Neurosymbolic Autonomous Agents for Cyber-Defense

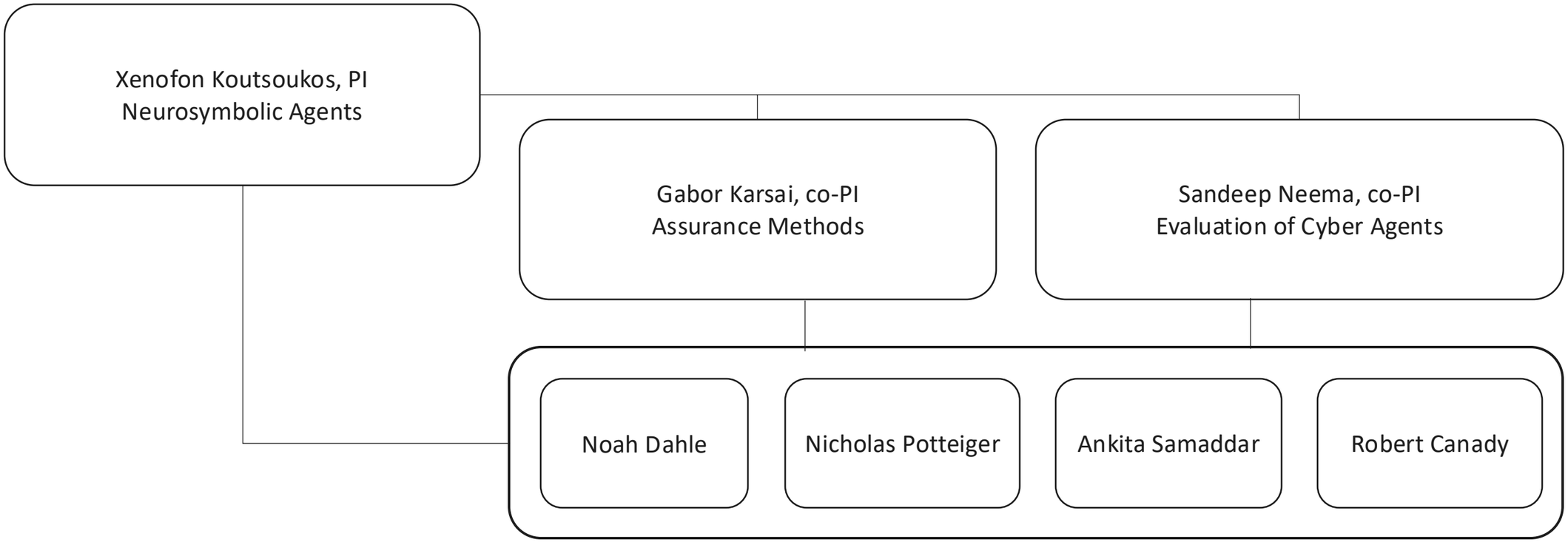
Xenofon Koutsoukos

Department of Computer Science

Institute for Software Integrated Systems

Vanderbilt University

Team


Xenofon Koutsoukos, PI
Neurosymbolic Agents
Gabor Karsai, co-PI
Assurance Methods
Sandeep Neema, co-PI
Evaluation of Cyber Agents
Noah Dahle
Nicholas Potteiger
Ankita Samaddar
Robert Canady

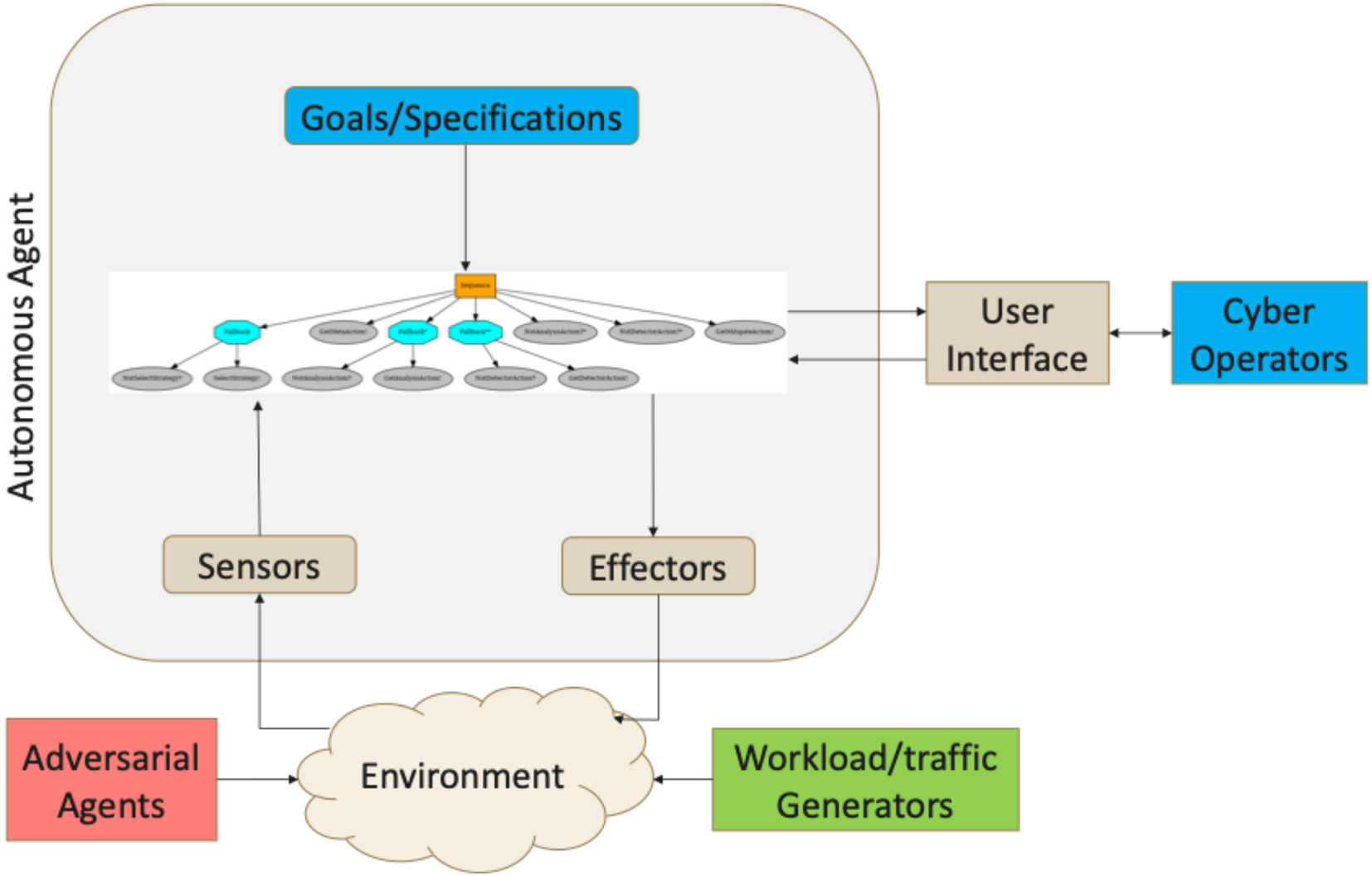
Project Vision and Research Challenges

- **Technical Rationale**
  - Autonomous agents for cyber applications need to learn, reason about, and adapt to deploy security mechanisms for defending networked computer systems while maintaining critical operational workflows.
- **Research Challenges**
  - Cyber agents need to complete **multiple interdependent tasks** over variable length time-intervals.
    - Many tasks can be realized using learning-enabled components (LECs) to handle and uncertainty and variability of the environment.
  - **Autonomous cyber agents** must continuously explore, improve tasks already learned, learn new tasks, and identify creative ways to synthesize goals, plans, and tasks to increase effectiveness.
  - **Robustness and generalizability** in new cyber environments is necessary to address novel and fast changing threats.
  - **Assurance methods** must provide evidence for the correctness of the agents.
  - **Interpretability** can improve human trust and human-machine teaming.
  - **Demonstration and evaluation** using a cyber operational environment which is scalable and fast enough to be used in RL training.

Overview

- Designing robust cyber-defense agents with evolving behavior trees
- Out-of-distribution detection for neurosymbolic autonomous cyber agents
- Demonstration and evaluation using emulation
- Designing cyber agents using LLMs (in progress)
- Multi-agent cyber defense (in progress)
  - CAGE Challenge 3 and 4
- Conclusions

Neurosymbolic Autonomous Agents



N. Potteiger, A. Samaddar, H. Bergstrom and X. Koutsoukos, "Designing Robust Cyber-Defense Agents with Evolving Behavior Trees," *International Conference on Assured Autonomy (ICAA)*, Nashville, TN, USA, Oct. 10-11, 2024.

