



CFG NINJA AUDITS

Security Assessment

Musk vs Zuck Token

June 24, 2023

Audit Status: Fail

Audit Edition: Pinksale













POWERED BY
BLADE POOL

Risk Analysis


















Classifications of Manual Risk Results

Classification	Description
 Critical	Danger or Potential Problems.
 High	Be Careful or Fail test.
 Low	Pass, Not-Detected or Safe Item.
 Informational	Function Detected

Manual Code Review Risk Results

Contract Privilege	Description
 Buy Tax	0%
 Sale Tax	0%
 Cannot Sale	Pass
 Cannot Sale	Pass
 Max Tax	3%
 Modify Tax	Yes
 Fee Check	Pass
 Is Honeygot?	Not Detected
 Trading Cooldown	Not Detected
 Can Pause Trade?	Not Detected.



Contract Priviledge	Description
 Pause Transfer?	Not Detected
 Max Tx?	Pass
 Is Anti Whale?	Not Detected
 Is Anti Bot?	Not Detected
 Is Blacklist?	Not Detected
 Blacklist Check	Pass
 is Whitelist?	Not Detected
 Can Mint?	Pass
 Is Proxy?	Not Detected
 Can Take Ownership?	Not Detected
 Hidden Owner?	Not Detected
 Owner	0x1F4D5f72068f0575F900D5004Bc9649b0E3800b8
 Self Destruct?	Not Detected
 External Call?	Not Detected
 Other?	Not Detected
 Holders	1
 Auditor Confidence	low

The following quick summary it's added to the project overview; however, there are more details about the audit and its results. Please read every detail.



Project Overview

Token Summary

Parameter	Result
Address	0x2891B63a8eD0d09C07205505580d65768677BC37
Name	Musk vs Zuck
Token Tracker	Musk vs Zuck (MUSKvsZUCK)
Decimals	9
Supply	100,000,000,000,000
Platform	Ethereum
compiler	v0.8.17+commit.8df45f5f
Contract Name	MuskvsZuck
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/address/0x2891b63a8ed0d09c07205505580d65768677bc37#code
Payment Tx	Corporate



