Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: EasyFarm Protocol
Platform: Binance Smart Chain
Website: easyfarm.me
Language: Solidity
Date: December 6th, 2021

Table of contents

Email: audit@EtherAuthority.io

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the EasyFarm team to perform the Security audit of the EasyFarm Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on December 6th, 2021.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

EasyFarm protocol is a smart contract, having functionality like mint, burn, add new pool, claim, etc.

Audit scope

Name	**Code Review and Security Analysis Report for EasyFarm Protocol Smart Contracts**
Platform	**BSC / Solidity**
File 1	EasyFarmToken.sol
File 1 MD5 Hash	11455D11E43557E3686F46A260AC6FCC
File 2	EasyFarmCore.sol
File 2 MD5 Hash	9606A469A5E31D6FEA354D4DF4059A44
Updated File 2 MD5 Hash	857B822BB4032701635A9D6F7EB69603
File 3	LaunchEFT.sol
File 3 MD5 Hash	8A3B1B55F008640E279248D3799CDADA
Updated File 3 MD5 Hash	97497A654FB21803BD28084C14B96C0C
Audit Date	December 6th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 EasyFarmToken.sol ● Decimal: 18 ● The EasyFarmToken Owner can access functions like: mint.	**YES, This is valid.**
File 2 EasyFarmCore.sol ● The EasyFarmCore Owner can set StartBlock, LockBlock, Strategy. ● The EasyFarmCore Auth can set dev and market addresses. ● The EasyFarmCore Gov can set rewards per block, EarnThreshold, DevPercents, MarketPercents, Gover, etc.	**YES, This is valid.**
File 3 LaunchEFT.sol ● The LaunchEFT Owner can access the set EndBlock , LockBlock, MinSupport, ExtraInfo, TotalLaunch, InviteInfo, withdraw,etc.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. This token contract does contain owner control, which does not make it fully decentralized.

We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 3 low and some very low level issues. These issues are not critical ones.
Major issues have been resolved/acknowledged in updated code.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 3 smart contract files. Smart contracts contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in EasyFarm Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the EasyFarm Protocol.

The EasyFarm Protocol team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **not** well commented on smart contracts.

Documentation

We were given an EasyFarm Protocol smart contracts code in the form of a code.The hash of that code is mentioned above in the table.

As mentioned above, code parts are **not well** commented. So it is not easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://easyfarm.me which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

EasyFarmToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyMinter	modifier	Passed	No Issue
3	mint	write	access only Minter	No Issue
4	burn	write	Passed	No Issue
5	totalBurned	write	Passed	No Issue
6	setMinter	write	Critical operation lacks event log	Refer Audit Findings
7	_transfer	internal	Passed	No Issue
8	delegates	external	Passed	No Issue
9	delegate	external	Passed	No Issue
10	delegateBySig	external	Critical operation lacks event log	Refer Audit Findings
11	getCurrentVotes	external	Passed	No Issue
12	getPriorVotes	external	Passed	No Issue
13	_delegate	internal	Passed	No Issue
14	_moveDelegates	internal	Passed	No Issue
15	_writeCheckpoint	internal	Passed	No Issue
16	safe32	internal	Passed	No Issue
17	getChainId	internal	Passed	No Issue
18	name	read	Passed	No Issue
19	symbol	read	Passed	No Issue
20	decimals	read	Passed	No Issue
21	totalSupply	read	Passed	No Issue
22	balanceOf	read	Passed	No Issue
23	transfer	write	Passed	No Issue
24	allowance	read	Passed	No Issue
25	approve	write	Passed	No Issue
26	transferFrom	write	Passed	No Issue
27	increaseAllowance	write	Passed	No Issue
28	decreaseAllowance	write	Passed	No Issue
29	_transfer	internal	Passed	No Issue
30	_mint	internal	Passed	No Issue
31	_burn	internal	Passed	No Issue
32	_approve	internal	Passed	No Issue
33	_beforeTokenTransfer	internal	Passed	No Issue
34	_afterTokenTransfer	internal	Passed	No Issue
35	owner	read	Passed	No Issue
36	onlyOwner	modifier	Passed	No Issue
37	renounceOwnership	write	access only Owner	No Issue

38	transferOwnership	write	access only Owner	No Issue
39	_transferOwnership	internal	Passed	No Issue

EasyFarmCore.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	nonReentrant	modifier	Passed	No Issue
3	owner	read	Passed	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	onlyAuth	modifier	Passed	No Issue
9	onlyGov	modifier	Passed	No Issue
10	validatePid	modifier	Passed	No Issue
11	receive	external	Passed	No Issue
12	add	write	Passed	No Issue
13	deposit	write	Passed	No Issue
14	withdraw	write	Passed	No Issue
15	withdrawAll	write	Passed	No Issue
16	claim	write	Passed	No Issue
17	claimAll	write	Infinite Loop	Refer Audit Findings
18	_claim	internal	Passed	No Issue
19	_deposit	internal	Passed	No Issue
20	_withdraw	internal	Passed	No Issue
21	earn	write	Passed	No Issue
22	updatePool	write	Passed	No Issue
23	pendingAll	read	Infinite Loop	Refer Audit Findings
24	pendingReward	read	Passed	No Issue
25	_incReward	internal	Passed	No Issue
26	getPoolLength	external	Passed	No Issue
27	getPoolInfo	external	Passed	No Issue
28	getUserInfo	external	Passed	No Issue
29	getTotalRewardPerBlock	external	Passed	No Issue
30	setStartBlock	external	access only Owner	No Issue
31	setLockBlock	external	access only Owner	No Issue
32	setStrategy	external	Passed	No Issue
33	setRewardPerBlock	write	access only Gov	No Issue

34	setEarnThreshold	write	Passed	No Issue
35	setDevPercents	write	Critical operation lacks event log	Refer Audit Findings
36	emergencyWithdrawStrategy	write	Passed	No Issue
37	setMarketPercents	write	access only Gov	No Issue
38	setGover	external	access only Gov	No Issue
39	setDev	write	access only Auth	No Issue
40	setMarket	write	access only Auth	No Issue
41	_safeEFTTransfer	internal	Passed	No Issue

LaunchEFT.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	Passed	No Issue
5	transferOwnership	write	Passed	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	support	write	Passed	No Issue
8	claim	write	Passed	No Issue
9	pending	read	Passed	No Issue
10	burnRemain	read	Passed	No Issue
11	getPrice	read	Passed	No Issue
12	getUserReferral	external	Passed	No Issue
13	getUserInfo	external	Passed	No Issue
14	getLaunchInfos	external	Passed	No Issue
15	setEndBlock	external	access only Owner	No Issue
16	setLockBlock	external	access only Owner	No Issue
17	setMinSupport	external	access only Owner	No Issue
18	setExtraInfo	external	access only Owner	No Issue
19	setTotalLaunch	external	access only Owner	No Issue
20	setInviteInfo	external	access only Owner	No Issue
21	withdraw	external	access only Owner	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Possible gas consuming loop / Infinite Loop: **EasyFarmCore.sol**

In claimAll, pendingAll functions for loops do not have poolInfo length limit , which costs more gas.

Resolution: We suggest setting the upper bound in for loops.

Status: Acknowledged.

(2) Function input parameters lack of check: **EasyFarmCore.sol**

Variable validation is not performed in below functions:

- deposit = _amount
- withdraw = _amount

Resolution: We advise to put validation like:

- integer type, variable should be greater than 0
- address type variable should not be address(0).

Status: Fixed.

(3) Transfer 0 token:

EasyFarmCore.sol

In the _withdraw function, 0 tokens are transferred.

LaunchEFT.sol

In the claim and burnRemain functions, 0 tokens are transferred.

Resolution: We suggest checking the amount should be greater than 0 before transfer.

Status: Fixed.

Very Low / Informational / Best practices:

(1) Critical operation lacks event log:

Some event need to log:

EasyFarmToken.sol

- setMinter
- delegateBySig

EasyFarmCore.sol

- add
- setStrategy
- emergencyWithdrawStrategy
- setDevPercents
- setEarnThreshold

Resolution: We suggest writing an event log for listed events.

