TC–ACN-Based Cyber Incident Reporting Using Large Language Models

Manuel Zambelli[1,*,†], Refat Othman[1,†] and Barbara Russo[1,†]

[1]Software Engineering and Autonomous Systems (SEAS), Faculty of Engineering, Free University of Bozen-Bolzano, Italy

Abstract

Artificial Intelligence has become essential for many tasks in Cybersecurity. Nevertheless, security specialists are often required to perform manual tasks, such as reporting an event related to cyber incidents. According to the NIS2 directive of the European Union and its national implementation in Italy, certain types of companies are required to communicate cyber incidents with a significant impact to the Computer Security Incident Response Team (CSIRT). The National Cybersecurity Agency in Italy (ACN) has provided a document that describes a taxonomy to simplify the description of cyber incidents. However, this represents additional costs for companies in terms of human resources. In this paper, we explore whether LLM models can assist practitioners by automating the generation of TC–ACN vectors. To evaluate the proposed solution, we applied LLM-based classification to 52 attack logs extracted from the BOTSv3 dataset and generated TC–ACN vectors both manually and through six different model configurations that varied in subscription level, prompting session, and inclusion of the TC–ACN file. We assessed performance using Hamming accuracy to quantify correctness and counted undetermined values to measure semantic completeness. The best results were obtained with GPT-5.1 Plus, when the TC–ACN file was supplied and each query was executed in a fresh session, achieving a mean Hamming accuracy of 0.579. Moreover, the best LLM configuration produces more undetermined values for the predicates of the vectors than the human expert.

Keywords

Cybersecurity Incident, Large Language Models, System Log Analysis

1. Introduction

With the introduction of the European NIS2 Directive [1] and the corresponding national decrees (e.g., for Italy [2]), which transpose it, a large number of private entities and public administrations are now required to report cybersecurity incidents to the national Computer Security Incident Response Team (CSIRT), which will assist the affected organization in resolving them. An incident is defined as "an occurrence that actually or imminently jeopardizes, without lawful authority, the integrity, confidentiality, or availability of information or an information system; or constitutes a violation or imminent threat of violation of law, security policies, security procedures, or acceptable use policies" [3]. Thus, to be able to report incidents, organizations must gather information from various sources that may reveal or be affected by the malicious event (e.g., network traffic). In Italy, the Agenzia per la Cybersicurezza Nazionale (ACN) has issued a guideline to help security professionals classify and report incidents [4]. The guideline includes a taxonomy (TC–ACN) composed of 22 criteria (called predicates) that describe an incident. The 22 criteria are grouped into four macrocategories, i.e., *Baseline Characterization* (BC), *Threat Type* (TT), *Threat Actor* (TA), and *Additional Context* (AC). Each macrocategory is composed of predicates (four predicates for the *Baseline Characterization*, nine predicates for the *Threat Type*, two predicates for the *Threat Actor*, and seven predicates for the *Additional Context*). Each predicate is further composed of different categories of values, summing up to 144 possible choices. Consequently, before submitting an incident report to the CSIRT, Italian organizations must collect and analyze relevant internal data and map the results to the 22 predicates choosing among

Joint National Conference on Cybersecurity (ITASEC & SERICS 2026), February 09-13, 2026, Cagliari, IT

*Corresponding author.

†These authors contributed equally.

manzambelli@unibz.it (M. Zambelli); rothman@unibz.it (R. Othman); brusso@unibz.it (B. Russo)

0009-0000-9843-543X (M. Zambelli); 0000-0003-1227-9734 (R. Othman); 0000-0003-3737-9264 (B. Russo)

the 144 possible values. According to this taxonomy, an incident can therefore be described by a vector of 22 predicates. Each entry of the vector is specified by the macrocategory followed by ':' a predicate followed by '-' and the predicate value. An example of a vector is reported in the following box. It describes a high-severity incident caused by an attack. For instance, the first entry `BC:IM-DE` indicates a Baseline Characterization "BC" that describes the Impact "IM" as Data Exposure "DE", and is the disclosure of sensitive or confidential data to third parties.

Example of TC–ACN vector

<BC:IM-DE BC:RO-MA BC:SE-HI BC:VG-IT TT:DE-OT TT:IG-OT TT:MA-UN TT:VU-OT TA:AM-OT TA:AD-CR AC:AB-OT AC:AS-IT AC:IN-OT AC:PH-OT>

There exist tools to support security specialists in gathering data, alerting attacks, and responding to incidents, such as the Security Information and Event Management (SIEM) [5] and the Security Orchestration, Automation, and Response (SOAR) software [6, 7]. The SIEM collects and analyzes system logs to trigger an alert of an attack. System logs trace events generated by devices in a computer network, such as servers, firewalls, and endpoints. They typically contain semi-structured data about the event and the technologies that originated it. The SIEM uses them to generate alerts. The SOAR streamlines incident response workflows. It analyzes, enriches, and automates the response, and eventually updates the SIEM output. Examples for SIEM solutions are Splunk Enterprise Security[1] and IBM Security QRadar[2], or the open source ELK Stack[3]. However, being developed for general context, neither SIEM nor SOAR solutions provide an output that can be directly used to report an incident with the Italian TC–ACN taxonomy. Thus, in this work, we aim to explore how to automate the submission of an incident to the CSIRT by generating the TC–ACN vectors from system logs. In particular, we investigate whether an LLM can produce these vectors from system logs. Incorporating an LLM into the incident response workflow could help security specialists fulfill the tasks and duties required by NIS2 and reduce the manual workload. The complete analysis and results produced in this study are available in the TC–ACN GitHub repository [8]. Overall, we aim to answer the following two research questions.

RQ1: *To what extent can ChatGPT generate human-defined TC–ACN taxonomy vectors from attack logs?*
To address this question, we first randomly extracted 52 system logs from the BOTSv3 corpus[4]. This repository contains logs of attacks. Then, we prompted ChatGPT 5 under two different service configurations: the free ChatGPT version, GPT-5.1[5], and the commercial version GPT-5.1 Plus[6]. We also evaluated both models with and without resetting the prompt session. We then compared the TC–ACN vectors generated in each configuration with the vectors that have been manually generated by one of the authors, who has greater expertise in the TC–ACN taxonomy. To compare the manual and generated vectors, we first use the Hamming distance and then qualitatively inspected the mismatching entries of a vector pair (manual vs generated).

RQ2: *To what extent can LLM generate complete TC–ACN taxonomy vectors?*
To address this question, we analyze the completeness of the information provided with a vector (manual or generated). To this aim, we study the distribution of the entries of the vectors whose value was not determined (e.g. Other (OT)) and we analyzed qualitatively the corresponding predicates. For these entries, we then qualitatively inspected the mismatching entries between the manual and generated vectors.

Answering the questions, we provide the first insight into the automation of the incident workflow that is compliant with the new regulation for the NIS2. In particular, we provide an LLM-based approach that can be easily integrated into an organization's incident reporting automatic workflow. With our

[1]https://www.splunk.com/en_us/products/enterprise-security.html
[2]https://www.ibm.com/products/qradar-soar and qradar-siem
[3]https://www.elastic.co/elastic-stack
[4]https://github.com/splunk/botsv3
[5]https://chatgpt.com/
[6]https://chatgpt.com/plans/plus/

approach, we provide a prompt template and different service configurations that can be adopted depending on the business model of a company (e.g., free vs business service plan).

