CFG NINJA AUDITS

Security Assessment

LotteryV2 Token

August 18, 2022

Table of Contents

Audit Summary

This report has been prepared for LotteryV2 Token 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.

Project Overview

Token Summary

Parameter	Result
Address	
Name	LotteryV2
Token Tracker	LotteryV2 (undefined)
Decimals	
Supply	
Platform	Binance Smart Chain
compiler	v0.8.16
Contract Name	FlameToken
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/token/0xe55bd75d7ce7bfde26a347a748d080d3acda7ffe
Payment Tx	0xeca43830214651f30d379acfab52a9901860290a4ae4c9eb647b9208e193dcde

Main Contract Assessed Contract Name

Name	Contract	Live
LotteryV2		Yes

TestNet Contract Assessed Contract Name

Name	Contract	Live
LotteryV2	0x49CbC42b5195f9a57AD1B181bA48cc0e18caf59a	Yes

Solidity Code Provided

SolID	File Sha-1	FileName
LotteryV2	b3158545a0ef006cc95d4316ad5bdacea02cbd13	LotteryV2.sol
LotteryV2	54fa8ab0f87984dac2db82c9a91a34b5acd20ee6	IFountain.sol
LotteryV2		
LotteryV2	undefined	
LotteryV2	undefined	

Smart Contract Vulnerability Checks

Vulnerability	Automatic Scan	Manual Scan	Result
Unencrypted Private Data On-Chain	Complete	Complete	Low / No Risk
Code With No Effects	Complete	Complete	Low / No Risk
Message call with hardcoded gas amount	Complete	Complete	Low / No Risk
Hash Collisions With Multiple Variable Length Arguments	Complete	Complete	Low / No Risk
Unexpected Ether balance	Complete	Complete	Low / No Risk
Presence of unused variables	Complete	Complete	Low / No Risk
Right-To-Left-Override control character (U+202E)	Complete	Complete	Low / No Risk
Typographical Error	Complete	Complete	Low / No Risk
DoS With Block Gas Limit	Complete	Complete	Low / No Risk
Arbitrary Jump with Function Type Variable	Complete	Complete	Low / No Risk
Insufficient Gas Griefing	Complete	Complete	Low / No Risk
Incorrect Inheritance Order	Complete	Complete	Low / No Risk
Write to Arbitrary Storage Location	Complete	Complete	Low / No Risk
Requirement Violation	Complete	Complete	Low / No Risk
Missing Protection against Signature Replay Attacks	Complete	Complete	Low / No Risk

Contract Ownership

The contract ownership of LotteryV2 is not currently renounced. The ownership of the contract grants special powers to the protocol creators, making them the sole addresses that can call sensible ownable functions that may alter the state of the protocol.

The current owner is the address 0x3cc6a3fa5bECF00B585E4575537F03d24891bD70 which can be viewed from:
HERE

The owner wallet has the power to call the functions displayed on the priviliged 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.

We recommend the team to use a Multisignature Wallet if contract is not going to be renounced, this will give the ability to the team to have more control over the contract.

KYC Information

The Project Onwers of LotteryV2 has provided KYC Documentation.

KYC Certificated can be found on the Following:
KYC Data

KYC Information Notes:

Auditor Notes: Asked project owner about KYC.

Project Owner Notes: Customer is KYC with PinkSale

Mythx Security Summary Checks

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	LotteryV2.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	LotteryV2.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	LotteryV2.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	LotteryV2.sol	L: 5 C: 0
SWC-104	Pass	Unchecked Call Return Value.	LotteryV2.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	LotteryV2.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	LotteryV2.sol	L: 0 C: 0
SWC-107	Low	Read of persistent state following external call.	LotteryV2.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	LotteryV2.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	LotteryV2.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	LotteryV2.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	LotteryV2.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	LotteryV2.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	LotteryV2.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-114	Pass	Transaction Order Dependence.	LotteryV2.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	LotteryV2.sol	L: 474 C: 15
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	LotteryV2.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	LotteryV2.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	LotteryV2.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	LotteryV2.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	LotteryV2.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	LotteryV2.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	LotteryV2.sol	L: 0 C: 0
SWC-123	Low	Requirement Violation.	LotteryV2.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	LotteryV2.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	LotteryV2.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	LotteryV2.sol	L: 0 C: 0
SWC-127	Medium	Arbitrary Jump with Function Type Variable.	LotteryV2.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	LotteryV2.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-129	Pass	Typographical Error.	LotteryV2.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	LotteryV2.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	LotteryV2.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	LotteryV2.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	LotteryV2.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	LotteryV2.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	LotteryV2.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	LotteryV2.sol	L: 0 C: 0

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

Security Check Details Page

SWC-107 - Reentrancy.

CWE-841: Improper Enforcement of Behavioral Workflow.

Description:

One of the major dangers of calling external contracts is that they can take over the control flow. In the reentrancy attack (a.k.a. recursive call attack), a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways.

