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

This work was funded, in part, by a grant from the NSA through the On-Ramp program and by the National Science Foundation under Grant Number 2114892.
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

How is RL different?

**Supervised Learning**

- Task Driven
- Labeled Data
- Direct Feedback
- Predict Future
- Predict Next Value
- For Classification & Regression Problem

![Training](Inputs → Outputs)

**Unsupervised Learning**

- Data Driven
- No Labels
- No Feedback
- Identify Clusters
- Find Hidden Structure in the Data

![Inputs → Outputs](Inputs → Outputs)

**Reinforcement Learning**

- Reward System
- Decision Process
- Learn From the Mistake
- Learn From Positive and Negative Reinforcement

![Rewards](Inputs → Outputs)

Reinforcement Learning in Action
Model-free vs Model-based

Model of transitions (e.g. known state transition probabilities)

Model based

Experience

Planning

Learning

Q-function (policies)/ Values of states

Model free

take actions

learning
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 ($q$)

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

Q-value of best policy

Best Possible Policy
Bellman Optimality Equation for RL

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

\[ q^*(s, a) = E[R_{t+1} + \gamma \max_{a'} q^*(s', a')] \]

- 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
- $\gamma = 0.99$ (Discount Factor)
- $q_{\_table[empty\ 4, \ right]}= -1 + 0.99 \times \max(0,0,0,0)$
- $q_{\_table[empty\ 4, \ right]} = -1$

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

<table>
<thead>
<tr>
<th>States</th>
<th>Left</th>
<th>Right</th>
<th>Up</th>
<th>Down</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Cricket</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Empty 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Empty 2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Empty 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Bird</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Empty 4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Empty 5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Empty 6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5 Crickets</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
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
\[ \text{loss} = \left( r + \gamma \max_{a'} \hat{Q}(s, a') - Q(s, a) \right)^2 \]

- **Reward**
- **Decay Rate**

**Target**

**Prediction**

---

**Diagram:**
- **Malware PE File**
- **Agent**
- **Action**
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 an 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
During the training process, our agent will choose an action from the table to perform on a executable.

<table>
<thead>
<tr>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>modify_machine_type</td>
</tr>
<tr>
<td>pad_overlay</td>
</tr>
<tr>
<td>append_benign_data_overlay</td>
</tr>
<tr>
<td>append_benign_binary_overlay</td>
</tr>
<tr>
<td>add_bytes_to_section_cave</td>
</tr>
<tr>
<td>add_section_strings</td>
</tr>
<tr>
<td>add_section_benign_data</td>
</tr>
<tr>
<td>add_strings_to_overlay</td>
</tr>
<tr>
<td>add_imports</td>
</tr>
</tbody>
</table>
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)

- 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()`
Examples of Functions

```python
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!