# CyberGuardian: An Interactive Assistant for Cybersecurity Specialists Using Large Language Models

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, Information Security, Cybersecurity Assistant, Security Officers, Llama, Fine-Tuning.

Abstract: Cybersecurity plays an important role in protecting people and critical infrastructure. Sectors such as energy, defense and healthcare are increasingly at risk from cyber threats. To address these challenges, dedicated Security Operations Centers (SOCs) continuously monitor threats and respond to critical issues. Our research focuses on the use of Large Language Models (LLMs) to optimize the tasks of SOCs and to support security professionals. In this work, we propose a framework, which we call CyberGuardian, whose main goal is to provide a chatbot application along with a set of tools to support SOC analysts in real-time cybersecurity tasks. We use state-of-the-art LLM techniques and start from a Llama 2 model, then fine-tune the base model using a new dataset containing mainly cybersecurity topics. The CyberGuardian framework has a plugin architecture that integrates processes such as Retrieval Augmented Generation (RAG), safeguard methods for interaction between human user and chatbot, integration with tools to manage tasks such as database interactions, code generation and execution, and plotting graphs just by specifying the task in a natural language. The work, along with the dataset we collected and reusable code to update or customize, is made available to the cybersecurity community as open source.



## 1 INTRODUCTION

In the context of our increasingly digitalized global environment, the importance of cybersecurity to protect individuals, organizations and critical infrastructure cannot be overstated. A variety of sectors and applications are increasingly becoming the focus of cybercriminal activity.

In response to these real-time threats, companies and institutions have set up specialized teams with different expertise. A typical team that works around the clock is called a Security Operations Center (SOC) (Mughal, 2022), whose members are known as SOC analysts. Their responsibilities include monitoring and detecting threats in real time, investigating incidents and escalating them to various stakeholders, and managing security tools and information. They often work with network engineers and architects to improve infrastructure protection against potential future attacks. Large language models (LLMs) are increasingly being used in numerous applications due to their high natural language processing (NLP) capabilities, allowing them to perform a variety of tasks.

**Contributions.** The goal of our research is to utilize the latest methods and tools in the field of large language models (LLMs) to support these security experts and roles through a real-time chatbot application. The goal is to streamline their work so that they can focus on critical aspects rather than redundant tasks that are often challenging. From this perspective, the methods developed allow users to interact with the systems, protocols, databases, code generation and execution, and internal tools using natural language requests, with the chatbot serving as the middle component. The framework developed is named *CyberGuardian*.

The contributions of this work can be summarized as follows:

- To the best of our knowledge, this is the first open-source work that investigates the application of chatbots for SOC specialists to support cybersecurity in real time using large language models.
- The first public, experimentally released corpora (datasets) with up-to-date information on aspects of cybersecurity, including PDF documents, video transcripts and markdown content. The collection contains more than 3000 PDF papers and 8000



Paduraru, C., Patilea, C. and Stefanescu, A.
CyberGuardian: An Interactive Assistant for Cybersecurity Specialists Using Large Language Models.
DOI: 10.5220/0012811700003753
In *Proceedings of the 19th International Conference on Software Technologies (ICSOFT 2024)*, pages 442-449
ISBN: 978-989-758-706-1; ISSN: 2184-2833

video transcripts (mainly selected conference presentations, industry practice tutorials and tools). T

- Reusable open-source code and dataset collection scripts respository \url{https://github.com/unibuc-cs/CyberGuardian}) that composes the aforementioned components through an extensible, user-customizable plugin architecture.

The paper is organized as follows. The next section describes works that we have either used as a source of inspiration or reused by adapting them to our context and use cases. Section 3 explains the data collection processes implemented in the project and the fine-tuning of the LLMs based on them. Section 4 describes the proposed plugin architecture, the currently implemented components and the communication flow. The evaluation and discussions can be found in Section 5. Finally, the last section contains conclusions and a plan for future work.

