GoodDollar

$G

Smart Contract Audit Report

August 25, 2022

Introduction 3
About GoodDollar 3
About ImmuneBytes 3
Documentation Details 3
Audit Process & Methodology 4
Audit Details 4
Audit Goals 5
Security Level Reference 5
Version 1: 6
High Severity Issues 6
Medium Severity Issues 6
Low Severity Issues 7
Recommendation / Informational 11
Version 2: 16
High Severity Issues 16
Medium Severity Issues 16
Low Severity Issues 16
Recommendation / Informational 16
Call Graph 18
Automated Audit Result 24
Fuzz Test Result 28
Concluding Remarks 31
Disclaimer 31

Introduction

1. About GoodDollar

GoodDollar is a 100% non-profit foundation looking to secure financial freedom for everyone in the world by launching a digital coin built on the blockchain and based on the principles of universal basic income (UBI). GoodDollar: Changing the Balance, For Good.

About Contract:
GoodDollar project is launching publicly. Its mechanism allows people & organizations to lock funds into an interest-bearing decentralized protocol, currently compound.finance, and donate its created interest towards the Global Basic Income cause. Anyone who proves they are not a bot can claim Global Basic Income.

Visit https://www.gooddollar.org/ to know more about it.

2. About ImmuneBytes

ImmuneBytes is a security start-up that provides professional services in the blockchain space. The team has hands-on experience in conducting smart contract audits, penetration testing, and security consulting. ImmuneBytes's security auditors have worked on various A-league projects and understand DeFi projects like AAVE, Compound, 0x Protocol, Uniswap, and dydx.

The team has been able to secure 175+ blockchain projects by providing security services on different frameworks. ImmuneBytes team helps start-ups with a detailed analysis of the system, ensuring security and managing the overall project.

Visit http://immunebytes.com/ to know more about the services.

Documentation Details

The GoodDollar team has provided the following doc for the purpose of audit:

1. https://www.notion.so/immunebytes/Standard-Contract-Audit-GoodDollar-Staking-9eeb3d3b68354d78844cf3aed308c79e
2. https://www.notion.so/gooddollar/G-Staking-Specifications-f227b09fe3884ef0a7adea20f4e7fc02
3. https://www.notion.so/gooddollar/Reserve-inflation-allocation-Specs-20915e4956834da3bc78a90d1fd762bb

Audit Process & Methodology

ImmuneBytes team has performed thorough testing of the project, starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line-by-line inspection of the Smart Contract to find potential issues like Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.

In the Unit testing phase, we run unit tests written by the developer in order to verify the functions work as intended. In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was audited by a team of independent auditors, including -

1. Structural analysis of the smart contract is checked and verified.
2. An extensive automated testing of all the contracts under scope is conducted.
3. Line-by-line Manual Code review is conducted to evaluate, analyze and identify the potential security risks in the contract.
4. Evaluation of the contract's intended behavior and the documentation shared is an imperative step to verify the contract behaves as expected.
5. For complex and heavy contracts, adequate integration testing is conducted to ensure that contracts perform in an acceptable manner while interacting with each other.
6. Storage layout verifications in the upgradeable contract are a must.
7. An important step in the audit procedure is highlighting and recommending better gas optimization techniques in the contract.

Audit Details

- Project Name: GoodDollar
- Contracts Name: GoodReserveCDAI, DistributionHelper, GoodDollarStaking, StakingRewardsFixedAPY, GoodDollarMintBurnWrapper
- Languages: Solidity(Smart contract), Typescript (Unit Testing)
- Github link: https://github.com/GoodDollar/GoodProtocol
- Branch: gooddollar-v3
- Commit hash: 5f27c8ccf650bd0806006f7e0c88cba153331c2e
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, SmartCheck

Audit Goals

The focus of the audit was to verify that the smart contract system is secure, resilient, and working according to its specifications. The audit activities can be grouped into the following three categories:

1. Security: Identifying security-related issues within each contract and the system of contracts.
2. Sound Architecture: Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
3. Code Correctness and Quality: A full review of the contract source code. The primary areas of focus include
    a. Correctness
    b. Readability
    c. Sections of code with high complexity
    d. Quantity and quality of test coverage

Security Level Reference

Every issue in this report were assigned a severity level from the following:

High severity issues will bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Issues(v1)	High	Medium	Low
Open	-	-	1
Closed	-	1	8

Issues(v2)	High	Medium	Low
Open	-	-	-
Closed	-	-	-

This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore; running a bug bounty program as a complement to this audit is strongly recommended.

Version 1:

High Severity Issues

No issues were found.

Medium Severity Issues

1. **No Pausable functionalities were found in the contract**

   Contract - GoodDollarStaking
   Explanation:
   During the manual review of the contract, it was found that according to the natspec annotations for the **stake** function, the intended behavior is only to allow the function executed when the contract is not paused.

```
94      /**
95       * @dev Allows a staker to deposit Tokens. Notice that `approve` is
96       * needed to be executed before the execution of this method.
97       * Can be executed only when the contract is not paused.
98       * @param _amount The amount of GD to stake
99       * @param _donationRatio percentage between 0-100
100      */
```

   However, as per the current architecture of the contract, no pausable features or modifiers were found in the contract.

   Recommendation:
   If Pause-Unpause features are an intended behavior for the functions, the contract must include relevant functions and modifiers to enable pause-unpause features in certain functions.

   Amended: The team has fixed the issue, and it is no longer present in the code.

Low Severity Issues

1. **Adequate range of _donationRatio is ambiguous**

   Contract - GoodDollarStaking
   Explanation:
   While staking, the caller is supposed to pass a uint value for the _donationRatio argument. As per the documentation, the _donationRatio is supposed to be between 0-100.

   However, as per the current function design of the stake function, it allows users to pass any value below 100, which means users can also pass zero as an argument for _donationRatio.

   Recommendation:
   If the above-mentioned scenario is not intended, then the _stake() function must define a minimum threshold for the _donationRatio value that is passed by the callers to ensure that only non-zero values are passed for the ratio.

   Amended: The team has fixed the issue, and it is no longer present in the code.

2. **External Visibility should be preferred**

   Contract - GoodDollarStaking
   Explanation:
   Functions never called throughout the contract should be marked as **external** visibility instead of **public** visibility.
   This will effectively result in Gas Optimization as well.

   Therefore, the following function must be marked as external within the contract:
   - withdrawRewards(), setMonthlyGOODRewards(uint256), setGdApy(uint128), getRewardsPerBlock(), getStaked(address), getUserPendingReward(address), totalRewardsPerShare(), goodStakerInfo(address)

   Recommendation:
   If the PUBLIC visibility of the above-mentioned functions is not intended, then the EXTERNAL Visibility keyword should be preferred.

   Partially Amended

3. **Absence of input validation**

   Contract - GoodDollarStaking
   Explanation:
   The **constructor** of **GoodDollarStaking** doesn't include adequate input validation for the daysUntilUpgrade state variable.

   It must be noted daysUntilUpgrade variable plays a significant role in deciding the upgrade time of the older GovernanceStaking and should therefore be updated with the valid uint32 values only.

   Recommendation:
   A require statement should be included in such functions to ensure no invalid uint value is passed for the daysUntilUpgrade state variable while deploying the contract.

   Amended: The team has fixed the issue, and it is no longer present in the code.

4. **No Events emitted after imperative State Variable modification**
   Line no: 224

   Contract - GoodDollarMintBurnWrapper
   Explanation:
   Functions that update an imperative arithmetic state variable contract should emit an event after the update.
   The following functions in the contract update crucial state variables, i.e., t**otalMintCap**, **updateFrequency**, etc., but do not emit any event after these modifications:
   - setUpdateFrequency()
   - setTotalMintCap()

   The absence of event emission for important state variables update also makes it difficult to track them off-chain as well.

   Recommendation:
   As per the best practices in smart contract development, an event should be fired after changing crucial arithmetic state variables.

   Amended: The team has fixed the issue, and it is no longer present in the code.

5. **Unused State variable found in contract**
   Line no: 144

   Contract - GoodDollarMintBurnWrapper
   Explanation:
   The contract includes a state variable, i.e., unused_tokenType, with private visibility.

   However, during the audit, it was found that this state variable is not used throughout the contract.

   Recommendation:
   Unused state variables, modifiers, or functions should be removed from the contract as they only add to the increase in deployment cost without providing any significant value.

   Amended: The team has fixed the issue, and it is no longer present in the code.

6. **Absence of Input Validations in _setMinterCaps function**
   Line no: 423-437

   Contract - GoodDollarMintBurnWrapper
   Explanation:
   The _setMinterCaps function allows the admin to update the minter details of an existing or new minter address in the contract.

   However, it was found that no adequate input validations were included to check the imperative uint arguments like globalLimit, perTxLimit, etc. Although the minter details can only be updated by the admin, it's considered a better practice to include necessary input validations or upper/lower thresholds for such uint values within the functions.

   Recommendation:
   Include adequate require statements to ensure that only valid arguments are passed to the functions while adding or updating a minter's detail.

7. **No Events emitted after imperative State Variable modification**
   Line no: 404

   Contract - GoodReserveCDai
   Explanation:
   Functions that update an imperative arithmetic state variable contract should emit an event after the update.

The following function in the contract updates crucial state variables, i.e., **distributionHelper**, **nonUbiBps**, etc., but does not emit any event after these modifications:

- setDistributionHelper()

The absence of event emission for important state variables update also makes it difficult to track them off-chain as well.

Recommendation:
As per the best practices in smart contract development, an event should be fired after changing crucial arithmetic state variables.

Amended: The team has fixed the issue, and it is no longer present in the code.

8. **External Visibility should be preferred**

Contract - GoodReserveCDai
Explanation:
Functions never called throughout the contract should be marked as **external** visibility instead of **public** visibility.
This will effectively result in Gas Optimization as well.

