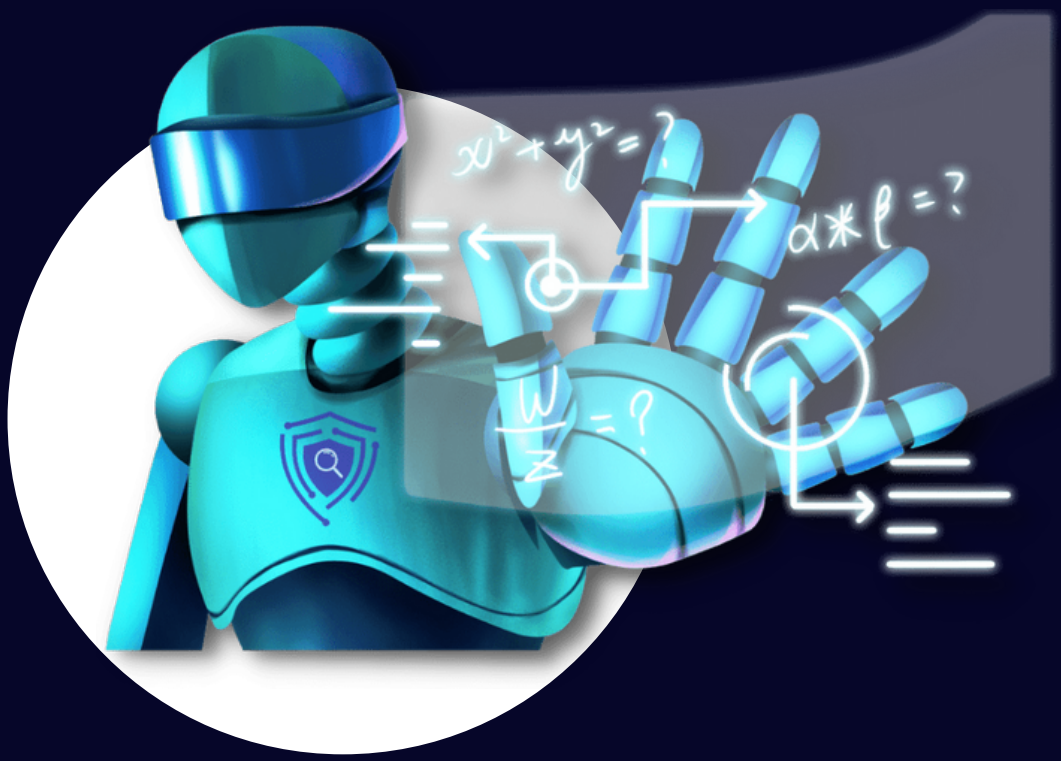
SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY

SAFUAUDIT

SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY



PROJECT: **HULK**

DATE: May 24, 2022

www.safuaudit.com

INTRODUCTION

Client	$HULK ($HULK)
Language	Solidity
Contract address	0x36f04EDb7a54E4f0075391879226C060bA09fE4C
Owner	0x9ab41C4290CC5d57c5F49187b141ad16f937630B
Deployer	0x9ab41C4290CC5d57c5F49187b141ad16f937630B
SHA1-Hash	90d133d05ac3fdc88acad4e2131bc62c8bc93f7f
Decimals	9
Supply	1,000,000,000
Platform	Binance Smart Chain
Compiler	v0.8.4+commit.c7e474f2
Optimization	Yes with 200 runs
Website	https://marveltech.crd.co/
Telegram	https://t.me/+qfZM9nWAc7NkNDI0
Twitter	https://twitter.com/HULK_TOKEN

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.

