



# SAFU AUDIT

SMART CONTRACT AUDITING

# TREASURY CASINO

## SMART CONTRACT AUDIT



March 13, 2022

Update: March 16, 2022

# INTRODUCTION

---

<b>Client</b>	Treasury Casino (CCHIPS)
<b>Language</b>	Solidity
<b>Contract address</b>	0x6DCf2718c61f821183Be76334a867401728b1097
<b>Decimals</b>	9
<b>Supply</b>	250,000,000
<b>Platform</b>	Binance Smart Chain
<b>Compiler</b>	v0.8.9+commit.e5eed63a
<b>Optimization</b>	Yes, with 200 runs
<b>Website</b>	<a href="https://treasury-casino.online/">https://treasury-casino.online/</a>
<b>Telegram</b>	<a href="https://t.me/treasurycasino">https://t.me/treasurycasino</a>
<b>Twitter</b>	<a href="https://twitter.com/fantasygirlnft">https://twitter.com/fantasygirlnft</a>

## Description

The treasury casino is a crypto casino. The casino chips CCHIPS will give you access to the Casino's staking vault which will have an opening staking pool.

# TABLE OF CONTENTS

## 01 INTRODUCTION

---

Introduction	02
Approach	04
Risk classification	05

## 02 ABSTRACT

---

Abstract	06
----------	----

## 03 VULNERABILITIES TEST

---

Vulnerabilities Test	07
----------------------	----

## 04 MANUAL ANALYSIS

---

Manual analysis	09
Contract Inspection	10
Inheritance Tree	14
Important Snippets	15
Good Practices	16

## 05 WEBSITE

---

Website Audit	17
---------------	----

## 06 CONCLUSIONS

---

Disclaimer	18
Audit Results	19
SafuScore	20
Summary	22

# 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
- MythX, Mytrhl
- SWC Registry
- Open Zeppelin Code Analyzer
- Solidity Code Complier

# 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 efficacy or security for features with a risk free factor.

# ABSTRACT

---

Fees	Ownership	Mint function
Buy Fees: 5% Sell Fees: 15%	Owned Update: Ownership renounced as of 3-16-2022	Yes Update: Mint stopped on 3-16-2022
Owner can set fees	Max Tx amount	Pause
Owner can set fees up to 10% (buy) / 15% (sell) Update: Ownership renounced	Owner can set max Tx amount Update: Ownership renounced	Owner can't pause trading

# Vulnerabilities Test

SWC ID	Description	
<b>SWC-100</b>	Function Default Visibility	<b>Passed</b>
<b>SWC-101</b>	Integer Overflow and Underflow	<b>Passed</b>
<b>SWC-102</b>	Outdated Compiler Version	<b>Passed</b>
<b>SWC-103</b>	FloatingPragma	<b>Passed</b>
<b>SWC-104</b>	Unchecked Call Return Value	<b>Passed</b>
<b>SWC-105</b>	Unprotected Ether Withdrawal	<b>Passed</b>
<b>SWC-106</b>	Unprotected SELF-DESTRUCT Instruction	<b>Passed</b>
<b>SWC-107</b>	Re-entrancy	<b>Passed</b>
<b>SWC-108</b>	State Variable Default Visibility	<b>Minor</b>
<b>SWC-109</b>	Uninitialized Storage Pointer	<b>Passed</b>
<b>SWC-110</b>	Assert Violation	<b>Passed</b>
<b>SWC-111</b>	Use of Deprecated Solidity Functions	<b>Passed</b>
<b>SWC-112</b>	Delegate Call to Untrusted Callee	<b>Passed</b>
<b>SWC-113</b>	DoS with Failed Call	<b>Passed</b>
<b>SWC-114</b>	Transaction Order Dependence	<b>Passed</b>
<b>SWC-115</b>	Authorization through tx.origin	<b>Passed</b>

