*future internet*

MDPI

*Article*

# Towards Proactive Domain Name Security: An Adaptive System for .ro domains Reputation Analysis

**Carmen Ionela Rotună [1,2,*], Ioan Ștefan Sacală [3] and Adriana Alexandru [1,4,*]**

[1] National Institute for Research and Development in Informatics, 011455 Bucharest, Romania
[2] Doctoral School of Computer Science, Information Technology and Systems Engineering, National University of Science and Technology Politehnica Bucharest, 060042 Bucharest, Romania
[3] Department of Automatic Control and Systems Engineering, University Politehnica of Bucharest, 313 Splaiul Independenței, 060042 Bucharest, Romania; ioan.sacala@upb.ro
[4] Faculty of Electrical Engineering, Electronics and Information Technology, Valahia University of Targoviste, 130004 Targoviste, Romania
* Correspondence: carmen.rotuna@ici.ro (C.I.R.); adriana.alexandru@ici.ro (A.A.)

**Abstract**

In a digital landscape marked by the exponential growth of cyber threats, the development of automated domain reputation systems is extremely important. Emerging technologies such as artificial intelligence and machine learning now enable proactive and scalable approaches to early identification of malicious or suspicious domains. This paper presents an adaptive domain name reputation system that integrates advanced machine learning to enhance cybersecurity resilience. The proposed framework uses domain data from .ro domain Registry and several other sources (blacklists, whitelists, DNS, SSL certificate), detects anomalies using machine learning techniques, and scores domain security risk levels. A supervised XGBoost model is trained and assessed through five-fold stratified cross-validation and a held-out 80/20 split. On an example dataset of 25,000 domains, the system attains accuracy 0.993 and F1 0.993 and is exposed through a lightweight Flask service that performs asynchronous feature collection for near real-time scoring. The contribution is a blueprint that links list supervision with registry/DNS/TLS features and deployable inference to support proactive domain abuse mitigation in ccTLD environments.

**Keywords:** domain reputation; cybersecurity; adaptive system; machine learning; threat intelligence; monitoring; ccTLD; DNS; WHOIS; XGBoost

check for updates

Academic Editors: Wei Zong and Yang-Wai Chow

Received: 4 September 2025
Revised: 14 October 2025
Accepted: 15 October 2025
Published: 18 October 2025

**Citation:** Rotună, C.I.; Sacală, I.Ș.; Alexandru, A. Towards Proactive Domain Name Security: An Adaptive System for .ro domains Reputation Analysis. *Future Internet* **2025**, *17*, 478. https://doi.org/10.3390/fi17100478

**Copyright:** © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

In a digital landscape marked by the exponential growth of cyber threats, the development of an automated domain reputation monitoring system is a necessity. Emerging technologies such as artificial intelligence and machine learning currently allow for proactive and scalable approaches to the early identification of malicious or suspicious domains [1,2]. From a technical point of view, this type of system is based on the real-time analysis of large volumes of data from various sources such as logs, WHOIS records, blacklists and other sources.

By correlating this data with behavioural patterns learned from the history of compromised domains, domain reputation systems can assign automated, dynamic risk scores that reflect the likelihood that a domain will be used for malicious purposes. Unlike traditional methods, where domain annotation is performed manually following post-incident investigations, an automated system can react in near real-time, giving authorities or Internet

service providers the ability to block or restrict suspicious domains before they cause significant damage [3,4].

The objective of this research paper is to design, build, and empirically validate an automated domain reputation system for the .ro ccTLD that distinguishes benign from malicious domains using a labelled dataset and machine learning. The goal is fulfilled by constructing a domain dataset from the .ro Domain Registry domains list intersected with whitelists and multiple abuse blacklists. For each domain, the activity is verified using the registry WHOIS service, registration information, nslookup (Name Server Lookup) and SSL/TLS features. Training and evaluation are performed by using an XGBoost classifier with stratified cross-validation while using a lightweight Flask service for real-time scoring.

In addition to its technical importance, a domain name reputation system is essential, considering the responsibilities and strategic directions of international actors involved in Internet governance. ICANN (Internet Corporation for Assigned Names and Numbers), as the global body responsible for coordinating the Domain Name System (DNS), promotes initiatives to secure the digital space and supports policies to prevent misuse of the DNS infrastructure. In the same registry, RIPE NCC, as a regional Internet registry for Europe, provides technical support for network operators and promotes best practices in IP resource allocation and network security [5].

*Introduction* chapter presents an automated, machine-learning domain-reputation system that correlates DNS, WHOIS, whitelist and blacklist telemetry for proactive domain reputation detection. Also highlights the work within ICANN and RIPE NCC security objectives.

*Related works* chapter overviews DNS reputation and DGA detection systems such as Notos, Kopis, Pleiades, DeepDGA, Segugio, Phoenix, and BotHunter, alongside operational tooling like DAAR, URLhaus, MISP, IntelOwl, and Spamhaus and identifies methodological strengths and weaknesses across these approaches.

Domain reputation lists used within the study are whitelists (Alexa, Cisco Umbrella, Cloudflare Radar) and blacklists (Spamhaus DBL, SURBL, PhishTank, OpenPhish, URLhaus, Google Safe Browsing). The proposed solution combines the 2 types of lists for training an adaptive system and will add behavioural and registration features.

*The method used* chapter constructs a labelled .ro corpus by merging Alexa whitelists with multiple blacklists under a blacklist precedence rule and validates activity through .ro WHOIS queries on port 4343. Engineers WHOIS and SSL/TLS features, trains an XGBoost classifier in a reproducible pipeline, and expose a minimal Flask scoring service.

*Evaluation of the model* applies stratified five-fold cross-validation followed by an 80/20 train and test split, reporting accuracy, precision, recall, F1, and a confusion matrix. It uses these diagnostics to characterize error modes and assess generalization.

*Discussion* reports an accuracy of 0.993 after processing, filtering, normalization and adding WHOIS and nameserver data. It raises caution about class imbalance and potential leakage and recommends calibration, grouped and time-aware validation, and comparisons across learners.

*Conclusion* summarizes a deployable reputation pipeline that integrates curated lists, activity verification, feature extraction, and supervised learning. Future work will include model calibration, cost-sensitive thresholds, and Random Forest versus XGBoost comparisons to strengthen external validity.

## 2. Related Works

This chapter surveys systems developed for malicious domain detection and reputation, with emphasis on DNS traffic analysis and the integration of threat-intelligence sources. It covers both foundational algorithmic models based on clustering, Bayesian

classification, statistical profiling, graph-based community detection, and recurrent neural networks. For each system, it is outlined with underlying assumptions, required data types, advantages and limitations. The comparison of these systems highlights accuracy, implementation cost, and robustness, with Table 1 synthesizing reported performance.

### 2.1. Notos

Notos was developed [1] as a domain reputation system based on passive DNS analysis. The system uses DNS query patterns and associated infrastructure to cluster domains according to shared characteristics. Notos analyses passive DNS attributes, namely IP addresses, autonomous systems (ASNs), and co-location with known benign or malicious domains for legitimate and malicious domains. The model uses a Bayesian classification framework for assigning reputation scores based on historical DNS behaviour patterns. The system also makes inferences about new domains that are ingested.

The advantage of Notos is its passive, infrastructure-based methodology, which eliminates the need for domain content analysis or dependence on pre-existing blacklists. This design enables the system to identify malicious domains at an early stage, before they are listed on conventional threat lists. The effectiveness of Notos is dependent on access to large-scale passive DNS datasets, a resource basically available only to major internet service providers or specialized security vendors. Its clustering algorithms have limitations when analyzing domains associated with fast-flux techniques or highly dynamic hosting infrastructures, which may result in a higher incidence of false negatives.

### 2.2. Kopis

Kopis, also developed by [6], extends Notos and introduces behavioural analysis for authoritative name servers (NS). While Notos focuses on the domain side, Kopis observes query patterns directed to authoritative DNS servers and analyses their response behaviour. The system models legitimate NS behaviour and compares it with patterns associated with malicious domains. It uses statistical profiling to detect anomalies that may indicate malicious domain hosting or domain generation algorithms (DGAs).

The advantage of Kopis is its upstream focus, which enables the detection of malicious domains by monitoring their DNS infrastructure instead of end-host queries. This can reveal coordinated malicious activities even before domains are widely queried. Kopis also faces deployment challenges because it requires monitoring at strategic points near authoritative DNS servers, limiting its applicability to certain network vantage points. The dependence on statistical baselines may also expose the system to evasion techniques employed by adaptive attackers [7].

Figure 1 illustrates the monitoring layers within the DNS infrastructure. Notos observes domain resolution activity at the recursive resolver (RDNS) level, where client queries are aggregated. Notos analyses passive DNS data and infers reputation scores from domain to infrastructure relationships. In contrast, Kopis operates one tier higher, at the authoritative DNS server (AuthNS) level, where it inspects query patterns directed to authoritative servers and characterizes abnormal response behaviours. The hierarchical arrows in the figure represent the direction of DNS query propagation from RDNS to TLD and root servers illustrating how Kopis extends the Notos approach by shifting the monitoring perspective upstream to detect coordinated malicious activities earlier in the resolution process.

