

Detecting email-based phishing websites using machine learning

23-123

Project Report

Mandira Pabasari L.

IT19966236

Supervisor: Ms.Jenny Krishara

B.Sc. (Hons) Degree in Information Technology

Specialization in Cyber Security

Department of Computer Systems Engineering

Sri Lanka Institute of Information Technology


Sri Lanka

September 2023

DECLARATION, COPYRIGHT STATEMENT AND THE STATEMENT OF THE SUPERVISOR

“I declare that this is our own work and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).”

Name	Student ID	Signature
Mandira Pabasari L.	IT19966236	

The above candidates are carrying out research for the undergraduate Dissertation. under my supervision.

Signature of the supervisor: Date:

Signature of the supervisor: Date:

Abstract

This project centers around the development of a robust machine learning-based classification system aimed at identifying phishing websites. Phishing websites impersonate legitimate entities, posing a significant cybersecurity threat. To mitigate this risk, the project encompasses three crucial phases: data preprocessing, model training, and model serialization.

In the data preprocessing phase, a carefully curated dataset is loaded and processed. The dataset includes features extracted from websites, such as URL structure, content, and server information. Data preprocessing ensures that the dataset is appropriately formatted and scaled, preparing it for model training.

For model training, a Random Forest Classifier is employed. This ensemble learning technique combines multiple decision tree classifiers to enhance predictive accuracy. The Random Forest Classifier is trained on the preprocessed dataset, undergoing cross-validation to evaluate its performance robustly. Cross-validation helps prevent overfitting and ensures the model generalizes well to unseen data.

Once the Random Forest Classifier demonstrates satisfactory performance, it is serialized into a JSON format. Serialization is a pivotal step, as it allows for the easy deployment of the model in various applications and environments. The JSON representation of the classifier includes details about the individual decision trees within the Random Forest, making it interpretable and transferable.

In summary, this project presents an end-to-end solution for classifying phishing websites, leveraging machine learning techniques and a Random Forest Classifier. The model, trained on a preprocessed dataset, can be seamlessly integrated into cybersecurity tools, web browsers, or other applications to bolster defense mechanisms against phishing attacks.

Acknowledgment

I would like to express my sincere gratitude to our supervisor, Ms. Jenny Krishara for her invaluable guidance and support throughout the research project. Her expertise and insights have been instrumental in shaping the direction of our work and ensuring its success. I am also grateful for the feedback and advice provided by the CDAP panel, which has helped us to refine our research and achieve our goals. Additionally, I would like to thank my team members for their collaboration and technical support, without which this project would not have been possible. Finally, I would like to extend my appreciation to all those who have contributed to this research in various ways.

Table of Contents

Declaration

Abstract

Acknowledgment

Table of Contents

List of Figures

List of Table

List of Abbreviations

List of Appendices

1. Introduction

- Background
- Research gap
- Research problem
- Research objectives

2. Methodology

- methodology
- commercialization aspects of the product
- Testing & Implementation

3. Results and discussion

4. Conclusion

List of figures

Figure 1. phishing example

Figure 2. phishing example

Figure 3. system diagram

Figure 4. Data splitting

Figure 5. Tree to json

Figure 6. Random forest to json

Figure 7. Connection flow

Figure 8. Neural networks

Figure 9. Random forests

Figure 10. Support vector machine

Figure 11. dataset

Figure 12. Cross validation and Accuracy

Figure 13. Legitimate site detection

Figure 14. Phishing site detection

List of Table

Table 1. Website features

I. INTRODUCTION

What is a phishing attack?

Phishing is a form of social engineering that is used to obtain user information, such as login information and credit card details. It happens when an attacker deceives the victim into clicking a dangerous link by posing as a reliable source. The attack may result in the installation of malware, system freeze, or the release of private information. As part of an advanced persistent threat event, it can also be utilized in business or governmental networks, leading to significant financial losses and security incidents. [1]

Phishing is a crime that involves both social engineering and technological fraud to steal users' financial account information and personal identity information. By tricking their victims into thinking they are dealing with a reliable, respectable entity, social engineering scams prey on unsuspecting victims. Technological deception approaches sometimes employ devices that intercept users' usernames and passwords for accounts or misdirect users to fake websites in order to collect credentials straight from PCs. [2]

Examples:

- As many faculty members as feasible receive a mass distribution of a counterfeit email purporting to be from myuniversity.edu.
 - The user's password, according to the email, is soon to expire. The instructions state that they must renew their password within 24 hours by visiting myuniversity.edu/renewal.
- [1]

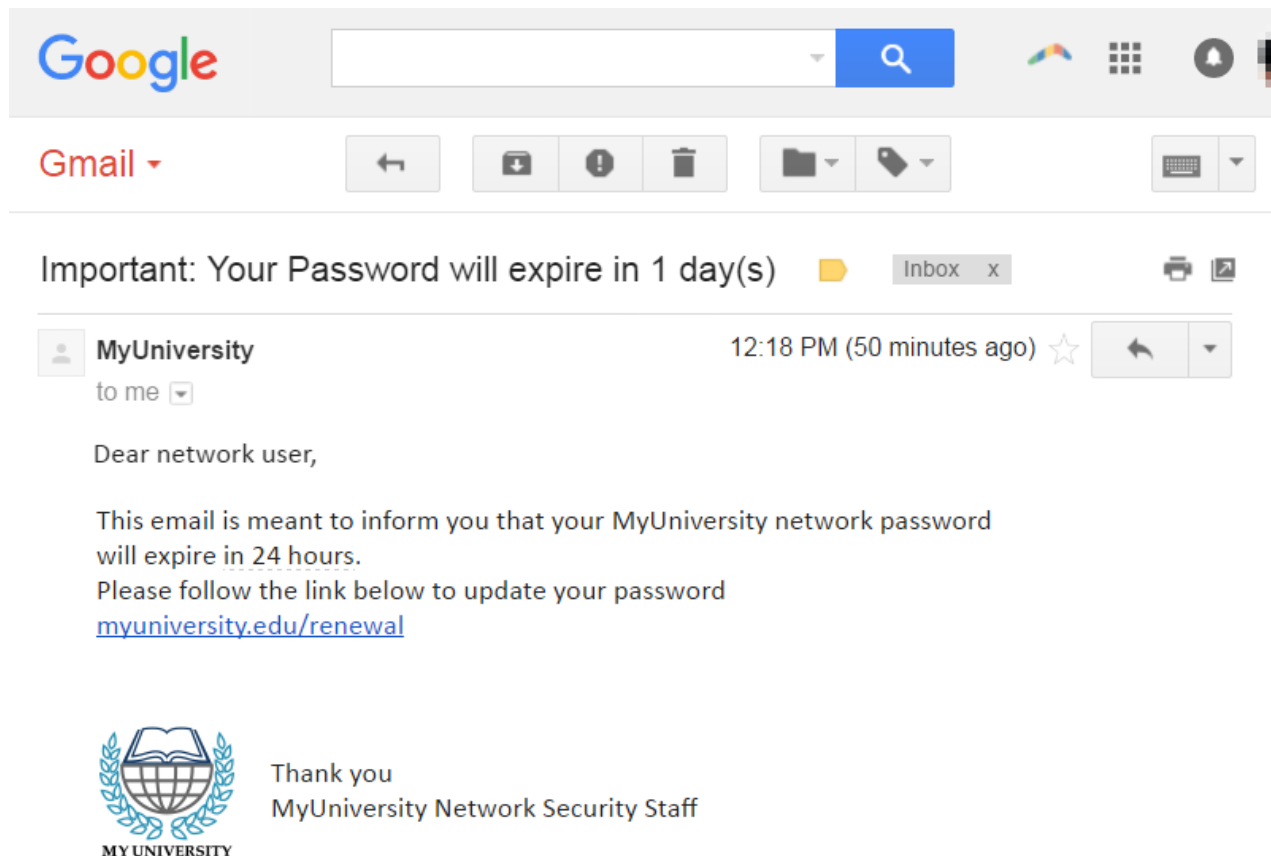


Figure 1. phishing example

Clicking the link can result in a number of outcomes. For instance:

- The user is forwarded to myuniversity.edurenewal.com, a fake page that looks exact like the legitimate renewal page and asks for both new and old passwords. While keeping an eye on the website, the attacker takes control of the original password and uses it to access restricted parts of the university network.
- The user is then sent to the password renewal page itself. A malicious script, however, starts running in the background as the user is being routed in order to steal their session cookie. As a result, an XSS attack is reflected, providing the offender access to the university network. [1]

