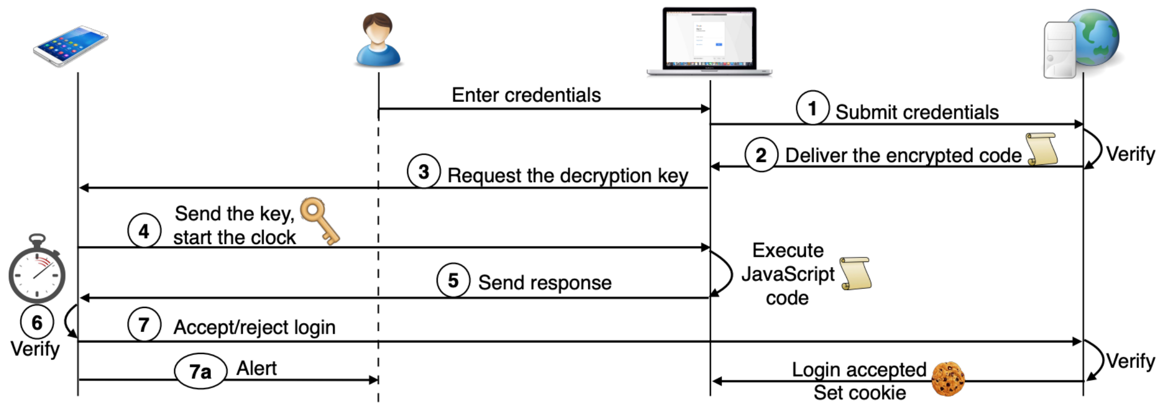


FIGURE 18: The workflow of 2FA-PP **[151]**.

3.4.6. Relevant reviews on anti-phishing education intervention

Technology-based solutions are a vital component in mitigating phishing attacks. Humans are the weakest link in information security, if an individual lacks sufficient security awareness that is the primary reason that a phishing attack succeeds [77]. Kelly, Amon, and Bertenthal [135] believed that although a person may have a high degree of security knowledge, it does not ensure that an individual appropriately responds to all relevant factors of a security threat. Thus, improving user security awareness is essential. Moreover, for an organization, Dawson [152] believed that advancing security education and knowledge of individuals may improve their own personal security habits. Meanwhile, Dawson also stated that the organizations security habits have been improved if their employees are more aware of the security risks that are currently prevalent.

Anti-phishing educational interventions play a critical role against phishing attacks by transforming the human weakest link to the strongest cybersecurity defence [153]. Previous research in anti-phishing education focused on three aspects; improving teaching methods, user interactions and user behaviour modification [154]. URL obfuscation refers to the innovative ways used by phishers to design a phishing URL to trick humans. Research on URL obfuscation techniques used is necessary in anti-phishing education. Current anti-phishing education is based on four obfuscation categories; Obfuscating the Hostname with an IP Address, Obfuscating the host with another domain, Obfuscating with large hostnames and

Domain name unknown or misspelt [155]. However, over time, phishers have abandoned the URL obfuscation approach techniques and have kept inventing new ways to launch a phishing attack. In [154] Fernando and Arachchilage presented two more URL obfuscation categories in order to catch up with the constantly changing phishing attack mechanisms. Obfuscating with HTTPS Schema Technically and Obfuscating with Internationalized Domain Names have been trending for phishing usage since 2019.

Bullee and Junger [156] conducted research on how effective social engineering interventions are. Two research questions were presented; whether an intervention can reduce the number of victims of social engineering attacks and whether the research features have an impact on intervention outcome. According to their findings, intervention effectively decreases the social engineering vulnerability, high-intensity interventions (such as lecture, game with Q&A and training) are more effective than low-intensity interventions (such as information with tips), although high-intensity intervention may require more time consumption. Moreover, narrowly focused interventions (intervention materials, such as malicious URL, malicious email, social engineering, cybercrime etc.) were more effective than broadly focused interventions.

In order to improve user security awareness, in particular to phishing attacks, many researchers designed a training session with various education paths, such as lectures, videos, and games. Experimental results revealed that game participants could easily recognise a fake website [157], so the game is a possible approach to engage participants and transfer relevant knowledge in a unique and easy way [158]. Various studies have been performed to design digital games for training users to identify and prevent phishing attacks [78]. Fatima et al [131] designed a game-based solution to educate participants to identify and thwart spear-phishing attacks. In this game, a storyline about a hospital engages participants to hack the (hospital) system using phishing attacks was performed, wherein participants play the role of a phisher to spoof target victims and devices through various attack techniques and attack material preparation. From this game-based learning, the participant not only understand the hazards of excessive online information disclosure, but also improves phishing awareness, such as how to identify a phishing email, how to identify a fake URL and how to identify similar and deceptive domains.

3.4.7. Summary of phishing mitigation approaches

Therefore, mitigating phishing attacks is a critical research topic that is worth studying. Although much research has been conducted, this threat still exists in the real world, with a constantly increasing prevalence. According to our systematic search in the relevant literature, the detection of phishing attacks is a challenging problem. Two core strategies are deployed to mitigate phishing attacks; either improving the performance of phishing detection technology or developing human education intervention.

Developing human education is a fundamental way to mitigate phishing attacks, as phishing attacks exploit weaknesses in human characteristics, rather than networks flaws. Also, humans always are the weakest link in social engineering attacks. A good education intervention is important to ensure users understand how to identify phishing attacks, and the quality of the education can affect the success of preventing phishing. Thus, an experiment of education intervention requires users to participate, because the quality of education is subject to the comparison of successful rates of phishing identification in participants before and after the associated learning.

However, for improving the performance of phishing detection technology, researchers focus on two aspects to mitigate phishing attacks; one is based on the detection of phishing emails as email is the most vulnerable medium to launch a phishing attack. Thus, phishing attacks would be blocked from the origination if the phishing email could be detected. The other is focused on the detection of phishing websites, because most phishing attacks exploit a phishing website to illegally gather victim data.

Compared to phishing website detection, the detection of phishing email may require the participation of users in order to achieve a better detection result. Because the success of phishing email will vary based on its context. In particular, when the premise of a phishing email aligns with a user's work context (or current situation). Thus, in this case, the user background and current situation are relevant features and there would be a high weight associated with those variables if researchers would implement machine learning to train the corresponding data. Also, the evaluation would require participants with various backgrounds for covering a variety of situations and tests.

In the detection of phishing websites, list-based and content-based are two main technical approaches that are explored and deployed to detect and classify phishing websites in general. Compared to the content-based approach, a list-based approach requires much human effort to classify phishing links and update the list library. In some cases, in order to increase the efficiency of a human identifying phishing website, researchers categorize the webpages according to their similarities (as most phishing website are designed by toolkit, which is high content and layout similarity). Thus, in order to verify the effectiveness of webpage categories, the evaluation always needs human participation, as the increased efficiency of identifying phishing websites is subject to the comparison website between before and after the webpage category.

In the content-based approach, some researchers confirm the phishing websites directly by identifying website content and features; some researchers confirm the phishing website according to the predicted result. Various machine learning approaches are derived, and many feature-based algorithms have been developed to automatically detect the accessed websites. Thus, for this kind of approach, the evaluation requires many phishing websites. A good content-based approach is subject to the efficiency of phishing website detection and the success rate of identifying phishing websites.

In a survey paper from the University of Badji, Mokhtar, Zaimi, Hafidi and Lamia [159] presented a taxonomy of anti-phishing solutions, defined into two categories of content-based anti-phishing detection and non-content based anti-phishing detection. The former contains URL analysis, image analysis and text analysis. The latter consists of blacklist & whitelist, DNS-based technique, user website rating and domain popularity. However, according to the reviewed literatures above, more mitigation approaches should be considered in this taxonomy. Thus, we improved this taxonomy, and proposed a more detailed one, as shown in Figure 19 below.

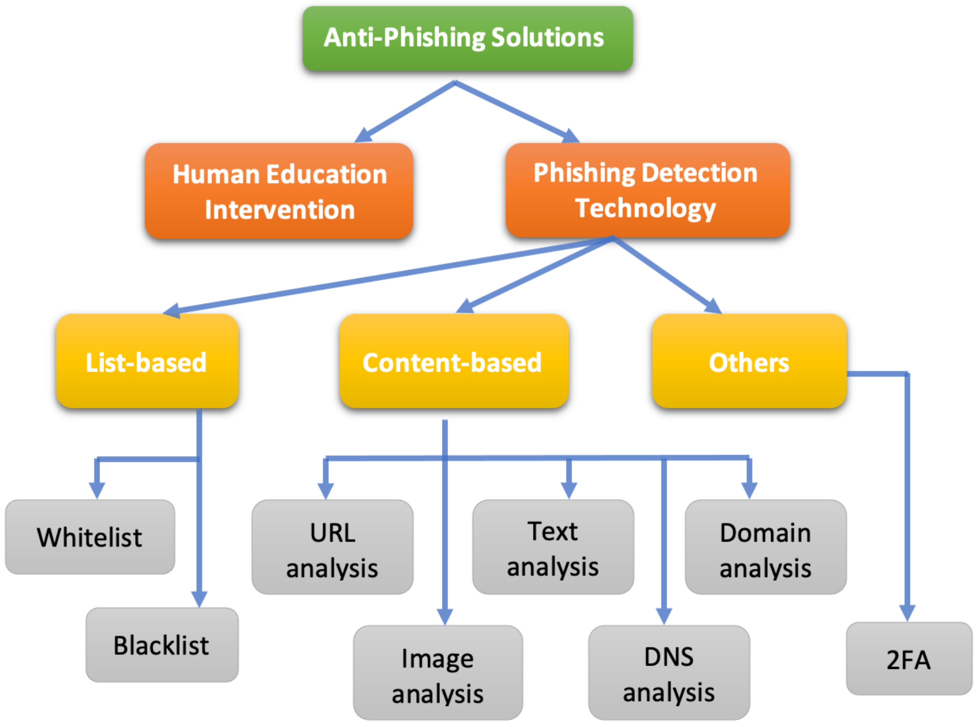


FIGURE 19: The taxonomy of anti-phishing solutions

In this thesis, we focus on improving the detection technique performance for identifying malicious websites. The mitigation approaches of phishing attacks based on improving the detection technique (in this Literature Review Chapter) are summarised Table 5 below:

Author	Approach Description	Reference
Liu et al	Their solution improves the performance of blacklist during the detection process through merging with computational technique, also it can be easily adopted by any existing manually verified blacklist.	[99]
Xiang et al	The authors proposed a hierarchical blacklist enhanced method to identify phishing attacks through exploiting existing backlists and applying fuzzy matching techniques in a probabilistic fashion. Because a large number of current phishing website are created with tools, there have been a high similarity of content	[101]

Geng et al	The authors proposed an approach that was based on mining brand resource requests to detect phishing attacks, because the loaded resources from the website may come from another domain. In their solution, the target URL and all queried domains was checked using both a blacklist and a whitelist.	[103]
Bell	An investigation of three popular phishing blacklists on Twitter. The OP blacklist removed a significant number of URLs from its dataset after a duration of 5 and 7 days, and no URLs were listed in OP for more than 21 days. Most malicious URLs were detected by the GSB blacklist within 6 hours of being tweeted	[104]
Drury, Lux and Meyer	The authors conducted an investigation on the life cycle of phishing attacks by analyzing the creation process of phishing websites. According to their findings, the certificates and WHOIS information might be used to detect phishing websites up to several days earlier. Other types of campaigns, such as launching attacks using hosting providers or the largest clusters, might still be detectable with predictive blocklists.	[105]
Sahingoz et al & Serrano et al	The authors proposed a machine learning based approach, which selected forty different NLP based features for training. Their result showed random forest algorithm gives the best performance in their solution. Subsequently, Serrano et al improved their solution, the experimental dataset has been changed in order to satisfy the situation closer to real-world scenario.	[28][29]
Zuhair, Selamat and Sallehs	The authors proposed a machine learning based approach, which selected fifty-eight features, including webpage content feature, URL features and online features separately.	[108]

Xiang and Hong	The authors proposed a novel hybrid detection approach based on both information extraction and information retrieval techniques. Two components are involved; identity-based components and keywords-retrieval components. They deployed a domain whitelist and a login form detector to filter out good web pages.	[109]
Basnet, Sung and Liu	The authors proposed a rule-based approach to detect phishing websites. Fifteen rules were proposed according to various factors, such as Search Engine-based Rules, Red Flagged Keyword-based Rule, Obfuscation-based Rules, Blacklist-based Rule, Reputation-based Rule and Content-based Rules. These rules were then used as features in Decision Tree and Logistic Regression learning algorithms to identify the phishing webpages.	[110]
Basnet, Sung and Liu	The authors evaluated the correlation-based and wrapper-type feature selection techniques. 177 features were applied, of which 38 features were from content-based, and the rest features were from URL-based. The experiments showed that a wrapper-based technique has a better accuracy.	[111]
Cui	The author proposed detection mechanism consists of two categories, the client-side of phishing detection and the server-side of phishing detection. A clustering-based approach was implemented in the client-side detection, as a common behaviour of phishers is to repeatedly launch their attacks with using different domain names and hosts. A machine learning approach was proposed for the server-side detection, and it was able to identify the specific patterns of outgoing traffic that are phishing.	[112]
Huh and Kim	The authors proposed a heuristic-based approach to identify phishing attack by detecting the search results	[113]

	returned from popular web search engines. Because a phishing website has a less searched results and low ranking.	
Botejue	The author explored the performance of various machine learning algorithms on phishing detection issues. Eight different algorithms were examined. According to the experimental findings, XG Boost algorithm provides the best classification performance compared to other algorithms.	[114]
Rosiello et al	The authors proposed a DOM tree-based approach. The system compares the structure of DOM tree between the first website where the data was initially entered and current accessing website where the data is reused.	[115]
Xiang et al	They authors proposed a comprehensive approach that exploited the DOM, search engines and third-party services with machine learning to detect phishing attacks. Two mechanisms were designed to filter the input webpages during the testing process in order to improve the performance. Various features are used; there are URL-based features, HTML-based features and web-based features. According to their findings, Bayesian Network learning algorithm performed the best result.	[116]
Xiang	Xiang proposed a cascaded approach to classify the phishing websites and he implemented different types of features into multiple stages of a single cascade. Each stage consists of a machine learning classifier using a different subset of features. The lightweight features are used in early stages of the cascade. Thus, his approach has a better runtime performance with ideal accuracy.	[49]
Feng, Zhang and Qiao	The authors also proposed a DOM tree-based approach to identify phishing attacks from the perspective of clustering	[118]

	via comparing the structure of web pages, as they believe the clustering of web pages is difference between benign website and phishing website.	
Dunlop, Groat and Shelly	The authors deployed OCR to identify phishing. They located the logo from the top position of webpage screenshot, then used OCR to extract the text content from the logo, and identify the legitimate website using Google Search.	[119]
Aung and Yamana	The authors proposed a new feature called the entropy of NAN characters for URL-based phishing detection. The specific symbols of URL, such as ".", "-" and "@", were selected and combined into a new vector to indicate if the target URL is malicious.	[120]
Nathezhtha, Sangeetha and Vaidehi	The authors proposed a web crawler based approach to detect phishing attacks. Various parameters were extracted through a web crawler, such as page content, URL content and interconnected links. Some conditions were used to classify if a target URL was a phishing website.	[121]
Niakanlahiji, Chu and Al-Shaer	The authors proposed a machine learning approach, fifteen features have been selected from the webpage without using third part services.	[122]
Tian	The author developed a python tool to identify phishing using machine learning on Github. The features are selected from html source code and image data.	[146]
Ulqinaku, Lain and Capkun	Their solution bases on 2FA, which relies on direct communication to confirm the user identity between the smartphone and the browser through the Web Bluetooth interface.	[151]

TABLE 5: The summary of phishing mitigation approaches.

From this summarised table, a blacklist approach usually needs enormous human effort to manually verify suspicious websites as phishing, and thus the reported phishing websites are slow, as human verification requires much time. Thus, we do not consider this kind of method. Various machine learning approaches are derived from a content-based scheme, but the accuracy of expected results is subject to the integrity of extracted features, and always has a high false positive value. Moreover, the runtime is increased if the features are sophisticated. Thus, we consider using a straightforward content-based approach to judge phishing attacks without using a machine learning method.

For preventing general phishing attacks, OCR is an ideal approach to recognize what the accessed website is from its logo, but we require a direct way to locate the logo position rather than using machine learning from the website screenshot, like GoldPhish. Also, the identification process between a phishing website and a legitimate website requires a dynamic approach without using any human effort, and this approach needs a reasonable indicator as it needs to deal with various and different scenarios. The domain comparison between a legitimate website and a phishing website is questioned, in particular, if the victim is under a DNS hijacking attack. Thus, we present a first hypothesis in this part:

*H1: "**An OCR approach can be implemented efficiently to detect phishing attacks without using machine learning to predict and category the results**"*

Most anti phishing solutions are implemented to mitigate general phishing attacks, but they ignore considering some specific situations, such as advanced phishing attacks. For preventing advanced phishing attacks, phishing websites are difficult to be detected if the victim is under a DNS hijacking attack as both URL content and website content are the same as the legitimate websites. Most content-based approaches may not work as the content of accessed URL is a considerable feature. However, a valid and trusted SSL certificate could not be forged. Thus, we consider identifying the consistency of SSL certificate between the accessed website and legitimate website to identify the phishing website that is under a DNS hijacking attack. Furthermore, for an ISP hijacking attack, phishers may not phish the victim from a legitimate website directly, but they could lure victims directing them into a phishing website by inserting convincing context in the legitimate website. Thus, we consider detecting the hijacked information to prevent victims into a fraud. The DOM tree is an ideal indicator to indicate whether the user's traffic is hijacked, as the distribution of nodes within a DOM tree is different if the website has undergone any changes. Therefore, we present the two following hypotheses to DNS hijacking attack and ISP hijacking attack experiments separately:

*H2: "**A phishing website that is under a DNS hijacking attack can be detected by implementing OCR and examining the associated SSL certificate**"*

*H3: "**ISP hijacking attacks can be identified by observing the consistency and differentiation of DOM trees**"*

For preventing subdomain takeover attacks, a phishing website is difficult to detect if phishers put this website into a subdomain taken from a legitimate website. No matter the web content, URL address and SSL certificate information, all of them will be the same as the legitimate website. Thus, for avoiding the occurrence of this situation, we consider deploying an automatic approach to determine the risky subdomains, not only improving the detection performance, but also reducing the human effort. Moreover, the approach of querying subdomains needs to be improved, as most current tools are based on brute force, the existing dictionary may not cover all cases of subdomains as some subdomains may be meaningless. Thus, we consider providing a prediction method to generate more realistic content. In this part, we present two more hypotheses as below:

*H4: "**Subdomains can be predicted using machine learning to cover the shortage of non-dictionary word domains in the existing dictionary**"*

*H5: "**The risky subdomain can be discovered by an automatic approach without any human efforts**"*

4. Research on General Phishing Attacks

In this chapter, the general phishing attack is researched. As general phishing attacks are the most common attacks, we first demonstrate the workflow of launching this attack from a technical viewpoint. We also explore phishing prevention differences between desktop and mobile platforms, in order to confirm and prove that mobile platforms have less protection against phishing attacks. The relevant methodology used in this research is thoroughly elaborated in subsequent sections. Then, the process of deriving prototypical systems along with the relevant technologies will be introduced and explained in detail. Finally, the evaluation is narrated, and the corresponding challenges faced are analysed and discussed.

4.1. Related Works & Pre-Experiments

In this section, we explored general phishing attacks from the following aspects:

- How easy is it to launch a phishing attack?
- What are the phishing prevention differences between desktop and mobile platforms.

From the first aspect, we can understand how easy it is to launch a phishing attack by a novice phisher. From the second aspect, the phishing prevention comparison between desktop and mobile platform will be investigated, and we also performed several experiments to determine this difference. At the end of this section, the specific research goal for mitigating general phishing attacks will be presented.

4.1.1. How easy is it to launch a phishing attack

Forging a phishing website may have been a complicated procedure in the past, as the phisher not only needed to master at least two programming languages, but also required to be familiar with Linux (or other) configurations for the back-end (server side). The programming languages, HTML and JavaScript are essential skills as a phisher needs both to develop a complete website page to imitate the corresponding official website. Also, a phisher requires another programming language for back-end development, where the purpose is to receive the information from the front-end. The most common back-end programming language is PHP. For the Linux configuration, the phisher needs to set up a public server and put this phishing project into this server. As this phishing webpage needs to be accessed via either HTTP or HTTPS, so the relevant server port and services require to be pre-configured before launching a malicious phishing campaign. Thus, Linux configuration abilities on the server side was also an essential skill for the phishers. In order to improve the camouflaging of a phishing attack, a phisher would also have to purchase a similar domain name to the corresponding official domain to confuse the victims. For example, if the phisher wanted to forge a malicious PayPal website, the similar domains they would be keen to buy would be PalPay, PayPay or PayPel etc.

However, due to the rapid development of technologies and the increasing number of programming languages, libraries and tools have been encapsulated and upgraded. Now, launching a phishing campaign is no longer technically difficult. Firstly, for the front-end phishing page, it is unnecessary to master the skills of HTML or JavaScript, as there are various ways of forging a phishing page similar to the official page. One of the easiest ways to implement duplicating a page is to copy HTML code from **View Page Source** to a new HTML file, as shown in Figure 20 below. Alternatively, by using the terminal command **curl example.com > phishpage.html**, the HTML code of *example.com* will be saved to the file of *phishpage.html*, as shown in Figure 21 below, essentially forging a PayPal login page.
(**NB**: Some images cannot be loaded as their path may be incorrect. In order to resolve this problem, you can either modify the image path in your phishing page or download the image and save to the same file and path within your phishing page.)

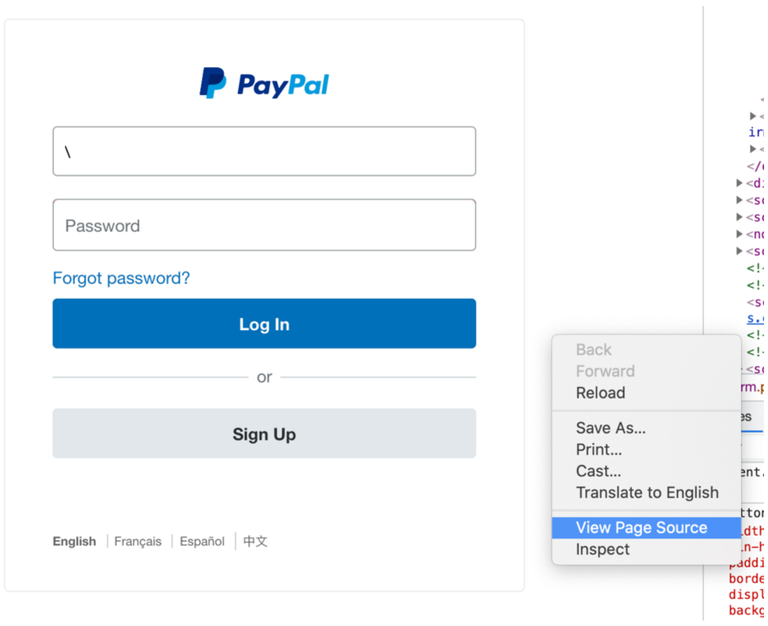
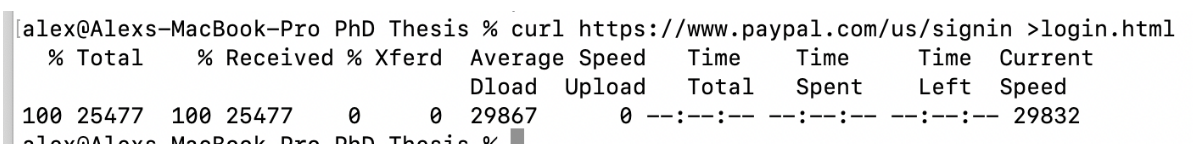


FIGURE 20: View Page Source option in Google Chrome Browser.



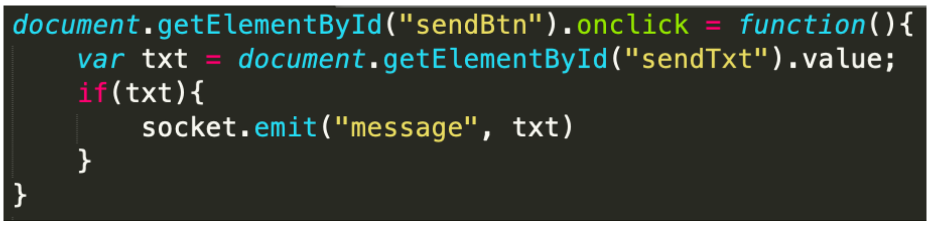
```
alex@Alexs-MacBook-Pro PhD Thesis % curl https://www.paypal.com/us/signin >login.html
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 25477  100 25477    0     0  29867      0 --:--:-- --:--:-- --:--:-- 29832
alex@Alexs-MacBook-Pro PhD Thesis %
```

FIGURE 21: Duplicating PayPal HTML code via terminal command on a MAC.

Secondly, we need to adapt the back-end system to receive the information (such as username, password) transferred from the front-end. In the past, most phishers preferred to use a powerful language, such as PHP, to create the back-end service. However, now there are other languages which can be used to completely replace PHP. Python is one option we can use to implement this function, but it needs to be learnt from the beginning. Thus, we recommend Node.js, which is designed to build scalable network applications through asynchronous event-driven JavaScript runtime calls. Not only does it not require learning another extra programming language as the usage is quite similar to JavaScript, but the implementation is also simple. For instance, there are many libraries encapsulated for Node.js, such as socket.io[7]. We use this library to receive the information from the front-end by using a built-in function of *socket.emit()* and *socket.on()* to establish the communication channel. We locate the information by using the function *document.getElementById().value* in the front-end, and declare an ID to this value. Next, the back-end server can receive this value by

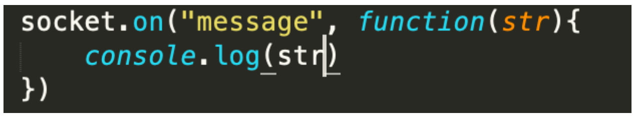
[7] Socket.io: It is a JavaScript library for real time web application. Available at: *https://socket.io/*.

recognizing the ID, as shown in Figures 22 and 23 below. Thus, in our experiment on forging a phishing webpage, we located a login button, and wrote script codes here for sending the username and password value to the back-end by using the socket.io library. Then in the back-end, we used the *console.log()* function to gather the received data.



```
document.getElementById("sendBtn").onclick = function(){
    var txt = document.getElementById("sendTxt").value;
    if(txt){
        socket.emit("message", txt)
    }
}
```

FIGURE 22: An example of *socket.emit()* function in the front-end for sending data to the back-end.



```
socket.on("message", function(str){
    console.log(str)
})
```

FIGURE 23: An example of *socket.on()* function in the back-end for receiving data from the front-end.

Subsequently, we purchased the latest ubuntu version server with an IP address from DigitalOcean[8], and installed a Nginx service on this server for running web servers for the phishing website. Installing either HTTP or HTTPS service is not difficult anymore since the emerging of Nginx, as with only few commands[9] on the server, a complete web server will be deployed. After setting up a web server, we need to upload all of the phishing project into our server, and install the relevant packages to ensure all the services can be run successfully. In this project, the following packages required to be installed before the implementation of the phishing campaign experiment, as shown in Table 6 below:

[8] DigitalOcean: It is a cloud vendor; its mission is to simplify cloud computing so users can spend more time creating software that changes the world. Available at: *https://www.digitalocean.com/*.
[9] The relevant documentation of how to install Nginx on Ubuntu server is in: *https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-20-04*.

Package name	Functionality	Install command
Node.js	For establishing the back-end service.	brew install node
Nodejs websocket	For establishing the web socket between the client and the server	nmp install nodejs-websocket
Socket.io	For sending and receiving data between front-end and back-end	npm install socket.io

TABLE 6: The relevant packages required to be installed on the server side.

Finally, we purchased a domain name, such as *example.com*, from a domain name vendor. This domain was also configured to be resolved to our server (IP address). Our phishing website was completed after we activated the back-end JavaScript file by using the command `node xxx.js` on the web server. The victims who accessed our domain *example.com* via a browser will be under a phishing attack, and any information they type in and submit will be transferred to our server. The process of launching this phishing campaign is illustrated in Figure 24 below. Moreover, in order to improve the camouflage of our malicious address *example.com*, we can use shortened URL or QR code techniques to confuse the victims.

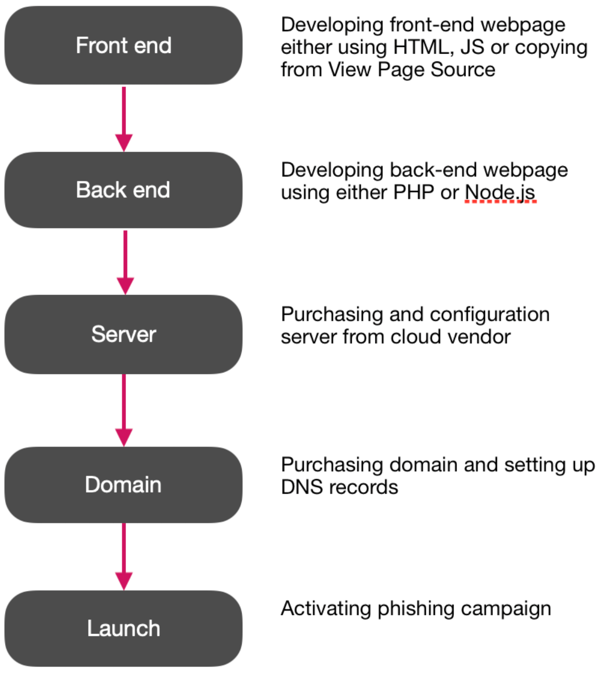

Front end
Developing front-end webpage either using HTML, JS or copying from View Page Source
Back end
Developing back-end webpage using either PHP or Node.js
Server
Purchasing and configuration server from cloud vendor
Domain
Purchasing domain and setting up DNS records
Launch
Activating phishing campaign

FIGURE 24: The process of activating a phishing campaign.

Therefore, there is not a high technical skill required to launch a phishing campaign. Everyone can be a phisher, even people who have a non-technological background, as long as a novice phisher can follow a procedure such as noted above.

4.1.2. Phishing prevention differences between Desktop and Mobile Platforms.

Presently, hackers have focused their attention onto mobile devices to gain more benefits as users have increasingly shifted towards using mobile devices for daily tasks [50]. Mobile application users are more vulnerable to phishing attacks [51] [93]. In order to verify the severity of mobile based phishing risks, we conducted a comparative experiment, as noted above, to examine the security of mobile browsers to defend against phishing attacks.

In this experiment, we collected 30 phishing URLs, reported from PhishTank[10]. All of these malicious URL connections elicited warnings when we tried to access them in browsers on desktop platforms such as the IOS system, with Google Chrome browser. Other browsers, such as Safari and Firefox, all used Google Safe Browsing API as a security mechanism to prevent phishing attacks [160] [161]. Subsequently, we accessed these malicious URLs on mobile platforms with different browsers. In this experiment, we used Apple's Safari browser in the IOS system, a Google Chrome browser in the IOS system and a Google Chrome browser in an Android system, because Google Chrome and Apple's Safari are the most popular mobile browsers [162]. From Statista in February 2022 [163], Google's Chrome browser and Apple's Safari browser are the leaders in the mobile internet browser market. Google's Chrome had a market share of 61.93%, and Safari had a market share of 25.56%, as shown in Figure 25 below.

[10] PhishTank: It is a collaborative clearing house for data and information about phishing on the Internet. Available at: *http://phishtank.org/*.

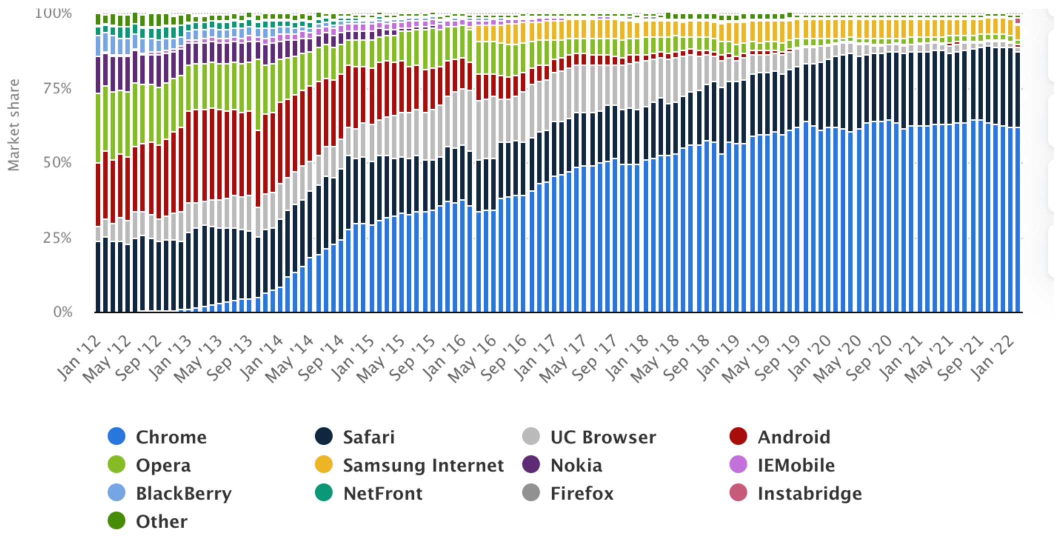


FIGURE 25: Mobile browser market share distribution in February 2022 **[163]**.

However, the positive alarm rates were not as good as expected compared to the results of desktop platform browsers, and the results were very low. According to our statistical data shown below, only 46% (14/30) of these malicious URLs were identified as a phishing attack or a risky connection when the mobile used is an Android system with the Google Chrome browser. The result on Apple's devices is subject to the vendor of the browser. On an IOS system, around 57% (17/30) of these malicious URLs were detected as True Negatives in Safari, but the Google Chrome browser accessed these malicious URLs as a benign connection without any warning notifications, and no URL was detected as either a phishing attack or a risky connection. The experiment result is shown in Figure 26 below.

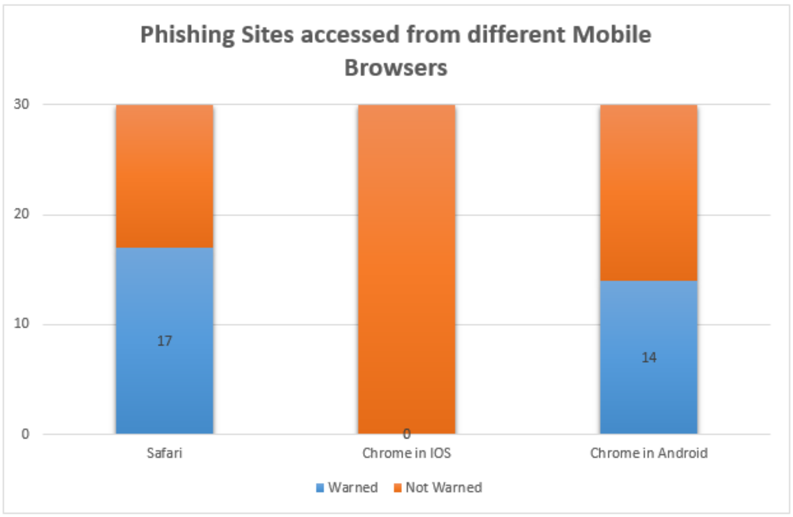


FIGURE 26: The phishing URL examination results with various browsers on mobile platforms.

There are other published research on evaluating phishing risks on mobile platforms. For example, Felt and Wagner assessed in this risk in [85]. The concept of control transfer was introduced, which is when web sites in mobile browsers should obey the standard Same Origin Policy[11] as otherwise they are potentially untrustworthy. This indicates that web sites on different domains should be isolated from each other. However, neither Android nor Apple restrict access to mobile web sites. Thus phishing attacks always occur during a control transfer, such as during a trusted inter-application link which may point to a malicious target instead.

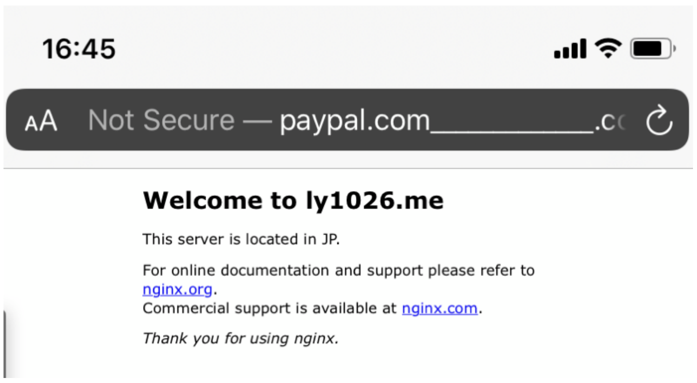
Mobile platforms have installed various security mechanisms to improve the user experience by ensuring they are under a safer environment, such as by using sandboxes, code-signing and unexpected IP address[12] connections [50]. However, there are more restrictions influencing mobile security issues. For example, Shahriar, Klintic and Clincy [94] listed ten important factors that make the mobile web different from desktop usage. Most factors are based on limited resources in mobile devices, such as screen, bandwidth, power usage and slower processes. In particular, for phishing attacks, mobile users have difficulty in verifying if a page is legitimate due to a relatively small screen, and some malicious URLs are not often completely displayed within mobile browsers.

For example, we bought two domains, named *stafocus.com* and *ly1026.me*. Assume *ly1026.me* has an A record with the IP address 45.32.49.95. We then declared a subdomain in *stafocus.com*, called *www.paypal.com______________.com.stafocus.com*, and also configured a CNAME record to point to *ly1026.me*. We then have:

www.paypal.com______________.com.stafocus.com **IN *CNAME*** ly1026.me
IN *A* 45.32.49.95

[11] Same Origin Policy: An import concept in web security. Under this policy, scripts can only access data from sites which are the same as the origin page. An origin is defined as a combination of URL scheme, host name and port number[195]. This means it ensure that the script running on a site can only access data from the origin website itself.
[12] It is hard to confirm the IP address for mobile use unless it is connected to Wi-Fi.

In this case, the usage of the redundant underline is to confuse the victim who considers that this connection is from PayPal as the browser URL bar on a mobile screen is short and this URL may only display the prefix part if the displaying order on mobile browser is from the beginning (left). Thus, when the user accesses the address on the browser with *www.paypal.com___________.com.stafocus.com*, the traffic is redirected to the domain *ly1026.me*. The URL bar will display the former website rather than *ly1026.me* due to the features of the CNAME record. Assuming that the domain *ly1026.me* is a suspicious PayPal website, a potential risk exists when the user accesses this URL using a mobile browser, as shown in Figure 27 below. (Before updating our IOS system, we used an iPhone 11, version of 13.6.1, see Figure 28. In this version, the display order on the mobile browser is from the left / beginning of domain name.)



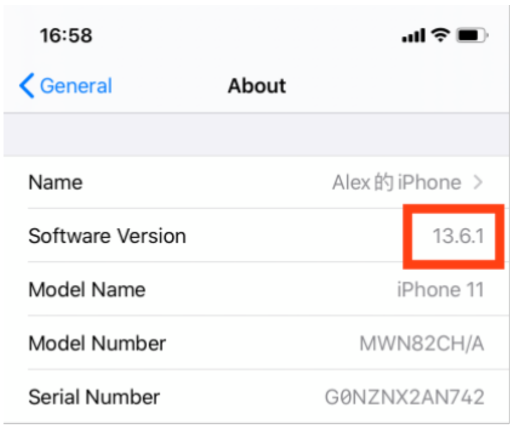
Welcome to ly1026.me

This server is located in JP.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

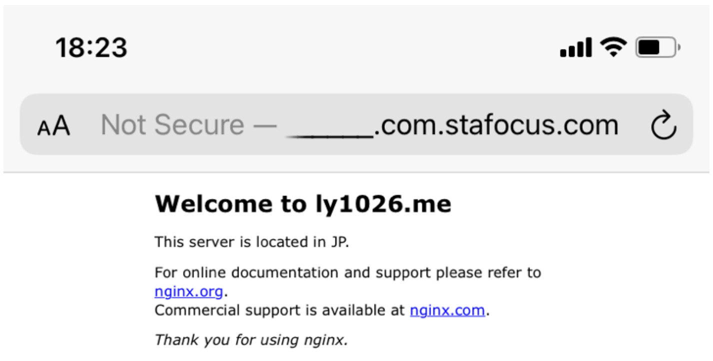
FIGURE 27: URL bar displayed result of *www.paypal.com_____________.com.stafocus.com* in IOS system version 13.6.1.



Name	Alex的 iPhone
Software Version	13.6.1
Model Name	iPhone 11
Model Number	MWN82CH/A
Serial Number	G0NZNX2AN742

FIGURE 28: iPhone general specification in the experiment (Before system update).

On mobile platforms, the URL bar displays the URL address from the beginning and ignores the remaining part of the URL address due to the limited length of screen. At first glance, this URL address can be recognized as PayPal if the user lacks security awareness. However, this potential risk has been fixed in the latest IOS system version. As shown in Figure 29, after the system version update (see Figure 30), the URL bar displays the domain part of the URL address rather than displaying the characters from the beginning, reducing the risk of incorrect recognition.


18:23

AA Not Secure — ______.com.stafocus.com

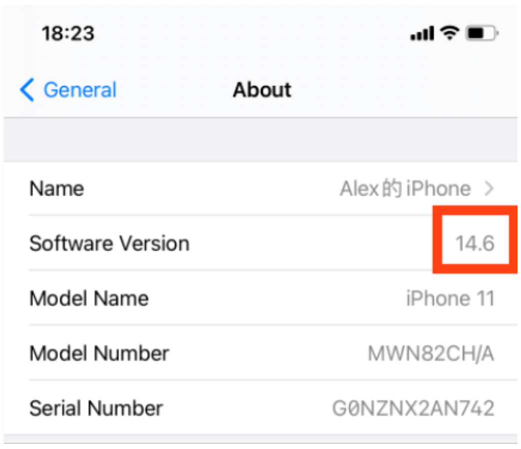
Welcome to ly1026.me

This server is located in JP.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

FIGURE 29: URL bar displayed result of *www.paypal.com______________.com.stafocus.com* in IOS system version 14.6.


18:23

General About

Name	Alex的 iPhone
Software Version	14.6
Model Name	iPhone 11
Model Number	MWN82CH/A
Serial Number	G0NZNX2AN742

FIGURE 30: iPhone general settings in the experiment - after a system update.

4.1.3. The specific research goal for general phishing attacks

In summary, the first part of this research was to investigate the mitigation of general phishing attacks. We attempted to implement OCR approach to defend against this kind of campaign. Moreover, there were several challenges faced when we implemented the related prevention mechanisms on mobile platforms as they have limited resources such as power, network bandwidth and processing. Thus, how to implement our approach for mobile users and how to reduce the resource consumption as much as possible for a mobile client will also be considered in the thesis.

4.2. Methodology and Design of Experiments

In this section, the methodology used to prevent general phishing attacks will be explained in detail. We implemented this OCR method, which allows the computer to dynamically identify these attacks without using machine learning based approach. This procedure is divided into three aspects from the intrinsic principle to the final technique adopted. Firstly, we designed how computers identify a phishing website, noting what this procedure is. Secondly, we narrated the relevant challenges faced when computers identify the phishing attacks are presented and solved. Finally, we attempted to address the resource consumption issues on a mobile platform, and the prototypes of the developed implementations on mobile platforms will then be presented.

4.2.1. How computers identify a phishing website.

For computers, the prevention of phishing is mainly based on two approaches: the blacklist approach and the content-based approach, both of which are mentioned in Chapter 3. As we know, both approaches have limitations due to various factors. Thus, we need a method that is different from current phishing mitigation techniques to defend against this kind of general phishing attack, and also overcome the existing limitations.

In explanation, we used a confirmed and reported phishing website from PhishTank, and the relevant details about this suspicious website are summarised in Table 7 below. The displayed result in the browser is shown in the following Figure 31.

Submission ID	Description	Reported Time	Suspicious URL	Reference
#7201412	PayPal login	Jun 27th 2021	See footnote[13]	See footnote[14]

TABLE 7: The details of the selected phishing website from PhishTank.

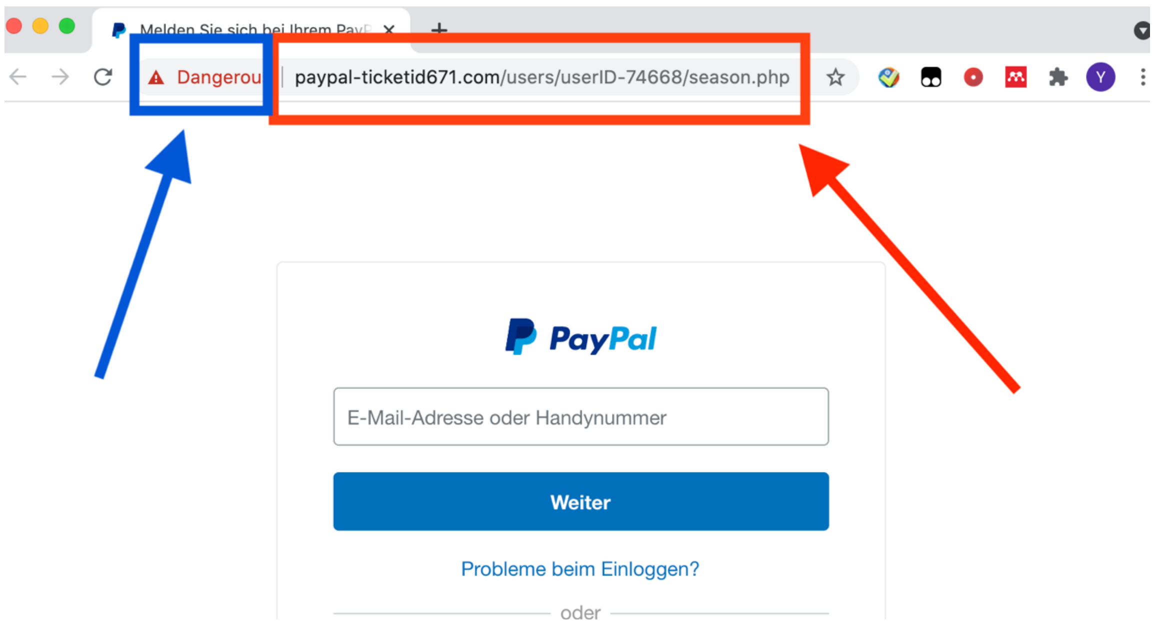

Melden Sie sich bei Ihrem PayP
Dangerou
paypal-ticketid671.com/users/userID-74668/season.php
PayPal
E-Mail-Adresse oder Handynummer
Weiter
Probleme beim Einloggen?
oder

FIGURE 31: The displayed result of the suspicious PayPal phishing website from PhishTank in a Google Chrome Browser. The red arrow points to the malicious URL, and the blue arrow points to the SSL certificate warning.

