Cyber Red Team Simulation: Attack Graph Pathfinding and Adversarial Machine Learning

A High-Fidelity Network Attack and Defense Simulation Engine with 45+ MITRE ATT&CK Techniques

STS Gym Research

Crab Meat Research Lab

April 2026

doi:10.5281/cyber-redteam-sim.0.2.0 · v0.2.0

ABSTRACT

We present **cyber-redteam-sim**, a high-fidelity network attack and defense simulation engine that models over 45 MITRE ATT&CK techniques across the full cyber kill chain. The engine provides a discrete-event simulation environment where red team operators execute realistic attack sequences against modeled enterprise networks, while blue team defenders detect, correlate, and respond to adversarial activity based on configurable maturity levels. We introduce three key innovations: (1) an

attack graph pathfinding system using A* search with state-dependent pruning to identify optimal compromise paths through network topologies, producing risk-scored paths that balance success probability against stealth; (2) a cumulative stealth model where detection probability grows with successive actions and decays over time, creating realistic operational tempo tradeoffs; and (3) adversarial machine learning agents using tabular Q-learning and REINFORCE policy gradients that learn attack strategies through simulated episodes. The simulation includes probabilistic cloud attack modeling, C2 infrastructure simulation with redirectors and dead drops, IEEE 1278.1 DIS federation for multi-exercise interoperability, and comprehensive after-action review with automated scoring. We demonstrate that the engine can simulate complex multi-stage attack campaigns against enterprise Active Directory environments, producing actionable detection gap analysis and remediation recommendations.

Keywords: cyber simulation, red team, attack graph, A* pathfinding, adversarial machine learning, MITRE ATT&CK, Q-learning, REINFORCE, network defense, purple team, after-action review

Table of Contents

---

1. Introduction

Modern enterprise networks face sophisticated, multi-stage cyber attacks that traverse the full kill chain from initial reconnaissance through data exfiltration. Traditional penetration testing and red team exercises, while valuable, are limited by cost, scope, and the difficulty of safely testing defensive capabilities against realistic adversarial behavior. Organizations need the ability to model attack paths, evaluate detection coverage, and train both offensive and defensive teams in a safe, repeatable environment.

cyber-redteam-sim addresses this gap by providing a high-fidelity simulation engine that models the complete adversarial lifecycle. Unlike network vulnerability scanners that identify weaknesses in isolation, our engine simulates attack campaigns—sequences of techniques that build upon each other, where successful exploitation creates new opportunities for lateral movement and privilege escalation. The engine models 45+ techniques

mapped to the MITRE ATT&CK framework [1], spanning 11 tactics from reconnaissance through command and control.

The key contributions of this work are:

1. **State-dependent attack graph pathfinding.** We model attack graphs as (host, privilege) node pairs connected by technique-labeled edges with probabilistic weights. An A*-based search algorithm finds optimal paths from initial access to domain admin, pruning infeasible edges based on prerequisite state satisfaction. Paths are scored by risk (success probability × stealth), enabling tradeoff analysis between speed and detectability.
2. **Cumulative stealth with temporal decay.** Rather than treating each action's detection probability independently, we introduce a stealth model where the cumulative noise of successive operations increases detection risk. The model incorporates technique-specific noise profiles (EDR bypass capability, log footprint, network visibility, disk artifacts) and temporal decay that allows operators to reduce their exposure by pausing activity—creating realistic operational tempo decisions.
3. **Adversarial ML agents.** We implement both tabular Q-learning and REINFORCE policy gradient agents that learn attack strategies through episodic simulation. The state space is encoded as a feature vector of host properties and global simulation state; the action space covers all technique×host combinations. Reward functions balance objective achievement, stealth, and detection avoidance, producing agents that discover non-obvious attack paths.
4. **End-to-end simulation fidelity.** The engine models protocol-level behavior (TCP state machines, SMB negotiation, Kerberos AS/TGS exchanges, LDAP queries), probabilistic outcomes based on patch levels and hardening, defender maturity levels that scale detection and response capabilities, and C2 infrastructure with redirectors, implants, and dead drops.
5. **Multi-exercise federation via DIS.** An IEEE 1278.1 Distributed Interactive Simulation (DIS) bridge enables interoperability with other simulation

platforms, mapping cyber events to DIS PDU types (Fire, Detonation, Signal, Event Report) for entity state, attack, detection, and C2 events.

