Cost-Effective OCR Implementation to Prevent Phishing on Mobile Platforms

Yunjia Wang
School of Computer Science
University of St Andrews
St Andrews, United Kingdom
yw43@st-andrews.ac.uk

Yang Liu
Independent Scholar
Beijing, China
liuyang1026@outlook.com

Tiejun Wu
NSFocus IT Co Ltd
Beijing, China
wutiejun@nsfocus.com

Ishbel Duncan
School of Computer Science
University of St Andrews
St Andrews, United Kingdom
immd@st-andrews.ac.uk

***Abstract*—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. Many users are sensible about this kind of attack from suspicious URL addresses or obvious warning information from browsers, but phishing still accounts for a larger proportion of all of malicious attacks. Moreover, these warning features will be eliminated if the victim is under a DNS hijacking attack. There is much research about the prevention and evaluation of phishing, in both PC platforms and mobile platforms, but there are still technical challenges to reducing the risk from phishing, especially in mobile platforms.**

We presented a novel method to prevent phishing attacks by using an Optical Character Recognition (OCR) technology in a previous paper. This method not only overcomes the limitation of current preventions, but also provides a high detection accuracy rate. However, whether this method can be implemented ideally in mobile devices needed to be further examined, especially in relation to the challenges of limited resources (power and bandwidth). In this paper, we apply the OCR method in a mobile platform and provide a prototype implementation scheme to determine applicability. Experiments are performed to test the technique under DNS hijacking attacks.

***Index Terms*—Phishing, OCR, Mobile Phishing Prevention**

I. Introduction

Phishing is a social engineering attack in which the purpose is to steal user data, such as private information or bank information etc. As shown by existing data from the Anti-Phishing Working Group (APWG) phishing activity trends report, the number of phishing sites detected was 266,387 in the third quarter of 2019, which was nearly double those detected in 2018 Q4 [1], as shown in Figure 1. Additionally, according to the Threat Landscape Report from ENISA in 2018 [2], 75% of EU's Member states disclosed that their enterprises had undergone cases of phishing. Over 90% of malware infection and 72% of data leakages in enterprises are from phishing attacks [3].

Phishing is currently defined as a criminal mechanism employing both social engineering and technical subterfuge to gather user personal identity data and financial account credentials by APWG [1]. With the increasing upgrade of technical approaches, phishing attacks have become more complicated. For example, general phishing can be identified through observing the URL path, because many users are sensible about this kind of attack from suspicious URL addresses

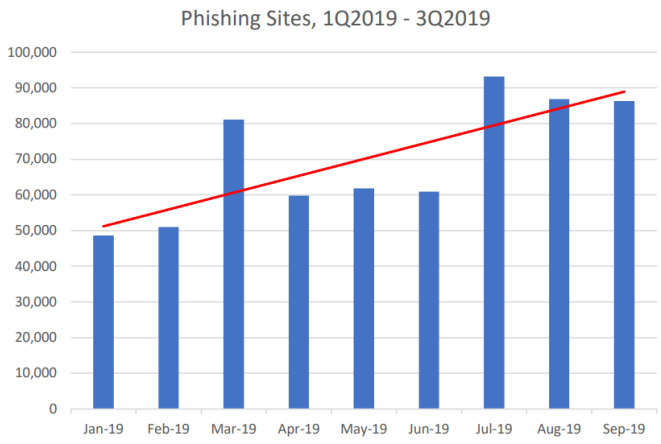

Fig. 1. APWG phishing activity trends report Q3 2019 [1]

or obvious warning information from browsers, as shown in Figure 2. However, all of these features will be eliminated if the victim is under a DNS hijacking attack. In this case, not only is the accessed URL the same as the official website, the associated SSL certificate is not displayed in an apparent way, as shown in Figure 3.

