Security Assessment

AmazingDoge

May 30th, 2022

Table of Contents

Summary

This report has been prepared for AmazingDoge to discover issues and vulnerabilities in the source code of the AmazingDoge project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	AmazingDoge
Platform	BSC
Language	Solidity
Codebase	https://bscscan.com/address/0x0EBc30459551858e81306d583025d12C7d795FA2
Commit	

Audit Summary

Delivery Date	May 30, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

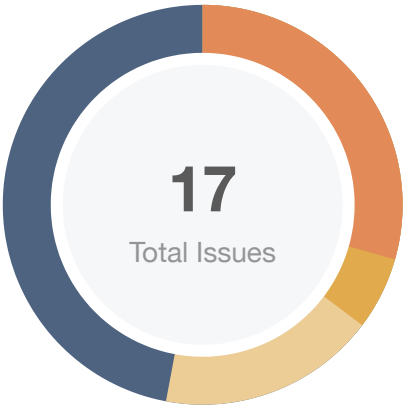
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	5	0	0	3	0	0	2
Medium	1	0	0	1	0	0	0
Minor	3	0	0	3	0	0	0
Informational	8	0	0	8	0	0	0
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
AMA	Amazing.sol	0f05ef4b13d2634fcf41d132f5aef952c2fda6e5b5a688f2edb6ab20f98f8ca4

Findings



Critical	**0** (0.00%)
Major	**5** (29.41%)
Medium	**1** (5.88%)
Minor	**3** (17.65%)
Informational	**8** (47.06%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
AMA-01	Potential Reentrancy Attack (Involving Ether)	Volatile Code	Major	Acknowledged
AMA-02	Centralization Risks In Amazing.sol	**Centralization / Privilege**	**Major**	Resolved
AMA-03	Initial Token Distribution	**Centralization / Privilege**	**Major**	Acknowledged
AMA-04	Centralized Risk In `addLiquidity`	**Centralization / Privilege**	**Major**	Resolved
AMA-05	Contract Gains Non-withdrawable BNB Via The `swapAndLiquify` Function	Logical Issue	Major	Acknowledged
AMA-06	Potential Sandwich Attacks	Logical Issue	Medium	Acknowledged
AMA-07	Third Party Dependencies	Control Flow	Minor	Acknowledged
AMA-08	Privilege Revoke	Logical Issue	Minor	Acknowledged
AMA-09	Lack Of Input Validation	Volatile Code	Minor	Acknowledged
AMA-10	Missing Emit Events	Coding Style	Informational	Acknowledged
AMA-11	Improper Usage Of `public` And `external` Type	Gas Optimization	Informational	Acknowledged
AMA-12	Variables That Could Be Declared As `constant`	Gas Optimization	Informational	Acknowledged

ID	Title	Category	Severity	Status
AMA-13	Unlocked Compiler Version	Language Specific	Informational	Acknowledged
AMA-14	Hardcode Address	Logical Issue	Informational	Acknowledged
AMA-15	Visibility Specifiers Missing	Language Specific	Informational	Acknowledged
AMA-16	Redundant Variable Initialization	Coding Style	Informational	Acknowledged
AMA-17	Discussion For Function `swapTokensForEth()`	Logical Issue	Informational	Acknowledged

AMA-01 | Potential Reentrancy Attack (Involving Ether)

Category	Severity	Location	Status
Volatile Code	Major	Amazing.sol: 619, 677, 680, 682~683, 688, 735~741, 751~758, 773	Acknowledged

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

External call(s)

File: Amazing.sol (Line 677, Function `Amazing._transfer`)

```
                swapAndLiquify(contractTokenBalance);
```

- This function call executes the following external call(s).
- In `Amazing.addLiquidity`,
  - `uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)`
- In `Amazing.swapTokensForEth`,
  - `uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp)`
- In `Amazing.transferToAddressETH`,
  - `recipient.transfer(amount)`
- This call sends Ether.