## 2 RELATED WORK

**Chatbots Applications for Cybersecurity.** In the paper by (Al-Hawawreh et al., 2023), the authors review the potential of ChatGPT, a chatbot-driven AI technology, in the field of cybersecurity and demonstrate its use in penetration testing and threat defense. In addition, the research paper includes a case study of ChatGPT in creating and executing false data injection attacks on vital infrastructure, including industrial control systems. Furthermore, the article discusses the existing challenges and outlines possible future directions for the integration of ChatGPT into the cybersecurity domain. In (Franco et al., 2020) and (Shaqiri, 2021), the authors apply classical NLP techniques to support high-level cybersecurity planning and management by identifying logs of past cyberattacks, suggesting solutions, and providing insights for decision making. The interaction is done by prompting users to extract or provide different types of solutions.

In the same area, research in (Arora et al., 2023) is developing a conversational chatbot that uses AI and sentiment analysis of Twitter data to predict cyber threats and provide tools and strategies for assessing cyber threats on social media. The potential of chatbots with deep learning as a proactive solution for detecting persistent threats and phishing attacks in real time is studied in (Tejonath Reddy, 2024). The conclusion of their research is that unlike traditional methods that struggle to keep up with cybercriminals' evolving strategies, deep learning algorithms continuously adapt to new paradigms and recognize subtle indicators of phishing attacks. By integrating these mechanisms into chatbots, stronger defenses can be created against the ever-changing phishing attempts and counterfeit prevention.

The study in (Abdelhamid et al., 2023) proposes the use of chatbots integrated with social networks as social collaborative agents for continuous cybersecurity awareness, education and training, providing instant advice and alerts to users in various threatening situations and allowing admin users to add training materials remotely. A similar educational goal is explored in (Fung et al., 2022), which presents a user-friendly cybersecurity chatbot built on Google Dialogflow that educates users about cyber risks, provides a knowledge base, self-quizzes, and advice for dealing with cybersecurity issues, effectively raising cybersecurity awareness,

**Data Security Through Chatbot Interaction.** On the other hand, an interesting topic is the security of data when interacting with chatbots. Since our proposed work and demo use case deal with sensitive user data but also expose important system features to users, we follow the state of the art in this area. Chatbots are gaining popularity in various fields, but their widespread use brings inherent security risks and vulnerabilities. In their systematic literature review, (Yang et al., 2023) the authors focus on identifying potential threats, proposing effective solutions and outlining promising avenues for future investigation.

## 3 DATASETS COLLECTION, FINE-TUNING, USE- CASES

This section presents the sources used for collecting cybersecurity-related information for the creation of a specific dataset and the fine-tuning methods that use them to move the Llama 2-Chat-7B (Touvron et al., 2023) foundation model towards better aligned cybersecurity knowledge and related tasks. Finally, further use cases that the fine-tuning method could address are discussed.

### 3.1 Data Collection

The corpora for the fine-tuning process are collected using human-in-the-loop as the first layer. The authors and volunteer practitioners contributed to a collection of links to arXiv papers, Youtube videos (with transcripts or at least a high-quality English translation) and markdown content from Github repositories or websites. Above all, the selected content should be newer than 2021 so that the fine-tuned model on

Llama 2 can also incorporate newer knowledge with that it was trained. However, the dataset also includes content published before this date, so that we could specialize the model on the most important cybersecurity techniques and concepts. In our implementation, we store the metadata about the content of $\mathcal{D}$ in database chunks using MongoDB[1]. With this strategy, we avoid duplication of content after automatic or human extraction of content and enable continuously updating of the dataset. By chunking the content, old information can also be forgotten or discarded. The content of the dataset is also indexed on disk using the Faiss library (Douze et al., 2024), (Johnson et al., 2019), and loaded into RAM for efficiency reasons, as far as the local machine can handle it. The process is shown in Figure 1. While the model is fine-tuned on the dataset $\mathcal{D}$, this is also indexed for RAG purposes, as in some cases the fine-tuning process may not adapt well enough to specific content, and this would be the backup.

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

## 3.2 Fine-Tuning

We evaluate the state-of-the-art open-source models of Llama 2 (Touvron et al., 2023) for the selection of the LLM foundation. After comparing different classes of models and trade-offs, we decided to fine-tune Llama 2-Chat-7B, which specializes in chat conversations and has 7 billion parameters.