Status: Fixed. Added logs for required events add, setStrategy, emergencyWithdrawStrategy, setEarnThreshold

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- mint: The EasyFarmToken Minter can access _mint().
- setMinter: The EasyFarmToken owner can set minter status.
- add: The EasyFarmCore owner can add a new pool.
- setStartBlock: The EasyFarmCore owner can set the start block.
- setLockBlock: The EasyFarmCore owner can set lock blocks.
- setStrategy: The EasyFarmCore owner can set strategy addresses.
- setRewardPerBlock: The EasyFarmCore Gov owner can set rewards per block.
- setEarnThreshold: The EasyFarmCore Gov owner can set an earn threshold.
- setDevPercents: The EasyFarmCore Gov owner can set dev percentages.
- setMarketPercents: The EasyFarmCore Gov owner can set market percentage.
- setGover: The EasyFarmCore Gov owner can set gover address.
- setDev: The EasyFarmCore Auth owner can set the dev address.
- setMarket: The EasyFarmCore Auth owner can set the market address.
- setEndBlock: The LaunchEFT owner can set the end block.
- setLockBlock: The LaunchEFT owner can set the lock block.
- setMinSupport: The LaunchET owner can set min support.
- setExtraInfo: The LaunchET owner can set extra information of _extraRemain and _extraRate.
- setTotalLaunch: The LaunchET owner can set _totalLaunch.
- setInviteInfo: The LaunchET owner can set invite information of level and discount.
- withdraw: The LaunchET owner can withdraw balance.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts, but they are not critical ones and some of them have been resolved. So, **it's good to go to production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured"**.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

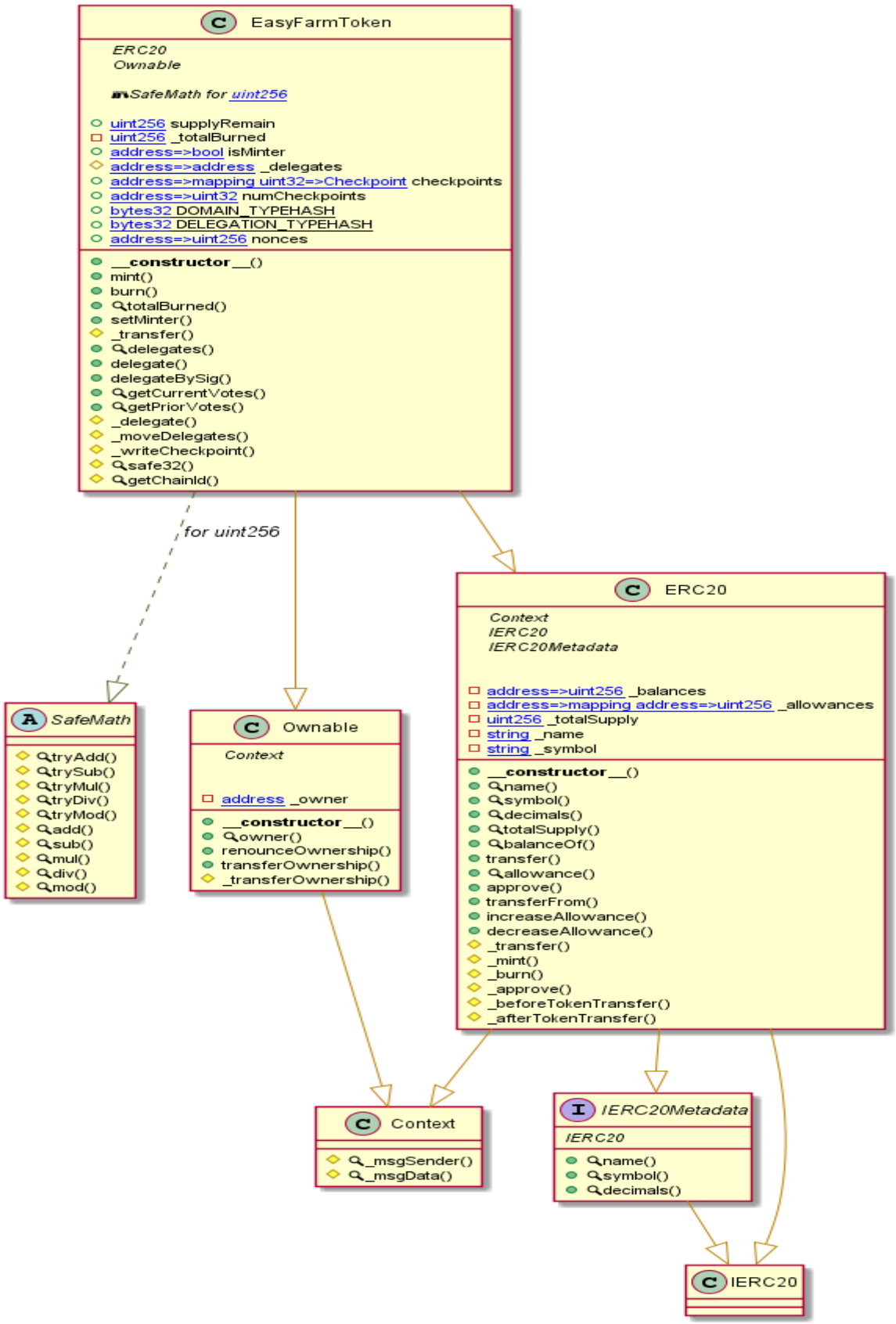
Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

