Homograph Attack Prevention

Rayan Bouhal
Virginia Tech
Blacksburg, Virginia
rayanb@vt.edu

Andrew Merdes
Virginia Tech
Blacksburg, Virginia
merdea03@vt.edu

Angelie Nguyen
Virginia Tech
Blacksburg, Virginia
angelienguyen@vt.edu

ACM Reference Format:
Rayan Bouhal, Andrew Merdes, and Angelie Nguyen. 2024. Homograph Attack Prevention. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Homograph attacks often involve subtle modifications, such as altering individual characters in the domain name. By creating these deceptive domain names, scammers exploit unknowing users via email and various other communication methods. Homographs have posed a significant challenge, as evidenced by the registration of over 1,516 deceptive domain names in 2018, mimicking the top 1,000 Alexa-ranked brands [5]. We aim to tackle the issue of homograph attacks on internationalized domain names through a Machine Learning approach, specifically by analyzing two decision tree algorithms and using the Python library decision tree algorithm as a benchmark for comparison. This will allow us to evaluate the effectiveness of different decision tree algorithms in detecting homograph attacks, and optimize our approach for the highest possible accuracy. Our ultimate goal is to enhance the security of internationalized domain names and protect users from potential threats.

2 RELATED WORK

When researching the homograph attack detection problem, our team found a few studies to be helpful in determining our project. The first paper is about a homograph detection study that uses optical character recognition (OCR) to determine if a domain is malicious [4]. This article explains the issue of having malicious domains with characters that are visually identical to the correct character. This is a basis for our project on detecting malicious domains. Another study we examined did a comparison of binary classification models. This study cited the decision trees as having the highest classification accuracy, as well as being a very flexible model [1]. When deciding on what decision trees to study, we used a paper giving descriptions on three popular decision tree algorithms [3]. We used another paper to have an understanding of entropy, information gain, and their relationship with decision trees [2].

Conference'17, July 2017, Washington, DC, USA
© 2024 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3 METHOD

Our project methods were based on two decision tree algorithms: the Iterative Dichotomiser 3 (ID3) algorithm and the Classification and Regression Tree (CART) algorithm. Our final project involves applying binary classification to two separate models using these algorithms. Additionally, we focused on creating data categories based on features that we observed from the valid and invalid domains in our dataset. The next sections describe how each part of the project was implemented, as well as the methodology behind the design choices. To effectively demonstrate and utilize these methods, we developed an interactive website. This platform serves as both a practical tool for real-time domain validity checks and a demonstration of our system's capabilities. It allows users to directly interact with our machine learning models by submitting domains for validation. Users can then view the classification results and receive explanations based on the model's decision-making process. This real-time application not only showcases our project's practical impact but also enhances our dataset with fresh examples that help to continually refine our models.

3.1 Data Collection and Parsing

We sourced our data from two key resources: the "List of Top 1 Million Domains" provided by Mozilla and the "List of Malicious Domains" supplied by Cert Poland. This data was instrumental in establishing the foundation for our supervised learning approach. In this context, 'X' represents the relevant features extracted from each domain, while 'y' denotes the validity of each domain name, indicating whether it is a legitimate domain or a homograph. Additionally, our website contributes to ongoing data collection efforts by logging user queries. When a user submits a domain through the website, it not only returns a prediction but also captures this input for future analysis. This mechanism enriches our dataset with real-world examples, continuously enhancing the model's robustness and accuracy.

3.2 Feature Extraction

To improve our algorithm accuracy, we wanted to extract a couple of features from the domain names. For feature extraction, we analyze each domain name to identify characteristics that are indicative of legitimate or malicious intent. Our features included:

- **Domain Length:** The total number of characters in the domain. Longer or unusually short domains can be indicative of malicious intent.
- **Character Count:** The count of alphabetical characters in the domain name before the top-level domain (TLD). Non-standard character distributions can suggest a homograph attack.

- **Digit Count:** The number of digits in the domain name before the TLD. Many legitimate domains do not include digits, whereas malicious domains might use them to mimic other domains.
- **Non-ASCII Character Count:** Counts the characters that are not part of the standard ASCII set, which are often used in Internationalized Domain Names (IDNs) that can be exploited in homograph attacks.
- **Domain Hash:** A hash value of the domain name before the TLD. This feature helps in reducing the dimensionality of the domain string for the model and captures the uniqueness of each domain.
- **TLD Hash:** A hash value of the top-level domain. Since certain malicious campaigns might prefer specific TLDs, hashing the TLD allows for a uniform representation in the model's feature set.

