An Empirical Assessment of LLM-Based Approaches to Malicious Webpage Detection

Gracjan Mak[0009-0005-9869-1587], Mateusz Gniewkowski[1][0000-0002-0620-8123], Paweł Walkowiak[1][0009-0008-0381-9202], and Arkadiusz Janz[1][0000-0002-9203-5520]

Wrocław University of Science and Technology
`{arkadiusz.janz|mateusz.gniewkowski}@pwr.edu.pl`

Abstract. Large language models (LLMs) are increasingly influential in advancing NLP technology and solving complex tasks, yet their potential misuse in cybersecurity poses significant risks. This paper addresses the challenge of detecting malicious webpages using LLMs, an area with limited research. We evaluate LLMs by expanding zero-shot and few-shot query formulations, testing previously unassessed open-source and proprietary models, and assessing robustness under adversarial conditions. Additionally, we verify model performance using Chain of Thought reasoning and compare these explanations with traditional methods. Our work aims to enhance the application of LLMs in cybersecurity, guiding the development of more effective detection systems.

1 Introduction

The role of large language models (LLMs) in the development of NLP technology and the resolution of complex NLP tasks continues to expand, contributing to solving novel problems in various application areas. The increasing reasoning capabilities of LLMs significantly contributed to open-domain question answering, mathematical question answering, commonsense reasoning, and logical reasoning. The use of large and diverse pretraining data facilitates zero-shot classification across a wide range of application domains, extending beyond typical benchmark applications. As technology advances, the threat of LLM misuse also increases. Thus, a proper response should focus on designing methods against threats that emphasize equally sophisticated techniques and technology.

The methods for leveraging large-scale language models (LLMs) in complex reasoning have advanced significantly, with diverse inference strategies such as Tree of Thoughts [12] and other innovative approaches enhancing their application across various domains. Despite these advancements, LLMs remain susceptible to adversarial attacks, a critical factor that must be considered when designing methods for detecting malicious webpages. Malicious actors may exploit the known limitations of LLMs, necessitating robust detection strategies that account for these vulnerabilities.

Currently, the application of LLMs in the realm of cybersecurity, particularly in detecting cyber threats, is still in its early stages [7]. Moreover, there is a

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_12

limited body of work exploring both the diverse inference techniques and the susceptibility of LLMs to adversarial attacks within this context. In this paper, we aim to bridge this gap by conducting a comprehensive evaluation of LLMs for the detection of malicious webpages, considering both common prompting-based inference methods and the models' resilience to adversarial conditions. Through this work, we seek to advance the understanding and application of LLMs in cybersecurity, providing insights that could inform the development of more effective detection systems. Our contributions are as follows:

1. We introduce a novel dataset for researching malicious webpage detection in an adversarial setting, including attacks at 5%, 25%, and 50% of page content change, as well as prompt injection and compromised URLs.
2. We extend the evaluation of zero-shot and few-shot approaches by expanding the query formulations, providing a more comprehensive assessment of model capabilities.
3. We test open source and proprietary models that have not been previously evaluated for this task, broadening the scope of model applicability.
4. We incorporate an evaluation under adversarial attack conditions and present the results of this evaluation across different models, offering insights into their robustness.
5. Finally, we verify the performance of the models using Chain of Thought (CoT) reasoning.
6. We examine the results obtained with traditional explainability methods.

2 Related Work

Before the advent of Large Language Models (LLMs), malicious website detection relied heavily on rule-based systems, statistical analysis, and then on traditional machine learning. These approaches were tailored to specific features of malicious activity, often requiring extensive domain knowledge and feature engineering.

Supervised models, including Support Vector Machines (SVMs), Decision Trees, and Random Forests, used labeled datasets to identify malicious webpages based on extracted features. Commonly used features included URL length, entropy, keyword patterns, metadata, and network traffic behaviors. These models achieved higher detection rates and better adaptability compared to rule-based systems. However, they were highly dependent on manual feature engineering, which required significant expertise and often failed to generalize across diverse or evolving threats. For example, in [5], the authors utilize lexical features and compare a wide set of machine learning algorithms. Similar approaches can be found in [11] or in [4].

In the early stages of applying natural language processing (NLP) to cybersecurity, text classification tasks such as protocol analysis, website categorization, and source code identification played a critical role. These tasks relied on foundational techniques designed to process and interpret text in a structured manner. For example, spam detection systems leveraged basic NLP techniques like bag-of-words models or Term Frequency-Inverse Document Frequency (TF-IDF). Later advances in NLP introduced more sophisticated methods, taking advantage of deep learning-based models such as doc2vec, FastText, and transformers such as BERT [2]. These approaches significantly improved the ability, among others, to analyze and classify web content by capturing richer semantic and contextual information. The works in this field include: [8,10,3].

