Anti-Phishing Browser Extension

Aaron King, Connor Gannaway, Caleb Kornegay, Christopher Canaday
The University of Tennessee, Knoxville
{aking100, cgannaw1, ckornega, ccanada6} @vols.utk.edu

Abstract

This paper proposes developing a Google Chrome compatible anti-phishing browser extension to evaluate the effectiveness of various phishing detection methods. The extension integrates techniques such as DNS record analysis, certificate chain verification, website feature extraction, URL matching, machine-learning analysis, and queries to an updated database of known phishing sites. We will test the extension against known phishing and non-phishing websites to measure detection accuracy, focusing on false-positive and false-negative rates. Through quantitative and qualitative analyses, we aim to identify the most effective detection methods. This project seeks to enhance cybersecurity by providing users with a lightweight, user-friendly tool to reduce exposure to phishing scams.

1. Introduction/Motivation

This project proposes developing an anti-phishing browser extension that evaluates which phishing detection methods are most effective. Phishing, a form of cybercrime, involves tricking users into revealing sensitive information like passwords and credit card numbers through deceptive websites or emails. The proposed extension will utilize a combination of DNS records, certificate chains, copyright on the website, URL matching, website feature extraction, and queries to a regularly updated database of known phishing sites to identify potential threats. When a suspicious site is detected, the extension will immediately alert the user and block access, thus minimizing the risk of data breaches. This tool is designed to be user-friendly, lightweight, and compatible with Google Chrome. By providing an additional layer of security, this extension will empower users to browse the internet more safely, reducing their exposure to phishing scams. The project seeks to contribute to overall cybersecurity by making protection accessible to all users. We have three research questions we would like to answer: “Which phishing detection method is most effective?”, “What combination of different methods would create the best possible browser extension?”, and “Is it possible to detect phishing sites with reasonable accuracy using only site metadata and no content analysis?” Our current hypotheses are: “Phishing sites cannot be categorized as phishing using only one method”, “A combination of query URL analysis and page content analysis with the assistance of a language model will lead to the best results”, and “Page content analysis will be required to achieve an adequate level of accuracy.”

2. Methods: Extension Development

2.1. React Extension

The React framework, in combination with Webpack, were chosen as the tools used to develop the browser extension. We started with *create-react-app's* built-in Chrome extension template, which sets up our application and includes a manifest.json file, which is required by Chrome. This command is: *npx create-react-app react-chrome-ext --template typescript.*

We developed three primary views for the user interface: a settings view, a test list & results view, and a base view that toggles between the two. Index.tsx is the entry point for the application. It renders App.tsx, which subsequently renders views/settings.tsx or settings/tests.tsx.

The settings page operates using Chrome's *localstorage API*. The two settings values being saved are *apiKeyValue*, an OpenAI API key, and *costlyTests*, a boolean on whether or not to run costly tests. The only test that is affected by *costlyTests* is Footer Check. The settings page also has buttons for clearing and exporting test data, as the extension saves results every time it runs.

Each test is in its own file in the tests folder. Each test has a return type of *Promise<TestState>*. *TestState* is an enum of different outcomes or errors a test can encounter - the definition is found in the types folder - and it is required to be a promise to let all of the tests run asynchronously together. The *runTests* function inside of views/tests.tsx is responsible for running all of the tests, waiting for the last one to resolve, and making a final decision on if the current site is a phishing site or not.

Code used by multiple tests is abstracted to the hooks folder, as is standard for react applications.

2.2. Background Worker

One test - Footer Check - requires the HTML source from the current website. This functionality requires Chrome's scripting permission and cannot be achieved from the extension itself. To achieve this, a background worker script with this functionality, found at workers/background.js, was created. This script is marked in the manifest as a background worker and is separate from the extension itself. For the extension to call this code, it sends a message to the Chrome runtime with a specific action request, prompting the background worker to respond.

