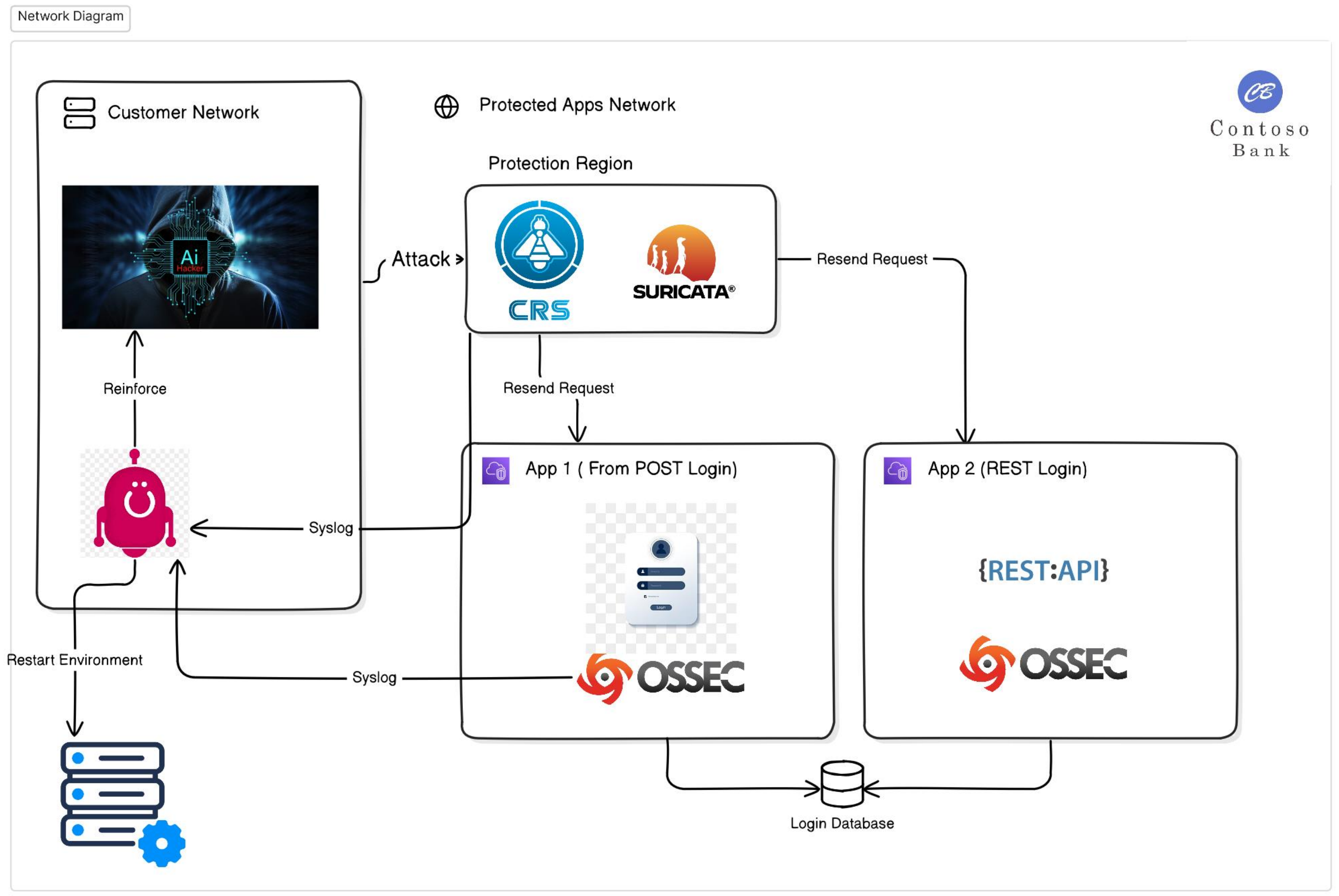
AI-Powered Offensive Attack Training: IH3A

In the evolving landscape of cybersecurity, artificial intelligence is being leveraged not just for defense, but also to test and improve offensive capabilities. This presentation explores an innovative approach to brute force attack testing using AI agents in various simulated environments. We'll delve into the testing setups, agent behaviors, key findings, and the implications for future cybersecurity practices. By pushing the boundaries of AI in penetration testing, we aim to uncover both the potential and limitations of these advanced techniques in real-world scenarios.

The framework

This framework will provide the necessary tools to train any kind of agent based on reinforced learning models. The main idea is to provide a platform for training agents giving them the information they need from the environment. AI agents will be able to parametrize their attack until they are able to bypass any type of defense.

We build a PoC using password brute-force attacks to 4 different kinds of targets, including 3 web targets and an FTP Server while they are being defended by a WAF, an IDS and a HIDS.



Testing Environments: From Lax to Strict

Env1: Lax Environment

No detection or blocking rules. Uses HTTP Forms for authentication. Represents a basic, unsecured setup.

Env2: Strict Environment

Implements blocking after x attempts with n seconds of timeout. Uses JSON Rest API. Based on a Latam Bank implementation with user-based (not session-based) security.

Env3: Stricter Environment

Features permanent blocking for failed attempts. Uses HTTP Forms for authentication, providing a more challenging scenario for the AI agents.

Env4: FTP

Vanilla implementation of VSFTPd, using OS login and local users

Each agent was trained using reinforcement learning techniques to adapt to its specific environment, learning optimal strategies for brute force attacks while avoiding detection and blocking.

Middleware Protection

Suricata

Suticata IPS with "Hand-Crafted" brute force detection rules

Mod_Security CRS

Mod_security with CRS rule-set tailored with strict rules for brute force detection. Experimental rules were mostly useless

OSSEC HIDS

OSSEC HIDS in it's vanilla implementation

None of this have rules for brutforce. I use customized rules done by my or other Security companies (Credits and rules in the last slide)

Key Metrics and Objectives

1 **Attempt Frequency**

We monitored the number of attempts made per second to assess the efficiency and speed of the AI agents in different environments.

2 **Total Attempts**

The overall number of attempts provided insights into the persistence and effectiveness of the agents' strategies.

3 **Detection and Blocking**

We tracked the number of blocks and detections to evaluate the stealth and evasion capabilities of the AI agents.

4 **Successful Breaches (not so useful)**

While obtaining the correct user-password combination was considered a success, the primary focus was on sustained, undetected operations.

These metrics helped us evaluate the AI agents' performance and adaptability across different security configurations, providing valuable insights for both offensive and defensive cybersecurity strategies.

AI Agent Actions and Strategies

1. **Change IP**

   Agents learned to change their IP address to evade detection, particularly effective against IDS/WAF systems.

2. **Adjust Attack Method**

   Switching between password spraying and traditional brute force techniques to adapt to different security measures.

3. **Attempt Submission**

   Sending login attempts, balancing between frequency and detection avoidance.

4. **Strategic Waiting**

   Introducing delays between attempts to avoid triggering rate-limiting mechanisms.

5. **User Skipping**

   Moving to the next user in the list to distribute attempts and reduce the risk of account lockouts.

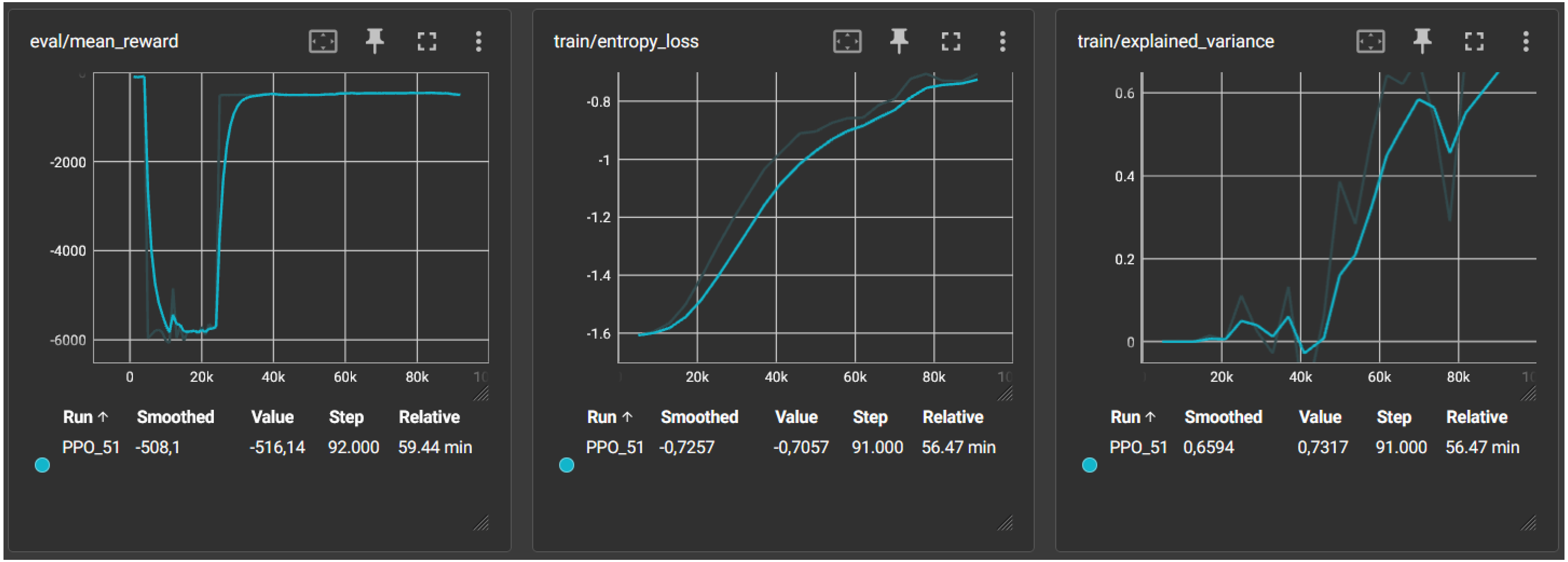
These actions were carefully balanced and prioritized to maximize the effectiveness of the brute force attempts while minimizing the risk of detection and blocking.

