Security Review of Aave Governance-V2 Update

Scope

The scope of the assessment is the contracts `Executor.sol` which was formally verified, and `AaveEcosystemReserveV2.sol`, `ProposalPayloadAaveEcosystemReserveWithVoting.sol`, and `ProposalPayloadNewLongExecutor.sol`, which were manually reviewed since the implementation involves only calls to function with concrete values. The first two contracts are version bumps of the existing Aave ecosystem reserve and long Executor contracts. The latter two contracts are payloads destined to be executed on the current short and long Executors to allow the version bumps.

One security engineer and one security researcher reviewed these contracts in detail.

The verification and the manual review were finished on the 11th of September, reviewing commit 06d71de. The formal specification can be found here.

⚠️ Note that the review's focus was the changes made to the existing contracts and not a complete review of the current code.

List of Main Issues Discovered

Severity: Low

Issue:	Zero minimum quorum
Description:	In `Executor.sol`, it is possible to set the `minimumQuorum` to zero. In this case, the significance of a quorum becomes redundant.
Response:	A check was added in commit 0d9f679

Severity: Low

Issue:	Zero or 100% vote differential
Description:	In `Executor.sol`, it is possible to set the `voteDifferential` to zero. In this case, it is expected that any proposal with more upvotes than downvotes will pass (even by a single vote). However, this criterion could be more strict due to rounding errors, meaning that a proposal with more upvotes than downvotes will still be rejected.
Response:	A check was added in commit 0d9f679

Severity: Low

Issue:	Zero or 100% proposition threshold
Description:	In `Executor.sol`, it is possible to set the `propositionThreshold` to zero. In this case, every proposal could be submitted with no restriction whatsoever on the voting power of the submitter. The threshold could also be set to 100%, but smaller values could prevent almost any proposal from being submitted.
Response:	A check was added in commit 0d9f679

Severity: Low

Issue:	Zero grace period
Description:	In `Executor.sol`, it is possible to set the `gracePeriod` in the constructor to zero. If done on purpose or by accident, executing proposals' action sets will be impossible.
Response:	A check was added in commit 06d71de

Informational

Issue:	Potential use of Level 1 proposal system as a voting proxy for Level 2 proposal system.

Issue:	**Potential use of Level 1 proposal system as a voting proxy for Level 2 proposal system.**
Description:	The process of using the `AaveEcosystemReserveV2`'s voting power on a Level 2 proposal can be similarly repeated in the future. By raising a proposal on Level 1 to upgrade the reserve's implementation and bump the revision number, `initialize` will be available to be used once again to boost a new Level 2 proposal, or in a worse case, a dedicated functionality for this procedure could be added. Considering the new lower Level 2 conditions and the current AAVE amount in reserve, such functionality, in effect, will allow passing a Level 2 proposal by passing a proposal on Level 1. Taking the current state of things, according to BGDLabs' proposal, the new 'YES' amount needed to pass a vote on Level 2 will be lowered to 1'040'000, while the `AaveEcosystemReserveV2` has a voting power of 1'625'351 votes. This is well above the necessary 'YES' amount needed to pass a proposal, making even the differential 'YES/NO' barrier very hard to use as a veto mechanism.
Aave Response:	We don't think anybody was aware of this possibility before the proposal. We want to make a clear statement that it should not be something to normalize. Yes, it is clear that it becomes easier to "cheat" on future Level 2 votes, but there are multiple protections to avoid that (Guardian with cancel, etc.).

Disclaimer

The Certora Prover takes a contract, and a specification as input and formally proves that the contract satisfies the specification in all scenarios. Notably, the guarantees of the Certora Prover are scoped to the provided specification, and the Certora Prover does not check any cases not covered by the specification.

We hope that this information is useful, but provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the contract is secure in all dimensions. In no event shall Certora or any of its employees be liable for any claim, damages, or other liability, whether in an action of contract, tort, or otherwise, arising from, out of, or in connection with the results reported here.

