11-2020

Nereus: A Proposal for Implementing Anti-phishing Software Using Corporate Branding Color Matching

Benjamin Heald
beh8823@rit.edu

Nereus: A Proposal for Implementing Anti-phishing Software Using Corporate Branding Color Matching

by

Benjamin Heald

A Thesis Submitted
in
Partial Fulfillment of the
Requirements for the Degree of
Master of Science
in
Computer Science

Supervised by

Dr. Rajendra Raj

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

November 2020

The thesis "Nereus: A Proposal for Implementing Anti-phishing Software Using Corporate Branding Color Matching" by Benjamin Heald has been examined and approved by the following Examination Committee:

Dr. Rajendra Raj
Professor
Thesis Committee Chair

Dr. Carol Romanowski
Professor

Dr. Reynold Bailey
Professor

Dedication

This thesis is dedicated to my parents Bonnie and David Heald. Without you both I would never have been able to reach this point. From a young age you both fostered in me a love of learning that has brought me here. Your love and support of me and my academic goals has been a constant in my life, and I thank you both from the bottom of my heart.

Acknowledgments

I am incredibly grateful to all the professors from both Rochester Institute of Technology, Gettysburg College, and Hebrew University of Jerusalem who have worked with me on this project over the past five years. To Dr. Rajendra Raj and Dr. Carol Romanowski of RIT who provided me with a multitude of opportunities to work and learn this past year. Professor Rob Olson, who taught me a great deal about the security industry from both technical and professional perspectives. I also give my deepest thanks to Dr. Clif Presser, Dr. Darren Glass, and Dr. Ivaylo Ilinkin of Gettysburg College who helped to hone and develop this idea from its conception. To Dr. Rod Tosten who encouraged me to pursue my Master's degree at RIT, I would not have reached this point without your advice. The encouragement and support from others like my friend Marc Crouse were also essential in developing my professional and academic skills. To Naufa Amirani, who fielded my constant questions and whose support was invaluable. Lastly, I would like to thank my parents, brothers, and friends who had to put up with me bouncing ideas off them about phishing for the last five years.

Abstract

Nereus: A Proposal for Implementing Anti-phishing Software Using Corporate Branding Color Matching

Benjamin Heald

Supervising Professor: Dr. Rajendra Raj

Over the years, many anti-phishing software packages have been developed that can reliably and accurately detect and delete phishing emails as they are received. As communication on the internet evolves however, these existing anti-phishing systems are becoming less effective. As more users migrate away from email and into emerging technologies such as Slack, Zoom, and Microsoft Teams, new effective anti-phishing filters must be created for each new communication platform. Developers are therefore fighting an uphill battle to keep users safe. An anti-phishing mechanism that positions itself instead directly between the user and the websites they visit is therefore proposed. This positioning allows the system to protect the user against phishing attacks no matter the communication medium. Existing research in this area suffers from impractical processing overhead, secure logic failures, and unreliability in the long term. This thesis overcomes these issues by using corporate branding color as a visual similarity measurement within a supervised learning algorithm to perform phishing identification. Since it has been shown that corporate branding colors change much less often than other design choices like HTML layout, this visual similarity comparison is able to maintain high accuracy over long periods of

time. This principle, combined with a fast machine learning algorithm, allows the application to be accurate, effective, and adaptable with little to no added overhead, overcoming the shortcomings in currently proposed solutions.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The majority of anti-phishing systems position themselves between the user and their communication platform. From this position, the software can easily intercept and flag incoming messages that are deemed potential phishing attacks before the user ever sees them. Since email is the main online communication platform for both businesses and individuals, most implementations of this approach are created only for use by email service providers. This intercept-and-filter model has been proven to be extremely effective at preventing phishing emails from reaching users, and has been widely implemented in major email systems for many years. Major email service providers such as G-Mail have developed extremely sophisticated systems in which around 100 million phishing messages are intercepted and filtered each day [2]. Over time this approach has begun to be ineffective as the number of available communication platforms skyrockets.

With the advent of applications such as WhatsApp, Zoom, Facebook Messenger, Slack, Microsoft Teams, and other instant messaging applications, more and more communication on the internet is conducted away from email. These new communication channels have existed for years, yet little work has been done to implement the same kind of intercept-and-filter mechanisms used by email on these platforms. With the fickle nature of consumers and the rapid adoption of new communication platforms, it is presumed a losing battle to try and implement effective anti-phishing filters in every new platform. Software that attempts to prevent phishing attacks once the user has visited a suspicious website is therefore theoretically more adaptable to this environment. These systems attempt to prevent the user

from entering sensitive data after they have already been tricked into visiting a URL leading to a phishing site. This identify-and-warn model enables the protection of users against phishing attacks even if the intercept-and-filter protection of the communication platform fails. While this protection is placed much closer to the attacker's end goal, it would be better suited to the rapidly changing landscape of the internet.

A piece of anti-phishing software that positions itself between the user and a phishing website needs adhere to the following design principles:

1. Accurate. The application should be able to accurately and reliably identify if the website visited is a phishing attack.

2. Reliable. The application should automatically adapt to new phishing attacks and be effective without requiring major updates from the software maintainers.

3. Fast. The application must ensure that its operation does not impact the speed of the user's internet browsing.

4. Secure. The application should be able to grant the user a reasonable degree of anonymity and protect their privacy.

Achieving these goals can be extremely difficult however. The transiency of webpages, constantly improving quality of attacks, and speed at which it takes to make an accurate decision all factor into this difficulty. Generic warnings and false positives also threaten to desensitize users, making it unlikely that they will heed the warnings presented by the application. Though many systems have been proposed and are currently in use to prevent users from interacting with potentially dangerous websites, it is believed that none completely match the criteria listed above.

1.1 Background

Phishing is defined as a criminal activity combining social engineering and technology to access private information without consent [5]. Phishing is one of the most common and

easily performed cyber-attacks, costing the world more than $450 billion per year or nearly 90% of the total estimated cost of cyber-crime. [15, 17]. Popular tools such as the ”Social Engineering Toolkit” make it trivial for cyber-criminals to automate the process of cloning websites and creating custom URLs for their victims to visit [20]. These URLS are then sent either to a great number of users via traditional phishing attack or to a specific person within an organization under spear phishing. Some form of text will usually accompany the URL in order to persuade the victim into clicking on the link and inputting personal data on the following web page. Any data entered by the victim on the phishing site is then usually sent directly to the attacker. The overall flow of a generic phishing attack is given below in figure 1.1.

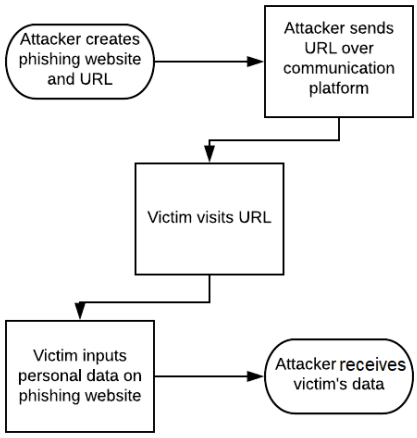

Figure 1.1: Generic Web Phishing Attack Flow

In the vast majority of web-based phishing attacks, the website will need to be able to persuade the victim that it is the actual website. As mentioned, the attacker can do this within the URL or HTML text, but the most effective method is overall visual similarity. [24, 6] It has even been found that most users will ignore small details of the site they visit, such as the URL, and focus entirely on the ”look” of the site to determine if it is legitimate. [24] By using visual similarity to the legitimate website, the attacker is able to convince

most web users that the site is legitamate. By using the tools mentioned earlier, the attacker can easily create a site that is remarkably visually similar to the target site. In figure 1.2, we observe a screenshot of the legitimate site 'Paypal.com'. In figure 1.3 we see the screenshot of the phishing site 'coaltur.com'. As can be seen, this phishing site is able to directly copy the visual layout of the target with only slight textual differences.

