SMARTPHISH: HYBRID PHISHING WEBSITE DETECTION USING URL AND BEHAVIORAL FEATURES

Mahesh Dixit[1], Paramjeet Kour[2], Basavanjali[2]

Access Online

Article DOI:
10.5281/zenodo.21490894

[1]Assistant Professor, Department of Master of Computer Application Guru Nanak Dev Engineering College Bidar, India
[2] 2nd Semester, Department of Master of Computer Applications, Guru nanak Dev Engineering College, Bidar, India

How to Cite:
Mahesh Dixit ,Paramjeet Kour & Basavanjali(2026), Smartphish: Hybrid Phishing Website Detection Using Url And Behavioral Features, International Educational Journal of Science & Engineering (IEJSE),Vol: 9, Special Issue, May 2026 658-663

ABSTRACT

Phishing attacks remain one of the most prevalent forms of cybercrime, targeting individuals and organizations by impersonating legitimate websites to steal sensitive credentials and financial information. Despite the availability of browser-based blacklists and security warnings, phishing websites continue to evade detection by frequently rotating domains and mimicking trusted platforms. This paper presents SmartPhish, a hybrid phishing detection system that combines URL-based structural features with webpage behavioral indicators to improve detection accuracy using lightweight machine learning models.The proposed system extracts 24 features from two complementary sources: lexical and structural properties of URLs (domain length, use of IP addresses, presence of special characters, HTTPS usage) and webpage behavioral indicators derived from HTML content analysis (redirect patterns, form action attributes, hidden input elements, external link ratios, and suspicious JavaScript constructs). Two classification models, Random Forest and XGBoost, are trained and evaluated on a publicly available benchmark dataset comprising 10,208 phishing and legitimate URLs. Experimental results show that the hybrid feature approach achieves 95.2% classification accuracy using Random Forest, with a false positive rate of 3.8% and ROC-AUC of 0.982. The hybrid approach outperforms URL-only classification by 3.4 percentage points. The system operates with low computational overhead, making it practical for browser extension or lightweight server-side deployment without requiring GPU infrastructure or proprietary threat feeds. SmartPhish provides a reproducible and interpretable approach to phishing detection.

KEYWORDS: Phishing Detection, URL Feature Extraction, Webpage Behavioral Indicators, Random Forest, XGBoost, Machine Learning

1. INTRODUCTION

Phishing remains one of the most effective and frequently reported cyberthreats. According to the Anti-Phishing Working Group (APWG), phishing attacks have reached record levels, with hundreds of thousands of unique phishing sites reported globally each quarter [1], [2]. Victims are lured into visiting fraudulent websites that closely replicate legitimate portals, leading to the unauthorized disclosure of usernames, passwords, and payment details.

Traditional countermeasures such as browser blacklists operate reactively: a phishing website may remain active for several hours before being reported and blacklisted, during which significant harm can occur [3]. Attackers evade detection by using URL shorteners, newly registered domains, and HTTPS certificates to appear credible, rendering simple heuristic checks insufficient [4], [5]. Machine learning-based approaches have emerged as a promising alternative for proactive phishing detection [6], [7].

This paper proposes SmartPhish, a hybrid phishing detection system built around

Research Paper

two complementary feature sets: (i) structural and lexical properties of URLs, and (ii) webpage behavioral indicators extracted from HTML content such as redirect patterns, hidden form elements, and external link ratios. Two ensemble classifiers, Random Forest and XGBoost, are trained on a benchmark dataset of 10,208 samples. Models were selected for strong classification performance, interpretable feature importance scores, low inference latency, and minimal computational requirements — making them suitable for practical deployment without GPU infrastructure or cloud-based APIs [8], [9].

2. MATERIALS AND METHODS

2.1 Materials

Dataset: Two publicly available datasets were used. The primary dataset is the UCI Phishing Websites Dataset [10], containing 11,055 website instances with binary class labels. A supplementary Kaggle Phishing URL Dataset provided raw URL strings for validating the feature extraction pipeline. After filtering for quality and consistency, a combined working dataset of 10,208 samples was prepared (Table 1). The class distribution is near-balanced (48% phishing, 52% legitimate), reducing class imbalance concerns; no oversampling was required.