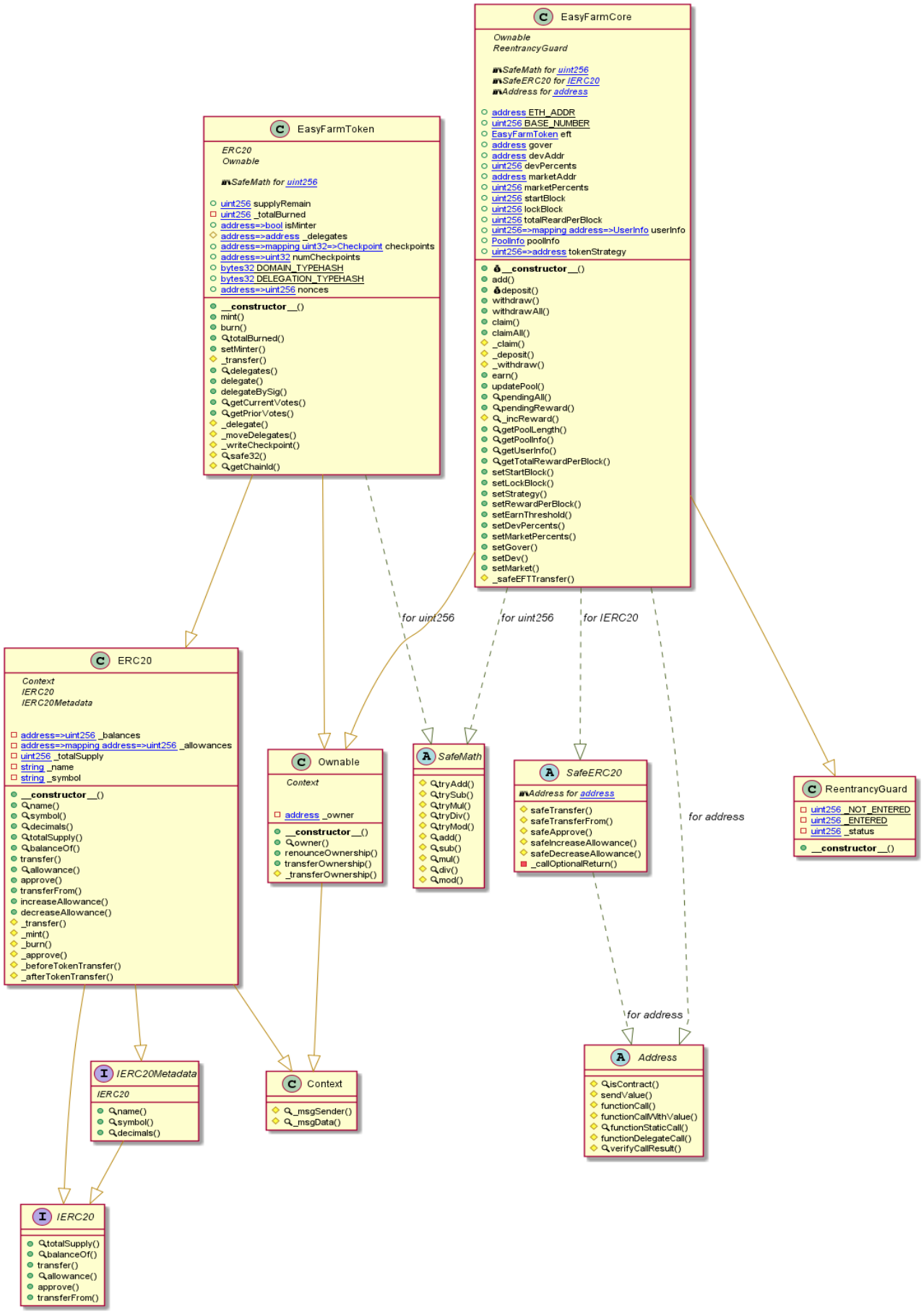
Appendix

Code Flow Diagram - EasyFarm Protocol

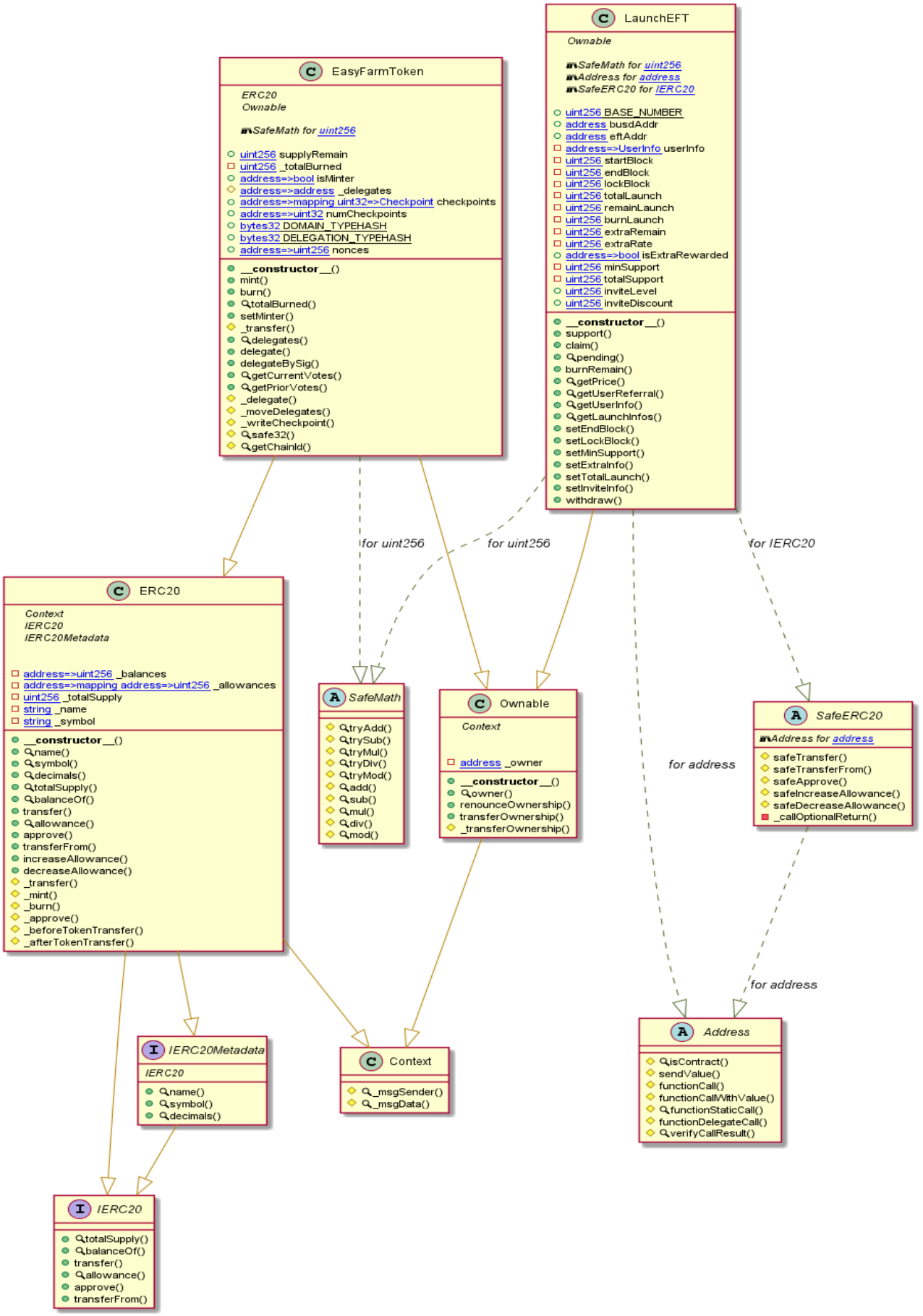
EasyFarmToken Diagram



EasyFarmCore Diagram



LaunchEFT Diagram



Slither Results Log

Slither log >> EasyFarmToken.sol

```
INFO:Detectors:
EasyFarmToken._writeCheckpoint(address,uint32,uint256,uint256) (EasyFarmToken.sol#949-967) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (EasyFarmToken.sol#959)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
EasyFarmToken.constructor(string,string,uint256)._name (EasyFarmToken.sol#705) shadows:
        - ERC20._name (EasyFarmToken.sol#383) (state variable)
EasyFarmToken.constructor(string,string,uint256)._symbol (EasyFarmToken.sol#706) shadows:
        - ERC20._symbol (EasyFarmToken.sol#384) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
EasyFarmToken.burn(uint256) (EasyFarmToken.sol#734-737) should emit an event for:
        - _totalBurned = _totalBurned.add(_amount) (EasyFarmToken.sol#736)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
EasyFarmToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (EasyFarmToken.sol#812-853) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,signature expired) (EasyFarmToken.sol#851)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
EasyFarmToken.getChainId() (EasyFarmToken.sol#974-978) uses assembly
        - INLINE ASM (EasyFarmToken.sol#976)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
EasyFarmToken.onlyMinter() (EasyFarmToken.sol#714-717) compares to a boolean constant:
        -require(bool,string)(isMinter[msg.sender] == true,permission denied) (EasyFarmToken.sol#715)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Context._msgData() (EasyFarmToken.sol#296-298) is never used and should be removed
EasyFarmToken._transfer(address,address,uint256) (EasyFarmToken.sol#748-751) is never used and should be removed
SafeMath.div(uint256,uint256) (EasyFarmToken.sol#197-199) is never used and should be removed
SafeMath.div(uint256,uint256,string) (EasyFarmToken.sol#253-262) is never used and should be removed
SafeMath.mod(uint256,uint256) (EasyFarmToken.sol#213-215) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (EasyFarmToken.sol#279-288) is never used and should be removed
SafeMath.mul(uint256,uint256) (EasyFarmToken.sol#183-185) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (EasyFarmToken.sol#230-239) is never used and should be removed
```

```
SafeMath.sub(uint256,uint256,string) (EasyFarmToken.sol#230-239) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (EasyFarmToken.sol#84-90) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (EasyFarmToken.sol#126-131) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (EasyFarmToken.sol#138-143) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (EasyFarmToken.sol#109-119) is never used and should be removed
SafeMath.trySub(uint256,uint256) (EasyFarmToken.sol#97-102) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (EasyFarmToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter EasyFarmToken.mint(address,uint256)._to (EasyFarmToken.sol#719) is not in mixedCase
Parameter EasyFarmToken.mint(address,uint256)._amount (EasyFarmToken.sol#719) is not in mixedCase
Parameter EasyFarmToken.burn(uint256)._amount (EasyFarmToken.sol#734) is not in mixedCase
Parameter EasyFarmToken.setMinter(address,bool)._minter (EasyFarmToken.sol#743) is not in mixedCase
Parameter EasyFarmToken.setMinter(address,bool)._status (EasyFarmToken.sol#743) is not in mixedCase
Variable EasyFarmToken._delegates (EasyFarmToken.sol#754) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (EasyFarmToken.sol#335-337)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (EasyFarmToken.sol#343-346)
symbol() should be declared external:
        - ERC20.symbol() (EasyFarmToken.sol#411-413)
decimals() should be declared external:
        - ERC20.decimals() (EasyFarmToken.sol#428-430)
totalSupply() should be declared external:
        - ERC20.totalSupply() (EasyFarmToken.sol#435-437)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (EasyFarmToken.sol#454-457)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (EasyFarmToken.sol#462-464)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (EasyFarmToken.sol#473-476)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (EasyFarmToken.sol#491-505)
```

```
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (EasyFarmToken.sol#491-505)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (EasyFarmToken.sol#519-522)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (EasyFarmToken.sol#538-546)
burn(uint256) should be declared external:
        - EasyFarmToken.burn(uint256) (EasyFarmToken.sol#734-737)
totalBurned() should be declared external:
        - EasyFarmToken.totalBurned() (EasyFarmToken.sol#739-741)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:EasyFarmToken.sol analyzed (7 contracts with 75 detectors), 41 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> EasyFarmCore.sol

