Automated Attack Testflow Extraction from Cyber Threat Report using BERT for Contextual Analysis

Faissal Ahmadou*, Sepehr Ghaffarzadegan*, Boubakr Nour§, Makan Pourzandi§, Mourad Debbabi*, Chadi Assi*
*Concordia University, Canada §Ericsson Security Research, Canada

***Abstract*—In the ever-evolving landscape of cybersecurity, the rapid identification and mitigation of Advanced Persistent Threats (APTs) is crucial. Security practitioners rely on detailed threat reports to understand the tactics, techniques, and procedures (TTPs) employed by attackers. However, manually extracting attack testflows from these reports requires elusive knowledge and is time-consuming and prone to errors. This paper proposes FLOWGUARDIAN, a novel solution leveraging language models (*i.e.,* BERT) and Natural Language Processing (NLP) techniques to automate the extraction of attack testflows from unstructured threat reports. FLOWGUARDIAN systematically analyzes and contextualizes security events, reconstructs attack sequences, and then generates comprehensive testflows. This automated approach not only saves time and reduces human error but also ensures comprehensive coverage and robustness in cybersecurity testing. Empirical validation using public threat reports demonstrates FLOWGUARDIAN's accuracy and efficiency, significantly enhancing the capabilities of security teams in proactive threat hunting and incident response.**

***Index Terms*—Cybersecurity, Security Automation, Testflow Extraction, Advanced Persistent Threats**

I. INTRODUCTION

Advanced Persistent Threats (APTs) [1] pose a complex security challenge due to their stealthy nature. APTs execute multi-stage campaigns, using *Living off the Land* strategies to evade detection and persist. To counteract these threats, threat intelligence companies (*e.g.,* Mandiant, CrowdStrike) and organizations meticulously document their encounters with APTs in threat reports[1]. These reports detail the specific observed campaigns, including the tactics, techniques, and procedures (TTPs) employed by the attackers. For example, Mandiant has extensively documented APT41's activities, detailing a specific campaign exploiting CVE-2019-3396 in [2], a campaign leveraging CVE-2019-19781 in [3], and the attack TTPs alongside the vulnerabilities the threat actor exploited in [4]. Moreover, Google has offered insights into new APT41 variants [5] which utilize Google C2C, Google Sheets, and Google Drive. Each of these reports provides crucial observables from the respective campaigns.

Understanding those reports is crucial to security practitioners, especially for extracting testflow (*e.g.,* security playbook) to verify if the organization is under such an attack. A testflow is a structured set of information that helps practitioners understand the sequence of steps an attacker takes during a cyber attack. Security practitioners start by thoroughly reading and understanding the report to grasp the scope, nature of the threat, affected systems, and exploitation details. After identifying the attack steps and objectives, they extract critical technical details (*e.g.,* specific software versions, configurations, and necessary conditions for the threat) which will guide the development of a detailed testflow. The process is inherently complex and demands not only advanced technical knowledge and analytical skills which are elusive, but also a significant time investment. The process becomes more complicated and cumbersome when multiple threat reports are released leading to further efforts to generate a comprehensive testflow. Therefore, automating testflow extraction from threat reports helps in ensuring consistency, accuracy, and efficiency. By implementing automation, security practitioners can parse through extensive and diverse reports coming from different sources simultaneously, identify common vulnerabilities that might lead to the attack, and consolidate similar testflow. This not only saves time and reduces the effort required by security teams but also minimizes human error and enhances the thoroughness of the testing process.

Generated testflows serve as essential resources for Security Operation Center (SOC) teams: (i) they enable proactive threat hunting by providing a framework to systematically search for signs of APTs (*e.g.,* Indicators of Compromise – IoCs) that match the behaviors described in the threat reports; and (ii) the testflows facilitate a more informed and efficient incident response. When an attack is detected, SOC teams can use the specific testflows relevant to the observed attack to quickly understand the nature of the threat and trace its origin and impact.

Unlike existing studies that concentrate on deriving attack techniques from threat reports (*e.g.,* [6]) or from CVEs (*e.g.,* [7]), our work primarily targets the extraction of attack testflows from previously observed attack campaigns documented in threat reports. This approach aims to enrich the cybersecurity playbook. Thus, in this paper, we make the following contributions: (1) we introduce a novel solution, FLOWGUARDIAN, tailored specifically for extracting testflows from threat reports (unstructured documents). Our solution is the first that uses language models to perform security contextual analysis and sequence analysis to systematically break down the attack into actionable steps and thus generate an accurate attack testflow. (2) we design an automated framework that integrates entity extraction and, sequence analysis,

[1]Throughout this paper, the term *threat report* is used broadly to describe any document that details a threat (*e.g.,* cyber threat reports, security/assessment reports, security troubleshooting reports).

enabling efficient extraction of testflows from raw text without manual intervention. (3) we validate our solution through empirical studies using public APT threat reports. The obtained results, reviewed and confirmed by security practitioners, show that FLOWGUARDIAN is able to extract accurate testflows and enrich the cybersecurity playbook.

II. MOTIVATION AND PROBLEM STATEMENT

Motivation: Whenever a new threat report is released, security analysts have a critical responsibility to review the document and assess whether their organization is subject to the reported attack. This process underscores the importance of testflow extraction from security reports. To effectively, yet proactively, secure their networks/systems, they must thoughtfully identify the attack infrastructure outlined in the report and meticulously construct the attack sequence and steps. This enables SOC teams to test if their organization is vulnerable to the documented attack steps. The ability to quickly and accurately extract attack testflow is crucial for proactive defense measures and ensuring that security postures are capable of withstanding current and emergent threats. This not only helps in identifying potential breaches but also assists in fortifying the organization's defenses against similar future attacks.

Problem Statement: The extraction of attack testflow from report is a critical but time-intensive task requiring extensive cybersecurity knowledge and analytical skills. Currently, the process is predominantly manual [8], leading to inefficiencies and potential for oversight. This is particularly problematic when considering the need for a comprehensive understanding of different APT campaigns, where integrating insights from multiple reports is essential. Fig. 1 presents an excerpt of APT41 reports from Mandiant published in 2024 [2], where security experts need to identify specific elements involved in the attacks.

