CyberGuardian 2: Integrating LLMs and Agentic AI Assistants for Securing Distributed Networks

Ciprian Paduraru[1], Catalina Camelia Patilea[1] and Alin Stefanescu[1,2]

[1]*Department of Computer Science, University of Bucharest, Romania*

[2]*Institute for Logic and Data Science, Romania*

Keywords: Large Language Models, Cybersecurity Assistant, Security Officers, Agentic AI, Fine-Tuning, Retrieval Augmented Generation.

Abstract: Robust cybersecurity measures are essential to protect complex information systems from a variety of cyber threats, which requires sophisticated security solutions. This paper explores the integration of Large Language Models (LLMs) to improve cybersecurity operations within Security Operations Centers (SOCs). The proposed framework has a modular plugin architecture where Agentic AI controls the information flow, includes Retrieval Augmented Generation (RAG), protection methods for human-chatbot interactions and tools for managing tasks such as database interactions, code generation and execution. By utilizing these techniques, the framework aims to streamline the workflows of SOC analysts, allowing them to focus on critical tasks rather than redundant activities. The study also explores the dynamic customization of LLMs based on client data, user experience, potential risks and language preferences to ensure a user-centric approach. The results show improvements in efficiency and effectiveness and highlight the potential of LLMs in cybersecurity applications.

1 INTRODUCTION

Large Language Models (LLMs) have shown remarkable potential in areas such as natural language processing, machine learning, and, more recently, cybersecurity: (Motlagh et al., 2024), (Shafee et al., 2024), (Paduraru et al., 2024). Our research explores the application of these advances in the development of interactive assistants that support security professionals by automating and optimizing the data collection processes and understanding for problem identification and providing informed guidance for problem remediation. Organizations and institutions have set up specialized teams with different areas of expertise to combat threats in real-time. One notable example is the Security Operations Center (SOC) (Mughal, 2022), where SOC analysts work around the clock. Their tasks include monitoring and detecting threats in real-time, investigating and escalating incidents, and managing security tools and information. They work with network engineers and architects to improve infrastructure protection against potential future attacks.

Contributions. Our research aims to leverage the latest advances in LLMs to support security professionals in real-time. The goal is to streamline their workflows so that they can focus on critical tasks rather than redundant and time-consuming activities. The goal is to allow customers to interact with systems, logs, databases, code generation and execution, user interface components, and internal tools using the provided methods via natural language queries, with the assistant acting as an intermediary. The contributions of this work can be summarized as follows, as an improvement of our previous work (Paduraru et al., 2024):

- Improving the process of fine-tuning the LLM model and datasets compared to our previous work. We further fine-tune the model from two different angles: a) doubling the original size of the cybersecurity-related dataset with new sources and b) matching the answers generated by the assistant with the answers annotated by humans using the Direct Preference Optimization (DPO) (Guo et al., 2024) technique and human-in-the-loop. On the technical level, we fine-tune the base model from scratch, moving from Llama-2 (Touvron et al., 2023) to the latest improvements in Llama-3.1 (Dubey et al., 2024).
- The problems reported in our previous work were solved by adding the new mechanisms of Adaptive-Retrieval Augmented Generation (RAG) (Jeong et al., 2024), (Asai et al., 2023), (Yan et al., 2024),

Paduraru, C., Patilea, C. C. and Stefanescu, A.
CyberGuardian 2: Integrating LLMs and Agentic AI Assistants for Securing Distributed Networks.
DOI: 10.5220/0013406000003928

In *Proceedings of the 20th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2025)*, pages 660-667
ISBN: 978-989-758-742-9; ISSN: 2184-4895

to allow the model to self-check whether the retrieved data is relevant to the user's prompt, evaluate the quality of the generated response, including the hallucination validator, and iterate to improve it. In addition, the reiteration of the response now includes mechanisms that allow for variety and novelty in the data retrieved from queries instead of a greedy selection.

- Our work architecturally reworks the previous project to be more independent and customized by using a plugin architecture: clients are now able to inject their fine-tuning data, data for RAG, tools, and validators into an Agentic AI (Khanda, 2024) flow.

Reusable open-source code and a dataset collection are provided at https://github.com/unibuc-cs/CyberGuardian.

