Large Scale Detection of IDN Domain Name Masquerading

Yahia Elsayed and Ahmed Shosha
Nile University
Cairo, Egypt
y.kandil@nu.edu.eg, ashosha@nu.edu.eg

***Abstract*—Introducing Unicode characters to domain names enabled end users to register a domain name in different languages, i.e., Russian, Arabic or Chinese. This process is defined as Internationalized Domain Names (IDN). The Unicode standard contains a large number of characters and character sets. Some of those Unicode characters' sets may resemble some ASCII characters (this is commonly referred as "homoglyph") which are the basic building blocks for a domain name address. As such, an attacker could use the concept of homoglyph to spoof a domain name and lure an innocent user to visit a decoy domain instead of a legitimate one. IDN domain spoofing could be best detected at the end user side or by using a centralized monitoring solution. This research work is focusing on the different IDN spoofing attack types, and it proposes a new centralized monitoring system that can detect those attacks.**

***Index Terms*—Homoglyph, homograph, domain names, DNS, security, IDNA, spoofing, phishing**

I. Introduction

In the early 90s when the internet was connecting the world, having internationalized applications and content on the internet with many languages' support was a major demand [1]. That's why several encoding techniques were introduced. The Unicode standard was one of the encoding techniques that came with a very significant advance over all other encoding methods. For the first time, all characters can be represented in a uniform manner, making it feasible for the vast majority of programs to be globalized. The Unicode is built to handle all human languages [2].

Unicode contains several character sets and incorporates the varied writing systems. However, the larger repertoire of characters in different sets (languages) resulted in a possible scope for visual spoofing. The similarity in visual appearance may lure a user to perform unsafe actions, such as accessing a malicious domain [3], [4]. Although the visual character spoofing exists in the standard ASCII[1] (for example, 0 "zero" may be similar to "O" in uppercase). Introducing the Unicode increased the opportunities for visual spoofing; which in turn facilitated social engineering and phishing attacks [5], [6].

Some character sets that are supported by Unicode contains letters that are very similar to the English alphabet, and it may confuse the end user. Those characters are called Homoglyph [7]. A Homoglyph is a figure which visually looks like another different figure in a different language - character set. Each homoglyph has a different Unicode code point. For example, the Cyrillic small letter a ("a") has a code point of U+0430, and it is visually identical to the Latin small letter a ("a") with a code point of U+0061 which exits in the English alphabet [8].

With the support of Unicode, and the people's desire to register their domain names in their languages and writing systems; The Internationalized Domain Name (IDN) was introduced as a mechanism for creating Domain Names in different languages, like Chinese, Arabic, and Russian [1], [9], [10]. Since IDN is based on the Unicode encoding, the Internationalization in Domain Names inherited most of the security risks existing in the Unicode including the visual spoofing issues [4], [7], [8].

DNS was deployed in the early days of the Internet, and it is a cornerstone of the Internet infrastructure. Changing the way of how DNS works by adding new character sets and language scripts was not the optimal way to support IDN. The most effective and efficient technique was by introducing the Internationalized Domain Names in Application (IDNA) without requiring any changes from the Internet infrastructure [9], [10]. As such, Domain Names can be stored in the DNS in the old legacy ASCII format (commonly called Punycode), while it will be represented in the application with the mapped IDN Unicode form. This process is performed using the conversion algorithms "ToAscii" and "ToUnicode" [11].

Fortunately, the design of IDNA2003 and the enhanced version IDNA2008 prevented a significant number of visual spoofing attacks [9], [10]. Those RFCs enforce the Internet applications like web browsers and mail clients to perform normalization, case folding and security checks on any IDN label before processing it in order to reduce the attack surface [12], [13]. However, visual spoofing can still present with many IDN domain names; on the other hand, most of the internet applications failed to comply with the cited RFCs; as stated in section III-A. Hence, attackers are still able to visually spoof a legit domain name.

This paper is structured as follows: in section II, we present in detail different IDN spoofing attack techniques that can be used to lure a user, followed by section III in which we list some existing mitigation techniques for the end user and the registrar. In section IV, we present the proposed system along

[1]http://www.asciitable.com

978-1-5386-4922-0/18/$31.00 ©2018 IEEE

with its internal modules and used algorithms in each module. Finally, in section V, we present the system evaluation on a large dataset by monitoring the newly registered domain names looking for IDN domain spoofing of social media domain names and Majestic top 100K.

The main contributions of this paper are: (1) realizing a working definition of IDN spoofing attacks, (2) how those IDN domains are being presented in the URL bar in some Internet browsers, (3) proposing a working solution that reports IDN spoofing attacks, (4) evaluating the solution by reporting the attacks against social media domain names and Majestic top 100K, and (5) analyzing the reported IDN domains that masquerade the social media domain names.

II. Visual Spoofing Based Attacks

The ultimate aim of domain name masquerading is to lure a user to think that he or she is accessing a legitimate website while they are accessing a decoy domain. This attack can be achieved by creating a domain name that is a clone of a real/legitimate domain address in such a way that the user may not notice the difference [3], [5], [14]–[16]. Domain name masquerading solely relies on user confusion. In this section, we list some of the domain name spoofing attacks and a summary for each attack type.

A. Domain Name Similarity Attacks

An attacker might try to register a slightly different domain name than the legitimate one by altering single or multiple characters (e.g., 'Gooogle.com' instead of 'Google.com'). A subset of this attack type is commonly called typosquatting attack which targets the Internet users who incorrectly type a website address into their web browser. As a result, the user would be lured to visit this counterfeit website when they make such a typographical error, or by mailing them the fake URL through a phishing campaign [15], [16].

B. SubDomain Spoofing Attacks

This attack takes advantage of the fact that the subdomain is displayed in the least significant label order. An attacker might attempt to confuse the user by registering a third level domain name that seems similar to the authentic domain name, then by crafting a long URL string that contains the fake domain name; a user may be lured into thinking that he or she is accessing the legitimate domain name.

C. IDN Based Attacks

Using the concept of Internationalized Domain Names, an attacker can register an IDN that looks exactly like the victim's domain name by exploiting the similarities between the characters within one or more character sets (homoglyph characters). This attack relies on the circumstance that some characters' glyphs are defined more than once while belonging to different sets with different code points, despite the fact that one of Unicode's primary goals is unification. For example; the Latin small letter 'o' U+006F can be confused with the Cyrillic small letter 'o' U+043E, the Greek small letter omicron 'o' U+03BF and the Myanmar letter wa 'ဝ' U+101D [14]. [3].

*1) **Single Script Spoofing:*** The fake IDN domain's characters are inherited from the same character set. An attacker will attempt to find a character set that has all needed characters to register the claimed IDN domain. Cyrillic, Latin, and Greek scripts are examples of the language scripts that could achieve such aim. уаһоо.com is an example of single script spoofing using Cyrillic character set [2]. Single script spoofing has advantages on Mixed script spoofing as most of the browsers won't convert the IDN domain name to its Punycode form as described in the section III-A.