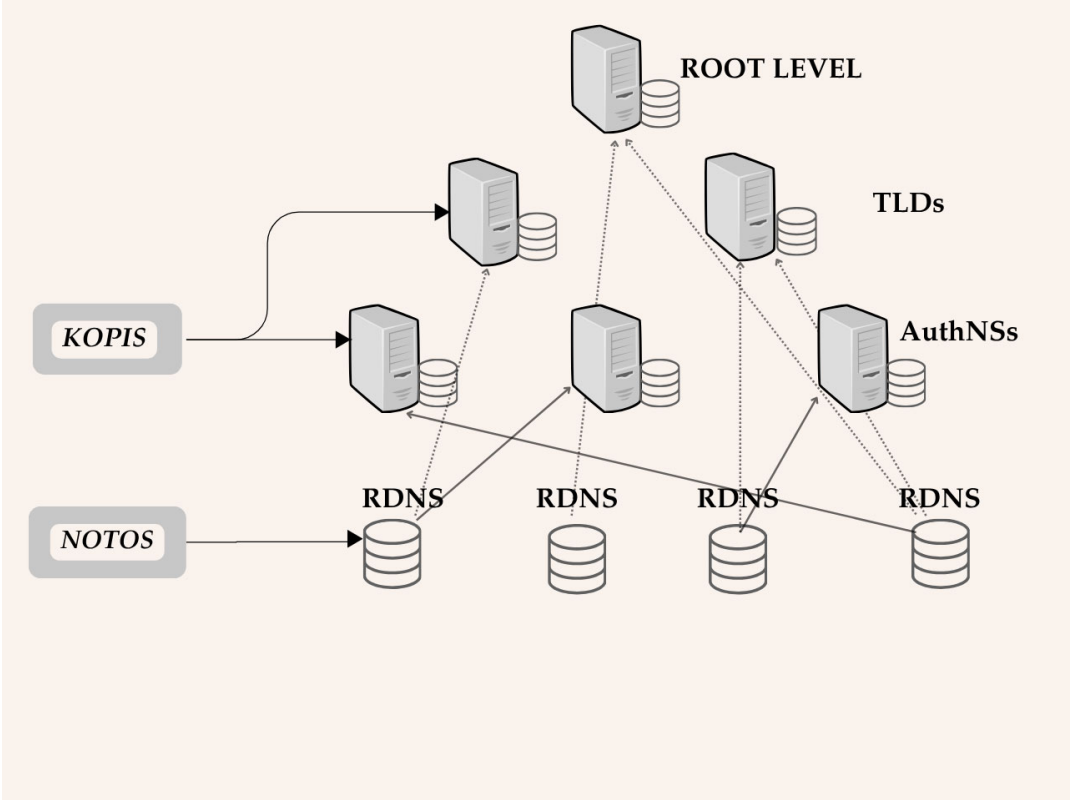


**Figure 1.** DNS monitoring hierarchy illustrating the observation points of Kopis and Notos.

By contrast to Notos and Kopis, the framework proposed within this paper operates at the registry infrastructure level, combining authoritative DNS resolution data, WHOIS registration information, and SSL/TLS certificate metadata. This position allows the system to evaluate domain reputation directly from registry sources without relying on passive DNS telemetry, making it both lightweight and suitable for ccTLD environments such as .ro.

### 2.3. Pleiades

Pleiades, developed by Bilge et al. in 2014 [8], is a domain reputation system focused on detecting domains generated by domain generation algorithms. It relies on graph-based clustering methods to analyze passive DNS traffic, focusing on NXDOMAIN responses. Pleiades creates bipartite graphs that connect hosts with the domains they query. The system identifies groups of domains that are frequently queried together by potentially compromised hosts. Community detection algorithms are subsequently applied to these graphs to isolate clusters exhibiting suspicious characteristics.

Pleiades detects low-frequency, stealthy DGA domains that traditional signature-based systems often miss. Its use of community detection provides a scalable method for large datasets. Pleiades requires comprehensive DNS traffic visibility, making it difficult to deploy in environments with partial network coverage. Malicious actors employing high-frequency DGAs or leveraging encrypted DNS protocols such as DoH (DNS over HTTPS) or DoT (DNS over TLS) can evade detection or obscure their activities, thereby diminishing the overall effectiveness of the system.

### 2.4. DeepDGA

DeepDGA, introduced by Anderson et al. in 2016 [9], applies recurrent neural networks (RNNs), long short-term memory (LSTM) models, to categorize domain names

based on their threat potential as either malicious or non-malicious. The system treats domain names as sequences of characters and trains a neural model to capture sequential dependencies characteristic of DGAs. Unlike feature-engineered methods, DeepDGA learns representations directly from domain name data, enabling it to generalize across multiple DGA families, including those skipped during training.

DeepDGA employs a deep learning approach that requires minimal engineering and demonstrates strong detection rates against diverse DGAs. It adapts well to the introduction of new malware families due to its sequence modelling capabilities. The model has a black-box nature that reduces interpretability, complicating incident response and forensic analysis. DeepDGA requires substantial labelled datasets for training, which can limit its applicability in environments with limited data. The computational overhead for inference can also be a concern for real-time applications.

### 2.5. Segugio

Segugio was developed by Perdisci et al. in 2016 [10] and combines DNS traffic analysis with malicious indicators to detect domain activities related to botnets. The system monitors DNS query response patterns throughout a network and correlates them with behavioural signatures of malware. The system continuously monitors the DNS query-response patterns across the network and correlates them with behavioural patterns exhibited by malware during command-and-control communications or domain generation algorithm (DGA) activity.

Segugio has a hybrid approach of behavioural monitoring and a signature analysis that improves its resilience against zero-day botnets and known malware. It can adapt well to varying network conditions and it is scalable. Segugio depends on maintaining updated threat intelligence patterns and it can produce false positives in networks with atypical DNS behaviours. The reliance on internal network traffic visibility limits its utility in scenarios where encrypted DNS protocols are prevalent [11].

### 2.6. Phoenix

Phoenix was developed by Ho et al. in 2017 [12] to address the challenge of detecting zero-day malicious domains at first sight. It uses a combination of temporal (query timing) and spatial (query source distribution) features to analyze DNS traffic patterns. The system models non-malicious domain query patterns and identifies deviations that may indicate new malicious domains. By comparison with traditional systems relying on content analysis or historical blacklists, Phoenix focuses on behavioural anomalies as early indicators of malicious activity.

Phoenix is efficient for early detection, enabling security teams to block malicious domains before they are widely recognized or included in blacklists. Its reliance on temporal and spatial features makes it difficult for malicious actors to evade without altering attack patterns. The main disadvantage of the system is that it may return false positives in dynamic or diverse network environments where query patterns vary. Phoenix requires continuous recalibration to maintain accuracy as network usage evolves.

### 2.7. BotHunter

BotHunter is a network tool developed by Gu et al. in 2007 [13] for tracking botnet infections by correlating network traffic patterns. BotHunter does not focus only on domain names. Instead, it monitors various stages of malware infection, such as inbound scanning, exploit delivery, binary download, and command and control (C2) communication. It relies on a correlation engine that matches sequences of events against known malware infection profiles. BotHunter is not a domain reputation system, but it can flag domains involved in botnet activities by monitoring C2 interactions and infection behaviours.

BotHunter has a comprehensive correlation approach, which examines multiple phases of infection instead of isolated events. This correlation significantly reduces false positives compared to single-event detectors. Its dependency on predefined infection models makes it less effective against new attack vectors. BotHunter was designed for network deployment and may not scale efficiently in high-throughput environments without significant resource allocation.

### 2.8. PyDGADetector

PyDGADetector v1.0.0 is an open-source tool, developed in Python for detecting domains generated by domain generation algorithms (DGAs). It implements machine learning models, such as logistic regression and random forests, trained on lexical features extracted from domain names. These features include character distributions, entropy and domain length. In contrast to deep learning methods, PyDGADetector prioritizes efficient deployment by employing traditional machine learning models built on explainable feature sets.

The advantage of PyDGADetector is its simplicity and transparency, which makes it suitable for rapid deployment and educational purposes. It enables potential users to experiment with DGA detection using feature sets without needing expertise. Its performance may be inferior to that of more advanced models, such as RNN-based detectors, particularly in complex or highly variable domain generation algorithms (DGAs). Additionally, reliance on static lexical features may lead to higher false positives or negatives in adversarial contexts [14].

### 2.9. DAAR (Domain Abuse Activity Reporting)

DAAR, developed by ICANN in 2017 [15], is a data analysis and reporting project developed for studying domain name abuse across top-level domains (TLDs), including country-code TLDs (ccTLDs). It collects publicly available data on domain abuse like passive DNS data, malware feeds, phishing sources, and spam reports. DAAR compiles statistics on abuse rates associated with specific TLDs, providing trend reports on malware hosting, phishing, botnet command-and-control domains, and spam related abuse. Although DAAR does not block domains, it offers an overview of abuse activities to assist registry operators in policy development and abuse mitigation strategies.

DAAR provides aggregated insights that help ccTLD operators identify trends and systemic abuse issues within their namespace. It supports transparency initiatives by offering objective data on domain abuse metrics. DAAR is a statistical reporting tool rather than a real-time detection system and it is better suited for policy and research purposes rather than immediate operational response. Additionally, the effectiveness of DAAR depends on the quality and coverage of the external feeds it integrates.

### 2.10. Abuse.ch Threat Intelligence Feeds

Abuse.ch [16] provides free threat intelligence feeds that focus on tracking malware, botnet activity, and malicious domain usage. Key projects include URLhaus, MalwareBazaar, and the SSL Blacklist (SSLBL). These feeds collect real-time indicators of compromise (IoCs) such as malicious domains, IP addresses, URLs, and SSL certificates associated with cybercriminal operations. ccTLD operators can use these feeds for their monitoring systems by cross-checking domain registrations, DNS queries, or user reports against known malicious indicators.

Because it is an open-source tool it is useful for ccTLD operators to monitor domain abuse in an effective way. It offers intelligence that can be integrated into automated screening tools or threat intelligence platforms like MISP [17]. However, since the feeds

focus on known abuse, it may not detect emerging threats or zero-day domains. The accuracy of the feeds depends on community contributions and partnerships [16].

### 2.11. MISP (Malware Information Sharing Platform)

MISP is an open-source threat intelligence platform developed for information sharing, correlation, and analysis of cybersecurity indicators. Initially developed by CIRCL (Computer Incident Response Center Luxembourg), MISP enables organizations, including ccTLD registries, to collect and disseminate threat data comprising malicious domains, IP addresses, hashes, and URLs. The platform supports both manual entry and automated ingestion of threat feeds, with correlation features that help identify links between different abuse events. MISP includes community sharing, allowing ccTLDs to benefit from collaborative threat intelligence networks.

MISP can be customized and supports both internal analysis and external threat sharing, which is essential for ccTLDs managing national or regional domain spaces. It can automate data enrichment, correlation, and alerting workflows. MISP is not a detection engine because it relies on external feeds and data sources. Operating MISP demands security expertise and resource investment to maintain its infrastructure effectively [17].