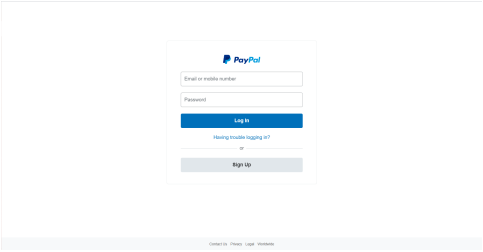

Figure 1.2: Screenshot of Paypal.com

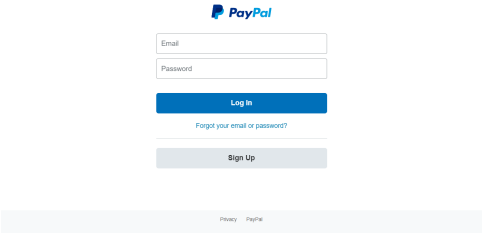

Figure 1.3: Screenshot of coaltur.com

As was researched in [3], users will most identify a corporate brand by their use of color. Color therefore plays a key role in a user's visual assessment of a website. Therefore, it is likely that an attacker will at least have to copy the website's branding colors in order to successfully impersonate the site.

1.2 Related Work

There are a number of current in-browser software packages that attempt to provide in-browser protection against web phishing attacks. All of these packages work to identify if a given website represents a phishing attack in real-time as the user visits the website. Working in real-time gives the application an extremely small window to operate in, as a user of a website expects it to be available for interaction immediately following page load.

Of all the current efforts to implement a effective system in this domain, Google Chrome's "Real-time phishing protection" [13] currently presents the best solution. In this model, when the user visits a new site, the URL is checked against a Google-generated blacklist of known phishing URLS that is updated regularly. If the URL is not present in the blacklist, the application then performs a secondary check wherein the URL is sent to Google to determine if it has the qualities of a phishing URL. If either of these two checks detect a phishing attack, a full-page banner is presented warning the user. While this approach to ant-phishing software achieves both accuracy and speed, it is doubtful that it is either adaptable or effective at ensuring users heed its warnings. Since it depends heavily on the presence of the URL on a blacklist, it would not be useful against an attacker who can quickly generate new domains for their attacks. This blacklist approach would also not be effective at preventing spear phishing attacks, wherein the URL is only sent to one person, as their blacklist is populated using mainly crowd-sourced data. While their phishing detection using URL characteristics is likely accurate, it is unlikely to remain so as attackers constantly adapt and learn what such algorithms identify as phishing attacks. Its effectiveness at keeping users from interacting with the potential phishing site is also in doubt, as the warning it presents is extremely generic. Such warnings are often ignored by users if the perceived reward of visiting the site outweighs the described risks.

The software package created by Mohammad et al. [18] proposes a machine learning approach to the task of identifying phishing attacks in real time. This approach entails learning a model that can accurately predict if a given URL represents a phishing site. This learning is done using certain qualities of the URL as features in a supervised learning

algorithm. While this attempt is adaptable to new attacks, it's effectiveness at ensuring users do not interact with the site is in doubt as it uses very poor warnings. This application additionally does not achieve the level of accuracy desired, as it learns solely based upon qualities of the URL. As with Google's model, this will likely not remain accurate in the long run as attackers adapt and modify their URL structure to avoid detection.

Zhang et al. propose [25] a groundbreaking theoretical framework CANTINA for automatically detecting phishing attacks. The authors begin by describing their TF-IDF detection structure [25], wherein their tool analyzes the top five most important terms from the given webpage. These top five terms are then inputted into a search engine, and the rank of the domain name is then judged. If the web domain appears within the top N results, then the page is judged to be a legitimate site. This search engine ranking algorithm was found to have a high false positive rate however, so Zhang et al. proposes a list of heuristics to assist their algorithm. These heuristics include the age of the domain, with the idea that a more recently registered domain name is more likely to be used in a phishing attack. Use of a well-known brand image is another flag for this algorithm. Suspicious URL characters such as the "@" and "-" symbols are also flagged. The number of subdomains, detection of an IP address, and the presence of HTML forms are also present. The implementation of this algorithm was able to achieve 97% accuracy at identifying a test set of domains. An analysis of the most effective heuristics showed that the age of the domain was the most important feature, followed by the presence of forms, IP address, and the number of subdomains. Though CANTINA is a famous and influential framework that has influenced almost all current anti-phishing efforts, there are several key problems that have emerged in the last thirteen years. The main and most effective feature of CANTINA is determined by Zhang et al. to be their TF-IDF domain ranking feature. While this was likely effective at the time, advances in the ability for an attacker to manipulate their search engine ranking means that this can be gamed. In order to game the system, an attacker could theoretically place non-visible text on their HTML page in order to manipulate the TF-IDF feature into inputting a desired word set into the search engine. Other features of this framework, like

the age of domain feature are also suspect. At the time, major hosting applications that allow users to quickly create a subdomain host on trusted domains such as herokuapp.com, appspot.com, and 000webhost.com did not exist. As they are extremely widespread among phishing attempts today, the age of domain heuristic would likely be less effective. In summary, while CANTINA is an influential framework that continues to affect anti-phishing tools, several of the features proposed would no longer be effective in the current day.

From these examples, it is clear that a more comprehensive software package was needed. An anti-phishing application that is adaptable to new techniques and attacks will need to implement a machine learning algorithm similar to the one proposed by Mohammad et al. [18]. To overcome the limitations of that approach however, the speed and accuracy of Google's method [13] will also need to be implemented. A real-time detection method that does not depend solely on qualities of the URL or a blacklist is also needed to overcome the limitations of Google's approach. In addition to these requirements, neither of these current solutions were judged to be effective at ensuring that the user heeds the presented warnings. A solution which optimizes speed, accuracy, and adaptability, while at the same time able to present a detailed and highly engaging warning is therefore required, and was developed for this thesis.

1.3 Hypothesis

It is clear from the review of current anti-phishing software packages that a new solution is needed due to the over-reliance on out-of-date security principles and high system overhead that make them impractical. A fast and accurate machine learning algorithm that learns not only from features of the URL, as in [18], but from branding color similarity is therefore developed here.

The key hypothesis of this thesis is as follows: ***As branding colors shown on any given web page will change with much less frequency than HTML layout or design, these colors can be used as a reliable visual similarity measure***. This measure of visual similarity, when used in conjunction with URL heuristic measurements, will create a model that

can identify phishing attacks with both accuracy and long-term reliability. The resulting system will therefore also be able to meet all the criteria set for a successful client-side anti-phishing software as outlined in section 1.

This hypothesis relies on the fact that any phishing site will need to accurately copy the corporate branding colors in order to trick the victim into believing their phish is the legitimate site. Therefore, by measuring the distance of the web page's color scheme from popularly phished brands, the algorithm is able to identify phishing sites accurately. This algorithm is also able to accurately identify phishing websites without overly burdening the user with processing time. This hypothesis allows for speed as no image processing is done other than the extraction of the color palette from a screenshot of the website. All of the features being inputted into the algorithm are therefore numeric and binary, which when inputted into the decision tree model produces extremely fast predictions.

Color is acknowledged as a critical element of the corporate identity, as can be observed in such cases as IBM (otherwise known as "The Big Blue"). It is highly influential given that it is perceived more quickly than symbols or text, and is memorable of the brand. Therefore it is reasonable to say that overhauls of a company's branding color is not common, especially those that are global, though they may often change their HTML, logo, or general design.[3] If a phishing website wishes to accurately impersonate a corporate brand, they will need to directly copy that brand's color scheme. If the branding color is not directly copied, it is theorized that the potential victim is much more likely to notice that the website is a phishing attempt.

The semi-permanent nature of branding colors provides a supervised learning algorithm with the opportunity to more accurately identify both the presence of a phishing website and the exact corporate identity the website is trying to mimic. Measuring how "close" a given website's color scheme is to a list of known corporate branding colors proved to be an extremely powerful feature in the supervised learning algorithm. Working in tandem with URL features, this produced a model that is able to identify a phishing website with high accuracy.

By implementing corporate color matching as a feature in a supervised machine learning model, Nereus achieves all four of the design principles described in section one.

1. Accurate. Nereus uses color matching along with URL features in order to accurately identify phishing attacks.

2. Reliable. Nereus is able to automatically adapt to new phishing attacks in the long term through constant re-learning of the model as well as depending on the semi-permanent nature of branding colors.

3. Fast. Nereus takes in an extremely small amount of data from the website on page load and transmit this to a server where the model will determine the phishing status. This distribution of machine duties will allow the system to operate with little noticeable overhead.

4. Secure. The ability of Nereus to identify a phishing site using only numeric and binary features enables it to be not only effective but also able to keep its users safe.

