BRETAGNE: Building a Reproducible and Efficient Training AI Gym for Network Environments

Yann Gourlet[1,2], Thomas Lefeuvre[1,2], Tobias Hürten[1,3], Johannes F. Loevenich[1,4], Florian Spelter[1,3], Erik Adler[1,5], Johannes Braun[1,6], Linnet Moxon[1,6], and Roberto Rigolin F. Lopes[1]
[1]Secure Communications & Information (SIX), Thales Deutschland, Ditzingen, Germany
[2]École Supérieure d'Ingénieurs de Rennes (ESIR), University of Rennes, Rennes, France
[3]Rheinische Friedrich Wilhelms Universität, Department of Computer Science IV, Bonn, Germany
[4]Department of Mathematics/Computer Science, University of Osnabrück, Osnabrück, Germany
[5]Medical Informatics, Heilbronn University of Applied Sciences, Heilbronn, Germany
[6]Department of Computer Science, University of Stuttgart, Stuttgart, Germany
Email: {tobias.huerten, johannes.loevenich, florian.spelter, erik.adler, roberto.rigolin}@thalesgroup.com

***Abstract*—This paper introduces BRETAGNE, a network simulation environment designed to serve as a training ground for autonomous defense agents using hybrid AI models in simulations. The platform integrates docker, a lightweight virtualization technology orchestrated by Kathara with widely used network protocols such as BGP, OSPF, HTTP, SSH and others to simulate production-like environments, thereby addressing the limitations of existing simulators that often lack applicability in real-world networks. We present a multi-agent architecture involving blue, red, green, and white agents, designed to create dynamic, communicative environments for training purposes. Furthermore, our work includes a comparative analysis of large language models (LLMs) for detecting cyber attacks, which highlights the benefits and constraints of using AI for autonomous decision-making in cybersecurity. Overall, the BRETAGNE framework offers a realistic and scalable solution for the training and deployment of autonomous agents in operational networks.**

***Index Terms*—Autonomous Cyber Defence, Multi-Agent Reinforcement Learning, Cyber Security, Network Virtualization**

I. Introduction

Recent literature has been training and testing Autonomous Cyber Defense (ACD) agents using gym environments hosting blue, red, green and white agents [1]. Generally, these gyms simplify the network environment and services to reduce memory usage and computing time [2]. However, Artificial Intelligence (AI) models may over-fit the gym making them less useful in realistic network environments. This fact motivated us to start the Building a Reproducible and Efficient Training AI Gym for Network Environments (BRETAGNE) project. The goal is to instrument a simulation environment with real Software Defined Networking (SDN) interfaces and IP dataflows to train and test teams of agents using Multi Agent Reinforcement Learning (MARL).

The increasing frequency and complexity of cyber threats highlight the importance of automating the protection of computer networks in critical infrastructures used by the military. Conventional cybersecurity methodologies, which depend on threat detection and manual responses, are becoming increasingly inadequate in the context of rapidly evolving attacks and the need to safeguard expanding and interconnected networks. In this context, AI, particularly Machine Learning (ML) models and Large Language Models (LLMs), presents a promising technology for the automation of network defense.

The BRETAGNE simulator is aligned with this objective, providing a realistic and efficient simulation platform for training autonomous defense agents. This platform addresses the shortcomings of existing simulation environments, which often oversimplify scenarios, limiting the applicability of trained models in real-world production environments.

This paper presents the key contributions of BRETAGNE as follows:

1) Choice of virtualization technique and explanation of how to implement it in BRETAGNE.
2) The challenges of generating realistic and random 'user' and 'attacker' network traffic.
3) Techniques used to interface different forms of AI to simulation, with presentation of the results obtained.

The rest of this paper is organized as follows. Section II examines existing research and the fundamental technologies utilized for network simulation and virtualization. Section III provides an in-depth examination of the design principles underlying the BRETAGNE framework. It describes the various simulated network architectures and delineates the roles of the different agents involved. Subsequently, Section IV presents the results of the experiments quantifying the performance of the LLMs and Section V describes the challenges encountered. In conclusion, Section VI discusses open challenges and future research directions.

