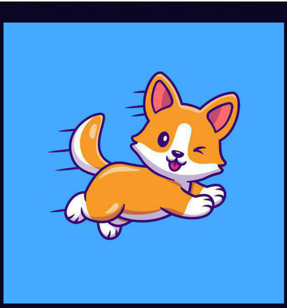
ASSURE DEFI®
THE VERIFICATION GOLD STANDARD

Security Assessment:
Hina Inu Token

April 7, 2024

- Audit Status: **Fail**
- Audit Edition: **Standard**



VERIFIED BY ASSURE DEFI®
INTEGRITY • TRUST • CREDIBILITY

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.
ℹ Informational	Function Detected

Manual Code Review Risk Results

Contract Privilege	Description
🟢 Buy Tax	2%
🟢 Sale Tax	2%
🟢 Cannot Buy	Pass
🟢 Cannot Sale	Pass
🔴 Max Tax	100%
🔴 Modify Tax	Yes
🔴 Fee Check	Fail
🔴 Is Honeypot?	Detected
🟢 Trading Cooldown	Not Detected
🟢 Can Pause Trade?	Pass
🟢 Pause Transfer?	Not Detected
🟢 Max Tx?	Pass
🟢 Is Anti Whale?	Not Detected
🟠 Is Anti Bot?	Detected

Contract Privilege	Description
🟢 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
ℹ Owner	0x15663A178808E99DEa937f958906f3a020E4f0D2
🟢 Self Destruct?	Not Detected
🟢 External Call?	Not Detected
🟢 Other?	Not Detected
🟢 Holders	10
🟠 Auditor Confidence	Medium-High Risk
🟡 KYC Present	No
🟡 KYC URL	

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	0x338b050D138529CD6d76AE2702fFcB02490dd828
Name	Hina Inu
Token Tracker	Hina Inu ($HINA)
Decimals	18
Supply	2,000,000,000
Platform	BASE
compiler	v0.8.20+commit.a1b79de6
Contract Name	HinaInu
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://basescan.org/address/0x338b050D138529CD6d76AE2702fFcB02490dd828#code
Payment Tx	Corporate

