QuillAudits

Audit Report

October, 2020

Contents

Introduction

This Audit Report mainly focuses on the overall security of BiffysLoveFarm Smart Contract. With this report, we have tried to ensure the reliability and correctness of their smart contract by complete and rigorous assessment of their system's architecture and the smart contract codebase.

Auditing Approach and Methodologies applied

The Quillhash team has performed rigorous testing of the project starting with analysing 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 any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

In the Unit testing Phase, we coded/conducted custom unit tests written for each function in the contract to verify that each function works as expected.

In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

- Testing the functionality of the Smart Contract to determine proper logic has been followed throughout the whole process.
- Analysing the complexity of the code in depth and detailed, manual review of the code, line-by-line.
- Deploying the code on testnet using multiple clients to run live tests.
- Analysing failure preparations to check how the Smart Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analysing the security of the on-chain data.

Audit Details

Project Name: BiffysLoveFarm
Website/Etherscan Code (Mainnet): 0x98a56c7b03eedb7acc46308611561622e51ec13a
Languages: Solidity (Smart contract)
Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Securify, Mythril, Contract Library, Slither, SmartCheck

The contract inherits from the following Opezeppelin contracts:
1. Address
2. Context
3. ERC20Mintable
4. Initializable
5. Math
6. MinterRole
7. Roles
8. ERC20
9. IERC20
10. Ownable
11. SafeMath

Any vulnerabilities exposed to these contracts will be of low severity.

Audit Goals

The focus of the audit was to verify that the Smart Contract System is secure, resilient and working according to the specifications. The audit activities can be grouped in the following three categories:

Security
Identifying security related issues within each contract and the system of contract.

Sound Architecture
Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

Code Correctness and Quality
A full review of the contract source code. The primary areas of focus include:

- Accuracy
- Readability
- Sections of code with high complexity
- Quantity and quality of test coverage

Security

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

High level severity issues

Issues on this level are critical to the smart contract's performance/ functionality and should be fixed before moving to a live environment.

Medium level severity issues

Issues on this level could potentially bring problems and should eventually be fixed.

Low level severity issues

Issues on this level are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Number of issues per severity

	Low	Medium	High	Recommendations
Open	6	3	0	10
Closed	0	0	0	0

Manual Audit

For this section the code was tested/read line by line by our developers. We also used Remix IDE's JavaScript VM and Kovan networks to test the contract functionality. The address of the contract on Kovan network: 0x6b2DDb361379D0A1177634b0318dBdA6E21EA25a

Low level severity issues

1. There are multiple pragama versions used within these contracts and are not locked as well. Solidity source files indicate the versions of the compiler they can be compiled with.

   ```
   pragma solidity ^0.5.0; // bad: compiles with 0.5.0 and above
   pragma solidity 0.5.0; // good : compiles with 0.5.0 only
   ```

   It is recommended to follow the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee.

   Different versions of Solidity are used in the contract, it is recommended to change all pragmas to 0.5.17.
   - ^0.5.0 (biffysLoveFarm.sol#70)
   - >=0.4.24<0.7.0 (biffysLoveFarm.sol#149)
   - ^0.5.0 (biffysLoveFarm.sol#214)
   - ^0.5.0 (biffysLoveFarm.sol#245)
   - ^0.5.5 (biffysLoveFarm.sol#327)
   - ^0.5.0 (biffysLoveFarm.sol#400)
   - ^0.5.0 (biffysLoveFarm.sol#432)
   - ^0.5.0 (biffysLoveFarm.sol#591)
   - ^0.5.0 (biffysLoveFarm.sol#826)
   - ^0.5.0 (biffysLoveFarm.sol#865)
   - ^0.5.0 (biffysLoveFarm.sol#916)
   - 0.5.17 (biffysLoveFarm.sol#949)

2. No need to wrap the sender within the address type for isContract call within the stake function of LPTokenWrapper contract as sender is already an address type variable.

3. Natspecs should be used to improve code readability. For instance explaining full form of LP and explaining function usability within LPTokenWrapper and BiffysLoveFarm contracts. NatSpec comments are a way to describe the behaviour of a function to end-users. It also allows us to provide more detailed information to contract readers. NatSpec includes the formatting for comments that the smart contract author will use, and which are understood by the Solidity compiler.

4. Camel case was not used to name variables at multiple occurrences. Please refer to this guideline for recommended naming conventions.

5. BiffysLoveFarm contract does not use custom events specific to the contract functionality. Events should be fired with all state variable updates as good practise. This makes it easier to build dApps on top of the contract's using existing tools. For instance no event was fired in stake function of LPTokenWrapper contract

6. The order of functions as well as the rest of the code layout does not follow solidity style guide, please read following documentation links to understand the correct order:
   - https://solidity.readthedocs.io/en/v0.5.3/style-guide.html#order-of-layout
   - https://solidity.readthedocs.io/en/v0.5.3/style-guide.html#order-of-functions

Medium level severity issues

1. Math library was not used within the LPTokenWrapper contract but was added to it.

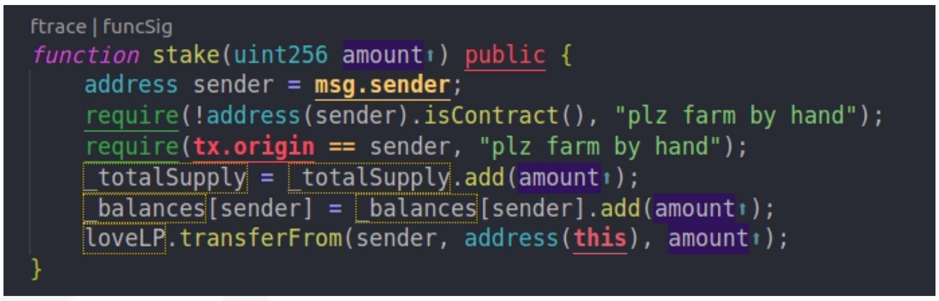
2. There is no way to take _loveLP tokens out from the contract once staked, though it is a recommended functionality it is highly recommended to include a way to withdraw user stake from the contract, as the tokens locked within the contract are rendered useless. As they cannot be transferred out of the contract in any way. Users should be explicitly made aware of this in case this functionality is not updated.

3. The following functions should be wrapped in require statements as they return boolean values on successful execution. A better way to handle this would be to use Zeppelin's safeERC20 library.

```
loveLP.transferFrom(sender, address(this), amount)
biffysLove.transfer(msg.sender,reward)
biffysLove.mint(address(this),initreward.sub(biffysLove.balanceOf(address(this))))
biffysLove.mint(address(this),initreward)
```

Recommendations

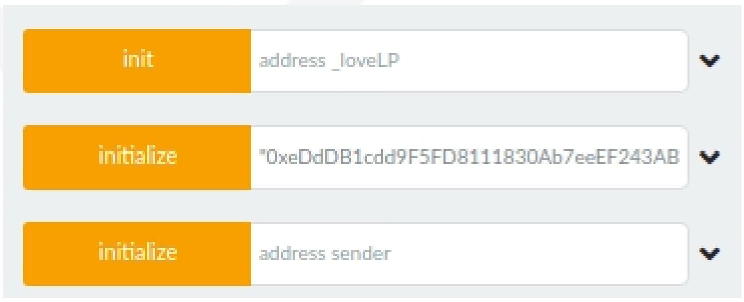
1. It is recommended to use msg.sender at all occurrences within the stake function of the LPTokenWrapper contract. It is also recommended to remove tx.origin == sender check as it is a duplication of isContract check above it. Also it is highly recommended to not use tx.origin until absolutely necessary.



```
ftrace | funcSig
function stake(uint256 amount) public {
    address sender = msg.sender;
    require(!address(sender).isContract(), "plz farm by hand");
    require(tx.origin == sender, "plz farm by hand");
    _totalSupply = _totalSupply.add(amount);
    _balances[sender] = _balances[sender].add(amount);
    loveLP.transferFrom(sender, address(this), amount);
}
```

2. The following condition: account != address(0) within updateReward modifier will never be false as always msg.sender is always passed as argument to it. It is recommended to remove it.

3. The contract has three initialising functions two have the same names. Once any one of the initialize is called, the other two are rendered useless. It is recommended to switch to constructors for LPTokenWrapper and BiffysLoveFarm to avoid confusion.



init	address _loveLP
initialize	"0xDdDB1cdd9F5FD8111830Ab7eeEF243AB
initialize	address sender

4. Functions setInitReward, fixRewardPerTokenStored and fixEmissionRate have centralised control and should be removed if not absolutely required. Overpowered owner (i.e. the owner that has too much power) is the project design where contract owner (or owners) can manually invoke critical functions of the system. As one great man said, with great power comes great responsibility. And that great power can be used in a malicious way. Here is a recommended read about overpowered owner roles.

5. It is recommended to add the following line at the beginning of contract code:
   // SPDX-License-Identifier: MIT
   SPDX reduces redundant work by providing a common format for companies and communities to share important data, thereby streamlining and improving compliance. You can read more about SPDX here.

6. It is risky to use block.timestamp. As block.timestamp can be manipulated by miners. Therefore block.timestamp is recommended to be supplemented with some other strategy in the case of high-value/risk applications.

7. The following list of functions should be declared external, to save gas during contract calls:

owner()	increaseAllowance(address,uint256)
renounceOwnership()	decreaseAllowance(address,uint256)
transferOwnership(address)	addMinter(address)
totalSupply()	renounceMinter()
balanceOf(address)	mint(address,uint256)
transfer(address,uint256)	initialize(IERC20,ERC20Mintable,uint256,uint256,uint256,address)
allowance(address,address)	getReward()
approve(address,uint256)	setCompleted(uint256)
transferFrom(address,address,uint256)	

8. All state variable changes should be done before external calls. This issue was spotted at occurrences mentioned below. This is a standard practice against reentrancy. To remove this we would suggest removing the external mint call from the checkhalve modifier and adding it in the respective functions wherever necessary.

   This is a standard design practice and is not compulsory since the external call is done to a standard ERC20 function whose code is also audited with this contract.

   External calls:
   - biffysLove.mint(address(this),initreward) (biffysLoveFarm.sol#1099)

   State variables written after the call(s):
   - periodFinish = block.timestamp.add(duration) (biffysLoveFarm.sol#1102)

   External calls:
   - checkhalve() (biffysLoveFarm.sol#1067)
   - biffysLove.mint(address(this),initreward) (biffysLoveFarm.sol#1099)

   State variables written after the call(s):
   - rewards[msg.sender] = 0 (biffysLoveFarm.sol#1070)

   External calls:
   - super.stake(amount) (biffysLoveFarm.sol#1059)
   - loveLP.transferFrom(sender,address(this),amount) (biffysLoveFarm.sol#982)
   - checkhalve() (biffysLoveFarm.sol#1057)
   - biffysLove.mint(address(this),initreward) (biffysLoveFarm.sol#1099)

   State variables written after the call(s):
   - super.stake(amount) (biffysLoveFarm.sol#1059)
   - _balances[sender] = _balances[sender].add(amount) (biffysLoveFarm.sol#981)
   - super.stake(amount) (biffysLoveFarm.sol#1059)
   - _totalSupply = _totalSupply.add(amount) (biffysLoveFarm.sol#980)
   - userRewardPerTokenPaid[msg.sender] = rewardPerToken() (biffysLoveFarm.sol#1062)