1.4 Road map

This thesis will give an overview of the design, results analysis, and conclusions that were made on the anti-phishing software Nereus. To begin, section two will give a detailed explanation on the data gathering, feature selection, data processing, data analysis, and model design for the phishing classifier. The feature discussion specifically will outline how corporate branding color matching was utilized as a feature within this thesis. Section two will also deal with the supporting components of the application, the Chrome extension GUI and the remote server architecture. A examination of how all these pieces fit together is given to finish section two. Section Three will then give an overview of the results that were gathered during the experiment, as well as providing analysis. The analysis is split into the four metrics, accuracy, speed, reliability, and security that were discussed earlier.

These metrics are considered in this thesis the qualifications for a successful application. Finally conclusions will be made from the analysis and future work will be discussed.

Chapter 2

Design & Implementation

Named for the ancient Greek god of fishermen, the design of Nereus can be subdivided into three smaller pieces of technology. First, some kind of program will need to be used to determine, based on some data or feature of the website visited, if the currently visited website is a phishing attack. Secondly, since Nereus is a client-side application, another program will need to be used to extract the data from the website on the client side and transmit it to the classifier. Finally, the classifier will need to exist on some kind of remote server that the client-side application interacts with. Finally, an overview of the complete architecture will be given. In this section, we will give an overview of the design choices made for each of these parts of the Nereus application.

2.1 Supervised Learning Classifier

This section will give an overview of the data, features, and model type chosen for the phishing classifier.

2.1.1 Data Acquisition

In order to produce an accurate classification algorithm, we first needed to find a reliable way to retrieve phishing websites that were live on the internet. Since phishing attacks are notoriously short-lived [4, 11], finding live attacks from which to draw data from can be difficult. This application, like many others, used PhishTank [21] as the ultimate source of its phishing data. PhishTank is a free to use, human-curated list of phishing attacks

currently live on the internet. The authors of PhishTank offer a convienent API which provides data on phishing attacks from the past 72 hours. This data was used as it is logical that the most recent data would be the most likely to still be online.

For legitimate web site data, several problems presented themselves. The naive approach would be to simply retrieve a list of the top most visited websites and utilize that as the data. This approach is flawed however, as it does not represent how actual web traffic is structured. As an example, such a ranked list might include popular domains such as www.google.com, www.twitter.com, etc. While these are popular websites likely to be visited by the average internet user, the URL is not. A user is much more likely to visit a URL with a filepath such as www.twitter.com/user/some_account. This discrepancy means that a ranked list cannot be used to represent actual internet traffic. To overcome this, internet traffic history over a period of three months from the author and an anonymous volunteer was used. While from a small sample size, this data is much more realistic than a simple ranked list. This kind of data has also been successfully used in other research in the past [26]. To overcome the limitations and bias of the small sample size, 500 of the most popular websites were also added to the data. This data made up 17.28% of the total size of the legitimate phishing data entries, which was 2,893.

Once the data was gathered, the dataset was made up of 5,661 entries. Of these, 2,893 were legitimate sites and 2,768 were phishing sites. This puts the proportion of non-phishing vs phishing data at 48.89% and 51.11% respectively.

2.1.2 Feature Discussion

Once the data was gathered, an analysis of what features to include was performed. As has been discussed, the data features used needed to be both small enough to be gathered quickly, and anonymous enough so that its collection by the tool would not be considered a privacy violation. This problem of anonymous data was particularly important, as the goal of this tool was to use visual similarity between known branding colors as a strong feature within the classifier. In the end multiple effective features were used, including

seven binary and integer features and ten visual similarity measurements. Of these, it was found that the visual similarity features provided this application with the required speed, accuracy, and reliability

URL & HTML Features

In the majority of previous anti-phishing applications, the features of the classifiers have been drawn from attributes of the URL or HTML of the page [4, 7, 10, 14, 16, 19, 24, 25, 26] . These existing systems do have some limitations and several, especially those presented by Zhang et al [25], likely do not apply to current web traffic. Several features were eventually used however, that were found to be both relevant and effective. The number of subdomains and URL length were the only two integer features extracted from the data. These have been used consistently in past research and shown to be extremely effective [4, 25]. Binary features including the presence of an IP address, the use of non-English Unicode characters, the presence of HTML form entities, the usage of a valid SSL certificate, and the usage of a top-level domain were also found in the literature and found to be extremely relevant [4, 19, 25, 26]. Table 2.1 provides a full list of these features, along with a description of each.

Visual Similarity Features

The main features of this classifier will be color branding distance measurements. This is useful as a feature as it has been shown that in order to successfully imitate a given site, the phishing attack most be visually similar. In the past, such visual similarity has been measured with HTML block analysis, earth mover distance, and more [8, 12, 16, 24]. In this thesis however, a novel feature is proposed. It is shown that most users correlate corporate branding colors with identity, for example ”Facebook Blue” or ”Netflix Red” [3]. A successful phishing attack therefore will need to feature these branding colors prominently. By measuring the distance of the colors on the potential phishing site to a dataset of corporate branding colors, it can be seen if the suspicious site is utilizing known corporate branding

Table 2.1: Nereus Data Feature Description

Data Feature Name	Feature Description
Num_Subdomains	Integer attribute signifying the number of subdomains in the URL. It has been found that phishing sites will have a multitude of subdomains to confuse the user.
URL_Length	Integer attribute representing the character length of the URL. It has been seen that phishing URLs are typically longer to confuse the user.
IP_Address	Binary attribute symbolizing if there is an IP address in the URL.
Has_Symbols	Binary attribute symbolizing if there exist non-English characters in the URL. Non-English characters are at times used in Phishing attacks to mimic English characters.
Has_Form	Binary attribute symbolizing if there are any HTML FORM tags present in the HTML code of the site. Since the goal of a phishing site is to steal data, the attacker will likely include an HTML FORM tag for the victim to input their data.
Is_HTTPS	Binary attribute symbolizing if the URL is using the HTTPS SSL scheme. It should be noted that with the proliferation of self-signed certificates, the presence of SSL on a site is becoming less and less relevant.
Top_Level_Domain	Binary attribute symbolizing if the domain extension is not a standard such as .com, .org, .edu, etc.

colors to impersonate a website. This measurement will provide a visual similarity metric that is both highly accurate and reliable, as corporate branding colors have been shown to change little over time due to high user identification [3].

In practical terms, the color similarity feature will be the color on the target website that most closely matches the color of known corporate branding. This will be represented by the euclidean distance between the RGB color palette of the target website and the RGB values of the top 10 most phished websites on the internet. In order to determine these distances, the algorithm first determines the most used colors being used by the website. This is done with a simple binning algorithm, with the top six most commonly used RGB values being stored as the color palette of the website. This color palette is then compared to known corporate branding colors through the euclidean distance of the RGB values. Each value in the color palette is compared to each branding color, and the value with the minimum distance is then stored as the feature. This algorithm is represented in Algorithm 1.

Algorithm 1: Corporate branding color distance measurement

```
Result: The minimum distance between the color palette and the branding colors.
min_distance = infinity;
for color in website_color_palette do
    for branding_color in known_branding_colors do
        if EUCLIDEAN_DISTANCE(color, branding_color) ≤ min_distance then
            min_distance = EUCLIDEAN_DISTANCE(color, branding_color);
        end
    end
end
```

At the conclusion of this feature extraction algorithm, ten features are produced, each representing the distance of each branding color to some color that was found on the web page. This feature can also potentially be used in identification of the brand that the phishing site is trying to imitate, as it is logically the feature with the lowest distance. The branding colors used can be found in table 2.2.

Table 2.2: Nereus Visual Feature Description

Data Feature Name	Feature Description
Distance_From_Twitter	Float value representing the minimum Euclidean distance between some color on the target webpage and the known branding colors of Twitter.
Distance_From_Facebook	Float value representing the minimum Euclidean distance between some color on the target webpage and the known branding colors of Facebook.
Distance_From_Google	Float value representing the minimum Euclidean distance between some color on the target webpage and the known branding colors of Google.
Distance_From_Paypal	Float value representing the minimum Euclidean distance between some color on the target webpage and the known branding colors of Paypal.
Distance_From_Adobe	Float value representing the minimum Euclidean distance between some color on the target webpage and the known branding colors of Adobe.
Distance_From_Apple	Float value representing the minimum Euclidean distance between some color on the target webpage and the known branding colors of Apple.
Distance_From_Amazon	Float value representing the minimum Euclidean distance between some color on the target webpage and the known branding colors of Amazon.
Distance_From_Orange	Float value representing the minimum Euclidean distance between some color on the target webpage and the known branding colors of Orange.
Distance_From_WhatsApp	Float value representing the minimum Euclidean distance between some color on the target webpage and the known branding colors of Whatsapp.
Distance_From_Netflix	Float value representing the minimum Euclidean distance between some color on the target webpage and the known branding colors of Netflix.

