USENIX Security '24 Artifact Appendix: Max Attestation Matters: Making Honest Parties Lose Their Incentives in Ethereum PoS

Mingfei Zhang
Shandong University
mingfei.zh@outlook.com

Rujia Li[⋆]
Tsinghua University
rujia@tsinghua.edu.cn

Sisi Duan[⋆†]
Tsinghua University
duansisi@tsinghua.edu.cn

A Artifact Appendix

A.1 Abstract

We present staircase attack, the first attack on the incentive mechanism of the Proof-of-Stake (PoS) protocol used in the Ethereum 2.0 beacon chain. Our attack can make honest validators suffer from penalties, even if they strictly follow the specifications of the protocol. We show theoretically that if the adversary controls 29.6% stake in a moderate-size system, the attack can be launched continuously, so eventually all honest validators will lose their incentives. We implement the staircase attack in our artifact and evaluate the effect of the staircase attack on a local-built testnet with 1,000 clients. The results match our theoretical analysis. This artifact aims to reproduce the results in Section 5 of our paper.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Our experiments can be launched using a local testnet (using even one local machine). Additionally, as reported in the paper, we have done responsible disclosure to the Ethereum Foundation. The Ethereum development team fixed the attack, and the mitigation took effect after the Deneb upgrade in March 2024.

A.2.2 How to access

The artifact can be accessed by cloning our public Github project. All the scripts, container images, source codes and sample output files can be accessed via the stable URL: https://github.com/Mart1i1n/Staircase-Attack/tree/85c772ece91965130d290eb1df6b489a6ba59af5.

A.2.3 Hardware dependencies

The experiments do not require particular hardware. The configuration of our computer is as follows: 2-core CPU, 2GB RAM, 100GB ROM, and a 100 Mbps bandwidth.

[⋆] Corresponding author.
[†] Sisi is also with Zhongguancun Laboratory, Shandong Institute of Blockchains, and Beijing National Research Center for Information Science and Technology

A.2.4 Software dependencies

We ran our experiments using Docker, which can be installed following the instructions of https://docs.docker.com/engine/install/. We also require Python3. The version of the Docker Engine is at least version 24.
To draw the figures in the paper, we use the Python libraries `matplotlib` and `pandas`. These can be installed via the *pip install* command.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

After installing Docker and Python3, run the following steps:

1. Git clone the repository:
   `git clone https://github.com/Mart1i1n/Staircase-Attack`
2. Enter the cloned repository directory:
   `cd Staircase-Attack/`
3. Switch to the stable version:
   `git checkout 85c772ece91965130d290eb1df6b489a6ba59af5`

A.3.2 Basic Test

After entering the repository directory `Staircase-attack` (denoted as `$HOME`), run the basic test by the command:

```
./start.sh 0
```

The following outputs are expected:

```
Input is zero, executing basic test...
[+] Building 501.3s (20/20) FINISHED
 => [internal] load build definition from geth.Dockerfile
 => => transferring dockerfile: 854B
 => [internal] load .dockerignore
 => => transferring context: 2B
 ...
[+] Building 229.0s (22/22) FINISHED
 => [internal] load build definition from attacker.Dockerfi
 le
 => => transferring dockerfile: 1.04kB
 => [internal] load .dockerignore
 => => transferring context: 2B
 ...
[+] Building 473.7s (23/23) FINISHED
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build definition from beacon.Dockerfile
 => => transferring dockerfile: 1.04kB
 ...
[+] Building 132.1s (23/23) FINISHED
 => [internal] load build definition from beacon.modify.Dock
 erfile
 => => transferring dockerfile: 1.06kB
 => [internal] load .dockerignore
 => => transferring context: 2B
 ...
[+] Building 35.3s (23/23) FINISHED
 => [internal] load build definition from validator.Dockerfi
 le
 => => transferring dockerfile: 1.02kB
 => [internal] load .dockerignore
 => => transferring context: 2B
 ...
[+] Building 37.3s (23/23) FINISHED
 => [internal] load build definition from validator.modify.D
 ockerfile
 => => transferring dockerfile: 1.04kB
 => [internal] load .dockerignore
 => => transferring context: 2B
 ...
 build ethtools image at /root/usenix24-ae/Staircase-Attack
[+] Building 1.4s (12/12) FINISHED
 => [internal] load build definition from ethtools.Dockerfile
 => => transferring dockerfile: 316B
 => [internal] load .dockerignore
 => => transferring context: 2B
 ...
[+] Running 9/9
 => Network staircase-attack_meta  Created
 => Container execute-1            Started
 => Container execute-2            Started
 => Container ethmysql             Started
 => Container beacon-1             Started
 => Container beacon-2             Started
 => Container attacker             Started
 => Container validator-1          Started
 => Container validator-2          Started
start testcase success
```

