Cyber-range green team emulation as code

Targeted Attack Emulation Using Customized Language Models

BAKHAT ILYAS
Author2

Research and Development project owner:
Dr Jérôme Dossogne PhD

Master thesis submitted under the supervision of
Dr Jérôme Dossogne PhD

the co-supervision of

in order to be awarded the Degree of
Master in Cybersecurity
Corporate Strategies

Academic year
2024 - 2025

Appendix D

Initial project proposal

D.1 Title

Title: Cyber-range green team emulation as code // Targeted Attack Emulation Using Customized Language Models

D.2 Proposal summary

D.2.1 Participants

Project Director/Owner: Dr Jérôme Dossogne PhD
Researchers(s):

- Bakhat Ilyas
- ??

D.2.2 Background

Virtualization

Virtualization is a foundational technology in modern computing, enabling multiple operating systems and applications to run on a single physical machine by abstracting the hardware. This allows for optimized resource use, flexible deployment, and enhanced scalability in cloud and DevOps environments.

Technical Overview

- **Hypervisors**: Hypervisors are the core of virtualization, creating and managing virtual machines (VMs) by abstracting hardware resources. There are two primary types:
  - **Type 1 (Bare-Metal) Hypervisors**: These operate directly on the hardware, providing high performance and efficient resource management. Examples include VMware ESXi and Microsoft Hyper-V. They are widely used in data centers and cloud environments due to their direct hardware access and minimal overhead [4].
  - **Type 2 (Hosted) Hypervisors**: These run on top of an operating system, making them more suitable for desktops and development environments. Examples include Oracle VirtualBox and VMware Workstation. Although easier to set up, they may have performance limitations compared to Type 1 hypervisors [1].
- **Resource Abstraction**: Virtualization abstracts compute resources like CPU, memory, storage, and networking, allowing multiple isolated VMs to share the same physical hardware. This abstraction provides security and isolation, as each VM operates independently, often with distinct operating systems [**?**].

- **Container-Based Virtualization**: Unlike traditional VMs, container-based virtualization shares the host OS kernel, making containers lightweight and efficient. Technologies such as Docker and Kubernetes use this model to deliver faster deployment, enhanced scalability, and minimal resource consumption, particularly beneficial in cloud-native applications [17].

- **Snapshots and Cloning**: Virtualization allows for snapshots and cloning, which are essential for backup, testing, and disaster recovery. Snapshots capture the exact state of a VM at a given point, while cloning creates exact duplicates, facilitating environment replication and rollback [26].

- **Virtual Networking and Storage**: Virtualized environments support advanced networking setups (e.g., virtual switches) and storage (e.g., storage area networks) that enable isolated networking and flexible storage allocation, critical for scalability and redundancy in virtualized infrastructure [25].

- **Resource Efficiency and Scalability**: Virtualization allows hardware consolidation by running multiple VMs on a single host, reducing energy and operational costs. This flexibility enhances scalability, as additional VMs can be created as demand grows, aligning resources dynamically with application requirements [29].

Docker

Docker is a leading platform for containerization, providing an efficient method for running applications in isolated environments called containers. Developed by Docker, Inc., the platform enables seamless creation, management, and orchestration of containers, forming a foundation for deploying applications across diverse environments with consistency and efficiency.

- **Container Basics:** Containers are a lightweight, efficient alternative to virtual machines (VMs). Unlike VMs, which require a separate OS for each instance, containers share the host OS kernel, reducing overhead and enabling faster deployments with minimal resource consumption. Docker containers, in particular, bundle applications with essential libraries and dependencies, ensuring consistent runtime behavior across systems.

- **The Docker Engine:** The Docker Engine is the core component of the Docker platform, comprising the Docker Daemon, a high-level container runtime (containerd), and a low-level runtime (runc). These work together to manage the container lifecycle, including starting, stopping, and monitoring. The modular design of Docker is aligned with the Open Container Initiative (OCI) standards, ensuring compatibility with other container tools. The current Docker Engine's architecture is shown in the figure D.1.

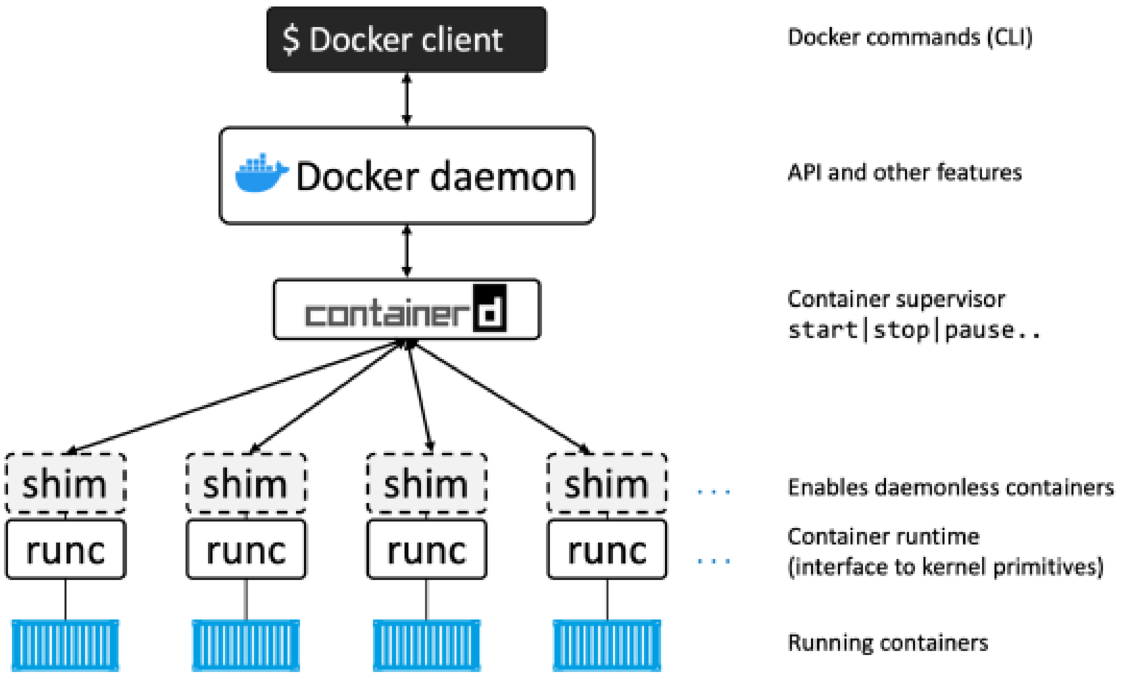


Figure D.1: A high-level view of the current Docker engine architecture [23]

- **Images and Containers:** Docker images are read-only templates from which containers are created, containing application code, dependencies, and a minimal OS layer. Each image consists of multiple layers that combine at runtime, allowing for efficient storage and reusability while minimizing redundancy across containers. [23]

- **Networking and Orchestration:** Docker includes networking solutions that facilitate container communication. This includes bridge networks for host-based networking and overlay networks for distributed applications. Docker Compose and Docker Swarm enable deployment and management of multi-container applications, simplifying orchestration in scenarios where containers work together as part of a larger system.

