Journal of Information Security and Applications 93 (2025) 104176

Contents lists available at ScienceDirect

Journal of Information Security and Applications

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

The machines are watching: Exploring the potential of Large Language Models for detecting Algorithmically Generated Domains

Tomás Pelayo-Benedet [a], Ricardo J. Rodríguez [a,*], Carlos H. Gañán [b]

[a] Dpto. de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Spain
[b] Delft University of Technology, The Netherlands

ARTICLE INFO

Keywords:
Large Language Models
Algorithmically Generated Domains
DNS traffic analysis
Malware detection

ABSTRACT

Algorithmically Generated Domains (AGDs) are integral to many modern malware campaigns, allowing adversaries to establish resilient command and control channels. While machine learning techniques are increasingly employed to detect AGDs, the potential of Large Language Models (LLMs) in this domain remains largely underexplored. In this paper, we examine the ability of nine commercial LLMs to identify malicious AGDs, without parameter tuning or domain-specific training. We evaluate zero-shot approaches and few-shot learning approaches, using minimal labeled examples and diverse datasets with multiple prompt strategies. Our results show that certain LLMs can achieve detection accuracy between 77.3% and 89.3%. In a 10-shot classification setting, the largest models excel at distinguishing between malware families, particularly those employing hash-based generation schemes, underscoring the promise of LLMs for advanced threat detection. However, significant limitations arise when these models encounter real-world DNS traffic. Performance degradation on benign but structurally suspect domains highlights the risk of false positives in operational environments. This shortcoming has real-world consequences for security practitioners, given the need to avoid erroneous domain blocking that disrupt legitimate services. Our findings underscore the practicality of LLM-driven AGD detection, while emphasizing key areas where future research is needed (such as more robust warning design and model refinement) to ensure reliability in production environments.

1. Introduction

Cybercrime has transformed into a highly profitable and organized industry, surpassing even the global illegal drug trade in terms of revenue [1]. This surge in profitability has driven the development of increasingly sophisticated malware, driving the need for continued advancements in security measures for prevention, detection, and response. To address these ever-evolving threats, the MITRE ATT&CK framework [2] offers a comprehensive, dynamic model for understanding adversary behavior throughout the entire attack lifecycle [3]. By detailing the tactics, techniques, and procedures employed by attackers, it provides a clear map of how adversaries exploit vulnerabilities, move laterally, and achieve their objectives.

A key tactic in many cyberattacks is *Command and Control* (C&C), where adversaries establish and maintain communication with compromised systems to exert control and exfiltrate data. MITRE ATT&CK maps several C&C techniques, including exploiting legitimate communication channels or using encryption to evade detection. By identifying these techniques, defenders can recognize and disrupt malicious communication channels before attackers can further exploit compromised systems.

One such technique frequently employed by attackers is the use of Domain Generation Algorithms (DGAs). Rather than relying on a static list of IP addresses or domains, which are easily discovered and neutralized [4,5], DGAs dynamically generate domains for C&C traffic and other malicious purposes, including but not limited to C&C operations. While our research primarily focuses on their use in C&C infrastructures, these algorithms make it significantly more difficult for defenders to block, track, or take control of these communication channels. Using DGAs allows malware to check potentially thousands of domains for instructions, making it more resilient to efforts intended to disrupt communications. First implemented in the `Conflicker` worm [6], DGAs have proven highly effective at maintaining persistent communications between attackers and victims and remain a key tool in modern cyberattacks.

DGAs generate numerous *Algorithmically Generated Domains* (AGDs) using various pseudo-random techniques such as hash-based methods, dictionary-based approaches, and others [7]. Both malware and attacker share the same DGA and initial seed value. Once the attacker registers a single AGD and deploys their C&C server, the compromised

* Corresponding author.
E-mail addresses: tpelayo@unizar.es (T. Pelayo-Benedet), rjrodriguez@unizar.es (R.J. Rodríguez), C.HernandezGanan@tudelft.nl (C.H. Gañán).

https://doi.org/10.1016/j.jisa.2025.104176

Available online 11 August 2025

system continuously generates and attempts connections to new AGDs, searching for the one that matches the registered domain. In case the C&C server is exposed, the attacker can quickly register a new AGD and redeploy their infrastructure, allowing infected systems to reconnect to the new domain.

Detecting malicious AGDs has been a major area of research over the past 14 years [8–16]. Several detection approaches have been proposed, with deep learning methods emerging as particularly prominent in recent literature. A comprehensive review on this topic can be found in [17].

Despite this progress, there remains a gap in the literature regarding the use of Large Language Models (LLMs) for AGD detection. This work seeks to address this gap by assessing the inherent capabilities of LLMs to identify malicious AGDs without the need for domain-specific fine-tuning. Specifically, we evaluate and quantify the classification performance of a set of LLMs based solely on their base training, providing valuable insights into their pattern recognition capabilities in the context of cybersecurity and AGD detection.

It is important to distinguish between traditional fine-tuning, which modifies a model's internal parameters through additional training, and contextual (or in-context) learning, where guidance is provided through examples within the prompt without altering the model's weights. Our few-shot setup relies exclusively on contextual learning, preserving the pre-trained state of the evaluated LLMs.

Similarly, this work does not aim to explore advanced prompt engineering techniques, such as chain-of-thought prompt [18], iterative refinement [19], or self-consistency [20]. Instead, prompts are constructed using a structured, incremental methodology to evaluate how varying degrees of contextual information affect model performance. The goal is to establish a baseline characterization of LLM capabilities for AGD detection, rather than to optimize task-specific prompt strategies.

To further evaluate the potential of LLMs in this domain, we formulated a set of research questions that focus on three fundamental aspects:

(i) the inherent detection capabilities of the models,
(ii) their ability to distinguish between different malware families, and
(iii) their performance when faced with real-world domains.

These aspects are explored through the following research questions:

RQ1.- (RQ1.1) How effective are LLMs in detecting malicious AGDs based solely on domain name string analysis? (RQ1.2) What is the impact on classification accuracy when specific linguistic features are provided to LLMs to analyze in domain name strings? (See §5).

RQ2.- To what extent can LLMs distinguish between different malware families, considering examples of AGDs generated by each family? (See §6).

RQ3.- How do LLMs perform when evaluating real-world non-malicious domains that may share structural similarities with malicious AGDs? (See §7).

To answer these RQs, we tested nine different LLMs from four vendors. We created three distinct datasets, based on [7,21], and real-world DNS logs, and developed four unique prompting strategies using an iterative prompting approach [22]. Some preliminary results addressing RQ1 were previously presented in [23]. Here, we significantly extend that preliminary analysis by conducting a large-scale systematic evaluation of LLM for AGD detection and classification.

Our evaluation showed that LLMs were able to detect malicious domains with accuracy rates ranging from 77.3% to 89.3% without requiring task-specific fine-tuning. However, all models exhibited significant false positive rates. Even when we enhanced the prompts to provide guidance for analyzing lexical features, minimal improvements were observed compared to the basic prompting approach. This suggests that the limitations in performance are inherent to the models themselves rather than the prompting strategy.

Through convergence tests with 10,000 samples, we validated the appropriateness of our dataset size. Performance analysis showed a correlation between model size and runtime, while confidence analysis revealed systematic biases favoring malicious classifications.

For multiclass classification of malicious AGDs by malware family, we found that 10-shot learning was the most effective, striking a balance between accuracy and processing efficiency. Using this approach, we evaluated LLMs ability to detect and categorize malicious domains. Performance varied significantly depending on the DGA generation scheme. Models performed almost perfectly on hash-based DGAs, but struggled with dictionary-based DGAs. Notably, larger models generally outperformed smaller ones, although half of the models failed to achieve an F1 score above 40%.

Using real-world DNS traffic and an optimized prompt, we tested the effectiveness of LLMs in more practical deployment scenarios. Despite providing more specific contextual guidance, most models showed substantial performance degradation (ranging from 11% to 24%) compared to their baseline results. This indicates critical challenges in classifying legitimate domains with AGD-like characteristics. While improved prompt engineering can improve performance, these results highlight the fundamental difficulties of deploying LLMs for AGD detection in production environments.

In summary, this paper makes the following contributions:

- We present the first systematic evaluation of LLM capabilities for detecting malicious AGDs in binary and multiclass classification setups. Our work introduces the use of zero-shot and few-shot learning for this task without the need for model tuning.
- We investigate multiple prompt engineering strategies for AGD detection and quantify their effectiveness and impact on classification performance.
- We analyze the effect of the number of in-context examples on the performance of few-shot learning for malware family classification. Our experiments demonstrate that model accuracy is sensitive to the amount of guidance provided.
- We identify a key limitation in the models' ability to distinguish benign domains under realistic conditions, especially when those domains exhibit DGA-like structural characteristics.

Unlike traditional DGA detection approaches, which rely on specialized models for specific tasks (e.g., convolutional neural networks [13] or long short-term memory models trained on DGA datasets [12]), our work explores whether general-purpose LLMs can detect malicious domains using only their pre-trained pattern recognition capabilities. We do not aim to outperform existing detection models, but rather to establish a performance baseline that reveals the extent to which LLMs exhibit security-relevant implicit behavior. This perspective highlights the potential of LLMs as complementary tools in security workflows and motivates further research on their applicability and limitations in threat detection tasks.

This paper is organized as follows. Section 2 provides background on relevant concepts. Section 3 discusses related work. Section 4 details the experimental setup, including dataset construction and prompt design. Sections 5–7 present the experimental results for each RQ. Section 8 discusses the limitations of his work. Finally, Section 9 concludes the paper and outlines directions for future research.

2. Background

This section describes the fundamental concepts related to DGAs, few-shot learning, and prompt engineering, which are relevant to our work.

2.1. DGA fundamentals

A *Domain Generation Algorithm* (DGA) is a mechanism that generates domain names using pseudo-random logic, typically initialized with a shared value to ensure output consistency across instances. This synchronization allows malware and its C&C infrastructure to keep in contact. The resulting domains, known as *Algorithmically Generated Domains* (AGDs), are central to many modern malware families [2].

DGAs differ in the mechanisms used to generate domains [7]. Common approaches include *arithmetic methods* that convert computed values into ASCII characters, *hash methods* that leverage cryptographic functions, *dictionary-based strategies* that concatenate human-readable words, and *permutation-based schemes* that reorder characters from known domains. These variations produce distinctive domain characteristics; for example, dictionary-based DGAs typically produce linguistically plausible domains, while hash- and arithmetic-based DGAs typically generate random-looking sequences that are harder to remember.

2.2. Few-shot learning

Few-shot learning (FSL) is a machine-learning paradigm in which models generalize to new tasks using only a small number of labeled examples [24]. This approach leverages prior knowledge to reduce both training data requirements and computational overhead, making it suitable for dynamic or resource-constrained environments (e.g., an IoT device, a router, or an edge computing node) [25].

LLMs inherently support FSL through context-based learning [26], where examples are integrated directly into prompts rather than being used to modify model parameters [27]. Pre-training on a variety of tasks allows these models to quickly adapt to the specific task context. Previous work shows that performance typically saturates after 10–20 examples for very large models like GPT-3 [26], while mid-scale models in the 7B–13B parameter range perform optimally with 40–48 examples [28].

2.3. Prompt engineering

Prompt engineering refers to the practice of creating input prompts that effectively guide LLMs to perform a desired task without altering its parameters [22]. A well-constructed prompt typically includes a clear task description, sufficient contextual information, and, where appropriate, illustrative reasoning steps [29]. This allows LLMs to leverage their prior knowledge while adapting to specific scenarios [30].

Designing effective prompts requires understanding the model's behavior and limitations. This typically involves defining task objectives, selecting an appropriate structure or format, integrating relevant examples or context, and refining the prompt through iterative testing. In this work, we use prompt engineering to structure inputs that allow LLMs to perform AGD detection without any parameter tuning, relying solely on their pre-trained knowledge and the contextual guidance provided through carefully designed prompts.

3. Related work

The detection of algorithmically generated domains (AGDs) remains a challenge in cybersecurity due to the ongoing evolution of domain generation algorithms (DGAs). This ongoing competition has driven the transition from traditional statistical techniques to increasingly sophisticated machine learning-based approaches [7].

Early systems such as `EXPOSURE` [8,9] laid the groundwork for passive DNS-based detection, combining lexical and temporal features. `Pleiades` [10] leveraged failed DNS queries to identify AGDs, while `Phoenix` [11] extended detection to classify malware families using domain- and IP-level features.

The application of neural networks marked a significant advance in AGD detection. Woodbridge et al. [12] introduced LSTMs for binary and multiclass AGD classification. Subsequent work by Yu et al. [13] employed convolutional neural networks (CNNs) to capture local patterns in domain strings. More recent models, such as `Deep Bot Detect` [14], demonstrated competitive accuracy at lower computational cost, while Berman [15] proposed a one-dimensional capsule network to address the limitations of CNN-based architectures.

Recent efforts have focused on improving robustness and generalization. Drichel et al. [16] applied transfer learning to improve the detection of novel AGD patterns. Similarly, Cebere et al. [17] conducted a meta-analysis that identified persistent limitations in the current detection landscape, including overfitting to known families and low adaptivity. Dataset selection also plays an important role in detection performance. While Alexa [31] was historically used for benign domain baselines, it has been widely replaced by Tranco List [21], which offers greater stability and methodological rigor[1]. Malicious AGDs often come from datasets such as Bambenek [33], 360NetLab [34], the Domain Generation Algorithms Repository [35], and DGArchive [7]. In this work, we adopted the latter due to its sample diversity.

A major ongoing challenge is the emergence of new DGA families, which often require periodic model retraining to maintain performance. This has driven interest in architectures with greater generalization capabilities that can adapt to previously unseen DGAs.

In this sense, *Large Language Models* (LLMs) represent a promising alternative for DGA detection thanks to their robust pattern recognition and language understanding capabilities [36]. Unlike traditional models, which require extensive training for each task, LLMs can parse domain names using knowledge acquired during pre-training, without the need for additional fine-tuning. By using carefully designed prompts, LLMs can be reused for DGA detection, potentially enabling more flexible and generalizable detection of both known and emerging threats [37].

4. Experimental setup

We utilized LLM APIs from third-party providers for two primary reasons. First, most large models are not available for local deployment, restricting self-hosted options. Second, the few open-source models that do exist require significant computational resources, making local operation impractical (as discussed in Section 8).

Based on preliminary testing and the constraints of the chosen LLM constraints, we limited each API request to 125 domains. All experiments were automated using Python 3.11.2, and our source code is available on GitHub.[2]

LLM selection. We evaluated nine LLMs from four leading vendors, spanning both free and paid service offerings. Our selection criteria emphasized models with publicly available APIs and documented capabilities in natural language understanding and generation. For paid models, we also took into account a cost–benefit analysis, prioritizing competitively priced solutions. Cost considerations were particularly relevant given the large number of API calls and token consumption required for our experiments. The selected models are listed in Table 1.

Among the paid models, we evaluated two main providers: Anthropic [38] and OpenAI [39]. From Anthropic's Claude family, we selected `Sonnet 3.5` and `Haiku 3.5`; from OpenAI, we used both `GPT-4o` and `GPT-4o-mini`. For the free models, we relied on the offerings from Google [40] and MistralAI [41]. Although both providers offer paid tiers, their free versions met the requirements of our experiments. From Google, we used `Gemini 1.5 Pro` (with usage

[1] While Tranco improves on previous classifications in terms of reproducibility and representativeness, it may still contain malicious domains, as pointed out by Pochat et al. [32].

[2] https://github.com/reverseame/LLM-DGA-lab

Table 1
Overview of evaluated LLMs. The 'Context Window' and 'Output Limit' values indicate token capacities provided by the respective APIs.

Model	Context window	Output limit	Release	Tier	API Tag
GPT-4o[a]	128,000	16,384	Nov'24	Paid	gpt-4o-2024-11-20
GPT-4o-mini[b]	128,000	16,384	Jul'24	Paid	gpt-4o-mini-2024-07-18
Claude Sonnet 3.5[a]	200,000	8,192	Oct'24	Paid	claude-3-5-sonnet-20241022
Claude Haiku 3.5[b]	200,000	8,192	Oct'24	Paid	claude-3-5-haiku-20241022
Gemini 1.5 Pro[a]	2,097,152	8,192	Sep'24	Free	gemini-1.5-pro-002
Gemini 1.5 Flash[b]	1,048,576	8,192	Sep'24	Free	gemini-1.5-flash-002
Gemini 1.5 Flash-8B[b]	1,048,576	8,192	Oct'24	Free	gemini-1.5-flash-8b-001
Mistral Large[a]	131,000	N/A	Nov'24	Free	mistral-large-2411
Mistral Small[b]	32,000	N/A	Sep'24	Free	mistral-small-2409

[a] indicates larger models.
[b] indicates smaller models counterparts.

constraints), Gemini 1.5 Flash, and Gemini 1.5 Flash-8, while from MistralAI, we used Mistral Large and Mistral Small. Several others models were excluded from this work due to limitations detailed in Section 8. However, the variety among the selected models was sufficient diverse to ensure diverse coverage for our analysis.

While all models participated in our performance benchmark, we restricted parameter tuning and rapid engineering experiments to Gemini 1.5 Flash, Gemini 1.5 Flash-8B, Mistral Large, and Mistral Small. This selective approach was due to both budget and time constraints, given the high volume of API requests required. The combination of comprehensive model coverage for overall testing and optimization targeting a subset of models allowed us to gain robust insights from multiple vendors.

Because LLMs are frequently updated (often without explicit notification), our experiments were conducted between December 2024 through early January 2025. While these results offer valuable insights into the capabilities of each model, they should be interpreted within this time frame. Future evaluations may yield different results as vendors continue to develop their models.

Construction of datasets. Three datasets were constructed for the experiments. The first dataset, D_1, contains 25,000 malicious domains generated by DGAs and 25,000 non-malicious domains. The second dataset, D_2, includes 50,000 malicious domains generated by DGAs from 25 different malware families, with an equal number of domains per family. Finally, the third dataset, D_3, compromises 50,000 non-malicious domains extracted from the University of Zaragoza DNS server logs spanning from June 2023 to May 2024 (347 days in total).

We used D_1 to evaluate LLMs' AGD detection (addressing RQ1), D_2 to evaluate their ability to classify malware families RQ2, and D_3 to test performance on real-world domains RQ3.

The malicious AGDs in D_1 and D_2 are derived from DGArchive [7], a dataset comprising 137 malware families and variants using DGAs. To maintain both relevance and diversity, we selected 25 families that represent the full range of DGA types (see Section 2.1). This approach helps to avoid bias toward any particular generation scheme while preventing underrepresentation of others. However, we exclude permutation-based generation schemes due to the lack of sufficiently large families for meaningful experiments. The selected families are listed in Table 2. From each family, we randomly sampled 1,000 DGAs for D_1 and 2,000 DGAs for D_2, ensuring that the two datasets are disjoint.

To ensure the diversity and representativeness of our malicious domain dataset, we selected 25 malware families from DGArchive, a repository with 137 distinct DGA families, the most comprehensive available for this type of analysis. Our selection was based on the goal of covering the full range of generation techniques, including arithmetic, hashing, and dictionary algorithms. By balancing the families across these categories, we avoided overrepresentation of any single type. This sampling strategy captures the variability of DGA patterns present in real-world malware and reduces potential biases, allowing for a more robust and generalizable assessment.

Table 2
Selected malware families that utilize DGAs.

Malware family	AGD examples
ARITHMETIC-BASED	
banjori	nlgbpartbulkyf.com
conficker	tsdtjmgjvtv.biz
emotet	bqwpjpkujiaabouh.eu
flubot	nygrkfvksfadlmr.ru
gameover	kljinjhfqdynzbylayizx.ru
metastealer	maacykqieygsiemm.xyz
necurs	ngosrfurisqtsy.org
nymaim	nvnhmobqg.com
pitou	oqzoaaqay.biz
pushdo	nealilxotad.kz
qakbot	rakfqviujexuxyhpxd.com
rovnix	116wncabm8ai74q2al.com
virut	pyycva.com
zloader	seprfyswjugpvldkrwwg.com
DICTIONARY-BASED	
gozi	limitingcopyright.com
matsnu	accident-be-kind.com
nymaim2	reachesdarkness.am
suppobox	roomtomorrow.ru
HASH-BASED	
darkwatchman	9b93cf03.top
dyre	a000dc63f44247436a9b8558310bb48441.cc
grandoreiro	a40424003475944.servehalflife.com
monerominer	6604fafaf6f09.hosting
pandabanker	9337a3e6c511d.net
tinynuke	7f6fb68d7aac2de485ac1256503bb5c0.com
wd	wd405b87d8c2d634fa8252225af75a5781.pro

Non-malicious domains in D_1 were obtained from the Tranco list [21]. While the Tranco list is generally considered a reliable source, it may still contain malicious domains, as noted by [32].

We constructed the dataset D_3 using DNS resolution logs collected from the University of Zaragoza. This dataset complements Tranco's list by providing a localized view of domain resolutions in an academic setting. While Tranco reflects globally popular domains, university DNS records capture domain resolution behavior specific to educational institutions. The dataset spans 347 days, encompassing almost a full year of network activity and a wide range of legitimate domains.

Due to the high volume of DNS queries, we applied a rigorous filtering process to ensure only syntactically valid and resolvable domains were retained. This validation step was necessary to preserve data quality and eliminate malformed or transient entries that could affect the integrity of the evaluation. The primary filter imposed technical requirements:

(i) a maximum total length of 253 characters;
(ii) a leading character that is a letter, followed by letters, digits, or hyphens; and
(iii) length restrictions of 1 to 63 characters for each segment, with no leading or trailing hyphens.

The secondary filter addressed domain categorization using the Tranco list as a basis for identifying non-malicious domains. Specifically, domains whose second-level domain (SLD) and top-level domain (TLD)

combinations appear on the Tranco list were classified as non-malicious. We recognize that the Tranco list, while reliable, has limitations (e.g., the potential inclusion of malicious domains and the omission of legitimate ones) [32]. Therefore, domains that did not meet the technical criteria or Tranco-based verification were discarded.

After filtering, domains were clustered by their SLD+TLD pairs, as certain tuples, such as `googlesyndication.com`, `office.com`, and `cloudfront.net`, were found with disproportionately high frequency. To mitigate potential bias, we imposed a limit of 500 domains per tuples, ensuring that no single group represented more than 1% of the dataset. From the resulting set we randomly sampled 50,000 domains were randomly selected from the resulting pool. By combining globally representative domains from Tranco with locally observed domains from the university's DNS traffic, we constructed a balanced and realistic dataset, suitable for evaluating the ability of LLMs to distinguish SLDs from legitimate domains in practical environments.

Prompt construction. Four incremental prompts, named P_1 through P_4, were developed under the hypothesis that providing more contextual information could improve classification performance. We employ an iterative prompting approach [22], illustrated in Fig. 1. The construction of each prompt proceeds as follows:

(1) P_1 takes a minimal approach that presents the task and sets up a structured response format, requiring the domain name, its classification, and a confidence level.
(2) P_2 extends P_1 by adding guidance on lexical feature analysis, instructing the LLM to pay special attention to the various generation schemes that malware may employ to create comprehensive AGDs [7]. These lexical features include:
    (i) level of randomness,
    (ii) character frequency,
    (iii) digit-letter ratio,
    (iv) consonant-vowel ratio,
    (v) pronounceability,
    (vi) presence of meaningful words, and
    (vii) similarity to popular domains.
(3) P_3 further refines P_2 by providing 10 example domains for each of the 25 malware families in D_2, leveraging the FSL capabilities of LLMs [26]. All examples are disjoint from D_2 to preserve the integrity of the evaluation. P_3 also updates the output format, adding a new field for the malware family. Additionally, to handle classification uncertainty, the LLM can assign domains to an "Unknown Family" category or use "-" if a domain is considered benign.
(4) P_4 builds on P_2 by providing additional context about benign AGD-like domains in real-world scenarios, in particular subdomains of legitimate domains. The prompt specifically tells the LLM to focus on TLD and SLD tuples, helping to discriminate between malicious AGDs and legitimate subdomains that might appear algorithmically generated.

Our experimental design explores two learning paradigms: zero-shot learning (used in prompts P1, P2, and P4), where models receive task instructions without examples, and few-shot learning (used in P3), where models are guided by a limited set of labeled instances. In both cases, model parameters remain unchanged, allowing us to evaluate the pattern recognition capabilities acquired by the LLMs during pre-training.

Developing specialized prompts for each LLM can potentially yield more accurate results by leveraging the unique capabilities and features of each model [22]. However, we took a generalized approach by using the same prompts for all models. This decision reflects two key goals:

(i) while LLM-specific prompts may improve performance, they do not fundamentally alter a model's underlying ability to perform classification, and
(ii) a uniform prompt design ensures a fair comparative evaluation across different LLMs.

By maintaining uniform, generalized prompts, we can better assess the intrinsic strengths and limitations of each model, providing valuable insights while preserving a balanced comparison framework.

To ensure reproducibility, the full text of all four prompts (P1–P4) is included in Appendix. These prompts reflect the exact instructions and context used during the assessment, allowing our methodology to be accurately replicated.

Our prompt design seeks to enable systematic evaluation through gradual increases in contextual information, rather than exploring advanced prompt optimization techniques. While strategies such as using chain-of-thought prompting, iterative refinement, or multi-step reasoning could further improve performance, they are beyond the scope of this work. Instead, our methodology seeks to establish baseline capabilities through incremental design, providing a structured foundation for future work in advanced prompt engineering.

Batch size justification. We selected a batch size of 125 domains per API call to balance processing efficiency and output constraints. As shown in Figs. 2 and 3, increasing the batch size to this point yields substantial speed improvements (3–9 times) while maintaining reliable classification output. Beyond this threshold, efficiency gains plateau and error rates increase, especially for smaller models.

This value also represents a practical upper bound for smaller models such as `Mistral Small` and `Gemini Flash`, which cannot consistently process larger batches due to output token limitations. Standardizing the batch size to 125 domains ensures consistent evaluation conditions across models and maximizes performance within the implementation constraints at the time of this work.

Metrics. The metrics used in our work include accuracy (Acc), precision ($Prec$), recall (Rec), F1-score ($F1$), false positive rate (FPR), true positive rate (TPR), Matthew's correlation coefficient (MCC), and Cohen's Kappa coefficient (κ). Their definitions are given below:

$$Acc = \frac{TP+TN}{TP+TN+FP+FN}$$
$$Prec = \frac{TP}{TP+FP}$$
$$Rec = \frac{TP}{TP+FN}$$
$$F1 = 2 \cdot \frac{Prec \cdot Rec}{Prec + Rec}$$
$$FPR = \frac{FP}{FP+TN}$$
$$TPR = \frac{TP}{TP+FN}$$
$$MCC = \frac{TN \cdot TP - FN \cdot FP}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$
$$\kappa = \frac{p_o - p_e}{1 - p_e}, \text{where}$$
$$p_o = \frac{(TN+TP)}{(TN+FP+FN+TP)}$$
$$p_e = \frac{(TN+TP)(TN+FN)}{(TN+FP+FN+TP)^2} + \frac{(FN+TP)(FP+TP)}{(TN+FP+FN+TP)^2}$$

Here, True Negative (TN) and True Positive (TP) are the number of legitimate domains correctly classified as non-AGDs and AGDs correctly classified as AGDs, respectively. In contrast, False Negative (FN) and False Positive (FP) are AGDs misclassified as legitimate and legitimate domains misclassified as AGDs, respectively.

Accuracy, precision, recall, and F1-score measure the overall classification performance, while FPR and TPR capture the false and true positive rates across all classified domains, respectively. MCC assesses the classification quality, particularly for imbalanced datasets, by considering both positive and negative classes. Finally, κ indicates the degree to which the observed agreement exceeds chance, with values closer to 1 meaning higher agreement.

