SEAS-8414 CYBER ANALYTICS

05

Autonomous Penetration Testing

MDP Formulation, Agent Architecture & Exploit Execution

Dr. Mallarapu · Breakwater Security Platform

What Phases 1-4 Built

The foundation for autonomous exploitation

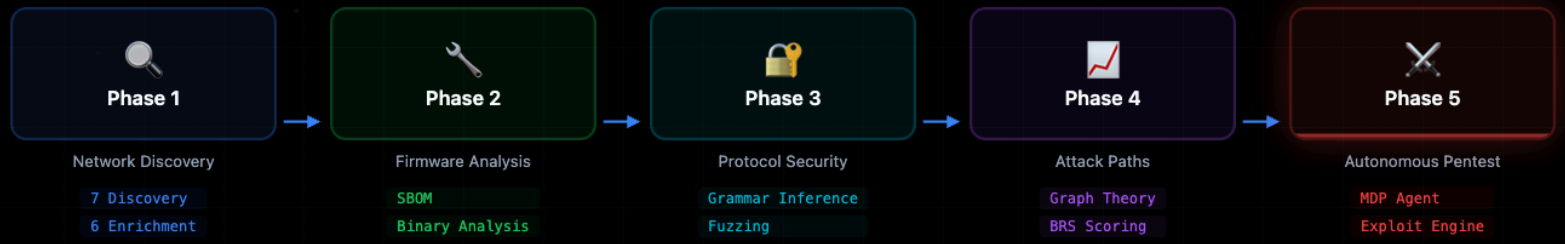
CHAPTER TAKEAWAY

March 2024. Meridian Capital Partners. Hedge fund in midtown Manhattan

ENRICHMENT VALUE

That connection is what this chapter builds

PROGRESSIVE SECURITY ARCHITECTURE



Phase 1
What is on the network?
573 tests

Phase 2
What is inside each device?
791 tests

Phase 3
How do devices behave?
2,101 tests

Phase 4
How do risks compound?
2,309 tests

Phase 5
Can we prove exploitation?
Phase 5

The Phase 4 Limitation

Theoretical risk vs confirmed exploitation

CHAPTER TAKEAWAY

Look at Table 5.0 in the textbook

ENRICHMENT VALUE

Descriptive: what is on the network?

📊 Theoretical (Phase 4)

- **Attack paths computed via graph theory**
  Mathematical probability only
- **BRS scores from static analysis**
  No runtime validation
- **CVE presence assumed exploitable**
  Many CVEs are not reachable
- **Remediation priority is estimated**
  No ground truth data

VS

⚔️ Confirmed (Phase 5)

- **Exploits executed with evidence**
  Cryptographic proof chains
- **Actual lateral movement verified**
  Real credential pivoting
- **False positives eliminated**
  Only confirmed vulnerabilities
- **Time-to-compromise measured**
  Empirical attack timelines

CHAPTER TAKEAWAY

Qualys found 214 CVEs on Meridian's trading floor

ENRICHMENT VALUE

That is the practical power of prescriptive analytics

> Of your 225 CVEs, only 8 are actually exploitable from the guest WiFi. Here is the evidence chain for each one.

- Phase 4 attack graph input
- Autonomous confirmation
- Evidence-backed proof
- Entry point validated
- SHA-256 hash proof
- Tamper-proof report

Phase 5 Architecture

MDP Agent -> Executor -> Evidence -> Report

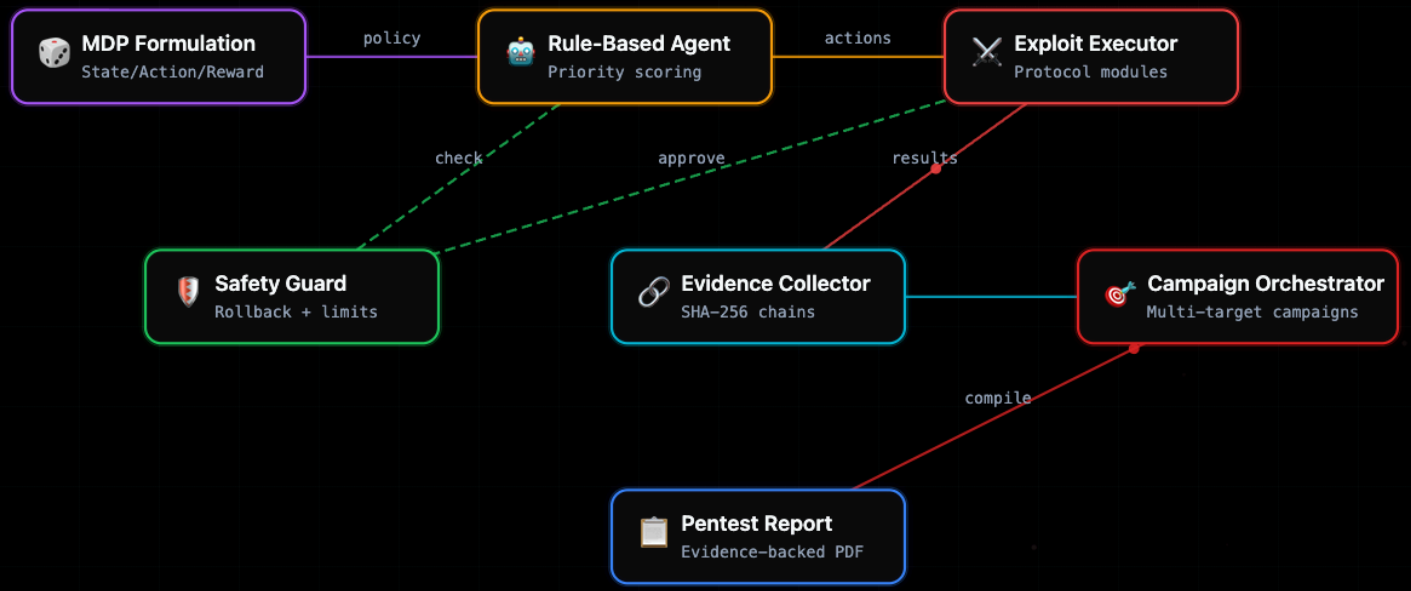
CHAPTER TAKEAWAY

The pentest engine does not operate in isolation

ENRICHMENT VALUE

All of this feeds the agent's decision space

Phase 5 introduces an autonomous penetration testing agent built on a Markov Decision Process formulation. The MDP agent selects exploit actions, the executor dispatches them to protocol-specific modules, and all results are cryptographically chained into tamper-proof evidence. A safety guard enforces blast-radius limits and rollback triggers throughout the pipeline.



Research Landscape

Key papers in autonomous penetration testing

CHAPTER TAKEAWAY

A pentest campaign has five stages

ENRICHMENT VALUE

And a hard timeout to prevent runaway campaigns

- 2020 **PentestGPT**
  Early LLM pentest
- 2022 **AutoAttacker**
  RL-based exploit selection
- 2023 **CLAP**
  Chen et al. — LLM agent planning
- 2024a **PwnGPT**
  Xu et al. — GPT-4 autonomous pentest
- 2024b **AutoPenBench**
  Gioacchini et al. — Benchmark framework
- 2025 **Breakwater Phase 5**
  MDP + safety + evidence chains

CLAP: LLM-Guided Penetration Testing

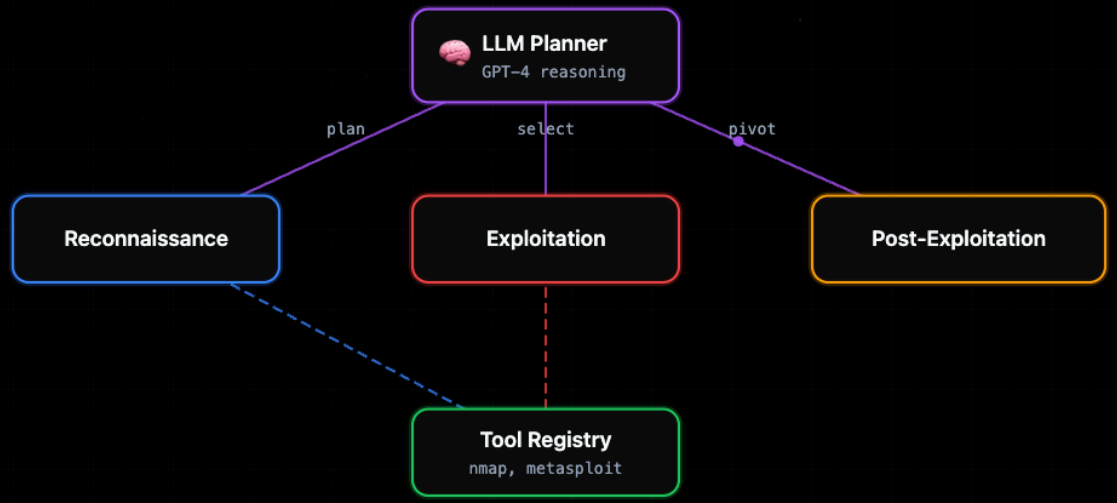
Chen et al. 2024 — Closed-Loop Agent for Pentesting

CHAPTER TAKEAWAY

Six parameters configure every campaign

ENRICHMENT VALUE

The legal document becomes the API request



KEY CONTRIBUTIONS

- LLM-in-the-loop planning
- Tool abstraction layer
- Iterative refinement
- No formal MDP

PwnGPT: GPT-4 Autonomous Pentest

Xu et al. 2024 — Success rates across scenarios

CHAPTER TAKEAWAY

Two filters before any action is taken

ENRICHMENT VALUE

Graduated escalation. Start broad and cheap. Narrow and approve

SCENARIO	SUCCESS RATE	AVG TIME	TOOLS USED
Web Application	72%	14 min	SQLMap, Nikto
Network Service	58%	23 min	Nmap, Metasploit
IoT Device	45%	31 min	Custom scripts
Privilege Escalation	33%	42 min	LinPEAS, GTFObins
Multi-hop Lateral	19%	67 min	Mixed

AutoPenBench: Standardized Benchmarking

Gioacchini et al. 2024 — Reproducible agent evaluation

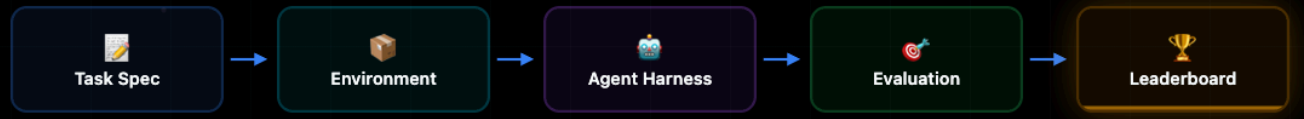
CHAPTER TAKEAWAY

The agent runs inside asyncio.to_thread

ENRICHMENT VALUE

Why? The agent's run method is synchronous

BENCHMARK PIPELINE



Completion Rate
Fraction of milestones achieved

Time Efficiency
Steps to reach each milestone

Tool Diversity
Number of distinct tools used

Reproducibility
Variance across repeated runs

MDP Definition

Formal 5-tuple for autonomous pentesting

CHAPTER TAKEAWAY

The pentest problem is a Markov Decision Process

ENRICHMENT VALUE

And penetration testing is exactly that: sequential decisions under uncertainty

Markov Decision Process

1. MDP 5-tuple

$$M = (S, A, T, R, \gamma)$$

2. State space: network knowledge at time t

$$S = \{s_0, s_1, \ldots, s_n\}$$

3. Action space: pentest operations

$$A = \{\text{scan}, \text{exploit}, \text{pivot}, \ldots\}$$

4. Transition: stochastic outcome of action

$$T(s' \mid s, a) = \Pr[s' \mid s, a]$$

5. Reward: exploit success, coverage gain

$$R(s, a, s') \in \mathbb{R}$$

6. Discount factor: prefer faster exploits

$$\gamma \in [0, 1)$$

State Space Components

s_t = (topology, services, vulns, creds, access)

CHAPTER TAKEAWAY

PentestState tracks seven things

ENRICHMENT VALUE

step_count and total_reward for bookkeeping

Network Topology
Discovered hosts, subnets, connectivity graph

s1

Service Map
Open ports, service versions, banners per host

s2

Vulnerability Set
Known CVEs, exploit availability, CVSS scores

s3

Credential Store
Harvested credentials, access tokens, keys

s4

Access Level
Current privilege level per compromised host

s5

Action Space

Six action types available to the pentest agent

CHAPTER TAKEAWAY

Eight discrete action types

ENRICHMENT VALUE

available_actions prunes impossible moves

ACTION	DESCRIPTION	PRECONDITION	RISK
SCAN	Port scan / service discovery on target	Target IP known	Low
EXPLOIT	Execute CVE exploit against service	Vuln + exploit available	High
CREDENTIAL	Try default/harvested credentials	Service accepts auth	Medium
PIVOT	Lateral movement to new host	Shell on current host	High
ESCALATE	Privilege escalation on current host	Low-priv shell	Medium
EXFIL	Evidence collection and data extraction	Sufficient access	Low

Action Type Details

Parameters, outputs, and example invocations

CHAPTER TAKEAWAY

Every action type maps to MITRE ATT&CK techniques

ENRICHMENT VALUE

The agent is not just attacking. It is generating a MITRE-annotated attack narrative

SCAN

params: target_ip, port_range, scan_type
output: open_ports[], service_versions[]

```
SCAN(192.168.1.10, 1-1024, SYN)
```

EXPLOIT

params: target_ip, port, cve_id, payload
output: shell_session | failure

```
EXPLOIT(10.0.0.5, 80, CVE-2024-1234, reverse_shell)
```

CREDENTIAL

params: target_ip, port, protocol, creds[]
output: authenticated_session | failure

```
CREDENTIAL(10.0.0.5, 22, ssh, admin:admin)
```

PIVOT

params: source_ip, target_ip, route_method
output: tunnel_established | failure

```
PIVOT(10.0.0.5, 10.0.1.3, ssh_tunnel)
```

ESCALATE

params: host_ip, technique_id
output: elevated_shell | failure

```
ESCALATE(10.0.0.5, sudo_nopasswd)
```

EXFIL

params: host_ip, evidence_type, path
output: evidence_artifact + hash

```
EXFIL(10.0.0.5, file, /etc/shadow)
```

Reward Component Details

Balancing exploitation depth with exploration breadth

CHAPTER TAKEAWAY

Seven reward components

ENRICHMENT VALUE

The step cost creates urgency. Without it, the agent would scan forever

🌐

Coverage

30%

Reward discovering new hosts and services

`+0.15 for finding 3 new hosts`

⚔️

Confirmation

40%

Reward confirming exploitable vulnerabilities

`+0.8 for confirmed RCE on critical host`

🤫

Stealth

15%

Penalize actions that trigger IDS alerts

`-0.3 for triggering Snort rule`

💡

Novelty

15%

Reward trying new action-target combinations

`+0.1 for first SSH attempt on host`

MDP Worked Example

Four-step attack sequence with state transitions and rewards

CHAPTER TAKEAWAY

The agent compromises the Meridian TV with default credentials

ENRICHMENT VALUE

Critical host? No, it is a consumer TV. No +20 bonus

t=0 **+0.15**

s0: 1 host known, 0 shells

`SCAN(10.0.0.5, 1–1024)`

3 ports found (22, 80, 8080)

t=1 **+0.60**

s1: 1 host, 3 services

`CREDENTIAL(10.0.0.5:22, admin:admin)`

SSH shell obtained

t=2 **+0.80**

s2: 1 host, user shell

`ESCALATE(10.0.0.5, CVE-2024-1086)`

Root shell obtained

t=3 **+0.25**

s3: 1 host, root shell

`PIVOT(10.0.0.5 -> 10.0.1.0/24)`

4 new hosts discovered

SECTION 04

Rule-Based Agent

Heuristic policy for exploit candidate selection

Candidate Generation

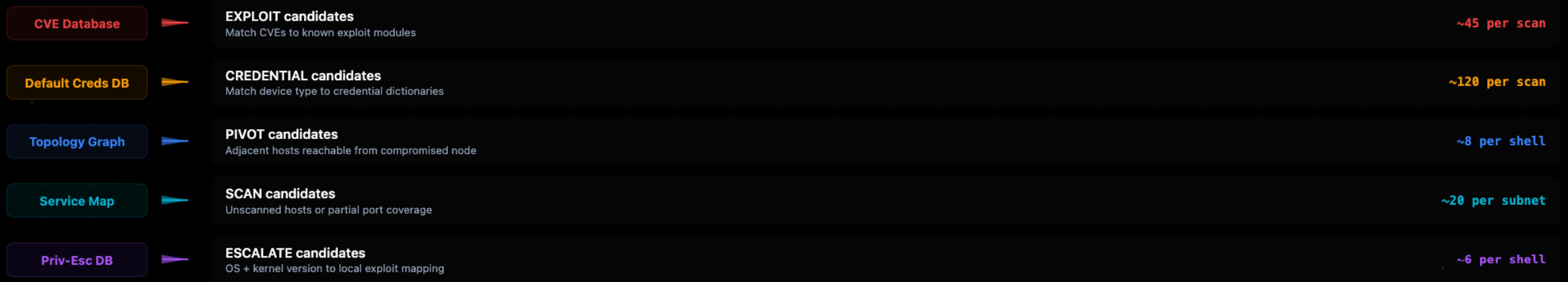
How the agent enumerates valid actions from current state

CHAPTER TAKEAWAY

Three hosts on the board

ENRICHMENT VALUE

Question: what are the first three actions the rule-based agent takes?



Agent Decision Loop

One iteration of the rule-based agent policy

CHAPTER TAKEAWAY

The simpler agent. Deterministic priority heuristic

ENRICHMENT VALUE

Credentials first. High-CVSS CVEs second. Lateral movement third

Each iteration of the agent loop encodes the current network state, enumerates roughly 200 candidate actions, scores and filters them through the safety controller, executes the top pick, and updates the state with newly discovered knowledge. This loop repeats until a termination condition is met (time budget, coverage threshold, or manual kill-switch).

AGENT LOOP (REPEATS UNTIL TERMINATION)

Credential Pivoting Strategy

Harvested credentials enable lateral movement across the network

CHAPTER TAKEAWAY

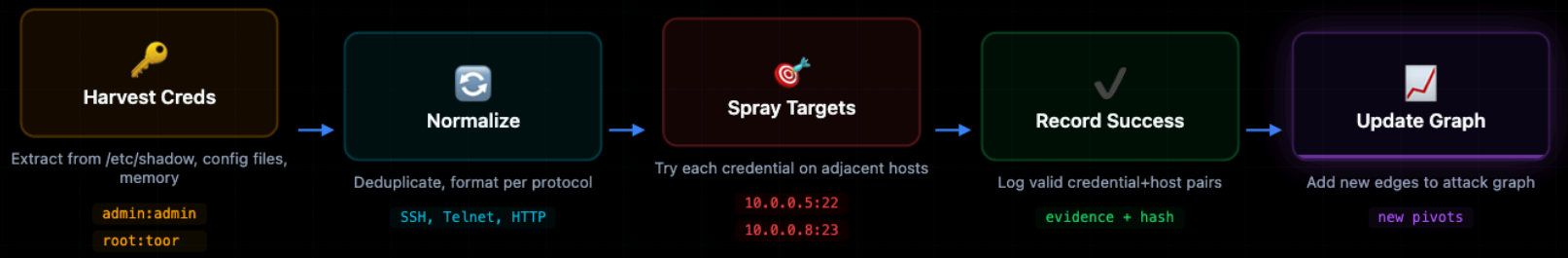
In real-world pentests, default credentials have the highest success rate

ENRICHMENT VALUE

Score of 100 means: try this first, every time

When the agent compromises a host, it harvests credentials from config files, shadow databases, and memory. These are normalized per protocol (SSH, Telnet, HTTP) and sprayed against adjacent hosts. Each successful login creates new edges in the attack graph, unlocking further lateral movement. This is the primary mechanism for deep network penetration beyond the initial foothold.

CREDENTIAL REUSE PIPELINE



Rule-Based Agent Limitations

Where heuristic policies fall short and what comes next

CHAPTER TAKEAWAY

Every action gets a SHA-256 hash

ENRICHMENT VALUE

Why sort_keys? Python dictionaries maintain insertion order since 3.7

Fixed Weight Sensitivity
Priority weights (w1=0.35, w2=0.30...) are hand-tuned and may not generalize across network topologies

FUTURE WORK
> RL-learned weights via PPO

Greedy Selection
Always picks highest-scored action without considering long-term payoff of multi-step attack chains

FUTURE WORK
> Monte Carlo Tree Search (MCTS)

No Memory of Failures
Repeated failures on similar targets are not learned from across campaigns

FUTURE WORK
> Experience replay buffer

Static Preconditions
Precondition checks are boolean; cannot handle partial or probabilistic preconditions

FUTURE WORK
> Fuzzy logic preconditions

Campaign Trace: 12-Step Example

Complete rule-based agent execution on IoT network

CHAPTER TAKEAWAY

The rule-based agent works on standard networks

ENRICHMENT VALUE

Proximal Policy Optimization. PPO

#1	SCAN	10.0.0.0/24	6 hosts found	0.85	#2	SCAN	10.0.0.5:1-1024	SSH(22), HTTP(80)	0.72
#3	CREDENTIAL	10.0.0.5:22	admin:admin SUCCESS	0.91	#4	EXFIL	10.0.0.5	/etc/shadow harvested	0.65
#5	ESCALATE	10.0.0.5	Root via sudo NOPASSWD	0.88	#6	PIVOT	10.0.0.5->10.0.1.0/24	3 hosts discovered	0.78
#7	SCAN	10.0.1.3:1-1024	RTSP(554), ONVIF(80)	0.70	#8	CREDENTIAL	10.0.1.3:554	admin:admin SUCCESS	0.93
#9	EXPLOIT	10.0.1.5:80	Path traversal confirmed	0.87	#10	CREDENTIAL	10.0.1.8:23	root:root SUCCESS	0.90
#11	EXFIL	10.0.1.8	Config file extracted	0.62	#12	TERMINATE	Campaign	4/6 hosts compromised	0.00

SECTION 05

Exploit Execution

Protocol-specific modules with evidence collection

Exploit Executor Design

Protocol-specific module registry with pluggable handlers

CHAPTER TAKEAWAY

The action space is two-dimensional: where and what

ENRICHMENT VALUE

Tactical level: probability over action types for the chosen host

PROTOCOL	PORT	MODULE	CAPABILITIES
HTTP	80/443	http_exploit.py	Path traversal, admin panel, debug endpoints
SSH	22	ssh_exploit.py	Credential stuffing, key auth, command exec
Telnet	23	telnet_exploit.py	Expect-style interaction, banner grab
RTSP	554	rtsp_exploit.py	Auth bypass, DESCRIBE without creds
MQTT	1883	mqtt_exploit.py	Wildcard subscribe, topic enumeration
ONVIF	80/8080	onvif_exploit.py	Device info, stream URI, PTZ control

Exploit Module Interface

Abstract base class all protocol modules must implement

CHAPTER TAKEAWAY

Not every host has every action type available

ENRICHMENT VALUE

This is a hard constraint, not a soft reward penalty

exploit_module.py — PYTHON

```python
class ExploitModule(ABC):
    """Base class for all protocol exploit modules."""

    @abstractmethod  ← Pre-flight validation
    async def check(self, target: Target) -> bool:
        """Pre-flight: is this exploit applicable?"""

    @abstractmethod  ← Core exploit logic
    async def execute(self, target: Target) -> ExploitResult:
        """Run the exploit and return result + evidence."""

    @abstractmethod  ← Mandatory cleanup
    async def cleanup(self, target: Target) -> None:
        """Rollback any changes made during exploitation."""

    def evidence_hash(self, data: bytes) -> str:  ← Evidence chain
        """SHA-256 hash for evidence integrity."""
        return hashlib.sha256(data).hexdigest()

    @property
    def risk_level(self) -> str:
        """LOW | MEDIUM | HIGH — used by safety guard."""
        return "MEDIUM"
```

Evidence Data Model

Cryptographically-linked evidence artifacts

CHAPTER TAKEAWAY

Start from the probability ratio

ENRICHMENT VALUE

Default epsilon: 0.2. So the clip window is 0.8 to 1.2

evidence_schema.py — PYTHON

```python
@dataclass
class ExploitEvidence:
    evidence_id: str          # UUID v4    ← Unique identifier
    timestamp: datetime       # UTC ISO-8601
    target_ip: str            # 10.0.0.5
    target_port: int          # 22
    protocol: str             # ssh | http | telnet | rtsp | mqtt | onvif
    exploit_type: str         # credential | cve | misconfiguration
    severity: str             # critical | high | medium | low
    success: bool             # True if exploitation confirmed
    request_data: bytes       # Raw request sent
    response_data: bytes      # Raw response received
    screenshot: bytes | None  # Optional visual evidence    ← Tamper detection
    credential: str | None    # Credential used (if applicable)    ← Chain link to prev
    sha256_hash: str          # Hash of (request + response + timestamp)    ← Campaign integrity
    prev_hash: str | None     # Previous evidence hash (chain link)
    merkle_root: str | None   # Campaign-level merkle root
```

Evidence Collection Pipeline

Six steps from exploit execution to tamper-proof storage

CHAPTER TAKEAWAY

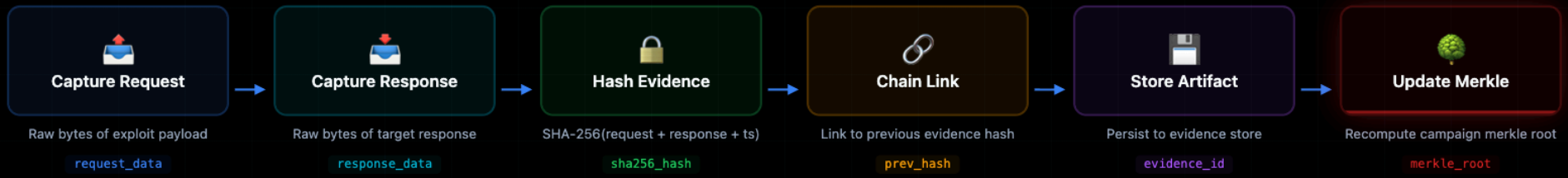
Board work. Draw the graph

ENRICHMENT VALUE

They keep updates conservative even when the gradient says "move a lot"

Every exploit execution generates a cryptographic evidence chain. Raw request and response bytes are captured, hashed with SHA-256 including a timestamp, and linked to the previous evidence hash to form an append-only chain. The campaign's Merkle root is recomputed after each new artifact, enabling O(log n) tamper verification of the entire evidence set.

EVIDENCE COLLECTION FLOW



Executor Module Architecture

Registry dispatches to protocol-specific exploit handlers

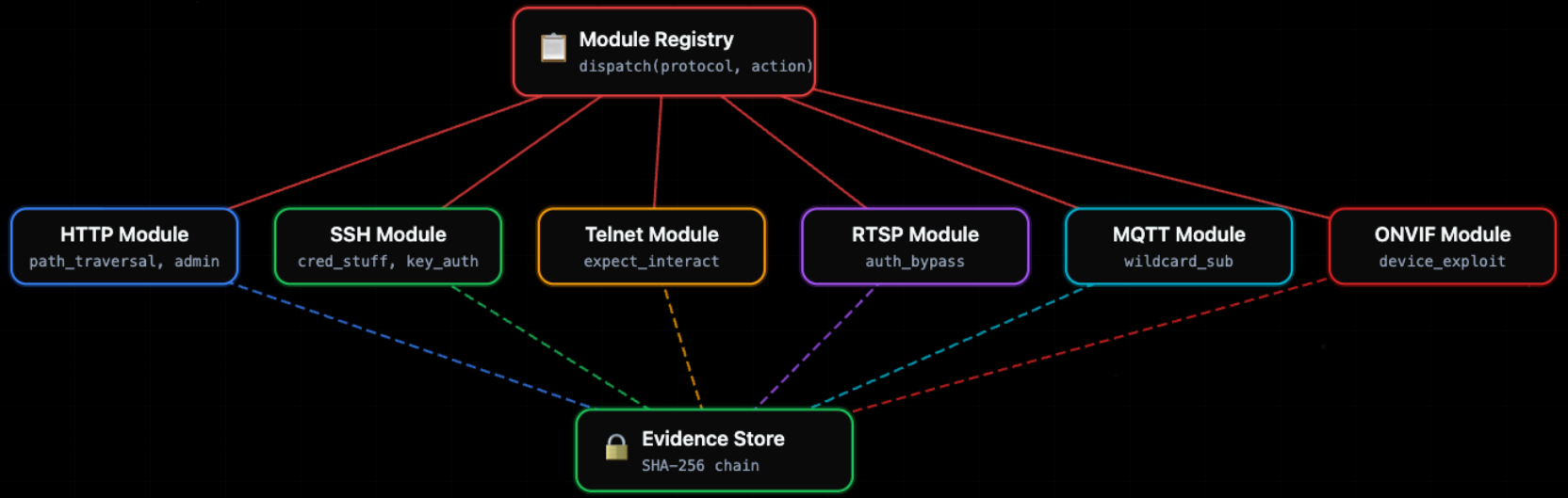
CHAPTER TAKEAWAY

The advantage measures: how much better was this action than average?

ENRICHMENT VALUE

The advantage measures: how much better was this action than average?

The module registry uses a dispatch pattern to route exploit actions to protocol-specific handlers. Each module (HTTP, SSH, Telnet, RTSP, MQTT, ONVIF) implements a common interface but encapsulates protocol-specific logic. All modules feed results into a shared evidence store with SHA-256 chain integrity, ensuring a uniform forensic record regardless of which protocol was exploited.



Telnet Credential Stuffing

Expect-style interaction pattern for IoT device login

CHAPTER TAKEAWAY

Four fine-grained components on top of the base reward

ENRICHMENT VALUE

Compliance focus? Crank novelty to 3.0 for MITRE coverage

telnet-exploit

```
# Telnet Credential Stuffing — Expect-Style Interaction

$ telnet-exploit --target 10.0.1.8:23 --creds default_iot.txt
Connecting to 10.0.1.8:23...
Banner: BusyBox v1.24.2 (2019-04-15)

EXPECT: "login: " -> SEND: "admin"
EXPECT: "Password: " -> SEND: "admin"
Login incorrect.

EXPECT: "login: " -> SEND: "root"
EXPECT: "Password: " -> SEND: "root"
# (root shell obtained)

$ cat /proc/version
Linux 3.18.24 armv7l

[CONFIRMED] Default creds root:root on 10.0.1.8:23 — CRITICAL
$
```

SSH Credential Testing

Paramiko-based credential stuffing with evidence capture

CHAPTER TAKEAWAY

Training on full networks from scratch is inefficient

ENRICHMENT VALUE

Without curriculum: over 500 episodes

ssh-exploit

```
# SSH Credential Testing Module

$ ssh-exploit --target 10.0.0.5:22 --mode credential-stuff
SSH Banner: SSH-2.0-OpenSSH_7.9p1
Auth methods: password, publickey

Trying admin:admin ... Authentication failed
Trying admin:password ... Authentication failed
Trying admin:1234 ... Authentication failed
Trying admin:admin123 ... SUCCESS

$ id && hostname && uname -a
uid=1000(admin) gid=1000(admin)
iot-camera-01
Linux 4.19.0-arm armv7l

[CONFIRMED] SSH cred admin:admin123 on 10.0.0.5 — HIGH
$
```

RTSP Authentication Bypass

DESCRIBE without credentials reveals video stream metadata

CHAPTER TAKEAWAY

Four modes of graduated control

ENRICHMENT VALUE

Autonomous is for the lab

rtsp-exploit

```
# RTSP Authentication Bypass — DESCRIBE Method

$ rtsp-exploit --target 10.0.1.3:554 --mode auth-bypass

>> DESCRIBE rtsp://10.0.1.3:554/stream1 RTSP/1.0
>> CSeq: 1
>> (no Authorization header)

<< RTSP/1.0 200 OK
<< Content-Type: application/sdp
<< m=video 0 RTP/AVP 96
<< a=rtpmap:96 H264/90000

AUTH BYPASS: Stream accessible without credentials!
Stream URI: rtsp://10.0.1.3:554/stream1
Codec: H.264, Resolution: 1920x1080

[CONFIRMED] RTSP auth bypass on 10.0.1.3:554 — HIGH
$
```

MQTT Wildcard Subscription

Unauthenticated broker exposes all topics via # wildcard

CHAPTER TAKEAWAY

The default. The safe starting point

ENRICHMENT VALUE

This is the mode that unlocks prescriptive analytics without operational risk

mqtt-exploit

```
# MQTT Wildcard Subscription — Topic Enumeration

$ mqtt-exploit --target 10.0.1.12:1883 --mode wildcard
Connecting to MQTT broker (no auth required)...
Connected! Client ID: breakwater-pentest-a3f4

$ SUBSCRIBE topic="#" QoS=0
Subscription confirmed for wildcard topic #

sensors/temp/room1     -> {"temp": 22.4, "unit": "C"}
sensors/door/front     -> {"status": "open", "ts": 1709...}
config/wifi            -> {"ssid":"CorpNet","psk":"P@ss123"}
admin/credentials      -> {"user":"admin","pass":"factory"}
devices/camera/status -> {"recording": true, "ip": "10.0.1.3"}

5 topics captured, 2 contain credentials
[CONFIRMED] MQTT no-auth + wildcard on 10.0.1.12 — CRITICAL
$
```

ONVIF Device Exploitation

Camera control via WS-Security with default credentials

CHAPTER TAKEAWAY

Only port_scan and service_enum permitted

ENRICHMENT VALUE

Use case: "which hosts would the agent try to exploit, and in what order?"

onvif-exploit

```
# ONVIF Device Exploitation — WS-Security bypass

$ onvif-exploit --target 10.0.1.3:80 --mode full
Step 1: GetDeviceInformation (no auth)...
Manufacturer: HikVision, Model: DS-2CD2142, FW: 5.4.5

Step 2: GetStreamUri (default creds admin:12345)...
Main: rtsp://10.0.1.3:554/Streaming/Channels/101
Sub:  rtsp://10.0.1.3:554/Streaming/Channels/102

Step 3: GetUsers...
Users: admin (Administrator), viewer (User)

Step 4: PTZ ContinuousMove (proving control)...
Pan: +0.1, Tilt: +0.1 — camera moved successfully

[CONFIRMED] ONVIF full control on 10.0.1.3 — CRITICAL
$
```

SHA-256 Evidence Hash Chains

Each evidence artifact links to the previous via cryptographic hash

CHAPTER TAKEAWAY

The production-safe mode for most organizations

ENRICHMENT VALUE

At Meridian: controlled mode against the TV and sensor only

Tampering with any evidence artifact **breaks the chain** and invalidates all subsequent hashes

Merkle Proof Structure

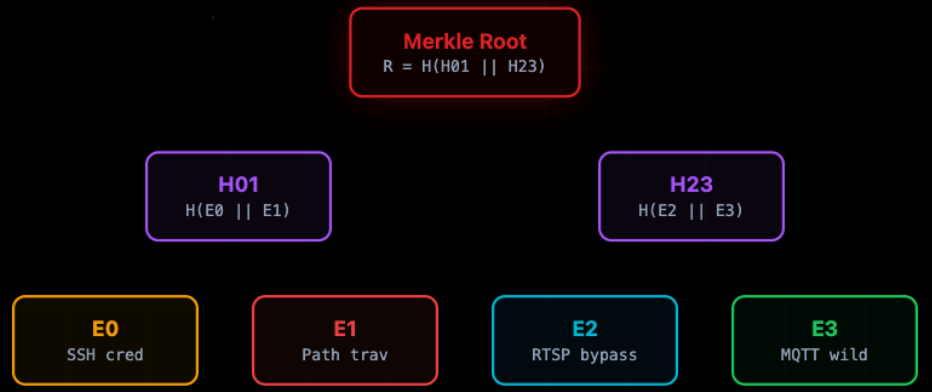
Campaign-level integrity via binary hash tree

CHAPTER TAKEAWAY

Everything permitted. Everything executed. Everything logged

ENRICHMENT VALUE

The fix was network segmentation, not further exploitation



Merkle root is stored in the pentest report. Any single tampered evidence leaf changes the root hash.

Evidence Chain: Worked Example

Five linked evidence artifacts from a single campaign

CHAPTER TAKEAWAY

Every action gets a SHA-256 hash

ENRICHMENT VALUE

Modify entry 3? Entries 4 through N all fail verification

ID	TIME	TARGET	TYPE	DETAIL	SHA-256	SEV
E-001	14:22:03	10.0.0.5:22	CREDENTIAL	admin:admin123 SSH login	a3f4b2c1d8e9...	HIGH
E-002	14:22:18	10.0.0.5	EXFIL	/etc/shadow extracted (3 users)	7c8d9e0f1a2b...	HIGH
E-003	14:23:01	10.0.0.5	ESCALATE	Root via sudo NOPASSWD	d1e2f3a4b5c6...	CRITICAL
E-004	14:24:45	10.0.1.3:554	EXPLOIT	RTSP auth bypass confirmed	b5c6d7e8f9a0...	HIGH
E-005	14:25:12	10.0.1.3:80	EXPLOIT	ONVIF PTZ control verified	f9a0b1c2d3e4...	CRITICAL

Exploit Module Coverage Matrix

21 exploit techniques across 6 protocol modules

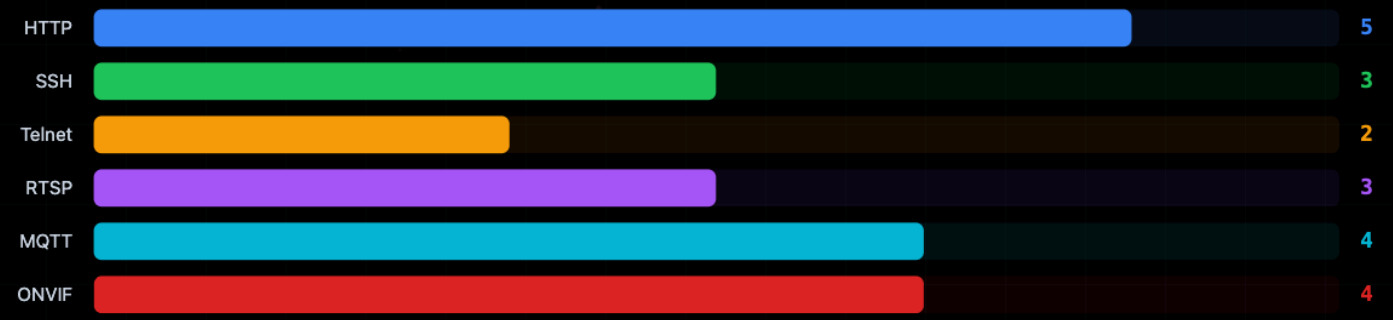
CHAPTER TAKEAWAY

For compact verification: the Merkle root

ENRICHMENT VALUE

Odd number of leaves? Duplicate the last one

EXPLOIT TECHNIQUES PER PROTOCOL



21
Total Exploit Types

6
Protocols Covered

15
Evidence Fields

5/campaign
Hash Chain Links

Evidence Integrity Mechanisms

Five layers ensuring tamper-proof pentest evidence

CHAPTER TAKEAWAY

Penetration testing without written authorization is a crime

ENRICHMENT VALUE

Is unauthorized access. Full stop. Even if the overall engagement was authorized

🔗 **SHA-256 Hash Chains** `Per-evidence`

Each evidence artifact hashes its content plus the previous hash, forming an append-only chain. Modifying any artifact invalidates all subsequent hashes.

🌳 **Merkle Tree Root** `Per-campaign`

All evidence hashes form a binary Merkle tree. The root hash is embedded in the final report. Verifiers can check any leaf in O(log n).

⏰ **Timestamp Attestation** `Per-evidence`

UTC timestamps are included in hash input. Evidence with future or retroactive timestamps fails validation during audit.

📦 **Request/Response Capture** `Per-evidence`

Raw bytes of exploit request and target response are preserved. Reproducibility: same request to same target should yield same response.

✓ **Chain Verification API** `On-demand`

GET /v1/pentest/campaigns/{id}/verify — server recomputes all hashes from stored artifacts and returns pass/fail per link.

SECTION 06

Safety Controller

Guardrails That Make Autonomous Pentesting Deployable

Why Safety Modes Matter

Autonomous pentesting without guardrails is a liability

CHAPTER TAKEAWAY

One of the most reliable real-world techniques

ENRICHMENT VALUE

One password, two compromises, two completely different device types

1. **Production systems at risk**
   Autonomous exploits could crash medical devices, PLCs, or SCADA controllers
2. **Legal and compliance boundaries**
   Unauthorized active exploitation violates regulations and audit requirements
3. **Operator trust gap**
   Security teams will not deploy tools they cannot constrain and observe
4. **Graduated confidence**
   Start passive, prove safety, then progressively unlock capabilities
5. **Forensic accountability**
   Every autonomous action must produce an auditable, tamper-evident log

Simulation Mode

Model everything, touch nothing

CHAPTER TAKEAWAY

The router between agent decisions and protocol-specific execution

ENRICHMENT VALUE

All simulation-only. Deterministic hash-based outcomes

Virtual network model
Cloned attack graph with simulated services

Exploit outcome prediction
Success probability from historical data

Zero packets sent
No network IO, no risk of disruption

Full telemetry
Logs every decision as if it were real

Replay and compare
Run multiple strategies, compare results

RISK PROFILE

0%

Network disruption risk

Use case: training RL agents, strategy validation, compliance demo

Shadow Mode

Observe and recommend -- log what it would exploit

CHAPTER TAKEAWAY

How stealthy is each exploitation technique?

ENRICHMENT VALUE

How stealthy is each exploitation technique?

