Security Review Report
NM-0054: Aave Starknet Bridge

NETHERMIND

(Sep 12, 2022)

Contents

1 Executive Summary

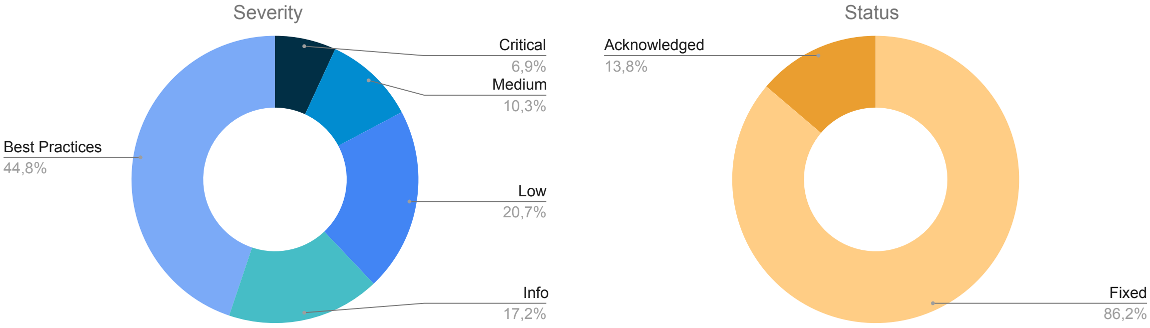
This document presents the security review performed by Nethermind on the AAVE Protocol - L2 contracts written in Cairo Language. The project is composed of 4 contracts, 2 libraries and 2 interfaces and together are used to allow movement of `staticATokens` between the Ethereum L1 and the Starknet L2 while continuing to accrue rewards. Rewards on the Starknet L2 are accrued through `rewAAVE` tokens which are exchanged for rewards on Ethereum L1 via AAVE `bridge` contracts which sit on both layers and allow communication between them. **As an overall assessment of the initial audit**, we note that the project can benefit from better technical documentation detailing a) use cases, b) functional requirements, c) non-functional requirements, d) diagrams showing how interactions take place, and e) a more extensive test suite. The audit was carried out using manual inspection of the code base. The instructions for running the test suite are not accurate, and the output changed for each auditor. Most of the reported issues are related to missing validations, access control, and handling `initialize` contracts. **During the re-audit, 25 (out of 29) issues were fixed and 4 issues have been indicated as `false positive` by the client** since these actions are controlled by the Aave Governance. The audit team has decided to consider these 4 issues as `acknowledged` because the code must be defensive. As requested by the client, **during the re-audit stage, the audit team made a careful second pass on the math supporting the protocol and no errors were detected**. Finally, the client reports that they have added additional changes in four commits that are not related to the findings. These commits are listed below and have not been reviewed in the scope of this audit.

- **Upgrade Cairo to version 0.9:** `494a6891941b039421e02eefe0f71c2e0923c4c8`

- **Insert in-line comments for L2:** `a0df8f39c14182992c475f7ed187ea68cba333c6`

- **Update the README.md:** 3cff841e91ba1d9e42e2d8e80d62870c2b0e115b

- **Add parameter to event:** `43d4dfb334dde15d9e581c04a17c8380504dc400`



(a) (b)

Distribution of Issues: **Critical** (1), **High** (1), **Medium** (3), **Low** (6), **Undetermined** (0), **Informational** (5), **Best Practices** (13).
Summary of the Status: **Fixed** (25), **Acknowledged** (4).

Summary of the Audit

Audit Type	Security Review
Initial Report	Jun 22, 2022
Response from Client	Aug 9, 2022
Final Report	Sep 12, 2022
Methods	Manual Review, Creation of Test Cases
Repository	https://github.com/aave-starknet-project/aave-starknet-bridge/tree/main/contracts/l2
Commit Hash (Initial Audit)	017e84df7cc886ab7d3ae6f2a5df826fcc23de95
Commit Hash (Re-audit)	8302ab87abc0a3cd992e6606543a70bc76caa351
Documentation Assessment	Medium

2 Contracts

	Contract	Lines of Code	Lines of Comments	Comments Ratio	Blank Lines	Total Lines
1	contracts/l2/bridge.cairo	288	24	8.3%	72	384
2	contracts/l2/interfaces/IERC20.cairo	27	0	0.0%	12	39
3	contracts/l2/interfaces/Istatic_a_token.cairo	49	0	0.0%	19	68
4	contracts/l2/lib/proxy.cairo	101	15	14.9%	21	137
5	contracts/l2/lib/wad_ray_math.cairo	215	5	2.3%	51	271
6	contracts/l2/test/wad_ray_tests.cairo	196	19	9.7%	85	300
7	contracts/l2/tokens/rewAAVE.cairo	125	8	6.4%	20	153
8	contracts/l2/tokens/static_a_token.cairo	206	6	2.9%	30	242
9	contracts/l2/tokens/incentivized_erc20.cairo	173	5	2.9%	24	202
	Total	**1380**	**82**	**5.9%**	**334**	**1796**

3 Summary of Findings

	Finding	Severity	Update
1	Contract can be initialized multiple times	Critical	Fixed
2	Contract can be initialized multiple times	Critical	Fixed
3	Missing safety features in bridge.set_reward_token(...).	Medium	Acknowledged
4	Missing safety features in proxy.change_proxy_admin(...)	Medium	Acknowledged
5	Missing safety features in proxy.upgrade_implementation(...)	Medium	Acknowledged
6	Missing Ethereum address bounds check in bridge.bridge_rewards(...)	Low	Fixed
7	Missing input validation in incentivized_erc20_set_l2_bridge(...)	Low	Fixed
8	Missing input validation in proxy.change_proxy_admin(...)	Low	Fixed
9	Missing input validation in proxy.initialize_proxy(...)	Low	Fixed
10	Missing input validation in rewAAVE.initialize_rewAAVE(...)	Low	Fixed
11	Possible invalid state when tokens are set without ownership	Low	Fixed
12	Inconsistent L1 to L2 message input validation	Info	Acknowledged
13	Missing event emissions	Info	Fixed
14	Missing input validation in proxy.constructor(...)	Info	Fixed
15	Storage variable l1_bridge cannot be changed once assigned.	Info	Fixed
16	Unnecessary initialization logic in proxy.initialize_proxy(...)	Info	Fixed
17	Comment does not accurately reflect logic	Best Practices	Fixed
18	Constant name spelling error	Best Practices	Fixed
19	Constant with zero value	Best Practices	Fixed
20	Getter functions with @external visibility	Best Practices	Fixed
21	Inconsistent variable naming across functions	Best Practices	Fixed
22	Incorrect function ordering	Best Practices	Fixed
23	Misleading function name bridge.only_l1_handler(...)	Best Practices	Fixed
24	Missing use of the Cairo boolean library	Best Practices	Fixed
25	Remove unused imports	Best Practices	Fixed
26	Typo in comment	Best Practices	Fixed
27	Unnecessary alloc_locals in bridge.set_reward_token(...)	Best Practices	Fixed
28	Unused variable naming across functions	Best Practices	Fixed
29	Use of unofficial OpenZeppelin contract package	Best Practices	Fixed