Therefore, the following function must be marked as external within the contract:

- mintRewardFromRR(address,address,uint256)
- currentPriceDAI()
- mintUBI(uint256,uint256,ERC20)
- setDistributionHelper(DistributionHelper,uint32)
- setDistributionHelper(DistributionHelper,uint32)
- setReserveRatioDailyExpansion(uint256,uint256)
- setReserveRatioDailyExpansion(uint256,uint256)
- end()
- recover(ERC20)
- claimGDX(address,uint256,bytes32[])
- pre(address,bytes32,bytes32)
- when()

Recommendation:
If the **PUBLIC** visibility of the above-mentioned functions is not intended, then the **EXTERNAL** Visibility keyword should be preferred.

Amended: The team has fixed the issue, and it is no longer present in the code.

9. **Multiplication is being performed on the result of Division**
   Line no - 298-306

   Contract - StakingRewardsFixedAPY
   Explanation:
   During the automated testing of the **StakingRewardsFixedAPY** contract, it was found that the **_undoReward** function in the contract involves multiplication on the result of a Division.

```
298
299         //the actual amount we undo needs to take into account the user donation ratio.
300         //rewardsPaidAfterDonation = (100% - donation%) * rewardsBeforeDonation
301         //rewadrdsBeforeDonation = rewardsPaidAfterDonation/(100% - donation%)
302         uint256 rewardsBeforeDonation = (100 *
303             PRECISION *
304             _rewardsPaidAfterDonation) /
305             (100 * PRECISION - stakersInfo[_to].avgDonationRatio);
306
```

   Integer Divisions in Solidity might truncate. Moreover, this performing division before multiplication might lead to a loss of precision.

   Recommendation:
   Solidity doesn't encourage arithmetic operations that involve division before multiplication. Therefore the above-mentioned function should be checked and tested once. It should be redesigned if they do not lead to the expected results.

   Amended: The team has fixed the issue, and it is no longer present in the code.

Recommendation / Informational

1. **Coding Style Issues in the Contract**

   Contract - GoodDollarStaking
   Explanation:
   Code readability of a Smart Contract is largely influenced by the Coding Style issues and, in some specific scenarios, may lead to bugs in the future.

```
Parameter StakingRewardsFixedAPY.getPrinciple(address)._account (myFlats/flatStaking.sol#4758) is not in mixedCase
Parameter StakingRewardsFixedAPY.earned(address)._account (myFlats/flatStaking.sol#4790) is not in mixedCase
Parameter GoodDollarStaking.stake(uint256,uint32)._amount (myFlats/flatStaking.sol#5090) is not in mixedCase
Parameter GoodDollarStaking.stake(uint256,uint32)._donationRatio (myFlats/flatStaking.sol#5090) is not in mixedCase
Parameter GoodDollarStaking.onTokenTransfer(address,uint256,bytes)._from (myFlats/flatStaking.sol#5128) is not in mixedCase
Parameter GoodDollarStaking.onTokenTransfer(address,uint256,bytes)._amount (myFlats/flatStaking.sol#5129) is not in mixedCase
Parameter GoodDollarStaking.withdrawStake(uint256)._amount (myFlats/flatStaking.sol#5146) is not in mixedCase
Parameter GoodDollarStaking.setMonthlyGOODRewards(uint256)._monthlyAmount (myFlats/flatStaking.sol#5314) is not in mixedCase
Parameter GoodDollarStaking.setGdApy(uint128)._interestRatePerBlock (myFlats/flatStaking.sol#5323) is not in mixedCase
Parameter GoodDollarStaking.getStaked(address)._user (myFlats/flatStaking.sol#5347) is not in mixedCase
Parameter GoodDollarStaking.getUserPendingReward(address)._user (myFlats/flatStaking.sol#5360) is not in mixedCase
Parameter GoodDollarStaking.goodStakerInfo(address)._user (myFlats/flatStaking.sol#5390) is not in mixedCase
```

   During the automated testing, it was found that the GoodDollarStaking contract had quite a few code-style issues.

Recommendation:
Therefore, it is recommended to fix the issues like naming convention, indentation, and code layout issues in a smart contract.

Amended: The team has fixed the issue, and it is no longer present in the code.

2. **Commented codes must be wiped-out before deployment**

Contract - GoodDollarStaking
Explanation:
The GoodDollarStaking contract includes quite a few commented codes within the contract body.
This badly affects the readability of the code.

Recommendation:
If these instances of code are not required in the current version of the contract, then the commented codes must be removed before deployment.

Amended: The team has fixed the issue, and it is no longer present in the code.

3. **Coding Style Issues in the Contract**

Contract - GoodDollarMintBurnWrapper
Explanation:
Code readability of a Smart Contract is largely influenced by the Coding Style issues and, in some specific scenarios, may lead to bugs in the future.

```
Parameter GoodDollarMintBurnWrapper.initialize(uint256,address,INameService)._totalMintCap (myFlats/flatMintBurn.sol#4367) is not in mixedCase
Parameter GoodDollarMintBurnWrapper.initialize(uint256,address,INameService)._admin (myFlats/flatMintBurn.sol#4368) is not in mixedCase
Parameter GoodDollarMintBurnWrapper.initialize(uint256,address,INameService)._nameService (myFlats/flatMintBurn.sol#4369) is not in mixedCase
Variable GoodDollarMintBurnWrapper.unused_tokenType (myFlats/flatMintBurn.sol#4335) is not in mixedCase
```

During the automated testing, it was found that the GoodDollarMintBurnWrapper contract had quite a few code-style issues.

Recommendation:
Therefore, it is recommended to fix the issues like naming convention, indentation, and code layout issues in a smart contract.

Amended: The team has fixed the issue, and it is no longer present in the code.

4. **Coding Style Issues in the Contract**

Contract - GoodReserveCDai
Explanation:
Code readability of a Smart Contract is largely influenced by the Coding Style issues and, in some specific scenarios, may lead to bugs in the future.

```
Parameter GoodReserveCDai.initialize(INameService,bytes32)._ns (myFlats/GoodReserve.sol#7420) is not in mixedCase
Parameter GoodReserveCDai.initialize(INameService,bytes32)._gdxAirdrop (myFlats/GoodReserve.sol#7420) is not in mixedCase
Parameter GoodReserveCDai.setGDXAirdrop(bytes32)._airdrop (myFlats/GoodReserve.sol#7445) is not in mixedCase
Parameter GoodReserveCDai.buy(uint256,uint256,address)._tokenAmount (myFlats/GoodReserve.sol#7484) is not in mixedCase
Parameter GoodReserveCDai.buy(uint256,uint256,address)._minReturn (myFlats/GoodReserve.sol#7485) is not in mixedCase
Parameter GoodReserveCDai.buy(uint256,uint256,address)._targetAddress (myFlats/GoodReserve.sol#7486) is not in mixedCase
Parameter GoodReserveCDai.mintRewardFromRR(address,address,uint256)._token (myFlats/GoodReserve.sol#7521) is not in mixedCase
Parameter GoodReserveCDai.mintRewardFromRR(address,address,uint256)._to (myFlats/GoodReserve.sol#7522) is not in mixedCase
Parameter GoodReserveCDai.mintRewardFromRR(address,address,uint256)._amount (myFlats/GoodReserve.sol#7523) is not in mixedCase
Parameter GoodReserveCDai.sell(uint256,uint256,address,address)._gdAmount (myFlats/GoodReserve.sol#7542) is not in mixedCase
Parameter GoodReserveCDai.sell(uint256,uint256,address,address)._minReturn (myFlats/GoodReserve.sol#7543) is not in mixedCase
Parameter GoodReserveCDai.sell(uint256,uint256,address,address)._target (myFlats/GoodReserve.sol#7544) is not in mixedCase
Parameter GoodReserveCDai.sell(uint256,uint256,address,address)._seller (myFlats/GoodReserve.sol#7545) is not in mixedCase
Parameter GoodReserveCDai.mintUBI(uint256,uint256,ERC20)._daiToConvert (myFlats/GoodReserve.sol#7652) is not in mixedCase
Parameter GoodReserveCDai.mintUBI(uint256,uint256,ERC20)._startingCDAIBalance (myFlats/GoodReserve.sol#7653) is not in mixedCase
Parameter GoodReserveCDai.mintUBI(uint256,uint256,ERC20)._interestToken (myFlats/GoodReserve.sol#7654) is not in mixedCase
Parameter GoodReserveCDai.setDistributionHelper(DistributionHelper,uint32)._helper (myFlats/GoodReserve.sol#7704) is not in mixedCase
Parameter GoodReserveCDai.setDistributionHelper(DistributionHelper,uint32)._bps (myFlats/GoodReserve.sol#7704) is not in mixedCase
Parameter GoodReserveCDai.setReserveRatioDailyExpansion(uint256,uint256)._nom (myFlats/GoodReserve.sol#7720) is not in mixedCase
Parameter GoodReserveCDai.setReserveRatioDailyExpansion(uint256,uint256)._denom (myFlats/GoodReserve.sol#7720) is not in mixedCase
Parameter GoodReserveCDai.recover(ERC20)._token (myFlats/GoodReserve.sol#7753) is not in mixedCase
Parameter GoodReserveCDai.claimGDX(address,uint256,bytes32[])._user (myFlats/GoodReserve.sol#7770) is not in mixedCase
Parameter GoodReserveCDai.claimGDX(address,uint256,bytes32[])._gdx (myFlats/GoodReserve.sol#7771) is not in mixedCase
Parameter GoodReserveCDai.claimGDX(address,uint256,bytes32[])._proof (myFlats/GoodReserve.sol#7772) is not in mixedCase
Parameter GoodReserveCDai.pre(address,bytes32,bytes32)._method (myFlats/GoodReserve.sol#7794) is not in mixedCase
```

During the automated testing, it was found that the GoodReserveCDai contract had quite a few code-style issues.

