Journal of
Sensor and Actuator Networks

MDPI

Article

Homoglyph Attack Detection Model Using Machine Learning and Hash Function

Abdullah M. Almuhaideb [1,*], Nida Aslam [2], Almaha Alabdullatif [2], Sarah Altamimi [2], Shooq Alothman [2], Amnah Alhussain [2], Waad Aldosari [2], Shikah J. Alsunaidi [2] and Khalid A. Alissa [1]

[1] SAUDI ARAMCO Cybersecurity Chair, Department of Networks and Communications, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, P.O. Box 1982, Dammam 31441, Saudi Arabia
[2] Department of Computer Science, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, P.O. Box 1982, Dammam 31441, Saudi Arabia
* Correspondence: amalmuhaideb@iau.edu.sa

Abstract: Phishing is still a major security threat in cyberspace. In phishing, attackers steal critical information from victims by presenting a spoofing/fake site that appears to be a visual clone of a legitimate site. Several Unicode characters are visually identical to ASCII characters. This similarity in characters is generally known as homoglyphs. Malicious adversaries utilize homoglyphs in URLs and DNS domains to target organizations. To reduce the risks caused by phishing attacks, effective ways of detecting phishing websites are urgently required. This paper proposes a homoglyph attack detection model that combines a hash function and machine learning. There are two phases to the model approach. The machine was being trained during the development phase. The deployment phase involved deploying the model with a Java interface and testing the outcomes through actual user interaction. The results are more accurate when the URL is hashed, as any little changes to the URL can be recognized. The homoglyph detector can be developed as a stand-alone software that is used as the initial step in requesting a webpage as it enhances browser security and protects websites from phishing attempts. To verify the effectiveness, we compared the proposed model on several criteria to existing phishing detection methods. By using the hash function, the proposed security features increase the overall security of the homoglyph attack detection in terms of accuracy, integrity, and availability. The experiment results showed that the model can detect phishing sites with an accuracy of 99.8% using Random Forest, and the hash function improves the accuracy of homoglyph attack detection.

Keywords: phishing detection; machine learning; forged websites; hash function; classification

Citation: Almuhaideb, A.M.; Aslam, N.; Alabdullatif, A.; Altamimi, S.; Alothman, S.; Alhussain, A.; Aldosari, W.; Alsunaidi, S.J.; Alissa, K.A. Homoglyph Attack Detection Model Using Machine Learning and Hash Function. *J. Sens. Actuator Netw.* **2022**, *11*, 54. https://doi.org/10.3390/jsan11030054

Academic Editor: Jordi Mongay Batalla

Received: 7 July 2022
Accepted: 7 September 2022
Published: 16 September 2022

1. Introduction

Homoglyph replacement is one of the techniques commonly used by malicious adversaries as part of their phishing campaigns and other spoofing attacks [1]. It is a way that allows attackers to manipulate characters to make two words look alike but have different values in the underlying Unicode. These can be found in various language character sets such as Latin, Cyrillic, Greek, Hebrew, Chinese, and Armenian [2]. For example, the letter "O" could be written in Latin O, Cyrillic O, and Greek O. The letters seem similar but are not assigned to the same code, the code in Latin = U + 004F, Cyrillic = U + 043E, and Greek O = U + 039F [2]. The attackers use homoglyph replacement in URL and DNS to make them seem like the original legitimate URL or DNS domain. Unicode characters have a visual appearance that is remarkably identical to or near to ASCII characters. For instance, anybody can create a (yahoo.com) domain via the clever option of Unicode characters, impossible to distinguish from the legitimate (yahoo.com) ASCII-only. As an example of the homoglyph, they are using the Cyrillic small letter "o" instead of the ASCII "o". According to [3], phishing victims by clicking a message has been increasing, just as phishing for

credentials from specific services where a single letter from the DNS domain is replaced by a similar character that looks the same.

1.1. Problem Statement(s) and Major Contributions

Hackers take advantage of the visual similarity of many Unicode characters to generate homoglyph-based URLs and DNS domains that impersonate real websites [4]. Many people and organizations may fall victim to phishing websites that appear to be ordinary and legitimate. According to a study by CHIBA et al. [5], the most popular brand domains are the most vulnerable to homoglyph attacks. Homoglyph domains harm the target's reputation and put website visitors at risk. Based on various types of phishing attacks, these threats can be exploited to steal sensitive data or gain access to secure assets. The attack on cryptocurrency exchange [6] is one example of IDN homograph attacks being a serious problem. When organizations such as Binance are targeted, the consequences for consumers and the corporation may be severe when sensitive information is disclosed. A homograph attack occurs when two different words have the same written form but vary in meaning, derivation, or pronunciation [7]. Some existing solutions use matching characters to avoid the possibility of phishing, but this matching cannot be updated with new existing methods that attackers used by implementing a higher similarity of characters as Unicode.

This paper will investigate how to detect DNS look-alike attacks and homoglyph domains accurately and efficiently, by proposing a detection model with high accuracy using hash functions and machine learning. Artificial intelligence (AI) in the form of machine learning (ML) enables computer programs to forecast outcomes more accurately without having been expressly taught to do so. Machine learning algorithms forecast new output values using historical data as input. A hash function, on the other hand, turns any size data into fixed-size values. The paper will also compare the proposed model to existing models. This paper's contributions are as follows:

- Proposes a homoglyph attack detection model that combines a hash function and machine learning.
- Achieves an accuracy of 99.8% using Random Forest, and the hash function improves the accuracy of homoglyph attack detection.
- Compares the proposed model on several criteria to existing phishing detection methods.

1.2. Related Works

The concept of homoglyph was previously discovered using character matching algorithms. Later, several studies were presented that contribute to providing solutions to the homoglyph detection domain. This research will evaluate and compare various techniques that were found in the literature to determine their efficiency in solving this problem and clarify their gaps.

In 2014 and 2017, the authors in [8,9] proposed a framework of the Knuth–Morris–Prat (KMP) pattern searching solution, which can be applied to homoglyph detection. This framework uses linear time to find pattern matching, which is considered a fast approach. KMP falls into two stages, the initial stage is for running the preprocessing algorithm that calculates the prefix function, and this function shows how the pattern matches while shifting the text. The second phase is for running the KMP matcher algorithm to find the occurrence of the pattern in the text. The major drawback of this algorithm is that it is not accurate, as it does not work with large alphabet sizes.

In 2018, the authors in [10] proposed the use of an Optical Character Recognition (OCR) system to detect an IDN homoglyph attack, as the OCR system allows full recognition of alphanumeric characters or handwritten characters by scanning the form at electronic speed. The OCR process includes character-by-character photo scanning and then converting the character image into character codes, such as ASCII. The authors utilized the OCR system to detect character similarity, as the homoglyph attack uses a domain name that visually looks like a legitimate domain name but is not. Therefore, their proposed model analyzes visual similarity by using automatic OCR to detect the homogeneous IDN attack, it converts

the URL into an image and then analyzes the image using the OCR. The main feature of this model is that it considers various readings of the same character. However, there is a security concern in this proposed model, which is a Unicode attack. Nowadays, the attackers know about these similar detecting methods, so they use different Unicode characters that are undetectable by OCR, such as looks-alike characters that could not be detected by the OCR algorithms that rely on the visual representation of characters. Additionally, the authors in [11] tried to increase the OCR-based solutions' accuracy by adopting the KNN (K-Nearest Neighbor) machine learning classifier to know about the look-alike characters, where the solution depends on calculating the similarity distances between the points. A high recognition rate was observed while testing the accuracy and overall working of this character-recognition approach. However, the drawbacks of this algorithm are the high dependency, it is considered a lazy algorithm, and it requires a large testing time.

