API Driven URL Phishing Detection using Machine Learning

Cibiananth K R
Department of Computer Science and Engineering
Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
Avadi, Chennai, Tamil Nadu, India
vtu22830@veltech.edu.in

Mouleeswaran S
Department of Computer Science and Engineering
Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
Avadi, Chennai, Tamil Nadu, India
vtu22826@veltech.edu.in

Dr. Natesan
Department of Computer Science and Engineering
Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
Avadi, Chennai, Tamil Nadu, India
drnatesan@veltech.edu.in

***Abstract*—Phishing attacks, which employ malicious URLs to obtain private data including usernames, passwords, and financial information, continue to be a major concern to internet users. Traditional detection methods based on static blacklists and fixed rules are frequently inadequate to offer dependable security since phishing websites are created and removed so quickly. This project offers a Hybrid URL Phishing Detection System that integrates real-time threat information, machine learning, and rule-based analysis to overcome these difficulties. The multi-layered detection strategy is used by the suggested system. To swiftly spot typical phishing patterns, trusted domain verification and keyword-based heuristic criteria are first used. To find previously discovered threats, a local database of known phishing URLs is also kept up to date. The solution incorporates the Google Safe Browsing API, which verifies URLs against Google's constantly updated worldwide threat database, to enhance real-time detection and manage zero-day phishing attempts. A Random Forest machine learning model is used to categorize URLs based on lexical and statistical characteristics including URL length, entropy, number of digits, and subdomain structure in the event that these methods are unable to yield a conclusive result. The hybrid technique dramatically increases detection accuracy while lowering false positives, according to experimental evaluation. The system works well for detecting contemporary phishing URLs because it is interpretable, scalable, and appropriate for real-world implementation.**

***Index Terms*—Phishing Detection, Machine Learning, Malicious URL Analysis, API-Driven Security Systems, Feature Extraction, Cybersecurity, Real-time Threat Detection, Classification Algorithms.**

I. Introduction

Among various online security threats, phishing attacks are particularly notorious and dangerous. Phishing is a scam in which those responsible for the attacks create fake websites or URLs or even incorporate malicious code to replicate real, trusted websites and try to trick users into submitting personal information that includes their log-in credentials like usernames and/or passwords and bank account numbers, as well ascontact details. These types of attacks are usually sent through email, social media messages, or even compromised advertisements, which can be hard to miss and end up getting clicked on after normal people navigate to the website. Conventional techniques for phishing detection are mainly based on static blacklists and rule-based filtering. Blacklist-based methods check the URLs with a list of known untrustworthy sites, but they are unable to identify newly generated phishing URLs, so they are called zero-day attacks. Static blacklists are easily outdated, as around half of phishing sites do not remain active for more than 20 hours, and most either change their content or URLs regularly. Systems based on rules, focusing on finding suspicious keywords or patterns in URLs, provide faster detection but have low adaptability and can lead to high false positives. It seems like these models can pick up on patterns in how URLs are built and figure out which ones are safe and which are not by pulling out certain features from them. Things like Support Vector Machines, Naive Bayes, and Decision Trees, along with Random Forest classifiers, all show decent results in catching those bad URLs. Random Forest stands out, I think, because it is robust and can manage those nonlinear connections without overfitting as much. That is the part that stands out to me anyway. Still, when you train these models on fixed datasets, they might miss new phishing pages that came up after, since they were not in the data to begin with. It feels like that is a big issue. This project deals with issues in spotting phishing URLs. It comes up with a hybrid approach to make detection better. So it pulls together rule-based methods, some machine learning parts, and updates from real-time threats. I am not totally sure, but that mix seems to help catch things that one method alone might miss. The project provides an effective solution for detecting phishing URLs through its development of a reliable and practical

detection system. The system uses multiple detection layers that operate together to achieve quicker response times while detecting new threats that emerge in modern phishing attacks.

