Securing Websites Against Homograph Attacks

Jemal Abawajy[1(✉)], A. Richard[2], and Zaher Al Aghbari[2]

[1] Faculty of Science and Technology, School of Information Technology, Deakin University, Melbourne, Australia
jemal@deakin.edu.au
[2] Department of Computer Science, College of Sciences, University of Sharjah, Sharjah, UAE
zaher@sharjah.ac.ae

Abstract. With the globalisation of the Internet, standard frameworks such as the Internationalized Domain Name (IDN) that enable everyone to code a domain name in their native language or script has emerged. While IDN enabled coding the domain names in different languages, it has also put users of web browsers that support IDNs at risk of homograph attacks. As IDN-based homograph attacks have recently become a significant threat in content-based attacks such as phishing and other fraudulent attacks against Internet users, an approach that could automatically thwart such attacks against web browsers is important to the Internet users. To this end, we propose a new approach to mitigate the Internationalised Domain Name homograph attacks in this paper. The proposed approach is very easy to deploy in the existing browsers and requires no change in the way the end-user interact with the web-browsers. We implemented the proposed approach as an add-on to a popular web-browser and demonstrate its effectiveness against the homograph attack. Our assessment of the proposed implementation shows that the proposed solution to the IDN-based homograph attack protects web browsers with no noticeable overhead.

Keywords: Internationalized Domain Name · Homograph attacks
Phishing attacks · Unicode attack · Homograph obfuscation
Web browsers security

1 Introduction

The development of IDN marks the departure from the English-centric web service to a web service globalization to meet the needs of the potential users' worldwide. An IDN is an Internet domain name that allows Internet users to create and use websites in many different languages. By globalizing offering of their services, companies such as PayPal Holdings that operate businesses worldwide stand to benefit from internationalized usability of the Web through the potential increase of customer base. The use of IDN is a trend that will only increase in both public and private organizations worldwide. However, with the introduction of IDNs came a slew of new security concerns, chief among them being the homograph attack [10]. Although the homograph attack is not new, it has recently resurfaced with the introduction of IDN [1] and

X. Lin et al. (Eds.): SecureComm 2017 Workshops, LNICST 239, pp. 47–59, 2018.
https://doi.org/10.1007/978-3-319-78816-6_4

has increasingly become one of the serious Web security problems [2]. With the availability and increased usage of Unicode-based web documents and non-ASCII codes in the domain name, Unicode-based homograph attacks are expected to be a severe web security problem [1, 5]. In particular, the classic scams involving identify theft, fraud, and corruption are anticipated to increase both in number and complexity.

With the increasing internationalization of the Internet, it is more important than ever to provide automated protection against IDN-based homograph attacks for the stability of the Internet. Generally, existing solutions place the burden on the end-users by requiring them to be vigilant about the attack. For example, the Unicode Consortium has been active at raising awareness of the Unicode-based homograph attacks and in providing recommended solutions [13]. While it is true that the users should be aware about the threat of homograph attack, unfortunately we cannot expect them to be vigilant all the time whether Uniform Resource Locator (URL) is legitimate or spoofed via a homograph attack. Existing techniques do not actively attempt to determine whether or not the IDN is a homograph attack. This is left to the end user to figure out which one is genuine and which one is fake. In this paper, we propose a new IDN-based homograph attack mitigation technique. The proposed technique takes the address and determines whether or not it is a spoof of another site. Because it is able to distinguish between legitimate and spoofing sites it allows the user to visit all IDNs safely without being restricted. In addition to this, the strategy proposed here alerts the user with a warning message when they visit a homograph site, which is something the currently implemented mitigation techniques do not do. In summary, this paper makes the following contributions:

- Evaluation of the existing web browsers defense mechanisms against IDN-based homograph attacks;
- A new effective mitigation technique that detects IDN-based homograph attacks and properly notify the users; and
- Implementation and evaluation of the proposed IDN-based homograph attack mitigation technique.

The reset of the paper is organized as follows. In Sect. 2, a brief background on IDN and Unicode as well as how phishers exploit them via homograph attacks to gain sensitive information from unwary users is described. In Sect. 3, the current defenses in place to stop these Unicode-based homograph attacks are discussed. In Sect. 4, a new Unicode-based attack mitigation technique is described. The implementation of the proposed Unicode-based attack mitigation technique into a Google Chrome and performance analysis are described in Sect. 4. The conclusion is given in Sect. 5.

