USENIX Security '25 Artifact Appendix: Does Finality Gadget Finalize Your Block? A Case Study of Binance Consensus

Rujia Li
Tsinghua University

Jingyuan Ding
Shandong University

Qin Wang
CSIRO Data61

Keting Jia
Tsinghua University

Haibin Zhang
Yangtze Delta Region Institute of Tsinghua University

Sisi Duan
Tsinghua University

A Artifact Appendix

A.1 Abstract

Fast Finality (FF) is a core mechanism of BNB Smart Chain (BSC). BSC claims that its FF mechanism can finalize blocks in $O(1)$ time, simultaneously reducing latency and improving stability. We present three attacks, showing BSC fails to finalize blocks in constant time and may even simply fail to achieve liveness. We implement attacks and evaluate the effect of the three attacks on a local-built testnet with 21 clients. The results match our theoretical analysis. This artifact aims to reproduce the results in Section 7 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). As reported in the paper (Section 1), we have done responsible disclosure to the Binance team. The team has confirmed and acknowledged all the attacks.

A.2.2 How To Access

Our artifact is openly available on GitHub at the fixed and stable commit https://github.com/tsinghua-cel/bsc-attack-experiment/tree/a40ca69ff22aa1b3a2ee61a109825c533044fb58. The repository contains all the scripts, container images, source codes, and sample output files. For long-term preservation, we have also archived an identical snapshot on Zenodo: https://doi.org/10.5281/zenodo.15852608.

A.2.3 Hardware Dependencies

The experiments do not require particular hardware. Our setup used a standard machine with an 8-core CPU, 32GB RAM, 500GB ROM storage, and a 100 Mbps internet connection.

Note: We recommend allocating at least **16GB** of memory to Docker to ensure the smooth execution of our experimental environment.

A.2.4 Software dependencies

We ran all experiments using Docker (version 27 or higher), which can be installed by following the official guide at https://docs.docker.com/engine/install/.

A.2.5 Benchmarks

Not applicable.

A.3 Set Up and Run Experiments

A.3.1 Installation

No additional setup is required. All dependencies and environment configurations are encapsulated in a Docker image to ensure consistent and reproducible results. For reviewers who are interested in customizing their local testing environment, detailed manual instructions along with raw test scripts and data are available at https://github.com/tsinghua-cel/bsc-attack-experiment/tree/a40ca69ff22aa1b3a2ee61a109825c533044fb58.

A.3.2 Pull Docker Images (One-time setup)

The following commands show how to pull the images for attack 1, attack 2, and attack 3 (involving two configurations).

```
docker pull erick785/bsc-attack-1:latest
docker pull erick785/bsc-attack-2:latest
docker pull
 ↪  erick785/bsc-attack-3-bootnode:latest
docker pull
 ↪  erick785/bsc-attack-3-staticnode:latest
```

A.3.3 Basic Test

Impact of the attacks:

- **attack 1**: Slowing down the finality of blocks.
- **attack 2**: No blocks can be finalized.
- **attack 3**: No blocks can be finalized for a period of time.

Table 1: Column meanings in datalog (results)

Column	Meanings	Description
1	The latest block height	The number of the latest block that has been generated by the current node (`LatestBlock`).
2	Finalized block height	Block number that has been finalized (`FinalizedBlock`).
3	Attestation of block header	`true` means that the block received a vote attestation, `false` means that it did not.

** All attacks start at block height 250 **

Reviewers can then run the artifact by executing:

For attack 1

- Run command:

```
touch 1.txt && docker run -it --rm -v
 ↪  ./1.txt:/app/query/21.txt
 ↪  erick785/bsc-attack-1:latest
```

- The following outputs are expected (cf. Table 1):

```
1,0,false
2,0,false
3,0,false
4,0,false   // votes not collected
...
210,0,false
211,0,false
212,0,true
213,211,true
214,212,true
...
```

The experiment for **attack 1** will be automatically terminated after 100 minutes of execution. The results generated by the Docker container will be saved to the file `1.txt`.

