CFG NINJA AUDITS

Security Assessment

CrypterToken Token

November 22, 2022

Table of Contents

Assessment Summary

This report has been prepared for CrypterToken Token on the Binance 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	2	2	0
Medium	1	1	0
Minor	2	2	0
Informational	2	2	0
Total	7	7	0

Project Overview

Token Summary

Parameter	Result
Address	
Name	CrypterToken
Token Tracker	CrypterToken (CRYPT)
Decimals	18
Supply	100,000,000,000
Platform	Binance
compiler	v0.6.12+commit.27d51765
Contract Name	CrypterToken
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/address/TBD
Payment Tx	0x688ec9dcd33239aed701f77518efba659fb09f3405ccdf1bbb918afe3887ec9f

Project Overview

Risk Analysis Summary

Parameter	Result
Buy Tax	6%
Sale Tax	6%
Is honeypot?	TBD
Can edit tax?	Yes
Is anti whale?	No
Is blacklisted?	No
Is whitelisted?	No
Holders	Clean
Confidence Level	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.

MainNet Contract was Not Assessed

TestNet Contract Assessed
Contract Name

Name	Contract	Live
CrypterToken	0x4dDb9dEe531562BEcAf338faB1a2c6dc84843576	Yes

Solidity Code Provided

SolID	File Sha-1	FileName
CrypterToken	130c36ee58fdc79f11d76826b30f678ac3f8ce7d	CrypterToken.sol

Mint Check

The project owners of CrypterToken have the ability to Mint New Tokens.

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

Mint Notes:

Auditor Notes: The Contract has a Mint and Burn, while Burn does not seem like a problem, Mint is defenitly something we should not not.

Project Owner Notes:

Fees Check

The project owners of CrypterToken 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 6% buy and 6% sale taxes, and cannot be set higher than 10%

Project Owner Notes:

Fees Can Be Changed up to a maximum of 25%

Blacklist Check

The project owners of CrypterToken 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 CrypterToken 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:

Pause Trade Check

The Project Owners of CrypterToken can stop or pause trading

We recommend the Team only allow Open Trade and never use Stop Trade, as this will be catastrophic for the Project and Investors.

We recommend the Team create a new contract without the stop trade function.

Pause Trade Notes:

Auditor Notes: There is an Open Trade and can be defined to false and block trades.

Project Owner Notes:

Owner can pause trading

Contract Ownership

The contract ownership of CrypterToken has been renounced.

Having no owner means that all the ownable functions in the contract can not be called by anyone, this often leads to more trust on the project.

Liquidity Ownership

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

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 Owners of CrypterToken is not KYC.

KYC Information Notes:

Auditor Notes:

Project Owner Notes:

Smart Contract Vulnerability Checks

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	CrypterToken.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	CrypterToken.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	CrypterToken.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	CrypterToken.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	CrypterToken.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	CrypterToken.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	CrypterToken.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	CrypterToken.sol	L: 0 C: 0
SWC-108	Low	State variable visibility is not set..	CrypterToken.sol	L: 858 C: 12
SWC-109	Pass	Uninitialized Storage Pointer.	CrypterToken.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	CrypterToken.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	CrypterToken.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	CrypterToken.sol	L: 0 C: 0

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

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

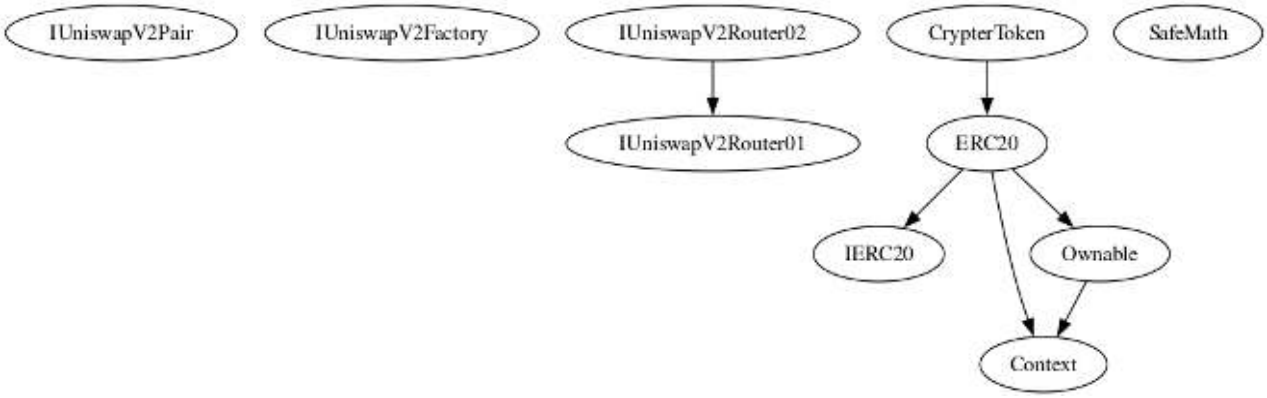
Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

References:

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

Inheritance

The contract for CrypterToken has the following inheritance structure.