Automating the extraction of testflow from diverse and unstructured threat reports would dramatically streamline this process, reduce time and resource expenditure, and increase the accuracy and comprehensiveness of the testflow. APT attacks are very sophisticated, and extracting the right testflow from the reports could be cumbersome and error-prone [1], resulting in incomplete, incorrect, or misinterpreted information. The process requires a lot of time and human resources, which can be costly and inefficient in the long run.

To address these concerns, it is important to consider how to automate the extraction of attack steps and generate testflows corresponding to attack vectors to improve the effectiveness of the incident response. Our approach does not focus on high-level testflow based on techniques (*i.e.,* TTPs) instead, we prioritize low-level testflow that captures the granular details of an attack to provide more comprehensive information and a more detailed testflow.

> Starting on January 20, 2020, APT41 used the IP address 66.42.98[.]220 to attempt exploits of Citrix Application Delivery Controller (ADC) and Citrix Gateway devices with CVE-2019-19781 (published December 17, 2019).
> The initial CVE-2019-19781 exploitation activity on January 20 and January 21, 2020, involved execution of the command 'file /bin/pwd', which may have achieved two objectives for APT41. First, it would confirm whether the system was vulnerable and the mitigation wasn't applied. Second, it may return architecture-related information that would be required knowledge for APT41 to successfully deploy a backdoor in a follow-up step.

Figure 1: Mandiant's APT41 threat report [2].

III. RELATED WORK

Cyber Threat Intelligence (CTI): CTI plays a crucial role in cybersecurity by enabling organizations to anticipate, prevent, and respond to cyber threats through structured intelligence sharing. Recent advancements focus on dynamic intelligence extraction, mapping attack patterns to frameworks (*e.g.,* MITRE ATT&CK), and leveraging machine learning for automation. However, existing CTI frameworks primarily address knowledge representation rather than attack testflow generation. While prior work (*e.g.,* [9]) has explored CTI sharing challenges, frameworks for structured intelligence extraction remain a gap in integrating language models for real-time, automated cybersecurity response. While threat reports compile insights into detailed narratives tailored for security practitioners, the increasing complexity and volume of these reports have outpaced the capabilities of traditional approaches, requiring the use of new efficient methods.

Language Models in Cybersecurity: LMs (*e.g.,* BERT [10], RoBERTa [11], MUM [12]) have been widely explored in cybersecurity for tasks such as intrusion detection, malware analysis, phishing detection, and vulnerability assessment [13]. While these studies demonstrate the potential of language models in various security applications, they primarily focus on general cybersecurity challenges rather than the automation of attack testflow extraction from CTI reports. Despite advances in generative AI for cybersecurity [14], there remains a gap in using them for attack testflow generation.

Attack Testflow Extraction: A testflow represents the sequence of attack steps, allowing organizations to assess vulnerabilities and validate security measures [15]. Several approaches exist for extracting attack testflows such as, information extraction [16], ontology-based Methods [17], and Machine Learning-based extraction [18]. Building on these foundational approaches, several systems have been proposed to further automate this process. EXTRACTOR [19] formalizes attack steps into evidence graphs but relies heavily on NLP accuracy, limiting its robustness for unstructured CTI. IC-SECURE [20] facilitates playbook generation but does not generalize to real-time attack testflow extraction. Similarly, THREATRAPTOR [21] and TTPDrill [22] focus on IoC extraction and ATT&CK mapping, yet their rule-based techniques lack adaptability to evolving threats, while LADDER [9] aims to extract attack patterns from CTI and map them to the MITRE ATT&CK framework. Despite these advancements, automated attack testflow extraction from unstructured reports remains an underexplored area. Current methods focus on structured knowledge representation rather than real-time adaptive testflow generation. Addressing this gap would enhance security operations by transforming unstructured knowledge from CTI into actionable intelligence for proactive defense.

Our approach, FLOWGUARDIAN, systematically combines domain-trained language models (*e.g.,* BERT) for contextual analysis with sequence analysis to extract structured testflows directly from unstructured threat reports. FLOWGUARDIAN

focuses not only on identifying entities and actions but also on reconstructing their order and contextual meaning to generate relevant testflows. This allows FLOWGUARDIAN to generate testflows with semantic coherence and operational relevance.

IV. AUTOMATED ATTACK TESTFLOW EXTRACTION

A. System Overview

Overview: FLOWGUARDIAN, depicted in Fig. 2, has four main steps: (1) text pre-processing: involve standardizing the text to be ready for testflow extraction. (2) contextual analysis: interpret its context using semantics and word relationships. Language model (*e.g.,* BERT) is used to extract relevant information, detect subtexts, and identify key elements such as named entities and action verbs that are crucial for generating relevant testflows. (3) sequence analysis: the identified entities and their context are analyzed to reconstruct the sequence and logic of the attack steps, providing a clearer understanding of the attack pattern and assisting in the testflow generation. (4) The final step is testflow generation. Using insights from the contextual analysis, we automatically build testflows. These are generated using LM guided by rules trained on the analyzed data, resulting in a concrete list of testflows for the APT attack sequences listed in the report. To improve this process, we further include a testflow validator to filter out invalid or redundant testflows and refine those that can be adjusted to meet validity criteria.

Novelty: The core solution of FLOWGUARDIAN involves three novel steps: (1) *Threat Contextualization*: refers to analyzing and understanding security events, incidents, or threats in the report. To define the security context, we include how a particular vulnerability or threat affects an organization in terms of its network configurations, software usage, business processes, and protective measures. (2) *Attack Steps Analysis*: refers to the systematic examination of all activities performed by the threat actor to achieve their objectives. The goal is to identify the various stages through which attackers move from the initial point of breaching security measures to their ultimate goal, such as stealing data, compromising systems, or achieving persistent presence. This method goes beyond existing approaches by offering a complete, contextualized, and actionable understanding of an adversary's movements. (3) *Automated Testflow Extraction*: refers to the process of using extracted information in combination with various techniques to generate testflows. Current methods often generate generic testflow, lacking relevance to specific cybersecurity scenarios. FLOWGUARDIAN uses cybersecurity-specific data and techniques to ensure that the generated testflows are directly aligned with the described attack.