Contracts Overview

It is believed that the current requirements for raising and passing governance proposals that require Level 2 permissions are too strict. These strict requirements make it very hard to pass proposals even when there's a significant consensus about them. In practice, it acts as an obstacle to proceeding with contributions to the ecosystem.

In order to lower the requirements of the long Executor, a Level 2 proposal must pass a community vote. Since it's believed that it's near impossible to reach the current voting orders, a plan was formulated by BGDLabs - as a one-time act, the AAVE tokens in the Aave DAO reserve will be used to vote in favor of this Level 2 proposal. This will add a boost of "yea" votes that will reduce the Level 2 requirements to more realistic and achievable values in the cost of passing a Level 1 governance proposal. To learn more about the proposal read the RFC in the Aave forum.

The set of contracts we reviewed implements this plan in the form of 2 payloads that will allow the ecosystem reserve and long Executor updates and the two aforementioned updated contracts.

Four contracts were reviewed:

1. `Executor.sol`: An update of the existing long Executor contract, which contains the following changes:

   1.1 Lower YES voting requirements on Level 2 to 6.5% of the supply or 1'040'000 AAVE.
   1.2 Lower YES/NO differential requirements on Level 2 to 6.5% or 1'040'000 AAVE.
   1.3 Lower proposing power requirements on Level 2 to 1.25% or 200'000 AAVE.
   1.4 Lower the voting delay system-wide to 1 day.

2. `AaveEcosystemReserveV2.sol`: An update of the existing AaveEcosystemReserveV2 that enables using the reserve voting power to vote on Level 2 proposals. It introduces a change in the `initialize` method implementation that allows voting for a single Level 2 proposal.

3. `ProposalPayloadAaveEcosystemReserveWithVoting.sol`: A payload contract, meant to run on the short Executor to use the AAVE Ecosystem Reserve's voting power for voting on a Level 2 proposal. In practice, the payload upgrades `AaveEcosystemReserveV2` to a new implementation and call `initialize` with input parameters to vote on a Level 2 proposal.

4. `ProposalPayloadNewLongExecutor.sol`: A payload contract meant to run on the long exectuor to authorize a new long Executor with looser requirements. The payload is:
   4.1 Setting a voting delay on the `AAVE_GOVERNANCE_V2` contract.
   4.2 Authorizing the new Executor to perform actions on the `AAVE_GOVERNANCE_V2` contract.
   4.3 Transfering the `AAVE_GOVERNANCE_V2`'s ownership, `AAVE_PROXY`'s administration, and `STK_AAVE_PROXY`'s administration from the old long Executor to the new one.
   4.4 Transfering the `ABPT_PROXY`'s administration and `STK_ABPT_PROXY`'s administration to the short Executor.

Formal Verification

Goals

Approaching verification, we decided to perform small and straightforward checks that target the changes introduced to the contracts rather than a purely manual review.

The main objective of this contract is to execute the proposal actions set (or payload) by using low-level calls, including delegate calls. This functionality is rather general and meant to allow the execution of future transactions approved by the Aave DAO, which we cannot predict in advance.

Thus we ignore the transaction execution.

Notations

✓ indicates the rule is formally verified on the latest reviewed commit. We write ✓* when the rule was verified on the simplified assumptions described above in "Assumptions and Simplifications Made During Verification".

❌ indicates the rule was violated under one of the tested versions of the code.

🔁 indicates the rule is timing out.

Our tool uses Hoare triples of the form {p} C {q}, which means that if the execution of program C starts in any state satisfying p, it will end in a state satisfying q. This logical structure is reflected in the included formulae for many of the properties below. Note that p and q here can be thought of as analogous to `require` and `assert` in Solidity.

The syntax {p} (C1 ~ C2) {q} is a generalization of Hoare rules, called relational properties. {p} is a requirement on the states before C1 and C2, and {q} describes the states after their executions. Notice that C1 and C2 result in different states. As a special case, C1 ~op C2, where op is a getter, indicates that C1 and C2 result in states with the same value for op.

