Automation of Cyber Exploitation through AI

Midshipman 2/C Matthew Frederick Zehnder
Prof. Dane Brown, Cyber Sciences Department

Abstract

With the rapid evolutions of Artificial Intelligence it is important to study the dangers they can bring. In this study, Hack the Box(HTB) capture the flags(CTFs) were utilized to represent cyber attacks that were used to put locally run AI agents to the test for their ability to complete end to end attacks. This involved giving the AI only the IP of the target to decipher if they were able to fully reason about how to proceed. The purpose of this was to test AI's ability to reason through simulated Cyber attacks to see what they would be able to work out on their own. The results showed a keen ability for the AI to solve the beginning stages of the CTFs, but an inability to go much further past the beginning stages. While unable to complete end to end cyber attacks, their usability in automating the beginning stages is promising.

Methods

Code components

- Two AI models, QwQ:32b**[1]** & llama3.1:8b**[2] (Fig 1)**, were pulled from Ollama, a tool to get and run LLM on a personal compuer, to be run in the code.
- Message Stream that is fed into the AIs at every call, querying the model and holding the memory for each instance.
- To direct the response, **tool_calls** were used **(Fig 2)**. Allowing for responses to be tailored.
- **Tmux** allowed for commands to be run on a separate terminal. This allowed user for user interaction when needed and IO blocking.

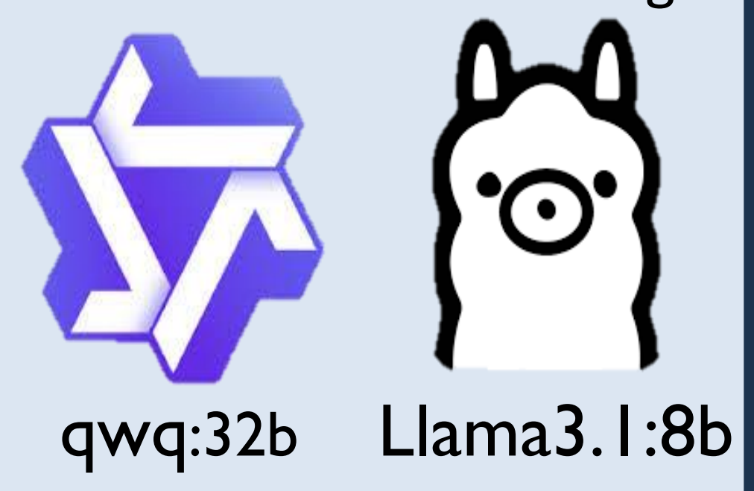

Figure 1. Left: Qwen's AI QwQ. Right: Meta's Llama3.1.

```
#tool call basic format found on https://ollama.com/docs/api
  get_code = [{
              'type': "function",
              'function': {
                 'name': 'extract_code',
                 'description': 'provides the next command to run',
                 'parameters' : {
                    'type': 'object',
                    'properties': {
                       "code_type" : {
                          "type": "string",
                          "description": "The type of code that is getting extract"
                       },
                       "code" :{
                          "type": "string",
                          "description": "The code that is getting extract"
                       },
                       "why" : {
                          "type": "string",
                          "description": "The reason for the choosing that code"
                       }
                    },
                    'required': ['code_type', 'code']
                 }
              }
           }]
```

Figure 2. get_code the tool_call function used in the AI's chat structure. Defines the three responses desired from the AI.

Running the Code

The principle of the code is a self feeding loop that automatically sends commands from the AI to the command line, which generates an output that go back to the AI **(Fig 3)**. The code was then run against 14 different CTF's from HTB. Most of the CTF's were under the easy or very easy category building the stepping stones for what the AI can do. Data from each instance were then compiled in the csv file for further analysis.

Analysis

The csv files contain the command number, the executable command, why it chose the command, how long it took to reason, output of the command, and how long the command took to run. These were used to judge how far in the AI made it into the CTF as well as used to create various charts to show the AI's logical reasoning.

