A Survey on Domain Impersonation

Derin Amal, Juliane Aulbach* Johannes Zirngibl*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: derin.amal@tum.de, aulbach@net.in.tum.de, zirngibl@net.in.tum.de

Abstract—**Domain Impersonation (DI) is a term that collects the different types of attacks that aim to make a user believe that they are communicating with the desired website when they are visiting a maliciously designed phishing website instead. This paper gives an overview of the different categories of domain impersonation, followed by their different types. After looking at the history of domain impersonation, we review countermeasures including browser employed mechanisms, Certificate Authorities (CAs), Certificate Transparency (CT) logs and an automated Framework called ShamFinder that was introduced recently but is not used yet.**

Index Terms—**domain impersonation, IDN homographs, domain name spoofing**

1. Introduction

The term Domain Impersonation (DI) refers to all attacks where attackers with malicious intentions claim to own a domain that is not theirs. With this method, attackers can create a phishing website that imitates a popular website and collect usernames and passwords. This type of attack can have severe consequences if the information phished is sensitive information like passwords for financial accounts and can costs users and the domain owner significant amounts of money. To prevent domain impersonation, one needs to understand the different approaches by users and their characteristics. For this purpose, we will categorize DI and give an overview of different countermeasures that address different types of attacks under those categories and discuss their weaknesses.

2. Background

We differentiate between two types of domain impersonations. First is where an attacker receives a certificate for a domain they do not own and can falsely claim to be someone they are not. The second type is when attackers create their domain and deceive the user by naming their domain similar to the target domain. There are different methods to achieve this, which we will explain in the following. These kinds of domain names are referred to as "spoofing domain names" [1].

It becomes evident that the significant difference is that in the case of the latter, attackers aim to deceive the user directly by abusing humans' proneness to be inattentive. The first category is characterized by attackers trying to deceive the browser. In a technical matter, browser deception means spoofing the mapping of an IP address to a domain name. Attackers aim to map the IP address of the target domain to their domain name. How this is achieved through different methods will be discussed in Section 3.2. Domain spoofing names, on the other hand, are mapped to their IP address. The attackers' goal is to make users think that the spoofing domain owned by attackers is actually the domain mapped to the target IP address [2], [3].

To provide the reader with the necessary background information, we will introduce some terms.

CAs are responsible for issuing certificates that bind together a domain name and its public cryptographic key. To verify a CA's trustworthiness, CAs can issue certificates for other CAs which leads to a CA hierarchy. [4]

Certificate Transparency (CT) logs aim to make the certificate issuance process transparent to the public. CT is an Internet security standard. It keeps a log of certificates issued by trusted CAs to help users identify maliciously issued certificates [2], [5].

Internationalized Domain Name (IDN) allows non-English characters such as Chinese, Cyrillic, and Arabic to be used in domain names and was first proposed by Dürst in 1996 [1]. Currently, IDN is used as an Internet standard [1].

Fully Qualified Domain Name (FQDN) During this paper, we will use the nomenclature proposed by Roberts et al. in which the complete domain name (e.g., google.example.com) is the fully-qualified domain name (FQDN). In our example, google.com would be the domain to be impersonated called the "target domain" and example.com the actual domain owned by the attackers. [2].

Top Level Domain (TLD) is the last segment of a FQDN, that is, what follows the rightmost dot, e.g., .com in example.com. TLDs can be either generic or country-specific and classify domain names according to their purpose, e.g. .edu for educational facilities or their location, e.g. .de for Germany-based domain names. [6]

3. Domain Impersonation Types and History

In this section, we will give an overview of the different categories of spoofing domain names and browser deception attacks and summarize the transformation of domain impersonation over time.

Type	Example
Typosquatting	facebook.com
Combosquatting	example-site.com could be a spoofing domain name for example.com
Target Embedding	difficult for an average user to differentiate between google.com.site.com and site.google.com
IDN Homograph	éxample.com trying to impersonate example.com

TABLE 1: Examples for different spoofing domain names

3.1. Spoofing Domain Names

There are four common types of spoofing domain names [2]: Target embedding, Unicode Homographs, Typosquatting, and Combosquatting, whereas target embedding is a category recently introduced by Roberts et al. [2].