*2) **Mixed Script Spoofing:*** In given circumstances, if the attacker was not able to find all the needed homoglyph characters in one character set (single language script) or the suggested domain name had been already registered, the attacker will have to widen the search scope by traversing all available homoglyph characters in different character sets to compose the malicious domain name. For example, the attacker may be able to easily construct 15 different combinations of google.com by replacing the 'O' character with the mentioned homographs. This results in IDN domains with mixed character sets or language scripts (i.e. gоοgle.com uses ASCII, Cyrillic and Greek scripts) [3], [6], [7]

*3) **Punycode Spoofing:*** This attack works on the fact that most of the end user application would convert the IDN domain to its ASCII format - Punycode form - using the ToAscii method [11]. Thus an attacker will attempt to craft a Unicode domain name which has a Punycode form that is similar to a legitimate domain name. For example, 䕮䕵䕶䕱.com will be converted to xn–google.com [14].

*4) **Syntax Spoofing:*** This attack is more dangerous than regular character spoofing. An attacker might attempt to spoof special characters that are used to compose a domain name like a FULL STOP '.' or Forward SLASH '/' which is used to compose a URL. For example, U+2215 (∕) DIVISION SLASH is much similar a regular ASCII '/' in many fonts [7].

For example, an attacker who owned a DNS server can register IDN domain name called facebook.com∕home.<XXXX>.com using the DIVISION SLASH and most of the currently available Internet browsers will parse this domain and convert it to its Punycode form facebook.xn–comhome-ef0d.<XXXX>.com. Although homoglyphs for those particular characters are banned in the RFC, most browsers are still processing part of them as per section III-A.

III. IDN Visual Spoofing Attack Mitigation Techniques

Most of the visual spoofing attacks can be mitigated at the domain registry side, while others must be mitigated at the user agents (For example browsers, email clients, and other programs that display and process URLs). The registry has most data available about the alternatives registered names and can handle that information most efficiently at the time of registration. However, this will require significant processing from the registry side and may complicate the domain name registration process. On the other hand, the registry can not prevent an attacker attempting to register a 3rd or 4th level

domain name, notably, if the attacker is using syntax spoofing attack. Hence, mitigation at the user agent side is needed as well.

In this section, we are listing most of the existing techniques that may help to identify and mitigate IDN domain masquerading at the user agents - web browsers mainly, and by using centralized monitoring solutions.

A. User Agents

Web browsers are the primary connection to the Internet, and multiple applications are relying on them to function. That makes the browser's security very critical, and it may be the last resort before being exploited. Web browsers such as Microsoft Internet Explorer, Mozilla Firefox, and Apple Safari are pre-installed on almost all operating systems. All of them are trying their best to improve their security to help the end user from recognizing suspicious web pages that could lead to a malicious domain [17]–[19]. For example, Chrome offers Safe Browsing[2] which prevents malwares and phishing campaigns by downloading a list of information to the browser about sites that may contain malicious software or are engaged in phishing activity.

Regarding IDN visual spoofing detection, most user agents will convert the IDN domain in the URL bar to its Punycode format; for example, http://yahoo.com will be converted to http://xn–80a2aar51d.com; when a user attempts to access that webpage. Each browser[3] will, however, behave differently when it comes to the IDN script type. If the provided domain name is all in Latin, Firefox will not convert it to the Punycode format [20].

We evaluated some Internet browsers to check their behavior by providing to them different IDN labels. After that, we captured their response to see if they are going to convert the provided IDN labels to their Punycode format or not; as listed in Tables II,III. For example, if the provided IDN domain's type is a single script most of the browsers[3] will not convert it to its Punycode format as shown in Table I, on the other hand, if the provided IDN domain is Mixed script all of them will convert it to its Punycode form as listed in Table II

TABLE I: Single Script IDN to Punycode Conversion

Script	IDN Domain	Firefox	Chrome	Safari	IE
Latin	eläketurvakeskus.net	✗	✗	✗	✓
Greek	вновостях.net	✗	✗	✓	✓
Cyrillic	yahoo.com	✗	✓	✓	✓

TABLE II: IDN Mixed Script to Punycode Conversion

Scripts	IDN Domain	Firefox	Chrome	Safari	IE
Latin, Greek	amazon.com	✓	✓	✓	✓
Greek,Cyrillic	google.com	✓	✓	✓	✓
Latin, Cyrillic	google.com	✓	✓	✓	✓

[2]https://support.google.com/chrome/answer/99020?hl=en

[3]Firefox 57.0 (64-bit) ‖ Chrome 61.0.3163.100 (Official Build) (64-bit) ‖ Safari 11.0 (12604.1.38.1.7) ‖ IE 11.0.9600

In terms of the RFC compliance, many Internet browsers[3] failed to follow the security standards while they are validating an IDN label. Most of them are violating one or more of Label Validation rules that are defined in section 4.2. in the RFC [9]. For example, Firefox processes the banned Unicode character 'U+2024' (One Dot Leader)[4]; while all of them would violate the BIDI rule [21] when it comes to Arabic Tashkeel [22]; for example, they will process facebook.com label which contains a Right-To-Left character 'U+064e' (the Arabic Fatha)[5]. Referencing the RFC [21], any Right-To-Left label shouldn't contain any Left-To-Right characters. Table III contains a list of some not valid IDN labels and the result of each browser that process those labels.

TABLE III: RFC Compliance - Invalid Labels Processing

Unicode Char	IDN Domain	Firefox	Chrome	Safari	IE
ARABIC FATHA	facebook.com	✓	✓	✓	✓
ONE DOT LEADER	www.era.net	✓	✗	✗	✗

An advanced user can change browsers' default behavior when it comes to IDN domain processing. For example, Firefox can be modified by tweaking the "about:config" settings to enforce the Punycode conversion for any supplied IDN label regardless their script group (character sets). Another technique could be used by installing one of the available plugins that block or allow IDN domains. For example, IDN-Safe[6] can be used to fully prevent IDN domains and whitelist what is needed.

Another web plugin was developed by Viktor Krammer to detect and prevent IDN spoofing-based attacks called Quero[7]. When a user attempts to access a given URL, Quero will highlight the recognized IDN domain as well as it will display the script groups (character sets) to the end user. If the IDN Domain is not matching the RFC, then Quero will block the access to that suspicious domain [17].

Tweaking the browser settings or even installing web plugins will not adequately protect end users against IDN attacks if the browser is not fully complying with the RFC [9], [10]. Web browsers have attempted various security fixes against the IDN attacks. However, it is obvious that those attempts are not sufficient to prevent the threats of the IDN attacks.

B. Monitoring Solutions

Unlike user agents, centralized monitoring solutions will process the domain registry data to detect any violations in newly registered domain names. That kind of solutions will work by providing a list of target domain names to be monitored; if they found any registered domain name that is close enough to a domain in the provided target list, an alert will be generated.

