iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XI **Month of publication:** November 2025

DOI: https://doi.org/10.22214/ijraset.2025.75389

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

Phishing Website Detection Using Artificial Intelligence

Nidhi Malik[1], Lakshay Sharma[2], Priyansh Sharma[3], Manav Jindal[4], Bharat[5]
[1]*Department of Computer Science & Engineering, HMR Institute of Technology & Management, GGSIPU Delhi, India*
[2, 3, 4, 5]*Department of Computer Science & Engineering (Cyber Security), HMR Institute of Technology & Management, GGSIPU Delhi, India*

Abstract: ***Phishing has turned out to be one of the most common and harmful cybercrimes, in which attackers deceive users into divulging important credentials, such as passwords, banking details, personal information, by spoofing legitimate websites. Traditional defence mechanisms, such as blacklisting-based and rule-based approaches, are reactive in nature and more often than not remain unable to detect newly created or rapidly evolving phishing sites, since attackers are easily modify URLs or elements within a page, which may escape static filters.***

In this regard, this paper proposes an effective real-time phishing detection system powered by Artificial Intelligence, which makes use of Machine Learning and Deep Learning models to dynamically analyze and classify websites based on multiple features. The proposed system will extract and process several indicators from a website, including URL structure, lexical and host-based features, webpage content, HTML and JavaScript behaviour, SSL certificate data, domain registration information, to identify whether a site is legitimate or malicious.

The architecture consists of three main components:

- ***Feature Extraction Layer: Real-time data is gathered from URLs and web pages***
- ***AI Classification Layer: This layer utilizes the pre-trained ML/DL models, such as Random Forest, Support Vector Machine, or Deep Neutral Networks, for website classification.***
- ***Decision Layer: Provides immediate feedback or alerts to the users for protection proactively.***

It leverages public phishing datasets like Phish Tank, UCI Machine Learning Repository, and Alexa Top Sites for training and testing. It applies data pre-processing techniques, feature selection methods, and model optimization to enhance accuracy and minimize false alarms. The performance of the system against real-time attacks can be corroborated based on the following metrics: accuracy, precision, recall, F-1 score, ROC-AUC, and detection latency.

These experimental results demonstrate that the AI-based system significantly outperforms traditional blacklist and heuristic methods in terms of detection accuracy with high adaptability against zero-day phishing attacks. The findings have important implications for the use of AI as an active, effective, and real-time defence mechanism in contemporary cyber security frameworks.

I. INTRODUCTION

The threat of phishing in the modern digital landscape continues to be both pervasive and challenging to address. It is an attempt to obtain sensitive information, usually login or financial information or personal data, by the creation of fraudulent websites or emails. Usually, in these kinds of cyber-attacks, the attacker seeks to impersonate well-known organizations, such as banks, online shopping portals, and government departments, with the objective of manipulating targets into feeling they are interacting with a well-recognized source. Once the victim has entered their private information, it is captured and used for identity theft, financial fraud, or unauthorized system access. According to recent cyber security reports, phishing attacks account for more than 36% of all data breaches across the globe, which means their prevalence and sophistication continue to increase. Unlike traditional malware attacks relying on malicious code execution, phishing relies on psychological manipulation and earned trust making it hard to detect using conventional security tools. Besides, it is relatively easy to create and host phishing sites, while quickly changing URLs, names, or templates of a website allows cyber attackers to easily bypass the traditional methods of detection.

Traditional security mechanisms, like firewalls, Ids, antivirus software, and blacklist-based URL filters, are based mostly on known signatures or static rules. While these methods can detect previously identified phishing sites, they cannot detect zero-day phishing attacks nor newly generated malicious URLs that have not been reported yet. Such systems also cannot dynamically respond to ever changing tactics employed by cyber criminals.

Due to these challenges, AI, specifically Machine Learning and Deep Learning, has emerged as a promising solution for real-time phishing detection. AI-based techniques can learn complex patterns and relationships from a large volume of datasets of both legitimate and phishing URLs autonomously. Through the analysis of various features including URL structure, domain age, webpage content, metadata, SSL certificates, and network behaviour, the AI system can identify phishing attempts even if it is obfuscated or the attackers try to mimic a legitimate website.