4 Risk Rating Methodology

The risk rating methodology used by Nethermind follows the principles established by the OWASP Foundation. The severity of each finding is determined two factors: **Likelihood** and **Impact**.

Likelihood is a measure of how likely the finding is to be uncovered and exploited by an attacker. This factor will be one of the following values:

a) **High**: The issue is trivial to exploit and has no specific conditions that need to be met;

b) **Medium**: The issue is moderately complex and may have some conditions that need to be met;

c) **Low**: The issue is very complex and requires very specific conditions to be met.

When defining the likelihood of a finding other factors are also considered. These can include but are not limited to: Motive, opportunity, exploit accessibility, ease of discovery and ease of exploit.

Impact is a measure of the damage that may be caused if the finding were to be exploited by an attacker. This factor will be one of the following values:

a) **High**: The issue can cause significant damage such as loss of funds or the protocol entering an unrecoverable state;

b) **Medium**: The issue can cause moderate damage such as impacts that only affect a small group of users or only a particular part of the protocol;

c) **Low**: The issue can cause little to no damage such as bugs that are easily recoverable or cause unexpected interactions that cause minor inconveniences.

When defining the impact of a finding other factors are also considered. These can include but are not limited: Data/state integrity, loss of availability, financial loss, reputation damage. After defining the likelihood and impact of an issue, the severity can be determined according to the table below.

		Severity Risk		
Impact	**High**	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Info/Best Practices	Low	Medium
	Undetermined	Undetermined	Undetermined	Undetermined
		Low	**Medium**	**High**
		Likelihood		

To address issues that do not fit a High/Medium/Low severity, Nethermind also uses three more finding severities: **Informational**, **Best Practices**, and **Undetermined**.

a) **Informational** findings do not pose any risk to the application, but they carry some information that the audit team intends to formally pass to the client;

b) **Best Practice** findings are used when some piece of code does not conform with smart contract development best practices;

c) **Undetermined** findings are used when we cannot predict the impact or likelihood of the issue.

5 Findings

5.1 General findings

5.1.1 [Info] Missing event emissions

File(s): `contracts/l2/*`

Description: Contracts are missing event emissions for important changes to state, such as owner/admin address, token address changes and initialization. A list of functions with missing event emissions is shown below:

Recommendation: Consider emitting events in the functions presented above.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `59e280f6ab536aefd746fa66299ca38a1223a43b`

5.1.2 [Best Practices] Getter functions with `@external` visibility

File(s): `contracts/l2/*`

Description: Contracts contain functions that have a visibility decorator of `@external` even though they do not make any changes to state. There are several "getter" functions that have this issue.

Recommendation: Review the decorator of all functions in all contracts. Functions that do not change the state such as "getters" can have their visibility descriptor set to `@view`.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `02d838cc6b3de79db9399c749d52bc8995dd34fc`.

5.1.3 [Best Practices] Missing use of the Cairo `boolean` library

File(s): `contracts/l2/*`

Description: Contracts use `felt` values `1` and `0` to represent boolean values rather than constants `TRUE` and `FALSE` from the Cairo boolean library. Some examples of felt values being used as booleans are shown below:

Recommendation: To improve overall code readability and maintainability it is recommended to use the Cairo boolean library rather than direct `1` and `0` `felt` values.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `ffb3ed4a3681fa29d2b497862e06537ff29fcffb`.

5.1.4 [Best Practices] Remove unused imports

File(s): `contracts/l2/*`

Description: Contracts contain unused imports which affect the readability and maintainability of the code. A non-exhaustive list of the unused imports is shown below:

Recommendation: Check all contract imports and remove imports that are unused to improve the overall readability and maintainability of the code.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `b06d4f314c6ac718d3e4108f11dc1b8f711c4182`

5.1.5 [Best Practices] Use of unofficial OpenZeppelin contract package

File(s): N/A

Description: The OpenZeppelin Cairo library contracts are retrieved from the npm package `@joriksch/oz-cairo` which does not appear to be an official OpenZeppelin package. The official OpenZeppelin Cairo library contracts are found here.

Recommendation: Consider using the official OpenZeppelin Cairo contract library.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `80827082de76308005d657369a2d09b840b965e9`.

5.2 `contracts/l2/lib/proxy.cairo`

5.2.1 [Medium] Missing safety features in `proxy.change_proxy_admin(...)`

File(s): `contracts/l2/lib/proxy.cairo`

Description: There are no safety features in place in case an incorrect or malicious address is set in `proxy.change_proxy_admin(...)`. This will allow unexpected behavior to occur as soon as the address is changed. The function is shown below:

Recommendation: Consider introducing safety features to mitigate a wrong address being set as the proxy admin. Some examples of safety features for address changes are *set then claim*, *set then confirm* and *timelock*. Consider checking `new_admin` for zero values.

Status: Acknowledged

Update from the client: This is not an error but a design choice. Aave governance process handles proxy admin's change. Commit Hash with the fix of the issue: `55ac73f0223176c45f09485508820a9f3bbe0a4b`.

Update from Nethermind: We understand that this function is protected, only callable by the admin. However, as the code shows above, any value can be inserted as the proxy admin, including `address(0x0)`. So, we consider this issue as acknowledged by the client.

5.2.2 [Medium] Missing safety features in `proxy.upgrade_implementation(...)`

File(s): `contracts/l2/lib/proxy.cairo`

Description: There are no safety features in place should an incorrect or malicious address be set in `proxy.upgrade_implementation(...)`. This will allow unexpected behavior to occur as soon as the address is changed. The function is shown below:

Recommendation: Consider introducing safety features to mitigate a wrong address being set as the implementation. Some examples of safety features for address changes are *set then confirm* and *timelock*.