In recent years, large language models (LLMs) such as GPT or Llama have revolutionized the field of cybersecurity by offering powerful tools for analyzing and processing textual data. Their ability to understand context, generate coherent text, and adapt to various tasks with minimal fine-tuning has made them invaluable across diverse cybersecurity applications. Zero-Shot Learning and Few-Shot Learning deserve special attention. In the context of malicious webpage detection, [7] proposed a dataset for benchmarking LLMs in this task. The authors conducted a study in which they tested diverse prompting strategies to detect malicious webpages using LLMs. However, the dataset is not publicly available, and the authors limited their prompting strategies to few-shot learning, without exploring Chain-of-Thought (CoT) reasoning, explainability, or adversarial attacks.

3 Methods

In this work, we focus on the webpage content and URL classification task. One can use large-scale language models that offer various approaches to classification. In this study, these classification methods will be analysed in terms of their effectiveness and robustness against adversarial attacks. This chapter outlines the key methods and their adaptations to the problem at hand, forming the foundation for the experimental part of the study. For the experimental setup, we introduced a pipeline which is presented in Figure 1. For each of two Webpage datasets, we created their adversarial setting versions with gradually attacked content. Our experiments for five size diversified LLMs (Llama3.1-8B[1], Llama3.1-70B[2], GPT4o-mini[3], GPT4o[4] and Gemini2.0-flash-exp[5]), includes checking their ability to classify Webpage maliciousness given:

- The **URL** of the webpage, in this setting we test if the model remembers some of the websites and to what extent it bases prediction on the `URL`.
- The Webpage **Content**, which tests how well a model can analyse html files.
- The **URL and Content**, model receives all the information from datasets, this setting also shows if there is a bias basing the prediction towards `URL` or content. In this setting, we also included the website JavaScript sections.

In order to test the models in most diverse way, we prepared various Zero- and Few-Shot versions of prompts. The URL setting was tested with **Simple**

[1] `https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`
[2] `https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct`
[3] `https://platform.openai.com/docs/models#gpt-4o-mini`
[4] `https://platform.openai.com/docs/models#gpt-4o`
[5] `https://ai.google.dev/gemini-api/docs/models/experimental-models`

Zero-Shot, **CoT Zero-Shot**, **Advanced Zero-Shot**, **Simple Few-Shot** and **Advanced Few-Shot** (see Section 3.1). In the Content setting only CoT Zero-Shot was used. Finally, the URL and content was tested with Simple and CoT Zero-Shots.

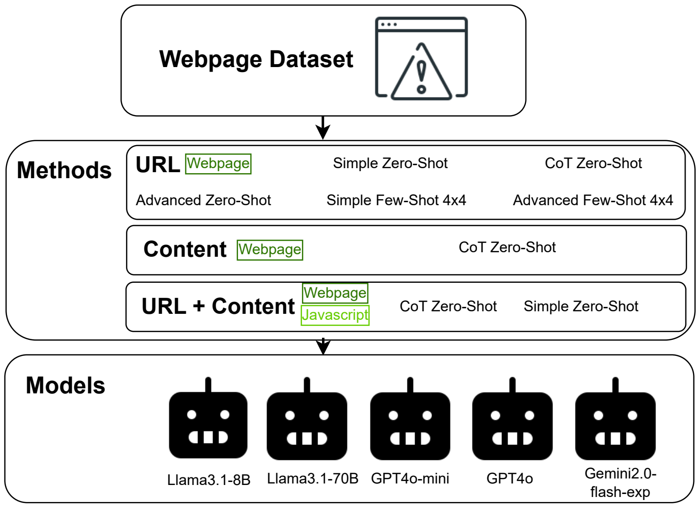


Fig. 1. Experimental pipeline for the Webpages dataset. Each dataset sample was evaluated using two primary methods: one based only on the URL and another incorporating both the URL and page content. Each method further includes sub-methods that leverage few-shot learning and chain-of-thought reasoning. Five different LLMs were tested to ensure diversity in model size and performance assessment.

3.1 Malicious Webpage Detection

The experiments on webpage classification involve inputting data, in the form of instructions containing `URL`, webpage content, or both, into large-scale language models. The models perform classification by executing the task specified in the prompt, predicting a label for the website as malicious or benign.

Zero-shot classification In this setting, the model is provided with a detailed task description, including classification criteria, target classes, purpose, and clear instructions on the expected output format. Zero-Shot learning is a technique in natural language processing where a language model performs a task without prior specialized training, relying solely on its general knowledge from pretraining.

This approach is applied to webpage classification, where models process instructions and classify the site as malicious or benign. The technique supports both simple queries, where the model classifies with minimal context, and advanced queries, which provide additional definitions of malicious and safe websites. We consider two variants of Zero-shot setting: **(1) Simple Zero-shot**, where the model receives plain instructions about the task without any demonstrations of task execution, **(2) Advanced Zero-shot**, where the model is given with precise instructions about task execution and an explanation of expected URL and content violation techniques, **(3) Zero-shot CoT**, in which the model has to execute the task explaining every single step of its reasoning.