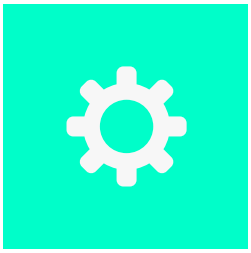
Audit Goals

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

- Buy Fees: 10%
- Sell Fees: 10%

Fees privileges

- Can set fees up to 100%

Ownership

- Owned

Minting

- No mint function

Max Tx Amount

- Can't set max Tx amount

Pause function

- Can't pause trading

Blacklist

- Can't blacklist

Other privileges

- Can exclude from fees
- Can exclude from rewards

CONTRACT INSPECTION

Imported contracts or frameworks used:

IERC20	Interface		
Context	Implementation		
Ownable	Implementation	Context	
SafeMath	Library		
Address	Library		
IUniswapV2Router01	Interface		
IUniswapV2Router02	Interface	IUniswapV2Router01	
IUniswapV2Factory	Interface		
IPinkAntiBot	Interface		
BaseToken	Implementation		

Tested Contract File:

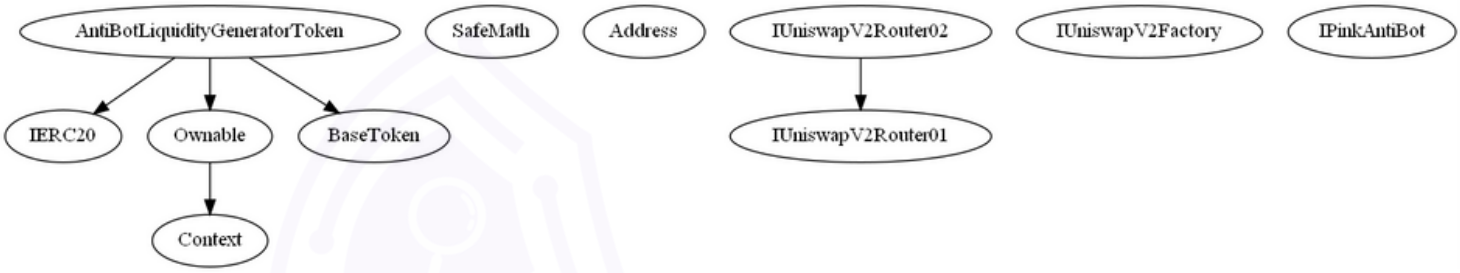
File Name	SHA-1 Hash
Hulk.sol	90d133d05ac3fdc88acad4e2131bc62c8bc93f7f

AntiBotLiquidityGeneratorToken	Implementation	IERC20, Ownable, BaseToken			
└	<Constructor>	Public ❗	💸	NO❗	
└	setEnableAntiBot	External ❗	🛑	onlyOwner	
└	name	Public ❗		NO❗	
└	symbol	Public ❗		NO❗	
└	decimals	Public ❗		NO❗	
└	totalSupply	Public ❗		NO❗	
└	balanceOf	Public ❗		NO❗	
└	transfer	Public ❗	🛑	NO❗	
└	allowance	Public ❗		NO❗	
└	approve	Public ❗	🛑	NO❗	
└	transferFrom	Public ❗	🛑	NO❗	
└	increaseAllowance	Public ❗	🛑	NO❗	
└	decreaseAllowance	Public ❗	🛑	NO❗	
└	isExcludedFromReward	Public ❗		NO❗	
└	totalFees	Public ❗		NO❗	
└	deliver	Public ❗	🛑	NO❗	
└	reflectionFromToken	Public ❗		NO❗	
└	tokenFromReflection	Public ❗		NO❗	
└	excludeFromReward	Public ❗	🛑	onlyOwner	
└	includeInReward	External ❗	🛑	onlyOwner	
└	_transferBothExcluded	Private 🔐	🛑		
└	excludeFromFee	Public ❗	🛑	onlyOwner	
└	includeInFee	Public ❗	🛑	onlyOwner	
└	setTaxFeePercent	External ❗	🛑	onlyOwner	
└	setLiquidityFeePercent	External ❗	🛑	onlyOwner	
└	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner	
└	<Receive Ether>	External ❗	💸	NO❗	
└	_reflectFee	Private 🔐	🛑		
└	_getValues	Private 🔐			
└	_getTValues	Private 🔐			
└	_getRValues	Private 🔐			
└	_getRate	Private 🔐			