DNSTwist[8] is one of the tools that can observer domain name spoofing attempts. It can detect some attacks such as

[4]https://unicode.org/cldr/utility/idna.jsp?a=http://www.era.net

[5]https://unicode.org/cldr/utility/idna.jsp?a=http://facebook.com

[6]https://github.com/AykutCevik/IDN-Safe

[7]http://www.quero.at/

[8]https://github.com/elceef/dnstwist

typosquatting with high precisions through altering, removing, adding or swapping characters for a given input domain name. After that, DNSTwist will perform DNS query to check if that new domain is registered or not. When it comes to IDN domain name spoofing, DNSTwist will attempt to swap each domain character with its corresponding homoglyph using its homoglyph collision database. The main drawback with it is, it has very limited homoglyphs in its collision database. Hence the results are not accurate or comprehensive.

Homoglyph Monitoring System is another solution developed by Joseph Miller focusing only on IDN spoofing attacks detection [23]. This system consists of four main modules: (1) Glyph matching module that attempts to guess all homoglyph possibilities in the Unicode landscape for all ASCII character using an automated process, then a user interaction is required for tuning purposes. (2) Attack vector matching module which takes the target domain strings from the user to monitor it, then it will iterate over all characters in the target string to swap them with their Unicode homoglyphs using the existing collision database to uniform "attack vector strings" which will be sent to the DNS Analyzer module. (3) DNS analyzer module will query the generated "attack vector strings" against the DNS to check if any of attack vector strings are registered (i.e., to check if there is an attempt to masquerade one of the target domains). (4) Action module which alerting the admin about the detected spoofing attempts. the main issue with this system is the scalability as the more significant collision database yield to higher DNS traffic would be as quoted below:

> Depending on the number of homoglyphs in each set, there could be hundreds of thousands of potential attack vector strings [23]

Having a wide range of Unicode characters (2^16 for the narrow build and 2^32 for the wide build) and the existence of Homoglyphs, made it highly challenging to maintain a comprehensive and accurate Homoglyoh collision database. A lot of attempts have been made to solve this problem. One of the solutions was created by Narges Roshanbin and James Miller to find existing homoglyphs in the Unicode space by using Normalized Compression Distance [24]. Another method was implemented by Anthony Y. Fu, Xiaotie Deng, and Liu Wenyin by constructing a Unicode character similarity list (UC-SimList) based on character visual and semantic similarities using a nondeterministic finite automaton [25].

The Unicode Consortium had maintained their Unicode confusable list as well. This list contains the similarities between the Unicode characters that can be used for IDN Domain spoofing attacks [8], [26]. The main drawback with the Unicode confusable list is, it contains a lot of Unicode characters that are not allowed in the IDN domain, for example, the Medium Mathematical Space "U+205F" exits in the list although it is not allowed to be used in constructing any IDN Domain. Another issue is the list is not mentioning many Unicode characters with their similarities, for example, the similarity between å and a is not listed.

IV. Proposed IDN Detection System

To understand the existing problems in the previous systems, we have added a fuzzing module that fuzzes any given domain name - the same technique used in the existing tools (DNSTwist and Homoglyph Monitoring System [23]) to generate all possible IDN attack vector strings. Then we added "yahoo.com" and "google.com" to the fuzzing list to generate all Unicode labels possibilities. We used a system with 3.1GHz and 32GB of RAM to execute the fuzzing module. Then, we captured the number of generated target strings as well as the required time to produce them.

TABLE IV: Attack Vector Strings Timing and Efforts

TargetDomain	charsets	Posible Strings	Time#
yahoo.com	Latin	7 M	4.4 sec
yahoo.com	Latin,Greek,Cyrillic	35 M	25 sec
yahoo.com	All	70 M	61.6 sec
google.com	Latin	86 M	104.2 sec
google.com	Latin,Greek,Cyrillic	273 M	275 sec
google.com	All	-	-

We've found that, when we used the Latin homoglyphs only to generate the attack vectors strings (fuzzing) of yahoo.com, the execution remained for 4.4 seconds and produced 7 Million attack vector possibilities. If we used all homoglyphs in the Homoglyphs database, fuzzing yahoo.com required almost 61 seconds with 70 Million attack possibility as per Table IV. Hence it will need 70 Million DNS query to check if any of those strings are registered or not under one TLD only. On the other hand, when we tried to fuzz "google.com" using all exiting homoglyphs, the system crashed. We concluded that this approach is not feasible.

The proposed system was designed in a way that resolves homoglyph collision and system scalability problems to get both accurate and scalable detection capabilities. This can be accomplished by employing the following design improvements:

1) Instead of generating all possible strings for a given domain name, the systems will use a dataset of all newly registered domain names and pivot it as a reference for comparison.
2) Instead of generating all homoglyphs for an ASCII character. The collision database will contain only the Unicode characters that have been used in the registered domains. Any newly registered IDN domain name with a Unicode character that is not observed before must be added to the Homoglyph collision database.
3) Adding analytics engine to identify and categorize the malicious domains.

Based on the above design decisions, the proposed system will download given DNS zone files, to extract the newly registered Punycode domains - domains with Unicode characters. Then, it will convert them to their Unicode form to elicit all the Unicode characters. Next, it will replace all the Unicode characters with their ASCII Homoglyphs producing all spoofing possibilities - attack vector strings. After that, the system will compare the resulted possibilities with the domains in the target domain list; if there is a similarity identified,