Status: Acknowledged

Update from the client: This is not an error but a design choice. Aave governance process handles implementation's change. Commit Hash with the fix of the issue: `3d345380e011f892243728ecdeb3ad13aedddeca`.

Update from Nethermind: We understand that this function is protected, only callable by the admin. However, as the code shows above, any measures have been taken to protect this code from a wrong input. So, we consider this issue as acknowledged by the client.

5.2.3 [Low] Missing input validation in `proxy.change_proxy_admin(...)`

File(s): `contracts/l2/lib/proxy.cairo`

Description: The function `proxy.change_proxy_admin(...)` does not check the argument `new_admin` for zero values. If the proxy admin were to be set to zero it will not be possible to recover the contract. The function is shown below:

Recommendation: Check the argument `new_admin` for zero values.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `55ac73f0223176c45f09485508820a9f3bbe0a4b`.

5.2.4 [Low] Missing input validation in `proxy.initialize_proxy(...)`

File(s): `contracts/l2/lib/proxy.cairo`

Description: The function `proxy.initialize_proxy(...)` does not check the argument `implementation_address` for zero values. The function is shown below:

Recommendation: Check the argument `implementation_address` for zero values.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `9fa8d7c784c36c9586b75dc7f7e27521363d6a04`.

5.2.5 [Info] Missing input validation in `proxy.constructor(...)`

File(s): `contracts/l2/lib/proxy.cairo`

Description: The constructor does not check the argument `proxy_admin` for zero values. The constructor is shown below:

Recommendation: Check the argument `proxy_admin` for zero values.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `cded7b6e3af94e6ab441a03e79cb3ae771931414`.

5.2.6 [Info] Unnecessary initialization logic in `proxy.initialize_proxy(...)`

File(s): `contracts/l2/lib/proxy.cairo`

Description: A proxy will typically keep track of whether it has been initialized to prevent calls to the `__default__` or `__l1_default__` before the proxy has been properly initialized. Given that calls to `__default__` and `__l1_default__` do not check the storage variable `Proxy_initialized`, the initialization logic is not necessary.

Recommendation: Consider replacing the functions `proxy.initialize_proxy(...)` and `proxy.upgrade_implementation` (which both set the implementation) with one function which has no initialization logic.

Status: Fixed

Update from the client: We replaced both initialize_proxy and upgrade_implementation by set_implementation. Commit Hash with the fix of the issue: `fb0ca2c5aad2d1ea0cc67cec7898378c0b73b030`.

5.3 `contracts/l2/lib/wad_ray_math.cairo`

5.3.1 [Best Practices] Comment does not accurately reflect logic

File(s): `contracts/l2/lib/wad_ray_math.cairo`

Description: There are two comments on L29 and L34 that state the return of their respective functions is of type `Ray`, where the functions actually return type `Wad`. A code snippet is shown below:

Recommendation: Ensure there is consistency between function logic and their respective comments.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `ad0b4634116e1e4c778845aa8600b55dfe58245e`.

5.3.2 [Best Practices] Constant name spelling error

File(s): `contracts/l2/lib/wad_ray_math.cairo`

Description: The constant `HALF_WAD_RAY_RATION` is spelled incorrectly, it should be `HALF_WAD_RAY_RATIO`. This spelling error occurs on L30 and L53.

Recommendation: Change `HALF_WAD_RAY_RATION` to `HALF_WAD_RAY_RATIO` on L30 and L53.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `00846ab1e92c5a0360cc1032dd0f21226c05b90c`.

5.3.3 [Best Practices] Inconsistent variable naming across functions

File(s): `contracts/l2/lib/wad_ray_math.cairo`

Description: The functions `wad_mul(...)`, `wad_div(...)`, `ray_mul(...)` and `ray_div(...)` refer to the quotient returned by `uint256_unsigned_div_rem(...)` with either the variable name `quo` or `quotient`.

Recommendation: To improve the overall readability of the codebase it is recommended only to use one consistent variable name across all functions that use `uint256_unsigned_div_rem(...)`.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `21c02030b302f6204204edca60525bf98e5cf41c`.

5.3.4 [Best Practices] Unused variable naming across functions

File(s): `contracts/l2/lib/wad_ray_math.cairo`

Description: The functions `wad_mul(...)` and `ray_mul(...)` both call `uint256_unsigned_vid_rem(...)` and store the remainder in a variable `rem` even though it is not used.

Recommendation: To improve code readability and maintainability it is recommended to not declare any variables that will not be used. Consider substituting `rem` with `_` for unused return values.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `f0806bc6f12dab8832615a38b547f0c6732a0e5b`.

5.4 `contracts/l2/tokens/incentivized_erc20.cairo`

5.4.1 [Low] Missing input validation in `incentivized_erc20_set_l2_bridge(...)`

File(s): `contracts/l2/tokens/incentivized_erc20.cairo`

Description: The function `incentivized_erc20.incentivized_erc20_set_l2_bridge(...)` does not check the argument `l2_bridge_` for zero values. If `0` is passed into this function, `incentivized_erc20.incentivized_erc20_only_bridge` will fail its assertion which prevents minting, burning and pushing the rewards index. The function is shown below:

Recommendation: Check the argument `l2_bridge_` for zero values.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `b737713256feadb3caee5674678ca61a81f6f635`.

5.5 `contracts/l2/tokens/rewAAVE.cairo`

5.5.1 [Critical] Contract can be initialized multiple times

File(s): `contracts/l2/tokens/rewAAVE.cairo`

Description: The function `rewAAVE.initialize_rewAAVE(...)` does not check if the contract is already initialized. Anybody can call the initialize function at any time allowing the token name, symbol, decimals and owner to change as well as minting `amount` tokens to `recipient`. These newly minted `rewAAVE` tokens can then be claimed at `contracts/l2/bridge.cairo` for `AAVE` tokens on L1. The function is shown below:

Recommendation: Ensure that `rewAAVE.initialize_rewAAVE(...)` can only be called once.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `0b6aa6b3636b06a1790e615e76a946c6717425f9`.