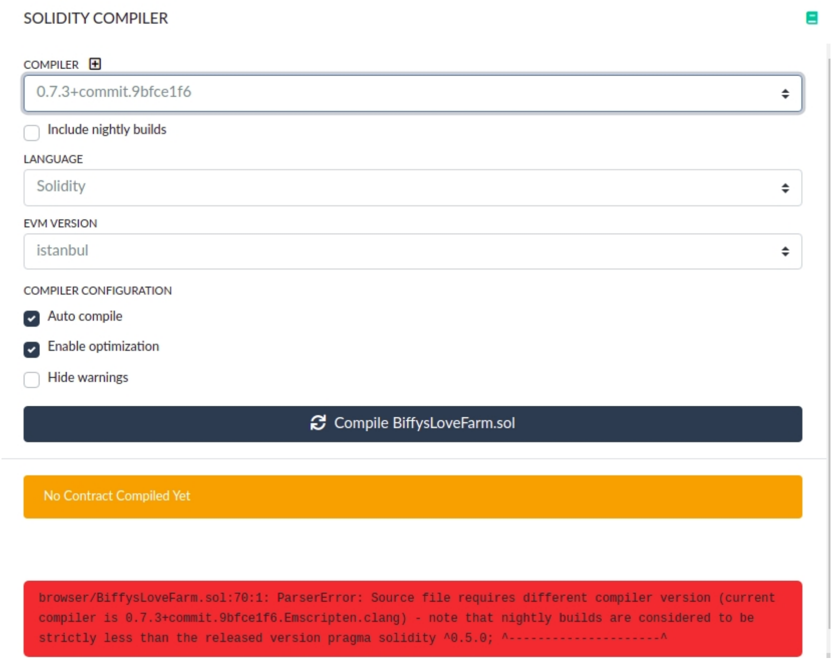
9. It is recommended to convert reward greater than zero check within the getReward function to a required statement. The failure of function execution can save user gas while trying to claim empty rewards.

10. It is recommended to use an interface of ERC20Mintable to deploy the contracts to save gas during deployment of the contracts.

Automated Audit

Remix Compiler Warnings

It throws warnings by Solidity’s compiler. If it encounters any errors the contract cannot be compiled and deployed.



SOLIDITY COMPILER

COMPILER: 0.7.3+commit.9bfce1f6

- [ ] Include nightly builds

LANGUAGE: Solidity

EVM VERSION: istanbul

COMPILER CONFIGURATION

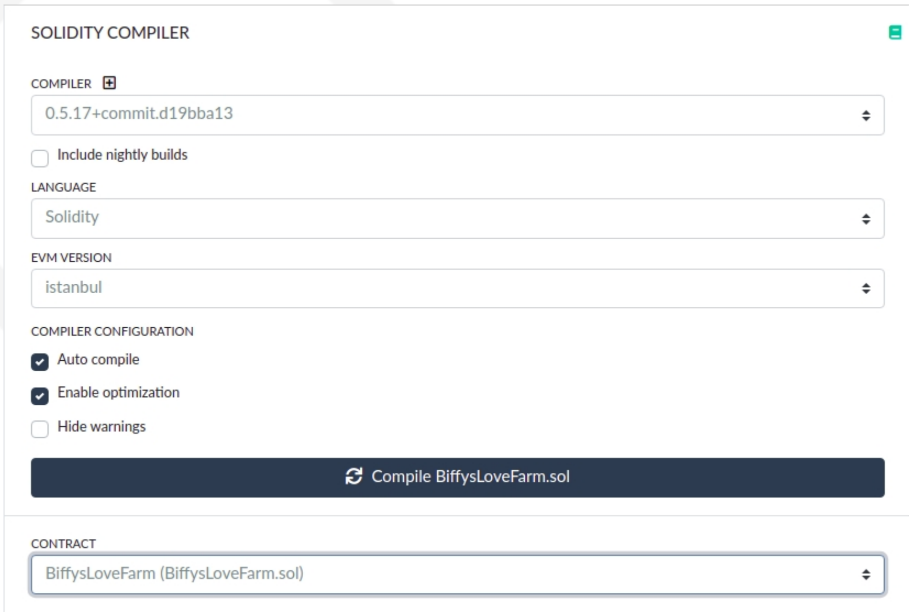
- [x] Auto compile
- [x] Enable optimization
- [ ] Hide warnings

Compile BiffysLoveFarm.sol

No Contract Compiled Yet

```
browser/BiffysLoveFarm.sol:70:1: ParserError: Source file requires different compiler version (current
compiler is 0.7.3+commit.9bfce1f6.Emscripten.clang) - note that nightly builds are considered to be
strictly less than the released version pragma solidity ^0.5.0; ^--------------------^
```

Multiple Pragma versions are used within the contract, making Remix direct to V0.7.3 which was unable to compile the contract. On explicitly setting the compiler version to v0.5.17 no error or warning was thrown and the contracts compiled with ease.



SOLIDITY COMPILER

COMPILER: 0.5.17+commit.d19bba13

- [ ] Include nightly builds

LANGUAGE: Solidity

EVM VERSION: istanbul

COMPILER CONFIGURATION

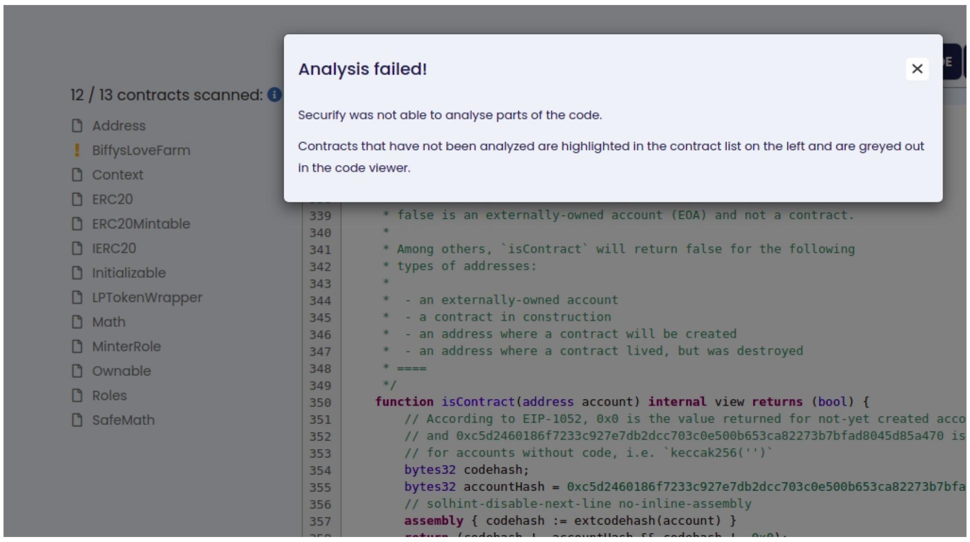
- [x] Auto compile
- [x] Enable optimization
- [ ] Hide warnings