<b>SWC-116</b>	Block values as a proxy for time	<b>Passed</b>
<b>SWC-117</b>	Signature Malleability	<b>Passed</b>
<b>SWC-118</b>	Incorrect Constructor Name	<b>Passed</b>
<b>SWC-119</b>	Shadowing State Variables	<b>Passed</b>
<b>SWC-120</b>	Weak Sources of Randomness from Chain Attributes	<b>Passed</b>
<b>SWC-121</b>	Missing Protection against Signature Replay Attacks	<b>Passed</b>
<b>SWC-122</b>	Lack of Proper Signature Verification	<b>Passed</b>
<b>SWC-123</b>	Requirement Violation	<b>Passed</b>
<b>SWC-124</b>	Write to Arbitrary Storage Location	<b>Passed</b>
<b>SWC-125</b>	Incorrect Inheritance Order	<b>Passed</b>
<b>SWC-126</b>	Insufficient Gas Griefing	<b>Passed</b>
<b>SWC-127</b>	Arbitrary Jump with Function Type Variable	<b>Passed</b>
<b>SWC-128</b>	DoS With Block Gas Limit	<b>Passed</b>
<b>SWC-129</b>	Typographical Error	<b>Passed</b>
<b>SWC-130</b>	Right-To-Left-Override control character (U+202E)	<b>Passed</b>
<b>SWC-131</b>	Presence of unused variables	<b>Passed</b>
<b>SWC-132</b>	Unexpected Ether balance	<b>Passed</b>
<b>SWC-133</b>	Hash Collisions With Multiple Variable Length Arguments	<b>Passed</b>
<b>SWC-134</b>	Message call with the hardcoded gas amount	<b>Passed</b>
<b>SWC-135</b>	Code With No Effects (Irrelevant/Dead Code)	<b>Passed</b>
<b>SWC-136</b>	Unencrypted Private Data On-Chain	<b>Passed</b>

# MANUAL ANALYSIS

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

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

# CONTRACT INSPECTION



```
| **Context** | Implementation | ||| |
| L | _msgSender | Internal 🔒 | |||
| L | _msgData | Internal 🔒 | |||
|||||
| **Ownable** | Implementation | Context |||
| L | <Constructor> | Public ! | ⚡ | NO! |
| L | owner | Public ! | | NO! |
| L | renounceOwnership | Public ! | ⚡ | onlyOwner |
| L | transferOwnership | Public ! | ⚡ | onlyOwner |
|||||
| **ERC20** | Interface | |||
| L | totalSupply | External ! | | NO! |
| L | balanceOf | External ! | | NO! |
| L | transfer | External ! | ⚡ | NO! |
| L | allowance | External ! | | NO! |
| L | approve | External ! | ⚡ | NO! |
| L | transferFrom | External ! | ⚡ | NO! |
|||||
| **SafeMath** | Library | |||
| L | add | Internal 🔒 | |||
| L | sub | Internal 🔒 | |||
| L | sub | Internal 🔒 | |||
| L | mul | Internal 🔒 | |||
| L | div | Internal 🔒 | |||
| L | div | Internal 🔒 | |||
| L | mod | Internal 🔒 | |||
| L | mod | Internal 🔒 | |||
|||||
| **ERC20** | Implementation | Context, IERC20 |||
| L | <Constructor> | Public ! | ⚡ | NO! |
| L | name | Public ! | | NO! |
| L | symbol | Public ! | | NO! |
| L | decimals | Public ! | | NO! |
| L | totalSupply | Public ! | | NO! |
| L | balanceOf | Public ! | | NO! |
```