5.5.2 [Low] Missing input validation in `rewAAVE.initialize_rewAAVE(...)`

File(s): `contracts/l2/tokens/rewAAVE.cairo`

Description: In the function `rewAAVE.initialize_rewAAVE(...)` the arguments `name`, `symbol`, `decimals`, `initial_supply` ,`recipient` and `owner` can be set to zero. If the initialize function is called where `owner` is zero, the owner cannot be changed. The function is shown below:

Recommendation: For the arguments `name`, `symbol`, `decimals`, `initial_supply` and `recipient` consider adding checks for zero values. For the argument `owner` add a check for zero values.

Status: Fixed

Update from the client: We allow initial_supply to be zero. Commit Hash with the fix of the issue: `964903789567e23ddd881a58a974e792c109e6f5`.

5.6 `contracts/l2/tokens/static_a_token.cairo`

5.6.1 [Critical] Contract can be initialized multiple times

File(s): `contracts/l2/tokens/static_a_token.cairo`

Description: The function `static_a_token.initialize_static_a_token(...)` checks if the contract has already been initialized by checking that the storage variables `ERC20_name_`, `ERC20_symbol_` and `ERC20_decimals_` are zero. If this function is called with the arguments `name`, `symbol` and `decimals` set to zero, the next time the initialize function is called the zero checks will pass. A malicious account can call this initialize function with zero values before the "official" initialize call by the Aave team. This allows the malicious account to mint any number of tokens (argument `initial_supply`) to any account of their choosing (argument `recipient`) and then exchange these tokens for their respective L1 token. When the official Aave initialize call executes with non-zero values for `name`, `symbol` and `decimals` the contract will properly initialize and this issue can no longer be exploited. The function is shown below:

Recommendation: Ensure that the function `static_a_token.initialize_static_a_token(...)` can only be called once. Check the arguments `name`, `symbol` and `decimals` for zero.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `6ba4633de872a640745c7dc2d1f6aa7508ad2995`.

5.6.2 [Best Practices] Incorrect function ordering

File(s): `contracts/l2/tokens/static_a_token.cairo`

Description: The function `static_a_token.get_user_claimable_rewards(...)` has the function decorator `@view`. However, according to the in-line documentation it is placed in the section for `@external` functions. The function is shown below:

Recommendation: Move the function to the "getters" section of the contract.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `6c389851eff52ed82ea6f4f33c7a76ef328d8292`.

5.7 `contracts/l2/bridge.cairo`

5.7.1 [Medium] Missing safety features in `bridge.set_reward_token(...)`.

File(s): `contracts/l2/bridge.cairo`

Description: There are no safety features in place should an incorrect or malicious address be set to the reward token with `bridge.set_reward_token(...)`. This will allow unexpected behavior to occur as soon as the address is changed. Because reward tokens can be claimed for L1 tokens through the bridge, any unexpected behavior related to the total supply of reward tokens (such as minting) could potentially lead to token being drained on the L1 bridge. The function is shown below:

Recommendation: Consider introducing safety features to mitigate a wrong address being set as the implementation. Some examples of safety features for address changes are *set then claim*, *set then confirm* and *timelock*. Consider checking the argument `reward_token` for zero values.

Status: Acknowledged

Update from the client: This is not an error, but a design choice. Aave governance process handles reward token's change. Commit Hash with the fix of the issue: `c68cb40f75a29652bf9381f8338f651505e57080`.

Update from Nethermind: We agree that the function is protected, so it is very likely that this finding will not represent an issue. However, the code must protect itself from external modules. Thus, we are marking this as `acknowledged`.

5.7.2 [Low] Missing Ethereum address bounds check in `bridge.bridge_rewards(...)`

File(s): `contracts/l2/bridge.cairo`

Description: The function `bridge.bridge_rewards(...)` communicates with L1 however the argument `l1_recipient` is not checked that it is within the bounds of `0` to `ETH_ADDRESS_BOUND`. Unaware users may lose funds by passing an invalid address into this function. The function is shown below:

Recommendation: Ensure that the argument `l1_recipient` is a valid Ethereum address.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `be6ad9193381d4935193fefb20fd917fc604d1e6`.

5.7.3 [Low] Possible invalid state when tokens are set without ownership

File(s): `contracts/l2/bridge.cairo`

Description: It is possible for the reward token and any `static_a_token` to be set in `bridge.cairo` without the bridge being the owner of the token. When the reward token is set without bridge ownership rewards cannot be bridged, deposits cannot be handled and rewards cannot be minted. When a `static_a_token` is set without bridge ownership withdrawals cannot be initiated, deposits cannot be handled, rewards cannot be minted and index updates will fail.

Recommendation: In reality, the ownership will most likely be transferred by the owner. However `bridge.set_reward_token(...)` and `bridge.approve_bridge(...)` should still assert that it has ownership over the token before it is used by the protocol as a safeguard storage to ensure the `bridge` contract does not enter an invalid state.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `35904e15b2e1ca53e897b2704a33761635167ed0`.

5.7.4 [Info] Inconsistent `L1` to `L2` message input validation

File(s): `contracts/l2/bridge.cairo`

Description: It is not possible for the L1 bridge to deposit a given token where its associated L2 token is zero. The contract `bridge.cairo` also does this zero address check. The L2 bridge should operate correctly, independently, and prevent incorrect messaging input regardless of the L1 bridge's implementation. However the same level of input sanitisation is not upheld for the other message parameters. For instance, block number `0` is not possible, yet there is no check for this in the `@l1_handler` function `bridge.handle_deposit(...)`.

Recommendation: Consistency with input sanitation between L1 and L2 contracts should be practiced.

Status: Acknowledged

Update from the client: Comments from the client: Adding too many checks increases costs for users who are using the bridge with valid parameters.

Update from Nethermind: The communication between the L1 and L2 layers must contain all the parameters and validations necessary for the requests to be matched and executed without any possibility of ambiguity. Otherwise, we may have unpredictable behaviors.

5.7.5 [Info] Storage variable `l1_bridge` cannot be changed once assigned.

File(s): `contracts/l2/bridge.cairo`

Description: The storage variable `l1_bridge` cannot be changed after being assigned. This is beneficial in terms of security, but it also imposes a hard limitation in case the L1 bridge needs to be changed in the future. The function is shown below:

Recommendation: This is a trade-off between security and flexibility, the development team must discuss the best approach.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `28fe1e1e787344a39fee514cde60fd36bb4b1e6e`.