Typosquatting is the attempt to trick a user by including any 'typos' into a commonly used domain and rely on a user mistaking it for the target domain. Alternatively, attackers who own a Typosquatting domain just hope for users to make a typo while writing a URL in a search bar and access the malicious website by accident [7]–[9].

Target Embedding is a category, where the target domain is embedded in the FQDN in the form of a subdomain. To identify the actual domain, one must read the URL from left to right; the domain name before the TLD is the actual domain. It is important to note that domain impersonation attacks do not include domains that include a target domain owned by the actual domain or cases where the target domain and actual domain are identical. Additionally, Roberts et al. define **subdomain spoofing** as "an umbrella term that includes any attempt at domain impersonation where the target of impersonation is primarily contained in one or more subdomains." [2]. **URL padding** is another form of spoofing, mainly used together with target embedding or combosquatting, where the spoofing domain name is so long that only the deceiving parts(consisting of the target domain) are visible on a user's screen [2].

Combosquatting is similar to target embedding. The target domain is fully included in the FQDN, but contrary to target embedding, the target domain is not a subdomain in this case. [10]

Homographs Unicode Homographs describe the creation of a domain where the name of a commonly used domain is manipulated by using homoglyphs of characters that appear in the target domain. This attack can be highly malicious since some characters are not only confusable but indistinguishable for the human eye and only differ in their Unicode. For the latter, there is no way for a user to detect the attack just by checking the URL.

3.2. Browser Deception Attacks

Cache poisoning in general, is an attack where attackers first request a domain resolution for the target domain and then spoof the response, so the IP address of a domain under their control is cached instead of the respective IP address of the target domain [11]. Another option for cache poisoning to succeed is for attackers to perform a man-in-the-middle attack and eavesdrop on a DNS request. Immediately after, attackers send a spoofed response to the same server. Since they could eavesdrop on the request, attackers know the transaction ID (TxID) entry they use in their spoofed response. This attack exploits the fact that DNS messages are sent without encryption or authentication [12]. Guessing the TxID would be possible, too, but is much less likely to succeed [12]. What makes cache poisoning different from fake certificates is that its essence is to exploit the cache system, which is a memory type. Hence, in case of a successful attack, the consequences will last as long as the false information is stored in the cache. Although it shows to be vulnerable, DNS caching is an essential feature that improves the performance of DNS [12].

Wrongly issued certificates The process of issuing certificates has been proven to be insecure if not carried out correctly [3]. There are different types of validation methods in practice, and they are prone to On-Path attacks. In general, a domain validation process consists of three steps; the application for a certificate, the CA posing a challenge and the applicant doing the challenge, and lastly, the CA checking the challenge and given it is completed, handing out a certificate. There are various ways for a CA to pose a challenge, but we will focus on the abstract process. The key point for an attacker is to fake the successful completion of the challenge, e.g., a DNS challenge where the applicant is supposed to publish a token in the DNS zone file. When the CA checks for completion using a DNS resolver, the attacker spoofs the response tricking the CA into believing that the domain in question is under their control. Note that this is different from cache poisoning since the entries in the DNS resolver cache are not changed. However, the CA is tricked into believing that the IP address of the malicious domain is mapped to the target domain when it is actually not. With a spoofed response, the CA unrightfully issues a certificate to the applicant, which the attacker can use for domain impersonation(e.g., phishing attacks). [13]

3.3. How Did DI Transform over Time?

Although IDN was proposed in 1996 and Gabrilovich and Gontmakher [1] already demonstrated a domain impersonation attack in 2002, homographs were not considered a real threat until IDN started to be widely used around the world with the number being as high as 7.5 million registered IDNs by December 2017 [1]. Also, Hu et al. show that Chrome's defense against IDN homographs that were once 100% effective was not so anymore in their study published two years later [14]. This implies that attackers continue to find new ways to overcome existing security mechanisms. One incident that shows that IDN homograph attacks are a severe issue is the attack on the cryptocurrency exchange company Binance [1]. When companies like Binance are attacked, the consequences for the users and the company can be severe since confidential information will be phished. In the past few years, the possibilities of free certificate issuances like Let's Encrypt have emerged, which had an impact on the number of domain impersonation attacks, too [13]. This is because free issuances give attackers the chance to try attacks without financial barriers [2]. In addition to that, Let's Encrypt uses a fully automated procedure that does not require ownership of domains, but "it suffices to demonstrate control over the domain's name server" [2].