Phishing techniques

1. Email phishing scams

Email phishing is a sophisticated form of social engineering that may be used to obtain user information including credit card numbers and login information. To reduce the recipient's awareness of the attack, attackers employ strategies such as email impersonation, invoking urgency, and using misspelled domain names or subdomains. Phishing can result in fraudulent transactions, money loss, or identity theft. It can also be used as a launching pad for larger attacks like advanced persistent threats (APT) to infiltrate business or governmental networks. Businesses that fall prey to phishing assaults frequently experience significant financial losses, decreased market share, reputational damage, and a loss of customer trust. A phishing attempt could turn into a security problem that is challenging to resolve depending on its breadth. [1]



Figure 2. phishing example

2. Spear phishing

Spear phishing targets particular people or organizations, therefore understanding their hierarchies is necessary. The names of personnel are looked up, and project invoices are accessed. They send a password-protected document link in an email to the project manager. When the PM logs in, the attacker steals their credentials and gains access to all restricted parts of the network. The initial stage of an APT can be carried out using this technique. [1]

How To Recognize Phishing

Scammers use phishing attacks to collect personal data like passwords, account numbers, or Social Security numbers. These attacks may result in account access or the sale of information to more scammers. Scammers regularly adapt their techniques to stay on top of the most recent news, and popular strategies include presenting a story, posing as a reputable organization, or using a strange website or app. These kinds of attacks are successful and frequently include opening attachments or clicking on links.

Prevent Phishing Attacks:

By examining the message's origin, software, and look, spam filters can identify spam emails.

- Only trustworthy websites should be permitted to open by changing browser settings.
- Never use the same password for several accounts and change your passwords frequently.
- It is advised to use a CAPTCHA system for increased security.
- Monitoring systems are used by banks and other financial institutions to stop phishing.
- People can inform industry groups about phishing to pursue legal action.
- Employees should receive security awareness training from their employers.
- To stop phishing, surfing behaviors must be altered. Make direct contact with the business if confirmation is necessary.
- All sites will eventually be required to have an SSL certificate, which secure websites with a "https" prefix start with. [3]

a. Background literature

- ❖ Using the UCI Dataset of Phishing Websites, the study analyzes machine learning strategies to defend against phishing attempts. The dataset includes 11055 entries, each with 30 attributes, of websites classed as benign and phishing. The dataset was reduced to 22 features, including URL Length and Google Index, in order to extract all of the features. Three algorithms—K-Nearest Neighbors (kNN), Support Vector Machines (SVM), and Random Forest—were taken into consideration for categorizing the given URL as safe or dangerous. In contrast to SVM, which separates input data into two classes with labels of either 0 or 1, KNN is used to solve both classification and regression issues. Decision trees are used by Random Forest to classify data, and numerous decision trees are created during training. Each of these trees has the same distribution and is made up of a combination of tree predictors. Due to the rule of big numbers, they do not overfit data and are an excellent tool for prediction. The results demonstrate that the best classifiers for identifying phishing attempts are TP True Positive, TN True Negative, FP False Positive, and FN False Negative. When a user enters a URL, a Google Chrome Extension is suggested to show whether the URL is legitimate or fraudulent. A Chrome Extension is coded using HTML, JavaScript, and CSS. When a user enters a URL, the extension uses Java script to transfer the URL to the Python code via the application. [4]

- ❖ In this study, a suggested approach for identifying phishing attempts by examining their website's properties is presented. The methodology focuses on detecting attacks by examining phishing website characteristics, including URLs, domain identities, security and encryption, source code, page style and contents, web address bars, and social human aspects. The Uniform Resource Locator (URL) is a tool used to distinguish between legitimate and fraudulent websites. Using C# and Microsoft Visual Studio Express 2013, the program PhishChecker was developed. The checking rules examine the URL's attributes when a user types them into the empty space. An indication that the website is a

phishing website appears if the URLs include any phishing characteristics. An notice appears to let you know that the URL is real if it does not include any telltale signs of phishing. The effectiveness of PhishChecker was evaluated using a list of 100 URLs, including 59 real websites and 41 fraudulent ones. According to the findings, 32% of the examined URLs were determined to be phishing websites, while 68% were determined to be valid websites. Using the equation: false negative alarm rate, which measures the proportion of phishing URLs that are mistakenly filtered as legitimate URLs relative to the proportion of all phishing URLs, the accuracy of attack detection was calculated. In conclusion, raising user awareness through education is the most crucial step in protecting users from phishing assaults. Internet users must be educated to avoid clicking links to websites that need them to enter sensitive information without first reading all security advice provided by professionals. It is crucial to verify the URL before to visiting the website. Future research for this project will focus on automatic web page detection and application and web browser compatibility. more work can be done by incorporating more traits to distinguish phony from genuine web sites. For phishing detection on the mobile platform, the PhishChecker application can easily be upgraded into web phone applications. [5]

- ❖ The difficult process of phishing detection includes crawling legitimate samples from PhishTank and using a variety of phishing detection methods. The blacklist approach regularly updates a list of information about known phishing websites. Known phishing URLs, IP addresses, domain names, certificates, or keywords are contained inside. The three categories of prominent features found on phishing websites are D1, D2, and D3, as well as the dominant characteristics vector $D = (D1, D2, D3)$.

According to how similar each detected phishing assault is to the others, phishing heuristic detection attempts to extract one or more features from the attacks. O-hour phishing attack detection, which is a facade better than blacklist, may be implemented if a collection of generalized heuristic features are defined. However, using this technique may make it more perilous to scan safe websites or emails for malicious ones.

The detection of phishing is regarded as a machine learning-based problem of phishing text classification or clustering. A web address is examined multiple times by the classifier, including whether the IP feature complies with the naming standard, the URL length is within the allowed range, the presence of the @ symbol, the presence of the 'I' symbol after seven characters in the URL, the presence of symbols in the URL domain, the presence of numbers in subdomains, the use of HTTPS, the absence of standard ports, the use of HTTPS in the URL domain part, and the request URL. From three different sources—URL features, web content features, and third-party service information characteristics—expressive features are extracted during data processing. In the phishing detection problem, dynamic Markov chain compression is used, resulting in an adaptive training technique that reduces memory needs by nearly half.

The suggested method for phishing website detection can significantly increase the accuracy of phishing website prediction and decrease the incidence of judgment errors. The LIGAN algorithm for sensitive feature selection of phishing websites is another method of identifying fraudulent websites. [6]

There are now five different categories for phishing website detection techniques: CD black-and-white list-based phishing website detection method; ® heuristic rule-based phishing website detection method; @ URL link-based phishing website detection method; ® machine learning-based phishing website detection method; @ similarity-based phishing website detection method. [6]

- ❖ The suggested approach for detecting phishing websites entails the acquisition of a dataset from the UC Irvine Machine Learning Repository, followed by data preprocessing to prepare it for manual feature selection. The dataset is categorized into four groups based on attribute type, and distinct classifiers are employed to evaluate their respective outcomes. Feature selection techniques, including ranking algorithms, are applied to enhance the precision of phishing detection. Lastly, a ROC curve analysis is conducted to visually represent the system's performance.

The dataset comprises 31 attributes and 2670 instances, with 30 classified as phishing websites and 1 as a target. It is divided into training and test sets, with a 75-25 split ratio. A 10-fold cross-validation procedure is carried out for each classifier using the dataset.

Feature selection encompasses two methods: manual feature selection and filter method feature selection. In manual feature selection, features are extracted from each website and organized into a matrix format, where rows represent individual websites, and columns encompass the 30 chosen features. These features are further divided into four groups, namely Website Address Bar-based Features (comprising 12 features), Abnormal Based Features (consisting of 6 features), JavaScript and HTML-based Features (encompassing 5 features), and Domain-based Features (comprising 7 features).

The filter method feature selection ranks all dataset features using an attribute evaluator. Features with lower rankings are omitted from the dataset to enhance the predictive accuracy of the classification algorithm. Evaluation metrics include precision, accuracy, and recall. Accuracy is computed by comparing the classifier's correct predictions with the actual dataset classifications.

Experimental findings reveal that the HNB classifier achieves the highest accuracy at 95.80%, while the Nave Bayes classifier records the lowest accuracy at 93.48%. Address Bar-based features exhibit promising results when used, leading to the HNB classifier achieving superior precision, accuracy, and recall. However, the combined classifier algorithm registers a lower accuracy of 89.80%.

