**Report- Adaptive Intrusion Detection System (IDS) using RL and Gpt-2**

**Problem and Impact:**

Due to the increasing sophistication of cybersecurity threats, such as zero-day attacks and advanced persistent threats (APTs), has made usual intrusion detection systems (IDS) inadequate. Existing IDS often rely on rule-based systems that struggle with new, unknown threats. Hence, we have the need for an intelligent system that can learn from its environment and adapt to new attack patterns. As the attack patterns get complicated so does the need for an adaptive system. The core problem addressed is that growing challenge of cyber-attacks, specifically Distributed denial of service (DDoS) and port scan attacks, targeting network systems.

**Market for RL in cybersecurity:**

The global Cybersecurity Market is projected to increase from $219 billion in the year 2023 to $345.5 billion by the year 2027, due to the increased digitalization and sophistication of cybersecurity attacks. Real time defence systems will be in high demand as soon as a solution for the high computational requirement for RL is fulfilled, especially in industries like finance, healthcare, and infrastructure, where one data breach could be catastrophic.

Competition- Companies like Palo Alto Network, CrowdStrike, and Cisco have integrated machine learning (ML) and artificial intelligence (AI) into cybersecurity. However, the integration of Reinforcement Learning (RL) and Large Language Models (LLMs) for dynamic, real-time threat detection and response is still in its infancy, offering a unique competitive edge.

**Proposed Solution:**

The solution to this involves building a CyberDefense agent that uses RL to make decisions such as detecting attacks, blocking them, and investigating logs. The RL agent is paired with a GPT-2 based LLM that gives Realtime feedback on the system’s actions. This LLM refines reward signals, which improves the agent’s decision-making process. This integration allows the system to learn from a relatively small set of examples, to quickly adapt to evolving attack patterns.

**Technical Approach:**

* **RL Agent**: The agent will detect attacks (e.g., DDoS, port scanning) and respond by taking actions like blocking or investigating. The RL agent learns optimal actions over time by interacting with the environment.
* **LLM Feedback**: An LLM (GPT-2) processes attack logs, providing feedback that helps refine the agent's behaviour and reward signals. This allows for the dynamic adjustment of the system’s defence strategies.
* **Reward Shaping**: LLM-generated feedback will modify the agent’s reward function to align with desired behaviour, like minimizing system downtime or neutralizing threats quickly.

This project uses Deep Q-networks (DQN) for the training of the RL agent. This agent interacts with a simulated cybersecurity environment to detect, block, or investigate cyberattacks. The dataset used is the CIC-IDS-2107 which gives the agent Realtime attack logs. The LLM processes real time attack logs and helps shape the reward function based on the severity of the threat and system response. The final goal is to create an intelligent, adaptive system capable of autonomously defending against novel cyber threats.

**Description of the Working Prototype**

**Introduction**:  
This prototype here demonstrates the integration of Reinforcement Learning (RL) and Large Language Models (LLMs) to build a prototype for an adaptive cybersecurity system. The RL agent learns to detect and block cyber-attacks, it is trained with 500 timesteps, simpler model architecture, and high exploration rates to allow faster testing. While the LLM gives feedback to the agent to correct its behaviour. The solution uses DQN (Deep Q-Network) for the agent and GPT-2 for log processing and reward shaping, providing a real-time, evolving defence system.

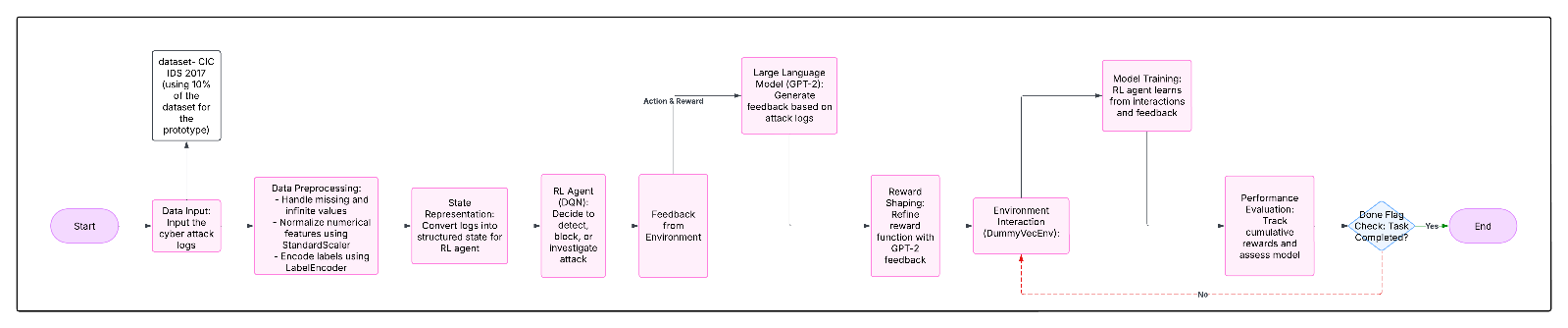


Diagram describing the flow of the project

**1. Technical Design Overview:**

The system consists of two key components:

1. **RL Agent (DQN):** The RL agent learns optimal defence strategies by interacting with the environment and adjusting its actions based on rewards.
2. **LLM Feedback (GPT-2):** The LLM processes textual attack logs and provides feedback to adjust the agent’s reward structure, enabling the agent to adapt to new attack types.

**Reinforcement Learning (RL) Agent:**

* **Action Space**: The agent takes one of three actions based on the environment’s state:
  + 0: Detect an attack.
  + 1: Block the attack.
  + 2: Investigate the attack logs.
* **Observation Space**: The agent’s state is represented as a vector of normalized numerical features extracted from the attack logs (e.g., packet size, source IP, attack type).
* **Reward Function**: The reward is shaped by using feedback from GPT-2. For instance, blocking an attack will give out a positive reward, while failing to respond will result in a negative reward. The feedback from the LLM makes sure that rewards reflect real-time attack severity.

**Large Language Model (LLM):**

* **LLM for Reward Shaping**: GPT-2 generates real-time feedback based on attack logs, which impacts the reward function. This adaptive feedback helps the agent to adjust its behaviour for more effective defence strategies.
* **LLM for State Representation**: GPT-2 processes attack logs to provide a structured presentation of the attack, helping the RL agent understand the environment more accurately.

**2. Gym Environment Setup:**

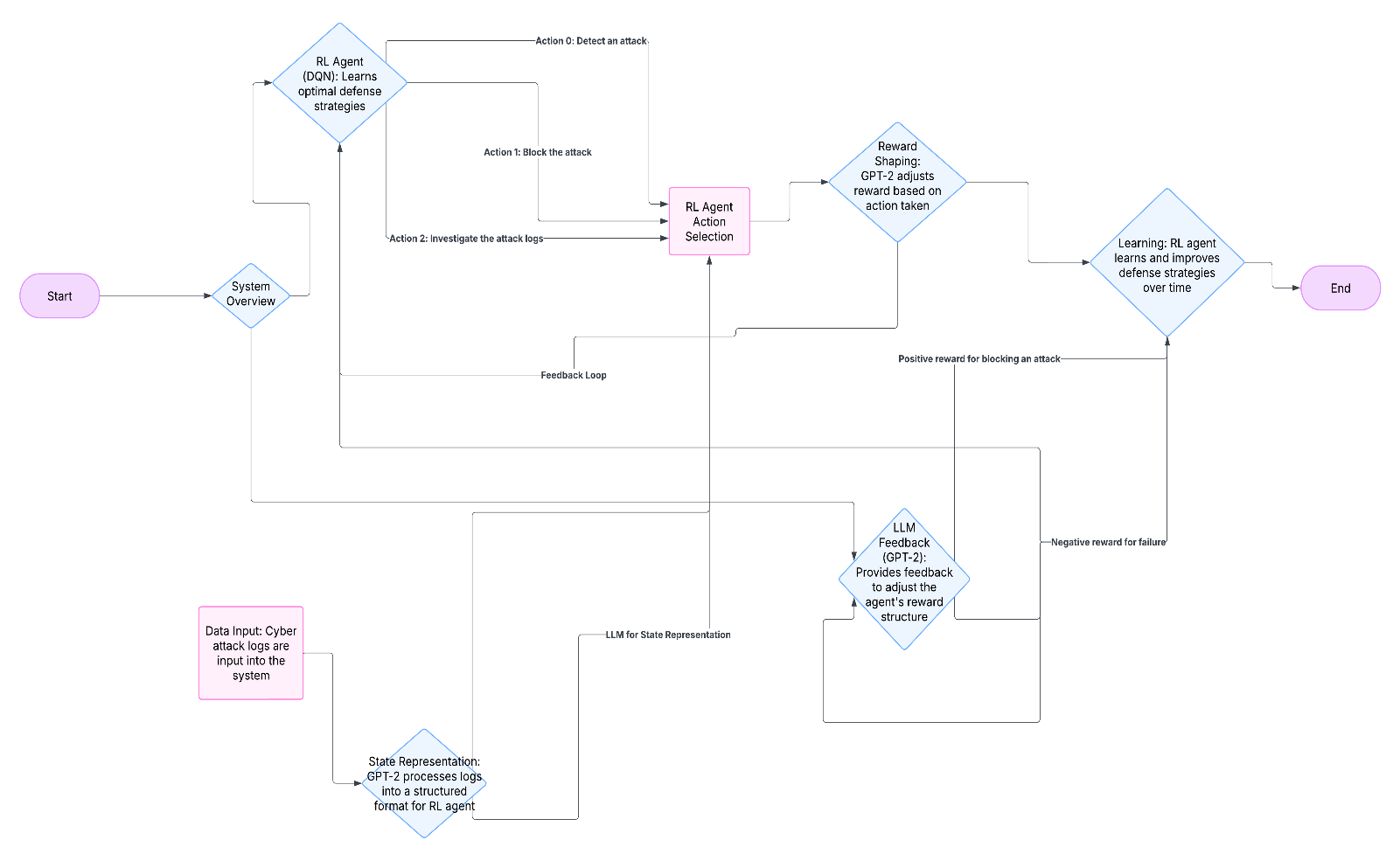
The system is built using Gymnasium, which provides a standardized interface for RL tasks. The CyberDefenseEnv class simulates the cybersecurity environment, and the agent learns to interact with it. To speed up the training process one environment is used at a time to train the RL agent. This was done to reduce computation time and focus on core functionality of the agent in the initial prototype phase. In future instances parallel environments can be used for scaling.

**3. Model Training:**

The **DQN model** is trained using 500 timesteps. The RL agent learns from the environment and receives feedback from the LLM, refining its reward function and policy over time. The agent’s goal is to maximize the cumulative reward, learning effective strategies for defending against a variety of cyber threats. In addition, **epsilon decay** is introduced to allow the agent to transition from high exploration to more exploitation as training progresses.  
  
**Epsilon decay-**To improve the learning efficiency and ensure that the agent transitions from exploration to exploitation, epsilon decay is applied. At the start, the agent starts with a high exploration rate (epsilon = 1.0), which encourages the agent to try different actions. Over time, the epsilon value gradually decays using a decay rate of 0.995, reducing the exploration and allowing the agent to focus more on using the strategy it learned in the process. This helps the agent with its decision-making, ensuring better defence strategies as it learns from feedback over time.

* **Training Process**:
  + The agent explores the environment and takes actions based on its current policy.
  + The LLM processes attack logs and provides feedback, which shapes the reward function and helps the agent learn faster.
  + The agent is updated using Q-learning, adjusting its policy to increase its long-term reward.

**4. Reward Shaping Using LLM:**

The integration of GPT-2 for reward shaping is a key here. The LLM processes attack logs which provides feedback, such as “Attack neutralized” or “Investigation required,” which adjusts the reward. This enables the agent to learn dynamically, adapting its behaviour based on feedback from the LLM rather than relying on predefined rules.  


**5. Prototype Workflow:**

The workflow of the system is as follows:

1. **Input**: The system receives attack logs from the environment.
2. **Processing**: GPT-2 processes the logs and provides structured feedback.
3. **Action**: The RL agent chooses an action (detect, block, investigate) based on its current state and feedback from the LLM.
4. **Reward**: The LLM generates feedback, which is used to shape the reward function and refine the agent’s actions.
5. **Output**: The agent continually adapts and improves its defence strategy.

**6. Evaluation and Results:**

**Evaluation:**

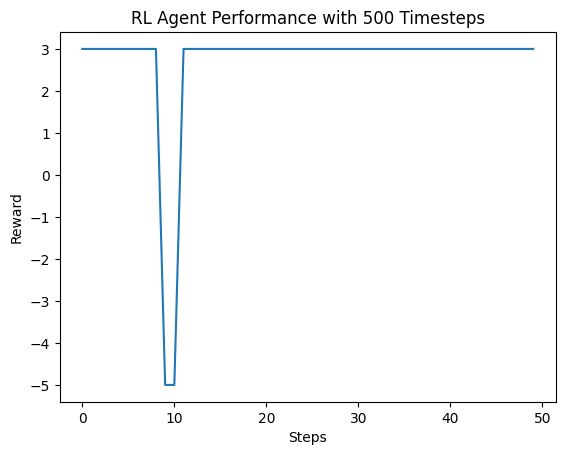
Based on the reward graph and the training logs, the reinforcement learning (RL) agent shows promising adaptation. The agent is successfully learning to detect attacks most of the time, with rewards dictating that its actions are generally moving in the right direction. However, there are occasional instances where the agent fails to act optimally, resulting in negative rewards (e.g., Reward: [-5. -5. -5.]). This suggests that while the agent is adapting well to its environment and improving its decision-making, it occasionally makes mistakes or takes suboptimal actions.