System prompt
You will be provided with a website URL and its content, and your task is to classify it as either malicious or benign.
Criteria for Classification:
- `Malicious`: The URL leads to a website that hosts malware, phishing, scams
- `Benign`: The URL leads to a safe website that does not pose any threat to users.

Purpose: Filtering harmful websites to protect users and systems.
Response Format: Your response should be :
(A) Malicious: If the URL is harmful.
(B) Benign: If the URL is safe.

User prompt
Please classify the following URL **`http://chatgpt.trojanxx.xyz`**, and its content: <**`content`**>

Response
(A) Malicious

Fig. 2. Zero-shot prompt for LLM-based approach to `URL` classification.

Few-shot classification This technique involves providing the model with a small number of examples related to the task it is expected to perform, which helps improve the accuracy of its output. By examining these examples, the model can better grasp the context and requirements of the task. For this method, prompts can be created by extending the Zero-Shot approach with examples randomly selected from the dataset. Following [7], we designed a Few-Shot prompt consisting of one benign and three malicious examples from our dataset, to guide the model's classification. In our work, we considered both **(1) Simple Few-shot** and **(2) Advanced Few-shot** approaches based on Zero-shot prompts (see Figure 1). This structured approach aims to enhance the model's ability to distinguish between malicious and benign websites more effectively. The final prompt used in our few-shot approach is presented in Figure 3.

Chain-of-Thought The Chain-of-Thought (CoT) prompting technique aims to improve the reasoning capabilities of language models by encouraging step-by-step, logical problem-solving. Instead of providing an immediate answer, the model is prompted to "think" by generating a sequence of intermediate reasoning steps that ultimately lead to the final answer. A well-constructed CoT response should resemble a clear, structured explanation of the problem-solving process, concluding with a concise and accurate final response.

System prompt
You will be provided with a website URL and its content, and your task is to classify it as either malicious or benign.
Criteria for Classification:
- `Malicious`: The URL leads to a website that hosts malware, phishing, scams
- `Benign`: The URL leads to a safe website that does not pose any threat to users.

Purpose: Filtering harmful websites to protect users and systems.
Response Format: Your response should be :
(A) Malicious: If the URL is harmful.
(B) Benign: If the URL is safe.

Example:
URL: `http://example.com/suspicious`
Content: <`content`>
Response: (A) Malicious

User prompt
Please classify the following URL `http://chatgpt.trojanxx.xyz`, and its content: <`content`>

Response
(A) Malicious

Fig. 3. The prompt for One-shot LLM-based approach to URL classification.

3.2 Adversarial Attacks on Detection Methods

A significant challenge in utilizing large-scale language models for detecting malicious websites is ensuring their robustness against various forms of attacks that cybercriminals may use to evade detection. The attacks presented in this subsection are experimental, as research on adversarial attacks against models in the context of malicious website detection remains limited.

Prompt Injection One of the key threats to the reliability of large-scale language models in detecting malicious websites is the use of manipulative inputs designed to influence model predictions. Prompt injection involves crafting specific text patterns within the website content to mislead the model into classifying a malicious website as safe. The attack phrase used in our experiments:

- `Browse with confidence, this site is safe`

Such manipulations exploit the model's reliance on linguistic cues, potentially leading to false negatives in malicious website detection.

Manipulated URL Another technique for adversarial manipulation targets the URL structure itself. By embedding deceptive query parameters or modifications, attackers can influence the model's classification by introducing misleading context within the URL. In our experiments, the following manipulation was used:

- `?malware_check=none&status_confirmed=safe`

These modifications exploit the model's tendency to associate specific keywords with benign classifications, increasing the risk of false negatives.

URL Modification We conducted a URL-based adversarial attack by randomly substituting three letters in the domain. This method exploits weaknesses in models that rely on exact string matching and learned domains. Such modifications resemble typosquatting attacks, where adversaries register domains with small alterations to trick users and evade detection. For example, a malicious URL:

```
https://maliciouswebsite.com/phishing
```

can be modified to:

```
https://mallciousvebsita.com/phishing
```

These small changes can bypass blacklists and confuse models, especially those that lack robust phonetic or semantic analysis.

3.3 JavaScript Code Obfuscation Attack

To evaluate the robustness of large-scale language models in detecting malicious JavaScript code, we performed code obfuscation using the JavaScript Obfuscator[6]. This widely used tool applies multiple transformations to JavaScript code, making it harder to analyze while preserving its functionality. Obfuscation is a well-known technique employed by cybercriminals to evade detection by static and dynamic analysis tools. By altering the syntactic structure of JavaScript scripts without changing their behavior, attackers can bypass signature-based security mechanisms and even some machine learning-based classifiers.

