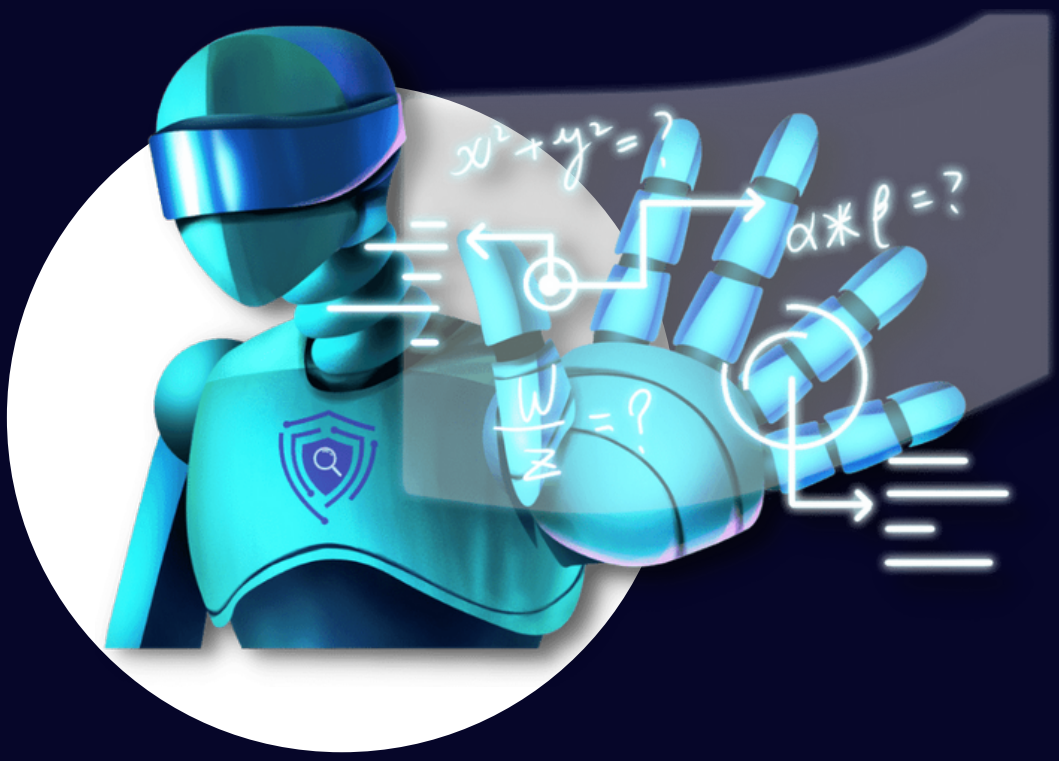
SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY

SAFUAUDIT

SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY



PROJECT: SOLAR FARM V2

DATE: June 30, 2022

www.safuaudit.com

INTRODUCTION

Client	Solar Farm V2
Language	Solidity
Contract address	0xd6E01219Ea4868b7ED1cB3d5B87791aD43FF97d0
Proxy	0xE52471bceA9c7d7e3FC2Da48ecf177C97E1107A8
Owner	0x480e02242b8f0fe94017836416091dcd907b4bff
Deployer	0x480e02242b8f0fe94017836416091dcd907b4bff
SHA1-Hash	3cc68b784ca050e93e503d051bdeebafdba4f280
Platform	Binance Smart Chain
Compiler	v0.8.4+commit.c7e474f2
Optimization	No with 200 runs
Website	https://app.solarfarm.finance/
Telegram	https://t.me/SolarFarmMinerOfficial
Twitter	https://twitter.com/SolarFarmMiner

TABLE OF CONTENTS

APPROACH

Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

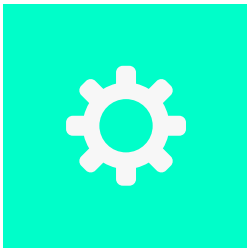
Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch, that lead to scams and rugpulls.

Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability



Tools

- Remix IDE
- Mythril
- Open Zeppelin Code Analyzer
- Solidity Code Complier
- Hardhat

RISK CLASSIFICATION

CRITICAL

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

MEDIUM

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

MINOR

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

INFORMATIONAL

Information level is to offer suggestions for improvement of efficacity or security for features with a risk free factor.