### 2.12. IntelOwl

IntelOwl is a threat intelligence gathering and enrichment framework that allows users to query multiple data sources and analysis tools from a single interface. Developed as a community project, IntelOwl integrates with various public APIs, threat intelligence feeds, and local analysers to compile structured reports on artefacts like domain names, IP addresses, URLs, and file hashes. It can be configured to work in both on-demand mode and automated workflows, making it suitable for integration into ccTLD abuse monitoring systems.

IntelOwl is an extensible tool allowing users to integrate custom analysers and threat intelligence. IntelOwl has a modular architecture that provides flexibility, allowing ccTLD operators to customize the system to their intelligence needs by adding or removing analysers. It supports querying of domain reputations. Similarly to MISP, IntelOwl does not provide intrinsic detection capabilities and relies on the APIs and feeds configured within the system. Some analysers require paid API keys or subscriptions, which limits their utility in environments constrained by budget [18].

### 2.13. Spamhaus DROP (Don't Route or Peer) List

The Spamhaus Don't Route Or Peer (DROP) List is a compilation of IP address ranges that are under the control of cybercriminal organizations, as maintained by the Spamhaus Project. This list specifically includes IP blocks that are directly implicated in activities such as spam dissemination, malware distribution, phishing, and other forms of network abuse. By comparison to reactive blacklists that target individual domains or specific IP addresses, the DROP List adopts a preventive strategy by addressing entire IP ranges known to be associated with persistent malicious activity.

The primary objective of the DROP List is to enable network operators, security teams, and internet registries to preventively block both inbound and outbound traffic to and from these high-risk networks. This proactive approach reduces exposure to known malicious infrastructure. The DROP List is available for non-commercial use, and it is a useful tool for country code top-level domain (ccTLD) operators and other stakeholders. It can be integrated into network security appliances or used to verify domain registration requests [19].

### 2.14. Spamhaus Botnet Controller List (BCL)

The Spamhaus Botnet Controller List (BCL) is a DNS-based blacklist (DNSBL) that contains domain names and IP addresses associated with active botnet command-and-control (C2) servers. It is maintained by the Spamhaus Project and designed for real-time querying by mail servers, firewalls, and security systems to prevent communication with known C2 infrastructure. The list is dynamically updated because the research team identifies new botnet controllers through honeypots, sinkholes, and intelligence gathering.

BCL provides selected intelligence information on active botnet controllers, and as a consequence, it is a useful resource for ccTLD operators focused on detecting malicious domains tied to botnets. It uses a real-time DNS-based querying method, which is easy to integrate into existing infrastructure. Access to the BCL for commercial or high-volume querying requires a commercial agreement with Spamhaus, potentially limiting its use in budget-constrained environments. Also, the BCL is designed more for network defence and email filtering rather than for domain registration checking or bulk analysis, requiring additional integration effort for ccTLD-specific applications [20].

Table 1 provides a structured taxonomy of domain reputation and detection.

**Table 1.** Taxonomy of domain reputation and detection systems.

| System | Category | Data Source | Detection Approach | Learning Type. |
|---|---|---|---|---|
| Notos [1] | DNS-based Analysis | Passive DNS | Bayesian classification | Supervised |
| Kopis [6] | DNS-based Analysis | Authoritative DNS | Statistical profiling of NS behaviour | Unsupervised |
| Pleiades [8] | Graph/Community-based | Passive DNS (NXDOMAIN) | Graph clustering and community detection | Semi-supervised |
| DeepDGA [9] | Deep Learning-based | Domain names | LSTM recurrent neural networks | Supervised |
| Segugio [10] | Hybrid Behavioural | DNS + malware indicators | Behavioural correlation | Supervised |
| Phoenix [12] | Hybrid Behavioural | DNS traffic (temporal and spatial) | Temporal–spatial anomaly detection | Unsupervised |
| BotHunter [13] | Network Correlation | Network traffic | Multi-phase infection | Rule-based |
| PyDGADetector [14] | Lightweight ML-based | Domain lexical features | Logistic regression/Random Forest | Supervised |
| DAAR [15] | Threat Intelligence Aggregation | Aggregated abuse data | Statistical reporting and analysis | N/A |
| Abuse.ch/URLhaus [16] | Threat Intelligence/Aggregation | Malware and phishing feeds | Indicator correlation | N/A |
| MISP [17] | Threat Intelligence/Aggregation | Shared IoCs from partners | Threat correlation and enrichment | N/A |
| IntelOwl [18] | Threat Intelligence/Aggregation | Multiple APIs and feeds | Automated enrichment | N/A |
| Spamhaus DROP [19] | Blacklist/Preventive Filtering | IP address ranges linked to abuse | Preventive blocking of malicious networks | N/A |
| Spamhaus Botnet Controller List (BCL) [20] | Blacklist/Preventive Filtering | Active botnet C2 domains | Real-time DNSBL lookup | N/A |

Systems discussed in this study, preserving the order in which each tool or framework is presented in the Related Works section. The Category column classifies each approach by its methodological focus, ranging from DNS-based analysis and graph clustering to deep learning, hybrid behavioural models, and blacklist-based intelligence systems. System identifies the specific implementation or platform. Data Source specifies the primary input used for analysis, such as passive DNS data, network traces, lexical features, or aggregated threat feeds. Detection Approach summarizes the computational or statistical technique applied, and Learning Type indicates whether the system employs supervised, unsupervised, or heuristic logic.

This taxonomy highlights the progressive evolution of domain reputation research, illustrating how systems have transitioned from static blacklist [21] filtering to dynamic, data-driven, and adaptive models integrating machine learning and multi-source intelligence.

Table 2 synthesizes, for the systems that have reported accuracy, a comparative overview of domain detection and reputation systems accuracy. Some systems report accuracy while some do not report it online. Neural network-based, learning supported DeepDGA [9] to achieve an accuracy of 98.5%.

**Table 2.** Accuracy comparison of domain reputation and security algorithms.

| System | Accuracy (Reported) | Comment |
|---|---|---|
| Notos [1] | ~96% | On passive DNS data for known malicious domains |
| Kopis [6] | ~98% | Using authoritative DNS data, DGA detection |
| Pleiades [8] | ~97% | Focused on low-volume DGAs; community detection |
| BotHunter [13] | ~95% | Focused on network activity, not just domain activity |
| DeepDGA [9] | ~98.5% | Neural network-based, DGA detection |

Recent work has extended domain security and reputation systems toward privacy and cryptographic protections. Adaptive differential privacy techniques, which change privacy budgets or noise levels based on model sensitivity or query adaptivity, have been proposed to preserve utility while protecting sensitive registration and WHOIS-like data in federated and centralized workflows. These adaptive DP methods (e.g., ALDP-FL and related adaptive DP formulations) improve the accuracy/privacy trade-off for evolving data streams [22].

Concurrently, federated learning research has produced Byzantine-robust aggregation and communication schemes that resist poisoned or malicious client updates. Recent 2025 proposals (e.g., BRACE and FedCmp variants) demonstrate both robustness and communication efficiency under Byzantine attacks, making decentralized collaborations between registries and ISPs more feasible [23].

Hybrid solutions that combine cryptographic with statistical privacy mechanisms are becoming practical. Several 2025 studies propose layered pipelines in which sensitive features or high-risk parameters are encrypted or computed while aggregated outputs are noise-calibrated, thereby enabling joint analytics without exposing raw WHOIS or certificate data. These hybrid approaches trade higher computation cost for strong privacy guarantees and are directly relevant to domain reputation [24].

## 3. Domain Reputation Lists

The XGBoost algorithm is used for supervised learning problems, where the training data, with multiple features, is used to predict a target variable. Whitelists enumerate widely used and safe domain names. Blacklists contain domains or URLs associated with abuse and are used for checking risk scores, trigger warnings, or block resolution. Both list

types embody data sources to trust, which units of analysis to rank (domain, subdomain, or URL), how frequently to refresh entries, and how to mitigate sampling bias. In domain reputation systems, they serve complementary functions for reducing misclassification risk, and within this study, they are used to develop the domain dataset for training.

Alexa Top Sites is a list of domains with a good reputation score, and it was used within many academic and industrial studies, even after it was withdrawn by Amazon. Alexa computed rankings are derived from aggregated browsing data, combining estimates of average daily unique visitors and pageviews over a three-month window [25].

Researchers use whitelists derived from independent or multi-source lists. Cisco Umbrella is the most queried list after Alexa and provides a DNS-centric alternative based on passive DNS observations from its global resolver network [26]. Cloudflare Radar publishes ordered top 100 sets and bucketed domain-ranking datasets up to the top one million, updated on daily and weekly cadences from Cloudflare's network telemetry, which can be used as a whitelist before documented recency [27].

Blacklists encompass multiple, sometimes overlapping ecosystems and vary in their primary focus and granularity. The Spamhaus Domain Blocklist (DBL) is a reputation domain list used widely in email and web filtering. It aggregates evidence from diverse signals and is published as a DNSBL zone for increased performance querying. The Spamhaus Project SURBL provides near-real-time URI-based datasets originally optimized for detecting spam payload links and now used more broadly across messaging and web security. PhishTank provides a verified feed and an open API suitable for academic use [28]. OpenPhish produces research feeds with enriched metadata [29]. Malware-distribution tracking is supported by projects such as abuse.ch [16], URLhaus, which maintains and exposes API access for a repository of malicious URLs used for payload delivery. Google Safe Browsing maintains a threat list of unsafe websites, including phishing and malware URLs, exposed via public APIs and integrated into major browsers that operate as a blacklisting service [30].

Whitelists based on popularity reduce false positives but can admit compromised or parked domains and may overweight specific geographies or client populations. Blacklists reduce false negatives but can be incomplete or differ in granularity. Robust systems, therefore, combine multiple lists with multiple features: DNS behavioural features, WHOIS activity signals, and hosting and reputation.

## 4. Methods

The process of extracting, combining, and verifying domain name sets has direct relevance in cybersecurity and reputation analysis of internet domains. The goal of this approach is to create a labelled dataset that allows training a machine learning model capable of evaluating the benign or malicious nature of domains registered in the .ro national space. This objective is achieved through a sequence of clear methodological steps: extracting and filtering raw data, integrating it into a coherent structure, and verifying it.

