Assessing Browser-level Defense against IDN-based Phishing

Hang Hu, *Virginia Tech;* Steve T.K. Jan, *University of Illinois at Urbana-Champaign/ Virginia Tech;* Yang Wang and Gang Wang, *University of Illinois at Urbana-Champaign*

https://www.usenix.org/conference/usenixsecurity21/presentation/hu-hang

This paper is included in the Proceedings of the 30th USENIX Security Symposium.

August 11–13, 2021

978-1-939133-24-3

Assessing Browser-level Defense against IDN-based Phishing

Hang Hu[*2], Steve T.K. Jan[*1,2], Yang Wang[1], Gang Wang[1]
[1]University of Illinois at Urbana-Champaign [2]Virginia Tech
{hanghu, tekang}@vt.edu, {yvw, gangw}@illinois.edu

Abstract

Internationalized Domain Names (IDN) allow people around the world to use their native languages for domain names. Unfortunately, because characters from different languages can look like each other, IDNs have been used to impersonate popular domains for phishing, *i.e.*, IDN homograph. To mitigate this risk, browsers have recently introduced defense policies. However, it is not yet well understood regarding how these policies are constructed and how effective they are.

In this paper, we present an empirical analysis of browser IDN policies, and a user study to understand user perception of homograph IDNs. We focus on 5 major web browsers (Chrome, Firefox, Safari, Microsoft Edge, and IE), and 2 mobile browsers (Android Chrome and iOS Safari) and analyze their current and historical versions released from January 2015 to April 2020. By treating each browser instance as a black box, we develop an automated tool to test the browser policies with over 9,000 testing cases. We find that all the tested browsers have weaknesses in their rules, leaving opportunities for attackers to craft homograph IDNs to impersonate target websites while bypassing browsers' defense. In addition, a browser's defense is not always getting stricter over time. For example, we observe Chrome has reversed its rules to re-allow certain homograph IDNs. Finally, our user study shows that the homograph IDNs that can bypass browsers' defense are still highly deceptive to users. Overall, our results suggest the need to improve the current defense against IDN homograph.

1 Introduction

The Internet is progressively globalizing, and yet for a long time, most Internet domain names were restricted to English characters (in combination with hyphen and digits) [43]. To allow people around the world to use their native languages for domain names, Internationalized Domain Names (IDN) were introduced and standardized in 2003 [28], which support Unicode characters from a variety of languages.

As more IDNs are registered, a growing concern is that IDN can be used to impersonate other domain names for phishing purposes. This is because different characters from different languages can look like each other. For example, the Latin character "a" looks similar to the Cyrillic character "а". As a result, attackers can register a domain name `apple.com` with the Cyrillic "а" to impersonate the official website of Apple. This is also called homograph attack [25]. Researchers have analyzed real-world DNS records and found homograph IDNs created for phishing [8, 35, 37, 61].

To mitigate the risk, browsers have designed defense strategies to detect and warn users about homograph IDNs. Commonly, browsers implement rules to detect homograph IDNs that are likely impersonating other legitimate domain names [18]. Once detected, browsers will no longer display the Unicode, but display their Punycode version. Punycode code is initially designed to translate IDNs to ASCII Compatible Encoding so that they can be recognized by legacy protocols and systems. For example, the Punycode for `apple.com` with the Cyrillic "а" is "`xn--pple-43d.com`". By displaying this Punycode in the address bar, browser vendors try to protect users from deception.

In this paper, we want to systematically assess the browser-level defense against homograph IDNs. We seek to answer three classes of research questions:

- First, what policies do major browser vendors implement to prevent IDN homographs, and how well do browser vendors enforce the claimed policies?
- Second, are there ways to systematically bypass existing policies to create homograph IDNs?
- Third, how well can end users recognize homograph IDNs? Are homograph IDNs blocked by browsers more or less deceptive than those that bypass existing defenses?

Empirical Tests. To answer the first two questions, we focus on five major web browsers (Chrome, Firefox, Safari,

[*]Co-first authors with equal contribution.

Microsoft Edge, and IE), and two mobile browsers (Android Chrome and iOS Safari). We analyze their current and historical versions released from January 2015 to April 2020. We treat each browser version as a "black box". Then we construct more than 9,000 testing cases to examine 1) the browser's enforcement of known IDN policies; and 2) possible ways to bypass existing policies. To run a large number of tests over various browsers and platforms, we build a tool to instrument browsers to load testing IDNs while video-recording browsers' reactions. Based on the recorded videos, we automatically analyze how browsers handle different IDNs.

Our analysis has several key findings. *First*, all the browsers have failed to detect certain types of homograph IDNs, with a failure rate ranging from 20.62% to 44.46% Chrome has implemented the strictest rules compared with other browsers. *Second*, we show that it is possible to craft homograph IDNs by exploiting the exceptions and blind spots of existing rules. Several evasion methods are very effective, such as impersonating less "popular" but critical websites (*e.g.*, `.gov`, `.org`), using extensive confusable characters and neglected Unicode blocks (*e.g.*, "Latin Extended-A"), and using whole-script confusables (*i.e.*, all the characters in a domain name are replaced by look-alike characters from a single script). *Third*, although Chrome has strengthened its defense over time, we find that certain rules have been recently revoked, allowing corresponding homograph IDNs to be displayed again. In addition, browsers such as Firefox have not updated their defense policies for years.

To examine whether (and how) the weaknesses in existing IDN policies are exploited in practice, we analyzed 300 million DNS records. We identified 900,000 real-world IDNs and found 1,855 are homograph IDNs that impersonate popular domain names. By loading these homograph IDNs against recent browsers, we showed that the best performing Chrome identified 64.1% of them (*i.e.*, displaying Punycode), while Safari and Firefox only identified 9.7% and 6.1%.

User Study. To answer the third question, we run a user study where participants examine a series of website screenshots. The domain names of the webpages are a mixture of real domain names and homograph IDNs (including those that are blocked by Chrome and those that can bypass Chrome). We study users' ability to judge the authenticity of the domain names under mild priming. Our study shows that users are significantly better at identifying real domain names (94.6% success rate) than identifying homograph IDNs. For example, participants only have a success rate of 48.5% on IDNs blocked by Chrome. In addition, we find homograph IDNs blocked by Chrome are indeed more deceptive than those not blocked. Even so, the homograph IDNs that Chrome missed can still deceive users for 45.8% of the time, posing a nontrivial risk. Finally, we show that users' education level, computing background, age, and gender have a significant correlation with their performance in judging domain authenticity, while website popularity and category are not significant factors.

Contributions. In summary, our key contributions are:

- *First*, we systematically test browser-level defenses against homograph IDNs. We show all of the tested browsers have weaknesses in their policies and implementations, making it possible for homograph IDNs to bypass the defense.
- *Second*, we develop a tool to automatically perform black-box testing on browser IDN policies across browser versions and platforms. The tool can be used to monitor and test future browsers.
- *Third*, we perform a user study to examine user perception of homograph IDNs, and demonstrate the need to enhance the current defense against IDN-based phishing. We have disclosed our findings to related browser vendors.

2 Background

We start by briefly introducing the background of internationalized domain names (IDN) and the related phishing risk.

Internationalized Domain Name (IDN). A domain name is an identification string for Internet hosts or services. Through the Domain Name System (DNS), a user-readable domain name can be mapped to its corresponding IP address. Originally, domain name only allowed ASCII (English) letters, digits and hyphens [43]. In 2003, Internationalized Domain Name (IDN) was introduced to allow people, especially non-English speakers, to use characters from their native languages to create domain names [28]. The new specification supports Unicode characters, which cover more than 143,000 characters from a variety of languages (154 scripts, divided into 308 blocks) [66].

Punycode. The challenge of using IDN is that non-ASCII characters are not supported everywhere. To maintain compatibility with existing protocols and systems, there needs to be a way to convert IDNs to ASCII Compatible Encoding (ACE) strings. The standardized mechanism is called Internationalizing Domain Names in Applications (IDNA) [10, 56]. IDNA converts Unicode labels to an ASCII Compatible Encoding (ACE) label which is also called Punycode. Punycode always starts with "`xn--`". For example, Unicode string "`bücher.de`" is mapped to Punycode "`xn--bcher-kva.de`." IDNA has been adopted by browsers and email clients to support IDNs. Before sending a DNS query for IDN, the domain name is usually translated to its Punycode first to ensure the success of the DNS resolving.

Homograph IDN and Phishing. IDN has been used for phishing because characters from different languages may look like each other. For example, ASCII "a" (U+0061) looks very similar to Cyrillic "a" (U+0430). An attacker thus can use the Cyrillic "a" to construct an IDN to spoof legitimate domain names that contain the ASCII "a", which is called homograph attack [25]. Existing works have performed real-world

Records	Count
DNS records	347,014,213
Unique domain names	143,482,491
Unique IDNs	916,805
Homograph IDNs	1,855

Table 1: Analysis results of `.com` DNS zone file.

measurements and found homograph IDNs that impersonate popular domain names [8, 35, 37, 61].

3 IDN Usage in the Wild

To provide the contexts of IDN usage and motivate our problem, we first empirically analyze the DNS zone files. Through the analysis, we aim to identify real-world homograph IDNs that impersonate popular domain names. Then we test these homograph IDNs against recent browsers to illustrate the problems of browser-level defenses.

Dataset. We obtained the access to the "`.com`" zone file from Verisign Labs[1] in January 2020. "`.com`" is a top-level domain (TLD) where most commercial websites are registered, and is one of the most popular TLDs. We choose `.com` to illustrate the problem, and the same analysis methodology can be applied to other TLDs too. As shown in Table 1, there are 347 million DNS records in the zone file. Among them, there are 143 million unique domain names. For each domain name, we check whether it contains any character outside of the ASCII table.

In total, we find 916,805 IDNs. While the percentage is not high (0.64% of all `.com` domain names), the absolute number of IDNs is still nearly 1 million. We observe that most IDNs come from East Asia and Europe, which is consistent with that of a prior study [37]. We also find script mixing is common. Out of the 916,805 IDNs, 315,671 (34.4%) domain names have script mixing.

Homograph IDNs. To identify homograph IDNs, we follow a common detection method: 1) we select the domain names from Alexa top 10,000 domains [1] as the impersonation target; 2) We search for homograph candidates using a database of look-alike characters (*e.g.*, "a" (U+0061) looks like "а" (U+0430)). We use a comprehensive homograph database from recent work [61]. This database covers look-alike characters across all Unicode blocks that can be displayed by browsers. To detect homograph IDNs, we recursively replace the characters in the target domain name with their look-alike characters (while keeping the TLD unchanged), and then search the modified domain name in our IDN list. Recursive character replacement means we would replace multiple characters in the domain name if there are candidate look-alike characters. If the modified domain name is in the list, we consider it as a homograph IDN.

In total, we identified 1,855 homograph IDNs that impersonate 674 popular domain names. The top five most impersonated targets were `amazon.com`, `google.com`, `paypal.com`, `canva.com`, and `gmail.com`. For example, `xn--gmal-spa.com` (`gmail.com`) impersonates `gmail.com`.

