MADONNA: browser-based malicious domain detection using optimized neural network by leveraging AI and feature analysis.

SENANAYAKE, J., RAJAPAKSHA, S., YANAI, N., KALUTARAGE, H. and KOMIYA, C.

2025

Computers & Security 152 (2025) 104371

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

MADONNA: Browser-based malicious domain detection using Optimized Neural Network by leveraging AI and feature analysis

Janaka Senanayake [a,*], Sampath Rajapaksha [a], Naoto Yanai [b], Harsha Kalutarage [a], Chika Komiya [b]

[a] School of Computing, Engineering and Technology, Robert Gordon University, Aberdeen, United Kingdom
[b] Department of Information Security Engineering, Osaka University, Japan

ARTICLE INFO

Keywords:
Malicious domain detection
Artificial intelligence
Feature engineering
Browser extension

ABSTRACT

Detecting malicious domains is a critical aspect of cybersecurity, with recent advancements leveraging Artificial Intelligence (AI) to enhance accuracy and speed. However, existing browser-based solutions often struggle to achieve both high accuracy and efficient throughput. In this paper, we present MADONNA, a novel browser-based malicious domain detector that exceeds the current state-of-the-art in both accuracy and throughput. MADONNA utilizes feature selection through correlation analysis and model optimization techniques, including pruning and quantization, to significantly enhance detection speed without compromising accuracy. Our approach employs a Shallow Neural Network (SNN) architecture, outperforming Large Language Models (LLMs) and state-of-the-art methods by improving accuracy by 6% (reaching 0.94) and F1-score by 4% (reaching 0.92). We further integrated MADONNA into a Google Chrome extension, demonstrating its practical application with a real-time domain detection accuracy of 94% and an average inference time of 0.87 s. These results highlight MADONNA's effectiveness in balancing speed and accuracy, providing a scalable, real-world solution for malicious domain detection.

1. Introduction

The incidence of cybercrime has surged dramatically in recent years, driven by the increasing sophistication and variety of tactics employed by adversaries. Among these tactics, the use of rogue domains has become particularly prevalent. These malicious domains serve multiple purposes for cybercriminals, including hosting command and control (C&C) servers, distributing malware, and setting up phishing websites to deceive unsuspecting users. Alarmingly, it has been reported that approximately 40,000 malicious domains are created daily, contributing to an estimated financial loss of $17,700 every minute (Saleem Raja et al., 2021). To evade detection mechanisms such as blacklists, attackers frequently generate short-lived domains using domain generation algorithms (DGAs). This evasion technique has significantly complicated the landscape of cybersecurity, driving the need for more sophisticated detection methods. As a result, the application of machine learning (ML) to detect these malicious domains has garnered significant attention and research interest in recent years (Yu et al., 2018b).

Given that the browser is the primary interface between the user and the internet, it is ideally positioned to provide real-time alerts against ongoing cyber threats, such as phishing attacks. However, despite the advancements in this area, there is still a pressing need to enhance both the inference throughput and the accuracy of existing malicious domain detection models to make them more efficient and practical for real-world deployment. In this paper, we propose the design of an AI-based domain detection application specifically tailored for web browsers, aiming to achieve higher detection accuracy while minimizing computational overhead, thereby improving throughput.

Developing such an application is a complex and challenging task. The primary challenge lies in balancing the trade-off between accuracy and throughput. Previous works have often sacrificed accuracy to improve throughput by employing simple neural network models or traditional machine learning models. While these models can process data more quickly, their simplicity often results in lower detection accuracy, which undermines their effectiveness in real-world applications. On the other hand, if a more sophisticated and enriched machine learning model is implemented without considering its computational demands, the resulting application may suffer from significantly reduced throughput (Iwahana et al., 2021). This reduction in throughput can make the application unsuitable for deployment in a web browser

* Corresponding author.
E-mail addresses: j.senanayake@rgu.ac.uk (J. Senanayake), s.rajapaksha@rgu.ac.uk (S. Rajapaksha), yanai@ist.osaka-u.ac.jp (N. Yanai), h.kalutarage@rgu.ac.uk (H. Kalutarage), c-komiya@ist.osaka-u.ac.jp (C. Komiya).

https://doi.org/10.1016/j.cose.2025.104371
Received 1 October 2024; Received in revised form 9 January 2025; Accepted 8 February 2025
Available online 17 February 2025

environment, where high performance and low latency are critical. Therefore, addressing the trade-off between accuracy and throughput is essential for developing a browser-based malicious domain detection application that is both effective and efficient.

To tackle these challenges, our approach is based on two key strategies. First, we focus on identifying and selecting the most relevant features for detecting malicious domains. Feature selection is a critical aspect of building an effective ML model, as it directly impacts both the accuracy and efficiency of the model (Tang and Mahmoud, 2021). One promising approach is to analyze feature correlations (Alhogail and Al-Turaiki, 2022; Li et al., 2020), which allows us to identify and eliminate redundant or irrelevant features. By streamlining the feature set, we can enhance the throughput of the model without sacrificing accuracy. Second, we employ model optimization techniques such as pruning and parameter quantization in deep learning models. Pruning involves removing unnecessary neurons and connections within the model, while parameter quantization reduces the precision of the model's parameters. Both techniques are designed to reduce the computational load of the model, thereby improving throughput, while maintaining a high level of accuracy. By carefully applying these techniques, we aim to develop a malicious domain detection application that is both highly accurate and efficient, making it suitable for deployment in web browsers.

Based on the above viewpoints, we propose *MADONNA (Browser-Based MAlicious Domain Detection using Optimized Neural Network by Leveraging AI and Feature Analysis)*. We demonstrate that MADONNA outperforms other state-of-the-art models, including LLMs, in both accuracy and throughput, thanks to its use of feature correlation analysis and optimized neural network learning techniques, achieving an accuracy of 94%. Our key contributions are as follows:

- We present MADONNA, an open-source browser-based extension (plug-in) that runs AI in the backbend to detect malicious domains in near real-time.
- We analyze feature correlations for malicious domain detection by removing highly correlated features to improve both throughput and accuracy.
- We show that parameter quantization and pruning in a deep learning model can strikingly improve throughput by keeping the same-level accuracy for malicious domain detection.
- We conduct a real-world experiment to distinguish benign and malicious domains in the real world and show that MADONNA can detect these domains precisely as an extension to our previous work (Senanayake et al., 2024).
- We show that applying the proposed neural network-based approach outperformed the LLM-based approach for malicious domain detection both in accuracy and throughput as an extension to our previous work (Senanayake et al., 2024).
- We demonstrate that MADONNA outperforms the benchmarked models with respect to the accuracy and throughput of malicious domain detection an extension to our previous work (Senanayake et al., 2024).

The rest of the paper is organized as follows: Section 2 contains preliminaries while Section 3 discusses the related works. Section 4 explains the methodology, and Section 5 discusses the results. Finally, the conclusions and future work directions are discussed in Section 6.

2. Preliminaries

This section provides background information on domain names and malicious domain detection, as well as AI technologies and LLMs, to support a better understanding of our work.

2.1. Domain names

Domain names serve as textual identifiers that are linked to network hosts and are managed through the Domain Name System (DNS), a crucial component of the internet infrastructure. DNS operates by translating human-readable domain names, such as "example.com", into numerical IP addresses that computers use to locate each other on the network. This translation process allows users to access websites using easy-to-remember names rather than complex numerical codes. Domain names are organized hierarchically within a structure known as a namespace, which is divided into zones. Each zone represents a portion of the domain name space that is administered by a specific entity or organization. At the apex of this hierarchy is the root zone, which acts as the ultimate authority for all domain names. Beneath the root zone are various top-level domains (TLDs), which are the most prominent and widely recognized components of domain names.

Top-level domains come in different forms, including generic TLDs (gTLDs) such as `.com`, `.org`, and `.net`, as well as country code TLDs (ccTLDs) like `.us` (United States), `.uk` (United Kingdom), and `.jp` (Japan). These TLDs are at the top of the DNS hierarchy, and they play a critical role in defining the overall structure of domain names. For instance, `.com` is the most popular TLD used globally, primarily for commercial websites, while `.org` is commonly associated with non-profit organizations. Within each TLD, there exist numerous second-level domains, which are typically the names registered by individuals or organizations, such as "example" in "example.com". These second-level domains can further be divided into subdomains, which add additional layers to the hierarchy. For instance, in "sub.example.com", "sub" is a subdomain of the second-level domain "example.com". Each level of the hierarchy, from the TLD down to subdomains, is managed in a distributed manner by different organizations, ensuring the scalability and robustness of the DNS system.

A typical URL (Uniform Resource Locator), which is used to locate resources on the web, is composed of several parts, each serving a specific function. These parts include the protocol (such as "http" or "https"), which indicates the method used to retrieve the resource, the optional subdomains, the domain name itself, the TLD, and any additional subdirectories or paths that lead to a specific resource within the domain. For example, in the URL "https://sub.example.com/path/page", "https" is the protocol, "sub" is the subdomain, "example" is the domain name, ".com" is the TLD, and "/path/page" represents the subdirectory structure within the website. This hierarchical and organized structure of domain names within URLs allows for the efficient and systematic navigation of the vast array of resources available on the internet.

2.2. Malicious domain detection

There are two primary approaches for detecting malicious domains: knowledge-based methods and ML-based methods (Yu et al., 2018b). The knowledge-based approach relies on human expertise, predefined rules, and heuristics to differentiate between benign and malicious domains. These methods typically involve the use of signature-based detection, where known patterns or characteristics of malicious domains are identified and flagged. While this approach can be effective in identifying known threats, it has significant limitations, particularly when it comes to detecting new or evolving threats. Because knowledge-based methods depend on previously identified patterns, they often fail to recognize novel attacks, leading to vulnerabilities known as zero-day exploits, where new, unseen threats can bypass detection systems entirely. In contrast, ML-based methods offer a more dynamic and adaptable approach to detecting malicious domains. These methods leverage the power of algorithms to analyze vast amounts of data and can effectively classify previously unknown domains as either benign or malicious (Palaniappan et al., 2020). ML models, especially in a supervised learning setting, are trained on large datasets containing

labeled examples of both malicious and benign domains. Through this training process, the model learns to identify patterns and features that are indicative of malicious behavior. Once trained, the model can then infer the nature of new, unseen domains with a high degree of accuracy by analyzing their features.

Supervised learning is particularly favored in this context due to its efficiency in handling large datasets and its ability to select the most relevant features from raw data for accurate classification (Rajapaksha et al., 2023; Rupa et al., 2021; Senanayake et al., 2021, 2023; Shi et al., 2018; Vinayakumar et al., 2018). By training the model on labeled data, where each domain is categorized as either benign or malicious, the system can effectively infer the classification of new domains. This capability makes machine learning-based methods highly effective in detecting malicious domains, even those that have not been previously encountered, thereby offering a robust defense against a wide range of cyber threats.