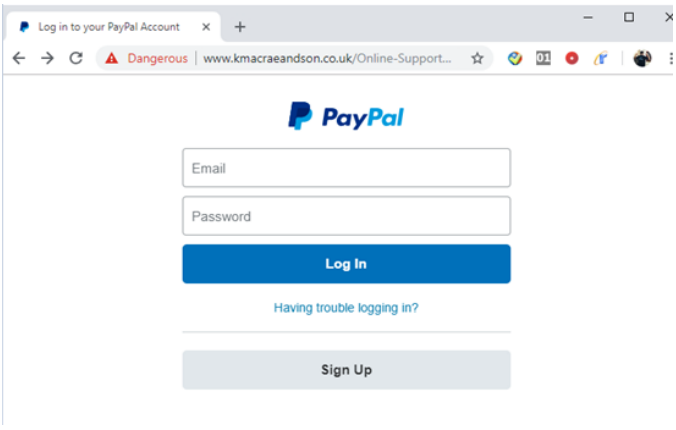

Fig. 2. general phishing URL about PayPal

There is much research about the prevention and evaluation

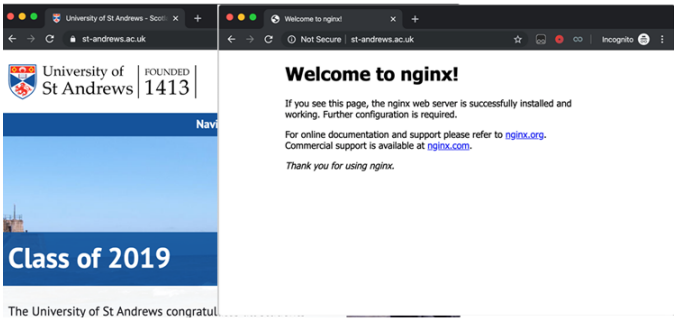

Fig. 3. two accessed results; normal and DNS hijacking

of phishing, in both PC platforms and mobile platforms, but it is still hard to reduce the risk from phishing. Spear phishing, a more targeted attack towards a specific individual, organisation or business [4], is the number one infection vector employed by 71 percent of organized groups according to the Internet Security Threat Report from Symantec in 2018, as shown in Figure 4 [5]. Also, most Advanced Persistent Threat (APT) groups have used spear phishing as the initial infection vector [5]. Thus, phishing threats are worth researching to determine cheaper and faster solutions. In our last paper [6], we presented a novel method to prevent phishing attacks by using an Optical Character Recognition (OCR) technology, and this method not only overcomes the limitation of current preventions, but also provides a high detection accuracy rate. The evaluation results were promising, but whether this method can be implemented ideally in mobile devices needed to be further examined, especially in relation to the challenges of limited resources (power and bandwidth). In this paper, we execute our novel method in a mobile platform and provide a prototype implementation scheme to determine its applicability. Also, this method will be examined under a DNS hijacking attack.

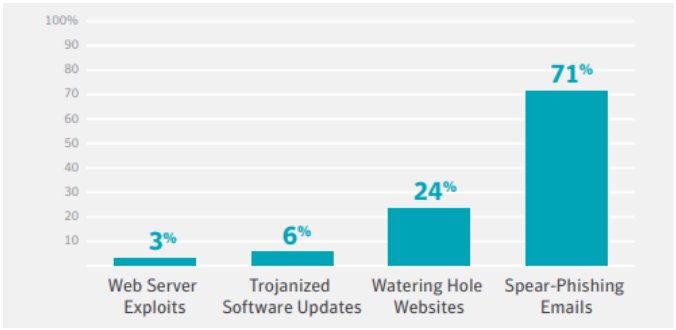

Fig. 4. Symantec, internet security threat report 2018

The remainder of this paper is structured as follows: in section 2, the related work is reviewed. In section 3, we describe the methodology used and subsequently the prototype implementation. The experimental results are evaluated in section 4 and finally, the conclusion is summarized in section 5.

II. Related Work

In this section, we focus on two aspects of the reviewed literature: DNS redirection and current phishing attacks on mobile devices. We also implemented several relevant experiments for each aspect to examine the risk.

DNS redirection, also known as 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. Normally, a targeted website IP address is indexed in a DNS mapping table after sending a DNS query from the user. Subsequently, this 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, the perpetrators 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. Typically, a DNS mapping table exists in both the user local computer (see Figure 5) and the remote system, such as a router or ISP server. Therefore, to perform this kind of DNS attack, perpetrators either take over a local DNS mapping table or intercept a DNS communication between the victim and a remote server [7].

```
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
```

Fig. 5. default local DNS setting on MAC OS

Mocan [8] summarized how most perpetrators 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

The author also mentioned another type of DNS hijacking, named as ISP hijacking, although it is not as dangerous as the DNS hijacking above [8]. ISP hijacking always take place in third-party advertisers; the purpose is to earn more profits by inserting extra HTML code to redirect user traffic, rather than to steal personal and financial data. Alternatively, the user will also be redirected to a fake website with several ads if they type in a website address that does not exist.

