Inuyasha

Smart Contract Review

Deliverable: Smart Contract Audit Report

Security Report

October 2021

Disclaimer

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the Company. The content, conclusions and recommendations set out in this publication are elaborated in the specific for only project.

eNebula Solutions does not guarantee the authenticity of the project or organization or team of members that is connected/owner behind the project or nor accuracy of the data included in this study. All representations, warranties, undertakings and guarantees relating to the report are excluded, particularly concerning – but not limited to – the qualities of the assessed projects and products. Neither the Company nor any personating on the Company's behalf may be held responsible for the use that may be made of the information contained herein.

eNebula Solutions retains the right to display audit reports and other content elements as examples of their work in their portfolio and as content features in other projects with protecting all security purpose of customer. The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities fixed - upon a decision of the Customer.

Report Summary

Title	**Inuyasha Smart Contract Audit**		
Project Owner	Inuyasha		
Type	Public		
Reviewed by	Vatsal Raychura	**Revision date**	18/10/2021
Approved by	eNebula Solutions Private Limited	**Approval date**	18/10/2021
		Nº Pages	**33**

Overview

Background

Inuyasha's team requested that eNebula Solutions perform an Extensive Smart Contract audit.

Project Dates

The following is the project schedule for this review and report:

- **October 18**: Smart Contract Review Completed *(Completed)*
- **October 18**: Delivery of Smart Contract Audit Report *(Completed)*

Review Team

The following eNebula Solutions team member participated in this review:

- Sejal Barad, Security Researcher and Engineer
- Vatsal Raychura, Security Researcher and Engineer

Coverage

Target Specification and Revision

For this audit, we performed research, investigation, and review of the smart contract of Inuyasha.

The following documentation repositories were considered in-scope for the review:

- Inuyasha Project: https://ropsten.etherscan.io/address/0xe043e3ee3a1c044abc013050356f15bdbbb5ba92#code

Introduction

Given the opportunity to review Inuyasha Project's smart contract source code, we in the report outline our systematic approach to evaluate potential security issues in the smart contract implementation, expose possible semantic inconsistencies between smart contract code and design document, and provide additional suggestions or recommendations for improvement. Our results show that the given version of smart contracts is ready to launch after resolving the mentioned issues, there are no critical or high issues found related to business logic, security or performance.

About Inuyasha: -

Item	Description
Issuer	Inuyasha
Type	ERC20
Website	www.inuyasha.io
Platform	Solidity
Audit Method	Whitebox
Latest Audit Report	October 18, 2021

The Test Method Information: -

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open-source code, non-open-source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant effect on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

The Full List of Check Items:

Category	Check Item
Basic Coding Bugs	Constructor Mismatch
	Ownership Takeover
	Redundant Fallback Function
	Overflows & Underflows
	Reentrancy
	MONEY-Giving Bug
	Blackhole
	Unauthorized Self-Destruct
	Revert DoS
	Unchecked External Call
	Gasless Send
	Send Instead of Transfer
	Costly Loop
	(Unsafe) Use of Untrusted Libraries
	(Unsafe) Use of Predictable Variables
	Transaction Ordering Dependence
	Deprecated Uses
Semantic Consistency Checks	Semantic Consistency Checks
	Business Logics Review

Advanced DeFi Scrutiny	Functionality Checks
	Authentication Management
	Access Control & Authorization
	Oracle Security
	Digital Asset Escrow
	Kill-Switch Mechanism
	Operation Trails & Event Generation
	ERC20 Idiosyncrasies Handling
	Frontend-Contract Integration
	Deployment Consistency
	Holistic Risk Management
Additional Recommendations	Avoiding Use of Variadic Byte Array
	Using Fixed Compiler Version
	Making Visibility Level Explicit
	Making Type Inference Explicit
	Adhering To Function Declaration Strictly
	Following Other Best Practices

Common Weakness Enumeration (CWE) Classifications Used in This Audit:

Category	Summary
Configuration	Weaknesses in this category are typically introduced during the configuration of the software.
Data Processing Issues	Weaknesses in this category are typically found in functionality that processes data.
Numeric Errors	Weaknesses in this category are related to improper calculation or conversion of numbers.
Security Features	Weaknesses in this category are concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management. (Software security is not security software.)
Time and State	Weaknesses in this category are related to the improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiple systems, processes, or threads.
Error Conditions, Return Values, Status Codes	Weaknesses in this category include weaknesses that occur if a function does not generate the correct return/status code, or if the application does not handle all possible return/status codes that could be generated by a function.
Resource Management	Weaknesses in this category are related to improper management of system resources.

Behavioral Issues	Weaknesses in this category are related to unexpected behaviors from code that an application uses.
Business Logics	Weaknesses in this category identify some of the underlying problems that commonly allow attackers to manipulate the business logic of an application. Errors in business logic can be devastating to an entire application.
Initialization and Cleanup	Weaknesses in this category occur in behaviors that are used for initialization and breakdown.
Arguments and Parameters	Weaknesses in this category are related to improper use arguments or parameters within function calls.
Expression Issues	Weaknesses in this category are related to incorrectly written expressions within code.
Coding Practices	Weaknesses in this category are related to coding practices that are deemed unsafe and increase the chances that an ex pilotable vulnerability will be present in the application. They may not directly introduce a vulnerability, but indicate the product has not been carefully developed or maintained.

Findings

Summary

Here is a summary of our findings after analyzing the Inuyasha's Smart Contract. During the first phase of our audit, we studied the smart contract sourcecode and ran our in-house static code analyzer through the Specific tool. The purpose here is to statically identify known coding bugs, and then manually verify (reject or confirm) issues reported by tool. We further manually review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.

Severity	No. of Issues
Critical	0
High	0
Medium	0
Low	2
Total	2

We have so far identified that there are potential issues with severity of **0 Critical, 0 High, 0 Medium, and 2 Low**. Overall, these smart contracts are well- designed and engineered, though the implementation can be improved and bug free by common recommendations given under POCs.