5.7.6 [Best Practices] Constant with zero value

File(s): `contracts/l2/bridge.cairo`

Description: The contract defines the constant `WITHDRAW_MESSAGE` set to zero. Since all storage and memory values are initially zero, this can potentially lead to unexpected behavior.

```
const WITHDRAW_MESSAGE = 0
```

Recommendation: Consider using a non-zero value for the constant `WITHDRAW_MESSAGE`.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `1e4034ecd1cea12e68dc4bbfc82a00b51547403c`.

5.7.7 [Best Practices] Misleading function name `bridge.only_l1_handler(...)`

File(s): `contracts/l2/bridge.cairo`

Description: The function `only_l1_handler(...)` checks if the sender is the L1 bridge. The function should therefore be renamed to `only_l1_bridge(...)`.

Recommendation: Consider updating the function name.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `0e98552f1b54b69f950b6d4b7b42ca12da4978fa`.

5.7.8 [Best Practices] Typo in comment

File(s): `contracts/l2/bridge.cairo`

Description: There is a typo in the comment on L311.

Recommendation: Change "recieve" to "receive" on L311.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `3883f5e6745f534cc70bf0851508e3ac9bb617d5`.

5.7.9 [Best Practices] Unnecessary `alloc_locals` in `bridge.set_reward_token(...)`

File(s): `contracts/l2/bridge.cairo`

Description: The statement `alloc_locals` in the function `bridge.set_reward_token(...)` is not necessary. The function is shown below:

Recommendation: Consider removing `alloc_locals`.

Status: Fixed

Update from the client: Commit Hash with the fix of the issue: `d5f25510e112457b7d337b1b92610095274ceda1`.

6 Documentation Evaluation

Documenting the code is adding enough information to explain what it does so that it is easy to understand the purpose and the underlying functionality of each file/function/line. Documentation can come not only in the form of a read-me but through comments, websites and even videos. Besides being a good programming practice, providing proper documentation improves the efficiency of audits. Less time can be spent understanding the protocol and more time can be put towards auditing, improving the efficiency and overall output of the audit.

We recommend the following improvements be made to the documentation of this project:

- Inline comments to describe each function (what it does, inputs, outputs);
- Inline comments for lines of code that could benefit from an added description for further context;
- Improve the architecture description of the system;
- Create user facing documentation;
- Formalize the functional requirements of the project, and the actions performed by each contract.

The technical documentation explaining the bridging system between L1 and L2 could be improved by taking inspiration from the original AAVE governance proposal and first Starknet update, where they explain the overall concept in a way that easy to understand.

We strongly recommend creating a complete documentation suite in order to maximize the output of external audits that will be performed in this code base. This will reduce the audit cost, and improve the audit efficacy. One of the most important goals of an audit is to match code against specification, which becomes impossible without a complete and well organized set of technical documents. Thus, despite the issues presented in this report, we highlight that the most important take away from this internal audit is the need to create high quality documentation able to present the quality of this project to the Web3 community.

7 Test Suite Evaluation

During the first audit, the tests fail to run when using the provided commit hash. The environment variables used to setup the Ethereum L1 test chain cannot be accessed through the script `testnet:l1` in `package.json`. Changes had to be made to the `testnet:l1` script in order to have tests run (the new `testnet:l1` script is provided below). Once these changes were made, the output of the tests were inconsistent where different team members would have a different number of failing tests. **During the re-audit, the tests were executed using a docker image. The output is presented below**.

