Applying Reinforcement Learning and Genetic Algorithms in Game-Theoretic Cyber-Security

Learning to Penetrate Networks

ȘTEFAN P. NICULAE







AFFILIATION



Experimental Research Unit, <u>Bitdefender</u>



Master Thesis Supervisor: M. Popescu, FMI UB



Research Project Supervisors: T. Bäck & K. Yang, LIACS

CONTENTS

- 1. Introduction
- 2. Prior Approaches
- 3. Game Definition
- 4. Techniques
- 5. Results
- 6. Conclusions
- 7. Future Work



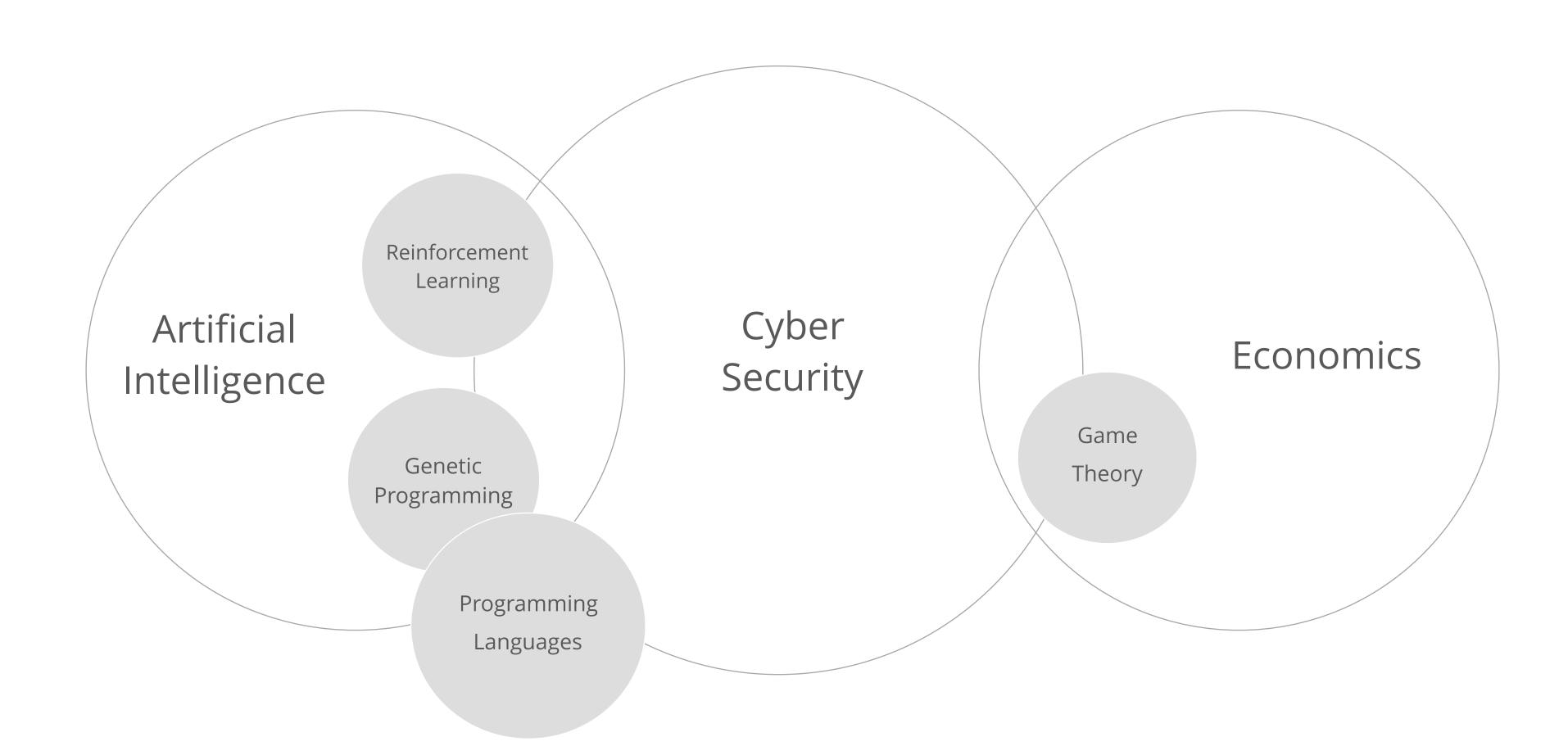
MOTIVATION

- Security solutions (especially AI) need to be validated
 - frequently, on demand
 - repeatably, in large amounts
- Penetration testing (pentesting) = attack your own system in order to reveal its security flaws
- Imperfect solution 1 manual pentesting:
 - slow, constrained by human interaction
 - runs not guaranteed to be identical, not always auditable
- Imperfect solution 2 automated pentesting:
 - next to no frameworks exist
 - existing ones offer severely ineffective attacks
- Solution: automate pentesting using a learning-based approach

OBJECTIVE

- Given a network of machines
- Find its weaknesses
 - faster than randomly trying available exploits
 - comparable in strength to a real attacker
- Approach: model pentesting as a game and learn strategies to win it
- Ultimate goal: allow efficient evaluation (an essential development step) of robust security solutions

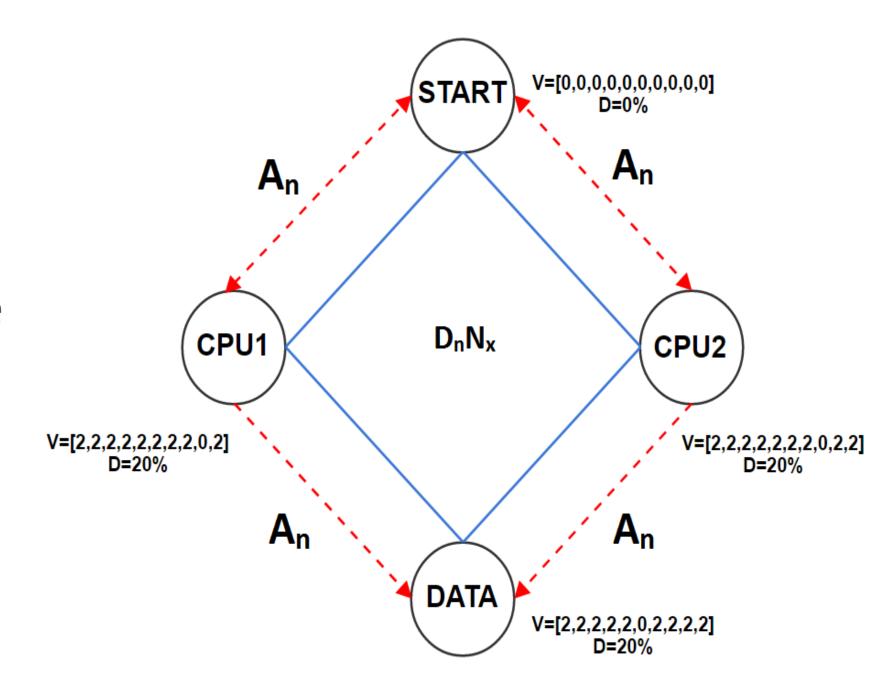
DOMAIN INTERSECTION





THE SIMPLE APPROACH

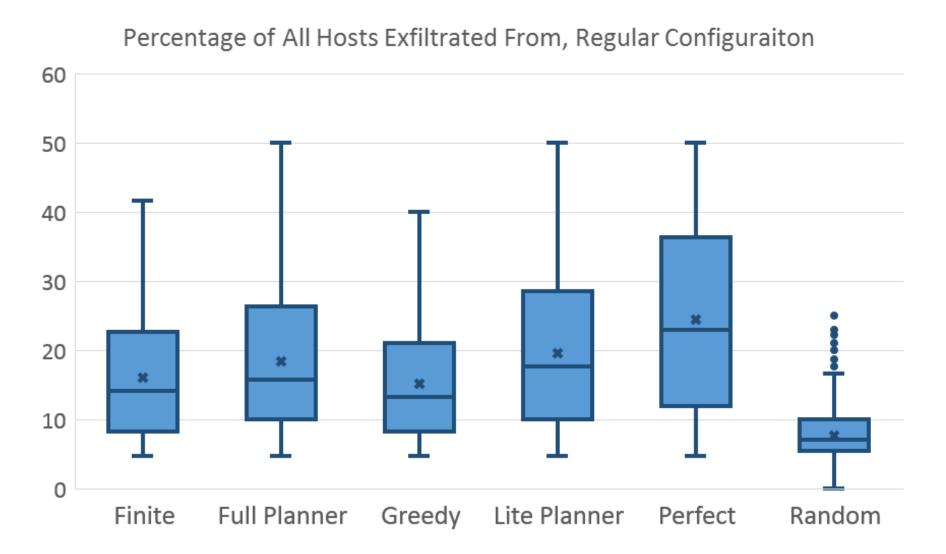
- Only 4 nodes, of which 1 is start and one is target
- Each node has a tuple of integers
- Attacker & defender can increment one value at a time
- An attack on a node is successful if attk_{val} > def_{val}
- Agents have no knowledge of the other's allocation



source: [5]

THE MITRE APPROACH

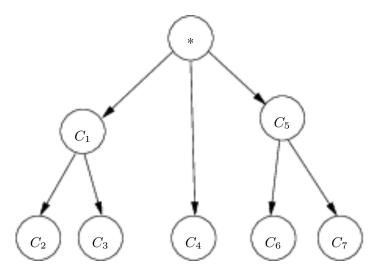
- Reconnaissance, exploit and cautionary actions
- Probabilistic attacker action success
- Include neutral user interaction
- Dynamic machine connections
- Addition/patching of vulnerabilities
- Both fixed-strategy and adaptive algorithms



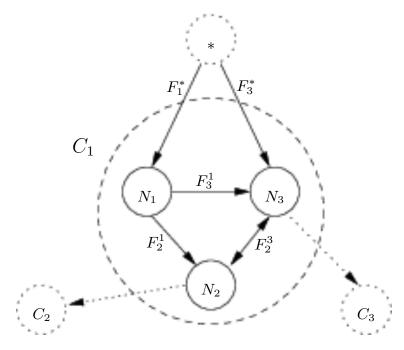
source: [6]

THE CORE APPROACH

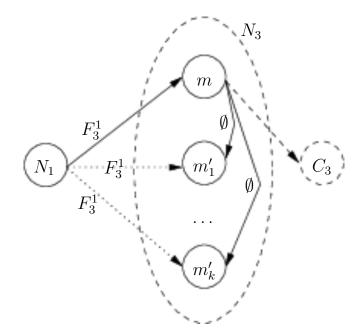
- Asymmetrical machine connections
- Target larger topologies, machine clusters
- Abstraction levels for "network levels"
- Attacker actions are limited to scans and a homogenous list of exploits
- Defender not modeled



(a) LN as tree of components C.



(b) Paths for attacking C_1 .



(c) Attacking N_3 from N_1 , using m first.

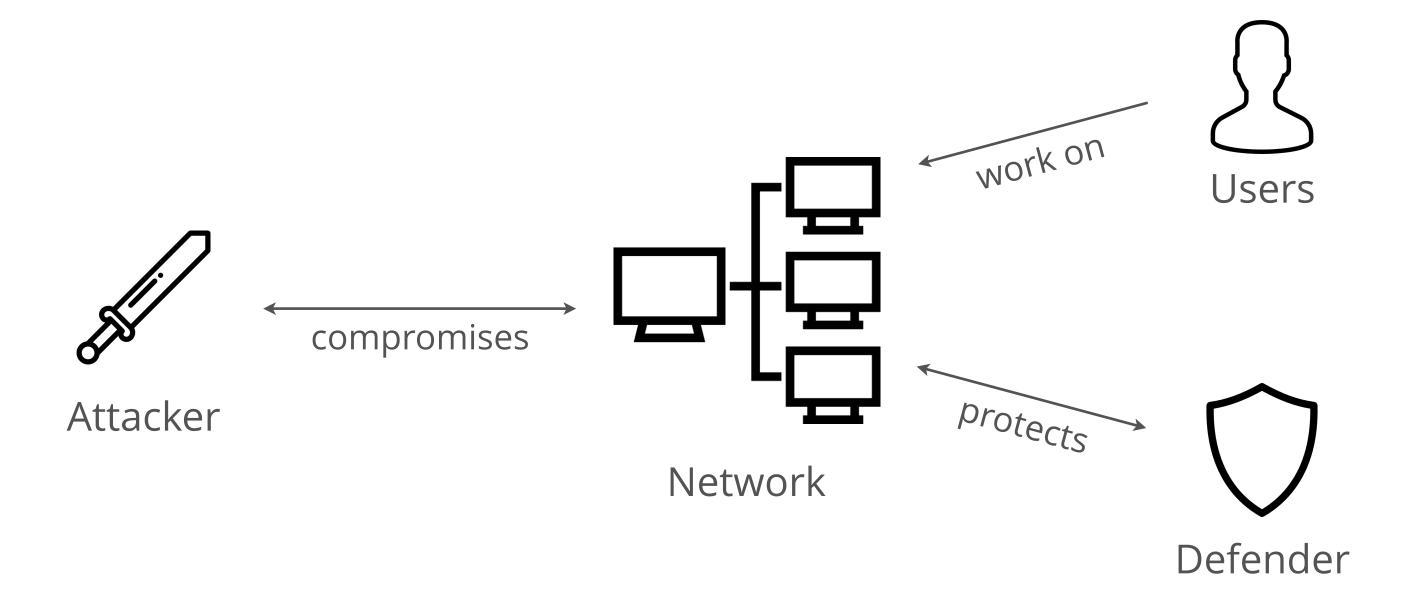
source: [7]



FORMALIZATION

- Model as a game between an attacker and a defender
- High-level abstraction
 - As close as possible to the real world
 - But still feasible to implement
- Described as a Partially Observable Markov Decision Process
- Outcome = the attacker's strategy: what action to pick, in a given environment state

GAME ACTORS



H NETWORK

- Models an enterprise network
- Star topology most common
- Each machine has some local admins
- Each machine has a number of vulnerabilities (governed by the type of user)
- Connections are bi-directional and static

S GREY AGENT

- Models noise generated by a normal users
- Has no objective, performs benign activity
- Probabilistically performs one of:
 - reboot a machine
 - log in to other machines
 - add vulnerabilities to a machine

ATTACKER

- Models a penetration tester
- (who mimics a malicious infiltrator)
- Follows a specific goal
 - exfiltrate some piece of sensitive data
 - gain as wide foothold as possible
 - depends on the scenario
- Actions modeled after Mitre's ATT&CK classifications



enumerate to reveal connections

scan machine vulnerabilities

RECONNAISSANCE

exploit a discovered vulnerability

migrate to another machine

login using dumped creds

GAINING FOOTHOLD

escalate session

persist against reboots & detection

dump creds of local admins

exfiltrate data on the machine

cleanup to not get found later

Post-Exploit

ATTACKER ACTIONS

wait to let defender cool down

evade next action will be stealthier

abandon when payoff < risk

Non-Targeted

ATTACKER ACTIONS COSTS

- reliability: probability of success
- duration: time steps taken
- noise: chance of detection
- reward: positive or negative
- crash chance: lose foothold, takes longer (only for exploits)

DEFENDER

- Models counteractions performed by:
 - ▶ an anti-virus solution (automatically) or
 - a security officer (manually)
- Aims to counter the attacker

Investigate	Instantly Detect	Prevent
past actions	attacker action	future actions time

DEFENSE MEASURES

- Instant Detection
 - based on action's noise
 - fended off by evasive maneuvers
 - kicks attacker off, warrants more attention
- Investigation
 - based on defender's suspicion
 - increases with each action, decreases with time
 - kicks attacker off, patches vulnerabilities

DEFENSE MEASURES

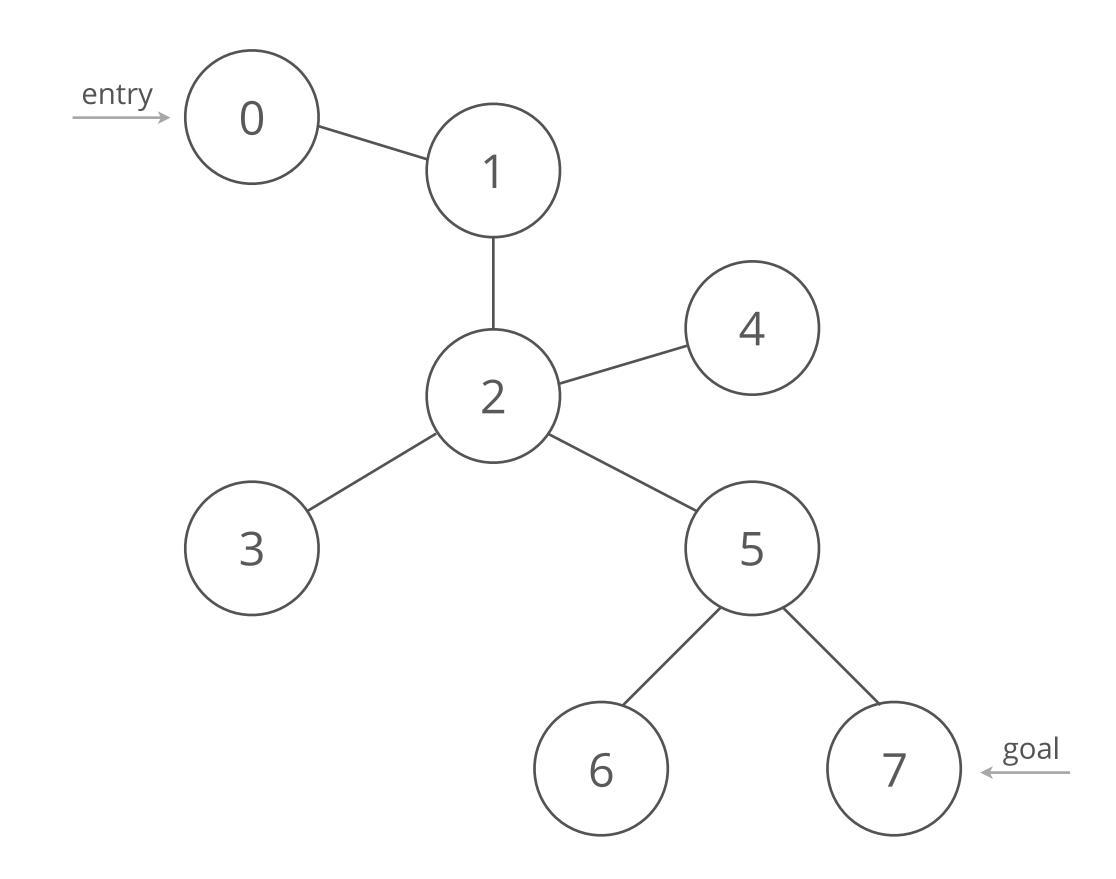
- Prevention
 - has a number of "attention resources"
 - allocates to key places
 - one resource blocks one action on a machine
- Protect those places that are valuable & often targeted
- Akin to a Security Game coming from Game Theory

GAME RULES

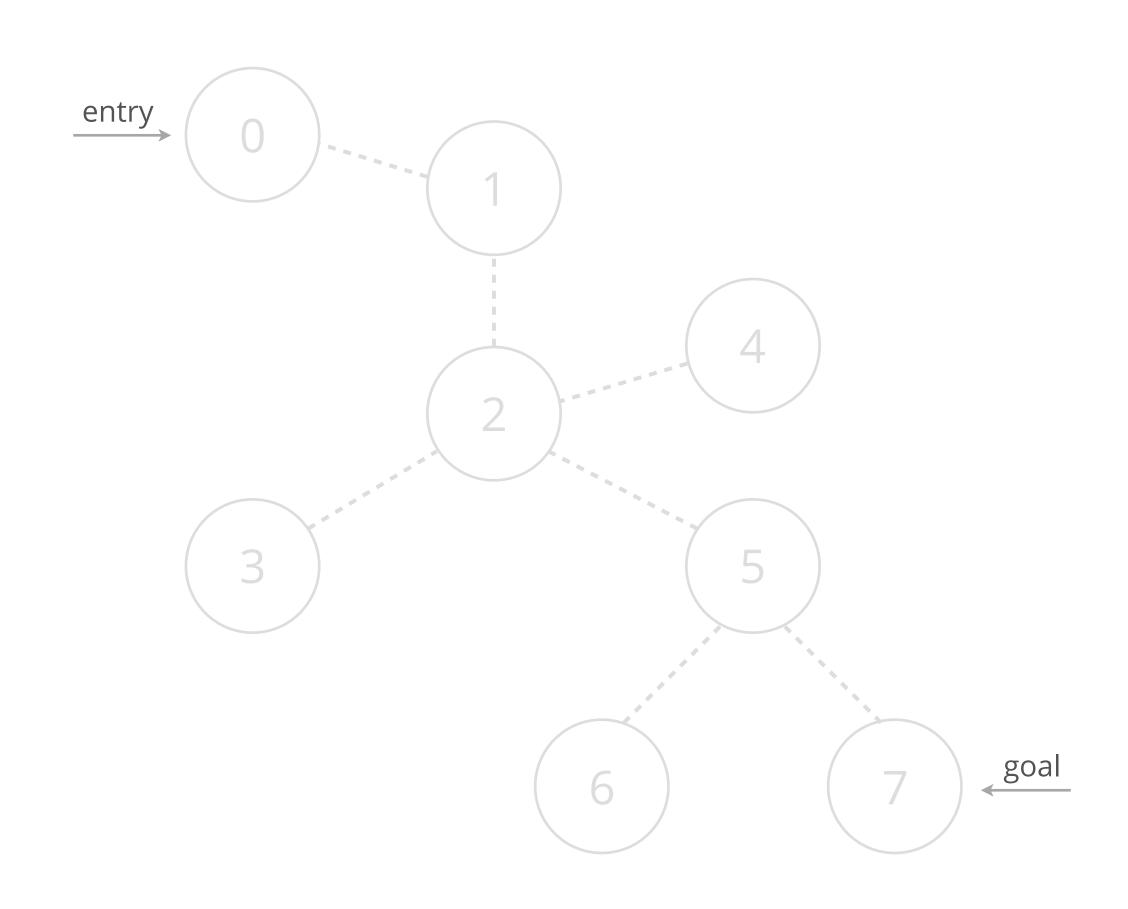
- Turn-based
- Actors act concomitantly
- End conditions:
 - Data exfiltrated from goal machine
 - Attacker gave up
 - ▶ Time limit reached
 - No more moves available