Given the advantages of machine learning in this domain, our focus is on developing and refining malicious domain detection techniques that leverage ML. The core idea is to use ML models to provide real-time inferences, determining whether a given domain is malicious or not. In practical terms, this process involves an ML model learning the features of domain names during the training phase, where it is exposed to a labeled dataset that includes both benign and malicious domains. These features could include various attributes such as the length of the domain name, the presence of certain keywords, the domain's registration information, and patterns within the domain's structure. Once the model has been trained, it enters the inference phase, where it is used to evaluate new domains. The model analyzes the features of a target domain and then predicts its label as either benign or malicious. In recent years, deep neural networks have become a popular and effective approach for this type of domain detection. These networks, with their ability to capture complex patterns and relationships in the data, are well-suited to the task of identifying malicious domains, especially in environments where threats are continuously evolving. By focusing on ML, particularly deep learning techniques, we aim to enhance the accuracy and efficiency of malicious domain detection, providing a more robust and scalable solution to this critical cybersecurity challenge.

2.3. AI technologies

AI has emerged as a transformative tool for enhancing web security, offering capabilities that surpass traditional methods. In particular, AI techniques, including ML and Deep Learning (DL), enable the analysis of large datasets, the identification of patterns, and the generation of highly accurate predictions. These capabilities are crucial for developing adaptive and proactive security solutions capable of detecting and mitigating novel and sophisticated cyber threats. AI is now applied across various domains of web security, including anomaly detection, threat intelligence, user behavior analysis, and malicious domain detection.

ML involves the development of algorithms that allow systems to learn from data and enhance their performance autonomously. In the context of web security, ML is employed for tasks such as spam filtering, phishing detection, and anomaly detection in network traffic. Common ML paradigms include supervised, unsupervised, and reinforcement learning. Supervised learning trains a model on labeled data, where the model learns to map input features to specific outputs, such as classifying domains as either malicious or benign. Unsupervised learning identifies hidden patterns in data without predefined labels, and is often used in clustering and anomaly detection. Reinforcement learning focuses on training agents to make sequential decisions, frequently applied in adaptive security systems designed to respond to evolving threat landscapes.

DL, a subset of ML, uses multi-layered neural networks (deep neural networks) to model complex data patterns. DL has shown significant potential in web security, particularly in scenarios requiring the analysis of high-dimensional data, such as CAPTCHA breaking, natural language processing for phishing detection, and behavior-based malware detection. Convolutional Neural Networks (CNNs), although primarily used in image recognition, have applications in phishing website detection through the analysis of visual similarities. Recurrent Neural Networks (RNNs) are particularly effective for sequence-based data, such as time-series analysis in detecting anomalous network traffic or fraudulent activities. Furthermore, transformer architectures, such as BERT and GPT, have proven effective in advanced natural language processing tasks, including phishing email detection and the identification of malicious scripts.

2.4. Language models

Language models (LMs) are computational systems designed to understand and generate human language. These models can predict the probability of word sequences or create new text based on a given input (Chang et al., 2023). LMs are categorized into Statistical Language Models (SLM), Neural Language Models (NLM), Pre-trained Language Models (PLM), and Large Language Models (LLM) (Zhao et al., 2023). PLMs, which are based on the transformer architecture, utilize a self-attention mechanism. One such model, Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), was developed by pre-training on large, unlabeled datasets with specific pre-training tasks (Zhao et al., 2023). BERT employs the masked language model (MLM) technique, where certain input tokens are masked, and the model predicts the original token using surrounding context (Devlin et al., 2019). Additionally, BERT includes a next-sentence prediction task, enabling joint pre-training of text pair representations. This architecture allows BERT to be fine-tuned for various NLP tasks, often delivering state-of-the-art results. Increasing model size and training data enhances performance in tasks like question-answering, language inference, and classification.

The larger PLMs commonly referred to as LLMs (Shanahan, 2024). A notable application of LLMs is ChatGPT, recognized for its advanced conversational abilities (Rajapaksha et al., 2024). Since these LLMs are pre-trained, they can be prompted directly using text commands, commonly referred to as prompts, to perform various tasks (Bucher and Martini, 2024). This enables the use of zero-shot and few-shot learning approaches. In zero-shot learning, only a prompt is needed to guide the LLM, while in few-shot learning, a few examples are included within the prompt to enhance performance. These pre-trained LLMs can also be fine-tuned for specific downstream tasks, such as classification, often yielding significant improvements over their pre-trained counterparts (Bucher and Martini, 2024).

3. Related works

We describe related works in three aspects: feature selection and feature engineering, AI-based malicious domain detection methods and browser-based applications.

3.1. Feature selection and feature engineering

The detection of malicious domains using AI methods has been a rapidly evolving field, with a significant amount of research dedicated to improving the accuracy and efficiency of detection models. Feature selection plays a critical role in this process, as the quality and relevance of features directly impact the performance of the machine learning models. Traditionally, the primary approach for malicious domain detection has been to utilize enriched models that analyze only domain names (Berman, 2019; Yang et al., 2020; Yu et al., 2018a). These models treat domain names as text data and leverage various text-based features to identify potentially harmful domains. However, while domain names provide valuable information, relying solely on them can be insufficient for comprehensive detection.

To enhance detection capabilities, researchers have increasingly incorporated additional data sources beyond domain names. For instance, DNS information, such as query patterns, response codes, and time-to-live (TTL) values, has been shown to significantly improve detection accuracy by providing contextual insights into domain behavior (Sun et al., 2019; Sun et al., 2020). Similarly, web content analysis, which involves examining the actual content hosted on a domain, can reveal indicators of phishing or other malicious activities (Abdelnabi et al., 2020; Ariyadasa et al., 2022). This multi-faceted approach, combining domain names, DNS data, and web content, forms the foundation of the feature set used in our work, following the methodology outlined in Iwahana et al. (2021).

In the realm of feature engineering, improving model accuracy often involves careful evaluation of feature importance. Identifying which features contribute most to the model's predictions allows for the refinement of the feature set, ensuring that only the most informative features are retained. One common technique for assessing feature importance is principal component analysis (PCA), which reduces the dimensionality of the data while preserving as much variance as possible (Zamir et al., 2020). Another approach is the use of decision trees, which inherently rank features based on their contribution to the classification task (Yahya et al., 2021). Beyond these, techniques like zero score computation (Shabudin et al., 2020) and analysis of data point equality (Zabihimayvan and Doran, 2019) can be employed to eliminate redundant or irrelevant features, thus enhancing the model's performance. In our work, we adopt feature correlation analysis as outlined in Li et al. (2020) to identify and remove redundant features, which is crucial for maintaining high model accuracy and reducing computational overhead.

3.2. AI-based malicious domain detection methods

Since the heuristic based malicious domain detection methods such as Antonakakis et al. (2011) and Sood and Enbody (2011) face challenges in scalability and the ability to detect previously unknown (zero-day) threats, recently, AI has emerged as a powerful tools for detecting malicious domains by analyzing domain-related features in a more dynamic and adaptive manner. AI-based approaches for malicious domain detection focus on leveraging domain-related features, such as lexical analysis, WHOIS information, DNS traffic patterns, and user behavior, to classify domains as benign or malicious. Machine learning algorithms, particularly supervised learning methods, are trained on large datasets of labeled domains to detect patterns indicative of malicious activity.

Lexical analysis of domain names has become a prominent technique in AI-based detection. Malicious domains often exhibit unique structural patterns, such as random strings or the inclusion of suspicious keywords (Zhao et al., 2019). Features like domain length, character distribution, and the use of non-alphabetic characters have been shown to be useful indicators for detecting domains generated by DGAs Schiavoni et al. (2014). In this approach, classification algorithms such as Random Forests, Support Vector Machines (SVM), and Neural Networks are employed to distinguish between legitimate and malicious domain names. For instance (Yu et al., 2018a) proposed a method utilizing the character-level convolutional neural networks (CNN) to analyze the lexical structure of domain names. By training on labeled datasets of both malicious and benign domains, this model was able to generalize well, achieving high detection accuracy even on previously unseen domains.

DNS-based features are central to many AI-based approaches, as DNS queries are an essential component of network communication. Patterns such as query frequency, TTL values, and query response times can reveal anomalies associated with malicious domains (Bilge et al., 2011). For example, domains with extremely short TTL values may indicate dynamic behavior often associated with malicious activities, such as fast-flux networks (Kumar and Xu, 2018). Authors in Yadav et al. (2012) proposed a DNS-based anomaly detection method that identifies malicious domains by analyzing features such as the distribution of query times and the entropy of domain names. This method, although effective in certain cases, could struggle with more sophisticated adversaries. AI-enhanced methods build upon this by incorporating additional contextual data and leveraging unsupervised learning to detect anomalous patterns.

Another line of research has focused on using WHOIS registration information as a feature for malicious domain detection. Attackers often register domains using false or privacy-protected information. Combining WHOIS features, such as registration duration, registrar reputation, and domain ownership changes, with machine learning techniques can improve detection rates (Marchal et al., 2014). Incorporating WHOIS data into detection systems allows for a more holistic view of domain characteristics. For example, supervised learning models can be trained to recognize patterns in registration behaviors that correlate with malicious domains, including unusually short registration periods or frequent changes in ownership details (Fernandez et al., 2024).

Behavioral analysis looks at how users interact with domains. For example, domains that prompt sudden spikes in traffic or exhibit specific click-through behaviors are often associated with phishing or malware campaigns. Using machine learning algorithms like k-means clustering, it is possible to group domains based on similar behavioral patterns and identify outliers that are likely to be malicious (G. Martín et al., 2021).

Deep learning techniques, particularly neural networks, have shown promise in the domain detection space due to their ability to learn complex and non-linear relationships between features. Recurrent neural networks (RNNs) and long short-term memory (LSTM) networks have been applied to domain name sequence analysis, allowing for the detection of malicious domains based on sequential patterns (Saxe and Berlin, 2015). They introduced a neural network-based approach that uses an LSTM model to classify domains based on both lexical features and DNS-related information. The model's ability to learn from sequential data allowed it to outperform traditional machine learning methods, especially in detecting domains generated by DGAs. Similarly, Vinayakumar et al. (2018) proposed a hybrid approach that combines CNNs with RNNs to capture both spatial and temporal features from domain names and DNS traffic data.

Few studies have employed large language models (LLMs) for the task of malicious URL classification. In Mahdaouy et al. (2024), the authors introduced DomURLs_BERT, a pre-trained language model designed to detect and classify malicious or suspicious domain names and URLs. DomURLs_BERT was pre-trained using the masked language modeling objective on a large-scale, multilingual corpus that includes URLs, domain names, and domain generation algorithm datasets, achieving state-of-the-art performance across multiple datasets. Similarly, URLBERT, a pre-trained representation learning model, was introduced for URL classification and detection tasks in Li et al. (2024), and it achieved state-of-the-art results in phishing URL detection, web page classification, and ad filtering. However, training LLMs is computationally expensive, and their inference latency is generally higher compared to shallow neural network models.