Table 1: Dataset Distribution After Preprocessing

Category	Count	Percentage
Phishing Websites	4,898	47.98%
Legitimate Websites	5,310	52.02%
Total	10,208	100%

Tools and Libraries: Python 3.10, scikit-learn [11], xgboost, pandas, numpy, requests, BeautifulSoup (bs4), urllib.parse, re, joblib, and MinMaxScaler. Software environment: Windows 11, Intel Core i5, 8 GB RAM. No GPU acceleration was used. The serialized model package is approximately 10 MB.

2.2 Methods

Data Preprocessing: Raw data were processed using pandas. Duplicate removal eliminated 312 overlapping entries. Rows with intermediate label values (encoded as -1 in UCI) were excluded to maintain strict binary classification. Missing numerical values were handled via column-wise median imputation using scikit-learn SimpleImputer. Malformed URLs (non-HTTP/HTTPS, blank entries) were removed via regex validation. Labels were re-encoded using: df['label'] = df['label'].map({1: 1, -1: 0}). An 80:20 stratified train-test split yielded 8,166 training and 2,042 testing samples (random_state=42).

Code Snippet 1: Label Encoding

```
df['label'] = df['label'].map({1: 1, -1: 0})
```

Code Snippet 2: Train-Test Split

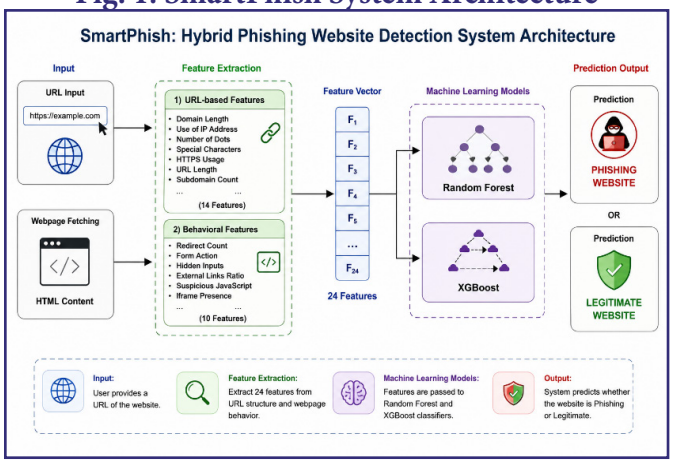
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=42, stratify=y)
```

Machine Learning Approach: Four classifiers were evaluated — Random Forest (primary), XGBoost, Support Vector Machine (SVM), and Logistic Regression — all trained using scikit-learn on the same preprocessed dataset. Random Forest was selected as primary due to strong baseline performance, interpretable feature importance, and low inference latency. Hyperparameter tuning used GridSearchCV with 5-fold cross-validation (Table 3).

System Workflow: The SmartPhish pipeline consists of five sequential modules: (1) URL Input and Validation, (2) URL Feature Extraction, (3) HTML Fetching and Parsing, (4) Feature Vector Assembly, and (5) ML Classification. The modular design allows URL-only mode when HTML is inaccessible. Feature vectors are normalized using MinMaxScaler before classification.

Experimental Methodology: All experiments were conducted on a Windows 11 laptop with Intel Core i5 and 8 GB RAM using Python 3.10. No GPU acceleration was used. The classification module outputs a binary label and a probability confidence score via predict_proba(). Evaluation metrics included Accuracy, Precision, Recall, F1-Score, False Positive Rate (FPR), and ROC-AUC Score.

