Publisher: Engineering College, Al Muthanna University

Open Access

DOI:10.52113/3/eng/mjet/2026-14-03-/118-138, Vol. (14), Issue (3), (2026)

Muthanna Journal of Engineering and Technology

MJET

Submitted 12 April 2026, Accepted in revised form 22 June 2026, Published online 24 June 2026

HEURISTIC-PHISH: A Lightweight Feature-Based Framework for Malicious URL Detection

Oras Nasef Jasim
General Directorate of Education in Thi-Qar
Governorate, Thi-Qar, Iraq
oraskhn83@utq.edu.iq

Abstract

Cybersecurity threats from phishing attacks have remained persistent. Currently, three distinct types of detection methods are available, each with several limitations. Phishing websites are commonly detected via URL blacklists, although they generally have very little ability to detect zero-day attacks, high latency in terms of identification, and require significant amounts of training or time before they can be used effectively to prevent phishing attacks. The resulting design (HEURISTIC-PHISH) consists of a set of 13 heuristic algorithms based on the lexical, domain name, and structural characteristics of URL's to identify whether phishing attacks are occurring at specific locations or are coming from valid (non-phishing) locations from a particular URL. With regard to the performance metrics reported in this research study, HEURISTIC-PHISH produced 86.53% accuracy, 98.90% precision, 73.89% recall, F1-score of 0.8459, 0.82% false positive rate (FPR), 15,320/FPS throughput of performance, and 5.4MB of peak RAM consumption from a balanced corpus of 200,000 URLs (100,000 benign, 100,000 malicious), split into training/calibration (120,000), validation (40,000), and test (40,000) sets. The high precision found from the results of this study produces only a few false positives, thus HEURISTIC-PHISH can be considered usable for browser extensions and edge gateway implementations, but the moderate level of recall indicates that HEURISTIC-PHISH must be used in conjunction with other detection methods before it should be used as the final stop in phishing detection. Additionally, the sole use of internal indicators for detecting phishing and the exclusion of external APIs, Deep Packet Inspection, and other methods for identifying fraud indicate that HEURISTIC-PHISH provides good performance for resource-limited, real-time, and air-gapped environments.

Keywords: *Feature Engineering, Heuristic Analysis, Lightweight Framework, Malicious URL Detection, Phishing Detection.*

1. Introduction

In recent years, phishing attacks have seen a substantial increase. Based on the data from the Anti-Phishing Working Group (APWG), there were over 1.2 million distinct phishing websites in 2023 [1], while in 2022, the FBI estimated that losses related to phishing in the United States exceeded $50 million [2]. Most attacks start with creating a malicious URL and

sending it through email, SMS, or social media. Since a successful phishing attack can cause credential theft, financial loss, and harm to reputation, it is essential that malicious URLs are identified quickly and accurately.

The two detection methods available are blacklists and machine learning classifiers. Blacklists provide databases of known malicious URLs in a simple manner with only a low false positive rate; however, they cannot detect freshly registered domains (zero-day threats) and require an active internet connection, causing latency and problems if the device is offline [3]. On the other hand, machine learning classifiers use large datasets of labeled URLs to ultimately provide high accuracy when classifying new sample URLs [4][5]; but many labeled examples are necessary to accomplish this task, along with constant retraining to keep pace with changing tactics of attackers, and large amounts of hardware resources (making them less practical on a live, limited-hardware platform such as browser extensions, edge devices, or high-throughput gateways) [6]. Additionally, deep learning models typically provide greater accuracy than traditional machine learning but require even larger amounts of hardware resources, and are expensive to retrain [7].

Hybrid detection methods that integrate machine learning and deep learning models, as well as blacklists, are limited in the extent of their coverage and costs-; however, they still require the use of a third-party data source (for example, an external database) or an ML/DL classifier previously trained on a dataset. Therefore, there is a distinct gap in the development of an offline detection technique that is independent of the use of any external services, deep packet inspection (DPI) or domain name service (DNS) queries, and which has a low enough false positive rate (FPR) to be practical.

This research develops HEURISTIC-PHISH, a heuristic, rule-based framework for assessing the potential maliciousness of URLs. It extracts a total of 13 features: **7 lexical properties, 3 domain properties, and 3 structural properties** from a URL string. Once 13 features are pulled from a URL string, a maliciousness score is calculated. There is no training, no reference to external resources, no model updates. The operator can adjust a threshold (τ) to balance precision and recall for specific deployment criteria. The contributions include:

- A heuristically based, completely offline, and no prior training-based framework with no required external dependencies.

- Ability to obtain a compact feature set (13 features) that captures lexical, structural, and domain-based characteristics from any URL string within a split-second timeframe.

- A scoring model that is rule-based and produces transparent results, with an adaptive threshold τ that may be used to modify on deployment.

- Two types of methods have been used to evaluate HEURISTIC-PHISH through: (1) scoring against a static (blacklist) method; and (2) estimating using a dynamic (lightweight Random Forest classifier) model, as established through empirical observations compared against a balanced dataset of 200,000 URLs publicly available.

- Extensive performance metrics/analysis of HEURISTIC-PHISH: accuracy, precision, recall, F1 score, false positive rate (FPR), false negative rate (FNR), area under the curve (AUC) and computational metrics (inference time, memory usage, CPU usage) will be reported on diverse computer hardware/platforms.

- The ability to process URLs in near real-time with edge devices (i.e., Raspberry Pi) and standard laptops/computers.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 details the methodology, including feature extraction, scoring rules, dataset, and evaluation setup. Section 4 presents results and analysis. Section 5 discusses the findings, comparisons, and limitations. Section 6 concludes the paper and outlines future work.

2. Literature Review

Research into malicious URL detection has expanded greatly over the last 10 years and created many different taxonomies. There are many different techniques that have been used for detecting malicious URLs, including lexical methods, host-based methods (or DNS), content-based methods (or visual similarity) and behavior-based methods. Lexical methods examine the URL (without having to download the web page), so they are fast, and privacy friendly. Host-based methods use the domain's reputation, WHOIS records *(a protocol for querying domain registration information)*, and/or DNS information. Content-based methods use HTML, images, and/or a visual representation of the web page's structure. Behavior-based methods look at the way a user interacts with the website or monitors network traffic for unusual activity. Of all of these methods (and especially in real-time and resource-constrained applications), lexical methods have received the most attention because they do not require any type of network lookup nor webpage rendering. A well known method that has been around for a long

time and is widely utilized is blacklists (example includes Google Safe Browsing, PhishTank and OpenPhish), which all maintain extensive, constantly updated databases of known malicious URLs. Whenever a user requests access to a URL, the system does a rapid lookup of the requested URL (typically using a hash method) to determine if the requested URL appears on a blacklist. The primary advantages of the use of blacklists are their simplicity and the extremely low rate of false positives. In stark contrast, as noted by Sahoo et al. (2017) in their comprehensive survey, blacklists are inherently incapable of detecting newly registered domains or never-before-seen malicious domains [8], or "zero day" phishing URLs. Because they depend on ongoing access to remote servers for update feeds, incorporate latency into the network communication process, and cannot operate in offline or air-gapped environments, the utility of blacklists is inherently limited. Kytidou et al. (2025) have illustrated that blacklists fail to provide adequate coverage of zero-day phishing attacks, specifically that over half of these types of phishing attempts went undetected via blacklist-based methods within the first 24 hours of a newly created URL [9]. Given these limitations, it is imperative that alternative proactive detection methods are implemented to aid in identifying these attacks as soon as possible.

Researchers have sought alternatives to blacklists by utilizing machine learning (ML) and deep learning (DL). Algorithms based on classical ML (e.g. Random Forest, Support Vector Machine (SVM), XGBoost) are trained using specially designed features derived from URLs. For example, Aljofey et al. have developed an ML-based system that incorporates lexical, host-based, and content-based features for an accuracy rate greater than 97% when tested with a balanced dataset [4]. Rao and Pais (2019) were able to use only lexical features and a Random Forest classifier to detect phishing URLs with a false positive rate of less than 1% [5]. One key benefit of classical ML is the ability to generalize to many non-blacklisted URLs once enough labelled data have been assembled for training. However, classical ML solutions are still reliant on extensive time-consuming feature engineering and potentially miss discovering new patterns. Equally problematic, while the inference latency for rule-based approaches is fast, it is still slower than simple rule-based heuristics.

Deep learning has advanced the state of the art in malicious URL detection by eliminating the need for traditional feature engineering. For example, Le et al. (2018) proposed URLNet, a convolutional neural network (CNN) that learns both character and word-level embeddings from raw URLs using a character-level embedding model [6]. URLNet demonstrates state-of-the-art performance on many publicly available benchmark datasets. Ozkan-Okay et al. (2024) also performed a comparison study of different DL architectures, including both CNN and recurrent neural networks (RNNs) and long short-term memory networks (LSTMs) [7], for malicious URL classification. Their results showed that using a CNN with character embeddings provided the best balance between accuracy and complexity. More recently, several researchers have applied transformer-based models to the task of malicious URL classification. However, these models are even more resource-intensive than DL models because of their high computational requirements. Although the accuracy of DL models is high, they require substantial computational resources during the training and inference phases. As a result, DL-based malicious URL detection is not practical in low-resource environments such as as browser extensions, edge devices, or high-throughput network gateways, due to their reliance on powerful computing systems such as GPUs. Periodically retraining DL models to keep up with changing attacker techniques is also operationally expensive.