3.3. Browser based applications for malicious domain detection

When it comes to browser-based applications, the integration of malicious domain detection into web browsers has seen significant developments. For example, specialized browsers have been developed specifically for detecting phishing sites (H.R. et al., 2020), although these are often not as widely used as mainstream browsers like Firefox or Google Chrome. More commonly, researchers have developed browser plug-ins that integrate phishing detection capabilities into existing browsers (Ariyadasa et al., 2022). These tools typically rely on whitelists and blacklists to flag suspicious domains, but they are

limited in scope and often do not address the broader spectrum of malicious domains. A recent survey (Tang and Mahmoud, 2021) highlights the prevalence of such approaches, noting the reliance on traditional detection mechanisms.

Our work seeks to expand on these existing solutions by focusing on general malicious domain detection, including but not limited to phishing sites, using machine learning as the core detection method. Unlike tools that rely on static lists, our approach leverages dynamic AI models capable of adapting to new threats. The closest existing work to our approach is MADMAX (Iwahana et al., 2021), a browser-based application that also employs feature selection for detecting malicious domains. Additionally, the work in Alhogail and Al-Turaiki (2022) provided valuable insights into feature importance evaluation, further informing our methodology. Our goal is to design a high-performance browser extension, MADONNA, which not only matches but exceeds the capabilities of benchmark models like MADMAX. To achieve this, MADONNA will be rigorously benchmarked against state-of-the-art models to ensure its superiority in terms of both accuracy and efficiency.

In terms of implementation, the development of deep learning models within web browsers has become increasingly feasible thanks to advances in libraries like TensorFlow.js (Smilkov et al., 2019) and JSDOOP (Morell et al., 2019), which enable the execution of complex machine learning algorithms directly in JavaScript. These tools have the potential to deliver performance on par with native JavaScript applications while offering the sophisticated capabilities of deep learning. Moreover, MADONNA is designed to function as a distributed platform, leveraging the principles outlined in Huang et al. (2022) to efficiently process data and make real-time inferences. While these libraries are primarily focused on building generalized machine learning platforms, MADONNA is specifically tailored for the detection of malicious domains, offering a specialized solution that is both powerful and practical for browser-based deployment.

By integrating these advanced techniques and leveraging the latest developments in browser-based AI, MADONNA aims to set a new standard for malicious domain detection, providing users with a robust, efficient, and accurate tool for protecting against a wide range of cyber threats.

4. Methodology

This section details the methodology used for detecting malicious domains, structured into five subsections: Problem Formulation, Feature Extraction, Model Training with LLMs, Neural Network-Based Model Training and Optimization, and Browser-Based Deployment. The Problem Formulation subsection defines the task of detecting malicious domains as a classification problem. Feature Extraction outlines how the system selects relevant features from domain data. Next, Model Training with LLM describes the use of LLMs like GPT-4o in the training process to benchmark the performance of MADONNA's neural network architecture. In the Neural Network-Based Model Training and Optimization section, the focus shifts to MADONNA's core model, a Shallow Neural Network, and the optimization techniques, such as pruning and quantization, used to streamline the model for faster and more efficient predictions. The Browser-Based Deployment subsection describes how the trained model is integrated into a browser extension. When a user interacts with the MADONNA extension to check the safety of a domain, the extension triggers the system's Application Programming Interface (API). The API performs feature extraction in real-time, passing the relevant data to the trained SNN model. The model processes this information and generates a prediction on the domain's status, which is promptly displayed to the user through the extension's interface. This ensures that MADONNA delivers fast and accurate malicious domain detection directly within the browser environment, making it a practical solution. Fig. 1 provides a visual summary of these steps, illustrating how each component contributes to the system's overall functionality.

4.1. Problem formulation

We formalize the problem of domain detection based on ML below. Let $\mathcal{F} = \{f_1, \ldots, f_l\}$ be a set of features. Each domain $d_i \in D$ has features $F_i = \{f_{i,1}, \ldots, f_{i,l}\}$, where D denotes a set of domains, and $l \in \mathbf{N}$ denotes the size of F_i, i.e., the number of features of each domain. In addition, each $d_i \in D$ has a label $L_i \in \{0, 1\} \subseteq L$, where each label denotes a benign domain by 0 and a malicious domain by 1. For the size n of D, i.e., the number of domains, $DFL = \{(d_1, F_1, L_1), \ldots (d_n, F_n, L_n)\}$ denotes the combinations with domains, features, and labels. Let $Model = M(DFL)$ denote a trained model, where M denotes a learning algorithm. If d_t is a test domain (test case) unseen by M during its training time, our goal is then to obtain an inference result, $L_t = Model(F_t)$, by extracting features $F_t = \{f_{t,1}, \ldots, f_{t,l}\}$ for the unlearned domain d_t.

4.2. Feature extraction

This study used the dataset introduced in Chien et al. (2021) and Iwahana et al. (2021), which includes 25 features encompassing text-based, DNS-based, and web-based characteristics. The text-based features capture information derived from domain name strings and explore whether malicious domains can be identified based on the domain names alone. DNS-based features provide insights from DNS records associated with each domain and analyze the differences between DNS records of malicious and benign domains. Web-based features focus on the content hosted on the domains and examine the distinctive traits of content served by malicious domains.

Throughput is a key criterion for the proposed model, as real-time or near-real-time detection of malicious domains is crucial. To optimize performance, it is essential to select the minimum number of features while maintaining the highest possible detection rate. Fig. 2 presents the feature correlation metrics for the 25 features.

The label feature represents the ground truth, with 1 indicating malicious domains and 0 indicating benign ones. Based on this, the features n_ns and ns_similarity show the highest Pearson correlations with the label, while mean_TTL and stdev_TTL have the lowest correlations. Additionally, some features exhibit strong correlations with each other. For instance, active_time and life_time have a correlation of 0.97. As a result, one of these highly correlated features can be eliminated from the ML model to reduce redundancy.

In the study (Iwahana et al., 2021), the authors used the permutation importance algorithm to select seven key features. They also applied backward selection to account for feature correlation and distribution. The selected features include: length, n_ns, n_vowels, n_vowel_chars, life_time, n_constant_chars, n_nums, ns _similarity, n_other_chars, entropy, n_countries, n_mx, and n_labels. Table 1 summarizes the descriptions of these features and their behavior in both benign and malicious scenarios based on our analysis.

The feature distributions for a selection of variables are illustrated in Fig. 3. The x-axis represents the malicious and benign class labels, while the y-axis shows the value ranges for each variable. None of the features achieve complete class separability. For instance, the variable with the highest correlation, ns_similarity (shown in Fig. 3), exhibits overlapping values between 0.71 and 1.00 for both benign and malicious domains. Additionally, this highlights the presence of outliers in the benign dataset. To improve generalization, these outliers are removed using the Z-score method, where values greater than +3 or less than −3 are considered thresholds for identifying outliers.

4.3. Model training with LLM

With recent advancements in LLMs, many of them are surpassing traditional AI models in natural language processing tasks. Since URLs can be regarded as a form of text, their effectiveness in the task of malicious domain classification was evaluated. Zero-shot, few-shot,

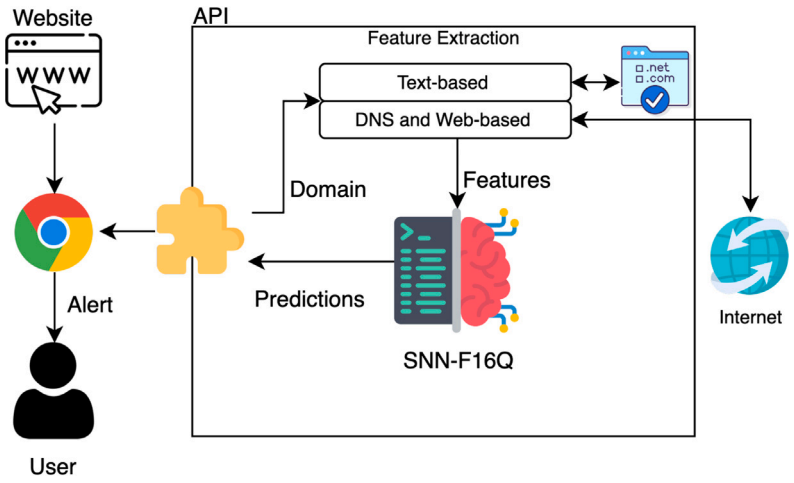


Fig. 1. The overview of MADONNA.

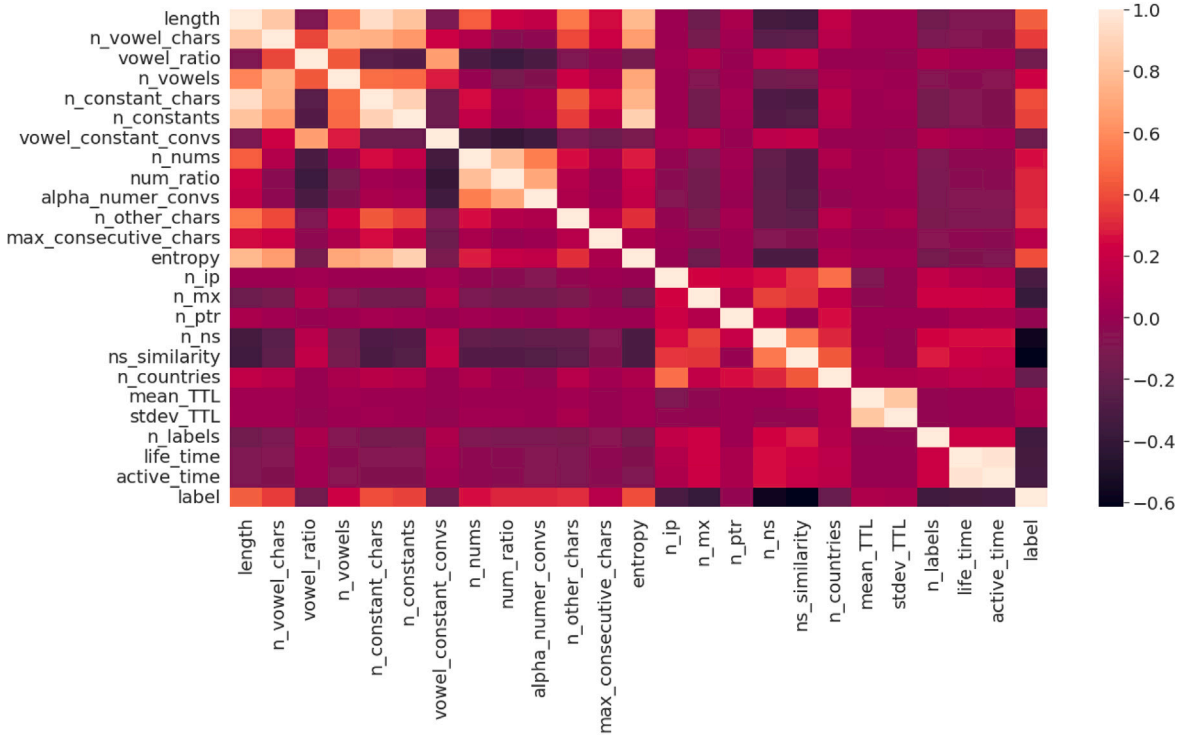
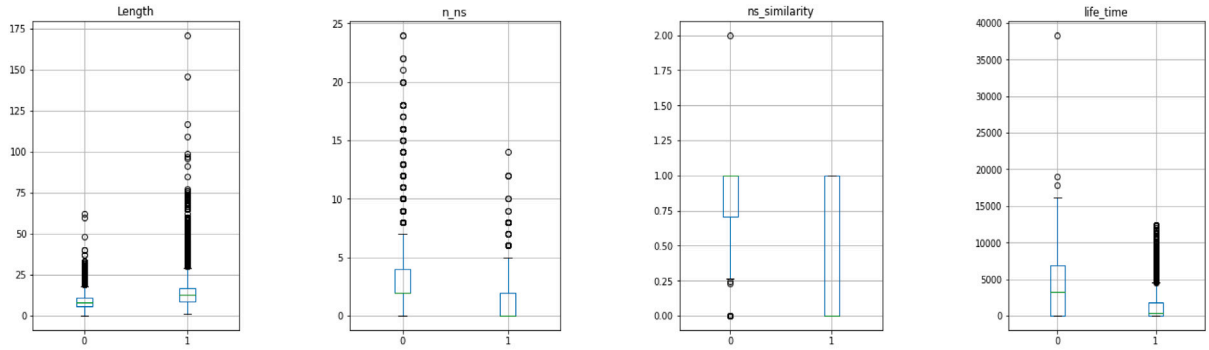