II. Background

Recent investigations report the design, development, and usage of simulated/emulated environments to train and test AI models for network management and security. Nowadays, many experiments aimed at using AI to secure or improve networks have already emerged. Examples include CYBORG [2], CyberVAN [3], Microsoft's CyberBattleSim [4], and NASim [5]. All these environments are specially designed

to train and test ML models and to demonstrate that training autonomous agents for the network is possible. To this end, these simulators employ discrete event network simulation. This type of simulation allows for rapid training due to the large amount of available data, as well as abstraction of configuration and high scenario scalability, making it a strongly favored choice by researchers in recent years. However, the use of discrete event network simulators does not allow these research efforts to be transferred to production networks due to the large gap in data generated by a real network and a discrete event simulated network.

These facts explain why recent research focused on training autonomous agents using virtualized network environments. This is the case, for example, with NASimEmu, which uses Vagrant and VirtualBox, allowing for highly realistic simulation with full virtualization and flexibility with the infrastructure as code concept proposed by Vagrant. Nevertheless, this type of virtualization does not support simulating large architectures due to its enormous resource requirements limiting the training of agents to simple architectures. For this reason, many projects such as CyRIS [6] or CYBORGemu [2] use cloud computing, notably with Amazon Web Services (AWS), to be able to utilize large computing power. However, this solution is not applicable in the military domain since deploying infrastructure with realistic traffic is too sensitive, as it requires sharing confidential information with the cloud provider.

These facts motivated the development of BRETAGNE with the following objectives:

- To implement a realistic simulation using widely used protocols, allowing for a network with transmission problems, latency, congestion, and many other issues that can occur on a production network.
- Ease of use and ability to enrich the simulator by providing configuration abstraction and virtualization techniques, while offering the possibility to add services or attack patterns to different scenarios.
- Optimization of resources by using a lightweight, resilient, and realistic virtualization technique. This will make the simulator easier to scale and allow the simulation to run on other machines without needing disk images.

A. Choosing the virtualization technique

Kathara [7] is a lightweight and flexible network simulation platform designed to create virtualized network environments using Docker. It allows for the configuration and simulation of complex networks while offering a simple interface for creating network topologies. One of Docker's main advantages over other virtualization techniques like KVM, Xen, or VirtualBox is its speed and efficiency. Docker uses containers that share the host operating system's kernel, which significantly reduces memory usage and improves performance compared to full virtual machines.

Kathara leverages Docker containers to simulate network nodes, with each container running a real instance of a Linux operating system. This means that the nodes can function like real machines, allowing the installation and execution of network software without modification, creating a simulation environment that closely mirrors reality. In comparison, NS-3 and QualNet, although highly accurate in their theoretical models, may sometimes diverge from the reality of actual network implementations.

The flexibility of Kathara is based on its efficient management of Docker containers, enabling the quick creation, deployment, and destruction of complex network topologies without the overhead associated with full virtual machines. Unlike Packet Tracer, which is limited to Cisco environments and offers less realistic simulations of lower network layers, Kathara can be used to simulate a variety of multi-vendor scenarios with different network technologies. Additionally, Docker allows for easy scaling, which is crucial when testing the robustness of an AI against large-scale attacks.

Kathara integrates easily with other modern automation tools thanks to its Docker-based architecture, enabling automated generation of network topologies and attack injection, for example, using Docker containers like Metasploit. Kathara also offers the possibility of hybridization. This allows for connecting a simulation with physical equipment or simulating with another virtualization technique. It thus provides the opportunity to incorporate licensed tools (such as complete Cisco solutions, for example) into the simulation.

III. Design

A. Network environments

The network scenario we have chosen is that used in Cage Challenge 4, as defined in [8]. The network consists of two theater networks, shown as 'A' and 'B' in Fig. 1, and a metropolitan network, shown as 'C', interconnected by a traditional carrier network. In addition, a corporate network is connected to the same operator. This scenario is interesting because of its similarity to a military network using an unsecured carrier network with potentially malicious users.