The paper is organized as follows. In Section 2 we describe the background, which includes a brief description of the NIS2 and the TC–ACN taxonomy, and related work. In Section 3 we illustrate our methodology. The database, which is used in our methodology, and how events are extracted from the database, are described in Section 3.1, while Section 3.2 describes how events are extracted from the database for further processing. The design of the prompt template is reported in Section 3.3, which is used in Section 4 for LLM queries for their evaluation. In Section 5 we compare the results of manual creation and automatic generation by LLM. Section 6 shows the limitations of our work. Section 7 concludes with final considerations.

2. Background and Related Work

In this section, we review some background and overview the literature for the automation of incident reporting. In particular, we describe the guideline for the TC–ACN taxonomy, and we review the most recent European and national directives on cybersecurity that motivate this work, and the existing frameworks that automate incident reporting.

The TC–ACN taxonomy guideline [4] provides a common language for sharing information on cybersecurity threats. It represents an alignment with international taxonomies in cybersecurity, for example, the *Reference Incident Classification Taxonomy* by the European Union Agency of Cybersecurity (ENISA) [9], and is adapted to the national regulatory context. As we mentioned, the taxonomy aims to describe four macrocategories. The macrocategory *Baseline Characterization* (BC) describes the impact, the root cause, the severity, and the geographic distribution of a cyber event. The macrocategory *Threat Type* provides detailed information about the scanning type, availability, the fraud type, the type of brand abuse, the type of data exposure, the type of information gathering technique, the type of malicious code, the type of social engineering technique, and the type of vulnerability. The macrocategory *Threat Actor* provides information about the adversary motivation and the adversary type. The macrocategory *Additional Context* (AC) provides supplementary details about the abusive content, the asset source geography, the involved physical asset, the involved operating system, the stability and the physical security of the compromised system, and the attack vector.

Each macrocategory is composed of more predicates (for example, the macrocategory *Baseline Characterization* is composed of the predicates *Impact*, *Root Cause*, *Severity*, and *Victim Geography*). Each predicate can assume one of different pre-defined values. For many predicates there is the possibility to choose an undetermined value, (*Other*, *None*, or *Unknown*), if no other possible values are applicable to the predicate to analyze.

The EU directives 2016/1148 (NIS 1) and 2022/2555 (NIS 2) of the European Union have established a legal framework for cybersecurity across member states [10], [1]. According to these directives, each EU member state is obliged to implement the national decrees to actuate them by the end of 2024. In Italy, a series of national decrees have been followed over the years to implement EU directives [11, 12, 13, 14, 15, 16, 2, 17]. These measures have led to the establishment of dedicated institutions and procedures for reporting and managing cyber incidents, including CSIRT Italia and the Agenzia per la Cybersicurezza Nazionale. CSIRT Italia was created in 2018 as the national structure responsible for receiving, analyzing, and responding to incidents, coordinating with EU counterparts and critical sectors. In 2021, the ACN was established to oversee the national cybersecurity strategy, manage risk, and ensure compliance with NIS 2. The national decrees have also introduced a formal notification process for reporting cybersecurity incidents based on the TC-ACN taxonomy: starting from January 2026, “essential” and “important” companies in each member state must report cyber incidents to the national CSIRT. The reporting process also requires organizations to maintain and use historical data for security analysis. One of the major sources of information in this respect are the system logs, that described events occurring in a system and possibly related to incidents, as incidents are events “compromising the availability, authenticity, integrity or confidentiality of stored, transmitted or processed data or of

the services offered by, or accessible via, network and information systems", NIS2 directive Art.6-(6) [1].

A key platform that collects system logs is Security Information and Event Management (SIEM). The platform collects logs across IT systems in order to trigger security alerts (for example, through the Windows Event Collector and the Linux Syslog Collector). Together with software such as Network Detection and Response (NDR) that collects network traffic and its anomalies and the Extended Detection and Response (XDR) that collects data on end-points and communication, it provides actionable insight on potential security events. Thus, the logs are used by the SIEM to identify threats and anomalies, which can generate security alerts. Security Orchestration, Automation, and Response (SOAR) tools [6, 7] work together with other security tools to retrieve and analyze the security data generated in a network. They also contain *playbooks*, which can initiate automatized steps to mitigate the cyber attack.

Recent work has focused on implementing frameworks for incident reporting. Vidal et al. [18] propose a framework for Operational Technology (OT) networks, focusing on the IT standards *NIST SP 800-61* and *ISO/IEC 27035*, and aiming to standardize how incidents are documented to improve coordination and analysis. Similarly, Agrarwal and Nene [19] introduced a unified schema to record AI-related incidents, emphasizing machine-readable fields for severity, causes, and harms. In addition, a key need was a simpler incident reporting method, as open formats often produced inconsistent or incomplete entries. The model introduced fixed fields for essential details (e.g., type, time, location, causes), improving consistency and later usability [19]. Both works highlight the need for consistent, high-quality data to enable meaningful analysis and decision-making in critical systems. Other frameworks are developed to support cybersecurity tasks, for example, to analyze and to report cyber incidents. Hu et al. [20] propose a framework for LLM-based threat analysis and response, Wei and Heim [21] propose a framework to understand the design of AI-powered incident reporting systems. Lin et al. [22] propose an adapted framework (based on LLMs) for different cybersecurity tasks, including log analysis and digital forensics. Mahmoud et al. [23] introduced ReACT_OCRS, an AI-driven voice-based cybercrime reporting system, which simplifies the reporting process. Machine learning has to be shown to be effective for threat prediction in critical infrastructure environments. An evaluation of 12,456 security events found that AI-driven models reached an accuracy of 89% when trained on data tailored to specific sectors. Despite these results, implementation remains challenging, with 63% of organizations reporting integration issues, particularly when deploying the technology alongside legacy systems [24].

However, the limitations of the approach depend on the LLMs [20] that are used. Moreover, the use of LLMs as General-Purpose artificial intelligence (GPAI) requires a cost-benefit analysis [21]. LLMs can also produce contradictory outputs affected by hallucinations [22]. Sometimes it is difficult to access real-world data [23], and therefore it is necessary to use synthetic data or data produced in a laboratory environment. These limitations can also be identified in our methodology. To address them, we give some advice in Section 7.

3. Methodology

This section gives an overview of the methodological pipeline adopted in our study as shown in Figure 1. In addition, it discusses all different configurations that are used in our work. The methodology requires a data set that contains a set of logs with different types of attack. For our experiments, it was important to find logs with a human-like description of the events, for example, "Failed user login", "Firewall Session Denied" or "Outbound TCP connection". These three examples illustrate simple descriptions of attack events. However, the descriptions can be very different, depending on the type of device that generates the event logs. A firewall can reveal the applied rule for an incoming or an outgoing network connection, while a Security Information and Event Management (SIEM) correlates information provided by different network devices, and therefore produces more high-level attack descriptions. These short text descriptions are extracted from the event logs, and they are then used to generate the TC–ACN vectors (either manually or automatically by LLMs), following the guidelines provided by ACN, the National Cybersecurity Agency in Italy. Finally, the manually and automatically generated vectors are compared.