2 Background

In this section, a brief background on IDN and Unicode as well as how phishers exploit them via homograph attacks to gain sensitive information from unwary users is described.

2.1 Internationalized Domain Name

IDN system uses Unicode as opposed to standard English ASCII characters. Unicode is a computer industry standard code that assigns unique numbers to languages in active use today. This allows using different scripts and languages in software, predominantly for applications in web links, web pages, and emails [4]. Unicode was created to replace the American Standard Code for Information Interchange (ASCII). Due to the globalization of Internet and wide penetration of information technology worldwide, ASCII become no longer sufficient. Unicode changed that by including all the characters from every writing system in the world, both current and ancient, as well as symbols and punctuation. Currently, Unicode has over 100,000 characters. Each character in Unicode has a unique number, regardless of the platform or languages, making the different languages and scripts compatible for information interchange. Because IDN is implemented on the application level, no changes are needed to the Domain Name System (DNS) protocol. All of the work is done at the application level by the browser. However, Unicode character set contain visually and semantically confusable characters, which can lead to a myriad of security risks. Specifically, it can aid in creating a bogus domain name chief of which is IDN-based homograph attack.

2.2 IDN-Based Homograph Attack

Homograph attack [9], also called Unicode attacks, visual spoofing, and homograph obfuscation [2, 4], is the type of spoofing attack where a fake website domain name that deceptively looks like a genuine one is created by substituting one or more similar but different characters in the legitimate website domain names. An example is the ‘microsoft.com’ and ‘microsoft.com’. In the second domain name, the second ‘o’ character is replaced by the Greek omicron ‘o’ character. The two URLs look identical. While it is impossible for a user to tell them apart visually, the computer will treat them as two totally different characters. This makes homograph popular targets for malicious users to use for phishing attacks and risky for regular users.

IDN-based homograph attack is an example of web browser security risks originating from the exploitation of the Unicode character sets to create a fake IDN. Unicode-based homograph attack is usually made possible by substituting one or more characters of the legitimate website address with their equivalent homographs in the Unicode character set. Many of these characters look similar if not identical which can lead to new and hard to detect phishing attacks. These attacks come in a variety of forms, such as mixed and whole script attacks, as well as character and word level similarity attacks.

Given 100,000 characters (many of which visually indistinguishable) contained in the Unicode, the potential for creating phony URLs that can fool even the most security savvy users is great. This attack vector can enable crafting of counterfeit websites commonly used by phishers to maliciously target unwary users in an attempt to deceive them into handing over their sensitive information such as their credit card details and causing billions of dollars of damage [6]. Moreover, these Unicode-based homograph attacks are potentially more dangerous than regular phishing attacks as they are harder to detect visually for the user. As the popularity of IDNs rises, so does the possibility of

these attacks. As the Web usage extends beyond the sole Latin script, this type of Web security risk is expected to increase significantly [1].

2.3 Problem Overview

The basic tenet of the attacks is to make the unaware target of the attack believe that she is using a legitimate Web site when in fact she is accessing a fake Web site that deceptively looks like a genuine one. This is attained by making the fake Web site address virtually indistinguishable from the real URL visually by using different characters from various alphabets such as Cyrillic and the Greek as well as ASCII alphanumeric characters (e.g., Zero stands for the letter "o"). Since the Unicode supports around 100,000 characters, the attackers can exploit resembling characters in various combinations.

We formally define the problem of homograph attacks detection as follows: *Let* $\mathcal{W} = \{w_1, w_2, \cdots, w_n\}$ *be a set of n legitimate Web sites in a DNS database such that each Web site* $w_i \in \mathcal{W}$ *has a domain name,* $DN(w) = \{c_1 c_2 \cdots c_L\}$*, of L characters derived from the standard ASCII code. Let* w *be a Web site with a domain name* $DN(w) = \{c_1 c_2 \cdots c_L\}$ *such that at least m characters of the domain name are from* Unicode *(e.g., Cyrillic, Greek, Latin or the mix of these scripts) such that* $1 \leq m \leq L$*. We want to know if* w *is a legitimate or a fake Web site.*