Main Contract Assessed Contract Name

Name	Contract	Live
Musk vs Zuck	0x2891B63a8eD0d09C07205505580d65768677BC37	Yes

TestNet Contract Assessed Contract Name

Name	Contract	Live
Musk vs Zuck	0x3fC91A3afd70395Cd496C647d5a6CC9D4B2b7FAD	Yes

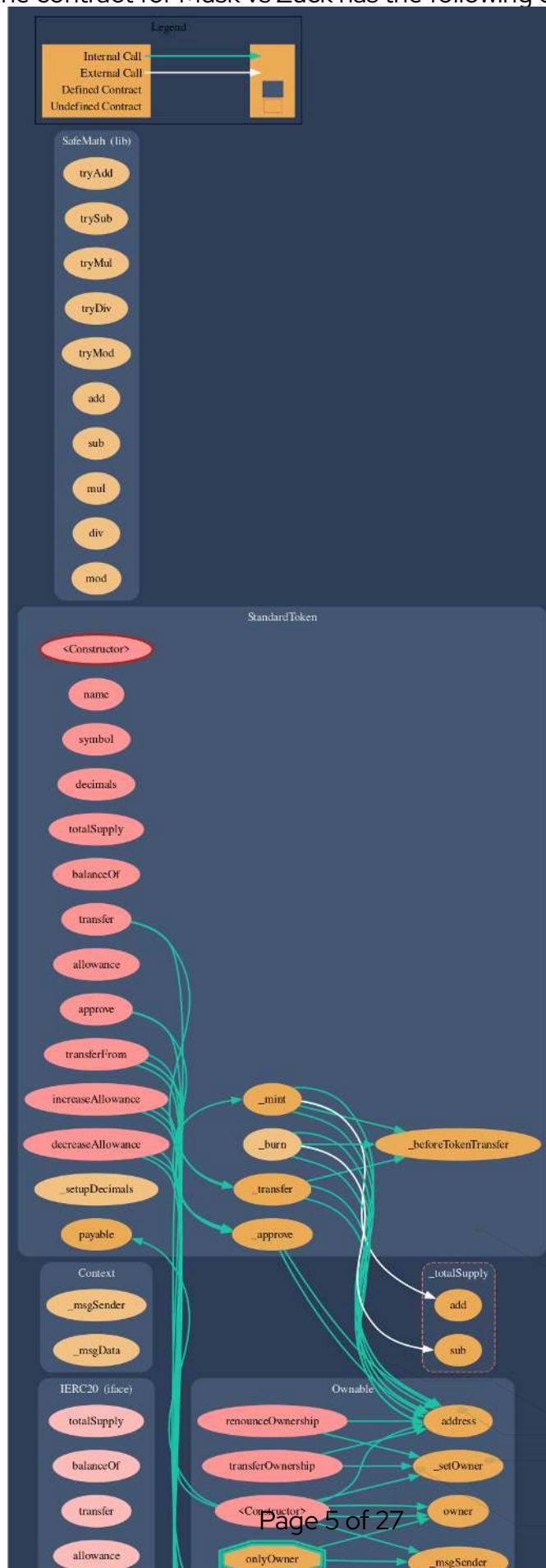
Solidity Code Provided

SolID	File Sha-1	FileName
MuskvsZuck	c13d13a571cf67e50a9dd35ceb3976515e0ccb52	MuskvsZuck.sol
MuskvsZuck		
MuskvsZuck		
MuskvsZuck		



Call Graph

The contract for Musk vs Zuck has the following call graph structure.



Smart Contract Vulnerability Checks

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are specific to smart contracts.

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	MuskvsZuck.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	MuskvsZuck.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	MuskvsZuck.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	MuskvsZuck.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	MuskvsZuck.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	MuskvsZuck.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	MuskvsZuck.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	MuskvsZuck.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	MuskvsZuck.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	MuskvsZuck.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	MuskvsZuck.sol	L: 0 C: 0



ID	Severity	Name	File	location
SWC-111	Pass	Use of Deprecated Solidity Functions.	MuskvsZuck.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	MuskvsZuck.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	MuskvsZuck.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	MuskvsZuck.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	MuskvsZuck.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	MuskvsZuck.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	MuskvsZuck.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	MuskvsZuck.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	MuskvsZuck.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	MuskvsZuck.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	MuskvsZuck.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	MuskvsZuck.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	MuskvsZuck.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	MuskvsZuck.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	MuskvsZuck.sol	L: 0 C: 0



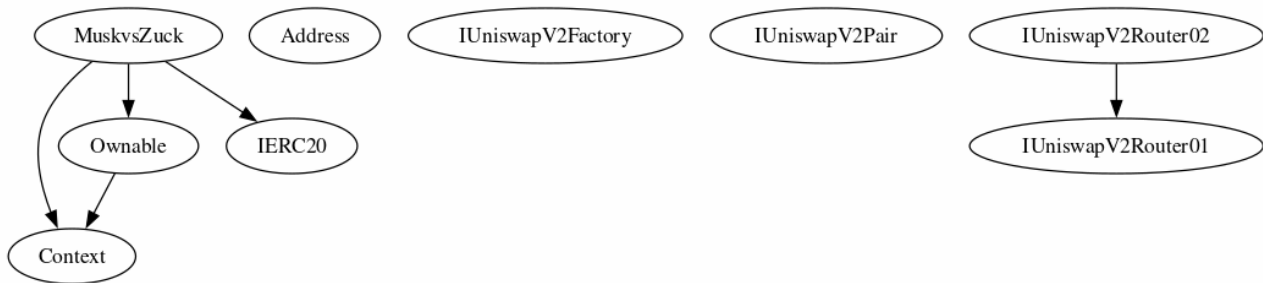
ID	Severity	Name	File	location
SWC-126	Pass	Insufficient Gas Griefing.	MuskvsZuck.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	MuskvsZuck.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	MuskvsZuck.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	MuskvsZuck.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	MuskvsZuck.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	MuskvsZuck.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	MuskvsZuck.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	MuskvsZuck.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	MuskvsZuck.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	MuskvsZuck.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	MuskvsZuck.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.



Inheritance

The contract for Musk vs Zuck has the following inheritance structure.