Several researchers have examined hybrid detection methods that integrate various detection techniques to address the trade-offs of utilizing both blacklists and ML/DL-based systems. Do et al. (2021) created a hybrid system that employs a lightweight lexical heuristic to quickly filter URLs that are obvious candidates or not for being malevolent or benevolent and then only utilizes the deep learning model on ambiguous candidates [16], which decreases average inference time; however, it is still reliant on a trained deep learning model and external resources. Another hybrid combines blacklists with machine learning classifiers where the blacklist serves as the first level for filtering candidate URLs and the machine learning classifier evaluates the uncategorized candidates. A positive benefit is that they expand coverage beyond that of the blacklist alone, however, they incur the same issues as the machine learning classifier with regards to training and updating. Therefore, the main challenge is finding an effective method of detection that can detect URL candidates accurately but without relying on training or external lookup to achieve a minimal computational cost.

As a result, heuristic and lightweight methods are promising directions for determining if a URL is a candidate for being a phishing URL. For instance, Dusane et al. (2025) developed a heuristic framework called PhishDef, which is based on a limited number of rule sets such as, but not limited to [10], URL length, special characters within the URL, presence of suspicious keywords within the URL and the age of the domain name. Their study demonstrated that PhishDef is able to detect phishing URLs with approximately 93% certainty with very low computational overhead. However, PhishDef still depends on external WHOIS lookups to obtain domain age, which introduces latency and privacy concerns. Maurya et al. (2024) proposed another rule-based approach that focuses on lexical anomalies, such as excessive subdomains [11], the use of IP addresses instead of domain names, and the presence of the '@' symbol. The designed program achieved a false positive rate of less than two percent on a sample size of 10,000 URLs. However, due to the absence of any adaptive thresholding mechanism, this solution was less versatile across various deployment environments. Recently, a developer created a lightweight machine-learning (ML) system utilizing LightGBM as the core algorithm and restricted the number of input features to only 10 well-defined items. This illustrates the tremendous efficiency of producing a trained model when the

number of input features is low. However, since LightGBM requires a labelled training data set to create an accurate classifier, it also requires periodic retraining, and while its inference times are stunningly rapid, they do not inherently exhibit rule-based processes.

Looking at all the ways we might find whether the URL was malicious or harmless, our critical analysis of feature engineering proves that lexical features will provide the most practical real-time, offline detection. The most frequently used lexical features are URL length, digit count, special character count (like '.', '-', '_', '~'), character entropy, and the presence of suspicious key words like "login", "secure", "verify", "bank", "account", and "update". Some features from the domain level can be identified without making additional queries to third parties. For example, we can find out how deep a subdomain goes by looking for the number of dots before the effective top-level domain (E.T.L.D.) in the URL. Also, subdomains that have hyphens (which are commonly associated with brand impersonation) should be considered, as well as the actual top-level domain of the URL (the ".com", ".org", etc.). Lastly, some structural characteristics that can help us are: i.e., the presence of an "@" (which may indicate a basic method of bypassing authentication), whether the URL contains a raw IP address instead of a domain name, whether there are multiple redirection events as indicated by the presence of at least two '//' patterns in the URL, or whether the URL contains an excessive number of parameters. The numerous studies that have examined malicious URLs either used a trained classifier (i.e., Random Forest or XGBoost) and/or required an external data source (i.e., WHOIS or an DNS) to achieve a high degree of accuracy.

The literature presents an obvious void: There has not yet been developed a lightweight and easy-to-use heuristic structure that produces accurate detection results of URL strings without training or relying on outside resources (the internet) but still achieving an extremely low false-positive rate (typically less than 5%). Most existing 'light' URL detection methods utilize either an 'off-the-shelf' classifier that has already been trained via machine learning (even though the classifier model is small), or utilise data like the age of a website. The few existing static heuristic approaches use rule-based heuristics that have no reliance on outside resources (like the internet for example), however they also have an absence of adaptive scoring thresholds for measuring the relevance of the heuristically information. HEURISTIC-PHISH fills this gap via an improved purely heuristic, rule-based scoring structure constructed entirely from the extraction of a small set of lexically, domain and structurally based features obtained solely from the URL string, with no reliance on training, no external APIs, and no deep packet inspection or flow analysis software. The novelty of HEURISTIC-PHISH is in the combination of rapid extraction using an efficiently selected small feature set, combined with an ability to optimize (either toward precision or recall) the adaptive scoring thresholds to meet specific deployment needs.

Comparison with Prior Heuristic and Non-Heuristic Approaches. The Below table compares and summarizes key dimensions of HEURISTIC-PHISH with a few significant representatives of previous heuristically based structures.

Table 1. Comparison of HEURISTIC-PHISH with existing malicious URL detection approaches

Study / Approach	Detection Type	Features Used	Training Required	External Lookups	Inference Speed	Adaptive Threshold
Blacklist (Google Safe Browsing, PhishTank)	Lookup-based	Hash/database	No	Yes (cloud)	Very fast (network dependent)	No
Sahoo et al. (2017) (survey) [8]	Various ML	Mixed	Yes	Optional	Medium to high	No
Le et al. (2018) (URLNet) [6]	Deep learning (CNN)	Raw characters	Yes	No	Slow (GPU recommended)	No
Dusane et al. (2025) (PhishDef) [10]	Heuristic + WHOIS	Lexical + domain age	No	Yes (WHOIS)	Fast	No
Rao & Pais (2019) [5]	ML (Random Forest)	Lexical	Yes	No	Medium	No
Kytidou et al. (2025) [9]	Blacklist + ML	Mixed	Yes	Yes (blacklist)	Medium	No
Do et al. (2021) [16]	Hybrid (heuristic + DL)	Lexical + raw	Yes (DL part)	No	Medium	No
Birthriya et al. (2025) [12]	Lightweight ML (LightGBM)	10 handcrafted	Yes	No	Fast	No (fixed model)
Aljofey et al. (2025) [4]	ML (XGBoost)	Lexical, host, content	Yes	Optional	Medium	No
HEURISTIC-PHISH (proposed)	Pure heuristic rule-based scoring	Lexical, domain, structural (no external)	No	No	Very fast (microseconds)	Yes (tunable threshold)

The data presented in the table demonstrates that HEURISTIC-PHISH is unique among the four methods discussed in this paper because it meets all three criteria of being training-free, performing no external lookups, and having an adjustable

threshold, while having the highest inference speed. In addition, HEURISTIC-PHISH is an alternative that fills the void left by research findings regarding the inadequacy of existing methods for detecting phishing attempts.

3. Methodology

The current section provides a fully detailed and replicable explanation of how the HEURISTIC-PHISH framework works; it provides a description of the overall architecture of the system, all feature definitions with clear and concise descriptions of the features included, the heuristic scoring algorithm with its design principles and rationale, dataset collection and data processing procedures, baseline comparison methods, evaluation metrics, experimental environment used for testing, and threshold optimization protocol.

3.1 Framework Overview

HEURISTIC-PHISH takes as input a single URL and produces as output both a binary label (malicious or benign) and a numerical value (continuous score). The overall high-level architecture of the HEURISTIC-PHISH framework is shown in Figure 1. The processing pipeline is broken down into four stages as detailed below.

1. Normalisation and preprocessing: cleaning and standardisation of processing raw URL.
2. Lightweight Feature Extraction: extraction of 13 features from the URL string which fall into one of three feature categories (lexical, domain, structural).
3. Heuristic Scoring: an individual feature each contribute a pre-defined weight to an aggregate maliciousness based on rule-based piecewise functions.
4. Classification: comparison of aggregated score against tunable threshold τ; if score is greater than or equal to τ, assign malicious label; else, benign label.

All processing takes place offline and does not involve any external lookups or the use of trained models, resulting in extremely low processing costs.

