JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

PHISHING URL DETECTION THROUGH MULTI-SIGNAL AGGREGATION AND MACHINE LEARNING

[1]Abdul Raheem, [2]Dr. Thriveni J

[1]Student , [2] Professor & Chairperson
[1]Computer Science and Engineering
[1]University of Visvesvaraya College of Engineering, Bengaluru, India

Abstract : Phishing is still one of the most efficient ways to commit online fraud and steal credentials. In this study, we provide a real-time URL classification technique that uses a variety of signals to determine if a URL is secure or phishing. Reputation checks from reputable threat intelligence sources (URLVoid, McAfee, Sucuri), verifying the SSL certificate and WHOIS information for the domain, calculating topological features for each domain (domain age, IP blacklist), and ultimately classifying the URL using an RF classifier trained on a variety of labeled phishing datasets are all part of the classification process. We gathered a variety of data from Phishtank and OpenPhish, along with a few secure URLs, to evaluate the system's performance. to assess the system's accuracy, recall, ROC-AUC, and F1 score. We also examined how obfuscation strategies affect the evolving characteristics of phishing and how that connects to our system's overall resilience. Based on our results, we think that the capacity to identify phishing is much enhanced by combining data from several sources and feature sets as opposed to depending just on one source or feature set. As a result, this project creates a validated, practical, and efficient real-time phishing detection system. It also supports the necessity of combining various signals and offers a framework for determining which characteristics are crucial for spotting potential threats and enhancing the capabilities of a phishing detection system.

IndexTerms **- Phishing Detection, URL Classification, Threat Intelligence Aggregation, Machine Learning, Real-Time Security, LightGBM, Random Forest.**

I. INTRODUCTION

The technique known as "phishing" entails thieves using fraudulent methods to get victims' banking, social media, and email passwords. Through carefully placed links provided via various kinds of communication (such as spoof emails, Short Message Service (SMS), Quick Response (QR) codes, instant messaging, etc.), phishing allows criminals to unintentionally direct people to a phony website. Usually, the features of the phony websites are made to resemble those of the authentic ones. These phony websites are usually made using a variety of phishing tools [1]. People should be cautious to keep an eye out for phishing indications since criminals utilize phishing strategies to exploit people's faith in reputable companies. People can improve their capacity to protect sensitive data from possible compromise by learning to recognize typical phishing behaviors. The only goal of phishing scammers is to fool people into disclosing their login information so they may access the victims' internet accounts. The stolen credentials can be used by the attackers to achieve their goals, which may include impersonation, defamation, espionage, and other financial gains. The most common trigger for serious cyberattacks, such as ransomware and virus attacks, is phishing. The many enhancements connected to a traditional phishing effort are shown in Fig. 1.

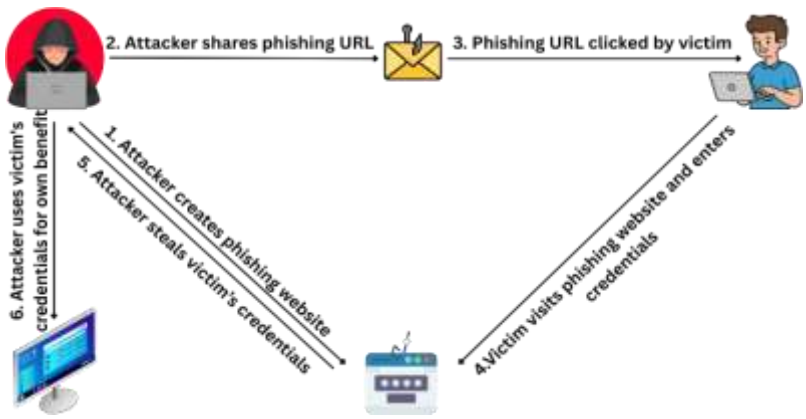

Fig 1. Life cycle of a phishing attack.

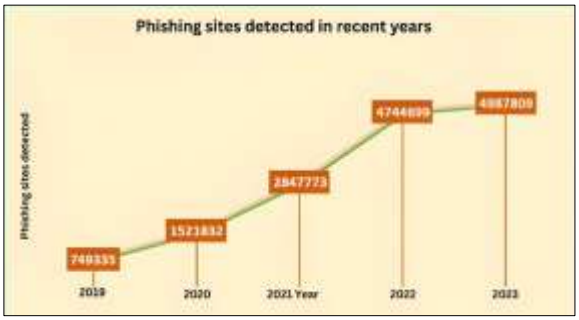

Fig 2. Phishing websites detected in last few years.

Fig 3. Phishing sites detected in last few quarters.

