CFA-bench: Cybersecurity Forensic LLM Agent Benchmark and Testing

Francesco De Santis*§, Kai Huang*§, Rodolfo Valentim†, Danilo Giordano*,
Marco Mellia*, Zied Ben Houidi‡, Dario Rossi‡
*Department of Control and Computer Engineering, Politecnico di Torino, Torino, Italy
{francesco.desantis, kai.huang, danilo.giordano, marco.mellia}@polito.it
†Computer Science Department, University of Turin, Torino, Italy
rodolfo.vieiravalentim@unito.it
‡Huawei Technologies Co. Ltd, Paris, France
{zied.ben.houidi, dario.rossi}@huawei.com
§Equal Contribution

***Abstract*—This paper investigates the capabilities and limitations of Large Language Model (LLM) agents in solving cybersecurity forensic tasks, including incident response, digital evidence correlation, and threat attribution. To enable a fair comparison of agents and LLMs, we introduce CFA-bench, a novel benchmark designed to evaluate their forensic reasoning abilities. We leverage a controlled testbed where vulnerable services are instantiated, attacked, and monitored, generating forensic evidence in the form of packet captures and log traces. With it, we create 20 curated incidents targeting 13 distinct services, focusing on recent vulnerabilities. Each incident presents progressively complex checkpoints, culminating in the identification of the specific Common Vulnerabilities and Exposure (CVE).**

We evaluate different LLM-powered agent architectures, equipping them with essential forensic tools such as a PCAP Reader and an Information Retriever. Each agent is asked to analyse the incidents so that we systematically track their performance across different forensic checkpoints. While preliminary, our findings demonstrate the potential of LLM agents in cybersecurity forensics, revealing their strengths and critical areas for improvement.

This study underscores the need for standardized benchmarks to assess LLM agents in cyber threat analysis rigorously. For this, we make CFA-bench open to the research community. Our results provide a foundation for future research aimed at refining agent architectures and enhancing their forensic reasoning capabilities.

***Index Terms*—Large Language Model, Agent, Cybersecurity, Forensics**

I. Introduction

The rapid evolution of cyber threats and the increasing sophistication of cyberattacks have made digital forensics a cornerstone of modern cybersecurity. Traditional forensic methodologies, which rely heavily on human expertise and rule-based systems, often struggle to keep pace with the dynamic and complex nature of contemporary cyber incidents. As the catalogue of vulnerabilities keeps expanding, security analysts face the challenge of processing vast amounts of forensic data while maintaining accuracy and efficiency.

Recent advancements in Artificial Intelligence (AI), particularly Large Language Models (LLMs), present a compelling opportunity to augment cybersecurity forensics. LLM-based agents are AI systems that leverage an LLM to autonomously process information, make decisions, and interact with other systems to achieve specific tasks. They can autonomously understand, reason, and act based on natural language inputs, enabling them to perform complex tasks with minimal human intervention. These agents can analyse unstructured forensic data, correlate evidence, extract actionable intelligence, and automate tedious investigative tasks. Up to autonomously solve the forensic case.

Despite their potential, the integration of LLM agents in cybersecurity forensics introduces several challenges, starting from their proper testing. Unlike deterministic rule-based systems, LLMs generate responses probabilistically, raising concerns about their trustworthiness in high-stakes forensic investigations. This calls for standardised and properly defined benchmarks for comparing and evaluating different LLM agents' forensic capabilities.

This paper addresses these gaps by proposing CFA-bench, a benchmark for LLM agent testing in forensic network trace analysis. Specifically, we aim to answer the following research questions:

- **RQ1**: How can we effectively assess LLM agents in the forensic analysis of network traces?
- **RQ2**: Are LLM agents a promising solution for forensic analysis?

To answer these questions, CFA-bench introduces a framework designed to evaluate LLM agents in forensic investigations. Our benchmark provides agents with real-world forensic scenarios where analysts must infer security incidents from heterogeneous data. We set up a controlled testbed where we can easily deploy a vulnerable server and attack it with a known CVE vulnerability while collecting network traffic and application logs, i.e., the evidence we later provide the agent to solve the forensic case. We already provide 20 test cases, involving 13 distinct services, and more can be easily added.

For each test case, we establish a comprehensive evaluation framework for agent forensics. Specifically, we assign the Agent-Under-Test (AUT) the following tasks: (i) determine whether the attack was successful; (ii) identify which vulnerable service the attacker targeted; (iii) confirm if the service

was indeed vulnerable; and (iv) pinpoint the exact CVE that was exploited. These tasks act as checkpoints that allow us to observe the AUT progress.

Armed with the benchmark, test cases, and tasks, we compare the performance of three different LLM-agent architectures, from the simple ReACT [1], ReACT+Summary [2], to the Decoupled ReACT+Summary. Here the LLMs are used to generate both reasoning traces and task-specific actions in an interleaved manner. At each iteration, a summary stage may constrain the amount of information they get to avoid overflowing the LLM context window. These agents can autonomously process the packet trace via a PCAP reader based on tshark, and browse the CVE vulnerability database via an Information Retriever tool.