Most security specialists use Secure Sockets Layers (SSL) to protect their traffic [7] [9]. This can also be used for preventing ISP hijacking due to the asynchronous encryption; an ISP would not read your data in plaintext as it does not have your private key. 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' in 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 6. In addition, as the not matched error has a higher risk, the browser will warn only on the 'not trusted' if the SSL certificate has both problems.

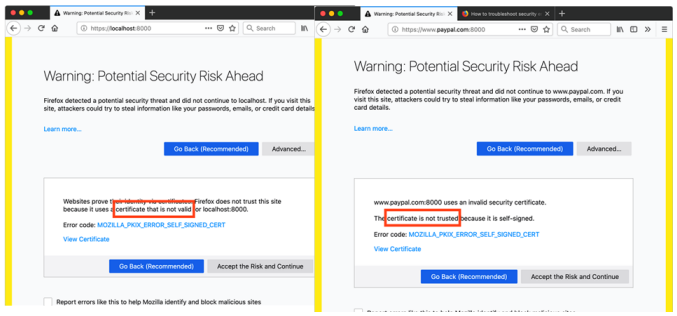

Fig. 6. SSL certificates with the not matched (valid) warning (left) and the not trusted warning (right)

An SSL channel could completely overcome this kind of hijacking attack, unless the SSL certificate has been stolen by perpetrators [9]. The fake website that has been redirected has an unmatched SSL certificate information with the targeted website. Alternatively, an evident Not Trusted error will result in a warning even if the perpetrators make their SSL certificate have the same information as the targeted one when under a DNS hijacking attack. However, this apparent warning can be eliminated through an HTTPS downgrade attack. The browser would not report any SSL certificate errors if the HTTPS connection has been downgraded to an HTTP connection. Only an unobvious 'not secure' symbol (see Figure 3) is displayed on the browser URL bar in either a PC or a mobile. Therefore, an advanced phishing attack, for example a DNS hijacking, can bypass the user's security awareness. So, not only does the accessed URL look the same as the official website, but also the SSL certificate warning is not displayed in an apparent way.

There are many open-source tools online which can be used to examine the network environment. Rash [9] mentioned two approaches to diagnose DNS hijacking, WHOISMYDNS[1] and Router Checker[2]. WHOISMYDNS is an online website, though it is 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.

Compared to a desktop or laptop computer, most mobile users claim that a mobile has less security software installed on their device. In 2018, around 25% of users indicated they did not know whether their mobile has been protected by any form of security software [10].

In addition, due to the global downward trend of buying computers, users have increasingly shifted towards using mobile devices for daily tasks. It is estimated that a mobile could be a user's only computer in less than two years [11]. Meanwhile, hackers have already focused their attention on mobile devices to gain more benefits, although a mobile is safer to use than PCs for several reason such as the sandbox on Android devices, code-signing on Androids and unexpected IP address[3] connections [12]. However, there are more restrictions influencing mobile security issues. Shahriar, Klintic and Clincy [13] listed 10 important factors that make the mobile web different from desktop usage. Most of the factors are based on the limited resources on mobile devices. Mobile application users are more vulnerable to phishing attacks [14]. They have difficulty in verifying if a page is legitimate due to a small screen, and some URLs are not often displayed within mobile browsers. Also, a fake address bar can be hidden by exploiting JavaScript scrollTo() within the page, though this risk has been fixed in the latest Android version [15]. Some mobile security software cannot be thoroughly executed to examine device security because of limited bandwidth, power usage and slower processes [13].