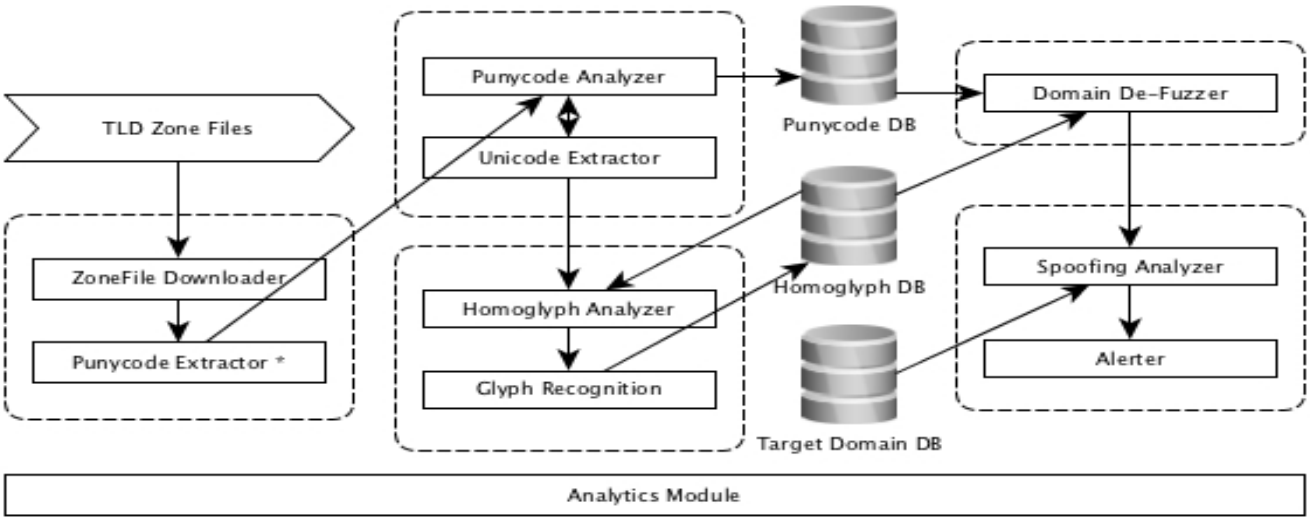

Fig. 1: Proposed IDN Monitoring Solution - System Design

the corresponding Punycode will be marked as a suspicious domain name. Finally, the solution will promote a suspicious domain to malicious if the suspicious domain is not hosted on IP Address space of the target domain or it is not registered by the target domain's organization.

For example, instead of generating all Unicode permutations for "google.com" the system will iterate over the newly registered Punycodes to reduce them by using the homoglyph collision database to get the possible "target attack strings" (e.g., xn–ggle-qqaa.com with IDN form "góógle.com" will be converted to "google.com" after substituting all Unicode characters with their ASCII homoglyphs). Then the system will compare the resulted "target attack strings" against the target domain list which has "google.com" to identify the suspicious Punycode. If the system found one (xn–ggle-qqaa.com), the system will check the IP address and the whois registrant organization of that suspicious domain to see if it's owned by the target domain "google.com" or not.

The proposed system consists of six main components: 1) The Punycode Extractor Module, 2) Punycode Analyzer Module, 3) Homoglyph Analyzer Module, 4) Fuzzer Module, 5) Spoofing Detection Module, and 6) Analytics Module. In this section, we will describe in details the system components and proposed IDN detection algorithms, as shown in Figure1.

A. Punycode Extractor Module

A daily copy of DNS zone files can be accessed and downloaded from publicly available organizations such as ICANN[9] and Verisign[10]. Those zone files contain a list of all registered domain names and their corresponding NS servers.

An average of 100k .com and 10k .net of domain names are being registered on a daily basis; some of them are Punycode domains - domain names with Unicode characters - almost ~1170 domains. Figure 2 shows the registered Punycode domain names during the system monitoring period. The Punycode Extractor, the first module, will process and extract all newly registered Punycode domain names and forward it for further analysis as described in Algorithm 1.

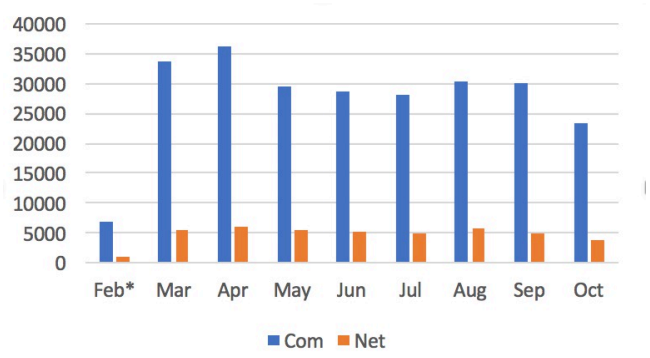

Fig. 2: Punycode Domains Registration Timeline, 2017

As shown in Figure 2, an average of 32000 Punycode domain names are being registered on each month[11]. Those domains are being extracted and indexed in the Punycode DB as will be shown in the next stage.

B. Punycode Analyzer Module

The input to this module is all the newly registered Punycode domains. This module will convert Punycode domains to their IDN equivalent form. For each IDN domain, the analysis module will extract the Unicode characters; then it would determine the domain ScriptType (i.e., Single or Mixed). After the extraction is performed, all information for each Punycode domain are stored, as metadata in the Punycode database; as shown in algorithm 2.

For example, the Punycode "xn–ggle-qqaa.com" will be converted to "góógle.com", then it will be sent for analysis

[9]https://czds.icann.org
[10]https://www.verisign.com

[11]Feb is being ignored as the solution started to function from Feb 22nd

```
Data: Downloaded Zone Files
Result: New Punycode Domains
new_punycodes;
for domain in zonefile do
    if domain.starts.with('xn–') then
        if domain not in old_punycode_db then
            new_punycodes += domain;
            old_punycode_db += domain;
        end
    end
end
```

Algorithm 1: Extracting New Punycodes

to extract the Unicode characters which are 'ó,' and 'ó'. After that it the module will check the extracted Unicode character sets, which in this case is LATIN for both of them. In the end, "xn–ggle-qqaa.com" will be marked as a SingleScript.

```
Data: New Punycode Domains
Result: Analyze and Index New Punycode Domains
unicode_chars;
metadata;
for punycode in punycodes do
    idn_format = to.unicode(punycode);
    unicodes = extract_unicode_chars(idn_format);
    for unicode_char in unicode_chars do
        scripts += get_script_name(unicode_char);
        unicode_chars += unicode_char;
    end
    if length(unique(scripts)) > 1 then
        metadata[punycode] = Mixed;
    else
        metadata[punycode] = Single;
    end
end
save_to_PunycodeDB(metadata);
```

Algorithm 2: New Punycode Analyzer

C. Homoglyph Analyzer Module

The primary objective here is to build up a Homoglyph collision database that contains a list of all used Unicode characters and their similarities –Homoglyphs. This module takes the Unicode characters that were extracted from the newly registered domains and verifies if there are any Unicode characters that have not been seen before using Algorithm 3. Then it sends all the newly observed Unicode characters for recognition.

The rogue character recognition process is performed using two stages: (1) Check if the Unicode character has a similarity in the Unicode confusable list [8]. (2) A manual process is incorporated in which a human analyst needs to verify the result. The analyst may also mark the identified Unicode character as a combining character that can be used along with another character to change its appearance. For example, the umlaut character if it combined with a will result in "ä", as described in algorithm 4.

```
Data: Extracted Unicodes, HomoglyphDB
Result: Get New Unicode Characters
new_unicode_chars;
for unicode_character in unicode_characters do
    if unicode_character not in HomoglyphDB then
        new_unicode_chars += unicode_character ;
    end
end
```

Algorithm 3: New Unicode Extractor

```
Data: New Unicodes and Unicode_Similarity_List
Result: Determine Similarities
data;
for unicode_character in unicode_characters do
    similarities =
     Unicode_Similarity_List.get(unicode_character)
     print_to_analyst(unicode_character, similarities);
    data[unicode_character] = read(similar_character);
    data[unicode_character] += read(similarity_level);
end
save_to_HomoglyphDB(data);
```

Algorithm 4: Character Recognition

D. Fuzzer Module

Using the Punycode database and the Homoglyph DB; the Fuzzer module will iterate over all newly registered Punycodes, then it will replace the Unicode characters in the Unicode equivalent form (IDN), with their ASCII similarities to generate the possible attack vector strings. Those strings will be passed to the Spoofing Analyzer module, as described in algorithm 5.