4. Countermeasures

This section looks at different countermeasures and their effectiveness.

4.1. Browser Employed Mechanisms

There are different mechanisms that browsers use to protect users from malicious phishing websites. Browsers show a lock icon when they could authenticate the website they connected to. However, this is not effective in the case of a spoofing domain name. The lock icon even proves counterproductive since users think that the lock icon ensures the website's "trustworthiness". However, when a user falls for a spoofing domain name attack, e.g., target embedding, they click on a malicious link. The browser authenticates that the user is connected to the URL he requested and shows the lock icon. [2] For the threat of IDN homographs, Browsers have introduced defense policies like once a possible threat is detected, the browser will display the Punycode version of the domain name [14]. Punycode was designed to translate IDN to ASCII compatible encoding, and in this case, it is supposed to warn users of a possibly malicious domain name. Nevertheless, studies have shown that after the browser warns the user with the Punycode mechanism, users are still prone to revisiting the spoofing domain since they are not educated on why their browser uses the Punycode [1]. Furthermore, Hu et al. [14] have shown that all of the browsers they tested (which were the most popular ones) have weaknesses in their mechanisms, and the homographs that bypass those measures are still highly deceiving, continuing to threaten users' data privacy.

4.2. Certificate Authorities

DNSSEC is a layer of security that adds cryptographic signatures to existing DNS records to provide authenticity and data integrity [15]. DNSSEC is one of the most effective options to prevent falsely issued certificates since it protects against both off-path, where attackers do not see the network traffic between the CA and the domain owner's servers but can spoof IP packets by claiming to be the domain owner, and on-path attacks, where attackers can eavesdrop on the network traffic and perform an active man-in-the-middle attack [13]. If the domain is not signed with DNSSEC, several best practices can protect against off-path attacks, e.g., DNS Cookies. Protecting against an on-path attack is not as easy. One solution could be to send redundant queries so that the attacker will not be able to spoof them all. [13] Schwittmann et al. note that "CAs either do not employ all available security measures or fail to implement them properly" [13]. Although DNSSEC is an essential step in fighting cache poisoning and avoiding falsely issued certificates, it has not been widely employed because it adds a layer of complexity [12]. Though DNSSEC is necessary to achieve authenticity and data integrity, it is not entirely secure and has further vulnerabilities that could be exploited. DNSSEC does not provide confidentiality, and it is prone to buffer overruns as well as DDoS attacks. In addition to that, DNSSEC does not tolerate malicious server failures. These are a few of the most critical vulnerabilities pointed out by Ariyapperuma et al.. [15], [16]

4.3. Certificate Transparency

The introduction of CT logs brought many advantages for users as well as domain owners. For instance, domain owners now can easily check for certificates that were issued without their knowledge and hence detect a fake certificate domain impersonation attack, as discussed in Section 3.2, before further harm can happen [17]. In addition to that, since there is a general move towards HTTPS, all sites, including phishing sites, need certificates. Scheitle et al. [17] note that because of that, CT logs can be used to detect phishing domain names They conclude this after a pilot experiment where all valid domain names of a popular company are removed from a list generated from a CT log. As a result, there are many domain names left which partly consist of the companies name and therefore have a high potential of being phishing websites. On the other hand, CT brings with them some risks, too. The transparency allows attackers to scan for unknown domains that would not have been publicly known if it was not for the CT logs [18].

