False Alarms, Real Damage: Adversarial Attacks Using LLM-based Models on Text-based Cyber Threat Intelligence Systems

Samaneh Shafee, Alysson Bessani, Pedro M. Ferreira

sshafee@fc.ul.pt, anbessani@fc.ul.pt, pmferreira@ciencias.ulisboa.pt
LASIGE, Faculty of Sciences, University of Lisbon, Lisbon, Portugal

Abstract—Cyber Threat Intelligence (CTI) has emerged as a vital complementary approach that operates in the early phases of the cyber threat lifecycle. CTI involves collecting, processing, and analysing threat data to provide a more accurate and rapid understanding of cyber threats. Due to the large volume of data, automation through Machine Learning (ML) and Natural Language Processing (NLP) models is essential for effective CTI extraction. These automated systems leverage Open Source Intelligence (OSINT) from sources like social networks, forums, and blogs to identify Indicators of Compromise (IoCs). Although prior research has focused on adversarial attacks on specific ML models, this study expands the scope by investigating vulnerabilities within various components of the entire CTI pipeline and their susceptibility to adversarial attacks. These vulnerabilities arise because they ingest textual inputs from various open sources, including real and potentially fake content. We analyse three types of attacks against CTI pipelines - evasion, flooding, and poisoning- and assess their impact on the system's information selection capabilities. Specifically, on fake text generation, the work demonstrates how adversarial text generation techniques can create fake cybersecurity and cybersecurity-like text that misleads classifiers, degrades performance, and disrupts system functionality. The focus is primarily on the evasion attack, as it precedes and enables flooding and poisoning attacks within the CTI pipeline. Our findings reveal that the False Positive Rate (FPR) for the evasion attack reached 97% for a specialized ML classifier model, indicating the model's high vulnerability to adversarial samples. Additionally, an FPR of 75% is observed for ChatGPT-4o as a classifier, indicating its susceptibility to adversarial examples. These results underscore the need for an additional verification component at the early stage of the CTI pipeline to detect and filter out misinformation before it spreads through the system.

Index Terms—Cyber Threat Intelligence, Fake text generation, Adversarial attacks, Chatbots, Security Operation Centers, Open Source Intelligence, Natural Language Processing, Generative AI

1 Introduction

The increasing complexity of cyber threats has made it difficult for existing security measures to provide effective protection [1]. Security systems such as firewalls and Intrusion Detection Systems (IDS) often cannot prevent breaches and fail to detect the spread of malicious activities. An enhancement is to keep defense mechanisms up to date with appropriate configurations while enriching them with relevant and up-to-date threat information. One promising approach to enhance defensive capabilities is Cyber Threat Intelligence (CTI), which provides insights into the tactics, techniques, and procedures (TTPs) used in modern cyber-attacks, along with the corresponding detection and mitigation strategies. CTI is an emerging field that analyses trends in cybercrime, hacktivism, and cyber spying by utilizing diverse sources, including open-source intelligence (OSINT), social media, and human intelligence [2]. An important challenge for CTI systems is the potential for attackers to inject fake intelligence across OSINT sources such as Twitter (currently named X[1]), Stack Overflow, dark web forums, and blogs, which can compromise the reliability of the intelligence gathered and pose challenges for organizations using such sources.

To address this, CTI pipelines have typically relied on trusted sources to reduce the risk of ingesting fake or manipulated information. More broadly, CTI data collection in the literature has followed two main approaches: (1) keyword-based extraction [3, 4] and (2) source-based curation from trusted entities [5, 6]. Keyword-based methods do not address the problem of misinformation, as they are vulnerable to ambiguous or misleading terms. Relying on trusted sources also introduces limitations: their accuracy and availability may degrade over time, and maintaining an up-to-date trusted source list is difficult, especially when such sources become compromised or outdated.

This work explores the behavior of CTI pipelines when data is collected from diverse and potentially unverified sources. This alternative approach avoids the limitations of relying solely on trusted sources and opens the possibility of exploring dark web forums. However, it becomes more vulnerable to misinformation and deliberate manipulation by adversaries. The CTI extraction tools' failure to filter false or misleading information makes them vulnerable to adversarial exploitation. Attackers can manipulate these systems and compromise the integrity of the gathered threat intelligence.

1. We use Twitter through the paper as our main dataset was published before the name change.

In this paper, we design an integrated CTI pipeline capturing the essential stages found across existing systems. This integrated pipeline serves as a unified reference framework for assessing the CTI pipeline against adversarial manipulation.

Despite their sophisticated workflows [7], each pipeline stage remains susceptible to adversarial attacks, such as evasion, flooding, and poisoning. These attacks can severely impact the reliability of the pipeline, leading to incorrect predictions by ML models, monitoring and validation disruptions, and increased False Positive (FP) and False Negative (FN) outcomes.

While prior work has emphasized the benefits of automation in CTI extraction by ML models, the impact of adversarial manipulation on data sources has received limited attention. This work focuses on this overlooked intersection, where automation interacts directly with deception. It assesses vulnerabilities of the CTI extraction pipeline rather than introducing entirely new attack methodologies, aiming to demonstrate how adversaries can create a platform to deploy fake data to systematically deceive CTI models and systems. Understanding these deception mechanisms allows for evaluating the robustness of automated threat intelligence extraction processes more effectively. In summary, the contributions of this work are as follows:

1) An integrated CTI extraction pipeline is proposed that captures and generalizes the typical stages and components used across prior CTI pipelines.
2) The analysis and investigation of the common limitations of CTI pipelines.
3) A comprehensive exploration of the threat landscape encountered by automated CTI systems, including attacks where adversaries intentionally induce FPs by injecting misleading information into the system.
4) An evaluation of the pipeline performance after applying evasion, flooding, and poisoning attacks.

The remainder of the paper is organized as follows. Section 2 introduces key definitions establishing the basis for the study. Section 3 explains the five stages of the proposed CTI pipeline and Section 4 presents a deep exploration of potential attacks on CTI pipelines. Section 5 focuses on the progress of implementing evasion, flooding, and poisoning attacks. The experimental results are presented in Section 6, along with our findings. Section 7 discusses the study's challenges, opportunities, and future work. Section 8 reviews related studies about adversarial text generation and adversarial attacks on ML models. Finally, Section 9 summarizes the key contributions and insights derived from this study.

2 Preliminary Definitions

This study focuses on a proposed CTI system that ingests textual data from OSINT to generate actionable alerts. This section provides fundamental definitions related to the input textual data types and their predicted outputs by ML models, which are the core of CTI systems.

Real vs. Fake text. It is important to distinguish between real and fake input text in CTI systems, but there are no universally standardized definitions. Typically, text generated by a machine is labeled as fake, while human-written text is considered real [8]. However, this study adopts a task-specific definition: real text refers to genuine security-related information that reflects actual cyber threats or vulnerabilities; fake text refers to false or misleading information crafted to deceive either humans or automated systems. Such fakes may use cybersecurity-like terminology while not corresponding to real-world threats. For example, a real security-related tweet is: *"Vulnerability Details: CVE-2024-52046 (CVSS 10/10) Apache MINA Remote Code Execution (RCE) Vulnerability."* A fake version could be: *"Exploit released for CVE-2024-52046 allowing full control over Apache MINA servers. No patch available yet—act now to secure your systems!"*. Although the fake version mimics the style and vocabulary of a real CTI statement, it conveys false urgency and misleading content.

Figure 1 illustrates the base taxonomy of text input used in this study. It categorizes text as either real or fake based on its truthfulness and intent. Furthermore, fake texts are subcategorized based on whether they are machine-generated or human-written.

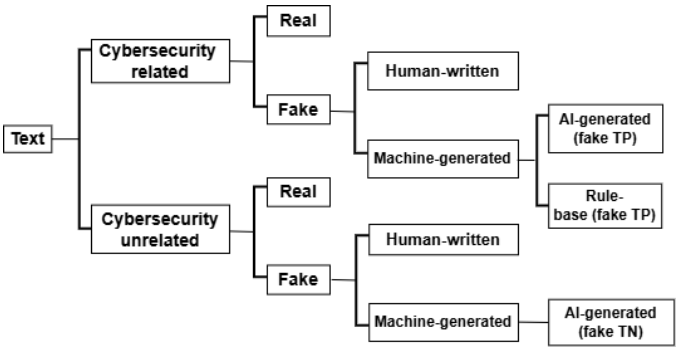

Fig. 1. Taxonomy of input text in a CTI Pipeline.

True vs. False Positives in CTI Classification. To evaluate the effectiveness of classification models used in the CTI pipeline, we must distinguish between correct and incorrect prediction outcomes. This study uses classic True Positive (TP) and False Positive (FP) definitions, focusing on binary classifier models that differentiate between security-related and unrelated texts. A TP reflects an accurate classification of a genuine CTI instance, while an FP often arises in adversarial scenarios where fake or unrelated content is intentionally crafted to resemble CTI text. For example: *"Vulnerability Details: CVE-2024-52046 (CVSS 10/10) Apache MINA Remote Code Execution (RCE) Vulnerability."* is a TP if correctly classified as CTI. In contrast: *"System Upgrade Details: CVE-2025-527656 (Performance Rating: 10/10) Apache MINA Remote Configuration Expansion (RCE) Module."* is an FP if the model misclassifies this fabricated security-irrelevant text as a real CTI alert due to the use of technical jargon and CVE-like formatting. This distinction is critical for analysing model performance, particularly under adversarial testing scenarios that seek to exploit the classifier's reliance on superficial lexical cues.