Testing Homograph IDNs against Browsers. Considering the potential risk of homograph IDNs, browsers have implemented defense mechanisms. For example, when users visit a homograph IDN, the browser will no longer display the Unicode in the address bar. Instead, the corresponding Punycode is displayed to protect users from potential deception. To understand the efficacy of browsers' IDN policies, we tested the 1,855 real-world homograph IDNs by displaying them in the recent Chrome 81.0, Safari 13.0, and Firefox 75.0. Displaying a Punycode means browsers can successfully detect and block the homograph IDN.

We find that Chrome displays Punycode for 1,189 homograph IDNs (64.1%); Safari and Firefox only display Punycode for 180 (9.7%) and 113 (6.1%) of them. Chrome's defense is stronger than that of Safari and Firefox. But even so, Chrome has missed 35.9% of the homograph IDNs (more than one third).

Note that our finding is slightly different from an earlier study from 2018 [37] which showed Chrome's defense was effective against homograph IDNs discovered at that time (100% detection rate). Our results indicate that attackers have already exploited new ways to construct homograph IDNs to bypass existing browser policies.

4 Testing IDN Policies in Browsers

To understand the reasons behind the above observation, we want to take a closer look into the major browsers' IDN defense policies, and build testing cases to systematically evaluate them. This current section (Section 4) will be focused on browser policies and constructing test cases. In Section 5, we will present our testing results on the latest browsers and their historical versions, and examine the longitudinal browser policy changes.

4.1 Browsers' IDN Policies

To understand how major browsers handle IDNs, we first select a set of popular browsers based on their current and historical market shares [46, 60, 69]. We choose Chrome, Safari, Firefox, IE, and Windows Edge to analyze their publicly-available documentations and compare their *claimed* IDN policies. Table 2 summarizes representative policies. Different browsers may share the same high-level policies (*e.g.*, prohibiting script mixing) but implement them differently (*e.g.*, by defining different mixing rules). Below, we discuss each browser's policies in detail.

[1] Verisign Labs have made their datasets open to researchers: `https://www.verisign.com/en_US/company-information/verisign-labs`

Policy	Chr.	Fir.	Saf.	IE	Edge
P1: Unicode script mix (blocked)	✓	✓			✓
P2: Unicode script mix (allowed)	✓	✓		✓	✓
P3: Skeleton rule (top domain)	✓				✓
P4: Confusable chars (blocked)	✓				✓
P5: Whole-script + allowed TLD	✓	✓			✓
P6: Unicode script (allowed)			✓		

Table 2: The claimed policies of different browsers based on public documentations.

Chrome. For Chrome, we focus on the main policies related to IDN homograph and omit those related to IDNA implementations [18]. First, Chrome defines policies to allow and disallow certain characters from different Unicode scripts to be mixed in a single domain name (P1 and P2). For example, Latin, Cyrillic or Greek characters cannot be mixed with each other. This is to prevent homograph IDNs such as "apple.com" where Cyrillic "a" (U+0430) is used to mix with other Latin characters. Latin characters in the ASCII range can be mixed only with Chinese, Japanese and Korean; Han can be mixed with Japanese and Korean. Such script mixing is allowed because these blocks rarely contain look-alike characters.

Second, Chrome will compare the "skeleton" of the IDN with top domain names[2] (and domain names recently visited by the user). The skeleton is computed, for example, by removing diacritic marks (googlé.com with "é" replaced by "e"). This rule is called *skeleton rule* (P3). Using the skeleton rule, googlé.com will be flagged due to its high similarity with the popular domain name google.com.

Third, if an IDN contains mixed scripts and also confusable characters or dangerous patterns, Chrome will display Punycode (P4). For example, this rule disallows U+0585 (Armenian Small Letter Oh "*o*") and U+0581 (Armenian Small Letter Co "*g*") to be next to Latin due to their similarity to the Latin letters o and g.

Finally, P5 is used on domain names of whole-script confusables. Whole-script confusable means the domain name does not have mixing characters from different scripts. Instead, all the characters are from a single script but they look similar with ASCII letters. In this case, Chrome will check if the TLD is allowable. For example, attackers can construct www.apple.com (xn--80ak6aa92e.com) where the domain name only contains Cyrillic characters. In this case, since TLD ".com" is not Cyrillic, it will be flagged by this rule.

Firefox. Firefox's policies [45], as shown in Table 2, are different from those of Chrome. For example, Firefox does not have the skeleton rule to compare the IDN with popular domain names. Before 2012, Firefox only allowed IDNs with certain TLDs to be displayed in Unicode. However, as ICANN continues to release new TLDs, this approach becomes burdensome because Firefox has to constantly update the list. After 2012, Firefox added the mixing script rules. The idea is similar to that of Chrome, but the rules define different allowed/disallowed script combinations. For example, Firefox allows "Latin + Han + Hiragana," "Latin + Han + Bopomofo," "Latin + Han + Hangul," and "Latin + any single other Recommended/Aspirational scripts except Cyrillic or Greek."

Safari. Based on a security update in 2016 [5], Safari maintains a list of allowed scripts. Any IDNs containing scripts that are not on the allowed will be displayed in Punycode. This is an aggressive policy since it excludes the entire Unicode blocks such as Cherokee, Cyrillic, and Greek that might contain Latin look-alike characters. The goal is to prevent homograph IDNs such as "apple.com" (where Cyrillic "a" (U+0430) is used).

Internet Explorer (IE). IE only allows ASCII characters to be mixed with a predefined set of scripts that are unlikely to have confusable characters [41].

Microsoft Edge. Edge has two generations. For the legacy Edge (based on EdgeHTML), we cannot find any public documentations on their IDN policies. The new generation of Edge is based on Chromium. We assume Edge Chromium has the same policy as Chrome (as marked in Table 2) and will run experiments in Section 5 to validate this assumption.

User Configurations. Certain browsers allow user configurations. For example, Firefox allows users to disable IDN altogether and always display Punycode [17]. For IE, user-configured "accept language" can affect the IDN display. For example, if an IDN contains characters that are not part of the "accept language," IE will display the Punycode [41].

4.2 Building Testing Cases

Next, we design testing cases to systematically evaluate browsers' IDN policies. We focus on two main aspects: 1) we design cases to test the implementation correctness of the rules in the claimed policies; 2) we design cases that are likely to bypass known policies. We seek to test a number of browsers (of different versions, across different platforms) to understand how the policy implementations evolve over time.

As shown in Table 3, we develop 10 categories of testing cases. For each category, we construct about 1,000 testing IDNs[3]. After generating the IDNs, we then remove the *live domains* to 1) avoid disruptions to these live domains; 2) to improve the speed and stability of large-scale testing (*i.e.*, live domains take a much longer time to resolve and display). We have verified that all browsers will execute the same policies

[2] Chrome has a hard-coded list of top domain names. According to the source code of Chromium, there are 5001 top domain names on the list.

[3] When constructing IDNs for a given category, we try to identify all the relevant Unicode blocks, and then randomly sample the same number of characters from each block. Sometimes, we do not get exactly 1,000 IDNs because 1,000 cannot be divided evenly by the number of related Unicode blocks or there are not enough qualified characters.

Category	Description	Policy	Example IDNs	# Testing IDNs
Test-1	Mixing Latin, Cyrillic and Greek characters	P1	`āἀ.com, aaἀ.com`	1,000
Test-2	Mixing Latin any other Unicode blocks	P2	`cɯ.com, c0.com`	1,442
Test-3	Whole-script-confusables and TLD	P5	`xjö.net, jҳ.com`	997
Test-4	"Dangerous" patterns and Unicode confusables	P4	`b一.com, bノb.com`	1,090
Test-5	Skeleton rules (top-ranked domains)	P3	`healtn.com, jetbluɵ.com`	978
Test-6	ASCII look-alikes	P6	`ἰ.com, һ.com`	166
Test-7	Extended confusable characters	P4	`fastcᴏmpany.com, gamèrsky.com`	493
Test-8	Skeleton rules (low-ranked domains)	P3	`uṡa.gov, princetoŋ.edu`	1,600
Test-9	Full-substitution of all characters in a domain name	P1	`ḿỏỏ.bot, ịḏệỏ.com`	873
Test-10	Mixing extension blocks of Latin, Cyrillic and Greek	P1	`àḷ.net, ṽçǵ.com`	880
Total				9,519

Table 3: Testing cases and their related browser policies (the list of browser policies is in Table 2).

regardless of whether the domain is live or not. We in total obtain 9,519 IDNs as testing cases.

Testing the Claimed Policies Directly. As shown in Table 3, categories 1–6 are designed to directly test the claimed policies to examine if they are implemented correctly. Each testing category is mapped to a policy in Table 2. We do not plan to test user configurations since they depend on user preference. These testing cases are focused on testing the claimed rules instead of aiming for high-quality impersonations.

- **Category 1.** Most browsers do not allow the mixing between Latin, Cyrillic and Greek characters (P1). To test this rule, we construct IDNs that consist of mixing characters randomly sampled from Latin, Cyrillic, and Greek Unicode blocks (17 blocks in total). We randomly sample 2 characters from each of the 17 Unicode blocks to generate the 1,000 mixing-script IDNS (covering 4 types of combination: Latin + Cyrillic, Latin + Greek, Cyrillic + Greek, Latin + Cyrillic + Greek).
- **Category 2.** Chrome and Firefox claim to allow Latin characters to be mixed with Chinese, Japanese and Korean (CJK) characters (P2). However, it is not clear if other combinations are allowed. We construct IDNs that mix Basic Latin and 172 other non-CJK Unicode blocks. By randomly sampling 3 characters per block, we mix them to generate 1,442 testing IDNs.
- **Category 3.** This category is designed for whole-script confusable domain names, *i.e.*, all the characters are from a single look-alike script without any mixing (P5). To test this rule, we construct 997 IDNs using whole-script confusables from Cyrillic as domain names combined with non-Cyrillic TLDs (3 ASCII TLDs `.com`, `.net`, `.org` and 2 IDN TLDs .닷컴, .ею).
- **Category 4.** Chrome claims that if the IDN matches some dangerous patterns, it will display Punycode. The dangerous patterns include certain Japanese characters that can be mistaken as slashes, certain Katakana and Hiragana characters that look like each other. It is also not allowed to use U+0307 (dot above) after 'i', 'j', 'l' or dotless 'i' (U+0131). We construct 1,090 testing cases to cover all the documented rules.
- **Category 5.** This category is used to test the skeleton rule (P3). Chrome checks whether the domain name looks like one of the top-ranked domains, after mapping each character to its spoofing skeleton. Chrome uses Unicode official confusable table [65] and 31 additional confusable pairs to map a spoofing character to its ASCII skeleton. We use the same confusable pairs to construct 978 homograph IDNs.
- **Category 6.** Safari claims to only allow scripts that do not have ASCII look-alikes (P6). For this category, we randomly pick characters from Cyrillic, Greek and Cherokee Unicode blocks (without any mixing) to form 166 IDNs.