Machine Learning algorithms, such as Random Forest, Support Vector Machine (SVM), Decision Trees, and gradient Boosting, can classify the websites with high accuracy based on the extracted features. On the other hand, deep learning models like Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN) can learn much deeper semantic and temporal patterns, which can enable robust detection in dynamic and large scale environments.

In addition, AI-powered systems can be directly integrated with web browsers, email gateways, and security platforms for real-time analysis and alerting well before the users access such malicious websites. This proactive approach adds to cyber security resilience by decreasing dependence on manually updated blacklists and optimizing response times against emerging threats. In all, phishing remains a critical cyber security problem that requires intelligent, adaptive, and real-time solutions. AI driven phishing detection is easy scalable and effectively identifies deceptive websites with minimum human interference. This makes AI integral in combating phishing and other cyber deceptions with the combination of automated learning, continuous model updates, and real-time decision-making.

II. LITERATURE REVIEW

Detection of phishing websites has been an active area of cyber security, with much research in the field applying Machine Learning and Deep Learning to enhance both accuracy and adaptability. Early systems employed heuristic and rule-based methods that used hand-crafted patterns together with blacklists of known malicious URLs. However, these methods lacked the ability to detect those newly appearing phishing websites that have not yet been reported or indexed. Lacking such generalizable methods, early works began to shift toward data-driven AI capable of discovering phishing behaviour through feature learning rather than predefined rules.

A. Machine Learning-Based Approaches

A number of traditional ML algorithms have been proven effective in several studies to classify phishing and legitimate websites.

Almomani et al. (2013) developed a phishing detection system that was based on lexical features of the URL, such as URL length, number of special characters, presence of subdomains, and usage of IP address instead of name. Their model, using algorithms like Decision Trees and Random Forests, attained a classification accuracy of about 95% and hence proved that lexical analysis could serve as a good discriminator between phishing URLs and benign ones.

Mohammad et al. (2014) developed a hybrid model that merged rule-based techniques with machine learning classifiers to improve the reliability of detections. The system extracted features from several dimensions, including URL-based, host-based, and content-based attributes. The integration of heuristic rules with models like Support Vector Machines and Naive Bayes led to better performance and reduced false positives for the authors, compared to pure ML models. The approach is important for producing system that are capable of dealing with diversified phishing tactics.

Other works introduced different algorithms such as k-Nearest neighbours (k-NN), Logistic Regression, and Gradient Boosting, using training datasets from PhishTank, UCI Machine Learning Repository, and Alexa Top Sites. These work combined to point out that ML algorithms can find subtle differences in URL structures and domain characteristics that could provide the basis for automated phishing detection.

B. Deep Learning-Based Approaches

As computational power grew and data become increasingly available, for instance, researchers started embracing DL techniques to overcome the limitation of manual feature extraction in ML models. For example, a DL model can automatically learn complex hierarchical representations from raw data, such as text sequences or webpage contents, without the need for handcrafted features.

Jain and Gupta (2018) proposed a framework based on deep learning using CNNs and RNNs to analyze static and dynamic features of web pages. CNNs were used to extract spatial patterns in webpage layout and structural analysis of HTML documents, whereas RNNs were employed to learn sequential dependencies in URLs and textual contents, especially LSTM networks. The study showed the adaptability of deep learning models to changes in phishing strategies for better performance compared to traditional ML techniques.

Several further studies have integrated NLP into phishing detection systems using various techniques. NLP thus enables the model to inspect webpage text, URLs, and metadata for semantic and contextual cues. The performance of hybrid AI models, integrating NLP with DL, has increased in the detection of deceptive linguistic patterns in real-time environments and improves the chances of detecting phishing. For example, systems using word embedding (Word2Vec, GloVe) or transformer-based architectures (BERT, GPT) have achieved superior detection rates by understanding the context and intent behind the textual content of phishing sites.

C. Hybrid and Ensemble Methods

Recent trends in research have focused on the combination of several AI techniques for robustness and scalability, such as ensemble ML models or hybrid DL architectures. Ensemble methods, combining outputs from classifiers like random Forest, Gradient Boosting, and Neural Networks, tend to outperform individual algorithms by reducing bias and variance. In the same way, integrating feature-based, content-based, and behavioural analysis into one framework enables more comprehensive phishing detection with the capability to handle zero-day attacks. Overview in general, the literature on phishing detection strategies indicates a very clear development: from static rule-based systems to an intelligent and adaptive AI – driven solution. The proposed use of machine learning models created the base for automated classification, while deep learning and NLP models enabled not only real-time detection but also better contextual understanding. These all reflect an increasing potential of AI in combating phishing threats and form the foundation for developing the proposed real-time phishing detection system discussed in this paper.