In 2019, the authors in [12] developed a framework called ShamFinder. The main idea of this framework is to build a new homoglyph database named SimChar. The database automatically updates to detect homoglyphs in IDN. ShamFinder first collects registered domain names for a TLD (Top Level Domain). Then, from the collected domain names, it extracts IDNs and compares them to the homoglyph database, which consists of SimChar and UN (Unicode confusable). The framework is effective in terms of two factors, the human perspective and the computational cost. The human perspective factor depends on determining whether the human realizes the confusability of the homoglyph, the confusability can be scaled as very distinct, distinct, neural, confusing, and very confusing. The main drawbacks of this framework are demonstrated in three limitations, namely, confusability test, font type, and measurement target.

In 2020, the authors in [13] proposed a Siamese Neural Network architecture to obtain the existing likeness of domain names and process names to detect homoglyph and spoofing attacks. The process is represented mathematically as {(dpi, DP', si, Si) n-i = 1}, S ∈ {0,1}, where DP is considered as the domain name, DP' is known as the spoofed domain, and S refers to the similarity which is equal to either 1 or 0 (0 = similar and 1 = dissimilar). The selected model has two similar neural networks. Each of the neural networks obtains different data with consideration of shared weights, and the outputs are integrated into an easy proportional energy method applying Euclidean distance. The Euclidean distance is utilized to measure the vector distance of strings, where a lesser distance shows increased likeness, and a large distance shows a difference. The key advantages of this architecture are its effectiveness and scalability. Security, on the other hand, has an adequate level of accuracy, but they ignored integrity and availability entirely.

Most of the previous studies on detecting homoglyphs are focusing on character matching or character similarity, which produced an inaccurate percentage of detecting the homoglyph attack. They also lack fulfilment of all requirements necessary as discussed in Section 3. Therefore, a novel solution will be proposed to meet the objective and find the best technique to detect homoglyphs.

1.3. Roadmap of the Paper

The rest of this paper is organized as follows. Section 2 presents the proposed solution using machine learning and hash function. An evaluation of the proposed solution is compared to the existing approaches in Section 3, in terms of performance, functionality, and security. A threat model analysis of the proposed approach is presented in Section 4. The conclusion is finally presented in Section 5.

2. Material and Methods

This section describes the proposed model for detecting homoglyph and phishing attacks, which is summarized in Figure 1. The model consists of two phases, development and deployment.

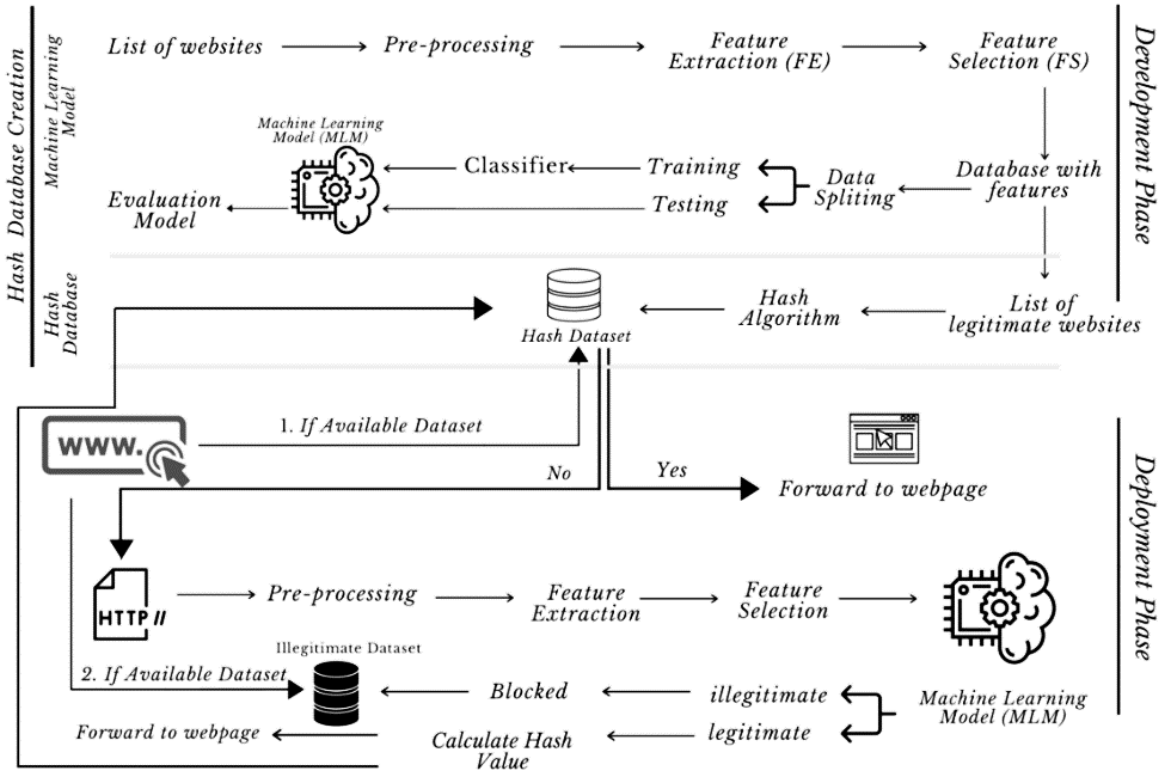

Figure 1. Proposed methodology.

The development phase first starts with preprocessing, a technique to turn the raw data into a processed clean dataset so that it can be useful for further analysis. Second, feature extraction turns the raw data into a set of features. Third, the selection of features uses a subset of the original set of features to obtain a smaller subset that can be used to model the problem. Feature selection lowers the resources needed for computation without missing essential or related information, decreases the redundant data for analysis, facilitates speed, and enhances the accuracy of the machine learning model. The outcome of the feature selection process is the dataset with a reduced set of features that can better represent the problem. Then, several classifiers will be used to train the model. Finally, the evaluation phase will represent the effectiveness of the proposed model to solve the homoglyph problem. The deployment phase describes the actual process of our proposed idea.

Hashing is used, as it is a process of converting plain text into encoded fixed-length text, as shown in Figure 2, and it is a one-way function that produces a unique value for each input file. The hash value will help in differentiating between look-alike URLs, such as having the URL www.google.com, (accessed on 18 July 2022) replace a letter from different languages, which results in the same domain name based on the appearance except that the hash value is what makes each one different from the other. The hash algorithm is used to enhance the accuracy of the results; the hashing property will detect any slight alteration or modification of the URL

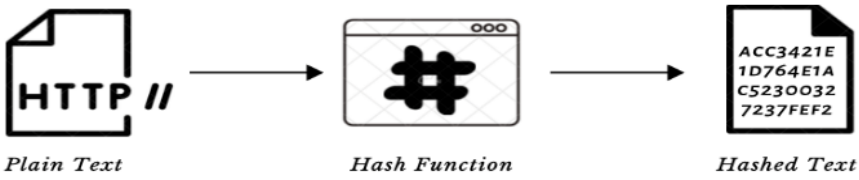

Figure 2. Hash function process.

The example below is a slight change in the URL that cannot be identified with the naked eye. The attacker takes advantage of look-a-like characters from a Latin Cyrillic, Greek, Hebrew, Chinese, and Armenian character to apply a homoglyph attack. For example, the attacker may change the "or" letter of the original Google domain into a Greek "or" letter, resulting in an entirely different hash value of the Google domain.

- The hash value of the original domain name: www.google.com, (accessed on 18 July 2022).

  191347BFE55D0CA9A574DB77BC8648275CE258461450E793528E0CC6D2DCF8F5

- The hash value after changing only one 'o' of the original domain name into a Greek 'o': www.google.com (accessed on 18 July 2022).

  79189C0C37E6A0EE3F83D81CB2365CEB9F1226356AC57D836851F9E1FAD44A33

By taking advantage of the hash function to store the hash of each legitimate URL, any manipulation by attackers on the URL will be detected. Figure 3 shows the process of creating the hash dataset, which will be discussed in the following.