2.3. API

During the development of the extension, we realized that some functionality of our tests would be unattainable from the extension alone. This includes storing a database of known phishing URLs, as that would be too large for an extension, and calling the source APIs can be rate-limited and cost money, as well as running a machine learning model to do predictions based on the URL. Our solution was to offload these tasks to our own HTTP API that we developed alongside the extension. This is a separate application, found in the API folder, that runs on the Node.js runtime with the Express package.

2.4. Availability

The project is available on Github [8].

3. Methods: Data Collection

3.1. Certificate Check

This method will query the host provider's server for their SSL certificate. We then check the first certificate root and its root (if available) against IoS 17 trusted root certificates [5]. If the issuer on the certificate matches a line in our trusted root certificate database then we trust the certificate, otherwise, we suspect the host to be a phishing site.

3.2. Database Check

This method involves checking the URL against a database of known phishing sites. Many phishing websites have already been discovered and proven to be phishing through various methods. Checking against this database is a very fast way to check if it has already been detected as phishing by others, expanding our accuracy through peer collaboration.

Due to the size of the dataset [6], we hosted the dataset on an external server. Along with this was an HTTP API that the client-side test could query.

3.3. DNS Check

This method incorporates checking the DNS A records to determine whether a site is valid. This is not a foolproof method of checking if a site is legitimate or not because phishing websites can have legitimate DNS providers. To add some more granularity to this method, we check the DNS provider's time-to-live (TTL) and see if it exceeds a threshold that we set. Having a high TTL means that the DNS record will be cached on the end device for a longer period of time. This may be useful for a phishing website, as it may be removed from a DNS provider but is still routable, so they could still prey on victims longer.

3.4. URL Machine Learning Analysis

This method uses a machine learning model to predict phishing sites using the website URL. To train the model we created a dataset that contains 1,350,205 benign URLs and 670,107 phishing URLs. This dataset was created by combining two published datasets. This first dataset contains 40 features of a URL and has 128,541 phishing URLs and 119,409 benign URLs [4]. The second dataset contains 270,783 phishing URLs and 615,398 benign URLs [7]. The second dataset only contains the URL and not the features of the URL. It also does not contain the scheme of the URL. Thus we added both possible schemes, http and https, to all URLs in the second dataset effectively doubling it in size. After this, we computed the features of all URLs in the second dataset and appended them to the first dataset. Using our newly created dataset we trained our ML model on 80% of this dataset and then tested the accuracy on 20% of the dataset. The confusion matrix is visible in Figure 1. The ML model is a Random Forest Classifier with 100 estimators and a random state of 0. After training, we also compute the importance of the different features; those results are graphed in Appendix 1. The generated model is larger than what would reasonably fit inside a browser extension so we have chosen to host the model on an external server and use an API within the extension to query the model for phishing classification. This API takes a URL, calculates its features, then uses the model to predict if the website is phishing, and returns the percentage chance that the website is phishing to the API caller. This model is 93% accurate on its training dataset.

Two of the calculated features involve the entropy of the URL. This entropy is the Shannon entropy.

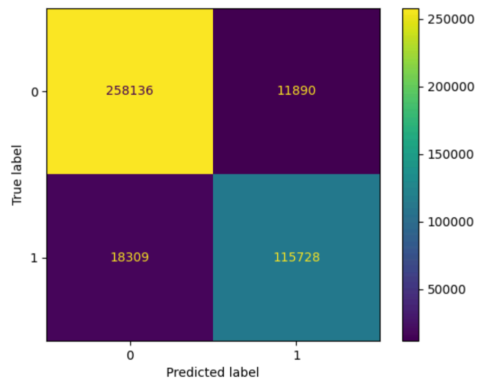


Figure 1. ML Model Confusion Matrix

3.5. GPT-4o URL Check

GPT-4o can be a great resource for checking if URLs are off by only a few characters or attempting to piggyback off legitimate websites or typo-squat. In our testing, we realized that GPT-4o is accurate in recognizing that URLs are very close to legitimate sites but only off by a few characters. This method was able to successfully determine that “myutk.edu” is not a legitimate URL, but “myutk.utk.edu” is. The prompt we use when querying GPT-4o is “Does this site look like a phishing URL? Answer 'Yes' or 'No' only please:” and then the URL we are testing. We conducted various other tests and it was very accurate. While this method is not a great method by itself, it is a valuable asset when paired with others in the browser extension.