- **Security Features:** Docker incorporates security measures through container isolation mechanisms such as namespaces and control groups, ensuring that containers operate independently. It also integrates Transport Layer Security (TLS) for secure communication between nodes in a Docker Swarm, alongside access control and image signature verification to maintain application integrity. [23]

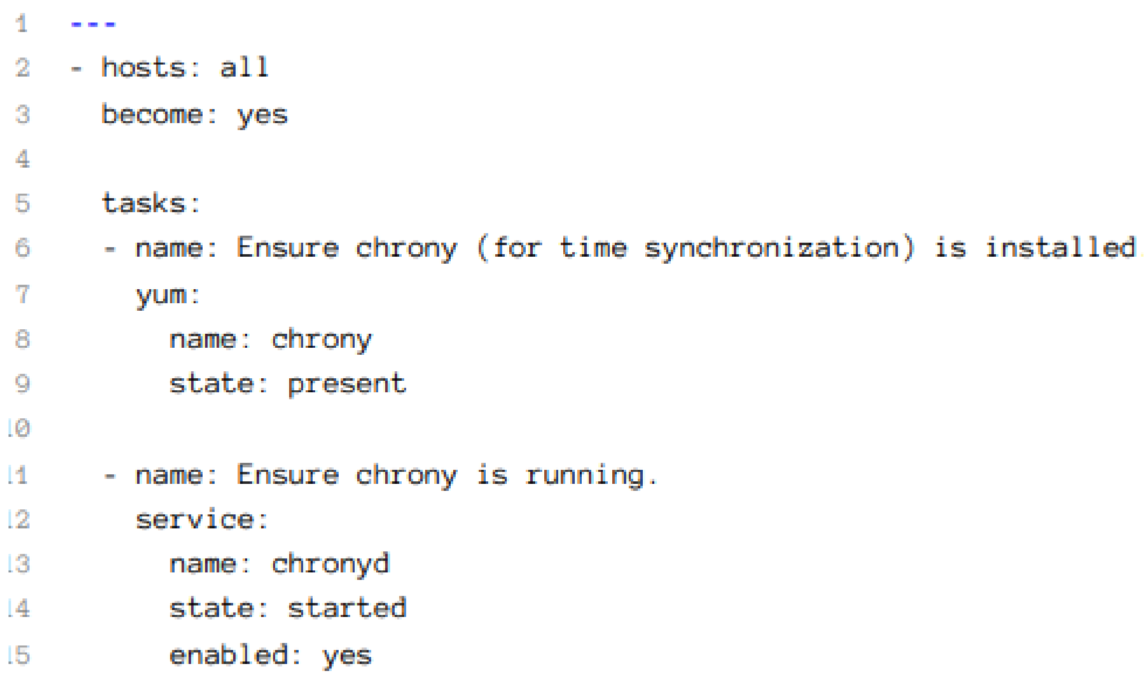
Docker's architecture supports modern development practices like DevOps and microservices by enabling organizations to achieve portability and scalability in application deployment, providing a reliable way to transition applications from development to production environments without compatibility concerns.

Ansible

Ansible is an open-source automation tool widely used for configuration management, application deployment, and infrastructure orchestration. Designed to be agentless, Ansible operates through secure SSH connections, making it efficient and lightweight across diverse environments. With its YAML-based configuration files, known as playbooks, Ansible provides a streamlined, human-readable syntax that facilitates automation. [6]

Technical Overview

- **Playbooks and YAML Syntax**: Playbooks are the core of Ansible's operations. They are written in YAML and contain sequences of plays, which in turn consist of tasks executed on defined hosts. Each task specifies a module—such as `yum` or `service`—to manage resources and services on remote nodes. Playbooks thus offer a systematic approach to automation by ensuring ordered and repeatable tasks. The figure X shows an example of a Playbook Syntax :



```
---
- hosts: all
  become: yes

  tasks:
  - name: Ensure chrony (for time synchronization) is installed
    yum:
      name: chrony
      state: present

  - name: Ensure chrony is running.
    service:
      name: chronyd
      state: started
      enabled: yes
```

Figure D.2: Example of YML file [6]

- **Inventory Management**: Ansible's inventory file identifies the target hosts, which can be organized into groups for more granular control. Each host or group can have specific variables, supporting flexibility across environments. Additionally, dynamic inventory scripts can automate host discovery from cloud providers, integrating seamlessly with cloud-based infrastructure.

- **Modules and Ad-Hoc Commands**: Ansible provides a comprehensive library of modules covering a wide range of operations, from networking to file management. These modules are reusable scripts that execute discrete tasks, allowing for standardized configurations. Ad-hoc commands further enhance Ansible's utility by enabling direct module execution from the command line, suitable for quick adjustments or debugging without needing a full playbook.

- **SSH and Connection Handling**: Ansible relies on SSH to securely interact with managed nodes. By default, it uses OpenSSH, with Paramiko as an alternative option. To optimize task execution, Ansible supports features like pipelining, which reduces connection overhead by transferring fewer files during task execution, activated by setting `pipelining=True` in the configuration.

- **Roles and Best Practices**: Ansible's "roles" feature provides a structured approach for organizing tasks, variables, and handlers. Roles encourage modularity and reusability by dividing configurations into structured directories. For instance, default variables can be stored in `defaults/main.yml` and adjusted as needed, while role-specific variables reside in `vars/main.yml`, ensuring consistent, manageable configurations.

- **Security and Ansible Vault**: Ansible Vault is a built-in feature for encrypting sensitive information, such as credentials or API keys, within playbooks. This security measure utilizes AES-256 encryption and can be managed via password files or scripts, providing confidentiality during automated tasks.

- **Advanced Features and Use Cases**: Ansible integrates with tools like Vagrant for local development and testing. Additionally, options such as `-check` mode allow for dry runs to validate playbooks without executing changes, and `-become` enables privilege escalation when administrative rights are necessary. Real-world playbooks often include complex setups, such as deploying a Node.js application on a CentOS server, where Ansible manages the entire process—from package installation to configuration and operational maintenance.

- **Ansible Molecule** Molecule is a specialized testing framework for Ansible roles and playbooks that ensures infrastructure code functions as intended before deployment. It enables developers to simulate and verify configurations across multiple environments, thus aligning with Infrastructure as Code (IaC) best practices by incorporating automated testing into infrastructure management workflows. Molecule allows the use of various drivers, including Docker, VirtualBox, and cloud platforms, for versatile testing setups.

  Using a "test-driven development" (TDD) approach, Molecule facilitates testing at each stage of an Ansible role's lifecycle—from initial development through to deployment. The framework includes support for linting, syntax checks, and idempotence testing, where each run achieves the same end state without altering existing configurations if rerun. This aspect is particularly useful for preventing unexpected infrastructure changes, ensuring Ansible roles are consistent and predictable (Geerling, Ansible for DevOps).

  Molecule operates with a modular structure and integrates CI/CD tools and platforms like GitHub Actions, Jenkins, and GitLab, streamlining continuous deployment for IaC projects. This focus on modularity also allows for integration with Ansible's extensive plugin ecosystem, such as Ansible Lint, which ensures adherence to syntax and style standards. Molecule's comprehensive testing cycle reflects its emphasis on robust and adaptable infrastructure configuration, addressing the demand for reliable automation tools in modern DevOps practices

Vagrant

Vagrant is an open-source tool developed by HashiCorp that automates the creation, configuration, and management of virtual environments, supporting DevOps practices through consistent and reproducible environments. By abstracting interactions with virtualization providers, Vagrant enables rapid setup of complex environments for development and testing.