The remainder of this paper describes the architecture (Section 2), attack modeling (Section 3), network topology (Section 4), defense modeling (Section 5), attack graph pathfinding (Section 6), ML agents (Section 7), stealth mechanics (Section 8), cloud attack modeling (Section 9), C2 infrastructure (Section 10), purple team operations (Section 11), after-action review (Section 12), federation (Section 13), and performance (Section 14).

2. Architecture

The simulation engine is implemented as a Go module organized into 18 packages, producing a single ~2.8 MB static binary. This monolithic design prioritizes deployment simplicity and deterministic reproducibility while maintaining clear internal package boundaries.

2.1 Simulation Loop

The engine operates as a tick-based discrete event simulation. Each tick represents a configurable time interval (default: 1 second). The core loop executes the following phases in order:

1. **Red team planning:** Generate candidate actions for the current state (reconnaissance, exploitation, lateral movement, privilege escalation, credential access, persistence, C2, collection, exfiltration).
2. **Action execution:** For each planned action, determine success probability based on host properties (patch level, hardening, EDR level, vulnerability exploitability) and roll for success.
3. **State update:** Update attack graph state, credential pools, compromised host registry, and C2 channels based on action results.
4. **Log generation:** Each action emits structured log events (auth, network, process, file) that feed into the detection pipeline.
5. **Blue team detection:** SIEM rules, EDR monitors, and correlation engines evaluate the generated events and produce alerts.
6. **Blue team response:** Incident response engine processes critical/high alerts, advancing cases through detect → analyze → contain → eradicate phases with realistic time delays.
7. **C2 beaconing:** Every 5 ticks, active C2 channels beacon, with detection probability determined by defender maturity and channel stealth.
8. **Objective check:** Determine if the red team objective (e.g., domain admin, data exfiltration) has been achieved.

2.2 Configuration and Scenarios

Scenarios are defined in YAML files specifying network topology, host configurations, security controls, and red team objectives. The engine provides an interactive configuration wizard with four built-in templates:

TEMPLATE	DESCRIPTION	HOSTS	OBJECTIVE
`enterprise-ad`		5+	Domain admin

TEMPLATE	DESCRIPTION	HOSTS	OBJECTIVE
	Corporate AD environment		
`dmz-breach`	Multi-zone DMZ to internal	3+	Data exfiltration
`cloud-kill-chain`	AWS/Azure multi-cloud	4+	Cloud admin
`apt-persistence`	Long-duration APT	5+	Persistence + C2

3. Attack Modeling

The attack package implements 45+ MITRE ATT&CK techniques spanning the full kill chain. Each technique is modeled with probabilistic success rates, detection probabilities, and realistic state transitions.

3.1 Technique Catalog

Techniques are organized by ATT&CK tactic. Each technique in the catalog specifies:

- **Success probability base rate** — the inherent likelihood of success absent defenses
- **Detection probability base rate** — how likely the action is to be detected
- **Prerequisites** — state keys that must be achieved before the technique can be attempted
- **Cost** — effort/time cost on a 0–1 scale
- **Noise level** — how much the technique alerts defenders (0–1)

For example, technique T1190 (Exploit Public-Facing Application) has a base success probability of 0.6, detection base rate of 0.2, no prerequisites, cost of 0.3, and noise level of 0.4. By contrast, T1558.001 (Kerberoasting) has success

0.75, detection 0.35, requires `ad-access` and `ldap-query` prerequisites, cost 0.15, and noise 0.3—reflecting its higher sophistication and lower visibility.

3.2 Kill Chain Progression

The engine's red team planner follows a 12-phase progression that mirrors the ATT&CK kill chain:

1. **Reconnaissance** — Network scanning of DMZ hosts, then progressive scanning of hosts reachable from compromised positions
2. **Exploitation** — Exploiting discovered vulnerabilities on known hosts, with success rates modulated by patch level and hardening
3. **Privilege escalation** — User→admin and admin→domain escalation, with up to 3 retry attempts per host
4. **Credential dumping** — Harvesting credentials from admin-level hosts (mimikatz-style)
5. **Credential reuse** — Pass-the-hash, pass-the-ticket, and valid account attacks on reachable hosts
6. **Defense evasion** — Disabling EDR/AV controls on compromised admin hosts
7. **Persistence** — Establishing scheduled tasks, registry keys, or service-level persistence
8. **C2 establishment** — Deploying C2 channels (HTTPS, DNS, HTTP) with beacons and implants
9. **Lateral movement** — PsExec, WMI, SSH, RDP to reachable neighbors with best-method selection
10. **Kerberos attacks** — Kerberoasting and AS-REP roasting when AD access is available
11. **Data collection** — Discovery and staging of data on compromised hosts
12. **Exfiltration** — Data transfer over C2 channels with volume-based detection modeling