The scenario is described in a Python program that employs generic functions to rapidly and efficiently generate a variety of network architectures. The virtualisation and interconnection of each node is ensured by Docker containers, which are orchestrated by Kathara. The routers utilize FRR 9.0.1, the switches employ Open vSwitch 3.0.1, the hosts are based on a Linux kernel 6.8.0-39, and the SDN controller relies on Floodlight 1.2, which manages the firewall. The routing protocols employed are BGP in the operator network and OSPF in other networks. The utilization of widely deployed network implementations and routing protocols serves to enhance the fidelity of the scenario, closely resembling the behavior of numerous architectures employed in production environments. Using these routing protocols will also enhance the robustness of learning, as the convergence time of the protocols and the transmission issues they may cause can lead to numerous potential false

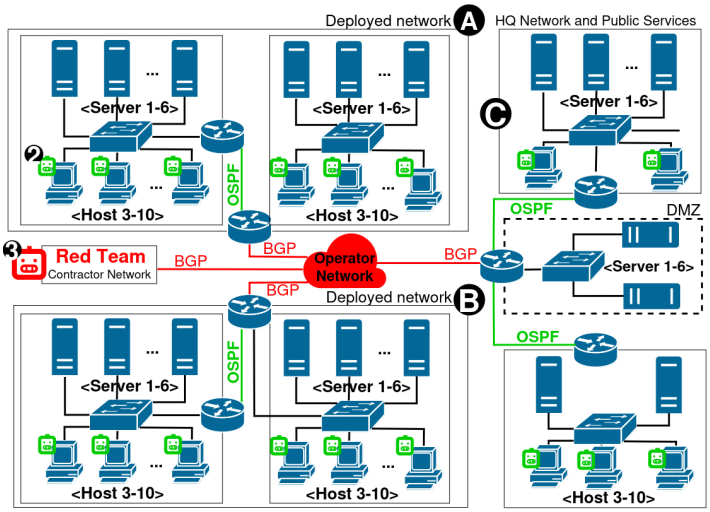

Fig. 1: Network architecture described in Cage challenge 4 simulated by BRETAGNE

positives. Additionally, this allows for other attacks, such as route injection, prefix hijacking, and many others that exploit vulnerabilities in these protocols.

A fully virtualized architecture allows us to achieve traffic patterns that are similar to those observed in physical network implementations. One of the challenges inherent to such an architectural configuration is the ability to extract all traffic. In this project, the utilization of port monitoring on Open vSwitch (OvS) in conjunction with TCPDUMP, enables the capture of real-time traffic from each network on the host machine.

The second challenge is to generate authentic user traffic on the network. It is therefore evident that in order to train an AI in the most realistic environment possible, it is necessary to provide it with realistic user activity on our network. This is the role of the green agent, represented as number 2 in Fig. 1. This green agent is deployed on each user machine and has a number of available actions, including accessing a web page, testing connectivity, file exchange, and SSH connection, among others, that can be extended. The green agent executes these actions randomly, for a random duration, and towards a random server. The number of simultaneous connections is not limited, which can lead to issues with user traffic retransmission, thereby creating additional potential patterns of false positives for the learning of the blue agent.

This network architecture is dynamic and controllable via command line. Users can deploy or remove a red or blue agent from the network at any time, as well as enable or disable traffic capture on the network. Many other actions can also be performed manually using the command line, such as launching an attack, generating user traffic, or directly opening a terminal on a machine within the simulation.

B. Red agent

In this simulation, the attacking team is represented by a red autonomous agent, shown as 3 on Fig. 1. In a cyber range environment, its role is to simulate attacking team. It aims to identify vulnerabilities and assess the response of defense teams to these attacks. This agent is based on Metasploit, a framework for developing and executing exploits that offers a multitude of possible attacks. In this simulation, it can be utilized in three ways:

- Manually via the Docker container console.
- Automatically and randomly using the BRETAGNE command.
- Orchestrated by following the instructions of the white agent.