Fig. 2. Feature correlations.



(a) length (b) n_ns (c) ns_similarity (d) life_time

Fig. 3. Feature selection.

Table 1
Selected features.

Feature name	Description
length	The length of the domain. The average length of malicious domains is about two times that of benign domains.
n_ns	The number of distinct name servers. n_ns values tend to be low for malicious domains.
n_vowels	The number of vowels in the domain. These values tend to be high for malicious domains.
life_time	The difference of expiration date and creation date of WHOIS data, in days. Generally, life_time is low for malicious domains.
n_vowel_chars	The number of vowel characters in the domain. n_vowel_chars has similar characteristics as n_vowels.
n_constant_chars	The number of constant characters in the domain. Malicious domains include more constant characters.
n_nums	The number of numeric characters in the domain. This is typically high in malicious domains.
n_other_chars	Number of characters other than digits and alphabets in the domain. This is comparatively high in malicious domains.
entropy	The entropy of the domain. High values can be observed for malicious domains.
ns_similarity	The similarity between name servers. This is significantly low for malicious domains.
n_countries	The number of countries obtained from GeoLite2 service queried using each of the distinct IP addresses. This tends to be greater than 1 for malicious domains.
n_mx	The number of distinct mail exchange records. Low values can be observed for malicious domains.
n_labels	The number of HTML elements of the content. This is significantly low in malicious domains.

and fine-tuning approaches were applied to classify URLs as either malicious or benign using LLMs. OpenAI models were chosen due to their superior performance across various Natural Language Processing (NLP) tasks and their cost-effectiveness compared to open-source LLMs. In OpenAI's Chat Completions API, the role is defined as "user" and the prompt is set as the content for zero-shot and few-shot learning tasks. However, fine-tuning requires the data to be structured as a list of messages for training. The dataset was formatted accordingly and uploaded via the Files API for use in fine-tuning jobs. The fine-tuning job was initiated using the OpenAI SDK.[1] Once the model was trained, the fine-tuned version was employed as a zero-shot learning model for inference.

4.4. Neural network-based model training and optimization

A supervised learning model was trained using the dataset from Chien et al. (2021), with only the optimized features selected. To assess model performance, various experiments were conducted by training models with different machine learning algorithms, including Logistic Regression (LR), Random Forest (RF), and boosting algorithms like Gradient Boosting (GB), eXtreme Gradient Boosting (XGB), Light Gradient Boosting (LGB), and Extreme Learning Machine (ELM). Additionally, Multilayer Perceptron (MLP) and Shallow Neural Network (SNN) models were trained and evaluated in terms of accuracy, F1-score, and throughput, with the SNN showing strong performance. As a result, the optimal neural network model was chosen for further experiments. A simple model architecture was selected to minimize detection latency, and grid search was employed to determine the best hyperparameters for the SNN model.

4.4.1. Pruning and quantization

The throughput of a Neural Network can be enhanced by removing the least significant weight parameters, aiming to maintain accuracy while improving efficiency. Magnitude-based pruning (Vadera and Ameen, 2022) is a straightforward yet effective technique that eliminates weights while preserving accuracy. It works by gradually removing insignificant weights during model training by setting them to zero. Since model accuracy is influenced by the degree of sparsity, the sparsity level must be carefully selected to maintain accuracy. The TensorFlow Model Optimization Toolkit was used to implement magnitude-based pruning. Initially, the model was trained with all parameters, followed by pruning to reach 50% parameter sparsity, starting from 0%. The pruned model is referred to as SNN-P in this study.

Quantization (Idelbayev and Carreira-Perpinan, 2021) is another optimization technique that reduces the precision of the numerical values used for model parameters. Typically, TensorFlow uses 32-bit floating-point numbers, but quantization improves throughput by converting these 32-bit numbers to 16-bit or 8-bit values. While this can slightly reduce model accuracy due to the loss of precision, it significantly enhances efficiency. The TensorFlow optimization toolkit offers several quantization options. In our work, we applied non-optimized quantization (SNN-NOQ), dynamic range quantization (SNN-DRQ), float16 quantization (SNN-F16Q), and int8 quantization (SNN-I8Q). Additionally, TensorFlow's quantization process converts the model into a more lightweight TFLite version, so SNN-NOQ, SNN-DRQ, SNN-F16Q, and SNN-I8Q are all TFLite models.

4.5. Browser deployment

The browser extension developed in this work, called MADONNA, is designed to detect malicious domains in near real-time while users browse websites. Given the widespread use of web browsers, Google Chrome was chosen as the platform for developing this extension. It can be downloaded from GitHub[2] and is easy to install in Google Chrome.

The MADONNA extension is integrated with a Python Flask web API. Users can start the API by executing the start_api.bat file (for Windows) or the start_api.sh file (for Linux). To run the API, Python 3 must be installed. Once the API is active in the background, the browser extension sends the URL that the user intends to visit to the API, which then extracts the necessary text-based, DNS-based, and web-based features of the URL. These features are then passed to the trained SNN model, which predicts whether the domain is malicious or benign. Based on the model's results, the browser extension informs the user whether the URL is safe to visit.

5. Results and discussion

This section evaluates the performance of MADONNA using an existing dataset (Chien et al., 2021). The analysis includes key metrics such as accuracy, model size, and throughput. Additionally, we benchmark the MADONNA extension in Google Chrome against MADMAX (Iwahana et al., 2021) to measure its performance.

5.1. Experimental setting

The dataset presented in Chien et al. (2021), consisting of 48,252 domains (24,126 benign and 24,126 malicious), was utilized to train the model using the 13 identified key features. The SNN model was trained with 28 nodes in the hidden layer for 50 epochs, employing a batch size of 128 and the Adam optimizer with a learning rate of 0.001. To reduce the risk of overfitting, early stopping was implemented. The

[1] https://platform.openai.com/docs/guides/fine-tuning/create-a-fine-tuned-model

[2] https://github.com/softwaresec-labs/MADONNA

Table 2
LLM prompts.

Notation	Prompt
Prompt 1	*Classify whether the following domain is benign or malicious.*
Prompt 2	*You are a cybersecurity expert. Your job is to classify whether the following domain is benign or malicious. A malicious domain could be associated with phishing, malware distribution, or other harmful activities. A benign domain is safe and does not engage in harmful activities.*
Prompt 3	*You are a cybersecurity expert specializing in domain analysis. Your task is to determine whether the following domain is benign or malicious based on various threat indicators. Consider the following attributes: Reputation of the domain (Is it newly registered or associated with any suspicious activity?) Use of misleading or suspicious keywords Presence of uncommon top-level domains (TLDs) History of association with phishing, malware, or other cyber threats Context of use: How would this domain typically be encountered by users (e.g., in spam emails, fake websites)?*
Prompt 4	*You are a cybersecurity expert trained to assess the safety of domains. Your goal is to classify the following domain as benign or malicious. Consider these critical factors when making your decision: Domain reputation: Has the domain been flagged in security databases or blacklists? Registration details: Is the domain newly registered, or is it lacking information about the owner? Domain structure: Does the domain contain suspicious patterns like random characters, typosquatting (misspelling of legitimate domains), or extra subdomains? Top-level domain (TLD): Is the TLD unusual or commonly associated with malicious activity (.xyz, .top, etc.)? Content hosting: Does the domain host suspicious content or redirect users to unknown locations? SSL/TLS status: Does the site lack proper security certificates or use outdated encryption? Traffic patterns: Is the domain associated with abnormal or suspicious traffic, such as low or no traffic, or sudden spikes? Context: How is the domain presented (e.g., in emails, pop-ups, or suspicious ads)? Association with phishing or malware: Has the domain been reported for distributing malware or conducting phishing attacks?*
Prompt 5	*You are a cybersecurity expert. Your task is to classify whether the following domain is benign or malicious. A malicious domain is often associated with phishing, malware distribution, or harmful activities, while a benign domain is safe and used for legitimate purposes. Here are some examples of previously classified domains: - **Malicious domains (responded as 1'): ** (example list) - **Benign domains (responded as 0'): ** (example list) # Using the above examples as guidance, classify the following domain:*
Prompt 6	*You are a cybersecurity expert. Your job is to classify whether the following domain is benign or malicious.*

Table 3
Comparison of Accuracy, Precision, Recall, F1-Score and Throughput for LLM Zero-shot learning.