4.4. Preventing Cache Poisoning

Although DNS caching creates the opportunity for cache poisoning attacks, it is an essential feature that improves DNS performance. In case of cache poisoning, security toolbars and phishing filters like Phishtank[1], where users can check if a website was voted to be a phishing website by other users if they suspect it to be one, will not work. Even worse, they will confirm that the domain in question is legitimate since the mapping from the target IP to a malicious domain is cached in the resolver. Since cache poisoning exploits the fact that neither DNS entries nor DNS servers are authenticated, DNSSEC can be used to fight off cache poisoning attacks. It will include an authenticating signature for every valid message. The local DNS server will not accept any responses from attackers who cannot sign their spoofed response. [12]

4.5. ShamFinder

As a response to browsers' insufficient countermeasures against IDN homographs, Suzuki et al. introduce a countermeasure named ShamFinder. ShamFinder is an automated framework to detect IDN homographs. ShamFinder abstractly works as follows: It extracts IDN homographs starting with a database of domain names in the wild. Those domain names are then filtered by the ones starting with the prefix "xn–", implying possible

1. https://www.phishtank.com/

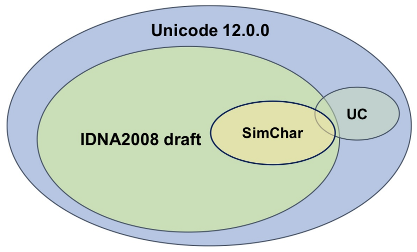

Figure 1: Contamination and overlap of character sets where UC is a recommended mapping for confusable characters [1]

homoglyphs. The next step is to find pairs of the extracted IDNs and popular domain names with the same length (meaning the same number of characters). Note that the extracted IDNs, which possibly are homographs and the list of popular domain names, are two separate sets of data of which the latter can be a ranking list from the Internet. For this purpose Suzuki et al. name Alexa Top Sites[2] as an example. The next step is an algorithm to identify Homographs in the set of extracted IDNs that forms the core of ShamFinder. For the pairs identified in step 2, each character is compared. If two characters at the same index are equivalent, then one proceeds to the next character. If two characters are nonequivalent, then a list of homoglyphs is checked to see whether those characters are homoglyphs. If that is not the case, then the IDN is not considered a homograph. If all characters are equivalent or listed as homoglyphs, the domain will be labeled a homograph. The list of homoglyphs mentioned here is the second contribution of Suzuki et al. [1]. It is named **SimChar** and was constructed as follows: First, each code point is represented as a visual image. Then with a formula as shown in equation 1, the number of different pixels between two glyphs is computed. To summarize, for each pixel, the difference between two images is computed by subtraction. All differences are added up together in the end to obtain a number representing the difference between two glyphs.

$$\Delta = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |I_1(i,j) - I_2(i,j)| \tag{1}$$

Thus a delta equal to zero signifies that both glyphs are visually identical. Now the question is what threshold should be defined to mark the border between homoglyphs and non-homoglyphs. The threshold chosen by Suzuki et al. is 4. A further survey verifies that four is suited as a threshold for what is perceived as confusable by the human eye. The final homoglyph database consists of SimChar combined with what Suzuki et al. call UC, a confusable character database provided by Unicode Technical Standard #39[3].

First of all, the progress made by SimChar in terms of identifying homoglyphs is noteworthy. As seen in Fig 1, SimChar and UC's intersection is relatively small, and it is evident that SimChar has a significant contribution to the number of possible homoglyphs. The major advantage of ShamFinder is that it is an automated framework, meaning it can be expanded whenever new homoglyphs need to be added to the list. The survey done by the authors with human participants verifies that SimChar is a set of glyphs perceived as highly confusing. Therefore one can conclude that the results of ShamFinder will be effective in detecting homographs.

2. https://www.alexa.com/topsites
3. http://unicode.org/reports/tr39/

5. Conclusion and Future Work

To conclude, we can say that DI is a broad topic that brings together vulnerabilities of DNS' different parts. Vulnerabilities in CAs certificate issuance, DNS servers, browsers, and users' behavior can give attackers opportunities to employ DI. One critical suggestions was DNSSEC which is widely known but not implemented by all CAs considered trustworthy. For users, proper education is indispensable and we aim to study the best education approaches in the future. All in all, the issue of DI remains a threat that attackers improve with time, and therefore ever-developing security mechanisms, as well as observation, is necessary. A secure use of IDN can only exist if both sides, user behavior and DNS security, of the problem are approached simultaneously.