Functional Overview

($) = payable function # = non-constant function	[Pub] public [Ext] external [Prv] private [Int] internal

+ [Int] IERC20
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] transfer #
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #

+ [Lib] SafeMath
  - [Int] add
  - [Int] sub
  - [Int] sub
  - [Int] mul
  - [Int] div
  - [Int] div
  - [Int] mod
  - [Int] mod

+ Context
  - [Int] _msgSender
  - [Int] _msgData

+ [Lib] Address
  - [Int] isContract
  - [Int] sendValue #
  - [Int] functionCall #
  - [Int] functionCall #
  - [Int] functionCallWithValue #
  - [Int] functionCallWithValue #
  - [Prv] _functionCallWithValue #

+ Ownable (Context)
  - [Pub] <Constructor> #
  - [Pub] owner
  - [Pub] renounceOwnership #
    - modifiers: onlyOwner
  - [Pub] transferOwnership #
    - modifiers: onlyOwner
  - [Pub] geUnlockTime
  - [Pub] lock #
    - modifiers: onlyOwner
  - [Pub] unlock #

+ [Int] IUniswapV2Factory
  - [Ext] feeTo
  - [Ext] feeToSetter
  - [Ext] getPair
  - [Ext] allPairs
  - [Ext] allPairsLength
  - [Ext] createPair #
  - [Ext] setFeeTo #
  - [Ext] setFeeToSetter #

+ [Int] IUniswapV2Pair
  - [Ext] name
  - [Ext] symbol
  - [Ext] decimals
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transfer #
  - [Ext] transferFrom #
  - [Ext] DOMAIN_SEPARATOR
  - [Ext] PERMIT_TYPEHASH
  - [Ext] nonces
  - [Ext] permit #
  - [Ext] MINIMUM_LIQUIDITY
  - [Ext] factory
  - [Ext] token0
  - [Ext] token1
  - [Ext] getReserves
  - [Ext] price0CumulativeLast
  - [Ext] price1CumulativeLast
  - [Ext] kLast
  - [Ext] burn #
  - [Ext] swap #
  - [Ext] skim #
  - [Ext] sync #
  - [Ext] initialize #

+ [Int] IUniswapV2Router01
  - [Ext] factory
  - [Ext] WETH

- [Ext] addLiquidity #
- [Ext] addLiquidityETH ($)
- [Ext] removeLiquidity #
- [Ext] removeLiquidityETH #
- [Ext] removeLiquidityWithPermit #
- [Ext] removeLiquidityETHWithPermit #
- [Ext] swapExactTokensForTokens #
- [Ext] swapTokensForExactTokens #
- [Ext] swapExactETHForTokens ($)
- [Ext] swapTokensForExactETH #
- [Ext] swapExactTokensForETH #
- [Ext] swapETHForExactTokens ($)
- [Ext] quote
- [Ext] getAmountOut
- [Ext] getAmountIn
- [Ext] getAmountsOut
- [Ext] getAmountsIn

+ [Int] IUniswapV2Router02 (IUniswapV2Router01)
- [Ext] removeLiquidityETHSupportingFeeOnTransferTokens #
- [Ext] removeLiquidityETHWithPermitSupportingFeeOnTransferTokens #
- [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
- [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
- [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ Inuyasha (Context, IERC20, Ownable)
- [Pub] <Constructor> #
- [Pub] name
- [Pub] symbol
- [Pub] decimals
- [Pub] totalSupply

- [Pub] balanceOf
- [Pub] transfer #
- [Pub] allowance
- [Pub] approve #
- [Pub] transferFrom #
- [Pub] increaseAllowance #
- [Pub] decreaseAllowance #
- [Pub] isExcludedFromReward
- [Pub] totalFees
- [Pub] deliver #
- [Pub] reflectionFromToken
- [Pub] tokenFromReflection
- [Pub] excludeFromReward #
  - modifiers: onlyOwner
- [Ext] includeInReward #
  - modifiers: onlyOwner
- [Prv] _transferBothExcluded #
- [Ext] <Fallback> ($)
- [Prv] _reflectFee #
- [Prv] _getValues
- [Prv] _getTValues
- [Prv] _getRValues
- [Prv] _getRate
- [Prv] _getCurrentSupply
- [Prv] _takeLiquidity #
- [Prv] calculateTaxFee
- [Prv] calculateLiquidityFee
- [Prv] removeAllFee #
- [Prv] restoreAllFee #
- [Pub] isExcludedFromFee
- [Prv] _approve #

- [Prv] _transfer #
- [Prv] swapAndLiquify #
  - modifiers: lockTheSwap
- [Prv] swapTokensForEth #
- [Prv] swapETHForTokens #
- [Prv] buyBackTokens #
  - modifiers: lockTheSwap
- [Prv] addLiquidity #
- [Prv] _tokenTransfer #
- [Prv] _transferStandard #
- [Prv] takeMarketing #
- [Prv] _transferToExcluded #
- [Prv] _transferFromExcluded #
- [Ext] prepareForPresale #
  - modifiers: onlyOwner
- [Ext] afterPresale #
  - modifiers: onlyOwner
- [Pub] excludeFromFee #
  - modifiers: onlyOwner
- [Pub] includeInFee #
  - modifiers: onlyOwner
- [Ext] setMarketingWallet #
  - modifiers: onlyOwner
- [Ext] setBuyBackDivisor #
  - modifiers: onlyOwner
- [Pub] setBuyBackEnabled #
  - modifiers: onlyOwner
- [Pub] SetBuyBackUpperLimitAmount #
- [Pub] buyBackUpperLimitAmount
- [Pub] buyBackDivisor
- [Ext] setTaxFeePercent #

- modifiers: onlyOwner
- [Ext] setMarketingFeePercent #
  - modifiers: onlyOwner
- [Ext] setLiquidityFeePercent #
  - modifiers: onlyOwner
- [Ext] setNumTokensSellToAddToLiquidity #
  - modifiers: onlyOwner
- [Ext] setMaxTxAmount #
  - modifiers: onlyOwner
- [Pub] setSwapAndLiquifyEnabled #
  - modifiers: onlyOwner