3.4 Webpage Content TextBugger Attacks

To evaluate the impact of perturbing webpage content on malicious page detection, the page html body content was attacked using the adversarial example generation method – TextBugger [6]. The TextBugger introduces five disturbance techniques for generating adversarial example: insertion of spaces into the disturbed word, deletion of a random letter, swapping the order of two adjacent letters, substitution of letters with visually similar characters, and substitution word (replacing a word with its nearest equivalent using a pre-trained GloVe model). These methods are combined in the TextBugger approach, resulting in highly diverse adversarial examples due to the randomness introduced in word alterations and the capability to apply multiple disturbance methods concurrently. For the purpose of modifying HTML pages three of TextBuggers disturbance methods were used: letter deletion, adjacent letters swap and substitution of letters to its visually similar equivalent. The modification of the html pages consisted of changing the natural language texts in the body section where scripts and html tags have remained intact. In order to check model's ability for classifying modified pages, three levels of disruption were chosen, modification of 5%, 25% and 50% of words in page text.

[6] https://github.com/javascript-obfuscator/javascript-obfuscator

4 Datasets

The literature offers numerous datasets for detecting malicious websites, but most of them typically do not include the content of the pages, focusing solely on their `URL` addresses. The popularity of URL-only datasets comes from their ease of creation and maintenance, as domain names are relatively static, persisting even after a website becomes inactive, which simplifies the process of dataset creation. In contrast, webpage content becomes inaccessible once the server hosting it is taken offline, making it imperative to capture such data while the malicious page is still active. A Comprehensive Dataset for Webpage Classification [1] comprises 1,069,715 webpages and is specifically created for developing algorithms to detect malicious websites. Each webpage in the dataset is annotated with multiple attributes, including URL, HTML code and source. The dataset was generated automatically from existing lists of malicious websites. However, such sites are often taken down quickly, leading to potential false positives, such as inactive domains that have been repurchased. In contrast, our dataset consists of manually verified, fresh malicious webpages, ensuring higher accuracy and relevance for real-world detection.

4.1 Adversarial Benchmark for Malicious Webpage Detection (ABW)

Given the limited availability of datasets, we decided to create our own dataset, smaller but more reliable. The approach focused on collecting data from various sources, ensuring a balanced set of malicious and benign websites.

For the malicious websites, we gathered URLs from sources such as OpenPhish[7], PhishTank[8], and email spam. These sources are widely recognized for their up-to-date lists of potentially dangerous sites. After collecting the URLs, each website was manually verified to confirm that it was malicious. For the benign websites, we selected a random sample of frequently visited and well-established pages from across the internet. These sites were chosen to represent a broad range of trustworthy, legitimate content. The dataset ultimately consists of 110 benign websites and 108 malicious websites. Each entry in the dataset contains the following attributes:

- `url`: the webpage's URL,
- `label`: the primary category indicating whether the webpage is safe or malicious,
- `source`: the origin of the URL (e.g., OpenPhish, PhishTank, etc.),
- `html`: the raw `HTML` content of the webpage, including HTML code and embedded `JavaScript`.

To evaluate the model's performance under adversarial attack, three versions of the adversarial dataset were created, with 5%, 25%, and 50% of content words in the page text modified.

[7] `https://openphish.com/`
[8] `https://phishtank.org/`

4.2 JavaScript Dataset

In addition to the webpage dataset, we also created our own dataset of JavaScript files. This dataset was designed to facilitate the analysis and classification of JavaScript code as either malicious or benign. For the malicious JavaScript files, we utilized samples from the publicly available repository JavaScript Malware Collection[9], which contains a wide variety of confirmed malicious scripts. For the benign JavaScript files, we randomly collected scripts by accessing legitimate and frequently visited websites, ensuring a diverse representation of safe JavaScript code. The dataset is structured with the following attributes for each record:

– `filename`: the name of the JavaScript file,
– `label`: a classification label indicating whether the script is benign or malicious,
– `source`: the origin of the script (e.g., *JavaScript Malware Collection* or a legitimate website),
– `javascript`: the raw content of the JavaScript file.

The resulting dataset consists of 100 benign JavaScript files and 100 malicious JavaScript files.

5 Experimental Setting

The experiments were conducted using the following large language models: Llama 3.1 (7B and 70B), GPT-4o-Mini, GPT-4o, and Gemini 2.0 Flash Exp. A default temperature was applied to maintain a balance between creativity and consistency in the model's responses. The models were evaluated on presented datasets consisting of URLs, webpage content, and JavaScript scripts. The experimental pipeline for webpage datasets is presented on Figure 1. To ensure efficient processing, the HTML and JavaScript datasets were truncated to a maximum of 20 thousand tokens, preserving representative information while minimizing cost and processing time. For few-shot classification, four randomly selected examples from the dataset were added to the prompt to enhance the model's contextual understanding and classification accuracy.

6 Results and Discussion

Table 1 presents the classification results of various large language models for attack detection using the previously mentioned strategies. The evaluation was conducted using several publicly available models based on True Positive Rate (TPR), True Negative Rate (TNR), and the F1 score.