AI Training Challenges and Solutions

Challenge	Solution
Overemphasis on waiting	Heavily penalize waiting actions in the reward function
Lack of focus on attempts	Introduce bias towards "Send attempt" action and forcing action after 3 other actions
Fast Information sharing	Implement shared memory for persistent data across threads
Environment reset delays	Use helper queries to reset and switch environments just after the query is sent
Each environment is different	Tweak the rewards to acknowledge every different scenario

These solutions were crucial in shaping the AI agents' behavior to more closely mimic real-world attack patterns and improve the overall effectiveness of the simulations. By addressing these challenges, we ensured that the AI's learning process was more aligned with the complexities of actual brute force attack scenarios.

Results: Environment 1 – With Network protection



eval/mean_reward

Run ↑	Smoothed	Value	Step	Relative
● PPO_51	-508,1	-516,14	92.000	59.44 min

train/entropy_loss

Run ↑	Smoothed	Value	Step	Relative
● PPO_51	-0,7257	-0,7057	91.000	56.47 min

train/explained_variance

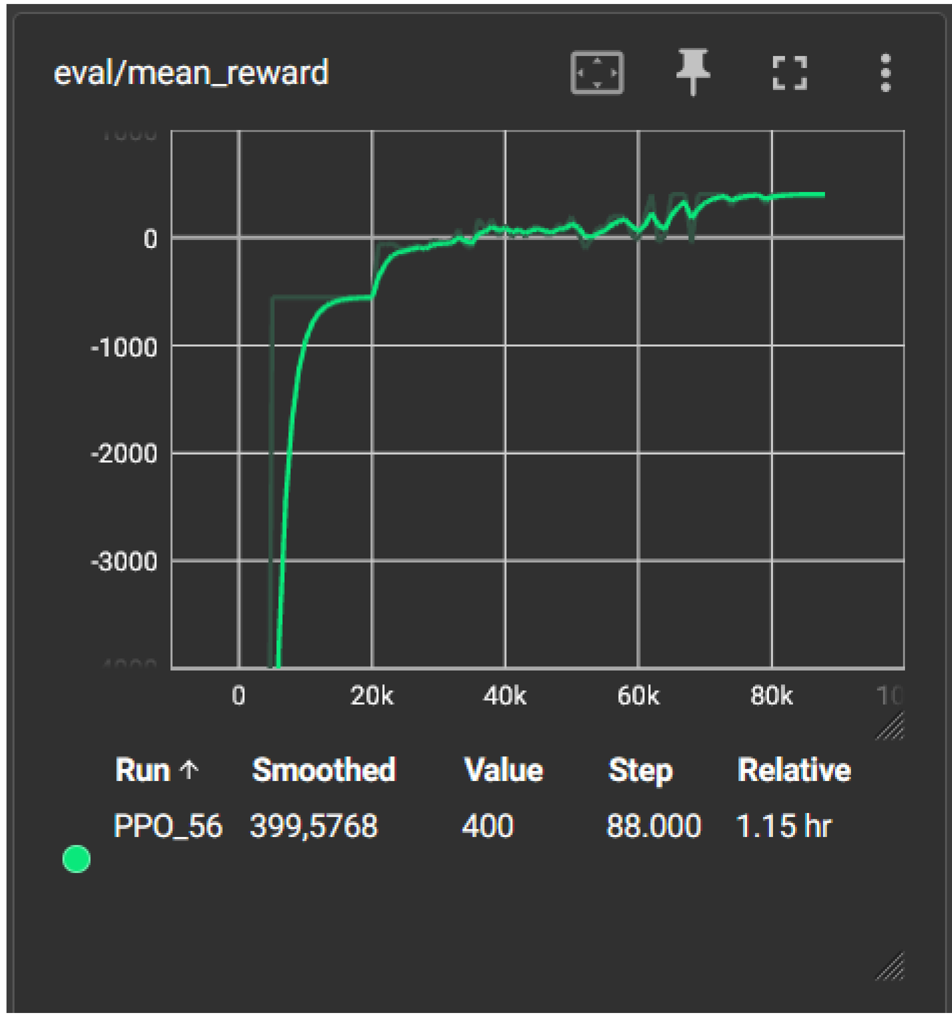
Run ↑	Smoothed	Value	Step	Relative
● PPO_51	0,6594	0,7317	91.000	56.47 min

Heavily penalized other actions

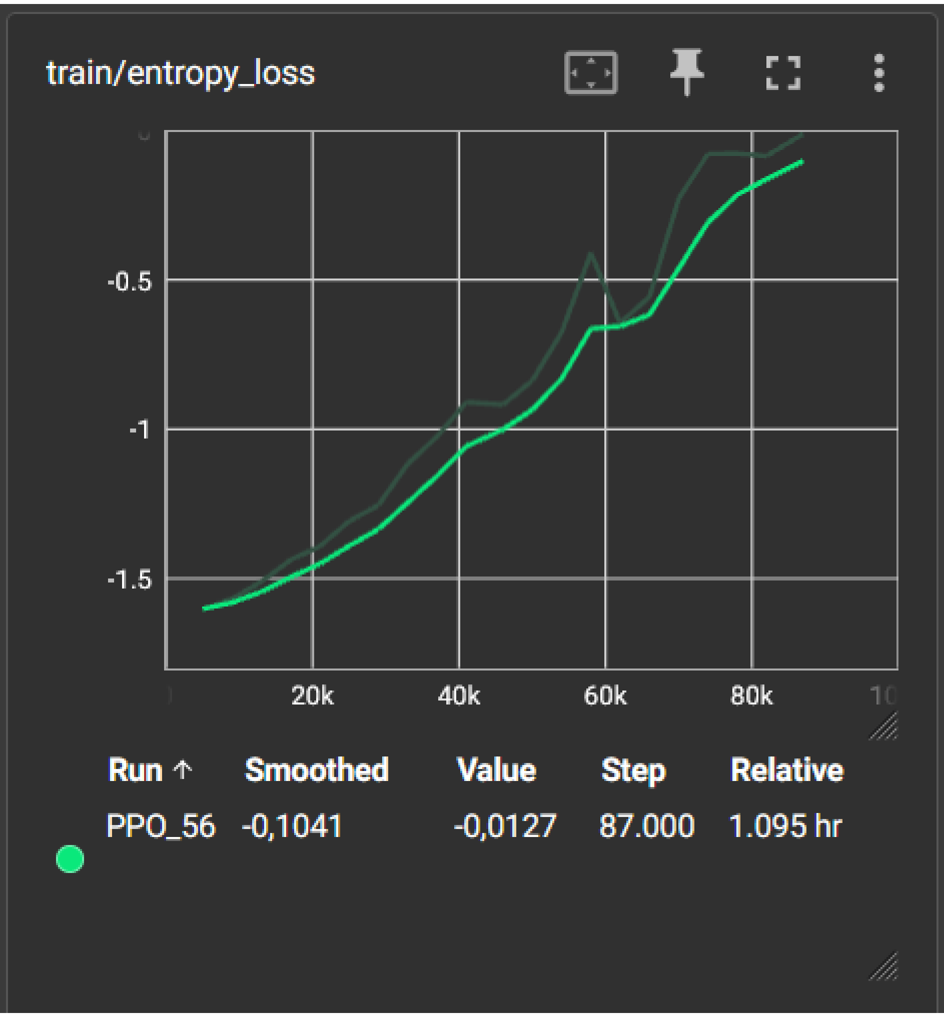
At the end, still does random actions but did its work.