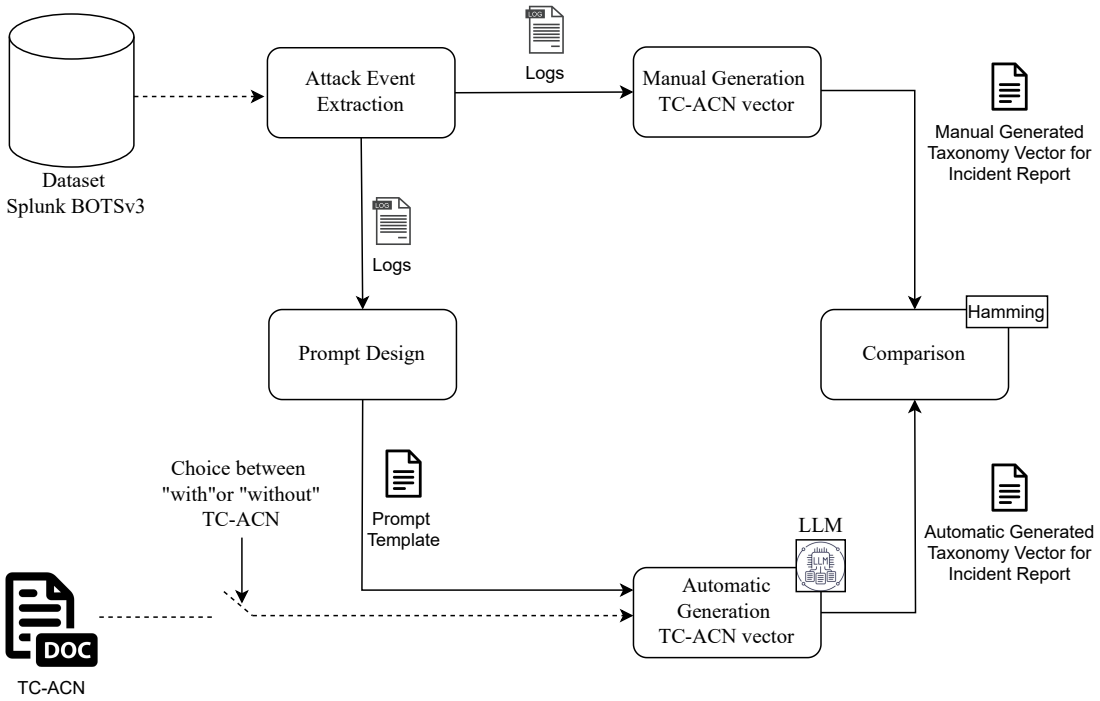

Figure 1: Overview of our methodology. Solid lines show the workflow between the activities, dashed lines show input/output of each activity. The switch symbol shows whether the TC–ACN guideline is included within the prompt.

To implement our proposed methodology, we collected 52 attack-annotated event logs from the Splunk BOTSv3 corpus (Section 3.1). We then designed a prompt leveraging the catalog [25] and identifying two parameters: the TC–ACN taxonomy instruction file and the logs. We selected ChatGPT 5.1 because it is a popular and the most recent conversational LLM that organizations can easily access depending on their business plan. In particular, we identified six different configurations to prompt the LLM, Table 1. These paths result from the combination of two LLM versions, i.e., the ChatGPT plus version and the ChatGPT free version. Both models are used with two different setups, one setup using the same session for all queries, and the other setup using a different session for each query. In addition, the ChatGPT plus version is also used with and without TC–ACN taxonomy instruction file. This yields four combinations for the ChatGPT plus version. In addition, also the ChatGPT free version is used with two setups (same session or different sessions), but not with the TC–ACN taxonomy file, because the free version does not permit uploading files to be included in the queries. The switch in Figure 1 shows the possibility to include or not the TC–ACN taxonomy instruction file. The last step of our methodology includes the comparison between the vectors generated by the human analyst and the LLM. The comparison is performed by measuring the distance between the pairs of vectors. For this, we used the Hamming distance [26] because it counts how many positions differ between two strings of equal length, but other suitable distance measures can be further considered. We further compare the pairs qualitatively, both to determine the predicates whose value mismatch (RQ1) and the predicates whose value is undetermined by both or one of the generated vectors.

3.1. Dataset

The publicly available Boss of the SOC version 3 (BOTSv3) dataset fits our purposes as it is a widely used corpus of system logs annotated for security-operations research[7]. Specifically, BOTSv3 provides

[7]https://github.com/splunk/botsv3

Table 1
Methodology configurations

Configuration	ChatGPT version	Session settings	TC–ACN file
C1	GPT-5.1 plus	different sessions	yes
C2	GPT-5.1 plus	same session	yes
C3	GPT-5.1 plus	different sessions	no
C4	GPT-5.1 plus	same session	no
C5	GPT-5.1 free	different sessions	no
C6	GPT-5.1 free	same session	no

a multi-source enterprise log collection containing more than two million events (2.083.056 events in total) from evidence captured during actual computer security incident or realistic lab recreations of security incidents. It contains information on benign user activity with several coordinated cyberattacks, including credential compromise, cloud account takeover, malware execution, web exploitation, lateral movement, and data exfiltration. Moreover, the dataset integrates heterogeneous telemetry sources such as firewall logs, Zeek network metadata, DNS activity, VPN authentication logs, Windows Event Logs, Linux syslogs, Sysmon process-level monitoring, web server logs, Active Directory authentication traces, and cloud-native security alerts such as AWS CloudTrail and GuardDuty findings. In addition, all events are stored in Splunk-compatible JSON/CSV format under the botsv3 index, with rich metadata fields capturing timestamps, hosts, sources, source types, network indicators, processes, and complete raw log payloads, thereby enabling comprehensive and fine-grained analysis across multiple stages of the attack lifecycle. To extract the relevant data, we filter the logs in BOTSv3 annotated as malicious or suspicious.

The resulting dataset contains `N = 3,983` attack-labeled events spanning multiple log sources. The dataset covers five primary source types of security-relevant telemetry: Firewall telemetry (`cisco asa`), HTTP traffic (`stream http`), Linux authentication logs (`linux_secure`), Windows Security events (`WinEventLog Security`), and cloudnative GuardDuty findings (`aws cloudwatch guardduty`). These categories correspond directly to the source types present in the dataset and collectively capture a broad range of attack behaviors. Table 2 shows examples of the logs in the dataset from all five source types. Table 3 summarizes the distribution of events across source types. Each record contains both the original raw message (`_raw`) and all associated Splunk metadata (e.g., timestamp, host, source type, source), along with the binary label `attack = 1`.

We defined a structured procedure to extract a representative number of attack logs from each source type while accounting for the strong class imbalance in the BOTSv3 dataset. Let N_i denote the total number of extracted attack events associated with source type i. For the three dominant source types, *Firewall telemetry*, *HTTP traffic*, and *Linux authentication logs*, we apply a logarithmic transformation with base 10 to reduce the impact of large frequency differences:

$$w_i = \log_{10}(N_i). \tag{1}$$

The resulting log-scaled values are normalized to obtain relative sampling proportions:

$$p_i = \frac{w_i}{\sum_{j \in S} w_j}, \tag{2}$$

where S denotes the set of the three dominant source types.

To determine the number of logs to extract, we select as a reference the source type with the smallest normalized proportion, denoted by

$$p_{\min} = \min_{i \in S} p_i. \tag{3}$$

A baseline of $b = 10$ instances is assigned to this source type. The number of selected logs for each remaining source type is then computed by proportional scaling:

$$n_i = \text{round}\left(b \cdot \frac{p_i}{p_{\min}} \right), \quad \forall i \in S. \tag{4}$$

Table 2
Examples of the attack logs.

