Smart Contract

Security Assessment

For EINSTEIN

15 June 2022

Ascendant

Ascendant

@ascendantproj

www.ascendant.finance.com

Ascendant

Table of Contents

3 Disclaimer

4 Executive Summary

5 Overview

6 Findings Summary & Legend

7 Manual Review
- Issue Checking Status
- Audit Findings
- Functional Test Status

21 Automated Review
- Solidity Static Analysis
- Inheritance Graph
- Unified Model Language
- Function ID Report

27 Conclusion

DISCLAIMER

Ascendant Finance ("Ascendant") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Ascendant.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Ascendant is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Ascendant or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Ascendant retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Ascendant is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Ascendant may, at its discretion, claim bug bounties from third-parties while doing so.

Executive Summary

Severity	Found
🔴 High	2
🟠 Medium	3
🟡 Low	13
🟣 Informational	60
Total	78

We performed an independent technical audit to identify Smart Contracts uncertainties. This shall protect the code from illegitimate authorization attempts or external & internal threats of any type. This also ensures end-to-end proofing of the contract from frauds. The audit was performed semi-manually. We analyzed the Smart Contracts code line-by-line and used an automation tool to report any suspicious code.

The following tools were used:

- Truffle
- Remix IDE
- Slither

Overview

This report has been prepared for EINSTEIN on the Binance network. Ascendant provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

Summary

Project Name	EINSTEIN
Platform	Binance
Language	Solidity

Contracts Assessed

Name	**Location**
EINSTEIN.sol	Not deployed
SafeMath.sol	In EINSTEIN Contract

Findings Summary

Severity	Found
🔴 High	2
🟠 Medium	3
🟡 Low	13
🟣 Informational	60
Total	78

Classification of Issues

🔴 High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
🟠 Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
🟡 Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
🟣 Informational	Consistency, syntax or style best practices, Generally pose a negligible level of risk, if any.

Manual Review

Ascendant

Issues Checking Status

Issue Description	Checking Status
Compiler errors	PASS
Race conditions and Reentrancy. Cross-function race conditions.	FAIL
Possible delays in data delivery.	PASS
Oracle calls.	PASS
Front running.	PASS
Timestamp dependence.	PASS
Integer Overflow and Underflow.	PASS
DoS with Revert.	PASS
DoS with block gas limit.	PASS
Methods execution permissions.	PASS
Economy model of the contract.	PASS
The impact of the exchange rate on the logic.	PASS
Private user data leaks.	PASS
Malicious Event log.	PASS
Scoping and Declarations.	PASS
Uninitialized storage pointers.	PASS

Arithmetic accuracy.	PASS
Design Logic.	FAIL
Cross-function race conditions.	PASS
Safe Open Zeppelin contracts implementation and usage.	PASS
Fallback function security.	PASS

Audit Findings

Severity	**High**
Contract	EINSTEIN.sol
Description	EINSTEIN.Blacklisted (Miner.sol#37) is never initialized. It is used in: - EINSTEIN.sellCrops() (Miner.sol#128-213)

Code Snippet	function sellCrops() public{
	 require(contractStarted, "Contract not yet Started.");
	if (blacklistActive) {
	 require(!Blacklisted[msg.sender], "Address is blacklisted.");
	 }

Recommendation	Currently, you cannot toggle blacklistActive to True. You also cannot add a user to the blacklist. Both require a function to correct the issue.

Severity	**High**
Contract	EINSTEIN.sol
Description	Dev1 does not receive fees.
Code Snippet	`function payFees(uint256 eggValue) internal returns(uint256){ uint256 tax = eggValue.mul(TAX).div(PERCENTS_DIVIDER); uint256 mktng = eggValue.mul(MKT).div(PERCENTS_DIVIDER); dev1.transfer(tax); mkt.transfer(mktng); return mktng.add(tax.mul(1)); }`
Recommendation	payFees returns a uint256. When this function is called within hireFarmers, the only value returned is the mktng value. It is recommended that you remove the return declaration and statement.

Severity	**Medium**
Contract	EINSTEIN.sol
Description	uses vulnerable isContract

Code Snippet	function isContract(address addr) internal view returns (bool) {
	uint size;
	 assembly { size := extcodesize(addr) }
	 return size > 0;
	 }

Recommendation	Check msg.sender != tx.origin to verify the caller is a contract.