Testing to Bypass the Claimed Policies. Next, we assume all the claimed policies are correctly implemented. Under this assumption, we construct IDNs that are likely to bypass existing policies. For these testing cases, we explicitly construct *homograph IDNs* that impersonate target domains.

- **Category 7.** Given the possibility that the Unicode confusable table used by browsers is incomplete, we test to use a more comprehensive confusable database provided by researchers [61]. We generate 493 homograph IDNs to impersonate 200 domains sampled from Alexa top 10K [1].
- **Category 8.** The skeleton rule is currently applied to 5K popular domain names. However, many important websites are not necessarily "popular" (*e.g.*, based on traffic volume). For example, websites of governments, military agencies, educational institutions, regional hospitals, and other organizations may have a high phishing value but are not necessarily ranked to the top. To explore this idea, we construct homograph IDNs for `.gov`, `.mil`, `.edu`, `.org` and `.net` target domain names[4] that are not in the top 5K domain list.
- **Category 9.** In this category, we test whole-script confusables beyond Cyrillic. We use extended sets of confusable

[4] Registering `.gov`, `.mil`, and `.edu` domain names requires additional verification. However, anecdotal evidence shows such verification can be abused or bypassed by attackers to obtain these domain names [32]. Domain names under `.org` and `.net` are open to the public for registrations.

scripts to construct homograph IDNs without mixing. We randomly sample 200 target domains from Chrome's top domain list, and generate up to 5 all-substitution homograph domains for each target domain. We also keep the original TLDs unchanged.

- **Category 10.** Most browsers prohibit the mixing between Latin, Cyrillic and Greek. However, each script has multiple Unicode blocks, and it is not clear we can mix different blocks under the same script. For example, Latin has at least 9 blocks including Basic Latin, Latin Extended-A to E (5 blocks), IPA Extensions, Latin Extended Additional, and Latin-1 Supplement. We want to understand, for instance, if Latin Extended-A and Latin Extended-B can be mixed. We construct 880 IDNs using characters within Latin look-alike Unicode blocks. All the IDNS are homograph domains impersonating 200 domain names randomly sampled from Alexa top 1 million list [1].

Biases and Limitations. Our testing cases are designed to identify the problems with existing IDN policies. Certain policies are designed at the Unicode block level (P1, P2, P6). From each related block, we randomly select a few characters and exhaustively test their combinations. As such, the testing result is representative because these policies make decisions at the block level. For policies that are concerning the character level (*e.g.*, P3, skeleton rule), we randomly sample popular target domains and search for confusable characters. This does not guarantee completeness (we do not cover all target domain names). Exhaustive testing at the character level is difficult to finish within a reasonable amount of time.

Note that for test categories 1–4, and 6, the character replacement does not attempt to use look-alike characters since the policies are about allowable Unicode blocks. Categories 5 and 7–10 use look-alike characters. Due to the space limit, we make the list of testing IDNs available under this link[5].

5 Measurement Methods and Results

With the testing cases, we present our empirical experiments on major browsers and their historical versions to understand the effectiveness of IDN policies. We test historical versions for two reasons. First, it helps us to understand how different policies and their implementations evolve over time. Second, many users and organizations are still using outdated browsers [6] – their IDN policies are worth investigating.

We design experiments to answer four key questions. *First*, how well do browsers enforce known IDN policies? *Second*, how effective are existing policies in detecting homograph IDNs that impersonate target domains? *Third*, how are browser defenses changing over time?

Desktop (Total # of Versions)	**Version Range**
Chrome (21)	43.0 – 81.0
Firefox (15)	54.0 – 75.0
Safari (4)	10.0 – 13.0
Edge Legacy (4)	15.0 – 18.0
Edge Chromium (2)	80.0 – 81.0
IE (4)	8.0 – 11.0
Mobile (Total # of Versions)	**Version Range**
Android Chrome (7)	5.0 – 9.0
iOS Safari (13)	10.2 – 13.2

Table 4: Tested browsers and their versions.

5.1 Testing Platform and Methods

Browser Versions. We performed the experiments during April – May in 2020. The browser versions are shown in Table 4. We have primarily focused on Chrome, Safari, Firefox and Microsoft Edge. Note that Microsoft has stopped IE at its last version at v11.0 in 2016 [42], and continued with the new Microsoft Edge browser. For completeness (and considering users may use outdated browsers [6]), we have tested the legacy versions of IE too. For mobile browsers, we have tested Android Chrome and iOS Safari across their latest and historical versions.

Regarding the historical versions, we did not start from a browser's first version because most browsers did not support internationalized domain names in the beginning. Without IDN support, there is no point to test IDN defense policies.

Testing Method. We run black-box testing on each browser. By loading the testing cases (*i.e.*, IDNs), we examine whether the browser displays the Unicode or the Punycode. We control the browser to load the testing IDNs sequentially, and record a video to capture the screenshots of the browser. We choose to record a video (continuously) instead of taking screenshot images one by one to speed up the testing. Another advantage of screen recording is that it works across browsers and platforms. To help with the post-analysis of the recorded videos, we choose to load a special delimiter URL "`http://aaaaaa---{index}`" into the address bar between two consecutive testing IDNs. This `index` field is the index number of the next IDN to be tested. Using this delimiter, we can accurately split video frames and map them to the corresponding IDN (based on the index number).

In order to fully automate the tests, a key challenge is to configure the right environment for the browsers. For example, we need different desktop platforms (*e.g.*, Windows, Linux) and mobile platforms (*e.g.*, Android, iOS) to run the tests. In order to test historical versions, we need the right *legacy OS versions* to support outdated browsers. To solve this problem, we used a cloud-based testing framework called LambdaTest [34]. LambdaTest provides remote Selenium for desktop browsers and Appium for mobile browsers that can be controlled by our scripts via APIs. Before each test, we first specify the operating system name and the version via

[5] `https://github.com/stevetkjan/IDN_Testing/blob/master/testcases.xlsx`

a configuration file, and LambdaTest will automatically set up the testing virtual machine (VM) in the cloud. Our scripts then remotely control the browser running in the VM to load the list of IDN URLs one by one while recording the screen.

Video Analysis. The video analysis aims to determine whether a given IDN is displayed as Unicode (allowed) or Punycode (blocked) by the browser. First, we slice the video frames and map them to the specific IDN. As mentioned before, between two consecutive IDNs, we have loaded a delimiter. For example, delimiter “`aaaaaa---b16`” means the next video frames should be mapped to testing case #16 in category 2 (based on “`b`”). After slicing the video frames, we remove duplicated images based on perceptual hash (or phash) [3]. Given an image, we first crop the image to focus on the browser address bar. Then we apply OCR (Optical Character Recognition) to extract the URL in text format from the image. We use Google’s Tesseract OCR tool [19] which is known to have a good performance. If the extracted URL starts with “`xn--`”, then we determine it is a Punycode. We have taken extra steps to improve the accuracy of ORC, *e.g.*, by converting images into black and white, and improving the image resolution. To ensure the reliability of Punycode identification, we randomly sampled 100 images for each browser for manual validation. Across these browsers, we had a 0% false negative rate and a false positive rate below 2%. Our code is available here[6].

Extended Testing vs. Simplified Testing. We divide the testing into two phases. First, we run an *extended test* using all 9,519 testing cases on the latest versions of the browsers. Our goal is to understand the effectiveness of the current IDN policies. This test covers Chrome 81.0, Firefox 75.0, Safari 13.0, Edge Chromium 81.0, Android Chrome 9.0, and iOS Safari 13.2. This test does not cover IE or Edge Legacy since Microsoft has chosen Edge Chromium over the other two (we consider IE and Edge Legacy as historical browsers). Second, for all other historical versions, we run a *simplified test* considering the scalability requirement for covering a large number of browsers versions on different platforms. We sample about 10% of the testing cases for each category. For certain categories, the sampling rates are slightly higher than 10% in order to cover all the relevant Unicode blocks. This test covers 1,027 IDNs in total.

Additional Validations on IDN Policy Execution. To ensure the validity of the testing results, we have performed further sanity checks on IDN policy executions. First, we confirm that IDN policies are hard-coded in the *client side*, *i.e.*, the policies are executed without querying any remote servers. We confirm this by manually reading the Chromium code and running browsers in a sandbox to analyze the network traffic. This ensures the testing results do not depend on external services (*e.g.*, remote blacklists). Second, by monitoring the network traffic, we find that IND policies are triggered (*e.g.*, displaying Punycode) before the browser queries DNS. This ensures we can test the IDN policies without using resolvable domain names. Third, browsers will display the Punycode after an IDN policy is triggered[7]. Chrome is the only browser that has an additional warning page for “look-alike URLs” [9]. This warning only applies to IDNs that look like a predefined set of popular domain names (P3 “skeleton-rule”). We will further discuss this warning mechanism later in Section 7.

	Chrome	Firefox	Safari	Edge
Unicode	1,963	4,233	4,085	1,963
Failure Rate	20.62%	44.46%	42.91%	20.62%

Table 5: Testing results of the latest browsers. In total, 9,519 IDNs are tested per browser. We report the number of IDNS displayed as Unicode (*i.e.*, IDNs that browsers fail to block).

5.2 Results: Desktop Browsers

We start with the latest versions of desktop browsers. In Table 5, we report a *failure rate* which is the ratio of IDNs that are displayed as Unicode over all the tested IDNs. Displaying Unicode indicates that the browser has failed to block the IDN. In Figure 1, we show the failure rate for each testing category. Note that the failure rate has slightly different meanings for categories 1–6 and 7-10. For categories 1–6, it means the browser does not fully execute the claimed policies, which gives attackers the *opportunity* to create homograph IDNs. For categories 7-10, since all the testing IDNs are already homograph IDNs, the failure rate indicates risks more directly.

Chrome and Edge (Chromium). The first observation is Chrome and Edge have identical numbers in both Table 5 and Figure 1. This indicates Edge has the same polices as Chrome due to the use of Chromium. As such, we use Chrome as an example to discuss them together.

Table 5 shows that Chrome has the strictest policies compared to Firefox and Safari. Only 1,963 out of 9,519 IDNs (20.62%) are displayed in Unicode by Chrome. Noticeably, Chrome (and all other browsers) has a failure rate of 0% under category-1 (Figure 1). It means browsers enforced the rule to prevent the mixing of Latin, Cyrillic or Greek characters.

However, for the other nine categories, Chrome’s failure rate is non-zero. The result of categories 2–6 suggests that Chrome does not fully enforce the rules as claimed. Category-3 has the highest failure rate (85.3%). It turns out that Chrome allows whole-script confusables from Cyrillic to be combined with common TLDs such as `.com` and `.net`. The other 14.7% IDNs in category-3 are blocked because they have triggered other rules (*e.g.*skeleton rule). The results in categories 4 and

[6] https://github.com/stevetkjan/IDN_Testing

[7] Note that other types of domain squatting (*e.g.*, typo-squatting [4], combo-squatting [29]) do not trigger such Punycode displaying since these squatting domains do not use Unicode characters.

