



CYBER BATTLE

Simulazione

Reti

Agenti

PENETRATION TESTING & ETHICAL HACKING :

PIATTAFORME PER CYBER BATTLE E NETWORK ATTACK SIMULATION

- Dalla simulazione al mondo reale

Giorgio Colella



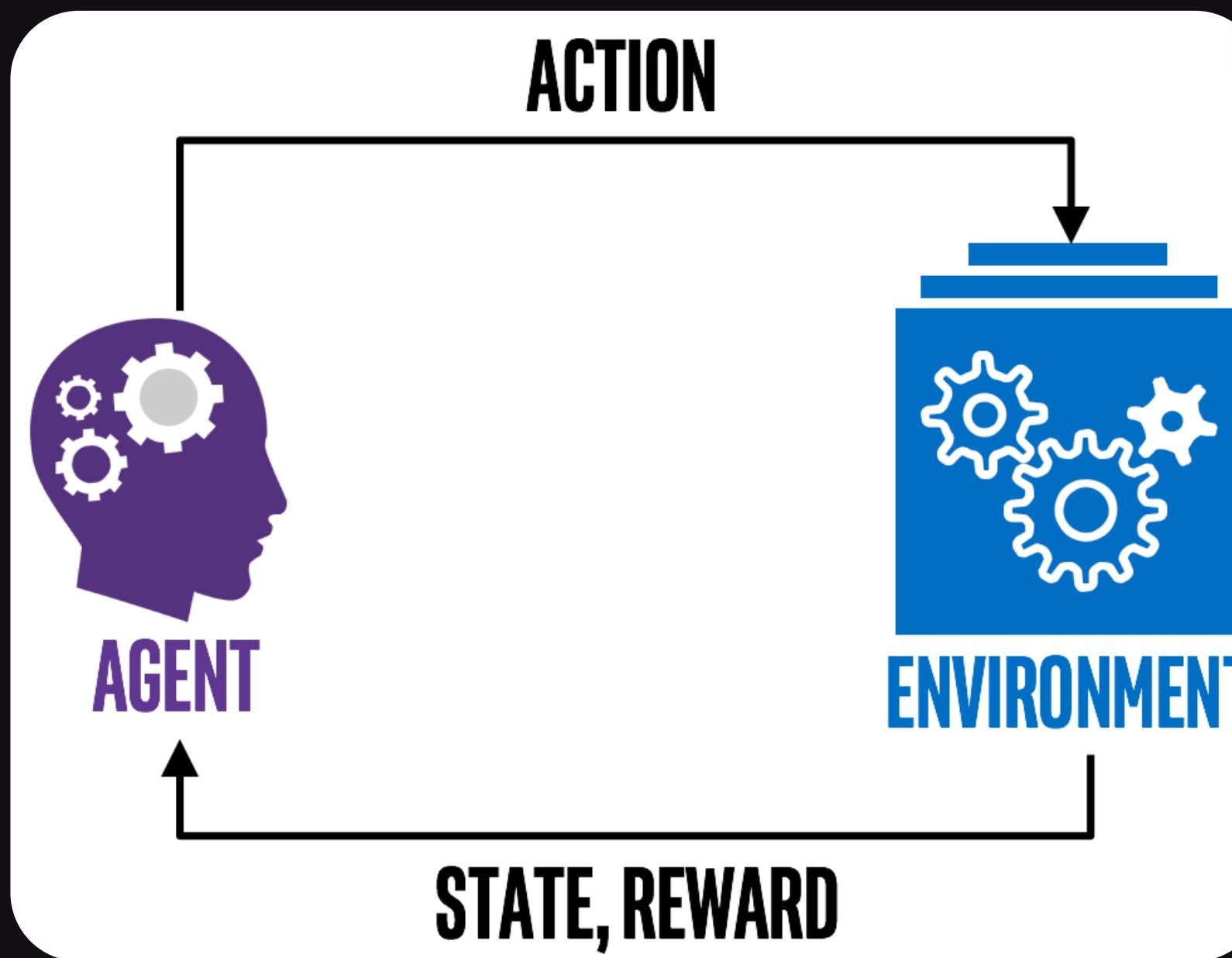


INTRODUZIONE

● Simulazione di Rete

Una simulazione di rete consiste nella costruzione di un modello astratto di un'infrastruttura IT (nodi, servizi, credenziali e regole di comunicazione) in cui è possibile riprodurre in modo controllato le fasi di un attacco informatico: scoperta, sfruttamento di vulnerabilità e spostamenti laterali. Grazie a questo approccio si può studiare il comportamento di un aggressore e valutare l'efficacia di contromisure senza rischiare sistemi reali.





INTRODUZIONE

- **Agenti**

L'attaccante nelle reti simulate è un agente basato su Reinforcement Learning il quale è un metodo di apprendimento automatico in cui un agente impara a prendere decisioni interagendo con un ambiente: ad ogni azione compiuta riceve un feedback numerico (reward) che misura quanto si avvicina all'obiettivo prefissato. Iterando tentativi ed errori, l'agente ottimizza la propria strategia detta policy massimizzando la somma dei reward cumulativi.





Microsoft/ CyberBattleSim

An experimentation and research platform to investigate the interaction of automated agents in an abstract simulated network environments.