Severity	Medium
Contract	EINSTEIN.sol
Description	Function does not follow Checks, Effects, and Interactions Flow (possible Reentrancy vulnerability)

Code Snippet	uint256 eggsPayout = payFees(msg.value);
	 totalStaked =
	totalStaked.add(msg.value.sub(eggsPayout));
	 totalDeposits = totalDeposits.add(1);
	 hireMoreFarmers(false);
	 }

Recommendation	payFees (because it includes a transfer) should occur AFTER the state of the blockchain is updated, meaning it should come after totalStaked and totalDeposits are updated.

Severity	**Medium**
Contract	EINSTEIN.sol
Description	Function does not follow Checks, Effects, and Interactions Flow (possible Reentrancy vulnerability)

Code Snippet	sellCrops() {
	...
	//
	uint256 eggsPayout = eggValue.sub(payFees(eggValue));
	 payable(address(msg.sender)).transfer(eggsPayout);
	 user.totalWithdrawn = user.totalWithdrawn.add(eggsPayout);
	 totalWithdrawn = totalWithdrawn.add(eggsPayout);

Recommendation	Refer to the previous explanation about Checks, Effects, and Interactions.

Severity	Low
Contract	EINSTEIN.sol
Description	CHANGE_OWNERSHIP function lacks zero-check validation

Code Snippet	function CHANGE_OWNERSHIP(address value) external {
	 require(msg.sender == owner, "Admin use only.");
	 owner = value;
	 }

Recommendation	Add a require statement that requires the new owner address to not be address(0), or the zero address.

Severity	**Low**
Contract	EINSTEIN.sol

Description	Contract lacks events emissions for the following:
	hireFarmers()
	PRC_TAX()
	PRC_MKT()
	BONUS_DAILY_COMPOUND()
	SET_CUTOFF_STEP()

Code Snippet	See the corresponding code for each function name.
Recommendation	Adding events to these functions provide clarity and transparency for important transactions or changes to the contract.

Functional Test Status

Function Name	Type/Return Type	Score
EGGS_TO_HIRE_1MINERS	read/public	PASS
REFERRAL	read/public	PASS
PERCENTS_DIVIDER	read/public	PASS
TAX	read/public	PASS
MKT	read/public	PASS
MARKET_EGGS_DIVISOR	read/public	PASS
MIN_INVEST_LIMIT	read/public	PASS
WALLET_DEPOSIT_LIMIT	read/public	PASS
COMPOUND_BONUS	read/public	PASS
COMPOUND_BONUS_MAX_TIMES	read/public	PASS
COMPOUND_STEP	read/public	PASS
WITHDRAWAL_LIMIT	read/public	PASS
WITHDRAWAL_TAX	read/public	PASS
COMPOUND_FOR_NO_TAX_WITHDRAWAL	read/public	PASS
totalStaked	read/public	PASS
totalDeposits	read/public	PASS
totalCompound	read/public	PASS
totalRefBonus	read/public	PASS

marketEggs	read/public	PASS
PSN	read/public	PASS
PSNH	read/public	PASS
blacklistActive	read/public	FAIL
Blacklisted	read/public	FAIL
CUTOFF_STEP	read/public	PASS
WUTHDRAW_COOLDOWN	read/public	PASS
owner	payable/public	PASS
dev1	payable/public	PASS
mkt	payable/public	PASS
initialDeposit	read/public	PASS
userDeposit	read/public	PASS
miners	read/public	PASS
claimedEggs	read/public	PASS
lastHatch	read/public	PASS
referrer	read/public	PASS
referralsCount	read/public	PASS
referralEggRewards	read/public	PASS
totalWithdrawn	read/public	PASS
dailyCompoundBonus	read/public	PASS

farmerCompoundCount	read/public	PASS
lastWithdrawTime	read/public	PASS
PSNH	read/public	PASS
users	read/public	PASS
isContract	internal	PASS
startFarm	payable/public	PASS
fundContract	payable/public	PASS
hireMoreFarmers	write/public	PASS
sellCrops	write/public	PASS
hireFarmers	payable/public	PASS
payFees	internal	FAIL
getDailyCompoundBonus	read/public	PASS
getUserInfo	read/public	PASS
getBalance	read/public	PASS
getTimeStamp	read/public	PASS
getAvailableEarnings	read/public	PASS
calculateTrade	read/public	PASS
calculateEggSell	read/public	PASS
calculateEggBuy	read/public	PASS
calculateEggBuySimple	read/public	PASS
calculateEggsYield	read/public	PASS