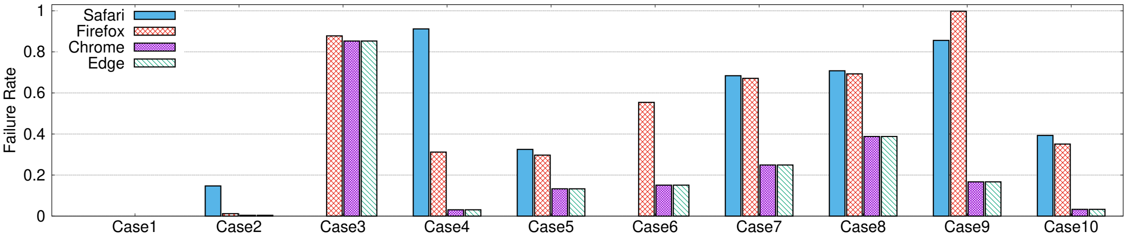


Figure 1: Failure rates of the 10 testing categories for the latest version of four browsers.

6 indicate Chrome does not fully cover Unicode confusables in the Unicode documentation and all the ASCII look-alike scripts. Category-5 has a failure rate of 13.3% (skeleton rule), indicating the skeleton comparison cannot perfectly capture all the confusable characters in the top domains.

For categories 7–10, the results confirm that our strategies to bypass Chrome policies are largely successful. In category-7, by using a more extensive confusable character table, we can cause more failures to the skeleton rule. In category-8, we focus on target domain names that are not in the top domain list (*e.g.*, those under `.edu`, `.gov`, `.mil`, `.org`, and `.net`), and Chrome fails to capture 40% of the homograph IDNs. Certain Unicode blocks are consistently missed. For example, when using confusable characters from the "Latin Extended-A" to impersonate these domain names, the failure rate is 100%. For categories 9 and 10, while the failure rates are lower, the results still indicate there are exceptions in the current mixing rule. For example, full-Substitution with "Latin Extended-A" causes a 100% failure rate, followed by a full-Substitution with "Cyrillic." Also, certain blocks within the Latin can be mixed without alerting Chrome (*e.g.*, mixing "Latin Extended-A" and "Latin-1 Supplement").

Safari. Safari has a failure rate of 42.91% overall. Compared to Chrome, Safari does not implement as many rules. For the rule that Safari did implement (*e.g.*, the rule corresponds to category-6), Safari does not make any mistakes. In addition, Safari blocks all the IDNs in category-1 (mixing script) and category-3 (whole-script Cyrillic). This is because Safari's allowed scripts have already excluded Latin look-alike scripts such as Greek and Cyrillic. Even so, it is still feasible to create homograph IDNs to bypass Safari. As shown in Figure 1, Safari has a failure rate over 60% on the homograph IDNS in categories 7, 8 and 9.

Firefox. Firefox has a higher failure rate of 44.46% among tested browsers. In particular, Firefox does not implement the skeleton rule, and thus the corresponding testing cases (categories 5, 7, 8) all have relatively higher failure rates.

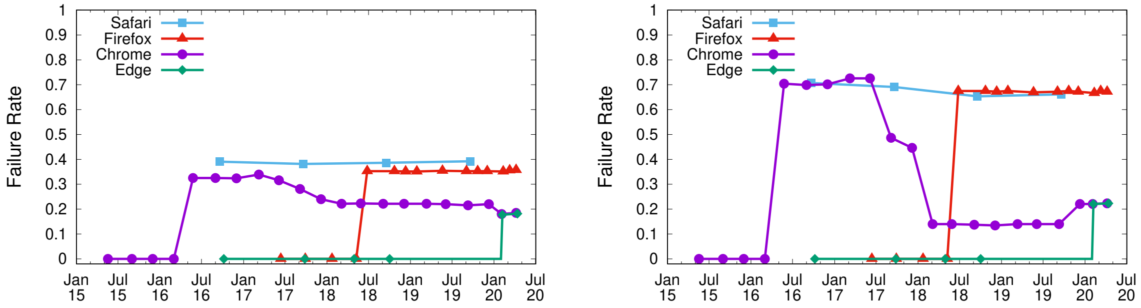
Case Studies. So far, we have discovered several strategies to bypass existing IDN policies. Some of the strategies are more useful than others to craft high-quality homograph IDNs. To illustrate the differences, in Table 6, we present example homograph IDNs crafted for Chrome, based on the mistakes Chrome made in each category (except for category-1 where Chrome has no failure). We find that it is easy to craft homograph IDNs for categories 3, 5, 7 and 8. For category-4, most of the dangerous patterns are mimicking non-English letters and symbols (such as slash). This limits the ability to generate homograph IDNs. For category-6, although we have found a large number of individual characters from different Unicode scripts missed by Chrome, it is not easy to craft high-quality homograph IDNs due to other rules (*e.g.*, non-mixing rules, skeleton rules). For categories 9 and 10, although we can easily find homograph IDNs, the IDNs need to be whole-script (*i.e.*, all the characters need to be replaced), and thus might sacrifice the quality of impersonation. Overall, the exception rules identified for categories 3, 5, 7 and 8 are the most effective ways to craft homograph IDNs.

Category	Example IDNs to Bypass Chrome
Test-2	`iú.edu, huĺu.com`
Test-3	`цcӏa.edu, цsçis.gov`
Test-4	`openaı.com, āi.google`
Test-5	`iħstagram.com, weılsfargo.com`
Test-6	`ţꞑţ.com, ңƀꝑ.com`
Test-7	`tumbır.com, ħyc.gov`
Test-8	`defẹnse.gov, pénnsylvania.gov`
Test-9	`ħŷć.gov, ħŷċ.gov`
Test-10	`ĵęţ.com, ǰđ.id`

Table 6: Example homograph IDNs that can bypass Chrome's policies to be displayed in Unicode.

5.3 Results: Mobile Browsers

We perform the same analysis on the mobile browsers including Android Chrome and iOS Safari. After analyzing their latest and historical versions, we find that the results are exactly the same as the corresponding web versions (Chrome and Safari). As such, we use "Chrome" and "Safari" to represent both the web and mobile versions. Note that mobile browsers present additional challenges for users to recognize web domain names due to the limited screen size. Some mobile browsers would only display part of the URLs or even hide the whole URLs in the address bar [38,39], which heightens the security risks. The user interface (UI) design, however, is beyond the scope of this paper.



(a) Testing categories 1–6

(b) Testing categories 7–10

Figure 2: Failure rates over time for different browser versions from January 2015 to April 2020.

5.4 Browser Policy Changes Over Time

Next, we analyze the historical browser versions to understand how their IDN policies change over time. Given a browser, we sort all its versions by the release dates. Then we select the most updated version for each quarter (4 quarters per year) to report their failure rates. As shown in Figure 2, we break down the results for categories 1–6 (Figure 2(a)) and categories 7–10 (Figure 2(b)) since their failure rates have different meanings. We have merged the curve for Edge Chromium and Edge Legacy since their release times do not overlap. We have also tested IE, but all the testing cases are displayed as Punycode. As such, we omit IE from Figure 2 for brevity.

Overall, most browsers follow a similar trend. *First*, the failure rates were initially at 0% because the browser did not support IDN yet in the early versions. All the testing IDNs are displayed as Punycode. These include Chrome browsers before version 51.0 (released in June 2016), Firefox browsers before version 61.0 (released in June 2018), and Edge browsers before 80.0 (released in February 2020). *Second*, once the browser started to support IDNs, the failure rates immediately jumped to a high level due to a lack of defense policies. *Third*, for browsers such as Chrome and Safari, their failure rates were gradually decreasing afterward as browsers added new IDN policies. For example, starting in March 2017, Chrome had a series of updates that significantly decreased the failure rate (mostly for categories 2, 5, 8, 9, and 10). In comparison, Firefox's failure rate has stayed at a similar level, indicating fewer or no updates of its IDN policies. As mentioned before, Edge changed to use Chromium in early 2020, and has followed Chrome's IDN policies since then.

One interesting observation (see Figure 2(b)) is that Chrome's failure rate went higher at the end of 2019, indicating certain policies were revoked. A further inspection shows the blocking decisions on many testing cases in categories 5, 7 and 8 were reversed — the new Chrome version re-allowed certain homograph IDNs to be displayed as Unicode. These re-allowed homograph IDNs contain characters from three main Unicode blocks: "Latin Extended-A," "Latin Extended-B," and "Latin-1 Supplement". Homograph IDNs such as `aŕmy.mil`, `yaĺe.edu`, `uchìcago.edu`, `canoń.com`, and `babblę.com` can be displayed in the updated Chrome even though they were blocked by earlier versions. The reasons behind this are not clear. We have checked the Chromium commit histories but did not find the information that can explain the reasons behind these changes. If they were not implementation errors, one possible explanation is that blocking these characters might hurt legitimate domain names with such characters.

6 User Study

We have shown that web browsers cannot block all the homograph IDNs. Next, we present a user study to examine how end users perceive the homograph IDNs in web browsing. In particular, we want to compare the homograph IDNs that browsers (*e.g.*, Chrome) block and those that can bypass existing policies. We focus on Chrome in this user study because Chrome has the strictest policies compared to other browsers. Our study aims to answer three research questions:

- **RQ1:** Would users fall for homograph IDNs (*i.e.*, incorrectly treating them as the real domain names)?
- **RQ2:** Would users have different rates of detecting IDNs that are blocked vs. not blocked by Chrome?
- **RQ3:** What factors are associated with users' rates of detecting IDNs? (association rather than causality)

6.1 Study Design

To answer these questions, we conducted an online experiment via Amazon Mechanical Turk (MTurk). Our study was approved by the IRB. The participation of the study was anonymous and voluntary. We also did not collect any personal identifiable information (PII) from the participants. Participants can choose to withdraw their data at anytime.

We presented the study as a generic survey on web browsing. We did not mention "security" or "phishing" in the study description to avoid priming users. Before the study started, we gave participants a short tutorial to explain what "domain name" and "browser address bar" are to ensure they can understand our terminology in the study. Upon finishing the study, we debriefed the participants by providing detailed explanations for the specific purpose of the study, and information on how homograph IDN is used for phishing[8].

Each participant was asked to browse a series of screenshots of website landing pages. As shown in Figure 3, a screenshot contains both the address bar[9] and the real landing page. Some of these screenshots impersonated domain names with homograph IDNs (e.g., `www.bankofamerı̦ca.com` in Figure 3), while the rest showed the real domain names. To see whether people can detect homograph IDNs, for each screenshot, we asked the participant a question about the authenticity of the website.

A key challenge was to determine how to phrase the question to the participants. At a high-level, we need to make sure users are making decisions based on the controlled information (*e.g.*, whether the domain name is a homograph IDN). This means we need to draw users' attention to the address bar. At the same time, we also wish to avoid over-priming users which will likely make the study unrealistic. In practice, users are often caught off-guard by phishing websites when they are not paying attention. Thus over-priming users could over-estimate users' ability to detect security threats [59].

