Asfalia

Security Audit

SimpleDEFI

Asfalia Audit on Oct 15th, 2022
Updated on 20/02/2023

Table of Contents

Summary

This report has been prepared for SimpleDEFI to discover issues and vulnerabilities in the source code of the SimpleDEFI project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilising Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from Medium to informational.
We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in
- public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	SimpleDEFI
Platform	EVM
Chain	Binance Smart Chain
Language	Solidity
Codebase	Files provided
Commit	**SimpleDefiDiamond:** ec7aa3fe9e85d220118fd947ce26e1324d442bcc **SimpleDefiSolo:** 3ded73e49269825aae522c3a558c6b70362940a9 **SimpleDefiBeacon:** a082f36f2bf1ac202c7d4cd30009f37e2d1a8333 **SimpleDefiToken:** ac462850a5c10c5b9aeec89cbecdf323bb2b4dc5 **SimpleDefiEasy:** 0x5953f0729fBF901846AF5a91C222DB15634d90E1

Audit Summary

Delivery Date	15/10/2022
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Good Security Scoring: 85 / 100

Total	Pending	Acknowledged	Unresolved	Partially Resolved	Resolved
38	0	11	0	0	26

Critical	High	Medium	Low	Informational	Optimization
0	1	5	9	15	8

Scope

Repository: N/A

Technical Documentation: N/A

Contracts:

Diamond.sol
DiamondFactory.sol
DiamondCutFacet.sol
DiamondLoupeFacet.sol
LibDiamond.sol
OwnershipFacet.sol
sdData.sol
sdDepositFunds.sol
sdInitialize.sol
sdMigration.sol
sdSystem.sol
sdPoolUtil.sol
slots.sol
combine_beacon.sol
combine_proxy.sol
CombineApp.sol
Interfaces.sol
Storage.sol
SimpleDefiToken.sol
SimpleDefiEasy.sol

Project Overview

At SimpleDEFI help Retail & Institutions automate and manage yield strategies anywhere on chain on a single UI. And SimpleDefi help Web3 projects attract liquidity and enable more utility for their tokens.

Project Architecture & Fee Models

N/A

Contract Dependencies

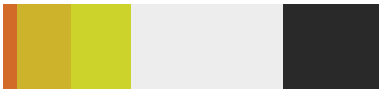
N/A

Privileged Roles

owner
adminUsers
godUsers

Findings

Total Issues: **37**



Critical	High	Medium	Low	Informational	Optimization
0	1	5	9	15	8

ID	Title	Type	Categories	Severity	Status
#G01	Floating Pragma	SWC-103	Coding Style	Low	Resolved
#G02	Floating Pragma	SWC-103	Coding Style	Low	Resolved
#G03	Code With No effects	SWC-135	Coding Style	Informational	Resolved
#G04	Code With No Effects	SWC-135	Coding Style	Informational	Resolved
#1	Typographical Error	SWC-129	Volatile Code	Low	Resolved
#2	Code With No Effect	SWC-135	Volatile Code	Informational	Acknowledged
#3	Function Default Visibility	SWC-100	Gas Optimization	Optimization	Resolved
#4	Code With No Effects	SWC-135	Coding Style	Informational	Resolved
#5	Function Default Visibility	SWC-100	Gas Optimization	Optimization	Resolved
#6	Code With No Effects	SWC-135	Coding style	Informational	Resolved
#7	Authorization through tx.origin	SWC-115	Volatile Code	Medium	Resolved
#8	Authorization through tx.origin	SWC-115	Block Timestap Manipulation	Medium	Acknowledged
#9	Function Default Visibility	SWC-100	Gas Optimization	Optimization	Resolved
#10	Authorization through tx.origin	SWC-115	Volatile Code	Low	Acknowledged
#11	Function Default Visibility	SWC-100	Gas Optimization	Optimization	Resolved
#12	Integer Overflow and Underflow	SWC-101	Coding Style	Informational	Resolved