III. METHODOLOGY

This methods section describes a rigorous, reproducible procedure for building and evaluating a phishing-URL detection system. It is recognized into four stages: Data collection, Feature Extraction, Model training, and Real-time Detection. Besides the above this also discusses the design of the evaluation, deployment considerations, and ethical/privacy safeguards. No code is included; the goal is to provide sufficient procedural detail so that another researcher can implement this pipeline and reproduce experiments.

A. Data Collection

Objectives

Collect a diverse set of labelled examples of phishing and legitimate URLs (and optional page artifacts) that reflect real world variability, support temporal evaluation, and enable reproducible experiments.

Sources and explanation:

- Phishing feeds/community lists: PhishTank,OpenPhish,Abuse.ch- community/analyst-verified phishing samples are provided for up-to-date attack patterns.
- Benign site lists: Alexa Top sites, Common Crawl, Majestic- representative of legitimate web properties.
- Public datasets: Kaggle or academic corpora containing labelled URL data useful for benchmarking and reproducibility.
- Enterprise Telemetry/ internal logs: If available, to capture real user exposure and corporate context – with strict privacy controls.
- Threat intelligence feeds/blocklists: augment the ground truth and provide metadata, such as blacklists and campaign tags.

B. Procedure for collection

1) Define the time window for collection and measure exact timestamps for every sample-colections_ts and first_seen_ts. Record source provenance at each URL.

2) Fetch raw artifacts, if possible: HTTP headers, HTML, rendered screenshots, and redirect chains. Store these as optional raw artifacts separate from the main feature store.

3) Canonicalizee URLs at ingestion time by applying RFC-compliant parsing, removing default ports, normalizing percent-encoding, and punycode consistently.

4) Remove duplicates and near duplicates (canonical forms). Optionally DE duplicate by host+path or by content hash depending on the experiments.

5) Keep a metadata field or label confidence e.g., verified by more than one feed, single-source, manually validated. Labelling, quality and timeline considerations

Prefer temporal labelling: label according to evidence at collection time and keep the timestamp to enable time-aware evaluation.

Where labels are noisy- community feeds: sample subsets for manual verification to estimate label noise rates.

The system should keep an explicit "retired/expired" flag for URLs that later become inactive or get re-assigned, so experiments can be stratified by domain lifetime.

C. Dataset splits and sampling

Use time-based splits: train or older periods validate and test on newer periods to mimc real deployment, and measure concept drift.
Recommended split by time: e.g.,70%training (earliest timeframe), 15% validation, 15% test (most recent).
For cross-validation use expanding window or walk-forward validation rather than random k-fold to preserve the ordering in time.

IV. FEATURE EXTRACTION

A. Purpose

Create a robust set of features predictive of phishing and resistant to simple evasion. Features are categorized by cost and availability: lexical, host/network, and content/dynamic, in order of low to high cost.

1) Feature categories(examples)

A. Lexical (URL-based, low cost- available synchronously) Whole URL length, path length
Special characters count: dots'.', hyphens '-','@','?', '%', '=', '_'
Numbers of path or query tokens; token n-grams (characters n-grams and token-grams)
Presence of IP address in host: (Boolean)
Entropy metrices of hostname- Shannon entropy- to discover randomness
Suspicious substrings; examples include: 'verify', 'account', 'login', 'secure'
Brand token similarity score (fuzzy matching)
Utilization of uncommon or new TLDs

2) Host/Network (requires lookups-medium cost)

Domain age (WHOIS creation date- age in days)
Time until expiry, registrar name, and registrar reputation features
DNS characteristics: TTLs, presence of CNAME chains, number of A/AAAA records, MX records
IP-based features: ASN, geolocation country, hosting provider type (shared hosting vs cloud provider)
Presence in known blacklists or threat intelligence feeds
SSL/TLS certificate metadata such as issuer, expiry, self-signed

3) Content-based (requires page fetch/ parsing – higher cost)

