

Hardening Blockchain Security with Formal Methods

FOR

0xMeowProtocol



► Prepared For:

Meow Protocol

https://twitter.com/0xMeowProtocol/

► Prepared By:

Benjamin Sepanski Andreea Buţerchi

► Contact Us: contact@veridise.com

▶ Version History:

Dec. 04, 2023 Initial Draft

© 2023 Veridise Inc. All Rights Reserved.

Contents

| Co | Contents | | | | |
|---------------------|---|--|--|--|--|
| 1 Executive Summary | | | | | |
| 2 | Project Dashboard | | | | |
| 3 | 3.1 Audit Goals and Scope 3.1 Audit Goals | | | | |
| 4 | Vulnerability Report 1.1 Detailed Description of Issues 4.1.1 V-MEW-VUL-001: Interest never accumulated 4.1.2 V-MEW-VUL-002: Anyone can set fee receivers 4.1.3 V-MEW-VUL-003: Removed Initializer interface 4.1.4 V-MEW-VUL-004: Missing call to refreshConfigInternal() 4.1.5 V-MEW-VUL-005: No call to _disableInitializers 4.1.6 V-MEW-VUL-006: Variable can be immutable 4.1.7 V-MEW-VUL-007: Removed zero-cost abstraction | 7 8 8 9 10 11 12 13 14 | | | |
| 5 | Methodology Detailed Description of Fuzzed Specifications V-MEW-SPEC-001: ERC20.01: transfer should revert if a user attempts to send more funds than they have V-MEW-SPEC-002: ERC20.02: Funds should be successfully transferred from sender to to as long as sender ≠ to V-MEW-SPEC-003: ERC20.03: transfer should not modify totalSupply, allowances, or balances other than sender and to V-MEW-SPEC-004: ERC20.04: transferFrom should enforce allowance and user balance V-MEW-SPEC-005: ERC20.06: transferFrom should not modify totalSupply, other allowances, or balances V-MEW-SPEC-006: ERC20.07: approve makes appropriate state changes V-MEW-SPEC-007: ERC20.08: increase Allowances makes appropriate state changes V-MEW-SPEC-008: ERC20.09: decrease Allowance makes appropriate state changes V-MEW-SPEC-009: Token Distributor: funds can only be sent to the original recipients | 15 15 15 16 16 17 18 19 20 21 22 23 24 | | | |
| Gl | ssary | 25 | | | |

From Nov. 27, 2023 to Nov. 29, 2023, Meow Protocol engaged Veridise to review the security of their project, 0xMeowProtocol. 0xMeowProtocol is an over-collateralized lending platform forked from AAVE's V1 protocol*. Veridise conducted the assessment over 6 person-days, with 2 engineers reviewing code over 3 days on commit 0xe8217ac. The auditing strategy involved a tool-assisted analysis of the source code performed by Veridise engineers as well as extensive manual auditing.

Code assessment. The 0xMeowProtocol developers provided the source code of the 0xMeowProtocol contracts for review. The review focused on two smart contracts in 0xMeowProtocol. First, a token distribution contract with a designated set of receivers, and second, a token contract representing liquidity deposited into the larger lending protocol. The audit also validated that the changes made to the remainder of the codebase did not lead to behavioral differences in the protocol. For a security analysis of the core logic of this protocol, we refer the reader to various publicly available audits of the AAVE V1 protocol[†].

No documentation was provided, and most of the internal documentation was removed from the original codebase. Veridise auditors used documentation from AAVE as an adequate substitute. The source code did not include a test suite. However, several files in the source code indicate that the developers use linting tools such as prettier.

Two behavioral changes from the AAVE V1 repository were made:

- ▶ *Removal of flashloan functionality:* Meow Protocol developers removed all functions, events, and variables related to the lending pools providing flashloans.
- ▶ Removal of versioned upgradeability: Meow Protocol developers replaced versioned upgradeability with standard OpenZeppelin Initializable contracts, which do not contain a queryable version.

Otherwise, the functional behavior of the protocol remains untouched. The Veridise auditors refer to prior audit reports publicly available on AAVE's website[†] for a full analysis of protocol security.

The two contracts which Veridise manually reviewed retained their primary functionality from the original protocol. The main changes consist of removing support for interest redirection, upgrading to modern Solidity standards, and removal of the various unused fields.