Privileged Functions (onlyOwner)

Function Name	Parameters	Visibility
renounceOwnership		public
transferOwnership	newOwner (address)	public
openTrading	bOpen (bool)	public
setExcludeFromFee	_account(address) _bool(bool)	public
updateMinSwapAmount	_minSwapAmount (uint256)	public
enableSwapToken	_enabled(bool)	public
approveToRouter	_amount(uint256)	public
setTaxAddresses	_staking(address), _rewardWallet(address), _marketingWalle(address)	public
setTaxFee	uint8 _stakingFee, uint8 _teamFee, uint8 _marketingFee	public

Smart Contract Advance Checks

ID	Severity	Name	Result	Status
CRYPT-01	Minor	Potential Sandwich Attacks.	Pass	Not-Found
CRYPT-02	Informational	Function Visibility Optimization	Fail	In-Review
CRYPT-03	Minor	Lack of Input Validation.	Fail	In-Review
CRYPT-04	Major	Centralized Risk In addLiquidity.	Pass	Not Found
CRYPT-05	Major	Missing Event Emission.	Fail	In-Review
CRYPT-06	Minor	Conformance with Solidity Naming Conventions.	Pass	Not Found
CRYPT-07	Minor	State Variables could be Declared Constant.	Fail	In-Review
CRYPT-08	Major	Dead Code Elimination.	Pass	Not Found
CRYPT-09	Major	Third Party Dependencies.	Pass	Not Found
CRYPT-10	Major	Initial Token Distribution.	Fail	Resolved
CRYPT-11	Informational	excludeFromDividends can be called and is missing an Include Function to revert action.	Pass	Not Found
CRYPT-12	Major	Centralization Risks In The X Role	Pass	Not Found
CRYPT-13	Informational	Extra Gas Cost For User..	Fail	Not Found
CRYPT-14	Informational	Unnecessary Use Of SafeMath	Fail	Not Found

CRYPT-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	Informational	CrypterToken.sol: 686,21, 687,21	In-Review

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
burn		public
openTrading		public
setExcludeFromFee		public
updateMinSwapAmount		public
enableSwapToken		public
approveToRouter		public
setTaxAddresses		public
setTaxFee		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.

CRYPT-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	Minor	CrypterToken.sol: 1031,14	In-Review

Description

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

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");
...
```

CRYPT-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	Major	CrypterToken.sol: 230,14	In-Review

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.

CRYPT-07 | State Variables could be Declared Constant.

Category	Severity	Location	Status
Coding Style	Minor	CrypterToken.sol: 872,20	In-Review

Description

Constant state variables should be declared constant to save gas.

STABLE

Remediation

Add the constant attribute to state variables that never changes.

https://docs.soliditylang.org/en/latest/contracts.html#constant-state-variables

CRYPT-10 | Initial Token Distribution.

Category	Severity	Location	Status
Centralization / Privilege	Major	CrypterToken.sol: 755,15	Resolved

Description

All of the CrypterToken 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

Partial Token Distribution goes to the msg.sender, and rest to Owner.
89.5000%/10.5000%

CRYPT-13 | Extra Gas Cost For User.

Category	Severity	Location	Status
Logical Issue	Informational	CrypterToken.sol: 938, 13	Not Found

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

CRYPT-14 | Unnecessary Use Of SafeMath

Category	Severity	Location	Status
Logical Issue	Medium	CrypterToken.sol: 370, 9	In-Review

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.
_balances[recipient] = _balances[recipient].add(amount);
magnifiedDividendPerShare = magnifiedDividendPerShare.add(
(amount).mul(magnitude) / totalSupply()
);
Note: Only a sample of 2 SafeMath library usage in this contract (out of 14) are shown above.

Remediation

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

Project Action

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/CrypterOfficial	Pass
Other		Fail
Website	http://www.crypter.io	Pass
Telegram	https://t.me/cryptermain	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	66/100
Auditor Score	50/100
Review by Section	**Score**
Manual Scan Score	16/35
SWC Scan Score	36 /37
Advance Check Score	14 /28

The Following Score System Has been Added to this page to help understand the value of the audit, the maximun score is 100, however to attain that value the project most 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.

Assessment Results

Important Notes:

- The following contract is using an older compiler with vulnerabilities in it 0.6.12, the latest compiler is 0.8.17.
- Read about compiler vulnerabilities in the following link https://docs.soliditylang.org/en/v0.6.12/bugs.html.
- Code has mint without limit, this can negatively impact the investors. we recommend the team fix those before releasing the same.
- Several problems can be found when performing functions due to a lack of validations, this can impact the transfer negatively.

Audit Fail

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.