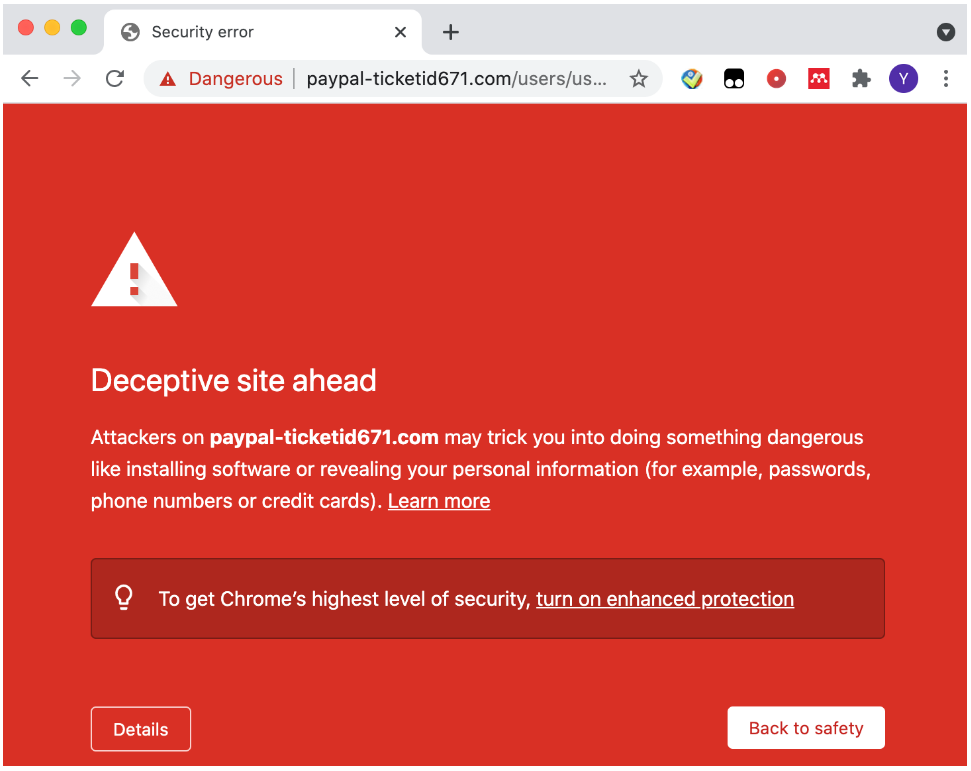
In our proposed solution, the procedure of recognizing such phishing websites by computers is divided into three steps. Firstly, from the background images or logo of this website, what the website is supposed to be and what the phisher expects from the user can be confirmed. This process is to identify the **purpose** of the suspicious website. In this case, the purpose is to gain the user's username and password from the PayPal website. Alternatively, the purpose of this suspicious website is to compromise PayPal and its users. Secondly, the official PayPal website can be identified by searching from a search engine in the browser, which is

[13] Suspicious URL: *https://paypal-ticketid671.com/users/userID-74668/season.php*, which has been reported as a phishing website in PhishTank during the experiment.
[14] Reference: *http://phishtank.org/phish_detail.php?phish_id=7201412*.

paypal.com. Subsequently, an obvious difference in domain name can be observed by comparing this accessed URL (marked by a red rectangle in Figure 31 above, showing the false domain as *paypal-ticketd671.com*) to the official URL. Finally, this accessed URL is identified as a phishing website.

(**NB**: In this procedure, the following situations are not considered:

- A warning notification from the browser, as shown in Figure 32 below.
- A warning notification from the SSL certificate, which is marked by the blue rectangle in Figure 31 above)



Deceptive site ahead

Attackers on **paypal-ticketid671.com** may trick you into doing something dangerous like installing software or revealing your personal information (for example, passwords, phone numbers or credit cards). Learn more

To get Chrome's highest level of security, turn on enhanced protection

FIGURE 32: A warning notification showing that the accessed website is suspicious is shown in a Google Chrome Browser.

Thus, for computer identification of the general phishing attack, the URL content is a major factor in distinguishing a phishing website from the official one. Using the logo or background image of the accessed website to confirm its purpose, the official URL is then determined through a search engine. The comparison of the two sites (accessed URL and official URL) determines whether or not phishing has occurred.

4.2.2. What challenges were faced in our proposed solutions

In our proposed solution, the following challenges we faced during the experiment are listed below as:

- **Challenge 1 (C1)**: How will a computer identify the purpose (activities or actions) of an accessed URL?
- **Challenge 2 (C2)**: How will a computer find the official URL according to its purpose?
- **Challenge 3 (C3)**: How will a computer identify a phishing website through the comparison between the accessed URL and the official URL?

For Challenge **C1**, humans can identify the purpose of the accessed URL from their perceptions and observation. We need to use an alternative technique in a computer to replace human perception with the recognition of the purpose of the accessed URL. According to our investigation on PhishTank, most phishing websites are about financial services and gathering email addresses, incorporating services such as a bank app, PayPal, outlook or Gmail. Most website logos consist of various characters within colour banners. Some logos have been reported frequently, as shown in Table 8 below. In order to confirm the characters in the logo, we adopted OCR technology to extract and recognize content, so that we could identify the purpose of the accessed URL.

Logo	Characters / Purpose	Industry
PayPal	*PayPal*	Online payment service
TSB	*TSB*	Bank app
Santander	*Santander*	Bank app
BANK OF SCOTLAND	*Bank of Scotland*	Bank app
HSBC UK	*HSBC UK*	Bank app
Google	*Google*	Email service
Microsoft	*Microsoft*	Email service

TABLE 8: Summary of the relatively higher exposure rate logos that have been reported frequently on PhishTank.

For Challenge **C2**, we attempted to adopt a Search Engine API[15], to achieve the goal of returning the relevant official URL (website), by using a key word search according to the previously obtained purpose of the accessed URL.

For Challenge **C3**, general phishing websites can be identified by observing and comparing the differences between the accessed URL and the official URL. However, in order to deal with more complicated situations, such as DNS hijacking where the accessed URL is the same as the official URL, we attempted to compare the SSL certificate details between the accessed URL and the corresponding official URL to identify if the accessed website was a phishing site.

The challenges and related solutions are summarised in the following Table 9:

	Identification in Computers
Confirm the purpose of URL	Extract the logo and confirm the purpose via **OCR**.
Confirm official URL	Determine the official URL via a **search engine API**.
Identify if a phishing URL	Compare the **SSL certificate** between the accessed URL and the official URL.

TABLE 9: The summary about the faced challenges and related solutions.

4.2.3. How to address the resource consumption issues on a mobile platform.

According to the computer procedure required to recognize phishing websites above, we adopted OCR technology as a mean of identifying phishing. The specific procedure was conducted via four steps, as shown below:

[15] Search Engine API: In our experiment, we adopt Google Search API. But other APIs are still applicable, it is subject to the user's present country. For example, Baidu Search API is better than Google Search API if the user is in China, as all services from Google are blocked.

1) Extract the accessed website background / logo image using a web crawler.

2) Recognize the extracted image content using OCR technology.

3) Confirm the official URL using a Search Engine, such as Google.

4) Verify the accessed URL through SSL certification comparison.

However, the resources, such as bandwidth and CPU, are greatly consumed during the first two steps. In step 1, the HTML response can be parsed from the accessed URL through a web crawler. Although the bandwidth would not be used up, some websites may contain many images or icons so that it is more difficult to locate where the actual logo image is. In our solution, we expect to extract all of the images to record these URLs. In step 2, all of these images of URLs need to be analysed through OCR technology. For example, if we adopt Google Vision API, many image requests will be sent to the Google Cloud. As the bandwidth consumption is extra, these manipulations should be executed on the server side rather than in the local device if we want to implement this approach in mobile platforms.

The ideal situation is to incorporate a plug-in script set up in a mobile browser. Before parsing the HTML response, the targeted URL address should be sent to the server side for identifying veracity. The above four steps from the OCR approach need to be implemented on the server side in order to improve the efficiency and reduce the mobile resource consumption. In order to determine the feasibility of this approach on mobiles, we simulated an environment that assumed a script in a mobile browser by using mitmproxy[16] , as shown in Figure 33 below.

[16] Mitmproxy: It is a free and open source interactive HTTPS proxy for debugging, testing, privacy measurements, and penetration testing. Available at: *https://mitmproxy.org/*.

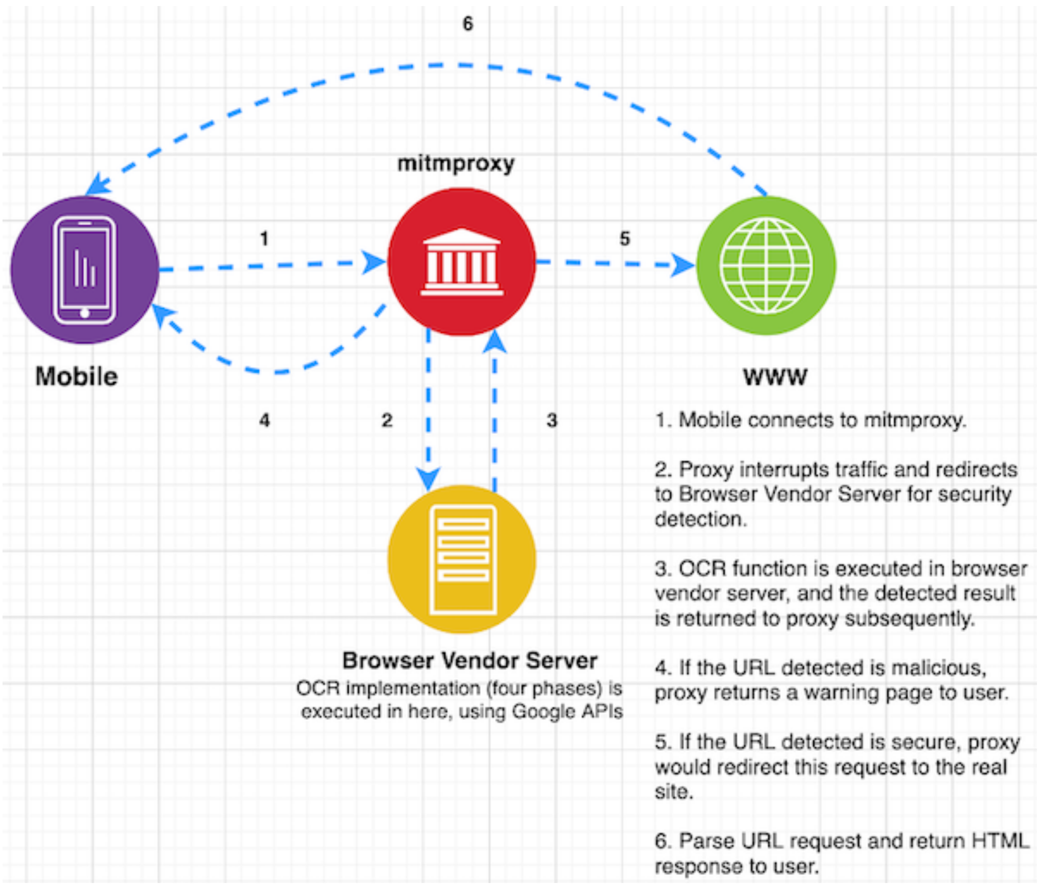

6
mitmproxy
1
5
Mobile
www
4
2
3
Browser Vendor Server
OCR implementation (four phases) is executed in here, using Google APIs
1. Mobile connects to mitmproxy.
2. Proxy interrupts traffic and redirects to Browser Vendor Server for security detection.
3. OCR function is executed in browser vendor server, and the detected result is returned to proxy subsequently.
4. If the URL detected is malicious, proxy returns a warning page to user.
5. If the URL detected is secure, proxy would redirect this request to the real site.
6. Parse URL request and return HTML response to user.

FIGURE 33: Illustration of experimental environment.

We established the connection between the mobile platform and mitmproxy. This proxy will interrupt the network traffic and redirect it to the server. The relevant manipulations regarding OCR are executed in the browser vendor server (here it was simulated by a host server) and returned a result (positive or negative) to mitmproxy. The final response is either a warning to the user's browser or a redirect request to the real site, confirmed by mitmproxy. The mobile and mitmproxy can be simulated as an extra functional browser on a mobile device. Therefore, in this environment, the resources on a mobile would not consume extra data. For the bandwidth, there are only two requests to be sent from the user's browser. One is being sent to the server in order to verify the security of a URL; the other is sent to the real site if the returned result displays the accessed URL is not a phishing site. All of the relevant OCR manipulations would be executed on the server side, rather than on the user's device, which means it would not take any extra resources consumption from the user's mobile.

We present this prototype of OCR implementation for preventing and identifying phishing websites in order to solve the resource consumption problem on mobile platforms. The applicability of this prototype will be proved in the next section.

4.2.4. Detailed research implementation.

As stated above, the first aim of this research is to identify phishing activity from accessed URLs by using OCR technology, and the applicability of our presented prototype needs to be proved in order to implement this approach on mobile platforms. The implementation of this research involves three steps:

- The comparison of various OCR APIs.
- The deployment of OCR technology for mitigating phishing attacks.
- The efficiency of the presented prototype.

4.3. Implementation

In this section, the experimental implementation includes three parts. Firstly, the OCR API was selected. Secondly, the preliminary requirements were listed and thirdly, the specific implementation was described in detail. We will narrate the detailed procedures of the OCR function before presenting a prototype designed to reduce the mobile resource consumption problem.

4.3.1. The comparison of OCR APIs

There are various vendors that provide available OCR APIs online. In our experiment, we selected OCR APIs from two leading vendors: Microsoft and Google. When we selected the OCR, we focused on their functionality and price. From the APIs overview on their official website specification, we have a preliminary understanding about their functionality and price. Next, we selected several logos from leading websites, such as Google, PayPal and other banking websites, to recognize the logo content for examining the API's result. For example, we expected to obtain the string "Google" or "google" from the Google logo. This result is defined as high precision. Otherwise, we define the result as unexpected, such as the result consisting of an incorrect word spelling. According to their specifications, the details of both APIs are listed in Table 10 below.

Vendor	Functionality	Price
Microsoft[17]	Supports *JPEG*, *PNG*, *GIF* and *BMP*.	5000 free per month, extra charge after that.
Google[18]	Supports *JPEG*, *PNG*, *GIF*, *BMP* and *SVG*.	Free up to 1000 per month, $1.5 per thousand beyond that.

TABLE 10: The details about Microsoft OCR API and Google OCR API.

Both OCR APIs have the required functionality, and the purpose of an image can be detected and recognized. However, we found the recognized results from the Microsoft OCR API are not as good as expected in some scenarios. For instance, if the logo text is located in a dark background, as shown in Figure 34 below, the recognized results were not as good as expected. Therefore, we decided to adopt the Google OCR API to conduct the further experiment in this part of the research, although the charge is relatively higher with only 1000 free query requests per month.

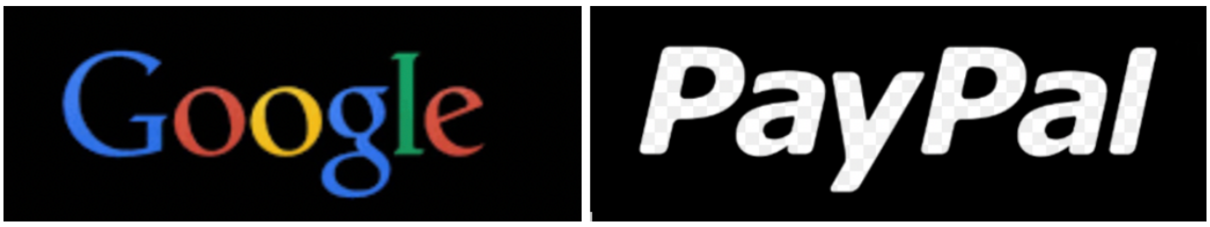


FIGURE 34: Microsoft OCR API does not work in cases in which the logo text is located in a purely dark background.

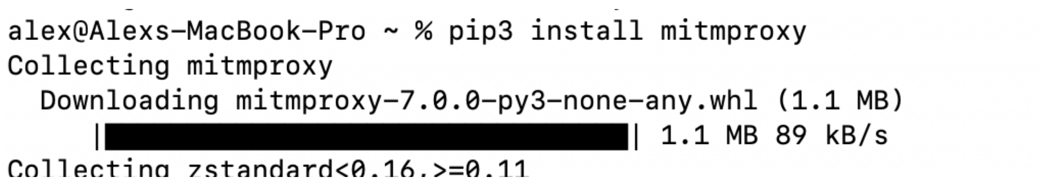
[17] More details about Microsoft OCR API please see the link: *https://azure.microsoft.com/en-in/services/cognitive-services/computer-vision/*.
[18] More details about Google OCR API please see the link: *https://cloud.google.com/vision/*.

4.3.2. Preliminary Requirement

In order to check the feasibility of the presented methodology, a python program was written. This program involves extracting logos, recognizing text within logos, searching related official websites and verifying any malicious website. The following open-source APIs need to be preliminary registered, and then enabled in the local server:

- Google Optical Character Recognition (OCR) API.
- Google Search API.

Also, the installation of mitmproxy was necessary in order to interrupt the network traffic for the further experiment. In our experiment, the machine used was a MacBook Pro and we used **pip3 install mitmproxy** to install it, as shown in Figure 35 below:



```
alex@Alexs-MacBook-Pro ~ % pip3 install mitmproxy
Collecting mitmproxy
  Downloading mitmproxy-7.0.0-py3-none-any.whl (1.1 MB)
     |████████████████████████████████| 1.1 MB 89 kB/s
Collecting zstandard<0.16,>=0.11
```

FIGURE 35: The installation of mitmproxy on MacBook.

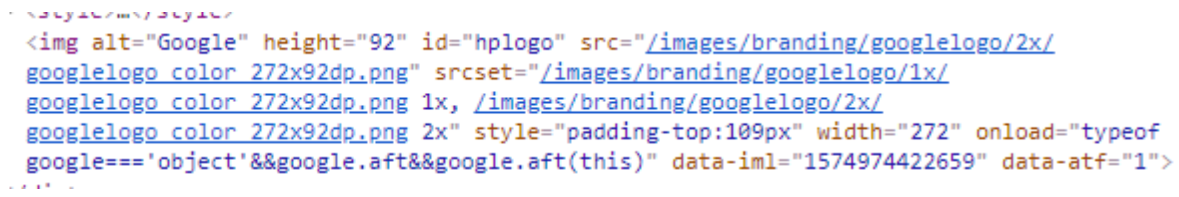
4.3.3. Implementations

The OCR Function Overview

Firstly, we narrated the implementation procedure of OCR approach, which includes four steps as follow:

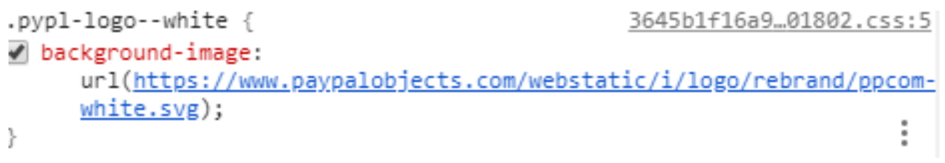
1) *Parse the given URL and extract related images:*

In order to confirm the purpose of the website, the first step was to use a web crawler to extract the related images, such as the logo or a background image. Due to the complexity and diversity of phishing websites, the logo image may be stored in different places. Generally, there are two methods to link the logo or background image; HTML code and .css files. In the first approach, the logo image is usually inserted under the <img> tag in HTML code, such as with *Google.com* or *Microsoft.com*, as shown in Figure 36 below. In the second approach, the logo image has been linked via a .css file, under the attributes of background or background-img, such as occurs in *PayPal.com*, as shown in Figure 37 below. Therefore, in order to cover both possibilities, HTML code and .css files need to be traversed. In this step, all of the images will be collated and stored for later recognition.



```
<img alt="Google" height="92" id="hplogo" src="/images/branding/googlelogo/2x/
googlelogo_color_272x92dp.png" srcset="/images/branding/googlelogo/1x/
googlelogo_color_272x92dp.png 1x, /images/branding/googlelogo/2x/
googlelogo_color_272x92dp.png 2x" style="padding-top:109px" width="272" onload="typeof
google==='object'&&google.aft&&google.aft(this)" data-iml="1574974422659" data-atf="1">
```

FIGURE 36: The Google logo path, stored in HTML code under the tag of <img>.



```
.pypl-logo--white {                              3645b1f16a9…01802.css:5
✓ background-image:
      url(https://www.paypalobjects.com/webstatic/i/logo/rebrand/ppcom-
      white.svg);
}
```

FIGURE 37: The PayPal logo path, stored in CSS code under the attribute of background-img.

2) Recognize the image content:

In this step, the Google Optical Character Recognition (OCR) API is called to recognize the content of the extracted image from step 1. This API only works for detecting specific text from an image, which means the result from a redundant image, such as a symbol icon from the .css file, scenery or character images from HTML <img> tags, is worthless, and will be withdrawn. All of a logo's content would be stored and moved to the next step. The aim of this step is to obtain the purpose of the accessed website, for example, if the accessed website is about a PayPal login page, we have already extracted the logo from step 1. Next, we use Google OCR API to recognize this logo, and gather the text content "PayPal", and we then identify the purpose of this accessed website as "PayPal", and we will execute the next step in the implementation according to this recognized result. However, an exception should be noted here, the program will be terminated if there is nothing detected from the images via the OCR API. For example, some websites may use a full screenshot from the official website as a phishing page. If the phishing websites used a screenshot to fill up the whole page, it was too much information for the API to process and the recognized result returns an error.

3) *Search and confirm the official website URL:*

From the description of the image content above, we can identify the expected purpose of this website. In this step, the related official URL address will be obtained by using a keyword search in the Search Engineering API. In our experiment, according to these keywords, the associated search information is responded through Google Search API. We kept the top three results from the returned list as a legitimate website will normally come up within the top results in a Google Search due to its high PageRank [119]. Alternatively, a phishing website would not be indexed in the top search results due to its low PageRank as this malicious website is available online for a short amount of time only [119]. For example, in one of our experiments, "PayPal" text was recognized from a phishing PayPal website, and the related official URLs were confirmed by using a Google Search using this text, as shown in Figure 38 below.

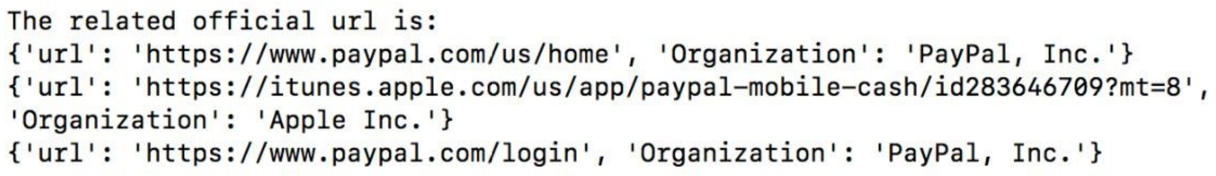
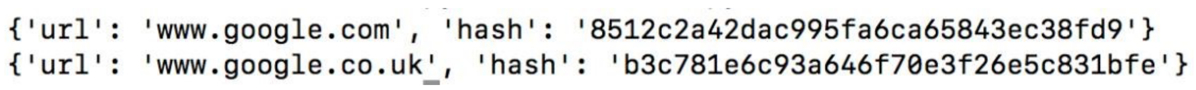

```
The related official url is:
{'url': 'https://www.paypal.com/us/home', 'Organization': 'PayPal, Inc.'}
{'url': 'https://itunes.apple.com/us/app/paypal-mobile-cash/id283646709?mt=8',
'Organization': 'Apple Inc.'}
{'url': 'https://www.paypal.com/login', 'Organization': 'PayPal, Inc.'}
```

FIGURE 38: Search result from the keyword 'PayPal' in our experiment.

4) *Verify the transmission security of parsed URL:*

At the end of 2017, Google announced they would flag all unencrypted traffic in order to make their user feel more secure on the internet. They used Secure Sockets Layer (SSL) to improve the security of traffic transmission. SSL encrypts traffic so data is private under an established channel between a browser and a server [164]. So far, most websites have already deployed SSL to establish the connection. Thus, we adopted SSL certificate information to verify the security of the URL. In this step, we retrieved the SSL certificate "**issued by**" and "**issued to**" attributes to confirm whether the accessed website and official website show the same result. We attempted to use the SSL thumbprint (the hash value of SSL certificate) to verify the result in the initial experiment, but the results were not ideal. The hash value is inconsistent if the websites are located in different enterprise branches. For example, as shown in Figure 39 below, the SSL certificate hash value is different between *https://www.google.com/* and *https://www.google.co.uk/*.



```
{'url': 'www.google.com', 'hash': '8512c2a42dac995fa6ca65843ec38fd9'}
{'url': 'www.google.co.uk', 'hash': 'b3c781e6c93a646f70e3f26e5c831bfe'}
```

FIGURE 39: Google SSL hash comparison between *.com* and *.co.uk*.

Therefore, the complete procedure to identify phishing sites using OCR implementation consists of the four steps above, from the extraction of the logo to recognizing its content to the comparison of SSL certificates between the accessed website and the official website. We now present an overview of a prototype to avoid the resource consumption issue on mobile platforms. The detailed implementation included four phases;

- Connect to mitmproxy.
- Interrupt traffic and redirect to the test server.
- Implement OCR functions in the test server.
- Execute the response from mitmproxy.

The overview of user network traffic that accesses the target website is illustrated in Figure 40 below. When the user wants to access the target website, mitmproxy would interrupt this traffic (see step 1) and redirect the target website URL to the cloud server (see step 2) for phishing security detection. The detected result is either positive or negative (after implementing an OCR approach on a Cloud Server) and is returned to mitmproxy (see step3). Finally, mitmproxy redirects the previous user request to the target website if the detected result is safe (see step 5), otherwise, mitmproxy will respond with a warning notification page to the user's browser if the detected result is malicious (see step 4). In this notification page, the system will provide two options to the user, "Block this traffic" or "Access this traffic". If the user selects the option of "Block this traffic", this URL will be blocked and discarded. If the user insists on accessing this URL, selecting the option of "Access this traffic", the traffic will be redirected to the target website (see step 5).

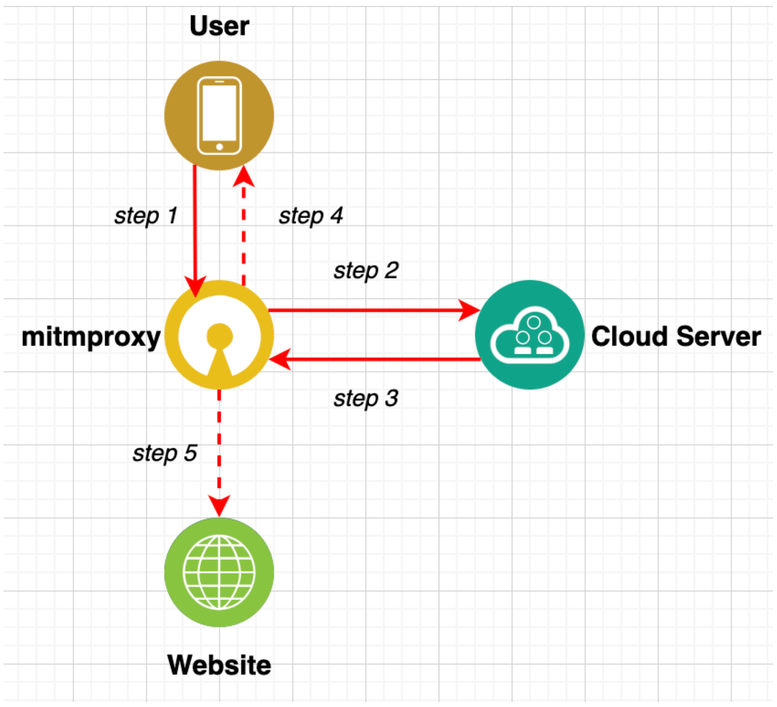
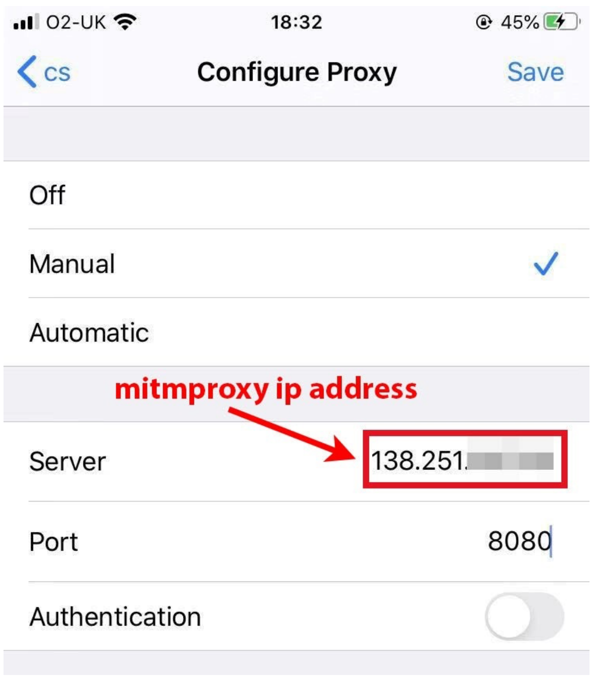


FIGURE 40: The overview of network traffic when the user wants to access a website on the browser in our mobile experiment.

The Implementation of Presented Prototype on Mobile Platforms.

Phase 1, Connect to mitmproxy.

Both the mobile device and mitmproxy require to be in the same network (Wi-Fi), to establish a connection between the mobile and mitmproxy. In our experiment, we used an iPhone 7 to connect the proxy, and this proxy was listening on port 8080. The configuration of the mobile phone is attached as shown in Figure 41 below indicating that the proxy is on the St Andrews network.


O2-UK 18:32 45%

cs Configure Proxy Save

Off

Manual ✓

Automatic

mitmproxy ip address

Server 138.251.[redacted]

Port 8080

Authentication

FIGURE 41: Network configuration on the mobile phone.

Phase 2, Interrupt Traffic and Redirect to Server.

A python program was written in mitmproxy for executing further traffic interruption. The function *request()*[19] and *response()*[20] are used to handle the traffic flow. The proxy retrieves the URL address from the request that is sent by the mobile. Secondly, this mitmproxy redirects the URL address to the cloud server (it can be a browser vendor server, in our experiment we used my server instead of the cloud server) to verify the security of the accessed URL through the OCR function. In our experiment, we set up mitmproxy and the server in the same machine (a MacBook Pro 2016, i5 Processor, 15GB Memory), an illustration is shown in Figure 42 below. Finally, this proxy will execute different responses according to the detected result from the server. This will be explained further in Phase 4.

[19] Mitmproxy *request()*: A built-in function in mitmproxy, which is used to handle the request traffic from the client before sending to the destination server address.
[20] Mitmproxy *response()*: A built-in function in mitmproxy, which is used to handle the response traffic from the server before arriving the client.

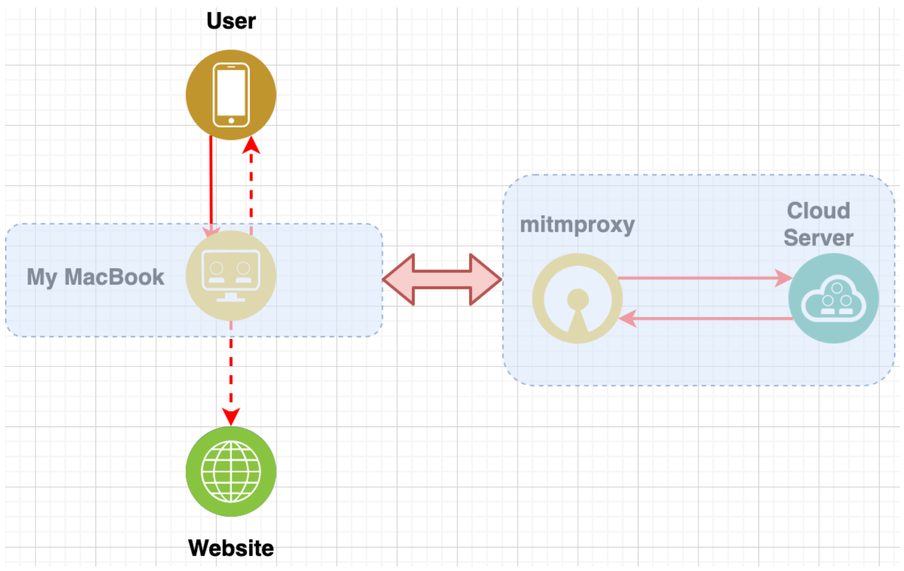


FIGURE 42: The architecture of experiment.

Phase 3, Implement OCR Functions in the test Server.

Once the server (machine) receives the URL request, the four steps OCR manipulation would be triggered.

- *1) Parse URL and extract all images:* After receiving the URL from the mitmproxy, the relevant HTML response would be parsed, and the web crawler used to extract the logo image.
- *2) Recognize image content:* In this step, the Google OCR API would be called to recognize the content of the extracted image from the last step.
- *3) Search official website URL:* In this step, the official website address from the parsed URL would be identified according to the analysed logo recognition result from the above phase by using Google Search API.
- *4) Verify the security of the parsed URL:* In this step, we retrieved the SSL certificate 'issued by' and 'issued to' attributes to verify the accessed website.

Subsequently, the detected result is returned to mitmproxy at the end of this phase.

Phase 4, Execute Response in mitmproxy.

According to the received result, this proxy would execute different responses. If this URL is verified to a malicious link, this proxy would not redirect this URL request to the real site but would return a warning page to the user's browser directly. Alternatively, a redirect request to the real site is made and the parsed response is returned to the user's browser. In this warning page, the system will provide two options to user, "Block this traffic" and "Access this traffic", the user can either access this URL or block this URL according to their personal decision.

4.4. Evaluation

In this section, the evaluation of our OCR approach and prototype implementation on mobile platforms for preventing general phishing attacks is discussed. Firstly, the applicability of the presented OCR approach is proved. Next, the presented prototype implementation for preventing general phishing attacks is verified on mobile platforms.

In order to detect the applicability and accuracy of our approach, we selected several malicious URLs which were detected and reported as phishing websites from PhishTank, and all of these URLs satisfy the following conditions:

• All of these URLs are live during the process of detection.

• Their logo must contain text, otherwise an API would not be able to return an available description back to the user.

• The content of the logo must be in English, as the current OCR API version only works on the loading of English in the prototype. Currently the returned description result of the logo may not be accurate if the logo is in other languages.

In the preliminary testing, we randomly collected 100 different phishing URLs from PhishTank. We gathered these URLs on different days and at different times of day, and ensured that these URLs were live (active) during the detection periods. Before we collected the phishing URL, we accessed this URL via Chrome browser manually on a desktop platform. If this URL was accessible and responded to a relevant malicious webpage, we defined this URL as live and gathered this URL for the further detection. Otherwise, if the displaying result of this URL is not accessible, such as a server 404 page, or redirected to another official page, then this URL was discarded. We found most of the reported phishing URLs in this blacklist were about financial accounting with 85% (85/100) sites being financial sites such as PayPal, American Express, NatWest, etc. In this 85%, the highest proportion was PayPal, at 81% (69/85).

Under our approach, we found 96% (96/100) phishing URLs were successfully identified, and only four URLs were not identified. We analyzed these failures and the main reason was due to an inaccurate result generated from the logo extraction phase. This means the purpose of the website in Step 2 (recognize the image content) would be incorrect if the logo image was extracted unclearly or incorrectly in Step 1 (parse the given URL and extract related images). When we further analyzed the results, the specific reasons were listed as follows:

- The logo was presented in text rather than by image, so it is quite hard to clearly identify this by using a web crawler (as it is difficult to identify which part of the text presents as a logo).
- In some lower quality designed phishing websites, the whole page was presented in an image, so there was too much detail for the OCR API to process

Undoubtedly, OCR technology is the crucial part in this method and the accuracy of the final result is subject to recognized details. At first, we conducted a comparative experiment to select the optimal OCR API. We noted that the Microsoft (Azure) OCR API is a good option as well, though the testing result was not as good as expected. As we stated with a comparative experiment between Google OCR API and Microsoft OCR API, the Microsoft API would not work under some special cases, such as a logo with a dark background colour. Also, it only supported the JPEG, PNG, GIF and BMP image formats. Some websites have used SVG files as the logo image format, thus, we switched to the Google OCR API in our implementation, which overcomes all the limitations of the former approach.

However, as all of the evaluated phishing websites were reported in PhishTank, this does not mean our mitigation approach is suitable for detecting the zero-day phishing websites. Thus, we designed new (test) phishing websites and put them in our purchased servers (with domain names of *stafocus.com*, *ly1026.me* and *allways-cam.com*) and the university server (with domain of *yw43.host.cs.st-andrews.ac.uk*) separately to simulate the zero-day attacks. As most phishing websites are about financial websites, the self-designed tested websites were copies of PayPal, Santander, Bank of Scotland, Microsoft Outlook, Google Gmail, Barclays, TSB, HSBC, Amazon, eBay, Facebook and Bank of America. The evaluation results were satisfactory, as all of these tested websites were identified as phishing websites. Also, we used our approach to check their related legitimate websites, and all legitimate websites were verified as the correct website without reporting the site as a phishing website. The evaluation results were summarized in Table 11 below.

	Identified as Phishing	Total Tested
Phishing websites (from PhishTank)	96	100
Phishing websites (own designed)	48	48

TABLE 11: The evaluation results about detecting reported and self-designed phishing websites.

Next, we evaluated our prototype implementation on mobile platforms, and we focused on the following aspects:

- The applicability (whether the solution is work) and accuracy (successful True Positive).
- The effectiveness of this prototype implementation.

Firstly, the applicability and accuracy were evaluated. We adopted the same 100 malicious websites as before to determine the phishing verification result. On mobile platforms, either with IOS or Android system, we obtained the same results, 96% (96/100) phishing URLs were successfully identified and the user was shown the relevant warning notification. This is because the implemented OCR API and our analysis would not be affected under different operating systems, either in a desktop or on a mobile platform. The successful detection of phishing is subject to the API, and the result would be different if we change to other OCR APIs.

Secondly, we conducted other experiments to prove the effectiveness of this prototype on the tested mobile platforms, this includes two aspects:

- How many network requests were sent from the mobile platforms during the detection process?

- How much phishing detection has been improved for users between before prototype implementation and after prototype implementation on mobile platforms?

Regarding network requests, in our prototype, there are not many network requests sent in the detection process. During this detection phrase, we did not develop a functional browser that required the OCR approach to prevent general phishing attacks. We deployed mitmproxy to simulate this environment. We established a connection between a mobile and a desktop, with mitmproxy also installed on the desktop. This desktop plays the role of a proxy that forwards a mobile's traffic to the Internet, which means any traffic from the mobile will go through this desktop (and mitmproxy) before arriving at the target website. Thus, we treat the mobile and the desktop mitmproxy as an entire functional system. Mitmproxy can also be imagined as a functional script that uses the OCR approach for preventing a general phishing attack on a mobile browser. The mobile browser only sends two extra network requests to the Internet if the accessed website is a benign website. Under our scenario, the first request is sent to a cloud server from the mobile side via mitmproxy. Although there are many requests sent during the four steps of the OCR approach (in particular, the extraction of logos using web crawler, and recognition of the logo's content using OCR API), all of these manipulations are implemented on the cloud server side, which means extra resources (such as network, power CPU) are not consumed on the mobile user side. Subsequently, mitmproxy will send the second request to the targeted website after receiving a benign detection result from the cloud server side. The flow of network traffic is illustrated in Figure 43 below.
(**NB**: The blue line shows the first request, from the mobile user's browser to the cloud server; The red line is the response, that is the OCR detection result, back to the mobile user's browser; The green dotted line is the second request from the mobile user's browser to the target website.)

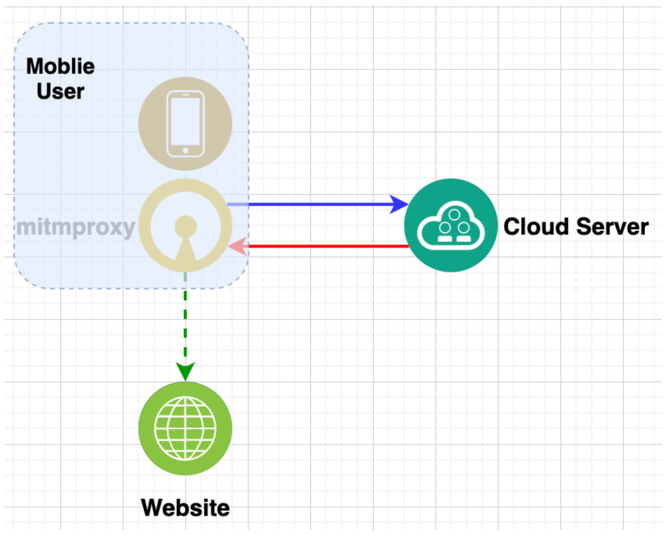


FIGURE 43: The flow of network traffic when the accessed website is benign.

Under second scenario, if the accessed website is a malicious phishing website, the mobile browser may need to send two extra network requests to the Internet. The first request is the same as the first scenario, this request is sent to the cloud server for implementing an OCR approach from the mobile use's browser. The mitmproxy will display a warning notification on the user's browser after receiving the detection result from the cloud server side if the accessed website is malicious. Thus, the second request is subject to the decision from the mobile user. If the user insists on accessing this malicious website, the mobile side will send the second request to the target website. Otherwise, further access will be denied, and the flow of network traffic is illustrated in Figure 44 below.

(**NB**: The blue line is the first request, from the mobile user's browser to the cloud server; The red line is the response that is the OCR detection result sent back to the mobile user; The green dotted line is the second request from the mobile user's browser to the target website, if the user insists in accessing this malicious website, the green dotted line shows connection, otherwise, the mobile user's browser would not send this second request and access is denied.)

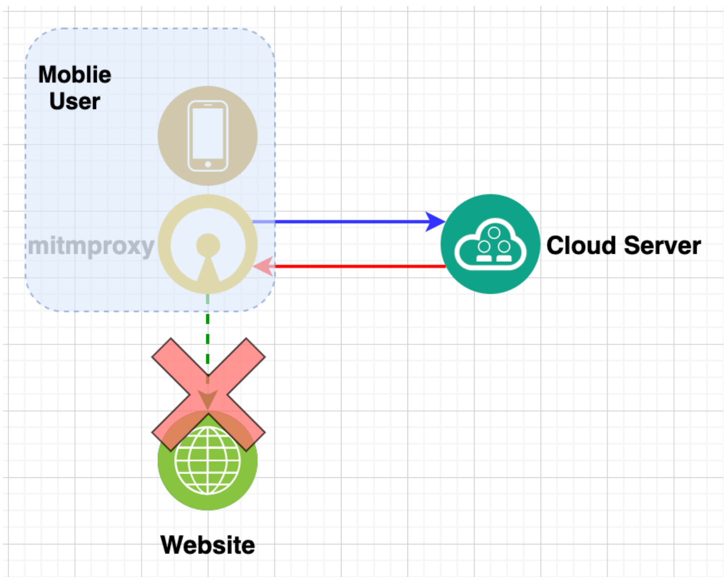


FIGURE 44: The flow of network traffic when the accessed website is malicious.

Regarding the improvement of detecting phishing attacks, we adopted the same URLs to compare the results of our method with the existing most popular mobile browsers (Safari on IOS is currently the most popular mobile browser [162]) between before and after implementing this prototype. We picked another 20 malicious URLs randomly from PhishTank. Prior to implementing our presented prototype, only 40% (8/20) malicious URLs resulted in warnings, and the rest of the URLs, 60% (12/20), could be accessed without any notifications. However, after implementing our presented prototype, the OCR approach was used to analyze accessed websites. In these 20 malicious URLs (the collection approach of these 20 phishing URLs is the same as the method of gathering the 100 malicious URLs above), 90% (18/20) malicious URLs resulted in warnings, and 10% (2/20) URLs bypassed the security detection without any notifications. Therefore, in our approach, the security has a significant improvement, from 40% (8/20) up to 90% (18/20). The comparison is summarised in Figure 45 below.

(**NB**: According to our previous experiment on phishing prevention comparison between an Android system and an IOS system, the result shows that the Safari browser on an IOS system has the highest security protection against phishing attacks compared to other browsers. Thus, we selected Safari to conduct this comparative experiment.)

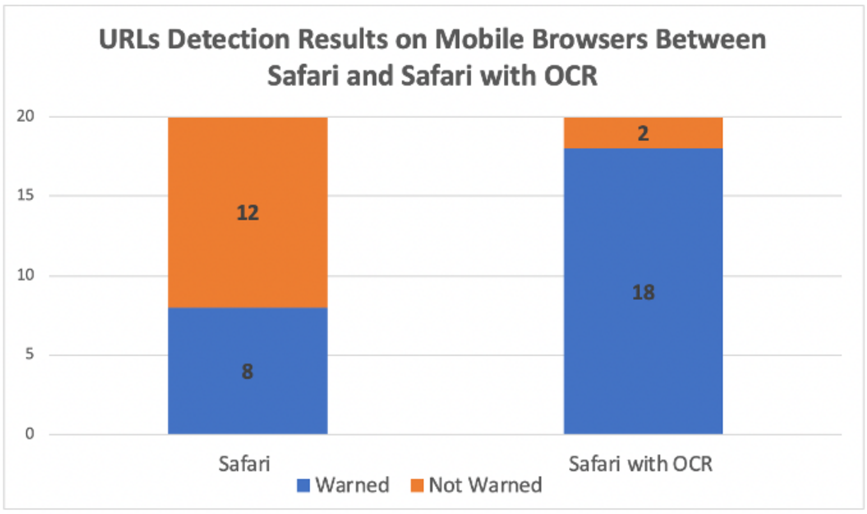


FIGURE 45: 20 malicious phishing URLs accessing results on Safari browser before and after implementing our solution.

4.5. Discussion

According to our experimental results in the evaluation chapter, our approach involves four steps, these are extracting the logo image, recognizing the content, searching the corresponding legitimate website, and verifying the security from the SSL certificate. The evaluation experiment shows that our solution not only demonstrates a high accuracy of phishing detection, but also deploys easily on both desktop and mobile platforms. However, we still face few potential challenges that need a further discussion.

- **The accuracy of the logo image extraction**

The accuracy of our approach is subject to the logo textual content on the target website. If the extracted logos are either incorrect or cannot to be recognized by OCR, the final detection result is worthless. In various versions of phishing websites, a phisher may duplicate the source code from the legitimate website, and therefore the generated phishing website has a high visual similarity to the legitimate website. In this case, our approach can accurately locate the logo image and recognize its content, although this process has a small time consumption. However, if the extracted logos either do not include any textual content or include too much textual content, the recognized results return a failure, which is not as good as we expected. For example, during our evaluation experiments, one failure case is if the whole phishing website page is designed as an image, so that there is too much textual content in the image. So, the logo image extraction is the crucial process in our solution as its results affect the accuracy of final detection.