Source type	Event log
Firewall telemetry	Aug 20 14:57:27 FROTHLYFW1 %ASA4106023: Deny udp src inside:192.168.9.20/38524 dst outside:192.168.10.106/514 by access-group ”inside_access_in” [0x0, 0x0]
Cloud-native GuardDuty findings	”version”:”0”,”id”:”70799bdd-f11d-d279-6b46-79193f7705bd”, ”detail-type”:”GuardDuty Finding”, ”source”:”aws.guardduty”, ”account”:”622676721278”, ... ”eventLastSeen”:”2018-08-20Y13:51:04Z”, ”severity”:2, ”title”:”Unprotected port on EC2 instance i-0cc93bade2b3cba63 is being probed.”
Linux authentication logs	Aug 20 14:10:22 ip-172.16.0.178 sshd[20228]: Invalid user admin from 113.162.80.84 port 3704
HTTP traffic	”endtime”:”2018-08-20T14:58:27.092084Z”,”timestamp”:”2018-08-20T14:58:27.092084Z”,”count”:1,”dest_ip”:”172.16.0.109”, ”site”:”34.227.100.38”,”status”:404,”uri_path”: ”phpmyadmin2index.php”,”sum$bytes_in$”:134,”sum$bytes_{o}ut$”: 478,”sum$time_taken$”:143053
Windows Security events	08/20/2018 09:29:43 AM LogName=Security SourceName=Microsoft Windows security auditing. EventCode=4625 EventType=0 Type=Information ... The authentication information fields provide detailed information about this specific logon request. - Transited services indicate which intermediate services have participated in this logon request. - Package name indicates which sub-protocol was used among the NTLM protocols. - Key length indicates the length of the generated session key. This will be 0 if no session key was requested.

Table 3
Distribution of extracted attack events across source types in the BOTSv3 attack dataset.

Source type	Count	Percentage	Selected instances
Firewall telemetry	3,738	93.85%	23
HTTP traffic	199	5.00%	15
Linux authentication logs	42	1.05%	10
Windows Security events	3	0.075%	3
Cloud-native GuardDuty findings	1	0.025%	1
Total	**3,983**	**100%**	**52**

For the two low-frequency source types, *Windows Security events* and *Cloud-native GuardDuty findings*, the logarithmic transformation is not applied; instead, all available attack logs are included in full due to their limited cardinality. The total number of selected logs is therefore given by:

$$T = \sum_{i \in S} n_i + \sum_{r \in R} N_r, \tag{5}$$

where R denotes the set of low-frequency source types.
Following this procedure, our final data set consists of T = 52 representative attack log entries, proportionally distributed across *Firewall telemetry*, *HTTP traffic*, and *Linux authentication logs*, while fully including all available *Windows Security events* and *Cloud-native GuardDuty findings*. Table 3 summarizes the distribution of attack logs across source types and reports the final number of selected logs obtained through the logarithmic proportional sampling strategy.

3.2. Creation of TC–ACN Vectors

According to our methodology, we have created the TC–ACN vectors both manually and with ChatGPT. From the process, we excluded the five predicates *Victim Geography*, *Adversary Type*, *Abusive Content*, *Asset Source Geography*, and *Outlook*, as the logs did not contain sufficient information to give them a value.

For the manual creation, we followed a strict, rule-based interpretation of ACN definitions to ensure internal consistency and reproducibility. No automated tools or LLM assistance were used during the process; all annotations were produced manually by using ACN taxonomic guidelines. As no description on how to assign specific values is reported in the guideline, we had to define our own guidelines for the interpretation of the predicates' values. Table 4 shows an example of the values' descriptions we defined for the macrocategory `Threat Type`.

Table 4
Values' interpretation for the macrocategory *Threat Type*

Predicate	Value	Description
TT:AC	CR	login attempts with different users
	NE	connection attempts, invalid or incomplete connections
	VU	for example, scanning of unprotected ports
	OT	other types of active scanning
TT:AV	DO	DoS attack or similar attack
	MI	non malicious configuration by humans
	OT	other types of availability issues
TT:FR	RE	improper use of credentials, channels, or services
	OT	other types of fraud
TT-BA	OT	other types of brand abuse
TT-DE	OT	other types of data exposure
TT:IG	AC	attempts to access communication channels
	SN	traffic monitoring
	SO	credential scanning
	OT	other types of information gathering
TT-MA	OT	no evidence of specific malware
TT-SO	OT	no evidence of a specific social engineering technique
TT:VU	SE	when the attempt was not blocked (by the firewall, etc.)
	OT	no evidence of a specific vulnerability

In case a predicate could not be assessed, according to the TC–ACN guidelines, one of the three values OT, NO and UN was used. For the LLM generation, we ran ChatGPT with the six configurations as in Table 1 and created 52 vectors.

3.3. Prompt Design

To enable consistent LLM-based incident classification, we rely on the complete ACN taxonomy, which includes 22 predicates and 144 officially defined values, together with their two-letter acronyms, ordering rules for macrocategories and predicates, and fallback mechanisms for missing information [4]. In our implementation, we use version 1.0 of the taxonomy released in November 2024. As mentioned in Section 3.2, we did not use the five predicates *Victim Geography*, *Adversary Type*, *Abusive Content*, *Asset Source Geography*, and *Outlook*. Due to its length and structural complexity, the full taxonomy is provided only to high-capacity frontier models (e.g., ChatGPT-5.1 plus) capable of handling long, structured context without degradation. Building on this taxonomy, we designed a unified prompt that instructs the model to parse log events, classify them, and generate the TC–ACN vector and the corresponding report. The prompt enforces deterministic outputs by explicitly defining the expected structure and constraining the vocabulary to the official taxonomy values. Following the prompt pattern catalog described by White et al. [25], we adopt the Persona pattern, specifying the model's role, the

Prompt Used for TC–ACN Classification

You are a cybersecurity incident analyst trained to classify events using the Italian "ACN - Tassonomia Cyber"
At the end of this prompt you will have LOGS/EVENTS about attack. Use ONLY the Italian "ACN - Tassonomia Cyber" (do not invent new categories) to classify and provide me the vector in one line.
————————————————————
YOUR TASKS:
1. Parse the logs (IPs, ports, accounts, processes, timestamps, auth details, errors, anomalies).
2. Classify the event strictly according to the ACN taxonomy.
3. Output EXACTLY one value for each of the 17 predicates (strict order):
`1 BC:IM-<value> 2 BC:RO-<value> 3 BC:SE-<value> 4 TT:AC-<value>`
`5 TT:AV-<value> 6 TT:FR-<value> 7 TT:BA-<value> 8 TT:DE-<value>`
`9 TT:IG-<value> 10 TT:MA-<value> 11 TT:SO-<value> 12 TT:VU-<value>`
`13 TA:AM-<value> 14 AC:IN-<value> 15 AC:OS-<value> 16 AC:PH-<value>`
`17 AC:VE-<value>`
- Always output 1 value per predicate.
————————————————————
The OUTPUT Response FORMAT:
One line: 17-field ACN vector (space-separated)
Example:
BC:IM-NO BC:RO-HU BC:SE-NO TT:AC-OT TT:AV-OT TT:FR-OT TT:BA-OT TT:DE-OT TT:IG-OT TT:MA-UN TT:SO-OT TT:VU-OT TA:AM-OT AC:IN-SR AC:OS-MI AC:PH-OT AC:VE-OT
————————————————————
LOGS/EVENTS TO ANALYZE
`<PUT HERE THE RAW LOG DATA>`

Figure 2: Prompt Template for the LLM

structured template, and instructions for refining answers. The final prompt template used in our methodology is illustrated in Figure 2.

