CFG NINJA AUDITS

Security Assessment

HOPIUM Token

November 1, 2022

Table of Contents

Assessment Summary

This report has been prepared for HOPIUM 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.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders
- Thorough line-by-line manual review of the entire codebase by industry experts.

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 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	0	0	0
Major	1	1	0
Medium	0	0	0
Minor	1	1	0
Informational	0	0	0
Total	2	2	0

Project Overview

Token Summary

Parameter	Result
Address	0x2f9d9629a62cD206E6adAF53DFD0D68baE36eBcf
Name	HOPIUM
Token Tracker	HOPIUM (HOPIUM)
Decimals	9
Supply	1,000,000,000,000
Platform	Binance Smart Chain
compiler	v0.8.4+commit.c7e474f2
Contract Name	LiquidityGeneratorToken
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/address/#code
Payment Tx	

Main Contract Assessed
Contract Name

Name	Contract	Live
HOPIUM	0x2f9d9629a62cD206E6adAF53DFD0D68baE36eBcf	Yes

TestNet Contract was Not Assessed

Solidity Code Provided

SolID	File Sha-1	FileName
LiquidityGenerator	3f58431742f1d7caddb877f74cd0b55bd5ee6023	LiquidityGeneratorv1.sol

Mint Check

The project owners of HOPIUM do not have a mint function in the contract, owner cannot mint tokens after initial deploy.

The Project has a Total Supply of 1,000,000,000,000 and cannot mint any more than the Max Supply.

Mint Notes:

Auditor Notes:

Project Owner Notes:

Fees Check

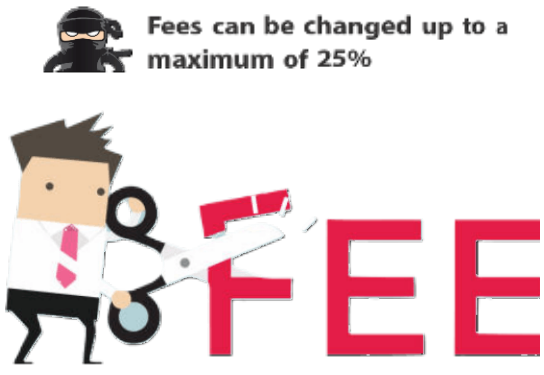
The project owners of HOPIUM do not have the ability to set fees higher than 25% .

The team May have fees defined; however, they can't set those fees higher than 25% or may not be able to configure the same.

Tax Fee Notes:

Auditor Notes: The contract currently has 15% buy and 15% sale taxes, and cannot be set higher than 25%

Project Owner Notes: .



Blacklist Check

The project owners of HOPIUM do not have a blacklist function their contract.

The Project allow owners to transfer their tokens without any restrictions.

Token owner cannot blacklist the contract: Malicious or compromised owners can trap contracts relying on tokens with a blacklist.

Blacklist Notes:

Auditor Notes:

Project Owner Notes: undefined

MaxTx Check

The Project Owners of HOPIUM cannot set max tx amount

The Team allows any investors to swap, transfer or sell their total amount if needed.

MaxTX Notes:

Auditor Notes:

Project Owner Notes:

Project Has No MaxTX

Pause Trade Check

The Project Owners of HOPIUM don't have the ability to stop or pause trading.

The Team has done a great job to avoid stop trading, and investors has the ability to trade at any given time without any problems

Pause Trade Notes:

Auditor Notes: .

Project Owner Notes:

Contract Ownership

The contract ownership of HOPIUM 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 0x959415dd0f62eb9f2c5f2d56a59b052ed597daa1 which can be viewed:
HERE

The owner wallet has the power to call the functions displayed on the privileged functions chart below, if the owner's wallet is compromised, they could exploit these privileges.

We recommend the team renounce ownership at the right time, if possible, or gradually migrate to a timelock with governing functionalities regarding transparency and safety considerations.