HTML structure features: number of forms, number of input fields, presence of password field, iframe usage
Features extracted from the script included: Number of inline scripts, obfuscation patterns – eval, long base64 strings
Resource distribution: external domain ration (number of external resource domains/ total resources)
Textual features: bag-of-words, TF-IDF, or lightweight embedding of page text
Visual similarity features: perceptual-hash (phash) against brand homepages or known templates

4) Dynamic / Behavioural (sandboxed analysis- highest cost)

Redirection chain length and redirection domains
Runtime behaviours in JS(JavaScript): network calls, form auto-submits, presence of key-logger- like listeners
Cloaking detection: divergent content served to benign vs. simulated crawlers/user agents

B. Extraction Pipeline & Engineering Practices

Synchronous fast path: compute lexical features immediately to allow low-latency decisions.
Asynchronous deep path: Schedule host lookups, content fetches, and sandboxed runs to run in worker pools; merge results to update records or trigger reclassification if necessary.
Caching: use TTL-aware caching for DNS/WHOIS/reputation to reduce cost and latency.
Normalization & encoding: standardize numeric features (robust scaling), one-hot or target encoding for categorical, and use sparse representations for large token spaces.

Missingness: include explicit missing-value indicators, such as 'whois_missing = 1', rather than blind imputation since the absence may be informative.
Adversarial hardening: punycode normalisation, Unicode homoglyph canonicalisation when appropriate, including features capturing obfuscation, such as mixed script use.
Feature catlog: Keep a documented feature catalog which includes information like feature name, type, computation cost, expected range, and known failure modes.

V. MODEL TRAINING

A. *Objectives*

Train models that generalize to new phishing tactics, provide calibrated confidence scores, and are interpretable enough for operator trust.

B. *Candidate model families*

Baselines: Logistic Regression, Decision Trees- fast, interpretable benchmarks.
Tree ensembles: Random Forests, Gradient Boosted Decisions Trees (XGBoost/ LightGBM/ CatBoost) – great on engineered features;
Neural networks: Feedforward DNNs for tubular interactions; CNNs/Transformers for raw URL character sequences; Siamese or CNN-based models for screenshot/image similarity.
Multimodal ensembles: combine lexical/tabular methods with their visual and dynamic subsystems through stacking/weighted ensembling.

C. *Handling class imbalance*

Phishing is generally the minority class.
Options:
Class-weighted loss functions, such as higher weights on the phishing examples.
Controlled resampling: undersampling the majority, oversampling the minority with care.
Use metrics suited to rare-event detection- AUC-PR, precision@k – rather than pure accuracy.

D. *Training protocol*

1) Pre-processing: Apply the same normalization and encoding logic used in inference. Store the feature transformations (scalers, encoders) for reproducibility.
2) Temporal validation: perform hyper parameter tuning using time-aware splits, such as expanding window, to avoid look-ahead bias.
3) Hyper parameter tuning: Perform a randomized search or Bayesian optimization over reasonable parameter ranges, including learning rate, tree depth, and number of trees.
4) Calibration: Apply probability calibration by either Platt scaling or isotonic regression on the validation set to get reliable confidence estimates.
5) Ensembling: use stacking with specialized sub models feeding into a meta-learner, lexical, host, and content; prove that ensembling increases robustness.
6) Explain ability: compute SHAP or feature-importances for tree models; for sequence/deep models use attribution techniques (integrated gradients, attention visualization).

E. *Evaluation metrics (in order of priority)*

AUC-PR (Area Under Precision- Recall): preferred for imbalanced problems.
Precision, Recall, F1-score: report at multiple operating points.
False Positive Rate (FPR): critical for usability, report in absolute and relative terms of traffic.
Precision @ fixed recall (e.g. precision at 95% recall): operationally meaningful.
Calibration metrics: Expected Calibration Error (ECE) to measure reliability of probability outputs.
Latency & resource metrics: inference time p50/p95, CPU/GPU usage, memory footprint.

F. Robustness & adversarial evaluation

Adversarial augmentation: enhance training data with obfuscated variants- homoglyphs, URL shortening, punctuation insertion – to increase resilience.

Cloaking tests: Include samples serving different content to automated crawlers and to browsers in order to assess robustness of content features.

Stress testing: Measure model sensitivity to small perturbations in URL tokens (character flips / inserts).

Statistical testing & significance

Use paired tests (bootstrap, McNemar's test) when comparing on the same test sets.