3.3 Success Probability Modeling

Success probabilities are not static; they are modulated by target host properties:

$$P_{\text{success}} = P_{\text{base}} \times (1 - \text{patchLevel} \times 0.5) \times (1 - \text{hardeningLevel} \times 0.2)$$

For lateral movement, the formula additionally incorporates credential strength and target hardening:

$$P_{\text{lateral}} = 0.6 \times (1 - \text{hardeningLevel}_{\text{target}} \times 0.3) + \text{credentialBonus}$$

This creates realistic scenarios where a fully patched, hardened host with EDR level 3 is significantly harder to compromise than an unpatched workstation with no endpoint protection.

4. Network Topology Model

The network package models enterprise networks as directed graphs of hosts connected by links with firewall rules. This is the foundation upon which all attack and defense operations are built.

4.1 Host Model

Each host is modeled with rich properties that directly affect attack and defense simulation:

PROPERTY	TYPE	DESCRIPTION
`patch_level`	0–1	Fraction of vulnerabilities patched; reduces exploit success
`hardening_level`	0–1	CIS hardening compliance; reduces all attack success rates
`edr_level`	0–3	Endpoint detection capability (0=none, 3=advanced)

PROPERTY	TYPE	DESCRIPTION
`av_level`	0–2	Anti-virus capability (0=none, 2=heuristic)
`log_level`	0–3	Logging verbosity; affects SIEM detection probability
`zone`	string	Network zone: dmz, internal, management, cloud
`role`	string	Host role: workstation, server, dc, firewall, router
`services`	[]	Running services with port, protocol, version, banner
`vulnerabilities`	[]	CVE entries with CVSS, exploit availability, remediation status

4.2 Network Zones and Reachability

The network is segmented into zones (DMZ, internal, management, cloud) with links that specify bandwidth, latency, and firewall rules between subnets. Reachability is computed on demand:

```
func (n *Network) IsReachable(srcIP, dstIP string, po
```

This function traverses the link graph between source and destination subnets, evaluating firewall rules at each hop. It enables realistic scenarios where lateral movement is constrained by network segmentation, and attackers must find paths through allowed ports and protocols.

4.3 Active Directory and Kerberos

The network model includes Active Directory domains with:

- Domain controllers with Kerberos KDC service (port 88)
- LDAP services (ports 389/636) for directory queries

- Domain trust relationships (one-way, two-way, transitive)
- SID filtering controls
- AD users, groups, OUs, and GPOs
- LAPS (Local Administrator Password Solution) configuration
- Tiered administration model

This enables realistic Kerberoasting, AS-REP roasting, pass-the-ticket, and DCSync attack simulation with protocol-level fidelity.

5. Defense Modeling

The defense package implements a multi-layered detection and response system that scales with organizational maturity. The model is designed to produce realistic detection gaps that mirror real-world blue team limitations.

5.1 Maturity Levels

The engine defines five security maturity levels, each unlocking progressively more capable detection and response:

LEVEL	NAME	CAPABILITIES	MTTD	MTTC
1	Basic	AV only, no SIEM, no IR plan	Days	Weeks
2	Intermediate	SIEM + basic EDR, IR playbook	Hours	Days
3	Advanced	NDR + threat hunting, 24/7 SOC	Minutes	Hours
4	Expert	ML detection, purple team, automated response	Seconds	Minutes
5	Elite			Automated

LEVEL	NAME	CAPABILITIES	MTTD	MTTC
		Full telemetry, proactive defense, zero-trust	Real-time	

5.2 SIEM Correlation Engine

The correlation engine aggregates log events within time windows and triggers alerts when correlation rules fire. It implements threshold-based detection (N events of type X within window Y) and cross-source correlation (auth log + network log + process log matching pattern). Each action in the simulation emits structured log events that are ingested by the correlation engine for pattern matching.

5.3 EDR Monitoring

The EDR monitor implements process-level detection with capability levels that scale with maturity:

- **Level 0 (None):** No endpoint visibility
- **Level 1 (Basic):** Hash-based detection, known IOCs
- **Level 2 (Standard):** Parent-child process correlation, command-line monitoring, behavioral analysis
- **Level 3 (Advanced):** Memory scanning for injection detection, hook detection, ML-based anomaly classification

5.4 Incident Response Engine

The IR engine models the full incident response lifecycle: detect → analyze → contain → eradicate → recover. Cases advance through phases with realistic time delays based on maturity level. Containment actions include host isolation, account disabling, and firewall blocking. The engine tracks case histories, response times, and outcome effectiveness.

6. Attack Graphs and Pathfinding

The attack graph module is the engine's strategic reasoning layer. It generates a directed graph from network topology and computes optimal attack paths using A* search with state-dependent pruning.

6.1 Graph Structure

An attack graph consists of nodes representing (host, privilege-level) pairs and edges representing technique-executable transitions. For example, node `dc01:3` represents domain admin privilege on the domain controller, while `web01:1` represents user-level access on a web server.

Edges are annotated with:

- **Success probability** — P(success | source state, target hardening)
- **Detection probability** — P(detection | technique, EDR level, maturity)
- **Cost** — Time/effort cost (0–1)
- **Impact** — Business impact if detected (0–1)
- **Prerequisites** — State keys that must be achieved before this edge is traversable

6.2 State Keys and Prerequisites

The attack state machine defines 17 state keys that gate technique availability:

STATE KEY	DESCRIPTION	GRANTED BY
`foothold`	Initial access achieved	Any user-level compromise
`admin-access`	Local admin on any host	Privilege escalation to admin
`admin-creds`	Administrator credentials obtained	Credential dumping, Kerberoasting
`ntlm-hash`	NTLM hash available	Credential dumping (T1003)

STATE KEY	DESCRIPTION	GRANTED BY
`kerberos-ticket`	Kerberos TGT/TGS available	Kerberoasting, AS-REP roasting
`ad-access`	Authenticated AD access	Domain credentials or Kerberos tickets
`c2-established`	Active C2 channel	C2 setup (T1071)
`data-collected`	Data staged for exfiltration	Data discovery (T1005)
`domain-admin`	Domain admin achieved	DC compromise + admin escalation

This prerequisite system ensures that the attack planner cannot, for example, execute Kerberoasting without first achieving AD access, or exfiltrate data without collecting it. It creates a realistic dependency chain that mirrors actual adversarial campaigns.

6.3 A* Pathfinding with State-Dependent Pruning

The `FindPaths` function searches for attack paths from an initial state to a target (host, privilege) node. It uses a priority queue ordered by risk score:

$$\text{RiskScore}(\text{path}) = P(\text{path success}) \times (1 - P(\text{path detection}))$$

The algorithm maintains an attack state that tracks achieved prerequisites. At each expansion step, edges whose prerequisites are not satisfied are pruned. This produces paths that are not only optimal in terms of risk score, but also feasible—every technique in the path has its prerequisites met by prior successful techniques.

The `FindFeasiblePaths` variant starts from currently achieved state nodes (rather than a single start node), enabling real-time path recommendations as the attack progresses. The `NextBestAction` function returns the first edge

of the highest-risk feasible path, providing an AI agent's action recommendation.

6.4 Tradeoff Analysis

Multiple paths to the same objective can be compared along different dimensions:

- **Fast + Loud:** Direct exploitation with high success probability but high detection risk
- **Slow + Stealthy:** Careful credential reuse and living-off-the-land with low detection but longer path
- **Balanced:** Moderate success and stealth, chosen by the risk score heuristic

The stealth scorer's `AnalyzeTradeoff` function explicitly computes both approaches and recommends based on the cumulative detection risk threshold. If the fast approach has detection risk below 0.5, it is recommended; if the quiet approach has significantly lower risk (<50% of fast), stealth is prioritized; otherwise, a balanced approach is suggested.

7. Machine Learning Agents

The ML package implements two adversarial learning agents: tabular Q-learning and REINFORCE policy gradients. Both agents learn to select attack actions that maximize a composite reward function balancing objective achievement, stealth, and detection avoidance.

7.1 State Representation

The simulation state is encoded as a feature vector with per-host features and global features:

Per-host features (7 dimensions): Compromised (binary), privilege level (0–3), EDR level (0–3 normalized), patch level (0–1), hardening level (0–1), service count (normalized), vulnerability count (normalized).

Global features (5 dimensions): Normalized tick count, fraction of hosts compromised, detection rate, stealth score, credential pool size.

A deterministic hash of the state vector enables Q-table lookup while maintaining sufficient state discrimination.

7.2 Action Space

The action space is the Cartesian product of technique IDs × source hosts × target hosts. Valid actions at any state are those where the source host is compromised (or external) and the target is not yet compromised at the same or higher privilege level.

7.3 Reward Function

The composite reward function is:

$$R = R_{\text{tick}} + N_{\text{new}} \times R_{\text{compromise}} + R_{\text{domain}} + R_{\text{objective}} + R_{\text{detect}} + S_{\text{stealth}} \times \beta - R_{\text{cumulative}} \times \gamma$$

Where:

- $R_{tick} = -2$ — Time penalty encouraging faster solutions
- $R_{compromise} = +5$ — Reward for each new host compromised
- $R_{domain} = +10$ — Large reward for achieving domain admin
- $R_{objective} = +3$ — Reward for completing the objective
- $R_{detect} = -5$ — Penalty for being detected
- $S_{stealth} \times 2$ — Bonus for high stealth actions
- $R_{cumulative} \times 3$ — Penalty for accumulated detection risk

7.4 Q-Learning Agent

The Q-learning agent uses ε-greedy exploration with decaying ε and experience replay. The Q-table is indexed by state hash and action index. The update rule is:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

Default hyperparameters: learning rate α=0.1, discount factor γ=0.95, initial ε=1.0 with decay 0.995 and minimum 0.01. Experience replay buffer holds up to 10,000 transitions with batch sampling.

7.5 REINFORCE Policy Gradient Agent

The policy gradient agent uses a linear policy with softmax action selection and a learned baseline:

$$\pi(a|s) = \text{softmax}(W_a \cdot s + b_a)$$

The update uses advantage estimation with a running average baseline:

$$\Delta W_{a,s} = \alpha \times (G_t - b) \times (\delta_{a,a_t} - \pi(a_t|s_t)) \times s_t$$

Weights are initialized with Xavier initialization (scale = $\sqrt{2/(\dim_{state} + \dim_{action})}$). Discount factor γ=0.99, baseline learning rate 0.01.

7.6 Agent Comparison

The package includes random and greedy baselines for evaluation. The random agent selects uniformly from valid actions, while the greedy agent always picks the last valid action (typically the most aggressive). Training runs generate `EpisodeResult` records tracking total reward, ticks, win condition, stealth average, detections, and unique attack paths discovered.

8. Stealth Mechanics

The stealth package implements a cumulative detection probability model that creates realistic operational tempo tradeoffs. Unlike simpler models that treat each action's detection independently, our model accounts for the accumulated noise of prior operations.

8.1 Technique Noise Profiles

Each technique is characterized by a noise profile with six dimensions:

DIMENSION	DESCRIPTION	RANGE
`BaseNoise`	Inherent detection probability	0–1
`EDRBypass`	How well it evades EDR	0–1 (higher = better evasion)
`SIEMBypass`	How well it evades SIEM correlation	0–1
`LogFootprint`	Log evidence left behind	0–1
`NetworkNoise`	Network visibility of the action	0–1
`DiskFootprint`	Disk artifacts left behind	0–1

For example, T1071.004 (C2 via DNS) has BaseNoise=0.15, EDRBypass=0.7, SIEMBypass=0.6, LogFootprint=0.2, NetworkNoise=0.3, DiskFootprint=0.02—reflecting its low visibility but limited bandwidth. By contrast, T1110 (Brute Force) has BaseNoise=0.8, EDRBypass=0.1, NetworkNoise=0.7—reflecting its inherently noisy nature.

8.2 Cumulative Detection Probability

The stealth score for each action is computed as:

$$\text{stealth} = (1 - \text{BaseNoise}) \times (1 - \text{EDR}_{\text{level}} \times 0.1 \times \text{EDRBypass}) \times \text{techniqueModifier} \times \text{opsecSafeBonus}$$

The detection risk for each action incorporates cumulative noise:

$$P_{\text{detect}} = (1 - \text{stealth}) \times (1 + \text{cumulativeRisk} \times 0.5)$$

After each action, cumulative risk increases:

$$\text{cumulativeRisk} \mathrel{+}= (1 - \text{stealth}) \times 0.3$$

And decays by 5% per tick:

$$\text{cumulativeRisk} \mathrel{*}= (1 - 0.05)$$

This creates a natural tension: rapid successive actions increase detection risk, while pauses allow it to decay. Operators must balance speed against stealth, just as real adversaries do.

8.3 Bayesian Cumulative Detection

The overall probability of detection across all actions is computed using Bayesian combination:

$$P(\text{detected}) = 1 - \prod_{i} \left(1 - P_i \times e^{-0.05 \times \text{age}_i} \right)$$

Where age_i is the number of ticks since action i. This weighting ensures recent actions contribute more to detection risk than older ones, modeling the defender's recency bias and log retention.

9. Cloud Attack Modeling

The cloud package models probabilistic attacks against AWS and Azure environments with defender maturity scaling. Unlike on-premises attacks, cloud attacks are modeled as probabilistic outcomes rather than simulation steps, with success and detection probabilities that scale with the target's cloud security posture.

9.1 Cloud Environment Model

A cloud environment includes:

- **Resource groups and compute instances** with managed identities and role assignments
- **Virtual networks** with subnets, NSGs, peering, and DDoS/flow log configuration
- **Storage accounts** with access levels, encryption, and container policies
- **Identity providers** (IAM, Entra ID) with conditional access policies, MFA configuration, and PIM settings
- **Landing zones** with management groups, Azure Policies, and hub-spoke topology

9.2 Attack Models

Three primary attack models are implemented:

Cloud Reconnaissance

Enumerates resources, public storage, managed identities, and MFA-less users. Detection probability uses a sigmoid model:

$$P_{\text{detect}} = \text{maturity} \times 0.03 + \sigma\!\left(\frac{\text{volume}}{20}\right) \times 0.3 \times (1 + \text{maturity} \times 0.3)$$

Where σ is the logistic function. This models the reality that low-volume enumeration is nearly invisible, while mass enumeration triggers anomaly detection.

Cloud Privilege Escalation

Models five escalation paths: managed identity role abuse, password spray against non-MFA users, over-privileged app consent, credential expiry exploitation, and public storage access. Each path has base success and detection rates that are modified by defender maturity and specific controls (PIM, conditional access, Azure Policy).

Cloud Data Exfiltration

Models seven exfiltration channels (S3 presigned URLs, Azure Blob SAS, cloud functions, API gateways, DNS tunneling, s3 sync, azcopy) with method-specific detection curves:

- **Linear:** Detection risk increases linearly with volume (S3 sync, azcopy)
- **Exponential:** Risk accelerates after a threshold (API gateway)
- **Threshold:** Nearly invisible until a volume threshold (DNS tunneling, cloud functions)

10. Command and Control Infrastructure

The C2 module models adversary command and control infrastructure including redirectors, implants, dead drops, and beacon behavior. This models the full C2 lifecycle from deployment through detection.

10.1 C2 Channel Profiles

Each compromised host can establish a C2 channel from available profiles, which are selected based on the host's network position and available services. Profiles specify:

- **Protocol:** HTTPS, DNS, HTTP, WebSocket, custom
- **Beacon interval:** How frequently the implant calls home
- **Jitter:** Random variation in beacon timing
- **Stealth level:** 0–1, how well the channel blends with legitimate traffic
- **Success probability:** Based on protocol and egress filtering
- **Detection probability:** Based on beacon regularity and protocol characteristics

10.2 C2 Infrastructure

The C2 infrastructure model includes:

- **Team server:** Central command with active implant tracking
- **Redirectors:** Front-end proxies that forward traffic to the team server
- **CDN fronting:** Legitimate CDN domains used as C2 fronts
- **Dead drops:** One-way communication channels (file shares, web services)
- **Implants:** Deployed payloads with configuration, capabilities, and health tracking

The `CheckInfrastructureResilience` function evaluates the C2 infrastructure's resilience by testing redirector availability, implant health, and detection surface area.

10.3 Beacon Detection

C2 beacons are simulated every 5 ticks. Each beacon has a detection probability determined by the channel's stealth characteristics and defender maturity. DNS tunneling beacons are additionally checked for pattern anomalies using the protocol-level DNS simulation.

11. Purple Team Operations

The team package implements multi-team exercise support with four team roles (red, blue, purple, white) and three interaction modes (competitive, cooperative, purple).

11.1 Exercise Management

An exercise is configured with rules that govern engagement:

- **Engagement windows:** Configurable start/end ticks for red team activity
- **Blue team delay:** Ticks before blue team detection activates
- **Scoring mode:** Points-based, objectives-based, or hybrid

- **Stealth bonus:** Multiplier for OPSEC-safe techniques (1.5×)
- **Detection penalty:** Point reduction for detected actions (0.5×)
- **Collateral penalty:** Penalty for affecting non-target systems
- **Deconfliction:** Whether teams must coordinate to avoid operating on the same targets

11.2 Purple Team Coordination

In purple team mode, red and blue teams share intelligence to maximize learning:

- **Intel sharing:** Technique details, detection indicators, lessons learned
- **Deconfliction:** Preventing teams from operating on the same hosts simultaneously
- **Mutual scoring:** Both teams score from detection improvement, not just from evasion success

11.3 Observer Mode

White team observers have full visibility into all team actions, detections, and communications. This enables exercise controllers to:

- Track all red team actions with stealth scores and detection outcomes
- Monitor all blue team detections and hunt results
- Record inter-team communications for post-exercise analysis
- Produce comprehensive after-action reports

12. After-Action Review and Replay

The AAR package generates comprehensive reports from simulation data, and the replay package provides time-travel debugging capabilities.

12.1 AAR Report Structure

Each simulation produces a report containing:

- **Red team summary:** Objectives met, hosts compromised, techniques used, stealth average, detection rate, effectiveness score
- **Blue team summary:** Alerts generated/confirmed/false-positive, MTTD/MTTC, hosts contained, detection gaps
- **Kill chain report:** Phase-by-phase progress through 12 ATT&CK tactics with technique mapping
- **Technique analysis:** Per-technique statistics (uses, success rate, stealth, detection rate)
- **Findings:** Critical/high/medium/low findings with evidence and MITRE mapping
- **Recommendations:** Prioritized remediation actions (detection, hardening, policy, monitoring) with effort/impact ratings
- **Simulation metrics:** Compromise rate, detection rate, risk level, red/blue team scores

12.2 Time-Travel Replay

The replay system captures periodic state snapshots (configurable interval) and event logs. It supports:

- **Forward/backward navigation:** Jump to any tick in the simulation
- **Labeled bookmarks:** Mark key moments (initial compromise, first detection, objective achieved)
- **Diff comparison:** Compare state changes between two ticks
- **Export:** Save replay data to JSON for external analysis

13. Federation and Interoperability

The federation package implements an IEEE 1278.1 DIS (Distributed Interactive Simulation) bridge that enables cyber-redteam-sim to interoperate with other simulation platforms, including military simulation systems and physical training environments.

13.1 PDU Mapping

Cyber events are mapped to DIS PDU types as follows:

CYBER EVENT	DIS PDU TYPE	NOTES
Exploit attempt	Fire PDU	Munition type = cyber exploit category
Host compromise	Detonation PDU	Result: 2=partial, 3=full compromise
C2 communication	Signal PDU	Payload contains C2 details as JSON
Detection/alert	Event Report PDU	Event type = cyber detection/alert
Host state change	Entity State PDU	Regular heartbeat (default: 5s)
Scanning	Event Report PDU	Cyber scan event type

13.2 Entity Mapping

Network hosts are registered as DIS entities with:

- **Entity type:** Domain=7 (cyber), Category mapped by role (server=1, workstation=2, firewall=3)
- **Force ID:** 1=friendly (blue), 2=opposing (red), 3=neutral
- **Marking:** Hostname as entity marking
- **World coordinates:** Mapped from network topology positions

13.3 Federation Manager

The federation manager handles lifecycle (join/leave), heartbeat publication, event translation (bidirectional), and remote entity tracking. It supports both async (heartbeat-based) and synchronous (tick-based) PDU publication.

14. Performance and Deployment

14.1 Binary Characteristics

The engine compiles to a single static binary of approximately 2.8 MB (Linux amd64), with no external runtime dependencies. This small footprint enables deployment in constrained environments including embedded systems and air-gapped networks.

PLATFORM	FORMAT	SIZE
Linux amd64	.tar.gz / .deb	~2.8 MB
Linux arm64	.tar.gz	~2.5 MB
macOS Intel	.tar.gz	~2.8 MB
macOS ARM	.tar.gz	~2.6 MB
Windows amd64	.zip	~2.8 MB
Docker	stsgym/cyber-redteam-sim:latest	Alpine-based

14.2 API Interfaces

The engine provides three API interfaces:

- **REST API (port 8080):** HTTP JSON API for starting simulations, querying state, and retrieving results
- **gRPC/JSON-RPC:** High-performance RPC interface with an adapter layer for the core engine

- **Web UI:** Leaflet.js tactical display with Server-Sent Events (SSE) for real-time network visualization

14.3 Distribution

The engine is distributed through multiple channels:

- Debian package (.deb) with systemd unit, postinst/prerm lifecycle scripts
- Homebrew tap (`brew install wezzels/tap/cyber-redteam-sim`)
- Docker image (Alpine 3.19 based, with health check)
- Platform-specific tarballs with install scripts
- Snap package configuration

Cross-compilation and release automation are handled by GoReleaser with GitHub Actions CI/CD for testing and GitLab CI for internal distribution.