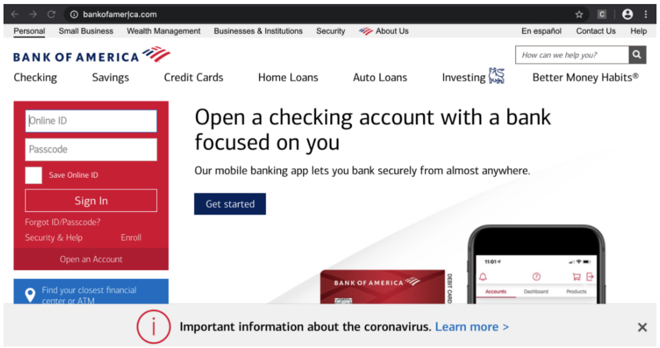
Pilot Studies. To explore the design space, we have conducted 4 pilot studies with 77 participants. We refer interested readers to Appendix-A for details. Here, we only briefly summarize our findings, and describe our final chosen design. The primary goal of the four pilot studies is to examine different priming levels. We use `bankofamerica.com` as an example:

At the *low priming level*, we showed the screenshot and asked "is the website the real `bankofamerica.com`?" We tried to avoid priming users to focus on the address bar.

At the *medium priming level*, we asked "is the domain name in the browser address bar `bankofamerica.com`?" By explicitly mentioning the address bar, we cued users to examine the address bar.

At the *high priming level*, we drew users' attention to the domain name even more by placing the homograph domain name directly in the question. We asked "is `bankofamerı̦ca.com` the same as `bankofamerica.com`?" We essentially asked the users to compare the two domain names side-by-side.

We also tested two different designs for the answering options. The first design is to use *binary answers*: participants can choose one from "Yes," "No," or "I can't tell." The second design is to use 5-point Likert scale answer options.



Open a checking account with a bank focused on you

Our mobile banking app lets you bank securely from almost anywhere.

Important information about the coronavirus. Learn more >

Figure 3: An example screenshot, which always shows the real webpage. The address bar was artificially added to display either the real domain name or a homograph IDN in Unicode (in this case, `www.bankofamerı̦ca.com`). Right below the screenshot, we asked *"Is the domain name in the browser address bar* `bankofamerica.com`*?"* Participants can choose one of three answers: *"Yes," "No," "I can't tell."*

Final Design. After comparing the results of the pilot studies, we decided to choose the *medium* priming level and *binary answer* (plus "I can't tell") as the final design. We asked "is the domain name in the browser address bar *[the real domain name]*?". Participants can choose one of three answers: "Yes," "No," "I can't tell." This is based on two reasons. First, we did not observe a need to use a 5-point Likert scale as the trend was the same for both conditions and using the Likert scale can complicate the tasks. People might also interpret the five levels differently. Instead, a binary answer (plus "I can't tell") can reduce the ambiguity. Second, the medium priming level (*i.e.*, mildly cuing users to check the address bar) is more suitable since our research questions are about domain names. The pilot studies show that users had a higher accuracy to label the domain authenticity under a higher priming level (see Appendix-A). While we use the medium priming level for our main study, the other pilot study results can serve as the lower/upper bounds of detection rates.

6.2 Main User Study

After determining the study design, we now introduce the setups of the main user study.

Websites. For the main user study, we use a diverse set of 90 websites. Out of the 90 target websites, 45 were from the Chromium top domain list (*i.e.*, "popular"), and the other 45 were not on the list (*i.e.*, "unpopular"). We select these websites from five common website categories (18 websites per category): "Shopping," "Banking," "Social Networking," "Education," and "Government & Military."

For each target website, we can choose to display the real domain name in the address bar of the screenshot ("Real"). We can also choose to display the homograph IDN to impersonate it. We consider two types of homograph IDNs: one IDN that

[8] After our study, we received messages from participants who thanked us for educating them about homograph IDNs.

[9] In the address bar of the screenshots, we always displayed the Unicode version of the homograph IDNs to examine how users perceive them and to fairly compare homograph IDNs missed by Chrome with the blocked ones. We wanted to understand whether the missed IDNs are more or less difficult to detect by users compared with the blocked ones in the Unicode format.

can be blocked by the latest Chrome (IDN-Block), and another IDN that can bypass Chrome's policy (IDN-Pass).

Out of the 90 websites, we set the ratio of "Real", "IDN-Block", and "IND-Pass" to be roughly 1:1:2. We included more IND-Pass domains because IDNs that can bypass Chrome's policies are less understood and studied. We covered more IDN-pass domains to better study this category. More specifically, we randomly chose 23 of the 90 websites to display the real domain names ("Real"), and select another 23 websites to display homograph IDNs that can be blocked by Chrome ("IDN-Block"). For the remaining 44 websites, we crafted homograph IDNs that would bypass Chrome's policies ("IDN-Pass"). A complete list of the websites and domain names is available here[10].

Factors. In addition to website category and popularity, we also considered other factors such as people's demographics (*e.g.*, age range, gender identity) and computing/Internet experiences (*e.g.*, years of using web browsers, computing background). These questions are available under this link[11].

Study Process. In April 2020, we conducted a study on MTurk. Each participant examined 30 websites. More specifically, we divided the 90 websites into 3 blocks (each block has 30 websites). In each block, the mixture ratio of "Real", "IDN-Block", and "IND-Pass" was still roughly 1:1:2. We randomly assigned each participant to one of the three blocks (each participant can work on one block only). Once the block was assigned, we presented a random order of the 30 sites in the block to the participant.

To ensure the reliability of results, we randomly selected one attention check question and inserted it in a random position in the task/question list. We have two attention questions to choose from. 1) "Is the domain name shown in the browser address bar a social networking website?" The screenshot shows the webpage of the Bank of America. 2) "Is the domain name shown in the browser address bar a hospital website?" The screenshot shows the webpage of Facebook. Both questions have the obvious answer "No." The attention questions were designed to help us filter out participants who did not look at the domain names or webpages and simply answered the subsequent question randomly. We also did not want the attention questions to prime the participants to pay more attention than they would otherwise. We found that these questions helped filter out several non-attentive participants.

To attract serious workers on MTurk, we used commonly applied filters: we recruited U.S workers who have an approval rate greater than 90%, and have completed more than 50 approved tasks. Each participant was compensated $1 for their time. The participants took 8 minutes on average to complete the study. The compensation was about $8 per hour.

Domain Type	Yes	No	I can't tell
Real	1,565 (**94.6%**)	86 (5.2%)	4 (0.2%)
IDN-Block	807 (48.8%)	803 (**48.5%**)	45 (2.7%)
IDN-Pass	1,353 (42.3%)	1,768 (**55.2%**)	79 (2.5%)

Table 7: Correct answer rates in the main study (6,510 answers): 94.6%, 48.5%, 55.2% for real, IDN-Block, IDN-Pass.

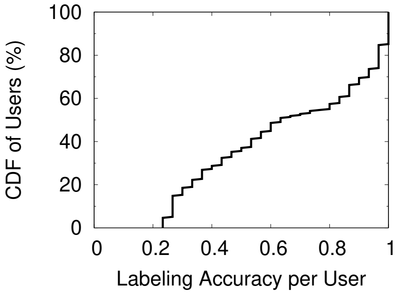


Figure 4: Cumulative distribution function (CDF) of labeling accuracy for each user.

Each worker can only participate in the study once. Pilot study participants were not allowed to take part in the main study, which had a total of 325 participants. After removing incomplete submissions and those who failed the attention check, we had 217 valid participants with 6,510 answers.

6.3 Overall Results

Table 7 shows the overall results of the main study. The results were consistent with the pilot studies. When the domain name was real, 94.6% of the answers were correct (by answering "YES"). In comparison, when the domain name was homograph IDN, only 55.2% of the answers were correct under IDN-Pass, followed by 48.5% under IDN-Block.

This result answers RQ1: our participants fell for a large percentage of homograph IDNs. We also examined how well individual participants correctly labeled the authenticity of websites based on the domain names. Figure 4 shows the cumulative distribution function (CDF) of each participant's labeling accuracy (based on the 30 websites the user has examined). All participants had an accuracy above 20%, and a small portion (15%) of them had a 100% accuracy. However, about half of the participants had an accuracy below 60%. Overall, the results suggest that the majority of users will struggle in correctly identifying homograph IDNs.

To answer RQ2, we then performed pair-wise comparisons between these three conditions using Chi-square tests with a Bonferroni correction (the adjusted p value threshold is .01). We found that the differences among these conditions were statistically significant: the correct answer rates for Real vs. IDN-Block ($\chi^2 = 859.3$, $p < 0.001$), Real vs. IDN-Pass ($\chi^2 = 782.7$, $p < 0.001$) and IDN-Block vs. IDN-Pass ($\chi^2 = 19.6$, $p < 0.001$). Comparing IDN-Block and IDN-Pass, we found that homograph IDNs blocked by Chrome were more deceptive (lower correct answer rate) than those not blocked.

[10] https://github.com/stevetkjan/IDN_Testing/blob/master/websites.xlsx

[11] https://github.com/stevetkjan/IDN_Testing/blob/master/Questions-IDN.pdf

Variable	**Coefficient**	**P-Value**
Domain Type: base = IDN-Block		
IDN-Pass	0.884	**0.006**
Real	3.441	<**0.001**
Website Category: base = Banking		
Education	0.415	0.199
Government & Military	0.301	0.287
Shopping	0.465	0.109
Social networking	0.496	0.088
Website Popularity: base = Popular		
Unpopular	-0.289	0.288
Browser experience: base = Short (<= 3 Yr)		
Long (> 3 Yr)	0.450	**0.001**
Computer background: base = NO		
YES	-0.371	<**0.001**
Gender: base = Female		
Gender: Male	0.235	<**0.001**
Age: base = Younger (<= 39)		
Age: Older (> 39)	0.133	**0.044**
Education: base = Lower (< Bachelor's)		
Higher Edu (Bachelor's or higher)	-0.823	<**0.001**

Table 8: Logistic regression results: using website and user factors to predict whether the authenticity of a website domain name was correctly labeled by a user.

However, it is alarming that 45% of un-blocked domain names (IDN-pass) were mistaken by our participants as real sites. Thus, they pose a substantial issue as about half of the times people fell for homograph IDNs not caught by Chrome.

6.4 Regression Analysis

To answer RQ3, we further analyzed the factors associated with user performance in detecting IDNs. We used the dataset of 6,510 answers and conducted logistic regression analyses in R to predict a binary outcome: whether the authenticity of a website domain name was correctly labeled by a user (*i.e.*, correct answers for Real and IDN-Block/Pass are "Yes" and "No," respectively). Table 8 shows the regression results.

Predictor Variables. The independent variables or predictors were all categorical variables, including the domain type, website category and popularity as well as people's demographics and computing experience.

We had three predictors related to websites. First, the domain types included Real, IDN-Pass and IDN-Block. We used IDN-Block as the baseline. Second, we had five website categories and hypothesized that the website category may affect users' judgment of the website authenticity. For example, users might be more likely to check the authenticity of banking websites than education websites. As such, we used banking websites as the baseline. Third, for website popularity, we hypothesized that users may perform better on popular websites since they might be more familiar with those. Thus, we used popular websites as the baseline.