Summary of issues detected. The audit uncovered 7 issues, 2 of which are assessed to be of high or critical severity by the Veridise auditors. Specifically, failure to accrue interest to liquidity providers (V-MEW-VUL-001) and lack of access control allowing anyone to withdraw the protocol fees (V-MEW-VUL-002). The Veridise auditors also identified 3 warnings and 2 informational findings, largely related to best practices for initializer routines.

Veridise Audit Report: 0xMeowProtocol

^{*} https://github.com/aave/aave-protocol/tree/4b4545fb5

[†]https://docs.aave.com/developers/v/1.0/developing-on-aave/security-and-audits

Recommendations. After auditing the protocol, the auditors had a few suggestions to improve the 0xMeowProtocol. First, the auditors strongly recommend adapting the entire AAVE test suite into the development process for this repository, relying on standard tools like CI/CD to prevent breaking changes. Second, the auditors suggest returning the comments from AAVE to the codebase. This will improve readability, and help future developers navigate the codebase more easily without needing to refer to AAVE.

Disclaimer. We hope that this report is informative 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 system is secure in all dimensions. In no event shall Veridise 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.

Table 2.1: Application Summary.

| Name | Version | Type | Platform |
|----------------|-----------|----------|----------|
| 0xMeowProtocol | 0xe8217ac | Solidity | Polygon |

Table 2.2: Engagement Summary.

| Dates | Method | Consultants Engaged | Level of Effort |
|-------------------------|----------------|---------------------|-----------------|
| Nov. 27 - Nov. 29, 2023 | Manual & Tools | 2 | 6 person-days |

Table 2.3: Vulnerability Summary.

| Name | Number | Resolved |
|-------------------------------|--------|----------|
| Critical-Severity Issues | 2 | 0 |
| High-Severity Issues | 0 | 0 |
| Medium-Severity Issues | 0 | 0 |
| Low-Severity Issues | 0 | 0 |
| Warning-Severity Issues | 3 | 0 |
| Informational-Severity Issues | 2 | 0 |
| TOTAL | 7 | 0 |

Table 2.4: Category Breakdown.

| Name | Number |
|------------------|--------|
| Logic Error | 2 |
| Access Control | 2 |
| Maintainability | 2 |
| Gas Optimization | 1 |

3.1 Audit Goals

The engagement was scoped to provide a security assessment of 0xMeowProtocol's smart contracts. In our audit, we sought to answer questions such as:

- ▶ Do all functions have proper access control?
- ► Are all rewards distributed correctly?
- ► Are the contracts vulnerable to any common Solidity security issues (reentrancy, large stakeholder attacks, etc.)?
- ▶ Does the upgrade to Solidity 0.8 lead to any breaking changes?
- ► Are any known vulnerabilities present in V1 of AAVE?
- ► Are all key AAVE invariants maintained?
- ▶ Is any core AAVE functionality disrupted?

3.2 Audit Methodology & Scope

Audit Methodology. To address the questions above, our audit involved a combination of human experts and automated program analysis & testing tools. In particular, we conducted our audit with the aid of the following techniques:

- ➤ Static analysis. To identify potential common vulnerabilities, we leveraged our custom smart contract analysis tool Vanguard. These tools are designed to find instances of common smart contract vulnerabilities, such as reentrancy, flash loans, and uninitialized variables.
- ► Fuzzing/Property-based Testing. We also leverage fuzz testing to determine if the protocol may deviate from the expected behavior. To do this, we formalize the desired behavior of the protocol as [V] specifications and then use our fuzzing framework OrCa to determine if a violation of the specification can be found. See Chapter 5 for more details.

Scope. The scope of this audit is limited to the fees/TokenDistirbutor.sol contract and the tokenization/BToken.sol contract. The remaining files were excluded from a detailed manual review. Veridise auditors programmatically compared the excluded files to their equivalents from the AAVE repository, and manually reviewed any differences to check that the functionality was unchanged. During the audit, the Veridise auditors referred to these excluded files, but assumed that they have otherwise been implemented correctly.

Methodology. Veridise auditors reviewed the reports of previous audits for 0xMeowProtocol, and read the AAVE documentation. They then began a manual audit of the code assisted by both static analyzers and automated testing.

3.3 Classification of Vulnerabilities

When Veridise auditors discover a possible security vulnerability, they must estimate its severity by weighing its potential impact against the likelihood that a problem will arise. Table 3.1 shows how our auditors weigh this information to estimate the severity of a given issue.

Table 3.1: Severity Breakdown.



