CFG NINJA AUDITS

Security Assessment

Flexvis Token

January 3, 2023

Table of Contents

Audit Summary

This report has been prepared for Flexvis 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	0x9305cbfCa019C10Cf32c9278719fBE264EAc680c
Name	Flexvis
Token Tracker	Flexvis (FLX)
Decimals	18
Supply	650,900
Platform	Binance Smart Chain
compiler	v0.8.9+commit.e5eed63a
Contract Name	Flexvis
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/address/0x9305cbfCa019C10Cf32c9278719fBE264EAc680c#code
Payment Tx	0x488743ea57d1b43c7302f0ef706fe5b778b668611c793468f6ea0c367bcf1170

Project Overview

Risk Analysis Summary

Parameter	Result
Buy Tax	0%
Sale Tax	0%
Is honeypot?	Clean
Can edit tax?	Yes
Is anti whale?	No
Is blacklisted?	Yes
Is whitelisted?	Yes
Holders	Clean
Security Score	70/100
Auditor Score	70/100
Confidence Level	Pass

The following quick summary has been added to the project overview, however there are more details about the audit and their results please read every details.

Main Contract Assessed Contract Name

Name	Contract	Live
Flexvis	0x9305cbfCa019C10Cf32c9278719fBE264EAc680c	Yes

TestNet Contract Assessed Contract Name

Name	Contract	Live
Flexvis	0x0Dfb622F637A087Bcd8BE5dEF0195eb5Ad0eC92B	Yes

Solidity Code Provided

SolID	File Sha-1	FileName
Flexvis	84a3a6bd2d26874b7463ec296f667070727ede9f	Flexvis.sol

Mint Check

The Project Onwers of Flexvis has the ability to Mint New Tokens.

We Recommend the team to create a new contract without a Mint Function.

Mint Notes:

Auditor Notes: owner has the ability to mint new toknes using the minter role, owner also have the option to stop minting.

Project Owner Notes:

Fees Check

The Project Owners of Flexvis does not have the ability to set fees higher than 25% .

Team May have fees defined, however they dont have the ability to set those fees higher than 25%.

Tax Fee Notes:

Auditor Notes: Contract currently have 0% buy tax and 0% sell tax and can't be modified

Project Owner Notes: .

Blacklist Check

The Project Onwers of Flexvis has the ability to Blacklist holders from transferring their tokens.

We Recommend the team to be careful with a blacklist function as this can basically prevent ah holder from buying/selling/transferring their assets. Malicious or compromised owners can trap contracts relying on tokens with a blacklist.

Blacklist Notes:

Auditor Notes: owner have the ability to block and unblock accounts.

Project Owner Notes: .

MaxTx Check

The Project Owners of Flexvis can set max tx amount.

The ability to set MaxTx can be used as bad actor, this can limit the ability of investors to sale their tokens at any given time if is set too low..

We recommend the project to set MaxTx to Total Supply or simiar to avoid swap or transfer from failures

MaxTX Notes:

Auditor Notes: Owner has Max buy and Max sell up to .

Project Owner Notes:

Pause Trade Check

The Project Owners of Flexvis have the ability to stop or pause trading.

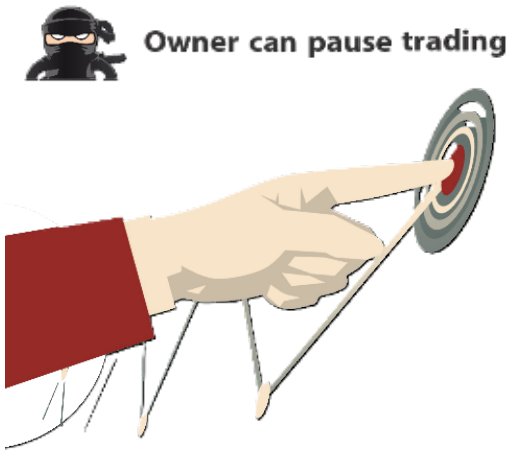
We recommend the team to only allow Open Trade and never use Stop Trade as this will be catastrofic for the project and investors.

We recommend the team to create a new contract without stop trade..

Pause Trade Notes:

Auditor Notes: Owner has the ability to pause transfers

Project Owner Notes:



Contract Ownership

The contract ownership of Flexvis 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 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.

Liquidity Ownership

The token does not have liquidity at the moment of the audit, block 24467927

If liquidity is unlocked, then the token developers can do what is infamously known as 'rugpull'. Once investors start buying token from the exchange, the liquidity pool will accumulate more and more coins of established value (e.g., ETH or BNB or Tether). This is because investors are basically sending these tokens of value to the exchange, to get the new token. Developers can withdraw this liquidity from the exchange, cash in all the value and run off with it. Liquidity is locked by renouncing the ownership of liquidity pool (LP) tokens for a fixed time period, by sending them to a time-lock smart contract. Without ownership of LP tokens, developers cannot get liquidity pool funds back. This provides confidence to the investors that the token developers will not run away with the liquidity money. It is now a standard practice that all token developers follow, and this is what really differentiates a scam coin from a real one.

Read More

KYC Information

The Project Onwers of Flexvis 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 passed KYC with PinkSale.

Project Owner Notes:

Smart Contract Vulnerability Checks

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	Flexvis.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	Flexvis.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	Flexvis.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	Flexvis.sol	L: 1636 C: 4
SWC-104	Pass	Unchecked Call Return Value.	Flexvis.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	Flexvis.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	Flexvis.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	Flexvis.sol	L: 0 C: 0
SWC-108	Low	State variable visibility is not set..	Flexvis.sol	L: 1482 C: 29
SWC-109	Pass	Uninitialized Storage Pointer.	Flexvis.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	Flexvis.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	Flexvis.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	Flexvis.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-113	Pass	Multiple calls are executed in the same transaction.	Flexvis.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	Flexvis.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	Flexvis.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	Flexvis.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	Flexvis.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	Flexvis.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	Flexvis.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	Flexvis.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	Flexvis.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	Flexvis.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	Flexvis.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	Flexvis.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	Flexvis.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	Flexvis.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	Flexvis.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-128	Pass	DoS With Block Gas Limit.	Flexvis.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	Flexvis.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	Flexvis.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	Flexvis.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	Flexvis.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	Flexvis.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	Flexvis.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	Flexvis.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	Flexvis.sol	L: 0 C: 0

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

Smart Contract Vulnerability Details

SWC-103 - Floating Pragma.

CWE-664: Improper Control of a Resource Through its Lifetime.

References:

Description:

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation:

Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

References:

Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.

Smart Contract Vulnerability Details

SWC-108 - State Variable Default Visibility

CWE-710: Improper Adherence to Coding Standards

Description:

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

Remediation:

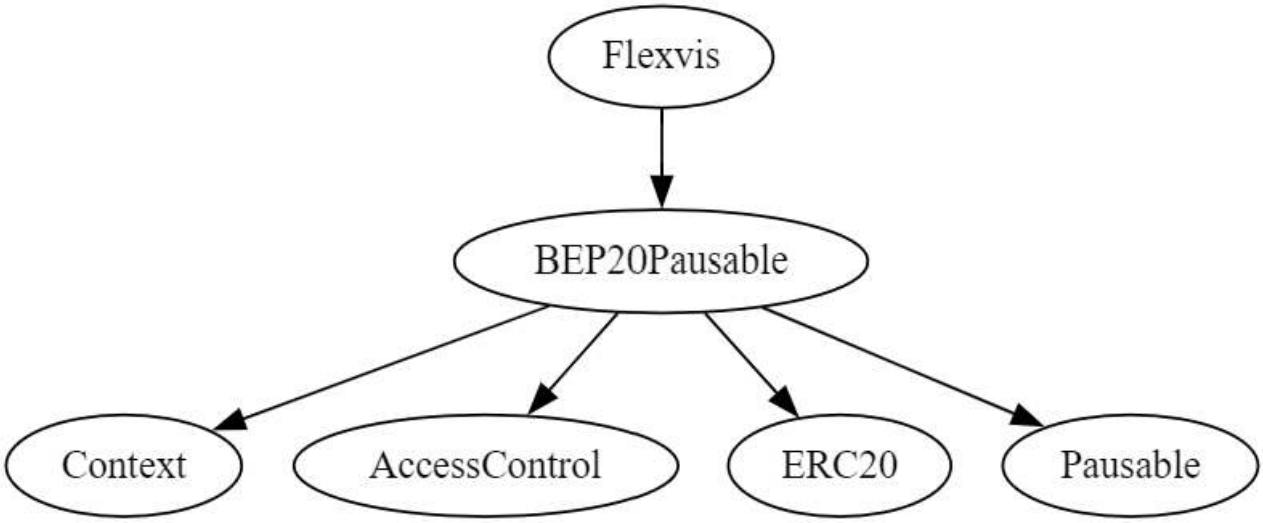
Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

References:

Ethereum Smart Contract Best Practices - Explicitly mark visibility in functions and state variables

Call Graph and Inheritance

The contract for Flexvis has the following call graph structure



Priviliged Functions (onlyOwner)

Function Name	Parameters	Visibility

Assessment Results

- Contract has taxes up to 0%.
- Owner can't set max tx amount.
- Owner can pause trading.
- Owner has mint and blacklist functions.
- No high-risk Exploits/Vulnerabilities Were Found in the Source Code.

Audit Fail

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/flexvis	Pass
Instagram	https://www.instagram.com/flexvis/	Pass
Website	https://www.flexvis.io/	Pass
Telegram	https://t.me/flexvis	Pass

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

Social Media Information Notes:

Auditor Notes: undefined

Project Owner Notes: Projects owners have other socials: https://linktr.ee/flexvis

Technical Findings Summary

Classification of Risk

Severity	Description
Critical	risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
Major	risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
Medium	risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
Minor	risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
Informational	errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
Critical	0	0	0
Major	0	0	0
Medium	0	0	0
Minor	0	0	0
Informational	0	0	0
Total	0	0	0

FLX-04 | Centralized Risk In addLiquidity.

Category	Severity	Location	Status
Coding Style	Major	Flexvis.sol: 720,12	Pending

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 FLX-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.

Remediation

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

FLX-11 | .

Category	Severity	Location	Status
	Major	Flexvis.sol: 0,0	Pending

Description

Remediation

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

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.