PSP-Mal: Evading Malware Detection via Prioritized Experience-based Reinforcement Learning with Shapley Prior
AI-powered Malware Detection

many ML-based methods have been proposed to determine the maliciousness of software.

Adversarial attack

The reliability of malware detections receives challenges from adversarial examples, where modified malware samples can avoid detection by imposing subtle adversarial perturbations.
Evasive malware

Problem Space

- Images
  - Parmen: 85.7%
  - Lionfish: 90.8%

- Executables
  - Malware: 94.3%
  - Goodware: 5.7%

Evasive feature

Feature Space

- Feature Mapping: $\phi$
- Feature-space Attack

Inverse Feature Mapping: $\phi^{-1}$

(Ling et al. 2023)
Reinforcement learning
<table>
<thead>
<tr>
<th>Method</th>
<th>Year</th>
<th>Target Model</th>
<th>Architecture</th>
<th>Action Space</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>gym-malware [2]</td>
<td>2018</td>
<td>GBDT</td>
<td>ACER</td>
<td>10 actions</td>
<td>$r=10$ or $0$</td>
</tr>
<tr>
<td>gym-plus [50]</td>
<td>2018</td>
<td>LightGBM</td>
<td>DoubleDQN</td>
<td>16 actions</td>
<td>-</td>
</tr>
<tr>
<td>DQEAf [17]</td>
<td>2019</td>
<td>GBDT</td>
<td>DQN Sarsa</td>
<td>4 actions</td>
<td>$r = 20^{-(t-1)/t_m} * 100 + 0$ or $0$</td>
</tr>
<tr>
<td>gym-malware-mini [8]</td>
<td>2020</td>
<td>GBDT</td>
<td>DoubleDQN</td>
<td>10 actions</td>
<td>$r=10$ or $-1$</td>
</tr>
<tr>
<td>RLAttackNet [15]</td>
<td>2020</td>
<td>DeepDetectNet</td>
<td>DQN</td>
<td>6 categories</td>
<td>$r = k * t_m / t$ or $0$</td>
</tr>
<tr>
<td>MAB-Malware [44]</td>
<td>2020</td>
<td>LightGBM</td>
<td>Multi-armed Bandit</td>
<td>8 macro actions</td>
<td>$r \sim$ Bernoulli($\theta$)</td>
</tr>
<tr>
<td>A3CMal [16]</td>
<td>2021</td>
<td>Winner/Novel</td>
<td>A3C</td>
<td>6 actions</td>
<td>$r = k - k * (t-1)/t_m$ or $0$</td>
</tr>
<tr>
<td>AME-VAC [14]</td>
<td>2021</td>
<td>LightGBM</td>
<td>VAC</td>
<td>10 actions</td>
<td>-</td>
</tr>
<tr>
<td>AIMED-RL [30]</td>
<td>2021</td>
<td>LightGBM</td>
<td>DiDDQN</td>
<td>10 actions</td>
<td>$r = r_{det} + r_{sim} + r_{dis}$</td>
</tr>
<tr>
<td>AMG-IRL [31]</td>
<td>2021</td>
<td>CNN</td>
<td>IRL</td>
<td>4 actions</td>
<td>automatically generate</td>
</tr>
<tr>
<td>Gilbert et al. [19]</td>
<td>2022</td>
<td>CNN Classifier</td>
<td>DDQN</td>
<td>insert NOP instruction</td>
<td>$r = -1 * (loss_t_{t-1} - loss_t)$</td>
</tr>
<tr>
<td>MERLIN [37]</td>
<td>2022</td>
<td>LightGBM</td>
<td>DQN</td>
<td>15 actions</td>
<td>$r = R$ or $p_t - p_{t-1}$</td>
</tr>
<tr>
<td>MallInfo [56]</td>
<td>2022</td>
<td>Virustotal</td>
<td>Dynamic Programming</td>
<td>Obfusmal Stealing</td>
<td>$r(a/s) = R_E(a/s)$</td>
</tr>
<tr>
<td>SRL [55]</td>
<td>2022</td>
<td>CFG-based Classifier</td>
<td>DQN</td>
<td>inject NOPs(28) into CFG blocks</td>
<td>$r = -1 * (loss_t_{t-1} - loss_t)$</td>
</tr>
</tbody>
</table>
It is a complex task for the agent to modify the malware to achieve evasion and obtain the final reward, and it requires a large amount of useful information to guide the agent’s training. However, there is a lack of information available in the black-box scenarios.

• The action space designed by existing methods contains excessive randomness, making it difficult for the agent to accurately predict the effects of the actions.
The guidance information obtained from the SHAP approach is used as the Shapley prior. By weighing transition utilized probability based on prior knowledge, the prioritization replay mechanism can elevate the efficiency of the experience utilization.

**Shapley prior-based PER**

Thompson sampling within actions

Thompson sampling is used to address the drawbacks caused by randomness within actions.
Adversary’s goal
The adversary aims at modifying the malware sample to evade the static Windows PE malware detector, i.e., causing the modified sample to be labeled as benign.

Adversary’s knowledge
we consider the black-box setting where the adversary does not access the internals of the target detector, can only perform a limited number of attempts and receive the prediction confidence.

Dataset
EMBER: nine groups of features extracted from 1.1 million PE files

SOREL-20M: a large-scale dataset consisting of nearly 20 million files collected from 2017 to 2019

Model
LightGBM

<table>
<thead>
<tr>
<th>Features</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1: Byte Histogram</td>
<td>Byte histogram over the entire binary file.</td>
</tr>
<tr>
<td>F2: Byte Entropy Histogram</td>
<td>The joint probability of byte value and local entropy.</td>
</tr>
<tr>
<td>F3: String Information</td>
<td>Printable characters about strings.</td>
</tr>
<tr>
<td>F4: General File Information</td>
<td>Basic information obtained from the PE header</td>
</tr>
<tr>
<td>F5: Header File Information</td>
<td>Information extracted from header (Machine, linker, OS, etc.)</td>
</tr>
<tr>
<td>F6: Section Information</td>
<td>Information of each sections (names, sizes, entropy, etc.)</td>
</tr>
<tr>
<td>F7: Imports Information</td>
<td>Information about imported libraries and functions.</td>
</tr>
<tr>
<td>F8: Exports Information</td>
<td>Information about exported functions.</td>
</tr>
<tr>
<td>F9: Data Directories</td>
<td>Extracts size and virtual address of the first 15 data directories.</td>
</tr>
</tbody>
</table>
State Space

The same feature representation as the malware detector is used as the state space of the environment, namely a 2381-dimensional feature vector.

Action Space

<table>
<thead>
<tr>
<th>Location</th>
<th>Abbr</th>
<th>Name</th>
<th>Type</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>MM</td>
<td>Modify Machine Type</td>
<td>M</td>
<td>4</td>
<td>Modify the machine type to one of candidates.</td>
</tr>
<tr>
<td>H</td>
<td>MT</td>
<td>Modify Timestamp</td>
<td>M</td>
<td>5</td>
<td>Modify the timestamp to one of candidates.</td>
</tr>
<tr>
<td>H</td>
<td>MO</td>
<td>Modify Option header</td>
<td>M</td>
<td>6</td>
<td>Modify the linker/image/operating system version.</td>
</tr>
<tr>
<td>H</td>
<td>RD</td>
<td>Remove Debug</td>
<td>M</td>
<td>1</td>
<td>Zero out the debug information in a binary.</td>
</tr>
<tr>
<td>H</td>
<td>BC</td>
<td>Break Checksum</td>
<td>M</td>
<td>1</td>
<td>Zero out the checksum value in the optional header.</td>
</tr>
<tr>
<td>H</td>
<td>IA</td>
<td>Add Imports</td>
<td>A</td>
<td>20</td>
<td>Add import functions from one of candidates.</td>
</tr>
<tr>
<td>S</td>
<td>MS</td>
<td>Modify Section Name</td>
<td>M</td>
<td>10</td>
<td>Modify the section name to a name of candidates.</td>
</tr>
<tr>
<td>S</td>
<td>CA</td>
<td>Section Cave Append</td>
<td>A</td>
<td>50</td>
<td>Append bytes to the unused space at the end of a section.</td>
</tr>
<tr>
<td>S</td>
<td>SA</td>
<td>Section Add</td>
<td>A</td>
<td>50</td>
<td>Add a new section.</td>
</tr>
<tr>
<td>E</td>
<td>OA</td>
<td>Overlay Append</td>
<td>A</td>
<td>50</td>
<td>Appends bytes at the end of a binary.</td>
</tr>
</tbody>
</table>

We design a novel malware modifier in PSP-Mal, where actions are considered a combination of item and content. The state and policy determine the action item, while the content is sampled from a data pool using Thompson sampling instead of random generation.

Reward

\[ r_t(s_t, a_t) = \begin{cases} 10, & \text{if evaded} \\ f_s(s_t, a_t) - f_s(s_i), & \text{otherwise.} \end{cases} \]
The average Shapley values $\kappa_j$ for each feature group $g_j$ are calculated as:

$$
\kappa_j = \frac{\sum_{i \in g_j} \eta_i}{|g_j|}.
$$

The prediction of the model $f(x)$ can be expressed as the accumulation of the contributions of the feature groups

$$
f(x) = \eta_0 + \sum_{j=1}^{9} \kappa_j \cdot |g_j|.
$$
The expected effect of action on classification can be estimated as:

$$\rho_{shap}(a) = \sum_{k=3} Topk(\pi) \times M(a),$$  \hspace{1cm} (3)

The sampling probability of transition $i$ can be expressed as

$$P(i) = \frac{p_i^{\alpha}}{\sum_{j=1}^{N} p_j^{\alpha}},$$  \hspace{1cm} (4)

In PSP-Mal, we redefine the metric for each transition priority by combining the estimated Shapley prior value that reflects the expected effect of the action with the TD error

$$p(t) = 1/rank(|\delta^{TD}|) + \zeta \cdot \rho_{shap}(s_t, a_t),$$  \hspace{1cm} (5)
Thompson sampling within actions

Choose action item

\[ \text{item} = \arg \max_q Q(s_{t+1}, a; \theta) \]

Choose action content

\[ \text{content} = D(\text{item}, \arg \max \phi_i) \]

- Modify Machine Type
- Modify Timestamp
- Modify Option Header
- Remove Debug
- Break Checksum
- Add Imports
- Modify Section Name
- Section Cave Append
- Section Add
- Overlay Append

1. DOS Header
2. DOS Stub
3. PE Header
4. PE Optional Header
5. Section Table

- .text
- .data
- .idata
- ......
Dataset: We randomly select 4000 samples from VirusTotal that can be accurately detected by the target models.

We use the Evasion Rate to indicate the ability of the adversarial examples to evade the PE malware detection system.
Different priority indicators

![Graphs showing different priority indicators for EMBER and SOREL-20M datasets.](image)

Different prior weight

![Graphs showing different prior weight for EMBER and SOREL-20M datasets.](image)
Ablation study comparing vanilla D3QN to different versions of PSP-Mal.
Sparse reward

Evasion detection is a complex task, where the reward for each action may be very sparse until the evasive malware sample is obtained, making it difficult for the reinforcement learning algorithm to converge.

The agent needs to explore the state space to collect informative experiences. Ideally, the adversary uses the output of the detector’s feature extractor as the state space. However, this is difficult to achieve in black-box scenarios.

Malware representation

Countermeasures

For PSP-Mal, since the attack tends to employ additive actions to modify the file, the defender can check if slack bytes are modified or the file is padded with a large number of unexecuted bytes.
Thank You