Technical Overview

- **Virtualization Abstraction**: Vagrant acts as a cross-platform interface for managing virtual machines. It integrates with providers such as VirtualBox, VMware, Docker, and Hyper-V, enabling users to define and manage environments without needing to directly interface with the virtualization software. [9]

- **Configuration with `Vagrantfile`**: The `Vagrantfile` is Vagrant's primary configuration file, defining the base image, network settings, provisioning scripts, and shared folders. Written in Ruby, the `Vagrantfile` supports version control, enabling reproducible environments across different development setups. [11]

- **Provisioning Capabilities**: Vagrant can provision VMs using tools like Ansible, Chef, and Puppet, automating software and dependency setups. This integration supports Infrastructure as Code (IaC) practices, as configurations can be scripted and applied consistently across environments. [10])

- **Multi-Machine Configurations**: Vagrant enables the orchestration of multiple virtual machines within a single environment, supporting complex infrastructure setups (e.g., database and web servers) defined in one `Vagrantfile`. This feature is crucial for testing distributed systems in environments that resemble production setups. [8]

- **Environment Consistency and Cross-Platform Compatibility**: Vagrant ensures consistent setups across different development machines by standardizing configurations in the `Vagrantfile`. This portability reduces configuration drift, making Vagrant valuable for team collaboration and continuous integration workflows. [5]

Large Language Models (LLMs)

Large Language Models (LLMs), like GPT-3 and GPT-4, are advanced neural networks designed to understand and generate natural language. They rely on the transformer architecture, introduced by Vaswani et al. in 2017, which brought a breakthrough in natural language processing (NLP) by handling long-range dependencies more effectively. Here's an in-depth explanation of the key components and stages involved in the construction and training of these models, including each concept's purpose and function. [28]

- **Transformer architecture:** The transformer architecture is the foundation of modern LLMs. Unlike RNNs (Recurrent Neural Networks) and LSTMs (Long Short-Term Memory networks), the transformer can process entire sequences in parallel rather than sequentially. This is achieved through an attention-based mechanism and makes transformers highly scalable for large language models. Transformers consist of two primary blocks: an encoder and a decoder, although most generative language models only use the decoder block. [28]

- **Attention mechanism:** The attention mechanism allows the model to focus on specific parts of the input sequence. The transformer uses "multi-head attention," where multiple "attention heads" each focus on different aspects of the sequence. For each word in the sequence, the model calculates three vectors: query, key, and value. The similarity between query and key vectors determines the attention weight for each word, allowing the model to weigh words differently depending on their relevance. These weights are then applied to the value vectors, creating a context-aware representation for each word. [28]

- **Causal self-attention:** In autoregressive LLMs, such as GPT-3 and GPT-4, a variant called causal self-attention is used. In this setup, each word in the sequence can only attend to the words that came before it. This ensures that the model does not "see the future" when predicting the next word, making it suitable for sequential generation tasks like text generation. [15]

- **Layer normalization and residual connections:** Transformers use layer normalization and residual connections around the attention and feedforward layers. Layer normalization stabilizes the model by scaling inputs within a layer, which speeds up training and prevents gradient issues. Residual connections, on the other hand, are used to prevent information loss across layers by adding the input of each layer directly to its output.

- **Feedforward layers:** Each attention block is followed by a feedforward neural network that applies a non-linear transformation to each position's representation independently. This helps the model to refine and enhance the contextual representation of each word.

- **Embedding and positional encoding:** LLMs require an embedding layer to convert words or tokens into dense vectors of fixed dimensionality. This layer captures semantic information, enabling the model to understand relationships between words (like synonyms or associations). Since transformers do not inherently interpret the order of words, positional encoding is added to embeddings to encode positional information. This can be done through sinusoidal or learned positional encodings, which help the model distinguish the positions of tokens within a sequence.

- **Large-scale training phase:** Training LLMs involves vast datasets and extensive computational power. This training phase has several critical steps. Models are initially trained on a vast corpus of data through language modeling tasks—most often an autoregressive objective (predicting the next word in a sequence). This approach leverages large datasets (e.g., extensive text from the internet) and requires significant computational resources. Optimization is usually achieved using stochastic gradient descent with advanced optimizers, such as Adam, to update model weights.

- **Fine-tuning:** Following pre-training, models are often fine-tuned on specific tasks or datasets to improve performance in targeted applications. Fine-tuning can involve supervised fine-tuning (SFT) or reinforcement learning techniques, like Reinforcement Learning from Human Feedback (RLHF). This latter approach uses feedback from humans to adjust model outputs toward desired responses, making the model safer and more aligned with human preferences.

- **Memory and long-sequence management:** Managing long sequences efficiently is essential for LLMs, as the computational cost increases quadratically with sequence length. Various methods have been introduced to optimize memory usage, including sparse attention or local attention windows, which limit the attention to nearby tokens, reducing computation while preserving the model's capacity to handle long-range dependencies.

- **Regularization techniques and precision control:** LLMs utilize various regularization techniques such as dropout (which randomly drops neurons during training) and weight decay (which reduces weight magnitudes) to prevent overfitting. Additionally, weight quantization and mixed-precision training (using lower precision arithmetic) can be used to reduce memory usage and increase training efficiency, which is vital for handling models with billions of parameters.

- **Deployment and inference optimization:** Once trained, LLMs must be optimized for deployment in production environments. The inference process can be computationally demanding, so it's often enhanced through techniques like weight pruning or knowledge distillation. Weight pruning removes redundant parameters, and knowledge distillation transfers knowledge from a large model to a smaller, faster model. Caching intermediate computations is also used to provide faster responses, which is especially useful for real-time interactions.

- **Ethics and safety measures:** Ethics and safety are paramount in LLMs due to their potential to generate harmful or biased outputs. Content filtering and guardrails are implemented to minimize these risks, ensuring that the model adheres to guidelines and avoids unintended content generation.

- **Conclusion:** In summary, LLMs are complex systems that rely on innovations in attention mechanisms, large-scale training on vast datasets, and optimizations for deployment. These models require immense computational infrastructure to train and deploy, but they enable highly sophisticated language understanding and generation capabilities. Each component, from embedding to inference optimization, contributes to the LLM's ability to perform a range of natural language tasks.

GHOST: Deploying NPCs with Targeted Large Language Model Integration

The GHOSTS (Generating Host Objects for Simulated Training and Scenarios) framework, developed by Carnegie Mellon University's SEI, is an open-source tool that creates and manages simulated NPC (non-player character) behaviors on a network, designed primarily for cyber training and experimentation. Through its flexible API, GHOSTS supports orchestrating complex scenarios that mimic real-world network behaviors, allowing cybersecurity teams to test responses, analyze vulnerabilities, and improve defense mechanisms in realistic settings. [14]

GHOSTS API and Components The GHOSTS API acts as the central controller, managing NPC activities across different client machines. It operates on Docker containers, making it easier to deploy and configure in varied environments. The API can handle multiple scenarios simultaneously by using "timelines" that dictate specific actions for NPCs, such as web browsing, document creation, and email interactions. These activities can be randomized or tailored to simulate particular user behavior patterns, enhancing the realism of cybersecurity exercises. [13]

Key Components API Server: The API server coordinates NPC activities, manages client connections, and facilitates scenario timelines. It's often used with a database (now PostgreSQL) for data persistence, ensuring reliable storage of activity logs and configurations. Clients: GHOSTS clients operate on Windows and Linux and act as the simulated "hosts" performing tasks. Each client can be configured for different actions, such as web browsing, sending emails, or creating documents, making it versatile for replicating various user behaviors.