The first step of the process aims to extract and normalize domains from two opposed sources in terms of reputation: on one hand, the Alexa Top 1 Million list [25], considered a trusted source for popular and generally legitimate domains, and on the other hand, several blacklists containing domains suspected or known for malicious behaviour.

The second step consists of combining the two resulting sets in a labelled CSV file. This file is structured so that each domain is associated with a binary label: 1 for domains from the blacklist, respectively, 0 for those from the whitelist. The merging process is precisely managed by using a dictionary where the key is the domain name and the value is the assigned label. Priority is given to the information from the blacklist: if a domain is present in both lists, it is considered unsafe and is labelled with 1, a hypothesis that reflects

a conservative approach, from a security perspective. The domains are then written to a final CSV file, sorted alphabetically, each line containing a domain followed by its label.

In the third step, the file resulting from the previous step is subjected to an additional validation check, which consists of determining the registration status information of each domain through WHOIS queries and nameserver information using nslookup function. This check is essential to guarantee that the Machine Learning model will be trained exclusively on active data, which reflects the current reality of the Romanian online space.

Queries are made through a TCP connection to the WHOIS server operated by .ro domain Registry, on port 4343. A minimalist client has been implemented, called WhoisClient, which avoids the use of external libraries and provides direct control over the data transmission and reception mode. Domains with "Status: NOT AVAILABLE" are considered active and kept in the final set, while those marked as "AVAILABLE" are excluded, in order to reduce noise or irrelevant data in the model training process.

An Important technical aspect of this stage Is the Introduction of a delay between WHOIS queries to avoid triggering protection measures from the Registry server, such as blocking the IP address or limiting access. Also, queries are managed in such a way that any corrupt or non-compliant responses are treated with caution, avoiding a complete halt to the process.

The entire processing flow is based exclusively on standard Python libraries (socket, csv, time), which gives the project a high degree of portability, predictability and ease of reproducing the results. The absence of external dependencies allows scripts to be run in restrictive environments and guarantees transparency of the implemented logic.

The presented methodology offers a robust and scalable solution for generating a labelled dataset, intended to serve as a basis for automatic domain reputation classification algorithms. By applying filtering, normalization, labelling and validation techniques, the document describes a complete application process, with direct potential for use in the development of fraud detection systems, malicious content blocking or advanced filtering in network infrastructures.

### *4.1. Data Sources and Objectives of the Model*

This study constructs a labelled corpus of Romanian domains by integrating two reputation sources: the Alexa Top 1 Million list, treated as a proxy for popular and generally legitimate domains, and multiple domain blacklists containing indicators of suspected or known malicious activity. The objective of the first stage is to obtain two candidate sets of .ro domains:

- **Whitelist: safe** domains originating from the Alexa list.
- **Blacklist: unsafe** domains originating from blacklist-type sources.

From the Alexa CSV file, only entries with the .ro and suffix are retained. Each domain is normalized to its registrable base by removing subdomains (e.g., mail.example.ro $\rightarrow$ example.ro) to prevent entity fragmentation and duplication. The resulting collection is converted to a set to enforce uniqueness and is sorted to ensure internal consistency.

- **Input**: input_files/top-1m.csv (CSV; e.g., 1, google.ro).
- **Intermediate Output**: output_files/alexa_ro_domains.txt (one domain per line).

Blacklists are provided as plain-text files, each containing one suspicious domain per line. Files are processed sequentially using the same normalization logic: www. Prefixes are removed; only domains ending in .ro are retained; and subdomains are collapsed to the registrable base. The consolidated set is deduplicated and alphabetically sorted, yielding a coherent list of .ro domains considered malicious.

- **Inputs**: input_files/blacklist_1.txt, input_files/blacklist_2.txt, input_files/blacklist_3.txt, input_files/blacklist_4.txt.
- ***Intermediate Output:*** output_files/blacklisted_ro_domains.txt (one domain per line).

### *4.2. Data Filtering and Normalization*

The filtering and normalization pipeline process ensures relevance, uniqueness, and consistent representation of entries.

Whitelist extraction (Alexa) includes the following stages:

- Read each row of the input CSV.
- Retain only domains ending in .ro.
- Normalize to the base form domain.ro (e.g., mail.google.ro → google.ro).
- Remove duplicates via set semantics.
- Sort alphabetically.
- Persist to output_files/alexa_ro_domains.txt.

Blacklist extraction comprises the next steps:

- Iterate through each blacklist file.
- Normalize by stripping the www. Prefix and retaining only .ro domains.
- If a subdomain is detected (e.g., abc.def.exemplu.ro), extract the base exemplu.ro.
- Remove duplicates and sort results.
- Persist to output_files/blacklisted_ro_domains.txt.

Both intermediate outputs are simple text lists with one domain per line and constitute the starting point for validation and feature extraction.

Within the dataset integration and labelling, the two normalized sets are merged into a single labelled CSV. A dictionary with a key domain name manages the merge, assigning a binary label to each entry: 1 for blacklisted (unsafe) and 0 for whitelisted (safe). A precedence rule is applied: if a domain appears in both sources, it is labelled 1 (unsafe). This prioritizes security by minimizing false negatives. The final artefact is an alphabetically sorted CSV with one record per line in the form domain, label.

### *4.3. WHOIS-Based Activity Validation and NSLOOKUP*

Within the next step domain activity is verified to ensure that the machine learning model is trained exclusively on currently active .ro domains. Registration status is determined through WHOIS queries to the domain Registry over a TCP connection to port **4343**. A minimalist client (WhoisClient) is implemented without external dependencies to maintain direct control over transmission and reception.

Server responses are interpreted as follows:

- "Status: NOT AVAILABLE"—the domain is **registered/active** → **retain**.
- "Status: AVAILABLE"—the domain is **unregistered/inactive** → **exclude**.

To avoid triggering server protections (e.g., IP blocking or rate limiting), a delay is introduced between queries. Fault handling is included to process corrupt or non-compliant responses cautiously, preventing complete pipeline interruption.

The entire workflow relies solely on standard Python libraries, socket, csv, and time, which enhances portability, transparency, and reproducibility. The absence of third-party dependencies enables execution in restrictive environments and guarantees full visibility into the implemented logic.

Some structured features are derived from registration data and SSL/TLS certificate metadata:

- domain_age: days since domain registration.
- expiration: days until domain expiry.

- has_dnssec: DNSSEC support (binary).
- registrar_id: numeric registrar identifier.
- has_ssl: presence of an SSL/TLS certificate (binary).
- ssl_certificate_issuer: issuing organization (categorical).
- ssl_certificate_valid_from: days since certificate issuance.
- ssl_certificate_validity: days until certificate expiry.
- tls_version: negotiated TLS protocol version (categorical).
- domain_length: number of characters in the domain name.

For the retrieval of domain information from domain Registry and SSL/TLS certificate metadata, Python functions were used. The following function retrieves information on domain age, expiration date, DNSSEC and registrar_id from the .ro domain Registry using the Registry's RestApi client:

```
FUNCTION get_domain_info(domain):
    Initialize API client
    Get domain information from API (async)
    IF no domain info returned:
          RETURN {
              domain\_age = None
              expiration = None
              has_dnssec = None
              registrar_id = None
          }
    today = current date
    domain_age = today - registration_date
    expiration = expiration_date - today
    RETURN {
          domain_age = number of days since registration
          expiration = number of days until expiration
          has_dnssec = whether DNSSEC records exist
          registrar_id = registrar ID from API
    }
```

The following code sample illustrates the steps used for obtaining SSL/TLS certificate metadata information:

```
FUNCTION get_tls_cert_info(domain):
    Create default SSL context
    TRY:
         Open SSL connection to domain on port 443 (with 5s timeout)
         Get SSL certificate
         Get TLS version
         Get server IP
         Close connection
     CATCH exception:
         TRY:
             Open non-SSL connection to domain on port 80 (with 5s timeout)
             Get server IP
             Close connection
         CATCH exception:
             IP = None

         RETURN {
```

```
            has_ssl = False
            ssl_certificate_issuer = None
            ssl_certificate_valid_from = None
            ssl_certificate_validity = None
            tls_version = None
            ip = IP
        }
    Extract issuer name from certificate
    Extract certificate validity dates (start and end)
    Initialize days_since_issued and days_until_expiry as None
    IF valid dates exist:
          TRY:
               Parse start and end dates
               now = current date
               days_since_issued = days since start date
               days_until_expiry = days until end date
          CATCH parsing error:
               Leave values as None
    RETURN {
        has_ssl = True
        ssl_certificate_issuer = issuer
        ssl_certificate_valid_from = days_since_issued
        ssl_certificate_validity = days_until_expiry
        tls_version = TLS version
        ip = IP
    }
```

Because SSL/TLS fields may sometimes be unavailable (e.g., due to network errors or server misconfiguration), modality-specific preprocessing was applied. For converting Boolean values to integers, True/False values for has_dnssec and has_ssl were converted to 1/0. Filling in missing values for categorical fields, in cases where TLS data was missing (due to handshake failure or HTTP usage), fields such as ssl_certificate_issuer and tls_version were filled in with the value 2.

In a domain-reputation context, the most pertinent outputs of nslookup are those that expose resolution state, infrastructure, and operational details. Resolution status and authority "authoritative" vs. "non-authoritative," NXDOMAIN, SERVFAIL, REFUSED) indicate whether a name exists and whether upstream validation or policy blocks are occurring. Nameserver (NS) records and any associated information highlight registrar/hosting alignment, the diversity and geography of the authoritative set, and misconfigurations. Mail exchange (MX) presence indicates whether the domain is positioned for messaging abuse or it is only a web asset. The SOA record (mname, rname, serial format, refresh/retry/expire, negative TTL) provides signals of zone stewardship and maintenance practices. These fields jointly furnish behavioural and configuration features that are predictive in supervised models reputation context.

The current feature set emphasizes registration (WHOIS) and SSL/TLS certificate attributes at inference time. Incorporating the nslookup attributes strengthens the model by capturing infrastructure dynamics that are orthogonal to registration and certificate evidence.