A wide variety of approaches have been proposed to address the homograph attack. Existing approaches can be generally classified as algorithmic analysis of the characters in the URL or user-oriented security approaches. Our approach combines both algorithmic analysis of the characters and user-oriented security approaches. A suite of user-oriented security approaches that aim to draw the attention of non-ASCII Web sites browsers is discussed in [1]. These approaches include visual security indicators such as enlarging font sizes and highlighting confusing letters. Although these approaches can provide users with visual clues about the possible threats, they cannot prevent the IDN-based homograph attack. An approach based on Unicode string coloring in which each language/script is displayed uniquely in color is discussed in [4]. A recent study on domain name highlighting effectiveness concluded that it is unreliable as the lone mitigation technique [7]. Moreover, they can render genuine Web sites with mixed scripts to appear suspicious to users and useless with color blind users [9]. Punycode by Unicode Consortium [14] is the common strategy used by the existing web browsers. The idea is to convert non-ASCII characters in IDN into basic ASCII characters (a–z, 0–9). When an address is converted to Punycode, a special marker 'xn--'is used to denote that the address is in Punycode format. For example, when the IDN 'paypăl.com' with ă from the Cyrillic set is encountered, the web browser will convert the address into Punycode format as 'http://xn--paypl-tof.com/'.

Existing web browsers seem to tackle the subject of IDNs differently. Google Chrome and Internet Explorer will convert the address to Punycode if it contains mixed scripts. This may rule out legitimate mixed-script addresses, and miss any single script homograph attacks. Opera and Firefox use a whitelist of top level domains, and converts any address not in these domains to Punycode. While this helps to ensure that most of the addresses not converted to Punycode are legitimate, many legitimate sites may be converted to Punycode just because they don't belong to certain top level

domains. Safari renders only problematic character sets, such as Greek and Cyrillic, as Punycode. This can lead to every legitimate Greek and Cyrillic IDN being converted to Punycode, greatly restricting the sites the user navigates to.

As user-oriented security approaches require more attentiveness from humans [6], there is a necessity for an automated analysis of the URL to prevent IDN-based mitigation attacks. An approach that extracts and verifies different terms of a URL using search engine spelling recommendation for automated phishing web site detection is discussed in [3]. A Bayesian-based approach that determines if a Unicode character in a word is be o detect whether a suspicious Unicode character in a word is visual spoofing or not is discussed in [5]. Helfrich and Dual [10] described an approach called a dual canonicalization for detecting if two encodings are homographs. The idea is based on homograph sets and deciding if two encodings are either belong to the same homograph set, or else that they belong to different homograph sets.

In spite of intense research to mitigate the problem, the IDN-based homograph attacks still occur. A recent case that highlights the homograph attack problem is the fake 'IIoydsbank.co.uk' domain complete with a high level of HTTPS and a valid TLS certificate [12]. The forged domain names easily fooled many users into trusting it as the legitimate banking website. This example shows that even a TLS certificate is just as easy to obtain a valid certificate for the forged website. Although there is a significant potential for abusing homographs, it is simply wrong to assume that all homographs are malicious or spoofing as it is commonly done in the web browsers today [5].

Moreover, the currently implemented defenses to fight against these homograph attacks are subpar. There are three main problems with the current techniques. First, the browser will still always display the page, whether the address has been converted to Punycode or not, it relies solely on the user noticing that the address is in Punycode and understanding what that means. The second problem is that none of the existing approaches gave the user a proper notification warning them of the potentially dangerous site they were about to visit. Third, none of these techniques can be sure that the site is actually a homograph attack, which can restrict the user from visiting many legitimate sites. The proposed approach aims to remedy these issues. It will achieve this by taking the site address and checking to see whether or not it is a spoof of another site. This should be an improvement over current mitigation techniques as it will actually determine between legitimate and illegitimate sites and properly notify the user when they are trying to navigate to a spoofed site. Although the existing approaches have proven successful to some degree in detecting homographs, they are not generic and not comprehensive enough.

3 Homograph Attack Mitigation Strategy

In this section, the proposed IDN-based homograph attack mitigation strategy is described. We will first discuss the proposed algorithm and then provide security analysis of the algorithm.

3.1 Homograph Attack Detection

Algorithm 1 shows the pseudo-code of the proposed strategy. The algorithm maintains a set of homograph characters $H(c)$ for each ASCII character c. For example, character 'a' will have $H(a)$ = {α, ά, ä, ă, Ά, …, Ă}. The domain address (URL) is the input to the algorithm and the output is whether or not the URL is spoofed.