Recommendation:
Therefore, it is recommended to fix the issues like naming convention, indentation, and code layout issues in a smart contract.

Amended: The team has fixed the issue, and it is no longer present in the code.

5. **Coding Style Issues in the Contract**

Contract - StakingRewardsFixedAPY
Explanation:
Code readability of a Smart Contract is largely influenced by the Coding Style issues and, in some specific scenarios, may lead to bugs in the future.

```
Parameter StakingRewardsFixedAPY.getPrinciple(address)._account (myFlats/flatStakingAPY.sol#1917) is not in mixedCase
Parameter StakingRewardsFixedAPY.earned(address)._account (myFlats/flatStakingAPY.sol#1949) is not in mixedCase
```

During the automated testing, it was found that the StakingRewardsFixedAPY contract had quite a few code-style issues.

Recommendation:
Therefore, it is recommended to fix the issues like naming convention, indentation, and code layout issues in a smart contract.

Amended: The team has fixed the issue, and it is no longer present in the code.

6. **Hardcoded Addresses can be avoided.**
   Line no - 55, 57

   Contract - DistributionHelper
   Explanation:
   Keeping in mind the immutable nature of smart contracts, it is not considered a better practice to hardcode any address in the contract before deployment.

   Recommendation:
   Instead of including hardcoded addresses in the contract, initialize those addresses within the constructors at the time of deployment.

   Amended: The team has fixed the issue, and it is no longer present in the code.

7. **Coding Style Issues in the Contract**

   Contract - DistributionHelper
   Explanation:
   Code readability of a Smart Contract is largely influenced by the Coding Style issues and, in some specific scenarios, may lead to bugs in the future.

```
Parameter DistributionHelper.initialize(INameService)._ns (myFlats/flatDistribution.sol#4050) is not in mixedCase
Parameter DistributionHelper.onDistribution(uint256)._amount (myFlats/flatDistribution.sol#4065) is not in mixedCase
Parameter DistributionHelper.addOrUpdateRecipient(DistributionHelper.DistributionRecipient)._recipient (myFlats/flatDistribution.sol#4088) is not in mixedCase
Parameter DistributionHelper.distribute(DistributionHelper.DistributionRecipient,uint256)._recipient (myFlats/flatDistribution.sol#4116) is not in mixedCase
Parameter DistributionHelper.distribute(DistributionHelper.DistributionRecipient,uint256)._amount (myFlats/flatDistribution.sol#4116) is not in mixedCase
```

   During the automated testing, it was found that the DistributionHelper contract had quite a few code-style issues.

   Recommendation:
   Therefore, it is recommended to fix the issues like naming convention, indentation, and code layout issues in a smart contract.

   Amended: The team has fixed the issue, and it is no longer present in the code.

8. **Commented codes must be wiped-out before deployment**

   Contract - DistributionHelper
   Explanation:
   The DistributionHelper contract includes quite a few commented codes within the contract body.

This badly affects the readability of the code.

Recommendation:
If these instances of code are not required in the current version of the contract, then the commented codes must be removed before deployment.

Amended: The team has fixed the issue, and it is no longer present in the code.

Version 2:

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

No issues were found.

Recommendation / Informational

1. **External Visibility should be preferred**

   Contract - StakingRewardsFixedAPY.sol
   Explanation:

   Functions never called throughout the contract should be marked as external visibility instead of public visibility.
   This will effectively result in Gas Optimization as well.

   Therefore, the following function must be marked as external within the contract:
   - amountToShares()

   Recommendation:
   If the PUBLIC visibility of the above-mentioned functions is not intended, then the EXTERNAL Visibility keyword should be preferred.

2. Coding Style Issues in the Contract

Contract - StakingRewardsFixedAPY.sol

Explanation:

Code readability of a Smart Contract is largely influenced by the Coding Style issues and in some specific scenarios may lead to bugs in the future.

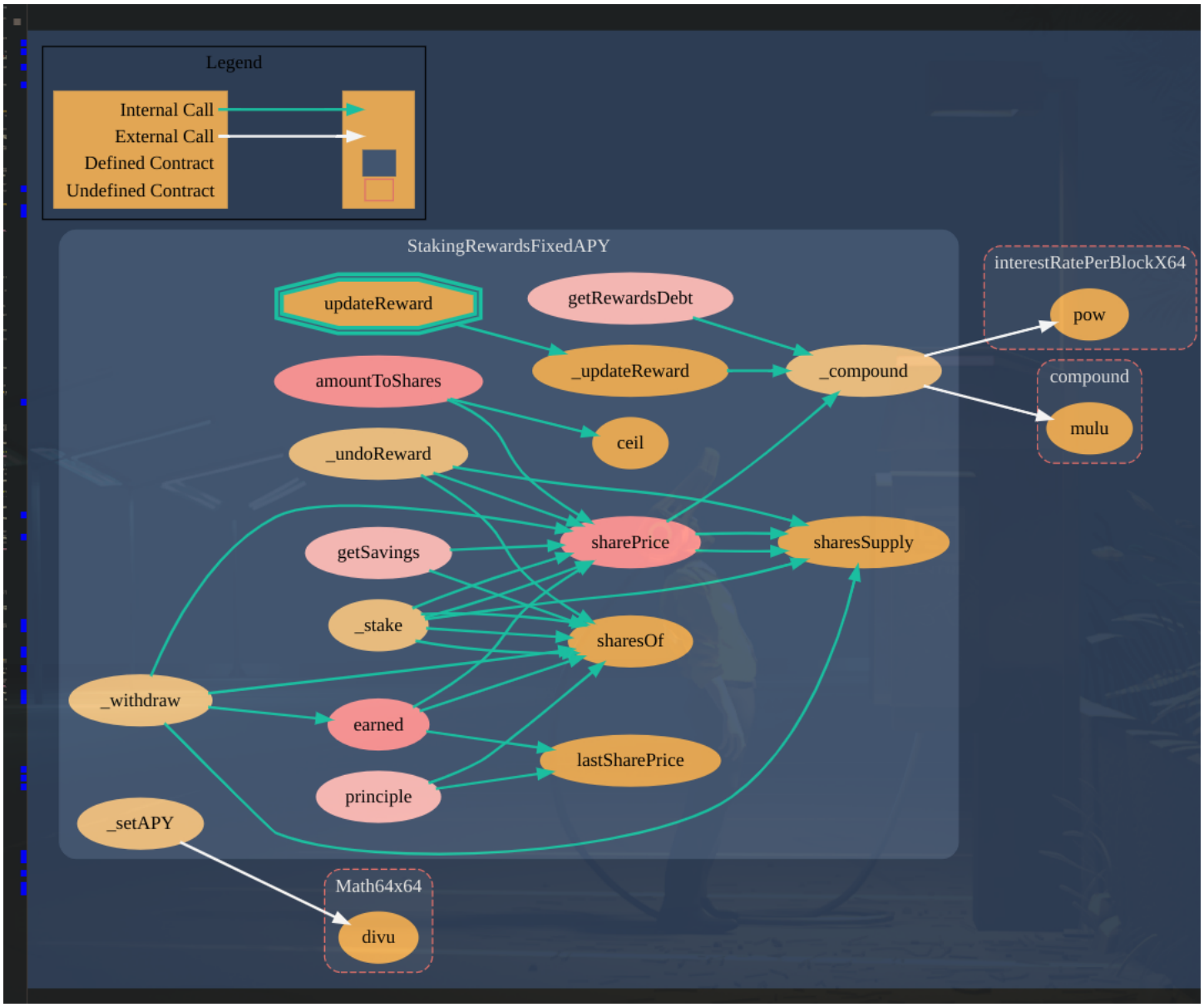
```
Parameter StakingRewardsFixedAPY.getSavings(address)._account (myFlats/flatStakingAPY.sol#1925) is not in mixedCase
Parameter StakingRewardsFixedAPY.amountToShares(uint256)._amount (myFlats/flatStakingAPY.sol#1944) is not in mixedCase
Parameter StakingRewardsFixedAPY.lastSharePrice(address)._account (myFlats/flatStakingAPY.sol#1959) is not in mixedCase
Parameter StakingRewardsFixedAPY.principle(address)._account (myFlats/flatStakingAPY.sol#1970) is not in mixedCase
Parameter StakingRewardsFixedAPY.earned(address)._account (myFlats/flatStakingAPY.sol#1984) is not in mixedCase
```