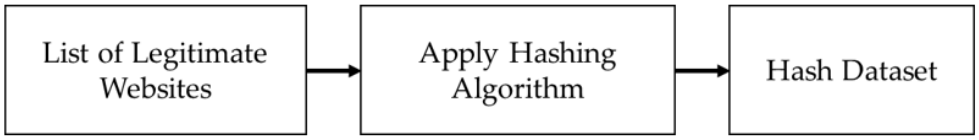

Figure 3. The process of creating the hash datasets.

The following subsections will discuss the two phases of the model, which are development and deployment.

2.1. Development Phase

In this section, we discuss the process of training machine learning including preprocessing, feature extraction, feature selection, and data splitting. Then, we explain the method of classification and evaluation using the model. The process of data analysis including selection, processing, and feature extraction and selection took 6 weeks.

2.1.1. Data Selection

In this phase, many approaches towards the research were conducted to find a suitable dataset that meets the requirements and goals. The dataset consists of 70,000 URLs combined by extracting 35,000 legitimate URLs from Alexa [14], and 35,000 illegitimate URLs extracted from PhishTank and PhishStat. These datasets were chosen because they are available in different formats and are open-source, and the Alexa dataset [14] contains the top one million legitimate websites, which is the largest amount used in previous studies. Furthermore, to cover all possibilities of phishing and homoglyph websites, a homoglyph dataset was generated since there is no homoglyph dataset available. We generated our homoglyph dataset from the legitimate dataset by using a Homoglyph generator.

2.1.2. Data Preprocessing

In the preprocessing data phase, the raw data for both legitimate and illegitimate datasets was simplified into a meaningful format, to minimize the redundant, inconsistent, and incomplete data involving the steps as follows:

Data Cleaning: This phase is working on cleaning any useless data by removing duplicated records in our dataset. Furthermore, some records have an incomplete URL that is considered inconsistent data.

- Data Integration: There is a large imbalance between the legitimate and illegitimate datasets, which leads to model overfitting for the majority class. To avoid this problem, we combined two illegitimate datasets (PhishTank 20% and Phishstat 80%) to have a more considerable record and to be more consistent with the legitimate dataset. We

balance the distribution number of records per class label as 50% for illegitimate URLs and 50% for legitimate in our dataset.

- Data Transformation: This is a pivotal role in converting unprocessed data into an understandable form. In this phase, we normalized, aggregated, and generalized our datasets. This modification helps to minimize time and complexity by arranging the columns and creating a summary for a faster overview.
- Data Reaction: This phase involves reducing large amounts of data into smaller and more meaningful fragments that can extract the features directly from it into more small and significant chunks.

2.1.3. Feature Extraction

This section will explain how the system finds the best representation of the dataset to learn and train the machine learning algorithm to detect homoglyph attacks. Thus, transforming the uncleaned raw data into preprocessed features that clearly depict the core problem of the predictive models, resulting in improved model accuracy on unseen data. The extracted features are categorized from the dataset into six respective areas:

- Blacklist Features: Contain existing URLs that are classified as a blacklist [15,16].
- Lexical Features: It focuses on string properties such as length, alpha-numeric distribution of characters, and several special characters ($-_.+!*(),%) [15–17].
- Host-Based Features: It contains the IP address characteristics, WHOIS information, location, and domain name properties [15,17].
- Content Features: It downloads content hosted via URL to acquire information about HTML and JavaScript features [15,17].
- URL-Based Features: For example, the protocol (HTTP and HTTPS), subdomain (www), domain (google), top-level domain (TLD) which has different types such as (infrastructure, generic, sponsored, country-code, and test), and path [18].
- URL-Based Language Features: It helps to extract the features of non-English languages from URLs using words as features and N-grams [19].

The website URLs are passed to the feature extractor that acquires feature values with the help of the already extracted and mentioned URL-based features. Various related studies were selecting and extracting different features from the URL. In this section, we present different research on varied feature selection, to find the most common features that help to detect phishing.

Utilizing the IP address: If the URL contains an IP address, which means it is used as an alternative domain name. For example (192.168.200.1...), from here, we can say that someone tries to steal the user's information.

1. URL length: The attacker can take advantage of a long URL to hide suspicious content. For instance, "https://luizaonlaine.com/5257iuhamkvnma024/index.php?o-de-panelas-tramontina-antiaderente-de-aluminio-vermelho-10-pecas-turim-20298-722%25252Fp%25252F144129900%25252Fud%25252Fpanl%25252F&id=1" (accessed on 18 July 2022). And to ensure accuracy, we calculate the total average of URLs and if ">75" it will be classified as phishing websites. Otherwise, it will be categorized as a legitimate website.
2. Special characters: Phishing websites usually hide suspicious content in the URL by using special characters. For example, the symbol "@" leads the browser to ignore the rest of the URL before the "@" symbol.
3. Redirecting using "//": This symbol '//' is used to forward the user to other websites since we eliminate (http//and https//). The presence of // indicates that the website is illegitimate.
4. Prefixes or suffixes disjointed by (-) to the domain: The attacker confuses the user by accumulating a prefix or suffixes disconnected from this symbol "-" to convince the user that the URL is legitimate, for example, (www.payment-amazon.com (accessed on 18 July 2022)).

5. Number of subdomains: The number of subdomains can be utilized by the number of dots (.). Based on the datasets, in case the number of dots is more than four, that means the website will be classified as illegitimate.
6. The occurrence of "HTTPS" in the domain of the URL. The attacker will have the ability to attach "HTTPS" symbols to the domain of the website URL so as to fool the audience, such as http://https-www-paypal-it-webapps-mpp-home.soft-hair.com/ (accessed on 18 July 2022). The phisher will probably use HTTP instead of HTTPS.
7. Homoglyph characters: The attackers manipulate the characters of the URL to be look-alike legitimate URLs using various writing options such as Latin, Cyrillic, and Greek to trick the user. For example, "www.google.com", (accessed on 18 July 2022) looks legitimate, but it uses Cyrillic letters to write "Google".

2.1.4. Feature Selection

The method chosen to implement the feature selection is the wrapper method. The wrapper method shortlists the feature set in the form of a search problem, where different groupings have been made, determined, and contrasted with other combinations. Wrappers work slowly in comparison to filters in regard to determining effective subsets as they rely on the resource requirement of the feature selection algorithm, despite the fact that the wrapper methods are slower than the filter. However, the filter method selects the feature without considering the classification model used and sometimes fails to achieve good outcomes. Therefore, the wrapper method was used as the main objective of our study is the accurate detection of homoglyph attacks. Figure 4 shows the process of the wrapper method [20].

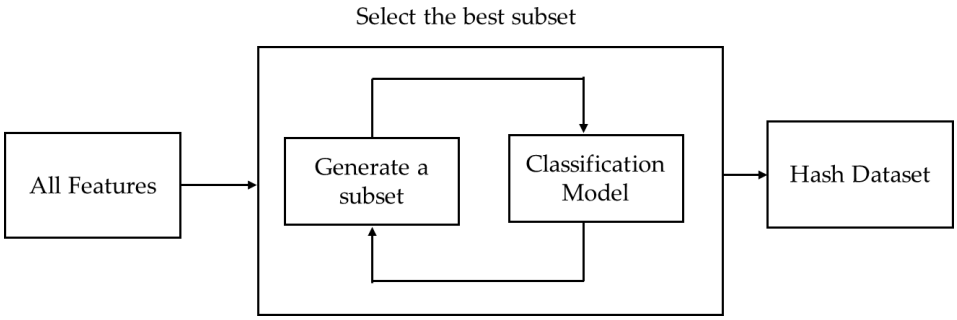

Figure 4. Wrapper method.