Table 2.3: Nereus Data Example

Data Feature Name	Data Value
Num_Subdomains	1
URL_Length	24
IP_Address	0
Has_Symbols	0
Has_Form	0
Is_HTTPS	1
Top_Level_Domain	1
Distance_From_Twitter	48.062
Distance_From_Facebook	65.931
Distance_From_Google	56.648
Distance_From_Paypal	199.183
Distance_From_Adobe	119.419
Distance_From_Apple	112.717
Distance_From_Amazon	42.743
Distance_From_Orange	218.643
Distance_From_WhatsApp	27.749
Distance_From_Netflix	92.109

This novel visual similarity measurement between the target website and the most phished websites was found to be an extremely strong feature that not only adds accuracy to the classifier, but also makes it much more accurate over longer periods of time.

Table 2.3 displays a single example of a vector in the dataset. In total, the dataset was made up of 5,661 entries. Of these, 2,893 were legitimate sites and 2,768 were phishing sites.

2.1.3 Data Processing

In order to extract the usable features from the raw collected data, several things needed to be done. First, the raw URL data for both phishing and non-phishing would need to be used to gather screenshots of the live website they represent as well as the HTML code of the site. This data was needed for the visual similarity features and HTML features as described in the previous section.

Collecting the HTML & website screenshots

First is to collect the URL and HTML features from both the phishing and non-phishing data. The raw data collected consisted only of URLs, and therefore it would be impossible to extract the HTML features without further processing. In addition, a screenshot of the webpage would need to be taken in order to extract an effective color palette for the visual similarity measurements. In order to accomplish this, a small script was written that visited each URL in turn, saved a copy of its HTML code, took a screenshot of the final loaded page, and then saved all this data within a CSV file. This script utilized the Python package Selenium to simulate a typical web browser environment [23].

Curating the raw data

Once the data was collected, a curation process was needed in order to filter out bad image data from the collection. In some of the captured images, the Selenium tool was unable to take an effective screenshot of the page, resulting in a blank white image, as can be seen in figure 2.1. Since this kind of data would likely skew the data of the visual similarity features, a curation tool was developed to filter them out. By using a simple python script, each of the 4,500 website screenshots were examined and determined to contain a good screenshot, as exemplified in figure 2.2.

Feature Extraction

In order to extract the features discussed in section 2.1.2, a simple python script was again created. For the URL and HTML binary features, the script simply looked for the presence of certain characteristics in the HTML or URL. For the visual similarity features, the program utilized Algorithm 1. The program then compiled these measurements into vectors and stored them within a CSV file. At the end of the data processing stage, over 5,500 pieces of data were collected, with 2,768 non-phishing sites and 2,893 phishing sites represented.

Figure 2.1: Bad Image Data

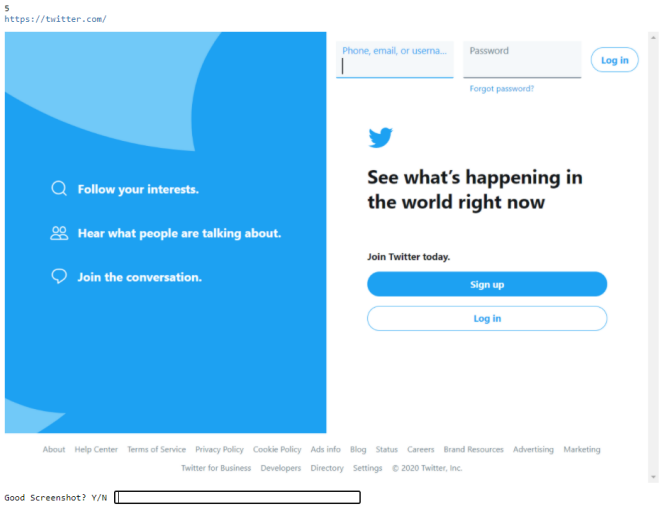

Figure 2.2: Good Image Data

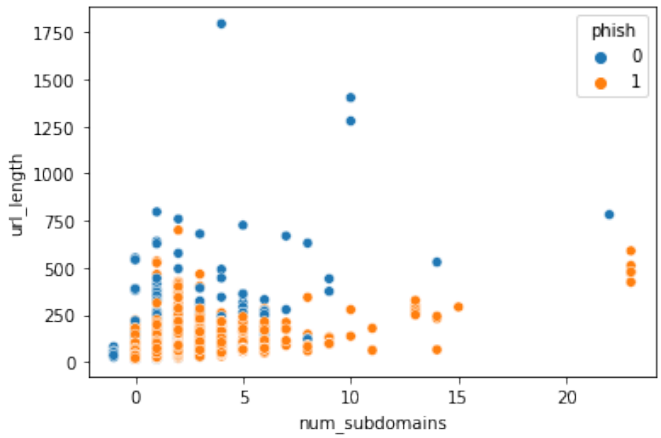

Figure 2.3: URL Length vs. Number of Subdomains

2.1.4 Data Analysis

Once all data was collected, some basic analysis was done to better understand it and how it could be used in the phishing model. In total, 5,661 vectors of data were gathered. Of these, 2,768 represented non-phishing sites and 2,893 represented phishing sites. While this dataset is not as large as some, it is significantly larger than those used by other researchers [19, 26]. In order to train the most accurate model possible, the data was gathered in two separate sets three months apart. This was done to avoid bias in both the phishing and non-phishing data. It should also be noticed that in the following data analysis, a ”phish” value of ”0” indicates a non-phishing site, while a value of ”1” indicates a phishing site.

Interestingly enough, there were several measurements that were surprising to see within the data. When comparing the number of subdomains and the length of the URL, it was surprising to learn that contrary to what was theorized, non-phishing data seemed to be correlated to larger values, as seen in figure 2.3. This is interesting, as previous research has found the opposite to be true [25, 4].

The reasoning for this is likely due to to either changing patterns in web traffic in the years since the previous research was published, or due to bias within the data. It is likely that the answer is the former, as the data collected was gathered in two separate batches

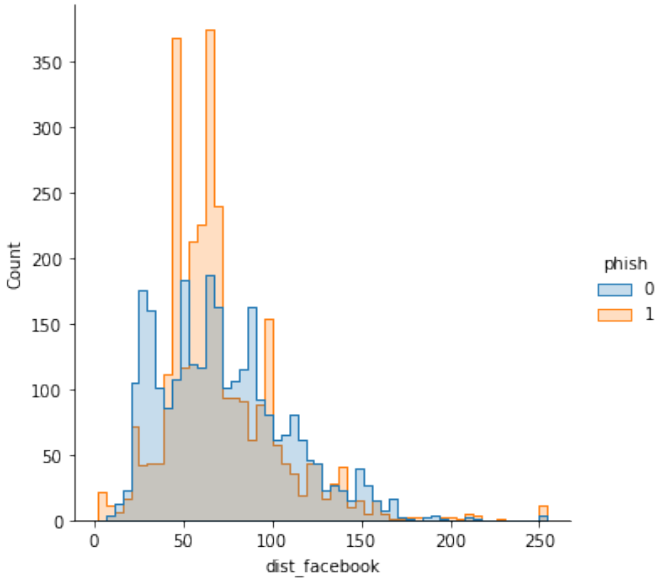

Figure 2.4: Distribution of the distance between the website's color palette and Facebook's branding colors.

three months apart specifically to avoid any bias. Since the theories set forth by Zhang et al. in CANTINA were made over ten years ago, it is likely therefore that internet traffic patterns have changed.

Furthering the data analysis, we were able to see data that greatly reinforced the hypothesis of the thesis. As can be seen in figures 2.4 and 2.5, phishing websites were seen to correlate to shorter distances between the branding colors of both Facebook and Microsoft. In these figures, the blue represents non-phishing sites, orange represents phishing sites, and grey represents overlap between the two. It can be assumed that the spikes we see in the histograms indicate a large amount of phishing attacks against these targets .Since these are two of the most popular targets for phishing attacks, it makes logical sense that these spikes occur. Interestingly, we see in both the Facebook and Microsoft histograms that a spike of non-phishing data occurs at a distance of around 35 - 45. This spike is theorized to occur as the actual sites are being visited often within the data, and would therefore present close color matches for non-phishing data.

