# Applications of reinforcement learning to computer security: problems, models, and perspectives

Fabio Massimo Zennaro
fm.zennaro@gmail.com

OsloMET
October 22th, 2021

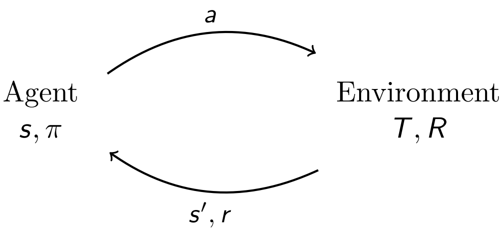
# Outline

1. *Introduction*: reinforcement learning and security.
2. *Defining the problem*: modeling security problems with reinforcement learning.
3. *Approaches to the problem*: preliminary work done.

# 1. Introduction

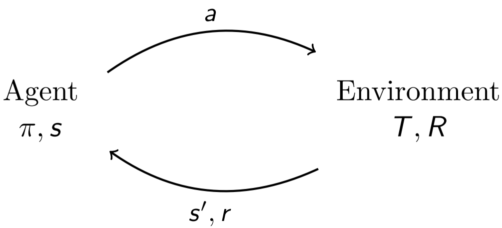
# Reinforcement learning (RL)

*Reinforcement learning* is a framework to train *agents* in a dynamic environment.




1. In *state* $s$, an agent takes *action* $a$ according to *policy* $\pi$
2. The environment returns:
   - a new *state* $s'$ (or an observation theoreof) according to the *transition function* $T$;
   - a *reward* $r$ according to the *reward function* $R$.

# Reinforcement learning (RL)




3. The agent updates its *policy* $\pi$ according to the result with the objective of *maximizing its return* (long-term discounted sum of rewards).

# RL Problem

RL had impressive success on *games*...

*Go*

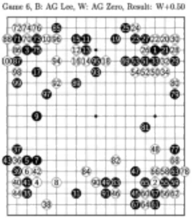



Image from [5]

*Starcraft II*


Image from [7]

...and *gamified problems* (such as economic policies [8] or spin Hamiltonians [2]).

# Security

Can we gamify *computer security* problems?

- What problems can we consider?
- How do we formalize them?
- What would we get?

# Capture the Flag (CTF)

We focus on **capture the flag** games, an artificial/educational version of *penetration testing*.

- Game environment given by a vulnerable system
- Flag as an objective
- Time limit
- Action restriction

# 2. Defining the Problem

# Mapping CTF to RL

$$\text{Ethical Hacker } (\pi, s) \;\; \underset{s', r}{\overset{a}{\rightleftarrows}} \;\; \text{Vulnerable machine } (T, R)$$

| | |
|---|---|
| *Actions* $a$ | strings, packets, commands |
| *State* $s$ | response of the machine |
| *Policy* $\pi$ | strategy of the ethical hacker |
| *Transition* $T$ | logic of the machine |
| *Reward* $r$ | signal for capture of the flag |

# Mapping CTF to RL

$$\begin{tikzcd}[column sep=huge] \text{Ethical Hacker} \atop \pi, s \arrow[r, bend left, "a"] & \text{Vulnerable machine} \atop T, R \arrow[l, bend left, "s', r"] \end{tikzcd}$$

*Where does our gamification hold?* Where does the analogy hold compared to games such as Go wrt:

- *Objectives*
- *Action space*
- *Game knowledge*
- *Structure disclosure*

# Objectives

**Go**

- Clear defined objective (*winning the game*)
- Uniform objective across games (*maximize territory*)
- Natural possible sub-objectives (*maximize territory*)

**CTF**

- Artificial defined objective (*capturing the flag*)
- Different objectives across games (*exploit different vulnerabilities*)
- Complex sub-objectives (*disclosing information*)

∴ *We need a class of CTF games with defined objective (and possibly sub-objectives).*

# Action Space

**Go**

- Finite defined set of moves (*placing a stone*)
- Set of homogeneous and same-level actions (*stone-level action*)

**CTF**

- Undefined set of actions (*what are all the actions of an ethical hacker?*)
- Set of inhomogeneous and different-level actions (*what constitutes an action for an ethical hacker?*)

∴ *We need to define a consistent set of actions at the right level of abstraction.*

# Game Knowledge

### Go

- All needed knowledge is localized
  (*board state*, *rules*)
- All needed knowledge is encodable
  (*board state*, *rules*)

### CTF

- Needed knowledge is varied and distributed
  (*network protocols*, *operating systems*, *software vulnerabilities*)
- Encoding all this knowledge seems unfeasible
  (*how do we express it?*)

$\therefore$ *We need to restrict to a game requiring a limited encodable knowledge.*

# Structure Disclosure

**Go**

- State of the game is complex and evolving (*positions across the board*)
- State of the game is perfectly known to the player (*open board*)
- Complex structure of the game allows for learning complex policies.