After running for 30 minutes, the experiment stops automatically. The incentives of each validator in each epoch are recorded in the file `$HOME/results/reward.csv`.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *The expected incentive of any honest validators becomes lower than 0 when the ratio of Byzantine validators exceeds 29.6%. This is proven by the experiments (E1&E2), which reproduce the results described in Figure 10, Section 5.*

A.4.2 Experiments

(Overview) *Our experiments are launched using 296, 310, 320, and 333 Byzantine validators, respectively. Our script starts the experiments for all four configurations at once and the experiment E1 is expected to finish in 96 hours. To draw the figure of our experimental result, execute E2 and it takes about 0.5 hour to generate the figure.*

(E1): *The experiment is expected to finish in 96 compute-hours.*

How to: *run the experiment with the command under* `$HOME` *:*

`./start.sh 24`

The shell script will conduct experiments for the four configurations sequentially. The output should be similar to that in the basic test.

Note: *Each experiment will run for 24 hours. If one does not want to wait for experiments so long, run the command with a smaller parameter. For example, one can run the command with parameter 1 to run each experiment for one hour.*

Results: *During each experiment, the incentive of each validator is recorded in* `csv` *files under* `$HOME/results`. An example of the `csv` *file* is as follows:

Epoch	Validator Index	Head	Target	Source
0	0	44188	101183	54483
0	1	44188	101183	54483
0	2	0	-147069	-79191
0	3	44188	101183	54483
0	4	0	-147069	-79191
0	5	0	-147069	-79191

(E2): *[0.5 compute-hour]: The experiment below processes the data from the results and plots the loss rate of honest validators.*

How to: *run the experiment with the command under* `$HOME` *:*

`./python3 plot.py`

The codes will calculate the loss rate of the first ten honest validators and plot.

Results: *The loss rate of the first ten honest validators can be seen in* `$HOME/results/figures`. An example of the figure is shown in Figure.1. The loss rate of the honest validator 0 tends to converge at 100 percent, indicating the results of Claim C1.

Note: *To accurately reproduce the results from the paper, the experiment E1 needs to be run for 96 hours. This experiment is probabilistic. If the experiment is run for a short period of time, one may not be able to obtain the same graph or observe the same trend as shown in Figure 10 of the paper. We recommend running each experiment for at least 4 hours to better observe the trend.*

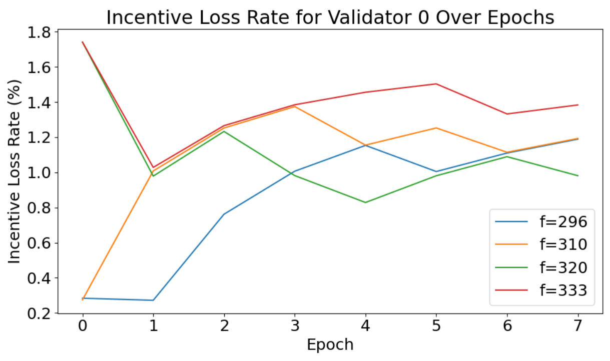

Figure 1: Loss rate of an honest validator after running the experiment for one hour.

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/usenixsec2024/`.