This article is structured as follows. The next section presents works that have inspired our project or that we have adapted to our specific context and use cases. Section 3 describes the methods of data collection and processing. Section 4 details the proposed architecture, methods, components and their interactions. The evaluation and discussions are presented in Section 5. Finally, the last section concludes the paper and outlines future work plans.

2 RELATED WORK

Recent research has investigated the use of LLMs in both defensive and adversarial cybersecurity contexts, offering a thorough review of contemporary practices and identifying research gaps. The extensive evaluation by (Motlagh et al., 2024) provides valuable insights into the potential risks and advantages of incorporating LLMs into cybersecurity strategies. This section will outline the primary sources of inspiration for the present study.

In their study, (Al-Hawawreh et al., 2023) explore the potential applications of ChatGPT in the field of cybersecurity, especially in penetration testing and threat defense. They present a case study in which ChatGPT is used to create and execute false data injection attacks on critical infrastructure, such as industrial control systems, and discuss the challenges and future directions for its integration into cybersecurity. Other studies, such as those by authors referenced in (Franco et al., 2020) and (Shaqiri, 2021), utilize conventional NLP techniques to aid in high-level cybersecurity planning and management. These studies focus on identifying logs of past cyberattacks, proposing solutions, and providing insights for decision making through user interactions that lead to the extraction and deployment of various solutions.

In (Arora et al., 2023), the authors develop a chatbot that uses AI and sentiment analysis of Twitter data to predict and assess cyber threats on social media. (Tejonath Reddy, 2024) examines chatbots that use deep learning as a proactive solution for detecting persistent threats and phishing attacks in real-time and concludes that deep learning algorithms can adapt to new paradigms and detect subtle phishing indicators, creating a stronger defense against evolving threats. The work by (Abdelhamid and et al., 2023) proposes chatbots integrated with social networks as collaborative agents for continuous cybersecurity awareness, education, and training. Similarly, another study in (Fung et al., 2022) presents a cybersecurity chatbot built on Google Dialogflow that educates users about cyber risks, provides a knowledge base, self-quizzes and advice, effectively improving cybersecurity awareness.

Conversely, ensuring data security in chatbot interactions is an important issue. For example, the research and demonstration applications we propose contain sensitive user data and introduce users to important system functions without violating current best practices. As chatbots become increasingly popular in various fields, they pose significant security risks and vulnerabilities. (Yang et al., 2023) has conducted a systematic literature review to identify potential threats, propose effective solutions, and outline promising future research directions.

3 DATASETS COLLECTION AND INDEXING

This section shows how the data sources for fine-tuning the LLM model were collected, post-processed, and indexed for faster retrieval. The overview of the process is shown in Figure 1 and is explained below. The principles in (Paduraru et al., 2024) are followed to collect and index the data to align the base model with foundational knowledge from the latest data and research in the field of cybersecurity. A formalized definition of this dataset can be found in Eq. (1). It consists of documents, video transcripts, and markdown post-processed data.

$$\mathcal{D} = \{PDF_k, Transcript(Video_i), Markdown_j\} \quad (1)$$

We follow the principles of Direct Preference Optimization (DPO) (Guo et al., 2024) to improve the performance of the model by aligning the response of the smaller model trained with both a teacher model and a human-in-the-loop.

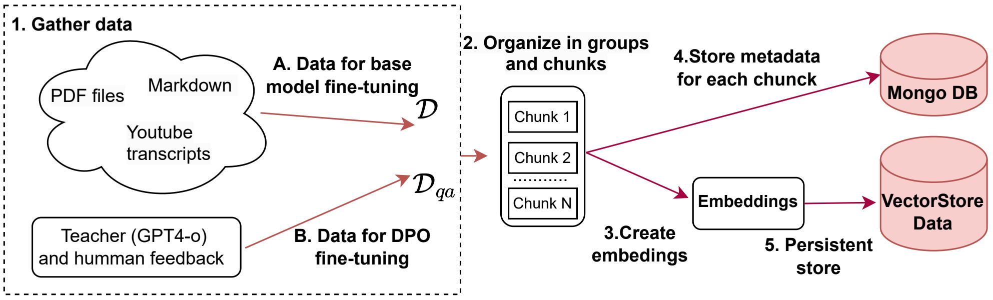

Figure 1: The data collection process. First, the data set, $\mathcal{D}$, is obtained from various sources of information that can be exported in text format (Step 1.A). Then, the GPT-4o model is used as a teacher to generate pairs of questions, answers and ideal answers. People randomly sample from this dataset to annotate the ideal answers from their point of view, resulting in the dataset $\mathcal{D}_{qa}$ (Step 1.B). In addition, the metadata for both datasets is stored on a MongoDB server to simplify the update process and avoid deduplication (Step 4). Finally, the data is split into smaller parts, embedded and permanently stored in a vector database (Steps 3 and 5). The red colored blocks represent the output of this process.