The frequency of reported phishing websites has not decreased in recent years, despite a great deal of research being done to address this issue. According to research on phishing activity patterns by the APWG (Anti-Phishing Working Group), recorded phishing incidents have increased in the post-COVID years starting in 2020. Phishing websites detected since 2019 are shown in Fig. 2. Different phishing sites reported on a quarterly basis during the previous two years are shown in Fig. 3. 1,624,144 instances of phishing. The number of phishing sites found in each quarter is higher than the annual records of the pre-COVID period, notwithstanding a little slowdown in the last few quarters. is ignorance. The messages are meant to look visually authentic, whether they are sent via email, SMS, QR codes, or other media. Additionally, they utilize deceptive wording intended to evoke feelings of anxiety, stress, panic, or urgency, encouraging users to ignore minute flaws in the phishing links and act quickly.

Phishing URL	Legitimate URL	Squatting Type
https://amaz0nn.asia	https://amazon.com	Typo-squatting
https://aeonbanick.com	https://aeonbank.com	Typo-squatting
https://bnl-bnpparibas.com	https://bnl-bnpparibas.com	Combo-squatting
https://steamcamunity.ru	https://steamcommunity.com	Sound-squatting
http://pankakeswape.com	http://pancakeswap.finance	Typo-squatting
https://faceb00k.homes	https://facebook.com/	Homograph-squatting

Table 1. Some examples of domain-squatting.

Phishing URL	Legitimate URL	Obfuscation Type
https://www.paypal.usid.me	https://www.paypal.com	Brand as subdomain
https://amazon.co.jp.ggbu14.com	https://amazon.co.jp	Brand as subdomain
http://pancakeswapfinance.li	http://pancakeswap.finance	Missing dot obfuscation
https://pancakeswap.services	https://pancakeswap.finance	Fake Top-level-Domain
https://bnpparibas.shop	https://bnpparibas.com	Fake Top-level-Domain
https://aw-store.ru/auth/netflix	https://netflix.com	Brand in path

Table 2. Some examples of URL obfuscation.

The attackers engage in domain-squatting [3], which is the purchasing of domain names that resemble well-known companies. They then either share their URL with the victims or wait for the victims to make a typographical error and visit the fraudulent website that the attacker has created. Domain squatting can take many different forms, depending on things like spelling mistakes [4], bit flips [5], homophones [6], mixing another term with the well-known brand [7], and utilizing similar-looking characters [8]. Some of the squatter phishing domains are shown alongside the domains of actual businesses in Table 1.

Attackers commonly employ URL obfuscation [9] in addition to domain squatting to deceive victims into believing malicious URLs to be authentic. Maintaining the brand name in the URL but using it as a subdomain of a phishing website is a common strategy. In order to falsify validity, attackers may potentially alter the country code TLD (ccTLD) or top-level domain (TLD). Inserting the brand name into the URL route or query string is another common obfuscation technique that further deceives visitors by creating the appearance of legitimacy while really connecting to a dangerous location. Some of the phishing domains registered for URL obfuscation are included in Table 2.

Phishing assaults on businesses have nearly increased in recent years, with 621 brands impacted in July 2022, according to research [10]. Approximately 55% of phishing attacks use a reputable brand to increase their validity, with LinkedIn being the top phished brand [11]. Customers must exercise caution and be aware of the tactics used by phishers in order to prevent a phishing attack. This isn't always possible because anybody can use the internet, regardless of their level of technological expertise. It is practically hard to provide training for every internet user to prevent phishers. Therefore, the best options for successfully thwarting phishing attempts are software-based phishing detection systems.

II. LITERATURE SURVEY

This section reviews recent and widely adopted academic approaches for phishing website detection.

To identify a zero-day phishing assault, a range of machine learning techniques employ a mix of several characteristics (NLP based, word vector based, hybrid). Using the Random Forest method and NLP characteristics, the authors [12] created a sizable dataset of 73575 URLs, with an accuracy of 97.98%. This method is extremely accurate, but when it comes to analyzing URLs with little or no domain information, it is less reliable.

In order to train a logistic regression classifier based on the source code of web pages, this article explains a method for feature extraction using hyperlinks. 98.4% classification accuracy was achieved using a logistic regression classifier trained using this technique [13]. However, because this approach solely relies on source code analysis, it may be attacked using a variety of techniques, including altering JavaScript references and links, favicons, and embedded objects.

Reference number [14] offered a different method for using search engines to find phishing-based websites. This method looks for any pertinent identification keywords in a web page's title, meta description, and content. The TF-IDF values are then used to determine the weight of the pertinent keywords depending on where they occur on the page. If a website's domain name shows up in the top results from the chosen third-party search engines, it might be accepted or rejected as legitimate or phishing-based. This method produced an accuracy of 89% on a test sample of 200 legal and phishing-based websites. The results that third-party search engines offer are crucial to the strategy.