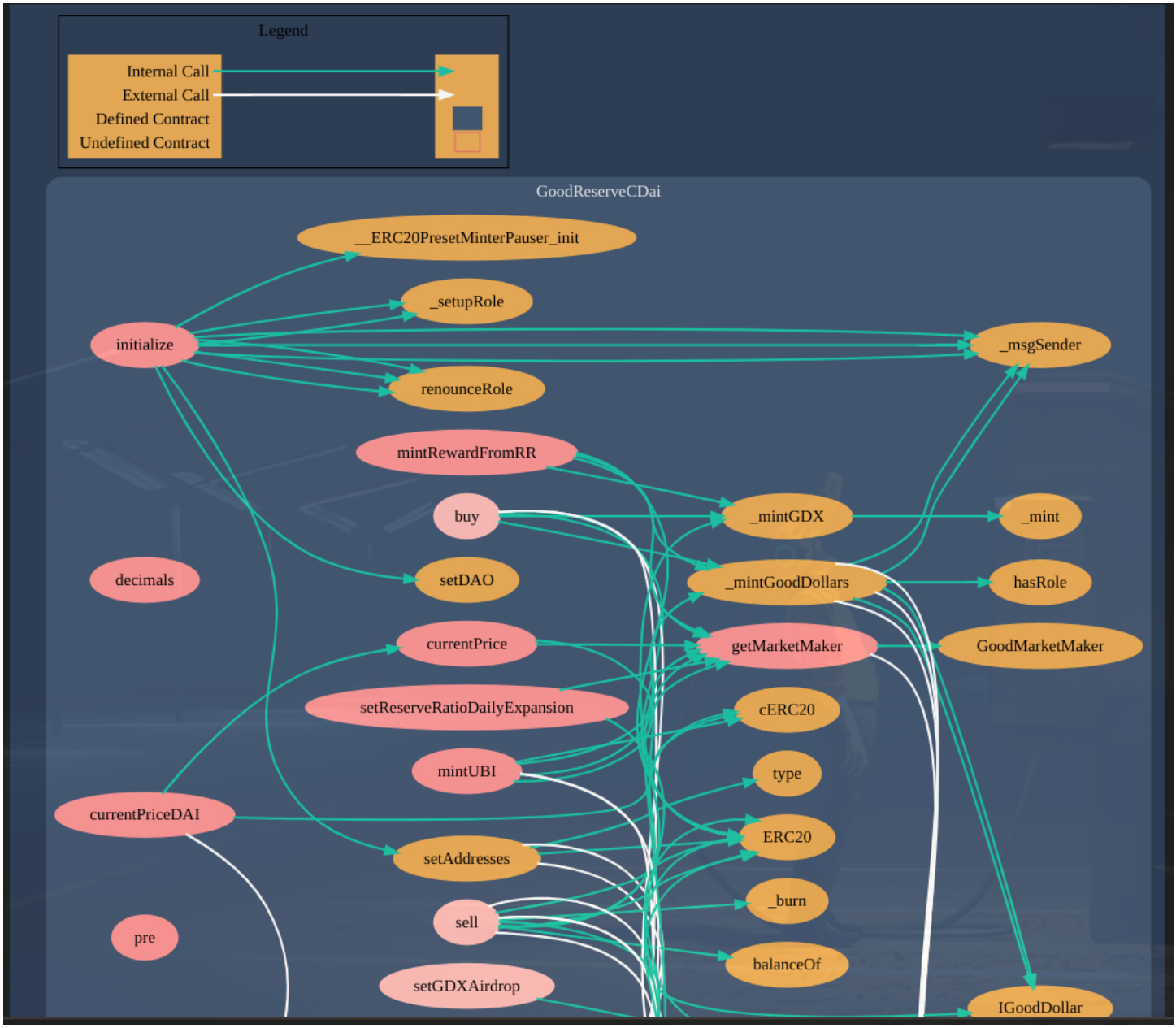
During the automated testing, it was found that the StakingRewardsFixedAPY contract had quite a few code-style issues.

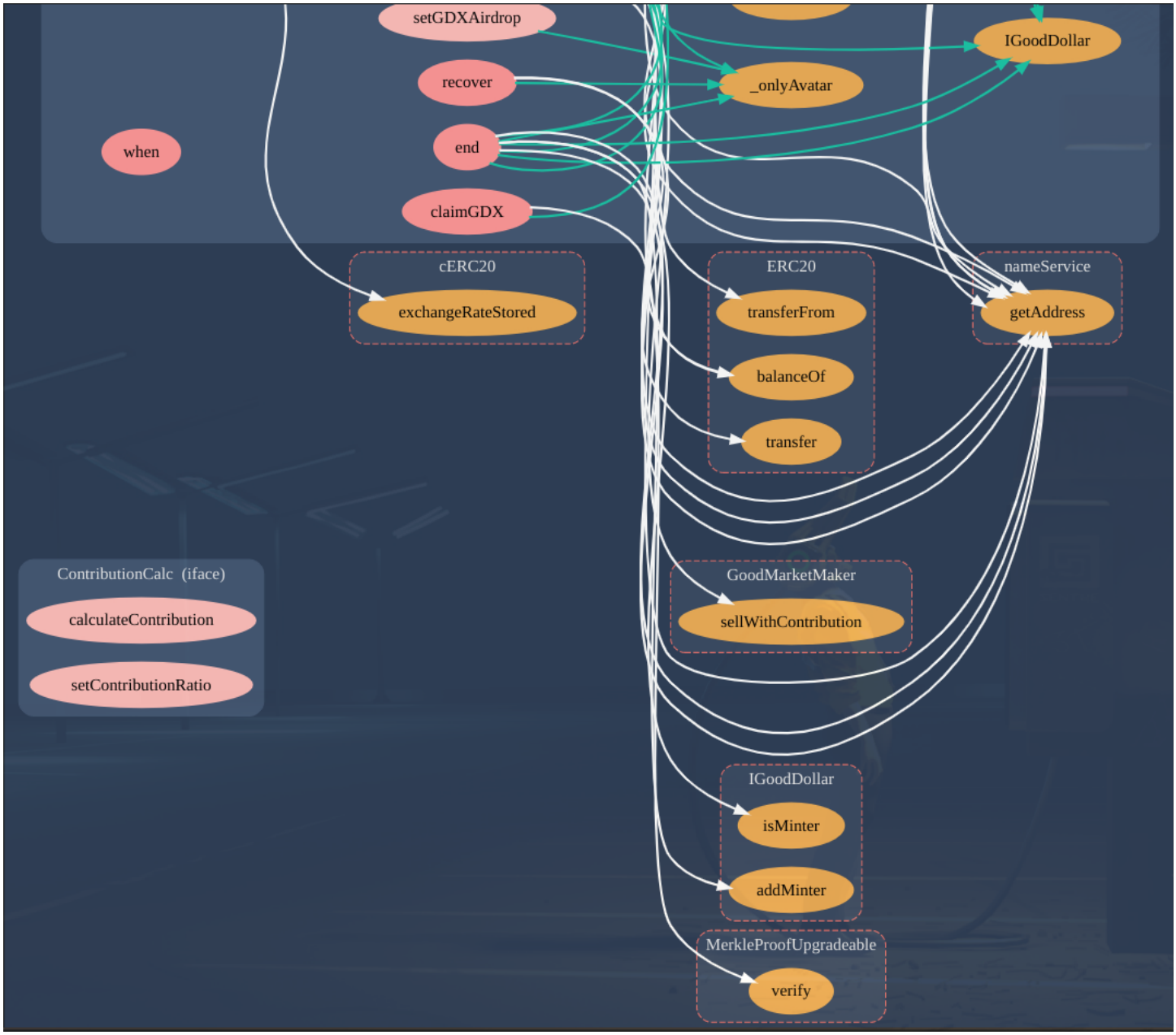
Recommendation:

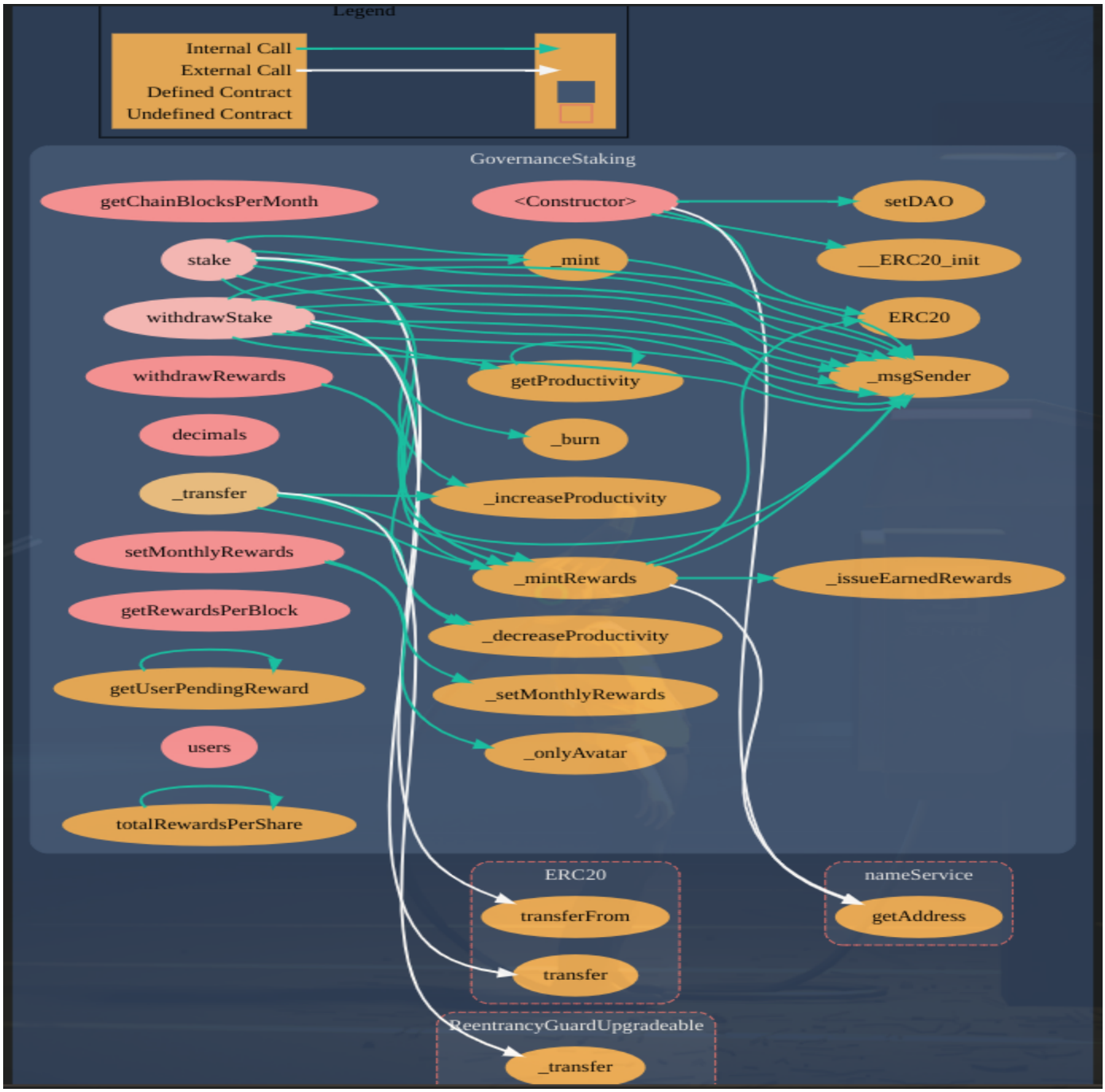
Therefore, it is recommended to fix the issues like naming convention, indentation, and code layout issues in a smart contract.