For example, the Punycode "xn–e-dla47fa3625aa.com" - with IDN form "googłe.com" - has five Unicode characters which are 'g', 'o', 'o', 'g', and 'ł'.The module will use those characters to query the Homoglyph DB to find their ASCII similarities. Then it will substitute all those Unicode characters with their similarities. In the end, the module will return one possible attack vector string which is "google".

```
Data: PunycodeDB, HomoglyphDB
Result: attack_vector_strings
attack_vector_strings;
for punycode_domain in punycode_domains do
    ascii_format = to.ascii(punycode);
    for unicode_char in ascii_format do
        homoglyphs += get_homoglyphs(unicode_char);
        for homoglyph in homoglyphs do
            attack_vector_strings+= replace(unicode_char,
             homoglyph);
        end
    end
end
```

Algorithm 5: Punycode Fuzzer

E. Spoofing Detection Module

This is the main module in the system, as the primary aim of the proposed solution, is to detect Domain Spoofing in the

Internationalized Domain Names. The potential attack vector strings – fuzzed Punycode domains – will be input to this module to check if any of the identified strings match a domain in the target Domain DB. If there is a match the corresponding Punycode will be marked as a suspicious domain name and will be sent for deep analysis to be checked if it is malicious or not. Any suspicious domains will be tagged as malicious if:

1) The domain is hosted on IP address that doesn't belong to the authentic domain address space, and
2) The registrant Organization is not the same as the authentic domain name

For example, the spoofing detection module will take "xn–e-dla47fa3625aa.com" and its possible attack vector string which is ["google"] after the substitution. Then it will compare the attack vector string against the Target Domain DB, which has "google.com" in its list. As a result, the Punycode "xn–e-dla47fa3625aa.com" will be marked as suspicious, and it will be sent for more analysis. The result of the analysis[12] will show that "xn–e-dla47fa3625aa.com" is hosted on a rouge IP address "34.202.122.77", and it does not belong to Google.com address space, as well as the registrant organization is "DYNADOT LLC" which is not matching with google.com registrant organization (Google LLC). That's why "xn–e-dla47fa3625aa.com" will be promoted from suspicious to a malicious domain name.

```
Data: Attack Vector String, Target_Domain_DB
Result: Alerts
for string in attack_vector_strings do
    if string in TargeT_Domain_DB then
        raise_alert(string);
    end
end
```

Algorithm 6: Spoofing Detection

F. Analytics Module

The analytics module has an API interface to the system components, including the Punycode and Homoglyph databases. Also, it is the integration gateway with the external modules Whois databases[13], Fuzzy Hash module and VirusTotal[14]. Also, the analytics module has its web browser, that would simulate a user accessing a given URL, and, of course, it can do DNS queries[15].

The analytics module would help the analyst to detect an active phishing attack - a spoofed domain hosts a webpage looks exactly like the target domain webpage - as well as it would help to detect if there is a phishing campaign against any domain in Target Domain List. In the end, it may give some insights about the trends in IDN domain registrations, and the attack groups. The analytics module as well can provide threat intelligence feeds to detection and prevention systems such as IPS/IDS and Security Information and Event Management (SIEM) solutions.

V. System Evaluation

Using the proposed solution, we started to monitor newly registered domain names under the TLDs .com and .net from 23rd Feb 2017 to 24th October 2017. During that period ~41500 Punycode domain names have been registered. We have narrowed our monitoring and attack detection for (1) Majestic top 100 Thousand[16] and (2) some of the social media domain names[17]. In this section, we will provide detailed results generated by the monitoring solution.

A. Social Media Monitoring

The proposed system started to monitor the social media domain names from the 23rd of Feb, 2017. Almost every single day there was an alert triggered indicating that a suspicious IDN domain has been registered. Interestingly all of them are marked as malicious as 1) all of them are hosted on a rouge IP address - IP address not owned by the real domain, and 2) the registrant organization in whois data is not matching those in the real domains.

For example, the Punycode "xn–facbok-rh8bmg.com" was registered on 12-04-2017 and its Unicode form is faceḅook.com - looks like facebook.com after substituting all Unicode characters with their similarities - was reported as a suspicious domain name. After that, It was sent for deep analysis[12], and we found that it was hosted on 103.18.6.118 and the registrant organization was "GMO INTERNET, INC". That's why this domain has been tagged as malicious. For further clarification, Some of the captured alerts are listed in the appendix - Tables VII, VIII.

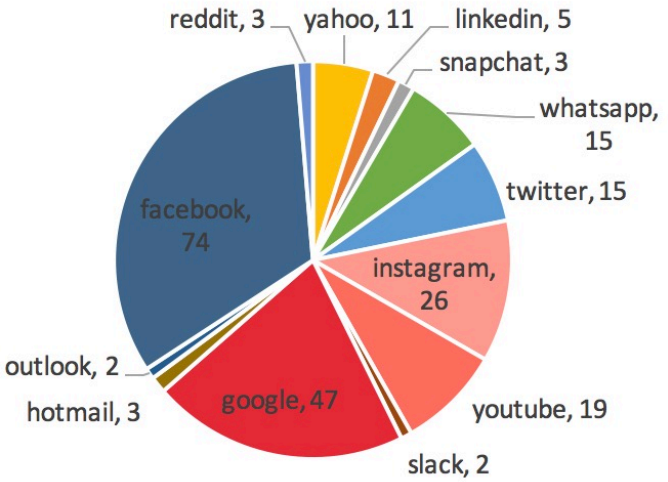

Fig. 3: Most Targeted Domains from Feb till Oct 2017

For demonstration purposes, the result of the malicious domains had been aggregated and grouped by Target Domains

[12] Analysis was kicked off on 2018-04-24
[13] https://www.whoisxmlapi.com
[14] https://www.virustotal.com
[15] https://github.com/YahiaKandeel/netutils/blob/master/resolver.py
[16] https://blog.majestic.com/development/majestic-million-csv-daily
[17] yahoo, linkedin, snapchat, whatsapp, pinterest, twitter, instagram, youtube, slack, google, hotmail, outlook, facebook, reddit

to graph the most targeted social domains in the stated period. As shown in Figure 3, the detection system has identified 225 malicious IDN domains that have been registered to masquerade the social media domain names. The top spoofing attempt was for facbook.com which was masqueraded 74 times; while google.com came in the 2nd place to be spoofed by 47 times.