4. Evaluations

To answer **RQ1**, we evaluate the model outputs using *Hamming accuracy* as our quantitative metric of correctness [26]. Because each TC–ACN vector consists of 17 independent predicates, we treat the task as a multi-label classification problem in which the model must correctly assign one categorical value per predicate [27, 28]. For each event i and predicate p, let $t_{i,p}$ denote the manually created value and $\hat{t}_{i,p}$ the corresponding value produced by the model. With N evaluated events and $P = 17$ predicates, Hamming accuracy is defined in Equation 6. We compute this value by parsing each model response into a structured 17-field TC–ACN vector, aligning each field with its corresponding manually created value, and counting the total number of predicate-level matches across all events ($17N$).

$$\text{Hamming Accuracy} = \frac{1}{PN} \sum_{i=1}^{N} \sum_{p=1}^{P} \left(\hat{t}_{i,p} = t_{i,p} \right) \tag{6}$$

This metric quantifies correctness at the predicate level and therefore provides a direct and comprehensive measure of how accurately an LLM can reproduce the TC–ACN taxonomy.
To answer **RQ2**, we evaluated the completeness of the TC–ACN vectors. We examine how much meaningful information is provided by each vector, independently of whether that information matches the manual generated dataset. In the TC–ACN taxonomy, a predicate may take either a determined value or an undetermined one, the latter typically represented as `OT` (Other), `NO` (None), or `UN` (Unknown). We therefore define completeness as the absence of such undetermined values.

Table 5
Example of a manually created TC–ACN vector and an automatically created TC–ACN vector (with the ChatGPT free version and always new session for each log). The log is a Linux log with description "Invalid user admin from 167.114.13.150 port 57198".

Creation	TC–ACN vector
Manual	<BC:IM-NO BC:RO-MA BC:SE-ME TT:AC-CR TT:AV-OT TT:FR-RE TT:BA-OT TT:DE-OT TT:IG-SO TT:MA-UN TT:SO-OT TT:VU-SE TA:AM-ES AC:IN-PR AC:OS-GN AC:PH-UN AC:VE-EV>
Automatic	<BC:IM-NO BC:RO-HU BC:SE-NO TT:AC-CO TT:AV-UN TT:FR-UN TT:BA-UN TT:DE-UN TT:IG-UN TT:MA-UN TT:SO-UN TT:VU-UN TA:AM-UN AC:IN-CO AC:OS-MI AC:PH-OT AC:VE-UN>

For each evaluated event i and predicate p, let $t_{i,p}$ denote the manual generated values and $\hat{t}_{i,p}$ the corresponding value produced by the model. Let $\mathcal{U} = \{\texttt{OT}, \texttt{NO}, \texttt{UN}\}$ denote the set of undetermined values. We define the indicator function:

$$u(x) = \begin{cases} 1, & \text{if } x \in \mathcal{U}, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Using this function, the number of undetermined predicates for the event i in the manually created and model-generated vectors is:

$$U_i = \sum_{p=1}^{P} u(t_{i,p}) \qquad \text{and} \qquad \hat{U}_i = \sum_{p=1}^{P} u(\hat{t}_{i,p}), \tag{8}$$

where $P = 17$ is the number of TC–ACN predicates. We then compute the mean number of undetermined values across all N evaluated events:

$$\overline{U} = \frac{1}{N} \sum_{i=1}^{N} U_i \qquad \text{and} \qquad \overline{\hat{U}} = \frac{1}{N} \sum_{i=1}^{N} \hat{U}_i. \tag{9}$$

A lower value indicates a semantically more complete vector, i.e., one that contains more fully specified predicate information. To understand where incompleteness occurs, we compute the number of *OT-equivalent mismatches* per predicate p, defined as cases where one vector contains an undetermined value and the other contains a determined one.

The manual generation of the TC–ACN vectors required a certain amount of time for the first samples. After becoming more familiar with the TC–ACN taxonomy, it took about 5-10 min for each sample at the beginning. This time is then decreased to a smaller value (2-5 min) for each sample after a first learning time. However, the processing time of ChatGPT (both plus version and free version) is on average 10-20 s. Table 5 shows a manual and an automatic created TC–ACN vector.

5. Preliminary Results

In this section, we show the preliminary results of our experiments and the results for the RQs.

5.1. RQ1: *To what extent can ChatGPT generate human-defined TC–ACN taxonomy vectors from attack logs?*

To answer RQ1, we evaluated ChatGPT in the six configurations in Table 1 using the Hamming Accuracy measure. Figure. 3 illustrates the boxplots of the measure for the six configurations. The best result (mean=0.579, median=0.588) is obtained with ChatGPT 5.1 plus prompted with the TC-ACN guideline and different sessions per log (*ChatGPTPlusWithTaxonomyFileWithDifferentSession*). Its narrow boxplot whiskers indicate consistent results across the 52 logs. Thus, in this case, the manually created vector

and the vector generated by the machine are constantly similar. With the free version instead (left boxplots), the similarity between human crafted and generated vectors varies the most among the logs. The worst similarity is obtained with the plus version of ChatGPT and without TC-ACN guideline (right boxplots). Finally, starting a new session for each log yields slightly greater similarity with the plus version, but not with the free version.

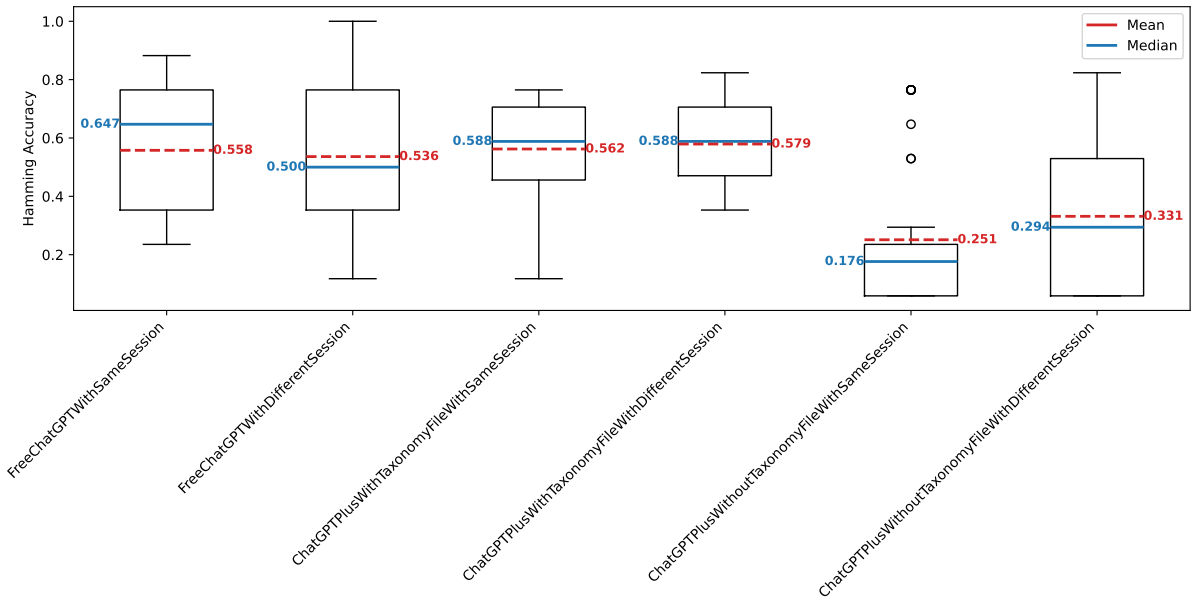

Figure 3: Hamming accuracy of the six LLM configurations.