OVERVIEW

Fees

- 6% Fee

Fees privileges

- Can't set fees above 10%

Ownership

- Owned - Proxy Contract

Minting

- Not Applicable

Max Tx Amount

- Not Applicable

Pause function

- Not Applicable

Blacklist

- Can blacklist

Other privileges

- Can Whitelist

CONTRACT INSPECTION 🔎

Tested Contract File:

1. TransparentUpgradeableProxy - 0xE52471bceA9c7d7e3FC2Da48ecf177C97E1107A8
2. SolarFarmV2 - 0xd6E01219Ea4868b7ED1cB3d5B87791aD43FF97d0

TransparentUpgradeableProxy contract delegates calls to SolarFarmV2 contract.

1. Proxy Contract: 0xE52471bceA9c7d7e3FC2Da48ecf177C97E1107A8

Imported contracts or frameworks used:

```
openzeppelin/contracts/proxy/beacon/BeaconProxy.sol
openzeppelin/contracts/proxy/beacon/UpgradeableBeacon.sol
openzeppelin/contracts/proxy/ERC1967/ERC1967Proxy.sol
openzeppelin/contracts/proxy/transparent/TransparentUpgradeableProxy.sol
openzeppelin/contracts/proxy/transparent/ProxyAdmin.sol
```

```
|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| TransparentUpgradeableProxy.sol | 1e4c1803ff6dd88d0a134d3a7cd8e4f7b91d25b2 |

| **AdminUpgradeabilityProxy** | Implementation | TransparentUpgradeableProxy |||
| └ | <Constructor> | Public ❗ |  ⛽  | TransparentUpgradeableProxy |
```

2. SolarFarmV2 - 0xd6E01219Ea4868b7ED1cB3d5B87791aD43FF97d0

Imported contracts or frameworks used:

```
openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol
utils/ContextUpgradeable.sol
proxy/utils/Initializable.sol
```

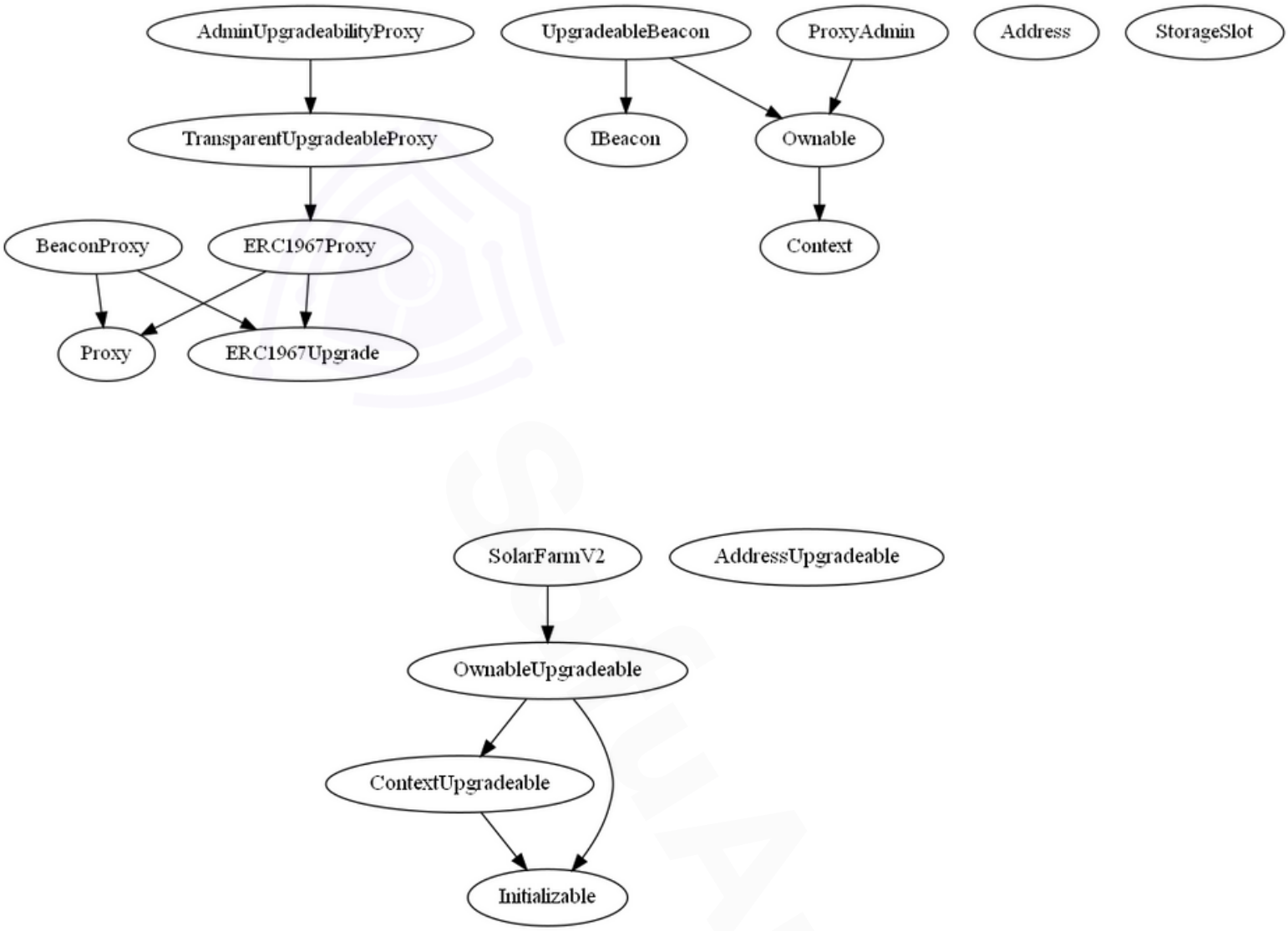
File Name	SHA-1 Hash
SolarFarmV2.sol	3cc68b784ca050e93e503d051bdeebafdba4f280