Remediation:

The best practices to avoid Reentrancy weaknesses are: Make sure all internal state changes are performed before the call is executed. This is known as the Checks-Effects-Interactions pattern Use a reentrancy lock.

References:

Ethereum Smart Contract Best Practices - Reentrancy

SWC-123 - Requirement Violation

CWE-573: Improper Following of Specification by Caller

Description:

The Solidity require() construct is meant to validate external inputs of a function. In most cases, such external inputs are provided by callers, but they may also be returned by callees. In the former case, we refer to them as precondition violations. Violations of a requirement can indicate one of two possible issues:

A bug exists in the contract that provided the external input.
The condition used to express the requirement is too strong.

Remediation:

If the required logical condition is too strong, it should be weakened to allow all valid external inputs.Otherwise, the bug must be in the contract that provided the external input and one should consider fixing its code by making sure no invalid inputs are provided.

References:

The use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM

SWC-127 - Arbitrary Jump with Function Type Variable

CWE-691: Insufficient Control Flow Management

Description:

Solidity supports function types. That is, a variable of function type can be assigned with a reference to a function with a matching signature. The function saved to such variable can be called just like a regular function.

The problem arises when a user has the ability to arbitrarily change the function type variable and thus execute random code instructions. As Solidity doesn't support pointer arithmetics, it's impossible to change such variable to an arbitrary value. However, if the developer uses assembly instructions, such as mstore or assign operator, in the worst case scenario an attacker is able to point a function type variable to any code instruction, violating required validations and required state changes.

Remediation:

The use of assembly should be minimal. A developer should not allow a user to assign arbitrary values to function type variables.

References:

Solidity CTF

Solidity docs - Solidity Assembly

Solidity docs - Function Types

SWC Information Notes:

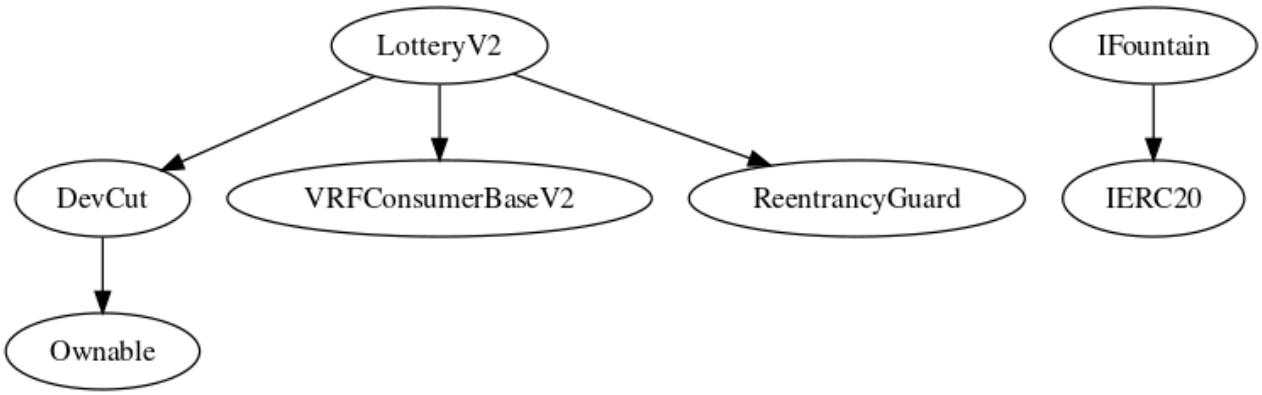
Auditor Notes:

Project Owner Notes:

Call Graph and Inheritance

The contract for LotteryV2 has the following call graph structure

The Project has a Total Supply of and has the following inheritance



Priviliged Functions (onlyOwner)

Function Name	Parameters	Visibility
renounceOwnership	none	public
transferOwnership	address newOwner	public
setBuyTax	_enable bool	external
setSellTax	_pool(address)	external
setAllTaxes	_transferTax uint256, _sell uint256, _buy uint256	external
setMainRouter	_contract address, _tax uint256	external

Important Notes To The Users:

- Stake Protocol Continue to build great products, this is just another example of the quality work.
- The owner have the ability to selfdestruct the contract, customer state this is a safeguard just in case there is a problem they can recover funds.
- No high-risk Exploits/Vulnerabilities Were Found in the Source Code.
- We review the code and scan it for best practices, we have made suggestions to the team and they have addressed all of them.

Audit Passed

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/Stake_Protocol	Pass
Medium	https://stake-protocol.medium.com/introducing-a-game-changer-15f403c53804	Pass
Website	https://stakeprotocol.app/	Pass
Telegram	http://T.me/stakeprotocolportal	Pass

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

Social Media Information Notes:

Auditor Notes: Reviewed the social media, customer could use some additional marketing. However everything is established

Project Owner Notes:

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 advocation 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.