EXAMPLE RUN

- Best way to understand the task
- Manually perform steps
- On a simple network
- From start, to episode completion



action durationreward receivedfailure reason



CONNECTION

- --- undiscovered
- discovered

MACHINE STATUS

- unknown
- scanned
- foothold
- elevated

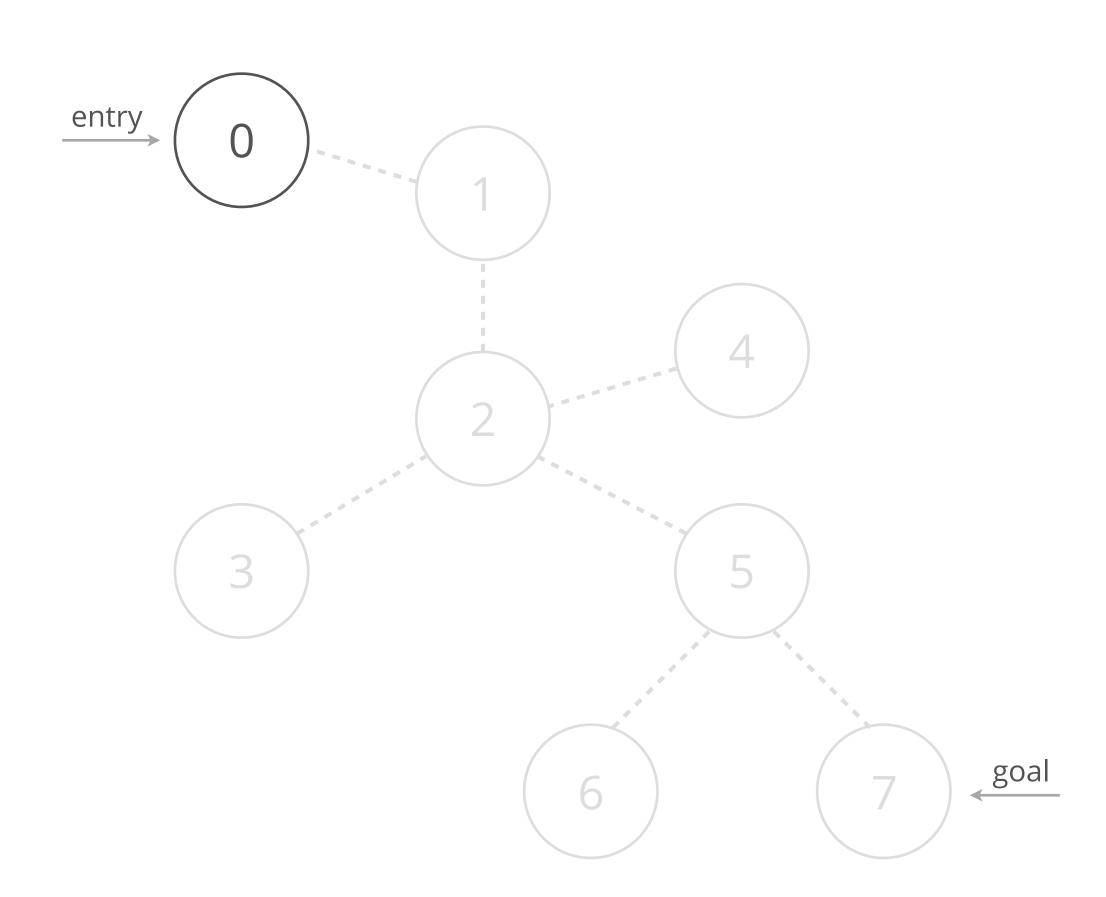
PERFORMED

- cleanup
- persistence
- Q dump
- exfiltrate

ACTION

Q scan: 0

4 🗵



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

persistence

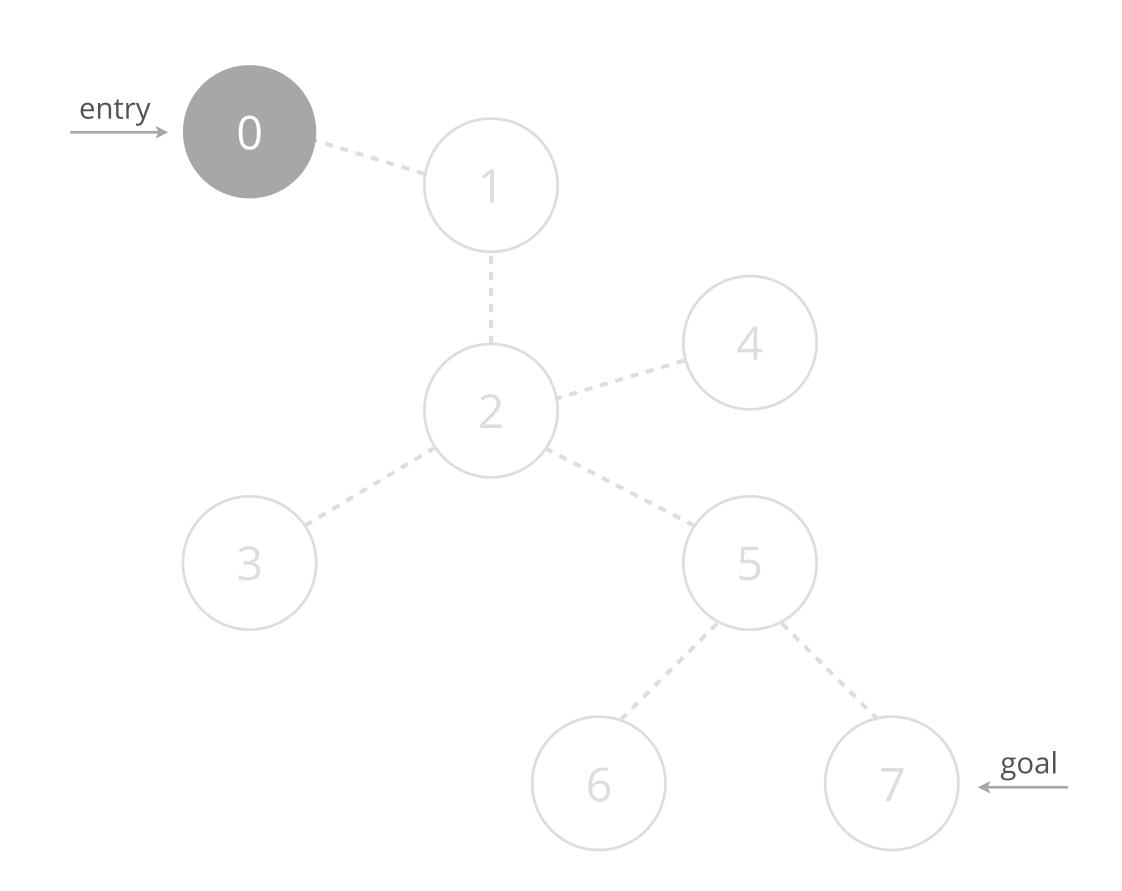
Q dump

ACTION

Sexploit A: 0

2 🗵

+5 ₩



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

persistence

Q dump

CONNECTION

--- undiscovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

Q dump

exfiltrate

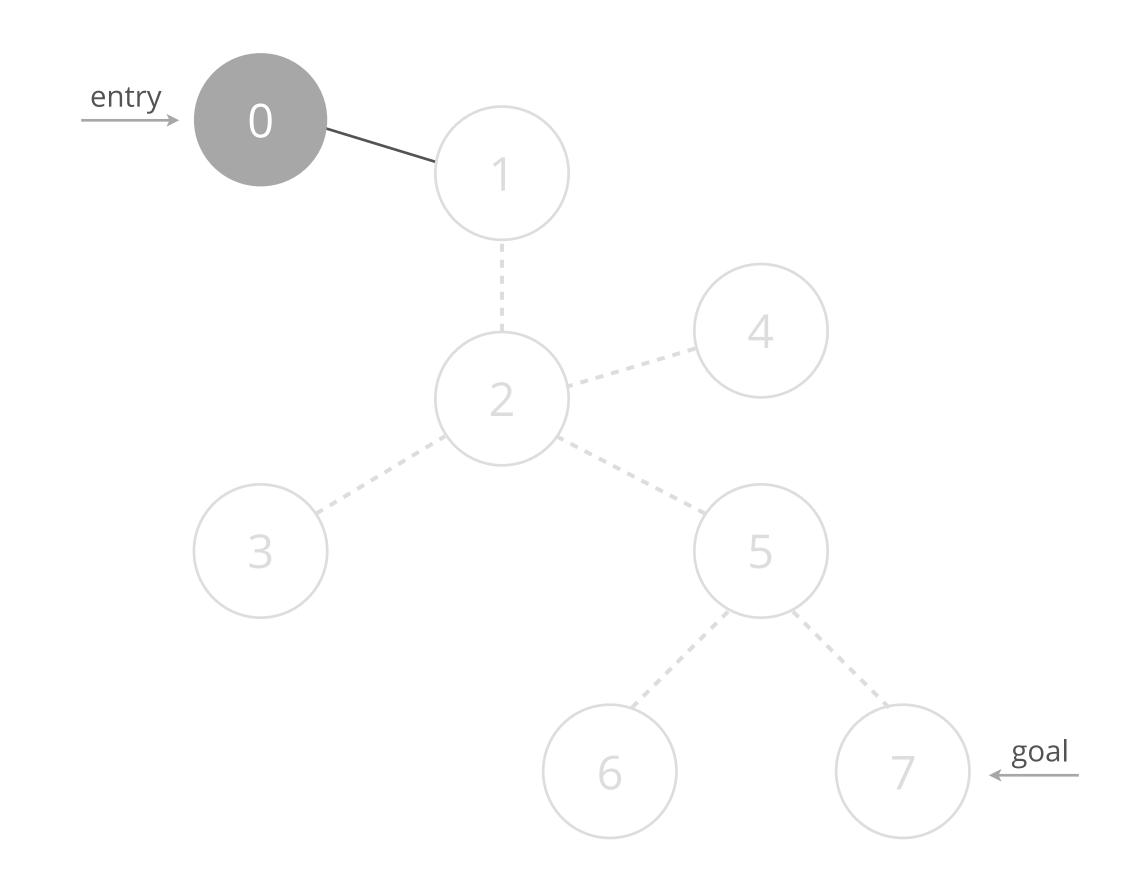
persistence

discovered

ACTION

enumerate: 0

5 ∑



ACTION entry Q scan: 1 4 🗵 goal 6

CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

persistence

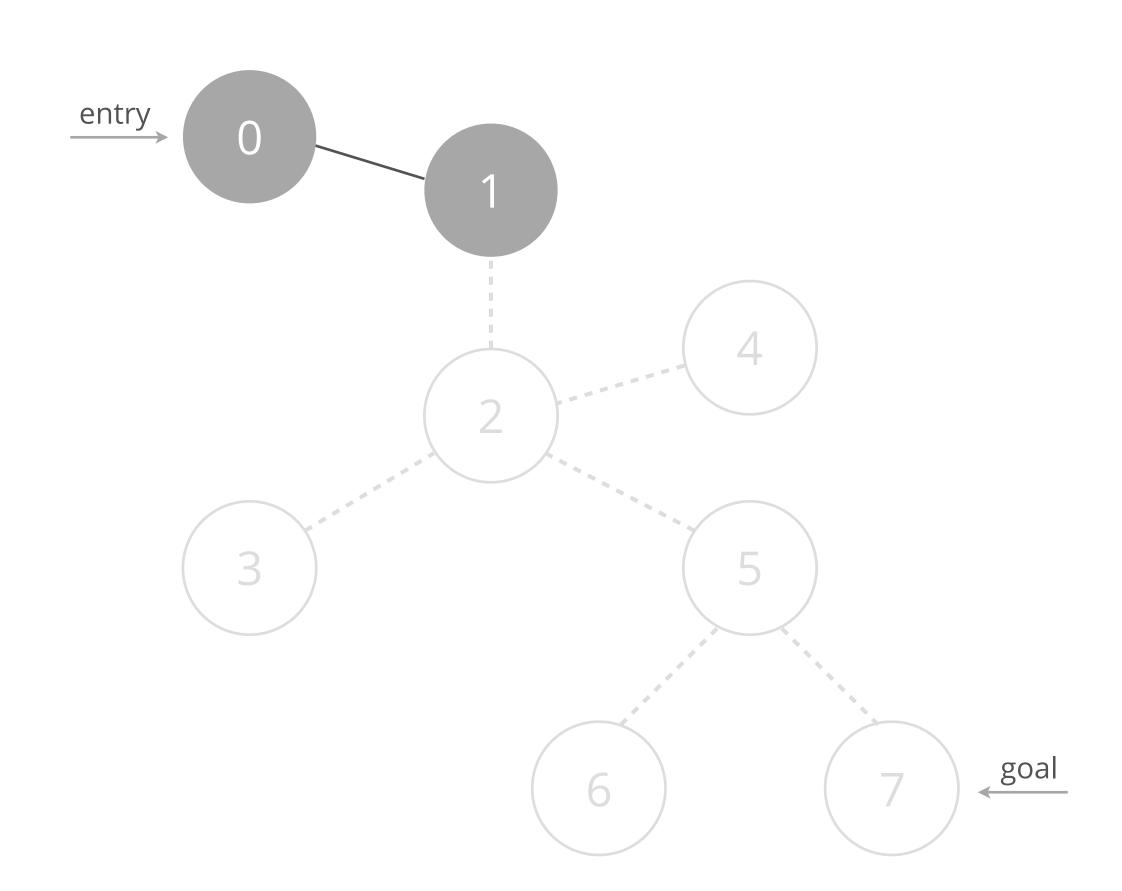
Q dump

ACTION

sexploit A: 1

2 🗵

x crashed



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

persistence

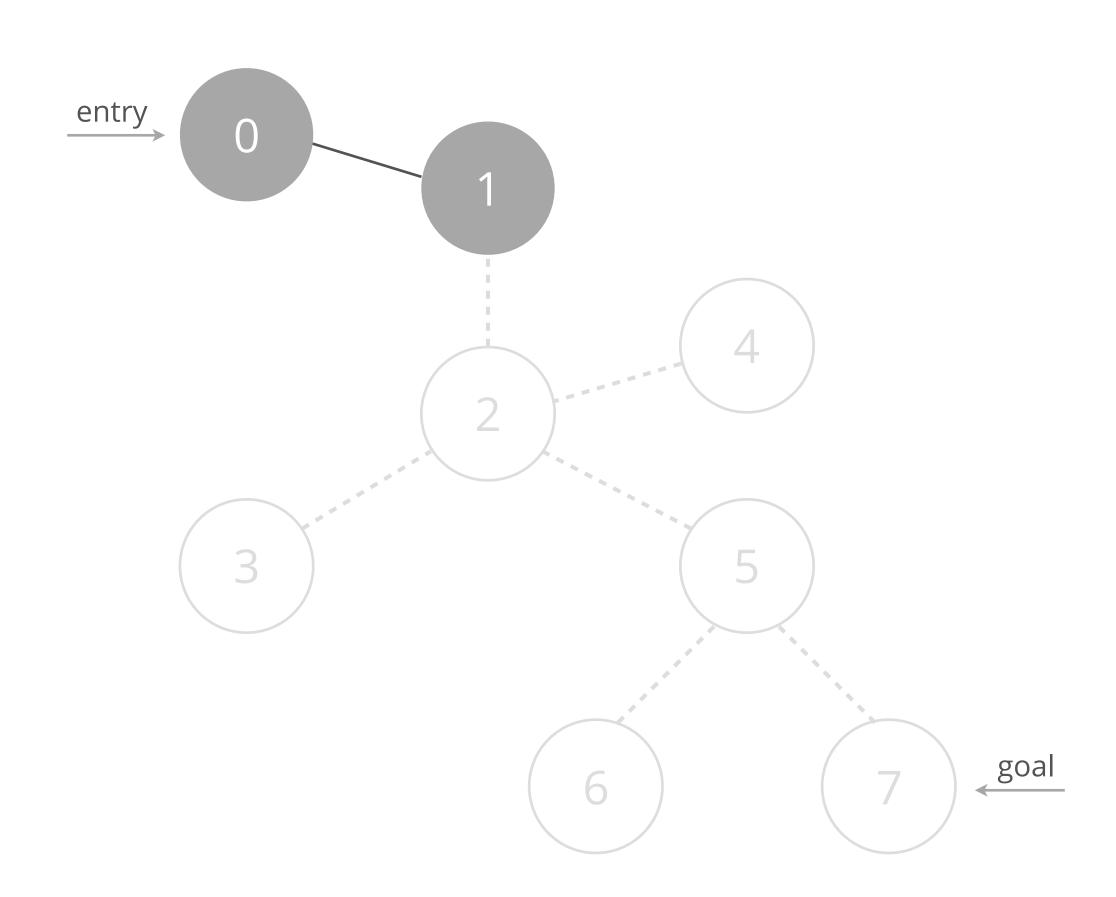
Q dump

ACTION

» exploit B: 1

3 🖫

+5 ₩



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

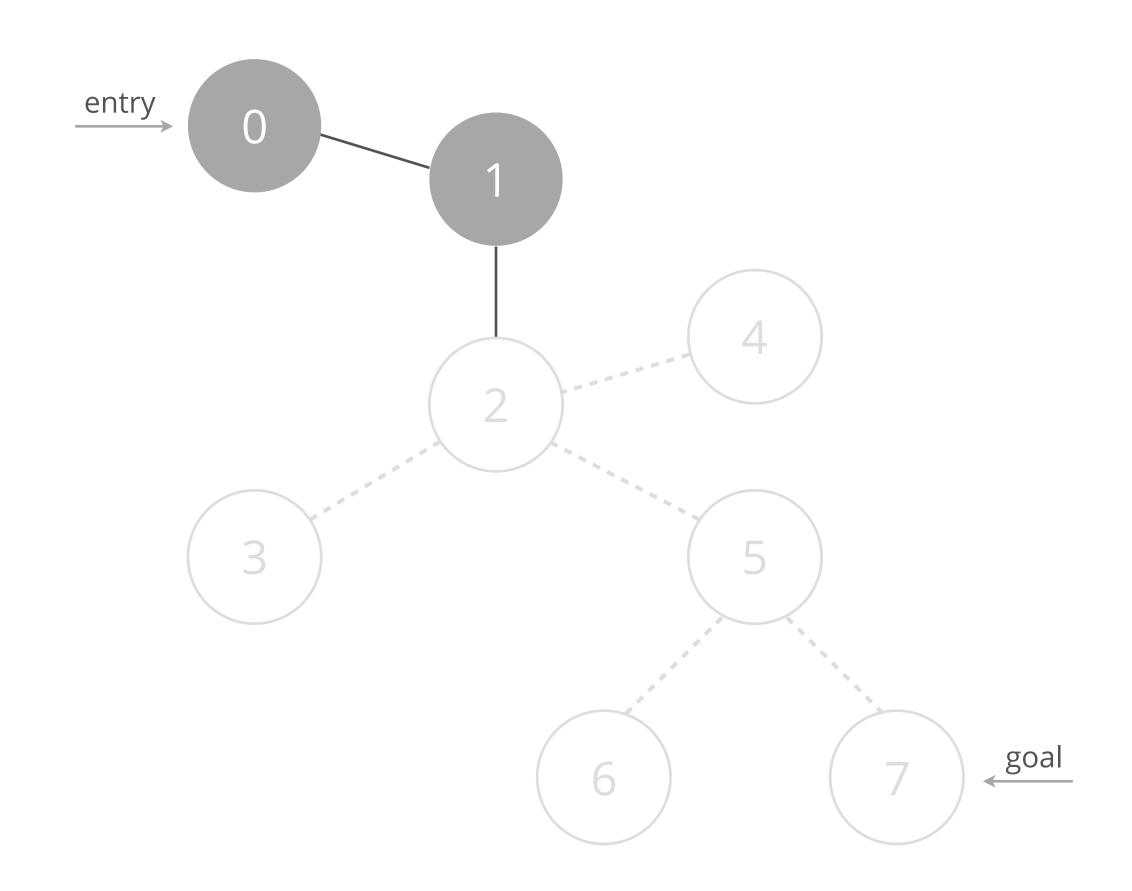
cleanup

persistence

Q dump

ACTION

enumerate: 1
5 \bar{\Sigma}



CONNECTION

- --- undiscovered
- discovered

MACHINE STATUS

- unknown
- scanned
- foothold
- elevated

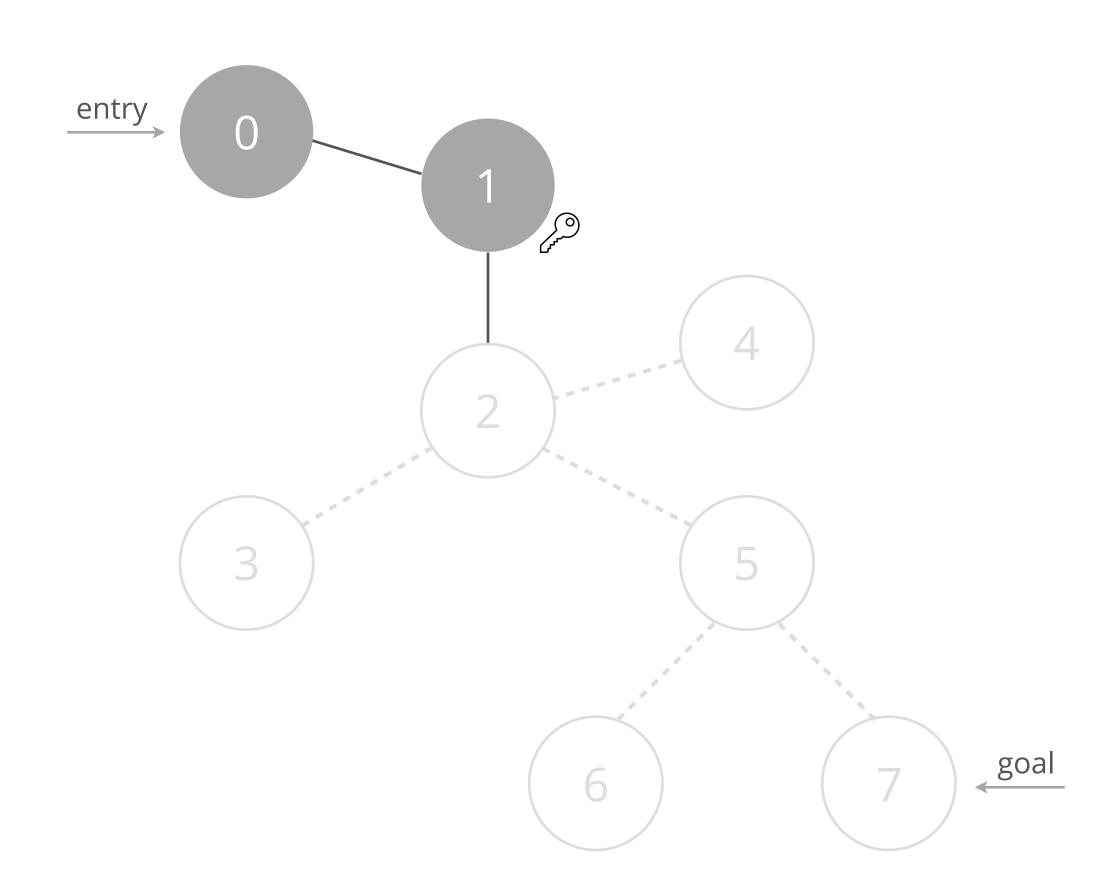
PERFORMED

- cleanup
- persistence
- Q dump
- exfiltrate

ACTION

Q dump: 1

2 🗵



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

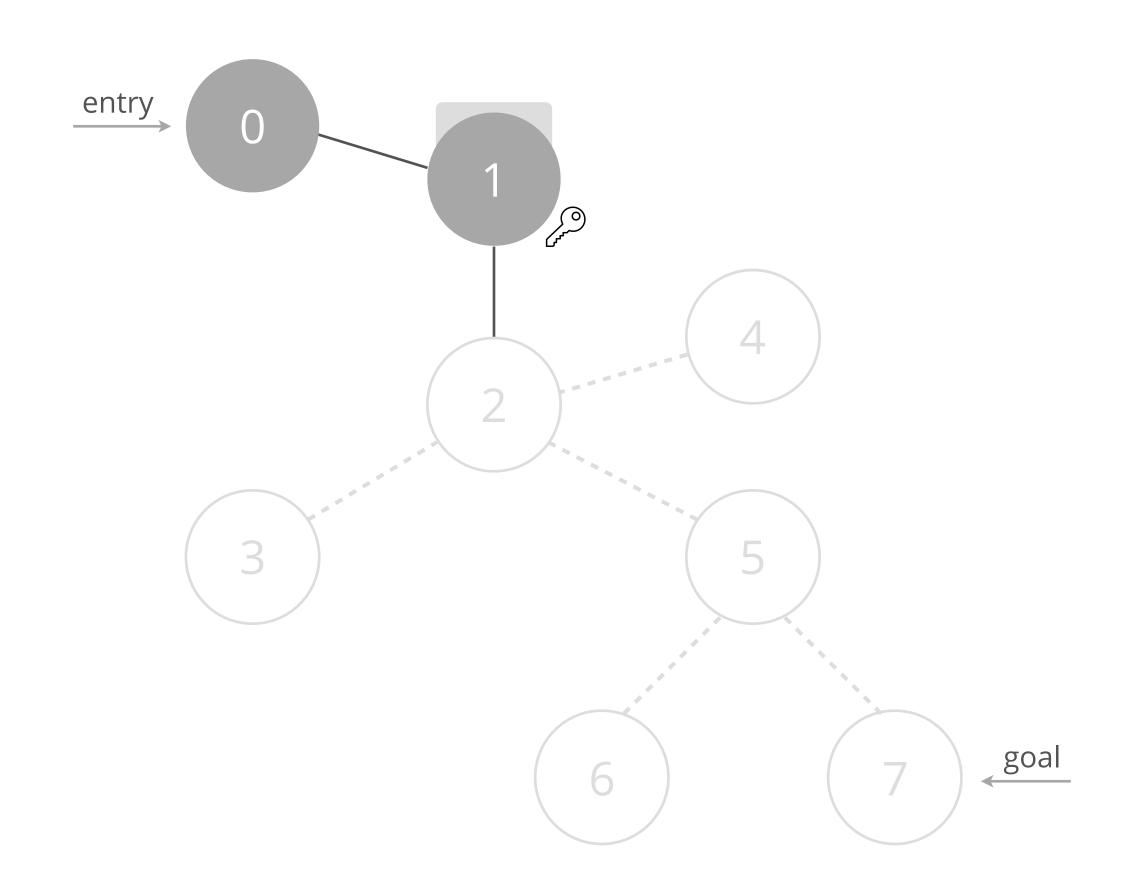
PERFORMED

cleanup

persistence

Q dump

ACTION cleanup: 1 3 \[\begin{align*} \text{3} \\ \text{3} \end{align*}



CONNECTION

- --- undiscovered
- discovered

MACHINE STATUS

- unknown
- scanned
- foothold
- elevated

PERFORMED

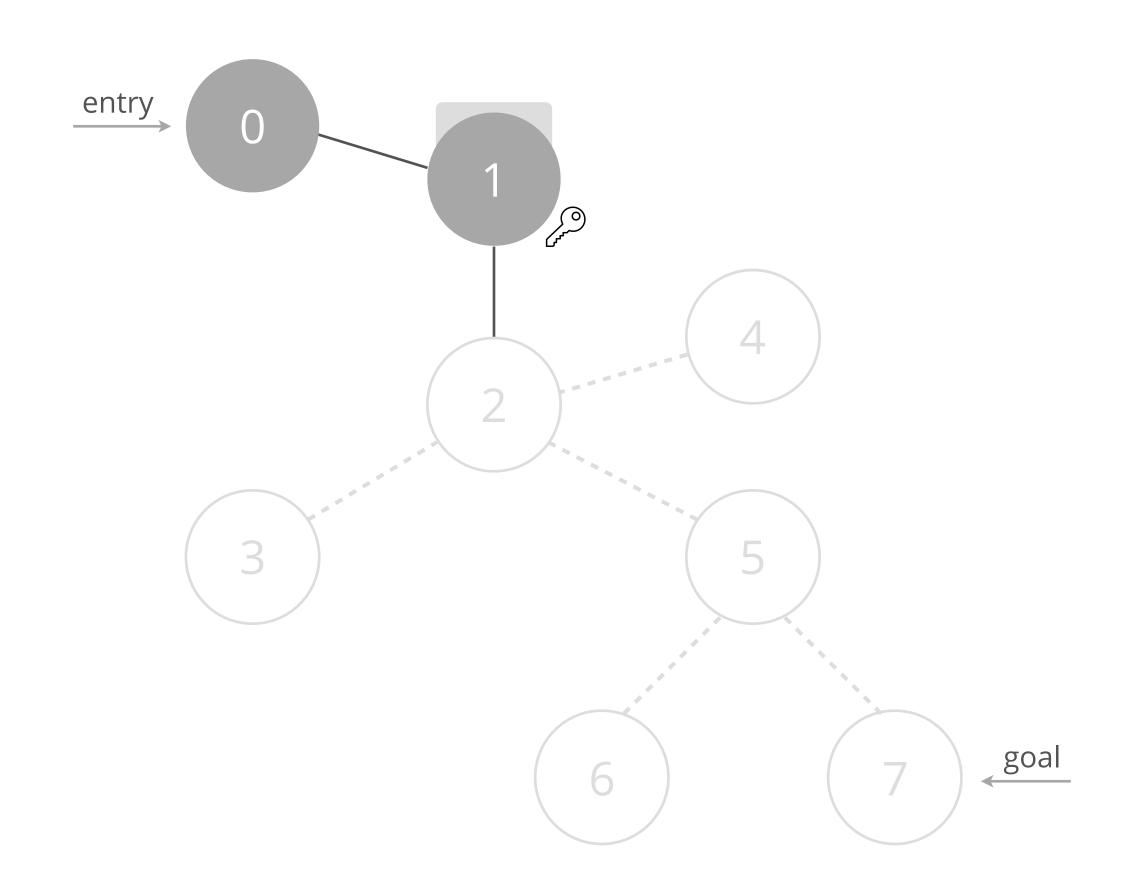
- cleanup
- persistence
- Q dump
- exfiltrate

ACTION

Q scan: 2

4 🗵

failed



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

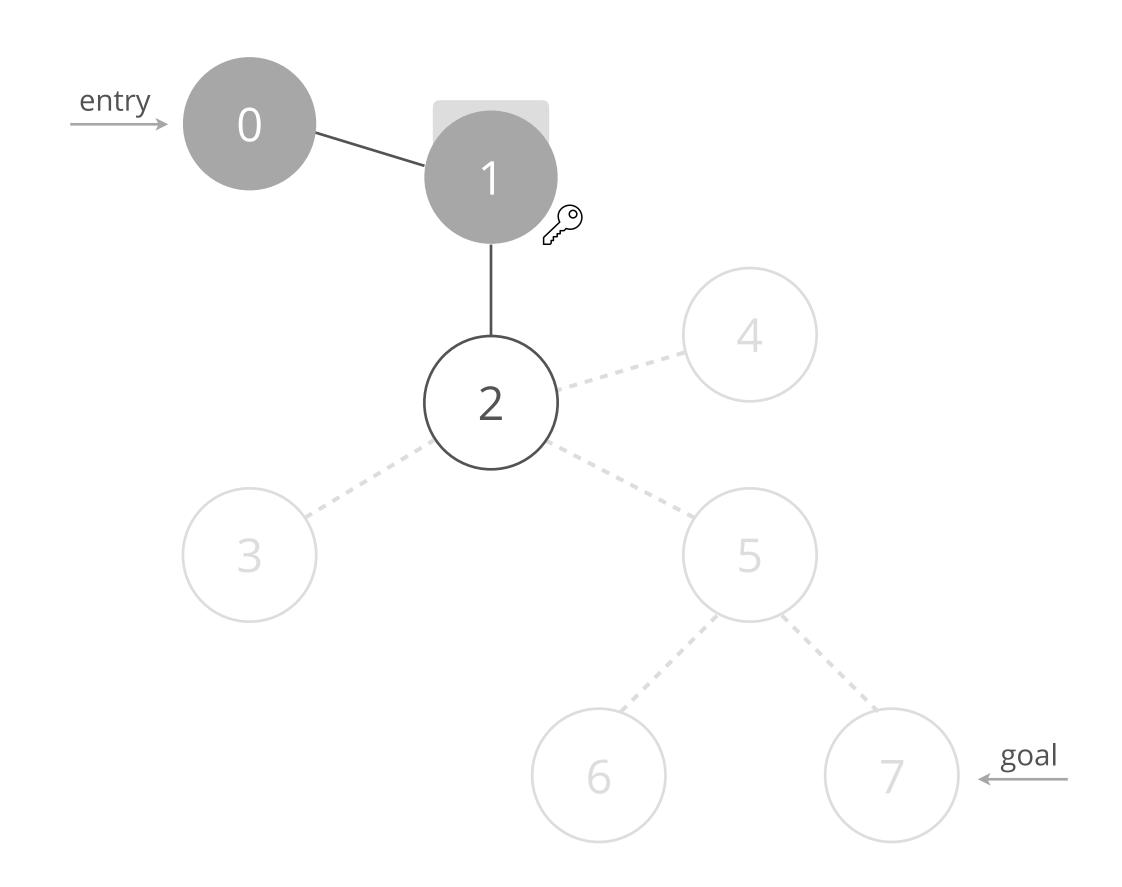
PERFORMED

cleanup

persistence

Q dump

ACTION \bigcirc scan: 2 \bigcirc 4 $\boxed{\mathbb{Z}}$



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

persistence

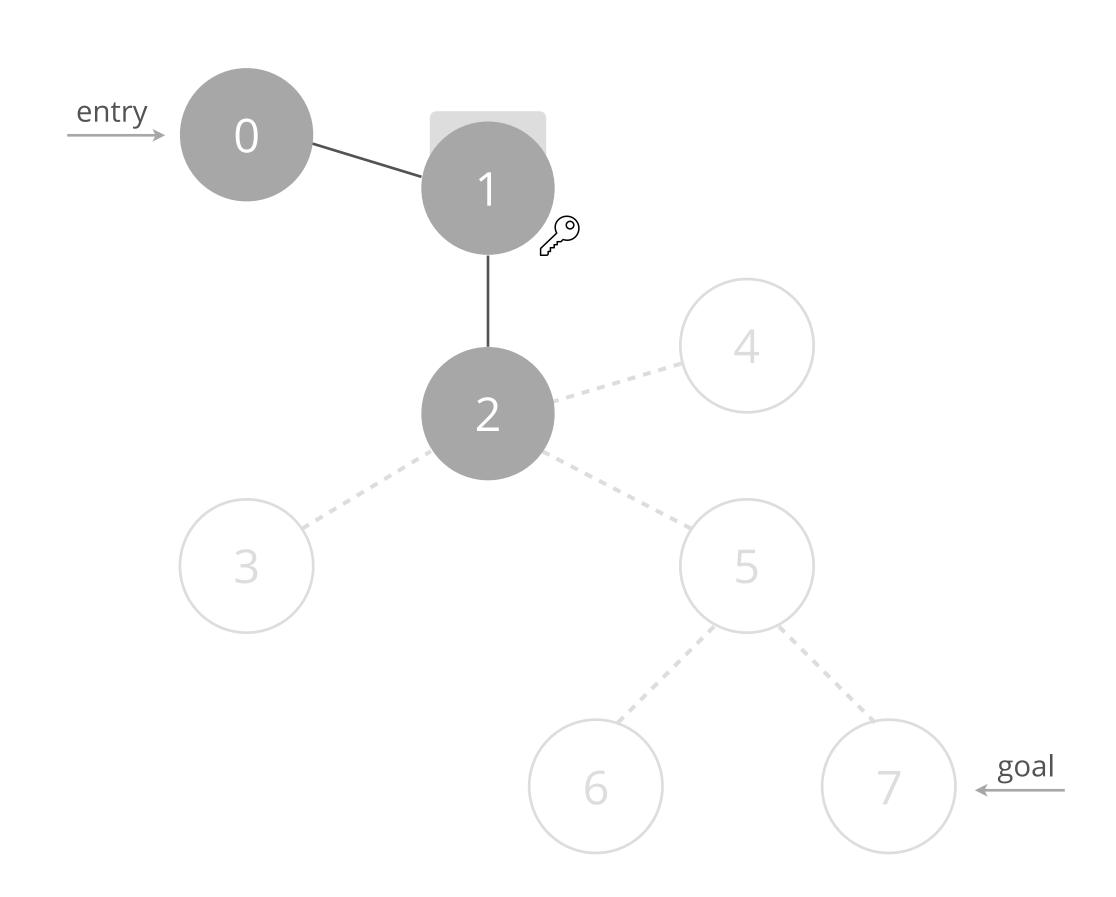
Q dump

ACTION

\$\mathcal{S}\$ exploit C: 2

2 🗵

+5 ₩



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

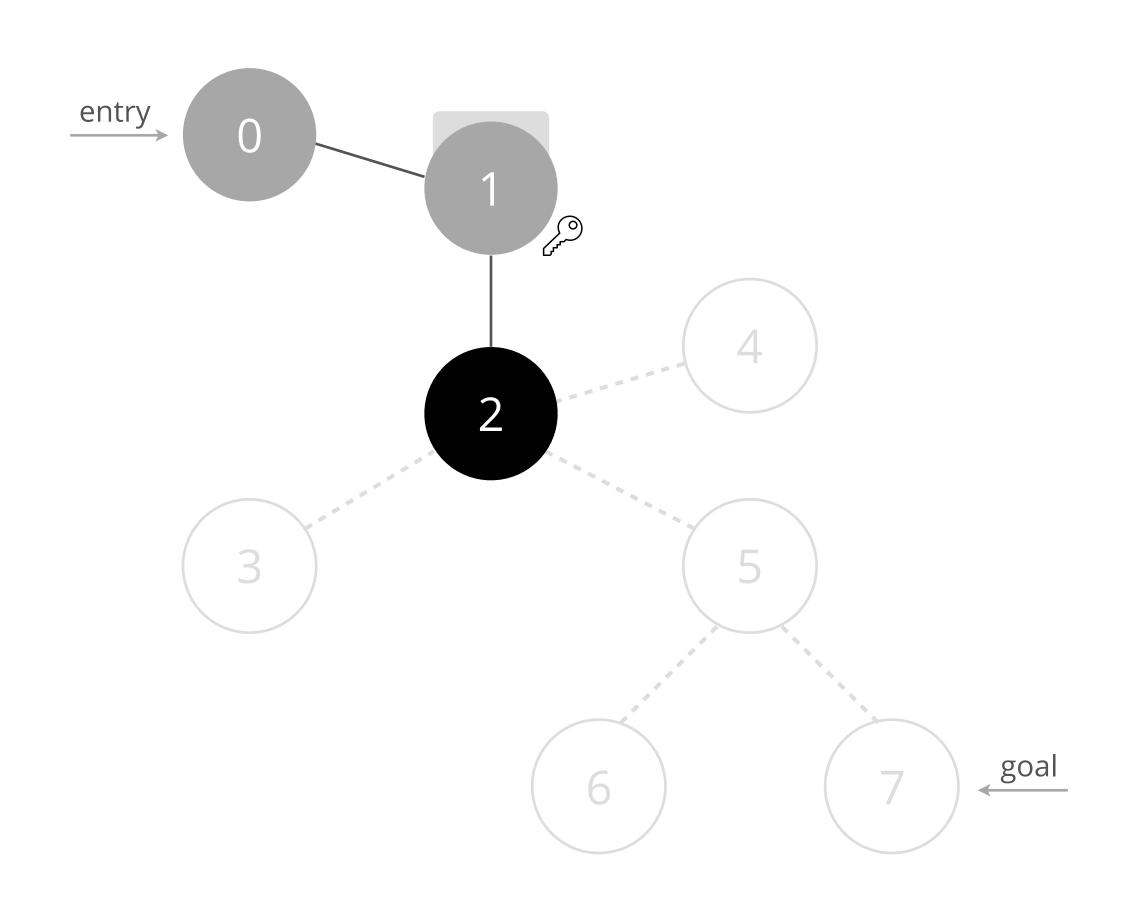
persistence

Q dump

ACTION

escalate: 2

2 🗵



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

persistence

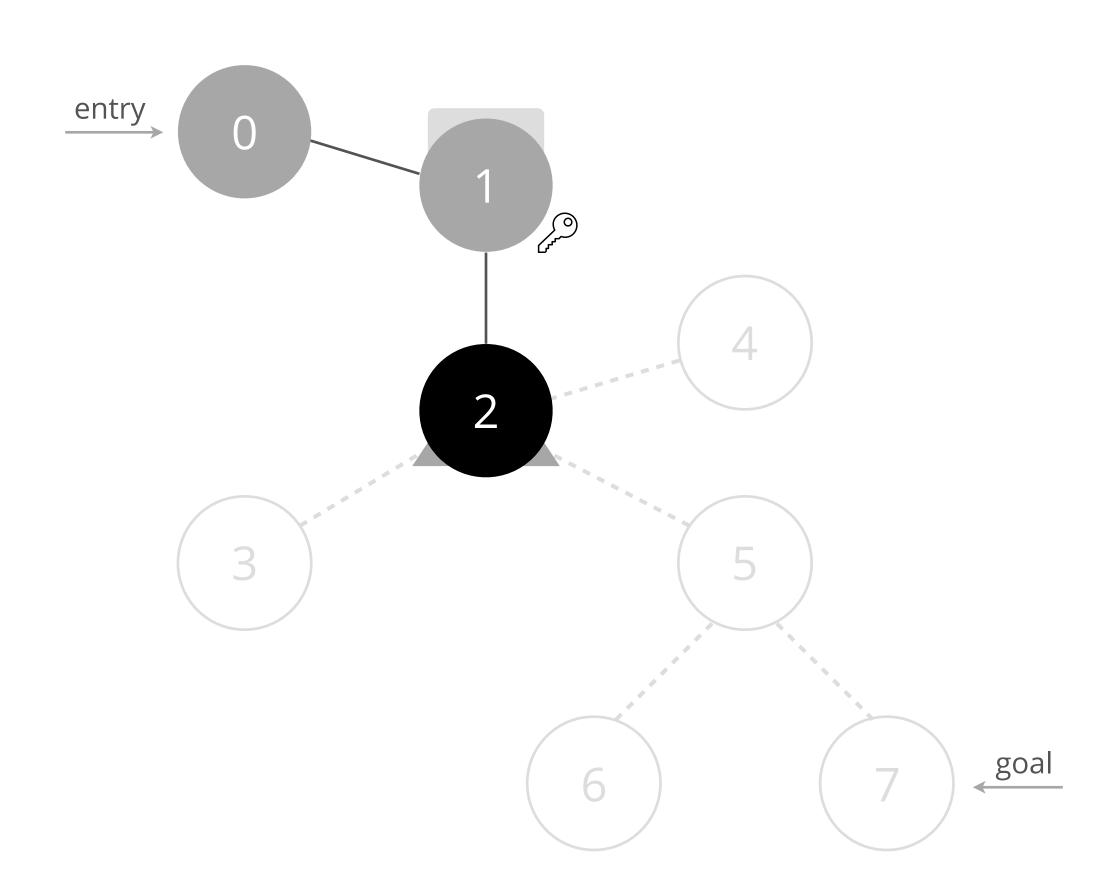
Q dump

ACTION



persist: 2

2 🗵



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

persistence

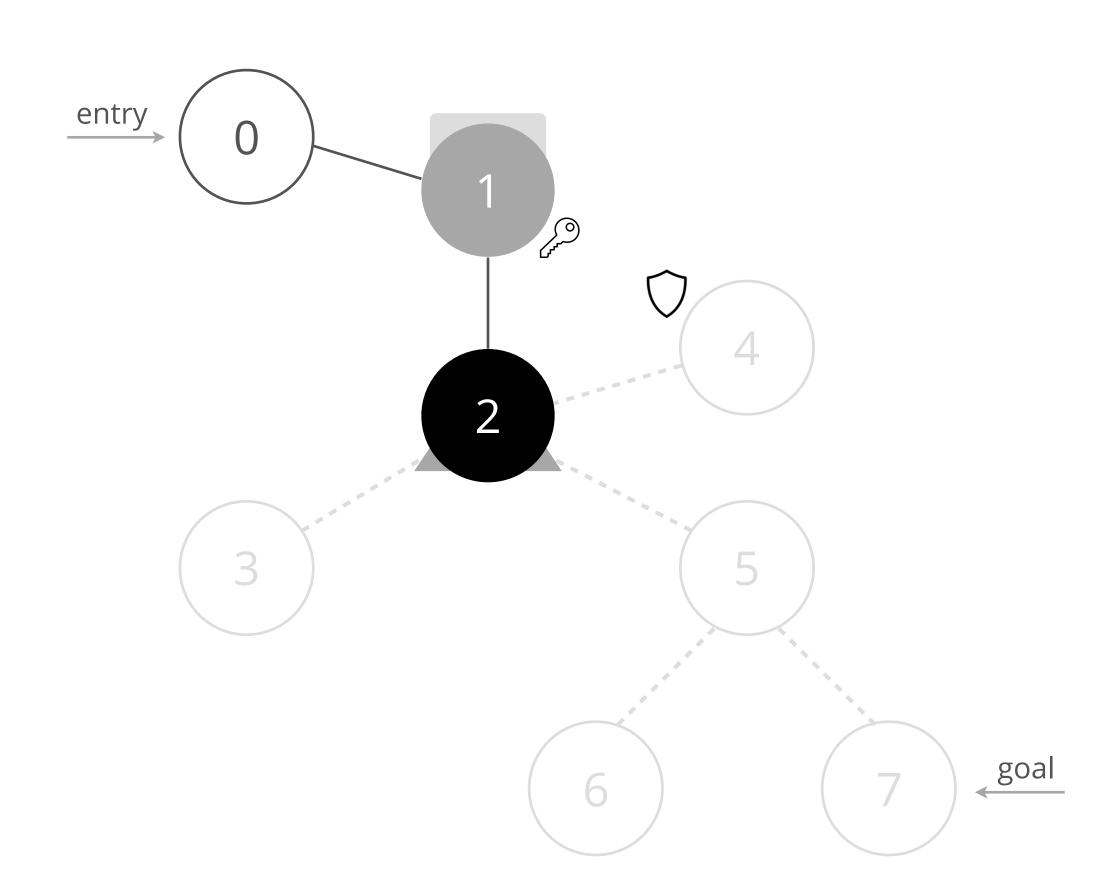
Q dump

ACTION

investigate: 0

-10 ₩

x caught



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

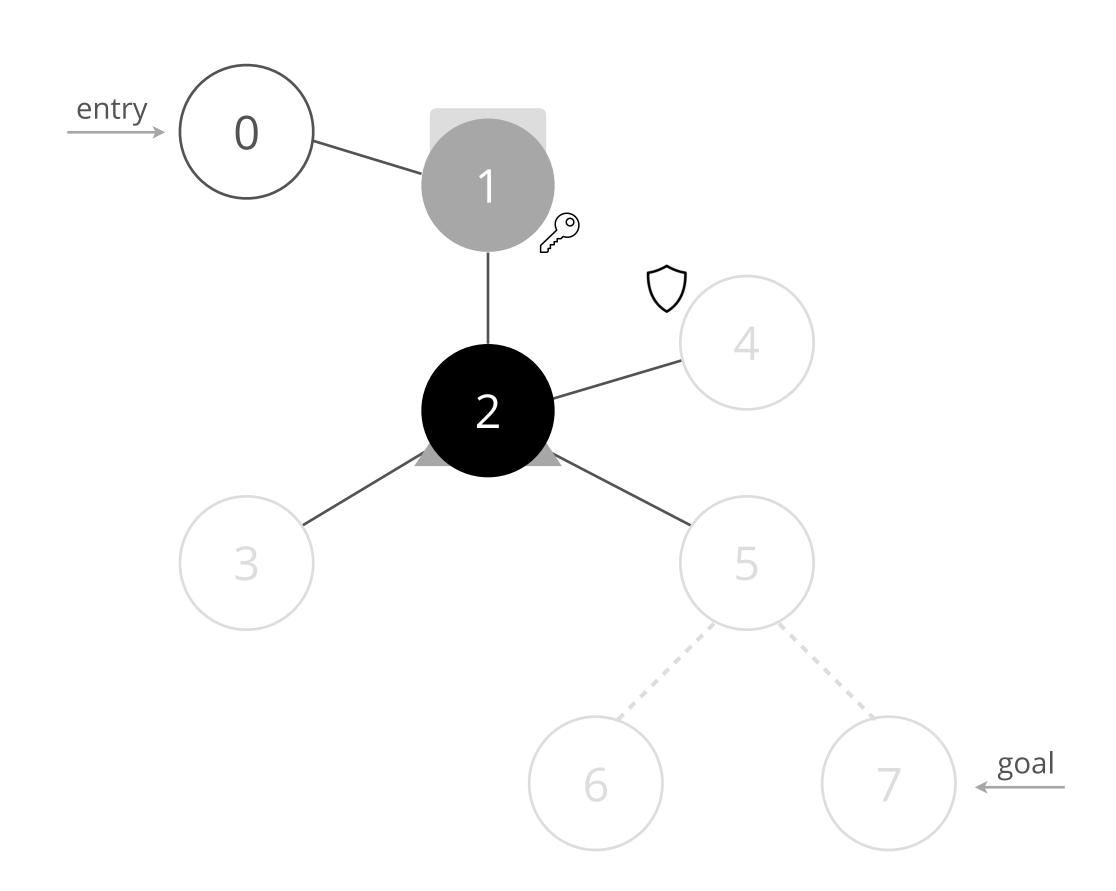
cleanup

persistence

Q dump

ACTION





CONNECTION

- --- undiscovered
- discovered

MACHINE STATUS

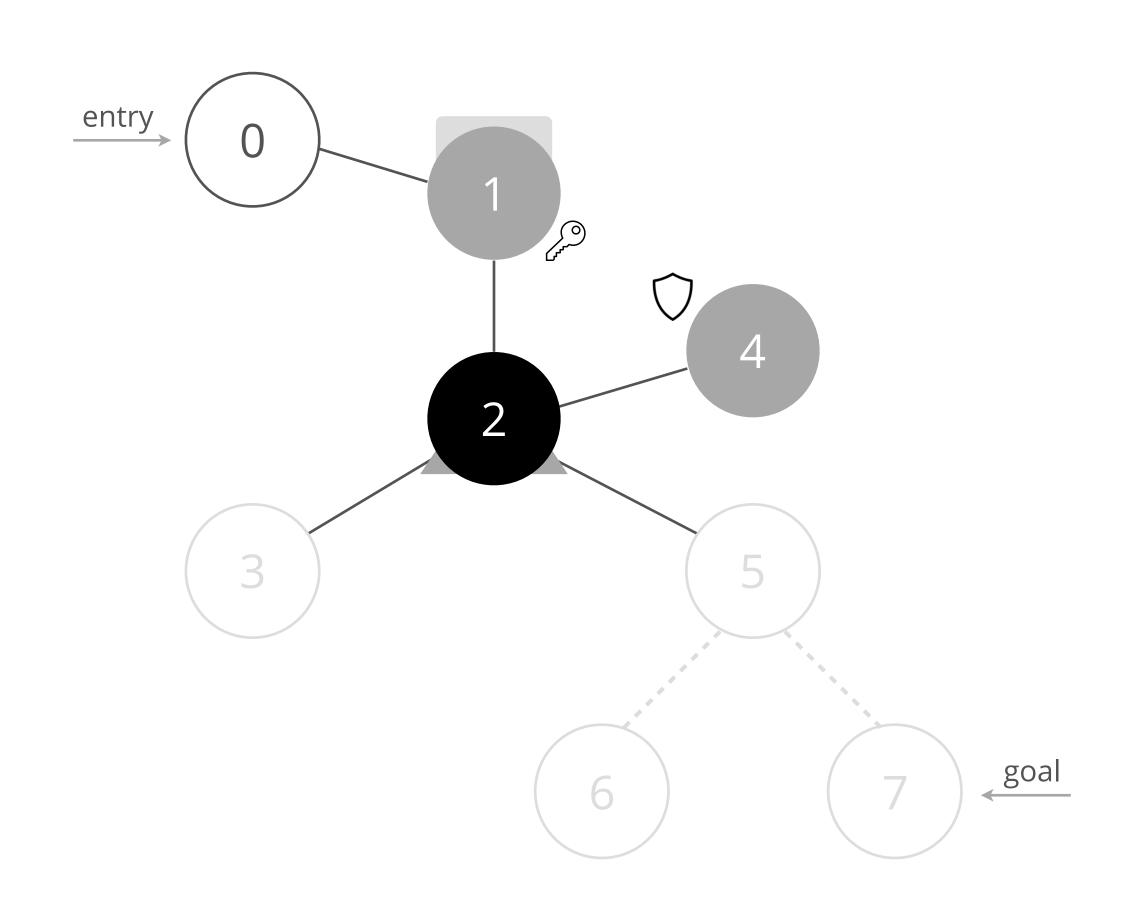
- unknown
- scanned
- foothold
- elevated

- cleanup
- persistence
- Q dump
- exfiltrate

ACTION

migrate: 4

2 \(\bar{2} \)
+5 \(\operatorname{3} \)



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

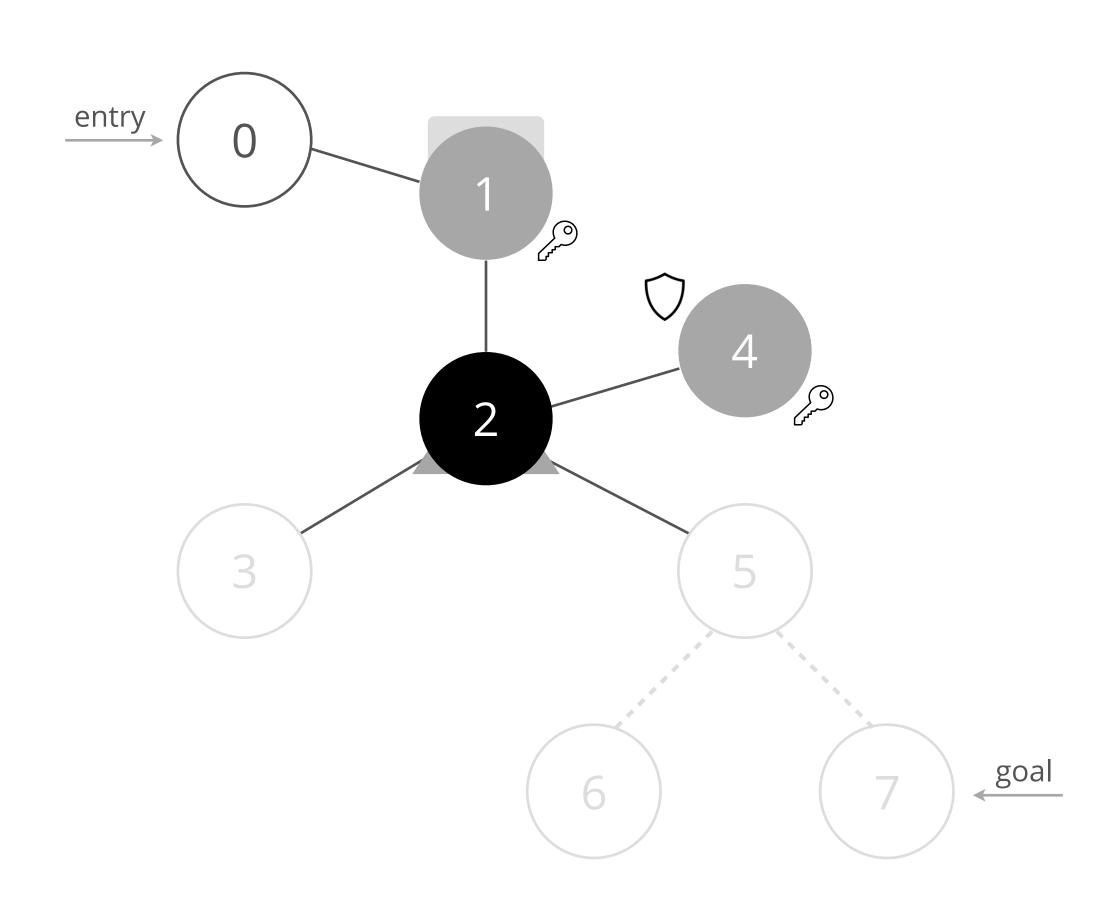
persistence

Q dump

ACTION

Q dump: 4

1 🗵



CONNECTION

- --- undiscovered
- discovered

MACHINE STATUS

- unknown
- scanned
- foothold
- elevated

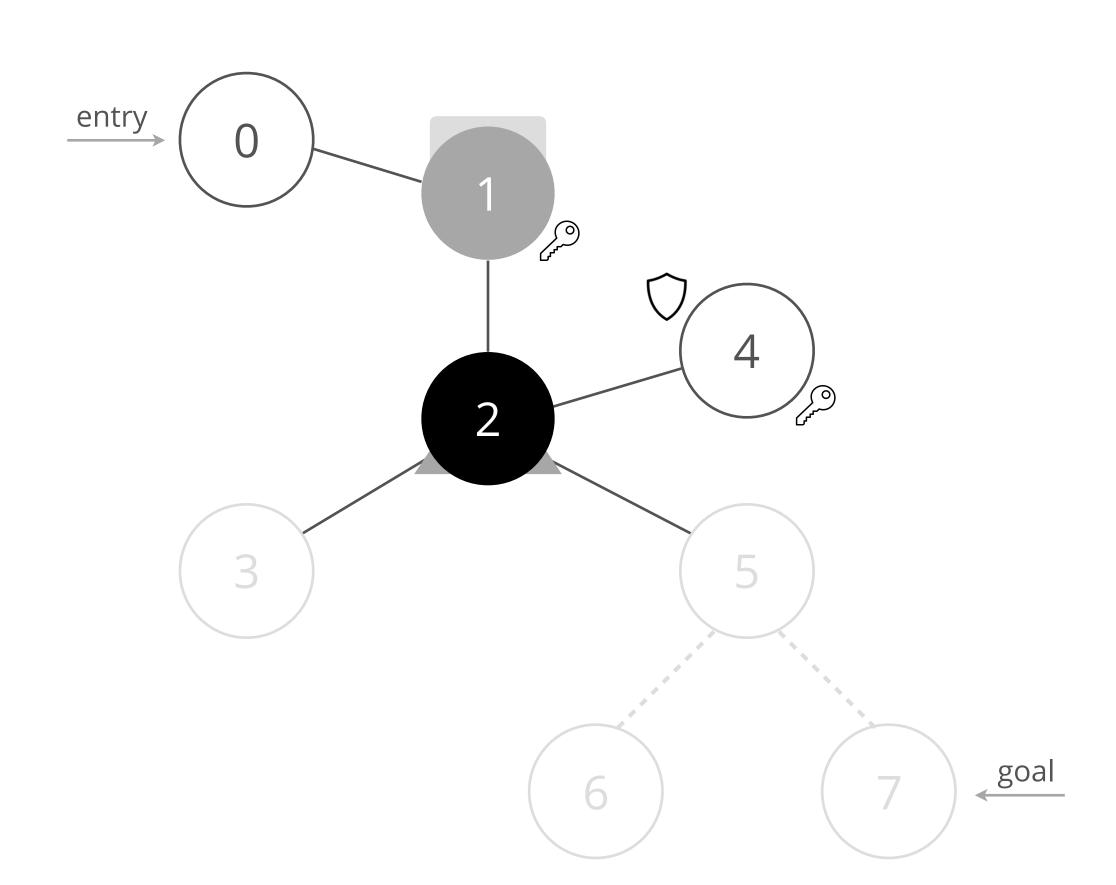
- cleanup
- persistence
- Q dump
- exfiltrate

ACTION



1 🔀

* blocked



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

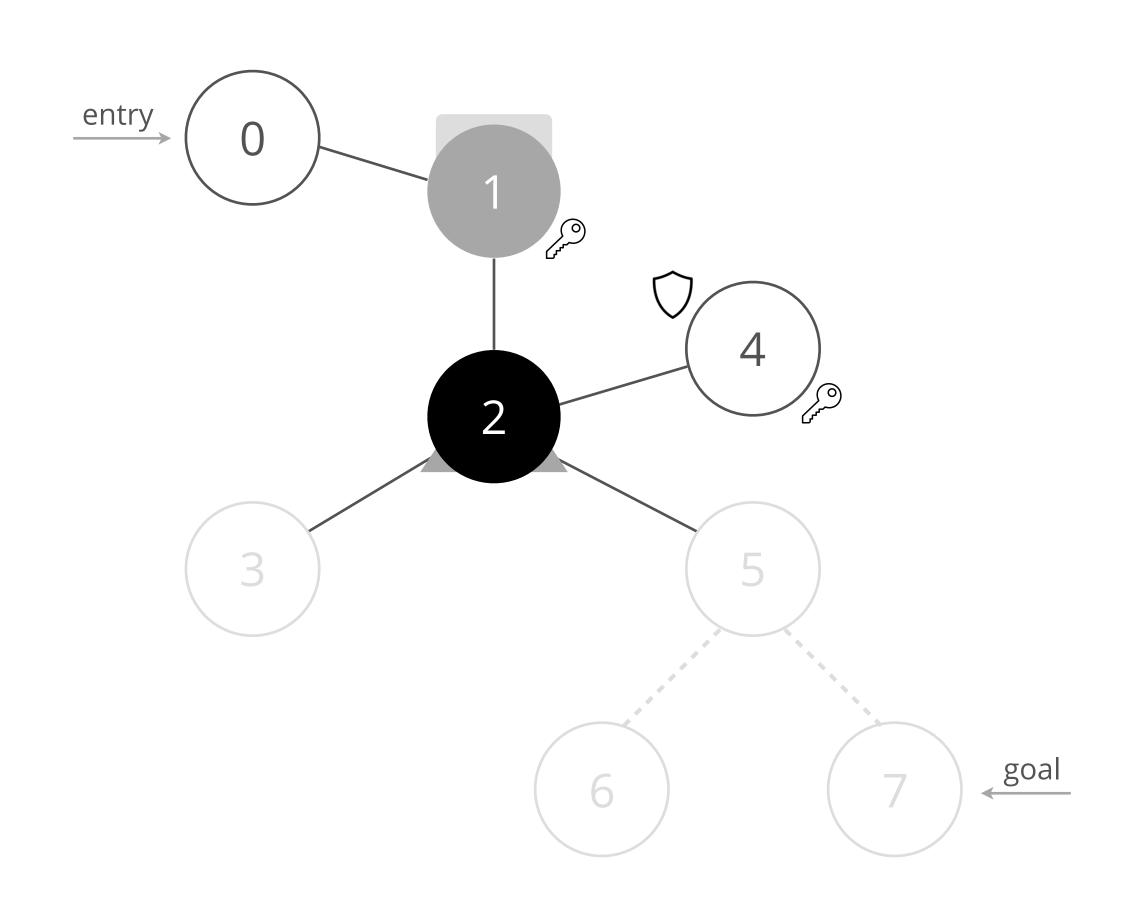
persistence

Q dump

ACTION

wait

5



CONNECTION

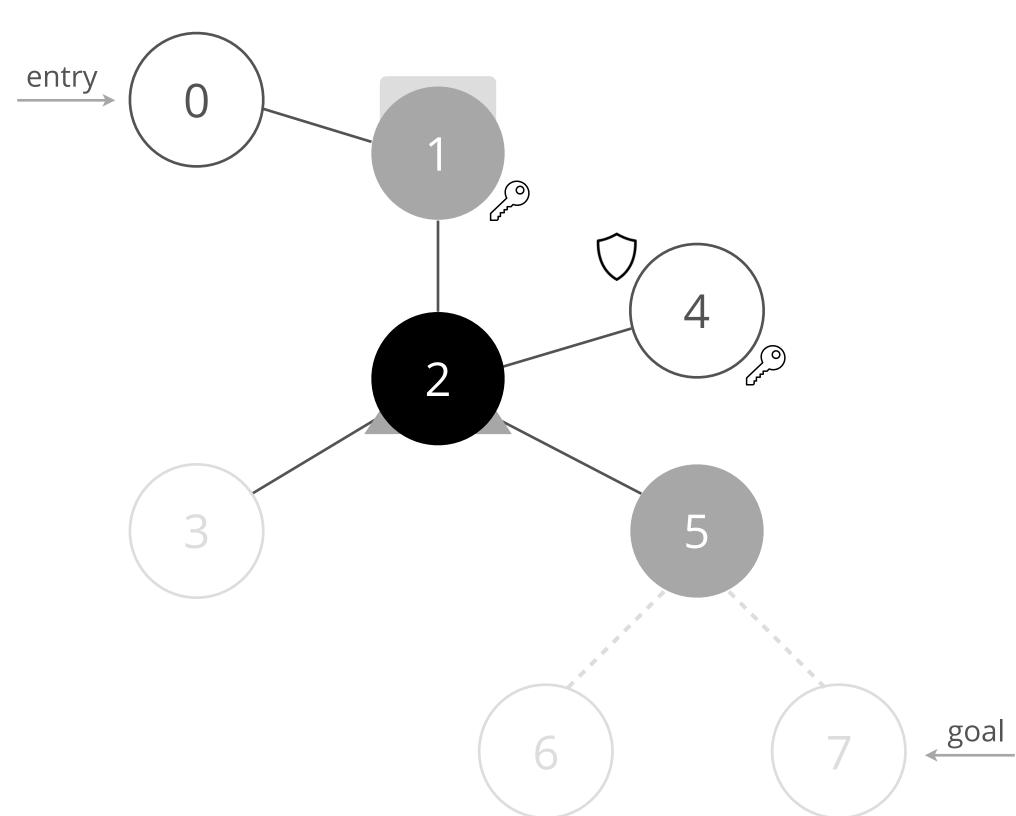
- --- undiscovered
- discovered

MACHINE STATUS

- unknown
- scanned
- foothold
- elevated

- cleanup
- persistence
- Q dump
- exfiltrate

ACTION entry **migrate** 2 🗵 +5 **(foothold)**



CONNECTION

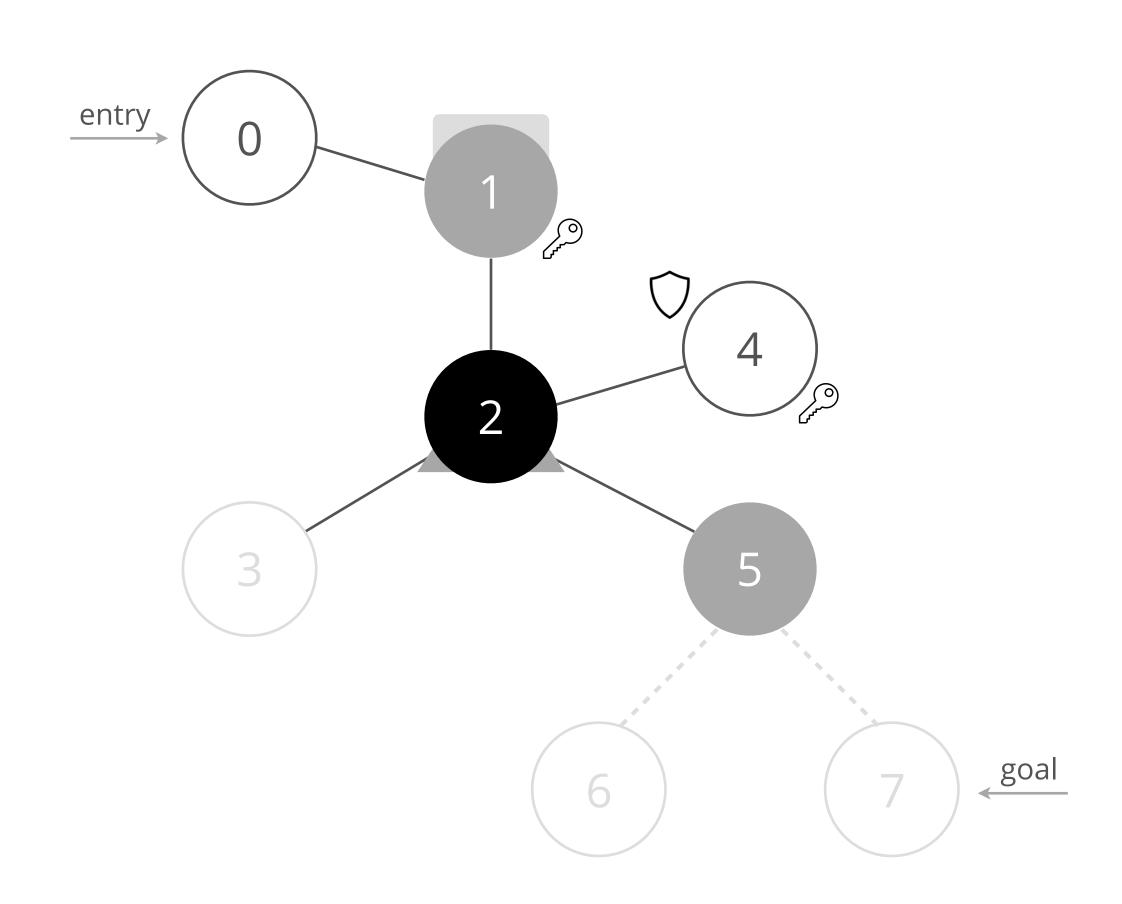
- --- undiscovered
- discovered

MACHINE STATUS

- unknown
- scanned
- foothold
- elevated

- cleanup
- persistence
- Q dump
- exfiltrate

ACTION wait 5 🗵



CONNECTION

- --- undiscovered
- discovered

MACHINE STATUS

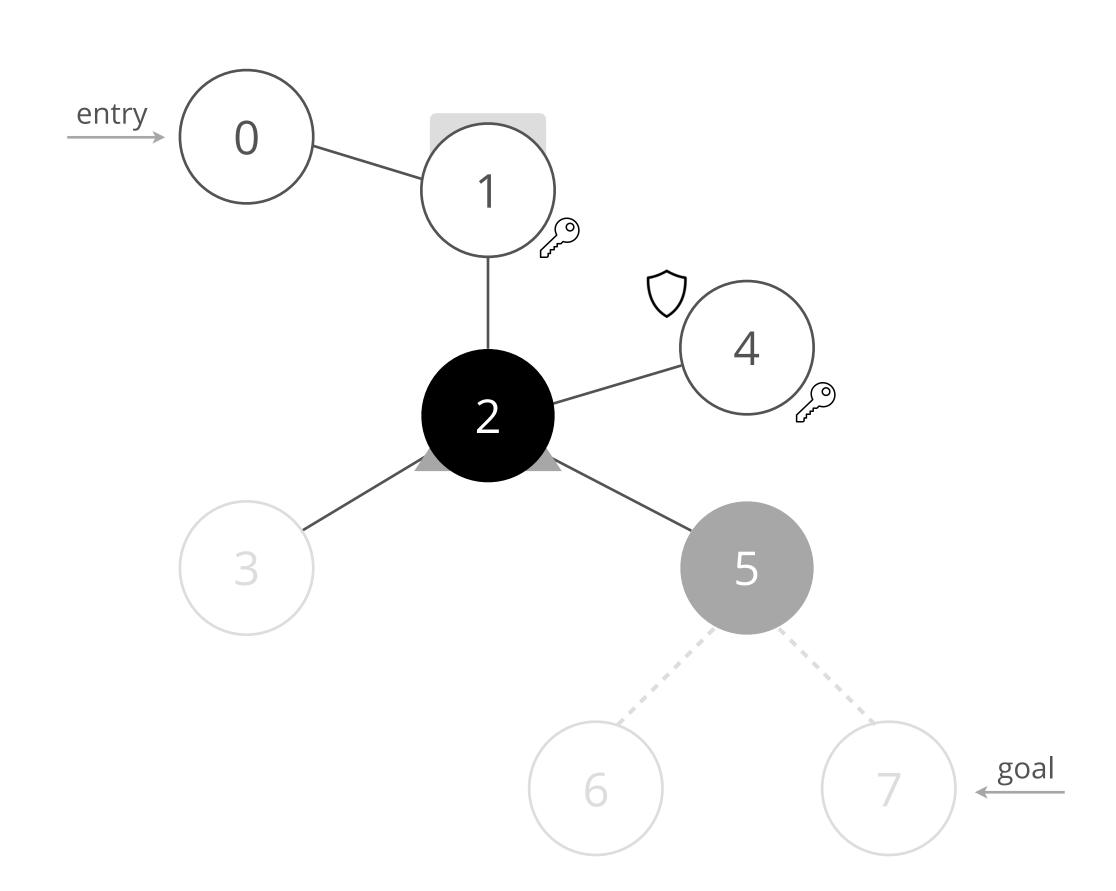
- unknown
- scanned
- foothold
- elevated

- cleanup
- persistence
- Q dump
- exfiltrate

ACTION



X lost foothold



CONNECTION

- --- undiscovered
- discovered

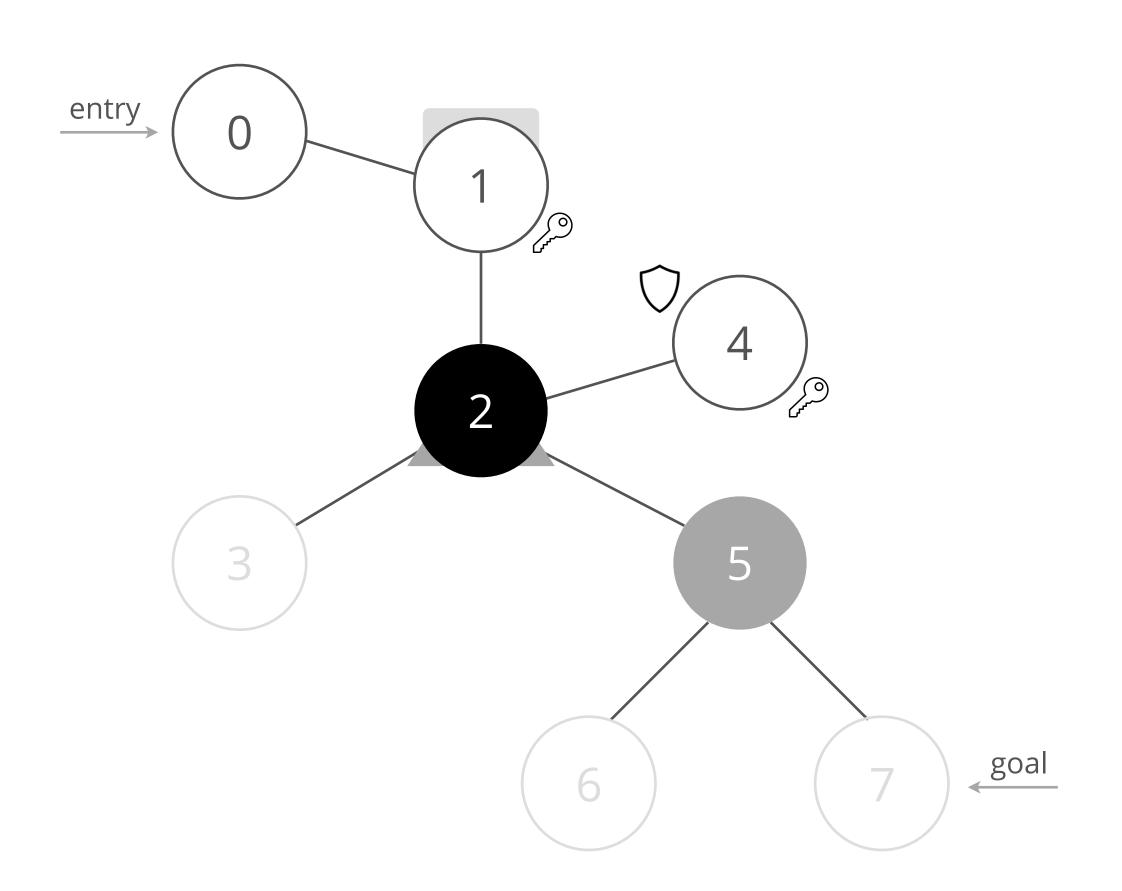
MACHINE STATUS

- unknown
- scanned
- foothold
- elevated

- cleanup
- persistence
- Q dump
- exfiltrate

ACTION

enumerate: 55 ∑



CONNECTION

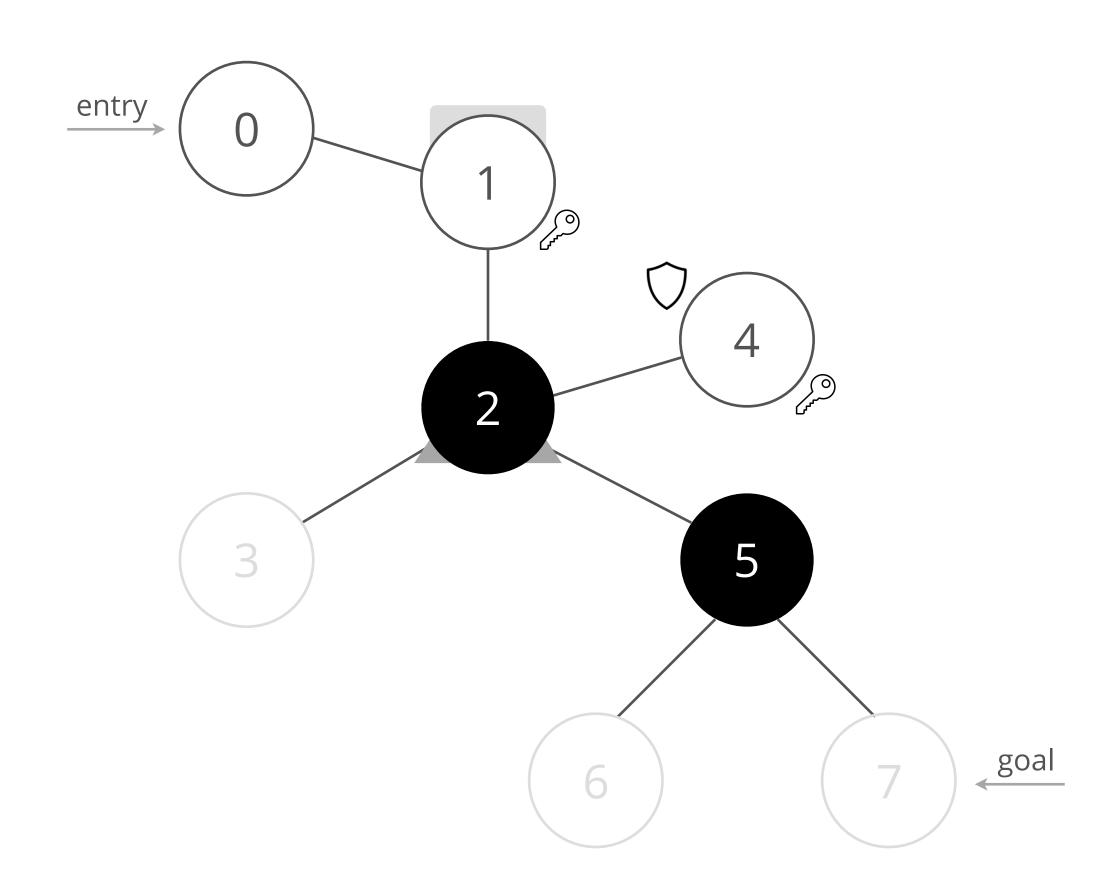
- --- undiscovered
- discovered

MACHINE STATUS

- unknown
- scanned
- foothold
- elevated

- cleanup
- persistence
- Q dump
- exfiltrate

ACTION



CONNECTION

- --- undiscovered
- discovered

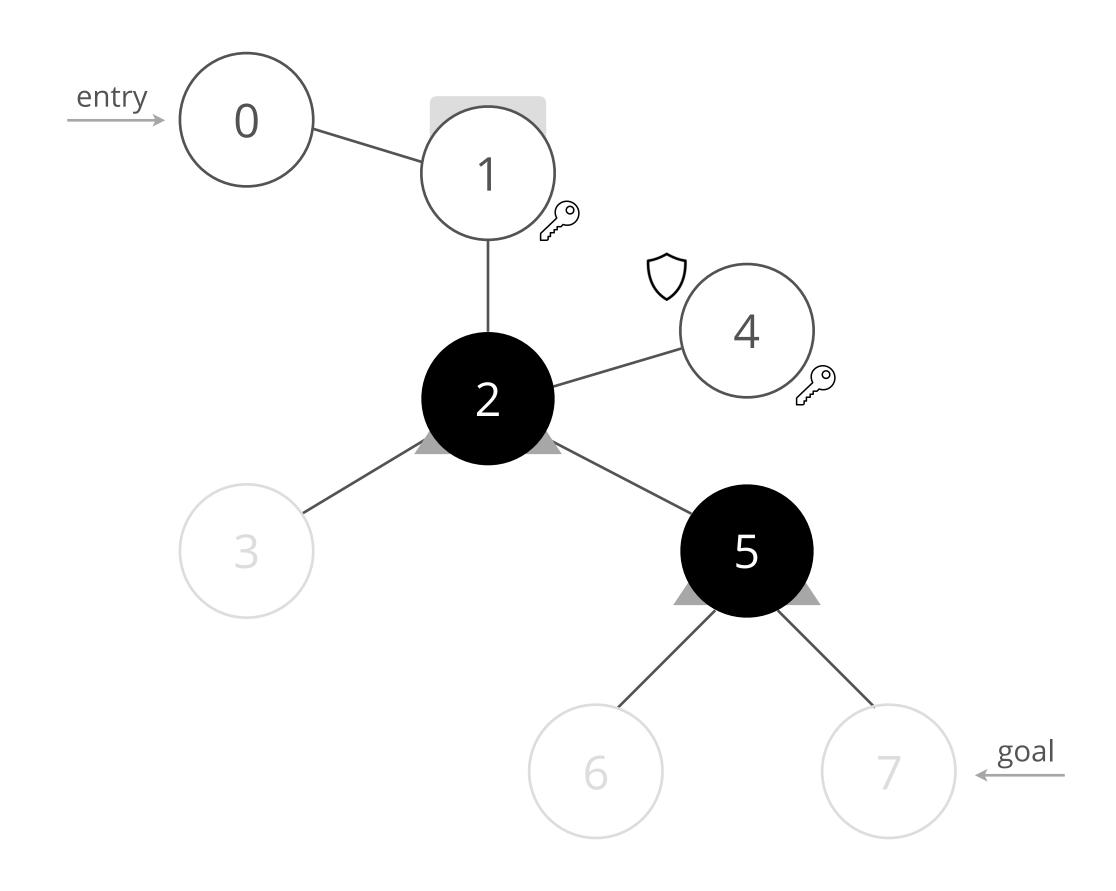
MACHINE STATUS

- unknown
- scanned
- foothold
- elevated

- cleanup
- persistence
- Q dump
- exfiltrate

ACTION





CONNECTION

- --- undiscovered
- discovered

MACHINE STATUS

- unknown
- scanned
- foothold
- elevated

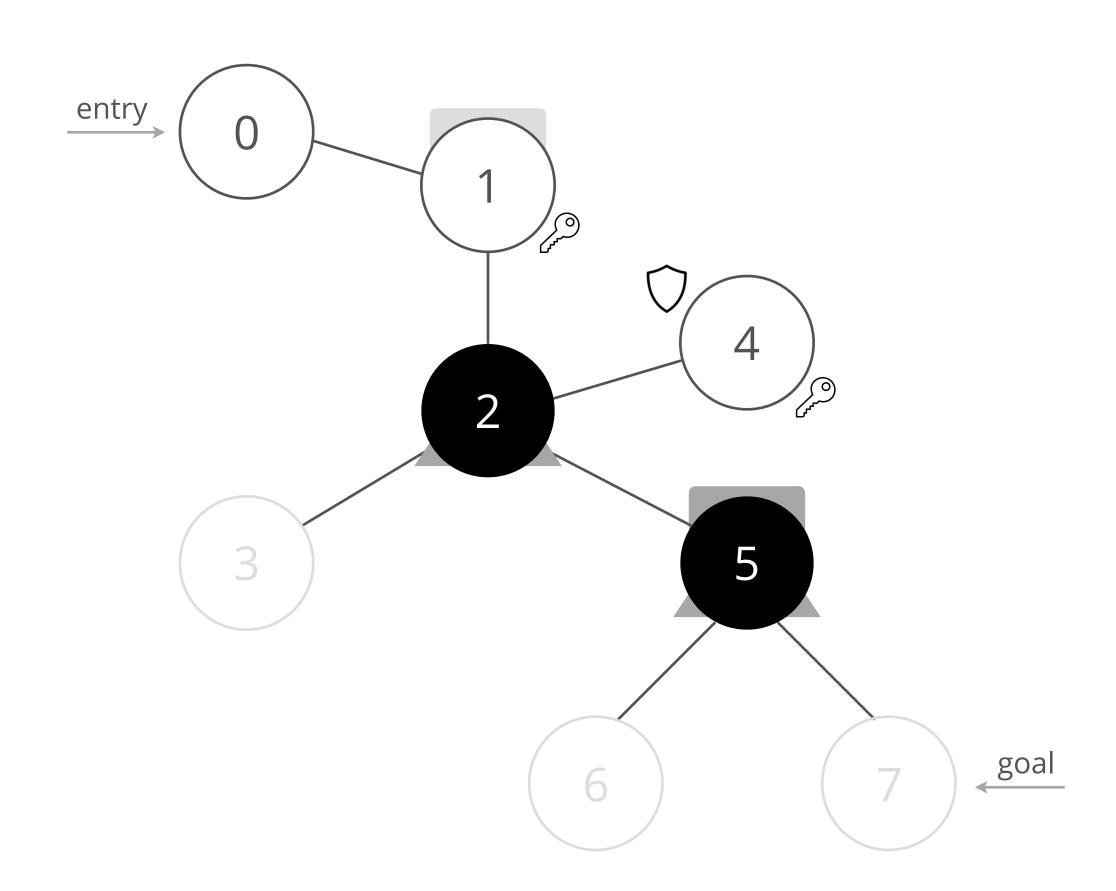
- cleanup
- persistence
- Q dump
- exfiltrate

ACTION



cleanup: 5

3 🗵



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

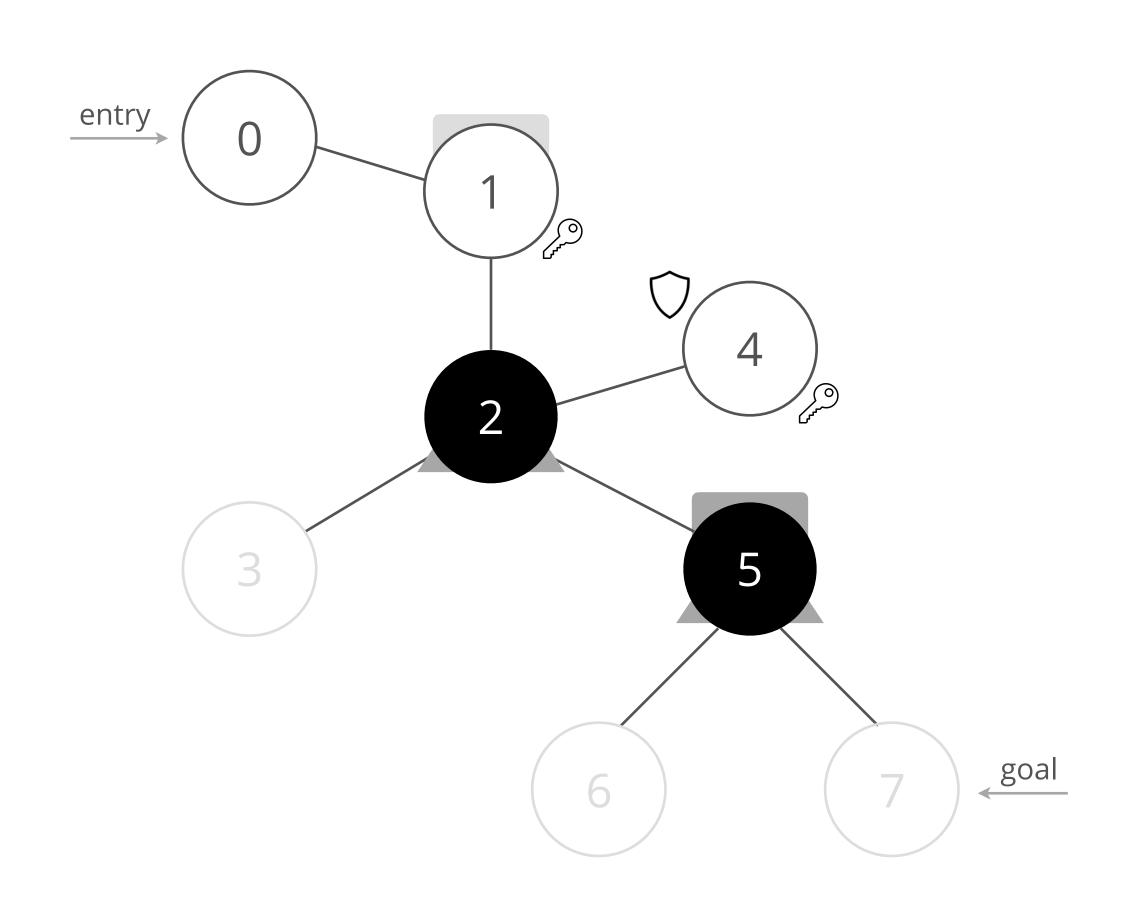
cleanup

persistence

Q dump

ACTION





CONNECTION

- --- undiscovered
- discovered

MACHINE STATUS

- unknown
- scanned
- foothold
- elevated

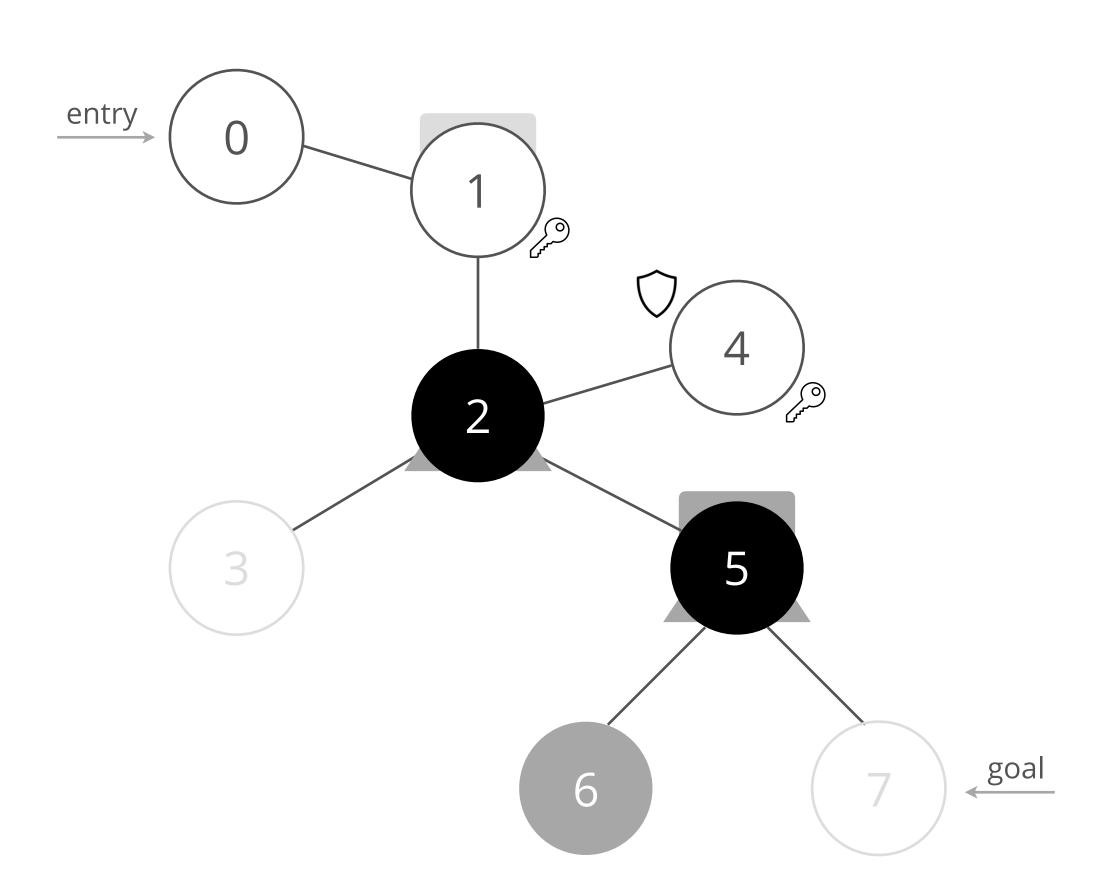
- cleanup
- persistence
- Q dump
- exfiltrate

ACTION

login: 6

1 🖫

+5 **(foothold)**



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

persistence

Q dump

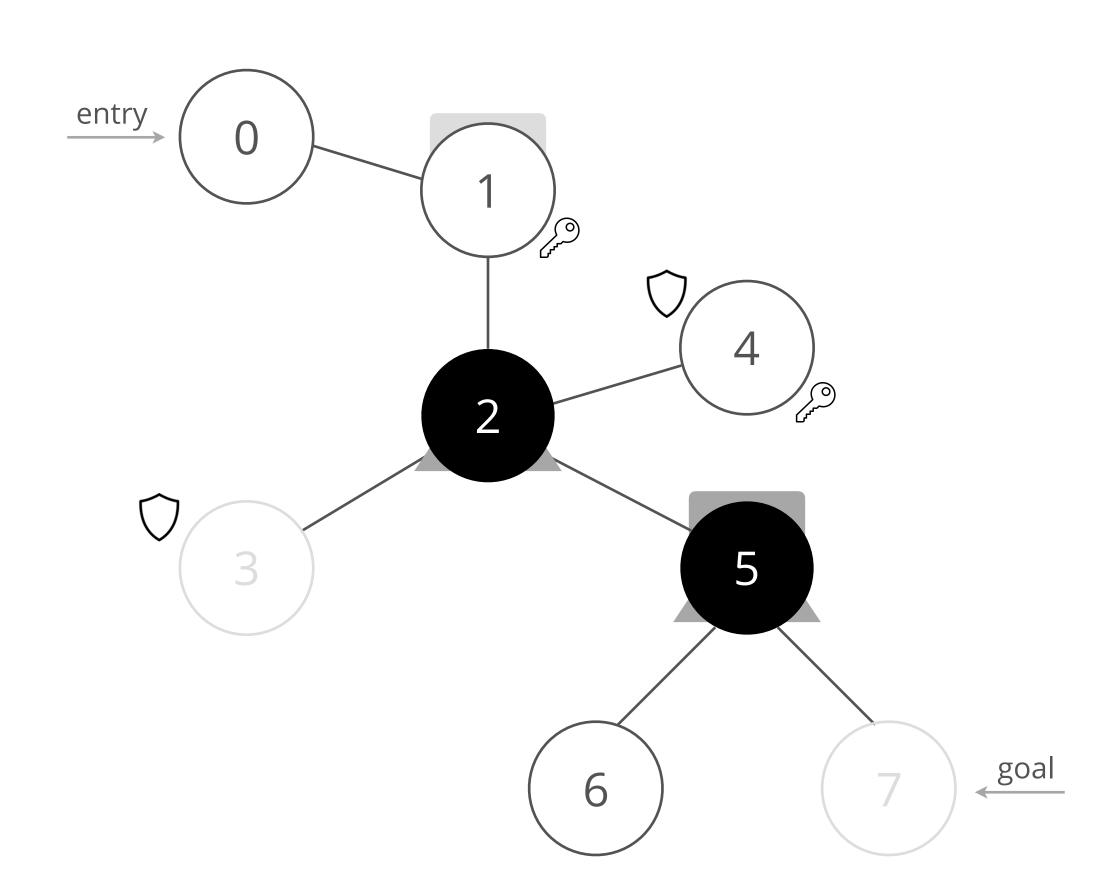
ACTION



1 🗵

-10 ₩

X detected



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

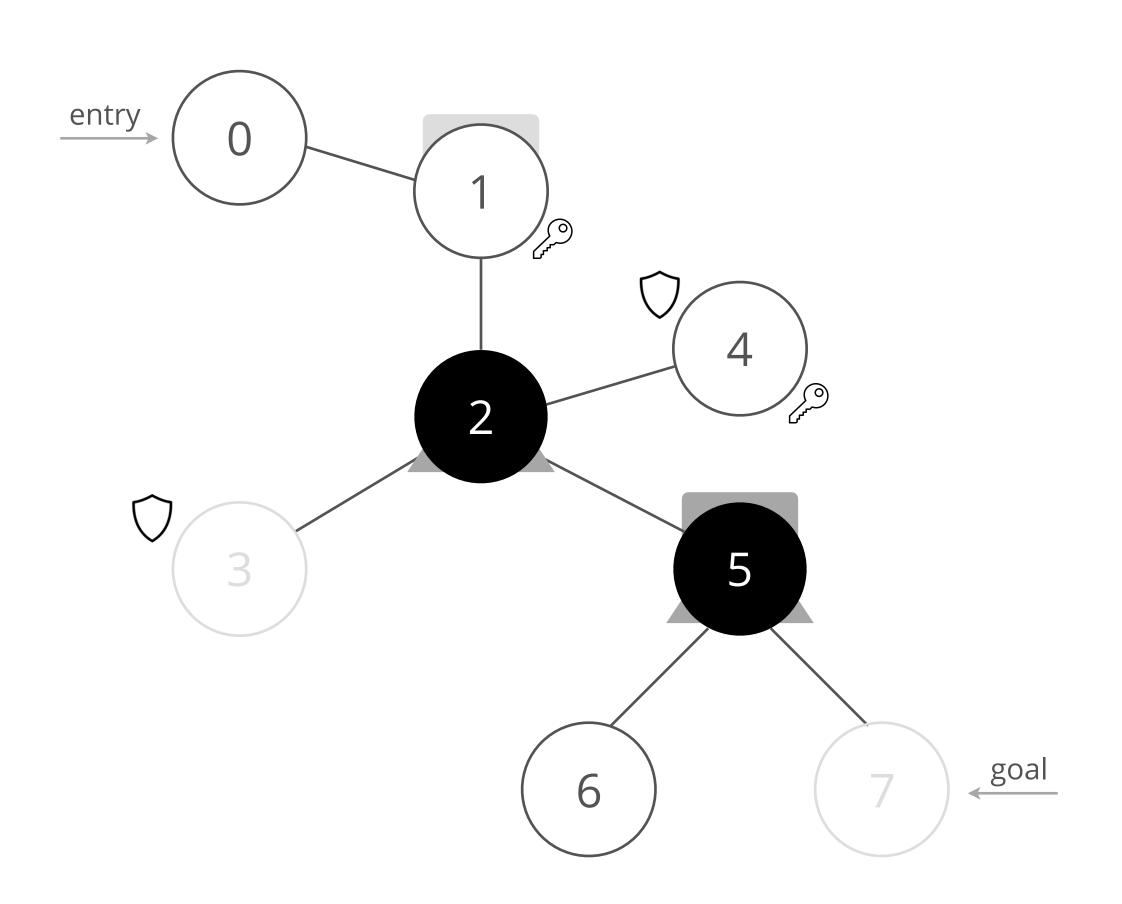
cleanup

persistence

Q dump

ACTION





CONNECTION

- --- undiscovered
- discovered

MACHINE STATUS

- unknown
- scanned
- foothold
- elevated

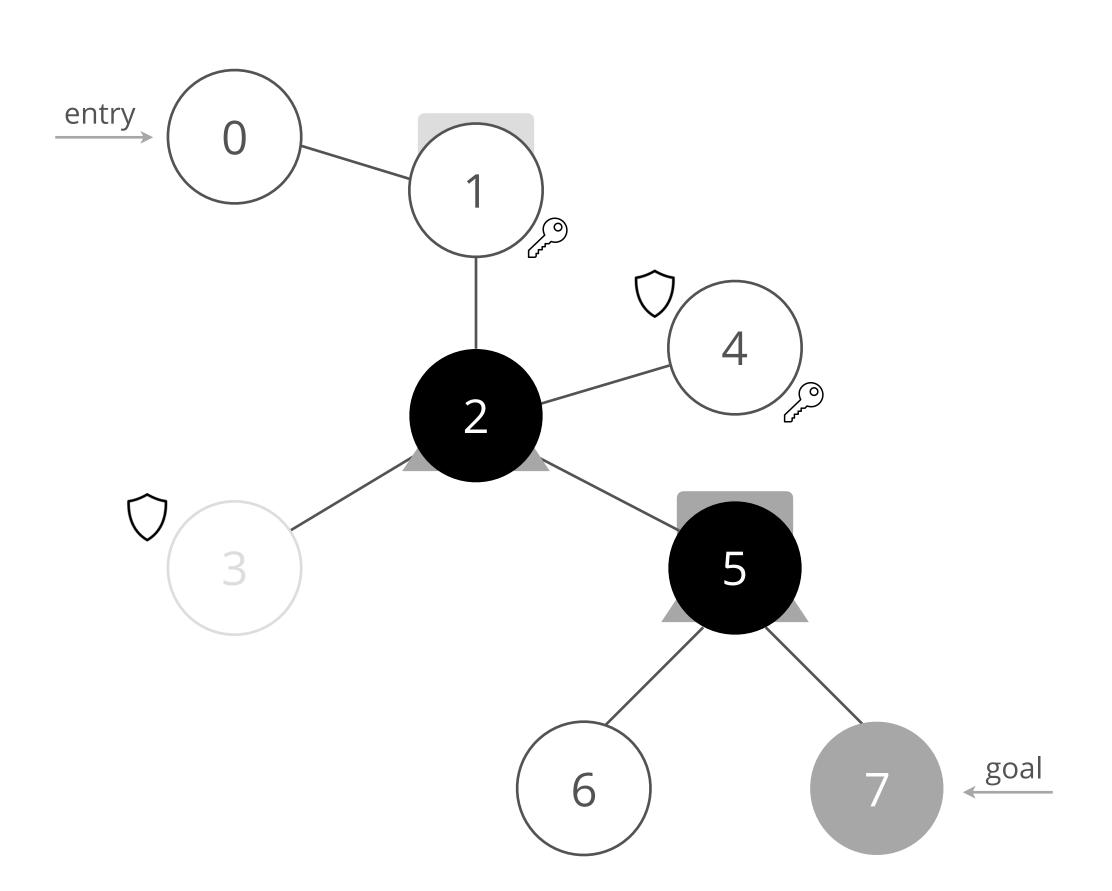
- cleanup
- persistence
- Q dump
- exfiltrate

ACTION

login: 7

1 🖫

+5 **(foothold)**



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

persistence

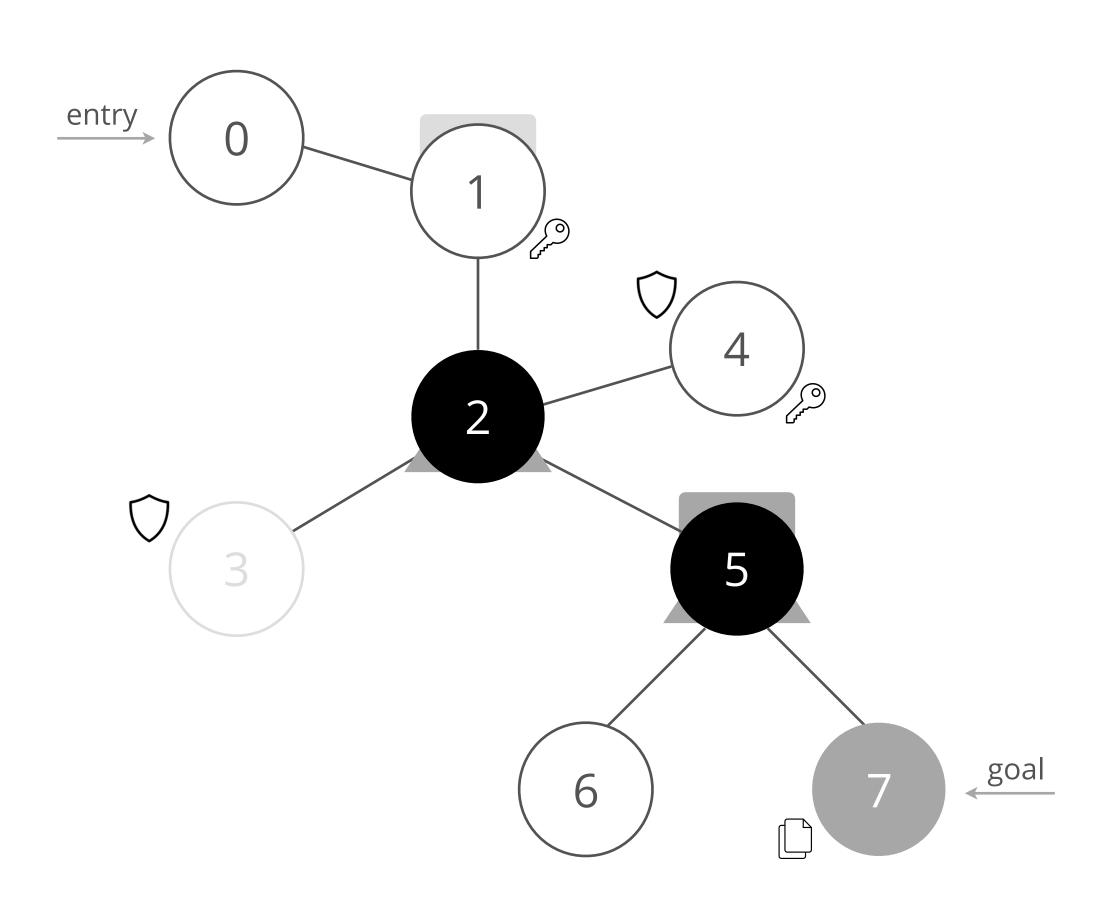
Q dump

ACTION

exfiltrate: 7

1 🗵

+100 ♥ (goal)



CONNECTION

--- undiscovered

discovered

MACHINE STATUS

unknown

scanned

foothold

elevated

PERFORMED

cleanup

persistence

Q dump

PERFORMANCE METRICS

- Objective reached:
 - exfiltrate data
 - compromise machines
 - steal credentials
- Stealthiness: times detected
- Swiftness: time steps taken
- Above encompassed by total reward
- Also encourage model parsimony

STATE

- Characterizes the env at the current time step
- Shown to the agent (what a human can deduce):
 - performed actions
 - machines compromised, connections discovered, user credentials obtained
 - available next actions
 - action properties
- Hidden:
 - network topology, position of the goal
 - what users have access where
 - defender strategy
 - grey agent probabilities

REWARDS

- Guides the agent towards desired objectives / penalizes unwanted behavior
- Positive (according to evaluation metrics / attacker objectives):
 - gain foothold (first time): small
 - exfiltrate data: small
 - exfiltrate goal: large
- Negative:
 - ▶ time: small to encourage swiftness
 - detection: large

HOW SCENARIOS DIFFER

- Network topology
- Machines exploitability
- Action properties
- Grey agent probabilities
- Defender's strength

SCENARIO GENERATION

- Connections and vulnerabilities: synthetization of typical networks
- Exploits properties: mapping of NVD database
- Action properties: estimated by security experts
- Vulnerabilities presence dictated by user kind:
 - Network administrator
 - Software developer
 - Non-technical employee

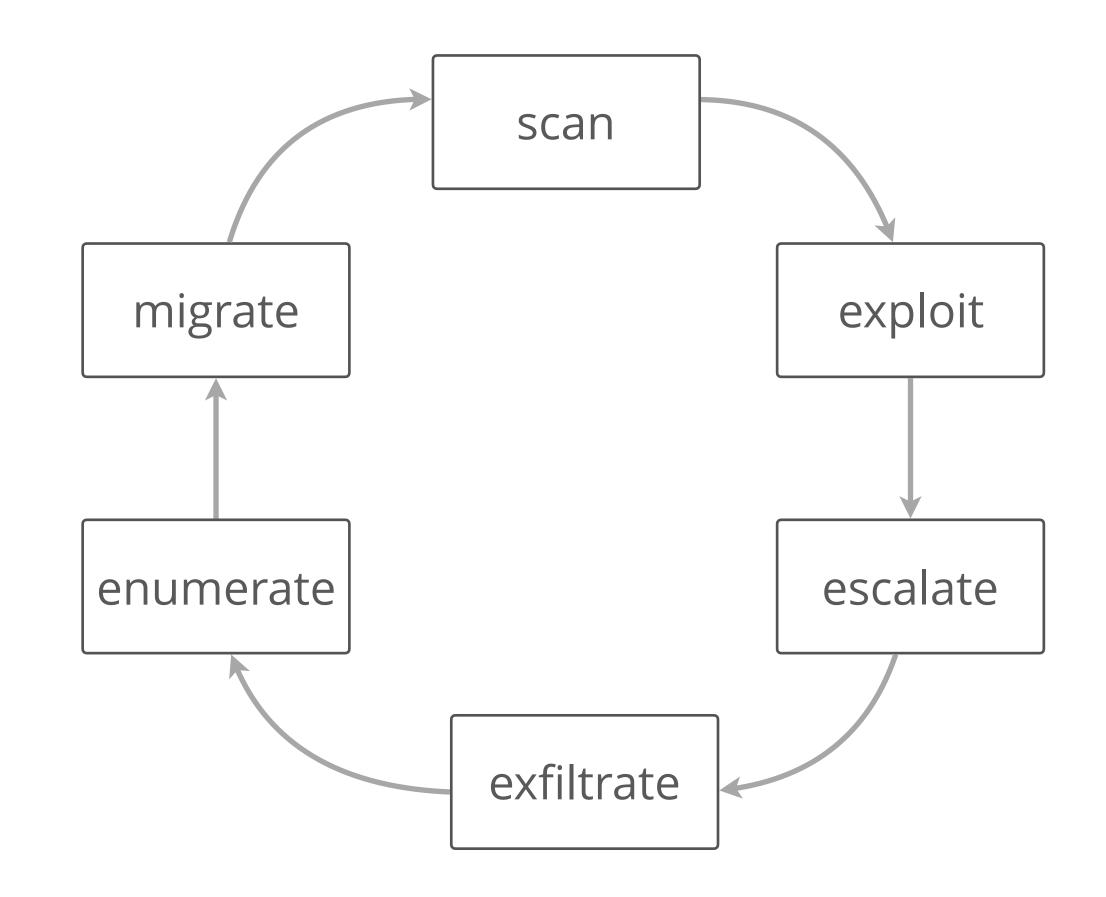


IMMEDIATE EXECUTING

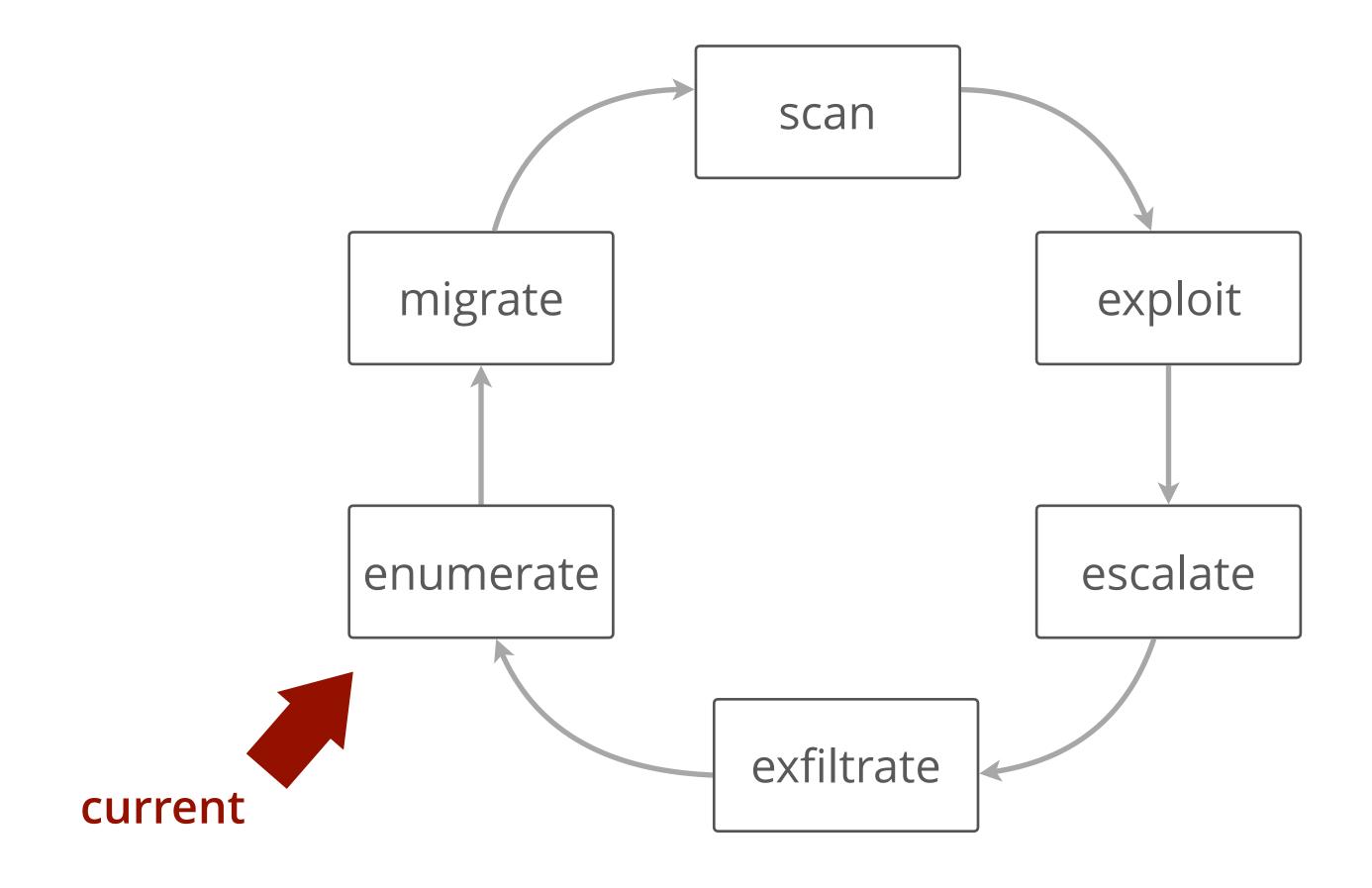
- Don't learn, fixed strategy
- Used as baseline
- Can provide initial knowledge for learning methods
- Simplest: Random Agent
 - picks one of the available actions, uniformly at random

Finite State Machine

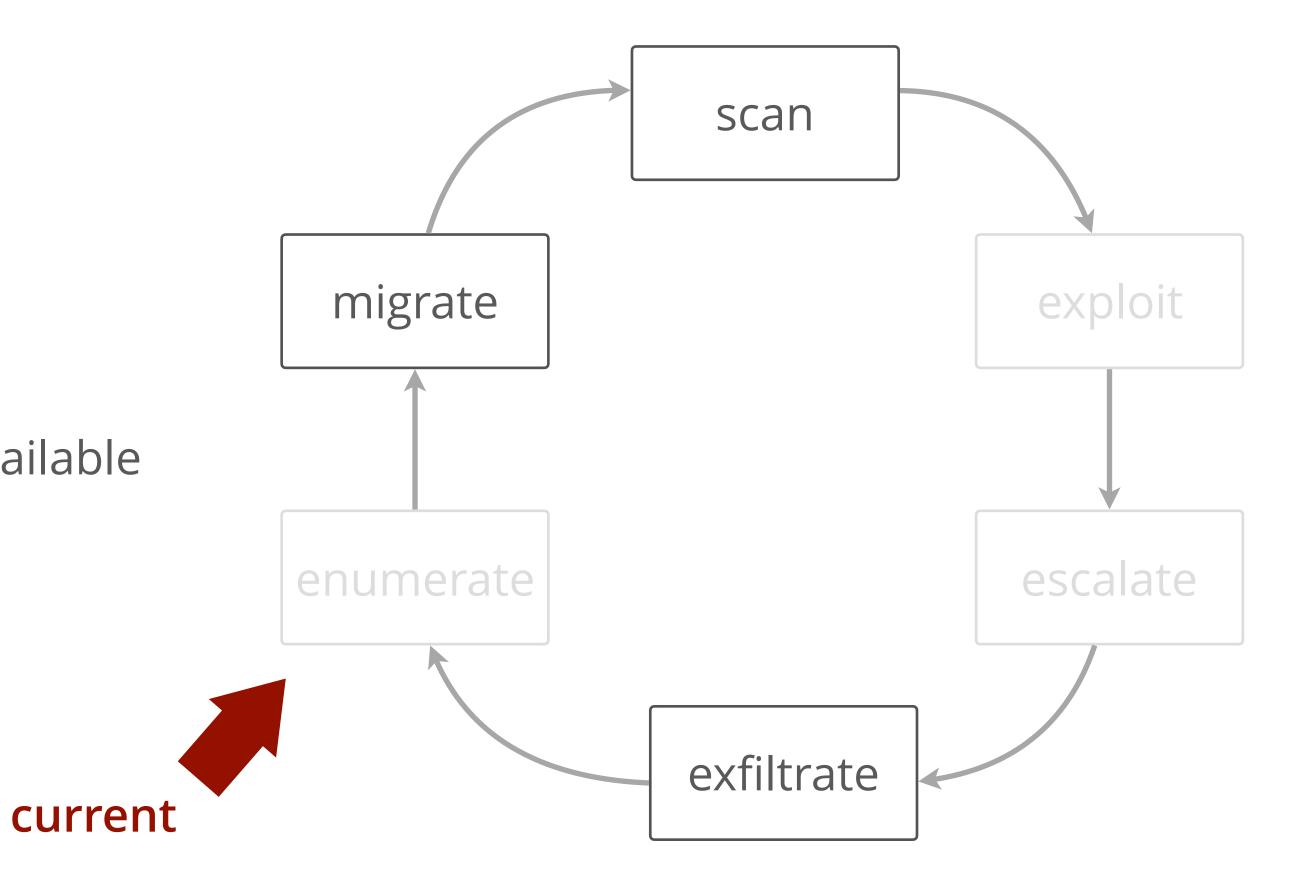
- Finite State Machine
- Has an order of actions



- Finite State Machine
- Has an order of actions
- Cycles through them

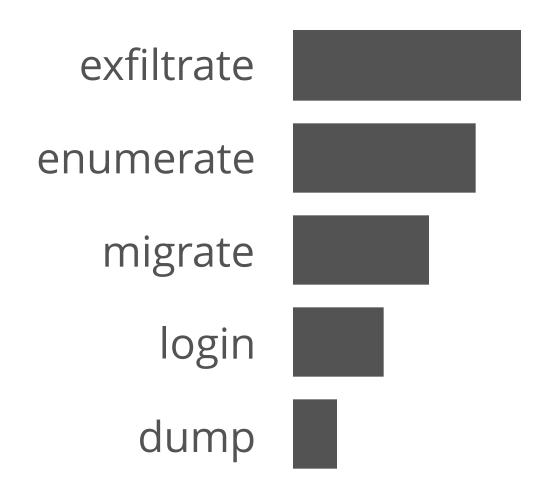


- Finite State Machine
- Has an order of actions
- Cycles through them
- Acts randomly if next one is unavailable



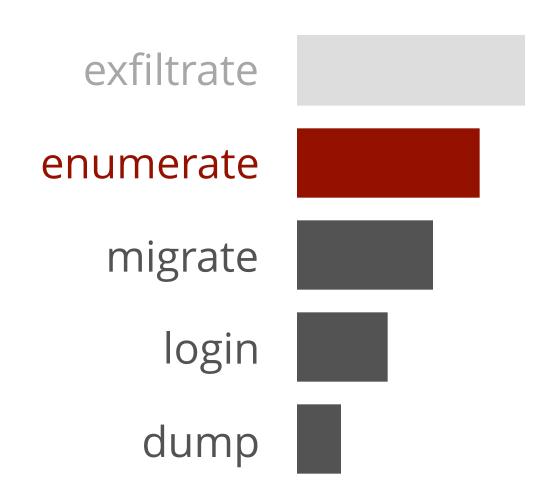
IMMEDIATE: GREEDY

Has a list of preferences



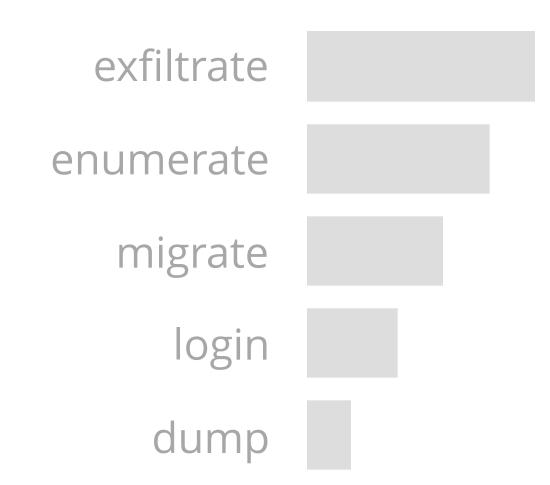
IMMEDIATE: GREEDY

- Has a list of preferences
- Always picks highest available



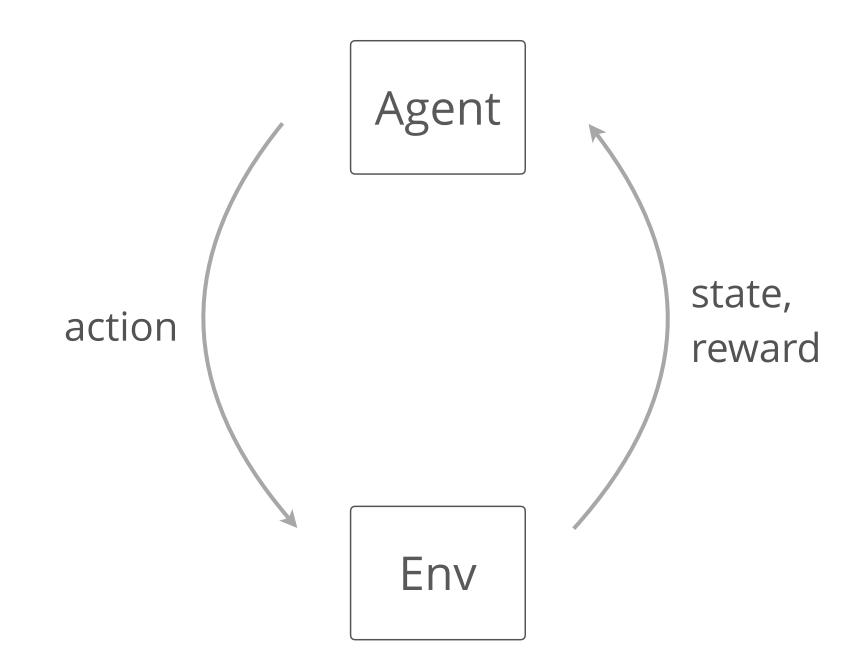
IMMEDIATE: GREEDY

- Has a list of preferences
- Always picks highest available
- Acts randomly when none available

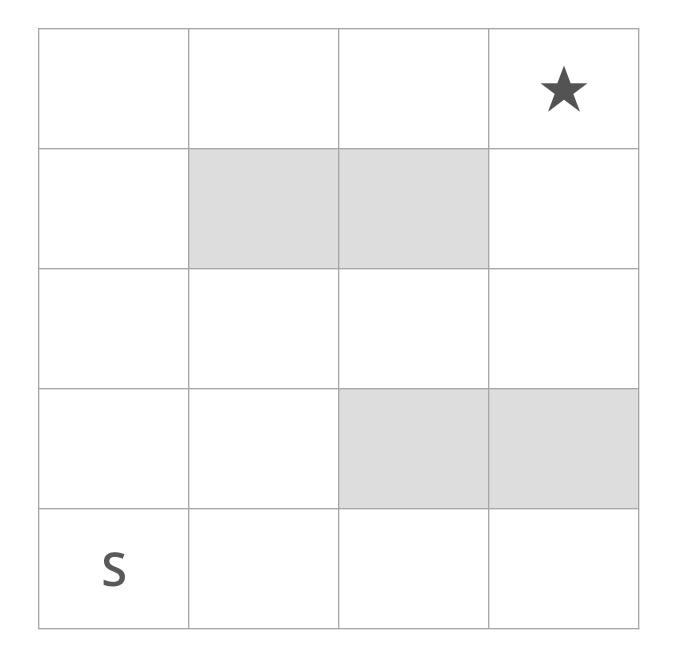


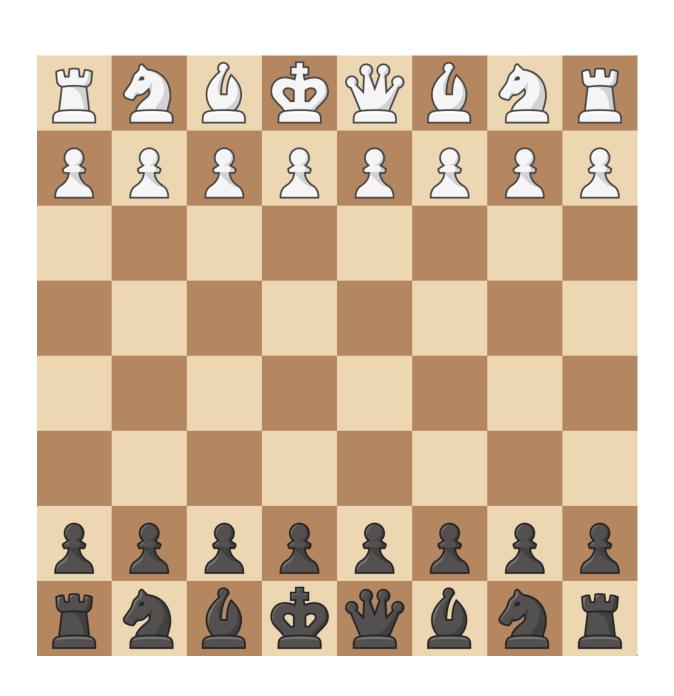
REINFORCEMENT LEARNING

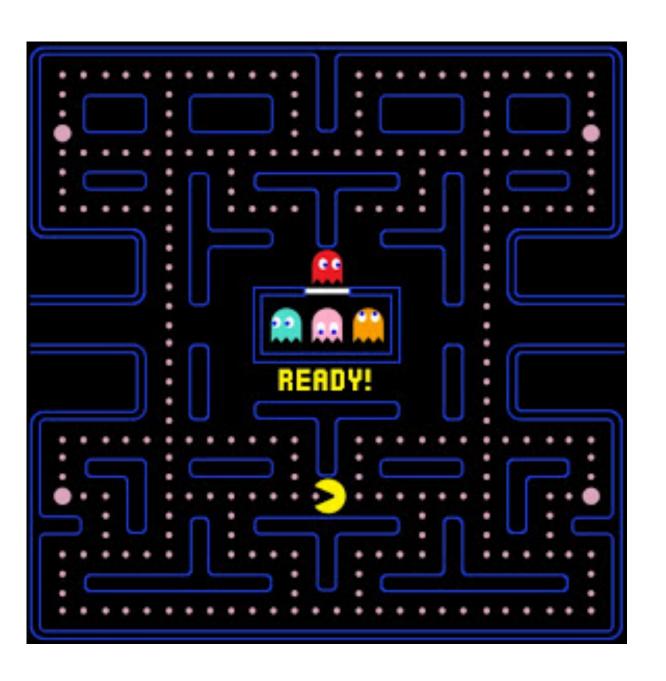
- Find best strategy for:
 - maximizing cumulative reward
 - in a sequential decision-making problem
- Agent in charge of:
 - learn from environment observations
 - steer the way new experience flows in



CLASSICAL SETTINGS

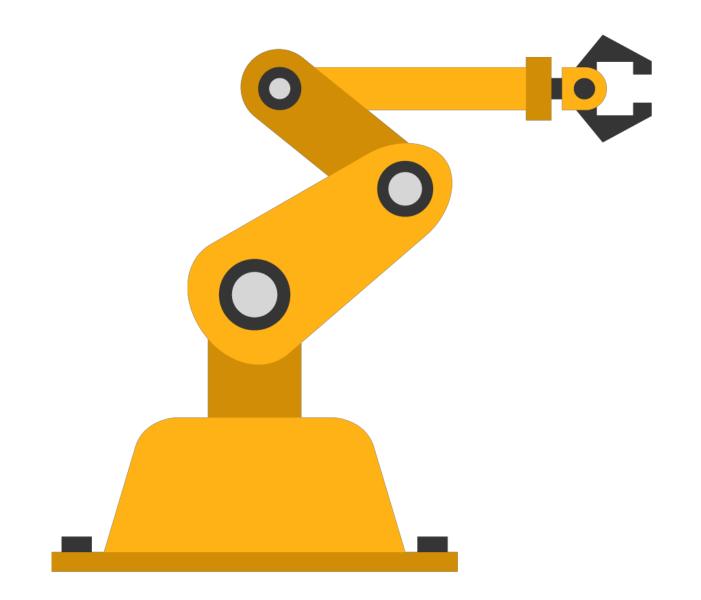






Toy: Grid-world Chess Modern: Pac-man

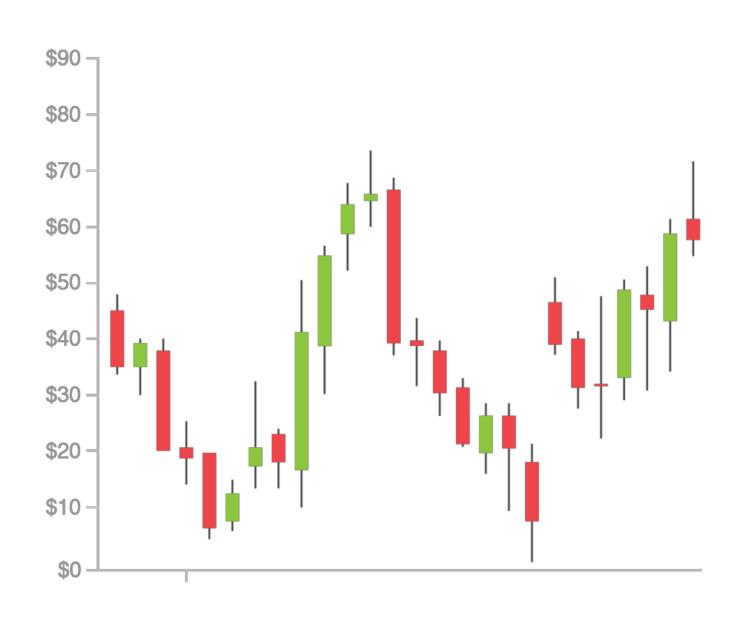
MORE SETTINGS



Robot Arm Control



Preventive Maintenance



Stock Exchange

SETTINGS FORMALIZATION

Setting	State	Actions	Rewards
Grid-world	x, y	$\leftarrow \rightarrow \uparrow \downarrow$	+1 on goal
Chess	pieces location	move pieces	+1 for victory, -1 for defeat
Pacman	player & enemies location	$\leftarrow \rightarrow \uparrow \downarrow$	+1 per pellet

SETTINGS FORMALIZATION

Setting	State	Actions	Rewards
Robot	x, y, z, velocity	activate motors	proximity to target
Maintenance	sensor measurements	continue / repair	–1 on accident
Stocks	companies info	buy / sell	profit

SETTINGS FORMALIZATION

Setting	State	Actions	Rewards
Pentesting	machines state	attacker actions	objective based

PENTESTING STATE

	enumer	scan	migrate	login	escalate	persist	dump	exfiltrate	cleanup	exploit ₁	exploit ₂	• • •
m_1					√	0	0	0	√	0	√	
m_2		0	0		0				0			
m_3					√	√		0	0		0	
m_4	0	0		0	√	0	0	√	√			
• • •												

	evade	wait	abandon
n/a		0	0

✓ performed

o available

- If you know how valuable each action is, in each possible state
- Then you have solved the problem: always select most valuable action

- Number each possible state from 0 to n
- Q table of value for each action, in each state

Action **a**₃ **a**₄ **a**₁ a_2 **S**1 **S**₂ **S**3 **S**4 **S**5 • • • • • • Sn

source: [1]

- Number each possible state from 0 to n
- Q table of value for each action, in each state
- Initial values: arbitrary
 - zero

			Action						
	_	a ₄	a ₃	a ₂	a ₁				
	S ₁	0	0	0	0				
	S ₂	0	0	0	0				
S	S 3	0	0	0	0				
State	S ₄	0	0	0	0				
	S 5	0	0	0	0				
	• • •		• •	•					
	Sn	0	0	0	0				

- Number each possible state from 0 to n
- Q table of value for each action, in each state
- Initial values: arbitrary
 - zero
 - non-zero to incentivize exploration

		Action						
		a ₄	a ₃	a ₂	a ₁			
	S ₁	0	3	0	1			
	S ₂	2	2	1	1			
S	S 3	1	1	3	3			
State	S ₄	3	1	3	2			
	S 5	3	2	1	3			
	• • •		•	• •				
	Sn	1	3	1	0			

- Number each possible state from 0 to n
- Q table of value for each action, in each state
- Initial values: arbitrary
 - zero
 - non-zero to incentivize exploration
 - can imbue apriori knowledge

			tion	Ac	
		a ₄	a ₃	a ₂	a ₁
	S ₁	3	0	7	0
	S ₂	3	0	7	0
Ŋ	S 3	3	0	7	0
State	S ₄	3	0	7	0
	S 5	3	0	7	0
	• • •		• •	•	
	Sn	3	0	7	0

- Based on observation s, a, r, s'
 - ▶ After taking action *a* in state *s*
 - Receiving reward r and landing on state s^{\prime}
 - eg: s_2 , a_3 , +8, s_4

		Action						
		a ₄	a ₃	a ₂	a ₁			
	S ₁	2	2	7	0			
	S ₂	4	5	0	6			
S	S 3	6	9	3	3			
State	S ₄	4	3	3	8			
	S 5	0	1	2	6			
	• • •		• •	•				
	Sn	8	2	5	4			

- Based on observation s, a, r, s'
 - ▶ After taking action *a* in state *s*
 - Receiving reward r and landing on state s'
 - eg: s_2 , a_3 , +8, s_4
- Update: $Q(s, a) \rightarrow r$

		Action					
		a 4	a ₃	a ₂	a ₁		
	S ₁	2	2	7	0		
	S ₂	4	5	0	6		
V	S 3	6	9	3	3		
State	S ₄	4	3	3	8		
	S 5	0	1	2	6		
	• • •		• •	•			
	Sn	8	2	5	4		

- Based on observation s, a, r, s'
 - ▶ After taking action *a* in state *s*
 - Receiving reward r and landing on state s'
 - eg: s_2 , a_3 , +8, s_4
- Update: $Q(s, a) \rightarrow r + maxQ(s', a_i)$
- Approximate future value Q(s')
 - ► $max \Rightarrow$ brave: judge itself by best case
 - $avg \Rightarrow$ cautious: judge itself by most probable
 - \blacktriangleright linear combination of the two, controlled by η

a ₁	a ₂	a ₃	a ₄	_	
0	7	2	2	S ₁	
6	0	5	4	S ₂	
3	3	9	6	S 3	S
8	3	3	4	S ₄	State
6	2	1	0	S ₅	
	•	• •		• • •	
4	5	2	8	Sn	

- Based on observation s, a, r, s'
 - ▶ After taking action *a* in state *s*
 - ▶ Receiving reward *r* and landing on state *s* ′
 - eg: s_2 , a_3 , +8, s_4
- Update: $Q(s, a) \rightarrow r + maxQ(s', a_i) \times \gamma$
- Approximate future value Q(s')
 - ► $max \Rightarrow$ brave: judge itself by best case
 - $avg \Rightarrow$ cautious: judge itself by most probable
- Discount future rewards by γ
 - ▶ small ⇒ visionary: care about future returns
 - ▶ large ⇒ hedonistic: prefer immediate reward

		Action						
		a ₄	a ₃	a ₂	a ₁			
	S ₁	2	2	7	0			
	S ₂	4	5	0	6			
S	S 3	6	9	3	3			
State	S 4	4	3	3	8			
_	S ₅	0	1	2	6			
	• • •		• •	•				
	Sn	8	2	5	4			

RL: TABULAR — ACT

- Based on the current state *s*
- Pick an action *a*

Action a_1 a_2 a_3 **a**₄ 0 **S**1 5 6 0 **S**2 3 3 **S**3 3 8 3 **S**4 6 **S**5 • • • • • • 5 Sn

RL: TABULAR — ACT

- Based on the current state *s*
- Pick an action *a*
- Select the most valuable one:

$$a = argmax Q(s, a_i)$$

		Action						
		a ₄	a ₃	a ₂	a ₁			
	S ₁	2	2	7	0			
	S ₂	4	5	0	6			
S	S 3	6	9	3	3			
State	S 4	4	3	3	8			
	S 5	0	1	2	6			
	• • •	• • •						
	Sn	8	2	5	4			

RL: TABULAR — ACT

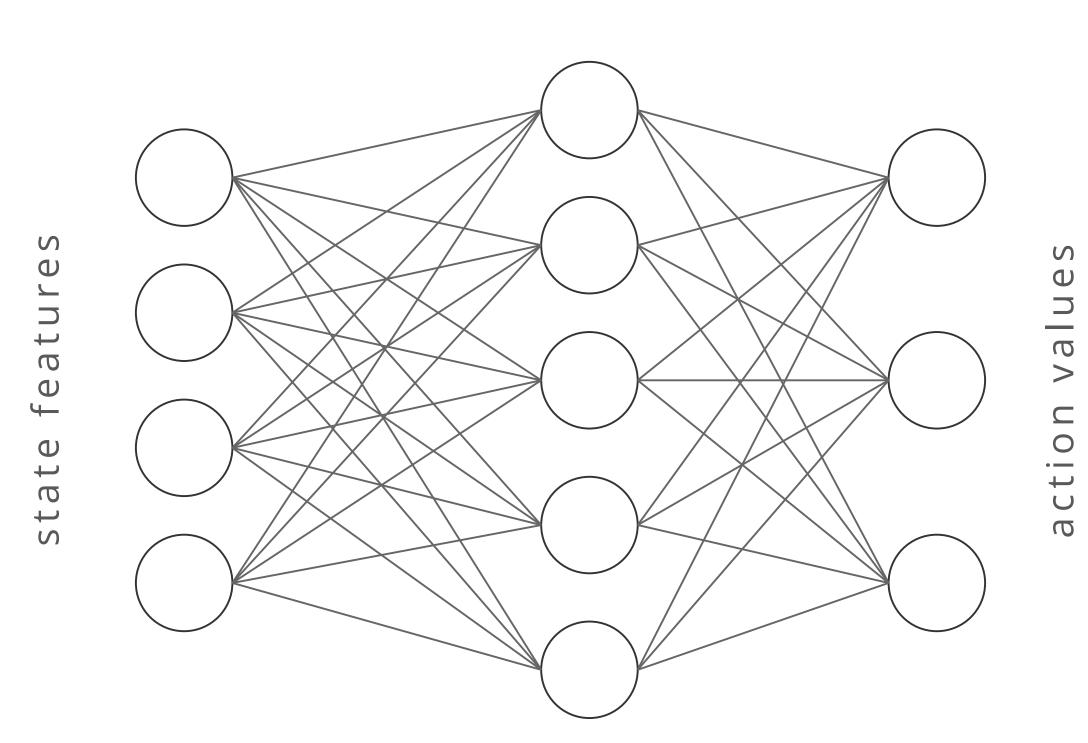
- Based on the current state *s*
- Pick an action *a*
- Select the most valuable one:

$$a = argmax Q(s, a_i)$$

- Sometimes, explore other options:
 - ightharpoonup pick randomly, with probability arepsilon
 - allows discovery of better paths

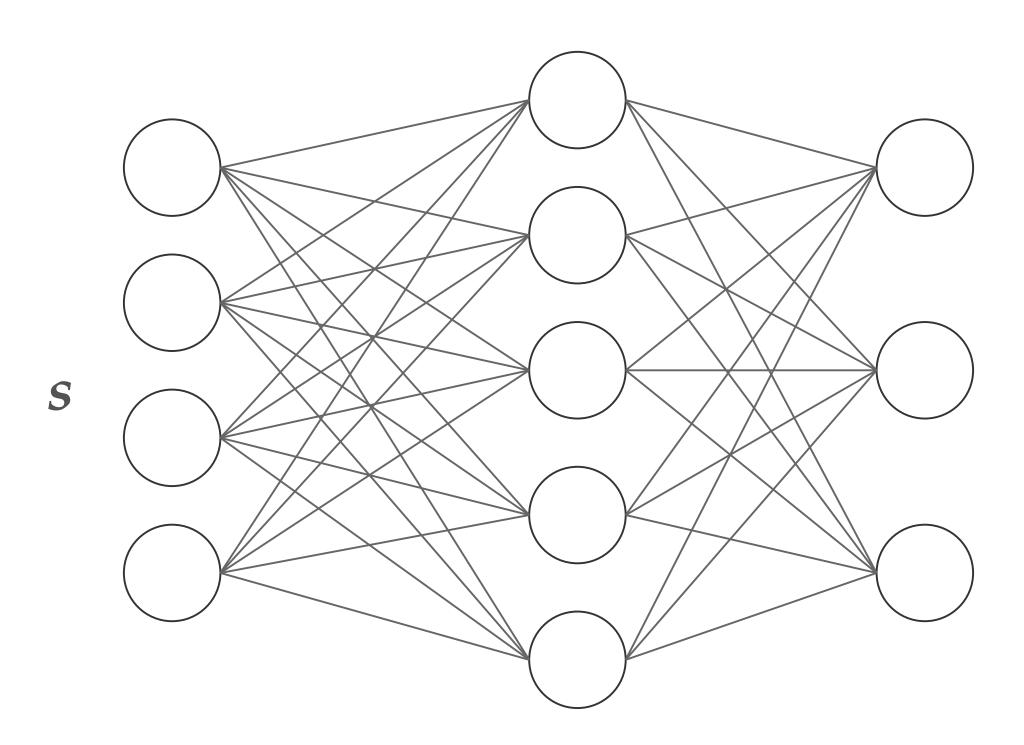
a ₁	a ₂	a ₃	a ₄		
0	7	2	2	S ₁	
6	0	5	4	S ₂	_
3	3	9	6	S 3	S
8	3	3	4	S ₄	State
6	2	1	0	S 5	
	• • •				
4	5	2	8	Sn	

- Replace lookup table with a predictive model
- Works with state features
 - leverage state similarities
 - enables generalization

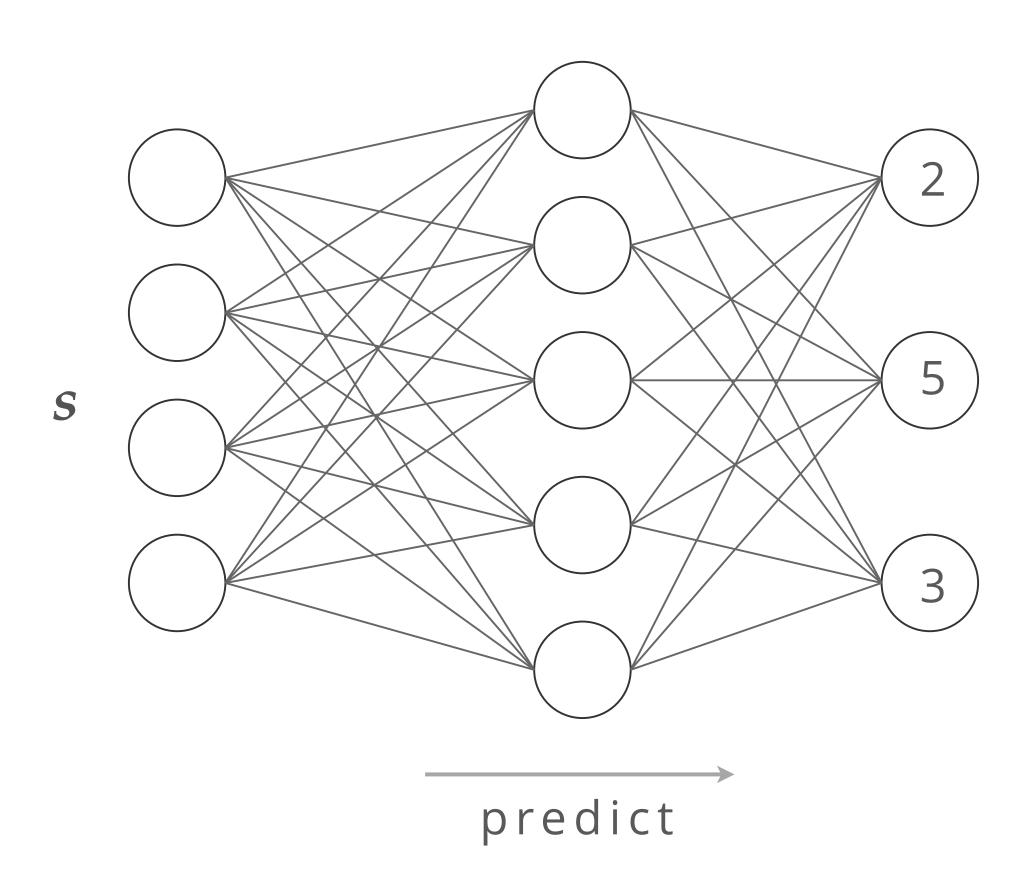


source: [1]

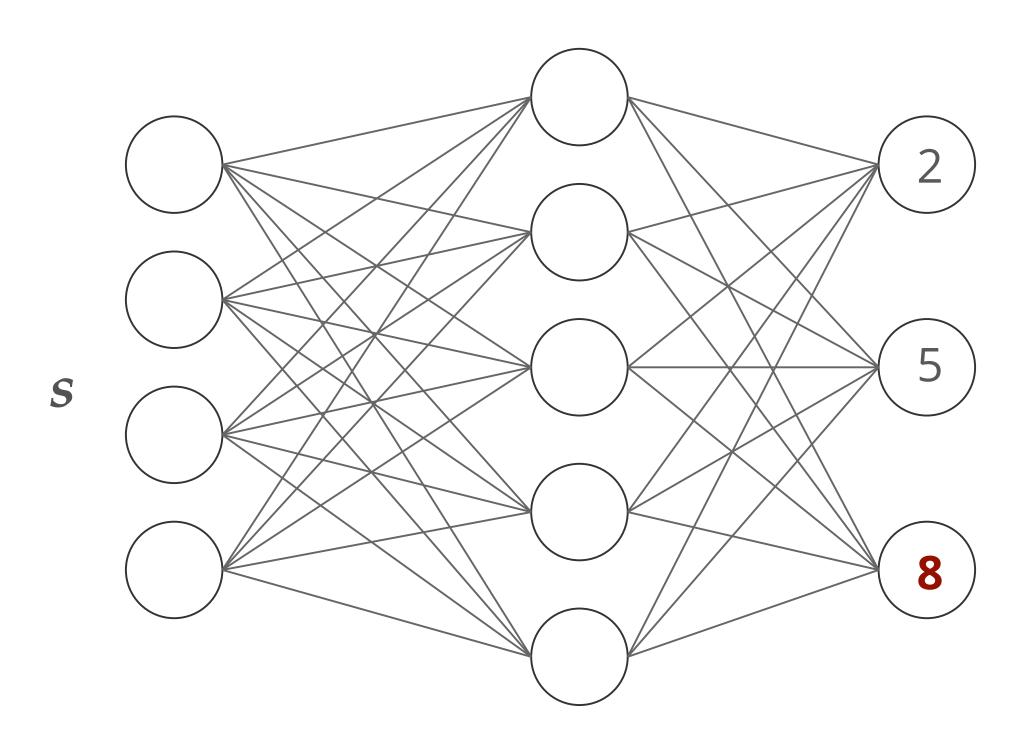
Based on observation s, a, r, s'



- Based on observation s, a, r, s'
- 1. Predict value of each action y = net(s)

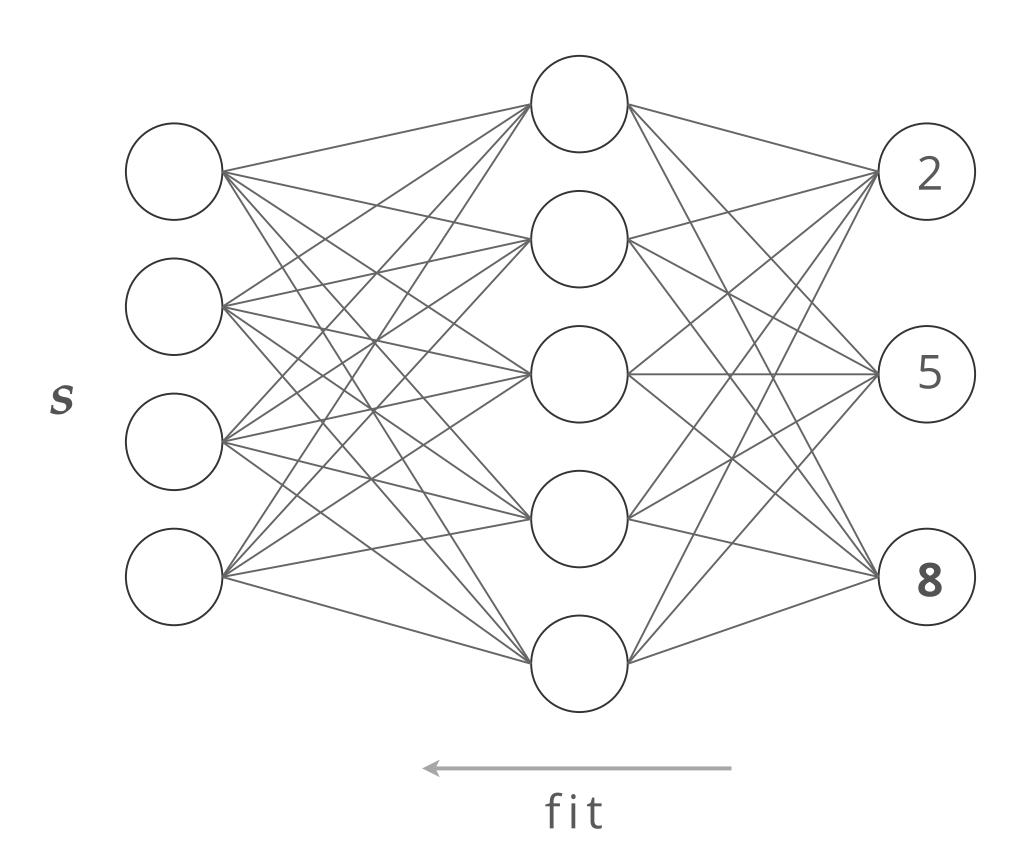


- Based on observation s, a, r, s'
- 1. Predict value of each action y = net(s)
- 2. Compute target g (same as Q update) $g = r + max \ net(s') \times \gamma$
- 3. Set observed action's value $y_a = g$



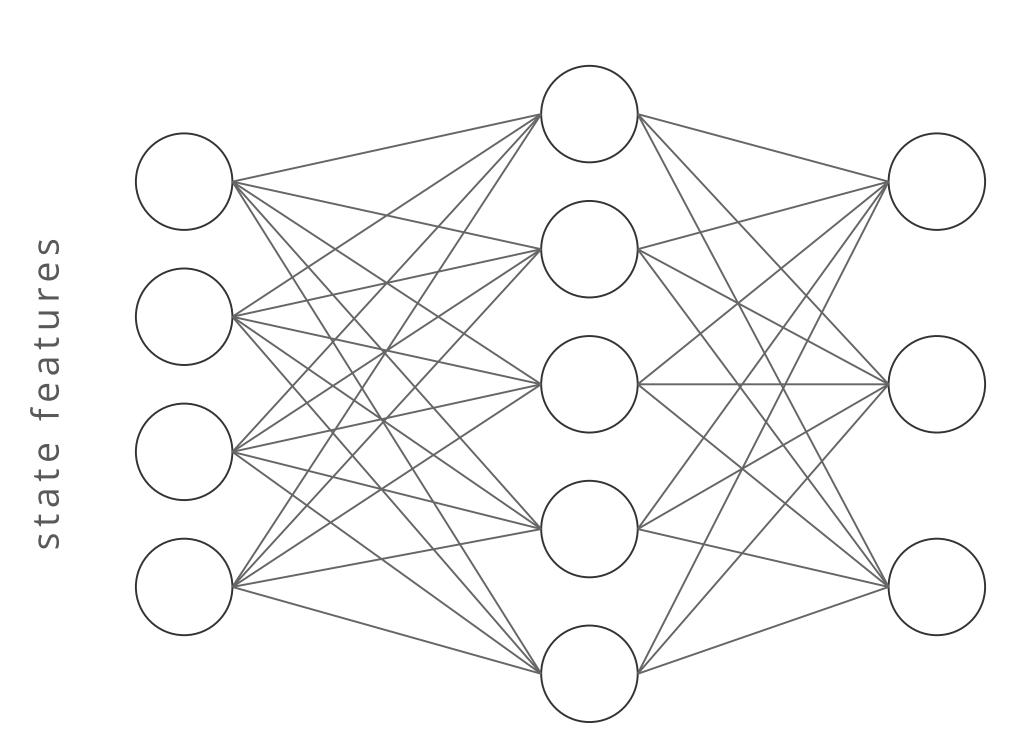
update

- Based on observation s, a, r, s'
- 1. Predict value of each action y = net(s)
- 2. Compute target g (same as Q update) $g = r + max \ net(s') \times \gamma$
- 3. Set observed action's value $y_a = g$
- 4. Train model on new pair (s, y)



RL: APPROXIMATE — ACT

- Based on the current state *s*:
- 1. Predict value of each action y = net(s)
- 2. Pick most valuable action a = argmax y
- ullet Sometimes (probability arepsilon) pick randomly



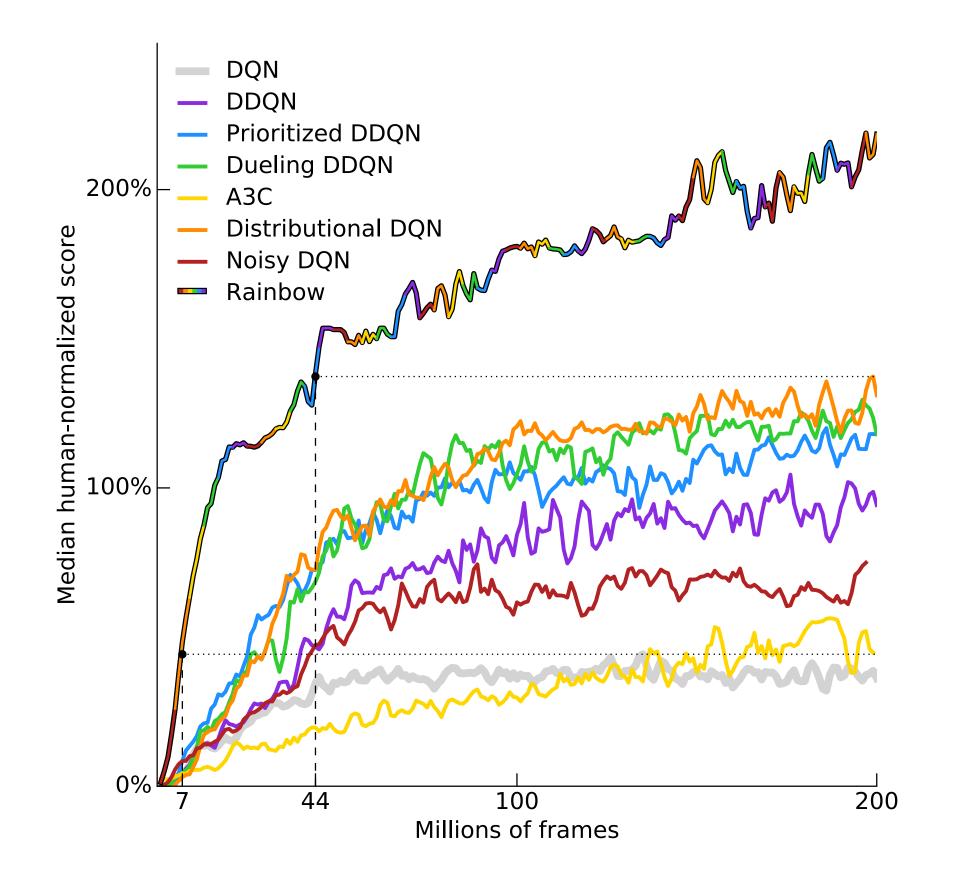
RL: APPROXIMATE — SOME EXTENSIONS

- Multi-step Returns: look ahead multiple time steps when estimating action value
- Double (2015): use a second network for predictions, updated slowly towards main network
- **Dueling** (2015): decouple Q(s, a) into state value V(s) and action advantage A(a)
- Prioritized Experience Replay (2015): favor transitions the network can learn the most from
- Bayesian Networks (2015): handle uncertainty by approximating a GP using dropout
- Bolzmann Exploration (2017): sample actions proportional to estimated values
- Distributional (2017): learn a distribution, instead of approximating a single q value
- Noisy Nets (2018): add parametric noise to weights
- Asynchronicity (2018): multiple independently interacting agents

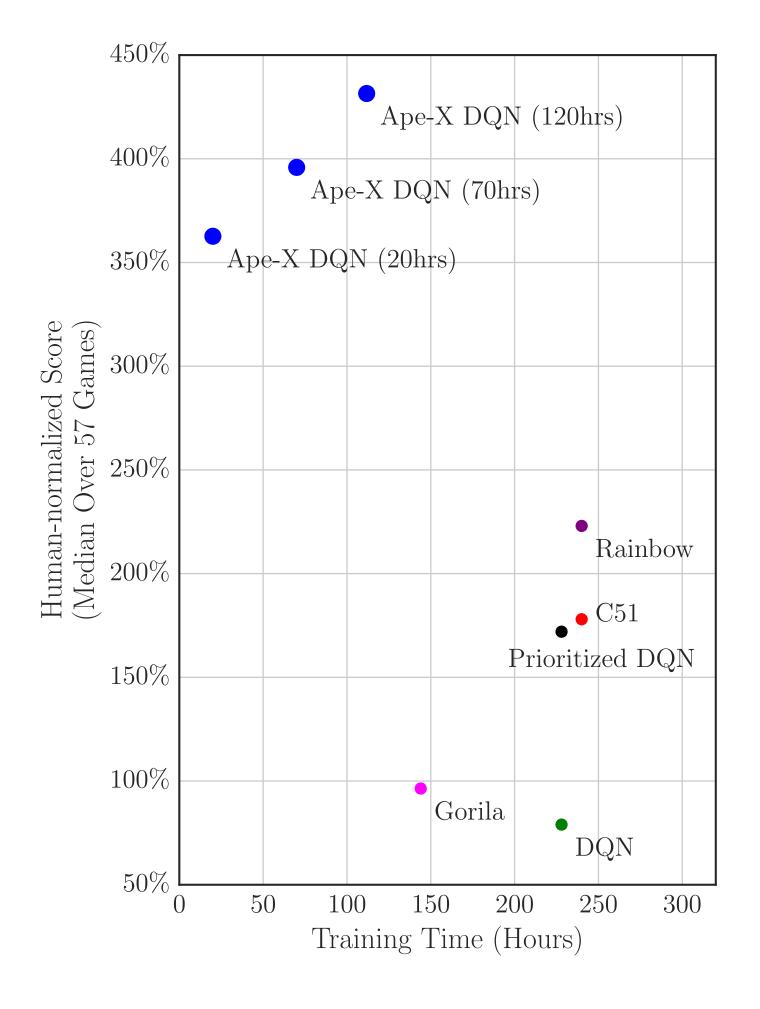
RL: APPROXIMATE — SPINOFFS

- Policy Gradients: learn policy directly, as a distribution over actions added stochasticity is essential in partially observable scenarios
- Actor-Critic: combine value iteration (QL) with policy iteration (policy gradient) also compute how much better actions turned out to be than expected
- Many more: Hierarchical RL, Recurrent DQN, Intrinsic Motivation, Trust-Region Optimization, etc.