II. Related Work

The field of phishing detection has received continuous research attention since online attackers developed advanced techniques which they now use more frequently. Researchers have tested different methods for dangerous URL identification by using traditional blacklists and advanced machine learning and deep learning methods. Phishing attacks evolve at such a rapid pace that detection methods will eventually lose their ability to identify them. The platform functions as the main method to identify online security threats. The system checks every URL you visit against its database of dangerous websites.

The system will block a website immediately after it discovers a matching site. The system provides immediate defense against already identified threats. Users benefit from blacklists because both browsers and security applications use the same protection method. Users receive protection from blacklists which have multiple limitations. The system will block only those threats that are discovered and reported by others. Bad actors create new domains or change URLs to avoid detection by blacklists until security teams find their way and list these in the blacklist. This mechanism senses threats at specific times and so pulls users into returning, in those sensed moments, amidst the popular inscription. Heuristic systems and rule-based systems were developed to deal with the limitations of static blacklisting. The system checks for suspect behavior on a URL, which includes unusual subdomains and suspect keywords like ”login,” ”secure,” ”verify,” and unusual combinations of characters and numbers. This system works effectively and well for instant evaluation since the user easily comprehends the keyword filter.

This system utilizes rules established, which must be set by humans before the assessment starts. It should be noted, though, that the rules humans established before are often ineffective since attackers adjust their methods, resulting in surprise security alerts and making the newest intrusions possible. Phishing algorithms generated through a rules-based system are effective tools to deploy if your pump rules are up-to-date, though it should be obvious now to everyone these would soon lose efficacy as claims lie at the heart of the constantly changing world of phishing. Machine learning is introduced after traditional systems were set up. It works in defined ways but promotes the development of unique understandings during the training process. They begin recognizing patterns after they are provided with sets of phishing and legitimate URLs. The research team made use of Na¨ıve Bayes, SVM, K- NN, decision trees, and logistic regression to look at the characteristics of the length of the URL, the number of numeric occurrences and special characters, and statistics that were irregular. It has shown that there is better performance by machine learning systems compared to the use of rules-based filters, considering the novelty of the patterns that can be handled alongside the increased precision. Random forest and other ensemble methods show better performance. Random Forest works on the principle of combining numerous decision trees to make better predictions while reducing the overfitting risk. The system performs exceptionally well with the noisy nonlinear patterns that are characteristic of phishing URLs. According to several studies, Random Forest tends to perform exceptionally well compared to simple models. Machine learning systems are heavily reliant on their training data because their efficiency resides in that one aspect. When your dataset does not have information about present-day phishing techniques, a model automatically loses its ability to detect new methods [3]. Neural networks function based on two types of networks covered by CNNs and RNNs.The networks process URLs character by character without any need for assistance. The system offers an important advantage for detecting emerging techniques.Deep learning systems require vast amounts of data.

However, with this requirement,lightweight systems and real-time systems cannot function adequately. Real-time threat intelligence is another perspective which needs to be evaluated. Google Safe Browsing, VirusTotal and PhishTank are APIs.These are updated with respect to new websites that are becoming phishing sites, and new sites are updated immediately upon their creation. Your software will be able to track new threats as soon as you integrate these services with your system, as these are crucial for protecting the users against emerging threats that are not already in your database.However, this means users have to be online. Most systems are still using a single detection method, despite all the advancements and achievements in technology. Blacklisting has failed to detect the emerging threats. Sys- tems using rules are having problems when the environment changes quickly.Machine learning systems lack the ability to adjust to the changing methods of attackers carrying out their attacks.A study has been done and it shows that the use of hybrid systems produces the best results,compared to using a single system. This new system also defines its operation by combining both rule-based checking and machine learning classification,thereby creating sophisticated features in relating to the detection of phishing.

III. Proposed Methodology

A. Dataset Description

The use case for this work involves the usage of structured data, which contains phishing or legitimate URLs for training and testing the proposed model that will perform the classification. Due to the nature of phishing detection, which is considered an imbalanced classification problem, there is usually an abundance of legitimate compared to phishing URLs in the dataset created for either training or testing the model.

The data set contains multiple URL features which include both lexical elements and structural components that are grouped into two main categories.