According to reference [15] suggested a lightweight method for quickly identifying phishing websites on mobile and Internet of Things devices (a resource-constrained environment) using just nine URL-based characteristics. Although the authors indicate an average detection accuracy of 99.5%, the technique's practical usefulness may be limited by the restricted feature set.

Based on their own dataset, the authors of this work achieved an Accuracy Rate of 96.76% by using hybrid features, which incorporated both URL-based and hyperlink-based techniques as well as text features, all contained inside the XGBoost classification model (reference [16]). An accuracy rate of 98.12% was achieved by using a mixture of ensembles using Canopy-based Feature Selection, Grid Search for parameter optimization, and Cross Validation techniques, as reported in Reference [17].

A hybrid phishing detection technique that combines the advantages of textual and visual data sources is proposed in the work in [18]. While the textual component uses n-grams taken from a web page's content, the visual module looks at the placement, coloring, and size of logos on websites. A search query that can determine whether or not any of the top-ranking websites produce results matching the query as authentic websites is created using the data from both modules. This strategy has been shown to have a 98.60% success rate. Even so, this is a major step in the direction of hybridization. The detection of URL obfuscation and domain-squatting assaults has not received much attention to yet. As a result, current detection techniques rely on databases and are unable to identify novel forms of domain-squatting attacks because they are restricted to using either a trusted source of trademarked names or a blacklist in conjunction with those URLs that have been identified with a valid DNS (domain name system) record.

With a success rate of roughly 15% of all SSL certificates issued, a multilingual transformer-based system proposed in [19] can generate phonetic squatting variants and locate them in SSL certificates. A context-free system that models the inconsistent domain names on the banking website in [20] is an alternative to the multilingual transformer model, but it is limited to financial domains.

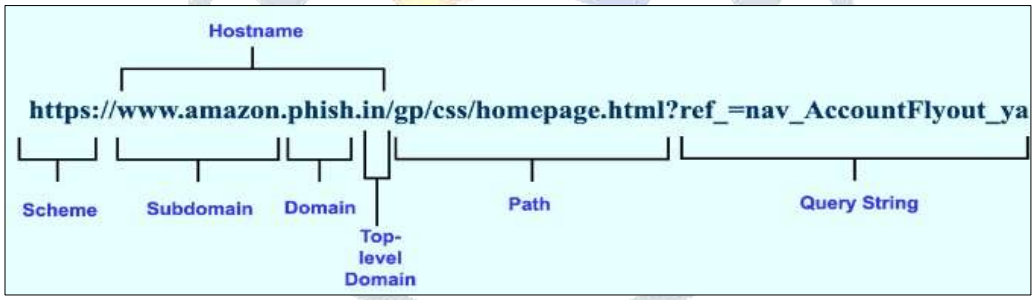

Fig 4. Components of a URL.

A lightweight framework for phishing detection on mobile devices with great computational efficiency is proposed by the study reported in [21]. Preliminary study revealed that employing an artificial neural network (ANN) model to execute this task on both URL and HTML inputs produced subpar results when the URL was the only input method. As a result, two transformer-based NLP models (BERT and ELECTRA) were trained directly on the URL input for additional testing. It was discovered that these models significantly outperformed earlier research employing feature-based detection techniques in terms of both generalization and reproducibility.

III. PROPOSED MODEL

This section describes how the service breaks down incoming URLs, the provenance and handling of the project's URL datasets, the preprocessing procedures before inference, the extracted feature families, and the FastAPI backbend's end-to-end scoring architecture.

1) COMPOSITION OF A URL

Any web resource, including HTML pages, media assets, API endpoints, and downloaded binaries, has a unique location called a URL. Each request is broken down into four canonical levels by the detector for analysis:

- **Scheme** determines whether TLS is anticipated and encodes the transport protocol (http, https). The prefix preceding :// is isolated by the parser; discrepancies between scheme and certificate state have an instant impact on the heuristic score.

- **Hostname** The origin server is uniquely identified by (subdomain + registered domain + TLD). The system can compare hosts to the Alexa Top 1M corpus, WHOIS data, temporary-domain lists, and typo-squatting variants by removing common prefixes (www.) and normalizing case.
- **Path** describes the host's directory structure. The part that comes after the hostname up to any? is used to calculate token counts, nesting depth, and suspicious folder names.
- **Query string** Key-value parameters are inserted after? and separated by & in the query string. These arguments are optional, although they frequently encode payloads or tracking data; the project counts and characterizes them as lexical characteristics.

To provide auditability, the API logs each URL component (scheme, hostname, path, and query) before feature extraction, as shown in Fig. 4.