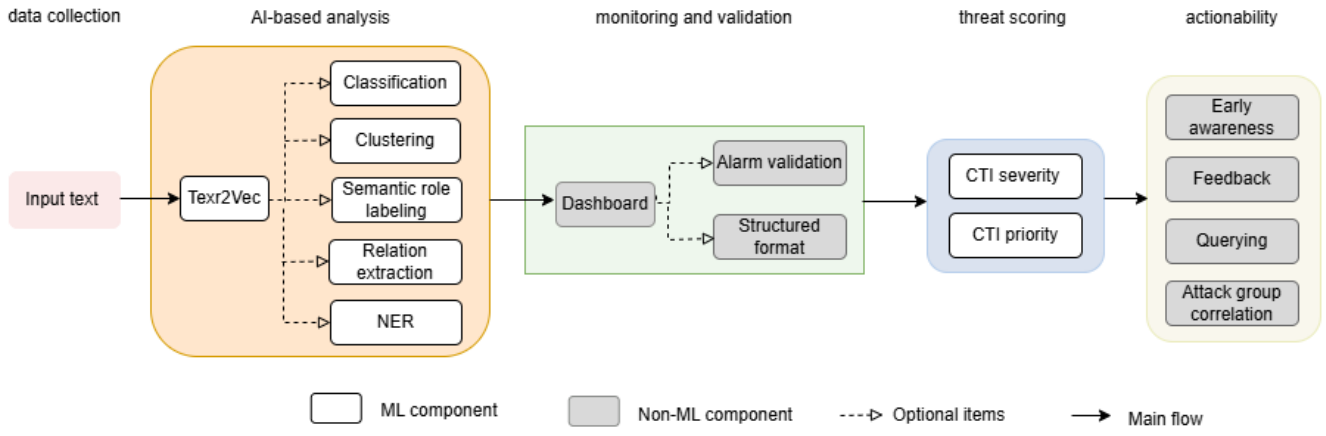

Fig. 2. Proposed integrated CTI extraction pipeline.

3 TEXT-BASED CTI PIPELINE

CTI extraction involves several key stages necessary to identify and extract relevant information, such as IoCs and TTPs. Although many CTI pipelines have been proposed, they typically feature custom stages and components customized for specific organizational or research objectives. An integrated text-based CTI extraction pipeline is proposed that unifies multiple extraction approaches into a structured framework. As depicted in Figure 2, it consists of five main stages: data collection, AI-based analysis, monitoring and validation, threat scoring, and actionability. This design captures functionalities observed in prior CTI systems while consolidating them into a unified framework suitable for empirical evaluation under adversarial attack scenarios. The following subsections describe each stage as adapted from prior works to form the integrated pipeline.

3.1 Data collection

At the data collection stage, raw textual inputs are gathered from various sources, including social media platforms, security blogs, vendor reports, CVE, and threat databases. These inputs may contain early IoCs, emerging threats, or discussions on vulnerabilities. The reliability and diversity of the collected data play a vital role in shaping the downstream analysis. Some CTI systems restrict data collection to trusted sources to minimize FPs. In contrast, others broaden their scope to include informal or unverified channels, which may enhance coverage but could introduce greater data variability and require additional filtering.

3.2 AI-based analysis

Researchers have employed NLP techniques alongside ML models to extract CTI from textual data. Although the extraction methods vary depending on the type of information, the overall process of CTI extraction in the pipeline remains the same. This applies whether extracting IoCs, TTPs, or other relevant threat intelligence components. Figure 2 provides an overview of techniques commonly used in CTI extraction pipelines, including classification, clustering, Named Entity Recognition (NER), semantic role labeling, and relation extraction [6, 9, 10, 11, 12, 13, 14, 15, 16, 17]. These techniques enable models to identify patterns, extract key entities, and map relationships in text. Table 1 lists specific examples applied in various studies that use diverse data sources and describes the contribution of the techniques to the purpose of the CTI tasks. It is important to note that the implementation details and sequence of techniques may differ among studies based on the specific goals and datasets used.

3.3 Monitoring and validation

The monitoring stage comprises key components such as the dashboard, alarm validation, and structured format. The dashboard provides a visual overview of recent events, which helps security operations center (SOC) users maintain situational awareness and respond to threats, as supported by research [15] and tools such as [20]. These implementations underscore the importance of dashboards in centralizing information and supporting decision-making in cybersecurity operations. Alarm validation is a valuable task performed by a skilled human analyst who reviews and verifies alerts generated earlier in the pipeline [21]. The validated alerts are then organized and shared in structured formats such as STIX or TAXII [22, 23].

3.4 Threat scoring

This stage evaluates the severity and priority of detected threats [24]. Severity reflects the potential impact based on factors such as exploitability and system exposure, while priority defines the response order considering severity and likelihood of exploitation [25]. Priority helps allocate resources effectively and ensures a timely response [26]. Severity is typically labeled as Critical, High, Medium, or Low, whereas priority includes Urgent, ASAP, Within 24 hours, and Low Priority levels [27]. This distinction improves decision-making and response efficiency in SOCs.

3.5 Actionability

This stage of the CTI pipeline builds on foundational components to enable organizations to anticipate, analyse, and act on threats. Key functions include:

TABLE 1
Overview of AI-based methods for extracting CTI from textual data.

Ref	Data sources	Extraction Techniques	Purposes
[16]	Publicly available incident reports	Classification	Automate cyber threat attribution
[17]	Cybersecurity reports	Classification, NER	Automate IoC detection
[9]	Twitter, CVE, Wikitext dataset	Classification	Detect cybersecurity-related text
[18]	CVE and ExploitDB, Security Blogs, Hacker Forum Posts	Classification	Vulnerability similarity analysis, IoC recognition
[6]	Security blogs, vendor bulletins, and hacking forums	Domain recognizer	IoC recognition, Threat severity
[14]	Twitter	Classification, NER	Detect cybersecurity-related text and extract cyber threat entities
[15]	Twitter	Classification & Clustering	Detect aggregate cybersecurity-related text
[12]	CTI reports	Semantic role labeling	Extraction of attack behavior, Threat hunting
[11]	Publicly Available Sources	NER, relation extraction	Knowledge graph construction from extracted entities and their relationships in a graph database
[10]	CTI Reports	TTPClassifier, NER	Attack Pattern Predictions, Transform unstructured CTI information into a structured knowledge graph
[13]	Threat analysis reports	Classification and Sequence Tagging, NER, relation extraction	Detect semantically similar attack patterns
[19]	Dataset of ref [14]	Classification, NER	Detect cybersecurity-related text and extract cyber threat entities

Early awareness. Extracting OSINT CTI from online platforms enables security professionals to identify potential threats before they fully emerge. Alves et al. [5] demonstrated that security-relevant tweets can be detected up to 148 days before NVD vulnerability disclosure, allowing proactive defense and risk mitigation.

Querying CTI. Structured CTI allows efficient querying of relevant information [28, 29]. For example, CTI extracted from hacker communities can be organized in a searchable portal where users can filter data by time or resource type to identify specific patterns or events [2, 30]. Such systems improve the accessibility and usability of CTI and enable security teams to make more effective, informed decisions. Structured querying of security knowledge graphs also identifies vulnerabilities in software libraries before deployment.

Feedback. Some CTI systems use feedback loops where SOC teams evaluate the quality of the extracted data, filtering out inaccuracies and improving future extractions by retraining models based on quality standards [31].

Attack group correlation. The extracted CTI helps link patterns between incidents with known threat actors and provides information on their tactics and priorities. This correlation helps to identify threats and allows defense contractors to prioritize responses based on the behavior and capabilities of the attacker [32].

4 Attacks on CTI pipeline

Each component of the CTI pipeline can become a potential target for adversaries aiming to compromise it. Therefore, we analyse attacks targeting automated text-based CTI pipelines, including their capabilities, knowledge, and goals.

4.1 Overview of attacks

This study examines three possible attacks to illustrate key threats. The objective is to provide a concise reference that explains the mechanisms and implications of each attack and demonstrates their impact on various system components, including ML models, the dashboard, and alarm validation.

Evasion attack. Evasion attacks occur during the test step, with trained models targeted to be misled. Once the extractor pipeline is trained, adversaries can generate Fake Negative (FaN) input texts to fool the model into misclassifying them as positives. This manipulation leads to increased FP and FN texts appearing on the dashboard, confusing SOC professionals and diverting attention from real threats [33]. In the test scenario, this attack focuses on generating FaN inputs that resemble real cybersecurity texts but are intentionally designed to cause misclassifications. For example, for a binary classifier that distinguishes between security- and non-security-related texts [14], an adversary can lead the model to misclassify non-security texts as security-related, possibly disrupting decision-making.

This attack is expressed as $\text{argmax}_{\hat{x}} \{\text{Loss}\left(f\left(\hat{x}\right), y\right)\}$, where $f\left(\hat{x}\right)$ is the classifier, $\hat{x}$ is the modified input, and y represents the true label. The goal is to modify inputs x to maximize the loss function, leading the model to incorrect predictions. This indicates how small targeted changes to the input can result in misclassification, which is central to evasion attacks. Evasion attacks can be categorized into two main approaches: maximum-confidence and minimum-distance [34]. Maximum-confidence attacks aim to create adversarial examples that are misclassified with high confidence, but often involve substantial changes to the input that decrease its similarity. In contrast, minimum-distance

attacks focus on minor and subtle modifications to maintain high similarity and make detection more difficult.

The minimum distance approach is adopted, since preserving the similarity of adversarial texts is crucial to bypass classifiers and aligns with the need for realistic adversarial examples in cybersecurity. Evasion attacks are a foundation for other attacks. A successful evasion attack increases the risk of poisoning attacks, where FPs enter the training set and corrupt future model retraining. Additionally, increased FPs can flood analyst workflows, leading to delayed responses to real security threats.

Flooding attack. Flooding attacks [35] overwhelm the CTI extraction pipeline by injecting a high volume of deceptive FaN and Fake Positive (FaP) texts into the system dashboard, making it difficult for security analysts to distinguish between real and misleading alerts. Additionally, attackers may use this technique to conceal their malicious activities within the flood of data to evade detection. This strategy can be considered a form of denial-of-service (DoS) attack, where the system becomes overwhelmed with excessive traffic, preventing it from processing legitimate requests. As a result, real positives become obscured, diminishing the system's ability to handle real threats. This overload weakens the reliability of the threat intelligence pipeline and increases ambiguity in threat detection, making it harder for analysts to respond to genuine risks. Beyond disrupting decision-making, flooding attacks waste the analyst's effort, leading to system failure and degrading the overall security response efficiency. Repeatedly encountering false alerts results in a loss of trust in the model's predictions, reducing confidence in automated classification outcomes. Furthermore, overwhelming SOC analysts strains security operations and effectively impairs their ability to manage real threats. The excessive volume of injected data also increases costs owing to log storage, requiring more computational resources to process and archive irrelevant information. This not only affects infrastructure overhead but also directly impacts the analyser's ability to process and interpret incoming data accurately. When encountering fake inputs, the analyser has two possible reactions: discard them, wasting computational resources and analyst time; or misclassify them as genuine, leading to a poisoning attack.