8 11

Contributors

● 11

Issues

★ 2k

Stars

🍴 269

Forks

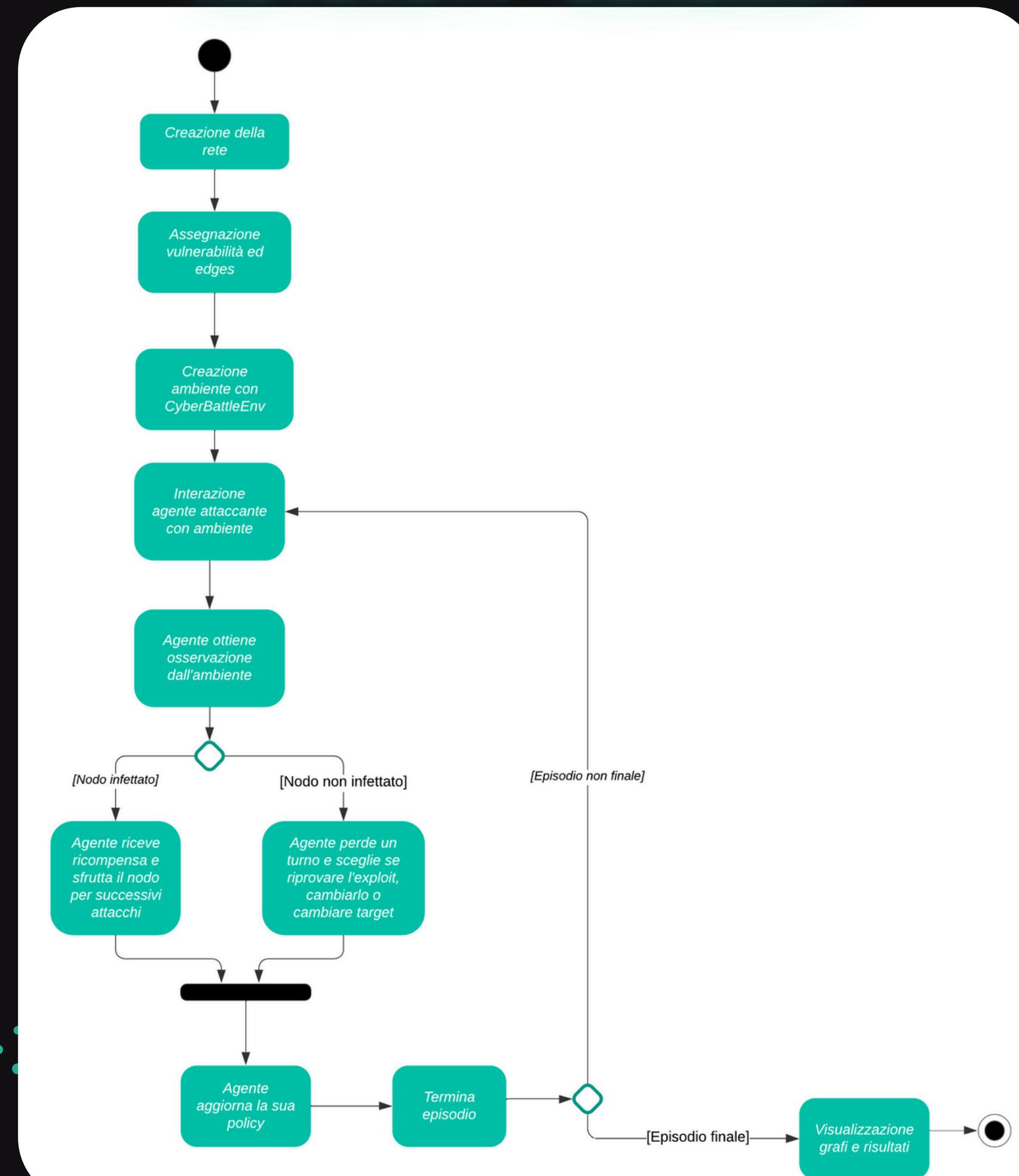
CYBERBATTLESIM

- Primo simulatore di rete analizzato

CyberBattleSim è un framework Python che espone un ambiente di simulazione per studiare attacchi in reti aziendali astratte. La topologia di rete viene definita come un grafo di NodeInfo, ciascuno con servizi, credenziali, firewall e vulnerabilità modellate tramite oggetti VulnerabilityInfo (tipo remoto/locale, costi, precondizioni, esito probabilistico e ricompense). Ad ogni passo l'agente può eseguire azioni di scanning (probe), exploit remoti, movimenti laterali, privilege escalation o estrazione di credenziali, ciascuna implementata come transizione logica con probabilità di successo. Un semplice defender che fa "scan e re-image" può rilevare e ripristinare i nodi compromessi.



ACTIVITY DIAGRAM

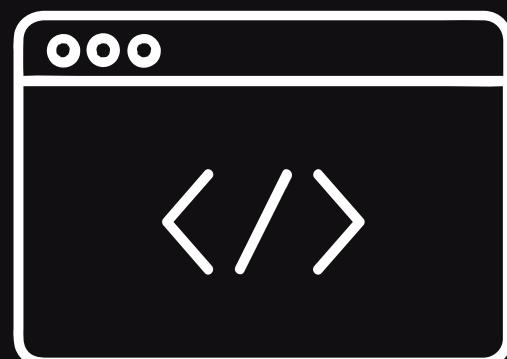




IMPLEMENTAZIONE RETE

● Nodo Rete

```
"ws4": NodeInfo(
    services=[ListeningService("SMB", allowedCredentials=["root:toor"]),
    firewall=FirewallConfiguration(
        incoming=[FirewallRule("SMB", ALLOW)],
        outgoing=default_allow_rules
    ),
    value=1.0,
    properties=["os:Windows", "role:workstation", "service:smb"],
    vulnerabilities={
        "SMBWeakShare": vuln_smb,
        "SMBcreds": vuln_smb_CREDS,
    }
),
```



● Vulnerabilità del Nodo

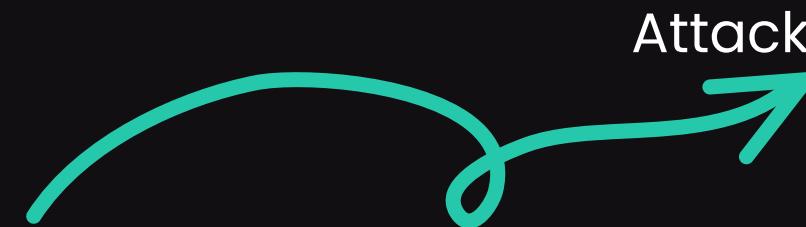
```
vuln_smb_CREDS = VulnerabilityInfo(
    description="SMB share creds leak",
    type=VulnerabilityType.REMOTE,
    outcome=LeakedCredentials(credentials=[
        CachedCredential(node="backup1", port="SMB", credential="root:toor")
    ]),
    precondition=Precondition("service:smb"),
    cost=1.4,
    reward_string="Backup server SMB root password leaked"
)
```



GLI AGENTI RED E BLUE

● Le basi dell'agente attaccante

- LateralMove
- CustomerData
- PrivilegeEscalation
- SystemEscalation
- AdminEscalation
- ProbeSucceeded
- ProbeFailed
- ExploitFailed
- LeakedCredentials
- LeakedNodesId



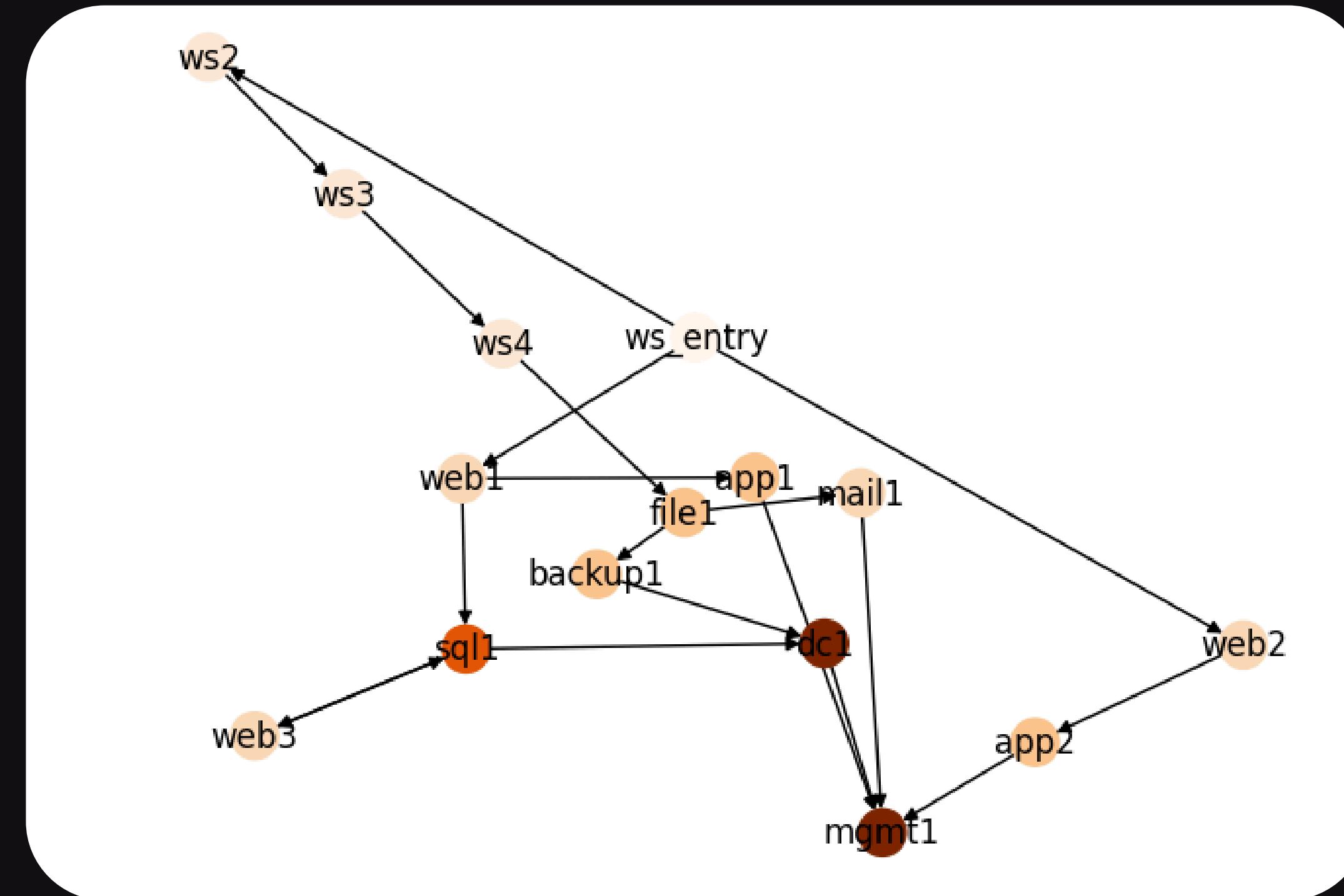
AttackerGoal(own_atleast_percent=1.0)

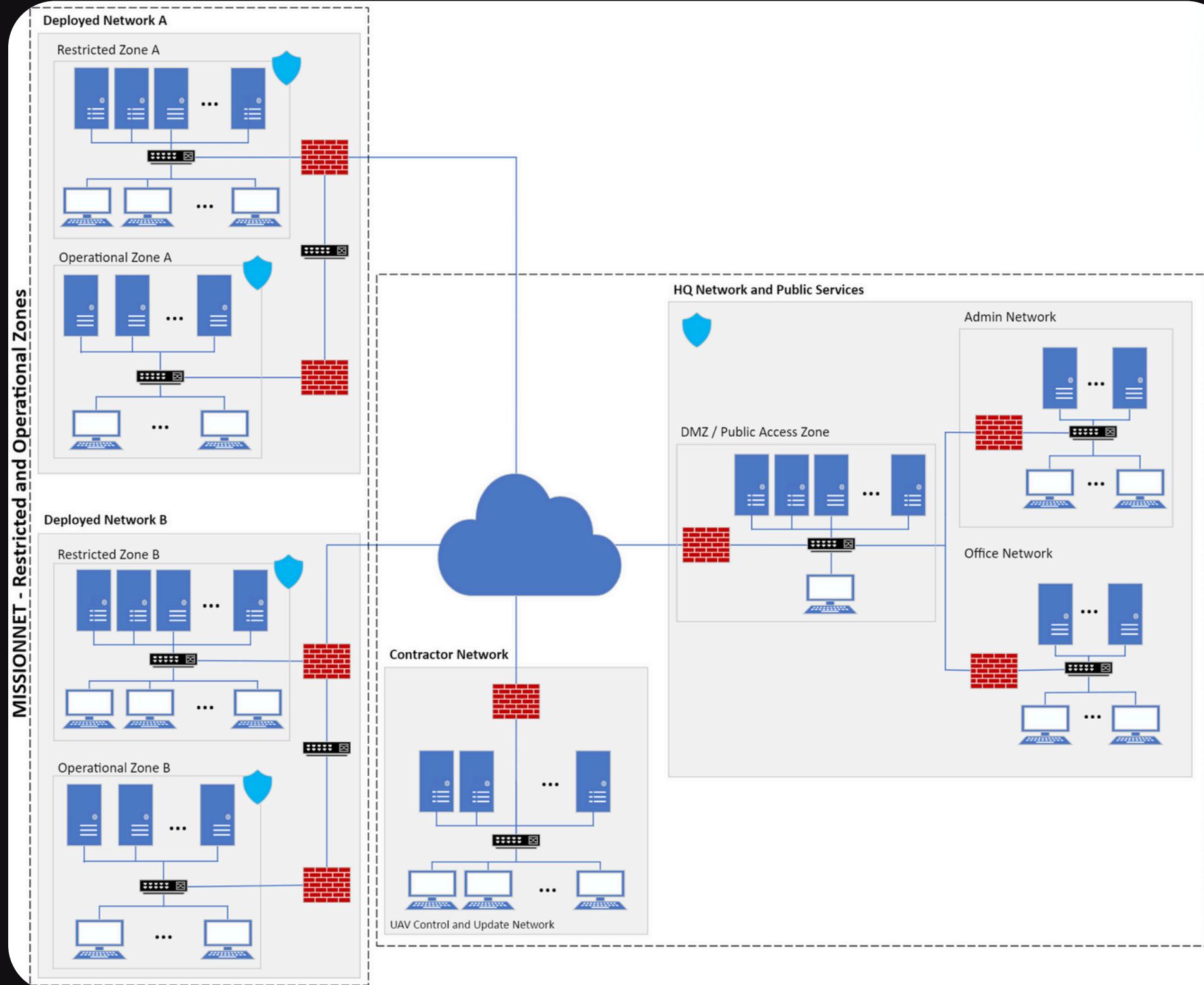
● Le basi dell'agente difensivo

- probability=0.7
- scan capacity=3
- scan frequency=7



TOPOLOGIA RETE CUSTOM





CYBORG CAGE 4

● Secondo simulatore di rete analizzato

CybORG (Cyber Operations Research Gym) è un ambiente di simulazione per lo sviluppo e la valutazione di agenti autonomi in scenari di sicurezza informatica. Nella Challenge 4, CybORG implementa una rete complessa composta da nodi reali (workstation, server, router) suddivisi in zone, ognuna delle quali presenta configurazioni, sistemi operativi e vulnerabilità differenti. Gli agenti (Red, Blue e Green) interagiscono con l'ambiente tramite un'API formale, eseguendo azioni codificate che generano effetti sullo stato dei nodi e sul reward cumulativo. L'agente rosso mira a compromettere macchine, mentre l'agente blu le difende con capacità di scansione e risposta.



Ambiente simulato di Challenge 4



GLI AGENTI RED E BLUE

● *Azioni agente attaccante*

- DiscoverRemoteSystems
- ServiceDiscovery (Aggressive / Stealth)
- ExploitRemoteService
- PrivilegeEscalate
- DiscoverDeception
- DegradeServices
- Impact
- Withdraw

● *Stati agente attaccante*

- K (Known)
- KD (Known + Discovery)
- S (Services)
- SD (Services + Discovery)
- U (User)
- UD (User + Discovery)
- R (Root)
- RD (Root + Discovery)
- F (Final)

● *Le basi dell'agente difensivo*

- Monitor
- Analyse
- DeployDecoy
- Remove
- Restore
- BlockTrafficZone
- AllowTrafficZone





Jjschwartz/ **NetworkAttackSimulator**

An environment for testing AI pentesting agents against a simulated network.

5

Contributors

12

Used by

190

Stars

53

Forks

NASIM

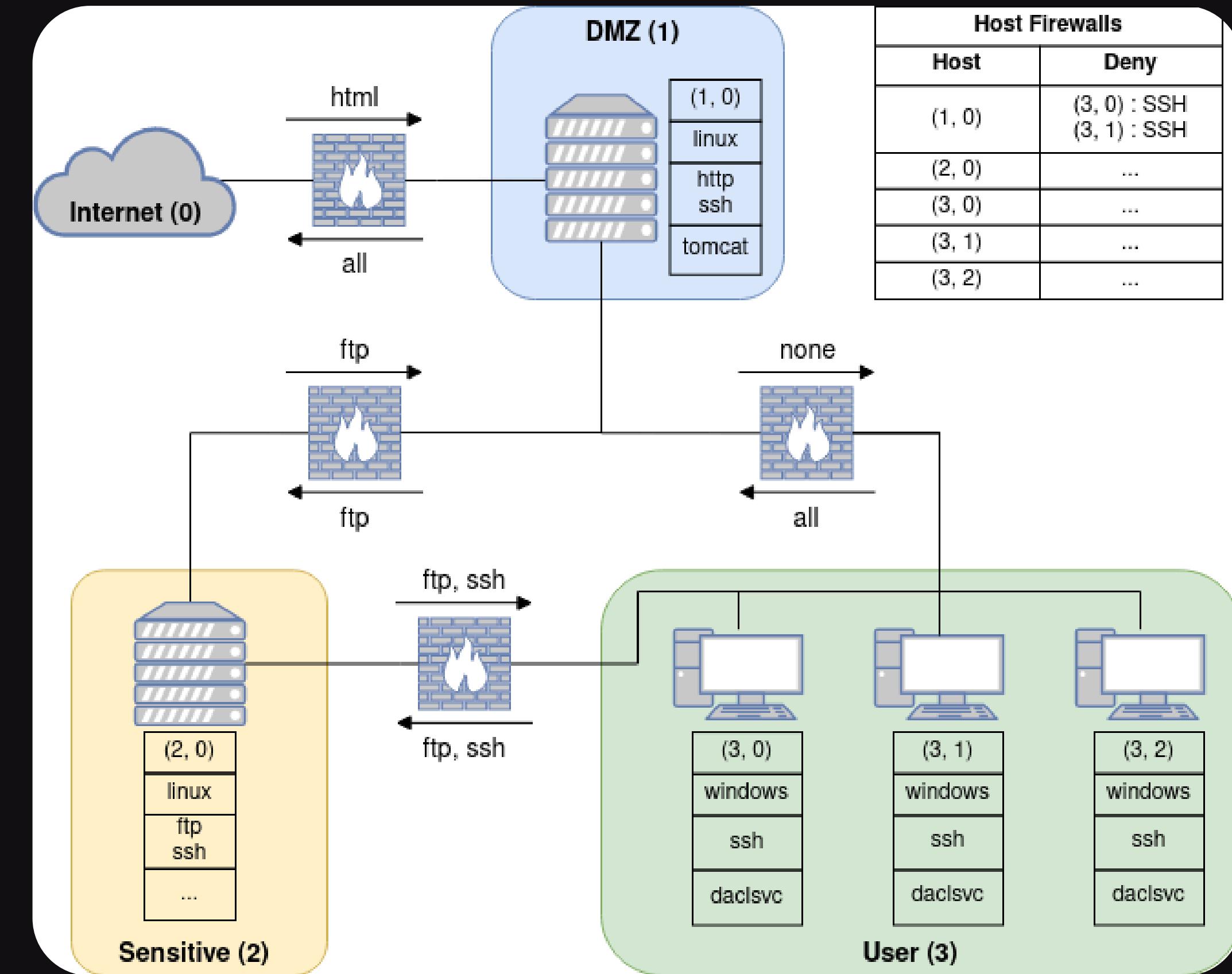
● Terzo simulatore di rete analizzato

Network Attack Simulator (NASim) è un simulatore open source progettato per modellare attacchi informatici su reti informatiche, con l'obiettivo di supportare lo sviluppo e la valutazione di agenti di apprendimento automatico.

Dal punto di vista tecnico, NASim rappresenta la rete come un grafo orientato, in cui i nodi sono host configurabili e gli archi rappresentano le connessioni di rete. Ogni host può essere definito in termini di sistema operativo, servizi attivi, vulnerabilità note, credenziali associate e valore strategico. Gli scenari, completamente personalizzabili, vengono descritti in file YAML dove è possibile definire la topologia della rete, le regole firewall, le informazioni disponibili all'attaccante e le probabilità di successo degli exploit.



TIPICO SCENARIO DI RETE NASIM





GLI AGENTI RED E BLUE

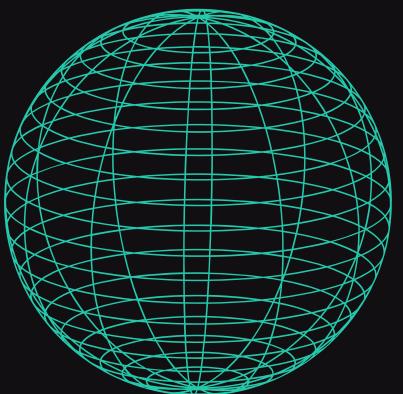
● Le basi dell'agente attaccante

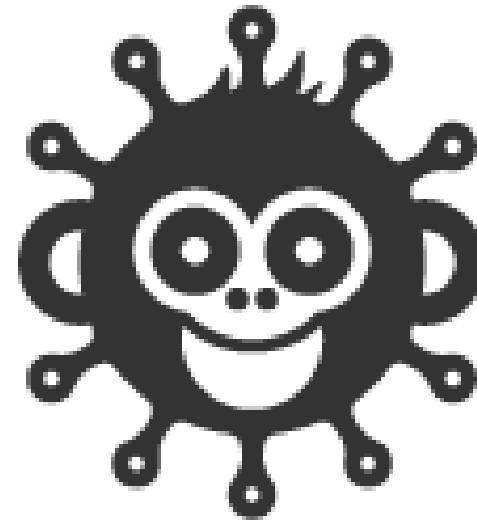
Le implementazioni di agenti fornite con NASim includono:

- keyboard_agent.py
- random_agent.py
- bruteforce_agent.py
- ql_replay_agent.py
- dqn_agent.py

● Le basi dell'agente difensivo

NASIM NON POSSIEDE UN AGENTE DIFENSIVO





Infection Monkey

INFECTION MONKEY

- Attaccante reale analizzato

Infection Monkey è uno strumento open-source sviluppato da Guardicore e progettato per effettuare effettivi attacchi informatici all'interno di reti aziendali al fine di testare la resilienza delle difese. È composto da due componenti principali: Monkey Island, che funge da server di comando e controllo con interfaccia web, e Monkey Agent, un software benigno che agisce come attaccante.

Dal punto di vista tecnico, Infection Monkey è compatibile con sistemi operativi Windows e Linux, ed è in grado di sfruttare exploit reali come SMBExec, WMIEexec, vulnerabilità MSSQL e Hadoop, eseguendoli in modalità controllata e non distruttiva. Il comportamento del Monkey Agent è completamente configurabile e modulare, con plugin attivabili o disattivabili in base al contesto dell'analisi.





MONKEY AGENT

- I plugin disponibili

- Cryptojacker
- Log4Shell
- Mimikatz
- MSSQL
- PowerShell
- Ransomware
- RDP
- SSH
- Zerologon

Exploiters

Configure the exploitation step of the attack

- Enabled exploiters
- Hadoop/YARN Exploiter
- Log4Shell Exploiter
- MSSQL Exploiter
- PowerShell Exploiter
- RDP Exploiter
- SMB Exploiter
- SNMP Exploiter
- SSH Exploiter





CYBER BATTLE

Simulazione

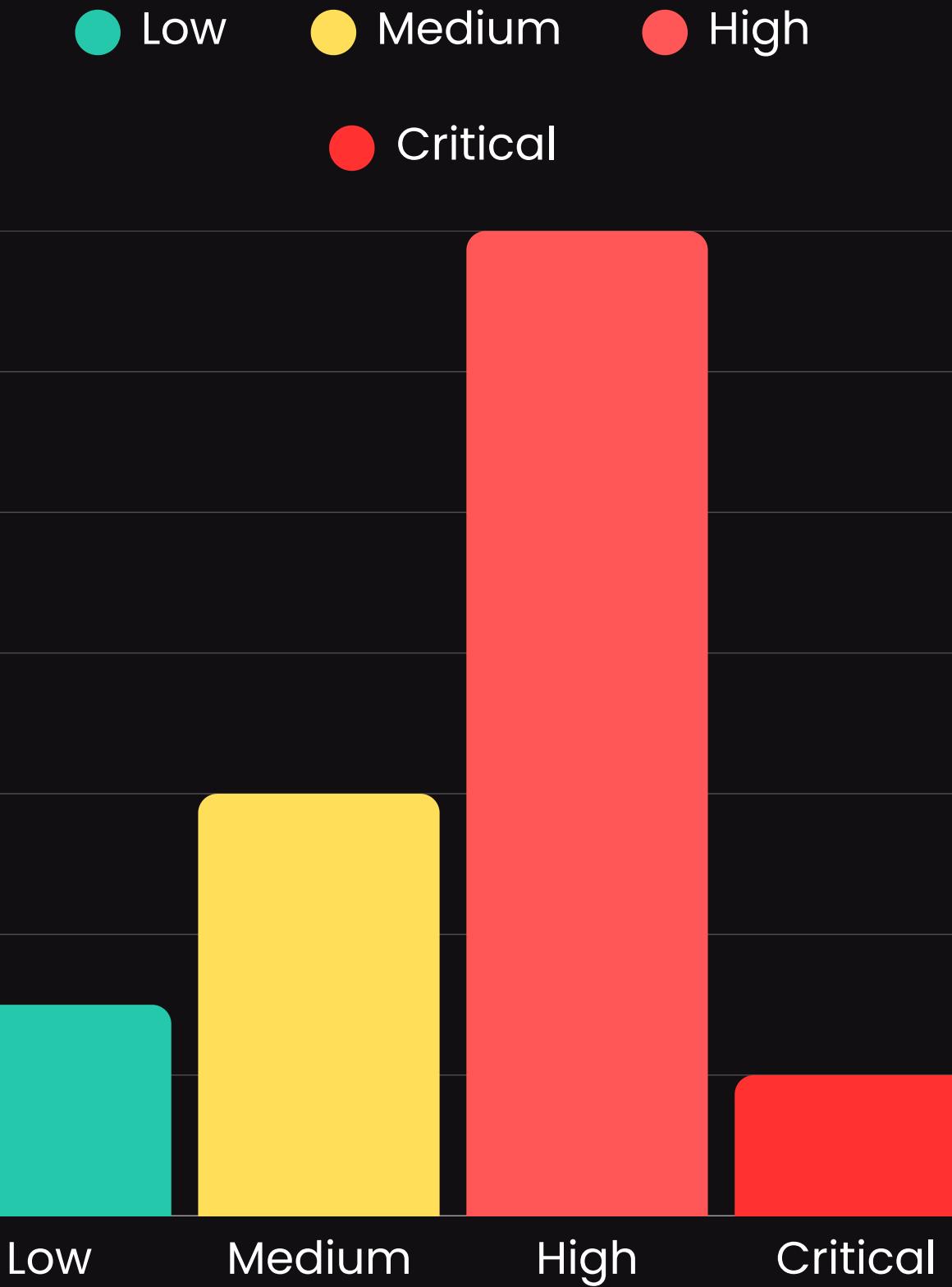
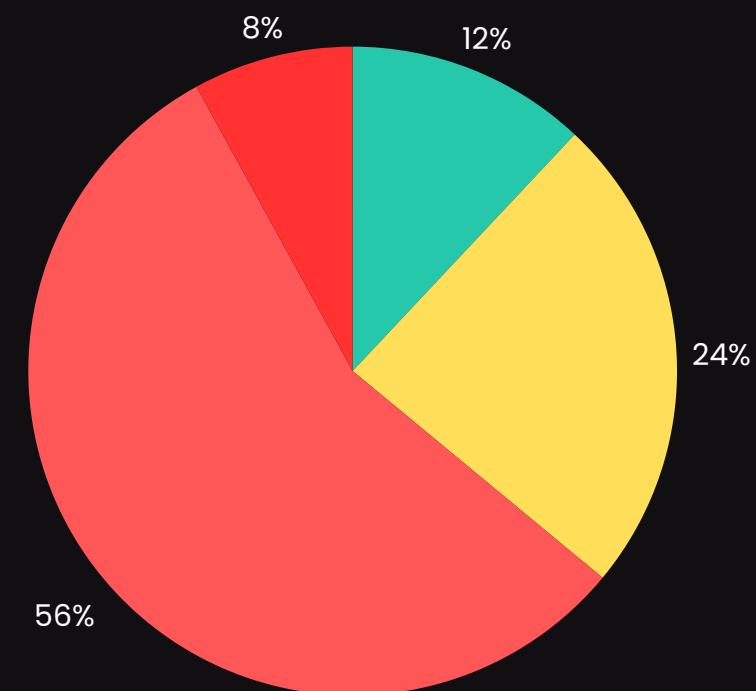
Reti

Agenti

RISULTATI REPORT CYBERBATTLESIM

Approccio

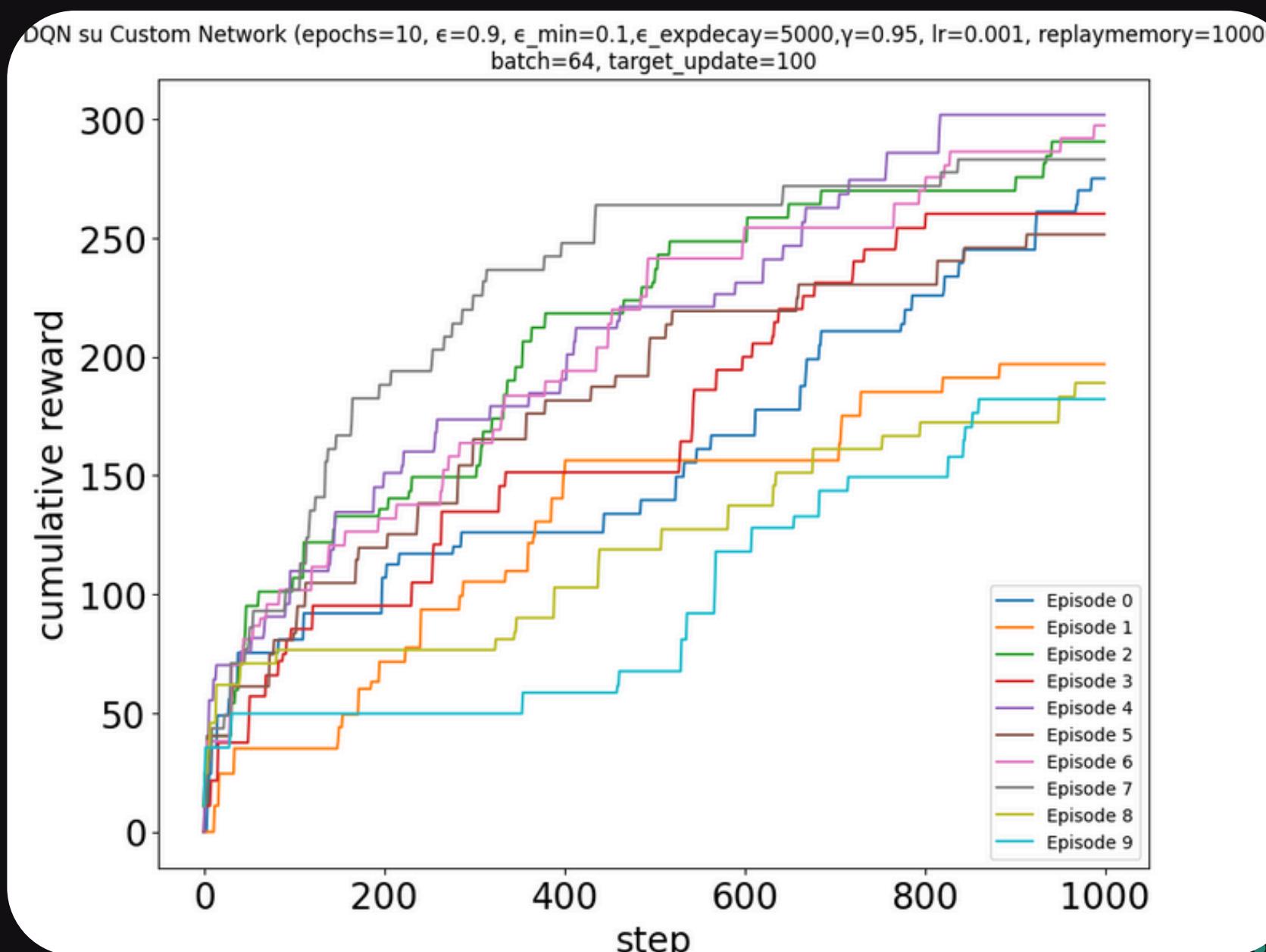
Per la costruzione del Penetration Testing Report si è deciso di analizzare CyberBattleSim all'interno della Custom Network. Alla fine poi, nell'appendice è presente l'effettivo confronto tra i diversi tool.



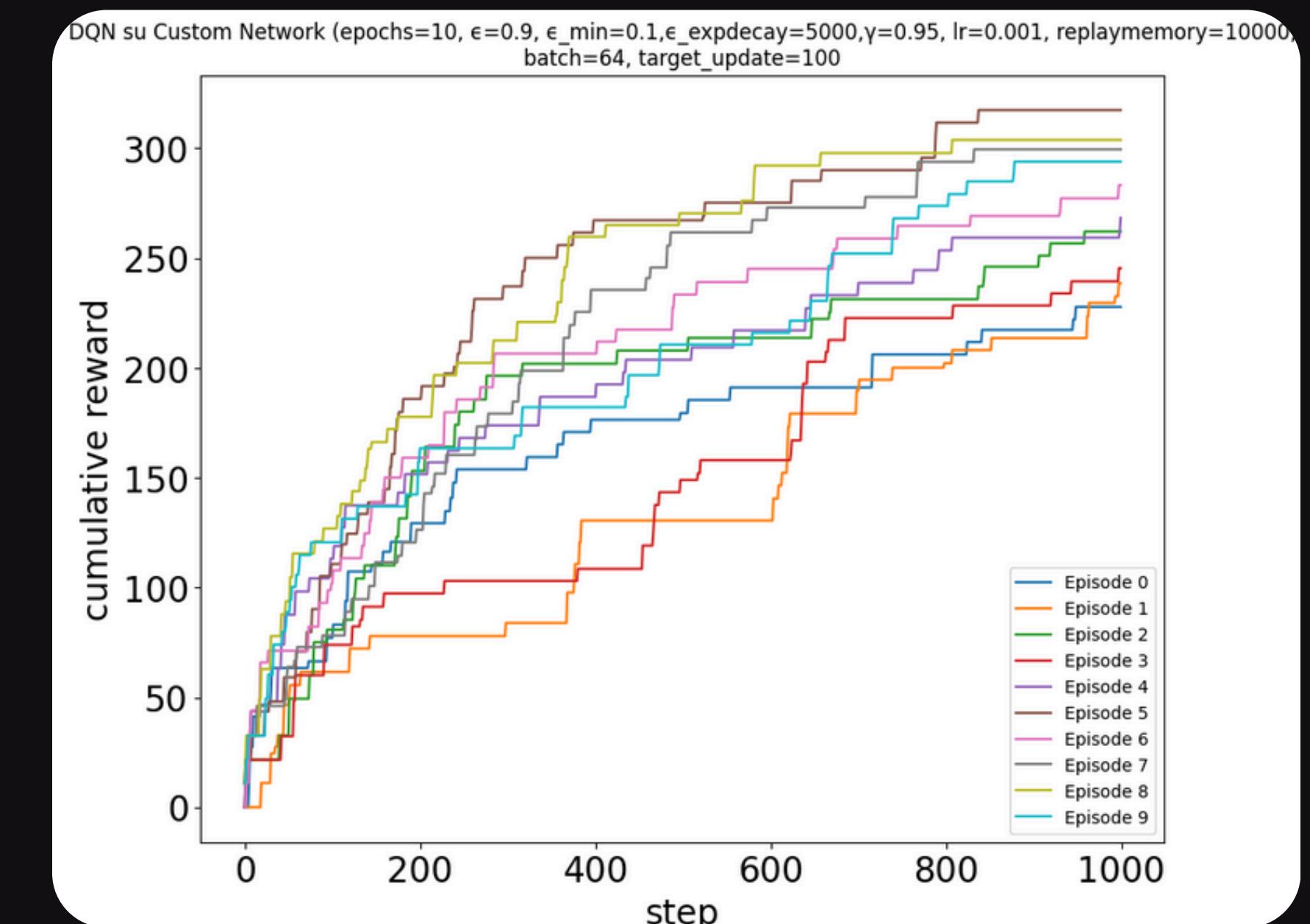


PERFORMANCE AGENTI CYBERBATTLESIM

Agente Red con defender



Agente Red senza defender (infatti caso migliore per Red)