A teacher model(OpenAI et al., 2024), in particular the GPT-4o[1], is used to collect cybersecurity data by extracting initial N_{qdoc} questions from a subset of $\mathcal{D}$ documents (especially from foundational books and recent tutorials or blogs) using prompts as described in (Ivison et al., 2024)). The same model is also asked to create N_{qbase} questions on various basic sub-areas of cybersecurity (see Section 5) that were not part of the first set, resulting in a total of $N_q = N_{qbase} + N_{qdoc}$ questions. The teacher model is then asked to respond to each of these questions with $n_{choices} = 4$ different versions, using a temperature of 0.8. Finally, we ask the teacher again to look at each question and answer and choose the best one from its point of view. We further denote this set of questions and answers by $\mathcal{D}_{qa}$.

Following the work in (Tihanyi et al., 2024), we then use three human skill levels (beginner, intermediate, senior) to check $N_{qhumansP}$ percent of the set $\mathcal{D}_{qa}$ and select which version out of the $n_{choices}$ answers they would prefer. Intuitively, a senior person might prefer shorter, concise answers without too much detail, as opposed to beginners who might want more information. Each person was given a random subset of the original and had access to a random source of information when answering. The results are summarized in the set $\mathcal{D}_{qa_H} \subset \mathcal{D}_{qa}$ and sorted by expertise in $\mathcal{D}^{Hlevel}_{qa_H} \subset \mathcal{D}_{qa}$. Including the human in the loop ensures that the dataset incorporates human feedback into the training loop to refine the model's understanding of what constitutes a preferred response. For efficiency reasons, the datasets are stored in an embedded format, which we refer to as $\mathcal{D}_{emb}$, respectively $\mathcal{D}_{qa_{emb}}$. In our implementation, we use the sentence transformer embedding model (Reimers and Gurevych, 2019). Specifically, we use the open source variant *all-mpnet-base-v2*[2]. Its purpose is to transform text data into float value vectors, a necessary format for subsequent processing by Faiss' indexing and querying system.

4 ARCHITECTURE AND IMPLEMENTATION

The architecture and components are implemented using a plugin pattern, allowing higher capacity models, alternative fine-tuning methods, various functionalities, and components to be toggled on or off based on specific use cases, as shown in Figure 2. This approach, known in software engineering as the separation of concerns, is a core concept applied in the proposed methods.

Client-End Data Customization. The client can add its own specific data to control the assistant according to the use case. For example, the model can be fine-tuned with SFT to more easily adapt to a specific use case terminology, language, and culture within its organization. RAG data can be specified such that the assistant can retrieve information that needs to be exact or fast to get, e.g., manuals of products, architecture of the infrastructure, high-level APIs, and frameworks available inside the system. Data can be organized in groups, each with its own metadata to follow the trend of hierarchical information retrieval (Goel and Chandak, 2024). Tools are specified such that the assistant can connect reasoning with actions performed on the client's infrastructures. A concrete example can be: suggesting SQL source code to blacklist an IP range of addresses in the firewall's

[1] https://openai.com/index/hello-gpt-4o/

[2] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

database, provision new virtual machines, add new users to a group, etc. In general, these are operations that are implemented in the client's source code, and the assistant has the information on all of these at the description or prototype level. This concept is discussed further in the paper and is known as Agentic AI.

RAG Support. To efficiently summarize or extract content from local system infrastructures such as logs, databases, and source code, a retrieval augmented generation (RAG) functionality is used. This reduces human effort and is important for users, particularly SOC teams, facing constantly updated internal knowledge. Various types of information and formats can be indexed and queried during the LLM reasoning process, including internal documents (e.g., whitepapers, communication, architecture, and infrastructure documentation), source code APIs and configuration files, and data sources like databases, JSON or CSV files. To ensure diverse information retrieval between iterations, the framework uses the MMR (Carbonell and Goldstein, 1998) sampling algorithm. Introducing noise into the RAG process helps address issues reported in previous studies (Paduraru et al., 2024), where users might prefer different data retrieved after being dissatisfied with the initial response.