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

Liquidity Ownership

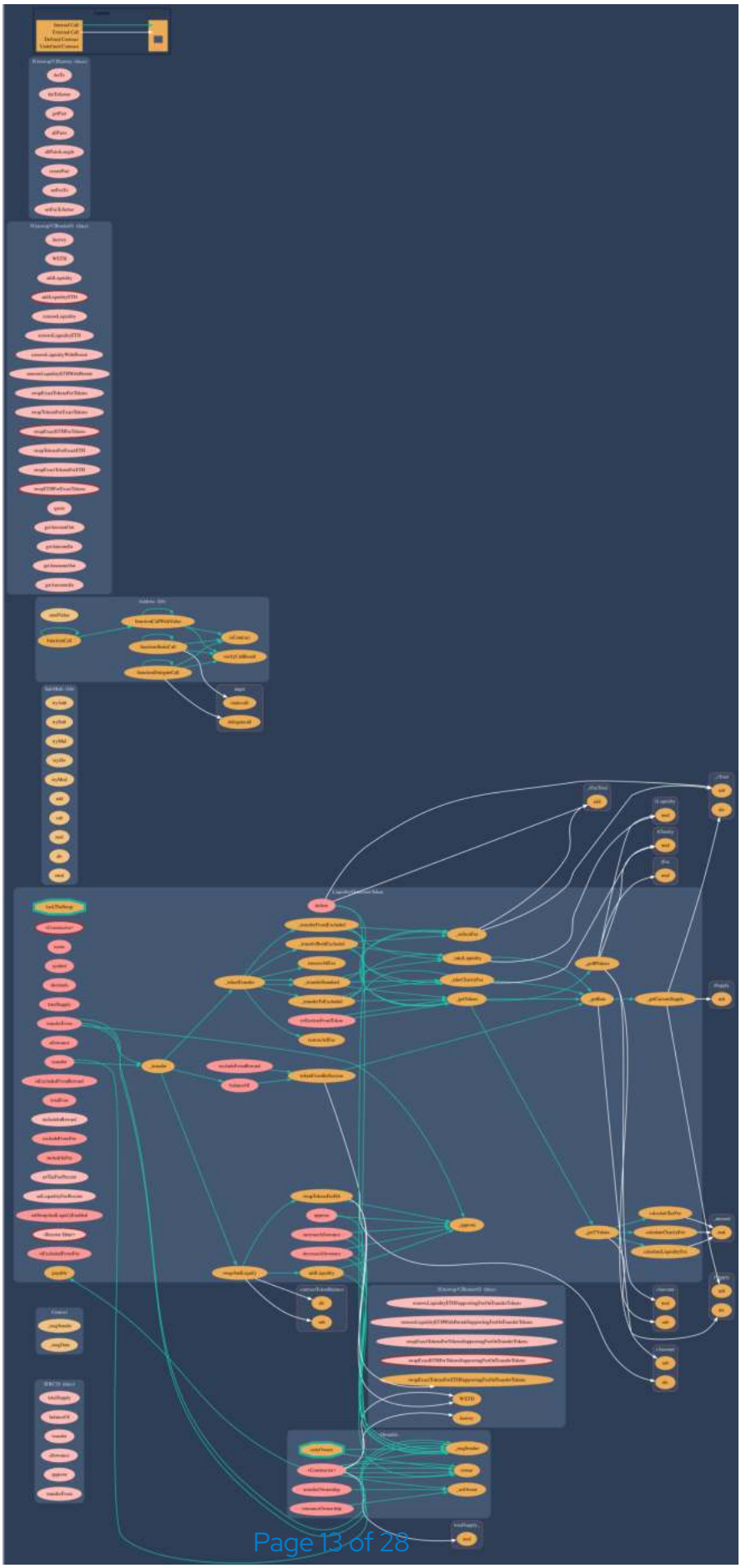
The token does not have liquidity at the moment of the audit, block 14393881

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

Call Graph

The contract for HOPIUM has the following call graph structure.



KYC Information

The Project Owners of HOPIUM have provided KYC Documentation.

KYC Certificated can be found on the Following:
KYC Data

KYC Information Notes:

Auditor Notes: Customer is processing Kyc with PinkSale

Project Owner Notes:

Smart Contract Vulnerability Checks

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	LiquidityGeneratorv1.sol	L: 3 C: 0
SWC-104	Pass	Unchecked Call Return Value.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-108	Low	State variable visibility is not set..	LiquidityGeneratorv1.sol	L: 957 C: 9
SWC-109	Pass	Uninitialized Storage Pointer.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	LiquidityGeneratorv1.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-112	Pass	Delegate Call to Untrusted Callee.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	LiquidityGeneratorv1.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-126	Pass	Insufficient Gas Griefing.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	LiquidityGeneratorv1.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	LiquidityGeneratorv1.sol	L: 0 C: 0

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

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:

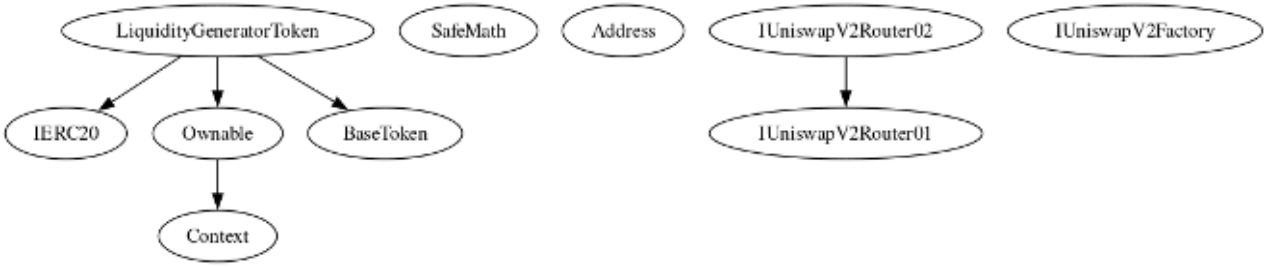
Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

References:

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

Inheritance

The contract for HOPIUM has the following inheritance structure.



Privileged Functions (onlyOwner)

Function Name	Parameters	Visibility
renounceOwnership		public
transferOwnership		public
excludeFromReward		external
includeInReward		public
ExcludeFromFees		public
includeInFee		external
setTaxFeePercent		external
setLiquidityFeePercent		external

Assessment Results

- The Following is a PinkSale Generated Contract, is important to do a Do Your Own Research on the project, while code maybe safe is important to understand the Project itself.
- Security Vulnerabilities have not been found and no major issues during the advance check and testing of the contract.
- No high-risk Exploits/Vulnerabilities Were Found in the Source Code.

Audit Passed

Smart Contract Advance Checks

ID	Severity	Name	Result	Status
HOPIUM-01	Minor	Potential Sandwich Attacks.	Fail	Pending
HOPIUM-02	Informational	Function Visibility Optimization	Pass	Acknowledged
HOPIUM-03	Minor	Lack of Input Validation.	Pass	Pending
HOPIUM-04	Major	Centralized Risk In addLiquidity.	Pass	Resolved
HOPIUM-05	Major	Missing Event Emission.	Pass	Pending
HOPIUM-06	Minor	Conformance with Solidity Naming Conventions.	Pass	Pending
HOPIUM-07	Major	State Variables could be Declared Constant.	Pass	Pending
HOPIUM-08	Major	Dead Code Elimination.	Pass	Pending
HOPIUM-09	Major	Third Party Dependencies.	Pass	Pending
HOPIUM-10	Major	Initial Token Distribution.	Fail	Pending
HOPIUM-11	Major	Modification of Math Library	Pass	Pending
HOPIUM-12	Major	Centralization Risks In The X Role	Pass	Pending
HOPIUM-13	Informational	Extra Gas Cost For User..	Pass	Pending

During our assessment review, we manually check against the following test cases, this help us identify potential missing items from any automated tools. Is an additional safety measure to ensure the contract is safe and follow industry standards.

HOPIUM-01 | Potential Sandwich Attacks.

Category	Severity	Location	Status
Security	Minor	LiquidityGeneratorv1.sol: 1546,13	Pending

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by back running (after the transaction being attacked) a transaction to sell the asset. The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- swapExactTokensForETHSupportingFeeOnTransferTokens()
- addLiquidityETH()

Remediation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

Referrences:

What Are Sandwich Attacks in DeFi — and How Can You Avoid Them?.

HOPIUM-10 | Initial Token Distribution.

Category	Severity	Location	Status
Centralization / Privilege	Major	LiquidityGeneratorv1.sol: 0,0	Pending

Description

All of the HOPIUM tokens are sent to the contract deployer when deploying the contract. This could be a
centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.

Remediation

We recommend the team to be transparent regarding the initial token distribution process, and the team
shall make enough efforts to restrict the access of the private key.

Project Action

Social Media Checks

Social Media	URL	Result
Twitter	https://www.twitter.com/hopiumdefi	Pass
Other	https://discord.gg/j6rhtr7vp9	Pass
Website	http://www.HopiumDeFi.com	Pass
Telegram	https://t.me/hopiumcommunity	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:

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 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 advocation 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.