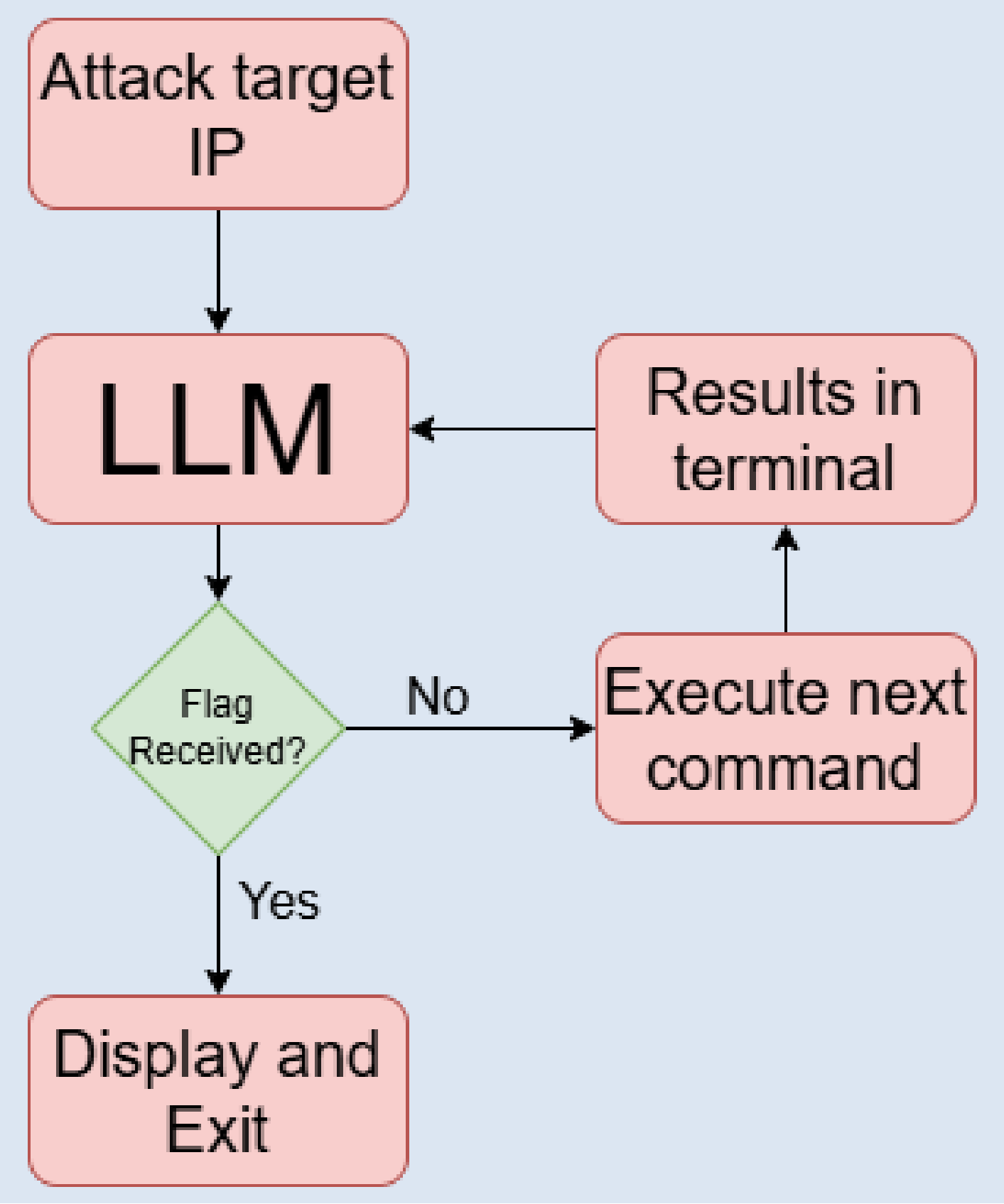

Figure 3. Flowchart of program's logical execution.