Assurance Challenges

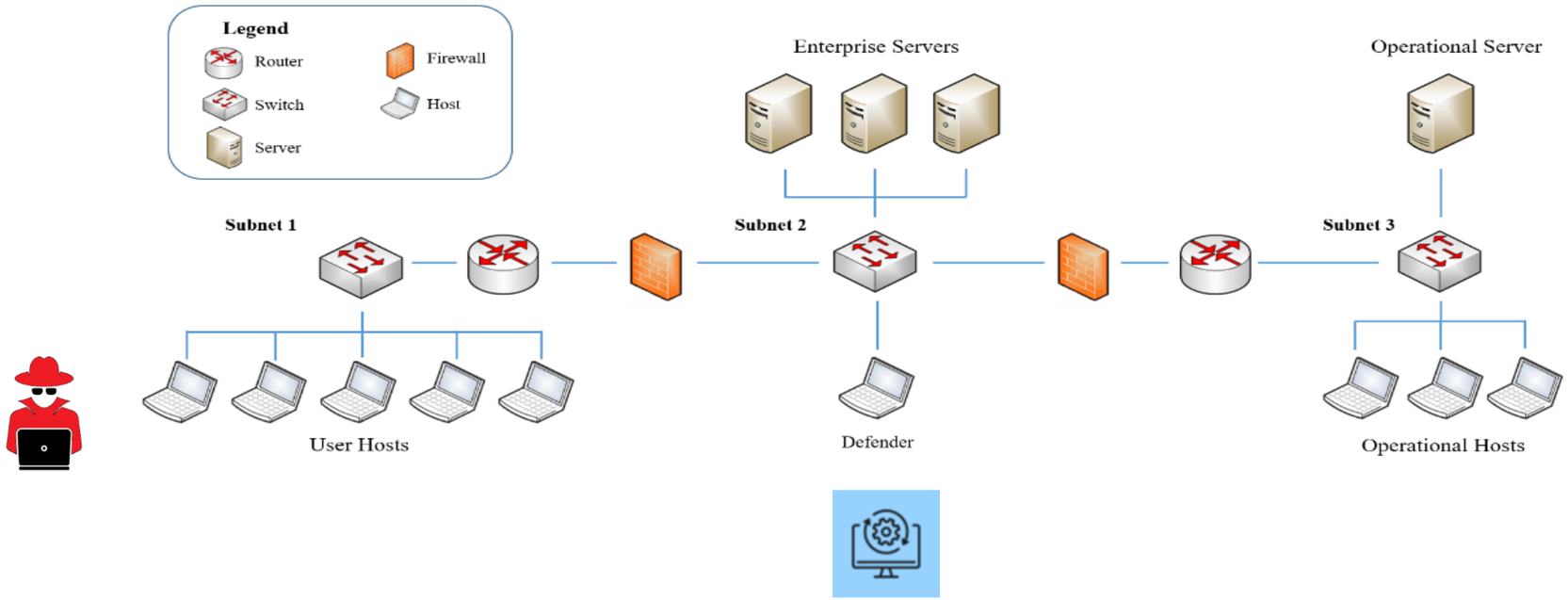
Uncertainty due to ***limited knowledge*** about the ***runtime behavior*** of the operational system and environment during training of the autonomous agents

Robustness and generalizability of the autonomous agents

Consequences can propagate deep into the system and ***impact*** system behaviors at all levels

Out-of-distribution detection (OOD) can be used to identify data that is nonconformal with the training distribution.

Problem Statement

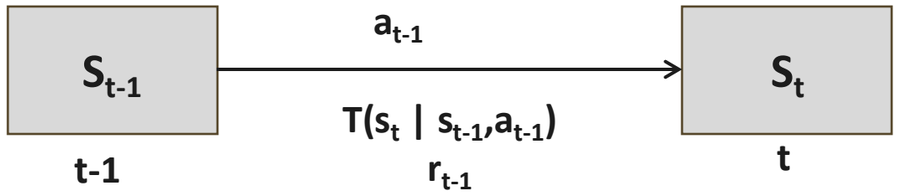


Given a network consisting of hosts, enterprise servers and operational servers and a neurosymbolic cyber-agent trained with a policy π*, our objective is to develop a* ***runtime monitoring algorithm*** *to detect shifts from the training distribution.*

A. Samaddar, N. Potteiger, and X. Koutsoukos. "Out-of-Distribution Detection for Neurosymbolic Autonomous Cyber Agents." *4th IEEE International Conference on AI in Cybersecurity (ICAIC).* Houston, TX, USA,, Feb. 5-7, 2025.

System Model

- The system can be represented by a **discrete-time Partially Observable Markov Decision Process (POMDP) M = (S, A, T, R, μ0)**
  - **S** : set of discrete and partially observable states
  - **A** : set of defender (blue agent) discrete actions
  - **T** : conditional transition probabilities
  - **R: S x A x S → R** : Reward function
  - **μ0**: initial state and action



- **Blue agent objective:**
- Select actions at each timestep to maximize the cumulative reward:

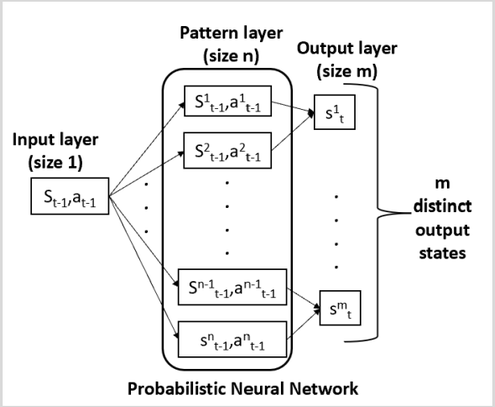
$$\sum_{t=1}^{t=\infty} r_{t\_1}$$

Out-of-Distribution (OOD) Detection

Data Generation → **Training** → **OOD Detection**

- Collect transitions $(s_{t-1}, a_{t-1}) \rightarrow s_t$ for τ timesteps over **N** episodes to generate the training data D_{train}

- Develop a **Probabilistic Neural Network (PNN)** the transition $(s_{t-1}, a_{t-1}) \rightarrow s_t$ probabilities for policy π over D_{train}



- s_t : State at timestep t given action a_{t-1} on state s_{t-1}
- $\{s^1_t, s^2_t, \ldots s^k_t\}$: set of **k** predicted current states from PNN
- If $s_t \in \{s^1_t, s^2_t, \ldots s^k_t\}$ and $Pr((s_{t-1}, a_{t-1}) \rightarrow s_t) > \rho$, then in-distribution
- Else s_t is out-of-distribution

Probabilistic Neural Network (PNN)

Red Agent

- **Meander Agent**
- **B-line Agent**

Blue Agent

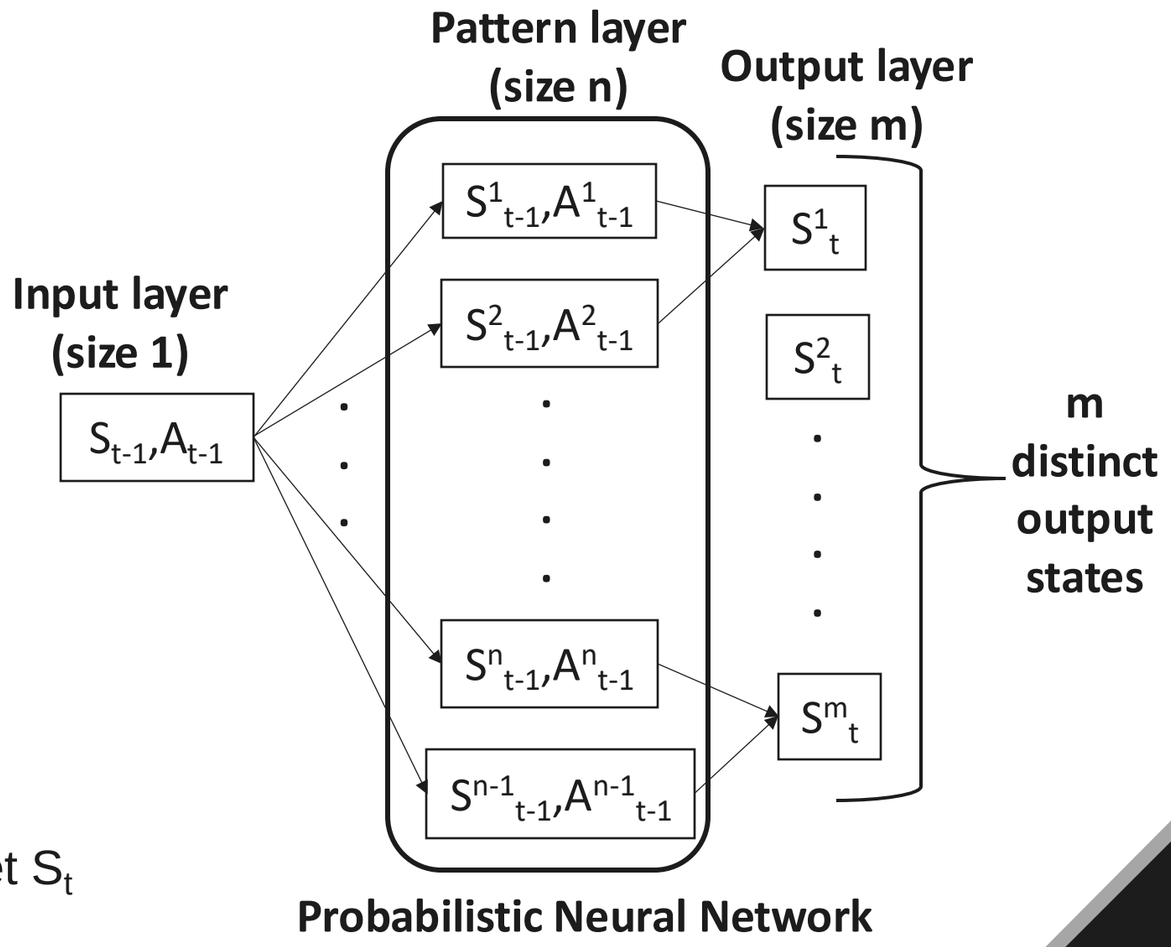
- **Evolving Behavior Tree (EBT)**

Training

- Given a Red and a Blue Agent, construct a PNN with the training data from CybORG.

Testing

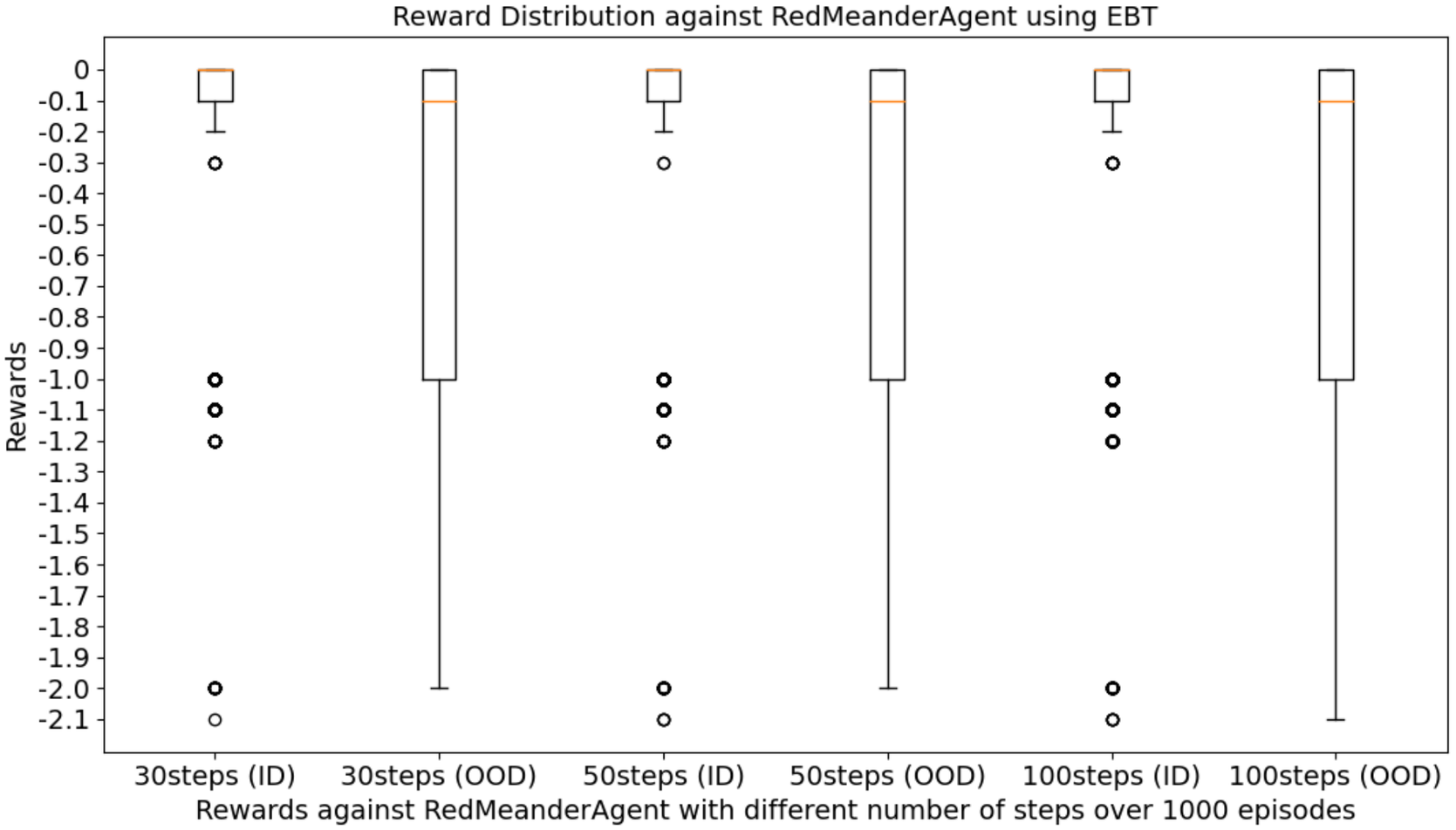
- For a given S_{t-1} , A_{t-1}, run CybORG simulator to get S_t
- Predict the set of current states from the PNN.



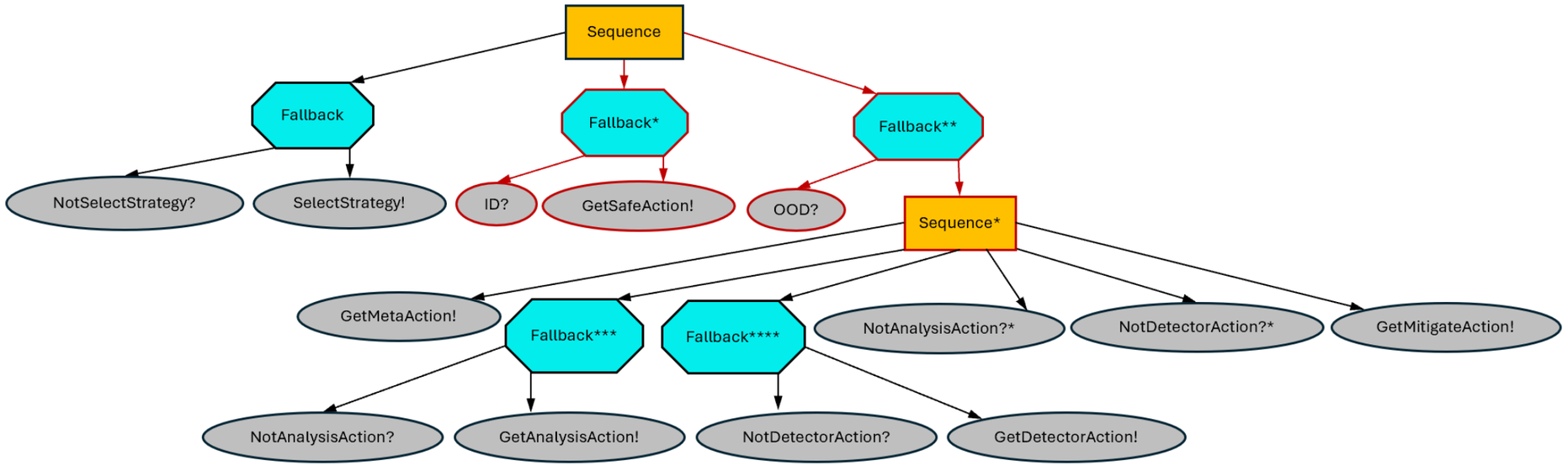
Simulation Results

Red Agent	Blue Agent	PNN	Number of steps in the Test data	Number of OOD episodes (out of 1000)
B-line Agent	EBT	Trained with data over 1000 episodes with 100 steps	30	16
			50	12
			100	13
Meander Agent	EBT	Trained with data over 1000 episodes with 100 steps	30	65
			50	95
			100	128

Out of Distribution Generalization



Integration of OOD Detection in EBT

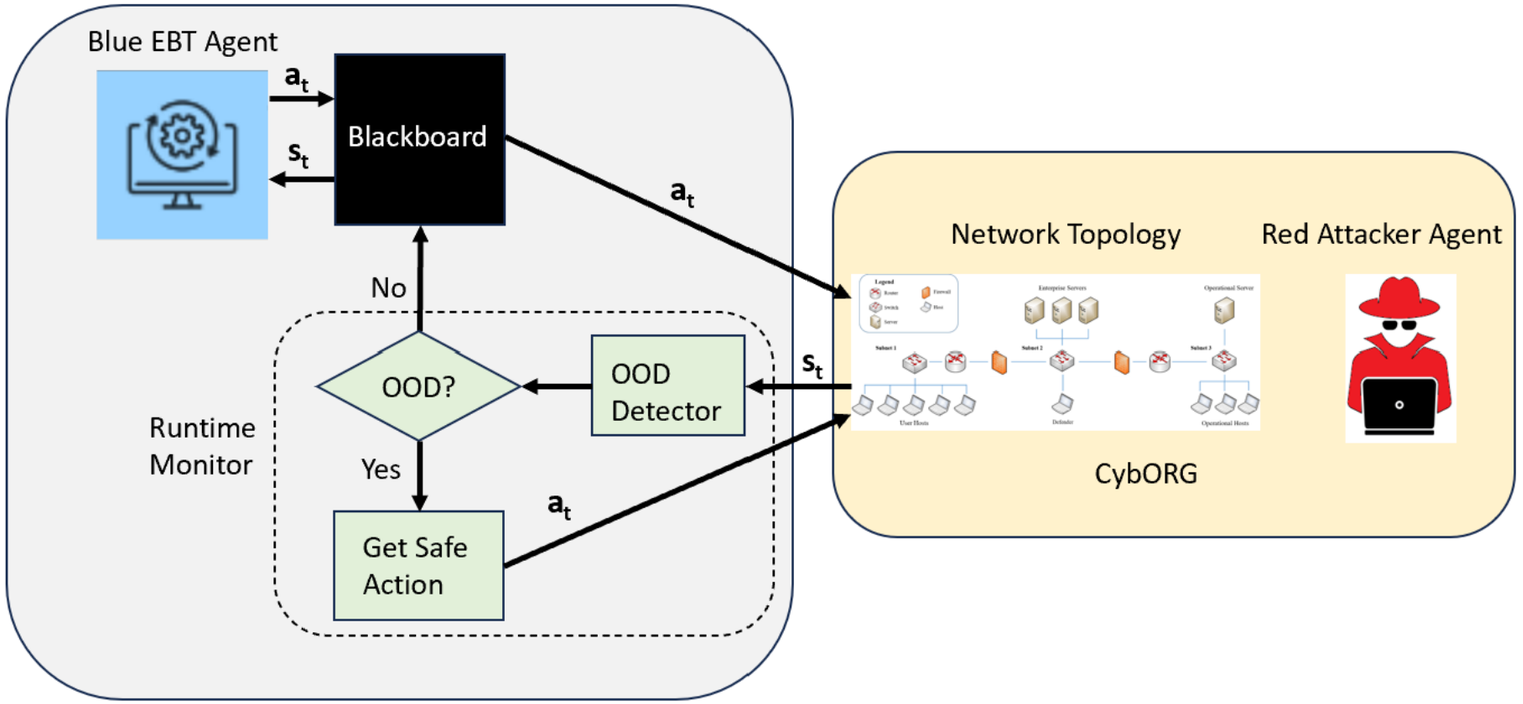


1. **ID? :** Determines if current state s_t is In-Distribution

2. **GetSafeAction! :** Executes *Restore* action to restore the affected host/server to a previously known *"safe"* state, to assure safety

3. **OOD? :** Returns Failure if current state s_t is In-Distribution to ensure normal execution of the system

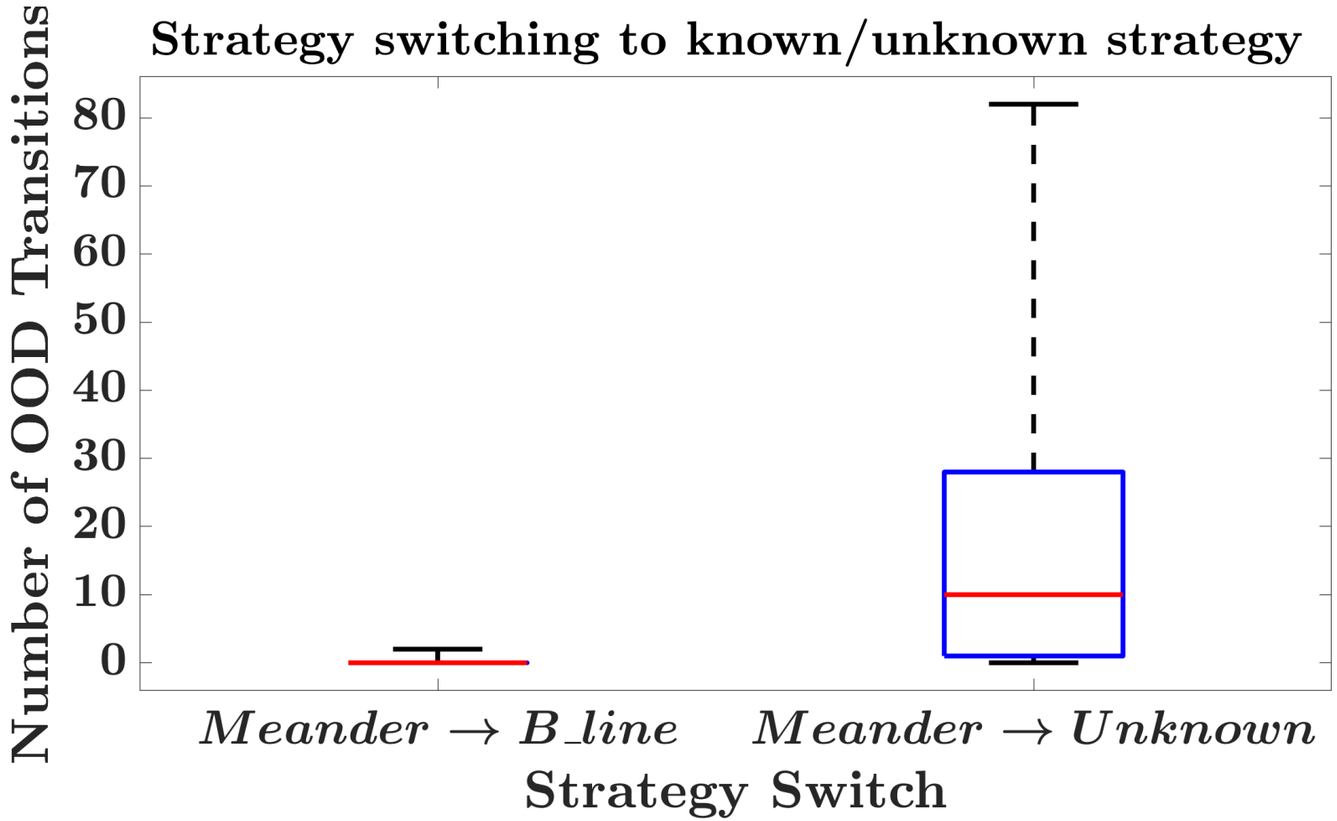
Experimental Setup



- CybORG CAGE Challenge 2
- Blackboard: Communication interface between the EBT and the simulator
- Experiments with two red agent strategies: *Meander* and *B_line*
- Generate D_{train} for each of these agents over 10,000 episodes each with 100 steps to train the PNN

14

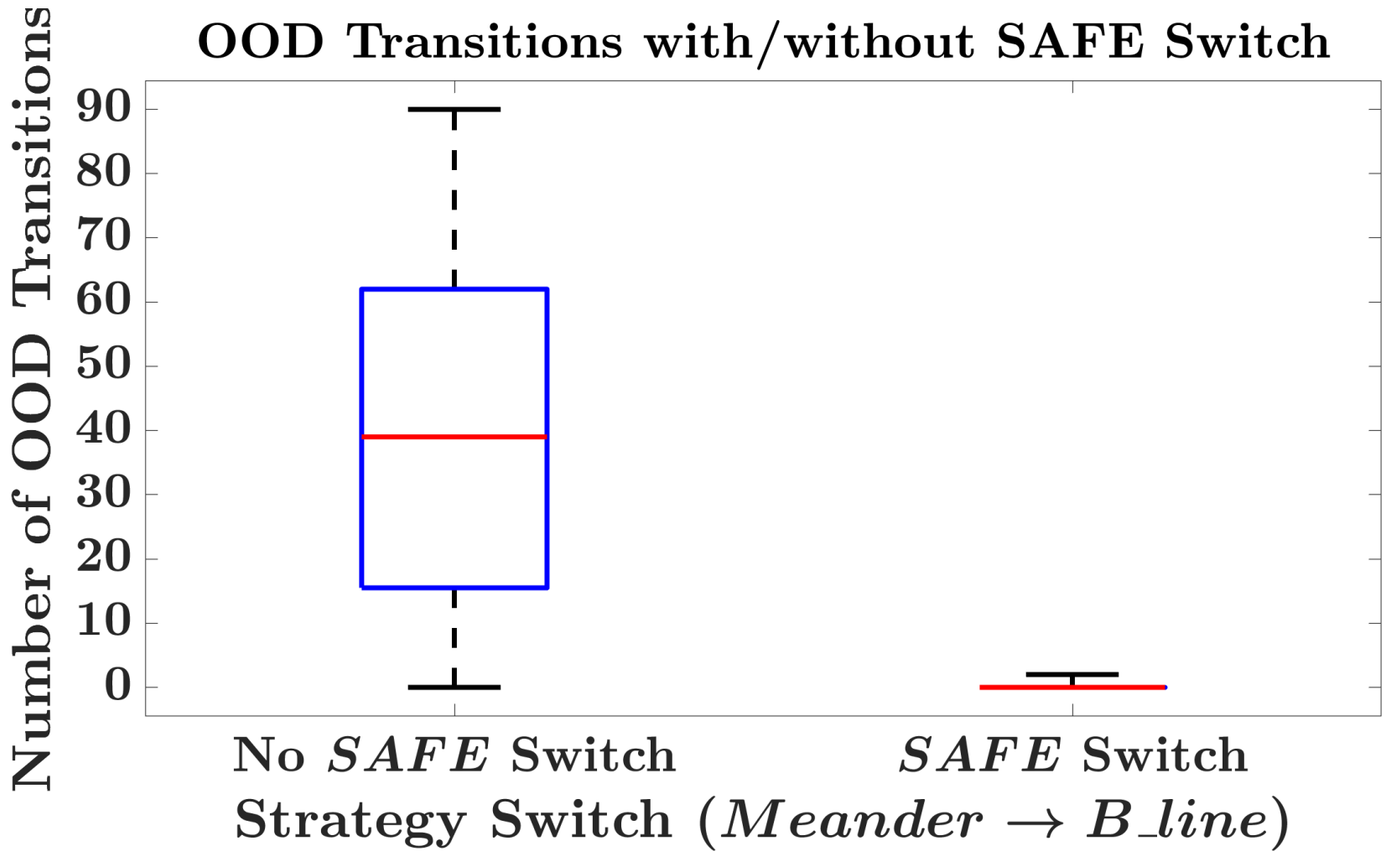
Results



Number of OOD transitions when the red agent switches to an unknown strategy is significantly high as the blue agent has no knowledge about the strategy.

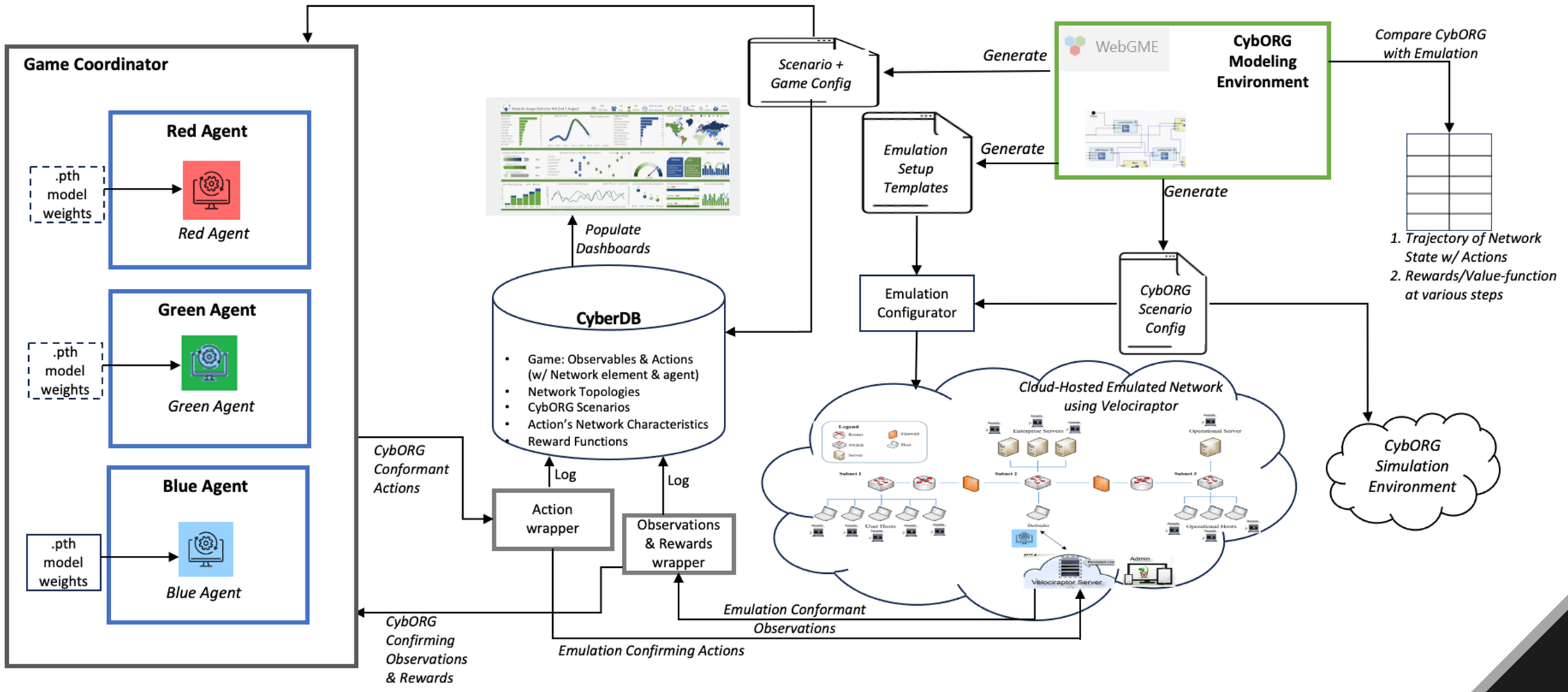
15

Results

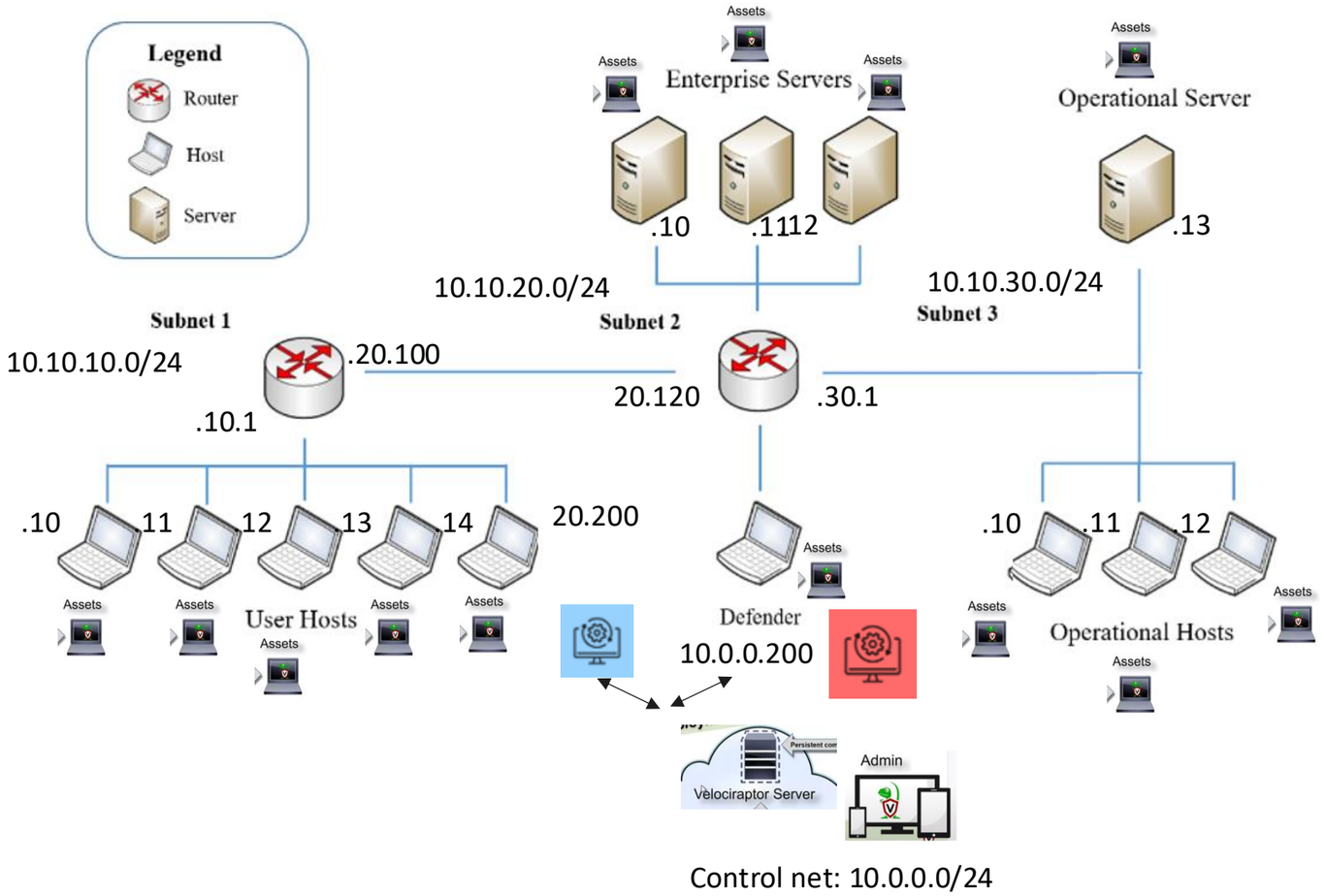


GetSafeAction! behavior in the EBT significantly reduces the number of OOD transitions by restoring the system to a "*safe*" state

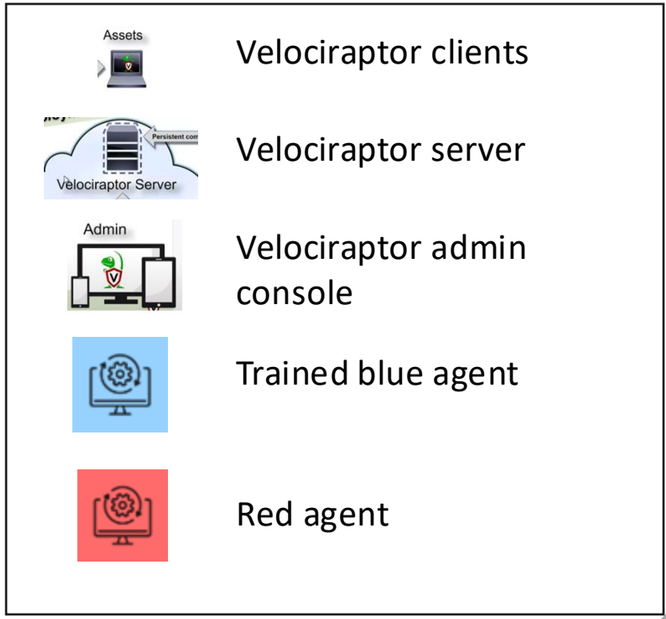
Emulation Testbed (DARPA CASTLE)



Emulation Testbed Architecture



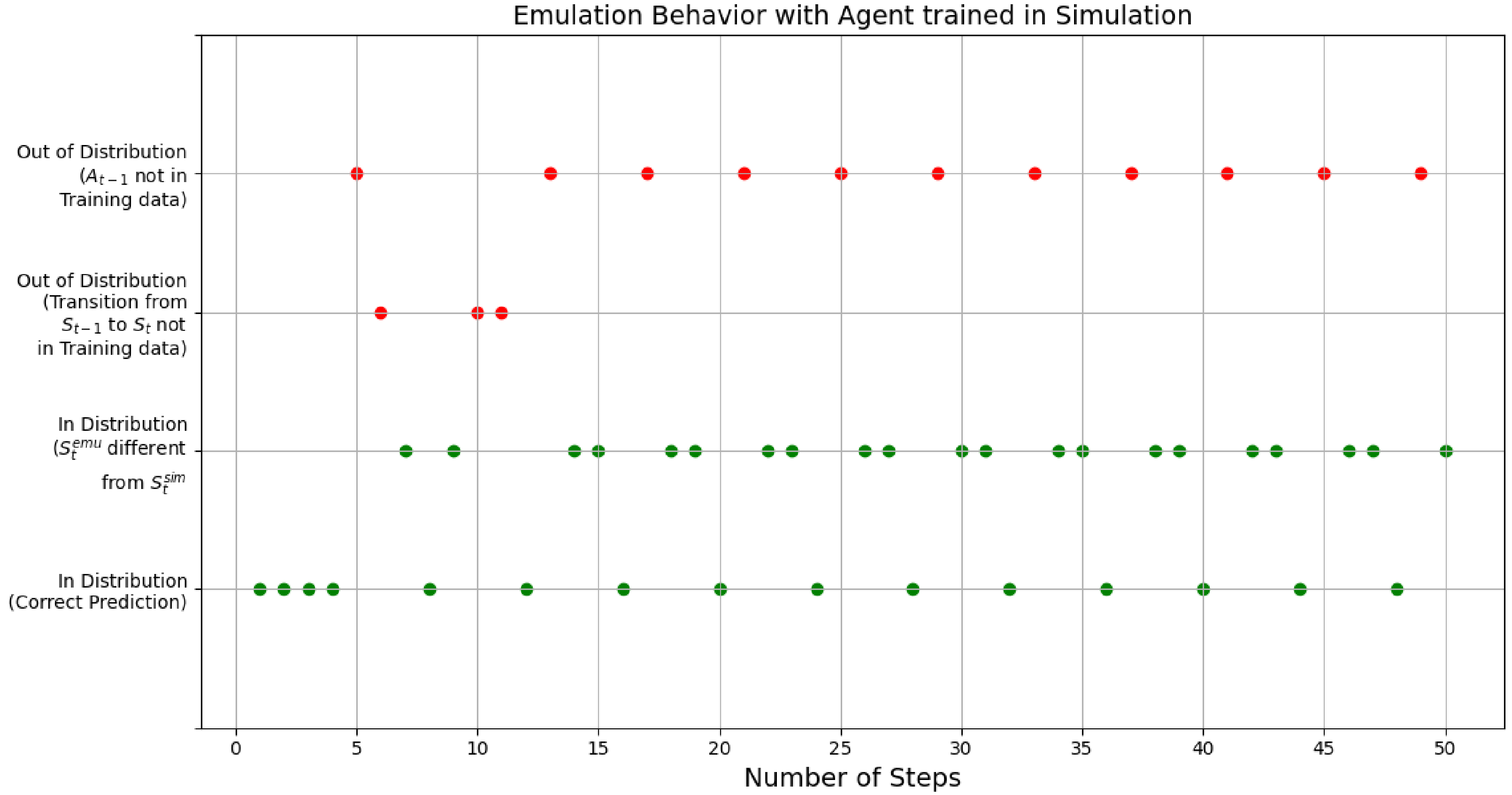
Key



Red agent: Executes an action based on its policy.
Goal: Reach Operational server and execute Impact action.

Blue agent: Selects action -> Calls Velociraptor Server -> Makes RPC calls to execute the action on the appropriate host.
Goal: Prevent the red agent from penetrating deep into the network.

Simulation vs Emulation Results



Example. In Emulator,

S_4 : [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0],

A_4 : Analyze User 2,

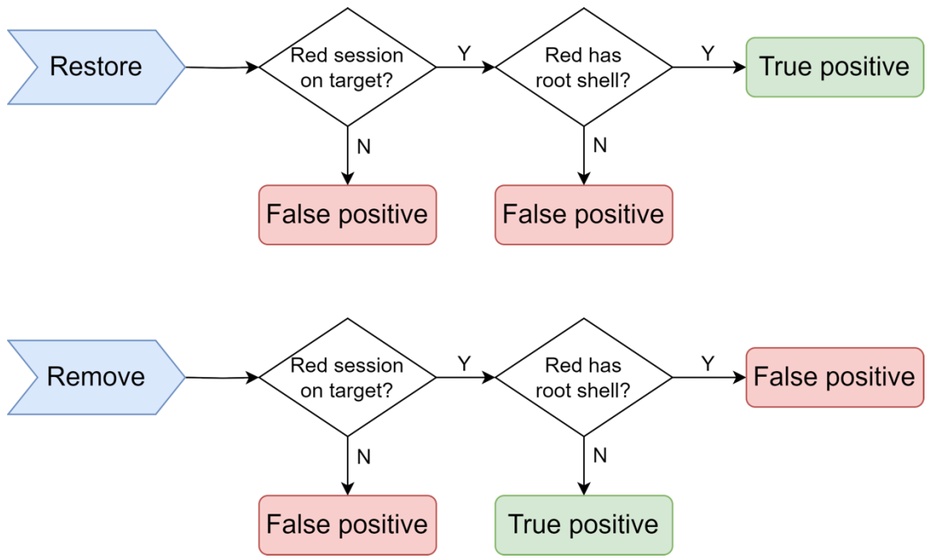
S_5 : [0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]

Transition **(S_4, Analyze User2) → S_5** , not in Training data

OOD results on **emulator** over **one episode with 50 steps** using PNN trained in **simulation** against **BlineAgent as Red Agent** and **EBT as Blue Agent** over **1000 episodes with 100 steps.**

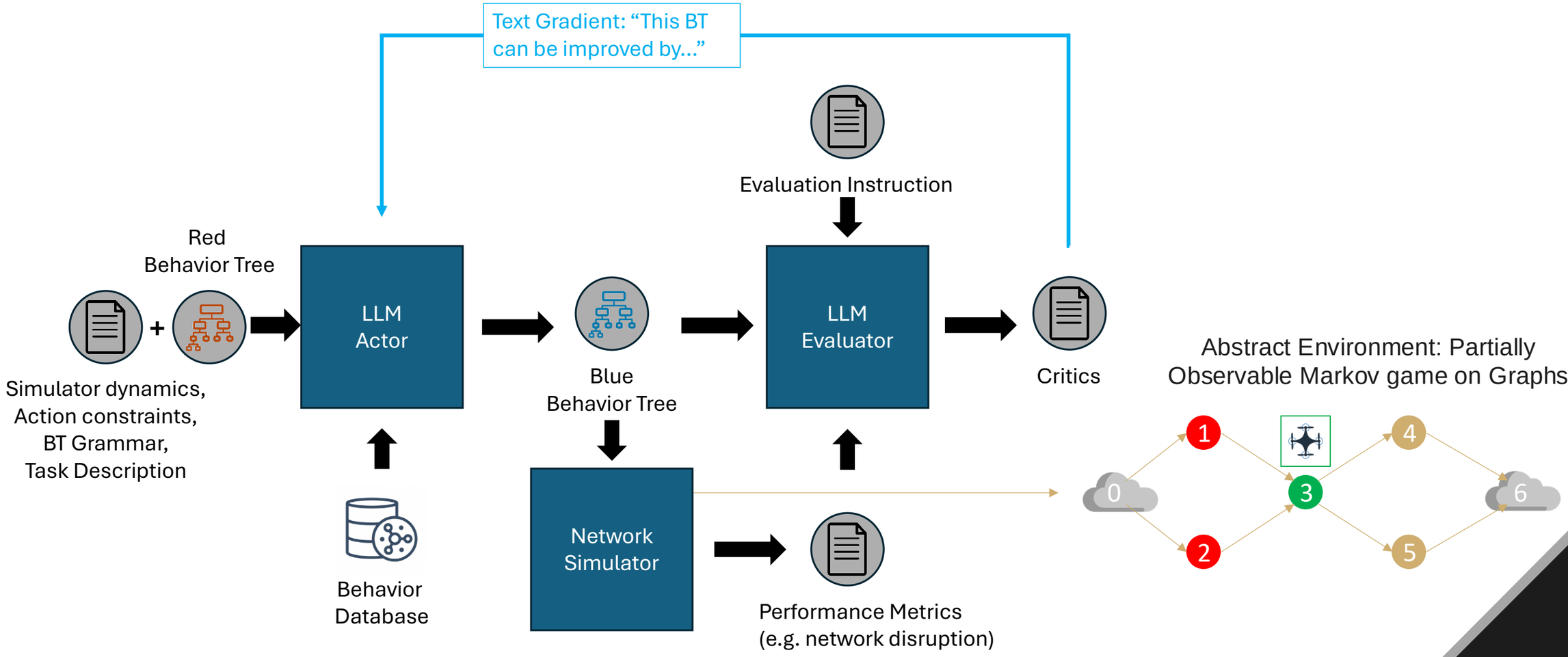
Precision and Recall (in progress)

- **Compromised host** is defined as:
  - Host that has one or more red sessions present.
  - Any type of red session, either user or root.
- **Recovered host** is defined as:
  - Host that does not have any red session at the current time step t, after previously being compromised at time step t-1.
  - The most recent blue action was a remove or restore.
- **Precision:** Number of steps where blue agent executes Restore/(Remove) operation with the red agent having/(not having) access to the root shell to the total number of steps where blue agent executes Restore/(Remove) operation.
  - Precision = TP/(TP + FP)
- **Recall:** Number of correctly recovered hosts out of all compromised hosts.
  - Recall = TP/(TP + FN)



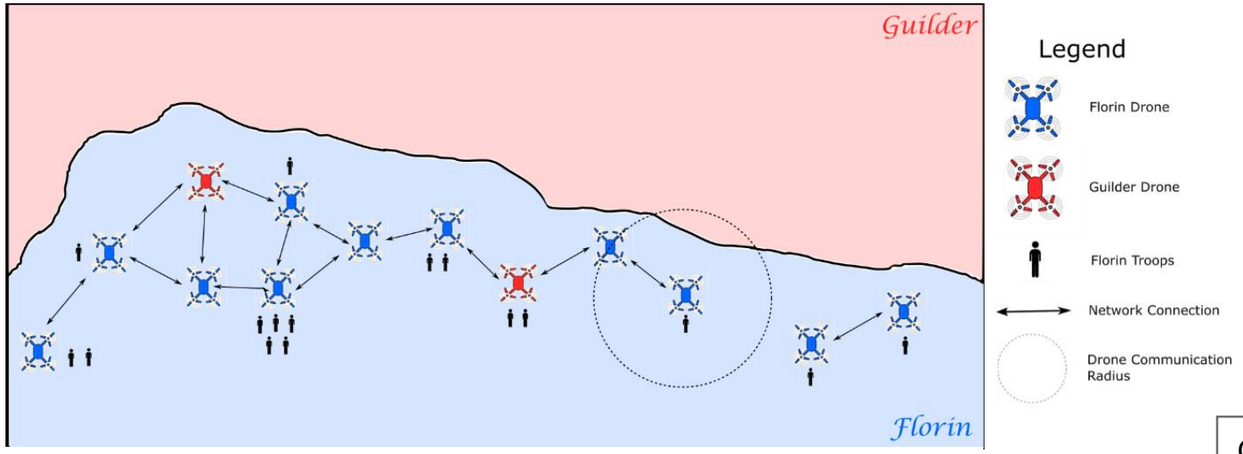
	Emulator (over 4 episodes)	Simulator (over 4 episodes)
Recall	Mean: 0.9198 Stdev: 0.0037	Mean: 0.58 Stdev: 0.113

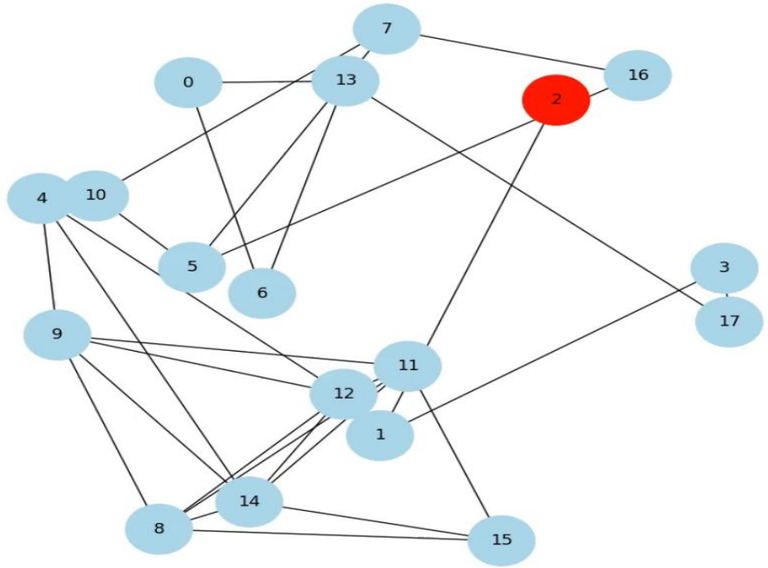
Using LLMs for Designing Behavior Trees

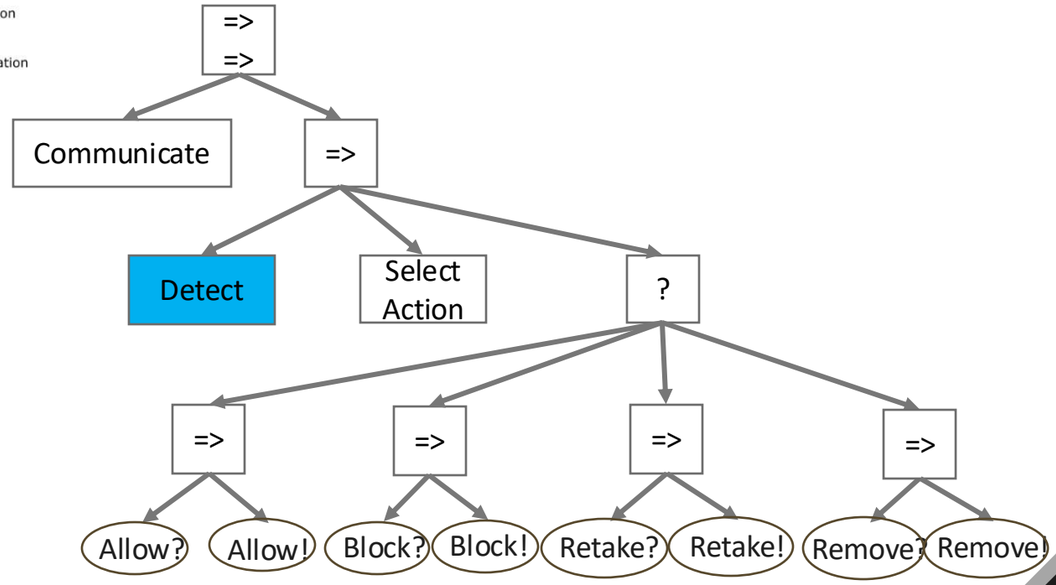


Feedback loop to optimize performance of a blue behavior tree.

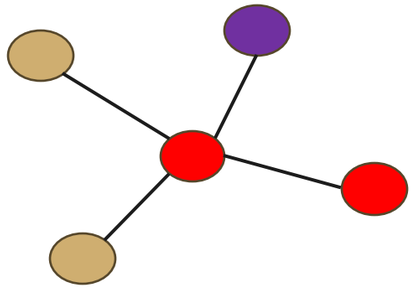
CAGE Challenge 3





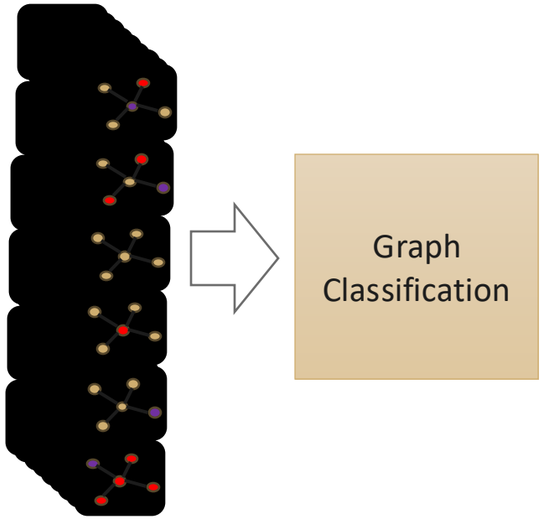


Detection of Compromised Nodes using Graph ML

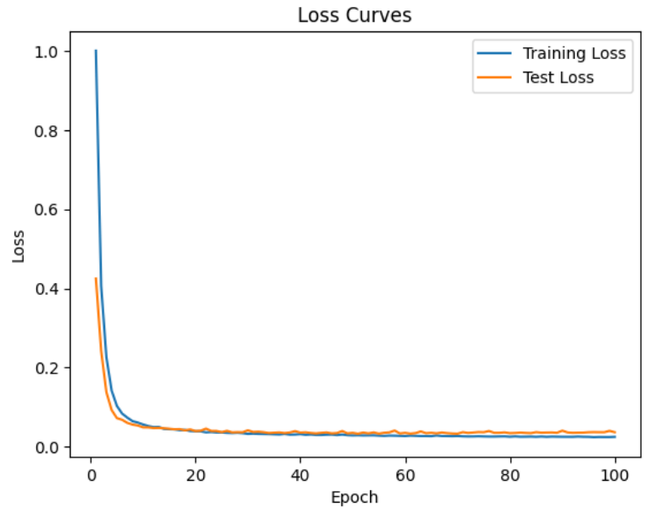


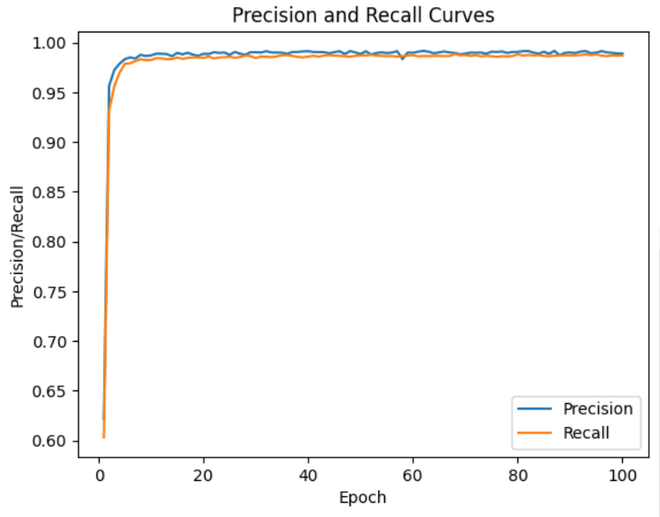
Ego Graphs

- PettingZoo Wrapper
  1. Previous action successful for blue agent
  2. Drones that have been blocked
  3. Malicious session found on host drone
  4. Number of malicious events from drones its connected to
  5. Position of host drone
  6. Drone ID
  7. Position of drones its connected to
  8. If a new session has been added to host drone