2) SYSTEM WORKFLOW

The whole FastAPI-based prediction pipeline utilized in this study is shown in Fig. 4. Either curated CSV files or API queries are used to obtain the URL input. The URL is initially normalized (lowercase, whitespace eliminated, schemes deduced, and the canonical network location retrieved) when it is received. Following URL normalization, duplicate URIs will be eliminated, class imbalance will be resolved in the training notebooks, and the URL will be compared to lists of known benign and harmful sources. Each URL will then undergo a poor domain analysis. The following techniques are used in the bad domain analysis to identify obfuscation and domain squatting indicators: searching for hosts that appear on the Alexa Top 1 million list by comparing; using Levenshtein similarity tests to generate typosquatting permutation URLs from the typo-squatting module; use the WHOIS data to locate recently registered domains and temporary registrars; comparing the IP sets of FireHOL with an IP literal URL. The reputation verification procedure then makes use of a local blacklist and many threat intelligence sources, including URLVoid, Norton, and McAfee. URLs may be terminated early if they are determined to be high-confidence malicious URLs. After passing through this series, URLs are compared to lexical characteristics. These checks are used to generate machine learning input vectors based on specific mathematical calculations pertaining to the structural/statistical properties of the URL (length, entropy, token counts, number of digits/hyphens, density of special characters, and presence of a malicious extension). Then, using binary classification, the RF-trained model will divide the URL into two groups: phishing and legitimate. A thorough weighted scoring model will be created using the findings from every examination. A penalty will be added for each check, with a starting score of 180. This will provide consistency and repeatability between model creation and real-time deployment by giving users human-readable JSON procedures.

3) FEATURES EXTRACTED

3.1 URL Normalization & Domain Analysis

In order to identify potential high-risk domains early on, an algorithm known as URL normalization examines the URL and conducts Risk Assessment at the Domain level. When the URL is received, it is converted to lower case, any extra spaces are eliminated, and if no scheme is found in the URL, an assumption is made about its type by default. In order to obtain all of the BAD Domain signs, for example, this will enable the URL to be parsed into its component parts. Are the domains on the Alexa Top 1 billion list eligible for early allowlisting, or are there any domains that closely resemble well-known brands and can be determined through typo-squatting permutation analysis? Have any Domain Registrars been designated as Temporary through WHOIS_DATA, or have any of the domains been newly created based on the estimated Domain Age through WHOIS_DATA? Additionally, before deciding whether to proceed with further investigation, any spoken URLs have been compared to the FireHOL Blacklists. Before additional research is done, each of these Domain Level Feature Set items provide extremely powerful signals to identify Brandjackers, Squares, and Infrastructure Abusers.

Algorithm 1: URL Normalization & Domain Analysis

1: **Step 1**: Normalize the URL
2: **function** NORMALIZE_URL (url)
3: url ← lowercase(trim(url))
4: if scheme missing then infer scheme as http
5: parse url into scheme, hostname, path, query
6: return normalized_url
7: **end function**
8: **Step 2:** Extract bad-domain features
9: **function** EXTRACT_DOMAIN_FEATURES (normalized_url)
10: **D1** ← check Alexa Top 1M membership
11: **D2** ← compute typo-squatting similarity with known brands
12: **D3** ← check temporary registrar using WHOIS
13: **D4** ← check domain age ≥ 3 months
14: **D5** ← detect IP-literal URL & check FireHOL IP sets
15: **return** {D1, D2, D3, D4, D5}
16: **end function**

3.2 Lexical URL Feature Extraction

To address both the structural and statistical aspects of URLs, a method for extracting URL features using lexical characteristics has been developed. All of the characteristics obtained from this approach are calculated locally when speed and reliability are taken into account. Total URL length, Shannon entropy of the domain to gauge randomness, IP address references, flags for malicious file extensions, counts of query parameters, counts of path tokens, counts of hyphens, counts of digits, and counts of special characters like (@, #, $, %, ^, &, *, _, and +) are among the lexical features produced by the second feature-extraction algorithm. When combined, these characteristics can generalize the most common ways to obfuscate and manipulate phishing URLs, which serves as the foundation for the input vector of the machine learning algorithm.

Algorithm 2: Lexical URL Feature Extraction

```
function EXTRACT_LEXICAL_FEATURES(normalized_url)
    L1 ← compute total URL length
    L2 ← compute domain Shannon entropy
    L3 ← detect IP address usage
    L4 ← detect malicious file extensions
    L5 ← count query parameters
    L6 ← count path tokens
    L7 ← count hyphens
    L8 ← count digits
    L9 ← detect special characters (@, !, #, $, %, ^, &, *, _, +)
    return {L1…L9}
end function
```

3.3 Reputation & Transport Analysis

The third method increases the system's confidence in accurately identifying a danger by combining transport-layer security signals with external reputation intelligence. Numerous external threat information repositories, such as URLVoid, Google Safe Browsing, Norton Safe Web, McAfee SiteAdvisor, Sucuri, and a locally kept blacklist, are queried by this algorithm. It simultaneously verifies the validity of the SSL certificate linked to the connection and the transport-level security of each connection (HTTPS). It offers current contextualized awareness of risks and, consequently, very high levels of confidence in recognizing known dangerous URLs by merging the external reputation signals with locally derived information.