Grafana Integration : The GHOSTS API integrates with Grafana for visualizing activity data, which is crucial for monitoring and analyzing scenario outcomes in real-time or post-exercise.

How It Works Through its structured timeline approach, the API schedules and synchronizes tasks across NPCs, using WebSockets to maintain continuous communication between the server and clients. This minimizes latency and allows for near-real-time activity orchestration. Additionally, GHOSTS provides a user interface for managing machine groups and timeline deployments, enhancing usability for large-scale simulations.
The framework has expanded to include functionalities for social interactions and content generation via LLM integration, enabling NPCs to exhibit more sophisticated, human-like responses in training simulations.
This comprehensive API and modular approach make GHOSTS suitable for creating dynamic, high-fidelity simulation environments for cybersecurity training and testing purposes.

Janus

Developed under the guidance of Dr. Jérôme Dossogne PhD, the Janus project is an adaptive cyber-range designed to provide customized cybersecurity training. Built on an Infrastructure as Code (IaC) and Configuration as Code (CaC) foundation, Janus uses tools like Vagrant and Ansible for automated, reproducible deployment. This cyber-range leverages a vulnerability database and dynamic scenario generator, enabling challenges to be adjusted to each user's skill level.

Janus employs a modular architecture combining virtual machines and Docker containers, ensuring secure environment isolation and flexibility for integrating new vulnerabilities or services as needed. Communication with the simulated agents is facilitated via Mattermost, creating an immersive, interactive experience. The architecture is depicted in the following figure.

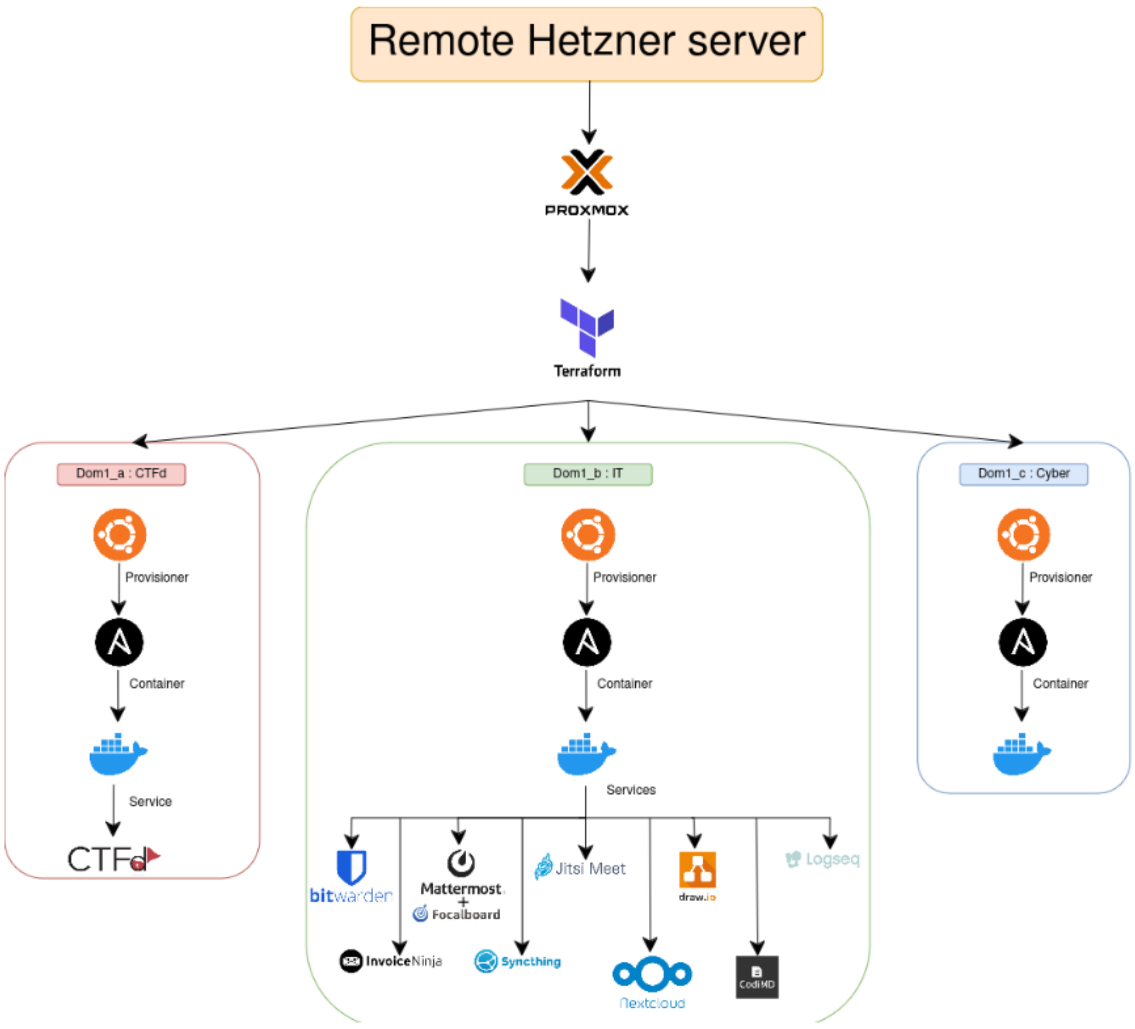


Figure D.3: Janus architecture

Social Engineering

Social engineering merges psychological manipulation with information security, using tactics to obtain unauthorized access by exploiting human behavior rather than technical defenses. Foundational works by Christopher Hadnagy and Robert Cialdini provide critical insights into the psychological aspects that social engineers exploit to influence individuals.

This background section will draw from their research to examine key social engineering principles, techniques, emotional triggers, and countermeasures. [3] [7].

- **Psychological Principles of Influence (Cialdini):** Robert Cialdini's six principles of persuasion , reciprocity, commitment, social proof, authority, liking, and scarcity—form the foundation for many social engineering techniques. These principles exploit ingrained behaviors that prompt individuals to comply with requests without fully analyzing the risks.

  - **Reciprocity:** Leveraging the tendency to return favors, attackers may provide seemingly helpful information or assistance, inducing a sense of obligation to reciprocate, which can lead to sharing sensitive information.
  - **Commitment and Consistency:** Once an individual commits to an action, they are more likely to follow through on related requests. Social engineers use this by starting with minor, low-risk requests that incrementally escalate.
  - **Social Proof and Authority:** Social engineers may impersonate trusted figures or peers to convey that complying with a request aligns with group behavior or is endorsed by authority.
  - **Liking and Scarcity:** Building rapport through flattery or presenting time-sensitive requests creates a sense of urgency and trust, causing individuals to respond without question.

- **Core Social Engineering Techniques (Hadnagy):** Christopher Hadnagy's analysis categorizes social engineering methods into practical tactics commonly seen in real-world attacks.

  - **Pretexting:** Attackers create credible scenarios to gain trust. With comprehensive research, they pose as an authority figure, such as an IT technician, prompting targets to divulge sensitive information.
  - **Phishing and Spear Phishing:** Phishing attacks deliver deceptive emails or messages to trick recipients into sharing credentials or downloading malware. Spear phishing adds a level of personalization, increasing its effectiveness.
  - **Impersonation, Tailgating, and Baiting:** Impersonation involves posing as trusted personnel, while tailgating enables unauthorized access to restricted areas. Baiting employs curiosity-inducing items like USB drives to prompt interaction.
  - **Quid Pro Quo and Dumpster Diving:** Quid pro quo offers a benefit in exchange for information. Dumpster diving involves retrieving discarded physical or digital data, potentially containing confidential information.

