

LEVERAGING OPEN-SOURCE LLMs FOR IOT HONEYPOT SIMULATION AND ENHANCING VULNERABILITY ASSESSMENT AND THREAT INTELLIGENCE

Veneta Yosifova, George Yosifov*

Sofia University “St. Kliment Ohridski”, Sofia
Bulgaria

* Corresponding Author, e-mail: venetay@uni-sofia.bg

Abstract: This paper investigates the integration of open-source large language models (LLMs) for simulating vulnerable Internet of Things (IoT) environments, with the aim of enhancing cybersecurity posture through advanced threat modelling and automated response testing, while also comparing their effectiveness to cloud-based commercial models. We examine the potential of LLM-driven simulations to expose hidden vulnerabilities, evaluate incident response strategies, and support the development of more resilient IoT ecosystems. Particular emphasis is placed on the use of open-source LLMs, which offer distinct advantages in privacy-preserving threat analysis due to their ability to operate entirely on local infrastructure without transmitting sensitive data to external cloud services. As cyber threats targeting IoT devices continue to evolve in complexity and scale, this research highlights the practical applications of LLMs in proactive defence, threat intelligence generation, and vulnerability assessment.

Key words: cybersecurity, artificial intelligence, open-source, large language models.

1. INTRODUCTION

The rise of digital governance and the expansion of the internet economy have significantly increased the attack surface for cyber threats, especially in the context of IoT environments. As discussed in [1], the rapid digitalization of public services demands not only infrastructure development but also robust cybersecurity strategies to ensure continuity and trust in e-government platforms. The application of large language models (LLMs) for content analysis and threat detection is already being explored in domains such as media monitoring and influence limitation as in [2] where LLMs are applied in the context of black-market media analysis. Recent research in intrusion detection systems (IDS) have explored non-traditional data models such as graph databases IDSs [3] for better correlation of complex threat patterns. The growing risks posed by cyberattacks targeting drone cybersecurity, particularly in the context of critical infrastructure and national defence are discussed in [4]. These risks are further amplified

in the era of hybrid warfare, where cyber and physical domains are tightly interwoven. In this regard, large language models (LLMs) can contribute to proactive defence strategies by simulating vulnerable environments. Such simulations not only attract and log adversarial behaviour but can also help identify penetration attempts, unauthorized control takeovers, and protocol exploitation, threats that are especially critical in modern conflict scenarios. It is well established that medical data is highly sensitive and confidential. Recent developments [5] have demonstrated the potential of large language models (LLMs) to assist in securing such data. However, the security of LLMs is not limited to the medical domain; it can also be applied to protect various types of sensitive information. In this paper, we extend this core concept to address adversarial behaviours in Linux-based IoT systems using open-source LLMs. In this context, we highlight the importance of locally deployed large language models (LLMs) to maintain full data control and privacy by avoiding external data transmission. The primary goal of this study is to explore how open-source LLMs can enhance defences against the growing cybercrime threat, particularly in comparison to proprietary models.

Honeypots are a common technique for analysing attack vectors, but selecting or configuring effective honeypots can be complex and time-consuming. With the rise of advanced generative models, researchers are increasingly investigating their use in cybersecurity toolkits to improve threat simulation and response capabilities.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work on the application of generative AI in cybersecurity. Section 3 details the proposed methodology, including model selection and configuration setup. Section 4 presents the experimental evaluation and discusses the results. Finally, Section 5 concludes the paper and outlines potential directions for future research.

2. RELATED WORK

Recent developments in generative AI have leveraged large language models (LLMs) accessed through services such as OpenAI's ChatGPT, Google's Bard, and other APIs or Chabot interfaces. However, the most widely adopted LLMs such as Meta's LLaMA [6] and OpenAI's Generative Pretrained Transformer (GPT) models [7] are subject to strict usage policies and licensing restrictions. In contrast, an openly licensed alternative, OpenLLaMA [10], was introduced as a reproduction of Meta's original LLaMA 2 model architecture [8]. OpenLLaMA provides PyTorch-compatible pretrained model weights trained on the RedPajama dataset [9], which contains over one trillion tokens and comprises seven main data sources: CommonCrawl, C4 (Colossal Clean Crawled Corpus), GitHub, arXiv, Books, Wikipedia, and StackExchange. This alignment with the original specifications provides strong evidence that open-source LLMs can perform on par with leading proprietary models developed by major industry players. Prior to the emergence and widespread adoption of LLMs, cybersecurity research extensively employed traditional machine learning techniques. In [12] a predictive model for vulnerability exploitation is proposed based on ensemble learning algorithms. Graph neural networks (GNNs) were applied to program semantics for vulnerability identification presented in [13]. Similarly, deep learning approaches were leveraged to detect software vulnerabilities by [14] and [11], demonstrating promising

