CFG NINJA

SECURITY ASSESSMENT
TROY TOKEN

January 10, 2024

Audit Status: Fail

BLADE POOL

RISK ANALYSIS | TROY.

Classifications of Manual Risk Results

Classification	Description
🔴 Critical	Danger or Potential Problems.
🔴 High	Be Careful or Fail test.
🟠 Medium	Improve is needed.
🟢 Low	Pass, Not-Detected or Safe Item.
ℹ Informational	Function Detected

Manual Code Review Risk Results

Contract Security	Description
🟢 Buy Tax	14%
🟢 Sale Tax	14%
🟢 Cannot Buy	Pass
🟢 Cannot Sale	Pass
🟢 Max Tax	14%
ℹ Modify Tax	Yes
🟢 Fee Check	Pass
🟢 Is Honeypot?	Not Detected
🟢 Trading Cooldown	Not Detected
🔴 Enable Trade?	False
🔴 Pause Transfer?	Detected

Contract Security	Description
🟢 Max Tx?	Pass
🔴 Is Anti Whale?	Detected
🟢 Is Anti Bot?	Not Detected
🔴 Is Blacklist?	Detected
🔴 Blacklist Check	Fail
🟢 is Whitelist?	Detected
🟢 Can Mint?	Pass
🟢 Is Proxy?	Not Detected
🟢 Can Take Ownership?	Not Detected
🟢 Hidden Owner?	Not Detected
ℹ Owner	0xa9aD84C69ca5cc34b37E93E63477A508a1bFD00c
🟢 Self Destruct?	Not Detected
🟢 External Call?	Detected
🟢 Other?	Not Detected
🟢 Holders	2
🔴 Audit Confidence	Low
🟢 Authority Check	Pass
🟢 Freeze Check	Pass

The summary section reveals the strengths and weaknesses identified during the assessment, including any vulnerabilities or potential risks that may exist. It serves as a valuable snapshot of the overall security status of the audited project. However, it is highly recommended to read the entire security assessment report for a comprehensive understanding of the findings. The full report provides detailed insights into the assessment process, methodology, and specific recommendations for addressing the identified issues.

CFG Ninja Verified on January 10, 2024

TROY

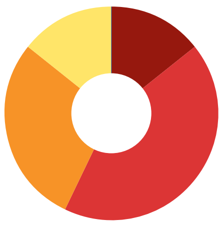
Executive Summary

TYPES	ECOSYSTEM	LANGUAGE
DeFi	BNBCHAIN	Solidity

Timeline

Audit Request	Onboarding Process	Audit Preview	Audit Release
2024-01-09	2024-01-10	2024-01-10	2024-01-10

Vulnerability Summary



5	0	5	5
Total Findings	Resolved	Pending	Unresolved

● 1 Critical	0 Resolved, 1 Pending	Critical risks are the most severe and can have a significant impact on the smart contracts functionality, security, or the entire system. These vulnerabilities can lead to the loss of user funds, unauthorized access, or complete system compromise.
● 3 High	0 Resolved, 3 Pending	High-risk vulnerabilities have the potential to cause significant harm to the smart contract or the system. While not as severe as critical risks, they can still result in financial losses, data breaches, or denial of service attacks.
● 2 Medium	0 Resolved, 2 Pending	Medium-risk vulnerabilities pose a moderate level of risk to the smart contracts security and functionality. They may not have an immediate and severe impact but can still lead to potential issues if exploited. These risks should be addressed to ensure the contracts overall security.
● 1 Low	0 Resolved, 1 Pending	Low-risk vulnerabilities have a minimal impact on the smart contracts security and functionality. They may not pose a significant threat, but it is still advisable to address them to maintain a robust security posture.
● 0 Informational		Informational risks are not actual vulnerabilities but provide useful information about potential improvements or best practices. These findings may include suggestions for code optimizations, documentation enhancements, or other non-critical areas for improvement.

PROJECT OVERVIEW | TROY.

Token Summary

Parameter	Result
Address	0x9299e4Ca258A8C6623708828Ac3F87E515FD831A
Name	TROY
Token Tracker	TROY (TROY)
Decimals	18
Supply	100,000,000,000
Platform	BNBCHAIN
Compiler	v0.8.10+commit.fc410830
Contract Name	TROY
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/token/0x9299e4Ca258A8C6623708828Ac3F87E515FD831A#code

Main Contract Assessed

Name	Contract	Live
TROY	0x9299e4Ca258A8C6623708828Ac3F87E515FD831A	Yes

TestNet Contract Was Not Assessed

Solidity Code Provided

SolID	File Sha-1	FileName
TROY	573c1bf047f4f341f5ae9a0e8f5eb7e6c335beaa	TROY.sol

TECHNICAL FINDINGS | TROY.