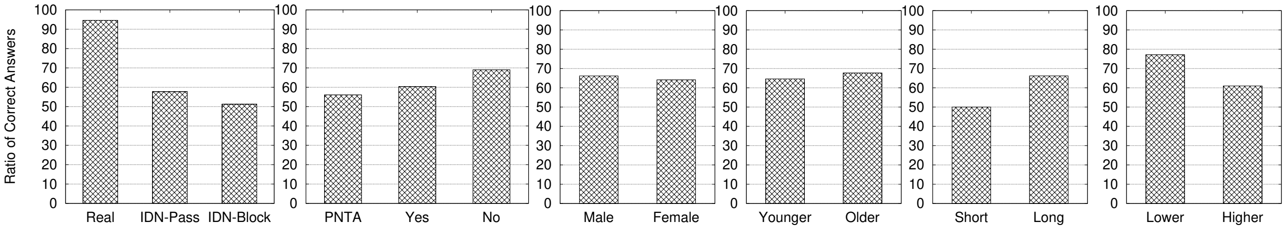
We had five predictors related to users. First, for users' years of experience using web browsers, we converted this variable to a binary variable: "short" and "long" using 3-year as a threshold. We chose this threshold by examining the sign of the regression coefficients of the original levels and found that 3-year was the level where the sign changed. To simplify our analysis, we applied this method in converting other user-related multi-level ordinal predictors to binary variables. We hypothesized that users with a long experience with web browsing may perform better in detecting IDNs. The second and third variables were users' computing background and gender identity. The fourth variable was age level, and we used a threshold of 39 to divide users into younger and older categories. The last user variable is education level, and we used "Bachelor's degree" to divide users into two levels.

Result Interpretation. As shown in Table 8, several factors were significantly correlated with users' performance in detecting IDNs. These results answer RQ3. Overall, we found that domain types and user-related factors were significantly associated with users' performance whereas website category and popularity were not. As a reference, in Figure 5, we further illustrate the raw percentage of correct answers for factors that have statistical significance.

First, the *domain type* results imply that participants were significantly more likely to label the real domains and IDN-Pass domains correctly compared to the baseline (IDN-Block). The result is consistent with that in Table 7. More specifically, Real has a β estimate of 31.23, which means the odds of labeling real domains correctly is $exp(3.441) = 31.23$ times of that of labeling IDN-Block correctly. Similarly, the odds of labeling IDN-Pass correctly is $exp(0.884) = 2.42$ times of that of labeling IDN-Block correctly. These results further confirm that Chrome indeed blocked the homograph IDNs that are more deceptive to users than IDNs that were not blocked (IDN-Pass). This does not necessarily mean IDN-Pass is safe for users. As discussed in Section 6.3, homograph IDNs that bypassed Chrome policies are also highly deceptive. Unlike domain type, website category and popularity were not found to be significantly associated with user performance.

Second, we found several user factors were significantly correlated with correctly labeling the website authenticity. For example, the odds of correct labeling for users with a longer (3-year) web browsing experience is $exp(0.450) = 1.59$ times of that of users with a shorter experience. Similarly, users' frequency of visiting the five categories of sites were also significantly and positively correlated with user performance. Male participants did better in correctly labeling the domain names. However, as shown in Figure 5(c), the performance difference between male and female participants was rather small (but statistically significant). Older participants seemed to perform better than their younger counterparts. Again the difference was small but statistically significant.

Third, perhaps counter-intuitively, computing background and educational level were also significantly correlated with



(a) Domain Type (b) Computing Background (c) Gender (d) Age (e) Browser Experience (f) Education Level

Figure 5: The percentages of correct answers for different groups. We include the factors that have statistical significance in Table 8. "PNTA" under computing background stands for "Prefer not to answer."

user performance albeit negatively. As shown in Figure 5(b), the differences were relatively small (but statistically significant). Users with a higher educational level or computing background seemed to perform worse. While we do not know the reason, one plausible explanation could be that they were overly confident about their knowledge/skills and overlooked the IDNs. Future research can investigate the reason(s).

Limitations of The User Study. Our user study has many limitations. First, there is an inherent limitation for conducting the study online via MTurk since we cannot guarantee our participants always paid attention to our questions. However, recent studies show that MTurk workers are at least as attentive as subject pool participants [23]. Furthermore, our inclusion of attention questions ameliorated this concern. Future work could consider using eye-tracking to further address this limitation. Second, there were still differences between our study setup and real-world web browsing. In particular, we showed a screenshot of a target website, and thus participants cannot interact with the websites. The non-interactive screenshots helped to protect users and also allowed us to focus on the domain names rather than other user strategies. We also reminded our participants to pay attention to domain names, while in practice, users are likely to make more mistakes if they are not reminded (per the results of pilot studies). The point is that even though we primed them, they still cannot identify a significant percentage of homograph IDNs. This suggests we need to provide more countermeasures to help users. Finally, our user study only examines user perception of the authenticity of websites (domain names), which is only the first step in web phishing. Future work can study how IDNs affect their follow-up actions such as login.

7 Discussion

IDN Homograph in Email and Social Network Services. Email systems and social network platforms are also popular channels to disseminate phishing messages. In these applications, IDN homograph can be used to deceive users too. We briefly investigated popular email and social network services regarding their IDN policies. Our overall observation was that most services had not established effective IDN policies. Due to space limit, we only briefly summarize our findings.

For email services, we looked into Gmail, iCloud and Outlook. As of May 2020, we tested to see if homograph IDNs (that impersonate popular domain names) can be displayed on email clients in Unicode. For this test, we need to register the homograph IDNs and set up the DNS records. We registered 3 IDNs. The first IDN is "강남교회.com". This domain name represents a legitimate usage of IDN and it does not violate any known IDN policies. We use this IDN to test if the email clients support IDN. The second IDN is "googīe.com". This IDN impersonates "`google.com`". This IDN represents homograph IDNs that can bypass browser defense. The third IDN is "`þaidu.com`". This IDN impersonates "`baidu.com`". This IDN represents homograph IDNs that can be detected by browser policies (*e.g.*, by Chrome). We use those IDNs as email domain names and send emails to our own accounts under popular email services. We then examine if the sender email address will be displayed as Unicode. For email clients that supported IDN, we found the homograph IDNs were consistently displayed in Unicode. For example, Gmail (web and mobile) and iCloud (mobile) supported IDN and displayed homograph IDNs in Unicode in the email sender addresses. This means attackers can use homograph IDNs to impersonate trusted senders. This observation is true even for a homograph IDN that is blocked by web browsers. The results suggest that email clients have not yet established effective policies to address homograph IDNs.

For social network applications, we examined how homograph IDNs are displayed in messages and posts. As of May 2020, we tested Facebook, Twitter, Messenger, iMessage, and Whatsapp with homograph IDN URLs that impersonate popular brands. We found that almost all of them displayed homograph IDNs in Unicode, except for Facebook (which displayed Punycode). The result again suggests that most social network applications do not have IDN defense policies.

Intent of IDN Homograph. Our measurement shows that homograph IDNs exist in practice and can bypass browser-level defenses (Section 3). We did not further analyze the "intent" of homograph IDN registrations because the browser policies focus on the impersonation behavior rather than the intent. According to prior studies, many homograph IDNs are registered by opportunistic domain squatters who seek to sell the domain name to the target brand for profits [37]. Concrete evidence also shows homograph IDNs are often used for phishing and abuse [15, 37, 54]. We defer the more in-depth analysis of the registrants' intent to future work.

Countermeasures. Our results suggest the need to improve the defense against homograph IDNs. We discuss the improvement strategies from two aspects: homograph IDN detection, and user warnings.

To improve homograph IDN detection, one way is to add new rules to address the failure cases discovered by our experiments. For example, browsers such as Chrome can extend the list of target domains (*e.g.*, for skeleton rule), use a more comprehensive confusable table (instead of the standard Unicode confusable table), and increase the number of prohibited Unicode blocks. Even so, it is difficult for the rules to guarantee completeness. For example, the skeleton rule matches the IDNs against a list of top domain names which do not cover all the domains. To extend the list (*e.g.*, to cover all the domains), the immediate challenge is efficiency. Considering the browsers' need to make decisions in real-time, it is costly to check the visual similarity between the IDN and hundreds of millions of domain names. Improving the scalability of the skeleton matching is an open challenge for future work. In addition, stricter policy may also hurt legitimate IDNs that have mixed Unicode characters. As mentioned in Section 3, script mixing is also common among legitimate domains (34.4% of the IDNs have script mixing). As such, to add new script-mixing rules, browser vendors must carefully assess their impact on legitimate IDNs.

In addition to browsers, another line of defense is domain registrars. Since domain registrars do not have to approve domain registration in a real-time, it is more feasible for domain registrars to run an extensive visual comparison against existing domain names before a (homograph) IDN can be registered. Certain domain registrars already have some preliminary rules in place (*e.g.*, mostly to prevent script mixing) [2].

Related to IDN homograph detection, a key question is how to communicate the detection decision to end users (*i.e.*, user warnings). For example, besides showing the Punycode, Chrome also shows a warning page to ask users: "Did you mean *[the real domain name]*?" with a short explanation of the reason for showing the warning [9]. Such a warning page is not yet available in other browsers. Future research can examine the effectiveness of such warnings, and further explore the warning design space. Note that user warnings still depend on accurate detection methods — if the browser misses the detection of a homograph IDN, the warning page may not be triggered.

Finally, browsers may take a more extreme approach by disabling IDNs *by default*. For users who can benefit from the support of IDNs (*e.g*, users speaking certain languages), browsers can prompt users to enable the IDN support for a small set of related Unicode scripts.

Responsible Disclosure. We have reported our findings to the corresponding bug/security teams of Chrome, Safari and Firefox. Microsoft IE uses Chromium, and thus is also covered. Chrome and Firefox have started to investigate and address the reported issues. Safari's team has not responded.

8 Related Work

Domain Squatting. Domain squatting generally refers to the behavior of registering Internet domains with the names similar to existing brands and trademarks. While most domain squatting activities are driven by profits (*i.e.*, to sell the domain name to the target brand for a high price), some of the squatting domain names are found to be used for phishing or distributing malware [21, 47, 64]. To mimic a target brand, squatting domain names can be generated by creating a typo (*i.e.*, typo squatting) [4, 44, 62], flipping a bit (*i.e.*, bit squatting) [47], or using a hyphen to connect related keywords (*i.e.*, combo-squatting) [29]. Another strategy is to register lookalike domain names of popular brands under newly released TLDs [20,21,30,52]. A recent work shows that such impersonation also occurred in TLS certificates [58]. Our work focuses on IDN homograph, which is a form of web homograph via character substitution [25]. Although IDN homograph is not necessarily the most prevalent domain squatting method (*e.g.*, combo-squatting is more prevalent [64]), empirical results show that IDNs can be used to construct highly deceptive phishing websites [35, 37, 74]).