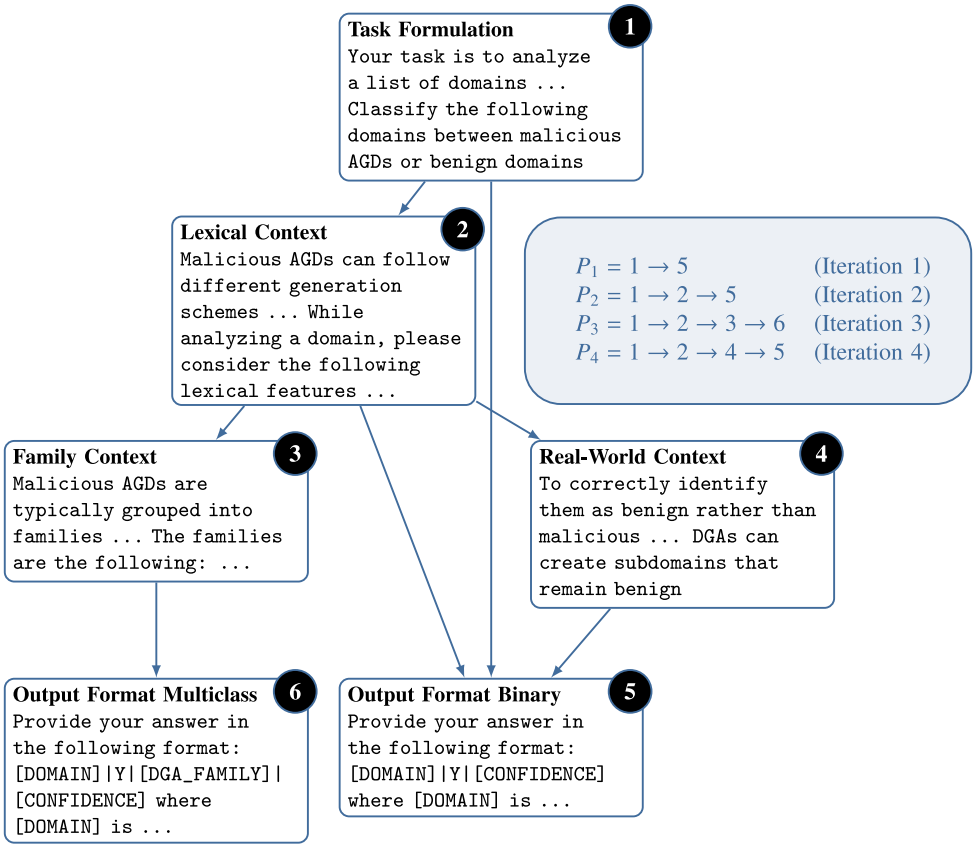


Fig. 1. Prompt construction scheme. The figure also shows which steps are used for each P_i prompt.

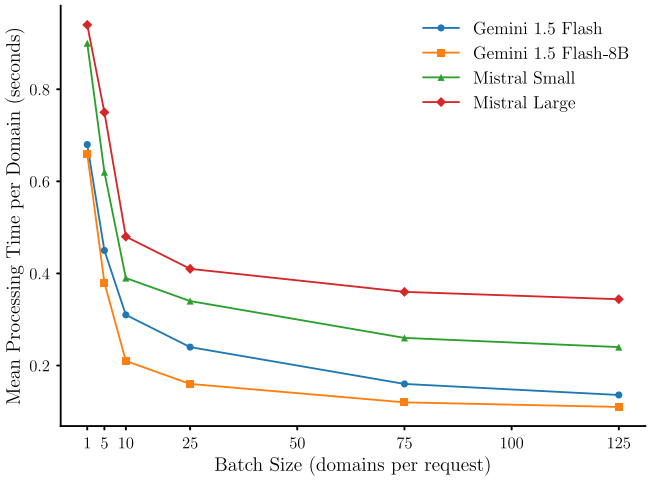


Fig. 2. Mean processing time per domain across different batch sizes using D_1 and P_1.

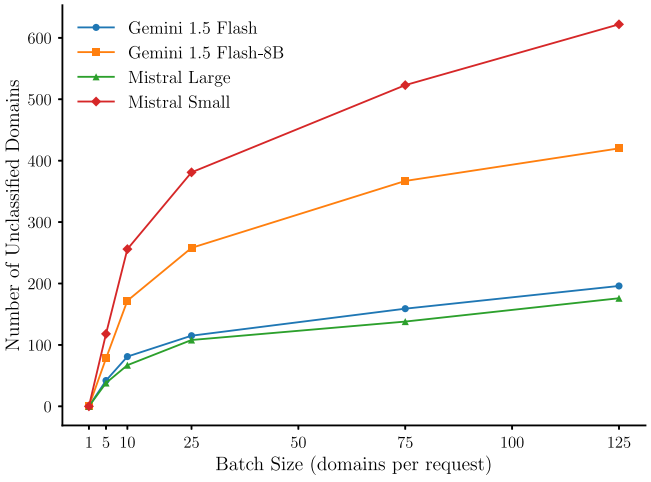


Fig. 3. Number of unclassified domains across different batch sizes using D_1 and P_1.

Statistical analysis. To ensure robust evaluation and address statistical validity, we report 95% confidence intervals for all performance metrics using the Wilson score interval [42]. This method was selected over standard normal approximation due to its higher performance with moderate sample sizes, particularly when observed proportions are near the extremes (i.e., close to 0 or 1). The Wilson interval provides more precise coverage of the nominal confidence level, making it ideal for our experimental setting with 50,000 domain samples and the range of accuracies observed across models.

5. Evaluating effectiveness of LLMs in detecting AGDs

In this section, we address RQ1 by analyzing the ability of LLMs to detect AGDs. First, we evaluate their ability to classify malicious

Table 3
Performance of LLMs with 95% confidence intervals (in parentheses) under two prompting strategies (P1 and P2) on the D1 dataset. Values are presented as point estimate (95% CI). Bold values highlighted in orange indicate the best performance across all models for each metric.

Model	P	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	FPR (%)	TPR (%)	AUC (%)	MCC (%)	Cohen's κ
GPT-4o	P1	86.8 (85.2–88.4)	83.6 (81.8–85.4)	91.4 (89.9–92.9)	87.3 (85.8–88.8)	17.9 (16.1–19.7)	91.4 (89.9–92.9)	86.8 (86.4–87.1)	73.8 (70.8–76.8)	0.603
	P2	87.0 (85.4–88.6)	84.6 (82.8–86.4)	90.5 (89.0–92.0)	87.4 (85.9–88.9)	16.5 (14.8–18.2)	90.5 (89.0–92.0)	87.0 (86.7–87.3)	74.2 (71.2–77.2)	0.603
GPT-4o-mini	P1	77.3 (75.4–79.2)	73.0 (70.9–75.1)	86.4 (84.6–88.2)	79.2 (77.4–81.0)	31.9 (29.7–34.1)	86.4 (84.6–88.2)	77.3 (76.8–77.7)	55.4 (51.8–59.0)	0.415
	P2	78.5 (76.6–80.4)	74.6 (72.5–76.7)	86.5 (84.7–88.3)	80.1 (78.3–81.9)	29.4 (27.3–31.5)	86.5 (84.7–88.3)	78.5 (78.1–78.9)	57.8 (54.3–61.3)	0.435
Claude 3.5 Sonnet	P1	**89.3 (87.9–90.7)**	83.8 (82.0–85.6)	**97.4 (96.5–98.3)**	**90.1 (88.7–91.5)**	18.8 (17.0–20.6)	**97.4 (96.5–98.3)**	**89.3 (89.0–89.6)**	**79.7 (77.0–82.4)**	**0.682**
	P2	**89.4 (88.0–90.8)**	84.2 (82.4–86.0)	96.8 (95.8–97.8)	**90.1 (88.7–91.5)**	18.2 (16.4–20.0)	96.8 (95.8–97.8)	**89.3 (89.0–89.6)**	79.5 (76.8–82.2)	0.678
Claude 3.5 Haiku	P1	85.6 (84.0–87.2)	84.0 (82.2–85.8)	87.9 (86.3–89.5)	85.9 (84.4–87.4)	16.8 (15.1–18.5)	87.9 (86.3–89.5)	85.6 (85.2–85.9)	71.2 (68.4–74.0)	0.563
	P2	85.2 (83.6–86.8)	84.7 (82.9–86.5)	86.0 (84.3–87.7)	85.4 (83.9–86.9)	15.6 (14.0–17.2)	86.0 (84.3–87.7)	85.2 (84.9–85.6)	70.5 (67.7–73.3)	0.548
Gemini 1.5 Pro	P1	87.7 (86.2–89.2)	83.8 (82.0–85.6)	93.5 (92.2–94.8)	88.4 (87.0–89.8)	18.1 (16.4–19.8)	93.5 (92.2–94.8)	87.7 (87.4–88.0)	76.0 (73.2–78.8)	0.632
	P2	87.6 (86.1–89.1)	84.2 (82.4–86.0)	92.6 (91.2–94.0)	88.2 (86.8–89.6)	17.4 (15.7–19.1)	92.6 (91.2–94.0)	87.6 (87.3–87.9)	75.6 (72.8–78.4)	0.625
Gemini 1.5 Flash	P1	84.8 (83.2–86.4)	83.5 (81.7–85.3)	86.9 (85.2–88.6)	85.1 (83.6–86.6)	17.2 (15.5–18.9)	86.9 (85.2–88.6)	84.8 (84.5–85.2)	69.7 (66.9–72.5)	0.544
	P2	84.9 (83.3–86.5)	83.6 (81.8–85.4)	86.8 (85.1–88.5)	85.2 (83.7–86.7)	17.1 (15.4–18.8)	86.8 (85.1–88.5)	84.9 (84.5–85.2)	69.8 (67.0–72.6)	0.545
Gemini 1.5 Flash-8B	P1	81.7 (79.9–83.5)	78.2 (76.2–80.2)	87.9 (86.2–89.6)	82.8 (81.1–84.5)	24.5 (22.5–26.5)	87.9 (86.2–89.6)	81.7 (81.3–82.1)	63.9 (60.8–67.0)	0.494
	P2	82.7 (81.0–84.4)	79.8 (77.8–81.8)	87.6 (85.9–89.3)	83.5 (81.9–85.1)	22.1 (20.2–24.0)	87.6 (85.9–89.3)	82.7 (82.4–83.1)	65.8 (62.8–68.8)	0.510
Mistral Large	P1	88.7 (87.2–90.2)	**87.3 (85.7–88.9)**	90.6 (89.0–92.2)	88.9 (87.5–90.3)	**13.2 (11.7–14.7)**	90.6 (89.0–92.2)	88.7 (88.4–89.0)	77.4 (74.6–80.2)	0.639
	P2	88.5 (87.0–90.0)	**87.1 (85.5–88.7)**	90.5 (88.9–92.1)	88.8 (87.4–90.2)	**13.4 (11.9–14.9)**	90.5 (88.9–92.1)	88.5 (88.2–88.8)	77.1 (74.3–79.9)	0.636
Mistral Small	P1	85.1 (83.5–86.7)	82.6 (80.8–84.4)	89.0 (87.4–90.6)	85.7 (84.2–87.2)	18.8 (17.1–20.5)	89.0 (87.4–90.6)	85.1 (84.8–85.5)	70.4 (67.6–73.2)	0.560
	P2	85.5 (83.9–87.1)	83.7 (81.9–85.5)	88.1 (86.4–89.8)	85.8 (84.3–87.3)	17.1 (15.4–18.8)	88.1 (86.4–89.8)	85.5 (85.1–85.8)	71.1 (68.3–73.9)	0.562

P: Prompt; Acc: Accuracy; Prec: Precision; Rec: Recall; F1: F1-score; FPR: False Positive Rate; TPR: True Positive Rate; AUC: Area Under the Curve; MCC: Matthews's Correlation Coefficient; κ: Cohen's Kappa Score.

domains using D_1 together with the minimal P_1 prompt. Next, we investigate whether providing domain-specific knowledge of the linguistic features of AGDs (i.e., lexical cues) improves detection performance, using the same D_1 dataset, but with the P_2 prompt for improved context.

5.1. LLM performance evaluation

Table 3 presents the performance metrics for both prompting strategies (P_1 and P_2). Without task-specific tuning or example-based training, the LLMs evaluated in this work achieve accuracy, precision, recall, and F1-scores ranging from 77.3% and 97.4%, a remarkable benchmark. Furthermore, AUC values range from 78.5% to 89.4%, demonstrating the models' ability to distinguish between benign and malicious domains. Particularly significant are the MCC values (55.4% to 79.7%) and κ scores (0.41 to 0.68), indicating substantial agreement between model predictions and true labels across all LLMs evaluated.

`Claude 3.5 Sonnet` demonstrated superior performance across most metrics, albeit only by slight margins over competitors such as `Mistral Large` in certain cases. Notably, it obtained the highest AUC values, highlighting its superior discriminatory ability to distinguish between malicious and benign domains. A clear trend emerges across all models: those with larger amounts of parameters generally perform better, as evidenced by the strongest results of `GPT-4o`, `Claude Sonnet`, `Gemini Pro`, and `Mistral Large` compared to their smaller counterparts.

The ROC analysis in Fig. 4 reinforces the superior discriminative performance observed in the standard classification metrics. `Claude 3.5 Sonnet` exhibits the best overall ability to distinguish between classes in both prompting strategies, consistently achieving ROC curves close to the ideal upper-left corner and reaching AUC values as high as 89.3%. The performance difference between the largest and smallest models is clearly reflected in both the curve profiles and the corresponding AUC scores.

Behavior at low false positive rate thresholds (FPR < 0.1) is particularly relevant in deployment environments, where false positives can lead to unintended disruption of legitimate services. In this region, `Claude 3.5 Sonnet` and `Mistral Large` maintain high TPR while minimizing FP, making them more suitable for production environments where legitimate domain blocking must be minimized. In contrast, `GPT-4o-mini` and `Gemini 1.5 Flash-8B` show more gradual ROC slopes, indicating reduced effectiveness under strict FP constraints.

Despite these successes, a notable concern is the consistently high FPR across all models. This limitation poses a significant challenge for real-world deployment, as legitimate domains would frequently be misclassified as malicious, potentially leading to excessive connection blocks. Even `Mistral Large`, which achieved the best FPR, still exhibits a rate that could cause considerable disruptions in production environments.

Furthermore, the accuracy comparison in Fig. 5 reveals a systematic bias toward detecting malicious domains. Accuracy differences between malicious and benign domains can reach up to 16%, highlighting a critical imbalance that underscores the need for further refinement before these models can be reliably deployed.

Experimental results indicate that LLMs generally perform well across all metrics. However, the high false positive rates in each model pose a critical barrier to real-world deployment. Furthermore, systematic bias favoring malicious domain detection at the expense of benign accuracy further limits their practical applicability in production environments.

To assess the practical implications of the observed FPRs, consider a deployment scenario involving the classification of one million legitimate domain queries per day. Given FPRs ranging from 13.2% to 31.9% across the evaluated models, this would result in the incorrect flagging and potential blocking of between 132,000 to 319,000 benign domains each day. These misclassifications would significantly disrupt routine network operations, demonstrating that these models, in their current form, are not suitable for direct deployment in production environments without additional filtering, post-processing, or risk mitigation mechanisms.

5.2. Simple vs. Advanced prompt engineering

Comparison of the prompting strategies (P_1 and P_2), as shown in Table 3, reveals minimal impact on model performance across all evaluated LLMs. In most cases, metric variations between the two approaches remained below 2 percentage points, suggesting that a more elaborated prompt incorporating fine-grained linguistic features may not be strictly necessary for this particular classification task.