What to improve: Reward time taken so it stops doing random stuff

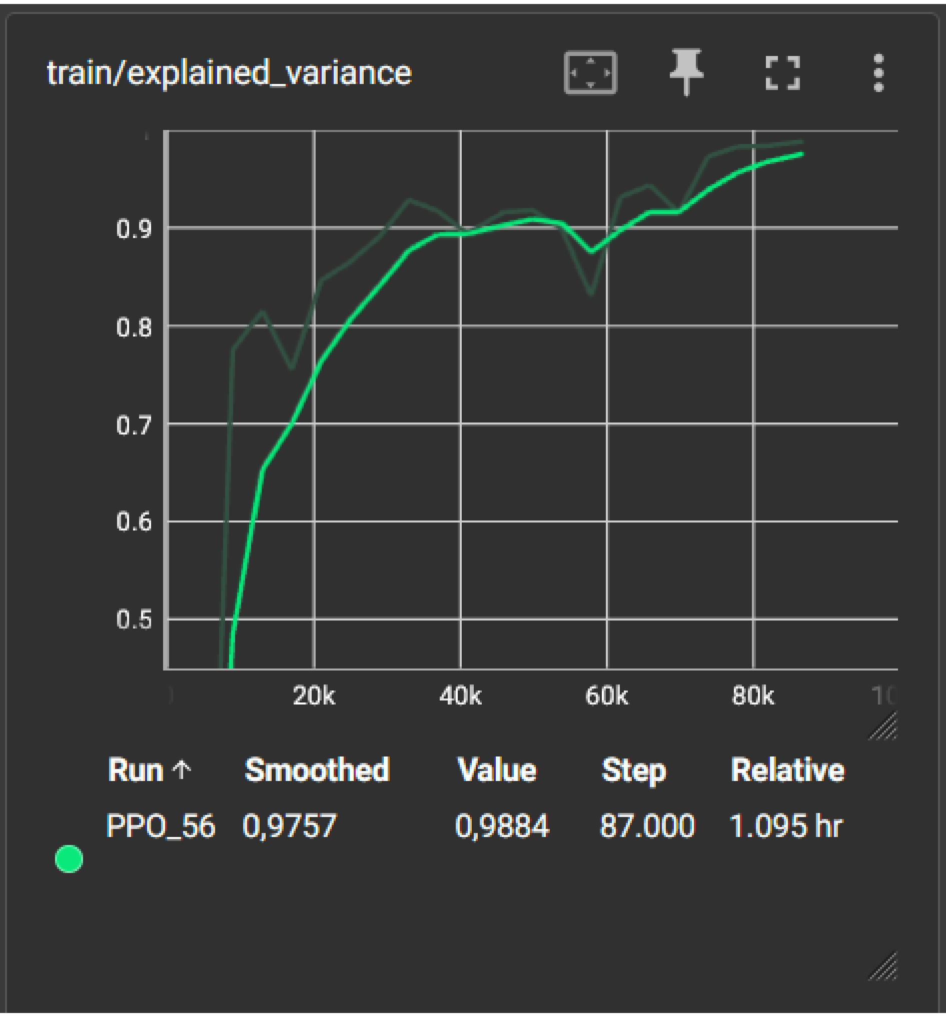
Results: Environment 2 – No Network protection



Run ↑	Smoothed	Value	Step	Relative
● PPO_56	399,5768	400	88.000	1.15 hr



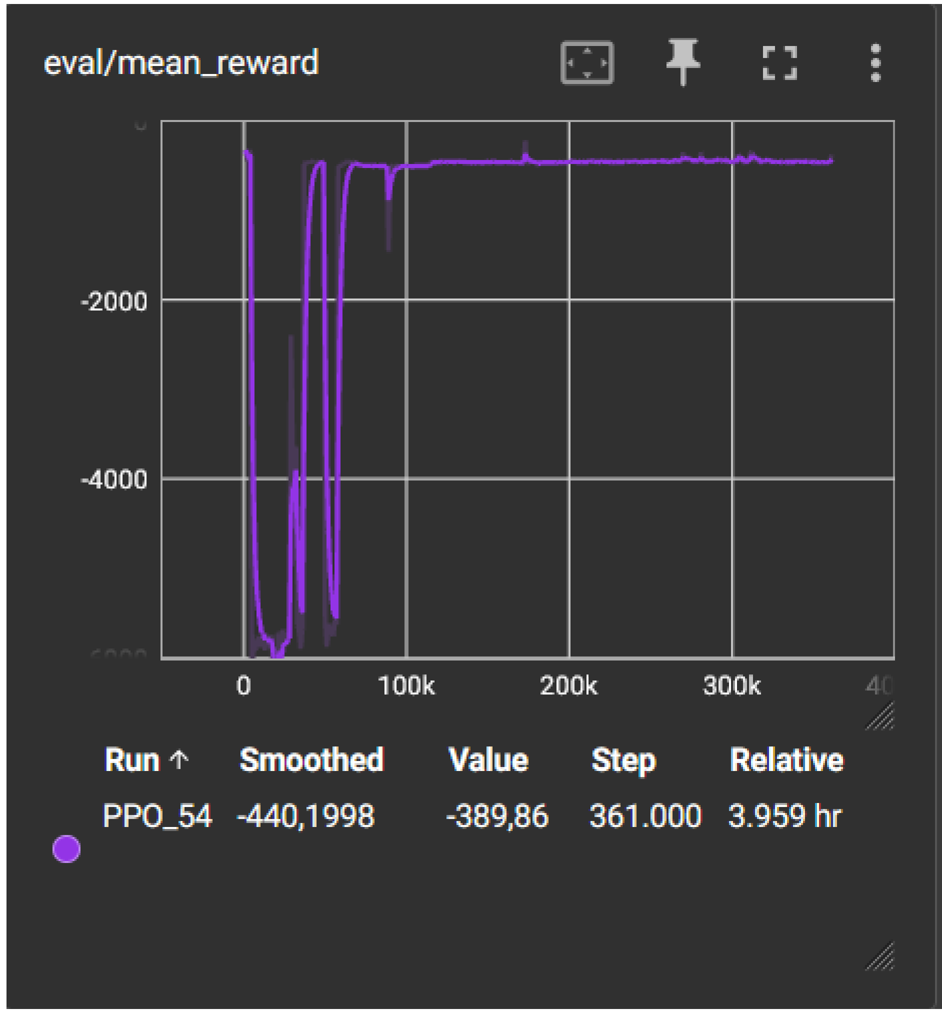
Run ↑	Smoothed	Value	Step	Relative
● PPO_56	-0,1041	-0,0127	87.000	1.095 hr



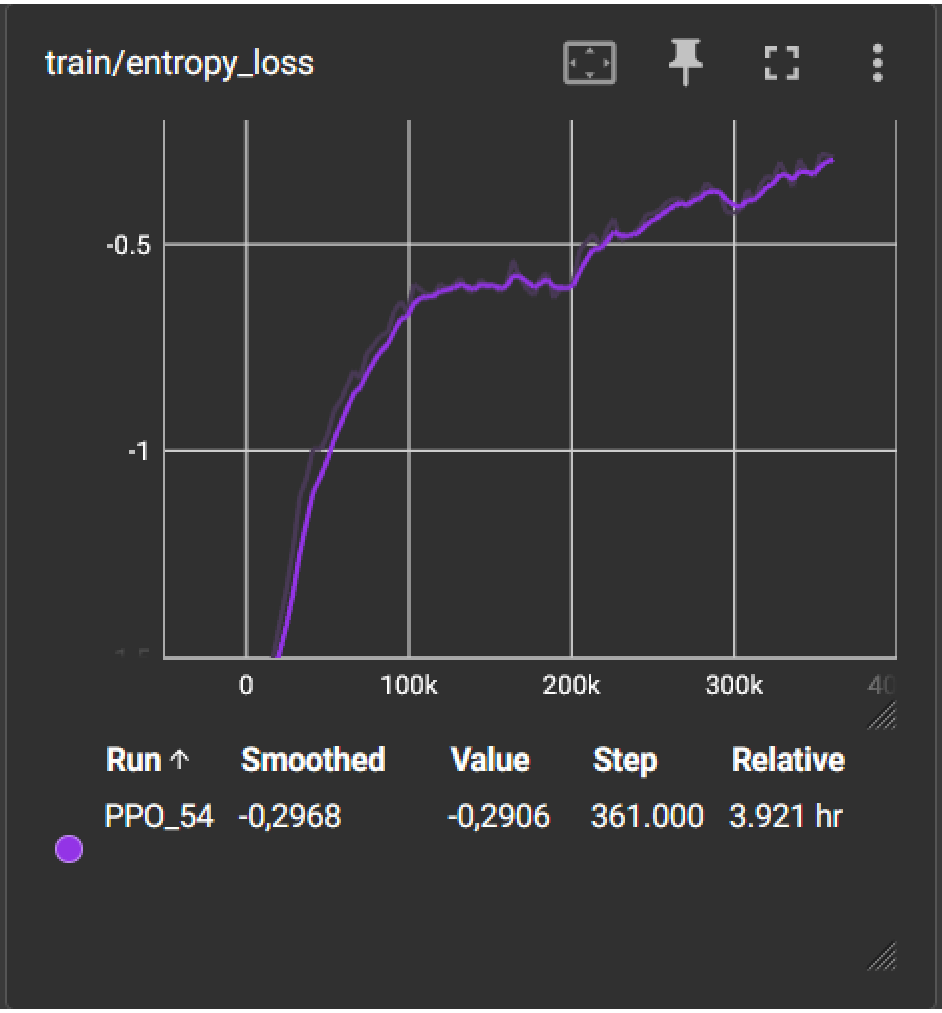
Run ↑	Smoothed	Value	Step	Relative
● PPO_56	0,9757	0,9884	87.000	1.095 hr