```
INFO:Detectors:
EasyFarmCore.earn(uint256) (EasyFarmCore.sol#1514-1535) sends eth to arbitrary user
        Dangerous calls:
        - address(strategyAddr).transfer(bal) (EasyFarmCore.sol#1528)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
EasyFarmCore._safeEFTTransfer(address,uint256) (EasyFarmCore.sol#1677-1684) ignores return value by eft.transfer(_to,eftBal) (EasyFarmCor
e.sol#1680)
EasyFarmCore._safeEFTTransfer(address,uint256) (EasyFarmCore.sol#1677-1684) ignores return value by eft.transfer(_to,_amount) (EasyFarmCo
re.sol#1682)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
EasyFarmToken._writeCheckpoint(address,uint32,uint256,uint256) (EasyFarmCore.sol#1292-1310) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (EasyFarmCore.sol#1302)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in EasyFarmCore._deposit(address,uint256,uint256) (EasyFarmCore.sol#1471-1485):
        External calls:
        - _claim(_userAddr,_pid) (EasyFarmCore.sol#1472)
                - eft.transfer(_to,eftBal) (EasyFarmCore.sol#1680)
                - eft.transfer(_to,_amount) (EasyFarmCore.sol#1682)
                - eft.mint(address(this),incReward) (EasyFarmCore.sol#1547)
                - eft.mint(devAddr,devReward) (EasyFarmCore.sol#1550)
                - eft.mint(marketAddr,marketReward) (EasyFarmCore.sol#1552)
                - IEasyFarmStrategy(strategyAddr).claim(pool.depositToken) (EasyFarmCore.sol#1466)
        - IERC20(pool.depositToken).safeTransferFrom(_userAddr,address(this),_amount) (EasyFarmCore.sol#1477)
        State variables written after the call(s):
        - pool.totalDeposited = pool.totalDeposited.add(_amount) (EasyFarmCore.sol#1479)
        - user.amount = user.amount.add(_amount) (EasyFarmCore.sol#1480)
        - user.rewardDebt = user.amount.mul(pool.accTokenPerShare).div(BASE_NUMBER) (EasyFarmCore.sol#1481)
Reentrancy in EasyFarmCore._withdraw(address,uint256,uint256) (EasyFarmCore.sol#1487-1512):
        External calls:
        - _claim(_userAddr,_pid) (EasyFarmCore.sol#1488)
                - eft.transfer(_to,eftBal) (EasyFarmCore.sol#1680)
                - eft.transfer(_to,_amount) (EasyFarmCore.sol#1682)
                - eft.mint(address(this),incReward) (EasyFarmCore.sol#1547)
                - eft.mint(devAddr,devReward) (EasyFarmCore.sol#1550)
                - eft.mint(marketAddr,marketReward) (EasyFarmCore.sol#1552)
```

```
                - IEasyFarmStrategy(strategyAddr).claim(pool.depositToken) (EasyFarmCore.sol#1466)
        State variables written after the call(s):
        - pool.totalDeposited = pool.totalDeposited.sub(_amount) (EasyFarmCore.sol#1494)
        - user.amount = user.amount.sub(_amount) (EasyFarmCore.sol#1495)
        - user.rewardDebt = user.amount.mul(pool.accTokenPerShare).div(BASE_NUMBER) (EasyFarmCore.sol#1496)
Reentrancy in EasyFarmCore.setRewardPerBlock(uint256,uint256,bool) (EasyFarmCore.sol#1642-1650):
        External calls:
        - updatePool(_pid) (EasyFarmCore.sol#1644)
                - eft.mint(address(this),incReward) (EasyFarmCore.sol#1547)
                - eft.mint(devAddr,devReward) (EasyFarmCore.sol#1550)
                - eft.mint(marketAddr,marketReward) (EasyFarmCore.sol#1552)
        State variables written after the call(s):
        - pool.rewardPerBlock = _rewardPerBlock (EasyFarmCore.sol#1648)
Reentrancy in EasyFarmCore.setStrategy(uint256,address,address) (EasyFarmCore.sol#1633-1640):
        External calls:
        - IEasyFarmStrategy(_oldStrategy).withdrawAll(_tokenAddr) (EasyFarmCore.sol#1637)
        State variables written after the call(s):
        - tokenStrategy[_pid] = _strategyAddr (EasyFarmCore.sol#1639)
Reentrancy in EasyFarmCore.updatePool(uint256) (EasyFarmCore.sol#1537-1558):
        External calls:
        - eft.mint(address(this),incReward) (EasyFarmCore.sol#1547)
        - eft.mint(devAddr,devReward) (EasyFarmCore.sol#1550)
        - eft.mint(marketAddr,marketReward) (EasyFarmCore.sol#1552)
        State variables written after the call(s):
        - pool.accTokenPerShare = pool.accTokenPerShare.add(incReward.mul(BASE_NUMBER).div(pool.totalDeposited)) (EasyFarmCore.sol#1553-1
555)
        - pool.lastUpdateBlock = block.number (EasyFarmCore.sol#1556)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

```
INFO:Detectors:
EasyFarmToken.constructor(string,string,uint256)._name (EasyFarmCore.sol#1048) shadows:
        - ERC20._name (EasyFarmCore.sol#726) (state variable)
EasyFarmToken.constructor(string,string,uint256)._symbol (EasyFarmCore.sol#1049) shadows:
        - ERC20._symbol (EasyFarmCore.sol#727) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
EasyFarmCore.setGover(address) (EasyFarmCore.sol#1665-1667) should emit an event for:
        - gover = _gover (EasyFarmCore.sol#1666)
EasyFarmCore.setDev(address) (EasyFarmCore.sol#1669-1671) should emit an event for:
        - devAddr = _devAddr (EasyFarmCore.sol#1670)
EasyFarmCore.setMarket(address) (EasyFarmCore.sol#1673-1675) should emit an event for:
        - marketAddr = _marketAddr (EasyFarmCore.sol#1674)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control
INFO:Detectors:
EasyFarmToken.burn(uint256) (EasyFarmCore.sol#1077-1080) should emit an event for:
        - _totalBurned = _totalBurned.add(_amount) (EasyFarmCore.sol#1079)
EasyFarmCore.setRewardPerBlock(uint256,uint256,bool) (EasyFarmCore.sol#1642-1650) should emit an event for:
        - totalReardPerBlock = totalReardPerBlock.sub(pool.rewardPerBlock) (EasyFarmCore.sol#1647)
        - totalReardPerBlock = totalReardPerBlock.add(pool.rewardPerBlock) (EasyFarmCore.sol#1649)
EasyFarmCore.setDevPercents(uint256) (EasyFarmCore.sol#1657-1659) should emit an event for:
        - devPercents = _devPercents (EasyFarmCore.sol#1658)
EasyFarmCore.setMarketPercents(uint256) (EasyFarmCore.sol#1661-1663) should emit an event for:
        - marketPercents = _marketPercents (EasyFarmCore.sol#1662)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
EasyFarmCore.constructor(EasyFarmToken,address,address,address,uint256,uint256,uint256,uint256)._gover (EasyFarmCore.sol#1378) lacks a ze
ro-check on :
                - gover = _gover (EasyFarmCore.sol#1387)
EasyFarmCore.constructor(EasyFarmToken,address,address,address,uint256,uint256,uint256,uint256)._devAddr (EasyFarmCore.sol#1379) lacks a
zero-check on :
                - devAddr = _devAddr (EasyFarmCore.sol#1388)
EasyFarmCore.constructor(EasyFarmToken,address,address,address,uint256,uint256,uint256,uint256)._marketAddr (EasyFarmCore.sol#1380) lacks
 a zero-check on :
                - marketAddr = _marketAddr (EasyFarmCore.sol#1389)
EasyFarmCore.setGover(address)._gover (EasyFarmCore.sol#1665) lacks a zero-check on :
                - gover = _gover (EasyFarmCore.sol#1666)
EasyFarmCore.setDev(address)._devAddr (EasyFarmCore.sol#1669) lacks a zero-check on :
```

```
                - devAddr = _devAddr (EasyFarmCore.sol#1670)
EasyFarmCore.setMarket(address)._marketAddr (EasyFarmCore.sol#1673) lacks a zero-check on :
                - marketAddr = _marketAddr (EasyFarmCore.sol#1674)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in EasyFarmCore._claim(address,uint256) (EasyFarmCore.sol#1453-1469):
        External calls:
        - updatePool(_pid) (EasyFarmCore.sol#1454)
                - eft.mint(address(this),incReward) (EasyFarmCore.sol#1547)
                - eft.mint(devAddr,devReward) (EasyFarmCore.sol#1550)
                - eft.mint(marketAddr,marketReward) (EasyFarmCore.sol#1552)
        State variables written after the call(s):
        - user.locked = user.locked.add(pending).sub(released) (EasyFarmCore.sol#1458)
        - user.lastRewardBlock = block.number (EasyFarmCore.sol#1459)
        - user.rewardDebt = user.amount.mul(pool.accTokenPerShare).div(BASE_NUMBER) (EasyFarmCore.sol#1460)
Reentrancy in EasyFarmCore.setRewardPerBlock(uint256,uint256,bool) (EasyFarmCore.sol#1642-1650):
        External calls:
        - updatePool(_pid) (EasyFarmCore.sol#1644)
                - eft.mint(address(this),incReward) (EasyFarmCore.sol#1547)
                - eft.mint(devAddr,devReward) (EasyFarmCore.sol#1550)
                - eft.mint(marketAddr,marketReward) (EasyFarmCore.sol#1552)
        State variables written after the call(s):
        - totalReardPerBlock = totalReardPerBlock.sub(pool.rewardPerBlock) (EasyFarmCore.sol#1647)
        - totalReardPerBlock = totalReardPerBlock.add(pool.rewardPerBlock) (EasyFarmCore.sol#1649)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in EasyFarmCore._claim(address,uint256) (EasyFarmCore.sol#1453-1469):
        External calls:
        - updatePool(_pid) (EasyFarmCore.sol#1454)
                - eft.mint(address(this),incReward) (EasyFarmCore.sol#1547)
                - eft.mint(devAddr,devReward) (EasyFarmCore.sol#1550)
                - eft.mint(marketAddr,marketReward) (EasyFarmCore.sol#1552)
        - _safeEFTTransfer(_userAddr,released) (EasyFarmCore.sol#1462)
                - eft.transfer(_to,eftBal) (EasyFarmCore.sol#1680)
                - eft.transfer(_to,_amount) (EasyFarmCore.sol#1682)
        - IEasyFarmStrategy(strategyAddr).claim(pool.depositToken) (EasyFarmCore.sol#1466)
        Event emitted after the call(s):
        - Claim(_userAddr,pending,released) (EasyFarmCore.sol#1468)
```

```
Reentrancy in EasyFarmCore._deposit(address,uint256,uint256) (EasyFarmCore.sol#1471-1485):
        External calls:
        - _claim(_userAddr,_pid) (EasyFarmCore.sol#1472)
                - eft.transfer(_to,eftBal) (EasyFarmCore.sol#1680)
                - eft.transfer(_to,_amount) (EasyFarmCore.sol#1682)
                - eft.mint(address(this),incReward) (EasyFarmCore.sol#1547)
                - eft.mint(devAddr,devReward) (EasyFarmCore.sol#1550)
                - eft.mint(marketAddr,marketReward) (EasyFarmCore.sol#1552)
                - IEasyFarmStrategy(strategyAddr).claim(pool.depositToken) (EasyFarmCore.sol#1466)
        - IERC20(pool.depositToken).safeTransferFrom(_userAddr,address(this),_amount) (EasyFarmCore.sol#1477)
        - earn(_pid) (EasyFarmCore.sol#1483)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (EasyFarmCore.sol#582)
                - (success,returndata) = target.call{value: value}(data) (EasyFarmCore.sol#346)
                - IERC20(pool.depositToken).safeTransfer(strategyAddr,bal) (EasyFarmCore.sol#1522)
                - IEasyFarmStrategy(strategyAddr).deposit(pool.depositToken) (EasyFarmCore.sol#1523)
                - IEasyFarmStrategy(strategyAddr).deposit(pool.depositToken) (EasyFarmCore.sol#1529)
        External calls sending eth:
        - earn(_pid) (EasyFarmCore.sol#1483)
                - (success,returndata) = target.call{value: value}(data) (EasyFarmCore.sol#346)
                - address(strategyAddr).transfer(bal) (EasyFarmCore.sol#1528)
        Event emitted after the call(s):
        - Deposit(_userAddr,pool.depositToken,_amount) (EasyFarmCore.sol#1484)
        - Earn(pool.depositToken,strategyAddr,bal) (EasyFarmCore.sol#1532)
                - earn(_pid) (EasyFarmCore.sol#1483)
Reentrancy in EasyFarmCore._withdraw(address,uint256,uint256) (EasyFarmCore.sol#1487-1512):
        External calls:
        - _claim(_userAddr,_pid) (EasyFarmCore.sol#1488)
                - eft.transfer(_to,eftBal) (EasyFarmCore.sol#1680)
                - eft.transfer(_to,_amount) (EasyFarmCore.sol#1682)
                - eft.mint(address(this),incReward) (EasyFarmCore.sol#1547)
                - eft.mint(devAddr,devReward) (EasyFarmCore.sol#1550)
                - eft.mint(marketAddr,marketReward) (EasyFarmCore.sol#1552)
                - IEasyFarmStrategy(strategyAddr).claim(pool.depositToken) (EasyFarmCore.sol#1466)
        - IEasyFarmStrategy(strategyAddr).withdraw(pool.depositToken,_amount.sub(bal)) (EasyFarmCore.sol#1502)
        - IERC20(pool.depositToken).safeTransfer(_userAddr,_amount) (EasyFarmCore.sol#1507)
        External calls sending eth:
        - address(_userAddr).transfer(_amount) (EasyFarmCore.sol#1509)
        Event emitted after the call(s):
```

```
        - Withdraw(_userAddr,pool.depositToken,_amount) (EasyFarmCore.sol#1511)
Reentrancy in EasyFarmCore.earn(uint256) (EasyFarmCore.sol#1514-1535):
        External calls:
        - IERC20(pool.depositToken).safeTransfer(strategyAddr,bal) (EasyFarmCore.sol#1522)
        - IEasyFarmStrategy(strategyAddr).deposit(pool.depositToken) (EasyFarmCore.sol#1523)
        - IEasyFarmStrategy(strategyAddr).deposit(pool.depositToken) (EasyFarmCore.sol#1529)
        External calls sending eth:
        - address(strategyAddr).transfer(bal) (EasyFarmCore.sol#1528)
        Event emitted after the call(s):
        - Earn(pool.depositToken,strategyAddr,bal) (EasyFarmCore.sol#1532)
Reentrancy in EasyFarmCore.updatePool(uint256) (EasyFarmCore.sol#1537-1558):
        External calls:
        - eft.mint(address(this),incReward) (EasyFarmCore.sol#1547)
        - eft.mint(devAddr,devReward) (EasyFarmCore.sol#1550)
        - eft.mint(marketAddr,marketReward) (EasyFarmCore.sol#1552)
        Event emitted after the call(s):
        - UpdatePool(incReward,devReward,marketReward) (EasyFarmCore.sol#1557)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
EasyFarmToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (EasyFarmCore.sol#1155-1196) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,signature expired) (EasyFarmCore.sol#1194)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (EasyFarmCore.sol#241-251) uses assembly
        - INLINE ASM (EasyFarmCore.sol#247-249)
Address.verifyCallResult(bool,bytes,string) (EasyFarmCore.sol#410-430) uses assembly
        - INLINE ASM (EasyFarmCore.sol#422-425)
