

# CHENINGA AUDITS



Security Assessment

**Safe Universe Staking**

September 19, 2022



# Table of Contents

## **1 Audit Summary**

## **2 Project Overview**

2.1 Token Summary

2.2 Risk Analysis Summary

2.3 Main Contract Assessed

## **3 Smart Contract Risk Checks**

3.1 Mint Check

3.2 Fees Check

3.3 Blacklist Check

3.4 MaxTx Check

3.5 Pause Trade Check

## **4 Contract Ownership**

## **5 Liquidity Ownership**

## **6 KYC Check**

## **7 Smart Contract Vulnerability Checks**

7.1 Smart Contract Vulnerability Details

7.2 Smart Contract Inheritance Details

7.3 Smart Contract Privileged Functions

## **8 Assessment Results and Notes(Important)**

## **9 Social Media Check(Informational)**

## **10 Technical Findings Summary**

## **11 Disclaimer**



# Assessment Summary

This report has been prepared for Safe Universe Staking on the Binance Smart Chain network. CFGNINJA provides both client-centered and user-centered examination of the smart contracts and their current status when applicable. This report represents the security assessment made to find issues and vulnerabilities on the source code along with the current liquidity and token holder statistics of the protocol.

A comprehensive examination has been performed, utilizing Cross Referencing, Static Analysis, In-House Security Tools, and line-by-line Manual Review.






The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Inspecting liquidity and holders statistics to inform the current status to both users and client when applicable.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Verifying contract functions that allow trusted and/or untrusted actors to mint, lock, pause, and transfer assets.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders
- Thorough line-by-line manual review of the entire codebase by industry experts.








# Technical Findings Summary

## Classification of Risk

Severity	Description
 Critical	risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
 Major	risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
 Medium	risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
 Minor	risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
 Informational	errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

## Findings

Severity	Found	Pending	Resolved
 Critical	0	0	0
 Major	0	1	0
 Medium	0	1	0
 Minor	0	0	0
 Informational	0	2	0
Total	4	4	0



# Project Overview

## Token Summary

Parameter	Result
Address	0xd972225785d907116824171A04271Ee8c02d622f
Name	Safe Universe
Token Tracker	Safe Universe (Farmv2)
Decimals	0
Supply	0
Platform	Binance Smart Chain
compiler	v0.8.16+commit.07a7930e
Contract Name	FarmV2
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	<a href="https://bscscan.com/address/0xd972225785d907116824171A04271Ee8c02d622f#code">https://bscscan.com/address/0xd972225785d907116824171A04271Ee8c02d622f#code</a>
Payment Tx	0x35904ec3561d05f66d0e72b8320719ce514f249b28c0e77088b2386c98bb2992



# Project Overview

## Risk Analysis Summary

Parameter	Result
Buy Tax	0%
Sale Tax	0%
Is honeypot?	Clean
Can edit tax?	No
Is anti whale?	No
Is blacklisted?	No
Is whitelisted?	No
Holders	Clean
Security Score	92/100
Auditor Score	90/100
Confidence Level	Pass

The following quick summary has been added to the project overview, however there are more details about the audit and their results please read every details.





## Main Contract Assessed Contract Name

Name	Contract	Live
Safe Universe	0xd972225785d907116824171A04271Ee8c02d622f	Yes

## TestNet Contract Assessed Contract Name

Name	Contract	Live
Safe Universe	0x8a007310c8f88006bc0531bac05FB2C60C0575c4	Yes

## Solidity Code Provided

SolID	File Sha-1	FileName
SafeUniverseStake	836e817c8ca383bc342d54a8817b0f844972bc2e	FarmV2.sol



# Call Graph

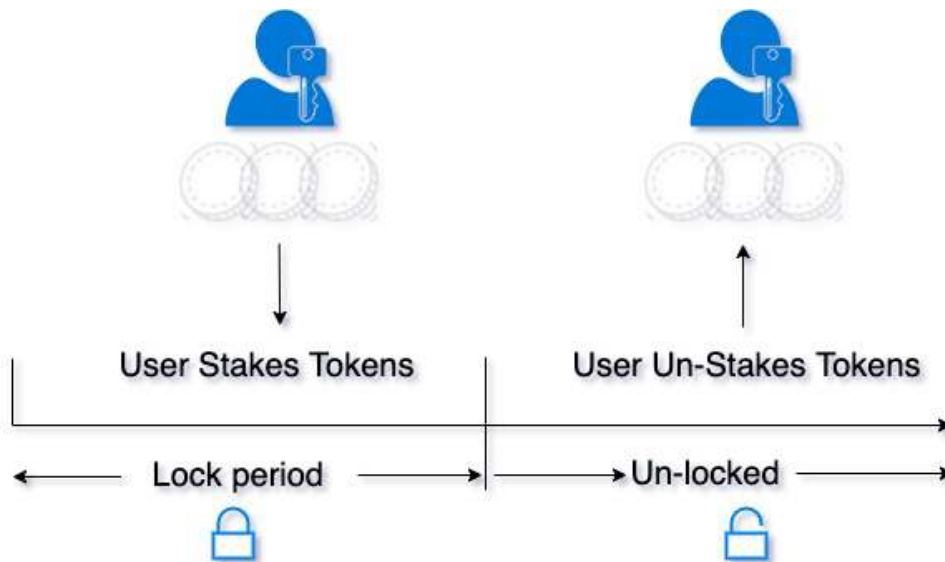
The contract for Safe Universe has the following call graph structure