Algorithm 1: Homograph Attack Detection (HAD)

```
INPUT: URL
OUTPUT: Warning message
BEGIN
     IF (Punycode (URL) == TRUE) THEN
         URL← Unicode (URL)    //Unicode version URL
     ENDIF
     IF (Unicode (URL) == TRUE) THEN
        URL← ASCII (URL)     //ASCII version URL
     ENDIF
     IF (ASCII (URL) == TRUE) THEN
         M←Lookup (URL)
         IF (|M| ≥ NULL) THEN
             Message (Warning)
         ENDIF
     ENDIF
```

END Algorithm 1

The algorithm first checks the type of the URL and take the appropriate action. Specifically, if the URL address is in Punycode, the URL is converted to its corresponding Unicode type. If the URL is in Unicode format, it is converted to ASCII code. The idea is to take the non-ASCII domain address and converted to the ASCII code. In order to convert the domain address to ASCII code, the algorithm checks for visually similar characters in the domain address and convert all Unicode characters to ASCII code. For example if the character 'ă' was found in the IDN, it would be removed and replaced with the character 'a'. This step would result with the original address such as www.paypăl.com being converted to the regular www.paypal.com.

The final step is to take the address generated in the second step and cross-check it in the DNS database (e.g., Google Public DNS, DNS Advantage and Norton Free DNS) to ensure its validity. The idea is that if the site is found to exist, then obviously the original IDN address (e.g., www.paypăl.com) is a homograph attack. In this case, the user is notified with a very clear and concise message that makes the danger obvious to the user. Figure 1 illustrates the proposed homographic attack mitigation technique using "www.paypăl.com". This strategy, unlike many of the current defenses in place to mitigate homograph attacks, is designed to actively attempt to determine if an IDN is a homograph attack or not. Moreover, it can detect single, mixed and whole script attacks.

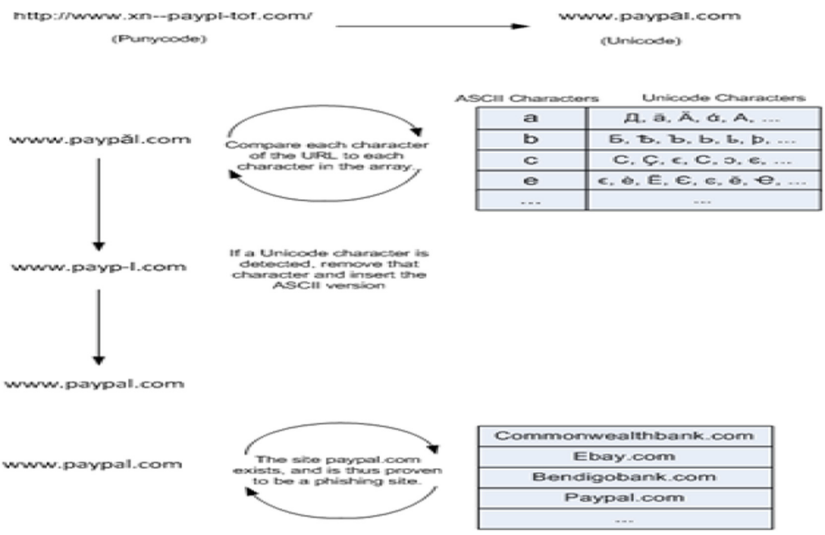

Fig. 1. Illustration of the proposed mitigation strategy.

3.2 Security Analysis

In this section, we conduct the security analysis and show that the proposed mitigation technique detects a variety of homograph attacks, such as single, mixed and whole script attacks. In the analysis, we assume that there is a set of $\mathcal{W} = \{w_1, w_2, \cdots, w_n\}$ legitimate Web sites with the ASCII domain name registered in DNS database. Also, we assume that an adversary has created a spoofed website $\mathcal{w}$ with a URL that visually appears to be a trusted site when in fact it is a malicious one.

Let $S = \{c_1 c_2 \ldots c_L\}$ be a sequence of L characters representing the domain name of a Web site $\mathcal{w}$. Suppose that $\mathcal{w}$ is a phishing site impersonating a legitimate Web site $w_k \in \mathcal{W} | 1 \leq k \leq n$ in the DNS. We now show that the proposed algorithm will detect $\mathcal{w}$ as a phishing site.

Lemma 1: The proposed algorithm detects IDN-based homograph attack.

Proof: Suppose that the domain name of $\mathcal{w}$ contains a set of $s \subseteq S$ non-ASCII characters such that $1 \leq |s| \leq |S|$. Note that when $s = 1$, it is a single character homograph attack. In contrast, the attack is said to be a mixed homograph attack when $1 < |s| < |S|$ and s contains characters from a variety of scripts such as Greek and Latin. The whole script homograph attacks occur when $|s| = |S|$. The proposed algorithm converts the domain name of $\mathcal{w}$, regardless of the type of the domain name (i.e., Punycode or Unicode) and types of the attack (i.e., single, mixed or whole), into the corresponding ASCII code. The lookup function, using the ASCII domain name of $\mathcal{w}$, will return NULL since it does not exist in the DNS thus proving that $\mathcal{w}$ is a phishing site. ■

Lemma 2: The proposed algorithm detects ASCII-based homograph attacks.