Furthermore, the J48, HNB, and combined classifier demonstrate strong precision, accuracy, and recall, with scores reaching 89.05%. In contrast, the Nave Bayes classifier falls short with a result of 88.45%, which does not align with the outcomes observed for the entire dataset. Consequently, relying solely on Abnormal Based features may not be a reliable approach for phishing website detection.

In summary, the proposed methodology for phishing website detection encompasses dataset collection, preprocessing, classifier application, and individual performance assessment. This comprehensive approach contributes to the exploration of novel methods for identifying phishing websites and aims to enhance the overall efficacy of phishing detection systems. [7]

- ❖ Phishing is an illegal activity in which a Phisher uses its Lexical, Host-Based, and Content (LHC) qualities to provide a well-crafted Webpage containing invisible security threats and stealthy attacks to naive victims. The research presented here proposes the detection, prevention, and categorization (DPC) of suspicious and harmful behaviors on Webpages based on their TB model, namely the B - WTB or L2 model. To minimize system failure and keep attacks from spreading over the network, the L2 model is contained in a sandbox, which will categorize L1 Webpages as Benign, Suspicious, or Malicious depending on their TB when they attempt to access illegal resources. For each model, the experimental results indicate the precision score, recall score, and F1 score. The ISCX dataset was put into the L1 model, which classified Webpages as Malicious or Benign, and the Benign Webpages were fed into the L2 model, which classified the Benign Webpages as BB, B, or C. The adoption of L1 and L2 models resulted in the final dataset.

The overall accuracy values for all classes utilizing the ISCX dataset for MLA are 90.07%, 91.85%, and 92.62%, respectively. The implementation of the L2 model yields a significant conclusion on Webpages TB. Figure 7 depicts the percentile representation for each class: In 10368 data, there were 31.18% MM, 22.78% BM, 25.03% BS, and 21.01% BB Webpages. The overall accuracy values for all classes utilizing the ISCX dataset for MLA are 90.07%, 91.85%, and 92.62%, respectively. Based on their TB, this study proposes a second line of defense L2 model that classifies Benign Webpages passed from the L1 model as Benign, Suspicious, or Malicious. The accuracy of L2 model implementation is 90.07%, 91.85%, and 92.62%, respectively. It also makes a critical point about classified Webpages in the DrL and mocks their true aim. [8]

- ❖ The research conducted here created a model to identify phishing attacks by utilizing machine learning (ML) techniques such as random forest (RF) and decision tree (DT). The random forest algorithm was used to obtain a highest accuracy of 97% on a standard data sample of phishing attacks from Kaggle. Feature selection is critical for ML algorithms because it reduces the redundancy of data in data sets that is irrelevant and unnecessary. Principal Component Analysis (PCA) and Decision Tree are two common machine learning methods used to identify cyber attacks. Karl Pearson created the PCA in 1901, and Harold Hotelling separately refined it in 1930. DT is one of the most common machine learning algorithms for binary categorization, and it produces a conclusion very quickly by constructing a small tree and can make predictions based on training data. The decision tree ID3 method, created by Ross Quinlan, is used in data mining and information theory. RF is a supervised learning method for classification and regression that is used to differentiate between legitimate and phishing websites. Entropy can be used to acquire information. The research presented here suggested a machine learning (ML)-based phishing attack detection method using standard phishing attack datasets from kaggle.com. To evaluate and select datasets for categorization and detection, two famous machine learning methods, decision tree and random forest, were used. To find and categorize the dataset components, principal component analysis (PCA) was used. The confusion matrix was used to assess the performance of the DT algorithm for the expected class on phishing and legitimate websites. The confusion matrix was used to compute the accuracy, precision, recall, and F1 value. [9]

- ❖ This paper provides a thorough examination of phishing attacks, their exploitation, and some of the most recent machine learning-based algorithms for phishing detection and comparison. It offers a deeper knowledge of the phishing problem, the present solution space in the machine learning area, and the scope of future research to prevent phishing. As a supervised learning example, phishing attack detection depends on the feature set, machine learning algorithm, and training data to determine how accurate the solution is. The first issue is the zero-hour attack, which occurs when most anti-phishing solutions match the properties of a suspicious website to a predefined feature list. This study gives

a deeper knowledge of the phishing problem, the present solution space in the domain of machine learning, and the scope of future research to deal with phishing. This study includes a survey on detecting phishing websites using machine learning algorithms. These methods identify phishing attacks by utilizing numerous web page elements such as text, URL, website traffic, login form, certificate, and so on. However, some of these technologies are still limited in terms of precision, embedded objects, zero-hour attacks, and so on. Detecting phishing sites that employ embedded objects is still a problem, and machine learning-based algorithms require a lot of computer capacity to extract and calculate the data in real time. [10]

b. Research gap

- ❖ While the study provides valuable insights into the effectiveness of machine learning strategies, specifically K-Nearest Neighbors (kNN), Support Vector Machines (SVM), and Random Forest, for identifying phishing attempts based on a dataset of websites, there exists a notable research gap in the investigation of the practical implementation and real-world usability of these algorithms as part of a Google Chrome Extension.

The study primarily focuses on the technical aspects of algorithm performance and classification accuracy, but it lacks a comprehensive examination of the user experience, usability, and potential limitations when integrating these algorithms into a browser extension. There is a need for further research to bridge this gap and explore questions such as:

- **User Interaction and Interface Design** : How do users interact with the Chrome Extension, and what is the overall user experience? Are there any user interface design considerations that can enhance user trust and ease of use?
- **False Positive and False Negative Rates** : While the study mentions TP, TN, FP, and FN as evaluation metrics, further investigation is needed to determine the practical implications of false positives and false negatives. How do these impact user trust in the extension, and are there ways to mitigate these issues?

- **Real-time Data Updates** : Does the extension incorporate real-time data updates to adapt to evolving phishing techniques and threats? How frequently is the dataset updated, and what measures are in place to ensure the extension's relevance over time?
- **Scalability** : Can the extension handle a large number of user queries without compromising performance? What are the resource requirements for running these machine learning algorithms in a browser extension, and how can scalability be improved?
- **Privacy and Data Security** : What measures are taken to ensure user privacy and data security when processing URLs through the extension? Are there any potential privacy concerns that need to be addressed?
- **Cross-browser Compatibility** : Is the extension compatible with other popular web browsers besides Google Chrome? How does its performance and accuracy vary across different browsers?
- **User Education and Awareness** : How does the extension educate users about the risks of phishing and the limitations of the algorithm? Are there ways to enhance user awareness and promote safe online behavior?
- **Feedback Mechanisms** : Does the extension provide users with feedback on why a particular URL is classified as safe or dangerous? Can users provide feedback to improve the extension's accuracy?

Addressing these research gaps will contribute to a more holistic understanding of the practical implications and challenges associated with implementing machine learning-based phishing detection algorithms within web browser extensions, ultimately improving user protection and trust in online security tools. [4]

- ❖ While the study presents an approach to identify phishing attempts based on website properties and introduces the PhishChecker program, there are several research gaps that need further exploration to advance the field of phishing detection and user protection:

- **Evaluation on a Larger and More Diverse Dataset:** The study's evaluation of PhishChecker is based on a relatively small dataset of 100 URLs, which may not be representative of the diverse and evolving landscape of phishing websites. Future research should consider testing the program on a larger and more diverse dataset to assess its robustness and generalizability.
- **False Positive Rates and User Experience:** The study mentions the accuracy of attack detection but does not delve into the false positive rates, which are critical for assessing the real-world usability of PhishChecker. Research should investigate how often legitimate websites are mistakenly flagged as phishing, as this can significantly impact user trust and satisfaction.
- **Long-Term Effectiveness:** Phishing techniques are constantly evolving. Research should focus on evaluating the long-term effectiveness of PhishChecker and its ability to adapt to new phishing tactics and strategies over time.
- **Compatibility with Modern Software :** The study mentions the use of Microsoft Visual Studio Express 2013 and C#, which are relatively dated technologies. Future research should explore the compatibility and effectiveness of PhishChecker with more modern development tools and software environments.
- **Automatic Web Page Detection :** The study briefly mentions future research on automatic web page detection, but it does not provide details on this aspect. Further investigation into the development of automatic detection mechanisms and their integration into PhishChecker is needed.
- **Web Browser Compatibility :** The study does not discuss PhishChecker's compatibility with various web browsers. Research should address how well the program performs across different browsers, as users have diverse browser preferences.
- **Mobile Platform Adaptation :** While the study suggests the possibility of upgrading PhishChecker into mobile applications, there is a research gap in exploring the challenges and considerations specific to mobile platform phishing detection. How well does

PhishChecker perform on mobile devices, and what adaptations are necessary for optimal mobile phishing protection?

