USENIX Security '26 Artifact Appendix: SoK: The Pitfalls of Deep Reinforcement Learning for Cybersecurity

Shae McFadden[†‡§], Myles Foley[‡*], Elizabeth Bates[‡], Ilias Tsingenopoulos[¶], Sanyam Vyas[‖‡], Vasilios Mavroudis[†], Chris Hicks[‡], Fabio Pierazzi[§]
[†]*King's College London*, [‡]*The Alan Turing Institute*, [§]*University College London*, [‖]*Cardiff University*, [¶]*KU Leuven*, [*]*Devotion AI Labs*

A Artifact Appendix

A.1 Abstract

Our artifact provides the code required to run and reproduce the results of the four case studies in our paper. Each case study corresponds to a specific subset of the 11 pitfalls identified across the four stages of agent development (modeling, training, evaluation, deployment):

AutoRobust (adversarial malware creation): This case study demonstrates the Gain Attribution (E-GA) pitfall by disentangling the contribution of the state/action space from the learned policy, by comparing MDP, POMDP, and Random policies.

Link (web security testing, XSS): This case study demonstrates the Modeling Correctness (M-MC) and Partial Observability (M-PO) pitfalls by comparing the original MDP formulation against a corrected Distinct States formulation.

SQIRL (web security testing, SQL injection): This case study demonstrates the Environment Complexity (E-EC) pitfall by evaluating agents on simplified (non-sanitized) versus realistic (sanitized) endpoints.

MiniCAGE (autonomous cyber defense): This case study demonstrates the Hyperparameter Reporting (T-HR), Variance Analysis (T-VA), Policy Convergence (T-PC) training pitfalls and the Underlying Assumptions (D-UA), Non-Stationarity (D-NS) deployment pitfalls.

Each case study directory contains the modified environment, the training/evaluation scripts, the MDP specification, and a `case_study.diff` documenting all changes from the original open-source repository released with their respective papers. The *reproducibility* directory provides scripts that build per-study Python environments and rerun both quick (single-seed) and full (20-seed) results.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The artifact includes deliberately vulnerable web-application targets (WAVSEP for Link and the SMB for SQIRL) and an environment that learns to evade a ML-based malware detector (AutoRobust). These components are intended only for local, isolated evaluation:

- The WAVSEP and SMB containers should **not** be exposed on externally reachable ports, as they contain intentionally vulnerable code. They are bound to `localhost` by default.
- The AutoRobust case study trains agents to modify dynamic-analysis reports to evade detection. No live malware is executed; the artifact operates on pre-collected dynamic-analysis report features. The malware/goodware dataset needs to be downloaded from Kaggle.

Overall, no destructive operations are performed on the host, and all exploitation of vulnerabilities is contained within docker containers.

A.2.2 How to access

The final artifact will be made available on Github and Zenodo, updating the current availability artifacts:

- https://doi.org/10.5281/zenodo.20209122
- https://github.com/alan-turing-institute/DRL4Sec-Pitfalls

A.2.3 Hardware dependencies

No special hardware is required, however a CUDA-compatible GPU is *recommended* for the AutoRobust case study. The full reproduction of 20 seeds is computationally-intensive, therefore, we recommend running a single seed (`reproducibility/reproduce-one-seed.sh`).

A.2.4 Software dependencies

- `python3.9` & `python3.10` are both required as the case studies have varying dependency requirements
- `Docker Engine` & `Docker Compose` are required for the Link and SQIRL case studies.
- `Node.js` & `Chrome`/`Chromium` with its system libraries are required for the Link case study.

- The python dependencies for each of the case studies can be found in `reproducibility/envs/{case-study-name}.txt`, with the venv setup automated by `reproducibility/setup-envs.sh`.

A.2.5 Benchmarks

- *AutoRobust:* the dataset needs to be downloaded from Kaggle and placed in `AutoRobust-Case-Study/data/`. This is used for fine-tuning the malware classifier and the agent's attacks. The dataset can be found at: https://www.kaggle.com/datasets/greimas/malware-and-goodware-dynamic-analysis-reports
- *Link:* no external data required, the WAVSEP docker file is provided.
- SQIRL*:* no external data required, SQLi MicroBenchmark (SMB) docker file is provided.
- *MiniCAGE:* the MiniCAGE ACD environment is provided. The `rl-reliability-metrics` package is cloned and installed automatically by `setup-envs.sh`.

A.3 Set-up

A.3.1 Installation

Clone the repository:

- `git clone https://github.com/alan-turing-institute/DRL4Sec-Pitfalls`
- `cd DRL4Sec-Pitfalls`

Then to create the case study virtual environments:

- `cd reproducibility`
- `chmod +x *.sh`
- `./setup-envs.sh`

This creates a venv for each case study and the reproduction scripts automatically activate the correct venv for each experiment.

A.3.2 Basic Test

After `setup-envs.sh` has created the four venvs (and Docker is running), the basic functionality test can be run as follows:

- `./smoke-test.sh`, or
- `./smoke-test.sh {case-study-name}`.

This script checks that every case study is functional by: checking the imports, docker containers, and running a truncated version of the experiments to ensure the full pipelines are working. If called with the name of a specific case study (`autorobust`, `link`, `sqirl`, or `minicage`) then the script will only check that case study.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *Gain Attribution (E-GA), AutoRobust.* The observed performance of an agent can be misattributed to policy learning when environment design may provide a large fraction of that performance. Through evaluating the MDP, POMDP, and Random policies, comparisons between random and the other policies demonstrates that the design of the environment provides substantial evasion success even without a learned policy.

(C2): *Modeling Correctness & Partial Observability (M-MC, M-PO), Link.* Correcting a conflated state-space formulation improves performance. The corrected *Distinct States* formulation increases the percentage of vulnerabilities found over the Original formulation by 10.1% and raises the lower bound of the 95% confidence interval by 17.3% across 20 runs.

(C3): *Environment Complexity (E-EC),* SQIRL. Performance measured in a simplified environment is over-inflated relative to a realistic one. Agents that never experience sanitized (defended) endpoints during training suffer up to a 47.4% drop in vulnerabilities found when evaluated against sanitized defenses.

(C4): *Training & Deployment pitfalls (T-HR, T-VA, T-PC / D-UA, D-NS), MiniCAGE.* (i) A single hyperparameter change can drastically alter performance, and evaluating over many runs with confidence intervals reveals a wide behavioral distribution. (ii) Minor violations of deployment assumptions (changing red/blue agent action ordering) and non-stationary adversaries (evaluating against unseen attacker strategies) severely degrade deployed-agent performance.

A.4.2 Experiments

The reproduction of all four case studies is automated through the same reproduction scripts, therefore, after initial setup only the following scripts need to be run:

- `reproducibility/reproduce-one-seed.sh`: reproduces all settings/experiments in the paper with one seed/agent each; pass case study name to run only that case study. *Parallel*: run the script in parallel terminals once for each case study (1 day). *Sequential*: run the script once without specifying case study (2-3 days).
- `reproducibility/reproduce-full.sh`: full paper scale reproduction with 20 seeds/agents per configuration (≈weeks).
- `reproducibility/results-summary.md`: contains the output from the reproducibility scripts. If they were run in parallel, you may need to activate the `minicage venv` and run `python extract-results.py` to generate the full report.

Each script activates the correct venv per case study, starts/tears down the Docker containers, runs training and evaluation, and writes a results summary. The per-study commands remain available for running an individual study in isolation, but are not required.

(Setup): [≈ 20 human-minutes + ≈ 1 compute-hour] Complete A A.3.1 and A A.3.2.

(Run) *[≈ 5 human-minutes + ≈ 1 parallel-compute day or ≈ 3 compute days]* run one seed reproducibility script described above and then the results will be outputted into `reproducibility/results-summary.md`.

(E1): *AutoRobust [≈ 1 compute-day]* See the AutoRobust table in `results-summary.md` (corresponding to Figure 3 in the paper). The mean malware score of the *Random* policy relative to the trained *MDP/POMDP* policies demonstrate that a large portion of the performance is attributable to the state/action space design rather than learning.

(E2): *Link [≈ 8 compute-hours]* See the Link table in `results-summary.md` (corresponding to Table 1 in the paper). the *Distinct States* formulation should improve vulnerabilities found and raise the 95% CI lower bound over the *Original* formulation.

(E3): SQIRL*[≈ 8 compute-hours]* See the SQiRL table in `results-summary.md` (corresponding to Table 2 in the paper). Comparing the *Eval Sanitized* and *Eval Non-Sanitized* columns, agents trained only on non-sanitized endpoints should show a larger drop in vulnerabilities found against sanitized endpoints.

(E4): *MiniCAGE [≈ 6 compute-hours]* See the MiniCAGE tables tables in `results-summary.md` (corresponding to Figure 2 & Table 3 in the paper). The hyperparameter ablation table reports the performance at the end of training, corresponding to the right most point of Figure 2 in the paper. This table demonstrates that changing even a single hyperparameter can drastically change performance and (if run on multiple seeds) demonstrates across the seeds to motivate variance analysis. The next two tables in `results-summary.md` corresponding to the top and bottom sections of Table 3 in the paper. These demonstrate the degradation of performance in the off-diagonal results which correspond to when underlying assumptions breakdown or non-stationarity is unhandled.

A.5 Notes on Reusability

Each case study is self-contained and can be run independently of the others. The `Modified-*/` directories provide the stable version of the environments used in our evaluations with `case_study.diff` isolating the exact modifications made to the original code bases. The `reproduce-*.sh` scripts can be scaled down for quick experimentation (fewer seeds/timesteps) or up to the paper configuration.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2026/.