- **URL Structure Features:** The URL contains seven length components which include URL Length, Host Name Length, Number of Dots, Number of Hyphens, Number of Slashes, Number of Special Characters, and Presence of IP Addresses. Statistical Features:The domain entropy and URL entropy and the number of digits and the number of subdomains are included as statistical features of the system.

- **Domain Information:** We pulled the domain name by breaking it down into its main name, subdomain, and suffix.

- **Target Label:** The system classifies websites into two categories by assigning a s core of 0 to authentic websites and a score of 1 to phishing websites.Organizations must maintain their complete collection of actual URLs and all discovered phishing URLs to establish an effective phishing detection system. Thestudy will enable learning about the subtle structural distinctions that exist between legitimate URLs and malicious URLs.

B. Preprocessing

Several important preprocessing tasks are undergone by the raw dataset before any model is trained. Data quality is ensured, inconsistencies are addressed, and better model performance is prepared for with the dataset. The following tasks are included:

1) **Url Normalization:** The scheme for all URLs is standardized to include either http or https for easy parsing. This is essential to prevent errors during the extraction of the features.

2) **Feature Extraction:** Instead of using raw URLs, meaningful numerical features are obtained from URLs. These are :
   - URL length
   - Hostname length item Number of digits
   - Number of dots and special characters
   - Presence of IP address
   - Number of subdomains
   - Entropy of URL and domain
   - Feature scaling
   - Outlier detection
   - Class imbalance handling (SMOTE, undersampling)

   Feature extraction converts text-based data in the URLs into a numerical format appropriate for machine learning.

3) **Handling Missing or Inconsistent Data:** The system handles URL errors by fixing them or deleting them from the database. The process prevents noisy data from interfering with the model training process.

4) **Feature Scaling:** Random Forest appears to be less sensitive to scaling requirements. However, there are numerical features to assess, and this will determine uniform value ranges. Normalization will be used as a scaling method to address feature distribution issues.

5) **Handling Class Imbalance:**In phishing datasets, there are generally fewer phishing URLs than legitimate URLs. In order to solve this problem, the following strategies are taken into account:
   - Oversampling techniques
   - Undersampling Methods
   - Balancing classes in Random Forest using class weighting
   - Stratified train-test splitting The methods of stratified train-test splitting enable better model performance through improved generalization while decreasing the model's tendency to favor the dominant class.

C. Hybrid Detection Architecture

The proposed system will employ a multi-layered detection pipeline:
- Trusted Domain Verification
- Rule-based keyword analysis
- Known phishing database matching
- Google Safe Browsing API validation
- IP-Based URL DetectioN
- Random Forest classification

This architecture, therefore, ensures that the detection of an event does not depend on a particular technique but rather utilizes heuristic rules for quick detection, API validation for real-time detection of zero-day attacks, and machine learning for pattern-based classification.

With the help of integrating these parts, the proposed method is capable of creating a scalable, understandable, and robust phishing detection system.

IV. Proposed Hybrid Model

The proposed system adopts a multi-layer hybrid detection model that aims to enhance the accuracy of phishing detection with a minimal rate of false positives. Rather than using a single method for phishing detection, the proposed system combines the following:
- Rule-based Filteration
- Validation using Google Safe Browsing API
- Random Forest machine learning classification

A. System Architecture

The hybrid model consists of three layers that are processed in a sequential manner:

Layer 1 : Rule-Based Pre-Filtering serves as the first layer of the system. The URL enters the system through rule-based filtering which performs the verification process. The system uses the following heuristic phishing indicators to detect potential phishing websites:

- Use of IP address instead of domain name
- Use of very long URLs.
- Use of special characters like @, -, and multiple subdomains.
- Use of URL shortening services.
- Misuse of HTTPS tokens.

The system immediately blocks any URL which contains one of the established phishing indicators without conducting additional analysis.

