SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Ripmex

AUDIT

SECURITY ASSESSMENT

02. March, 2024

FOR

ripmex

Introduction

SolidProof.io is a brand of the officially registered company MAKE Network GmbH, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams.
Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

Disclaimer

SolidProof.io reports are not, nor should be considered, an "endorsement"or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc'...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	Ripmex
Website	https://ripmex.com/
About the project	Ripmex; It aims to make it possible for banks, payment providers, digital asset exchanges and companies to send money and value globally, of any size, securely, with low transaction fees and almost free of charge, using advanced blockchain technology.
Chain	Binance smart chain
Language	Solidity
Codebase	https://bscscan.com/address/0xc49bda8D35C450D7CE70A7530C1C42F9cC4169ba#code
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	https://t.me/Ripmex
Twitter	https://twitter.com/RipmexOfficial
Facebook	https://www.facebook.com/RipmexOfficial
Instagram	N/A
GitHub	https://github.com/RipmexOfficial
Reddit	https://www.reddit.com/user/RipmexOfficial/
Medium	https://ripmex.medium.com/
Discord	N/A
YouTube	https://www.youtube.com/@RipmexOfficial
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Change Log
v1.0	02. March 2024	· Layout Project · Automated/ Manual-Security Testing · Summary

Note – The following audit report presents a comprehensive security analysis of the smart contract utilized in the project that includes outside manipulation of the contract's functions in a malicious way. This analysis did not include functional testing (or unit testing) of the contract/s logic. We cannot guarantee 100% logical correctness of the contract as we did not functionally test it. This includes internal calculations in the formulae used in the contract.

File Overview

The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
contracts/Ripmex.sol	37b51b559008694a121fb0be04bf93d9a598ede1

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) be an indication of a changed state or potential vulnerability that was not the subject of this scan.

Imported packages.

Used code from other Frameworks/Smart Contracts.

N/A

Note for Investors: We only audited contracts mentioned in the scope above. All contracts related to the project apart from that are not a part of the audit, and we cannot comment on its security and are not responsible for it in any way.

External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

State variables

State variables are variables that are stored on the blockchain as part of the contract'sstate. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be needed within visibility modifier, such as public, private or internal, which determines the access level of the variable.

Components

📝 Contracts	📚 Libraries	🔍 Interfaces	🎨 Abstract
2	2	5	1

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐 Public	💰 Payable
108	5

External	Internal	Private	Pure	View
76	96	26	19	48

StateVariables

Total	🌐 Public
32	10

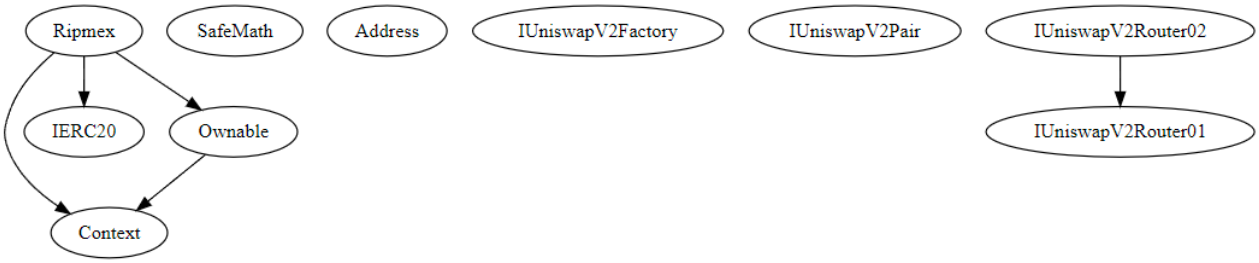
Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
^0.8.11	----------	Yes	yes (2 asm blocks)	----------

Transfers ETH	Low-Level Calls	Delegate Call	Uses Hash Functions	ECRecover	New/Create/Create2
yes					

Inheritance Graph

An inheritance graph is a graphical representation of the inheritance hierarchy among contracts. In object-oriented programming, inheritance is a mechanism that allows one class (or contract, in the case of Solidity) to inherit properties and methodsfrom another class. It shows the relationships between different contracts and how they are related to each other through inheritance.



Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit the vulnerability and the impact of that event on the organization or system.The risk level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executingthe contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers andsmart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   a. Reviewing the specifications, sources, and instructions provided to
   SolidProof to ensure we understand the size, scope, and functionality of the
   smart contract.
   b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
   c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.

2. Testing and automated analysis that includes the following:
   a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
   b. Symbolic execution, which is analysing a program to determine what inputs cause each part of a program to execute.