#13	Unchecked Call Return Value	SWC-104	Coding Style	Informational	Resolved
#14	Code With No Effects	SWC-135	Coding Style	Informational	Resolved
#15	Authorization through tx.origin	SWC-115	Block Timestamp Manipulation	Medium	Acknowledged
#16	Code With No Effects	SWC-135	Coding Style	Informational	Resolved
#17	Uninitialized Storage Pointer	SWC-109	Coding Style	Informational	Acknowledged
#18	Centralization Related Risk	**Centralization / Privilege**	Volatile Code	Medium	Acknowledged
#19	Assembly Usage	**Custom**	Centralization / Privilege	Low	Acknowledged
#20	Function Default Visibility	SWC-100	Gas Optimization	Optimization	Resolved
#21	Code With No Effects	SWC-135	Coding Style	Informational	Resolved
#22	Code With No Effects	SWC-135	Coding Style	Informational	Resolved
#23	ETH Locked	**Custom**	Coding Style	Low	Acknowledged
#24	Assembly Usage	**Custom**	Gas Optimization	Optimization	Acknowledged
#25	Boolean Equality	**Custom**	Coding Style	Informational	Resolved
#26	Function Default Visibility	SWC-100	Gas Optimization	Optimization	Resolved
#27	Sandwich Attack	**Custom**	Coding Style	High	Acknowledged
#28	Typographical Error	SWC-129	Volatile Code	Low	Resolved
#29	Message call with hardcoded gas amount	SWC-134	Coding Style	Informational	Resolved
#30	Code With No Effects	SWC-135	Coding Style	Low	Acknowledged
#31	Code With No Effects	SWC-135	Coding Style	Medium	Resolved
#32	Code With No Effects	SWC-135	Coding Style	Informational	Resolved
#33	Function Default Visibility	SWC-100	Gas Optimization	Optimization	Resolved

General findings

Description

ID : G01
Lines 2
Type: SWC-103
Category: Coding Style
Sev: Low

Descript: Floating pragmas are used in the contracts within this repo.
Recom: Solidity version specified in pragma should be defined explicitly to avoid any potential errors when compiling.

ID : G02
Lines 2
Type: SWC-103
Category: Coding Style
Sev: Low

Descript: Pragma is not consistent between smart contracts.
Recom: Specify the same Solidity version for all smart contracts in order to avoid any inconsistencies between versions which may lead to unforeseen issues in contract interactions. Recommend use of Sol 0.8.7 as stable 0.8.X version.

ID : G03
Lines 3
Type: SWC-135
Category: Coding Style
Sev: Informational

Descript: Multiple contracts specify to use ABIEncoderV2, including: DiamondCutFacet.sol, DiamondLoupeFacet.sol, DiamondBeacon.sol, Diamond.sol, LibDiamond, combineApp.sol, combine_beacon.sol, IERC165.sol, IDiamondLoupe.sol & IDiamondCut.sol.
Recom: As of Sol 0.8.0, `experimental ABIEncoderV2` is deprecated as ABI Coder V2 is activated by default.

See https://docs.soliditylang.org/en/v0.8.15/080-breaking-changes.html.

ID : G04
Lines
Type: SWC-135
Category: Coding Style
Sev: Informational

Descript: Multiple contracts contain functions with ‘override’ despite not overriding from a virtual function.
Recom: Remove code with no effects.

CombineBeacon.sol

#1 SWC-129 Typographical Error

Category	Severity	Location	Status
Volatile Code	Low	Line 80	Resolved

Description
In Solidity multiplication should come before division to avoid rounding errors.

Recommendation
Restructure the mathematical operation.

Alleviation
Refactored calculation

#2 SWC-135 Code With No Effects

Category	Severity	Location	Status
Volatile Code	Informational	Line 47	Acknowledged

Description
Declared state variable bitFlip is declared but not used.