The wrapper has two methods: the forward selection method starts with one or a few features selected according to a method-specific selection criterion, and then more features are added iteratively until a stopping criterion is met; the other method is the backward elimination method that starts with all features and iteratively removes one feature or bunches of features. In the current study, we used recursive feature elimination (RFE) for the feature selection.

2.1.5. Data Splitting

After feature extraction and selection, all gathered data are collected in a new dataset. There are various methods for data partitioning for training and testing. We used the standard data partitioning (70-30) method, where normally approximately 70% of the total data is utilized for training and the rest of the 30% of data is utilized for testing [21].

2.1.6. Classifier Selection

This subsection discusses some classification techniques used in machine learning, such as SVM, artificial neural network (ANN), decision tree (DT), and k-nearest neighbor (KNN). Classification is a supervised machine learning technique that predicts the value of a categorical variable by using several numerical or categorical features.

A comparison of the four algorithms was made using "Weka software" in terms of accuracy for each algorithm. Table 1 shows the outcome summaries acquired after implementing the relevant classifiers on the data, Table 1 also depicts that RF attained the highest accuracy in comparison to other classifiers.

Table 1. Accuracy of the models used in the current study.

Algorithms	Accuracy
Support Vector Machine	99.3%
Decision Tree	98.9%
K Nearest Neighbor	99.5%
Random Forest	99.8%

To specify the method to be applied, an analysis must be undertaken first to identify the classifier we will use. To do this, as shown in Table 2, we create a summary to compare the different classifiers used in various research as used previously in the feature extraction phase. The technique that produced the highest results is underlined in the table. The benchmarking will analyze the different papers and the classifier used by each paper, as well as the tools used and the highest findings in each classifier. By using Weka, we used different types of classifiers in different folds and percentages to have a more accurate result. We worked on a quickie, which is the Naïve Bayes, the laziest, which is Kstar, a type of KNN classifier, and the Random Forest, which is the classifier most used in the previous studies.

Table 2. Summary of the studies mentioned in the literature review.

Ref.	Year	Dataset	Technique	Findings
[22]	2014	PhishTank, Yahoo directory	Multi-label-classifier-based, Associative Classification (MCAC)	The accuracy (%) approximately is: MCAC Accuracy = 94.5%
[23]	2017	UCI	KNN, SVM, and Random Forest	NA
[24]	2017	PhishTank, Statscrop	ELM, LC-ELM	ELM, LC-ELM achieved: accuracy = 99.04%
[25]	2018	PhishTank and Google	DT, RF, GBM	Accuracy = 98.4%, recall = 98.59 Precision = 97.70%
[26]	2018	UCI	ELM	Accuracy = 95.34%
[27]	2018	UCI	lazy K.Star	Accuracy = 97.58%
[28]	2018	UCI	HEFS	Accuracy = 94.6%
[29]	2018	PhishTank	RF	Accuracy of 95%
[18]	2019	PhishTank's and Ebubekirbbr	DT, NN, NB	Accuracy = 78.4%
[17]	2019	PhishTank, Alexa records, UCI	FVV	Accuracy = 94.5% Precision = 96.4%, recall = 93.6%, F1_score = 95.4%
[30]	2019	Phishload, PhishTank	Longest Common Subsequence (LCS), Damerau–Levenshtein Edit Distance (DLE)	90.54% true positive, 94.18% true negative, 5.82% false positive, 9.46% false negative, and 92.72% accuracy.
[31]	2019	PhishTank, Openfish	RF	Accuracy = 97.98%
[32]	2020	PhishTank, Google	C4.5 classifier	Accuracy = 89.5%, precision 88.26% Recall = 89.39%, F_measure = 91.16%

To summarize Table 2, we simplified the results that achieved the highest accuracy, precision, Recall, F1_measure, and false positive to conclude the best achievement among the papers and to prepare our development phase to have the highest result. Table 3 shows

the highest test results of the implemented classifiers in the literature review, which helped with choosing the best classifier.

Table 3. Highest results of classifiers.

Ref.		Number of Features Used	Highest	Name of Model
[33]	Accuracy	16	99.04%	ELM, LC-ELM
[25]	Precision	6	97.70%	Random Forest
[25]	Recall	5	98.59%	Random Forest
[23]	F1_Measure	3	96%	Random Forest
[33]	False positive	2	0.53	LC-ELMs

2.2. Deployment Phase

After training the machine learning algorithm, as shown in Figure 5, this phase goes through the implementation of the system to determine the legitimacy of the clicked URL. The process sequences are clarified below. In the deployment phase, when the user clicks the URL, there are three cases.

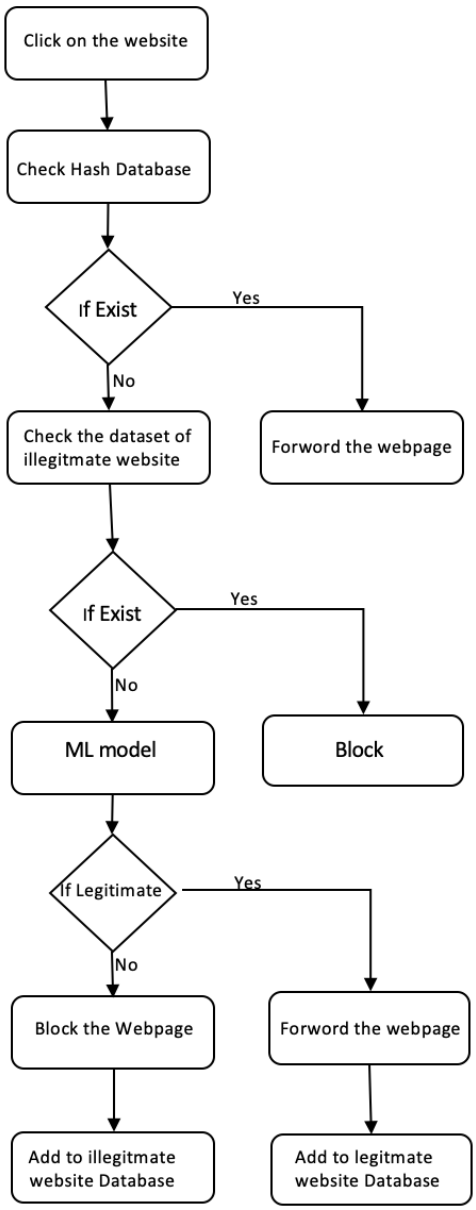

Figure 5. The sequential flow of the deployment phase.

- Case 1: Check if the URL exists in the hash dataset:
  - If it does exist, the user will be forwarded to a website.
  - If it does not exist in the hash dataset, check the illegitimate dataset.
- Case 2: If the URL is existing in the illegitimate dataset, block the website.
- Case 3: If the URL does not exist in both "Hash Dataset" and "Illegitimate Dataset", then it will go through the process of machine learning.

Employing machine learning to detect homoglyph attacks will help the users to protect themselves from being victims of phishing websites. It is accomplished by utilizing a dynamic dataset to be further utilized on new websites. The comparison will be between the URL that is clicked by users and the URL that exists in a dataset. The process is completed quickly and accurately as the hash function is used for storing the dataset.

The system aims to input the URL and extract features to use it in the classifier, and then the classifier decides if the URL is legitimate or not. The machine learning model is trained using a dataset of legitimate and illegitimate URLs. The hash function is used to store legitimate URLs. Furthermore, if the user clicks the homoglyph URL, only the hash value is checked for the decision. However, if the clicked URL is legitimate, the user is allowed through, otherwise, the request is blocked.

2.2.1. Homoglyph Detection Implementation

When implementing the Homoglyph detector, it can be placed as a proxy to act as a middleware between a PC user and internet users for making sure the system is safe and secure. It is an automated process that sits between the firewall and internet proxy to process and analyze all requests from users in and out before it goes to the internet as shown in Figure 6. The proxy utilizes multiple servers at the same time to control all requests.