3.6. Footer Check

The footer check is the only content-analysis-based check we incorporated. It is powered by GPT-4o. We recognize that a lot of branded sites have a footer, and within that footer contains identifying information about the brand, such as the name itself, addresses, or other links. By feeding the footer HTML of a site to GPT-4o with the instructions to identify the brand, we can discover the site the attackers are attempting to imitate. The query used for this is: “The following is the HTML for the footer of a website. Your job is to identify the brand of the website. You may only respond with the brand name itself, or with 'Unknown' if you cannot determine the brand. Footer: “. If GPT-4o returns “Unknown,” we mark the test as inconclusive as the website either does not have a footer or there is not enough information in the footer to identify a brand. If we do have a brand, we query GPT-4o again, asking for the current website for this brand. It is likely that GPT-4o will be able to respond with the current correct site as the models are being updated with new information regularly. The model used throughout this project has a training tag of 2024-08-06. If GPT-4o cannot find the live site, the test is marked as inconclusive. Finally, we apply regex to both the URL returned by GPT-4o and the current URL and check to see if they match. If they do, the test passes. The query for finding the URL is: “What is the current URL for the following brand. Only respond with the URL itself or with 'Unknown' if you cannot determine the URL. Brand: “.

4. Methods: Data Analysis

After moving our extension's test into something we could run outside of the browser for automated testing, we collected a large set of phishing and non-phishing URLs and had a classification of which ones were phishing and which were not to check against later. After collecting all of our data, we then used it to extract our true positive (TP), false positive (FP), true negative (TN), and false negative (FN) rates, from these we derived classification accuracies. For each test, this was relatively simple, check each against the actual and count how many are TP, FP, TN, and FN. Accuracy is calculated using the formula (TP + TN) / (TP + FP + TN + FN). The overall accuracy has a little more nuance, in Figure 8, the overall accuracy is derived from each test's decision coinciding with the decisions of the other tests (i.e. all tests have to say phishing for it to be phishing). In Figure 9, the accuracy is calculated as one test classification, if one test declares phishing, then it is marked as phishing.

5. Results

5.1. Certificate Check

The Certificate Check had a 70.71% accuracy. See Figure 2 for the confusion matrix. These results are about what was expected, a good distribution of true positives and true negatives with a lower rate of false negatives and very low false positives. The false positives are probably due to sites in another country that had certificate authorities that were not on our list to check against. The high false negative rate is because phishing websites can have valid certificates. They may be hosted on a site provider like Wix or something else.

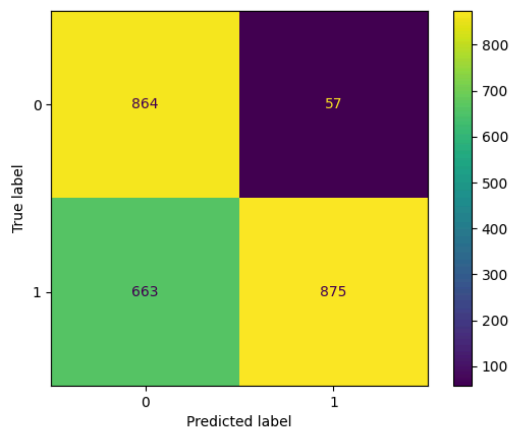


Figure 2. Certificate Check Confusion Matrix

5.2. Database Check

The Database Check has an accuracy of 39.08%. There is a good reason for the accuracy of this test being as low as it is. To preserve the integrity of our testing, we sourced the database used for this test from a different source than our testing data. Because of this, the accuracy of detecting phishing sites relies on the natural human reporting these datasets are created from. Since we used live sites for testing, as all but one of our tests require site metadata, there is a greater chance that a site doesn't appear on every phishing database than if it did. In practice, this test should pull from multiple sources, which would result in a higher accuracy. See Figure 3 for the confusion matrix.