In this case, we judge the likelihood of a vulnerability as follows in Table 3.2:

Table 3.2: Likelihood Breakdown

| Not Likely | Not Likely A small set of users must make a specific mistake | | |
|-------------|--|--|--|
| | Requires a complex series of steps by almost any user(s) | | |
| Likely | - OR - | | |
| • | Requires a small set of users to perform an action | | |
| Very Likely | Can be easily performed by almost anyone | | |

In addition, we judge the impact of a vulnerability as follows in Table 3.3:

Table 3.3: Impact Breakdown

| Somewhat Bad | hat Bad Inconveniences a small number of users and can be fixed by the user | |
|---|---|--|
| | Affects a large number of people and can be fixed by the user | |
| Bad | - OR - | |
| | Affects a very small number of people and requires aid to fix | |
| Affects a large number of people and requires aid to fix | | |
| Very Bad | - OR - | |
| | Disrupts the intended behavior of the protocol for a small group of | |
| | users through no fault of their own | |
| Protocol Breaking Disrupts the intended behavior of the protocol for a large group of | | |
| users through no fault of their own | | |
| | - | |

In this section, we describe the vulnerabilities found during our audit. For each issue found, we log the type of the issue, its severity, location in the code base, and its current status (i.e., acknowledged, fixed, etc.). Table 4.1 summarizes the issues discovered:

Table 4.1: Summary of Discovered Vulnerabilities.

| ID | Description | Severity | Status |
|---------------|---|----------|--------|
| V-MEW-VUL-001 | Interest never accumulated | Critical | Open |
| V-MEW-VUL-002 | Anyone can set fee receivers | Critical | Open |
| V-MEW-VUL-003 | Removed Initializer interface | Warning | Open |
| V-MEW-VUL-004 | Missing call to refreshConfigInternal() | Warning | Open |
| V-MEW-VUL-005 | No call to _disableInitializers | Warning | Open |
| V-MEW-VUL-006 | Variable can be immutable | Info | Open |
| V-MEW-VUL-007 | Removed zero-cost abstraction | Info | Open |

4.1 Detailed Description of Issues

4.1.1 V-MEW-VUL-001: Interest never accumulated

| Severity | Critical | | Commit | e8217ac |
|------------------|-----------------------------|--|----------------|---------|
| Type | Logic Error | | Status | Open |
| File(s) | contracts/tokenization/MTok | | ion/MToken.sol | |
| Location(s) | cumulateBalanceInternal() | | eInternal() | |
| Confirmed Fix At | | | | |

The cumulateBalanceInternal() function relies on the overriden implementation of balanceOf() to compute the balanceIncrease of a user's account due to interest.

```
uint256 previousPrincipalBalance = super.balanceOf(_user);

//calculate the accrued interest since the last accumulation
uint256 balanceIncrease = balanceOf(_user).sub(previousPrincipalBalance);
```

Snippet 4.1: Snippet from cumulateBalanceInternal()

However, the MToken contract does *not* override balanceOf(). In the above snippet, both function calls refer to the same method, and balanceIncrease will always be zero.

Note that the AAVE V1 implementation does override balanceOf(), as can be seen at this link. This implementation uses the calculateCumulatedBalanceInternal() to accumulate interest. The corresponding function in MToken is unused.

Impact Liquidity providers will receive no benefit from their deposits.

Recommendation Override balanceOf() using the same implementation from AAVE.

4.1.2 V-MEW-VUL-002: Anyone can set fee receivers

| Severity | Critical | Commit | e8217ac |
|------------------|---------------------------|--------|---------|
| Type | Access Control | Status | Open |
| File(s) | fees/TokenDistirbutor.sol | | |
| Location(s) | initialize() | | |
| Confirmed Fix At | | | |

The TokenDistirbutor contract doles out fees to a set of receiver addresses determined at initialization. In the AAVE implementation, this contract is initializable and the receivers are set during initialization.

```
function initialize(
   address[] memory _receivers,
   uint256[] memory _percentages

public initializer {
   internalSetTokenDistribution(_receivers, _percentages);
   emit DistributionUpdated(_receivers, _percentages);
}
```

Snippet 4.2: Definition of initialize in AAVE's implementation of TokenDistributor.sol

Note the use of the initializer modifier, which prevents the method from being called twice.

The developers of 0xMeowProtocol have removed the upgradeability feature, and also the modifier. However, the initialize function is still public.

```
function initialize(address[] memory _receivers, uint256[] memory _percentages)
public {
  internalSetTokenDistribution(_receivers, _percentages);// percentage express in x
  /10000
```

Snippet 4.3: Implementation of initialize in 0xMeowProtocol's implementation.