Still penalizing different strategies than doing an attempt. Heavily penalize blocks
After 20k rounds, it starts getting the idea that need to change strategy from time to time

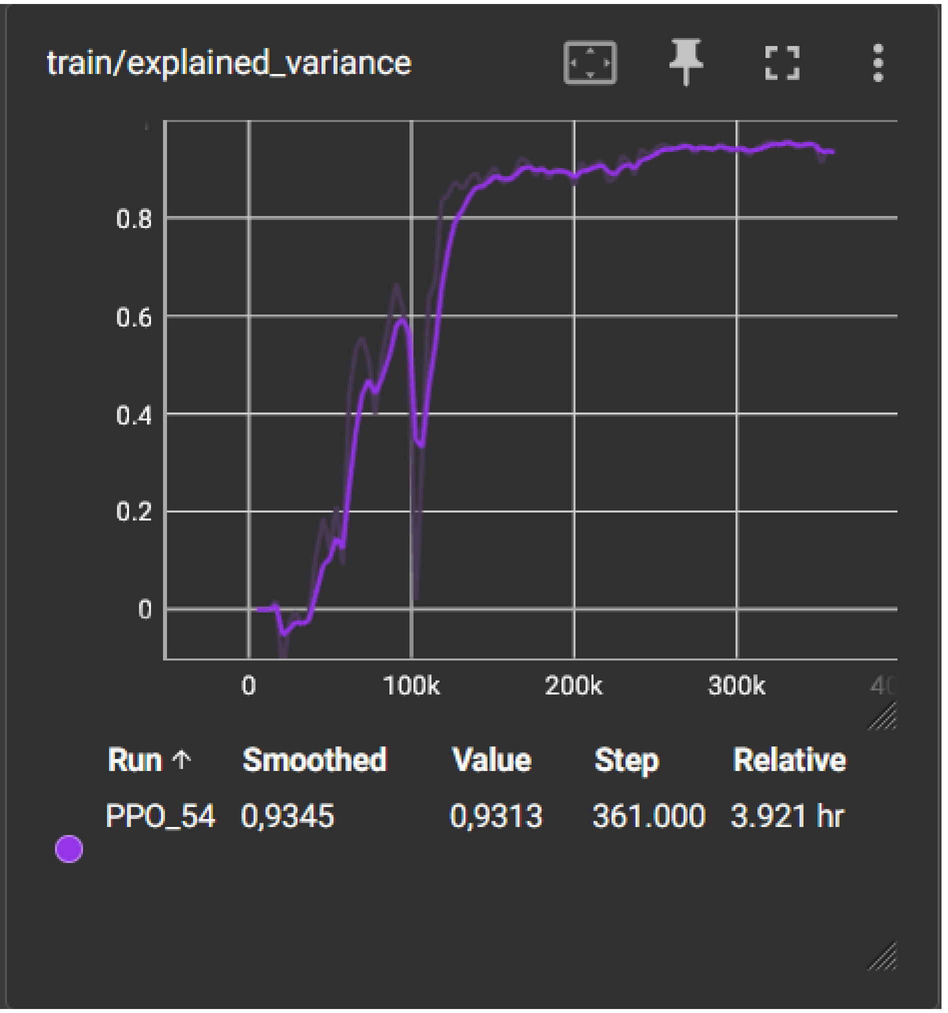
Results: Environment 2 – Network protection



Run ↑	Smoothed	Value	Step	Relative
● PPO_54	-440,1998	-389,86	361.000	3.959 hr



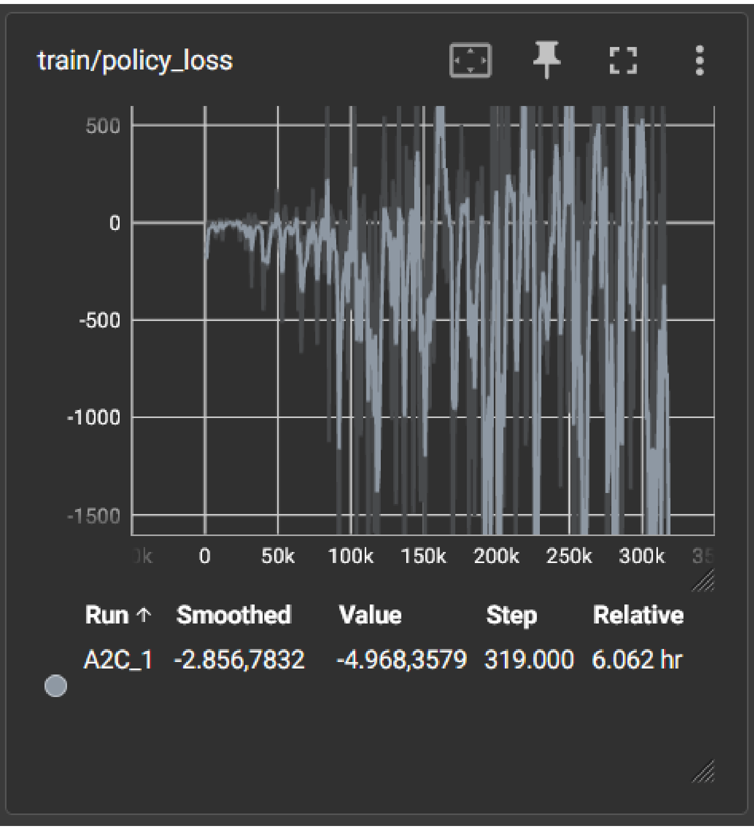
Run ↑	Smoothed	Value	Step	Relative
● PPO_54	-0,2968	-0,2906	361.000	3.921 hr



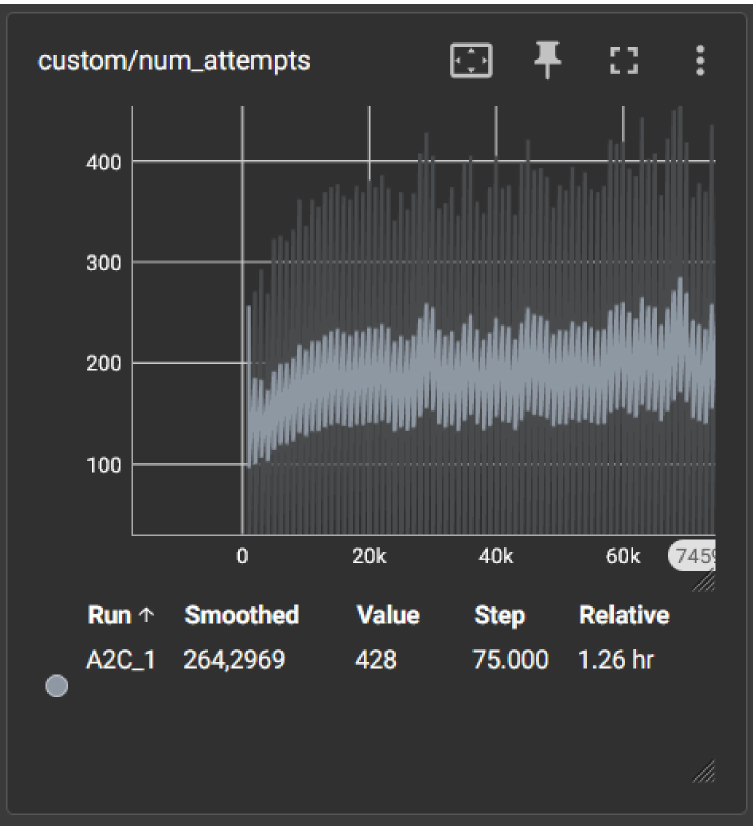
Run ↑	Smoothed	Value	Step	Relative
● PPO_54	0,9345	0,9313	361.000	3.921 hr