Model	Prompt	Accuracy	Precision	Recall	F1-Score	Inference time(s)
GPT-4o mini	Prompt 1	85%	0.66	0.62	0.63	0.002
GPT-4o mini	Prompt 2	81%	0.63	0.66	0.64	0.004
GPT-4o mini	Prompt 3	82%	0.64	0.67	0.65	0.005
GPT-4o mini	Prompt 4	85%	0.68	0.69	0.69	0.008
GPT-4o	Prompt 1	81%	0.64	0.67	0.65	0.01
GPT-4o	Prompt 2	82%	0.63	0.64	0.64	0.01
GPT-4o	Prompt 3	82%	0.66	0.71	0.68	0.04
GPT-4o	Prompt 4	88%	0.66	0.68	0.67	0.07

ReLU activation function was used in the hidden layer, while the classification layer utilized the softmax function. The model architecture was simple, featuring only 533 trainable parameters, which helped achieve low latency. The SNN model was developed using TensorFlow and Keras libraries, with both training and inference performed in the standard Google Colab environment, equipped with 12 GB of RAM.

5.2. LLM environment

The latest models, OpenAI GPT-4o (gpt-4o-2024-05-13) and GPT-4o Mini (gpt-4o-mini-2024-07-18), were used as the LLMs. Since LLM outputs depend on the prompts, different prompts were used during inference, as outlined in Table 2. These prompts were generated using ChatGPT. For few-shot learning, varying numbers of examples were used for both benign and malicious URLs, with balanced distribution between the two categories. This included sets of 10, 25, and 50 examples for each class. For all models, the max_token parameter was set to 1, as the predicted class was binary: 0 for benign and 1 for malicious URLs. Since this was a classification task, the temperature was set to 0 to ensure deterministic responses. During fine-tuning, the default hyperparameters were used, with a batch size of 76 and a learning rate of 1.8. The model was trained for 5 epochs to achieve a higher accuracy while minimizing costs.

Table 4
Comparison of Accuracy, Precision, Recall, F1-Score and Throughput for LLM Few-shot learning.

Model	Prompt	Accuracy	Precision	Recall	F1-Score	Inference time(s)
GPT-4o mini	Prompt 5	84%	0.68	0.69	0.69	0.009
GPT-4o	Prompt 5	86%	0.70	0.70	0.71	0.02

5.3. Accuracy and throughput of the LLMs

For the zero-shot learning experiments, prompts 1, 2, 3, and 4 were used, while prompt 5 was applied for the few-shot learning experiments with varying numbers of examples. Prompt 6 was utilized for inference with the fine-tuned model. The performance comparison for zero-shot learning is presented in Table 3. Despite using different prompts, there was no significant performance variation between GPT-4o and GPT-4o Mini. This could be because multiple factors must be considered to determine whether a URL is benign or malicious, making it difficult to capture all these aspects in a single, concise prompt. In the zero-shot learning experiments, GPT-4o Mini with prompt 4 achieved the highest F1-Score of 0.69. Additionally, GPT-4o Mini exhibited lower latency compared to GPT-4o due to its smaller model size. Since latency is also affected by the number of tokens, GPT-4o Mini with prompt 1 had the fastest inference time in the zero-shot learning experiments. Prompts 3 and 4 depend on real-time or up-to-date contextual information, such as domain reputation, SSL/TLS status, and traffic patterns. While GPT-4o mini has limited internet access, obtaining reliable data for these features remains challenging due to its restricted access to real-time or dynamic data for all URLs.

The performance comparison for the few-shot learning experiments is displayed in Table 4. Despite using different numbers of examples (10, 25, and 50), no significant variation in performance was observed across these sets. Table 4 specifically shows the results for experiments using 50 examples. A marginal improvement of 0.2 in F1-Score was noted for GPT-4o in the few-shot setting compared to zero-shot learning. Generally, few-shot learning performed better than zero-shot in most cases. However, since the classification of URLs as benign or malicious cannot be fully determined by simply analyzing the URLs, few-shot learning for this task did not yield significantly

Table 5
Comparison of Accuracy, Precision, Recall, F1-Score and Throughput for LLM Fine-tuning.

Model	Prompt	Accuracy	Precision	Recall	F1-Score	Inference time(s)
GPT-4o mini	Prompt 6	91%	0.90	0.91	0.90	0.003

better results than zero-shot learning. Additionally, few-shot learning for URL classification may be biased towards the provided examples, potentially failing to classify URLs with unseen characteristics accurately. It is difficult to create a well-representative set of benign and malicious examples, as URL characteristics can vary widely. Table 5 shows the results for fine-tuning GPT-4o Mini. As expected, fine-tuning significantly improved both zero-shot and few-shot learning, achieving an F1-Score of 0.90 with an average inference time of 0.003s. Although GPT-4o could potentially yield a higher F1-Score, it has higher latency compared to GPT-4o Mini. Given the critical importance of inference time for malicious URL detection, fine-tuning experiments with GPT-4o were not conducted.

5.4. Accuracy and throughput of the ML model

To assess the accuracy and F1-score of MADONNA's SNN model, traditional machine learning algorithms and boosting algorithms were trained using the same dataset and feature set. A 5-fold cross-validation was conducted for all models. Additionally, the SNN model was optimized for throughput through pruning and quantization techniques. Since the API's initialization time depends on the model size, it is essential to have a smaller model for improved performance. This is especially important for detecting malicious domains, as web users prefer fast browsing experiences without added delays. Therefore, it is crucial to ensure that domain analysis is completed within a reasonable timeframe to determine if a domain is malicious. The precision, recall, F1-score, model size, and inference time of these models, including the best-performing LLM, are compared in Table 6.

As shown in Table 6, boosting algorithms (XGB, GB, LGB) yielded higher accuracies and F1-scores compared to the other ML algorithms (LR, RF, ELM, and MLP). However, the SNN model surpassed all traditional machine learning models, achieving 94% accuracy and a 0.92 F1-score. Additionally, it was noted that the optimized SNN variants performed at a similar level, with the exception of a slight decrease in performance observed in the SNN-I8Q model.

The best-performing ML-based model was XGB, which required the largest model size, while the ELM model exhibited the longest inference time. Although ELM has a short training time, it necessitates a substantial number of model parameters to learn the patterns of benign and malicious domains, resulting in longer inference times. In contrast, the SNN model, with its simple architecture, only requires 33KB of memory and has an inference time of 64 μs. The pruned model (SNN-P) optimized these metrics further by removing insignificant weights. As anticipated, the quantized models improved both model size and inference time. However, despite its lower precision, SNN-I8Q had a slightly longer inference time compared to the SNN-F16Q model. This discrepancy may be due to the fact that quantized int models perform optimally on ARM devices, such as the Raspberry Pi.

We conducted experiments with LLMs to evaluate their effectiveness and compare them with the NN model. However, the final proposed model exclusively incorporates the NN model, as it proved to be more effective. Taking into account all the evaluation metrics, including the accuracy, precision, recall, F1-Score, model size, and inference time, the SNN-F16Q model was chosen for integration with the browser extension to detect malicious domains.

5.5. Performance of browser extension

Users receive notifications indicating the predicted malicious status of a domain when they click on the MADONNA extension icon in Google Chrome. Examples of these notifications are shown in Fig. 4. The extension displays a notification similar to Fig. 4(b) if a domain is benign (e.g., https://www.google.com). On the other hand, if the checked URL contains a malicious domain (e.g., https://chromnius.download/browser2/?mrddp=1&mrddz=2353135), a notification resembling Fig. 4(c) appears. Fig. 4(a) illustrates the notification displayed when checking a domain's malicious status.

The proposed browser extension version in this paper prioritizes swift malicious domain detection to avoid disrupting the user's browsing experience. While we acknowledge that sub resources within a domain can occasionally lead to malicious behavior, evaluating all resource domains introduces challenges in efficiency. For instance, analyzing the site paperswithcode.com in Google Chrome showed that loading all resources took 2.06 s, and processing each through MADONNA's model would significantly increase analysis time. To maintain usability, the current prototype focuses on detecting the primary domain. However, expanding to include resource domain checks is a potential future enhancement, contingent on optimizing the model or implementing efficient parallel processing techniques.

5.6. Comparison with existing works

A comparison between MADONNA and MADMAX was performed, as MADMAX represents the closest high-accuracy AI-based approach for detecting malicious domains. The results of this comparison are shown in Table 7, which reveals that MADONNA incorporates a greater number of text-based and DNS-based features than MADMAX, selected based on their importance in feature analysis. By integrating these features with an optimized and quantized SNN model, MADONNA achieved superior accuracy, F1-score, precision, and recall compared to MADMAX. Specifically, MADONNA exceeded MADMAX by 6% in accuracy and 4% in F1-score, marking a significant enhancement.

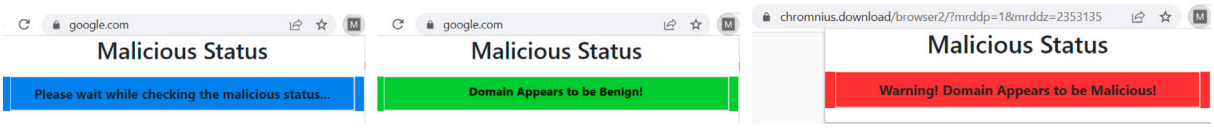
MADONNA is compatible with the widely used Google Chrome browser, and its backend model delivers significantly faster inference times (10 μs compared to MADMAX's 198 μs), which are notable advantages. Furthermore, the MADONNA browser extension predicts the malicious status of domains more quickly, with an average prediction time of 2.43 s, in contrast to MADMAX's 3.3 s. The MADONNA extension connects to an internally hosted web API, emphasizing user privacy, whereas MADMAX's extension relies on an external server for predictions. In summary, MADONNA outperforms MADMAX in every aspect.

We also conducted real-world experiments to assess the performance of MADONNA, validating the effectiveness of the MADONNA Chrome extension by visiting well-known benign and malicious sites from sources such as CyberCrime, PhishTank, and Tranco websites (Iwahana et al., 2021). The experimental setup included a machine with a Core i5 processor, 16 GB of RAM, and 66.6 Mbps fiber broadband internet connectivity. The MADONNA extension is able to predict whether a URL is malicious or benign in an average of 2.43 s, which is reasonable for practical use. Although the SNN model can determine the malicious status of a given feature set in just 10 μs, the MADONNA extension requires more time to extract certain DNS-based and web-based features, which accounts for the average notification time of 2.43 s. These experiments are categorized into accurate classifications and inaccurate classifications based on the listed malicious status in Table 8, Table 9, Table 10, and Table 11. The predictions made by MADONNA were validated against the CyberCrime, PhishTank, and Tranco[3] repositories, allowing us to derive the True Positive (TP), True

[3] https://tranco-list.eu/

Table 6
Comparison of Accuracy, Precision, Recall, F1-Score, Throughput.