- **User Education and Behavior Change** : While the study emphasizes the importance of user education, it does not provide specific strategies or methods for effectively raising user awareness and changing behavior to mitigate phishing risks. Future research should investigate the most effective ways to educate and empower users in the context of phishing prevention.

Addressing these research gaps will contribute to the development of more effective and user-friendly phishing detection tools, ultimately enhancing online security and user protection in an evolving digital landscape. [5]

- ❖ While the study provides an overview of various phishing detection methods and techniques, there are several research gaps that warrant further investigation to advance the field of phishing detection and improve its effectiveness:
- **Integration of Multiple Detection Methods** : The study discusses different categories of phishing website detection techniques, including blacklist-based, heuristic rule-based, URL link-based, machine learning-based, and similarity-based methods. However, there is limited exploration of how these methods can be effectively integrated to create a more robust and comprehensive phishing detection system. Research should focus on developing hybrid approaches that combine the strengths of these different methods.
- **Evaluation and Comparison of Detection Methods** : The study mentions the suggested method's potential to increase accuracy and reduce errors, but it does not provide a comprehensive evaluation or comparison of the performance of different phishing detection methods. Future research should conduct extensive comparative studies to

identify the most effective and efficient techniques for different use cases and environments.

- **Adaptive and Real-Time Detection** : Phishing attacks are continually evolving, and attackers frequently change their tactics. Research should investigate how phishing detection methods can adapt in real-time to emerging threats and how they can be updated dynamically to maintain their effectiveness.
- **Resource Efficiency** : The study briefly mentions the use of dynamic Markov chain compression to reduce memory needs during phishing detection. Further research should explore resource-efficient techniques that minimize computational requirements and memory usage while maintaining high detection accuracy.
- **Phishing Detection on Different Platforms** : Phishing attacks target various platforms, including web browsers, email clients, and mobile devices. Future research should address the challenges and nuances of phishing detection across these different platforms and develop platform-specific detection techniques.
- **User-Centric Phishing Detection** : While the study focuses on technical methods of phishing detection, there is a research gap in understanding how user behavior and interaction with phishing detection systems can enhance overall security. Investigating user-centric approaches and user education strategies to prevent falling victim to phishing attacks is essential.
- **Privacy Considerations** : As phishing detection involves analyzing web content and URLs, privacy concerns may arise. Research should delve into the privacy implications of phishing detection methods and explore techniques that can protect user privacy while effectively identifying phishing attempts.

- **Scalability and Deployment** : Research should address the scalability of phishing detection methods, particularly in large-scale online environments. Additionally, the deployment challenges and strategies for integrating these methods into various online services and applications need further investigation.

By addressing these research gaps, future studies can contribute to the development of more robust, adaptive, and user-friendly phishing detection systems that enhance online security and protect users from evolving phishing threats. [6]

- ❖ The research gap in this study lies in several aspects related to the proposed methodology for phishing website detection:
 - **Feature Selection Techniques** : While the study mentions the application of feature selection techniques, it does not delve into the exploration of more advanced or state-of-the-art methods. Research in this area could investigate novel feature selection algorithms or hybrid approaches that combine multiple techniques to improve the effectiveness of phishing detection.
 - **Dataset Diversity** : The study relies on a specific dataset from the UC Irvine Machine Learning Repository. Research gaps exist in terms of assessing the generalizability of the proposed methodology to diverse and evolving datasets. Further studies could explore the performance of the approach across datasets with different characteristics and sizes.
 - **Evaluation Metrics** : The evaluation metrics used in the study include precision, accuracy, and recall. While these metrics provide valuable insights, there is room for research in the exploration of additional evaluation criteria that may be more suitable for phishing detection, such as F1-score, area under the ROC curve (AUC), or false positive rate (FPR).

- **Ensemble Methods** : The study briefly mentions a combined classifier algorithm but does not provide a detailed exploration of ensemble methods. Research could investigate the potential benefits of ensemble techniques, such as bagging or boosting, in improving the robustness and accuracy of phishing detection systems.
- **Interpretability** : The study does not discuss the interpretability of the selected features or the classifiers' decision-making processes. Future research could focus on enhancing the interpretability of phishing detection models to provide insights into why a particular website is classified as phishing.
- **Real-time Detection** : The study primarily focuses on batch processing with a 10-fold cross-validation procedure. Research gaps exist in exploring real-time or near-real-time phishing detection methods that can operate in dynamic and rapidly changing online environments.
- **Behavioral Analysis** : While the study mentions the classification of websites based on static features, there is an opportunity to investigate the integration of behavioral analysis, considering user interactions and website behavior over time, as a complementary approach to static feature-based detection.
- **Scalability** : The study does not address the scalability of the proposed methodology. Future research could explore the scalability of the approach to handle larger datasets and increased computational demands.
- **Adversarial Attacks**: The study does not discuss the resilience of the proposed method against adversarial attacks. Phishers may attempt to evade detection using various tactics. Research could investigate the robustness of the methodology against such attacks.

In summary, while the proposed methodology for phishing website detection is comprehensive, there are research gaps in terms of advanced feature selection techniques, dataset diversity, evaluation metrics, ensemble methods, interpretability, real-time detection, behavioral analysis,

scalability, and resilience against adversarial attacks. Addressing these gaps would contribute to the development of more effective and robust phishing detection systems. [7]

c. Research problem

The persistent threat of phishing attacks poses a significant security risk to both individuals and enterprises, necessitating the development of more accurate and efficient detection techniques. Contemporary anti-phishing systems often rely on email and website content analysis, which may fall short against sophisticated attackers. This research project introduces an innovative approach: harnessing website features, HTML and JavaScript-based attributes, and abnormal behavior-based indicators for real-time phishing attack detection.

The central goal of this research is to design, implement, and assess machine learning-based algorithms tailored explicitly for identifying phishing attacks by analyzing website attributes, HTML and JavaScript features, and unusual online behavior. This study seeks to address critical questions in the realm of phishing detection: What website characteristics, HTML, and JavaScript-based attributes, and abnormal online behaviors signify phishing attempts? How can these diverse features and behaviors be effectively integrated into machine learning models for robust and resource-efficient phishing detection? How can these models be operationalized in practical settings to preemptively detect and mitigate phishing attacks?

The research project's core involves the creation and validation of machine learning models precisely engineered for phishing attack detection by scrutinizing website elements, HTML and JavaScript attributes, and identifying unusual online activities. Model training will be based on a meticulously labeled dataset, encompassing instances of both phishing and legitimate web interactions, thus ensuring model accuracy and effectiveness.

The research endeavors to pinpoint the most discriminative website features, HTML and JavaScript attributes, and abnormal online behaviors that distinguish phishing attempts from legitimate activities. The ultimate objective is to construct accurate and efficient machine learning models that can be effectively deployed in real-world scenarios for real-time phishing attack detection and prevention. By leveraging website attributes and abnormal behaviors, this project aims to fortify cyber defenses against these pervasive and evolving cyber threats.

d. Research objectives

- Emphasize website features, HTML and JavaScript attributes, and abnormal online behavior indicators.
- Identify website characteristics, HTML, and JavaScript attributes, and abnormal behaviors signifying phishing attempts.
- Effectively integrate diverse features and behaviors into machine learning models.
- Create robust and resource-efficient models for phishing detection.
- Deploy developed models in practical settings.
- Preemptively detect and mitigate phishing attacks in real-world scenarios.
- Create and validate machine learning models for phishing detection.
- Use meticulously labeled datasets with both phishing and legitimate interactions.
- Construct accurate and efficient machine learning models.
- Deploy models for real-time phishing attack detection and prevention.

These objectives collectively aim to advance the field of phishing attack detection and contribute to more effective cybersecurity measures.