Moreover, for the mobile phishing attack, a user has less protection to avoid risks. In one previous experiment we collected 30 phishing URLs, which had been reported from PhishTank[4], and all of these URL connections elicited warnings when we tried to access them on a desktop browser. However, only 57% of the URLs [17] were identified as a phishing attack when the mobile used is an Android platform. Apple's devices fared even more poorly; less than half of the phishing URLs [13] were detected in a Safari browser, and the Chrome browser (on IOS) missed all of them, as shown in Figure 7:

There are two general approaches to mitigate phishing, a blacklist-based or a content based [13] [14] [16]. A blacklist-based scheme is a static approach, all of the malicious URLs in this list can be detected and identified to the user. A content based scheme detects phishing through extracting features from the URL. Some machine learning approaches are derived from this approach, such as using lexical and host-based analysis to categorize the features [17] [18] [19]. Nevertheless, both approaches are still not good enough to mitigate phishing attacks due to their limitations. In the first approach, a zero-day phishing attack cannot be detected as the blacklist-based scheme is not dynamic [13]. It may still steal user credentials if this is a newly created phishing website not already included in

[1]Available at:http://whoismydns.com/

[2]Available at:
https://www.f-secure.com/gb-en/home/free-tools/router-checker

[3]It is hard to confirm the IP address for mobile use unless it is connected to Wi-Fi

[4]Available at:http://www.phishtank.com/

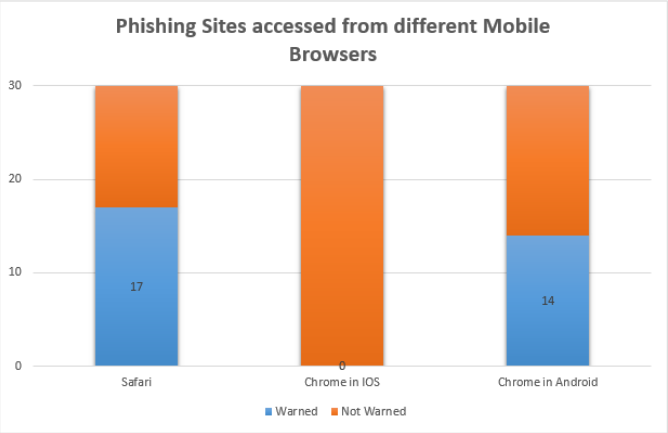

Fig. 7. mobile browser examination results

this list. In the second approach, the analysis result depends on the extracted features from the target website. For instance, in the host-based step, the server properties are investigated from WHOIS, collecting data such as the IP address, registration information etc. However, gathering this information may become a hard problem due to possible restrictions in the future [6]. In order to comply with the EU's General Data Protection Regulation (GDPR) legislation [20], most of this information may have been wiped as it releases too much private information.

In our last paper [6], we presented a novel method to prevent phishing by using Optical Character Recognition (OCR) technology. By analysing the logo content from a target website and comparing against the official website, the website could be confirmed as official. Subsequently the phishing website could be identified through the comparison of SSL certificates between the target website and the official website. This approach enables a high detection accuracy rate and the evaluation results look promising. However, there are several challenges if implementing this approach on mobile platforms. Unlike the desktop, it is hard to execute OCR computing on mobile devices due to their limited resources, either in power or in network bandwidth. Therefore, in this paper, we consider the following research questions:

- How to implement our OCR approach for mobile users?
- How to reduce the resource consumption as much as possible on a mobile client.

III. Methodology & Implementation

A. Research Purpose & Plan

In our previous paper [6], we researched OCR technology as a mean of identifying phishing. The specific procedure was divided to four steps, as shown below:

1) Extract the targeted website background / logo image using web crawler
2) Recognize the extracted image content using OCR technology
3) Confirm the official URL using Google Search
4) Verify the accessed URL through SSL certification comparison

The resources, either in bandwidth or in 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 might contain many images or icons so that it is difficult to locate where the actual logo image is. In our solution, we extracted all of the images to record these URLs. Thus, the CPU resource consumption will be high here. In step 2, all of these image URLs need to be analysed through OCR technology, so we adopted Google Vision API, which means many requests will be sent to the Google Cloud. As the bandwidth consumption is high, 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 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 needs to be implemented on the server side in order to improve the efficiency and reduce the mobile resource consumption. In order to prove the feasibility of this approach on mobiles, we simulated an environment that assumed a script in a mobile browser by using mitmproxy[5] , as shown in Figure 8 below.