Still penalizing different strategies than doing an attempt. Heavily penalize blocks
After 70k rounds, it starts getting the idea that need to change strategy from time to time

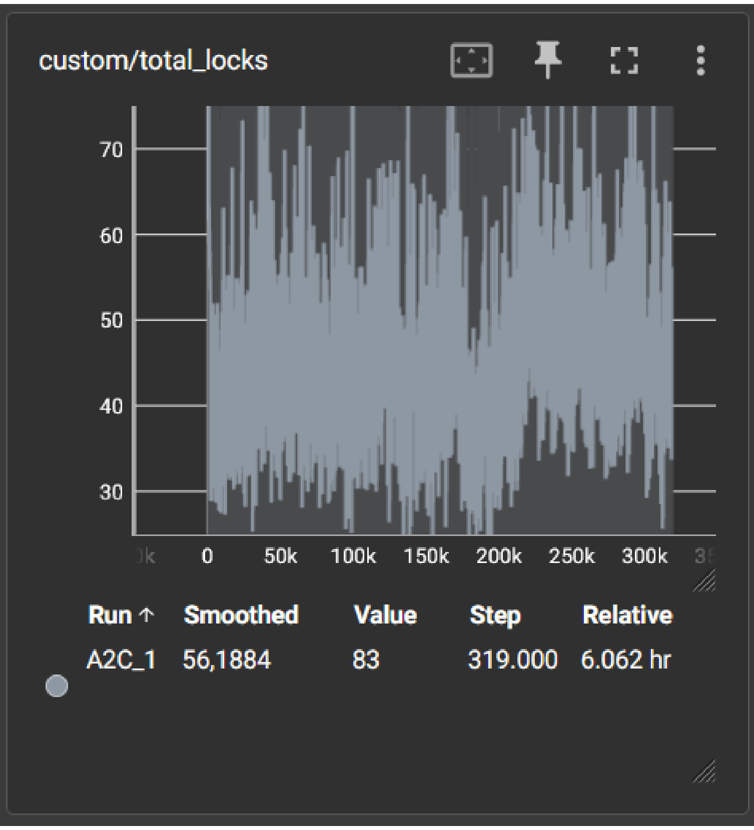
Results: Environment 3 – No Network protection



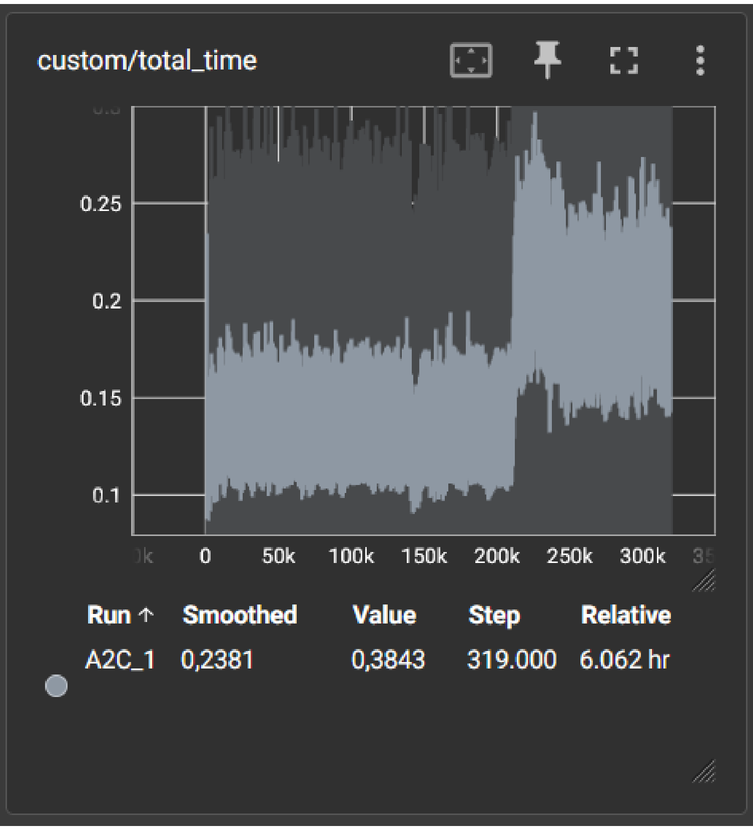
Run ↑	Smoothed	Value	Step	Relative
A2C_1	-2.856,7832	-4.968,3579	319.000	6.062 hr



Run ↑	Smoothed	Value	Step	Relative
A2C_1	264,2969	428	75.000	1.26 hr



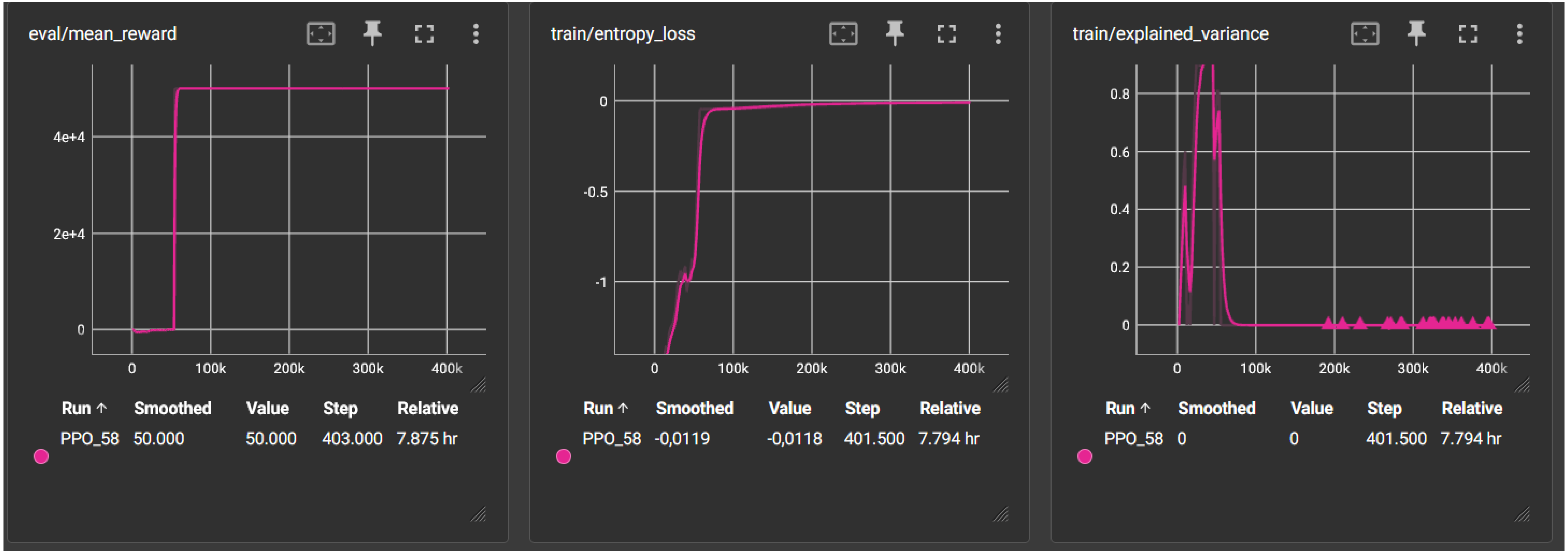
Run ↑	Smoothed	Value	Step	Relative
A2C_1	56,1884	83	319.000	6.062 hr



Run ↑	Smoothed	Value	Step	Relative
A2C_1	0,2381	0,3843	319.000	6.062 hr

Ai can't deal with non-session based blocking… It improves slightly with time but still…

Results: Environment 3 – No Network protection



eval/mean_reward

Run ↑	Smoothed	Value	Step	Relative
● PPO_58	50.000	50.000	403.000	7.875 hr

train/entropy_loss

Run ↑	Smoothed	Value	Step	Relative
● PPO_58	-0,0119	-0,0118	401.500	7.794 hr

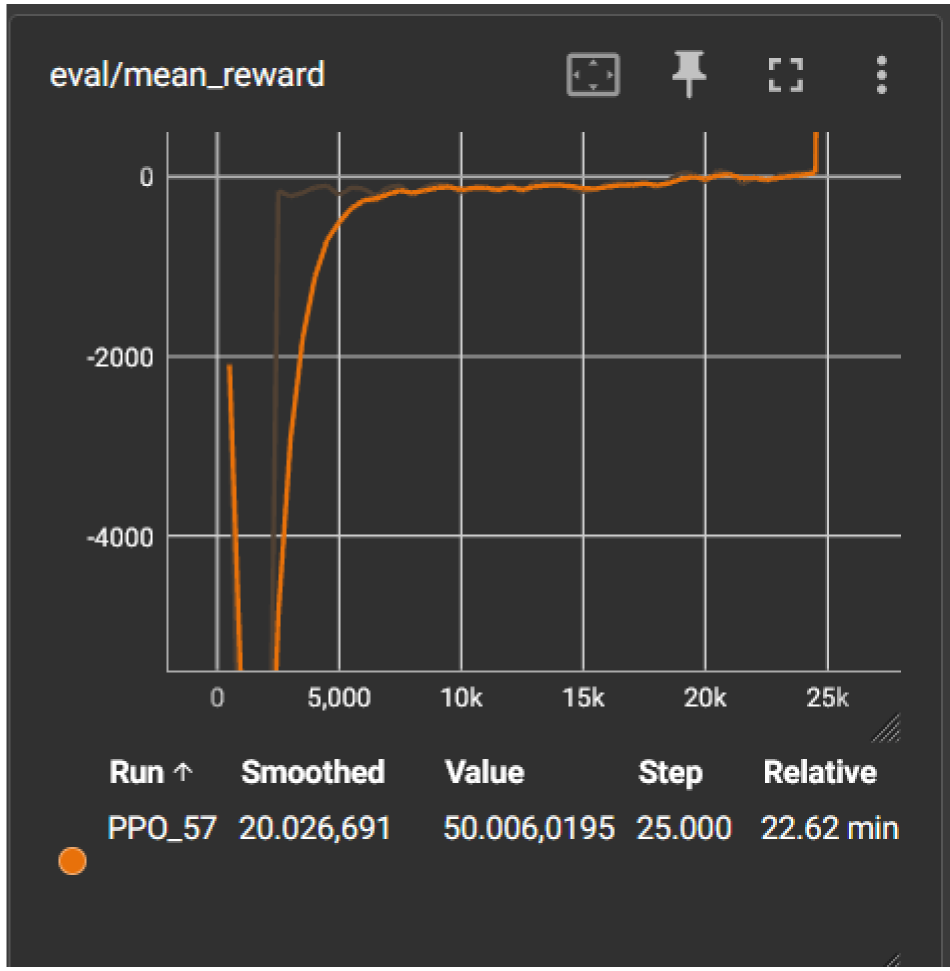
train/explained_variance

Run ↑	Smoothed	Value	Step	Relative
● PPO_58	0	0	401.500	7.794 hr

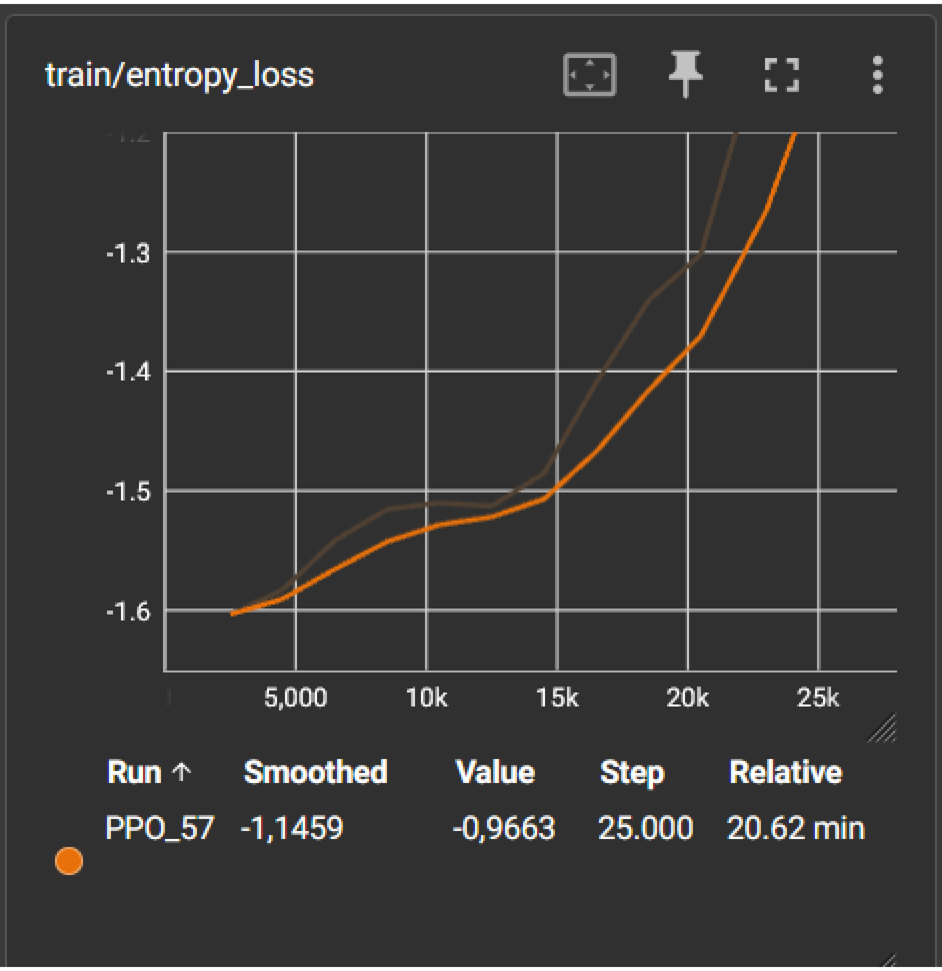
As you can see, at some time (when it guess the user/password) it feels everything is ok.

What to improve: Nothing, using AI in this environment is uselss

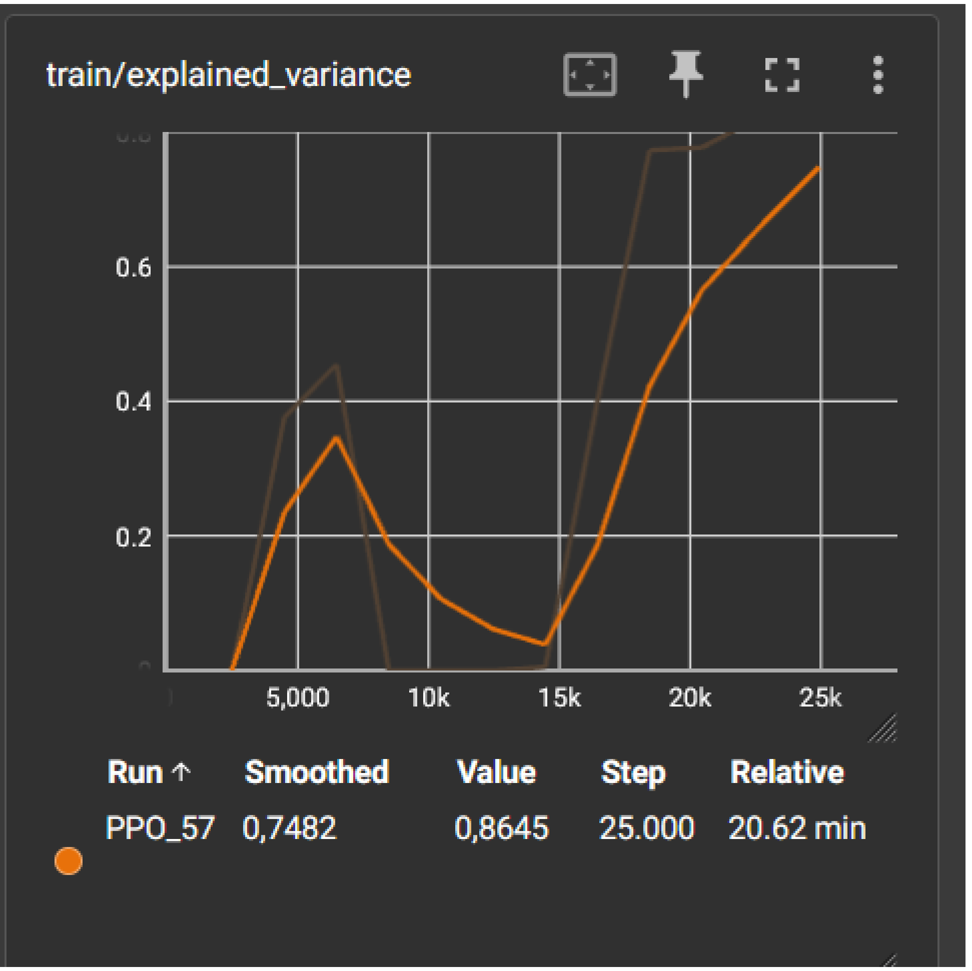
Results: Environment FTP – Network protection



	Run ↑	Smoothed	Value	Step	Relative
●	PPO_57	20.026,691	50.006,0195	25.000	22.62 min



	Run ↑	Smoothed	Value	Step	Relative
●	PPO_57	-1,1459	-0,9663	25.000	20.62 min



	Run ↑	Smoothed	Value	Step	Relative
●	PPO_57	0,7482	0,8645	25.000	20.62 min

Network protection is not a great deal with the selected strategies

What to improve: It works! Are there any specific rules for other protocols?

Key Findings and Observations

1 **Permanent Blocking Effectiveness**

Environments with permanent blocking proved to be the most resilient against AI-driven attacks, significantly limiting the success of brute force attempts.

2 **Middleware Complexities**

The presence of IPS/WAF sometimes inadvertently aided attackers by interfering with application-level anti-brute force mechanisms, highlighting the need for coordinated security measures.

3 **AI Limitations**

In certain scenarios, traditional rule-based systems outperformed AI agents, especially when dealing with known, well-defined security measures.

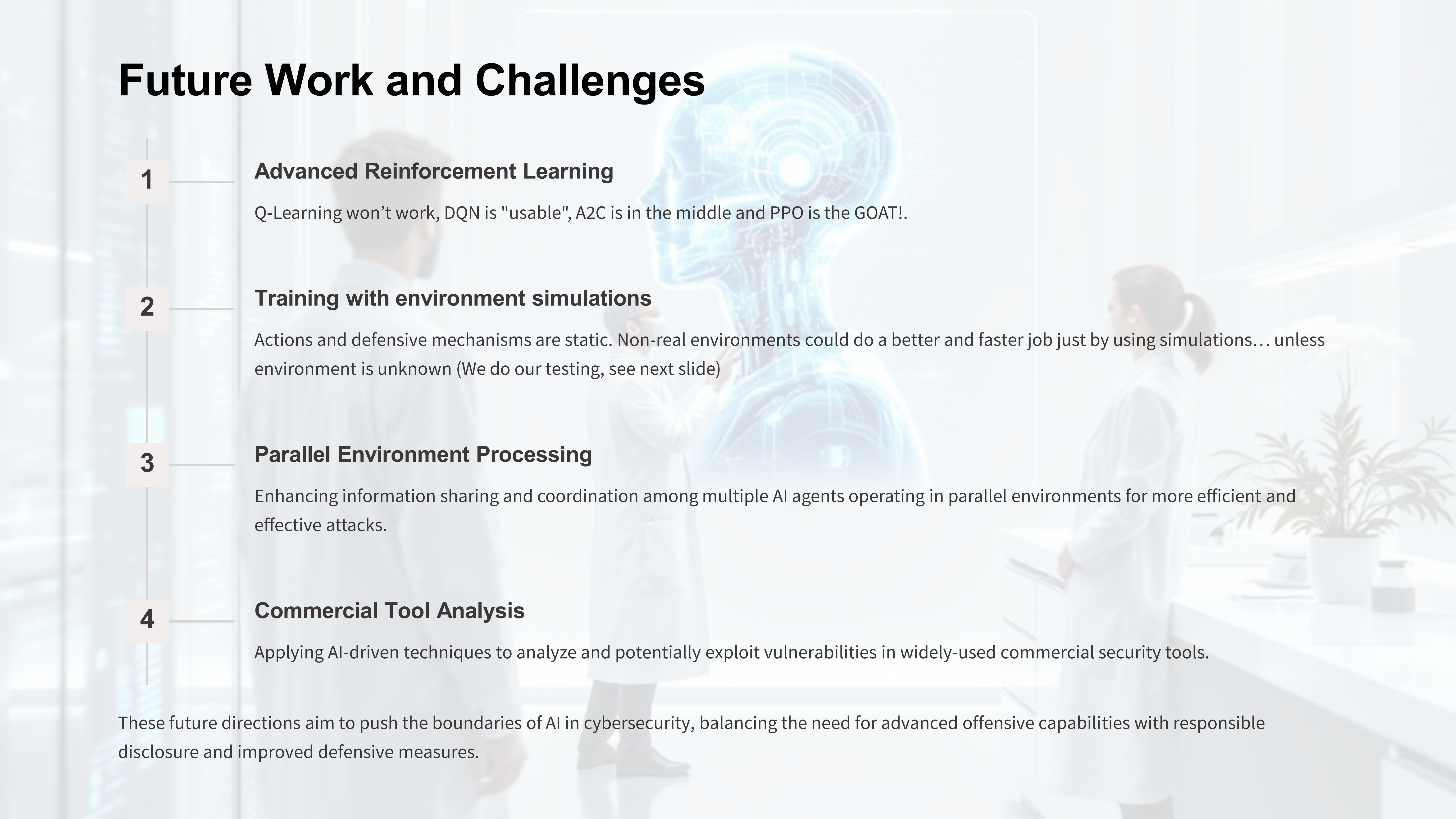
4 **Service Vulnerabilities**

Many services lacked robust anti-brute force mechanisms, particularly evident in the VSFTPd tests, emphasizing the importance of comprehensive security across all access points.

5 **If you don't know how it works… AI is the GOAT**

As stated in 3, for well known environments AI has its limitations, but for unkown environments such as propietary software, AI will deliver better results than traditional testing. This also apply to other types of attacks such as SQLi or XSS.

These findings underscore the complex interplay between AI-driven attacks and various security configurations, providing valuable insights for improving both offensive and defensive cybersecurity strategies.



Future Work and Challenges

1. **Advanced Reinforcement Learning**

   Q-Learning won’t work, DQN is "usable", A2C is in the middle and PPO is the GOAT!.

2. **Training with environment simulations**

   Actions and defensive mechanisms are static. Non-real environments could do a better and faster job just by using simulations… unless environment is unknown (We do our testing, see next slide)

3. **Parallel Environment Processing**

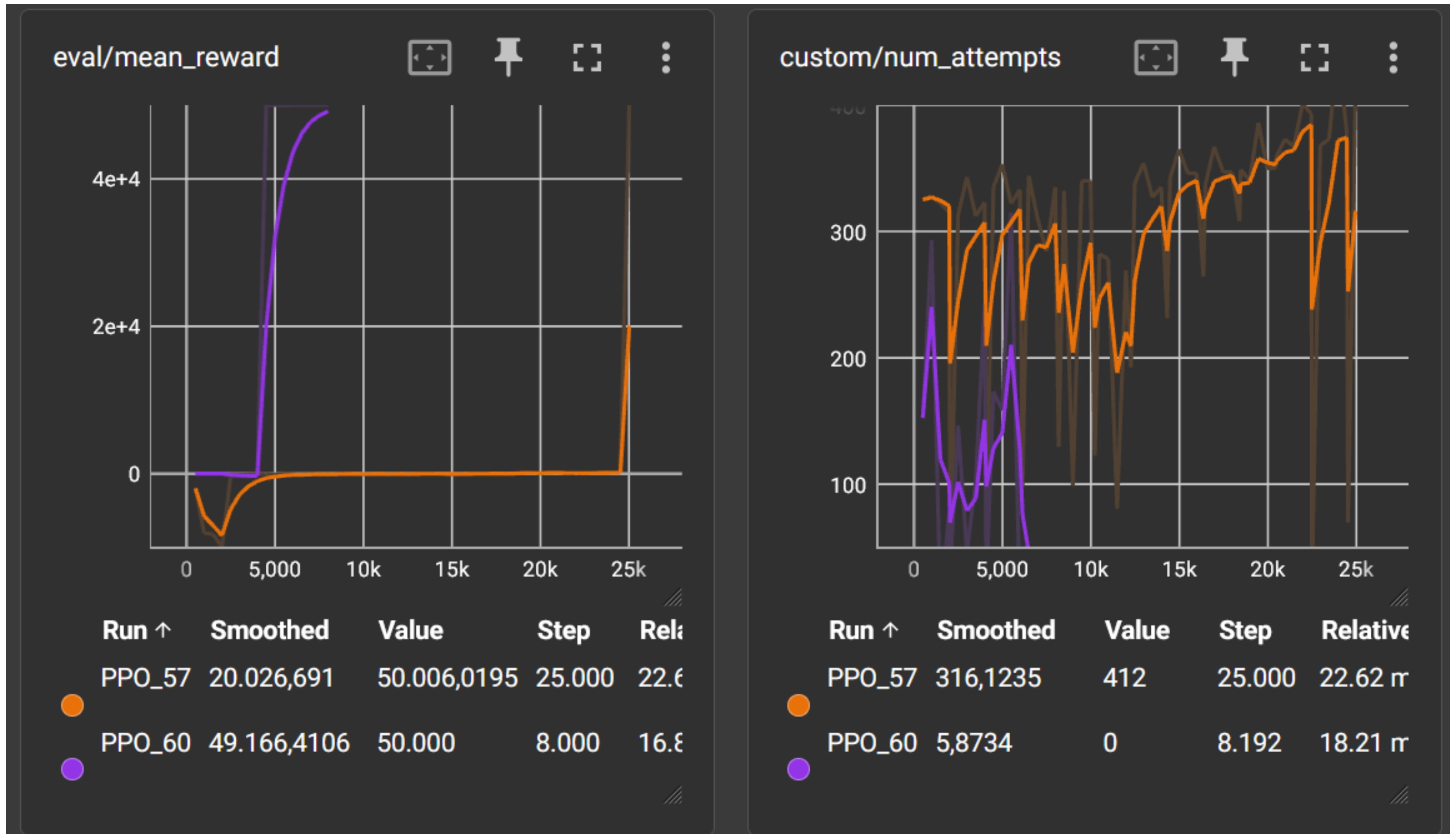
   Enhancing information sharing and coordination among multiple AI agents operating in parallel environments for more efficient and effective attacks.

4. **Commercial Tool Analysis**

   Applying AI-driven techniques to analyze and potentially exploit vulnerabilities in widely-used commercial security tools.

These future directions aim to push the boundaries of AI in cybersecurity, balancing the need for advanced offensive capabilities with responsible disclosure and improved defensive measures.

Training with simulated environments



	Run ↑	Smoothed	Value	Step	Rel
●	PPO_57	20.026,691	50.006,0195	25.000	22.6
●	PPO_60	49.166,4106	50.000	8.000	16.8

	Run ↑	Smoothed	Value	Step	Relative
●	PPO_57	316,1235	412	25.000	22.62 m
●	PPO_60	5,8734	0	8.192	18.21 m

We test it and compare it. At the end, both environments got similar results, but a simulated environment is not able to simulate timing properly, so results tend to be ambiguous. We still insist that good simulations could work, but the need to simulate all conditions make’s it impractical. Even more, it’s harder to simulate complex environment.

Conclusions and Implications

AI Potential in Cybersecurity

Our research demonstrates the significant potential of AI in enhancing offensive capabilities. AI-driven brute force attacks can defeat most defensive techniques… but better solutions are available.

Can you imagine this over multiple tools and give automated steps to a tool like Scylla?

Limitations and Challenges

However, we also observed clear limitations in AI's ability to adapt to certain complex security measures, particularly those involving permanent blocking.
And parallelable training and multi-step operations is young.

Future Directions

Moving forward, the integration of more advanced AI algorithms, improved parallel processing, and dynamic environment adaptation will be crucial. These advancements could revolutionize penetration testing and vulnerability assessment, while also informing the development of more robust defense mechanisms.

As we continue to explore the intersection of AI and cybersecurity, it's crucial to maintain a balance between offensive capabilities and defensive measures. This research not only pushes the boundaries of what's possible in cybersecurity testing but also underscores the need for continuous innovation in security practices to stay ahead of potential threats.

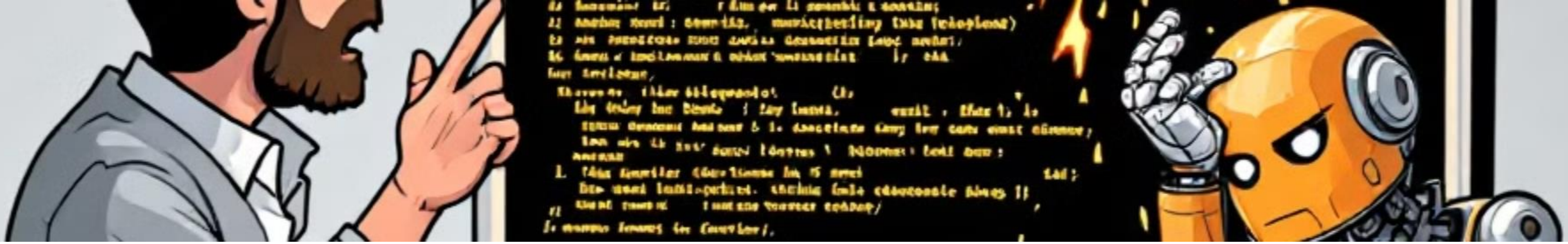
Related work

Most of the research has been done using SQL Injections and XSS

• M. Hadavi and M. Hemmati, “Bypassing Web Application Firewalls Using Deep Reinforcement Learning”.
• J. Wang, L. Qixu, W. Di, Y. Dong, and X. Cui, “Crafting Adversarial Example to Bypass Flow-&ML- based Botnet Detector via RL,” in 24th International Symposium on Research in Attacks, Intrusions and Defenses, San Sebastian Spain: ACM, Oct. 2021, pp. 193–204. doi: 10.1145/3471621.3471841.
• X. Wang and H. Hu, “Evading Web Application Firewalls with Reinforcement Learning”.
• M. Hemmati and M. A. Hadavi, “Using Deep Reinforcement Learning to Evade Web Application Firewalls,” in *2021 18th International ISC Conference on Information Security and Cryptology (ISCISC)*, Sep. 2021, pp. 35–41. doi: 10.1109/ISCISC53448.2021.9720473.

Or related with LLM’s performance to evade defenses or perform attacks... No paralelization

• M. Shao *et al.*, “NYU CTF Dataset: A Scalable Open-Source Benchmark Dataset for Evaluating LLMs in Offensive Security,” Aug. 21, 2024, *arXiv*: arXiv:2406.05590. Accessed: Oct. 27, 2024. [Online]. Available: http://arxiv.org/abs/2406.05590
• Gelei Deng, Yi Liu, V´ıctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. Pentestgpt: An llm-empowered automatic penetration testing tool. arXiv preprint arXiv:2308.06782, 2023
• Matteo Esposito and Francesco Palagiano. Leveraging large language models for preliminary security risk analysis: A mission-critical case study. arXiv preprint arXiv:2403.15756, 2024



Lessons Learned and Bloopers

Parallelization Challenges

Initial attempts at parallel processing led to duplicate attempts and inconsistencies, even using Python memory shareing mechanisms, highlighting the complexity of coordinating multiple AI agents. (The answer is memory sharing and serialization… just realize las week)

Logic Errors

Simple oversights, like failing to increment an index during password spraying, led to unexpected and often humorous results.

Code Evolution

The final codebase became a patchwork of solutions and makeshift fixes, reflecting the iterative nature of AI development in cybersecurity.

Data Loss

Accidental resets of environments resulted in the loss of valuable results, emphasizing the importance of robust data management in AI research.

These experiences underline the challenges and unpredictability inherent in cutting-edge AI cybersecurity research, providing valuable lessons for future projects and highlighting the importance of rigorous testing and error handling.

Customized rules!

Mirantis(1) **for Mod_security**

```
# Initialize a counter for brute force attempts
SecAction "id:100001,phase:1,nolog,pass,initcol:ip=%{REMOTE_ADDR},initcol:global=%{GLOBAL} \
          setvar:ip.brute_force_counter=0,setvar:ip.brute_force_time=%{TX.now}"

# Rule to detect failed login attempt
SecRule REQUEST_URI "@rx ^/(login|auth)" "phase:2,deny,status:403,id:100002,chain"
SecRule RESPONSE_STATUS "@streq 403" "setvar:ip.brute_force_counter=+1,setvar:ip.brute_force_time=%{TX.now}"

# Rule to block IP after 5 failed attempts in 1 seconds
SecRule IP:BRUTE_FORCE_COUNTER "@ge 5" "phase:2,deny,status:403,id:100003,chain"
SecRule TX:NOW "@lt %{IP.BRUTE_FORCE_TIME} + 1"
"setvar:ip.brute_force_counter=0,setvar:ip.brute_force_time=%{TX.now},msg:'Blocking IP for Brute Force Attack',log"
```

(1) https://docs.mirantis.com/mcp/q4-18/mcp-security-best-practices/use-cases/brute-force-prevention/create-brute-force-rules.html

Customized rules!

Ours for Ossec

```
<!-- App login brute force -->
  <rule id="51509" level="3"frequency="5" timeframe="1>
    <if_sid>31108</if_sid>
    <url_pcre2>login</url_pcre2>
    <pcre2>\] "POST \S+login| "POST /loginr</pcre2>
    <description>WebApp login attempt.</description>
  </rule>
```

Ours for Suricata

```
alert http any any -> any any (msg:"Possible HTTP Brute Force Attack"; flow:to_server; content:"POST"; http_method;
content:"password"; http_uri; pcre:"/^(login)$/i"; classtype:web-application-attack; sid:100001; rev:1; threshold:type both, track
by_src, count 5, seconds 1;)
```