Recommendation
Remove code with no effects.

Alleviation

bitFlip is used in the debugging and is a placeholder to eliminate storage issue.

#3 SWC-100 Function Default Visibility

Category	Severity	Location	Status
Gas Optimization	Optimization	Line 131, 217, 250, 258	Resolved

Description
Functions that are not used internally in the function are declared as public.

Recommendation
Function visibility modifier should be changed to external for functions. This will save on gas when calling these functions as external functions do not need to save parameters to memory, unlike public functions which do.

Alleviation

Fixed Visibility

DiamondCutFacet.sol

No vulnerabilities detected

DiamondLoupeFacet.sol

#4 SWC-135 Code With No Effects

Category	Severity	Location	Status
Coding Style	Informational	Line 20 - 23	Resolved

Description

Lines contain commented code.

Recommendation

Remove code with no effects.

Alleviation

N/A

OwnershipFacet.sol

No vulnerabilities detected

sdData.sol

#5 SWC-100 Function Default Visibility

Category	Severity	Location	Status
Gas Optimization	Optimization	Line 39, 44, 51, 54, 61, 65, 69, 75, 176	Resolved

Description
Functions that are not used internally in the function are declared as public.

Recommendation
Function visibility modifier should be changed to external for functions. This will save on gas when calling these functions as external functions do not need to save parameters to memory, unlike public functions which do.

Alleviation
N/A

#6 SWC-135 Code With No Effects

Category	Severity	Location	Status
Coding Style	Informational	Line 98 -100	Resolved

Description
tokenBalance() function appears to not be used within the repository.

Recommendation
Remove code with no effects.

Alleviation
N/A

#7 SWC-129 Typographical Error

Category	Severity	Location	Status
Volatile Code	Medium	Line 157, 169	Resolved

Description

If adminUser, godUser is not transferred before status is set to false, the role will become unusable and could risk a new contract deployment required.

Recommendation

Ensure a new adminUser, godUser is set before the original godUser status is able to set to false.

Alleviation

N/A

sdDepositFunds.sol

#8 SWC-129 Typographical Error

Category	Severity	Location	Status
Block Timestamp Manipulation	Medium	Line 126, 129, 130	Acknowledged

Description

block.timestamp can be manipulated by miners with the following constraints

- It cannot be stamped with an earlier time than its parent
- It cannot be too far in the future

Recommendation

Don't use block.timestamp for a source and use block.number instead.

Alleviation

N/A

sdDepositFunds.sol

#9 SWC-100 Function Default Visibility

Category	Severity	Location	Status
Gas Optimization	Optimization	Line 28, 91	Resolved

Description

Functions that are not used internally in the function are declared as public. (Also declared as external in their interface on DiamondFactory.sol()

Recommendation

Function visibility modifier should be changed to external for functions. This will save on gas when calling these functions as external functions do not need to save parameters to memory, unlike public functions which do.

Alleviation

N/A

sdMigration.sol

#10 SWC-115 Authorization through tx.origin

Category	Severity	Location	Status
Volatile Code	Low	Line 28, 91	Acknowledged

Description

Use of tx.origin to verify caller is a godUser

Recommendation

Using “tx.origin” as a security control can lead to authorization bypass vulnerabilities. Consider using “msg.sender” instead.

Alleviation

N/A

#11 SWC-100 Function Default Visibility

Category	Severity	Location	Status
Gas Optimization	Optimization	Line 55, 72, 99, 115, 121, 136	Resolved

Description

Functions that are not used internally in the function are declared as public.

Recommendation

Function visibility modifier should be changed to external for functions. This will save on gas when calling these functions as external functions do not need to save parameters to memory, unlike public functions which do.

Alleviation

N/A

sdSystem.sol

#12 SWC-101 integer Overflow and Underflow

Category	Severity	Location	Status
Divide Before Multiply	Informational	Line 166, 167, 172, 273, 280	Resolved

Description

Performing integer division before multiplication truncates the low bits, losing the precision of calculation.