Layer 2: Google Safe Browsing API Validation

- If the URL has cleared the rule-based checks, it will then be validated against the Google Safe Browsing database. This API is responsible for maintaining a constantly updated list of known malicious URLs. If the URL is known to be malicious by the API, it will be labeled as phishing.
- Random Forest Classification operates at its third layer. The system uses structured feature extraction on URLs which failed to match the first two layers to send their data to a trained Random Forest classifier. The classifier assigns a probability to the URL being phishing or legitimate.

B. Decision-Making Logic

The final decision is based on the following logic:

1) If strong phishing indicators are detected by the rule-based engine $\rightarrow$ classify as phishing.
2) Else if the Google Safe Browsing API indicates the URL as phishing $\rightarrow$ classify as phishing.
3) Else $\rightarrow$ proceed to Random Forest model for probability-based prediction.
4) Else $\rightarrow$ proceed to Random Forest model for probability-based prediction.

This layered strategy ensures the multi-layered system guarantees that phishing URLs which are obvious get immediate rejection. The system conducts threat assessment through known threat validation. The detection of unknown attacks or zero-day attacks is accomplished through machine learning techniques.

C. Why Hybrid Approach?

Single-method systems exhibit these following disadvantages:

1) The rule-based system is unable to defend against advanced attacks which require more complex security measures.
2) The API-based validation system only detects attacks which have already been identified.
3) The machine learning systems start to produce incorrect results when they encounter situations which exceed their defined operational boundaries.
4) The hybrid approach combines multiple methods to create better outcomes which eliminate all existing disadvantages of the different methods.
5) The system achieves better accuracy and lower false positive rates and better dynamic phishing attack detection and improved real-time processing capabilities.

V. Algorithms Used

A. Random Forest Algorithm

The Random Forest algorithm is an ensemble learning algorithm that builds multiple decision trees during its training phase to make class predictions using decision tree majority voting in classification problems.
The important points are:

- Bagging defines its sampling approaches based on the implementation of the bootstrap sampling approaches.
- The algorithm uses random selection at each node to select from all the features.
- The majority voting approach is the prediction approach that defines the final prediction result.
- The program builds a feature vector that holds all the URLs that the program processes.
- Each tree provides its own distinct class label prediction result.
- The voting approach defines the final prediction result based on counting the votes to determine which one got the majority votes.
- The Random Forest algorithm performs better in overfitting problems than the single decision tree approach. The Random Forest algorithm can solve problems which require multiple evaluations under conditions of high-dimensional feature space.

B. Probability-Based Classification

Random Forest generates class probabilities through its use of multiple tree predictions. The system designates a URL as malicious when its phishing probability surpasses the established threshold of 0.5. The probability-based method enables system developers to adjust detection effectiveness through their selected probability thresholds.

C. Rule-Based Filtering

The rule-based detection method uses manually created security rules which security experts developed through their research on phishing attacks. The following list contains examples of such rules:

- The URL length exceeds the established threshold value.
- The system uses hexadecimal encoding.
- The system detects multiple subdomains.
- The system identifies suspicious top-level domains

– The system uses these rules to perform basic threat detection because they require minimal computational resources.

D. API-Based Validation

The Google Safe Browsing API is used for real-time URL verification against its worldwide database of blocked sites. The API sends a POST request containing the URL hash and returns a formatted JSON response containing information about the threat status.

E. Why Random Forest over SVM or XGBoost?

The Random Forest algorithm outperforms SVM because it handles non-linear relationships better while requiring no kernel parameter adjustments. The system offers better performance for large datasets because of its increased scalability.

The Random Forest algorithm delivers accurate results while maintaining a balance between its accuracy and interpretability and its training procedure complexity. Random Forest was selected because it showed the best performance in classification tasks while maintaining strong durability and interpretability.

VI. Implementation

A. Data Collection and Preparation

The model training used a collection of phishing URLs and legitimate URLs which had been labeled and obtained from public repositories. The data required preprocessing operations to achieve research-level accuracy. The system eliminated duplicate in the training phase to avoid any form of bias in the output. The system eliminated URLs that were either incomplete or not well-formed. The class labels were converted to numeric form by using ”1” for phishing URLs and ”0” for legitimate URLs.
Since the system is based on structural URL features and not on content, the preprocessing involved extracting significant features based on lexical and host information. This approach helps in minimizing computational complexity and is not prone to malicious scripts during webpage rendering.