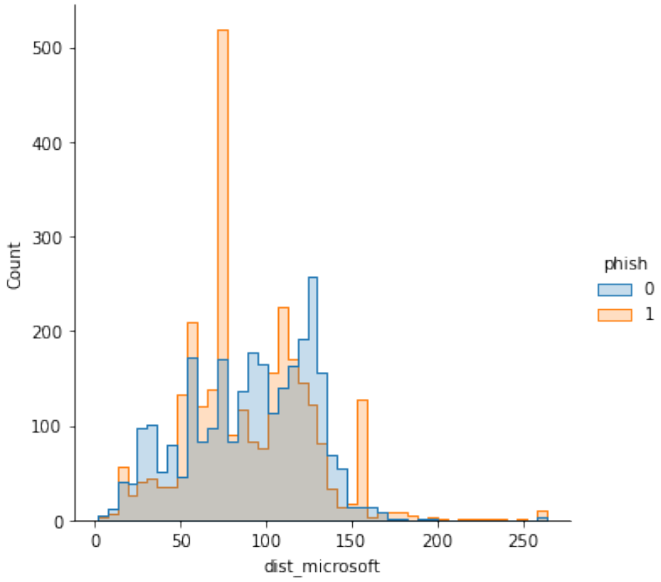

Figure 2.5: Distribution of the distance between the website's color palette and Microsoft's branding colors.

In several pairwise plots, we were able to see extremely tight positive correlation between the distance measurements of such branding colors as Amazon and Orange, and Adobe and Netflix. This correlation can be seen in figures 2.6 and 2.7. It is theorized that this tight correlation between the color distances is due to the branding colors being very similar. This is demonstrated by comparing the RGB value of Adobe [”r”:255,”g”:0,”b”:0] to that of Netflix, [”r”:229,”g”:9,”b”:20]. As can be seen, these branding colors are both dominated by the color red, which is likely why the two distance measurements are so correlated. Due to this correlation, is it unlikely that the distance measurement could reliably be used to identify between them.

2.1.5 Model Design

After thorough investigation, it is clear that there are two logical options for the design of the classification algorithm. The first, as was used in several previous approaches to this issue, was a heuristic based algorithm. This kind of algorithm has been used in several

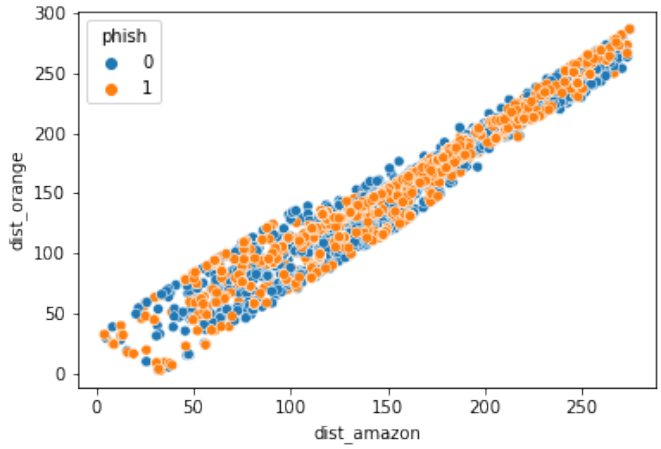

Figure 2.6: Distance between Amazon branding colors and Orange branding colors.

anti-phishing applications over the years, notably including CANTINA, LinkGuard, and the target identification system proposed by Khonji et al. [25, 7, 14]. These kind of heuristic algorithms can produce reliable classification relatively quickly. Unfortunately, these systems are also extremely vulnerable to gaming by the attacker. In both CANTINA and LinkGuard, the heuristic algorithms depend solely on attributes of the URL and HTML pages. Since these applications are both open-source, a determined attacker could simply modify their URL pattern or HTML attributes to avoid a phishing classification. While this kind of gaming might not exist for a robust heuristic algorithm, an increase in complexity would undoubtedly lead to greater program size and running time, decreasing the application's ability to be used in a practical setting. These drawbacks led to the conclusion that a supervised machine learning classifier should be used.

In contrast to a heuristic classifier, a machine learning model would be more opaque in design and function, not to mention more adaptable over time. This was shown in multiple other anti-phishing tools developed over the years [19, 26]. A key problem with many types of classifier models however is that they often take much longer than heuristic classifiers to make a prediction. Care was taken therefore to discover which classifier would be able to maximize both running speed and practical accuracy. After careful research, three logical

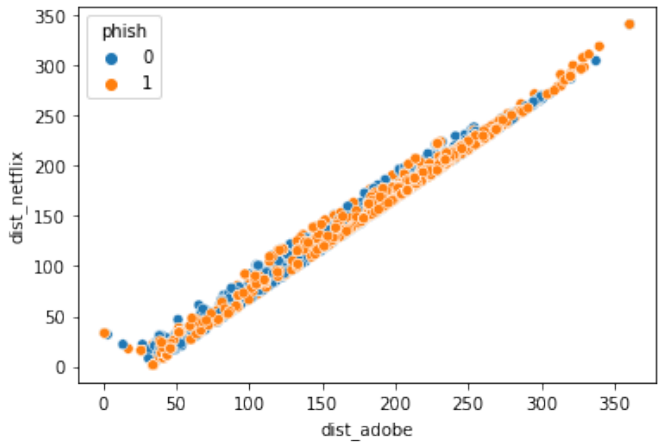

Figure 2.7: Distance between Adobe branding colors and Netflix branding colors.

candidates presented themselves; Support Vector Machine, Logistic Regression, and Random Forest Decision Tree. While the final model was eventually chosen to be a Random Forest, a background of each of these will be given with explanations why each candidate was chosen.

Support Vector Machine

A Support Vector Machine is a popular supervised machine learning model. A SVM works by creating a separation between the data points of two or more classes. Usually this seperation is linear, but other kernels can be used as well. This separation is then used to classify new data, with the identification being made based upon which side of the separation line the data falls upon [22]. The SVM model was chosen here as most of the features of the data, such as the branding color distance were found to be fairly separable, as can be seen in section 2.1.4. This is also true for the number of sub-domains and the length of the URL features. It was also hoped that this model would produce fast and reliable results for the classifier. Ultimately however this model was not used, as it was found that it was not accurate at classifying the data. This is likely due to the fact that while the data for many of the features is numeric, many are binary. An SVM model, as with regression models,

depends on a wide range of values in order to determine where the separations between the classes are. Since the majority of the data features are binary in this dataset, an SVM model cannot achieve high accuracy. The spikes of phishing vs. non-phishing data seen at the 30-45 mark could also cause the classification model to incorrectly predict one class or another due to poor separation.

Logistic Regression

A logistic regression model is similar to a linear regression model, in that regression is used to produce a prediction between two binary classes. In opposition to linear regression however, logistic regression utilizes a sigmoid function to produce a probability function [9]. Logistical regression also differs from linear regression in how it calculates the regression coefficients. In logistic regression, Maximum Likelihood Estimation is usually used to calculate parameter values. All of these changes aim to make logistical regression more accurate at predicting data that either is not very linear or has many outliers. As was found in the SVM model however, the predictions produced were not highly accurate. While the SVM is likely to produce good predictions for the numeric values such as URL length or the color distance features, this data is supported by the binary features. Binary data by design cannot be well used in a regression algorithm.

Random Forest Decision Tree

A Random Forest decision tree was the last machine learning model examined, and proved to be the most accurate. In this type of model, a ”decision tree” is constructed in order to correctly classify the data. At the beginning of each ”branch” of the tree, the data will be split by some measurement of a feature. For the binary features this would mean the data would be split between the true and false values. For the integer and float values, the split would occur between some set numeric interval. These splits are optimized by using either entropy or information gain. The leaf nodes are usually determined once all data in the branch is homogeneously one class or another. Once the tree is constructed, new data is

fed into the tree. Once the data arrives in a leaf node with a class label, the decision tree classifies the data accordingly. In opposition to a classic decision tree however, a Random Forest model uses an ensemble learning procedure to produce a more accurate result [1]. The Random Forest model creates a set number of decision trees with random starting features, finds their classification, and makes a prediction based on the classification of the majority of the trees. This kind of ensemble learning avoids the bias of classic decision tree models.

Random Forest was chosen in the end for this application as it produces accurate results while also maintaining high speeds. Once the model is created, new data inputted into the algorithm can be tested quickly, especially as most of the features in this data are either numeric or binary. The ensemble learning aspect also helps to avoid the over-fitting common in traditional decision tree models [1].