- **The cost of OCR API**

There are various APIs which can be deployed to recognize the text from the image. However, not all of them are free. There is a free quota per month, such as 5000 times free usage in Microsoft OCR API, 1000 times free usage in Google OCR API (for more detail see Chapter 6.1.1), but any extra checks need to be charged for. Thus, our solution is suitable for individual research and experiment due to the limited free OCR usage in daily. If you want to process on a large industrial level, for example, a browser vendor or a security vendor launching a product that calls this OCR solution to prevent phishing attacks, then the cost of the OCR API should be considered. Alternatively, a free OCR API developed by a security developer would be a simple solution.

- **The efficiency of the system**

Phishing websites can be varied and complex; a phishing URL may store a lot of images or .css files, which are website content style sheets. If we have this situation, the computing process may be much longer, either for the web crawler, or the image recognition process. For example, in our experiment, in most phishing websites it is easy to locate the logo path, recognize the logo and compare to the official website, so the entire process is less than 5 seconds on average. However, in some cases, there are a lot of images obtained from the HTML webpage code and *.css* file (some have many *.css* files), such as when we collect the logo using a web crawler. Every image obtained in either HTML code or *.css* file needs to be considered, which means the recognition time using the OCR API is increased at the same time, and some cases take over 20 seconds in the entire phishing detection process.

Along with the above limitations, some details during the procedure still need further discussion; these include the selection of a search engine, the selection of searched results from the chosen search engine and the structure of warning messages if the accessed website is malicious.

- **The selection of search engine**

Our solution is vulnerable to attacks on Google's search service because we deploy the Google search engine to gather the legitimate website details according to the recognized text from the logo image. However, the gathered result may be incorrect if, for example, the phishers attack the Google PageRank algorithm and force a returned list of sites to include the malicious site [119]. Therefore, in order to overcome this challenge, we recommend decreasing the dependence on a single search engine, by implementing multiple search engines to prevent these attacks, although this will increase the time consumption.

- **The selection of searched results from search engine**

In our solution, we selected the top three results from the Google Search API, as a legitimate website will normally come up in the top results in a Google Search due to its high PageRank [119]. Alternatively, a phishing website would not be indexed in the top search results due to its short lifecycle [119]. During our evaluation, all legitimate websites can be successfully identified from the top three results. This option is adjustable, this value can be extended to more than three in order to improve the integrity and complexity of the detection process. But the time consumption is increased with the extending of searched results. According to our experiment, the top three results is a fair value, not only providing a high accuracy, but also keeping a high efficiency.

- **The structure of warning message if the accessed website is malicious**

In our experiment, we did not block the malicious website directly as our approach still has a false alarm rate, although the possibility is rare. We supported two options in the warning messages, "Block this traffic" and "Access this traffic", and the user can either access this URL or block this URL according to their decision.

Furthermore, two more challenges need to be discussed when we deploy the OCR approach on mobile platforms, and these are:

- How to implement this OCR approach for the user?
- How can the consumption of mobile resources (such as CPU, network bandwidth and power) be minimized?

In our presented prototype, both challenges are eliminated. The mitmproxy can be simulated as a browser plug-in (script), when the user accesses a website on a mobile browser, the requested URL needs to be interrupted in this (plug-in) script on the browser before forwarding to the Internet. This URL will be redirected to the browser's server (cloud) and the relevant four steps of the OCR approach are conducted on this server side. The detected result is returned to the script on the mobile browser. Finally, the script will execute different responses according to the received result. If the result shows that the accessed URL is benign, the script will forward this request to the destination address on the Internet; otherwise, the script will respond with a warning page back to the user's browser for notifying them the accessed URL is malicious. The further manipulation either insisting or rejecting accessing this website is subject to user's decision on the mobile side. Thus, most manipulations that consume resources are deployed on the server. On the user side, only one request will be sent to the browser's server before accessing the targeted address. The consumption of mobile resources, such a CPU, network bandwidth and power are minimized as much as possible.

Compared to existing phishing mitigation approaches, our OCR approach has several advantages. Firstly, we provide a dynamic method to detect the phishing attacks. A list-based approach is a static method, as it requires a large human effort to maintain the blacklist, and the accuracy of detection is subject to the integrity of the list. Zero-day attacks can't be detected, but blacklists can be updated once such an attack can be identified. Our solution overcomes these limitations as the system launches an active detection to detect the target website without requiring any human resources.

Secondly, our approach provides multiple detection, not only mitigating general phishing attacks, but also preventing DNS hijacking attacks (we proved this in the next Chapter). Most phishing detections, such as machine learning based approaches, can successfully identify a general phishing attack, but there is not a good option for preventing DNS hijacking attacks, as both URL address and webpage content between the legitimate website and phishing website are same, which results in the features from the URL and webpage having no differences. For example, as mentioned in the Literature Review, Chapter 3, the studies [28] [108] [122], selected the features from URLs and webpages. The predicted results are not as good as expected if the accessed website is a phishing website that is under a DNS hijacking attack, because the features gathered from the accessed website have a high similarity with the corresponding legitimate website, so that affects the prediction results.

Thirdly, our approach is easy to deploy that does not use machine learning, and is available on both PC platforms and mobile platforms. Moreover, our proposed prototype on mobile platforms would not result in extra resource consumption as most of the computation process is executed on a server. In a machine learning based solution, the result is predicted according to the features content. The process of features collection from the target website requires a large time consumption, and this consumption grows with feature complexity and quantity.

Compared to the GoldPhish system, proposed by Dunlop, Groat and Shelly [119], our solution is more efficient and accurate. Firstly, GoldPhish takes the screenshot from the browser with the resolution of 1200 * 400 pixels, and then recognizes the text from this screenshot. The processing time is subject to the screenshot size, which means too much information will be input to the search engine in the next step and results in a higher time consumption, although the system only focuses on the top position of the screenshot. In our approach, we extract the logo image by using a web crawler that focuses on the image only, which reduces the information redundancy before being input to the search engine, and this results in the improvement of efficiency. Secondly, GoldPhish identifies the phishing website by comparing the top-level and second-level domains between the legitimate website that was obtained from the search engine and the accessed website. However, a false positive (identifying a legitimate website as a phishing website) occurs when, for example, the accessed website is *google.co.uk* and the obtained legitimate website from search engine is *google.com*. Because the top-level *co.uk* is different to *com*. In our approach, this problem does not exist, and the detection accuracy is improved as we identify the phishing website by the comparison of "**issued by**" and "**issued to**" from the SSL certificate between the legitimate website and the accessed website. Both content of "**issued by**" and "**issued to**" are the same if the websites belong to a same enterprise, even if there are under a different branch.

5. Research on Advanced Phishing Attacks

In this chapter, the current state of advanced phishing attacks is fully researched. Many users are sensible about suspicious activity (abnormality) from suspicious URL addresses or obvious warning information from browsers [28] [29] [30]. However, these abnormalities will be eliminated through implementing a technical mean, for example, if the victim is under a hijacking attack, either through a DNS (Domain Name System) hijacking attack or by an ISP (Internet Service Provider) hijacking attack. In advanced phishing attacks, both the URL address and the website content are the same as the related official site. We firstly illustrate the principle of both DNS hijacking attacks and ISP hijacking attacks from technical aspects. The relevant methodologies used to detect DNS hijacking attacks and ISP hijacking attacks are separately elaborated in subsequent sections. Then, the process of deriving prototypical systems along with the relevant technologies will be introduced and explained in detail. Finally, the evaluation is narrated, and the corresponding challenges faced are analysed and discussed.

5.1. Related Works & Pre-Experiments

In this section, we describe the principles and workflow for this kind of advanced phishing attack. We focused on two aspects, DNS hijacking attacks and ISP hijacking attacks. At the end of this section, the specific research goal for advanced phishing attacks will be presented according to the results of our reviewed literature.

5.1.1. Reviews on DNS hijacking attacks

We start by describing DNS hijacking attack procedures and how to launch a DNS hijacking attack. Next, current DNS hijacking prevention approaches and their limitations will be discussed.

The procedure of DNS hijacking attacks

The Domain Name System (DNS) is the core technology that directs users to target websites on the internet [165]. By querying DNS records on the DNS server, the appropriate IP address can be located and returned, and the HTML code will be parsed and displayed to the user. An illustration of DNS resolution is shown in Figure 46 below, with the explanation following.

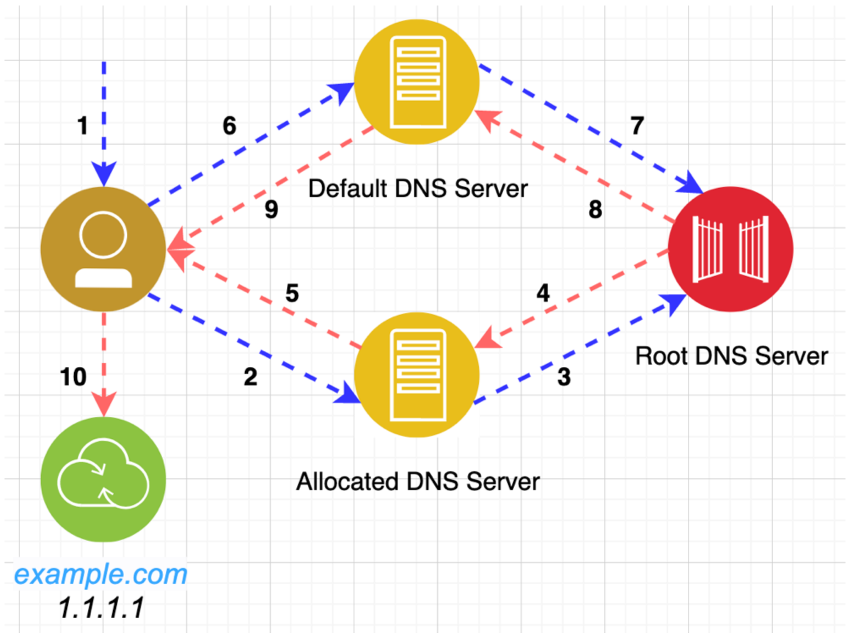
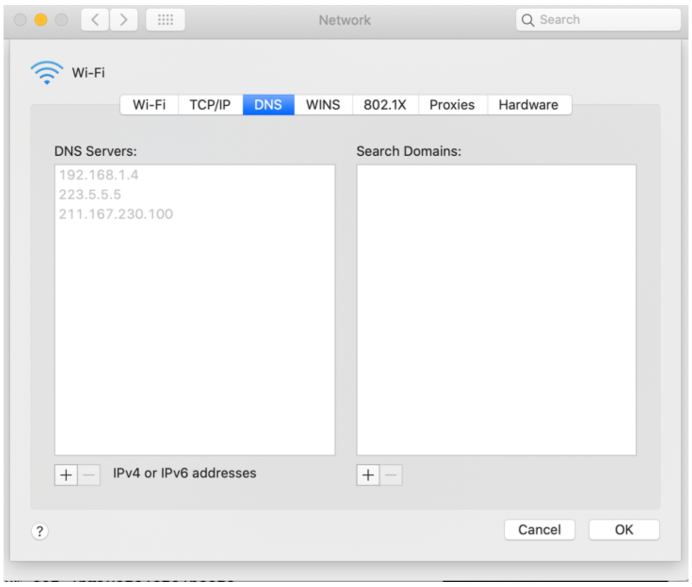


FIGURE 46: An illustration of the DNS resolution of example.com with IP address 1.1.1.1.

Assume a user wants to access a website on their browser with the URL of *example.com*, where the IP address is 1.1.1.1. The procedure involves the following steps:

Step 1: Firstly, the user's browser queries the local DNS mapping table. If the local DNS table has the record of the IP address of the target website (in this case, it points to *example.com*), the browser will jump to step 10 and access *example.com* with address 1.1.1.1.

Step 2: if there is not any record in the local DNS mapping table, then the user browser queries the allocated DNS server (this allocated DNS server requires a pre-configuration, see Figure 47 below). If the corresponding information on *example.com* exists in the allocated DNS server, the IP address 1.1.1.1 will be returned to the user's browser, and then it can access *example.com* with the correct address, through Step 5 and Step 10 as shown. Otherwise, it executes Step 3.



DNS Servers:

192.168.1.4
223.5.5.5
211.167.230.100

IPv4 or IPv6 addresses

Search Domains:

FIGURE 47: Allocated DNS Server setup on IOS system.

Step 3: if there are no useful records in the allocated DNS server, another query will be sent from the allocated DNS server to the root DNS server. Next, the corresponding information about *example.com* will be returned to the user browser, through steps 4 and 5. The user browser then accesses *example.com* with 1.1.1.1 once they receive the IP address in Step 10.

Step 4: The IP address of *example.com* (1.1.1.1) is sent to the allocated DNS server from the root DNS server.

Step 5: The IP address of *example.com* (1.1.1.1) is sent to the user from the allocated DNS server.

Step 6: There is an alternative way that DNS information queries can be made using the default DNS server rather than an allocated DNS server. If the user has not pre-configured an allocated DNS server, the subsequent DNS query packages will be sent to a default DNS server. As in Step 2, If the corresponding information about *example.com* exists in the default DNS server, the IP address 1.1.1.1 will be sent to the user browser which can then access *example.com* with the address 1.1.1.1, see Steps 9 and 10. Otherwise, it executes Step 7.

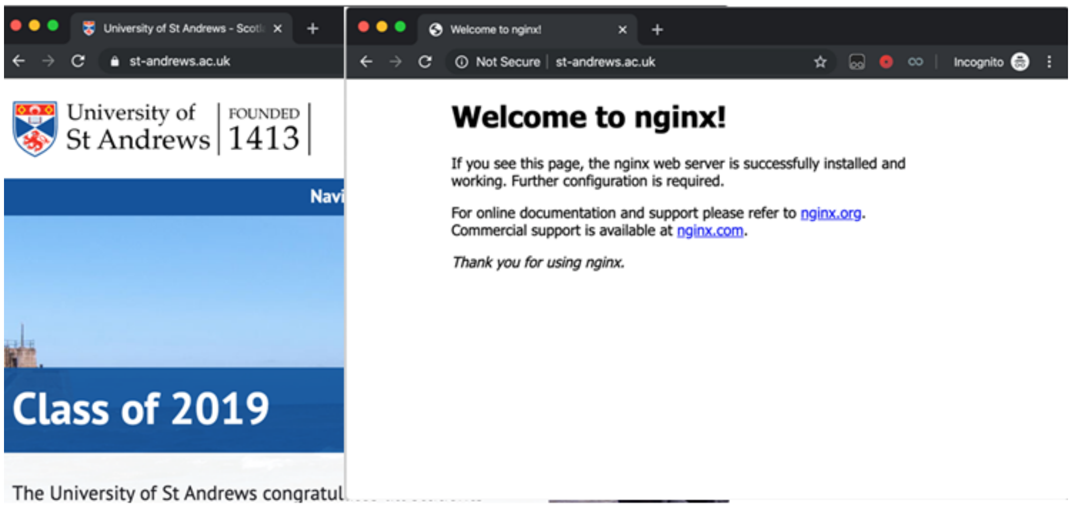
Step 7: If there are no useful records in the default DNS server, another query will be sent from the default DNS server to the root DNS server. Next, the corresponding information about *example.com* will be sent to the user browser, see Steps 8 and 9. The user accesses *example.com* on the browser with 1.1.1.1 once they receive the IP address in Step 10.

Step 8: The IP address of *example.com* (1.1.1.1) is sent to the default DNS server from the root DNS server.

Step 9: The IP address of *example.com* (1.1.1.1) is sent to the user from the default DNS server.

Step 10: The user browser establishes the connection with IP address 1.1.1.1 for accessing *example.com.*

DNS redirection, also known as a DNS hijacking attack, is a type of DNS attack that redirects the user to an unexpected malicious site, without any user knowledge, through an incorrectly resolved DNS query traffic. Compared to a general phishing attack, a phishing website in DNS hijacking attacks has higher camouflage as this kind of attack is not as easily identifiable as a suspicious URL address or an obvious SSL certificate warning notification. Under this attack, most suspicious user observations are eliminated, and not only is the accessed URL the same as the official website, but also the associated SSL certificate is not displayed in an obvious way, as shown in Figure 48 below.



The University of St Andrews congratul

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

FIGURE 48: Two accessed results; benign (left) and DNS hijacking (right).

Normally, the IP address of a targeted website is indexed in a DNS mapping table after a DNS query is sent from the user. Subsequently, the user can receive the corresponding IP address from the DNS server. The relevant HTML response will be parsed according to this IP address. However, under a DNS hijacking attack, in order to return a specific fake IP address to this user browser, the phishers usually tamper with the DNS mapping table. Any DNS query traffic in this DNS server, regarding this targeted website, will be redirected to a malicious site. An illustration on the DNS resolution of *example.com* under a DNS hijacking attack is shown in Figure 49 below. Once the DNS mapping table has been tampered with by a phisher, the user will receive a suspicious IP address, say 2.2.2.2, for *example.com* on the browser. This user will then access *example.com* with the false IP address 2.2.2.2 rather than the expected IP address of 1.1.1.1 on their browser.

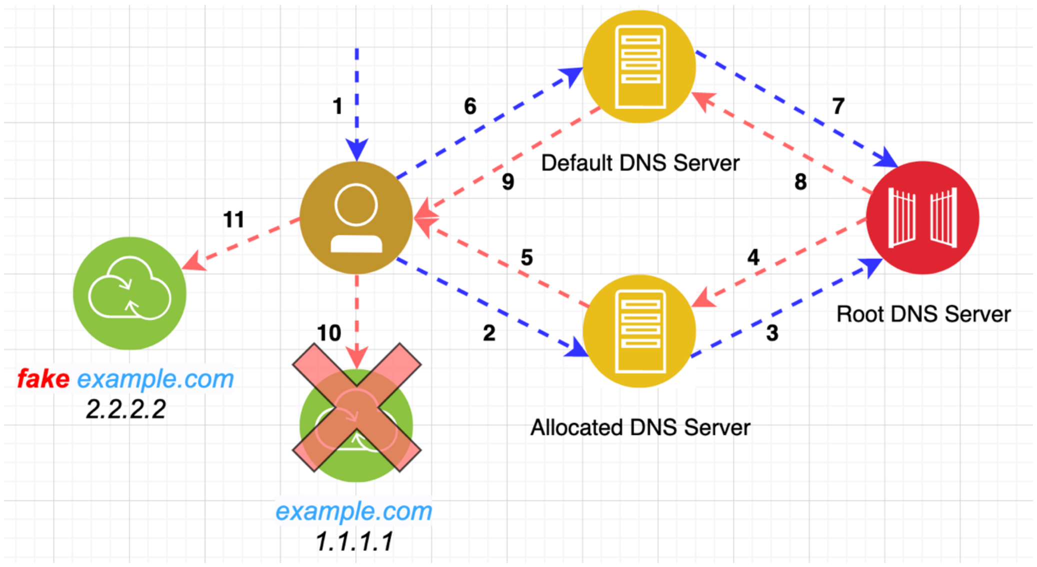
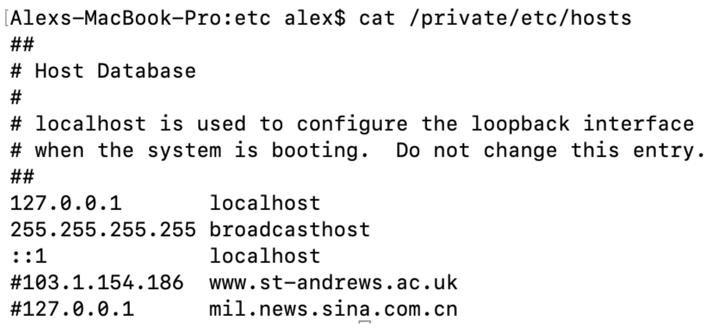


FIGURE 49: An illustration of the DNS resolution of example.com with a false IP address (2.2.2.2) under a DNS hijacking attack.

Typically, a DNS mapping table exists in both the user's local computer (see Figure 50 below) and the remote system, such as a router or ISP (Internet Service Provider) server. Therefore, to perform this kind of DNS attack, phishers either take over a local DNS mapping table or intercept a DNS communication between the victim and a remote server [166].



```
Alexs-MacBook-Pro:etc alex$ cat /private/etc/hosts
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
#103.1.154.186  www.st-andrews.ac.uk
#127.0.0.1      mil.news.sina.com.cn
```

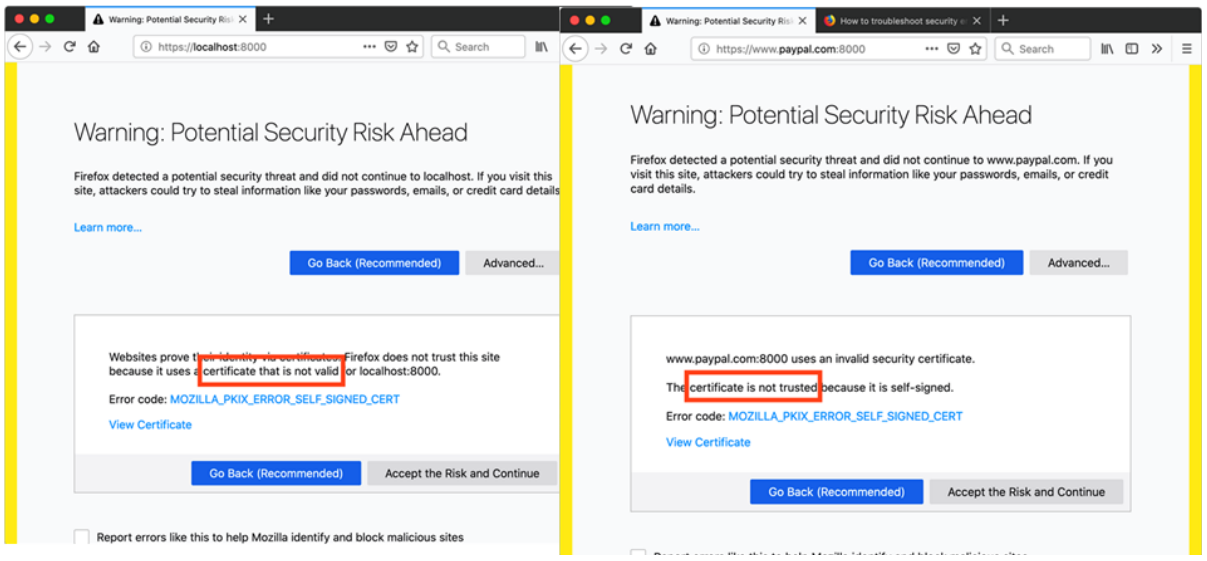
FIGURE 50: Default local DNS setting on a MAC OS.

Current DNS hijacking prevention approaches and their limitations

Mocan [167] summarized how most phishers manage to perform DNS hijacking. Typically, there are several basic types, which are:

- Through Malware.
- By Compromising DNS Servers.
- By Setting Up Rogue DNS Servers.

Most security specialists use Secure Sockets Layers (SSL) to protect their traffic [166] [168]. This can also be used for preventing hijacking due to the asynchronous encryption. During SSL communication, there are two kinds of potential errors regarding SSL certification; '**Not Matched**' and '**Not Trusted**'. In the Not Matched error, the browser states that the 'certificate is not valid' and returns an error code if the SSL certificate information is not matched with the accessed website. In the Not Trusted error, the browser will notify that the 'certificate is not trusted' if the SSL certificate is not issued by a trusted certificate organization, as shown in Figure 51 below. In addition, as the not matched error has a higher risk, the browser will only warn about the 'not trusted' if the SSL certificate has both problems.



Warning: Potential Security Risk Ahead

Firefox detected a potential security threat and did not continue to localhost. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details

Learn more...

Go Back (Recommended) Advanced...

Websites prove t[illegible] Firefox does not trust this site because it uses a certificate that is not valid for localhost:8000.

Error code: MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT

View Certificate

Go Back (Recommended) Accept the Risk and Continue

Report errors like this to help Mozilla identify and block malicious sites

Warning: Potential Security Risk Ahead
How to troubleshoot security
https://www.paypal.com:8000
Search

Warning: Potential Security Risk Ahead

Firefox detected a potential security threat and did not continue to www.paypal.com. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

Learn more...

Go Back (Recommended) Advanced...

www.paypal.com:8000 uses an invalid security certificate.

The certificate is not trusted because it is self-signed.

Error code: MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT

View Certificate

Go Back (Recommended) Accept the Risk and Continue

FIGURE 51: SSL certificates with the not matched (valid) warning (left) and the not trusted warning (right).

An SSL channel could completely overcome this kind of hijacking attack, unless the SSL certificate has been stolen by phishers (such as, stealing both the private and public keys of the SSL certificate) [168]. Checking an SSL certificate is a good inspection mechanism for preventing the occurrence of DNS hijacking. If the phisher redirects the user’s HTTPS traffic to an unexpected malicious website, the browser would display a ’Not Matched’ warning, because the SSL certificate in this accessed (malicious) website is not matched with the user’s expected website. To try to prevent the warning, the hijacker could forge an SSL certificate that has the same certificate information as the user’s expected website. However, a warning notification would still be shown in the user’s browser once the browser determines that the SSL certificate is not issued by a trusted organization.

HTTP downgrade attacks have been exploited to avoid the occurrence of SSL certificate warnings. The SSL certificate warning could be eliminated, resulting in the expected HTTPS connection being redirected to a malicious HTTP phishing website by the hijacker. An HTTP connection is established to this phishing website, and an unobvious ’not secure’ symbol is displayed on the browser bar (see Figure 52 above), which is shown instead of the SSL certificate warning. This kind of MITM-attack scheme also has been introduced by the tool SSLstrip [169]. However, there is the possibility of this attack not being able to be executed successfully due to the appearance of a new prevention mechanism, the HTTP Strict Transport Security (HSTS) protocol. HSTS is an HTTP header that is deployed to improve the security for the current domain. In particular, any conforming browser header that has received an HSTS header from, for example, ***http****://www.aaa.com* will automatically forward the request to site ***https****://www.aaa.com* [170]. HSTS is used to ensure communication security by redirecting the plain HTTP connection to an encrypted HTTPS connection. A variety of cyber-attacks, such as DNS spoofing and HTPS stripping, have been mitigated due to the configuration of the HSTS header. However, other websites, which do not configure the HSTS policy, still face this threat. Therefore, an advanced phishing attack, for example a phishing website under a DNS hijacking, can bypass the user’s security awareness. So, not only does the accessed URL look the same as the legitimate website, but also the SSL certificate warning is not displayed in an apparent way.

DNS-related attacks have a large impact on various industries, and even well-known enterprises cannot escape this onslaught of attacks. According to a Cisco report [165], The New York Times and Twitter had been redirected to a hacktivist website during attacks in

2013 and 2014. Google branches (Vietnam and Malaysia) were hijacked via a DNS attack in 2015. With the rise of electronic coins, the Blockchain (company) suffered the same DNS hijacking attack in 2016. Recently, more government sectors suffered such attacks. As the published report from CrowdStrike's threat intelligence team revealed [171], 28 organizations in twelve different countries were hijacked during 2017 to 2019. The detailed summary is listed in the Appendix Chapter.

DNS hijacking detection approaches are summarized by Wang [172]. There are three main categories: passive monitoring detection, false packet detection and cross check query. Also, the relevant benefits and drawbacks have been subsequently analysed. Some free DNS protection products have been introduced. For example, Rash [168] mentioned two approaches to diagnose DNS hijacking, WHOISMYDNS[21] and Router Checker[22]. WHOISMYDNS is an online website and not a dynamic approach. It can detect your relevant DNS server information automatically, such as the DNS server IP address, the name of the reverse DNS and the IP owner information. The suspicious DNS server can be identified if this is indexed in their blacklist. A Router Checker is a tool which compares your DNS server with an authorized DNS server to identify any mismatches. Unlike WHOISMYDNS, this authorized server does not need to be updated frequently, but this tool does require a pre-installation. In addition, an overview of popular DNS protection online tools was published by Keary [173], who listed Neustar UlreaDNS[23], Comodo Dome Shield[24], Cloudflare[25], StackPath DNS[26], Paloalto

[21] WHOISMYDNS: It is a free way to detect if your router has been hijacked by phishers. Available at *http://whoismydns.com/.*

[22] Router Checker: It is a free and instant DNS hijacking test. It detects if your router settings have been tampered by phishers. Available at: *https://www.f-secure.com/gb-en/home/free-tools/router-checker*.

[23] Neustar UlreaDNS: It is a DNS protection solution for enterprise users, with high performance that guarantees 100% website availability. Available at: *https://www.publicdns.neustar/*.

[24] Comodo Dome Shield: It is a secure DNS filtering solution for enterprise users, it monitors the real-time website traffic and blocks the risky websites. Available at: *https://www.comodo.com/*.

[25] Cloudflare: It is a DNS provider, which manage all your domains through user interface or API. Available at: *https://www.cloudflare.com/*.

[26] StackPath DNS: It is a DNS protection tool with high speed and low latency. Available at: *https://www.stackpath.com/*.

Networks DNS Security[27], Cyren DNS Security[28] and DNSFilter[29]. Most of these products are based on the summarized detection approaches above, and they are charged services.

However, the presented DNS hijacking detection tools are used to diagnose whether a user's DNS server is healthy, but a phishing website cannot be detected if the user is under a DNS hijacking attack. Also, most phishing mitigations do not consider this case, and machine learning based approaches that select the features from URLs and webpages may not be appropriate. As the accessed URL content is the same as the legitimate website, the features gathered from the accessed website have a high similarity with the corresponding legitimate website, which will affect machine learning prediction results.

5.1.2. Reviews on ISP hijacking attacks

There is another major hijacking attack that is not as dangerous as the DNS (Domain Name System) hijacking above, the ISP (Internet Service Provider) hijack. Unlike DNS hijacking attacks, the purpose of ISP hijacking attacks is to redirect the user (browser) to third-party advertisers for earning profit through inserting extra codes in the webpage [167]. An example is shown in Figure 52 below. The user's traffic has to pass through their ISP either arriving at the destination website or receiving the response from the destination website. The ISP hijack (also called the HTTP hijack) occurs in Phase 5 when data is passed back to the user's browser. The third-party ads will be inserted into this response traffic (usually they exist in an extra pop-up window or an extra JavaScript file) before arriving at the user's browser if this connection is unencrypted HTTP (plaintext) traffic.

[27] Paloalto Networks DNS Security: It is a DNS protection tool, which automatically blocks malicious in real-time by using URL filtering, predictive analytics and machine learning. Available at: *https://www.paloaltonetworks.co.uk/*.

[28] Cyren DNS Security: It is a cloud-based web filtering tool for protecting against DNS attack. Available at: *https://www.cyren.com/*.

[29] DNSFilter: It is a cloud-based AI-powered DNS protector, being designed to detect against cyber threats, such as malware, viruses, phishing attacks etc. Available at: *https://www.dnsfilter.com/*.

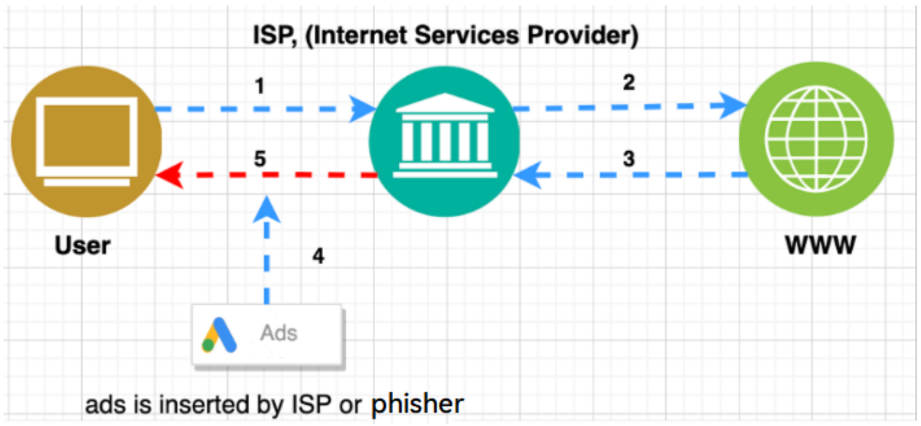
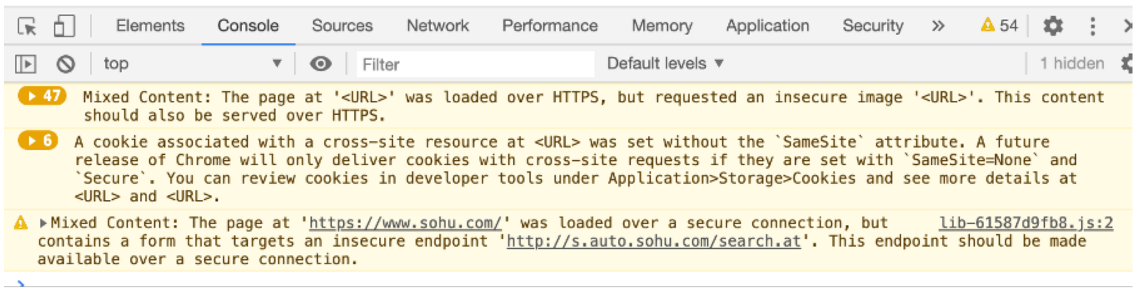

ISP, (Internet Services Provider)
1
2
5
3
4
User
WWW
Ads
ads is inserted by ISP or phisher

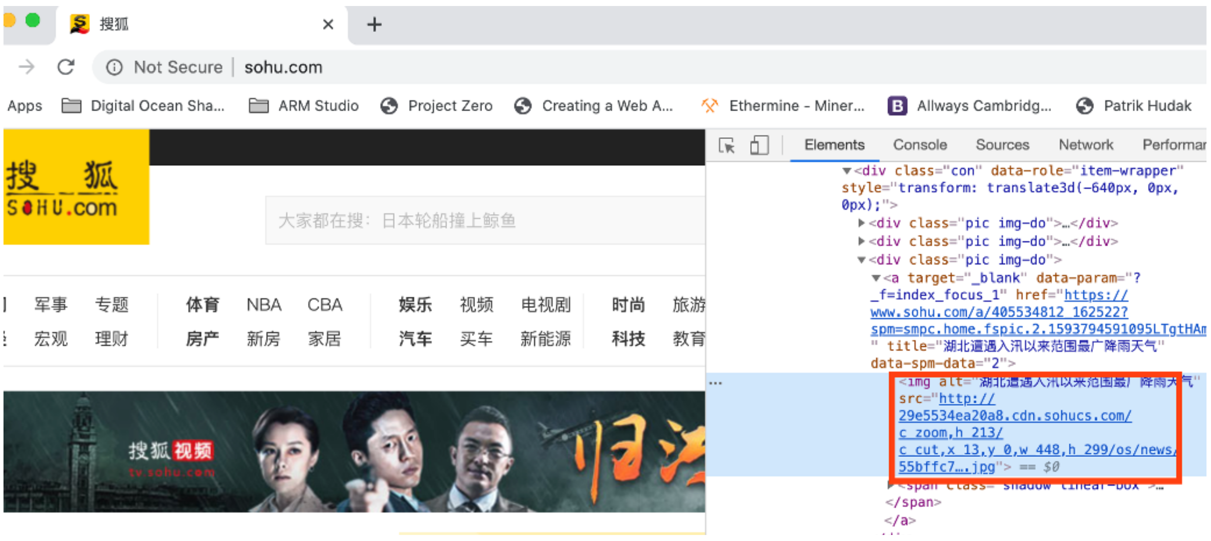
FIGURE 52: The illustration of ISP (HTTP) hijacking.

Undeniably, with the increased demand for HTTPS connections, many websites (more than one half at 51.8%) now actively redirect their connection to HTTPS in order to improve communication security [174]. However, HTTP connection usage is still very common. Moreover, for HTTPS websites, this does not mean all the contents of this website are based on encrypted HTTPS connections. Some content may use an external HTTP request. For instance, the website Sohu (available at: *https://sohu.com*), is one of the most famous news websites in China. The website is loaded over an HTTPS connection, but some internal images are from an external insecure HTTP resource, as shown in Figures 53 and 54 below. All of these insecure resources can be replaced with other unexpected content such as an advertisement, by an ISP (or a phisher), as these connections use HTTP.



```
47  Mixed Content: The page at '<URL>' was loaded over HTTPS, but requested an insecure image '<URL>'. This content
    should also be served over HTTPS.
6  A cookie associated with a cross-site resource at <URL> was set without the `SameSite` attribute. A future
   release of Chrome will only deliver cookies with cross-site requests if they are set with `SameSite=None` and
   `Secure`. You can review cookies in developer tools under Application>Storage>Cookies and see more details at
   <URL> and <URL>.
▸Mixed Content: The page at 'https://www.sohu.com/' was loaded over a secure connection, but      lib-61587d9fb8.js:2
contains a form that targets an insecure endpoint 'http://s.auto.sohu.com/search.at'. This endpoint should be made
available over a secure connection.
```

FIGURE 53: A warning about an HTTPS connection which contains an insecure HTTP resource.



```
▼<div class="con" data-role="item-wrapper"
style="transform: translate3d(-640px, 0px,
0px);">
  ▶<div class="pic img-do">…</div>
  ▶<div class="pic img-do">…</div>
  ▼<div class="pic img-do">
    ▼<a target="_blank" data-param="?
    _f=index_focus_1" href="https://
    www.sohu.com/a/405534812_162522?
    spm=smpc.home.fspic.2.1593794591095LTgtHAm
    " title="湖北遭遇入汛以来范围最广降雨天气"
    data-spm-data="2">
      <img alt="湖北遭遇入汛以来范围最广降雨天气"
      src="http://
      29e5534ea20a8.cdn.sohucs.com/
      c_zoom,h_213/
      c_cut,x_13,y_0,w_448,h_299/os/news/
      55bffc7….jpg"> == $0
     ▶<span class="shadow linear-box">…
      </span>
     </a>
```

FIGURE 54: An external insecure HTTP resource.

Both DNS hijacking and ISP hijacking have a similar goal; the malicious actions will be executed once the accessed target website is indexed by a phisher. In the DNS hijacking attack, the user traffic is re-directed to an unexpected and suspicious website to steal their sensitive information. In the ISP hijacking attack, an 'annoying' advertisement is inserted into the user's traffic in order to increase advert visibility and gain potential profit. Also, the phisher could lure victims directing them into a phishing website by inserting convincing context and links in the legitimate website.

Generally, the risk of DNS hijacking mainly involves traffic spoofing and stealing data [2]. In traffic spoofing, the hijacker (phisher) intends to expand the exposure of their or other websites in order to gain increasing website traffic. For example, users intend to access a benign website A on the browser, say *http://www.aaa.com*, but the accessed result shows the content of website B on the browser, say *http://www.bbb.com*. In this case, website B is a completely different website from website A, although the accessed URL still reads as website A on the browser, *http://www.aaa.com*. phishers redirect the user's traffic from websites A to B, and hence increase the hits of website B. Although the user would not lose any personal or sensitive data, website A is also a victim as it's status (traffic) has been reduced inadvertently. When stealing data, the phisher redirects the victims' traffic to their malicious website for stealing a user's personal or sensitive data. In particular, with the rapid development of payment systems, phishers have increasingly targeted financial data, such as third-party payment platform accounts and online banking accounts, etc. In this scenario, for instance, users still access a benign website A on the browser, but the accessed result is the content of a suspicious website C on the browser, say *http://www.ccc.com*, where website C is a fake pre-setup website created by the phisher, though the accessed URL is still seen as website A on the browser, *http://www.aaa.com*. Website C has the same content as website A, but its back-end server is connected to the phisher's server, which means any traffic will be transferred to the phisher once the user POSTs their data.

Compared to the DNS hijacking attack, an ISP hijacking attack is less threatening to users as well as being harder to detect using current defence tools. The implementation of a successful ISP hijacking attack necessitates that this traffic has to pass through the phisher's server. In a DNS hijacking attack, the phisher does not require this traffic as the attack can be executed by tampering with the DNS query table on either the victim's local machine or router. Generally, ISP hijacking attacks occur over an HTTP connection as the extra content could not be inserted into an unreadable, encrypted traffic stream. From the ProofPoint report in 2019 [175], across all domains, 53% of users use an HTTP response. Even for the largest global websites, at least 20% of the sites do not use HTTPS [176]. Hunt listed the specific details regarding these websites in [177]. In addition, data shown in WebsiteSetup [178] demonstrates that Asia has the largest percentage of Internet users, 51.8% of world users as of 2021. In China, in order to reduce the load on servers, most websites have not yet used HTTPS. Meanwhile, an ISP or a phisher can exploit the limitations of the HTTP connection to insert unwanted advertisement for promoting traffic and increasing exposure to vulnerabilities.

Undeniably, an HTTPS website has a stronger defence against ISP hijacking attacks, but not completely. HTTPS is an encrypted HTTP connection with an asymmetric algorithm, so the communication channel is more secure. Even if the network traffic is captured through a MITM attack, the message is still unreadable for the phisher. The phisher does not have the private key that establishes the initial communication between a user and the server to decrypt the secret data. For ISP hijacking attacks, the encrypted traffic could not be tampered with due to a variety of protection mechanisms, such as SSL certificate. However, there is an exception; if an unencrypted source is involved in an encrypted website, then this unencrypted source can be tampered with. For example, for an HTTPS website, it might not encrypt all of its sources, as some elements, such as images, may come from external HTTP websites. This means these images can be altered as they are sent in a plaintext format during the communication. In order to investigate this issue of HTTP resource access loaded over an HTTPS website, several extra experiments were undertaken to examine this problem in this thesis. Sites used in these experiments are:

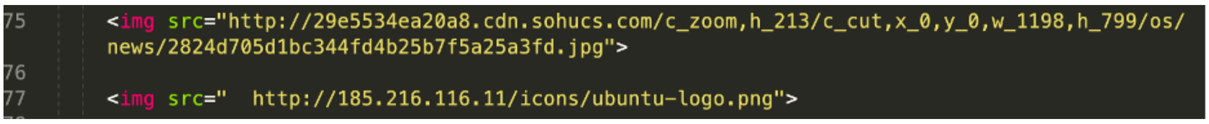
Website A: https, 103.1.154.186, or *https://allways-cam.com*.
Initially, all resources within Website A are over an HTTPS connection, which means the traffic is encrypted and cannot be parsed to a readable content by a phisher, even if this traffic is captured in the Man in the Middle (MITM attack), as the phisher does not have the corresponding keys.

Website B: https*, https://sohu.com*.
A news website, although the used protocol is HTTPS, the URL bar on the browser still displays a not secure message while accessing this link, as not all content is over the HTTPS connection. For example, some images are from the external resources via an HTTP connection.

Website C: http, 185.216.116.11.
C is a Virtual Private Server (VPS) located in Hong Kong. The initial system specification is based on an Apache system and port 80 is open for an HTTP connection. A JavaScript file named as *yw.js*[30] can be accessed and called via the link *http://185.216.116.11/yw.js*. Also, an image is stored under the path **/icons/ubuntu-logo.png**

Firstly, a test page is configured in Website A under the path of **/index_en.html**. Two images are loaded from external resources via an HTTP connection. One is from an external source from Website B; the other is from Website C, and the relevant HTML code is shown in Figure 55 below:

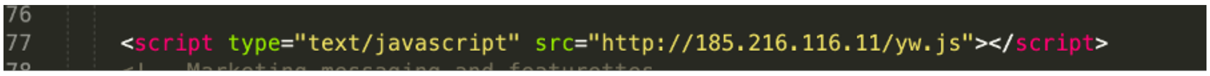
[30] *yw.js*: This JavaScript file was used to investigate the issue of HTTP resource access loaded over an HTTPS website.



```
<img src="http://29e5534ea20a8.cdn.sohucs.com/c_zoom,h_213/c_cut,x_0,y_0,w_1198,h_799/os/
news/2824d705d1bc344fd4b25b7f5a25a3fd.jpg">

<img src="  http://185.216.116.11/icons/ubuntu-logo.png">
```

FIGURE 55: The source paths of the two inserted images.

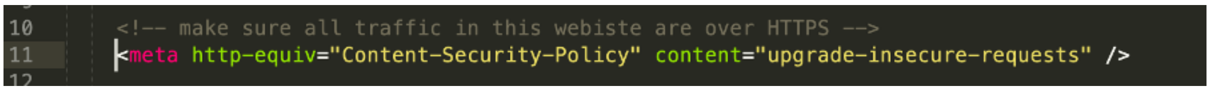
Then, a JavaScript file is inserted from Website C, as shown in Figure 56 below:



```
<script type="text/javascript" src="http://185.216.116.11/yw.js"></script>
```

FIGURE 56: The source detail of the inserted JavaScript file.

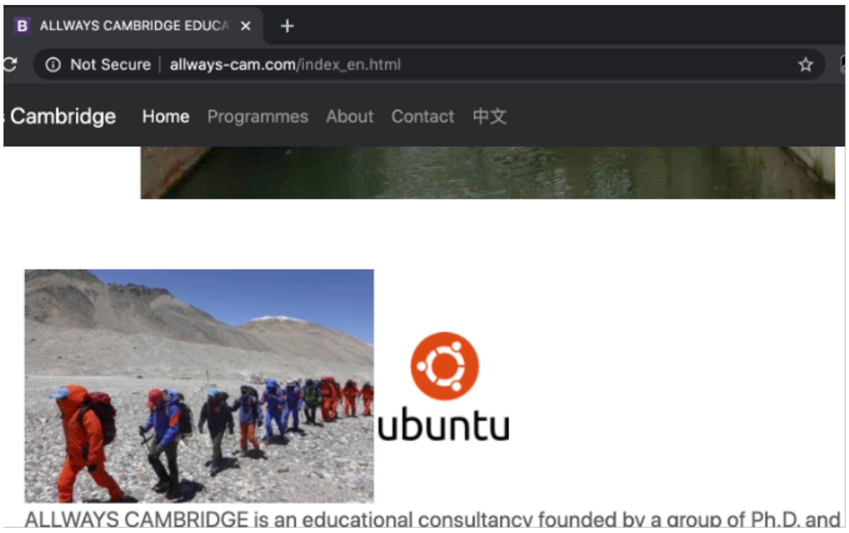
In addition, there is an HTML policy (we call this ***P***) regarding the traffic within this page. This policy ***P*** states that communication must be over an HTTPS connection and is applied under different scenarios, as shown in Figure 57 below:



```
<!-- make sure all traffic in this webiste are over HTTPS -->
<meta http-equiv="Content-Security-Policy" content="upgrade-insecure-requests" />
```

FIGURE 57: HTTPS Content Security Policy *P*.