Main Contract Assessed
Contract Name

Name	Contract	Live
Hina Inu	0x338b050D138529CD6d76AE2702fFcB02490dd828	Yes

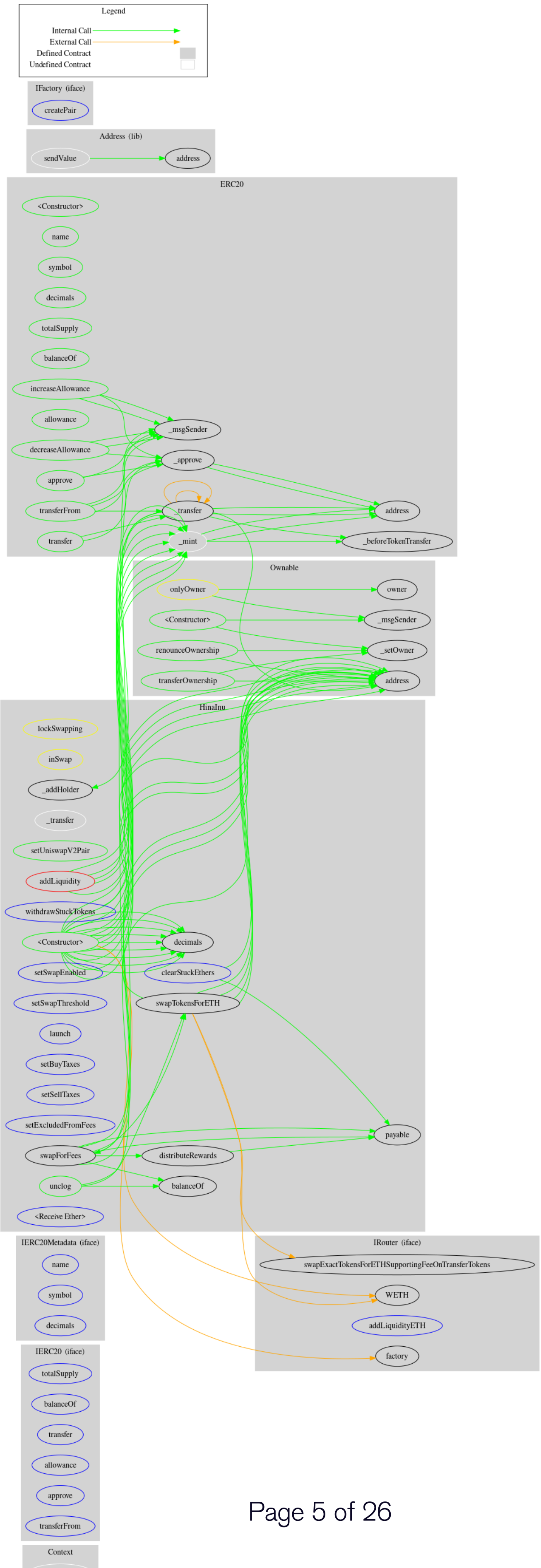
TestNet Contract was Not Assessed

Solidity Code Provided

SolID	File Sha-1	FileName
hinainu	af8b10a2951f865c991e5a9f3f5498d2a52afde4	hinainu.sol
hinainu		
hinainu		
hinainu		
hinainu		
hinainu	undefined	

Call Graph

The contract for Hina Inu has the following call graph structure.



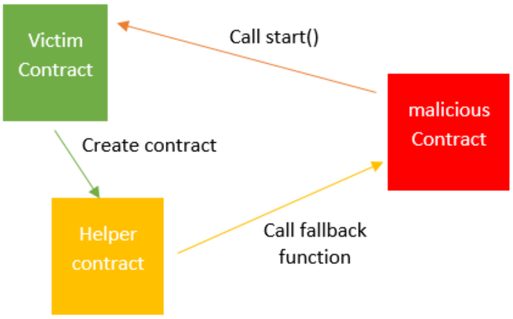
Reentrancy Check

The Project Owners of Hina Inu have not configure the Reentrancy Guard library.

You can read more about Reentrancy issues in the following link.
Reentrancy After Istanbul.

We recommend the team to add the library to the contract to avoid potential issues.

We recommend the team to create a new contract with Reentrancy Guard added to the same.



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	hinainu.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	hinainu.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	hinainu.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	hinainu.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	hinainu.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	hinainu.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	hinainu.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	hinainu.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	hinainu.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	hinainu.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	hinainu.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	hinainu.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	hinainu.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	hinainu.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	hinainu.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-115	Pass	Authorization through tx.origin.	hinainu.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	hinainu.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	hinainu.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	hinainu.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	hinainu.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	hinainu.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	hinainu.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	hinainu.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	hinainu.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	hinainu.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	hinainu.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	hinainu.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	hinainu.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	hinainu.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	hinainu.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	hinainu.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	hinainu.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	hinainu.sol	L: 0 C: 0