Detailed Results

Issues Checking Status

1. **State Variable Default Visibility**

- SWC ID:108
- Severity: Low
- Location: https://ropsten.etherscan.io/address/0xe043e3ee3a1c044abc013050356f15bdbbb5ba92#code
- Relationships: CWE-710: Improper Adherence to Coding Standards
- Description: State variable visibility is not set. It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwapAndLiquify" is internal. Other possible visibility settings are public and private.

```
730
731     bool inSwapAndLiquify;
732     bool public swapAndLiquifyEnabled = false;
733
```

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

2. **Block values as a proxy for time**

- SWC ID:116
- Severity: Low
- Location: https://ropsten.etherscan.io/address/0xe043e3ee3a1c044abc013050356f15bdbbb5ba92#code
- Relationships: CWE-829: Inclusion of Functionality from Untrusted Control Sphere
- Description: A control flow decision is made based on The block.timestamp environment variable. The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

```
465     //Unlocks the contract for owner when _lockTime is exceeds
466     function unlock() public virtual {
467         require(_previousOwner == msg.sender, "You don't have permission to unlock");
468         require(block.timestamp > _lockTime , "Contract is locked until 7 days");
469         emit OwnershipTransferred(_owner, _previousOwner);
470         _owner = _previousOwner;
471     }
472 }
```

- Remediations: Developers should write smart contracts with the notion that block values are not precise, and the use of them can lead to unexpected effects. Alternatively, they may make use oracles.

