Understanding Malware Behavior with Reinforcement Learning

Ezekiel Ajayi, Mike Anoruo, Nomso A., Aritran Piplai
University of Maryland, Baltimore County

OnRamp II Symposium
13-14 October 2021

Acknowledgements

- Dr. Ahmad Ridley, NSA Contact
- Dr. Anupam Joshi, Professor
- Dr. Tim Finin, Professor
- Priyanka Ranade, Graduate Student
- Aritran Piplai, Graduate Student

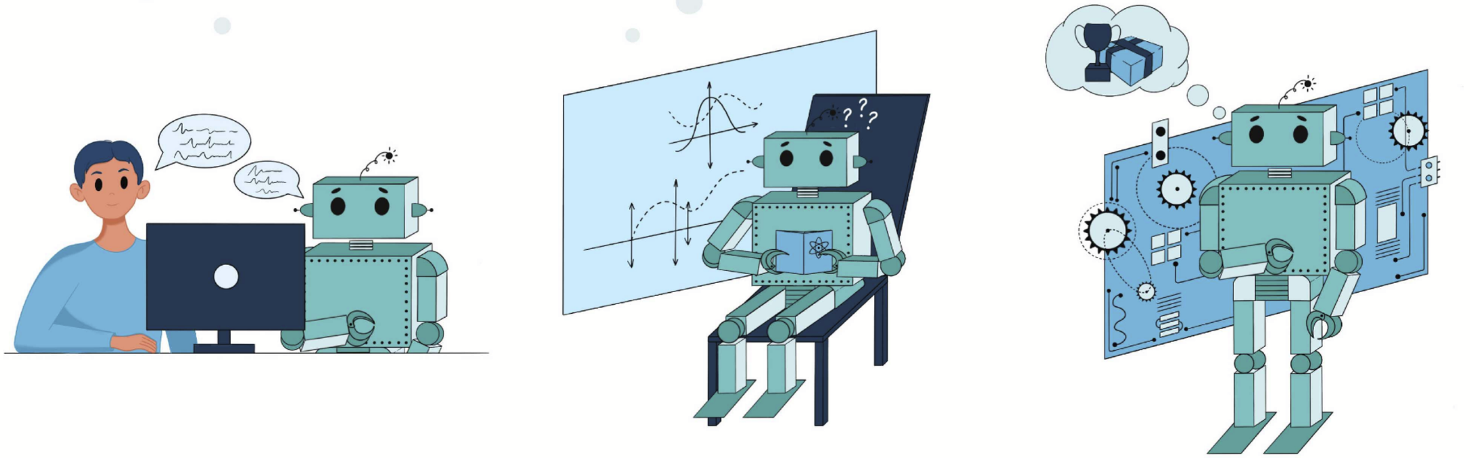
Thank you to the NSA for their support!

Overview

- **Reinforcement Learning Fundamentals**
- Application of RL for malware evasion
- Prior knowledge collection for malware
- Future work: using prior knowledge in RL

Reinforcement Learning Fundamentals

Types of Machine Learning



Supervised Learning | **Unsupervised Learning** | **Reinforcement Learning**

Image from - https://blog.autify.com/en/machine-learning-in-software-testing

How is RL different?