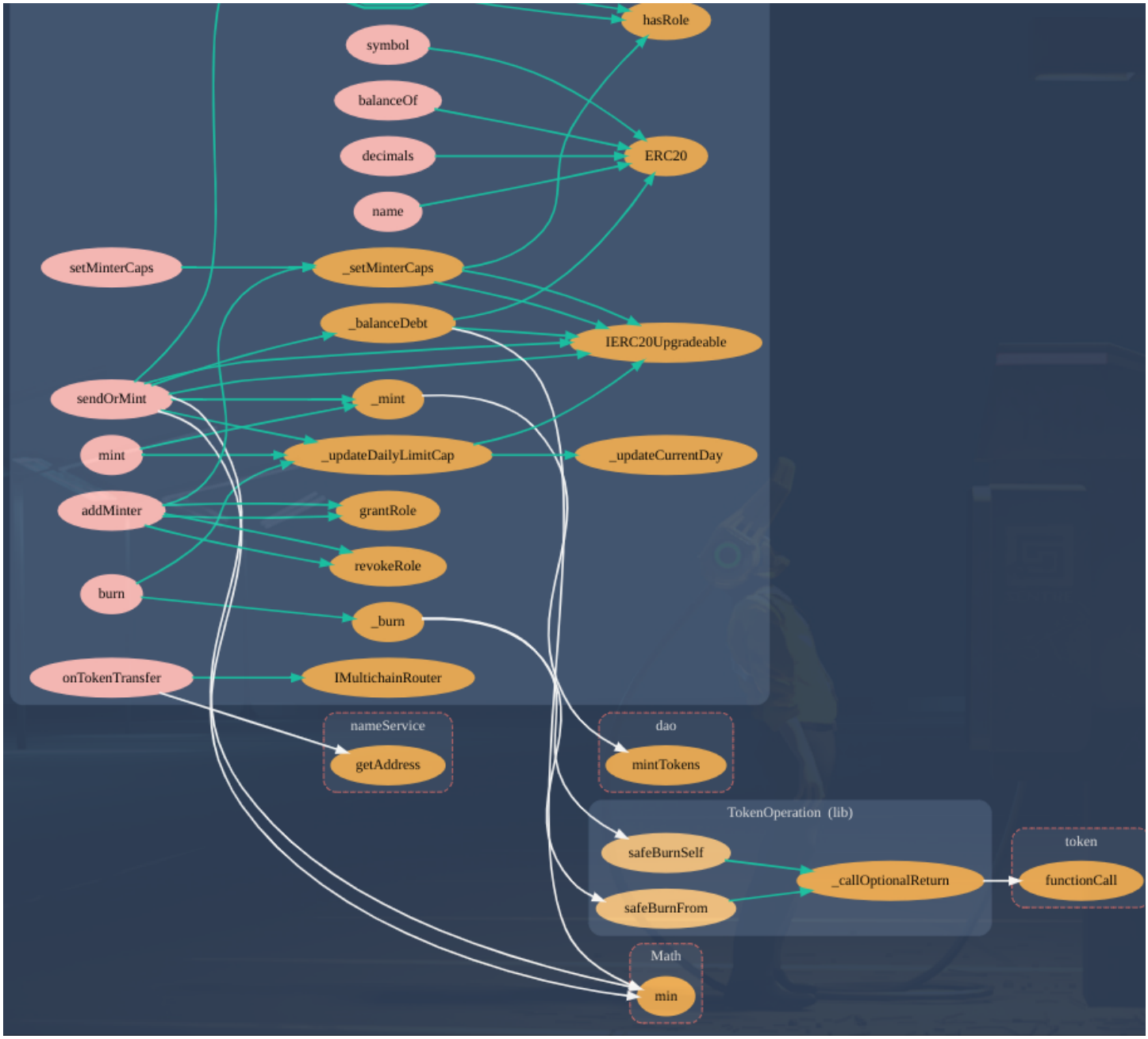
Call Graph



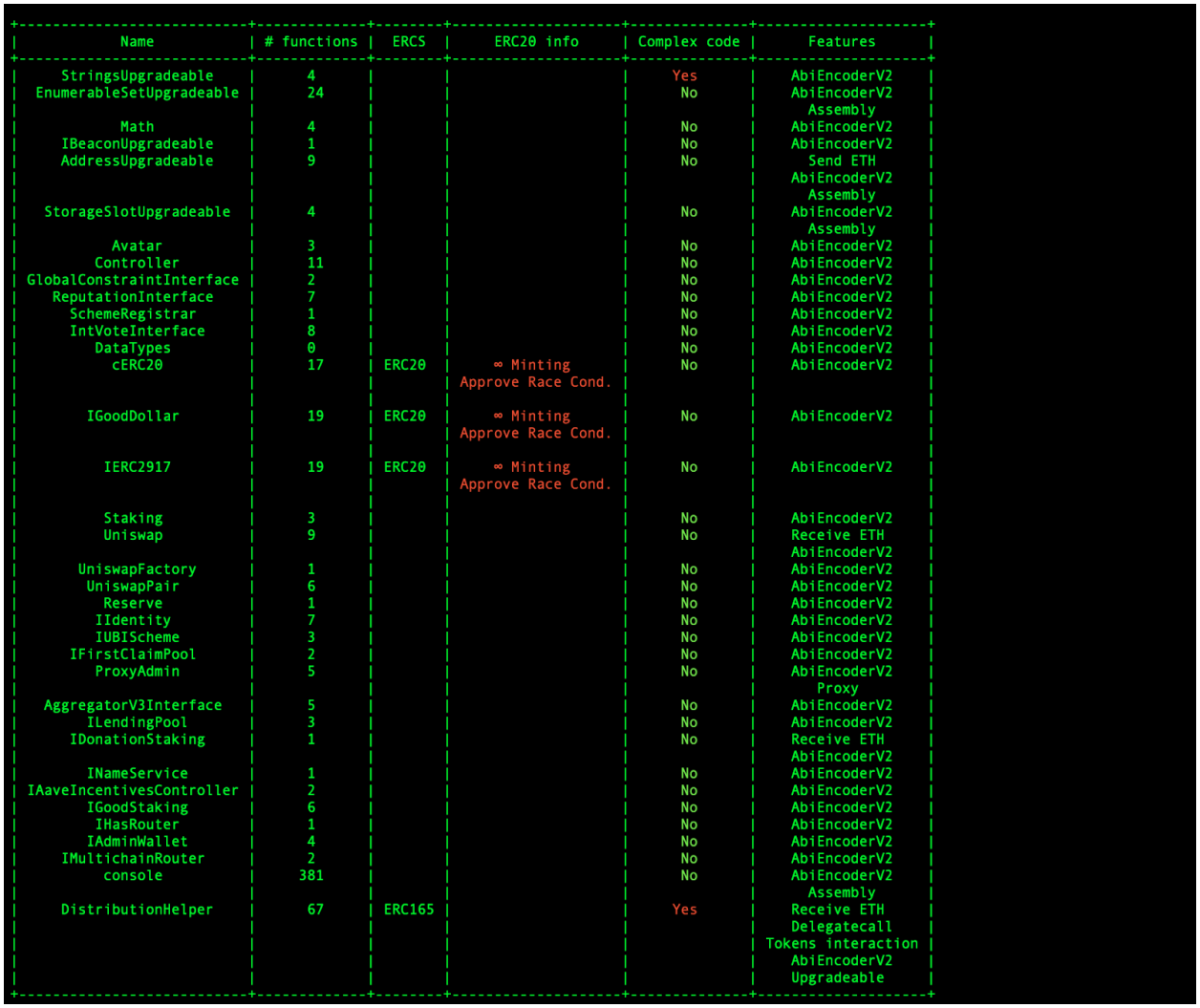








Automated Audit Result

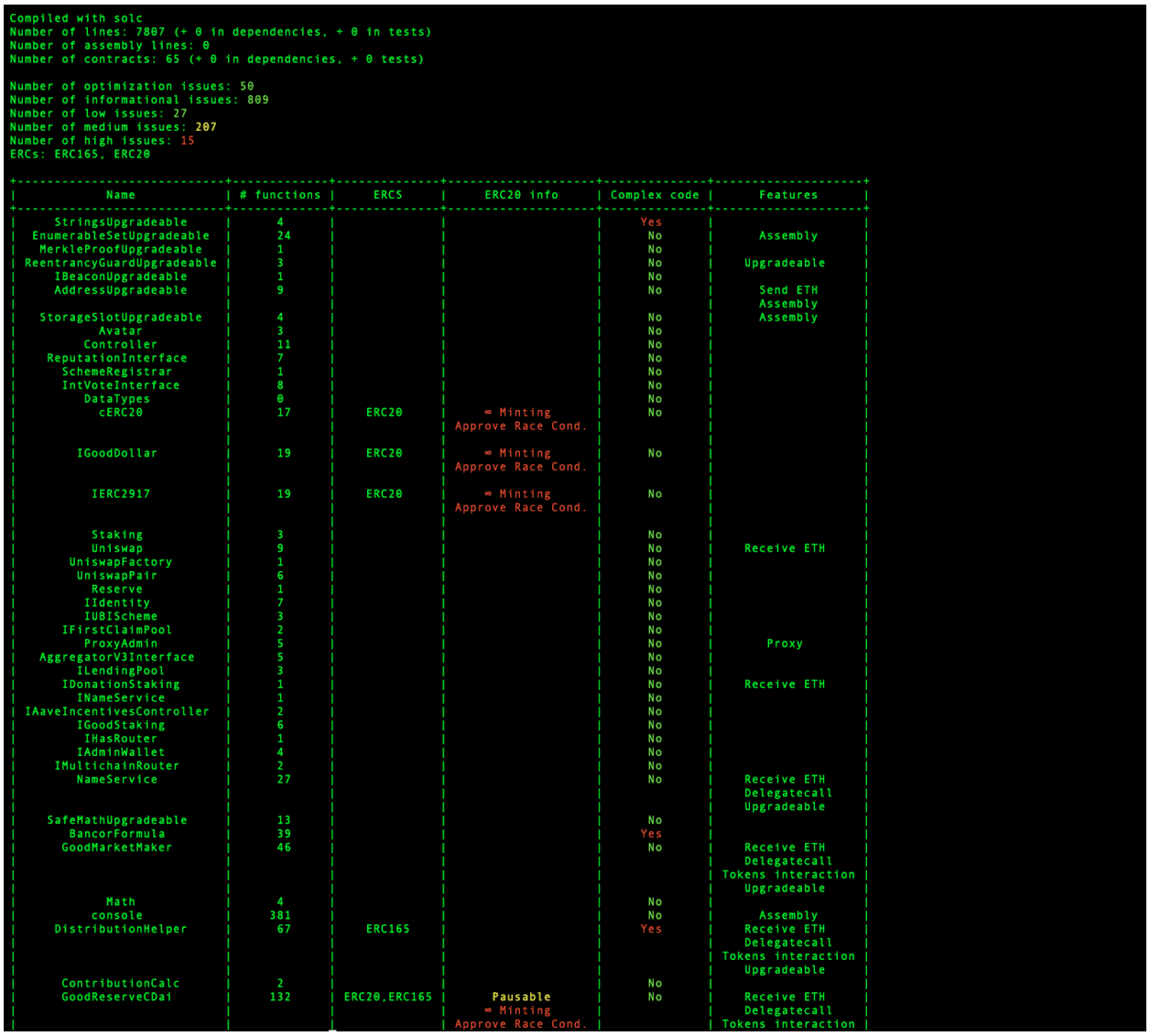


Name	# functions	ERC5	ERC20 info	Complex code	Features
StringsUpgradeable	4			Yes	AbiEncoderV2
EnumerableSetUpgradeable	24			No	AbiEncoderV2 Assembly
Math	4			No	AbiEncoderV2
IBeaconUpgradeable	1			No	AbiEncoderV2
AddressUpgradeable	9			No	Send ETH AbiEncoderV2 Assembly
StorageSlotUpgradeable	4			No	AbiEncoderV2 Assembly
Avatar	3			No	AbiEncoderV2
Controller	11			No	AbiEncoderV2
GlobalConstraintInterface	2			No	AbiEncoderV2
ReputationInterface	7			No	AbiEncoderV2
SchemeRegistrar	1			No	AbiEncoderV2
IntVoteInterface	8			No	AbiEncoderV2
DataTypes	0			No	AbiEncoderV2
cERC20	17	ERC20	∞ Minting Approve Race Cond.	No	AbiEncoderV2
IGoodDollar	19	ERC20	∞ Minting Approve Race Cond.	No	AbiEncoderV2
IERC2917	19	ERC20	∞ Minting Approve Race Cond.	No	AbiEncoderV2
Staking	3			No	AbiEncoderV2
Uniswap	9			No	Receive ETH AbiEncoderV2
UniswapFactory	1			No	AbiEncoderV2
UniswapPair	6			No	AbiEncoderV2
Reserve	1			No	AbiEncoderV2
IIdentity	7			No	AbiEncoderV2
IUBIScheme	3			No	AbiEncoderV2
IFirstClaimPool	2			No	AbiEncoderV2
ProxyAdmin	5			No	AbiEncoderV2 Proxy
AggregatorV3Interface	5			No	AbiEncoderV2
ILendingPool	3			No	AbiEncoderV2
IDonationStaking	1			No	Receive ETH AbiEncoderV2
INameService	1			No	AbiEncoderV2
IAaveIncentivesController	2			No	AbiEncoderV2
IGoodStaking	6			No	AbiEncoderV2
IHasRouter	1			No	AbiEncoderV2
IAdminWallet	4			No	AbiEncoderV2
IMultichainRouter	2			No	AbiEncoderV2
console	381			No	AbiEncoderV2 Assembly
DistributionHelper	67	ERC165		Yes	Receive ETH Delegatecall Tokens interaction AbiEncoderV2 Upgradeable

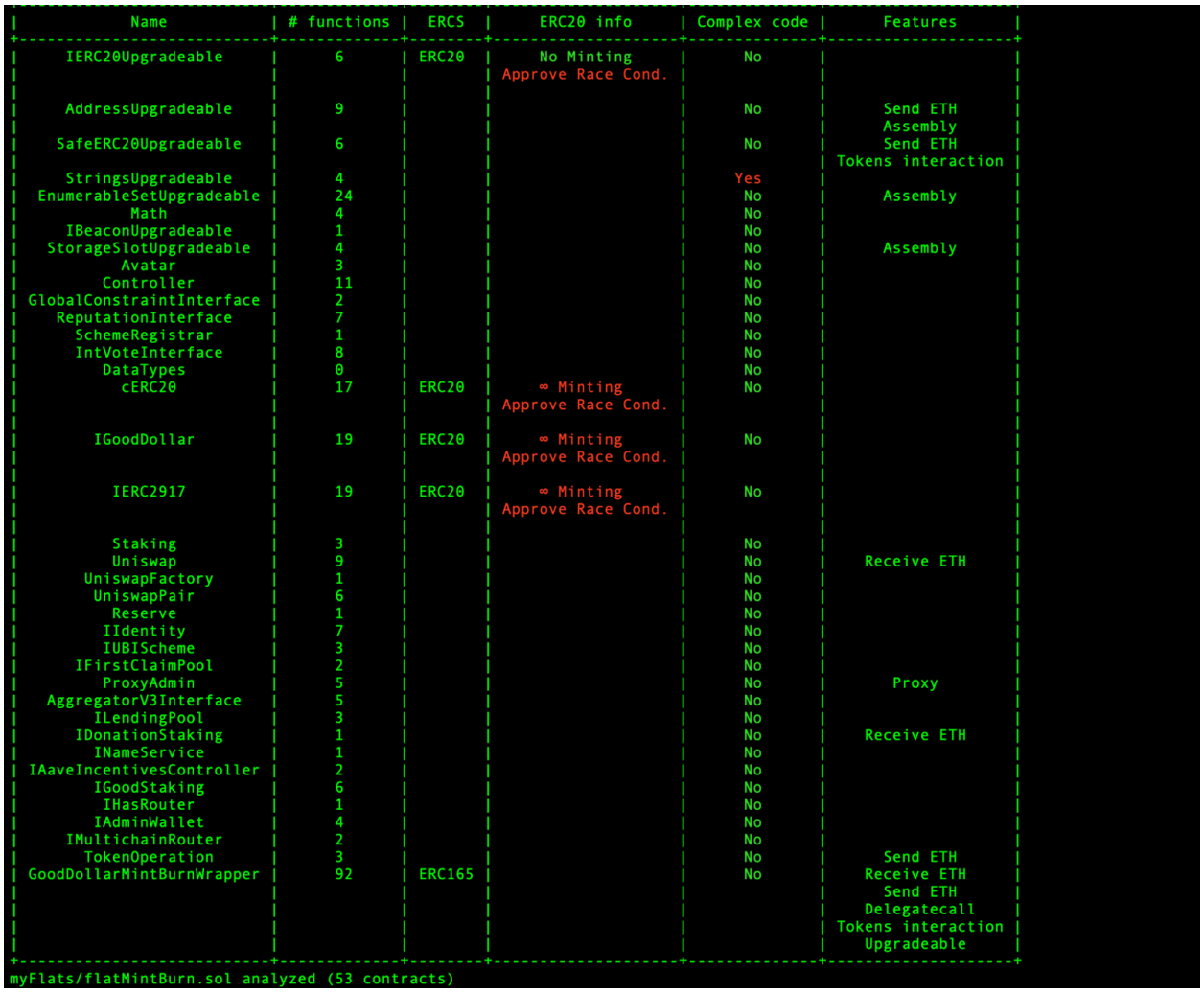
This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore; running a bug bounty program as a complement to this audit is strongly recommended.

```
Compiled with solc
Number of lines: 2146 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 2 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 0
Number of informational issues: 26
Number of low issues: 0
Number of medium issues: 8
Number of high issues: 0


+-------------------------+-------------+------+------------+--------------+----------+
|          Name           | # functions | ERCS | ERC20 info | Complex code | Features |
+-------------------------+-------------+------+------------+--------------+----------+
|       Math64x64         |      7      |      |            |     Yes      |          |
| StakingRewardsFixedAPY  |      11     |      |            |      No      |          |
+-------------------------+-------------+------+------------+--------------+----------+
```


```
Compiled with solc
Number of lines: 2146 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 2 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 0
Number of informational issues: 26
Number of low issues: 0
Number of medium issues: 8
Number of high issues: 0


+-------------------------+-------------+------+------------+--------------+----------+
|          Name           | # functions | ERCS | ERC20 info | Complex code | Features |
+-------------------------+-------------+------+------------+--------------+----------+
|       Math64x64         |      7      |      |            |     Yes      |          |
| StakingRewardsFixedAPY  |      11     |      |            |      No      |          |
+-------------------------+-------------+------+------------+--------------+----------+
```