Design Challenges: We present below the main challenges and how we address each. (i) *Complexity of Cybersecurity Language*: the complexity involved in using specific terms related to cyber protection. In addition, this language often requires deep knowledge of technical concepts, making it difficult for NLP efforts. (ii) *Data Scarcity and Low-Quality reports*: insufficient relevant data poses a significant challenge in training machine learning models and conducting meaningful analysis. Additionally, reports with low-quality, incomplete, or complex language further complicate the analysis process. Standard language models, struggle to handle such reports, failing to generate meaningful outputs. (iii) *Extraction of Relevant Information*: general purpose NLP models, are designed for broad language tasks. However, they lack specialized capabilities for cybersecurity contexts. In our work, we use LM and NLP to perform contextual analysis to extract cybersecurity entities and actions. FLOWGUARDIAN integrates spaCy and fine-tunes them with BERT to enable accurate recognition of all the relevant entities in the cybersecurity domain. We also use sequence analysis to understand the context and relationships between entities, which helps in extracting testflows. Finally, we use semantic mapping of these sequences into a related test.

B. Text Processing

This phase involves several NLP techniques [23] to refine raw, unstructured data such as: a) text cleaning, which prepares raw text data for further processing and analysis, b) normalization, which converts data into a consistent and standard format, simplifies analysis and c) segmentation, which simplifies and enhances the effectiveness of data processing. The result of the text processing is a pre-processed text, which has been both cleaned and structured. This prepared text serves as the input for the contextual analysis, enabling more sophisticated analytical processes.

C. Contextual Analysis

Extracting and interpreting the meaning of text requires considering its cybersecurity context, whether within a sentence, paragraph, or broader document. Contextual analysis is vital for understanding both the content and implications of the text. FLOWGUARDIAN employs LM to extract key threat-related entities and action verbs within the threat reports.

FLOWGUARDIAN NER Approach: Traditional NER often struggles to recognize and extract emerging cybersecurity entities [24] such as threat actors, malware names, attack types, and techniques, which are essential for generating meaningful testflows. FLOWGUARDIAN addresses this limitation by: (i) Pre-training models using general datasets like spacy. (ii) Fine-tuning BERT on our custom cybersecurity dataset, enabling the extraction of cybersecurity-specific entities.

LM Training and Fine-Tuning: To train models that mimic the expertise of security experts, a security dataset is required for LM training. FLOWGUARDIAN leverages BERT [10] to analyze and extract security information from threat reports, enabling precise identification of hidden threats. BERT's advanced contextual comprehension allows FLOWGUARDIAN to detect subtle linguistic nuances in cyber threat reports. By utilizing BERT's pre-trained language representations, we enhance the accuracy and efficiency of our threat intelligence and analysis.

For cybersecurity applications, BERT is fine-tuned on labeled NER datasets to extract domain-specific entities from CTI reports with high precision. Unlike large generative models (*e.g.,* ChatGPT, LLaMA), which are prone to hal-

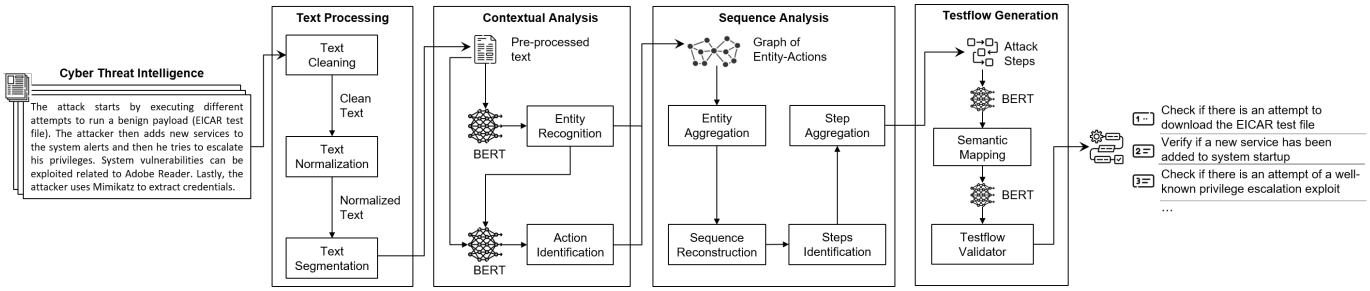


Figure 2: High-level overview of FLOWGUARDIAN.

lucinations, producing plausible but incorrect or unverifiable information. BERT offers greater stability and reliability. Since BERT is designed for discriminative tasks rather than text generation, it avoids the unpredictability of generative LLMs and produces consistent outputs. This makes BERT a strong and justified choice for structured information extraction in the cybersecurity context. Additionally, FLOWGUARDIAN remains compatible with other LM-based models, ensuring adaptability across various industries and applications.

During the fine-tuning (see Fig. 3), we used SpaCy, an NLP model specifically designed for named entity recognition (NER). However, despite being tailored for NER tasks, the model struggled to accurately identify certain entities. To overcome this, we fine-tuned the model with a customized dataset (see subsection V-B), significantly enhancing our entity recognition accuracy. The customized dataset refines cybersecurity entity categorization into eight key groups to better capture underrepresented concepts, such as: (i) `TEC` (Technique): methods or strategies employed by threat actors to exploit vulnerabilities or execute malicious actions, *e.g.,* `phishing`. (ii) `TAC` (Tactic): attack strategies from MITRE ATT&CK, *e.g., initial access*. (iii) `THA` (Threat Actor): known adversarial groups like *APT41*. Additionally, we also trained the model to map attack steps into the testflow using a set of predefined mapping rules (see subsection IV-E). The objective is to efficiently correlate the analyzed text with specific attack descriptions, leveraging the accurately identified entities from the previous step.

Cybersecurity Action Identification: This process focuses on finding action verbs within sentences, and helps uncover behavior patterns, automate analysis, and understand the relationships between entities. Action identification involves pinpointing verbs and verb phrases, such as *"downloading file", "adding new services", "exploiting vulnerabilities"*, and understanding how these actions related to entities in the context – which is very challenging using traditional NLP. Leveraging LMs for action verb extraction enhances the analysis of threat reports by identifying relevant actions based on the used language and learned patterns.

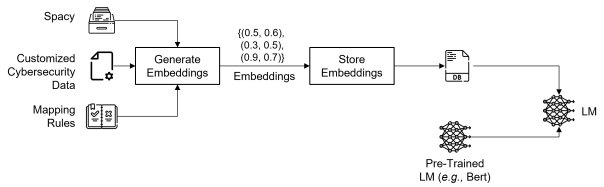


Figure 3: Overview of steps to train/fine-tune the BERT model.