Poisoning attack. Poisoning attacks [36] manipulate training data to degrade CTI extraction model accuracy and are classified into integrity and availability attacks. Integrity attacks alter specific data points to cause targeted misclassification. Availability attacks modify large portions of the training set and reduce overall model reliability [37]. These attacks primarily target the system's test step and training set components. Evasion attacks can facilitate poisoning by contaminating the training data and allowing access to undetected FaN or FaP texts. If these are misclassified as TPs or TNs, respectively, they corrupt the model's learning process and degrade model performance. The consequences can be significant, generating false security alerts, spreading misinformation across interconnected systems and professionals, and causing analysts to respond to non-existent threats while overlooking real ones.

4.2 Attacker conceptual model

Understanding the knowledge, capabilities, and objectives of the attacker is essential for modeling realistic threat scenarios, including how these factors appear in evasion, flooding, and poisoning attacks. These factors influence how and where the adversary may attempt to compromise the CTI pipeline, as different attacker profiles adopt different strategies.

Attacker's capability. In the evasion attack, the attacker generates adversarial inputs with minimal detectable changes to exploit model vulnerabilities and degrade performance. In the flooding attack, the attacker injects coherent false data streams to overwhelm the system, disrupting data integrity and operational efficiency. Additionally, in a poisoning attack, the attacker generates FaN texts that closely resemble real security-related texts, deceive both the ML models and analysts, and ultimately corrupt the training dataset.

Attacker's knowledge. The attacker's knowledge is formalized by a tuple $\kappa \in K$, where K represents the abstract knowledge space encompassing all knowledge dimensions an attacker might have about the target. Specifically, $\kappa = (D, X, f, L, w)$, where D denotes the training data, X represents the feature set of the victim model, f refers to the learning algorithm, L is the training objective function, and w corresponds to the trained model's parameters. The attacker cannot directly access D but may approximate it using publicly available OSINT datasets, such as cybersecurity repositories (e.g., CVE) that could overlap with D. Using these resources, the attacker constructs a substitute model to mimic the target system's behavior. While features X, f, and L remain unknown, the attacker can infer likely ones based on common cybersecurity and ML practices. Finally, the trained model's parameters w are entirely inaccessible. In this black-box attack scenario, the attacker's knowledge κ is a constrained instance of K, reflecting the practical limitations attackers face. Black-box attacks are more challenging than white-box or gray-box attacks because they rely only on input-output interactions without assuming any knowledge of the model [38].

Attacker's goal. In the evasion attack, the attacker aims to increase the percentage of FP, misleading the analyser and compromising the system's reliability. Misclassification disrupts decision-making, increases errors, and risks of overlooking real threats hidden among FPs. The flooding attack overwhelms dashboards with excessive FaN and FaP texts, shifting the focus from actual threats, causing system slowdowns, crashes, analyst fatigue, or infrastructure strain, ultimately reducing security effectiveness and costs. In a poisoning attack, the attacker aims to corrupt the learning process of the model by generating deceptive texts that are mistakenly included in the training dataset.

5 Methodology

This section outlines the methods for simulating evasion, flooding, and poisoning attacks in the CTI extraction pipeline, considering the data set, target models, text generation, and attack procedures.

5.1 Dataset

A publicly available Twitter dataset with 31281 tweets [39] was the basis for generating FaN and FaP texts. Although it was collected in 2016, it offers unique advantages that make it suitable for this study. Since the target models are binary classifiers, a dataset with a binary classification structure is essential. It was employed to train the target model [14], ensuring experimental consistency and compatibility with the attack scenario. It includes manually labeled named entities extracted from the cybersecurity-related tweets, essential for generating realistic FaP texts in the flooding attack setup. The dataset includes a classification label ("relevant class") that indicates whether a tweet is security-related (1) or not (0). Although the tweet sizes of the dataset are limited to 256 characters, the FaP and FaN texts generated for evasion, flooding, and poisoning attacks vary in length. The clean_tweet feature in the dataset, which contains refined tweet text, was used to create adversarial examples.

5.2 Target models

Two models are targeted in the experiments. The first is a state-of-the-art binary classifier developed by Dionisio et al. [14] based on Convolutional Neural Networks (CNNs), employed within CTI extraction pipelines to recognize whether a tweet is related to cybersecurity or not. This classifier was selected based on an evaluation [40], which identified it as an effective classifier. Additionally, it was trained on real tweets, which makes it a suitable and realistic target for adversarial evaluation. The second target model is ChatGPT-4o, selected for exceptional performance in classifying security-related and non-security texts [41]. Unlike traditional classifiers, ChatGPT-4o integrates contextual understanding with classification tasks to effectively detect subtle cyber threat patterns. Moreover, the model's widespread adoption and versatility make it a practical benchmark for testing adversarial robustness in real-world scenarios. By targeting ChatGPT-4o, we aim to evaluate the resilience of cutting-edge language models against evasion attacks. The evasion attack targets both models, while the flooding and poisoning attacks are specific to the specialized classifier [14].

5.3 Adversarial text generation

Adversarially generated FaN texts are crucial for all the attacks considered in this study. While previous research has explored adversarial text generation in non-security domains [42, 43, 44, 45], these approaches often lack public implementations or detailed documentation. Furthermore, while some of these methods are complex, the advent of large language models (LLMs) has simplified the process. This study demonstrates that even publicly accessible LLMS, such as GPT, can be effectively prompted without fine-tuning, generating adversarial inputs that closely resemble cybersecurity-related texts. They can mimic the features and structures of real text, and their advanced ability to follow instructions allows for a highly controlled and flexible generation process. This observation reveals a major vulnerability in CTI pipelines, as they can be manipulated using generic tools without the need for sophisticated, domain-tuned generation frameworks. To better understand how this vulnerability arises, this work investigates how LLMs can generate adversarial texts that closely mimic the language and structure of cybersecurity content. Open-source LLMs like LLaMA 3, including the 8 billion (8B) and 70 billion (70B) parameter versions [46] released in April 2024, were assessed for their potential in adversarial text generation. However, they failed to meet quality requirements, lacking diversity and producing unrealistic adversarial samples.

ChatGPT-4o was leveraged to generate texts that mimic the structure and terminology of cybersecurity texts while remaining unrelated to real cybersecurity. While LLM-based text generation has limitations, such as potential biases, repetition, and inaccuracies [47], the method proposed in this study exceeded the expectations. To generate effective adversarial texts with ChatGPT-4o, prompts were designed using real tweets. Figure A1 in Appendix A shows the prompt utilized for generating FaN texts, which incorporates key tokens to achieve the desired style while excluding specific security details [48]. An attention-based mechanism is proposed to extract the contextually key tokens from the real cybersecurity tweets. In the following, we first define the problem of generating adversarial text and then explain the generation process and its components as shown in Figure 3.

Problem Definition. The problem can be stated as: *How can we generate FaN texts by modifying input text while preserving its cybersecurity-like semantics?* Let $x = \{t_1, t_2, \ldots, t_n\}$ represent a text sample consisting of n tokens, where a token can be a word, subword, or special character, depending on the tokenizer's segmentation process. The sample x belongs to the sample space X and the target classification model $F : X \to Y$ assigns x to a class $y \in Y$, i.e., $F(x) = y$.

To generate FaN texts, key tokens $t_{\text{imp}} \subset x$ are identified using an attention-based mechanism that quantifies the importance of each token based on its contribution to the model's decision. Unlike conventional adversarial attacks that directly perturb important tokens, the proposed approach preserves t_{imp}, modifying instead surrounding tokens to shift the overall meaning while maintaining a cybersecurity resemblance. The transformation process modifies tokens around t_{imp}, ensuring t_{imp} remains unchanged to retain the structure of a cybersecurity-related text without conveying real security text.

Important token extraction. Using a pre-trained SecureBERT model [49], the input texts are tokenized and embedded in vector representations. SecureBERT is a specialized language model trained on cybersecurity corpora, enhancing the effectiveness in understanding and distinguishing security-related text [50]. The embeddings are processed by a Long Short-Term Memory (LSTM) layer to capture sequential relationships. The LSTM is a recurrent neural network designed to model temporal sequences and capture long-term dependencies, effectively addressing issues like the vanishing gradient problem [51]. Subsequently, an attention layer assigns weights α_i to each token t_i, reflecting its contextual importance:

$$\alpha_i = \frac{\exp(e_i)}{\sum_{j=1}^{n} \exp(e_j)}, \quad e_i = \tanh(h_i W_a + b_a),$$

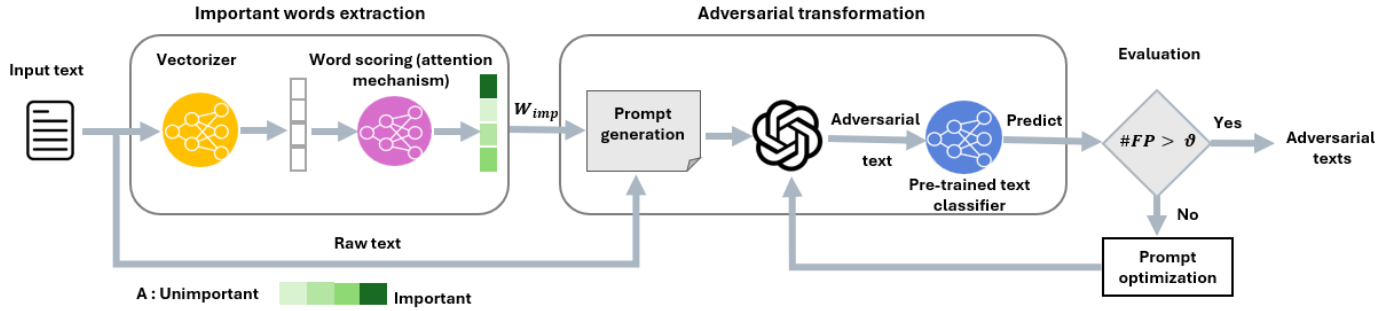

Fig. 3. Illustration of generating adversarial text using the attention mechanism and ChatGPT-4o. The generated adversarial text is input to a pre-trained binary classifier to mislead its predictions. A: The color intensity indicates the importance of words, ranging from less important (light) to highly important (dark). The top three most important tokens in each tweet were incorporated into the prompt.

where e_i is the intermediate relevance score of t_i in the given context, h_i is the hidden state of the LSTM, and W_a and b_a are the trainable parameters of the attention layer. The top-k tokens with the highest attention scores α_i contribute the most to the model decision and are selected as t_{imp}, the most influential tokens in the text.