2.2 Chrome Extension

The secondary piece of this thesis will be a Google Chrome browser extension. This piece of software will act as the main GUI for the application. The user will install this extension in order to use the system. The extension will position itself between the user and every website they visit, collect the necessary data on the URL and color branding, and pass it along to the trained machine learning model that will exist on a remote server. The features will be extracted in much the same way as described in section 2.1.3. The Chrome extension will take a screenshot of the website currently visited and use Algorithm 1 to calculate the color distance feature values. This screenshot is not stored in any way, and is recycled from memory once the extension has extracted the color features. The Chrome extension is written in JavaScript and will use an XHR request to transmit the feature vector to the remote server. Great pains were taken to ensure that the data retrieved by the extension is accurate. To avoid a ”bad” screenshot, as shown in figure 2.1, the extension attempts to wait until all HTML and JavaScript on the page has completely finished loading before

capturing the page. This was done particularly to correctly capture those websites that utilize NODE JS technologies to render their page, as they load the bulk of their site after the HTML has loaded. If the extension did not wait for the JavaScript to finish executing, the page would not be correctly rendered during the capture.

2.3 Remote Server

A key part of the effectiveness of this application is the running time of the application. As was explored earlier, an anti-phishing application that interrupts client-side load will likely not be used long by a user, as it interrupts their normal browsing patterns. Therefore the remote server that houses the trained Random Forest model will need to maximize its response time. In order to determine the kind of remote server architecture that would be most effective, two candidates presented themselves. The first, an Apache server that runs PHP, and the second, a Simple HTTP Python server. This second was proven to be the most effective, as will be shown.

2.3.1 Apache & PHP Server

In the first trial, a remote server running the framework Apache was used, and the programming was done using PHP. This PHP program accepted the feature vector from the Google Chrome extension, and used the ”EXEC()” native function to execute a python script that would produce the classification. After trials, this method took around 1.4s to respond to the request. In practical terms, this was much too great a delay to be used. This delay is likely to be caused by the separation of technologies. The data was received by a PHP script, which would then execute a local Python script to classify the data. This separation between the receiving technology and the prediction technology created a unavoidable transfer delay. To avoid this, a native Python server was next tested.

2.3.2 SimpleHTTP Python Server

In the second and more successful trial, a Python server framework called ”SimpleHTTP” was used. The response time of this server was much improved from the PHP server, with an average response time of 0.6s. This response time was found to be acceptable for the application, as it did not create a noticeable delay in the responsiveness of browsing. This server is better than the PHP server mainly as the model was able to execute within the same program that initially received the feature vector.

2.4 Overarching System Architecture

This design and implementation section has attempted to give a thorough review of all aspects of the completed Nereus anti-phishing application. Due to the size and complexity of the design, an simple overview of how the individual pieces discussed above will now be given. In figure 2.8 we can see that the system architecture of Nereus consists of two major components. The typical workflow of the application is therefore traced here. When visiting a website for the first time, the Google Chrome extension discussed in section 2.2 begins by extracting all the features from section 2.1.2 into a feature vector. This vector of data is then transmitted using JavaScript to the remote server from section 2.3. This remote server accepts this data vector using a SimpleHTTP Python, and inputs it into the trained Random Forest model, which also runs in Python. This model will then return a prediction of either phishing or non-phishing to the Chrome extension. The Chrome extension will then present a warning to the user or allow them to continue browsing, based upon whatever prediction the classifier has made.

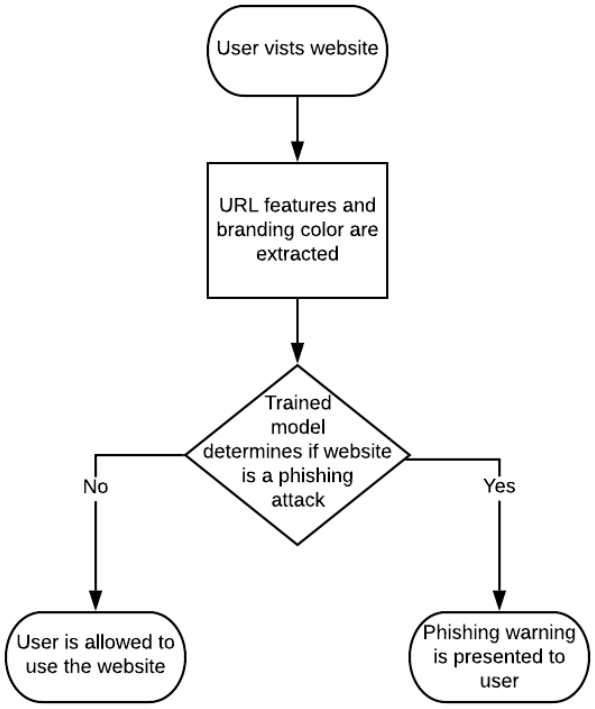

Figure 2.8: Nereus System Architecture

Chapter 3

Analysis

The purpose of this thesis was to determine if a machine learning model that used corporate branding color matching, along with other features, would produce results that are in general both reliable and accurate. We also wished to use this model to produce an effective anti-phishing application. In order to analyze this hypothesis, several types of machine learning models were tested on the proposed data features. One in particular, the Random Forest, was found to adapt well to the mix of binary and numeric data, and produced classifications that were found to be both highly accurate and reliable over long periods of time. As was discussed, the hypothesis can only be proved if all four qualifications are reached. These qualifications are accuracy, speed, reliability, and security. How well Nereus measures against each of these will be examined in this section.

3.1 Accuracy

In order to examine the accuracy and effectiveness of Nereus, several metrics will be shown. The main determination we would like to analyze here is whether or not the color features increase the accuracy of the model significantly. In order to test this hypothesis, we split the prepared data into two sets, with 70% of the data being used for the training and 30% for the test data. The models were trained several times and the best performing model in terms of overall accuracy was recorded using the PICKLE Python module. These best performing models were then analyzed in order to determine the best model overall.

The accuracy ratings, or how likely the model was to accurately classify a data vector

from the test data is represented by the ROC curves shown below. An ROC curve is an excellent visualization of model performance, as it plots the true positive rate versus the false positive rate. This graph therefore shows the model performance at all classification thresholds. A well-performing model maximizes the area found under the curve, or AUC, as it would have a high true positive rate and a low false positive rate at all thresholds. As can be seen in figure 3.1, the SVM model produces an AUC of 0.77. The SVM model produced an AUC of 0.75. Finally, the AUC from the Random Forest model was found to be 0.92, as seen in figure 3.3.

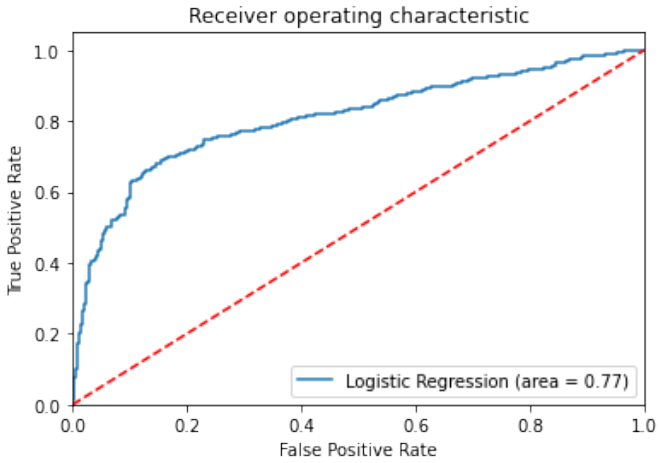

Figure 3.1: ROC Curve for Logistic Regression Model.

From these ROC curves, we can clearly see that the Random Forest model was the most accurate among the three types of models. To further demonstrate this we can examine the confusion matrix, shown in figure 3.4. This confusion matrix demonstrates that the accuracy of the model at predicting true positives is at a strong 94%. This is important, as the application will be judged in practical terms at how often it correctly identifies phishing attacks. The false positive rate was unfortunately found to be quite high at around 8%. This value will likely drive down user usefulness, as a high rate of false positives will likely discourage users from keeping the application installed.