└	_getCurrentSupply	Private 🔐		
└	_takeLiquidity	Private 🔐	🛑	
└	_takeCharityFee	Private 🔐	🛑	
└	calculateTaxFee	Private 🔐		
└	calculateLiquidityFee	Private 🔐		
└	calculateCharityFee	Private 🔐		
└	removeAllFee	Private 🔐	🛑	
└	restoreAllFee	Private 🔐	🛑	
└	isExcludedFromFee	Public ❗		NO❗
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForEth	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	
└	_transferStandard	Private 🔐	🛑	
└	_transferToExcluded	Private 🔐	🛑	
└	_transferFromExcluded	Private 🔐	🛑	

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	Passed
Total Supply	Yes	Passed
Buy Back	Yes	N/A
Burn	Yes	N/A
Mint	Yes	N/A
Rebase	Yes	N/A
Pause	Yes	N/A
Blacklist	Yes	N/A
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	Passed
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	Minor
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	Passed
V-15	Transaction Order Dependence	Passed
V-16	Authorization through tx.origin	Passed
V-17	Block values as a proxy for time	Passed

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
CE-01	Centralization	Fees up to 100%	Medium
CE-OF	Centralization	Owner Accessible Functions	Minor
V-01	Vulnerabilities	State Variable Default Visibility	Minor
CS-01	Coding Standards	Dead Code	Informational

V-09: State Variable Default Visibility

Line #973

```
bool inSwapAndLiquify;
```

Description

Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

Recommandation

Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

CE-OF: Owner Accessible Functions

Description

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

1. renounceOwnership()
2. transferOwnership()
3. setEnableAntiBot()
4. excludeFromReward()
5. includeInReward()
6. excludeFromFee()
7. includeInFee()
8. setTaxFeePercent()
9. setLiquidityFeePercent()
10. setSwapAndLiquifyEnabled()

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.

Recommandation

- We advise the client to carefully manage the privilege accounts' private key to avoid any potential risks of being hacked.
- Renounce Ownership at some point in time.

CE-01: Centralization Risk

CE-01: Fees up to 100%
Line #1280-1283, 1285-1294

```
function setTaxFeePercent(uint256 taxFeeBps) external onlyOwner {
    require(taxFeeBps >= 0 && taxFeeBps <= 10**4, "Invalid bps");
    _taxFee = taxFeeBps;
}

function setLiquidityFeePercent(uint256 liquidityFeeBps)
    external
    onlyOwner
{
    require(
        liquidityFeeBps >= 0 && liquidityFeeBps <= 10**4,
        "Invalid bps"
    );
    _liquidityFee = liquidityFeeBps;
}
```

Description

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.

Recommandation

- Functions need to have at least one restriction.
  For example: setting fees to a maximum of 25%, setting tokenPriceUsdt with a limit

CS-01: Coding Standards

Line # 125: Dead Code

Description

Address library functions: *functionCall(), functionCallWithValue(), functionDelegateCall(), functionStaticCall(), isContract(), sendValue(), verifyCallResult()* are never used.

SafeMath library functions: *div(), mod(), tryAdd(), tryDiv(), tryMod, tryMul(), trySub()* are never used.

Recommandation

- Remove unused functions for code clarity and easier review.

GOOD PRACTICES

- The owner cannot mint new tokens after deployment
- The owner cannot stop or pause the contract
- The owner cannot set a transaction limit
- The smart contract utilizes "SafeMath" to prevent overflows

```
function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
    unchecked {
        uint256 c = a + b;
        if (c < a) return (false, 0);
        return (true, c);
    }
}
function trySub(uint256 a, uint256 b) internal pure returns (bool, uint256) {
    unchecked {
        if (b > a) return (false, 0);
        return (true, a - b);
    }
}
function tryMul(uint256 a, uint256 b) internal pure returns (bool, uint256) {
    unchecked {
        if (a == 0) return (true, 0);
        uint256 c = a * b;
        if (c / a != b) return (false, 0);
        return (true, c);
    }
}
function tryDiv(uint256 a, uint256 b) internal pure returns (bool, uint256) {
    unchecked {
        if (b == 0) return (false, 0);
        return (true, a / b);
    }
}
function tryMod(uint256 a, uint256 b) internal pure returns (bool, uint256) {
    unchecked {
        if (b == 0) return (false, 0);
        return (true, a % b);
    }
}
```

WEBSITE

Website	https://marveltech.crd.co/
Domain Registry	https://www.markmonitor.com
Domain Expiry Date	2025-07-19
Response Code	200
SSL Checker and HTTPS Test	Passed
Deprecated HTML tags	Passed
Robots.txt	Passed
Sitemap Test	Passed
SEO Friendly URL	Passed
Responsive Test	Passed
JS Error Test	Passed
Console Errors Test	Passed
Site Loading Speed Test	1.1 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.