Subsequently, we demonstrate these cases with two scenarios respectively. The policy ***P*** is configured under Scenario 1 and not configured under Scenario 2. We discovered that if a website does not configure policy ***P***, both external images are displayed on the page clearly (see Figure 58 below), but the JavaScript file is blocked. The console result on the user's browser is a mix-content error, as shown in Figures 59 and 60 below:

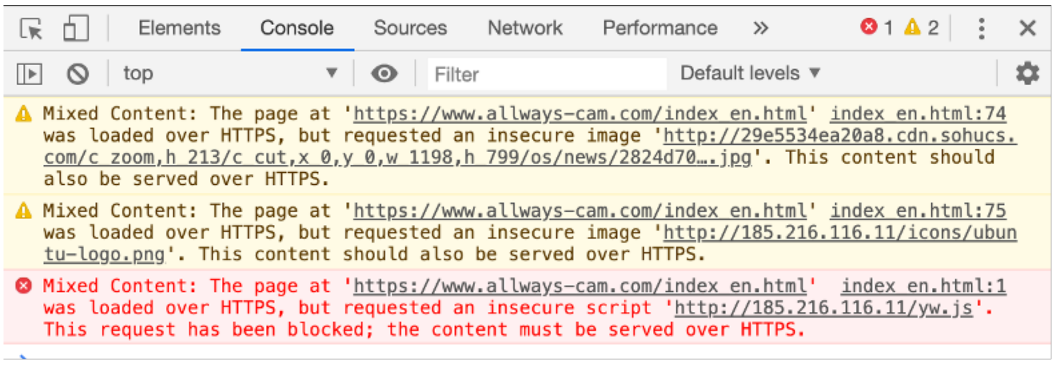

ALLWAYS CAMBRIDGE EDUCA

Not Secure | allways-cam.com/index_en.html

Cambridge Home Programmes About Contact 中文

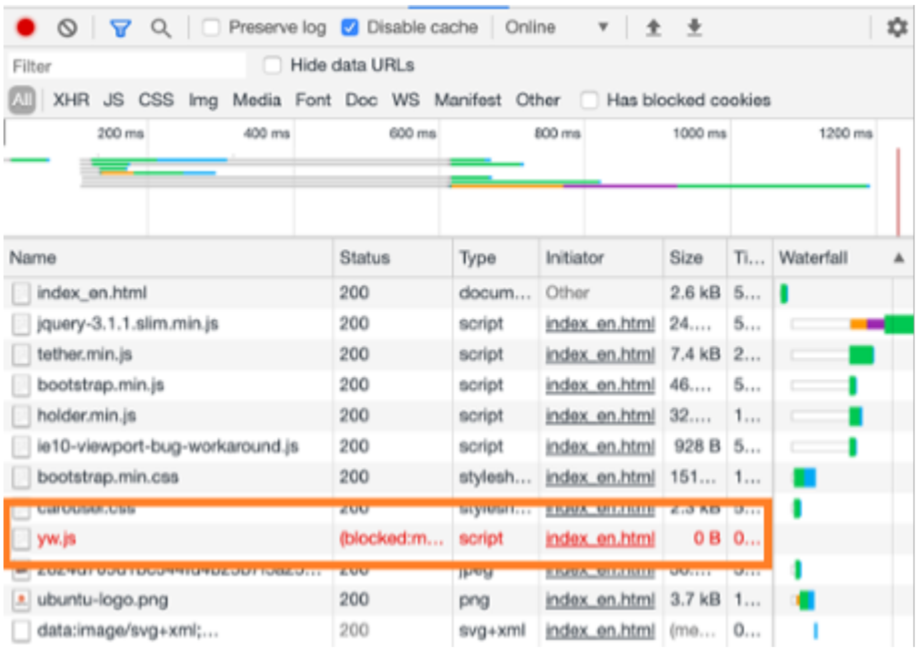
ALLWAYS CAMBRIDGE is an educational consultancy founded by a group of Ph.D. and

FIGURE 58: Accessed result of *https://allways-cam.com*.



```
Mixed Content: The page at 'https://www.allways-cam.com/index_en.html' index_en.html:74
was loaded over HTTPS, but requested an insecure image 'http://29e5534ea20a8.cdn.sohucs.
com/c_zoom,h_213/c_cut,x_0,y_0,w_1198,h_799/os/news/2824d70….jpg'. This content should
also be served over HTTPS.

Mixed Content: The page at 'https://www.allways-cam.com/index_en.html' index_en.html:75
was loaded over HTTPS, but requested an insecure image 'http://185.216.116.11/icons/ubun
tu-logo.png'. This content should also be served over HTTPS.

Mixed Content: The page at 'https://www.allways-cam.com/index_en.html'  index_en.html:1
was loaded over HTTPS, but requested an insecure script 'http://185.216.116.11/yw.js'.
This request has been blocked; the content must be served over HTTPS.
```

FIGURE 59: The notification detail about *https://allwayscam.com* in browser console.

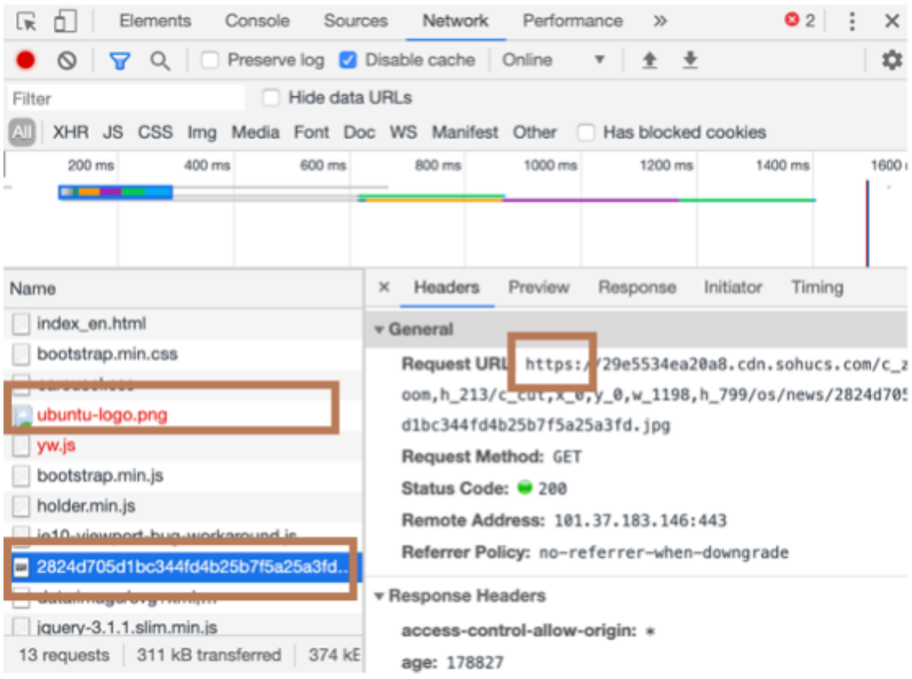


Name	Status	Type	Initiator	Size	Ti…	Waterfall
index_en.html	200	docum…	Other	2.6 kB	5…	
jquery-3.1.1.slim.min.js	200	script	index_en.html	24.…	5…	
tether.min.js	200	script	index_en.html	7.4 kB	2…	
bootstrap.min.js	200	script	index_en.html	46.…	5…	
holder.min.js	200	script	index_en.html	32.…	1…	
ie10-viewport-bug-workaround.js	200	script	index_en.html	928 B	5…	
bootstrap.min.css	200	stylesh…	index_en.html	151…	1…	
carousel.css	200	stylesh…	index_en.html	2.3 kB	5…	
yw.js	(blocked:m…	script	index_en.html	0 B	0…	
[illegible]	200	jpeg	index_en.html	[illegible]	[illegible]	
ubuntu-logo.png	200	png	index_en.html	3.7 kB	1…	
data:image/svg+xml;…	200	svg+xml	index_en.html	(me…	0…	

FIGURE 60: An error notification regarding the requested JavaScript File.

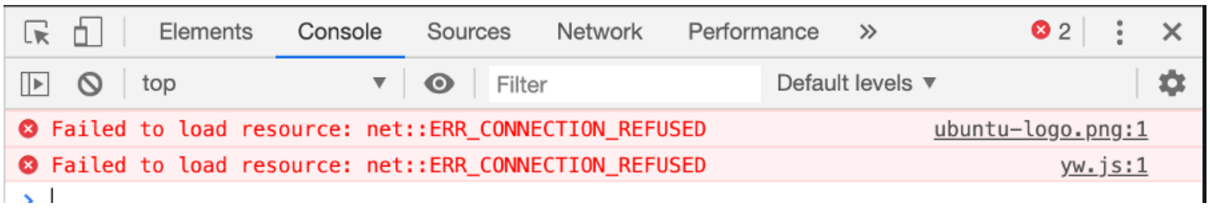
However, if the website does configure policy ***P*** correctly, all the source connections within this page compulsorily require HTTPS connections. Under this situation, the requests of both external images over HTTP need to be upgraded to HTTPS requests.

Loaded resources will fail and be dropped if the newly upgraded HTTPS request is not fulfilled. For example, an external image is cited from ***http**://abc.com/a.png*. Before configuring the policy ***P***, the website requests this HTTP connection for displaying the a.png file. After configuring policy ***P***, all requests loading within this page must be over an HTTPS connection, which means this website needs to request the previous image from ***https**://abc.com/a.png*. The accessed result is the same as the previous page, if this HTTPS request can be executed. However, this new (image) address does not exist (or is not reachable) if the domain of *abc.com* has not configured SSL certificates on their website. Thus, the accessed result of an HTTPS request to *https://abc.com/a.png* also fails as the image will not be displayed on the page. Moreover, the JavaScript file is still blocked, and the console result on the browser is "**ERR_CONNECTIO_REFUSED**", as shown in Figures 61 and 62.



```
Name                                   × Headers  Preview  Response  Initiator  Timing
index_en.html                          ▾ General
bootstrap.min.css                        Request URL: https://29e5534ea20a8.cdn.sohucs.com/c_z
ubuntu-logo.png                          oom,h_213/c_cut,x_0,y_0,w_1198,h_799/os/news/2824d705
yw.js                                    d1bc344fd4b25b7f5a25a3fd.jpg
bootstrap.min.js                         Request Method: GET
holder.min.js                            Status Code: ● 200
ie10-viewport-bug-workaround.js          Remote Address: 101.37.183.146:443
2824d705d1bc344fd4b25b7f5a25a3fd...      Referrer Policy: no-referrer-when-downgrade
                                       ▾ Response Headers
jquery-3.1.1.slim.min.js                 access-control-allow-origin: *
13 requests | 311 kB transferred | 374 kB  age: 178827
```

FIGURE 61: Both previous image requests are upgraded to an HTTPS connection. One image is accessible due to the existence of its HTTPS request; The other image (ubuntu-logo.png) is blocked.



```
Failed to load resource: net::ERR_CONNECTION_REFUSED        ubuntu-logo.png:1
Failed to load resource: net::ERR_CONNECTION_REFUSED                  yw.js:1
```

FIGURE 62: Error notification after upgrading to HTTPS connections.

Therefore, the results of this experiment are summarised as follow:

In scenario 1: The parent HTTPS website does require the external content to be loaded over an HTTPS connection (**with** policy ***P***):

- If the external HTTP resource is an image, this HTTP request will be upgraded to HTTPS. This image can then be displayed properly if this new (image) HTTPS request is accessible. Otherwise, the browser indicates an error about "**ERR_CONNECTION_REFUSED**".
- If the external HTTP resource is a JavaScript file, this JavaScript request will be blocked. Also, the browser will indicate an error regarding "**ERR_CONNECTION_REFUSED**".

In scenario 2: The parent HTTPS website does not require the external content to be loaded over an HTTPS connection (**without** policy ***P***):

- If the external HTTP resource is an image, this image can be displayed properly, although the HTTPS SSL certificate will show an insecure notification in the browser.
- If the external HTTP resource is a JavaScript file, this JavaScript request will be blocked. Also, the browser will indicate an error regarding content must be served over HTTPS in console.

The majority of users are not aware of this kind of ISP hijacking attack, as not only is the URL unchanged, but also many browsers do not display a warning. Until recently, there was no effective approach to detecting ISP hijacking attacks. Most protection or detection products in the market are good options to examine network environments and detect whether the user's DNS server has any suspicions about sites. However, they are not flexible when they face this kind of information insertion through ISP hijacking attacks, because the user's DNS server is pointed to the ISP DNS, and these DNS servers are legal and secure.

There is some existing literature that improves security and prevents potential attacks by using the structure of the DOM tree. For instance, as Nadji, Saxena and Sone presented [179], XSS attacks can be prevented by monitoring the DOM tree structure. Reis, Gribble, Kohno and Weaver [180] noted that the structure of DOM trees can also be used to detect page changes. By this method, both the client and service (server) provider need to implement their application so that a healthy (not hijacked) DOM tree can be used to compare against the target. However, with the rapid development of network architecture, most servers are being shifted to clouds, which means the implementation of this application in a service provider may be inapplicable for those servers.

5.1.3. The specific research goal for advanced phishing attacks

In the second part of this research, the focus was on the mitigation of advanced phishing attacks, including both DNS and ISP hijacking attacks:

- For preventing the phishing website in DNS hijacking attacks, we required an effective approach for both desktop platforms and mobile platforms, in particular, an approach that required as little resource consumption as possible on a mobile client.

Research on the detection of ISP hijacking for eliminating advertisement insertion has been conducted. We present a novel approach to examine the security of the user's HTML response page. Unlike previous research on mitigation, our solution simplifies the implementation procedure.

5.2. Methodology and Design of Experiments

In this section, the advanced phishing attacks were investigated from two aspects, DNS hijacking attacks and ISP hijacking attacks. The methodology used to mitigate advanced phishing attacks is explained in detail separately from the principles behind the final techniques adopted in this experimental research.

5.2.1. DNS hijacking attack

The methodology used to prevent a DNS hijacking attack is introduced. Firstly, we review both the influence and the consequence of a DNS hijacking attack and then illustrate the principles of our solution. Finally, the research aim is determined.

The Influence of DNS Hijacking Attacks

As we mentioned in the previous chapter, a DNS hijacking attack, also known as DNS redirection, is a type of DNS attack that redirects the user to an unexpected (suspicious) website without any user knowledge through an incorrectly resolved DNS query traffic. Compared to general phishing attacks, a DNS hijacking attack has a higher camouflage as the accessed URL address in the URL bar on the user browser displays the same URL address as the user expected. Thus, the identification of phishing websites that are under DNS hijacking attacks is quite hard, in particular for the victim who has less security awareness or technical background.

Checking the SSL certificate is a good mechanism to identify this kind of attack. As we know, the goal of most phishing websites is to spoof and steal user sensitive data [16] [17], so these websites need a POST function to transfer their data from the malicious website to the phisher's server. However, for legitimate websites, the encryption of data transmission is required to improve personal data security as long as the data is sent from the user, in particular, for the transmission of such sensitive data. Websites will be upgraded to HTTPS from HTTP once transmitted data is encrypted. Therefore, an SSL certificate is a good feature to distinguish phishing websites and even if a phishing campaign is caused by a DNS hijacking attack, this SSL certificate check will be incorporated into our solution.

The Principle of our Solution

Working through our previous OCR approach, at the veracity check we can identify a potential phishing website though the SSL certificate, such as using the attributes of "**issued by**" and "**issued to**". In a phishing website, the website content is forgeable, but not so for an SSL certificate, which both attributes "issued by" and "issued to" are matched as the official website SSL certificate. For example, under a DNS hijacking attack, we have the following three scenarios:

Scenario 1:

The phisher forges a phishing website, and this malicious website involves an unmatched SSL certificate. If the SSL certificated was issued by an authenticated organization, an SSL certificate warning will be issued while accessing this phishing website as the attribute content of “issued to” is different to the official website.

Scenario 2:

The phisher forges a phishing website, and this malicious website has an unmatched SSL certificate. In order to overcome the difference from the attribute content “issued to” between the phishing website and the official website, the phisher can issue an SSL certificate that has the same content in the “issued to” attribute as the official website but from an unauthenticated organization. However, here an SSL certificate warning will be issued while accessing this phishing website, as the attribute content of “issued by” is different to the official website.

Scenario 3:

In order to avoid the SSL certificate warning while the victim is accessing the phishing website, the phisher may use an HTTP downgrade attack (see section 5.1.1 for an explanation of this attack) to eliminate this notification, though both attribute content “issued by” and “issued to” are still different to the official website, as the content of the SSL certificate is empty in an HTTP website.

Therefore, the phishing website that is under a DNS hijacking attack can be identified through the content of an SSL certificate. We decided to implement our OCR approach to detect and prevent DNS hijacking attacks, as the phishing sites are identified by comparing the SSL certificate content between accessed website and official website. The applicability of this OCR approach is examined in both desktop and mobile platforms.

Detailed Research Implementation

The aim of this second part of the experimental research is to identify phishing activities from a DNS hijacking campaign. The efficiency of the presented OCR approach will be examined in both desktop and mobile platforms.

5.2.2. ISP hijacking attacks

In this section, the methodology used to prevent ISP hijacking attacks will be introduced. Firstly, we reviewed ISP hijacking, and the defence principles. The methodology behind our presented solution is then given**.**

The Explanation of ISP Hijacking Attacks Experiment

In order to demonstrate ISP hijacking, we deployed the mitmproxy tool to execute a MitM attack. Mitmproxy is used to tamper with the response from the target website and return this hijacked response to the user's browser.

As shown in Figure 63 below, traffic from the user to their target website will go through an Internet Service Provider (ISP), as in phases 1 and 2. In our experiment, an ISP was simulated by using mitmproxy. Subsequently, any response will be returned through the ISP in phase 3. However, in any ISP, the response traffic may be altered. For example, in order to earn a third-party advertisement fee, a phisher or an ISP may insert an extra element, such as a *<div>,* a pop-up window or a JavaScript file, to promote a third–party site. Thus, the final response that returns to the user's browser is altered in phase 4, and although it may not be malicious, it still contains unwanted information. This attack includes the following:

- An ISP hijacking attack needs to tamper with the (returning) page content; thus it mainly occurs in HTTP websites. However, as with the previous experiment, this kind of hijacking attack still occurs within a HTTPS website as long as there is an HTTP request embedded within this website.

- The traffic must go through the phisher (either an ISP or a phisher), as otherwise the phisher would not be able to control the traffic.

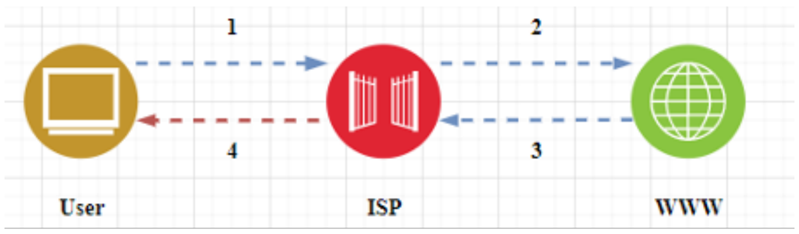


FIGURE 63: The process of an ISP hijacking attack.

Therefore, any traffic that goes through either the phisher or the ISP loads and over the HTTP connection has the potential to be tampered with.

The Design of Defence

When a webpage is loaded, the HTML file will be parsed. A Document Object Model (DOM) of this page will be generated by the browser. If the user is under a hijacking attack, the construction of this DOM tree will be affected due to the insertion or modification of an extra element (node or information) in the HTML. This means an unexpected change in the DOM tree can be considered as suspicious behaviour. Thus, we used a whitelist to record the initial architecture of the DOM tree for target websites and we call this tree the healthy (benign) DOM tree (i.e. original and not hijacked). By comparing the construction of the DOM trees between the accessed website and the target website in this list, the integrity and consistency of generated DOM trees for the accessed website in the user browser can be confirmed.

However, some challenges still need to be considered before implementing this approach. For example, if a security vendor provides this kind of ISP (HTTP) hijacking detection service, how does this security vendor ensure that the DOM trees they obtain are benign? Especially if this hijacking manipulation is executed by a middleman, such as a phisher. This security vendor can generate a healthy DOM tree as their traffic is not under the phisher's control. However, if this suspicious action is implemented by an ISP, this security vendor may face the same challenges with users as they may have the same ISP network. Moreover, as this whitelist-based approach is not dynamic, in order to keep up with the updates of websites, how often do their DOM trees need to be refreshed? Hence, these challenges have been summarised and listed as follows:

- C1: How to obtain a healthy DOM tree?
- C2: How often are these DOM trees updated?
- C3: How to verify the status of a DOM tree by comparison with the whitelist?

- **The method of obtaining a benign DOM tree**

For challenge C1, Reis, Gribble, Kohno and Weaver [180], presented an approach in which both the service provider and the client needed to install their application. In order to ensure the tree is not changed during the network transmission, the service provider needs to generate a checksum related to the structure of their DOM tree, and then sends this to the client for the subsequent verification. This checksum can be seen as a reference to the healthy DOM tree. However, this kind of implementation will be a heavy workload as it needs to deploy on both client and server sides. Despite its decreasing usage, HTTP still constitutes a large proportion of the internet traffic [175] [178]. The websites are categorised into two parts: cloud-based and physical-based (server is not located in the cloud). In cloud-based websites, cloud usage not only provides a low maintenance cost and high flexibility, but also theoretically improves the security as the internal communication in a cloud is under a virtual channel. This means that the data transmission should be encrypted and should not be altered if the security company places their services with the same cloud as these cloud-based websites. A schematic is shown in Figure 64 below.

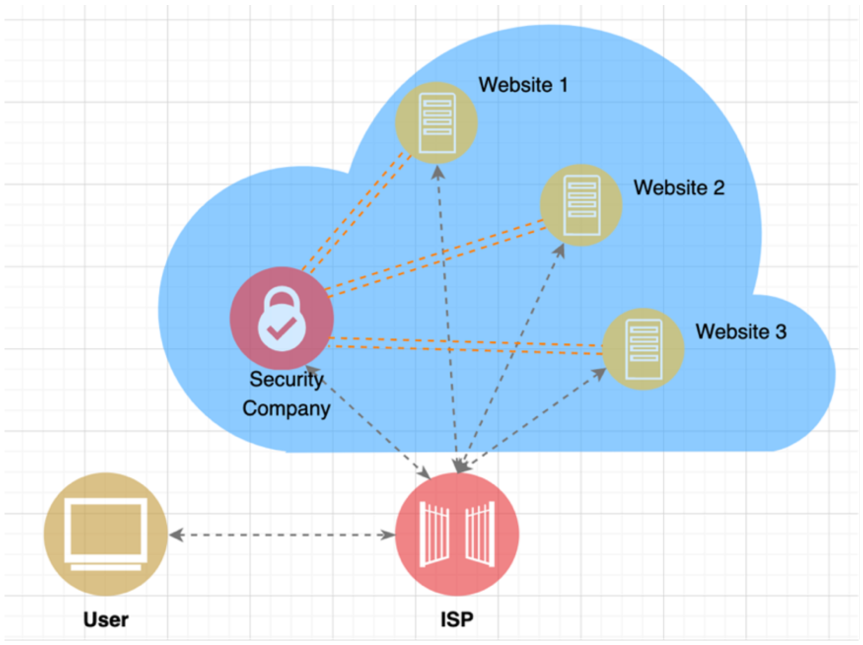


FIGURE 64: The proposed architecture of security vendor vs cloud-based websites.

However, in physical-based websites, most servers are placed in the physical building, rather than in the remote cloud. For this case, a security company cannot ensure the security of traffic during the data transmission. Hence, these websites have to establish a safe channel to the security company in order to improve the security of the data transmission. Alternatively, the maintainer of physical-based websites provides the benign DOM tree actively, or the hash of a DOM tree, and send this to the security company for integrity checking.

Therefore, the security company that provides ISP hijacking detection needs to place their server in the same cloud, to ensure that the traffic between the security company and the target websites is localised. When acquiring a healthy DOM tree from the target website, the cloud virtual network is an encrypted channel. Also, a security company needs to establish an extra safe channel to obtain the healthy DOM tree from the physical-based websites which have not placed their servers in a remote cloud, though these websites are not numerically large, as the Cloudwards [181] revealed that nearly 94% of all companies use clouds so far, also Financeonline stated [182] that half of all data will be stored in the cloud by 2025. Alternatively, the maintainer of any physical-based websites needs to provide a benign DOM tree actively and regularly to their security companies for further protection.

- **Updating a DOM tree**

For challenge C2, the whitelist does not require a frequent update. For many cloud-based websites, the information and traffic exist mainly in a cloud, and therefore the whitelist can obtain a healthy DOM tree through real-time processing as the communication is point-to-point within the cloud. For physical-based websites, the whitelist can obtain a healthy DOM tree through either establishing a safe channel between the website and the security vendor or via an active website maintainer. If this physical website has frequent updates, our solution may not be suitable as it would require a large consumption of human effort.

- **Verification Procedure: targeted DOM vs referenced DOM**

For challenge C3, hijacking result can be varied, as an unexpected piece of information may either be inserted or modified, such as through a JavaScript file, pop-up windows or an image. We believe that the highest threat is changing a JavaScript file (here, the changing involves insertion and/or modification) as the webpage function can be changed through JavaScript code, and most malicious activities, such as redirect user into a malicious website, CSRF attack, are implemented by using JavaScript.

For the verification procedure between a targeted DOM tree and a referenced DOM tree (i.e. held in a whitelist), both the (node) count and the content of DOM tree nodes need to be considered for verification. This process is classified into two aspects due to the user requirements; **Consistency Verification** (to verify whether two DOMs are consistent); and **Differentiation Verification** (to confirm which part of the tree is different, if two DOMs are not consistent).

In **Consistency Verification**, a hash function is used to verify the consistency between the targeted DOM and the referenced DOM. This involves two basic types: the HTML-based and the JavaScript-based. In the HTML-based types, the structure of a DOM tree is an object and the verification result is confirmed by comparing the Hash value of the DOM trees, Hash (DOM_{obj}), between the targeted and referenced trees. Then, we have the comparison predicate as follows:

$$\textit{If}\ \text{Hash}\ (DOM_{target}) == \text{Hash}\ (DOM_{reference})$$

In the JavaScript-based type, the verification result is determined from the content of each JavaScript file. In our approach, we use the *fetch()*[31] function to gather the content of the JavaScript file as it is impossible to extract content by using the function *document.documentElement*[32]. Subsequently, the corresponding Hash value of this content is computed by using the *Hash()* function, and we repeat this manipulation on each JavaScript file in this webpage. Finally, each hash value will be accumulated for the further comparison. We have the **Formula** as follows:

$$\sum_{k=1}^{n} \text{Hash}\ (fetch\ (JS_k))$$

Where: *n* represents the number of JavaScript files; *JS* means a JavaScript file and JS_{k} refers to a particular JavaScript file[33].

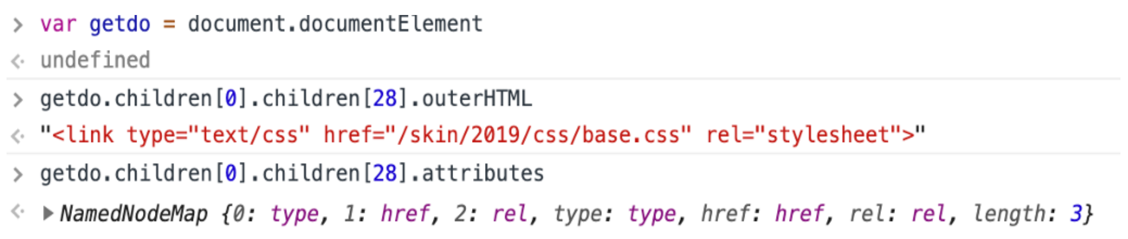
Therefore, both webpage types (HTML and JavaScript) need to be compared with the content of the corresponding whitelist DOM tree. The targeted website can be seen as a benign website as long as both results are consistent. If a difference occurs in the targeted website, we can execute the next differentiation verification for the further detection.

In **Differentiation Verification**, three element attributes are used to determine differences between the targeted DOM tree and the referenced DOM tree. These are:

- nodeName: gathered by *document.documentElement.nodeName*[34].
- attributes: gathered by *document.documentElement.attributes*[35].
- innerHTML: gathered by *document.documentElement.innerHTML*[36].

[31] *fetch():* A JavaScript function, which is adopted to send network package, also, the content of package is configurable in this function.
[32] *document.documentElement*: A JavaScript function, which is adopted to obtain a completed resources of webpage and response file.
[33] The computing process may be slow which is subject to the count of JS files, in some cases, JS files are a lot, but this is necessary, because the changing of JS file has the highest threat as most malicious activities can be executed by using JS code.
[34] *document.documentElement.nodeName*: A JavaScript function, which is adopted to obtain the node name of element.
[35] *document.documentElement.attributes*: A JavaScript function, which is adopted to obtain the attributes content of element.
[36] *document.documentElement.innerHTML*: A JavaScript function, which is adopted to obtain the inside detail of element.

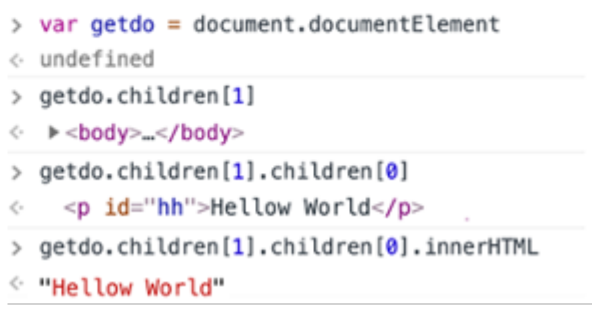
Where, nodeName is used to obtain the name of tag (node) in the DOM tree. For example, in a node of *< div id = "xx" class = "..." >*, the result of nodeName is *DIV*. The attributes method is used to gather the attributes within the node. For example, in a node of *< img id = "cover" class = "img" src = "cover.jpg" href = "xxx" >*, the result from the attributes method is *< id = "cover" class = "img" src = "cover.jpg" href = "xxx" >*. An example of an attribute method is shown in Figure 65 below:



```
> var getdo = document.documentElement
<· undefined
> getdo.children[0].children[28].outerHTML
<· "<link type="text/css" href="/skin/2019/css/base.css" rel="stylesheet">"
> getdo.children[0].children[28].attributes
<· ▶NamedNodeMap {0: type, 1: href, 2: rel, type: type, href: href, rel: rel, length: 3}
```

FIGURE 65: An example of attributes listing.

The innerHTML method is used to determine the internal content of a node, such as, in a node of *< p id = "hh" > Hello World </p>*, and the result of applying innerHTML is *Hello World*, as shown in Figure 66 below.

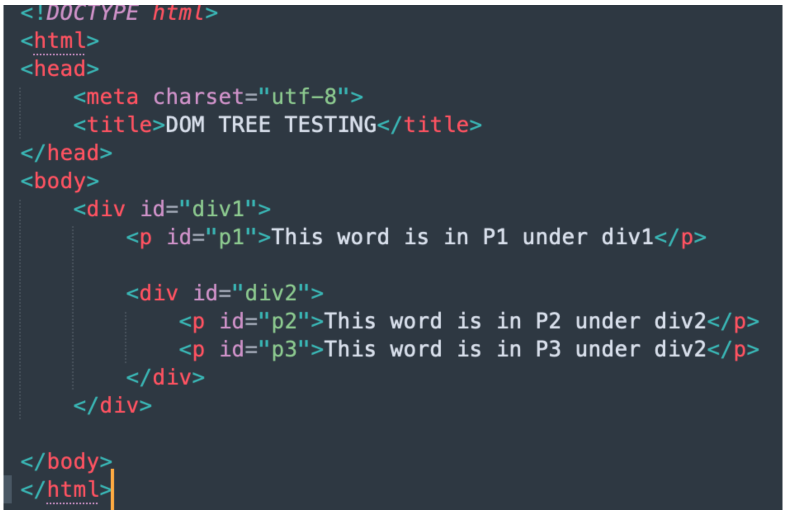


```
> var getdo = document.documentElement
<· undefined
> getdo.children[1]
<·  ▶<body>…</body>
> getdo.children[1].children[0]
<·    <p id="hh">Hellow World</p>
> getdo.children[1].children[0].innerHTML
<· "Hellow World"
```

FIGURE 66: code showing the use of the method innerHTML.

However, if this node involves children, using innerHTML will display all of the children’s contents. In order to avoid redundant information from child elements, we only focus on the node when the value of *childElementCount* is 0, and other nodes where the value of *childElementCount* is not 0, its value will set up to null.

For instance, we built an example webpage, the HTML code and its DOM tree structure are shown in Figures 67 and 68 below.



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>DOM TREE TESTING</title>
</head>
<body>
    <div id="div1">
        <p id="p1">This word is in P1 under div1</p>

        <div id="div2">
            <p id="p2">This word is in P2 under div2</p>
            <p id="p3">This word is in P3 under div2</p>
        </div>
    </div>

</body>
</html>
```

FIGURE 67: The HTML code in an example webpage.

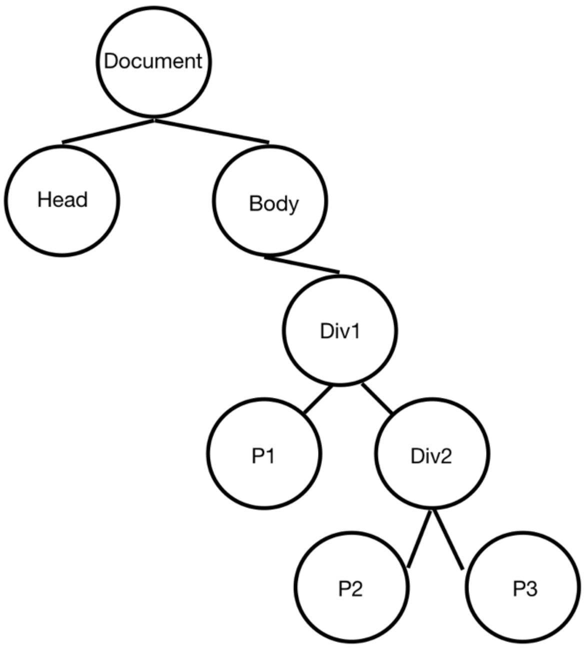
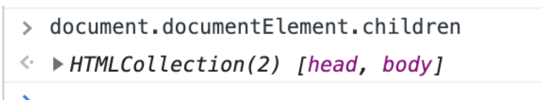


FIGURE 68: The DOM tree structure of this example webpage.

From the DOM tree structure of this example webpage, we confirm the children count in each node. The summary is listed in Table 12 below:

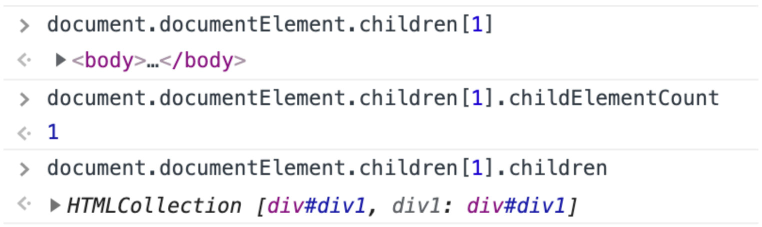
Node Name	Children Count	Children Detail	Evidence
Document	2	Head, Body	See Figure 69
Body	1	Div1	See Figure 70
Div1	2	P1, Div2	See Figure 71
P1	0		See Figure 72
Div2	2	P2, P3	See Figure 73
P2	0		See Figure 74
P3	0		

TABLE 12: The summary of each node in example DOM tree.



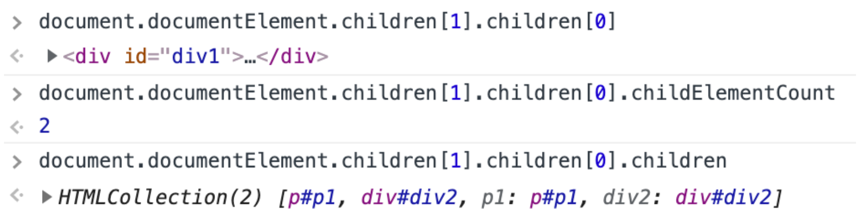
```
> document.documentElement.children
< ▶HTMLCollection(2) [head, body]
```

FIGURE 69: The children detail of node "document". Two children, "head" and "body".



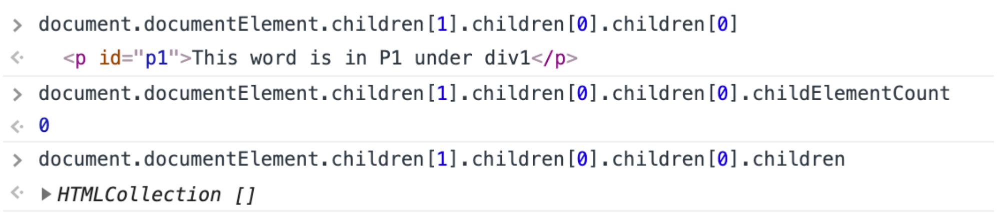
```
> document.documentElement.children[1]
<  ▶<body>…</body>
> document.documentElement.children[1].childElementCount
< 1
> document.documentElement.children[1].children
< ▶HTMLCollection [div#div1, div1: div#div1]
```

FIGURE 70: The children detail of node "body". One child, "div1".



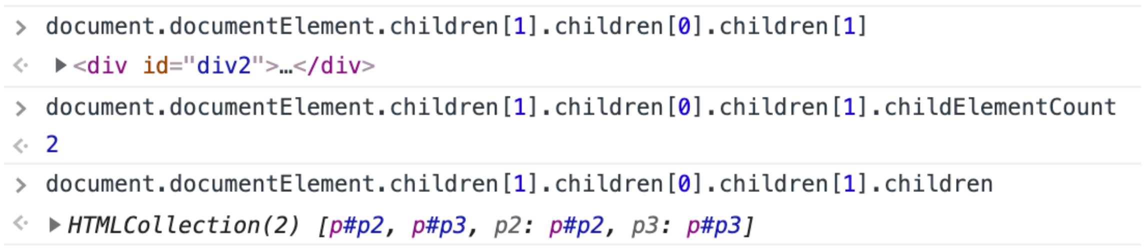
```
> document.documentElement.children[1].children[0]
<  ▶<div id="div1">…</div>
> document.documentElement.children[1].children[0].childElementCount
< 2
> document.documentElement.children[1].children[0].children
< ▶HTMLCollection(2) [p#p1, div#div2, p1: p#p1, div2: div#div2]
```

FIGURE 71: The children detail of node "div1". Two children, "p1" and "div2".



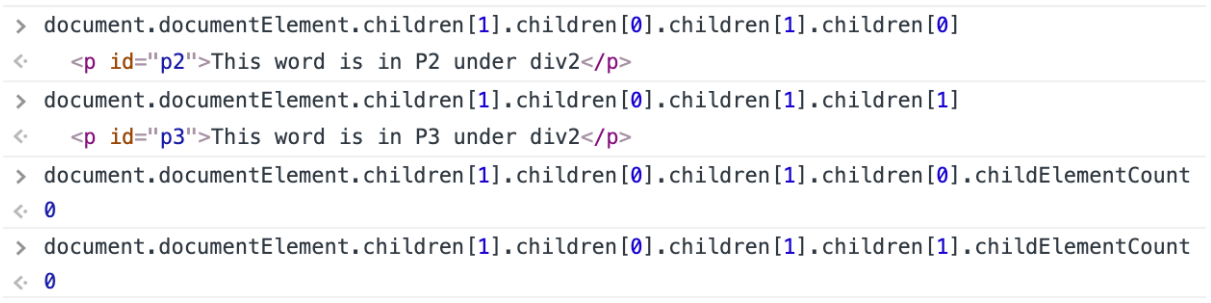
```
> document.documentElement.children[1].children[0].children[0]
<   <p id="p1">This word is in P1 under div1</p>
> document.documentElement.children[1].children[0].children[0].childElementCount
< 0
> document.documentElement.children[1].children[0].children[0].children
< ▶HTMLCollection []
```

FIGURE 72: The children detail of node "p1". Child is null.



```
> document.documentElement.children[1].children[0].children[1]
<· ▶<div id="div2">…</div>
> document.documentElement.children[1].children[0].children[1].childElementCount
<· 2
> document.documentElement.children[1].children[0].children[1].children
<· ▶HTMLCollection(2) [p#p2, p#p3, p2: p#p2, p3: p#p3]
```

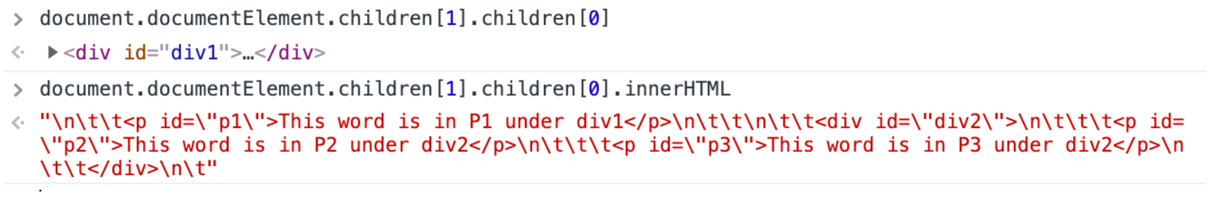
FIGURE 73: The children detail of node “div2”. Two children, “p2” and “p3”.



```
> document.documentElement.children[1].children[0].children[1].children[0]
<·   <p id="p2">This word is in P2 under div2</p>
> document.documentElement.children[1].children[0].children[1].children[1]
<·   <p id="p3">This word is in P3 under div2</p>
> document.documentElement.children[1].children[0].children[1].children[0].childElementCount
<· 0
> document.documentElement.children[1].children[0].children[1].children[1].childElementCount
<· 0
```

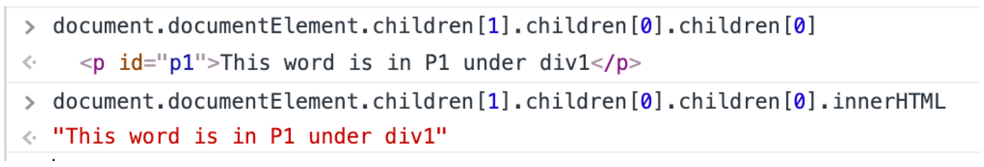
FIGURE 74: The children detail of node “p2” and “p3”. Their children are null.

As previously mentioned, the node content can be gathered using innerHTML. However, the redundant information occurred when the node’s children is not null. Such as, the node “div1” has two children, “p1” and “div2”. When we used innerHTML to obtain the node content, it involved all of information that came under this node (in this case, the information includes p1 content and div2 content, where div2 also includes p2 content and p3 content), as shown in Figure 75 below. The information is repeated if we go to the deeper node, such as p1, as shown in Figure 76 below. The parent node consists of its children content if we execute innerHTML in a node when its value of *childElementCount* is not 0. Therefore, we only collect the innerHTML information from the children’ node, and we set up innerHTML value to null in a parent node.



```
> document.documentElement.children[1].children[0]
<· ▶<div id="div1">…</div>
> document.documentElement.children[1].children[0].innerHTML
<· "\n\t\t<p id=\"p1\">This word is in P1 under div1</p>\n\t\t\n\t\t<div id=\"div2\">\n\t\t\t<p id=
   \"p2\">This word is in P2 under div2</p>\n\t\t\t<p id=\"p3\">This word is in P3 under div2</p>\n
   \t\t</div>\n\t"
```

FIGURE 75: The innerHTML detail of node “div1”.