Interestingly, all captured social media spoofing attempts were using single script IDN domains, and most of them are utilizing the Latin language script – as presented in the appendix - Table VII. Because of most browsers except Microsoft Internet Explorer will not convert those fake IDN domains to their equivalent form; the user most likely will be lured into accessing malicious webpage instead of the legitimate one.

Diving deep in the analysis, all registered fake IDN domains had been grouped by their Whois registrar. We have found that NAMECHEAP and GoDaddy were the top used registrars as per Figure 4. Maybe they are the best choice to register a spoofed IDN domain name. Another notice is, 21 of those IDN domains' data were missing as their owners didn't complete the registration process. We have found as well; some malicious IDN domains had been registered more than once during the monitoring period. For example, "xn–fcebk-j11bxma.com" it was captured for the first time on 2017-06-23, and the second time was on "2017-09-03". It is possible that the domain had been taken down by a legal entity.

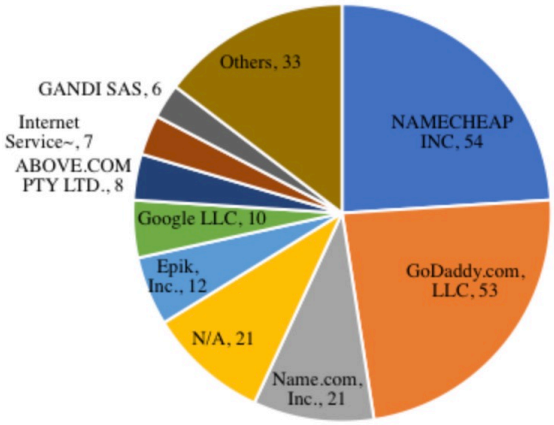

Fig. 4: Malicious IDN Domains - Registrars

In order to profile the registrants (attackers) we started to query the malicious domains against the whois database to extract the registrant telephone number, then we grouped the results based on that telephone number. We found that the number +5078365503 were the top used number by 49 out of 225 times and the next one, was +17208009072 by 20 times.

What has been noticed as well, that some of the malicious domain names were redirecting our hits to the authentic websites (e.g., xn–yaoo-h84a.com were redirecting to www.yahoo.com). Although these fake domains are categorized as phishing website by VirusTotal. We concluded that behavior as, during a targeted phishing campaign, it is important for an attacker not to be detected by site categorization tools to maintain the domain and IP address reputation in order not be blocked by the anti-malware or anti-phishing security client agents. Hence, the attacker can whitelist a specific IP address ranges for those whom they are targeting, and for the rest, they would be redirected to the authentic website.

During the analysis period, we observed that some malicious domains were sharing the same IP address, as listed in table V. This could be additional evidence that the same attacker/group is making use of multiple spoofed IDN domain to trick users to take unsafe actions.

TABLE V: Malicious IDN Domains - Top IP Addresses

IP Address	Count	Owner Organization
198.54.117.200	12	Namecheap, Inc.
91.195.240.82	11	SEDO GmbH
72.52.4.120	6	Akamai Technologies, Inc.
103.224.182.244	4	TRELLIAN-AU
103.18.6.118	3	RUNSYSTEM-VN

To validate the captured results, in 31-03-2018 we have checked the malicious IDN domains against VirusTotal; to see if it was able to identify the reported IDN domains as malware or phishing domain names. What we have found is that 102 out of the 225 domains are reported as phishing domains, and 97 out of the 225 domains were hosting malicious URLs.

B. Majestic Top 100 Thousand

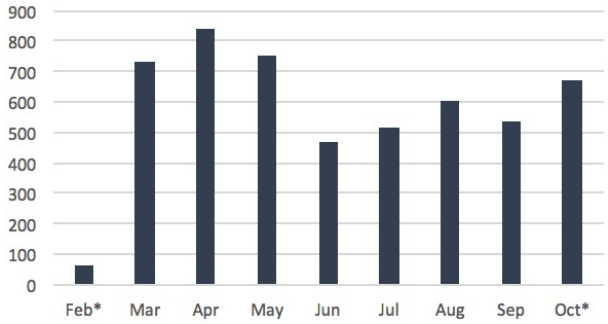

Fig. 5: Suspicious IDN Domains Creation Timeline

We have monitored as well all domain names listed in Majestic Top 100K for seven months. We found that an average of 600 suspicious IDN domain names has been registered on a monthly basis - as shown in Figure 5. That could indicate the effectiveness and prevalent of the IDN-based attacks. Due to the number of whois-query limitations, we opted not to validate if the suspicious domains were malicious or not. Instead, we performed a string analysis on the suspicious domains only to infer the most character set (language script) script used in composing those domains as well as the most spoofed ASCII character.

From the analytics module, we constructed a list that indicates the most used character set (language Script) to spoof a domain name, as shown in Figure 6. From that, we observed

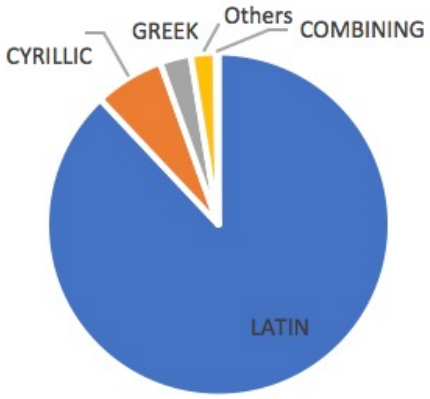

Fig. 6: Most Used Character Sets

that the attackers prefer to use the Latin's character set to masquerade a legit domain name.

We retrieved a list of all used Unicode characters from the HomoglyphDB. Then for each Unicode character, we counted how many times it was used in the reported suspicious IDN domains - Unicode domains that are registered in order to masquerade the Majestic top 100K. Table VI lists the top ten used Unicode characters. From that table, we can see that the Unicode letter " é " has been used by 939 times, and " ö " was used by 567.

TABLE VI: Top 10 Unicodes Used in Masquerading Attacks

Unicode	Description	Count#
é	Latin Small Letter E With Acute	939
ö	Latin Small Letter O With Diaeresis	567
í	Latin Small Letter I With Acute	465
ä	Latin Small Letter A With Diaeresis	460
ı	Latin Small Letter Dotless I	422
ü	Latin Small Letter U With Diaeresis	401
á	Latin Small Letter A With Acute	350
ó	Latin Small Letter O With Acute	326
ø	Latin Small Letter O With Stroke	324
å	Latin Small Letter A With Ring Above	278

Based on the above results, a histogram of the most spoofed ASCII character was assembled as shown in Figure 8. This figure presents the ASCII character and how many times it was spoofed, and as such, it most likely to be used in a domain spoofing and phishing attack. Figure 7 is about a histogram of frequency of different characters in the Majestic Top 100K. From that figure, we can see the e, a, o, i are the most used character for assembling a domain name.