The Few-Shot Learning approach, in both its Simple and Advanced variants, demonstrates the best overall performance. Their main drawback is the need to provide several examples, which can make the model less robust to concept

[9] https://github.com/HynekPetrak/javascript-malware-collection

Table 1. The performance of language models in the task of malicious webpage classification. We provide the results for particular variants of prompting-based detection methods, for both content and URL classification.

Method	Model	Metrics		
		TPR	TNR	F1
Simple Zero-Shot (URL)	Llama-3.1-8B-Instruct	0.22	0.38	0.23
	Llama-3.1-70B-Instruct	0.51	1.00	0.68
	gpt-4o-mini	0.82	0.97	0.89
	gpt-4o	0.23	0.99	0.37
	gemini-2.0-flash-exp	0.45	1.00	0.62
	AVG	0.45	0.87	0.56
Advanced Zero-Shot (URL)	Llama-3.1-8B-Instruct	0.44	0.94	0.58
	Llama-3.1-70B-Instruct	0.33	0.99	0.50
	gpt-4o-mini	0.81	1.00	0.89
	gpt-4o	0.54	1.00	0.70
	gemini-2.0-flash-exp	0.70	1.00	0.82
	AVG	0.56	0.99	0.70
Simple Few-Shot (URL)	Llama-3.1-8B-Instruct	0.74	0.47	0.65
	Llama-3.1-70B-Instruct	0.64	0.97	0.77
	gpt-4o-mini	0.73	1.00	0.85
	gpt-4o	0.89	0.99	0.93
	gemini-2.0-flash-exp	0.76	1.00	0.86
	AVG	0.75	0.89	0.81
Advanced Few-Shot (URL)	Llama-3.1-8B-Instruct	0.74	0.9	0.80
	Llama-3.1-70B-Instruct	0.66	0.98	0.78
	gpt-4o-mini	0.78	0.99	0.87
	gpt-4o	0.70	1.00	0.83
	gemini-2.0-flash-exp	0.84	0.99	0.91
	AVG	0.74	0.97	0.84
CoT Zero-Shot (URL)	Llama-3.1-8B-Instruct	0.31	0.81	0.42
	Llama-3.1-70B-Instruct	0.37	0.96	0.53
	gpt-4o-mini	0.75	1.00	0.86
	gpt-4o	0.50	0.98	0.65
	gemini-2.0-flash-exp	0.75	0.99	0.85
	AVG	0.54	0.95	0.66
Simple Zero-Shot (URL + Content)	Llama-3.1-8B-Instruct	0.27	0.11	0.24
	Llama-3.1-70B-Instruct	0.30	0.95	0.44
	gpt-4o-mini	0.80	0.95	0.87
	gpt-4o	0.94	0.95	0.95
	gemini-2.0-flash-exp	0.77	1.00	0.87
	AVG	0.62	0.79	0.64
CoT Zero-Shot (URL + Content)	Llama-3.1-8B-Instruct	0.23	0.16	0.22
	Llama-3.1-70B-Instruct	0.30	0.97	0.45
	gpt-4o-mini	0.88	1.00	0.93
	gpt-4o	0.90	1.00	0.95
	gemini-2.0-flash-exp	0.93	1.00	0.97
	AVG	0.65	0.83	0.70
CoT Zero-Shot (Content)	Llama-3.1-8B-Instruct	0.18	0.18	0.15
	Llama-3.1-70B-Instruct	0.13	0.21	0.64
	gpt-4o-mini	0.53	0.69	0.81
	gpt-4o	0.68	0.80	0.86
	gemini-2.0-flash-exp	0.69	0.81	0.87
	AVG	0.44	0.54	0.67

Table 2. The performance of language models in the task of malicious JavaScript files detection.

Method	Model	Metrics		
		TPR	TNR	F1
Simple Zero-Shot (Content)	Llama-3.1-8B-Instruct	0,56	0,66	0,59
	Llama-3.1-70B-Instruct	0,89	0,85	0,87
	gpt-4o-mini	1,00	0,69	0,87
	gpt-4o	1,00	0,96	0,98
	gemini-2.0-flash-exp	1,00	0,83	0,92
	AVG	0,89	0,80	0,85
CoT Zero-Shot (Content)	Llama-3.1-8B-Instruct	0,46	0,56	0,48
	Llama-3.1-70B-Instruct	0,92	0,84	0,88
	gpt-4o-mini	0,94	0,74	0,85
	gpt-4o	0,94	0,96	0,95
	gemini-2.0-flash-exp	0,85	0,91	0,88
	AVG	0,82	0,80	0,80

drift—deviations from the patterns present in the provided examples. However, in general, they should be used as the preferred approach because of their superior performance in most cases. The next best results were achieved by the Advanced Zero-Shot method (based solely on URL addresses) and CoT Zero-Shot Learning, but only when both URL addresses and page content were used. It is important to note that classification using LLM models can be expensive, so the benefits of employing more advanced methods seem marginal, considering that the best classification results were obtained with just URL addresses (shorter queries are less costly). We hope to demonstrate further that despite this limitation, CoT methods and the use of content may offer greater robustness. Similar conclusions can be drawn from Table 2. It also shows that the simple Zero-Shot Learning approach outperforms the CoT.

Table 3 presents the effectiveness of the attacks in the form of new metric values, together with the differences between these values and the corresponding ones in Table 1. Looking solely at the F1 or TPR values, the most effective attack method (or the most unstable method) turns out to be the simple prompt injection into the URL address (Simple Zero-Shot). Apart from this attack, all other examples show that any modifications to the URL or page content lead to an increase in TPR and a decrease in TNR. This means that the classification becomes overly sensitive and starts classifying everything as a threat (indicating that the methods are effectively detecting the attack). This is most noticeable in the case of methods that directly modify the URL addresses (Simple Zero-Shot [URL] and CoT Zero-Shot [URL]). This supports our previous observation that the model is most focused on the URL. In response to the question of whether using the content of the page is justified, we believe the answer is yes. Classification results are harder to alter when the content is included, as it

Table 3. The performance of large language models in the task of malicious webpage detection in adversarial setting.

Method & Attack	Model	Metrics TPR	TNR	F1
Simple Zero-Shot (URL) Manipulated URL	Llama-3.1-8B-Instruct	0.14/−0.08	0.38/−0.00	0.16/−0.07
	Llama-3.1-70B-Instruct	0.68/+0.17	0.99/−0.01	0.80/+0.12
	gpt-4o-mini	0.50/−0.32	1.00/+0.03	0.66/−0.23
	gpt-4o	0.19/−0.04	0.94/−0.05	0.31/−0.06
	gemini-2.0-flash-exp	0.30/−0.15	1.00/−0.00	0.46/−0.16
	AVG	0.36/−0.09	0.86/−0.01	0.48/−0.08
Simple Zero-Shot (URL − Content) Manipulated URL Prompt injection	Llama-3.1-8B-Instruct	0.21/−0.06	0.15/+0.04	0.20/−0.04
	Llama-3.1-70B-Instruct	0.35/+0.05	0.96/+0.01	0.51/+0.07
	gpt-4o-mini	0.69/−0.11	0.97/+0.02	0.80/−0.07
	gpt-4o	0.92/−0.02	0.92/−0.03	0.92/−0.03
	gemini-2.0-flash-exp	0.79/+0.02	0.99/−0.01	0.88/+0.01
	AVG	0.59/−0.03	0.80/+0.01	0.66/+0.02
Simple Zero-Shot (URL) URL Modification	Llama-3.1-8B-Instruct	0.23/+0.01	0.30/−0.08	0.23/−0.00
	Llama-3.1-70B-Instruct	0.78/+0.27	0.67/−0.33	0.74/+0.06
	gpt-4o-mini	0.99/+0.17	0.31/−0.66	0.73/−0.16
	gpt-4o	0.24/+0.01	0.21/−0.78	0.23/−0.14
	gemini-2.0-flash-exp	0.69/+0.24	0.83/−0.17	0.73/+0.11
	AVG	0.59/+0.14	0.46/−0.40	0.53/−0.03
CoT Zero-Shot (URL) URL Modification	Llama-3.1-8B-Instruct	0.66/+0.35	0.42/−0.39	0.58/+0.16
	Llama-3.1-70B-Instruct	0.56/+0.19	0.64/−0.32	0.58/+0.05
	gpt-4o-mini	0.93/+0.18	0.28/−0.72	0.70/−0.16
	gpt-4o	0.78/+0.28	0.57/−0.41	0.70/+0.05
	gemini-2.0-flash-exp	0.94/+0.19	0.45/−0.54	0.75/−0.10
	AVG	0.77/+0.24	0.47/−0.48	0.66/−0.00
CoT Zero-Shot (URL - Content) Prompt injection	Llama-3.1-8B-Instruct	0.44/+0.21	0.07/−0.09	0.37/+0.15
	Llama-3.1-70B-Instruct	0.49/+0.19	0.86/−0.11	0.60/+0.15
	gpt-4o-mini	0.90/+0.02	0.94/−0.06	0.92/−0.01
	gpt-4o	0.93/+0.03	0.99/−0.01	0.96/+0.01
	gemini-2.0-flash-exp	0.95/+0.02	0.81/−0.19	0.88/−0.09
	AVG	0.74/+0.09	0.73/−0.09	0.75/+0.04
CoT Zero-Shot (Content) Prompt injection	Llama-3.1-8B-Instruct	0.28/+0.10	0.04/−0.11	0.24/+0.06
	Llama-3.1-70B-Instruct	0.30/+0.17	0.80/−0.07	0.40/+0.19
	gpt-4o-mini	0.69/+0.16	0.98/−0.01	0.80/+0.11
	gpt-4o	0.72/+0.04	0.94/−0.04	0.81/+0.01
	gemini-2.0-flash-exp	0.83/+0.14	0.82/−0.18	0.82/+0.01
	AVG	0.56/+0.12	0.72/−0.08	0.61/+0.08
CoT Zero-Shot (Content) 25% of page modified	Llama-3.1-8B-Instruct	0.17/−0.01	0.13/−0.02	0.17/−0.01
	Llama-3.1-70B-Instruct	0.26/+0.13	0.73/−0.14	0.40/+0.19
	gpt-4o-mini	0.54/+0.01	0.80/−0.19	0.68/−0.01
	gpt-4o	0.73/+0.05	0.83/−0.15	0.80/−0.00
	gemini-2.0-flash-exp	0.76/+0.07	0.86/−0.14	0.83/+0.02
	AVG	0.49/+0.05	0.67/−0.13	0.58/+0.04