Smart contract security audits classify risks into several categories: Critical, High, Medium, Low, and Informational. These classifications help assess the severity and potential impact of vulnerabilities found in smart contracts.

Classification of Risk

Severity	Description
● Critical	Critical risks are the most severe and can have a significant impact on the smart contracts functionality, security, or the entire system. These vulnerabilities can lead to the loss of user funds, unauthorized access, or complete system compromise.
● High	High-risk vulnerabilities have the potential to cause significant harm to the smart contract or the system. While not as severe as critical risks, they can still result in financial losses, data breaches, or denial of service attacks.
● Medium	Medium-risk vulnerabilities pose a moderate level of risk to the smart contracts security and functionality. They may not have an immediate and severe impact but can still lead to potential issues if exploited. These risks should be addressed to ensure the contracts overall security.
● Low	Low-risk vulnerabilities have a minimal impact on the smart contracts security and functionality. They may not pose a significant threat, but it is still advisable to address them to maintain a robust security posture.
ⓘ Informational	Informational risks are not actual vulnerabilities but provide useful information about potential improvements or best practices. These findings may include suggestions for code optimizations, documentation enhancements, or other non-critical areas for improvement.

By categorizing risks into these classifications, smart contract security audits can prioritize the resolution of critical and high-risk vulnerabilities to ensure the contract's overall security and protect user funds and data.

TROY-04 | Centralized Risk In addLiquidity.

Category	Severity	Location	Status
Coding Style	🔴 High	TROY.sol:	Detected

Description

uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this), tokenAmount, 0, 0, owner(), block.timestamp);

The addLiquidity function calls the uniswapV2Router.addLiquidityETH function with the to address specified as owner() for acquiring the generated LP tokens from the TROY-WBNB pool.
As a result, over time the _owner address will accumulate a significant portion of LP tokens.If the _owner is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

We advise the to address of the uniswapV2Router.addLiquidityETH function call to be replaced by the contract itself, i.e. address(this) , and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the _owner account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

1. Indicatively, here are some feasible solutions that would also mitigate the potential risk:
2. Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
3. Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;

Introduction of a DAO / governance / voting module to increase transparency and user involvement

Mitigation

References:

Centralization Risk in Crypto: How Decentralized Is Crypto?

TROY-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	● Low	TROY.sol: L: 397 C: 14, L: 405 C: 14, L: 594 C: 14, L: 1670 C: 14	Detected

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.The linked code does not create an event for the transfer.

Recommendation

Emit an event for critical parameter changes. It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Mitigation

References:

Understanding Events in Smart Contracts

TROY-11 | IterableMapping is not verified..

Category	Severity	Location	Status
Optimizatio n	🟠 High	TROY.sol: L: 0 C: 0	Detected

Description

We detected that iterableMapping was not verified in the contract.

Recommendation

Please verify the library

Mitigation

References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

TROY-12 | Centralization Risks In The Role or Function.

Category	Severity	Location	Status
Centralization / Privilege	● High	TROY.sol:	Detected

Description

In the contract , the role has authority over the following functions:
function burn(), to burn anyone's account at any amount.
function burnFrom(), to burn anyone's account at the number in the range of _allowed .
Any compromise to the account may allow the hacker to take advantage of this authority.
We understand the role could be assigned to the smart contract , however, the
is a map and more addresses could be added.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation
and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the
client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In
general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized
mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets.

Mitigation

References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

TROY-14 | Unnecessary Use Of SafeMath.

Category	Severity	Location	Status
Logical Issue	● Medium	TROY.sol: L: 0 C: 0	Detected

Description

The SafeMath library is used unnecessarily. With Solidity compiler versions 0.8.0 or newer, arithmetic operations
will automatically revert in case of integer overflow or underflow.
library SafeMath {
An implementation of SafeMath library is found.
using SafeMath for uint256;
SafeMath library is used for uint256 type in contract.

Recommendation

We advise removing the usage of SafeMath library and using the built-in arithmetic operations provided by the
Solidity programming language.

Mitigation

References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

TROY-18 | Stop Transactions by using Enable Trade.