State variables written after the call(s)

File: Amazing.sol (Line 680, Function `Amazing._transfer`)

```
            _balances[sender] = _balances[sender].sub(amount, "Insufficient Balance");
```

File: Amazing.sol (Line 688, Function `Amazing._transfer`)

```
            _balances[recipient] = _balances[recipient].add(finalAmount);
```

File: Amazing.sol (Line 682-683, Function `Amazing._transfer`)

```
            uint256 finalAmount = (isExcludedFromFee[sender] ||
isExcludedFromFee[recipient]) ?
                                        amount : takeFee(sender, recipient, amount);
```

- This function call executes the following assignment(s).
- In `Amazing.takeFee`,
    - `_balances[address(this)] = _balances[address(this)].add(feeAmount)`

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

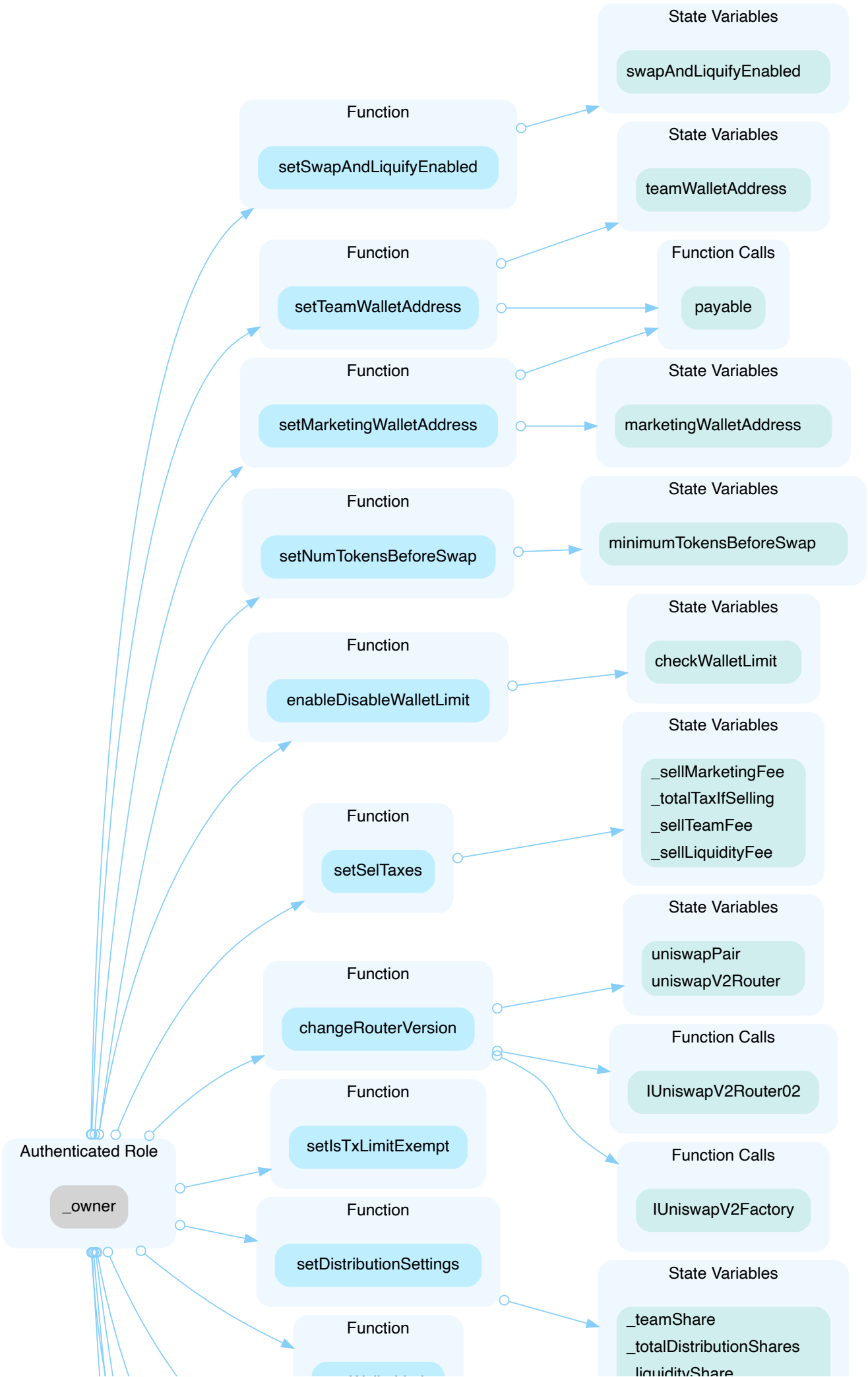
`[AmazingDoge]`: Issue acknowledged.