```
> document.documentElement.children[1].children[0].children[0]
<·   <p id="p1">This word is in P1 under div1</p>
> document.documentElement.children[1].children[0].children[0].innerHTML
<· "This word is in P1 under div1"
```

FIGURE 76: The innerHTML detail of node “p1”.

- **Statistical comparison**

The BoW (bag of words) model (Frequency Representation) is used to represent the DOM tree to conduct further statistical analysis, such as:

$$T_1 = \{\text{“}key_1\text{”: value, “}key_2\text{”: value, “}key_3\text{”: value, “}key_4\text{”: value, …}\}$$

Each key consists of three attributes: *nodeName*, *attributes* and *innerHTML*, as in:

$$key_1 = [nodeName_1, attributes_1, innerHTML_1]$$
$$key_2 = [nodeName_2, attributes_2, innerHTML_2]$$
$$key_3 = [nodeName_3, attributes_3, innerHTML_3]$$
$$key_4 = [nodeName_4, attributes_4, innerHTML_4]$$

Where, a *key* is the list of combination of three attributes, and *value* is the count of the corresponding key.

Next, we calculate the differentiation value. In order to evaluate the threat degree, we need to locate the specific differentiation. Any addition, removing and modifying will result in the differentiation between the target DOM tree and the reference DOM tree. For example:

- **Scenario 1: Addition.**

The target website may have had a new element inserted by a phisher, which means this target DOM tree has an extra node in its structure compared to the reference DOM tree. An illustration is shown in Figure 77 below where the differentiation node is shown by the red block.

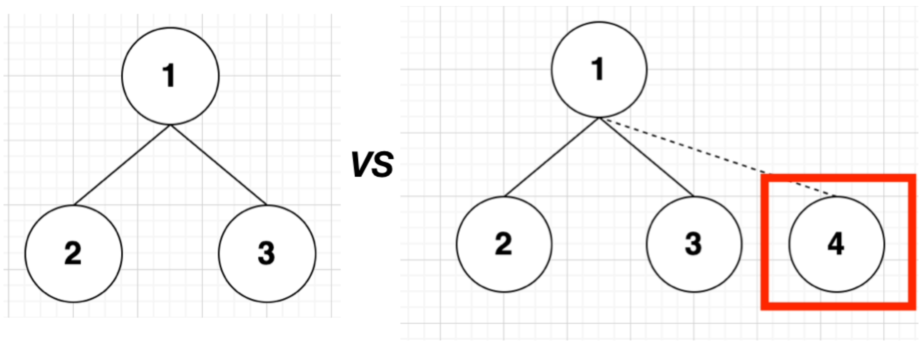


FIGURE 77: The differentiation between the reference DOM tree (left) and the target DOM tree (right), a newly added node is labelled in target DOM tree by the red block.

- **Scenario 2: Removal.**

An element is removed from the target website by a phisher, which means the reference DOM tree has an extra node in its structure comparing to the target DOM tree. An illustration is shown in Figure 78 below where the differentiation node is shown by the red block.

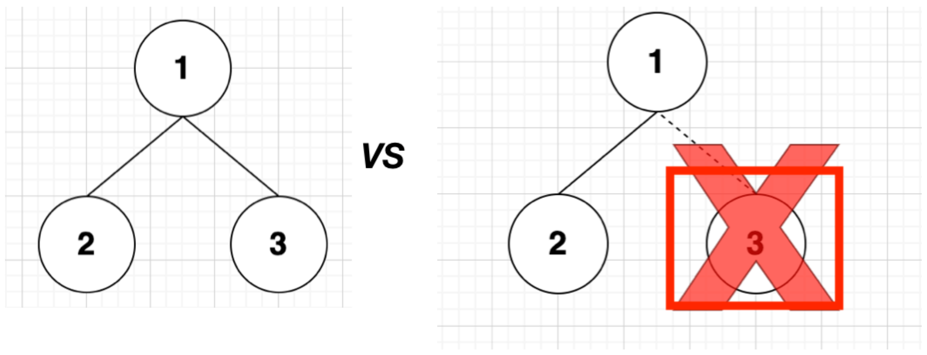


FIGURE 78: The differentiation between the reference DOM tree (left) and the target DOM tree (right), a removed node is labelled in target DOM tree by the red block.

- **Scenario 3: Modification.**

An element in the target website is modified by a phisher, which means there are two different nodes occurring in both the target DOM tree and the reference DOM tree, as the previous node has been changed to a new node. An illustration is shown in Figure 79 below and the differentiation nodes are shown by the red blocks.

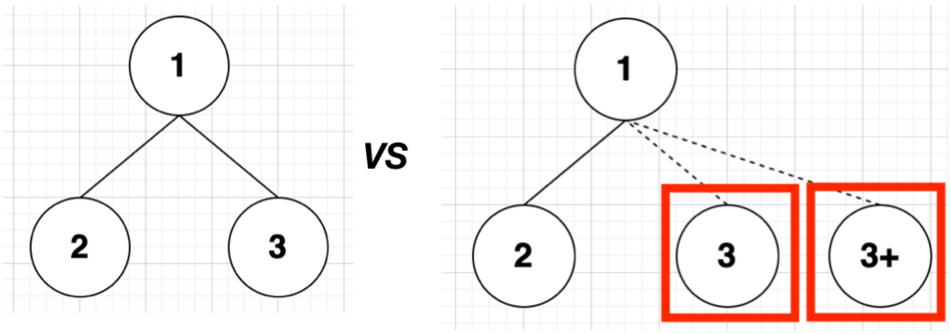


FIGURE 79: The differentiation between the reference DOM tree (left) and the target DOM tree (right), two (different) nodes are labelled in target DOM tree by the red blocks.

Thus, in our experiment, the differentiation is computed by taking the difference between the **Union** and the **Intersection**. We have the **Formula** as follows:

$$(\ T_{target}\ \cup\ T_{reference}) - (\ T_{target}\ \cap\ T_{reference})$$

Suspicious elements can be deduced from the above formula, and the final threat will be evaluated, also the specific difference will be responded to user.

Detailed Research Implementation

The aim of this part of the experimental research is to detect potential ISP hijacking attacks by evaluating the differences in DOM trees between the targeted website and a referenced website in a whitelist. The implementation of this prototype involves three steps:

- Pre-store the referenced (legitimate) DOM tree details in a whitelist.
- Verify the structure of the DOM tree
- Deploy the prototype solution

5.3. Implementation

In this section, the second experiment implementation is focussed on the prevention of ISP hijacking attacks as we have shown that the OCR approach can be used to prevent the phishing website that is under a DNS hijacking attack. (The results of our OCR approach in detecting DNS hijacking will be detailed in the next section.)

The implementation of ISP hijacking attack prevention is explained in three parts. Firstly, the pre-stored whitelist that is used to store the reference DOM tree is defined. Subsequently, the verification process is described and finally, the specific prototype deployment will be described.

5.3.1. Whitelist pre-store

In our prototype solution, a whitelist is pre-setup to store details of several websites (scalability issues are noted later when the prototype is discussed). For each website, the relevant information is stored in an object, which consists of three keys: Domain Name, Path and Tree Construction. An example of the whitelist is shown in Table 13 below. The whitelist will have individual tree constructions due to the differences of the accessing paths. Thus, we mark them separately with a different attribute number. For the individual tree construction, we still store objects by using the BoW model which we noted in the Methodology Chapter.

Whitelist Content			
NO	**Domain**	**Path**	**Tree Construction**
1	www.aaa.com	/login.php?	***Tree 1***
2		/shopping/wallet?	***Tree 2***
3	www.bbb.com	/login.php?	***Tree 3***
…		…	
N	www.xxx.com	/example#/…	***Tree n***

TABLE 13: An example of whitelist construction.

5.3.2. DOM tree verification

Initially, a lexical-based approach is used to confirm the domain name and the path of a targeted website. Subsequently, the construction of a DOM tree will be indexed from the whitelist according to the analyzed targeted website results, including domain name and path.

Finally, two verification options as noted below, can be selected depending on the user's requirements.

If the user wants to execute a quick diagnosis to detect a potential ISP hijacking attack for their network environment, a Consistency Verification is a better option (than Differentiation Verification). By comparing the DOM trees between the targeted website and the whitelist indexed tree construction, we can confirm whether they are matched (consistent). If the comparison shows no changes, then there is no apparent hijacking attack on the user's site. Otherwise, a potential hijacking risk exists in user's network environment if the match result is negative. Accordingly, the user can select a further diagnosis, the Differentiation Verification.

Under Differentiation Verification, the specific differences can be identified by calculating the difference value between the targeted DOM tree and the indexed DOM tree from the whitelist using the above Formula which determines the difference between the ***Union*** and the ***Intersection*** (mentioned in Methodology Chapter). A warning message will be sent to the user if the result of this formula is not null.

5.3.3. The deployment of the presented prototype

The deployment of this prototype involves two aspects; they are Prototype to Client (***PtoC***) and Prototype to Business (***PtoB***). In the model of *PtoC*, the purpose is to improve the security network environment and protect the user by diagnosing the threat from ISP hijacking attacks. In the model of *PtoB*, the aim is to diagnose the security of an enterprise's traffic before it arrives at the user-side, for protecting the traffic from being tampered by ISP hijacking attacks.

The PtoC model:

The security vendor implements this model on the end-side user by providing a user active detection, to protect their network security against ISP hijacking attacks.

An HTTP hijacking attack may occur in either all HTTP traffic or in specific HTTP-based websites, so we firstly classify these specific HTTP-based kind of websites as an A-Level in the whitelist. The other HTTP-based websites that are frequently accessed by the user are classified as B-Level. Security vendors will store these classified websites (that is, their DOM tree) in their whitelist.

For active user detection, a user needs to launch a network detection check proactively from the user-side, so the system will select several websites randomly from both A-Level and B-Level websites in the whitelist. Next, the user will send HTTP requests to these selected websites and convert the responding result into the construction of the DOM tree. Finally, after sending these converted DOM trees to the server-side, the verification is checked by comparing the DOM trees between the targeted tree and the corresponding tree in the whitelist.

The implementation detail of the *PtoC* prototype is described as shown in Figure 80 below:

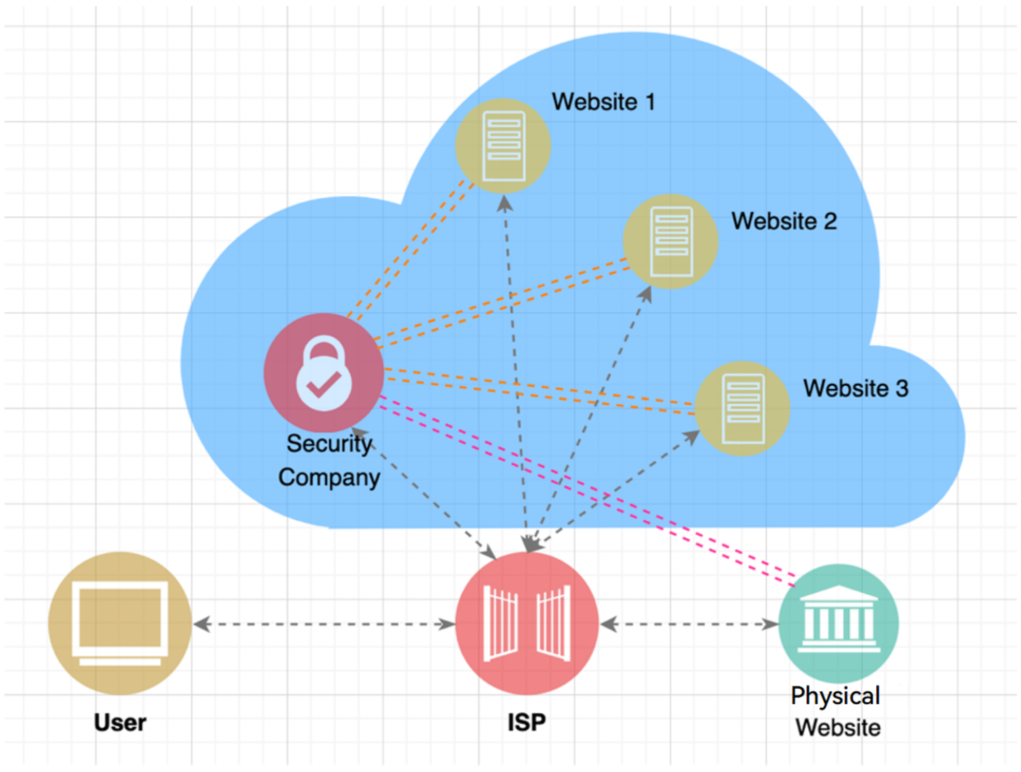


FIGURE 80: The architecture of security vendor vs HTTP connections in both cloud-based and physical-based websites with the *PtoC* model.

Initially, the whitelist is set up in a security company in the cloud, and incorporates a record of visited HTTP websites, such as a physical-based website. Then, there are several steps:

Step 1: In the user-side, a user launches an active network detection by using a script (this can be a security company's product, for example, the security company can install this script in their browser and launch this service).

Step 2: According to the threat intelligence analysis from the security companies, frequently accessed websites are identified. For example, in this case, Websites 1 – 3 are labelled (listed), also the physical-based website needs to be protected as well. Thus, these four sites will be considered as reference websites for further verification.

Step 3: In the user-side, a user sends HTTP requests to these four referenced websites (This traffic goes through the user's ISP).

Step 4: The accessed response of these four websites will be converted into DOM tree structures, and then sent to the security company via their virtual channel (point-to-point connection).

Step 5: In the cloud, the security company will verify the consistency and differences between received DOM trees from the user and corresponding DOM trees from the whitelist.

Step 6: The final verification result will be sent back to user.

The PtoB model:

The security vendor implements this model on end-side enterprises. The enterprise is the client of a security company, and they purchase this kind of service from a security company to protect their user traffic. This model consists of whitelist and network probes.

Initially, the whitelist in this model is same as in the *PtoC* model. Network probes are deployed to positions before the traffic arrives at the end-side user and after the traffic goes through the ISP. The security vender distributes their whitelist details to these network probes so that when the end-side user accesses a website, if the traffic involves a targeted website the network probe will verify the consistency of the website DOM tree.

The implementation detail of this prototype is described below, as shown in Figure 81 below:

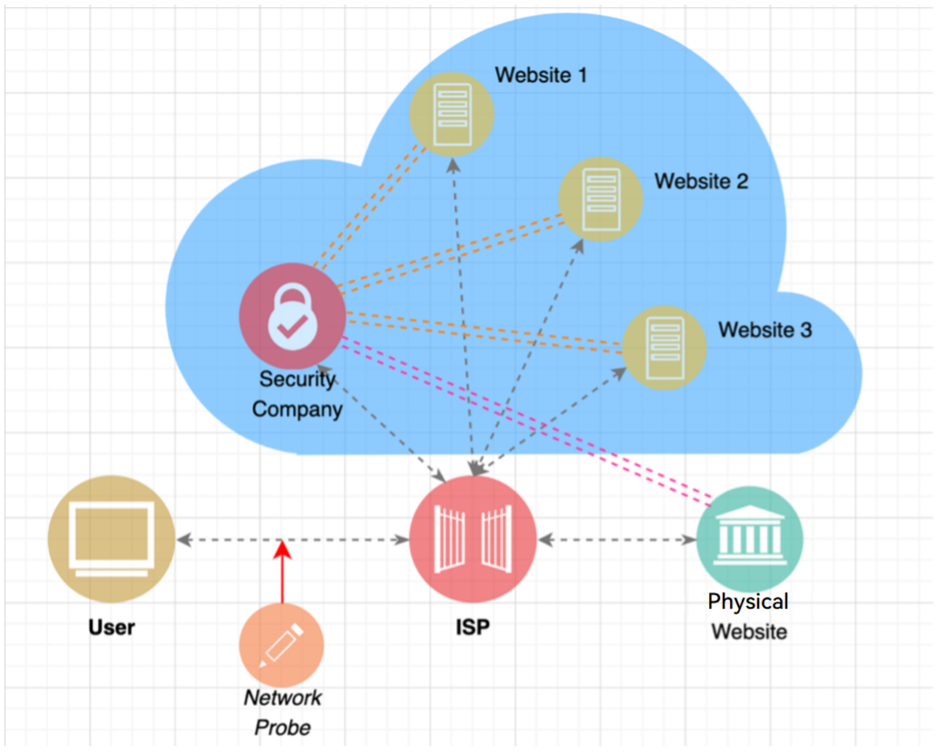


FIGURE 81: The architecture of security vendor vs HTTP connection in both cloud-based and physical-based websites with the *PtoB* model.

Step 1: The enterprise needs to purchase this kind of service from the security vendor and provides their DOM tree. If this enterprise's website has a frequent update, they need to deploy their server in the same cloud as the security vendor for establishing a point-to-point safe channel to share the DOM tree details.

Step 2: The security vendor distributes these DOM tree details to their network probes before the traffic arrives at the end-side user site.

Step 3: Monitoring the network traffic in each probe before it arrives at the user, the probe will conduct the verification manipulation if the traffic involves any protected enterprise website. For example, if Web 1, Web 2 and a physical-based website purchase this network servicer from the security vendor, the probe only focuses on the traffic that involves these three websites. The DOM tree data from these three sites would only be stored if they purchase this security service from the security vendor.

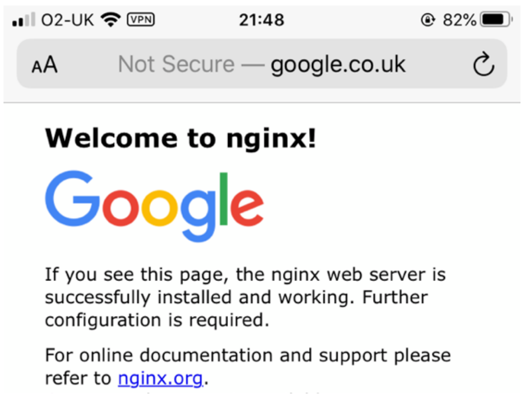
Step 4: The final result will be notified to both the user and the enterprise if the detection shows a potential hijacking problem.

5.4. Evaluation

In this section, the evaluation of the prevention of advanced phishing attacks in our presented solutions will be performed for the two separate approaches taken. Firstly, the applicability of the previously presented OCR approach will be examined as to whether it can be used to prevent the phishing website that is under a DNS hijacking attack. Next, the DOM tree consistency and differentiation approach for preventing ISP hijacking attacks will be verified by comparing a benign network and a hijacked network.

The evaluation of the OCR approach for preventing DNS hijacking attacks

In order to verify the effectiveness of the OCR approach and build an OCR prototype implementation on mobile platforms for detecting phishing website in DNS hijacking attacks, we built a fake 'Google' website and configured the mobile DNS route to our suspicious DNS server. Any query about *http://wwww.google.co.uk* was parsed and redirected to our fake webpage, the accessed result is shown in Figure 82 below.



Welcome to nginx!

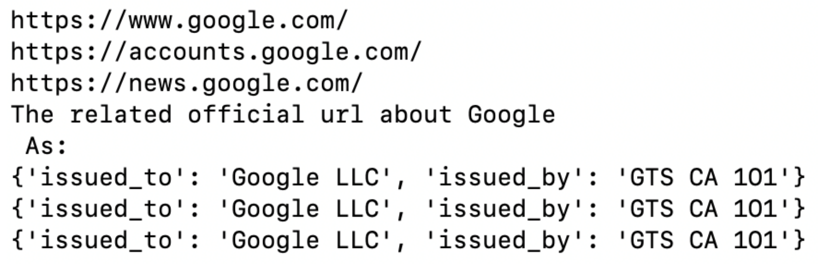
Google

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.

FIGURE 82: The access result of Google website under a DNS hijacking attack on a Safari mobile browser.

Under a DNS hijacking attack, a phisher must downgrade the network protocol from HTTPS to HTTP, otherwise the browser will pop up a warning notification stating that the SSL certificate is not valid. During the detection phase of our approach, the text 'Google' was extracted successfully from the logo and recognized by using the Google OCR API. Also, the relevant official Google websites and their SSL certificate information were indexed correctly, and the result is shown in Figure 83 below. However, the accessed URL has been downgraded to an HTTP connection, which means the SSL certificate does not exist as the traffic is not encrypted over an HTTPS connection. Thus, the value of both attributes '**issued to'** and '**issued by**' are null. Finally, the accessed URL was identified as a phishing site through comparison of the SSL certificate information between the targeted URL and the official URL.



```
https://www.google.com/
https://accounts.google.com/
https://news.google.com/
The related official url about Google
 As:
{'issued_to': 'Google LLC', 'issued_by': 'GTS CA 101'}
{'issued_to': 'Google LLC', 'issued_by': 'GTS CA 101'}
{'issued_to': 'Google LLC', 'issued_by': 'GTS CA 101'}
```

FIGURE 83: The result of the official Google website and it's SSL certificate information in our system.

Therefore, the OCR approach can be used to detect phishing websites that are in DNS hijacking attacks successfully. The accuracy of the detected result is again subject to the logo extraction and content recognition phases. Both false positive and false negative may exist if the recognition result is inaccurate. Thus, the OCR approach that is deployed to detect phishing websites in DNS hijacking attacks faces the same challenges as the approach used to detect general phishing attacks.

The evaluation of DOM tree consistency and a differentiation system for preventing ISP hijacking attacks

We designed an environment to demonstrate an ISP hijacking attack and the architecture is shown in Figure 84 below. A plug-in (extension) is configured on the browser, this plug-in is used to store the whitelist, and to simulate a product of a security vendor for providing ISP hijacking detection to browser users. A laptop is used as an ISP provider, or a phisher, and the traffic from the user side to the target website goes through this laptop. We then deployed the mitmproxy tool in this laptop to control and tamper with the transferred traffic. Extra advertising was inserted into the HTML response if the user requested traffic was based on an HTTP connection. The targeted website is a real website that we built for evaluating this experiment, and its server was purchased from DigitalOcean (*https://digitalocean.com*) and located on its cloud. Finally, the hijacked response was returned to the user's browser. The

ISP hijacking detection result will be executed on this browser plug-in (extension) according to the DOM tree comparison between the response traffic and indexed data in the whitelist.

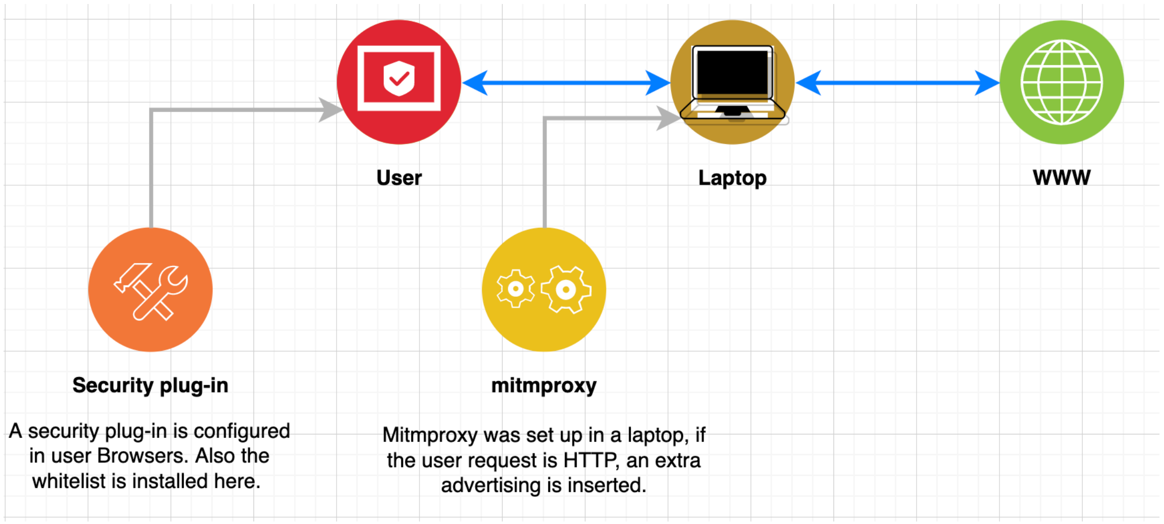
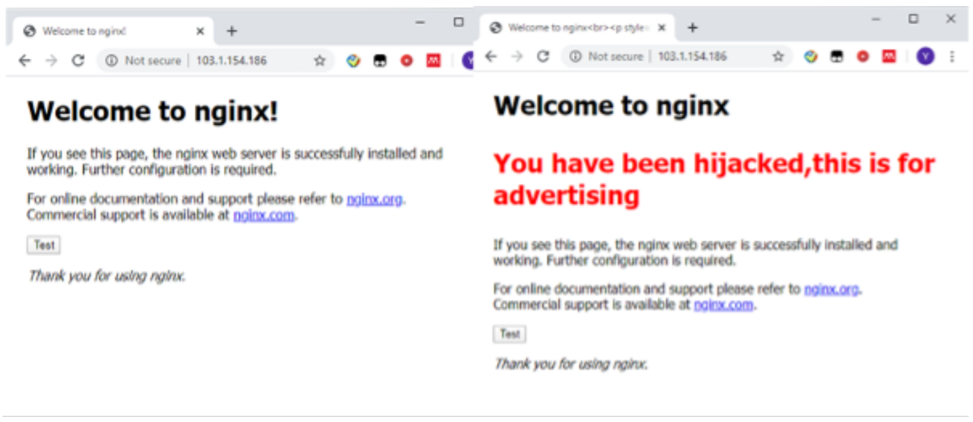

User
Laptop
WWW
Security plug-in
A security plug-in is configured in user Browsers. Also the whitelist is installed here.
mitmproxy
Mitmproxy was set up in a laptop, if the user request is HTTP, an extra advertising is inserted.

FIGURE 84: The architecture of the ISP hijacking attack in the experiment.

Subsequently, the simulated attack is executed in this website, and the browser accessed results in both a secure network and a hijacked network have been recorded separately as shown in Figure 85 below. (In this case, we used a simple element <p> instead of the embedded advertisement from the phisher. As with many real cases, this advertisement is encapsulated in an <iframe> element.)


Welcome to nginx!
Not secure | 103.1.154.186
Welcome to nginx
<p style=
Not secure | 103.1.154.186

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Test

Thank you for using nginx.

Welcome to nginx

You have been hijacked,this is for advertising

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Test

Thank you for using nginx.

FIGURE 85: Accessed results between a normal response and a hijacked response.

We used our plug-in to detect the security of the user network environment. If this targeted website exists in the whitelist, the system can index its details. Then, by comparison of the consistency and differentiation between both the accessed DOM tree and the referenced DOM tree, this newly inserted element will be identified as the potential attack vector. Finally, a notification about this network environment, stating that it may not be secure, will be displayed to the user's browser, as this plug-in successfully detects an extra node in the tree.

In our experiment, the DOM tree information of the targeted websites was stored and referenced in the whitelist. Also, an HTTP response was altered as if under an ISP hijack attack. We simulated this hijacked environment by adding nodes, removing nodes and modifying nodes with different elements in order to cover simulate real world situations. Finally, we tested our system and it successfully detected differences in all cases. The system detection summary is listed as shown in Table 14 below.

Nodes changes	Elements	Detected Result
Add	JavaScript file	**Successful**
	Pop-up windows (iframe)	**Successful**
	img	**Successful**
	Others	**Successful**
Remove	JavaScript file	**Successful**
	Pop-up windows (iframe)	**Successful**
	img	**Successful**
	Others	**Successful**
Modify	JavaScript file	**Successful**
	Pop-up windows (iframe)	**Successful**
	img	**Successful**
	Others	**Successful**

TABLE 14: The summary of detected results by ourselves using DOM Tree node changes.

5.5. Discussion

According to our experimental results, our approach not only enables a high detection accuracy rate with multiple hijacking approaches, but the detection process is also efficient. For preventing phishing websites in DNS hijacking attacks, current DNS detection tools are used to diagnose whether a user's DNS server is healthy, rather than detecting the phishing websites. Also, most phishing mitigations do not consider this case, and machine learning based approaches that select the features from URLs and webpages may not be a good option as the accessed URL content is same as the legitimate website. This results in the features gathered from the accessed website having a high similarity with the corresponding legitimate website, which affects the machine learning prediction results.

Our presented OCR approach can be used to detect phishing websites that are used in DNS hijacking attacks by identifying and comparing the SSL certificate information between the accessed website and the related legitimate website, because a trusted and correct SSL certificate is not being forged by phishers so far. The accuracy of our detected result is again subject to the logo extraction and content recognition phases. Both false positive and false negative results may exist if the recognition result is inaccurate. Thus, the OCR approach that is deployed to detect phishing websites in DNS hijacking attacks faces the same advantages and disadvantages as the approach used to detect general phishing attacks.

We then proposed two prototypes to deploy this solution for both individual and enterprise to prevent ISP hijacking attacks. However, we still faced challenges. The integrity of the whitelist is the crucial part in our method. User accessed website information must exist in this whitelist and the corresponding DOM tree construction indexed from the list. Otherwise, the system would not be able to locate a correct (benign) DOM tree construction as a reference for the verification process. Thus, an important element is:

- The integrity of the whitelist; common user accessed websites must exist in the list.

Theoretically, this challenge is hard to be overcome as the security vendor cannot promise to gather all HTTP website information, as it would be a large consumption on both human and financial resources. However, due to the continuous upgrading and evolving network architecture, most enterprises would likely shift their servers to clouds as this kind of centralized management is not only cheap, but allegedly more secure. Therefore, for a security vendor, it would not consume a lot of resources to gather commonly used HTTP website information. The security vendor only needs to focus on the collection of the websites which are not on the same cloud, as the websites that are located on the same cloud with the used security company would establish a point-to-point virtual channel during the accessing, which means the network communication could not be hijacked (this was explained in the Methodology Chapter).

However, there are some situations which are more complicated. For example, if a website of a small enterprise is not on a cloud, and they do not have enough resources to establish a virtual safe channel to a security vendor for data transmission, and it still has a frequent daily update, effecting DOM tree changes, then this website may not be whitelisted. For this case, our solution may not be suitable, because the accuracy (timeliness) of the whitelist needs to be considered. Essentially, if the websites in the whitelist are not updated to the latest version, then the comparison is not sound.

Along with the limitations above, we still face some critical issues that need to be further discussed, there are:

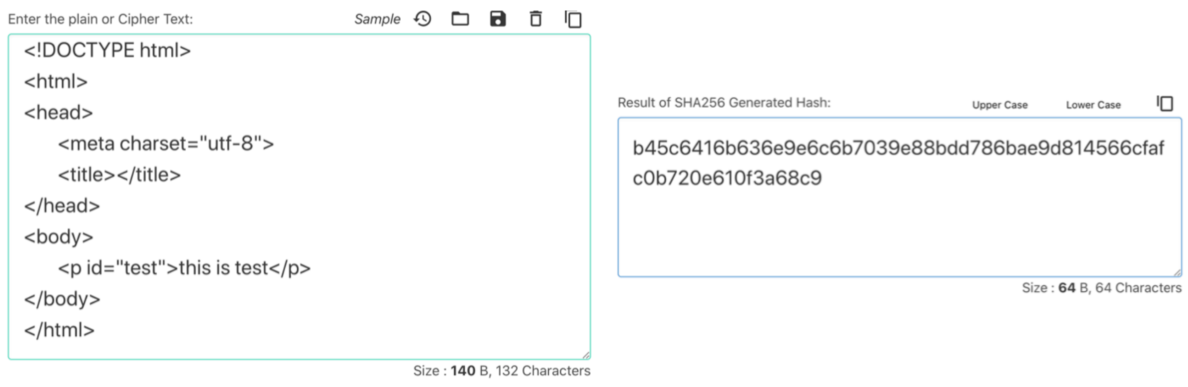
- Whether the DOM archives (reference trees) become a target for hackers?
- Whether the DOM is changed if one element is replaced by another?
- Why this is still a threat since Google shows warnings for HTTP website?
- The webpage may show a different content when users refresh the page, so how to verify the DOM tree in this case?
- The maintenance of a whitelist is expensive, so who and why would they will be willing to support this?

- **Whether the DOM archives become a target for hackers**

The premise of a successful ISP hijacking attack is that the traffic must go through the phisher, otherwise, the phisher cannot implement any malicious activity on this traffic. For the phishers, the easy way to launch this attack is to execute a MITM attack on the victims' router or somewhere before the traffic arrives at the victim’s site. Alternatively, attacks could execute the malicious activity on the traffic source, which is a target website that the victim is accessing, but this is a separate difficulty, as the attacks need to compromise the target’s server. Compared to executing a MITM attack before traffic arrives at the victim, compromising an enterprise’s server is harder and more complicated. In our design, the DOM archives are the reference DOM trees, which is either being stored in the whitelist in the security vendor or the security vendor sends the request to the target website through cloud internal communication. Thus, if the phishers want to tamper with the DOM archives, they require to compromise either the security vendor’s server or a target website’s server, which is a great difficulty as well. From the cost of technology and time consumption, the phishers are unlikely to compromise the DOM archives for implementing an ISP hijacking attack. In addition, even if the phishers successfully compromise either the security vendor’s server or a target website’s server for tampering the DOM archives, this kind of malicious activity would be quickly detected by their security teams comparing victims’ accessed DOM trees.

- **Whether the DOM is changed if one element is replaced by another**

There are some studies that detect the phishing website by comparing the structure of DOM trees, such as [115] [118], which we mentioned in the Literature Review, Chapter 3. Different to our approach, they only focus on the name of each node from the DOM tree, and convert to a vector for later machine learning computation. In our approach, the complete information of each node requires to be considered, which means node name, attributes and node content need to be merged and converted to a hash value during the **Consistency Verification** process. Therefore, any single information changes in the DOM tree, the generated hash value will be affected. For instance, we designed an example webpage, its HTML code and corresponding hash value as shown in Figure 86 below. Then, we modified the original example webpage in two ways. First, we inserted an attribute “class=”test”” to tag <p>. Second, we changed the content of tag <p>, from “this is test” to “this is test!!!”. The generated hash values are changed under both cases, as shown in Figure 87-88 below.


Enter the plain or Cipher Text:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title></title>
</head>
<body>
    <p id="test">this is test</p>
</body>
</html>
```

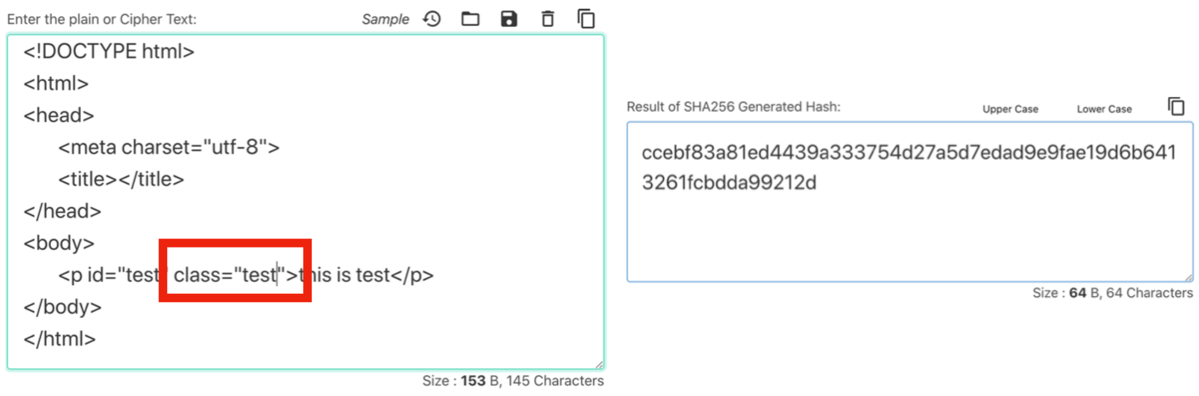
Size : **140** B, 132 Characters

Result of SHA256 Generated Hash:

```
b45c6416b636e9e6c6b7039e88bdd786bae9d814566cfafc0b720e610f3a68c9
```

Size : **64** B, 64 Characters

FIGURE 86: The original example webpage, HTML code and its hash value.


Enter the plain or Cipher Text:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title></title>
</head>
<body>
    <p id="test" class="test">this is test</p>
</body>
</html>
```

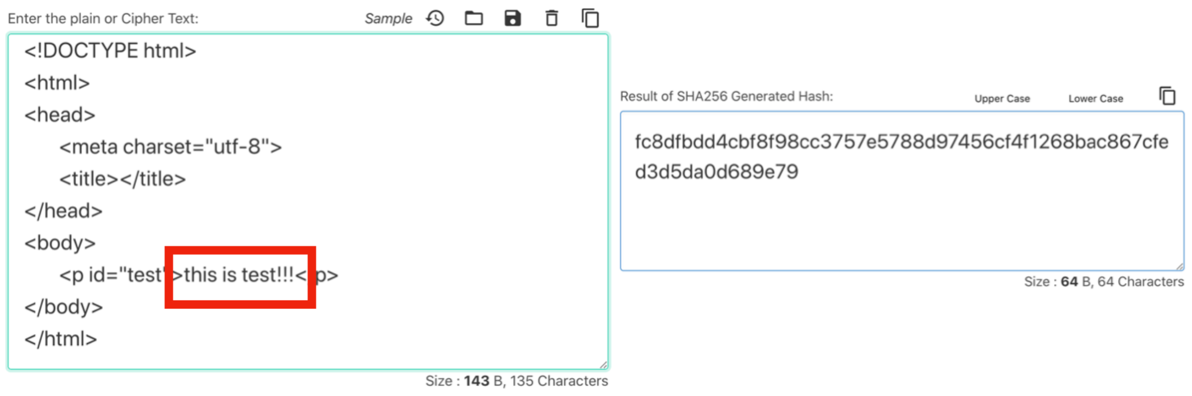
Size : **153** B, 145 Characters

Result of SHA256 Generated Hash:

```
ccebf83a81ed4439a333754d27a5d7edad9e9fae19d6b6413261fcbdda99212d
```

Size : **64** B, 64 Characters

FIGURE 87: Inserted an attribute to tag <p>, HTML code and its hash value.


Enter the plain or Cipher Text:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title></title>
</head>
<body>
    <p id="test">this is test!!!</p>
</body>
</html>
```

Size : **143** B, 135 Characters

Result of SHA256 Generated Hash:

```
fc8dfbdd4cbf8f98cc3757e5788d97456cf4f1268bac867cfed3d5da0d689e79
```

Size : **64** B, 64 Characters

FIGURE 88: Changed node content, HTML code and its hash value.

- **Why this is still a threat since Google shows a warning for HTTP website**

Since the release of Google Chrome version 68 in July 2018, a Chrome browser shows the "not secure website" warning in the address bar for the website that involves HTTP traffic [183]. This warning tells the user that the traffic between the user and the target website is HTTP traffic, rather than having all traffic are under the HTTPS protocol. As previous data has shown, HTTP traffic still has a large usage in the world. From the ProofPoint report in 2019 [175], across all domains, 53% of users use HTTP. Even for the largest global websites, at least 20% of them do not use HTTPS [176]. Moreover, for HTTPS websites, it does not mean all the contents of this website are based on encrypted HTTPS connections. Some content may use an external HTTP request. As shown in the previous example, the website Sohu (available at: *https://sohu.com*), is one of the most famous news websites in China. The website is loaded over an HTTPS connection, but some images are from an external insecure HTTP resource. All of these insecure resources can be replaced with other unexpected content, such as an advertisement, by a phisher, as this connection is over an HTTP.

- **The webpage may show different content when the users refresh the page, so how to verify the DOM tree in this case**

There are various methods to insert the image into an HTML webpage when the developers design the website. For example, images are inserted to the tag of <img> in HTML source; images are inserted to JavaScript file and are displayed with a slideshow; images are from an external API and are called in a JavaScript file. The websites select images and display the images which may be subject to some specific conditions, such as user's location or user's interest. In these cases, the content of the obtained DOM tree is different as the structure of the DOM tree is generated after loading and parsing the HTML response. Thus, if the images are different, the attribute of the image source (paths) are not the same. Therefore, for these cases, our solution requires a proper adjustment with the premise of decreasing security as little as possible.

According to our analysis in the Methodology chapter, Chapter 5, we believe that the highest threat is the alteration of JavaScript file as the webpage function can be changed through JavaScript code. Most malicious activities, such as to redirect user into a malicious website, a CSRF attack, are implemented by using JavaScript. However, images from the different sources would affect the HTML code (displaying different image sources), rather than JavaScript content. Therefore, during the verification process, both path and content of JavaScript file still need to be considered and computed.

The proper adjustment is deployed on the attribute of each image tag <img>. Due to the different image source (path), it is hard to verify whether the accessed website has a benign DOM tree, as the difference exists between the accessed DOM tree and reference DOM tree. So we need to ignore the attribute of each image tag during the verification process, and focus on the nodename in each image tag. However, to ignore the image attribute may result in a potential risk. For example, if the ISP hijacking attack takes place on an image, and phishers insert an alternative *.gif* image, the user may be convinced. Therefore, this is a limitation for our approach if we ignore the image attribute during the verification process.

- **The maintenance of a whitelist is expensive, so who and why would they be willing to do this**

As we discussed in Implementation chapter, Chapter 6, we proposed two deployment models to prevent this kind of hijacking attacks, Prototype to Client (*PtoC*) and Prototype to Business (*PtoB*). Both prototypes require expensive maintenance in either human effort or financial cost. Thus, this approach is more suitable for deployment by an organization, such as a security vendor, rather than an individual. There are many benefits for a security vendor who deploys this approach, because most tools and solutions are good at detecting phishing attacks and DNS hijacking attacks, but there is an inability to detect this kind of ISP (HTTP) hijacking attacks. Therefore, in the model of *PtoC*, users need this service to detect whether their network environment is safe or not. If a security vendor provides this service, it will be welcome indeed. In the model of *PtoB*, any organization does not want their traffic tampered with before arriving at the users, but the potential risk exists if the traffic involves HTTP traffic, and the organization can do nothing about this situation. Thus, if a security vendor provides this kind of protection service to detect the tampered traffic before it arrives at the users, it is hardening the security, which benefits both organization and users.

6. Research on Subdomain Takeover Attacks

In this chapter, the subdomain takeover attack research is discussed. In this attack, the phisher exploits the website configuration flaws in general, which results in both the URL address and the SSL certificate potentially being legitimate. We firstly illustrate the workflow of a subdomain takeover attack. We then explore the current tools and approaches that query website subdomains. The relevant methodologies used to query subdomains and detect the risky subdomains are separately elaborated in subsequent sections. Then, the process of deriving prototypical systems along with the relevant technologies will be introduced and explained in detail. Finally, the evaluation is narrated, and the corresponding challenges faced are analysed and discussed.

6.1. Related Works & Pre-Experiments

In this section, prior to narrating subdomain takeover attacks, three pieces of information are needed. These are:

- The identification of subdomain takeover attacks.
- The querying subdomains approach.
- The comparison of current subdomain enumeration tools.

Firstly, we explain what a subdomain takeover attack is, and how it happens. For the second, the methodology of the subdomain enumeration approach will be introduced. For the final part, the comparison of popular subdomain discovery tools is performed and discussed. At the end of this section, the specific research goals for subdomain takeover attacks will be presented.

6.1.1. The identification of subdomain takeover attacks.

A subdomain takeover is the process of registering a non-existing domain name to gain control over another domain [35] [184]. Before describing this attack, we need to explain how

DNS resolution works with a CNAME record in a benign process. The detailed procedure is shown in Figure 89 below:

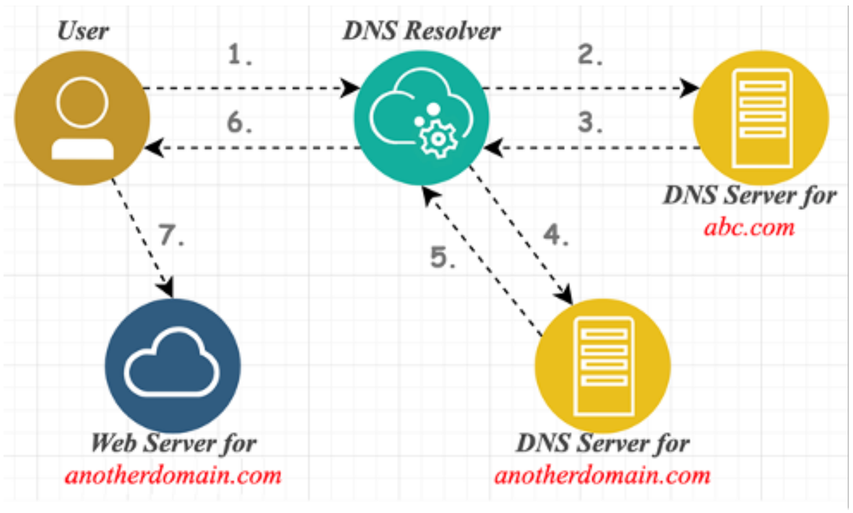


FIGURE 89: An illustration of accessing *subdomain.abc.com* with IP address 1.1.1.1.

Step 1. In order to access the targeted website *subdomain.abc.com*, a user will send a DNS query to a DNS resolver requesting the corresponding IP address of *subdomain.abc.com*.

Step 2. The DNS resolver requests the IP address of *subdomain.abc.com* from the DNS server of *abc.com*.

Step 3. In the DNS server of *abc.com*, a CNAME record has been preliminary configured. The target website, *subdomain.abc.com* is pointed to *anotherdomain.com*, which has the configuration:

subdomain.abc.com **IN *CNAME*** anotherdomain.com

Thus, this DNS server will return a message that says the targeted website has a CNAME record to *anotherdomain.com*.

Step 4. Subsequently, the DNS resolver will send another queried package to the DNS server of *anotherdomain.com* asking for the IP address of *anotherdomain.com*.

Step 5. In the DNS server of *anotherdomain.com*, an A record (which refers to the IP address) can be found regarding the required website *anotherdomain.com*, for example, anotherdomain.com **IN *A*** 1.1.1.1. This IP address will then be returned to the DNS resolver.

Step 6. The DNS resolver will return this IP address, 1.1.1.1, to the user.

Step 7. Finally, the user's browser will establish the connection to *subdomain.abc.com* according to the returned IP address of 1.1.1.1. During this process, the browser requests *subdomain.abc.com* rather than *anotherdomain.com* as the browser is not aware that *anotherdomain.com* even exists. Even though the CNAME record is used, the URL bar in the browser still displays *subdomain.abc.com*.

However, in this procedure, a subdomain takeover risk may exist if the domain *anotherdomain.com* is available for re-registration by anyone else for some reason, such as it has expired or shutdown. A phisher could obtain full control of the website *subdomain.abc.com* through purchasing the available domain *anotherdomain.com* if the previous CNAME setup has not been deleted in the DNS server of *abc.com*. For example, see Figure 90 below.

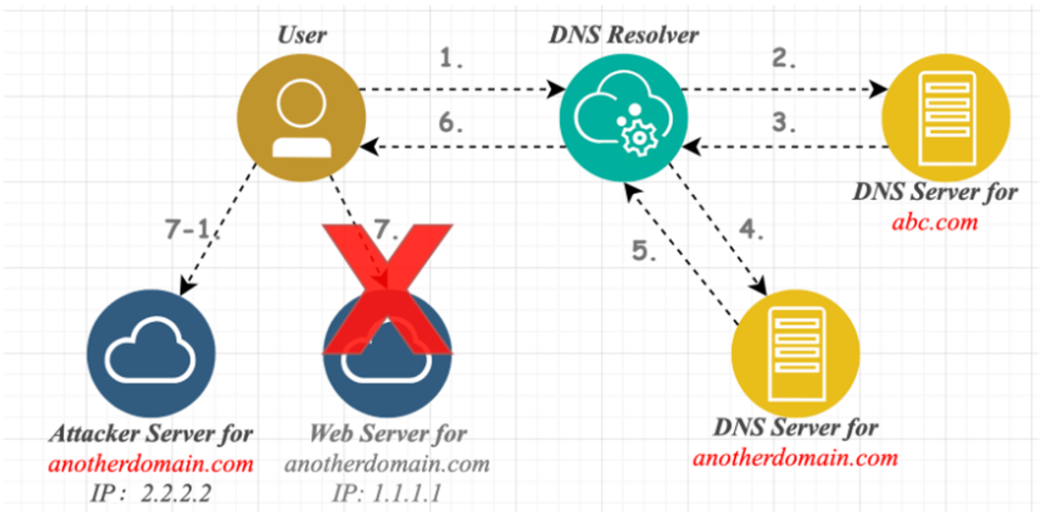


FIGURE 90: An illustration of the takeover of the subdomain *subdomain.abc.com* with IP address 2.2.2.2.

In the DNS server of *anotherdomain.com*, the A record regarding *anotherdomain.com* can be changed if the phisher purchases this domain and configures it to point to their server, in this example, the IP address 2.2.2.2. Subsequently, the DNS server of *anotherdomain.com* will return this IP address, 2.2.2.2, to the DNS resolver.

Finally, the DNS resolver will deliver this IP address to the user's browser and they will establish the connection to the phisher's server (IP address 2.2.2.2) rather than the previous server (IP address 1.1.1.1). This means the targeted website, *subdomain.abc.com*, will be fully controlled by the phisher who purchased *anotherdomain.com* as the CNAME record has not been updated from the DNS server of *abc.com*. Deleting the record will also protect against this attack.

This type of CNAME subdomain takeover attack does not require an advanced technical skill, but it will cause a serious impact on any enterprise. Most organizations prefer to configure **.abc.com* in their issued SSL certificate for establishing an encrypted connection. This is a significant convenience as, through this kind of configuration in their SSL certificate, all subdomains of *abc.com* will be automatically given this certificate. However, a potential risk has been introduced at the same time under this configuration; the phisher could use this SSL certificate and obtain the same permissions once any subdomain is hijacked through a successful CNAME takeover attack.

6.1.2. The querying subdomains approach.

From the detailed explanation above, in order to detect vulnerable subdomains, the critical part is in discovering all the subdomains, otherwise there is a potential risk from the subdomain list not being comprehensive. Borges [185] summarises the usage of popular terminal-based subdomain enumeration tools, the detail in shown in Table 15 below.

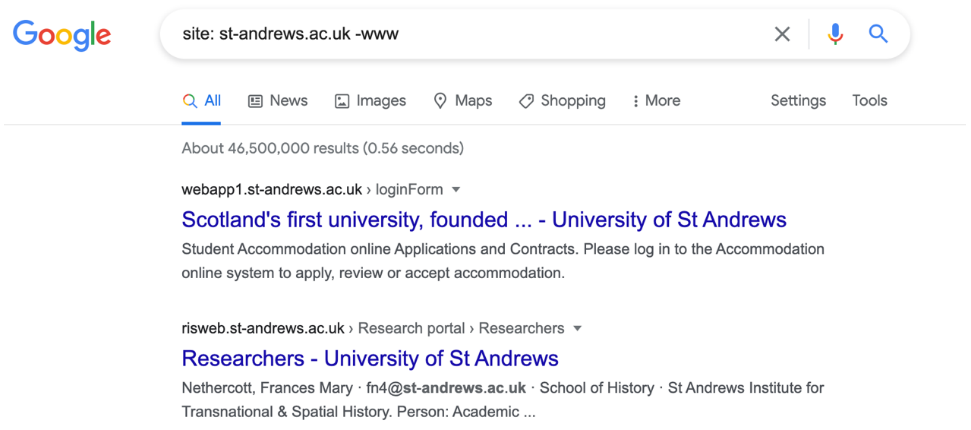
Tools	Usage / Terminal Command
Amass	amass -d example.com
SubBrute	./subbrute.py example.com
Knock	knockpy example.com (knockpy example.com -w wordlist.txt)
DNSRecon	./dnsrecon.py -d example.com
Sublist3r	./sublist3r.py -d example.com

TABLE 15: Popular terminal-based subdomain discovering tools.

