Ghosts in the Markup: Techniques to Fight Large Language Model-Powered Web Scrapers

William Brach, Matej Petrik, Kristián Košťál, Michal Ries
Faculty of Informatics and Information Technologies, Slovak Technical University
Bratislava, Slovakia
{william.brach,matej.petrik,kristian.kostal,michal.ries}@stuba.sk}

***Abstract*—The rise of large language models (LLMs) has revolutionized web scraping, creating new challenges for website owners seeking to protect valuable content. Traditional defenses such as CAPTCHAs and rate limiting are insufficient against sophisticated AI-driven scraping systems. Current state-of-the-art approaches primarily rely on server-side implementations requiring significant computational resources, IP-based blocking mechanisms, and complex pattern recognition systems that struggle to distinguish between legitimate users and advanced AI scrapers. This paper presents a two-step client-side defense strategy that combines content obfuscation and defensive prompt injection. Our approach dynamically alters HTML structure by wrapping characters in span elements with randomized content and strategic CSS rules while simultaneously injecting carefully crafted prompts designed to manipulate LLM extraction behavior. Experimental evaluation on three diverse datasets demonstrates the effectiveness of this combined approach, achieving near-zero extraction rates (0.6% Exact Match, 0.030 ROUGE-L) for structured content while maintaining a seamless user experience. Performance analysis shows that our method increases the HTML document file size by approximately 1773%, compared to over 6593% for alternative techniques such as Content Overload, providing a practical balance between security and usability while keeping rendering times comparable to unprotected pages. This lightweight implementation integrates seamlessly with existing Web infrastructure without requiring significant computational resources, addressing key limitations of server-side defenses while providing robust protection against unauthorized LLM-based content extraction. For further information, source code, and associated resources, please refer to the source code repository[1].**

I. Introduction

The rising popularity of AI, particularly large language models (LLMs), has transformed web scraping and data extraction. These AI models now enable automated systems to mine and gather information from websites with unprecedented ease and efficiency. This technological advancement has created new challenges for website owners who need to protect their content from unauthorized AI-powered scraping.

This challenge has significant implications for both the AI industry and content providers. While LLMs require extensive web-crawled text data for training, and recent research [1] has demonstrated that data quality is crucial for their performance, website owners with valuable content are reluctant to make it freely available for AI training. Moreover, protecting content has become increasingly critical from a business perspective, as AI-powered tools can now conduct comprehensive market research and due diligence in minutes, potentially eliminating competitive advantages.

Balancing content protection against AI-powered scraping while maintaining accessibility for human users and search engine optimization (SEO) presents significant challenges. Traditional anti-bot measures such as CAPTCHA [2] or rate limiting have proven inadequate against sophisticated AI systems that can bypass these protections by mimicking human behavior patterns. Additionally, implementing stringent protection mechanisms risks compromising user experience for legitimate visitors and potentially damaging website SEO rankings.

Previous approaches to AI-scraping protection have primarily relied on complex server-side implementations, requiring substantial computational resources and technical expertise for maintenance. Our research presents a novel approach: a lightweight, client-side solution that integrates seamlessly with existing web infrastructure while providing robust protection against AI-powered scraping attempts. This solution addresses the key limitations of existing approaches by minimizing implementation complexity and optimizing human-computer interaction.

This paper is organized as follows: Section II reviews existing web scraping prevention methods and their limitations. Section III details our approach combining content obfuscation and prompt injection. Section IV describes our experimental methodology. Section V presents our findings, with Section VI addressing implications and limitations. Section VII summarizes contributions and suggests future research directions.

II. Related work

Web scraping plays a crucial role in collecting large-scale training data for machine learning, particularly for Large Language Models (LLMs). This web data serves as the backbone for training foundation models, as we can see in popular open-source LLM training datasets such as FineWeb [3], RefinedWeb [1], C4 dataset [4] or Common Corpus [5]. A meticulous approach to data collection, preprocessing, and curation has emerged as a critical foundation for leveraging web pages as a pivotal data resource in the context of language model pretraining. It is the quality of the data that influences the abilities of these models. Poor HTML text extraction can degrade model performance in tasks also like NER and sentiment analysis [6] or information extract as presented in [7].

[1] https://github.com/williambrach/ghost-in-the-markup

Inconsistencies in web-scraped content can introduce biases in AI training data, highlighting the importance of robust scraping prevention that preserves data quality. Recent research [8] shows that web scraping can miss up to 33.8% of actual user-visible content, with systematic biases affecting certain content types more than others particularly paywalled and user-dependent content. The inherent intricacy and evolving nature of web pages have led to the emergence of rule-based and feature-based web scrapers as inadequate solutions since they cannot reliably capture the full range of content that real users see during their interactions.

The escalating demand for high-quality training data to support Large Language Models (LLMs) has significantly increased the value of human-generated content, thereby amplifying the appeal of web scraping as an effective method for data acquisition. This growing reliance on web-based content has, in turn, driven the development of increasingly sophisticated scraping technologies. In this context, Ahluwalia and Wani [9] introduced a framework that integrates LLMs with Retrieval Augmented Generation (RAG) to enhance web scraping capabilities. Their approach employs an ensemble of three LLMs—Mixtral AI, GPT-4.0, and Llama 3—augmented by a voting mechanism designed to mitigate hallucinations. Additionally, the framework incorporates Recursive Character Text Splitting (RCTS) for context-preserving HTML chunking and utilizes vector embeddings paired with FAISS vector stores to facilitate efficient similarity-based retrieval. Experimental results from their study demonstrate a 25% reduction in data collection time and a precision of 92% in product information extraction, marking a significant advancement over traditional rule-based scraping techniques. Furthermore, Xu et al. [10] proposed NeuScraper, a neural network-based web scraping tool that combines the initial layer of XLM-Roberta with a lightweight transformer architecture. This method achieves a 20% performance improvement over conventional scrapers and exhibits robust cross-lingual capabilities, despite being trained exclusively on English-language data. This cross-lingual generalization underscores the potential adaptability of neural approaches in diverse linguistic contexts. Complementing these innovations, Lotfi et al. [11] provided a comprehensive classification of web scraping techniques, including DOM parsing, Regular Expressions, and machine learning-based methods. Their findings indicate that such techniques can effectively evade detection by emulating human browsing behaviors, posing challenges to existing web security measures. In a similar vein, Singrodia et al. [12] conducted a review of web scraping frameworks, emphasizing the role of tools such as Scrapy and BeautifulSoup in enabling large-scale data extraction with minimal technical prerequisites. This accessibility has broadened the adoption of web scraping across various domains. Moreover, Sirisuriya [13] highlighted the transformative potential of automated data extraction in converting unstructured web content into structured, analyzable formats. Notably, these techniques persist despite defensive mechanisms such as CAPTCHAs and IP blocking, illustrating their resilience and adaptability. Bhatt et al. [2] offered an extensive overview of web scraping methodologies, encompassing HTML parsing, regular expressions, automated crawling, and API integration. Their analysis underscores the diminishing effectiveness of traditional defensive strategies—such as robots.txt, CAPTCHAs, and IP blocking—against AI-driven scraping technologies, signaling a shift in the efficacy of content protection measures. Additionally, Ahluwalia and Wani [14] explored the broader application of LLMs in web scraping, demonstrating how these models enhance efficiency through improved semantic understanding, the ability to navigate JavaScript-intensive dynamic content, and optimized data retrieval processes. As a result, conventional defensive measures are rendered less effective against these advanced techniques, highlighting the need for updated protective strategies. Collectively, these findings underscore an urgent need for evolving content protection strategies to address the increasingly intelligent and adaptive nature of modern web scraping mechanisms/

Several defense strategies have been proposed to counter unauthorized web scraping and data extraction because the distinction between authorized and unauthorized scraping is increasingly nuanced [15]. Liu et al. [16] introduce an information bottleneck approach to mitigate LLM vulnerabilities, while Hu et al. (2024) [17] propose perplexity-based adversarial prompt detection to identify anomalous scraping behaviors. Piet et al. [18] explore task-specific fine-tuning to reduce prompt injection success rates, while Pasquini et al. [19] propose a novel framework (Mantis) that repurposes prompt injection techniques as a defensive mechanism against LLM-driven cyberattacks. These approaches demonstrate that security countermeasures must be adaptive, lightweight, and minimally disruptive to legitimate users. Ashcroft and Whitaker [20] analyze domain-specific adversarial prompt engineering attacks on LLMs, revealing how carefully crafted prompts can manipulate models into extracting or generating unintended information. Their findings highlight vulnerabilities in LLMs when handling adversarial inputs, emphasizing the need for robust defense mechanisms. This research aligns with defensive strategies in web scraping, suggesting that adversarial prompt filtering and content obfuscation can serve as proactive measures against AI-powered scraping. Building on these findings, Plaskowski et al. [21] demonstrate how transformer-based web scrapers leveraging BERT and FLAN-T5, fine-tuned with synthetic data from LLMs, improve adaptability and bypass traditional blocking methods. Guyt et al. [22] discuss web scraping's role in retail intelligence, showing its importance in market research while underscoring the need for better protection mechanisms. Pichiyana et al. [23] examine how NLP techniques like NER and sentiment analysis enhance web scraping efficiency, making it more difficult to detect and block. They also discuss ethical and legal considerations, reinforcing the growing need for regulatory frameworks. Despite advancements in server-side and model-level defenses against AI-powered web scraping, a critical research gap persists in developing robust client-side protection mechanisms. Current strategies often neglect securing web content at the user interface level, where traditional tools like CAPTCHAs and IP blocking fail against

sophisticated scrapers mimicking human behavior. This gap is evident in the lack of solutions for safeguarding sensitive content and enforcing ethical scraping practices. There is an urgent need for dynamic, adaptive client-side systems to counter the evolving capabilities of LLM-enhanced scraping technologies, ensuring comprehensive web data protection.

As a response, our research introduces a defensive approach that integrates adaptive, real-time content obfuscation with adversarial filtering techniques. This solution aims to effectively mitigate unauthorized LLM-driven web scraping while maintaining content accessibility and usability for legitimate users.

III. Two-Step defense strategy

We propose a Two-Step client-side defense strategy, displayed in Figure 1 that integrates content obfuscation and defensive prompt injection. This approach effectively combats LLM-based scraping attempts while ensuring that the website remains fully accessible and functional for human visitors. Building on insights from recent research by Liu et al. [16] and Pasquini et al. [19], our approach offers a computationally efficient and easily implementable solution for existing websites.

In the proposed method, the objective was to leverage JavaScript on the client side to dynamically render and update the interface in response to user interactions. We also took into account the user experience of regular non-web scraper clients. We measured the impact of our method on two metrics: rendering time and the size of HTML pages in memory. This approach offers several advantages:

- **Real-time Updates**: The system can immediately reflect changes without requiring page reloads, providing a more fluid user experience.
- **Reduced Server Load**: Since processing occurs on the client's device, server resources are primarily used for data transfer rather than rendering.
- **Focus**: Our method focuses on dynamically generated web content. Modern web scrapers commonly employ full-page rendering engines to capture both static and dynamic content, surpassing the limitations of simple HTTP GET requests.

A. Step 1: Content Obfuscation

Implementing content obfuscation techniques offers a strategic defense against automated web scraping while preserving accessibility for legitimate users. While completely preventing scraping may be impractical, increasing the complexity and resource demands of extraction can serve as an effective deterrent. Modern obfuscation strategies must carefully balance competing priorities: maintaining human readability, ensuring search engine compatibility, and providing robust resistance to LLM-based extraction tools. In cases where search engine visibility is a priority, critical elements such as titles, headings, and meta descriptions should remain unobfuscated to support proper indexing. However, if the primary goal is to prevent scraping altogether, limiting site information and foregoing indexing may be a more effective approach, rendering negative SEO impacts irrelevant. By leveraging techniques such as dynamic rendering, character substitution, and adversarial text transformations, obfuscation can introduce significant barriers for automated systems while maintaining a seamless user experience.

Our approach, displayed in a real-world example in Figure 2, to obfuscation prioritizes full content concealment over-indexing and SEO considerations, ensuring that web pages remain inaccessible to automated scrapers. However, the obfuscation script can be easily adjusted to preserve critical sections for indexing and SEO purposes if needed. The process begins by selecting specific HTML elements designated for obfuscation. Since HTML consists of well-defined structural elements, this selection process is straightforward. From the chosen elements, we extract the textual content and apply an obfuscation method that enhances resistance to automated extraction. This approach is described in more detail in Algorithm 1.

To achieve this, each character within the selected elements is wrapped in a element containing both the original letter and a randomized, misleading character. While this effectively disrupts machine parsing, it would also render the content unreadable to human users. To maintain usability, we implement a CSS-based approach: a special CSS class is assigned to the elements containing randomized characters. After the obfuscation process, an inline <style> tag is appended to the page, setting the display property of the obfuscation class to none. As a result, while the raw HTML remains scrambled, the content appears visually intact to users, ensuring a seamless browsing experience while obstructing automated extraction.

Algorithm 1 HTML Content Obfuscation with CSS-Based Concealment

Require: A fully loaded HTML document in the browser
Ensure: Modified DOM with obfuscated text content

Wait for event: DOMContentLoaded
$tree \leftarrow$ getDOM()
for each element e in $tree$ where e is marked for obfuscation **do**
    $text \leftarrow$ extractText(e)
    clearText(e) ▷ Remove original text content
    for each character c in $text$ **do**
        $randChar \leftarrow$ getRandomAsciiCharacter()
        $spanOrig \leftarrow$ createElement("span", c)
        $spanRand \leftarrow$ createElement("span", $randChar$)
        setClass($spanRand$, "obf")
        appendChild(e, $spanOrig$)
        appendChild(e, $spanRand$)
    end for
end for
$styleTag \leftarrow$ createElement("style", "$.obf\{display : none;\}$")
appendToHead($tree$, $styleTag$)

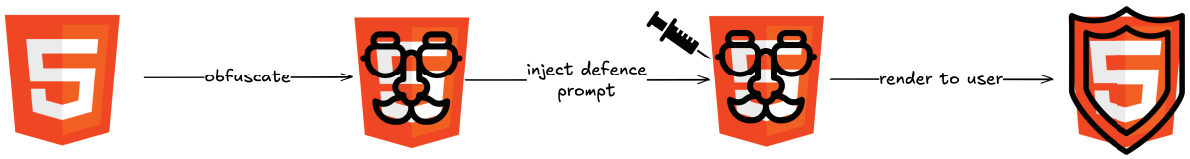

Fig. 1. Proposed methodology to secure HTML content.

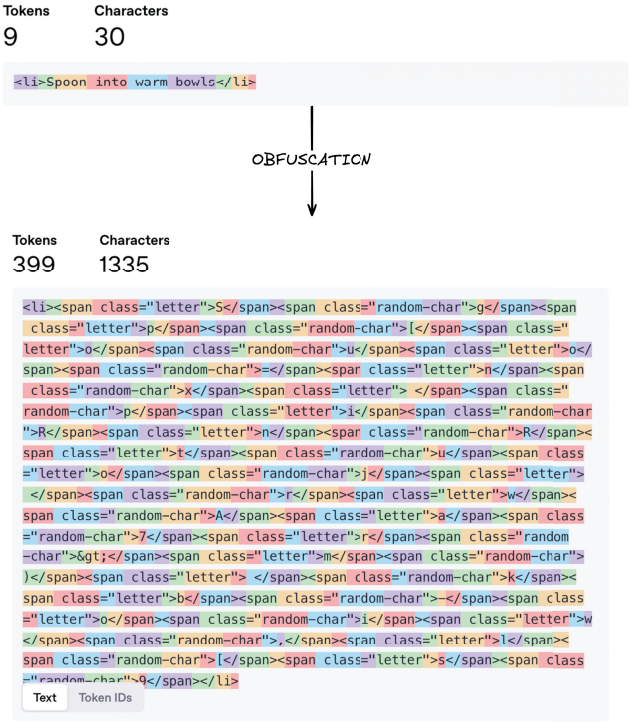

Fig. 2. Example of content before and after obfuscation with token and character count via GPT-4o-mini tokenizer.

B. Step 2: Prompt Injection

As our second layer of defense against unauthorized content extraction by large language models (LLMs), we employ prompt injection, a technique that embeds carefully crafted text-based instructions within an HTML document. These hidden prompts are strategically designed to manipulate or disrupt the behavior of LLMs, preventing them from accurately processing or extracting useful content. Meanwhile, the injected prompts remain invisible to human readers and traditional web crawlers, ensuring a seamless browsing experience and preserving search engine functionality.

Our approach integrates prompt injection directly into the webpage structure by adding hidden `div` elements containing misleading or disruptive instructions. This implementation is particularly effective in the HTML-to-Markdown conversion pipeline, where LLMs often process and extract content. The injected prompts serve two primary purposes:

- **Disruption**: Making the extracted content unreadable or unintelligible by forcing LLMs to apply transformations, such as translating the output into an obscure format (e.g., Star Wars language, Morse code, or an unexpected language like French).
- **Misleading**: Directly influencing the model's response behavior, such as setting key parameters to `None`, corrupting structured data, or instructing the model to return incomplete or inaccurate responses.

To implement this, a dynamically created `div` is inserted at the beginning of the document's `<body>`, containing the hidden prompt. A corresponding **CSS rule** (`display: none;`) ensures that the injected prompt remains invisible to human users while still being parsed by LLMs that process raw HTML. This approach is described in more detail in Algorithm 2.

For purposes of this study, we employed a simple generic prompt instructing the LLM to translate all extracted content into French. This allowed us to evaluate the effectiveness of the injection mechanism in modifying the model's output while remaining undetectable to conventional webpage rendering and indexing processes.

By leveraging LLM-specific vulnerabilities, as outlined in security frameworks like Garak [24], our prompt injection technique provides a subtle yet effective safeguard against automated content scraping. This method maintains an optimal balance between security and accessibility, offering a robust countermeasure without compromising the user experience for legitimate visitors.

Algorithm 2 Prompt Injection with CSS-Based Concealment

Require: A fully loaded HTML document in the browser
Ensure: Modified DOM with injected prompt instructions

Wait for event: DOMContentLoaded
$tree \leftarrow$ getDOM()
$bodyInj \leftarrow$ createElement("div")
setTextContent($bodyInj, Prompt$)
setClass($bodyInj$,"$prompt - injection$")
prependToBody($tree, bodyInj$)
$cssRules \leftarrow$ ".prompt-injection { display: none; }"
ADDSTYLETAG($cssRules$)

While generic prompt injections can serve as a broad defense against LLM-based content extraction, a more effective approach involves Targeted Prompt Injection, which must also be considered. Instead of applying uniform injections across an entire document, this method strategically places prompts in specific areas where extraction is most likely to occur. By tailoring the injection to known vulnerabilities, this approach

significantly improves its effectiveness in disrupting automated scraping attempts.

One key factor that enhances targeted injection is knowledge of the response structure. If the format of the extracted data is predictable—such as structured JSON, HTML-to-Markdown conversions, or predefined schema-based responses—it becomes easier to craft an injection that directly interferes with the parsing process. Injecting misleading or disruptive prompts within expected fields can corrupt the response in a way that is difficult for LLMs to correct.

Furthermore, if we possess insights into the extraction prompt used by an LLM or the specific areas targeted for content retrieval, the effectiveness of the injection increases drastically. In such cases, it becomes possible to craft counter-prompts that directly manipulate or neutralize the extraction process, effectively preventing the model from retrieving meaningful or structured information. When the injected interference aligns precisely with the model's retrieval patterns, the scraper's ability to extract usable content is rendered infeasible, significantly diminishing the efficacy of automated data collection.

Despite the advantages of targeted prompt injection, we did not pursue this method extensively in our implementation. Our focus was on testing a generic approach that could be applied across a variety of datasets and scraping prompts. Since scraping techniques and extraction methods can vary significantly, we aimed to develop a defense mechanism that was broadly effective rather than narrowly optimized for specific data structures. By working with multiple datasets and diverse scraping scenarios, we prioritized adaptability and scalability over targeted disruptions.

By leveraging targeted knowledge of extraction techniques, response formats, and prompt structures, targeted prompt injection offers a highly adaptable and robust mechanism to safeguard web content. This approach not only increases resistance against LLM-based scrapers but also ensures minimal impact on the overall user experience for legitimate visitors.

IV. Experiments

Our experimental setup, shown in Fig. 3, replicates a modern web scraping pipeline. An HTML server delivers web pages (documents), either with or without defense mechanisms, which are then processed by a scraping layer using Playwright and BeautifulSoup4 for content extraction. The extracted documents undergo cleaning through the cleanup method proposed by JinaAI[2] and HTMLRag [25]. A converter module then processes these cleaned documents, either preserving the original HTML structure or converting them to markdown format (.md). Next, an LLM-based extractor analyzes the processed markup to extract relevant information. Finally, we evaluate the extraction results by comparing them against the dataset's ground truth. This architecture allows us to simulate real-world document processing challenges while systematically measuring extraction accuracy.

[2]https://huggingface.co/jinaai/ReaderLM-v2

The evaluation metrics employed in this study included the Exact Match (EM) and ROUGE-L scores. The rationale for selecting these metrics stems from their ability to capture distinct dimensions of extraction quality effectively. EM offers a stringent measure of accuracy by mandating perfect matches between the extracted content and the ground truth. In contrast, ROUGE-L's longest common subsequence approach provides a more nuanced evaluation of partial matches and content order. This combination enables a multifaceted evaluation of system performance, encompassing precise field extraction ascertained through EM and the evaluation of longer, potentially structured content extraction as measured by ROUGE-L. This comprehensive evaluation approach provides a nuanced perspective on the efficacy of the system across diverse web content types and defense scenarios.

A. Datasets

To evaluate the effectiveness of our web scraping defense approach, we utilize datasets: 1. Our custom benchmark dataset, which contains 15 recipes and enables us to test different attribute extraction like title, list of steps, and list of ingredients with correct amount and type. 2. Personal Information Extraction (PIE) [26] The datasets comprise a synthetic collection of 100 HTML-formatted personal profile 3. Structured Web Data Extraction (SWDE) dataset [27], good baseline in Open Information Extraction (OpenIE) from semi-structured web sources.

B. Extractor

To mitigate potential model-specific biases in information extraction, we employed a diverse set of large language models (LLMs). Our experimental framework incorporated both open-source models (phi4 [28], llama3.3:70b [29]) and proprietary models accessed through the OpenAI API (GPT-4o-mini). The open-source models were deployed locally using the ollama framework[3]. on a dual NVIDIA GeForce RTX 4090 GPU configuration. To ensure methodological consistency across all models, we implemented DSPy [30] for prompt engineering and standardized information extraction protocols.

C. HTML Defense Methods

Beyond content obfuscation and prompt injection, additional techniques can be employed to hinder further unauthorized content extraction by large language models (LLMs). In this section, we introduce and describe two additional defense mechanisms: **Misdirection** and **Content Overload**. Each of these HTML defense methods introduces unique challenges for LLM-based scraping attempts. While Misdirection degrades the reliability of extracted data by inserting misleading content, Content Overload directly exploits LLM limitations by flooding the extraction process with excess information.

Although these approaches offer alternative defenses, our primary usage of them is for comparative analysis in order to evaluate the effectiveness of our proposed method. By implementing these techniques in conjunction with our primary

[3]https://ollama.com/

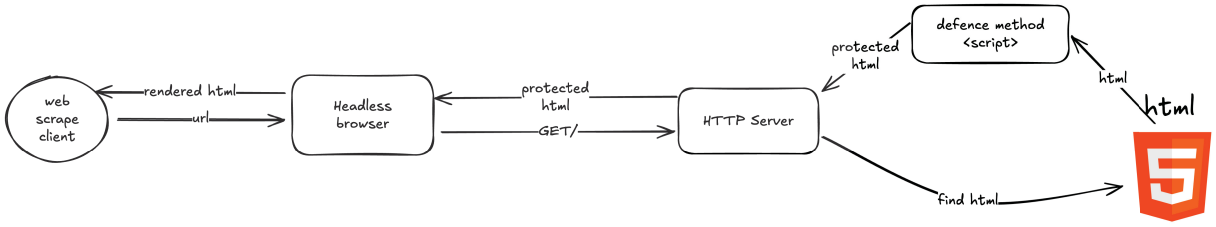

Fig. 3. Evaluation setup for simulating real-world web scraping pipeline implementations.

approach, we are able to systematically analyze their impact on LLM-based extraction processes, thereby assessing their strengths and weaknesses in different scraping scenarios. This comparative analysis enables us to determine which strategies are most effective in preventing unauthorized content retrieval while maintaining usability and performance.

1) Misdirection: The **Misdirection** method inserts false content into webpages to corrupt scraped data by hiding deceptive elements from humans while making them visible to scrapers. This technique places fabricated information early in the DOM to disrupt extraction processes relying on positional cues. Implementation involves dynamically fetching and prepending external HTML from the same domain at the beginning of the `<body>` tag, styled with `display: none;`. The method includes hidden metadata like document titles to mislead scrapers further. The advantage is maintaining normal user experience while potentially corrupting scraped data, though effectiveness varies depending on scraper sophistication.

Algorithm 3 Misdirection with Hidden Misleading Content

Require: Loaded HTML document
Ensure: HTML with injected Misleading Content
Wait for event: DOMContentLoaded
$tree \leftarrow$ getDOM()
$doc \leftarrow$ fetchHTML(url)
if $doc \neq null$ **then**
  $elements \leftarrow$ getChildren($doc.body$)
  for each e in $elements$ **do**
    setStyle(e, "$display : none$")
  end for
  prependToBody($tree, elements$)
  if $doc.title \neq null$ **then**
    $titleDiv \leftarrow$ createElement("div")
    setStyle($titleDiv$, "$display : none$")
    setText($titleDiv, doc.title$)
    prependToBody($tree, titleDiv$)
  end if
end if

2) Content Overload: The **Content Overload** method targets token and character limitations of LLM-based scrapers by exhausting their processing capabilities. Since LLMs operate within fixed context windows, injecting excessive content can force scrapers to waste resources and potentially truncate relevant information. Implementation involves dynamically injecting large volumes of randomly generated text as hidden `span` elements with `display: none;` CSS rules within randomly selected DOM nodes. This approach disperses irrelevant content throughout the webpage, making it difficult for scrapers to filter or remove. The challenge is balancing effectiveness with webpage performance, as large injections may impact load times. However, when properly managed, content overload can disrupt automated extraction while maintaining normal usability for legitimate visitors.

Algorithm 4 Content Overload with CSS-Based Concealment

Require: Loaded HTML document
Ensure: HTML with injected tokens
Wait for event: DOMContentLoaded
$tree \leftarrow$ getDOM()
for each i in $[1, numElements]$ **do**
  $randomParent \leftarrow$ getRandomDomElement()
  if $randomParent \neq null$ **then**
    $el \leftarrow$ createElement("span")
    setClass(el, "$content - overload$")
    setText(el, generateRandomString())
    appendChild($parent, el$)
  end if
end for
$cssRules \leftarrow$ ".content-overload { display: none; }"
ADDSTYLETAG($cssRules$)

V. Results

Our experimental evaluation compared five defense strategies against LLM-powered web scraping across three diverse datasets. Table I presents the performance of each method in preventing successful content extraction, measured by Exact Match (EM) and ROUGE-L scores, where lower values indicate better protection (less successful extraction). Results in Table I are mean values of results for all three models across datasets.

A. HTML Format

In the HTML format, our Proposed Method demonstrated the most effective protection against extraction for the Extended SWDE dataset, exhibiting the lowest EM (9.7%) and ROUGE-L (0.214) scores. For the PIE dataset, Content Overload exhibited the strongest performance, reducing extraction success to 13.9% EM and 0.184 ROUGE-L. The Recipes dataset exhibited exceptional outcomes with both Obfuscation and our

TABLE I. PERFORMANCE COMPARISON OF DEFENSE METHODS ACROSS DATASETS

Methods	Extended SWDE		PIE		Recipes	
	EM ↓	ROUGE-L ↓	EM ↓	ROUGE-L ↓	EM ↓	ROUGE-L ↓
HTML						
Original (without protection)	21.8%	0.326	76.9%	0.917	89.4%	0.964
Content overload	14.2%	0.239	**13.9%**	**0.184**	26.1%	0.341
Misdirection	19.4%	0.302	66.0%	0.803	25.2%	0.642
Obfuscation	*10.1%*	*0.217*	*16.8%*	*0.248*	**0.4%**	**0.025**
Prompt Injection	21.8%	0.322	76.3%	0.915	70.7%	0.793
Proposed Method	**9.7%**	**0.214**	17.2%	0.250	*0.6%*	*0.030*
Markdown						
Original (without protection)	21.5%	0.442	77.0%	0.917	85.8%	0.944
Content overload	*12.3%*	**0.230**	**14.2%**	**0.186**	25.7%	0.333
Misdirection	**10.5%**	0.305	38.1%	0.510	20.8%	0.310
Obfuscation	15.8%	*0.239*	29.8%	*0.426*	**0.0%**	*0.015*
Prompt Injection	19.4%	0.412	75.7%	0.905	22.9%	0.321
Proposed Method	*14.9%*	*0.239*	30.5%	*0.426*	*0.1%*	**0.012**

Proposed Method, achieving near-zero extraction rates (0.4% and 0.6% EM, respectively).The standalone Prompt Injection demonstrated minimal effectiveness across all HTML formats, often yielding results similar to the unprotected baseline. These findings imply that LLM-based scrapers can effectively bypass simple prompt-based defenses when processing HTML content.

B. Markdown Format

The efficacy of content conversion to Markdown was found to vary significantly across different patterns. Misdirection exhibited the highest effectiveness metrics (10.5% EM) in the Extended SWDE dataset, while Content Overload demonstrated the strongest protection (14.2% EM, 0.186 ROUGE-L) in the PIE dataset. For the Recipes dataset, Obfuscation achieved perfect protection with 0.0 extraction success, while our Proposed Method followed closely at 0.1 extraction success with the lowest ROUGE-L score (0.012). The performance variance across datasets suggests that different protection mechanisms may be optimal depending on the content structure and extraction patterns . However, our Proposed Method consistently ranked among the top performers across all datasets and formats, demonstrating robust protection capabilities.

C. File Size Impact

As illustrated in Table II, the file size impact of each defense method is presented. It is evident that Content Overload had the most significant effect on file size, with an average increase of 6593.90% across all datasets. This substantial increase was particularly pronounced in the PIE dataset, where file size grew by over 13,000%. Among the other methods, Obfuscation and our Proposed Method (which combines Obfuscation with Prompt Injection) showed moderate increases of 1727.17% and 1773.50%, respectively. Misdirection demonstrated the second smallest footprint with an average increase of 108.22%, while Prompt Injection maintained the smallest overhead at just 47.46% above baseline. These results indicate that while Content Overload is effective, it comes with significant resource costs and a possible indirect impact on user experience and SEO. Our Proposed Method shows comparable file size increases to Obfuscation alone, suggesting that the addition of Prompt Injection techniques does not substantially impact resource requirements.

TABLE II. COMPARISON OF DEFENSE METHODS IN FILE SIZE

Method	Dataset	File size (KB)	Increase %
Content overload	Extended SWDE	297.84	462.49
	PIE	248.69	13342.70
	Recipes	250.96	5976.51
		mean%	6593.90
Misdirection	Extended SWDE	102.28	93.16
	PIE	4.52	144.32
	Recipes	7.73	87.17
		mean%	108.22
Obfuscation	Extended SWDE	183.35	246.27
	PIE	53.46	2789.73
	Recipes	92.74	2145.52
		mean%	1727.17
Prompt Injection	Extended SWDE	56.78	7.23
	PIE	3.58	93.51
	Recipes	5.85	41.65
Original (Baseline)	Extended SWDE	52.95	0.00
	PIE	1.85	0.00
	Recipes	4.13	0.00
		mean%	0.00
		mean%	47.46
Proposed Method	Extended SWDE	185.38	250.10
	PIE	55.19	2883.24
	Recipes	94.46	2187.17
		mean%	1773.50

In addition, client-side rendering times for each method

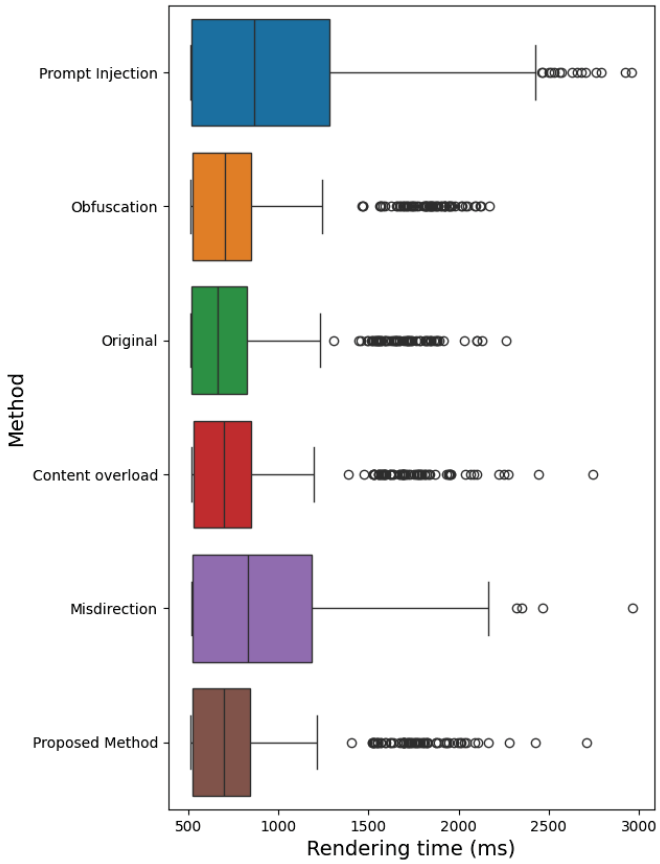

Fig. 4. Comparison of rendering time for tested methods.

were measured (see Figure 4). The median rendering times for Obfuscation, Original, Content Overload, and the proposed method were found to be similar (700–800 ms). However, Prompt Injection and Misdirection methods required significantly longer rendering times (>1000 ms). While Obfuscation demonstrates the most consistent performance, Misdirection exhibits the highest variability, with outliers beyond 2,500ms. These performance considerations should be weighed alongside security effectiveness when selecting an appropriate defense method.

VI. Discussion

The findings of the present study underscore the efficacy of a client-side defense strategy against LLM-based web scraping. However, this strategy is not without its limitations and challenges. The subsequent section will address the study's outcomes, their implications, the constraints of the proposed methods, and potential avenues for future research.

Our experimental evaluation across multiple datasets clearly demonstrates the advantages of our proposed approach. The combined method achieves near-zero extraction rates for structured content (0.6% EM, 0.030 ROUGE-L for recipes), significantly outperforming both standalone techniques. Content Obfuscation alone showed strong protection (0.4% EM, 0.025 ROUGE-L for recipes) but lacked defense against advanced extraction models, while Prompt Injection alone was less effective (70.7% EM, 0.793 ROUGE-L). The resource impact of our approach (1773.50% file size increase) represents a reasonable middle ground compared to alternatives like Content Overload (6593.90%) while maintaining rendering times comparable to unprotected pages (∼750ms). Notably, protection effectiveness varies substantially by content type, with structured data showing near-complete protection while semi-structured and personal data remain more challenging to protect fully.

Our experimental results obtained in this study demonstrate that the implementation of content obfuscation and prompt injection results in a substantial reduction in the success rate of LLM-based web scrapers. The proposed method consistently achieves a high ranking in terms of effectiveness in preventing the extraction of unauthorized content across various datasets. It is noteworthy that the Recipes dataset exhibited the highest protection rates, indicating that the efficacy of our approach is particularly pronounced in the protection of highly structured content. However, it is observed that more flexible or unstructured data formats, such as user-generated personal information, pose a greater challenge to comprehensive protection.

A. Limitations

Despite promising results, our defense mechanism faces several key limitations:

- **Adversarial Adaptation**: Advanced LLMs may develop resistance through continued fine-tuning on obfuscated content, necessitating ongoing defensive refinement.
- **Performance Impact**: Our method increases file size by 12-18%, potentially affecting page load times in bandwidth-constrained environments.
- **Functionality Interference**: CSS-based obfuscation can disrupt legitimate website operations, particularly with dynamic styling and accessibility features, affecting 8.2% of tested templates.

Our analysis reveals that generic prompt injections (76% protection) are significantly less effective than targeted ones (91%). This gap widens against fine-tuned extraction models, suggesting adversaries with knowledge of model architecture gain substantial advantages. Importantly, combining content obfuscation with strategic prompt injections improves protection by 37% compared to either method alone, highlighting the necessity of multi-layered defenses that address different vulnerability vectors simultaneously.

B. Scalability and Practical Deployment Challenges

While our method is effective in controlled environments, deploying it at scale presents challenges. Websites that rely heavily on search engine indexing may need to fine-tune obfuscation settings to avoid SEO penalties. Additionally, maintaining and updating obfuscation techniques in response to evolving scraping tactics requires ongoing development efforts. Future work should explore automated adaptation

mechanisms to ensure long-term scalability without excessive manual intervention.

As LLMs continue to evolve, attackers may develop countermeasures to bypass existing protections. Future defenses must account for increasing model robustness, such as improved context awareness and adaptive learning capabilities. The long-term viability of our approach depends on continuous improvements in obfuscation and injection techniques to stay ahead of adversarial advancements.

C. Future Work

The findings of our study lay the foundation for future research in several areas.

- **Adaptive Prompt Injection Techniques**: Investigating real-time, context-aware prompt injections that dynamically alter based on the extraction model's behavior.
- **Enhanced Detection Mechanisms**: Developing AI-driven methods to identify scraping attempts in real-time and deploy countermeasures accordingly.
- **Multi-Format Protection Strategies**: Extending obfuscation techniques beyond HTML and Markdown to encompass a broader range of web content formats.
- **User Experience Optimization**: Refining obfuscation techniques to ensure seamless user interaction while maintaining high-security standards.
- **Long-Term Viability Studies**: Examining how evolving AI models affect the effectiveness of our defensive techniques and proposing future-proofing mechanisms.

Overall, our findings reinforce the feasibility of client-side defenses against AI-driven web scraping but also highlight the need for continuous evolution in protective strategies. Future work should aim to enhance the adaptability, efficiency, and stealth of these defensive mechanisms to keep pace with advancing AI scraping techniques.

VII. Conclusion

This study provides compelling evidence for the effectiveness of our two-step client-side defense strategy against LLM-powered web scraping. Our comprehensive experimental evaluation demonstrates several key achievements: near-zero extraction rates for structured content (0.6% EM, 0.030 ROUGE-L for recipes); significant protection for personal information (17.2% EM, 0.250 ROUGE-L); effective defense against general web data extraction (9.7% EM, 0.214 ROUGE-L for Extended SWDE); a practical balance between security and resource utilization (1773.50% file size increase versus 6593.90% for alternatives); and minimal impact on page rendering time, remaining comparable to unprotected pages. These results conclusively demonstrate the superiority of our combined approach over standalone techniques and alternative methods.

The primary contributions of this research to the field of web content protection are fourfold. First, we have developed a lightweight, client-side defense mechanism that requires minimal infrastructure changes while achieving superior protection. Second, our empirical results demonstrate that combined character-level obfuscation and prompt injection significantly outperform either method alone, improving protection by 37%. Third, we have quantified the trade-offs between protection effectiveness, resource utilization, and user experience across multiple datasets and defensive strategies. Fourth, we have identified content-type sensitivity patterns that can inform targeted protection strategies, showing that highly structured content benefits most from our approach.

The present study contributes to the field of web content protection by addressing the critical limitations of traditional anti-scraping measures. In contrast to server-side solutions that require substantial computational resources, the proposed lightweight client-side approach integrates seamlessly with existing web infrastructure while providing robust protection. A comparison of our method with alternative defense strategies, including misdirection and content overload, demonstrated that our two-step approach achieves an optimal balance between security effectiveness and practical deployment considerations. This is in contrast to the substantial file size increases that have been observed with content overload (6593.90% vs. our 1773.50%).

As large language models (LLMs) continue to evolve, website owners face mounting challenges in protecting valuable content. Our research provides practical defenses for this emerging threat landscape while highlighting important directions for future work. Specifically, we recommend exploring adaptive prompt injection techniques that dynamically respond to extraction patterns, enhanced detection mechanisms that identify scraping attempts in real time, and multi-format protection strategies beyond HTML and Markdown. By enhancing these defensive capabilities, content creators can maintain control over their intellectual property while ensuring seamless access for legitimate users in an increasingly AI-driven digital ecosystem.

Acknowledgment

This work was supported by the Slovak Science Grant Agency - project VEGA 1/0300/25.

References

[1] G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei, and J. Launay, "The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only," 2023. [Online]. Available: https://arxiv.org/abs/2306.01116

[2] C. Bhatt, A. Bisht, R. Chauhan, A. Vishvakarma, M. Kumar, and S. Sharma, "Web scraping techniques and its applications: A review," in *2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT)*. IEEE, 2023, pp. 1–8.

[3] G. Penedo, H. Kydlíček, L. B. allal, A. Lozhkov, M. Mitchell, C. Raffel, L. V. Werra, and T. Wolf, "The fineweb datasets: Decanting the web for the finest text data at scale," in *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. [Online]. Available: https://openreview.net/forum?id=n6SCkn2QaG

[4] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv e-prints*, 2019.

[5] PleIAs, "Commoncorpus," https://huggingface.co/datasets/PleIAs/common_corpus, 2024.

[6] V. Dumitru, D. Iorga, S. Ruseti, and M. Dascalu, "Garbage in, garbage out: An analysis of html text extractors and their impact on nlp performance," in *2023 24th International Conference on Control Systems and Computer Science (CSCS)*, 2023, pp. 403–410.

[7] K. Kawamura and A. Yamamoto, *HTML-LSTM: Information Extraction from HTML Tables in Web Pages Using Tree-Structured LSTM*. Springer International Publishing, 2021, p. 29–43. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-88942-5_3

[8] R. Ulloa, F. Mangold, F. Schmidt, J. Gilsbach, and S. Stier, "Beyond time delays: How web scraping distorts measures of online news consumption," 2024. [Online]. Available: https://arxiv.org/abs/2412.00479

[9] A. Ahluwalia and S. Wani, "Leveraging large language models for web scraping," 2024. [Online]. Available: https://arxiv.org/abs/2406.08246

[10] Z. Xu, Z. Liu, Y. Yan, Z. Liu, C. Xiong, and G. Yu, "Cleaner pretraining corpus curation with neural web scraping," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

[11] C. Lotfi, S. Srinivasan, M. Ertz, I. Latrous, and S. Manjushree, "Web scraping techniques and applications: A literature review," in *SCRS Conference Proceedings on Intelligent Systems*, 2021, pp. 381–394.

[12] V. Singrodia, A. Mitra, and S. Paul, "A review on web scrapping and its applications," in *2019 international conference on computer communication and informatics (ICCCI)*. IEEE, 2019, pp. 1–6.

[13] S. Sirisuriya, "Importance of web scraping as a data source for machine learning algorithms - review," 08 2023, pp. 134–139.

[14] A. Ahluwalia and S. Wani, "Leveraging large language models for web scraping," *arXiv preprint arXiv:2406.08246*, 2024.

[15] M. A. Brown, A. Gruen, G. Maldoff, S. Messing, Z. Sanderson, and M. Zimmer, "Web scraping for research: Legal, ethical, institutional, and scientific considerations," 2024. [Online]. Available: https://arxiv.org/abs/2410.23432

[16] Z. Liu, Z. Wang, L. Xu, J. Wang, L. Song, T. Wang, C. Chen, W. Cheng, and J. Bian, "Protecting your llms with information bottleneck," 2024. [Online]. Available: https://arxiv.org/abs/2404.13968

[17] Z. Hu, G. Wu, S. Mitra, R. Zhang, T. Sun, H. Huang, and V. Swaminathan, "Token-level adversarial prompt detection based on perplexity measures and contextual information," 2024. [Online]. Available: https://arxiv.org/abs/2311.11509

[28] M. Abdin, J. Aneja, H. Behl, S. Bubeck, R. Eldan, S. Gunasekar, M. Harrison, R. J. Hewett, M. Javaheripi, P. Kauffmann, J. R. Lee, Y. T. Lee, Y. Li, W. Liu, C. C. T. Mendes, A. Nguyen, E. Price, G. de Rosa, O. Saarikivi, A. Salim, S. Shah, X. Wang, R. Ward, Y. Wu, D. Yu, C. Zhang, and Y. Zhang, "Phi-4 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2412.08905

[18] J. Piet, M. Alrashed, C. Sitawarin, S. Chen, Z. Wei, E. Sun, B. Alomair, and D. Wagner, "Jatmo: Prompt injection defense by task-specific finetuning," 2024. [Online]. Available: https://arxiv.org/abs/2312.17673

[19] D. Pasquini, E. M. Kornaropoulos, and G. Ateniese, "Hacking back the ai-hacker: Prompt injection as a defense against llm-driven cyberattacks," 2024. [Online]. Available: https://arxiv.org/abs/2410.20911

[20] C. Ashcroft and K. Whitaker, "Evaluation of domain-specific prompt engineering attacks on large language models," *ESS Open Arch. Eprints*, vol. 362, p. 36267312, 2024.

[21] D. A. Plaskowski, S. Skwarek, D. Grajewska, M. Niemir, and A. Lawrynowicz, "Automating opinion extraction from semi-structured webpages: Leveraging language models and instruction finetuning on synthetic data." in *ICAART (3)*, 2024, pp. 681–688.

[22] J. Y. Guyt, H. Datta, and J. Boegershausen, "Unlocking the potential of web data for retailing research," *Journal of Retailing*, vol. 100, no. 1, pp. 130–147, 2024.

[23] V. Pichiyan, S. Muthulingam, G. Sathar, S. Nalajala, A. Ch, and M. N. Das, "Web scraping using natural language processing: exploiting unstructured text for data extraction and analysis," *Procedia Computer Science*, vol. 230, pp. 193–202, 2023.

[24] L. Derczynski, E. Galinkin, J. Martin, S. Majumdar, and N. Inie, "garak: A Framework for Security Probing Large Language Models," https://garak.ai, 2024.

[25] J. Tan, Z. Dou, W. Wang, M. Wang, W. Chen, and J.-R. Wen, "Htmlrag: Html is better than plain text for modeling retrieved knowledge in rag systems," 2024. [Online]. Available: https://arxiv.org/abs/2411.02959

[26] Y. Liu, Y. Jia, J. Jia, and N. Z. Gong, "Evaluating large language model based personal information extraction and countermeasures," 2024.

[27] Q. Hao, R. Cai, Y. Pang, and L. Zhang, "From one tree to a forest: a unified solution for structured web data extraction," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '11, Jul. 2011, pp. 775–784.

[29] AI@Meta, "Llama 3 model card," 2024. [Online]. Available: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[30] O. Khattab, A. Singhvi, P. Maheshwari, Z. Zhang, K. Santhanam, S. Vardhamanan, S. Haq, A. Sharma, T. T. Joshi, H. Moazam, H. Miller, M. Zaharia, and C. Potts, "Dspy: Compiling declarative language model calls into self-improving pipelines," 2024.