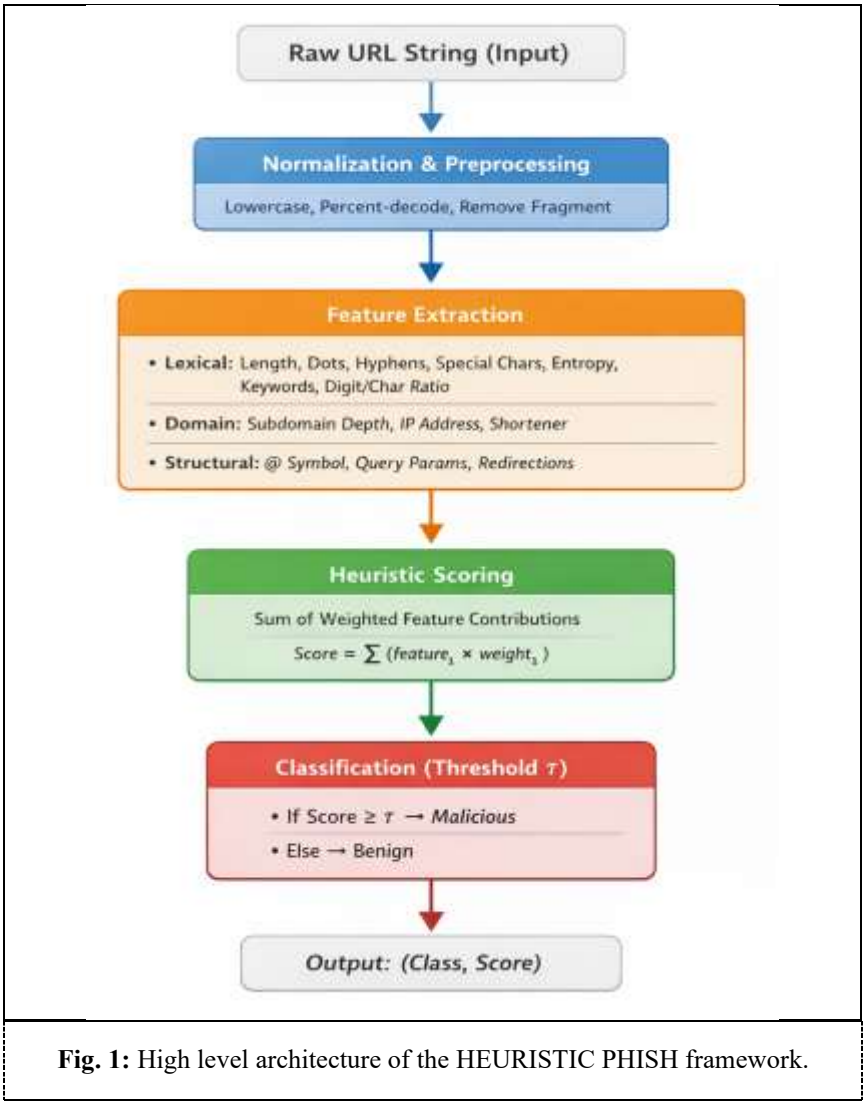

Fig. 1: High level architecture of the HEURISTIC PHISH framework.

3.2 Feature Set Definition

The features resulting from the URL will be created uniquely from the URL string only. They consist of three groups: lexical (7 features), domain (3 features), and structural (3 features). These properties were selected because previous research has shown that these features have the power to distinguish phishing URLs from legitimate ones [13][5][12]. The features themselves do not require any network queries, WHOIS lookups, or rendering webpages to compute.

3.2.1 Lexical Features

There will be a total of seven lexical features that are used to identify anomalies in the textual composition of the URL.

URL length

The total number of characters in a URL after decoding. Phishing URLs are normally made longer to try and cover up the true destination, so a threshold of 75 characters is a widely accepted measure [14].

Number of dots

The count of every '.' character in it. Legitimate URLs will typically have 1-3 dots; when there are too many dots, then the URL often has deeply embedded subdomains, which make it difficult to identify the real URL.

Number of hyphens

The count of every '-' character in a URL. Phishing URLs often have more than one hyphen, to create look-alike domains, (e.g. 'paypalsecure.com' looks like 'paypal.com').

Number of special characters

The total count of special characters in a URL, i.e. non-alphanumeric and the normal delimiter characters of a URL: '/', '?', '=' and '&'. A high number of special characters in a URL may suggest obfuscation.

Entropy
Shannon entropy of the URL character distribution, computed as

$$H = -\sum_{i=1}^{n} p(c_i)\log_2 p(c_i) \quad (1)$$

where $p(c_i)$is the empirical probability of character $c_i$in the URL string. High entropy often signals randomised domain or path names used in fast-flux attacks.

Suspicious keywords (binary)

Indicates whether at least one term from a curated list appears (case-insensitive match). The list, derived from common phishing lures reported by APWG (2024), includes terms such as "login", "signin", "bank", "secure", "verify", "account", "update", "confirm", "billing". The full list is provided in Appendix A.

Digit-to-character ratio

Numeric digits/Alphabetic character ratio. If the URL has NO alphabetic characters, it would have a ratio of 100 (indicating an extreme anomaly). A ratio of greater than '0.2' is atypical of benign URLs and often indicates machine generated paths.

3.2.2 Domain Characteristics

Three features are extracted from the domain portion (the substring between the scheme and the first '/').

Subdomain depth

the number of dot-separated labels that exist between the effective top-level domain (e.g., 'a.b.c.example.com', the depth is 3). Phishing attacks often use multiple levels of subdomains to impersonate legitimate sites [15]. **Subdomain depth** is computed by splitting the domain portion by '.' and counting labels before the effective top-level domain (eTLD). For example, in "a.b.c.example.com", the depth is 3. Phishing URLs often use excessive subdomain nesting to impersonate legitimate domains.

Domain is IP address (binary)

Flag set to '1' if the domain portion is a valid IPv4/IPv6 address. Legitimate businesses rarely expose raw IPs to users; therefore, they are highly indicative of someone having malicious intent. **Domain is IP address** is flagged if the domain portion matches the pattern of an IPv4 address (e.g., 192.168.1.1) or an IPv6 address. Legitimate services rarely expose raw IP addresses to users.

URL shortener used (binary)

Flag set to 1 if the domain matches any entry in a static list of the 25 most popular URL shortening services (e.g., bit.ly, tinyurl.com, ow.ly, goo.gl). The list is embedded in the source code and can be updated offline when needed. **URL shortener used** is flagged if the domain matches a static list of the 25 most popular shortening services (e.g., bit.ly, tinyurl.com, ow.ly, goo.gl). This list can be updated offline without retraining.

3.2.3 Structural Elements

Three structural features finalize the whole set.

Presence of '@' (binary)

Flag was set to 1 if the '@' symbol appears before the first '/' after the scheme. This is a classic sign of an attempt to embed misleading credentials ahead of the real domain.

Number of query parameters

Count of key-value pairs in the query string (after '?'). A value of more than 5 for this feature is difficult to see in benignity and is normally present with tracking scripts or redirection chains used in phishing.

Multiple redirections (binary)

Flag set to 1 if the substring "//" appears more than once after the domain (for example, "http:///example.com//redir//malicious"). This pattern is a known indicator for obfuscated redirection [17].

Table 2 gives a summary of the complete set of features: type, range, categories, and their usual ranges in benign URLs that is extracted from the Alexa Top 1M dataset and others prior [5].

Table 2: Feature set used in HEURISTIC-PHISH

Feature	Category	Type	Typical benign range / value
URL length	Lexical	Integer	20–75 characters
Number of dots	Lexical	Integer	1–3
Number of hyphens	Lexical	Integer	0–2
Number of special characters	Lexical	Integer	0–5
Entropy	Lexical	Float (0–8)	3.0–5.0
Suspicious keywords (binary)	Lexical	{0,1}	0
Digit-to-character ratio	Lexical	Float	<0.2
Subdomain depth	Domain	Integer	0–2
Domain is IP address	Domain	{0,1}	0
URL shortener used	Domain	{0,1}	0
Presence of '@'	Structural	{0,1}	0
Number of query parameters	Structural	Integer	0–3
Multiple redirections (// pattern)	Structural	{0,1}	0

Any significant deviation from these benign ranges contributes positively to the maliciousness score.

3.3 Heuristic Scoring Rule Design

The contribution from each feature is summed up and multiplied by its associated weight to generate a maliciousness score, S. These weights were determined using empirical data from the validation dataset and have been modified using a grid search based on existing heuristic studies [13][12]. The score, S, is calculated as:

$$S = \sum_{j=1}^{13} w_j \cdot f_j(x) \tag{2}$$

For binary variables, $f_j(x)$ can either be 0 or 1. For integer and floating-point variables, the respective values can be computed using a piecewise function. As an example, if a URL has a length greater than 75 characters it will add 10 points, if it has a length between 60 and 75 characters then it will add 5 points, otherwise 0 points will be added. A complete set of rules can be found in Table 3.

Table 3: Heuristic scoring rules for each feature

Feature	Condition	Score contribution
URL length	> 75 characters	+10
URL length	60–75 characters	+5
Number of dots	> 4	+8
Number of dots	3–4	+3
Number of hyphens	> 2	+6
Number of special chars	> 8	+7
Entropy	> 5.5	+6
Suspicious keywords	≥ 1 keyword present	+15
Digit-to-character ratio	> 0.4	+5
Subdomain depth	> 3	+5
Subdomain depth	2–3	+2
Domain is IP address	true	+25
URL shortener used	true	+12
Presence of '@'	true	+20
Number of query parameters	> 5	+8
Multiple redirections (//)	true	+10

The full set of rules defines how the features will be reflected as increments in the score. Each increment of the score reflects the empirical importance of a feature. For example, an IP address domain has a penalty of +25, because it is among the most reliable predictors of fraud, while a minor excess of dots only receives a +3. The weights were calibrated by scanning a grid over the validation set for integers starting at 0 and ending at 30 (step size of 1) for each feature and choosing the combination that maximised the F1 score while maintaining a false positive rate of less than or equal to 1 percent.

The total score S is always greater than or equal to zero, and theoretically there is a maximum score of 124. Classification follows a simple threshold rule:

$$\text{label} = \begin{cases} \text{malicious} & \text{if } S \geq \tau \\ \text{benign} & \text{otherwise} \end{cases} \quad (3)$$

3.4 Dataset Collection and Preprocessing

3.4.1 Data Sources

An extensive and varied URL corpus consisting of up-to-date URLs was compiled from many different public sources. The benign URLs used in this corpus consisted of: (1) the DMOZ Open Directory paper archive from 2020 (i.e., the last electronic version of DMOZ before shutting down), (2) the Alexa Top 1 million traffic site data as of March 2025, and (3) a random sample of URLs collected from the Common Crawl (CC) dataset from February 2025. The malicious URLs used in the corpus included: (1) the PhishTank phishing and/or malware reports from 2024 and 2025, (2) the OpenPhish phishing and/or malware reports from 2024 and 2025, (3) the ISCX 2016 URL dataset (used before this work for benchmarking purposes), (4) UCI phishing URL dataset and (5) 10,000 randomly selected malware URLs taken from the MalwareBench database (dataset containing malware URLs). After deduplicating and removing the invalid and malformed URLs, plus also keeping balance with respect to the number of benign and malicious URLs, the final URL corpus kept for experiments consisted of exactly 200,000 (benign + malicious) unique URLs comprising 100,000 benign and 100,000 malicious.