User Preference and Conversation Safeguarding. In general, organizations have specific cultural communication rules. More, users may have diverse preferences when interacting with an assistant, such as wanting concise or detailed responses or preferring formal or polite tones. These preferences can be managed both through prompt engineering and fine-tuning for specific groups. Also, users may have varying levels of expertise, which can be matched to specific fine-tuned models. This is illustrated in Figure 3. For security measures, as prototyped in our framework, they can be registered in an Access Control List (ACL), granting different levels of access to internal resources. For example, some users may have the privilege to execute code or initiate services (e.g., a honeypot). To ensure that assistant conversations adhere to various rules (e.g., gender-neutral language, politeness), the Llama Guard model (Inan et al., 2023) filter is utilized, along with its commonly used taxonomy in LLMs, protecting conversations by enforcing these guidelines.

The Fine-Tuning Process. The process for fine-tuning a base model to better align with cybersecurity knowledge and related tasks is illustrated in Figure 3. This workflow and the source code are independent of the chosen base input model; for our prototype and evaluation, the Llama 3.1-8B-Instruct (Dubey et al., 2024) was used along with the datasets presented in Section 3. The final version of the model, *CyberGuardian2LLM*, is split into three categories based on expertise level. This example is generic and can be adapted according to user needs. With larger preference datasets, the methods can also be tailored to other factors, such as the text style of assistant responses and combinations thereof.

Base Fine-Tuning. For this step, we follow the standard supervised training (SFT) methodology commonly used in previous research. The model processes batches of dataset samples and compares them to the corresponding ground truth using cross-entropy as a loss function. At predefined intervals, the model's performance is evaluated using the perplexity metric.

```
$Sample\,batch\,\mathcal{B} \sim \mathcal{D}_{emb}$ # containing inputs and desired model
      answers
# Pass this through the model to get the predicted answers
$Predictions = CyberGuardian2LLM\,(\mathcal{B}\,[''Input''])$
# Loss represents how close the output of the model is
      compared to the ground truth in the batch
$Loss = CrossEntropy\,(Predictions, \mathcal{B}\,[''Answer''])$
$Update\,weights\,of\,CyberGuardian2LLM$
```

Listing 1: Fine-tuning process.

Direct Responses and Preferences Fine-Tuning. The loss function of DPO (Chen et al., 2024) is adapted to our use case. Specifically, during the fine-tuning process sample pairs of prompts (x), chosen and rejected answers (y_c, respectively y_r) are sampled from dataset $\mathcal{D}_{qa}$. These are passed through two models: a) the base model, π_{base}, being the trained model in the previous step $CyberGuardian2LLM_{base}$), and b) the model under current fine-tuning π_θ, representing the output model $CyberGuardian2LLM$.

Conversation Management. The observations from the work of (Maharana et al., 2024) are followed to avoid losing the conversation context over many iterations. The conversation history between the user and assistant is handled using two structures:

- The pair of last 5 messages exchanged in the conversation between them: short-term memory, Mem_{sh}.
- The conversation summary of all other older messages condensed into a single paragraph: the long memory, Mem_l.

Adaptive RAG. Our methods follow the ideas from (Jeong et al., 2024), in using RAG in combination with the AI agent to check two main things:

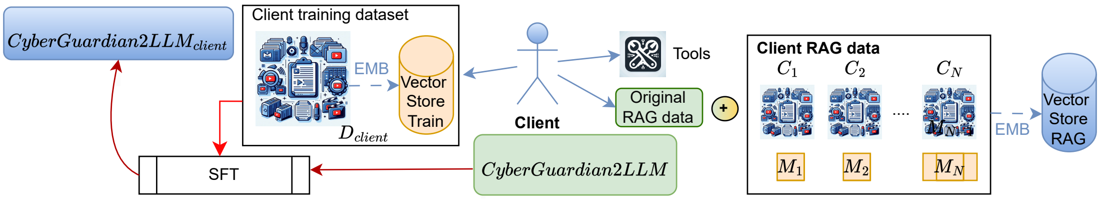

Figure 2: Adding client specifics to the assistant knowledge. Three types of data can be added to control completely the answers generations: the base fine-tuning data, RAG documents, and tools to interact with one infrastructure. The blue colored components represent the output of adding the client-specific data into the previously fine-tuned model and RAG support.