Impact Once a large amount of fees have accumulated, anyone may set themselves as the receiver and take the funds (e.g. by calling distributeWithAmount). If the contract is not monitored, this may go unnoticed and allow theft for some time.

Recommendation Keep the contract as upgradeable and return the initialize modifier to its former place. This will allow upgrading the receivers (and possibly the entire token distribution mechanism) in a permissioned way.

4.1.3 V-MEW-VUL-003: Removed Initializer interface

| Severity | Warning | Commit | e8217ac |
|------------------|--|--------|---------|
| Type | Maintainability | Status | Open |
| File(s) | configuration/LendingPoolParametersProvider.sol, | | |
| | fees/FeeProvider.sol | | |
| Location(s) | N/A | | |
| Confirmed Fix At | | | |

The LendingPoolParametersProvider and FeeProvider contracts inherit from the VersionedInitializable contract in the AAVE V1 implementation. See, for example, the below snippet.

1 contract LendingPoolParametersProvider is VersionedInitializable {

Snippet 4.4: Snippet from configuration/LendingPoolParametersProvider.sol in the AAVE V1 repository.

The initialize() function is still present in FeeProvider, but unguarded. The initialize() function in LendingPoolParametersProvider is no longer present.

Impact While the current implementations will not directly cause errors, it may not be clear to future developers that these contracts can be upgraded. This may lead to misuse or incorrect modifications in the future.

Recommendation Make the two contracts Initializable with a guarded initialization function.

4.1.4 V-MEW-VUL-004: Missing call to refreshConfigInternal()

| Severity | Warning | Commit | e8217ac |
|------------------|---------------------------------|--------|---------|
| Type | Logic Error | Status | Open |
| File(s) | lendingpool/LendingPoolCore.sol | | |
| Location(s) | initialize() | | |
| Confirmed Fix At | | | |

The LendingPoolCore.initialize() function in the AAVE protocol repository includes a call to refreshConfigInternal().

```
function initialize(LendingPoolAddressesProvider _addressesProvider) public
   initializer {
   addressesProvider = _addressesProvider;
   refreshConfigInternal();
}
```

Snippet 4.5: Implementation of initialize() in the AAVE protocol repository.

 ${\tt refreshConfigInternal()}\ sets\ the\ {\tt lendingPoolAddress}\ to\ the\ one\ provided\ by\ the\ {\tt addressesProvider}$

```
function refreshConfigInternal() internal {
    lendingPoolAddress = addressesProvider.getLendingPool();
}
```

Snippet 4.6: Implementation of refreshConfigInternal().

In the client's implementation, initialize() does not call refreshConfigInternal().

Impact Not calling this function will leave the contract partially uninitialized when deployed, leading to potential errors until refreshConfiguration() is called by the LendingPoolConfigurator

Recommendation Call refreshConfigInternal() in the initialize() function.

4.1.5 V-MEW-VUL-005: No call to _disableInitializers

| Severity | Warning | Commit | e8217ac |
|------------------|-----------------------|--------|---------|
| Type | Access Control | Status | Open |
| File(s) | See issue description | | |
| Location(s) | See issue description | | |
| Confirmed Fix At | | | |

Several contracts inherit from OpenZeppelin's Initializable contract, but do not call the function _disableInitializers() in their constructor. These contracts include:

- ► LendingPool
- ► LendingPoolLiquidationManager
- ► LendingPoolDataProvider
- ► LendingPoolCore
- ► LendingPoolConfigurator

Impact The implementation contracts may be initialized by malicious actors, leading to possible scams.

Further, deployment scripts may incorrectly initialize the implementations instead of the proxies, leading to errors and the opportunity for third-party actors to initialize the proxies.

Recommendation As recommended by OpenZeppelin, call _disableInitializers() in the constructor of each Initializable contract.

4.1.6 V-MEW-VUL-006: Variable can be immutable

| Severity | Info | Commit | e8217ac |
|------------------|-------------------------|--------|---------|
| Type | Gas Optimization | Status | Open |
| File(s) | tokenization/MToken.sol | | |
| Location(s) | underlyingAssetDecimals | | |
| Confirmed Fix At | | | |

The underlyingAssetDecimals variable is set in the constructor() and never written to again. This variable can be replaced with an immutable one to save on gas.