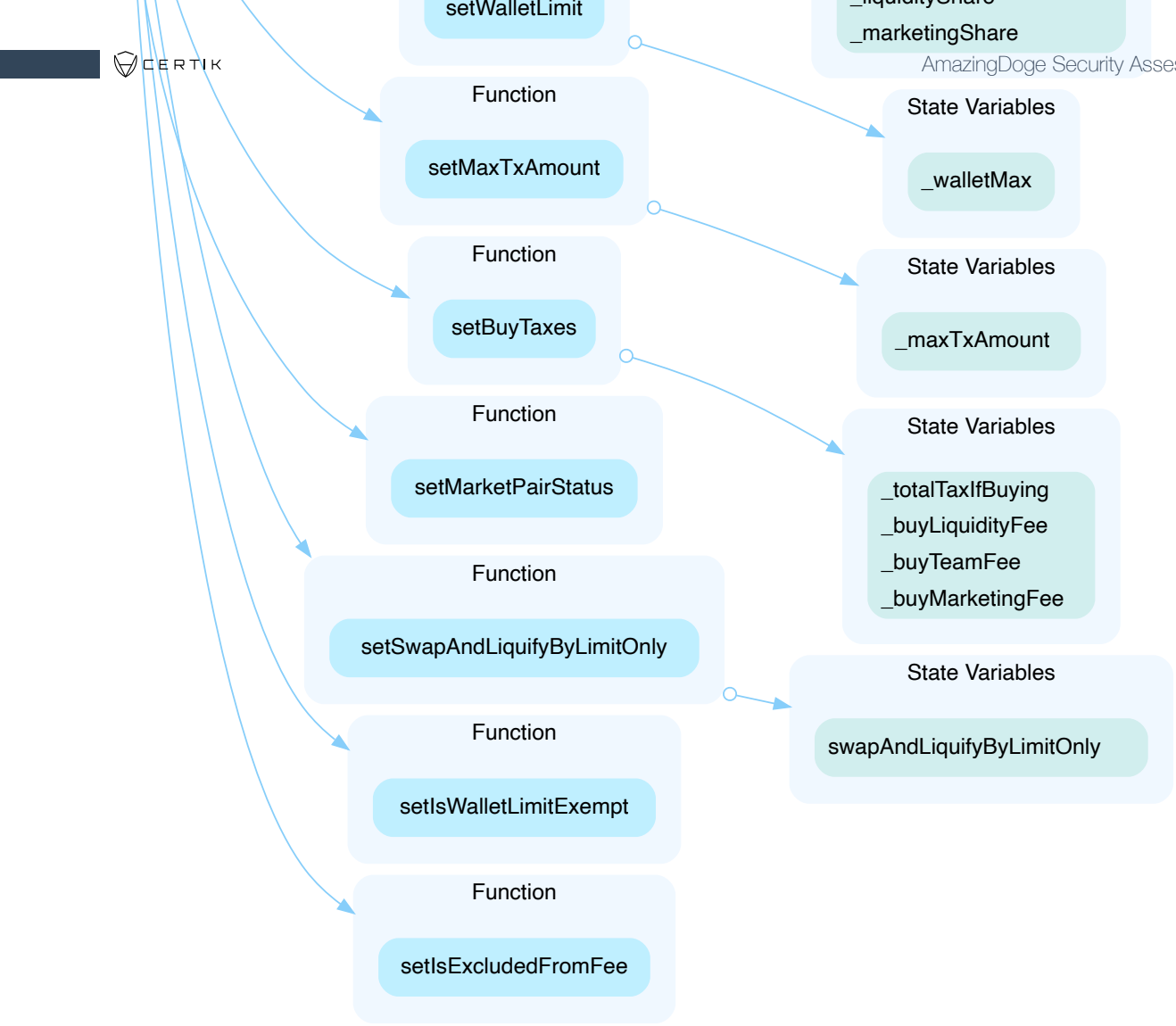
AMA-02 | Centralization Risks In Amazing.sol