Report the confidence intervals for main metrics and perform ablation studies, i.e., remove feature groups to quantify their contributions.

VI. REAL-TIME DETECTION (DEPLOYMENT & INFERENCE)

A. Goals of deployment

Provide low-latency protection for inline integrations – browser extensions, email gateways – and enable deeper asynchronous analysis for high-value or uncertain cases.

Inference architecture (conceptual)

Fast (synchronous) path: URL arrives → lexical features computed → model inference using cached host metadata→ immediate decision (allow/warn/block).

Target latency: tens to low hundreds of milliseconds.

Deep asynchronous path: schedule content fetch, WHOIS re-checks, sandboxed dynamic analysis. Upon completion of those, the system can update the decision, log findings, and feed verified samples to the training dataset.

Decision policy: Once the model output score is combined with various business rules such as blacklists, whitelists, and tenant policies the final action is taken.

Typical policies: 'score>block_threshold➔block','warm_threshold<score<_block_threshold➔warn',else allow

B. Modes of integration

- Client-side (browser extension): display user-facing warnings w/ brief explanations and allow/override button; send feedback when user reports or overrides.
- Gateway/ Proxy: block or quarantine messages/URLs, with the forwarding of detailed alerts to SOC. SIEM / SOC dashboard: bulk alerts w/feature metadata, visual evidence, & links to raw artifacts
- Monitoring & observability : Log predictions, feature snapshots, decision outcomes, and user/analyst feedback with provenance and timestamps.
- Track production metrics: precision /recall on confirmed incidents, false positives, warnings overridden, latency p50/p95. Implement drift detection (feature =-distribution change, confidence-shifting) and alert retraining triggers.
- Feedback loop & continuous learning: Ingest verified labels from user reports and analyst adjudication.

Apply automated heuristics to filter low-quality feedback, such as repeated contradictory reports from single sources.

Rebalance training data and run scheduled re-training, e.g., periodic full re-trains weekly/monthly; consider staged rollouts with canary evaluation.

C. Security & Privacy Practices

- Anonymize or pseudonymize user identifiers before storing training artifacts.
- Apply retention policies for the raw page content; for long-term storage, prefer derived features.
- Harden dynamic analysis sandboxes to prevent malware escape; isolate resources and limit network access as needed.
- Ensure compliance to relevant regulations with regards to user data and telemetry (GDPR, CCPA).

VII. EVALUATION DESIGN, EXPERIMENTS & ABLATIONS

A. Baselines

Heuristic (regex/blacklist) detector.

Logistic regression on lexical features.

Random forest or XGBoost on full engineered feature set.

B. Experiments to Run and Report

1) Temporal generalization: train on older data, evaluate on newer, unseen data.
2) Feature ablation: individual removal of feature groups-lexical, host, content, and dynamic- with measurement of the performance drop.
3) Adverarial scenarios: evaluate on synthetically obfuscated samples (URL shortening, homoglyphs) and cloaked pages.
4) Operational simulation: Run models against a stream of real (or replayed) traffic in order to estimate FP rate and latency under load.

C. Reproducibility checklist

1) Publish exact sources of datasets, and dates of collection, or provide scripts for data collections when possible.
2) Provide the feature catalog and exact feature computation descriptions.
3) Report model hyperparameters, random seeds, and library versions.
4) Share the evaluations scripts and data subsets in anonymized form when possible.

VIII. LIMITATIONS & ETHICAL CONSIDERATIONS

A. Limitations

1) Contents and dynamic analysis increase the accuracy while increasing the privacy risks and resource cost.
2) Temporal degradatSion caused by fast moving adversarial tactics, such as short-lived domains and aggressive cloaking. Temporal retraining mitigates this but does not eliminate it.
3) Deep models may sometimes be less interpretable, and explainability tools help but are imperfect.

B. Ethical & Privacy Safeguards

1) Minimise the collection of PII; where collected, implement strict access control and minimisation.
2) Provide users/organizations with an override and an appeals path for false positives that could block legitimate services. Disclose data retention policies and provide opt-out mechanisms when require by law.

