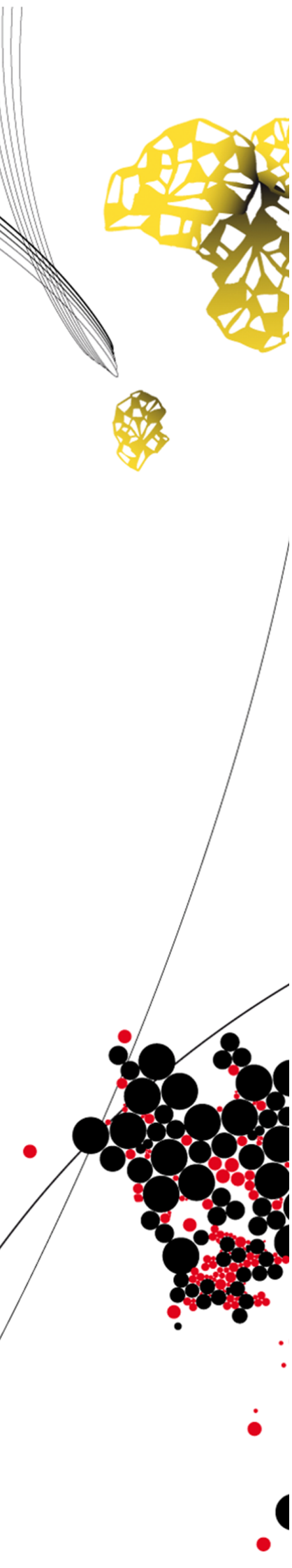


UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering,
Mathematics & Computer Science

Detection of Malicious IDN Homoglyph Domains Using Active DNS Measurements

Ramin Yazdani

Master of Science Thesis
August 2019

Supervisors:
dr. Anna Sperotto
Olivier van der Toorn, MSc

Graduation Committee:
prof.dr.ir. Aiko Pras
dr. Anna Sperotto
dr.ir. Roland van Rijswijk-Deij
dr. Doina Bucur
Olivier van der Toorn, MSc

DACS Research Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Preface

Throughout conducting this research, I received great support from several people. I would like to express my appreciation to my supervisor, dr. Anna Sperotto for her great support in formulating my research. I received great critical and encouraging feedback from you during our meetings. Thanks for always being concerned about my progress as well as whether I liked the topic.

I would also like to thank Olivier van der Toorn, MSc for always being enthusiastic and willing to help me when I had difficulties in figuring out some aspects of my research. I interrupted you every now and then, but you always kindly helped me.

In addition, I would like to thank other members of my graduation committee members, prof.dr.ir. Aiko Pras, dr.ir. Roland van Rijswijk-Deij, dr. Doina Bucur and members of DACS research group for their precious remarks during this research which helped me to better steer my thesis.

Last but not least, a special thanks to my family. Words cannot express how grateful I am to them for always being supportive during different stages of my studies.

Ramin Yazdani
Enschede, July 2019

Summary

At early stages of Internet development, users were only able to register or access domains with ASCII characters. The introduction of IDN (Internationalized Domain Name) which uses the larger Unicode character set, made it possible for regional users to deal with domain names using their local language alphabet. Beside the advantages provided by IDN, a new type of network threats has also emerged. The reason behind this is that there are many similar-looking characters in Unicode system, called homoglyphs. These characters could be used by an attacker to lure users by replacing one or more characters of a benign domain.

Although there are many homoglyphs in the Unicode system, there is no absolute way to group them and some researches are done to create homoglyph confusion tables. However, performance of these tables is not assessed. Quality of existing confusion tables is explored during this thesis by applying them to different domain datasets, as well as comparing their performance to a proposed table. A Unicode character might have both a Unicode and an ASCII homoglyph; however, due to the computational limitations, only ASCII homoglyphs of Unicode characters are considered to extract homoglyph domain pairs in this research. Results show that using the proposed table we are able to detect more homoglyph domains than existing tables. Besides, considering the time gap between registration of a malicious domain and actively using it to perform an attack, it would be possible to reveal attacks at their infancy or even before they happen using the traces left in the DNS (Domain Name System) data. The database provided by OpenINTEL active DNS measurement platform is used in this research to detect malicious IDN homoglyph domains. "*ASN*" (Autonomous System Number) and four DNS records, namely "*A*" (IPv4 host address), "*AAAA*" (IPv6 host address), "*NS*" (Name Server) and "*MX*" (Mail eXchanger), are used to distinguish between malicious and benign domains. Results show on average 42 days early detection of malicious domains, compared to appearance of them on existing blacklists.

The main outcome of this research is documented as a paper to be submitted to IEEE/IFIP NOMS 2020 conference, and this thesis only includes documentation of parts of the research which are not covered in the paper. For an easier comprehension of the contents, it is highly recommended to first read the NOMS paper in

Appendix A, before continuing with the rest of this thesis.

Contents

Chapter 1

Introduction

This document is titled as "*Master of Science Thesis*"; however, it has a major difference in its structure compared to traditional theses. In an agreement with graduation committee, it has been decided to document the outcome of this research in the form of a paper. This paper is meant to be submitted to IEEE/IFIP Network Operations and Management Symposium (NOMS) 2020 which is a well known conference in the field of networking. This paper is included in Appendix A. The main body of thesis only includes documentation of parts of the research which are not covered in the NOMS paper. Thus, for an easier comprehension of the contents, it is highly recommended to first read the NOMS paper in Appendix A, before continuing with the rest of this thesis.

During the organization of this thesis, I have tried to fulfill the requirements set by the examination committee. These requirements and how they are met through this research are discussed in the following subsections. I have been fully involved in all steps of this research during its 28 weeks long period, starting from defining the research, followed by doing a literature review, production of the results for the proposed methodology and validation of these results and finally writing a paper as the main outcome that concludes this research. While I am the main author of the above-mentioned paper, my supervisors have contributed to it by providing their valuable feedback.

1.1 Assessment Standards

In order to conclude a master degree study, students are supposed to write a thesis. This thesis must document the research done during their graduation assignment. A list of requirements and standards [1] are set by each faculty which need to be considered in the thesis. The set of learning objectives for assessment standards are grouped into 3 categories: scientific quality, organization and communication. In

the following subsections, an elaboration is given on how each of these requirements are satisfied.

1.1.1 Scientific quality

- **Interpret a possibly general project proposal and translate it to more concrete research questions.**

  Despite some master degree programs in which there is a part separated from thesis called "*Research Topics*", in Electrical Engineering there is not such a course and complete thesis is done in one step as a graduation assignment. However, as any other research project, this is a fundamental part that formulates research. Thus, similar steps were taken which formed a research proposal and partially the "*Introduction*", "*Background*" and "*Related Work*" sections of the resulting paper in Appendix A. Through a literature study, following three research questions were defined as the basis of this research:

  - How large could be the problem of malicious IDN homoglyph domains?
  - How could we decide whether an IDN domain which has an ASCII homoglyph, is a malicious one?
  - Can we detect malicious IDNs before they appear on blacklists and if so, what is the achievable time advantage?

  These research questions would be summarized in the overall research question "*What would be the advantage of detecting malicious IDN homoglyph domains using active DNS measurements?*" to get a global idea of the goal behind this research.

- **Find and study relevant literature, software and hardware tools, and critically assess their merits.** Research questions mentioned above are based on the literature study, critically assessing pros and cons of the existing work. These information was used to explore subjects in the literature which still needed to be explored. Based on the literature study, available tools and feasibility of various methods were investigated. Important parts of the literature review which did not make it to the paper due to the limited number of pages, are discussed with more details in Chapter 2.

- **Work in a systematic way and document your findings as you progress.** Working according to a timetable is a necessity in conducting every project in time and efficiently. Thus, during definition phase of this research a time table was defined to make sure that the deadlines are met. All of the scripts

used during this thesis were documented with details on how to use them, so that the results are reproducible. This documentation also made it possible to identify errors and verify results whenever needed.

- **Work in correspondence with the level of the elective courses you have followed.**

  Although it is hard to pinpoint the relation of each course to one part of this thesis, there were absolute lessons learned during either elective or compulsory courses which were applied to conduct this research. An example would be the programming skills developed during projects for these courses. Considering the fact that the outcome of this research is to be submitted to a reputable conference, would guarantee that its level is comparable to the courses taken.

- **Perform original work that has sufficient depth to be relevant to the research in the chair.** As already mentioned, the outcome of this research is to be submitted to a highly reputable conference in the field of networking and computer science. This already emphasizes necessity for originality and scientific depth of the research. Besides, many other researches in similar topics is being conducted in DACS research group, which proves its relevance to research group.

1.1.2 Organization, planning, collaboration

- **Work independently and goal oriented under the guidance of a supervisor.**

  The majority of this work was done independently and goal oriented, under critical assessment of my supervisors. We had regular meetings to discuss thesis progress with dr. Anna Sperotto and Olivier van der Toorn, MSc. During these meetings I presented what I had done since our last meeting and got feedback on it as well as getting recommendations on how to continue with the rest of project.

- **Seek assistance within the research group or elsewhere, if required and beneficial for the project.**

  As there are multiple researcher in DACS group dealing with projects in the same field, it was highly beneficial for me to get assistance from ones who had the highest experience around the issues I faced.

- **Benefit from the guidance of your supervisor by scheduling regular meetings, provide the supervisor with progress reports and initiate topics that will be discussed.**

During this thesis I had regular meetings with dr. Anna Sperotto and Olivier van der Toorn, MSc. I prepared progress reports for each session to make meetings more efficient as well as documenting feedback without missing important points. After each meeting I got a clear scope of how to tackle issues I faced.

- **Organize your work by making a project plan, executing it, adjusting it when necessary, handling unexpected developments and finish within the allotted number of credits.**

  During definition of the proposal for this thesis, a time plan was made with a time period allocated to each of the predefined tasks. This time table is available in Appendix B.

1.1.3 Communication

- **Write a Master thesis that motivates your work for a general audience, and communicates the work and its results in a clear, well-structured way to your peers.**

  This thesis including the paper derived from it explains the drawbacks of the existing work around its topic, which motivates why it was beneficial to conduct this research. Results are given in an order which helps the reader to keep track of how each research question is answered. Besides, metrics used in results are meant to clearly deliver outcomes.

- **Give a presentation with similar qualities to fellow-students and members of the chair.**

  In order to update other researchers with relevance of the topic and progress of the work, a progress report presentation was conducted on 24th of April 2019. The thesis will have a defence session to present the committed work to a larger group of audiences. In case the paper to be submitted to IEEE/IFIP NOMS 2020 gets accepted, there will be another chance to present the research. Table 1.1 summarizes the details of each presentation.

The remainder of this thesis is structured as follows. In Chapter 2 details of the study during literature review which were not discussed in the NOMS paper are presented. Chapter 3 discusses additional results and graphs of the research. In Chapter 4, reflections and lessons learned during this research are given. Chapter 5 concludes the thesis and discusses future work. The paper to be submitted to the NOMS 2020 conference is given in Appendix A. The thesis proposal defined at the starting phase of this research is presented in Appendix B. Sample rows of the

Table 1.1: Summary of the presentations for this research

Date	**Occasion**	**Place**	**Audience**
2019-04-24	Progress report	University of Twente	DACS group
2019-08-21	Thesis defence	University of Twente	DACS group, friends
2020-04-20 2020-04-24	NOMS presentation (if accepted)	NOMS conference, Budapest	Experts in the field

existing homoglyph tables are represented in Appendix C. The revisions made to form the proposed Unicode homoglyph table are given in Appendix D.

Chapter 2

Literature Review

This chapter discusses the additional background of IDN homoglyph domains which is summarized in "*Background*" and "*Related Work*" sections of NOMS paper. As a starting point, it is important to get an idea about size of the problem we are facing with.

2.1 IDN growth

Internationalized Domain Names were proposed in 1996 and implemented in 1998 for the first time for generic TLDs (Top Level Domains). However, using IDNs for ccTLDs (country code Top Level Domains) was not approved till 2009 and first IDN ccTLDs were then installed in 2010. EURid (EUropean Registry for internet domains) [2] provides annual world reports on internationalized domain names. They have been studying IDNs since 2011, and gathered data starting from 2009. Figure 2.1 plots the total number of IDNs reported by EURid. As seen in this plot, the number of IDNs has been monotonically increasing from their introduction till 2016. However there is a reduction of approximately 14% which is mainly caused by the change of policy for "*.vn*" ccTLD. In December 2017, 71% of registered IDNs were at the second level (such as IDNs in "*.com*" TLD), and 29% were at the top level (such as IDNs in "*.xn--p1ai*" ccTLD) [2].

The distribution of IDNs in different TLDs and SLDs (Second Level Domains) are depicted in Figure 2.2. These figures reveal that a large portion of IDNs are registered using Eastern Asian language scripts such as Chinese, Japanese and Korean. Characters used in these languages mainly do not have a high level of similarity with ASCII characters. Thus considering the main scope of this thesis, which is to find IDN homoglyphs for ASCII domains, investigation of these TLDs would not be interesting.

Figure 2.3 represents the contribution of various language scripts used on IDNs

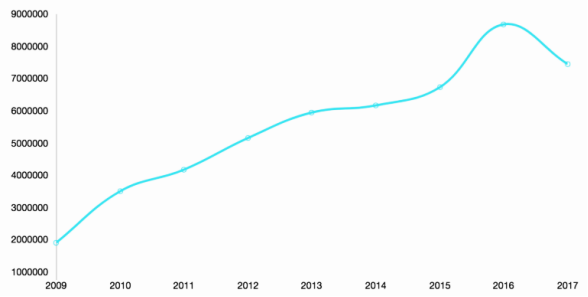
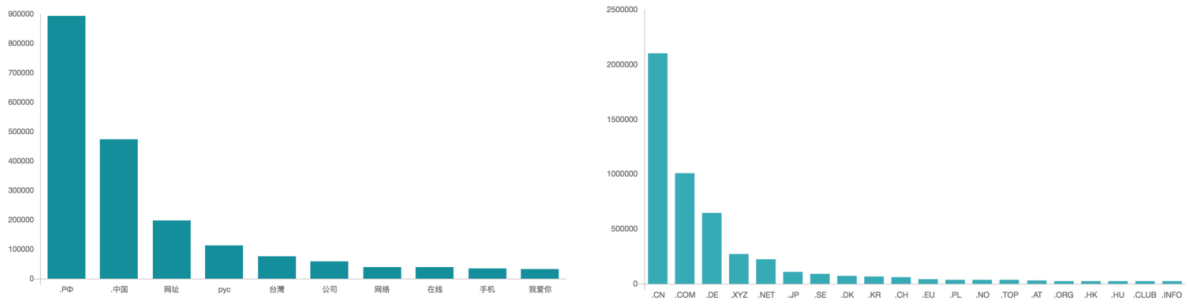


Figure 2.1: IDN growth reported by EURid [2]

reported by EURid. As seen in this figure, roughly 42% of IDNs are based on Latin and Cyrillic scripts. IDNs using these scripts might be registered as a malicious homoglyph for an ASCII domain. The dataset used in this thesis is obtained from OpenINTEL platform [3], covering "*.com*" TLD and 9 ccTLDs ("*.se*", "*.nu*", "*.ca*", "*.fi*", "*.at*", "*.dk*", "*.ru*", "*.xn--p1ai*" and "*.us*") which approximately covers 26% of total IDNs. Comparing this dataset to Figure 2.3, roughly 62% of total IDNs with a potential ASCII homoglyph domain are covered by the dataset used in this research.