Category	Severity	Location	Status
Centralization / Privilege	**Major**	Amazing.sol: 169, 174, 541, 545, 549, 553, 561, 569, 577, 581, 585, 589, 593, 597, 601, 605, 610, 622	Resolved

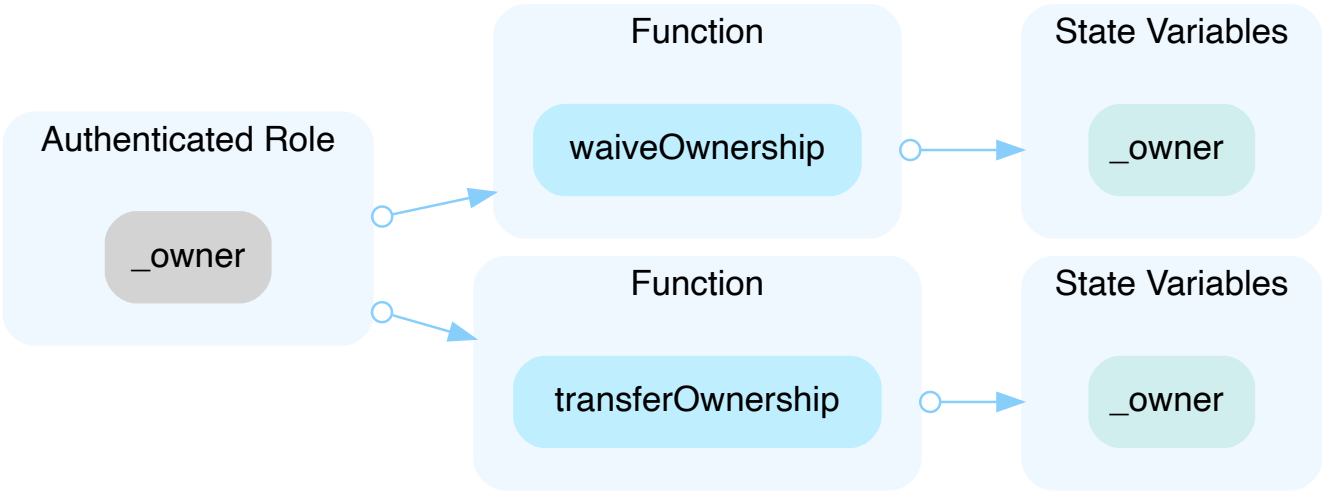
Description

In the contract `Amazing` the role `_owner` has authority over the functions shown in the diagram below.


State Variables
swapAndLiquifyEnabled
Function
setSwapAndLiquifyEnabled
State Variables
teamWalletAddress
Function
setTeamWalletAddress
Function Calls
payable
Function
setMarketingWalletAddress
State Variables
marketingWalletAddress
Function
setNumTokensBeforeSwap
State Variables
minimumTokensBeforeSwap
State Variables
checkWalletLimit
Function
enableDisableWalletLimit
State Variables
_sellMarketingFee
_totalTaxIfSelling
_sellTeamFee
_sellLiquidityFee
Function
setSelTaxes
State Variables
uniswapPair
uniswapV2Router
Function
changeRouterVersion
Function Calls
IUniswapV2Router02
Function Calls
IUniswapV2Factory
Function
setIsTxLimitExempt
Function
setDistributionSettings
State Variables
_teamShare
_totalDistributionShares
Function
Authenticated Role
_owner



In the contract `Ownable` the role `_owner` has authority over the functions shown in the diagram below.



Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and update the sensitive settings and execute sensitive functions of the project.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

`[CertiK]`: The team heeded the advice and resolved the finding by revoking the ownership in the deployment.

AMA-03 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	**Major**	Amazing.sol: 486	Acknowledged

Description

`1000000000000 * 10 ** 9` tokens were sent to the `owner` were sent to the contract when deploying the contract. This could be a centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process.

Alleviation

`[AmazingDoge]`: Token distribution adopts the method of public pre-sale, which is very transparent

AMA-04 | Centralized Risk In `addLiquidity`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	Amazing.sol: 756	Resolved

Description

The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function with the `to` address specified as `owner` for acquiring the generated LP tokens from the pair. As a result, over time the `owner` address will accumulate a significant portion of LP tokens. If the `owner` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND

- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

`[CertiK]`: The team heeded the advice and resolved the finding by revoking the ownership in the deployment.

AMA-05 | Contract Gains Non-withdrawable BNB Via The `swapAndLiquify` Function

Category	Severity	Location	Status
Logical Issue	Major	Amazing.sol: 702	Acknowledged

Description

The `swapAndLiquify` function converts half of the amount tokens to BNB. The other half of tokens and part of the converted BNB are deposited into the pair on DEX as liquidity. For every `swapAndLiquify` function call, a small amount of BNB is leftover in the contract. This is because the price of tokens drops after swapping the first half of tokens into BNBs, and the other half of tokens require less than the converted BNB to be paired with it when adding liquidity. The contract doesn't appear to provide a way to withdraw those BNB, and they will be locked in the contract forever.

Recommendation

It's not ideal that more and more BNB are locked into the contract over time. The simplest solution is to add a `withdraw` function in the contract to withdraw BNB. Other approaches that benefit the token holders can be:

- Distribute BNB to token holders proportional to the amount of token they hold.
- Use leftover BNB to buy back tokens from the market to increase the price of tokens.

Alleviation

`[AmazingDoge]`: Issue acknowledged.

AMA-06 | Potential Sandwich Attacks

Category	Severity	Location	Status
Logical Issue	Medium	Amazing.sol: 735, 751	Acknowledged

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by back running (after the transaction being attacked) a transaction to sell the asset.

The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- `uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens()`
- `uniswapV2Router.addLiquidityETH()`

Recommendation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

Alleviation

`[AmazingDoge]`: Issue acknowledged.

AMA-07 | Third Party Dependencies

Category	Severity	Location	Status
Control Flow	Minor	Amazing.sol: 464~465, 635	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party DEX. The scope of the audit would treat those 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties may be compromised and lead to assets being lost or stolen.

Recommendation

We understand that the business logic of the contract requires the interaction DEX for adding liquidity to the pair pool and swap tokens. We encourage the team to constantly monitor the status of those 3rd parties to mitigate negative outcomes when unexpected activities are observed.

Alleviation

`[AmazingDoge]`: Issue acknowledged.

AMA-08 | Privilege Revoke

Category	Severity	Location	Status
Logical Issue	Minor	Amazing.sol: 622	Acknowledged

Description

The above functions should revoke the privilege of the existing member. For example:

- exclude from `isWalletLimitExempt`
- exclude from `isMarketPair`
- update existing router's allowance to 0

Recommendation

We advise the client to revoke the privilege of the existing member.

Alleviation

`[AmazingDoge]`: Issue acknowledged.

AMA-09 | Lack Of Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	Amazing.sol: 558, 566, 574	Acknowledged

Description

The variables `_totalTaxIfBuying`,`_totalTaxIfSelling` and `_totalDistributionShares` should not exceed 100 respectively.