EasyFarmToken.getChainId() (EasyFarmCore.sol#1317-1321) uses assembly
        - INLINE ASM (EasyFarmCore.sol#1319)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
EasyFarmToken.onlyMinter() (EasyFarmCore.sol#1057-1060) compares to a boolean constant:
        -require(bool,string)(isMinter[msg.sender] == true,permission denied) (EasyFarmCore.sol#1058)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Address.functionCall(address,bytes) (EasyFarmCore.sol#294-296) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (EasyFarmCore.sol#323-329) is never used and should be removed
```

```
Address.functionDelegateCall(address,bytes) (EasyFarmCore.sol#383-385) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (EasyFarmCore.sol#393-402) is never used and should be removed
Address.functionStaticCall(address,bytes) (EasyFarmCore.sol#356-358) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (EasyFarmCore.sol#366-375) is never used and should be removed
Address.sendValue(address,uint256) (EasyFarmCore.sol#269-274) is never used and should be removed
Context._msgData() (EasyFarmCore.sol#595-597) is never used and should be removed
EasyFarmToken._transfer(address,address,uint256) (EasyFarmCore.sol#1091-1094) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (EasyFarmCore.sol#534-547) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (EasyFarmCore.sol#558-569) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (EasyFarmCore.sol#549-556) is never used and should be removed
SafeMath.div(uint256,uint256,string) (EasyFarmCore.sol#185-194) is never used and should be removed
SafeMath.mod(uint256,uint256) (EasyFarmCore.sol#145-147) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (EasyFarmCore.sol#211-220) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (EasyFarmCore.sol#162-171) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (EasyFarmCore.sol#16-22) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (EasyFarmCore.sol#58-63) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (EasyFarmCore.sol#70-75) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (EasyFarmCore.sol#41-51) is never used and should be removed
SafeMath.trySub(uint256,uint256) (EasyFarmCore.sol#29-34) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (EasyFarmCore.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (EasyFarmCore.sol#269-274):
        - (success) = recipient.call{value: amount}() (EasyFarmCore.sol#272)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (EasyFarmCore.sol#337-348):
        - (success,returndata) = target.call{value: value}(data) (EasyFarmCore.sol#346)
Low level call in Address.functionStaticCall(address,bytes,string) (EasyFarmCore.sol#366-375):
        - (success,returndata) = target.staticcall(data) (EasyFarmCore.sol#373)
Low level call in Address.functionDelegateCall(address,bytes,string) (EasyFarmCore.sol#393-402):
        - (success,returndata) = target.delegatecall(data) (EasyFarmCore.sol#400)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter EasyFarmToken.mint(address,uint256)._to (EasyFarmCore.sol#1062) is not in mixedCase
Parameter EasyFarmToken.mint(address,uint256)._amount (EasyFarmCore.sol#1062) is not in mixedCase
Parameter EasyFarmToken.burn(uint256)._amount (EasyFarmCore.sol#1077) is not in mixedCase
```

```
Parameter EasyFarmToken.setMinter(address,bool)._minter (EasyFarmCore.sol#1086) is not in mixedCase
Parameter EasyFarmToken.setMinter(address,bool)._status (EasyFarmCore.sol#1086) is not in mixedCase
Variable EasyFarmToken._delegates (EasyFarmCore.sol#1097) is not in mixedCase
Parameter EasyFarmCore.add(address,uint256,uint256)._depositToken (EasyFarmCore.sol#1413) is not in mixedCase
Parameter EasyFarmCore.add(address,uint256,uint256)._rewardPerBlock (EasyFarmCore.sol#1413) is not in mixedCase
Parameter EasyFarmCore.add(address,uint256,uint256)._earnThreshold (EasyFarmCore.sol#1413) is not in mixedCase
Parameter EasyFarmCore.deposit(uint256,uint256)._pid (EasyFarmCore.sol#1428) is not in mixedCase
Parameter EasyFarmCore.deposit(uint256,uint256)._amount (EasyFarmCore.sol#1428) is not in mixedCase
Parameter EasyFarmCore.withdraw(uint256,uint256)._pid (EasyFarmCore.sol#1435) is not in mixedCase
Parameter EasyFarmCore.withdraw(uint256,uint256)._amount (EasyFarmCore.sol#1435) is not in mixedCase
Parameter EasyFarmCore.withdrawAll(uint256)._pid (EasyFarmCore.sol#1439) is not in mixedCase
Parameter EasyFarmCore.claim(uint256)._pid (EasyFarmCore.sol#1443) is not in mixedCase
Parameter EasyFarmCore.earn(uint256)._pid (EasyFarmCore.sol#1514) is not in mixedCase
Parameter EasyFarmCore.updatePool(uint256)._pid (EasyFarmCore.sol#1537) is not in mixedCase
Parameter EasyFarmCore.pendingAll(address)._userAddr (EasyFarmCore.sol#1560) is not in mixedCase
Parameter EasyFarmCore.pendingReward(uint256,address)._pid (EasyFarmCore.sol#1569) is not in mixedCase
Parameter EasyFarmCore.pendingReward(uint256,address)._userAddr (EasyFarmCore.sol#1569) is not in mixedCase
Parameter EasyFarmCore.getPoolInfo(uint256)._pid (EasyFarmCore.sol#1609) is not in mixedCase
Parameter EasyFarmCore.getUserInfo(uint256,address)._pid (EasyFarmCore.sol#1613) is not in mixedCase
Parameter EasyFarmCore.getUserInfo(uint256,address)._user (EasyFarmCore.sol#1613) is not in mixedCase
Parameter EasyFarmCore.setStartBlock(uint256)._startBlock (EasyFarmCore.sol#1625) is not in mixedCase
Parameter EasyFarmCore.setLockBlock(uint256)._lockBlock (EasyFarmCore.sol#1629) is not in mixedCase
Parameter EasyFarmCore.setStrategy(uint256,address,address)._pid (EasyFarmCore.sol#1633) is not in mixedCase
Parameter EasyFarmCore.setStrategy(uint256,address,address)._tokenAddr (EasyFarmCore.sol#1633) is not in mixedCase
Parameter EasyFarmCore.setStrategy(uint256,address,address)._strategyAddr (EasyFarmCore.sol#1633) is not in mixedCase
Parameter EasyFarmCore.setRewardPerBlock(uint256,uint256,bool)._pid (EasyFarmCore.sol#1642) is not in mixedCase
Parameter EasyFarmCore.setRewardPerBlock(uint256,uint256,bool)._rewardPerBlock (EasyFarmCore.sol#1642) is not in mixedCase
Parameter EasyFarmCore.setRewardPerBlock(uint256,uint256,bool)._withUpdate (EasyFarmCore.sol#1642) is not in mixedCase
Parameter EasyFarmCore.setEarnThreshold(uint256,uint256)._pid (EasyFarmCore.sol#1652) is not in mixedCase
Parameter EasyFarmCore.setEarnThreshold(uint256,uint256)._earnThreshold (EasyFarmCore.sol#1652) is not in mixedCase
Parameter EasyFarmCore.setDevPercents(uint256)._devPercents (EasyFarmCore.sol#1657) is not in mixedCase
Parameter EasyFarmCore.setMarketPercents(uint256)._marketPercents (EasyFarmCore.sol#1661) is not in mixedCase
Parameter EasyFarmCore.setGover(address)._gover (EasyFarmCore.sol#1665) is not in mixedCase
Parameter EasyFarmCore.setDev(address)._devAddr (EasyFarmCore.sol#1669) is not in mixedCase
Parameter EasyFarmCore.setMarket(address)._marketAddr (EasyFarmCore.sol#1673) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
Reentrancy in EasyFarmCore._deposit(address,uint256,uint256) (EasyFarmCore.sol#1471-1485):
        External calls:
        - earn(_pid) (EasyFarmCore.sol#1483)
                - address(strategyAddr).transfer(bal) (EasyFarmCore.sol#1528)
        External calls sending eth:
        - earn(_pid) (EasyFarmCore.sol#1483)
                - (success,returndata) = target.call{value: value}(data) (EasyFarmCore.sol#346)
                - address(strategyAddr).transfer(bal) (EasyFarmCore.sol#1528)
        Event emitted after the call(s):
        - Deposit(_userAddr,pool.depositToken,_amount) (EasyFarmCore.sol#1484)
Reentrancy in EasyFarmCore._withdraw(address,uint256,uint256) (EasyFarmCore.sol#1487-1512):
        External calls:
        - address(_userAddr).transfer(_amount) (EasyFarmCore.sol#1509)
        Event emitted after the call(s):
        - Withdraw(_userAddr,pool.depositToken,_amount) (EasyFarmCore.sol#1511)
Reentrancy in EasyFarmCore.earn(uint256) (EasyFarmCore.sol#1514-1535):
        External calls:
        - address(strategyAddr).transfer(bal) (EasyFarmCore.sol#1528)
        Event emitted after the call(s):
        - Earn(pool.depositToken,strategyAddr,bal) (EasyFarmCore.sol#1532)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (EasyFarmCore.sol#634-636)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (EasyFarmCore.sol#642-645)
symbol() should be declared external:
        - ERC20.symbol() (EasyFarmCore.sol#754-756)
decimals() should be declared external:
        - ERC20.decimals() (EasyFarmCore.sol#771-773)
totalSupply() should be declared external:
        - ERC20.totalSupply() (EasyFarmCore.sol#778-780)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (EasyFarmCore.sol#797-800)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (EasyFarmCore.sol#805-807)
approve(address,uint256) should be declared external:
```

```
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (EasyFarmCore.sol#797-800)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (EasyFarmCore.sol#805-807)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (EasyFarmCore.sol#816-819)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (EasyFarmCore.sol#834-848)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (EasyFarmCore.sol#862-865)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (EasyFarmCore.sol#881-889)
burn(uint256) should be declared external:
        - EasyFarmToken.burn(uint256) (EasyFarmCore.sol#1077-1080)
totalBurned() should be declared external:
        - EasyFarmToken.totalBurned() (EasyFarmCore.sol#1082-1084)
add(address,uint256,uint256) should be declared external:
        - EasyFarmCore.add(address,uint256,uint256) (EasyFarmCore.sol#1413-1426)
withdraw(uint256,uint256) should be declared external:
        - EasyFarmCore.withdraw(uint256,uint256) (EasyFarmCore.sol#1435-1437)
withdrawAll(uint256) should be declared external:
        - EasyFarmCore.withdrawAll(uint256) (EasyFarmCore.sol#1439-1441)
claim(uint256) should be declared external:
        - EasyFarmCore.claim(uint256) (EasyFarmCore.sol#1443-1445)
claimAll() should be declared external:
        - EasyFarmCore.claimAll() (EasyFarmCore.sol#1447-1451)
pendingAll(address) should be declared external:
        - EasyFarmCore.pendingAll(address) (EasyFarmCore.sol#1560-1567)
setRewardPerBlock(uint256,uint256,bool) should be declared external:
        - EasyFarmCore.setRewardPerBlock(uint256,uint256,bool) (EasyFarmCore.sol#1642-1650)
setDevPercents(uint256) should be declared external:
        - EasyFarmCore.setDevPercents(uint256) (EasyFarmCore.sol#1657-1659)
setMarketPercents(uint256) should be declared external:
        - EasyFarmCore.setMarketPercents(uint256) (EasyFarmCore.sol#1661-1663)
setDev(address) should be declared external:
        - EasyFarmCore.setDev(address) (EasyFarmCore.sol#1669-1671)
setMarket(address) should be declared external:
        - EasyFarmCore.setMarket(address) (EasyFarmCore.sol#1673-1675)
```

```
setMarket(address) should be declared external:
        - EasyFarmCore.setMarket(address) (EasyFarmCore.sol#1673-1675)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:EasyFarmCore.sol analyzed (12 contracts with 75 detectors), 128 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
root@server:/chotan/gaza/mycontracts#
```