Compile BiffysLoveFarm.sol

CONTRACT: BiffysLoveFarm (BiffysLoveFarm.sol)

Securify

Securify is a tool that scans Ethereum smart contracts for critical security vulnerabilities. Securify statically analyzes the EVM code of the smart contract to infer important semantic information (including control-flow and data-flow facts) about the contract.
Securify was unable to analyse BiffysLovefarm contract:



Analysis failed!

Securify was not able to analyse parts of the code.

Contracts that have not been analyzed are highlighted in the contract list on the left and are greyed out in the code viewer.

12 / 13 contracts scanned:

- Address
- ! BiffysLoveFarm
- Context
- ERC20
- ERC20Mintable
- IERC20
- Initializable
- LPTokenWrapper
- Math
- MinterRole
- Ownable
- Roles
- SafeMath

```
     * false is an externally-owned account (EOA) and not a contract.
     *
     * Among others, `isContract` will return false for the following
     * types of addresses:
     *
     *  - an externally-owned account
     *  - a contract in construction
     *  - an address where a contract will be created
     *  - an address where a contract lived, but was destroyed
     * ====
     */
    function isContract(address account) internal view returns (bool) {
        // According to EIP-1052, 0x0 is the value returned for not-yet created acco
        // and 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470 is
        // for accounts without code, i.e. `keccak256('')`
        bytes32 codehash;
        bytes32 accountHash = 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfa
        // solhint-disable-next-line no-inline-assembly
        assembly { codehash := extcodehash(account) }
```

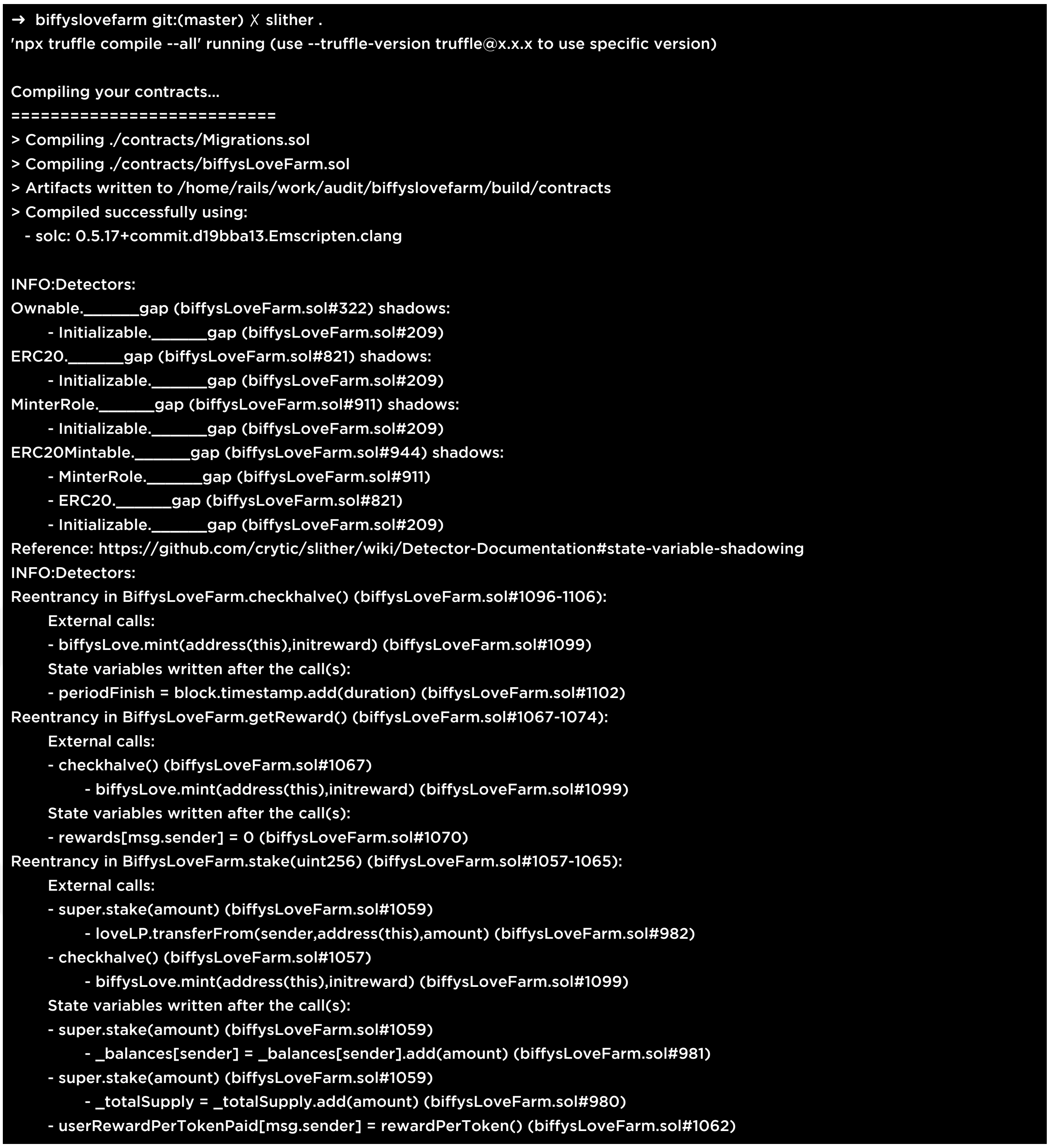
Though other contracts were analysed successfully and the report can be accessed here:
https://securify.chainsecurity.com/
report/51272fc9f9b26250539c2ea9547c7e68510120e93cb39e32a80f062616cc1ebb
All the major vulnerabilities detected by Securify are already covered in the manual audit.