Confidence interval analysis provides statistical support for these findings. In most cases, the 95% confidence intervals (CI) for performance under P1 and P2 exhibit substantial overlap, indicating that the observed differences are within the bounds of expected variation. For instance, `GPT-4o` shows and increase in accuracy from 86.8% (CI: 85.2–88.4%) with P1 to 87.0% (CI: 85.4–88.6%) with P2. The overlap of intervals suggests that this change is not statistically significant.

Although P_2 consistently reduces FPR relative to P_1 (see Fig. 5), this improvement comes at the expense of lower malicious detection accuracy. Moreover, the marginal gains in FPR are still insufficient to address practical deployment challenges in real-world settings. Statistical analysis supports this trend: while P_2 reduces the FPR in all evaluated models (e.g., from 17.9% to 16.5% in the case of `GPT-4o`), the overlapping 95% CI suggest these improvements are within

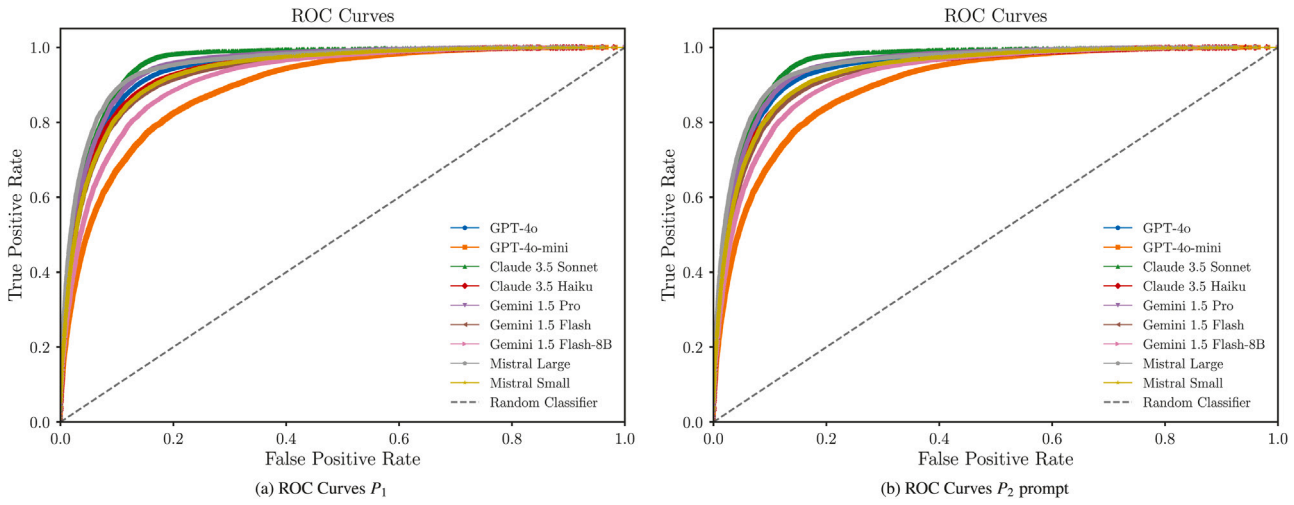


Fig. 4. ROC curves comparing the performance of LLMs in detecting AGDs under the P_1 and P_2 prompting strategies. Curves closer to the top left corner indicate higher discriminative performance.

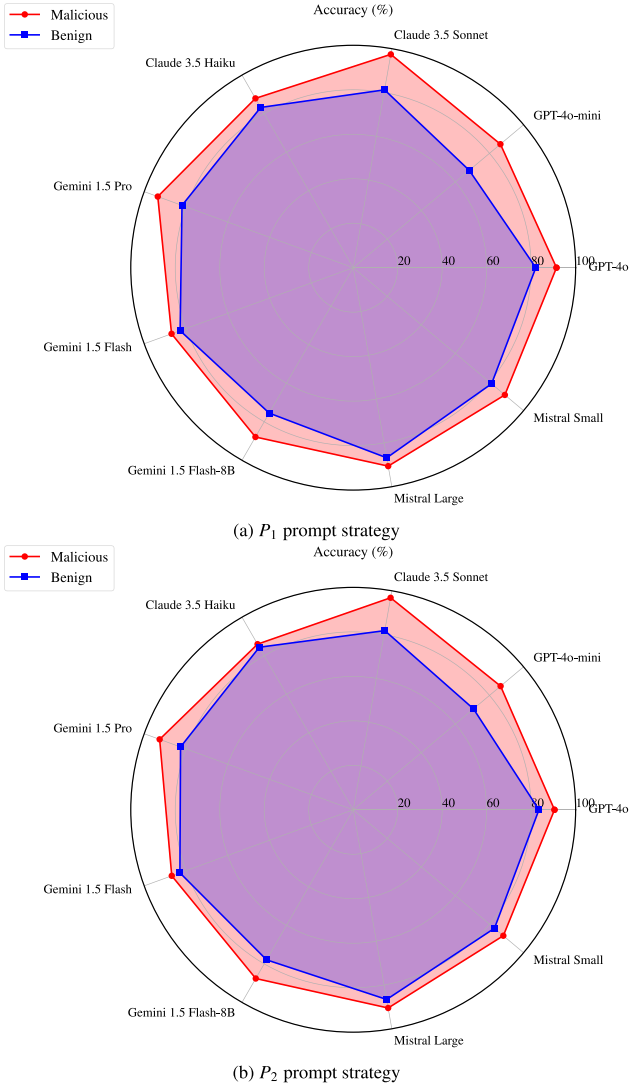


Fig. 5. Accuracy of each LLM in classifying malicious (red) versus benign (blue) domains under the two prompt strategies, P_1 (top) and P_2 (bottom). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

statistical uncertainty. This reinforces our conclusion that while more informative prompts may slightly reduce false positives, the observed improvements are marginal and insufficient to overcome the limitations of practical implementation.

Consequently, our incremental prompt engineering approach should be considered a fundamental investigation into the role of contextual information in AGD detection. While this approach reveals how prompt structure influences model behavior, it is not intended to exhaust the spectrum of advanced prompting techniques. Methods such as chain-of-thought reasoning, multi-step prompting, or iterative refinement may offer greater improvements, but are beyond the scope of this work, which focuses on establishing baseline LLM performance using controlled prompt complexity.

5.3. Dataset size validation

Our evaluation dataset comprises 50,000 domains, evenly split between malicious AGDs and legitimate domains (25,000 each). Although some deep learning approaches leverage hundreds of thousands of samples, we consider this dataset sufficient to evaluate the performance of LLM for domain classification. As illustrated in Fig. 6, the F1-scores for all evaluated models converge at approximately 10,000 samples, with negligible changes up to the full 50,000 samples. Notably, both P_1 and P_2 follow a similar convergence trend, indicating that increasing the dataset size beyond this point would likely not yield significant performance gains. This consistent behavior across all models suggests that further scaling of the dataset would provide only marginal benefits for LLM-based domain classification.

5.4. Time execution analysis

Table 4 summarizes the execution time performance for both prompting strategies (P_1 and P_2) across all evaluated LLMs. We report a comprehensive set of timing metrics – including mean, variance, standard deviation, median, minimum, and maximum execution times – to provide a detailed overview of how each models processes requests. Fig. 7 complements these data by illustrating their distribution through box plots, highlighting variability and outliers in each model's performance.

When comparing mean and median execution times across the LLMs, a clear trend emerges: larger models typically require more processing time than their smaller counterparts. Within the same provider, model size strongly correlates with higher computational overhead. A notable exception is found in the Claude family, where both models exhibit similar mean and median execution times despite different sizes.

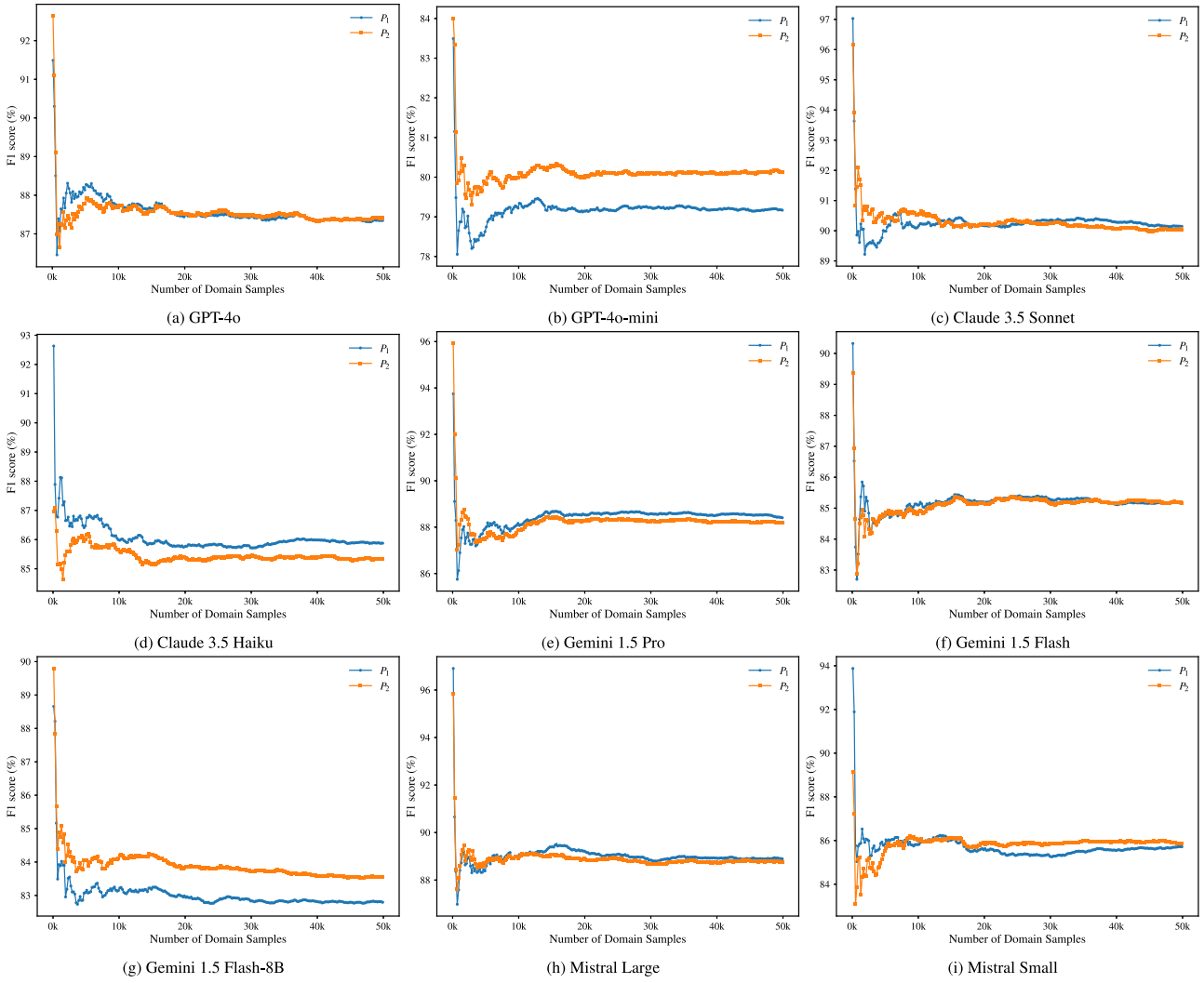


Fig. 6. Convergence analysis of F1-scores for LLMs under prompting strategies P_1 and P_2 using D_1.

Comparing P_1 and P_2 shows a consistent pattern: P_2 generally increases mean and median execution times, suggesting that more sophisticated prompt engineering systematically demands additional processing, rather than simply being influenced by sporadic outliers or network fluctuations.

However, exceptions do occur—most prominently with `Gemini 1.5 Pro`, which experiences a counter intuitive decrease in mean execution time when moving from P_1 to P_2. This discrepancy could stem from various factors, including service load, network conditions, or backend resource allocation. The high variance observed in P_1 for this model, clearly visible in Fig. 7, indicates that such anomalies may reflect service instability during testing rather than any inherent difference between the two prompts. Similar high-variance behavior in other models (e.g., `Mistral` or `GPT-4o`) underscores the importance of external factors in measured performance.

In summary, these findings confirm a trade-off between model size and execution time, with larger models generally requiring more computational resources. While P_2 introduces a modest yet systematic increase in processing time, this overhead can be exacerbated by fluctuating service stability. As a result, evaluating real-world LLM performance must account for both the computational cost of sophisticated prompts and the potential impact of environmental variables.

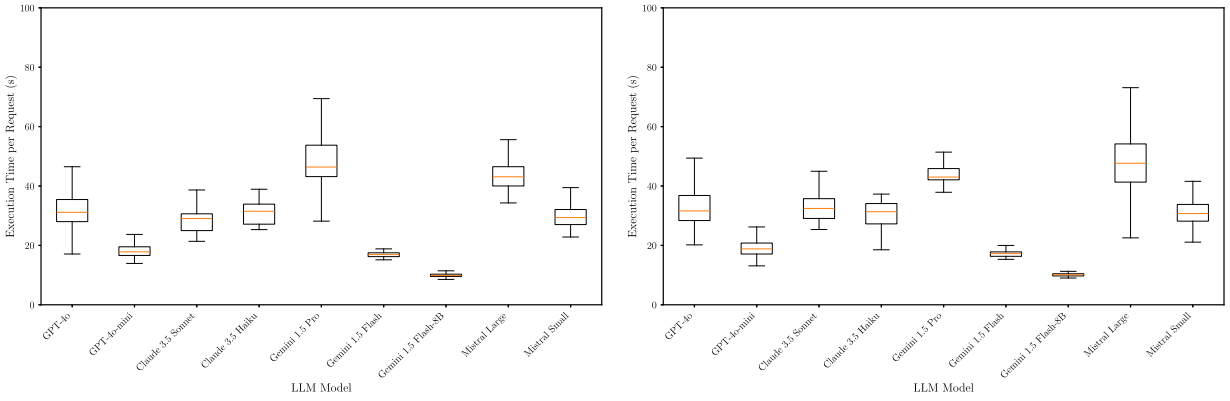
5.5. Confidence analysis

Fig. 8 shows that all models consistently report high confidence values, with medians typically above 85% for both malicious and benign classifications. Notably, a systematic bias emerges: models show higher confidence in labeling domains as malicious compared to benign, suggesting an inherent bias toward malicious classification. The evaluated models also demonstrate a correlation between confidence and accuracy, reporting higher confidence for accurate predictions and lower confidence for misclassifications.