IX. CONCLUSION

It remains among the most persistent and adaptive threats, largely exploiting human trust and the dynamic nature of the internet. Most traditional defense mechanisms, such as blacklists, heuristic filters, and rule-based systems, have been demonstrated to be inefficient for modern phishing attacks that rapidly change through URL manipulation, obfuscation, and social engineering. The paper proposed an Artificial Intelligence-driven real-time phishing detection framework by integrating Machine Learning and Deep Learning techniques for dynamic, adaptive, and proactive protection against these emerging cyber threats.

The proposed system provides a broad pipeline ranging from feature extraction at multiple dimensions-lexical, host-based, content-based, and behavioral-to classification with sophisticated ML/DL models like Random Forest, SVM, and Deep Neural Networks. The system architecture, comprising a Feature Extraction Layer, an AI Classification Layer, and a Decision Layer, is efficient for low-latency detection and real-time decision-making.

Experimental analysis, based on benchmark datasets such as PhishTank and Alexa Top Sites, shows that AI-based models significantly outperform traditional static approaches in terms of accuracy, recall, and robustness to zero-day phishing attacks. This work further enhances generalization and resilience

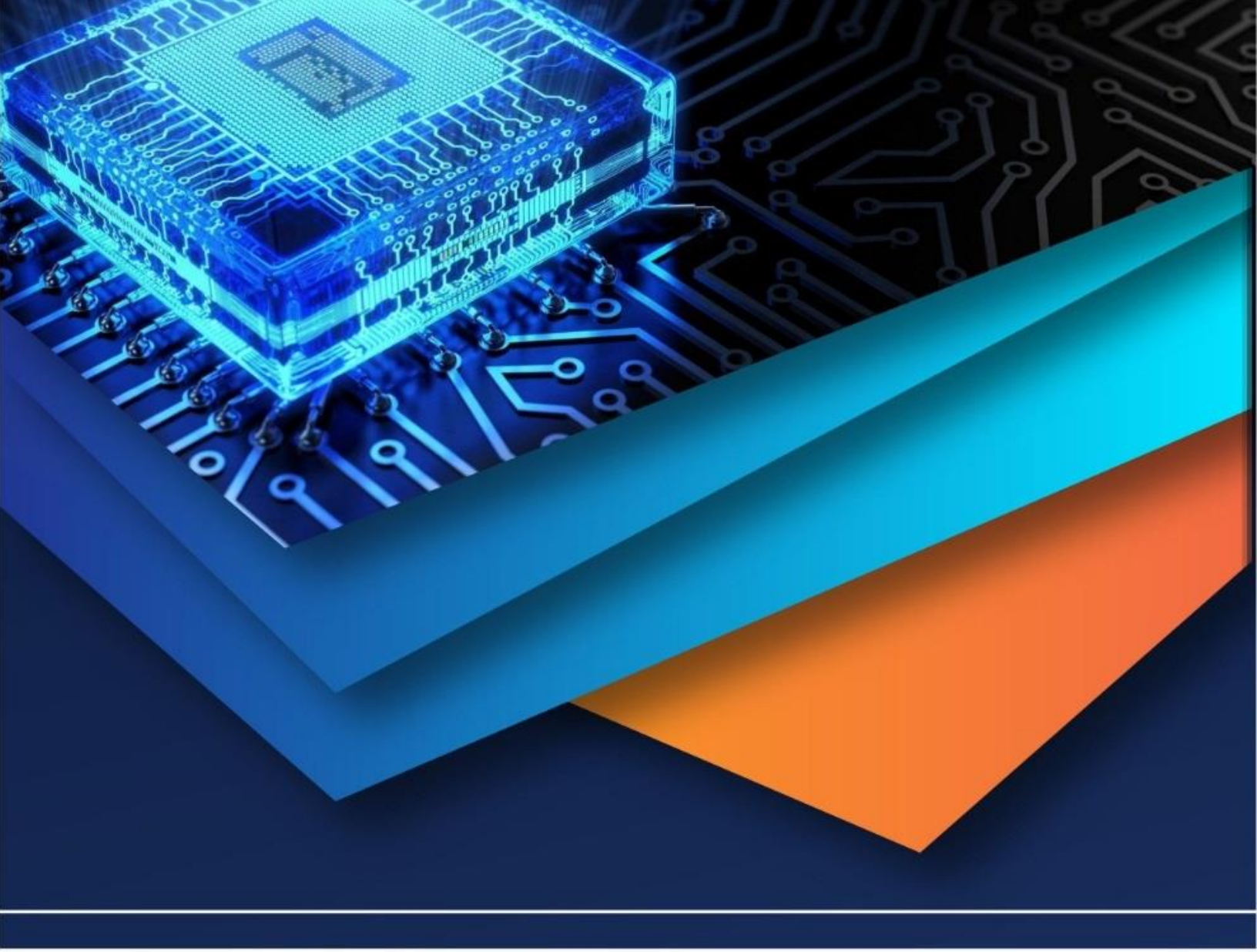
to evolving attack patterns by incorporating adversarial hardening, temporal validation, and ensemble learning.