SolarFarmV2	Implementation	OwnableUpgradeable		
└	initialize	External ❗	🛑	initializer
└	getTotalValueLocked	Public ❗		NO❗
└	getSolarPanels	Public ❗		NO❗
└	getRemainingAllowance	Public ❗		NO❗
└	getFreshValue	Public ❗		NO❗
└	getAmountSold	Public ❗		NO❗
└	getAmountDeposited	Public ❗		NO❗
└	getSellsCount	Public ❗		NO❗
└	getTotalCompoundsCount	Public ❗		NO❗
└	getCurrentCompoundsCount	Public ❗		NO❗
└	getLastSellTimestamp	Public ❗		NO❗
└	getLastCompoundTimestamp	Public ❗		NO❗
└	getCompoundBonusTier	Public ❗		NO❗
└	getReferrer	Public ❗		NO❗
└	getReferees	Public ❗		NO❗
└	getBotStatus	Public ❗		NO❗
└	checkBaseAbuseStatus	Public ❗		NO❗
└	checkDumpAbuseStatus	Public ❗		NO❗
└	checkSpamAbuseStatus	Public ❗		NO❗
└	checkRewardsBalance	Public ❗		NO❗
└	checkPowerTotal	Public ❗		NO❗
└	checkFreshPower	Public ❗		NO❗
└	checkMinimum	Private 🔐		
└	computeFraction	Private 🔐		
└	computeTrade	Private 🔐		
└	computeBuyTrade	Private 🔐		
└	computeSellTrade	Private 🔐		
└	computeSimulatedBuy	Public ❗		NO❗
└	computeSimulatedSell	Public ❗		NO❗
└	setBuyDampener	External ❗	🛑	onlyOwner
└	setSellDampener	External ❗	🛑	onlyOwner
└	setCompoundDampener	External ❗	🛑	onlyOwner
└	setGridFee	External ❗	🛑	onlyOwner
└	buyPanels	External ❗	💵	NO❗

└	sellPower	External ❗	🛑	NO❗
└	compoundPower	Public ❗	🛑	NO❗
└	catchAbuser	External ❗	🛑	onlyOwner
└	freeInnocent	External ❗	🛑	onlyOwner
└	activateMiner	External ❗	🧱	onlyOwner
└	releaseGuard	External ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	🧱	NO❗

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable
🔐	Private function
🔒	Internal function
NO❗	Function has no modifier

INHERITANCE TREE



Inheritance is a feature of the object-oriented programming language. It is a way of extending the functionality of a program, used to separate the code, reduces the dependency, and increases the re-usability of the existing code. Solidity supports inheritance between smart contracts, where multiple contracts can be inherited into a single contract.