Recommendation

We recommend applying multiplication before division to avoid loss of precision.

Alleviation

N/A

#13 SWC-104 Unchecked Call Return Value

Category	Severity	Location	Status
Coding Style	Informational	Line 235	Resolved

Description
The return value of an external transfer/transferFrom call is not checked.

Recommendation
Use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

Alleviation
N/A

LibDiamond.sol

#14 SWC-135 Code With No Effect

Category	Severity	Location	Status
Coding Style	Informational	Line 109	Resolved

Description
L109 contains commented code.

Recommendation
Remove code with no effects.

Alleviation
N/A

sdPoolUtil.sol

#15 SWC-129 Typographical Error

Category	Severity	Location	Status
Block Timestamp Manipulation	Medium	Line 126, 129	Acknowledged

Description

block.timestamp can be manipulated by miners with the following constraints

- It cannot be stamped with an earlier time than its parent
- It cannot be too far in the future

Recommendation

Avoid using block.timestamp for a source and use block.number instead. or add additional time-based equations.

Alleviation

N/A

#16 SWC-135 Code With No Effect

Category	Severity	Location	Status
Coding Style	Informational	Line 219, 250, 294, 311, 326-328	Resolved

Description

Line contains commented code.

Recommendation

Remove code with no effects.

Alleviation

N/A

Diamond.sol

#17 SWC-109 Uninitialized Storage Pointer

Category	Severity	Location	Status
Coding Style	Informational	Line 48	Acknowledged

Description
Uninitialized storage variables can point to unexpected storage locations.

Recommendation
Initialize variable “ds” or set the storage attribute “memory”.

Alleviation
As part of the EIP2535 diamond pattern, the storage is initialized in the assembly block immediately following the declaration in lines 50 - 53

DiamondFactory.sol

#18 Custom Centralization Related Risk

Category	Severity	Location	Status
Centralization / Privilege	Medium	#	Acknowledged

Description
Refer to General notes - Centralization Risk.

Alleviation
• God User account is being setup as a GNOSIS safe account requiring at least 3 of 5 founder/management signatures to execute.

• Admin User is required for automated processes and keys are stored in secured infastructure

• Use of assembly is required to clone contract, no possible fixes.

• Fixed visilbility

#19 Custom Assembly Usage

Category	Severity	Location	Status
Centralization / Privilege	Low	Line 223-233	Acknowledged

Description
N/A

Recommendation
Use “require” for invariants modifying the state.
DiamondFactory._clone(address,uint256) (src/DiamondFactory.sol#223-233) uses assembly
- INLINE ASM (src/DiamondFactory.sol#225-231)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Alleviation
N/A

#20 SWC-100 Function Default Visibility

Category	Severity	Location	Status
Gas Optimization	Optimization	Line 67, 74, 83, 93, 105, 124, 187, 203, 209, 213, 238, 244	Resolved

Description
Functions that are not used internally in the function are declared as public.

Recommendation
Function visibility modifier should be changed to external for functions. This will save on gas when calling these functions as external functions do not need to save parameters to memory, unlike public functions which do.

Alleviation
N/A

slots.sol

#21 SWC-135 Code With No Effects

Category	Severity	Location	Status
Coding Style	Informational	Line 46	Resolved

Description
_slotID does not equal MAX_SLOT+1 check unrequired.

Recommendation
Remove or change if unintended functionality

Alleviation
N/A

#22 SWC-135 Code With No Effects

Category	Severity	Location	Status
Coding Style	Informational	Line 64	Resolved

Description
InactivePool reversion error message.