Automated Tools Results

Slither: -

Smart Contract Audit

```
Reentrancy in Inuyasha._transfer(address,address,uint256) (Inuyasha.sol#990-1023):
        External calls:
        - swapAndLiquify(contractTokenBalance) (Inuyasha.sol#1007)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Inuyasha.sol#1101-1108)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Inuyasha.sol#1
064-1070)
        - buyBackTokens(balance.div(_buyBackDivisor)) (Inuyasha.sol#1010)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,deadAddress,block.timestamp.add(300)) (Inuy
asha.sol#1080-1085)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (Inuyasha.sol#1007)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Inuyasha.sol#1101-1108)
        - buyBackTokens(balance.div(_buyBackDivisor)) (Inuyasha.sol#1010)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,deadAddress,block.timestamp.add(300)) (Inuy
asha.sol#1080-1085)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount) (Inuyasha.sol#1022)
                - _marketingFee = _previousmarketingFee (Inuyasha.sol#975)
                - _marketingFee = 0 (Inuyasha.sol#969)
        - _tokenTransfer(from,to,amount) (Inuyasha.sol#1022)
                - _previousLiquidityFee = _liquidityFee (Inuyasha.sol#964)
        - _tokenTransfer(from,to,amount) (Inuyasha.sol#1022)
                - _previousTaxFee = _taxFee (Inuyasha.sol#963)
        - _tokenTransfer(from,to,amount) (Inuyasha.sol#1022)
                - _previousmarketingFee = _marketingFee (Inuyasha.sol#965)
        - _tokenTransfer(from,to,amount) (Inuyasha.sol#1022)
                - _tFeeTotal = _tFeeTotal.add(tFee) (Inuyasha.sol#899)
        - _tokenTransfer(from,to,amount) (Inuyasha.sol#1022)
                - _taxFee = _previousTaxFee (Inuyasha.sol#973)
                - _taxFee = 0 (Inuyasha.sol#967)
Reentrancy in Inuyasha.constructor() (Inuyasha.sol#750-772):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (Inuyasha.sol#761-762)
        State variables written after the call(s):
        - _isExcludedFromFee[owner()] = true (Inuyasha.sol#768)
        - _isExcludedFromFee[address(this)] = true (Inuyasha.sol#769)
        - uniswapV2Router = _uniswapV2Router (Inuyasha.sol#765)
Reentrancy in Inuyasha.swapAndLiquify(uint256) (Inuyasha.sol#1025-1053):
        External calls:
        - swapTokensForEth(buybackShare) (Inuyasha.sol#1036)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Inuyasha.sol#1
064-1070)
        - swapTokensForEth(half) (Inuyasha.sol#1044)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Inuyasha.sol#1
064-1070)
        State variables written after the call(s):
        - swapTokensForEth(half) (Inuyasha.sol#1044)
                - _allowances[owner][spender] = amount (Inuyasha.sol#906)
Reentrancy in Inuyasha.swapAndLiquify(uint256) (Inuyasha.sol#1025-1053):
        External calls:
        - swapTokensForEth(buybackShare) (Inuyasha.sol#1036)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Inuyasha.sol#1
064-1070)
        - swapTokensForEth(half) (Inuyasha.sol#1044)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Inuyasha.sol#1
064-1070)
        - addLiquidity(otherHalf,newBalance) (Inuyasha.sol#1050)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Inuyasha.sol#1101-1108)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (Inuyasha.sol#1050)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Inuyasha.sol#1101-1108)
        State variables written after the call(s):
        - addLiquidity(otherHalf,newBalance) (Inuyasha.sol#1050)
                - _allowances[owner][spender] = amount (Inuyasha.sol#906)
Reentrancy in Inuyasha.transferFrom(address,address,uint256) (Inuyasha.sol#889-813):
        External calls:
        - _transfer(sender,recipient,amount) (Inuyasha.sol#810)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Inuyasha.sol#1101-1108)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,deadAddress,block.timestamp.add(300)) (Inuy
asha.sol#1080-1085)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Inuyasha.sol#1
064-1070)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (Inuyasha.sol#810)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Inuyasha.sol#1101-1108)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,deadAddress,block.timestamp.add(300)) (Inuy
asha.sol#1080-1085)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (Inuyasha.sol#811)
                - _allowances[owner][spender] = amount (Inuyasha.sol#906)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Reentrancy in Inuyasha._transfer(address,address,uint256) (Inuyasha.sol#990-1023):