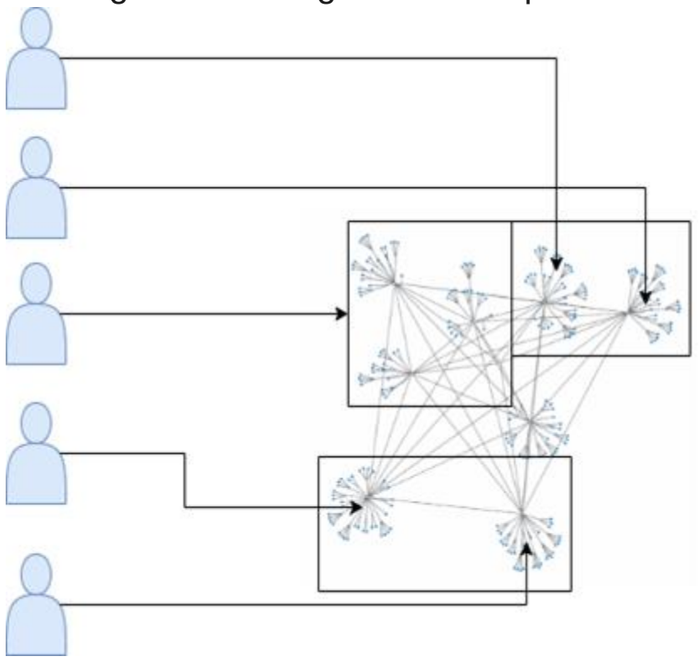
- Collected 55000 graphs
  - 18 Drones
  - Simulations with 200 time-steps
- Graph Classification
  - Graph Convolutional Network (PyG)

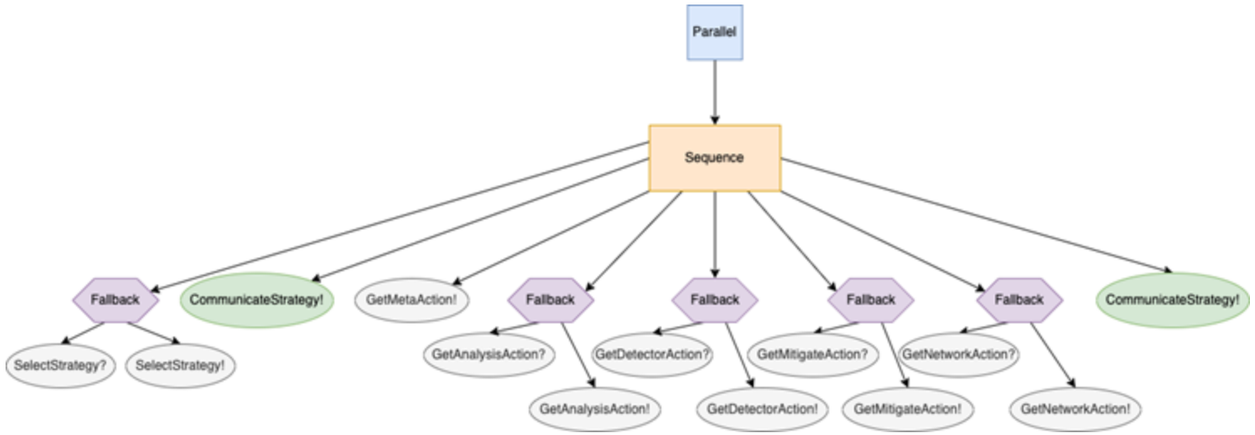


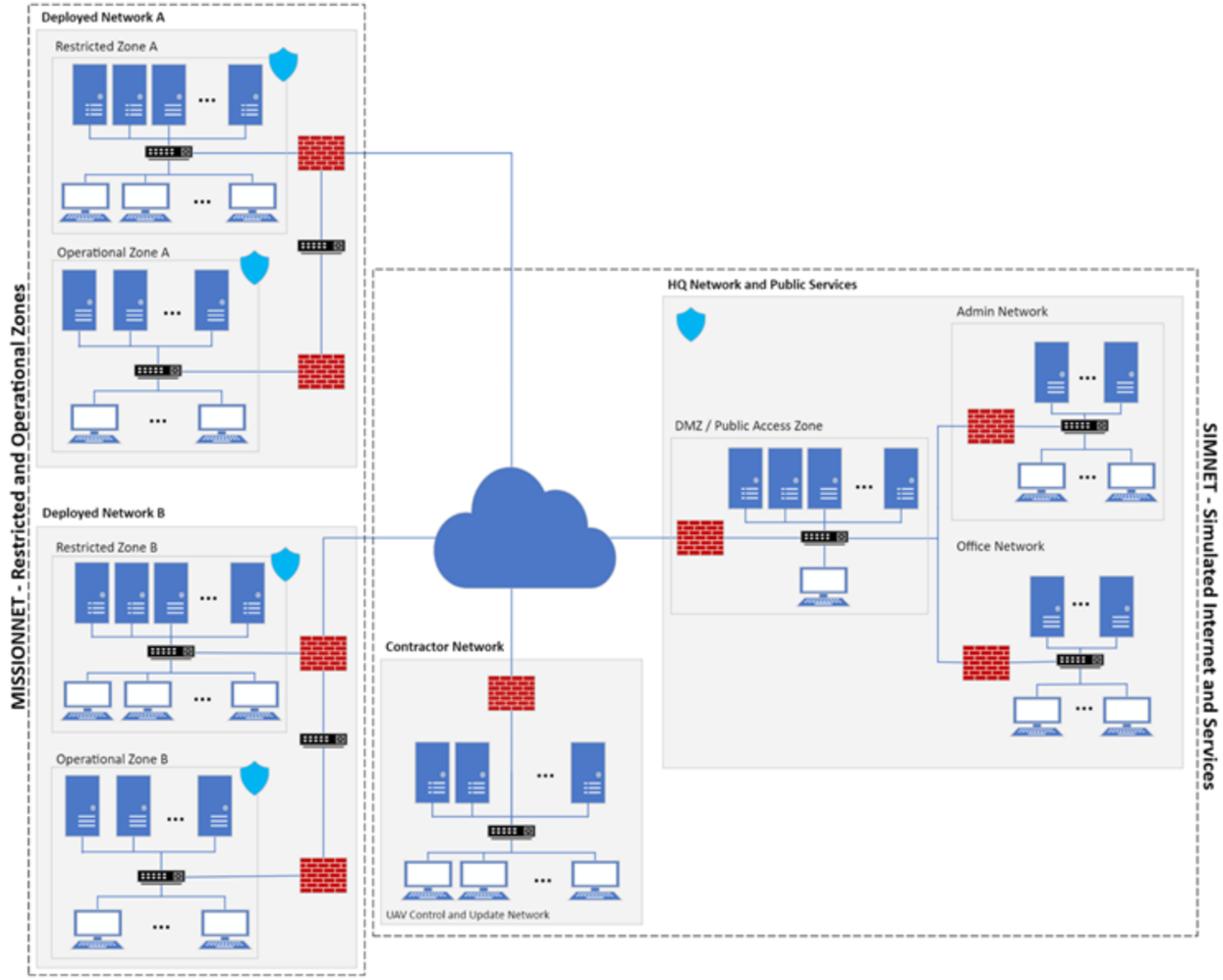


CAGE Challenge 4

Abstract Environment: Partially Observable
Multi-Agent Markov game on Graphs







Conclusions

- Neurosymbolic autonomous agents for cyber defense based on evolving behavior trees
  - Symbolic components captured by the structure of the behavior tree.
  - Neural components are used to realize the various behaviors.
- Out-of-distribution detection for neurosymbolic autonomous cyber agents
  - OOD detection for RL agents with discrete states and actions.
  - Integration of OOD Detection into neurosymbolic agents.
  - Demonstration and evaluation using CAGE Challenge 2 based on CybORG simulation and DARPA CASTLE emulation.
  - Improved robustness and generalizibilty of cyber defense agents.
- Current and future work
  - EBT design using LLMs.
  - Multi-agent systems: CAGE Challenge 3 and 4.

- N. Potteiger, A. Samaddar, H. Bergstrom and X. Koutsoukos, "Designing Robust Cyber-Defense Agents with Evolving Behavior Trees," *International Conference on Assured Autonomy (ICAA*), Nashville, TN, USA, Oct. 10-11, 2024.
- A. Samaddar, N. Potteiger, and X. Koutsoukos. "Out-of-Distribution Detection for Neurosymbolic Autonomous Cyber Agents." *4th IEEE International Conference on AI in Cybersecurity (ICAIC).* Houston, TX, USA,, Feb. 5-7, 2025.