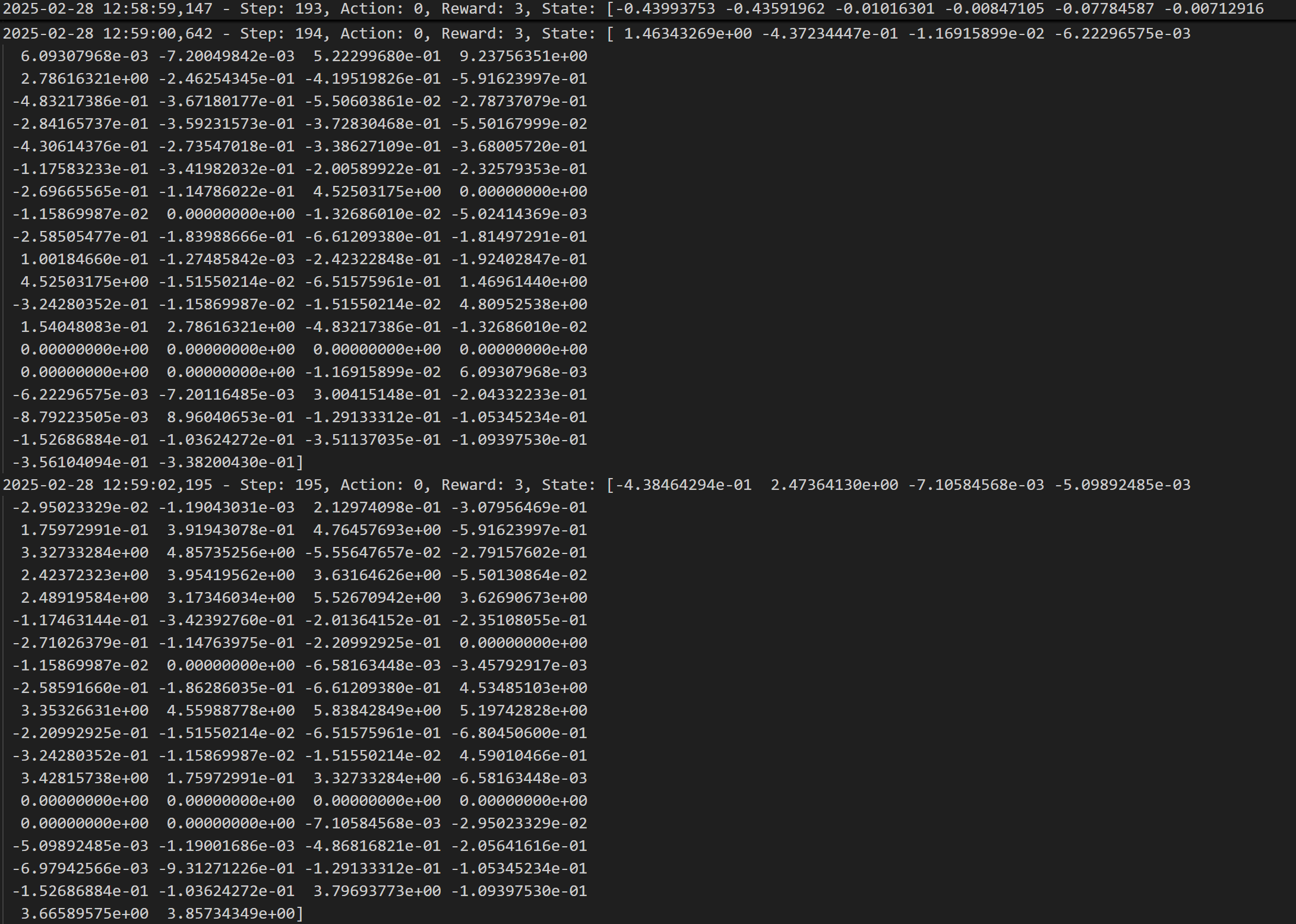