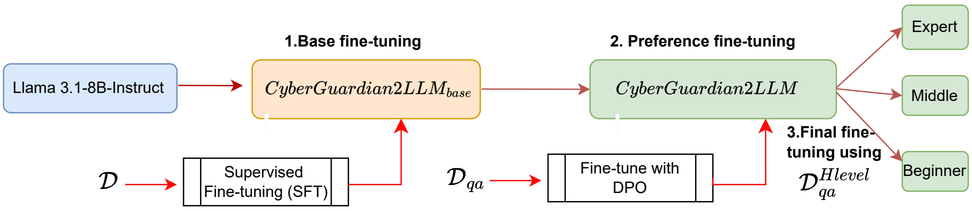
1.Base fine-tuning
2. Preference fine-tuning
Llama 3.1-8B-Instruct
$CyberGuardian2LLM_{base}$
$CyberGuardian2LLM$
Expert
Middle
Beginner
3.Final fine-tuning using
$\mathcal{D}_{qa}^{Hlevel}$
$\mathcal{D}$
Supervised Fine-tuning (SFT)
$\mathcal{D}_{qa}$
Fine-tune with DPO

Figure 3: The process of fine-tuning the Llama 3.1-8B-Instruct (as an example) to a model aligned more on the cybersecurity field. There are two main steps in this process: 1) the base model fine-tuning, and 2) aligning with a teacher and human-in-the-loop. The final models are slightly fine-tuned (Step 3) for different specifics, in our demonstration, to consider the level of expertise of the human users.

- Is the retrieved information relevant to the user question? This is needed specifically because the retrieval mechanisms tend to score high enough pieces of documents that might not be relevant. Thus, the LLM is used as a judge by using its natural language understanding capabilities.
- Can get the information from other sources, for example using existing tools registered by the client (e.g., web-search, internal platform search, etc.), or even human-in-the-loop, by reporting that no sources were found and asking for help.

Due to space constraints, the prompts utilized are available in the repository. Briefly, the LLM is instructed to: (a) determine if the retrieved content is relevant to the question (binary response: yes or no); (b) perform a hallucination check by assessing whether the final generated answer is grounded in the provided documents; and (c) evaluate the quality of the response by ascertaining if it adequately addresses the original user's question.

Tools and Interaction with the Backend Systems. To enhance SOC specialists' productivity and response times, our work utilizes advanced methods focusing on user interaction with LLM and system processes. These methods, referred to as *agents* and *tools*. ReACT agents (Yao et al., 2023) are the core of our Agentic AI implementation. This method involves first prompt engineering to provide the LLM with brief descriptions of available tools. The LLM model is fine-tuned over a few epochs using 50 examples of tool calls and parameter extraction. The main steps are: act (call specific tools), observe (pass tool output back to the model), and reason (determine the next action based on tool output). Figure 4 shows how the LLM interacts with various tools in the framework. At each step, the LLM evaluates the problem, the available tools, and decides whether to invoke a tool that interacts with the organization's infrastructure or provide a response. This method allows for *task decomposition*, where complex queries are broken down into manageable pieces and resolved using appropriate tools. The set of tools implemented in the source code as exemplification are commonly used such as ACL operation, databases access, office tools, python code generation (by invoking internally another specialized model, Code Llama 7B (Rozière et al., 2024)), and safety checker tool (invoking Llama Guard (Inan et al., 2023)).