Algorithm 3: Reputation & Transport Analysis

```
function REPUTATION_AND_SSL_CHECK (url)
    R1 ← query URLVoid
    R2 ← query Google Safe Browsing
    R3 ← query Norton Safe Web
    R4 ← query McAfee SiteAdvisor
    R5 ← query Sucuri blacklist
    R6 ← check local blacklist
    T1 ← verify SSL certificate validity
    T2 ← check HTTPS usage
    return {R*, T*}
end function
```

3.4 Reputation & Transport Analysis

To provide a comprehensible result of phishing or not, we integrate domain indications with other elements such as language, trust signals, and transport-related tests. URLs categorized as "safe" will be automatically allocated to a lawful category by using previous whitelist criteria, removing any needless assessments. The remaining URLs are then assessed using a trained RF Model for lexical indicators in conjunction with a weighted scoring technique that aggregates all three categories of indicators. In the end, a risk value and a classification choice will be provided, and any poor assessments or incorrect forecasts will be punished. We can produce more accurate findings, build more robust systems, and enable system users to comprehend the rationale behind each URL's classification by using a variety of methods to aggregate the data.

Algorithm 4: Phishing Classification & Scoring

```
function CLASSIFY_URL (url)
    normalized_url ← NORMALIZE_URL (url)
    {D1…D5} ← EXTRACT_DOMAIN_FEATURES (normalized_url)
        if D1 = = TRUE then
                Note URL as Legitimate (early allowlist)
                return Legitimate
        end if
```

```
8:  {L1…L9} ← EXTRACT_LEXICAL_FEATURES (normalized_url)
9:  {R*, T*} ← REPUTATION_AND_SSL_CHECK (normalized_url)
10:     y_pred ← RF. predict ({L1…L9})
11:     score ← 180
12:     apply weighted penalties based on D*, R*, T*, and y_pred
13:     if score < threshold then
14:         label URL as Phishing
15:     else
16:         label URL as Legitimate
17:     end if
18:     return final verdict and score
19: end function
```

---

4) DATASET CONSTRUCTION & LABELLING

The dataset will be created by combining open phishing corpus and benign URL lists with references to the Alexa Top 1 million Domains from many CSV files included in the data/ directory. Prior to processing, all URLs were normalized by converting them to lowercase, removing netlocs, and inferencing schemes. Duplicates were then removed and combined from all sources. In order to ensure that the URLs remain constant throughout processing, reputation-based cues are only used. The final state of all URLs is labeled depending on the source's label, either as phishing or benign (which the source maintains). In order to address the prevailing benign bias in the dataset, preprocessing notebooks will partition the dataset into training and validation phases and use either sampling or weighting strategies to address the class imbalance. After processing and labeling are finished, the resulting artifact from the model created by RF will be used for deployment.

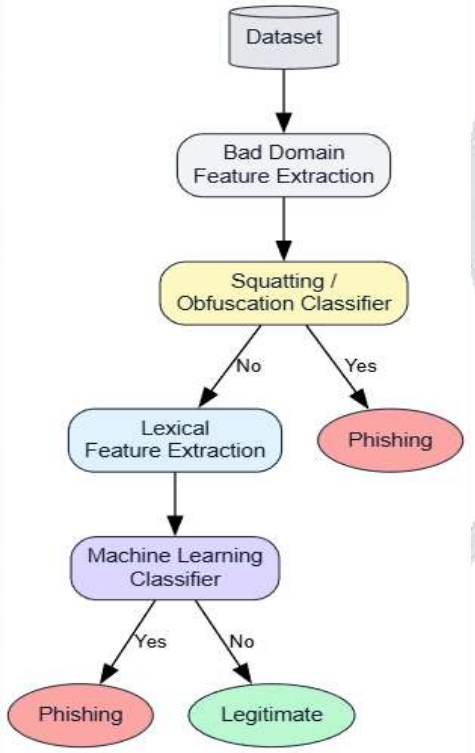

Fig 5. System workflow of the proposed approach

5) MODEL TRAINING AND HYPERPARAMETER SETUP

A Random Forest model is kept as a baseline, and the final classifier is an RF model chosen based on comparison tests carried out in ModelTrainin. Nine lexical URL features—URL length, entropy, IP address flag, malicious extension flag, query parameter count, path token count, hyphen count, digit count, and special-character presence—are used to train the model. Evaluation is done via held-out validation or cross-validation, and hyperparameters are adjusted over learning rate, number of leaves, tree depth, minimum data in leaf, feature fraction, bagging fraction, and boosting iterations. RF was selected because of its resilience over feature interactions and excellent accuracy latency trade-off. The FastAPI backend loads the learned model for inference at runtime.