SmartCheck

Smartcheck is a tool for automated static analysis of Solidity source code for security vulnerabilities and best practices. SmartCheck translates Solidity source code into an XML-based intermediate representation and checks it against XPath patterns. Smartcheck shows significant improvements over existing alternatives in terms of false discovery rate (FDR) and false negative rate (FNR). It gave the following result for the BiffysLoveFarm contract: https://tool.smartdec.net/
scan/647e0d32efd5419a8c2f55ffe74650a9.
SmartCheck did not detect any high severity issue. All the considerable issues raised by SmartCheck are already covered in the Manual Audit section of this report.

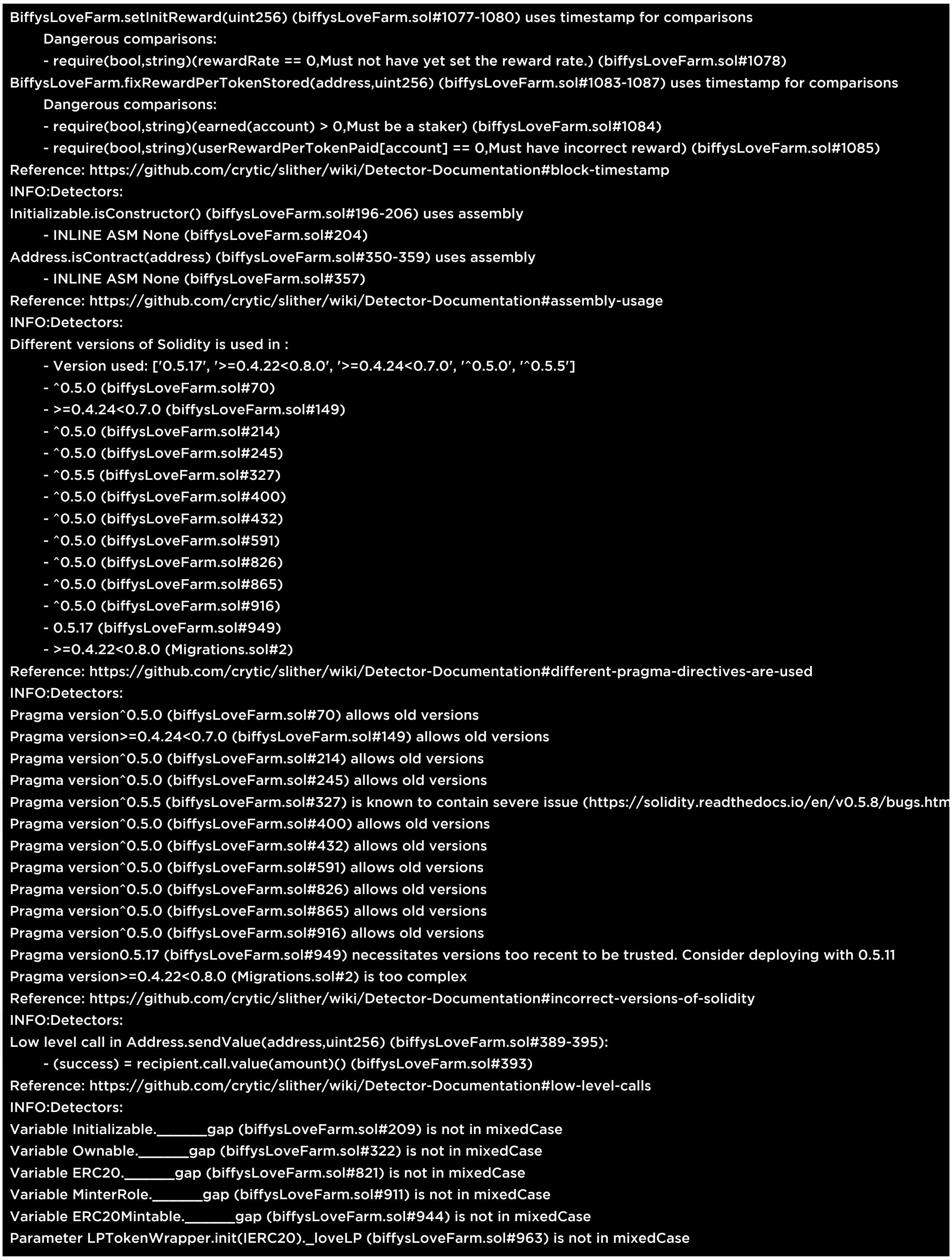
Slither

Slither, an open-source static analysis framework. This tool provides rich information about Ethereum smart contracts and has the critical properties. While Slither is built as a security-oriented static analysis framework, it is also used to enhance the user's understanding of smart contracts, assist in code reviews, and detect missing optimizations.



```
➜  biffyslovefarm git:(master) ✗ slither .
'npx truffle compile --all' running (use --truffle-version truffle@x.x.x to use specific version)

Compiling your contracts...
===========================
> Compiling ./contracts/Migrations.sol
> Compiling ./contracts/biffysLoveFarm.sol
> Artifacts written to /home/rails/work/audit/biffyslovefarm/build/contracts
> Compiled successfully using:
   - solc: 0.5.17+commit.d19bba13.Emscripten.clang

INFO:Detectors:
Ownable.______gap (biffysLoveFarm.sol#322) shadows:
        - Initializable.______gap (biffysLoveFarm.sol#209)
ERC20.______gap (biffysLoveFarm.sol#821) shadows:
        - Initializable.______gap (biffysLoveFarm.sol#209)
MinterRole.______gap (biffysLoveFarm.sol#911) shadows:
        - Initializable.______gap (biffysLoveFarm.sol#209)
ERC20Mintable.______gap (biffysLoveFarm.sol#944) shadows:
        - MinterRole.______gap (biffysLoveFarm.sol#911)
        - ERC20.______gap (biffysLoveFarm.sol#821)
        - Initializable.______gap (biffysLoveFarm.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
INFO:Detectors:
Reentrancy in BiffysLoveFarm.checkhalve() (biffysLoveFarm.sol#1096-1106):
        External calls:
        - biffysLove.mint(address(this),initreward) (biffysLoveFarm.sol#1099)
        State variables written after the call(s):
        - periodFinish = block.timestamp.add(duration) (biffysLoveFarm.sol#1102)
Reentrancy in BiffysLoveFarm.getReward() (biffysLoveFarm.sol#1067-1074):
        External calls:
        - checkhalve() (biffysLoveFarm.sol#1067)
                - biffysLove.mint(address(this),initreward) (biffysLoveFarm.sol#1099)
        State variables written after the call(s):
        - rewards[msg.sender] = 0 (biffysLoveFarm.sol#1070)
Reentrancy in BiffysLoveFarm.stake(uint256) (biffysLoveFarm.sol#1057-1065):
        External calls:
        - super.stake(amount) (biffysLoveFarm.sol#1059)
                - loveLP.transferFrom(sender,address(this),amount) (biffysLoveFarm.sol#982)
        - checkhalve() (biffysLoveFarm.sol#1057)
                - biffysLove.mint(address(this),initreward) (biffysLoveFarm.sol#1099)
        State variables written after the call(s):
        - super.stake(amount) (biffysLoveFarm.sol#1059)
                - _balances[sender] = _balances[sender].add(amount) (biffysLoveFarm.sol#981)
        - super.stake(amount) (biffysLoveFarm.sol#1059)
                - _totalSupply = _totalSupply.add(amount) (biffysLoveFarm.sol#980)
        - userRewardPerTokenPaid[msg.sender] = rewardPerToken() (biffysLoveFarm.sol#1062)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
LPTokenWrapper.stake(uint256) (biffysLoveFarm.sol#976-983) uses tx.origin for authorization: require(bool,string)
(tx.origin == sender,plz farm by hand) (biffysLoveFarm.sol#979)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-usage-of-txorigin
INFO:Detectors:
LPTokenWrapper.stake(uint256) (biffysLoveFarm.sol#976-983) ignores return value by loveLP.transferFrom(sender,address(this),
amount) (biffysLoveFarm.sol#982)
BiffysLoveFarm.getReward() (biffysLoveFarm.sol#1067-1074) ignores return value by biffysLove.transfer(msg.sender,reward)
(biffysLoveFarm.sol#1071)
BiffysLoveFarm.fixEmissionRate(uint256) (biffysLoveFarm.sol#1090-1094) ignores return value by biffysLove.mint(address(this),
initreward.sub(biffysLove.balanceOf(address(this)))) (biffysLoveFarm.sol#1092)
BiffysLoveFarm.checkhalve() (biffysLoveFarm.sol#1096-1106) ignores return value by biffysLove.mint(address(this),initreward)