The final labelled dataset contained 15,245 benign (.ro domains from the Alexa whitelist) and 12,535 malicious (.ro domains from blacklists), corresponding to a distribution of 54.88% safe vs. 45.12% unsafe. This almost balanced ratio reduced the risk of severe

class imbalance. Stratified sampling was applied in both five-fold cross-validation and the 80/20 train–test split to preserve class proportions in each fold.

The methodology relies on a scalable labelled dataset suitable for training automatic domain reputation classifiers. Through rigorous filtering, normalization, conservative labelling, and WHOIS-based validation, the pipeline provides a high-quality foundation for applications such as fraud detection, malicious content blocking, and advanced network-level filtering.

The implementation relies on the XGBoost model training routine illustrated in Figure 2. The XGBClassifier class was used.

```python
import pandas as pd
from xgboost import XGBClassifier
import joblib

# Load and clean dataset
data = pd.read_csv("features_initial.csv")
data = data.drop(columns=["domain", "ip"])
data["has_dnssec"] = data["has_dnssec"].astype(int)
data["has_ssl"] = data["has_ssl"].astype(int)
data["ssl_certificate_issuer"] = data["ssl_certificate_issuer"].fillna("None")
data["tls_version"] = data["tls_version"].fillna("None")
data = pd.get_dummies(data, columns=["ssl_certificate_issuer", "tls_version"])

# Split features and target
X = data.drop(columns=["is_blacklisted"])
y = data["is_blacklisted"]

# Initialize and train model
model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
model.fit(X, y)

# Save model and feature column order
model.save_model("xgboost_blacklist_model.json")
joblib.dump(X.columns.tolist(), "feature_columns.pkl")
```

**Figure 2.** Applying the XGBOOST algorithm.

To assess the generalization ability of the model and ensure that it does not overfit the training data, 5-fold stratified cross-validation was used. This validation technique divides the dataset into five equal parts, preserving the proportion of blacklisted and safe domains in each subset.

In each of the five iterations, four folds were used to train the model, and one was reserved for performance testing. The process was repeated so that each subset served as the test set only once. The average accuracy obtained on the five folds was then calculated.

This method exposes the model to varied subsets of the data, thereby increasing its robustness against overfitting and ensuring consistent performance on previously unseen examples. It is particularly useful in security-focused datasets, where the diversity of malicious patterns can vary.

After training, the model was saved to disc in XGBoost's native JSON format. This allowed for easy reloading of the model during inference, especially in production scenarios where real-time or batch classification of domain records is required.

A clearer conceptual framework is presented in Table 3. A taxonomy of the methods employed for constructing, validating, and operationalizing the .ro domain reputation

model. The taxonomy organizes the workflow into five methodological layers, each defined by its objective, data type, technique, and output artefact.

**Table 3.** Methods taxonomy.

| Layer | Objective | Data Type/Source | Techniques Applied | Output/Artefact |
| --- | --- | --- | --- | --- |
| 1. Data Acquisition | Collect representative .ro domains from safe and unsafe sources | Whitelists (Alexa, Cisco Umbrella, Cloudflare Radar); Blacklists (Spamhaus DBL, SURBL, PhishTank, OpenPhish, URLhaus, Google Safe Browsing) | Parsing, filtering by TLD, deduplication | Raw .whitelist and .blacklist domain sets |
| 2. Data Integration & Labelling | Merge sources into a unified corpus with binary reputation labels | Combined whitelist and blacklist sets | Rule-based labelling (blacklist precedence), CSV merge | Labelled dataset (domain, label) |
| 3. Validation & Feature Extraction | Verify active registration and enrich with infrastructure features | WHOIS data, DNS lookup, SSL/TLS metadata | WHOIS queries (TCP 4343), nslookup, certificate parsing | Verified active domains with structured features |
| 4. Model Training & Evaluation | Build and assess a predictive classifier for malicious domain detection | Labelled dataset (27,780 records) | XGBoost training, 5-fold stratified CV, metrics (accuracy, precision, recall, F1) | Trained and validated XGBoost model |
| 5. Deployment & Inference | Operationalize real-time domain scoring for .ccTLD use | Flask API, Registry API, async data retrieval | Asynchronous feature collection, model inference, risk score presentation | Lightweight Flask web service (DoReSi) |

## 5. Evaluation of the Model

Before fitting the final model, the script conducts a diagnostic five-fold stratified cross-validation using accuracy as the scoring function. Stratification preserves class proportions within each fold so that the estimates reflect performance under the observed class distribution. The resulting accuracies, along with their mean and standard deviation, provide a quick check for issues such as degenerate features or label misalignment. Importantly, cross-validation failures (for instance, due to pathological data) are caught and logged rather than halting the pipeline, so that users can obtain a final train/test evaluation when possible.

Subsequently, the data are partitioned into training and test subsets using an 80/20 split with stratification and a fixed random seed. The script reports standard point metrics on the test set. Accuracy, presented in Figure 3 is printed as a single scalar, and a full classification report summarizes precision, recall (sensitivity), F1-score.

```
[Eval] Test accuracy: 0.9939
[Eval] Classification report:
              precision    recall  f1-score   support

           0     1.0000    0.9895    0.9947       759
           1     0.9857    1.0000    0.9928       552

    accuracy                         0.9939      1311
   macro avg     0.9929    0.9947    0.9938      1311
weighted avg     0.9940    0.9939    0.9939      1311
```

**Figure 3.** XGBoost model results.

To make error modes concrete, a confusion matrix is computed over the test predictions and printed in the conventional layout which makes visible the trade-offs between sensitivity and specificity implied by the chosen threshold. In practical use, the same matrix can be rendered graphically with scikit-learn's *ConfusionMatrixDisplay.from_predictions*, which computes the matrix and provides a standardized visualization, as illustrated in Figure 4.

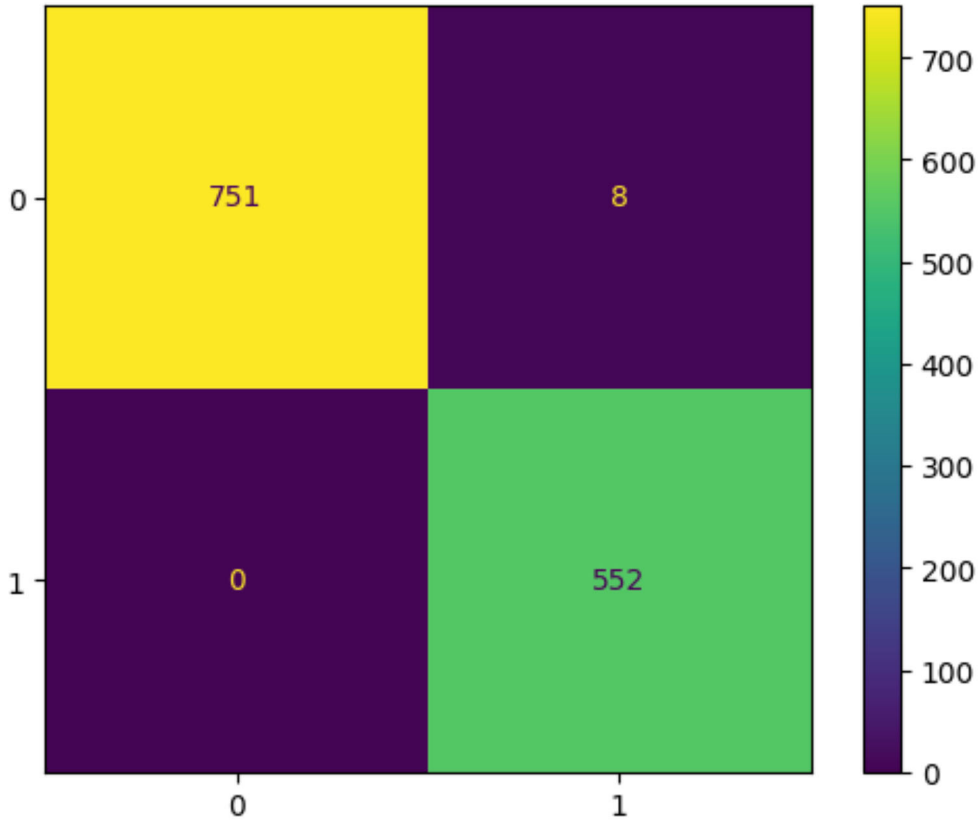


**Figure 4.** Confusion Matrix for XGBoost.

Let TP, TN, FP, FN denote the entries of the confusion matrix computed on the held-out test set. The following metrics are relevant:

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP})$$

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN})$$

$$\text{F1} = 2\ \text{Precision}\ \text{Recall}/(\text{Precision} + \text{Recall})$$

For threshold-based analysis, predicted labels are defined by $\hat{y} = 1\{ \hat{p} \geq \theta \}$ with $\theta \in [0, 1]$, yielding:

$$TPR = TP/(TP + FN)$$

$$FPR = FP/(FP + TN)$$

Because probability outputs from tree-ensemble models (e.g., gradient boosting) can be incorrectly calibrated, $\hat{p}$ should not be interpreted as absolute risk without calibration.

The model was first validated using 15-fold cross-validation, which yielded an average accuracy of 0.9992 ($\pm$0.0014), indicating stable performance across folds. After fitting on the training data, the final evaluation on the held-out test set (n = 1311) reported an overall accuracy of 0.9939.

For benign domains (class 0, n = 759), the model achieved precision = 1.0000, recall = 0.9895, and F1 = 0.9947. For malicious domains (class 1, n = 552), the model reached precision = 0.9857, recall = 1.0000, and F1 = 0.9928. Macro-averaged metrics were precision = 0.9929, recall = 0.9947, and F1 = 0.9938. Weighted averages were consistent (0.994). These results confirm balanced performance across both classes, with very few false negatives and a small number of false positives.

These complementary metrics confirm that the reported performance is not solely due to accuracy under near-balanced classes. The model achieves perfect precision for benign domains and perfect recall for malicious domains, meaning it rarely mislabels clean domains as malicious and almost never misses a malicious domain. The very high F1-scores (>0.99 for both classes) demonstrate that the performance is consistent across precision–recall trade-offs. Together with AUROC = 0.999 and AUPRC = 0.999, these results provide evidence that the system achieves genuine discriminative power rather than benefiting from dataset artefacts.