        External calls:
        - swapAndLiquify(contractTokenBalance) (Inuyasha.sol#1007)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Inuyasha.sol#1101-1108)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Inuyasha.sol#1
864-1870)
        - buyBackTokens(balance.div(_buyBackDivisor)) (Inuyasha.sol#1016)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,deadAddress,block.timestamp.add(300)) (Inuy
asha.sol#1080-1085)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (Inuyasha.sol#1007)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Inuyasha.sol#1101-1108)
        - buyBackTokens(balance.div(_buyBackDivisor)) (Inuyasha.sol#1016)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,deadAddress,block.timestamp.add(300)) (Inuy
asha.sol#1080-1085)
        Event emitted after the call(s):
        - SwapETHForTokens(amount,path) (Inuyasha.sol#1087)
                - buyBackTokens(balance.div(_buyBackDivisor)) (Inuyasha.sol#1016)
        - Transfer(sender,marketingWallet,tMarketing) (Inuyasha.sol#1103)
                - _tokenTransfer(from,to,amount) (Inuyasha.sol#1022)
        - Transfer(sender,recipient,tTransferAmount) (Inuyasha.sol#1104)
                - _tokenTransfer(from,to,amount) (Inuyasha.sol#1022)
        - Transfer(sender,recipient,tTransferAmount) (Inuyasha.sol#1151)
                - _tokenTransfer(from,to,amount) (Inuyasha.sol#1022)
        - Transfer(sender,recipient,tTransferAmount) (Inuyasha.sol#1174)
                - _tokenTransfer(from,to,amount) (Inuyasha.sol#1022)
        - Transfer(sender,recipient,tTransferAmount) (Inuyasha.sol#809)
                - _tokenTransfer(from,to,amount) (Inuyasha.sol#1022)
Reentrancy in Inuyasha.constructor() (Inuyasha.sol#756-772):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (Inuyasha.sol#761-762)
        Event emitted after the call(s):
        - Transfer(address(0),_msgSender(),_tTotal) (Inuyasha.sol#771)
Reentrancy in Inuyasha.swapAndLiquify(uint256) (Inuyasha.sol#1025-1053):
        External calls:
        - swapTokensForEth(buybackShare) (Inuyasha.sol#1030)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Inuyasha.sol#1
864-1870)
        - swapTokensForEth(half) (Inuyasha.sol#1044)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Inuyasha.sol#1
864-1870)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Inuyasha.sol#987)
                - swapTokensForEth(half) (Inuyasha.sol#1044)
Reentrancy in Inuyasha.swapAndLiquify(uint256) (Inuyasha.sol#1025-1053):
        External calls:
        - swapTokensForEth(buybackShare) (Inuyasha.sol#1030)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Inuyasha.sol#1
864-1870)
        - swapTokensForEth(half) (Inuyasha.sol#1044)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Inuyasha.sol#1
864-1870)
        - addLiquidity(otherHalf,newBalance) (Inuyasha.sol#1050)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Inuyasha.sol#1101-1108)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (Inuyasha.sol#1050)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Inuyasha.sol#1101-1108)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Inuyasha.sol#987)
                - addLiquidity(otherHalf,newBalance) (Inuyasha.sol#1050)
        - SwapAndLiquify(half,newBalance,otherHalf) (Inuyasha.sol#1052)
Reentrancy in Inuyasha.swapETHForTokens(uint256) (Inuyasha.sol#1073-1088):
        External calls:
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,deadAddress,block.timestamp.add(300)) (Inuyasha.sol
#1080-1085)
        Event emitted after the call(s):
        - SwapETHForTokens(amount,path) (Inuyasha.sol#1087)
Reentrancy in Inuyasha.transferFrom(address,address,uint256) (Inuyasha.sol#809-813):
        External calls:
        - _transfer(sender,recipient,amount) (Inuyasha.sol#810)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Inuyasha.sol#1101-1108)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,deadAddress,block.timestamp.add(300)) (Inuy
asha.sol#1080-1085)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Inuyasha.sol#1
864-1870)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (Inuyasha.sol#810)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Inuyasha.sol#1101-1108)
                - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,deadAddress,block.timestamp.add(300)) (Inuy
asha.sol#1080-1085)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Inuyasha.sol#987)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (Inuyasha.sol#8
11)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

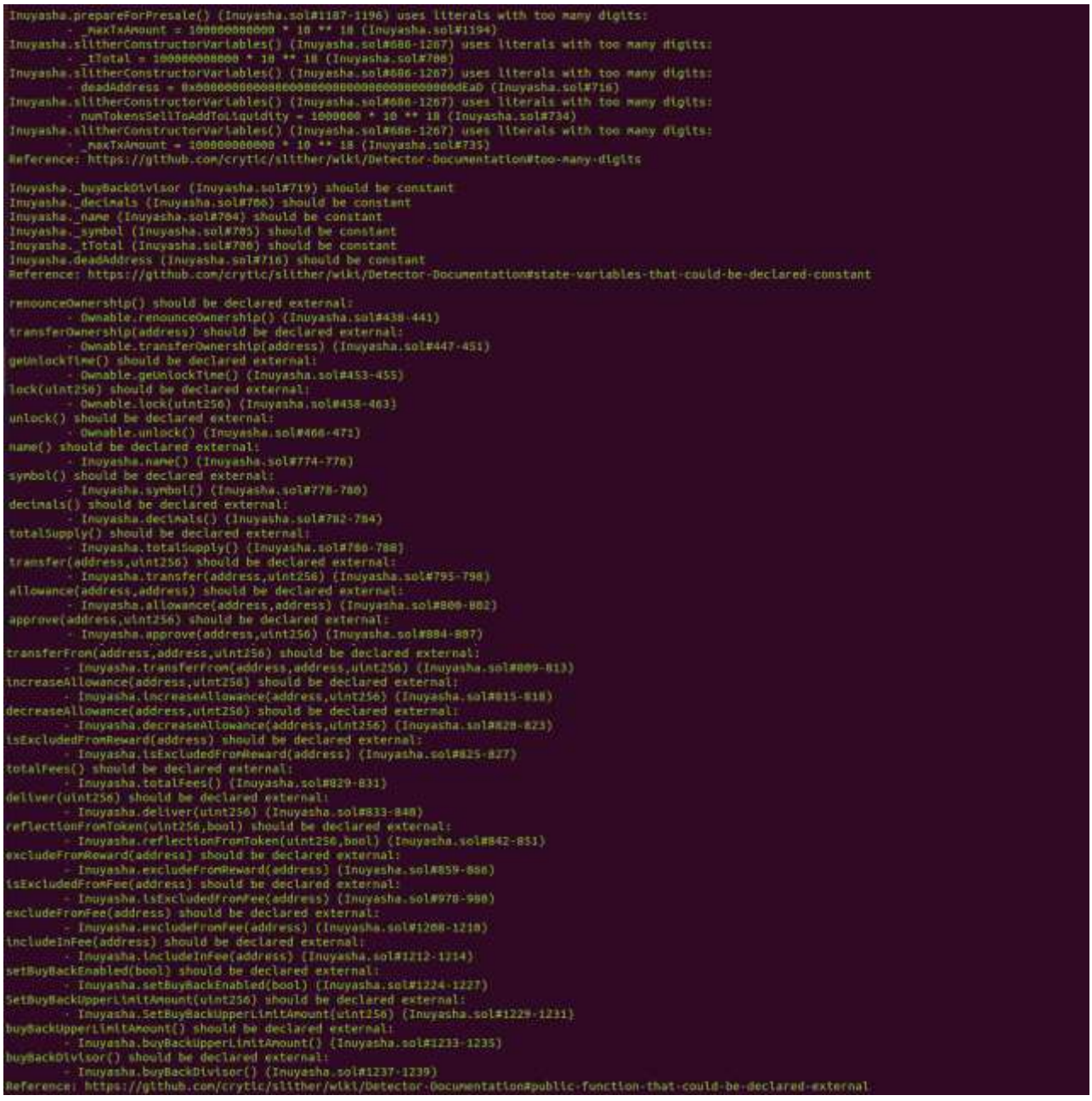
Smart Contract Audit