For attack 2

- Run command:

```
touch 2.txt && docker run -it --rm
 ↪  -v ./2.txt:/app/query/21.txt
 ↪  erick785/bsc-attack-2:latest
```

- The following outputs are expected:

```
1,0,false
2,0,false
3,0,false
4,0,false  // votes not collected
...
```

The experiment for **attack 2** will be automatically terminated after 100 minutes of execution. The results generated by the Docker container will be saved to the file `2.txt`.

For attack 3 (bootnode mode)

- Run command:

```
touch 3.1.txt && docker run -it --rm -v
 ↪  ./3.1.txt:/app/query/21.txt -e
 ↪  DELAY_INTERVAL_MS=25
 ↪  erick785/bsc-attack-3-bootnode:latest
```

- The following outputs are expected:

```
1,0,false
2,0,false
3,0,false
4,0,false  // votes not collected
...
```

The experiment for **attack 3 (bootnode mode)** will be automatically terminated after 100 minutes of execution. The results generated by the Docker container will be saved to the file `3.1.txt`. Notably, the `DELAY_INTERVAL_MS` field can be set to *25*, *50*, or *75*, indicating three separate experiments for the bootnode mode. We only present the case with 25-ms delay.

For attack 3 (full connection mode)

- Run command:

```
touch 3.2.txt && docker run -it --rm -v
 ↪  ./3.2.txt:/app/query/21.txt -e
 ↪  DELAY_INTERVAL_MS=25
 ↪  erick785/bsc-attack-3-staticnode:latest
```

- The following outputs are expected:

```
1,0,false
2,0,false
3,0,false
4,0,false  // votes not collected
...
```

The experiment for **attak 3 (full connection mode)** will be automatically terminated after 100 minutes of execution. The results generated by the Docker container will be saved to the file `3.2.txt`. Notably, the `DELAY_INTERVAL_MS` field can be set to *25*, *50*, or *75*, indicating three separate experiments for the full connection mode.

Note: Each execution launches a one-time Docker container. Once exited (e.g., via `Ctrl+C`), the container is removed and must be restarted from scratch. There is no need to delete the txt file (e.g., `1.txt`); re-running the command will overwrite the txt file with new results.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *In attack-I (attack 1), three blocks are finalized among 21 blocks, and the attack slows down the finality of blocks. This is proven by the experiment (E1), which reproduces the results described in Figure 8(a), Section 7.*

(C2): *In Attack-II (attack 2), the finalized block height remains unchanged, indicating that the system is unable to finalize new blocks and experiences a liveness failure. This is proven by the experiments (E2), which reproduce the results described in Figure 8(b), Section 7.*

(C3): *In attack-III (attack 3), we assess the **bootnode** mode and **full connection** mode of P2P. We set up the experiments so that Byzantine validators send their blocks 25 ms, 50 ms, and 75 ms ahead of time, and the results show that there exist some periods where no blocks can be finalized. In the extreme case, none of the 130 slots can be finalized (there is uncertainty in the outcome of the run). This is proven by the experiments (E3), which reproduce the results described in Figures 8(c)-8(f), Section 7.*

A.4.2 Experimental results

Overview: *In each experiment, we run more than 2,000 slots on our testnet and assess the fast finality and reward. Our attack is launched at the 250-th slot.*

(E1): For **attack 1**, the experiment is expected to finish in 100 minutes. However, for quicker testing, one may choose to run only 300 slots, which takes about 15 minutes.

Results: *During this experiment, the number of the latest block and finalized block is recorded in* `1.txt`. *An example of the txt file is as follows:*

```
1,0,false
2,0,false
3,0,false
4,0,false
...
210,0,false
211,0,false
212,0,true
213,211,true
214,212,true
...
247,245,true
248,246,true
249,247,true
250,248,true # Start attack.
251,248,false # Finalized block stops to increase
252,248,false
253,248,false
...
266,248,false
267,248,false
268,248,true
269,267,true
270,268,true # Finalized blocks catch up to 268
271,269,true # Repetitive pattern for below blocks
272,269,false
273,269,false
274,269,false
...
289,270,false
290,288,true
291,289,true
292,290,true
293,290,false
...
```

The three data fields are: The latest block height, finalized block height, and presence of attestation.