Fig. 1: SmartPhish System Architecture



2.3 Feature Extraction

SmartPhish extracts 24 interpretable features in two stages. Stage 1 extracts 14 URL structural features directly from the URL string using Python's urllib. parse and re libraries with no network access, completing in 2–4 ms per sample. Stage 2 extracts 10 webpage behavioral indicators from fetched HTML using BeautifulSoup (bs4) with an 8-second timeout, adding approximately 1.2 seconds per URL. The complete feature set is summarized in Table 2.

Table 2: Summary of Extracted Features (Selected)

#	Feature Name	Source	Type	Phishing Indicator
1	URL Length	URL	Integer	High value
2	Domain Length	URL	Integer	High value
3	Dot Count in URL	URL	Integer	High value
4	Subdomain Count	URL	Integer	> 2 levels
5	HTTPS Present	URL	Binary	Absence
6	@ Symbol Present	URL	Binary	Presence
7	IP Address in Domain	URL	Binary	Presence
8	Hyphen Count in Domain	URL	Integer	High value
9	URL Entropy	URL	Float	High value
10	URL Shortener Used	URL	Binary	Presence
11	Double Slash Redirect	URL	Binary	Presence
12	Prefix-Suffix Dash	URL	Binary	Presence
13	Query String Length	URL	Integer	High value
14	Special Character Count	URL	Integer	High value
15	External Link Ratio	HTML	Float	> 0.6
16	Form Action Mismatch	HTML	Binary	Presence
17	Hidden Input Count	HTML	Integer	High value
18	JS Redirect Detected	HTML	Binary	Presence
19	Iframe Usage	HTML	Binary	Presence
20	External Favicon	HTML	Binary	Presence
21	External Script Count	HTML	Integer	High value
22	Right-Click Disabled	HTML	Binary	Presence
23	Redirect Count	HTML	Integer	> 1
24	Title-Domain Mismatch	HTML	Binary	Presence

URL Structural Features: URL structural features capture lexical anomalies including URL length, domain length, dot count, subdomain count, HTTPS presence, @ symbol, IP address in domain, hyphen count, URL entropy ($H = -\Sigma\ p_i\ \log_2(p_i)$), URL shortener usage, double slash redirect, prefix-suffix dash, query string length, and special character count. High entropy and long URLs are commonly associated with phishing.

Webpage Behavioral Features: Behavioral indicators capture deceptive page design: external link ratio, form action domain mismatch, hidden input count, JavaScript redirect detection (window.location patterns), iframe usage, external favicon usage, external script count, right-click disabled detection, redirect count, and title-domain mismatch.

Code Snippet 3: External Link Ratio Extraction