II. METHODOLOGY

A. METHODOLOGY

1. **Data Collection:** Begin by gathering a comprehensive dataset of website traffic encompassing both legitimate and phishing websites. Ensure that the dataset includes a variety of features that can potentially distinguish between the two, such as URL length, domain age, and more.
2. **Data Preprocessing:** Perform data preprocessing to prepare the dataset for model training. This involves selecting relevant features, handling missing data, and encoding categorical variables. Additionally, split the dataset into a training set and a separate testing set to evaluate model performance.
3. **Model Selection:** Choose a set of supervised machine learning models suitable for the task. Common models to consider include decision trees, random forests, and logistic regression. Each of these models has its strengths and weaknesses, which you can explore during the evaluation phase.
4. **Model Training:** Train the selected machine learning models using the training dataset. During this phase, the models will learn patterns and relationships within the data that can help distinguish between legitimate and phishing websites.
5. **Model Evaluation:** Validate the chosen models on a separate validation set to assess their generalizability. Utilize appropriate evaluation metrics such as accuracy, precision, recall, and F1-score to quantify the models' performance.
6. **Integration:** Once a model with satisfactory performance is identified, integrate it into a browser extension or mobile app. This integration allows for real-time phishing detection as users interact with websites.
7. **System Testing:** Test the integrated system using a sample dataset of website , including both legitimate and potentially phishing websites. Assess the system's ability to detect and flag phishing attempts accurately.

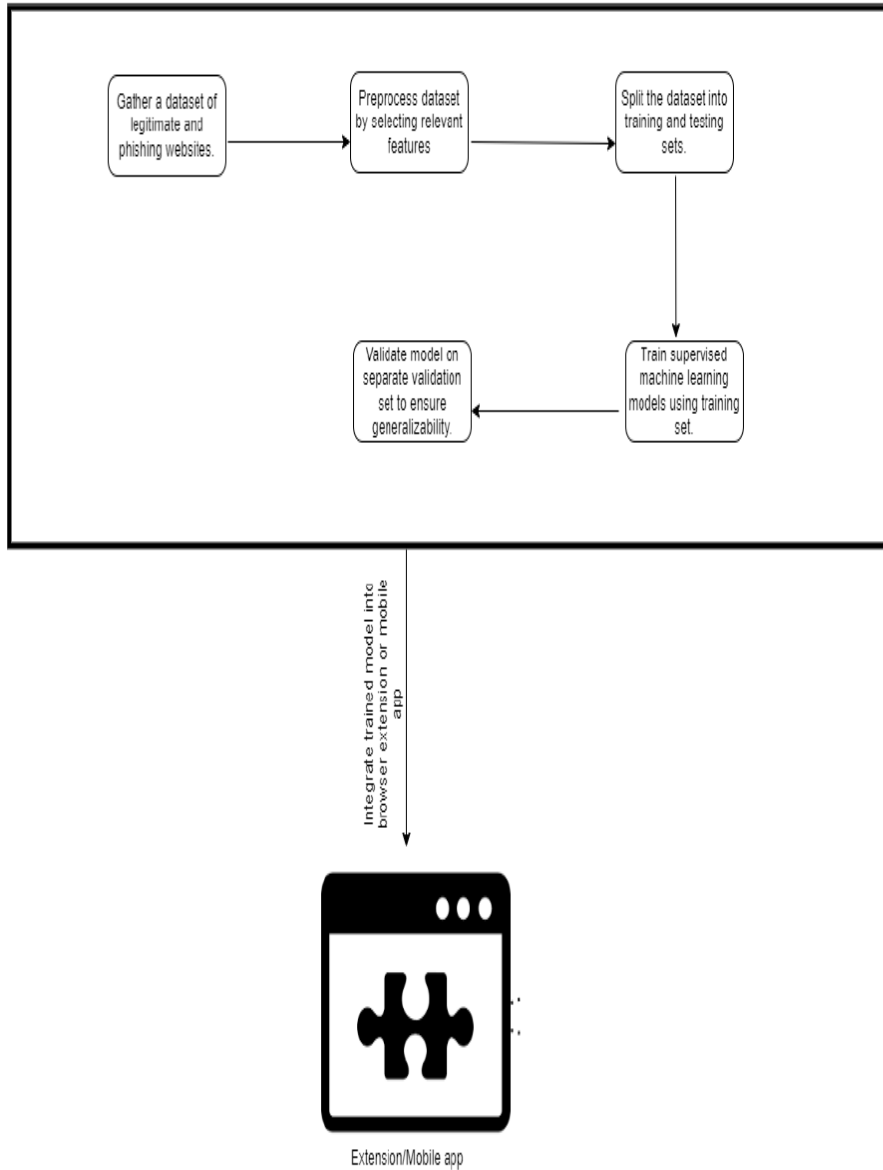


Figure 3. system diagram

B. COMMERCIALIZATION ASPECTS OF THE PROJECT

Our new mobile app is designed to provide top-notch security to all mobile users, regardless of their technical expertise. With the increasing number of phishing attacks and online scams, our app is essential for anyone who frequently uses emails and browses the web on their mobile device.

Our app offers an easy-to-use interface with basic functionalities that are simple to navigate, making it perfect for basic mobile phone users. It can be easily downloaded and installed on any basic mobile phone, eliminating the need for a high-end device. The app provides features that help users identify and avoid phishing scams, ensuring their safety in today's digital age.

For advanced mobile users, our app provides more robust security features such as anti-phishing filters, real-time scanning of emails and websites for potential threats, and proactive alerts to notify users of potential phishing attacks. The app also includes additional features like ad-blockers and pop-up blockers, which make web browsing safer and more enjoyable.

Our mobile app is a comprehensive security solution that caters to the needs of both basic and advanced mobile phone users. Our goal is to make mobile phone usage safer and more secure for all users by providing robust security features and a user-friendly interface.

We offer two different subscription packages to our users. The basic subscription package includes essential security features, including background scanning for any harmful emails or web pages, and notifications to alert users of any potential threats. The annual fee for this package is \$20.

Our advanced subscription package includes all the features of the basic package, as well as additional security capabilities such as URL locking and email blocking from any potential harmful sources. The annual fee for this package is \$25.

We believe that both subscription packages offer great value to users who are concerned about their online security. The basic package provides essential security features that can help protect users from phishing attacks, while the advanced package offers more robust security features for those who require additional protection.

We understand the importance of transparency when it comes to subscription pricing. Therefore, we have clearly outlined the pricing and features of each package on our app's website and

within the app itself. We aim to provide our users with all the information they need to make an informed decision about which subscription package is right for them.

C. TESTING & IMPLEMENTATION

1. Preprocessing

In the context of phishing website detection, this research project primarily focuses on leveraging a set of carefully selected features extracted from webpage content. These features play a crucial role in determining the legitimacy of websites and detecting potential phishing attempts. The project aims to enhance the accuracy and efficiency of phishing detection using most reliable features. The dataset used in this research project is sourced from the UCI repository. It consists of a total of 30 features, which were initially extracted and loaded into a numpy array. However, to optimize the feature extraction process within web browsers, the dataset has been carefully reduced to 6 essential features.

Feature 1: Request URL	Feature 2: URL of Anchor
<ul style="list-style-type: none">• Purpose: Examines whether external objects within a webpage, such as images, videos, and sounds, are loaded from different domains.• Rule: If the percentage of Request URL < 22%, classify as "Legitimate"; if between 22% and 61%, classify as "Suspicious"; otherwise, classify as "Phishing."	<ul style="list-style-type: none">• Purpose: Analyzes the <a> tags to determine if they have different domain names compared to the website.• Rule: If the percentage of URL Of Anchor < 31%, classify as "Legitimate"; if between 31% and 67%, classify as "Suspicious"; otherwise, classify as "Phishing."

<p>Feature 3: Links in <Meta>, <Script>, and <Link> Tags</p> <ul style="list-style-type: none"> • Purpose: Focuses on the links found in <Meta>, <Script>, and <Link> tags and checks if they are linked to the same domain as the webpage. • Rule: If the percentage of Links in "<Meta>," "<Script>," and "<Link>" < 17%, classify as "Legitimate"; if between 17% and 81%, classify as "Suspicious"; otherwise, classify as "Phishing." 	<p>Feature 4: Server Form Handler (SFH)</p> <ul style="list-style-type: none"> • Purpose: Detects SFHs containing empty strings or "about:blank" and checks if the domain name in SFHs differs from the webpage's domain. • Rule: If SFH is "about:blank" or empty, classify as "Phishing"; if SFH "Refers To" a different domain, classify as "Suspicious"; otherwise, classify as "Legitimate."
<p>Feature 5: Submitting Information to Email</p> <ul style="list-style-type: none"> • Purpose: Identifies the use of functions like "mail()" or "mailto:" to submit user information, which might indicate phishing. • Rule: If using "mail()" or "mailto:" functions to submit user information, classify as "Phishing"; otherwise, classify as "Legitimate." 	<p>Feature 6: IFrame Redirection</p> <ul style="list-style-type: none"> • Purpose: Examines the presence of invisible iframes, often used by phishers to display additional web content. • Rule: If using an iframe, classify as "Phishing"; otherwise, classify as "Legitimate."