- **Emotional Manipulation and Cognitive Biases:** Emotional triggers are critical in bypassing rational decision-making:

  - **Fear, Curiosity, and Greed:** These emotions drive impulsive actions. Social engineers often create artificial crises, like security alerts, or use enticing messages to provoke curiosity or greed.

  - **Authority and Confirmation Biases:** Individuals are likely to comply with perceived authority, and social engineers exploit this by impersonating figures of power. Confirmation bias is used to align with the target's preconceptions, reducing scrutiny.

- **Advanced Social Engineering Strategies and Green Team Simulations:** In advanced scenarios, social engineers deploy multi-stage attacks that combine reconnaissance, pretexting, and phishing. Green Team simulations are used to evaluate an organization's resilience by simulating these realistic social engineering scenarios in a controlled environment, allowing teams to assess employee awareness and response to different techniques.

- **Mitigation Techniques:** Countermeasures for social engineering include:
  - **Awareness Training and Behavioral Cues:** Regular, scenario-based training educates employees on detecting and responding to social engineering attempts, enhancing overall vigilance.
  - **Policy Enforcement and Technical Safeguards:** Policies that enforce strict access controls and technical safeguards like multi-factor authentication (MFA) add layers of defense against unauthorized access.

Social engineering leverages psychological vulnerabilities to bypass technical defenses, making it crucial for organizations to incorporate human-centered security measures. Hadnagy's and Cialdini's work informs these strategies, and implementing Green Team simulations can reveal potential areas for improvement.

D.2.3 Objectives

The goal of this project is to :

- **Develop a Framework for Green Team Emulation:** Design an automated framework that enables the simulation of Green Team behavior within a cyber range using large language models (LLMs).

- **Enhance Threat Detection Capabilities:** Leverage LLMs to improve the detection and understanding of social engineering and other sophisticated attack tactics.

- **Test Defensive Mechanisms in Simulated Environments:** Implement realistic, repeatable scenarios to test and evaluate the effectiveness of cybersecurity defenses against targeted attacks..

D.2.4 Expected outcome

The project will produce a number of deliverables:

- **Functional Emulation Model:** A fully functional Green Team emulation model that mimics real-world team operations within a cyber range.

- **Training and Testing Framework:** A reusable platform for cybersecurity training and testing, specifically in social engineering scenarios.

- **Master Thesis and Publication Draft:** A completed master thesis and an optional draft for a publication, depending on results and future research possibilities.

D.3 Proposal description (max. 4 pages, ref's included)

D.3.1 Aim of the study and relevance for designated target group

The aim of this study is to simulate a "green team" using a large language model (LLM) to aid cybersecurity experts in analyzing and testing the effectiveness of their responses to social engineering threats. By emulating green team behaviors, this research seeks to enhance understanding of threat detection and response strategies within cybersecurity frameworks.

Relevance to Cybersecurity Experts

This study is particularly relevant for cybersecurity professionals due to:

- **Threat Analysis Enhancement**: The simulation provides insights into the behaviors and tactics of a green team, allowing experts to refine their approaches to identifying and countering social engineering attempts.
- **Testing Defensive Mechanisms**: By simulating real-world scenarios, the study enables cybersecurity teams to evaluate the robustness of their defense strategies against social engineering threats.
- **Training and Development Tool**: The research offers a framework for developing training modules that help cybersecurity experts better understand social engineering tactics and improve their response capabilities.

D.3.2 State of the art

Objectives of the Section

- **A** - Review literature on cyber ranges, user behavior emulation, and LLM-powered agents.
- **B** - Provide a detailed exploration of cyber range architecture and the roles of various teams, including the Green Team.
- **D** - Examine current advances in LLMs and their potential for emulating and analyzing user behavior within secure environments.
- **E** - Analyze the intersection of social engineering and LLM-based AI, proposing strategies for enhancing defense mechanisms.

Approach

This state-of-the-art analysis begins by reviewing key concepts relevant to this research, including cyber range architecture, Green Team operations, and user behavior emulation. During this phase, various tools and techniques were evaluated to determine their suitability for simulating realistic social engineering scenarios.
The Janus infrastructure provides a foundation for integrating these concepts within a secure and adaptable environment, enabling simulations that support cybersecurity training. Agile methodology guided the development process, allowing for iterative refinement and alignment with project goals.

A : Cyber ranges, User behavior emulation.

Cyber ranges are advanced virtual environments designed to replicate real-world information technology infrastructures, providing platforms for cybersecurity training, testing, and research. According to the **NIST Cyber Range Guide** [27], a cyber range can serve multiple functions, from educational settings to testing defensive and offensive strategies. Cyber ranges generally support teams such as Red Teams (attackers), Blue Teams (defenders), and Purple Teams (combined attacker-defender). The architecture, tools, and capabilities of cyber ranges have evolved to create more realistic simulations and improve cybersecurity training outcomes.

Cyber Ranges Cyber ranges are utilized in various forms, from educational setups in academic institutions to advanced military simulations. Early implementations were largely static, but modern cyber ranges support dynamic, large-scale environments that emulate complex network configurations and cyber-physical systems. Recent studies emphasize the integration of hybrid cloud architectures in cyber ranges to replicate real-world network conditions, enabling the simulation of sophisticated attack vectors and multi-stage adversarial campaigns . Notable examples include the NATO Cooperative Cyber Defence Centre of Excellence (CCDCOE), which uses advanced cyber ranges to conduct joint exercises involving both civilian and military entities, simulating realistic adversarial conditions.

User Behavior Emulation User behavior emulation is critical for creating authentic cyber range scenarios, especially in the context of insider threats and social engineering attacks. Initial user behavior emulation models focused on simple, automated user actions without complex decision-making logic. In recent years, advancements in machine learning and synthetic data generation have enhanced the realism of user emulation, allowing cyber ranges to simulate diverse user profiles. users.

Despite these advancements, there remains limited research on the emulation of specialized roles beyond administrators and regular users, such as Green Teams, which support infrastructure setup and management within the range. This gap highlights an area where future research could enhance the realism and utility of cyber range exercises, specifically by enabling more complex simulations that involve all operational roles.

LLM-Powered Agents. The integration of large language models (LLMs) within cyber ranges represents a recent innovation, especially for simulating social engineering and natural-language-based attacks. LLMs like GPT-3 are increasingly used to generate sophisticated phishing messages that adapt to responses from Blue Teams, creating an evolving challenge for defenders [2]. These agents enhance the authenticity of cyber exercises by introducing adversaries that use language fluently, mimicking real-world attackers who exploit human vulnerabilities. Studies have demonstrated that LLM-powered agents can improve Blue Team preparedness by exposing them to adaptive, realistic threats that traditional simulation methods fail to capture [22].

B : Detailed Exploration of Cyber Range Architecture and Roles of Various Teams

Cyber range architectures are designed with multiple layers to support the realistic emulation of network environments, organizational roles, and attack-defense scenarios. The **NIST Cyber Range Guide** [27] identifies several core components that are essential for effective cyber range operations, including the orchestration, virtualization, and threat intelligence layers. Below is a detailed analysis of these components and the roles of various teams in cyber range exercises.