7.1 Test output

```
yarn run v1.22.19
$ hardhat --network l1_testnet test
Starknet plugin using environment at .venv
Using network l2_testnet at http://testnet-l2:5050
Downloading compiler 0.8.10
Downloading compiler 0.6.12
Warning: This declaration shadows an existing declaration.
  --> @aave/core-v3/contracts/dependencies/openzeppelin/contracts/ERC20.sol:57:15:
   |
57 |   constructor(string memory name, string memory symbol) {
   |               ^^^^^^^^^^^^^^^^^^
Note: The shadowed declaration is here:
  --> @aave/core-v3/contracts/dependencies/openzeppelin/contracts/ERC20.sol:66:3:
   |
66 |   function name() public view returns (string memory) {
   |   ^ (Relevant source part starts here and spans across multiple lines).
Warning: This declaration shadows an existing declaration.
  --> @aave/core-v3/contracts/dependencies/openzeppelin/contracts/ERC20.sol:57:35:
   |
57 |   constructor(string memory name, string memory symbol) {
   |                                   ^^^^^^^^^^^^^^^^^^^^
Note: The shadowed declaration is here:
  --> @aave/core-v3/contracts/dependencies/openzeppelin/contracts/ERC20.sol:74:3:
   |
74 |   function symbol() public view returns (string memory) {
   |   ^ (Relevant source part starts here and spans across multiple lines).
Warning: This declaration shadows an existing declaration.
  --> @aave/core-v3/contracts/protocol/tokenization/base/IncentivizedERC20.sol:74:5:
   |
74 |     string memory name,
   |     ^^^^^^^^^^^^^^^^^^
Note: The shadowed declaration is here:
  --> @aave/core-v3/contracts/protocol/tokenization/base/IncentivizedERC20.sol:86:3:
   |
86 |   function name() public view override returns (string memory) {
   |   ^ (Relevant source part starts here and spans across multiple lines).
Warning: This declaration has the same name as another declaration.
  --> @aave/core-v3/contracts/protocol/tokenization/base/IncentivizedERC20.sol:75:5:
   |
75 |     string memory symbol,
   |     ^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
  --> @aave/core-v3/contracts/protocol/tokenization/base/IncentivizedERC20.sol:91:3:
   |
91 |   function symbol() external view override returns (string memory) {
   |   ^ (Relevant source part starts here and spans across multiple lines).
Warning: This declaration has the same name as another declaration.
  --> @aave/core-v3/contracts/protocol/tokenization/base/IncentivizedERC20.sol:76:5:
   |
76 |     uint8 decimals
   |     ^^^^^^^^^^^^^^
Note: The other declaration is here:
  --> @aave/core-v3/contracts/protocol/tokenization/base/IncentivizedERC20.sol:96:3:
   |
96 |   function decimals() external view override returns (uint8) {
   |   ^ (Relevant source part starts here and spans across multiple lines).
Warning: This contract has a payable fallback function, but no receive ether function. Consider adding a receive ether
↪  function.
  --> @aave/core-v3/contracts/dependencies/openzeppelin/upgradeability/BaseAdminUpgradeabilityProxy.sol:14:1:
   |
14 | contract BaseAdminUpgradeabilityProxy is BaseUpgradeabilityProxy {
   | ^ (Relevant source part starts here and spans across multiple lines).
Note: The payable fallback function is defined here.
  --> @aave/core-v3/contracts/dependencies/openzeppelin/upgradeability/Proxy.sol:17:3:
   |
17 |   fallback() external payable {
   |   ^ (Relevant source part starts here and spans across multiple lines).
Warning: This contract has a payable fallback function, but no receive ether function. Consider adding a receive ether
↪  function.
  --> @aave/core-v3/contracts/dependencies/openzeppelin/upgradeability/InitializableUpgradeabilityProxy.sol:11:1:
   |
11 | contract InitializableUpgradeabilityProxy is BaseUpgradeabilityProxy {
   | ^ (Relevant source part starts here and spans across multiple lines).
Note: The payable fallback function is defined here.
  --> @aave/core-v3/contracts/dependencies/openzeppelin/upgradeability/Proxy.sol:17:3:
   |
17 |   fallback() external payable {
   |   ^ (Relevant source part starts here and spans across multiple lines).
Warning: This contract has a payable fallback function, but no receive ether function. Consider adding a receive ether
↪  function.
  --> @aave/core-v3/contracts/dependencies/openzeppelin/upgradeability/InitializableAdminUpgradeabilityProxy.sol:12:1:
   |
12 | contract InitializableAdminUpgradeabilityProxy is
   | ^ (Relevant source part starts here and spans across multiple lines).
Note: The payable fallback function is defined here.
  --> @aave/core-v3/contracts/dependencies/openzeppelin/upgradeability/Proxy.sol:17:3:
   |
17 |   fallback() external payable {
   |   ^ (Relevant source part starts here and spans across multiple lines).
Warning: Function state mutability can be restricted to pure
  --> contracts/l1/mocks/IncentivesControllerMock.sol:14:5:
   |
14 |     function getAssetData(address)
   |     ^ (Relevant source part starts here and spans across multiple lines).
Warning: Function state mutability can be restricted to pure
  --> contracts/l1/mocks/IncentivesControllerMock.sol:27:5:
   |
27 |     function assets(address)
   |     ^ (Relevant source part starts here and spans across multiple lines).
Warning: Function state mutability can be restricted to pure
  --> contracts/l1/mocks/IncentivesControllerMock.sol:42:5:
   |
42 |     function getClaimer(address) external view override returns (address) {
   |     ^ (Relevant source part starts here and spans across multiple lines).
Warning: Function state mutability can be restricted to pure
  --> contracts/l1/mocks/IncentivesControllerMock.sol:57:5:
   |
57 |     function getRewardsBalance(address[] calldata, address)
   |     ^ (Relevant source part starts here and spans across multiple lines).
Warning: Function state mutability can be restricted to pure
  --> contracts/l1/mocks/IncentivesControllerMock.sol:66:5:
   |
66 |     function claimRewards(
   |     ^ (Relevant source part starts here and spans across multiple lines).
Warning: Function state mutability can be restricted to pure
  --> contracts/l1/mocks/IncentivesControllerMock.sol:74:5:
   |
74 |     function claimRewardsOnBehalf(
   |     ^ (Relevant source part starts here and spans across multiple lines).
Warning: Function state mutability can be restricted to pure
  --> contracts/l1/mocks/IncentivesControllerMock.sol:83:5:
   |
83 |     function getUserUnclaimedRewards(address)
   |     ^ (Relevant source part starts here and spans across multiple lines).
```

```
Warning: Function state mutability can be restricted to pure
  --> contracts/l1/mocks/IncentivesControllerMock.sol:92:5:
   |
92 |     function getUserAssetData(address, address)
   |     ^ (Relevant source part starts here and spans across multiple lines).
Warning: Function state mutability can be restricted to pure
   --> contracts/l1/mocks/IncentivesControllerMock.sol:101:5:
    |
101 |     function PRECISION() external view override returns (uint8) {
    |     ^ (Relevant source part starts here and spans across multiple lines).
Warning: Function state mutability can be restricted to pure
   --> contracts/l1/mocks/IncentivesControllerMock.sol:105:5:
    |
105 |     function DISTRIBUTION_END() external view override returns (uint256) {
    |     ^ (Relevant source part starts here and spans across multiple lines).
Warning: Function state mutability can be restricted to pure
   --> contracts/l1/mocks/IncentivesControllerMock.sol:109:5:
    |
109 |     function EMISSION_MANAGER() external view returns (address) {
    |     ^ (Relevant source part starts here and spans across multiple lines).
Compiled 76 Solidity files successfully
  Bridge
Duplicate definition of RewardsClaimed (RewardsClaimed(address,address,uint256),
↪   RewardsClaimed(address,address,address,uint256))
     dai and usdc whales convert their tokens to aTokens (28078ms)
     set L2 implementation contracts (19932ms)
     initialize L2 static_a_tokens (19914ms)
     set L1 token bridge as implementation contract (6762ms)
     initialize the bridge on L1 and L2 (27258ms)
Duplicate definition of RewardsClaimed (RewardsClaimed(address,address,uint256),
↪   RewardsClaimed(address,address,address,uint256))
     set a distribution end in the future (109ms)
     deposit Dai and Usdc (15379ms)
     deposit aDai and aUsdc (19643ms)
     jump into the future
     withdraw aDai and aUsdc to L1 user (39818ms)
     withdraw Dai and Usdc (14866ms)
     reverts withdrawal when wrong l2rewards index is provided (6325ms)
     L2 user sends back reward accrued to L1 user (23075ms)
  Proxy
     Verify that owner is the proxy admin (1421ms)
     initialize token A through proxy (14302ms)
Invoke transaction 0x2d1d31a0fbebe0328bb301fdc7a4135f8cfec001baf9578f0ea8e89d4b50ff0 is REJECTED.
Transaction rejected. Error message:
/home/projects/cairo-contracts/env/lib/python3.8/site-packages/starkware/starknet/common/syscalls.cairo:52:5: Error at
↪   pc=0:91:
Got an exception while executing a hint.
Cairo traceback (most recent call last):
src/openzeppelin/account/Account.cairo:94:6: (pc=0:719)
src/openzeppelin/account/Account.cairo:106:36: (pc=0:663)
/home/projects/cairo-contracts/src/openzeppelin/account/library.cairo:189:30: (pc=0:389)
/home/projects/cairo-contracts/src/openzeppelin/account/library.cairo:208:19: (pc=0:411)
Error in the called contract (0x1b6b00c12bee0b5c60cd15f2165b4def4f897706e4720d535d43511ede8d321):