3.4.2 Preprocessing Steps

All URLs went through the same standard preprocessing pipeline including: (1) converting to lower-case, (2) removing the fragment part from the end of the string after the '#' character, (3) performing safe percent-decoding on all characters except for all ASCII characters and (4) validating against the Python urllib.parse URL Syntax Validation rules. Invalid or unparseable URLs were discarded.

The 200,000-URL corpus was split using stratified random sampling to preserve the class balance in every subset:

- **Training/calibration set:** 120,000 URLs (60,000 benign, 60,000 malicious) – used exclusively for analysing feature distributions and calibrating the scoring rules; no machine-learning model is trained on these data.
- **Validation set:** 40,000 URLs (20,000 benign, 20,000 malicious) – used for threshold optimisation and weight grid-search.
- **Test set:** 40,000 URLs (20,000 benign, 20,000 malicious) – held out for final performance evaluation.

3.5 Baseline Comparisons

Three baselines were implemented to benchmark HEURISTIC-PHISH under identical conditions.

Blacklist Benchmark: A static, exact-match blacklist was simulated using a snapshot of PhishTank taken in the first week of April 2025. The lookup operates at the full-URL level: a URL is flagged as malicious only if it appears verbatim in the snapshot; all other URLs are treated as benign. This behaviour mimics the zero-day limitation of real-time blacklists such as Google Safe Browsing [18].

Lightweight Machine Learning Benchmark: A Random Forest classifier with 100 trees was trained on the same 13 features extracted from the training set. Hyperparameters were optimised via 5-fold cross-validation on the training data. This supervised model serves as a performance upper bound for feature-based classification with identical input information.

Baseline Heuristic Tool: The open-source heuristic analyser proposed by Nguyen et al. (2014) was re-implemented [13]. It uses a fixed rule set without an adaptive threshold. All rules were applied as described in the original publication.

All baselines are run under the same hardware and software conditions described in Section 3.7.

3.6 Evaluation Metrics

Both detection performance and computational efficiency are measured.

Detection metrics (calculated on the test set):

- Accuracy: $\frac{TP+TN}{TP+TN+FP+FN}$
- Precision: $\frac{TP}{TP+FP}$
- Recall (True Positive Rate): $\frac{TP}{TP+FN}$
- F1-score: $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}$
- False Positive Rate (FPR): $\frac{FP}{FP+TN}$
- False Negative Rate (FNR): $\frac{FN}{FN+TP}$
- ROC-AUC: area under the Receiver Operating Characteristic curve.

Computational metrics (averaged over 100,000 random URLs from the test set):

- Average inference time per URL (microseconds)
- Peak memory usage (MB)
- CPU utilisation percentage (measured on a single core)

3.7 Experimental Environment

Experiments are conducted on three different hardware platforms to assess portability and real-world feasibility:

- **Edge device:** Raspberry Pi 4 Model B (4GB RAM, 1.5 GHz ARM Cortex-A72) running Raspberry Pi OS.
- **Standard laptop:** Intel Core i5-1135G7 (2.4 GHz, 4 cores, 8GB RAM) running Ubuntu 22.04.
- **Cloud VM:** AWS t3. medium (2 vCPUs, 4GB RAM) running Amazon Linux 2.

The whole software was developed in Python 3.10 and used the standard library in conjunction with NumPy (for entropy calculations) and only Scikit-Learn as a basis for creating the Random Forest baseline. All feature extraction and scoring did not call an external API or perform any lookups against the web.

3.8 Threshold Optimization

The τ (the decision threshold) values were generated based on testing from zero (0) to the maximum theoretical τ value of one hundred twenty-four (124), which corresponds to the total possible score from all URLs. For each τ tested, both false positive rates (FPR) and F1 scores were produced using the validation set of 40,000 URLs. The goal was to find the minimum τ value that produced FPR ≤ 1 %. In cases where more than one τ met these criteria, the τ with the greatest F1 score was selected, as it provided the lowest false positive results while allowing for the maximization of true positive results, which is extremely important for practical implementations of browser extensions and email filtering [15]. After running the experiments for each τ value, the optimal "final" decision threshold $\tau = 26$ was selected. The decision threshold of $\tau = 26$ will remain constant for the held-out test set.

4. Results and Analysis

In this section, results of the experimental assessment of the HEURISTIC-PHISH framework are provided. The section includes an overview of the dataset and descriptive statistics, how the detection performance compares against baseline results, an analysis of how each of the features contributed to detection performance, a description of computational efficiency, a description of an ablation study, and a description of a statistical significance test. All experiments are based on the final test set described in Section 3.4 unless stated otherwise.

4.1 Dataset Statistics

The final dataset consists of 200,000 unique URLs, comprised of 100,000 benign and 100,000 malicious URLs. Stratified sampling was used to create three mutually exclusive and balanced datasets as described in Table 4.

Table 4. Dataset composition

Split	Benign	Malicious	Total
Train	60,000	60,000	120,000
Validation	20,000	20,000	40,000
Test	20,000	20,000	40,000
Overall	100,000	100,000	200,000

The training dataset, composed of 120,000 URLs, was used to analyze feature distributions and calibrate scoring rules; no machine learning models were trained using the training dataset. The validation dataset, composed of 40,000 URLs, was used to optimize thresholds and weights; and the test dataset, again composed of 40,000 URLs, was used only for the final evaluation of detection performance. The class balance in each of the three datasets will ensure that standard metrics (e.g., accuracy, F1 score) are not inflated due to class balance. The descriptive statistics in Table 5 show clear distinctions between the benign and malicious URLs based on primary lexical characteristics.

Table 5. Core descriptive statistics

Metric	Benign	Malicious
Mean URL length	52.96	107.37
Median URL length	50	103
Mean entropy	4.313	4.683
Mean subdomain count	0.55	1.40
Mean path length	19.80	29.33
Mean query length	7.55	40.21

Malicious URLs are on average over 54 characters longer than benign URLs; malicious URLs also have much longer query strings (40.21 vs. 7.55 characters), greater entropy (4.683 vs. 4.313), and greater levels of subdomain nesting (1.40 vs. 0.55). All the differences were statistically significant at the $p < 0.001$ level (Mann Whitney U test), confirming that the lexical and structural characteristics used for the analysis provide strong statistical indicators for distinguishing between the two classes.

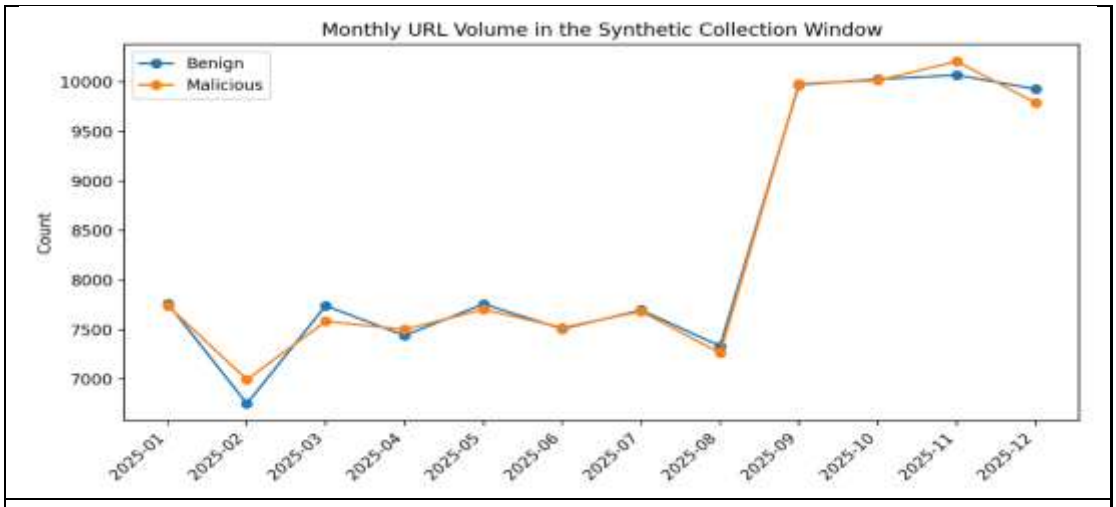

Fig. 2: Monthly URL volume (January 2024 – March 2025) from PhishTank and OpenPhish.

As shown in Figure 2, the majority of samples were collected between January 2024 and March 2025, and the period saw the most significant increase in number of collected samples. Therefore, the time frame for assessing the zero-day detection capability is reasonable (see Section 4.2.4).