We challenge each agent to solve each test case of CFA-bench. We provide the PCAP file and ask the agent to answer the four tasks while observing their actions and output, finally enabling their fair comparison. Our preliminary results show the potential of LLM agents in solving forensic cases, supporting future research in both extending the agent architectures as well as enhancing their abilities to process more information during the investigations. For instance, we are working on offering the agents the ability to harvest the attacked system logs via direct CLI access. The benchmark, along with the associated code and implementation details, is publicly available[1]

II. Related Work

Evaluating the cybersecurity capabilities of Large Language Models (LLMs) has gained increasing attention, leading to the development of multiple benchmark datasets. Table I summarizes key efforts in this space, highlighting their focus areas and methodologies.

Several benchmarks assess LLMs in static security-related tasks, such as vulnerability detection, debugging, and security knowledge evaluation. For example, Securityeval [3], PythonSecurityEval [12], and Assessing Cybersecurity Vulnerabilities in Code LLMs [17] focus on evaluating the effectiveness of LLMs in identifying and patching security flaws in code. Similarly, debugging-oriented datasets such as DebugBench [9] measure the model ability to detect and correct bugs.

Other benchmarks focus on security knowledge assessment via multiple-choice or Q&A tasks. OWL-Bench [5], SecEval [7], CyberMetric [10], and CyberBench [10] prompt the LLM to directly evaluate its general cybersecurity expertise. Meanwhile, SecQA [8] and SECURE [14] benchmark models in cybersecurity advisory and problem-solving tasks.

Some benchmarks extend beyond security knowledge and examine practical applications in system and network administration. For instance, NetEval [6] and Can LLMs Understand Computer Networks? [13] assess LLMs' network operations capabilities, while OpsEval [11] provides a comprehensive suite for IT operations evaluation.

[1]URL of the github repository: https://github.com/francescoTheSantis/Agent-Benchmark-for-Networking.git

Moreover, there are recent efforts focusing on creating specialised benchmarks for agents to evaluate cybersecurity capabilities [15], [18], [19]. For example, SecBench introduces a multi-faceted approach, incorporating multiple-choice and short-answer questions, assessing knowledge retention and logical reasoning in both Chinese and English [15]. Similarly, HackSynth utilises Capture The Flag (CTF) challenges to test autonomous penetration testing, providing a standardised framework for evaluating LLMs in cybersecurity [18]. InterCode-CTF also adopts the CTF format, tasking LLM agents with uncovering flags from vulnerable computer programs, thereby evaluating skills such as reverse engineering and forensics [19]. AgentQuest offers a modular framework supporting diverse benchmarks and agent architectures, along with metrics like progress and repetition rates to debug agent behaviour [20]. [16] proposes a complementary open-source benchmark featuring vulnerable Docker containers across two difficulty levels with basic in-vitro *pentest* scenarios and real-world cases. Again, there is a gap in these benchmarks to evaluate their forensics capabilities in a realist scenario.

In summary, existing benchmarks primarily evaluate knowledge retrieval, reasoning, and static code or text-based security tasks. However, real-world cybersecurity threats often involve dynamic, evolving attack patterns that require proactive and realistic reasoning capabilities. Current benchmarks lack an explicit focus on measuring how well LLMs can detect cyber-attacks from live network traffic and anomaly detection.

CFA-bench aims to fill this gap by introducing a standardized evaluation framework for assessing LLM agents in realistic network-based attack detection. Unlike existing works, our benchmark focuses on:

- Identifying attack signatures in raw network traffic.
- Evaluating anomaly detection capabilities of LLMs in network logs.
- Measuring response accuracy when distinguishing between benign and malicious activities.

By addressing this gap, our benchmark provides a critical evaluation framework for researchers and practitioners, ensuring LLMs can contribute effectively to network security beyond static assessments.

III. Incident Data Collection

In the literature, datasets containing labeled network traces are often outdated, incomplete, or unsuitable for Natural Language Processing (NLP). Datasets like CSIC 2010 Web Application Attacks [21], CSE-CIC-IDS2018 [22], and NSL-KDD [23] have been commonly used for years in cybersecurity detection tools. However, there is a significant concern that this data may have already been included in the training datasets of large language models, potentially compromising the accuracy of the results [24].

To address this gap and collect a more realistic dataset of labelled incident traces, we set up a controlled testbed where we run a (vulnerable or not) service and attack it with a known vulnerability. With this testbed We gather data from