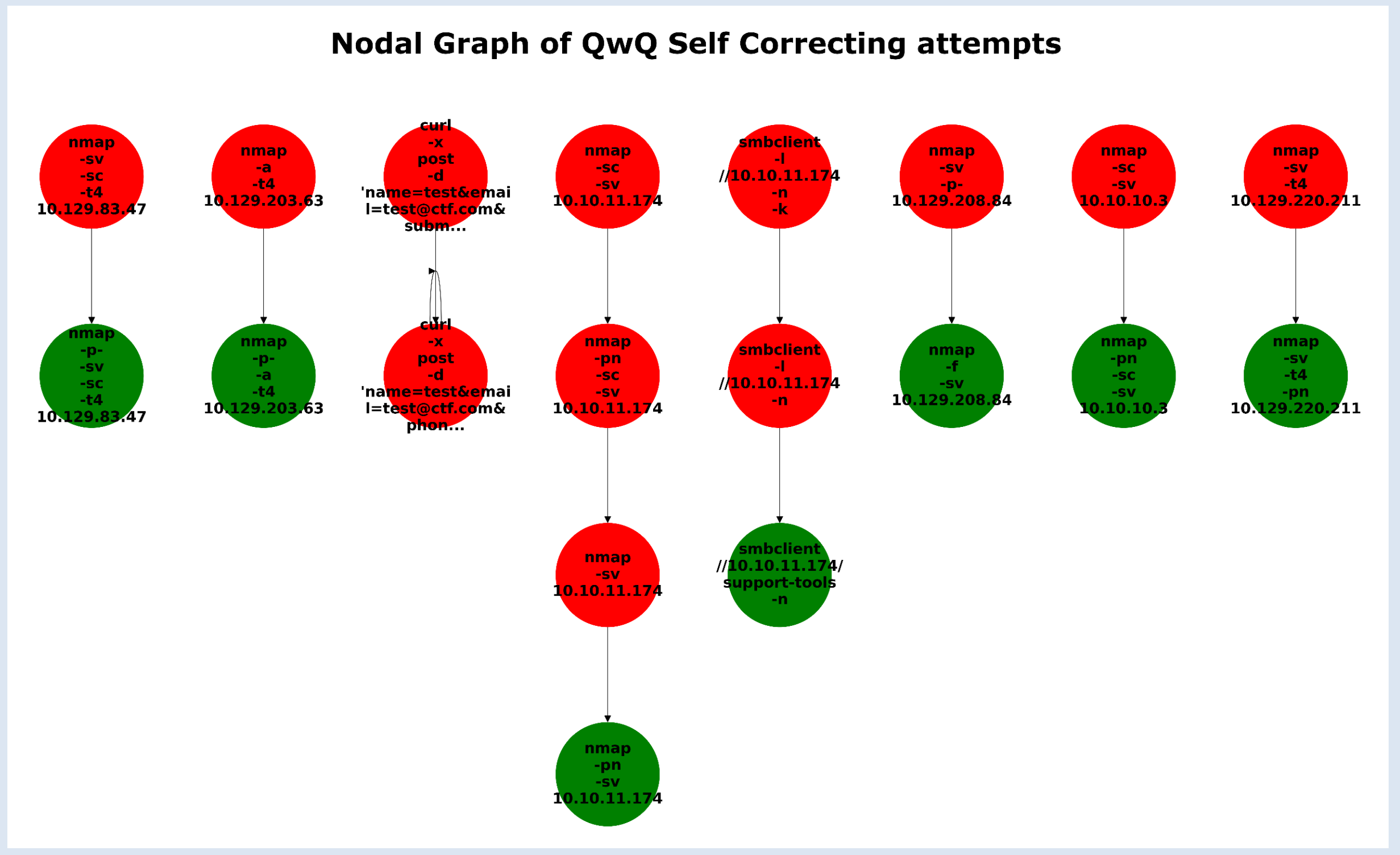

Figure 4. A nodal graph showing QwQ attempts at self correcting. This was captured by finding similar succeeding commands and seeing if they moved on successfully.