# What is a Staking Contract

A smart contract which allows users to stake and un-stake a specified ERC20 token. Staked tokens are locked for a specific length of time (set by the contract owner at the outset). Once the time period has elapsed, the user can remove their tokens again.

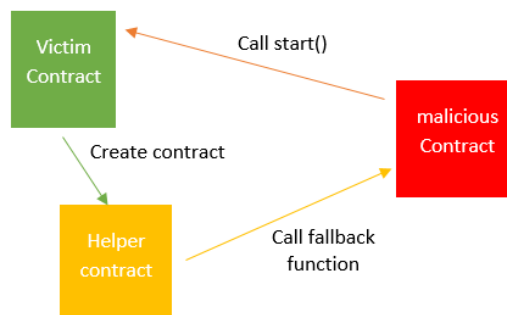


# Reentrancy Check

The Project Owners of Safe Universe have implemented Reentrancy Guard Library

The Team has done a great job to avoid potential reentrancy issues in the contract.

You can read more about the reentrancy library used.  
[ReentrancyGuard](#)



# KYC Information

**The Project Owners of Safe Universe is not KYC. .**

**The owner wallet has the power to call the functions displayed on the privileged functions chart below, if the owner wallet is compromised this privileges could be exploited.**

**We recommend the team to renounce ownership at the right timing if possible, or gradually migrate to a timelock with governing functionalities in respect of transparency and safety considerations.**

KYC Information Notes:

Auditor Notes:

Project Owner Notes:



# Smart Contract Vulnerability Checks

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	FarmV2.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	FarmV2.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	FarmV2.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	FarmV2.sol	L: 2 C: 1
SWC-104	Pass	Unchecked Call Return Value.	FarmV2.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	FarmV2.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	FarmV2.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	FarmV2.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	FarmV2.sol	L: 290 C: 12
SWC-109	Pass	Uninitialized Storage Pointer.	FarmV2.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	FarmV2.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	FarmV2.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	FarmV2.sol	L: 0 C: 0



ID	Severity	Name	File	location
SWC-113	Pass	Multiple calls are executed in the same transaction.	FarmV2.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	FarmV2.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	FarmV2.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	FarmV2.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	FarmV2.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	FarmV2.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	FarmV2.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randomness.	FarmV2.sol	L: 421 C: 12, L: 496 C: 22, L: 497 C: 28
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	FarmV2.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	FarmV2.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	FarmV2.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	FarmV2.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	FarmV2.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	FarmV2.sol	L: 0 C: 0



ID	Severity	Name	File	location
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	FarmV2.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	FarmV2.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	FarmV2.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	FarmV2.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	FarmV2.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	FarmV2.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	FarmV2.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	FarmV2.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	FarmV2.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	FarmV2.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry standard security scanning tool