In the architecture of a domain scoring system, the main server module plays a crucial role, acting as a bridge between the user interface and the underlying analytics engine powered by machine learning. Essentially, this component is a Flask web server that allows users to interactively submit domain names and receive real-time scores based on pre-trained models and live feature collection.

Flask is a web micro-framework written in Python that provides the essential tools for developing web applications simply and flexibly. In this case, the server exposes two main routes:

- The root route renders a basic HTML form where users enter domain names.
- A scoring route processes the submitted domains, collects relevant features, and provides a predictive score.

While Flask allows for rapid development and clear routing logic, its architecture also supports integration with asynchronous operations and machine learning-based workflows. This is essential, especially since domain evaluation involves network operations such as querying DNS data and extracting information from SSL certificates.

The evaluation process begins with the collection of domain-specific features. These are obtained through two asynchronous routines executed concurrently:

- A query to the registry API, which extracts attributes such as the domain_age, expiration, has_dnssec, registrar_id, has_ssl, ssl_certificate_issuer, ssl_certificate_valid_from, ssl_certificate_validity, tls_version, domain_length;
- A query using nslookup for obtaining information concerning authoritative/non-authoritative, NXDOMAIN, SERVFAIL, REFUSED, NS records, Mail exchange (MX)

Both routines are executed in parallel using Python's asyncio.gather, which reduces overall latency and allows the application to scale efficiently. After collecting the feature

data, an additional synthetic feature is computed locally: the domain length. This provides a lightweight but relevant heuristic in domain classification tasks.

The most important component of the backend infrastructure is the integration of a trained XGBoost model. This model, previously trained on historical data labelled with a binary variable indicating whether a domain is blacklisted, is serialized and saved to disc. At application startup, the model is loaded once using the load_model method in XGBoost, and the corresponding list of feature columns is restored using joblib. This ensures consistency between the training and inference stages.

When a domain is evaluated, the collected features are transformed into a single-row DataFrame. One-hot encoding is applied to categorical fields such as SSL certificate issuer and TLS version to align with the training schema. This dynamic alignment is essential for robustness in production, especially in contexts where features may vary from one instance to another.

After preprocessing, the features are fed into the XGBoost model. The prediction is binary: 1 for blacklisted domains and 0 for clean ones. The label is returned along with the raw feature values, providing transparency and interpretability in the web interface.

The HTML interface is intentionally minimalist, using inline templates to avoid dependency on external files. This choice is consistent with the purpose of the application, which functions as a backend utility, and not as a complete end-user product. When a domain is submitted, the processing flow is as follows:

1. The evaluation route reads the domain from the form.
2. Asynchronous feature extraction is performed.
3. The model evaluates the domain.
4. The results are displayed as an unordered HTML list, with the prediction and raw feature values.

The application can be used in both test scenarios and light production environments, such as internal assessments performed by a registry. The server module embodies the convergence of asynchronous programming, machine learning-based inference, and web application design. It leverages the capabilities provided by Flask as minimalist routing, asyncio for efficient I/O-dependent operations, and a trained XGBoost model for predictive analytics. The overall architecture is modular, extensible, and adequate for integration into a larger domain monitoring or threat detection system.

The overall design reflects sound architectural decisions, decoupling data collection from the inference phase, ensuring robustness to missing values, and providing fast feedback to end users. All of this is performed with a minimal number of dependencies and a focus on maintainability.

## 6. Discussion

The accuracy of the designed framework has a substantial performance 0.993 (after pre-processing, filtering, normalization, and WHOIS-based information). The result of the proposed domain reputation system (DoReSi) outperforms the results of previous systems illustrated in "Table 1. Accuracy comparison of domain reputation and security algorithms".

The final dataset consisted of 15,245 benign domains (54.88%) and 12,535 malicious domains (45.12%), a near-balanced distribution. Because of this, no oversampling or synthetic sample generation was required. Instead, stratified sampling and XGBoost's internal class weighting ensured sensitivity toward the minority class, so the reported accuracy and F1 (0.993) reflect genuine discriminative power rather than artefacts of imbalance. Future work will examine cost-sensitive thresholding and resampling under stronger imbalance conditions.

Table 4 illustrates the contribution of each feature to the model's predictions. SSL/TLS-related features (issuer, validity, version, and presence of certificates) and registry-based attributes (domain age, registrar ID, expiration) were most influential in distinguishing benign from malicious domains.

**Table 4.** SHAP-based feature importance ranking for the XGBoost classifier.

| Feature | Importance |
|---|---|
| ssl_certificate_issuer | 0.507085 |
| domain_age | 0.111686 |
| tls_version | 0.096477 |
| ssl_certificate_validity | 0.053947 |
| registrar_id | 0.049856 |
| ssl_certificate_valid_from | 0.045324 |
| expiration | 0.040915 |
| has_dnssec | 0.038311 |
| domain_length | 0.037436 |
| has_ssl | 0.018963 |

SHAP analysis confirmed that the most influential predictor was the SSL certificate issuer (importance 0.51), followed by domain age (0.11) and TLS version (0.10). Other features with moderate contributions included SSL certificate validity, registrar ID, certificate issuance date, and expiration period. DNSSEC support and domain length also provided useful signals, though at lower importance. These findings highlight that infrastructure and cryptographic attributes dominate the model's decision-making. Recently registered or inconsistently certified domains are more likely to be malicious.

The SHAP based feature importance analysis indicated that *ssl_certificate_issuer* contributed most significantly to the model's predictions, followed by *domain_age* and *tls_version*. In this context, the risk of multicollinearity is limited because *domain_age* originates from registry WHOIS data, whereas *ssl_certificate_issuer* and *tls_version* are derived from SSL/TLS certificate and DNS-based sources, reflecting distinct operational layers. Nevertheless, to ensure interpretability and confirm that the observed SHAP importances represent independent predictive effects, future work will include correlation matrix analysis and variance inflation factor (VIF) diagnostics. These analyses will help quantify inter-feature dependencies and verify that the model's interpretive signals are not influenced by residual feature overlap.

Figure 5 synthesizes, for the systems that have reported accuracy, from the list from chapter 2. a comparative overview of domain detection and reputation systems accuracy. Early systems such as Notos [1] and Kopis [6] demonstrated high accuracy (96–98%) by leveraging passive and authoritative DNS data, respectively, to identify malicious domains and detect domain generation algorithms (DGAs). Pleiades [8], refined the model by using community graph analysis to address DGA activity, achieving a precision of around 97%. On the other hand, BotHunter [13] shifted the emphasis from domain information to network activity correlation, leading to variable detection rates. Advances in machine learning supported DeepDGA [9] in applying neural networks to achieve an accuracy of approximately 98.5%.

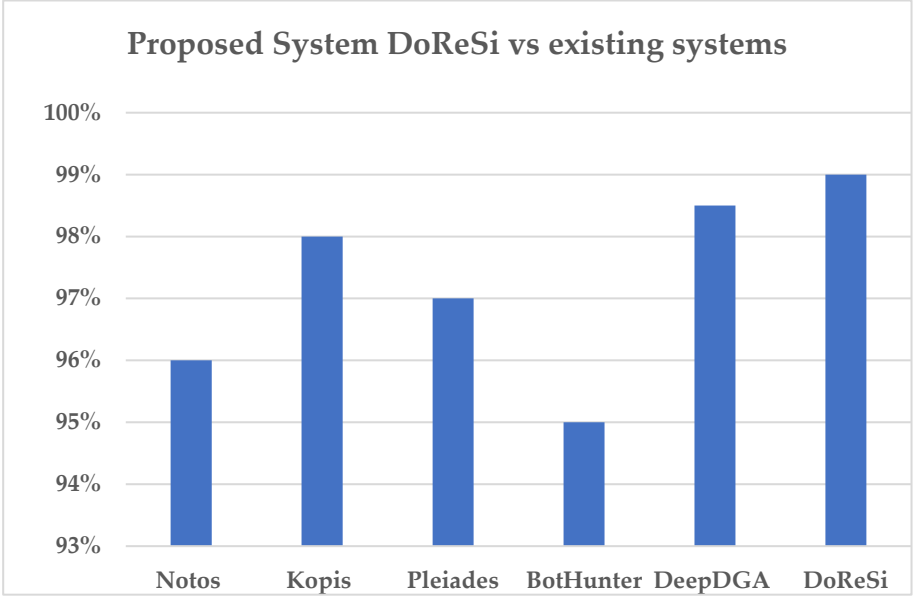


**Figure 5.** Comparison between proposed system and existing systems.

While this magnitude of improvement is encouraging, it warrants careful interpretation. First, accuracy can be misleading under class imbalance complementary metrics (precision, recall, F1, AUROC, AUPRC), and calibration analyses should accompany accuracy to ensure that improvements reflect genuine discriminative power rather than distributional artefacts. Second, the normalization to registrable base domains and the removal of inactive entries likely reduced label noise and train–test contamination.

Accordingly, variance estimates (e.g., 95% CIs via bootstrap) and significance tests (e.g., McNemar's test on paired predictions, DeLong's test for AUROC) are recommended to quantify the robustness of the gain.

From a data perspective, the conservative labelling rule (blacklist precedence) reduces false negatives but may increase false positives. The accuracy obtained (0.993) raises the possibility of residual leakage (e.g., temporally inconsistent labels, overlapping sources) or overfitting to list-specific artefacts. To mitigate these risks, future developments should adopt grouped cross-validation to prevent subdomain leakage, and external validation on held-out .ro domains and, where feasible, other ccTLDs to assess generalizability.

To translate these findings into practice, the domain reputation interface will be further developed, relying on the validated model [29,30]. The interface will expose a probabilistic risk score with confidence intervals, categorical risk bands (e.g., low/medium/high) tied to pre-registered thresholds, explanations of the top contributory features for each prediction (e.g., SHAP-based summaries), and provenance metadata (snapshot time, WHOIS, and certificate evidence). The service layer will incorporate rate-limited WHOIS queries, short-TTL caching, and a degradation path that falls back to registration and lexical features when SSL/TLS data are unavailable. Comprehensive logging, audit trails, and an appeal workflow will support accountability and continuous improvement.