MANUAL FUNCTIONS ANALYSIS

The contract is verified to check if functions do and work as they should and malicious code is not inserted.

	Tested	Result
Transfer	Yes	N/A
Total Supply	Yes	N/A
Buy Back	Yes	N/A
Burn	Yes	N/A
Mint	Yes	N/A
Rebase	Yes	N/A
Pause	Yes	N/A
Blacklist	Yes	Medium
Lock	Yes	N/A
Max Transaction	Yes	N/A
Transfer Ownership	Yes	Passed
Renounce Ownership	Yes	Passed

VULNERABILITIES TEST

ID	Description	
V-01	Function Default Visibility	Passed
V-02	Integer Overflow and Underflow	Passed
V-03	Outdated Compiler Version	Passed
V-04	Floating Pragma	Minor
V-05	Unchecked Call Return Value	Passed
V-06	Unprotected Ether Withdrawal	Passed
V-07	Unprotected SELF-DESTRUCT Instruction	Passed
V-08	Re-entrancy	Passed
V-09	State Variable Default Visibility	Passed
V-10	Uninitialized Storage Pointer	Passed
V-11	Assert Violation	Passed
V-12	Use of Deprecated Solidity Functions	Passed
V-13	Delegate Call to Untrusted Callee	Passed
V-14	DoS with Failed Call	Medium
V-15	Transaction Order Dependence	Passed
V-16	Authorization through tx.origin	Passed
V-17	Block values as a proxy for time	Minor

V-18	Signature Malleability	**Passed**
V-19	Incorrect Constructor Name	**Passed**
V-20	Shadowing State Variables	**Passed**
V-21	Weak Sources of Randomness from Chain Attributes	**Passed**
V-22	Missing Protection against Signature Replay Attacks	**Passed**
V-23	Lack of Proper Signature Verification	**Passed**
V-24	Requirement Violation	**Passed**
V-25	Write to Arbitrary Storage Location	**Passed**
V-26	Incorrect Inheritance Order	**Passed**
V-27	Insufficient Gas Griefing	**Passed**
V-28	Arbitrary Jump with Function Type Variable	**Passed**
V-29	DoS With Block Gas Limit	**Passed**
V-30	Typographical Error	**Passed**
V-31	Right-To-Left-Override control character (U+202E)	**Passed**
V-32	Presence of unused variables	**Passed**
V-33	Unexpected Ether balance	**Passed**
V-34	Hash Collisions With Multiple Variable Length Arguments	**Passed**
V-35	Message call with the hardcoded gas amount	**Passed**
V-36	Code With No Effects (Irrelevant/Dead Code)	**Passed**
V-37	Unencrypted Private Data On-Chain	**Passed**

FINDINGS

ID	Category	Issue	Severity
V-14	Vulnerabilities	DoS with Failed Call	Medium
LI-01	Logical Issue	Auto Blacklisting during startup	Medium
V-04	Vulnerabilities	Floating Pragma is set	Minor
V-17	Vulnerabilities	Block values as a proxy for time	Minor
CE-OF	Centralization	Owner Accessible Functions	Minor

V-14: DoS with Failed Call

Description

Multiple calls are executed in the same transaction.

External calls can fail accidentally or deliberately, which can cause a DoS condition in the contract. To minimize the damage caused by such failures, it is better to isolate each external call into its own transaction that can be initiated by the recipient of the call. This is especially relevant for payments, where it is better to let users withdraw funds rather than push funds to them automatically (this also reduces the chance of problems with the gas limit).

Recommendation

It is recommended to follow call best practices:

- Avoid combining multiple calls in a single transaction, especially when calls are executed as part of a loop
- Always assume that external calls can fail
- Implement the contract logic to handle failed calls

LI-01: Auto Blacklisting during startup

Description

For the miner to be fully initialized, owner has to activate miner through **activateMiner()** function and **releaseGuard()** function.
However, between the **activateMiner()** call and **releaseGuard()**, if a user (wallet) connects and buys a panel, he will be automatically blacklisted.

Recommendation

- We advise the client to:
  - carefully advise everyone to invest only when miner is fully activated.
  - activate and release guard at once.
  - whitelist real accounts through **freeInnocent()** function.