TABLE I: Cybersecurity Benchmark Datasets for LLM Evaluation

Benchmark Dataset	Publication Date	Task Description	How they use LLM	What They Measure
Securityeval dataset [3]	2022-11-09	Mining vulnerability examples to evaluate ML-based code generation	Prompt to Generate Code Snippets	Vulnerability detection and patching
LLMSecEval [4]	2023-03-16	Dataset of natural language prompts for security evaluations	Prompt to Generate Code Snippets	Response accuracy to security-related prompts
OWL-Bench [5]	2023-09-17	Benchmarking LLMs on operational tasks in cybersecurity	Multiple-choice, Q&A	Operational and cybersecurity task handling
NetEval [6]	2023-09-19	Evaluating netops capabilities of pre-trained LLMs	Multiple-choice, Q&A	Network operations knowledge and practical skills
SecEval [7]	2023-12-20	Comprehensive evaluation of foundation models in cybersecurity	Multiple-choice, Q&A	Cybersecurity knowledge
SecQA [8]	2023-12-26	Evaluating LLMs in computer security through QA	Multiple-choice, Q&A	Security knowledge and problem-solving
DebugBench [9]	2024-01-11	Evaluating debugging capabilities of LLMs	Code Snippets Pass Rate	Debugging accuracy, bug detection and fixing
CyberMetric [10]	2024-02-12	Evaluating LLMs' knowledge in cybersecurity	Multiple-choice, Q&A	Cybersecurity knowledge
OpsEval [11]	2024-02-16	Comprehensive IT operations benchmark suite for LLMs	Multiple-choice, Q&A	IT operations knowledge and performance
PythonSecurityEval [12]	2024-02-19	Assessing LLMs' ability to patch security issues	Prompt to Generate Code Snippets	Patch effectiveness, correctness, and security impact
Can LLMs Understand Computer Networks? [13]	2024-04-22	Assessing understanding of computer networks for virtual system admins	Q&A	Network management and troubleshooting skills
SECURE [14]	2024-05-30	Benchmarking generative LLMs for cybersecurity advisory	Multiple-choice, Q&A	Advisory quality, accuracy, and relevancy
CyberBench [10]	2024-06-03	Evaluating LLMs' performance in cybersecurity-related tasks	Multiple-choice, Q&A	Broad range of cybersecurity competencies
SecBench [15]	2024-12-30	Evaluating LLMs in computer security through QAs	Multiple-choice, Q&A	Cybersecurity knowledge and reasoning
AutoPenBench [16]	2024-10-04	Evaluating LLM agents in automated penetration testing	Solve pentesting and CTF	Success rate to solve penetration testing tasks

20 incidents targeting 13 distinct services, including databases, middleware, servers, etc., see Table II for details.

The benchmark includes 15 vulnerabilities with disclosure dates spanning from 2016 to 2024. The CVSS severity scores [25] range from low (3.7, which exposes limited system information) to critical (10, allowing for arbitrary remote code execution). Of the total incidents, 15 are successful, while 5 are unsuccessful. Note that CFA-bench easily allows to extend the set of incidents.

A. Incident Collection Methodology

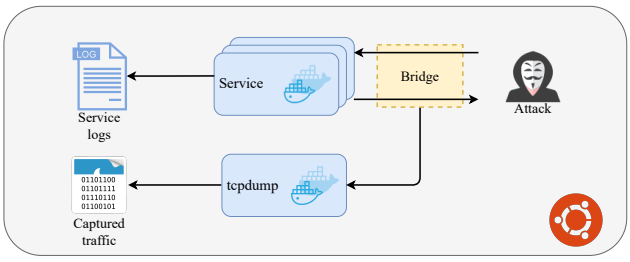

Fig. 1: Schema of the benchmark collection procedure. The service runs in container(s) and we attack it while collecting traffic and logs

TABLE II: List of collected incidents for benchmark.

Service	Version	CVE	Vulnerable	Attack success
Couchdb	3.2.1	CVE-2022-24706	True	True
Grafana	8.2.6	CVE-2021-43798	True	True
Apache HTTP Server	2.4.49	CVE-2021-41773	True	True
Apache HTTP Server	2.4.50	CVE-2021-42103	True	True
Jenkins	2.441	CVE-2024-23897	True	True
Joomla	4.2.7	CVE-2023-23752	True	True
Apache Solr	8.11.0	CVE-2021-44228	True	True
phpMyAdmin	4.4.15.6	CVE-2016-5734	True	True
phpMyAdmin	4.8.1	CVE-2018-12613	True	True
Cacti	1.2.22	CVE-2022-46169	True	True
Apache Airflow	1.10.10	CVE-2020-11981	True	False
Apache Airflow	1.10.10	CVE-2020-11981	True	True
SaltStack	2019.2.3	CVE-2020-11651	True	True
SaltStack	3002	CVE-2020-11651	False	False
Apache APISIX	2.9	CVE-2021-45232	True	True
Apache APISIX	2.9	CVE-2021-45232	False	False
Apache ActiveMQ	5.14.2	CVE-2017-15709	False	False
Apache ActiveMQ	5.13.2	CVE-2017-15709	True	True
GitLab	13.10.3	CVE-2021-22205	False	False
GitLab	13.10.0	CVE-2021-22205	True	True

Incidents are generated by deploying various applications and launching attacks that exploit the CVEs associated with them. Figure 1 illustrates the overall setup. We begin by deploying a virtual machine configured with a container runtime. For enhanced isolation, we run a separate container to capture

network traffic through a dedicated network bridge that is not shared with the virtual machine. The experiment is conducted on Ubuntu 20.22 using the Docker container engine.

We leverage information from open sources like Exploit-DB[2]. The attack is initiated from the virtual machine, with each attack executed using a script designed to exploit the specific CVE. For each incident, we collect the following data to be used as input for the agents:

- **Dockerfile**, used by the administrator to configure the service environment.
- **Service configuration files**, customized for the specific running environment.
- **Network traffic** from the incident, provided in raw .pcap format, retaining only the traffic related to the malicious attack.
- **Service logs** recorded during the attack.