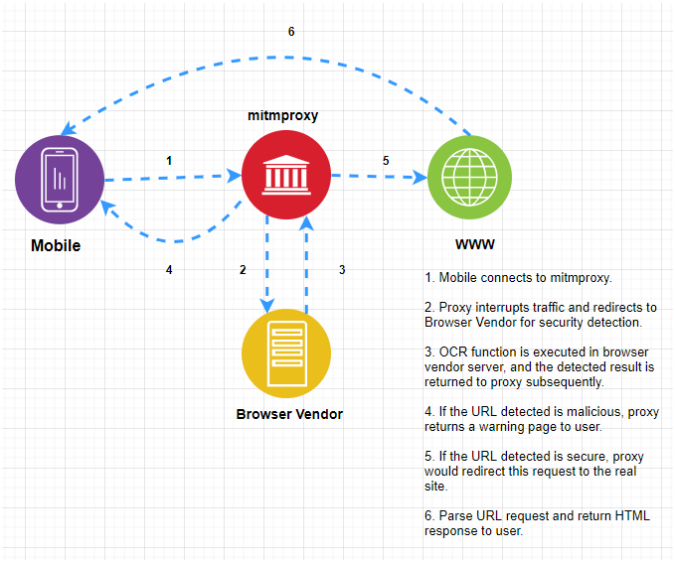

Fig. 8. illustration of experimental environment

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 our server and return a result (true or false) to the proxy. The final response is either a warning to the user or a redirect request to the real site, confirmed by the proxy. The mobile and mitmproxy can be assumed to

[5]Mitmproxy, is a free and open source interactive HTTPS proxy. Available at:http://mitmproxy.org/

be an extra functional browser on a mobile device, the server is the vendor of this browser. Therefore, in this environment, both the bandwidth and CPU resources on a mobile would not consume extra data. For the bandwidth, there are only two requests to be sent from the user. One is sent to the server for verifying 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. For the CPU resources, all of the relevant OCR manipulations would be executed on the server side, rather than in the user's device, which means it would not take any extra data consumption from the user.

B. Preliminary Requirement

In order to implement the OCR function, 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 in the test server was necessary. In our experiment, the machine used was a Mac-Book Pro and we used 'pip3 install mitmproxy' to install it.

C. Implementation

The specific implementation included four phases: connect to mitmproxy, interrupt traffic and redirect to the test server, implement OCR functions in the test server, and execute the response from mitmproxy.

Phase 1, Connect to mitmproxy

Both mobile and mitmproxy require to be in a same network (Wi-Fi), to establish the connection between this 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 on the mobile phone is shown in Figure 9 below.

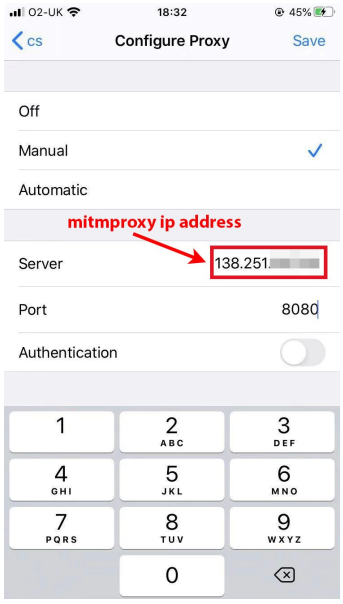

Fig. 9. network configuration on the mobile phone

Phase 2, Interrupt Traffic and Redirect to Server

A python program was written and named 'mitm.py' in mitmproxy. The function 'response' is used to handle the traffic flow. In this proxy, firstly, it retrieves the URL address from the request that is sent by the mobile. Secondly, the proxy redirects this URL address to the browser vendor server to verify its security through the OCR function. In our experiment, we set up the proxy and the server in the same machine; a MacBook Pro 2016, i5 Processor, 15GB Memory. Finally, this proxy would execute different responses according to the detected result from the browser vendor server. This will be explained further in Phase 4.