Recommendation
Remove or change if unintended functionality

Alleviation
N/A

CombineBeacon.sol

Alleviation
Removed ABIEncoderV2

CombineProxy.sol

#23 Custom ETH Locked

Category	Severity	Location	Status
ETH Locked	Low	Line 14 - 89	Acknowledged

Description

Contract locking ether found:
Contract combine_proxy (src/combine_proxy.sol#14-89) has payable functions:
- combine_proxy.receive() (src/combine_proxy.sol#23)
- combine_proxy.initialize(string,address,address,uint256) (src/combine_proxy.sol#32-49)
- combine_proxy.fallback() (src/combine_proxy.sol#69-88)

But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

Alleviation

• BNB/ETH is not directly stored in the contract, all funds are kept in liquidity pools
• Value is withdrawn through the liquidate function
• Line 190 value is transferred back to the owner of the contract

#24 Custom Assembly Usage

Category	Severity	Location	Status
Gas Optimization	Optimization	Line 223 - 233	Acknowledged

Description

combine_proxy.fallback() (src/combine_proxy.sol#69-88) uses assembly
- INLINE ASM (src/combine_proxy.sol#74-83)

proxyFactory.deploy(uint256,uint256) (src/combine_proxy.sol#202-220) uses assembly
- INLINE ASM (src/combine_proxy.sol#208-219)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Recommendation

N/A

Alleviation

Assembly use required in proxy pattern.

#25 Custom Boolean Equality

Category	Severity	Location	Status
Coding Style	Informational	Line 32	Resolved

Description
Booleans can be checked directly, do not require a comparison to false or true.

Recommendation
Remove the equality to the boolean constant.

Alleviation
Fixed Boolean Equality.

#26 SWC-100 Function Default Visibility

Category	Severity	Location	Status
Gas Optimization	Optimization	Line 32, 64, 112, 119, 129, 141, 171, 187, 193	Resolved

Description
Functions that are not used internally in the function are declared as public.

Recommendation
Function visibility modifier should be changed to external for functions. This will save on gas when calling these functions as external functions do not need to save parameters to memory, unlike public functions which do.

Alleviation
Fixed Visibility.

CombineApp.sol

#27 Custom Sandwich Attacks

Category	Severity	Location	Status
Volatile Code	High	Line 395	Acknowleged

Description

Potential Sandwich Attacks
A sandwich attack might happen when an attacker observes a transaction removing liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by back running (after the transaction being attacked) a transaction to sell the asset.

The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large,

Recommendation

We advise the client to use Oracle to get an estimation of prices and setting minimum amounts based on the prices when calling the aforementioned functions.

Alleviation

Sandwich Attacks

• This will need some investigation and we acknowledge the risk.
• This may require major restructuring of code as we are nearing limits imposed by the blockchain in regards to size of code.

#28 SWC-129 Typographical Error

Category	Severity	Location	Status
Volatile Code	Low	Line 261, 262, 370	Resolved

Description

In Solidity multiplication should come before division to avoid rounding errors.

Recommendation

Restructure the mathematical operation.

Alleviation

261 - multiplication is done before division
262, 370 - fixed

#29 SWC-134 Message call with hardcoded gas amount

Category	Severity	Location	Status
Coding Style	Informational	Line 45, 29	Resolved

Description

Booleans can be checked directly, do not require a comparison to false or true.

Recommendation

Remove the equality to the boolean constant.

Alleviation

Fixed boolean checks.

#30 Custom Low Level Call

Category	Severity	Location	Status
Coding Style	Low	Line 168 - 175	Acknowledged

Description

Low level call in combineApp.pendingReward(slotsLib.sSlots) (src/combineApp.sol#168-175):
- (data) = _slot.chefContract.staticcall(abi.encodeWithSignature(_slot.pendingCall,_slot.poolId,address(this))) (src/combineApp.sol#169)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Alleviation

Due to differences in the pending reward function between dex's we need to make this call dynamic
• Pancakeswap uses: pendingCake
• Biswap Uses: pendingBSW

