USENIX Security '25 Artifact Appendix: Surviving in Dark Forest: Towards Evading the Attacks from Front-Running Bots in Application Layer

Zuchao MA

A Artifact Appendix

A.1 Abstract

Real-world strategies for evading front-running remain under-explored in their taxonomy and distribution due to their covert nature. Understanding these evasion tactics is vital for assessing the resilience of the current blockchain application layer and identifying areas for potential enhancement, thereby strengthening the ecosystem. In this work, we take the first step to demystify evading strategies in Ethereum and BNB Smart Chain. We propose EVScope, a novel framework combining binary analysis and machine learning to detect known and unknown evading strategies. The artifact includes the source code of EVScope, the transactions that adopt evading strategies, and the guidance to recreate results and run tools. The goals to be reproduced in the artifact evaluation include:

• Our findings uncover 32 refined strategies involving access control, profit control, execution split, and code obfuscation (Table 3 in RQ1).

• The F1-measure of EVScope reaches 0.97 on average (Table 4 in RQ4).

• We develop a straw-man approach (SA) as the baseline to conduct the comparison with EVScope. The F1-measure of SA reaches 0.30 on average (Table 4 in RQ4).

• The clustering accuracy in iterations (Figure 14 in RQ4).

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

None.

A.2.2 How to access

The original artifact can be accessed in https://zenodo.org/records/14735789. In the artifact availability evaluation, reviewers provide some useful comments to optimize our `readme` file, and I update the `readme` file in the artifact to create version V2 (https://zenodo.org/records/14778705).

A.2.3 Hardware dependencies

None.

A.2.4 Software dependencies

Please use Ubuntu OS and docker for executing the artifact. In our prepared docker image `evasion.tar`, we already set all software dependencies.

A.2.5 Benchmarks

The dataset of evading strategies is in `src_rqs.zip/RQs/dataset`. We also prepare the dataset in the docker image `evasion.tar`, which means that reviewers have no need to collect dataset.

A.3 Set-up

A.3.1 Installation

Installing docker in Ubuntu OS, for example docker 27.3.1.

A.3.2 Basic Test

Use the command to check whether docker has been installed.
`docker -version`
The expected output should be like
`Docker version XXX (e.g., 27.3.1), build YYY (e.g., ce12230)`
Import docker image by the command
`sudo docker load < evasion.tar`
Then you can see the image id of the image by the command
`sudo docker images`
Assume the image id is [image-id]. Start the container by the command
`sudo docker run -itd -name evasion [image-id]`
We can see the container id of the container by the command
`sudo docker ps`
Assume the container id is [container-id]. Go into the container by command
`sudo docker attach [container-id]`
After going into the container, please start mongodb database for querying results by the command

```
mongod -f /etc/mongod.conf &
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Our findings uncover 32 refined strategies involving access control, profit control, execution split, and code obfuscation. This is proven by the experiment described in in Section 5.2 (RQ1) whose results are illustrated in Table 3.

(C2): The F1-measure of EVScope reaches 0.97 on average. This is proven by the experiment described in Section 5.5 (RQ4) whose results are illustrated in Table 4.

(C3): We develop a straw-man approach (SA) as the baseline to conduct the comparison with EVScope. The F1-measure of SA reaches 0.30 on average. This is proven by the experiment described in Section 5.5 (RQ4) whose results are illustrated in Table 4.

(C4): The clustering accuracy increases in iterations. This is proven by the experiment described in Section 5.5 (RQ4) whose results are illustrated in Figure 14. Considering terminal interface cannot illustrate the figure, the artifact prints the points of the curves of Figure 14 in terminal for review.

A.4.2 Experiments

(E1): [1 human-minute + 5 compute-minutes + 60GB disk]: run two scripts to generate the content of Table 3 to two csv files. Then use `cat` command to show the content of Table 3 on the terminal.
Preparation: In the container, go into the folder `/home/ubuntu/RQs` by the command
`cd /home/ubuntu/RQs`
Execution: Execute the script `gen_table.py` to generate a `table.csv` file that contains line 1 to 24, line 26 to 30 of Table 3, by the command
`python3 gen_table.py`
Execute the script `gen_tab_lock.py` to generate a `table_lock.csv` file that contains line 25, line 31 to 32 of Table 3, by the command
`python3 gen_tab_lock.py`
Results: Print the content of `table.csv` to terminal by the command
`cat table.csv`
Then compare the content of terminal with Table 3 (line 1 to 24, line 26 to 30) of the paper to check the result.
Print the content of `table_lock.csv` to terminal by the command
`cat table_lock.csv`
Then compare the content of terminal with Table 3 (line 25, line 31 to 32) of the paper to check the result.

(E2): [1 human-minute + 1 compute-minute]: run a script to generate the content of Table 4 (EVScope row).
Preparation: In the container, go into the folder `/home/ubuntu/RQs` by the command
`cd /home/ubuntu/RQs`
Execution: Execute the script `precision.py` to generate the the result of EVScope row in Table 4, by the command
`python3 precision.py`
Results: Compare the content of terminal with Table 4 (EVScope row) of the paper to check the result.

(E3): [1 human-minute + 1 compute-minute]: run a script to generate the content of Table 4 (Straw-man row).
Preparation: Go into the folder `/home/ubuntu/RQs` by the command
`cd /home/ubuntu/RQs`
Execution: Execute the script `strawman.py` to generate the the result of Straw-man row in Table 4, by the command
`python3 strawman.py`
Results: Compare the content of terminal with Table 4 (Straw-man row) of the paper to check the result.

(E4): [1 human-minute + 1 compute-minute]: run a script to generate the content of Figure 14.
Preparation: Go into the folder `/home/ubuntu/RQs` by the command
`cd /home/ubuntu/RQs`
Execution: Execute the script `cluster_accuracy.py` to generate the the data points in Figure 14, by the command
`python3 cluster_accuracy.py`
Results: Compare the data points of terminal with Figure 14 of the paper to check the result.

A.5 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/.