BREWFORCE

Rewards Internal Review

Summary

Project Name	Espresso
Language	Rust
Codebase	https://github.com/EspressoSystems/espresso-network
Delivery Date	15/04/2025
Team	0xKato, Jarred Parr
Commit(s)	913eae9dbc6efd977eed49cf7bde399c35def73f

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	3	0	0	0	0	0
● High	2	0	0	0	0	0
● Medium	4	0	0	0	0	0
● Low	6	0	0	0	0	0

STAKE-1 | Delegators amount is overwritten upon new delegation

Category	Severity	Location	Status
State Violation	● Critical	stake_table.rs::L121	Confirmed

Description

When delegating, the confirmation layer reads the emitted events and and increments the total delegated state as well as the individual delegators delegated amount, the problem is in the way a new delegation from the same delegator is handled:

```
validator_entry.delegators.insert(delegator, amount);
```

This will override the delegators initial delegation with the new amount and they will not be accounted for by the full amount that was delegated to them but rather their last delegated amount.

Recommendation

Increment the delegators amount if they have an entry or insert new entry if there isn't any.

Resolution

L1-1 | Failure To Recover If Block Depth Too Large

Category	Severity	Location	Status
Liveness Violation	● Critical	l1.rs::L881	Confirmed

Description

If the depth of messages is too large, the node can never recover. For example, suppose a low-effort spam attack occurs on the stake table (which can currently happen due to an issue where we can trivially spam state (GLOBAL-1)), an attacker could create a flood of messages. Even a smaller-scale attack can put undue pressure on the memory of the hosted machine.

Recommendation

We should try to capture information related to the latest state of the stake table to ensure that we don’t have to read from genesis each time.

Resolution

STAKE-2 | Invalid block height being passed to stake table fetch

Category	Severity	Location	Status
Liveness Violation	● Critical	stake_table.rs::L832	Confirmed

Description

The hotshot instead of L1 block height is passed to `get_stake_table_by_epoch` which will cause the node to never catch up, halting startup.

```
get_stake_table_by_epoch(epoch, address, block_header.height())
```

Recommendation

Switch to use the l1 finalized block height.

Resolution

STAKES-1 | Lost funds with multiple undelegate

Category	Severity	Location	Status
Implementation Error	● High	StakeTable.sol::L403	Confirmed

Description

Users can delegate and undelegate their tokens to increase or decrease a validator's stake. The current undelegation implementation is quite unrestricted. When undelegating, a user specifies an amount of their already delegated tokens to remove. This amount is immediately subtracted from their delegation and added to an `undelegations` mapping, where it becomes withdrawable after a fixed period of time.

```
delegations[validator][delegator] -= amount;
undelegations[validator][delegator] = Undelegation({ amount: amount, unlocksAt: block.timestamp + exitEscrowPeriod });
```

A problem arises if a user initiates an undelegation and later decides to undelegate additional tokens. The second call will overwrite the previous undelegation entry, while the previously subtracted tokens remain untracked. This results in a loss of funds, as there is no way to recover the tokens from the contract once overwritten.

Recommendation

Only allow a single undelegate to take place at a time

Resolution

GLOBAL-1 | delegation spam can cause excessive computation on a node

Category	Severity	Location	Status
Implementation Error	● High	Global	Confirmed

Description

There is no limit of amount of delegatees a validator can have or a min amount required to delegate meaning a malicious actor can spam a validator with unlimited amount of delegates which the validator will have to compute reward splits for a potentially high volume of delgators, which can waste computation. This may also create tons of message spam which due to L1-1 and cause a lookup failure from our RPC provider.

Recommendation

Consider adding a min delegate amount to increase the cost.

Resolution

TYPES-1 | Underflow can occur in root block calculation

Category	Severity	Location	Status
Panic	● Medium	utils.rs::L353	Confirmed

Description

The `root_block_in_epoch` function calculates the root of the current epoch as `epoch_height * epoch - 5`, but it lacks checks to ensure the resulting height is above a minimum threshold. If misconfigured, this can lead to an underflow.

Recommendation

Add a check to ensure the *`epoch_height > 4`*

Resolution

L1-2 | Websocket urls should not be fetched in arbitrary order

Category	Severity	Location	Status
Implementation Error	● Medium	l1.rs::L459	Confirmed

Description

The provider is fetched via a mod over an infinitely increasing integer value. Because of this, we get a url at random from the list. We should not do this and, instead, always use the first entry in the list (the on-prem RPC provider) and always use the others as fallbacks.