C. Blue agent

The role of the blue agent is to analyze network traffic and make decisions based on the observed network elements. The main actions that the blue agent can take are blocking traffic, allowing traffic, or resetting a host. These actions can be easily implemented in our architecture since it uses a centralized SDN controller that enables flow management throughout our simulation. The use of this technology also simplifies the architecture, as a single connection to the web interface of the controller on a separate management plane provides complete control and enhances security by reducing the attack surface compared to using multiple links.

D. Blue agent using LLM

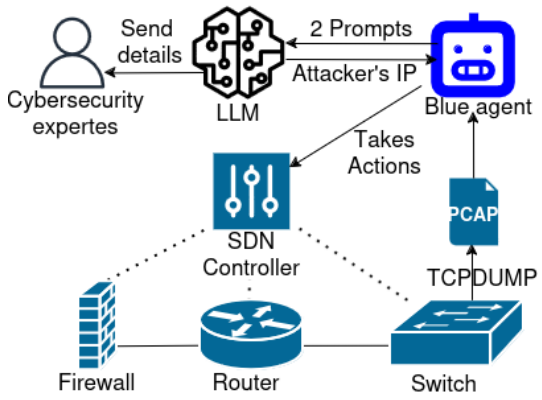

Fig. 2: Structure of the blue agent using an LLM

Fig. 2 shows the LLM and the set of network devices (e.g. switch, router, firewall, controller) used by the blue agent. The agent captures network traffic in pcap format generated by TCPDUMP, formats the data, and adjusts the prompt parameters to send the IP traffic to the LLM for classification (attack or no-attack).

The prompt serves as a crucial tool to guide the model toward generating precise, relevant, and concise responses. In our study, we use two prompts, one for the cybersecurity experts, which is very simple and allows for providing as much detail as possible about the situation, and one for the blue agent formatted as follows: *"I have network data captured on our network. Could you analyze this network traffic and say whether or not an attack has taken place? The*

answer should be as short as possible: yes or no with the attacker IP." This prompt is designed to be clear, concise, and specifically oriented toward the task of detection. It ensures that the model focuses on the essential task ofdetermining if an attack has occurred and, if so, identifying the attacker's IP address. Additionally, we limited the number of tokens to 20 to keep the response concise, while a temperature of 0 ensures the response is the most probable and accurate. The top_k parameter,when set to the maximum, increases the model's options and improves the chances of finding the best response in the given context. This approach yields results that can be easily processed by the blue agent for decision-making. The results section provides more details on the outcomes achieved with this technique and discusses the potential limitations observed.

E. White Agent

In a gym environment or cyber range, a white agent refers to agents with complete knowledge and visibility of the network environment. Their primary function is to:

- Monitor and Evaluate: Observe the actions of other participants (often red teams simulating attackers), assess their tactics, techniques, and procedures, and evaluate the effectiveness of defensive measures.
- Provide Feedback and Guidance: Compute rewards, insights and recommendations to blue agents based on defensive/offensive actions, helping the teams to improve their cybersecurity policies and strategies.
- Control Scenario Progression: In some cases, white agents may manipulate the cyber range scenario to introduce new challenges or test specific defensive capabilities of blue agents.

F. Blue agent using ML

Another way to automate the blue agent's decision-making is to use ML models, as shown in Fig. 3. In this diagram, the white agent represented as number 1 is a crucial component of the system. Indeed, it acts as the game master. This agent orchestrates the user traffic (green agent) and attacks (red agent). The white agent randomly decides whether an attack will occur and then randomly selects a target and an IP address within the network where the red agent is deployed. This approach allows for a greater number of scenarios and prevents the model from making correlations based on the attacker's IP address. Each attack is then described as a function and randomly selected by the program, resulting in a wide variety of patterns for training the model.