IDN Homograph. Prior works have looked into IDN homograph by conducting empirical measurements. Researchers find that many of the IDNs are owned by domain squatters [35, 37] while some IDNs are used for phishing and abuse [15, 37, 54]. A related project shows that most users do not have the knowledge of internationalized domain names [8], which helps to explain why IDNs can be deceptive. Compared to prior work [8, 35, 37, 61], our novelty comes from the detailed analysis of browser-level defense, and the discovered weaknesses of current IDN policies.

Phishing. Our work is related to the broad topic of phishing. A large body of prior work has looked into *phishing emails* and studied different detection methods [12, 14, 16, 26, 40, 53, 67]. Unlike generic spam emails [55, 70], phishing emails can be highly targeted and thus are more difficult to detect [24]. To deceive victims, attackers can spoof a trusted domain name as the sender email address [7, 27] or directly use squatting domain names [31, 33]. Our work is more closely related to *phishing websites*, which are usually the landing pages of the URLs in phishing emails [11, 22, 48, 50, 51, 68, 71–73]. A recent project looks into the end-to-end life cycle of phishing attacks by jointly analyzing phishing URLs, websites, and phishing emails [49].

Security Indicators on URLs. Researchers have examined how users perceive and react to different URL presentations in browsers under security contexts. Most studies have reported negative results. For example, a recent study shows HTTPS Extended Validation (EV) certificate has little impact on users' security behavior [63]. In addition, prior work shows that domain name highlighting has limited effectiveness in warning users about malicious URLs [13, 36]. A closely related project looks into how different URL obfuscation methods (including IND homograph) affect users' ability to judge the authenticity of URLs [57]. The overall results are consistent with ours, showing that users have difficulties to correctly recognize obfuscated URLs. Compared with [57], our user study further examines the differences in users' perceptions of homograph IDNs blocked by browsers and those that can bypass existing defenses. In addition, our results highlight the need to improve the *detection* of IDN homograph, which is the prerequisite for effective user warning.

9 Conclusion

In this paper, we present a detailed analysis of browsers' defense policies against IDN homograph. Using more than 9,000 testing cases, we measure the effectiveness of IDN policies in existing web and mobile browsers and their historical versions from 2015 to 2020. We show that browsers' IDN policies are not yet effective to detect homograph IDNs. Our user studies show that the homograph IDNs that can bypass browsers' defense are still highly deceptive to users. Overall, the results highlight the need to improve the defense policies.

Acknowledgment

We would like to thank our shepherd Nick Nikiforakis and anonymous reviewers for their constructive comments and suggestions. This work was supported in part by NSF grants CNS-2030521, CNS-1717028 and CNS-1652497.

References

[1] Alexa top 1 million websites. https://www.alexa.com/topsites.

[2] Idn registration rules of verisign, 2020. https://www.verisign.com/en_US/channel-resources/domain-registry-products/idn/idn-policy/registration-rules/index.xhtml.

[3] Perceptual hash, 2020. https://www.phash.org/.

[4] Pieter Agten, Wouter Joosen, Frank Piessens, and Nick Nikiforakis. Seven months' worth of mistakes: A longitudinal study of typosquatting abuse. In *Proc. of NDSS*, 2015.

[5] Apple. About safari international domain name support, 2016. https://support.apple.com/kb/TA22996?locale=en_US&viewlocale=en_US.

[6] Patricia Callejo, Rubén Cuevas, and Áangel Cuevas. An Ad-driven measurement technique for monitoring the browser marketplace. *IEEE Access*, 7, 2019.

[7] Jianjun Chen, Vern Paxson, and Jian Jiang. Composition kills: A case study of email sender authentication. In *Proc. of USENIX Security*, 2020.

[8] Daiki Chiba, Ayako Akiyama Hasegawa, Takashi Koide, Yuta Sawabe, Shigeki Goto, and Mitsuaki Akiyama. Domainscouter: Understanding the risks of deceptive IDNs. In *Proc. of RAID*, 2019.

[9] Catalin Cimpanu. Google chrome to get warnings for lookalike urls, 2019. https://www.zdnet.com/article/google-chrome-to-get-warnings.

[10] Adam Costello. Punycode: A bootstring encoding of unicode for internationalized domain names in applications (IDNA). RFC 3492, 2003. https://tools.ietf.org/html/rfc3492.

[11] Qian Cui, Guy-Vincent Jourdan, Gregor V. Bochmann, Russell Couturier, and Iosif-Viorel Onut. Tracking phishing attacks over time. In *Proc. of WWW*, 2017.

[12] Prateek Dewan, Anand Kashyap, and Ponnurangam Kumaraguru. Analyzing social and stylometric features to identify spear phishing emails. In *Proc. of eCrime*, 2014.

[13] Rachna Dhamija, J. D. Tygar, and Marti Hearst. Why phishing works. In *Proc. of CHI*, 2006.

[14] Sevtap Duman, Kubra Kalkan-Cakmakci, Manuel Egele, William K. Robertson, and Engin Kirda. Emailprofiler: Spearphishing filtering with header and stylometric features of emails. In *Proc. of COMPSAC*, 2016.

[15] Yahia Elsayed and Ahmed Shosha. Large scale detection of IDN domain name masquerading. In *Proc. of eCrime*, 2018.

[16] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In *Proc. of WWW*, 2007.

[17] Mattias Geniar. Show idn punycode in firefox to avoid phishing urls, 2018. https://ma.ttias.be/show-idn-punycode-firefox-avoid-phishing-urls/.

[18] Google. Internationalized domain names (IDN) in google chrome, 2020. https://chromium.googlesource.com/chromium/src/+/master/docs/idn.md.

[19] Google. Tesseract orc, 2020. https://opensource.google/projects/tesseract.

[20] Tristan Halvorson, Matthew F. Der, Ian Foster, Stefan Savage, Lawrence K. Saul, and Geoffrey M. Voelker. From .academy to .zone: An analysis of the new tld land rush. In *Proc. of IMC*, 2015.

[21] Tristan Halvorson, Kirill Levchenko, Stefan Savage, and Geoffrey M. Voelker. Xxxtortion? inferring registration intent in the .xxx tld. In *Proc. of WWW*, 2014.

[22] Xiao Han, Nizar Kheir, and Davide Balzarotti. Phisheye: Live monitoring of sandboxed phishing kits. In *Proc. of CCS*, 2016.

[23] D.J. Hauser and N. Schwarz. Attentive turkers: Mturk participants perform better on online attention checks than do subject pool participants. *Behavior Research Methods*, 48:400–407, 2016.

[24] Grant Ho, Aashish Sharma, Mobin Javed, Vern Paxson, and David Wagner. Detecting credential spearphishing in enterprise settings. In *Proc. of USENIX Security*, 2017.

[25] Tobias Holgers, David E. Watson, and Steven D. Gribble. Cutting through the confusion: A measurement study of homograph attacks. In *Proc. of USENIX ATC*, 2006.

[26] Jason Hong. The state of phishing attacks. *Communications of the ACM*, 55(1), 2012.

[27] Hang Hu and Gang Wang. End-to-end measurements of email spoofing attacks. In *Proc. of USENIX Security*, 2018.

[28] IETF.org. Internationalizing domain names in applications (IDNA), 2003. https://tools.ietf.org/html/rfc3490.

[29] Panagiotis Kintis, Najmeh Miramirkhani, Charles Lever, Yizheng Chen, Rosa Romero-Gómez, Nikolaos Pitropakis, Nick Nikiforakis, and Manos Antonakakis. Hiding in plain sight: A longitudinal study of combosquatting abuse. In *Proc. of CCS*, 2017.

[30] Maciej Korczynski, Maarten Wullink, Samaneh Tajalizadehkhoob, Giovane C. M. Moura, Arman Noroozian, Drew Bagley, and Cristian Hesselman. Cybercrime after the sunrise: A statistical analysis of dns abuse in new gtlds. In *Proc. of Asia CCS*, 2018.

[31] Viktor Krammer. Phishing defense against IDN address spoofing attacks. In *Proc. of PST*, 2006.

[32] Brian Krebs. It's way too easy to get a .gov domain name, 2019. https://krebsonsecurity.com/2019/11/its-way-too-easy-to-get-a-gov-domain-name/.

[33] Ponnurangam Kumaraguru, Yong Rhee, Alessandro Acquisti, Lorrie Faith Cranor, Jason Hong, and Elizabeth Nunge. Protecting people from phishing: The design and evaluation of an embedded training email system. In *Proc. of CHI*, 2007.

[34] LambdaTest. Lambdatest: Cross browser testing cloud, 2020. https://www.lambdatest.com/.

[35] Victor Le Pochat, Tom Van Goethem, and Wouter Joosen. Funny accents: Exploring genuine interest in internationalized domain names. In *Proc. of PAM*, 2019.

[36] Eric Lin, Saul Greenberg, Eileah Trotter, David Ma, and John Aycock. Does domain highlighting help people identify phishing sites? In *Proc. of CHI*, 2011.

[37] Baojun Liu, Chaoyi Lu, Zhou Li, Ying Liu, Hai-Xin Duan, Shuang Hao, and Zaifeng Zhang. A reexamination of internationalized domain names: The good, the bad and the ugly. In *Proc. of DSN*, 2018.

[38] Meng Luo, Pierre Laperdrix, Nima Honarmand, and Nick Nikiforakis. Time does not heal all wounds: A longitudinal analysis of security-mechanism support in mobile browsers. In *Proc. of NDSS*, 2019.

[39] Meng Luo, Oleksii Starov, Nima Honarmand, and Nick Nikiforakis. Hindsight: Understanding the evolution of ui vulnerabilities in mobile browsers. In *Proc. of CCS*, 2017.

[40] D. Kevin McGrath and Minaxi Gupta. Behind phishing: An examination of phisher modi operandi. In *Proc. of LEET*, 2008.

[41] Microsoft. Changes to IDN in IE7 to now allow mixing of scripts, 2006. https://docs.microsoft.com/en-us/archive/blogs/ie/changes-to-idn-in-ie7-to.

[42] Microsoft. Lifecycle FAQ - Internet explorer and Edge, 2016. https://docs.microsoft.com/en-us/lifecycle/faq/internet-explorer-microsoft-edge.

[43] Paul Mockapetris. Domain names - concepts and facilities. RFC 1034, 1987. https://tools.ietf.org/html/rfc1034.

[44] Tyler Moore and Benjamin Edelman. Measuring the perpetrators and funders of typosquatting. In *International Conference on Financial Cryptography and Data Security*, 2010.

[45] Mozilla. Firefox IDN display algorithm, 2017. https://wiki.mozilla.org/IDN_Display_Algorithm.

[46] NetMarketShare. Browser market share, 2020. https://netmarketshare.com/browser-market-share.aspx.

[47] Nick Nikiforakis, Steven Van Acker, Wannes Meert, Lieven Desmet, Frank Piessens, and Wouter Joosen. Bitsquatting: Exploiting bit-flips for fun, or profit? In *Proc. of WWW*, 2013.

[48] Adam Oest, Yenganeh Safaei, Penghui Zhang, Brad Wardman, Kevin Tyers, Yan Shoshitaishvili, Adam Doupé, and Gail-Joon Ahn. Phishtime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists. In *Proc. of USENIX Security*, 2020.