Category	Severity	Location	Status
Logical Issue	🔴 Critical	TROY.sol: L: 393 C: 14	Detected

Description

Enable Trade is present on the following contract and when combined with Exclude from fees it can be considered a whitelist process, this will allow anyone to trade before others and can represent and issue for the holders.

Recommendation

We recommend the project owner to carefully review this function and avoid problems when performing both actions.

Mitigation

References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

TROY-19 | Centralization Privileges of TROY.

Category	Severity	Location	Status
Coding Style	● Medium	TROY.sol: L: 393 C: 14,L: 385 C: 14,L: 341 C: 14,L: 306 C: 14,L: 299 C: 14,L: 269 C: 14	Detected

Description

In a smart contract, the concept of "onlyOwner" functions refers to certain functions that can only be executed by the owner or creator of the contract. These functions are typically designed to perform critical actions or modify sensitive data within the contract. By restricting access to these functions, the contract owner maintains control and ensures the integrity and security of the contract.

Function Name	Parameters	Visibility
renounceOwnership		Public
transferOwnership	address newOwner	Public
updateDividendToken		Public
distributeDividends		Public
withdrawDividend		Public
excludeFromDividends		Public
updateSwapAmount		Public
updateDividendTracker		Public
updateDividendToken		Public
updateUniswapV2Router		Public

Function Name	Parameters	Visibility
excludeFromFees		Public
addToBlacklist		Public
excludeMultipleAccountsFromFees		Public
setAutomatedMarketMakerPair		Public
SetupLiquidityTokenAddress		Public
TeamlimitLiquidityReleaseTo20Percen t		Public
TeamUnlockLiquidityInSeconds		Public
TeamReleaseLiquidity		Public
updateLiquidityWallet		Public
updateMarketingWallet		Public
updateGasForProcessing		Public
updateClaimWait		External
updateMinimumTokenRequirement		External
tradingEnabled		Public
updateBiggestBuy		Public
burningEnabled		Public
addLP		External
letsGoLive		External
letsGetStarted		External
updateBuyFees		Public

Function Name	Parameters	Visibility
updateSellFees		Public
updateMaxWallet		Public
updateMaxBuySell		Public
withdrawRemainingToken		Public
withdrawRemainingBEP20Token		Public
burnRemainingToken		Public
excludeFromDividends		External
updateClaimWait		External
updateMinimumTokenRequirement		External
setBalance		External
processAccount		Public

Recommendation

Inheriting from Ownable and calling its constructor on yours ensures that the address deploying your contract is registered as the owner. The onlyOwner modifier makes a function revert if not called by the address registered as the owner. It is important that deployr or owner secure the credentials that has owner priviledge to ensure the security of the project.

Mitigation

References:

Guide to Ownership and Access Control in Solidity

Writing Clean Code for Solidity: Best Practices for Solidity Development

FINDINGS

In this document, we present the findings and results of the smart contract security audit. The identified vulnerabilities, weaknesses, and potential risks are outlined, along with recommendations for mitigating these issues. It is crucial for the team to address these findings promptly to enhance the security and trustworthiness of the smart contract code.

Severity	Found	Pending	Resolved
● Critical	1	1	0
● High	2	3	0
● Medium	3	2	0
● Low	1	1	0
● Informational	0	0	0
Total	7	5	0

In a smart contract, a technical finding summary refers to a compilation of identified issues or vulnerabilities discovered during a security audit. These findings can range from coding errors and logical flaws to potential security risks. It is crucial for the project owner to thoroughly review each identified item and take necessary actions to resolve them. By carefully examining the technical finding summary, the project owner can gain insights into the weaknesses or potential threats present in the smart contract. They should prioritize addressing these issues promptly to mitigate any risks associated with the contract's security. Neglecting to address any identified item in the security audit can expose the smart contract to significant risks. Unresolved vulnerabilities can be exploited by malicious actors, potentially leading to financial losses, data breaches, or other detrimental consequences. To ensure the integrity and security of the smart contract, the project owner should engage in a comprehensive review process. This involves understanding the nature and severity of each identified item, consulting with experts if needed, and implementing appropriate fixes or enhancements. Regularly updating and maintaining the smart contract's codebase is also essential to address any emerging security concerns. By diligently reviewing and resolving all identified items in the technical finding summary, the project owner can significantly reduce the risks associated with the smart contract and enhance its overall security posture.

SOCIAL MEDIA CHECKS | TROY.