EXTENSIONS IMPACT



source: [4]



GENETIC ALGORITHMS

- Use GA as the discovery mechanism: many (guided) random individuals
- And the update mechanism: evolve fittest using genetic operators

Rule-Based ML

- Rule-Based ML
- Binary encode states and actions



- Rule-Based ML
- Binary encode states and actions
- Creates mapping: state ~ action



- Rule-Based ML
- Binary encode states and actions
- Creates mapping: state ~ action
- Simple *if-then* rules



- Rule-Based ML
- Binary encode states and actions
- Creates mapping: state ~ action
- Simple *if-then* rules
- Wildcard character #



- Individual := single classifier
- Population := multiple classifiers
- Solution := entire population
- Rule := context dependent relationship state ~ action
- Classifier = a rule and its properties:
 - fitness
 - age
 - reward-prediction accuracy
 - descriptive statistics: avg/max/etc
 - numerosity

GA: LCS — LEARN

- Based on observation s, a, r, s'
 - ▶ Classifier *A* acted in state *s*
 - lacktriangleright Classifier B acted in state s'
- Update A's fitness (same as Q update)

$$F_A \rightarrow r + F_B \times \gamma$$

• Update classifier's age & statistics

	State			Act	Fitness	
0	1	0	~	0	0	2
#	0	#	~	1	1	4
0	1	1	~	1	1	3
0	1	0	~	0	1	3
#	1	1	~	1	0	1

GA: LCS — EVOLVE

- Initial population empty
- Apply genetic operators:
 - Highly elitist (most of population retained)
 - ▶ Binary *crossover* & *mutation*
- Cleanup
 - Subsumption: merge redundant classifiers
 - ▶ Deletion: *select* inversely proportional to fitness

Based on the current state s

• eg: 0 1 1

	State		-	Action		Fitness
0	1	0	~	0	0	2
#	0	#	~	1	1	4
0	1	1	~	1	1	3
0	1	0	~	0	1	3
#	1	1	~	1	0	1

- Based on the current state *s*
 - eg: 0 1 1
- Find matching actions

State				Act	Fitness	
0	1	0	~	0	0	2
#	0	#	~	1	1	4
0	1	1	~	1	1	3
0	1	0	~	0	1	3
#	1	1	~	1	0	1

- Based on the current state *s*
 - eg: 0 1 1
- Find matching actions
- Select classifier proportional to fitness

State				Act	Fitness	
0	1	0	~	0	0	2
#	0	#	~	1	1	4
0	1	1	~	1	1	3
0	1	0	~	0	1	3
#	1	1	~	1	0	1

- Based on the current state *s*
 - eg: 0 1 1
- Find matching actions
- Select classifier proportional to fitness
- If no rule matches, cover:
 - ▶ introduce randomly generated rule

	State			Act	ion	Fitness
0	1	0	~	0	0	2
#	0	#	~	1	1	4
0	1	1	~	1	1	3
0	1	0	~	0	1	3
#	1	1	~	1	0	1

IMPORTANT CONSIDERATIONS

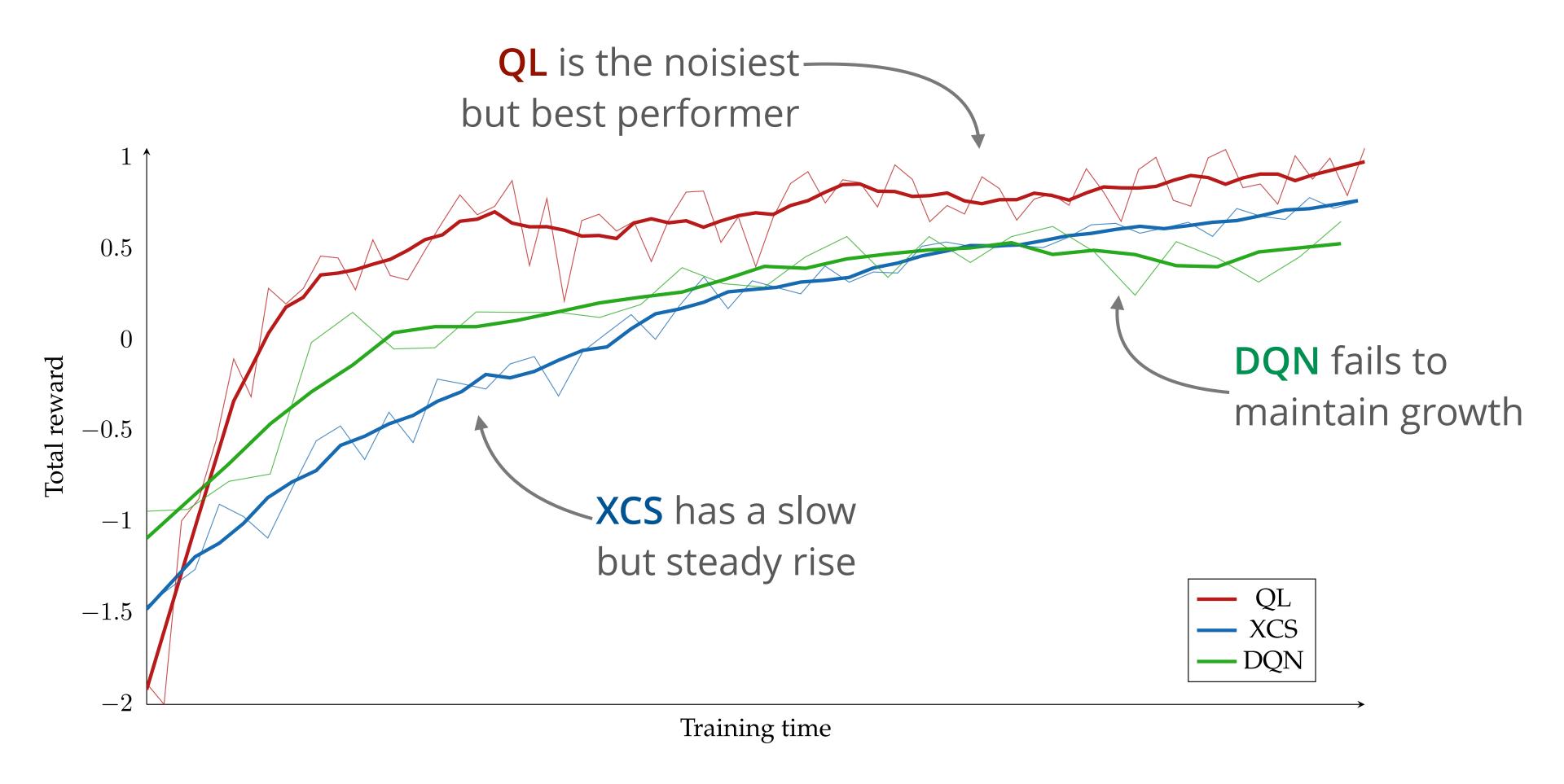
- Feature selection
 - immediate agents (zero features) do better than learning agents with all features
 - but learning agents overcome with a subset of features
- Exploratory starts
 - don't learn from scratch
 - have the immediate agents' experience as a starting point
- Rewards range
 - some agents deal best in the (-1, +1) range

IGNORED TECHNIQUES

- Classical planning
- Perfect knowledge simulation

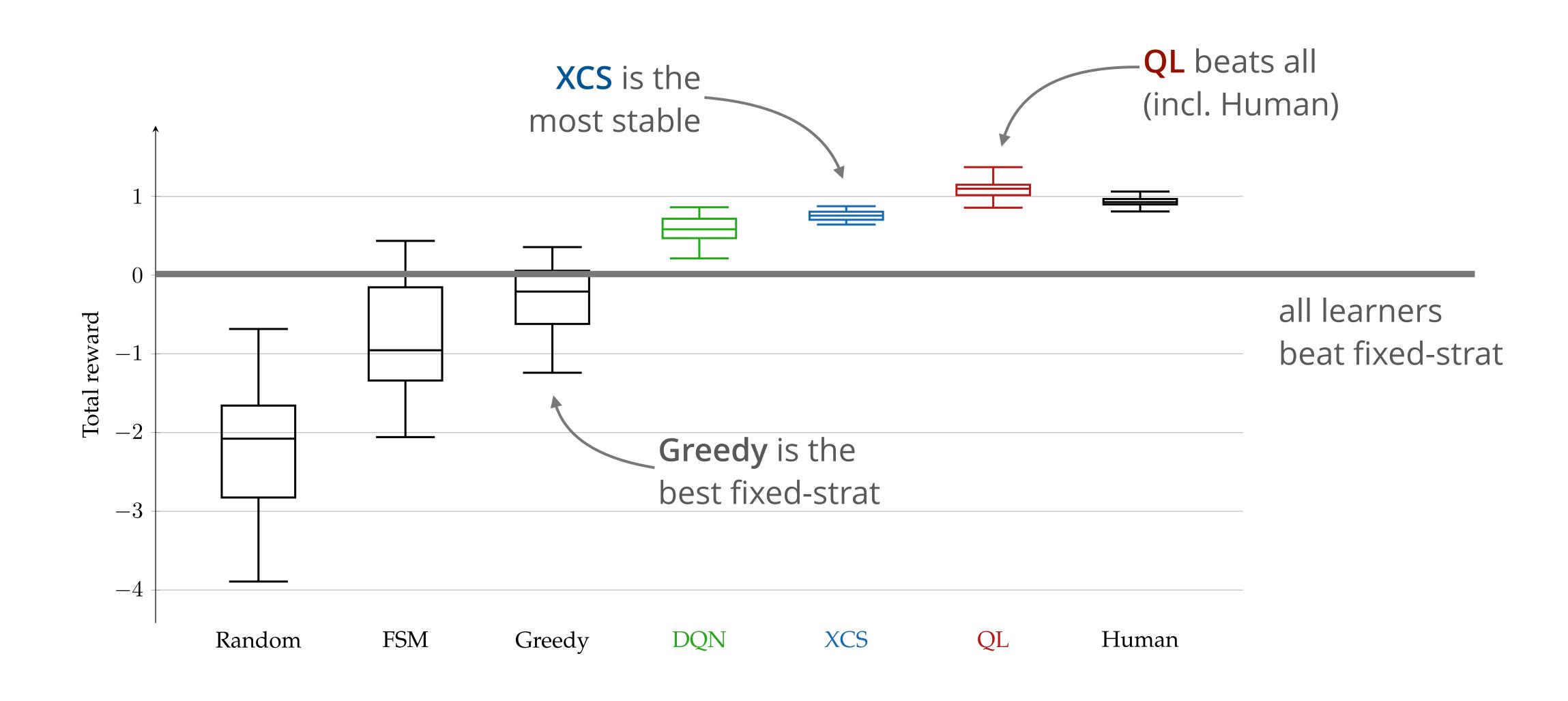


LEARNING BEHAVIOR

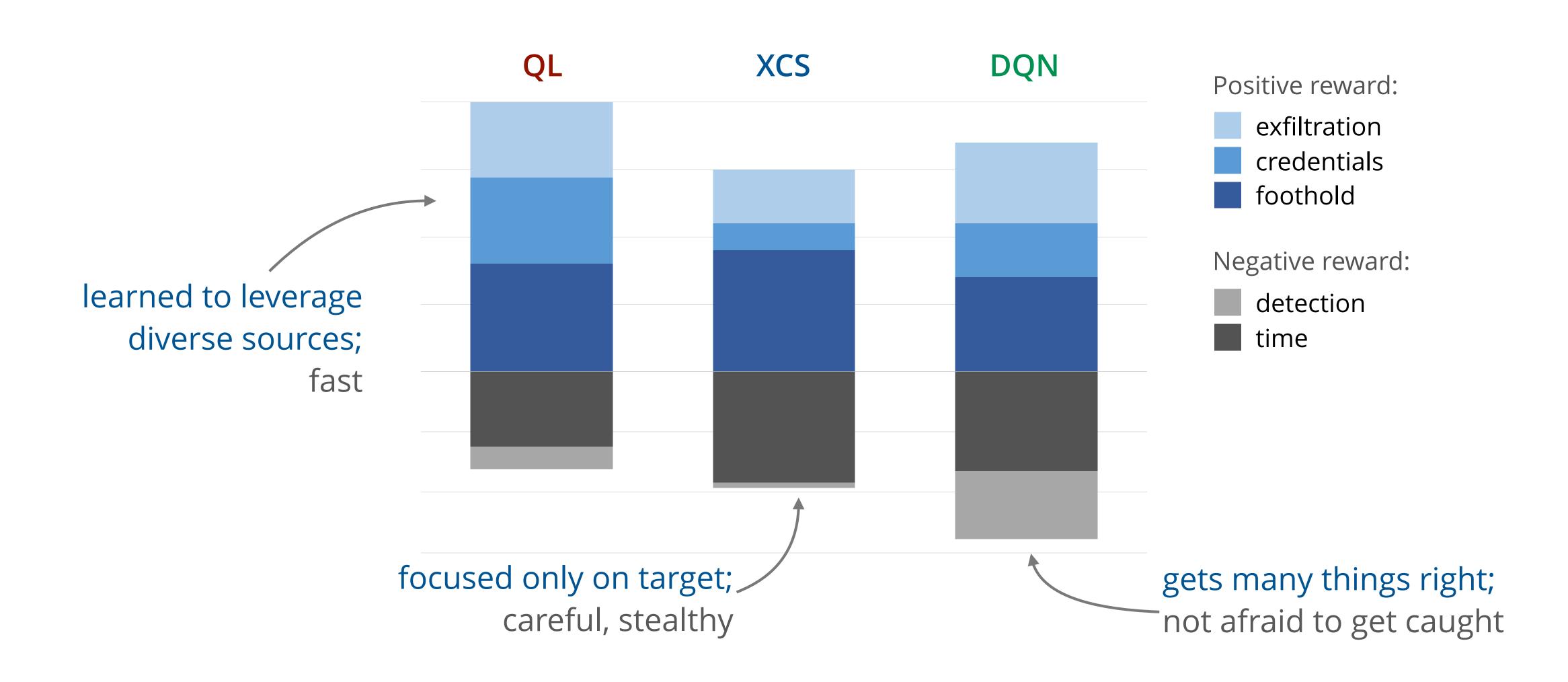


AXIS VALUES!

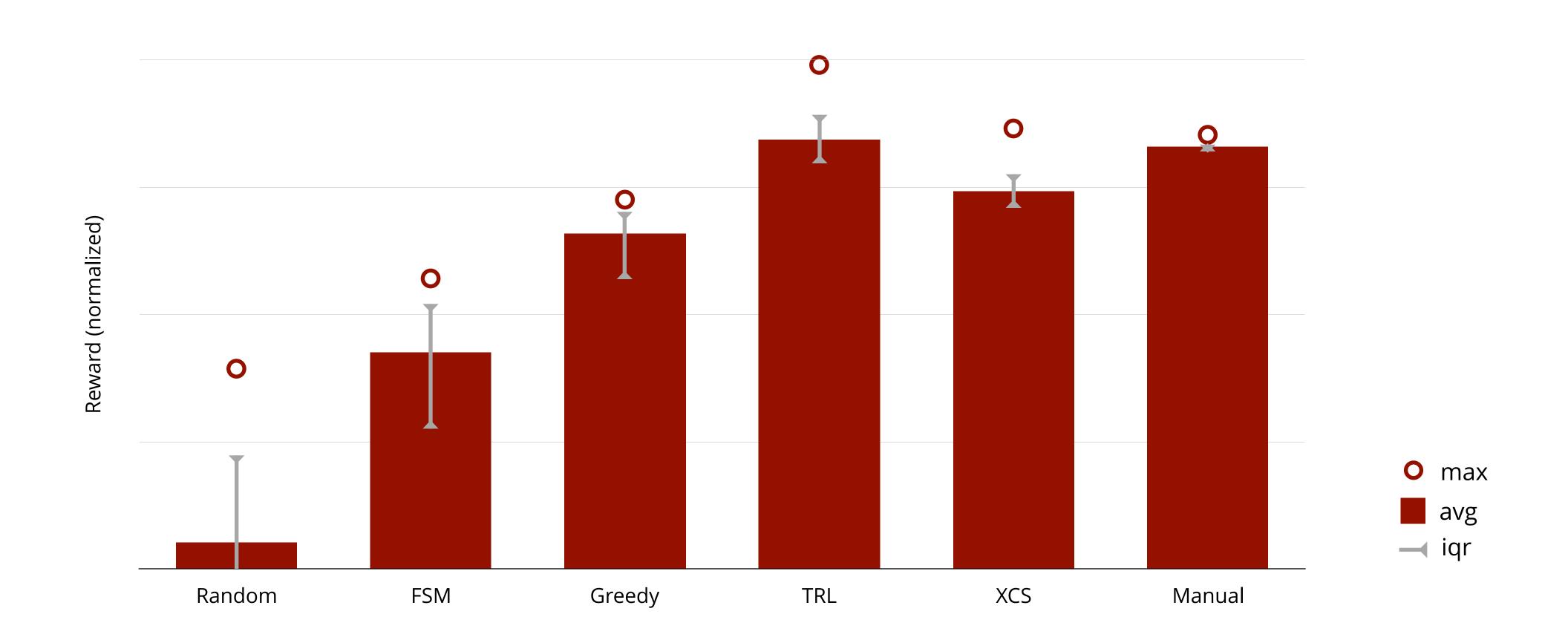
AGENTS PERFORMANCE



OBJECTIVES FOCUS



ALGORITHMS COMPARISON





CHALLENGES

- Game definition
 - Very few relevant papers
 - ▶ Hard to balance real-world closeness / implementation feasibility
- Algorithms fit:
 - Dissimilarity from classical RL application
 - Not evident state definition
 - Sparsity of rewards
 - Computational (in)efficiency of simulating the environment
- Algorithms training:
 - ▶ Large number of hyper-parameters (up to 34); very high training times (over 2 days)
 - No mature model-selection frameworks; not straightforward feature selection
 - ▶ Lack of generally-applicable advanced RL / efficient GA methods implementations

CONCLUSIONS

- Learning-based methods surpass fixed-strategy ones
- QL manages to overcome human performance
- Diverse strategies could be valuable in emulating various attacker types:
 - QL is fast and makes use of many techniques
 - XCS is methodical and stealthy
 - DQN is effective but also reckless
- Introduced hyper-parameter η proved useful for all algorithms, balancing caution / aggression

CONFERENCE CRITIQUE

Venue: ACM Computer and Communications Security 2018 (A* CORE ranking)

- + from real world data (Metasploit, NVD)
- + RL could be very impactful on attack design
- + easy to understand for non-expert RL
- + good insight on framework design
- + comparison shows advantages of RL agent

- simplistic pentest model
- incl. no social engineering
- model hard to scale to large networks
- no in-depth analysis of RL
- no actionable security advice

Bottom line: better suited for a RL rather than a security venue



ENVIRONMENT — LONGER TIMESPANS

- Add missing *ATT&CK* actions:
 - collection: key-logger, webcam, etc
 - command & control: periodically communicate externally (in/out)
- Can measure actions impact in the long run
- Some modern attacks focus on of stealth, long-term infiltrations

ENVIRONMENT — LARGER NETWORKS

- Leverage the clustering nature of enterprise networks
- Generalize same approach to a larger environment
- Employ abstraction layers:
 - 1. choose a connected component
 - 2. choose a machine
 - 3. choose an action

DEFENDER — MORE SOPHISTICATED

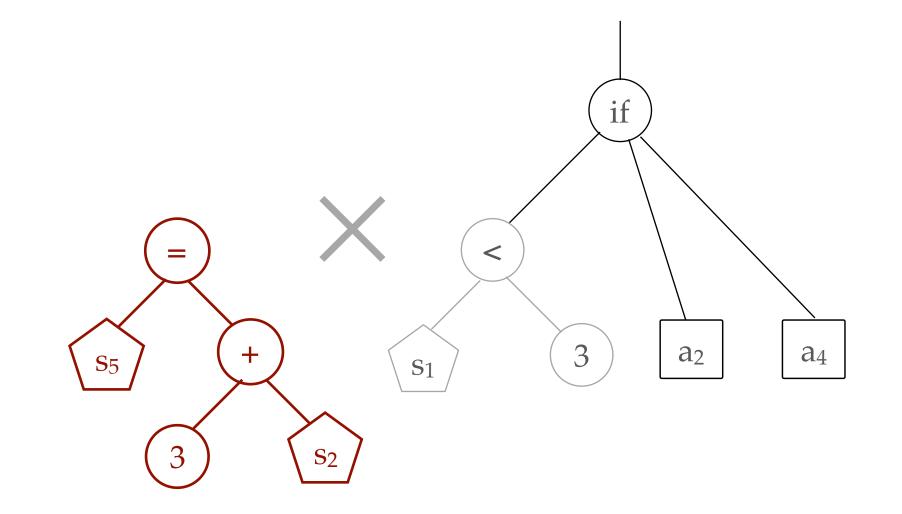
- Stackelberg Equilibrium
 - from Game Theory, Economics
 - protect those places that are valuable & often targeted
 - real-life eg: allocate policemen to airport gates
- Other defense strategies
- Measure agents performance against multiple levels of defender strength

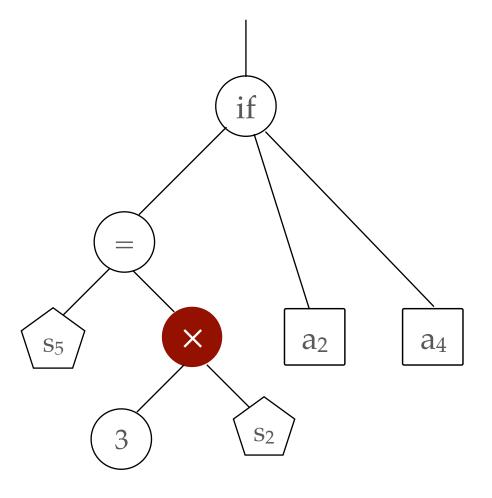
ATTACKER — PRESS FURTHER

- Finer DQN hyper-parameter tuning
- Other DQN flavors
- Let them run for longer

ATTACKER — ADVANCED GA METHOD

- Evolve data structures which are executable programs
- Generalized Decision Trees
- Nodes operations:
 - ▶ branching: *if-then*
 - ▶ arithmetic: +, -, ×, /
 - ▶ comparison: >, ≤, ≠
 - ▶ logical: and, or, not
- Input (root): state features
- Output (leaves): action to take





APPLY TO REAL WORLD

- Map actions to actual commands
- They are already loosely based on *Metasploit* (one of the most popular pentest tools)
- Run and benchmark against real anti-virus solutions

Thank you, QUESTIONS?

SELECTED BIBLIOGRAPHY

- Techniques fundaments
 - 1. Reinforcement Learning: An Introduction, Sutton & Barto (1998)
 - 2. <u>Handbook of Evolutionary Computation</u>, Fogel & Bäck (1997)
 - 3. Towards a Science of Security Games, USC (2016)
 - 4. Rainbow: Combining Improvements in Deep Reinforcement Learning, DeepMind (2017)
- Problem formalization
 - 5. <u>Adversarial Reinforcement Learning in a Cyber Security Simulation</u>, U Groningen (2017)
 - 6. Analysis of Automated Adversary Emulation Techniques, MITRE (2017)
 - 7. POMDPs Make Better Hackers: Accounting for Uncertainty in Penetration Testing, Core (2012)