| L | transfer | Public | |  | NO | |

| L | allowance | Public | |  | NO | |

| L | approve | Public | |  | NO | |

| L | transferFrom | Public | |  | NO | |

| L | increaseAllowance | Public | |  | NO | |

| L | decreaseAllowance | Public | |  | NO | |

| L | \_transfer | Internal  | |  | |

| L | \_mint | Internal  | |  | |

| L | \_burn | Internal  | |  | |

| L | \_approve | Internal  | |  | |

| L | \_setupDecimals | Internal  | |  | |

| L | \_beforeTokenTransfer | Internal  | |  | |

|||||

| \*\*ERC20Capped\*\* | Implementation | ERC20 | |

| L | <Constructor> | Public | |  | NO | |

| L | cap | Public | |  | NO | |

| L | \_beforeTokenTransfer | Internal  | |  | |

|||||

| \*\*ERC20Burnable\*\* | Implementation | Context, ERC20 | |

| L | burn | Public | |  | NO | |

| L | burnFrom | Public | |  | NO | |

|||||

| \*\*ERC20Mintable\*\* | Implementation | ERC20 | |

| L | mintingFinished | Public | |  | NO | |

| L | mint | Public | |  | canMint |

| L | finishMinting | Public | |  | canMint |

| L | \_finishMinting | Internal  | |  | |

|||||

| \*\*IUniswapV2Router01\*\* | Interface | |

| L | factory | External | |  | NO | |

| L | WETH | External | |  | NO | |

| L | addLiquidity | External | |  | NO | |

| L | addLiquidityETH | External | |  | NO | |

| L | removeLiquidity | External | |  | NO | |

| L | removeLiquidityETH | External | |  | NO | |

| L | removeLiquidityWithPermit | External | |  | NO | |

| L | removeLiquidityETHWithPermit | External | |  | NO | |

L	swapExactTokensForTokens	External !		NO
L	swapTokensForExactTokens	External !		NO
L	swapExactETHForTokens	External !		NO
L	swapTokensForExactETH	External !		NO
L	swapExactTokensForETH	External !		NO
L	swapETHForExactTokens	External !		NO
L	quote	External !		NO
L	getAmountOut	External !		NO
L	getAmountIn	External !		NO
L	getAmountsOut	External !		NO
L	getAmountsIn	External !		NO

|||||

\*\*IUniswapV2Router02\*\*	Interface	IUniswapV2Router01		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External !		NO
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External !		NO
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External !		NO
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External !		NO
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External !		NO

|||||

\*\*IUniswapV2Factory\*\*	Interface			
L	feeTo	External !		NO
L	feeToSetter	External !		NO
L	getPair	External !		NO
L	allPairs	External !		NO
L	allPairsLength	External !		NO
L	createPair	External !		NO
L	setFeeTo	External !		NO
L	setFeeToSetter	External !		NO

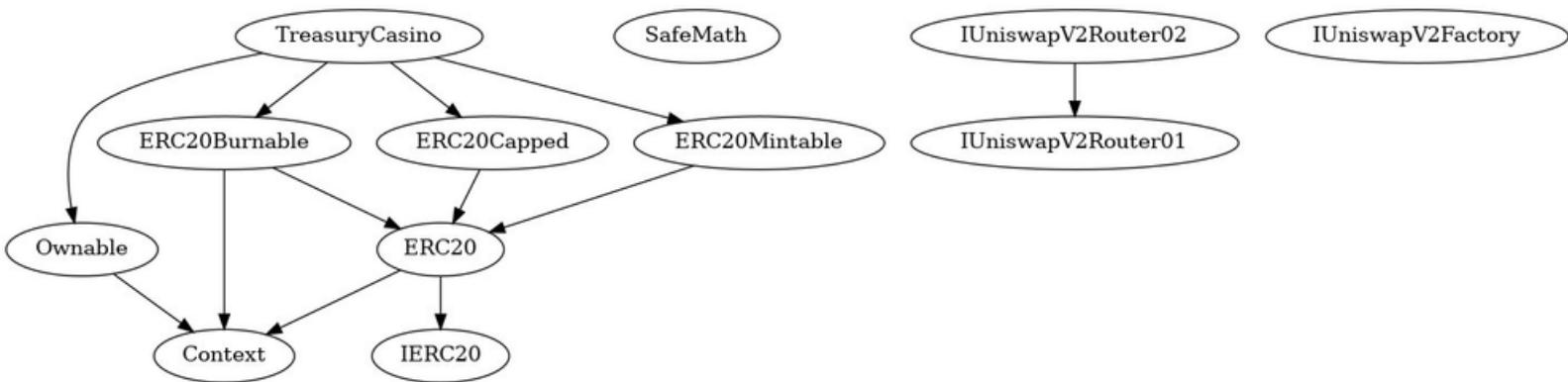
|||||

\*\*TreasuryCasino\*\*	Implementation	ERC20Capped, ERC20Mintable, ERC20Burnable, Ownable		
L	<Constructor>	Public !		ERC20 ERC20Capped
L	<Receive Ether>	External !		NO
L	excludeFromFees	Public !		onlyOwner
L	setAutomatedMarketMakerPair	Public !		onlyOwner
L	\_setAutomatedMarketMakerPair	Private 		
L	isExcludedFromFees	Public !		NO

```
| └ | _transfer | Internal 🔒 | 🔴 ||  
| └ | swapTokensForBUSD | Private 💰 | 🔴 | lockTheSwap |  
| └ | setSwapTokensAtAmount | External 🚧 | 🔴 | onlyOwner |  
| └ | setBuyTax | External 🚧 | 🔴 | onlyOwner |  
| └ | setSellTax | External 🚧 | 🔴 | onlyOwner |  
| └ | setMaxTxAmount | External 🚧 | 🔴 | onlyOwner |  
| └ | setSwapEnabled | Public 🚧 | 🔴 | onlyOwner |  
| └ | setMarketingWallet | External 🚧 | 🔴 | onlyOwner |  
| └ | _mint | Internal 🔒 | 🔴 | onlyOwner |  
| └ | _finishMinting | Internal 🔒 | 🔴 | onlyOwner |  
| └ | _beforeTokenTransfer | Internal 🔒 | 🔴 ||
```