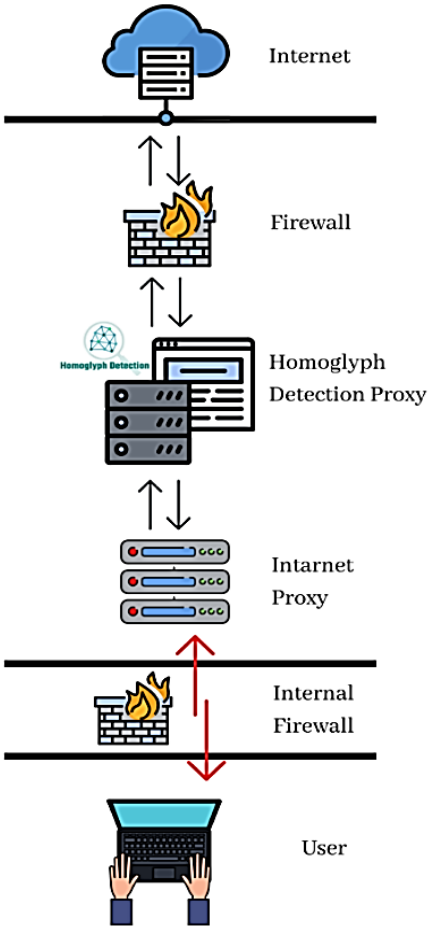

Figure 6. Proxy model.

2.2.2. Accessibility Interface Prototype

Users in the system have only two functions. They are allowed to enter a URL to check the legitimacy of the entered URL and provide feedback in the case of disagreement with the system's decision, as Figure 7 shows. Allowing the user to add feedback will help the admin to reduce the false-positive rate.

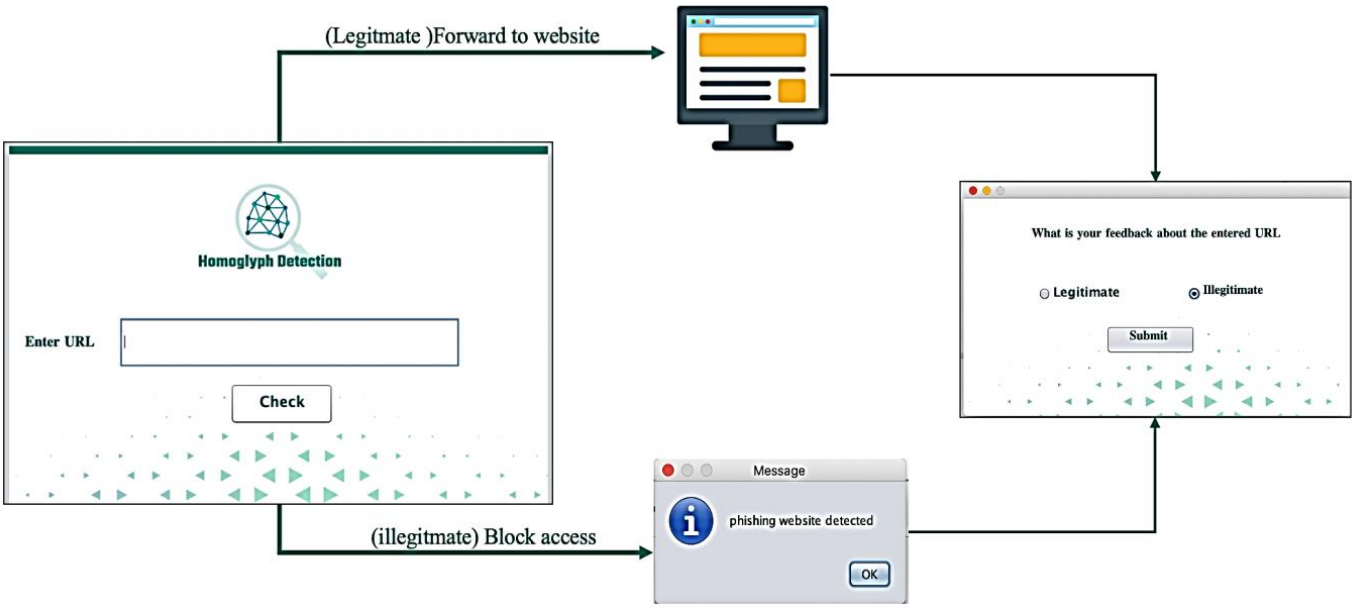

Figure 7. User access scenario.

Here are two scenarios for the user to interact with the machine learning process.

- Legitimate scenario: When the user enters a legitimate URL, the pop-up message will show that the URL is legitimate and ask the user if they want to continue with the URL of the webpage or not as shown in Figure 8.

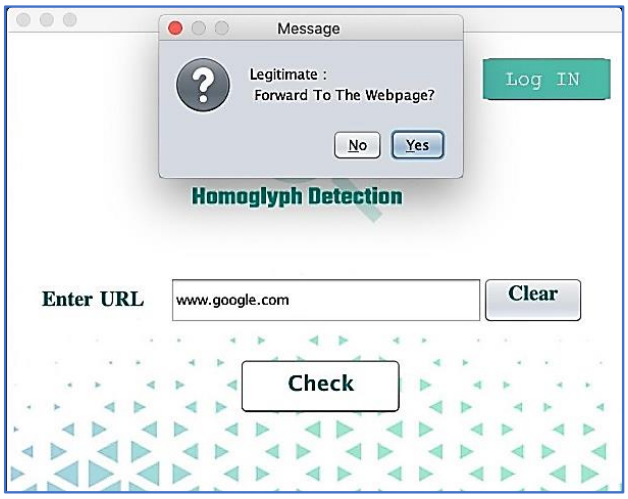

Figure 8. Legitimate Site.

- Illegitimate scenario: When the user enters a mix of IP and homoglyph letters, e.g., www.talente\T1\dj/192.168.4.8, (accessed on 18 July 2022) as shown in Figure 9.

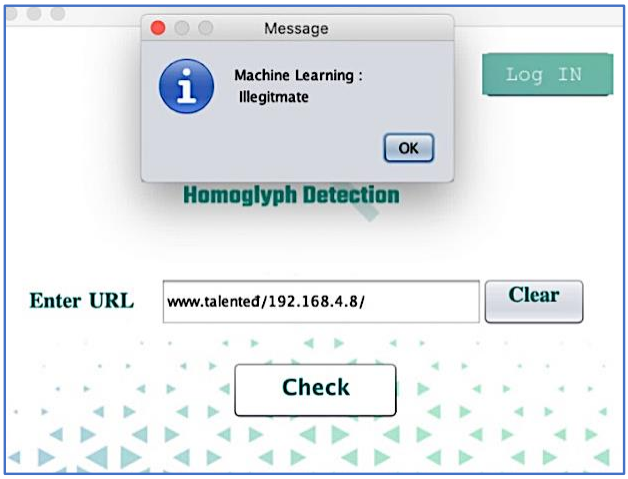

Figure 9. Phishing site.

3. Experimental Analysis, Observations, and Results

This section provides a clear justification for what makes the homoglyph detector a better solution in terms of different criteria in Table 4. This will be completed through a comparison based on the following solution requirements.

Table 4. Comparison with existing techniques.

Scheme/Feature	Performance					Functionality			Security		
	Speed	Effectiveness	Scalability	Error Rate	Ease of Use	Implementation	Economical	Independence	Accuracy	Integrity	Availability
Optical character recognition (OCR) [10]	○	●	◑	○	◑	○	●	○	◑	□	□
Fuzzy logic controller [34]	○	○	●	◑	●	○	○	□	◑	□	□
Siamese neural network [13]	□	●	●	○	◑	○	◑	○	◑	□	□
ShamFinder [12]	●	●	●	◑	●	◑	□	○	●	□	□
k-nearest neighbors algorithm (k-NN) [11]	○	□	○	◑	●	●	○	○	○	□	□
HitZone map [28]	○	●	◑	○	◑	○	□	○	●	□	□
KMP [9,35]	●	●	◑	◑	○	●	◑	●	○	□	□
Ours	●	●	●	○	●	●	○	○	●	●	●

○ Does not provide property; ● Provides property; ◑ Partially provides property; ○ Almost provides property; □ Null.