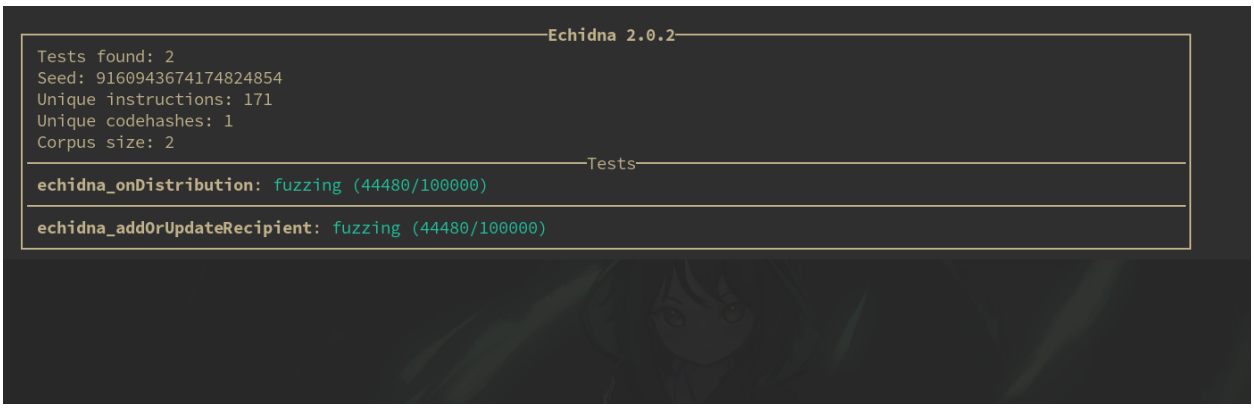
```
Compiled with solc
Number of lines: 7807 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 65 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 50
Number of informational issues: 809
Number of low issues: 27
Number of medium issues: 207
Number of high issues: 15
ERCs: ERC165, ERC20

+---------------------------+------------+--------------+--------------------+--------------+--------------------+
|           Name            | # functions |     ERCS     |     ERC20 info     | Complex code |      Features      |
+---------------------------+------------+--------------+--------------------+--------------+--------------------+
|     StringsUpgradeable    |      4     |              |                    |     Yes      |                    |
|  EnumerableSetUpgradeable |     24     |              |                    |      No      |      Assembly      |
|   MerkleProofUpgradeable  |      1     |              |                    |      No      |                    |
| ReentrancyGuardUpgradeable |      3     |              |                    |      No      |    Upgradeable     |
|     IBeaconUpgradeable    |      1     |              |                    |      No      |                    |
|     AddressUpgradeable    |      9     |              |                    |      No      |      Send ETH      |
|                           |            |              |                    |              |      Assembly      |
|   StorageSlotUpgradeable  |      4     |              |                    |      No      |      Assembly      |
|          Avatar           |      3     |              |                    |      No      |                    |
|        Controller         |     11     |              |                    |      No      |                    |
|    ReputationInterface    |      7     |              |                    |      No      |                    |
|      SchemeRegistrar      |      1     |              |                    |      No      |                    |
|      IntVoteInterface     |      8     |              |                    |      No      |                    |
|         DataTypes         |      0     |              |                    |      No      |                    |
|           cERC20          |     17     |    ERC20     |     ∞ Minting      |      No      |                    |
|                           |            |              | Approve Race Cond. |              |                    |
|       IGoodDollar         |     19     |    ERC20     |     ∞ Minting      |      No      |                    |
|                           |            |              | Approve Race Cond. |              |                    |
|         IERC2917          |     19     |    ERC20     |     ∞ Minting      |      No      |                    |
|                           |            |              | Approve Race Cond. |              |                    |
|          Staking          |      3     |              |                    |      No      |                    |
|          Uniswap          |      9     |              |                    |      No      |    Receive ETH     |
|       UniswapFactory      |      1     |              |                    |      No      |                    |
|        UniswapPair        |      6     |              |                    |      No      |                    |
|          Reserve          |      1     |              |                    |      No      |                    |
|         IIdentity         |      7     |              |                    |      No      |                    |
|         IUBIScheme        |      3     |              |                    |      No      |                    |
|      IFirstClaimPool      |      2     |              |                    |      No      |                    |
|         ProxyAdmin        |      5     |              |                    |      No      |       Proxy        |
|   AggregatorV3Interface   |      5     |              |                    |      No      |                    |
|        ILendingPool       |      3     |              |                    |      No      |                    |
|      IDonationStaking     |      1     |              |                    |      No      |    Receive ETH     |
|       INameService        |      1     |              |                    |      No      |                    |
| IAaveIncentivesController |      2     |              |                    |      No      |                    |
|        IGoodStaking       |      6     |              |                    |      No      |                    |
|         IHasRouter        |      1     |              |                    |      No      |                    |
|        IAdminWallet       |      4     |              |                    |      No      |                    |
|     IMultichainRouter     |      2     |              |                    |      No      |                    |
|        NameService        |     27     |              |                    |      No      |    Receive ETH     |
|                           |            |              |                    |              |    Delegatecall    |
|                           |            |              |                    |              |    Upgradeable     |
|    SafeMathUpgradeable    |     13     |              |                    |      No      |                    |
|       BancorFormula       |     39     |              |                    |     Yes      |                    |
|      GoodMarketMaker      |     46     |              |                    |      No      |    Receive ETH     |
|                           |            |              |                    |              |    Delegatecall    |
|                           |            |              |                    |              | Tokens interaction |
|                           |            |              |                    |              |    Upgradeable     |
|           Math            |      4     |              |                    |      No      |                    |
|          console          |     381    |              |                    |      No      |      Assembly      |
|     DistributionHelper    |     67     |    ERC165    |                    |     Yes      |    Receive ETH     |
|                           |            |              |                    |              |    Delegatecall    |
|                           |            |              |                    |              | Tokens interaction |
|                           |            |              |                    |              |    Upgradeable     |
|      ContributionCalc     |      2     |              |                    |      No      |                    |
|      GoodReserveCDai      |     132    | ERC20,ERC165 |      Pausable      |      No      |    Receive ETH     |
|                           |            |              |     ∞ Minting      |              |    Delegatecall    |
|                           |            |              | Approve Race Cond. |              | Tokens interaction |
```

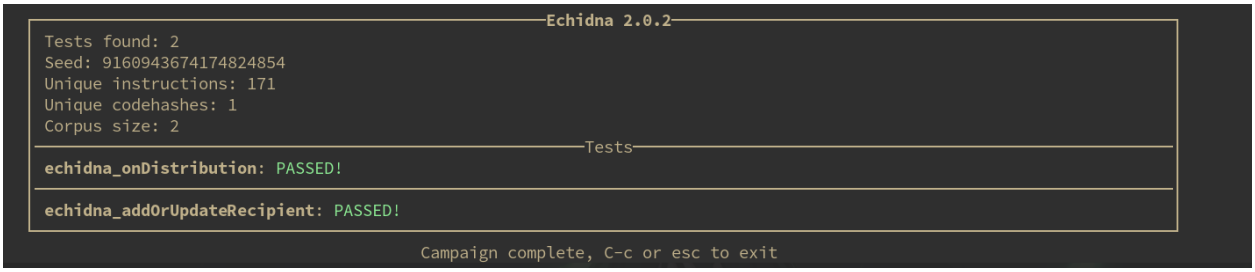


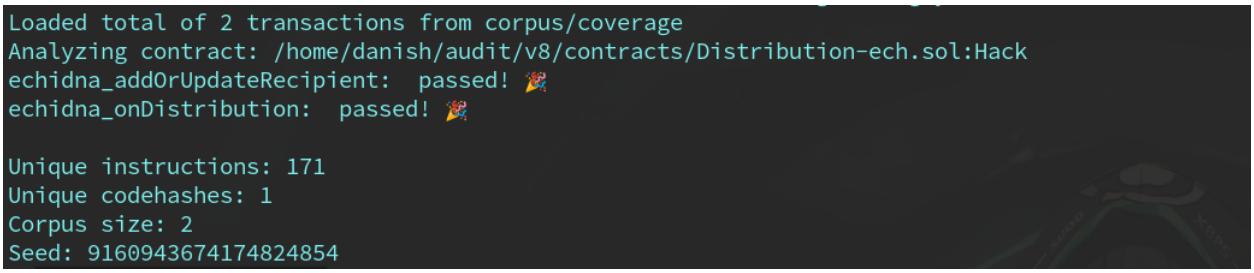
Name	# functions	ERCS	ERC20 info	Complex code	Features
IERC20Upgradeable	6	ERC20	No Minting Approve Race Cond.	No	
AddressUpgradeable	9			No	Send ETH Assembly
SafeERC20Upgradeable	6			No	Send ETH Tokens interaction
StringsUpgradeable	4			Yes	
EnumerableSetUpgradeable	24			No	Assembly
Math	4			No	
IBeaconUpgradeable	1			No	
StorageSlotUpgradeable	4			No	Assembly
Avatar	3			No	
Controller	11			No	
GlobalConstraintInterface	2			No	
ReputationInterface	7			No	
SchemeRegistrar	1			No	
IntVoteInterface	8			No	
DataTypes	0			No	
cERC20	17	ERC20	∞ Minting Approve Race Cond.	No	
IGoodDollar	19	ERC20	∞ Minting Approve Race Cond.	No	
IERC2917	19	ERC20	∞ Minting Approve Race Cond.	No	
Staking	3			No	
Uniswap	9			No	Receive ETH
UniswapFactory	1			No	
UniswapPair	6			No	
Reserve	1			No	
IIdentity	7			No	
IUBIScheme	3			No	
IFirstClaimPool	2			No	
ProxyAdmin	5			No	Proxy
AggregatorV3Interface	5			No	
ILendingPool	3			No	
IDonationStaking	1			No	Receive ETH
INameService	1			No	
IAaveIncentivesController	2			No	
IGoodStaking	6			No	
IHasRouter	1			No	
IAdminWallet	4			No	
IMultichainRouter	2			No	
TokenOperation	3			No	Send ETH
GoodDollarMintBurnWrapper	92	ERC165		No	Receive ETH Send ETH Delegatecall Tokens interaction Upgradeable