The complete flow from the foundation model to the fine-tuned version used by our methods, *CyberGuardianLLM*, is shown in Figure 1. We use the embedding model of the sentence transformer (Reimers and Gurevych, 2019), more precisely a variant of it that is available as open source, *all-mpnet-base-v2*[2]. Its task is to convert the text format into vectors of float values, as further required by the Faiss indexing and query system (further details in Section 4.2).

For efficiency, $\mathcal{D}$ is stored in an embedded format, which we refer to as $\mathcal{D}_{emb}$. Batches of samples from it are run through the model and compared to the corresponding (ground truth) for similarity using the cross entropy (Hui and Belkin, 2020) as a loss function. To evaluate the performance of the model at predefined steps, the perplexity metric (Meister and Cotterell, 2021) is used.

```
Sample batch of examples B ~ D_emb
# Pass through the model the inputs
    from B to get the predicted
    answers
PredAnswers = CyberGuardianLLM(B["Input"])
# Compute loss for the optimization
    process by comparing how close
    the output of the model is
    compared to the example test.
Loss = CrossEntropy(PredAnswers, B["Answer"])
Update weights of CyberGuardianLLM
```

Listing 1: Fine-tuning process.

# 4 ARCHITECTURE AND IMPLEMENTATION

The implementation of the architecture and components uses a plugin pattern so that higher capacity models or other methods of fine-tuning, different functionalities and components can be switched on and off as required by use cases. In software engineering, this is also known as *separation of concerns*, which we try to utilize as a concept when developing the framework.

The interaction of the system and its components during a user conversation is shown in Figure 2. To connect the interfaces of the components, track the conversation and control the message flow, we have used the LangChain[3] APIs.

## 4.1 Chat Conversation Management

Given a conversation history and a new request from the user, the first step is to ask the LLM model to create a standalone question that contains both the history and the new user's request. In the figure, a concrete example is given by the element *SQ*. In this step, the ability of LLMs to summarize long texts is used. If the input text is longer than the 4096 token limit, we only keep the last part of the conversation within this limit. A simple strategy studied in the literature, that was also evaluated in our case, was to create hierarchical summaries from top to bottom, all within the same boundary (Li et al., 2023). However, in a conversational environment, it might be intuitively beneficial to store a limited queue of recent conversational messages, as our empirical tests have shown. The request to LLM is made via a prompt template as specified in Listing 2.

[1] MongoDB

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

[3] https://www.langchain.com/

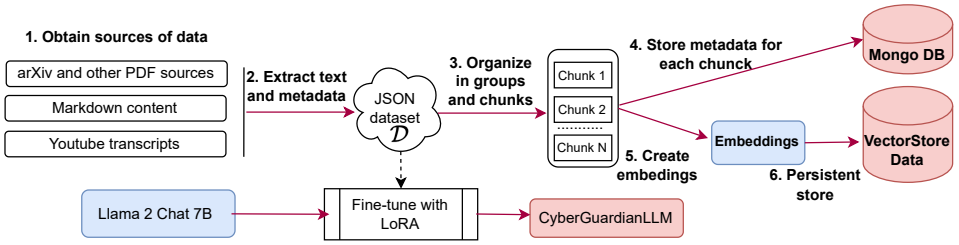


Figure 1: The pipeline for fine-tuning the Llama 2-Chat-7B base model to a dataset containing the latest cybersecurity content. The dataset, $\mathcal{D}$, is first exported as a series of JSON files. In addition, its metadata is stored on a MongoDB server. Finally, $\mathcal{D}$ is chunked and permanently stored in a vector database with FAISS (Douze et al., 2024). The red colored blocks represent the output of this process, while the white blocks are intermediate steps or data. The blue blocks are imported/taken over models from external sources. Further details can be found in Section 3.

```
[INST]Rephrase the following conversation and
      subsequent question so that it is a stand-alone
      question, in its original language.
Conversation:
{conversation_var}
Follow-up question:
{question_var}
Stand-alone question:
[/INST]
```

Listing 2: The prompt template used to create an independent question from the course of the conversation and a new question. Inside the curly brackets are the variables that are used to fill the prompt with concrete examples.