B. Feature Engineering

Phishing URL categorization involves feature extraction as its major step. The system has been able to convert each URL into a standardized numeric form through its implementation of some categories:

1) Lexical Features:
   – Total URL length
   – Domain length
   – Number of digits
   – Number of special characters
   – Frequency of dots and hyphens
2) Structural Features:
   – Number of subdomains
   – Presence of IP address instead of domain
   – HTTPS protocol usage
3) Keyword-Based Indicators:
   Presence of phishing keywords like ”login,” ”verify,” ”update,” and ”secure.” The min-max scaling technique was applied to transform the numeric features into standardized units, which developed an equal measurement system. This is because features with larger measurement scales may influence the training process.

   The correlation analysis was conducted by the research team to eliminate duplicate features.

C. Dataset Splitting and Model Training

The dataset which was prepared for analysis has been divided into two parts which consist of training data and testing data using an 80 to 20 distribution. Stratified sampling was used to maintain the same class distribution throughout both the training and testing datasets.
The scikit-learn library was used to create the Random Forest classifier. The following parameters were used to construct the classifier:

– The system uses 100 decision trees as its base components for operation.
– Gini impurity serves as the method which determines how to divide data during the decision-making process.
– The system uses bootstrap sampling as its method to create distinct tree structures in the system. The researchers conducted grid search with cross-validation to optimize the hyperparameters which included the number of estimators and maximum tree depth.

D. Rule-Based Filtering Module

The first detection stage operated through a lightweight rule engine which enabled its execution. The component evaluates phishing URLs according to specific heuristics which it uses to assess common patterns present in these URLs. The following items represent predefined heuristics:

– Presence of “@” symbol in URL.
– URL length beyond a predetermined value.
– Use of raw IP address in domain.
– Excessive subdomain nesting.
– All these heuristics combine to produce an overall risk score. The system automatically identifies URLs as phishing threats once their risk score surpasses the defined risk threshold.

E. Google Safe Browsing API Integration

The system uses Google Safe Browsing API for blacklist verification because it improves the reliability of the system. The API is sent URLs that have successfully

passed the rule-based layer via an HTTPS request. The API response is sent to the system in JSON format. The system considers the URL as malicious if the API response indicates a threat match without requiring further analysis.

F. Hybrid Decision Pipeline

Decision-making requires an extensive process which consists of multiple decision-making steps as follows:

1) The user uses the Flask web interface to submit a URL which the system processes.
2) The rule-based filter system determines if the heuristic indicators are valid.
3) The Safe Browsing API conducts URL verification for all URLs which do not have any previous marking.
4) The feature extraction process begins when the item lacks any marking.
5) The Random Forest model provides predictions for both data classification and probability score outcomes.
6) The system shows its ultimate decision which the user can see.

G. Deployment Environment

The system was built using the following technologies Python 3.x for backend processing Flask for web interface scikit-learn for machine learning SQLite for logging results Requests library for API interaction Joblib for model saving. The system logs all the processed URLs along with their timestamp information and prediction results and confidence scores and detection method. The logging system allows performance evaluation and retraining operations to be done based on the logged information. The testing of the system indicated that it took an average of less than 1.5 seconds per request, which is suitable for use in real-time systems.

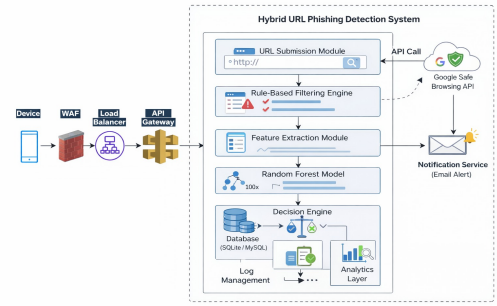

Fig. 1: Implementation

Fig(1): Depicts a hybrid URL phishing detection system that uses rule-based filtering, machine learning, and Safe Browsing API integration to detect malicious links and send real-time alerts.