Using a white agent with a global view of the entire scenario will allow for greater training possibilities for the blue agent's model. Indeed, without the white agent, the available data is limited to what is captured on the network, lacking specific context, limiting training opportunities to unsupervised learning. While this type of training offers advantages such as flexibility, it also presents significant

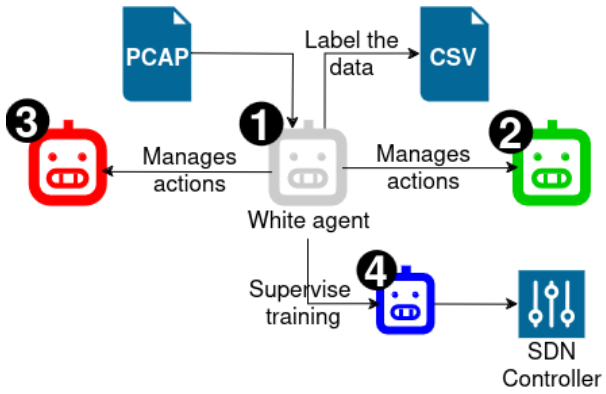

Fig. 3: Structure of the blue agent using Machine Learning

drawbacks in terms of implementation, model evaluation difficulties, and data interpretation.

The white agent will enable data labeling as it has a complete view of the simulation. To do this, it will associate each packet or group of packets with the attacking machine's address, the victim machine's address, and a flag indicating whether an attack has occurred. This data can then be saved in a CSV file, making it easier to process in a Python program.

This labeled data will facilitate the implementation of supervised learning for labeling each packet or multi-instance learning (MIL) for groups of packets. This type of learning will enhance precision, evaluation ease, and interpretability of results.

In other configurations, the white agent enables Reinforcement Learning (RL). In this method, the blue agent learns to make decisions by interacting with an environment. It receives rewards or penalties based on its actions, allowing it to learn to maximize rewards over time. The goal is to optimize a strategy to achieve the best possible outcome. In our configuration Fig. 3, the white agent knows the exact state of the simulation through PCAP network captures and the state of other agents. This allows the agent to timely compute rewards with precision.

IV. Results

A. LLM assessment

This section reports a comparative analysis of five LLMs implementing the decision making of a blue agent. The choice of an LLM depends on its performance generating two types of responses: one identifying cyber attacks in IP data flows, and second generating explanations about network events to assist a cybersecurity expert. The evaluation focuses on the precise response intended for the blue agent, while the LLM is trying to identify cyber attacks in the logs. Furthermore, the response for the cybersecurity expert aims to guide research or alert about potential dangers. This response will, in any case, be verified by a human, requiring more detail but less precision than the response expected by the blue agent. This behavior is easier to achieve with a LLM than a simple and concise result. Therefore, the evaluation here is conducted on

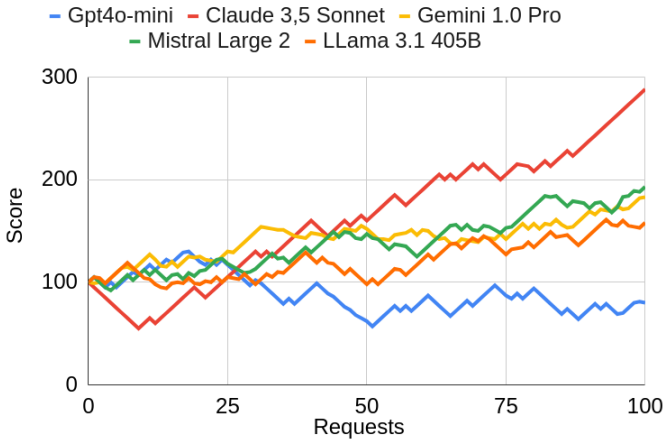

Fig. 4: Score as a function of requests for 5 LLMs.

the response expected by the blue agent based on the requests made to the LLM.

For our evaluation, we use the prompt technique explained earlier in Section III-D (*"Design/Blue agent using LLM"*). The main evaluation criterion is the accuracy of the responses generated by the different LLMs. To assess these responses, we established a scoring system to quantify the models' accuracy:

- If no attack is launched and the LLM returns that no attack is detected: +5
- If an attack is launched and the LLM returns the attacker's IP: +5
- If an attack is launched and the LLM returns that no attack is detected: -1
- If an attack is launched and the LLM returns an incorrect IP: -3
- If no attack is launched and the LLM returns an IP: -5
- If an attack is launched and the LLM indicates that an attack has been detected but without an IP: +1

To obtain representative results possible, we performed 100 prompt requests for each LLM. The analysis of the responses was automated by the white agent which calculated the scores for each LLM, displayed the progress and final results. This functionality is similar to that used by the white agent to oversee the reinforcement learning of the blue agent. The initial score for the test is set at 100. If a LLM's score reaches zero, the test is immediately terminated, and the model is eliminated.