## 4.2 Retrieval Augmented Generation (RAG)

In addition to indexing the knowledge in the base dataset, Figure 1, which leads to the creation of a vector database *VecStoreData*, our method also enables a generic representation for the private set of internal knowledge in a separate vector database called $VectorStore_{int}$. Indexing of internal information and internal data is done by integrating state-of-the-art indexing and retrieval methods for vector stores (Douze et al., 2024), (Johnson et al., 2019). Various types of common sources can be indexed immediately, but the list can be extended on user side:

- Internal whitepapers, documentation on systems, communication, infrastructure.
- Source code, local or private repositories, configuration files.
- Data sources such as SQL databases, JSON files, pandas, csv files, etc.

The main reason for splitting the vector database into two parts is that most questions and support requests from users initially relate to internal knowledge. For a user query $P$, it therefore makes sense to first search for an answer in $VectorStore_{int}$. Only if the similarity score determined by FAISS is not above a certain threshold $T_{RAG}$ (empirically set to 0.85 in our evaluation), the search follows in $VectorStore_{Data}$. In terms of performance, the internal knowledge store is also significantly smaller and faster to access than the other. The RAG process is shown in Figure 3.

## 4.3 User Preference in the Chatbot Interaction

It is important to also adapt the chatbot's responses to the user's preferences. In our demo, we experimentally added some options to the user registration forms, such as

- Should the responses be concise or contain long explanations?
- Does the user prefer answers from the chatbots with emoticons?
- Does the user want the assistant to be very polite or formal?

These options are used through prompt engineering and by using *systems prompts*. Such a prompt, as shown in Listing 3, appears before any other prompt message from the user's side, including, for example, the standalone question generated (Listing 2). This strategy prompts the LLM to follow some suggested rules, even if it is not always forced to do so. The available set of options for each template variable is show in Eq. 2.

```
<<<SYS>>>
In your reponses please follow the following
     instructions:
{responses_type_var}
{use_emoticons_var}
{politeness_var}
<<</SYS>>>
```

Listing 3: The prompt template is used to personalize the interaction between each user and the chatbot according to their preferences. The variables in curly brackets are filled with the options of the respective user.

$$
\begin{aligned}
&responses\_type\_var \in \{\text{"Be concise in your response"},\\
&\text{"Provide long contextual based responses"}\}\\
&useemoticons\_var \in \{\text{"Use emoticons"},\\
&\text{"Do not use emoticons"}\}\\
&politeness\_var \in \{\text{"Be polite"},\text{"Be formal"}\} \qquad (2)
\end{aligned}
$$

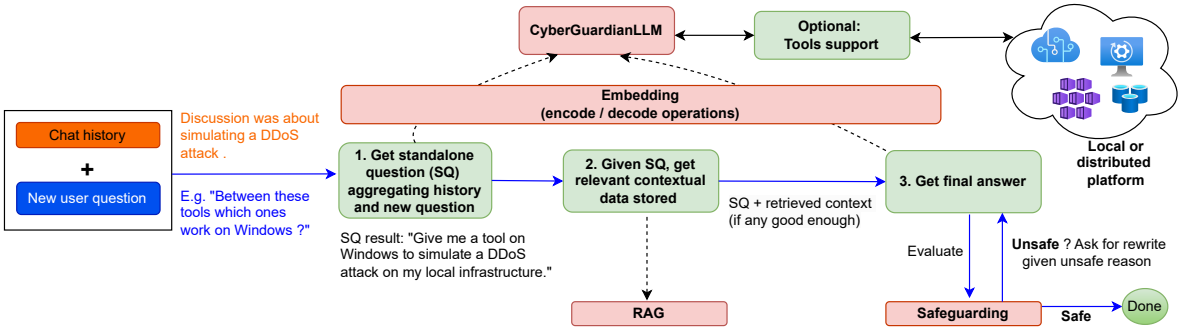


Figure 2: The conversational chat system implemented in the proposed method and its associated components. The flow is represented by the blue arrows. The three main steps (the green boxes) are used to aggregate the conversation flow, retrieve internally relevant stored data, and then obtain the final response after the safeguard component approves the results. The optional step is used by the LLM to interact with the registered tools on the deployed platform.

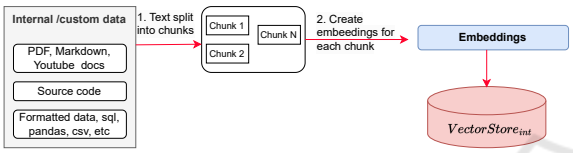


Figure 3: The pipeline for creating the RAG support for the internal/customized data depends on the user settings and knowledge of the platform.