The methodology of each subdomain enumeration approach is summarised as follows, and the limitation of each approach is listed in Table 16.

- *Querying subdomain through search engines*

Most subdomains can be gathered using specialised hacking queries. For example, Google hacking queries are often used to gather subdomains from the results of the latest Googlebot crawl, using the command of site: example.com – www, as shown in Figure 91 below. The response is useful for discovering subdomains that are not protected by robots.txt.



site: st-andrews.ac.uk -www

About 46,500,000 results (0.56 seconds)

webapp1.st-andrews.ac.uk › loginForm
Scotland's first university, founded ... - University of St Andrews
Student Accommodation online Applications and Contracts. Please log in to the Accommodation online system to apply, review or accept accommodation.

risweb.st-andrews.ac.uk › Research portal › Researchers
Researchers - University of St Andrews
Nethercott, Frances Mary · fn4@**st-andrews.ac.uk** · School of History · St Andrews Institute for Transnational & Spatial History. Person: Academic ...

FIGURE 91: Google search result by using Google hacking queries.

Some subdomain discovery tools rely on this type of built-in hacking query language to find subdomains from various search engines, such as Google, Bing or Yahoo. However, a challenge exists in this approach; the subdomain that has not been previously queried by search engines would not be found by this method.

- *Discover subdomains through brute force*

In some discovery tools, such as Sublist3r or SubDomainBrute, the recursive brute force approach has been implemented with an extra dictionary (or plugin wordlist) to generate the relevant subdomain names. The applicability of subdomains will be detected through traversing the dictionary. So far, the gathered results have a relatively high completion / hit rate, though the time consumption of this process is the largest of all of the subdomain discovery approaches. However, the generated results are unlikely to be complete as it is still subject to the integrity of the associated dictionary. Also, the full procedure has a higher order time consumption if the dictionary is more comprehensive.

- *Running DNS zone transfers*

The remote DNS zone content can be fully duplicated through the method of DNS zone transfer, and all the configured subdomain information can be revealed. This approach only works when the DNS zone is not protected or limited by the system administrators. Moreover, this approach requires an interface or permission to connect to the remote (primary) DNS.

- *Discover SSL public information*

An SSL certificate is used to encrypt the data during the network connection between the user's browser and servers. It is also a useful tool for penetration information research as within the SSL certificate, further information can be explored. For example, the Subject Alternate Name (SAN) in an SSL certificate can be used to extract domains and subdomain names. However, a potential limitation may exist when discovering subdomains from the SSL certificate. As mentioned above, all subdomains that are under this targeted website (domain) will achieve the same certificate automatically if the organization configures a statement similar to **.example.com* in their issued SSL certificate. This means that the phisher can use this SSL certificate to obtain the same permissions once any subdomain is hijacked through a successful CNAME takeover attack.

Approaches	Limitation
Brute Force	The integrity of discovered subdomains is subject to the used dictionary.
Search Engineering	The website would not be recorded in a search engineering approach if this website has never been previously searched for online.
SSL Certificate	The obtained result may not be complete, some subdomains are expired or not alive.
DNS zone transfer	Some DNS servers need permission to access.

TABLE 16: The limitation of each subdomain discovery approach.

6.1.3. A comparison of subdomain enumeration tools.

The current subdomain enumeration tools associated with one or more functions to query existing subdomain names, some of which were introduced by Borges [185] and Rashid et al [186], include amass, SubBrute, Knock and Sublist3r. Due to their various different functionalities, a security researcher can always pick one or more tools to discover and list the targeted domain.

Jaspher et al [187] provided a comparative study to analyze five subdomain enumeration tools (Amass[37], Subfinder[38], Knock[39], Sublist3r[40] and Altdns[41]), using usage, methodology and time consumption. The tool reconnaissance is classified into five aspects; active information gathering, passive information gathering, brute force, DNS enumeration and the use of search engines. The architectural diagram of each enumeration tool is illustrated in Figure 92 below:

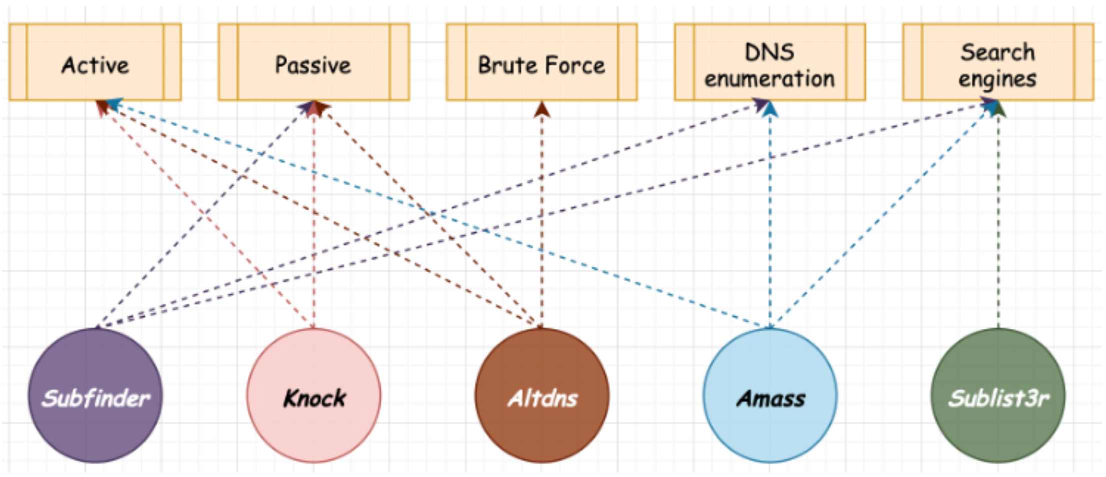


FIGURE 92: The methodology in each enumeration tool used.

[37] Amass: In-depth Attack Surface Mapping and Asset Discovery, designed by OWASP. Available at: *https://github.com/OWASP/Amass*.
[38] Subfinder: It is a subdomain enumeration tool that discovers valid subdomain for a website. The methodology designed as a passive framework, which is useful for bug bounties and safe for penetration testing. Available at: *https://github.com/projectdiscovery/subfinder*.
[39] Knock: Also named as knockpy, designed to discover subdomains on a target domain through dictionary search by python3. Available at: *https://github.com/guelfoweb/knock*.
[40] Sublist3r: It is a python tool designed to discover subdomain of website using OSINT (Open-Source Intelligence), such as search engines (Google, Bing, Yahoo), Netcraft. Available at: *https://github.com/aboul3la/Sublist3r*.
[41] Altdns: It is a DNS recon tool that allows for the discovery of subdomains through alterations and permutations. Available at: *https://github.com/infosec-au/altdns*.

Jaspher et al [187] compared the time consumption of each enumeration tool and indicated that Subfinder is the fastest amongst the five with the final speed ranking as: Subfinder < Sublist3r < Knock < Altdns < Amass. Although the completion of the generated result in these tools was illustrated in the later sections of this paper, the description is not clear and there is not a specific demonstration or explanation to prove their result.

6.1.4. The specific research goal for subdomain takeover attacks.

Overall, research on subdomain enumeration tools or subdomain takeover detection systems is limited as most of the published documentation is focused on the usage and function of the tools. Consequently, in the third part of this research, in order to improve the efficiency and accuracy of current enumeration tools as well as to overcome the risks from subdomain takeover attacks, we focussed on both perspectives. We not only present a novel prediction approach to generate more extraneous content for subdomain discovery, but we also provide an auto-detection system to discover potential takeover risks from subdomains.

6.2. Methodology and Design of Experiments

In order to overcome the threat from a subdomain takeover attack, an automatic detection tool is worth researching, in particular, to support the security maintenance team in any organization. According to the illustration of subdomain takeover detection scheme by Rashid et al [186]. The current approaches to detect this potential vulnerability is concluded involving the following three steps:

- **Querying Subdomains**:

This is done via utilizing subdomain enumeration tools, such as Amass, Massdns or Sublist3r, to collect existing subdomain names. However, in this step, in order to improve the integrity of the result, the combination of two or more enumeration tools are used.

• **Discovering the related information**:

This is discovered from picking the relevant subdomains which involve the CNAME record. (In this experiment, we only focus on CNAME issues).

• **Examining the usability of the destination domain**:

This is done by confirming the usability of the destination domain from collected CNAME records. Any destination domains which are without an IP address would be labelled as risky. Risky subdomains may be registerable in a Name Vendor, such as Namecheap[42]. Vulnerable subdomains can be identified if their destination domain is available (to be registered) in a Name Vendor.

However, there is not any available tools and platforms to fully effect these manipulations for auto-detecting this potential vulnerability. For either novices, security teams or bonus hunters, the detection effectiveness will increase significantly by implementing an auto-system, not only to reduce the manipulation complexity, but also to improve the accuracy of the result. In our research, we present a scheme that combines the steps of querying, discovering and auto examining potential risks in subdomains. Moreover, in order to overcome the limitation of a dictionary (for domains) being based on known words, we generate potential existing subdomains through Char-RNN [188], which is a well-designed machine learning model.

Thus, this section involves two sub-sections to describe the Char-RNN model and our auto-detection tool. The methodology will be explained in detail from the theoretic principles to the technique adopted in this prototypical research.

6.2.1. The Char-RNN model

RNN is a Recurrent Neural Network that is based on time cycles, constructed as sequences for recognition. For example, a text or a speech signal has internal cycles that indicate the

[42] Namecheap: A online domain vendor, it makes registering, hosting, and managing domains for users. Available at: *https://www.namecheap.com/*.

presence of short-term memory in the network. RNN often works on handling sequence issues and involves three layers; the input layer, a hidden layer and an output layer [188]. The cell of RNN is circulative in a neural network, as shown in Figure 93 below:

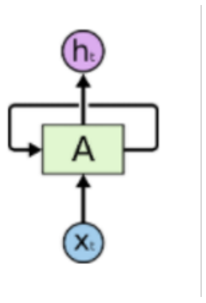


FIGURE 93: The architecture of an RNN cell **[189]**.

In the RNN model, the input sequence x_t is encoded to a fixed-length hidden state at time *t*. *A* is a hidden state that is updated with the passage of time. In the cyclic part, the data is transferred from one step to the next one within the neural network. A accepts the input data from x_t and uses cyclic data from the previous *t-1* state to then generate an output h_t. In this way, the RNN cell retains the text sequence message. Also, the passage of time is considered within the RNN model; the weight of earlier input sequence is degrading during iterations.

Both Char-RNN and Word-RNN belong to the Recurrent Neural Network model, the former results in the text generation result based on character-by-character analysis and the latter results in the text generation result based on word-by-word analysis. Char-RNN has longer time steps that are needed to consider the dependencies amongst more text tokens. Thus, a larger hidden layer is required in Char-RNN, since the hidden state persists character relationships and models short term dependencies between them. The advantage of Char-RNN is that the character level model does not need to deal with much vocabulary as the count of unique characters is much less than the count of unique words. Thus, Char-RNN is an appropriate way for the subdomain generation with limited string length as the components of subdomain names can usually be set to characters, such as shop, edu, vpn etc [190]. Also, the components of a subdomain may be some words that do not make any sense, such as *abc* or *yw43*. The present dictionary cannot cover this situation. Therefore, in our experiment, we adopted Char-RNN to predict the potential subdomain names and to generate a better dictionary. Here, the classic N vs N model was utilized in Char-RNN, with the input a sequence of length N, and the output is a sequence of the same length. In the Char-RNN model, the input sequence is a letter in a sentence and the output is the next letter, and hence this model is used to predict the probability of the next letter until a terminator symbol is predicted, which means this prediction round is finished.

Therefore, the construction of an N vs N RNN model is generated after executing the cycles above, as shown in Figure 94 below:

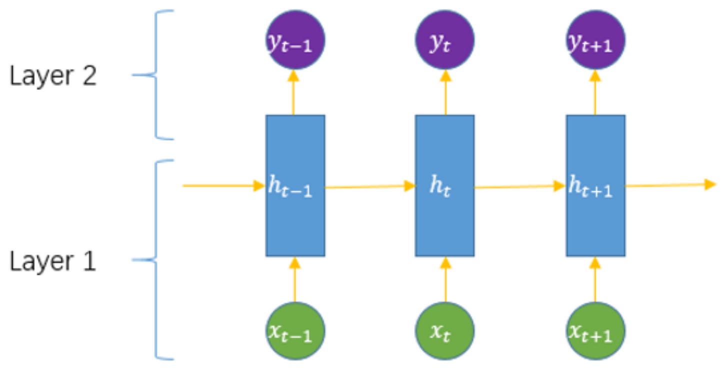


FIGURE 94: The architecture of N vs N in an RNN model **[189]**.

The following equation is used between the input layer and the hidden layer, which is shown in the process of Layer 1:

$$h_t = f\ (Ux_t + Wh_{t-1} + b)$$

Where;

- U is the matrix of $n*m$, which represents the weight[43] from the input layer to the hidden layer.
- x_t is the matrix of $m*1$, which represents the current input.
- W is the matrix of $n*n$, which represents the weight between the hidden layers.
- h_t is the matrix of $n*1$, which is the hidden state at the time of t.

[43] Generally, weight is the learnable parameters of the model, control the influence of the inputs on the hidden layer.

- h_{t-1} is the matrix of *n*1*, which is the hidden state at the time of *t-1*.
- *b* is the vector matrix, which is a bias[44] parameter.
- *f* is the activation function.

The following equation is used between the hidden layer and the output layer, which is shown in the process of Layer 2:

$$y_t = Softmax\ (Vh_t\ +\ c\ )$$

Where;

- *V* is the matrix of *n*n*, which represents the weight[45] between the hidden layer to the output layer.
- *c* is the vector matrix, which is a bias parameter.
- y_t is the current output.

For example, for a simple word Hello, the input sequence is {**H,e,l,l,o**}, and the output sequence is {**e,l,l,o,!**}. These two sequences have the same length, thus RNN N vs N model is used to model this case.

During the process of sequence generation, a character (x_1) is selected (in the case of **Hello**, it is "*h*") and set up to read the initial letter. The probability of the next corresponding character (x_2) is obtained by using the trained model. In order to output the relevant possibility, the Softmax[46] activation function is used in the output layer of the model. A character will be output (in this case, the most probable value refers to letter "*e*") based on this function. Subsequently, this letter "*e*" will be the next input for x_2 and used to execute the next prediction that generates the next letter. A string is generated by repeating this procedure until the last step. In the last step, the input is letter "*o*", and the next prediction character is a terminator symbol, which means this round prediction is finished.

[44] A constant which is an additional input to the next layer.
[45] Control the influence of each neuron in hidden layer on the output layer.
[46] Softmax: A built-in function of classification problem in machine learning, it is responsible for converting the vector to the probabilities.

6.2.2. The description of an auto-detection system

Prior to narrating the schema for a subdomain auto-detection system, a comparative experimentation was conducted to examine five popular subdomain enumeration tools. From this comparison, we identify the enumeration tool which has the best performance in either efficient computing speed or accurate subdomain results. Subsequently, an enumeration tool will be selected and re-developed for the next experiment. The queried subdomains will be examined for a destination domain according to their specific DNS information.

In order to examine the CNAME takeover attack, we only focus on the usability of final destination domains as the risk often comes from the last CNAME record / address. We assumed two scenarios to explain this threat. For example, the following DNS information is generated from the tool dig, as in **dig edu.abc.com**:

Scenario 1 (S1):

edu.abc.com **IN *CNAME*** edu.yw43.com

edu.yw43.com **IN *CNAME*** edu.xz32.com

edu.xz32.com **IN *CNAME*** stafocus.com

stafocus.com **IN *A*** 10.10.10.10

The CNAME may be configured from a subdomain to another subdomain (or domain), such as the DNS configured information from Scenario 1. From the original subdomain *edu.abc.com* CNAME points to another subdomain *edu.yw43.com*, and the next CNAME to another subdomain *edu.xz32.com*, which the CNAME associates with another domain *stafocus.com*. Finally, the domain *stafocus.com* is assigned to an IP address 10.10.10.10 from the A record. This is a normal DNS configuration of subdomain *edu.abc.com* and the process does not show any potential CNAME takeover risks.

Scenario 2 (S2):

edu.abc.com **IN *CNAME*** edu.yw43.com

edu.yw43.com **IN *CNAME*** edu.xz32.com

edu.xz32.com **IN *CNAME*** stafocus.com

There is a potential takeover risk under Scenario 2 as the final CNAME destination domain *stafocus.com* is not assigned with an A record. Three reasons may cause this phenomenon:

- **Reason 1 (R1)**: The Network administrator forgot to assign an IP address to the domain of *stafocus.com*.

- **Reason 2 (R2)**: The IP address of domain *stafocus.com* is expired and recycled by a cloud vendor.

- **Reason 3 (R3)**: The domain *stafocus.com* is expired and recycled by a Name vendor. This causes the previous DNS configuration of this domain to be emptied.

The takeover attack appears in R2 and R3. In R2, as an IP address will be randomly assigned after purchase from a cloud vendor. So, this IP address will be available and assigned to another purchaser if the previous purchaser's rights had expired. The assignment of an IP address is random, so this takeover risk exists if the new purchaser is assigned to a previous IP address, although the probability is rare. Unlike an IP address, an available domain can be freely registered from a Name vendor as the specific name is selectable. Compared to R1 and R2, R3 has the highest success rate. Anyone can take over this domain if the domain is expired from the previous purchase.

However, in both scenarios, what happens if the domain of *xz32.com* is shut down (either expired or discarded). The configuration under this domain will be emptied, any DNS information is removed, which means the next CNAME record *edu.xz32.com* CNAME *stafocus.com* does not exist. Thus, the result of **dig edu.abc.com** in both scenarios becomes:

edu.abc.com **IN *CNAME*** edu.yw43.com

edu.yw43.com **IN *CNAME*** edu.xz32.com

In this case, we use the command **dig xz32.com** to verify the existence of the domain *xz32.com* and then attempt to register this domain with any Name vendor to identify whether it is available. Therefore, the risk is always determined from the CNAME record, and the usability of the final destination domain is the key to evaluating a subdomain takeover threat. The specific implementation will be demonstrated in the next chapter.

6.2.3. Detailed research implementation

The aim of this part of the research involves two actions. Firstly, the subdomain dictionary is predicted by using the Char-RNN model. Secondly, an auto-detection tool to discover the potential risky subdomains has to be developed. The implementation of this part of the research is divided into three steps:

- To compare the current subdomain enumeration tools to obtain an optimal approach (or tool)
- To generate a new dictionary via the Char-RNN model to predict usable subdomains.
- To design and develop an auto-detection tool to discover the potential risky subdomains.

6.3. Implementation

In this section, the third experiment's implementation is divided into two parts: the implementation of querying subdomain using the Char-RNN model and the implementation of an auto-detection system.

6.3.1. The implementation of querying subdomains using the Char-RNN model

In this experiment, the training data of the Char-RNN text generation model was taken from an open resource in GitHub [191]. The Char-RNN model was discussed in the Methodology Chapter and is used to predict potential subdomain names.

Data Pre-processing

In the structure of a domain name, there are various implications for each level of label, and the character distribution is quite different. For example, the common top-level domain includes IP suffixes such as *.com*, *.net*, *.org* etc. Second-level domain involves companies such as Baidu, Facebook, Google, etc. The third-level domain names contain lower level directories as used by organisations: news, game, sports. The dataset of domain characters is summarised in Table 17 below.

Classification	Character Set
Number	0 - 9
Letter	a - z, A - Z
Specific Character (Symbol)	-, _

TABLE 17: The character sets used in domain names.

From this table, three classifications are involved: Number, letter and specific character (symbol). Thus, as the name of domains are not case sensitive, the domain character dataset can be considered as a total of 38 characters with the initial character only allowed to be a number or a letter, according to domain name generation regulations [192].

Text Vectorization

The dataset of the gathered subdomain (training data, open source, see [191]) is composed of various strings and the input of the Char-RNN model must be a vector as text data processing cannot be used directly in the machine learning model. Thus, it is necessary to encode each character. Here, the vector of one-hot encoding is utilized to represent the character in the dataset.

One-hot encoding is a simple and widely used encoding method [193]. For example, there is a string [‘shop’], in order to convert it into vectors, this string needs to be read as a series of characters, such as [‘s’, ‘h’, ‘o’, ‘p’]. Next, each character is coded with 28-dimensional feature vectors as there are 26 English letter (ignore case sensitive) and two specific symbols (‘-’ and ‘_’). So, in each character, we have a 26-dimension vector, such as [0, 1, 0, 0, 0, …, 0] for the character ‘b’. Finally, the string [shop], will be represented to [shop] -> [[], [], [], []], as shown in Figure 95 below.

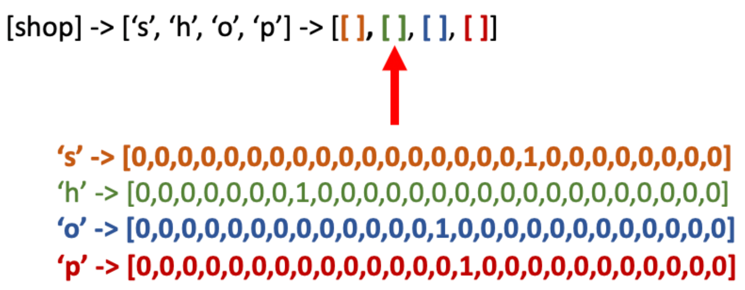

[shop] -> [‘s’, ‘h’, ‘o’, ‘p’] -> [[ ], [ ], [ ], [ ]]
‘s’ -> [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0]
‘h’ -> [0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
‘o’ -> [0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0]
‘p’ -> [0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0]

FIGURE 95: An example of converting string [shop] to vectors using one-hot.

After encoding the characters, the vector representation results of all unique characters are obtained and stored in the form of a dictionary. In this way, before a subdomain is entered into the model, the input system needs to check the dictionary to get its related vector representation, and then the sequence of vectors would be entered into the model for training.

Model training and implementing

Firstly, during model training, the weight matrix of each layer (mentioned in the Methodology Chapter, Chapter 5) is randomly initialized. Subsequently, the main prefix part of a domain name is set to the input text data. For example, in the case of *app.baidu.com*, app is the prefix of this domain. The input to the dataset comes from this part of the subdomain, rather than the entire domain name as we need to predict the subdomain part. This model considers the relationships between each used character as well as predicting the next character for each character input.

Based on the original text data, we will add a start symbol <s> and a terminator symbol <e> to the string. The architecture of the Char-RNN is many-to-many, where the sequence of inputs is maintained (the order information of the input characters is kept implicitly) and outputs are given simultaneously. Thus, for a subdomain string prediction problem, for example, the input series is a subdomain string and each character in this string is fed into a neural network which generate one character output at a time based on the characters that have come in sequence before. This will then allow us to generate a new subdomain string.

More concretely, as shown in Figure 96, we predict the next character in the string "apple" at each time step. The label is the output data that generated from the character through trained model, the character is the input data that is same to the last label. We also add two extra tokens "start of sequence" <s> and "end of sequence" <e> for effective model training. During training, the initial input token is the start-of-string <s> token, every time the model predicts a word we add it to the output string, and if it predicts the <e> token we stop this round prediction.

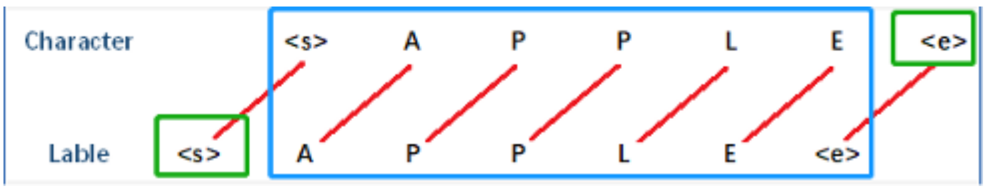


FIGURE 96: The explanation of training process.

Subsequently, further inference is conducted by using the trained RNN model. The initial domain letter can be set to any specific letter or be random. The length of the generated label is controllable as well. For example, if this length is set to 1000, the final count of the generated result would be 1000. This length is an option parameter, the generated result is unlimited and random if this setup is empty.

6.3.2. The implementation of an auto-detection system for risky subdomains

Prior to starting the implementation, we used five subdomain enumeration tools (Amass, Massdns, Knockpy, Sublist3r, SubDomainBrute) to discover subdomain information from the targeted domain. A remote server was purchased to implement this experiment (we initially implemented this experiment in a virtual machine, but the time consumption was large as the specification in the virtual machine was lower, hence we shifted this experiment to our remote server). The setup of this server was: CPU with 1 vCore, RAM with 1024 MB and a Storage with 25 GB SSD.

Next, the gathered content (subdomains) is recorded for further comparison, and the time consumption for each subdomain enumeration tool is shown in Table 18 below. In this experiment, I wanted to select UK websites as a preliminary test. As an overseas student in the UK, I am more familiar with university websites, thus, I selected my own University's website, and four other famous UK University websites for this comparative experiment. Also, as the university consists of the various departments, thus the count of subdomain may be large. The sites are: University of St Andrews, University of Cambridge, University of Oxford, Imperial College London and University College London.

Domain	Amass	Massdns	Knockpy	Sublist3r	SubDomainBrute
StA	> 30 m	ct: 7.4 s	7 m 32 s	22 m 7 s	11 m 24 s
Cam	> 30 m	ct: 9.2 s	1 m 29 s	17 m 26 s	15 m 18 s
Ox	> 30 m	ct: 7.1 s	2 m 49 s	18 m 28 s	28 m 33 s
IC	> 30 m	ct: 10.6 s	3 m 24 s	11 m 38 s	5 m 56 s
UCL	> 30 m	ct: 13.2 s	3 m 18 s	13 m 33 s	5 m 27 s

TABLE 18: Time consumption for various targeted domains using each enumeration tool.

Where:

- The domain of **StA**: *st-andrews.ac.uk*.
- The domain of **Cam**: *cam.ac.uk.*
- The domain of **Ox**: *ox.ac.uk.*
- The domain of **IC**: *imperial.ac.uk.*
- The domain of **UCL**: *ucl.ac.uk.*

According to the results from our experiment, the most efficient approach is based on the querying of the SSL certificate, although most subdomains may not be alive in this method. Some subdomains that are obtained from checking the SSL certificate are not accessible and their traffic is not reachable. The most accurate (detailed, or completed) method is Brute Force, though the generated result is subject to the integrity and comprehensive of the dictionary.

Therefore, the first conclusion we reached, regarding these subdomain enumeration tools, is that SubDomainBrute has the highest performance in either efficient computing speed or in subdomain result accuracy. Amass also provided a comprehensive subdomain result, though it will consume a lot of time. The transparency of the SSL certificate is utilized in Massdns, but most generated subdomain names are not alive. In consequence of these results, we referred to the framework and methodology of the SubDomainBrute enumeration tool to redesign the workflow of discovery process for our further experiments.

Subsequently, two available domains (*stafocus.com* and *ly1026.me*) were registered from *Namecheap.com*, and the specific DNS configurations are shown in Tables 19 and 20 as below. Where, the value is the result of type, for example, in URL *stafocus.com*, it has an A record, its value is 68.183.32.17. In URL *blog.stafocus.com*, it has an CNAME record, its value is *liziling98.com*.

The domain of: *stafocus.com*			
Type	**Host**	**URL**	**Value**
A	@	*stafocus.com*	68.183.32.17
CNAME	blog	*blog.stafocus.com*	*liziling98.com*
CNAME	cs	*cs.stafocus.com*	*hubs.sta.social*
CNAME	edu	*edu.stafocus.com*	*allways-cam.com*
CNAME	education	*education.stafocus.com*	*allways-cam.com*
CNAME	pay	*pay.stafocus.com*	*ly1026.me*
CNAME	payment	*payment.stafocus.com*	*ly1026.me*
CNAME	search	*search.stafocus.com*	*baidu.com*
CNAME	vpn	*vpn.stafocus.com*	*allways-cam.com*
CNAME	vpn1	*vpn1.stafocus.com*	*ly1026.me*

TABLE 19: Advanced DNS configuration in domain: *stafocus.com*.

(**NB**: The URL of *liziling98.com* is a personal blog, and a traffic redirection (301) has been configurated in this server; the URL of *hubs.sta.social* is a social website for the HCI group in the School of Computer Science at the University of St Andrews; The URL of *baidu.com* is one of most famous search engines in China; The URL of *allways-cam.com* is a website for an education organization; The URL of *ly1026.me* is a domain we registered for further experimentation.)

The domain of: *ly1026.me*			
Type	**Host**	**URL**	**Value**
A	@	*ly1026.me*	45.32.49.95
CNAME	blog	*blog.ly1026.me*	*blog.staofcus.com*
CNAME	edu	*edu.ly1026.me*	*edu.stafocus.com*

TABLE 20: Advanced DNS configuration in domain: *ly1026.me*.

According to the DNS configuration of the test registered domains noted above, we attempted to execute the terminal command **dig** to reveal the map relationship for each subdomain. The result is summarized as follows:

The stafocus.com domain:

stafocus.com **IN *A*** 68.183.32.17

blog.stafocus.com **IN *CNAME*** liziling98.com **IN *A*** 149.248.36.25

cs.stafocus.com **IN *CNAME*** hus.sta.social

edu.stafocus.com **IN *CNAME*** allways-cam.com

education.stafocus.com **IN *CNAME*** allways-cam.com

pay.stafocus.com **IN *CNAME*** ly1026.me **IN *A*** 45.32.49.95

payment.stafocus.com **IN *CNAME*** ly1026.me **IN *A*** 45.32.49.95

search.stafocus.com **IN *CNAME*** baidu.com **IN *A*** 39.156.69.79 (220.181.38.148)

vpn.stafocus.com **IN *CNAME*** allways-cam.com

vpn1.stafocus.com **IN *CNAME*** ly1026.me **IN *A*** 45.32.49.95

The ly1026.me domain:

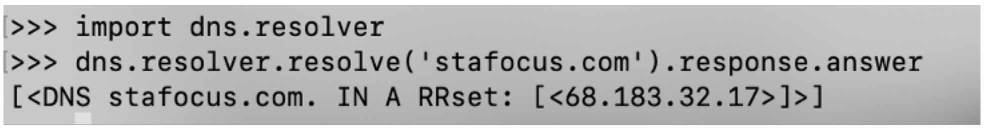
ly1026.me **IN *A*** 45.32.49.95

blog.ly1026.me **IN *CNAME*** blog.stafocus.com **IN *CNAME*** liziling98.com **IN *A*** 149.248.36.25

edu.ly1026.me **IN *CNAME*** edu.stafocus.com **IN *CNAME*** allways-cam.com

From the mapping relationships above, it appears that both domains have potential subdomain takeover risks. In the *stafocus.com* domain, the risks may exist in the subdomain of *cs.stafocus.com*, *edu.stafocus.com*, *education.stafocus.com* and *vpn.stafocus.com*. In the domain of *ly1026.me*, risks may exist in the subdomain of *edu.ly1026.me*. Thus the purpose of our system is to identify these potentially risky subdomains from the target domain.

In most subdomain enumeration tools, a python library, named dnspython, has been used to resolve targeted domains to reveal the relevant DNS information. For example, as shown in Figure 97 below, the domain stafocus.com is resolved through calling the function *dns.resolver.resolve()* and using the *.response.answer* function to show the result.



```
>>> import dns.resolver
>>> dns.resolver.resolve('stafocus.com').response.answer
[<DNS stafocus.com. IN A RRset: [<68.183.32.17>]>]
```

FIGURE 97: The use of dnspython library to resolve domains

However, an exception will be notified if an A record does not exist in the targeted domain. For example, under the following scenarios we see different results.

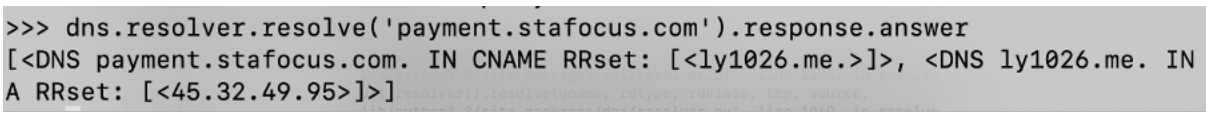
Scenario 1:

payment.stafocus.com **IN *CNAME*** ly1026.me **IN *A*** 45.32.49.95

Scenario 2:

edu.stafocus.com **IN *CNAME*** allways-cam.com

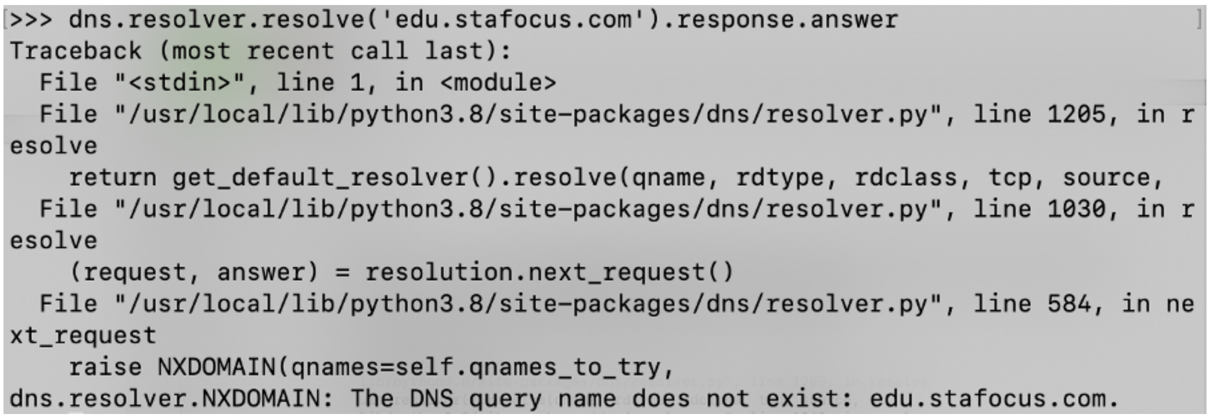
A correct and complete result is resolved from the targeted website *payment.stafocus.co*m by using dnspython in Scenario 1, as shown in Figure 98 below:



```
>>> dns.resolver.resolve('payment.stafocus.com').response.answer
[<DNS payment.stafocus.com. IN CNAME RRset: [<ly1026.me.>]>, <DNS ly1026.me. IN
A RRset: [<45.32.49.95>]>]
```

FIGURE 98: The DNS resolution of *payment.stafocus.com* in the dnspython library.

In Scenario 2, an exception is shown if we resolve the targeted website *edu.stafocus.com*. This is because an A record does not exist in the destination address *allways-cam.com*, as shown in Figure 99 below:



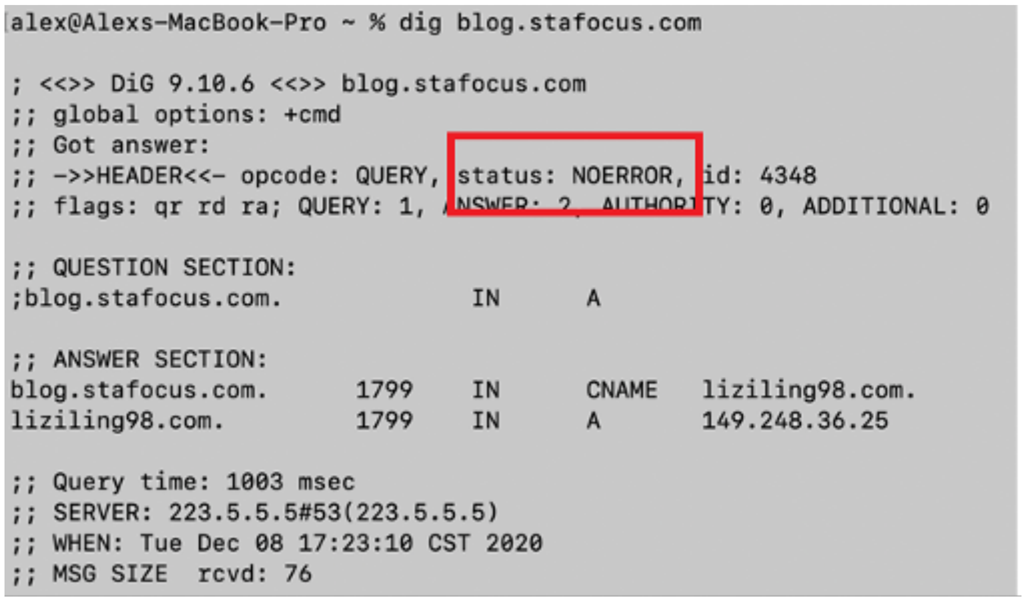
```
>>> dns.resolver.resolve('edu.stafocus.com').response.answer
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python3.8/site-packages/dns/resolver.py", line 1205, in r
esolve
    return get_default_resolver().resolve(qname, rdtype, rdclass, tcp, source,
  File "/usr/local/lib/python3.8/site-packages/dns/resolver.py", line 1030, in r
esolve
    (request, answer) = resolution.next_request()
  File "/usr/local/lib/python3.8/site-packages/dns/resolver.py", line 584, in ne
xt_request
    raise NXDOMAIN(qnames=self.qnames_to_try,
dns.resolver.NXDOMAIN: The DNS query name does not exist: edu.stafocus.com.
```

FIGURE 99: The exception in the dnspython library.

Therefore, for most current subdomain enumeration tools, there is an appropriate option to query relevant subdomains, rather than for discovering a risky subdomain. Any subdomain that does not have an A record will be discarded during the collection procedure. In addition, these discarded addresses may involve non-existent addresses as well, so it is hard to identify risky addresses.

For example, if we have strings [*abc, aa, bb, pay, edu*] in the dictionary, and the targeted domain is *sta.com*, then, using brute force, each string in the dictionary will combine with the targeted website (we have [*abc.sta.com*, *aa.sta.com*, *bb.sta.com*, *pay.sta.com* and *edu.sta.com*]) and then each is resolved separately. However, only the address that return a correct result will be stored. Which means, if *edu* is an existing (with no risk) subdomain, *pay* is a risky subdomain which has a takeover issue, and *abc*, *aa* and *bb* do not exist, after querying by the enumeration tools, only *edu.sta.com* will be stored and displayed to user. The non-existent addresses, *abc*, *aa* and *bb* are discarded, and if the risky subdomain *pay* cannot be resolved to an A record, this will be discarded as well. Thus, in these discarded addresses, it is hard to identify which has the takeover risk as there may be too many addresses if the length of original dictionary is large.

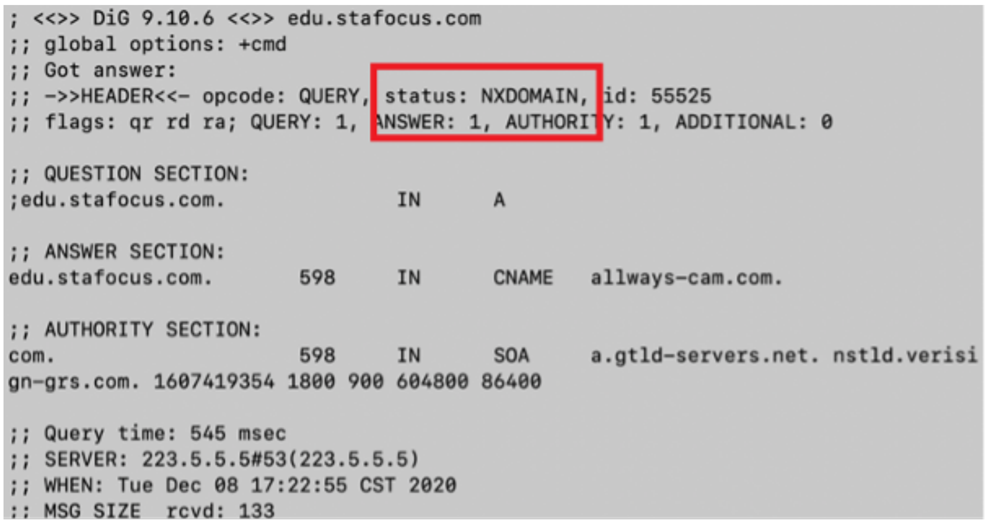
In the first step of our auto detection system, in order to address the discarding issues, we utilize the command **dig** to query the subdomains. Then, key words such as “status: NOERROR” and “CNAME” are used to confirm whether the queried address exists, and we gather the related subdomain information. For example, for a correct address, the status will display NOERROR, as shown in Figure 100 below:



```
alex@Alexs-MacBook-Pro ~ % dig blog.stafocus.com

; <<>> DiG 9.10.6 <<>> blog.stafocus.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4348
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;blog.stafocus.com.             IN      A

;; ANSWER SECTION:
blog.stafocus.com.      1799    IN      CNAME   liziling98.com.
liziling98.com.         1799    IN      A       149.248.36.25

;; Query time: 1003 msec
;; SERVER: 223.5.5.5#53(223.5.5.5)
;; WHEN: Tue Dec 08 17:23:10 CST 2020
;; MSG SIZE  rcvd: 76
```

FIGURE 100: An example of dig *blog.stafocus.com*.

However, for either a non-existent or risky address, the status shows NXDOMAIN, which is shown by the red rectangle in Figure 101 below. In our experiment, the purpose is to prevent the CNAME takeover issue, therefore, in our system, the key word “CNAME” functions as a filter when gathering these risky addresses in the ANSWER SECTION.



```
; <<>> DiG 9.10.6 <<>> edu.stafocus.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 55525
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;edu.stafocus.com.              IN      A

;; ANSWER SECTION:
edu.stafocus.com.       598     IN      CNAME   allways-cam.com.

;; AUTHORITY SECTION:
com.                    598     IN      SOA     a.gtld-servers.net. nstld.verisi
gn-grs.com. 1607419354 1800 900 604800 86400

;; Query time: 545 msec
;; SERVER: 223.5.5.5#53(223.5.5.5)
;; WHEN: Tue Dec 08 17:22:55 CST 2020
;; MSG SIZE  rcvd: 133
```

FIGURE 101: An example of dig *edu.stafocus.com*.

In the second step, we still use the key word "CNAME" to filter the relevant subdomains which involve the CNAME record from the step above. In the last step, we utilized the python library dnspython to resolve the gathered subdomains that match the key word condition in the second step. For each examined subdomain, the risky subdomain can be identified and evaluated if an exception has been notified. Therefore, all risky subdomains will be listed, and the relevant information will be displayed to the user. The specific workflow of our auto-detection system is illustrated in Figure 102 below:

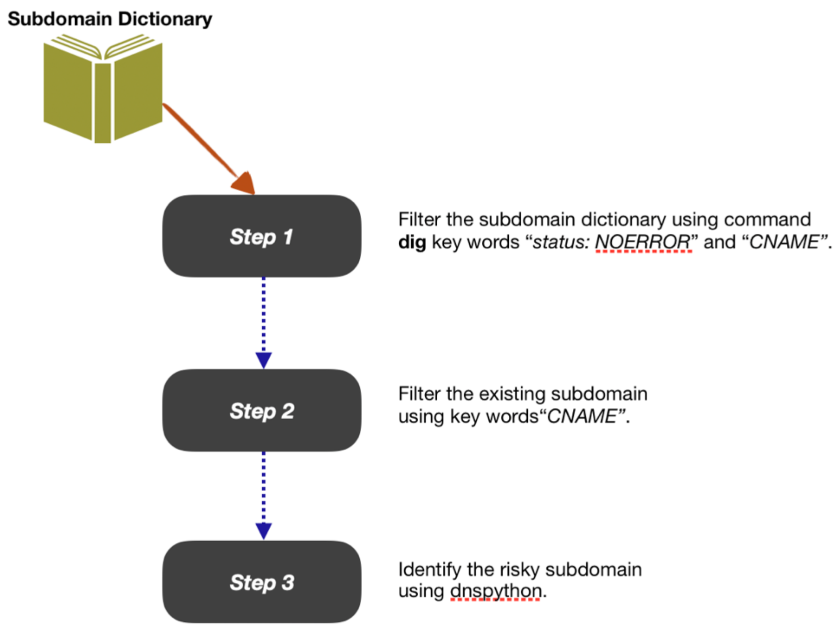

Subdomain Dictionary
Step 1
Filter the subdomain dictionary using command dig key words "status: NOERROR" and "CNAME".
Step 2
Filter the existing subdomain using key words"CNAME".
Step 3
Identify the risky subdomain using dnspython.

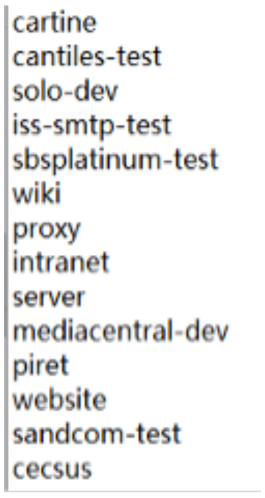
FIGURE 102: The workflow of our auto-detection system.

6.4. Evaluation

In this section, the evaluation of a novel subdomain prediction approach and an automated risky subdomain detection system will be considered. Firstly, the applicability of our presented novel subdomain enumeration approach which is based on the Char-RNN model is evaluated. Next, the accuracy of our auto-detection system is verified by discovering the risky subdomains on our websites.

The Evaluation of Querying Subdomains using the Char-RNN Model

In our experiment, we set a random character as the initial letter, rather than a specific one for the start of the Char-RNN input. The final generated result is either a known word or a strange letter combination, a sample of predicted subdomains is shown in Figure 103 below.

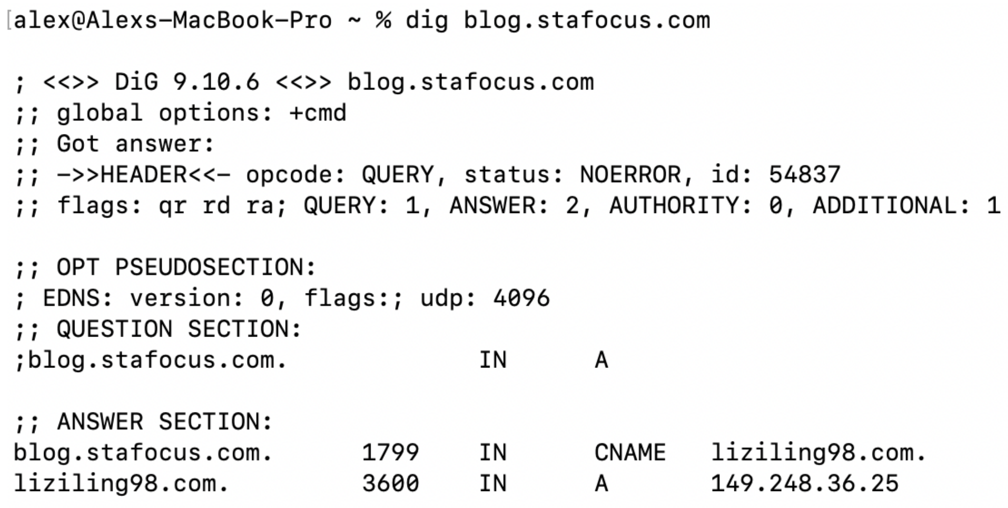


```
cartine
cantiles-test
solo-dev
iss-smtp-test
sbsplatinum-test
wiki
proxy
intranet
server
mediacentral-dev
piret
website
sandcom-test
cecsus
```

FIGURE 103: The output of predicted subdomains using our querying subdomains approach using the Char-RNN model.