3.1. Performance Analysis

The system has shown its efficiency in terms of speed, scalability, effectiveness, error rate, and ease of use. In terms of speed, the homoglyph detector is similar to the ShamFinder and KMP, which takes approximately 0.07 s to show the result per user. In terms of effectiveness, the system is functioning as expected, and different scenarios were tested to validate the operation of the system. The system provides scalability by having a flexible database that accepts a large number of new URLs that need to be processed using machine learning, as shown in Figure 10. The system scalability will benefit system performance and efficiency and enhance system security. Other languages can be included in the future to support expanding the system all over the world and detect phishing and homoglyph attacks in global ways. Furthermore, it is easy to implement and use. The user must write the URL that needs to be checked and press the check button to obtain the result.

In addition, there is a small possibility to have a wrong result; to fix that and enhance the system performance, we added a feature to obtain user feedback. Users can provide feedback on the result and send it to the admin. Later, the admin will check all user feedback and make sure that the ML has provided a correct decision on the URL and that it has been added to the dataset.

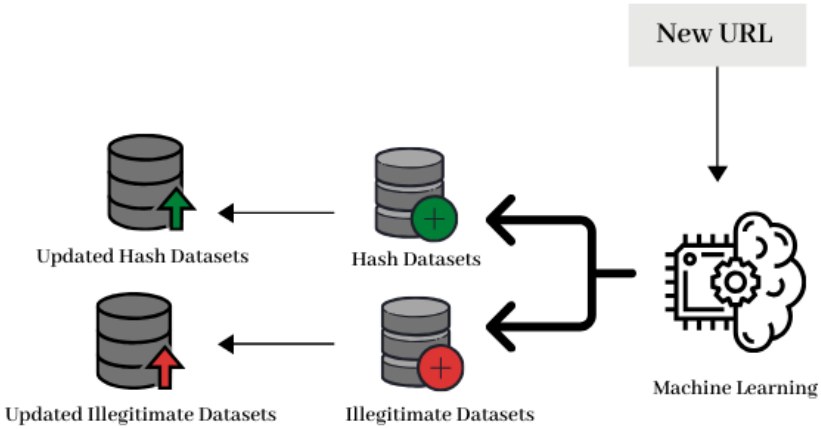

Figure 10. Dataset growth.

3.2. Functionality Analysis

The functionality of this program depends on three features, namely: implementation, economics, and independence. The program is implemented in Java using Weka libraries. However, can be implemented in different programming languages. We considered it as economical because it only requires software installation of the solution into the organization server with no additional hardware or tools required. Moreover, independence is measured based on the need for external data or software to operate the homoglyph detector.

3.3. Security Analysis

In the security aspect, integrity, accuracy, and availability were considered. Where integrity means the data was used as is and was not subject to any changes or modifications. Availability means you can reach the system and data whenever needed and there are no obstacles to reaching what you are supposed to reach. While accuracy is a percentage of how true the result is compared to reality. Comparing these three aspects with related works shows that the homoglyph detector provides better results in conjunction with machine learning. With the software combined with machine learning, it enhances compliance and maintenance strategies for the detection of defects. With the help of this out-of-the-box approach, the rate of data stored and captured will rise and will offer more practical and real-time data comparison and will be flexible for new data that will be utilized for conditional determination. This revolutionary approach will provide real-time data analysis and allow for the use of new data for assessment. The data integrity cannot be altered unless the admin is allowed because s/he has the ability to add data manually. Talking about the accuracy, as shown in Table 5, the system depends upon the result of the model used "Random Forest," which achieved 99.82%, and it can be higher over time because of the machine learning and the update in the datasets illustrated in Figure 10. While the ShamFinder approach [12] uses machine learning but they used fixed datasets that do not add new data over time, so the rate of accuracy is unlikely to improve, which is 97%. Regarding the availability, the homoglyph detector has proven its availability as it is considered available to the user as a first step for entering a webpage. Moreover, the solution can function on different web browsers and operating systems properly. As an example of the percentage comparison with previous related work [11], homoglyph detection shows a higher result in Table 5.

Table 5. Homoglyph detection accuracy.

Criteria	Homoglyph Detection	OCR Improved by KNN [11]
Accuracy	99.8%	96.4%
Precision	99.6%	93%
Recall	99.8%	92%
F1_measure	99.7%	91%
False Positive	0.2%	3.6%

4. Discussion

In this section, we will discuss several attacks that may target the system and will threaten the functionality and security of the system and how these attacks will be mitigated. Figure 11 illustrates a threat model containing potential threats targeting admin datasets and machine learning algorithms.

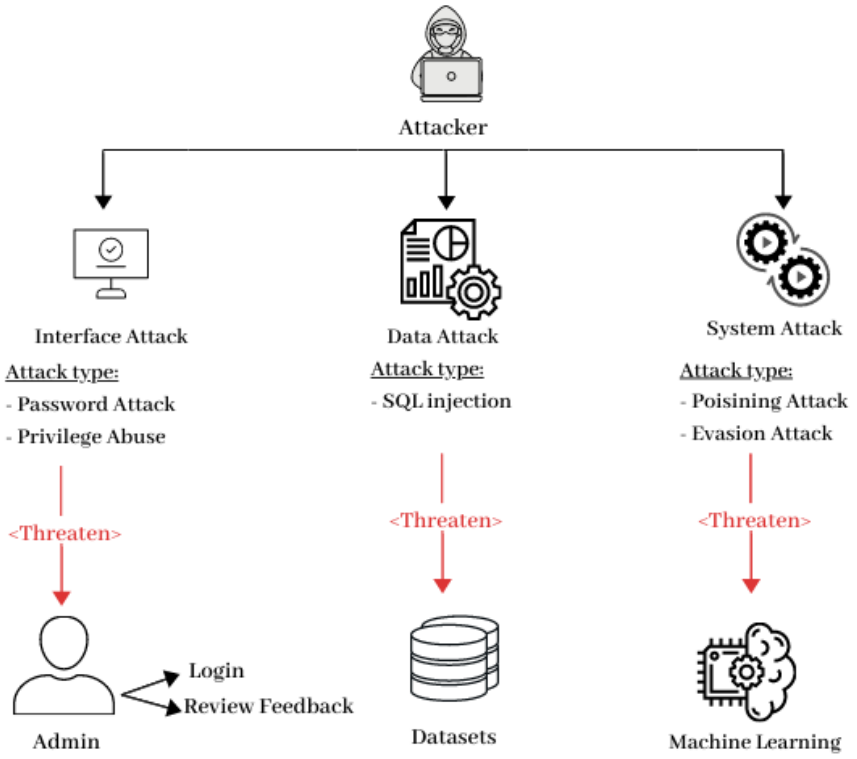

Figure 11. Threat modeling of the proposed solution.

4.1. Interface Attack

The following attacks targeted the admin account:

- Password attack: The attacker can gain unauthorized access to the system through an admin account. The following are examples of password attacks.
- Brute force attack: It is an attack when multiple attempts and combinations are tested by the hacker to attain the account password.
- Dictionary attack: The technique utilized to break the system and gain unauthorized access by going through every password in the dictionary word list.
- Rainbow attack: Method used to compare hashed passwords in the rainbow table with the entered password.
    - Mitigation: (1) limit failed login attempts; (2) two-factor authentication (2FA); (3) using a strong password.
- Privilege abuse: A privilege abuse attack occurs when the attacker tries to use the admin account inappropriately by entering the system with the admin account.