Finally, once collected, we annotate the dataset considering the tasks we desire the agent to fulfil.

IV. Benchmark Design

CFA-bench is designed to be as realistic as possible, closely mimicking how a security analyst would evaluate an incident. Its goal is not only to assess LLMs' cybersecurity knowledge but also to challenge the agent's ability to adapt to real world scenarios. For example, the volume of the evidence traces can easily exceed the context window of LLMs. We evaluate the AUT by tasking it to solve a problem while meticulously recording each step. By establishing checkpoints throughout the process, we incrementally observe and assess the agent's performance, gaining detailed insights into its operational capabilities.

Therefore, we provide the model with the path to the folder containing the task evidence files that it must autonomously open and process. The AUT is equipped with various tools for parsing these traces, from reading the raw files to running specific commands for analysis.

To evaluate the forensic capabilities of LLM agents, we define a ground truth composed of high-level checkpoints that serve as intermediate tasks for assessing the agent's understanding of the incident across multiple dimensions. These unordered checkpoints ensure that the accuracy of the agent's forensic conclusions is effectively measured, while also mitigating the risk of error propagation from previous missteps.

The checkpoints are binary classification problems, and include:

- **Attack Success Evaluation**: Did the agent correctly assess whether the attack was successful?
- **Service Identification**: Did the agent correctly identify the service under attack?
- **Service Vulnerability Assessment**: Did the agent correctly identify if the service was vulnerable to the attack?
- **CVE detection**: Was the specific vulnerability (CVE) accurately identified?

[2]https://www.exploit-db.com/

For each incident, we manually label the 4 checkpoints. Given then the AUT output, we check if it correctly completes the checkpoints by simple pattern matching operations.

V. Agent Design

A. What is an LLM-powered Agent?

An agent is an autonomous system that interacts with its environment to achieve specific goals. It perceives its surroundings, processes information, and autonomously takes actions to influence the environment based on its objectives [26]. In the context of LLM-powered agents, these systems integrate LLMs to enhance reasoning, decision-making, and planning. The core structure of an agent consists of the following key components [26]:

- **Reasoning & Decision-Making:** By utilizing the LLM, the agent analyses observations, formulates strategies, and determines the next course of action, i.e., it formulates thoughts.
- **Memory:** The agent may incorporate short-term or long-term to retain past interactions and enhance future decision-making. Following [16], we denote the long-term memory as *Scratchpad*.
- **Actions:** The agent operates within a predefined action space, allowing it to interact with both internal memory and the external environment. Additionally, it can execute operations by leveraging tools through a text-based interface, i.e., composing CLI commands.

Through iterative cycles of perception, reasoning, and action, the agent continuously learns and adapts, allowing for the autonomous execution of complex tasks.

B. Agent Architecture

Several agent architectures have been presented in the literature. Here we focus on the three most commonly adopted:

a) ReACT Reasoning [1]: In the ReACT framework, the LLM is tasked with providing both a thought and an action. The thought represents the reasoning process, while the action refers to the next step or decision. The environment responds with an observation, which is stored in the *Scratchpad*. This memory allows the LLM to track previous interactions and use them for more informed decisions in subsequent steps, facilitating a more dynamic and context-aware reasoning process. The diagram in Figure 2 illustrates this basic flow of ReACT reasoning, where the LLM's thought-action cycle is reinforced by continuous feedback from the environment. Which in one case is the result of operating on the incident evidence, i.e., the PCAP file, and retrieving information from the web.

b) ReACT+Summary [2]: Building upon the ReACT framework, ReACT+Summary incorporates the concept of summarizing the ongoing analysis. After the LLM performs the reasoning step, it is prompted to generate a concise summary of the analysis performed so far. This summary serves as a new layer of context, which is then used as input by the LLM. By incorporating this self-reflection mechanism, the LLM can refine its reasoning over time, integrating the context provided by the summary to produce more precise

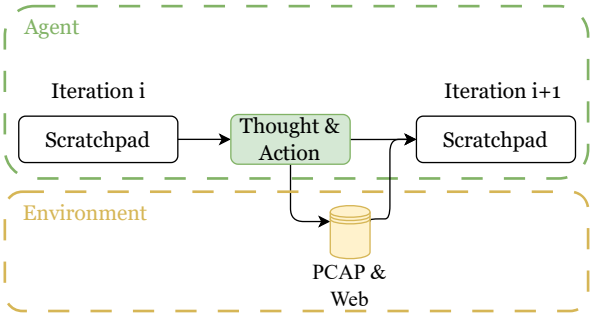

Fig. 2: The LLM is asked to provide both a thought and an action. Consequently, an observation from the environment is obtained and stored in the so-called scratchpad (memory)

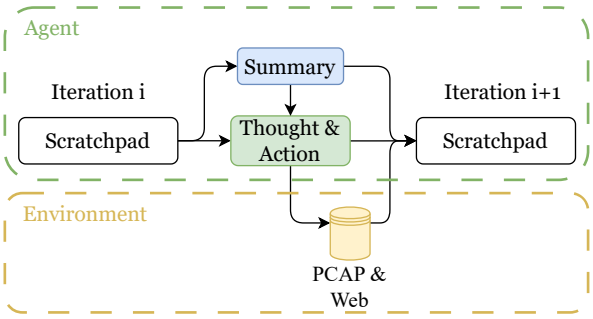

Fig. 3: The LLM is asked to provide a concise summary of the analysis performed so far. It is used to provide more context to the same LLM in the elaboration of the thought/action.