(a) Top TLD IDNs (b) Top SLD IDNs

Figure 2.2: Top TLD and SLD IDNs reported by EURid [2]

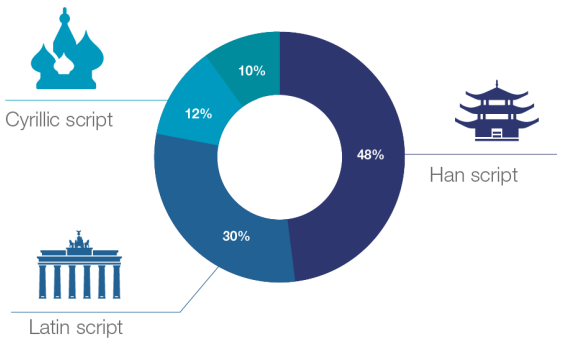


Figure 2.3: Language scripts of IDNs reported by EURid [2]

2.2 Punycode Algorithm

The DNS protocol being one of the cornerstones of the internet was designed long before the introduction of IDNs and thus it is only compatible with ASCII characters. In order to keep backward compatibility, IDNs had to be converted to an ASCII equivalent string before being implemented in DNS. This ASCII Compatible Encoding (ACE) needs to be easy to implement as well as minimizing the length of the encoded string. This is necessary since a domain label is limited to 63 characters (RFC 1034) [4]. To convert an IDN to the ACE (also called Punycode format), the Punycode algorithm [5] was introduced. This algorithm is explained here giving an example. Consider the domain name “exámple.com” (with the equivalent Punycode string “xn--exmple-qta.com”) in which the label “exámple” contains the non-ASCII character *latin small letter “a” with acute “á”* (U+00E1) with a decimal value of 225. In order to convert this label to the ACE, the Punycode algorithm starts with copying all ASCII characters existing in the IDN to the output (exmple) and adding a hyphen character to the resulting string (exmple-). This hyphen notifies the end of ASCII characters present in the original label. In the next step, the non-ASCII characters (“á” in “exámple.com”) and their location in the original string (third character in “exámple.com”) must be encoded in the output. In order to keep the encoded string short, only an integer value is embedded in the output (“qta” with decimal value 681 in “xn--exmple-qta.com”) for each non-ASCII character. This integer represents both the Unicode character and its location in the original label. Besides, rather than conventional integer representation generalized variable length encoding (base-36) is used which avoid using delimiters between consecutive integers. Finally an “*xn--*” prefix is added to output string to make it distinguishable form basic ASCII strings.

The method used to encode integers (“qta”) in the output string is better understood considering the decoder side. A decoder is a finite state machine with two counters i and n [5]. Counter i represents the possible locations to insert a non-ASCII character and always can be an integer between 0 and current length of the string k (k=6 in “exmple”). Counter n represents the decimal codepoint of non-ASCII characters and starts from 128 (decimal codepoint for the first non-ASCII character) and is incremented till the right extended character is found (“á” with decimal codepoint of 225). For each n value, the value of i increments with steps of one, resetting to zero when it reaches the length of the string, and then n is incremented by one. This process is continued till (n-128)*(1+k)+i is equal to the encoded integer (n=225, k=6, i=2). At this point the decoder understands that it has to put n^{th} Unicode character (“á”) at (i+1)th slot (third character) in the sting and then continues with decoding remaining non-ASCII characters (if there were any).

Table 2.1: Sample homoglyphs for “example.com"

Domain name	Punycode format	Code point of characters
example.com	example.com	0065, 0078, 0061, 006D, 0070, 006C, 0065, 002E, 0063, 006F, 006D
example.com	xn--exmple-4nf.com	0065, 0078, **0430**, 006D, 0070, 006C, 0065, 002E, 0063, 006F, 006D
example.com	xn--ml-6kctd8d6a.com	**0435**, **0445**, **0430**, 006D, **0440**, 006C, **0435**, 002E, 0063, 006F, 006D

2.3 Homoglyph characters

IDNs provide an advantage for local users to access domains in their native language alphabet. However, this feature might be misused by attackers due to the existence of many similar looking Unicode characters called homoglyphs. An attacker is able to replace one or more characters in a domain name to lure a victim who is intending to access a benign domain. As an example, two potential attack vectors for “*example.com*” are given in Table 2.1. The first row of this table represents the domain name with ASCII-only characters. In the second and third rows 1 and 5 characters are replaced by their Unicode homoglyphs, respectively.

The similarity of two characters does not have a clear definition and there is no concrete method to define whether two characters are similar. Thus, homoglyph confusion tables are introduced in the literature, in which similar-looking characters in the Unicode system are grouped. There are a number of studies providing these confusion tables. “*Unicode Confusables*” [6] and “*UC-SimList0.8*” [7] are two publicly available homoglyph tables used in this research. Sample rows of these tables are given in Appendix C. In order to construct the “*UC-SimList*”, the Microsoft Arial Unicode MS font is used in [8], since it covers more Unicode characters than other existing fonts. English, Chinese and Japanese are three languages considered to develop this table. The visual similarity is calculated by calculating the similarity of each pair of characters c_1, c_2 and is denoted with $vs(c_1, c_2)$ [8],

$$vs(c_1, c_2) = \frac{|OverlapPix(c_1, c_2)|}{p|Pix(c_1)| + (1-p)|Pix(c_2)|}, \tag{2.1}$$

where $|OverlapPix(c_1, c_2)|$ is the number of overlapping pixels of the bitmaps of c_1 and c_2, $|Pix(c)|$ is the number of pixels of the character c, and $p \in [0, 1]$ is the factor for tuning the similarity computation validity. It is mentioned in [8] that the best experimental value for p is 1 when the number of pixels of character c_1 is larger than the number of pixels of character c_2, and 0 otherwise.

Chapter 3

Results

This chapter contains results achieved during this thesis which are not covered in NOMS paper due to the limited number of available pages. These results are aligned with answering the research questions discussed in Section 1.1.1.

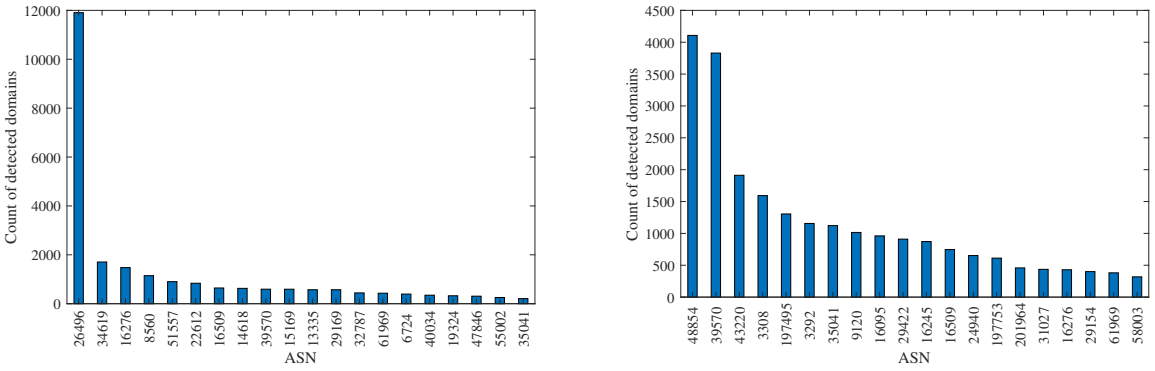
3.1 Homoglyph tables

As discussed in Chapter 2, there are multiple studies which aim to provide Unicode homoglyph confusion tables. Efficiency of "*Unicode Confusables*" [6] and "*UC-SimList0.8*" [7] were explored in this research. Another table which is a combination of these two tables with some revision is proposed in this research. It includes addition of 81 missing characters and replacing 42 characters form "*UC-SimList0.8*" for which a better homoglyph existed. The proposed table is publicly available online on "https://www.tide-project.nl/blog/noms2020/". Details about revisions made to form the proposed table are given in Appendix D.

3.2 ASN distribution

The "*ASN*" record obtained from OpenINTEL platform would be useful to qualify if malicious IDN homoglyph domains are mostly registered by a specific authority. In order to do so, Top-20 ASN distributions for detected IDN homoglyph domains in "*.com*" and "*ccTLDs*" are plotted in Figure 3.1. At first glance, a difference between these two plots is that the distribution for "*ccTLDs*" is scattered over different AS numbers, however in "*.com*" a big group of domains are registered using the same "ASN".

A closer look at the owner information of these ASNs reveals that for ".com" TLD, roughly 31% of domains have an "ASN" corresponding to "*GoDaddy*" which is the largest domain registrar for this TLD. On the other hand in "*ccTLDs*" the AS numbers



(a) ".com" TLD (b) "ccTLDs"

Figure 3.1: Distribution of ASNs

with highest amount of corresponding domains are owned by "*Zitcom*" and "*Loopia*" which are two Danish and Swedish hosting companies. Since our "*ccTLDs*" dataset included "*.dk*" and "*.se*", this ASN distribution does not reveal any unusual behaviour.

3.3 ccTLD parsing

The growth of IDNs plotted in the NOMS paper exhibited a weekly pattern for "*ccTLDs*". In order to investigate this behaviour, contribution of each unique ccTLD in this group was looked up. Results are plotted in Figure 3.2. This figure reveals that "*xn--p1ai*" ccTLD is causing this weekly pattern in which the number of existing domains typically increases monotonically till Mondays and drops on Tuesdays. Further investigation is necessary to figure out why this happens, which was out of scope of this research. However, a potential reason might be de-registration policy of expired domains implemented by registrars for this ccTLD.

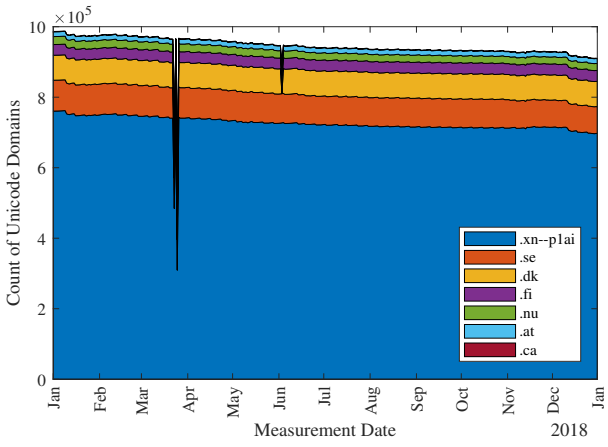


Figure 3.2: Contribution of each countrycode TLD in "ccTLDs" dataset

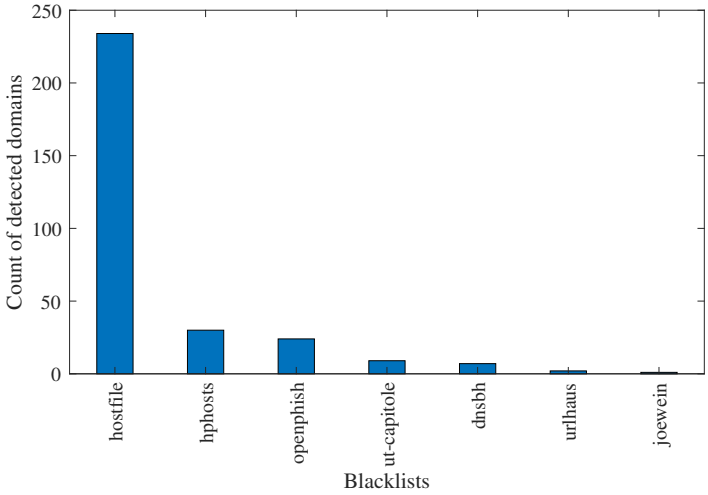


Figure 3.3: Domains detected by blacklists

Comparing Figure 3.1b and Figure 3.2, one might also notice that, although "*.xn--p1ai*" has a large share in our "*ccTLDs*" dataset (roughly 87%), the corresponding "*ASN*" for these domains are not concentrated as the case for "*.se*" and "*.dk*". The reason behind this phenomena would be the wide geographic distribution of registrants of these domains.

3.4 Blacklists

Appearance of detected domains on a list of existing blacklists (called "*RBL*" in this research) is investigated in NOMS paper. Results show that a very limited number of detected domains by the proposed method have appeared on the blacklists. Count of the domains detected by each blacklist is plotted in Figure 3.3. This plot shows that roughly 76% of domains are detected by the "*hostfile*" blacklist.

3.5 Maliciousness scores

The method proposed in the NOMS paper to calculate scores for differences in "*ASN*" and DNS records, considers records as different only if both domains have an entry for that record in OpenINTEL measurements. However, this could be done in various ways. For example one might consider two records as different even when one of them is empty. Another approach would be considering different weights for records to emphasize ones that are considered to have a higher impact on our decision. Figure 3.4 plots the scores for domains extracted in "*.com*" and "*ccTLDs*" when an empty record compared to a nonempty corresponding record is also counted as

a different one. Comparing these scores to ones presented in Fig. 8 of NOMS paper, we notice that the number of domains with an score higher than 3, considerably increases. Although this would result in detection of more malicious domains, the false positive rate would increase as well.

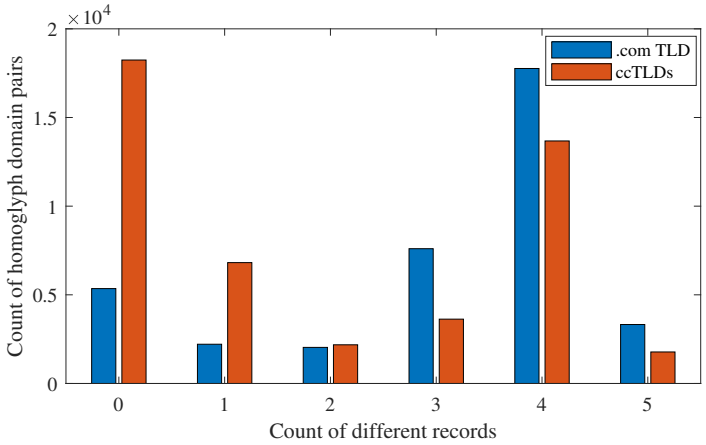


Figure 3.4: Scores for extracted domains

Chapter 4

Lessons Learned

This chapter discusses the lessons learned through this research. Although contents of this chapter are a personal reflection rather than a scientific context, it would still provide valuable points about the thesis.

4.1 Unicode system

Despite the ASCII encoding which is frequently used in programming languages, the Unicode system used to extend the basic ASCII characters is not that prevailing. Before conducting this research I did not have a clear view on how the Unicode system works. Besides, as discussed in this thesis, it is not possible to apply the Unicode characters directly on DNS, which increases the complexity around this topic. During this thesis I got hands on experience with the Unicode character system and algorithms used to convert them to DNS compatible strings, without which I wouldn't be able to manage my research properly.

4.2 Hadoop ecosystem

The Hadoop framework is used to store and process big data in a distributed manner. This framework is utilized in OpenINTEL measurements platform to store and process DNS queries. Although this ecosystem was not something that I directly needed to conduct my research, the knowledge I achieved around how this framework is built up, would be a fruitful lesson I learned during this thesis.

4.3 SQL queries

I had used SQL queries before this research, however it was limited in terms of both frequency and complexity of the queries. Since DNS measurements of OpenINTEL platform used to provide the input dataset of this research, were accessed through Impala engine (which provides a SQL-like interface), proper SQL queries were an inseparable part of this research to ensure the integrity of achieved results. This is important because an incorrect SQL query will still provide you with results that you might not be able to verify them due to large size of your dataset. Thus, I tried to parse my dataset into small subsets and check for their integrity during various steps of this research.