Adversarial transformation. First, based on empirical analysis to select the threshold, the three most important tokens, $t_{\text{imp}} = \{t_{\text{imp1}}, t_{\text{imp2}}, t_{\text{imp3}}\}$, are identified. The threshold aims to balance the need for sufficient perturbation to influence the model's output while preserving the original semantic structure of the text. Formally, given an input sequence of tokens,

$$x = \{t_1, t_2, \ldots, t_{k-1}, t_{\text{imp1}}, t_{k+1}, \ldots, t_{l-1}, t_{\text{imp2}}, t_{l+1}, \ldots, t_{m-1}, t_{\text{imp3}}, t_{m+1}, \ldots, t_n\},$$

a transformed text is generated,

$$\hat{x} = \{t_1^*, t_2, \ldots, t_{k-1}^*, t_{\text{imp1}}, t_{k+1}, \ldots, t_{l-1}^*, t_{\text{imp2}}, t_{l+1}^*, \ldots, t_{m-1}, t_{\text{imp3}}, t_{m+1}^*, \ldots, t_n\},$$

where $t_{\text{imp1}}, t_{\text{imp2}}, t_{\text{imp3}}$ remain unchanged to maintain key cybersecurity indicators. Instead of modifying important tokens, ChatGPT-4o replaces some surrounding tokens, denoted by t_i^*, with contextually similar but modified terms that eliminate security-related content while preserving the overall sentence structure. The positions of t^* are not the same for all sentences. They are dynamically determined based on the text structure and context. To achieve this, ChatGPT-4o selects replacements from domains unrelated to security that are suggested in the prompt. The modified text must meet two primary objectives: *(1) Preservation of the textual structure and semantic similarity*, so the generated text closely resembles a cybersecurity-related text structurally and semantically; *(2) Changing the security-related nature of the text* to shift the content away from cybersecurity topics.

Figure 3 illustrates the FaN text generation methodology[2]. The attention mechanism identifies the top three tokens, and ChatGPT-4o modifies surrounding words as suggested in the prompt to create adversarial texts that still appear cybersecurity-related. The prompt follows an iterative refinement, where each cycle evaluates the generated FaNs based on their ability to deceive the pre-trained text classifier used in our pipeline. The prompt is considered adequate if the classifier produces a number of FPs ($\#FP$) exceeding a predefined threshold ϑ. Otherwise, another refinement iteration is executed. The finalized prompt was effective enough to surpass the ϑ setting of 80% of the input texts. It was then used to generate the adversarial FaN texts, forming the basis for subsequent evaluation.

Human evaluation. The last iteration of prompt optimization involved a human evaluation step, performed to further validate the effectiveness of the generated FaN texts. The data was distributed to three analysts who were informed that the texts were fake and were tasked to identify those that resembled cybersecurity texts. This result was vital to achieving a final prompt surpassing the ϑ threshold. The importance of the iterative prompt refinement with human validation is demonstrated by the difference between the initial and final FPR presented in Section 6.

5.4 Evasion attack

The evasion attack was conducted by passing the adversarially generated FaN texts through the target classifiers. A successful attack occurs when a FaN is incorrectly classified as positive, resulting in an FP. Otherwise, it is considered a TN. The effectiveness of the attack is quantified using the FPR, allowing an assessment of the vulnerability of the classifier to adversarially generated inputs at inference time.

5.5 Flooding attack

A flooding attack is executed using a combination of FaN texts, paraphrased FaP, and rule-based generated FaP texts.

Paraphrasing TP texts. Multiple variations of a single real TP text are generated through paraphrasing. These texts serve as an effective means of flooding the model with slightly different yet semantically similar inputs. To achieve this, GPT-3.5-Turbo API was employed as a paraphrasing tool, applying it to the clean_tweet feature of the dataset. By generating ten paraphrases for each tweet from the positive class (1) 110740 FaP input texts were created.

Rule-based fake texts. The generation of FaP texts using a rule-based approach involves replacing words based on their named entity type to preserve contextual consistency. The tweets include entity features such as organization names, product names, vulnerabilities, and version information. Among the 11074 positive tweets, 4552 include

2. Code will be available on https://github.com/samanehshf/Fake-CTIs

organization names, 10218 mention product names, 5137 contain vulnerabilities, and 3640 refer to version information. To ensure coherence, the replacement words for each entity type are sourced from other occurrences of the same type within the dataset. This guarantees that substitutions remain contextually valid and do not introduce semantic inconsistencies. Entities within each entity type are categorized into predefined semantic groups, ensuring that replacements occur only within the same conceptual category. For example, we categorized vulnerability entities into attack types, attributes, and execution methods. Within the attack type category, terms such as ransomware, trojan, and malware are substituted only with other attack types. This structured approach ensures the generated FaP texts maintain a realistic cybersecurity-related appearance.

Flooding attack mechanism. The mechanism of a flooding attack on a CTI extraction system is illustrated in Figure 4, showing the workflow for injecting FaN and FaP texts. The honest security hacker on the left generates real positive threat intelligence texts (green arrows). Conversely, an adversary generates and injects a large volume of fake texts of different types, designed to bypass the trained model. The trained model filters some of the inputs but fails to reject all malicious inputs, leading to a significant number of FaP (green, yellow) and FaN texts (red) passing through and appearing on the dashboard, along with the real TP text (green). The overwhelming volume of excessive and conflicting alerts confuses analysts, making it challenging to differentiate between real and deceptive inputs and leading to unnecessary investigation time and errors, such as discarding real threats (green) or leaving fake alerts in the system.

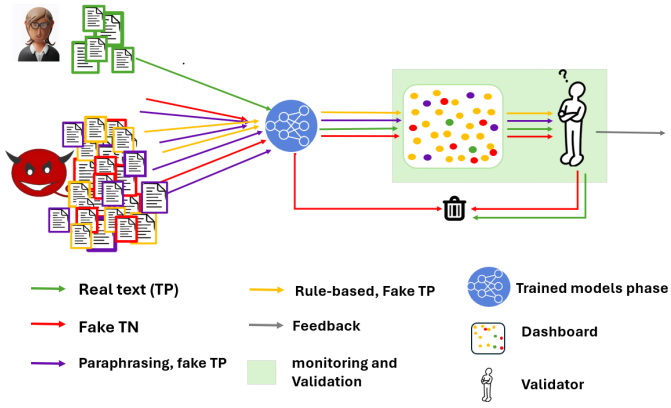

Fig. 4. Flooding attack workflow.

5.6 Poisoning attack

In the proposed CTI pipeline (Figure 2), a poisoning attack can occur through feedback in the actionability stage. Alerts confirmed by analysts are incorporated into the dataset used to retrain the model, introducing a vulnerability that adversaries can exploit. This poisoning attack gradually corrupts the retraining process, building upon successful evasion or flooding attacks described in previous subsections.

To demonstrate the impact of a poisoning attack, FaN texts that were misclassified as positive (FP) by the model [14] and human analysts in the experimental evaluation of the evasion attack were gradually injected into the model training dataset over several retraining rounds. In each round, a random number of FaNs were injected to simulate realistic and unpredictable successive poisoning conditions. As a result, the classifier begins to gradually learn incorrectly, which impacts its classification decision boundaries.

Although not demonstrated in this work, poisoning can also occur through flooding attacks, from which also FaP texts can reach the training set. In this case, not only FPs affect model classification performance, but also FaPs can introduce model bias and data imbalance.

5.7 Evaluation methodology

The attacks studied are characterized in Table 2 in terms of the adversarial text input type, purpose, and the corresponding classifier responses, explaining how the different attacks exploit the model's vulnerabilities and the evaluation metrics used in the experiments. In evasion and flooding attacks, the adversary generates FaN texts to cause the model to misclassify them as FP samples, thus increasing the FPR. In flooding attacks, the adversary further intensifies the disruption by generating FaP texts, creating a flood of adversarial texts to overwhelm the system and deceptively increase the True Positive Rate (TPR). Moreover, the presence of FaN texts in flooding attacks amplifies evasion attacks. The poisoning attack can cause the model to misclassify input into FP or FN, which are better captured by the F_1 score.

6 Experimental Results

The effectiveness of the proposed adversarial FaN text generation method was evaluated by applying it to the three types of attacks considered. After generating the FaN texts as described in Section 5.4, three cybersecurity professionals reviewed one-third of the data each, flagging the samples they judged not to resemble security content. Based on the feedback, 1340 out of 11074 samples were rejected, resulting in a final dataset of 9734 FaN texts resembling cybersecurity. A semantic similarity score was computed to assess how closely the generated texts resemble the real cybersecurity tweets, helping determine whether the fake texts potentially maintain cybersecurity-related tweets' typical structure and semantic meaning. The filtered dataset was then used as input in the three attack scenarios.