CAPABILITY MATRIX

- ✓ Port scanning
- ✓ Service version detection
- ✓ Passive OS fingerprinting
- ✓ Banner grabbing (read-only)
- ✗ Exploit execution
- ✗ Credential brute-force
- ✗ Payload delivery
- ✗ Lateral movement

SHADOW OUTPUT

```
> would_exploit: CVE-2023-1234
> target: 10.0.1.42:22
> confidence: 0.87
> expected_impact: shell_access
> action: LOGGED_ONLY
```

Controlled Mode

Active exploitation requires human-in-the-loop approval

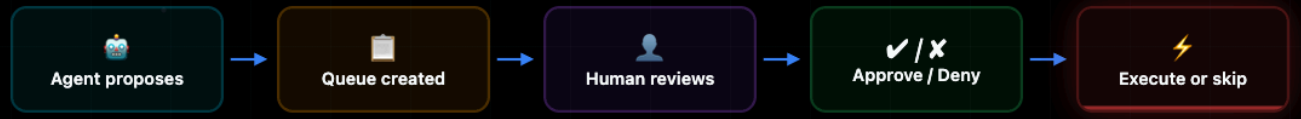
CHAPTER TAKEAWAY

Three report components

ENRICHMENT VALUE

Without that sentence, someone will mistake the report for confirmed exploitation

APPROVAL FLOW



APPROVAL QUEUE

SSH brute-force 10.0.1.42	Medium	PENDING
CVE-2023-1234 exploit 10.0.1.50	High	APPROVED
Lateral pivot via SMB 10.0.2.10	Critical	DENIED

Audit Trail Entry

Every autonomous action produces a tamper-evident log record

CHAPTER TAKEAWAY

Rule-based agent on 22-device sim network

ENRICHMENT VALUE

Use rule-based for compliance. Use PPO for discovery

audit_log.jsonl — JSON

```
{
  "timestamp": "2026-03-04T14:22:07.431Z",
  "campaign_id": "camp_8f3a",
  "action_id": "act_00042",          ← Current safety mode
  "safety_mode": "controlled",       ← Which RL agent acted
  "agent_id": "agent_ppo_v3",
  "action_type": "exploit",
  "target": "10.0.1.42:22",
  "technique": "CVE-2023-1234",
  "decision": {
    "confidence": 0.87,
    "expected_reward": 3.2,
    "policy_entropy": 0.41
  },
  "approval": {
    "required": true,
    "approved_by": "operator@corp.com",   ← Human approval record
    "approved_at": "2026-03-04T14:21:55Z"
  },
  "outcome": {
    "success": true,
    "evidence": "shell_access",
    "duration_ms": 1240
  },                                 ← Hash chain for tamper detection
  "hash_chain": "sha256:a3f8...prev_b7c2"
```

Scope Restrictions

Three dimensions of constraint: target, action, time

CHAPTER TAKEAWAY

The full attack path one more time

ENRICHMENT VALUE

Two point one million dollars approved the next business day

TARGET SCOPE

Allowed CIDRs: 10.0.1.0/24, 10.0.2.0/24

Excluded hosts: 10.0.1.1 (gateway), 10.0.1.254 (DNS)

Max target count: 50 hosts per campaign

ACTION SCOPE

Allowed: port_scan, service_probe, known_cve_exploit

Denied: dos_attack, data_exfil, ransomware_deploy

Max severity: high (no critical unless overridden)

TIME SCOPE

Window: Sat 02:00 – 06:00 UTC (maintenance)

Max duration: 4 hours per campaign

Auto-halt if window expires mid-action

Safety Mode Progression

Worked example: 4-week rollout on a hospital OT network

CHAPTER TAKEAWAY

Prescriptive analytics transforms theoretical risk into confirmed exposure

ENRICHMENT VALUE

We know the attack paths. Now: what happens if we fix them?

Week 1	**Simulation**	Train PPO agent on cloned graph, validate strategy
Week 2	**Shadow**	Deploy alongside manual pentest, compare findings
Week 3	**Controlled**	Agent proposes 12 exploits, operator approves 9
Week 4	**Autonomous**	Full autonomous campaign during maintenance window

14	91%	8/9	23 vulns
Findings (sim)	Shadow match rate	Controlled success	Auto campaign

SECTION 07

RL for Pentesting

Teaching Agents to Think Like Attackers

Why Reinforcement Learning?

Rule-based agents hit a ceiling -- RL agents keep improving

CHAPTER TAKEAWAY

Ten exercises in the Phase 5 lab

ENRICHMENT VALUE

Exercise 10: full campaign with reporting

📋 Rule-Based Agent

- **Fixed decision trees**
  Hand-coded if/else chains for every scenario
- **Cannot generalize**
  New network topology = new rules needed
- **No learning from failure**
  Same mistakes repeated across campaigns
- **Brittle to defenses**
  IDS signature change breaks the playbook

VS

🧠 RL Agent

- **Learned policy network**
  Weights encode strategy from experience
- **Generalizes across topologies**
  Feature vectors abstract away specifics
- **Improves with every campaign**
  Negative rewards steer away from failures
- **Adapts to defenses**
  Policy shifts when old actions get blocked

Exploration vs. Exploitation

The fundamental RL tradeoff applied to penetration testing

CHAPTER TAKEAWAY

A vulnerability report says: "this might be broken"

ENRICHMENT VALUE

See you next week for Digital Twin

Epsilon-Greedy

Random action with probability epsilon, best-known action otherwise

PRO: Simple, guaranteed exploration

CON: Wastes budget on bad actions, no directional exploration

UCB (Upper Confidence Bound)

Pick action that maximizes Q + exploration bonus based on visit count

PRO: Principled exploration, less waste

CON: Assumes stationary environment, hard to scale

PPO (Policy Gradient)

Policy network outputs action probabilities; entropy bonus encourages exploration

PRO: Learns when and where to explore, scales to large action spaces

CON: Requires more computation, needs reward shaping

Key Numbers

Phase 5 -- Autonomous Penetration Testing by the numbers

CHAPTER TAKEAWAY

Can transfer learning let a PPO agent trained on one network generalize to another?

ENRICHMENT VALUE

Can transfer learning let a PPO agent trained on one network generalize to another?

10944 Policy Network Parameters	**6** Protocol Exploit Modules	**12** Action Types in MDP	**5** Safety Verification Layers
79% Host Compromise Rate (PPO)	**8/13** ATT&CK Tactics Covered	**4x** Faster Than Manual Pentest	**0** OT Safety Violations (3 campaigns)

PHASE 5 COMPLETE

Autonomous Penetration Testing

RL-Driven Exploitation with Safety Guarantees

150	18	Phase 6
Slides	Sections	Next

SEAS-8414 · Breakwater Security Platform