4.4 Programming

Python is a programming language which is popular nowadays and as many other researchers I have used it during various projects. However, I always learn new lessons in programming when dealing with a new topic. As an example, processing Unicode strings and characters was something that I had never performed in python or any other script. Therefore I faced new errors which were not only relatively simple syntax errors, but ones that were not discovered till looking closer at the outputs.

4.5 Writing skills

In order to write a paper which is appealing to readers, we need to be able to critically assess our writings to find out parts which are not well discussed or missing. This is sometimes tricky, since considering your paper from another person's point of view who is not thoroughly familiar with your topic, is not trivial. Our skill in writing develops as we do and this is the case for me as well. I try to learn from my previous writings to improve my future papers.

Chapter 5

Conclusions and Future Work

This chapter consists of conclusions of the thesis as well as recommendations for future work.

5.1 Conclusions

A malicious IDN homoglyph detection method was proposed in this thesis. In the following a conclusion is given based on each research question discussed in Section 1.1.1.

5.1.1 IDN homoglyphs

The IDNs existing in OpenINTEL DNS measurements were investigated in this research to determine the size of malicious homoglyph IDNs problem and answer **RQ1**. Since there is no absolute method to group homoglyph domains, two existing homoglyph confusion tables were used in this research. Beside the utilization of existing homoglyph tables to extract similar looking domains, an improved table was proposed in this research which is a combination of existing tables by considering some revisions. It is shown that using the proposed table there is a higher opportunity to detect homoglyph domains. Results achieved by using the proposed table reveal that roughly 23% of the added IDNs in "*.com*" TLD and 13% of the added IDNs in "*ccTLDs*" group have an ASCII homoglyph domain. Considering a threshold of 3, roughly 44% of the domains with a homoglyph in "*.com*" TLD (10% of added IDNs) and 24% of these domains with a homoglyph in "*ccTLDs*" group (3% of added IDNs) are highly suspicious homoglyph domains. It is noteworthy that results for "*ccTLDs*" might be biased as "*xn--p1ai*" ccTLD which is based on Cyrillic script, has a large share in our "*ccTLDs*" dataset and a homoglyph ASCII counterpart for these domains rarely could be found.

5.1.2 Detection method

The extracted domains were investigated to decide whether they are meant for malicious intent. This was done since not all IDN homoglyph domains are malicious and there are domains proactively registered by a company for protection purposes. In order to differentiate between suspicious and benign domains and answer **RQ2**, "*ASN*" and four DNS records from OpenINTEL platform were queried for homoglyph domain pairs. Based on these records, a score of maliciousness is calculated and if it is greater than a threshold, the Unicode domain is high likely used for malicious activity. Since there is not enough evidence to mark these domains as malicious, they are only considered as candidates which are highly suspicious for further investigation.

5.1.3 Time advantage

By defining **RQ3**, I was concerned about the achievable time advantage through early detection of malicious domains. Previous studies propose that due to a time gap between registration of malicious domains and actively using them in attacks, there is a possibility to detect them before they are used for attacks. This feature was utilized here, using traces left in DNS data. By comparing detection results of the proposed method to the appearance of detected domains on existing blacklists, on average an early detection of 42 days is achieved. This gets more important if we consider domains detected by the proposed method which have not yet ended up on blacklists.

5.2 Future work

The proposed method to detect malicious IDN homoglyphs in this thesis could be improved in several ways. These are discussed in the following.

First of all, in this research only one ASCII homoglyph is used for each Unicode character. This could be improved in two ways, by considering Unicode homoglyphs for Unicode characters as well as considering multiple ASCII homoglyphs of a Unicode character. Although this would increase the computational complexity of the proposed method, the outcome might include interesting observations.

Another aspect to be investigated more is to use a larger dataset in terms of TLD coverage, specially those which include many Latin based characters.

Finally, a combination of whois data and OpenINTEL measurements would be used as enrichment data to increase the accuracy of the proposed method.

Bibliography

[1] “Master’s thesis assessment standards.” [Online]. Available: https://www.utwente.nl/en/mee/programme-information/Master’s%20thesis/Description%20of%20the%20master’s%20thesis/#assessment-committee

[2] “EURid IDN world report,” https://idnworldreport.eu/.

[3] “OpenINTEL,” https://www.openintel.nl/.

[4] P. V. Mockapetris, “Domain names-concepts and facilities,” 1987.

[5] A. Costello, “Punycode: A bootstring encoding of unicode for internationalized domain names in applications (idna),” Tech. Rep., 2003.

[6] “Unicode Confusables list.” [Online]. Available: https://unicode.org/Public/security/

[7] “Unicode Similarity List.” [Online]. Available: http://people.csail.mit.edu/ayf/IRI/UCSimList/UCSimList/UC_SimList0.8.txt

[8] A. Y. Fu, X. Deng, L. Wenyin, and G. Little, “The methodology and an application to fight against unicode attacks,” in *Proceedings of the second symposium on Usable privacy and security*. ACM, 2006, pp. 91–101.

Appendix A

IEEE NOMS 2020

This appendix contains the final version of the paper to be submitted to IEEE/IFIP Network Operations and Management Symposium 2020 (NOMS) as the main outcome of this research. The paper is titled "*Detection of Malicious IDN Homoglyph Domains Using Active DNS Measurements*".

Detection of Malicious IDN Homoglyph Domains Using Active DNS Measurements

Ramin Yazdani
University of Twente
Enschede, The Netherlands
r.yazdani@student.utwente.nl

Anna Sperotto
University of Twente
Enschede, The Netherlands
a.sperotto@utwente.nl

Olivier van der Toorn
University of Twente
Enschede, The Netherlands
o.i.vandertoorn@utwente.nl

***Abstract*—The possibility to include Unicode characters in domain names lets local users to deal with domains in their regional languages, which is done through the introduction of Internationalized Domain Names (IDN). Due to the visual similarity of Unicode characters in different languages - technically called homoglyphs - a new type of potential threat has been of concern. The IDN homograph attack is a way that an attacker might impersonate a benign server by replacing one or more characters by their homoglyph. Although there are many homoglyphs in the Unicode system, there is no absolute way to create Unicode homoglyph confusion tables. The quality of existing confusion tables is explored in this paper by applying them to different domain datasets, as well as comparing their performance to a proposed table. Results show that using the proposed table we are able to detect more homoglyph domains than existing tables. Besides, considering the time gap between the registration of a malicious domain and actively using it to perform an attack, it would be possible to reveal attacks at their infancy or even before they happen using the traces left in the DNS data. The database provided by the OpenINTEL active DNS measurement platform is used in this research to detect malicious IDN homoglyph domains. "ASN" record and four DNS records, namely "A", "AAAA", "NS" and "MX", are used to distinguish between malicious and benign domains. Results show on average 42 days early detection of malicious domains, compared to appearance of them on existing blacklists.**

***Index Terms*—homoglyph, IDN, homograph attacks, malicious domains, active DNS measurements**

I. INTRODUCTION

In order to store and represent characters of Latin alphabet in early stages of computer systems, ASCII encoding standard was developed which maps each character to an 8 bits string of zeros and ones. However, expanding this encoding scheme and creating a unified character system was necessary to include all of the characters from different regional languages. The Unicode standard [1] was introduced to solve this issue, in which each character is mapped to one to four bytes using UTF-8 (8-bit Unicode Transformation Format) variable width encoding scheme. The Internationalized domain name (IDN), originally proposed in 1996 [2], is the term used for a domain name in the Internet, containing one or more labels in a language-specific script or alphabet, such as Greek, Cyrillic, Arabic, Chinese or the Latin-based characters with diacritics, etc by making use of the Unicode characters. Beside the advantages of the Unicode system in the user experience, there are major security risks with introduction of the Unicode characters into the domain name space. The Unicode system consists of many similar-looking characters, called homoglyphs. These characters could be abused by attackers to register domains visually looking similar to a benign domain, in order to lure a user. Although a Unicode character might have both Unicode and ASCII homoglyphs, due to the computational limitations, only ASCII homoglyphs of the Unicode characters are investigated in this paper and a method is proposed which extracts IDNs having a similar-looking ASCII domain. However, not all homoglyph domains are malicious and they might be proactively registered by brand owners to prevent homograph attacks. In order to detect and mark suspicious homoglyph domains, homoglyph domain pairs are investigated more using DNS (Domain Name System) data. The proposed method is based on OpenINTEL [3], [4] active DNS measurements. The main contributions of this paper are: (1) evaluation of the existing Unicode homoglyph tables through introduction of an improved table, (2) using a comprehensive dataset in terms of time span and covering ccTLDs (country-code Top Level Domains), (3) applying the proposed detection method on all existing domains and not limited to a number of brand domains, (4) replacing the utilization of whois data with "ASN" (Autonomous System Number) record and four DNS records, and (5) evaluation of achievable time advantage through early detection of malicious domains.

The remainder of this paper is organized as follows. In Section II, the background of IDN and homoglyph domains are given. Section III discusses the related works in the literature. In Section IV, the proposed methodology is presented. Section V introduces the datasets used in this research. Results of our study are presented in Section VI. Finally, Section VII concludes the paper.

II. BACKGROUND

Domain Name System (DNS) protocol, as one of the cornerstones of the Internet was designed much earlier than the introduction of IDNs, restricted to the utilization of ASCII characters only. In order to keep backward compatibility with DNS protocol in use and avoid upgrading the existing infrastructure, IDNs are first converted into an ASCII-Compatible Encoding (ACE) string which is done using the "Punycode" [5] algorithm. To do so, the Punycode algorithm uses an algorithm called "Bootstring" which keeps all ASCII char-

acters, encodes the location of non-ASCII characters, and re-encodes the non-ASCII characters with generalized variable-length integers. An "xn--" prefix is added to the converted Punycode after the above-mentioned process. Since ACE strings are meaningless for end users, this process is reversed by applications to compute and display the Unicode values. The Unicode-ASCII mapping is dealt with in applications by means of Internationalized Domain Names in Applications (IDNA) leveraging two conversion algorithms "ToASCII" and "ToUnicode" forming Bootstring algorithm together. [6].

The Unicode system incorporates numerous writing systems and languages, in which many similar-looking characters - technically called homoglyphs - such as Greek letter "O" (U+039F), Latin letter "O" (U+004F), and Cyrillic letter "O" (U+041E) are assigned to different code points. The IDN homograph attack is a way that an attacker might impersonate a benign server and lure users about the identity of the server they intend to communicate with, by replacing a character with its homoglyph. This would provide a possibility for malicious usage of the Unicode to perform security attacks since there is little or no visual difference in the glyphs for these characters in most of the fonts. One of the first IDN homograph attack incidents was detected in 2005 [7], where a spoofed PayPal website lured scam victims by replacing the first Latin character "a" (U+0061) by a Cyrillic "a" (U+0430). Although IDN homoglyph attacks have existed for a long time, they still occur frequently nowadays and strong countermeasures are needed to deal with them.

The DNS protocol provides a number of benefits to detect malicious domains such as containing only a small fraction of the overall network traffic, caching feature and unencrypted data. Besides, considering the fact that normally there is a time gap between registration of a malicious domain and the instance when it is actively used to perform an attack [8], [9], it would be possible to reveal attacks at their early stages or even before they happen, due to some traces left in the DNS data. Collection of the DNS data to perform analysis, could be done either actively or passively. There are a number of challenges using any of these methods. In passive measurements, it is not possible to put traffic sensors in a large enough network and achieve a global behaviour of internet traffic. Also due to the sensitivity of the DNS data, it is normally not possible to access the data form public DNS servers. On the other hand active DNS measurements do not present the pattern of real users' behaviour.

III. Related Work

In this section results of studying the existing work is described. Unicode attacks are generally classified in three classes in the literature: *spam attacks*, *web identity attacks* and *phishing attacks*. Although the intention of attackers in these classes are different, the principle behind all is the same, which places these attacks under the Unicode attacks category. Various studies have tried to detect and mitigate IDN homograph attacks, which is the main focus of this research. Fu et al. [10] construct a Unicode Character Similarity List (UC-SimList) in which characters in English, Chinese and Japanese are paired with their visually and semantically similar Unicode characters. In this list, different levels of similarity could be selected by applying a threshold. "UC-SimList" is then used to check validity and similarity of a domain name requested to be registered. Roshanbin et al. [11] propose a method to measure the degree of similarity between Unicode glyphs using the Normalized Compression Distance (NCD) metric, which could be used to build a Unicode character Similarity List.

A number of spoofing defences try to improve the UI of browsers [12], [13]. They implement a client-side anti-phishing extension into the browsers which prints characters of different subsets of Unicode in different colors in the address bar. Also when a domain name consists of both digits and characters, they are printed in different colors, for example to avoid confusions between Latin small letter L "l" and digit one "1", etc.

Alvi et al. [14] propose a method to detect plagiarism in texts when obfuscation is made using the Unicode characters. This is a similar issue to IDN homoglyph domains where the intention is to create visually similar strings which are treated differently by computers. The "Unicode Confusables" list provided by the Unicode Consortium [15] and normalized hamming distance are two basic features used in their research to detect plagiarism. A phishing IRI/IDN pattern generation tool called REGAP is proposed in [16], where a keyword level non-deterministic finite automaton (NFA) is used to identify the potential IRI/IDN-based phishing patterns. This method is able to detect semantic similarity in domain names, however it has to be done manually considering that the number of domains to protect is limited.

In [17], authors propose a phishing domain classification strategy which uses seven domain name based features and models the relationship between the domain name and the visible content of a web page. One of the features for phishing domains used in this research is when a domain name contains non-alphabetical characters (digits and hyphen). However, there are many legitimate domain names which contain such characters. Holgers et al. [18] perform a measurement study by first passively collecting a nine-day-long trace of domain names accessed by users in a department and then generating corresponding confusable domain names. In order to reduce the bias of gathered domain names, the list of Alexa [19] top 500 sites was added to the collected list. In the next step, an active measurement study is performed to check whether those domains are actually registered.

Qiu et al. [20] propose a Bayesian framework to calculate the posterior distribution of a suspicious character in a domain name. If the probability of the suspicious character is above a threshold or maximal among all the probabilities of its homoglyphs, the character is detected as legitimate character, otherwise, as spoofing. In order to further improve results of their method, three extra rules are considered, such as assuming that a legitimate string appears more frequently than the spoofing ones on the web pages.