- Mitigation: (1) access control: regulating access to the system by using ACL to minimize the risk of unauthorized users. (2) Audit trail: auditing and logging the admin's activities.

4.2. Data Attack

The following attacks target the data:

- SQL injection: The SQL injection attack is when the attacker tries to gain unauthorized access by injecting a malicious SQL query to retrieve the desired information from the database. In the case of the proposed model within this paper, the attacker tends to replace the URL with an SQL query [36]. That adds an illegitimate URL in the legitimate dataset to ensure future access to the website.
- Mitigation: The input validation mechanism must occur. The input validation will check the input given by the user. The input must be a URL that will be checked by the machine to determine its legitimacy and add it to the suitable dataset. If the input were an SQL query, the system would drop the input to block any attempt from the user to apply the direct modification to the dataset, which is the responsibility of the machine only [36].

4.3. System Attacks

The following attacks target the system:

- Evasion attacks: They are the most prevalent type of attacks that may be encountered during system operation. This attack happened during the learned model, e.g., in our case, the attacker changes some words in the URL and it seems readable, but actually, the system fails to classify the result in the classification process of the ML.
- Mitigation: It evaluated the efficacy of the classifier model by validating the input as shown in Figure 12.

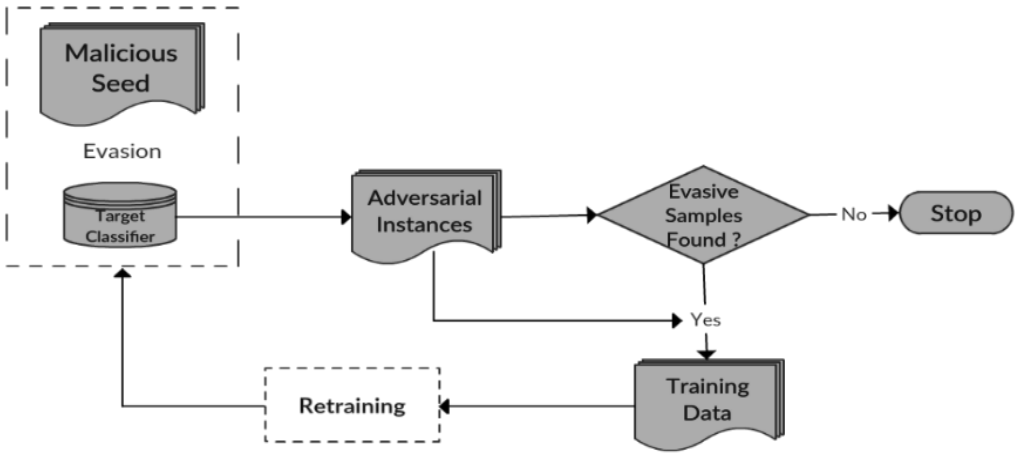

Figure 12. Retraining classifier.

- Poisoning attacks: This attacker is targeting ML during the deployment model. This attack targets the availability and integrity of the ML by injecting so much bad data into the system that whatever boundary the model learns becomes useless.
- Mitigation: The most common type of defense is outlier detection. The idea is when the infection is poisoning the machine learning system, the attacker is injecting something into the training aggregation that is very different from what it should include, and this should be detected.

The experiments and key findings show that the proposed model is effective at detecting homoglyph attacks. Moreover, the proposed model was evaluated against various types of attacks.

5. Conclusions

This paper considered visually similar characters known as homoglyphs. The similarity of these characters "Unicode or ASCII" has recently been used in different kinds of attacks such as phishing attacks and other homoglyph attacks. In this paper, various techniques were analyzed and compared to come up with the best solution that helps to detect the homoglyph within URLs and DNS domains to reduce the number of attacks being launched. As a result, a hashing technique was proposed with machine learning to detect these malicious websites. The model approach consists of two phases. The development phase worked on training the machine and achieved the highest accuracy of 99.8 using a Random Forest classifier after testing. The deployment phase consisted of deploying the model with a Java interface and testing the results with real interaction with the user. The URL checked with the hash adds more accuracy to the results where any slight modification or alteration of the URL will be easily detected. The homoglyph detector can be implemented as standalone software that acts as a first step in requesting a webpage. It helps increase the security of the browser and helps organizations utilize this feature to protect their websites against phishing attacks. In spite of the considerable outcome achieved by the proposed model, there is still room for further improvement. In the current study, Random Forest achieved the highest results. However, the nature of the RF model is a black box. It lacks interpretability and explainability. Furthermore, the dataset needs to be expanded to train the model for new types of attacks.