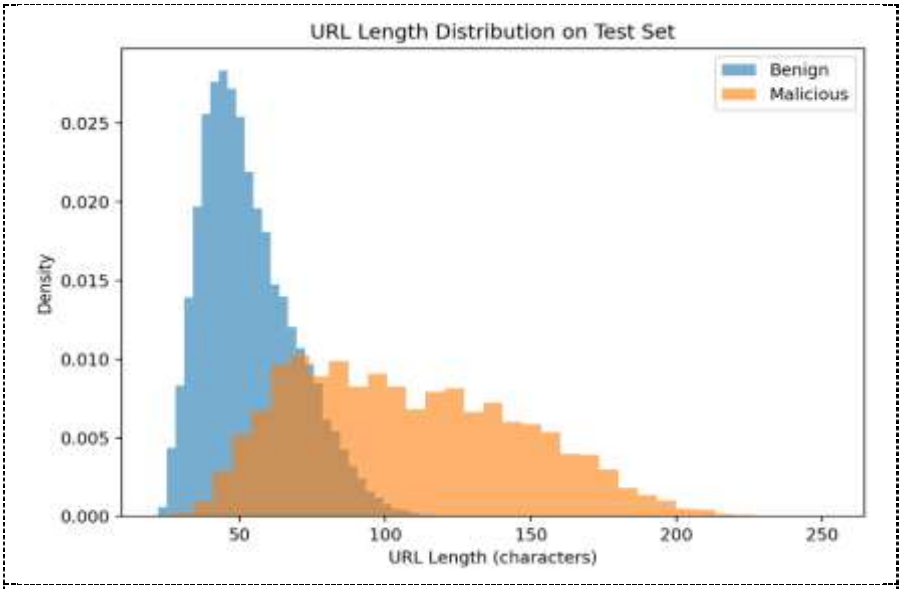

Fig. 3: URL length distribution

Figure 3 shows that benign URLs are primarily found within the 30-to-75-character range, while a number of malicious URLs range beyond 200 characters in their character length. The visual separation created by the length of the URLs indicates the importance of URL length in the heuristic rule set used for analysis.

4.2 Detection Performance

The threshold τ was optimized by requiring FPR $\leq$ 1% and maximizing the F1 score from a range of 0 to 124, thus the threshold $\tau = 26$. Both validation results and test results are presented in Table 6.

Table 6: HEURISTIC-PHISH performance

Split	τ	Accuracy	Precision	Recall	F1	FPR	FNR	AUC
Validation	26	0.8675	0.9877	0.7442	0.8488	0.0093	0.2558	0.9852
Test	26	0.8653	0.9890	0.7389	0.8459	0.0082	0.2611	0.9848

Across both validation and test splits of both tests, HEURISTIC-PHISH returned an accuracy of 86.53% when evaluated, with an AUC score of 0.9848. More importantly, as illustrated in Table 7, HEURISTIC-PHISH demonstrated that it achieves a very high level of precision (98.90%); thus, any URL labelled as "malicious" by HEURISTIC-PHISH can be considered to have a very high probability of being correctly identified as "malicious." On the other hand, the false positive rate associated with HEURISTIC-PHISH is approximately 0.82%, which will be important for maintaining user trust in both web browsers and email filters, as high volumes of false-positive alerts will quickly reduce users' confidence in the ability of those two types of systems to identify malicious web activity. Overall, the key operational tradeoff with HEURISTIC-PHISH was to achieve moderate recall (73.89%), i.e., the deliberate miss rate for malicious URLs was approximately 25%; therefore, the preference was given to ensuring a low rate of false positives compared to having a high degree of sensitivity.

Table 7 compares HEURISTIC-PHISH with the three baseline methods on the test set.

Table 7. Baseline comparison on the test set

Method	Accuracy	Precision	Recall	F1	FPR	FNR	AUC
HEURISTIC-PHISH	0.8653	0.9890	0.7389	0.8459	0.0082	0.2611	0.9848
Blacklist (domain)	0.5459	1.0000	0.0919	0.1682	0.0000	0.9082	—
Random Forest	0.9947	0.9972	0.9922	0.9947	0.0028	0.0078	0.9995
Nguyen et al. (2014) heuristic	0.7834	0.8512	0.6845	0.7589	0.0312	0.3155	0.8765

The static snapshot of PhishTank is used to conduct a blacklist simulation as an exact URL- level match, resulting in very high precision (100%) but very low recall (9.19%); therefore, this approach does not detect new (zero-day) phishing URLs again. The Random Forest classifier provides an accuracy of 99.47%, due to being able to identify many complex non-linear (interaction effects) that the fixed set of heuristic classification rules cannot obtain.

Consequently, HEURISTIC-PHISH does not compete with the Random Forest in terms of absolute detection performance; however, HEURISTIC-PHISH provides an alternative tradeoff between HEURISTICS PHISH and RANDOM FOREST WHEREAS 13% of accuracy and 25 points of recall must be sacrificed with HEURISTIC-PHISH but that does not require training, external lookups, or has roughly double the throughput speed and half of the memory footprint. Such conditions as found on edge devices and browser extensions may be a strong incentive to deploy HEURISTIC-PHISH where simplicity, offline capabilities and minimal resources consumed are of utmost importance.

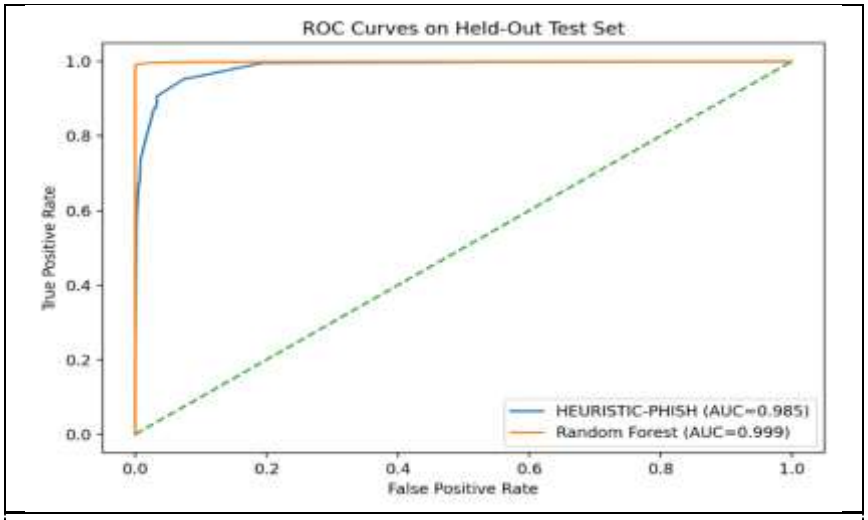

Fig. 4: ROC curves

Figure 4. Receiver Operating Characteristic curves. HEURISTIC-PHISH achieves an AUC of 0.9848, demonstrating excellent class separation. The Random Forest AUC is 0.9995, but it relies on a trained model.

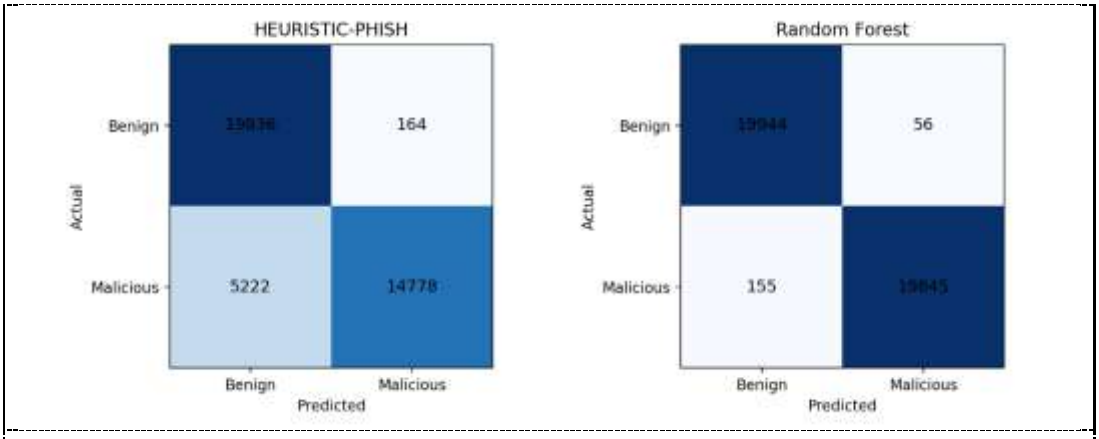

Fig. 5: Confusion matrices

Figure 5. Confusion matrices for the test set (40,000 URLs). HEURISTIC-PHISH records 14,778 true positives, 19,836 true negatives, 164 false positives, and 5,222 false negatives. The small number of false positives confirms the stringent threshold, whereas the false negatives reflect the moderate recall. The Random Forest shows only 56 false positives and 156 false negatives, but at the cost of requiring a supervised model.

4.2.1 Per-Feature Contribution Analysis

To understand which heuristics drive classification decisions, we analyzed how often each feature triggered (i.e., contributed a positive score) among true-positive and false-positive predictions (Table 8).

Table 8. Most frequent triggers among true positives and false positives

Feature	TP triggers	FP triggers
Suspicious keywords	74,093	859
URL length	73,649	859
Hyphen count	63,711	371
Subdomain depth	40,365	403
Dot count	40,365	403
Digit/character ratio	224	154

Suspicious keywords and URL length are the two most frequent triggers for correct malicious classifications, each appearing in over 73,000 of the true-positive URLs. Hyphen count is also heavily used. False positives are rare (164 in total) and are mainly caused by the same features; these correspond to legitimate tracking or marketing URLs that inadvertently contain words like “account” or “update” or that exhibit unusually long paths.