6) COMPUTATIONAL COMPLEXITY ANALYSIS

Lexical feature extraction and URL parsing take linear time in relation to the length of the URL. Due to CSV scanning, domain lookups against the Alexa Top 1 million list now have linear-time complexity; however, indexed data structures can reduce this to constant time. The creation of typos and the assessment of similarity are constrained by predetermined bounds, resulting in useful linear behavior for the number of candidates. RF's short trees and constrained feature dimensionality result in little processing cost for machine learning inference. Network-bound reputation queries account for the majority of overall request delay, whereas completely local inference finishes each URL in a matter of milliseconds.

7) THREAT MODEL AND ASSUMPTIONS

It is expected that the attacker can create arbitrary URLs using obfuscation methods including IP-literal addressing, query stuffing, typo squatting, and short-lived domain registration. IP blacklists, WHOIS-derived information, benign domain corpora, domain reputation services, and a trained lexical machine learning classifier are all available to the defense. The approach takes Alexa Top 1 million membership as a strong benign prior and makes the assumption that WHOIS and reputation services are reasonably available. Client-side exploitation, compromised lawful domains, and content-based assaults are deemed outside of the scope. Without reputation cues, identification relies only on machine learning predictions and lexical heuristics, which are less reliable.

IV. RESULTS AND DISCUSSION

1) Experimental Setup

Experiments with a recently created phishing detection system based on a sizable publicly accessible dataset comprising phishing sites from several sources, including Kaggle, PhishTank, and OpenPhish, as well as lists of supplied benign sites, were finished using a variety of Python ML libraries, including Scikit-learn and RF. Following data pre-processing, normalization, and de-duplication, the final dataset had about 1,541,538 distinct labeled URL samples, of which 1 were phishing URLs and 0 were benign URLs. The tagged URLs are divided into training and testing sets using conventional procedures, and the appropriate sampling and weighting techniques are applied to reduce class imbalances. The following standard metrics—accuracy, precision, recall, F1 score, and ROC-AUC—were used to assess the models in relation to the cyber security unbalanced datasets.

2) Dataset Statistics

The dataset distribution for each experiment is shown in this table (Table 3). The dataset contains 966,982 benign URLs and 574,556 phishing URLs, offering sufficient quantity and variety for extensive testing. Therefore, it is reasonable to assume that the suggested system's applicability will translate effectively to actual phishing scenarios.

Table 3. Dataset Distribution Used for Experiments

Category	Phishing URLs	Benign URLs	Total
Available in dataset	574,556	966,982	1,541,538

3) Performance Comparison of ML Models

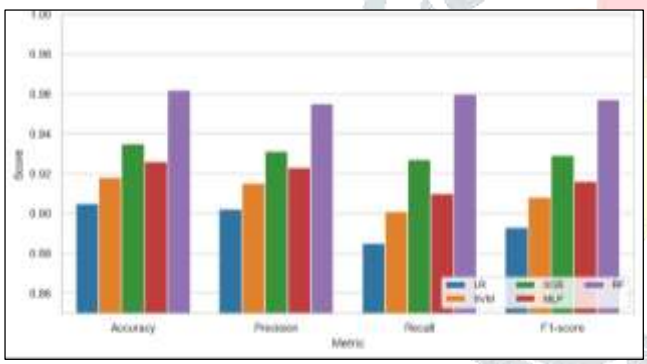

Fig 6. Before Feature Engineering.

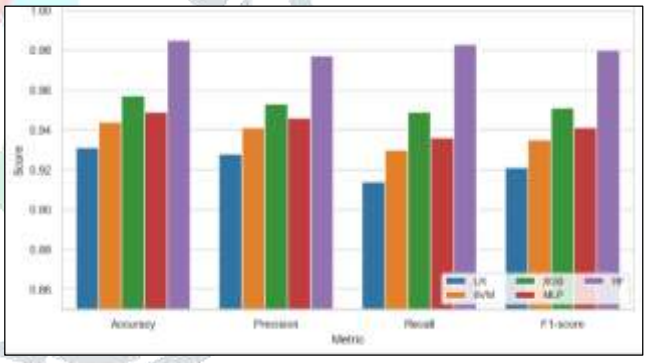

Fig 7. After Feature Engineering.

The several machine learning methods employed in this paper to detect phishing websites are compared in two charts, Figs. 6 and 7. The machine learning techniques are compared in Fig. 6 without any extra feature engineering or data enrichment (baseline value). Figure 6 illustrates how the baseline values of the LR and SVM algorithms are significantly lower than those of the other methods. Both algorithms' low recall and F1 scores show that their capacity to identify intricate phishing schemes was constrained. On the other hand, models built using ensemble and neural components performed better, but they were still constrained by the lack of further enriched features.