Slither log >> LaunchEFT.sol

```
INFO:Detectors:
EasyFarmToken.constructor(string,string,uint256)._name (LaunchEFT.sol#999) shadows:
        - ERC20._name (LaunchEFT.sol#677) (state variable)
EasyFarmToken.constructor(string,string,uint256)._symbol (LaunchEFT.sol#1000) shadows:
        - ERC20._symbol (LaunchEFT.sol#678) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
EasyFarmToken.burn(uint256) (LaunchEFT.sol#1028-1031) should emit an event for:
        - _totalBurned = _totalBurned.add(_amount) (LaunchEFT.sol#1030)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
LaunchEFT.constructor(address,address,uint256,uint256,uint256,uint256,uint256,uint256,uint256[],uint256[])._busdAddr (LaunchEFT.s
ol#1321) lacks a zero-check on :
                - busdAddr = _busdAddr (LaunchEFT.sol#1335)
LaunchEFT.constructor(address,address,uint256,uint256,uint256,uint256,uint256,uint256,uint256[],uint256[])._eftAddr (LaunchEFT.so
l#1322) lacks a zero-check on :
                - eftAddr = _eftAddr (LaunchEFT.sol#1336)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in LaunchEFT.burnRemain() (LaunchEFT.sol#1407-1415):
        External calls:
        - EasyFarmToken(eftAddr).burn(remainLaunch) (LaunchEFT.sol#1410)
        State variables written after the call(s):
        - burnLaunch = remainLaunch (LaunchEFT.sol#1411)
Reentrancy in LaunchEFT.support(address,uint256) (LaunchEFT.sol#1350-1383):
        External calls:
        - IERC20(busdAddr).safeTransferFrom(msg.sender,address(this),_amount) (LaunchEFT.sol#1359)
        State variables written after the call(s):
        - extraRemain = extraRemain.sub(1) (LaunchEFT.sol#1367)
        - isExtraRewarded[msg.sender] = true (LaunchEFT.sol#1366)
Reentrancy in LaunchEFT.support(address,uint256) (LaunchEFT.sol#1350-1383):
        External calls:
        - IERC20(busdAddr).safeTransferFrom(msg.sender,address(this),_amount) (LaunchEFT.sol#1359)
        - IERC20(busdAddr).safeTransfer(msg.sender,refund) (LaunchEFT.sol#1372)
        State variables written after the call(s):
        - totalSupport = totalSupport.add(_amount) (LaunchEFT.sol#1375)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Address.verifyCallResult(bool,bytes,string) (LaunchEFT.sol#405-425) uses assembly
        - INLINE ASM (LaunchEFT.sol#417-420)
EasyFarmToken.getChainId() (LaunchEFT.sol#1268-1272) uses assembly
        - INLINE ASM (LaunchEFT.sol#1270)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
EasyFarmToken.onlyMinter() (LaunchEFT.sol#1008-1011) compares to a boolean constant:
        -require(bool,string)(isMinter[msg.sender] == true,permission denied) (LaunchEFT.sol#1009)
LaunchEFT.support(address,uint256) (LaunchEFT.sol#1350-1383) compares to a boolean constant:
        -extraRemain > 0 && isExtraRewarded[msg.sender] == false (LaunchEFT.sol#1363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Address.functionCall(address,bytes) (LaunchEFT.sol#289-291) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (LaunchEFT.sol#318-324) is never used and should be removed
Address.functionDelegateCall(address,bytes) (LaunchEFT.sol#378-380) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (LaunchEFT.sol#388-397) is never used and should be removed
Address.functionStaticCall(address,bytes) (LaunchEFT.sol#351-353) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (LaunchEFT.sol#361-370) is never used and should be removed
Address.sendValue(address,uint256) (LaunchEFT.sol#264-269) is never used and should be removed
Context._msgData() (LaunchEFT.sol#590-592) is never used and should be removed
EasyFarmToken._transfer(address,address,uint256) (LaunchEFT.sol#1042-1045) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (LaunchEFT.sol#529-542) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (LaunchEFT.sol#553-564) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (LaunchEFT.sol#544-551) is never used and should be removed
SafeMath.div(uint256,uint256,string) (LaunchEFT.sol#180-189) is never used and should be removed
SafeMath.mod(uint256,uint256) (LaunchEFT.sol#140-142) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (LaunchEFT.sol#206-215) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (LaunchEFT.sol#157-166) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (LaunchEFT.sol#11-17) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (LaunchEFT.sol#53-58) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (LaunchEFT.sol#65-70) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (LaunchEFT.sol#36-46) is never used and should be removed
SafeMath.trySub(uint256,uint256) (LaunchEFT.sol#24-29) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (LaunchEFT.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (LaunchEFT.sol#264-269):
        - (success) = recipient.call{value: amount}() (LaunchEFT.sol#267)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (LaunchEFT.sol#332-343):
        - (success,returndata) = target.call{value: value}(data) (LaunchEFT.sol#341)
Low level call in Address.functionStaticCall(address,bytes,string) (LaunchEFT.sol#361-370):
        - (success,returndata) = target.staticcall(data) (LaunchEFT.sol#368)
Low level call in Address.functionDelegateCall(address,bytes,string) (LaunchEFT.sol#388-397):
        - (success,returndata) = target.delegatecall(data) (LaunchEFT.sol#395)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter EasyFarmToken.mint(address,uint256)._to (LaunchEFT.sol#1013) is not in mixedCase
Parameter EasyFarmToken.mint(address,uint256)._amount (LaunchEFT.sol#1013) is not in mixedCase
Parameter EasyFarmToken.burn(uint256)._amount (LaunchEFT.sol#1028) is not in mixedCase
Parameter EasyFarmToken.setMinter(address,bool)._minter (LaunchEFT.sol#1037) is not in mixedCase
Parameter EasyFarmToken.setMinter(address,bool)._status (LaunchEFT.sol#1037) is not in mixedCase
Variable EasyFarmToken._delegates (LaunchEFT.sol#1048) is not in mixedCase
Parameter LaunchEFT.support(address,uint256)._referral (LaunchEFT.sol#1350) is not in mixedCase
Parameter LaunchEFT.support(address,uint256)._amount (LaunchEFT.sol#1350) is not in mixedCase
Parameter LaunchEFT.pending(address)._userAddr (LaunchEFT.sol#1396) is not in mixedCase
Parameter LaunchEFT.getPrice(address)._userAddr (LaunchEFT.sol#1417) is not in mixedCase
Parameter LaunchEFT.getUserReferral(address)._userAddr (LaunchEFT.sol#1429) is not in mixedCase
Parameter LaunchEFT.getUserInfo(address)._userAddr (LaunchEFT.sol#1433) is not in mixedCase
Parameter LaunchEFT.setEndBlock(uint256)._endBlock (LaunchEFT.sol#1450) is not in mixedCase
Parameter LaunchEFT.setLockBlock(uint256)._lockBlock (LaunchEFT.sol#1454) is not in mixedCase
Parameter LaunchEFT.setMinSupport(uint256)._minSupport (LaunchEFT.sol#1458) is not in mixedCase
Parameter LaunchEFT.setExtraInfo(uint256,uint256)._extraRemain (LaunchEFT.sol#1462) is not in mixedCase
Parameter LaunchEFT.setExtraInfo(uint256,uint256)._extraRate (LaunchEFT.sol#1462) is not in mixedCase
Parameter LaunchEFT.setTotalLaunch(uint256)._totalLaunch (LaunchEFT.sol#1467) is not in mixedCase
Parameter LaunchEFT.setInviteInfo(uint256[],uint256[])._inviteLevel (LaunchEFT.sol#1471) is not in mixedCase
Parameter LaunchEFT.setInviteInfo(uint256[],uint256[])._inviteDiscount (LaunchEFT.sol#1471) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (LaunchEFT.sol#629-631)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (LaunchEFT.sol#637-640)
symbol() should be declared external:
```

```
symbol() should be declared external:
        - ERC20.symbol() (LaunchEFT.sol#705-707)
decimals() should be declared external:
        - ERC20.decimals() (LaunchEFT.sol#722-724)
totalSupply() should be declared external:
        - ERC20.totalSupply() (LaunchEFT.sol#729-731)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (LaunchEFT.sol#748-751)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (LaunchEFT.sol#756-758)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (LaunchEFT.sol#767-770)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (LaunchEFT.sol#785-799)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (LaunchEFT.sol#813-816)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (LaunchEFT.sol#832-840)
burn(uint256) should be declared external:
        - EasyFarmToken.burn(uint256) (LaunchEFT.sol#1028-1031)
totalBurned() should be declared external:
        - EasyFarmToken.totalBurned() (LaunchEFT.sol#1033-1035)
support(address,uint256) should be declared external:
        - LaunchEFT.support(address,uint256) (LaunchEFT.sol#1350-1383)
claim() should be declared external:
        - LaunchEFT.claim() (LaunchEFT.sol#1385-1394)
burnRemain() should be declared external:
        - LaunchEFT.burnRemain() (LaunchEFT.sol#1407-1415)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LaunchEFT.sol analyzed (10 contracts with 75 detectors), 87 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

EasyFarmToken.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 977:8:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 852:16:

Gas & Economy

Gas costs:

Gas requirement of function ERC20.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 404:4:

Gas costs:

Gas requirement of function EasyFarmToken.delegates is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 788:4:

Gas costs:

Gas requirement of function EasyFarmToken.delegate is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 800:4:

Gas costs:

Gas requirement of function EasyFarmToken.getPriorVotes is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 877:4:

Miscellaneous

Constant/View/Pure functions:

IERC20:transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 303:4:

Similar variable names:

EasyFarmToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) : Variables have very similar names "_delegates" and "delegatee". Note: Modifiers are currently not considered by this static analysis.

Pos: 853:36:

Similar variable names:

EasyFarmToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) : Variables have very similar names "nonce" and "nonces". Note: Modifiers are currently not considered by this static analysis.

Pos: 836:16:

Similar variable names:

EasyFarmToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) : Variables have very similar names "nonce" and "nonces". Note: Modifiers are currently not considered by this static analysis.

Pos: 851:16:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 716:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 850:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 882:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 971:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 903:36:

EasyFarmCore.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 332:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in EasyFarmCore.earn(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1509:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in EasyFarmCore.updatePool(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1532:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 1314:8:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1189:16:

Low level calls:

Use of "delegatecall": should be avoided whenever possible.
External code, that is called can change the state of the calling contract and send ether from the caller's balance.
If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 395:50:

Gas & Economy

Gas costs:

Gas requirement of function ERC20.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 741:4:

Gas costs:

Gas requirement of function EasyFarmToken.delegates is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 1125:4:

Gas costs:

Gas requirement of function EasyFarmToken.delegate is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 1137:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.
Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1443:8:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.
Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1556:8:

Miscellaneous

Constant/View/Pure functions:

Address.isContract(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 236:4:

Constant/View/Pure functions:

EasyFarmCore.setEarnThreshold(uint256,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1647:4:

Similar variable names:

EasyFarmToken.delegate(address) : Variables have very similar names "_delegates" and "delegatee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1138:37:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 1402:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 1629:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 1240:36:

LaunchEFT.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 332:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in LaunchEFT.burnRemain(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1407:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in LaunchEFT.withdraw(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1477:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 1270:8:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 1145:16:

Gas & Economy

Gas costs:

Gas requirement of function ERC20.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 697:4:

Gas costs:

Gas requirement of function LaunchEFT.claim is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 1385:4:

Miscellaneous

Constant/View/Pure functions:

Address.isContract(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 236:4:

Similar variable names:

EasyFarmToken._writeCheckpoint(address,uint32,uint256,uint256) : Variables have very similar names "numCheckpoints" and "nCheckpoints". Note: Modifiers are currently not considered by this static analysis.
Pos: 1257:12:

Similar variable names:

EasyFarmToken._writeCheckpoint(address,uint32,uint256,uint256) : Variables have very similar names "numCheckpoints" and "nCheckpoints". Note: Modifiers are currently not considered by this static analysis.
Pos: 1257:40:

No return:

IERC20Metadata.symbol(): Defines a return type but never explicitly returns a value.
Pos: 662:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1472:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1196:36:

Solhint Linter

EasyFarmToken.sol