Proof: Suppose the string S contains a set of $s \subseteq S$ visually similar ASCII characters as a Web site $w_k \in \mathcal{W}$ such that $1 \leq |s| \leq |S|$. An example of such case is 'w_k = Lloydsbank.com.uk' and '$\mathcal{w}$ = IIoydsbank. co. uk' where the first letter 'L' in the legitimate Web site is replaced with a capital 'i' letter 'I'. For $\mathcal{w}$ to be flagged as a phishing site, the following must hold:

$$\mathcal{M} \neq NULL \tag{1}$$

For this case, the algorithm produces $\mathcal{M} = NULL$, which indicates that $\mathcal{w}$ is not in the DNS database. Hence, the algorithm detects that $\mathcal{w}$ is a phishing site. ■

4 Implementation and Testing

In this section, we will test both the defenses currently implemented in a number of web browsers and the mitigation technique proposed in this paper. The proposed homograph attack mitigation strategy is written in Javascript and implemented into the Google Chrome browser as an add-on. The results will be compared in an attempt to analyze the effectiveness of the proposed mitigation technique.

4.1 Methods

A number of tests were devised to test how the current mitigation techniques implemented in browsers respond to IDN-based homograph attack. These tests will cover a range of different kinds of homograph attacks such as mixed-script and whole-script spoofing, and should act to test whether or not the proposed approach can successfully detect them as homograph Web sites. A legitimate address will also be used to see if this add-on can, unlike current mitigation techniques, determine if a mixed-script IDN is legitimate. Table 1 shows the test sites used in the experiments.

The **google.com** is a spoof of 'google.com', where the second 'g' has been replaced with a Latin small letter 'g' (U+0261). This is a single-script homograph attack. It uses only Latin characters to spoof the google.com site. This test will act to show how the current IDN homograph mitigation techniques in browsers treat IDNs that use an extended Latin script. The **paypăl.com** is a spoof of 'paypal.com', where the second 'a' has been replaced by the Cyrillic 'ă' (U+04D1). This is an example of a mixed-script homograph attack, as it uses both Latin and Cyrillic characters. This test will act to determine how the current mitigation techniques in browsers treat IDN homograph

Table 1. Test sites used in the experiments.

Site	Unicode	Punycode
goo**g**le.com	0067 006F 006F **0261** 006C 0065 002E 0063 006F 006D	xn--goole-tmc.com
payp**ă**l.com	0070 0061 0079 0070 **04D1** 006C 002E 0063 006F 006D	xn--paypl-tof.com
βeta.com	**03B2** 0065 0074 0061 002E 0063 006F 006D	xn--eta-rxc.com
ҽъау.com	**04BD 044A 0430 0443** 002E 0063 006F 006D	xn--80a2bt37b.com

attacks that use mixed scripts. The **βeta.com** is not a spoof as there is no beta.com. This is to represent the possibility of a legitimate site that uses mixed scripts. βeta.com uses the Greek 'β' (U+03B2). This test act to establish how the current mitigation techniques in browsers treat mixed-script IDNs that are legitimate and not homograph attacks. The **ҽъay.com** is a spoof of 'ebay.com', where the 'e' has been replaced by the Cyrillic 'ҽ' (U+04BD), the 'b' has been replaced by the Cyrillic 'ъ' (U+044A), the 'a' has been replaced by the Cyrillic 'a' (U+0430), and the 'y' has been replaced by the Cyrillic 'y' (U+0443). This is an example of a whole-script spoof, where every Latin character has been replaced by a visually similar Cyrillic character. This test will act to determine how current mitigation techniques in browsers treat whole-script homograph attacks.

We tested the Internet Explorer, Google Chrome, Mozilla Firefox, Opera, Safari and Avant. No changes were made to the browsers, they were be used with the default settings. The results of these tests will mainly be looking at whether or not the address was converted to Punycode and whether or not the browser alerted the user that they were visiting a homograph site. We used the following aspects to test each browsers:

- Converts Address to Punycode (#1)
- Makes a visual distinction (#2)
- Notifies the user that the website may be illegitimate (#3).