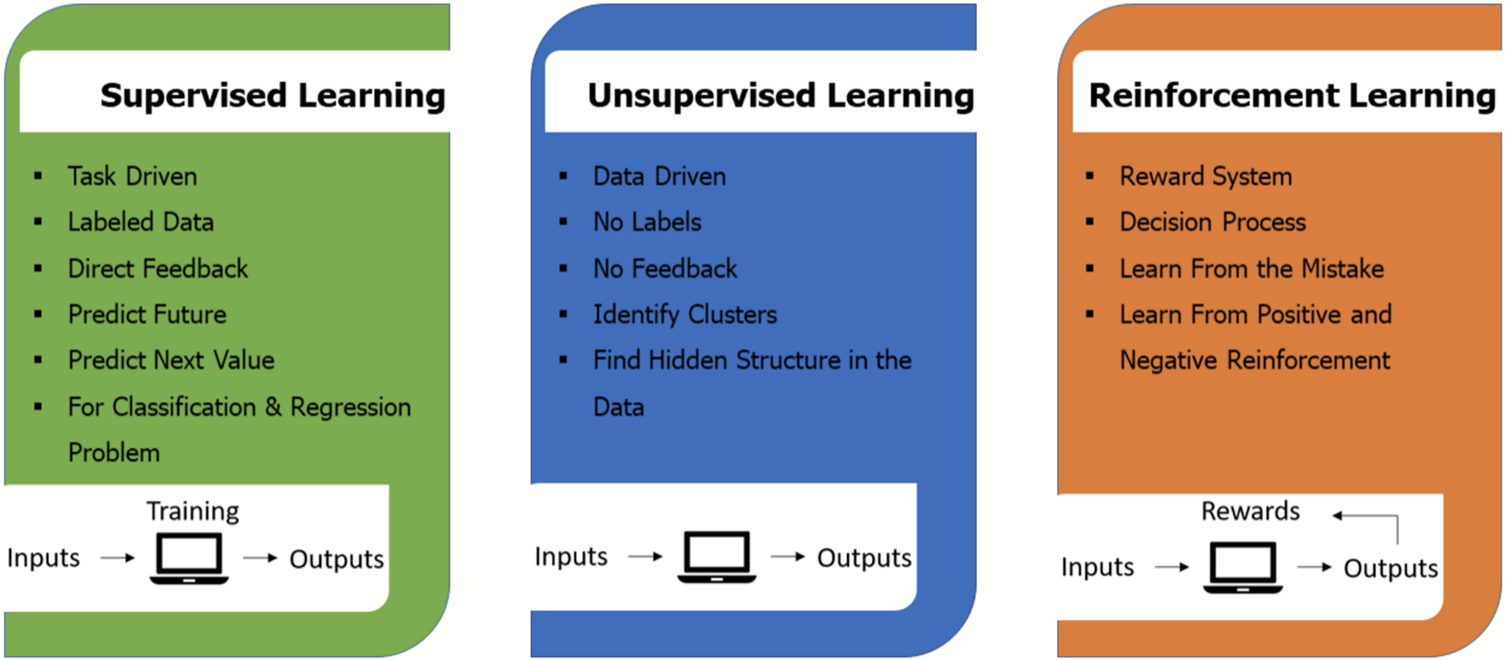
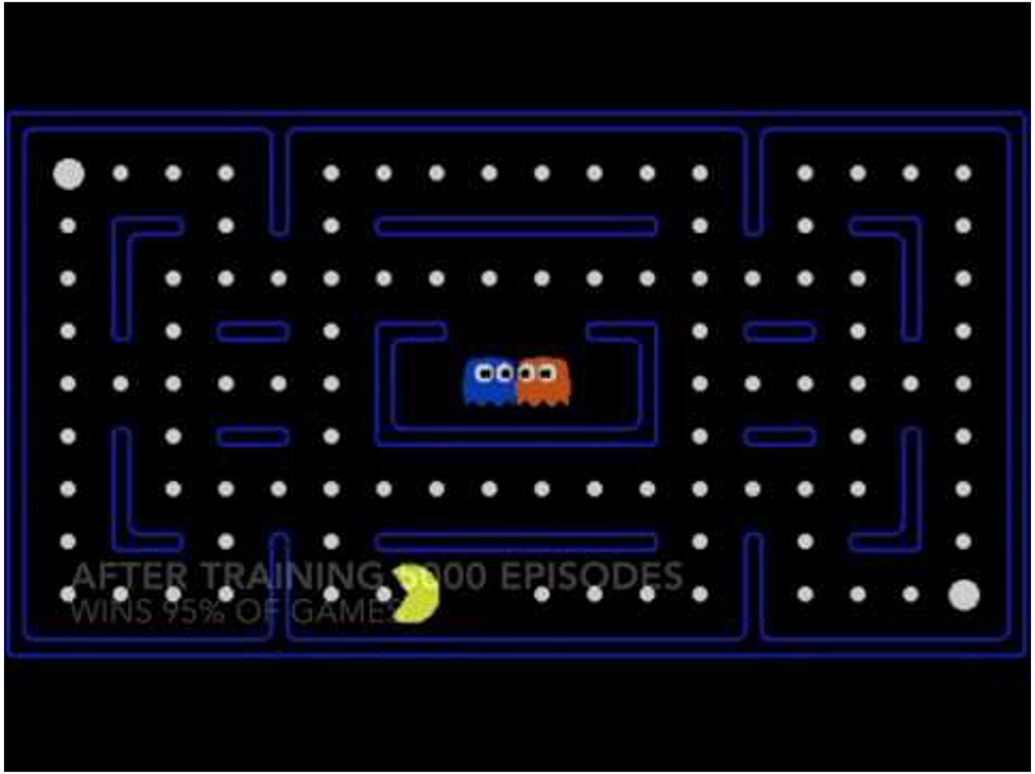
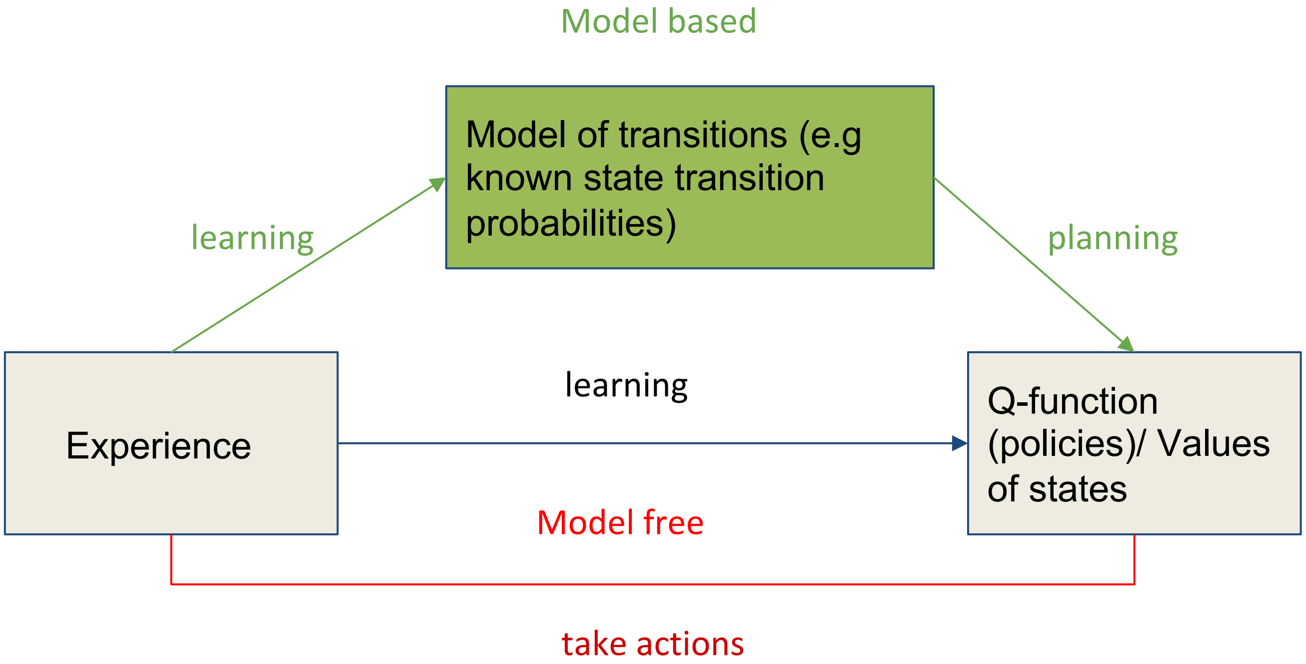


Image from - https://harshgajjar8996.blogspot.com/2020/10/types-of-machine-learning.html

Reinforcement Learning in Action



Model-free vs Model-based



Elements of RL (MDP)

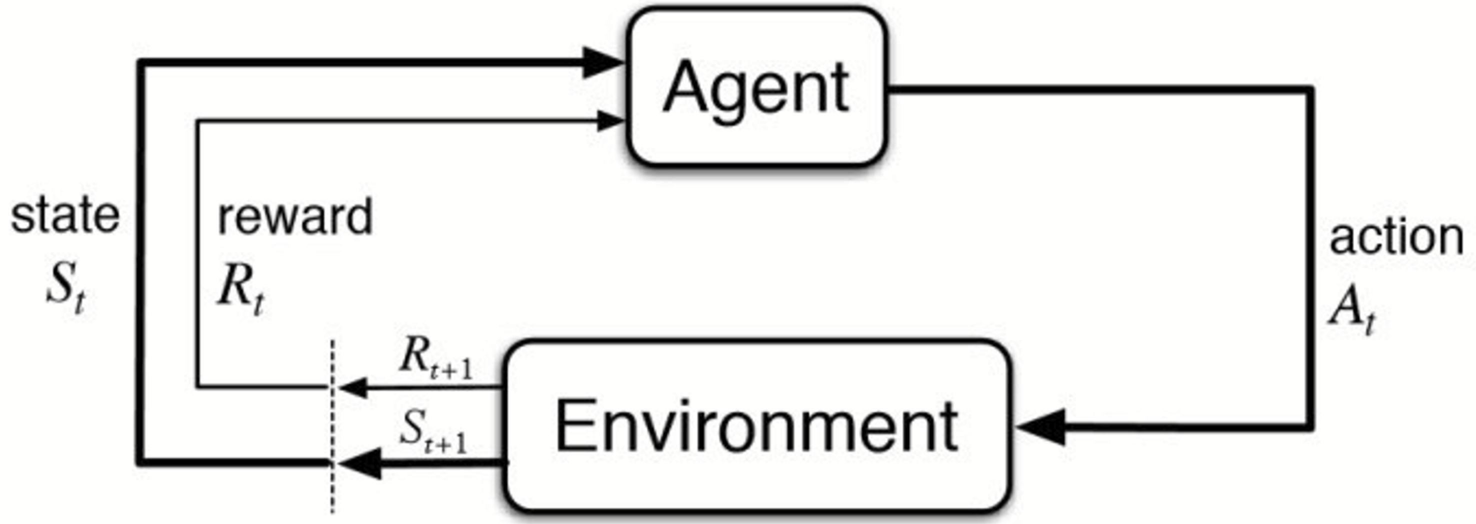


Image from - https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html

Maximizing Reward w/ Q-Learning

- We consider Q-learning for MDP optimality
- Q-learning involves calculating the optimal action-value (***q***)

$$\boxed{q_*}(s, a) = \max_\pi \boxed{q_\pi}(s, a)$$

Q-value of best policy

Best Possible Policy

Bellman Optimality Equation for RL

- When considering Q-Learning, the agent must satisfy the following equation

$$q_*\boxed{(s,a)} = E\left[\boxed{R_{t+1}} + \boxed{\gamma \max_{a'} q_*(s',a')}\right]$$

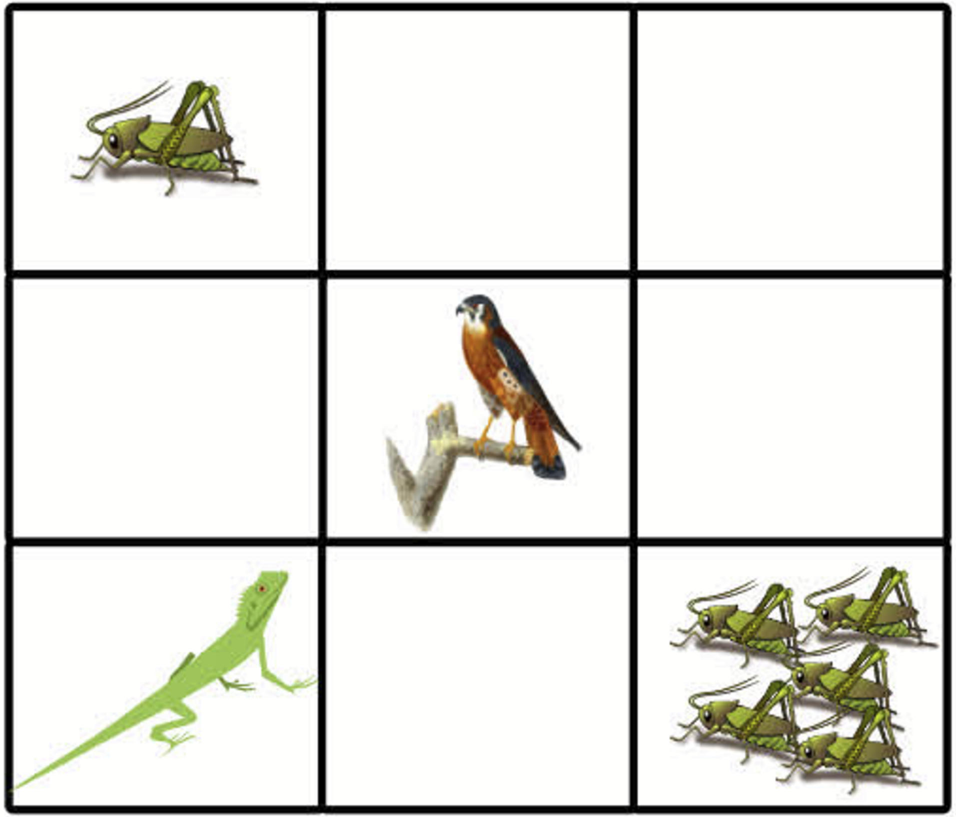
State-action pair

Expected Reward

Maximum expected discounted return

Example - Lizard Game

- The Lizard is allowed to move: up, down, left, or right
- Each tile represents a state
- Types of tiles:
  - 1 Cricket (+1 pt)
  - 5 Crickets (+10 pts / Win)
  - Empty (-1 pt)
  - Bird (-10 pts / Game Over)
- At the start, the lizard knows nothing



Example - Lizard Game (cont.)

- Q-table initializes to zero values
- Over time, the agent (lizard) will play multiple episodes
  - Q-values for each state-action pair will change
  - Learn from previous episode Q-tables
  - Calculate the highest Q-value for a current state

- γ = 0.99 (Discount Factor)
- q_table[empty 4, right]=   -1 + 0.99 *max(0,0,0,0)

- q_table[empty 4, right] = -1

$$q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right]$$

		Actions			
		Left	Right	Up	Down
States	1 Cricket	0	0	0	0
	Empty 1	0	0	0	0
	Empty 2	0	0	0	0
	Empty 3	0	0	0	0
	Bird	0	0	0	0
	Empty 4	0	0	0	0
	Empty 5	0	0	0	0
	Empty 6	0	0	0	0
	5 Crickets	0	0	0	0

Overview

- Reinforcement Learning Fundamentals
- **Application of RL for malware evasion**
- Prior knowledge collection for malware
- Future work: using prior knowledge in RL

Malware Evasion Task

The malware evasion task allowed us to put reinforcement learning into practice.

Goal: Use reinforcement learning to mutate Windows Portable Executable (PE’s)so they evade detection by malware classifiers

$$loss = \Big(\underbrace{\overset{\text{Reward}}{r} + \overset{\text{Decay Rate}}{\gamma} \max_{a`} \hat{Q}(s, a`)}_{\text{Target}} - \underbrace{Q(s, a)}_{\text{Prediction}}\Big)^2$$

Evasion Task Set Up

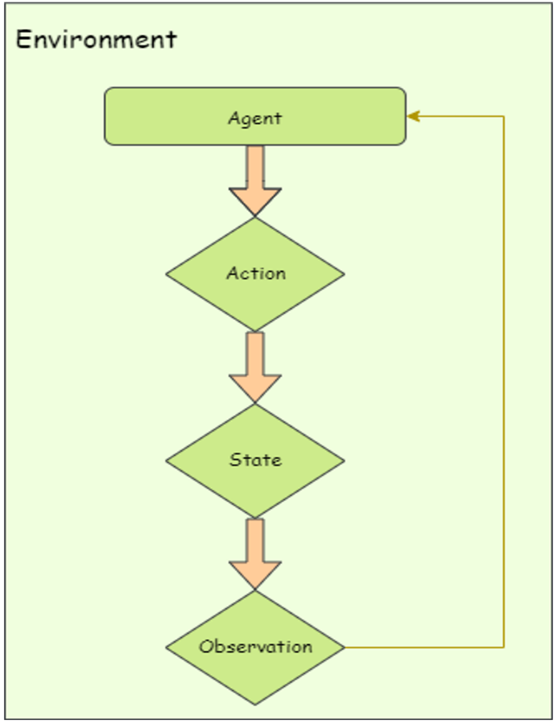
Environment: Consists of the malware classifier and the perturbations of the PE files after actions are taken

Agent: Responsible for manipulating PEs to be undetectable by classifier

Actions: All possible modifications available for PE

State: PE after it has been modified

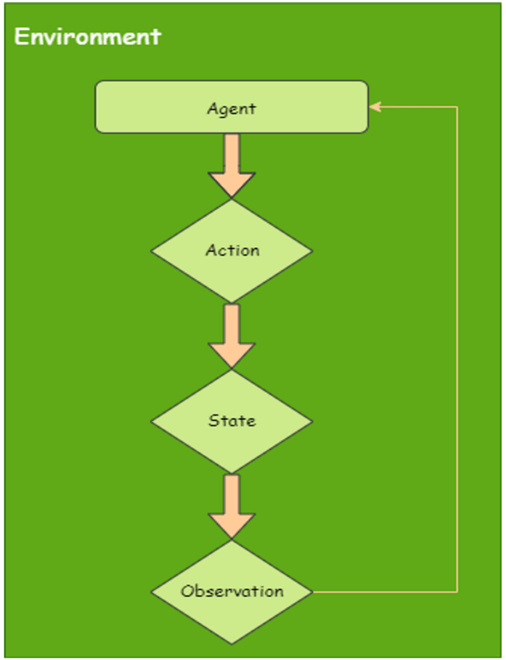
Observation: PE Detected or not



Environment

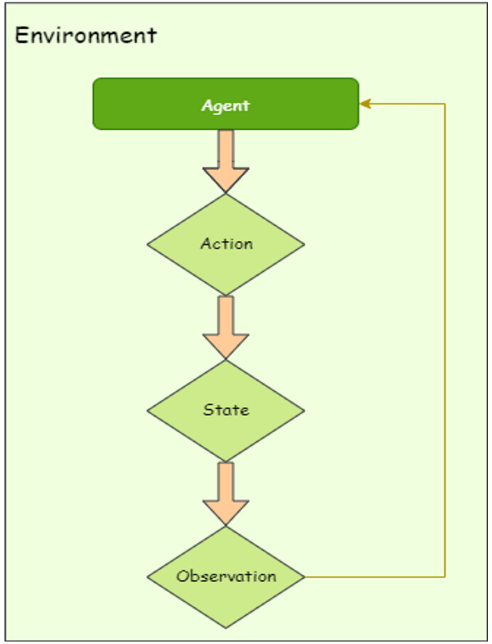
- In this task the environments explored are the Ember and MalConv Classifiers.

- These Classifiers check different properties of a executable to identify whether or not it is malicious or safe.



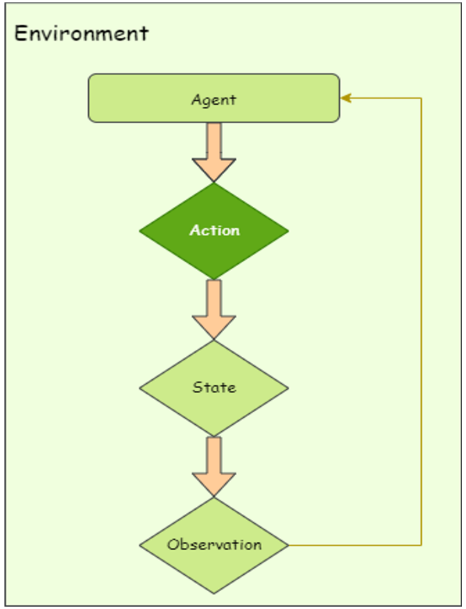
Agent

- We deploy a agent for this task.
  - The agent is given a batch of PE’s to train from.
  - The agent’s goal is to make as many PE’s undetectable by the classifier as possible
  - Overtime, the agent learns from its interaction with previous PE.



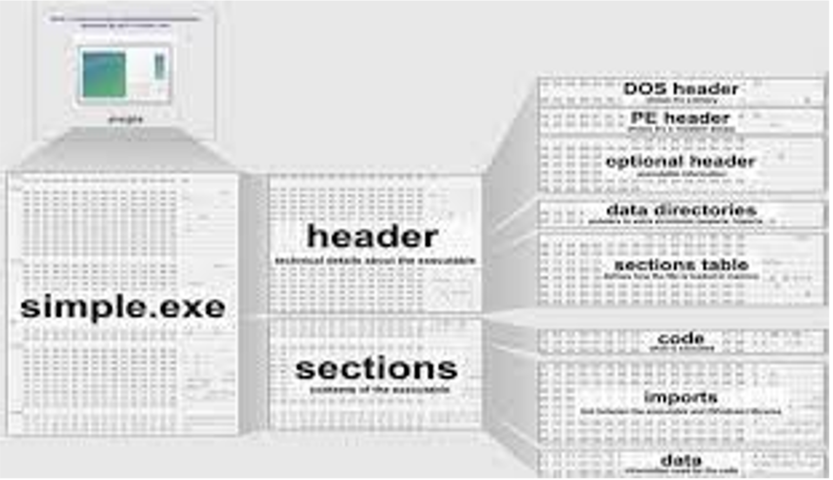
Actions

- The MalwareRL environment provides various actions that can be taken to alter a executable
- We do this using the LIEF library
- Most of the actions provided modify one of the following properties of the executable:
    - Header
    - Section
    - Imports
    - Overlay



Action Space Cont.

During the training process, our agent will choose an action from the table to perform on a executable.



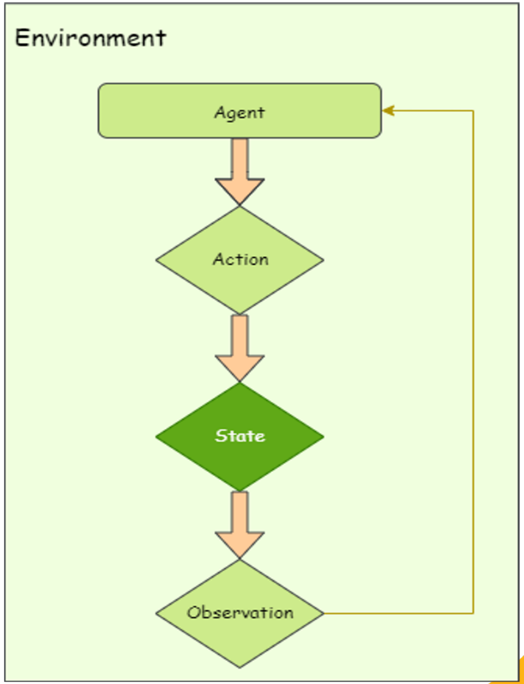
Actions
modify_machine_type
pad_overlay
append_benign_data_overlay
append_benign_binary_overlay
add_bytes_to_section_cave
add_section_strings
add_section_benign_data
add_strings_to_overlay
add_imports

State

- The state of a PE would be the new changed file

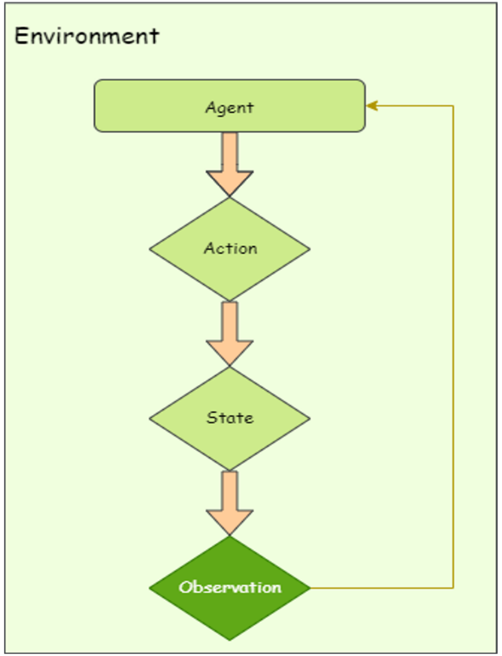
- For example, a PE that has had its header modified could be considered to be in a "modified header" state.

- States are important because they allow our agent to learn what actions are desirable.



Observation

- After modifying a PE, the agent will then pass the altered file into a classifier.
- The classifier will relay whether or not the modified malware PE was successfully undetected.
- The agent will then observe this result, and if successful record what action was last taken to obtain the desired state.
- On each PE the agent will begin to identify from previous experience the best actions to perform on a malicious PE to make it undetectable.



Observation Cont.

Agent receives reward of +10 if PE file successfully evades detector

Agent receives reward of -1 if PE file does not evade detector

Agent receives reward of -1 if PE file does not evade after a finite number of actions

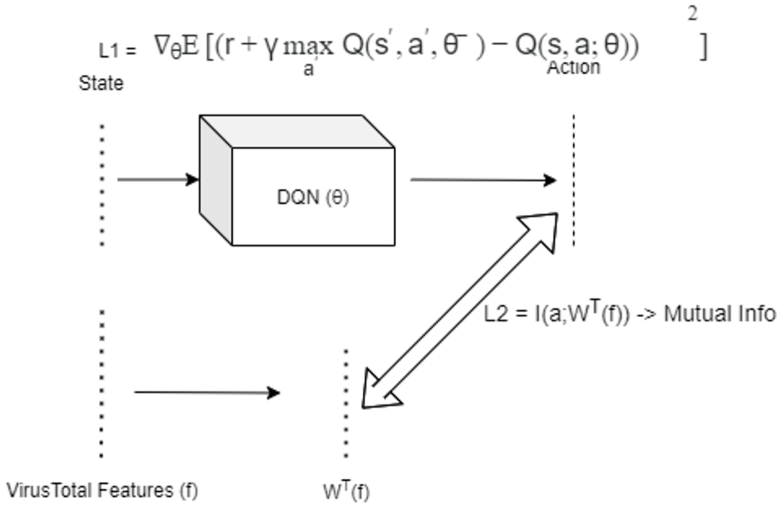
Results

- We analyzed 500 PE files
- The successful evasion rate: 89.7 %
- Average length of action sequence: 7.4

Overview

- Reinforcement Learning Fundamentals
- Application of RL for malware evasion
- **Prior knowledge collection for malware**
- Future work: using prior knowledge in RL

Prior knowledge Incorporation (WIP)



L3 = *a*.L1 + *b*.L2

- Prior knowledge can help us guide RL algorithms
- Prior knowledge can be in the form of CKGs
  - CKG can have data from unstructured CTI
  - CKG can have data from structured sources like VirusTotal

What is VirusTotal ?

- 70 antivirus scanners and URL/domain blocklisting services
- Other tools to extract signals from files
- Free and unbiased
- Has API to access data

Public Vs Premium API Key

Public api key

- 500 requests per day at a rate of 4 requests per minute

Private api key

- unlimited requests
- all functions are available
- returns more threat detection data

Functions

These functions are used to interact with the Core part of the API

-get_relationship()

-get_votes

-info_domain()

-add_vote()

-analyse-file()

`get_relationship`

`download(file_hash, output_dir='./', timeout=None)`

`get_votes(file_hash, limit=None, cursor=None, timeout=None)`

UMBC

Results

/files/{id}

Retrieve information about a file

GET https://www.virustotal.com/api/v3/files/ id

Try It

cURL

```
curl --request GET \
  --url https://www.virustotal.com/api/v3/files/{id} \
  --header 'x-apikey: <your API key>'
```

200 OK Metadata Examples

{...}

PATH PARAMS

id* string

SHA-256, SHA-1 or MD5 identifying the file

0a0f706955b59cbda99e12a345d

RESPONSE 200

200

HEADERS

x-apikey* string

Your API key


```
{
  "data" : {
    "attributes" : {
      "type_description" : "Win32 EXE"
      "tlsh" :
      "T19F94BE16349188F3CC4284B10AAAAF70BBF7DA30083419D3DBD5ED9D6E695D19B1E317"
      "vhash" :
      "045046655d15113070200 1d007b7z4015zb2z92031zc9z"
      "trid" : [...]
      "crowdsourced_yara_results" : [...]

      "creation_date" : 1253905052
      "names" : [...]
      "signature_info" : {...}
      "last_modification_date" : 1618568266
      "type_tag" : "peexe"
      "times_submitted" : 7
      "total_votes" : {
        "harmless" : 0
        "malicious" : 0
      }
      "size" : 422009
      "popular_threat_classification" : {
        "suggested_threat_label" : "pua."
        "popular_threat_category" : [
        ...]
      }
      "authentihash" :
      "7727ebadabcc22f214dc36afa13a9b6fce10033fd0d231fd605174783ff718f7"
      "last_submission_date" : 1582299459
      "meaningful_name" :
      "VirusShare_0a0f706955b59cbda99e12a345d8e8c6_x1kVJG4.EXE"
```

Examples of Functions

```
import os
import virustotal3.core

API_KEY = 

#virustotal object
livehunt = virustotal3.core.Files(API_KEY)
#info file is a function that our results came from
rulesets = livehunt.info_file('f88d7abc32debea82beaeaa2b4c6c37ef1f0ef2a8bc6142be84456afc23836cb')

print(rulesets)
```