```
EasyFarmToken.sol:11:18: Error: Parse error: missing ';' at '{'
EasyFarmToken.sol:24:18: Error: Parse error: missing ';' at '{'
EasyFarmToken.sol:36:18: Error: Parse error: missing ';' at '{'
EasyFarmToken.sol:53:18: Error: Parse error: missing ';' at '{'
EasyFarmToken.sol:65:18: Error: Parse error: missing ';' at '{'
EasyFarmToken.sol:161:18: Error: Parse error: missing ';' at '{'
EasyFarmToken.sol:184:18: Error: Parse error: missing ';' at '{'
EasyFarmToken.sol:210:18: Error: Parse error: missing ';' at '{'
EasyFarmToken.sol:501:18: Error: Parse error: missing ';' at '{'
EasyFarmToken.sol:542:18: Error: Parse error: missing ';' at '{'
EasyFarmToken.sol:575:18: Error: Parse error: missing ';' at '{'
EasyFarmToken.sol:624:18: Error: Parse error: missing ';' at '{'
```

EasyFarmCore.sol

```
EasyFarmCore.sol:17:18: Error: Parse error: missing ';' at '{'
EasyFarmCore.sol:30:18: Error: Parse error: missing ';' at '{'
EasyFarmCore.sol:42:18: Error: Parse error: missing ';' at '{'
EasyFarmCore.sol:59:18: Error: Parse error: missing ';' at '{'
EasyFarmCore.sol:71:18: Error: Parse error: missing ';' at '{'
EasyFarmCore.sol:167:18: Error: Parse error: missing ';' at '{'
EasyFarmCore.sol:190:18: Error: Parse error: missing ';' at '{'
EasyFarmCore.sol:216:18: Error: Parse error: missing ';' at '{'
EasyFarmCore.sol:563:18: Error: Parse error: missing ';' at '{'
EasyFarmCore.sol:843:18: Error: Parse error: missing ';' at '{'
EasyFarmCore.sol:884:18: Error: Parse error: missing ';' at '{'
EasyFarmCore.sol:917:18: Error: Parse error: missing ';' at '{'
EasyFarmCore.sol:966:18: Error: Parse error: missing ';' at '{'
```

LaunchEFT.sol

```
LaunchEFT.sol:12:18: Error: Parse error: missing ';' at '{'
LaunchEFT.sol:25:18: Error: Parse error: missing ';' at '{'
LaunchEFT.sol:37:18: Error: Parse error: missing ';' at '{'
LaunchEFT.sol:54:18: Error: Parse error: missing ';' at '{'
LaunchEFT.sol:66:18: Error: Parse error: missing ';' at '{'
LaunchEFT.sol:162:18: Error: Parse error: missing ';' at '{'
LaunchEFT.sol:185:18: Error: Parse error: missing ';' at '{'
LaunchEFT.sol:211:18: Error: Parse error: missing ';' at '{'
LaunchEFT.sol:558:18: Error: Parse error: missing ';' at '{'
LaunchEFT.sol:794:18: Error: Parse error: missing ';' at '{'
LaunchEFT.sol:835:18: Error: Parse error: missing ';' at '{'
LaunchEFT.sol:868:18: Error: Parse error: missing ';' at '{'
LaunchEFT.sol:917:18: Error: Parse error: missing ';' at '{'
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io