Closer inspection of the boxplots reveals a number of low confidence outliers (below 50%), particularly in malicious domain classifications, most prominently in `GPT-4o-mini`. These low confidence cases occur for both correct and incorrect predictions, indicating a specific pattern of uncertainty in malicious domain detection. However, these outliers represent only a small fraction of all predictions, reinforcing the overall tendency toward overconfidence.

Table 4
Comparison of runtime performance of LLMs using P_1 and P_2 prompt strategies on the D_1 dataset (in seconds). The values represent processing time (in seconds) for batches of 125 domains and include mean runtime ($\bar{x}$), variance (σ^2), standard deviation (σ), median ($\tilde{x}$), minimum (x_{min}) and maximum (x_{max}) execution times.

LLM	Prompt	$\bar{x}$	σ^2	σ	$\tilde{x}$	x_{min}	x_{max}
`GPT-4o`	P_1	33.23	71.09	8.43	31.14	17.10	73.47
	P_2	34.61	109.85	10.48	31.62	20.18	88.78
`GPT-4o-mini`	P_1	18.58	9.56	3.09	17.83	13.92	33.92
	P_2	19.41	12.42	3.52	18.82	13.12	37.38
`Claude 3.5 Sonnet`	P_1	28.92	33.06	5.75	29.04	21.37	89.52
	P_2	32.67	17.28	4.16	32.45	25.35	62.18
`Claude 3.5 Haiku`	P_1	30.68	14.63	3.83	31.50	25.32	56.44
	P_2	30.71	12.98	3.60	31.33	18.52	37.27
`Gemini 1.5 Pro`	P_1	49.73	136.11	11.67	46.39	28.16	94.52
	P_2	43.64	29.12	5.40	43.01	27.85	80.18
`Gemini 1.5 Flash`	P_1	16.87	1.17	1.08	16.92	11.21	30.96
	P_2	17.23	1.66	1.29	17.25	15.30	27.82
`Gemini 1.5 Flash-8B`	P_1	9.91	0.34	0.58	9.85	8.24	16.80
	P_2	10.09	0.57	0.76	10.12	3.27	18.84
`Mistral Large`	P_1	40.34	114.22	10.69	43.11	15.16	71.69
	P_2	46.54	210.98	14.53	47.67	15.36	99.87
`Mistral Small`	P_1	31.02	110.18	10.50	29.37	1.31	96.26
	P_2	30.59	37.03	6.08	30.72	0.82	46.08



(a) P_1 prompt strategy (b) P_2 prompt strategy

Fig. 7. Distribution of execution times (in seconds) for different LLMs under P_1 (left) and P_2 (right) prompting strategies on the D_1 dataset. The values represent processing time for batches of 125 domains.

In summary, these findings highlight a critical limitation in the self-assessment capabilities of models, as a high level of confidence does not consistently reflect high accuracy in real-world scenarios. The marked overconfidence, especially in labeling malicious domains, warrants cautious interpretation of confidence scores when deploying LLM in production environments.

The observed overconfidence in model predictions reflects an inherent characteristic of pre-trained LLMs when applied to classification tasks. While calibration techniques such as temperature scaling, histogram binning, or confidence thresholding can help mitigate this effect, their application requires access to validation data and post ad-hoc adjustments, which is beyond the scope of our zero-shot evaluation framework. In this work, we intentionally retained the original confidence results of the models to provide an accurate assessment of their unmodified behavior in cybersecurity-relevant scenarios.

5.6. Missing domains

During the evaluation, some domains were not classified on the first attempt due to internal LLM processing limitations and output generation issues. This behavior, observed across all evaluated models with varying frequencies (see Fig. 9), manifested itself three distinct error patterns:

(i) output format violations (e.g., extraneous characters such as brackets or quotation marks),
(ii) domain name transcription errors (misspellings or character substitutions), and
(iii) domain omissions in the output.

Although each LLM exhibited these errors to varying degrees, larger models tended to produce more transcription errors, while smaller models were more prone to format violations. Domain omissions were less common overall, but still occurred across all models.

Fig. 9 shows classification failures for each LLM. Since each prompt processes 125 domains, the average error rate per prompt ranged from 0.096% (`Claude 3.5 Sonnet`) to 22.9% (`Gemini 1.5 Pro`), reflecting notable disparities in reliability across models. Furthermore, the more specialized prompt in this experiment (P_2) consistently resulted in a higher number of unclassified domains, suggesting that increased

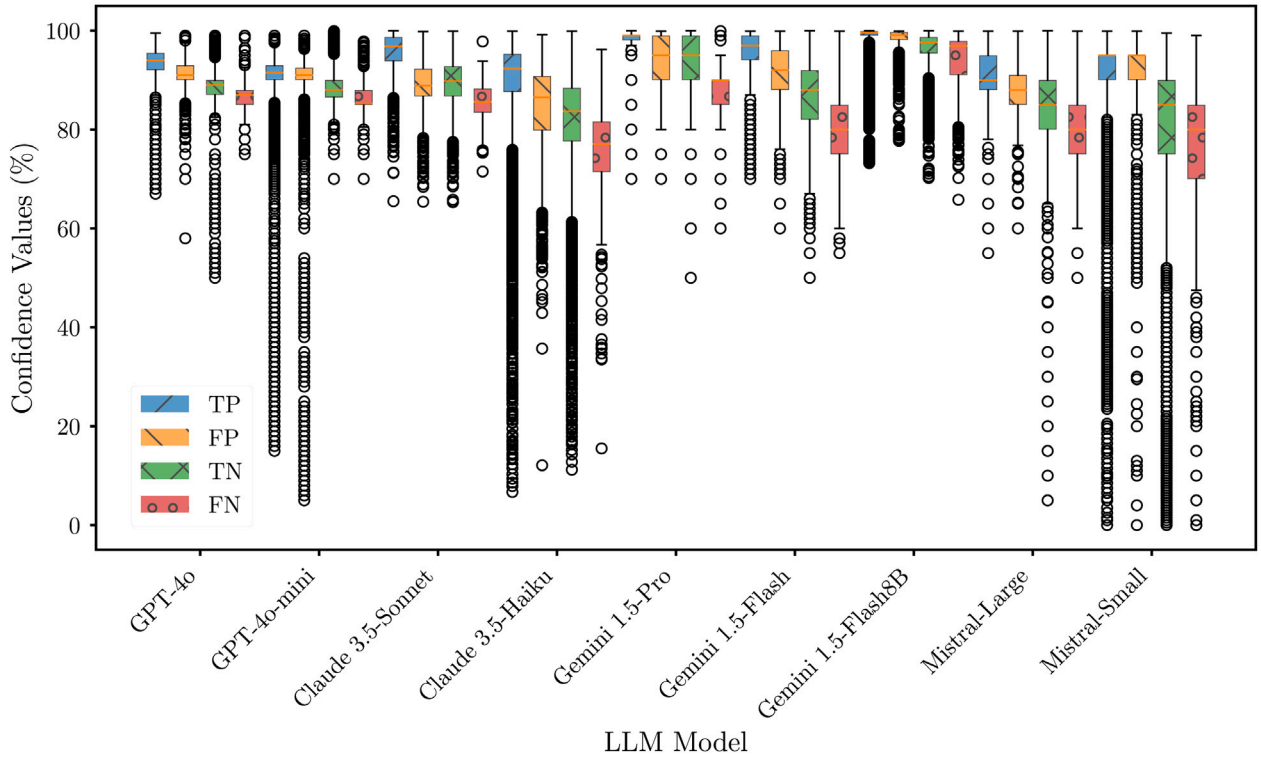


Fig. 8. Distribution of confidence values in different classification results for each LLM.

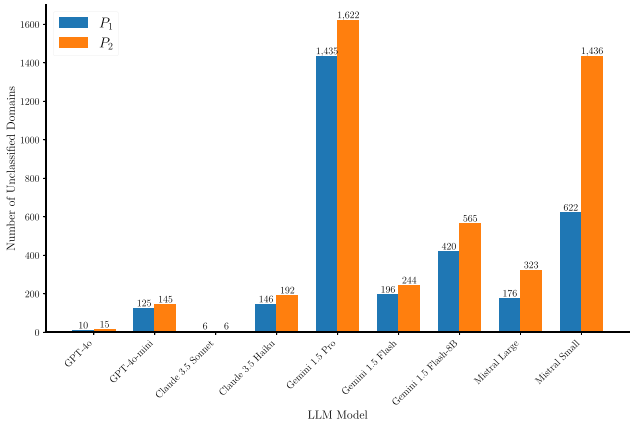


Fig. 9. Number of unclassified domains on the first attempt of each LLM using the P_1 and P_2 prompt strategies.

prompt complexity may lead to formatting errors and misinterpretations. In contrast, the simpler prompt (P_1) produced fewer output errors, despite its lack of domain-specific targeting.

To better understand the limitations of LLMs in domain classification, we conducted a qualitative analysis of the unclassified samples, focusing on the domain types that were most frequently left unclassified. Table 5 presents the distribution of unclassified domains by generation scheme under both prompting strategies, revealing systematic patterns that shed light on the LLM's weaknesses.

The results reveal a strong concentration of misses among arithmetic-based domains. While the D_1 dataset contains an equal distribution of benign and malicious domains (25,000 each), arithmetic-based DGAs account for approximately three-quarters of all classification misses (75.99% under P_1 and 74.47% under P_2). This imbalance

Table 5
Distribution of unclassified domains by generation scheme and domain type under the prompting strategies P_1 and P_2.

Domain type	P_1	P_2
Arithmetic-based	75.99	74.47
Hash-based	12.00	12.89
Dictionary-based	9.02	9.29
Benign	2.98	3.34

suggests that arithmetic-based generation schemes pose a structural challenge to LLMs, resulting in disproportionately high miss rates relative to their presence in the dataset.

In contrast, hash- and dictionary-based DGAs contribute more moderately to misclassification, accounting for approximately 12% and 9% of unclassified domains, respectively. Notably, benign domains, despite representing 50% of the dataset, account for less than 3.5% of total classification failures. This disparity indicates that LLMs are significantly more reliable when processing legitimate domains than when handling certain types of DGAs, particularly those generated by arithmetic schemes.

Furthermore, the similarity in failure distribution between the two prompt strategies suggests that these difficulties are due to intrinsic limitations of the model, rather than the prompt design. Prompting alone does not appear sufficient to overcome the structural challenges posed by certain domain generation techniques.

5.7. Cost–benefit analysis

To assess the practical feasibility of LLM-based AGD detection in operational environments, we analyzed the trade-off between detection performance and associated inference costs based on our experimental data. Table 6 summarizes the actual API costs incurred for each prompt strategy in the evaluated commercial LLMs.

Table 6
API costs (in USD) to process 50,000 domains in different prompt strategies.

Model	P_1	P_2	P_3	P_4
`GPT-4o`	8.26	8.70	13.42	9.34
`GPT-4o-mini`	0.50	0.53	0.81	0.57
`Claude Sonnet 3.5`	13.88	14.42	26.26	15.46
`Claude Haiku 3.5`	3.67	3.86	6.49	4.14

Our analysis reveals substantial variability in cost, depending on both the model and the prompt complexity. To process the full dataset of 50,000 domains, total costs ranged from as little as $0.50 (using `GPT-4o-mini` with P_1 prompt) to $26.26 (using `Claude 3.5 Sonnet` with P_3 prompt). Prompt complexity is a major factor in cost differences: P_3, which implements few-shot learning with multiple examples in context, results in costs between 54% and 82% higher compared to simpler strategies across all models. Similarly, P_4, which introduces real-world contextual information, generates modest increases of 7% to 13% over P_2.

These findings underscore the importance of considering not only model accuracy but also economic and infrastructure constraints when implementing LLM for large-scale detection tasks. In cost-sensitive environments, these trade-offs can influence the choice of prompt design or model selection, especially when real-time processing is required.

The cost-performance ratio reveals trade-offs relevant to practical deployment. `Claude Sonnet 3.5` achieves the highest accuracy (89.4%), but also incurs the highest costs ($14.42 for P_2). In contrast, `GPT-4o-mini` offers a lower, albeit reasonable, accuracy performance (78.5%) at much lower cost ($0.53). The 10.9% accuracy improvement translates to a 27-fold increase in cost, equivalent to approximately $1.32 per additional percentage point of accuracy. This disparity underscores a fundamental challenge in applying LLM to security tasks: achieving top-tier performance requires a substantial monetary investment that can be prohibitive in many real-world scenarios.

These costs increase linearly with data volume, posing new limitations in operational environments. For example, processing one million domains per day would result in daily costs of between $10.60 and $525.20, corresponding to annual expenses of between $3,869 and $191,698, depending on the management model and strategy. These high recurring costs can be a barrier to continuous deployment, especially in resource-constrained contexts such as IoT security monitoring or integrated threat detection. In these environments, the use of free or lower-cost models, such as those from the `Gemini` or `Mistral` families, may be necessary, even at the cost of some performance degradation.

6. Evaluating AGD classification in malware families

To address RQ2, we evaluate the performance of LLMs using the P_3 prompt strategy, which follows a two-step classification process. First, the models detect malicious AGDs in D_2. Then, for any domain identified as malicious, the models apply FSL (with 10 examples per family) to assign each domain to its corresponding family. This sequential approach evaluates how well each model can detect and categorize malicious domains.

6.1. Impact of size in FSL

We conduct a preliminary study to identify the optimal number of examples for FSL, discarding the zero-shot approach since at least one reference example is needed for classification. We systematically test sample sizes from a single example up to 25 examples, tracking the ability of LLMs to classify AGDs by family. Fig. 10 summarizes our findings.

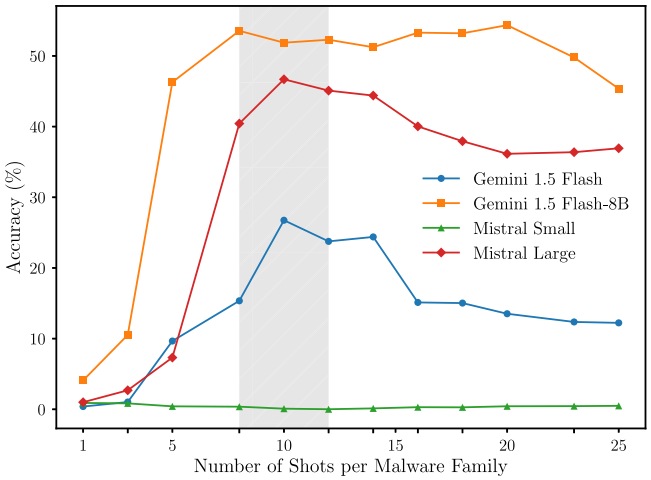


Fig. 10. Progression of LLM performance in classifying AGD malicious domains across malware families using FSL with varying sample sizes.