4.2 Results and Discussions

Table 2 shows the results of the experiments. Google Chrome converts all the addresses to Punycode, even the site βeta.com which isn't a spoofing site. It is impossible to tell legitimate IDNs from fake IDNs. Mixed IDNs are always shown as Punycode, even if they are legitimate sites. The browser makes no attempt to notify the user that the site may be unsafe. It converts the IDN based on the languages that the user has listed. By default, English is the only language listed, and the user can add more. If at least one letter in the IDN is not in the user's languages the IDN is not

Table 2. Results of the experiments.

Browser	goo**g**le.com			payp**ă**l.com			**β**eta.com			**ҽъay**.com		
	1	2	3	1	2	3	1	2	3	1	2	3
Chrome	√			√			√			√		
Explorer	√		√	√		√	√		√	√		√
Firefox	√			√			√			√		
Avant			√			√			√			√
Safari	√			√			√			√		
Opera	√			√			√			√		
Ours	√	√	√	√	√	√	√	√	√	√	√	√

displayed. This is quite restrictive and may result in many legitimate IDNs being converted to Punycode and becoming unreadable to humans.

Internet Explorer converts all the addresses to a Punycode, even the site βeta.com which isn't a spoofing site. This makes it impossible to tell legitimate IDNs from fake IDNs. Mixed IDNs are always shown as Punycode, even if they are legitimate sites. Explorer notifies the user that the web address contains letters that are from a different language. It converts the IDN based on the languages that the user has listed. By default English is the only language listed, and the user can add more. If at least one letter in the IDN is not in the user's languages the IDN is not displayed. In addition to this, Internet Explorer notifies the user that the web address contains letters or symbols that cannot be displayed with the current language settings, and allows the user to change these settings. This is restrictive and may result in many legitimate IDNs being converted to Punycode and unreadable to humans. Mixed IDNs are always shown as Punycode, even if they are legitimate sites. It makes no attempt to notify the user that the site may be unsafe.

Mozilla Firefox loads the pages but converts addresses to Punycode, regardless of what Unicode character set or if they are mixed or single script, unless they meet the whitelist standards. A whitelist of top level domains is used, where the registrars must take care to not allow any homograph-confusable International Domain Names to be registered. Any IDNs that are not from one of these top level domains, even if they are legitimate, will be displayed in Punycode and are therefore not human readable. In addition to the standard tests, a test was done using a top level domain name other than ".com". The address paypăl.info is displayed and treated as if it was legitimate, despite it having a mixed script. This leaves Firefox users open to homograph attacks.

In the Avant browser, none of the sites were converted to Punycode, whether they were legitimate or spoof sites. The browser notifies the user that the web address contains letters or symbols that are not from the users preferred languages, but despite this it displays the site and displays the IDN in Unicode. Clicking on the notification allows the user to add languages to their preferred language list. Avant loads the page and does not convert the address to Punycode. There is no distinction between the real address and fake address. While this may lead to the user visiting phishing sites, they are also able to go to all legitimate IDNs without being restricted. This leaves Avant users open to being the victims of homograph attacks. Mixed IDNs are always shown as Punycode, even if they legitimate sites. No attempts are made to alert the user that the site may be unsafe. Safari uses a list of allowed universal character sets, which by default includes all scripts except Cherokee, Cyrillic and Greek, because these 3 scripts contain characters that are visually similar to many characters in the Latin script. While this can result in many possible homograph sites being ruled out, legitimate IDNs may also be converted to Punycode and thus become unreadable for the user.

Opera converts all mixed script sites to Punycode, even the site βeta.com which isn't a spoofing site. This results in it being impossible to tell legitimate mixed-script IDNs from fake mixed-script IDNs. The browser makes no attempt to notify the user that the site may be unsafe. It loads the pages but converts addresses to Punycode, regardless of what Unicode character set or if they are mixed or single script, unless they meet the whitelist standards. Opera uses a whitelist of top level domains, where

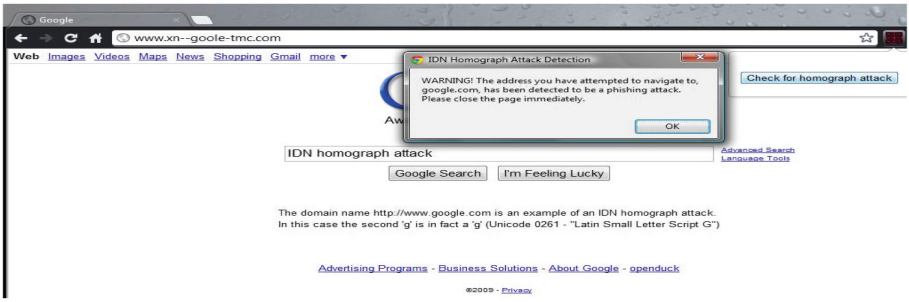

Fig. 2. Test google.com using the proposed approach.