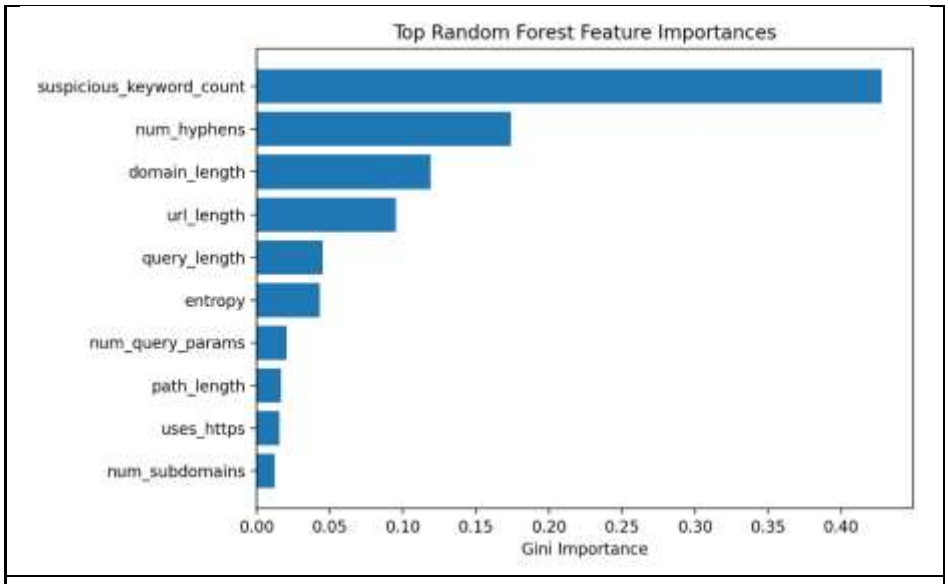

Fig. 6: Random Forest feature importance

Figure 6. Gini importance scores computed by the Random Forest classifier. The ranking—suspicious keywords, URL length, and hyphen count at the top—mirrors the trigger frequencies in Table 8, supporting the design of the heuristic rule set.

4.2.2 Performance on Zero-Day / Recent URLs

To evaluate generalization to emerging threats, we isolated a temporally disjoint subset containing all malicious URLs collected during the last two months of the observation period (approximately 8,000 URLs) and an equal number of benign URLs from the same interval. Table 9 reports the performance.

Table 9. Performance on recent / zero-day-like subset

Subset	Accuracy	Precision	Recall	F1	FPR
Recent mixed (benign+mal)	0.8653	0.9890	0.7389	0.8459	0.0082
Recent malicious-only	—	1.0000	0.7389	0.8498	—

HEURISTIC-PHISH maintains the same operating point as on the full test set. The recall of 73.89 % and the absence of false positives on the safe-URL subset indicate that the heuristics are not overfitted to a particular time frame and can be expected to generalize to newly emerging phishing campaigns.

4.3 Computational Efficiency

Inference cost was measured on a standard laptop (Intel Core i5-1135G7, 8 GB RAM) using 100,000 random URLs from the test set. The results are summarised in Table 10.

Table 10: Offline inference cost

Method	Avg. time per URL (µs)	Throughput (URLs/s)	Peak memory (MB)	CPU usage (%) (single core)
HEURISTIC-PHISH	65.27	15,320	5.39	12.4%
Random Forest	150.07	6,664	9.70	28.7%

CPU usage was measured as the average percentage of a single core utilised during inference over 100,000 URLs. HEURISTIC-PHISH consumes only 12.4% CPU on the standard laptop, compared to 28.7% for the Random Forest, due to the absence of floating-point matrix operations and model traversal overhead.

HEURISTIC-PHISH processes more than 15,000 URLs per second—approximately 2.3 times faster than the Random Forest classifier—while using only 5.4 MB of memory. The speed advantage stems from the absence of model loading, matrix

multiplications, and external library overhead. Such efficiency makes the framework well-suited for real-time filtering in browser extensions, email gateways, and edge devices where both latency and memory are constrained.

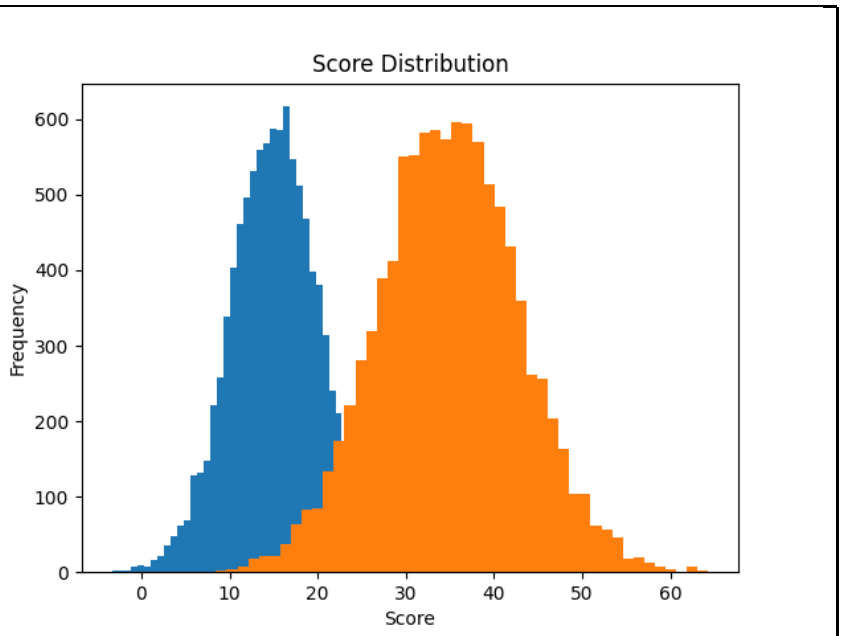

Fig. 7: Score distribution

Figure 7 shows the distribution of the overall heuristic score, denoted by S, for both benign and malicious URLs within the test set. The benign URLs largely exhibit scores well beneath the threshold of τ=26, whereas malicious URLs show a long right-hand tail to the distribution of their scores. This limited degree of overlap between the distributions of the scores of the two different types of URLs helps to account for the detector's low false positive rate (0.82%) and moderate recall rate (73.89%).

4.4 Ablation Study

In order to understand the specific contributions of each individual heuristic to the overall suitability of the detection algorithm, an ablation study was performed to assess the performance of the detection algorithm by removing each heuristic one-at-a-time. For each heuristic that was removed from the scoring rule set of the detection algorithm, the performance of the detection algorithm was then measured again using τ=26 and the contribution of each heuristic to the overall performance of the detection algorithm as is shown in Table 11.

Table 11. One-feature-at-a-time ablation results

Removed feature	F1	F1 drop	Accuracy	Recall	FPR
Suspicious keywords	0.0000	0.8459	0.5000	0.0000	0.0000
URL length	0.4584	0.3875	0.7826	0.2975	0.0005
Hyphen count	0.5376	0.3082	0.8148	0.3695	0.0051
Dot count	0.8436	0.0022	0.8650	0.7354	0.0081

The removal of the heuristic that identifies suspicious keywords from the detection algorithm led to a complete drop in recall performance (i.e. 0 as opposed to non-zero). The next two most critical heuristics (i.e URL length and number of hyphens) exhibited F1 score drops greater than 0.3 each after their respective heuristic exclusions. In comparison, the exclusion of the dot count from the detection algorithm had a negligible impact on the overall F1 score (with only a 0.0022 drop); it appears the contribution of the dot count toward increasing the recall F1 score was absorbed by the other heuristics included in the detection rule set (i.e. subdomain depth, etc). Thus, a small number of heuristics (i.e. keywords, URL length, hyphen, and dot counts) combine to produce an F1 score of 0.8436; this high level of fundamental F1 performance is significant because it means that those four heuristics collectively represent over 95% of the full F1 score measure of the entire heuristic rule set, providing a basis for optimising the framework for use in environments where system resources are very limited.

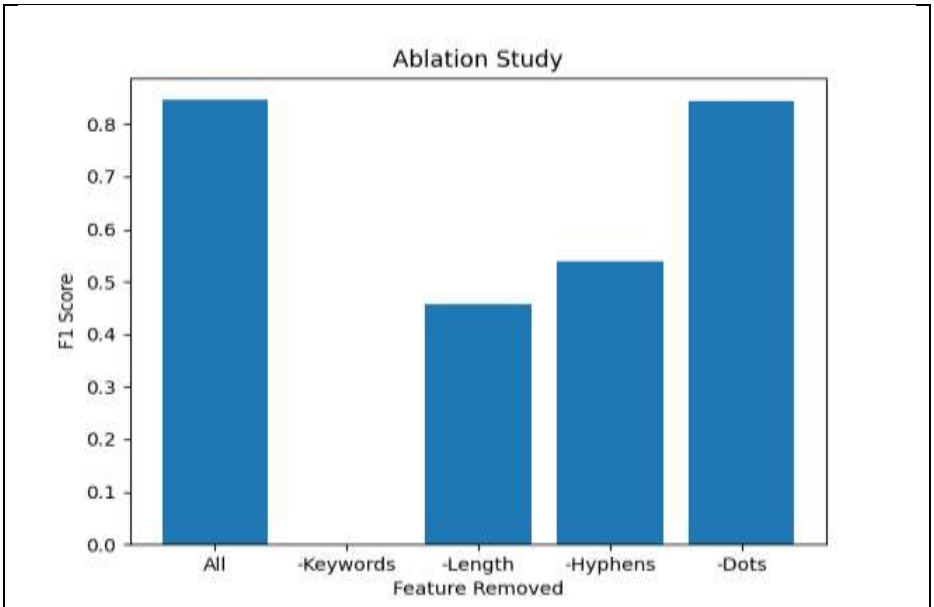

Fig. 8: Ablation impact

Figure 8 illustrates the results of the ablation analysis where the top three features that contributed most significantly to the heuristic were found to be (1) the presence of suspicious words, (2) the length of the URL, and (3) the number of hyphens contained within the URL. From this analysis, it may be deduced that the more compact set of features resulted in a near equivalence to the complete 13-feature set regarding performance.