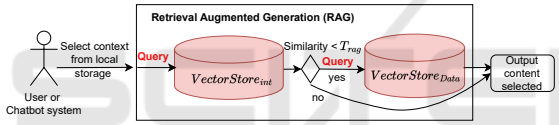


Figure 4: The process for extracting relevant context from the indexed vector databases for RAG. First, the system tries to find information in the internal knowledge base that most closely matches the user's needs and should be quick to search. If nothing is found, the system tries to find relevant context in the data set used for fine-tuning as a backup.

## 4.4 Safeguarding Interactions

Our method first experimented with the integration of Llama Guard (Inan et al., 2023), an LLM safeguard model that fulfills the aforementioned use case. It provides also a taxonomy of security risks. When prompted, the model first outputs the status, i.e. safe or not, and in the case of *unsafe* it provides details of the element in the taxonomy of unsafe reasons.

Considering the computational requirements of loading another 7B model, our method has also explored and integrated packages from the NLTK (Bauer et al., 2020) based on classical NLP techniques that partially resolve various safety classifications in the prompts. There is a trade-off between the two solutions. The first solution has the advantage of providing more accurate results without the need to create a local taxonomy of things to test and output reasons. The advantage of importing or customizing the NLTK packages is on the request side, as they are fast and require little memory.

## 4.5 Tools and Interaction with the Backend Systems

To improve the productivity and response time of SOC specialists, our work integrates state-of-the-art methods that address the interaction between the user, the LLM and the system processes. The concepts are known in the literature under the terms *agents*, and *tools* (Yao et al., 2023), (Schick et al., 2023), (Patil et al., 2023).

In our framework, the interaction between LLM and tools is shown in Figure 5. At each step, the LLM considers the problem it needs to further solve, the set of available tools, and decides whether to invoke one of the tools that could further interact with the organization's infrastructure and services, or just return the currently available response. The benefit of this method is comparable to the concept of *task decomposition* in computer science, where a problem is broken down into smaller tasks and solved piece by piece. For LLM, and especially for small models, this is important because it can leverage external capabilities and break the problem into smaller pieces that are easier to manage.

We use the ReACT agents (Yao et al., 2023) and their Langchain API. The main idea behind the method is to do prompt engineering and inform the LLM about a short description of each of the available tool. A prompt template similar to the one in Listing 2 is used, where instead of the variable *conversation_var*, the framework fills in the description of each tool and how it should be called. Using a small set of 50 examples of tool calls and parameter extraction examples, we fine-tuning *CyberGuardianLLM* over a few epochs.

Most of the core tools in our smart home demo

application (Section 5) are reused from Langchain and enable interaction with the organization's systems through natural language queries, such as:

- Interaction with SQL or Pandas databases, e.g. filter, select or aggregate data from multiple tables.
- Office tools, e.g. sending emails or adding entries to the calendar.
- Python code generation. Code Llama 7B (Rozière et al., 2024) is used in the demo app.
- Python code execution with PythoREPL[4]. This tool can shorten the time it takes to call the code and get results and lets the LLM do the unnecessary work in the background.

The *Identify layer* block in Figure 5 is used as an intermediate step to help the LLM select one of the tools by identifying keywords in the prompt and asking the model to make a correlation between the prompts and the tools' tags. Our evaluation has shown that tagging prompts and tools is more helpful in recognizing the correct tool than the standard Langchain's method of prompt engineering (default in the ReACT implementation). Note that the tool suite can be customized by the user. Langchain, for example, already contains a large list of tools that can be extended by those implemented by the user.

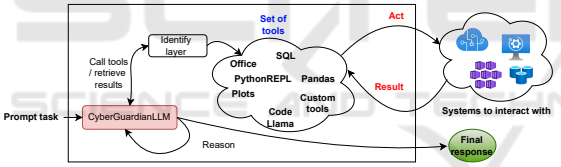


Figure 5: Starting from a specific prompt or task, the figure shows the flow of interaction with tools, services, and systems provided by the organization.