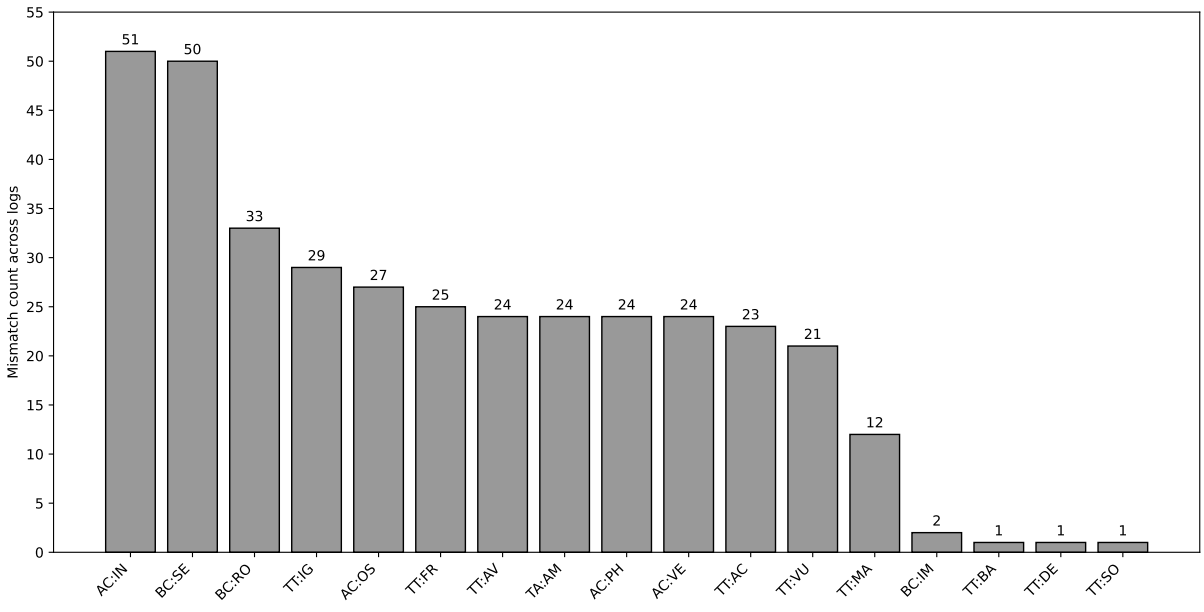

Figure 4: Predicate mismatch counts for ChatGPT-5.1 plus across different sessions using the ACN taxonomy.

Figure 4 illustrates the number of predicate mismatches aggregated across all 52 analyzed incident logs for ChatGPT-5.1 plus utilizing the taxonomy file using different sessions configuration, which achieved the highest Hamming accuracy among the six setups. Each bar represents how often a given TC-ACN predicate diverges between the LLM-generated vector and the manually curated one, considering all

mismatch types. The distribution is highly imbalanced: the large number of mismatches occurs for AC:IN and BC:SE, followed by BC:RO and TT:IG, indicating predicates where the model most frequently disagrees with human analysts. For a Cisco device, for example, the models provide either *Server* (AC:IN-SR) or *Other* (AC:IN-OT), whereas the human classifies the asset as a generic network device (AC:IN-NE), but it could also be classified as a security appliance (AC:IN-SA). Also the severity of an event (BC:SE) can allow different interpretations. In most of the logs, the machine does not classify the severity with one of the three levels *High*, *Medium*, or *Low*, but computes *Other*. Conversely, the human analyst tries to assign a specific value to the severity, depending on the type of event, for example, failed user login, or an attempt to access a public endpoint of a webpage, and the sender address (either from inside or outside respect to the analyzed network, described by a corresponding private or public IP address). In contrast, several predicates show a very low mismatch count, highlighting categories that remain largely stable and consistently classified across logs. However, future directions of this research should use more complete information about the system and the cyber incidents, in order to generate more complete TC–ACN vectors and to minimize the number of undetermined values. This could increase the Hamming accuracy. Nevertheless, an alternative calculation for the Hamming accuracy can be performed without considering the undetermined values.

Summary of RQ1 Findings

Prompting the Plus version of ChatGPT with the TC–ACN taxonomy guidelines file yields the highest similarity between the manually crafted vectors and those generated by the model. In contrast, the free version of ChatGPT produces similarities with high variability. It is also not advisable to omit the taxonomy file from the prompt.

5.2. RQ2: *To what extent can LLM generate complete TC–ACN taxonomy vectors?*

To answer RQ2, we assess the completeness of the information contained in the TC–ACN vectors. To this aim, we selected the best ChatGPT configuration resulting in RQ1 (i.e., *ChatGPTPlusWithTaxonomyFileWithDifferentSession*) and measured the number of undetermined predicate values (i.e., Other (OT), None (NO), or Unknown (UN)) in both manually created and automatically generated vectors. Figure 5 shows that the generated vectors contain, on average, more undetermined predicates than the manually crafted ones, indicating that the model produces less informative vectors from the logs. By counting the number of predicates for which the human expert and the model disagree and where one of the two selects an undetermined value (right boxplot in the figure), we can see that on average about half of the undetermined entries of the generated vectors are instead defined by the human expert.
We further analyzed the values of the predicates associated with OT-equivalent mismatches (i.e. the predicates for which the human expert and the model disagree and where one of the two selects an undetermined value). Figure. 6 shows the mismatch counts per predicate.
The two predicates with the highest number of undetermined values are BC:SE and AC:IN. BC:SE (Baseline characteristic: Severity) refers to the severity of an impact of a cyber event. AC:IN (Additional Context: Involved Asset) identifies the resources involved in the cyber event. These two predicates are, in fact, more difficult to determine with no knowledge of the system that generates the logs and its environment (the logs are too generic). The predicates of Baseline Characterization BC:IM (Impact) and BC:RO (Root cause), and three predicates of the Treat Type, TT:BA (Threat Type: Brand abuse), TT:DE (Threat Type: Data Exposure), and TT:SO (Threat Type: Social Engineering) are valued similarly (only one mismatch) by the human and the machine. The predicates and their values are more well-defined and easier to recognize. The figure also indicates that semantic incompleteness is not uniform across the taxonomy: the LLM struggles primarily with a specific subset of predicates that require more contextual or domain-specific cues, while others remain consistently stable. When we compare specific predicates between manually and automatically assigned values, we can notice particular approaches. For the predicate *Root Cause*, the manual decision says that the root cause is a human error, while the LLM classifies it as malicious. For the predicate *Fraud* the human says is can be a resource misuse, if someone