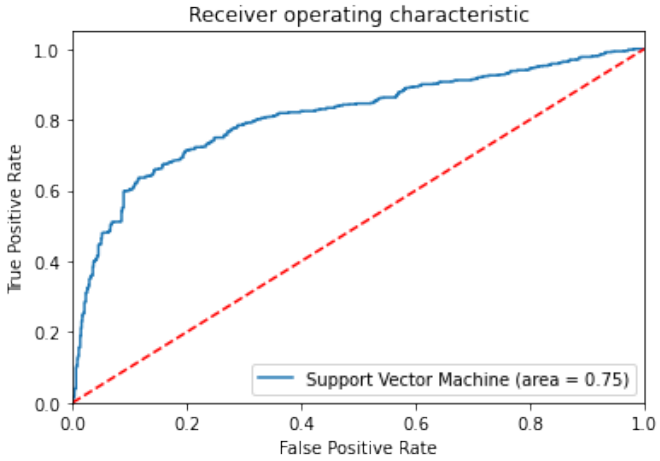

Figure 3.2: ROC Curve for SVM Model.

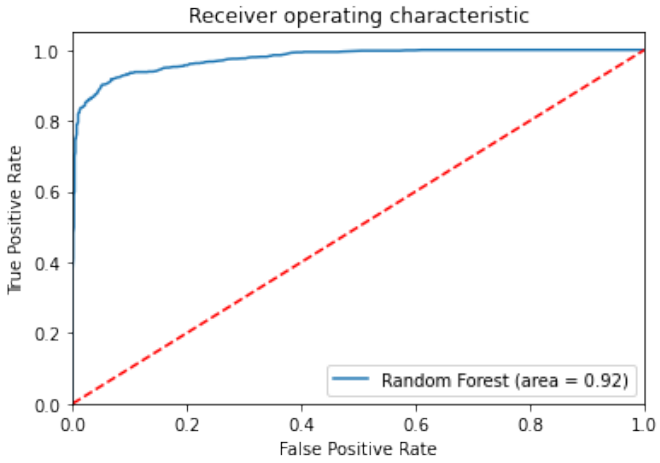

Figure 3.3: ROC Curve for Random Forest Model.

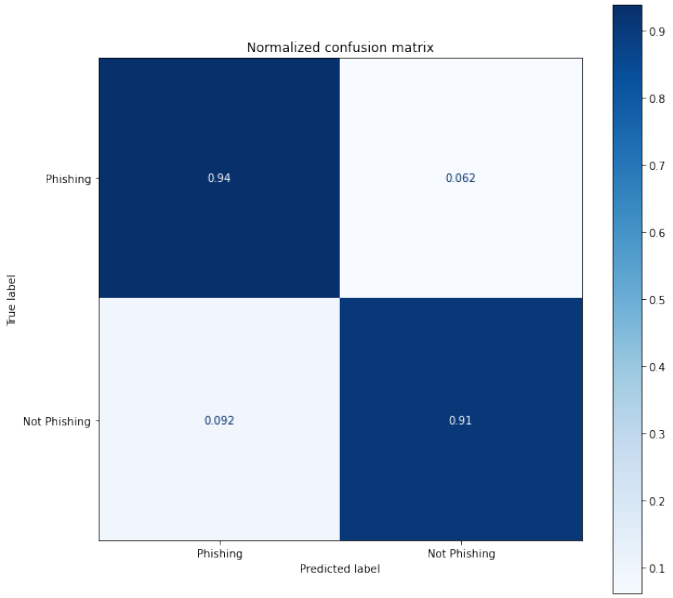

Figure 3.4: Confusion Matrix for Random Forest Model.

Since the hypothesis specifically concerned the effectiveness of the color matching features, a Random Forest model was also trained with the same hyper parameters, but with only the URL and HTML features. The ROC curve for this model is shown in figure 3.5, with the AUC value being 0.78. The confusion matrix for this model in figure 3.6 clearly demonstrates that the color features increase the accuracy of the model by around 10%.

3.1.1 Color Similarity Feature Performance Evaluation

In order to properly validate the hypothesis, we need to confirm that the color branding similarity features not only added to the accuracy but determine which measurements most contributed to this increase. In order to do this, the trained model was used to rank the features based on impurity-based importances. This value was found by averaging the decrease in impurity over trees within the random forest. The results of this can be seen in table 3.1. As can be seen, the most effective features were the presence of an SSL certificate, URL length, followed by five color similarity features. Of these, the distance

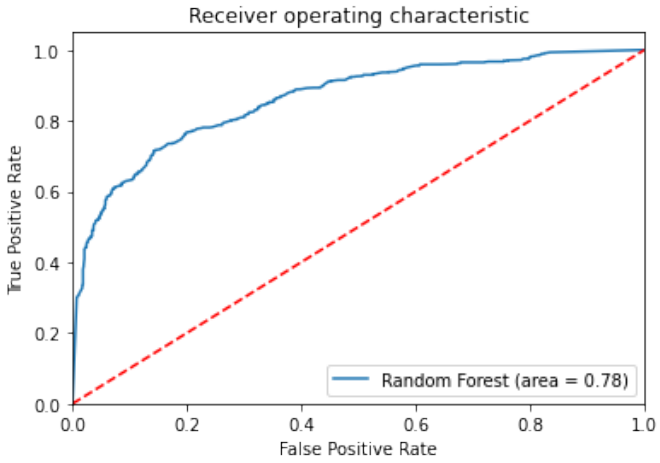

Figure 3.5: ROC Curve for Random Forest Model with no color matching features.

measurement from Paypal and Facebook were found to be the most important. This is likely due to the relative frequency at which Paypal and Facebook are phished. Since these two site most likely made up a large portion of the phishing targets, their importance is explained.

In conclusion, table 3.2 displays the relative accuracies of the four models presented in this section. It is clear therefore that the hypothesis is validated, as the accuracy of the Random Forest model was increased by 13.9% once all the color features were included.

3.2 Speed

In terms of speed, Nereus performs extremely well when using the Random Forest model. This model, combined with the SimpleHTTP Python server returned a prediction to the Google Chrome extension within an average of 0.8 seconds. This added overhead is done immediately after page load, and does not block the usage of the page. This is a key result, as an application that blocks page load is unlikely to be employed by users in the long term. From this metric, we can determine that Nereus maintains a good speed and could be used without adding undue overhead to the user.

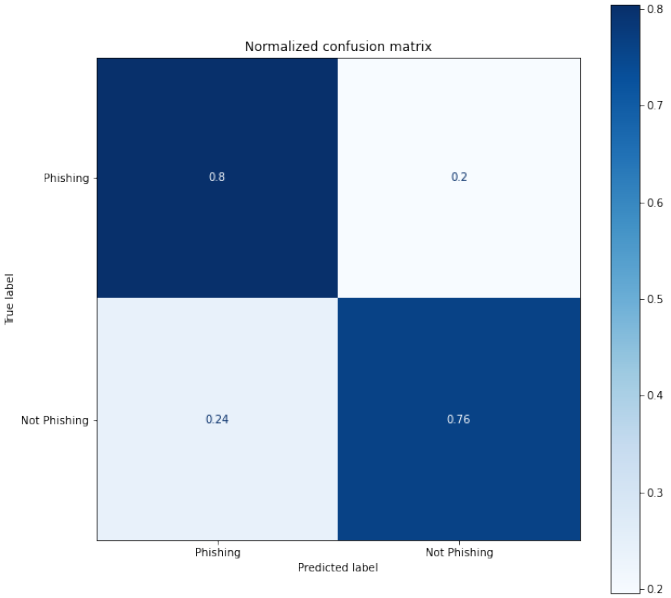

Figure 3.6: Confusion Matrix for Random Forest Model with no color matching features.

Table 3.1: Data Feature Importance

Data Feature Name	Feature Importance (Gini impurity)
Is_HTTPS	0.131244
URL_Length	0.118354
Distance_From_Paypal	0.085505
Distance_From_Facebook	0.083437
Distance_From_WhatsApp	0.068681
Distance_From_Microsoft	0.068655
Distance_From_Google	0.067008
Num_Subdomains	0.064044
Distance_From_Apple	0.063598
Distance_From_Netflix	0.060769
Distance_From_Adobe	0.059037
Distance_From_Orange	0.057924
Distance_From_Amazon	0.056716
Top_Level_Domain	0.013072
Has_Form	0.001384
IP_Address	0.000314
Has_Symbols	0.000258

Table 3.2: Model Accuracy Ratings

Model Accuracy Ratings			
SVM	Logistic Regression	Random Forest without color features	Random Forest with color features
74.9%	76.2%	78.2%	92.1%

3.3 Reliability

In order to judge long-term reliability, a separate longitudinal study would have to be done over a period of several years. In place of this however, a smaller scale experiment has been done. As was described in section 2, the data was gathered in two parts. The first half of the data, 2,486 vectors, were gathered three months before the remaining 3,176. It was found that the Random Forest model described above, when trained only on 70% of the older data, was able to achieve an accuracy of 97.6% on the newer data. This small experiment may show that the color matching data features are more reliable in the long run than the traditional URL and HTML features used by others.