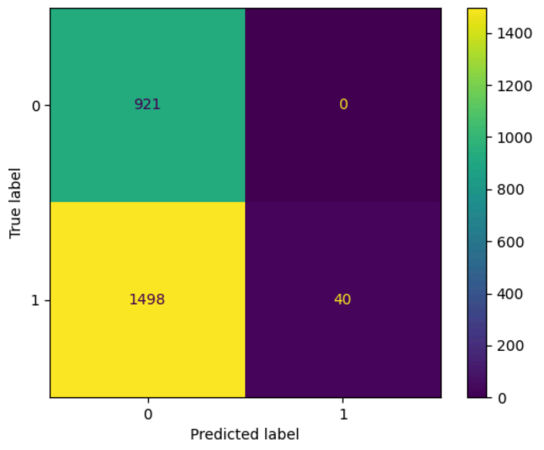


Figure 3. Database Check Confusion Matrix

5.3. DNS Check

The DNS Check has an accuracy of 34.73%. See Figure 4 for the confusion matrix. This test performed worse than expected; however, its saving grace is that the false positive rate is relatively low. The reliance on DNS alone is not a good phishing detection method. Time To Live (TTL) was a metric we used to determine if a site was phishing. The threshold was 3600 [seconds], whereas the industry standards are 30 and 60 seconds for commercial and personal use, respectively. End-user devices also do not play by the TTL rules, as they will not evict a cached DNS entry until much later, as querying a DNS server takes more time.

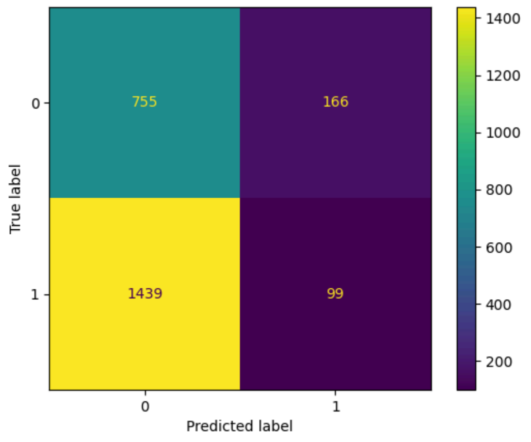


Figure 4. DNS Check Confusion Matrix

5.4. URL Machine Learning Analysis

The URL ML Analysis achieved an accuracy of 63.81%. See Figure 5 for the confusion matrix. Based on these results we believe this test to be very promising. Although lacking in accuracy, the model rarely identifies a legitimate site as a phishing site. It does, however, commonly think phishing sites are legitimate sites. So it wouldn't get in the way of a user's normal use, but it is also possible that it does not properly warn a user of a phishing site and even cause them to mistakenly trust a phishing website. Given the relatively small dataset used to train this model, we believe that this model, with improvement, could achieve high accuracy and become a good method of phishing detection. We believe improvements such as larger curated datasets of benign and phishing URLs and more URL features could improve the results of this test.