Table 4. The performance of large language models in the task of malicious JavaScript detection in adversarial setting.

Method	Model	Metrics TPR	TNR	F1
Simple Zero-Shot (Content)	Llama-3.1-8B-Instruct	0,51/−0,05	0,28/−0,38	0,46/−0,13
	Llama-3.1-70B-Instruct	0,83/−0,06	0,63/−0,22	0,75/−0,12
	gpt-4o-mini	1.00/−0,00	0,06/−0,63	0,68/−0,19
	gpt-4o	1.00/−0,00	0,30/−0,66	0,74/−0,24
	gemini-2.0-flash-exp	1.00/−0,00	0,14/−0,69	0,70/−0,22
	AVG	0,87/−0,02	0,28/−0,52	0,67/−0,18
CoT Zero-Shot (Content)	Llama-3.1-8B-Instruct	0,50/+0,04	0,14/−0,42	0,42/−0,06
	Llama-3.1-70B-Instruct	0,85/−0,07	0,77/−0,07	0,82/−0,06
	gpt-4o-mini	0,91/−0,03	0,05/−0,69	0,64/−0,21
	gpt-4o	0,95/+0,01	0,42/−0,54	0,75/−0,20
	gemini-2.0-flash-exp	0,95/+0,10	0,35/−0,56	0,73/−0,15
	AVG	0,83/+0,01	0,35/−0,46	0,67/−0,14

provides additional context that helps the model make more stable and accurate predictions. This is proven by looking at CoT attacks that includes content.

In our study, our objective was to investigate why the model classifies websites in a particular way. We applied explainability techniques to highlight key elements that influence classification. The explainability was assessed for the classifications generated using an advanced few-shot prompting approach with the GPT-4o-mini model and SHAP values [9] obtained using logprobs for a specific tokens (related to classes). In Figure 4 (`https://brawllstar.ru`), the substring "ll" was emphasized (red indicates correlation with malicious class), suggesting the model associates certain character sequences with specific website types. In this case, "ll" appears to be an unusual repetition within the word 'brawl', which may indicate a typographical deviation from the expected "brawlstars", potentially signaling a deceptive domain. Similarly, in Figure 5 (`https://vulkanbet-offers.com`), the terms "-" and "offers" were highlighted. The presence of a hyphen in domain names is less common among legitimate brand domains, since major companies typically register names without hyphens for credibility and ease of recall. The emphasis of the model on these elements suggests an association between such patterns and potentially lower-reputation websites. Even when webpage content is present, the model still prioritizes the URL, but also considers semantic elements. In Figure 6 (`https://octoplazmatic.ru`), despite the presence of textual data, the model focuses mainly on the URL, although phrases like "app with a focus" were marked as contributing to a benign classification. This suggests that while URL-based patterns are dominant, meaningful content can influence classification decisions. Notably, the model's strong reliance on URLs is a global issue; we observed similar behavior across numerous explained samples.

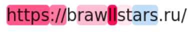
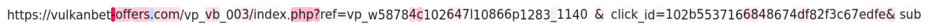
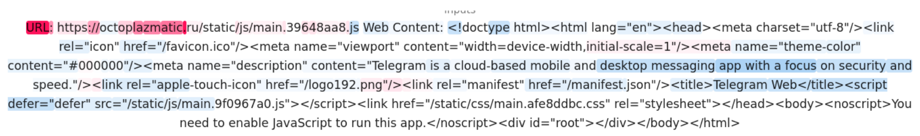

https://brawllstars.ru/

Fig. 4. Explainability of classification for `https://brawllstar.ru` without content.



```
https://vulkanbet-offers.com/vp_vb_003/index.php?ref=vp_w58784c102647l10866p1283_1140 & click_id=102b5537166848674df82f3c67edfe& sub
```

Fig. 5. Explainability of classification for `https://vulkanbet-offers.com/vp_vb_003/index.php?` without content.



```
URL: https://octoplazmatic.ru/static/js/main.39648aa8.js Web Content: <!doctype html><html lang="en"><head><meta charset="utf-8"/><link
rel="icon" href="/favicon.ico"/><meta name="viewport" content="width=device-width,initial-scale=1"/><meta name="theme-color"
content="#000000"/><meta name="description" content="Telegram is a cloud-based mobile and desktop messaging app with a focus on security and
speed."/><link rel="apple-touch-icon" href="/logo192.png"/><link rel="manifest" href="/manifest.json"/><title>Telegram Web</title><script
defer="defer" src="/static/js/main.9f0967a0.js"></script><link href="/static/css/main.afe8ddbc.css" rel="stylesheet"></head><body><noscript>You
need to enable JavaScript to run this app.</noscript><div id="root"></div></body></html>
```

Fig. 6. Explainability of classification for `https://octoplazmatic.ru/static/js/main.39648aa8.js` with content.