the registrars must take care to not allow any homograph-confusable International Domain Names to be registered. It is important to note that IDNs that use Latin 1 characters, those being mostly Western European languages, are accepted and displayed by Opera. Of great concern is the fact that the whole-script spoofing site ebay.com is treated as legitimate by the Opera browser. This leaves Opera users open to the possibility of being the victims of homograph attacks. This is particularly concerning given how popular Opera is on mobile devices (Figs. 3, 4 and 5).

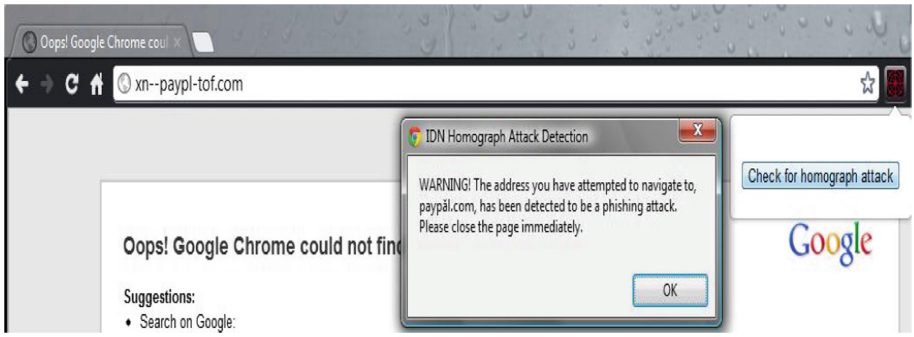

Fig. 3. Test paypăl.com using the proposed approach

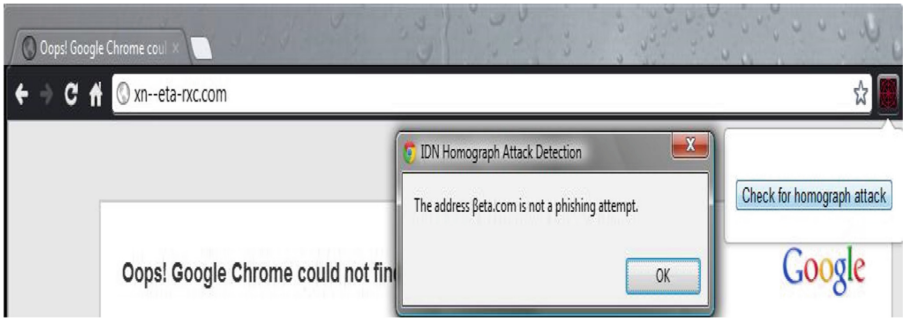

Fig. 4. Test βeta.com using the proposed approach.

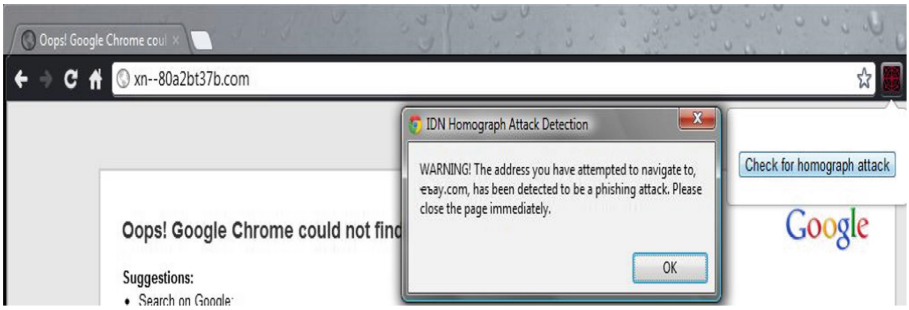

Fig. 5. Test ebay.com using the proposed approach

The proposed approach is implemented in Google Chrome as add-on. It was able to properly detect that the two illegitimate mixed-script sites, google.com and paypăl.com, were homograph attacks. It then warns the user to navigate away from the site as shown in Figs. 1 and 2. The add-on was also able to properly ascertain that the site βeta.com was legitimate despite the fact that it consists of mixed scripts, unlike every other browser. It accomplished this by checking to see if there was a beta.com that the site βeta.com could possibly be spoofing. Since this was not the case it proved that the site βeta.com was legitimate. The add-on properly detected the whole-script spoofing site ebay.com as a homograph attack. This is something the Opera and Avant browser were unable to do. Overall the add-on proved effective, successfully determining which of the test sites were homograph attacks and which were legitimate. None of the current homograph attack mitigation techniques enabled in browsers were able to do this. Finally, unlike the other browsers, this add-on was able to properly inform the user that they had navigated to a phishing site and should immediately leave.