In order to evaluate the accuracy of our subdomain predictions, we set various string lengths during the implementation, using different string sizes respectively. Subsequently, these subdomains will be combined with the targeted website, and we verified the *live rate* by using the command **dig**. For example, if the targeted domain is *stafocus.com*, and a predicted word / subdomain is *blog*. Then we merged the predicted subdomain and the targeted domain to a completed subdomain address, which is *blog.stafocus.com*. We then queried the information in the “QUESTION SECTION” and “AUTHORITY SECTION” using the command **dig**

blog.stafocus.com. In this case, we can verify whether the subdomain *blog.stafocus.com* exists (is alive) and the result is shown in Figure 104 below.

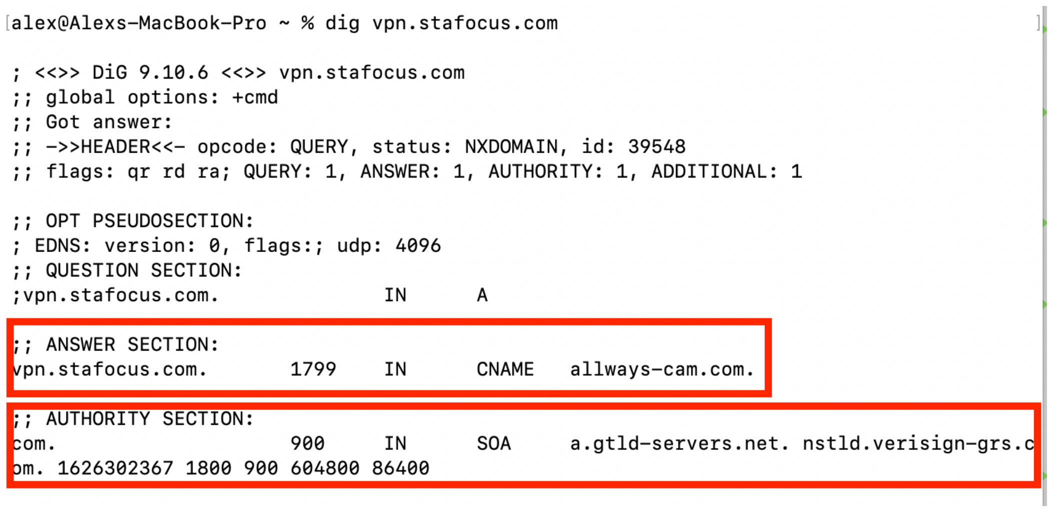


```
[alex@Alexs-MacBook-Pro ~ % dig blog.stafocus.com

; <<>> DiG 9.10.6 <<>> blog.stafocus.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54837
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;blog.stafocus.com.             IN      A

;; ANSWER SECTION:
blog.stafocus.com.      1799    IN      CNAME   liziling98.com.
liziling98.com.         3600    IN      A       149.248.36.25
```

FIGURE 104: The result of the command dig *blog.stafocus.com* in the laptop terminal.

If the "AUTHORITY SECTION" includes a key word "SOA[47]", and there are some detailed records in the "ANSWER SECTION", which means this CNAME record exists, but the destination address (*allways-cam.com)* is not regularly used. As shown in Figure 105 below, the result of applying the command **dig vpn.stafocus.com**, shows that this subdomain has a CNAME record, pointing to the domain *allways-cam.com*, but the domain *allways-cam.com* is without an A record or other further configuration, so it results in SOA being stated in the "AUTHORITY SECTION". In this case, the subdomain *vpn.stafocus.com* is existed, but it may have risks.

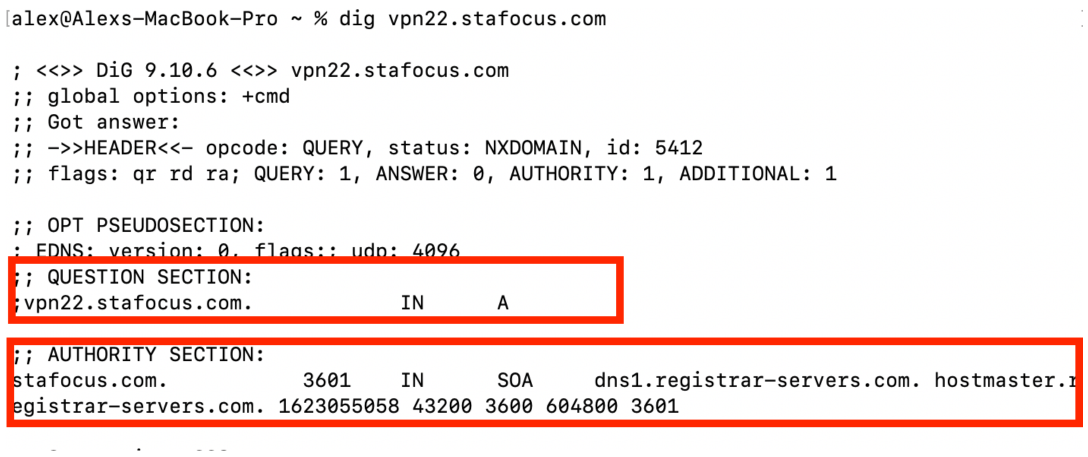
[47] SOA: The full name of SOA is "Start of Authority", which is a DNS record stores important information regarding a domain or zone, such as administrator's email address, that last update date of this domain [196].



```
[alex@Alexs-MacBook-Pro ~ % dig vpn.stafocus.com                                    ]

; <<>> DiG 9.10.6 <<>> vpn.stafocus.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 39548
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;vpn.stafocus.com.              IN      A

;; ANSWER SECTION:
vpn.stafocus.com.       1799    IN      CNAME   allways-cam.com.

;; AUTHORITY SECTION:
com.                    900     IN      SOA     a.gtld-servers.net. nstld.verisign-grs.c
om. 1626302367 1800 900 604800 86400
```

FIGURE 105: The result of command dig *vpn.stafocus.com* in the terminal.

However, if the "AUTHORITY SECTION" includes a key word "SOA", and the information given in the "ANSWER SECTION" does not have a detailed record, this means this subdomain does not exist (as it does not include a complete A record). Figure 106 below, shows the result of the command **dig vpn22.stafocus.com**. The subdomain *vpn22.stafocus.com* is without an A record and other further records, also the key word "SOA" occurs in the "AUTHORITY SECTION", thus we identify the queried subdomain *vpn22.stafocus.com* as not existing.



```
[alex@Alexs-MacBook-Pro ~ % dig vpn22.stafocus.com

; <<>> DiG 9.10.6 <<>> vpn22.stafocus.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 5412
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;vpn22.stafocus.com.            IN      A

;; AUTHORITY SECTION:
stafocus.com.           3601    IN      SOA     dns1.registrar-servers.com. hostmaster.r
egistrar-servers.com. 1623055058 43200 3600 604800 3601
```

FIGURE 106: The result of command dig *vpn22.stafocus.com* in terminal.

In order to test the efficiency of our approach, a small comparative experiment was conducted between our generated dictionary and the SubDomainsBrute's[48] dictionary. Moreover, the behaviour of frequently sending network requests to a target website (server) can be seen as a kind of DOS attack, thus we did not consider multiple levels of subdomain to launch the querying in order to decrease the amount of query traffic. For example, in *a.example.com*, ***a*** is the subdomain of *example.com*, which has one level. In *a.b.example.com*, *a.b* is the subdomain of *example.com*, which has two levels. In our experiment, we focused on the former case of one level only for simple analysis. We picked SubDomainsBrute's dictionary, noted as ***dict_sub***, and generated our dictionary with the same string length (15962) as ***dict_sub***, noted as ***dict_us***. Then we selected 12 websites for this comparative experiment, where five websites are famous university websites which would have a lot of traffic but because most phishing websites are purporting to be financial websites, we selected seven financial-related websites, such as the most famous online payment system website, the most famous online shopping website, and bank websites. The sites are *cam.ac.uk*, *st-andrews.ac.uk*, *ox.ac.uk*, *ucl.ac.uk*, *imperial.ac.uk*, *paypal.com*, *ebay.co.uk*, *santander.co.uk*, *bankofscotland.co.uk*, *barclays.co.uk*, *bankofamerica.com* and *hsbc.co.uk*. According to our findings, the generated dictionaries are successful in covering the shortage of non-dictionary word domains in the existing dictionary. The results are shown in Table 21 below:

(NB: For example, if we use tool's dictionary to query the subdomain for *example.com*, the queried results are *aaa* (which means the full URL regarding this subdomain is *aaa.example.com*, but we only store aaa, and ignore the domain part), *bbb*, *ccc*, *ddd*, *eee*. The number of the confirmed subdomains in dict_sub is 5. Next, if we implement our dictionary to query the subdomain for *example.com*, the queried results are *aaa*, *bbb*, *abc*, *xyz*. The number of confirmed subdomain in dict_us is therefore 4. Both results are then compared, and we find the subdomains queried using SubDomainsBrute's dictionary missed the result of *abc* and *xyz* that are queried using our dictionary. So, the number of missed subdomains in dict_sub is 2.)

[48] SubDomainBrute: This is an open source fast subdomain brute tool for pentesters. Available at: *https://github.com/lijiejie/subDomainsBrute*.

Domain	Queried Subdomains in dict_sub	Queried Subdomains in dict_us	Missed Subdomains in dict_sub
cam.ac.uk	33	31	11
st-andrews.ac.uk	110	102	21
ox.ac.uk	61	55	11
ucl.ac.uk	50	73	39
imperial.ac.uk	46	37	5
paypal.com	45	34	4
ebay.com	201	115	9
santander.co.uk	40	33	5
bankofscotland.co.uk	27	13	1
barclays.co.uk	54	40	3
bankofamerica.com	52	79	49
hsbc.co.uk	27	17	2

TABLE 21: The summary of a comparative experiment to generate a dictionary vs using the subdomainsbrute's dictionary.

The Evaluation of an Auto-Detection System

In our Auto-Detection system, the dictionary we used is from the subdomain enumeration tool SubDomainsBrute[49]. The subdomains are combined with targeted domains through traversing this dictionary. All subdomains that contain a potential takeover risk will be shown to the user after executing this system. However, each reported subdomain has two risky vectors of attack, one is the CNAME record; the other is the A record. For example, if the final CNAME record of *example.com* is sub.example.com IN *CNAME* abc.com, *abc.com* cannot be further resolved, which means the domain *abc.com* is unreachable. There are two important concerns here; The first reason is the domain of *abc.com* may be expired, and it

[49] SubDomainBrute: This is an open source fast subdomain brute tool for pentesters. Available at: *https://github.com/lijiejie/subDomainsBrute*.

has not been purchased by anyone else. This means that the DNS configuration in *abc.com* is empty, so that *abc.com* is unreachable. In this case, anyone who purchases this domain will compromise *sub.example.com*. The second reason is that the A record (IP address) of *abc.com* may be expired. In this case, anyone who purchases this IP address will compromise *sub.example.com*. Note that an IP address will be randomly assigned after purchase from a cloud vendor, so this IP address will be available and assigned to another purchaser if the previous purchaser's rights had expired. The assignment of an IP address is random, so this takeover risk exists if the new purchaser is assigned to a previous IP address, although the probability is rare, it was proven by Liu, Hao and Wang in [34]. To date there is no available interface or API on any Domain Name Vendor to allow a user to check whether a domain is available. Therefore, the user needs to go to a Domain Name Vendor website for verifying whether this domain is purchasable.

In our evaluation, we purchased two domains (*stafocus.com* and *ly1026.me*) to simulate and configure a vulnerable website. In the *stafocus.com* domain, the risky subdomains are identified to include *edu.stafocus.com*, *education.stafocus.com* and *vpn.stafocus.com*. In the domain of *ly1026.me*, the risky subdomain is identified to be *edu.ly1026.me* (For more DNS configuration details please see the above section 6.3.2). All of the final CNAME destination domains in this experiment are *allways-cam.com*, and hence we can confirm the corresponding (initial) subdomain has a takeover risk. The domain *allways-cam.com* on Name Vendor is available, as shown in Figure 107 below,

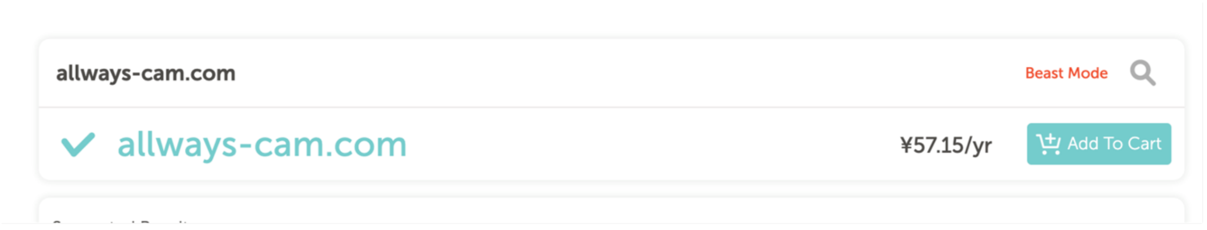


FIGURE 107: The searched result of the targeted domain *allways-cam.com* in Namecheap.

In this section, two scenarios on CNAME records (in DNS configuration) were considered, there are the one time CNAME record, and multiple times CNAME records. For example, in a one time CNAME record, the subdomain *edu.stafocus.com*, its CNAME record was edu.stafocus.com **IN** ***CNAME*** allways-cam.com. In the multiple times CNAME records scenario, using the subdomain *edu.ly1026.me*, its CNAME record was edu.ly1026.me **IN** ***CNAME*** edu.stafocus.com **IN** ***CNAME*** allways-cam.com. In the real word, the CNAME record can be claimed multiple times for any subdomain in DNS configuration without the restriction of levels, so both scenarios may exist, and we successfully detected these scenarios in our system.

6.5. Discussion

A successful subdomain takeover attack has a higher threat level as a controllable subdomain owns the same SSL certificate with its parent website, yet it does not require an advanced technical skill to exploit this weakness. In particular, to exploit controllable subdomains for a phishing attack. For example, if a legitimate website, *www.example.com*, has a subdomain, *abc.example.com* under a subdomain takeover attack. The SSL certificate in this subdomain may be as same as the SSL certificate in a legitimate website if the legitimate website configures **.example.com* to release its certificate. Under this case, most phishing detection approaches may not be useful if the features are selected from the SSL certificate, web content etc. For example, in our OCR approach, we identify the phishing website by comparing the "**issued by**" and "**issued to**" from the SSL certificate between the accessed website and legitimate website. However, if the SSL certificates are the same, the attributes "**issued by**" and "**issued to**" show the same value, which results in our approach failing.

In the detection of a Subdomain Takeover Attack, the current approach to detect this potential vulnerability is divided into three experimental steps. Firstly, querying the subdomains of a targeted website. Next, resolving the related records (such as CNAME, NS and MX). Finally, detecting the usability of the destination domain. However, there is currently not an automatic approach or tool (with a higher accuracy or a faster speed) to execute the above steps to detect this potential vulnerability.

In our research, two techniques have been presented as potential solutions. One is a query approach based on machine learning for querying existing subdomains and the second is an autodetection system to identify the potentially risky subdomains. According to our experimental findings, our solutions show an ideal result. Although both techniques have limitations, they are still viable for real world applications, and the corresponding limitations can be improved upon.

In our querying subdomains approach, the generated dictionary (predicted subdomains) can be used to effectively query the website subdomains. From the data of the short comparative experiment above, we find that although the existing subdomains in the prediction approach is fewer in number than the existing dictionary, the predicted subdomains successfully cover the shortage of non-dictionary word domains in the existing dictionary. Compared to other tools that build their dictionary using the Brute Force approach, the subdomain generation algorithm that is based on Char-RNN does not depend on the limitation of word composition in the existing dictionary. It supports a comprehensive index list, which not only includes part of the content of existing dictionaries, but also provides a prediction method to generate the extra content that does not currently exist in the dictionary by learning the existing subdomain names. This means that a generated word (string) that is a strange composition of various letters (an unknown word) from this model is not valueless, it remedies the limitations of the present dictionary, and this newly generated word may actually exist as a subdomain. For example, a common subdomain would be "*shop*" but "*xshop*" would not be common, but still possible.

Compared to the current brute force approach, we have the same limitation, which is where the result is subject to the integrity of the dictionary. However, we have an obvious improvement as the current dictionary from the brute force mechanism misses the long non-dictionary word subdomains, and these subdomains may exist in some domains. Our approach successfully overcomes this shortage, although the querying of common subdomains is not as good as the existing dictionary in most situations, see table 21.

In our auto-detection system, we presented an automatic workflow to discover the subdomains that are under a potential subdomain takeover attack. Compared to other subdomain enumeration tools, most tools merge various approaches to provide a full discovery when gathering existing subdomains, but it is hard to identify which subdomains have potential risks. In our automated detection system, we provide a comprehensive procedure to find the subdomains that may contain a potential takeover risk, although the final step in confirming the availability of the target domain requires us to search corresponding domain name buying websites by ourselves. Initially, the time consumption for detecting the risky subdomain is large as we adopted serial computing in our system. The system traverses the dictionary from the beginning, one by one, until the end. Each string in the dictionary requires to execute the command **dig** and filter the relevant information we needed for further detection. Thus, the time consumption is increased with the growth of the dictionary length. For example, in our experiment, we adopted two different files as the dictionary for detecting risky subdomains. One file is a small size (4 KB); the other file is a larger (512KB). The detection process had a significant time increase when we implemented the experiment with the large size file. We used the command time to record the results, the detail of this experiment is summarised in Table 22 below.

File	Size	String[50]	Cost Time
File 1 (small size)	4 KB	25	0 m 2.6 s
File 2 (large size)	512 KB	76119	6 m 16 s

TABLE 22: The summary of file details and time cost in the experiment.

Moreover, the content of the dictionary determines the accuracy of the result, that is the success rate of detecting risky subdomains. For example, in our experiment, *vpn.stafocus.com* is a risky subdomain as the destination CNAME address (*allways-cam.com*) is available to be purchased by anyone from Domain Name Vendor. However, if the dictionary did not consist of the string "*vpn*", this risky subdomain (*vpn.stafocus.com)* would not be detected, and the final result that would be returned to the user would not include this risky subdomain. Therefore, in a future system version, we must first consider adopting parallel computing to optimize the time consumption issue during the detection process. Also, we intend to improve the dictionary integrity issue by using the Char-RNN model.

[50] In this table, String means how many strings in the dictionary file.

Compared to other research about related-domain attacks, our approach has several advantages. In [34], Liu, Hao and Wang studied the threat posed by 'dangling' DNS records (which are called Dare). Their research focused on four threats, Dare A record, Dare CNAME record, Dare NS record and Dare MX record. They explained the specific risk and causal reasons for each threat in detail, although the probability of occurrence of a risky A record, NS record and MX record is rare in dangling DNS records. In their detection approach, especially, for identifying risky CNAME in Dare, they considered this threat incorporating with MX record, thus they required the parameter "*domain_expired*" to confirm the results. In order to check whether a domain has expired, they used WHOIS for querying at first, then crosschecked with the Internet Domain Registrars GoDaddy as WHOIS is not always reliable. The algorithm is:

```
If type ∈["CN", "MX"] then
    If domain_expired(data) then
        DARES
```

However, for their algorithm, the verification process is slow and redundant, if the subdomain has been configurated with multiple times CNAME records, which means each domain after the CNAME record will be checked by using both WHOIS and GoDaddy. It is unnecessary as this can be verified by checking the usability of the final destination domain from the dig record, which we explained this case in above section. Thus our approach is more efficient. Also, in their experiment, they queried the subdomains using a dictionary, the integrity of the enumerated subdomains is subject to the content of dictionary. But existing dictionaries have limitations, as we have shown; the subdomain may be a non-dictionary word, and may consist of seemingly random letters. Our predicted dictionary covers this shortage.

7. Discussion

7.1. Thesis Objective

The research of phishing attacks is an ongoing topic and deserves and requires more effort. According to the reviewed literature in Chapter 3, phishing is explored from multiple ways, many of the doctoral studies come from universities researching a variety of phishing related solutions, such as presenting a survey or taxonomy of phishing, focussing on exploring the causes and motivation behind the attacks, and analyzing the phishing attacks life cycle. Different to these researchers, in this thesis, we presented the phishing mitigation approaches from a comprehensive perspective, we focused on web phishing attacks and explored its mitigations. Compared to other relevant web phishing detection literature, we considered more situations in web phishing attacks, and conducted a detailed categorisation. We categorised web phishing attacks into three different types, general phishing attacks, advanced phishing attacks and subdomain takeover attacks according to both attack sophistication and features. Because web phishing attacks are complicated, features that are used to identify phishing websites are different due to the different attack types (see Table 2 in Chapter 1). From the general phishing attacks, through advanced phishing attacks to the deeper subdomain takeover attacks, both attack sophistication and concealment degree are increased, which means a common mitigation may not be suitable for all web phishing attacks, and most current anti-phishing solutions do not consider this situation.

In general phishing attacks, the phishing website may have the identical content to the related legitimate website, but the URL address and SSL certificate are different in comparison to the legitimate website. In advanced phishing attacks, either DNS hijacking attacks or ISP hijacking attacks, phishers launch the phishing attacks by exploiting some technical means, which results in both the URL address and the website content potentially being the same as the related legitimate website. In subdomain takeover attacks, phishers exploit the network configuration flaws to launch phishing attacks, so the phishing website has the same SSL certificate to its parent legitimate website. Most researchers present machine learning based approaches with various feature algorithms to classify phishing websites, such as extracting web-based features, that is, HTML-based features. However, the predicted results may not be correct as the extracted features from the phishing website are most likely to be legitimate and reasonable, such as using “URL dot count”, “URL length”. These HTML based features used to train data are the same as the related legitimate website if the victims are under an advanced phishing attack or a subdomain takeover attack. Therefore, we believe that it is necessary to present various solutions considering the different attack types in order to enable detection of more situations. We are the first authors to categorise web phishing attacks into three different cases according to attack sophistication and features.

Researchers have focused on two strategies to mitigate phishing attacks; improving the performance of phishing detection technology or developing human education, as phishing attacks exploit the weaknesses in human characteristics [91] [92]. We focused on improving phishing detection technical performance, resulting in the technical solution becoming faster and more accurate. We also required solutions to be acceptable for multiple platforms (desktop and mobile).

However, for human education, this is important to ensure users understand how to identify phishing attacks, and the quality of the education can affect the successful of preventing phishing. Some forms of education like gamification can make the learning process more effective. Nevertheless, an automated technical solution will likely still be more effective as it is more consistent and not subject to human error. However, both the technical solution and human education are indispensable, in particular, for an organization, staff require a higher security awareness than their individual users. Otherwise, both individual and enterprise data may be exposed to phishers. Moreover, with the constant evolution of phishing attacks, it is likely that a novel attack may be designed and be applied on future, so the existing technical solutions may not be available to detect all newly developed attacks. Therefore, human education is still a necessary factor to improve security against phishing attacks, and a worthy research topic.

List-based and content-based are the two main technical approaches that are explored and deployed to detect and classify phishing attacks in general, although these have advantages and disadvantages. Various machine learning approaches are derived from content-based approaches, with many feature-based algorithms developed to automatically detect phishing websites, but these approaches are prone to false positives [49] [99]. Comparing these two approaches, the list-based approach requires a large human effort to classify the phishing links and update the list library. In some list-based anti-phishing solutions, in order to increase the efficiency of a human identifying a phishing website, researchers categorize webpages according to their content similarities as a large number of current phishing websites are created with tools, such as toolkits. Also, the detection accuracy is subject to the integrity of the list library, and it is difficult to detect the newly (zero day) phishing website. In the content-based approach, malicious websites can be classified through analysing the website features, and this approach makes the detection more dynamic and improves the efficiency of detection. Many feature-based algorithms have been developed to automatically detect the accessed websites. The accessed websites can be classified if they are a malicious page, once the users trigger the suspicious link on their browser, although the analysis process requires time. In particular, in a machine learning based approach, the time consumption is increased with the number of, or sophistication of, extracted features. Therefore, we focused on presenting a straightforward content-based approach to judge phishing attacks without using a machine learning approach.

7.2. Thesis Hypotheses

From Chapter 4 to Chapter 6, we presented different solutions to detect corresponding phishing attacks, and the challenges faced were discussed at the end of each chapter. Also, the comparison between our solution and other solutions have been conducted to show how beneficial our solutions are.

In the detection of general phishing attacks, we designed a straightforward approach with four steps implementing OCR to recognize phishing websites. The procedure is simple and does not use a machine learning approach to derive a result. We also demonstrated how to deploy this approach on mobile platforms. Although phishing attacks that take place on mobile devices is a rising trend, most researchers have not focussed on this research, or not mentioned how to implement their solution on mobile devices when they presented their novel approach to detecting phishing attacks. Our research is therefore valuable, as mobile devices have less security protection mechanisms and limited resources.

Compared to existing phishing mitigation approaches, our OCR approach has several advantages:

1) *Dynamic method and without machine learning*:

A list-based approach is a static method, as it requires a large human effort to maintain the blacklist. Also, zero-day attacks can't be detected. Our OCR solution overcomes these limitations as the system launches an active detection to detect the target website without requiring any human resources.

2) *Multiple detections*:

Our OCR approach provides multiple detection, not only mitigating general phishing attacks, but also preventing DNS hijacking attacks. Most recent anti-phishing solutions cannot be implemented to detect the phishing website if it is under a DNS attack, in particular, features are selected from the URL content in some machine learning approaches.

3) *Multiple Implementation Platforms*:

Our approach is easy to deploy in that it does not use machine learning, and is available on both PC platforms and mobile platforms. Moreover, our proposed prototype on mobile platforms would not result in extra resource consumption as most of the computation process is executed on a server. Most recent anti-phishing solutions have not considered this situation, although the phishing on mobile platforms is an increasing trend.

Therefore, the feasibility of the first hypothesis "***An OCR approach can be implemented efficiently to detect phishing attacks without using machine learning to predict and category the results***" is proven.

Compared to other OCR and visual-based approaches, our OCR solution has a better performance. The deployment is easy, and procedure is faster, as our solution provides a straightforward detection approach without using machine learning; Moreover, others OCR approach identify the phishing website by comparing the domain name between the accessed website and legitimate website, thus, their solution would fail if the phishing website is under a DNS attack. But our solution overcomes this limitation and supports a multiple detection.

In the detection of advanced phishing attacks, we first prove how our OCR approach detects the DNS hijacking attack successfully and we also present a prototype to illustrate how to deploy this approach on a mobile platform. Different to existing DNS diagnostic tools, current DNS detection tools are used to diagnose whether a user's DNS server is healthy, rather than detecting the phishing websites. Also, most current phishing mitigations do not consider this case, and most machine learning based approaches that select the features from URLs and webpages may not be a good option as the accessed URL content is the same as the legitimate website. This results in the features gathered from the accessed website having a high similarity with the corresponding legitimate website, which affects the machine learning prediction results.

So, the feasibility of the second hypothesis "***A phishing website that is under a DNS hijacking attack can be detected by implementing OCR and examining the associated SSL certificate***" is proven.

Our solution provides a dynamic approach to detect the phishing website that is under a DNS hijacking attack by identifying and comparing the SSL certificate information between the accessed website and the related legitimate website, and does not require a user to launch any active diagnosis request. The accuracy of our detected result is again subject to the logo extraction and content recognition phases. Both false positive and false negative results may exist if the recognition result is inaccurate. Thus, the OCR approach that is deployed to detect phishing websites in DNS hijacking attacks faces the same advantages and disadvantages as the approach used to detect general phishing attacks.

Furthermore, we determine ISP hijacking attacks through comparing the construction of a site's DOM tree. Compared to other solutions, we provide a comprehensive detection, even the hijacked node (point of deviation) can be indexed, and this is what other research lacked. We simulated a hijacked environment by adding nodes, removing nodes and modifying nodes with different elements in order to cover simulate real world situations, and our system successfully detected the differences in all cases. Moreover, we presented two reasonable deployment schemes that are suitable for different clients with current network development, there are Prototype to Client (*PtoC*) and Prototype to Business (*PtoB*). In the *PtoC* model, the security vendor focuses on the protection of the end-side user. In the *PtoB* model, the main beneficiary is shifted to the enterprise who would purchase this kind of service from the security vendor.

So, the feasibility of the third hypothesis "***ISP hijacking attacks can be identified by observing the consistency and differentiation of DOM trees***" is proven.

Other DOM tree based approach solutions identify the phishing website by detecting the vector distribution of the associated DOM tree, as most phishing websites are designed by toolkits, which results in the distribution of DOM tree being different to the original (legitimate) website. However, when a website is under an ISP hijacking attack, phishers may not change too much information on the webpage, they may insert or change webpage content somewhere to convince and redirect the victims into another phishing websites, which means the vector distribution of DOM tree would not be significantly changed. Thus, these DOM tree based approaches are an ideal anti-phishing solution that detects the phishing websites which are designed by toolkits, but these solutions are not suitable to detect a phishing website if it is under an ISP hijacking attack.

In the detection of subdomain takeover attacks, we first present a subdomain enumeration approach using the Char-RNN algorithm, the generated dictionary (predicted subdomains) can be used to effective query the website subdomains. Compared to existing subdomain enumeration tools and approaches, the subdomain generation algorithm that is based on Char-RNN does not depend on the limitation of word composition in the existing dictionary. It supports a comprehensive index list, which not only includes part of the content of existing dictionaries, but also provides a prediction method to generate the extra content that are not in the dictionary by learning the existing subdomain names.

So, the feasibility of the fourth hypothesis “***Subdomains can be predicted using machine learning to cover the shortage of non-dictionary word domains in the existing dictionary***” is proven.

We designed an automated subdomain takeover detection system subsequently wherein the workflow is simple and easily deployed. It not only solves the problem of detecting risky subdomains manually, but it is also suitable for various users. Currently, there are only a few academic papers focussed on the topic of subdomain takeover attacks, most of them focus attention on related domain attacks, and proved the specific threats. Although they implemented various approach to detect the risky subdomains, all of them required incorporation with other systems or manual online application and no one explored how to detect this kind of threat from an automated technical approach. Hence, we addressed this research gap. Our approach is more efficient by checking the usability of the final destination domain, in particular, the verification process is simple and not redundant if the subdomain has been configurated with multiple CNAME records, and this is what other research lacked.

So, the feasibility of the last hypothesis “***The risky subdomain can be discovered by an automatic approach without any human efforts***” is proven.

Compared to other research about related-domain attacks, our approach is an automagical approach without using much human involvement, which is more efficient. For other approaches, the verification process is slow and redundant, if the subdomain has been configurated with multiple times CNAME records, which means each domain after the CNAME record will be checked by using both WHOIS and GoDaddy. This is necessary and can be verified by checking the availability of the final destination domain from the dig record. Also, in their experiment, the authors queried the subdomains using a dictionary, and the integrity of the enumerated subdomains is subject to the content of dictionary. However, the existing dictionaries have limitations; the subdomain may be a non-dictionary word, and may consist of seemingly random letters. Our predicted dictionary covers this shortage.

7.3. Limitations

After evaluating our solutions and comparing with others, we found that there are still some challenges that need to be overcome in the future. For example, the efficiency issue is an important factor that requires to be improved. A fast response result is an essential factor for a solution, especially, when it comes to security issues. A system needs to identify and isolate the risks in real time once the threat happens. In our technical solutions, a malicious phishing website is classified automatically, and the result will be immediately returned to the users after the detection, without any human intervention. But the detection procedure takes time, for example, in our OCR approach, the consumed time was up 20 seconds in some cases, which will result in a decreased user experience. So, we need to improve the efficiency of our detection procedure, such as using parallel computing, and so this needs to be considered in the future.

Phishing attacks are a broad problem, they are a constant challenge due to their continual evolution. In the 2021 RSA Conference, Rivner and Englund [194] presented a novel threat, named a Deep Social Engineering Attack. The phisher here exploits the human psychological weakness to lure the victims to a phishing fraud. Different to a general social engineering attack, this novel fraud convinces victims with more realistic contents or stories, and it is inspired from a Sherlock Holmes case, “The adventure of the red headed league”. Victims do not realize they are being scammed, even after they transferred money to the phishers, and all behaviours are voluntary for these victims. During Rivner and Englund’s presentation, they proposed a machine learning based approach to identify this kind of fraud, as the features are selected from human behaviour, such as click count, voice changes, and the time of page view etc. Although they have not deployed large experiments to evaluate this approach, as a deep social engineering attack is hard to detect from a technical way only, we have to consider human behaviour as a factor and this is what does our current solutions lack. In fact, with the constant evolution of phishing attacks, deep social engineering is another attack trend, and hence the human factor needs to be considered in future research.

Our three presented automatic solutions that are used to mitigate and prevent general phishing attacks, advanced phishing attacks and subdomain takeover attacks separately are complementary to current systems and add in an extra defence. A range of attack scenarios, such as zero-days attacks in general phishing attacks, multiple changes in advanced phishing attacks and multiple CNAME configuration in subdomain takeover attacks, were considered, the detection results were successful and satisfactory, we have made the phishers job harder to convince victims. However, testing against unknown threats, such as newly exploited phishing attacks, was not evaluated, as it was not feasible in the timeframe. Using an independent outsider for tests would be a future work, especially prior to any industrial development. Moreover, the security research is a battle between a protected system and user experience, in particular, the user experience is often decreased with the increase of system security. In this thesis, we evaluated the proposed solutions, but we have not examined the user experience, and although this is out of scope of this thesis, we are going to consider this examination in the future, as the proposed scheme serves the users, thus, a scheme is not acceptable if it has a lower user experience.

8. Conclusion and Future Works

8.1. Conclusion

The modern Internet has become an indispensable element in our daily life, providing significant resources to people whether for play, work or education. Phishing attacks have accounted for a large proportion of all malicious attacks on the Internet as they are always cheap to produce and easy to deploy, although many users are sensible about this kind of attack. Phishing attacks are one of the major cyber threats, either to an individual user or organization. For the individual, sensitive credentials are always of interest to phishers, in particular, due to the development of E-commerce. For an enterprise, a successful phishing attack, such as a subdomain takeover attack, may affect their organization's reputation as well as cause financial loss.

Currently, most vendors have been using different approaches to prevent phishing. However, these solutions cannot keep up with the constant updating of phishing websites. In this thesis, we reviewed related literature about phishing attacks and preventions. Firstly, we re-classified phishing attacks into three different categories according to attack sophistication and features, from the shallower to the deeper. These are the General Phishing Attack, Advanced Phishing Attack and Subdomain Takeover Attacks. Secondly, we presented corresponding solutions to automatically mitigate and detect these attacks.

For general phishing attacks, we presented a novel approach to identify phishing websites by using an OCR technique. Unlike previous research, our approach overcomes the limitations of current methods and research solutions, not only providing a dynamic detection method, but also avoids feature limited issues, such as where the selected features are incomplete or incorrect in machine learning. Even if the phishing server has been compromised, it can also be identified in our system. Although this technique has a few limitations that need to be improved, it enables a high detection accuracy rate and the evaluation results are promising.

Moreover, in order to improve phishing prevention on mobile platforms, we also presented an effective prototype to implement a phishing attack solution on mobile platforms. We highly recommend that browser vendors insert a function to examine the user traffic before forwarding to the target site. This traffic will then be redirected to the remote browser server for further analysis. The browser can then respond with different webpages according to the result from the server. In our prototype, the mobile device resources would not be used up as most of the computation is executed on a server, and consequently the consumption is estimated to be similar to normal processing.

For advanced phishing attacks, the attacks are classified into two aspects; DNS hijacking attacks and ISP hijacking attacks, and we presented different mitigations to each aspect for improving user security. For DNS hijacking attacks, the accessed URL will be same as the legitimate website address. Current solutions cannot keep pace with the constant updating of phishing websites. Hence we detected and identified DNS hijacking attacks by using our OCR technique and we found our solution not only overcomes the limitation on existing solutions, but also provides a high detection accuracy rate.

There is a general problem in detecting ISP hijacking attacks. In particular, all the traffic appears to be from a trusted organization. We have considered using the construction of a site's DOM tree to determine the security of a user's network environment by verification of DOM tree consistency and verifiable differences, i.e. to check whether the accessed ISP is safe or not. The experimental results demonstrate a high detection accuracy rate, and the detection process is efficient. We also presented two deployment models regarding this prototype according to the different requirements of Prototype to Client (*PtoC*) and Prototype to Business (*PtoB*). In the *PtoC* model, the security vendor focuses on the protection of the end-side user. In the *PtoB* model, the main beneficiary is shifted to the enterprise who would purchase this kind of service from the security vendor.

For subdomain takeover attacks, a successful subdomain takeover attack has a higher threat level as a controllable subdomain owns the same SSL certificate with its parent website, though it does not require an advanced technical skill to exploit this weakness. In this thesis, two techniques have been presented as potential solutions. One is a query approach based on machine learning for querying existing subdomains and the second is an autodetection system to identify the potentially risky subdomains. Although both techniques have limitations, they are still viable for real world applications, and the corresponding limitations can be improved upon.

The presented subdomain querying approach is based on a Char-RNN neural net model. The final generated result includes a comprehensive subdomain name content list, which not only covers the content of an existing dictionary, but also provides a prediction method to generate more extraneous content. In a future version, in order to improve the accuracy of the associated dictionary, the classification of industry naming conventions regarding potential targeted domains may need to be considered, because there are a variety of subdomain names used by different industries.

In an auto-detection system, parallel or cloud-based computing could be implemented to reduce time consumption in the entire procedure. Also, an accurate dictionary that is generated from the Char-RNN model could be used to replace existing dictionaries, so that the system could provide a more sophisticated and accurate detection.

8.2. Future Works

In this thesis, we presented three automatic solutions to mitigate and prevent general phishing attacks, advanced phishing attacks and subdomain takeover attacks separately. Although each solution is applicable and effective, we still face few challenges. In future research, we would first focus on improving and overcoming these challenges. In the solution of mitigating general phishing attacks, we need to consider an alternative free way to solve charging for the required OCR API. We will design and develop an own OCR API for recognizing logo content, as the current OCR APIs are charged after a limited free usage. Moreover, the system efficiency needs to be improved, as time consumption will increase if the size of the extracted image from the target website is too large in our current version.

In the future development of the solution for preventing advanced phishing attacks, we are going to focus on how to verify the DOM tree when the image loaded on the webpage is different after refresh the webpage, and therefore we need to focus on the *nodename* in each image tag. In further future work, the image attribute will be considered during the verification process, and after a categorisation process we will discover what attributes result in potential risks.

In the search for discovering subdomain takeover issues, we are going to implement a Char-RNN model to replace the existing dictionaries for improving the complexity and integrity of querying subdomains. We will further consider implementing parallel computing to improve the efficiency of the detection process.

A phishing attack is an extensive problem; this threat has existed since the establishment of the Internet, and it is still a large proportion of all cyber-attacks. The development of a complete anti-phishing solution is a major challenge, as many phishing attacks exploit weaknesses in human characteristics, rather than appropriating network flaws. A technical solution, not including any human factors, is needed to prevent phishing attacks, with high accuracy and efficiency. However, with the constant evolution of phishing attacks, the phishers are continually improving attack strategies and bypassing the detection of current anti-phishing solutions. Moreover, in some situations, the human factor is a necessary to the mitigating solution. For example, deep social engineering is a trend which uses a more convincing narrative to lure victims. The victims do not realize that they are under attack, even after they have transferred money or information to the phishers. In this kind of attack, it is hard to detect the attack in a technical way without considering human factors, as the phishers may contact the victim and narrate a plausible story via a voice call. The phishers control the victim who transfers money or information via a legitimate website, resulting in all victim manipulations being legitimate. In the RSA 2021 conference, Rivner and Englund demonstrated the existence of such a case, and presented a solution. In their solution, the human behaviour, such as click count, voice changes, and the time of page view etc, were considered as classifiers of suspicious behaviour and normal behaviour and were used to identify if the victim is under an attack. This system designs and develops an anti-phishing solution involving human behaviour, to detect suspicious behaviour in order to identify the fraud. Therefore, we would consider using this work as the basis of a combination approach of human factors and automated detection through gathering human and technical data to enable machine learning to distinguish malicious activities for preventing phishing attacks.

Along with enabling technical solutions, human education and intervention are important factors in preventing phishing attacks. A good anti-phishing education will improve users' security knowledge and awareness of phishing defence. The victims of phishing attacks are users and therefore a successful phishing attack requires users to trigger malicious emails or websites. The probability of triggering a phishing attack is reduced if the user's security awareness is increased. Gamification is one way to make the learning process more effective, but it would always be guiding users in arrears of current attacks. It is likely that a novel attack would be designed and applied in future in line with the constant evolution of phishing attacks. Therefore, how to design and develop a suitable gamification anti-phishing education platform that explains techniques with appropriate feedback and is up to date with the constant evolution of phishing attacks is a worthy future research topic.

Many cyber-attacks can be originated from phishing, such as botnet and the APT (Advanced Persistent Threat). Phishers exploit phishing email to spread their malicious software; a malicious payload will be installed in the victims' devices once they download and access this software. Both botnet and APT are high-level threats for either organizations or countries. Under botnets, the victim's device is controlled by a bot master where not only the device may be exposed to the bot master, but the device itself could be controlled to execute suspicious manipulations without the users' awareness, such as with DDOS attacks. Under APT, phishers could control the victim's device through a malicious payload; either the victim‘s data or the victim's access permission to sensitive data inside of an organization or government could be exploited. Thus, the detection of phishing email is another research topic in the future, either to deploy a straightforward detection solution or to implement a prediction machine learning based approach. Both methods need further research. Moreover, the detection of phishing email would be divided into two paths; the first would focus on the content detection (not including the attachment), using the email details, such as sender information, email content etc, to classify the phishing email. Secondly, we would focus on the attachment. The formats of attachments are easy to identify, but the detection of any malicious payload inside of an attachment is a major ongoing challenge. For example, the malicious payload could be inserted into the "Macro" in a .doc file. Moreover, an APT always exploits the Zero-day attack, which requires automatic detection involving a comprehensive and complex analysis.

In this thesis, the proposed solutions can be deployed to automatically counter web phishing attacks, in particular, general phishing attacks, advanced phishing attacks and subdomain takeover attacks. There are still more challenges to be overcome from integrating multiple phishing approaches, and this needs further exploration through future research.

9. References

[1] Y. Wang, Y. Liu, T. Wu, and I. Duncan, “ISP Hijacking protection via DOM tree comparison”. *Pending for IEEE Cyber Security Conference 2023*.

[2] Y. Wang, and I. Duncan, “A novel method to prevent phishing by using OCR technology”. *In Proceedings of International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2019*, pp. 1-5, Jun 2019. DOI: 10.1109/CyberSecPODS.2019.8885101.

[3] M. Powell, “11 Eye Opening Cyber Security Statistics for 2019 - CPO Magazine”. Jun 2019. [Online]. Available: https://www.cpomagazine.com/tech/11-eye-opening-cyber-security-statistics-for-2019/. [Accessed: 06-Mar-2020].

[4] R. Sobers, “134 Cybersecurity Statistics and Trends for 2021 | Varonis”. Mar 2021. [Online]. Available: https://www.varonis.com/blog/cybersecurity-statistics/. [Accessed: 16-Aug-2021].

[5] Phishlabs, “2019 PHISHING TRENDS AND INTELLIGENCE REPORT The Growing Social Engineering Threat”. [Online]. Available: https://info.phishlabs.com/hubfs/2019 PTI Report/2019 Phishing Trends and Intelligence Report.pdf [Accessed: 15-Jun-2020]

[6] GOV.UK, “Cyber Security Breaches Survey 2020 - GOV.UK”. [Online]. Available: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2020/cyber-security-breaches-survey-2020. [Accessed: 16-Aug-2021].

[7] P. Abbate, “2020 Internet Crime Report”. 2020. [Online] Available: https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf [Accessed: 10-Sep-2020].

[8] TESSIAN, “Understand the mistakes that compromise your company s cybersecurity”. 2020. [Online]. Available: https://www.tessian.com/research/the-psychology-of-human-error/ [Accessed: 15-Jun-2020]

[9] N. Kumaran and S. Lugani, “Protecting against cyber threats during COVID-19 and beyond | Google Cloud Blog”. Apr 2020. [Online]. Available: https://cloud.google.com/blog/products/identity-security/protecting-against-cyber-threats-during-covid-19-and-beyond. [Accessed: 16-Aug-2021].

[10] A.F. Al-Qahtani, S. Cresci, “The COVID-19 scamdemic: a survey of phishing attacks

and their countermeasures during COVID-19”. *In Journal of IET Information Security*, vol. 16, no. 15, pp. 324-345, Sep 2022. DOI: 10.1049/ise2.12073.

[11] K. Nirmal, B. Janet, and R. Kumar, “Phishing - The threat that still exists”. *In Proceedings of the International Conference on Computing and Communications Technologies, ICCCT 2015*, pp. 139–143, Feb 2015. DOI: 10.1109/ICCCT2.2015.7292734.

[12] M. Thaker, M. Parikh, P. Shetty, V. Neogi, and S. Jaswal, “Detecting Phishing Websites using Data Mining”. *In Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018*, pp. 1876–1879, Mar 2018. DOI: 10.1109/ICECA.2018.8474820.

[13] PhishProtection, “History of Phishing: How Phishing Attacks Evolved From Poorly Constructed Attempts To Highly Sophisticated Attacks | PhishProtection.com”. [Online]. Available: https://www.phishprotection.com/resources/history-of-phishing/. [Accessed: 16-Aug-2021].

[14] Phishing.org, “Phishing | History of Phishing”. [Online]. Available: https://www.phishing.org/history-of-phishing. [Accessed: 16-Aug-2021].

[15] K. Hanko, “Startling Phishing Statistics to Know in 2022”. [Online]. Available: https://clario.co/blog/phishing-statistics/. [Accessed: 04-Jun-2022].

[16] APWG, “Phishing Activity Trends Report”. Q1 2019. [Online]. Available: https://apwg.org/trendsreports/ [Accessed: 10-Jun-2021].

[17] P. Langlois, “Data Breach Investigations Report”. Jul 2020. [Online] Available: https://www.cisecurity.org/wp-content/uploads/2020/07/The-2020-Verizon-Data-Breach-Investigations-Report-DBIR.pdf [Accessed: 21-Jun-2020].

[18] C. Budd, “Are shortened URLs safe?”. May 2016. [Online]. Available: https://blog.trendmicro.com/are-shortened-urls-safe/. [Accessed: 16-Aug-2021].

[19] SecurityScorecard, “12 Types of Phishing Attacks and How to Identify… | SecurityScorecard”. May 2021. [Online]. Available: https://securityscorecard.com/blog/types-of-phishing-attacks-and-how-to-identify-them. [Accessed: 04-Jun-2022].