Using the feature engineering techniques from this work produced a consistent improvement when compared to all assessment metrics on all types of classifiers, as Fig. 7 illustrates. Given that Random Forest (RF) was the most effective classifier type and that the Ensemble Based model had the best overall performance as determined by Recall and F1 Scores, it is the classifier type that gained the most from the additional features. Because there must be as few false negatives as possible, these parameters are crucial for phishing detection.

The study's findings show that employing a multi-signal feature set improves the model's performance and that the best method for implementing phish detection models is to use ensembles during real-time phish detection.

4) Discussion of Results

As shown in Figures 6 and 7, ensemble-based classifiers outperform linear models in every assessment metric. The Recall and F1-scores of the Logistic Regression and Support Vector Machine classifiers were lower than those of the other classifiers, indicating that the nonlinear and complex character of patterns frequently linked to phishing URLs was not well captured by these linear classifiers.

Out of all the classifiers tested, the Random Forest Classifier performed the best in terms of both Recall and F1-score. This is significant since both metrics are crucial for reducing false negatives in security applications. The value of the multi-signal feature sets suggested to improve class differentiation is supported by the higher scores attained through feature engineering.

The findings imply that a high-performance method for real-time phishing URL identification across highly unbalanced datasets is produced by combining ensemble learning with robust feature representations.
datasets.

5) Practical Implications

According to the data provided, its suggested Phishing Detection Framework will function with real-world systems in a Real-Time Deployment Context, such as network security solutions, browser extensions, and email gateways. Low Inference Latency is made possible by this approach's primary focus on URL-Based characteristics and Computable at a Local Level, which boosts confidence in the incorporation of Reputation Signals into its Detection Capabilities. The suggested strategy may be applied as a workable way to lessen the impact of phishing threats on a global scale, as the performance attained shows.

V. CONCLUSION

Attackers have continuously improved their obfuscation techniques. The increasing amount of trust that users put into their online experiences creates a great environment for these types of cybercriminals.

In this research, we introduce a multi-signal framework for detecting phishing websites through two essential components: domain-level analysis through domain names, lexical characteristics, and reputation-based intelligence.

When we performed a control experiment of approximately 1,500,000 phishing sites, we found that performing feature engineering had an extensive amount of influence over the performance of each of the machine learning models that we tested in this paper. Of all the models tested, the Random Forest Model demonstrated the highest level of recall and F1 score, making it the best approach for detecting phishing attacks where it is critical to minimize false negatives. These results demonstrate that the combination of multiple complementary signals provides much greater fraud detection capability than the individual use of any of the individual feature sets.

The framework's suggested computational efficiency, interpretability, and capacity to be implemented for real-time, useful security solutions (such as browser extensions, email gateways, and network protection systems) enable it to fulfill its objectives of being a solid, comprehensible, and dependable product in the current market. In addition to enhancing current adaptive learning systems to take into account changing phishing tactics, future work will involve developing extensions to enable application in the areas of recognizing and addressing compromised genuine domains.

REFERENCES

[1] B. "Leaky kits: The increased risk of data exposure from phishing kits," B. Tejaswi, N. Samarasinghe, S. Pourali, M. Mannan, and A. Youssef, Proceedings of the APWG Symp. Electron. Crime Res. (eCrime), November 2022, pp. 1–13.

[2] APWG (2024). Report on Phishing Activity Trends. Accessed in February of 2024.

[3] N. Kumar, S. Ghewari, H. Tupsamudre, M. Shukla, and S. Lodha, "When diversity meets hostility: A study of domain squatting abuse in online banking," in Proceedings of the APWG Symp. Electron. Crime Res. (eCrime), December 2021, pp. 1–15.

[4] Y.-M. Wang, D. Beck, J. Wang, C. Verbowski, and B. Daniels, "Strider typo-patrol: Discovery and analysis of systematic typo-squatting," SRUTI, vol. 6, pp. 31–36, March 2006.

[5] N. Nikiforakis, S. Van Acker, W. Meert, L. Desmet, F. Piessens, and W. Joosen, "Bitsquatting: Exploiting bit-flips for fun, or profit?" in Proceedings of the 22nd International Conference on the World Wide Web, May 2013

[6] Sound-skwatter (did you mean: Sound-squatter?) AI-powered generator for phishing prevention," by R. Valentim, I. Drago, M. Mellia, and F. Cerutti 2023, arXiv:2310.07005.