V-04: Floating Pragma is set

Description

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation

- Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

V-17: Block values as a proxy for time

Description

Contracts often need access to time values to perform certain types of functionality. Values such as block.timestamp, and block.number can give you a sense of the current time or a time delta, however, they are not safe to use for most purposes.

Recommendation

- block values are not precise, and the use of them can lead to unexpected effects. Alternatively, developers may make use oracles.

CE-OF: Owner Accessible Functions

Description

The owner has the permission through onlyOwner modifier to the following:

1. catchAbuser()
2. freeInnocent()
3. activateMiner()
4. releaseGuard()
5. setGridFee()
6. setCompoundDampener()
7. setSellDampener()
8. setBuyDampener()

The role OnlyOwner has authority over the above functions that can manipulate the project functionality without restrictions. Any compromise to the owner account may allow a hacker to take advantage of this authority.

Recommendation

- We advise the client to carefully manage the privilege accounts' private key to avoid any potential risks of being hacked.

WEBSITE

Website	https://app.solarfarm.finance/
Domain Registry	http://www.godaddy.com/
Domain Expiry Date	2023-04-24
Response Code	200
SSL Checker and HTTPS Test	Passed
Deprecated HTML tags	Passed
Robots.txt	Passed
Sitemap Test	Informative
SEO Friendly URL	Passed
Responsive Test	Passed
JS Error Test	Passed
Console Errors Test	Passed
Site Loading Speed Test	3.02 seconds - Passed
HTTP2 Test	Passed
Safe Browsing Test	Passed

DISCLAIMER

SafuAudit.com is not a financial institution and the information provided on this website does not constitute investment advice, financial advice, trading advice, or any other sort of advice. You should not treat any of the website's content as such. Investing in crypto assets carries a high level of risk and does not hold guarantees for not sustaining financial loss due to their volatility.

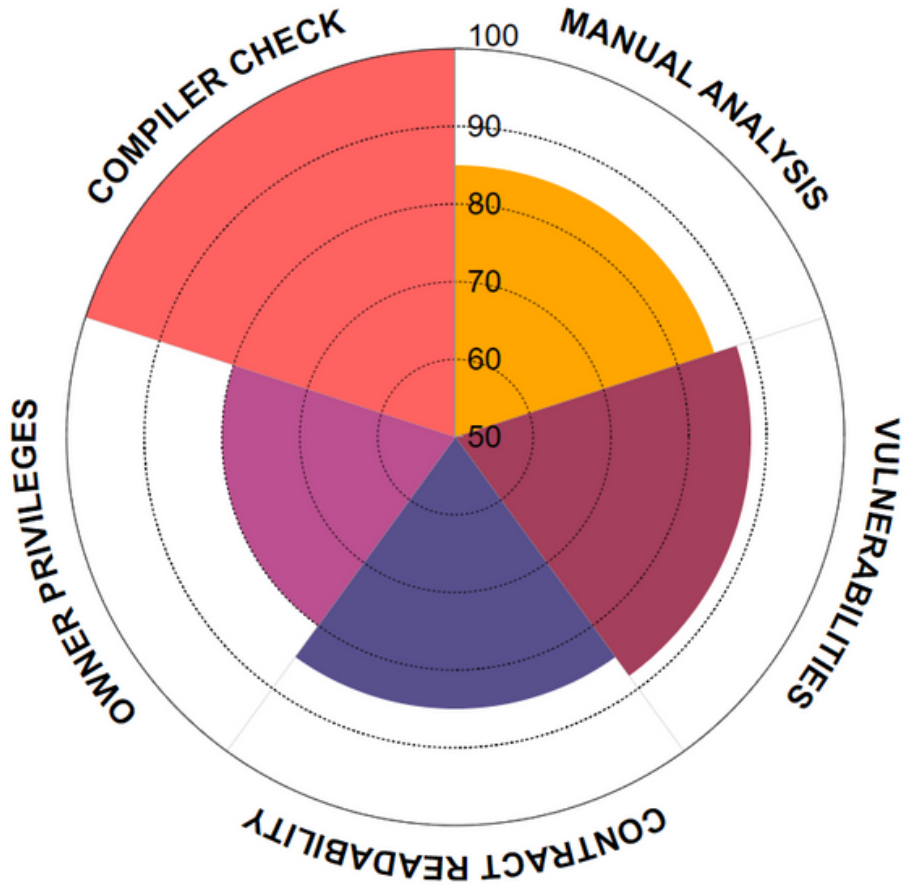
Accuracy of Information

SafuAudit will strive to ensure the accuracy of the information listed on this website although it will not hold any responsibility for any missing or wrong information. SafuAudit provides all information as is. You understand that you are using any and all information available here at your own risk. Any use or reliance on our content and services is solely at your own risk and discretion.