# 5 EVALUATION

## 5.1 Quantitative and Qualitative Evaluation

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

- **RQ1.** How is the fine-tuned model able to understand the cybersecurity domain?
- **RQ2.** How well is the overall *CyberGuardian* system able to meet user needs?

**RQ1 Analysis.** To understand how helpful the fine-tuned CyberGuardianLLM is from a quantitative perspective, we evaluate the usefulness of the responses using a common method (Zheng et al., 2023) of automated evaluation using a much larger model, specifically GPT-4 , as a *judge*.

We assume 5 topics in cybersecurity needed by SOC specialists: a) protection of systems from security risks and malware, b) cryptography, c) configuration of security protocols such as firewalls, intrusion detection systems (IDS), d) network security infrastructure (firewalls, VPNs, web proxy, IDS/IPS), e) investigation of data breaches and data leaks. We denote these clusters with $Topics = (tp_i)_{i=1,...5}$. The *judge*, GPT-4, is asked via a prompt to create 20 questions for each of the 5 topics, resulting in a total of 100 questions, denoted by the quantity *Que*. This set is obtained by filling in the topic variable in the template shown in Listing 4.

```
[INST] Consider yourself an interviewer for a SOC
    specialist position.
Ask 20 relevant. different questions on the topic: "{
    topic_var}"[/INST]
```

Listing 4: The template prompt used to ask questions to the judge LLM.

To compare the answers of two LLMs, *LLM1_resp* and *LLM2_resp*, we ask the *judge* again via a prompt which one he prefers, Listing 5. We fill the template variables for each question $Q \in Que$.

```
[INST] Given the question: "{Q_var}", tell me which
    of the following two answers you prefer. Write
    only Response1 or Response2.
### Response 1: {LLM1_resp}.
### Response 2: {LLM2 resp}. [/INST]
```

Listing 5: The template for classifying the answers with judge LLM.

We measure the CyberGuardianLLM responses for each $Q \in Que$ against the 3 Llama 2 models: Llama 2-Chat-7B, 13B, and 70B. Table 1 shows the average preference in percent between these models. The results are similar even when broken down by topic, which is kind of expected since the data collection was done for all of these categories.

The BLEU metric, which is also used in the evaluation of other LLMs (Touvron et al., 2023), compares the judge's reference response (GPT-4) with the one generated by each LLM. The BLEU scores for our test under the above conditions are shown in Figure 6. The scores range from 0 to 100, with a higher score indicating a stronger match between the generated response and the reference. A score of 100 represents a perfect match, while a score of 0 means that there is no overlap between the generated response and the reference.

**RQ2 Analysis.** For this analysis, we reused the work of (Cristea et al., 2022), where different smart home applications connect to a central hub server from different locations either directly or via other servers. In

[4]https://realpython.com/python-repl/

Table 1: Head-to-head preference of responses using the GPT-4 as a judge. The second column shows the percentage of cases in which each of the compared models was preferred to the CyberGuardianLLM.

| Model | Preferred over CyberGuardianLLM |
|---|---|
| Llama-Chat-7B | 26% |
| Llama-Chat-13B | 39% |
| Llama-70B | 58% |

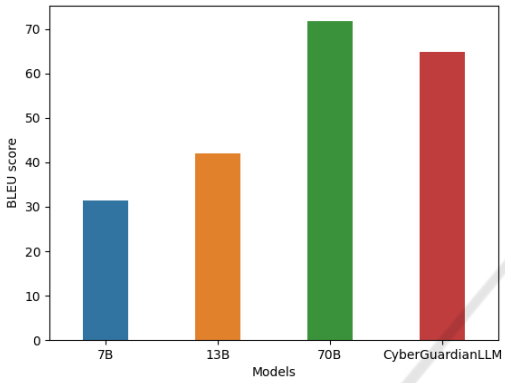


Figure 6: The BLEU metric compared between the 4 models under evaluation.

addition, a simulator for different types of attacks, including DDoS (de Neira et al., 2023), has been implemented. The snapshots of the attacks provided various data tables with statistics on the utilization of resources (e.g. servers, routers, local hub systems), 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 [5] libraries and tools (including visualizations). All these simulations and screenshots can be found in the repository.