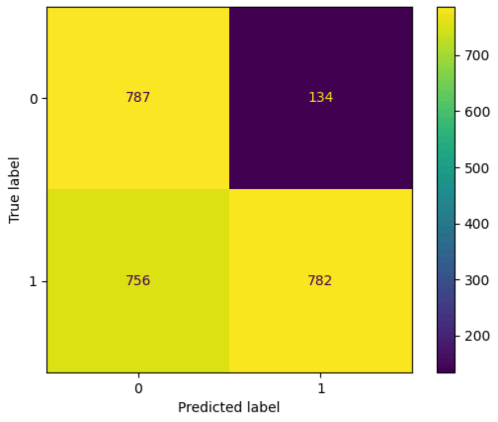


Figure 5. URL ML Analysis Confusion Matrix

5.5. GPT-4o URL Check

The GPT-4o URL Check achieved an accuracy of 71.74%. See Figure 6 for the confusion matrix. This check worked surprisingly well. It turns out that just asking GPT-4o was our most reliable way of detecting a phishing website outside of content analysis. Looking at the confusion matrix, we can see that it is very rare for GPT-4o to classify a legitimate website as a phishing website. It does however tend to classify phishing websites as legitimate websites. We presume that this happens when phishing websites imitate a website that is not in GPT-4o's training set. We also believe that the reason this test does so well compared to our own URL ML Analysis is that GPT-4o is trained on a much larger dataset than ours. This test should improve as GPT-4o upgrades over time. The one true drawback of this test is that you must have an API key with OpenAI and every website you visit will incur a fee. This fee is small but is still a large barrier to entry.

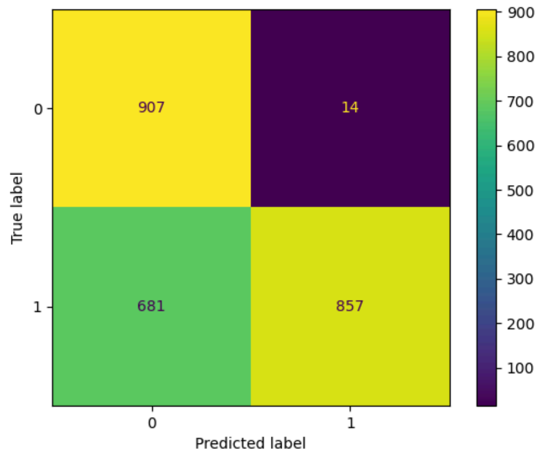


Figure 6. GPT-4o URL Check Confusion Matrix

5.6. Footer Check

The Footer Check achieved an accuracy of 83.64%. See Figure 7 for the confusion matrix for this test. It was expected that this test, because it is a content-analysis-based test, would be one of the higher-performing phishing tests this paper explores. It ended up being the highest. It was very accurate when classifying a phishing site, with a True Positive rate of 92.19%. This is likely due to many phishing sites attempting to imitate a different brand, which is exactly what this test is designed to catch. This test does not come without some drawbacks, however. The test had a True Negative rate of 81.04%, meaning it misclassified almost 20% of non-phishing sites. This could be due to a handful of things, but most likely due to how the check handles subdomains and sub-brands (i.e. Gmail, Google Drive, Google Docs, Google Search).

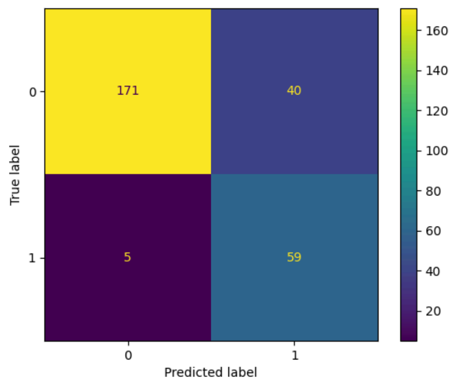


Figure 7. Footer Check Confusion Matrix

5.7. Overall Results

This does not include the Footer Check, as we want to evaluate checks based on metadata only. Figure 8 is the confusion matrix of when all tests agree on their decision to mark a site as phishing. We can see that even on actual phishing websites, the confidence of all tests agreeing that it is phishing is 0%, meaning we didn't classify a single phishing website as phishing categorically. Using this evaluation method we achieved an accuracy of 24.64%. Figure 9 is the confusion matrix of one test declaring the site as phishing classifies the site as phishing, which has a much higher accuracy; however, this could be improved upon by having the lower performing tests removed, or by having a threshold of the number of tests required to classify as phishing, maybe two or three instead of one. Using this method we achieved an accuracy of 75.07%. It may also be a good idea to weigh each method to achieve an even higher accuracy.