#31 Custom Automated Re-entrancy Detection

Category	Severity	Location	Status
Coding Style	Medium	Line 212 - 222	Resolved

Recommendation

Apply check effects pattern to all states altered

Reentrancy in combineApp.harvest(uint64,string) (src/combineApp.sol#212-222):
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

Reentrancy in combineApp.addFunds(slotsLib.sSlots,uint256,bool) (src/combineApp.sol#244-267):
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Reentrancy in combineApp.addFunds(slotsLib.sSlots,uint256,bool) (src/combineApp.sol#244-267):
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Alleviation

• Function header includes a "lockFunction" modifier that reverts if the _locked state variable is true, and "allowAdmin" which restricts access to this function.
• When modifier is called, _locked is set to true, and when function is done, set to false
• Only an admin user or owner may call this function which further restricts access
• Any funds are then sent back to the contract owner.
• NOTE: all funds in this contract belong to the contract owner.

Storage.sol

#32 SWC-108 State Variable Default Visibility

Category	Severity	Location	Status
Coding Style	Informational	Line 15, 20, 28	Resolved

Description

It is best practice to set the visibility of state variables explicitly. The default visibility for state variables is internal.

Recommendation

Define state variable visibility as public, internal or private.

Alleviation

Fixed Visibility.

SimpleDefiToken

SimpleToken.sol

#33 SWC-100 Function Default Visibility

Category	Severity	Location	Status
Gas Optimization	Optimization	Line 29	Resolved

Description

Functions that are not used internally in the function are declared as public.

Recommendation

Function visibility modifier should be changed to external for functions. This will save on gas when calling these functions as external functions do not need to save parameters to memory, unlike public functions which do.

Alleviation

N/A

#34 Custom addRelease new function added

Category	Severity	Location	Status
Code Addition	Low	Line 11 - 88	Resolved

Description

A new function was added to add the ability to distribute tokens without allowing somebody to prematurely deploy an LP.

Recommendation

N/A

```
8       struct mintTo {
9           address to;
10          uint256 amount;
11  +       uint256 blocknumber;
12      }
13
14      mapping(address=>uint) public releaseAddresses;
    @@ -39,7 +40,7 @@ contract EasyToken is ERC20Capped, ERC20Burnable, ERC20Snapshot, Ownable {
40      /// @dev emits address, and block number
41      /// @param _addr         - Address of user to allow transfers
42      /// @param _blockNumber - block to allow transfers
    -   function addRelease(address _addr, uint _blockNumber) external onlyOwner {
43  +   function addRelease(address _addr, uint _blockNumber) public onlyOwner {
44          uint _rd = releaseAddresses[_addr];
45          if (_rd > 0 && _blockNumber > _rd) revert invalidBlockNumber();
46          releaseAddresses[_addr] = _blockNumber;
    @@ -68,11 +69,23 @@ contract EasyToken is ERC20Capped, ERC20Burnable, ERC20Snapshot, Ownable {
69          require(subtotal + totalSupply() <= cap(), "Total amount exceeds cap");
70          for (uint i = 0; i < _mintTo.length; i++) {
71              _mint(_mintTo[i].to, _mintTo[i].amount);
72  +           if (_mintTo[i].blocknumber > 0) {
73  +               addRelease(_mintTo[i].to,_mintTo[i].blocknumber);
74  +           }
75          }
76          emit MintRelease(address(this),subtotal);
77      }
78
79
80  +   /// @notice Transfers tokens from one address to anothera, but locks until specified block number
81  +   /// @param _to - address to transfer tokens to
82  +   /// @param _amount - amount of tokens to transfer
83  +   /// @param _blocknumber - block number until lock is released
84  +   function transfer(address _to, uint _amount, uint _blocknumber) public onlyOwner {
85  +       if (_blocknumber>0) addRelease(_to, _blocknumber);
86  +       super.transfer(_to, _amount);
87  +   }
88  +
89      /// @notice Part of OpenZeppelin contract to take snapshot of token holders
90      /// @return - returns snapshot id
91      function snapshot() public onlyOwner returns (uint){
```

#35 Custom Forge Conversion

Category	Severity	Location	Status
Code Refractor	Low	Line 13 - 61	Resolved

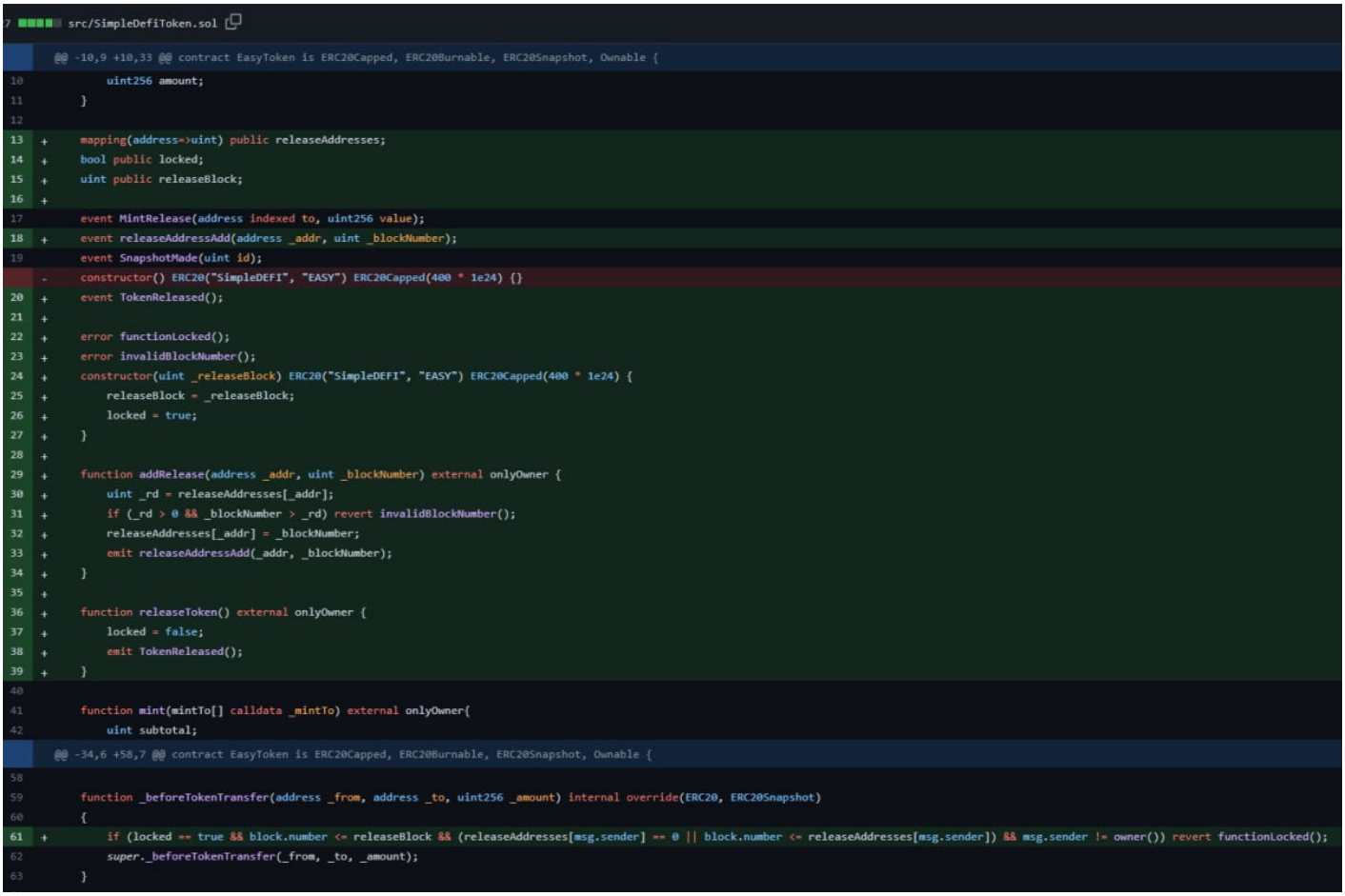
Description

The SimpleDefi code base was recractor to support Forge development. Appendix can be found here: **Forge**