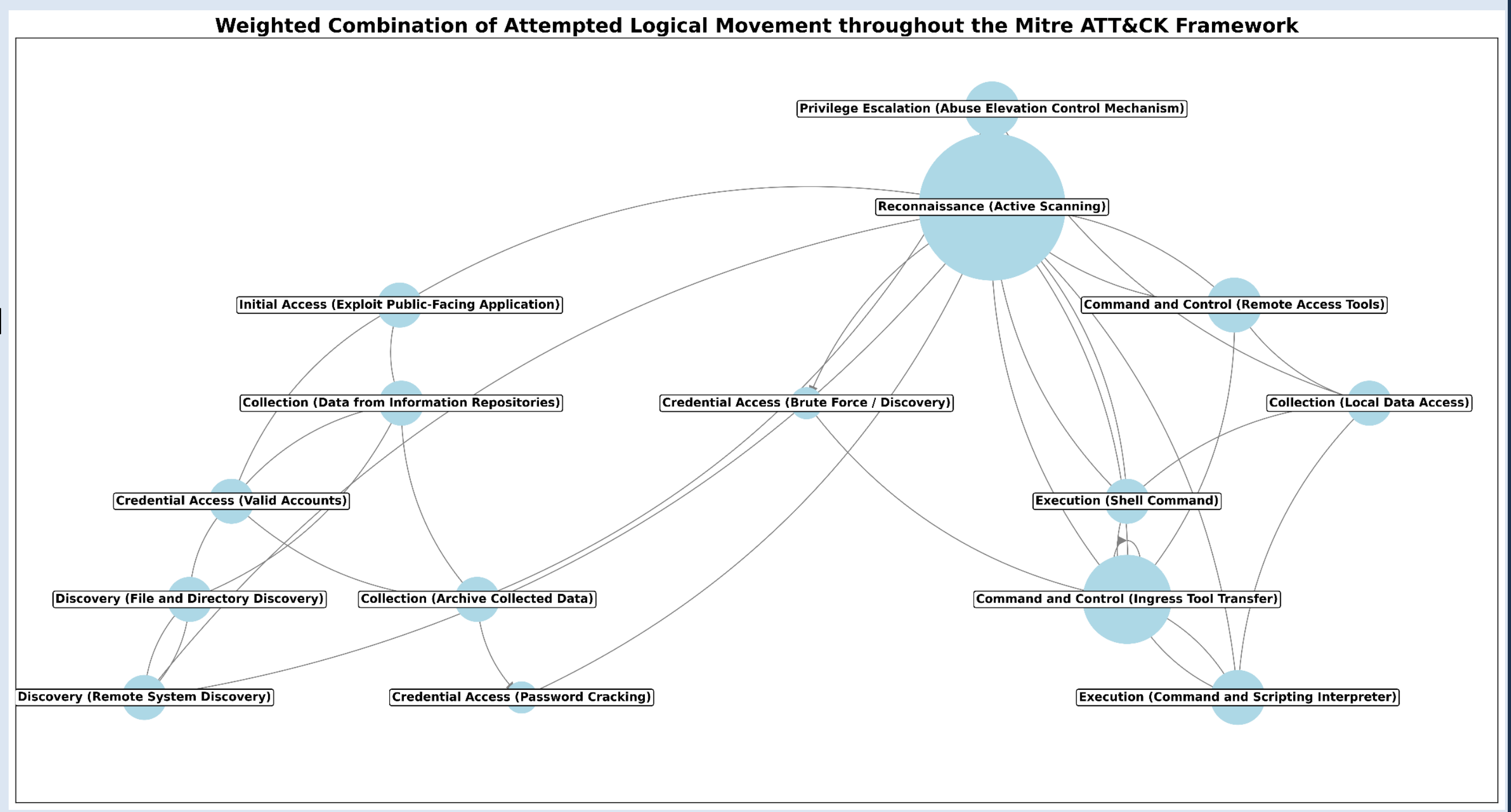
Weighted Combination of Attempted Logical Movement throughout the Mitre ATT&CK Framework
Privilege Escalation (Abuse Elevation Control Mechanism)
Reconnaissance (Active Scanning)
Initial Access (Exploit Public-Facing Application)
Command and Control (Remote Access Tools)
Collection (Data from Information Repositories)
Credential Access (Brute Force / Discovery)
Collection (Local Data Access)
Credential Access (Valid Accounts)
Execution (Shell Command)
Discovery (File and Directory Discovery)
Collection (Archive Collected Data)
Command and Control (Ingress Tool Transfer)
Discovery (Remote System Discovery)
Credential Access (Password Cracking)
Execution (Command and Scripting Interpreter)