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.

# Important Snippets



## Owner can mint new tokens and also stop the minting (Update: Ownership renounced)

```
function _mint(address account, uint256 amount) internal override onlyOwner {
    super._mint(account, amount);
}

function _finishMinting() internal override onlyOwner {
    super._finishMinting();
}
```

## Owner can set fees with a limit (Update: Ownership renounced)

```
function setBuyTax(uint256 _marketingBuyFee) external onlyOwner() {
    require(_marketingBuyFee <= 10, "buy tax must be less than or equal to 10");
    marketingBuyFee = _marketingBuyFee;
}

function setSellTax(uint256 _marketingSellFee) external onlyOwner() {
    require(_marketingSellFee <= 15, "buy tax must be less than or equal to 15");
    marketingSellFee = _marketingSellFee;
}
```

## Owner can set max tx but not to 0 (Update: Ownership renounced)

```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner() {
    require(maxTxAmount > totalSupply().div(1000), "transaction amount must be greater than 0.1% of total supply");
    _maxTxAmount = maxTxAmount * (10**9);
}
```

## Owner can exclude accounts from fees (Update: Ownership renounced)

```
function excludeFromFees(address account, bool excluded) public onlyOwner {
    require(_isExcludedFromFees[account] != excluded, "CCHIPs: Account is already the value of 'excluded'");
    _isExcludedFromFees[account] = excluded;

    emit ExcludeFromFees(account, excluded);
}
```

## Owner can change swap settings (Update: Ownership renounced)

```
function setSwapEnabled(bool _enabled) public onlyOwner {
    swapEnabled = _enabled;
    emit SwapEnabledUpdated(_enabled);
}
```

# GOOD PRACTICES ✓

---

- **Ownership renounced as of 3-16-2022**
- The owner cannot stop or pause the smart contract
- The owner cannot set fees over a certain percentage
- The owner cannot set max Tx to 0
- The smart contract utilizes "SafeMath" to prevent overflows

```
library SafeMath {
    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 {
            // Gas optimization: this is cheaper than requiring 'a' not being zero, but
            // benefit is lost if 'b' is also tested.
            // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
            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



<b>Website</b>	<a href="https://treasury-casino.online">https://treasury-casino.online</a>
<b>Domain Registry</b>	<a href="http://www.wordpress.com">http://www.wordpress.com</a>
<b>Domain Expiry Date</b>	2023-02-11
<b>Response Code</b>	200
<b>SSL Checker and HTTPS Test</b>	Passed
<b>Deprecated HTML tags</b>	Passed
<b>Robots.txt</b>	Passed
<b>Sitemap Test</b>	Passed
<b>SEO Friendly URL</b>	Low
<b>Responsive Test</b>	Passed
<b>JS Error Test</b>	Passed
<b>Console Errors Test</b>	Low
<b>Site Loading Speed Test</b>	2.38 seconds - Passed
<b>HTTP2 Test</b>	Passed
<b>Safe Browsing Test</b>	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 accuracy of 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 analyse the on-chain smart contract source code, and to provide 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 to smart contract's 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.

# AUDIT RESULTS

---

## UPDATE 3-16-2022

---

Project Owner requested an update on this audit.

- As of 3-16-2022, Ownership has been renounced and minting stopped.
- Every owner privileges stated above are now disabled.
- SafuScore is now increased.

## CRITICAL

---

- Owner can mint new tokens - resulting in tokens being highly inflated. However, calling the `_finishMinting()` function will stop the minting. Also, renouncing the ownership can prevent minting functions ever being called again.

## MEDIUM

---

No medium severity issues have been found.

## MINOR

---

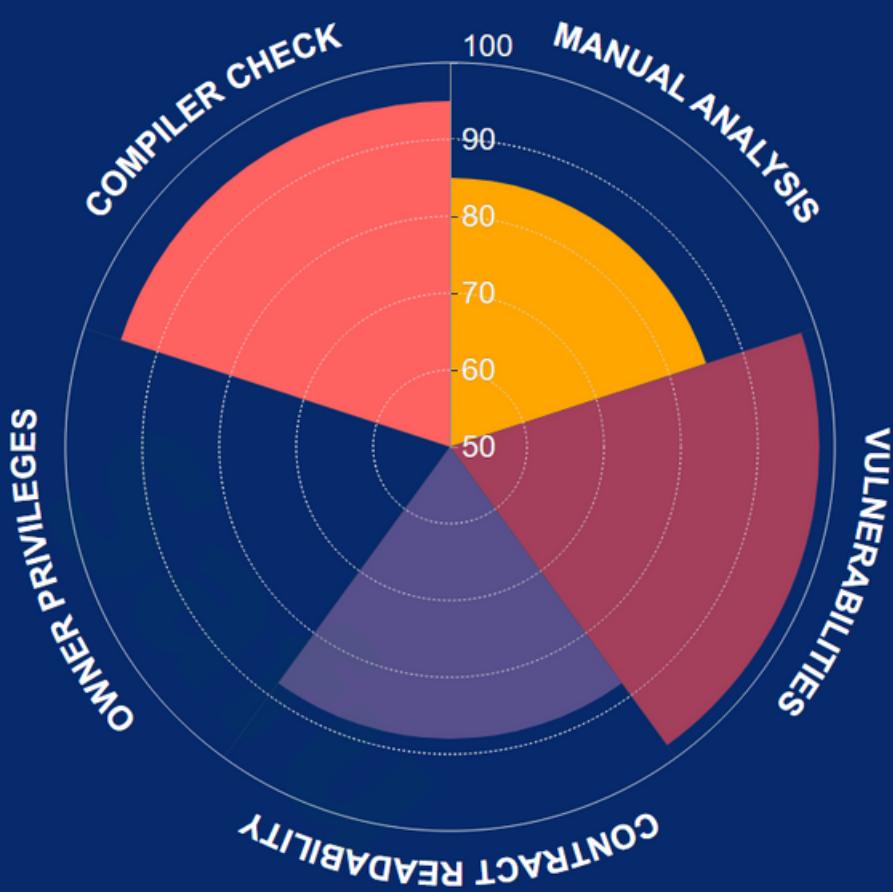
- State variable visibility is not set. It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwap" is internal.

## INFORMATIONAL

---

The standard audit model does not offer suggestions and consulting for improvements of efficacy.

# SAFUSCORE



Manual Analysis



Vulnerabilities



Contract Readability



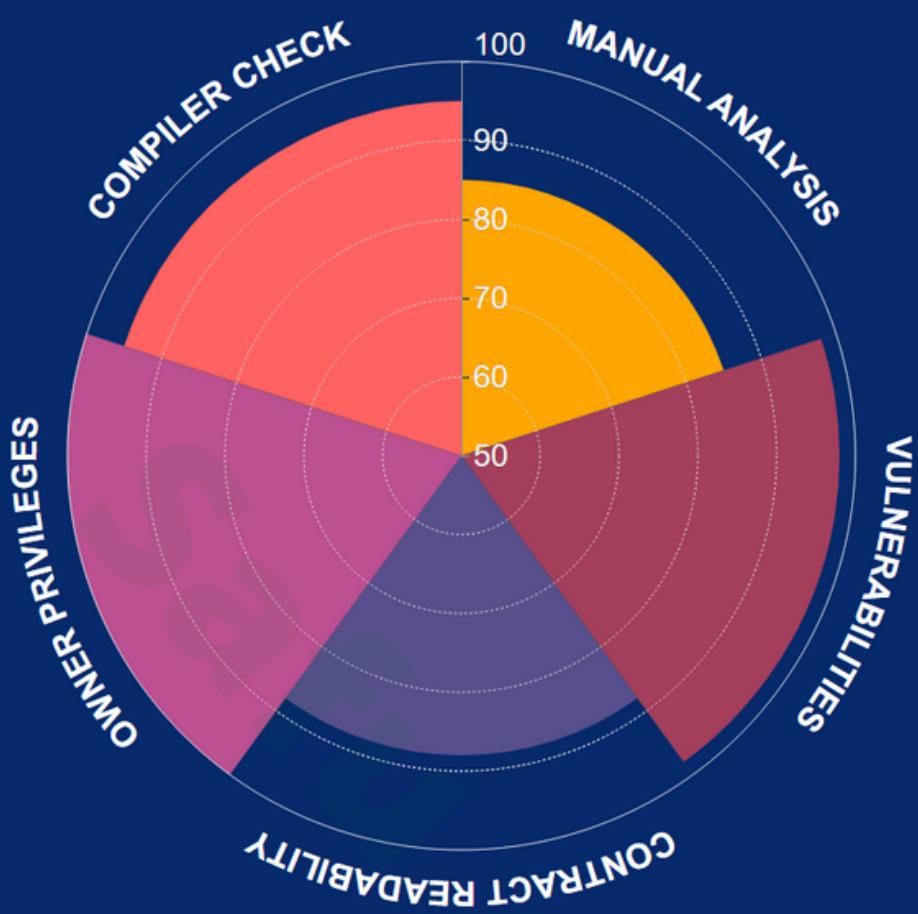
Owner Privileges



Compiler Check

**Final Score: 83.2**

# SAFUSCORE - AFTER UPDATE



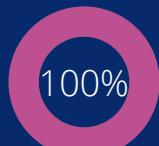
Manual Analysis



Vulnerabilities



Contract Readability



Owner Privileges

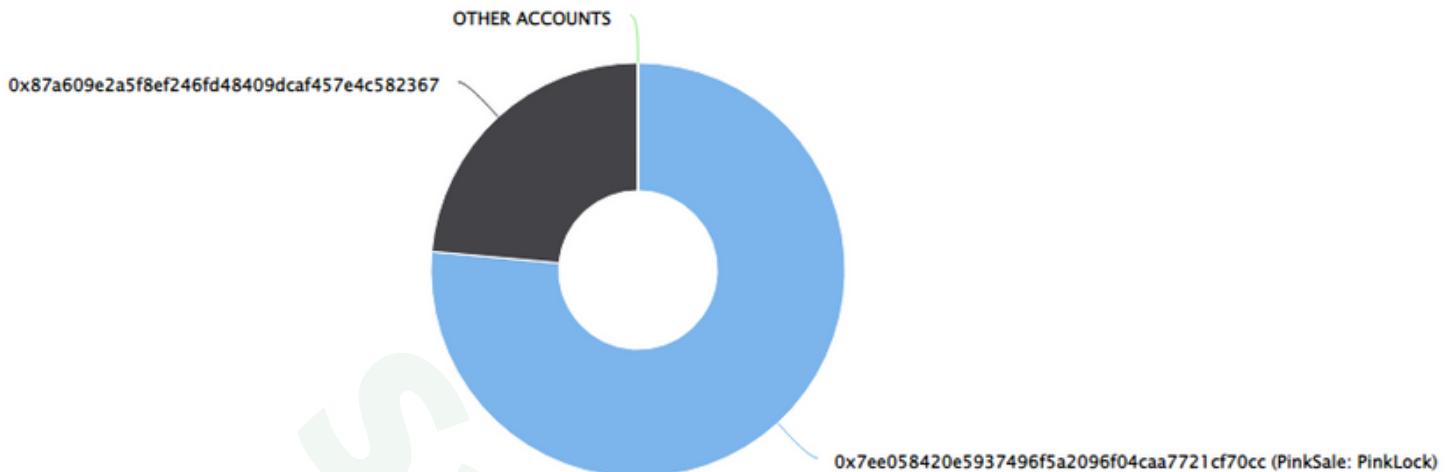


Compiler Check

**Final Score: 93.2**

# SUMMARY

## Top 10 holders



Rank	Address	Quantity (Token)	Percentage
1	<a href="#">PinkSale: PinkLock</a>	191,069,800	76.4279%
2	<a href="#">0x87a609e2a5f8ef246fd48409dcaf457e4c582367</a>	58,930,200	23.5721%

## Conclusion

Project Treasury Casino contains a risk characteristic. If the mint functionality is abused, tokens would be highly inflated.

Update: Mint has been stopped and ownership renounced. Owner privileges are now disabled.

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





**SafuAudit.com**