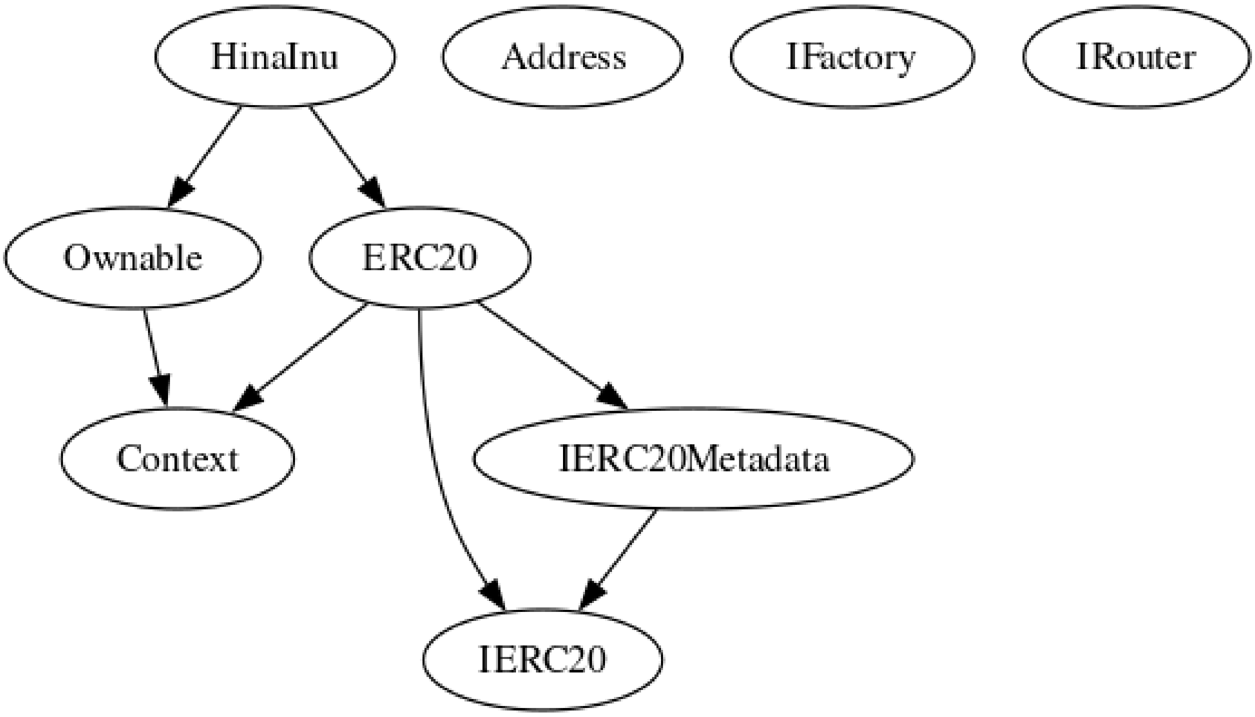
ID	Severity	Name	File	location
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	hinainu.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	hinainu.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	hinainu.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	hinainu.sol	L: 0 C: 0

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

Inheritance

The contract for Hina Inu has the following inheritance structure.

The Project has a Total Supply of 2,000,000,000



Privileged Functions (onlyOwner)

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

Function Name	Parameters	Visibility
transferOwnership	address newOwner	Public
renounceOwnership		Public
setUniswapV2Pair	address _pair	Public
setSwapEnabled	bool state	External
setSwapThreshold	uint256 new_amount	External
launch		External
setBuyTaxes	uint256 _tax	External
setSellTaxes	uint256 _tax	External
setExcludedFromFees	address _address, bool state	External
withdrawStuckTokens	address _token, address _to	External
clearStuckEthers	uint256 amountPercentage	External
unclog		Public
distributeRewards	uint256 rewardAmount	Private

$HINA-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	Informational	hinainu.sol: L: 593 C: 14, L: 264 C: 14, L: 456 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
transferOwnership,address newOwner,Public	setUniswapV2Pair,address _pair,Public	unclog,,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.

$HINA-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	🟢 Low	hinainu.sol: L: 456 C: 14, L: 489 C: 14, L: 532 C: 14, L: 532-593 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 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. missing required function.

$HINA-11 | Withdrawal Functions Potential for Exploitation.

Category	Severity	Location	Status
Optimization	● Low	hinainu.sol: L: 577 C: 14	Detected

Description

Functions like withdrawStuckTokens and clearStuckEthers could be exploited if proper checks and balances are not in place.

Remediation

Add safeguards and limit the scope of these functions to prevent misuse.

Project Action

$HINA-16 | Taxes can be up to 100%.

Category	Severity	Location	Status
Logical Issue	🔴 Critical	hinainu.sol: L: 559-564 C: 14	Detected

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.

```
feeInTokens > senderBalance &&
(feeInTokens / 100) * 95 <= senderBalance
```

due to the logic written in here may results in loss of funds.

Remediation

We advise the team to review the following logic function

```
function setFee(uint256 redisFeeOnBuy, uint256 redisFeeOnSell, uint256 taxFeeOnBuy, uint256 taxFeeOnSell) public onlyOwner {
    _redisFeeOnBuy = redisFeeOnBuy;
    _redisFeeOnSell = redisFeeOnSell;
    _taxFeeOnBuy = taxFeeOnBuy;
    _taxFeeOnSell = taxFeeOnSell;
}
```

Project Action

$HINA-18 | Stop Transactions by using Enable Trade.