Model	Accuracy	Precision	Recall	F1-Score	Model size(KB)	Inference time(μs)
LR	87%	0.87	0.86	0.87	443	151
RF	88%	0.92	0.83	0.87	480	41
GB	89%	0.91	0.89	0.89	422	112
MLP	83%	0.88	0.78	0.82	78	69
LGB	89%	0.91	0.89	0.89	482	98
XGB	90%	0.92	0.89	0.90	526	27
ELM	87%	0.88	0.86	0.87	312	198
SNN	94%	0.96	0.89	0.92	33	64
SNN-P	94%	0.96	0.90	0.92	19	36
SNN-NOQ	94%	0.96	0.90	0.92	4	19
SNN-DRQ	94%	0.96	0.90	0.92	4	16
SNN-F16Q	94%	0.96	0.90	0.92	3	10
SNN-I8Q	93%	0.95	0.89	0.91	3	12
GPT-4o mini	91%	0.90	0.91	0.90	–	3000



(a) Checking Domain (b) Benign Domain (c) Malicious Domain

Fig. 4. Chrome browser extension notifications.

Table 7
Comparison of MADMAX and MADONNA.

Aspect	MADMAX	MADONNA
Used Text-based features	length, n_constant_chars, n_vowel_chars, num_ratio	length, n_vowels, n_vowel_chars, n_constant_chars, n_nums, entropy, n_other_chars
Used DNS-based Features	n_ns	n_ns, ns_similarity, n_mx, n_countries
Used Web-based Features	life_time, n_labels	life_time, n_labels
Model Inference Time	198 μs	10 μs
Supported Browser	Firefox	Chrome
Avg. Prediction time in Browser	3.3 s	2.43 s
Accuracy	88%	94%
F1-Score	0.88	0.92
Precision	0.90	0.96
Recall	0.86	0.90
Connectivity	Externally-hosted Sever	Internally-hosted API

Negative (TN), False Positive (FP), and False Negative (FN) statuses of the results. These experiments serve to illustrate selected samples for TP, TN, FP, and FN cases, and we have confirmed that the accuracy and F1-score are consistent with the model's performance during training.

According to Table 8 and Table 10, all feature values exhibit similar distributions for the training dataset, as illustrated in Fig. 3. For instance, the feature life_time shows higher values exceeding 2000 for benign domains, while malicious domains typically have values below 365. Since these patterns are learned during training, the URLs in Tables 8 and Table 10 classify accurately. Fig. 5 presents the feature value distributions for four highly significant features. Labels 0 and 1 indicate the feature value distributions for the training data, while label FN shows the feature value distributions for FNs presented in Table 9. In this context, the feature values for these URLs fall within the learned ranges for benign features, which is also true for other features. Consequently, these malicious domains are misclassified as benign. Fig. 6 displays the feature value distributions of the same variables for false positive (FP) URLs. Similar to the FNs, the feature values of these URLs overlap with those of the malicious domain features. In particular, the values of n_ns coincide with the benign value range. However, it appears that the influence of one of the highly important features is insufficient to alter the overall prediction.

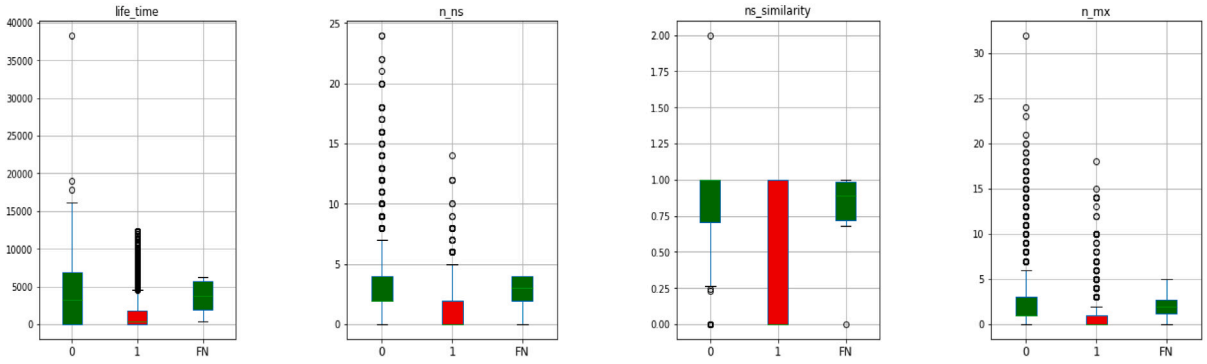
Additionally, we conducted experiments to compare MADONNA's performance with the malicious domain detection capabilities of modern web browsers including Google Chrome (without enabling the MADONNA plugin), Mozilla Firefox, and Microsoft Edge. For this, we used a subset of URLs comprising both malicious and benign domains, which were also utilized in evaluating MADONNA's results. These results are listed in Table 12. In this sample testing experiment, MADONNA achieved an accuracy of 83.33% and an F1-Score of 0.83, while the evaluated major web browsers recorded an accuracy of 75% and an F1-Score of 0.67. This highlights that MADONNA's capabilities surpass those of the built-in techniques for malicious site detections in existing web browsers.

5.7. Limitations

The misclassification results indicate that some malicious domains display benign feature values, while certain benign domains exhibit characteristics typical of malicious ones for the selected features. This suggests a need for more advanced and distinct web-based features to further reduce misclassifications. Features such as pop-up messages, alert boxes, a high percentage of advertisements, and site redirection are examples of malicious web-based features that could be considered during the feature analysis phase. However, these features must be extracted after the page has fully loaded in the browser, which may hinder the throughput and real-time usability of the solution.

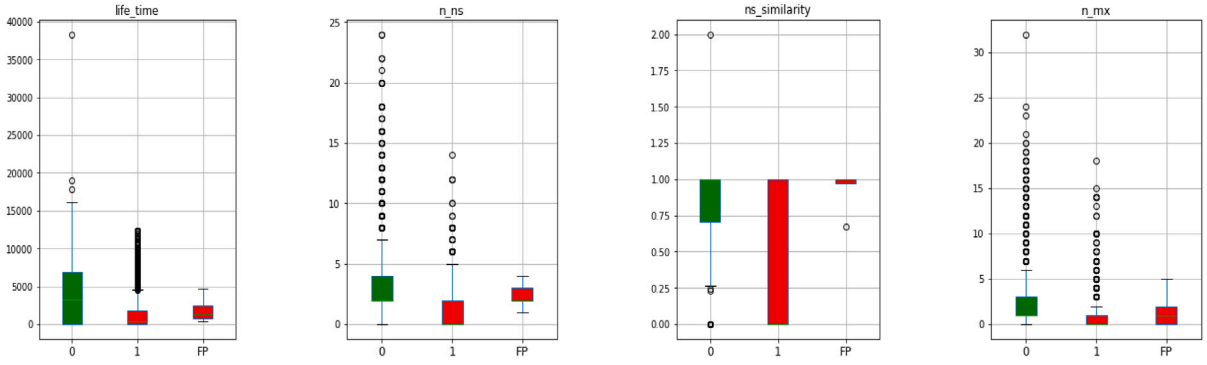
Another notable challenge is the computational overhead associated with connecting to a hosted API on the local machine, which could affect the system's responsiveness. Additionally, the reliance on continuous internet connectivity for real-time predictions introduces vulnerabilities in scenarios where stable internet access is not guaranteed.

While the MADONNA model boasts an impressive inference time of just 0 μs, the average prediction time via the browser extension is 2.43 s. This delay is primarily attributed to internet connectivity and cannot be easily mitigated with current web-engineering techniques.



(a) life_time (b) n_ns (c) ns_similarity (d) n_mx

Fig. 5. Feature values distribution - FN.



(a) life_time (b) n_ns (c) ns_similarity (d) n_mx

Fig. 6. Feature values distribution - FP.

The authors investigated the possibility of utilizing Pyscript[4] to eliminate the API execution step and develop a fully browser-based model; however, this approach proved unfeasible due to limited library support for extracting web and DNS-based features. Consequently, MADONNA continues to rely on a connection to a hosted API on the local machine, which adds computational overhead to the overall process.

5.8. Ethical consideration

Regarding cybersecurity ethics in the real-world experiments with MADONNA, we ensured that our activities did not impact the behavior or economics of any domains involved, including during feature extraction. We successfully conducted inferences on malicious domains already listed in existing blacklists, ensuring we did not undermine the trust associated with any domain.

To begin, we used Tranco (Pochat et al., 2018), a public list designed for cybersecurity research that is based on commercial services like Alexa. As noted in the study (Chien et al., 2021), Tranco is utilized to identify benign domains. Domains that appear in both Tranco and public databases of malicious domains are treated as both benign and malicious. While services such as Alexa are commercially driven, MADMAX could potentially compromise the effectiveness of these products by labeling their domains as malicious. However, MADMAX may also offer benefits to service providers. Specifically, by analyzing domains identified as malicious, providers could uncover previously undetected harmful services. Additionally, MADMAX can enhance the ranking of related services.

Furthermore, the features selected for this study focus on the general characteristics of malicious domains. In this context, we recommend providing feedback to the owners or organizations of domains that are inaccurately flagged as malicious, encouraging them to update their configurations. As previously mentioned, our goal is to support the detection of potentially harmful services. To prevent benign domains from being mistakenly identified as malicious in the future, we strongly advise a reevaluation of domain configurations.

6. Conclusion and future work

MADONNA offers a compelling and innovative approach to detecting malicious domains. We have made substantial contributions by optimizing feature extraction techniques, exploring and comparing various machine learning methods, and introducing a Shallow Neural Network architecture. This combination of advancements highlights MADONNA's superiority over existing state-of-the-art methods in terms of both accuracy and throughput. One of the key strengths of MADONNA lies in its use of correlation analysis for feature selection, which enhances the relevance of input data and improves the model's performance. The inclusion of model optimization techniques, such as pruning and quantization, allows for a significant reduction in computational complexity without compromising accuracy. Additionally, MADONNA's comparison with LLMs like GPT-4o and GPT-4o mini demonstrates its efficiency and precision, outperforming these more extensive, more resource-intensive models. Collectively, these contributions emphasize MADONNA's potential for real-world application in browser-based malicious domain detection.

Future research could focus on minimizing prediction time, possibly by further optimizing the model's architecture or exploring decentralized processing strategies that reduce dependency on external

[4] https://pyscript.net/

Table 8
Evaluation results of MADONNA - A sample of accurate benign classification domains (TN).