5 EVALUATION

5.1 Evaluation Setup

To iterate over the architectural quality of the framework, and test Agentic AI and the RAG component, the solution presented was tested first with a group of students (master's degree, from the University of Bucharest). Each group was asked to simulate a dis-

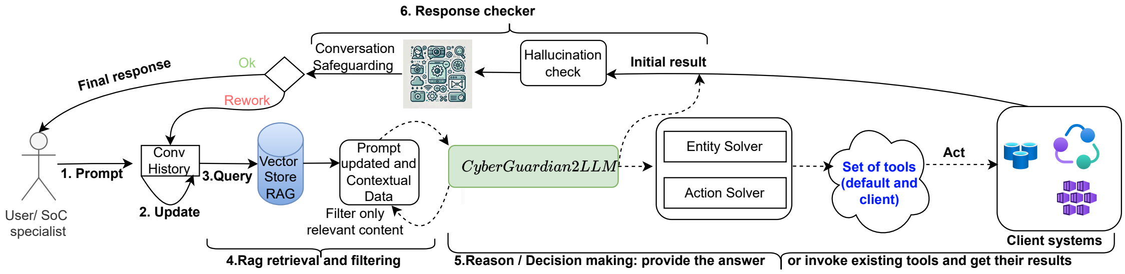
6. Response checker
Conversation Safeguarding
Hallucination check
Initial result
Final response
Ok
Rework
User/ SoC specialist
1. Prompt
Conv History
2. Update
3.Query
Vector Store RAG
Prompt updated and Contextual Data
Filter only relevant content
4.Rag retrieval and filtering
CyberGuardian2LLM
Entity Solver
Action Solver
Set of tools (default and client)
Act
Client systems
5.Reason / Decision making: provide the answer
or invoke existing tools and get their results

Figure 4: The flow of the Agentic AI: (1) receiving the user's prompt, (2) reformulating the current query based on the memories of previous conversations, (3) giving the reformulated prompt, retrieving and filtering relevant content from the indexed data, (4) deciding whether the answer can be given or must be given via the exposed services on the client side (using tools/function calls); the decision is based on LLM and some NLP-based processing techniques for easier identification of the entities and actions requested in the prompt, (5) checking the response for hallucinations and security; if the response matches the guidelines, returning it to the user and updating the internal knowledge; if it needs to be revised instead, add the reviewer's response to the history so that the model can assess what was wrong with the original response the next time. Note that while the framework repository only contains illustrative examples, the tool suite, models, and validators can be customized by each client site.

tributed application considering a use-case from different domains such as medical, gas stations, smart home systems, etc. In total, we counted 17 different applications across various domains. For each, the request was to gather at least 3-4 PDF manuals own made or from ithe nternet that could represent the client data for that particular use case. The target was to have $\sim$ 100 pages of data on each case to perform RAG tests. Each team of students represented the *Client* class in Figure 2. The focus from an architectural point of view was to make the methods reusable across use cases.

Table 1 shows different values that are used to fine-tune the current version of the model as mentioned in Section 3. The high temperature used when generating the responses from the teacher model, 0.8 was able to produce diverse answers, but still almost correct. In this sense, we followed the observations of (Ivison et al., 2024) which mentions that better preference data leads to the largest improvements from all other factors.

Table 1: Table containing the values used in Section 4. The first group represents the dataset dimensions, the second is the number of humans who participated in the experiments, while the retrieval parameters are presented in the last group.

Description	Value
Dataset	
N_{qdoc}	5000
N_{qbase}	4000
N_{qdoc}	1000
$N_{qhumansP}$	30%
No. of humans participating	83
$\mathcal{D}_{qa_H} beginners$	40
$\mathcal{D}_{qa_H} intermediate$	39
$\mathcal{D}_{qa_H} seniors$	4

5.2 Quantitative and Qualitative Evaluation

There are two research questions that we address in our study:

- **RQ1.** How well does the Cyberguardian2LLM understand the cybersecurity domain?
- **RQ2.** What is the real feedback of human users engaged in the experiments?

RQ1 Analysis. To quantitatively assess the effectiveness of the fine-tuned CyberGuardian2LLM, we measure the response usefulness using an established method (Zheng et al., 2023), involving automated evaluation with a larger model, specifically GPT-4, serving as the *judge*. This is also applied in the cybersecurity domain in the work of (Paduraru et al., 2024) and (Tihanyi et al., 2024), which datasets we also reuse in the process of fine-tuning our model.

Five topics in cybersecurity relevant to SOC specialists: have been selected:

- System protection against security risks and malware.
- Cryptography and authentication mechanisms.
- Configuring security protocols such as firewalls and intrusion detection systems (IDS).
- Network security infrastructure (including firewalls, VPNs, web proxies, IDS/IPS).
- Investigation of data breaches and leaks.