Author Contributions: Conceptualization, A.M.A., N.A., A.A. (Almaha Alabdullatif), S.A. (Sarah Altamimi), S.A. (Shooq Alothman), A.A. (Amnah Alhussain), W.A. and S.J.A.; data curation, N.A. and A.A. (Almaha Alabdullatif); formal analysis, N.A., A.A. (Almaha Alabdullatif), S.A. (Sarah Altamimi), S.A. (Shooq Alothman) and A.A. (Amnah Alhussain); funding acquisition, A.M.A. and K.A.A.; investigation, N.A., A.A. (Almaha Alabdullatif), S.A. (Sarah Altamimi), S.A. (Shooq Alothman), A.A. (Amnah Alhussain) and W.A.; methodology, A.M.A., N.A., A.A. (Almaha Alabdullatif), S.A. (Sarah Altamimi), S.A. (Shooq Alothman), A.A. (Amnah Alhussain) and W.A.; project administration, A.M.A.; resources, A.M.A., N.A. and S.J.A.; software, A.A. (Almaha Alabdullatif), S.A. (Sarah Altamimi), S.A. (Shooq Alothman), A.A. (Amnah Alhussain) and W.A.; supervision, A.M.A., N.A. and S.J.A.; validation, N.A., A.A. (Almaha Alabdullatif), S.A. (Sarah Altamimi), S.A. (Shooq Alothman), A.A. (Amnah Alhussain), W.A., S.J.A. and K.A.A.; visualization, N.A., A.A. (Almaha Alabdullatif), S.A. (Sarah Altamimi), S.A. (Shooq Alothman), A.A. (Amnah Alhussain), W.A. and S.J.A.; writing—original draft, A.A. (Almaha Alabdullatif), S.A. (Sarah Altamimi), S.A. (Shooq Alothman), A.A. (Amnah Alhussain), W.A. and S.J.A.; writing—review and editing, A.M.A., N.A., S.J.A. and K.A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by SAUDI ARAMCO Cybersecurity Chair at Imam Abdulrahman Bin Faisal University, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to express their appreciation to the Journal Editor, an Associate Editor, and the five anonymous reviewers for their insightful comments. We would like to thank Aminullah Tora (at the Information Protection Department, Saudi Aramco Company, Dhahran, Saudi Arabia) for the early discussion on the use of ML as a potential solution for the detection of Homoglyph attacks. Additionally, we would like to thank him for the paper proofreading. We gratefully acknowledge Nazar A. Saqib, and Maryam T. Ahmed, for their comments and revisions. We would further like to thank Imam Abdulrahman Bin Faisal University for facilitating access to the resources used in this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Woodbridge, J.; Anderson, H.S.; Ahuja, A.; Grant, D. Detecting Homoglyph Attacks with a Siamese Neural Network. In Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 24 May 2018; pp. 22–28. [CrossRef]
2. Elsayed, Y.; Shosha, A. Large scale detection of IDN domain name masquerading. In Proceedings of the 018 APWG Symposium on Electronic Crime Research (eCrime), San Diego, CA, USA, 15–17 May 2018; pp. 1–11. [CrossRef]
3. Thao, T.P.; Sawaya, Y.; Nguyen-Son, H.-Q.; Yamada, A.; Kubota, A.; Van Sang, T.; Yamaguchi, R.S. Human Factors in Homograph Attack Recognition. In *Applied Cryptography and Network Security*; Springer: Cham, Switzerland, 2020; pp. 408–435.
4. Simpson, G.; Moore, T.; Clayton, R. Ten years of attacks on companies using visual impersonation of domain names. In Proceedings of the 2020 APWG Symposium on Electronic Crime Research (eCrime), Boston, MA, USA, 16–19 November 2020. [CrossRef]
5. Chiba, D.; Hasegawa, A.A.; Koide, T.; Sawabe, Y.; Goto, S.; Akiyama, M. DomainScouter: Analyzing the Risks of Deceptive Internationalized Domain Names. *IEICE Trans. Inf. Syst.* **2020**, *E103.D*, 1493–1511. [CrossRef]
6. Summary of the Phishing and Attempted Stealing Incident on Binance–Binance. pp. 7–9. Available online: https://support.binance.com/hc/en-us/articles/360001547431-Summary-of-the-Phishing-and-Attempted-Stealing-Incident-on-Binance (accessed on 18 July 2022).
7. Helfrich, J.N.; Neff, R. Dual canonicalization: An answer to the homograph attack. In Proceedings of the 2012 eCrime Researchers Summit, Las Croabas, PR, USA, 23–24 October 2012; pp. 1–10. [CrossRef]
8. Pandey, G.; Martolia, M.; Arora, N. A Novel String Matching Algorithm and Comparison with KMP Algorithm. *Int. J. Comput. Appl.* **2017**, *179*, 6–8. [CrossRef]
9. Pandey, S.K.; Dubey, N.K.; Sharma, S. A Study on String Matching Methodologies. *Int. J. Comput. Sci. Inf. Technol.* **2014**, *5*, 4732–4735.
10. Sawabe, Y.; Chiba, D.; Akiyama, M.; Goto, S. Detecting homograph IDNs using OCR. In Proceedings of the Asia-Pacific Advanced Network, Singapore, 25–29 March 2018; pp. 56–64.
11. Barnouti, N.H.; Abomaali, M.; Al-Mayyahi, M.H.N. An efficient character recognition technique using K-nearest neighbor classifier. *Int. J. Eng. Technol.* **2018**, *7*, 3148–3153. [CrossRef]
12. Suzuki, H.; Chiba, D.; Yoneya, Y.; Mori, T.; Goto, S. ShamFinder: An automated framework for detecting IDN homographs. In Proceedings of the Internet Measurement Conference, Amsterdam, The Netherlands, 21–23 October 2019; pp. 449–462. [CrossRef]
13. Vinayakumar, R.; Soman, K.P. Siamese neural network architecture for homoglyph attacks detection. *ICT Express* **2020**, *6*, 16–19. [CrossRef]
14. Alexa Top 1 Million Sites. Available online: https://www.kaggle.com/datasets/cheedcheed/top1m (accessed on 18 May 2022).
15. Sahoo, D.; Liu, C.; Hoi, S.C. Malicious URL Detection using Machine Learning: A Survey. *arXiv* **2017**, arXiv:1701.07179.
16. Le, H.; Pham, Q.; Sahoo, D.; Hoi, S.C.H. URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection. *arXiv* **2018**, arXiv:1802.03162.
17. Zhu, E.; Chen, Y.; Ye, C.; Li, X.; Liu, F. OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network. *IEEE Access* **2019**, *7*, 73271–73284. [CrossRef]
18. Vanhoenshoven, F.; Napoles, G.; Falcon, R.; Vanhoof, K.; Koppen, M. Detecting malicious URLs using machine learning techniques. In Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, Greece, 6–9 December 2016; pp. 1–8. [CrossRef]
19. Baykan, E.; Henzinger, M.; Weber, I. A comprehensive study of techniques for URL-based web page language classification. *ACM Trans. Web* **2013**, *7*, 1–37. [CrossRef]
20. Talavera, L. An Evaluation of Filter and Wrapper Methods for Feature Selection in Categorical Clustering. In *Advances in Intelligent Data Analysis VI*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 440–451.
21. Kotthoff, L.; Thornton, C.; Hoos, H.H.; Hutter, F.; Leyton-Brown, K. Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization. In *Automated Machine Learning: Methods, Systems, Challenges*; Springer: Cham, Switzerland, 2019; p. 81.
22. Abdelhamid, N.; Ayesh, A.; Thabtah, F. Phishing detection based Associative Classification data mining. *Expert Syst. Appl.* **2014**, *41*, 5948–5959. [CrossRef]
23. Desai, A.; Jatakia, J.; Naik, R.; Raul, N. Malicious web content detection using machine leaning. In Proceedings of the 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 19–20 May 2017; pp. 1432–1436. [CrossRef]
24. Machado, L.; Gadge, J. Phishing Sites Detection Based on C4.5 Decision Tree Algorithm. In Proceedings of the 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 17–18 August 2017; pp. 1–5. [CrossRef]
25. Tyagi, I.; Shad, J.; Sharma, S.; Gaur, S.; Kaur, G. A Novel Machine Learning Approach to Detect Phishing Websites. In Proceedings of the 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 22–23 February 2018; pp. 425–430. [CrossRef]
26. Sonmez, Y.; Tuncer, T.; Gokal, H.; Avci, E. Phishing web sites features classification based on extreme learning machine. In Proceedings of the 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, Turkey, 22–25 March 2018; pp. 1–5. [CrossRef]

27. Karabatak, M.; Mustafa, T. Performance comparison of classifiers on reduced phishing website dataset. In Proceedings of the 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, Turkey, 22–25 March 2018; pp. 1–5. [CrossRef]
28. Ginsberg, A.; Yu, C. Rapid Homoglyph Prediction and Detection. In Proceedings of the 2018 1st International Conference on Data Intelligence and Security (ICDIS), South Padre Island, TX, USA, 8–10 April 2018; pp. 17–23. [CrossRef]
29. Parekh, S.; Parikh, D.; Kotak, S.; Sankhe, S. A New Method for Detection of Phishing Websites: URL Detection. In Proceedings of the 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 20–21 April 2018; pp. 949–952. [CrossRef]
30. Chiew, K.L.; Tan, C.L.; Wong, K.; Yong, K.S.; Tiong, W.K. A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. *Inf. Sci.* **2019**, *484*, 153–166. [CrossRef]
31. Sahingoz, O.K.; Buber, E.; Demir, O.; Diri, B. Machine learning based phishing detection from URLs. *Expert Syst. Appl.* **2019**, *117*, 345–357. [CrossRef]
32. Sonowal, G.; Kuppusamy, K. PhiDMA–A phishing detection model with multi-filter approach. *J. King Saud Univ.-Comput. Inf. Sci.* **2020**, *32*, 99–112. [CrossRef]
33. Zhang, W.; Jiang, Q.; Chen, L.; Li, C. Two-stage ELM for phishing Web pages detection using hybrid features. *World Wide Web* **2017**, *20*, 797–813. [CrossRef]
34. Mertooetomo, E.R.; Chen, J. Character recognition with fuzzy features and fuzzy regions. In Proceedings of the 1997 Annual Meeting of the North American Fuzzy Information Processing Society-NAFIPS (Cat. No.97TH8297), Syracuse, NY, USA, 21–24 September 1997; pp. 166–171. [CrossRef]
35. Fu, A.Y.; Deng, X.; Liu, W.; Little, G. The methodology and an application to fight against Unicode attacks. In Proceedings of the Second Symposium on Usable Privacy and Security, Pittsburgh, PA, USA, 12–14 July 2006; Association for Computing Machinery: New York, NY, USA, 2006; Volume 149, pp. 91–101. [CrossRef]
36. Burtescu, E. Database security-attacks and control methods. *J. Appl. Quant. Methods* **2009**, *4*, 449–454.