[49] Adam Oest, Penghui Zhang, Brad Wardman, Eric Nunes, Jakub Burgis, Ali Zand, Kurt Thomas, Adam Doupé, and Gail-Joon Ahn. Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale. In *Proc. of USENIX Security*, 2020.

[50] Peng Peng, Chao Xu, Luke Quinn, Hang Hu, Bimal Viswanath, and Gang Wang. What happens after you leak your password: Understanding credential sharing on phishing sites. In *Proc. of Asia CCS*, 2019.

[51] Peng Peng, Limin Yang, Linhai Song, and Gang Wang. Opening the blackbox of virustotal: Analyzing online phishing scan engines. In *Proc. of IMC*, 2019.

[52] Shahrooz Pouryousef, Muhammad Daniyal Dar, Suleman Ahmad, Phillipa Gill, and Rishab Nithyanand. Extortion or expansion? an investigation into the costs and consequences of icann's gtld experiments. In *Proc. of PAM*, 2020.

[53] Pawan Prakash, Manish Kumar, Ramana Rao Kompella, and Minaxi Gupta. Phishnet: Predictive blacklisting to detect phishing attacks. In *Proc. of INFOCOM*, 2010.

[54] F. Quinkert, T. Lauinger, W. Robertson, E. Kirda, and T. Holz. It's not what it looks like: Measuring attacks and defensive registrations of homograph domains. In *Proc. of CNS*, 2019.

[55] Anirudh Ramachandran, Nick Feamster, and Santosh Vempala. Filtering spam with behavioral blacklisting. In *Proc. of CCS*, 2007.

[56] P. Resnick and P. Hoffman. Mapping characters for internationalized domain names in applications (IDNA). RFC 5895, 2008. https://tools.ietf.org/html/rfc5895.

[57] Joshua Reynolds, Deepak Kumar, Zane Ma, Rohan Subramanian, Meishan Wu, Martin Shelton, Joshua Mason, Emily Stark, and Michael Bailey. Measuring identity confusion with uniform resource locators. In *Proc. of CHI*, 2020.

[58] Richard Roberts, Yaelle Goldschlag, Rachel Walter, Taejoong Chung, Alan Mislove, and Dave Levin. You are who you appear to be: A longitudinal study of domain impersonation in tls certificates. In *Proc. of CCS*, 2019.

[59] Stuart Schechter, Rachna Dhamija, Andy Ozment, and Ian C Fischer. The emperor's new security indicators an evaluation of website authentication and the effect of role playing on usability studies. In *Proc. of IEEE SP*, 2007.

[60] StatCounter. Browser market share worldwide, 2020. `https://gs.statcounter.com/browser-market-share`.

[61] Hiroaki Suzuki, Daiki Chiba, Yoshiro Yoneya, Tatsuya Mori, and Shigeki Goto. Shamfinder: An automated framework for detecting IDN homographs. In *Proc. of IMC*, 2019.

[62] Janos Szurdi, Balazs Kocso, Gabor Cseh, Jonathan Spring, Mark Felegyhazi, and Chris Kanich. The long "Taile" of typosquatting domain names. In *Proc. of USENIX Security*, 2014.

[63] Christopher Thompson, Martin Shelton, Emily Stark, Maximilian Walker, Emily Schechter, and Adrienne Porter Felt. The web's identity crisis: Understanding the effectiveness of website identity indicators. In *Proc. of USENIX Security*, 2019.

[64] Ke Tian, Steve T. K. Jan, Hang Hu, Danfeng Yao, and Gang Wang. Needle in a haystack: Tracking down elite phishing domains in the wild. In *Proc. of IMC*, 2018.

[65] Unicode.org. Unicode confusables, 2015. `https://www.unicode.org/Public/security/8.0.0/confusables.txt`.

[66] Unicode.org. Unicode 13.0.0, 2020. `https://unicode.org/versions/Unicode13.0.0/`.

[67] Amber van der Heijden and Luca Allodi. Cognitive triaging of phishing attacks. In *Proc. of USENIX Security*, 2019.

[68] Javier Vargas, Alejandro Correa Bahnsen, Sergio Villegas, and Daniel Ingevaldson. Knowing your enemies: leveraging data analysis to expose phishing patterns against a major us financial institution. In *Proc. of eCrime*, 2016.

[69] W3Counter. Browser & platform market share, 2020. `https://www.w3counter.com/globalstats.php`.

[70] Jingguo Wang, Tejaswini Herath, Rui Chen, Arun Vishwanath, and H. Raghav Rao. Research article phishing susceptibility: An investigation into the processing of a targeted spear phishing email. *IEEE Transactions on Professional Communication*, 55(4):345–362, 2012.

[71] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. In *Proc. of NDSS*, 2010.

[72] Yue Zhang, Serge Egelman, Lorrie Cranor, and Jason Hong. Phinding Phish: Evaluating Anti-Phishing Tools. In *Proc. of NDSS*, 2007.

[73] Yue Zhang, Jason I Hong, and Lorrie F Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proc. of WWW*, 2007.

[74] Xudong Zheng. Phishing with unicode domains, 2017. `https://www.xudongz.com/blog/2017/idn-phishing/`.

Appendix-A: Pilot Studies

To explore the design space, we conducted 4 pilot studies to experiment with different design choices, as shown in Table 9. We framed the questions slightly differently in each pilot study to prime users to focus on the domain names. Below, we use `bankofamerica.com` as an example website.

In pilot study 1, we presented users with the screenshot and asked: "is the domain name shown in the browser address bar `bankofamerica.com`?" By explicitly mentioning the address bar, we cued users to examine the address bar.

In pilot study 2, we asked the same question albeit with 5-point Likert scale answer options: *"1 - I'm very confident it is," "2 - I'm somewhat confident it is," "3 - I can't tell," "4 - I'm somewhat confident it is not,"* and *"5 - I'm very confident it is not."* We tested it to see whether having finer-grained answers would help differentiate conditions (*e.g.*, detection rates of IDNs that are blocked vs. not blocked by Chrome).

In pilot study 3, we tried to avoid priming users to focus on the address bar. We just asked "is the website the real `bankofamerica.com`?" We tested this version because in practice users might not pay attention to the address bar when browsing the web. This should give us a lower bound estimate of people's IDN detection rate.

In pilot study 4, we drew users' attention to the domain name even more by placing the homograph domain name directly in the question. We asked "is `bankofamerı̦ca.com` the same as `bankofamerica.com`?" We essentially asked the users to compare the two domain names side-by-side. We tested this version because it should give us an upper bound estimate of people's IDN detection rate.

Website Selection. We select diverse websites from five common website categories: "Shopping," "Banking," "Social Networking," "Education" and "Government & Military." For each category, we selected two sets of domains: popular and unpopular domains. The popular domains were randomly selected from the Chromium top domain list (3 domains per set). In total, we selected $5 \times 2 \times 3 = 30$ domain names.

For each target domain, we then generated two homograph IDNs: one IDN that can be blocked by the latest Chrome (IDN-Block), and the other IDN can bypass Chrome's policy (IDN-Pass). Thus, for each target domain, we had three choices: IDN-block, IDN-pass, and the real domain name.

Pilot Study Results. In April 2020, we conducted the four pilot studies on MTurk. Each participant examined 30 websites. For each website, we randomly chose to display the real, IDN-Block, or IDN-Pass domain name. Each participant can only participate in one of the pilot studies and for only once. Each participant was compensated $1 for their time.

Study	Experimental Setups			Error Rate			# Participants
	Question	Priming	Answer	Real	IDN-Block	IDN-Pass	(# Answers)
Pilot 1	*"Is the domain name shown in the browser address bar [target domain x]?"*	Medium	Binary	8.75%	46.25%	31.07%	20 (600)
Pilot 2	*"Is the domain name shown in the browser address bar [target domain x]?"*	Medium	Likert Scale	1.98%*	53.29%*	39.85%*	19 (570)
Pilot 3	*"Is the website the real [target domain x]?"*	Light	Binary	16.67%	55.56%	50.79%	18 (540)
Pilot 4	*"Is [homograph domain name] the same as [target domain x]?"*	Heavy	Binary	8.75%	22.5%	16.43%	20 (600)

Table 9: Pilot study set up and their results. *Note that for pilot study 2, we used a five-point Likert scale. We regard the first two points (*i.e.*, "very confident" and "somewhat confident") as the "yes" answer to calculate the error rate. All other pilot studies used the binary answers of "yes" and "no" plus "I can't tell."

To attract serious workers on MTurk, we used commonly applied filters: we recruited U.S. workers who have an approval rate greater than 90%, and have completed more than 50 approved tasks. For each pilot study, we recruited 18 – 20 participants. In total, we had 77 participants with 2,310 answers (domain names).

In Table 9, we show the error rate for the questions in each study. More specifically, if participants answered "Yes", it means they believed the site was the real site. As such, for real websites, answering "Yes" is correct; for IDN websites, answering "Yes" is incorrect. Note that for pilot study 2 where we used a 5-point Likert scale, we considered the first two answers as "YES." Across the four studies, we have two consistent observations. First, participants performed well when they are presented with the *real domain names*. Across the four pilot studies, users' error rates are between 1.98% to 16.67%. Second, when displaying homograph IDNs (either IDN-Block or IDN-Pass), there was a large percentage of wrong answers (i.e., a high error rate). For example, when showing IDN-pass, 16.43% – 50.79% of the times users mistook it as the real domain name.

After comparing the results of the pilot studies, we decided to choose the setting of *Pilot-1* as our main study for the following reasons. First, comparing Pilot-1 and Pilot-2, we did not observe a need to use a 5-point Likert scale as the trend was the same for both conditions and using the Likert scale can complicate the tasks. People might also interpret the five levels differently. Instead, a binary answer (plus "I can't tell") can reduce the ambiguity. Pilot-3 did not prime users to check the domain names in the address bar. Table 9 shows users were more likely to make mistakes as we expected (a lower bound of detection rate). Given that our goal was to test the impact of homograph IDNs, we wanted to examine whether people can identify homograph IDNs when they looked at the domain names. As such the setting of Pilot-3 was not adopted. Pilot-4 represented the other extreme by over-priming users: forcing users to compare the displayed domain names with the real domain names. Table 9 shows that users had an lower error rate as we expected (performance upper bound). However, Pilot-4's setting is too unrealistic as we cannot expect users to do this when browsing the Internet.

Demographics	# Participants
Gender	
Male	139
Female	78
Age	
18-29	75
30-39	83
40-49	36
50 or above	23
Education Level	
High school graduate or less	18
Some college or two-year associate degree	42
Bachelor's degree	114
Some graduate school	11
Master's or professional degree	29
Ph.D.	3
Browser Use History	
Less than a year	2
1-3 years	8
3-5 years	25
More than 5 years	182
Computing Background	
Yes	81
No	130
Prefer not to answer	6

Table 10: Demographic information of participants of the main user study (N=217). We only include participants who passed the attention check.

Appendix-B: Main Study Information

Table 10 shows the demographic information of participants.