[7] P. Kintis, N. Miramirkhani, C. Lever, Y. Chen, R. Romero-Gómez, N. Pitropakis, N. Nikiforakis, and M. Antonakakis, "Hiding in plain sight: A longitudinal study of combosquatting abuse," in Proceedings of the ACM SIGSAC Conf. Comput. Commun. Secur., October 2017, pp. 569–586.

[8] F. Quinkert, T. Lauinger, W. Robertson, E. Kirda, and T. Holz, "It's not what it looks like: Measuring attacks and defensive registrations of homograph domains," in IEEE Conf. Commun. Netw. Secur. (CNS), June 2019, pp. 259–267.

[9] H. Tupsamudre, A. K. Singh, and S. Lodha, "Everything is in the name—AURL based approach for phishing detection," in Proceedings of the International Symposium on Cybersecurity, Cryptography, and Machine Learning, 2019, pp. 231–248.

[10] Statista (2009). Phishing attacks targeted legitimate brands between January 2009 and October 2022. February 2024. of brands-hijacked-by-phishing-attacks/

[11] StationX (2024). Top Phishing Statistics for 2024: Current Data and Patterns. [Online] Accessed: February 2024.

[12] R. Goenka, M. Chawla, and N. Tiwari, "A comprehensive survey of phishing: Mediums, intended targets, attack and defense techniques and a novel taxonomy," Int. J. Inf. Secur., vol. 23, no. 2, pp. 819–848, April 2024

[13] O.K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," Expert Syst. Appl., vol. 117, pp. 345–357, March 2019.

[14] B. B. Gupta, K. Yadav, I. Razzak, K. Psannis, A. Castiglione, and X. Chang, ‘‘A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment,’’ Comput. Commun., vol. 175, pp. 47–57, Jul. 2021.

[15] A. K. Jain and B. B. Gupta, ‘‘A machine learning based approach for phishing detection using hyperlinks information,’’ J. Ambient Intell. Humanized Comput., vol. 10, no. 5, pp. 2015–2028, May 2019.

[16] A.Aljofey,Q.Jiang,A.Rasool,H.Chen,W.Liu,Q.Qu,andY.Wang,‘‘An effective detection approach for phishing websites using URL and HTML features,’’ Sci. Rep., vol. 12, no. 1, May 2022, Art. no. 8842.

[17] A. K. Jain, S. Parashar, P. Katare, and I. Sharma, ‘‘PhishSKaPe: A content-based approach to escape phishing attacks,’’ Proc. Comput. Sci., vol. 171, pp. 1102–1109, Jan. 2020.

[18] C. C. L. Tan, K. L. Chiew, K. S. C. Yong, Y. Sebastian, J. C. M. Then, and W.K.Tiong, ‘‘Hybrid phishing detection using joint visual and textual identity,’’ Expert Syst. Appl., vol. 220, Jun. 2023, Art. no. 119723.

[19] R. V. Valentim, I. Drago, M. Mellia, and F. Cerutti, ‘‘X-squatter: AI multilingual generation of cross-language sound-squatting,’’ ACM Trans. Privacy Secur., vol. 27, no. 3, pp. 1–27, Aug. 2024, doi: 10.1145/3663569.

[20] A. Karim, M. Shahroz, K. Mustofa, S. B. Belhaouari, and S. R. K. Joga, ‘‘Phishing detection system through hybrid machine learning based on URL,’’ IEEE Access, vol. 11, pp. 36805–36822, 2023.

[21] N. Kumar, S. Ghewari, H. Tupsamudre, M. Shukla, and S. Lodha, ‘‘When diversity meets hostility: A study of domain squatting abuse in online banking,’’ in Proc. APWG Symp. Electron. Crime Res. (eCrime), Dec. 2021, pp. 1–15.

[22] A. Costello. (2003). Punycode: A Bootstring Encoding of Unicode for Internationalized Domain NamesinApplications(IDNA).IETF.Accessed: Jan. 20, 2024. [Online]. Available: https://tools.ietf.org/html/rfc3492

[23] T. Koide, N. Fukushi, H. Nakano, and D. Chiba, ‘‘PhishReplicant: A language model-based approach to detect generated squatting domain names,’’ in Proc. Annu. Comput. Secur. Appl. Conf., Dec. 2025, pp. 1–13.

[24] K. Haynes, H. Shirazi, and I. Ray, ‘‘Lightweight URL-based phishing detection using natural language processing transformers for mobile devices,’’ Proc. Comput. Sci., vol. 191, pp. 127–134, Jan. 2021.

[25] A. Prasad, S. Chandra, M. Uddin, T. Al-Shehari, N. A. Alsad han, and S. Sajid Ullah, ‘‘PermGuard: A scalable framework for Android malware detection using permission-to-exploitation mapping,’’ IEEE Access, vol. 13, pp. 507–528, 2025, doi: 10.1109/ACCESS.2024.3523629.