Fig. 4 plots the evolution of scores as a function of requests for each LLM (different lines). This plot shows that most models have scores that trend positively, except for GPT-4o mini. This indicates that the number of correct responses exceeds the number of incorrect ones. However, there is considerable variation for each model, suggesting that the number of incorrect responses is also present and relatively consistent. In summary:

- Claude 3.5 Sonnet provided the highest number of correct responses during our tests. However, it is also the most expensive model to use and demands high availability of computing power and memory capacity, which poses a significant issue for military applications at the tactical edge.
- Mistral Large 2 also shows good results with accurate responses and can develop as requested. Additionally, it can be installed locally and fine-tuned, making it the most attractive model for deployable networks.
- Llama 3.1 405B also shows very good results, comparable to those of Mistral, and can be installed locally and fine-tuned. However, its responses are long and have lower precision, if compared to Mistral's.
- GPT-4o mini shows rather poor results in this study due to a high number of false positives, which are heavily penalized in our scoring system. It is unfeasible to block potentially critical communications in a military context. Additionally, the demand for computational and memory resources limits the use of this model at the tactical edge at time of writing.

As a result, this comparative study supported the decision to select Mistral Large 2 (2407) for interaction with our blue agent, as this model can address most constraints posed by our network scenario.

V. Discussion

This investigation employed a LLM to assist cybersecurity experts and network administrators in detecting cyber attacks. This setup can complement a supervision or Intrusion Detection System (IDS) by generating additional alerts or providing more detailed information about intrusions. However, our initial findings regarding the use of LLMs to automate decision-making on the network, such as blocking traffic or allowing flows, demand further investigation. Indeed, several limitations to the use of LLMs have been observed:

- The challenge of generating concise responses that can be easily processed by a machine.
- The lack of accuracy in the generated responses.
- The high likelihood of generating false positives, making autonomous decision-making in a network very challenging.
- The significant demand for resources, or the reliance on cloud models, which requires a high level of trust in the cloud provider.

To improve the results obtained from LLMs, one solution would be to fine-tune the models to train them for detection and appropriate action-taking. However, this process requires significant resources because of the size of LLMs. Therefore, the most advantageous method for autonomous decision-making is to use machine learning to train a specialized model. Initial work aimed at training this type of model has already been conducted in [1].

Our primary objective has been to reuse these models and train them on a realistic simulation by creating an interface that abstracts away the simulation and generates data similar to CYBORG. While useful, this method abstracts away actual

IP data flows in the network, which means the model will be much less adaptable to new attacks and potential different patterns of packets emitted by an attacker.

An alternative approach to train a model for autonomous decision-making on the network is to use machine learning on real data generated by the network (logs, network captures, etc.). However, this method requires a substantial amount of data to ensure effective training of a high-performing model. This is why many studies in this area utilize discrete event simulation, which can yield a large volume of data.

By employing virtualization techniques, the network behaves realistically (albeit slowly), making it difficult to expedite training. One solution is to increase the network architecture size or simulate multiple identical network architectures in parallel to obtain more potential training data. Using a lightweight virtualization technique, such as containerization, facilitates this scalability, allowing us to generate significant amounts of realistic data with minimal resources.

Despite this, our research has shown that SDN is a very useful technology for integrating an autonomous blue agent with a network controller. This technology enables complete software-based and centralized control of the network, facilitating the quick integration of intelligent agents without significant modifications to the network architecture or policies.