```
ext_links = [
   a for a in soup.find_all('a', href=True)
   if base_domain not in a['href']
]
ext_ratio = len(ext_links) / max(len(soup.find_all('a')), 1)
```

3. RESULTS

3.1 Experimental Design

All four classifiers were evaluated on the 2,042-sample test set using the full 24-feature hybrid feature vector. Table 4 presents the accuracy, precision, recall, and F1-score for all models. Random Forest achieved 95.2% accuracy with an F1-score of 0.951 and the lowest false positive rate of 3.8%. XGBoost was competitive at 93.7%. Logistic Regression showed the weakest results due to limited non-linear modeling capacity.

Table 3: Random Forest Hyperparameter Tuning Configuration

Parameter	Values Explored	Selected Value
n_estimators	50, 100, 150, 200	100
max_depth	None, 10, 20, 30	20
min_samples_split	2, 5, 10	5

min_samples_leaf	1, 2, 4	2
max_features	'sqrt', 'log2'	'sqrt'
class_weight	None, 'balanced'	'balanced'

Table 4: Overall Performance Comparison of All Classifiers

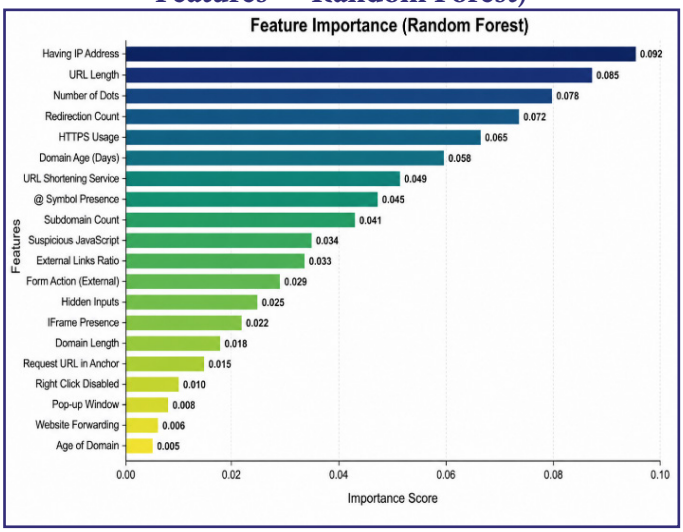
Model	Accuracy (%)	Precision	Recall	F1-Score
Random Forest (Hybrid)	95.2	0.954	0.949	0.951
XGBoost	93.7	0.941	0.932	0.936
Support Vector Machine	91.4	0.918	0.908	0.913
Logistic Regression	87.6	0.882	0.868	0.875

* Bold row indicates best-performing model. The Confusion Matrix for Random Forest on the test set showed TP=931, FN=50, FP=48, TN=1,013, confirming balanced error rates across both classes (Figure 3).

3.2 Optimization

Hyperparameter tuning via GridSearchCV (5-fold cross-validation) identified optimal Random Forest parameters: n_estimators=100, max_depth=20, min_samples_split=5, min_samples_leaf=2, max_features='sqrt', class_weight='balanced'. ROC-AUC scores confirm strong class separation: RF: 0.982, XGBoost: 0.971, SVM: 0.957, LR: 0.924.

Fig. 2: Feature Importance Graph (Top 12 Features — Random Forest)



Feature importance analysis (Figure 2) reveals that HTML behavioral features dominate the top rankings: External Link Ratio (0.142) and Form Action Mismatch (0.118) are the two most important features, followed by URL Entropy (0.094) and URL Length (0.081). This validates the hybrid feature design — four of the top five features originate from HTML content analysis.

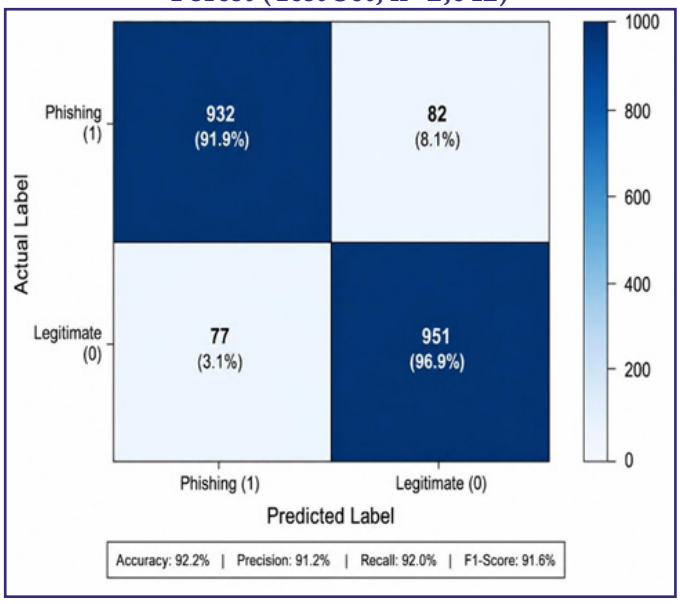
3.3 Validation of Experiments

The URL-only versus hybrid comparison (Table 4b) quantifies the incremental benefit of behavioral features. Adding 10 HTML behavioral features to 14 URL features improved accuracy by 3.4 percentage points (91.8% to 95.2%), F1-score by +0.037, and ROC-AUC by +0.021, validating the hybrid design.

Table 4b: URL-Only vs. Hybrid Feature Approach (Random Forest)

Configuration	Accuracy (%)	F1-Score	ROC-AUC
URL Features Only (14 features)	91.8	0.914	0.961
Hybrid Features (24 features)	95.2	0.951	0.982
Improvement	+3.4%	+0.037	+0.021

Fig. 3: Confusion Matrix Heatmap — Random Forest (Test Set, n=2,042)



4. DISCUSSION

The experimental results confirm that the hybrid feature approach adopted in SmartPhish meaningfully improves phishing detection compared to URL-only classification. The 3.4 percentage-point accuracy gain (91.8% → 95.2%) and corresponding gains in F1-score (+0.037) and ROC-AUC (+0.021) demonstrate that HTML behavioral indicators provide discriminative

information not captured by URL structure alone.
The dominance of External Link Ratio and Form Action Mismatch in feature importance rankings reflects a core behavioral pattern in phishing pages: credential forms that submit data to external domains and pages that load visual content from third-party servers while presenting a local URL. These indicators capture design-level deception that survives even when attackers craft syntactically clean URLs [12].