```
Ownable.unlock() (Inuyasha.sol#466-471) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > _lockTime,Contract is locked until 7 days) (Inuyasha.sol#468)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (Inuyasha.sol#271-280) uses assembly
        - INLINE ASM (Inuyasha.sol#278)
Address._functionCallWithValue(address,bytes,uint256,string) (Inuyasha.sol#364-385) uses assembly
        - INLINE ASM (Inuyasha.sol#377-380)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address._functionCallWithValue(address,bytes,uint256,string) (Inuyasha.sol#364-385) is never used and should be removed
Address.functionCall(address,bytes) (Inuyasha.sol#324-326) is never used and should be removed
Address.functionCall(address,bytes,string) (Inuyasha.sol#334-336) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Inuyasha.sol#349-351) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (Inuyasha.sol#359-362) is never used and should be removed
Address.isContract(address) (Inuyasha.sol#271-280) is never used and should be removed
Address.sendValue(address,uint256) (Inuyasha.sol#298-304) is never used and should be removed
Context._msgData() (Inuyasha.sol#243-246) is never used and should be removed
SafeMath.mod(uint256,uint256) (Inuyasha.sol#216-218) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Inuyasha.sol#232-235) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Inuyasha._rTotal (Inuyasha.sol#701) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
Inuyasha._previousTaxFee (Inuyasha.sol#709) is set pre-construction with a non-constant function or state variable:
        - _taxFee
Inuyasha._previousLiquidityFee (Inuyasha.sol#712) is set pre-construction with a non-constant function or state variable:
        - _liquidityFee
Inuyasha._previousmarketingFee (Inuyasha.sol#720) is set pre-construction with a non-constant function or state variable:
        - _marketingFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version0.8.4 (Inuyasha.sol#7) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Inuyasha.sol#298-304):
        - (success) = recipient.call{value: amount}() (Inuyasha.sol#302)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (Inuyasha.sol#364-385):
        - (success,returndata) = target.call{value: weiValue}(data) (Inuyasha.sol#368)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IUniswapV2Pair.DOMAIN_SEPARATOR() (Inuyasha.sol#510) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (Inuyasha.sol#511) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (Inuyasha.sol#527) is not in mixedCase
Function IUniswapV2Router01.WETH() (Inuyasha.sol#548) is not in mixedCase
Parameter Inuyasha.calculateTaxFee(uint256)._amount (Inuyasha.sol#948) is not in mixedCase
Parameter Inuyasha.calculateLiquidityFee(uint256)._amount (Inuyasha.sol#954) is not in mixedCase
Parameter Inuyasha.setBuyBackEnabled(bool)._enabled (Inuyasha.sol#1224) is not in mixedCase
Function Inuyasha.setBuyBackUpperLimitAmount(uint256) (Inuyasha.sol#1229-1231) is not in mixedCase
Parameter Inuyasha.setBuyBackUpperLimitAmount(uint256)._newLimit (Inuyasha.sol#1229) is not in mixedCase
Parameter Inuyasha.setSwapAndLiquifyEnabled(bool)._enabled (Inuyasha.sol#1202) is not in mixedCase
Variable Inuyasha._taxFee (Inuyasha.sol#708) is not in mixedCase
Variable Inuyasha._liquidityFee (Inuyasha.sol#711) is not in mixedCase
Variable Inuyasha._marketingFee (Inuyasha.sol#724) is not in mixedCase
Variable Inuyasha._maxTxAmount (Inuyasha.sol#735) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (Inuyasha.sol#244)" inContext (Inuyasha.sol#238-247)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Inuyasha.sol#553) is too sim
ilar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Inuyasha.sol#554)
Variable Inuyasha.reflectionFromToken(uint256,bool).rTransferAmount (Inuyasha.sol#848) is too similar to Inuyasha._transferBothExcluded(address,addres
s,uint256).tTransferAmount (Inuyasha.sol#882)
Variable Inuyasha._getValues(uint256).rTransferAmount (Inuyasha.sol#904) is too similar to Inuyasha._getTValues(uint256).tTransferAmount (Inuyasha.sol
#911)
Variable Inuyasha._transferBothExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#882) is too similar to Inuyasha._getTValues(uint256).tT
ransferAmount (Inuyasha.sol#911)
Variable Inuyasha._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#919) is too similar to Inuyasha._transferFromExcluded(add
ress,address,uint256).tTransferAmount (Inuyasha.sol#1178)
Variable Inuyasha.reflectionFromToken(uint256,bool).rTransferAmount (Inuyasha.sol#848) is too similar to Inuyasha.takeMarketing(address,uint256,uint25
6,uint256).tTransferAmount (Inuyasha.sol#1164)
Variable Inuyasha._transferStandard(address,address,uint256).rTransferAmount (Inuyasha.sol#1145) is too similar to Inuyasha._transferBothExcluded(addr
ess,address,uint256).tTransferAmount (Inuyasha.sol#882)
Variable Inuyasha._transferFromExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1170) is too similar to Inuyasha._transferToExcluded(ad
dress,address,uint256).tTransferAmount (Inuyasha.sol#1160)
Variable Inuyasha.takeMarketing(address,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#1154) is too similar to Inuyasha._transferToExcluded(ad
dress,address,uint256).tTransferAmount (Inuyasha.sol#1160)
Variable Inuyasha._transferFromExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1170) is too similar to Inuyasha._transferFromExcluded(
address,address,uint256).tTransferAmount (Inuyasha.sol#1178)
Variable Inuyasha._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#919) is too similar to Inuyasha._getTValues(uint256).tTra
nsferAmount (Inuyasha.sol#911)
Variable Inuyasha._transferStandard(address,address,uint256).rTransferAmount (Inuyasha.sol#1145) is too similar to Inuyasha.takeMarketing(address,uint
256,uint256,uint256).tTransferAmount (Inuyasha.sol#1154)
Variable Inuyasha.takeMarketing(address,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#1154) is too similar to Inuyasha._transferFromExcluded(
address,address,uint256).tTransferAmount (Inuyasha.sol#1170)
```

Smart Contract Audit