4.1.7 V-MEW-VUL-007: Removed zero-cost abstraction

| Severity | Info | Commit | e8217ac |
|------------------|--|--------|---------|
| Type | Maintainability | Status | Open |
| File(s) | configuration/LendingPoolAddressesProvider.sol | | |
| Location(s) | N/A | | |
| Confirmed Fix At | | | |

 $The \ developers \ removed \ the \ ILending Pool Addresses Provider. sol \ interface.$

Recommendation Since this interface has no direct cost during deployment, but does provide a useful tool for development on top of the protocol, we recommend restoring the interface and having LendingPoolAddressesProvider inherit from the interface.

5.1 Methodology

Our goal was to fuzz test 0xMeowProtocol to assess its functional correctness (i.e, whether the implementation deviates from the intended behavior).

We used OrCa, Veridise's specification-guided fuzzer, and wrote invariants – logical formulas that should hold after every transaction. We then encoded those invariants as assertions in [V], Veridise's in-house specification language.

We used Foundry to setup the environment and write the deployment scripts. Based on the deployed artifacts, we performed fuzzing campaigns using OrCa in order to find violations for the specifications detailed below.

5.2 Properties Fuzzed

Table 5.1 describes the invariants we fuzz-tested. The second column describes the invariant informally in English, and the third shows the total amount of compute time spent fuzzing this property. The last column notes whether we found a bug when fuzzing the invariant (\boldsymbol{X} indicates no bug was found and \checkmark means fuzzing this invariant revealed a bug).

The Veridise auditors devoted a total of 12 compute-hours to fuzzing this protocol, identifying a total of 1 bug.

See Section 5.3 for more details on the specifications.

Table 5.1: Invariants Fuzzed.

| Specification | Invariant | Minutes Fuzzed | Bugs Found |
|----------------|---|----------------|-------------------|
| V-MEW-SPEC-001 | ERC20.01: transfer should revert if a user atte | 180 | X |
| V-MEW-SPEC-002 | ERC20.02: Funds should be successfully transfer | 180 | X |
| V-MEW-SPEC-003 | ERC20: static totalSupply | 180 | X |
| V-MEW-SPEC-004 | ERC20: allowance/balances | 180 | X |
| V-MEW-SPEC-005 | ERC20: no extra modfications | 180 | X |
| V-MEW-SPEC-006 | ERC20.07: approve makes appropriate state changes | 180 | X |
| V-MEW-SPEC-007 | ERC20: increaseAllowance correctness | 180 | X |
| V-MEW-SPEC-008 | ERC20: decreaseAllowance correctness | 180 | X |
| V-MEW-SPEC-009 | TokenDistributor: funds can only be sent to the | 1 | ✓ |

5.3 Detailed Description of Fuzzed Specifications

5.3.1 V-MEW-SPEC-001: ERC20.01: transfer should revert if a user attempts to send more funds than they have

Minutes Fuzzed 180 Bugs Found 0

Scope MToken.sol

Natural Language transfer should revert if a user attempts to send more funds than they have.

```
vars: MToken t
inv: reverted(t.transfer(to, amt), amt > t.balanceOf(sender))
```

5.3.2 V-MEW-SPEC-002: ERC20.02: Funds should be successfully transferred from sender to to as long as sender ≠ to

Minutes Fuzzed 180 Bugs Found 0

Scope MToken.sol

Natural Language Funds should be successfully transferred from sender to to as long as $sender \neq to$.

```
vars: MToken t
inv: finished(t.transfer(to, amt),
    to != sender |=>
        t.balanceOf(sender) = old(t.balanceOf(sender)) - amt &&
        t.balanceOf(to) = old(t.balanceOf(to)) + amt
)
```

5.3.3 V-MEW-SPEC-003: ERC20.03: transfer should not modify totalSupply, allowances, or balances other than sender and to

Minutes Fuzzed 180 Bugs Found 0

Scope MToken.sol

Natural Language transfer should not modify totalSupply, allowances, or balances other than sender and to.

```
vars: MToken t, address o1, address o2, address o3
inv: finished(t.transfer(to, amt),
    o1 != sender && o1 != to |=>
        t.totalSupply() = old(t.totalSupply()) &&
        t.balanceOf(o1) = old(t.balanceOf(o1)) &&
        t.allowance(o2, o3) = old(t.allowance(o2, o3))
}
```

5.3.4 V-MEW-SPEC-004: ERC20.04: transferFrom should enforce allowance and user balance

Minutes Fuzzed 180 Bugs Found 0

Scope MToken.sol

Natural Language transferFrom should enforce allowance and user balance.

transferFrom should revert when the amount requested is greater than what the spender owns or beyond the recipient's allowance.

```
vars: MToken t
inv: reverted(t.transferFrom(from, to, amt),
amt > t.balanceOf(from) || (from != sender && amt > t.allowance(from, sender))
}
```

5.3.5 V-MEW-SPEC-005: ERC20.06: transferFrom should not modify totalSupply, other allowances, or balances

Minutes Fuzzed 180 Bugs Found 0

Scope ERC20 tokens.