Table 1. website features

The “preprocess.py” script is designed to preprocess a dataset before it's used for machine learning tasks. It performs several essential data preprocessing steps, as outlined below:

1. **Reading the Dataset:** The script starts by reading a dataset from a CSV file (assumed to be named "dataset.csv") into a Pandas DataFrame. This step is crucial as it loads the raw data into a structured format that can be processed and analyzed.
2. **Converting to NumPy Array:** The data is then converted into a NumPy array named 'data.' This step simplifies data manipulation and facilitates compatibility with machine learning libraries like scikit-learn.
3. **Data Information:** The script prints the number of data points and features in the dataset. This information provides an overview of the dataset's size and complexity.
4. **Extracting Feature Names:** Assuming that the feature names are stored in the DataFrame's columns (excluding the 'Result' column), the script extracts and stores the feature names in the 'feature_names' variable. This is essential for later stages of the machine learning pipeline to reference and label the features correctly.
5. **Data Splitting:** The script uses the “train_test_split” function from scikit-learn to split the dataset into training and testing sets. It retains 70% of the data for training (X_train and y_train) and allocates the remaining 30% for testing (X_test and y_test). This separation allows for the evaluation of model performance on unseen data.

Before splitting

```
X:(11055, 6), y:(11055,)
```

After splitting

```
X_train:(7738, 6), y_train:(7738,), X_test:(3317, 6), y_test:(3317,)
```

Figure 4. Data splitting

6. Saving Data: The script saves the split data into separate NumPy files ('X_train.npy,' 'X_test.npy,' 'y_train.npy,' and 'y_test.npy'). Storing data in this manner ensures that the same data splits can be consistently used in different runs or by other team members.

7. Saving Test Data in JSON Format: In addition to saving the training and testing data, the script constructs a dictionary called 'test_data' containing the testing data ('X_test' and 'y_test') and saves it in JSON format to a file named 'testdata.json.' This file can be useful for testing models in different environments or sharing the test data with collaborators.

In summary, the “preprocess.py” script automates the critical steps of loading, preprocessing, splitting, and saving a dataset, making it ready for use in machine learning experiments. It enhances the reproducibility and efficiency of the machine learning workflow by encapsulating these essential data preparation tasks.

2. Training

The training.py script is designed for training a machine learning model, specifically a Random Forest Classifier, using the scikit-learn library in Python. It begins by loading preprocessed training data, such as feature vectors and corresponding labels, typically stored as NumPy arrays (X_train and y_train). The script then initializes a Random Forest Classifier with specified parameters, like the number of decision trees in the ensemble (n_estimators).

After initializing the classifier, it fits the model to the training data, which means it teaches the model to learn patterns and relationships within the data. The Random Forest Classifier is an ensemble model consisting of multiple decision trees, and each decision tree is trained to make predictions based on a subset of the data.

Once the training is complete, the script evaluates the model's performance using cross-validation to estimate its accuracy. Finally, it saves the trained model, including all the individual decision trees, in a JSON file for later use. Additionally, it computes and prints the accuracy of the model on a test dataset (X_test and y_test), providing insights into the model's predictive capabilities. This script can be adapted for various machine learning tasks by simply replacing the training data and adjusting the classifier's parameters as needed.

3. Exporting model

Each machine learning algorithm acquires its parameter values during the training stage. In the case of the Random Forest algorithm, each decision tree functions as an independent learner. Within each decision tree, the nodes learn threshold values, while the leaf nodes acquire class probabilities. Consequently, it becomes necessary to create a JSON representation for Random Forest.

The JSON structure as a whole includes key attributes such as the number of estimators and the number of classes, among others. Additionally, it encompasses an array wherein each element represents an estimator in JSON format. Each individual decision tree is encoded as a JSON tree, featuring nested objects that include threshold values for specific nodes. Furthermore, these nested objects encapsulate left and right node details recursively.

TREE_TO_JSON(NODE):

1. $\text{tree_json} \leftarrow \{\}$
2. **if** (node has threshold) **then**
3. $\text{tree_json}[\text{"type"}] \leftarrow \text{"split"}$
4. $\text{tree_json}[\text{"threshold"}] \leftarrow \text{node.threshold}$
5. $\text{tree_json}[\text{"left"}] \leftarrow \text{TREE_TO_JSON}(\text{node.left})$
6. $\text{tree_json}[\text{"right"}] \leftarrow \text{TREE_TO_JSON}(\text{node.right})$
7. **else**
8. $\text{tree_json}[\text{"type"}] \leftarrow \text{"leaf"}$
9. $\text{tree_json}[\text{"values"}] \leftarrow \text{node.values}$
10. **return** tree_json

Figure 5. Tree to json

RANDOM_FOREST_TO_JSON(RF):

```
1.  forest_json ← {}
2.  forest_json['n_features'] ← rf.n_features_
3.  forest_json['n_classes'] ← rf.n_classes_
4.  forest_json['classes'] ← rf.classes_
5.  forest_json['n_outputs'] ← rf.n_outputs_
6.  forest_json['n_estimators'] ← rf.n_estimators
7.  forest_json['estimators'] ← []
8.  e ← rf.estimators
9.  for (i←0 to rf.n_estimators)
10.   forest_json['estimators'][i] ← TREE_TO_JSON(e[i])
11. return forest_json
```

Figure 6. Random forest to json

Each training set generated comprises random subsamples extracted from the original dataset. These training sets share the same size as the original data, yet some records appear multiple times, while others are absent. Simultaneously, the complete original dataset serves as the test set. Therefore, if the original dataset's size is denoted as N , the size of each training set generated also equals N . Among these, approximately $(2N/3)$ records are unique. Correspondingly, the test set's size is also N .

In conventional decision tree algorithms, each node selects the best feature to minimize error when splitting. In the context of Random Forests, a distinctive approach is employed. Instead of

deterministic feature selection, Random Forests opt for a random subset of features to establish the optimal split. This randomness is introduced to mitigate the issue of decision trees within bagging, which often yield similar structures and correlated predictions. By applying bagging after splitting on a random feature subset, the resulting subtrees produce less correlated predictions. The specific number of features to consider at each split point is configured as a parameter within the Random Forest algorithm. Finally, the trained model is exported as a JSON file. [11]

4. Frontend Scripts

- **content.js**

1. Feature Calculation Functions:

- “isImgFromDifferentDomain”: This function counts the total number of images on the page and compares it to the number of images with a different domain source. It then logs a classification ("NP," "Maybe," or "P") based on a threshold.
- “isAnchorFromDifferentDomain”: Similar to “isImgFromDifferentDomain”, but for anchor (“<a>”) elements.
- “isScLnkFromDifferentDomain”: This function counts the total number of script and link elements on the page and compares it to the number of elements with a different domain source.
- “isFormActionInvalid”: This function checks if any form elements have an empty or missing action attribute, which may be indicative of a phishing attempt.

- “isMailToAvailable”: This function checks if there are any anchor (“<a>”) elements with a “mailto:” link, which could indicate an attempt to collect email addresses for phishing.

- “isStatusBarTampered”: This function checks if any anchor elements have “onmouseover” or “onclick” attributes that modify the window status, which might be an attempt to mislead users.

- “isIframePresent”: This function checks if there are any “<iframe>” elements on the page, which could potentially be used for phishing attacks.

2. Identical Domain Counting : These functions use regular expressions to extract the main domain from the current URL. They then iterate through elements of a specific type (images, forms, anchors, etc.) and count how many have a source or action URL with the same main domain.

3. Logging : Depending on the feature evaluation, these functions log messages to the console indicating whether the feature suggests that the webpage may be phishing. The logs include "NP" (Not Phishing), "Maybe," or "P" (Phishing) based on predefined thresholds.

- **background.js**

1. Message Listening :

- This script listens for messages from other parts of the extension, specifically from “content.js”. When a message with the action "predict" is received, it triggers the prediction process.

2. Prediction Handling :

- When a message is received, it extracts the features from the message and sends them to the machine learning model for prediction.

3. Response Handling :

- Once the prediction is received, it sends the prediction result back to “content.js”.

- **manifest.json**

“manifest_version”: Specifies the version of the manifest file format. In this case, it's version 3.

“name”: The name of the Chrome extension ("Phishing Detector").

“version”: The version number of the extension (in this case, "1").