As shown in Figure 8, the most spoofed ASCII characters are the vowels. Letter "e" came on the top list of spoofing attempts by more than 2700 times. On the other hand, the least spoofed character was "q" by seven times.

VI. System Limitation

Currently, the proposed solution will detect only the malicious IDN domain names which are registered under the .net and .com TLDs. We can extend our monitoring scope to include more TLDs such as .info and .org. Another modification can be done to enhance and speed up the Homoglyph

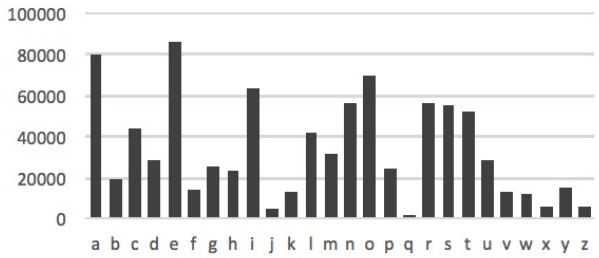

Fig. 7: Majestic ASCII Character Histogram

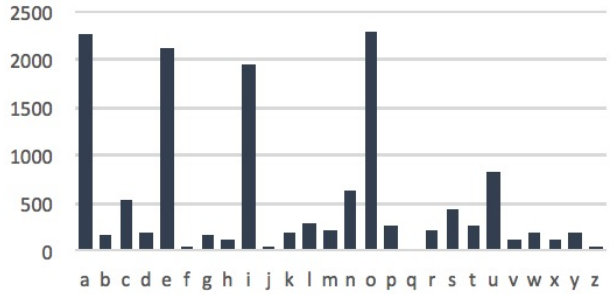

Fig. 8: Most Spoofed ASCII Character

detection process by deploying OCR mechanism as part of homoglyph identification process instead of depending on human interactions and feedback.

Also, we can add micro-services to enhance the detection process such as:

1) Fuzzy Hashing[18] module, to check if the hosted webpage on the malicious domain looks the same as the authentic one.
2) A typosquatting detection module can be added to the system, by calculating the Levenshtein distance[19] between the target domains and the newly registered domains.
3) A severity level can be calculated based on the similarity between the malicious IDN domain and the target domain.

Besides to those micro-services, a correlation engine can be added to correlate between the past and new events to detect which are groups/entities that are launching those IDN attacks against the target domain names.

VII. Conclusion

Internationalization in Domain Names (IDN) inherited some of Unicode security risks, by allowing Homograph in domain names. Attackers are abusing those risks and spoofing a domain name that visually appears legitimate while it is a malicious domain and used in social engineering and phishing attacks.

IDN spoofing attacks target the well-trained end user, making phishing identifications by well-trained human eyes,

[18]https://ssdeep-project.github.io/ssdeep/index.html
[19]http://rosettacode.org/wiki/Levenshtein_distance

sometimes, almost impossible. That would elevate the IDN Domain Name spoofing to one of the most critical types of attacks.

The problem, however, is not in the of Homographs concept. The lack of browsers URL verification and user alert is one of the reasons that could facilities this attack. Some browsers do not fully comply with the proposed Homographs RFC [9], [10] leaving millions of users at risk of being a victim of this attack.

In this paper, we explained the IDN attack types and the browser behavior for each attack type. Then, we proposed a solution to detect IDN based attacks. After that, we have evaluated the solution by detecting the newly registered domains that masquerade Majestic Top 100K, and some of the social-media domains.

The result of the proposed solution showed that an average of 550 suspicious IDN domains are being registered on a monthly basis, and most of them are utilizing the Latin character set. Hence most of the available browsers will not convert those IDN domains to their Punycode form, that could explain why users may be lured to access the fake website. The analysis result, also, showed that attackers might prefer a particular registrar to host their malicious IDN domain.

References

[1] M. J. Duerst, "Urls and internationalization," December 1996. [Online]. Available: http://lists.w3.org/Archives/Public/uri/1996Dec/0038.html

[2] T. U. Consortium, "Unicode standard version 10," June 2017. [Online]. Available: http://www.unicode.org/versions/Unicode10.0.0/

[3] X. Zheng, "Phishing with unicode domains," April 2017. [Online]. Available: https://www.xudongz.com/blog/2017/idn-phishing/

[4] E. Gabrilovich and A. Gontmakher. (2002) The homograph attack. [Online]. Available: http://www.cs.technion.ac.il/%7Egabr/papers/homograph_full.pdf

[5] ICANN, "Statement on idn homograph attacks and request for public comment," February 2005. [Online]. Available: https://www.icann.org/news/announcement-2005-02-23-en

[6] "Unicode character database." [Online]. Available: http://www.unicode.org/ucd/

[7] M. Davis and M. Suignard, "Unicode security considerations," September 2014. [Online]. Available: http://unicode.org/reports/tr36

[8] T. U. Consortium", "Unicode confusables." [Online]. Available: http://www.unicode.org/Public/security/revision-03/confusablesSummary.txt

[9] J. Klensin, "Internationalized domain names in applications (idna): Protocol," RFC 5891, 2010. [Online]. Available: https://tools.ietf.org/html/rfc5891

[10] A. Costello, "Ipunycode: A bootstring encoding of unicode for internationalized domain names in applications (idna)," RFC 5891, 2003. [Online]. Available: https://tools.ietf.org/html/rfc5891

[11] P. Hoffman and A. Costello, "Internationalized domain names in applications (idna)," RFC 3490, 2003. [Online]. Available: https://tools.ietf.org/html/rfc3490

[12] M. Davis and M. Suignard, "Unicode idna compatibility processing," June 2017. [Online]. Available: http://www.unicode.org/reports/tr46/

[13] M. Davis and K. Whistler, "Unicode normalization forms," May 2017. [Online]. Available: http://www.unicode.org/reports/tr15/

[14] S. Patil, "Spoofing around the urls," September 2009. [Online]. Available: https://www.symantec.com/connect/blogs/spoofing-around-urls

[15] K. M. Slavitt, "Protecting your intellectual property from domain name typosquatters," 2004. [Online]. Available: http://corporate.findlaw.com/intellectual-property/protecting-your-intellectual-property-from-domain-name.html

[16] "Typosquatting: How 1 mistyped letter could lead to id theft." [Online]. Available: http://www.bankrate.com/finance/credit/typosquatting-identity-theft.aspx

[17] V. Krammer, "Phishing defense against idn address spoofing attacks," 2006. [Online]. Available: http://www.quero.at/papers/idn_spoofing.pdf

[18] US-CERT, "Securing your web browser," September 2015. [Online]. Available: https://www.us-cert.gov/publications/securing-your-web-browser

[19] F. Comity, "Idn display algorithm," April 2017. [Online]. Available: https://wiki.mozilla.org/IDN_Display_Algorithm

[20] P. Hannay and C. Bolan, "Assessment of internationalized domain name homograph attack mitigation," in ***Australian Information Security Management Conference***, Perth, Western Australia, December 2009.