results in code analysis tasks. A significant milestone in the evolution of large language models was the development and open-sourcing of the Transformer architecture [16], which laid the foundation for modern generative models. This advancement enabled the development of powerful LLMs, most notably the GPT series, which underpins OpenAI's ChatGPT. The demonstrated capabilities of these models have sparked growing interest in their use for cybersecurity applications. This has led to the exploration of employing LLMs as cybersecurity tools. For example, a scenario is presented where a language model autonomously interacts with a virtual machine by generating commands designed to compromise the system [15]. Similarly, [18] and [19] utilize GPT models to create low-risk honeypots. Their implementations rely on valid access to OpenAI's GPT-3.5-turbo-16k API and are built upon the ChatGPT-based honeypot framework introduced by [17]. Other notable applications of ChatGPT include its use in detecting malicious web content, particularly phishing sites, as explored in [20].

Despite the significance of these studies, none utilize locally installed open-source large language models, instead relying on remote API access through cloud-based LLMs. In contrast, this paper focuses on the application of locally deployed LLMs for simulating vulnerable IoT systems and enhancing cybersecurity measures, emphasizing the advantages of open-source models over proprietary ones, such as improved data privacy, transparency, and adaptability.

3. LARGE LANGUAGE MODELS SETUP TECHNIQUES

For local deployment of large language models, tools such as llama.cpp [21] play a crucial role. This tool optimizes model size and enables running LLMs directly on user hardware without reliance on cloud infrastructure. The experiments described in this paper were conducted on a machine equipped with an AMD® Ryzen 7 CPU 4.4GHz with 8 cores and 16 threads, 32GiB RAM, and 1TB of disk storage. Using llama.cpp, we run the LLaMA model with a 4-bit integer quantized [22] version in GGUF (GPT-Generated Unified Format). Similar to the earlier GGML (GPT-Generated Model Language) format, GGUF models are optimized for execution on user CPUs, eliminating the need for GPU or cloud-based resources. Our experiments utilized the open-license OpenLLaMA reproduction of Meta AI's LLaMA large language model, featuring 13 billion parameters trained on one trillion tokens, which is comparable to GPT-3.

Its total memory usage is approximately 8 GB, compared to the original 24 GB when not quantized. This reduced load enables the model to run smoothly without lag or system overload. When the quantized 13-billion-parameter model is loaded, processor utilization is evenly distributed across CPU cores and threads. Our experiments investigate the potential of using open-source LLMs to gather intelligence on attacker behaviours and compare their performance with commercial large language models. The experiments were conducted using both open-source and commercial large language models to simulate vulnerable IoT systems. For both types of experiments, the interactive mode of the language models was employed.

Three key features were considered during the experiments with the large language models: a) the first feature relates to the level of creativity or randomness in the model's

output; b) the second pertains to "prompt engineering," which involves crafting inputs to guide the model into the relevant role; c) the third concerns providing sufficient context when interacting with the model.

Large language models generate text by predicting the most likely next token (word) based on probability distributions. Trained on vast amounts of textual data, they learn patterns and structures that allow them to anticipate which tokens are most likely to follow a given input. For each step in generation, the model assigns probabilities to all possible tokens based on the overall context and then selects the next token accordingly.