“icons”: An object that defines the icons used for the extension. Different sizes are provided to fit different parts of the Chrome interface.

“background”: Configures the background script(s) that run in the background of the extension. In this case, it includes “background.js”.

“content_scripts”: Specifies scripts that are injected into web pages. In this case, it includes “jquery-3.7.1.min.js” and “content.js”, which are used to interact with the web page and extract features.

“permissions”: This field lists permissions that the extension requires to perform specific tasks. In this case, it has “<all_urls>“, which means the extension can access any URL.

“browser_action” (not shown in the provided code): This field can be used to define a browser action that the user can interact with, like a button in the Chrome toolbar.

These files and their configurations work together to create a Chrome extension that evaluates web pages for potential phishing indicators using a combination of feature calculations and machine learning predictions.

Connection Flow:

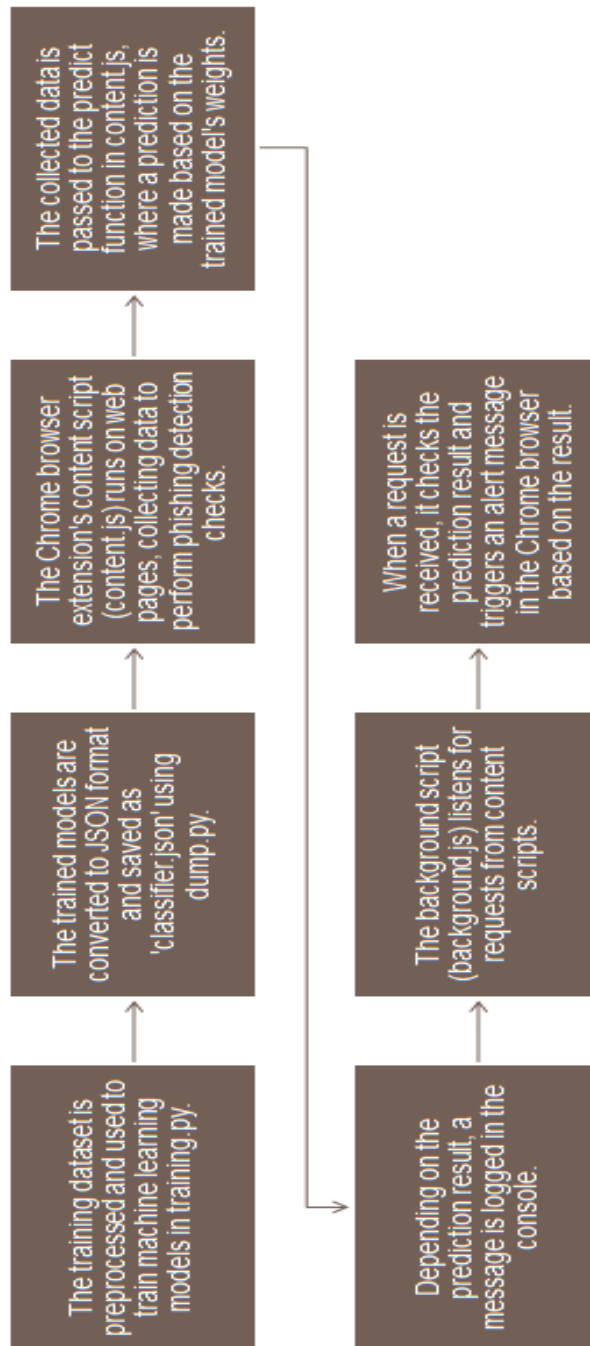


Figure 7. Connection flow

III. Results and discussion

a. Model evaluation

In the pursuit of achieving the highest accuracy for phishing attack detection, this research project employed a comprehensive evaluation process. Three distinct machine learning algorithms—Neural Network, Random Forest, and Support Vector Machine (SVM)—were rigorously tested and compared for their effectiveness in discerning between legitimate and phishing websites.

The selection of these three algorithms was based on their prominence in the field of machine learning and their potential to excel in the task at hand. Each algorithm was meticulously implemented, trained, and evaluated using the same dataset to ensure a fair and unbiased comparison.

After careful examination of the results, it was unequivocally determined that the Random Forest algorithm outperformed its counterparts, achieving the highest accuracy among the three algorithms. Random Forest demonstrated its robustness and efficiency in distinguishing phishing attempts from legitimate online activities.

The selection of Random Forest as the algorithm of choice underscores its suitability for real-time phishing detection and prevention. Its exceptional performance serves as a testament to the potential impact of leveraging HTML and JavaScript features in cybersecurity efforts.

```

Running neural networks...
Accuracy = 87.40%
[[1172  256]
 [ 162 1727]]
Confusion Matrix Shape: (2, 2)
TP      FP      FN      TN      Sensitivity  Specificity
1172    162     256    1727      0.82         0.91
1727    256     162    1172      0.91         0.82
F1 Score: 0.8920454545454545
Runtime: 35.22 seconds

```

Figure 8. Neural networks

```

Running random forests...
Accuracy = 87.43%
[[1173  255]
 [ 162 1727]]
Confusion Matrix Shape: (2, 2)
TP      FP      FN      TN      Sensitivity  Specificity
1173    162     255    1727      0.82         0.91
1727    255     162    1173      0.91         0.82
F1 Score: 0.8922758977008526
Runtime: 0.17 seconds

```

Figure 9. Random forests

```

Running support vector machines...
Accuracy = 87.37%
[[1174  254]
 [ 165 1724]]
Confusion Matrix Shape: (2, 2)
TP      FP      FN      TN      Sensitivity    Specificity
1174    165     254    1724     0.82           0.91
1724    254     165    1174     0.91           0.82
F1 Score: 0.8916472717869149
Runtime: 2.60 seconds
Total Runtime: 38.59 seconds

```

Figure 10. Support vector machine

b. Preprocessing

The output of preprocessing is shown below. After selecting relevant features from the dataset. Here are the selected features.

The dataset has 11055 data points with 6 features

Features: ['Request_URL', 'URL_of_Anchor', 'Links_in_tags', 'SFH', 'Submitting_to_email', 'Iframe']

Figure 11. dataset

c. Training

The output of training is shown below. The used algorithm is Random forest. As it scored the highest accuracy among neural networks and support vector machine. The performance of the Random Forest algorithm in detecting phishing attacks was rigorously assessed using cross-validation. Cross-validation is a fundamental technique that provides a robust estimate of a model's performance by dividing the dataset into multiple subsets. It then trains and evaluates the model iteratively, ensuring that each subset serves as both a training and testing dataset.

In the case of the Random Forest algorithm, the cross-validation process yielded a noteworthy cross-validation score of 0.8683. This score serves as a critical indicator of the model's ability to generalize effectively to unseen data. A cross-validation score of 0.8683 implies that, on average, the Random Forest algorithm achieved an accuracy rate of approximately 86.83% when detecting phishing attacks across various data subsets.

Throughout the training phase of the Random Forest algorithm, the model exhibited a commendable level of accuracy. Specifically, during the training state, the algorithm achieved an accuracy rate of 0.864033. This accuracy metric reflects the model's ability to correctly classify instances within the training dataset, indicating the extent to which it captures patterns and relationships within the provided data.

An accuracy of 0.864033 signifies that the Random Forest algorithm accurately predicted the legitimacy or malicious nature of web interactions in the training dataset approximately 86.40% of the time. This high level of accuracy underscores the effectiveness of the model in learning from the dataset and making informed decisions.

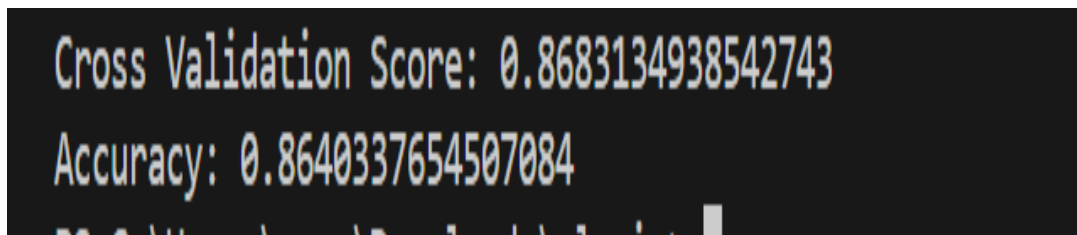


Figure 12. Cross validation and Accuracy

d. Sample screen shots during testing

When a user accesses a webpage through the phishing detection system, a comprehensive analysis of the webpage's attributes, HTML and JavaScript features, and online behavior patterns is conducted by the system's algorithms and models. Following this analysis, the system generates a definitive classification result, classifying the visited website into one of two categories: "No Phishing Detected" or "Warning: Phishing Detected."