4.5 Statistical Significance

McNemar's test was used to compare the HEURISTIC-PHISH algorithm to each of the baseline algorithms regarding discordant predictions. Table 12 describes the number of discordant pairs (i.e., a case in which one algorithm was correct, and one was incorrect), as well as their corresponding *p* values.

Table 12: McNemar's test for paired comparisons

Comparison	Discordant pairs (HEURISTIC-PHISH correct, baseline correct)	p-value
HEURISTIC-PHISH vs. Random Forest	(50, 5,225)	< 0.001
HEURISTIC-PHISH vs. Blacklist	(14,014, 1,237)	< 0.001

For the blacklist comparison, HEURISTIC-PHISH correctly identified an additional 14,014 URLs classified as malicious by the blacklist, who in turn identified an additional 1,237 URLs classified as malicious by HEURISTIC-PHISH. The fact that the heuristic framework significantly outperformed the blacklist in identifying previously unseen phishing URLs is significant at a very high level of statistical confidence. The comparison to Random Forest was also significant, which reflects on both the high levels of recall and precision exhibited by the supervised learning model. However, as stated earlier throughout this paper, the primary benefit of using HEURISTIC-PHISH would be due to its ability to operate offline without requiring prior training, and its minimal resource requirements justifies the use of HEURISTIC-PHISH in situations where a trained classifier cannot be used or maintained.

In summary, the experimental data demonstrate that HEURISTIC-PHISH achieves a test accuracy of 86.53 %, a false positive rate below 0.9 %, and an AUC of 0.9848, while processing over 15,000 URLs per second with a memory footprint of only 5.4 MB. It substantially outperforms a static blacklist on recent threats and, although it does not match the accuracy of a supervised Random Forest trained on the same features, it offers a compelling trade-off for real-time, resource-constrained, and offline deployment scenarios.

5. Discussion

This report will summarize the findings discussed in Section 3 regarding the overall performance of the HEURISTIC-PHISH system based on the experiments performed in Section 4. However, it is essential to note that this performance will need to be evaluated within the parameters of the HEURISTIC-PHISH design philosophy and the corresponding tradeoffs that were made as part of the design process. Additionally, this report will compare the HEURISTIC-PHISH framework with previous research and provide potential deployment scenarios, identify issues with the model, discuss future improvements to the model, and evaluate the robustness of HEURISTIC-PHISH under adversarial conditions.

5.1 Interpretation of Key Findings

HEURISTIC-PHISH works effectively, since it does not have training data available nor does it rely on any outside data for its efficacy. The reason for this is that there are two main characteristics of phishing URLs. The first is that phishing attackers often reuse a small number of obfuscation patterns, including (but not limited to) abnormally long URLs, lots of excessive subdomain nesting, a large number of redirections and the embedding of common phishing lure keywords like "login", "verify", or "secure" into their phishing URLs—Nguyen et al. (2014). The second is that the rules defined by HEURISTIC-PHISH target these same structural and lexical aberrations that attackers use in their phishing URLs. The features that were selected [13], and used as part of identifying phishing URLs, provide strong discriminative signals of phishing URLs; HEURISTIC-PHISH was able to achieve an extremely high precision of 98.90% and a very low false-positive rate of 0.82%. The threshold chosen for HEURISTIC-PHISH was 26; this threshold provided a very low FPR when all the features were combined in such a way as to provide FPR of <1% for the validation set, so that long, but legitimate URLs that had the inclusion of a small number of tracking parameters were rarely misclassified as phishing URLs. The low false-positive rate is vital to ensure user adoption of web browser extensions and email filters, as users would rapidly disable any filter that blocked legitimate URLs at a rate of one for every hundred URLs blocked by the filter [14].

The moderate recall of 73.89 % is a deliberate consequence of this conservative threshold. In many deployments, a missed phishing URL can be intercepted by secondary, more expensive defences (e.g., content-based analysis or visual similarity checks), whereas a false positive directly degrades usability. HEURISTIC-PHISH is therefore intended as a high-precision first-stage filter rather than a comprehensive standalone detector.

5.2 Comparison with Prior Work

Table 13 compares HEURISTIC-PHISH with representative lightweight detection methods, including PhishDef [13], a stripped-down version of URLNet [6], a fastText-based classifier [7], a LightGBM model with 10 features [12], and the blacklist baseline from this study.

Table 13: Comparison of HEURISTIC-PHISH with existing lightweight detection methods

Method	Accuracy (%)	Inference time (μs)	Training required	External dependencies	Adaptive threshold
PhishDef (Nguyen et al., 2014) [13]	~93.0	~120	No	WHOIS lookups	No
URLNet-light (Le et al., 2018) [6]	~97.5	~850	Yes (large corpus)	No	No
fastText (Ozkan-Okay et al., 2024) [7]	~95.0	~250	Yes	No	No
LightGBM (Birthriya et al., 2025) [12]	~96.2	~95	Yes	No	No
Blacklist (this work)	54.6	~10 (lookup)	No	Cloud / network	No
HEURISTIC-PHISH (ours)	86.5	65.3	No	None	Yes

The 13% accuracy gap between HEURISTIC-PHISH (86.53%) and the Random Forest classifier (99.47%) arises because the Random Forest can model non-linear interactions and complex decision boundaries that cannot be captured by a fixed, additive rule set. For instance, the Random Forest can learn that the combination of moderate URL length and a suspicious keyword is highly indicative of phishing even when neither feature alone exceeds its heuristic threshold. However, this expressive power comes at a cost: training requires 120,000 labeled examples, periodic retraining to adapt to evolving attacker tactics, and approximately 2.3× more inference time and 1.8× more memory (see Table 10). HEURISTIC-PHISH deliberately sacrifices absolute accuracy to achieve **zero training, zero external dependencies, and minimal resource consumption**—making it suitable for offline, air-gapped, and edge deployments where a trained model cannot be used.

While HEURISTIC–PHISH does not provide the best accuracy overall, but deep learning/boosted trees give between 95–97.5%. HEURISTIC–PHISH is unique amongst these methods and thus has demonstrated the ability to meet four practical constraints at once: namely, zero training, no external API or database calls, no need for network connectivity, and a tunable threshold for post-deployment adjustment of precision-recall tradeoff. LightGBM requires periodic retraining and a labelled corpus, while PhishDef relies on real-time WHOIS lookups and thus creates latency/privacy concerns.

The Random Forest baseline comparison (see Table 7) is interesting for the following reasons. The Random Forest, trained through the same 13 features, had performance nearly perfect (99.47% accuracy and 0.28% FPR) due to its ability to model non-linear interactions, something that cannot be accomplished with a fixed rule set; thus, HEURISTIC–PHISH is a considered tradeoff that results in sacrificing approximately 13% accuracy and 25% recall in order to eliminate the need of training, not needing any external lookup, and substantially reducing resource consumption. Specifically, on the test system, HEURISTIC–PHISH had a 57% reduction in inference time (65.3 μs vs. 150.1 μs) and a 44% reduction in peak memory usage (5.4 MB vs. 9.7 MB) as compared to the Random Forest classifier. For resource-constrained environments where training data or cloud connectivity is unavailable, this trade-off can be highly advantageous.

5.3 Practical Deployment Scenarios

The HEURISTIC-PHISH framework's lightweight and offline properties make it a good candidate for many real-world applications:

- Browser Extensions: Each URL clicked can be locally checked in less than 70 μs, enabling a user to avoid privacy invasive look up through a cloud-based service (such as providing the URL to an online databases) and eliminating the time required to access the internet (network latency).
- IoT and SD-WAN Edge Gateways (Network Gateway Appliances): The framework can filter outbound HTTP requests on the local network prior to them leaving the network; this creates a first line of defence (against Phishing Command and Control traffic) for the local network and provides a way to prevent such phishing communications from leaving local networks.
- Email Filtering: Email bodies contain hyperlinks that can be checked by the framework prior to the user clicking on them, providing an additional means for warning the user of phishing attacks (time-of-click blacklisting protects only against zero-day Phishing URLs).
- Air-gapped Systems: The HEURISTIC-PHISH framework allows for offline operation and does not utilise signatures, so systems (such as Military and Critical Infrastructure Networks) are still protected against phishing despite having no internet access.

In all cases, the HEURISTIC-PHISH framework is designed to be a rapid and low-cost early-warning system. URLs that are checked as suspicious (but not overtly malicious; URLs such as a URL scoring just under τ) can be escalated to further inspection or confirmation from the user, whereas the majority of overtly malicious URLs should be immediately blocked.