5 Conclusion

The introduction of IDN has enabled everyone to code a domain address in their native vernacular. At the same time, IDNs make it easier for criminals to impersonate or spoof web sites by mixing different scripts leading to IDN homograph attacks. In this paper, security risks to various Web due to non-ASCII domain names have been outlined. Generally, existing approaches expect the end users to be more aware of possible threats and proactively inform themselves not falling for the attacks. As security solutions currently in place to mitigate IDN homograph attacks are inadequate, an approach that automatically thwarts homograph attacks is proposed in this paper. The proposed IDN homograph attack mitigation strategy is implemented into the Google Chrome browser as an add-on. The effectiveness of the proposed approach was verified through tests that include single-script, mixed-script and whole-script spoofs, as well as an example of a legitimate mixed script address. Thus, this paper makes significant contribution towards secure browsing experiences for end users.

Acknowledgment. The authors wish to thank Maliha Omar. Without her help, this paper will not be possible to be completed. The authors would also like to extend their appreciation to Deakin University for partially funding this project.

References

1. Al Helou, J., Tilley, S.: Multilingual web sites: internationalized domain name homograph attacks. In: 12th IEEE International Symposium on Web Systems Evolution (WSE), pp. 89–92 (2010)
2. Roshanbin, N., Miller, J.: Finding homoglyphs - a step towards detecting unicode-based visual spoofing attacks. In: Bouguettaya, A., Hauswirth, M., Liu, L. (eds.) WISE 2011. LNCS, vol. 6997, pp. 1–14. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24434-6_1
3. Maurer, M.-E., Höfer, L.: Sophisticated phishers make more spelling mistakes: using url similarity against phishing. In: Xiang, Y., Lopez, J., Kuo, C.-C.J., Zhou, W. (eds.) CSS 2012. LNCS, vol. 7672, pp. 414–426. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35362-8_31
4. Wenyin, L., Fu, A.Y., Deng, X.: Exposing homograph obfuscation intentions by coloring unicode strings. In: Zhang, Y., Yu, G., Bertino, E., Xu, G. (eds.) APWeb 2008. LNCS, vol. 4976, pp. 275–286. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78849-2_29
5. Qiu, B., Fang, N., Wenyin, L.: Detect visual spoofing in unicode-based text. In: 20th International Conference on Pattern Recognition (ICPR), pp. 1949–1952 (2010)
6. Abawajy, J.: User preference of cyber security awareness delivery methods. J. Behav. Inf. Technol. **33**(3), 236–247 (2014)
7. Lin, E., Greenberg, S., Trotter, E., Ma, D., Aycock, J.: Does domain highlighting help people identify phishing sites? In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2075–2084 (2011)
8. Canova, G., Volkamer, M., Bergmann, C., Borza, R., Reinheimer, B., Stockhardt, S., Tenberg, R.: Learn to spot phishing URLs with the android nophish app. In: Bishop, M., Miloslavskaya, N., Theocharidou, M. (eds.) WISE 2015. IAICT, vol. 453, pp. 87–100. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18500-2_8
9. Helfrich, J.N., Neff, R.: Dual canonicalization: an answer to the homograph attack, eCrime Researchers Summit (2012)
10. Baasanjav, U.B.: Linguistic diversity on the internet: Arabic, Chinese and Cyrillic script top-level domain names. Telecommun. Policy **38**(11), 961–969 (2014)
11. Hamid, I.R.A., Abawajy, J.H.: An approach for profiling phishing activities. Comput. Secur. **45**, 27–41 (2014)
12. Cluley, G.: Lloydsbank, IIoydsbank - researcher highlights the homographic phishing problem, 29 June 2015
13. Davis, M., Suignard, M.: Unicode security considerations, Unicode Technical Report #36 (2014). http://unicode.org/reports/tr36/. Accessed 10 Aug 2015