(biffysLoveFarm.sol#1099)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
BiffysLoveFarm.initialize(IERC20,ERC20Mintable,uint256,uint256,uint256,address)._owner (biffysLoveFarm.sol#1010) shadows:
	- Ownable._owner (biffysLoveFarm.sol#259) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in BiffysLoveFarm.checkhalve() (biffysLoveFarm.sol#1096-1106):
	External calls:
	- biffysLove.mint(address(this),initreward) (biffysLoveFarm.sol#1099)
	State variables written after the call(s):
	- rewardRate = initreward.div(duration) (biffysLoveFarm.sol#1101)
Reentrancy in BiffysLoveFarm.fixEmissionRate(uint256) (biffysLoveFarm.sol#1090-1094):
	External calls:
	- biffysLove.mint(address(this),initreward.sub(biffysLove.balanceOf(address(this)))) (biffysLoveFarm.sol#1092)
	State variables written after the call(s):
	- rewardRate = initreward.div(periodFinish.sub(now)) (biffysLoveFarm.sol#1093)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in BiffysLoveFarm.checkhalve() (biffysLoveFarm.sol#1096-1106):
	External calls:
	- biffysLove.mint(address(this),initreward) (biffysLoveFarm.sol#1099)
	Event emitted after the call(s):
	- RewardAdded(initreward) (biffysLoveFarm.sol#1103)
Reentrancy in BiffysLoveFarm.getReward() (biffysLoveFarm.sol#1067-1074):
	External calls:
	- biffysLove.transfer(msg.sender,reward) (biffysLoveFarm.sol#1071)
	- checkhalve() (biffysLoveFarm.sol#1067)
		- biffysLove.mint(address(this),initreward) (biffysLoveFarm.sol#1099)
	Event emitted after the call(s):
	- RewardPaid(msg.sender,reward) (biffysLoveFarm.sol#1072)
Reentrancy in BiffysLoveFarm.stake(uint256) (biffysLoveFarm.sol#1057-1065):
	External calls:
	- super.stake(amount) (biffysLoveFarm.sol#1059)
		- loveLP.transferFrom(sender,address(this),amount) (biffysLoveFarm.sol#982)
	- checkhalve() (biffysLoveFarm.sol#1057)
		- biffysLove.mint(address(this),initreward) (biffysLoveFarm.sol#1099)
	Event emitted after the call(s):
	- Staked(msg.sender,amount) (biffysLoveFarm.sol#1064)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
BiffysLoveFarm.getReward() (biffysLoveFarm.sol#1067-1074) uses timestamp for comparisons
	Dangerous comparisons:
	- reward > 0 (biffysLoveFarm.sol#1069)
```



```
BiffysLoveFarm.setInitReward(uint256) (biffysLoveFarm.sol#1077-1080) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(rewardRate == 0,Must not have yet set the reward rate.) (biffysLoveFarm.sol#1078)
BiffysLoveFarm.fixRewardPerTokenStored(address,uint256) (biffysLoveFarm.sol#1083-1087) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(earned(account) > 0,Must be a staker) (biffysLoveFarm.sol#1084)
        - require(bool,string)(userRewardPerTokenPaid[account] == 0,Must have incorrect reward) (biffysLoveFarm.sol#1085)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Initializable.isConstructor() (biffysLoveFarm.sol#196-206) uses assembly
        - INLINE ASM None (biffysLoveFarm.sol#204)
Address.isContract(address) (biffysLoveFarm.sol#350-359) uses assembly
        - INLINE ASM None (biffysLoveFarm.sol#357)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used in :
        - Version used: ['0.5.17', '>=0.4.22<0.8.0', '>=0.4.24<0.7.0', '^0.5.0', '^0.5.5']
        - ^0.5.0 (biffysLoveFarm.sol#70)
        - >=0.4.24<0.7.0 (biffysLoveFarm.sol#149)
        - ^0.5.0 (biffysLoveFarm.sol#214)
        - ^0.5.0 (biffysLoveFarm.sol#245)
        - ^0.5.5 (biffysLoveFarm.sol#327)
        - ^0.5.0 (biffysLoveFarm.sol#400)
        - ^0.5.0 (biffysLoveFarm.sol#432)
        - ^0.5.0 (biffysLoveFarm.sol#591)
        - ^0.5.0 (biffysLoveFarm.sol#826)
        - ^0.5.0 (biffysLoveFarm.sol#865)
        - ^0.5.0 (biffysLoveFarm.sol#916)
        - 0.5.17 (biffysLoveFarm.sol#949)
        - >=0.4.22<0.8.0 (Migrations.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version^0.5.0 (biffysLoveFarm.sol#70) allows old versions
Pragma version>=0.4.24<0.7.0 (biffysLoveFarm.sol#149) allows old versions
Pragma version^0.5.0 (biffysLoveFarm.sol#214) allows old versions
Pragma version^0.5.0 (biffysLoveFarm.sol#245) allows old versions
Pragma version^0.5.5 (biffysLoveFarm.sol#327) is known to contain severe issue (https://solidity.readthedocs.io/en/v0.5.8/bugs.htm
Pragma version^0.5.0 (biffysLoveFarm.sol#400) allows old versions
Pragma version^0.5.0 (biffysLoveFarm.sol#432) allows old versions
Pragma version^0.5.0 (biffysLoveFarm.sol#591) allows old versions
Pragma version^0.5.0 (biffysLoveFarm.sol#826) allows old versions
Pragma version^0.5.0 (biffysLoveFarm.sol#865) allows old versions