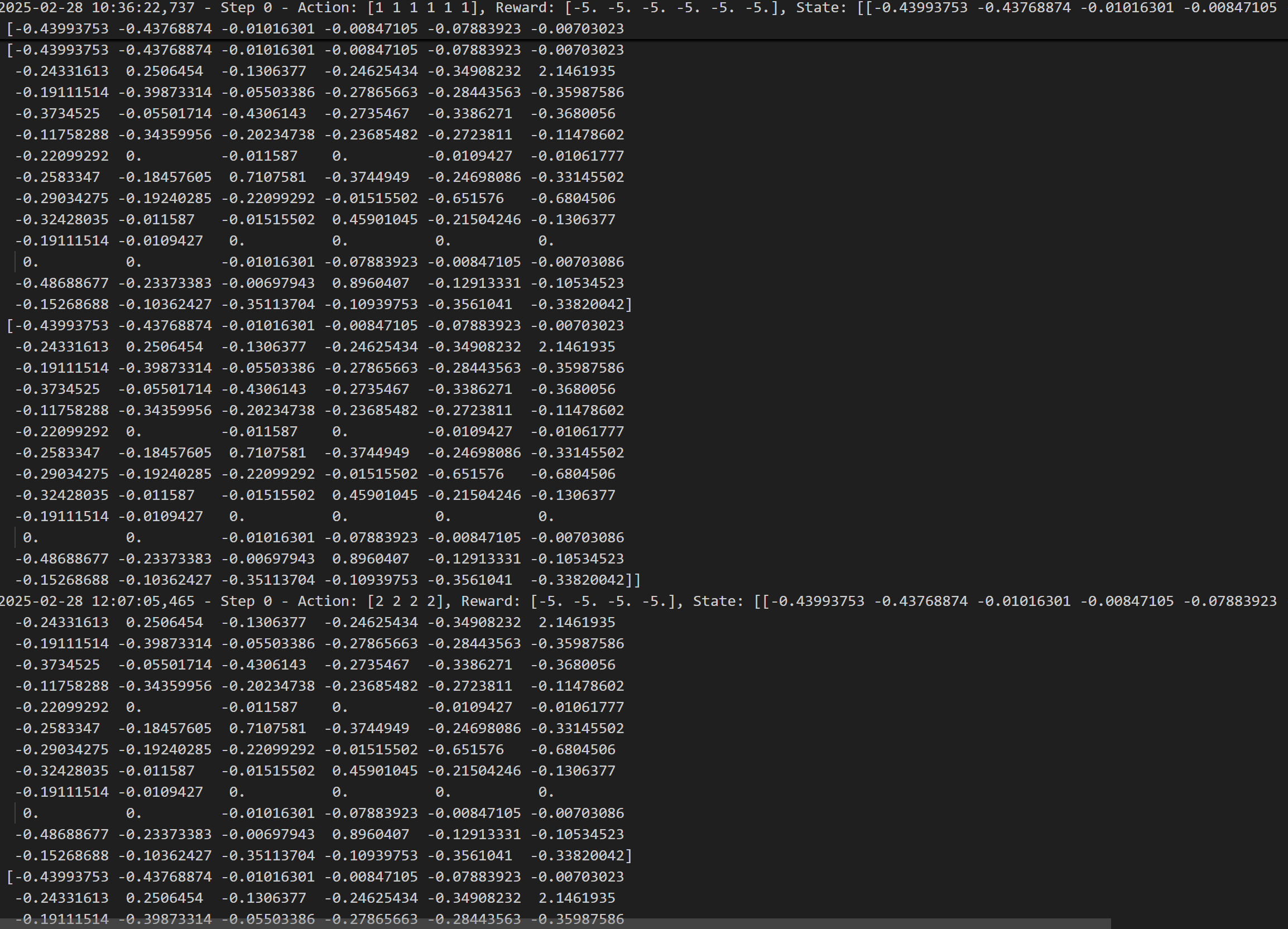
**Training Log Analysis:**

* **Adapting to the Environment**: The training logs depict that, as the agent’s progress is good, its actions become more centred with the attack data. For instance, detecting attacks and blocking them often results in positive rewards.
* **Occasional Failure**: Despite improving, the agent still sometimes fails to take the correct action, which leads to negative rewards (e.g., in cases where it investigates instead of blocking a real-time attack).
* **Exploration vs. Exploitation**: The agent is still in a phase of high exploration (due to epsilon decay), trying various actions. This behaviour is expected during early training, and it is likely that with more iterations, the agent will refine its strategy and become more efficient.

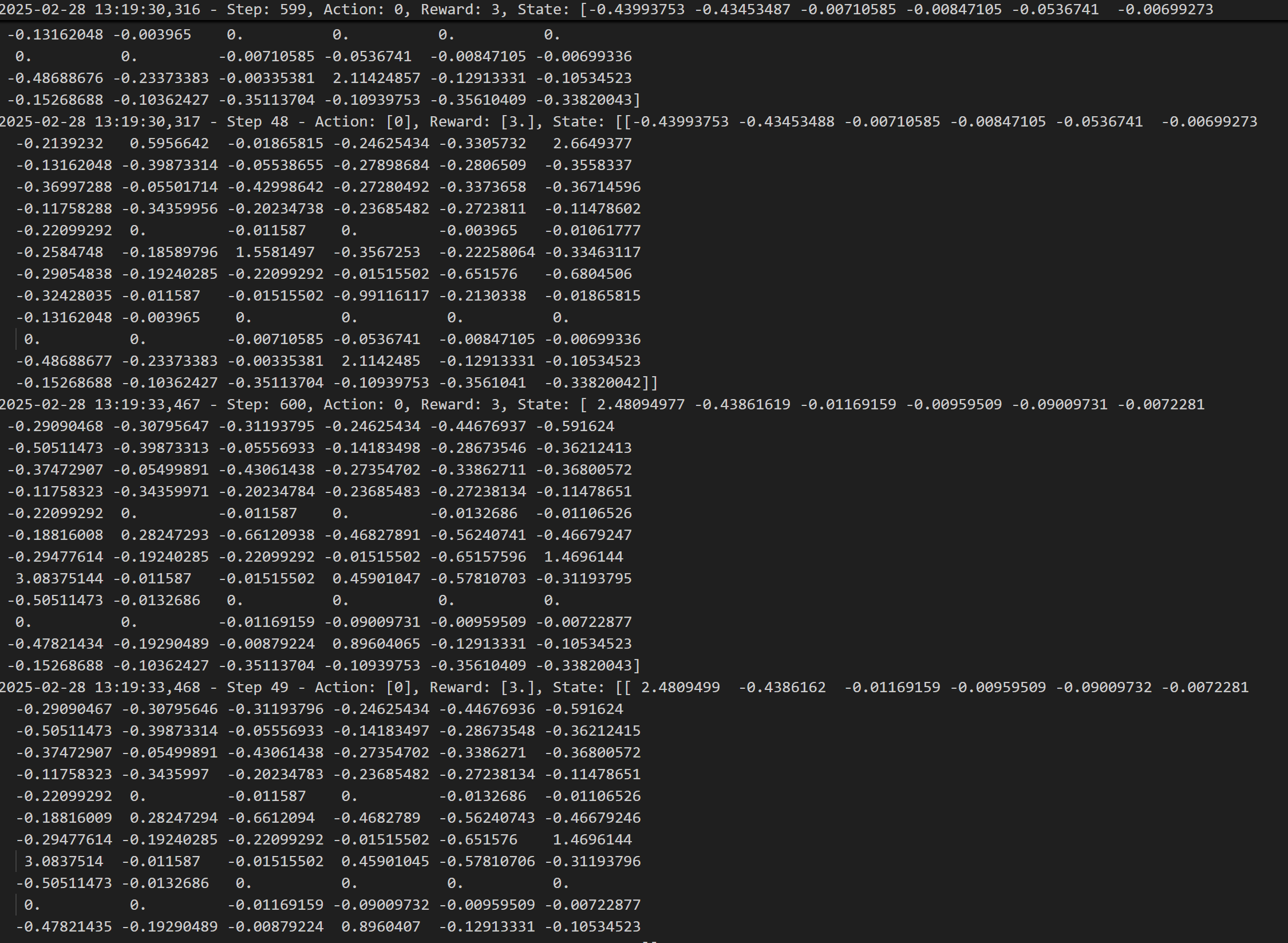
**Results:**

* The reward graph shows that, while the agent initially struggles for a bit, it adapts over time and increasingly detects and reacts to attacks.
* Occasional mistakes are reflected by negative rewards, indicating that while the agent is improving, it still needs additional time and iterations to optimize its policy.
* The rapid fluctuation in rewards (a few positive spikes) indicates that the agent is experimenting with different strategies, and once it is given more iterations, it will likely converge to an optimal defense strategy.





Model leaning at initial stage and getting feedback After some iterations is able to identify the pattern



Correct detection of attacks after multiple iterations

**Code Snippets**

