[

proof-of-reserve_banner

1920×768 101 KB

](https://europe1.discourse-cdn.com/business20/uploads/aave/original/2X/3/3e56ae2fd735b05cf7164dd801ae273822470ea2.jpeg)

TL;DR

We have been working on the solution to integrate Chainlink's Proof-of-Reserve (PoR) into Aave, and now we present the details of the solution, initially to be applied to Aave v2 and v3 Avalanche.

The repository with the solution is on https://github.com/bgd-labs/aave-proof-of-reserve

The scope

Before starting working on this project, we knew PoR would be beneficial to the Aave protocol. But the most fundamental aspect is to define what we want to solve in this initial phase and what can be improved later on.

Our current proposal to the community is as follows:

- The initial release will only cover one network where Aave is deployed: Avalanche, for both Aave v2 and Aave v3 instances

. Only 1 network makes sense because it allows adding more protections to one of the biggest non-Ethereum Aave networks. In addition, Chainlink has been working on this instance of PoR for the longest time, so we feel confident enough about the reliability.

- Not all the cases that can be covered by PoR will be in this phase.

A really extensive integration implies changes to the core Aave v3 protocol, which have associated more development, big gas implications, and security procedures time, given their nature. We think it is better to try to introduce some protections in the short term, but not all, compared with not introducing anything in the short term and introducing everything later.

The problem

At the moment, in Aave v3 Avalanche there are multiple assets with a "bridged" nature: DAI.e, USDC.e, USDT.e, AAVE.e, LINK.e, WBTC.e, BTC.b and WETH.e.

As explained in the introductory post, these assets have 2 sides: one deposit account in the network where the asset originated (Ethereum in the case of the *.e assets) and a token smart contract on the "usage" chain (Avalanche in this case).

Given that architecture, there are 2 main attack vectors to exploit "bridged" assets:

1. An exploit on the origination network, extracting tokens from the deposit contract.
2. An exploit on the usage network, generally minting tokens that have no backing on the origination network, usually called an "infinite minting attack".

The solution

At the moment, not Aave v2, but Aave v3 already has some protection against attack type 2), infinite minting, by setting supply caps on all the assets.

Our solution is focused on adding more protection for type 1) (extraction of deposited tokens on the origination network), for both Aave v2 and v3, together with adding an extra layer of security against 2) (infinite minting on usage network, e.g. Avalanche):

- A ProofOfReserveAggregator

smart contract, acting as a registry of pairs (asset address, proof of reserve feed address) and also implementing and exposing an areAllReservesBacked()

function, which, for a list of asset addresses does the validation of proof of reserve feed value ≥ total supply of the asset

. If any asset passed on the list of inputs will not fulfill that requirement, areAllReservesBacked()

will return a false value, together with a list of assets not fulfilling the reserve requirements.

This contract is designed to be used by both Aave v2 and v3 Avalanche systems.

- Each Aave v2 and Aave v3 Avalanche pool will have its own associated ProofOfReserveExecutor

smart contract, exposing mainly 2 functions:

1. isEmergencyActionPossible()

. Returning at any time if all the assets with proof of reserve feed associated are properly backed.

1. executeEmergencyAction()

Callable by anybody and allowing to execute the appropriate protective actions on the Aave pool if isEmergencyActionPossible()

is true.

- isEmergencyActionPossible()

. Returning at any time if all the assets with proof of reserve feed associated are properly backed.

- executeEmergencyAction()

Callable by anybody and allowing to execute the appropriate protective actions on the Aave pool if isEmergencyActionPossible()

is true.

- On Aave v2, the emergency action is to disable borrowing of all pool assets if any of them has a problem regarding Proof of Reserve, together with freezing the asset/s with the failure on PoR. This factually stops any additive action on the pool, making it impossible to both open new borrowings and use the “exploited” asset as collateral anyhow (e.g. to replace “healthy” collateral that would get withdrawn).

- On Aave v3, the emergency action is more precise than on v2: freeze the asset and set its LTV to 0, which, different than on v2, factually removes all the “collateral power” from the asset, even on existing positions.

- The ProofOfReserveExecutor

of Aave v3 will have RISK_ADMIN role from the Aave v3 protocol, allowing this way to halt freeze assets when the defined conditions are met.

- To allow the ProofOfReserveExecutor

of Aave v2 to freeze assets and disable borrowing, as the permissions system on Aave v2 is less granular than in v3, we have added a new role PROOF_OF_RESERVE_ADMIN on the v2 addresses provider smart contract and updated the pool configurator contract to allow both the pool admin (previously) and the new proof of reserve admin (the ProofOfReserveExecutor of v2) to freeze and enable/disable borrowing.

- The addition/removal of assets with proof of reserve will be controlled via the standard Aave governance procedures. Everything else (monitoring if all reserves are backed, and executing the emergency action if not) is completely permissionless and algorithmically defined.

- The ProofOfReserveExecutor

contracts will also be compatible and integrated with Chainlink Automation, adding more assurances on execution timing.

[

proof-of-reserve_diagram (1)

1920×1513 84.1 KB

](https://europe1.discourse-cdn.com/business20/uploads/aave/original/2X/d/d0aded42f7127e76d3f1c1f9fdb4eb9ba74f8ab4.jpeg)

High-level flow of the system

Emergency actions

One of the main complexities of this development is: once we know there is a problem with the backing of an asset, which is

the best way of protecting Aave, given the limitations of the protocol itself?

The answer is not really simple for a protocol like Aave, as no solution can be perfect. In addition, Aave v3 has a better mechanism than v2 regarding risk control, so the action needs to be adapted per version.