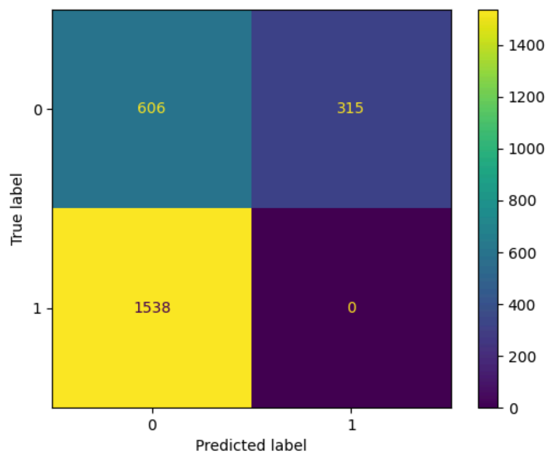


Figure 8. Overall Confusion Matrix (all tests coincide)

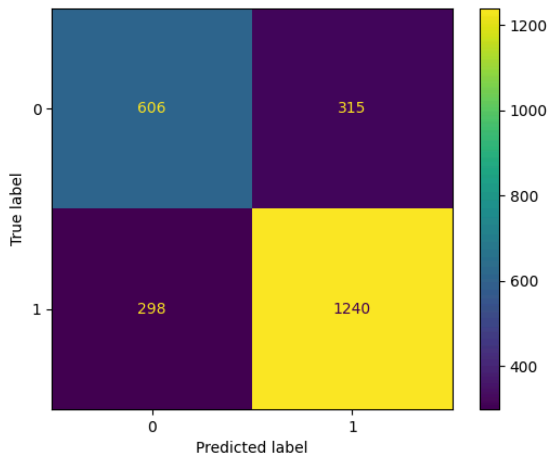


Figure 9. Overall Confusion Matrix (any test marks a site as phishing)

6. Limitations

The footer extraction test could not be run on the entire dataset because the token input into GPT-4o was high, which would incur a large cost. However, for a normal user browsing the net (not visiting 2000+ URLs quickly), the cost should be relatively bearable.

7. Related Work

Previous work has mainly focused on developing different methods to detect phishing and evaluating their accuracy. We have used an analysis of these different methods to guide our choices of phishing detection methods [1]. There have been efforts to create a machine learning-based browser extension to detect phishing, but we wanted to focus on using the metadata of the site [2].

8. Budget

An OpenAI API key is the only cost factor for the development or running of this project. This cost is paid for by the user and is required for the URL Check and Footer Check to function properly. Throughout the development, testing, and evaluation of this project, our final OpenAI bill was $3.00.

9. Conclusion

In conclusion, we believe that metadata-based phishing detection is a promising phishing detection method, but currently, content analysis is required to have strong confidence in the prediction. Our results have aligned with our hypothesis on our research questions.

References

[1] O. B. Singh and H. Tahbildar, “(PDF) A literature survey on anti-phishing browser extensions,” https://www.researchgate.net/publication/282837211_A_Literature_Survey_on_Anti-Phishing_Browser_Extensions (accessed Sep. 20, 2024).

[2] L. Thaçi, A. Halili, K. Vishi, and B. Rexha, “Efficient chrome extension for

phishing detection using Machine Learning Techniques," https://arxiv.org/html/2409.10547v1 (accessed Sep. 20, 2024).

[3] S. Maurya, H. Saini, and A. Jain, "Browser Extension based Hybrid Anti-Phishing Framework using Feature Selection," https://thesai.org/Downloads/Volume10No11/Paper_78-Browser_Extension_based_Hybrid_Anti_Phishing_Framework.pdf (accessed Sep. 20, 2024).

[4] T. M. Ahmed, I. M. Kabirul, B. Touhid, and S. Abdus, "Dataset of suspicious phishing URL detection," *Frontiers in Computer Science*, 2024. https://www.frontiersin.org/journals/computer-science/articles/10.3389/fcomp.2024.1308634 (accessed Oct. 16, 2024).

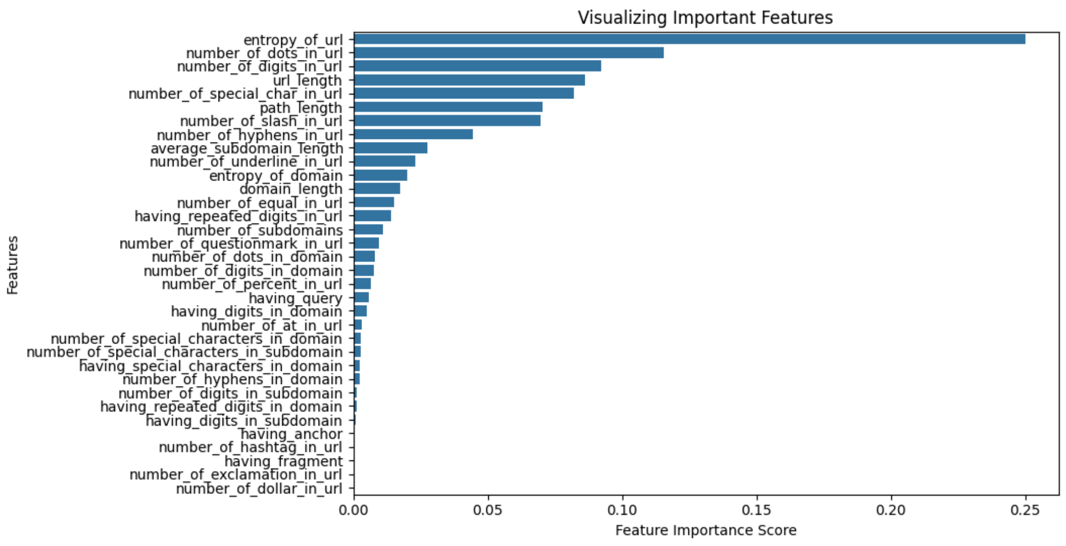
[5] Apple Support, "List of Available Root Certificates in IoS 17," https://support.apple.com/en-us/105116 (accessed Oct. 23, 2024).

[6] M. Krog, "mitchellkrogza/Phishing.Database." Phishing.Database, Github, https://github.com/mitchellkrogza/Phishing.Database (accessed Nov. 19, 2024).

[7] A. K. Natarajan, "dataset-phishing2," *Huggingface.co*, 2015. https://huggingface.co/datasets/itsprofarul/dataset-phishing2 (accessed Nov. 19, 2024).

[8] C. Canaday, C. Gannaway, A. King, and C. Kornegay, "COSC469", Repository, Github, https://github.com/CalebKornegay/COSC469.

Appendix 1 - Important Features Graph


Visualizing Important Features
entropy_of_url
number_of_dots_in_url
number_of_digits_in_url
url_length
number_of_special_char_in_url
path_length
number_of_slash_in_url
number_of_hyphens_in_url
average_subdomain_length
number_of_underline_in_url
entropy_of_domain
domain_length
number_of_equal_in_url
having_repeated_digits_in_url
number_of_subdomains
number_of_questionmark_in_url
number_of_dots_in_domain
number_of_digits_in_domain
number_of_percent_in_url
having_query
having_digits_in_domain
number_of_at_in_url
number_of_special_characters_in_domain
number_of_special_characters_in_subdomain
having_special_characters_in_domain
number_of_hyphens_in_domain
number_of_digits_in_subdomain
having_repeated_digits_in_domain
having_digits_in_subdomain
having_anchor
number_of_hashtag_in_url
having_fragment
number_of_exclamation_in_url
number_of_dollar_in_url
Features
Feature Importance Score