Domain	length	n_ns	n_vowels	life_time	n_vowel_chars	n_consstant_chars	n_nums	n_other_chars	entropy	n_mx	ns_similarity	n_countries	n_labels	Predicted	Pred. Time (s)
google.com	10	4	2	11322	4	5	0	0	2.6464	1	0.93	1	353	0	2.26
dl.acm.org	10	2	2	11324	2	6	0	0	3.1219	0	1.00	1	1987	0	3.4
phishtank.com	13	2	3	6208	3	9	0	0	3.5466	1	1.00	1	203	0	1.32
stackoverflow.com	17	4	3	74342	5	11	0	0	3.6901	5	0.55	1	907	0	2.32
youtube.com	11	4	3	6574	5	5	0	0	3.0958	1	0.93	1	415	0	5.4
facebook.com	12	4	3	12418	5	6	0	0	3.0221	1	0.94	1	198	0	1.63
ubuntu.com	10	3	2	7305	4	5	0	0	2.8464	1	0.94	1	1067	0	1.78
whatsapp.com	12	4	2	8400	3	8	0	0	3.2516	2	0.94	1	909	0	1.88
linkedin.com	12	8	3	8036	4	7	0	0	3.2516	4	0.77	1	655	0	2.37
plugins.jetbrains.com	21	4	5	9862	6	13	0	0	4.0114	0	0.70	1	55	0	1.48
17track.net	11	2	2	4748	2	6	2	0	3.2776	2	1.00	1	914	0	4.71
espncricinfo.com	16	4	3	5843	5	10	0	0	3.3278	2	0.66	1	1737	0	1.77
www.fifa.com	8	4	3	10226	3	4	0	0	2.7500	0	0.92	1	77	0	1.08
skyscanner.net	14	12	2	7670	3	10	0	0	3.1820	1	0.71	1	756	0	1.74
booking.com	11	4	2	9130	4	6	0	0	3.0272	2	0.93	1	2297	0	3.05
ieeexplore.ieee.org	19	0	3	12783	11	6	0	0	2.7926	0	0.00	1	148	0	2.23
conferenceranks.com	19	2	3	3287	6	12	0	0	3.2866	2	1.00	1	152	0	1.45
github.com	10	8	3	6219	3	6	0	0	3.3219	5	0.55	1	1519	0	1.58
tensorflow.org	14	4	2	2922	4	9	0	0	3.3249	5	0.97	1	788	0	2.43
paperswithcode.com	18	2	4	2557	6	11	0	0	3.7255	5	1.00	1	654	0	2.29
en.wikipedia.org	16	0	4	8305	7	7	0	0	3.4528	0	0.00	1	1076	0	1.71

Table 9
Evaluation results of MADONNA - A sample of inaccurate benign classification domains (FN).

Domain	length	n_ns	n_vowels	life_time	n_vowel_chars	n_consstant_chars	n_nums	n_other_chars	entropy	n_mx	ns_similarity	n_countries	n_labels	Predicted	Pred. Time (s)
linki.ee	8	4	2	365	4	3	0	0	2.5	5	0.68	1	0	0	2.42
hieraktualisieren0.yolasite.com	31	0	5	5478	14	14	1	0	3.8976	0	0.0	1	144	0	3.01
hudosantakakuuritai.com	23	2	4	2192	11	11	0	0	3.6211	1	1.0	1	6	0	2.59
rdsforum.ro	11	4	2	5905	3	7	0	0	2.8454	3	0.93	1	63	0	2.59
firepulsesports.com	19	2	4	1826	6	12	0	0	3.5766	2	1.0	1	118	0	3.21
afreebieempire.com	18	4	4	6209	9	8	0	0	3.1916	2	0.85	1	416	0	2.73

resources. Moreover, integrating web-based features and additional contextual data from online sources could further bolster MADONNA's capabilities, enabling it to adapt more dynamically to emerging threats. Another promising avenue for improvement is the incorporation of explainable AI (XAI) techniques, which would allow the plugin to provide more transparent and interpretable reasons behind each detection. Additionally, converting the current prototype plugin to a binary execution approach in future can simplify the process for users, eliminating the need to run Python in the backend. Despite these challenges, MADONNA marks a significant leap forward in the field of cybersecurity, particularly in malicious domain detection. Its innovative use of AI, model optimization, and real-world testing demonstrates great promise for future development. With continued refinement and the integration of additional features, MADONNA has the potential to become a robust and indispensable tool in browser-based threat detection systems.

Table 10
Evaluation results of MADONNA - A sample of accurate malicious classification domains (TP).

Domain	length	n_ns	n_vowels	life_time	n_vowel_chars	n_consstant_chars	n_nums	n_other_chars	entropy	n_mx	ns_similarity	n_countries	n_labels	Predicted	Pred. Time (s)
chromnius.download/browser2	18	2	4	365	6	11	0	0	3.6835	5	1.00	2	276	1	2.32
is.gd/qxAOYq	5	2	1	0	1	3	0	0	2.3219	5	1.00	1	58	1	1.86
abd-lqwm4.ml	12	4	1	0	1	8	1	1	3.2516	0	0.94	1	211	1	3.65
mesh-solutions.com.au	21	2	5	0	8	10	0	1	3.6538	1	1.00	1	207	1	3.42
15cn.ga/agency/rico/official	7	0	1	0	1	3	2	0	2.8074	2	0.00	1	9	1	2.52
1inch-io.network	16	2	3	0	5	8	1	1	3.6250	1	1.00	1	599	1	2.59
seekpair.org	12	2	4	365	5	6	0	0	3.2516	2	1.00	1	463	1	2.18
promo-take.com	14	2	3	365	5	7	0	1	3.3249	0	1.00	2	43	1	1.37
leiloescopart.org	17	2	4	365	7	9	0	0	3.4548	0	1.00	3	228	1	3.27
appcontajuridica.com	20	2	4	365	8	11	0	0	3.5464	0	1.00	2	32	1	1.73
gala-games-site-v.com	21	2	4	365	7	10	0	3	3.5585	5	1.00	3	2	1	2.03
www.wenolira.top	12	2	4	365	5	6	0	0	3.4182	0	1.0	1	9	1	3.06
fqnvsdaas.su	12	6	2	365	3	8	0	0	3.022	1	0.92	1	0	1	2.93

Table 11
Evaluation results of MADONNA - A sample of inaccurate malicious classification domains (FP).

Domain	length	n_ns	n_vowels	life_time	n_vowel_chars	n_consstant_chars	n_nums	n_other_chars	entropy	n_mx	ns_similarity	n_countries	n_labels	Predicted	Pred. Time (s)
coderzcolumn.com	16	4	3	1461	5	10	0	0	3.2806	0	0.97	1	497	1	2.81
pyscript.net	12	2	2	365	2	9	0	0	3.2516	1	1.0	1	126	1	3. 14
call4paper.com	14	4	3	1287	4	8	1	0	3.2359	0	0.67	1	3558	1	4.21
towardsdev.com	14	2	3	730	4	9	0	0	3.5216	1	1.0	1	596	1	3.18
intellipaat.com	15	2	4	2749	6	8	0	0	3.3736	1	1.0	1	5746	1	2.89

Table 12
Comparison of MADONNA's Detection Results with Chrome, Firefox and Edge Browsers Results.

Domain	MADONNA's Results	Chrome's Results	Firefox's Results	Edge's Results	Actual Status: Malicious/ Benign
paperswithcode.com	Domain Appears to be Benign!	Proceed to the Page	Proceed to the Page	Proceed to the Page	Benign
conferenceranks.com	Domain Appears to be Benign!	Proceed to the Page	Proceed to the Page	Proceed to the Page	Benign
phishtank.com	Domain Appears to be Benign!	Proceed to the Page	Proceed to the Page	Proceed to the Page	Benign
17track.net	Domain Appears to be Benign!	Proceed to the Page	Proceed to the Page	Proceed to the Page	Benign
rdsforum.ro	Domain Appears to be Benign!	Proceed to the Page	Proceed to the Page	Proceed to the Page	Malicious
towardsdev.com	Domain Appears to be Malicious!	Proceed to the Page	Proceed to the Page	Proceed to the Page	Benign
chromnius.download/browser2	Domain Appears to be Malicious!	Proceed to the Page	Proceed to the Page	Proceed to the Page	Malicious
leiloescopart.org	Domain Appears to be Malicious!	Proceed to the Page	Proceed to the Page	Proceed to the Page	Malicious
abd-lqwm4.ml	Domain Appears to be Malicious!	Dangerous site	Domain Blocked	Domain Blocked	Malicious
mesh-solutions.com.au	Domain Appears to be Malicious!	Dangerous site	Domain Blocked	Domain Blocked	Malicious
is.gd/qxAOYq	Domain Appears to be Malicious!	Dangerous site	Domain Blocked	Domain Blocked	Malicious

CRediT authorship contribution statement

Janaka Senanayake: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Sampath Rajapaksha:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Naoto Yanai:** Writing – review & editing, Validation, Supervision, Methodology, Investigation, Conceptualization. **Harsha Kalutarage:** Writing – review & editing, Validation, Supervision, Methodology, Formal analysis, Conceptualization. **Chika Komiya:** Software, Investigation, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank Robert Gordon University, United Kingdom and Osaka University, Japan for their support.

Data availability

Data will be made available on request.

References

Abdelnabi, S., Krombholz, K., Fritz, M., 2020. VisualPhishNet: Zero-day phishing website detection by visual similarity. In: Proc. of CCS 2020. ACM, pp. 1681–1698.

Alhogail, A.A., Al-Turaiki, I., 2022. Improved detection of malicious domain names using gradient boosted machines and feature engineering. Inf. Technol. Control. 51 (2), 313–331.

Antonakakis, M., Perdisci, R., Lee, W., II, N.V., Dagon, D., 2011. Detecting malware domains at the upper DNS hierarchy. In: 20th USENIX Security Symposium. USENIX Security 11, USENIX Association, San Francisco, CA, pp. 1–16, URL https://www.usenix.org/conference/usenix-security-11/detecting-malware-domains-upper-dns-hierarchy.

Ariyadasa, S., Fernando, S., Fernando, S., 2022. Combining long-term recurrent convolutional and graph convolutional networks to detect phishing sites using URL and HTML. IEEE Access 10, 82355–82375. http://dx.doi.org/10.1109/ACCESS.2022.3196018.

Berman, D.S., 2019. DGA CapsNet: 1D application of capsule networks to DGA detection. Inf. 10 (5), 157.

Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M., 2011. EXPOSURE: Finding malicious domains using passive DNS analysis. In: Ndss. pp. 1–17.

Bucher, M.J.J., Martini, M., 2024. Fine-Tuned'Small'LLMs (still) significantly outperform zero-shot generative AI models in text classification. arXiv preprint arXiv:2406.08660.

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., et al., 2023. A survey on evaluation of large language models. ACM Trans. Intell. Syst. Technol..

Chien, C.-J., Yanai, N., Okamura, S., 2021. Design of malicious domain detection dataset for network security. URL http://www-infosec.ist.osaka-u.ac.jp/~yanai/dataset.pdf.

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Assoc. for Comp. Ling.: Human Language Technologies. 1, pp. 4171–4186.

Fernandez, S., Hureau, O., Duda, A., Korczynski, M., 2024. WHOIS right? An analysis of WHOIS and RDAP consistency. In: International Conference on Passive and Active Network Measurement. Springer, pp. 206–231.

G. Martín, A., Fernández-Isabel, A., Martín de Diego, I., Beltrán, M., 2021. A survey for user behavior analysis based on machine learning techniques: current models and applications. Appl. Intell. 51 (8), 6029–6055.

