CFG NINJA AUDITS

Security Assessment

Okami DAO Token

March 15, 2023

Audit Status: Pass

Audit Edition: Advance

Table of Contents

Assessment Summary

This report has been prepared for Okami DAO Token on the Ethereum 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.

Project Overview

Token Summary

Parameter	Result
Address	0x37a3bb95891932622F6aA3cb8072eff3C2c93636
Name	Okami DAO
Token Tracker	Okami DAO (OKDAO)
Decimals	18
Supply	10,000,000
Platform	Ethereum
compiler	v0.8.17+commit.8df45f5f
Contract Name	OKDAO
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/token/0x37a3bb95891932622f6aa3cb8072eff3c2c93636#code
Payment Tx	Corporate

Project Overview

Risk Analysis Summary

Parameter	Result
Buy Tax	0%
Sale Tax	0%
Is honeypot?	Clean
Can edit tax?	Yes
Is anti whale?	Yes
Is blacklisted?	No
Is whitelisted?	Yes
Holders	9
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.

Project Overview

Simulation Summary

Parameter	Result
Transfer From Owner	Pass
Transfer From Holder	Pass
Add Liquidity	Pass
Buy from Owner	Pass
Buy from Holder	Pass
Remove Liquidity	Pass
SwapAndLiquify	Pass
RemoveLiquidity	Pass
LaunchPad	PinkSale

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
Okami DAO	0x37a3bb95891932622F6aA3cb8072eff3C2c93636	Yes

TestNet Contract Assessed
Contract Name

Name	Contract	Live
Okami DAO	0xB55494776769A28552EE9CAF790BDBe462cdFA5C	Yes

Solidity Code Provided

SolID	File Sha-1	FileName
okami	78cdb083fdd8ba6dacd2eccb8408dac3896b6882	okami.sol
okami		
okami		

Mint Check

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

The Project has a Total Supply of 10,000,000 and cannot mint any more than the Max Supply.

Mint Notes:

Auditor Notes:

Project Owner Notes:

Owner can't mint new coins

Fees Check

The project owners of Okami DAO have the ability to set higher than 25%
We Recommend the team to create a review contract and set it with fees restrictions to avoid any problems, as alternative the team can use multi signature wallet to ensure the project is safe from a potential fee increase.

Tax Fee Notes:

Auditor Notes: The contract currently has 0% buy and 0% sale taxes and total fees cannot be higher than 30% for buy, however sale tax can be as high as 40%.

Project Owner Notes:

Blacklist Check

The project owners of Okami DAO 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: There is not a blacklist present, however there is a whitelist for people to buy or trade before is open to public.

Project Owner Notes: undefined

MaxTx Check

The Project Owners of Okami DAO can set max tx amount.

The ability to set MaxTx can be used as bad actor, this can limit the ability of investors to sale their tokens at any given time if is set too low..

We recommend the project to set MaxTx to Total Supply or simiar to avoid swap or transfer from failures

MaxTX Notes:

Auditor Notes: uint256 public _maxTxAmount = _tTotal.div(100).mul(1); //1% - max tx limit on sale only uint256 public _walletHoldingMaxLimit = _tTotal.div(100).mul(1); // 1% - wallet holding max limit

Project Owner Notes:

Pause Trade Check

The Project Owners of Okami DAO 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: There is not a stop trade, however there isa open trade.

Project Owner Notes:

Contract Ownership

The contract ownership of Okami DAO 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 0x44FC6C7F78da42B65461f2385E59F7b02806E045 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