Semantic similarity Scores. The Cosine Similarity (CS) was used to quantify the semantic similarity between real and FaN tweets [52]. It is mathematically defined as,

$$\mathrm{CS}(A,B) = \frac{\mathbf{m}\cdot\mathbf{n}}{\|\mathbf{m}\|\cdot\|\mathbf{n}\|} = \frac{\sum_{i=1}^{k} m_i \cdot n_i}{\sqrt{\sum_{i=1}^{k} m_i^2}\cdot\sqrt{\sum_{i=1}^{k} n_i^2}}, \quad (1)$$

where A and B represent the feature vectors of two tweets and $\mathbf{m}$ and $\mathbf{n}$ correspond to their respective embeddings. To visualize how the similarity scores are distributed within the dataset, Figure 5 presents a scatter diagram of all computed cosine similarity values between FaN texts and real positive tweets, sorted in ascending order. Each point reflects the semantic similarity between a generated FaN and a real positive tweet. The distribution shows that most scores fall

TABLE 2
Classification of fake text inputs in evasion, flooding, and poisoning attacks based on their class, intent, and model outcomes.

Attacks	Adversary		Model prediction		
	Generated fake text	Intent	Positive	Negative	Eval
Evasion	FaN	Misclassify as TP	FP	TN	FPR
Flooding	FaN	Flooding dashboard, and possibly amplify a FP evasion attack	FP	TN	FPR
	Paraphrasing, FaP	Flooding the dashboard, amplify TP	TP	FN	TPR
	Rule-Based, FaP		TP	FN	
Poisoning	FaN	Degrade decision boundaries	FP	FN	F_1 score

between 0.5 and 0.8, indicating moderate semantic closeness. A smaller number of samples have scores below 0.5, while a few approach 1.0, suggesting high similarity in certain cases.

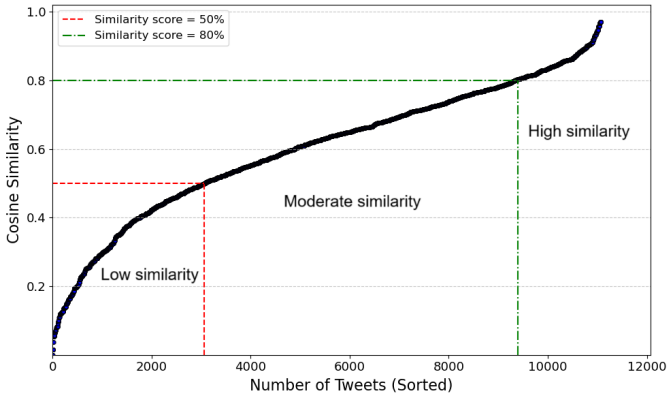

Fig. 5. Distribution of semantic similarity scores within the dataset of tweets.

6.1 Evasion attack

The evasion attack's overall efficiency is analysed by assessing the FPR and gradient distribution.

False Positive Rate. The FPR measures how well the generated texts misled the classifier, representing the proportion of non-security generated texts incorrectly classified by the model as security-related. Table 3 compares the FPR achieved by ChatGPT-4o and the binary classifier proposed by [14], both evaluated on the original generated text and on the final text optimized by prompt refinement. ChatGPT-4o has a dual role in this experiment: as a generator of adversarial texts and a classifier responsible for detecting security-related content. This reveals a key weakness of the model, as it produced misleading samples but failed to classify them correctly. The results show a significant improvement in FPR due to prompt refinement. Regarding the models, the Dionisio et al. [14] model is significantly more vulnerable to adversarial attacks, with an FPR of 87% for the original texts and 97% for the optimized texts. Even with a very high FPR, ChatGPT-4o is significantly less vulnerable than the specialized model. These findings

TABLE 3
FPR (%) of adversarial text generated by ChatGPT-4o on the Twitter dataset [14], evaluated both in its original form and after optimization.

Models	Original text	Optimised text
ChatGPT-4o	0.51	0.75
[14]	0.87	0.97

underscore how adversarial optimized prompt amplifies the FPR and reveal different levels of classifier robustness.

Gradient distribution. Gradients can be used to quantify the model's sensitivity to minor input variations, being essential for assessing robustness against adversarial attacks [53]. The target binary classifier model [14] is assessed by evaluating FaN adversarial text gradient behavior against the baseline gradient of real texts. Kernel density estimation [54] is employed, which is a non-parametric technique for estimating the Probability Density Functions (PDFs). Figure 6 presents PDF plots for real (blue) and adversarial (yellow) gradients. Dashed vertical lines indicate the mean of each distribution. Both exhibit Gaussian-like shapes, with means centered near zero. This result emphasizes three key aspects: (1) adversarial inputs effectively mimic the overall gradient structure of the real text; (2) slight differences, such as a narrower variance for adversarial gradients, reveal distinct characteristics useful for attack detection; and (3) some high-magnitude gradients are replaced by ones closer to zero, indicating a feature cancellation effect. Supporting metrics, including gradient means, variances, and KL divergence, are provided in Table 4. The adversarial gradient mean and variance closely match real text, suggesting effective imitation. However, the slightly narrower variance of adversarial gradients indicates reduced content diversity, likely due to constraints in the adversarial generation process. The KL divergence (fake $\rightarrow$ real) confirms that both PDFs are very similar, reinforcing the effectiveness of the attack. The cosine distance confirms the directional similarity between gradients, indicating that adversarial inputs affect the model while preserving detectable features. Finally, the Wasserstein distance [55] suggests a strong geometric alignment between adversarial and real gradients [56].

6.2 Flooding attack

The flooding attack experimental results presented in in Table 5 showcase the impact of the FaN and FaP texts in

TABLE 4
Statistical comparison of gradients for real and FaN texts.

Metrics	fake	Real
Mean	-1.10e-06	-1.28e-06
Variance	2.30e-05	2.67e-05
Wasserstein distance	-	0.0012
KL divergence	-	0.05
Cosine distance	-	0.85

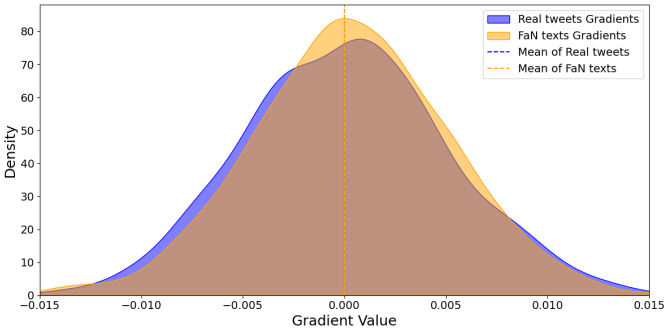

Fig. 6. Kernel Density Estimation Plot of real tweet and FaN text gradients.

the performance of the specialized ML binary classifier [14]. Each text type influenced the classifier's performance differently. The target model misclassified 85% FaN instances as security-related (FP). For paraphrased FaP texts, the model correctly identified 88% instances as TP but predicted the remaining as FN. Finally, for the rule-based approach, the classifier correctly identified 82% of cases but generated 4282 FNs. These results highlight the high adversarial efficiency of ChatGPT-4o and show that the efficiency of the FaP generation techniques is similar, with a small advantage for the LLM.

6.3 Poisoning attack

The impact of the poisoning attack is evident in the efficiency with which the incremental FaN injection in the training dataset over successive retraining iterations degrades the classifier's performance. Recall that the evaluation of the poisoning attack is done on the basis of 9402 FaN tweets that deceived the model and passed the human validation.

Table 6 reports the FN, FP, precision, recall, and F_1 score results of the targeted model, over the successive retraining rounds, simulating progressive poisoning attacks. In addition, the table explicitly shows the number of injected samples per round and their cumulative total. The model retains a performance above 90% in the early stages, until the cumulative number of poisoned samples reaches 4500. In the fourth round, the F_1 score, precision, and recall dropped more sharply to around 0.85. In the seventh round, after injecting all the fake samples, the classifier shows a significant performance drop, with F_1 score little above 0.5 (0.57), demonstrating the growing inability of the model to correctly identify cybersecurity-related samples. This progression demonstrates how incremental poisoning gradually undermines the model decision boundaries.

Figure 7 shows the decline in F_1 score, precision, and recall during the retraining rounds, emphasizing the temporal degradation of model performance metrics. Although the attack does not directly inject FN texts, accumulating mislabeled instances confuses the classifier's internal class concepts, causing it to misclassify genuine security-related text more frequently. Therefore, the recall metric drops considerably from the sixth round, separating from Precision and F_1 Score.

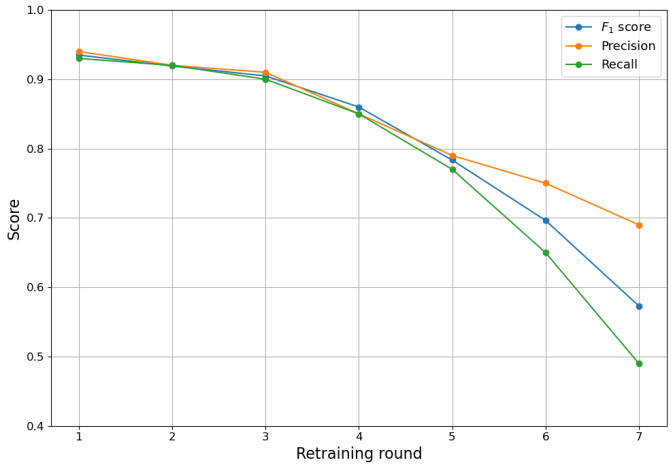

Fig. 7. Effect of poisoning attack via FaN on Dionisio et al. [14] model over retraining rounds.