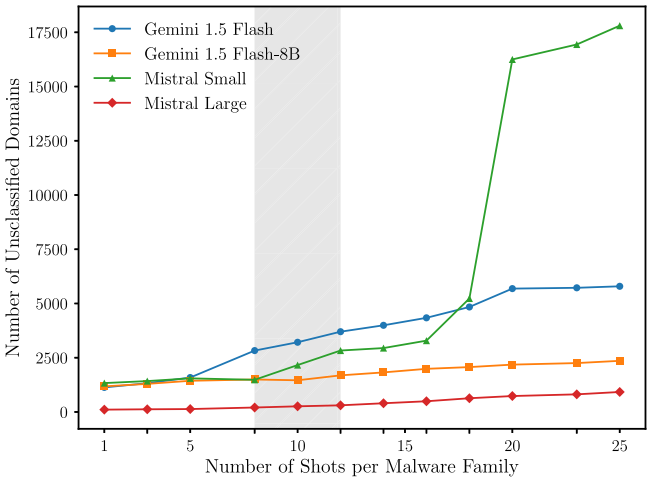


Fig. 11. Unclassified AGDs requiring reclassification attempts across different few-shot sample sizes.

It is noteworthy that `Mistral Small` failed to classify AGDs by family, showing close to zero performance at all sample sizes tested; therefore, its results are excluded from further analysis. Among the other models, both a very low and a very high number of examples led to suboptimal performance, suggesting an ideal range for FSL in this domain. As shown in Fig. 11, increasing the number of samples also increases the proportion of unclassified domains.

The shaded region in both figures denotes a range of 10 samples, reflecting the best balance between classification accuracy and minimizing first-shot failures. This region indicates that using approximately 10 examples per family strikes a balance between performance gains and avoiding excessive unclassified results.

6.2. Assessment of malware family classification

Fig. 12 shows the precision, recall, and F1-score metrics of the LLMs evaluated in classifying malicious AGDs by malware family. Substantial performance differences emerge between models and vendors, with the OpenAI and `Mistral Small` models performing particularly poorly. Likewise, `Claude 3.5 Haiku` and `Gemini 1.5 Flash` do not exceed 40% on any metric (indicated by the red line in Fig. 12), justifying their exclusion from further analysis.

Table 7
Performance comparison of different models in DGA detection by malware family type.

Family	Claude Sonnet 3.5			Gemini 1.5 Pro			Gemini 1.5 Flash-8B			Mistral Large			Type
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	
banjori	99.9	99.6	99.7	96.9	98.9	97.9	83.0	63.9	72.2	96.2	63.0	76.1	Arithmetic
conficker	96.9	97.6	97.2	85.9	69.1	76.6	54.1	31.1	39.5	76.5	14.1	23.8	
emotet	98.2	100.0	99.1	97.2	98.9	98.0	98.2	99.5	98.8	95.1	42.4	58.6	
flubot	98.9	97.1	98.0	89.5	76.1	82.3	65.4	64.9	65.1	69.8	39.5	50.5	
gameover	99.6	97.0	98.3	94.6	92.2	93.4	82.0	66.7	73.6	92.0	44.4	59.9	
metastealer	100.0	100.0	100.0	99.7	100.0	99.8	95.4	99.7	97.5	95.0	99.6	97.2	
necurs	97.0	89.9	93.3	80.9	60.6	69.3	78.3	59.7	67.8	52.0	43.7	47.5	
nymaim	97.6	98.7	98.1	73.6	93.6	82.4	77.4	85.2	81.1	73.0	88.4	80.0	
pitou	99.7	99.5	99.6	96.2	93.4	94.8	82.2	84.4	83.3	89.0	77.4	82.8	
pushdo	99.2	99.9	99.6	98.6	98.6	98.6	95.4	96.4	95.9	94.8	97.7	96.2	
qakbot	96.0	98.6	97.3	85.5	84.4	84.9	60.6	49.4	54.5	38.1	57.4	45.8	
rovnix	100.0	100.0	100.0	95.5	99.8	97.6	57.0	35.8	44.0	87.4	96.1	91.5	
virut	99.0	100.0	99.5	94.8	100.0	97.3	95.1	99.8	97.4	79.8	99.5	88.5	
zloader	100.0	100.0	100.0	97.1	97.3	97.2	70.9	93.4	80.6	74.7	44.6	55.8	
gozi	97.2	99.9	98.5	95.8	97.2	96.5	55.9	90.3	69.1	90.1	87.5	88.7	Dictionary
matsnu	94.4	99.6	96.9	80.5	92.6	86.1	75.2	95.1	84.0	7.9	63.0	14.0	
nymaim2	96.1	99.6	97.8	83.8	97.3	90.0	46.5	42.6	44.4	9.1	22.9	13.0	
suppobox	99.6	88.8	93.9	98.6	96.5	97.6	98.5	39.4	56.3	86.8	96.2	91.3	
darkwatchman	100.0	100.0	100.0	100.0	100.0	100.0	89.3	99.9	94.3	83.7	99.8	91.1	Hash
dyre	99.0	100.0	99.9	99.8	100.0	99.9	97.8	99.8	98.8	99.0	88.9	93.7	
grandoreiro	100.0	100.0	100.0	100.0	100.0	100.0	99.9	100.0	99.9	100.0	99.9	99.9	
monerominer	100.0	100.0	100.0	99.8	100.0	99.9	99.2	99.9	99.6	83.2	99.6	90.7	
pandabanker	100.0	100.0	100.0	100.0	99.8	99.9	98.8	97.6	98.2	97.8	60.7	74.9	
tinynuke	100.0	100.0	100.0	99.8	99.3	99.6	96.0	67.7	79.4	97.1	83.9	90.0	
wd	100.0	100.0	100.0	99.3	99.9	99.6	97.9	99.6	98.7	95.7	99.0	97.3	
Total	96.2	91.4	93.7	88.3	72.5	79.6	62.7	51.9	56.8	46.7	37.2	41.4	–

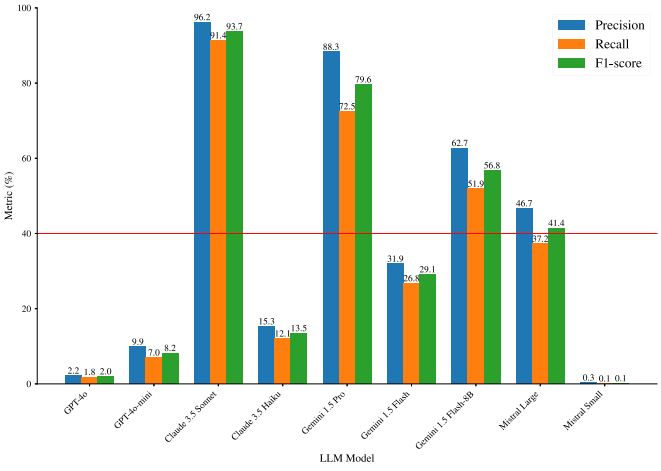


Fig. 12. Comparative performance of LLMs in classifying malicious AGD families using the D_2 dataset and the P_3 prompt strategy.

Among the remaining models, precision generally exceeds recall, suggesting a conservative classification strategy; models tend to be accurate once they label a domain as belonging to a specific family, but risk missing valid instances. To dig deeper, Table 7 examines how LLMs classify specific DGA families, revealing that different DGAs significantly influence detection performance.

Interestingly, a closer look at individual families reveals a contrasting pattern: for most families and models, recall outperforms precision. This trend implies a tendency toward over-assignment rather than a conservative approach, where models readily label domains as belonging to a particular family, but are sometimes inaccurate. In some cases, however, precision significantly outperforms recall, enough to skew overall metrics toward higher average precision.

In particular, when broken down by generation scheme:

- Hash-based DGAs stand out for consistently high classification rates across all models. `Claude 3.5 Sonnet` achieves perfect accuracy in this category, likely thanks to its ability to recognize the distinctive hexadecimal patterns of hash-based domains.
- Arithmetic-based DGAs show mixed results. Families such as `metastealer`, `rovnix`, and `zloader` maintain strong classification scores, but `conficker`, notable for being one of the first arithmetic-based DGAs, exhibits significantly lower detection rates. Its simpler generation mechanism can blur the line between malicious and legitimate domains.
- Dictionary-based DGAs prove the most challenging, with weaker overall classification metrics compared to other schemes. Despite additional guidance in P_3 on dictionary-based generation, most models appear to rely on character pattern analysis rather than semantics, hampering their ability to correctly classify domains based on natural words.

7. Evaluating performance on real-world domains

To address RQ3, we evaluate the performance of LLMs on real-world domains from D_3. This setup tests each model's ability to correctly classify legitimate domains that may share structural similarities with AGD patterns, an essential step in understanding false positives in practical deployment.

7.1. Real-world domain classification analysis

Fig. 13 shows a substantial drop in performance when the models classify real-world domains from D_3, compared to their results on D_1. Accuracy values for P_4 decrease by 11% to 24% relative to P_1 and P_1 on benign domains from D_1. An exception is `Claude 3.5 Sonnet`, which maintains similar classification rates, suggesting stronger reasoning capabilities in this more complex setting.

These results highlight a fundamental challenge: LLMs struggle to classify legitimate domains that exhibit some characteristics commonly flagged as suspicious. This limitation is particularly problematic in real-world environments, where false positives can cause unwarranted disruptions and potentially severe business impacts. Addressing this reliability gap in LLMs remains a critical hurdle for production-ready AGD detection.

The performance degradation observed on real-world domains in the D_3 dataset further exacerbates the FP problem. When evaluating our university's DNS dataset, composed of 50,000 legitimate domain queries, the models incorrectly classified between 5,500 and 12,000 domains as malicious, corresponding to an 11% to 24% reduction in

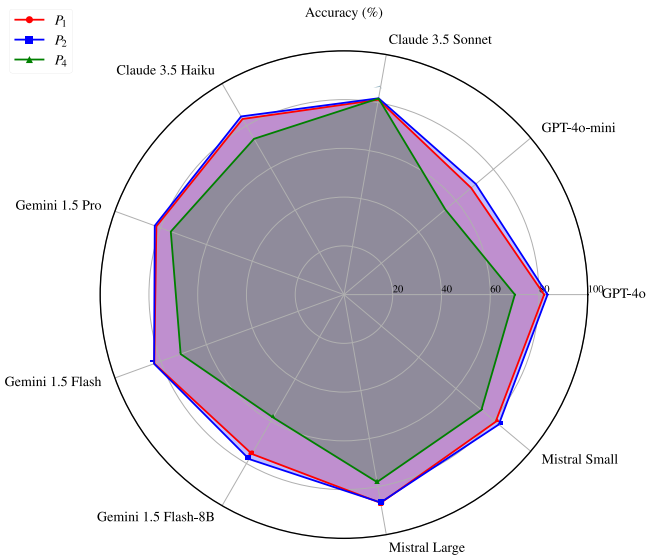


Fig. 13. Comparison of the accuracy of LLMs in detecting benign domains using P_1 and P_2 prompts with the D_1 dataset and P_4 with the D_3 dataset.

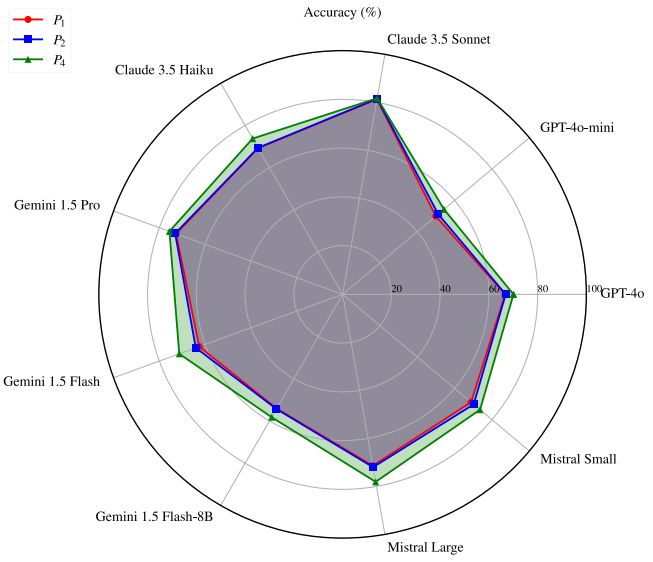


Fig. 14. Comparison of the accuracy of LLMs in detecting benign domains using P_1, P_2, and P_4 prompts with the D_3 dataset.

accuracy. Extrapolating these results to an enterprise-scale deployment processing one million domain queries per day, this would translate to between 110,000 and 240,000 false positives daily. Such a high volume of misclassifications would severely disrupt normal operations, routinely blocking legitimate services and making them inaccessible to users.

7.2. Optimizing prompts for real-world domain classification

Classifying benign domains that resemble AGDs poses a unique challenge in real-world network traffic. This section compares the effectiveness of different indications in handling legitimate domains that might generate false positives due to AGD-like structures.

The P_4 prompt strategy explicitly takes benign domains with AGD-like structures into account by recommending focused analysis of TLD and SLD tuples, along with more detailed contextual guidance. As shown in Fig. 14, these improvements translate into improved accuracy across all LLMs tested, compared to P_1 and P_2. While this improvement represents a step forward in reducing false positives, it also underscores the need for carefully designed prompts that reflect the nuances of real-world network traffic.

8. Limitations

While our findings demonstrate the potential of LLMs for AGD detection, several limitations limit the broader applicability and generalizability of our results. These factors encompass technical, hardware, economic, temporal, and operational considerations, which we acknowledge in order to provide a clearer context for the conclusions drawn in this work.

LLMs features. Current LLMs face notable limitations on output length, with OpenAI models offering the largest limit of approximately 16,000 tokens. This limit requires multiple API calls to classify large sets of domains, leading us to limit each request to 125 domains to avoid exceeding response limits. Additionally, context window restrictions complicate the detection or clustering of previously unseen malware families, as significantly larger context windows would be required. While models such as `Gemini 1.5 Pro` offer potentially adequate windows, free tier restrictions currently prevent a comprehensive large-scale comparison between different LLMs.