thoughts and actions. This iterative feedback loop is visualized in Figure 3, where the produced summary, in addition to the *Scratchpad*, guides the subsequent reasoning process.

c) Decoupled [16]: The idea behind this architecture is to further enhance the reasoning capabilities by decouling the thought and action components from the ReACT framework. This decoupling should provide the LLM with a clearer distinction between the cognitive process (thought) and the decision-making process (action), potentially allowing for more elaborate reasoning steps. The thought phase can be used to analyze the situation deeply, while the action phase focuses on implementing the most effective strategy based on the analysis. This separation, as shown in Figure 4, increases the ability of LLM to reason more thoroughly and systematically, leading to better decision making.

Note that the more complex ReACT+Summary and Decoupled architectures may underperform in real tests as the agent may lose focus. Contrary to the expectation that breaking tasks down should improve the agent performance, state-of-the-art models like GPT-4 and GPT-4o demonstrate better results when handling multiple instructions simultaneously (Multi-Task Inference) rather than sequentially (Single-Task Inference) [27]. In this paper, we investigate if this holds for forensic tasks too.

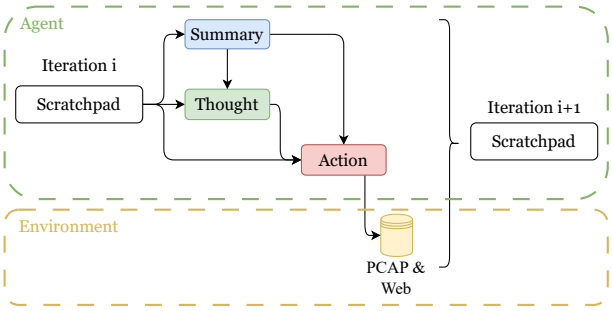

Fig. 4: We further decompose the Agent reasoning by also decoupling thought and action in order to increase the reasoning capabilities of the Agent.

C. Tools

We enable the agent to interact with the data and external sources through the usage of specifically designed tools:

a) PCAP Reader: This tool allows the LLM to analyze network traffic captured in PCAP files by composing and executing a `tshark` command from the command line. Specifically, it runs the command in Listing 1 to extract information about the relevant packet. It allows the agent to parse, analyze, and retrieve information from the raw network traces. By running the `tshark` command and processing its output, the LLM can gain deeper insights into the network activity during the incident, assisting in making informed decisions based on the PCAP trace.

```
tshark -r {pcap_file} \
    -Y frame.number=={frame_number} \
    -T fields -e data
```

Listing 1: PCAP Reader command

b) Information Retriever: The Information Retriever tool allows the LLM to retrieve information from external sources, such as the web, to supplement its internal knowledge with further information. To optimize the retrieval process, we use a Retrieval-Augmented Generation (RAG). RAG ranks the available information chunks based on their relevance to the query at hand, ensuring that the LLM is shown the most pertinent data. The information retriever helps the LLM gain access to up-to-date and contextually relevant information, improving the quality of its reasoning and actions. A detailed explanation of this tool is shown in Appendix A.

VI. Prompts

The LLM results are directly related to the quality of the prompt, a concept known as *prompt engineering*. By crafting the prompt effectively, we can optimize the performance of LLMs for specific tasks. In our approach, besides the Task Prompt, we adopt a Chain of Thought (CoT) prompting strategy [28] for all the prompts. Below, we provide the prompts as

listings. Introduced in 2022, chain-of-thought (CoT) prompting enables complex reasoning capabilities through intermediate reasoning steps. Refer to Listings 2–6 for the complete definitions.

Listings 2, 3, 4, 5 and 6 show the Task Prompt, the Summary Prompt, the Thought Prompt, the Action Prompt and the ReACT Prompt used in our agents.

```
Analyze the provided PCAP (Packet Capture) file to accomplish
the following tasks:

1. Analyze the PCAP in order to have an understanding of the
   network traffic.
2. Identify the services and applications involved in the
   network traffic.
3. Determine the relevant CVEs (Common Vulnerabilities
   and Exposures).
4. Assess whether the services or applications are vulnerable
   to the identified attacks.
5. Evaluate whether the attacks were successful.

HINTS:
- Use the tools provided to extract and expand the data from
  the PCAP file.
- Utilize the online search tool to verify the findings and gather
  additional information.
```

Listing 2: Task Prompt

```
Role: You are a specialized network forensics analyst.
You are working towards the final task on a step by step manner.

Instruction:
- Provide a complete summary of the provided prompt.
- Highlight what you did and the salient findings to accomplish the
      task.
- Your summary will guide an autonomous agent in choosing the correct
       action in response to the last observation to accomplish the
      final task.

Context: {context}
```