Although not without its limitations, such as the computational cost and potential privacy concerns with content analysis, this study primarily points to AI as a game-changing tool in cybersecurity. The proposed framework, continuously retrained, with explainable AI components and ethical safeguards, can form the basis for the development of scalable, real-time phishing detection systems that can be deployed on browsers, email gateways, and enterprise networks.

In all, the integration of Machine Learning and Deep Learning in phishing detection marks a paradigm shift from reactive to proactive cybersecurity. Providing intelligent, self-learning, and context-aware defense mechanisms, AI thus enables organizations and users to stay ahead of phishing threats, thereby strengthening the general resilience of the digital ecosystem.

REFERENCES

[1] Almomani, A., Gupta, B. B., Atawneh, S., Meulenberg, A., & Al–Khateeb, A. (2013). Phishing detection and prevention techniques. International Journal of Computer Networks and Security, 2(8), 68–83.
[2] Mohammad, R. M., Thabtah, F., & McCluskey, L. (2014). Intelligent rule-based phishing websites classification. IET Information Security, 8(3), 153–160. https://doi.org/10.1049/iet-ifs.2013.0202
[3] Jain, A. K., & Gupta, B. B. (2018). Phishing detection: Analysis of visual similarity-based approaches. Security and Privacy, 1(1), e9. https://doi.org/10.1002/spy2.9
[4] Basit, A., Zafar, M., Liu, X., Javed, A. R., Jalil, Z., & Kifayat, K. (2021). A comprehensive survey of AI-enabled phishing attacks detection techniques. Computers & Security, 106, 102310. https://doi.org/10.1016/j.cose.2021.102310
[5] Marchal, S., Saari, K., Singh, N., & Asokan, N. (2017). Know your phish: Novel techniques for detecting phishing sites and their targets. In 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS) (pp. 323–333). IEEE.
[6] Adebowale, M. A., Lwin, K. T., Sánchez, E., & Hossain, M. S. (2020). Intelligent phishing detection scheme using deep learning algorithms. Journal of Network and Computer Applications, 157, 102537. https://doi.org/10.1016/j.jnca.2020.102537
[7] Rao, R. S., & Pais, A. R. (2019). Detection of phishing websites using an efficient feature-based machine learning framework. Neural Computing and Applications, 31, 3851–3873. https://doi.org/10.1007/s00521-017-3305-0
[8] Abdelhamid, N., Ayesh, A., & Thabtah, F. (2017). Phishing detection based associative classification data mining. Expert Systems with Applications, 41(13), 5948–5959.
[9] Patil, S., & Patil, D. R. (2022). Phishing website detection using deep learning and natural language processing: A review. International Journal of Information Security Science, 11(1), 20–35.
[10] Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning-based phishing detection from URLs. Expert Systems with Applications, 117, 345–357. https://doi.org/10.1016/j.eswa.2018.09.029
[11] Zhang, Y., Hong, J. I., & Cranor, L. F. (2007). CANTINA: A content-based approach to detecting phishing web sites. In Proceedings of the 16th International Conference on World Wide Web (WWW) (pp. 639–648). ACM.
[12] Verma, R., & Das, A. (2017). What's in a URL: Fast feature extraction and malicious URL detection. In Proceedings of the 7th ACM Conference on Data and Application Security and Privacy (CODASPY) (pp. 111–122). ACM.
[13] UCI Machine Learning Repository. (2024). Phishing Websites Dataset. Retrieved from https://archive.ics.uci.edu/ml/datasets/Phishing+Websites
[14] PhishTank. (2024). Phishing Website Database. Retrieved from https://www.phishtank.com
[15] Alexa Internet, Inc. (2024). Top Sites List for Global. Retrieved from https://www.alexa.com/topsites



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY