No more Reviewer #2: Subverting Automatic Paper-Reviewer Assignment using Adversarial Learning
Thorsten Eisenhofer, Ruhr University Bochum; Erwin Quiring, Ruhr University Bochum and International Computer Science Institute (ISCI) Berkeley; Jonas Möller, Technische Universität Berlin; Doreen Riepel, Ruhr University Bochum; Thorsten Holz, CISPA Helmholtz Center for Information Security; Konrad Rieck, Technische Universität Berlin

https://www.usenix.org/conference/usenixsecurity23/presentation/eisenhofer

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 32nd USENIX Security Symposium.
August 9–11, 2023 • Anaheim, CA, USA
978-1-939133-37-3

Open access to the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.
The main experiments require the following files

evaluation
|-- models
| |-- overlap_0.70
| |-- victim
| |-- problemspace
| |-- bibsources
| |-- llms
| |-- synonyms
|-- submissions
| |-- oakland_22
| |-- targets
| |-- budget-vs-transformer.json
| |-- featurespace-search.json
| |-- surrogates
| | |-- surrogate_targets_4.json

Targets files and bib sources are included in the repository. Pre-trained models and required target submissions are provided at blinded. Due to licensing issues, we cannot make these submissions publicly available. We do, however, publish all of our crawling scripts, dummy examples, and pre-trained models. Refer to the repository for more information.

A.2.3 Hardware dependencies

The evaluation does not require special hardware. All experiments can be performed on a “regular” server using only CPUs. We performed the experiments on a server with 256 GB RAM and two Intel Xeon Gold 5320 CPUs.

A.2.4 Software dependencies

We provide a docker container that setups the required environment and which can be used to run the experiments.

A.2.5 Benchmarks

There are no benchmarks required to evaluate this artifact.
A.3 Set-up

A.3.1 Installation

For ease of use, we include a Dockerfile with all necessary tools to reproduce the results from the paper. It can be build via

```
git clone
→ https://github.com/RUB-SysSec/adversarial-papers.git
→ adversarial-papers
cd adversarial-papers; ./docker.sh build
```

After building the container, it is possible to spawn a shell with
```
./docker.sh shell
```

All containers get automatically deleted after the shell exits (cf. the --rm flag from docker run). To make the evaluation results both easily accessible and persistent, we map subdirectories of the evaluation folder evaluation from the host inside the container. To setup all paths correctly, it is therefore necessary to invoke the docker.sh in the base directory of the project.

A.3.2 Basic Test

After building the docker container, you can test your setup by running the following command

```
./docker.sh run "python3
→ /root/adversarial-papers/src/attack.py --targets_file
→ /root/adversarial-papers/evaluation/targets/test.json
→ --format_level --workers 1 --trial_name basic-test"
```

This will start the attack for the target described in evaluation/targets/test.json. If everything is working properly, the attack should run for one iteration and immediately return successful. Results are stored in evaluation/trials/basic-test.

The main entry point for the attack is in the src/attack.py file. There are options provided to configure almost every aspect of the attack grouped into general, feature-space and problem-space specific configurations. See the documentation in the repository for further details.

When setting the number of workers to 1, the attack produces verbose output for debugging. For larger numbers, this output is not send to stdout but stored only as a log file in the respective result directory.

A.4 Evaluation workflow

The full evaluation consists of ten experiments, which requires about 6.5 CPU years to fully execute. In the following, we therefore describe only the subset of experiments we think are necessary to reproduce the major claims in the paper. Refer to the documentation in the repository for a complete description of all ten experiments include the hyperparameter search, and how to train your own models.

Each experiment is configured with a file describing all considered targets (--targets_file). These files are located at evaluation/targets. The scripts to re-generate these files are located at scripts/targets. Each target is optimized to run on a single-core and the experiments are therefore highly amenable for parallelization across CPU cores and machines. Note, that depending on the experiments more or less computer memory might be required (e.g., the black-box experiments require more memory per instance to store the surrogate models). Depending on the machine, this might limit the number of parallel executions. To get a good estimate, we will additionally report an approximated (!) maximal memory per instance (e.g., with 100 workers the experiment requires 100× the amount of this value).

Finally, for almost any experiment, it is possible to continuously check the current results which might allow to stop experiments early if the numbers have sufficiently converged (see the expected results for each experiment). Sending the interrupt signal (w/ CTRL+C) should usually stop all processes, but sometimes the scripts need a bit more persuasion. In this case, stopping the container proved to be an effective strategy (i.e., docker kill <container-id> with the container id either being autocomplete with pressing TAB or using docker ps).

A.4.1 Major Claims

In the paper, we investigate three major claims:

(C1): First, we show that the proposed attack is effective in crafting adversarial papers in a white-box setting. This is investigated with experiment (E1) described in the feature-space search paragraph in Section 4.1. The results of the experiment are reported inline in the text as well as Table 2.

(C2): Second, we demonstrate that the attack extends to different classes of transformations. This is described in the all transformations paragraph in Section 4.1 and is part of experiment (E2). The results of the experiment are reported in Figure 4.

(C3): Finally, we analyze if the attack remains viable in a black-box scenario as described in Section 4.2. We consider this in experiment (E3) and simulate the attack with varying numbers of surrogate models.

A.4.2 Experiments

(E1): Feature-space search [800 CPU hours + 31GB disk]:

We start our evaluation by examining the feature-space search of our attack. For this experiment, we consider format-level transformations that can realize arbitrary changes. Other transformations are evaluated as part of experiment (E2).
The experiment can be executed with:

```bash
WORKERS=100
./docker.sh run "python3
  /root/adversarial-papers/src/attack.py
  --targets_file
  /root/adversarial-papers/evaluation/targets/featurespace-search.json
  --reviewer_window 6
  --reviewer_offset 2
  --no_successors 256
  --beam_width 4
  --problem_space_block_features
  --feature_problem_switch 8
  --format_level
  --workers ${WORKERS}
  --trial_name featurespace-search"
```

Per worker, roughly 850MB of memory are expected. Adjust the number of parallel executions accordingly. Raw results are stored in `evaluation/trials/featurespace-search` and can be analyzed with

```bash
./docker.sh run "python3 /root/adversarial-papers/evaluation/scripts/00_featurespace_search.py"
```

Expected output (cf. Table 2 and inline in text)

**FEATURE-SPACE SEARCH**

<table>
<thead>
<tr>
<th></th>
<th>Overall success rate</th>
<th>Overall run-time</th>
<th>L1</th>
<th>L1f</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>99.67%</td>
<td>median: 7m 12s</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(E2): All transformations [1200 CPU hours + 32GB disk]:

In experiment (E1), we have focused on format-level transformations to realize manipulations. These transformations exploit intrinsics of the submission format, which effectively allows us to make arbitrary changes to a PDF file. In experiment (E2) we consider different classes of transformations as introduced in Section 3.2.

The experiment can be executed with:

```bash
WORKERS=100
./docker.sh run "python3
  /root/adversarial-papers/src/attack.py
  --targets_file
  /root/adversarial-papers/evaluation/targets/featurespace-search.json
  --reviewer_window 6
  --reviewer_offset 2
  --no_successors 256
  --beam_width 4
  --problem_space_block_features
  --feature_problem_switch 8
  --format_level
  --workers ${WORKERS}
  --trial_name featurespace-search"
```

Per worker, roughly 2300MB of memory are expected. Adjust the number of parallel executions accordingly. Raw results are stored in `evaluation/trials/budget-vs-transformer` and can be analyzed with

```bash
./docker.sh run "python3 /root/adversarial-papers/evaluation/scripts/04_all_transformations.py"
```

**Expected output (cf. left part of Figure 4)**

- Switches found no trials

(E3): Surrogates [1000 CPU hours + 46GB disk]: In practice, an attacker typically does not have unrestricted access to the target system. We therefore also assume a black-box scenario and consider an adversary with only limited knowledge.

The experiment can be executed with:

```bash
WORKERS=50
./docker.sh run "python3
  /root/adversarial-papers/src/attack.py
  --targets_file
  /root/adversarial-papers/evaluation/targets/surrogates/surrogate_targets_4.json
  --reviewer_window 2
  --delta -0.16
  --reviewer_offset 1
  --no_successors 128
  --beam_width 4
  --problem_space_block_features
  --feature_problem_switch 8
  --format_level
  --workers ${WORKERS}
  --trial_name surrogates-4"
```

Per worker, roughly 2000MB of memory are expected. Adjust the number of parallel executions accordingly. Raw results are stored in `evaluation/trials/surrogates-4` and can be analyzed with

```bash
./docker.sh run "python3 /root/adversarial-papers/evaluation/scripts/05_surrogates.py"
```

Expected output (cf. Figure 5 with ensemble size 4)

- Saved plot @ `evaluation/plots/surrogates.pdf`

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20202926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at [https://secartifacts.github.io/usenixsec2023/](https://secartifacts.github.io/usenixsec2023/).