Cyber Range Architecture components: The architecture of a cyber range typically consists of the following core components:

- **Orchestration Layer:** This layer is responsible for provisioning, configuring, and managing virtualized resources in the cyber range. Orchestration tools like *Terraform* and *Ansible* are commonly employed to automate the deployment of virtual machines and networks, enabling rapid and flexible setup of training scenarios .
- **Virtualization Layer:** Cyber ranges leverage virtualization technologies such as *Proxmox* and *Docker* to create isolated environments for various team exercises. This layer supports multi-tenant environments, allowing different teams to operate simultaneously without interference .
- **Threat Intelligence Integration:** Incorporating real-time threat intelligence feeds into cyber ranges enables Red and Blue Teams to interact with current, realistic adversarial tactics, techniques, and procedures (TTPs). This integration allows participants to develop skills in detecting and mitigating contemporary cyber threats, improving their readiness for real-world scenarios .
- **Learning Management System (LMS):** The LMS tracks the performance of participants and provides feedback on their response effectiveness. It enables trainers to tailor exercises to specific skill levels, fostering personalized learning paths within the cyber range environment .

Roles of Various Teams. Cyber range exercises involve distinct teams, each fulfilling roles that simulate real-world cybersecurity operations. These roles are essential for creating realistic training scenarios and fostering collaboration between defensive and offensive security teams. [31]

- **Red Team:** Acts as the offensive team, employing tactics like spear-phishing, malware deployment, and network exploitation to compromise Blue Team defenses. Studies show that Red Team exercises expose security weaknesses, enabling organizations to improve their defensive strategies.
- **Blue Team:** Responsible for defending the simulated environment from Red Team attacks. Blue Teams use tools such as Security Information and Event Management (SIEM) systems to detect, analyze, and respond to threats. Realistic Blue Team training is enhanced by dynamic threat simulations and user behavior emulation, as well as regular feedback from the LMS.

- **Green Team:** In a cyber range context, Green Teams provide operational support, ensuring that the virtual environment is configured and maintained correctly. However, literature addressing the specific role of Green Teams within cyber range exercises is sparse. This indicates a potential research gap in understanding how Green Teams could contribute to more complex, realistic cyber simulations.
- **White Team:** Oversees and evaluates the exercise, ensuring that all protocols are followed and collecting data for post-exercise analysis. The White Team's assessments are critical for refining training programs and addressing any identified deficiencies in team responses.

C : Current advances in LLMS

Large Language Models (LLMs) have achieved remarkable advancements in **scalability**, **emergent abilities**, and **fine-tuning techniques**, offering substantial potential for simulating a green team in cybersecurity. Their ability to adapt dynamically, simulate human interactions, and analyze complex scenarios aligns well with the operational needs of a green team tasked with defending against social engineering attacks and other cyber threats.

In-Context Learning for Real-Time Adaptability One emergent feature in models like GPT-3 and GPT-4 is **in-context learning**, which enables real-time adaptability. This capability allows LLMs to respond to new scenarios based on minimal instruction, facilitating dynamic responses without requiring task-specific retraining [32]. Such adaptability is critical in green team simulations, where models must respond proactively to evolving social engineering tactics and interpret input context to generalize effectively across varied tasks.

Ethical and Safe Responses Through RLHF To ensure **ethical and safe outputs**, reinforcement learning from human feedback (RLHF) is applied to fine-tune LLMs, aligning their responses with human values and preventing harmful outputs [20, 21]. This aspect is particularly relevant in green team simulations where ethical considerations are essential. RLHF enhances the model's ability to differentiate between benign and malicious behaviors, crucial for detecting manipulative tactics and avoiding inadvertent validation of deceptive prompts.

Chain-of-Thought Reasoning for Multi-Step Analysis LLMs' reasoning capabilities are further improved by **Chain-of-Thought (CoT) prompting**, which supports multi-step analysis of sequential data [30] This structured reasoning is highly applicable to green team operations, where it enables models to dissect complex, phased attacks like social engineering. CoT allows the model to break down interactions into stages, systematically identifying signs of escalation that can be indicative of threats.

Multimodal Input Integration for Comprehensive Monitoring Recent multimodal advancements in LLMs allow them to process **text, image, and audio inputs** simultaneously, supporting a holistic approach to threat detection [16]. In green team simulations, this capability enables LLMs to monitor diverse data channels and detect social engineering tactics across different communication formats, which is vital for comprehensive situational awareness.

Tool Manipulation for Dynamic Data Retrieval Finally, LLMs can now leverage **external tools and APIs** such as search engines or databases, enhancing their adaptability through real-time data retrieval and verification [18]. This feature enables green team simulations to simulate real-world defensive processes, such as fact-checking or identity verification, that a human defender might conduct during a cyber incident.

These advancements underscore the effectiveness of LLMs in cybersecurity applications, particularly for simulating green team responses that require intelligent, ethical, and adaptable handling of human-centered threats.

D : Social Engineering

Understanding social engineering is crucial in designing AI-based cybersecurity defenses that can preemptively recognize and respond to human-centered attacks. Social engineering tactics exploit psychological principles to deceive individuals into breaching security protocols, and thus, analyzing these tactics is directly relevant for our research. The following works offer a comprehensive foundation to help build models that detect these techniques in digital interactions.

- **Historical Foundations and Practical Techniques**: Kevin Mitnick's *The Art of Deception* explores various methods of manipulation, such as pretexting and impersonation, to exploit human vulnerabilities. This work provides numerous real-world scenarios demonstrating how attackers bypass technical defenses by leveraging trust and authority. In our research, these scenarios serve as valuable case studies for training AI systems to recognize manipulation cues based on behavior and context. [19]

- **Psychological Mechanisms and Influence Tactics**: Robert Cialdini's *Influence: The Psychology of Persuasion* outlines six key principles—reciprocity, commitment, social proof, authority, liking, and scarcity—that underpin many social engineering attacks. By understanding these principles, our research can focus on identifying specific language and behavioral patterns that suggest attempts at psychological influence. For instance, an AI system can be designed to detect urgency or authority-triggered compliance within user interactions. [3]

- **Application of Social Engineering in Ethical Contexts**: Christopher Hadnagy's *Human Hacking* addresses social engineering from both offensive and defensive perspectives, highlighting the importance of ethical social engineering for testing systems. This approach is beneficial for our research, as it enables the development of AI systems that can distinguish between benign interactions and genuine threats by assessing contextual factors. The ethical aspect helps balance sensitivity and accuracy, allowing for effective filtering of malicious attempts. [7]

- **Adapting Tactics to Individual Vulnerabilities**: The study *Social Engineering in the Context of Cialdini's Psychology of Persuasion and Personality Traits* links social engineering effectiveness to individual personality traits, emphasizing that susceptibility varies widely across different types of people. This research can be useful for our project, as it underscores the need for adaptive AI models that can tailor their defensive responses based on inferred user profiles or behavioral patterns. [24]

- **Recent Trends in Social Engineering**: The review *A Multivocal Literature Review on Growing Social Engineering Based Cyber-Attacks/Threats During the COVID-19 Pandemic* discusses how attackers rapidly adjust tactics in response to global events, demonstrating the dynamic and evolving nature of social engineering threats. This insight informs our research by highlighting the importance of AI-driven solutions that can quickly learn and respond to new tactics, allowing for continual adaptation to emerging threats. [12]

- **Example of AI Designed to Resist Social Manipulation:** The online game GANDALF presents an AI tasked with protecting a password against players' social engineering attempts. This game effectively illustrates how AI can detect manipulation cues by analyzing contextual and behavioral indicators in human interactions. Although our research focuses on fully automated systems, GANDALF remains relevant as it demonstrates how the green team might respond to social engineering attempts (though not prompt injections as seen in GANDALF).