For the hashing process, we use the SHA-256 algorithm, truncating the result to the first 16 hexadecimal characters to maintain a balance between collision resistance and computational efficiency. These features are then passed to the classifier algorithms within our system, enabling effective learning and prediction of domain validity. This transformation allowed us to convert these categorical features into a format that could be effectively processed by our algorithms. These features are selected because they are likely to reveal patterns that differentiate genuine domains from their homographed counterparts. Once the features are extracted, they will be passed to all the algorithms.

3.3 Implementing the ID3 Algorithm

Our team implemented the ID3 algorithm with our own python code. The ID3 algorithm has three main steps. The first is to compare the entropy of the entire dataset to the entropy of each category. The entropy is calculated using the following equation:

$$Entropy = \sum_i log_2(p_i)$$

Note that p_i is the probability of a given classification. Then we calculate the information gain of each category split, and choose to split the category with the highest information gain. The information gain is calculated using the following equation:

$$I_G = E(D) - E(c_i)$$

Note that $E(D)$ represents the entropy of the data set before the category split, and $E(c_i)$ represents the entropy after the category split. The last step is to recursively split the data by category until all of the data points are classified. Because of the nature of the ID3 algorithm, the splits on each run of the recursive algorithm iterations can only be a binary split. Because some of the data features were not binary, we had to preprocess the data to make extra "features" that represented a range of the data values. For example, in the domain length category the postprocessed category would be a set of three new binary categories that represented the three ranges of 0-6 characters, 7-18 characters and 18+ characters. For each input, only one of these new categories would have a value of 1 while the rest would have a value of 0.

3.4 Implementing the CART Algorithm

Our team implemented the CART algorithm by using the built in algorithm in the sci-kit library. The CART algorithm has four main steps. The first is to calculate the Gini impurity. This is a value that represents the probability of miss-classifying a random input. The Gini impurity is calculated using the following equation:

$$Gini(t) = 1 - \sum_{i=1}^{2} P(i|t)^2$$

Note that $P(i|t)$ represents the ratio of the class at the "ith" node. The next step is to find the best split points that reduce the Gini impurity. Lastly, this process is repeated until it reaches a maximum tree depth that was previously specified.

3.5 Website

The project's web platform, accessible at **https://homographdetector.pro/**, is an integral part of our approach to detecting homograph attacks. Designed using the Flask framework, the website provides a user-friendly interface where individuals can test the validity of domain names in real-time. This section elaborates on the key features and functionalities of the website, which enhance the accessibility and practical application of our research. The website comprises several pages, each serving a distinct function:

Home Page: The landing page introduces the core feature of the website. This interactive form allows users to input a domain name, which is then processed by our back-end to determine its validity. The results, along with a detailed breakdown of the domain's features and the model's decision, are displayed.

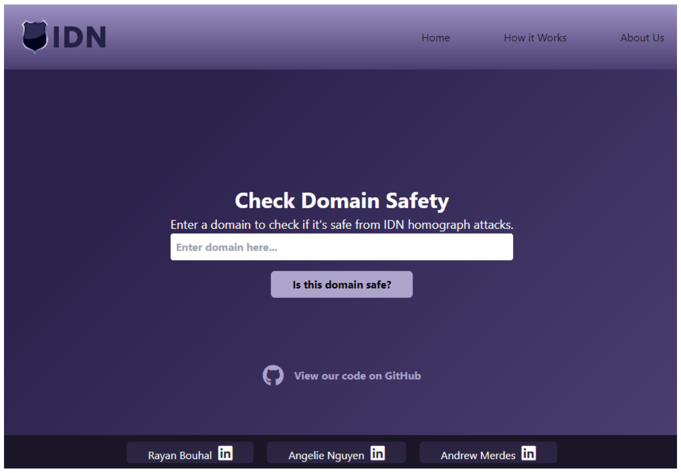
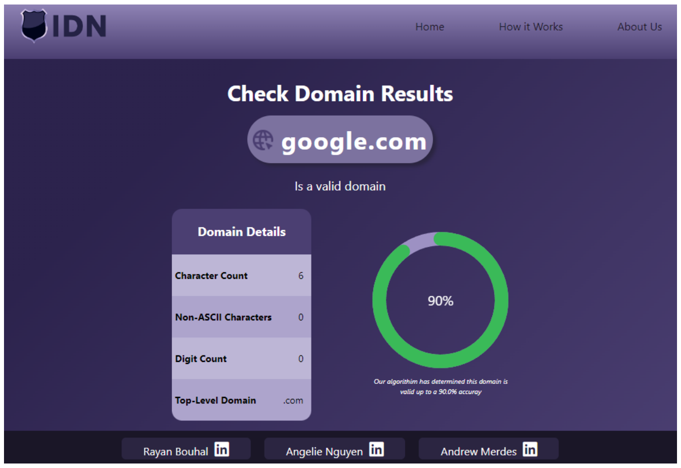