These clusters are denoted as $Topics = (tp_i)_{i=1,...5}$. The *judge*, GPT-4, is prompted to generate 20 questions for each of the five topics, totaling 100 questions, denoted by *Que*. This set is created by inserting the topic variable into the template shown in Listing 2.

```
Consider yourself an interviewer for a SOC specialist
     position. Ask 20 relevant. different questions on the
      topic {topic_var}
```

Listing 2: The template prompt used to ask questions to the judge LLM.

We measure the CyberGuardian2LLM responses for each $Q \in Que$ against two vanilla (no fine-tuned) Llama 3.1 models (Llama 3.1-8B-Instruct, and 70B-Instruct), and the model from (Paduraru et al., 2024). To compare the answers head-to-head, the *judge* is prompted to respond to which one he prefers, Listing 3. The template variable for each question Q_{var} is filled in, along with the responses of the two models compared, $LLM1\_resp$, and $LLM2\_resp$. The observation made from the results shown in Table 2, is that in the synthetic experiments, the CyberGuardian2LLM's answers are preferable over the previous ones, except the bigger class model (70B) which needs significantly more computational resources to infer (GPU memory: $\sim 9.6GB$ vs $\sim 84GB$). The results shown suggest that a fine-tuned model can overpass or come close to the much larger (sometimes impractical for end-user machines), as reported also in literature (Yao et al., 2023).

Table 2: Head-to-head comparison of response preferences, with GPT-4 serving as the evaluator. The second column shows the percentage of cases where each model was preferred over CyberGuardian2LLM. The Llama versions shown in the first two rows are the public vanilla versions, without any fine-tuning.

Model	Preferred over CyberGuardian2LLM
Llama3.1-8B	29%
Llama3.1-70B	61%
CyberGuardianLLM	33%

```
Given the question: {Q_var}, respond which of the following
     two answers you prefer. Write only Version1 or
     Version2.
# Version1: {LLM1_resp}.
# Version2: {LLM2 resp}.
```

Listing 3: The template for classifying the answers with judge LLM.

RQ2 analysis. To evaluate from a different perspective, during each case, the assistant emulated signals that could indicate such an attack. The 83 human participants were asked to use the assistant to solve the issue. Solving these issues involves things as such: finding the flooding IPs, blacklisting them within the firewall database, starting a honeypot server, analyzing data flow specifics in terms of size and procurance, and so on. An emulator for different types of attacks, including DDoS or ransomware attacks (de Neira et al., 2023), has been used for evaluation. The snapshots of the attacks provided various data tables with statistics on the utilization of resources (e.g. servers, routers, local hub systems), and connection logs of users including their location, time, and resources consumed in the network. The deployment interface was handled via RestAPI and the client was implemented with Streamlit [3] libraries and tools (including visualizations). All these simulations and screenshots can be found in the repository.

The task was correctly solved by 65 people within a single instance of the test. The rest had to restart the test and try once or twice. Their feedback after each question-response pair is collected using a rating score 1-5 and optional natural language description. The average score of responses is 4.17. One important observation is that users considered that the cybersecurity assistant helped them in two ways:

- they can ask contextual items around the topic that is trying to solve.
- the functionalities of each use-case were faster retrieved (RAG), and actions could be taken using natural language requests.

6 CONCLUSIONS AND FUTURE WORK

The CyberGuardian framework demonstrates the potential to enhance the efficiency and effectiveness of Security Operations Centers (SOCs) through the application of LLMs and Agentic AI. By leveraging techniques such as Retrieval Augmented Generation (RAG) and ensuring secure human-chatbot interactions, CyberGuardian 2 addresses various cybersecurity tasks, including database management, firewall configuration, and code execution. The framework's adaptability to different user experience levels and its client-independent architecture make it a versatile tool for diverse applications. Future work will focus on further fine-tuning the model, expanding its capabilities, and integrating more complex multi-agent functions to support advanced cybersecurity operations.

ACKNOWLEDGEMENTS

This research was partially supported by the project "Romanian Hub for Artificial Intelligence - HRIA",

[3] https://streamlit.io/

Smart Growth, Digitization and Financial Instruments Program, 2021-2027, MySMIS no. 334906 and European Union's Horizon Europe research and innovation programme under grant agreement no. 101070455, project DYNABIC.