Natural Language transferFrom should not modify totalSupply, other allowances, or balances.

```
vars: MToken t, address o1, address o2, address o3, address o4
inv: finished(t.transferFrom(from, to, amt),
    o1 != from && o1 != to && o2 != sender && o3 != from |=>
        t.balanceOf(o1) = old(t.balanceOf(o1)) &&
        t.allowance(from, o2) = old(t.allowance(from, o2)) &&
        t.allowance(o3, o4) = old(t.allowance(o3, o4)) &&
        t.totalSupply() = old(t.totalSupply())
```

5.3.6 V-MEW-SPEC-006: ERC20.07: approve makes appropriate state changes

Minutes Fuzzed 180 Bugs Found 0

Scope MToken.sol

Natural Language approve makes appropriate state changes.

approve should never finish in a state where the allowance of the spender is not equal to the given amount. totalSupply, other allowances and balances should not be modified.

```
vars: MToken t, address o1, address o2, address o3
inv: finished(t.approve(spender, amt),
    o2 != sender && o1 != spender |=>
        t.allowance(sender, spender) = amt &&
        t.allowance(sender, o1) = old(t.allowance(sender, o1)) &&
        t.allowance(o2, o3) = old(t.allowance(o2, o3)) &&
        t.balanceOf(o3) = old(t.balanceOf(o3)) &&
        t.totalSupply() = old(t.totalSupply())
    )
```

5.3.7 V-MEW-SPEC-007: ERC20.08: increaseAllowances makes appropriate state changes

Minutes Fuzzed 180 Bugs Found 0

Scope MToken.sol

Natural Language increaseAllowances makes appropriate state changes.

The function should increase a user's allowance by the indicated amount. totalSupply, other allowances, and balances should not be modified.

5.3.8 V-MEW-SPEC-008: ERC20.09: decreaseAllowance makes appropriate state changes

Minutes Fuzzed 180 Bugs Found 0

Scope MToken.sol

Natural Language decreaseAllowance makes appropriate state changes.

The function should decrease a user's allowance by the indicated amount. totalSupply, other allowances, and balances should not be modified.

5.3.9 V-MEW-SPEC-009: TokenDistributor: funds can only be sent to the original recipients

Minutes Fuzzed 1 Bugs Found 1

Scope TokenDistributor.sol

Natural Language distribute/ distributeWithAmount/ distributeWithPercentage should only send funds to the original recipients. Hence initialize is public-facing, it can be called by malicious actors to alter the list of original recipients. This is a PoC for the issue reported here [add link to the V-MEW-VUL-002 issue].

Formal Specification The [V] specification below can be can interpreted as follows: it is never the case that we can call initialize twice with two different sets of receivers.

```
vars: TokenDistributor td, address u
spec: []!(finished(td.initialize(r1, p1), foreach(x : r1, x != u)) && X<>finished(td.
    initialize(r2, p2), !foreach(x : r2, x != u)))
```

Another [V] specification that simulates the property above is the following one:

This [V] specification says that it is never the case that after a call to initialize with a specific set of recipients it is possible to distribute the funds to other accounts that are not recipients (their balances should remain the same).



- **AAVE** Aave is an Open Source Protocol to create Non-Custodial Liquidity Markets to earn interest on supplying and borrowing assets. To learn more, visit https://aave.com.1
- **OpenZeppelin** A security company which provides many standard implementations of common contract specifications. See https://www.openzeppelin.com.1
- prettier A code formatting tool, see https://prettier.io/docs/en/integrating-with-linters.
 html to learn more. 1
- smart contract A self-executing contract with the terms directly written into code. Hosted on a blockchain, it automatically enforces and executes the terms of an agreement between buyer and seller. Smart contracts are transparent, tamper-proof, and eliminate the need for intermediaries, making transactions more efficient and secure.. 1, 25
- **Solidity** The standard high-level language used to develop smart contracts on the Ethereum blockchain. See https://docs.soliditylang.org/en/v0.8.19/ to learn more. 1