Category	Severity	Location	Status
Logical Issue	🔴 Critical	hinainu.sol: L: 551 C: 14	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

$HINA-19 | Potential Reentrancy in swapTokensForETH.

Category	Severity	Location	Status
Logical	● Medium	hinainu.sol: L: 500 C: 14	Detected

Description

The swapTokensForETH function interacts with an external contract and could be susceptible to reentrancy attacks if the external call is not properly handled.

Remediation

Use the Checks-Effects-Interactions pattern and consider adding a reentrancy guard for functions interacting with external contracts.

Project Action

$HINA-20 | Unbounded Loops in distributeRewards.

Category	Severity	Location	Status
Logical	🔴 Critical	hinainu.sol: L: 489 C: 14	Detected

Description

The distributeRewards function contains a loop that iterates over the entire _holders array, which could lead to out-of-gas errors if the array grows too large, making the function fail and potentially locking funds.

Remediation

Implement a mechanism to limit the number of iterations per transaction or use a more gas-efficient distribution method.

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.
ℹ 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	3	0
🟠 High	0	0	0
🟡 Medium	2	2	0
🟢 Low	1	1	0
ℹ Informational	1	1	0
Total	7	7	0

Social Media Checks

Social Media	URL	Result
Twitter	https://x.com/realhinainu	Pass
Other	no	N/A
Website	https://hinainu.com	Pass
Telegram	https://t.me/hinainubaseportal	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	35/100
Auditor Score	40/100
Review by Section	Score
Manual Scan Score	0
SWC Scan Score	37
Advance Check Score	-2

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 most pass and provide all the data needed for the assessment. Our Passing Score has been changed to 84 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below.

Audit Fail

Assessment Results

Important Notes:

- The contract implements a tax system with different rates for buy and sell transactionsı
- totBuyTax: A 2% tax on buy transactions.ı
- totSellTax: A 2% tax on sell transactions.ı
- Additionally, there are higher taxes for the first 20 minutes after launch, decreasing over timeı
- First 5 minutes: 50% taxı
- 5-10 minutes: 40% taxı
- 10-15 minutes: 30% taxı
- 15-20 minutes: 20% taxı
- After 20 minutes: 2% tax (standard rate)ı
- These taxes are taken from transactions and can be used for various purposes like rewards distribution or project funding.ı
- Reentrancy swapTokensForETH could be susceptible to reentrancy attacks.ı
- Centralization risks Owner has too much control (e.g., launch, setSwapEnabled).ı
- Arbitrary minting: _mint in constructor could be a risk if not audited properly.ı
- Unbounded loops: distributeRewards uses a loop, potential

for gas limit issues.ı

- Use of tx.origin: Not present, which is good as it's a common security risk.ı
- Floating pragma: Fixed to 0.8.20, which is good practice.ı
- Lack of input validation: Some functions lack input validation beyond zero address checks.ı
- External contract dependencies: Relies on external IRouter and IFactory, potential risks if these are compromised.ı
- Hardcoded addresses: Could indicate centralization or special privileges.ı
- Withdrawal functions: withdrawStuckTokens and clearStuckEthers could be exploited if not handled properly.ı
- Lack of circuit breaker: No mechanism to pause contract in case of an attack.ı
- No safeMath needed: Solidity 0.8.x has overflow checks built-in.ı
- transfer function: Custom logic could introduce bugs or unexpected behavior.ı
- distributeRewards onlyOwner: Ensures only owner can call, but centralizes power.ı
- lockSwapping modifier: Prevents reentrancy for unclog, but not used universally.ı
- In conclusion, this contract has several areas of concern, particularly around centralization of control, potential for reentrancy, and unbounded loops that could lead to

performance issues. It's crucial to address these risks to ensure the security and reliability of the contract.ı

- Score: 40/100ı
- The score reflects the contract's centralization risks, potential for reentrancy, and other noted issues. It's recommended to address these concerns to improve the security score.ı

Auditor Score =40
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

Assure Defi 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 Assure Defi is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will Assure Defi 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 Assure Defi 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.

ASSURE DEFI™
THE VERIFICATION GOLD STANDARD