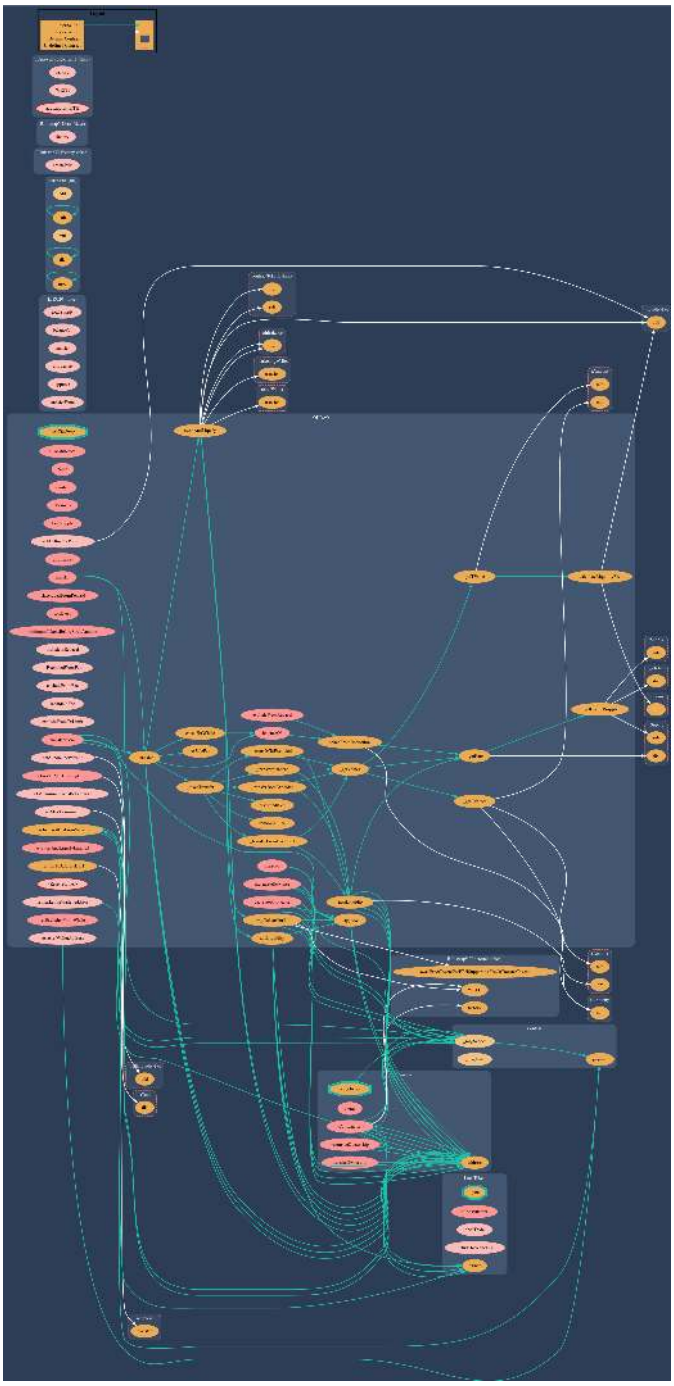
The token does not have liquidity at the moment of the audit, block 16776467

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 Okami DAO has the following call graph structure.



KYC Information

The Project Owners of Okami DAO have provided KYC Documentation.

KYC Certificated can be found on the Following:
KYC Data

KYC Information Notes:

Auditor Notes:

Project Owner Notes:

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	okami.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	okami.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	okami.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	okami.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	okami.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	okami.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	okami.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	okami.sol	L: 0 C: 0
SWC-108	Low	State variable visibility is not set..	okami.sol	L: 226 C: 12, L: 227 C: 12, L: 228 C: 12,L: 240 C: 9
SWC-109	Pass	Uninitialized Storage Pointer.	okami.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-110	Pass	Assert Violation.	okami.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	okami.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	okami.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	okami.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	okami.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	okami.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	okami.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	okami.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	okami.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	okami.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	okami.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	okami.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	okami.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	okami.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	okami.sol	L: 0 C: 0

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

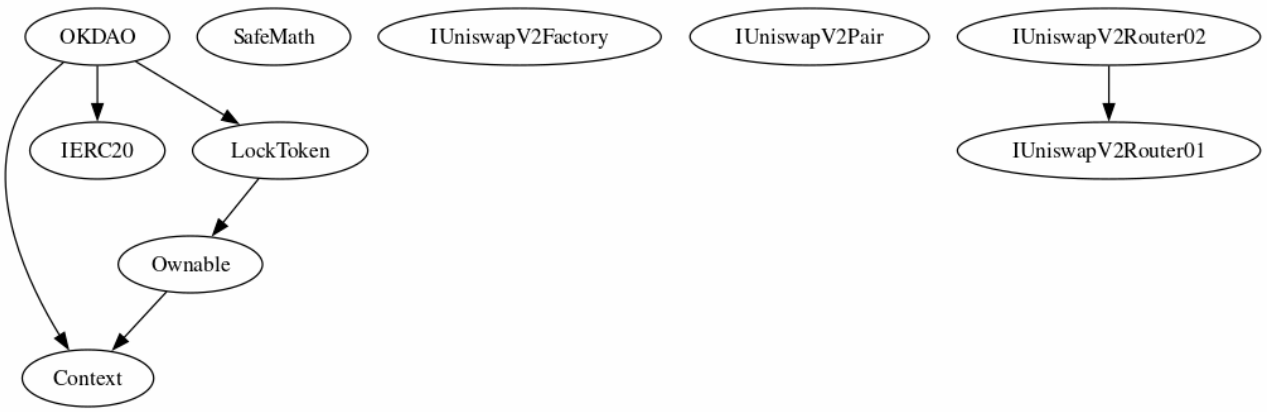
Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