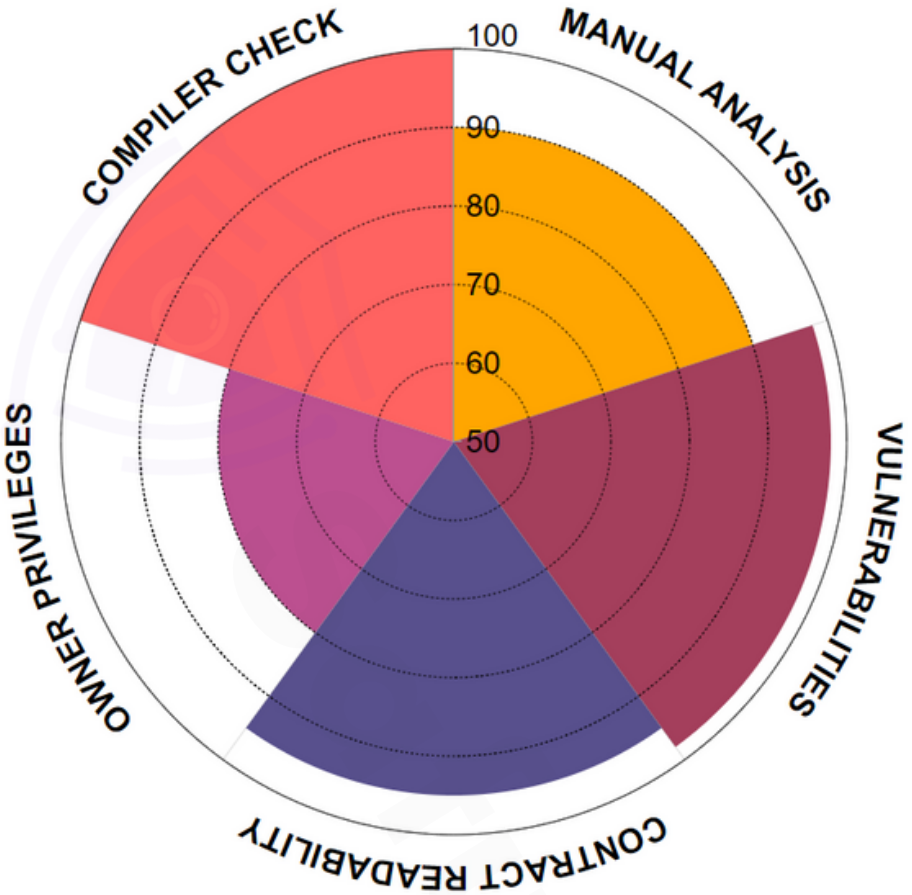
Accuracy of Information

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

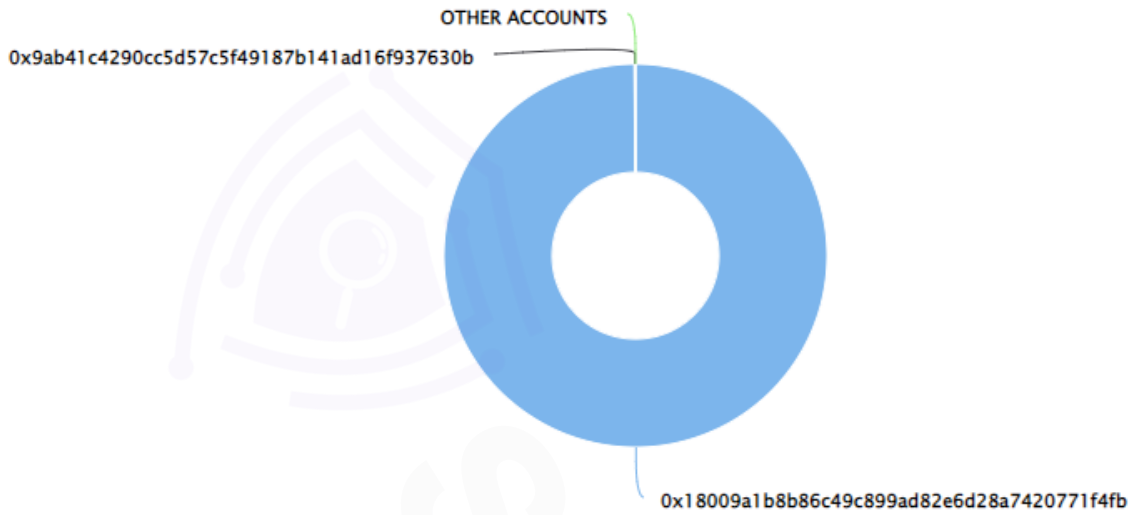
Owner Privileges

Compiler Check

Final Score: **92.6**

SUMMARY

Top 10 holders



Rank	Address	Quantity (Token)