Figure 1: Home Page of the Website

Valid Result: This screenshot shows the result page for a valid domain check. The user has entered "google.com," which the system has confirmed as a valid domain with a 90 percent accuracy rating. The page details various domain attributes such as character count, the presence of non-ASCII characters, and digit count, complemented by the top-level domain. A prominent feature is the accuracy meter, which visually underscores the confidence level of the validation.



Check Domain Results

google.com

Is a valid domain

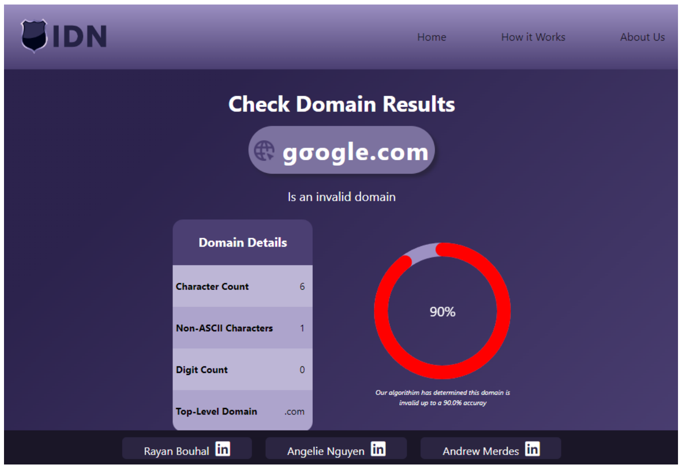
Domain Details	
Character Count	6
Non-ASCII Characters	0
Digit Count	0
Top-Level Domain	.com

90%

Our algorithim has determined this domain is valid up to a 90.0% accuray

Figure 2: User enters domain: google.com

Invalid Result: This screenshot illustrates the result page for an invalid domain check. The user has entered "google.com," with the first "o" replaced by a σ, which the system has flagged as an invalid domain with a 90 percent accuracy rating. This alteration, notably due to the presence of non-ASCII characters, suggests a potential homograph attack. The page provides details on domain attributes such as character count, non-ASCII character presence, and top-level domain. Similarly, the accuracy meter is prominently displayed, visually reinforcing the validation's high confidence level and highlighting the domain's potential risks.



Check Domain Results

gσogle.com

Is an invalid domain

Domain Details	
Character Count	6
Non-ASCII Characters	1
Digit Count	0
Top-Level Domain	.com

90%

Our algorithim has determined this domain is invalid up to a 90.0% accuray

Figure 3: User enters domain: gσogle.com

When a domain is submitted through the "Home Page", the Flask application captures the input and processes it through our machine learning models. The models, implemented as described in the earlier sections, utilize features extracted from the domain to classify it as either valid or potentially malicious. The results, which include the classification outcome and the confidence score, are then displayed to the user. Each query not only serves the immediate need of the user but also logs data that contributes to the continuous improvement of our models. By capturing new domain examples through user interactions, we enrich our dataset, which is essential for adapting and refining our models to handle new and evolving types of homograph attacks effectively. Our website also features two informative pages: the "About Us" page and the "How It Works" page. The "About Us" page offers background information about our team, the goals of the project, and the methodologies we employ. Conversely, the "How It Works" page provides detailed explanations of the underlying technology, algorithms, and data processing steps, serving to educate users on how domain validation is conducted.