Listing 3: Summary Prompt

```
Role: You are a specialized network forensics analyst.
You are working towards the final task on a step by step manner.

Instruction:
- I will give you the summary of the task and the previous steps, the
       last action and the corresponding observation.
- By thinking in a step by step manner, provide only one single
      reasoning step in response to the last observation and the task
- Your thought will guide an autonomous agent in choosing the next
      action to accomplish the final task.

Summary: {summary}
Last Step: {last_step}
```

Listing 4: Thought Prompt

```
Role: You are a specialized network forensics analyst.
You are working towards the final task on a step by step manner.

Instruction:
- I will give you the summary of the task and the previous steps and
      a thought devising the strategy to follow.
- Focus on the task and the thought and provide the action for the
      next step.

Summary: {summary}
Last Step: {last_step}
New Thought: {thought}
```

Listing 5: Action Prompt

```
Role: You are a specialized network forensics analyst.
You are working towards the final task on a step by step manner.

Instruction:
- I will give you the task context, the previous steps and the last
      thought, action and the corresponding observation.
- By thinking in a step by step manner, provide only one single
      reasoning step in response to the last observation and the
      action for the next step.

Context: {context}
Last Step: {last_step}
```

Listing 6: ReACT Prompt

VII. Results

Here, we present our results from evaluating the three agent architectures. We use GPT-4o as the LLM with a temperature[3] set to 0, we test each agent to solve each of the 20 incidents. Given the inherent non-determinism of the LLM, we repeat each analysis three times, and report the average performance. Additionally, we limit each agent to stop after 50 iterations to prevent it from getting stuck in infinite loops or repeatedly revisiting the same actions, ensuring efficiency and timely results. We consider the output as **Correct** if it matches the ground truth label in the checkpoint. If the agent's output is incorrect, or if it prematurely aborts its investigation due to the exhaustion of the context window[4], we deem the response incorrect.

Table III summarizes GPT-4o's performance, presenting the success rate (percentage of Correct answers) for each checkpoint type, as well as additional diagnostic metrics, including the average token count generated during analysis (Verbosity), the average number of abort due to context window exhaustion (Aborted), and the average number of iterations required to complete a benchmark (Iterations).

TABLE III: Performance of LLM agents on CFA-bench.

Metric	ReACT	ReACT+Summary	Decoupled
Attack Success Evaluation	13.3%	23.5%	17.3%
Service Identification	42.3%	37.5%	22.5%
Service Vulnerability Assesment	42.85%	40.0%	25.0%
CVE Detection	14.28%	7.5%	0.0%
Verbosity	26k	54k	47k
Aborted	1.5	4.5	3.5
Iterations	13.3	23.3	17.3

Overall, the agents exhibit promising capabilities in analysing most incidents in the benchmark. In terms of performance, the ReACT agent achieves the highest success rates in Service Identification (42.3%), Service Vulnerability Assessment (42.85%) and CVE Detection (14.28%), demonstrating the strongest forensic capabilities. The ReACT+Summary

[3]The temperature parameter in an LLM controls the randomness of its output. Lower values (e.g., close to 0) make the model more deterministic, favoring high-probability responses, while higher values (e.g., above 1) increase diversity and creativity by allowing more low-probability words.

[4]The context window in LLMs refers to the maximum number of tokens the model can process in a single pass, which limits the amount of text it can consider.

agent, while lagging in CVE Detection (7.5%), outperforms the other methods in Attack Success Evaluation (23.5%), suggesting that its summarization approach enhances high-level reasoning but at the cost of detailed accuracy. In contrast, the Decoupled agent underperforms significantly, achieving only 22.5% in Service Identification and failing entirely in CVE Detection (0.0%).

Overall, the results highlight that ReACT strikes the best balance between accuracy and efficiency. In contrast, ReACT+Summary trades off precision for verbosity, and Decoupled struggles across all key metrics. All in all, the straightforward ReACT strategy consistently outperforms the others, underscoring its superior efficiency and effectiveness in the task at hand.

These findings confirm that adding a summarization subtask (ReACT+Summary) or decoupling the reasoning and action steps (Decoupled) appears to confuse the agent, causing it to lose focus. As previously said, state-of-the-art models like GPT-4o demonstrate better results when handling multiple instructions simultaneously rather than sequentially [27]. As observed in other fields, this unexpected improvement may stem from the model's "look-ahead" effect, where exposure to subsequent tasks provides contextual cues that help the model format its responses to earlier tasks more effectively. In a nutshell, the less one guides the agents, the better their performance.

The additional performance indicators reveal that ReACT+Summary is the most verbose, generating the highest token count. In fact, it also has the highest failure rate, averaging 4.5 aborted analyses, likely due to the increased token load that overloads the context window. Interestingly, the ReACT agent is also the most efficient (13.3 iterations), while the ReACT+Summary, requiring the most steps (23.3 iterations). This is again due to its verbosity that confuses the LLM.

VIII. Conclusion

In this paper, we introduced CFA-bench, a novel benchmark designed to evaluate LLM-powered agents in cybersecurity forensic tasks, addressing two fundamental research questions in this domain.

We demonstrated that a structured benchmark incorporating real-world forensic scenarios provides an effective evaluation framework. By supplying agents with authentic PCAP files from controlled testbed environments and establishing clear, objective evaluation criteria across four dimensions reflecting different levels of understanding, our methodology enables a rigorous assessment of forensic capabilities in LLMs – an aspect largely absent or implicit in previous benchmarks.