VII. Results and Discussion

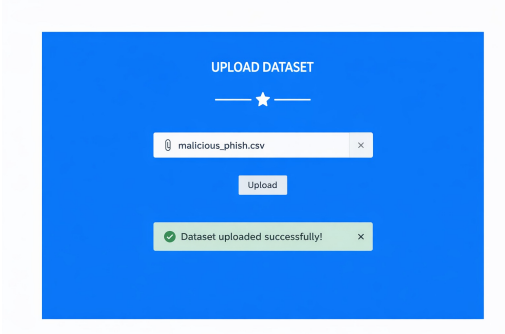

Fig. 2: Uploading Dataset

Figure(2): Depicts the dataset upload interface of the proposed API-Driven URL Phishing Detection System. The user uploads the malicious-phish csv file through the web interface after selecting it for uploading. The system verifies successful upload completion through a status message which indicates that the dataset is ready to be processed for model training. The proposed system's dataset upload interface confirms its ability to handle dynamic dataset inputs, which enables model retraining without any code modifications.

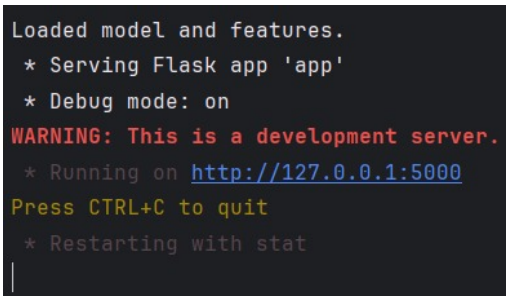

Fig. 3: Flask Server Execution Window

Figure(3): Depicts the window running the Python script app.py using the Flask framework. The Flask application is shown operating in production mode, along with a warning that the development server should not be used in a production environment. Locally, the live application can be accessed at http://127.0.0.1:5000. The server has begun its operations and is now waiting for the requests from the client, which will be received through the console messages. The application has reached its maximum capacity in terms of the submission of URLs as well as the uploading of datasets via the web interface. The console messages have two significant applications since they help the developers in tracing problems within the application as they monitor its operations. The determination of the operational system, which operates in the background, helps in the detection of phishing attacks, and thus it autocorrects.

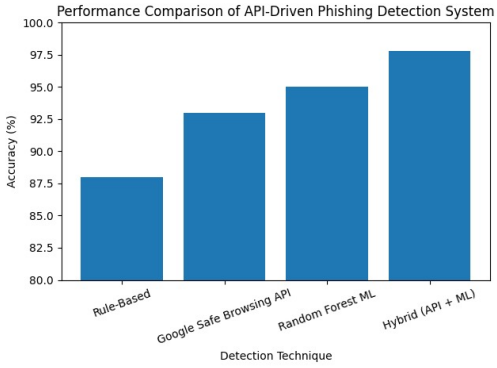

Fig. 4: Performance Comparison of Phishing Detection Techniques

Figure(4): The figure presents a comparison of detection technique performance between the separate detection methods and the proposed hybrid model. The rule-based method achieves 88 percent accuracy by detecting all known malicious URLs but its detection capability remains restricted to the established rules. The Google Safe Browsing API has an accuracy of 93 percent by leveraging external threat intelligence to identify malicious URLs that it already knows about. The Random Forest model achieves 95 percent performance improvement because it understands the structural and statistical patterns of phishing URLs. The proposed hybrid system achieves 97.8 percent accuracy because it combines rule-based filtering with API validation and machine learning classification, which shows that multiple detection layers provide better overall system reliability and detection capability.