Phase 3, Implement OCR Functions in the test Server

Once the server (machine) receives the URL request, the 4 step 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. Due to the complexity and diversity of phishing websites, the logo image may be stored in different place, such as in HTML code or in a .css file. When the logo image is in HTML code, it is always inserted under the tag of <img>, as shown in Figure 10. In the other case, a logo image is linked via a .css file, under the attributes of background or background-img, as shown in Figure 11. Therefore, in order to cover all possibilities, both HTML code and .css files need to be traversed. In this step, all of the images will be traversed and stored for recognition.

Fig. 10. Google logo path

Fig. 11. PayPal logo path

2) Recognize image content: In this step, the Google Optical Character Recognition (OCR) API would be called to recognize the content of the extracted image from the last step. This API only works for detecting the specific text from an image, which means the detected result from a redundant image, such as a symbol icon from the .css file, is worthless. All of a logo's content would be stored and moved to the next step. An exception should be noted her, the program will be terminated if there is nothing detected from the images via OCR API. For example, some websites may use a full screenshot from the official website to be a phishing page. In our last paper, we adopted 40 phishing URLs to evaluate the accuracy of this approach. We found only 4 phishing URLs were missed. Two of these phishing websites used a screenshot to fill up the whole page, so there was too much information for the API to process.

3) Search official website URL: In this step, the official website address from the parsed URL would be identified according to the analysed content from the above phase. By using a keyword search in the Google Search API, the related official website URL addresses can be gathered. In our experiment, we only collected the top three results from the returned result list from the search API. Typically, at least one of them is the official URL; the rest of them may be the Wiki link, related news or other branches of this website. Figure 12 shows an example of the obtained results if the search keyword is 'PayPal'.

```
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.'}
```

Fig. 12. search result from the keyword 'PayPal'

4) Verify the security of parsed URL: At the end of 2017, Google announced they will flag all unencrypted traffic in order to make their user feel secure on the internet. Secure Sockets Layer (SSL) is used to improve the security of the traffic transmission based on encrypted technology, which ensures the traffic is private and integral under an established channel between a browser and a server [21]. So far, most websites have already deployed SSL to establish the connection. Thus, we adopt the SSL certificate information to verify the security of the URL. In this step, we retrieve the SSL certificate 'issue by' and 'issue to' attributes to confirm whether there are same. We attempted to use SSL thumbprint to verify the result in the initial experiment, but the result is not ideal. The hash value is inconsistent if the websites are located in different enterprise branches, as shown in Figure 13. Subsequently, the detected result is returned to mitmproxy at the end of this phase.

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

Fig. 13. Google SSL hash comparison between .com and .co.uk

Phase 4, Execute Response in mitmproxy

According to the received result, this proxy would execute different responses. If this URL is verified to belong to a malicious link, this proxy would not redirect this URL request to the real site and return a warning page to the user directly. This is shown in Figure 14. Alternatively, a redirect request to the real site is made and the parsed response is returned to the user.

IV. Evaluation

In this section, the experimental results would be summarised to detect the availability and accuracy of our approach. Subsequently, we conduct the evaluation to examine why this approach is an improvement over current techniques.

A. Experimental Results

In our experiment, 60 URLs were collected from PhishTank, all of these malicious URLs must satisfy the following conditions:

Fig. 14. warning page from mitmproxy

- These malicious websites must be online during the experiment.
- These malicious websites have to include a logo image, and this logo must contain text, otherwise the OCR API would not return the recognition result..
- Their logo must be in English, as this OCR API only works in English so far.

In these 60 URLs, 75% (45/60) were financial-based website, such as PayPal, American Express, etc. In this 75%, the highest proportion was PayPal, at 77% (35/45). Under our solution, 92% (55/60) phishing URLs were identified successfully, only 5 URLs were missed. We further analysed the failure results, the specific reasons were:

- The URL was redirected to other addresses during the web crawler parsing.
- The logo was generated in text rather than in an image.

Subsequently, we examined the effectiveness of this protection against DNS hijacking attacks. We built a fake 'Google' website and connected the mobile DNS to our suspicious DNS server. Any search for 'http://www.google.co.uk' was redirected to our webpage, as shown in Figure 15:

Under a DNS hijacking attack, a hacker must downgrade the protocol from HTTPS to HTTP, otherwise, the browser will warn that the SSL certificate is not valid. During the detection of our approach, the text 'Google' was extracted successfully from the logo and recognized by the OCR API. Also, relevant official websites and their SSL certificate information were indexed correctly, as shown in Figure 16. The accessed URL has been downgraded to an HTTP connection, which means the SSL certificate does not exist or is not accessed. Thus, both 'issued_to' and 'issued_by' are null. Finally, the accessed URL was identified as a phishing site through the comparison of the SSL certificate information.