calculateEggSellForYield	read/public	PASS
getSiteInfo	read/public	PASS
getMyMiners	read/public	PASS
getMyEggs	read/public	PASS
getEggsSinceLastHatch	read/public	PASS
min	read/public	PASS
CHANGE_OWNERSHIP	write/public	PASS
PRC_EGGS_TO_HIRE_1MINERS	write/public	PASS
PRC_TAX	write/public	PASS
PRC_MKT	write/public	PASS
PRC_REFERRAL	write/public	PASS
SET_WITHDRAWAL_TAX	write/public	PASS
BONUS_DAILY_COMPOUND	write/public	PASS
BONUS_DAILY_COMPOUND_BONUS_MAX_TIMES	write/public	PASS
BONUS_COMPOUND_STEP	write/public	PASS
SET_INVEST_MIN	write/public	PASS
SET_WITHDRAWL_LIMIT	write/public	PASS
SET_CUTOFF_STEP	write/public	PASS
SET_WITHDRAW_COOLDOWN	write/public	PASS
SET_WALLET_DEPOSIT_LIMIT	write/public	PASS
SET_COMPOUND_FOR_NO_TAX_WITHDRAWAL	write/public	PASS

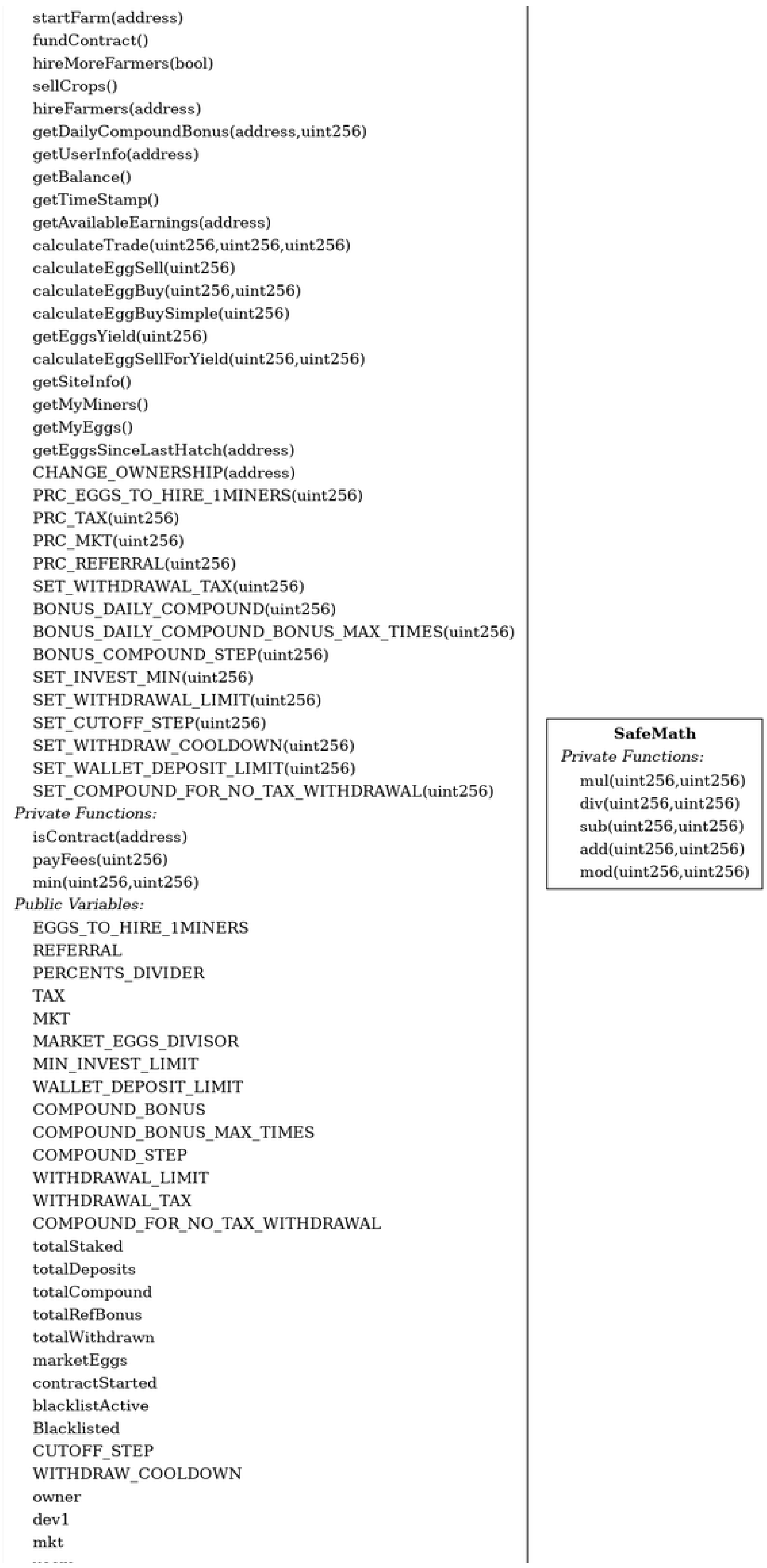
Automated Review

Ascendant

Solidity Static Analysis

Issue	Severity
Check-effects-interaction: Potential violation of Checks-Effects-Interaction pattern in EINSTEIN.sellCrops(): Could potentially lead to re-entrancy vulnerability. Pos: 123:4: Check-effects-interaction: Potential violation of Checks-Effects-Interaction pattern in EINSTEIN.hireFarmers(address): Could potentially lead to re-entrancy vulnerability. more Pos: 205:4:	Medium
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.	Informational
EINSTEIN.hireMoreFarmers(bool) : Variables have very similar names "user" and "users".	Informational
Similar variable names: EINSTEIN.payFees(uint256) : Variables have very similar names "mkt" and "mktng".	Informational

Inheritance Graph



startFarm(address)
fundContract()
hireMoreFarmers(bool)
sellCrops()
hireFarmers(address)
getDailyCompoundBonus(address,uint256)
getUserInfo(address)
getBalance()
getTimeStamp()
getAvailableEarnings(address)
calculateTrade(uint256,uint256,uint256)
calculateEggSell(uint256)
calculateEggBuy(uint256,uint256)
calculateEggBuySimple(uint256)
getEggsYield(uint256)
calculateEggSellForYield(uint256,uint256)
getSiteInfo()
getMyMiners()
getMyEggs()
getEggsSinceLastHatch(address)
CHANGE_OWNERSHIP(address)
PRC_EGGS_TO_HIRE_1MINERS(uint256)
PRC_TAX(uint256)
PRC_MKT(uint256)
PRC_REFERRAL(uint256)
SET_WITHDRAWAL_TAX(uint256)
BONUS_DAILY_COMPOUND(uint256)
BONUS_DAILY_COMPOUND_BONUS_MAX_TIMES(uint256)
BONUS_COMPOUND_STEP(uint256)
SET_INVEST_MIN(uint256)
SET_WITHDRAWAL_LIMIT(uint256)
SET_CUTOFF_STEP(uint256)
SET_WITHDRAW_COOLDOWN(uint256)
SET_WALLET_DEPOSIT_LIMIT(uint256)
SET_COMPOUND_FOR_NO_TAX_WITHDRAWAL(uint256)

Private Functions:

isContract(address)
payFees(uint256)
min(uint256,uint256)

Public Variables:

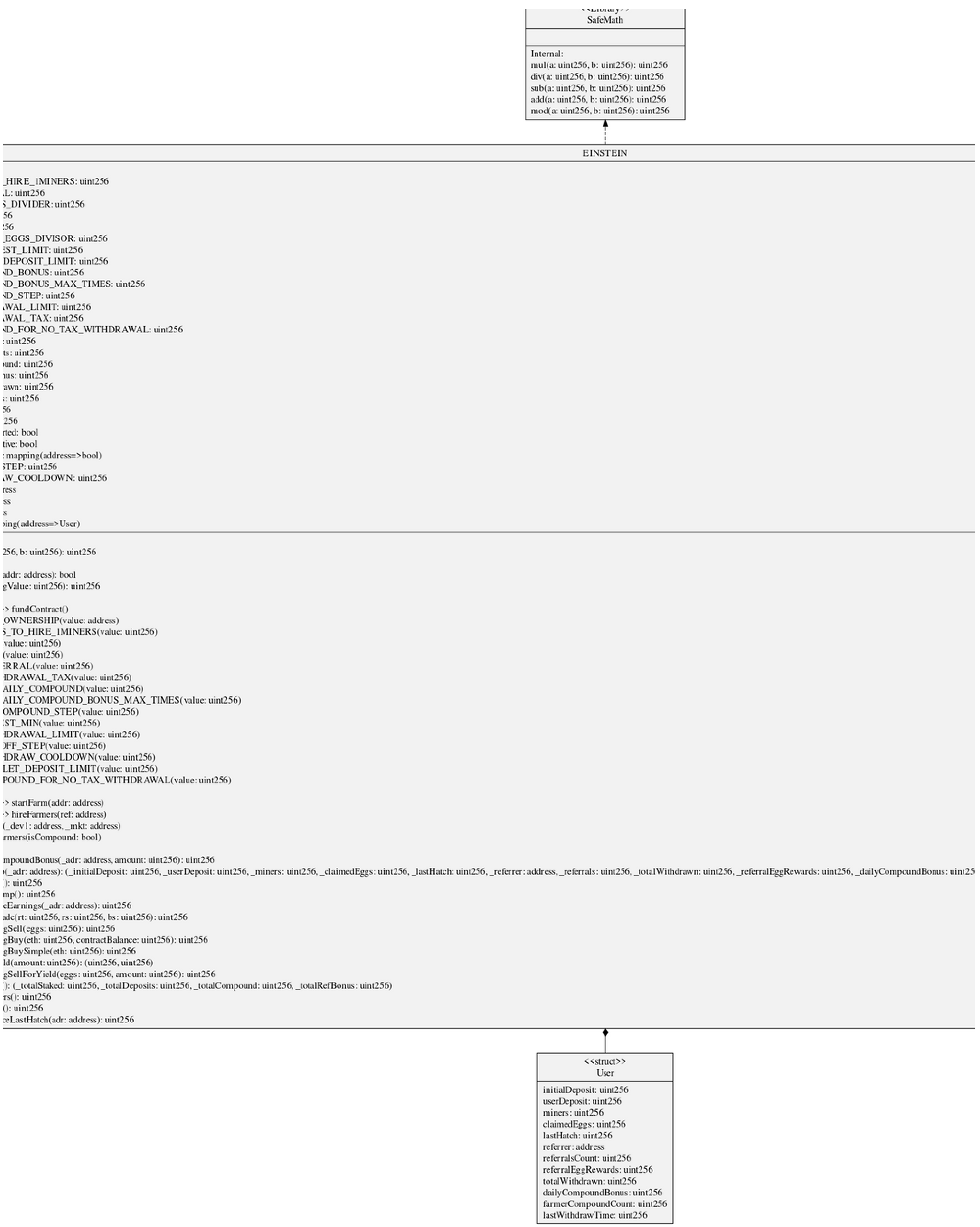
EGGS_TO_HIRE_1MINERS
REFERRAL
PERCENTS_DIVIDER
TAX
MKT
MARKET_EGGS_DIVISOR
MIN_INVEST_LIMIT
WALLET_DEPOSIT_LIMIT
COMPOUND_BONUS
COMPOUND_BONUS_MAX_TIMES
COMPOUND_STEP
WITHDRAWAL_LIMIT
WITHDRAWAL_TAX
COMPOUND_FOR_NO_TAX_WITHDRAWAL
totalStaked
totalDeposits
totalCompound
totalRefBonus
totalWithdrawn
marketEggs
contractStarted
blacklistActive
Blacklisted
CUTOFF_STEP
WITHDRAW_COOLDOWN
owner
dev1
mkt

SafeMath

Private Functions:

mul(uint256,uint256)
div(uint256,uint256)
sub(uint256,uint256)
add(uint256,uint256)
mod(uint256,uint256)