Aave v2: disable borrowing on all assets and freeze the affected ones

Disabling all assets' borrowing and freezing the one/s exploited protects against the following:

- An asset is exploited on the origination chain, and all deposited funds are extracted, so the backing becomes factually 0. Other assets (e.g. non-bridged) are not exploited. Disable borrowing of all assets on the usage chain protects in good part all the "healthy" assets, as no value can be extracted by borrowing, while freezing the exploited ones protects against existing positions depositing it, in order to withdraw "healthy" collaterals present on their positions already.

It is worth it to say that this mechanism can't fully protect against existing positions with both "healthy" and "bad" assets as collateral pre-exploit, as they would be able to withdraw the "healthy" collateral and leave the "bad" one. To protect against this, an LTV mechanism like the one used on v3 is necessary.

Aave v3: freeze the asset and set the LTV to 0

As mentioned before, Aave v3 has better levers in terms of risk control than Aave v2. One of them is the mechanism of setting the LTV of collateral to 0, which allows for:

- The asset still keeps counting in value in what regards Health Factor, as this only considers the liquidation threshold.
- The protocol introduces a mechanism of withdrawal priority, based on LTV equal/not to 0: if an asset is used as collateral and has LTV == 0, the user always needs to withdraw first this asset (or any other with LTV == 0 on his position), before withdrawing other collaterals with LTV not equal to 0.

This way, when setting the LTV of an asset to 0 via the Aave PoR system, factually the "collateral" power of it will count only for liquidations, but not for new borrowings or exploits based on withdrawals of "healthy" assets.

Alongside setting LTV to 0, freezing of the compromised assets will prevent more complex cases when existing positions can deposit the affected asset in order to withdraw another LTV-zero, but somehow healthier. In addition, it doesn't seem logical to allow any type of supply of an asset with clear problems.

Other solutions considered and why they don't sound like good ideas are:

- Discounting the "unbacked" amount post-exploit from the bridged asset price.

The reality is that when an exploit of this nature happens, all or the big majority of the assets are usually extracted. This strategy would mean putting the price of the bridged assets to 0 or close, which 1) will create immediate bad debt of the protocol, and chaotic behavior, if there is an incentive or not to liquidate assets, a really complex scenario for any kind of recovery 2,) no chance of recovery or incredibly complex, given the aforementioned chaotic involvement of numerous parties in liquidations.

- Pausing the protocol (stopping all possible actions, including liquidations)

. This is the last resort mechanism of the protocol, even stopping liquidations. Usually, it should be avoided at all costs if there are alternatives because not enabling liquidations means the potential creation of bad debt depending on market price fluctuations. At the same time, it creates a situation where users potentially not affected by the exploit can't withdraw their funds, refill their position, repay their borrowing, etc.

We think there is room to work on the direction of some of those solutions in certain scenarios, but generally, they involve really deep architectural changes on both the Aave protocol, pricing, and bridging infrastructures.

FAQ

What happens after an emergency action is executed?

- Once an emergency action (no matter if v2/v3), it will be up to the Aave governance to take a decision on what to do next.

Given that the scenarios can be quite diverse, it doesn't make sense to try to automate a reaction/re-activation, without first analyzing the specifics.

Is it possible to have "false positives" on

areAllReservesBacked()

?

- It is theoretically possible, but at the moment Chainlink updates show exactly a 0% historic rate of false positives while running the system in production for 4 months.

A “false positive” in this model could happen if, for example, a user would try to bridge 100 AAVE from Ethereum to Avalanche, and the minting event on the AAVE.e on the Avalanche network would happen before the AAVE.e PoR feed updates with +100 AAVE included.

In practice, the monitoring system of Chainlink + submission of the transaction on the Avalanche side (update of PoR) is always faster than the “bridging time” of the asset, so it is a pretty remote case.

In the case where tokens get bridged from Avalanche to Ethereum, it is not possible to have “false positives”, as the burning of for example AAVE.e happens first, so the condition of proof of reserve feed value ≥ total supply gets even “stronger”.

Does this new system affect existing ones?

- Generally, the system is non-invasive, built on top of Aave v2 and v3, so nothing should be affected. From the perspective of any existing platform integrating Aave, an emergency action would be as if the Aave governance would decide to disable borrowing/freeze/set LTV to 0 on one or multiple assets of the pool.

Does this system add new procedures to the ecosystem?

- Yes, mainly 2:

- Connection with Chainlink Automation (or any bot) to monitor and trigger emergency actions, if applicable, as described in the documentation of the codebase.

- On listings of new assets, if a PoR is available, it should be registered into the corresponding ProofOfReserveExecutor

associated with the pool, and into the ProofOfReserveAggregator

(if not registered already for a different pool on the same network).

- Connection with Chainlink Automation (or any bot) to monitor and trigger emergency actions, if applicable, as described in the documentation of the codebase.

- On listings of new assets, if a PoR is available, it should be registered into the corresponding ProofOfReserveExecutor

associated with the pool, and into the ProofOfReserveAggregator

(if not registered already for a different pool on the same network).

So now bridged assets are completely safe on Aave?

- NO

. Bridged assets, same as other asset types, have their own risks, both technical and economical. The community should still evaluate them carefully before listing them because Aave <> PoR only adds some extra protection, not 100% protection.

In addition, the protocol design has inherent limitations (more on v2) due to which is not straightforward to fully protect the system without big UX implications.

Next steps

Currently, the audits of the system are getting finished by both Certora and SigmaPrime. Once that happens, we will create a Snapshot vote to activate PoR, followed (if approved) by the factual deployment of the additional smart contract components.