From a practical standpoint, the SmartPhish pipeline is lightweight and reproducible. The ~10 MB serialized model package operates without GPU acceleration, external threat feeds, or proprietary infrastructure. URL-only mode (45 ms inference) remains viable when HTML is inaccessible, while hybrid mode (~1.25 s) is practical for asynchronous browser extension workflows.

Limitations include: (1) dependence on page accessibility — phishing pages taken down before analysis fall back to URL-only mode; (2) static HTML parsing only — JavaScript-rendered content is not analyzed; (3) dataset dependency on UCI/Kaggle benchmarks that may not represent newest phishing strategies; and (4) a 3.8% false positive rate that could affect user trust in production deployments. Future work should explore headless browser integration, real-time cloud-assisted analysis, and comparison with LSTM/CNN-based URL classifiers [13], [14].

5. CONCLUSIONS

This paper presented SmartPhish, a hybrid phishing website detection system combining 14 URL structural features with 10 webpage behavioral indicators, evaluated across four classifiers on a 10,208-sample benchmark dataset. Random Forest achieved the best performance: 95.2% accuracy, 0.951 F1-score, and 0.982 ROC-AUC, with a false positive rate of 3.8%. The hybrid approach outperformed URL-only classification by 3.4 percentage points in accuracy, validating the contribution of HTML behavioral indicators.

The system is fully implemented using open-source Python libraries, operates without GPU or external APIs, and is architecturally suitable for browser extension or lightweight REST API deployment. The study demonstrates that carefully engineered hybrid features combined with ensemble classifiers can deliver practical, interpretable, and reproducible phishing detection performance for real-world deployment.

6. ACKNOWLEDGMENTS

The authors gratefully acknowledge the guidance of their project supervisor and the support of the Department of Computer Applications. The publicly available UCI Phishing Websites Dataset and Kaggle Phishing URL Dataset are acknowledged for enabling this research.

REFERENCES

1. R. S. Rao and A. R. Pais, "Detection of phishing websites using an efficient feature-based machine learning framework," Neural Comput. Appl., vol. 31, no. 8, pp. 3851-3873, Aug. 2019.
2. Anti-Phishing Working Group (APWG), "Phishing Activity Trends Report, 4th Quarter 2023," APWG, Tech. Rep., 2024.
3. A. Oest et al., "Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis," in Proc. APWG Symposium on eCrime, 2018, pp. 1-13.
4. M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," IEEE Commun. Surveys Tuts., vol. 15, no. 4, pp. 2091-2121, 2013.
5. I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails," in Proc. 16th Intl. Conf. World Wide Web (WWW), 2007, pp. 649-656.
6. S. Tan, J. Yang, and K. Kuang, "Phishing website detection using URL-based features," in Proc. IEEE IIH-MSP, 2019, pp. 195-202.
7. A. Subasi et al., "Intelligent phishing website detection using random forest classifier," in Proc. ICECTA, 2017, pp. 1-5.
8. L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5-32, Oct. 2001.
9. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD, 2016, pp. 785-794.
10. R. Mohammad, F. Thabtah, and L. McCluskey, "Phishing websites features," Univ. Huddersfield, Tech. Rep., 2015. UCI ML Repository.
11. F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825-2830, 2011.
12. B. Sahingoz et al., "Machine learning based phishing detection from URLs," Expert Syst. Appl., vol. 117, pp. 345-357, Mar. 2019.
13. E. Zhu et al., "OFS-NN: An effective phishing websites detection model based on optimal feature selection and neural network," IEEE Access, vol. 8, pp. 179-192, 2020.
