CFG NINJA AUDITS

Security Assessment

Original Wassie Token

July 15, 2023

Audit Status: Fail

Audit Edition: Advanced

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	0%
Sale Tax	0%
Cannot Sale	Pass
Cannot Sale	Pass
Max Tax	0%
Modify Tax	No
Fee Check	Pass
Is Honeypot?	Detected
Trading Cooldown	Not Detected
Can Pause Trade?	Detected.

Contract Priviledge	Description
Pause Transfer?	Detected
Max Tx?	Fail
Is Anti Whale?	Detected
Is Anti Bot?	Not Detected
Is Blacklist?	Not Detected
Blacklist Check	Pass
is Whitelist?	Detected
Can Mint?	Pass
Is Proxy?	Not Detected
Can Take Ownership?	Not Detected
Hidden Owner?	Not Detected
Owner	0x1882A6548bE39B4b9b23a8315BBfF63887E06e43
Self Destruct?	Not Detected
External Call?	Not Detected
Other?	Not Detected
Holders	1
Auditor Confidence	None

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	0x15d5D13fd59A62EF8847332203011e6628cf1Cd5
Name	Original Wassie
Token Tracker	Original Wassie (WASSIE)
Decimals	18
Supply	690,000,000,000
Platform	Ethereum
compiler	v0.8.4+commit.c7e474f2
Contract Name	OriginalWassie
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/token/0x15d5d13fd59a62ef8847332203011e6628cf1cd5#code
Payment Tx	Corporate

Main Contract Assessed Contract Name

Name	Contract	Live
Original Wassie	0x15d5D13fd59A62EF8847332203011e6628cf1Cd5	Yes

TestNet Contract Assessed Contract Name

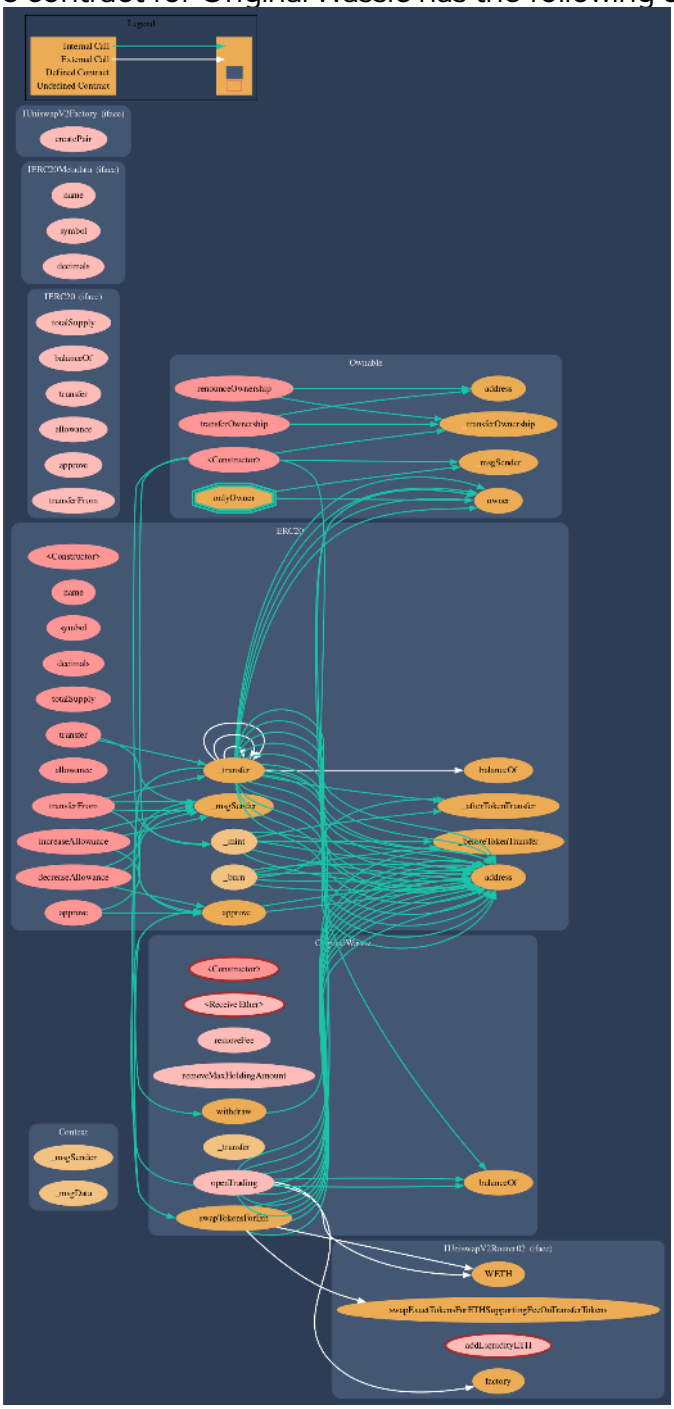
Name	Contract	Live
Original Wassie	0xe7007B9598E137B51861F99b6b10519a2161261F	Yes