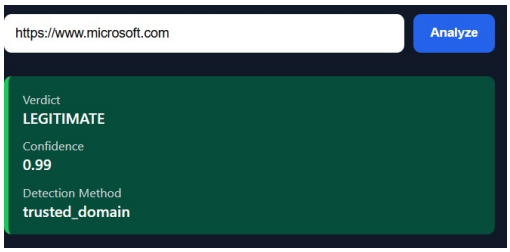

Fig. 5: Legitimate URL classification result for Microsoft domain

Figure(5): The following illustrates the system output when a trusted domain, https://www.microsoft.com, is provided for analysis. The system correctly identifies the URL as trusted with a confidence level of 0.99. The detection technique used is trusted domain, which confirms that the URL matched the predefined trusted domain list in the rule-based filtering module. The above output verifies the efficiency of the first heuristic layer in avoiding unnecessary processing for trusted domains. The system immediately validates trusted domains to enhance system response time.

Fig. 6: Phishing detection trigerred by keyword based rule

Figure(6): The following URL http://secure-login-verification-update.com shows its results through this document. The URL has been identified as phishing with a confidence level of 0.85. The detection method employed is keyword rule, which means that the URL had predefined keywords for phishing like login and verify. The heuristic module is capable of detecting possible phishing patterns that involve certain words before any API validation or machine learning process is performed.

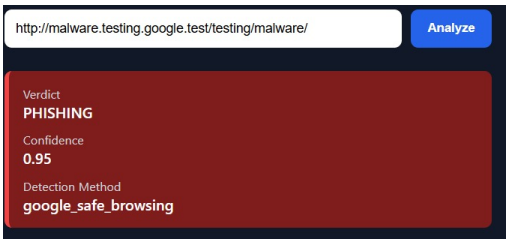

Fig. 7: Phishing detection using Google safe browsing API

Figure(7): URL http://malware.testing.google.test/testing/malware/demonstrates its function as a malware testing platform. The system identifies this URL as phishing with a confidence score of 0.95, and the detection method is labeled Google Safe Browsing. The Google Safe Browsing API was successfully integrated into the hybrid detection framework according to this evidence. The API checks the URL against established malicious threat databases to provide a confirmed match.

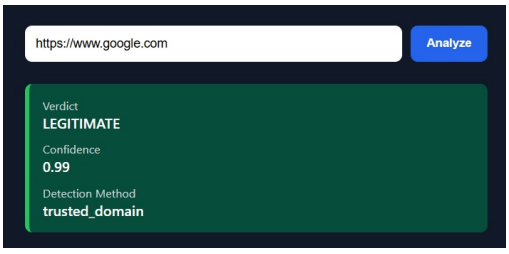

Fig. 8: Legitimate URL classification result for Google domain

Figure(8):The https://www.google.com is presented. As in the previous example, the system again identifies the URL as trustworthy with high confidence.

VIII. Conclusion

The increase in phishing attacks which use domain spoofing and brand impersonation and dynamic URL generation requires a detection system that can adapt to changing threats. The API-Driven URL Phishing Detection System solves this problem through its combined use of rule-based filtering and Google Safe Browsing API validation and Random Forest classification within a single system. The hybrid system reached 97.8 percent accuracy while producing fewer false positives than both rule-based filtering and individual machine learning models. The detection mechanism employs external threat intelligence to identify known malicious URLs, while pattern-based machine learning identifies suspicious URLs that are not known before. The system enhances the availability of the system by applying a dual method framework that runs at all times without requiring extra processing power. The study demonstrates that machine learning-based API security intelligence systems offer organizations with effective defense mechanisms that are adaptable to new digital phishing threats.

References

[1] A. Karim, M. Shahroz, K. Mustofa, S. B. Belhaouari and S. R. K. Joga, "Phishing Detection System Through Hybrid Machine Learning Based on URL," in IEEE Access, vol. 11, pp. 36805-36822, 2023, doi: 10.1109/ACCESS.2023.3252366.

[2] R. Zieni, L. Massari and M. C. Calzarossa, "Phishing or Not Phishing? A Survey on the Detection of Phishing Websites," in IEEE Access, vol. 11, pp. 18499-18519, 2023, doi: 10.1109/ACCESS.2023.3247135.