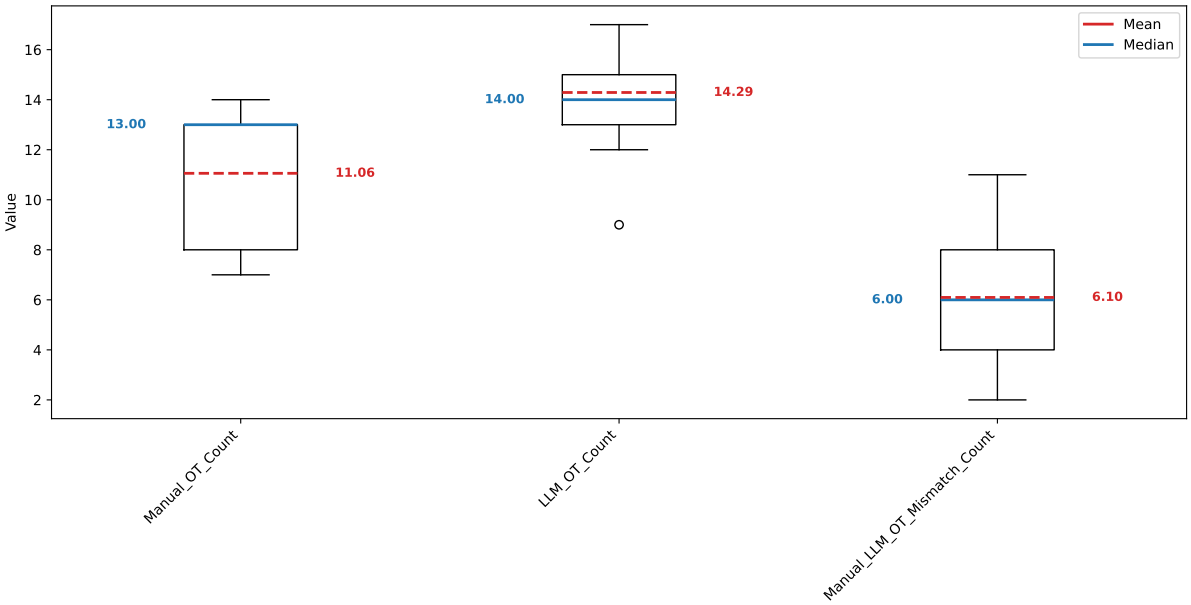

Figure 5: Distribution of OT-equivalent values across manually created, GPT predictions, and model–human mismatch counts.

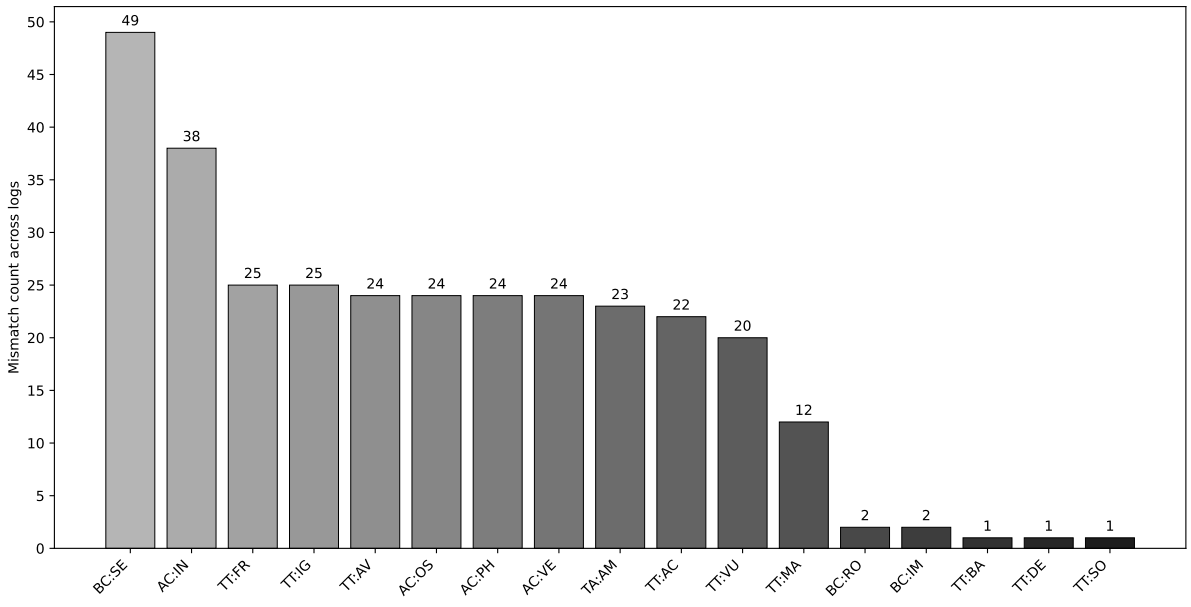

Figure 6: OT equivalent mismatch counts per TC–ACN predicate.

or some device tries to access a non existing account or a hidden or non existing folder on a webpage from outside the network, whereas the models assign sometimes the value *Other*.

Summary of RQ2 Findings

ChatGPT in its best configuration produces more undetermined values for the predicates of the vectors than the human expert.

6. Threats to Validity

This section discusses the main threats that may affect the validity of our study.

Construct validity. Construct validity concerns how well the collected data reflects the theoretical constructs under investigation [29]. In our work, this refers to the ability of LLMs to produce TC–ACN vectors consistent with the taxonomy. A key threat arises from the manual creation (see Section 3.2), which may introduce subjective judgments and variation across annotators. This risk is elevated when only one analyst is responsible, but can also occur in teams where experts interpret the taxonomy differently due to varying knowledge or experience, reducing consistency. Future research would benefit from standardized, publicly available datasets, with curated and manually created entries, to improve reliability and reproducibility.

Internal validity. Internal validity relates to how confidently observed results can be attributed to the methodological choices made during the study [30]. A threat in our context is the dependence on specific LLM versions. The access to paid models such as ChatGPT Plus may be limited, while free versions introduce constraints related to rate limits, and model availability. These factors could influence output quality or consistency. Additionally, the dataset used for evaluation (Splunk BOTSv3; Section 3.1) contains logs that originate from a limited set of devices and do not include generalized incident narratives typical of SIEM tools. To reduce these risks, we relied on uniform experimental settings; however, larger and more diverse datasets would further reinforce internal validity.

External validity. External validity refers to the extent to which study outcomes can be generalized beyond the experimental setting [30]. The practical adoption of our approach may vary across organizations due to differences in financial resources, infrastructure, and access to LLM subscriptions. Smaller organizations or SOC teams may face barriers in adopting paid tools, which could limit real-world applicability. Moreover, our work is tied to TC–ACN version 1.0 (November 2024), while version 2.0 of the taxonomy has since been released. Changes in predicate definitions, values, or category structure may reduce compatibility with the model outputs presented here. Future work should evaluate and adapt the methodology to reflect updated taxonomy guidelines and explore its performance on real-time SIEM data streams.

7. Conclusions

In this paper, we analyzed ChatGPT in extracting from logs of attacks the information needed to report the incident to the CSIRT according to the Italian incident report taxonomy. The extracted information is represented in the form of a vector, and the vectors crafted by a human expert and the ones generated by the machine are then compared. The results show that using the commercial version of ChatGPT prompted with the official Italian guidelines for reporting incidents is the best option to obtain vector similar to the ones of the human expert, but the machine provides vector with less complete information.