Social Media	URL	Result
Website	https://troy.com.co	Pass
Telegram	https://t.me/TROYECOMATIC	Pass
Twitter	@TROYECOMATIC	Pass
Facebook		N/A
Reddit	N/A	N/A
Instagram		N/A
CoinGecko	N/A	N/A
Github		N/A
CMC	N/A	N/A
Email	info@etfbtc.site	Contact
Other		N/A

From a security assessment standpoint, inspecting a project's social media presence is essential. It enables the evaluation of the project's reputation, credibility, and trustworthiness within the community. By analyzing the content shared, engagement levels, and the response to any security-related incidents, one can assess the project's commitment to security practices and its ability to handle potential threats.

Social Media Information Notes:

Auditor Notes:

Project Owner Notes:

ASSESSMENT RESULTS | TROY.

Score Rsesults

Review	Score
Overall Score	71/100
Auditor Score	60/100

Review by Section	Score
Manual Scan Score	39
SWC Scan Score	37
Advance Check Score	-5

Our security assessment or audit score system for the smart contract and project follows a comprehensive evaluation process to ensure the highest level of security. The system assigns a score based on various security parameters and benchmarks, with a passing score set at 80 out of a total attainable score of 100.The assessment process includes a thorough review of the smart contracts codebase, architecture, and design principles. It examines potential vulnerabilities, such as code bugs, logical flaws, and potential attack vectors. The evaluation also considers the adherence to best practices and industry standards for secure coding. Additionally, the system assesses the projects overall security measures, including infrastructure security, data protection, and access controls. It evaluates the implementation of encryption, authentication mechanisms, and secure communication protocols. To achieve a passing score, the smart contract and project must attain a minimum of 80 points out of the total attainable score of 100. This ensures that the system has undergone a rigorous security assessment and meets the required standards for secure operation.

AUDIT FAILED

Important Notes for TROY

Auditor Score =60

Audit Fail

AUDIT
FAILED

Appendix

Finding Categories

Centralization / Privilege

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

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM 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 howblock.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functionsbeing 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 mayresult in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to makethe 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 require statements on the input variables than a setterfunction.

Coding Best Practices

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

Disclaimer

The purpose of this disclaimer is to outline the responsibilities and limitations of the security assessment and smart contract audit conducted by Bladepool/CFG NINJA. By engaging our services, the project owner acknowledges and agrees to the following terms:

1. Limitation of Liability: Bladepool/CFG NINJA shall not be held liable for any damages, losses, or expenses incurred as a result of any contract malfunctions, vulnerabilities, or exploits discovered during the security assessment and smart contract audit. The project owner assumes full responsibility for any consequences arising from the use or implementation of the audited smart contract. 2. No Guarantee of Absolute Security: While Bladepool/CFG NINJA employs industry-standard practices and methodologies to identify potential security risks, it is important to note that no security assessment or smart contract audit can provide an absolute guarantee of security. The project owner acknowledges that there may still be unknown vulnerabilities or risks that are beyond the scope of our assessment. 3. Transfer of Responsibility: By engaging our services, the project owner agrees to assume full responsibility for addressing and mitigating any identified vulnerabilities or risks discovered during the security assessment and smart contract audit. It is the project owner s sole responsibility to ensure the proper implementation of necessary security measures and to address any identified issues promptly. 4. Compliance with Applicable Laws and Regulations: The project owner acknowledges and agrees to comply with all applicable laws, regulations, and industry standards related to the use and implementation of smart contracts. Bladepool/CFG NINJA shall not be held responsible for any non-compliance by the project owner. 5. Third-Party Services: The security assessment and smart contract audit conducted by Bladepool/CFG NINJA may involve the use of third-party tools, services, or technologies. While we exercise due diligence in selecting and utilizing these resources, we cannot be held liable for any issues or damages arising from the use of such third-party services. 6. Confidentiality: Bladepool/CFG NINJA maintains strict confidentiality regarding all information and data obtained during the security assessment and smart contract audit. However, we cannot guarantee the security of data transmitted over the internet or through any other means. 7. Not a Financial Advice: Bladepool/CFG NINJA please note that the information provided in the security assessment or audit should not be considered as financial advice. It is always recommended to consult with a financial professional or do thorough research before making any investment decisions.

By engaging our services, the project owner acknowledges and accepts these terms and releases Bladepool/CFG NINJA from any liability, claims, or damages arising from the security assessment and smart contract audit. It is recommended that the project owner consult legal counsel before entering into any agreement or contract.

CFG NINJA AUDITS