Recommendation

We advise the client to check the variables .

Alleviation

`[AmazingDoge]`: Issue acknowledged.

AMA-10 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	Amazing.sol: 541, 545, 549, 553, 561, 569, 577, 581, 585, 589, 593, 597, 601, 610, 622	Acknowledged

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

`[AmazingDoge]`: Issue acknowledged.

AMA-11 | Improper Usage Of `public` And `external` Type

Category	Severity	Location	Status
Gas Optimization	Informational	Amazing.sol: 174, 510, 514, 519, 528, 541, 549, 605, 610, 622, 644, 649	Acknowledged

Description

`public` functions that are never called by the contract could be declared as `external`. `external` functions are more efficient than `public` functions.

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

Alleviation

`[AmazingDoge]`: Issue acknowledged.

AMA-12 | Variables That Could Be Declared As `constant`

Category	Severity	Location	Status
Gas Optimization	Informational	Amazing.sol: 391, 392, 393, 397, 423	Acknowledged

Description

The linked variables could be declared as `constant` since these state variables are never modified.

Recommendation

We recommend to declare these variables as `constant`.

Alleviation

`[AmazingDoge]`: Issue acknowledged.

AMA-13 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	Amazing.sol: 10	Acknowledged

Description

The contract contains unlocked compiler versions. An unlocked compiler version in the contract's source code permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be difficult to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

It is a general practice to alternatively lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and in doing so be able to identify emerging ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

`[AmazingDoge]`: Issue acknowledged.

AMA-14 | Hardcode Address

Category	Severity	Location	Status
Logical Issue	Informational	Amazing.sol: 395~396, 461	Acknowledged

Description

There are many hardcode addresses in this codebase.

Recommendation

We advise changing to the correct address before the contract is deployed onto blockchain.

Alleviation

`[AmazingDoge]`: Issue acknowledged.

AMA-15 | Visibility Specifiers Missing

Category	Severity	Location	Status
Language Specific	Informational	Amazing.sol: 431	Acknowledged

Description

The linked variable declaration does not have a visibility specifier explicitly set.

Recommendation

Inconsistencies in the default visibility the Solidity compilers impose can cause issues in the functionality of the codebase. We advise that visibility specifier for the linked variable is explicitly set.

Alleviation

`[AmazingDoge]`: Issue acknowledged.

AMA-16 | Redundant Variable Initialization

Category	Severity	Location	Status
Coding Style	Informational	Amazing.sol: 433	Acknowledged

Description

All variable types within Solidity are initialized to their default "empty" value, which is usually their zeroed out representation. Particularly:

- `uint` / `int`: All `uint` and `int` variable types are initialized at `0`
- `address`: All `address` types are initialized to `address(0)`
- `byte`: All `byte` types are initialized to their `byte(0)` representation
- `bool`: All `bool` types are initialized to `false`
- `ContractType`: All contract types (i.e. for a given `contract ERC20 {}` its contract type is `ERC20`) are initialized to their zeroed out address (i.e. for a given `contract ERC20 {}` its default value is `ERC20(address(0))`)
- `struct`: All `struct` types are initialized with all their members zeroed out according to this table

Recommendation

We advise that the linked initialization statements are removed from the codebase to increase legibility.

Alleviation

`[AmazingDoge]`: Issue acknowledged.

AMA-17 | Discussion For Function `swapTokensForEth()`

Category	Severity	Location	Status
Logical Issue	Informational	Amazing.sol: 707	Acknowledged

Description

The function `swapAndLiquify` does not check the contract's current BNB balance before invoking the function `swapTokensForEth()`. In that case, the user can acquire exactly the amount of BNB that the swap creates, and not make the liquidity event include any BNB that has been manually sent to the contract

Recommendation

Consider adding a check of BNB balance before invoking the function `swapTokensForEth()`

Alleviation

`[AmazingDoge]`: Issue acknowledged.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM 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 how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being 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 may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/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 and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.