H.R., M.G., M.V., A., S., G.P., S., V., 2020. Development of anti-phishing browser based on random forest and rule of extraction framework. Cybersecurity 3 (1), 1–20.

Huang, Y., Qiao, X., Dustdar, S., Li, Y., 2022. AoDNN: An auto-offloading approach to optimize deep inference for fostering mobile web. In: Proc. of INFOCOM 2022. pp. 2198–2207.

Idelbayev, Y., Carreira-Perpinan, M.A., 2021. An empirical comparison of quantization, pruning and low-rank neural network compression using the LC toolkit. In: 2021 International Joint Conference on Neural Networks. IJCNN, pp. 1–8. http://dx.doi.org/10.1109/IJCNN52387.2021.9533730.

Iwahana, K., Takemura, T., Cheng, J.C., Ashizawa, N., Umeda, N., Sato, K., Kawakami, R., Shimizu, R., Chinen, Y., Yanai, N., 2021. MADMAX: Browser-based malicious domain detection through extreme learning machine. IEEE Access 9, 78293–78314.

Kumar, S.A., Xu, B., 2018. A machine learning based approach to detect malicious fast flux networks. In: 2018 IEEE Symposium Series on Computational Intelligence. SSCI, IEEE, pp. 1676–1683.

Li, T., Kou, G., Peng, Y., 2020. Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods. Inf. Syst. 91, 101494.

Li, Y., Wang, Y., Xu, H., Guo, Z., Cao, Z., Zhang, L., 2024. URLBERT: A contrastive and adversarial pre-trained model for URL classification. arXiv preprint arXiv:2402.11495.

Mahdaouy, A.E., Lamsiyah, S., Idrissi, M.J., Alami, H., Yartaoui, Z., Berrada, I., 2024. DomURLs_BERT: Pre-trained BERT-based model for malicious domains and URLs detection and classification. arXiv preprint arXiv:2409.09143.

Marchal, S., François, J., State, R., Engel, T., 2014. PhishStorm: Detecting phishing with streaming analytics. IEEE Trans. Netw. Serv. Manag. 11 (4), 458–471. http://dx.doi.org/10.1109/TNSM.2014.2377295.

Morell, J.A., Camero, A., Alba, E., 2019. JSDoop and TensorFlow.js: Volunteer distributed web browser-based neural network training. IEEE Access 7, 158671–158684.

Palaniappan, G., S, S., Rajendran, B., Sanjay, Goyal, S., B S, B., 2020. Malicious domain detection using machine learning on domain name features, host-based features and web-based features. Procedia Comput. Sci. 171, 654–661.

Pochat, V.L., Van Goethem, T., Tajalizadehkhoob, S., Korczyński, M., Joosen, W., 2018. Tranco: A research-oriented top sites ranking hardened against manipulation. arXiv preprint arXiv:1806.01156.

Rajapaksha, S., Kalutarage, H., Al-Kadri, M.O., Petrovski, A., Madzudzo, G., Cheah, M., 2023. AI-based intrusion detection systems for in-vehicle networks: A survey. ACM Comput. Surv. 55 (11), http://dx.doi.org/10.1145/3570954.

Rajapaksha, S., Rani, R., Karafili, E., 2024. A RAG-based question-answering solution for cyber-attack investigation and attribution. arXiv preprint arXiv:2408.06272.

Rupa, C., Srivastava, G., Bhattacharya, S., Reddy, P., Gadekallu, T.R., 2021. A machine learning driven threat intelligence system for malicious URL detection. In: Proc. of ARES 2021. ACM, pp. 1–7.

Saleem Raja, A., Vinodini, R., Kavitha, A., 2021. Lexical features based malicious URL detection using machine learning techniques. Mater. Today: Proc. 47, 163–166.

Saxe, J., Berlin, K., 2015. Deep neural network based malware detection using two dimensional binary program features. In: 2015 10th International Conference on Malicious and Unwanted Software. MALWARE, IEEE, pp. 11–20.

Schiavoni, S., Maggi, F., Cavallaro, L., Zanero, S., 2014. Phoenix: DGA-based botnet tracking and intelligence. In: Dietrich, S. (Ed.), Detection of Intrusions and Malware, and Vulnerability Assessment. Springer International Publishing, Cham, pp. 192–211.

Senanayake, J., Kalutarage, H., Al-Kadri, M.O., 2021. Android mobile malware detection using machine learning: A systematic review. Electronics 10 (13), 1606. http://dx.doi.org/10.3390/electronics10131606, URL https://www.mdpi.com/2079-9292/10/13/1606.

Senanayake, J., Kalutarage, H., Al-Kadri, M.O., Petrovski, A., Piras, L., 2023. Android source code vulnerability detection: A systematic literature review. ACM Comput. Surv. 55 (9), http://dx.doi.org/10.1145/3556974.

Senanayake, J., Rajapaksha, S., Yanai, N., Komiya, C., Kalutarage, H., 2024. MADONNA: Browser-based malicious domain detection through optimized neural network with feature analysis. In: Meyer, N., Grocholewska-Czuryło, A. (Eds.), ICT Systems Security and Privacy Protection. Springer Nature Switzerland, Cham, pp. 279–292.

Shabudin, S., Sani, N.S., Ariffin, K.A.Z., Aliff, M., 2020. Feature selection for phishing website classification. Int. J. Adv. Comput. Sci. Appl. 11 (4).

Shanahan, M., 2024. Talking about large language models. Commun. ACM 67 (2), 68–79.

Shi, Y., Chen, G., Li, J., 2018. Malicious domain name detection based on extreme machine learning. Neural Process. Lett. 48 (3), 1347–1357.

Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., Zhang, K., Cai, S., Nielsen, E., Soergel, D., Bileschi, S., Terry, M., Nicholson, C., Gupta, S.N., Sirajuddin, S., Sculley, D., Monga, R., Corrado, G., Viégas, F.B., Wattenberg, M., 2019. TensorFlow.js: Machine learning for the web and beyond. http://dx.doi.org/10.48550/ARXIV.1901.05350, arXiv. URL https://arxiv.org/abs/1901.05350.

Sood, A.K., Enbody, R.J., 2011. Spying on the browser: dissecting the design of malicious extensions. Netw. Secur. 2011 (5), 8–12. http://dx.doi.org/10.1016/S1353-4858(11)70050-2, URL https://www.sciencedirect.com/science/article/pii/S1353485811700502.

Sun, X., Tong, M., Yang, J., Xinran, L., Heng, L., 2019. HinDom: A robust malicious domain detection system based on heterogeneous information network with transductive classification. In: Proc. of RAID 2019. USENIX Association, pp. 399–412.

Sun, X., Yang, J., Wang, Z., Liu, H., 2020. HGDom: Heterogeneous graph convolutional networks for malicious domain detection. In: Proc. of NOMS 2020. IEEE, pp. 1–9.
Tang, L., Mahmoud, Q.H., 2021. A survey of machine learning-based solutions for phishing website detection. Mach. Learn. Knowl. Extr. 3 (3), 672–694.
Vadera, S., Ameen, S., 2022. Methods for pruning deep neural networks. IEEE Access 10, 63280–63300. http://dx.doi.org/10.1109/ACCESS.2022.3182659.
Vinayakumar, R., Soman, K., Poornachandran, P., 2018. Detecting malicious domain names using deep learning approaches at scale. J. Intell. Fuzzy Systems 34 (3), 1355–1367.
Yadav, S., Reddy, A.K.K., Reddy, A.L.N., Ranjan, S., 2012. Detecting algorithmically generated domain-flux attacks with DNS traffic analysis. IEEE/ACM Trans. Netw. 20 (5), 1663–1677. http://dx.doi.org/10.1109/TNET.2012.2184552.
Yahya, F., W Mahibol, R.I., Ying, C.K., Anai, M.B., Frankie, S.A., Nin Wei, E.L., Utomo, R.G., 2021. Detection of phising websites using machine learning approaches. In: Proc. of ICoDSA 2021. IEEE, pp. 40–47.
Yang, L., Liu, G., Dai, Y., Wang, J., Zhai, J., 2020. Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework. IEEE Access 8, 82876–82889.
Yu, B., Pan, J., Hu, J., Nascimento, A., De Cock, M., 2018a. Character level based detection of DGA domain names. In: Proc. of IJCNN 2018. IEEE, pp. 1–8.
Yu, T., Zhauniarovich, Y., Khalil, I., Dacier, M., 2018b. A survey on malicious domains detection through DNS data analysis. ACM Comput. Surv. 51 (4).
Zabihimayvan, M., Doran, D., 2019. Fuzzy rough set feature selection to enhance phishing attack detection. In: Proc. of FUZZ-IEEE 2019. IEEE, pp. 1–6. http://dx.doi.org/10.1109/FUZZ-IEEE.2019.8858884.
Zamir, A., Khan, H.U., Iqbal, T., Yousaf, N., Aslam, F., Anjum, A., Hamdani, M., 2020. Phishing web site detection using diverse machine learning algorithms. Electron. Libr. 38 (1), 65–80.
Zhao, H., Chang, Z., Bao, G., Zeng, X., 2019. Malicious domain names detection algorithm based on *N*-Gram. J. Comput. Netw. Commun. 2019, 1–9.
Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al., 2023. A survey of large language models. arXiv:2303.18223.

Janaka Senanayake is a Lecturer in Cybersecurity at the School of Computing, Engineering and Technology at Robert Gordon University, United Kingdom. His main focus in research is software security, malware analysis, and web security. Previously, he worked as a lecturer in several foreign universities and as a software engineer and researcher in various international companies.

Sampath Rajapaksha is an AI/ML Engineer at Katoni Engineering, United Kingdom and also a KTP associate in the School of Computing, Engineering and Technology at Robert Gordon University, United Kingdom. His main expertise is in security in large language models, data engineering and automotive security. Previously he worked as a Data engineer and software engineer in various international companies.

Naoto Yanai is an Associate Professor of the Information Security Engineering Laboratory at Osaka University, Japan. His research interests are information security, especially cryptography, network security, blockchain, and machine learning applications. He worked at several international companies prior to joining Osaka University.

Harsha Kalutarage is is an Associate Professor in Cyber Security and the Cybersecurity Research Lead at the School of Computing, Engineering, and Technology at Robert Gordon University, United Kingdom. Previously, he was a Senior Research Engineer (R&D) specializing in Security Data Analytics at the Centre for Secure Information Technologies (CSIT) at Queen's University Belfast. Prior to that, he worked as a postdoctoral researcher on the ACiD project at Coventry University. His research focuses on the intersection of Artificial Intelligence (AI) and Cyber Security, with a particular emphasis on leveraging AI techniques for security applications, such as IoT and Cyber-Physical Systems, as well as enhancing the security of AI-embedded systems by analyzing vulnerabilities in AI algorithms.

Chika Komiya is a researcher at Osaka University, Japan, who specializes in web-based security. Prior to joining Osaka University, he worked for several information security companies.