D. Sequence Analysis

This involves examining text to extract valuable information. A sequence is an ordered list of elements or events following a specific pattern, whether temporal or sequential. Analyzing the identified NER entities and action verbs within their context helps reconstruct the sequence and logic of attack steps. Advanced techniques focus on entity aggregation, sequence reconstruction, step identification, and step aggregation, each playing a distinct role in interpreting structured data.

Entity Aggregation: Entity aggregation combines related entities based on their occurrence and context within the text by grouping them into a single aggregated entity. This process reduces redundancy, enhances analysis efficiency, and simplifies the management of entities. The primary goal is to minimize the number of entities for easier analysis. Using the extracted NER and identified action verbs, we can aggregate entities based on their types and associated actions. For example, "*The attacker*" and the pronoun "*he*" are recognized as the same entity, despite the shift in reference.

Sequence Reconstruction: Sequence reconstruction involves assembling segments into a coherent order in order to arrange attack steps based on their narrative flow or temporal cues like "*first*", "*then*", or "*finally*". The narrative flow from the text provides the order of operations that the attacker followed, and we map each action to an appropriate phase and tactic based on MITRE ATT&CK framework. To achieve sequence reconstruction, we rely on a natural process that mimics how humans understand the flow of events by interpreting their relationships and context.

Step Identification: This step involves detecting and parsing distinct actions or events within a sequence, highlighting each step's characteristics and role. This step clarifies the sequence by identifying relationships between steps, such as how initial access through a *phishing email* leads to *malware installation*. This process is essential for tracing the flow of activities, enabling a deeper understanding of attack schemes, patterns, or workflows within complex sequences. To accomplish this task, FLOWGUARDIAN firstly, analyzes the relationships between the identified NER and the corresponding action verbs, as well as labeled with step identifiers (*e.g., Step 1*, *Step 2*); and then orders the steps using natural step and logical flow based on temporal sequence indicators such as "*First*", "*Next*", "*Then*",

and "*Finally*". In this process, the relationships between NER and actions guide the step ordering. This relationship is what drives the logical flow of attack, helping to determine which step comes before or after another.

Step Aggregation: This step is similar to entity aggregation but focuses on action verbs. Texts may describe similar steps differently or repeat them, so consolidating these avoids redundancy and emphasizes unique attack vectors. One key method is combining identified steps into broader stages or phases, making the sequence more concise and easier to understand. Grouping smaller actions into strategic categories simplifies the attack or testflow analysis. For this, we use a clustering algorithm to group related steps based on feature similarity.

In doing so, we follow the following process: (i) *Grouping*: When multiple steps involve the same entity, they can be grouped into one single aggregated step. (ii) *Labeling*: we label each aggregated step with a phase name according to MITRE ATT&CK framework to each aggregated step. Rule-based matching (defined based on tactics in MITRE ATT&CK) is used to match specific actions or keywords to the corresponding tactic. (iii) *Validation*: we ensure that the steps are logically grouped and represent meaningful phases of the attack. The validation consists of comparing the steps and aggregated phases against the MITRE tactics and techniques to ensure that all relevant stages of the attack are covered.

E. Testflow Generation

Testflow generation involves creating detailed tests or sequences to verify whether the described attack exists within a system, ensuring security measures are effective.

Semantic Mapping: To enhance semantic understanding, context-based action verbs are linked directly to the identified entities. We created a set of labeling rules for generating testflow, detailing how entities and actions should be structured. These rules are used to train the BERT model, enabling it to automatically generate detailed, contextually appropriate testflows.

Mapping rules: BERT is meticulously trained using a set of mapping rules that convert detailed security incident descriptions into specific, actionable testflow instructions (see Fig. 3). These rules help verify the presence and impact of various cybersecurity threats. Each rule is linked to a particular security issue (*e.g.,* malware, ransomware) and defines the corresponding action (*e.g.,* run, encrypt) along with a related test command. These mapping rules are integral to the model's ability to effectively verify the presence and assess the impact of various cybersecurity threats within a system. Each rule is carefully linked to a specific security issue, such as malware infections, ransomware attacks, or unauthorized access attempts. The rules define the corresponding actions required to address these issues, such as executing a script, encrypting data, or initiating a network scan. Furthermore, these rules are not arbitrarily chosen; they are carefully selected and designed based on the core primitives of testflow, ensuring that they align with fundamental security operations. By grounding the rules in these core primitives, we ensure that the testflows generated by the LM model are not only relevant but also comprehensive in covering the essential aspects of testflow generation.

Testflow Validator: To improve the quality and reliability of our testflow generation process, we introduce a tesflow validator component. This component serves two main purposes: (a) *filtering out invalid tests:* ensure that testflow generated by FLOWGUARDIAN are valid and meet the required criteria, and (b) *eliminating redundancy:* remove duplicate or redundant testflow to streamline the output. The output of this component is manually cross-validated to confirm the effectiveness of the filtering and refinement process. During this manual iteration, any remaining redundant or invalid testflows are identified and removed.

Fig. 4 shows examples of testflows generated by FLOWGUARDIAN. The generated testflows align with standard security operations and highlight how FLOWGUARDIAN effectively proposes relevant testflow for the used threat report.

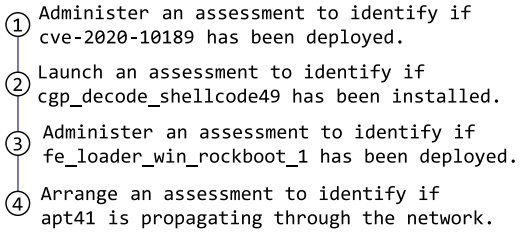


Figure 4: Example of testflows generated by FLOWGUARDIAN.

V. IMPLEMENTATION

A. Environment

We implemented FLOWGUARDIAN using Python 3 programming language. Our experiments were conducted on Intel Xeon E312xx, 6 vCPU @ 2.69Ghz, with 32GB RAM running Ubuntu 20.04. For linguistic analysis, we employed the `spaCy`[2] library, a comprehensive tool for natural language processing in Python, to extract action verbs. Specifically, we utilized the `en_core_web_sm` model.