Deploying AI models directly on the SDN controller would be a cost-effective solution for implementing this technology for autonomous decision-making in networks. However, a more sustainable solution would be to develop an autonomous blue agent using a lighter protocol implementation to communicate with the various equipment in the network, as it has been shown that the use of SDN requires a significant amount of bandwidth on the control plane to function.

Regarding the simulator we developed, it effectively meets the requirements of an easy-to-use demonstrator with an intuitive command-line interface and documentation to reproduce the experiments described in this paper. However, many additional features could be added:

- A modeling feature that allows users to quickly and visually create their own network architecture with the appropriate protocols.
- Proposing other types of connections, such as radio links, by incorporating EMANE [9].
- Adding a modeling interface for scripting user interactions or replayable attacks.

Nonetheless, the absence of these features is not limiting, as modifications to the architecture or the description of actions on the network can easily be accomplished using the functions available in the source code.

VI. Conclusion

This document outlines an ongoing investigation aimed at designing and developing a simulation environment for training and testing teams of autonomous agents utilizing hybrid AI models, including Multi-Agent Reinforcement Learning (MARL). Our solution integrates lightweight virtualization technologies, such as Docker, to implement a complex and interoperable multi-agent architecture. This approach has enabled the creation of a realistic and scalable simulation environment that closely mirrors the complexities of real-world networks, particularly those with critical security requirements, such as military infrastructures. Ultimately, our goal is to develop a robust, scalable, and efficient platform for training AI-driven cybersecurity defense systems that can be directly applied to operational networks.

We conducted a comparative analysis of five large language models (LLMs) for network traffic analysis. Our findings highlight the potential of LLMs in supporting cybersecurity operations, particularly by assisting human experts with detailed analyses and insights into network traffic. However, when it comes to autonomous decision-making, the limitations of LLMs become apparent. These include challenges in generating concise, machine-processable responses, a lack of accuracy in identifying and responding to threats, and significant resource demands for deployment. Additionally, the reliance on cloud-based LLMs poses substantial security risks in contexts where data privacy is paramount, such as military networks.

References

[1] J. F. Loevenich, E. Adler, R. Mercier, A. Velazquez, and R. R. F. Lopes, "Design of an Autonomous Cyber Defence Agent using Hybrid AI models," in *2024 International Conference on Military Communication and Information Systems (ICMCIS)*, Koblenz, Germany, 2024, pp. 1–10.

[2] M. Standen, D. Bowman, S. Hoang, T. Richer, M. Lucas, R. V. Tassel, P. Vu, M. Kiely, K. C., N. Konschnik, and J. Collyer. (2022) CybORG: Cyber Operations Research Gym. Available: https://github.com/cage-challenge/CybORG. [Accessed: 20 May 2024].

[3] R. Chadha, T. Bowen, C.-Y. J. Chiang, Y. M. Gottlieb, A. Poylisher, A. Sapello, C. Serban, S. Sugrim, G. Walther, L. M. Marvel, E. A. Newcomb, and J. Santos, "Cybervan: A cyber security virtual assured network testbed," in *MILCOM 2016 - 2016 IEEE Military Communications Conference*, 2016, pp. 1125–1130.

[4] "Cyberbattlesim," https://www.microsoft.com/en-us/research/project/cyberbattlesim/, 2020, created by William Blum, James Bono and Jugal Parikh.

[5] "Network attack simulator," https://networkattacksimulator.readthedocs.io/, 2020, created by Jonathon Schwartz.

[6] R. B. Cuong Phan, Dat Tang and K. ichi Chinen, "Cyber range instantiation system for facilitating security training," Japan, 2016, pp. 1–2.

[7] L. A. Mariano Scazzariello and T. Caiazzi, "Kathará: A lightweight network emulation system," Rome, Italy, 2020, pp. 1–2.

[8] T. C. W. Group, "Ttcp cage challenge 4," https://github.com/cage-challenge/cage-challenge-4, 2023.

[9] U. N. R. Laboratory. (2023) Extendable Mobile Ad-hoc Network Emulator (EMANE). details: https://www.nrl.navy.mil/Our-Work/Areas-of-Research/Information-Technology/NCS/EMANE/.