References:

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

Inheritance

The contract for Okami DAO 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	newOwner (address)	public
excludeFromReward	address account	public
includeInReward	address account	external
excludeFromFee	address account	external
includeInFee	address account	external
excludeFromTxLimit	address account, bool _value	external
setAllBuyFeePercent	uint256 liquidityFee, uint256 marketingFee, uint256 reliefFee	external
setAllSaleFeePercent	uint256 liquidityFee, uint256 marketingFee, uint256 reliefFee	external

setMaxTxAmount

uint256 _mount

external

setMinimumTokensB
eforeSwap

uint256 _minimumTokensBeforeSwap

		external
setmarketingWalletAddress	address _marketingWallet	external
setreliefWalletAddress	address _reliefWallet	external
setSwapAndLiquifyEnabled	bool _enabled	public
setExcludedFromWhale	address account, bool _enabled	public
setWalletMaxHoldingLimit	uint256 _amount	public

Smart Contract Advance Checks

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

ID	Severity	Name	Result	Status
OKDAO-16	Medium	Invalid collection of Taxes during Transfer.	Pass	Not-Found

OKDAO-01 | Potential Sandwich Attacks.

Category	Severity	Location	Status
Security	Minor	okami.sol: 466,13	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?.

OKDAO-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	Minor	okami.sol: 240,9	Pending

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
inSwapAndLiquify		internal
_sellLiquidityFee		internal
_sellMarketingFee		internal
_sellReliefFee		internal
setWalletMaxHoldingLimit		public
setExcludedFromWhale		public
excludeFromReward		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.

OKDAO-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	Minor	okami.sol: 380, 13	Pending

Description

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

The given input is missing the check for the excludeFromFee,includeInFee, excludeFromTxLimit,setMinimumTokensBeforeSwap, setmarketingWalletAddress, missing required function.

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. excludeFromFee,includeInFee, excludeFromTxLimit,setMinimumTokensBeforeSwap, setmarketingWalletAddress, missing required function.

OKDAO-04 | Centralized Risk In addLiquidity.

Category	Severity	Location	Status
Coding Style	Major	okami.sol: 486,13	Pending

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 OKDAO-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

.

OKDAO-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	Major	okami.sol: 478, 14	Pending

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.

OKDAO-06 | Conformance with Solidity Naming Conventions.

Category	Severity	Location	Status
Coding Style	Minor	okami.sol: 294,14	Not-Found

Description

Solidity defines a naming convention that should be followed. Rule exceptions: Allow constant variable name/symbol/decimals to be lowercase. Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
setreliefWalletAddres
setmarketingWalletAddress
```

Remediation

Follow the Solidity naming convention.

https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-convention

OKDAO-13 | Extra Gas Cost For User.

Category	Severity	Location	Status
Logical Issue	Informational	okami.sol: 236, 8	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

OKDAO-14 | Unnecessary Use Of SafeMath

Category	Severity	Location	Status
Logical Issue	Medium	okami.sol: 20, 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.
_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

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	3	0	0
Medium	1	0	0
Minor	2	0	0
Informational	2	0	0
Total	8	0	0

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/Okami_DAO_ETH	Pass
Other	https://www.linkedin.com/feed/update/urn:li:activity:7040363584945590272/	Pass
Website	https://okamidao.com	Pass
Telegram	https://t.me/OkamiDAO	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	80/100
Auditor Score	80/100
Review by Section	**Score**
Manual Scan Score	42/50
SWC Scan Score	36 /37
Advance Check Score	2 /16

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 Passed

Assessment Results

Important Notes:

- No vulnerabilities or issues found.
- By default code has a hardcoded tax of 7%, currently on mainnet is set to 0. this can change on demand based on ownership.
- There are a few items that need to be addressed on the code, however since the customer has KYC not sure are needed to be updated.
- Always DYOR on any project.

Auditor Score =80
Audit Passed

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.