B. Datasets

Threat Reports: We use threat reports from the APTNotes repository[3], which gathers different threat intelligence from vendors like Mandiant and TrendMicro. Given the lack of comprehensive testflows in existing reports and the difficulty of manual extraction, we focus solely on APT41 for the experiment discussion, a sophisticated APT with diverse techniques and multiple public reports. Indeed, we use five different APT41 reports published by different threat intelligence companies describing different APT41 campaigns. For example, the Mandiant #1 [4] is 68 pages long and covers backdoors, reconnaissance, and credential theft in some detail. This detail allows for the extraction of highly detailed and valid testflows, reducing redundancy. Other reports, such as Mandiant #2 [3] and Mandiant #4 [25] are shorter 9 pages each and often

[2]spaCy: https://spacy.io/

[3]APTNotes repository: https://github.com/aptnotes/data/

do not have the level of granularity required to produce comprehensive testflows. Consequently, the extracted testflows may be more general and tend to be redundant because of the small amount of usable detail.

Quality of threat reports: The quality and structure of the reports play a critical role in the reliability of the testflows that are extracted from them. BertScore [26] is a metric that uses deep contextual embeddings from pre-trained models like BERT to evaluate semantic similarity. In contrast to traditional metrics that focus on syntactic matches, BERTScore captures nuanced relationships between words and phrases, taking into account synonyms, paraphrases, and contextual shifts. This makes it an ideal tool for assessing the semantic similarity of texts. We use BERTScore to assess the semantic similarity of reports processed by our solution, ensuring meaningful and coherent outputs. Additionally, we compare testflows generated by our system with those manually created by practitioners to validate their alignment with domain expertise.

Fig. 5 shows the BertScore of the used threat report. We can see that reports Mandiant #2 and Mandiant #4 have the highest similarity (0.84), indicating shared patterns in their testflows. In contrast, Mandiant #3 exhibits the lowest similarity, particularly with Mandiant #1 and Mandiant #4, suggesting distinct attack scenarios (see Section VI-D for further analysis). Indeed, the report quality significantly impacts generated testflows. Well-structured, detailed reports improve accuracy and relevance, while similar reports yield consistent testflows, reinforcing the importance of high-quality input data.

Testflow Primitives: In this work, we focused on the core functions of attack steps and derived 30 core cybersecurity primitives through NERs. Using specific mapping rules, NER entities were linked to action verbs in security testing scenarios. We also developed 44 mapping rules to structure testflows based on the identified NER and action verbs. These fine-tuned primitives target various security issues, enhancing the system's ability to identify and mitigate vulnerabilities effectively.

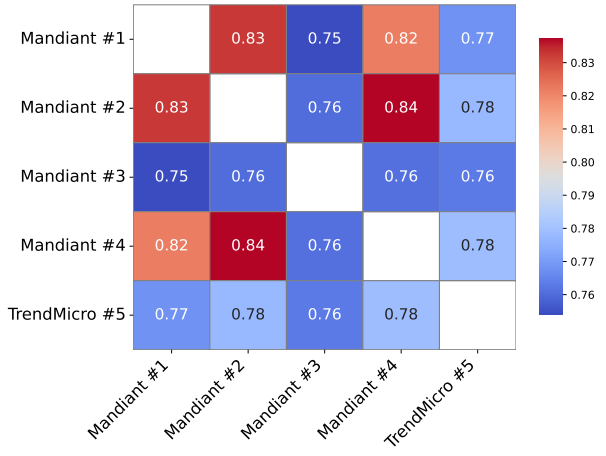


Figure 5: Semantic Similarity between reports using BertScore [26].

C. Benchmarked Models

In the absence of established tools for generating testflows from threat reports, we introduce four distinct models to benchmark and compare their performance against FLOWGUARDIAN's results. These models leverage the capabilities of `ChatGPT-4o-mini`[4] and `Llama3-70b`[5] as foundational components:

- Off-the-shelf Models: `ChatGPT-off-the-shelf` and `Llama-off-the-shelf` utilize the raw power of the underlying LLMs without any modifications. In these configurations, the threat report is directly fed into the LLM, and the outputted testflows are analyzed to gauge the baseline performance.
- Off-the-shelf Models with Partial FLOWGUARDIAN's Pipeline: we integrated the pipeline of FLOWGUARDIAN (PFG), as depicted in Fig. 2, into Off-the-shelf models `ChatGPT-with-PFG` and `Llama-with-PFG` to demonstrate the impact of FLOWGUARDIAN's architecture on testflow generation, providing a clear evaluation of our system's design efficacy.

D. Ground truth data

In the absence of a public testflow dataset, we enlisted the expertise of two security practitioners to manually create testflows for APT41's threat report. Table I shows the number of manually generated testflow (*i.e.,* ground truth) of both security practitioners per each report. We also verified the combined testflows to eliminate any redundant or duplicate testflows, and then applied a cross-validation by both security practitioners to ensure the accuracy of our results. The unique testflows are used as a ground truth. To simulate real-world scenarios, we instructed one security practitioner to focus on creating high-level and generalized testflows, while the other security practitioner concentrated on developing more granular and specific testflows. This dual approach mirrors the diversity in practices among security practitioners. Seasoned practitioners often prioritize fewer and more targeted tests, whereas less experienced practitioners may employ a larger number of tests to ensure comprehensive coverage.

Table I: Summary manual generated testflow and their characteristics (*i.e.,* redundant and unique).

Threat Report	Mandiant #1	Mandiant #2	Mandiant #3	Mandiant #4	TrendMicro #5
Manual Testflows	27	16	42	48	33
Redundant Testflow	3	5	9	6	8
Unique Testflow	24	11	33	42	25