15. Conclusion

We have presented cyber-redteam-sim, a high-fidelity network attack and defense simulation engine that models the complete adversarial lifecycle from initial reconnaissance through data exfiltration. The engine's three key innovations—state-dependent attack graph pathfinding, cumulative stealth with temporal decay, and adversarial ML agents—address fundamental gaps in existing simulation tools.

The attack graph module enables strategic reasoning about multi-step compromise paths, producing risk-scored paths that balance success probability against detection risk. The prerequisite state machine ensures that only feasible paths are considered, while the A* search efficiently finds optimal routes through the exponentially large space of possible attack sequences.

The cumulative stealth model creates realistic operational tempo constraints where attackers must balance speed against detection risk. The Bayesian cumulative detection probability, weighted by temporal decay, models how defenders' awareness builds with repeated suspicious activity—a

phenomenon well-documented in real incident response but absent from simpler simulation models.

The ML agents demonstrate that both tabular Q-learning and REINFORCE policy gradient methods can learn effective attack strategies through episodic simulation. The Q-learning agent's experience replay and the policy gradient's advantage estimation provide complementary learning dynamics: Q-learning excels at learning precise state-action values for frequently visited states, while REINFORCE generalizes better across similar states.

The engine's protocol-level fidelity (TCP state machines, SMB negotiation, Kerberos exchanges, LDAP queries), probabilistic cloud attack modeling, C2 infrastructure simulation, and comprehensive defense modeling (SIEM correlation, EDR monitoring, incident response playbooks) produce simulation outputs that closely mirror real-world red team engagement results.

Future work includes extending the cloud attack models to GCP, adding ICS/SCADA protocol simulation (Modbus, DNP3), implementing Active Directory schema modeling with full ACL/ACE support, and exploring Monte Carlo Tree Search as an alternative to A* for attack graph exploration in environments with stochastic outcomes.

References

1. **MITRE ATT&CK Framework.** MITRE Corporation, 2024. Available at: `https://attack.mitre.org/`
2. **Ning, P., Xu, D., Healy, C.G., et al.** "Building an attack graph for enterprise networks." Proceedings of the 2002 Workshop on Rapid Malcode (WORM'02), 2002.
3. **Ou, X., Boyer, W.F., McQueen, M.A.** "A scalable approach to attack graph generation." Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS'06), 2006.
4. **Watkins, C.J.C.H., Dayan, P.** "Q-learning." Machine Learning, 8(3):279–292, 1992.
5. **Williams, R.J.** "Simple statistical gradient-following algorithms for connectionist reinforcement learning." Machine Learning, 8(3):229–256, 1992.
6. **Mnih, V., Kavukcuoglu, K., Silver, D., et al.** "Human-level control through deep reinforcement learning." Nature, 518(7540):529–533, 2015.

7. **IEEE Standard for Distributed Interactive Simulation (DIS).** IEEE Std 1278.1-2012, IEEE, 2012.
8. **NIST Cybersecurity Framework (CSF) 2.0.** National Institute of Standards and Technology, 2024.
9. **CMMC Model v2.0.** Department of Defense, 2021.
10. **MITRE SHIELD: Active Defense.** MITRE Corporation, 2020. Available at: `https://shield.mitre.org/`
11. **Almorsy, M., Grundy, J., Ibrahim, A.M.** "Supporting architecture vulnerability analysis using architectural models." Proceedings of the 2nd International Workshop on Software Engineering for Secure Systems (SESS'06), 2006.
12. **Silver, D., Schrittwieser, J., Simonyan, K., et al.** "Mastering the game of Go without human knowledge." Nature, 550(7676):354–359, 2017.
13. **Noel, S., Jajodia, S.** "Optimal IDS sensor placement for attack prediction." Proceedings of IFIP SEC 2006, 2006.
14. **Chess, D.M., Palmer, C.C., White, S.R.** "Security and the SWARM: Using networked embedded systems to secure enterprise networks." Proceedings of the 2007 Workshop on Scalable Information Systems (InfoScale'07), 2007.
15. **Sutton, R.S., Barto, A.G.** Reinforcement Learning: An Introduction. MIT Press, 2nd edition, 2018.

This paper describes the open-source simulation engine available at `idm.wezzel.com/crab-meat-repos/cyber-redteam-sim`