# Smart Contract Vulnerability Details

## SWC-103 - Floating Pragma.

### CWE-664: Improper Control of a Resource Through its Lifetime.

#### References:

#### 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.

#### Remediation:

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

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

#### References:

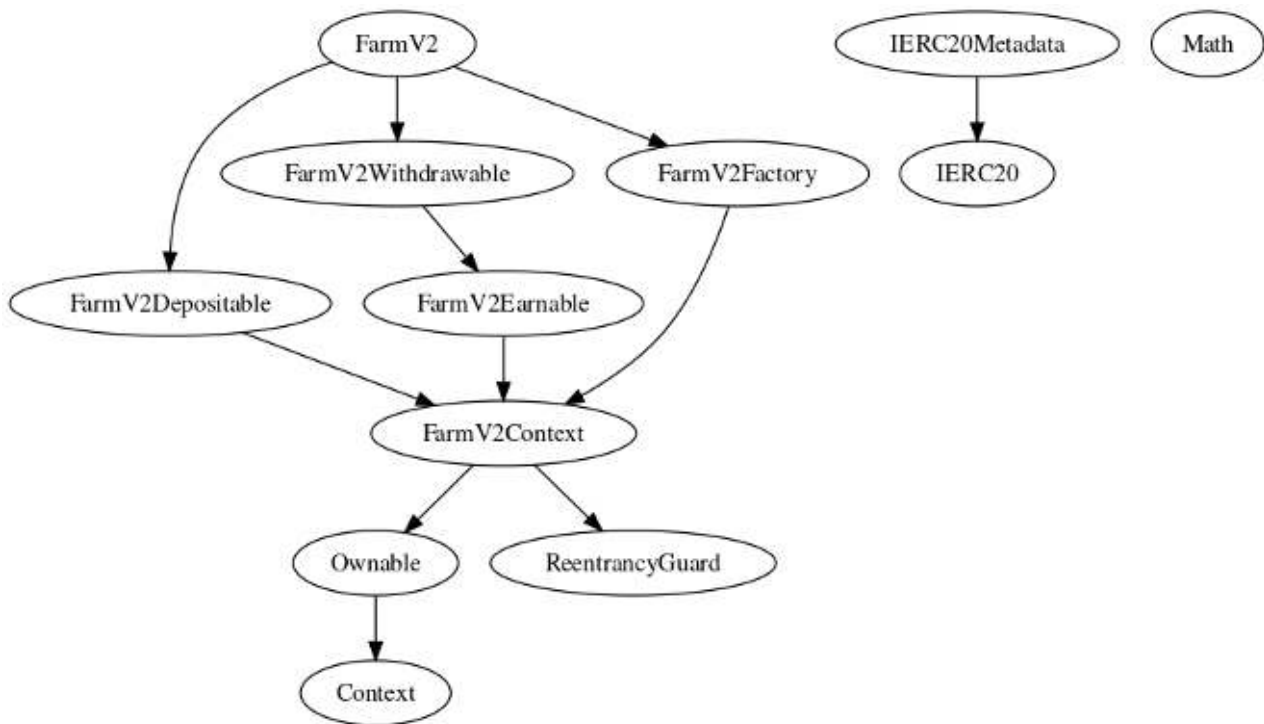
Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.





# Inheritance

The contract for Safe Universe has the following inheritance structure



## Privileged Functions (onlyOwner)

Function Name	Parameters	Visibility
setLockStatus	none	public
setMaxDeposit	none	public
setRewardsRate	none	public
setStartTime	none	public





## Assessment Results

- The following Security assessments have been executed for Safe Universe Staking Contract.
- We noticed the contract have a few redundant areas and a few changes that may need to be revisit by the team.
- This will be a place holder note to inform the customer of our current status until they answer and review our suggestions on the staking contract itself.
- The modification of the Math library is a concern and need to be validated and confirmed before moving forward.

## Audit Fail



## Farmv2-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	 Informational	FarmV2.sol: 0,0	 Pending

### Description

The following functions are declared as public and are not invoked in any of the contracts contained within the projects scope:

Function Name	Parameters	Visibility
apr		public
duration		public
decimals		public
isLocked		public
stakeToken		public
totalStaked		public
rewardsToken		public
rewardsRate		public
rewardsPool		public
available		public
maxDeposit		public

The functions that are never called internally within the contract should have external visibility



## Remediation



We advise that the functions' visibility specifiers are set to external and the array-based arguments change their data location from memory to calldata , optimizing the gas cost of the function.

References:

external vs public best practices.



## Farmv2-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	 Minor	FarmV2.sol: 17,16	 Pending

### Description

The given input is missing the check for the non-zero address.



### Remediation

We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
...  
    require(receiver != address(0), "Receiver is the zero address");  
...
```



## Farmv2-11 | Modification of Math Library.

Category	Severity	Location	Status
Security	 Major	FarmV2.sol: 20,19	 Pending

### Description

During our review we noticed the math library was changed from the default library from OpenZeppelin the original library can be found in. <https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/math/Math.sol> and the changes on the math.sol are on the Max function this can create an potential issue on the smart contract `function max(uint256 a, uint256 b) internal pure returns (uint256) {return a >= b ? a : b;}`

### Remediation

restore this library to the default one.

### Project Action

Pending Customer Response





# Social Media Checks

Social Media	URL	Result
Twitter	<a href="https://twitter.com/Safe_Universe">https://twitter.com/Safe_Universe</a>	Pass
Instagram		N/a
Website	<a href="http://safeuniverse.io">http://safeuniverse.io</a>	Pass
Telegram	<a href="https://t.me/SafeUniverseGlobalOfficial">https://t.me/SafeUniverseGlobalOfficial</a>	Pass

We recommend to have 3 or more social media sources including a completed working websites.

**Social Media Information Notes:**

**Auditor Notes:** undefined

**Project Owner Notes:** No other social media



# Appendix

## Finding Categories

### Centralization / Privilege

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

### Gas Optimization

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

### Logical Issue

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

### Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

### Volatile Code

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

### Coding Style

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

### Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different requirements on the input variables than a setter function.

### Coding Best Practices

ERC 20 Coding Standards are a set of rules that each developer should follow to ensure the code meets a set of criteria and is readable by all the developers.



## Disclaimer

CFGNINJA 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 advocacy 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.

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 CFGNINJA is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will CFGNINJA 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.

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

