USENIX Security '25 Artifact Appendix: Voluntary Investments, Mandatory Minimums, or Cyber Insurance: What Minimizes Losses?

Adam Hastings

A Artifact Appendix

A.1 Abstract

These artifacts contain the following:

- The code needed to perform the empirical work in our paper
- The datasets needed to reproduce the figures in the paper
- The code (and runtime environment) needed to run the simulations and reproduce the data used in the paper

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

We do not forsee any security, privacy, or ethical concerns with running this artifact.

A.2.2 How to access

The codebase for this work is maintained at http://github.com/columbiacastl/monte-carlo-security-games/. We also provide a stable reference via Zenodo at https://doi.org/10.5281/zenodo.14728685.

A.2.3 Hardware dependencies

None.

A.2.4 Software dependencies

We have tried to make our artifacts as portable as possible. To this end we recommend installing the following:

1. Docker: https://docs.docker.com/engine/install/
2. Some Linux command line tools: `tar`, `curl`, `git`

We recommend using a Linux environment for artifact evaluation.

A.2.5 Benchmarks

None.

A.3 Set-up

We provide two methods of installation. The first—Zenodo—contains a stable artifact *and all data used in the paper*. This is necessary for a strict verification of Experiment E2 below. You may also install via GitHub which contains code but not data. Via the GitHub installation, you may *reproduce* the data used in the paper (Experiment E3) and then verify that the data matches the data in the paper (Experiment E2).

A.3.1 Installation via Zenodo

```
# download the containerized application
wget https://zenodo.org/records/15042431/files/
    artifacts.tar.gz

# extract and change directories
tar -xzvf artifacts.tar.gz
cd monte-carlo-security-games/
```

A.3.2 Installation via GitHub

```
git clone https://github.com/columbia-castl/monte-
    carlo-security-games.git
cd monte-carlo-security-games/
```

A.3.3 Building the Docker container

If you have a Docker environment, you can build a Docker image from the Dockerfile provided in the repository. This step may take 10 minutes to complete.

```
# pull a stable version of ubuntu
sudo docker pull ubuntu:22.04

# build the docker image from the Dockerfile
sudo docker build -t ae_image .

# run the docker image in an interactive container
sudo docker run -it ae_image
```

A.3.4 Basic Test

Once you have a shell to the running docker container, execute the following within the container:

```
cd root/simulator/

# build the simulator
make release

# run a small input
./run/release/run_games configs/fullsize_tiny.json
```

You may be prompted to overwrite the existing logfile for the input config `fullsize_tiny.json`. If so type "y" to proceed. The output should look something like this:

```
Creating logs/fullsize_tiny.csv

This file already exists: logs/fullsize_tiny.csv
Do you want to replace it (Y)? Or append to it (A)? Y/A/n
 >> y
started 1 games at Tue Feb 25 22:27:50 2025
finished computation at Tue Feb 25 22:28:02 2025
elapsed time: 11.2933s
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1) The curve fittings and regressions in Section 3 of the paper follow from the data acquired.
(C2) The model output data generates the figures in Sections 5–10 of the paper.
(C3) The model produces data similar to the data used in (C2).

A.4.2 Experiments

All experimental verification will happen inside the running docker container.

(E1): *Verifying (C1) [10 human-minutes + 1 compute-minute]: ...*

```
cd /root/parameter-calcs/scripts

# Re-create wealth regression (Section 3.2)
# parameters will be included in the output
# mu=1.1356, sigma=-1.1184
# other outputs located in ../figures/wealthfitting
python3 fit_marketcap_revenue_curves.py

# Re-create Figure 1 (Section 3.3)
# output will be in ../figures/ransom_regression/
python3 ransom_regression_plot.py

# Re-create security posture regression
# output will be in ../figures/posture_fitting
python3 curve_fit_posture.py

# Re-create Figure 2 (Secton 3.6)
# output will be in /root/parameter-calcs/figures/erf/
python3 plot_erf.py
```

(E2): *Verifying (C2) [10 human-minutes + 10 compute-minute]: ...*

```
cd /root/simulator/scripts/

# Generate baseline model figures (Section 5)
# outputs (Figures 3--6)  located figures/fullsize_short/
python3 run_all.py ../logs/fullsize_short.csv

# Generate the sensitivity analysis
# output  (Figure 7) located in figures/sensitivity_analysis/sensitivity_analysis_MAX_ITERATIONS=500*
python3 plot_sensitivity_analysis.py

# Generate figures for the mandated securty investments model
# outputs (Figure 8--10) in figures/fullsize_short_MANDATORY_INVESTMENT*/
python3 ./run_all.py ../logs/fullsize_short_MANDATORY_INVESTMENT\=0.01.csv
python3 ./run_all.py ../logs/fullsize_short_MANDATORY_INVESTMENT\=0.02.csv
python3 ./run_all.py ../logs/fullsize_short_MANDATORY_INVESTMENT\=0.03.csv
python3 ./run_all.py ../logs/fullsize_short_MANDATORY_INVESTMENT\=0.04.csv
python3 ./run_all.py ../logs/fullsize_short_MANDATORY_INVESTMENT\=0.05.csv


# Generate figures for the mandatory insurance model
# outputs (Figure 11--12) in figures/fullsize_short_mandatory_insurance
python3 ./run_all.py ../logs/fullsize_short_mandatory_insurance.csv

# Generate figures for the actuarially fair model
# outputs (Figures 13--14) in figures/fullsize_short_selfless_insurers
run_all.py ../logs/fullsize_short_selfless_insurers.csv

# Generate figures for model with growth
# outputs (Figures 15--16) in figures/fullsize_short_with_asset_growth_GROWTH_RATE=*/
python3 run_all.py ../logs/fullsize_short_with_asset_growth_GROWTH_RATE\=0.001.csv
python3 run_all.py ../logs/fullsize_short_with_asset_growth_GROWTH_RATE\=0.002.csv
python3 run_all.py ../logs/fullsize_short_with_asset_growth_GROWTH_RATE\=0.01.csv
python3 run_all.py ../logs/fullsize_short_with_asset_growth_GROWTH_RATE\=0.02.csv
python3 run_all.py ../logs/
```

```
    fullsize_short_with_asset_growth_GROWTH_RATE
    \=0.04.csv
```

(E3): *Verifying (C3) [1 human-hour + 20 compute-hours]:* ...

To reproduce the data used in (C2), you can re-run the simulator on all inputs.

```
cd /root/simulator/
make release # build the release version if
    you  haven't already

# Run the baseline model
./run/release/run_games configs/fullsize_short
    .json

# Run the parameter sweeps for sensitivty
    analysis
# NOTE: the sweep simulations require the
    debug binary for serial execution!
./run/debug/run_games configs/sweeps/
    MAX_ITERATIONS=500/
    sweep_ATTACKS_PER_EPOCH.json
./run/debug/run_games configs/sweeps/
    MAX_ITERATIONS=500/
    sweep_CTA_SCALING_FACTOR.json
./run/debug/run_games configs/sweeps/
    MAX_ITERATIONS=500/sweep_DEPRECIATION.
    json
./run/debug/run_games configs/sweeps/
    MAX_ITERATIONS=500/sweep_INEQUALITY.json
./run/debug/run_games configs/sweeps/
    MAX_ITERATIONS=500/
    sweep_INVESTMENT_SCALING_FACTOR.json
./run/debug/run_games configs/sweeps/
    MAX_ITERATIONS=500/sweep_LOSS_RATIO.json
./run/debug/run_games configs/sweeps/
    MAX_ITERATIONS=500/sweep_NUM_QUOTES.json
./run/debug/run_games configs/sweeps/
    MAX_ITERATIONS=500/sweep_RANSOM_B0.json
./run/debug/run_games configs/sweeps/
    MAX_ITERATIONS=500/sweep_RANSOM_B1.json
./run/debug/run_games configs/sweeps/
    MAX_ITERATIONS=500/
    sweep_RECOVERY_COST_BASE.json
./run/debug/run_games configs/sweeps/
    MAX_ITERATIONS=500/
    sweep_RECOVERY_COST_EXP.json
./run/debug/run_games configs/sweeps/
    MAX_ITERATIONS=500/
    sweep_RETENTION_REGRESSION_FACTOR.json


# Run the mandatory investment simulations
./run/release/run_games configs/
    fullsize_short_MANDATORY_INVESTMENT=0.01.
    json
./run/release/run_games configs/
    fullsize_short_MANDATORY_INVESTMENT=0.02.
    json
./run/release/run_games configs/
    fullsize_short_MANDATORY_INVESTMENT=0.03.
    json
./run/release/run_games configs/
    fullsize_short_MANDATORY_INVESTMENT=0.04.
    json
./run/release/run_games configs/
    fullsize_short_MANDATORY_INVESTMENT=0.05.
    json


# Run the mandatory insurance simulations
./run/release/run_games configs/
    fullsize_short_mandatory_insurance.json

# Run the actuarially fair simulations
./run/release/run_games configs/
    fullsize_short_selfless_insurers.json

# Run the simluations with growth
./run/release/run_games configs/
    fullsize_short_with_asset_growth_GROWTH_RATE
    =0.001.json
./run/release/run_games configs/
    fullsize_short_with_asset_growth_GROWTH_RATE
    =0.002.json
./run/release/run_games configs/
    fullsize_short_with_asset_growth_GROWTH_RATE
    =0.01.json
./run/release/run_games configs/
    fullsize_short_with_asset_growth_GROWTH_RATE
    =0.02.json
./run/release/run_games configs/
    fullsize_short_with_asset_growth_GROWTH_RATE
    =0.04.json
```

A.5 Notes on Reusability

In addition to the above arficat assets, we also include a live online demo of our model. This allows for users to run the model on custom inputs via a web browser and is much more accessible than the containerized version above. However, for usability reasons it only runs one instance of the game at a time and so it would be impractical to use it to fully reproduce the work above. However, it provides a low-resolution approximation with may be useful regardless. It is available here: https://cyberspending.cs.columbia.edu/

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/usenixsec2025/.