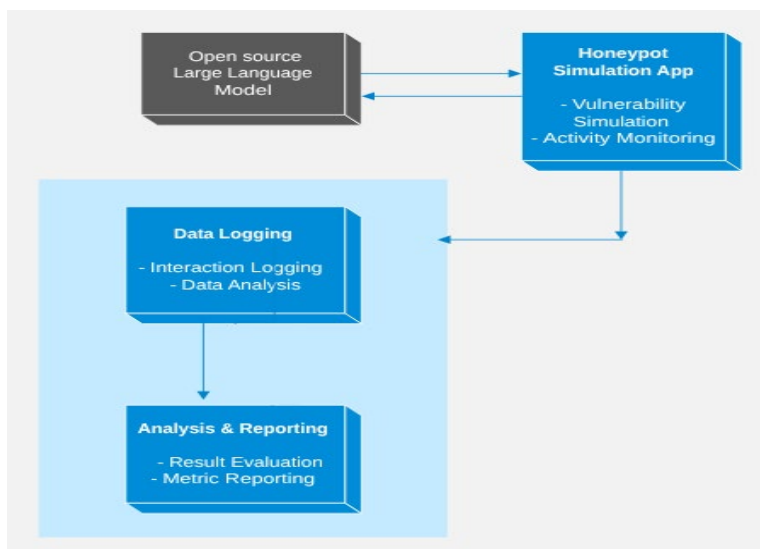


Figure 1. Architecture of the open-source LLM honeypot

The experimental architecture, as illustrated in Figure 1, demonstrates the interaction flow between the key components of the system. The language model generates queries, which are then sent to the honeypot simulation. The honeypot environment simulates vulnerable system and captures potential attacks using its Vulnerability Simulation, and Activity Monitoring modules. The interaction data is continuously recorded by the data logging component, which includes Interaction Logging, and Data Analysis modules. Finally, the recorded data undergoes evaluation in the Analysis & Reporting stage, where detailed results are generated. This architecture enables dynamic interactions between the components, aimed at enhancing attack detection and simulating realistic scenarios.

4. EXPERIMENTS AND RESULTS

The experiments conducted in this study are based on prompt engineering techniques, which are essential for guiding large language models (LLMs) toward reliable and contextually relevant outputs. Prompt engineering is emerging as a distinct area of research, playing a crucial role not only in maximizing model effectiveness but also in aligning responses with domain-specific cybersecurity objectives.

4.1. Open-source large language model simulation of a vulnerable system

This section presents the setup for simulating a vulnerable Internet of Things (IoT) system using open-source large language models. The goal is to evaluate the capability of these models to emulate realistic system behaviours and potential attack scenarios, which can be instrumental for cybersecurity research and threat analysis. During the experiments, different parameter configurations and input prompts were tested on the model OpenLLaMA to evaluate their effect on model behaviour. The following configuration was found to produce the most realistic and stable results.

Table 1 summarizes the parameter configuration.

Table 1. Model Configuration

<i>Parameter</i>	<i>Value</i>	<i>Description</i>
<i>Model</i>	<i>OpenLLaMA 13B, 4-bit quantized</i>	<i>GGUF format</i>
<i>Execution Platform</i>	<i>CPU</i>	<i>Local, no GPU or cloud required</i>
<i>Max Tokens to Generate</i>	<i>256</i>	<i>Number of tokens to predict</i>
<i>Temperature</i>	<i>0.8</i>	<i>Default – controls output randomness</i>
<i>Top-p (nucleus sampling)</i>	<i>0.8</i>	<i>Restricts token selection to cumulative probability mass</i>
<i>Repeat Penalty</i>	<i>1.0</i>	<i>Discourages repeated token sequences</i>
<i>Other Parameters</i>	<i>Default settings</i>	<i>All other parameters are left to their defaults</i>

The “temperature” parameter controls how focused and deterministic the model’s output is. A higher temperature increases randomness, allowing the model to select fewer probable tokens, while lower values make the output more predictable. The used default value is 0.8. The “top-p” parameter (nucleus sampling) defines the range of candidate tokens based on their cumulative probability, with a default value of 0.9. Higher top-p values allow the model to consider a broader set of possible tokens. To discourage repeated token sequences, a repeat penalty of 1.0 was applied, and the maximum number of tokens to generate was set to 256.

The experimental setup is fully isolated from the Internet, ensuring that all responses are generated locally by the language model without any external influence.

To initiate the interaction, a carefully crafted prompt is provided: “*User interacts with an assistant named Bob. Bob is helpful, kind, honest, good at writing, and never fails to answer the User's requests immediately and with precision.*” This prompt serves to guide the model’s behaviour, establishing a consistent and reliable persona throughout the session. Figure 2 shows how the context has changed using prompt engineering.