FUNZIONAMENTO AGENTE BLUE CYBORG CAGE 4

- Osservazione che mostra azione Remove di Blue Agent avvenuta con successo

```
state: "R"
[BlueDefenderAgent] get_action called. observation type: <class 'dict'>
observation: {'success': <TernaryEnum.TRUE: 1>, 'action': Remove restricted_zone_a_subnet_router, 'restricted_zone_a_subnet_server_host_0': {'Processes': [{Connections': [{'remote_port': 56114, 'local_address': IPv4Address('10.0.96.199'), 'remote_address': IPv4Address('10.0.240.254')}]}], 'Connections': [{Connections': [{'remote_port': 56114, 'local_address': IPv4Address('10.0.96.199'), 'remote_address': IPv4Address('10.0.240.254')}]}]}, 'Interface': [{ip_address: IPv4Address('10.0.96.199')}], 'System info': {'Hostname': 'restricted_zone_a_subnet_server_host_0', 'OSType': <OperatingSystemType.LINUX: 3>, 'OSDistribution': <OperatingSystemDistribution.UBUNTU: 8>, 'OSVersion': <OperatingSystemVersion.UNKNOWN: 1>, 'Architecture': <Architecture.x64: 2>, 'position': array([0., 0.])}}}
[BlueDefenderAgent] get_action called. observation type: <class 'dict'>
observation: {'success': <TernaryEnum.TRUE: 1>, 'action': Remove operational_zone_a_subnet_router}
[BlueDefenderAgent] get_action called. observation type: <class 'dict'>
observation: {'success': <TernaryEnum.TRUE: 1>, 'action': Remove restricted_zone_b_subnet_router}
[BlueDefenderAgent] get_action called. observation type: <class 'dict'>
observation: {'success': <TernaryEnum.TRUE: 1>, 'action': Remove operational_zone_b_subnet_router}
[BlueDefenderAgent] get_action called. observation type: <class 'dict'>
observation: {'success': <TernaryEnum.TRUE: 1>, 'action': Remove public_access_zone_subnet_router}
```





FUNZIONAMENTO AGENTE RED CYBORG CAGE 4

- Osservazione che mostra azione di Impact (senza successo) e di Privilege Escalation (successo) di Red Agent

```
** Turn 4 for red_agent_0 **
Action: Impact contractor_network_subnet_user_host_9
Action Success: FALSE

Observation:
{'contractor_network_subnet_user_host_9': {'Interface': [{'Subnet': IPv4Network('10.0.109.0/24'),
   'ip_address': IPv4Address('10.0.109.41')}], 'Sessions': [{Type: <SessionType.RED_ABSTRACT_SESSION: 10>,
   'agent': 'red_agent_0',
   'session_id': 0,
   'username': 'root'}], 'System info': {'Hostname': 'contractor_network_subnet_user_host_9'}}}

Host States:
{'10.0.109.41': {'hostname': 'contractor_network_subnet_user_host_9',
   'state': 'R'}}

** Turn 2 for red_agent_3 **
Action: PrivilegeEscalate restricted_zone_b_subnet_user_host_1
Action Success: TRUE

Observation:
{'restricted_zone_b_subnet_user_host_1': {'Interface': [{'Subnet': IPv4Network('10.0.96.0/24'),
   'ip_address': IPv4Address('10.0.96.138')}], 'Sessions': [{Type: <SessionType.RED_ABSTRACT_SESSION: 10>,
   'agent': 'red_agent_3',
   'session_id': 0,
   'username': 'root'}], 'System info': {'Hostname': 'restricted_zone_b_subnet_user_host_1'}}}
```

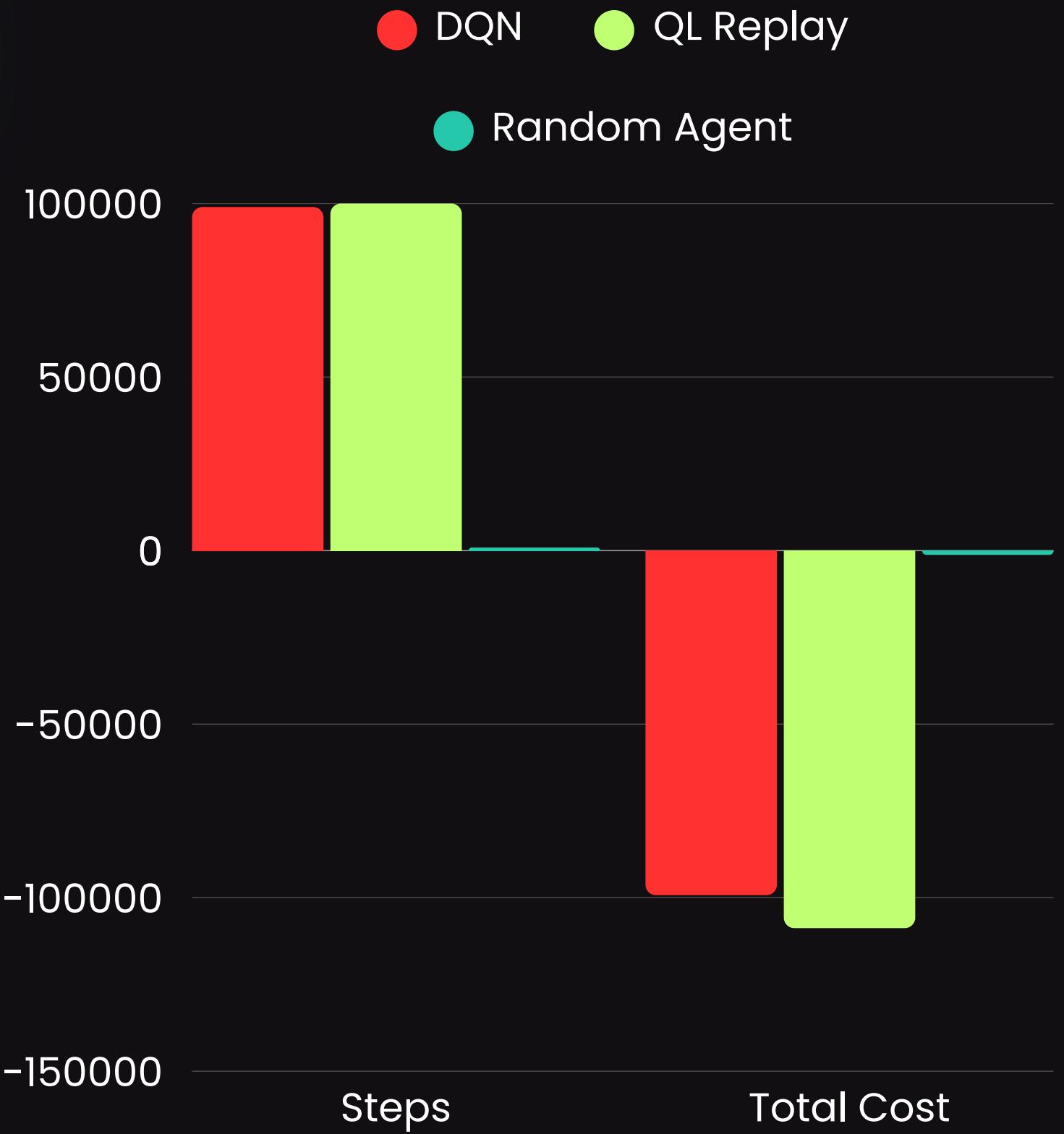


RISULTATI AGENTI NASIM

Approccio

In questo caso sono stati testati DQN, QL with Replay Memory e Random Agent. Il parametro steps indica quanti steps() sono stati necessari per vincere e total cost indica il costo total di tutte le attività di scansione e attacco effettuate (non è la reward).

DQN e Random Agent sono riusciti a vincere ma Random Agent con una quantità molto minore di step() testimoniano che DQN abbia bisogno di un miglior training e fine-tuning. QL non è riuscito a vincere perchè, usando le Q-Tables il suo training è estremamente lento e quindi per ragioni di tempo ed insufficienti capacità computazionali è stato ridotto a pochi steps.





RISULTATI INFECTION MONKEY

Il test con Infection Monkey è stato effettuato su una rete LAN con due VM vulnerabili by design (Stapler1 e Metasploitable2) in una Nat Network. La scansione è stata eseguita da una VM Ubuntu 18.04 connessa alla NAT Network. Nonostante la NAT Network ben configurata, Infection Monkey è riuscito a vedere tutti i dispositivi della LAN testimoniando la presenza di un potente tool di scansione.

Machine-related Recommendations

- 10.0.2.7 (10.0.2.7)

1. Change user passwords to complex one-use passwords that are not shared with other computers on the network. Protect private keys with a pass phrase.
The machine is vulnerable to an SSH attack. An Infection Monkey Agent authenticated over the SSH protocol using stolen/configured credentials.

[Read More...](#)

Attempts a brute-force attack against SSH using known credentials, including SSH keys.

- 10.0.2.5 (10.0.2.5)

1. Change user passwords to complex one-use passwords that are not shared with other computers on the network. Protect private keys with a pass phrase.
The machine is vulnerable to an SSH attack. An Infection Monkey Agent authenticated over the SSH protocol using stolen/configured credentials.

[Read More...](#)

Attempts a brute-force attack against SSH using known credentials, including SSH keys.

The Network from Infection Monkey's Eyes

Infection Monkey discovered 5 machines and successfully breached 2 of them.

40% of scanned machines exploited

Infection Monkey discovered 9 open services on 5 machines:

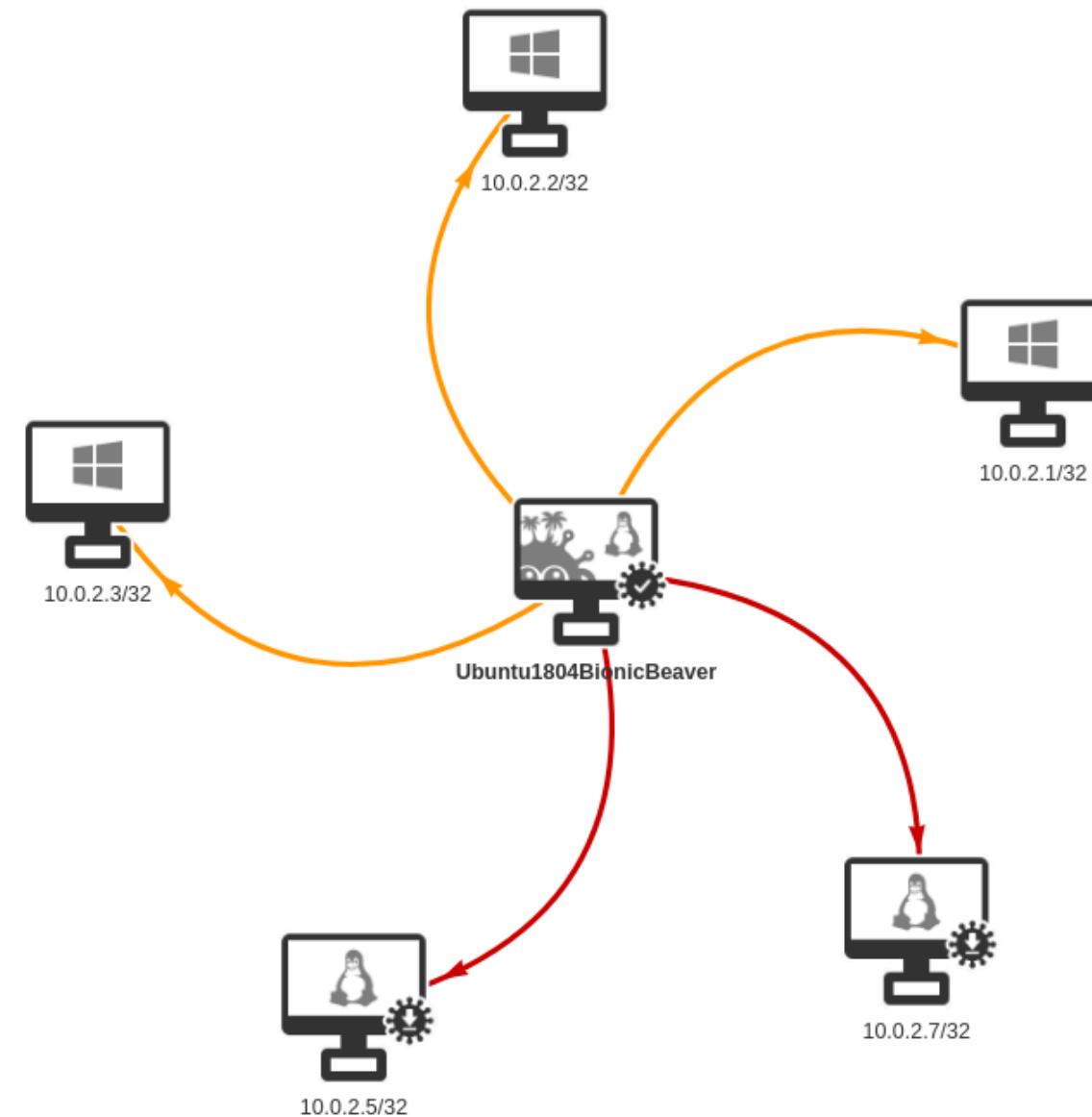
Scanned Servers	
Machine	Services found
(10.0.2.1/32)	
(10.0.2.3/32)	
(10.0.2.2/32)	10.0.2.2:135 - unknown 10.0.2.2:445 - unknown 10.0.2.7:22 - ssh 10.0.2.7:80 - http 10.0.2.7:445 - smb 10.0.2.7:3306 - unknown
(10.0.2.7/32)	

Infection Monkey successfully breached 2 machines:

Breached Servers		
Machine	IP Addresses	Exploits
10.0.2.7	10.0.2.7	SSH Exploiter
10.0.2.5	10.0.2.5	SSH Exploiter

2. Infection Map

Legend: Exploit — | Scan — | Tunnel — | Island Communication —





CONCLUSIONI

Il confronto tra CyberBattleSim, NASim, CybORG e Infection Monkey evidenzia come ogni tool presenti vantaggi specifici:
NASim è il più semplice da usare e personalizzare, ideale per chi vuole creare rapidamente scenari realistici ma gestibili.
CyberBattleSim è astratto ma utile per prototipazione rapida e analisi di algoritmi, pur richiedendo più interventi manuali.
CybORG è il più realistico e completo, supporta attacco-difesa su infrastrutture reali ma richiede competenze avanzate e risorse.
Infection Monkey, invece, funziona su ambienti reali con una buona scansione di rete, ma offre funzionalità di attacco limitate, risultando adatto solo a contesti veloci o preliminari, meno a test professionali strutturati.





CYBER BATTLE

**GRAZIE PER
L'ATTENZIONE**