3. Review best practices, i.e., review smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.

4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security

Upgradeability

Contract is not an upgradable	✅ **Deployer cannot update the contract with new functionalities.**
Description	The contract is not an upgradeable contract. The Deployer is not able to change or add any functionalities to the contract after deploying.
Comment	N/A

Ownership

Contract ownership is not renounced.	❌ **The ownership is not renounced.**
Description	The owner has not renounced the ownership that means that the owner retains control over the contract's operations, including the ability to execute functions that may impact the contract's users or stakeholders. This can lead to several potential issues, including: • Centralizations • The owner has significant control over contract's operations.
Comment	N/A

Note – *The contract cannot be considered as renounced till it is not deployed or having some functionality that can change the state of the contract.*

Ownership Privileges

These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.

Minting tokens

Minting tokens refer to the process of creating new tokens in a cryptocurrency or blockchain network. This process is typically performed by the project's owner or designated authority, who has the ability to add new tokens to the network's total supply.

Contract owner can mint new tokens.	❌ **The owner can mint new tokens.**
Description	The owner is able to mint new tokens once the contract is deployed.
Comment	The owner can mint an unlimited amount of tokens, which is not recommended as this can manipulate the supply of tokens as this can manipulate the price of the tokens. it is recommended that there must be a fixed supply so that the tokens cannot be minted not more than that particular amount.

File/Line(s): L492-496

Codebase:

```
0 references | Control flow graph | a0712d68 | 3 references | ftrace | funcSig
function mint(uint256 amount) public onlyOwner
{
    1 reference
    uint256 rate =  _getRate();
    2 references
    uint256 rAmount =  amount.mul(rate);
    _rOwned[owner()] += rAmount;
    _tTotal += amount;
    _rTotal += rAmount;
    emit Transfer(address(0), _msgSender(), amount);
}
```

Burning tokens

Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.

Contract owner cannot burn tokens	✅ **The owner cannot burn tokens.**
Description	The owner is not able burn tokens without any allowances.
Comment	N/A

Blacklist addresses

Blacklisting addresses in smart contracts is the process of adding a certain address to a blacklist, effectively preventing them from accessing or participating in certain functionalities or transactions within the contract. This can be useful in preventing fraudulent or malicious activities, such as hacking attempts or money laundering.

Contract owner can blacklist addresses.	❌ **The owner can blacklist wallets.**
Description	The owner can blacklist wallets from transferring of tokens.
Comment	The owner can blacklist wallets from transferring tokens for an indefinite period of time as this is not recommended. There must be a locking period so that the user should not be locked permanently in the contract.

File/Line(s): L511-514

Codebase:

```
0 references | Control flow graph | 455a4396 | 1 reference | 1 reference | ftrace | funcSig
function blacklistAddress(address account, bool value) external onlyOwner
{
    _isBlacklisted[account] = value;
}
```

Fees and Tax

In some smart contracts, the owner or creator of the contract can set fees for certain actions or operations within the contract. These fees can be used to cover the cost of running the contract, such as paying for gas fees or compensating the contract's owner for their time and effort indeveloping and maintaining the contract.

Contract owner can set fees more than 25%.	✗ **The owner can set fees more than 25%.**
Description	The owner can set fees of more than 25%.
Comment	The owner can set any arbitrary amount in the tax, liquidity, and marketing fees in the contract, which is not recommended as this can cause the loss of funds for the users. It is recommended that the fees should not be more than 25% in the contract.

File/Line(s): L890-903

Codebase:

```
0 references | Control flow graph | 061c82d0 | 2 references | ftrace | funcSig
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
    _previousTaxFee = taxFee;
}

0 references | Control flow graph | 8ee88c53 | 2 references | ftrace | funcSig
function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() {
    _liquidityFee = liquidityFee;
    _previousLiquidityFee = liquidityFee;
}

0 references | Control flow graph | 457c194c | 2 references | ftrace | funcSig
function setMarketingFeePercent(uint256 marketingFee) external onlyOwner() {
    _marketingFee = marketingFee;
    _previousMarketingFee = marketingFee;
}
```

Lock User Funds

In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When token or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.

Contract owner can lock function.	❌ **The owner can lock function.**
Description	The owner can lock the contract.
Comment	The owner can update any arbitrary amount in the max transaction and max wallet amount including zero as this can cause the lock of the transfer function in the contract. There must be a minimum threshold limit so that the transfer function should not be locked for an indefinite period of time.