7 Discussion

This study reveals a consistent pattern of vulnerability throughout the stages of the CTI extraction pipeline when confronted with adversarially generated texts. Despite leveraging advanced ML models and LLMs, the system remains highly susceptible to deceptive inputs crafted to resemble genuine cybersecurity content. The results show that publicly accessible generative models like ChatGPT can be effectively prompted without fine-tuning to produce adversarial sentences that convincingly mimic security-related text. Although these fake texts are entirely synthetic and have no real-world cybersecurity relevance, the classifier consistently mislabels them as genuine CTI. Regarding poisoning attacks, it was observed that early-stage retraining with adversarially injected FP samples did not immediately degrade model performance. However, with continued injection over multiple retraining rounds, the model exhibited a considerable drop in recall and overall F_1 score, which demonstrates the cumulative impact of progressive data poisoning. In the flooding attack scenario, paraphrasing tweets using the generative model to create FaP is more effective at evading detection than rule-based techniques. This suggests that semantic coherence in adversarial inputs plays a significant role in misleading CTI extractors. These findings underscore the urgent need for early-stage defenses and robust verification mechanisms to safeguard automated CTI pipelines against evolving adversarial threats. In the following, we discuss key challenges, opportunities, and directions for future improvements.

Text length. This study utilizes a Twitter-based dataset, where each sample is limited to 256 characters. The short length restricts attackers' ability to generate highly deceptive FaN texts. Short texts lack space for complex details,

TABLE 5
Flooding attack on Dionisio et al. [14] classifier (victim model) based on three types of prepared texts

Text generation techniques	Fake Text Generator	Text	TP	FP	TN	FN	FPR	TPR
Adversarial texts generation	GPT-4o	11074 (FaN)	0	9402	332	0	0.97	0
Paraphrasing	GPT-3.5-Turbo	110740 (FaP)	97865	0	0	12875	0	0.88
Rule-based	semantic grouping	23547 (FaP)	19266	0	0	4282	0	0.82

TABLE 6
Effect of poisoning attack on the victim classifier model [14] over multiple retraining rounds. Each row represents the model performance after injecting additional FaN samples.

Round	FN	FP	Recall	Precision	F_1 score	FaN Injected	Cumulative FaN
1	734	638	0.93	0.94	0.9349	500	500
2	921	910	0.92	0.92	0.9199	1000	1500
3	1000	890	0.90	0.91	0.9048	3000	4500
4	1508	1500	0.85	0.85	0.8599	1500	6000
5	750	1000	0.77	0.79	0.7834	1242	7242
6	3876	2399	0.65	0.75	0.6964	1492	8734
7	5537	2373	0.49	0.69	0.5730	668	9402

enabling defenders to more easily detect anomalies like non-security-related patterns or generic language in adversarially generated texts. This constraint represents a worst-case scenario for the attacker, as the limited space leaves little room to manipulate the tweets in subtle ways. However, longer texts provide attackers with more space to embed deceptive content that mimics cybersecurity terminology, potentially increasing the likelihood of evading detection. For instance, when prompted with a longer real tweet, ChatGPT can generate longer synthetic texts that blend security-related jargon with misleading information, which are harder to distinguish as FaN texts. On the other hand, longer texts are prone to inconsistencies or errors due to architectural limitations in transformer-based models, such as input truncation or degraded attention mechanisms [57, 58]. These models often struggle to maintain coherence in lengthy texts, as they cannot effectively process long-range dependencies, leading to detectable flaws like contradictory statements. Consequently, attackers should balance the deceptive potential of longer texts against the risk of introducing noise, while defenders should leverage these inconsistencies to improve FaN text detection.

Human analysts. One of the major costs in the FaN generation method proposed is the reliance on human experts to manually review and validate adversarially generated FaN texts. This introduces a human overhead, especially when scaling the evaluation to large datasets. The recruited security experts can make judgment errors during the evaluation of generated FaN texts. Experts might recognize cybersecurity resembling texts differently depending on their experience, fatigue, or contextual understanding. Moreover, a key limitation in our proposed CTI pipeline is that human experts' evaluations in the FaN text generation phase are utilized for the alarm validation component in the monitoring and validation stage in the CTI pipeline. Then, if the FaN texts deceive the analysts, they are added to the training dataset, and the model is retrained. While the experts were tasked with labeling texts based on their resemblance to cybersecurity content, their prior involvement in generating these texts could introduce bias, as they knew the texts were artificially generated. It is important to note that the goal is not to evaluate the experts' ability to detect fake texts but to recognize the potential bias in their assessments due to their knowledge that the texts were generated. This approach was chosen due to resource constraints, which avoided using separate evaluators for the generated FaN texts and alarm validation component of the monitoring and validation stage in the proposed CTI pipeline in Figure 2.

Alleviate fake text in CTI pipelines. One of the fundamental limitations of CTI extraction pipelines is the absence of a robust fact-checking mechanism at the early stages to filter fake or misleading inputs. Without such components, evasion attacks can easily inject FaN texts [59, 60]. Although fact-checking can be a promising approach to mitigate evasion attacks by validating extracted information, it remains insufficient against paraphrased inputs used in flooding attacks. This is because variants of a text often preserve semantic plausibility while bypassing exact-match verification. Fact Checker assists in reducing the burden on analysers in the monitoring and validation stage by pre-filtering clearly invalid or unverifiable content, thereby improving system efficiency and robustness.

Future research should explore integrating AI-assisted fact-checking mechanisms for evasion attacks in the following directions: (1) Source credibility assessment: fact-checkers can analyse the provenance of input texts, especially those collected from social networks such as Twitter, by evaluating account metadata such as age, posting behavior, bot likelihood, and reputation indicators. In addition, URLs found in content can be examined using WHOIS data [61, 62] and archival records. Threat intelligence services such as VirusTotal, which aggregate blacklists of known malicious domains, can help assess the legitimacy of linked sources. (2) Content-based validation: fact-checking can be performed using techniques that assess data consistency against structured cybersecurity corpora (e.g., CVE, MITRE ATT&CK), such as semantic entailment, contradiction detection, and entity linking. However, it is essential to recognize that some vulnerabilities and threats often first emerge on informal platforms like Twitter [5]. To maintain the early-warning capabilities of CTI systems, verification frameworks should not rely solely on structured databases like CVE. They should also assess whether the reported information makes sense in its context and matches expected timelines.

While sourcing data from informal platforms can help detect early threats, it also creates challenges for real-world CTI use. Organizations differ in how they act on incoming intelligence. For example, national security teams or research groups may find value in collecting data from forums or the dark web. However, enterprise security teams, particularly in sectors such as finance, telecommunications,

and healthcare, typically prioritize timely and verified threat intelligence to ensure operational continuity and minimize false alarms. Therefore, CTI systems must adjust their data collection policies based on who uses them and how much uncertainty they can tolerate.

8 Related work

This study demonstrates that text-based CTI extraction pipelines are vulnerable to adversarial attacks. However, to the best of our knowledge, limited research directly investigates text-based attacks on CTI pipelines. Existing studies primarily focus on two areas: (1) adversarial text generation, which involves creating misleading text to challenge models, and (2) adversarial attacks, which involve generating or modifying text inputs to bypass detection mechanisms.

Recent approaches have explored how misleading or manipulative texts can be generated and leveraged in evasion, poisoning, or flooding attacks targeting machine learning systems. For instance, Ranade et al. fine-tuned a GPT-2 model to generate plausible CTI descriptions from an initial prompt intending to mislead cyber-defense systems. However, this study lacks validation metrics to assess the quality of the generated texts, such as their similarity to human-written descriptions [42]. Furthermore, its reliance on the older GPT-2 model limits the effectiveness and realism of the outputs, especially when compared to state-of-the-art transformer architectures available today.

However, several studies have explored adversarial text generation outside the cybersecurity domain. For example, ARGH [63] is a framework that fine-tunes GPT-2 to generate social media rumors on topics such as COVID-19 and politics, reducing the detection accuracy for humans and machines. Ren et al. [45] proposed a method using a conditional VAE-GAN to generate adversarial movie reviews, improving scalability and fluency while misleading classifiers. Another study introduced Grover [44], a controllable text generation model designed to synthesize fake news articles to train and evaluate fake news detection systems more effectively. Grover leverages GPT-2 to produce raw outputs; however, this approach suffers from limited realism in the generated text. Additionally, in a user study, Huynh et al. observe that outputs randomly generated by GPT-2 are often easily distinguishable by human evaluators. This observation emphasizes the need for more refined adversarial text generation techniques [43].

Beyond text generation, researchers are exploring how adversarial texts can be applied in practical attack scenarios against machine learning models. EaTVul adopts a multi-step attack framework that selects vulnerable text samples using support vector machines (SVMs). It then generates adversarial text using LLMs and finally refines the generated samples through a fuzzy genetic algorithm to maximize attack effectiveness [64]. This approach has demonstrated up to a 100% success rate in evasion scenarios and proves its effectiveness in generating high-quality adversarial examples. TextJuggler [65] employs a black-box, word-level attack using a BERT-based model to identify key tokens affecting a classifier's decision boundary. These tokens are minimally perturbed through insertion or substitution, preserving semantic similarity and linguistic fluency. Furthermore, locality-sensitive hashing (LSH) minimizes the number of model queries, improving overall attack efficiency. Experimental results show that TextJuggler outperforms baselines regarding Attack Success Rate (ASR), textual similarity, and fluency on various classification tasks [65]. TextGuise [66] combines word and sentence-level perturbations to craft high-quality adversarial examples with minimal distortion. It achieves over 80% ASR at perturbation ratios below 0.2 and outperforms prior methods across three top classifiers and five datasets. Its strong transferability and multilingual support further demonstrate its effectiveness.

While prior studies have introduced various adversarial text generation strategies, key challenges persist, such as maintaining semantic realism, validating outputs, and adapting to transformer models. Generating fake CTI content remains difficult due to the specialized language of cybersecurity and the demand for large-scale adversarial data. EaTVul focuses on high-quality generation via optimization, while TextJuggler and TextGuise offer efficient methods that preserve semantics, textual fluency, and ASR.

9 Conclusion

This paper presented an overview of vulnerabilities in automated CTI extraction systems. We identified stages of the CTI pipeline and analysed how adversaries exploit them through evasion, flooding, and poisoning attacks. To demonstrate the practical impact of vulnerabilities, we conducted empirical experiments these attacks, revealing that CTI systems are highly susceptible to adversarial manipulation and require immediate mitigation. While automation is essential for handling the increasing volume of cyber threat data, our findings demonstrate that it also introduces new security challenges that adversaries can exploit. Addressing these challenges requires a balanced approach that strengthens automation while ensuring robust security measures. Future research should focus on developing adaptive defense mechanisms to counter fake text entering CTI systems and ensuring that CTI systems remain resilient and effective in evolving cyber threat landscapes.