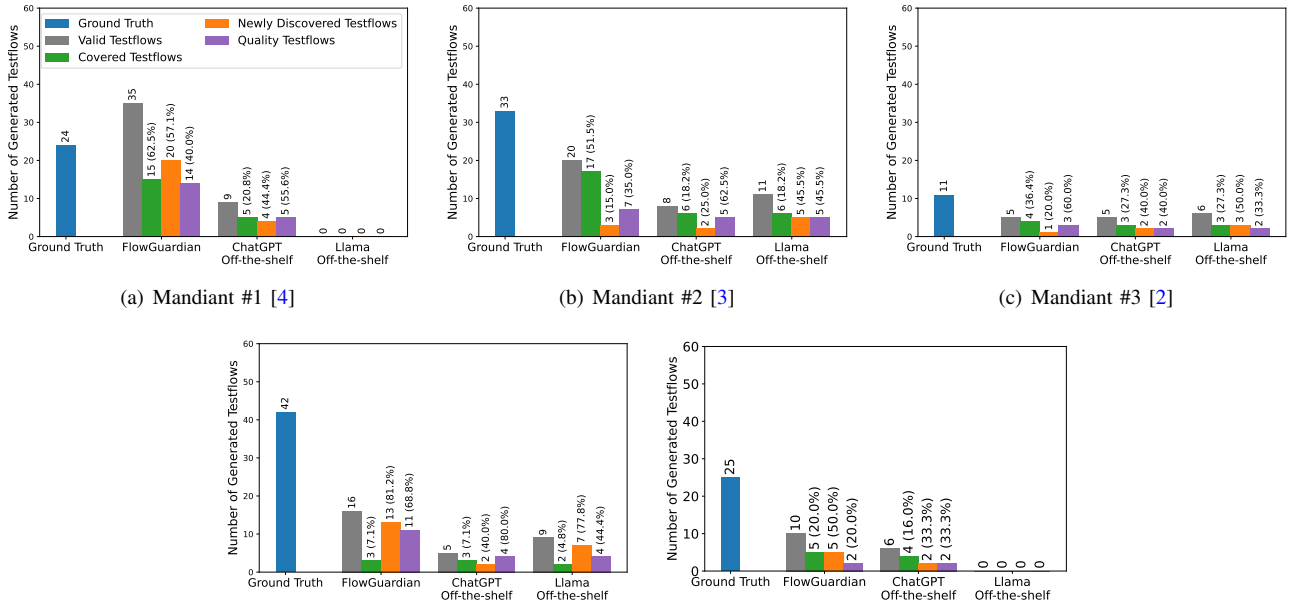
VI. EVALUATION

A. Evaluated Metrics

To evaluate the testflows generated either by an automated solution or a security practitioner, we introduce four metrics: *testflow coverage*, *testflow validity*, *newly discovered testflow*, and *quality testflow*.

Testflow Coverage: The testflow coverage is defined as the proportion of manually generated testflows that are successfully captured by the testflows produced by a given model.

[4]GPT-4o mini: https://platform.openai.com/docs/models/gpt-4o-mini
[5]Llama3: https://ai.meta.com/blog/meta-llama-3/



(a) Mandiant #1 [4] (b) Mandiant #2 [3] (c) Mandiant #3 [2]

(d) Mandiant #4 [25] (e) TrendMicro #5 [27]

Figure 6: Comparison of testflows generated by FLOWGUARDIAN and Off-the-Shelf Models across five APT41 reports.

A valid testflow exact match or through semantic equivalence with the manually generated testflow. A semantic equivalence means that more than one testflow will collectively match one testflow.

Testflow Validity: The testflow validity refers to testflow that aligns with the practical requirements and expectations of security practitioners, and satisfies both of the following conditions: (i) *relevance*: testflow is relevant and makes sense from the security perspective, reflecting real-world scenarios; (ii) *applicability*: testflow is applicable in the context of testing APT in question.

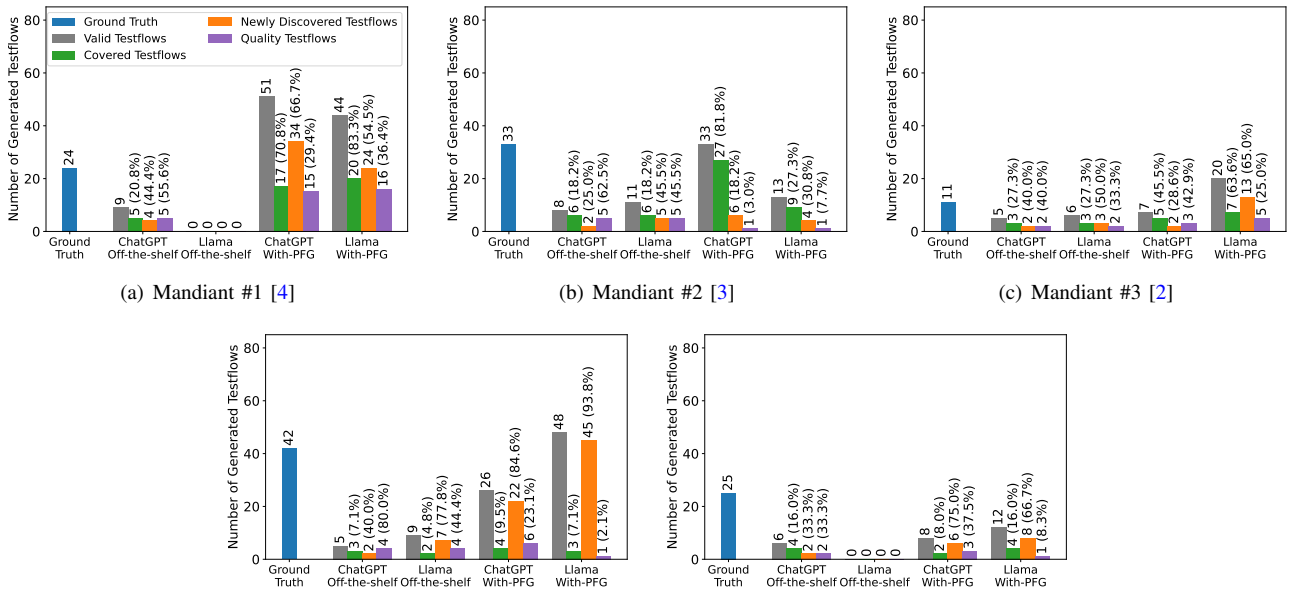
Newly Discovered Testflow: The newly discovered testflow refers to a testflow generated by an automated solution that is valid yet not part of the testflows originally created by practitioners. These testflows are cross-validated by practitioners to confirm their validity and ensure they represent new, previously uncovered tests that were not addressed in the practitioners' original test set.

Quality Testflow: A quality testflow refers to a testflow that is highly specific to the APT41 as described in the report. Among the testflow generated by the automated solution, some may be generic (applicable to various scenarios), while others are directly linked to the APT41 TTPs. The focus is on capturing the unique behaviors of the threat actor to ensure the testflow relevance.