4 RESULTS AND EVALUATION

In our study, we compared the performance of two decision tree algorithms: ID3 and CART. For benchmarking purposes, we also used a Python library decision tree algorithm, which had an accuracy of around 90 percent. When observing the final accuracy of the ID3 and CART algorithms, we found that the CART algorithm had a better accuracy than the ID3 algorithm. The CART accuracy was roughly around 90 percent, matching the benchmark, while the ID3 algorithm had an accuracy of roughly 75 percent. We found that the reason for this discrepancy had to do with how each algorithm split the data. Because the ID3 algorithm used information gain, which is based on the features of the data, the decision tree was overfitting to the training data. The final model would only classify valid domains that had the same number value feature as one of the valid features in the training data set. It had trouble generalizing to a range of values that were close to the input data. This is likely due to how we preprocessed the data into a binary format for the ID3 algorithm. In contrast, the CART algorithm did not have this overfitting problem because it can handle the range of data values with the Gini impurity splitting. When observing the testing on cases with visually identical characters, the CART algorithm accurately identified the difference between the safe domains and the malicious domains, which is an important improvement in modern homograph attack detection efforts.

5 CONCLUSION AND FUTURE WORK

In conclusion, our project has successfully demonstrated the potential of using machine learning algorithms to identify and prevent homograph attacks through domain validation. By employing the ID3 and CART decision tree algorithms, we have established a robust system that enhances the security of domain names. The CART algorithm, in particular, has shown promising results in terms of accuracy and its ability to handle diverse data inputs, including those with a variety of Unicode characters. Moving forward, we plan to enrich our dataset by including more examples of valid and invalid domains that predominantly contain Unicode characters. This addition will better depict real-world scenarios and help in refining our models' accuracy and generalizability. To further improve our system's prediction capabilities, we will explore advanced feature engineering techniques and integrate additional classification models. New features such as the frequency of character repetition, the ratio of numeric to alphabetic characters, and the use of top-level domain-specific frequencies could provide deeper insights into domain validity. Currently, our project utilizes two separate CSV files, "valid-domains.csv" and "invalid-domains.csv", for reading and writing domain data. Transitioning from this setup to a dynamic database will enhance data management efficiency and scalability, facilitating more effective data retrieval and ongoing system training. Moreover, the incorporation of a web scraping tool specifically designed for the Chrome browser will allow us to automate the collection of domain data from various online sources, thereby continually enriching our dataset and keeping our system updated with the latest trends in domain usage and abuse. These enhancements will contribute significantly to the ongoing improvement of our project, making it a highly accurate tool for combating cybersecurity threats associated with IDN homograph attacks and thus ensuring a safer digital environment for users worldwide.

REFERENCES

[1] Vedant Bahel, Sofia Pillai, and Manit Malhotra. 2020. A Comparative Study on Various Binary Classification Algorithms and their Improved Variant for Optimal Performance. (2020), 495–498. https://doi.org/10.1109/TENSYMP50017.2020.9230877

[2] Bahzad Jijo and Adnan Mohsin Abdulazeez. 2021. Classification Based on Decision Tree Algorithm for Machine Learning. *Journal of Applied Science and Technology Trends* 2 (01 2021), 20–28.

[3] Ibomoiye Domor Mienye, Yanxia Sun, and Zenghui Wang. 2019. Prediction performance of improved decision tree-based algorithms: a review. *Procedia Manufacturing* 35 (2019), 698–703. https://doi.org/10.1016/j.promfg.2019.06.011 The 2nd International Conference on Sustainable Materials Processing and Manufacturing, SMPM 2019, 8-10 March 2019, Sun City, South Africa.

[4] Yuta Sawabe, Daiki Chiba, Mitsuaki Akiyama, and Shigeki Goto. 2019. Detection Method of Homograph Internationalized Domain Names with OCR. *Journal of Information Processing* 27 (09 2019), 536–544. https://doi.org/10.2197/ipsjjip.27.536

[5] Tran Phuong Thao, Yukiko Sawaya, Hoang-Quoc Nguyen-Son, Akira Yamada, Ayumu Kubota, Tran Van Sang, and Rie Shigetomi Yamaguchi. 2020. Human Factors in Homograph Attack Recognition. In *Applied Cryptography and Network Security*, Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi (Eds.). Springer International Publishing, Cham, 408–435.