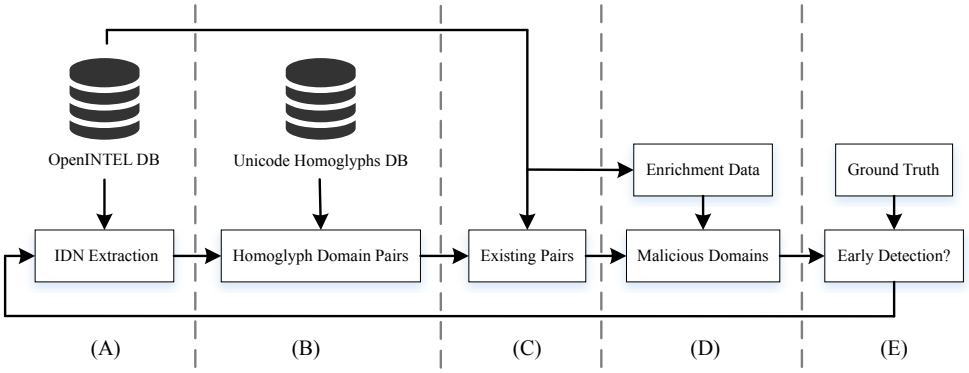


Fig. 1: High-level overview of the proposed method

Elsayed et al. [21] extract newly registered Unicode domains by downloading DNS zone files for ".net" and ".com" TLDs (Top Level Domains), and then replace the Unicode characters by their ASCII homoglyph character from the "Unicode confusables" list to decide whether a domain is meant for phishing. One of the performed tests in their research is to check the IPV4 addresses and registrant organizations of two domains using whois data. If both of these records were different, the Unicode domain is considered to be a phishing domain. Several issues happen to be problematic when it comes to usage of whois data to differentiate between benign and malicious domains. First of all, there are many domains with masked whois data for privacy reasons, which might result in a high number of False Positives. Second, there is a possibility to spoof whois fields of a benign domain, which in turn hides malicious domain from being detected and reduce the number of True Positives. Another drawback of using whois data to differentiate between malicious and benign domains is that, on average malicious domains have a short lifetime. Thus, it is not possible to access their whois data when studying historical DNS measurements. Besides, parsing failures of the whois crawler might also happen as the case in [22] where roughly 50% of whois data are successfully obtained.

Although the existing research discussed in this section, addresses detection of malicious IDN domains to some extent, to the best of authors' knowledge there is no research which has applied these methods on a big dataset (including ccTLDs) and the DNS data used in these researches could offer only a limited local view of the threats. Besides, most of the researches done to detect IDN homograph attacks, focus on highly reputed brand names such as social media domain names, and thus are unable to achieve a global perspective of the existing problem. Finally, the maturity of different Unicode homoglyph tables is not studied yet and there is no measure of how useful these tables are. The proposed method to address these issues is discussed in the next section.

IV. METHODOLOGY

Our proposed method to address the drawbacks mentioned in the previous section, are discussed here. A high-level view of the proposed detection mechanism for malicious IDN homoglyphs is depicted in Fig. 1, divided in five major steps (A) to (E). These steps are summarized as follows:

(A) Gathering IDNs from OpenINTEL database which are domains containing at least one Unicode character.
(B) Processing queried domains using various Unicode homoglyph tables to form homoglyph domains pairs.
(C) Querying homoglyph domain pairs from OpenINTEL to discard non-existing pairs.
(D) Decision making on likelihood of IDN to be malicious using enrichment data gathered from OpenINTEL.
(E) Looking up extracted domains on existing blacklists to quantify achievable time advantage through early detection of malicious domains.

The above mentioned process is repeated on a daily basis. Details of these steps are elaborated in the following.

A. IDN Extraction

The proposed method starts with extraction of IDNs from our databases. As discussed in Section II, all IDNs start with a "xn--" prefix, which makes it easy to separate these domains from the rest. The objective here is to extract IDNs in OpenINTEL measurements which are domain names consisting of one or more Unicode characters and then acquire their growth rate. OpenINTEL DNS measurements are used as the data source in this research. Currently, OpenINTEL platform captures daily DNS measurements for all domains under the main generic TLDs (including .com, .net and .org, comprising approximately 50% of the global DNS name space) and 12 country-code TLDs, as well as Alexa top 1 million, Infrastructure measurements and Cisco Umbrella top 1 million domains. This results in approximately 217 million domains measured on a daily basis. Due to the comprehensive coverage of domain names, OpenINTEL provides a suitable dataset to investigate existence of malicious IDN homoglyph domains and is used in this research.

According to the 2018 IDN report provided by European Registry for internet domains (EURid) [23], there were approximately 6 and 7.5 million IDNs by the end of 2014 and 2017 respectively, which counts for approximately 2%

Table I: Tables of homoglyph Unicode character

	Unicode Confusables	**UC-SimList0.8**	**Proposed table**
Total characters	6296	29880	2627
Characters with an ASCII homoglyph	2236	536	2627
Character to string mapping	✓	-	✓

of entire domain name space. In 2017 roughly 70% of these domains were registered on a ccTLD, which verifies the importance of exploring ccTLDs for malicious homoglyphs.

B. Homoglyph Domains

In this step, extracted IDNs are investigated to form homoglyph domain pairs. Although there are many homoglyphs in the Unicode system, there is no absolute way to determine whether two characters are similar. The list of confusable characters published by the Unicode Consortium (referred to as "Unicode Confusables" in this paper) consists of a list of 6296 pairs of homoglyphs in its 12.0.0 version. This list makes it possible to investigate IDN homoglyph domains. Besides, the "UC-SimList0.8" [10], [24] is used as a complementary input to investigate homoglyph domains. Although a Unicode character might have both a Unicode and an ASCII homoglyph, due to the computational limitations, only ASCII homoglyphs of Unicode characters are considered to extract homoglyph domain pairs in this paper. We have noticed some irregularities in the "UC-SimList0.8". For example Latin small letter dotless I "ı" (U+0131) is considered as a homoglyph to exclamation mark "!" (U+0021), however Latin small letter I "i" would be a better choice as we rarely find a label including exclamation mark.

Quality of the above-mentioned confusion tables is explored more in this research by introduction of a third table, which is a mix of the "Unicode Confusables" and the "UC-SimList0.8" by replacing irregularities and adding a number of missing characters. The proposed table is publicly available online[1]. Specifications of each confusion table is given in Table I. Comparing two existing homoglyph tables, it is seen that although the "UC-simList0.8" contains more characters in total, it covers much less characters with an ASCII homoglyph than the "Unicode Confusables". This is because the "UC-SimList0.8" covers many characters from Chinese and Japanese alphabet for which an ASCII homoglyph does not exist. Another major difference between these tables is that the "Unicode Confusables" provides homoglyph strings for Unicode characters that can not be replaced by a single character. A frequently observed example on ccTLDs is the Latin small letter AE "æ" (U+00E6) which could be replaced by the Latin small letter A "a" (U+0061) plus the Latin small letter E "e" (U+0065). However, this feature is not supported on the "UC-SimList0.8". On the other hand, the "UC-SimList0.8" provides multiple homoglyphs for each Unicode character (when they exist), ordered by their degree of similarity. For simplicity we only use the ASCII homoglyph with the highest similarity score for each Unicode character in this paper. In the next step, the Unicode characters present in each IDN domain are replaced by an ASCII homoglyph using each of the three confusion tables to create an ASCII homoglyph counterpart domain. The performance of these three tables is compared in Section VI-A.

[1]https://www.tide-project.nl/blog/noms2020/

C. Existing Homoglyph Pairs

The extracted homoglyph domains only represent a list of potential pairs. Since ASCII homoglyph domains are created by replacing the Unicode characters with their homoglyph counterpart, we do not expect all of these domains to exist. These pairs are checked across the OpenINTEL database and when an ASCII homoglyph domain does not exist, the pair is discarded.

D. Malicious Homoglyph Detection

In order to detect malicious domains, the list of homoglyph domain pairs created in the previous step are compared for "ASN" record and four DNS records namely "A" (IPv4 address), "AAAA" (IPv6 address), "NS" (Name Server) and "MX" (Mail eXchanger) queried from OpenINTEL database. Based on these information a score of maliciousness is calculated. If any of these records were different for two domains, the score is incremented by one. Thus, the maliciousness score will be an integer in [0,5]. Finally a decision is made on whether the IDN domain is highly likely to be meant for malicious activity if the score was higher than a threshold. This threshold value has to be selected based on a trade-off between desired True Positive and False Positive rates. Utilization of the whois data to differentiate between malicious and benign domains is replaced with DNS records and "ASN" record to overcome the drawbacks regarding the use of whois data mentioned in Section III.

E. Time Advantage

Malicious domains on average have a shorter lifetime compared to benign domains [25], [26]. A main reason for this behaviour is that their registrants want to keep their activity hidden from detection systems. This implies that there is a limited time to perform reactions against attacks which use these domains. To qualify the achievable time advantage by detecting malicious IDN homoglyphs before they appear on blacklists, historical OpenINTEL data was used to detect malicious IDN homoglyph domains and study the time gap between their detection and appearance on existing blacklists. The blacklists used in this research include "Hostfile" [27], "hpHosts" [28], "Ransomwaretracker" [29], "DNS-BH" [30], "Openphish" [31], "Malwaredomainlist" [32], "Joewein"

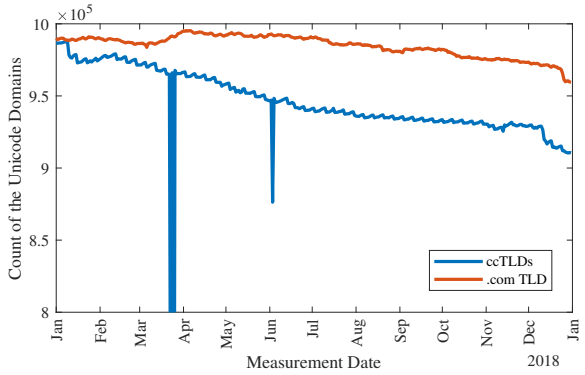


Fig. 2: IDNs in ".com" TLD and ccTLDs in 2018

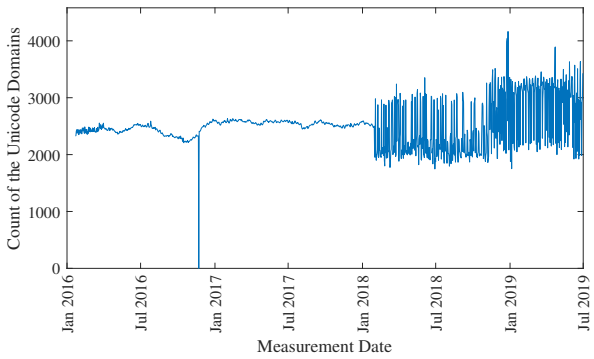


Fig. 3: IDNs in Alexa top 1M domains

[33], "Feodotracker" [34], "Zeustracker" [35], "Malcode" [36], "Squidblacklist" [37], "C2Dom" [38], "Urlhaus" [39], "Cybercrimetracker" [40], "Dshield" [41], "VXVault" [42], "UT-Capitole" [43], "Ponmocup" [44] and "Urlvir" [45]. In the rest of this paper this set of blacklists is refered to as "RBL".

V. DATASETS

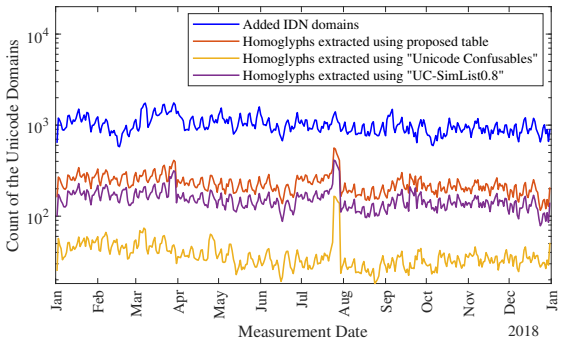
To form the input datasets of this research, a query of Unicode domains present in ".com" and "ccTLDs" is done on a daily basis from 1st Jan. 2018 till 31th Dec. 2018. The "ccTLDs" group used in this research include ".se", ".nu", ".ca", ".fi", ".at", ".dk", ".ru", ".рф" and ".us". The measurement databases for different TLDs have been added on OpenINTEL at various points in time. To avoid confusion in the results and make the plots easy to read, the time span of our dataset is selected such that the TLDs under study are not added during this period. Comparing the lists of domains for each pair of consecutive days, the list of domains registered on a specific date are extracted, which are then processed using three Unicode confusion tables mentioned in Section IV-B to detect domains with a potential ASCII homoglyph domain. For these pairs of domains, five records mentioned in Section IV-D are obtained from OpenINTEL to perform the detection procedure.

VI. RESULTS

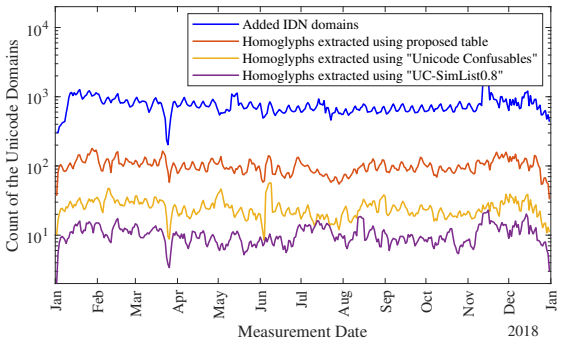
Before applying our detection method to extract suspicious domains, some numerical details about the dataset explained in Section V are discussed here. Growth of IDNs in ".com" and "ccTLDs" are depicted in Fig. 2. This plot shows a negative growth trend for IDNs, which has a steeper descent in "ccTLDs". Besides, three deeps are visible for "ccTLDs" which after closer investigation turned out to be caused by a failure in the OpenINTEL measurements. A weekly pattern in "ccTLDs" is visible in which the number of domains grows monotonically till Mondays and drops on Tuesdays. This is caused by ".рф" ccTLD and might be due to the de-registration policy of the expired domains applied by corresponding registrars. However, exact reasoning on this pattern needs more investigation and is out of scope of this research. Considering the total number of 7.5 million IDNs on Dec. 2017 reported by EURid, our dataset includes roughly 26% of all registered Unicode domains. A big portion of remaining IDNs include characters of Eastern Asian languages for which an ASCII homoglyph does not exist. Thus our dataset covers a significant part of IDNs that could be used for homograph attacks. Fig. 3 depicts the number of IDN domains in Alexa top 1M from 22nd Jan. 2016 till 30th Jun. 2019. As seen in this figure, the number of Unicode domains in Alexa top 1M exhibits extreme oscillations from 01st Feb. 2018 onward. This is mainly caused by the number of IDNs in ".рф" ccTLD and ".com" TLD which end up on Alexa. Regardless of these oscillations the average number of IDNs on Alexa stays constant during this time period of 3.5 years of observation, which approximately contributes to a percentage of 0.2% on Alexa 1M. Comparing this to the total IDN count on entire name space reveals that IDNs are much less popular than ASCII domain names among the Internet users. Our database shows no IDNs on Alexa 1M on 22 Nov. 2016, which is due to a measurement failure.

A. Comparison of Confusion Tables

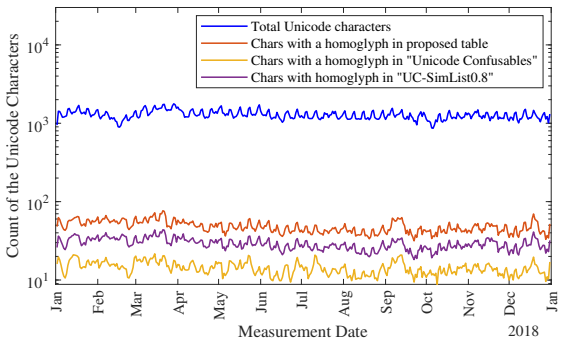
Extraction of homoglyph domain pairs is done using the three Unicode confusion tables discussed in Section IV-B for each dataset. Results for ".com" and "ccTLDs" are plotted in Fig. 4. In order to better represent the difference between three homoglyph tables and make plots readable, rather than absolute values, curves are smoothed using a moving average filter and the y-axis of these figures are log-scaled. Fig. 4a shows that on average using the "UC-SimList0.8", we are able to extract three times more homoglyph domains than using the "Unicode Confusables" for domains in ".com" TLD. The proposed confusion table is able to extract more domains than two existing tables (5 times more than "Unicode Confusables" and 0.5 times more than "UC-SimList0.8"), since it covers both tables plus some missing characters. Considering the number of characters present in different confusion tables mentioned in Table I, one would have expected that the "Unicode Confusables" must be able to extract more domains compared to the "UC-SimList0.8", which turns out not always to be a correct assumption. This is mostly because the



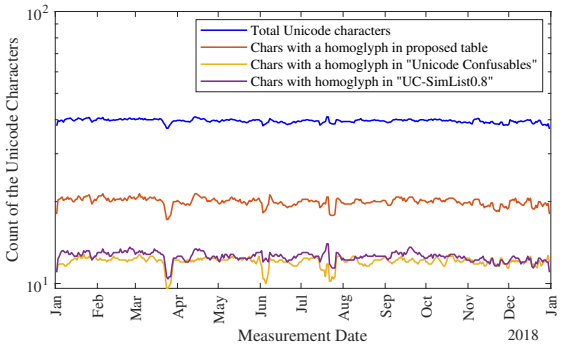
(a) Extracted homoglyphs for added IDNs in ".com"



(b) Extracted homoglyphs for added IDNs in "ccTLDs"



(c) Characters with a homoglyphs for added IDNs in ".com"



(d) Characters with a homoglyphs for added IDNs in "ccTLDs"

Figure 4: Extracted homoglyph domains and characters for added IDNs in ".com" and "ccTLDs"

"Unicode Confusables" includes many punctuation characters which rarely appear on a domain name.