Smart Contract Advance Checks



ID	Severity	Name	Result	Status
MUSKvsZUC K-01	Low	Potential Sandwich Attacks.	Pass	Not-Found
MUSKvsZUC K-02	Informational	Function Visibility Optimization	Fail	Detected
MUSKvsZUC K-03	Low	Lack of Input Validation.	Fail	Detected
MUSKvsZUC K-04	High	Centralized Risk In addLiquidity.	Pass	Not-Found
MUSKvsZUC K-05	Low	Missing Event Emission.	Fail	Detected
MUSKvsZUC K-06	Low	Conformance with Solidity Naming Conventions.	Pass	Not-Found
MUSKvsZUC K-07	Low	State Variables could be Declared Constant.	Pass	Not-Found
MUSKvsZUC K-08	Low	Dead Code Elimination.	Pass	Not-Found
MUSKvsZUC K-09	High	Third Party Dependencies.	Fail	Detected
MUSKvsZUC K-10	High	Initial Token Distribution.	Pass	Not-Found
MUSKvsZUC K-11	High	claimStuckTokens can claim own tokens.	Fail	Detected
MUSKvsZUC K-12	High	Centralization Risks In The X Role	Pass	Not-Found
MUSKvsZUC K-13	Informational	Extra Gas Cost For User..	Fail	Detected



ID	Severity	Name	Result	Status
MUSKvsZUC K-14	Medium	Unnecessary Use Of SafeMath	Pass	Not Detected
MUSKvsZUC K-15	Medium	Symbol Length Limitation due to Solidity Naming Standards.	Fail	Detected
MUSKvsZUC K-16	Medium	Taxes can be up to 100%	Pass	Not Detected
MUSKvsZUC K-17	Logical Issue	Highly Permissive Role Access.,`	Fail	Detected
MUSKvsZUC K-18	Critical	Stop Transactions by using Enable Trade.	Pass	Not Detected



MUSKvsZUCK-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	 Informational	MuskvsZuck.sol: L: 256 C: 14	 Detected

Description

The following functions are declared as public and are not invoked in any of the contracts contained within the projects scope:

Function Name	Parameters	Visibility
updateFeeBuy		public
updateFeeSell		public
updateFeeSell		public

The functions that are never called internally within the contract should have external visibility

Remediation



We advise that the function's visibility specifiers are set to external, and the array-based arguments change their data location from memory to calldata, optimizing the gas cost of the function.

References:

external vs public best practices.



MUSKvsZUCK-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	 Low	MuskvsZuck.sol: L: 787 C: 14, L: 289 C: 14	 Detected

Description

The given input is missing the check for the non-zero address.

The given input is missing the check for the setSwapEnabled, .