Acknowledgement

This work was funded by the European Commission through the SATO Project (H2020/IA/957128) and by FCT through the LASIGE Research Unit (UID/00408/2025 - LASIGE) and Ph.D. grant (2023.00280.BD).

References

[1] M. R. Rahman, R. M. Hezaveh, and L. Williams, "What are the attackers doing now? automating cyberthreat intelligence extraction from text on pace with the changing threat landscape: A survey," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–36, 2023.

[2] S. Samtani, R. Chinn, H. Chen, and J. F. Nunamaker Jr, "Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence," *Journal of Management Information Systems*, vol. 34, no. 4, pp. 1023–1053, 2017.

[3] H. Shin, W. Shim, S. Kim, S. Lee, Y. G. Kang, and Y. H. Hwang, "# twiti: Social listening for threat intelligence," in *Proceedings of the Web Conference 2021*, 2021, pp. 92–104.

[4] A. Bose, V. Behzadan, C. Aguirre, and W. H. Hsu, "A novel approach for detection and ranking of trendy and emerging cyber threat events in twitter streams," in *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2019, pp. 871–878.

[5] F. Alves, A. Andongabo, I. Gashi, P. M. Ferreira, and A. Bessani, "Follow the blue bird: A study on threat data published on twitter," in *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*. Springer, 2020, pp. 217–236.

[6] J. Zhao, Q. Yan, J. Li, M. Shao, Z. He, and B. Li, "Timiner: Automatically extracting and analyzing categorized cyber threat intelligence from social data," *Comput. Secur.*, vol. 95, p. 101867, 2020.

[7] D. Schlette, M. Caselli, and G. Pernul, "A comparative study on cyber threat intelligence: The security incident response perspective," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2525–2556, 2021.

[8] E. N. Crothers, N. Japkowicz, and H. L. Viktor, "Machine-generated text: A comprehensive survey of threat models and detection methods," *IEEE Access*, vol. 11, pp. 70 977–71 002, 2023.

[9] H.-S. Shin, H.-Y. Kwon, and S.-J. Ryu, "A new text classification model based on contrastive word embedding for detecting cybersecurity intelligence in twitter," *Electronics*, vol. 9, no. 9, p. 1527, 2020.

[10] M. T. Alam, D. Bhusal, Y. Park, and N. Rastogi, "Looking beyond iocs: Automatically extracting attack patterns from external cti," in *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, 2023, pp. 92–108.

[11] H. Jo, Y. Lee, and S. Shin, "Vulcan: Automatic extraction and analysis of cyber threat intelligence from unstructured text," *Computers & Security*, vol. 120, p. 102763, 2022.

[12] K. Satvat, R. Gjomemo, and V. Venkatakrishnan, "Extractor: Extracting attack behavior from threat reports," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 598–615.

[13] N. Rastogi and M. T. Alam, "Cyber threat intelligence for soc analysts," 2023.

[14] N. Dionísio, F. Alves, P. M. Ferreira, and A. Bessani, "Towards end-to-end cyberthreat detection from twitter using multi-task learning," in *2020 international joint conference on neural networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[15] F. Alves, A. Bettini, P. M. Ferreira, and A. Bessani, "Processing tweets for cybersecurity threat awareness," *Information Systems*, vol. 95, p. 101586, 2021.

[16] U. Noor, Z. Anwar, T. Amjad, and K.-K. R. Choo, "A machine learning-based fintech cyber threat attribution framework using high-level indicators of compromise," *Future Generation Computer Systems*, vol. 96, pp. 227–242, 2019.

[17] Z. Long, L. Tan, S. Zhou, C. He, and X. Liu, "Collecting indicators of compromise from unstructured text of cybersecurity articles using neural-based sequence labelling," in *2019 international joint conference on neural networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[18] J. Zhao, Q. Yan, X. Liu, B. Li, and G. Zuo, "Cyber threat intelligence modeling based on heterogeneous graph convolutional network," in *23rd international symposium on research in attacks, intrusions and defenses (RAID 2020)*, 2020, pp. 241–256.

[19] B. Cui, J. Li, and W. Hou, "Atdg: An automatic cyber threat intelligence extraction model of dpcnn and bigru combined with attention mechanism," in *International Conference on Web Information Systems Engineering*. Springer, 2023, pp. 189–204.

[20] S. EMK. (2020) CTI extractor – ECHO network. [Online]. Available: https://www.echocti.com/en/

[21] N. Afzaliseresht, Y. Miao, S. Michalska, Q. Liu, and H. Wang, "From logs to stories: Human-centred data mining for cyber threat intelligence," *IEEE Access*, vol. 8, pp. 19 089–19 099, 2020.

[22] M. R. Rahman and L. Williams, "From threat reports to continuous threat intelligence: a comparison of attack technique extraction methods from textual artifacts," *arXiv preprint arXiv:2210.02601*, 2022.

[23] O. Briliyant, N. P. Tirsa, and M. A. Hasditama, "Towards an automated dissemination process of cyber threat intelligence data using stix," *2021 6th International Workshop on Big Data and Information Security (IWBIS)*, pp. 109–114, 2021.

[24] M. T. Alam, D. Bhushl, L. Nguyen, and N. Rastogi, "Ctibench: A benchmark for evaluating llms in cyber threat intelligence," *arXiv preprint arXiv:2406.07599*, 2024.

[25] R. Kerkdijk, S. Tesink, F. Fransen, and F. Falconieri, "Evidence-based prioritization of cybersecurity threats," 2021.

[26] F. Jalalvand, M. Baruwal Chhetri, S. Nepal, and C. Paris, "Alert prioritisation in security operations centres: A systematic survey on criteria and methods," *ACM Computing Surveys*, 2024.

[27] M. Vielberth, F. Böhm, I. Fichtinger, and G. Pernul, "Security operations center: A systematic study and open challenges," *Ieee Access*, vol. 8, pp. 227 756–227 779, 2020.

[28] A. Piplai, S. Mittal, A. Joshi, T. Finin, J. Holt, and R. Zak, "Creating cybersecurity knowledge graphs from malware after action reports," *IEEE Access*, vol. 8, pp. 211 691–211 703, 2020.

[29] L. Neil, S. Mittal, and A. Joshi, "Mining threat intelligence about open-source projects and libraries from code repository issues and bug reports," in *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2018, pp. 7–12.

[30] S. Samtani, K. Chinn, C. Larson, and H. Chen, "Azsecure hacker assets portal: Cyber threat intelligence and malware analysis," in *2016 IEEE conference on intelligence and security informatics (ISI)*. Ieee, 2016, pp. 19–24.

[31] zvelo. (2020) Cyber threat intelligence (CTI): Analysis, dissemination, and feedback. [Online]. Available: https:

//zvelo.com/cti-analysis-dissemination-feedback/
[32] J. Li, J. Liu, and R. Zhang, "Advanced persistent threat group correlation analysis via attack behavior patterns and rough sets," *Electronics*, vol. 13, no. 6, p. 1106, 2024.
[33] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, "A taxonomy and survey of attacks against machine learning," *Computer Science Review*, vol. 34, p. 100199, 2019.
[34] D. Li and Q. Li, "Adversarial deep ensemble: Evasion attacks and defenses for malware detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3886–3900, 2020.
[35] R. Saranya, S. S. Kannan, and N. Prathap, "A survey for restricting the ddos traffic flooding and worm attacks in internet," in *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, 2015, pp. 251–256.
[36] A. E. Cinà, K. Grosse, A. Demontis, B. Biggio, F. Roli, and M. Pelillo, "Machine learning security against data poisoning: Are we there yet?" *Computer*, vol. 57, no. 3, pp. 26–34, 2024.
[37] Z. Tian, L. Cui, J. Liang, and S. Yu, "A comprehensive survey on poisoning attacks and countermeasures in machine learning," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–35, 2022.
[38] S. Bhambri, S. Muku, A. Tulasi, and A. B. Buduru, "A survey of black-box adversarial attacks on computer vision models," *arXiv preprint arXiv:1912.01667*, 2019.
[39] N. Dionísio, F. Alves, P. M. Ferreira, and A. Bessani, "Cyberthreat detection from twitter using deep neural networks," in *2019 international joint conference on neural networks (IJCNN)*. IEEE, 2019, pp. 1–8.
[40] S. Altalhi and A. Gutub, "A survey on predictions of cyber-attacks utilizing real-time twitter tracing recognition," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–13, 2021.
[41] S. Shafee, A. Bessani, and P. M. Ferreira, "Evaluation of llm-based chatbots for osint-based cyber threat awareness," *Expert Systems with Applications*, p. 125509, 2024.
[42] P. Ranade, A. Piplai, S. Mittal, A. Joshi, and T. Finin, "Generating fake cyber threat intelligence using transformer-based models," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–9.
[43] L. Huynh, T. Nguyen, J. Goh, H. Kim, and J. B. Hong, "Argh! automated rumor generation hub," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3847–3856.
[44] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending against neural fake news," *Advances in neural information processing systems*, vol. 32, 2019.
[45] Y. Ren, J. Lin, S. Tang, J. Zhou, S. Yang, Y. Qi, and X. Ren, "Generating natural language adversarial examples on a large scale with generative models," in *ECAI 2020*. IOS Press, 2020, pp. 2156–2163.
[46] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.
[47] J. J. Y. Chung, E. Kamar, and S. Amershi, "Increasing diversity while maintaining accuracy: Text data generation with large language models and human interventions," *arXiv preprint arXiv:2306.04140*, 2023.
[48] K. Huang, G. Huang, Y. Duan, and J. Hyun, "Utilizing prompt engineering to operationalize cybersecurity," in *Generative AI Security: Theories and Practices*. Springer, 2024, pp. 271–303.
[49] E. Aghaei, X. Niu, W. Shadid, and E. Al-Shaer, "Securebert: A domain-specific language model for cybersecurity," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2022, pp. 39–56.
[50] M. Levi, Y. Alluouche, D. Ohayon, and A. Puzanov, "Cyberpal. ai: Empowering llms with expert-driven cybersecurity instructions," *arXiv preprint arXiv:2408.09304*, 2024.
[51] A. Graves and A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.
[52] D. K. Sharma and S. Garg, "Ifnd: a benchmark dataset for fake news detection," *Complex & intelligent systems*, vol. 9, no. 3, pp. 2843–2863, 2023.
[53] R. Ganz, B. Kawar, and M. Elad, "Do perceptually aligned gradients imply robustness?" in *International Conference on Machine Learning*. PMLR, 2023, pp. 10 628–10 648.
[54] J. Kim and C. D. Scott, "Robust kernel density estimation," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2529–2565, 2012.
[55] L. Rüschendorf, "The wasserstein distance and approximation theorems," *Probability Theory and Related Fields*, vol. 70, no. 1, pp. 117–129, 1985.
[56] V. M. Panaretos and Y. Zemel, "Statistical aspects of wasserstein distances," *Annual review of statistics and its application*, vol. 6, no. 1, pp. 405–431, 2019.
[57] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.
[58] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.
[59] Z. Wu, F. Tang, M. Zhao, and Y. Li, "Kgv: Integrating large language models with knowledge graphs for cyber threat intelligence credibility assessment," *arXiv preprint arXiv:2408.08088*, 2024.
[60] M. Kanaani, "Triple-r: Automatic reasoning for fact verification using language models," in *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 2024, pp. 16 831–16 840.
[61] GoDaddy team. (2025) WHOIS domain lookup - find website owners - GoDaddy IE. [Online]. Available: https://www.godaddy.com/en/offers/whois-b
[62] Who.is. (2025) WHOIS search, domain name, website, and IP tools - who.is. [Online]. Available: https://who.is/
[63] L. Huynh, T. Nguyen, J. Goh, H. Kim, and J. B. Hong, "Argh! automated rumor generation hub," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, ser. CIKM