```
Variable Inuyasha.reflectionFromToken(uint256,bool).rTransferAmount (Inuyasha.sol#848) is too similar to Inuyasha._transferToExcluded(address,address,
uint256).tTransferAmount (Inuyasha.sol#1108)
Variable Inuyasha.reflectionFromToken(uint256,bool).rTransferAmount (Inuyasha.sol#848) is too similar to Inuyasha._transferFromExcluded(address,addres
s,uint256).tTransferAmount (Inuyasha.sol#1178)
Variable Inuyasha._transferFromExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1178) is too similar to Inuyasha._getTValues(uint256).t
TransferAmount (Inuyasha.sol#911)
Variable Inuyasha._transferBothExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#882) is too similar to Inuyasha._transferFromExcluded(a
ddress,address,uint256).tTransferAmount (Inuyasha.sol#1178)
Variable Inuyasha._transferStandard(address,address,uint256).rTransferAmount (Inuyasha.sol#1145) is too similar to Inuyasha._transferToExcluded(addres
s,address,uint256).tTransferAmount (Inuyasha.sol#1108)
Variable Inuyasha.takeMarketing(address,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#1154) is too similar to Inuyasha._getTValues(uint256).t
TransferAmount (Inuyasha.sol#911)
Variable Inuyasha._transferStandard(address,address,uint256).rTransferAmount (Inuyasha.sol#1145) is too similar to Inuyasha._transferFromExcluded(addr
ess,address,uint256).tTransferAmount (Inuyasha.sol#1178)
Variable Inuyasha._transferToExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1108) is too similar to Inuyasha._transferToExcluded(addr
ess,address,uint256).tTransferAmount (Inuyasha.sol#1108)
Variable Inuyasha.reflectionFromToken(uint256,bool).rTransferAmount (Inuyasha.sol#848) is too similar to Inuyasha._getTValues(uint256).tTransferAmount
 (Inuyasha.sol#911)
Variable Inuyasha.takeMarketing(address,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#1154) is too similar to Inuyasha._transferBothExcluded(
address,address,uint256).tTransferAmount (Inuyasha.sol#882)
Variable Inuyasha._getValues(uint256).rTransferAmount (Inuyasha.sol#904) is too similar to Inuyasha._transferBothExcluded(address,address,uint256).tTr
ansferAmount (Inuyasha.sol#882)
Variable Inuyasha._transferBothExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#882) is too similar to Inuyasha._transferBothExcluded(a
ddress,address,uint256).tTransferAmount (Inuyasha.sol#882)
Variable Inuyasha._transferStandard(address,address,uint256).rTransferAmount (Inuyasha.sol#1145) is too similar to Inuyasha._getTValues(uint256).tTran
sferAmount (Inuyasha.sol#911)
Variable Inuyasha._transferToExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1108) is too similar to Inuyasha._transferFromExcluded(ad
dress,address,uint256).tTransferAmount (Inuyasha.sol#1178)
Variable Inuyasha._getValues(uint256).rTransferAmount (Inuyasha.sol#904) is too similar to Inuyasha.takeMarketing(address,uint256,uint256,uint256).tTr
ansferAmount (Inuyasha.sol#1154)
Variable Inuyasha._transferBothExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#882) is too similar to Inuyasha.takeMarketing(address,u
int256,uint256,uint256).tTransferAmount (Inuyasha.sol#1154)
Variable Inuyasha._transferToExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1108) is too similar to Inuyasha._getTValues(uint256).tTr
ansferAmount (Inuyasha.sol#911)
Variable Inuyasha._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#919) is too similar to Inuyasha._transferBothExcluded(add
ress,address,uint256).tTransferAmount (Inuyasha.sol#882)
Variable Inuyasha._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#919) is too similar to Inuyasha.takeMarketing(address,uin
t256,uint256,uint256).tTransferAmount (Inuyasha.sol#1154)
Variable Inuyasha._getValues(uint256).rTransferAmount (Inuyasha.sol#904) is too similar to Inuyasha._transferToExcluded(address,address,uint256).tTran
sferAmount (Inuyasha.sol#1108)
Variable Inuyasha._transferBothExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#882) is too similar to Inuyasha._transferToExcluded(add
ress,address,uint256).tTransferAmount (Inuyasha.sol#1108)
Variable Inuyasha._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#919) is too similar to Inuyasha._transferToExcluded(addre
ss,address,uint256).tTransferAmount (Inuyasha.sol#1108)
Variable Inuyasha._getValues(uint256).rTransferAmount (Inuyasha.sol#904) is too similar to Inuyasha._transferFromExcluded(address,address,uint256).tTr
ansferAmount (Inuyasha.sol#1178)
Variable Inuyasha.takeMarketing(address,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#1154) is too similar to Inuyasha._getValues(uint256).tT
ransferAmount (Inuyasha.sol#903)
Variable Inuyasha._transferFromExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1178) is too similar to Inuyasha._getValues(uint256).tT
ransferAmount (Inuyasha.sol#903)
Variable Inuyasha._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#919) is too similar to Inuyasha._transferStandard(address
,address,uint256).tTransferAmount (Inuyasha.sol#1145)
Variable Inuyasha._transferToExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1108) is too similar to Inuyasha._transferBothExcluded(ad
dress,address,uint256).tTransferAmount (Inuyasha.sol#882)
Variable Inuyasha.reflectionFromToken(uint256,bool).rTransferAmount (Inuyasha.sol#848) is too similar to Inuyasha._transferStandard(address,address,ui
nt256).tTransferAmount (Inuyasha.sol#1145)
Variable Inuyasha._transferFromExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1178) is too similar to Inuyasha._transferBothExcluded(
address,address,uint256).tTransferAmount (Inuyasha.sol#882)
Variable Inuyasha._transferBothExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#882) is too similar to Inuyasha._transferStandard(addre
ss,address,uint256).tTransferAmount (Inuyasha.sol#1145)
Variable Inuyasha._getValues(uint256).rTransferAmount (Inuyasha.sol#904) is too similar to Inuyasha._transferStandard(address,address,uint256).tTransf
erAmount (Inuyasha.sol#1145)
Variable Inuyasha._transferToExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1108) is too similar to Inuyasha.takeMarketing(address,ui
nt256,uint256,uint256).tTransferAmount (Inuyasha.sol#1154)
Variable Inuyasha._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#919) is too similar to Inuyasha._getValues(uint256).tTran
sferAmount (Inuyasha.sol#903)
Variable Inuyasha.reflectionFromToken(uint256,bool).rTransferAmount (Inuyasha.sol#848) is too similar to Inuyasha._getValues(uint256).tTransferAmount
 (Inuyasha.sol#903)
Variable Inuyasha._transferStandard(address,address,uint256).rTransferAmount (Inuyasha.sol#1145) is too similar to Inuyasha._transferStandard(address,
address,uint256).tTransferAmount (Inuyasha.sol#1145)
Variable Inuyasha._transferToExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1108) is too similar to Inuyasha._transferStandard(addres
s,address,uint256).tTransferAmount (Inuyasha.sol#1145)
Variable Inuyasha.takeMarketing(address,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#1154) is too similar to Inuyasha.takeMarketing(address,
uint256,uint256,uint256).tTransferAmount (Inuyasha.sol#1154)
Variable Inuyasha._transferFromExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1178) is too similar to Inuyasha.takeMarketing(address,
uint256,uint256,uint256).tTransferAmount (Inuyasha.sol#1154)
Variable Inuyasha.takeMarketing(address,uint256,uint256,uint256).rTransferAmount (Inuyasha.sol#1154) is too similar to Inuyasha._transferStandard(addr
ess,address,uint256).tTransferAmount (Inuyasha.sol#1145)
Variable Inuyasha._transferFromExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1178) is too similar to Inuyasha._transferStandard(addr
ess,address,uint256).tTransferAmount (Inuyasha.sol#1145)
Variable Inuyasha._transferBothExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#882) is too similar to Inuyasha._getValues(uint256).tTr
ansferAmount (Inuyasha.sol#903)
Variable Inuyasha._getValues(uint256).rTransferAmount (Inuyasha.sol#904) is too similar to Inuyasha._getValues(uint256).tTransferAmount (Inuyasha.sol#
903)
Variable Inuyasha._transferStandard(address,address,uint256).rTransferAmount (Inuyasha.sol#1145) is too similar to Inuyasha._getValues(uint256).tTrans
ferAmount (Inuyasha.sol#903)
Variable Inuyasha._transferToExcluded(address,address,uint256).rTransferAmount (Inuyasha.sol#1108) is too similar to Inuyasha._getValues(uint256).tTra
nsferAmount (Inuyasha.sol#903)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