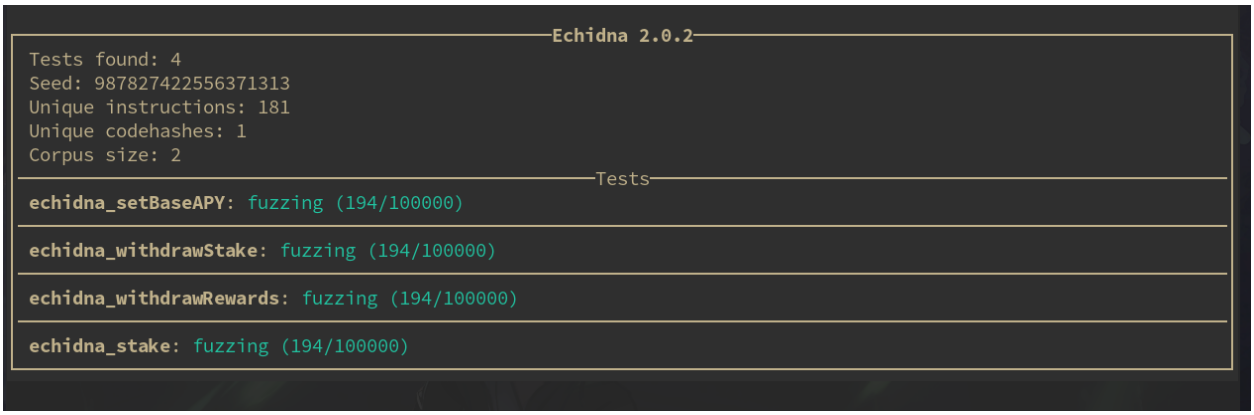
myFlats/flatMintBurn.sol analyzed (53 contracts)

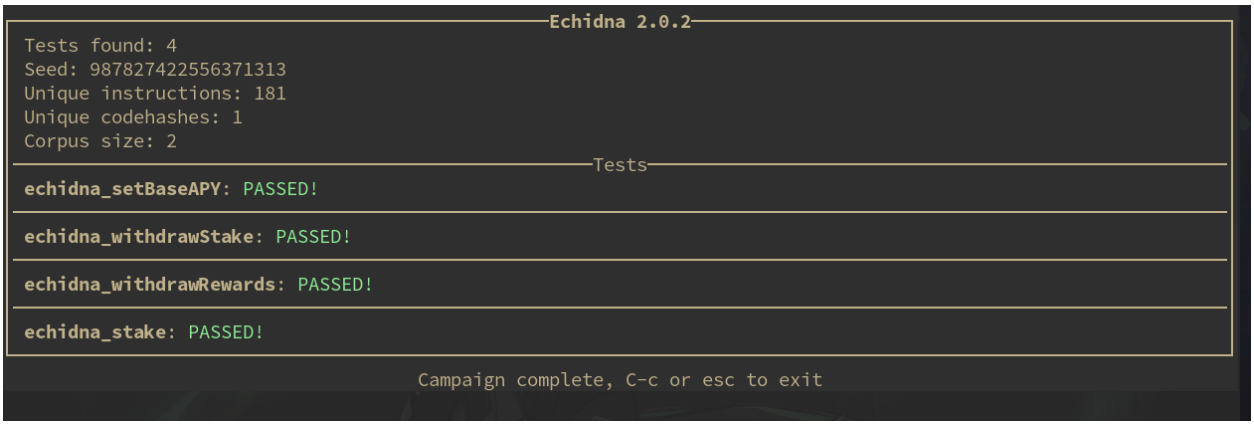
Fuzz Test Result

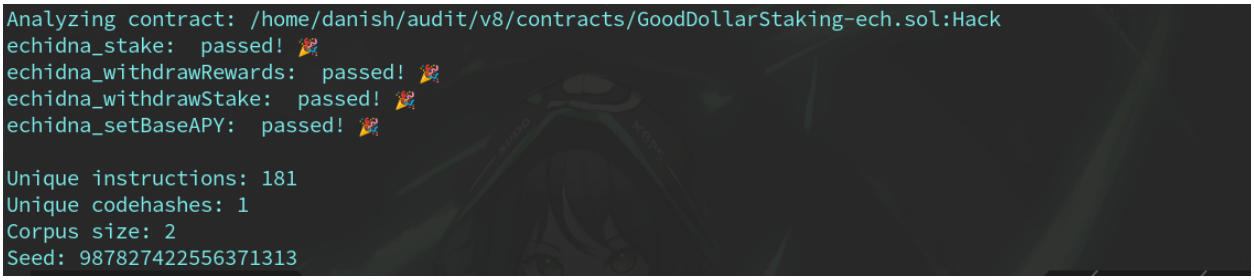


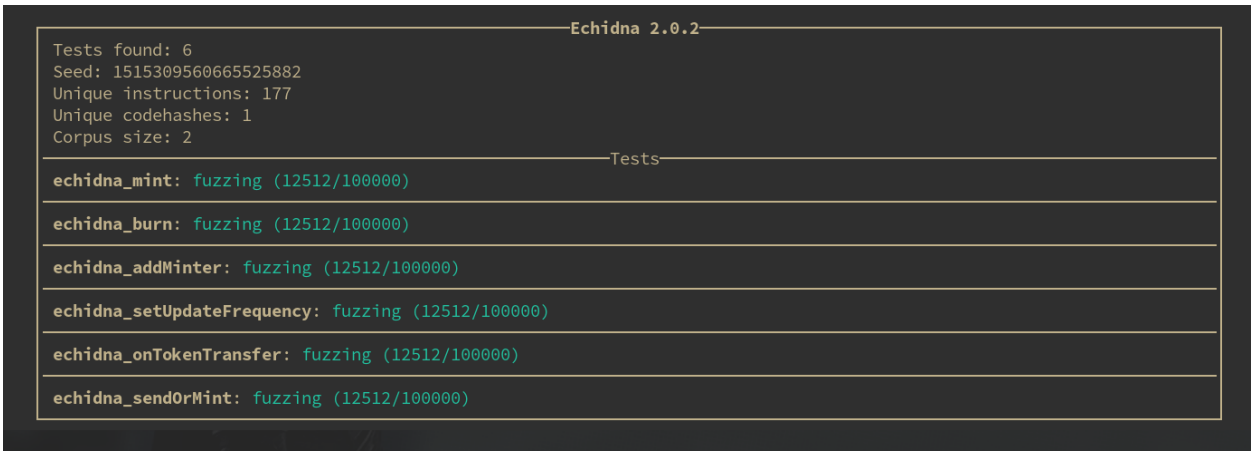


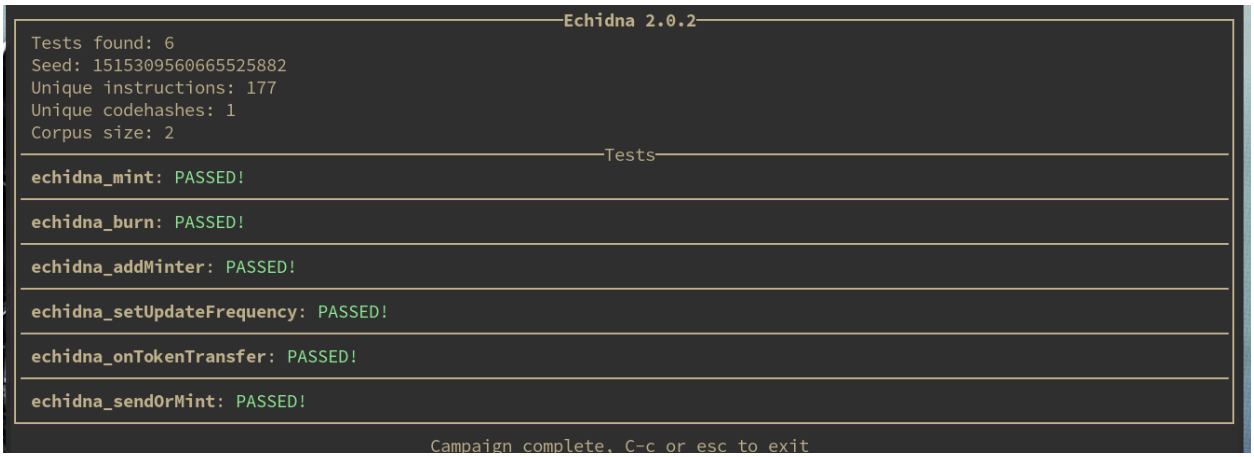












Concluding Remarks

While conducting the audits of the GoodDollar smart contracts, it was observed that the contracts contain Medium and Low severity issues.

Our auditors suggest that Medium and Low severity issues should be resolved by the developers. The recommendations given will improve the operations of the smart contract.

Disclaimer

ImmuneBytes's audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process; therefore, running a bug bounty program as a complement to this audit is strongly recommended.

Our team does not endorse the GoodDollar platform or its product nor this audit is investment advice.
Notes:

- Please make sure contracts deployed on the mainnet are the ones audited.
- Check for the code refactor by the team on critical issues.

ImmuneBytes