The purpose of the audit is to analyze the on-chain smart contract source code and to provide a basic overview of the project.

While we have used all the information available to us for this straightforward investigation, you should not rely on this report only — we recommend proceeding with several independent audits Be aware that smart contracts deployed on a blockchain aren't secured enough against external vulnerability or a hack. Be aware that active smart contract owner privileges constitute an elevated impact on the smart contract safety and security. Therefore, SafuAudit does not guarantee the explicit security of the audited smart contract. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

RATING



Manual Analysis

Vulnerabilities

Contract Readability

Owner Privileges

Compiler Check

Final Score: **87.6**

CONCLUSION

Project Solar Farm does not contain any severe issues. However it uses proxy contract. This allows the contract/part of contract to be swapped out at any time without changing the address of the primary entry point.

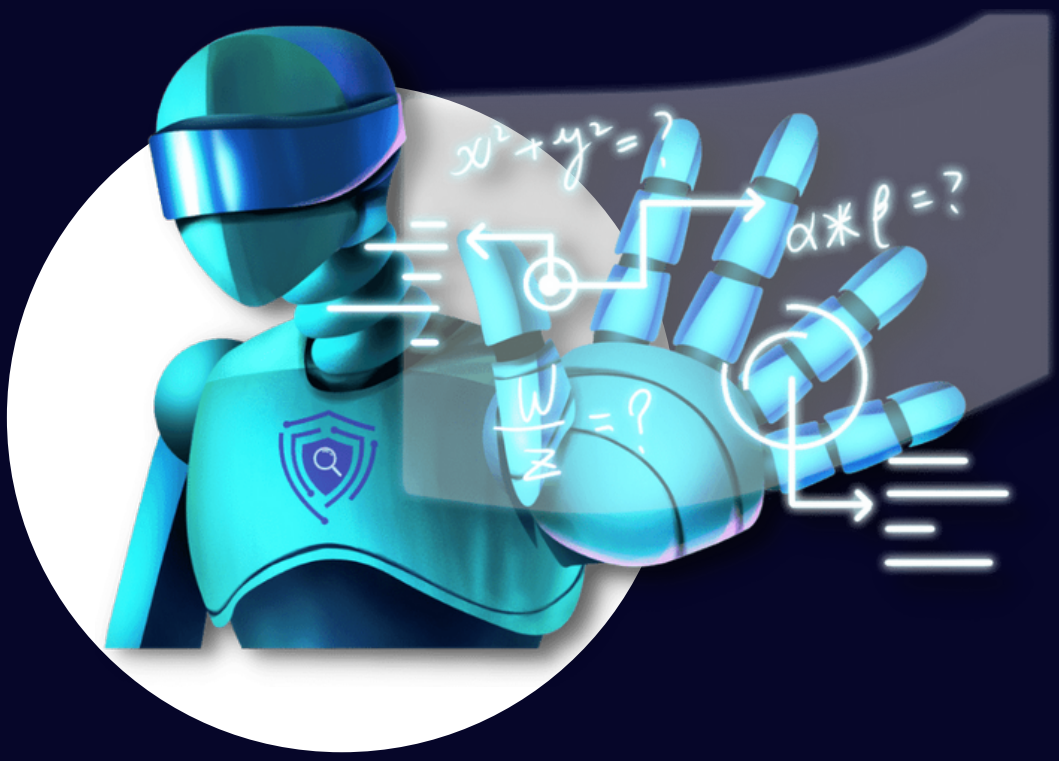
In general, a proxy contract is not safe if the owner is not trusted (they can swap out the implementation at any time). If owners are trusted however, proxies are recommended (for a better security and bug fixes).

SafuAudit has tested the security based on manual and automated tests. Please note that we don't offer any warranties for the business model.

SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY

SAFUAUDIT

SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY



"Only in growth, reform, and change, paradoxically enough, is true security to be found."

www.safuaudit.com