Our future work will focus on extending the evaluation to larger datasets and validating the vectors with CSIRT experts. We will also use the updated TC–ACN version 2.0 to assess the model's adaptability to evolving reporting standards. Finally, we will investigate alternative models by comparing local LLMs (e.g., an Ollama model) with online services (e.g., GPT Plus), considering trade-offs between performance, cost, and privacy for practical adoption in operational environments.

Acknowledgments

This work is funded by the European Union - Next Generation EU, Mission 4 Component 1 CUP: I52B24000510005. The authors also gratefully acknowledge the support of the Cybersecurity Laboratory (CSLab) at the Free University of Bozen-Bolzano funded by the EFRE-FESR 2021–2027 program, project EFRE1039, CUP: I53C23001690009. This research is supported by the European Social Fund Plus (ESF+), Project ESF2f30005, CUP: B56F24000100001.

Declaration on Generative AI

During the preparation of this work, the authors used Writefull for: Grammar and spelling check, Improve writing style. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content. ChatGPT was used for the research outcome.

References

[1] E. Union, EUR-lex, https://eur-lex.europa.eu/eli/dir/2022/2555/oj/eng, 2022.

[2] G. Ufficiale, Decreto legislativo 4 settembre 2024, n.138, https://www.gazzettaufficiale.it/eli/id/2024/10/01/24G00155/SG, 2024.

[3] N. I. of Standards, Technology, Incident response recommendations and considerations for cybersecurity risk management, special publication 800-61r3, https://doi.org/10.6028/NIST.SP.800-61r3, 2025.

[4] A. per la Cybersicurezza Nazionale, La tassonomia cyber dell'ACN, https://www.acn.gov.it/portale/documents/d/guest/acn_tassonomia_cyber_clear, 2025.

[5] S. Bhatt, P. K. Manadhata, L. Zomlot, The operational role of security information and event management systems, IEEE Security & Privacy 12 (2014) 35–41. doi:10.1109/MSP.2014.103.

[6] C. J. Brooks, P. A. C. Jr., Practical Industrial Cybersecurity, 1 ed., Wiley, 2022.

[7] R. A. Bridges, A. E. Rice, S. Oesch, J. A. Nichols, C. Watson, K. Spakes, S. Norem, M. Huettel, B. Jewell, B. Weber, C. Gannon, O. Bizovi, S. C. Hollifield, S. Erwin, Testing SOAR tools in use, Computers & Security 129 (2023). doi:https://doi.org/10.1016/j.cose.2023.103201.

[8] R. Othman, M. Zambelli, TCACN: Repository of analysis and results for TC–ACN experiments, https://github.com/CybersecurityLab-unibz/TC-ACN, 2025.

[9] E. U. A. for Cybersecurity, Reference incident classification taxonomy, https://www.enisa.europa.eu/publications/reference-incident-classification-taxonomy, 2018.

[10] E. Union, EUR-lex, https://eur-lex.europa.eu/eli/dir/2016/1148/oj/eng, 2016.

[11] G. Ufficiale, Decreto del presidente del consiglio dei ministri 17 febbraio 2017, https://www.gazzettaufficiale.it/eli/id/2017/04/13/17A02655/sg, 2017.

[12] G. Ufficiale, Decreto legislativo 18 maggio 2018, n. 65, https://www.gazzettaufficiale.it/eli/id/2018/06/09/18G00092/SG, 2018.

[13] G. Ufficiale, Decreto-legge 21 settembre 2019, n. 105, https://www.gazzettaufficiale.it/eli/id/2019/09/21/19G00111/s, 2019.

[14] G. Ufficiale, Decreto del presidente del consiglio dei ministri 14 aprile 2021, n. 81, https://www.gazzettaufficiale.it/eli/id/2021/06/11/21G00089/SG, 2021.

[15] G. Ufficiale, Testo coordinato del decreto-legge 14 giugno 2021, n. 82, https://www.gazzettaufficiale.it/atto/serie_generale/caricaDettaglioAtto/originario?atto.dataPubblicazioneGazzetta=2021-08-04&atto.codiceRedazionale=21A04841&elenco30giorni=true, 2021.

[16] G. Ufficiale, Legge 28 giugno 2024, n. 90, https://www.gazzettaufficiale.it/eli/id/2024/07/02/24G00108/SG, 2024.

[17] G. Ufficiale, Decreto del presidente del consiglio dei ministri 21 febbraio 2025, https://www.gazzettaufficiale.it/atto/serie_generale/caricaDettaglioAtto/originario?atto.dataPubblicazioneGazzetta=2025-05-21&atto.codiceRedazionale=25A02961&elenco30giorni=true, 2025.

[18] N. Vidal, N. Moradpoor, L. Maglaras, Everyone needs AIR: An agnostic incident reporting framework for cybersecurity in operational technology, arXiv preprint arXiv:2510.20858 (2025).

[19] A. Agarwal, M. J. Nene, Standardised schema and taxonomy for AI incident databases in critical digital infrastructure, in: 2024 IEEE Pune Section International Conference (PuneCon), IEEE, 2024, pp. 1–6.

[20] T. Hu, S. Zhuang, z. Guo, J. Sun, Y. Liu, W. Ma, H. Wang, L. Zhao, X. Zhang, A novel LLM approach

of cybersecurity threat analysis and response, in: Proceedings of the 16th International Conference on Internetware, Internetware '25, Association for Computing Machinery, 2025, p. 13–23. URL: https://doi.org/10.1145/3755881.3755888. doi:10.1145/3755881.3755888.
[21] K. Wei, L. Heim, Designing incident reporting systems for harms from general-purpose AI, 2025. URL: https://arxiv.org/abs/2511.05914. doi:10.48550/arXiv.2511.05914.
[22] X. Lin, J. Zhang, G. Deng, T. Liu, X. Liu, C. Yang, T. Zhang, Q. Guo, R. Chen, IRCopilot: Automated incident response with large language models, 2025. URL: https://arxiv.org/abs/2505.20945. doi:10.48550/arXiv.2505.20945.
[23] A. A. S. Mahmoud, W. Shishah, N. R. Mistry, ReACT_OCRS: An ai-driven anonymous online reporting system using synergized reasoning and acting in language models, IEEE Access 13 (2025) 92800–92815. doi:10.1109/ACCESS.2025.3571526.
[24] S. Busetti, F. M. Scanni, Evaluating incident reporting in cybersecurity. From threat detection to policy learning, Government Information Quarterly 42 (2025) 102000. doi:https://doi.org/10.1016/j.giq.2024.102000.
[25] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, D. C. Schmidt, A prompt pattern catalog to enhance prompt engineering with chatgpt, in: Proceedings of the 30th Conference on Pattern Languages of Programs, PLoP '23, The Hillside Group, 2023. URL: https://arxiv.org/abs/2302.11382.
[26] R. W. Hamming, Error detecting and error correcting codes, The Bell system technical journal 29 (1950) 147–160.
[27] A. Lentzas, E. Dalagdi, D. Vrakas, Multilabel classification methods for human activity recognition: A comparison of algorithms, Sensors 22 (2022) 2353.
[28] G. Tsoumakas, I. Katakis, Multi-label classification: An overview, International Journal of Data Warehousing and Mining (IJDWM) 3 (2007) 1–13.
[29] D. I. Sjøberg, G. R. Bergersen, Construct validity in software engineering, IEEE Transactions on Software Engineering 49 (2022) 1374–1396.
[30] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, et al., Experimentation in software engineering, volume 236, Springer, 2012.

A. Online Resources

The sources for the evaluation process are available at

- GitHub,