'21. New York, NY, USA: Association for Computing Machinery, 2021, p. 3847–3856. [Online]. Available: https://doi.org/10.1145/3459637.3481894

[64] S. Liu, D. Cao, J. Kim, T. Abraham, P. Montague, S. Camtepe, J. Zhang, and Y. Xiang, "{EaTVul}:{ChatGPT-based} evasion attack against software vulnerability detection," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 7357–7374.

[65] H. Peng, Z. Wang, C. Wei, D. Zhao, G. Xu, J. Han, S. Guo, M. Zhong, and S. Ji, "Textjuggler: fooling text classification tasks by generating high-quality adversarial examples," *Knowledge-Based Systems*, vol. 300, p. 112188, 2024.

[66] G. Chang, H. Gao, Z. Yao, and H. Xiong, "Textguise: Adaptive adversarial example attacks on text classification model," *Neurocomputing*, vol. 529, pp. 190–203, 2023.

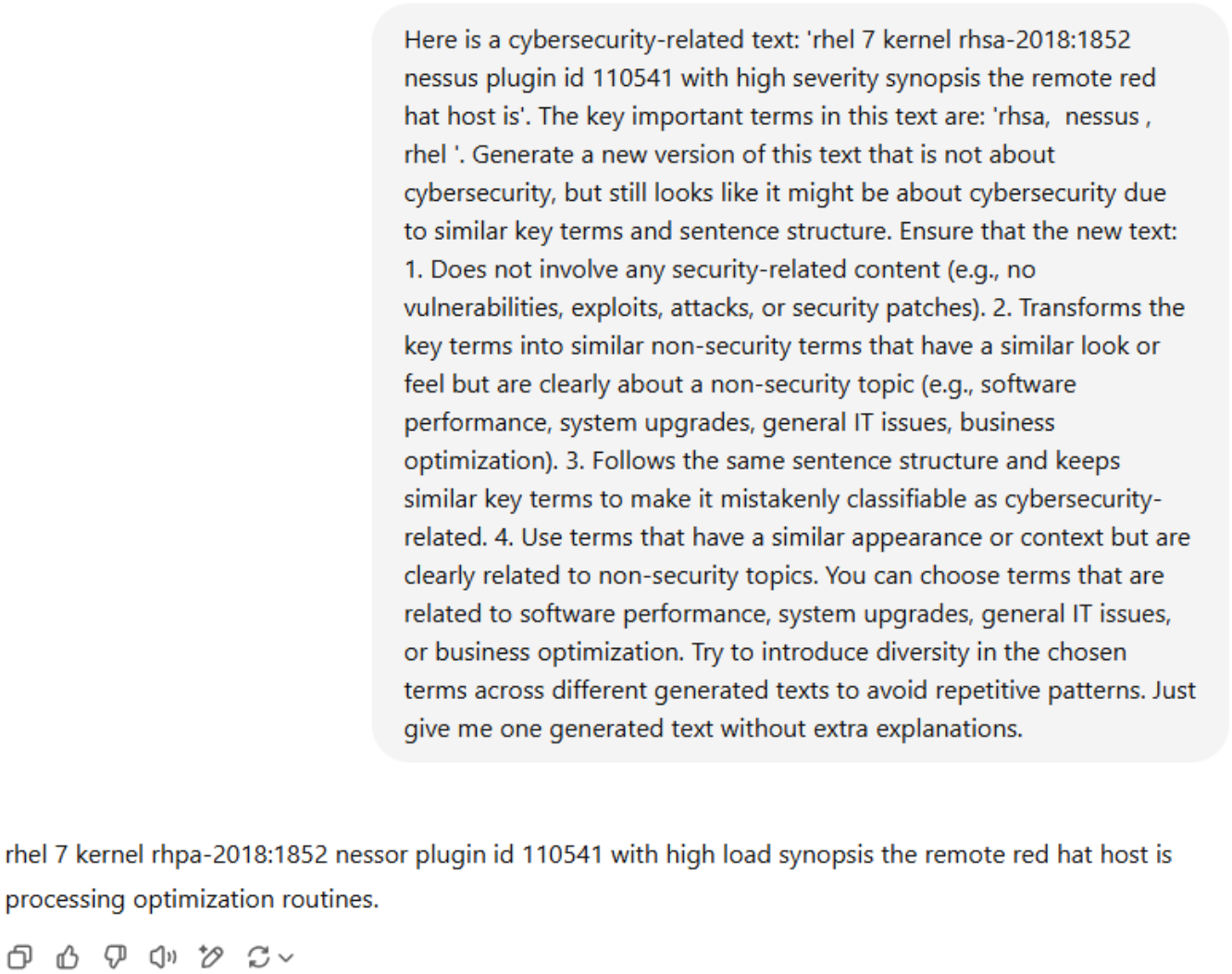

Fig. A1. First prompt: Optimized final prompt to send ChatGPT-4o and its response. Second prompt: Testing ChatGpt as a classifier.

APPENDIX A

The prompt, shown in Figure A1, is carefully structured to guide ChatGPT-4o in generating realistic texts that mimic cybersecurity-related texts while avoiding actual security-related content. The introductory phrase sets the context by presenting a cybersecurity-related example, which ensures the model understands the domain and theme. The next section explicitly highlights key terms such as ”rhsa, nessus, rhel”, directing the model to preserve their stylistic and structural role in the transformation. The instructions then define essential constraints, requiring the generated text to: (1) Avoid security-related content (e.g., vulnerabilities, exploits). (2) Replace key terms with non-security equivalents while maintaining a similar look or structure. 3) Follow the same sentence structure to enhance plausibility as a cybersecurity-related message. To guide these transformations, the prompt provides clear examples (”software performance, system upgrades, general IT issues”), ensuring the output maintains contextual relevance. Additionally, the instruction to introduce term diversity prevents repetitive patterns, further improving the quality of the generated text. By structuring the prompt, each section directly controls the model’s output, ensuring the generated text meets the intended characteristics.

After ChatGPT generates the texts, we open another session to send a prompt asking whether the given sentence is security-related, as shown in Figure A2. This prompt was designed as part of Study [41], demonstrating that ChatGPT can serve as a high-performance classifier capable of accurately distinguishing between security-related and non-security sentences.

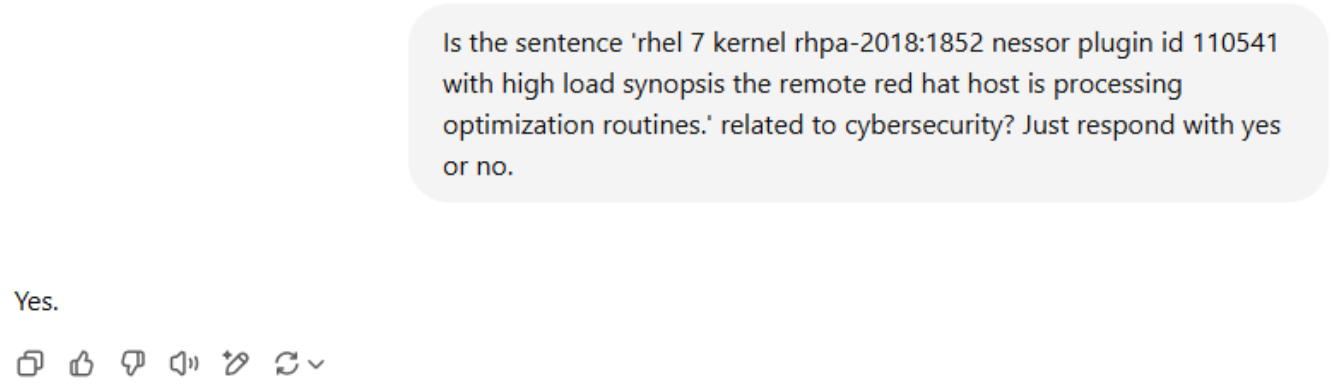

Fig. A2. First prompt: Optimized final prompt to send ChatGPT-4o and its response. Second prompt: Testing ChatGPT as a classifier.