Error message: Proxy: caller is not admin
/aave-starknet-bridge/contracts/l2/dependencies/openzeppelin/upgrades/library.cairo:65:13: Error at pc=0:182:
An ASSERT_EQ instruction failed: 1180552335781213199188962152961691779244409946440537413350305133978804979588 !=
↪   20596759951375991457633380310041163031717389243715232810820868874136108825.
            assert admin = caller
            ^******************^
Cairo traceback (most recent call last):
/aave-starknet-bridge/contracts/l2/lib/proxy.cairo:41:6: (pc=0:370)
func set_implementation{syscall_ptr : felt*, pedersen_ptr : HashBuiltin*, range_check_ptr}(
     ^****************^
```

```
/aave-starknet-bridge/contracts/l2/lib/proxy.cairo:47:5: (pc=0:349)
    Proxy.assert_only_admin()
    ^***********************^
     disallows random user to upgrade (4528ms)
     allows owner (proxy_admin) to change proxy_admin to random user (6020ms)
     allows owner to upgrade (6067ms)
  static_a_token
     Deploy accounts (24684ms)
     Deploy token and reward token (59814ms)
     allows owner to set l2 token bridge (4643ms)
Invoke transaction 0x2d2a059ef407f19ae488551487c4e1bcab25fe75d259b70a103d2ff136aab8f is REJECTED.
Transaction rejected. Error message:
/home/projects/cairo-contracts/env/lib/python3.8/site-packages/starkware/starknet/common/syscalls.cairo:52:5: Error at
↪  pc=0:91:
Got an exception while executing a hint.
Cairo traceback (most recent call last):
src/openzeppelin/account/Account.cairo:94:6: (pc=0:719)
src/openzeppelin/account/Account.cairo:106:36: (pc=0:663)
/home/projects/cairo-contracts/src/openzeppelin/account/library.cairo:189:30: (pc=0:389)
/home/projects/cairo-contracts/src/openzeppelin/account/library.cairo:208:19: (pc=0:411)
Error in the called contract (0x442b167c7efd105de92f4b3fd20ca2f2bec614522e8f3bf68d8c6bbc5844765):
Error message: Ownable: caller is not the owner
/aave-starknet-bridge/contracts/l2/dependencies/openzeppelin/access/ownable.cairo:46:13: Error at pc=0:1342:
An ASSERT_EQ instruction failed: 1777315646660512548367789871145689583254361065018702317818863813590971680412 !=
↪  2144281840703669607578820663997386699993676463909798375948849229948128629884.
            assert owner = caller
            ^*******************^
Cairo traceback (most recent call last):
/aave-starknet-bridge/contracts/l2/tokens/static_a_token.cairo:37:6: (pc=0:2106)
func set_l2_bridge{syscall_ptr : felt*, pedersen_ptr : HashBuiltin*, range_check_ptr}(
     ^***********^
/aave-starknet-bridge/contracts/l2/tokens/static_a_token.cairo:41:5: (pc=0:2084)
    Ownable.assert_only_owner()
    ^*************************^
     disallows non-owner to set l2 token bridge (4596ms)
     allows bridge to mint (6245ms)
     allows bridge to burn (6587ms)
Invoke transaction 0x2a65d8c26fe6e38a62b1f547acfc7e7e4dac1a4919464d2767132142232aa5d is REJECTED.
Transaction rejected. Error message:
/home/projects/cairo-contracts/env/lib/python3.8/site-packages/starkware/starknet/common/syscalls.cairo:52:5: Error at
↪  pc=0:91:
Got an exception while executing a hint.
Cairo traceback (most recent call last):
src/openzeppelin/account/Account.cairo:94:6: (pc=0:719)
src/openzeppelin/account/Account.cairo:106:36: (pc=0:663)
/home/projects/cairo-contracts/src/openzeppelin/account/library.cairo:189:30: (pc=0:389)
/home/projects/cairo-contracts/src/openzeppelin/account/library.cairo:208:19: (pc=0:411)
Error in the called contract (0x442b167c7efd105de92f4b3fd20ca2f2bec614522e8f3bf68d8c6bbc5844765):
Error message: Caller address should be bridge: {l2_bridge_} (Cannot evaluate ap-based or complex references:
↪  ['l2_bridge_'])
/aave-starknet-bridge/contracts/l2/tokens/incentivized_erc20.cairo:244:9: Error at pc=0:2001:
An ASSERT_EQ instruction failed: 2144281840703669607578820663997386699993676463909798375948849229948128629884 !=
↪  1372230931195349345258302106925467725186894800524563684099591909164756476672.
        assert caller_address = l2_bridge_
        ^********************************^
Cairo traceback (most recent call last):
/aave-starknet-bridge/contracts/l2/tokens/static_a_token.cairo:284:6: (pc=0:2843)
func mint{syscall_ptr : felt*, pedersen_ptr : HashBuiltin*, range_check_ptr}(
     ^**^
/aave-starknet-bridge/contracts/l2/tokens/static_a_token.cairo:287:5: (pc=0:2821)
    incentivized_erc20_only_bridge()
    ^******************************^
     disallows non-bridge to mint (4564ms)
Invoke transaction 0x42e9727ec699ad3e26e12221a2d66ff9236f27bb793d3b7c1a6347e4495ecb is REJECTED.
Transaction rejected. Error message:
/home/projects/cairo-contracts/env/lib/python3.8/site-packages/starkware/starknet/common/syscalls.cairo:52:5: Error at
↪  pc=0:91:
Got an exception while executing a hint.
Cairo traceback (most recent call last):
src/openzeppelin/account/Account.cairo:94:6: (pc=0:719)
```

```
src/openzeppelin/account/Account.cairo:106:36: (pc=0:663)
/home/projects/cairo-contracts/src/openzeppelin/account/library.cairo:189:30: (pc=0:389)
/home/projects/cairo-contracts/src/openzeppelin/account/library.cairo:208:19: (pc=0:411)
Error in the called contract (0x442b167c7efd105de92f4b3fd20ca2f2bec614522e8f3bf68d8c6bbc5844765):
Error message: Caller address should be bridge: {l2_bridge_} (Cannot evaluate ap-based or complex references:
↪  ['l2_bridge_'])
/aave-starknet-bridge/contracts/l2/tokens/incentivized_erc20.cairo:244:9: Error at pc=0:2001:
An ASSERT_EQ instruction failed: 2144281840703669607578820663997386699993676463909798375948849229948128629884 !=
↪  1372230931195349345258302106925467725186894800524563684099591909164756476672.
        assert caller_address = l2_bridge_
        ^*********************************^
Cairo traceback (most recent call last):
/aave-starknet-bridge/contracts/l2/tokens/static_a_token.cairo:297:6: (pc=0:2879)
func burn{syscall_ptr : felt*, pedersen_ptr : HashBuiltin*, range_check_ptr}(
     ^**^
/aave-starknet-bridge/contracts/l2/tokens/static_a_token.cairo:300:5: (pc=0:2857)
    incentivized_erc20_only_bridge()
    ^******************************^
     disallows non-bridge to burn (4504ms)
     allows bridge to update rewards index (6107ms)
Invoke transaction 0x45ba7d3a314e0f8c9e7b11d249e7a8a096e06f10500ce91a7e275f9e880fd04 is REJECTED.
Transaction rejected. Error message:
/home/projects/cairo-contracts/env/lib/python3.8/site-packages/starkware/starknet/common/syscalls.cairo:52:5: Error at
↪  pc=0:91:
Got an exception while executing a hint.
Cairo traceback (most recent call last):
src/openzeppelin/account/Account.cairo:94:6: (pc=0:719)
src/openzeppelin/account/Account.cairo:106:36: (pc=0:663)
/home/projects/cairo-contracts/src/openzeppelin/account/library.cairo:189:30: (pc=0:389)
/home/projects/cairo-contracts/src/openzeppelin/account/library.cairo:208:19: (pc=0:411)
Error in the called contract (0x442b167c7efd105de92f4b3fd20ca2f2bec614522e8f3bf68d8c6bbc5844765):
Error message: Caller address should be bridge: {l2_bridge_} (Cannot evaluate ap-based or complex references:
↪  ['l2_bridge_'])
/aave-starknet-bridge/contracts/l2/tokens/incentivized_erc20.cairo:244:9: Error at pc=0:2001:
An ASSERT_EQ instruction failed: 2144281840703669607578820663997386699993676463909798375948849229948128629884 !=
↪  1372230931195349345258302106925467725186894800524563684099591909164756476672.
        assert caller_address = l2_bridge_
        ^*********************************^
Cairo traceback (most recent call last):
/aave-starknet-bridge/contracts/l2/tokens/static_a_token.cairo:324:6: (pc=0:2967)
func push_rewards_index{syscall_ptr : felt*, pedersen_ptr : HashBuiltin*, range_check_ptr}(
     ^****************^
/aave-starknet-bridge/contracts/l2/tokens/static_a_token.cairo:327:5: (pc=0:2943)
    incentivized_erc20_push_rewards_index(block_number, rewards_index)
    ^****************************************************************^
/aave-starknet-bridge/contracts/l2/tokens/incentivized_erc20.cairo:161:5: (pc=0:1902)
    incentivized_erc20_only_bridge()
    ^******************************^
     disallows random account from updating rewards index (4555ms)
     only allows increases in rewards index (4876ms)
     rejects old block numbers (4659ms)
     returns correct user pending rewards before claim (2087ms)
     mints no rewards tokens if caller has no claimable rewards (12388ms)
     claims rewards and mints correct amount of rewards tokens to caller (12463ms)
     returns correct pending rewards after claim (1764ms)
     keeps track of each user rewards index correctly (2824ms)
     mints rewards correctly to different user (23659ms)
     Rewards of user are not lost if L2 tokens are burnt before claiming (18807ms)
  wad_ray_math
     multiplies wads (1845ms)
     reverts on wad multiplication overflows (1428ms)
     divides wads (1738ms)
     reverts on wad divide by zero (1462ms)
     reverts on wad divide overflows (1534ms)
     multiplies rays (2337ms)
     reverts on ray multiplication overflows (1470ms)
     divides rays (1730ms)
     reverts on ray divide by zero (1437ms)
     reverts on ray divide overflows (1490ms)
     multiplies_no_rounding rays (1615ms)
     reverts on ray multiplication_no_rounding overflows (1400ms)
     converts rays to wads (1434ms)
     converts wads to rays (1457ms)
     no_rounding converts rays to wads (1551ms)
  52 passing (12m)
Done in 709.07s.
```