Pragma version^0.5.0 (biffysLoveFarm.sol#916) allows old versions
Pragma version0.5.17 (biffysLoveFarm.sol#949) necessitates versions too recent to be trusted. Consider deploying with 0.5.11
Pragma version>=0.4.22<0.8.0 (Migrations.sol#2) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (biffysLoveFarm.sol#389-395):
        - (success) = recipient.call.value(amount)() (biffysLoveFarm.sol#393)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable Initializable.______gap (biffysLoveFarm.sol#209) is not in mixedCase
Variable Ownable.______gap (biffysLoveFarm.sol#322) is not in mixedCase
Variable ERC20.______gap (biffysLoveFarm.sol#821) is not in mixedCase
Variable MinterRole.______gap (biffysLoveFarm.sol#911) is not in mixedCase
Variable ERC20Mintable.______gap (biffysLoveFarm.sol#944) is not in mixedCase
Parameter LPTokenWrapper.init(IERC20)._loveLP (biffysLoveFarm.sol#963) is not in mixedCase
Parameter BiffysLoveFarm.initialize(IERC20,ERC20Mintable,uint256,uint256,uint256,address)._loveLP (biffysLoveFarm.sol#1005)
is not in mixedCase
Parameter BiffysLoveFarm.initialize(IERC20,ERC20Mintable,uint256,uint256,uint256,address)._biffysLove (biffysLoveFarm.sol#1006)
 is not in mixedCase
Parameter BiffysLoveFarm.initialize(IERC20,ERC20Mintable,uint256,uint256,uint256,address)._duration (biffysLoveFarm.sol#1007)
is not in mixedCase
Parameter BiffysLoveFarm.initialize(IERC20,ERC20Mintable,uint256,uint256,uint256,address)._initreward (biffysLoveFarm.sol#1008)
is not in mixedCase
Parameter BiffysLoveFarm.initialize(IERC20,ERC20Mintable,uint256,uint256,uint256,address)._starttime (biffysLoveFarm.sol#1009)
is not in mixedCase
Parameter BiffysLoveFarm.initialize(IERC20,ERC20Mintable,uint256,uint256,uint256,address)._owner (biffysLoveFarm.sol#1010)
is not in mixedCase
Parameter BiffysLoveFarm.fixRewardPerTokenStored(address,uint256)._rewardPerToken (biffysLoveFarm.sol#1083) is not in
mixedCase
Parameter BiffysLoveFarm.fixEmissionRate(uint256)._initReward (biffysLoveFarm.sol#1090) is not in mixedCase
Variable Migrations.last_completed_migration (Migrations.sol#6) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
ERC20Mintable.______gap (biffysLoveFarm.sol#944) is never used in ERC20Mintable (biffysLoveFarm.sol#927-945)
Ownable.______gap (biffysLoveFarm.sol#322) is never used in BiffysLoveFarm (biffysLoveFarm.sol#987-1112)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (biffysLoveFarm.sol#274-276)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (biffysLoveFarm.sol#300-303)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (biffysLoveFarm.sol#309-311)
totalSupply() should be declared external:
        - ERC20.totalSupply() (biffysLoveFarm.sol#633-635)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (biffysLoveFarm.sol#640-642)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (biffysLoveFarm.sol#652-655)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (biffysLoveFarm.sol#660-662)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (biffysLoveFarm.sol#671-674)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (biffysLoveFarm.sol#688-692)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (biffysLoveFarm.sol#706-709)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (biffysLoveFarm.sol#725-728)
addMinter(address) should be declared external:
        - MinterRole.addMinter(address) (biffysLoveFarm.sol#893-895)
renounceMinter() should be declared external:
        - MinterRole.renounceMinter() (biffysLoveFarm.sol#897-899)
mint(address,uint256) should be declared external:
        - ERC20Mintable.mint(address,uint256) (biffysLoveFarm.sol#939-942)
initialize(IERC20,ERC20Mintable,uint256,uint256,uint256,address) should be declared external:
        - BiffysLoveFarm.initialize(IERC20,ERC20Mintable,uint256,uint256,uint256,address) (biffysLoveFarm.sol#1004-1018)
getReward() should be declared external:
        - BiffysLoveFarm.getReward() (biffysLoveFarm.sol#1067-1074)
setCompleted(uint256) should be declared external:
        - Migrations.setCompleted(uint256) (Migrations.sol#16-18)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-as-external
INFO:Slither:. analyzed (14 contracts with 46 detectors), 72 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither raised one high severity issue within Zeppelin contracts but they are well tested and the issue has been taken care of. All other vulnerabilities of importance have already been covered in the manual audit section of the report.

Unit Tests

Unit test results using Open zeppelin tools:

```
T→  biffyslovefarm git:(old-test) ✗ npm test

> @ test /home/rails/work/audit/biffyslovefarm
> npx mocha --exit --recursive test --timeout 12000



  BiffysLoveFarm
    ✓ has totalSupply
    ✓ has a duration
    ✓ has init reward
    ✓ has zero balance with owner
    ✓ has loveLP
    ✓ has biffysLove
    ✓ has owner
    ✓ returns true for isOwner when called by owner
    ✓ returns false for isOwner when not called by owner
    ✓ has zero rewards for owner
    ✓ has zero earned for owner
    ✓ has zero userRewardPerTokenPaid for owner
    ✓ has lastTimeRewardApplicable
    ✓ has rewardPerToken
    ✓ has starttime
    ✓ has periodFinish
    ✓ has lastUpdateTime
    stake
      ✓ should fail calls by a contract (114ms)
      ✓ should fail when trying to stake more than the approved amount (86ms)
      ✓ should fail with zero amount (54ms)
      ✓ should update totalSupply and balance (109ms)
      ✓ should update initreward (94ms)
      ✓ should update rewardRate (82ms)
      ✓ should emit RewardAdded (71ms)
      ✓ should emit Staked event (48ms)

  ERC20Mintable
    ✓ has totalSupply
    ✓ has balance with owner
    ✓ has owner as a minter
    total supply
```

```
   ✓ returns the total amount of tokens
 balanceOf
   when the requested account has no tokens
     ✓ returns zero
   when the requested account has some tokens
     ✓ returns the total amount of tokens
 transfer
   when the recipient is not the zero address
     when the sender does not have enough balance
       ✓ reverts
     when the sender transfers all balance
       ✓ transfers the requested amount (64ms)
       ✓ emits a transfer event
     when the sender transfers zero tokens
       ✓ transfers the requested amount (68ms)
       ✓ emits a transfer event
   when the recipient is the zero address
     ✓ reverts (51ms)
 transfer from
   when the token owner is not the zero address
     when the recipient is not the zero address
       when the spender has enough approved balance
         when the token owner has enough balance
           ✓ transfers the requested amount (50ms)
           ✓ decreases the spender allowance (60ms)
           ✓ emits a transfer event
           ✓ emits an approval event
         when the token owner does not have enough balance
           ✓ reverts
       when the spender does not have enough approved balance
         when the token owner has enough balance
           ✓ reverts
         when the token owner does not have enough balance
           ✓ reverts
     when the recipient is the zero address
       ✓ reverts
   when the token owner is the zero address
     ✓ reverts
 approve
   when the spender is not the zero address
     when the sender has enough balance
       ✓ emits an approval event
       when there was no approved amount before
         ✓ approves the requested amount (60ms)
       when the spender had an approved amount
         ✓ approves the requested amount and replaces the previous one (42ms)
     when the sender does not have enough balance
       ✓ emits an approval event
       when there was no approved amount before
         ✓ approves the requested amount (49ms)
       when the spender had an approved amount
         ✓ approves the requested amount and replaces the previous one (51ms)
   when the spender is the zero address
     ✓ reverts
 mint
   when the sender has minting permission
     when token minting is not finished for a zero amount
```

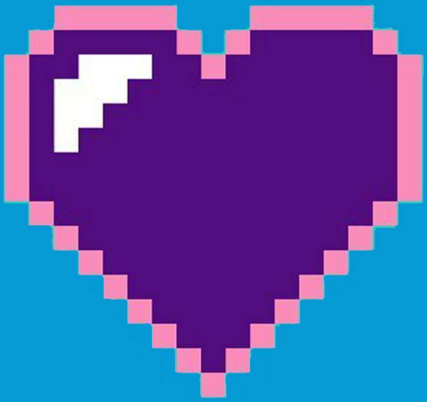
```
          ✓ mints the requested amount
          ✓ emits a mint and a transfer event
        for a non-zero amount
          ✓ mints the requested amount
          ✓ emits a mint and a transfer event
    when the sender doesn't have minting permission
      ✓ reverts

LPTokenWrapper
  ✓ has totalSupply
  ✓ has zero balance with owner
  ✓ has loveLP
  stake
    ✓ should fail calls by a contract (82ms)
    ✓ should fail when trying to stake more than the approved amount (41ms)
    ✓ should update totalSupply and balance (77ms)


64 passing (16s)
```

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the BiffysLoveFarm contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

Summary

Altogether, the code is written and demonstrates effective use of abstraction, separation of concerns, and modularity. The contract has no high severity issue but there are a number of vulnerabilities / recommendations which could be tackled in various other severity levels, and it is recommended to fix the medium severity issues before



QuillAudits

448-A EnKay Square, Opposite Cyber Hub,
Gurugram, Harayana, India - 122016

audits.quillhash.com

hello@quillhash.com