Unified Modeling Language(UML)



<<Library>>
SafeMath
Internal:
mul(a: uint256, b: uint256): uint256
div(a: uint256, b: uint256): uint256
sub(a: uint256, b: uint256): uint256
add(a: uint256, b: uint256): uint256
mod(a: uint256, b: uint256): uint256
EINSTEIN
_HIRE_1MINERS: uint256
L: uint256
S_DIVIDER: uint256
56
256
_EGGS_DIVISOR: uint256
EST_LIMIT: uint256
DEPOSIT_LIMIT: uint256
ND_BONUS: uint256
ND_BONUS_MAX_TIMES: uint256
ND_STEP: uint256
WAL_LIMIT: uint256
WAL_TAX: uint256
ND_FOR_NO_TAX_WITHDRAWAL: uint256
: uint256
ts: uint256
ound: uint256
nus: uint256
awn: uint256
: uint256
56
256
rted: bool
tive: bool
: mapping(address=>bool)
STEP: uint256
W_COOLDOWN: uint256
ress
ss
s
ping(address=>User)
56, b: uint256): uint256
addr: address): bool
gValue: uint256): uint256
> fundContract()
OWNERSHIP(value: address)
S_TO_HIRE_1MINERS(value: uint256)
value: uint256)
(value: uint256)
ERRAL(value: uint256)
HDRAWAL_TAX(value: uint256)
AILY_COMPOUND(value: uint256)
AILY_COMPOUND_BONUS_MAX_TIMES(value: uint256)
OMPOUND_STEP(value: uint256)
ST_MIN(value: uint256)
HDRAWAL_LIMIT(value: uint256)
OFF_STEP(value: uint256)
HDRAW_COOLDOWN(value: uint256)
LET_DEPOSIT_LIMIT(value: uint256)
POUND_FOR_NO_TAX_WITHDRAWAL(value: uint256)
> startFarm(addr: address)
> hireFarmers(ref: address)
(_dev1: address, _mkt: address)
rmers(isCompound: bool)
mpoundBonus(_adr: address, amount: uint256): uint256
o(_adr: address): (_initialDeposit: uint256, _userDeposit: uint256, _miners: uint256, _claimedEggs: uint256, _lastHatch: uint256, _referrer: address, _referrals: uint256, _totalWithdrawn: uint256, _referralEggRewards: uint256, _dailyCompoundBonus: uint25
): uint256
mp(): uint256
eEarnings(_adr: address): uint256
ade(rt: uint256, rs: uint256, bs: uint256): uint256
gSell(eggs: uint256): uint256
gBuy(eth: uint256, contractBalance: uint256): uint256
gBuySimple(eth: uint256): uint256
ld(amount: uint256): (uint256, uint256)
gSellForYield(eggs: uint256, amount: uint256): uint256
): (_totalStaked: uint256, _totalDeposits: uint256, _totalCompound: uint256, _totalRefBonus: uint256)
rs(): uint256
(): uint256
ceLastHatch(adr: address): uint256
<<struct>>
User
initialDeposit: uint256
userDeposit: uint256
miners: uint256
claimedEggs: uint256
lastHatch: uint256
referrer: address
referralsCount: uint256
referralEggRewards: uint256
totalWithdrawn: uint256
dailyCompoundBonus: uint256
farmerCompoundCount: uint256
lastWithdrawTime: uint256