8 About Nethermind

Founded in 2017 by a small team of world-class technologists, Nethermind builds Ethereum solutions for developers and enterprises. Boosted by a grant from the Ethereum Foundation in August 2018, our team has worked tirelessly to deliver the fastest Ethereum client in the market. Our flagship Ethereum client is all about performance and flexibility. Built on .NET core, a widespread, enterprise-friendly platform, Nethermind makes integration with existing infrastructures simple, without losing sight of stability, reliability, data integrity, and security

Nethermind is made up of several engineering teams across various disciplines, all collaborating to realize the Ethereum roadmap, by conducting research and building high-quality tools. Teams focus on specific areas of the Ethereum problem space. Each consists of specialists and experienced developers working alongside interns, learning the ropes in the Nethermind Internship Program.

Our mission is to gather passionate talent from around the world, and to tackle some of the blockchain's most complex problems. Nethermind provides software solutions and services for developers and enterprises building the Ethereum ecosystem. We offer security reviews to projects built on EVM compatible chains and StarkNet. We have expertise in multiple areas of the Ethereum ecosystem, including protocol design, smart contracts (written in Solidity and Cairo), MEV, etc. We develop some of the most used tools on Starknet and one of the most used Ethereum clients. Learn more about us at `https://nethermind.io`.

Disclaimer

This report is based on the scope of materials and documentation provided by you to Nethermind in order that Nethermind could conduct the security review outlined in **1. Executive Summary** and **2. Audited Files**. The results set out in this report may not be complete nor inclusive of all vulnerabilities. Nethermind has provided the review and this report on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. This report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on this report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, Nethermind disclaims any liability in connection with this report, its content, and any related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Nethermind does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and Nethermind will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.