[21] C. Karp, "Right-to-left scripts for idna," August 2010. [Online]. Available: https://tools.ietf.org/html/rfc5893

[22] T. U. Consortium, "Unicode arabic extended-a," 2017. [Online]. Available: http://unicode.org/charts/PDF/U08A0.pdf

[23] J. A. Miller, "Homoglyph monitoring," U.S. Patent 13/659 577, 16, 2014.

[24] N. Roshanbin and J. Miller, "Finding homoglyphs - a step towards detecting unicode-based visual spoofing attacks," in ***International Conference on Web Information Systems Engineering***, Sydney, Australia, October 2011.

[25] A. Y. Fu, X. Deng, and L. Wenyin, "Regap: A tool for unicode-based web identity fraud detection," Hong Kong SAR, P. R. China, 2006.

[26] T. U. Consortium", "Unicode confusables." [Online]. Available: https://unicode.org/cldr/utility/confusables.jsp

Appendix

TABLE VII: IDN Domain Name Spoofing in September 2017

Date	Target Domain	Fake IDN Domain	Punycode Format	Unicode#	ScriptType	CharacterSet
9/1/17	google	gоogłe.com	xn–e-dla47fa3625aa.com	5	Single	LATIN
9/4/17	instagram	instɑgrɑm.com	xn–instgrm-obdc.com	2	Single	LATIN
9/5/17	facebook	fạcebọọk.com	xn–fcebk-j11bxma.com	3	Single	LATIN
9/7/17	google	góógle.com	xn–ggle-qqaa.com	2	Single	LATIN
9/7/17	instagram	ịnstagram.com	xn–istagram-i99c.com	1	Single	LATIN
9/9/17	reddit	reddît.com	xn–reddt-8sa.com	1	Single	LATIN
9/9/17	google	gôógle.com	xn–ggle-qqaf.com	2	Single	LATIN
9/9/17	facebook	fäcebóók.com	xn–fcebk-gra2ia.com	3	Single	LATIN
9/9/17	instagram	instagràm.com	xn–instagrm-5ya.com	1	Single	LATIN
9/9/17	twitter	twïttër.com	xn–twttr-qsaz.com	2	Single	LATIN
9/9/17	youtube	yọutube.com	xn–yutube-3wb.com	1	Single	LATIN
9/9/17	youtube	yóutube.com	xn–yutube-bxa.com	1	Single	LATIN
9/9/17	snapchat	snapchät.com	xn–snapcht-bxa.com	1	Single	LATIN
9/10/17	twitter	twıtter.net	xn–twtter-q9a.net	1	Single	LATIN
9/15/17	instagram	ịnstagram.com	xn–istagram-i99c.com	1	Single	LATIN
9/16/17	facebook	faceboök.com	xn–facebok-f1a.com	1	Single	LATIN
9/20/17	google	googl□.com	xn–googl-0e9c.com	1	Single	LATIN
9/21/17	instagram	instagrâm.com	xn–instagrm-oza.com	1	Single	LATIN
9/21/17	linkedin	link̨edin.com	xn–linedin-6gb.com	1	Single	LATIN
9/22/17	instagram	instágràm.com	xn–instgrm-ewag.com	2	Single	LATIN
9/22/17	instagram	ìnstàgram.com	xn–nstgram-bwa8h.com	2	Single	LATIN
9/22/17	instagram	instàgram.com	xn–instgram-2ya.com	1	Single	LATIN
9/26/17	facebook	facebooḱ.com	xn–faceboo-uw3c.com	1	Single	LATIN
9/26/17	facebook	fácébóók.com	xn–fcbk-5na3c0da.com	4	Single	LATIN
9/27/17	facebook	facebo̱o̱k.com	xn–faceoo-6g7b8o.com	2	Single	LATIN

TABLE VIII: IDN Suspicious Domain Name Analysis - 01/04/2018

Domain	Fake IDN Domain	WhoisOrg	Registrar	IPAddress	Redirect	Phighing?	Mal.URLs#
facebook	xn–faceoo-6g7b8o.com	Domains By Prox	GoDaddy.com, LLC	50.63.202.41	False	True	1
facebook	xn–faceboo-uw3c.com	Above.com Domai	ABOVE.COM PTY LTD.	184.168.221.55	False	True	1
facebook	xn–fcbk-5na3c0da.com	G2	GoDaddy.com, LLC	50.63.202.48	False	False	0
instagram	xn–instgrm-ewag.com	None	-	-	False	False	0
instagram	xn–nstgram-bwa8h.com	None	-	-	False	False	0
instagram	xn–instgram-2ya.com	None	-	-	False	False	0
instagram	xn–instagrm-oza.com	None	Cronon AG	-	False	False	1
linkedin	xn–linedin-6gb.com	Domains By Prox	GoDaddy.com, LLC	50.63.202.54	False	True	1
google	xn–googl-0e9c.com	Domain Protecti	Name.com, Inc.	69.195.124.232	False	True	1
facebook	xn–facebok-f1a.com	None	OVH, SAS	-	False	False	0
instagram	xn–istagram-i99c.com	WhoisGuard, Inc	NAMECHEAP INC	199.188.200.146	False	True	1
twitter	xn–twtter-q9a.net	None	-	-	False	False	0
reddit	xn–reddt-8sa.com	Domain Protecti	Name.com, Inc.	91.195.240.82	False	False	0
google	xn–ggle-qqaf.com	None	GoDaddy.com, LLC	109.66.14.53	False	True	1
facebook	xn–fcebk-gra2ia.com	Domain Protecti	Name.com, Inc.	91.195.240.82	False	True	1
instagram	xn–instagrm-5ya.com	Domain Protecti	Name.com, Inc.	91.195.240.82	False	True	1
twitter	xn–twttr-qsaz.com	Domain Protecti	Name.com, Inc.	91.195.240.82	False	False	0
youtube	xn–yutube-3wb.com	None	GoDaddy.com, LLC	50.63.202.19	False	True	1
youtube	xn–yutube-bxa.com	Domain Protecti	Name.com, Inc.	91.195.240.82	False	True	1
snapchat	xn–snapcht-bxa.com	Domain Protecti	Name.com, Inc.	91.195.240.82	False	False	0
google	xn–ggle-qqaa.com	None	GoDaddy.com, LLC	184.168.221.49	False	True	2
instagram	xn–istagram-i99c.com	WhoisGuard, Inc	NAMECHEAP INC	199.188.200.146	False	True	1
facebook	xn–fcebk-j11bxma.com	None	UNIREGISTRAR CORP	104.31.85.139	False	True	1
instagram	xn–instgrm-obdc.com	WhoisGuard, Inc	NAMECHEAP INC	192.64.119.4	False	True	1
google	xn–e-dla47fa3625aa.com	None	DYNADOT LLC	34.202.122.77	True	True	1