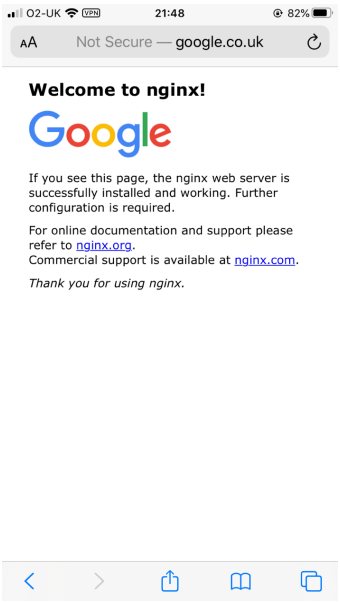

Fig. 15. accessing the fake Google website under a DNS hijacking attack on a mobile

```
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 1O1'}
{'issued_to': 'Google LLC', 'issued_by': 'GTS CA 1O1'}
{'issued_to': 'Google LLC', 'issued_by': 'GTS CA 1O1'}
```

Fig. 16. relevant official website and their SSL certificate information

B. Evaluation

In order to prove the effectiveness of our approach on the mobile platform, we used the same URLs to compare the results of our method with existing mobile browser (Safari on IOS). Only 40% (8/20) malicious URLs resulted in warnings, and the other 60% (12/20) could be accessed without any notifications, as shown in Figure 17. Therefore, in our approach, the security has a significate increase, from 40% (8/20) up to 92% (55/60).

The challenges of mobile implementation were presented in our last paper; such as how this approach points to the user and how the consumption of mobile resources can be minimized, are both eliminated. The requested URL needs to be interrupted in a script on the mobile before forwarding to the real, official site; this URL will be redirected to the browser's server (cloud) and the relevant OCR functions are conducted on the server side. The detected result is returned to the script on the mobile and, according to the result, the script will respond with different pages; either a warning page, or the parsed HTML result from the official site.

However, this prototype mobile implementation still has the same limitations as on desktop. They are [6]:

- The accuracy of detected image content result from OCR.

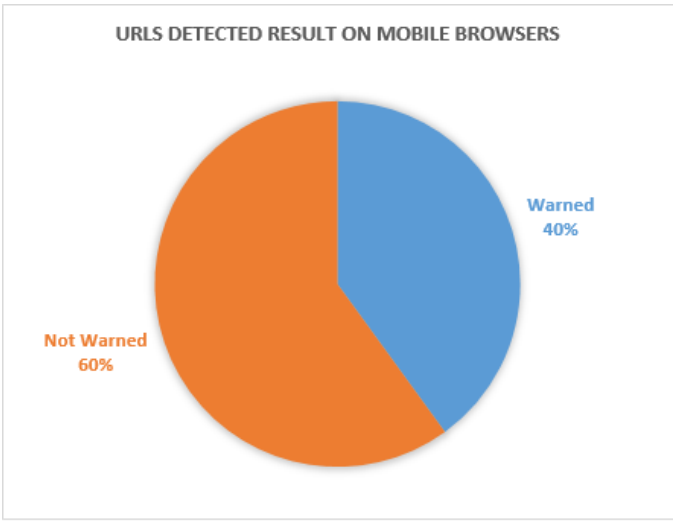

Fig. 17. same Phishing URLs accessing result on existing mobile browser (Safari on IOS)

- The cost of OCR API queries.
- The efficiency of the test server and the cloud API.

V. Conclusion

Phishing attacks are always cheap to produce and easy to deploy. Although many users are sensible about this kind of attack, a phishing attack still has the highest proportion of attacks in a global cyber threat. With the increase in technical approaches, phishing attacks have become more complicated. In particular, in the case of a DNS hijacking attack, the accessed URL will be same as the official address, and the SSL warning can be mitigated. Current solutions cannot keep up with the constant updating of phishing websites. Here we presented a novel approach to identifying phishing websites by using an OCR technique, not only to overcome the limitation on existing solutions, but also to provide a high detection accuracy rate even under a DNS hijacking attack.