Our findings reveal interesting potential for LLM-based agents in cybersecurity forensics. While the preliminary agents we presented successfully analysed complex network traces and identified key aspects of security incidents, the best-performing agent correctly identifying the targeted service in only 42.3% of cases. In particular, the precise CVE detection remains a challenge, with the best agent succeeding in only 14.23% of cases. This emphasises the need for further advancements in security-specific reasoning, and offers opportunities to improve the agent design, e.g., empowering them with more tools.

A particularly unexpected outcome of our experiments was that the simpler ReACT architecture consistently outperformed more complex variants such as ReACT+Summary and Decoupled across most metrics. This counter-intuitive result aligns with emerging research suggesting that state-of-the-art LLMs perform better when processing multiple instructions simultaneously rather than sequentially.

Future research should focus on enriching forensic context by integrating detailed system logs and environmental metadata. Expanding evaluations to encompass diverse network traffic patterns, including benign activity, will provide a more comprehensive assessment of agent performance. Additionally, improving information retrieval mechanisms will be critical to supplying agents with high-quality data, ultimately enhancing their decision-making and investigative accuracy. These efforts will pave the way for more robust and reliable AI-driven forensic analysis in real-world cybersecurity scenarios.

Ethics

This research adheres to strict ethical guidelines to ensure that all data collection, handling, and analysis are performed responsibly and transparently. All data for this study are collected within a controlled laboratory environment and involve no human intervention. Network traces, application logs, and other data are generated exclusively in this lab setting, where no real-world systems or user data are involved. Simulated attacks occur in isolated virtual environments, ensuring that no actual users or production systems are affected. Since all data are generated under controlled conditions, there is no exposure to personal or sensitive information. All network traffic and logs are anonymized, and no identifiable information is included.

Acknowledgment

The research leading to these results has been partly funded by the Huawei Technologies, by the projects ACRE (AI-Based Causality and Reasoning for Deceptive Assets—2022EP2L7H) and xInternet (eXplainable Internet—20225CETN9), funded by the European Union - Next Generation EU within the PRIN 2022 program (D.D. 104 - 02/02/2022 Ministero dell'Università e della Ricerca). This manuscript reflects only the authors' views and opinions and the Ministry cannot be considered responsible for them.

References

[1] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023.

[2] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th annual acm symposium on user interface software and technology*, 2023, pp. 1–22.

[3] M. L. Siddiq and J. C. S. Santos, "Securityeval dataset: mining vulnerability examples to evaluate machine learning-based code generation techniques," in *Proceedings of the 1st International Workshop on Mining Software Repositories Applications for Privacy and Security*, ser. MSR4P&S 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 29–33. [Online]. Available: https://doi.org/10.1145/3549035.3561184

[4] C. Tony, M. Mutas, N. E. D. Ferreyra, and R. Scandariato, "Llmseceval: A dataset of natural language prompts for security evaluations," 2023. [Online]. Available: https://arxiv.org/abs/2303.09384

[5] H. Guo, J. Yang, J. Liu, L. Yang, L. Chai, J. Bai, J. Peng, X. Hu, C. Chen, D. Zhang, xu Shi, T. Zheng, liangfan zheng, B. Zhang, K. Xu, and Z. Li, "OWL: A large language model for IT operations," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=SZOQ9RKYJu

[6] Y. Miao, Y. Bai, L. Chen, D. Li, H. Sun, X. Wang, Z. Luo, Y. Ren, D. Sun, X. Xu, Q. Zhang, C. Xiang, and X. Li, "An empirical study of netops capability of pre-trained large language models," 2023. [Online]. Available: https://arxiv.org/abs/2309.05557

[7] G. Li, Y. Li, W. Guannan, H. Yang, and Y. Yu, "Seceval: A comprehensive benchmark for evaluating cybersecurity knowledge of foundation models," https://github.com/XuanwuAI/SecEval, 2023.

[8] Z. Liu, "Secqa: A concise question-answering dataset for evaluating large language models in computer security," 2023. [Online]. Available: https://arxiv.org/abs/2312.15838

[9] R. Tian, Y. Ye, Y. Qin, X. Cong, Y. Lin, Y. Pan, Y. Wu, H. Hui, W. Liu, Z. Liu, and M. Sun, "Debugbench: Evaluating debugging capability of large language models," 2024. [Online]. Available: https://arxiv.org/abs/2401.04621

[10] N. Tihanyi, M. A. Ferrag, R. Jain, T. Bisztray, and M. Debbah, "Cybermetric: A benchmark dataset based on retrieval-augmented generation for evaluating llms in cybersecurity knowledge," 2024. [Online]. Available: https://arxiv.org/abs/2402.07688

[11] Y. Liu, C. Pei, L. Xu, B. Chen, M. Sun, Z. Zhang, Y. Sun, S. Zhang, K. Wang, H. Zhang, J. Li, G. Xie, X. Wen, X. Nie, M. Ma, and D. Pei, "Opseval: A comprehensive it operations benchmark suite for large language models," 2024. [Online]. Available: https://arxiv.org/abs/2310.07637