Smart Contract Audit



MythX: -

Report for Inuyasha.sol
https://dashboard.mythx.io/#/console/analyses/1aa3c4c7-046a-47a5-8ff7-9cc156e4bf70

Line	SWC Title	Severity	Short Description
107	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
139	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
162	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
163	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
198	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
234	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "%" discovered
461	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
700	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
700	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
701	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "%" discovered
701	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
718	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
718	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
731	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
734	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
734	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
735	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
735	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
870	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered

871	(SWC-110) Assert Violation	Unknown	Out of bounds array access
872	(SWC-110) Assert Violation	Unknown	Out of bounds array access
872	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
872	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
931	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
932	(SWC-110) Assert Violation	Unknown	Out of bounds array access
933	(SWC-110) Assert Violation	Unknown	Out of bounds array access
934	(SWC-110) Assert Violation	Unknown	Out of bounds array access
950	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
956	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
1058	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1059	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1076	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1077	(SWC-110) Assert Violation	Unknown	Out of bounds array access
1194	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
1194	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
1254	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
1254	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
1259	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
1259	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered

Mythril: -

```
==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Ownable
Function name: unlock()
PC address: 426
Estimated Gas Usage: 1888 - 1963
A control flow decision is made based on The block.timestamp environment variable.
The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, bloc
k number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Do
n't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into
miners.
--------------------
In file: Inuyasha.sol:460

require(block.timestamp > _lockTime , "Contract is locked until 7 days")

--------------------
Initial State:

Account: [CREATOR], balance: 0x1, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: lock(uint256), txdata: 0xdd467064ffffffffffffffffffffffffffffffffffffffffffffffffffffffff, value: 0x0
Caller: [CREATOR], function: unlock(), txdata: 0xa69df4b5, value: 0x0
```

```
==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Ownable
Function name: lock(uint256)
PC address: 1060
Estimated Gas Usage: 12034 - 52929
A control flow decision is made based on The block.timestamp environment variable.
The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, bloc
k number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Do
n't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into
miners.
--------------------
In file: #utility.yul:57

if

--------------------
Initial State:

Account: [CREATOR], balance: 0x0, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: lock(uint256), txdata: 0xdd467004000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0
```

Solhint: -

```
Linter results:

Inuyasha.sol:7:1: Error: Compiler version 0.8.4 does not satisfy the r semver requirement

Inuyasha.sol:410:5: Error: Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)

Inuyasha.sol:463:21: Error: Avoid to make time-based decisions in your business logic

Inuyasha.sol:468:17: Error: Avoid to make time-based decisions in your business logic

Inuyasha.sol:510:5: Error: Function name must be in mixedCase

Inuyasha.sol:511:5: Error: Function name must be in mixedCase

Inuyasha.sol:527:5: Error: Function name must be in mixedCase

Inuyasha.sol:548:5: Error: Function name must be in mixedCase

Inuyasha.sol:686:1: Error: Contract has 20 states declarations but allowed no more than 15

Inuyasha.sol:731:5: Error: Explicitly mark visibility of state

Inuyasha.sol:756:5: Error: Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)

Inuyasha.sol:895:32: Error: Code contains empty blocks
```


Imuyasha.sol:1027:9: Error: Variable name must be in mixedCase
Imuyasha.sol:1060:13: Error: Avoid to make time-based decisions in your business logic
Imuyasha.sol:1004:13: Error: Avoid to make time-based decisions in your business logic
Imuyasha.sol:1107:13: Error: Avoid to make time-based decisions in your business logic
Imuyasha.sol:1229:5: Error: Function name must be in mixedCase

Basic Coding Bugs

1. **Constructor Mismatch**
   - Description: Whether the contract name and its constructor are not identical to each other.
   - Result: PASSED
   - Severity: Critical
2. **Ownership Takeover**
   - Description: Whether the set owner function is not protected.
   - Result: PASSED
   - Severity: Critical
3. **Redundant Fallback Function**
   - Description: Whether the contract has a redundant fallback function.
   - Result: PASSED
   - Severity: Critical
4. **Overflows & Underflows**
   - Description: Whether the contract has general overflow or underflow vulnerabilities
   - Result: PASSED
   - Severity: Critical
5. **Reentrancy**
   - Description: Reentrancy is an issue when code can call back into your contract and change state, such as withdrawing ETHs.
   - Result: PASSED
   - Severity: Critical
6. **MONEY-Giving Bug**
   - Description: Whether the contract returns funds to an arbitrary address.
   - Result: PASSED
   - Severity: High

7. **Blackhole**

   - Description: Whether the contract locks ETH indefinitely: merely in without out.
   - Result: PASSED
   - Severity: High

8. **Unauthorized Self-Destruct**

   - Description: Whether the contract can be killed by any arbitrary address.
   - Result: PASSED
   - Severity: Medium

9. **Revert DoS**

   - Description: Whether the contract is vulnerable to DoS attack because of unexpected revert.
   - Result: PASSED
   - Severity: Medium

10. **Unchecked External Call**

    - Description: Whether the contract has any external call without checking the return value.
    - Result: PASSED
    - Severity: Medium

11. **Gasless Send**

    - Description: Whether the contract is vulnerable to gasless send.
    - Result: PASSED
    - Severity: Medium

12. **Send Instead of Transfer**

    - Description: Whether the contract uses send instead of transfer.
    - Result: PASSED
    - Severity: Medium

13. **Costly Loop**

- Description: Whether the contract has any costly loop which may lead to Out-Of-Gas exception.
- Result: PASSED
- Severity: Medium

14. **(Unsafe) Use of Untrusted Libraries**

- Description: Whether the contract use any suspicious libraries.
- Result: PASSED
- Severity: Medium

15. **(Unsafe) Use of Predictable Variables**

- Description: Whether the contract contains any randomness variable, but its value can be predicated.
- Result: PASSED
- Severity: Medium

16. **Transaction Ordering Dependence**

- Description: Whether the final state of the contract depends on the order of the transactions.
- Result: PASSED
- Severity: Medium

17. **Deprecated Uses**

- Description: Whether the contract use the deprecated tx.origin to perform the authorization.
- Result: PASSED
- Severity: Medium

Semantic Consistency Checks

- Description: Whether the semantic of the white paper is different from the implementation of the contract.
- Result: PASSED
- Severity: Critical

Conclusion

In this audit, we thoroughly analyzed Inuyasha's Smart Contract. The current code base is well organized but there are promptly some low-level Type issues found in the first phase of Smart Contract Audit.

Meanwhile, we need to emphasize that smart contracts as a whole are still in an early, but exciting stage of development. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

About eNebula Solutions

We believe that people have a fundamental need to security and that the use of secure solutions enables every person to more freely use the Internet and every other connected technology. We aim to provide security consulting service to help others make their solutions more resistant to unauthorized access to data & inadvertent manipulation of the system. We support teams from the design phase through the production to launch and surely after.

The eNebula Solutions team has skills for reviewing code in C, C++, Python, Haskell, Rust, Node.js, Solidity, Go, and JavaScript for common security vulnerabilities & specific attack vectors. The team has reviewed implementations of cryptographic protocols and distributed system architecture, including in cryptocurrency, blockchains, payments, and smart contracts. Additionally, the team can utilize various tools to scan code & networks and build custom tools as necessary.

Although we are a small team, we surely believe that we can have a momentous impact on the world by being translucent and open about the work we do.

For more information about our security consulting,
please mail us at – contact@enebula.in