The performance of various homoglyph tables on "ccTLDs", is plotted in Fig. 4b. We notice that the proposed confusion table is able to extract 3 times more homoglyph pairs than "Unicode Confusables" and 8.7 times more than "UC-SimList0.8". Comparing the "Unicode Confusables" and "UC-SimList0.8" reveals that despite the case of domains in ".com", the "Unicode Confusables" achieves a better performance in "ccTLDs" by extracting 1.4 times more homoglyph domains compared to the "UC-SimList0.8". This happens because many domains in each ccTLD share a limited number of Unicode characters which are covered by the "Unicode Confusables". It is noteworthy that in Fig. 4b we neglect three aforementioned measurement failures of Fig. 2 for "ccTLDs", so that our results are not negatively affected. Besides, we notice that although a small set of missing characters are added in our proposed table, this makes a big difference in the number of extracted domains. This implies that there are frequently used Unicode characters which are not covered by existing tables. This could be quantified by looking at the count of the Unicode characters present in added IDNs per day and comparing it to the number of characters covered using each homoglyph table. Fig. 4c and Fig. 4d plot the number of total Unicode characters and the Unicode characters covered by each table for IDNs added per day in ".com" and "ccTLDs" respectively. It is seen from these figures that although the "UC-SimList0.8" outperforms the "Unicode Confusables" for domains in ".com" TLD, there is negligible difference between the performance of these two tables in "ccTLDs". Also, we noticed that despite a slightly higher number of characters with a homoglyph in the "UC-SimList0.8" for "ccTLDs", these characters yield in detection of a smaller set of homoglyph domain pairs as seen in Fig. 4b. This implies that the "Unicode Confusables" provides homoglyphs which are frequently seen on "ccTLDs".

B. Detection results

Extracted homoglyph domains produced in the previous step are fed into the detection module where "ASN" record and four DNS records are queried form OpenINTEL as enrichment data and a score of maliciousness is calculated as the summation of the count of different records for these domains (see Section IV-D for more details. Based on this score domains are marked as suspicious if the score is higher than a threshold. A high threshold value results in marking a lower number of domains as suspicious and consequently lower false positive rate. On the other hand with a low threshold more domains will be detected resulting in a higher false positive rate. Since marking a benign domain as malicious might be much more costly than not detecting a suspicious one, threshold value has to be

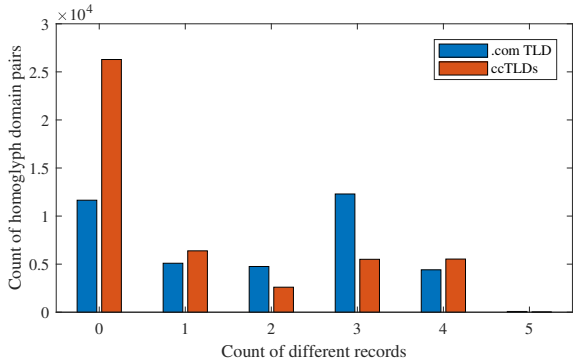


Fig. 5: Scores for extracted domains

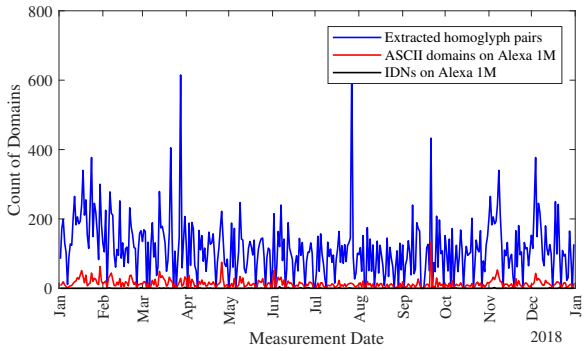


Fig. 6: Extracted domains on Alexa top 1M

selected so that the number of false positives is minimized. This threshold could be selected once a look up of the detected domains on the existing blacklists is performed. This is discussed in Section VI-E. However, not all domains include an entry for all 5 records we are looking at, causing them not to achieve a high score by our method and consequently not being marked as a suspicious domain. For example only 2% of homoglyph domain pairs both have an "AAAA" record among our extracted domains from ".com" TLD. In order to limit the number of false positives in our detection method, a record is counted as different only if both domains have an entry for it.

The distribution of records difference score is plotted in Fig. 5. It is seen that for "ccTLDs" a big portion of the homoglyphs have no difference in records, while for ".com" there is a large group of domains with three different records. This suggests a relatively higher chance of malicious intent behind extracted IDN domains in ".com" compared to "ccTLDs".

C. Top Targeted domains

In order to get a better insight on the domains which are highly targeted by IDN homoglyphs, we have listed extracted ASCII domains based on the number of times a homoglyph IDN domain corresponding to each of them is seen. Results are plotted in Fig. 7. The domain name targeted the most in ".com" TLD, belongs to a crypto-currency service with 238 IDN homoglyphs registered through 2018. Besides, considering 30 highly targeted domains in ".com" TLD, we notice that 18 domains are related to crypto-currency and financial service providers with overall 1051 IDN homoglyph domains. The social media and IT service providers are in the second rank consisting of 7 ASCII domains with overall 325 IDN homoglyph domains. This characteristic is not seen in "ccTLDs" where each ASCII domain has a handful of corresponding IDN homoglyphs.

D. Validation

The validation process for detection methods is usually done based on using different ground truth which consists of both black and white lists. Different blacklists are available to cross check detected domains against them including but not limited to the "RBL" group of blacklists mentioned in Section IV-E.

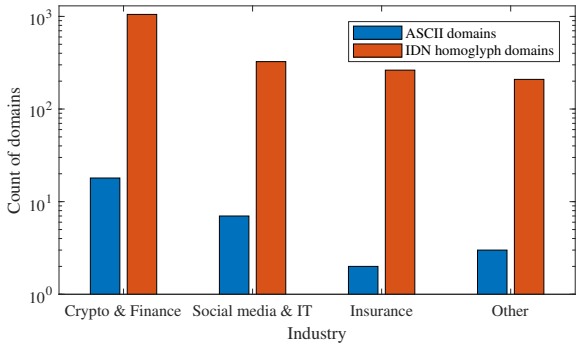


Fig. 7: Top targeted domains on ".com" TLD

However, there might be domains detected by the proposed method which are not yet covered by blacklists. Comparison of the detected domains against blacklists is discussed in Section VI-E. As a white-list, the Alexa top domains is usually utilized. Although not all domains on Alexa are benign, the chance of them being benign is higher than a random set of domains. Neglecting these shortcomings, whitelists and blacklists are still useful to a high extent to validate the proposed method. Results of looking up the detected IDN homoglyph pairs on the Alexa 1M is represented in Fig. 6. It is seen that IDN homoglyph domains rarely end up on the Alexa 1M (the curve for IDNs on the Alexa 1M has very small values and sticks to x-axis), however a higher number of their ASCII counterparts appear on the Alexa. The difference gets bigger when we consider that each ASCII domain potentially corresponds to multiple homoglyph pairs as detailed in Section VI-C.

Due to the lack of sufficient proof, especially for domains that are detected by the proposed method and have not appeared on the blacklists yet, rather than being marked as malicious, they are considered as highly suspicious domains in this research. The proposed method could be used in parallel with other methods to further investigate the malicious intent behind these domains.

E. Time advantage

In this section the achieved time advantage for proposed detection approach over a set of existing blacklists called

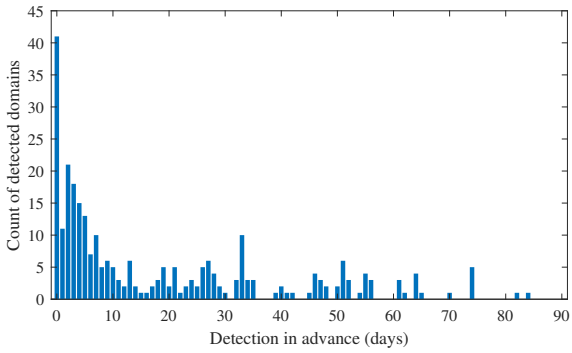


Fig. 8: Early detection compared to RBL blacklist

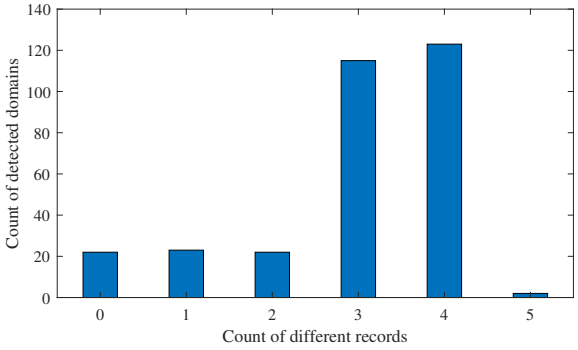


Fig. 9: Scores for domains appeared on RBL

"RBL" is investigated. This time advantage is calculated as the window (in days) between detection of malicious domains by the proposed method and the time at which these domains appears on one of the aforementioned blacklists. Detected domains over 2018 have been checked in RBL from 1st Jan. 2018 till 19th Jun. 2019. 307 domains out of entire extracted domains in our dataset, have been detected by at least one of the blacklists in RBL during this period. Although this number is pretty low compared to the number of domains detected by the proposed method, it signifies that our method detects a specific subset of malicious domains which have been given free room to a high extent. Besides, 305 of these domains are ones detected in ".com" TLD and only 2 domains are from "ccTLDs". This reveals that detection procedures implemented by blacklists have a higher concentration on the generic TLDs and it needs to be improved to better cover ccTLDs. However, as already discussed in Fig. 5 the probability for an IDN homoglyph in "ccTLDs" to be malicious is lower than ".com". Fig. 8 depicts the early detection in days for domains detected in ".com" and "ccTLDs" compared to the appearance of these domains on the RBL blacklist set, limited to first 90 days. Four different categories have been defined based on the time difference between registration of a domain and its detection by the blacklists as follows:

1) domains detected on the same day as their registration
2) domains detected at least a day after their registration and at most in a month
3) domains detected with a difference between a month and a year
4) domains detected after a year

Out of 307 domains detected by blacklists, 41 domains (13.4%) fall in the first category, 169 in the second (55%), 89 in the third (29%) and 8 domains in the fourth category (2.6%). On average 42 days of early detection is achieved using the proposed method considering the domains that are already blacklisted. The maliciousness score distribution for domains detected by blacklists is plotted in Fig. 9. 78% of these domains have achieved a score of 3 or higher. besides, the reason for majority of remaining domains achieving a lower score is that they only had a limited number of record entries. Considering these results, a threshold value of 3 to mark a domain as highly suspicious seems logical.

VII. Conclusions

The introduction of Internationalized Domain Names has enabled local users register and access domain names using their regional languages leveraging the Unicode character system. Beside the convenience provided by this feature, a new type of threat is introduced in which an attacker is able to lure users by replacing an ASCII character by its similar-looking Unicode character called a homoglyph. In order to study homoglyph IDNs in a large scale, active DNS measurements of OpenINTEL for ".com" TLD and 9 ccTLDs from 1st Jan. 2018 till 31th Dec. 2018 are used as input database in this research.

Although there are many homoglyphs in the Unicode system, there is no absolute way to determine whether two characters can be considered similar. However, there are studies in the literature which aim to provide a list of Unicode homoglyph tables. In this paper performance of two existing Unicode homoglyph tables namely the "Unicode Confusables" and the "UC-SimList0.8" is compared to a proposed homoglyph table. Results of this research show that the proposed table outperforms existing homoglyph tables and is able to extract more homoglyph domains. As a detection strategy, extracted homoglyph domains are compared for "ASN" record and four DNS records including "A", "AAAA", "NS" and "MX" records. Based on these records a score of maliciousness is calculated. If this score is higher than a threshold there is high chance that the IDN is meant for some malicious activity. Cross checking detected domains across existing blacklists shows that majority of the domains that are already detected by a blacklist have 3 or more different records than their ASCII counterpart. Besides, comparing the time these domains are detected using the proposed method with the first time they appear on one of the blacklists, on average an early detection of 42 days is achieved.

Acknowledgment

The research leading to the results presented in this paper was made possible by OpenINTEL, a joint project of the University of Twente, SURFnet and SIDN.