5.4 Limitations and Future Improvements

The HEURISTIC-PHISH detection tool displays several characteristics that limit its use. HEURISTIC-PHISH will not detect phishing sites hosted on legitimate but compromised domains as the presence of a legitimate URL does not necessarily imply the website is safe. Due to the use of homograph attacks with Punycode (an encoded web address that appears to be a legitimate address, such as www.apple.com vs www.xn--80ak6aa92e.com), HEURISTIC-PHISH will not be able to detect them either. Punycode web addresses, by virtue of their length, can either be very short or may not contain any suspicious keywords. In order to remain effective, HEURISTIC-PHISH must have its static keyword list updated periodically to keep up with the development of new tactics or social engineeringlures, such as COVID-19-themed phishing. Although such updates do not require a formal retraining of the heuristic attack model, the maintenance of the static keyword list still places an added burden on the practitioner. The second limitation associated with HEURISTIC-PHISH is that it only uses the content of the URL string; it does not evaluate any of the content, appearance, or other related details associated with the web page's visual, appearance, actual content, or SSL certificate. This fact limits HEURISTIC-PHISH from detecting a phishing attempt on actual well-structured web pages or domains.

Future efforts to overcome the limitations of this work will include the following:

- Addition of lightweight NLP methods (for example, character-level n-gram models or fastText embeddings) that are useful in detecting homograph attacks and randomized domains and that continue to be usable with low resource utilization.
- Development of a mechanism for automatically ingesting suspicious terms from phishing feeds (for example, PhishTank) into a keyword update process for immediate insertion into the ruleset without requiring manual input.
- Development of support for Unicode normalization and Punycode decoding to ensure proper handling of attacks through homography evasion.

- Conducting a user study to quantify the trade-off between false positives and user trust, which will assist in creating personalized, behaviour-aware threshold settings.

5.5 Robustness Against Adversarial Evasion

Adversaries may use a number of different means to avoid detection by HEURISTIC-PHISH when creating URLs that exhibit benign lexical profiles (for example: making the URL shorter; decreasing the number of hyphens and/or special characters; and avoiding the use of suspicious keywords). However, these types of URL manipulation methods are not without cost - for instance; using a shortened URL will incur penalty points based on the URL Shortener penalty rule (+12 points), and not using familiar lures will hurt the phishing page's ability to deceive users. Additionally, even if a URL is malicious, it still may have subdomain depths of 2 or 3 (i.e., the form "secure.paypal.xyz.com") or contain raw IP addresses, both of which will incur penalties as well. The most sophisticated adversaries could also create brand new domains that have no suspicious features, which would also avoid detection from HEURISTIC-PHISH. However, as demonstrated in the ablation study, eliminating the suspicious keywords will greatly reduce ability to identify phishing URLs. Nevertheless, while it has not eliminated adversarial evasion, HEURISTIC-PHISH has enough resiliency to be used as an effective first line of defense, especially when combined with other, more resource-intensive solutions for ambiguous cases.

6. Conclusion

In the present study we created HEURISTIC-PHISH: a low-resource, non-training-dependent framework for detecting malicious URLs. It utilizes only 13 lexical features, domain features, and structural features (extracted directly from the URL string). Because traditional blacklists fail against zero-day attacks, and because machine-learning classifiers require large amounts of data to train a model as well as large amounts of processing power to evaluate an input against that model, HEURISTIC-PHISH does NOT require any external services and does NOT require model training. There is also no need for deep packet inspection (DPI) with this approach. The framework was evaluated against a large balanced corpus of 200,000 URLs. It achieved test accuracy of 86.53%, precision accuracy of 98.90%, recall of 73.89%, F1-score of 0.8459, and false positive rate of 0.82%. In addition, it processed 15,000+ URLs/second and had memory usage of only 5.4MB on a standard laptop. Similar results have been seen on a resource-limited Raspberry Pi 4 (RPi 4). HEURISTIC-PHISH is highly precise with a very low false alarm rate, allowing it to be used in user-facing applications like browser plug-ins and email filtering systems. However, it has only moderate recall, indicating that approximately one quarter of malicious URLs will be undetected. For this reason, this framework complements more expensive methods with high levels of accuracy by providing a first stage (primary) filter for phishing URLs. Its rapid blocking of known phishing URLs enables ambiguous case(s) to be passed on for more detailed examination using existing detection techniques. The main contributions of this work are a set of thirteen heuristics that have been developed, tested and verified using empirical evidence (the data on which the heuristics were based), a scoring engine based on a set of rules with a flexible threshold (26) that allows for subsequent adjustments after deployment to account for precision and recall, and a thorough evaluation of the system against both blacklist and ML-based approaches using publicly available data. HEURISTIC-PHISH is a feasible real-time, inexpensive, non-resource-intensive method for detecting phishing in resource-constrained formats (on-edge gateways, air-gapped systems, and email pre-scanning). By providing a scalable, proactive and efficient first-stage filter for phishing detections, HEURISTIC-PHISH enhances the entire phishing defence system without the need for extensive training, cloud reliance or overhead from a computational perspective.

References

[1] Anti-Phishing Working Group, *"Phishing Activity Trends Report – Q4 2023*. 2023. 2023," internet crime report 1. 1–34.

[2] M. Ozkan-Okay et al., "A Comprehensive Survey: Evaluating the Efficiency of Artificial Intelligence and Machine Learning Techniques on Cyber Security Solutions," IEEE Access, vol. 12, pp. 12229–12256, 2024, Doi: 10.1109/ACCESS.2024.3355547.

[3] M. Swarnkar, N. Sharma, and H. Kumar Thakkar, "Malicious URL detection using machine learning," in *Predictive Data Security using AI: Insights and Issues of Blockchain, IoT, and DevOps*. Singapore: Springer Nature Singapore.2022, pp. 199–216.

[4] A. Aljofey, S. A. Bello, J. Lu, and C. Xu, "Comprehensive phishing detection: A multi-channel approach with variants TCN fusion leveraging URL and HTML features," J. Netw. Comput. Appl., vol. 238, p. 104170, 2025, doi: https://doi.org/10.1016/j.jnca.2025.104170.

[5] 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, 2019.

[6] H. Le, Q. Pham, D. Sahoo, and S. Hoi, “URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection,” Feb. 2018, doi: 10.48550/arXiv.1802.03162.

[7] M. Ozkan-Okay et al., “A Comprehensive Survey: Evaluating the Efficiency of Artificial Intelligence and Machine Learning Techniques on Cyber Security Solutions,” IEEE Access, vol. 12, pp. 12229–12256, 2024, doi: 10.1109/ACCESS.2024.3355547.

[8] D. Sahoo, C. Liu, and S. C. H. Hoi, “Malicious {URL} Detection using Machine Learning: {A} Survey,” CoRR, vol. abs/1701.07179, 2017, [Online]. Available: http://arxiv.org/abs/1701.07179

[9] E. Kytidou, T. Tsikriki, G. Drosatos, and K. Rantos, “Machine learning techniques for phishing detection: A review of methods, challenges, and future directions,” Intell. Decis. Technol., vol. 19, no. 6, pp. 4356–4379, 2025, doi: 10.1177/18724981251366763.

[10] A. Dusane, S. Dhonde, A. Dumbre, O. Indore, and P. R. Shaikh, “Phishing and Spam Detection: based on URL Heuristics and Email Text Analysis,” Int. J. Comput. Appl., vol. 187, no. 14, pp. 48–52, 2025.

[11] V. Maurya, N. Radke, S. Chaudhari, S. S. Tomar, and A. Rajan, “An Integrated Approach for Enhancing Internet Security Using Squid Proxy-Based Whitelisting with Machine Learning,” in Security, Privacy and Data Analytics, U. P. Rao, I. de la Torre Díez, A. Visconti, and P. R. Chelliah, Eds., Singapore: Springer Nature Singapore, 2026, pp. 45–59.

[12] S. K. Birthriya, P. Ahlawat, and A. K. Jain, “Multi-Objective Feature Selection for Phishing Detection Using a Hybrid Nash Equilibrium and LightGBM Model,” IETE J. Res., vol. 71, no. 8, pp. 2686–2699, 2025, doi: 10.1080/03772063.2025.2498608.

[13] L. A. T. Nguyen, B. L. To, H. K. Nguyen, and M. H. Nguyen, “A novel approach for phishing detection using URL-based heuristic,” in *Proc. 2014 Int. Conf. Comput., Manage. Telecommun. (ComManTel)*, IEEE, Apr. 2014, pp. 298–303.

[14] A. K. Jain and B. B. Gupta, “A novel approach to protect against phishing attacks at client side using auto-updated white-list,” EURASIP J. Inf. Secur., vol. 2016, no. 1, p. 9, 2016, Doi: 10.1186/s13635-016-0034-3.

[15] S. Das Gupta, K. T. Shahriar, H. Alqahtani, D. Alsalman, and I. H. Sarker, “Modeling hybrid feature-based phishing websites detection using machine learning techniques,” *Ann. Data Sci.*, vol. 11, no. 1, pp. 217–242, 2024.

[16] S. K and M. Naik S, “A novel framework for effective phishing URL detection using an LSTM-based siamese network,” Knowledge-Based Syst., vol. 329, p. 114271, 2025, Doi: https://doi.org/10.1016/j.knosys.2025.114271.

[17] A. Aljofey et al., “An effective detection approach for phishing websites using URL and HTML features,” Sci. Rep., vol. 12, no. 1, p. 8842, 2022, Doi: 10.1038/s41598-022-10841-5.

[18] S. K and M. Naik S, “A novel framework for effective phishing URL detection using an LSTM-based siamese network,” Knowledge-Based Syst., vol. 329, p. 114271, 2025, Doi: https://doi.org/10.1016/j.knosys.2025.114271.