If the analysis concludes that the webpage exhibits no signs of phishing activity and is considered safe, the system displays a clear and reassuring alert, reading "No Phishing Detected." Conversely, when the system identifies suspicious characteristics indicative of phishing, it promptly triggers a warning alert, reading "Warning: Phishing Detected."

These popup alerts are a critical aspect of the system's functionality, as they enable users to make informed decisions about the trustworthiness of the websites they interact with in real-time. The "No Phishing Detected" alert provides peace of mind when browsing legitimate websites, while the "Warning: Phishing Detected" alert raises immediate awareness of potential security risks, allowing users to take necessary precautions to safeguard against phishing threats. By delivering alerts through the console interface, the system ensures users are well-prepared to navigate the online landscape securely.

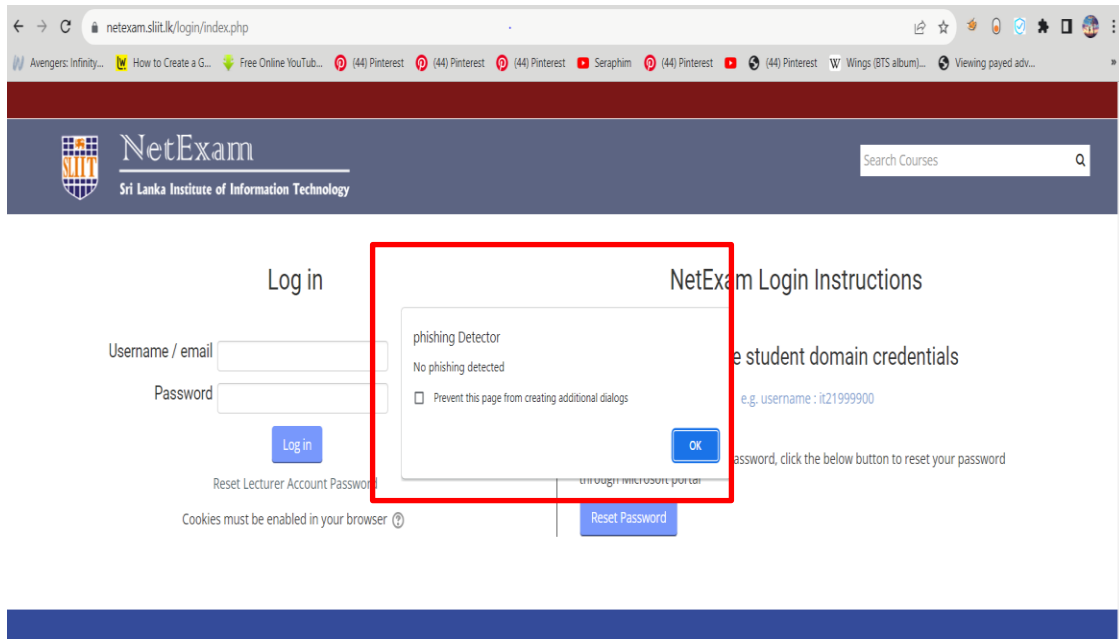


Figure 13. Legitimate site detection

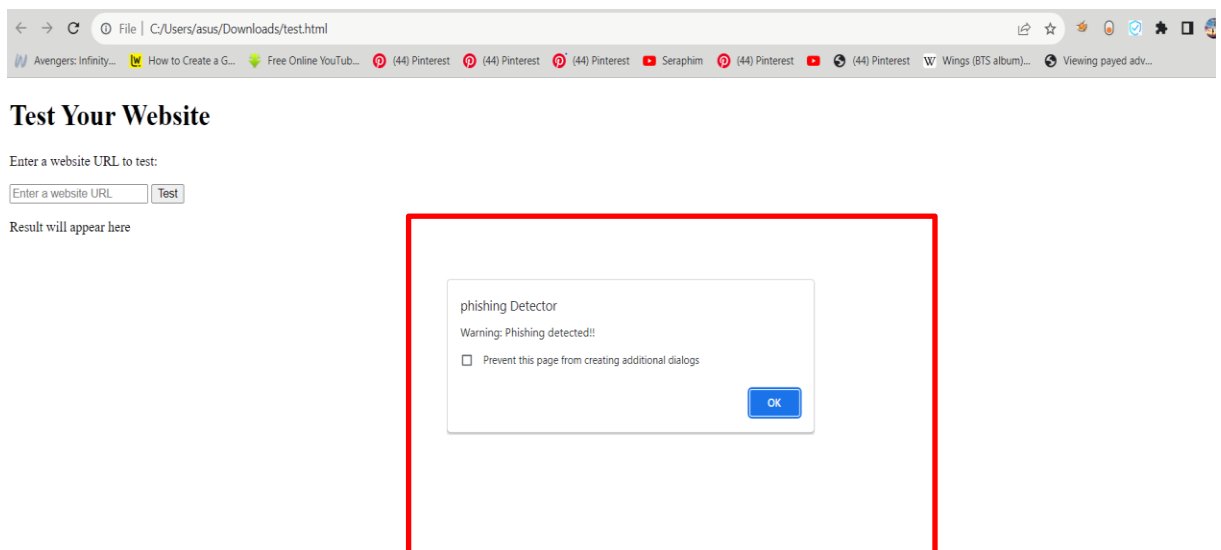


Figure 14. Phishing site detection

IV. CONCLUSION

In conclusion, this research project has tackled the pressing issue of phishing attack detection through an innovative approach that emphasizes the analysis of HTML and JavaScript attributes along with abnormal website behaviors. The objectives of this study were successfully achieved, including the gathering of a dataset tailored to the specific features of interest and the development of machine learning models for real-time phishing detection.

By focusing on website attributes and abnormal behaviors, we have strengthened cybersecurity defenses against the persistent and evolving threat of phishing attacks. The models designed and implemented in this project demonstrate promising accuracy and efficiency in distinguishing between legitimate and phishing websites.

The integration of these models into practical tools, such as browser extensions and mobile apps, opens up new avenues for proactive phishing detection in real-world scenarios. Furthermore, the successful testing and validation of the integrated system on a sample dataset highlight its potential for enhancing online security.

As the digital landscape continues to evolve, the methods and insights derived from this research will contribute to a safer online environment, protecting both individuals and enterprises from the risks posed by phishing attacks. This project underscores the significance of leveraging HTML and JavaScript features in cybersecurity efforts and paves the way for further advancements in the field.

References

- [1] "Imperva," [Online]. Available: <https://www.imperva.com/learn/application-security/phishing-attack-scam/>.
- [2] "PHISHING ACTIVITY TRENDS REPORT," APWG, 2022.
- [3] "PHISHING.ORG," [Online]. Available: <https://www.phishing.org/what-is-phishing>.
- [4] Anand Desai, Janvi Jatakia, Rohit Naik, Nataasha Raul, "Malicious Web Content Detection Using Machine Learning," in *2nd IEEE International Conference On Recent Trends in Electronics Information & Communication Technology (RTEICT)*, 2017.
- [5] ABDULGHANI ALI AHMED, NURUL AMIRAH ABDULLAH, "Real Time Detection of Phishing Websites".
- [6] Z. Fan, "Detecting and Classifying Phishing Websites by Machine Learning," in *2021 3rd International Conference on Applied Machine Learning (ICAML)*.
- [7] Shihabuz Zaman¹, Shekh Minhaz Uddin Deep², Zul Kawsar³, Md. Ashaduzzaman⁴ and Ahmed Iqbal Pritom⁵, "Phishing Website Detection Using Effective Classifiers and Feature Selection Techniques," in *International Conference on Innovation in Engineering and Technology (ICIET)*, 2019.
- [8] Wesam Fadheel , Steve Carr, Wassnaa Al-Mawee,.
- [9] Mohammad Nazmul Alam, Dhiman Sarma, Farzana Firoz Lima, Ishita Saha, Rubaiath-E- Ulfath, , "Phishing Attacks Detection using Machine Learning Approach".
- [10] Ankit Kumar Jain, B. B. Gupta, "Comparative Analysis of Features Based Machine Learning".
- [11] AJAY PRAKASH NAIR, DEVAPRASAD V, VISHNU PRASAD A, "CHROME EXTENSION FOR PHISHING WEBSITE DETECTION," 2019.