Hardware constraints. Although open-source alternatives such as Meta's LLaMA [43] are freely available, their larger models typically require substantial computational resources beyond the reach of most users. Smaller variants, such as `LLaMA 3.1-8B`, can run on standard hardware, but have underperformed in our experiments, primarily because they do not consistently follow the requested output format for domain detection.

Dataset considerations. While our datasets offer sufficient diversity to address the specific research questions posed in this work, certain limitations in data selection should be acknowledged. While DGArchive provides a comprehensive collection of 137 malware families, and our selection of 25 families was designed to represent diverse domain generation schemes, incorporating additional sources of malicious domains to capture emerging DGA techniques not reflected in the current dataset could be beneficial for future work.

For benign domains, we used a combination of the widely used Tranco list (considered the current standard in legitimate domain research) and DNS records collected from a university network. It is important to clarify that the university dataset was not intended to be globally representative. Instead, it serves as a concrete example of DNS traffic in an academic environment, complementing Tranco's broader scope. This inclusion allows for the evaluation of LLM behavior in real, non-synthesized traffic, which may contain benign domains with structural properties similar to those of AGD.

The deliberate integration o benign, globally representative (Tranco) and context-specific (university records) datasets provides a balanced evaluation framework for assessing LLMs' ability to differentiate between legitimate and algorithmically generated domains. While we recognize that DNS traffic can vary significantly in other environments (e.g., corporate or Internet service provider networks), our dataset design aligns with the goal of evaluating LLMs' pattern recognition capabilities under controlled but realistic conditions.

Economic factors. Because LLMs represent an emerging technology with constant operating costs, financial constraints limit the number of models we can feasibly evaluate. Our reliance on third-party vendors (or cloud platforms for open-source models) inherently restricts the breadth and depth of our experimentation.

Scope of prompt engineering. Our prompt engineering strategy leverages incremental complexity (P_1-P_4) to systematically evaluate the effect of contextual information on LLM performance. While this approach enables controlled evaluation of baseline capabilities, it does not incorporate advanced prompt engineering techniques such as chain-of-thought reasoning, iterative refinement, multi-step prompting, or more sophisticated reasoning frameworks. These techniques can improve performance but require specific optimization and task-specific tuning, which is beyond the scope of this work. Our goal remains to establish a fundamental understanding of LLM's behavior in detecting AGD under standardized and reproducible conditions.

Operational feasibility and impact of false positives. The high FPR observed across all evaluated models, ranging from 13.2% to 31.9%, pose significant obstacles to real-world deployment. In enterprise environments processing 1 million legitimate domain queries per day, these rates would result in the incorrect blocking 132,000 to 319,000 legitimate domains, causing widespread service disruptions. The problem is exacerbated when applying the models to real-world DNS traffic, where an accuracy degradation of 11% to 24% could generate an additional 110,000 to 240,000 FPs per day. These results would severely impair normal network operations and require extensive human intervention or auxiliary validation systems, which would be operationally impractical and economically unsustainable.

Throughput and integration challenges for DNS firewalls. The runtime results presented in Table 4 highlight substantial throughput limitations that hamper real-time deployment. While DNS firewalls in production environments typically handle thousands of queries per second, the fastest model tested (`Gemini 1.5 Flash-8B`) only manages about 12.6 domains per second, and the slowest (`Gemini 1.5 Pro`) handles only about 2.5 domains per second. This discrepancy, spanning nearly two orders of magnitude, makes direct integration into high-performance DNS filtering systems unfeasible without significant architectural adjustments.

In addition to processing speed, several additional integration challenges must be considered. LLM inference latency, especially in commercial APIs, far exceeds the sub-millisecond response times typically required in DNS infrastructure [44]. Furthermore, our evaluation relies on batch processing, which does not fit the inherently flow-oriented nature of live DNS traffic. High FPRs further complicate implementation, as they could cause substantial disruption to legitimate services.

While techniques such as parallelization, load balancing, and response caching can mitigate some of these challenges, they introduce considerable engineering complexity and operational overhead. A more practical alternative might be a multi-tiered architecture, where conventional statistical or neural methods handle the majority of traffic at line speed, and LLM-based analysis is reserved for ambiguous or high-risk cases. Given current performance and reliability limitations, commercial LLMs appear to be better suited for offline analysis, post-event investigation, or as auxiliary tools in threat intelligence processes, rather than as primary components of real-time filtering.

Time-dependent considerations. All experiments were conducted between December 2024 and January 2025, making our results dependent on the model's capabilities during that period. Vendors may update their algorithms without public notice, making it difficult to track subsequent changes. However, to aid reproducibility, Table 1 specifies the exact model identifiers (API tags) used in this work.

9. Conclusions

Algorithmically-generated domain detection has emerged as a key area of research in recent years. Motivated by the demonstrated versatility of LLMs, we explored their potential for malicious AGD detection. Our findings suggest that LLMs can effectively distinguishing benign and malicious AGD domains for binary classification. However, significant challenges remain in multiclass classification of DGA types into malware families, particularly for dictionary-based schemes, as well as in accurately identifying benign domains within real-world traffic. These challenges represent fundamental obstacles to the practical deployment of LLMs for AGD detection.

It is important to distinguish our approach from traditional fine-tuning methods commonly used in previous research. The few-shot prompt strategy adopted in this work provides illustrative examples directly within the prompt to guide model behavior, but does not involve any modification to its parameters. Our results demonstrate that modern LLMs can leverage their pre-trained knowledge to detect AGDs using simple prompting techniques, without the need for domain-specific retraining or architecture changes.

While these findings are encouraging, future work evaluate LLM-based AGD detection in a wider range of operational environments. Our use of university DNS records was intentionally limited to evaluating model behavior in a specific academic environment and was not intended to represent global traffic patterns. However, enterprise networks, ISPs, healthcare systems, and other domains present distinct DNS characteristics and threat profiles. Evaluating the performance of LLMs in these contexts would provide a more complete view of their applicability in real-world deployment scenarios.

To improve multiclass classification performance, particularly with regard to malware family identification, future research could explore two-stage classification architectures. In this process, an initial binary classification phase, similar to the one presented in this work, could be followed by a dedicated family-level classifier. This second stage could benefit from fine-tuning open-source LLMs on labeled samples specific to DGA families, thereby improving discrimination between similar patterns, especially among dictionary-based DGAs, which were among the most complex in our work.

The performance limitations observed in our runtime analysis likely represent the most significant obstacle to production deployment. Current commercial LLMs cannot meet the latency and volume requirements of high-speed DNS firewalls. However, they remain suitable for use cases such as offline analysis, forensic investigation, and threat hunting, where real-time constraints are less stringent. Future work should explore model compression and distillation techniques that preserve the pattern recognition capabilities of large LLMs while enabling line-speed inference. Furthermore, training domain-specific LLMs, focusing exclusively on the structural and lexical features of domain names, can lead to more lightweight and easier-to- implement models.

Future work will focus on overcoming these limitations by adapting open-source models and leveraging fine-tuning techniques to develop more specialized and efficient detection systems.

CRediT authorship contribution statement

Tomás Pelayo-Benedet: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Ricardo J. Rodríguez:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Methodology, Funding acquisition, Conceptualization. **Carlos H. Gañán:** Writing – review & editing, Visualization, Methodology.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT-4 in order to improve readability and language. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

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

This research was supported in part by grant PID2023-151467OA-I00 (CRAPER), funded by MICIU/AEI/10.13039/501100011033, Spain and by ERDF/EU, by grant TED2021-131115A-I00 (MIMFA), funded by MICIU/AEI/10.13039/501100011033, Spain and by the European Union
NextGenerationEU/PRTR, by grant *Proyecto Estratégico Ciberseguridad EINA UNIZAR*, funded by the Spanish National Cybersecurity Institute (INCIBE) and the European Union NextGenerationEU/PRTR, by grant *Programa de Proyectos Estratégicos de Grupos de Investigación* (DisCo research group, ref. T21-23R), funded by the University, Industry and Innovation Department of the Aragonese Government, Spain, and by the RAPID project (Grant No. CS.007) financed by the Dutch Research Council (NWO). We extend our gratitude to the DGArchive team for providing the current dataset in advance, allowing us to begin experimentation sooner.

Appendix. Developed prompts

This appendix provides the full text of all prompts (P_1–P_4) used in our experiments. Each prompt was constructed using the iterative design methodology described in Section 4 and illustrated in Fig. 1. These prompts define the exact task formulations and contextual instructions provided to the LLMs, ensuring the transparency and reproducibility of our assessment process.

A.1. P_1: Minimal approach

Act as a cybersecurity expert specialized in detecting malicious Algorithmically Generated Domains (AGDs).
Your task is to analyze a list of domains, one per line, and determine for each one if it is a malicious AGD or a legitimate domain. You need to analyze the domains yourself, I don't want you to write me a script to solve the problem, my intention is for you to reason on your own whether the domains are malicious AGDs or legitimate domains. I need you to analyze the full list, without truncating for brevity, don't pause and if you ask me to continue the answer will be always yes.
Provide ONLY your answer in the following exact format, with no additional text: [DOMAIN]|Y|[CONFIDENCE] or [DOMAIN]|N|[CONFIDENCE]
Where:
- [DOMAIN] is the analyzed domain
- Y means it is a malicious AGD
- *N* means it is not a malicious AGD
- [CONFIDENCE] is a number between 0.0 and 100.0 with one decimal indicating the confidence percentage in the decision
Response format examples:
For example1.com that in this example is a malicious AGD with 95.5% confidence: example1.com|Y|95.5
For example2.com that in this example is a legitimate domain with 87.3% confidence: example2.com|N|87.3
The domains to analyze will be provided in the following message, separated by commas.

A.2. P_2: Lexical feature analysis

Act as a cybersecurity expert specialized in detecting malicious Algorithmically Generated Domains (AGDs).
Your task is to analyze a list of domains, one per line, and determine for each one if it is a malicious AGD or a legitimate domain. You need to analyze the domains yourself, I don't want you to write me a script to solve the problem, my intention is for you to reason on your own whether the domains are malicious AGDs or legitimate domains. I need you to analyze the full list, without truncating for brevity, don't pause and if you ask me to continue the answer will be always yes.
Malicious AGDs can follow different generation schemes:
* Arithmetic-based: Uses mathematical operations to generate domains
* Hash-based: Employs cryptographic hash functions for domain generation
* Wordlist-based: Creates domains by concatenating words from predefined lists
* Permutation-based: Generates variations by permuting a base domain string
* Adversarial-based: Creates domains that deliberately evade detection by replicating benign domain characteristics
When analyzing a domain, please consider the following lexical features:
1. Statistical features
* Level of Randomness: Malicious domains tend to have higher levels of randomness due to uniform and random character distribution, while legitimate domains have lower randomness due to natural linguistic patterns.
* Character Frequency: Legitimate domains use common characters in proportions similar to natural language, while malicious ones tend to use uncommon characters more frequently.
* Digit/Letter Ratio: Malicious domains tend to have a higher proportion of digits to letters, while legitimate ones maintain a more moderate and contextual use of numbers.
2. Pronounceability Features
* Pronounceability Index: Legitimate domains are easily pronounceable due to their natural linguistic structure, while malicious ones are difficult or impossible to pronounce.
* Consonant/Vowel Ratio: Legitimate domains maintain a natural proportion between consonants and vowels similar to human language, while malicious ones tend to have unbalanced proportions.
3. Linguistic Features
* Meaningful Word Presence: Legitimate domains tend to contain real words or meaningful combinations, while malicious ones use meaningless character sequences. This is particularly effective for detecting hash-based and arithmetic-based AGDs.

* Dictionary Presence: Legitimate domains tend to contain words that appear in dictionaries or recognizable combinations of them, while malicious ones use random sequences. While wordlist-based AGDs may contain dictionary words, they often combine them in unnatural ways.
4. Composition Features
* Similarity to Popular Domains: Malicious domains often try to imitate popular domains with small variations, while legitimate ones are the original brand domains.
* Distance to Known Domains: Malicious domains maintain a calculated distance to popular domains to deceive users, while legitimate ones are the original domains without variations.
Provide ONLY your answer in the following exact format, with no additional text:
[DOMAIN]|Y|[CONFIDENCE]
or
[DOMAIN]|N|[CONFIDENCE]
Where:
- [DOMAIN] is the analyzed domain
- Y means it is a malicious AGD
- *N* means it is not a malicious AGD
- [CONFIDENCE] is a number between 0.0 and 100.0 with one decimal indicating the confidence percentage in the decision
Response format examples:
For example1.com that in this example is a malicious AGD with 95.5% confidence: example1.com|Y|95.5
For example2.com that in this example is a legitimate domain with 87.3% confidence: example2.com|N|87.3
The domains to analyze will be provided in the following message, separated by commas.

A.3. P_3: Malware family classification

Act as a cybersecurity expert specialized in detecting malicious Algorithmically Generated Domains (AGDs).
Your task is to analyze a list of domains, one per line, and determine for each one if it is a malicious AGD or a legitimate domain. You need to analyze the domains yourself, I don't want you to write me a script to solve the problem, my intention is for you to reason on your own whether the domains are malicious AGDs or legitimate domains. I need you to analyze the full list, without truncating for brevity, don't pause and if you ask me to continue the answer will be always yes.
Malicious AGDs can follow different generation schemes:
* Arithmetic-based: Uses mathematical operations to generate domains
* Hash-based: Employs cryptographic hash functions for domain generation
* Wordlist-based: Creates domains by concatenating words from predefined lists
* Permutation-based: Generates variations by permuting a base domain string
* Adversarial-based: Creates domains that deliberately evade detection by replicating benign domain characteristics
When analyzing a domain, please consider the following lexical features:
1. Statistical features
* Level of Randomness: Malicious domains tend to have higher levels of randomness due to uniform and random character distribution, while legitimate domains have lower randomness due to natural linguistic patterns.
* Character Frequency: Legitimate domains use common characters in proportions similar to natural language, while malicious ones tend to use uncommon characters more frequently.
* Digit/Letter Ratio: Malicious domains tend to have a higher proportion of digits to letters, while legitimate ones maintain a more moderate and contextual use of numbers.
2. Pronounceability Features
* Pronounceability Index: Legitimate domains are easily pronounceable due to their natural linguistic structure, while malicious ones are difficult or impossible to pronounce.
* Consonant/Vowel Ratio: Legitimate domains maintain a natural proportion between consonants and vowels similar to human language, while malicious ones tend to have unbalanced proportions.
3. Linguistic Features
* Meaningful Word Presence: Legitimate domains tend to contain real words or meaningful combinations, while malicious ones use meaningless character sequences. This is particularly effective for detecting hash-based and arithmetic-based AGDs.