Note: This experiment is probabilistic. The exact data may vary across runs. However, the observed trend remains consistent: before the attack is launched, the difference between the latest block height and the finalized block height consistently stays at 2(e.g., [247,245], [248, 246], [249, 247]..). After initiating the attack at slot 250, this gap begins to widen. The finality of blocks no longer progresses in step with the increasing latest block height. This observation aligns with our claims in Table 1 and Figure 8(a): Attack 1 slows down the finality of blocks.

(E2): For **attack 2**, the experiment is expected to finish in 100 minutes. For quicker testing, one may choose to run only 300 slots, which takes about 15 minutes.

Results: *During this experiment, the number of the latest block and finalized block is recorded in* `2.txt`. *The fields in the txt file are the same as in **E1**. The output of the txt file is as follows:*

```
...
210,0,false
211,0,false
212,0,true
213,211,true
214,212,true
...
248,246,true
249,247,true
250,248,true # Start attack
251,248,false # Finalized blocks stall
252,248,false
253,248,false
254,248,false
255,248,false
256,248,false
...
```

Before launching the attack, the difference between the latest block height and the finalized block height consistently remained at 2. After we launch the attack at slot 250, we can observe that the last finalized block height remains fixed at 248, indicating that none of the subsequent blocks can receive more than $N - f$ matching attestations. As a result, no new block can be finalized. This observation supports our claims in Table 1 and Figure 8(b): Attack 2 causes a liveness failure.

(E3): For **attack 3**, we consider both bootnode mode and full connection mode, with each mode supporting configurable delay durations. For quicker testing, one may run only 300 slots, which takes around 15 minutes.

——- E3.1 bootnode mode ———-

Results: *During this experiment, the number of the latest block and finalized block is recorded in* `3.1.txt`. *The fields in the txt file are the same as in **E1**. The output of the txt file is as follows:*

```
...
432,429,false
433,429,true
434,429,false
435,429,false
...
445,429,false
446,429,false
447,429,false
448,429,false
449,429,false
450,429,false # At block 450, the finalized
block height remains 429 (21 blocks delayed)
451,429,false
452,429,false
...
470,429,false
471,429,false
472,429,false
473,429,true
474,429,false
475,429,true
476,474,true # After a delay, the finalized
block starts catching up
477,474,false
478,474,true
479,477,true
...
```

Note: This experiment is probabilistic. The exact data may vary from run to run. However, the observed trend remains consistent: After launching attack-III, the finalized blocks no longer increase in synchronization with the latest blocks, but rather increase in a delayed manner. This observation supports our claims in Table 3 and Figure 8(c)-8(e), Section 7 that attack-III prevents blocks from being finalized for a period of time.

——- E3.2 full node mode ———-

Results: *During this experiment, the number of the latest block and finalized block is recorded in* `3.2.txt`. *The fields in the txt file are the same as in **E1**. The output of the txt file is as follows:*

```
...
210,0,false
211,0,false
212,0,true
213,211,true
214,212,true
...
651,632,false
652,632,false
653,632,false
654,632,false
655,632,false
656,632,true
657,632,false
658,632,true
659,657,true
660,657,false
661,657,false
662,657,true
663,657,false
664,657,true
665,663,true
666,663,false
667,663,true
668,666,true
669,667,true
670,667,false
671,667,true
...
```

Note: This experiment is probabilistic. The exact data varies from run to run. However, the observed trend remains consistent: after launching attack 3, the finalized blocks no longer increase in synchronization with the latest blocks, but rather increase in a delayed manner. This observation supports our claims in Figure 8(d) and 8(f) that attack-III prevents blocks from being finalized for a period of time.

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/.