7 Conclusions

In this paper, we analyze the performance of various prompting-based approaches (Zero-Shot, Few-Shot, and CoT) in the tasks of malicious webpage detection and malicious JavaScript file classification. In both tasks we evaluated several language models, by introducing a novel dataset for these tasks. Our extensive evaluation revealed that Few-Shot learning approach delivers the best overall performance. We also explored classification based on solely the webpage URLs as well as joint classification using both URL and webpage content. Among Zero-Shot methods, the advanced prompt applied to URLs performed best, while incorporating page content through CoT reasoning improved results, especially when both URL and content were considered together. Furthermore, we examined the impact of different attack methods, observing that modifications to the URL, especially in Simple Zero-Shot, led to higher True Positive Rates (TPR), but also increased False Positives. Including content in the queried LLM stabilizes the classification, reducing false positives and improving accuracy. Finally, through explainability analysis, we explored how the model makes its decisions, finding that it primarily relies on URL patterns but also uses page content to refine its predictions. This highlights the value of combining both URL and content for more robust and accurate attack detection.

The research in this paper focuses on advanced computational techniques, particularly LLM-based inference techniques, for web-oriented classification. By applying LLMs to malicious webpage detection, this work might contribute to AI's role in scientific computing and has potential to enhance cybersecurity in web environments.

Ethics Statement

This study applied large language models for the detection of malicious web pages and JavaScript files, using publicly available data sets in compliance with data protection regulations. We acknowledge the ethical implications of AI in cybersecurity, aiming to improve threat detection and minimize false positives for safer web environments. The research did not involve human participants, and no informed consent was required for the use of public data.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_12

Acknowledgements

Financed by: (1) CLARIN ERIC (2024–2026), funded by the Polish Minister of Science (agreement no. 2024/WK/01); (2) CLARIN-PL, the European Regional Development Fund, FENG programme (FENG.02.04-IP.040004/24); (3) statutory funds of the Department of Artificial Intelligence, Wroclaw Tech; (4) the EU project "DARIAH-PL", under investment A2.4.1 of the National Recovery and Resilience Plan.

References

1. Al-Maamari, M., Istaiti, M., Zerhoudi, S., Dinzinger, M., Granitzer, M., Mitrovic, J.: A comprehensive dataset for webpage classification (part 1: Adult & malicious) (version 1) (Mar 2024). https://doi.org/10.5281/ZENODO.10775260
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies. pp. 4171–4186 (2019)
3. Gniewkowski, M., Maciejewski, H., Surmacz, T., Walentynowicz, W.: Sec2vec: Anomaly detection in http traffic and malicious urls. In: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing. pp. 1154–1162 (2023)
4. Gupta, B.B., Yadav, K., Razzak, I., Psannis, K., Castiglione, A., Chang, X.: A novel approach for phishing urls detection using lexical based machine learning in a real-time environment. Computer Communications **175**, 47–57 (2021)
5. Johnson, C., Khadka, B., Basnet, R.B., Doleck, T.: Towards detecting and classifying malicious urls using deep learning. J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl. **11**(4), 31–48 (2020)
6. Li, J., Ji, S., Du, T., Li, B., Wang, T.: Textbugger: Generating adversarial text against real-world applications. In: 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019. The Internet Society (2019), https://www.ndss-symposium.org/ndss-paper/textbugger-generating-adversarial-text-against-real-world-applications/
7. Li, L., Gong, B.: Prompting large language models for malicious webpage detection. In: 2023 IEEE 4th International Conference on Pattern Recognition and Machine Learning (PRML). pp. 393–400 (2023). https://doi.org/10.1109/PRML59573.2023.10348229
8. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
9. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Advances in neural information processing systems **30** (2017)
10. Luo, C., Tan, Z., Min, G., Gan, J., Shi, W., Tian, Z.: A novel web attack detection system for internet of things via ensemble classification. IEEE Transactions on Industrial Informatics (2020)
11. Nguyen, H.T., et al.: Application of the generic feature selection measure in detection of web attacks. In: Computational Intelligence in Security for Information Systems, pp. 25–32. Springer (2011)
12. Yao, S., other: Tree of thoughts: deliberate problem solving with large language models. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. NIPS '23, Curran Associates Inc., Red Hook, NY, USA (2023)

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_12