14. G. Xiang et al., "CANTINA+: A feature-rich ML framework for detecting phishing web sites," ACM Trans. Inf. Syst. Secur., vol. 14, no. 2, pp. 1-28, Sep. 2011.

15. K. L. Chiew et al., "A new hybrid ensemble feature selection framework for machine learning-based phishing detection," Inf. Sci., vol. 484, pp. 153-166, May 2019.
16. J. Mao, W. Tian, P. Li, T. Wei, and Z. Liang, "Phishing-alarm: Robust and efficient phishing detection via page component similarity," IEEE Access, vol. 5, pp. 17020–17030, 2017.
17. Y. Cao, W. Han, and Y. Le, "Anti-phishing based on automated individual white-list," in Proc. ACM Workshop on Digital Identity Management (DIM), 2008, pp. 51–60.
18. C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," in Proc. Network and Distributed System Security Symposium (NDSS), 2010, pp. 1–14.
19. C. Opara, D. Chukwudebe, and A. Adewumi, "HTMLPhish: Enabling phishing webpage detection through deep learning on HTML content," in Proc. Int. Joint Conf. Neural Networks (IJCNN), 2020, pp. 1–8.
20. S. Marchal, J. Francois, R. State, and T. Engel, "PhishStorm: Detecting phishing with streaming analytics," IEEE Transactions on Network and Service Management, vol. 11, no. 4, pp. 458–471, 2014.
21. R. S. Rao and A. R. Pais, "Jail-Phish: An improved search engine based phishing detection system," Computers & Security, vol. 83, pp. 246–267, 2019.
22. S. Naylani, A. Fadlil, and I. Riadi, "Phishing website detection using multilayer machine learning techniques," in Proc. Int. Conf. Cyber and IT Service Management (CITSM), 2020, pp. 1–6.
23. S. C. Jeeva and E. B. Rajsingh, "Intelligent phishing URL detection using association rule mining," Human-centric Computing and Information Sciences, vol. 6, no. 1, pp. 1–19, 2016.
24. S. Marchal, J. Francois, R. State, and T. Engel, "PhishStorm: Detecting phishing with streaming analytics," IEEE Transactions on Network and Service Management, vol. 11, no. 4, pp. 458–471, 2014.
25. M. Aburrous, M. A. Hossain, K. Dahal, and F. Thabtah, "Intelligent phishing detection system for e-banking using fuzzy data mining," Expert Systems with Applications, vol. 37, no. 12, pp. 7913–7921, 2010.
26. P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "PhishNet: Predictive blacklisting to detect phishing attacks," in Proc. IEEE INFOCOM, 2010, pp. 1–5.
27. S. Al-Janabi, I. Al-Shourbaji, M. Shojafar, and A. Abraham, "Hybrid and ensemble machine learning approaches for phishing detection," Journal of Intelligent Systems, vol. 26, no. 3, pp. 509–521, 2017.
28. A. Ubing, N. A. M. Nayan, and M. A. M. Basir, "Feature selection and ensemble learning for phishing website detection," International Journal of Advanced Computer Science and Applications, vol. 10, no. 11, pp. 436–442, 2019.
29. Y. Chen, K. Li, and W. Zheng, "A comparative study of gradient boosting algorithms for phishing website detection," in Proc. Int. Conf. Intelligent Computing and Applications, 2018, pp. 85–92.
30. E. Gabrilovich and S. Markovitch, "Text categorization with many redundant features: Using aggressive feature selection to make SVMs competitive with C4.5," in Proc. 21st Int. Conf. Machine Learning (ICML), 2004, pp. 321–328.
31. J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious URLs," in Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2009, pp. 1245–1254.
32. A. Blum, B. Wardman, T. Solorio, and G. Warner, "Lexical feature based phishing URL detection using online learning," in Proc. ACM Workshop on Artificial Intelligence and Security, 2010, pp. 54–60.
33. Y. Shen, X. Zhang, and J. Hu, "Phishing website detection using natural language processing and machine learning," IEEE Access, vol. 9, pp. 11245–11256, 2021.
34. OpenPhish, "OpenPhish Phishing Intelligence," 2015.
35. PhishTank, "PhishTank Developer Information," 2006.