These works collectively provide a strong theoretical and practical basis for understanding and countering social engineering. By integrating insights from psychology, ethical hacking, and adaptable AI, our research aims to develop context-sensitive AI that can react to different types of social engineering attempts as seen in the example of GANDALF.

D.3.3 Research Strategy

The research strategy for this project involved a systematic review of literature across multiple domains relevant to cybersecurity simulation, social engineering, and adaptive AI. Key areas of research included cyber range infrastructure, virtualization, containerization, and the application of Large Language Models (LLMs) to simulate human behavior. The following approaches were adopted:

- **Literature Review:** The literature review process involved sourcing foundational and recent works from academic databases and online research tools, with a primary reliance on Google Scholar for finding well-cited papers and publications. Access to IEEE sources was limited, so alternative sources were prioritized. The search focused on cyber range architectures, transformer models for LLMs, and best practices in infrastructure and configuration management.
- **Psychology and Social Engineering Resources:** Relevant studies in social engineering and psychological manipulation were identified, in part, through coursework in Human Factors, which provided a strong foundation in these areas. Additional resources were found using targeted searches for psychological principles in social engineering, ensuring these findings were integrated into the simulation of realistic, adaptive AI responses.
- **Evaluation of Tools and Techniques:** The project required an in-depth assessment of automation and virtualization tools, specifically Vagrant, Docker, and Ansible, to ensure they met the requirements for managing virtualized environments in a cyber range. Technical manuals and open-access guides offered insights into each tool's capabilities and limitations, guiding decisions on tool selection and best practices for infrastructure deployment.
- **Communication Strategy Development (In Progress):** A structured communication strategy is in development to facilitate continuous updates and coordination with stakeholders. This strategy will support the effective dissemination of project findings and outcomes, promoting alignment with project objectives.

D.3.4 Collaboration

Still need to know who I must add here.

D.3.5 Expected outcome

The project will produce a number of deliverables:

- **Functional Emulation Model:** A fully functional Green Team emulation model that mimics real-world team operations within a cyber range.
- **Training and Testing Framework:** A reusable platform for cybersecurity training and testing, specifically in social engineering scenarios.
- **Master Thesis and Publication Draft:** A completed master thesis and an optional draft for a publication, depending on results and future research possibilities.

D.3.6 Feasibility & risks

Strengths

- **Existing Mattermost Integration:** Previous successful integration of Mattermost provides a solid foundation for real-time communication within the cyber range.
- **Robust Tooling Setup:** The project leverages established tools like Vagrant, Docker, and Ansible, ensuring a scalable and reproducible infrastructure.
- **Expert Supervision and Guidance:** Access to experienced supervisors and research facilities provides strong support for overcoming technical challenges.

Weaknesses

- **LLM Adaptation Complexity:** Customizing LLMs to add memory, consistent personalities, and adaptability is complex and may require significant fine-tuning.
- **Infrastructure Dependency and Compatibility Issues:** Dependency on multiple tools (Vagrant, Docker, etc.) could lead to compatibility challenges, especially when automating across local and remote environments.
- **Resource Intensity:** High computational requirements for real-time LLM processing and continuous integration pipelines may strain available resources.

Opportunities

- **Emerging Research Area:** Green Team simulation using LLMs in a cyber range is a novel research area, with the potential to contribute valuable insights to cybersecurity training and defense strategies.
- **Enhanced Realism in Training:** Adaptive Green Team personalities could significantly improve training realism, providing valuable experience in handling complex social engineering threats.
- **Potential for Publication and Future Collaboration:** Positive results could lead to publications, opening opportunities for further research and collaboration with industry or academia.

Threats

- **Rapid Changes in AI Technology:** Fast-paced advancements in LLMs and AI may require ongoing adjustments to keep the project's LLM components up-to-date.
- **Project Timeline Constraints:** The complexity of LLM adaptation and multi-layered infrastructure setup may lead to delays, affecting project deliverables and deadlines.

Bibliography

[1] Abramson, D.: Virtualization in enterprise. Intel Technology Journal 10(3), 1–8 (2006) [Cited on page 29.]

[2] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners (2020), `https://arxiv.org/abs/2005.14165` [Cited on page 42.]

[3] Cialdini, R.B.: Influence: The Psychology of Persuasion. Harper Business (2006) [Cited on pages 39 and 45.]

[4] Dakić, V., Kovač, M., Slovinac, J.: Evolving high-performance computing data centers with kubernetes, performance analysis, and dynamic workload placement based on machine learning scheduling. Electronics 13(13) (2024) [Cited on page 29.]

[5] Gajda, W.: Pro Vagrant. Apress (2015) [Cited on page 34.]

[6] Geerling, J.: Ansible for DevOps: Server and configuration management for humans. Midwestern Mac, LLC, second edition edn. (2020) [Cited on pages VI and 32.]

[7] Hadnagy, C.: Social Engineering: The Science of Human Hacking. John Wiley & Sons (2018) [Cited on pages 39 and 45.]

[8] HashiCorp: https://developer.hashicorp.com/vagrant/docs/multi-machine, `https://loopbin.dev/tutos/vagrant-06-provider-fournisseur/` [Cited on page 34.]

[9] HashiCorp: Vagrant documentation - providers, `https://developer.hashicorp.com/vagrant/docs/providers` [Cited on page 34.]

[10] HashiCorp: Vagrant documentation - provisioning, `https://developer.hashicorp.com/vagrant/docs/provisioning` [Cited on page 34.]

[11] HashiCorp: Vagrant documentation - vagrantfile, `https://developer.hashicorp.com/vagrant/docs/vagrantfile` [Cited on page 34.]

[12] Hijji, M., Alam, G.: A multivocal literature review on growing social engineering based cyber-attacks/threats during the covid-19 pandemic: Challenges and prospective solutions. IEEE Access (January 2021) [Cited on page 46.]

[13] Institute, C.M.U.S.E.: Ghosts api - core documentation (2024), `https://cmu-sei.github.io/GHOSTS/core/api/` [Cited on page 36.]

[14] Institute, C.M.U.S.E.: Ghosts npc framework (2024), `https://github.com/cmu-sei/GHOSTS` [Cited on page 36.]

[15] Lee, M.: A mathematical interpretation of autoregressive generative pre-trained transformer and self-supervised learning. Mathematics 11(11) (2023) [Cited on page 35.]

[16] Markov, T., Zhang, C., Agarwal, S., Eloundou, T., Lee, T., Adler, S., Jiang, A., Weng, L.: A holistic approach to undesired content detection in the real world (2023), `https://arxiv.org/abs/2208.03274` [Cited on page 44.]

[17] Merkel, D.: Docker: Lightweight linux containers for consistent development and deployment. In: Linux journal. vol. 2014, p. 2 (2014) [Cited on page 30.]

[18] Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., Grave, E., LeCun, Y., Scialom, T.: Augmented language models: a survey (2023), `https://arxiv.org/abs/2302.07842` [Cited on page 45.]