The test was attended by 23 practitioners with different levels of cybersecurity knowledge, but most of them are still at the very beginning of their careers. During our demo use case, the system triggered signals that could indicate a DDoS attack, e.g. unusually many timeouts for various operations in the IoT hub, response code errors in class 500, etc. The human's task was to first recognize that there could be a potential DDoS in the background and solve it with the support of CyberGuardian. The resolution path then involves: a) creating 2D/3D diagrams of the resources in use at different sites (using the plotting tool), b) filtering IPs that appear to be flooding the servers through database queries, and c) asking the chatbot to write a piece of Python code that adds the identified attacking IPs to a blacklist firewall rule using the one custom tool.

The task was solved correctly by 17 people within a single instance of the test. The rest had to restart the test and try once or twice. We collect their feedback after each question-response pair and the final survey. For each feedback, they could write it in natural language and also give a score between 1-5. All feedback received were collected via the Trubrics[6] platform. The average score of the responses was 4.2, while the average reported time to solve the task was $\sim 37$ minutes, with a minimum of 14 and 49 maximum minutes respectively. The natural language feedback could later be correlated with the ratings and text in the user chatbot interaction to learn a reward function and perform reinforcement learning from human feedback (RLHF), as the Llama models do, but this requires certain resources, and for now this is left as future work given the scope. Considering the feedback from the users, their abilities and the potentially difficult sequential step problem they had to solve, the results are promising in our view.

## 5.2 Discussion

**Observations from the RQ2 Sessions.** The first important observation we made through the human evaluation, which we iterated a few times in different scales and versions of CyberGuardian, is that the *task decomposition* with the support of LLM reasoning traces and tools was significant. The impact can be even greater for large organizations and a significant number of tools, which is also known in the industry and frameworks as *multi-agent workflows*[7]. Another important observation is that the models with small classes may have problems understanding long contexts, with unexpected results such as repetition of the context or inability to understand and respond appropriately to long contexts. Breaking it down into small steps and prompts is recommended.

# 6 CONCLUSIONS AND FUTURE WORK

This paper explores the intersection of cybersecurity and Large Language Models (LLMs) in the context of Security Operations Centers (SOCs). The proposed framework, CyberGuardian, aims to improve the tasks of SOCs through the use of LLM tech-

[5]https://streamlit.io/

[6]https://trubrics.com/

[7]https://blog.langchain.dev/langgraph-multi-agent-workflows/

niques. Its plugin includes features such as Retrieval Augmented Generation (RAG), methods for securing human-chatbot interaction and natural language interaction for managing cybersecurity tasks related to databases, firewalls, plotting graphs, code generation and execution.

## ACKNOWLEDGEMENTS

This research was supported by European Union's Horizon Europe research and innovation programme under grant agreement no. 101070455, project DYNABIC.

## REFERENCES

Abdelhamid, S. 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).

Bauer, T., Devrim, E., Glazunov, M., Jaramillo, W. L., Mohan, B., and Spanakis, G. (2020). # metoomaastricht: Building a chatbot to assist survivors of sexual harassment. In *International Workshops of ECML PKDD 2019*, pages 503–521. Springer.

Cristea, R., Feraru, M., and Paduraru, C. (2022). Building blocks for iot testing - a benchmark of iot apps and a functional testing framework. In *SERP4IoT@ICSE*, pages 25–32. ACM.

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.

Douze, M. et al. (2024). The faiss library. *arXiv preprint arXiv:2401.08281, https://github.com/facebookresearch/faiss*.

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.

Hui, L. and Belkin, M. (2020). Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. *arXiv preprint arXiv:2006.07322*.

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

Johnson, J., Douze, M., and Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Li, M., Hovy, E., and Lau, J. H. (2023). Summarizing multiple documents with conversational structure for meta-review generation. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Meister, C. and Cotterell, R. (2021). Language model evaluation beyond perplexity. In *ACL-IJCNLP*, pages 5328–5339.

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.

Patil, S. G. et al. (2023). Gorilla: Large language model connected with massive apis. *CoRR*, abs/2305.15334.

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.

Schick, T. et al. (2023). Toolformer: Language models can teach themselves to use tools. In Oh, A. et al., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551.

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)*.

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

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 *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

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.