Properties

1. properDelay ✔

The execution delay is always bounded by the minimum and maximum values.

```
MINIMUM_DELAY() <= getDelay() ∧ getDelay() <= MAXIMUM_DELAY()
```

2. nonZeroGracePeriod ✘

The grace period is always larger than zero.

```
GRACE_PERIOD() > 0
```

3. nonZeroVotingDuration ✔

The voting duration is always larger than zero.

```
VOTING_DURATION() > 0
```

4. nonZeroPropositionalThreshold ✘

The proposition threshold is always larger than zero.

```
PROPOSITION_THRESHOLD() > 0
```

5.nonZeroMinimumQuorum ✘

The minimum quorum is always larger than zero.

```
MINIMUM_QUORUM() > 0
```

6. thresholdLessThan100 ✘

The proposition threshold is always smaller than 100%.

```
PROPOSITION_THRESHOLD() < ONE_HUNDRED_WITH_PRECISION()
```

7. differentialLessThan100 ✘

The voting differential is always smaller than 100%.

```
VOTE_DIFFERENTIAL() < ONE_HUNDRED_WITH_PRECISION()
```

8. onlyAdminChangesAdmin ✓

Only the admin (or pending admin) can change the admin.

```
{
    _admin = getAdmin() ∧
    _pendingAdmin = getPendingAdmin()
}
    < call to any function f >
{
    admin_ = getAdmin()
    ∧ (f == acceptAdmin() => admin_ == _pendingAdmin && msg.sender == _pending
    ∧ (f ≠ acceptAdmin() => _admin != admin_ => msg.sender == _admin)
}
```

Manual Review Goals

During the review of the code, the following checks have been performed:

Addresses

1. All addresses of external contracts being used match the existing contracts.

Correct Setting of Parameters and Values

2. The new long Executor contains all the correct values for the parameters from BGD's proposal, i.e., a `propositionThreshold` of 200,000 AAVE tokens, a `minimumQuorum` of 1,040,000 AAVE tokens, and a `voteDifferential` of 1,040,000 AAVE tokens.
3. `AAVE_GOVERNANCE_V2` is set with a correct voting delay of 1 day.

Privileges

4. The long and short Executors have the necessary privileges to execute the payload actions.
5. The payload sets new administration and ownership roles that match BGD's proposal, e.g., The long Executor will receive an authorization role over the `AAVE_GOVERNANCE_V2` and more.

Proxy and implementation's upgrade

6. The new `AaveEcosystemReserveV2` implementation contract keeps the same storage variables' order for correct correlation to the proxy storage variables.
7. The new implementation enables the invocation of the `initialize` method.

8. The `initialize` method can be invoked only once per version update.

Main Review Conclusions

Addresses

1. The addresses specified in the contracts fully matched the existing relevant contracts on the network.

Correct Setting of Parameters and Values

2. The new long Executor was set correctly with all the parameter values.
3. `AAVE_GOVERNANCE_V2` has been set with a voting delay of 7200 blocks, which translates into one day, assuming an average mining rate of 1 block every 12 seconds.

Privileges

4. The short Executor does have a `AaveEcosystemReserveV2 proxy` ownership permission, and the current long Executor does have a `AAVE_GOVERNANCE_V2` ownership role along with `AAVE_PROXY`, `STK_AAVE_PROXY`, `ABPT_PROXY` and `STK_ABPT_PROXY` administration roles. Therefore all the actions implemented in the payload should not revert due to inadequate privileges.
5. In `ProposalPayloadNewLongExecutor.sol`, the long Executor sets all the administration and ownership roles to match BGDLabs' proposal.

Proxy and implementation's upgrade

6. The new implementation contract does keep the same storage variables order; hence there's a correct correlation to the proxy storage variables.
7. The new `AaveEcosystemReserveV2` implementation changes the value of the `REVISION` variable from 4 to 5; therefore, after the upgrade, it will be possible to call `initialize.`
8. The new implementation allows calling `initialize` exactly once.