In this paper, we 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 official site. This traffic will then be redirected to the remote browser server, and analysed. 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 computation process is executed on a server, and consequently the consumption is no different from normal accessing.

References

[1] APWG, "Phishing Activity Trends Report, 3Q, 2019."
[2] ENISA, "ENISA Threat Landscape Report 2018 — ENISA." [Online]. Available: https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2018. [Accessed: 02-Dec-2019].
[3] ICO, "data-security-incidents-201718." [Online]. Available: https://ico.org.uk/media/action-weve-taken/csvs/2014850/data-security-incidents-csv-201718.xlsx. [Accessed: 02-Dec-2019].

[4] Kaspersky, "What is Spear Phishing? — Definition and Risks — Kaspersky Lab UK," KasperskyLab, 2018. [Online]. Available: https://www.kaspersky.co.uk/resource-center/definitions/spear-phishing. [Accessed: 02-Dec-2019].

[5] Symantec, "Internet Security Threat Report, Volume XIII," 2008.

[6] Y. Wang and I. Duncan, "A novel method to prevent phishing by using OCR technology," in 2019 International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2019, 2019.

[7] imperva.com, "What is a DNS Hijacking — Redirection Attacks Explained — Imperva," 2019. [Online]. Available: https://www.imperva.com/learn/application-security/dns-hijacking-redirection/. [Accessed: 02-Dec-2019].

[8] T. Mocan, "What Is DNS Hijacking? (How to Stop DNS Hijacking) — CactusVPN." [Online]. Available: https://www.cactusvpn.com/beginners-guide-online-security/dns-hijacking/. [Accessed: 02-Dec-2019].

[9] W. Rash, "How to Avoid the New DNS Hijacking Attacks - eWEEK." [Online]. Available: https://www.eweek.com/security/how-to-avoid-the-new-dns-hijacking-attacks. [Accessed: 02-Dec-2019].

[10] CBS, "Mobile phones less often secure than computers." [Online]. Available: https://www.cbs.nl/en-gb/news/2018/38/mobile-phones-less-often-secure-than-computers. [Accessed: 02-Dec-2019].

[11] CHRISTINA BONNINGTON, "In Less Than Two Years, a Smartphone Could Be Your Only Computer — WIRED," Wired, 2015.

[12] B. Jones, "Internet Security: Is Your Smartphone Safer Than Your PC? - PSafe Blog." [Online]. Available: https://www.psafe.com/en/blog/internet-security-smartphone-safer-pc/. [Accessed: 02-Dec-2019].

[13] L. Wu, X. Du, and J. Wu, "Effective Defense Schemes for Phishing Attacks on Mobile Computing Platforms," IEEE Trans. Veh. Technol., vol. 65, no. 8, pp. 6678–6691, Aug. 2016.

[14] T. Klintic and V. Clincy, "Mobile Phishing Attacks and Mitigation Techniques," J. Inf. Secur., vol. 6, pp. 206–212, 2015.

[15] Y. Niu, F. Hsu, and H. Chen, "iPhish: Phishing Vulnerabilities on Consumer Electronics.".

[16] N. Mazher, I. Ashraf, and A. Altaf, "Which web browser work best for detecting phishing," in ICICT 2013 - Proceedings of the 2013 5th International Conference on Information and Communication Technologies: Using Technology to Create a Better World, 2013.

[17] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, "Detecting malicious URLs using lexical analysis," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016, vol. 9955 LNCS, pp. 467–482.

[18] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. 2009.

[19] 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."

[20] BBC, "GDPR 'risks making it harder to catch hackers' - BBC News." [Online]. Available: https://www.bbc.co.uk/news/technology-44290019. [Accessed: 02-Dec-2019]..

[21] Kavya, "Why Google is Forcing You To Have SSL Certificate on Your Websites." [Online]. Available: https://serverguy.com/ssl/google-forcing-ssl-certificate-websites/. [Accessed: 02-Dec-2019].