Solidity Code Provided

SolID	File Sha-1	FileName
Wassie	8a9434ebab938bd8a9d4f3b0b6d64a1986c29e7a	WASSIE.sol
Wassie		
Wassie		
Wassie		

Call Graph

The contract for Original Wassie 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	WASSIE.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	WASSIE.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	WASSIE.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	WASSIE.sol	L: 7 C: 0
SWC-104	Pass	Unchecked Call Return Value.	WASSIE.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	WASSIE.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	WASSIE.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	WASSIE.sol	L: 0 C: 0
SWC-108	Low	State variable visibility is not set..	WASSIE.sol	L: 625 C: 33
SWC-109	Pass	Uninitialized Storage Pointer.	WASSIE.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	WASSIE.sol	L: 0 C: 0

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

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

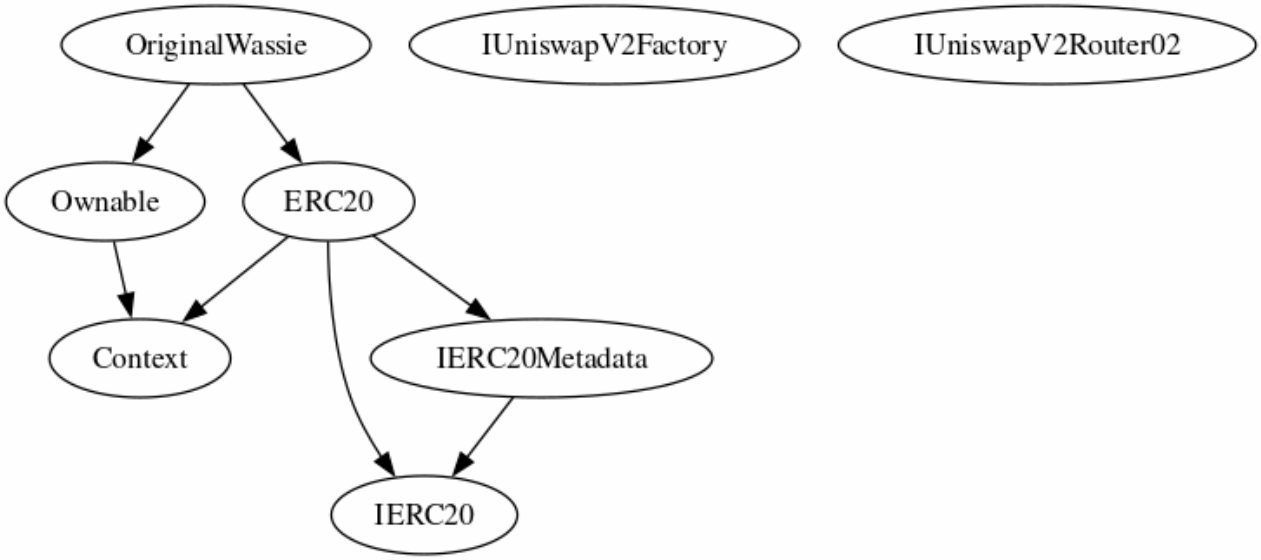
Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

References:

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

Inheritance

The contract for Original Wassie has the following inheritance structure.



Smart Contract Advance Checks

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

ID	Severity	Name	Result	Status
WASSIE-16	Medium	Taxes can be up to 100%	Pass	Not Detected
WASSIE-17	Logical Issue	Highly Permissive Role Access.,`	Pass	Detected
WASSIE-18	Critical	Stop Transactions by using Enable Trade.	Fail	Detected

WASSIE-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	Informational	WASSIE.sol: L: 0 C: 0	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
router		internal
SUPPLY		internal
fee		internal

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.

WASSIE-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	Low	WASSIE.sol: L: 639 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 allOnly Owners.

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. allOnly Owners.

WASSIE-04 | Centralized Risk In addLiquidity.

Category	Severity	Location	Status
Coding Style	High	WASSIE.sol: L:657, C: 44	Detected

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 WASSIE-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

Project Action

WASSIE-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	Low	WASSIE.sol: L: 639 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.

WASSIE-13 | Extra Gas Cost For User.

Category	Severity	Location	Status
Logical Issue	Informational	WASSIE.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

WASSIE-18 | Stop Transactions by using Enable Trade.

Category	Severity	Location	Status
Logical Issue	Critical	WASSIE.sol: L: 716 C: 0	Detected

Description

Enable Trade is presend 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 and 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	1	0	0
High	1	0	0
Medium	0	0	0
Low	2	0	0
Informational	2	0	0
Total	6	0	0

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/OriginalWassie	Pass
Other		Fail
Website	https://originalwassie.com	Pass
Telegram	https://t.me/originalwassie	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	72/100
Auditor Score	50/100
Review by Section	**Score**
Manual Scan Score	13/33
SWC Scan Score	33 /37
Advance Check Score	26 /30

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.

Audit Fail

Assessment Results

Important Notes:

- One vulnerability was found that needs to be addressed.
- Blacklist Present, Enable Trade Present and taxes to 100%.
- Please DYOR on the project.

Auditor Score =50
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.