[12] K. Alrashedy, A. Aljasser, P. Tambwekar, and M. Gombolay, "Can llms patch security issues?" 2024. [Online]. Available: https://arxiv.org/abs/2312.00024

[13] D. Donadel, F. Marchiori, L. Pajola, and M. Conti, "Can llms understand computer networks? towards a virtual system administrator," 2024. [Online]. Available: https://arxiv.org/abs/2404.12689

[14] D. Bhusal, M. T. Alam, L. Nguyen, A. Mahara, Z. Lightcap, R. Frazier, R. Fieblinger, G. L. Torales, B. A. Blakely, and N. Rastogi, "Secure: Benchmarking large language models for cybersecurity," 2024. [Online]. Available: https://arxiv.org/abs/2405.20441

[15] P. Jing, M. Tang, X. Shi, X. Zheng, S. Nie, S. Wu, Y. Yang, and X. Luo, "Secbench: A comprehensive multi-dimensional benchmarking dataset for llms in cybersecurity," 2024.

[16] L. Gioacchini, M. Mellia, I. Drago, A. Delsanto, G. Siracusano, and R. Bifulco, "Autopenbench: Benchmarking generative agents for penetration testing," 2024. [Online]. Available: https://arxiv.org/abs/2410.03225

[17] M. I. Hossen, J. Zhang, Y. Cao, and X. Hei, "Assessing cybersecurity vulnerabilities in code large language models," 2024. [Online]. Available: https://arxiv.org/abs/2404.18567

[18] L. Muzsai, D. Imolai, and A. Lukács, "Hacksynth: Llm agent and evaluation framework for autonomous penetration testing," 2024.

[19] J. Yang, A. Prabhakar, S. Yao, K. Pei, and K. R. Narasimhan, "Language agents as hackers: Evaluating cybersecurity skills with capture the flag," in *Multi-Agent Security Workshop@ NeurIPS'23*, 2023.

[20] L. Gioacchini, G. Siracusano, D. Sanvito, K. Gashteovski, D. Friede, R. Bifulco, and C. Lawrence, "Agentquest: A modular benchmark framework to measure progress and improve llm agents," 2024.

[21] Information Security Institute, CSIC, "CSIC 2010 HTTP Dataset for Web Attack Detection," 2010, accessed: 2025-02-17. [Online]. Available: https://www.isi.csic.es/dataset/

[22] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*. SciTePress, 2018, pp. 108–116.

[23] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," *Proceedings of the 2nd IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pp. 1–6, 2009.

[24] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 3971–3988.

[25] FIRST - Forum of Incident Response and Security Teams, "Common Vulnerability Scoring System (CVSS)," 2024, accessed: 2025-02-17. [Online]. Available: https://www.first.org/cvss/

[26] T. R. Sumers, S. Yao, K. Narasimhan, and T. L. Griffiths, "Cognitive architectures for language agents," *arXiv preprint arXiv:2309.02427*, 2023.

[27] G. Son, S. Baek, S. Nam, I. Jeong, and S. Kim, "Multi-task inference: Can large language models follow multiple instructions at once?" *arXiv preprint arXiv:2402.11597*, 2024.

[28] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2022.

[29] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[30] Y. Zhang, Y. Li, L. Cui, D. Cai, L. Liu, T. Fu, X. Huang, E. Zhao, Y. Zhang, Y. Chen *et al.*, "Siren's song in the ai ocean: a survey on hallucination in large language models," *arXiv preprint arXiv:2309.01219*, 2023.

[31] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[32] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

[33] N. Reimers, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

APPENDIX

A. Information Retriever

Recent research has highlighted the unreliability of LLMs, which are prone to generating hallucinations [29], [30]. To address this issue, Retrieval-Augmented Generation (RAG) [31] has been introduced. RAG leverages a smaller language model to assess the relevance of textual information in relation to a given query, appending the most pertinent content to enrich the query. This additional context helps condition the LLM to generate more accurate and reliable responses. In this work, we encourage the LLM to utilize this tool to retrieve relevant information for task completion (e.g., querying the web for Common Vulnerabilities and Exposures (CVEs) potentially associated with a malicious attack). The retrieval process consists of the following steps:

1) *Document Retrieval*: Given an initial query (e.g., "CVEs related to path traversal"), a set of relevant documents is retrieved from the web. The LLM generates a refined query based on the provided causal query, optimizing it for search engine retrieval. In our experiments, we used the DuckDuckGo search engine to extract content from the top 10 web pages. Each retrieved webpage is then segmented into smaller chunks using a sliding window approach, sampling a 512-token segment every 256 tokens.

2) *Ranking*: The LLM refines the query using a *query transformation* approach [32] to enhance semantic alignment with relevant document chunks. The retrieved chunks and the transformed query are then processed by a sentence transformer [33], specifically the `multi-qa-mpnet-base-dot-v1` model. Cosine similarity is computed between the embedded transformed query and each chunk, ranking them based on relevance.
3) *Context Integration*: The LLM generates a response to the causal query using the additional context retrieved in the previous steps. Specifically, the top five chunks with the highest cosine similarity scores are selected to provide relevant and supportive information, improving the accuracy of the generated response.