B. Performance of FLOWGUARDIAN *vs Off-the-Shelf Models*

Fig. 6 shows the comparison results of the testflows generated by FLOWGUARDIAN and Off-the-Shelf models across the five APT41 reports. We can see that FLOWGUARDIAN generates more than twice the number of valid testflows compared to `ChatGPT-off-the-shelf` and `Llama-off-the-shelf` across all reports. This demonstrates FLOWGUARDIAN's superior ability to produce valid testflows and underscores the significance of fine-tuning FLOWGUARDIAN with a customized NER model. In contrast, general-purpose LLMs, such as `ChatGPT-off-the-shelf` and `Llama-off-the-shelf`, lack accuracy in this specialized context. More specifically, FLOWGUARDIAN consistently achieves higher coverage rates compared to off-the-shelf models. For example, in Mandiant #1 and Mandiant #2, FLOWGUARDIAN achieves nearly three times the coverage of `ChatGPT-off-the-shelf` and `Llama-off-the-shelf`. However, in Mandiant #3, Mandiant #4, and TrendMicro #5, the coverage rates are almost identical across all models. These results highlight that FLOWGUARDIAN not only generates the highest number of testflows but also aligns closely with the practices of manual practitioners. While FLOWGUARDIAN covers the majority of testflows, it also demonstrates the ability to discover new valid testflows. FLOWGUARDIAN identifies new testflows nearly twice as often as the off-the-shelf models for most reports, except for Mandiant #2, where `Llama-off-the-shelf` discovers more new testflows than both FLOWGUARDIAN and `ChatGPT-off-the-shelf`. This exception is justified by the fact that, for this particular report, FLOWGUARDIAN generated a higher number of covered tests, resulting in fewer opportunities for discovering new tests.

Although FLOWGUARDIAN excels in generating valid, covered, and newly discovered testflows, its primary strength lies in producing high-quality testflows. FLOWGUARDIAN consistently generates almost twice as many high-quality testflows, except the TrendMicro #5 report. Additionally, we observed that `ChatGPT-off -the-shelf` tends to generate testflows for which it has high confidence. Upon



(a) Mandiant #1 [4] (b) Mandiant #2 [3] (c) Mandiant #3 [2]

(d) Mandiant #4 [25] (e) TrendMicro #5 [27]

Figure 7: Comparison of testflows generated by Off-the-Shelf models and FLOWGUARDIAN's across five APT41 reports.

manually reviewing its output, many of the testflows were found to be nearly identical to the original text from the reports. While this approach ensures accuracy for individual testflows, it results in the omission of numerous testflows typically expected from an APT report. Consequently, the model produces very few testflows, rendering it less comprehensive for sophisticated attacks such as APT41[6]. On the other hand, `Llama-off-the-shelf` generates more testflows compared to `ChatGPT-off-the-shelf`. However, as shown in Figs. 6(a) and 6(e), there are instances where `Llama-off-the-shelf` fails to produce any results. This limitation arises from the model's inability to effectively process certain reports, particularly those with extensive size or complexity. Even after splitting the input data into smaller sections, the model struggled to generate testflows for these challenging reports.

C. FLOWGUARDIAN's Pipeline with Off-the-Shelf Models

This experiment demonstrates the added value of integrating our FLOWGUARDIAN pipeline into off-the-shelf models. As shown in Fig. 7, the integration of our pipeline significantly enhances the capability of these models to generate a greater number of valid testflows. Specifically, the number of valid testflows produced by `ChatGPT-off-the-shelf` and `Llama-off-the-shelf` increased by at least fourfold in Mandiant #1, #2, and #4. Moreover, even in Mandiant #3 and TrendMicro #5, where the off-the-shelf models struggled, the integration of FLOWGUARDIAN still led to a significant improvement. This highlights the critical role of FLOWGUARDIAN in resolving logical inconsistencies and enriching the context with domain-specific entities.

In report Mandiant #1, the number of covered testflows generated by `ChatGPT-off-the-shelf` increases from 5 to 17, and in report Mandiant #2, from 6 to 27 – more than quadrupling the original numbers. Similarly, for `Llama-off-the-shelf`, coverage rises from 6 to 9. Reports Mandiant #3, Mandiant #4, and TrendMicro #5 show moderate but consistent improvements, further demonstrating the significant impact of FLOWGUARDIAN on these models. Similarly, the number of newly discovered testflows increased significantly, particularly for `ChatGPT-off-the-shelf`, which saw a large rise, and for `Llama-off-the-shelf`, which experienced a moderate increase. We also observed that the total number of high-quality testflows tripled for both `ChatGPT-off-the-shelf` and `Llama-off-the-shelf` in report Mandiant #1, indicating improvement in the quality of testflows when FLOWGUARDIAN was integrated.

D. Similarity Analysis between Generated Testflows

Fig. 8 shows the semantic similarity scores between the testflows generated by FLOWGUARDIAN and the ground truth testflows across five reports. FLOWGUARDIAN demonstrates consistent semantic similarity across all reports, ranging from 0.82 to 0.87. This consistency highlights FLOWGUARDIAN's effectiveness in generating testflows that are semantically aligned with the ground truth, regardless of the report. `ChatGPT-with-PFG` also shows consistent performance, with minor variations across the reports. However, in Mandiant #2, a slight drop in the similarity score is observed. This decline may be attributed to the report's characteristics, such as ambiguity, less structure, or reliance on images. Conversely, the same figure shows that `LLaMA-with-PFG` exhibits consistent performance across the reports, albeit with slightly higher variability compared to `ChatGPT-with-PFG`. Similar to `ChatGPT-with-PFG`, `LLaMA-with-PFG` appears

[6]APT41: https://attack.mitre.org/groups/G0096/

to perform better with textual and structured reports.

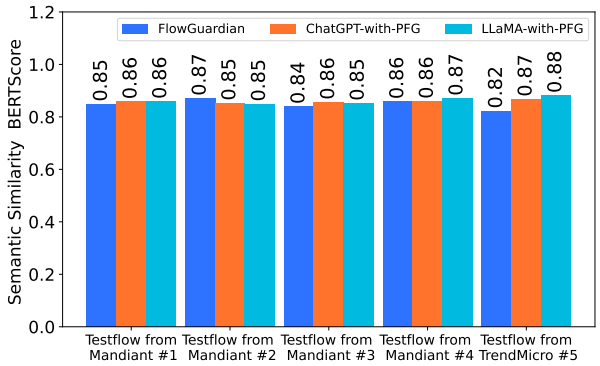


Figure 8: Semantic similarity of testflows generated by FLOWGUARDIAN and Off-the-Shelf models vs ground truth across five APT reports.