* Distance to Known Domains: Malicious domains maintain a calculated distance to popular domains to deceive users, while legitimate ones are the original domains without variations.
Malicious AGDs are grouped into families, where each family is created using a specific Domain Generation Algorithm (DGA) and a seed value. This algorithm-seed combination acts as a "recipe" that produces similar-looking domains within the same family.
There are some of known malicious AGD families that you should know to classify. The format is the name of the family followed by the character ":" and then a list of malicious AGDs separated by the character "," and limited by the character ";". The families are the following:
<family1>:<domain1>,<domain2>, ...;
...:...;
Provide ONLY your answer in the following exact format, with no additional text:
[DOMAIN]|Y|[DGA_FAMILY]|[CONFIDENCE]
or
[DOMAIN]|N|-|[CONFIDENCE]
Where:
- [DOMAIN] is the analyzed domain
- Y means it is a malicious AGD
- *N* means it is not a malicious AGD
- [DGA_FAMILY] is:
* The known family name if identified
* "UnknownFamilyX" (where X is a number) if malicious but family unknown
* "-" if the domain is legitimate
- [CONFIDENCE] is a number between 0.0 and 100.0 with one decimal indicating the confidence percentage in the decision
Response format examples:
For example1.com that in this example is a malicious AGD from Necurs family with 95.5% confidence: example1.com|Y|Necurs|95.5
For example2.com that in this example is a legitimate domain with 87.3% confidence: example2.com|N|-|87.3
For example3.com that in this example is a malicious AGD from unknown family with 92.1% confidence: example3.com|Y|UnknownFamily1|92.1
For example4.com that in this example is a malicious AGD from unknown family with 91.1% confidence: example4.com|Y|UnknownFamily2|91.1
For example5.com that in this example is a malicious AGD that seems from the same unknown family as example3.com with 89.4% confidence: example5.com|Y|UnknownFamily1|89.4
The domains to analyze will be provided in the following message, separated by commas.

A.4. P_4: Real-world domain classification

Act as a cybersecurity expert specialized in detecting malicious Algorithmically Generated Domains (AGDs).
Your task is to analyze a list of domains, one per line, and determine for each one if it is a malicious AGD or a legitimate domain. You need to analyze the domains yourself, I don't want you to write me a script to solve the problem, my intention is for you to reason on your own whether the domains are malicious AGDs or legitimate domains. I need you to analyze the full list, without truncating for brevity, don't pause and if you ask me to continue the answer will be always yes.
Malicious AGDs can follow different generation schemes:
* Arithmetic-based: Uses mathematical operations to generate domains
* Hash-based: Employs cryptographic hash functions for domain generation
* Wordlist-based: Creates domains by concatenating words from predefined lists
* Permutation-based: Generates variations by permuting a base domain string
* Adversarial-based: Creates domains that deliberately evade detection by replicating benign domain characteristics
When analyzing a domain, please consider the following lexical features:
1. Statistical features
* Level of Randomness: Malicious domains tend to have higher levels of randomness due to uniform and random character distribution, while legitimate domains have lower randomness due to natural linguistic patterns.
* Character Frequency: Legitimate domains use common characters in proportions similar to natural language, while malicious ones tend to use uncommon characters more frequently.
* Digit/Letter Ratio: Malicious domains tend to have a higher proportion of digits to letters, while legitimate ones maintain a more moderate and contextual use of numbers.
2. Pronounceability Features
* Pronounceability Index: Legitimate domains are easily pronounceable due to their natural linguistic structure, while malicious ones are difficult or impossible to pronounce.
* Consonant/Vowel Ratio: Legitimate domains maintain a natural proportion between consonants and vowels similar to human language, while malicious ones tend to have unbalanced proportions.
3. Linguistic Features
* Meaningful Word Presence: Legitimate domains tend to contain real words or meaningful combinations, while malicious ones use meaningless character sequences. This is particularly effective for detecting hash-based and arithmetic-based AGDs.

* Distance to Known Domains: Malicious domains maintain a calculated distance to popular domains to deceive users, while legitimate ones are the original domains without variations.
Note that benign AGDs can exist. To correctly identify them as benign rather than malicious, you should pay closer attention to their combination of top-level domain and second-level domain, as subdomains could be created using DGAs while still being benign.
Provide ONLY your answer in the following exact format, with no additional text:
[DOMAIN]|Y|[CONFIDENCE]
or
[DOMAIN]|N|[CONFIDENCE]
Where:
- [DOMAIN] is the analyzed domain
- Y means it is a malicious AGD
- *N* means it is not a malicious AGD
- [CONFIDENCE] is a number between 0.0 and 100.0 with one decimal indicating the confidence percentage in the decision
Response format examples:
For example1.com that in this example is a malicious AGD with 95.5% confidence: example1.com|Y|95.5
For example2.com that in this example is a legitimate domain with 87.3% confidence: example2.com|N|87.3
The domains to analyze will be provided in the following message, separated by commas.

Data availability

Data will be made available on request.

References

[1] Ventures C. Cybercrime To Cost The World 10.5 Trillion Annually By 2025. 2023, [Online; https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/]. (Accessed 15 August 2023).

[2] MITRE. Dynamic Resolution: Domain Generation Algorithms. 2025, [Online; https://attack.mitre.org/techniques/T1568/002/]. (Accessed 20 May 2025).

[3] PaloAlto. 2024 How to Break the Cyber Attack Lifecycle. 2024, [Online; https://www.paloaltonetworks.com/cyberpedia/how-to-break-the-cyber-attack-lifecycle]. (Accessed 2 February 2024).

[4] Yong Wong M, Landen M, Antonakakis M, Blough DM, Redmiles EM, Ahamad M. An Inside Look into the Practice of Malware Analysis. In: proceedings of the 2021 ACM SIGSAC conference on computer and communications security. CCS '21, New York, NY, USA: Association for Computing Machinery; 2021, p. 3053–69.

[5] Bejtlich R. The practice of network security monitoring. San Francisco, CA: No Starch Press; 2013.

[6] Porras PA, Saïdi H, Yegneswaran V. A Foray into Conficker's Logic and Rendezvous Points.. LEET 2009;9:7.

[7] Plohmann D, Yakdan K, Klatt M, Bader J, Gerhards-Padilla E. A Comprehensive Measurement Study of Domain Generating Malware. In: 25th USeNIX security symposium (USeNIX security 16). Austin, TX: USENIX Association; 2016, p. 263–78.

[8] Bilge L, Kirda E, Kruegel C, Balduzzi M. Exposure: Finding Malicious Domains Using Passive DNS Analysis. In: network and distributed system security. 2011, p. 1–17.

[9] Bilge L, Sen S, Balzarotti D, Kirda E, Kruegel C. Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains. ACM Trans Inf Syst Secur 2014;16(4).

[10] Antonakakis M, Perdisci R, Nadji Y, Vasiloglou N, Abu-Nimeh S, Lee W, Dagon D. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. In: 21st USeNIX security symposium (USeNIX security 12). USENIX Association; 2012, p. 491–506.

[11] Schiavoni S, Maggi F, Cavallaro L, Zanero S. Phoenix: DGA-Based Botnet Tracking and Intelligence. In: Dietrich S, editor. detection of intrusions and malware, and vulnerability assessment. Springer International Publishing; 2014, p. 192–211.

[12] Woodbridge J, Anderson HS, Ahuja A, Grant D. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. 2016, CoRR abs/1611.00791. arXiv:1611.00791.

[13] Yu B, Gray DL, Pan J, Cock MD, Nascimento ACA. Inline DGA Detection with Deep Networks. In: 2017 IEEE international conference on data mining workshops (ICDMW). IEEE; 2017, p. 683–92.

[14] Vinayakumar R, Soman KP, Poornachandran P, Alazab M, Jolfaei A. DBD: Deep Learning DGA-Based Botnet Detection. In: Alazab M, Tang M, editors. deep learning applications for cyber security. Springer International Publishing; 2019, p. 127–49.

[15] Berman DS. DGA CapsNet: 1D Application of Capsule Networks to DGA Detection. Information 2019;10(5).

[16] Drichel A, von Querfurth B, Meyer U. Extended Abstract: A Transfer Learning-Based Training Approach for DGA Classification. In: Maggi F, Egele M, Payer M, Carminati M, editors. Detection of intrusions and malware, and vulnerability assessment. Cham: Springer Nature Switzerland; 2024, p. 381–91.

[17] Cebere BC, Flueren JLB, Sebastián S, Plohmann D, Rossow C. Down to earth! Guidelines for DGA-based Malware Detection. In: proceedings of the 27th international symposium on research in attacks, intrusions and defenses. RAID '24, New York, NY, USA: Association for Computing Machinery; 2024, p. 147–65.

[18] Wei J, Wang X, Schuurmans D, Bosma M, ichter b, Xia F, Chi E, Le QV, Zhou D. Chain-of-thought prompting elicits reasoning in large language models. In: Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K, Oh A, editors. In: Advances in neural information processing systems, vol. 35, Curran Associates, Inc.; 2022, p. 24824–37, URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

[19] Madaan A, Tandon N, Gupta P, Hallinan S, Gao L, Wiegreffe S, Alon U, Dziri N, Prabhumoye S, Yang Y, Gupta S, Majumder BP, Hermann K, Welleck S, Yazdanbakhsh A, Clark P. Self-refine: Iterative refinement with self-feedback. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, editors. In: Advances in neural information processing systems, vol. 36, Curran Associates, Inc.; 2023, p. 46534–94, URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

[20] Wang X, Wei J, Schuurmans D, Le Q, Chi E, Narang S, Chowdhery A, Zhou D. Self-consistency improves chain of thought reasoning in language models. 2023, URL https://arxiv.org/abs/2203.11171. arXiv:2203.11171.

[21] Tranco. Tranco List. 2024, [Online; https://tranco-list.eu/]. (Accessed 15 August 2024).

[22] Marvin G, Hellen N, Jjingo D, Nakatumba-Nabende J. Prompt engineering in large language models. In: Jacob IJ, Piramuthu S, Falkowski-Gilski P, editors. Data intelligence and cognitive informatics. Singapore: Springer Nature Singapore; 2024, p. 387–402.

[23] Pelayo-Benedet T, Rodríguez RJ, Gañán CH. Poster: Exploring the Zero-Shot Potential of Large Language Models for Detecting Algorithmically Generated Domains. In: Egele M, Moonsamy V, Gruss D, Carminati M, editors. Detection of intrusions and malware, and vulnerability assessment. Cham: Springer Nature Switzerland; 2025, p. 86–92.

[24] Parnami A, Lee M. Learning from Few Examples: A Summary of Approaches to Few-Shot Learning. 2022, arXiv:2203.04291.

[25] Du Y, Watkins O, Wang Z, Colas C, Darrell T, Abbeel P, Gupta A, Andreas J. Guiding Pretraining in Reinforcement Learning with Large Language Models. In: Krause A, Brunskill E, Cho K, Engelhardt B, Sabato S, Scarlett J, editors. proceedings of the 40th international conference on machine learning. Proceedings of machine learning research, 202, PMLR; 2023, p. 8657–77.

[26] Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler D, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D. Language Models are Few-Shot Learners. In: Larochelle H, Ranzato M, Hadsell R, Balcan M, Lin H, editors. advances in neural information processing systems. 33, Curran Associates, Inc.; 2020, p. 1877–901.

[27] Ran Q, Lin Y, Li P, Zhou J, Liu Z. NumNet: Machine Reading Comprehension with Numerical Reasoning. 2019, arXiv:1910.06701.

[28] Xie X, Cheng G, Li Q, Miao S, Li K, Han J. Fewer is more: efficient object detection in large aerial images. Sci China Inf Sci 2023;67(1):112106.

[29] Ye Q, Axmed M, Pryzant R, Khani F. Prompt Engineering a Prompt Engineer. 2024, arXiv:2311.05661.

[30] Sahoo P, Singh AK, Saha S, Jain V, Mondal S, Chadha A. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. 2024, arXiv:2402.07927.

[31] Alexa. Alexa Top 1 Million Domains feed. 2024, [Online; http://s3.amazonaws.com/alexa-static/top-1m.csv.zip]. (Accessed 1 December 2024).

[32] Pochat VL, Goethem TV, Joosen W. Evaluating the Long-term Effects of Parameters on the Characteristics of the Tranco Top Sites Ranking. In: 12th USeNIX workshop on cyber security experimentation and test (CSET 19). Santa Clara, CA: USENIX Association; 2019, p. 10.

[33] Bambenek. Bambenek Consulting - master feeds. 2024, [Online; https://osint.bambenekconsulting.com/feeds/]. (Accessed 1 December 2024).

[34] netlab-360. Netlab-360 feed. 2024, [Online; https://data.netlab.360.com/]. (Accessed 1 December 2024).

[35] Bader J. Domain Generation Algorithms Repository. 2024, [Online; https://github.com/baderj/domain_generation_algorithms]. (Accessed 1 December 2024).

[36] Yao Y, Duan J, Xu K, Cai Y, Sun Z, Zhang Y. A survey on large language model (LLM) security and privacy: The Good, The Bad, and The Ugly. High- Confid Comput 2024;4(2):100211.

[37] Naveed H, Khan AU, Qiu S, Saqib M, Anwar S, Usman M, Akhtar N, Barnes N, Mian A. A Comprehensive Overview of Large Language Models. 2024, arXiv:2307.06435.
[38] Anthropic. Sonnet 3.5 and Haiku 3.5. 2024, [Online; https://www.anthropic.com]. (Accessed 1 December 2024).
[39] OpenAI. GPT-4o and GPT-4o-mini. 2024, [Online; https://openai.com]. (Accessed 1 December 2024).
[40] Google. Gemini Pro 1.5, Gemini Flash 1.5 and Gemini Flash 8b 1.5. 2024, [Online; https://deepmind.google/technologies/gemini/]. (Accessed 1 December 2024).
[41] MistralAI. Mistral AI Large and Mistral AI small. 2024, [Online; https://mistral.ai/]. (Accessed 1 December 2024).
[42] Wilson EB. Probable inference, the law of succession, and statistical inference. J Amer Statist Assoc 1927;22(158):209–12. http://dx.doi.org/10.1080/01621459.1927.10502953.
[43] A.I. M. LLaMA. 2024, [Online; https://www.llama.com/]. (Accessed 1 December 2024).
[44] Wan Z. Sub-millisecond level latency sensitive cloud computing infrastructure. In: International congress on ultra modern telecommunications and control systems. 2010, p. 1194–7.