Test result: **ok. 17 passed; 0 failed; finished in 4.08s**

Recommendation

N/A



```
src/SimpleDefiToken.sol
@@ -10,9 +10,33 @@ contract EasyToken is ERC20Capped, ERC20Burnable, ERC20Snapshot, Ownable {
        uint256 amount;
    }

+   mapping(address=>uint) public releaseAddresses;
+   bool public locked;
+   uint public releaseBlock;
+
    event MintRelease(address indexed to, uint256 value);
+   event releaseAddressAdd(address _addr, uint _blockNumber);
    event SnapshotMade(uint id);
-   constructor() ERC20("SimpleDEFI", "EASY") ERC20Capped(400 * 1e24) {}
+   event TokenReleased();
+
+   error functionLocked();
+   error invalidBlockNumber();
+   constructor(uint _releaseBlock) ERC20("SimpleDEFI", "EASY") ERC20Capped(400 * 1e24) {
+       releaseBlock = _releaseBlock;
+       locked = true;
+   }
+
+   function addRelease(address _addr, uint _blockNumber) external onlyOwner {
+       uint _rd = releaseAddresses[_addr];
+       if (_rd > 0 && _blockNumber > _rd) revert invalidBlockNumber();
+       releaseAddresses[_addr] = _blockNumber;
+       emit releaseAddressAdd(_addr, _blockNumber);
+   }
+
+   function releaseToken() external onlyOwner {
+       locked = false;
+       emit TokenReleased();
+   }

    function mint(mintTo[] calldata _mintTo) external onlyOwner{
        uint subtotal;
@@ -34,6 +58,7 @@ contract EasyToken is ERC20Capped, ERC20Burnable, ERC20Snapshot, Ownable {

    function _beforeTokenTransfer(address _from, address _to, uint256 _amount) internal override(ERC20, ERC20Snapshot)
    {
+       if (locked == true && block.number <= releaseBlock && (releaseAddresses[msg.sender] == 0 || block.number <= releaseAddresses[msg.sender]) && msg.sender != owner()) revert functionLocked();
        super._beforeTokenTransfer(_from, _to, _amount);
    }
```

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that actagainst the nature of decentralization, such as explicit ownership or specialized access roles incombination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrectoperations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on howblock.timestamp works.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that mayresult in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Block Timestamp

Be aware that the timestamp of the block can be manipulated by a miner.

Forge

Foundry is a blazing fast, portable and modular toolkit for Ethereum application development written in Rust.
Foundry consists of:
Forge: Ethereum testing framework (like Truffle, Hardhat and DappTools).

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement.

This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Asfalia’s prior written consent in each instance.This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Asfalia to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-freenature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. Asfalia’s position is that each company and individual are responsible for their own due diligence and continuous security. Asfalia’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Asfalia is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Project is potentially vulnerable to 3rd party failures of service - namely in the form of APIs providing the price for the currencies used by the project. Project could become at risk if these APIs provided incorrect pricing.

Audit does not claim to address any off-chain functions utilized by the project.

The firm was started by a team with over ten years of network security experience to become a global force. Our goal is to make the blockchain ecosystem as secure as possible for everyone.

With over 30 years of combined experience in the DeFi space, our team is highly dedicated to delivering a product that is as streamlined and secure as possible. Our mission is to set a new standard for security in the auditing sector, while increasing accessibility to top tier audits for all projects in the crypto space. Our dedication and passion to continuously improve the DeFi space is second to none.