E. Execution Time

Manually extracting testflows involves thoroughly reading and analyzing the entire report, a process that typically takes several hours. For example, longer reports like Mandiant #1 and TrendMicro #5 are executed at 493.13 sec and 306.27 sec, respectively. In comparison to security practitioners, who take several hours/days, FLOWGUARDIAN's ability to handle such reports in under 10 minutes is a remarkable improvement. Shorter reports like Mandiant #2 and Mandiant #4 are executed at 35.9 sec and 82.29 sec, respectively. FLOWGUARDIAN processes them in less than two minutes, showing that FLOWGUARDIAN scales well across different report sizes and lengths. Medium reports that are neither too long nor too short show even good execution times. For example, in report Mandiant #3, FLOWGUARDIAN can process the report in just 70.87 sec, which is far quicker than manual practitioners. Overall, FLOWGUARDIAN demonstrates exceptional efficiency by reducing the analysis time from several hours to just minutes (average times 197.692 sec), depending on report size. By drastically reducing processing time, FLOWGUARDIAN enables security teams to focus on higher-value analytical tasks rather than spending time on the labor-intensive creation of testflows, significantly enhancing operational efficiency.

VII. CONCLUSION & FUTURE WORK

This paper introduced FLOWGUARDIAN, an automated solution for attack testflows extraction from unstructured cyber threat reports. FLOWGUARDIAN employs a multi-component architecture that includes contextual analysis, sequence analysis, and testflow generation modules. The contextual analysis module leverages language models to understand the semantics and context of the extracted information, ensuring that relevant details are accurately captured. The sequence analysis component organizes these details into coherent sequences that reflect the logical flow of cyber attacks. Finally, the testflow generation module translates these sequences into actionable testflows that can be directly used by SOCs for threat hunting and incident response. Our evaluations indicate that FLOWGUARDIAN not only accelerates the extraction process but also enhances the quality of the resulting testflows. This advancement enables the SOC team to improve its threat hunting and incident response capabilities by providing structured, actionable intelligence. For future work, we intend to integrate FLOWGUARDIAN with existing SOC tools and workflows to provide dynamic testflows at run-time, facilitating seamless adoption by practitioners and enhancing operational efficiency.

REFERENCES

[1] A. Alshamrani *et al.*, "A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities," *IEEE COMST*, vol. 21, no. 2, pp. 1851–1877, 2019.

[2] Google Cloud, "APT41 Initiates Global Intrusion Campaign Using Multiple Exploits," 2024.

[3] Mandiant - Google Cloud, "Game Over: Detecting and Stopping an APT41 Operation," 2023.

[4] Mandiant - Google Cloud, "Double Dragon APT41, a dual espionage and cyber crime operation," 2022.

[5] Google Cybersecurity Action Team, "Threat Horizons - April 2023 Threat Horizons Report," 2023.

[6] L. Li *et al.*, "Automated discovery and mapping ATT&CK tactics and techniques for unstructured cyber threat intelligence," *Computers & Security*, vol. 140, 2024.

[7] B. Abdeen *et al.*, "Smet: Semantic mapping of cve to att&ck and its application to cybersecurity," in *IFIP DBSec*, 2023.

[8] W. P. M. III *et al.*, "An Interview Study on Third-Party Cyber Threat Hunting Processes in the U.S. Department of Homeland Security," in *USENIX Security*, 2024.

[9] M. T. Alam *et al.*, "Looking Beyond IoCs: Automatically Extracting Attack Patterns from External CTI," in *RAID*, 2023.

[10] J. Devlin *et al.*, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv*, 2018.

[11] Y. Liu *et al.*, "Roberta: A robustly optimized BERT pretraining approach," *arXiv*, 2019.

[12] P. Nayak, "MUM: A new AI milestone for understanding information," *Google*, vol. 18, 2021.

[13] F. N. Motlagh *et al.*, "Large language models in cybersecurity: State-of-the-art," *arXiv*, 2024.

[14] D. Bhusal *et al.*, "Secure: Benchmarking generative large language models for cybersecurity advisory," *arXiv*, 2024.

[15] Y. Guo *et al.*, "A framework for threat intelligence extraction and fusion," *Computers & Security*, vol. 132, p. 103371, 2023.

[16] N. Prayogo *et al.*, "Context-aware attended-over distributed specificity for information extraction in cybersecurity," in *IEEE IEMCON*, 2022.

[17] H. Belani *et al.*, "Ontology-based cybersecurity for well-being, aging and health: A scoping review," in *IEEE MeditCom*, 2023.

[18] P. Abdurehim *et al.*, "A short review of relation extraction methods," in *ICICTA*, 2020.

[19] K. Satvat *et al.*, "Extractor: Extracting attack behavior from threat reports," in *IEEE EuroS&P*. IEEE, 2021.

[20] R. Kremer *et al.*, "IC-SECURE: Intelligent System for Assisting Security Experts in Generating Playbooks for Automated Incident Response," *arXiv*, 2023.

[21] P. Gao *et al.*, "Enabling efficient cyber threat hunting with cyber threat intelligence," in *IEEE ICDE*. IEEE, 2021.

[22] G. Husari *et al.*, "TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources," in *ACSAC*, 2017.

[23] B. H. S. Durga *et al.*, "Information extraction from text messages using natural language processing," in *ICCCI*, 2023, pp. 1–6.

[24] Z. Qiao *et al.*, "Improving cybersecurity named entity recognition with large language models," in *CSECS*, 2023.

[25] Google Cloud, "Apt41 us state governments: Threat intelligence insights," 2023.

[26] H. Saadany *et al.*, "Bleu, meteor, bertscore: evaluation of metrics performance in assessing critical translation errors in sentiment-oriented text," *arXiv*, 2021.

[27] Trend Micro, "Earth Baku An APT Group Targeting Indo-Pacific Countries With New Stealth Loaders and Backdoor," 2021.