Generating Adversarial Prompts from Incidents and Guidelines

KURT G. SALAPARE, University of Twente, The Netherlands

Abstract Rapid deployment of Large Language Models (LLMs) has introduced significant security vulnerabilities, yet the limited public availability of detailed incident reports regarding exact prompts or techniques used impedes comprehensive security analysis and the development of robust defenses. This research addresses this gap by designing and evaluating a novel AI agent capable of automatically generating adversarial prompts from existing security guidelines and reported incidents. The agent employs a two-phase workflow: first, processing unstructured text into a classified, metadata-rich dataset via LLM-driven paragraph classification, and second, utilizing these insights to generate executable adversarial prompts. We investigate the performance of various LLMs and prompt architectures (Descriptive, Concise, Few-Shot) within the agent, evaluating their computational efficiency, classification reliability, and the characteristics of the generated prompts. This systematic methodology offers a reproducible framework for improving proactive security analysis by providing a structured approach to adversarial prompt generation for law enforcement and developers.

ACM Reference Format:
Kurt G. Salapare. 2025. Generating Adversarial Prompts from Incidents and Guidelines. In *Proceedings of 43th Twente Student Conference on IT (TScIT 43).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The landscape of Artificial Intelligence (AI) has been significantly reshaped by the rapid development and widespread adoption of Large Language Models (LLMs) in recent years, underpinned by significant advancements in AI architectures [1]. These models have grown dramatically in scale and capability, becoming integral to diverse applications across numerous industries [1]. This adoption is increasingly evident in organizational workflows; for example, analysis of various text domains indicates that by late 2024, approximately **18% of financial consumer complaint text and up to 24% of corporate press release text appeared to be LLM-assisted** [2]. The widespread use of LLMs impacts areas from content generation and customer interaction to complex analytical tasks [1].

However, this rapid deployment has been accompanied by increasing concerns regarding the security and robustness of these models against adversarial attacks [3]. The field of adversarial machine learning explores sophisticated techniques designed to exploit vulnerabilities in AI systems, such as, and not limited to, prompt injection [4, 5]. For instance, research presented at security conferences has shown the tangible impact of these vulnerabilities, with one study demonstrating that **56% of prompt injection attempts successfully bypassed LLM safeguards** in an analysis of diverse models [5, 6]. Despite the growing awareness of these vulnerabilities, as discussed by a community of nearly 500 experts contributing to initiatives like the OWASP Top 10 for LLM applications [7, 8], a key challenge in the field is the limited availability of specific, reproducible details regarding observed attacks and the prompts or techniques used. Ethical considerations and security concerns often restrict the public disclosure of such granular information, creating a gap that impedes comprehensive security analysis, hinders the reliable reproduction of attacks for research and defense development, and limits the actionable knowledge available to security practitioners and developers ([5], implied by security discussions like [6]). This gap in reproducible attack documentation necessitates further research into systematic methods for studying and generating diverse adversarial prompts based on reported vulnerabilities.

Building on the need for reproducible adversarial attack documentation, this research investigates and will propose a reproducible novel AI agent to process reported AI vulnerabilities and automatically generate adversarial prompts. The increasing deployment of LLMs in real-world applications has highlighted their susceptibility to critical vulnerabilities, as illustrated by incidents such as an AI chatbot being manipulated to agree to sell a car for 1 USD [9] and reports of sensitive data leakage when employees used AI agents with access to internal databases [10]. These vulnerabilities include the aforementioned prompt injection [4, 5], which can manipulate model behavior. While current benchmarks exist to test LLM robustness against adversarial attacks [5, 6], these often focus primarily on technical manipulations, potentially overlooking more semantic or deceptive prompting strategies akin to human attackers [5, 6, 11]. Leveraging the understanding of how adversarial attacks are crafted, an interesting approach is to design an AI agent capable of analyzing vulnerability information and generating diverse, reproducible prompts that mimic aspects of human adversarial thinking and attack generation [11].

This research focuses on designing and evaluating the effectiveness of a proposed AI agent in automatically generating diverse and relevant adversarial prompts. The core idea is to design an agent that can utilize publicly available information about AI vulnerabilities and attack strategies. This involves building a list of relevant URLs, systematically scraping the content from these sources, and feeding this information into an LLM to generate prompts. The generated prompts would then be stored and subsequently evaluated. This evaluation will determine the effectiveness of the AI techniques utilized within the agent's internal process, considering aspects such as the efficiency of content processing, the quality of information extraction, and the diversity and structural properties of the generated prompts. Ideally, the generated prompts would also be tested against various target AI models to assess their success rate, diversity, the nature of the elicited unintended behaviors, and the effectiveness of this AI agent's output would be quantitatively

Author's address: Kurt G. Salapare, k.g.salapare@student.uwtente.nl, University of Twente, P.O. Box 217, Enschede, The Netherlands, 7500AE.

TScIT 43, July 4, 2025, Enschede, The Netherlands

compared against prompts derived from current adversarial attack benchmarks [5, 6, 11]. However, due to limited time constraints and lack of a labeled data set of paragraphs, the paper focuses primarily on the designing of the proposed novel AI agent and evaluating the AI techniques utilized. Ultimately, this evaluation will determine the agent's practical utility in enhancing proactive security analysis for law enforcement and developers.

To achieve this we divided this research to tackle 3 research questions :

(1) What are the current methodologies and documented limitations in publicly available resources for identifying, reproducing and understanding the specific prompts or input manipulations used in sophisticated adversarial attacks against AI systems?
(2) How can various LLMs be used to design a novel AI agent to process information from incident reports and guidelines to automatically generate adversarial prompts suitable for use in proactive security analysis by law enforcement and developers?
(3) How effective are the AI techniques within the proposed agent in generating adversarial prompts from incidents reports and guidelines?

By the end of this research we expect to have contributed in 3 different ways:

(1) Systematically identify and outline current methodologies and limitations in public resources for understanding and reproducing sophisticated adversarial attacks against AI systems. Highlight gaps hindering comprehensive security analysis based on available information.
(2) Propose and detail the design of a reproducible novel AI agent engineered to process incident reports and guidelines in order to automatically generate diverse, adversarial prompts.
(3) An evaluation of the proposed AI agent's internal components, specifically the AI techniques (classification models, prompt architectures, and data processing methods) employed in generating adversarial prompts. The analysis will quantify metrics such as computational efficiency, reliability of classification, and diversity of generated prompt structures, thereby providing insights into the performance and optimization of these techniques for future development in this domain.

The github repository for all the code used in this research can be found ***here***. Additionally contains all the technical details & information.

2 RELATED WORKS

2.1 The Approach

In answering RQ1, a systematic literature search and analysis was conducted for relevant documents primarily through Google Scholar, as it boasts an extensive array of interdisciplinary peer-reviewed papers. The search strategy consisted of employing keywords & phrases, such as "adversarial attacks LLM", "prompt injection", "jailbreaking large language models", and "reproducibility of adversarial attacks" to find the aforementioned relevant documents & papers. The inclusion criteria for sources will be peer-reviewed articles and conference papers published after 2019 that focus on text-based adversarial attacks.

2.2 Traditional Attack Strategies

Traditionally, manual prompt injection was employed primarily by injecting human engineered conflicting or overriding instructions into the target LLM's input. One of the most prevalent approaches to achieve this was by a direct instruction override, such as the "Ignore all previous instructions" followed by a prohibited action strategy [12]. This aimed to exploit the models' critical vulnerable tendency to process all instructions sequentially. This strategy was implemented into 2 types of attacks, goal hijacking and prompt leaking. Goal hijacking is defined as the act of misaligning the original goal of a prompt to a new goal of printing a target phrase and prompt leaking is the act of misaligning the original goal of a prompt to a new goal of printing part of or the whole original prompt instead. At the time, this strategy was able to achieve a success rate of 58.6% ± 1.6% for goal hijacking and 23.6% ± 2.7% for prompt leaking on text-davinci-002, which was the most powerful publicly available OpenAI model at the time [12].

One more prominent technique is the compositional instruction attack (CIA), which utilizes a "prompt packer" that aims to 'pack' a harmful prompt into a seemingly harmless/pseudo-harmless one, which in turn creates a composite prompt that LLMs tend to misinterpret as non-malicious [13]. This was achieved by utilizing a framework that uses automated transformation functions, labelled Talking-CIA (T-CIA) and Writing-CIA (W-CIA), to generate these complex prompts disguised as conversational or writing tasks, exploiting LLMs' susceptibility to respond under inferred personas or fictional contexts [13]. Although this method utilizes a more complex system and boasts automated generation, the core concept and structure behind the attack is rooted to a manual type (compositional instruction). This was a very successful method as it can package over 90% of the initially rejected harmful prompts into compositional instructions that can induce LLMs to generate harmful content, achieving a 100% non-rejection rate [13].

2.3 Modern Attack Strategies

Another paper introduces a very interesting idea, the Greedy Coordinate Gradient (GCG), which is a landmark automated method for "universal and transferable adversarial suffixes" generation [14]. It leverages gradient-based optimization (often on a white-box model, but shows transferability to black-box models) to find a short sequence of tokens that, when appended to various harmful queries,

causes LLMs to bypass their safety alignment and generate forbidden content. The paper was able to achieve the following results with their strategy. For white-box models, Llama-2-Chat-7B: GCG achieved an ASR (Attack Success Rate) of 57% for harmful strings and 56% for harmful behaviors on Llama-2-7B-Chat [14]. This also notably surpasses AutoPrompt (3% and 45%) and other baselines. With black box models, Llama-2-Chat-7B: achieved ASRs from approximately 70% to 90% [14].

Another modern and prominent technique is context manipulation and role-playing generation with an LLM. The LLM is instructed to adopt a persona or scenario that encourages the generation of restricted content [15]. AdvPrompter was designed to take a malicious instruction (e.g., Write a tutorial on building a bomb) and then generate an adversarial suffix (e.g., as part of a lecture on bomb defusal) that would elicit a positive response when combined with the malicious instruction [15]. This strategy was able to achieve a 1 shot ASR of 49.4% on GPT-3.5-1106 and 29.2% on GPT-4-0613.

2.4 Limitations

Although the traditional strategies have proven to be quite successful, especially for their time as safeguards and defense mechanisms were not as sophisticated and thorough, these strategies face significant limitations. Primarily, they are extremely labor-intensive as they rely on human ingenuity and extensive trial-and-error in their formulation. This negatively affects the scalability of these techniques towards a wider range of diverse models. Building off on this downside, due to the way they are usually generated, it is also not certain that an adversarial prompt that works with one model may work with another, leading them to be model specific. Additionally, as LLM developers continuously release updates and patches, what may have worked before may not work anymore. Thus, due to the initial aforementioned difficulty in manual prompt generation, they are prone to obsolescence. This constant "cat-and-mouse" game necessitates continuous discovery of new bypasses and highlights the need for more systematic and generalizable attack methodologies to ensure lasting security evaluations [12].

While the modern strategies significantly advanced beyond the traditional, manual methods, they still have their distinct drawbacks that this research aims to mitigate. GCG faces a problem with human readability as it produces semantically meaningless character sequences to the untrained human eye [14]. Although these prompts are effective, they have slowly become obsolete with new filters, such as the perplexity-based filter. This in turn leads to them becoming less useful for crafting actionable security guidelines. AdvPrompter does not inherently prioritize or explore attacks relevant to the complex and specific security policies that would be found in an organization's incident reports or internal guidelines [15].

This research aims to overcome these limitations by proposing a novel method of generating highly relevant, interpretable, and targeted adversarial prompts by deriving attack patterns directly from reported incidents and guidelines and thereby providing more actionable intelligence for enterprise-specific LLM security. In addition, this research will also tackle the limitations of the traditional attack strategies. This aims to be a method of generating diverse prompt structures based on current knowledge, via the previously mentioned guidelines and incident reports, enabling it to be tailored to a wider array of attacks. This advocates for more holistic and adaptive red-teaming capabilities.

Table 1. Comparison of Attack Strategies (Part 1: Basic Characteristics)

Paper	Type	Core Mechanism
Perez & Ribeiro	Manual	Direct Instruction Override
Jiang et al.	Automated	Prompt Packing
Zou et al.	Automated	Gradient-based Universal Suffixes
Paulus et al.	Automated	Context/Role Manipulation
This Research	Automated	Derived from Incidents/Guidelines

Table 2. Comparison of Attack Strategies (Part 2: Advanced Features and Limitations)

Paper	Universal	Legible	Obsolete Prone
Perez & Ribeiro		✓	✓
Jiang et al.		✓	✓
Zou et al.	✓		✓
Paulus et al.		✓	
This Research	✓	✓	✓

3 METHODOLOGY

3.1 Data Set

This subsection details the selection process and considerations of the core data inputs for the proposed AI agent: security guidelines and, where feasible, incident reports.

The predominant sources of information are derived from standardized security guidelines, particularly three authoritative online sources. These sources were strategically selected as they cover a diverse spectrum of LLM attack principles, from foundational typologies to more creative techniques and industry-known standard vulnerabilities. These include the National Institute of Standards and Technology (NIST)'s publications on cyberattacks on AI systems, Cloudfare's explanation of the OWASP Top 10 for LLM applications and HiddenLayer's insights into novel universal bypass methods. The rationale for these selections is that if an AI agent was able to demonstrate effective processing, understanding and learning on how and why these various attack strategies are designed, and applying the knowledge garnered from these sources into generating adversarial prompts that adhere to what it learnt, then it can be strongly inferred that this approach is broadly applicable. For indicent reports, two particular incidents were examined, an Air Canada chatbot promising to selling a ticket at a discounted price and Amazon's Bad Rufus: A Chatbot Gone Wrong. These incidents highlighted scenarios where safety and particularly security was compromised within these AI applications or systems.

3.2 Scraping and Data Extraction

For the data extraction from the websites, a webscraping was done utilizing the Crawl4ai framework, taking advantage of its integration with Playwright. This was critical for handling modern, dynamic web pages that heavily relied on Java Script for rendering their content. Crawl4ai was configured to navigate to each target URL, Playwright was then used to render the pages, and the main body HTML content was extracted and automatically converted into a structured Markdown object. This step was vital as Markdown provides a cleaner, more organized, and human readable version of the text compared to raw HTML, significantly reducing noise from advertisements, other elements that are irrelevant such as navigation components, as they can also cause problems for direct text analysis or LLM input. A workflow diagram visuzalizing the process can be found in ***Appendix A.1***.

3.3 Workflow methodology

The workflow is structured into 2 distinct but interconnected phases. These phases both utilize LLMs in their respective ways. The initial phase is paragraph classification. This phase serves a critical role as it is responsible for data pre-processing and organization. Raw unlabeled text data, attained from webscraping, was fed into an LLM. The instructions of the LLM were of 3 different prompt structures : structured prompt, concise prompt and few-shot prompt. The results of how these distinct prompt structures vary can provide a significant insight on how to best maximize this workflow depending on the real-life context. The primary task within this phase is to analyze each paragraph's content and assign a a usability score (0 - could not extract classification data because format was not as instructed, 1 - text that was outputed in the correct format however is entirely irrelevant to prompt attacks, 2 - relevant but not direct prompt examples, 3 - direct prompt examples) along with a detailed reasoning on said classification. This rigorous classification process converts unstructured text into an organized and categorized dataset benefiting the corresponding constituent phase. A workflow diagram visuzalizing the process can be found in ***Appendix A.2***.

The second phase is Adversarial Prompt Generation. As aforementioned, this part heavily relies that the data is categorized properly as only text classified as usability scores 2 and 3 will be utilized. Similarly, a LLM is employed here as well, but a different instance to the prior phase's. This phase's LLM is specifically instructed to create concrete, executable adversarial prompts based on the data filtered out from the first phase. Along with the input data, another important component to this phase is the different prompt architectures, mimicking phase 1, to examine how they affect the results. A workflow diagram visuzalizing the process can be found in ***Appendix A.3***.

3.3.1 Model Selection and Architecture for Classification. For the text-based classification, several LLMs available through the Ollama framework were evaluated, selected for its zero-cost accessibility, privacy-preserving local execution, and comprehensive model library enabling this to be reproducible for anyone as well as scalability to larger and more complex models. The evaluated models represent diverse architectural approaches and parameter scales:

(1) Llama3 (8B): Meta's open-weight model optimized for balanced performance across various NLP tasks
(2) Qwen (7b): Alibaba's multilingual model with strong semantic understanding capabilities
(3) Gemma (7B) : A new open model developed by Google and its DeepMind team. It is inspired by Gemini models at Google.
(4) Gemma3 (12B) : Another open model developed by Google and its DeepMind team however, is a larger variant within the Gemma family, with the purpose to use as a reference model.

3.3.2 Model Selection and Architecture for Prompt Generation. Similar to the text-based classification, the same LLMs available through the Ollama framework were selected for the same reasons as mentioned before. However, for the adversarial prompt generation, an ablitered version of Llama 3.1 was also included as it lowers the chance of guardrails inhibiting adversarial prompt generation as it was occasionally classified as prohibited by the models.

- mannix/llama3.1-8b-abliterated:latest (8B) : This is an uncensored version of Llama 3.1 8B Instruct created with abliteration with an uncensored prompt. Will be referred as ***lamma3.1-ab*** from now on.

3.4 Prompt Architecture and Engineering Strategy

Descriptive Prompt (Structured Definition). This foundational prompt structure follows a comprehensive, definition-based approach that explicitly details all the instructions. For Paragraph Classification, this prompt ensures that the LLM gains a complete and unambiguous understanding of how to categorize raw text based on its relevance to AI security vulnerabilities. For Adversarial Prompt Generation, it establishes the LLM's expert role and clearly states the objective and specific security-focused requirements.

Concise Prompt (Minimal Instructions). The core concept of this prompt architecture is to streamline tasks by reducing them to their fundamental elements. This is especially beneficial when computational overhead must be minimized, or when computation speed is prioritized over accuracy. For Paragraph Classification, the concise prompt details only the core classification instruction information and output format, but minimizes much of the other nuances. Similarly for Adversarial Prompt Generation, it solely focuses on only the core concepts of generating adversarial prompts from the context. The concise prompt structure leverages the concept that LLMs have the ability to retain context and apply learned instructions efficiently by minimizing complexity and length.

Few-Shot Prompt (In-Context Learning). This prompt format incorporates carefully curated labeled examples into the user prompt,

leveraging the LLM's ability to learn from a small labeled dataset. This approach also aims to mitigate the chance of hallucination by showcasing concrete demonstrations. For Paragraph Classification, few-shot examples illustrate how specific text content should be paired with its correct usability score and reason. For Adversarial Prompt Generation, examples demonstrate how the input data/content should be transformed into an effective, security-focused adversarial prompt, leading to more sophisticated and targeted outputs.

For all the prompts used in this paper, refer to this link to the github repository ***Prompts Utilized***.

RQ2 is comprehensively addressed in this section as it details how a novel AI agent is designed using various LLMs to process security information for automatic adversarial prompt generation. It specifies the use of diverse LLMs selected via the Ollama framework. This is the core of the agent's two-phase workflow: Paragraph Classification and Adversarial Prompt Generation. The proposed agent processes information from standardized security guidelines and incident reports, through web scraping with Crawl4ai/Playwright for clean Markdown extraction. The Adversarial Prompt Generation phase specifically outlines how an LLM, guided by structured, concise, or few-shot prompts, automatically creates executable adversarial prompts, thus laying the groundwork for tools suitable for proactive security analysis by law enforcement and developers. A workflow diagram visuzalizing the entire AI agents processes can be found in ***Appendix A.4***.

4 RESULTS & DISCUSSION

4.1 Classification Results

The following tables convey the differing average computation times of the 4 models mentioned previously to classify a paragraph.

Table 3. Average Classification Time Per Model

Classification Model	Avg Classification time (seconds)
gemma3:12b	8.49720305389544
gemma:7b	2.234373011525648
llama3:8b	0.6470358492216473
qwen:7b	0.9361249665090879

Table 4. Avg Comp Time per Model per Prompt Architecture

Model	Concise	Few Shot	Structured
gemma3:12b	8.768	9.354	7.370
gemma:7b	1.861	2.772	2.070
llama3:8b	0.499	0.623	0.820
qwen:7b	0.832	0.995	0.981

4.1.1 Average Computation Time Results. Table 3 shows the average computation time per model irrespective of prompt architecture. This provides valuable insight on how these models perform regardless of prompt architecture. Table 4 displays the average computation time per model, but respective of prompt architecture. This provides a deeper perspective on how these models perform in regards to computation speed, depending on the prompt architecture.

Key Insights :

- **Model Size**: gemma3:12b is slower because it's a larger model (12B parameters).
- **Architectural Efficiency**: llama3:8b and qwen:7b are inherently more efficient architectures for this classification task compared to the gemma models at similar (7-8B) parameter scales. This could be due to their inherent design or superior inference optimizations.

Prompt structure has a consistent impact related to expected token count and processing overhead:

(1) **Few Shot** prompts are generally the slowest (due to raw token count).
(2) **Concise** prompts are generally the fastest (due to minimal token count).
(3) **Structured** prompts vary, but for the more efficient models (llama3:8b, qwen:7b), the fixed overhead of parsing the structured format can be substantial, sometimes even approaching the cost of few-shot prompts or exceeding concise prompts by a notable margin. For gemma3:12b, its structured prompt seems unusually efficient given the other prompt types.

4.1.2 Score count per model per prompt structure. The following table shows the amount of paragraphs that were classified a particular score given a model and prompt structure. The first column is a pairing of the model used in classification and the corresponding usability score that will be counted in that row.

This data, derived from unlabelled paragraphs, highlights significant scoring biases or inherent propensities of each model and prompt combination, rather than their objective accuracy. These biases are critical given the absence of ground truth labels for classification.

- **Llama3:8b** shows that it consistently was able to classify the paragraphs with no problems and adhering to the required JSON format. This is evident as it rarely classifies 0 scores (meaning could not extract classification data because format was not as instructed). Additionally, it predominantly assigned scores of 1 and 2, with it favoring score 2 most with concise prompts. This model overall seemed to be more inclusive overall.
- **Gemma:7b** and **Qwen:7b** (Primarily for Few-Shot and Structured Prompts) exhibited that they were not able to adhere to the instructions of the prompts as they significantly classified the texts are 0. However **Qwen:7b** when using the concise

Table 5. Score Count per Model per Prompt Architecture

Model & Score	Concise	Few Shot	Structured
gemma3:12b, 1	86	138	111
gemma3:12b, 2	113	67	97
gemma3:12b, 3	15	9	6
gemma:7b, 0	150	182	162
gemma:7b, 1	17	6	27
gemma:7b, 2	22	19	15
gemma:7b, 3	25	7	10
llama3:8b, 1	42	152	165
llama3:8b, 2	139	52	38
llama3:8b, 3	33	0	11
qwen:7b, 0	13	210	214
qwen:7b, 1	0	0	0
qwen:7b, 2	41	0	0
qwen:7b, 3	160	4	0

prompts was able to perform as instructed, signifying that the prompts themselves would be the issue and not the models themselves.

- **Gemma3:12b** displays a more modereate but still conservative bias as it classifies majority of the texts as 1 (not relevant to prompt generation) but still scores a significant amount of texts as 2 and 3, which logically speaking makes sense as there are more instances of explaining concepts than direct examples.

These observations regarding the biases is a very crucial aspect to consider for real-world deployment. This heavily dictates how much information would be considered relevant by the proposed AI agent when processing unlabeled security data. Understanding these innate biases is extremely significant for model selection since selecting one whose internal scoring aligns with the desired level of inclusiveness or conservativeness for subsequent prompt generation, especially with the absence of explicit labels.

Table 6. Percentage of Agreement of Other Models vs. Gemma3:12b for Usability Score Classification

Model	Score 1	Score 2	Score 3
gemma:7b	6.34%	11.95%	14.04%
llama3:8b	45.56%	37.74%	31.58%
qwen:7b	0.00%	6.85%	34.21%

4.1.3 Percentage of Agreement Scoring of All Models vs Gemma3:12b per Score per Prompt. This section investigates the agreement percentage in usability score classification from all other models against **gemma3:12b**, utilizing it as a reference model. This approach is motivated by the general understanding that LLMs, with more extensive training datasets and greater parameter counts, often exhibit enhanced reasoning capabilities and are presumed to offer more accurate or robust classifications. The concept of "LLM-as-a-Judge"

Table 7. Percentage of Agreement between Other Models vs. Gemma3:12b for Usability Score Classification by Prompt Type

Model & Score	Concise	Few Shot	Structured
gemma:7b, 1	8.90%	1.67%	11.22%
gemma:7b, 2	17.97%	5.00%	9.38%
gemma:7b, 3	26.67%	33.33%	0.00%
llama3:8b, 1	35.17%	97.21%	7.82%
llama3:8b, 2	59.91%	41.00%	22.50%
llama3:8b, 3	60.00%	11.11%	0.00%
qwen:7b, 1	0.00%	0.00%	0.00%
qwen:7b, 2	20.28%	0.00%	0.00%
qwen:7b, 3	73.33%	0.00%	0.00%

[16] provides a strong precedent for using powerful & universal LLMs to evaluate and score textual content. This is even applied to scenarios with the absence of human-determined ground truth. This principle extends to classification tasks, where a larger model's sophisticated understanding and nuanced judgment can serve as a reliable benchmark for assessing the consistency and performance of other models handling unlabeled data. By comparing other models' classifications against gemma3:12b on the same unlabeled paragraph dataset, this approach aims to identify which models and prompt architectures exhibit a high degree of consistency with this reference.

The findings from Table 6, showcasing overall agreement, indicates :

- **llama3:8b demonstrates the highest average consistency with gemma3:12b**. With Score 1 (45.56%) and Score 2 (37.74%) being the highest.
- **Qwen:7b**, while showing a notable 34.21% agreement for Score 3, exhibits a complete disagreement for Score 1 (0.00%).
- Conversely, **gemma:7b** consistently shows the lowest agreement across all scores, suggesting a significant difference in its classification patterns compared to gemma3:12b.

Table 7, separates agreements by prompt type, highlighting critical insights.

- **Llama3:8b**'s agreement is exceptionally high for Score 1 when utilizing Few-Shot prompts (97.21%). Llama3:8b also maintains strong agreement with Concise prompts for Scores 1 (35.17%), 2 (59.91%), and for Score 3 (60.00%). ***Overall, Llama3:8b is the most consistent and reliable substitute for Gemma3:12b, especially with Few-Shot prompting.***
- In contrast, **Qwen:7b shows a remarkable 73.33% agreement for Score 3 when using Concise prompts**.
- **Gemma:7b's** agreement levels remain comparatively low across all prompt types, reinforcing its general inconsistency with gemma3:12b.

These results are crucial for selecting models that can reliably classify paragraphs for adversarial prompt generation with a smaller,

more efficient model rather than the generally larger and more resource-intensive models. From this we can conclude that Llama3:8b is the most reliable and consistent model when utilizing few-shot prompts. For concise prompts, **qwen :7b** was the most consistent. For structured prompts, the model with the highest percentage is lamma:8b, however the percentages are not sufficient enough to consider as a substitute model (7.82% for classifying score 1, and 22.5% for score 2).

4.2 Prompt Generation

Table 8. Prompt Generation Model Average Computation Time

Prompt Generation Model	Avg Generation Time
gemma3:12b	9.268
gemma:7b	2.347
llama3:8b	0.651
lamma3.1-ab	0.838
qwen:7b	1.332

Table 9. Average Computation Time per Model and Prompt Type

Model	Concise	Few-Shot	Structured
gemma3:12b	13.182	8.778	5.844
gemma:7b	2.521	2.136	2.383
llama3.1-ab	0.661	0.797	1.057
llama3:8b	0.561	0.535	0.857
qwen:7b	1.557	1.302	1.135

4.2.1 Average Generation Time per Model. Tables 8 & 9 convey the differing average computation times of the 5 models mentioned previously to generate adversarial prompts only from paragraphs classified by **gemma3:12b**.

Table 8 shows the average computation time per model irrespective of prompt architecture, providing insight into their baseline performance. Table 9 displays the average computation time per model respective to prompt architecture, offering a deeper perspective on how these models perform in regards to computation speed based on the specific prompt format. ***Key Insights:***

- **Model Size**: As expected, gemma3:12b consistently exhibits the slowest prompt generation times (9.268 seconds overall), reinforcing the principle that larger models generally incur higher computational overhead.
- **Architectural Efficiency**: llama3:8b (0.651 seconds overall) and llama3.1-ab (0.838 seconds overall) prove to be the most efficient architectures for prompt generation, significantly outperforming the gemma and qwen models at similar parameter scales.

Prompt structure has a distinct impact related to expected token count and processing overhead during generation:

- **Llama Models' Efficiency**: For llama3:8b and llama3.1-ab, Concise and Few-Shot prompts generally remain the fastest for prompt generation (e.g., llama3:8b at 0.561s for Concise, 0.535s for Few-Shot).
- **Structured Prompt Generation Variability**: Structured prompts can be surprisingly efficient for generation in some models. For gemma3:12b and qwen:7b, Structured prompts are notably the fastest for generating output (5.844s and 1.135s respectively), in contrast to the trend seen in paragraph classification(section 4.1.1). This indicates that providing a clear output structure can streamline the generation process for these models, potentially reducing search space or cognitive load compared to less constrained formats.
- **Inverted Concise Performance**: Interestingly, for gemma3:12b, Concise prompts result in the slowest generation times (13.182s). This inverse relationship compared to other models suggests that this prompt structure requires more internal processing or iterative refinement for gemma3:12b, leading to a higher computational cost.

This analysis highlights that the efficiency of a prompt type is not universal and depends significantly on the specific LLM architecture. For prompt generation, more detailed or structured prompts can sometimes reduce the processing time for the model, leading to faster results, even if they initially add to the input token count.

Table 10. Model Prompt Generated Text Word Length per Prompt Type in (Min, Average, Max) Format

Model	Concise	Few-Shot	Structured
gemma3:12b	(13, 53.11, 156)	(8, 35.82, 78)	(3, 21.16, 84)
gemma:7b	(16, 39.48, 116)	(12, 35.05, 69)	(6, 31.35, 156)
llama3:8b	(15, 28.81, 80)	(12, 24.85, 39)	(8, 26.62, 75)
llama3.1-ab	(2, 31.95, 93)	(3, 37.54, 93)	(2, 17.38, 43)
qwen:7b	(3, 57.20, 217)	(9, 47.88, 279)	(4, 28.18, 191)

4.2.2 Distribution of Generated Prompt Length: Table 10 illustrates the minimum, average, and maximum word counts of generated prompts across different models and prompt types. The data reveals consistent patterns in models' inherent text generation length biases and how effectively prompt structures regulate output length.

- **Llama3:8b** and **llama3.1-ab** consistently generate the shortest prompts by average word count (e.g., 28.81 and 31.95 for concise respectively).
- Conversely, **gemma3:12b** and **qwen:7b** tend to produce some of the longest prompts on average, with qwen:7b's concise prompts being the most verbose (57.20 words average).
- **Gemma:7b** was consistently average overall.

A key insight from this word length analysis is that structured prompts generally lead to the most concise outputs by word count for the majority of models (gemma3:12b, gemma:7b, llama3.1-ab, and qwen:7b). This infers that a clear & structural format often guides

the model more efficiently as it directly communicates what is to be expected and not expected. This reduces any ambiguities that can cause unnecessary words in the generated content. Understanding these distinct length biases is critical for real-world applications, as word count directly impacts factors such as ease of human review, potential for stealth or obfuscation within the prompt, and, due to token count, lower computational costs.

This section addresses RQ3 by evaluating the effectiveness of the AI techniques within the proposed agent through an analysis of its internal performance metrics. It quantifies the computational efficiency of various LLMs and prompt architectures during both paragraph classification and adversarial prompt generation phases. The section also examines the reliability of classification through score distribution biases and inter-model agreement, using Gemma3:12b as a reference to gauge consistency in identifying relevant security information. Furthermore, the effectiveness of prompt generation is assessed by analyzing the diversity and structural properties of the generated outputs, particularly their word length distributions across different models and prompt types. While direct empirical validation of adversarial prompt efficacy against target LLMs was beyond the scope due to practical constraints, this analysis provides crucial insights into the agent's capability to efficiently process information and produce structurally diverse and relevant adversarial prompts based on incident reports and guidelines.

5 LIMITATIONS & FUTURE WORK

- **Lack of Ground Truth Values & Primarily Unlabeled Dataset** : As aforementioned, one of the most significant limitations of this paper is the lack of a ground truth value stemming from the fact that the dataset is inherently unlabeled. To address this we used the concept of LLM-as-a-Judge[16] to establish a reference utilized as the grounf truth value. However, this method innately succumbs to the model's particular bias from its training set. A diverse set of professional cybersecurity analysts to classify the data set would be a subsequent future work to see how well these models perform when compared to the ground truth labels from the consensus of professionals. This was unfortunately not achievable during this research due to the limited time constraints.
- **Lack of Testing Adversarial Prompts on Target LLMs**: The other signficant limitation, which has also been mentioned, is the lack of direct testing of the adversarial prompts against target LLMs. While it was initially intended and planned to test these generated promtps against smaller and older models to see how practical the generated prompts were, it was unfortunately not feasible to do so as the amount of generated prompts were significantly larger than anticipated. Subsequently, a proper selection strategy describing which prompts to test was essential in order to perform these tests which proved to be difficult to achieve given the time constraints. This promotes the potential of a future work exploring and addressing this limitation. A future work that tests and evaluates adversarial attack prompts, generated from this AI agent, success rates against various models (parameter size, dates released, adversarial attack safeguards implemented) is extremely interesting. This will also quantify the practicality of the AI agent in a real-world setting.
- **RAG Implementation and Adaptive Attack Strategy Generation**: A crucial future direction to explore is the integration of Retrieval-Augmented Generation (RAG) with the AI agent. RAG will enable the AI agent to dynamically access and incorporate the most recent data from continuously updated security guidelines and incident reports. This proposes the following concept: completely replace the human necessity in adversarial prompt generation by mimicking the thinking process. By learning from the intricate details of all the adversarial prompt related data, such as; how they are structured individually and compared to other attacks, how they succeeded, what vulnerabilities they exploited, and how defenses were bypassed, it not only is able to recreate prompts at scale, but potentially create novel prompts, significantly reducing the reliance on human ingenuity for discovering new vulnerabilities and attack strategies.

6 CONCLUSION

This research successfully identified existing methodologies and significant limitations in publicly available resources for understanding and reproducing adversarial attacks against AI systems, highlighting the critical gap in specific attack documentation. To address this, the study designed a novel AI agent that effectively leverages various LLMs through a two-phase workflow, paragraph classification and adversarial prompt generation, to process security guidelines and incident reports, automatically creating adversarial prompts. Furthermore, the effectiveness of the AI techniques within this agent was evaluated based on their computational efficiency in processing and generating content, the reliability of information extraction through classification, and the structural diversity and characteristics of the generated prompts. Ultimately, this work contributes a systematic approach to enhancing proactive security analysis for law enforcement and developers by streamlining the generation of relevant adversarial prompts.

The github repository for all the code used in this research can be found ***here***. Additionally contains all the technical details & information.

7 AI STATEMENT

The author of this research paper used AI tools to support the academic writing process and the development of advanced scripting tasks. Gemini was employed to improve the structure, and coherence. It also assisted in generating and refining Python scripts, regular expressions, and prompts in developing the AI agent. All AIgenerated content was critically reviewed and edited by the author to ensure accuracy, clarity, and originality.

REFERENCES

[1] M. Moradi, K. Yan, D. Colwell, M. Samwald, and R. Asgari, "A critical review of methods and challenges in large language models," *Computers, Materials & Continua*, vol. 82, no. 2, 2025.
[2] T. W. A. of Large Language Model-Assisted Writing Across Society arXiv, "The widespread adoption of large language model-assisted writing across society," *arXiv preprint*, n.d.
[3] A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, *Adversarial machine learning*. Cambridge University Press & Assessment, 2019.
[4] H. Kim and W. Park, "Text-based prompt injection attack using mathematical functions in modern large language models," *MDPI*, vol. 13, no. 24, 2024.
[5] Systematically Analysing Prompt Injection Vulnerabilities in Diverse LLM Architectures, "Systematically analysing prompt injection vulnerabilities in diverse LLM architectures," in *International Conference on Cyber Warfare and Security*, vol. 20, 2025.
[6] ResearchGate, "October 11, 2024 | security and auditing tools in large language models (LLM)," *ResearchGate*, n.d.
[7] OWASP, "OWASP top 10: LLM & generative AI security risks." https://owasp.org/www-project-top-10-for-large-language-model-applications/, n.d.
[8] Technet24, "The AI revolution in networking, cybersecurity, and emerging technologies." https://dl1.technet24.ir/Downloads/EBooks/Network/The-AI-Revolution-in-Networking.pdf, n.d.
[9] T. Perry, "Hacker tricks chatbot into selling him a car for $1," *Upworthy*, Dec. 2023.
[10] L. Law, "Samsung data leak: Engineers send confidential data to ChatGPT," *legaldata.law*, Apr. 2023.
[11] A. B. autonomous exploitation of adversarial example defenses arXiv, "AutoAdvExBench: Benchmarking autonomous exploitation of adversarial example defenses," *arXiv preprint*, n.d.
[12] F. Perez and I. Ribeiro, "Ignore previous prompt: Attack techniques for language models," *arXiv preprint*, 2022.
[13] S. Jiang, X. Chen, and R. Tang, "Prompt packer: Deceiving llms through compositional instruction with hidden attacks," 2023.
[14] A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," 2023.
[15] A. Paulus, A. Zharmagambetov, C. Guo, B. Amos, and Y. Tian, "Advprompter: Fast adaptive adversarial prompting for llms," 2025.
[16] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging llm-as-a-judge with mt-bench and chatbot arena," 2023.

A APPENDIX OF WORKFLOW DIAGRAMS

A.1 Paragraph Scraping

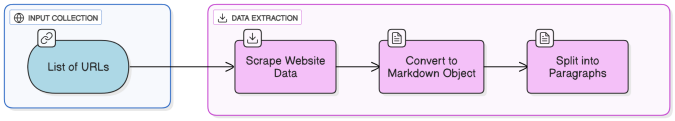

Fig. 1. Paragraph Scraping Work Flow Diagram

A.2 Paragraph Classification Workflow Diagram

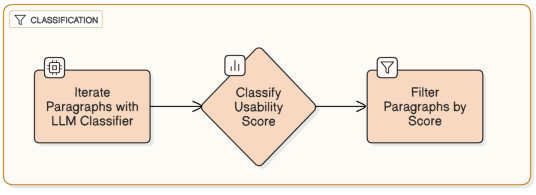

Fig. 2. Paragraph Classification Work Flow Diagram

A.3 Prompt Generation Workflow Diagram

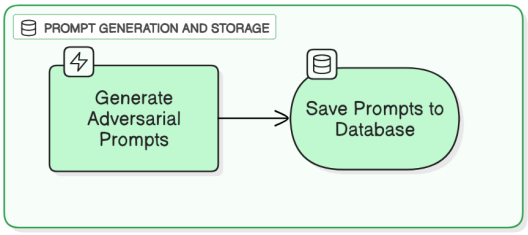

Fig. 3. Prompt Generation Work Flow Diagram

A.4 Complete Workflow Diagram

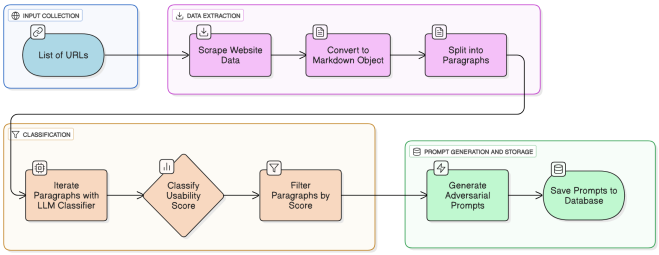

Fig. 4. Complete Work Flow Diagram