REFERENCES

Abdelhamid, S. and et al. (2023). Cybersecurity awareness, education, and workplace training using socially enabled intelligent chatbots. In *Creative Approaches to Technology-Enhanced Learning for the Workplace and Higher Education*, pages 3–16, Cham. Springer Nature Switzerland.

Al-Hawawreh, M., Aljuhani, A., and Jararweh, Y. (2023). ChatGPT for cybersecurity: practical applications, challenges, and future directions. *Cluster Computing*, 26(8):3421–3436.

Arora, A., Arora, A., and McIntyre, J. (2023). Developing chatbots for cyber security: Assessing threats through sentiment analysis on social media. *Sustainability*, 15(17).

Asai, A. et al. (2023). Self-rag: Learning to retrieve, generate, and critique through self-reflection.

Carbonell, J. and Goldstein, J. (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proc. of ACM SIGIR*, pages 335–336.

Chen, C., Liu, Z., Du, C., Pang, T., Liu, Q., Sinha, A., Varakantham, P., and Lin, M. (2024). Bootstrapping language models with dpo implicit rewards.

de Neira, A. B., Kantarci, B., and Nogueira, M. (2023). Distributed denial of service attack prediction: Challenges, open issues and opportunities. *Computer Networks*, 222:109553.

Dubey, A. et al. (2024). The llama 3 herd of models.

Franco, M. F. et al. (2020). Secbot: a business-driven conversational agent for cybersecurity planning and management. In *2020 16th International Conference on Network and Service Management (CNSM)*, pages 1–7.

Fung, Y.-C. et al. (2022). A chatbot for promoting cybersecurity awareness. In *Cyber Security, Privacy and Networking*, pages 379–387, Singapore. Springer Nature Singapore.

Goel, K. and Chandak, M. (2024). Hiro: Hierarchical information retrieval optimization. *CoRR*, abs/2406.09979.

Guo, S. et al. (2024). Direct language model alignment from online ai feedback.

Inan, H., Upasani, et al. (2023). Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*.

Ivison, H. et al. (2024). Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback.

Jeong, S. et al. (2024). Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity.

Khanda, R. (2024). Agentic ai-driven technical troubleshooting for enterprise systems: A novel weighted retrieval-augmented generation paradigm.

Maharana, A. et al. (2024). Evaluating very long-term conversational memory of LLM agents. In *Proceedings of ACL*, pages 13851–13870, Bangkok, Thailand. Association for Computational Linguistics.

Motlagh, F. N. et al. (2024). Large language models in cybersecurity: State-of-the-art.

Mughal, A. A. (2022). Building and securing the modern security operations center (soc). *International Journal of Business Intelligence and Big Data Analytics*, 5(1):1–15.

OpenAI et al. (2024). GPT-4 technical report.

Paduraru, C., Patilea, C., and Stefanescu, A. (2024). CyberGuardian: An interactive assistant for cybersecurity specialists using large language models. In *Proc. of ICSOFT'24*, volume 1, pages 442–449. SciTePress.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Rozière, B. et al. (2024). Code llama: Open foundation models for code.

Shafee, S., Bessani, A., and Ferreira, P. M. (2024). Evaluation of llm chatbots for osint-based cyber threat awareness.

Shaqiri, B. (2021). Development and refinement of a chatbot for cybersecurity support. Master's thesis, University of Zurich, Zurich, Switzerland.

Tejonath Reddy, K. (2024). How deep learning chatbots empower cybersecurity against phishing attacks. *International Center for AI and Cyber Security Research and Innovations (CCRI)*.

Tihanyi, N., Ferrag, M. A., Jain, R., Bisztray, T., and Debbah, M. (2024). Cybermetric: A benchmark dataset based on retrieval-augmented generation for evaluating llms in cybersecurity knowledge. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 296–302.

Touvron, H. et al. (2023). Llama 2: Open foundation and fine-tuned chat models.

Yan, S.-Q. et al. (2024). Corrective retrieval augmented generation.

Yang, J. et al. (2023). A systematic literature review of information security in chatbots. *Applied Sciences*, 13(11).

Yao, S. et al. (2023). ReAct: Synergizing reasoning and acting in language models. In *ICLR'23*.

Zheng, L. et al. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS*, volume 36, pages 46595–46623.