Recommendation

Implement fallback behavior instead of a round robin RPC access.

Resolution

STAKE-3 | Lack of schnorr key validation allows for key theft

Category	Severity	Location	Status
Implementation Error	● Medium	stake_table.rs::L61	Confirmed

Description

The schnorr keys are not validated in the contract or the rust consumer, resulting in the potential for a malicious actor to spoof a schnorr key of another node. This is dangerous because the key is leaked on each invocation of the contract, so anyone could read the key and then front-run the person and insert themselves for the same key, blocking a node from being entered into the stake table.

Recommendation

Verify the schnorr key.

Resolution

REWARD-1 | Leader may accumulate all rewards

Category	Severity	Location	Status
Implementation Error	● Medium	reward.rs::L349	Confirmed

Description

Due to GLOBAL-1 a validator is incentivized and able to dilute their delegation pool such that the rounding error introduced in reward calculation can result in the validator obtaining the entire reward by default.

Recommendation

Add a minimum stake requirement and set a cap on the number of delegators.

Resolution

STAKE-4 | Improve mapping in the contract

Category	Severity	Location	Status
Implementation Error	● Low	stake_table.rs::L534	Confirmed

Description

The Stake Table recovery mechanism does not currently have a reliable way to get state without recovering from genesis if the state does not currently exist in the persistence layer.

Recommendation

Add a new slot to `StakeTable.sol` which allows for a validator to be mapped to its delegators. This is a reliable source of truth as the validator and delegator can be updated alongside any changes (like the withdrawal of a delegator). The stake table code in rust can then just checkpoint from the contract.

```
mapping(address validator => address[] delegators) public validatorToDelegators;
```

Resolution

STAKES-2 | Nodes that de-register cannot ever re-join the network

Category	Severity	Location	Status
Implementation Error	● Low	StakeTable.sol::L356	Confirmed

Description

If a node de-registers, the enum type prevents them from re-joining the network ever.

Recommendation

Change the enum to not check for the excited state.

Resolution

STAKES-3 | We take two slots in the contract whenever a key update occurs

Category	Severity	Location	Status
Implementation Error	● Low	StakeTable.sol::L474	Confirmed

Description

When switching between keys in `updateConsensusKeys` we never deallocate the memory associated with the prior key, leaving it in the state forever. Over time, this could consume more resources than necessary.

Recommendation

Zero out old keys via: `blsKeys[_hashBlsKey(OLD_KEY)] = false` and change the signature to also get and track the old key so that way it is known.

Resolution

REWARD-2 | Leader can double-dip in rewards

Category	Severity	Location	Status
Implementation Error	● Low	reward.rs::L349	Confirmed

Description

Due to the current system structure, where leaders who want to stake their own funds must delegate to themselves, a conflict arises in how rewards are calculated. Leaders are allowed to charge a commission on rewards earned by delegators. However, the system treats all delegators equally including the leader’s own delegation resulting in the leader earning rewards both from the commission and from their own stake, which also gets reduced by their own commission.

Recommendation

Consider whether a leader should be able to delegate as well as having a commission on top

Resolution

STAKE-5 | Superfluous code

Category	Severity	Location	Status
Superfluous code	● Low	stake_table.rs::L462	Confirmed

Description

The `new_stake` method uses the exact same method for polling `eligible_leaders` and the `stake_table` there’s no point in having distinct lists, and we use them without transformation in the end of the function as well.

Recommendation

Just merge the two things together.

Resolution

STAKE-6 | You can update to a key that is already in the set

Category	Severity	Location	Status
Implementation Error	● Low	stake_table.rs::L151	Confirmed

Description

The function `ConsensusKeysUpdated` does not check that a given bls key is not already in use, which could be done erroneously or deliberately, resulting in a node losing the ability to be a validator in the worst case.

Recommendation

Check if the key is in use before assignment. Also, validate the key.

Resolution

Disclaimer

This report is an internal review and should not be considered an “endorsement” or “disapproval” of any particular part of the codebase. It does not provide any warranty or guarantee regarding the absolute bug-free nature of the analyzed technology, nor does it reflect the economics, value, business model, or legal compliance.

This report should not be used to make investment or involvement decisions. It is not a substitute for external reviews and should not be taken as investment advice. Instead, it serves as part of an internal assessment process aimed at helping improve the quality of the code.

The goal is to help reduce attack vectors and risks associated with evolving technologies, we do not claim any guarantees regarding security or functionality of the technology analyzed. We do not guarantee the explicit security of the audited code, regardless of the findings.