**CTF**

- State of the game may be trivial (*running/violated*)
- State of the game may be perfectly hidden (*empty responses until capture of the flag*)
- Lack structure of the game risks reducing the problem to guessing.

∴ *We need a game with enough structure to allow learning.*

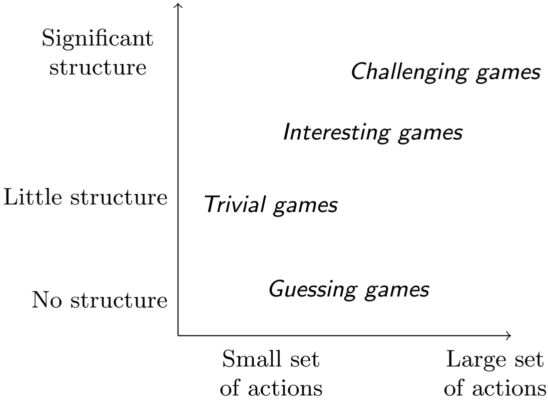
# Need for formalization

Not all these challenges are unique to CTF.

The central challenge is **define a formal RL problem with**:

1. **a manageable set of actions**
2. **enough structure to allow for learning**

# Actions vs Structure




- RL excels in the upper-right corner
- Real-world security problems may well live in the bottom-right corner

# 3. Approaches to the Problem

# Preliminary works

Joint work with:

**NTNU**
László Erdődi

**UiO**
Fabio Massimo Zennaro
Åvald Åslaugson Sommervoll
Robert Chetwyn

**MILA**
Manuel Del Verme
Simone Totaro

- All papers available on *ArXiv*[1]
- All code available on *github*[2]
- Most environments are available as *OpenAI gym* environments[3]

[1] https://arxiv.org/
[2] https://github.com/
[3] https://gym.openai.com/

# Modeling Penetration Testing with RL

An empirical work addressing the questions[4]:

- Can we practically model CTF challenges?
- Can we solve these challenges?
- How can we make the problem solvable?

[4]Fabio Massimo Zennaro, and László Erdodi. "Modeling Penetration Testing with Reinforcement Learning Using Capture-the-Flag Challenges: Trade-offs between Model-free Learning and A Priori Knowledge." arXiv preprint arXiv:2005.12632 (2021)

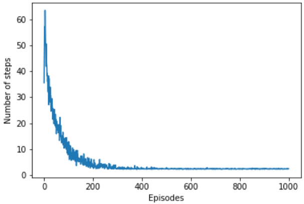
# Modeling Penetration Testing with RL

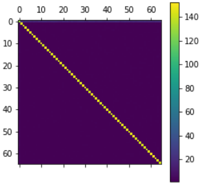
A *trivial game* of port scanning.

*Actions:* scanning, exploiting.

*Structure:* state of the ports.

A simple *tabular Q-learning* agent [6] can learn optimal policy in stationary and non stationary environments.



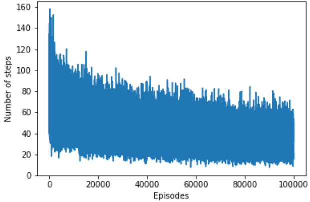
# Modeling Penetration Testing with RL

An *interesting game* of server exploitation.

*Actions:* probing, interacting, exploiting.

*Structure:* state of the ports and services.

We already have more than $2 \cdot 10^6$ states. A simple tabular Q-learning requires *lazy loading* to learn some of the structure in this game.



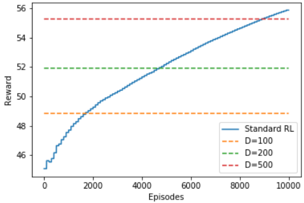
# Modeling Penetration Testing with RL

An more *interesting game* of website exploitation.

*Actions:* discovering, interacting, exploiting.

*Structure:* state and relationships among files.

A simple tabular Q-learning requires *state aggregation* [6] and/or *imitation learning* [1] to learn some of the structure in this game.



# Modeling Penetration Testing with RL

Conclusions:

- ✓ Feasibility of modelling CTF and applying RL
- ✓ Computational cost of even simple challenges
- ✓ Usefulness of introducing a priori knowledge to allow the agent to explore the action space more efficiently (lazy loading, state aggregation, imitation learning)

# The Agent Web Model

A conceptual work addressing the questions[5]:

- How can we express a ladder of abstractions for CTF games?
- Which aspects should be captured?

[5] Erdődi, László, and Fabio Massimo Zennaro. "The Agent Web Model: modeling web hacking for reinforcement learning." International Journal of Information Security (2021): 1-17.

# The Agent Web Model

**LAYER 7**

**Server modification layer –** The agent can modify the web server by adding new files through vulnerable objects or creating new database objects and data

**LAYER 6**

**Server structure layer –** The agent can observe the structure of the server, e.g. objects outside the web root, database objects with their structure

**LAYER 5**

**Http header layer –** The agent can use the http header values such as the session pairs to obtain restricted content of the objects and obtain the response header with the response code

**LAYER 4**

**Web method layer –** The agent can use the GET and POST methods to send parameters in associative arrays

**LAYER 3**

**Dynamic content layer –** The agent can observe multiple versions of an object using parameters for the request

**LAYER 2**

**Hidden link layer –** The agent can use the text of the objects to find hidden references to other files such as default objects

**LAYER 1**

**Link layer –** The agent can find the links inside the objects and map the linked structure of the site

# The Agent Web Model

Conclusions:

- ✓ A framework to abstract CTF challenges
- ✓ A roadmap of challenges for RL
- ✓ Some baselines for tasks on the lower ladders

# Simulating SQL Injection Vulnerability

An experimental work addressing the questions[6]:

- Can we model an actual real-world problem?
- Which RL models can we use to solve the problem?

[6] László Erdődi, Åvald Åslaugson Sommervoll, Fabio Massimo Zennaro. "Simulating SQL Injection Vulnerability Exploitation Using Q-Learning Reinforcement Learning Agents." Journal of Information Security and Applications (2021): 61, 102903.

# Simulating SQL Injection Vulnerability

An *interesting game* of SQL injection.

A website receives a parameter *s* and embeds it in a query with a form like:

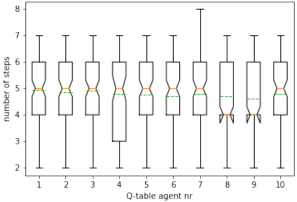
$$\texttt{SELECT Column3}, \texttt{Column4 FROM Table2 WHERE Column1} =' \texttt{s}'$$

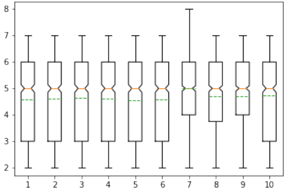
*Actions:* a restricted pre-defined set of SQL statements.

*Structure:* finding right escape character, finding right exploit.

# Simulating SQL Injection Vulnerability

Both a *tabular Q-learning* agent and a *DQN* agent [3] manage to solve the problem.



# Simulating SQL Injection Vulnerability

Conclusions:

- ✓ Successfully learning to perform SQL injection.
- ✓ Need for a high level of prior knowledge in the form of pre-defined actions.
- ✓ Use of flexible neural network-based models.

# SQL Injections and Reinforcement Learning

An experimental work addressing the questions[7]:

- Can we reduce the a priori information that the agent requires?
- Can we make the action space of the SQL more natural?
- Which RL models can we use to solve the problem?

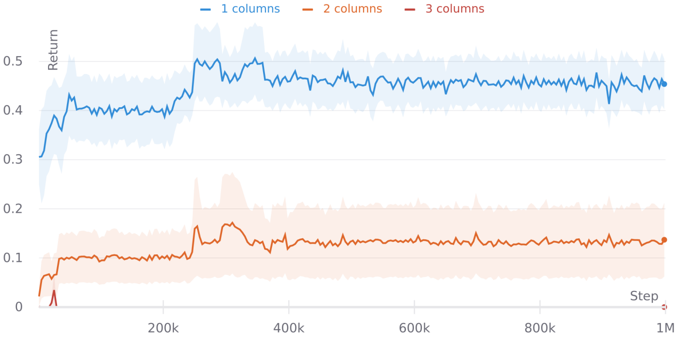
[7] Manuel Del Verme, Åvald Åslaugson Sommervoll, Laszlo Erdodi, Simone Totaro, and Fabio Massimo Zennaro. "SQL Injections and Reinforcement Learning: An Empirical Evaluation of the Role of Action Structure."

# SQL Injections and Reinforcement Learning

Same problem of SQL injection.

*Actions:* composition of a SQL statement from basic tokens and strings.

A *PPO* agent [4] struggles and learn under simplified conditions.



# Simulating SQL Injection Vulnerability

Conclusions:

- ✓ Reducing a priori knowledge increases the computational cost.
- ✓ An agent with less a priori knowledge may discover unexpected solution.
- ✓ We look for a trade-off between a priori knowledge and free exploration.

# Current directions of research

- Can we generate automatically interesting and diverse environments and produce data for learning?
- Can we use collected human data to train an agent?
- Can we integrate natural language processing modules?
- Can we learn to generate commands?
- How much a priori knowledge can we drop?

# Thanks!

Thank you for listening!

# References I


[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[2] Kyle Mills, Pooya Ronagh, and Isaac Tamblyn. Finding the ground state of spin hamiltonians with reinforcement learning. *Nature Machine Intelligence*, 2(9):509–517, 2020.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

# References II


[5] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017.

[6] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[7] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[8] Stephan Zheng, Alexander Trott, Sunil Srinivasa, David C Parkes, and Richard Socher. The ai economist: Optimal economic policy design via two-level deep reinforcement learning. *arXiv preprint arXiv:2108.02755*, 2021.