The proposed domain reputation system assumes an adversary who seeks to register and operate malicious domains under the .ro ccTLD for activities such as phishing, malware distribution, or botnet command-and-control. The adversary may attempt to evade detection by mimicking characteristics of benign domains (e.g., adopting popular registrar providers, using valid SSL/TLS certificates, or aligning DNS configurations with legitimate patterns). The defender, in this case the registry or a security monitoring entity, has access to registry WHOIS data, DNS resolution traces, and SSL/TLS certificate metadata. The

system does not assume access to full passive DNS telemetry or endpoint traffic, making it lightweight but reliant on registration- and infrastructure-based features.

Adversaries may attempt several evasion strategies:

- Certificate manipulation using free and widely trusted certificate authorities to reduce the discriminative power of SSL certificate issuer as a feature.
- Domain ageing and re-registration for keeping domains dormant to bypass "new domain" heuristics.
- Adopting DNSSEC, long validity SSL certificates, or popular registrars to blend in with benign domains.
- Fast-flux hosting or domain churn by rapidly rotating DNS records or registering disposable domains to avoid consistent labelling.
- Adversarial perturbations by registering domains with lexical patterns that resemble benign domains (typosquatting or homoglyph attacks).

To evaluate scalability, the Flask API used to expose the model predictions under varying concurrent request loads was benchmarked. With feature collection mocked, the service sustained hundreds of requests per second with sub-millisecond median latency, confirming that inference is lightweight. When WHOIS, DNSSEC, and SSL/TLS lookups were enabled, latency increased due to external I/O but remained stable and error-free, demonstrating that the service can handle practical loads. These results indicate that the system can be deployed as a near real-time scoring API, with caching and asynchronous lookups further mitigating latency in production.

For deployment, the proposed domain reputation system (DoReSi) should be accompanied by supporting infrastructure to ensure robustness and easy maintenance. Logging and monitoring are critical assets. Prediction requests, feature values, and model outputs should be logged with timestamps, while dashboards can track latency, error rates, and class distribution in real time. Second, retraining frequency will be aligned with the dynamics of the threat landscape. In practice, monthly or quarterly retraining on updated WHOIS, DNS, and certificate data would help counteract concept drift as adversaries evolve their tactics. Third, model versioning and rollback are recommended to ensure that updates can be audited and reverted if performance regressions occur. Finally, security and compliance controls such as rate-limiting API access, anonymizing sensitive WHOIS fields, and maintaining audit trails will support safe operation.

While the proposed framework demonstrates high accuracy, it is important to acknowledge the trade-offs between alternative methodological choices. From an accuracy perspective, the integration of registry, WHOIS, and SSL/TLS features significantly boosts performance, but this comes at the cost of increased **data access requirements** and potential **privacy implications** (e.g., reliance on WHOIS data that may contain personal identifiers). Computationally, XGBoost provides strong discriminative power with efficient training and inference times, but communication cost can increase when live feature collection (WHOIS queries, SSL/TLS handshakes) is performed at inference time. By contrast, lighter lexical approaches (domain length, character entropy) incur minimal I/O cost and no privacy risk but generally achieve lower accuracy and poorer generalization. Similarly, resampling techniques or deep learning models may further improve recall yet would demand higher computational resources and longer retraining cycles. The present design therefore balances accuracy and operational feasibility by combining high-signal infrastructure features with asynchronous collection and caching, while avoiding excessive dependence on sensitive or high-latency data sources. Future extensions may more systematically benchmark these trade-offs under different operational constraints (e.g., low-latency registry environments, strict privacy policies, or large-scale batch scoring).

From a governance and compliance perspective, privacy-preserving machine learning techniques such as Federated Learning (FL) are directly relevant to the .ro domain reputation context. The .ro ccTLD registry manages sensitive WHOIS data that can include personal identifiers of registrants; therefore, GDPR imposes strict constraints on centralized data collection and processing. Integrating FL would allow multiple registries or security partners (e.g., national CERTs or ISPs) to collaboratively train reputation models without sharing raw WHOIS or DNS data, instead exchanging encrypted model updates. This decentralized approach maintains analytical power while ensuring legal compliance with GDPR principles of data minimization and purpose limitation.

Similarly, applying adaptive differential privacy during training can protect registrant-level attributes by introducing calibrated noise to gradients or aggregated statistics, ensuring that no individual domain record can be reverse engineered from the model. In the DoReSi framework, this could be implemented at the model retraining stage, for example, when updating the XGBoost model with new .ro domain data, to enable safe periodic learning without violating privacy obligations.

These methods thus strengthen the practical deployment of the .ro reputation system. FL enables cross-organizational learning between registry and network operators, while differential privacy and cryptographic safeguards maintain compliance with European data protection laws. Together, they provide a pathway toward federated, privacy-preserving threat intelligence across ccTLD environments, aligning the DoReSi prototype with both operational feasibility and GDPR requirements.

This work has limitations. List supervision can contain outdated or disputed labels, and registry semantics may evolve, affecting labelling and feature availability. The .ro focus constrains external validity, and replication across other ccTLDs and gTLDs is needed to check accuracy of the model across multiple datasets. The performance reported was obtained using the current data repository.

Although the final dataset used for model training exhibited an almost balanced distribution (54.88% benign vs. 45.12% malicious), the initial raw data extracted from whitelists and blacklists presented a more uneven distribution. Specifically, blacklist sources contained substantially fewer .ro domains compared to global lists, while whitelist entries were overrepresented due to the broader coverage of Alexa and Umbrella datasets. To mitigate this initial imbalance, a multi-stage normalization and filtering process was applied—restricting all entries to active .ro domains, collapsing subdomains to their registrable base, and enforcing a blacklist-precedence labelling rule. These steps effectively reduced sampling bias and produced a dataset suitable for stratified cross-validation. Consequently, no oversampling or synthetic sample generation was required, and XGBoost's internal class-weighting further ensured sensitivity to minority samples. The near-balanced final distribution reflects the outcome of this controlled data curation rather than the characteristics of the raw input sources.

One limitation of this study concerns the absence of direct comparative experiments with previous systems such as Notos and Kopis. This limitation arises from fundamental differences in data availability and system architecture. The proposed framework was designed and validated on a dedicated .ro ccTLD dataset constructed from WHOIS, DNS, and SSL/TLS data sources, combined with curated blacklist and whitelist information. In contrast, Notos and Kopis operate on entirely different data foundations—large-scale passive DNS telemetry and authoritative DNS query streams—which are proprietary and not publicly accessible. As a result, these systems cannot be re-executed or fairly benchmarked on the same dataset used in this work. The comparative analysis presented therefore relies on published performance metrics from the literature, serving as a contextual rather

than experimental benchmark. This distinction reflects the practical constraints of data compatibility rather than methodological omission.

While the reported performance metrics (accuracy = 0.993, precision = 0.993, recall = 0.994, F1 = 0.993) demonstrate strong discriminative power, these results are presented as point estimates without associated confidence intervals or statistical significance tests. This limitation may affect the interpretation of the model's robustness, as high accuracy could partially result from the dataset size, sampling variability, or specific data-split contingencies. To address this, future work will include statistical resampling techniques such as bootstrapped confidence intervals and formal tests of significance (e.g., McNemar's test for paired predictions and DeLong's test for AUROC). These methods will enable the quantification of uncertainty around the reported metrics and provide stronger evidence that the observed performance reflects genuine model generalization rather than dataset-specific effects. Incorporating such statistical validation will reinforce the reliability and reproducibility of the evaluation results.

A limitation of this study is the lack of direct comparative experiments with prior systems such as Notos, Kopis, or Pleiades. These approaches depend on proprietary passive DNS feeds that were not available in our experimental setting, making reimplementation on the .ro dataset not feasible. Accordingly, the comparison presented in Figure 5. relies on reported values from the literature, which provides a broad sense of progress but does not guarantee strict comparability under identical conditions. To address this, future work will explore benchmarking DoReSi against open-source baselines (e.g., lexical DGA detectors, deep learning models trained on public DNS datasets) using consistent metrics. Such evaluations would complement the current analysis and provide a stronger empirical basis for comparing detection performance across approaches.

Planned comparative modelling between Random Forest (RF) and XGBoost (XGB) will clarify whether the observed results persist across learners with different variance profiles and missing-data. A nested cross-validation design will be implemented with identical preprocessing, consistent folds, and cost-sensitive thresholding. For RF, key hyperparameters will include the number of trees and feature subsampling; for XGB, learning rate, tree depth, and column/row subsampling will be tuned on logarithmic grids.

Because XGB handles missing values, while RF benefits from explicit imputation, ablations will be reported with and without SSL/TLS features to quantify modality sensitivity. Model calibration and decision-curve analysis will be used to align operating points with security objectives (e.g., high-recall regimes).

## 7. Conclusions

Incidents involving malicious code dissemination through vulnerabilities in websites have increased in recent years. Consequently, the need to rigorously assess website security risk has intensified. In practice, such risk is typically evaluated using a combination of static and dynamic analysis [31–33].

The study describes a machine-learning system that classifies domain names as safe or malicious using registry data and DNS/HTTP/SSL/WHOIS/NSLOOKUP information. The repository provides training scripts, a REST API, and an example dataset of approximately 25,000 entries. A XGBoost model was trained on the merged list data reports with a cross-validated accuracy and F1 $\approx$ 0.993, with top importances dominated by TLD indicators, and whois/network features. The result was obtained for the .ro dataset and might be optimistic for live traffic due to sampling bias and potential feature leakage from processing domain lists. The feature engineering, learning objective, validation, and calibration were formalized and used for the deployment plan [32].

Future work will therefore prioritize longitudinal analyses, semi-supervised learning to reduce reliance on static lists, integration of signals (passive DNS, HTTP banners) to harden the model against missing SSL/TLS data, and a Random Forest vs. XGBoost comparison. These steps will test the preprocessing gains and support a domain reputation scoring system suitable for operational deployment.

The datasets presented in this article are not readily available because some data are subject to privacy policy of .ro ccTLD.