[19] Mitnick, K.D., Simon, W.L.: The Art of Deception: Controlling the Human Element of Security. John Wiley & Sons Inc (October 2002) [Cited on page 45.]

[20] OpenAI, ...: Gpt-4 technical report (2024), `https://arxiv.org/abs/2303.08774` [Cited on page 44.]

[21] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., Lowe, R.: Training language models to follow instructions with human feedback (2022), `https://arxiv.org/abs/2203.02155` [Cited on page 44.]

[22] Pastor-Galindo, J., Nespoli, P., Ruipérez-Valiente, J.A.: Generative agent-based social networks for disinformation: Research opportunities and open challenges. Theme Article: Synthetic Realities and Artificial Intelligence-Generated Contents (2023), available from University of Murcia [Cited on page 42.]

[23] Poulton, N.: Docker Deep Dive: Zero to Docker Edition 2023. Leanpub (2023) [Cited on pages VI and 31.]

[24] Quiel, S.: Social Engineering in the Context of Cialdini's Psychology of Persuasion and Personality Traits. Ph.D. thesis, Hamburg University of Technology (July 2013), bachelor Thesis [Cited on page 45.]

[25] Raj, H., Chowdhury, M., De Vleeschauwer, B., Landherr, A., Tutschku, K., Wu, J.: Distributed virtual network embedding: Framework, solutions, and future directions. Journal of Computer Communications 33(11), 1395–1407 (2010) [Cited on page 30.]

[26] Smith, J.N., Nair, R.: Comparative performance evaluation of vm-based virtual servers. In: Proceedings of the 2005 USENIX Annual Technical Conference. pp. 263–276 (2005) [Cited on page 30.]

[27] of Standards, N.I., Technology: The cyber range: A guide (2023), available at the National Institute of Standards and Technology website [Cited on pages 42 and 43.]

[28] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017) [Cited on pages 34 and 35.]

[29] Waldspurger, C.A.: Memory resource management in vmware esx server. In: Proceedings of the 5th symposium on Operating systems design and implementation. pp. 181–194. ACM (2002) [Cited on page 30.]

[30] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., Zhou, D.: Chain-of-thought prompting elicits reasoning in large language models (2023), `https://arxiv.org/abs/2201.11903` [Cited on page 44.]

[31] Yamin, M.M., Katt, B., Gkioulos, V.: Cyber ranges and security testbeds: Scenarios, functions, tools and architecture. Computers Security 88, 101636 (2020) [Cited on page 43.]

[32] Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.Y., Wen, J.R.: A survey of large language models (2024), `https://arxiv.org/abs/2303.18223` [Cited on page 44.]

D.4 Phasing of the project (max. 4 pages)

Start date:

End date:

D.4.1 Workpackage 1: Environment Setup and Infrastructure (Step 1)

Start date:

End date: **Objective:** Establish the foundational target environment for simulation, incorporating a Debian-based virtual machine with Docker to host Mattermost and other essential services.

- **Tasks:**
  - Set up Vagrant to automate virtual machine deployment.
  - Configure Docker to run essential containers, including Mattermost.
  - Implement Ansible playbooks for initial setup, ensuring reproducibility and security compliance.
  - Begin setting up a continuous integration (CI) pipeline for automated deployment and updates.
- **Deliverables:** A functional, isolated virtual environment with CI pipelines established for automated deployment.

D.4.2 Workpackage 2: Green Team Simulation (Step 2)

Start date:

End date: **Objective:** Develop the Green Team emulation within the cyber range, focusing on creating a realistic "team" environment.

- **Tasks:**
  - Deploy virtual machines (VMs) with containers running scripts to simulate Green Team tasks.

  - Set up the GHOSTS framework for automated scenario management, if not already integrated.
  - Enable LLM functionality to initiate basic responses in Mattermost interactions.

- **Deliverables:** A preliminary Green Team simulation capable of basic automated responses and interactions within Mattermost.

D.4.3 Enhanced LLM Integration and Personality Development (Step 3)

Start date:

End date: **Objective:** Increase the realism of the Green Team by embedding LLM-based personas with awareness and consistent interactions.

- **Tasks:**
  - Refine LLM integration to allow personality development for each Green Team member.
  - Implement features that enable Green Team agents to recognize each other and simulate memory of past interactions.
  - Modify the cyber range to simulate vulnerabilities (e.g., password exposure) to assess the agents' responses.
- **Deliverables:** A dynamic Green Team simulation with individualized, adaptive LLM personas interacting meaningfully within the cyber range.

D.4.4 Workpackage 4: Documentation and Finalization (Step 4)

Start date:

End date: **Objective:** Finalize research findings, compile data, and complete the project report and thesis.

- **Tasks:**
  - Collect and analyze all relevant data gathered from simulation phases.
  - Compile findings and evaluate the impact of adaptive Green Team simulations on cybersecurity training and defense strategies.
  - Finalize writing of the thesis, including relevant diagrams, evaluations, and references.
- **Deliverables:** A comprehensive master thesis, completed project report, and a draft for potential future publication.

D.5 Expertise of the project's research team (max. 2 pages)

LATER:

D.5.1 Expertise

D.5.2 Publications & porfolio relevant to the project proposal

D.6 Requirement (equipment, skills, ...)

D.6.1 Equipment

To carry out this project, the team will rely on specific infrastructures that ensure optimal performance. The **Janus** infrastructure will be utilized for the **cyber range**, providing a secure and controlled environment for testing and simulations. Additionally, a **Hetzner** server with **64 GB of RAM** is available for use, enabling the management of resource-intensive applications and services. This hardware serves as a solid foundation for the experiments and development of the solutions proposed within the project.

D.6.2 Skills

The project will require a range of technical skills that are essential for its successful execution. These include, but are not limited to:

- **Vagrant**: for managing and configuring reproducible development environments.
- **Docker**: for containerizing applications, facilitating their deployment and scalability.
- **Ansible**: for automating configuration and system management tasks.
- **Molecule**: for developing and testing Ansible roles, ensuring that configurations are correct and functional.

In addition to these skills, the team will need to acquire knowledge through the reading of technical documentation, specialized books, and hands-on experimentation. This ongoing learning process will enable the team to stay updated on the latest technologies and industry practices, ensuring they are well-equipped to tackle the challenges presented by the project.

D.7 Evaluation & Self-Evaluation

D.7.1 Self-Evaluation

- 8 /10: Relevance and scope : Is the relevance of the proposed research well defined? Is the scope of project well described and delimited?
- 8 /10: Efficiency : Are the required/allocated means means, i.e. personnel, infrastructure and equipment, consistent with the projected outcome of the project?
- 7 /10: State of the Art : How is the state of the art, related to this proposal at a national and international level, described? Are the proposers aware of past and current similar research activities?
- 8 /10: Research Strategy and Phasing : Are the phases of the project realistic, coherent and in line with the intended objectives? How realistic and well argumented are the objectives? Is the research team well organized and able to perform the planned research
- Remarks:

D.7.2 Evaluation

- ... /10: Relevance and scope
- ... /10: Efficiency
- ... /10: State of the Art
- ... /10: Research Strategy and Phasing
- Remarks:

Appendix E

Project management records / artifacts

E.1 Gantt charts

E.2 Pomodoro charts

E.3 Agile Methodology's User stories

E.4 Workpackages, Objectives and tasks

E.5 Diagrams & user produced artifacts: use cases, user stories, UML, ...