Remediation



We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
...
require(receiver != address(0), "Receiver is the zero address");
...
...
require(value X limitation, "Your not able to do this function");
...
```

We also recommend customer to review the following function that is missing a required validation. setSwapEnabled, .



MUSKvsZUCK-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	 Low	MuskvsZuck.sol: L: 557 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.

Remediation

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



MUSKvsZUCK-09 | Third Party Dependencies.

Category	Severity	Location	Status
Volatile Code	 High	MuskvsZuck.sol: L: 68, C: 14	 Detected

Description

The contract is serving as the underlying entity to interact with third party `0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470` protocols. The scope of the audit treats 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Remediation



We understand that the business logic of Musk vs Zuck requires interaction with `0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470`, etc. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Project Action

Update Library to latest version.



MUSKvsZUCK-11 | claimStuckTokens can claim own tokens..

Category	Severity	Location	Status
Optimization	 High	MuskvsZuck.sol: L: 554 C: 14	 Detected

Description

claimStuckTokens can claim own tokens, effectively creating a problem for the contract.



Remediation

add a require function to avoid own address.

Project Action



MUSKvsZUCK-13 | Extra Gas Cost For User.

Category	Severity	Location	Status
Logical Issue	 Informational	MuskvsZuck.sol: L: 702, C: 0	 Detected

Description

The user may trigger a tax distribution during the transfer process, which will cost a lot of gas and it is unfair to let a single user bear it.



Remediation

We advise the client to make the owner responsible for the gas costs of the tax distribution.

Project Action



MUSKvsZUCK-15 | Symbol Length Limitation due to Solidity Naming Standards.

Category	Severity	Location	Status
Logical Issue	 Medium	MuskvsZuck.sol: L: 0 C: 0	 Not Detected

Description

The Symbol is one of the most important part of the identity of a project, as industry standard this usually match the leng of stock market traditions. The Symbol used in contract is too long, this can create issues for most Dapps including uniswap, pancakeSwap and Metamask.

The current character limit for metamask is 11 and will be increased overtime to 20 characters, however APIS like uniswap,pancakeswap and coinmarketcap may have issues readins such symbols.

Remediation

We advise removing the limiting the symbol to more industry standard naming to avoid issues with dapps and others. is suggested to use between 3 to 4 characters for a project, however for a coin is recommended no more than seven.

Project Action



MUSKvsZUCK

References:

Increase Token Symbol Length - Metamask



MUSKvsZUCK-17 | Highly Permissive Role Access.

Category	Severity	Location	Status
Logical Issue	 Logical Issue	MuskvsZuck.sol:	 Detected

Description

The following is an example scenario of how the owner may manipulate the funds of accounts which do not belong to it.

During initialization, a set of "excluded" (via `excludeFromReward(..)`) accounts is created, whose balances - for the sake of argument - are small relative to the total supply, but are big enough in absolute values to consider those accounts rich.

Let X = total reflection balance of the "excluded" accounts.

$X \gg 0$

Let T = total tax collected on all the transfers that were

$T \gg 0$

needed to set up the "excluded" accounts.

Let Z

$0 = (X - T) - Z$

$Z \gg 0$

$Z \gg 0$

Let Y = total token balance of the "excluded" accounts.

$Y \gg 0$

Let W

$0 = (Y - W) - V$

$V \gg 0$

Then, an additional "excluded" account A is introduced, whose reflection balance at the moment of exclusion was X . After that, multiple transfers occur, with the total tax accrued equal to T .

According to the formulas in the contract, the reflection-to-token exchange rate at this point is:

If now account A gets "included" (via `includeInReward(..)`), the exchange rate becomes:

The ratio between the new and the old rate:

Introducing some convenience definitions:

It is possible to rewrite the expression as:

In other words, the holders of plain token i.e. "excluded" wallets, become $\frac{X}{Y}$ % richer (in reflections), and the rest "not excluded" wallets become poorer by the respective percentage. Becoming richer for excluded wallets will not be immediately observable: for example, the ERC20 balance of such an account will not increase as a number;



nevertheless, it can be accounted via the contract interface, that those wallets actually became richer; eventually, the ecosystem would recognize who has become relatively richer/poorer. It is possible to make the gain for the "excluded" wallets immediately obvious by "including" such accounts - that would lead to exchanging their tokens to reflections with the new rate; as a result, their ERC20 balances would reflect the gain literally. Example: if $\frac{1}{100}$ is 8% of $\frac{1}{100}$, and is 51.1% of $\frac{1}{100}$, then the rate $\frac{1}{100}$ increases by 10%.

Remediation

Remove the functionality or provide documentation with its description.

Project Action








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.
 High	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
 Low	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 recommended to improve the code's style 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	1	0	0
 High	2	0	0
 Medium	0	0	0
 Low	3	0	0
 Informational	2	0	0
Total	8	0	0



Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/muskvszuck_wtf	Pass
Other	contact@muskvszuck.wtf	Pass
Website	https://muskvszuck.wtf	Pass
Telegram	https://t.me/MuskvsZuckportal	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:



Assessment Results

Score Results

Review	Score
Overall Score	71/100
Auditor Score	75/100
Review by Section	Score
Manual Scan Score	18/53
SWC Scan Score	37 /37
Advance Check Score	16 /19

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 80 Points, if a project does not attain 80% is an automatic failure. Read our notes and final assessment below.

Audit Fail



Assessment Results

Important Notes:

- No issues or vulnerabilities were found.
- The contract could use some minor improvements.
- Please DYOR on the project.

Auditor Score =75
Audit Fail



Appendix

Finding Categories

Centralization / Privilege

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

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM 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 how `block.timestamp` works.

Control Flow

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

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the 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 requirements on the input variables than a setter function.



Coding Best Practices

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



Disclaimer

CFGNINJA has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocacy for the Project, and users relying on this 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 the Project in question may pursue, and 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 entirely 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. Information is provided 'as is, and CFGNINJA is under no covenant to audited completeness, accuracy, or solidity of the contracts. 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 or actions with regards to the information provided in this audit report.

The assessment services provided by CFGNINJA are subject to dependencies and are under continuing development. You agree that your access 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 with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.