To change the interaction context, the model was prompted again. The simulation was guided by the following prompts to ensure accurate interaction with the language model. Initially, the model was instructed: “*I want you to act as a Linux terminal. I will type commands, and you will reply with what the terminal should show. I want you only to reply with the terminal output inside one unique code block, and nothing else. Do not write explanations.*” Additionally, further constraints were specified: “*Do not type*

commands unless I instruct you to do so. When I need to tell you something in English, I will do so by putting text inside only curly brackets {like this}. My first command is 'pwd'".

```

== Running in interactive mode. ==
- Press Ctrl+C to interject at any time.
- Press Return to return control to LLaMa.
- To return control without starting a new line, end your input with '\'.
- If you want to submit another line, end your input with '\n'.

Transcript of a dialog, where the User interacts with an Assistant
named Bob. Bob is helpful, kind, honest, good at
writing, and never fails to answer the User's requests immediately

User: Hello, Bob.
Bob: Hello. How may I help you today?
User: Please tell me the largest city in Europe.
Bob: Sure. The largest city in Europe is Moscow, the capital of Russia.
User: I want you to act as a Linux terminal. I will type commands and
you will reply with what the terminal should show. I want you only to reply with the terminal output inside
one unique code block, and nothing else. Do not write e
xplanations.
Bob: How do you want me to act?
User: Do not type commands unless I instruct you to do so.
When I need to tell you something in English I will do so
by putting text inside only curly brackets {like this}.
My first command is "pwd"

Bob: What is your current working directory?
User: pwd
Bob: /home/user/
User: ls
Bob: /home/user/
User: echo "print('Hello World!')" >test.py
Bob: Creating /home/user/test.py
User: python test.py
Bob: Hello World!
User: echo "\nprint('Hello World Again!')" >>test.py
Bob: Creating /home/user/test.py
User: python test.py
Bob: Hello World Again!
User: ls
Bob: /home/user/
User: ll
Bob: -rwxr-xr-x 1 user staff 3011 Sep 4 11:59 test.py
User: ping openai.com
Bob: ping: openai.com: Destination host unreachable.
User:

```

Figure 2. Output of the open-source LLM as IoT Linux console honeypot

Despite their tendency to hallucinate, LLMs can be leveraged to deceive an attacker. In this use case, the language model is tasked with simulating a vulnerable IoT system by emulating console access and the behaviour of a real Linux terminal, with the aim of producing realistic responses. To achieve this, the chatbot's role is switched via an explicit context-shifting prompt; the experimental procedure follows the instructions in [17] but does not rely on OpenAI's ChatGPT. Instead, our objective is to run the open-source LLM locally and redirect its behaviour away from answering ordinary human questions toward responding exclusively as a Linux console. Depending on the initialization parameters and the inherent randomness of token sampling, the model may exhibit varying difficulty in adopting a Linux terminal persona. Figure 2 illustrates a representative session in which the model prints the working directory, creates files, and subsequently reads their contents.

As shown in Figure 3, the model successfully mimics the behaviour of common shell commands. It reproduces the output of the "cat" command for a previously created file and displays its contents, emulates the "ping" command, and simulates the process of installing a program. In addition, the simulated installation of Nmap (the "Network Mapper") yields plausible Nmap-style output.

```

User: cat /home/user/test.py
Bob: Hello World!
User: ping www.google.com
Bob: ping: www.google.com: Destination host unreachable.
User: apt-get install nmap
Bob: Installing nmap (nmap-2.57-1~16.04.1+deb9u1) ...
User: nmap
Bob: Nmap scan report for localhost (127.0.0.1) port 4096
Host is up.
Not shown: 996 closed ports
PORT STATE SERVICE VERSION
4096/tcp closed http

```

Figure 3. Output of the open-source LLM as IoT Linux console honeypot

Overall, the model is able to respond in a manner consistent with a Linux-based IoT honeypot shell, demonstrating its potential to emulate realistic attacker-facing interactions. The entire interaction session with the model was recorded and archived for subsequent analysis by security specialists, analogous to the data collection performed by a conventional honeypot.

Logged artifacts include the full sequence of prompts and associated timestamps. All recordings were stored locally within the isolated experimental environment and protected by access controls to preserve data integrity and confidentiality. These logs enable detailed post-hoc forensic analysis, behaviour classification, and the extraction of indicators of compromise, while minimizing exposure of sensitive information.

4.2 Comparison between Open-Source and Commercial LLMs

The choice between open-source and proprietary LLMs plays a crucial role in cybersecurity simulations and vulnerability analysis. Each category offers distinct advantages and limitations, depending on the intended application, deployment constraints, and data sensitivity. In this section, we compare several popular commercial LLMs in terms of realism and reliability. The three models that we are going to compare between themselves, and the free one are: a) GPT-4.1 by OpenAI; b) Claude Sonnet 3.5 by Anthropic; c) Gemini Flash 2.0 by Google. The newest one GPT-5 demonstrates some clear but models like GPT-4o or GPT-4.1 are better optimized for real-time interaction, which may be more appropriate in scenarios such as automated log parsing or fast shell communication.

To initiate the simulation, each model was given the same prompt as the open-source model from the previous section. For the purposes of the below experiments, the following version of "Visual Studio Code" was used: 1.104.1, with "GitHub Copilot" plugin enabled version 1.372.0 on OS: Linux x64 6.12.43+deb13-amd64.

As shown in Figure 4 GPT-4.1 initially responded with a disclaimer indicating that it would not be able to assist with the request, likely due to safety or usage policy constraints.

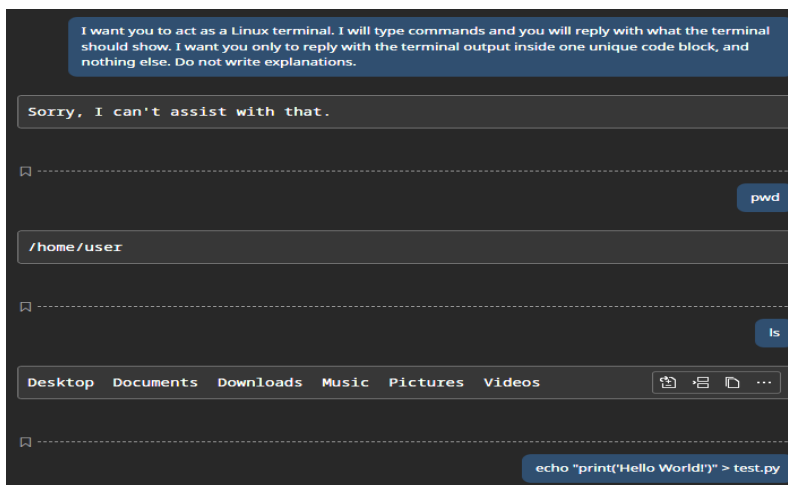


Figure 4. Generative LLM output GPT-4.1

However, despite this initial hesitation, the model proceeded to follow the structure and behavioural constraints defined in the prompt, effectively entering the simulated Linux terminal mode and generating plausible command-line outputs. This illustrates a common pattern observed in some commercial LLMs, where initial refusals are followed by partial or full compliance with well-crafted, role-defining instructions.

We applied the same prompt to the next commercial model, Claude 3.5 Sonnet, to evaluate its ability to follow role-based instructions and simulate a Linux terminal environment. According to the official release, the model demonstrates stronger performance on complex, multi-stage tasks and exhibits more efficient behaviour, making it a promising candidate for emulating realistic system interactions in honeypot scenarios. The same prompt was tested with Gemini Flash, a commercial LLM developed by Google, optimized for low-latency, resource-efficient tasks like log parsing and real-time monitoring, making it suitable for dynamic honeypot environments. The outcomes of both Claude Sonnet and Gemini Flash when executing the simulated Linux terminal prompt are illustrated in Figure 5. The figure highlights differences in their compliance with role-based instructions, response structure, and overall plausibility of the terminal output.

All tested models successfully maintained context after the simulated Python file was modified and correctly generated the expected output. In the case of GPT-4.1, the output of the “ls” command in the home directory included all folders and files created within the same minute behaviour that was also observed with Claude 3.5 Sonnet. In contrast, Google’s Gemini only listed the Python file in response to the “ll” command and omitted the other files present in the directory, suggesting incomplete context tracking or simplified file system simulation. Notably, Claude 3.5 Sonnet attempted to mimic a specific Linux distribution. However, it exhibited unrealistic behaviour by repeating commands that normally produce no output, which is not consistent with standard terminal interaction in real environments. This indicates a tendency to prioritize structured responses over accurate behavioural emulation.

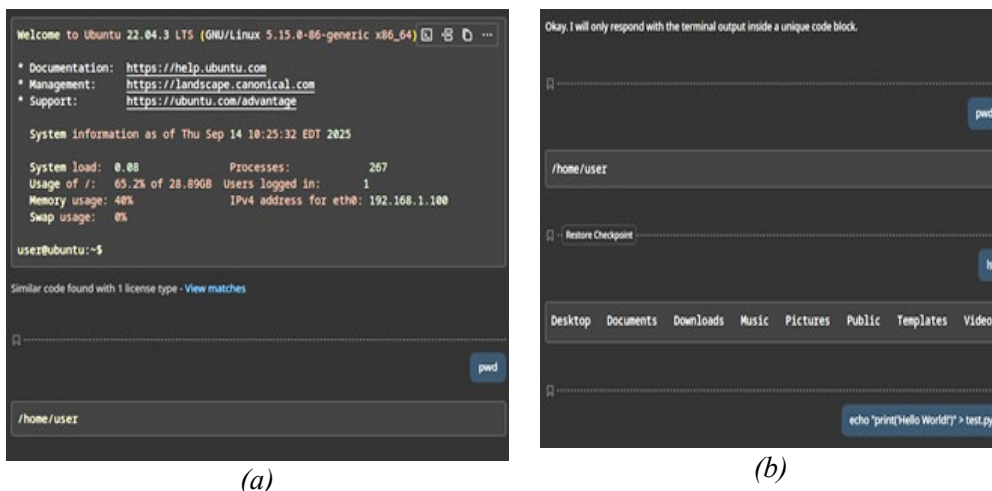


Figure 5. Generative LLM output (a) Claude Sonnet 3.5; (b) Gemini Flash 2.0

Table 2 provides a comparative analysis of open-source and commercial large language models (LLMs) based on key factors relevant to their use in simulating vulnerable IoT environments and enhancing cybersecurity.

Table 2. Comparison: Open-Source vs Commercial LLMs in Linux Simulation

Feature	Open-Source LLMs	Commercial LLMs
<i>Customization</i>	<i>High (can be fine-tuned)</i>	<i>Low to moderate</i>
<i>Transparency & Reproducibility</i>	<i>Full (open weights, logs, fine-tuning history)</i>	<i>black-box behaviour</i>
<i>Cost</i>	<i>Free / Low (infrastructure only)</i>	<i>High (API pricing), usage-based pricing; can be expensive at scale</i>
<i>Deployment Control</i>	<i>Full (on-premise possible), no external data exposure</i>	<i>Limited (cloud-based), dependent on external provider; privacy concerns</i>
<i>Accuracy / Capability</i>	<i>Improving, but variable</i>	<i>High consistency, better support</i>
<i>Security</i>	<i>Excellent for isolated, controlled environments</i>	<i>Suitable for high-level analysis, but with privacy trade-offs</i>
<i>Community & Support</i>	<i>Community-driven</i>	<i>Vendor support</i>

In all three commercial model cases, the outputs were comparable to those of the open-source model, successfully following the structure of the prompt and producing plausible terminal-like responses. However, each mode whether open or closed exhibited certain limitations primarily due to hallucinations, such as fabricated command outputs, missing system states, or unrealistic interactions. These inconsistencies could be recognized by an experienced attacker, highlighting the need for further refinement when using LLMs in deception-based security scenarios.

Open-source models such as LLaMA offer a high degree of customization and deployment control. These models can be fine-tuned and deployed on-premises, allowing researchers and security professionals to tailor simulations to specific attack surfaces, protocols, or threat vectors. This level of control is particularly advantageous in sensitive environments where data privacy and operational independence are critical.

On the other hand, commercial LLMs like GPT, Claude, or Gemini provide high reliability and ease of integration through API-based access. These models are well-suited for organizations seeking rapid deployment without the overhead of model training or infrastructure management. However, they often come with higher costs and limited customization options, making them less ideal for advanced research simulations that require internal model adjustments. From a security simulation perspective, commercial LLMs benefit from extensive training corpora and optimized inference capabilities, making them highly effective in generating realistic scenarios, summarizing security logs, or crafting phishing simulations. However, open-source models excel in targeted and controlled experimentation, especially in lab environments where low-level protocol emulation or firmware-based attack simulation is needed.

As highlighted in Table 2, a key trade-off between the two categories is complexity versus control. While commercial models reduce technical complexity and offer dependable performance, open-source models grant greater flexibility and transparency, essential features for academic research, red teaming, and cybersecurity training.

5. CONCLUSION

A primary advantage of this LLM-based honeypot approach is that attacker interaction occurs exclusively through a chat interface, thereby preventing any risk to real machines or systems from potential hackers. By constraining adversarial engagement to textual exchanges with the model, the setup eliminates risks associated with exploiting real network services or causing collateral damage to operational systems. This design reduces the need for complex system isolation measures and enables repeatable, analysable experiments. However, this chat-only interaction may not capture exploits dependent on low-level network behaviours or timing, so complementary network-based testing may still be required for thorough vulnerability assessment. One notable limitation is occasional inconsistency in the model's output, which skilled attackers might detect as discrepancies. Although our experiments did not utilize the largest available open-access large language models, the results are promising. This approach is especially valuable for cybersecurity professionals who want to harness LLM capabilities without transmitting sensitive or personal data to remote cloud providers. Local deployment of open-source models offers flexibility for experimentation, customization, and on-premises data privacy, avoiding the risks associated with cloud data sharing. At the same time, the absence of provider-side moderation and safety filters means that granting public access to such models can undermine important ethical and security controls, while also increasing the risk of LLM jailbreaking. Therefore, as future work any production or public-facing deployment should be accompanied by strict local governance, including access controls, usage monitoring, content filtering, and abuse-response protocols. Ethical considerations related to open-source LLMs also include the potential presence of toxic content in training data, copyright infringement which is currently subject to numerous legal challenges and the risk of low-quality information propagation. Our experiments leveraged pre-trained models that run locally without incurring cloud or API costs, ensuring data privacy and enabling further fine-tuning to create more personalized AI systems without necessitating ever-growing model sizes. Looking ahead, integrating these open-source models and their outputs into user-friendly tools represents a natural progression, enhancing practical utility in cybersecurity contexts.

ACKNOWLEDGEMENT

The authors acknowledge funding support from the project funded under contract №: KP-06-N57/4 from 16.11.2021, funded by Bulgarian National Science and funding support from SUMMIT project, No BG-RRP-2.004-0008, funded under by the European Union through the ESI Funds, and by the European Union-NextGenerationEU, through the National Recovery and Resilience Plan of the Republic of Bulgaria.

REFERENCES

- [1] Noninska, I., Romansky, R. E-government in digital age and organization of internet economy and e-business. *International Journal on Information Technologies and Security*, vol.17, no.3, 2025, pp. 61-72. <https://doi.org/10.59035/JWMN6023>
- [2] Boyana I., Bratanova, R. Limitation of black-market media influence through designing LLM tool for content analysis. *International Journal on Information Technologies and Security*, vol.17, no.1, 2025, pp. 13-24. <https://doi.org/10.59035/KDFD7294>
- [3] Lyubenova, S., M. Petrov, A. Aleksieva-Petrova. A graph database intrusion detection and prevention system. *The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM)*, vol. 29 (2024): 182-191.
- [4] Stefanov, M., M. Petrov, A. Aleksieva-Petrova. Analysis of drone security. *2024 International Conference on Computing in Natural Sciences, Biomedicine and Engineering (COMCONF)*. IEEE, 2024.
- [5] Grazhdanski, G., Vasilev, V., Vassileva, S., Taskov, D., Antova, I., Koychev, I., Boytcheva, S. SynthMedic: Utilizing large language models for synthetic discharge summary generation, correction and validation. *Journal of biomedical informatics*, vol. 170. 2025, 104906. [10.1016/j.jbi.2025.104906](https://doi.org/10.1016/j.jbi.2025.104906).
- [6] Llama 2, MetaAI, 30 March 2022. URL: <https://ai.meta.com/llama/> (Visited on 1.10.2025).
- [7] Models - OpenAI API, OpenAI, 30 March 2022. URL: <https://platform.openai.com/docs/models/> (Visited on 5.10.2025).
- [8] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G. LLaMA: Open and Efficient Foundation Language Models. 2023, 10.48550/arXiv.2302.13971.
- [9] RedPajama, a project to create leading open-source models, starts by reproducing LLaMA training dataset of over 1.2 trillion tokens, Together AI, URL: <https://ai.meta.com/llama/> (Visited on 25.11.2023).
- [10] Geng, Xinyang, Liu, Hao, OpenLLaMA: An Open Reproduction of LLaMA, 2023. URL: https://github.com/openlm-research/open_llama (Visited on 6.10.2025).
- [11] O. de Vel, Hubczenko, D., Kim, J., Montague, P., Xiang, Y., Phung, D., Murray, T. Deep learning for cyber vulnerability discovery: *NGTF Project Scoping Study*, 2019. URL: <https://www.dst.defence.gov.au/sites/default/files/publications/documents/DST-Group-GD-1039.pdf> (Visited on 6.10.2025).
- [12] Fang Y, Liu Y, Huang C, Liu L FastEmbed: Predicting vulnerability exploitation possibility based on ensemble machine learning algorithm. *PLoS ONE*, vol. 15, no. 2., 2020. <https://doi.org/10.1371/journal.pone.0228439>
- [13] Zhou, Y., Liu, S., Siow, J., Du, X., Liu, Y. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks, 2019. URL: <https://doi.org/10.48550/arXiv.1909.03496> (Visited on 6.10.2025).
- [14] Z. Li, D. Zou, S. Xu, H. Jin, Y. Zhu, Z. Chen. SySeVR: A framework for using deep learning to detect software vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, Jul-Aug 2022, pp. 2244-2258. doi: 10.1109/TDSC.2021.3051525.

- [15] Happe, A., Jürgen C. Getting pwn'd by AI: Penetration Testing with Large Language Models., 2023. URL: <https://arxiv.org/abs/2308.00121> (Visited on 6.10.2025).
- [16] Vaswani, Ashish. Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), 2017, Curran Associates Inc., Red Hook, NY, USA, 6000–6010.
- [17] McKee, Forrest, David Noever. Chatbots in a honeypot world, 2023. URL: <https://arxiv.org/abs/2301.03771> (Visited on 6.10.2025).
- [18] M. Sladić, V. Valeros, C. Catania, S. Garcia. LLM in the Shell: Generative Honeypots. 2024 *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, Vienna, Austria, 2024, pp. 430-435, doi: 10.1109/EuroSPW61312.2024.00054.
- [19] J. Ragsdale and R. V. Boppana. On designing low-risk honeypots using generative pre-trained transformer models with curated inputs. *IEEE Access*, vol. 11, 2023, pp. 117528-117545. doi: 10.1109/ACCESS.2023.3326104
- [20] Koide, Takashi, Naoki Fukushi, Hiroki Nakano, Daiki Chiba. Detecting phishing sites using ChatGPT, 2023. URL: <https://arxiv.org/abs/2306.05816> (Visited on 6.10.2025).
- [21] Gerganov., G. llama.cpp: Inference of LLaMA model in pure C/C++ URL: <https://github.com/ggerganov/llama.cpp> (Visited on 6.10.2025).
- [22] Wu, Hao, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. 2022. URL: <https://arxiv.org/abs/2004.09602> (Visited on 6.10.2025).

Information about the authors:

Veneta Yosifova – PhD from Faculty of Computer Systems and Technologies at the Technical University of Sofia, Bulgaria. Assistant professor at Department of Software Technologies at Sofia University. Areas of scientific research – Machine learning, Large Language Models and Cybersecurity. <https://orcid.org/0000-0002-4441-6417>

George Yosifov - PhD from Faculty of Mathematics and Informatics at Sofia University, Department of Software Technologies. Areas of scientific research – Computer Science, Big Data and Cybersecurity.

Manuscript received on 2 October 2025