Function ID Report

```
EINSTEIN:
+----------------------------------------------+------------+
|                     Name                     |     ID     |
+----------------------------------------------+------------+
|         constructor(address,address)         | 0x4525f804 |
|              startFarm(address)              | 0xda5d4cc5 |
|                fundContract()                | 0xbd097e21 |
|            hireMoreFarmers(bool)             | 0x18c819d8 |
|                 sellCrops()                  | 0x57386225 |
|             hireFarmers(address)             | 0x50cf1c7a |
|    getDailyCompoundBonus(address,uint256)    | 0x50637dbd |
|             getUserInfo(address)             | 0x6386c1c7 |
|                 getBalance()                 | 0x12065fe0 |
|                getTimeStamp()                | 0xda235b22 |
|        getAvailableEarnings(address)         | 0x64c03a5e |
|   calculateTrade(uint256,uint256,uint256)    | 0x229824c4 |
|           calculateEggSell(uint256)          | 0x8e316327 |
|       calculateEggBuy(uint256,uint256)       | 0x26fd8422 |
|        calculateEggBuySimple(uint256)        | 0x7e56fde5 |
|            getEggsYield(uint256)             | 0xbdd1ca27 |
|  calculateEggSellForYield(uint256,uint256)   | 0xcc3e9c78 |
|                getSiteInfo()                 | 0x4ce87053 |
|                getMyMiners()                 | 0x0a76e5ed |
|                 getMyEggs()                  | 0x43ce7422 |
|       getEggsSinceLastHatch(address)         | 0xd7c8843b |
|          CHANGE_OWNERSHIP(address)           | 0x2b039d0e |
|       PRC_EGGS_TO_HIRE_1MINERS(uint256)      | 0xe6dc9558 |
|               PRC_TAX(uint256)               | 0x298ea310 |
|               PRC_MKT(uint256)               | 0x1a7b8d4f |
|            PRC_REFERRAL(uint256)             | 0x570c2979 |
|         SET_WITHDRAWAL_TAX(uint256)          | 0xbfa9f304 |
|        BONUS_DAILY_COMPOUND(uint256)         | 0x6f969d28 |
| BONUS_DAILY_COMPOUND_BONUS_MAX_TIMES(uint256) | 0x9b9cb69f |
|         BONUS_COMPOUND_STEP(uint256)         | 0x959c95b3 |
|           SET_INVEST_MIN(uint256)            | 0x45f98c29 |
|        SET_WITHDRAWAL_LIMIT(uint256)         | 0xdecfee3a |
|           SET_CUTOFF_STEP(uint256)           | 0x7c8e4b4c |
|        SET_WITHDRAW_COOLDOWN(uint256)        | 0x45a6a6e0 |
|       SET_WALLET_DEPOSIT_LIMIT(uint256)      | 0x7ee28e3c |
|  SET_COMPOUND_FOR_NO_TAX_WITHDRAWAL(uint256)  | 0xe7576943 |
|            EGGS_TO_HIRE_1MINERS()            | 0x195a7339 |
|                  REFERRAL()                  | 0xc63568c7 |
|              PERCENTS_DIVIDER()              | 0x01c234a8 |
|                    TAX()                     | 0x68f58b03 |
|                    MKT()                     | 0x2bc82f7f |
|            MARKET_EGGS_DIVISOR()             | 0x59eec895 |
|              MIN_INVEST_LIMIT()              | 0xcd329fc3 |
|            WALLET_DEPOSIT_LIMIT()            | 0x1848b8dc |
|               COMPOUND_BONUS()               | 0xd7206d5d |
|
COMPOUND_BONUS_MAX_TIMES()         | 0xc688f0fb |
|              COMPOUND_STEP()              | 0x752a2628 |
|             WITHDRAWAL_LIMIT()            | 0x82ee0d1d |
|              WITHDRAWAL_TAX()             | 0x0420c98e |
|       COMPOUND_FOR_NO_TAX_WITHDRAWAL()      | 0xf6f62886 |
|               totalStaked()               | 0x817b1cd2 |
|              totalDeposits()              | 0x7d882097 |
|              totalCompound()              | 0x7db07c9d |
|              totalRefBonus()              | 0x69b11dd5 |
|              totalWithdrawn()             | 0x4b319713 |
|                marketEggs()               | 0x2e9392bb |
|             contractStarted()             | 0x333f57b3 |
|             blacklistActive()             | 0xb6e6fcf6 |
|            Blacklisted(address)           | 0xffa4e618 |
|               CUTOFF_STEP()               | 0x3578584f |
|            WITHDRAW_COOLDOWN()            | 0x950d91e9 |
|                  owner()                  | 0x8da5cb5b |
|                   dev1()                  | 0xa327c45d |
|                   mkt()                   | 0x7cc5b1e6 |
|               users(address)              | 0xa87430ba |
+-------------------------------------------+------------+
```

Conclusion

The smart contracts reviewed in this audit contain no high severity issues and all High to Medium issues have either been corrected or acknowledged.

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.

Ascendant

Ascendant

@ascendantproj

www.ascendant.finance.com

Ascendant