In addition to the short-term longitudinal study, the fact that the model was able to achieve such a high accuracy of 92.1% while only using color matching of ten known corporate branding colors is highly significant. If ten features were able to increase the accuracy of the model by 13.9%, it is believed that adding more over time would not only make the model more accurate, but more reliable.

By using the corporate branding color features, Nereus becomes an extremely reliable model for classifying potential phishing sites. Over time as more users use the application and more diverse data is inputted into the training set, it is believed that the model could grow more accurate still.

3.4 Security

As with any anti-phishing tool, the user expects the solution to not be worse than the problem. While some anti-phishing software running on the client needs to take screenshots of every running page to transmit into their remotely running model, Nereus does not [12, 8]. By performing all feature extraction on the client side in memory before transmitting to the remote server for evaluation, the privacy of the user is protected. After review of the features included, we have judged that it would be extremely difficult for a malicious actor to use the data retrieved by Nereus to violate the privacy of the user. While it could be possible to identify the site that user has visited in broad terms, such as Facebook or Paypal, no personally identifiable information or PII would be stored. It is possible that some novel attack could be conceived as no security or privacy is guaranteed when using a remote tool. In spite of this however, the precautions that Nereus takes in order to protect user privacy positions itself well against any such attacks.

Chapter 4

Conclusions

4.1 Current Status

As it exists today, this thesis has produced a fully functional anti-phishing application. This application positions itself between the user and the websites they visit, protecting them from phishing attacks no matter how they came to visit those sites in the first place. The Google Chrome extension extracts the features rapidly from the site, transmits them to a remote server, which then returns the prediction from the model. The Random Forest model used to make the prediction has been shown to be extremely accurate, especially when it utilizes color matching features. The application is able to identify phishing websites with 92.1% accuracy while adding an average asynchronous overhead time of only 0.8 seconds. It was found that over a period of three months, the model was able to consistently identify phishing attacks with over 90% accuracy. These results have therefore validated the hypothesis that corporate branding color matching features can be used as an extremely strong feature in a phishing classifier. While there are some limitations to this work as it exists today, it is believed to be in a very strong position.

4.2 Future Work

There are several limitations to this work that in the future, would need to be addressed. First and foremost, the data gathered to train the model is likely somewhat biased in that the non-phishing data was gathered only from a few volunteers. In order to make the model more generally accurate, the data needs to be amplified by more data from a variety of

sources. One way in which this could be done is have some users of the application manually verify the results of the model. This curated data could then be fed back into the training set and used to train the model with new data as well as the old. Though the application utilizes the corporate color matching features to great effect, adding more branding colors would likely increase the accuracy of the model. In a commercial application, companies could apply for their branding colors to be included in the application. This would not only increase the accuracy of the application, but also increase the relevancy of the results for many corporate entities. Since most anti-phishing software is used by corporate security, this could make the application more useful for a key demographic. Finally, in the future, the ability to use Nereus from more browsers would be ideal. As it stands, the application can only operate from within the Google Chrome browser. This limits the type of phishing attacks it can identify to those affecting desktop internet users. Since phishing also plagues mobile device users, it would be ideal if Nereus could be used within them as well. At the moment it is unknown if any mobile internet browsers allow third party extensions, but such integration would make Nereus much more widespread and effective.

4.3 Lessons Learned

Many lessons were learned throughout the development of this thesis. Over the course of the project, it was demonstrated by preliminary results that the data needed to be diversified as much as possible. This led to the final data gathering approach as discussed in section 2. The analysis of the results also posed an interesting problem, as such a large and multi-faceted application had many different ways of measuring effectiveness. By breaking the effectiveness down into the four metrics of speed, accuracy, reliability, and security, we were able to focus on what truly mattered to the user of the software. Throughout this process invaluable experience was gained, and it was in no small part due to the help of my many advisors and colleagues that Nereus is able to be presented today.

Bibliography

[1] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[2] Elie Bursztein. Understanding why phishing attacks are so effective and how to mitigate them, 2019.

[3] Jose Luis Caivano and Mabel Amanda Lopez. Chromatic identity in global and local markets: analysis of colours in branding. *Journal of the International Colour Association*, 1(3):1–14, 2007.

[4] Ye Cao, Weili Han, and Yueran Le. Anti-phishing based on automated individual white-list. In *Proceedings of the 4th ACM Workshop on Digital Identity Management*, DIM '08, page 51–60, New York, NY, USA, 2008. Association for Computing Machinery.

[5] A. Carella, M. Kotsoev, and T. M. Truta. Impact of security awareness training on phishing click-through rates. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 4458–4466, 2017.

[6] E. H. Chang, K. L. Chiew, S. N. Sze, and W. K. Tiong. Phishing detection via identification of website identity. In *2013 International Conference on IT Convergence and Security (ICITCS)*, pages 1–4, 2013.

[7] J. Chen and C. Guo. Online detection and prevention of phishing attacks. In *2006 First International Conference on Communications and Networking in China*, pages 1–7, 2006.

[8] Teh-Chung Chen, Torin Stepan, Scott Dick, and James Miller. An anti-phishing system employing diffused information. *ACM Trans. Inf. Syst. Secur.*, 16(4), April 2014.

[9] Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, 35(5):352 – 359, 2002.

[10] M. Dunlop, S. Groat, and D. Shelly. Goldphish: Using images for content-based phishing analysis. In *2010 Fifth International Conference on Internet Monitoring and Protection*, pages 123–128, 2010.

[11] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 649–656, New York, NY, USA, 2007. Association for Computing Machinery.

[12] A. Y. Fu, L. Wenyin, and X. Deng. Detecting phishing web pages with visual similarity assessment based on earth mover's distance (emd). *IEEE Transactions on Dependable and Secure Computing*, 3(4):301–311, 2006.

[13] Google. Phishing protection.

[14] M. Khonji, A. Jones, and Y. Iraqi. A novel phishing classification based on url features. In *2011 IEEE GCC Conference and Exhibition (GCC)*, pages 221–224, 2011.

[15] Christian Konradt, Andreas Schilling, and Brigitte Werners. Phishing: An economic analysis of cybercrime perpetrators. *Computers & Security*, 58:39–46, 2016.

[16] G. Liu, B. Qiu, and L. Wenyin. Automatic detection of phishing target from phishing webpage. In *2010 20th International Conference on Pattern Recognition*, pages 4153–4156, 2010.

[17] James Moar. The future of cybercrime & security: Financial and corporate threats & mitigation. *Juniper, Dec*, 2015.

[18] Rami M. Mohammad, Fadi Thabtah, and Lee McCluskey. Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications*, 25(2):443–458, Aug 2014.

[19] Y. Pan and X. Ding. Anomaly based web phishing page detection. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 381–392, 2006.

[20] N. Pavković and L. Perkov. Social engineering toolkit — a systematic approach to social engineering. In *2011 Proceedings of the 34th International Convention MIPRO*, pages 1485–1489, 2011.

[21] PhishTank. Phishtank — join the fight against phishing, 2020.

[22] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13), Sep 2014.

[23] Sagar Shivaji Salunke. *Selenium Webdriver in Python: Learn with Examples*. CreateSpace Independent Publishing Platform, North Charleston, SC, USA, 1st edition, 2014.

[24] Wenyin Liu, Xiaotie Deng, Guanglin Huang, and A. Y. Fu. An antiphishing strategy based on visual similarity assessment. *IEEE Internet Computing*, 10(2):58–65, 2006.

[25] Yue Zhang, Jason I. Hong, and Lorrie F. Cranor. Cantina: A content-based approach to detecting phishing web sites. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 639–648, New York, NY, USA, 2007. Association for Computing Machinery.

[26] Zhao Zhang, Qinggang He, and Bailing Wang. A novel multi-layer heuristic model for anti-phishing. In *Proceedings of the 6th International Conference on Information Engineering*, ICIE '17, New York, NY, USA, 2017. Association for Computing Machinery.