Figure 5. A weighted nodal map of attempted movement through the MITRE ATT&CK[3] framework based on commands associated with each stage.

Results

The AI was able to quickly and easily complete the beginning steps of solving the CTF's, consistently able to find the path to the vulnerability. As well as knowing the initial commands to start the exploitation sequence. Even though the AI knew what tools to use, it was unable to effectively use them, especially with more niche clients such as **redis-cli**. Once in these clients QwQ would often either give up not responding with a tool call, killing the program, or it would provide non-working commands repeatedly.

QwQ showed an ability to self-correct its commands**(Fig 4)**. Oftentimes, the model would get the original command wrong, forgetting an argument or similar mistake, and then would correct itself based on what the response was from the command line. This goes to show the model's ability to reason with itself, adding a persistent factor to an exploitation program that is not available without AI.

Discussion

Even though QwQ did not fully complete sophisticated cyber attack techniques in any CTF attempt, it showed a diverse intent to traverse the MITRE ATT&CK Framework**(Fig 5).** These attempts show the AI is not limited by one field of thought when approaching a CTF. But, these results were also heavily influenced by the CTFs chosen and may see different results if applied in the real world.

Due to the AI's proficiency in the initial steps of solving the CTFs, even now AIs could be used in mass attacks as an initial contact strategy in the sense of being able to deliver a possibly vulnerable IP to the adversary, if the AI given a bank of target IP addresses. While not being a sophisticated cyber cracker the AI has shown itself as a reasonable cyber exploit sniffer, with the ability to be run 24/7 could lay as a threat. This has been seen partly before when pen testing with AI **[4]**. Which showed GTP3.5's ability of privilege escalation.

A major expected issue was the AI refusing to produce commands due to their inherent malicious potential of the commands needed. With the only gearing of saying the mission was to solve a CTF, the AI never second-guessed intent, producing malicious code in commands such as **gobuster** or **zipcracker** without complaint.

Future Directions

- With AIs evolving at the rapid rate they currently are, this testing can be perpetual as new, more powerful AI comes to light.
- Testing the limits using an embedded database such as **chromadb,** will allow the AI model to reference a database full of applicable commands in tandem with the AI's reasoning. This dynamic system would allow for quick and easy testing of different data methods.
- Training an AI model on various CTF materials. Using objects such as CTF walkthroughs, **metasploit** tutorial, and CTF cheat sheets can allow for the Ai itself to be more in tune with cyber threat capabilities.

References

[1] Qwen Team, "QwQ: A lightweight and efficient LLM-based AI agent," Alibaba Group, 2024. [Online]. Available: https://huggingface.co/Qwen

[2] Meta AI, "LLaMA 3.1 Language Model," Meta Platforms, Apr. 2025. [Online]. Available: https://ai.meta.com/llama

[3] MITRE Corporation, "MITRE ATT&CK Framework," MITRE, 2024. [Online]. Available: https://attack.mitre.org

[4] A. Happe and J. Cito, "Getting pwn'd by AI: Penetration testing with large language models," *ESEC/FSE 2023*, San Francisco, USA, 2023.

Team Members

https://github.com/zehngator/AACT