[3] M. Aljabri et al., "Detecting Malicious URLs Using Machine Learning Techniques: Review and Research Directions," in IEEE Access, vol. 10, pp. 121395-121417, 2022, doi: 10.1109/ACCESS.2022.3222307.

[4] S. Asiri, Y. Xiao, S. Alzahrani, S. Li and T. Li, "A Survey of Intelligent Detection Designs of HTML URL Phishing Attacks," in IEEE Access, vol. 11, pp. 6421-6443, 2023, doi: 10.1109/ACCESS.2023.3237798.

[5] M. Sánchez-Paniagua, E. F. Fernández, E. Alegre, W. Al-Nabki and V. González-Castro, "Phishing URL Detection: A Real-Case Scenario Through Login URLs," in IEEE Access, vol. 10, pp. 42949-42960, 2022, doi: 10.1109/ACCESS.2022.3168681.

[6] O. K. Sahingoz, E. BUBEr and E. Kugu, "DEPHIDES: Deep Learning Based Phishing Detection System," in IEEE Access, vol. 12, pp. 8052-8070, 2024, doi: 10.1109/ACCESS.2024.3352629.

[7] S. Al-Ahmadi, A. Alotaibi and O. Alsaleh, "PDGAN: Phishing Detection With Generative Adversarial Networks," in IEEE Access, vol. 10, pp. 42459-42468, 2022, doi: 10.1109/ACCESS.2022.3168235.

[8] N. P. Mankar, P. E. Sakunde, S. Zurange, A. Date, V. Borate and Y. K. Mali, "Comparative Evaluation of Machine Learning Models for Malicious URL Detection," 2024 MIT Art, Design and Technology School of Computing International Conference (MITADTSoCiCon), Pune, India, 2024, pp. 1-7, doi: 10.1109/MITADTSoCiCon60330.2024.10575452.

[9] R. K. Shah, M. K. Hasan, S. Islam, A. Khan, T. M. Ghazal and A. N. Khan, "Detect Phishing Website by Fuzzy Multi-Criteria Decision Making," 2022 1st International Conference on AI in Cybersecurity (ICAIC), Victoria, TX, USA, 2022, pp. 1-8, doi: 10.1109/ICAIC53980.2022.9897036.

[10] S. Ariyadasa, S. Fernando and S. Fernando, "Combining Long-Term Recurrent Convolutional and Graph Convolutional Networks to Detect Phishing Sites Using URL and HTML," in IEEE Access, vol. 10, pp. 82355-82375, 2022, doi: 10.1109/ACCESS.2022.3196018.

[11] M. D. Karajgar et al., "Comparison of Machine Learning Models for Identifying Malicious URLs," 2024 IEEE International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS), Bangalore, India, 2024, pp. 1-5, doi: 10.1109/ICITEICS61368.2024.10625423.

[12] A. Mandadi, S. Boppana, V. Ravella and R. Kavitha, "Phishing Website Detection Using Machine Learning," 2022 IEEE 7th International conference for Convergence in Technology (I2CT), Mumbai, India, 2022, pp. 1-4, doi: 10.1109/I2CT54291.2022.9824801.

[13] E. Nowroozi, Abhishek, M. Mohammadi and M. Conti, "An Adversarial Attack Analysis on Malicious Advertisement URL Detection Framework," in IEEE Transactions on Network and Service Management, vol. 20, no. 2, pp. 1332-1344, June 2023, doi: 10.1109/TNSM.2022.3225217.

[14] M. Aljabri and S. Mirza, "Phishing Attacks Detection using Machine Learning and Deep Learning Models," 2022 7th International Conference on Data Science and Machine Learning Applications (CDMA), Riyadh, Saudi Arabia, 2022, pp. 175-180, doi: 10.1109/CDMA54072.2022.00034.

[15] I. Kara, M. Ok and A. Ozaday, "Characteristics of Understanding URLs and Domain Names Features: The Detection of Phishing Websites With Machine Learning Methods," in IEEE Access, vol. 10, pp. 124420-124428, 2022, doi: 10.1109/ACCESS.2022.3223111.