References

[1] H. Suzuki, D. Chiba, Y. Yoneya, T. Mori, and S. Goto, "Shamfinder: An automated framework for detecting idn homographs," in *Proceedings of the Internet Measurement Conference*, 2019, pp. 449–462.

[2] R. Roberts, Y. Goldschlag, R. Walter, T. Chung, A. Mislove, and D. Levin, "You are who you appear to be: A longitudinal study of domain impersonation in tls certificates," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2489–2504.

[3] Y. Zeng, T. Zang, Y. Zhang, X. Chen, and Y. Wang, "A comprehensive measurement study of domain-squatting abuse," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[4] C. Crane. (2020) What is a certificate authority (ca) and what do they do? [Online]. Available: https://www.thesslstore.com/blog/what-is-a-certificate-authority-ca-and-what-do-they-do/

[5] C. transparency. How ct works. [Online]. Available: https://certificate.transparency.dev/howctworks/

[6] techopedia. (2021) Top-level domain (tld). [Online]. Available: https://www.techopedia.com/definition/1348/top-level-domain-tld

[7] J. Szurdi, B. Kocso, G. Cseh, J. Spring, M. Felegyhazi, and C. Kanich, "The long "taile" of typosquatting domain names," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 191–206.

[8] J. Spaulding, S. Upadhyaya, and A. Mohaisen, "The landscape of domain name typosquatting: Techniques and countermeasures," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*. IEEE, 2016, pp. 284–289.

[9] P. Agten, W. Joosen, F. Piessens, and N. Nikiforakis, "Seven months' worth of mistakes: A longitudinal study of typosquatting abuse," in *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)*. Internet Society, 2015.

[10] P. Kintis, N. Miramirkhani, C. Lever, Y. Chen, R. Romero-Gómez, N. Pitropakis, N. Nikiforakis, and M. Antonakakis, "Hiding in plain sight: A longitudinal study of combosquatting abuse," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 569–586.

[11] J. Trostle, B. Van Besien, and A. Pujari, “Protecting against dns cache poisoning attacks,” in *2010 6th IEEE Workshop on Secure Network Protocols*. IEEE, 2010, pp. 25–30.

[12] R. Bassil, R. Hobeica, W. Itani, C. Ghali, A. Kayssi, and A. Chehab, “Security analysis and solution for thwarting cache poisoning attacks in the domain name system,” in *2012 19th International Conference on Telecommunications (ICT)*. IEEE, 2012, pp. 1–6.

[13] L. Schwittmann, M. Wander, and T. Weis, “Domain impersonation is feasible: A study of ca domain validation vulnerabilities,” in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 544–559.

[14] H. Hu, S. T. Jan, Y. Wang, and G. Wang, “Assessing browser-level defense against idn-based phishing,” in *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[15] cloudfare. How dnssec works. [Online]. Available: https://www.cloudflare.com/de-de/dns/dnssec/how-dnssec-works/

[16] S. Ariyapperuma and C. J. Mitchell, “Security vulnerabilities in dns and dnssec,” in *The Second International Conference on Availability, Reliability and Security (ARES’07)*. IEEE, 2007, pp. 335–342.

[17] O. Gasser, B. Hof, M. Helm, M. Korczynski, R. Holz, and G. Carle, “In log we trust: Revealing poor security practices with certificate transparency logs and internet measurements,” in *International Conference on Passive and Active Network Measurement*. Springer, 2018, pp. 173–185.

[18] Q. Scheitle, O. Gasser, T. Nolte, J. Amann, L. Brent, G. Carle, R. Holz, T. C. Schmidt, and M. Wählisch, “The rise of certificate transparency and its implications on the internet ecosystem,” in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 343–349.