File/Line(s): L906-908, L1002-1005

Codebase:

```
0 references | Control flow graph | ec28438a | 1 reference | ftrace | funcSig
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner() {
    _maxTxAmount = maxTxAmount;
}
```

```
0 references | Control flow graph | 44d4225f | 1 reference | ftrace | funcSig
function  setWalletMaxHoldingLimit(uint256 _amount) public onlyOwner
{
        _walletHoldingMaxLimit = _amount;
}
```

Centralization Privileges

Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example,if the contract is controlled by a single entity or if certain participants have special permissions or abilities that others do not.

In the project, there are authorities that have access to the following functions:

File	Privileges
Ripmex.sol	➢ The owner can lock and unlock the ownership. ➢ The owner can withdraw ETH from the contract. ➢ The owner can mint an unlimited number of tokens. ➢ The owner can burn his tokens. ➢ The owner can blacklist wallets from transferring tokens. ➢ The owner can exclude/include wallets in fees. ➢ The owner can set the tax fees, liquidity fees, and marketing fees more than 100%. ➢ The owner can update any arbitrary amount in the max transaction amount. ➢ The owner can update any arbitrary amount in the minimum swap amount. ➢ The owner can update any arbitrary address as the marketing wallet address. ➢ The owner can enable/disable swapping. ➢ The owner can set the fees and maximum transaction amount for presale. ➢ The owner can exclude wallets from the max wallet limit. ➢ The owner can set any arbitrary amount in the max wallet limit.

Recommendations

To avoid potential hacking risks, it is advisable for the client to manage the private key of the privileged account with care. Additionally, we recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart-contract-based accounts, such as multi-signature wallets.

Here are some suggestions of what the client can do:

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security e.g. Gnosis Safe

- Use of a timelock at least with a latency of e.g. 48-72 hours for awareness of privileged operations
- Introduce a DAO/Governance/Voting module to increase transparency and user involvement
- Consider Renouncing the ownership so that the owner cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.

Audit Result

Critical Issues

No critical issues

High Issues

No high issues

Medium Issue

#1 | The owner can mint tokens.

File	Severity	Location	Status
Ripmex.sol	Medium	L492-496	Open

Description – The owner can mint an unlimited number of tokens, which is not recommended as this can manipulate the supply of tokens as this can manipulate the price of the tokens.
Remediation – It is recommended that there must be a fixed supply so that the tokens cannot be minted more than that particular amount.

#2 | The owner can blacklist wallets.

File	Severity	Location	Status
Ripmex.sol	Medium	L511-514	Open

Description – The owner can blacklist wallets from transferring tokens for an indefinite period of time as this is not recommended. There must be a locking period so that the user should not be locked permanently in the contract.

#3 | Liquidity is added to externally owned address.

File	Severity	Location	Status
Ripmex.sol	Medium	L717-730	Open

Description – The contract's liquidity is automatically added to the 'owner' address, which is not recommended because, in an extreme scenario, this can be used to drain liquidity from the contract.

#4 | The owner can set fees more than 100%.

File	Severity	Location	Status
Ripmex.sol	Medium	L717-730	Open

Description – The owner can set any arbitrary amount in the tax, liquidity, and marketing fees in the contract, which is not recommended as this can cause the loss of funds for the users. It is recommended that the fees should not be more than 25% of the contract.

Remediation – Add a 'require' check so that the total fees cannot be set to more than 25% in the contract.

#5 | The owner can lock tokens.

File	Severity	Location	Status
Ripmex.sol	Medium	L906-908, L1002-1005	Open

Description – The owner can update any arbitrary amount in the max transaction and max wallet amount including zero as this can cause the lock of the transfer function in the contract.
Remediation – There must be a minimum threshold limit so that the transfer function should not be locked for an indefinite period of time.

Low Issue

#1 | Missing events arithmetic.

File	Severity	Location	Status
Ripmex.sol	Low	L890-893, L895-899, L900-903, L906-910, L911-915, L1002-1005	Open

Description – Emit all the critical parameter changes.

#2 | Missing zero or dead address check.

File	Severity	Location	Status
Ripmex.sol	Low	L916	Open

Description – It is recommended to check that the address cannot be set to zero or dead.

#3 | Local variable shadowing.

File	Severity	Location	Status
Ripmex.sol	Low	L552, L630	Open

Description – Rename the local variables that shadow another component.

#4 | Remove Safemath library.

File	Severity	Location	Status
Ripmex.sol	Low	L552, L630	Open

Description – The compiler version above 0.8.0 has the ability to control arithmetic overflow/underflow. It is recommended to remove the unwanted code in order to avoid high gas fees.

Informational Issue

#1 | NatSpec Documentation missing.

File	Severity	Location	Status
Dogelegion.sol	Informational	--	Open

Description – If you started to comment on your code, also comment on all other functions, variables, etc.

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY