



CFG NINJA AUDITS

Security Assessment

Token Token

June 15, 2023

Audit Status: Fail





Audit Edition: Advanced












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 Priviledge	Description
 Buy Tax	4%
 Sale Tax	4%
 Cannot Sale	Pass
 Cannot Sale	Pass
 Max Tax	4%
 Modify Tax	Yes
 Fee Check	Pass
 Is Honeypot?	Not Detected
 Trading Cooldown	Not Detected
 Can Pause Trade?	Fail
 Pause Transfer?	Detected, dev needs to enable trading.



Contract Priviledge	Description
 Max Tx?	Pass
 Is Anti Whale?	Not Detected
 Is Anti Bot?	Not Detected
 Is Blacklist?	Detected
 Blacklist Check	Fail
 is Whitelist?	Not Detected
 Can Mint?	Pass
 Is Proxy?	Not Detected
 Can Take Ownership?	Not Detected
 Hidden Owner?	Not Detected
 Owner	0xc6a46d2dbdc50f922f16d634f7b16a4af863aae2
 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	0xf1754d47DC93B3d3FF6BAE670b388E4eF9e81d54
Name	Token
Token Tracker	Token (X)
Decimals	18
Supply	100,000,000,000,000
Platform	Binance Smart Chain
compiler	v0.6.12+commit.27d51765
Contract Name	X
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/token/0xf1754d47dc93b3d3ff6bae670b388e4ef9e81d54#code
Payment Tx	Corporate



Project Overview

Simulation Summary

Parameter	Result
Transfer From Owner	Pass
Transfer From Holder	Pass
Add Liquidity	Fail
RemoveLiquidity	Pass
Buy from Owner	Pass
Buy from Holder	Pass
Sale from Owner	Pass
Sale from Holder	Pass
Remove Liquidity	Pass
SwapAndLiquify	Fail
SwapAndSale w/Fee	Fail
SwapAndSale TX	
SwapAndSaleNoFee	Pass
SwapAndSale No/Fee TX	
ExcludeFromFees	Pass
LaunchPad	PinkSale



Parameter	Result
Pool Creation	Pass
Pool Creation TX	
Pool Finalize	Pass
Pool Finalize TX	
Enable	Fail

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.



Main Contract Assessed

Contract Name

Name	Contract	Live
Token	0xf1754d47DC93B3d3FF6BAE670b388E4eF9e81d54	Yes

TestNet Contract Assessed

Contract Name

Name	Contract	Live
Token	0xE6cEddEA1D07fbd9D6961A34ACb41b07435D0C5c	Yes

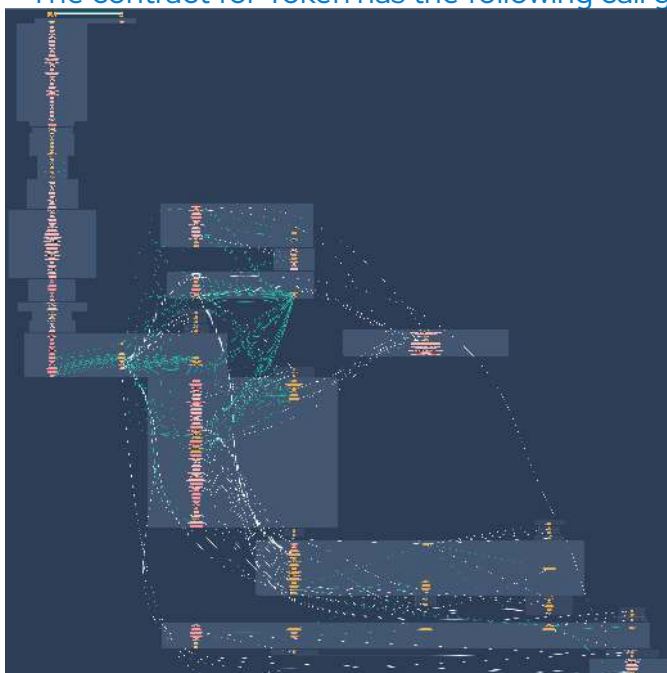
Solidity Code Provided

SolID	File Sha-1	FileName
X	f5db3921706fbe7a91f0b458c466825855cd7c85	X.sol
X		
X		
X		



Call Graph

The contract for Token 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	X.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	X.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	X.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	X.sol	L: 6 C: 0, L:1767 C:0
SWC-104	Pass	Unchecked Call Return Value.	X.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	X.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	X.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	X.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	X.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	X.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	X.sol	L: 0 C: 0



ID	Severity	Name	File	location
SWC-111	Pass	Use of Deprecated Solidity Functions.	X.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	X.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	X.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	X.sol	L: 0 C: 0
SWC-115	Low	Authorization through tx.origin.	X.sol	L: 794 C: 84, L: 939 C: 88
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	X.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	X.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	X.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	X.sol	L: 0 C: 0
SWC-120	Low	Potential use of block.number as source of randomness.	X.sol	L: 839 C: 18, L: 900 C: 39, L: 920 C: 93
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	X.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	X.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	X.sol	L: 0 C: 0



ID	Severity	Name	File	location
SWC-124	Pass	Write to Arbitrary Storage Location.	X.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	X.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	X.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	X.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	X.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	X.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	X.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	X.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	X.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	X.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	X.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	X.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	X.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-115 - Authorization through tx.origin

CWE-477: Use of Obsolete Function

Description:

tx.origin is a global variable in Solidity which returns the address of the account that sent the transaction. Using the variable for authorization could make a contract vulnerable if an authorized account calls into a malicious contract. A call could be made to the vulnerable contract that passes the authorization check since tx.origin returns the original sender of the transaction which in this case is the authorized account.

Remediation:

tx.origin should not be used for authorization. Use msg.sender instead.

References:

Solidity Documentation - tx.origin

Ethereum Smart Contract Best Practices - Avoid using tx.origin

SigmaPrime - Visibility.



Smart Contract Vulnerability Details

SWC-120 - Weak Sources of Randomness from Chain Attributes

CWE-330: Use of Insufficiently Random Values

Description:

Solidity allows for ambiguous naming of state variables when inheritance is used. Contract A with a variable x could inherit contract B that also has a state variable x defined. This would result in two separate versions of x, one of them being accessed from contract A and the other one from contract B. In more complex contract systems this condition could go unnoticed and subsequently lead to security issues.

Shadowing state variables can also occur within a single contract when there are multiple definitions on the contract and function level.

Remediation:

Using commitment scheme, e.g. RANDAO. Using external sources of randomness via oracles, e.g. Oraclize. Note that this approach requires trusting in oracle, thus it may be reasonable to use multiple oracles. Using Bitcoin block hashes, as they are more expensive to mine.

References:

How can I securely generate a random number in my smart contract?)

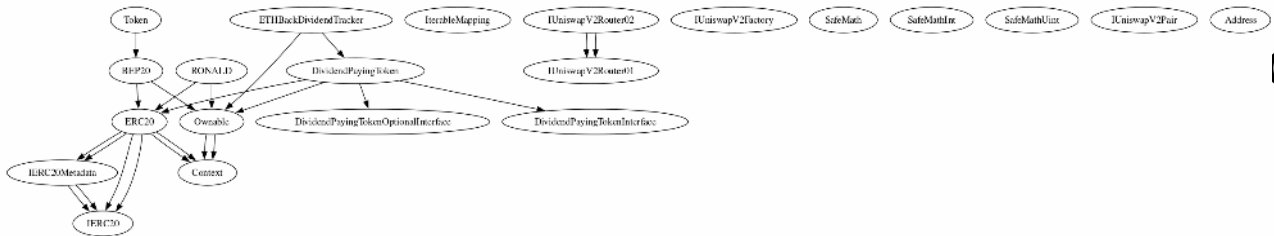
When can BLOCKHASH be safely used for a random number? When would it be unsafe?

The Run smart contract.



Inheritance

The contract for Token has the following inheritance structure.



Privileged Functions (onlyOwner)

Please Note if the contract is Renounced none of this functions can be executed.

Function Name	Parameters	Visibility
renounceOwnership		Public
transferOwnership	address newOwner	Public
setSwapTokensAtAmount	uint256 newValue	Public
updateDividendTracker	address newAddress	Public
updateUniswapV2Router	address newAddress	Public
excludeFromFees	address account, bool excluded	Public
excludeMultipleAccountsFromFees	address[] calldata accounts, bool excluded	public
setMarketingWallet_1	address payable wallet	External
setMarketingWallet_2	address payable wallet	External



Function Name	Parameters	Visibility
setMarketingWallet_3	address payable wallet	External
setETHRewardsFee	uint256 value	External
setLiquiditFee	uint256 value	External
setMarketingFee	uint256 value	External
setMarketingFee_2	uint256 value	External
setETHRewardsFee_2	uint256 value	External
setLiquiditFee_2	uint256 value	External
setAutomatedMarketMakerPair	address pair, bool value	Public
updateGasForProcessing	uint256 newValue	External
updateClaimWait	uint256 claimWait	External
excludeFromDividends	address account	External
setSwapAndLiquifyEnabled	bool status	Public



Function Name	Parameters	Visibility
setAirDropEnable	bool status	Public
setTransferFree	bool s	Public
bclistAddress	address[] calldata addresses, bool value	Public
multi_bclist	address[] calldata addresses, bool value	Public
L	bool s, uint256 muchB	Public
distributeETHDividends	uint256 amount	External
excludeFromDividends	address account	External
updateClaimWait	uint256 newClaimWait	External
setBalance	address payable account, uint256 newBalance	External
processAccount	address payable account, bool	



automatic

External



Smart Contract Advance Checks



ID	Severity	Name	Result	Status
X-01	Low	Potential Sandwich Attacks.	Pass	Not-Found
X-02	Low	Function Visibility Optimization	Fail	Detected
X-03	High	Lack of Input Validation.	Fail	Pending
X-04	High	Centralized Risk In addLiquidity.	Pass	Not-Found
X-05	Low	Missing Event Emission.	Pass	Not-Found
X-06	Low	Conformance with Solidity Naming Conventions.	Pass	Not-Found
X-07	Low	State Variables could be Declared Constant.	Pass	Not-Found
X-08	Low	Dead Code Elimination.	Pass	Not-Found
X-09	High	Third Party Dependencies.	Pass	Not-Found
X-10	High	Initial Token Distribution.	Pass	Not-Found
X-11	High		Fail	Pending
X-12	High	Centralization Risks In The X Role	Pass	Not-Found
X-13	Informational	Extra Gas Cost For User..	Fail	Pending
X-14	Medium	Unnecessary Use Of SafeMath	Fail	Pending
X-15	Medium	Symbol Length Limitation due to Solidity Naming Standards.	Pass	Not-Found
X-16	Medium	Taxes can be up to 100%	Fail	Not-Found



ID	Severity	Name	Result	Status
X-17	Informational	Conformance to numeric notation best practice.	Pass	Not-Found
X-18	Informational	Stop Transactions by using Enable Trade.	Fail	Pending



X-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	 Low	X.sol: L: 623 C: 14, L: 631 C: 14, L: 648 C: 14, L: 657 C: 14, L: 664 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
setSwapTokensAtAmount	address newValue	public
updateDividendTracker	address newAddress	public
excludeFromFees	address account, bool excluded	public
updateUniswapV2Router	address newAddress	public
excludeMultipleAccountsFromFees	address[] calldata accounts, bool excluded	public
setAutomatedMarketMakerPair	address pair, bool value	public
setSwapAndLiquifyEnabled	bool status	public
setAirDropEnable	bool status	public
setTransferFree	bool s	public
bclistAddress	address account, bool value	public
multi_bclist	address[] calldata addresses, bool value	public



Function Name	Parameters	Visibility
L	bool s, uint256 muchB	public
distributeETHDividends	uint256 amount	public
updateClaimWait	uint256 claimWait	public
processAccount	address payable account, bool automatic	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.



X-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	 High	X.sol: L: 623 C: 14, L: 738 C: 14, L: 762 C: 14, L: 810 C: 14, L: 1472 C: 14	 Pending

Description

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

The given input is missing the check for the .

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. .



X-11 | .

Category	Severity	Location	Status
Optimization	 High	X.sol: L: 814 C: 17, L: 837 C: 14, L: 820 C: 14	 Pending

Description

We found an airdrop function inside the code where they can send tokens to other addresses.



Remediation

We recommend the team to remove these functions.

Project Action



X-13 | Extra Gas Cost For User.

Category	Severity	Location	Status
Logical Issue	 Informational	X.sol: L: 809, C: 17	 Pending

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



X-14 | Unnecessary Use Of SafeMath

Category	Severity	Location	Status
Logical Issue	 Medium	X.sol: L: 204 C: 11	 Pending

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.

Remediation



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

Solidity programming language

Project Action



X-16 | Taxes can be up to 100%.

Category	Severity	Location	Status
Logical Issue	 Medium	X.sol:	 Not-Found

Description

The current definition of taxes can be set up to 100% for specific wallets, we suggest to modify the function not to be dynamic but to be a static resolution.

```
function setSniperFee(address[] memory account, uint8 _sellFee, uint8 _buyFee) public
onlyOwner {
    for (uint256 i = 0; i < account.length; i++) {
        if (_sellFee > 0) {
            sellSniperFee[account[i]] = _sellFee;
        }
        if (_buyFee > 0) {
            buySniperFee[account[i]] = _buyFee;
        }
    }
}
```

due to the logic written in here may results in a honeypot.



Remediation

We advise the team to review the following logic..

Project Action



X-18 | Stop Transactions by using Enable Trade.

Category	Severity	Location	Status
Logical Issue	 Informational	X.sol: L: 837 C: 14	 Pending

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 an issue for the holders.

Remediation






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

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	3	0	0
 High	0	0	0
 Medium	1	0	0
 Low	1	0	0
 Informational	2	0	0
Total	7	0	0



Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/X_X_Crypto	Pass
Other		Fail
Website	https://x-coin.world/	Pass
Telegram	https://t.me/xvientiane	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	70/100
Auditor Score	60/100
Review by Section	Score
Manual Scan Score	26/53
SWC Scan Score	31/37
Advance Check Score	13 /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:

- the contract has several issues and is failed.
- Please DYOR on the project.

Auditor Score =60
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.