REFERENCES

[1] "The Unicode Consortium," https://www.uincode.org/.
[2] M. J. Dürst, "Urls and internationalization." World Wide Web Consortium, 1996.
[3] "OpenINTEL," https://www.openintel.nl/.
[4] R. van Rijswijk-Deij, M. Jonker, A. Sperotto, and A. Pras, "The internet of names: A dns big dataset," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 91–92, 2015.
[5] A. Costello, "Punycode: A bootstring encoding of unicode for internationalized domain names in applications (idna)," Tech. Rep., 2003.
[6] P. Falstrom, P. Hoffman, and A. Costello, "Internationalizing domain names in applications (idna)(RFC 3490)," *Cisco, IMC & VPNC, University of California, USA*, 2003.
[7] E. Johanson, "IDN hacking disclosure." ShmooCon, 2005.
[8] E. Kidmose, E. Lansing, S. Brandbyge, and J. M. Pedersen, "Detection of malicious and abusive domain names," in *2018 1st International Conference on Data Intelligence and Security (ICDIS)*. IEEE, 2018, pp. 49–56.
[9] S. Hao, N. Feamster, and R. Pandrangi, "Monitoring the initial dns behavior of malicious domains," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 269–278.
[10] A. Y. Fu, X. Deng, L. Wenyin, and G. Little, "The methodology and an application to fight against unicode attacks," in *Proceedings of the second symposium on Usable privacy and security*. ACM, 2006, pp. 91–101.
[11] N. Roshanbin and J. Miller, "Finding homoglyphs-a step towards detecting unicode-based visual spoofing attacks," in *International Conference on Web Information Systems Engineering*. Springer, 2011, pp. 1–14.
[12] V. Krammer, "Phishing defense against idn address spoofing attacks," in *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services*. ACM, 2006, p. 32.
[13] J. Al Helou and S. Tilley, "Multilingual web sites: Internationalized domain name homograph attacks," in *2010 12th IEEE International Symposium on Web Systems Evolution (WSE)*. IEEE, 2010, pp. 89–92.
[14] F. Alvi, M. Stevenson, and P. Clough, "Plagiarism detection in texts obfuscated with homoglyphs," in *European Conference on Information Retrieval*. Springer, 2017, pp. 669–675.
[15] "Unicode confusables list," https://unicode.org/Public/security/.
[16] A. Y. Fu, X. Deng, and L. Wenyin, "Regap: A tool for unicode-based web identity fraud detection," *Journal of Digital Forensic Practice*, vol. 1, no. 2, pp. 83–97, 2006.
[17] H. Shirazi, B. Bezawada, and I. Ray, "Kn0w thy doma1n name: Unbiased phishing detection using domain name based features," in *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*. ACM, 2018, pp. 69–75.
[18] T. Holgers, D. E. Watson, and S. D. Gribble, "Cutting through the confusion: A measurement study of homograph attacks." in *USENIX Annual Technical Conference, General Track*, 2006, pp. 261–266.
[19] "Alexa top sites," http://aws.amazon.com/alexa-top-sites/.
[20] B. Qiu, N. Fang, and L. Wenyin, "Detect visual spoofing in unicode-based text," in *2010 20th International Conference on Pattern Recognition*. IEEE, 2010, pp. 1949–1952.
[21] Y. Elsayed and A. Shosha, "Large scale detection of idn domain name masquerading," in *2018 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 2018, pp. 1–11.
[22] B. Liu, C. Lu, Z. Li, Y. Liu, H.-X. Duan, S. Hao, and Z. Zhang, "A reexamination of internationalized domain names: The good, the bad and the ugly." in *DSN*, 2018, pp. 654–665.
[23] "EURid 2018 report," https://idnworldreport.eu/2018-2/.
[24] "Unicode Similarity List," http://people.csail.mit.edu/ayf/IRI/UCSimList/UCSimList/UC_SimList0.8.txt.
[25] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive dns analysis." in *Ndss*, 2011, pp. 1–17.
[26] "The distribution of malicious domains." The domaintools report, 2016.
[27] "Hostfile," https://hosts-file.net/hphosts-partial.txt.
[28] "hpHosts," https://hosts-file.net/download/hosts.txt.
[29] "Ransomwaretracker," https://ransomwaretracker.abuse.ch/downloads/RW_DOMBL.txt.
[30] "DNS-BH – Malware Domain Blocklist by RiskAnalytics," http://mirror1.malwaredomains.com/files/justdomains.
[31] "OpenPhish," https://www.openphish.com/.
[32] "Malwaredomainlist," http://www.malwaredomainlist.com/hostslist/hosts.txt.
[33] "Joewein," http://www.joewein.net/dl/bl/dom-bl.txt.
[34] "Feodotracker," https://feodotracker.abuse.ch/blocklist/?download=domainblocklist.
[35] "Zeustracker," https://zeustracker.abuse.ch/blocklist.php?download=domainblocklist.
[36] "Malc0de," http://malc0de.com/bl/ZONES.
[37] "Squidblacklist," https://www.squidblacklist.org/downloads/dg-malicious.acl.
[38] "C&C domain," http://osint.bambenekconsulting.com/feeds/c2-dommasterlist.txt.
[39] "Urlhaus," https://urlhaus.abuse.ch/downloads/text/.
[40] "Cybercrimetracker," http://cybercrime-tracker.net/all.php.
[41] "Dshield," https://dshield.org/feeds/suspiciousdomains_Medium.txt.
[42] "Vxvault," http://vxvault.net/URL_List.php.
[43] "UT-Capitole," http://dsi.ut-capitole.fr/blacklists/download/.
[44] "Ponmocup," http://security-research.dyndns.org/pub/malware-feeds/ponmocup-infected-domains-shadowserver.csv.
[45] "Urlvir," http://www.urlvir.com/export-hosts/.

Appendix B

Thesis Proposal

This appendix contains the thesis proposal defined at the starting phase of the graduation assignment. It summarizes the literature review of the topic as well as definition of the research questions and initial methodology to find proper answers to these questions. Initial sections of this proposal overlap with NOMS paper given in Appendix A. The last section of this proposal defines the time plan of the project.

Detection of IDN Homoglyph phishing in OpenINTEL DNS Measurements

Ramin Yazdani
r.yazdani@student.utwente.nl
University of Twente

ABSTRACT

The possibility to include Unicode characters in domain names lets local users to deal with domains in their regional languages, which is done through introduction of Internationalized Domain Names (IDN). Due to the visual similarity of Unicode characters in different languages called homoglyphs, a new type of potential threat has been of concern. The IDN homograph attack is a way that an attacker might impersonate a benign server by replacing one or more characters by their homoglyph. Domain Name System (DNS) protocol - being an essential part of the Internet - is beneficiary to detect domain spoofing due to its characteristics. Considering the time gap between registration of a malicious domain and actively using it to perform an attack, it would be possible to reveal attacks at their infancy or even before they happen using the traces left in the DNS data. OpenINTEL is a platform which performs active DNS measurements, covering a large portion of the entire DNS name space. The database provided by OpenINTEL will be used in this project to detect IDN homographs in DNS traffic.

KEYWORDS

Homoglyph, IDN, Homograph attacks, Phishing, DNS measurements, OpenINTEL

1 INTRODUCTION

In order to store and represent characters of Latin alphabet in computer systems ASCII encoding standard was developed which maps each character to an 8 bits string of zeros and ones. However, expanding this encoding scheme and creating a unified character system was necessary to include all characters of different regional languages. Unicode standard [1] was introduced to solve this issue by mapping each character to 16 bits (32 bits in the latest version). Internationalized domain name (IDN), originally proposed in 1996 [2], is the term used for a domain name in the Internet, containing at one or more labels in a language-specific script or alphabet, such as Greek, Cyrillic, Arabic, Chinese or the Latin-based characters with diacritics, etc by making use of Unicode characters.

Domain Name System (DNS) protocol, as the core component of the Internet was designed much earlier than the introduction of IDNs, restricted to use of ASCII characters only. In order to keep backward compatibility with DNS protocol in use and avoid upgrading the existing infrastructure, IDNs are applied to DNS protocol as ASCII strings using Punycode algorithm. Thus, Unicode-ASCII mapping is dealt with in applications by means of Internationalized domain names in Applications (IDNA) leveraging two conversion algorithms "ToASCII" and "ToUnicode" [3].

Unicode system incorporates numerous writing systems and languages, in which many similar-looking characters - technically called homoglyphs - such as Greek letter "O" (U+039F), Latin letter "O" (U+004F), and Cyrillic letter "O" (U+041E) are assigned to different code points. The IDN homograph attack is a way that an attacker might impersonate a benign server and deceive users about the identity of the server they intend to communicate with by replacing a character with its homoglyph. An example of such an attack would be the replacement of the Latin character "a" (U+0061), with the Cyrillic character "a" (U+0430) in "www.example.com" to register a new domain. This would provide a possibility for malicious usage of Unicode to perform security attacks since there is no visual difference in the glyphs for these characters in most of the fonts. One of the first IDN homograph attack incidents was detected in 2005 [4], where a spoofed PayPal website lured scam victims by replacing the first Latin character "a" (U+0061) by a Cyrillic "a" (U+0430).

Detection of IDN spoofing domains could be done at various nodes of the network such as registries, registrars, DNS, web browsers, etc. DNS protocol being the core component of the Internet, provides a number of benefits to detect malicious domains, compared to other approaches. Containing only a small fraction of the overall network traffic, together with caching feature of DNS, provides a significant decrease in the amount of data to be analyzed. On the other hand, a large portion of DNS traffic is unencrypted compared to other protocols, making it possible to perform easier inspection. Considering the fact that normally there is a time gap between registration of a malicious domain and the instance when it is actively used to perform an attack [5, 6], it would be possible to reveal attacks at their early stages or even before they happen, due to some traces left in the DNS data.

DNS data needed to perform analysis, could be collected either actively or passively. In order to actively obtain DNS data, a data collector would deliberately send DNS queries and record the corresponding DNS responses. Typical lists to query include popular domains lists such as the Alexa Top Sites [7], zone files of authoritative servers and various blacklists. On the other hand, passive DNS data collection is done by deploying sensors in front of DNS servers or by having access to DNS server logs to obtain DNS queries and responses created by actual users. There are a number of challenges regarding DNS data collection. In passive measurements, it is not possible to put traffic sensors in a large enough network and achieve a global behaviour of internet traffic. Also due to the sensitivity of DNS data, it is normally not possible to access the data form public DNS servers. On the other hand active DNS measurements might not present the pattern of real users' behaviour.

OpenINTEL [8, 9] is a high-performance infrastructure which performs Internet-scale active measurements, currently querying over 50% of the DNS name space on a daily basis. It is designed from the ground up to enable big data analysis approaches on,

e.g., a Hadoop cluster. By performing active measurements, rather than passively collecting DNS data, it build consistent and reliable time series of the state of the DNS. Due to the comprehensive coverage of domain names, OpenINTEL provides a suitable dataset to investigate existence of IDN phishing and will be used in this project.

A general problem statement for this research would be to determine whether a domain name is a phishing or benign one. To pursue the goals of this research, the following research questions (RQ) are defined as the basis:

- **RQ1:** How large could be the problem of IDN phishing?
- **RQ2:** How could we detect whether an IDN is a malicious one and how high is the performance of detection method?
- **RQ3:** Can we detect malicious IDNs before they appear on blacklists and if so, what is the achievable time advantage?

The remainder of this proposal is organized as follows. Section 2 will discuss the related works in the literature. In section 3, the strategies to answer research questions are discussed. Finally, a planning for the thesis structure is given in Section 4.

2 RELATED WORK

In this section results of studying existing work is described. Unicode attacks are generally classified in three classes: *spam attacks*, *web identity attacks* and *phishing attacks*. Although the intention of attackers in these classes are different, the principle behind all is the same, which places these attacks under Unicode attacks category. Various studies have tried to detect and mitigate IDN phishing attacks, which is the main focus of this research. Fu et al. [10] construct a Unicode Character Similarity List (UC-SimList) in which characters in English, Chinese and Japanese are paired with their visually and semantically similar Unicode characters. In this list, different levels of similarity could be selected by applying a threshold. UC-SimList is then used to check validity and similarity of a domain name requested to be registered. Roshanbin and Miller [11] propose a method to measure the degree of similarity between Unicode glyphs using the Normalized Compression Distance (NCD) metric, which could be used to build a Unicode character Similarity List.

A number of spoofing defences try to improve the UI of browsers [12], [13]. They implement a client-side anti-phishing extension into the browsers which prints characters of different subsets of Unicode in different colors in the address bar. Also when a domain name consists of both digits and characters, they are printed in different colors, for example to avoid confusions between small letter L “l” and digit one “1”, etc.

Alvi et al. [14] propose a method to detect plagiarism in texts when obfuscation is made using Unicode characters. This is a similar issue to IDN phishing where the intention is to create visually similar strings which are treated differently by computers. Unicode confusables list provided by Unicode Consortium [15] and normalized hamming distance are two basic features used in their research to detect plagiarism. A phishing IRI/IDN pattern generation tool called REGAP is proposed in [16], where a keyword level non-deterministic finite automaton (NFA) is used to identify the potential IRI/IDN-based phishing patterns. This method is able to detect semantic similarity in domain names, however it has to be done manually considering that the number of domains to protect is limited.

In [17], authors propose a phishing domain classification strategy which uses seven domain name based features and models the relationship between the domain name and the visible content of a web page. One of the features for phishing domains used in this research is when a domain name contains non-alphabetical characters (digits and hyphen). However, there are many legitimate domain names which contain such characters. Holgers et al. [18] perform a measurement study by first passively collecting a nine-day-long trace of domain names accessed by users in a department and then generating corresponding confusable domain names. In order to reduce the bias of gathered domain names, the list of Alexa top 500 sites was added to the collected list. In the next step, an active measurement study is performed to check whether those domains are actually registered.

Qiu et al. [19] propose a Bayesian framework to calculate the posterior distribution of a suspicious character in a domain name. If the probability of the suspicious character is above a threshold or maximal among all the probabilities of its homoglyphs, the character is detected as legitimate character, otherwise, as spoofing. In order to further improve results of their method, three extra rules are considered, such as assuming that a legitimate string appears more frequently than spoofing ones on web pages. Elsayed and Shosha [20] extract newly registered Unicode domains by downloading DNS zone files for .net and .com TLDs, and then replace the Unicode characters by their ASCII similar character from Unicode confusables list to decide whether a domain is meant for phishing. one of the performed tests is to check the IP addresses that host two domains. If they were different, the Unicode domain is considered to be a phishing domain.

Although the existing research discussed in this section, addresses IDN phishing detection to some extent, to the best of author’s knowledge there is no research which has applied these methods on a big data set (including ccTLDs) and the DNS data used in these researches could offer only a limited local view of global threats. In order to address this issue, the proposed method is discussed in the next section.

3 METHODOLOGY

In order to address drawbacks mentioned in the previous section, OpenINTEL measurement data will be used in this research. Currently, OpenINTEL measurement platform captures daily DNS measurements for all domains under the main generic TLDs (including .com, .net and .org, comprising approximately 50% of the global DNS name space) and ten country-code TLDs, as well as Alexa top 1 million, Infrastructure measurement and Cisco Umbrella top 1 million domains. This results in 216 million domains measured in a daily basis. A high level view of the proposed IDN phishing detection mechanism is depicted in Figure 1, covering three research questions mentioned in section 1. These research questions are discussed briefly in the following.

3.1 Growth of IDN

According to the 2018 report of EURid [21] there were approximately 6 and 7.5 million IDNs by end of 2014 and 2017 respectively,

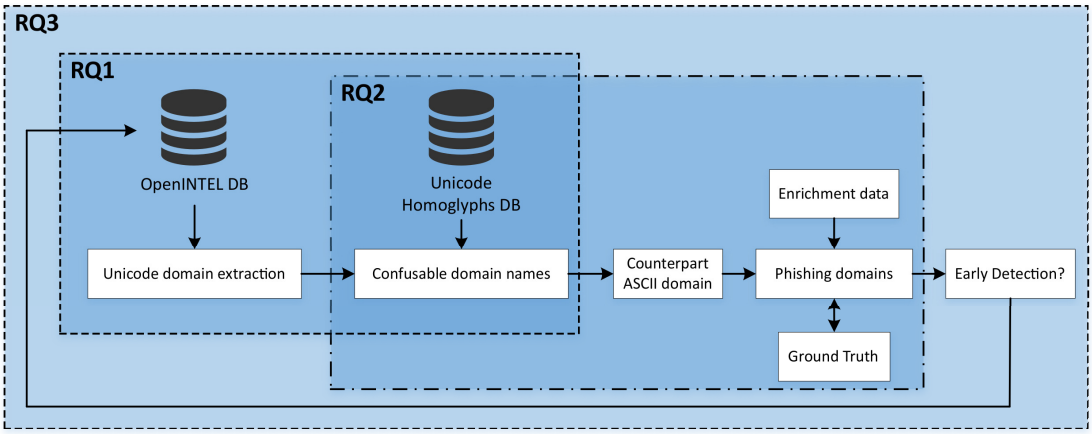


Figure 1: Proposed IDN phishing detection method

which counts for approximatley 2% of entire domain name space. Although there are many homoglyphs in the Unicode system, there is no absolute way to determine whether two characters are similar. The list of confusable characters published by Unicode Consortium consists of a list of 6296 pairs of homoglyphs in its 12.0.0 version. This provides an almost complete list to investigate IDN phishing. Besides, the UC-SimList proposed by Fu et al. [10] considering different levels of similarity, could be used as a complementary input to investigate homoglyph domains.

In order to answer RQ1, the objective here is to extract IDNs in OpenINTEL measurements and acquire the growth rate of them. In the next step, extracted IDNs will be investigated using Unicode homoglyph tables to measure the hit rate per confusable character to see which characters are prone to be used in phishing attacks. As a result of this step, relevance of different homoglyph tables used in the literature could be measured.

3.2 Phishing Detection Method and validation

This subsection aims to answer RQ2. In order to detect phishing domains, the list of domain names consisting of one or more Unicode characters will be extracted by looking up the OpenINTEL database. Then Using Unicode homoglyphs lists, the corresponding ASCII domain name will be created. Two domains will be compared for some parameters such as the Geo-location of IP addresses hosting them. Based on these information, a decision will be made on whether the IDN is meant for phishing.

In order to evaluate detected phishing domains, benign and malicious ground truth will be used. As a malicious ground truth, blacklists such as PhishTank [22] and OpenPhish [23] would be used. As a ground truth for benign domains, Alexa top sites list is frequently used in the literature.

3.3 Time Advantage

Results of a study on domains used in spam emails by Hao et al. [6] indicate that roughly half of the domains used by spammers are registered less than 24 hours before actively being used. This implies that there is a limited time to perform reactions against such attacks before they occur. To find an answer for RQ3 and qualify the achievable time advantage by detecting IDN phishing before they appear on blacklists, historical OpenINTEL data will be used to detect IDN phishing domains and study the time gap between its detection and appearance in Blacklists.

4 PLANNING

In this section, the planning for this research is shortly discussed. The study has been split into two main parts of literature research and thesis analysis, as a guideline which can be seen in the Table 1. The literature research part consists solely of studying similar researches that focuses on discovering pros and cons of methods used in these works and proposing approaches which can answer the research questions. Relevant information learned from this study is used to develop the methodology of this research as well as being integrated in a survey that will form the first part of the thesis.

Following the literature research, in the thesis analysis part, the methodology of this research will be deployed which consists of developing the scripts to acquire the answers to research questions, with time allotted at the end to integrate the results into the thesis.

REFERENCES

[1] The Unicode Consortium. https://www.uincode.org/.
[2] Martin J Dürst. Urls and internationalization. World Wide Web Consortium, 1996.
[3] P Falstrom, Paul Hoffman, and A Costello. Internationalizing domain names in applications (idna)(RFC 3490). *Cisco, IMC & VPNC, University of California, USA*, 2003.
[4] Eric Johanson. IDN hacking disclosure. ShmooCon, 2005.
[5] Egon Kidmose, Erwin Lansing, Søren Brandbyge, and Jens Myrup Pedersen. Detection of malicious and abusive domain names. In *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, pages 49–56. IEEE, 2018.
[6] Shuang Hao, Nick Feamster, and Ramakant Pandrangi. Monitoring the initial dns behavior of malicious domains. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 269–278. ACM, 2011.
[7] Alexa top sites. http://aws.amazon.com/alexa-top-sites/.
[8] OpenINTEL. https://www.openintel.nl/, .
[9] Roland van Rijswijk-Deij, Mattijs Jonker, Anna Sperotto, and Aiko Pras. The internet of names: A dns big dataset. *ACM SIGCOMM Computer Communication Review*, 45(4):91–92, 2015.
[10] Anthony Y Fu, Xiaotie Deng, Liu Wenyin, and Greg Little. The methodology and an application to fight against unicode attacks. In *Proceedings of the second*

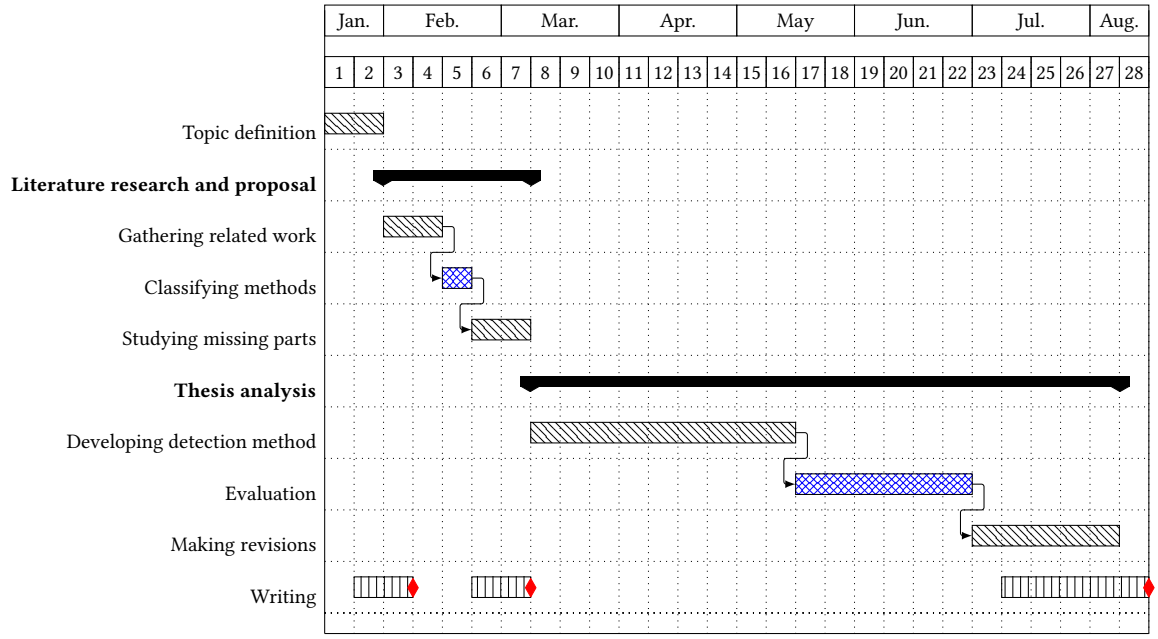


Table 1: Time plan

symposium on Usable privacy and security, pages 91–101. ACM, 2006.

[11] Narges Roshanbin and James Miller. Finding homoglyphs-a step towards detecting unicode-based visual spoofing attacks. In *International Conference on Web Information Systems Engineering*, pages 1–14. Springer, 2011.

[12] Viktor Krammer. Phishing defense against idn address spoofing attacks. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services*, page 32. ACM, 2006.

[13] Johnny Al Helou and Scott Tilley. Multilingual web sites: Internationalized domain name homograph attacks. In *2010 12th IEEE International Symposium on Web Systems Evolution (WSE)*, pages 89–92. IEEE, 2010.

[14] Faisal Alvi, Mark Stevenson, and Paul Clough. Plagiarism detection in texts obfuscated with homoglyphs. In *European Conference on Information Retrieval*, pages 669–675. Springer, 2017.

[15] Unicode confusables list. https://unicode.org/Public/security/.

[16] Anthony Y Fu, Xiaotie Deng, and Liu Wenyin. Regap: A tool for unicode-based web identity fraud detection. *Journal of Digital Forensic Practice*, 1(2):83–97, 2006.

[17] Hossein Shirazi, Bruhadeshwar Bezawada, and Indrakshi Ray. Kn0w thy doma1n name: Unbiased phishing detection using domain name based features. In *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*, pages 69–75. ACM, 2018.

[18] Tobias Holgers, David E Watson, and Steven D Gribble. Cutting through the confusion: A measurement study of homograph attacks. In *USENIX Annual Technical Conference, General Track*, pages 261–266, 2006.

[19] Bite Qiu, Ning Fang, and Liu Wenyin. Detect visual spoofing in unicode-based text. In *2010 20th International Conference on Pattern Recognition*, pages 1949–1952. IEEE, 2010.

[20] Yahia Elsayed and Ahmed Shosha. Large scale detection of idn domain name masquerading. In *2018 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–11. IEEE, 2018.

[21] EURid 2018 report. https://idnworldreport.eu/2018-2/.

[22] OpenDNS PhishTank. https://www.phishtank.com/.

[23] OpenPhish. https://www.openphish.com/, .

Appendix C

Samples of Existing Homoglyph Tables

This appendix represents sample rows of the "*Unicode Confusables*" and "*UC-SimList0.8*" Unicode homoglyph tables. Each row in the "*Unicode Confusables*" table includes the codepoint of the Unicode character, the codepoint of the ASCII homoglyph character (or string of characters), the visual glyph for the Unicode character, the visual glyph for the ASCII character(s), the name of the Unicode character and the name of the ASCII character(s), respectively. A row in the "*UC-SimList0.8*" starts with a character for which homoglyphs are given in the same line along with their degree of similarity (a number between 0.8 and 1) in a descending order of similarity.

Sample rows of the "Unicode Confusables" table:

```
237A ;  0061 ;  MA      #* ( α → a ) APL FUNCTIONAL SYMBOL ALPHA → LATIN SMALL LETTER A     # →α→
FF41 ;  0061 ;  MA      # ( ａ → a ) FULLWIDTH LATIN SMALL LETTER A → LATIN SMALL LETTER A   # →a→
1D41A ; 0061 ;  MA      # ( 𝐚 → a ) MATHEMATICAL BOLD SMALL A → LATIN SMALL LETTER A #
1D44E ; 0061 ;  MA      # ( 𝑎 → a ) MATHEMATICAL ITALIC SMALL A → LATIN SMALL LETTER A       #
1D482 ; 0061 ;  MA      # ( 𝒂 → a ) MATHEMATICAL BOLD ITALIC SMALL A → LATIN SMALL LETTER A  #
1D4B6 ; 0061 ;  MA      # ( 𝒶 → a ) MATHEMATICAL SCRIPT SMALL A → LATIN SMALL LETTER A       #
1D4EA ; 0061 ;  MA      # ( 𝓪 → a ) MATHEMATICAL BOLD SCRIPT SMALL A → LATIN SMALL LETTER A  #
1D51E ; 0061 ;  MA      # ( 𝔞 → a ) MATHEMATICAL FRAKTUR SMALL A → LATIN SMALL LETTER A      #
1D552 ; 0061 ;  MA      # ( 𝕒 → a ) MATHEMATICAL DOUBLE-STRUCK SMALL A → LATIN SMALL LETTER A        #
1D586 ; 0061 ;  MA      # ( 𝖆 → a ) MATHEMATICAL BOLD FRAKTUR SMALL A → LATIN SMALL LETTER A #
1D5BA ; 0061 ;  MA      # ( 𝖺 → a ) MATHEMATICAL SANS-SERIF SMALL A → LATIN SMALL LETTER A   #
1D5EE ; 0061 ;  MA      # ( 𝗮 → a ) MATHEMATICAL SANS-SERIF BOLD SMALL A → LATIN SMALL LETTER A      #
1D622 ; 0061 ;  MA      # ( 𝘢 → a ) MATHEMATICAL SANS-SERIF ITALIC SMALL A → LATIN SMALL LETTER A    #
1D656 ; 0061 ;  MA      # ( 𝙖 → a ) MATHEMATICAL SANS-SERIF BOLD ITALIC SMALL A → LATIN SMALL LETTER A      #
1D68A ; 0061 ;  MA      # ( 𝚊 → a ) MATHEMATICAL MONOSPACE SMALL A → LATIN SMALL LETTER A    #
0251 ;  0061 ;  MA      # ( ɑ → a ) LATIN SMALL LETTER ALPHA → LATIN SMALL LETTER A  #
03B1 ;  0061 ;  MA      # ( α → a ) GREEK SMALL LETTER ALPHA → LATIN SMALL LETTER A  #
1D6C2 ; 0061 ;  MA      # ( 𝛂 → a ) MATHEMATICAL BOLD SMALL ALPHA → LATIN SMALL LETTER A     # →α→
1D6FC ; 0061 ;  MA      # ( 𝛼 → a ) MATHEMATICAL ITALIC SMALL ALPHA → LATIN SMALL LETTER A   # →α→
1D736 ; 0061 ;  MA      # ( 𝜶 → a ) MATHEMATICAL BOLD ITALIC SMALL ALPHA → LATIN SMALL LETTER A      # →α→
1D770 ; 0061 ;  MA      # ( 𝝰 → a ) MATHEMATICAL SANS-SERIF BOLD SMALL ALPHA → LATIN SMALL LETTER A  # →α→
1D7AA ; 0061 ;  MA      # ( 𝞪 → a ) MATHEMATICAL SANS-SERIF BOLD ITALIC SMALL ALPHA → LATIN SMALL LETTER A  #
0430 ;  0061 ;  MA      # ( а → a ) CYRILLIC SMALL LETTER A → LATIN SMALL LETTER A   #
FF21 ;  0041 ;  MA      # ( Ａ → A ) FULLWIDTH LATIN CAPITAL LETTER A → LATIN CAPITAL LETTER A       # →A→
1D400 ; 0041 ;  MA      # ( 𝐀 → A ) MATHEMATICAL BOLD CAPITAL A → LATIN CAPITAL LETTER A     #
1D434 ; 0041 ;  MA      # ( 𝐴 → A ) MATHEMATICAL ITALIC CAPITAL A → LATIN CAPITAL LETTER A   #
1D468 ; 0041 ;  MA      # ( 𝑨 → A ) MATHEMATICAL BOLD ITALIC CAPITAL A → LATIN CAPITAL LETTER A      #
1D49C ; 0041 ;  MA      # ( 𝒜 → A ) MATHEMATICAL SCRIPT CAPITAL A → LATIN CAPITAL LETTER A  #
1D4D0 ; 0041 ;  MA      # ( 𝓐 → A ) MATHEMATICAL BOLD SCRIPT CAPITAL A → LATIN CAPITAL LETTER A     #
1D504 ; 0041 ;  MA      # ( 𝔄 → A ) MATHEMATICAL FRAKTUR CAPITAL A → LATIN CAPITAL LETTER A  #
1D538 ; 0041 ;  MA      # ( 𝔸 → A ) MATHEMATICAL DOUBLE-STRUCK CAPITAL A → LATIN CAPITAL LETTER A    #
1D56C ; 0041 ;  MA      # ( 𝕬 → A ) MATHEMATICAL BOLD FRAKTUR CAPITAL A → LATIN CAPITAL LETTER A     #
1D5A0 ; 0041 ;  MA      # ( 𝖠 → A ) MATHEMATICAL SANS-SERIF CAPITAL A → LATIN CAPITAL LETTER A       #
1D5D4 ; 0041 ;  MA      # ( 𝗔 → A ) MATHEMATICAL SANS-SERIF BOLD CAPITAL A → LATIN CAPITAL LETTER A  #
```

Sample rows of the ”UC-SimList0.8” table:

```
0041    1:FF41:ａ       1:0061:a        1:0430:а        1:0391:Α        1:FF21:Ａ       1:0410:А        1:0041:A
        0.9496855:1EA0:Ạ        0.9413919:1EA1:ạ        0.9207317:1FBC:Ᾱ        0.9041916:1E00:Ḁ
        0.8892733:1E01:ạ        0.8653198:0105:ą        0.8505747:04E9:ө        0.8494208:0275:ɵ
        0.8328173:0386:Ά        0.8294573:0E25:ค        0.8210117:0259:ə        0.8210117:04D9:ә
        0.8210117:01DD:ǝ        0.8132296:0250:ɐ        0.8015564:01A8:ƨ
0042    1:0412:В        1:FF42:ｂ       1:0062:b        1:0392:Β        1:FF22:Ｂ       1:0042:B
        0.956044:1E04:Ḅ 0.948718:0185:ƅ 0.9446367:1E05:ḅ        0.9166667:0253:ɓ        0.9133858:1E06:Ḇ
        0.8921568:1E07:ḇ        0.8881579:00FE:þ        0.8662421:0183:Ƃ        0.8563218:0182:Ƃ
        0.8158845:10EE:ბ        0.8095238:1E2B:ḫ        0.8095238:1E96:ẖ        0.8095238:0068:h
        0.8095238:1E25:ḥ        0.8095238:FF48:ｈ       0.8095238:04BB:h        0.8074713:0411:Б
        0.8047138:1E55:ṕ
0043    1:0441:с        1:FF43:ｃ       1:217D:ⅽ        1:0063:c        1:03F2:ϲ        1:0043:C        1:FF23:Ｃ
        1:216D:Ⅽ        1:0421:С        0.8957346:0454:є        0.8954248:0404:Є        0.8926175:0480:Ҁ
        0.8625:04AA:Ҫ   0.8260869:04AB:ҫ        0.8256228:10BA:Ⴋ        0.8204334:00C7:Ç        0.8146552:00E7:ç
        0.8105263:0254:ɔ        0.8103448:0255:ɕ        0.8006231:041E:О        0.8006231:FF2F:Ｏ
        0.8006231:039F:Ο        0.8006231:004F:O
0044    1:0064:d        1:FF44:ｄ       1:217E:ⅾ        1:0044:D        1:FF24:Ｄ       1:216E:Ⅾ
        0.9518072:1E0C:Ḍ        0.9442509:1E0D:ḍ        0.922807:0257:ɗ 0.9186441:010F:ď        0.909396:0111:đ
        0.9054441:1E0E:Ḏ        0.8914474:1E0F:ḏ        0.8901408:1E12:Ḓ        0.8753462:1E10:Ḑ
        0.8741935:1E13:ḓ        0.8630573:018C:ƌ        0.8566879:1E11:ḑ        0.8522013:0256:ɖ
        0.8424909:03AC:ά        0.8424909:1F70:ὰ        0.8424909:1F71:ά        0.8413284:10EB:ძ
        0.827044:02A0:ɋ 0.8214286:0151:ő        0.8062284:1FB0:ᾰ        0.8:020D:ȏ
0045    1:0065:е        1:0435:е        1:FF45:ｅ       1:0395:Ε        1:0045:E        1:0415:Е        1:FF25:Ｅ
        0.942029:1EB8:Ẹ 0.9384615:1EB9:ẹ        0.9098361:0258:ɘ        0.8841699:0275:ɵ        0.8812261:04E9:ө
        0.8695652:1E18:Ḙ        0.8666667:1E1A:Ḛ        0.8647541:FF4F:ｏ       0.8647541:03BF:ο
        0.8647541:006F:o        0.8647541:043E:о        0.8621908:1E19:ḙ        0.8591549:1E1B:ḛ
        0.8565574:025E:ɞ        0.852459:0118:Ę 0.8508065:1ECD:ọ        0.8467153:0473:ѳ        0.8413793:0119:ę
        0.8278689:0259:ə        0.8278689:0250:ɐ        0.8278689:04D9:ә        0.8278689:01DD:ǝ
        0.8237705:0251:ɑ        0.8039867:0182:Ƃ
0046    1:FF46:ｆ       1:0066:f        1:FF26:Ｆ       1:0046:F        0.9012346:01AD:ƭ        0.8451328:03DC:Ϝ
        0.814433:0413:Г 0.814433:0393:Γ 0.8026316:FF54:ｔ       0.8026316:0074:t
0047    1:0261:g        1:0067:g        1:FF47:ｇ       1:FF27:Ｇ       1:0047:G        0.9394813:01E5:ǧ
        0.9375:01E4:Ǧ   0.9350283:0122:Ģ        0.8292683:0039:9        0.8292683:FF19:９       0.8217523:FF2F:Ｏ
```

Appendix D

Added and Revised Homoglyphs

This appendix represents details of the proposed homoglyph table. Only differences of this table compared to the "*UC-SimList0.8*" in terms of added and replaced characters are given. Added characters were selected based on a manual look up of characters which appear frequently on domain names. In Table C.1, when there is no value in the "*Existing homoglyph (codepoint)*" column, it means that the character didn't have any ASCII homoglyph in the "*UC-SimList0.8*" table. Besides, the existing ASCII homoglyphs in this column are replaced with ones in "*Proposed homoglyphs (codepoint)*" column due to higher similarity.

Table C.1: Revisions made to propose new homoglyph table

	Unicode character description	Unicode character (codepoint)	Existing homoglyph (codepoint)	Proposed homoglyph (codepoint)
1	latin small letter dotless i	ı (U+0131)	! (U+0021)	i (U+0069)
2	latin small letter n with tilde	ñ (U+00F1)	h (U+0068)	n (U+006E)
3	latin small letter i with dot below	ị (U+1ECB)	I (U+0049)	i (U+0069)
4	latin small letter l with acute	Í (U+013A)	| (U+007C)	l (U+006C)
5	latin small letter l with cedilla	Ļ (U+013C)	| (U+007C)	l (U+006C)
6	latin small letter turned e	ə (U+01DD)	o (U+006F)	e (U+0065)
7	latin small letter schwa	ə (U+0259)	o (U+006F)	e (U+0065)
8	cyrillic small letter en	н (U+043D)	u (U+0075)	H (U+0048)
9	latin small letter l with dot below	ḷ (U+1E37)	| (U+007C)	l (U+006C)
10	latin small letter i with tilde below	ḭ (U+1E2D)	j (U+006A)	i (U+0069)
11	latin small letter n with dot above	ṅ (U+1E45)	h (U+0068)	n (U+006E)
12	latin small letter p with acute	ṕ (U+1E55)	b (U+0062)	p (U+0070)
13	latin small letter l with retroflex hook	ɭ (U+026D)	[(U+005B)	l (U+006C)
14	greek small letter iota	ι (U+03B9)	! (U+0021)	i (U+0069)
15	broken bar	¦ (U+00A6)	| (U+007C)	l (U+006C)
16	latin capital letter i with acute	Í (U+00CD)	| (U+007C)	l (U+006C)
17	latin capital letter i with tilde	Ĩ (U+0128)	] (U+005D)	l (U+006C)
18	latin capital letter i with ogonek	Į (U+012E)	[(U+005B)	l (U+006C)
19	latin capital letter i with dot above	İ (U+0130)	| (U+007C)	l (U+006C)
20	latin capital letter l with acute	Ĺ (U+0139)	[(U+005B)	L (U+004C)
21	latin small letter l with middle dot	ŀ (U+0140)	I (U+0049)	l (U+006C)
22	latin small letter o with double acute	ő (U+0151)	d (U+0064)	o (U+006F)

23	latin small letter long s	ſ (U+017F)	l (U+006C)	f (U+0066)
24	latin small letter j with caron	ǰ (U+01F0)	] (U+005D)	j (U+006A)
25	latin small letter o with double grave	ȍ (U+020D)	d (U+0064)	o (U+006F)
26	latin letter small capital g	ɢ (U+0262)	o (U+006F)	G (U+0047)
27	latin small letter upsilon	ʊ (U+028A)	o (U+006F)	u (U+0075)
28	latin letter small capital h	ʜ (U+029C)	u (U+0075)	H (U+0048)
29	greek small letter alpha with tonos	ά (U+03AC)	d (U+0064)	a (U+0061)
30	greek small letter alpha	α (U+03B1)	q (U+0071)	a (U+0061)
31	greek rho symbol	ϱ (U+03F1)	g (U+0067)	p (U+0070)
32	cyrillic small letter koppa	ҁ (U+0481)	o (U+006F)	c (U+0063)
33	cyrillic small letter schwa	ə (U+04D9)	o (U+006F)	e (U+0065)
34	cyrillic capital letter barred o	Ө (U+04E8)	Q (U+0051)	O (U+004F)
35	georgian capital letter can	Ⴚ (U+10BA)	C (U+0043)	G (U+0047)
36	latin capital letter i with dot below	Ị (U+1ECA)	| (U+007C)	l (U+006C)
37	greek small letter alpha with varia	ὰ (U+1F70)	d (U+0064)	a (U+0061)
38	greek small letter alpha with oxia	ά (U+1F71)	d (U+0064)	a (U+0061)
39	greek small letter alpha with vrachy	ᾰ (U+1FB0)	d (U+0064)	a (U+0061)
40	greek small letter alpha with ypogegrammeni	ᾳ (U+1FB3)	q (U+0071)	a (U+0061)
41	proportional to	∝ (U+221D)	q (U+0071)	a (U+0061)
42	divides	∣ (U+2223)	| (U+007C)	l (U+006C)
43	latin small letter i with circumflex	î (U+00EE)	-	i (U+0069)
44	cyrillic small letter palochka	Ӏ (U+04CF)	-	l (U+006C)
45	latin small letter b with dot above	ḃ (U+1E03)	-	b (U+0062)
46	latin small letter z with dot above	ż (U+017C)	-	z (U+007A)
47	latin small letter a with hook above	ả (U+1EA3)	-	a (U+0061)
48	latin small letter o with diaeresis	ö (U+00F6)	-	o (U+006F)

49	latin small letter g with breve	ğ (U+011F)	-	g (U+0067)
50	latin small letter a with ring above	å (U+00E5)	-	a (U+0061)
51	latin small letter e with diaeresis	ë (U+00EB)	-	e (U+0065)
52	latin small letter e with breve	ĕ (U+0115)	-	e (U+0065)
53	latin small letter o with tilde	õ (U+00F5)	-	o (U+006F)
54	latin small letter o with stroke	ø (U+00F8)	-	o (U+006F)
55	latin small letter y with diaeresis	ÿ (U+00FF)	-	y (U+0079)
56	latin small letter c with caron	č (U+010D)	-	c (U+0063)
57	latin small letter e with double grave	ȅ (U+0205)	-	e (U+0065)
58	latin small letter r with double grave	ȑ (U+0211)	-	r (U+0072)
59	latin small letter o with dot above	ȯ (U+022F)	-	o (U+006F)
60	latin small letter t with curl	ȶ (U+0236)	-	t (U+0074)
61	latin small letter g with circumflex	ĝ (U+011D)	-	g (U+0067)
62	latin small letter g with dot above	ġ (U+0121)	-	g (U+0067)
63	latin small letter i with diaeresis	ï (U+00EF)	-	i (U+0069)
64	latin small letter i with tilde	ĩ (U+0129)	-	i (U+0069)
65	latin small letter i with ogonek	į (U+012F)	-	i (U+0069)
66	latin small letter s with caron	š (U+0161)	-	s (U+0073)
67	latin small letter u with horn	ư (U+01B0)	-	u (U+0075)
68	latin small letter o with horn	ơ (U+01A1)	-	o (U+006F)
69	latin small letter e with cedilla	ȩ (U+0229)	-	e (U+0065)
70	latin small letter n with left hook	ɲ (U+0272)	-	n (U+006E)
71	latin small letter e with retroflex hook	ᶒ (U+1D92)	-	e (U+0065)
72	latin small letter d with dot above	ḋ (U+1E0B)	-	d (U+0064)
73	latin small letter e with macron and grave	ḕ (U+1E15)	-	e (U+0065)
74	latin small letter f with dot above	ḟ (U+1E1F)	-	f (U+0066)

75	latin small letter g with macron	ḡ (U+1E21)	-	g (U+0067)
76	latin small letter h with dot above	ḣ (U+1E23)	-	h (U+0068)
77	latin small letter h with cedilla	ḫ (U+1E29)	-	h (U+0068)
78	latin small letter i with diaeresis and acute	ḯ (U+1E2F)	-	i (U+0069)
79	latin small letter k with acute	ḱ (U+1E31)	-	k (U+006B)
80	latin small letter l with line below	ḻ (U+1E3B)	-	l (U+006C)
81	latin small letter l with circumflex below	ḽ (U+1E3D)	-	l (U+006C)
82	latin small letter m with acute	ḿ (U+1E3F)	-	m (U+006D)
83	latin small letter m with dot above	ṁ (U+1E41)	-	m (U+006D)
84	latin small letter o with tilde and acute	ṍ (U+1E4D)	-	o (U+006F)
85	latin small letter o with tilde and diaeresis	ṏ (U+1E4F)	-	o (U+006F)
86	latin small letter p with dot above	ṗ (U+1E57)	-	p (U+0070)
87	latin small letter r with dot above	ṙ (U+1E59)	-	r (U+0072)
88	latin small letter r with line below	ṟ (U+1E5F)	-	r (U+0072)
89	latin small letter s with dot above	ṡ (U+1E61)	-	s (U+0073)
90	latin small letter s with acute and dot above	ṥ (U+1E65)	-	s (U+0073)
91	latin small letter s with caron and dot above	ṧ (U+1E67)	-	s (U+0073)
92	latin small letter s with dot below and dot above	ṩ (U+1E69)	-	s (U+0073)
93	latin small letter t with dot above	ṫ (U+1E6B)	-	t (U+0074)
94	latin small letter t with circumflex below	ṭ (U+1E71)	-	t (U+0074)
95	latin small letter v with tilde	ṽ (U+1E7D)	-	v (U+0076)
96	latin small letter w with grave	ẁ (U+1E81)	-	w (U+0077)
97	latin small letter w with acute	ẃ (U+1E83)	-	w (U+0077)
98	latin small letter w with dot above	ẇ (U+1E87)	-	w (U+0077)
99	latin small letter x with dot above	ẋ (U+1E8B)	-	x (U+0078)
100	latin small letter x with diaeresis	ẍ (U+1E8D)	-	x (U+0078)

101	latin small letter y with dot above	ẏ (U+1E8F)	-	y (U+0079)
102	latin small letter z with circumflex	ẑ (U+1E91)	-	z (U+007A)
103	latin small letter a with breve and acute	ắ (U+1EAF)	-	a (U+0061)
104	latin small letter e with hook above	ẻ (U+1EBB)	-	e (U+0065)
105	latin small letter e with tilde	ẽ (U+1EBD)	-	e (U+0065)
106	latin small letter e with circumflex and acute	ế (U+1EBF)	-	e (U+0065)
107	latin small letter e with circumflex and tilde	ễ (U+1EC5)	-	e (U+0065)
108	latin small letter o with horn and hook above	ở (U+1EDF)	-	o (U+006F)
109	latin small letter a with diaeresis	ä (U+00E4)	-	a (U+0061)
110	latin small letter i with hook above	ỉ (U+1EC9)	-	i (U+0069)
111	latin small letter u with acute	ú (U+00FA)	-	u (U+0075)
112	latin small letter a with circumflex	â (U+00E2)	-	a (U+0061)
113	latin small letter a with tilde	ã (U+00E3)	-	a (U+0061)
114	latin small letter i with inverted breve	î (U+020B)	-	i (U+0069)
115	latin small letter e with dot above	ė (U+0117)	-	e (U+0065)
116	latin small letter u with horn and dot below	ự (U+1EF1)	-	u (U+0075)
117	latin small letter u with circumflex	û (U+00FB)	-	u (U+0075)
118	latin small letter s with acute	ś (U+015B)	-	s (U+0073)
119	latin small letter c with acute	ć (U+0107)	-	c (U+0063)
120	latin small letter o with circumflex	ô (U+00F4)	-	o (U+006F)
121	latin small letter e with notch	ⱸ (U+2C78)	-	e (U+0065)
122	vai syllable zoo	ꖉ (U+A589)	-	8 (U+0038)
123	combining dot above	◌̇ (U+0307)	-	Removing the character