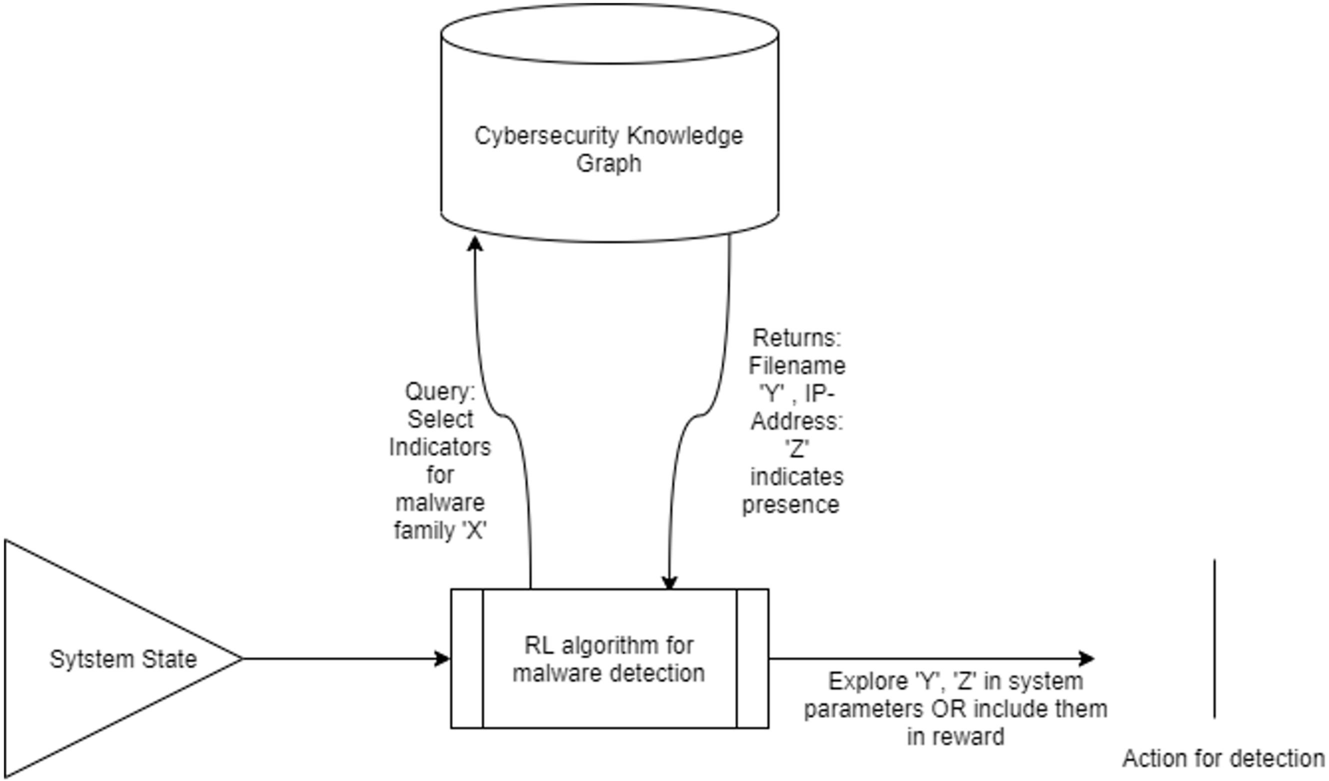
Overview

- Reinforcement Learning Fundamentals
- Application of RL for malware evasion
- Prior knowledge collection for malware
- **Future work: using prior knowledge in RL**

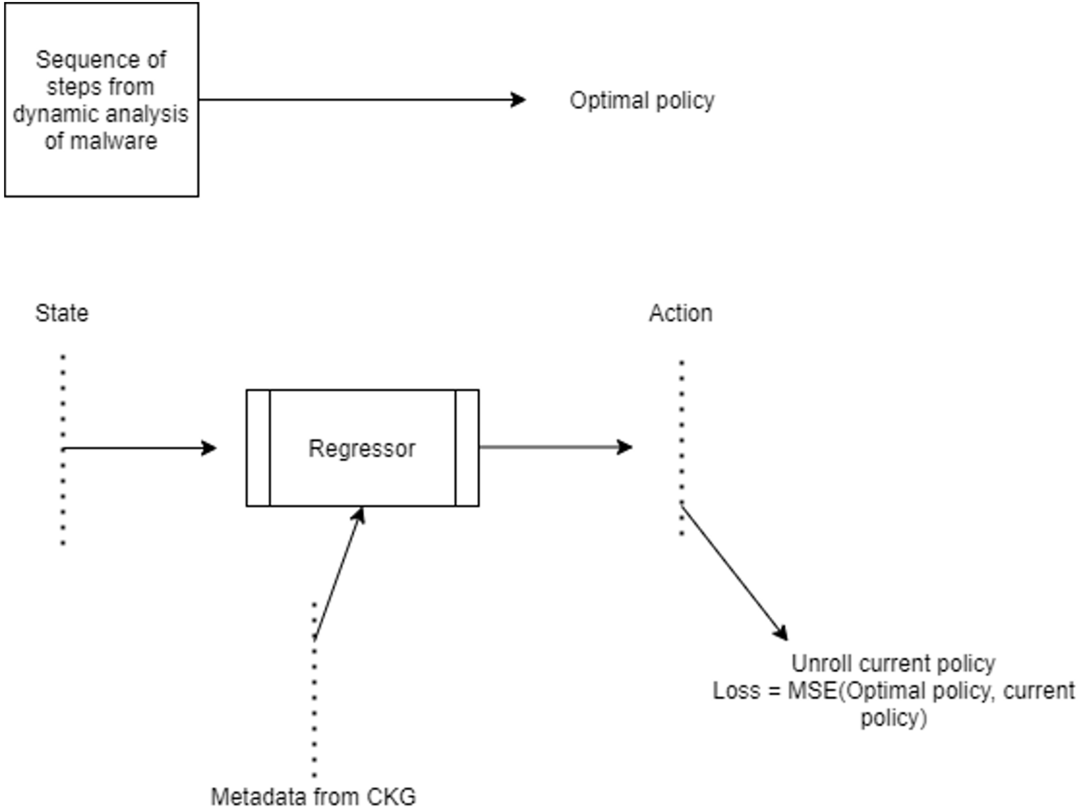
Future Work

- Apply combination of RL + CKG/VirusTotal for malware generation
- Apply combination of RL + CKG for malware detection
- Use prior knowledge from multiple sources that can help us model Red Team/ Blue Team behavior
- Retrain classifiers with generated malware samples

Use CKG parameters for RL algorithm



Use CKG parameters for modeling Red/Blue Team



Questions?

For questions please contact –
apiplai1@umbc.edu

THANK YOU!