**Author Contributions:** Conceptualization, C.I.R.; methodology, C.I.R.; software, C.I.R.; validation, C.I.R., A.A. and I.Ș.S.; formal analysis, C.I.R.; investigation, C.I.R.; resources, C.I.R.; data curation, C.I.R.; writing—original draft preparation, C.I.R.; writing—review and editing, C.I.R., A.A. (Sections 1 and 2) and I.Ș.S. (Section 3); visualization, C.I.R.; supervision, A.A. and I.Ș.S.; project administration, A.A.; funding acquisition, not applicable. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** Data is unavailable due to privacy restrictions. The datasets used in this article are not readily available because the data includes .ro domain related information which are not publicly available due to Registry privacy policy.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| ccTLD | country-code Top-Level Domain |
| TLD | Top-Level Domain |
| DNS | Domain Name System |
| DNSSEC | Domain Name System Security Extensions |
| WHOIS | Registration data lookup protocol (WHOIS) |
| TLS | Transport Layer Security |
| SSL | Secure Sockets Layer |
| IP | Internet Protocol |
| ASN | Autonomous System Number |
| NS | Name Server (DNS NS record) |
| MX | Mail Exchanger (DNS MX record) |
| RDNS | Recursive DNS resolvers |
| NXDOMAIN | Non-Existent Domain (DNS response code) |
| SERVFAIL | Server Failure (DNS response code) |
| TTL | Time To Live |
| API | Application Programming Interface |
| REST | Representational State Transfer |
| JSON | JavaScript Object Notation |
| URL | Uniform Resource Locator |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| CSV | Comma-Separated Values |
| DGA | Domain Generation Algorithm |
| C&C (C2) | Command and Control |
| MISP | Malware Information Sharing Platform & Threat Sharing |
| DAAR | Domain Abuse Activity Reporting |
| RIPE NCC | Réseaux IP Européens Network Coordination Centre |
| ICANN | Internet Corporation for Assigned Names and Numbers |
| DNSBL | DNS-based Block List |

| | |
|---|---|
| DBL | Domain Block List (Spamhaus) |
| SURBL | Spam URI Realtime Blocklists |
| DROP | Don't Route Or Peer (Spamhaus list) |
| BCL | Botnet Controller List (Spamhaus) |
| FQDN | Fully Qualified Domain Name |
| XGBoost | Extreme Gradient Boosting |
| ROC | Receiver Operating Characteristic |
| PR | Precision–Recall |
| AUROC | Area Under the ROC Curve |
| AUPRC | Area Under the Precision–Recall Curve |
| TP | True Positive |
| FP | False Positive |
| TN | True Negative |
| FN | False Negative |
| TPR | True Positive Rate (Recall) |
| FPR | False Positive Rate |
| SHAP | SHapley Additive exPlanations |
| CV | Cross-Validation |
| ISP | Internet Service Provider |
| IDS | Intrusion Detection System |
| TCP | Transmission Control Protocol |
| RNN | Recurrent Neural Network |
| LSTM | Long Short-Term Memory |
| NSLOOKUP | Name Server Lookup utility |

## References

1. Antonakakis, M.; Perdisci, R.; Dagon, D.; Lee, W.; Feamster, N. Building a Dynamic Reputation System for DNS. In Proceedings of the USENIX Security Symposium, Washington, DC, USA, 11–13 August 2010.
2. Bilge, L.; Kirda, E.; Kruegel, C.; Balduzzi, M. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In Proceedings of the 18th Annual Network & Distributed System Security Symposium (NDSS 2011) Proceedings, San Diego, CA, USA, 6–9 February 2011; NDSS Symposium, DBLP. Internet Society: Reston, VA, USA, 2011; p. 17.
3. Dumitrache, M.; Săcală, I.Ș.; Rotună, C.I.; Gheorghiță, A.; Sandu, I.; Smadă, D. Developing an intelligent security monitoring platform for Internet domains: A practical implementation approach. *Stud. Inform. Control* **2025**, *34*, 75–84. [CrossRef]
4. Rotună, C.I.; Gheorghiță, A.; Sandu, I.; Dumitrache, M.; Udroiu, M.; Smadă, D. A generic architecture for building a domain name reputation system. *Stud. Inform. Control* **2023**, *32*, 39–49. [CrossRef]
5. RIPE NCC. RIPE Atlas and DNS Monitoring for Resilience. 2021. Available online: https://www.ripe.net/ (accessed on 27 August 2025).
6. Antonakakis, M.; Perdisci, R.; Lee, W.; Vasiloglou, I.N.; Damballa, Inc. Method and System for Detecting DGA-Based Malware. U.S. Patent 9,922,190, 20 March 2018.
7. Bilge, L.; Balzarotti, D.; Robertson, W.K.; Kirda, E.; Kruegel, C. Disclosure: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis. In Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC 2012), Orlando, FL, USA, 3–7 December 2012; pp. 129–138.
8. Antonakakis, M.; Perdisci, R.; Nadji, Y.; Vasiloglou, N.; Abu-Nimeh, S.; Lee, W. Detecting and Tracking the Rise of DGA-Based Malware. *USENIX* **2012**, *37*, 16–27.
9. Anderson, H.S.; Woodbridge, J.; Filar, B. DeepDGA: Adversarially-tuned domain generation and detection. *arXiv* **2016**, arXiv:1610.01969. Available online: https://arxiv.org/abs/1610.01969 (accessed on 27 August 2025).
10. Rahbarinia, B.; Perdisci, R.; Antonakakis, M. Segugio: Efficient Behaviour-Based Tracking of Malware-Control Domains in Large ISP Networks. In Proceedings of the 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, 22–25 June 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 403–414.
11. Galloway, T.; Karakolios, K.; Ma, Z.; Perdisci, R.; Antonakakis, M.; Keromytis, A.D. Practical Attacks Against DNS Reputation Systems. In Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2024; IEEE Computer Society: Los Alamitos, CA, USA, 2024.

12. Schiavoni, S.; Maggi, F.; Cavallaro, L.; Zanero, S. Phoenix: DGA-Based Botnet Tracking and Intelligence. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2014)*; Dietrich, S., Ed.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; Volume 8550, pp. 192–211. [CrossRef]
13. Gu, G.; Porras, P.A.; Yegneswaran, V.; Fong, M.W.; Lee, W. BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. In Proceedings of the 16th USENIX Security Symposium (USENIX Security '07), Boston, MA, USA, 6–10 August 2007; USENIX Association: Berkeley, CA, USA, 2007; pp. 167–182.
14. Bader, J. Detecting Domain Generation Algorithms with Machine Learning; GitHub Repository. 2017. Available online: https://github.com/baderj/domain_generation_algorithms (accessed on 2 July 2025).
15. ICANN OCTO. Domain Abuse Activity Reporting (DAAR). 2017. Available online: https://www.icann.org/octo-ssr/daar (accessed on 2 July 2025).
16. Abuse.ch. URLhaus—Malware URL Exchange. Available online: https://urlhaus.abuse.ch/ (accessed on 27 August 2025).
17. Wagner, C.; Dulaunoy, A.; Iklody, A.; Wagener, G. MISP: The Open Source Threat Intelligence Platform. MISP Project. 2016. Available online: https://www.misp-project.org/ (accessed on 5 July 2025).
18. IntelOwl Community. IntelOwl: Open Source Threat Intelligence Platform; GitHub Project. 2020. Available online: https://github.com/intelowlproject/IntelOwl (accessed on 5 July 2025).
19. Spamhaus Project. Spamhaus DROP (Don't Route Or Peer) List. 2003–2025. Available online: https://www.spamhaus.org/drop/ (accessed on 7 July 2025).
20. Spamhaus Project. Botnet Controller List (BCL). 2014–2025. Available online: https://www.spamhaus.org/bcl/ (accessed on 8 July 2025).
21. Saxe, J.; Berlin, K. expose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. *arXiv* **2017**, arXiv:1702.08568. Available online: https://arxiv.org/abs/1702.08568 (accessed on 27 August 2025).
22. Xu, Y.; Zhang, L.; Li, Y.; Wang, J. Adaptive Local Differential Privacy for Federated Learning. *Sci. Rep.* **2025**, *15*, 12575. [CrossRef] [PubMed]
23. Xie, Z.; Zhang, S.; Zhang, C.; Liu, Y.; He, Z. BRACE: Byzantine-Robust All-Reduce Communication in Federated Learning. *arXiv* **2025**, arXiv:2501.17392. Available online: https://arxiv.org/abs/2501.17392 (accessed on 30 September 2025).
24. Gong, Y.; Wang, R.; Chen, X.; Huang, Z. Hybrid Privacy-Preserving Machine Learning with Homomorphic Encryption and Differential Privacy. *arXiv* **2025**, arXiv:2501.12911. Available online: https://arxiv.org/abs/2501.12911 (accessed on 30 September 2025).
25. Kaggle. Alexa Top 1 Million Sites. 2025. Available online: https://www.kaggle.com/datasets/cheedcheed/top1m (accessed on 27 August 2025).
26. Cisco. Umbrella Popularity List. Available online: https://umbrella-static.s3-us-west-1.amazonaws.com/ (accessed on 27 August 2025).
27. Cloudflare. Domain Rankings—Cloudflare Radar. Available online: https://radar.cloudflare.com/domains (accessed on 27 August 2025).
28. PhishTank. PhishTank—Join the Fight Against Phishing. Available online: https://phishtank.org/ (accessed on 27 August 2025).
29. OpenPhish. Phishing Intelligence—Feeds and Datasets. Available online: https://openphish.com/ (accessed on 27 August 2025).
30. Google. Safe Browsing Lists—Safe Browsing APIs (v4); Developer Documentation. Available online: https://developers.google.com/safe-browsing (accessed on 27 August 2025).
31. Salmi, F. FQDN Model; GitHub Repository. 2025. Available online: https://github.com/fabriziosalmi/fqdn-model/ (accessed on 27 August 2025).
32. Banciu, D.; Cîrnu, C.E. AI Ethics and Data Privacy Compliance. In Proceedings of the 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Ploiești, Romania, 30 June–1 July 2022; IEEE: New York, NY, USA, 2022; pp. 1–5.
33. Lee, Y.-J. A study on the measuring methods of website security risk rate. *Appl. Sci.* **2024**, *14*, 42. [CrossRef]


**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.