1	0x18009a1b8b86c49c899ad82e6d28a7420771f4fb	999,180,000
2	0x9ab41c4290cc5d57c5f49187b141ad16f937630b	820,000

CONCLUSION

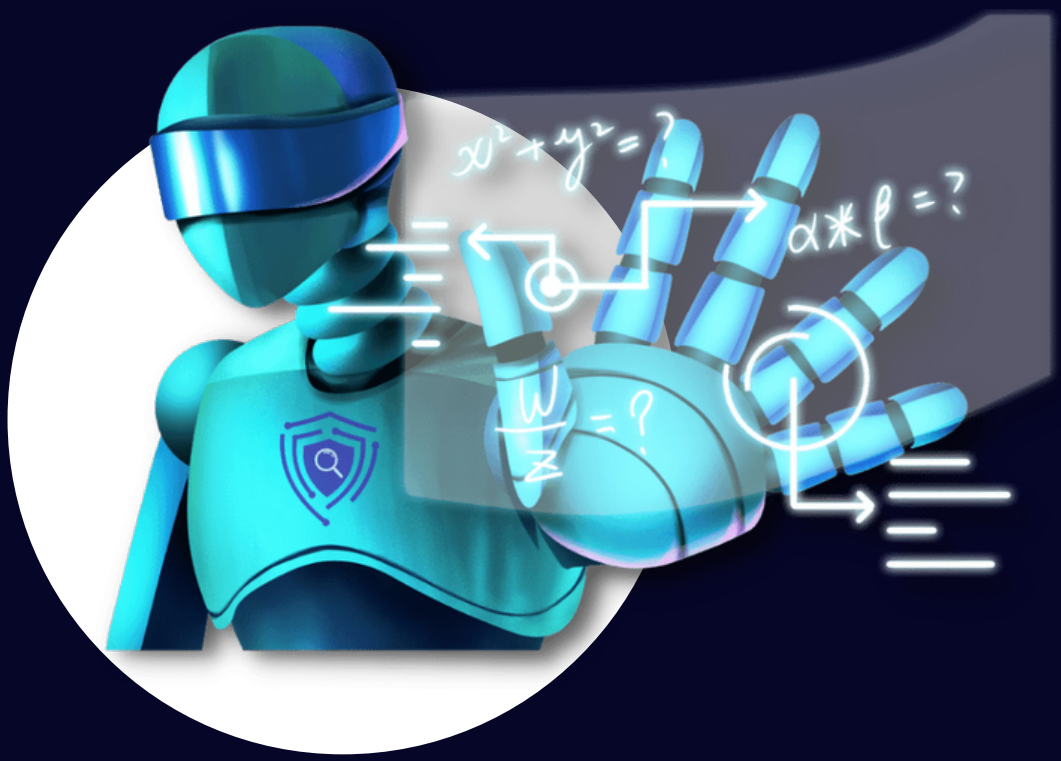
Project HULK does not contain any severe issues. Centralization risk is medium: owner can set fees up to 100%.

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