[20] PandaSecurity, “11 Types of Phishing + Real-Life Examples”. 2021. [Online]. Available: https://www.pandasecurity.com/en/mediacenter/tips/types-of-phishing/. [Accessed: 04-Jun-2022].

[21] Fortinet, “19 Types of Phishing Attacks with Examples | Fortinet”. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/types-of-phishing-attacks. [Accessed: 04-Jun-2022].

[22] D. Aleksandrova, “Exploiting common URL redirection methods to create effective phishing attacks - Help Net Security”. May 2021. [Online]. Available: https://www.helpnetsecurity.com/2021/05/10/exploiting-url-redirection-methods/. [Accessed: 04-Jun-2022].

[23] A. K. Jain and B. B. Gupta, “Phishing detection: Analysis of visual similarity based approaches”. *In Journal of Security and Communication Networks*, vol. 2017, Article ID. 5421046, pp. 1-20, Jan 2017. DOI: 10.1155/2017/5421046.

[24] Cisco Umbrella, “2021 Cyber security threat trends phishing crypto top the list”. 2021. [Online]. https://umbrella.cisco.com/info/2021-cyber-security-threat-trends-phishing-crypto-top-the-list [Accessed:15-Sep-2022].

[25] K. Weber, A. E. Schütz, T. Fertig, and N. H. Müller, “Exploiting the Human Factor: Social Engineering Attacks on Cryptocurrency Users”. *In Proceedings of International Conference on Human-Computer Interaction (HCII ‘2020): Learning and Collaboration Technologies. Human and Technology Ecosystems. Part of the Lecture Notes in Computer Science book series,* vol. 12206, pp. 650-668, Springer, Cham, Jul 2020. DOI: 10.1007/978-3-030-50506-6_45.

[26] Verizon, “2022 Data Breach Investigations Report”. 2022. [Online]. Available at: https://www.verizon.com/business/en-gb/resources/reports/dbir/ [Accessed: 15-Sep-2022].

[27] M. Rosenthal, “Phishing Statistics (Updated 2021) | 50+ Important Phishing Stats | Tessian”. 2021. [Online]. Available: https://www.tessian.com/blog/phishing-statistics-2020/. [Accessed: 03-Mar-2021].

[28] F. Castaño *et al.*, “Evaluation of state-of-art phishing detection strategies based on machine learning”. *In Proceedings of the 34th Cybersecurity Research. National Conference on Cybersecurity Research,* Jan 2021. DOI: 10.18239/jornadas_2021.34.06.

[29] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, “Machine learning based phishing detection from URLs.” *In Journal of Expert Systems with Applications*, vol. 117, pp. 345–357, Mar 2019. DOI: 10.1016/j.eswa.2018.09.029.

[30] Y. Wang, Y. Liu, T. Wu, and I. Duncan, “A Cost-Effective OCR Implementation to Prevent Phishing on Mobile Platforms.” *In Proceedings of International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2020*, pp. 1-8, Jun 2020. DOI: 10.1109/CyberSecurity49315.2020.9138873.

[31] NCSC, “Ongoing DNS hijacking and mitigation advice - NCSC.GOV.UK”. Jul 2019. [Online]. Available: https://www.ncsc.gov.uk/news/ongoing-dns-hijacking-and-mitigation-advice. [Accessed: 06-Mar-2020].

[32] NCSC, “DNS hijacking activity - NCSC.GOV.UK”. Jan 2019. [Online]. Available: https://www.ncsc.gov.uk/news/alert-dns-hijacking-activity. [Accessed: 06-Mar-2020].

[33] K. Borgolte, T. Fiebig, S. Hao, C. Kruegel, and G. Vigna, “Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates”. *In Proceedings of the Applied Networking Research Workshop (ANRW '18),* vol. 18, Association for Computing Machinery, New York, NY, USA, pp. 4, Jul 2018. DOI: 10.1145/3232755.3232859.

[34] D. Liu, S. Hao, and H. Wang, “All your DNS records point to us: Understanding the security threats of dangling DNS records”. *In Proceedings of the ACM Conference on Computer and Communications Security,* vol. 16, pp. 1414–1425, Oct 2016. DOI: 10.1145/2976749.2978387.

[35] M. Squarcina *et al.*, “Can I Take Your Subdomain? Exploring Related-Domain Attacks in the Modern Web”. *In Proceedings of the 30th USENIX Security Symposium, USENIX Security 2021*, pp. 2917-2934. Aug 2021.

[36] I. Modin, “hackerone-reports/· GitHub”. 2021. [Online]. Available: https://github.com/reddelexc/hackerone-reports/blob/master/tops_by_bug_type/TOPSUBDOMAINTAKEOVER.md. [Accessed: 03-Mar-2021].

[37] P. Hudak, “#325336 Subdomain takeover on svcgatewayus.starbucks.com”. 2018. [Online]. Available: https://hackerone.com/reports/325336. [Accessed: 03-Mar-2021].

[38] P. Hudak, “#570651 Subdomain takeover of mydailydev.starbucks.com”. 2019. [Online]. Available: https://hackerone.com/reports/570651. [Accessed: 03-Mar-2021].

[39] P. Zenker, “#665398 Subdomain takeover of datacafe-cert.starbucks.com”. 2019. [Online]. Available: https://hackerone.com/reports/665398. [Accessed: 03-Mar-

2021].

[40] Symantec, “Executive Summary 2018 Internet Security Threat Report ISTR Volume 23”. 2018. *Symantec Annual Report 2018*, vol. 23.

[41] T. Vidas, E. Owusu, S. Wang, C. Zeng, L. F. Cranor, and N. Christin, “QRishing: The Susceptibility of Smartphone Users to QR Code Phishing Attacks”. *In Proceedings of International Conference on Financial Cryptography and Data Security (FC 2013): Financial Cryptography and Data Security. Part of the Lecture Notes in Computer Science book series,* vol 7862, pp. 52–69, Springer, Berlin, 2013. DOI: 10.1007/978-3-642-41320-9_4.

[42] R. Benard, “Phishing News: QR Code Phishing Scheme”. 2016. [Online]. Available: https://www.vadesecure.com/en/blog/phishing-news-qr-code-phishing-scheme [Accessed: 10-Jan-2023].

[43] Wandera, “Mobile Phishing Report 2018”. 2018. [Online] Available: http://go.wandera.com/rs/988-EGM-040/images/mobile-phishing-report.pdf [Accessed: 20-Sep-2019].

[44] Proofpoint, “Analyst report | The Ponemon 2021 Cost of Phishing Study”. 2021. [Online]. Available at: https://www.proofpoint.com/us/resources/analyst-reports/ponemon-cost-of-phishing-study [Accessed: 15-Sep-2022].

[45] C. Rivett, “Report by Lloyd's says global cyber attack could cost $120bn”. May 2020. [Online]. Available: https://technologymagazine.com/cloud-and-cybersecurity/report-lloyds-says-global-cyber-attack-could-cost-dollar120bn [Accessed: 13-Jan-2023].

[46] IBM, “Cost of a data breach 2022”. 2022. [Online] Available at: https://www.ibm.com/reports/data-breach [Accessed: 15-Sep-2022]

[47] APWG, “Phishing Activity Trends Reports”. Q2 2022. [Online]. Available: https://apwg.org/trendsreports/. [Accessed: 15-Sep-2022].

[48] N. Mazher, I. Ashraf, and A. Altaf, “Which web browser work best for detecting phishing”. *In Proceedings of the 5th International Conference on Information and Communication Technologies (ICICT 2013)*, pp. 1-5, Dec 2013. DOI: 10.1109/ICICT.2013.6732784.

[49] G. Xiang, “Toward a Phish Free World: A Feature-type-aware Cascaded Learning Framework for Phish Detection”. *Language Technologies Institute, School of*

Computer Science, Carnegie Mellon University, PhD Thesis, 2013.

[50] Psafe, “Internet Security: Is Your Smartphone Safer Than Your PC?”. 2016. [Online]. Available: https://www.psafe.com/en/blog/internet-security-smartphone-safer-pc/. [Accessed: 22-Sep-2019]

[51] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “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*, vol. 09, Association for Computing Machinery, New York, USA, pp. 1245–1254, Jun 2009. DOI: 10.1145/1557019.1557153.

[52] Y. Wang, Z. Li, T. Wu, I. Duncan, and Q. Lyu, “An Empirical Study: Automated Subdomain Takeover Threat Detection”. *In Proceedings of International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA 2021)*, pp. 1–10, Jun 2021. DOI: 10.1109/CyberSA52016.2021.9478220.

[53] TESSIAN, “What is a Malicious Payload? | Examples of Malicious Payloads | Tessian”. Jan 2021. [Online]. Available: https://www.tessian.com/blog/what-is-a-malicious-payload/. [Accessed: 16-Aug-2021].

[54] Techslang, “What is a Malicious Payload? — Definition by Techslang”. [Online]. Available: https://www.techslang.com/definition/what-is-a-malicious-payload/. [Accessed: 16-Aug-2021].

[55] Wikipedia, “General Data Protection Regulation - Wikipedia”. 2021. [Online]. Available: https://en.wikipedia.org/wiki/General_Data_Protection_Regulation. [Accessed: 16-Aug-2021].

[56] Intersoft Consulting, “General Data Protection Regulation (GDPR) – Official Legal Text”. [Online]. Available: https://gdpr-info.eu/. [Accessed: 16-Aug-2021].

[57] Wikipedia, “Hypertext Transfer Protocol - Wikipedia”. 2021. [Online]. Available: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. [Accessed: 16-Aug-2021].

[58] MDN Web Docs, “An overview of HTTP - HTTP | MDN”. 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview. [Accessed: 16-Aug-2021].

[59] Wikipedia, “HTTPS - Wikipedia”. 2021. [Online]. Available: https://en.wikipedia.org/wiki/HTTPS. [Accessed: 16-Aug-2021].

[60] Cloudflare, “What is HTTPS? | Cloudflare”. [Online]. Available: https://www.cloudflare.com/en-gb/learning/ssl/what-is-https/. [Accessed: 16-Aug-2021].

[61] J. Scarpati and J. Burke, “What is a URL (Uniform Resource Locator)?”. 2019. [Online]. Available: https://searchnetworking.techtarget.com/definition/URL. [Accessed: 16-Aug-2021].

[62] MDN Web Docs, “What is a URL? - Learn web development | MDN.” [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL. [Accessed: 16-Aug-2021]

[63] Wikipedia, “URL shortening - Wikipedia”. 2021. [Online]. Available: https://en.wikipedia.org/wiki/URL_shortening. [Accessed: 16-Aug-2021].

[64] Wikipedia, “Man-in-the-middle attack - Wikipedia”. 2021. [Online]. Available: https://en.wikipedia.org/wiki/Man-in-the-middle_attack. [Accessed: 16-Aug-2021].

[65] D. Swinhoe, “What is a man-in-the-middle attack? How MitM attacks work and how to prevent them | CSO Online”. Feb 2019. [Online]. Available: https://www.csoonline.com/article/3340117/what-is-a-man-in-the-middle-attack-how-mitm-attacks-work-and-how-to-prevent-them.html. [Accessed: 16-Aug-2021].

[66] Wall Street Consultancy, “A Complete Guide to Man in The Middle Attack (MitM)”. 2021. [Online]. Available: https://wallstreetinv.com/cyber-security/man-in-the-middle-attack-mitm/. [Accessed: 16-Aug-2021].

[67] W3School, “JavaScript HTML DOM”. [Online]. Available: https://www.w3schools.com/js/js_htmldom.asp. [Accessed: 06-Mar-2020].

[68] DNSimple, “What’s an A Record? - DNSimple Help”. [Online]. Available: https://support.dnsimple.com/articles/a-record/. [Accessed: 16-Aug-2021].

[69] Cloudflare, “DNS A record | Cloudflare”. [Online]. Available: https://www.cloudflare.com/en-gb/learning/dns/dns-records/dns-a-record/. [Accessed: 16-Aug-2021].

[70] NS1, “CNAME Record - How it Works, Alternatives & Advanced Use Cases.” [Online]. Available: https://ns1.com/resources/cname. [Accessed: 16-Aug-2021].

[71] DNSimple, “What’s a CNAME record? - DNSimple Help”. [Online]. Available: https://support.dnsimple.com/articles/cname-record/. [Accessed: 16-Aug-2021].

[72] DNSimple, “Differences Between A and CNAME records - DNSimple Help”. [Online].

Available: https://support.dnsimple.com/articles/differences-a-cname-records/. [Accessed: 16-Aug-2021].

[73] T. Nidecki, “What are DNS zone transfers (AXFR)?”. 2019. [Online]. Available: https://www.acunetix.com/blog/articles/dns-zone-transfers-axfr/. [Accessed: 16-Aug-2021].

[74] M. Pramatarov, “DNS zone transfer and zone file - ClouDNS Blog”. 2021. [Online]. Available: https://www.cloudns.net/blog/zone-transfer-zone-file-domain-namespace/. [Accessed: 16-Aug-2021].

[75] Cloudflare, “What is robots.txt? | How a robots.txt file works | Cloudflare”. [Online]. Available: https://www.cloudflare.com/learning/bots/what-is-robots.txt/. [Accessed: 16-Aug-2021].

[76] Google Search Central, “Robots.txt Introduction & Guide | Google Search Central”. [Online]. Available: https://developers.google.com/search/docs/advanced/robots/intro. [Accessed: 16-Aug-2021].

[77] C. D. Stafford, “Weakest Link: Assessing factors that influence susceptibility to falling victim to phishing attacks and methods to mitigate”. *Cyber Security, Utica College, Master Thesis*, Aug 2020.

[78] R. Dhamija, J. D. Tygar, and M. Hearst, “Why phishing works”. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. Association for Computing Machinery, New York, USA, pp. 581–590, Apr 2006. DOI: 10.1145/1124772.1124861.

[79] M. Volkamer, K. Renaud, B. Reinheimer, and A. Kunz, “User experiences of TORPEDO: TOoltip-poweRed Phishing Email DetectiOn”. *In Journal of Computers & Security*, vol. 71, pp. 100–113, Nov 2017. DOI: 10.1016/j.cose.2017.02.004.

[80] R. Vidwans, “Beware of Shortened URLs in Phishing Scams.” [Online]. Available: https://www.clearedin.com/blog/shortened-urls-in-phishing-scams. [Accessed: 16-Aug-2021].

[81] Carnegie Mellon University, “Shortened URL Security - Information Security Office - Computing Services - Carnegie Mellon University”. [Online]. Available: https://www.cmu.edu/iso/aware/dont-take-the-bait/shortened-url-security.html. [Accessed: 16-Aug-2021].

[82] K. Krombholz, P. Frühwirt, P. Kieseberg, I. Kapsalis, M. Huber, and E. Weippl, “QR Code Security: A Survey of Attacks and Challenges for Usable Security”. *In Proceedings of International Conference on Human Aspects of Information Security, Privacy, and Trust (HAS 2014): Human Aspects of Information Security, Privacy, and Trust. Part of the Lecture Notes in Computer Science book series,* vol 8533. Springer, Cham, pp. 79–90, 2014. DOI: 10.1007/978-3-319-07620-1_8.

[83] A. Kharraz, E. Kirda, W. Robertson, D. Balzarotti, and A. Francillon, “Optical delusions: A study of malicious QR codes in the wild”. *In Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 192-203. Jun 2014. DOI: 10.1109/DSN.2014.103.

[84] V. Sharma, “A Study of Malicious QR Codes”. *In International Journal of Computational Intelligence and Information Security*, vol. 3, no. 5, pp. 12-17, May 2012.

[85] A. P. Felt and D. Wagner, “Phishing on Mobile Devices”. *In Proceedings of 2011 IEEE Symposium Conference on Security and Privacy, workshop program: web 2.0 security and privacy 2011, w2sp 2011*, Oakland, California, May 2011.

[86] Y. Niu, F. Hsu, and H. Chen, “iPhish: Phishing Vulnerabilities on Consumer Electronics”. *In Proceedings of the 1st Conference on Usability, Psychology, and Security (UPSEC'0*8). *USENIX Association,* USA, Article No.10, pp. 1–8, Apr 2008. DOI: 10.5555/1387649.1387659.

[87] K. Peng, H. Sanabria, D. Wu, and C. Zhu, “Security Overview of QR Codes”. *Electrical Engineering and Computer Science,* School of Engineering, *Massachusetts Institute of Technology, Master Thesis,* May 2014*.*

[88] P. Kieseberg *et al.*, “QR code security”. *In Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia, MoMM2010,* Paris*,* pp. 430-435, Nov 2010. DOI: 10.1145/1971519.1971593.

[89] K. Krombholz, P. Frühwirt, T. Rieder, I. Kapsalis, J. Ullrich, and E. Weippl, “QR Code Security - How Secure and Usable Apps Can Protect Users Against Malicious QR Codes”. *In Proceedings of the 10th International Conference on Availability, Reliability and Security*, pp. 230-237, Aug 2015. DOI: 10.1109/ARES.2015.84.

[90] H. Yao and D. Shin, “Towards preventing QR code based attacks on android phone using security warnings”. *In Proceedings of the 8th ACM SIGSAC symposium on*

Information, computer and communications security (ASIA CCS '13). Association for Computing Machinery, New York, USA, pp. 341–346, May 2013. DOI: 10.1145/2484313.2484357.

[91] M. S. Mark, “An analysis of factors influencing phishing threat avoidance behavior: a quantitative study”. *School of Business and Technology, Capella University, PhD Thesis,* Mar 2021.

[92] J. Nicholson, L. Coventry, and P. Briggs, “Can we fight social engineering attacks by social means? assessing social salience as a means to improve phish detection”. *In Proceedings of the 13th USENIX Conference on Usable Privacy and Security (SOUPS '17). USENIX Association*, USA, 2017, pp. 285–298, Jul 2017. DOI: 10.5555/3235924.3235948.

[93] L. Wu, X. Du, and J. Wu, “Effective Defense Schemes for Phishing Attacks on Mobile Computing Platforms”. *In Journal of IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 6678-6691, Aug. 2016, DOI: 10.1109/TVT.2015.2472993.

[94] H. Shahriar, T. Klintic, V. Clincy, H. Shahriar, T. Klintic, and V. Clincy, “Mobile Phishing Attacks and Mitigation Techniques”. *In Journal of Information Security,* vol. 6, no.3, pp. 206-212, May 2015. DOI: 10.4236/jis.2015.63021.

[95] Y. Zhang, J. I. Hong, and L. F. Cranor, “Cantina: A content-based approach to detecting phishing web sites”. *In Proceedings of the 16th international conference on World Wide Web (WWW '07)*. Association for Computing Machinery, New York, USA, pp. 639–648, May 2007. DOI: 10.1145/1242572.1242659.

[96] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, “Detecting malicious URLs using lexical analysis”. *In Proceedings of International Conference on Network and System Security (NSS 2016): Network and System Security. Part of the Lecture Notes in Computer Science book series,* vol 9955. Springer, Cham. pp. 467–482, Sep 2016. DOI: 10.1007/978-3-319-46298-1_30.

[97] S. Sheng and B. Wardman, “ An Empirical Analysis of Phishing Blacklists”. *In Proceedings of the 6th International Conference on Email and Anti-Spam (CEAS '2009),* California, USA, July 2009. DOI: 10.1184/R1/6469805.V1.

[98] BBC, “GDPR ‘risks making it harder to catch hackers’ ”. May 2018. [Online]. Available: https://www.bbc.co.uk/news/technology-44290019. [Accessed: 16-Aug-2021].

[99] G. Liu, G. Xiang, B. A. Pendleton, J. I. Hong, and W. Liu, “Smartening the crowds:

Computational techniques for improving human verification to fight phishing scams”. *In Proceedings of the 17th Symposium on Usable Privacy and Security (SOUPS '11).* Association for Computing Machinery, New York, USA, Article 8, pp. 1–13, Jul 2011. DOI: 10.1145/2078827.2078838.

[100] S. Afroz and R. Greenstadt, “PhishZoo: Detecting Phishing Websites by Looking at Them”. *In Proceedings of 2011 IEEE 5th International Conference on Semantic Computing*, Palo Alto, USA, Sep 2011, pp. 368-375. DOI: 10.1109/ICSC.2011.52.

[101] G. Xiang, B. A. Pendleton, J. Hong, and C. P. Rose, “A hierarchical adaptive probabilistic approach for zero hour Phish detection”. *In Proceedings of the 15th European conference on Research in computer security (ESORICS '10). Part of the Lecture Notes in Computer Science book series,* vol 6345. Springer, Berlin, Heidelberg, pp. 268-285, 2010. DOI: 10.1007/978-3-642-15497-3_17.

[102] A. Z. Broder, “Syntactic clustering of the Web”. *In Journal of Computer Networks and ISDN Systems,* vol. 29, no. 8-13, pp. 1157-1166, Sep. 1997. DOI: 10.1016/S0169-7552(97)00031-7.

[103] G. G. Geng, Z. W. Yan, Y. Zeng, and X. B. Jin, “RRPhish: Anti-phishing via mining brand resources request”. *In Proceedings of 2018 IEEE International Conference on Consumer Electronics (ICCE '2018)*, pp. 1-2, Jan 2018. DOI: 10.1109/ICCE.2018.8326085.

[104] S. Bell, “How Many Phish Can Tweet? Investigating the Effectiveness of Twitter’s Phishing and Malware Defence System”. *Department of Information Security, Royal Holloway, University of London, PhD Thesis*, Jun 2020.

[105] V. Drury, L. Lux, and U. Meyer, “Dating Phish: An Analysis of the Life Cycles of Phishing Attacks and Campaigns”. *In Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES '22).* Association for Computing Machinery, New York, USA, Article 14, pp. 1–11, Aug 2022. DOI: 10.1145/3538969.3538997.

[106] APWG, “Phishing Activity Trends Reports”. Q3 2017. [Online]. Available: https://apwg.org/trendsreports/. [Accessed: 04-Jun-2022].

[107] APWG, “Phishing Activity Trends Reports”. Q4 2020. [Online]. Available: https://apwg.org/trendsreports/. [Accessed: 04-Jun-2022].

[108] H. Zuhair, A. Selamat, and M. Salleh, “New hybrid features for phish website

prediction". *In International Journal of Advances in Soft Computing & Its Applications,* vol. 8, no. 1, pp. 28-43, Mar 2016.

[109] G. Xiang and J. I. Hong, "A hybrid phish detection approach by identity discovery and keywords retrieval". *In Proceedings of the 18th international conference on World wide web (WWW '09).* Association for Computing Machinery, New York, USA, pp. 571–580, Apr 2009. DOI: 10.1145/1526709.1526786.

[110] R. Basnet, A. Sung, and Q. Liu, "Rule-Based Phishing Attack Detection". *In Proceedings of the International Conference on Security and Management (SAM). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2011*.

[111] R. Basnet, A. Sung, and Q. Liu, "Feature Selection for Improved Phishing Detection". *In Proceedings of Advanced Research in Applied Artificial Intelligence, (IEA/AIE 2012). Part of the Lecture Notes in Computer Science book series*, vol. 7345, Springer, Berlin, pp. 252-261, 2012. DOI: 10.1007/978-3-642-31087-4_27.

[112] Q. Cui, "Detection and Analysis of Phishing Attacks". *School of Electrical Engineering and Computer Science, Faculty of Engineering, University of Ottawa, PhD Thesis*, 2019.

[113] J. H. Huh and H. Kim, "Phishing Detection with Popular Search Engines: Simple and Effective". *In Proceedings of the 4th Canada-France MITACS conference on Foundations and Practice of Security (FPS'11). Part of the Lecture Notes in Computer Science book series,* vol. 6888, Springer-Verlag, Berlin, Heidelberg, pp. 194–207, 2011. DOI: 10.1007/978-3-642-27901-0_15.

[114] J. W. I. A. Botejue, "Phishing Website Detection". *School of Computing, University of Colombo, Master Thesis*, 2022.

[115] A. P. E. Rosiello, E. Kirda, C. Kruegel, and F. Ferrandi, "A layout-similarity-based approach for detecting phishing pages". *In Proceedings of the 3th International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm 2007*, Nice, France, pp. 454-463, Sep 2007. DOI: 10.1109/SECCOM.2007.4550367.

[116] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites". *In Journal of ACM Transactions on Information and System Security,* vol. 14, no. 2, Article 21, pp. 1-28,

Sep 2011. DOI: 10.1145/2019599.2019606.

[117] Q. Cui, G. V. Jourdan, G. V. Bochmann, R. Couturier, and I. V. Onut, “Tracking phishing attacks over time”. *In Proceedings of the 26th International Conference on World Wide Web (WWW '17). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE*, pp. 667–676, Apr 2017. DOI: 10.1145/3038912.3052654.

[118] J. Feng, Y. Zhang, and Y. Qiao, “A detection method for phishing web page using DOM-based Doc2Vec model”. *In Journal of Computing and Information Technology*, vol. 28, pp. 19-31. July 2020. DOI: 10.20532/cit.2020.1004899.

[119] M. Dunlop, S. Groat, and D. Shelly, “GoldPhish: Using images for content-based phishing analysis”. *In Proceedings of the 5th International Conference on Internet Monitoring and Protection*, Barcelona, Spain, pp. 123-128, May 2010. DOI: 10.1109/ICIMP.2010.24.

[120] E. S. Aung and H. Yamana, “URL-based phishing detection using the entropy of non- A lphanumeric characters”. *In Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services (iiWAS ‘2019). Association for Computing Machinery, New York, USA, pp. 385–392, Dec 2019. DOI: 10.1145/3366030.3366064.*

[121] T. Nathezhtha, D. Sangeetha, and V. Vaidehi, “WC-PAD: Web crawling based phishing attack detection”. *In Proceedings of 2019 International Carnahan Conference on Security Technology (ICCST)*, Chennai, India, pp. 1-6, Oct 2019. DOI: 10.1109/CCST.2019.8888416.

[122] A. Niakanlahiji, B. T. Chu, and E. Al-Shaer, “PhishMon: A machine learning framework for detecting phishing webpages”. *In Proceedings of 2018 IEEE International Conference on Intelligence and Security Informatics (ISI),* Miami, USA, pp. 220-225, Dec 2018. DOI: 10.1109/ISI.2018.8587410.

[123] M. Blythe, H. Petrie, and J. A. Clark, “F for fake: Four studies on how we fall for phish,” *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11).* Association for Computing Machinery, New York, USA, pp. 3469–3478, May 2011. DOI:10.1145/1978942.1979459.

[124] M. Jakobsson, “The Human Factor in Phishing”. *In Journal of Japan’s Economic Power and Security*, vol. 07, 2007. DOI: 10.4324/9780203257487_chapter_8.

[125] M. J. Dupuis and S. Smith, “Clickthrough Testing for Real-World Phishing Simulations”. *In Proceedings of the 21st Annual Conference on Information Technology Education (SIGITE '20).* Association for Computing Machinery, New York, , USA, pp. 347, Oct 2020. DOI:10.1145/3368308.3415443.

[126] K. K. Greene, M. P. Steves, M. F. Theofanos, and J. Kostick, “User Context: An Explanatory Variable in Phishing Susceptibility”. *In Proceedings of the Network and Distributed Systems Security (NDSS) Symposium, Workshop on Usable Security (USEC) 2018*, San Diego, CA, US, 2018. DOI: 10.14722/usec.2018.23016.

[127] M. P. Steves, K. K. Greene, and M. F. Theofanos, “A Phish Scale: Rating Human Phishing Message Detection Difficulty”. *In Proceedings of 2019 Workshop on Usable Security*, San Diego, California, United States, 2019. DOI: 10.14722/usec.2019.23028.

[128] C. N. Gutierrez *et al.*, “Learning from the ones that got away: Detecting new forms of phishing attacks”. *In Journal of IEEE Transactions on Dependable and Secure Computing,* vol. 15, no. 6, pp. 988-1001, Dec. 2018. DOI: 10.1109/TDSC.2018.2864993.

[129] S. P. Ripa, F. Islam, and M. Arifuzzaman, “The emergence threat of phishing attack and the detection techniques using machine learning models”. *In Proceedings of 2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*, Rajshahi, Bangladesh, pp. 1-6, Jul 2021. DOI: 10.1109/ACMI53878.2021.9528204.

[130] S. R. Curtis, P. Rajivan, D. N. Jones, and C. Gonzalez, “Phishing attempts among the dark triad: Patterns of attack and vulnerability”. *In Journal of Computers in Human Behaviour*, vol. 87, pp. 174–182, Oct. 2018. DOI: 10.1016/j.chb.2018.05.037.

[131] R. Fatima, A. Yasin, L. Liu, and J. Wang, “How persuasive is a phishing email? A phishing game for phishing awareness”. *In Journal of Computer Security,* vol. 27, no. 6, pp 581–612, Oct 2019. DOI: 10.3233/JCS-181253.

[132] S. Mansfield-Devine, “Bad behaviour: exploiting human weaknesses,”. *In Journal of Computer Fraud & Security*, vol. 2017, no. 1, pp. 17-20, Jan 2017. DOI: 10.1016/S1361-3723(17)30008-8.

[133] A. Yasin, R. Fatima, L. Liu, J. Wang, R. Ali, and Z. Wei, “Understanding and deciphering of social engineering attack scenarios”. *In Journal of Security and Privacy*, vol. 4, no. 4, pp. e161, Jul 2021. DOI: 10.1002/spy2.161.

[134] J. W. H. Bullée, L. Montoya, W. Pieters, M. Junger, and P. Hartel, “On the anatomy of social engineering attacks—A literature-based dissection of successful attacks”. *In Journal of Investigative Psychology and Offender Profiling*, vol. 15, no. 1, pp. 20-45, Jan 2018. DOI: 10.1002/jip.1482.

[135] T. Kelley, M. J. Amon, and B. I. Bertenthal, “Statistical models for predicting threat detection from human behavior”. *In Journal of Frontiers in Psychology*, vol. 9, pp. 466, Apr 2018. DOI:10.3389/fpsyg.2018.00466.

[136] D. M. Sarno, J. E. Lewis, C. J. Bohil, and M. B. Neider, “Which Phish Is on the Hook? Phishing Vulnerability for Older Versus Younger Adults”. *In Journal of Human Factors*, vol. 62, no. 5, pp. 704-717, Aug 2020. DOI: 10.1177/0018720819855570.

[137] T. Lin *et al.*, “Susceptibility to spear-phishing emails: Effects of internet user demographics and email content,” *In Journal of ACM Transactions on Computer-Human Interaction*, vol. 26, no. 5, Article. 32, pp. 1-28, Oct 2019. DOI: 10.1145/3336141.

[138] R. Mata, A. K. Josef, G. R. Samanez-Larkin, and R. Hertwig, “Age differences in risky choice: A meta-analysis”. *In Journal of the Annals of the New York Academy of Sciences*, vol. 1235, no. 1, pp. 18-29, Oct 2011. DOI: 10.1111/j.1749-6632.2011.06200.x.

[139] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, “Advanced social engineering attacks”. *In Journal of Information Security and Applications*, vol. 22, pp. 113-122, Jun 2015. DOI: 10.1016/j.jisa.2014.09.005.

[140] L. T. Hove, “Strategies Used to Mitigate Social Engineering Attacks”. *College of Management and Technology, Walden University, PhD Thesis, 2020*.

[141] L. V. Astakhova and I. A. Medvedev, “An Information Tool for Increasing the Resistance of Employees of an Organization to Social Engineering Attacks”. *In Journal of Scientific and Technical Information Processing,* vol. 48, no. 1, pp. 15-20, May 2021. DOI: 10.3103/S0147688221010020.

[142] R. Heartfield and G. Loukas, “Detecting semantic social engineering attacks with the weakest link: Implementation and empirical evaluation of a human-as-a-security-sensor framework”. *In Journal of Computers & Security,* vol. 76, pp. 101-127, Jul 2018. DOI: 10.1016/j.cose.2018.02.020.

[143] S. Lswarya and D. S. Preetha, “Social Engineering Attacks in Online Banking-Analysis of

Trends and Prevention". *In Journal of Eurasian Journal of Analytical Chemistry*, vol. 13, pp. 390-400, Apr 2019.

[144] S. Abraham and I. S. Chengalur-Smith, "An overview of social engineering malware: Trends, tactics, and implications". *In Journal of Technology in Society,* vol. 32, no. 3, pp. 183-196, Aug 2010. DOI: 10.1016/j.techsoc.2010.07.001.

[145] M. Lansley, N. Polatidis, S. Kapetanakis, K. Amin, G. Samakovitis, and M. Petridis, "Seen the villains: Detecting Social Engineering Attacks using Case-based Reasoning and Deep Learning". *In Proceedings of the 21st International Conference on Case-Based Reasoning (ICCBR '2019), Workshop on Case-Based Reasoning and Deep Learning, Germany,* vol. 2567, pp. 39-48, Sep 2019.

[146] K. Tian, "SquatPhish/3-Phish-Page-Detection". [Online]. Available: https://github.com/SquatPhish/3-Phish-Page-Detection. [Accessed: 04-Jun-2022]

[147] F. Aloul, S. Zahidi, and W. El-Hajj, "Two factor authentication using mobile phones". *In Proceedings of 2009 IEEE/ACS International Conference on Computer Systems and Applications*, Rabat, Morocco, pp. 641-644, May 2009. DOI: 10.1109/AICCSA.2009.5069395.

[148] CenterPoint, "Did You Know That New Phishing Attacks Make 2FA Useless? ". [Online]. Available: https://www.centerpointit.com/know-new-phishing-attacks-make-2fa-useless/. [Accessed: 04-Jun-2022].

[149] S. Hawa Apandi, J. Sallim, and R. Mohd Sidek, "Types of anti-phishing solutions for phishing attack". *In Proceedings of 2020 IOP Conference Series: Materials Science and Engineering: the 6th International Conference on Software Engineering & Computer Systems,* Pahang, Malaysia, vol. 769, no. 1, Sep 2019. DOI: 10.1088/1757-899X/769/1/012072.

[150] Titanadmin, "Does 2-Factor Authentication Stop Phishing Attacks? - SpamTitan". 2019. [Online]. Available: https://www.spamtitan.com/blog/does-2-factor-authentication-stop-phishing-attacks/. [Accessed: 04-Jun-2022].

[151] E. Ulqinaku, D. Lain, and S. Capkun, "2FA-PP: 2nd factor phishing prevention". *In Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '19)*. Association for Computing Machinery, New York, USA, pp. 60–70, May 2019. DOI: 10.1145/3317549.3323404.

[152] A. Dawson, "Exploring Strategies for Implementing Information Security Training and

Employee Compliance Practices”. *College of Management and Technology, Walden University, PhD Thesis*, Dec 2019.

[153] N. A. G. Arachchilage, S. Love, and K. Beznosov, “Phishing threat avoidance behaviour: An empirical investigation”. *In Journal of Computers in Human Behavior*, vol. 60, pp. 185-197, Jul 2016. DOI: 10.1016/j.chb.2016.02.065.

[154] M. Fernando and N. Arachchilage, “Why Johnny can’t rely on anti-phishing educational interventions to protect himself against contemporary phishing attacks?”. *In Proceedings of the 30th Australasian Conference on Information Systems,* Perth (Australia), Dec 2019.

[155] S. Garera, N. Provos, M. Chew, and A. D. Rubin, “A framework for detection and measurement of phishing attacks”. *In Proceedings of the 2007 ACM workshop on Recurring malcode (WORM '07)*. Association for Computing Machinery, New York, USA, pp. 1–8, Nov 2007. DOI: 10.1145/1314389.1314391.

[156] J. W. Bullee and M. Junger, “How effective are social engineering interventions? A meta-analysis”. *In Journal of Information and Computer Security*, vol. 28, no. 5, pp. 801-830, Nov 2020. DOI: 10.1108/ICS-07-2019-0078.

[157] M. Edwards, R. Larson, B. Green, A. Rashid, and A. Baron, “Panning for gold: Automatically analysing online social engineering attack surfaces”. *In Journal of Computers & Security,* vol. 69, pp. 18-34, Aug 2017. DOI: 10.1016/j.cose.2016.12.013.

[158] N. H. Flores, A. C. R. Paiva, and P. Letra, “Software Engineering Management Education through Game Design Patterns”. *In Journal of Procedia - Social and Behavioral Sciences*, vol. 228, pp. 436-442, Jul 2016. DOI: 10.1016/j.sbspro.2016.07.067.

[159] R. Zaimi, M. Hafidi, and M. Lamia, “Survey paper: Taxonomy of website anti-phishing solutions”. *In Proceedings of the 7th International Conference on Social Networks Analysis, Management and Security (SNAMS ‘2020)*, Paris, France, pp. 1-8, Dec 2020. DOI: 10.1109/SNAMS52053.2020.9336559.

[160] MozillaWiki, “Security/Safe Browsing - MozillaWiki”. May-2021. [Online]. Available: https://wiki.mozilla.org/Security/Safe_Browsing. [Accessed: 16-Aug-2021].

[161] I. Bonifacic, “Apple puts additional walls between your browsing data and Google on iOS 14.5 | Engadget”. 2021. [Online]. Available: https://www.engadget.com/ios-14-5-safari-safe-browsing-173836695.html?guccounter=1. [Accessed: 16-Aug-2021].

[162] DeviceAtlas, “The most popular mobile browsers”. 2019. [Online]. Available: https://deviceatlas.com/blog/the-most-popular-mobile-browsers. [Accessed: 04-Jun-2022].

[163] Statista, “Mobile browser market share worldwide 2012-2021 | Statista”. 2021. [Online]. Available: https://www.statista.com/statistics/263517/market-share-held-by-mobile-internet-browsers-worldwide/. [Accessed: 04-Jun-2022].

[164] K. Bali, “Why Google is Forcing You To Have SSL Certificate on Your Websites”. 2020. [Online]. Available: https://serverguy.com/ssl/google-forcing-ssl-certificate-websites/. [Accessed: 16-Aug-2021].

[165] B. Nahorney, “DNS under attack - Cisco Blogs”. Jul 2019. [Online]. Available: https://blogs.cisco.com/security/dns-under-attack. [Accessed: 06-Mar-2020].

[166] Imperva, “What is a DNS Hijacking | Redirection Attacks Explained | Imperva”. [Online]. Available: https://www.imperva.com/learn/application-security/dns-hijacking-redirection/. [Accessed: 16-Aug-2021].

[167] T. Mocan, “What Is DNS Hijacking? (How to Stop DNS Hijacking) | CactusVPN”. Jun 2019. [Online]. Available: https://www.cactusvpn.com/beginners-guide-online-security/dns-hijacking/. [Accessed: 16-Aug-2021].

[168] W. Rash, “How to Avoid the New DNS Hijacking Attacks - eWEEK”. 2019. [Online]. Available: https://www.eweek.com/security/how-to-avoid-the-new-dns-hijacking-attacks/. [Accessed: 16-Aug-2021].

[169] A. A. Maksutov, I. A. Cherepanov, and M. S. Alekseev, “Detection and prevention of DNS spoofing attacks”. *In Proceedings of 2017 Siberian Symposium on Data Science and Engineering (SSDSE '2017),* Novosibirsk, Russia, pp. 84-87, Oct 2017. DOI: 10.1109/SSDSE.2017.8071970.

[170] L. Garron, A. B. Dropbox, and D. Boneh, “The State of HSTS Deployment: A Survey and Common Pitfalls,” *In Journal of Computer Science Article*, 2014.

[171] M. Dahl, “Widespread DNS Hijacking Activity Targets Multiple Sectors”. Jan 2019. [Online]. Available: https://www.crowdstrike.com/blog/widespread-dns-hijacking-activity-targets-multiple-sectors/. [Accessed: 16-Aug-2021].

[172] P. Wang and X. Chen, “Co_Hijacking Monitor: Collaborative Detecting and Locating Mechanism for HTTP Spectral Hijacking”. *In Proceedings of 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive*

Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), Orlando, FL, USA, pp. 61-67, Nov 2017. DOI: 10.1109/DASC-PICom-DataCom-CyberSciTec.2017.218

[173] T. Keary, “8 Best DNS Protection Solutions for Your Network (Top DNS Tools)”. Jul 2019. [Online]. Available: https://www.comparitech.com/net-admin/best-dns-protection-solutions/. [Accessed: 06-Mar-2020].

[174] T. Foltýn, “Majority of the world’s top million websites now use HTTPS | WeLiveSecurity”. Sep 2018. [Online]. Available: https://www.welivesecurity.com/2018/09/03/majority-worlds-top-websites-https/. [Accessed: 15-Mar-2021].

[175] Proofpoint, “2019 Proofpoint Domain Fraud Report”. 2019. [Online] Available: https://www.proofpoint.com/us/resources/white-papers/domain-fraud-report [Accessed: 15-Mar-2020]

[176] N. Lorenz, “20% of the world 502 largest websites do not use HTTPS - Avira Blog”. [Online]. Available: https://blog.avira.com/20-of-the-world-502-largest-websites-do-not-use-https/. [Accessed: 06-Mar-2020].

[177] T. Hunt, “Why No HTTPS? The World’s Largest Websites Not Redirecting Insecure Requests to HTTPS.” [Online]. Available: https://whynohttps.com/. [Accessed: 06-Mar-2020].

[178] Websitesetup, “Internet Statistics & Facts (Including Mobile) for 2021”. 2011. [Online]. Available: https://websitesetup.org/news/internet-facts-stats/. [Accessed: 15-Mar-2021].

[179] Y. Nadji, P. Saxena, and D. Song, “Document Structure Integrity: A Robust Basis for Cross-site Scripting Defense”. *In Proceedings of the Network and Distributed System Security Symposium, (NDSS) 2009*, San Diego, California, USA, Feb 2009.

[180] C. Reis, S. D. Gribble, T. Kohno, and N. C. Weaver, “Detecting In-Flight Page Changes with Web Tripwires,” *In Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08). USENIX Association*, USA, pp. 31–44, Apr 2008. DOI: 10.5555/1387589.1387592

[181] V. Sumina, “26 Cloud Computing Statistics, Facts & Trends for 2022”. Jun 2022. [Online]. Available: https://www.cloudwards.net/cloud-computing-statistics/.

[Accessed: 23-Jul-2022].

[182] N. Gilbert, “64 Significant Cloud Computing Statistics for 2022: Usage, Adoption & Challenges”. 2022. [Online]. Available: https://financesonline.com/cloud-computing-statistics/. [Accessed: 23-Jul-2022].

[183] I. Messina, “Not secure website warning in browser - SupportHost”. 2021. [Online]. Available: https://supporthost.com/not-secure-website/. [Accessed: 04-Jun-2022].

[184] P. Hudak, “Subdomain Takeover: Basics”. [Online]. Available: https://0xpatrik.com/subdomain-takeover-basics/. [Accessed: 03-Mar-2021].

[185] E. Borges, “Top 7 Subdomain Scanner tools to find subdomains”. 2019. [Online]. Available: https://securitytrails.com/blog/subdomain-scanner-find-subdomains. [Accessed: 03-Mar-2021].

[186] S. M. Z. U. Rashid, M. I. Kamrul, and A. Islam, “Understanding the Security Threats of Esoteric Subdomain Takeover and Prevention Scheme”. *In Proceedings of the 2nd International Conference on Electrical, Computer and Communication Engineering (ECCE '2019),* Bangladesh, pp. 1-4, Apr 2019. DOI: 10.1109/ECACE.2019.8679122.

[187] G. Jaspher *et al.*, “Comparative analysis of subdomain enumeration tools and static code analysis”. *In Journal of Mechanics of Continua and Mathematical Sciences*, vol. 15, no. 6, pp. 158-173, Jun 2020. DOI: 10.26782/jmcms.2020.06.00013.

[188] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. E. Mohamed, and H. Arshad, “State-of-the-art in artificial neural network applications: A survey”. *In Journal of Heliyon*, vol. 4, no. 11, pp. e00938, Nov 2018. DOI: 10.1016/j.heliyon.2018.e00938.

[189] C. Olah, “Understanding LSTM Networks”. Aug 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 03-Mar-2021].

[190] N. Leonard, “Language modeling a billion words”. Jul 2016. [Online]. Available: http://torch.ch/blog/2016/07/25/nce.html. [Accessed: 03-Mar-2021].

[191] J. Ling, “OneForAll/· GitHub”. 2021. [Online]. Available: https://github.com/shmilylty/OneForAll/blob/master/docs/en-us/README.md. [Accessed: 03-Mar-2021].

[192] Affordable Domains, “How many characters can a domain name have?”. [Online]. Available: https://www.affordabledomains.co.nz/knowledgebase/36/How-many-characters-can-a-domain-name-have.html. [Accessed: 16-Aug-2021].

[193] P. Cerda, G. Varoquaux, and B. Kégl, “Similarity encoding for learning with dirty categorical variables”. *In Journal of Machine Learning*, vol. 107, pp. 1477–1494, Jun 2018. DOI: 10.1007/s10994-018-5724-2.

[194] U. Rivner and E. Englund, “*Sherlock and Holmes fight Deep Social Engineering". In Proceedings of the 30th RSA Conference*, May 2021.

[195] Wikipedia, “Same-origin policy - Wikipedia”. 2021. [Online]. Available: https://en.wikipedia.org/wiki/Same-origin_policy. [Accessed: 16-Aug-2021].

[196] Cloudflare, “What is a DNS SOA record? | Cloudflare”. [Online]. Available: https://www.cloudflare.com/en-gb/learning/dns/dns-records/dns-soa-record/. [Accessed: 16-Aug-2021].

10. Appendix

The published report from CrowdStrike's threat intelligence team revealed, 28 organizations in twelve different countries were hijacked. The detailed summary is listed in Table 23 below:

Reported (Malicious)IP Address	Date	Affected Organizations' Country (Sector)
142.54.179.69	Feb 2017	Jordan (Government)
89.163.206.26	Feb 2017	Jordan (Government)
185.15.247.140	Dec 2017 and Jan 2018	Kuwait (Government) Albania (Government)
146.185.143.158	Aug 2018	UAE (Government)
128.199.50.175	Sep 2018	UAE (Unidentified Sector)
185.20.187.8	Sep 2018	UAE (Law Enforcement) UAE (Government) Lebanon (Government) Lebanon (Civil Aviation)
82.196.8.43	Oct 2018	Iraq (Government)
188.166.119.57	Oct and Nov 2018	Egypt (Government) Libya (Government)
206.221.184.133	Nov 2018	Egypt (Government)
37.139.11.155	Nov 2018	UAS (Unidentified Sector)
199.247.3.191	Nov 2018	Iraq (Government) Albania (Government)
185.161.209.147	Nov 2018	Lebanon (Insurance)
139.162.144.139	Dec 2018	Jordan (Government)
37.139.11.155	Dec 2018	UAE (Unidentified Sector)
178.62.218.244	Dec 2018	UAE (Government)
139.59.134.216	Dec 2018	Sweden (Internet Infrastructure)

		Saudi Arabia (Internet Services) Lebanon (Internet Services)
82.196.11.127	Dec 2018	Sweden (Internet Infrastructure) U.S. (Internet Infrastructure)
46.101.250.202	Dec 2018 and Jan 2019	Saudi Arabia (Government)

TABLE 23: The summary of organization hijacked sector since 2017 – 2019 **[171]**.