ASSURE DEFI®
THE VERIFICATION GOLD STANDARD

Security Assessment:

Hina Inu Token

April 11, 2024

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

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 Privilege	Description
Buy Tax	2%
Sale Tax	2%
Cannot Buy	Pass
Cannot Sale	Pass
Max Tax	50%
Modify Tax	No
Fee Check	Fail
Is Honeypot?	Not 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
Owner	0x66810c591bdE95399450Fe1fb77B45991A7fb991
Self Destruct?	Not Detected
External Call?	Not Detected
Other?	Not Detected
Holders	9
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	0xA66896dBce8bebD8ECAe69e098B74f4aa6274fa2
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/0xA66896dBce8bebD8ECAe69e098B74f4aa6274fa2#code
Payment Tx	Corporate

Main Contract Assessed
Contract Name

Name	Contract	Live
Hina Inu	0xA66896dBce8bebD8ECAe69e098B74f4aa6274fa2	Yes

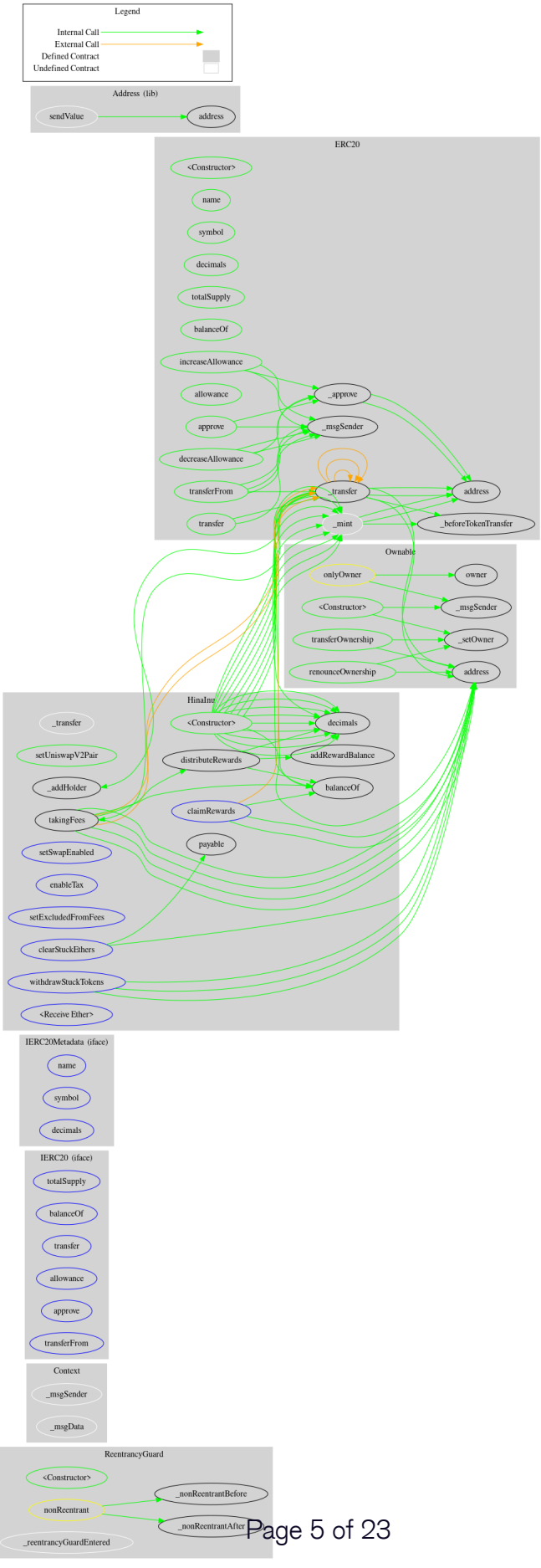
TestNet Contract was Not Assessed

Solidity Code Provided

SolID	File Sha-1	FileName
Hina2	af8b10a2951f865c991e5a9f3f5498d2a52afde4	hinainu.sol
Hina2		
Hina2		
Hina2		
Hina2		
Hina2	undefined	

Call Graph

The contract for Hina Inu 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	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

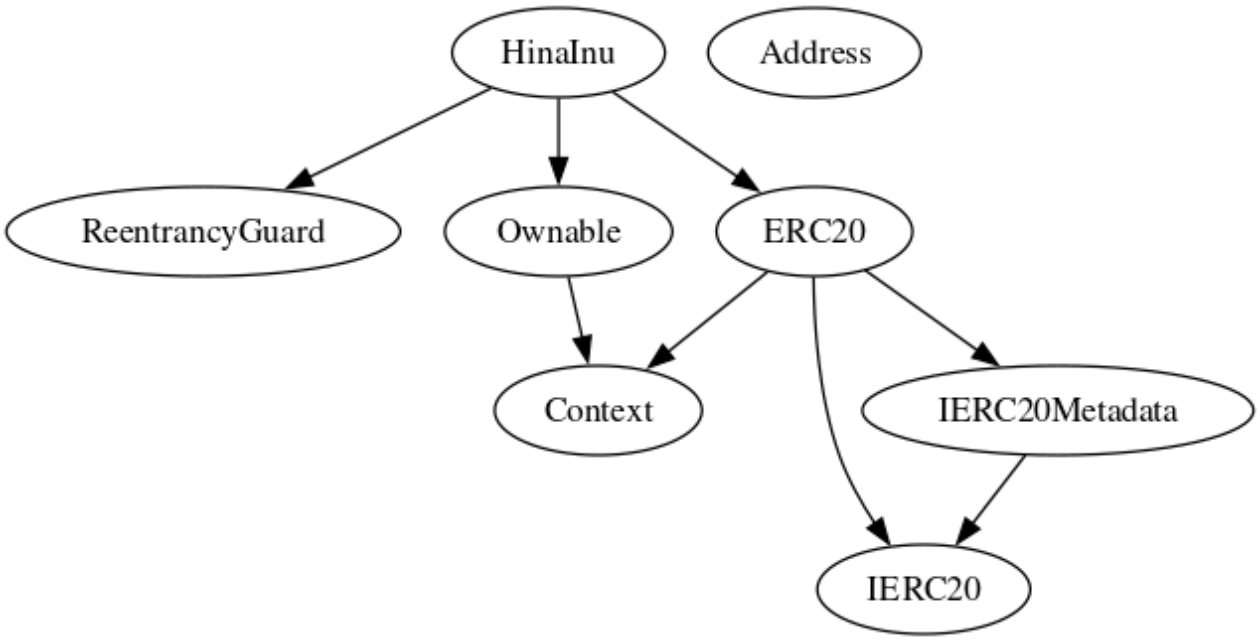
ID	Severity	Name	File	location
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	hinainu.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	hinainu.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	hinainu.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted 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
setUniswapV2Pair	address _pair	Public
setSwapEnabled	bool state	External
enableTax		External
setExcludedFromFees	address _address, bool state	External
withdrawStuckTokens	address _token, address _to, uint256 _amount	External
clearStuckEthers	uint256 amountPercentage	External
renounceOwnership		Public
transferOwnership	address newOwner	Public

$HINA-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	Low	hinainu.sol: L: 548 C: 14, L: 534 C: 14, L: 474 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-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	Low	hinainu.sol: L: 474 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.

$HINA-11 | Withdrawal Functions Potential for Exploitation.

Category	Severity	Location	Status
Optimization	Low	hinainu.sol: L: 556-573 C: 14	Detected

Description

The withdrawStuckTokens and clearStuckEthers functions allow the owner to withdraw assets without restrictions.

Remediation

Add safeguards such as time locks or multi-sig requirements.

Project Action

$HINA-20 | Unbounded Loops in distributeRewards.

Category	Severity	Location	Status
Logical	Critical	hinainu.sol: L: 506 C: 14	Partially Resolved

Description

The function uses a loop that could hit gas limits with many holders.

Remediation

Implement a capped loop or a more gas-efficient reward distribution method.

Project Action

$HINA-21 | Withdraw Own Token Balance.

Category	Severity	Location	Status
Logical	High	hinainu.sol: L: 556 C: 14	Detected

Description

The contract allows the owner to withdraw the contract's own tokens.

Remediation

Prevent the withdrawal of the contract's own tokens to ensure intended functionality

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	1	2
High	1	1	0
Medium	0	0	2
Low	3	2	0
Informational	0	0	1
Total	5	4	4

Social Media Checks

Social Media	URL	Result
Twitter	https://x.com/realhinainu	Pass
Other		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	64/100
Auditor Score	75/100
Review by Section	**Score**
Manual Scan Score	1
SWC Scan Score	37
Advance Check Score	26

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:

- Tax System: High initial taxes reduced, with a maximum of 50% in the first 5 minutes post-launch, decreasing over time to a standard 2% tax. (Resolved)ı
- Reentrancy Guard: Implemented OpenZeppelin's ReentrancyGuard to mitigate reentrancy attacks. (Resolved)ı
- Centralization Risks: Owner retains significant control over key functions, presenting a centralization risk. (Unresolved)ı
- Arbitrary Minting: Minting occurs only in the constructor, posing no risk after deployment. (Resolved)ı
- Unbounded Loops: distributeRewards still uses an unbounded loop, which could lead to gas limit issues. (Unresolved)ı
- Floating Pragma: Pragma is fixed at 0.8.20, which is good practice. (Resolved)ı
- Lack of Input Validation: Some functions lack comprehensive input validation beyond zero address checks. (Unresolved)ı
- External Contract Dependencies: No evidence of reliance on external IRouter and IFactory in the provided snippet. (Potentially Resolved)ı
- Hardcoded Addresses: Hardcoded addresses in the constructor suggest centralization or special privileges. (Unresolved)ı

• Withdrawal Functions: Withdrawal functions for stuck tokens and Ether still present a risk if not properly managed. (Unresolved)ı

• Lack of Circuit Breaker: No mechanism to pause or stop transactions in case of an emergency. (Unresolved)ı

• SafeMath: Not needed due to Solidity 0.8.x overflow checks. (Resolved)ı

• Custom Transfer Logic: Custom logic in the transfer function could introduce unexpected behavior. (Unresolved)ı

• distributeRewards onlyOwner: Centralizes power to the owner. (Unresolved)ı

• Withdraw Own Token Balance: The contract allows the owner to withdraw the contract's own tokens, which could be exploited. (Unresolved)ı

• Overall Risk: Medium-Highı

• Conclusion: The contract has made some improvements, particularly in mitigating reentrancy attacks and setting a fixed pragma. However, significant risks remain due to centralization, potential for gas limit issues with unbounded loops, lack of comprehensive input validation, and the ability for the owner to withdraw the contract's own tokens. These issues should be addressed to improve the security and trustworthiness of the contract. The score reflects the contract's current state with both resolved and unresolved issues, indicating that while progress has been made, there is still room for improvement.

Auditor Score =75

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.