CERTIK

Security Assessment

DogeZilla Coin

Nov 30th, 2021

Table of Contents

Summary

This report has been prepared for DogeZilla Coin to discover issues and vulnerabilities in the source code of the DogeZilla Coin 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	DogeZilla Coin
Platform	BSC
Language	Solidity
Codebase	https://bscscan.com/address/0x7a565284572d03ec50c35396f7d6001252eb43b6
Commit	

Audit Summary

Delivery Date	Nov 30, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

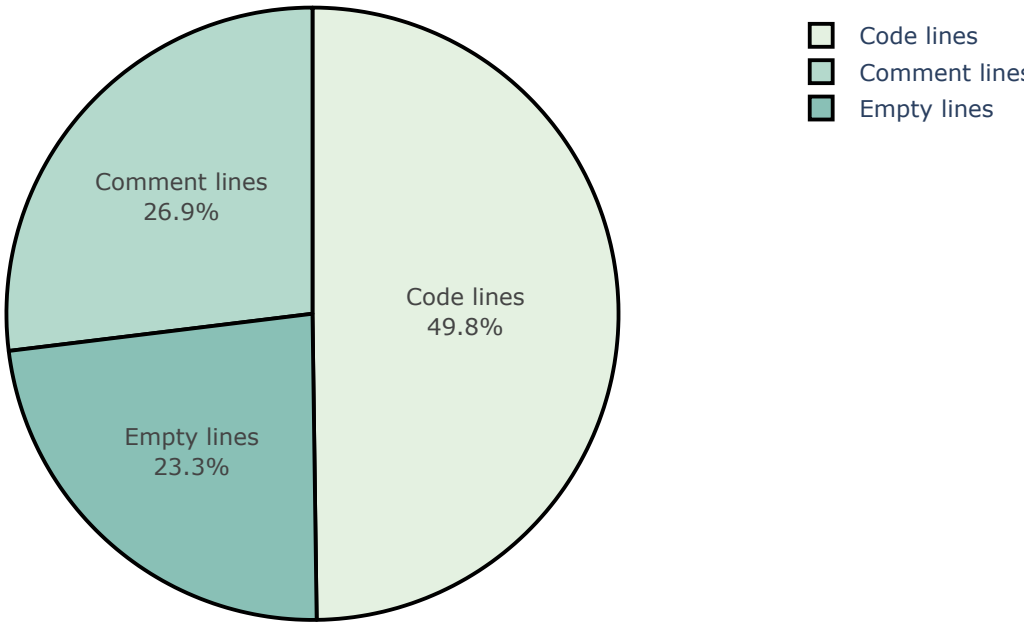
Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	3	0	0	3	0	0
Medium	0	0	0	0	0	0
Minor	3	0	0	3	0	0
Informational	5	0	0	5	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
DZD	DogeZilla.sol	791c35311e073198488ed31e4f758b83c21c8329f32a214ca1fc04df57284ea0

Diagrams

Source Line Chart



Findings

Critical	**0** (0.00%)
Major	**3** (27.27%)
Medium	**0** (0.00%)
Minor	**3** (27.27%)
Informational	**5** (45.45%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
DZC-01	Unlocked Compiler Version	Language Specific	Informational	Acknowledged
DZC-03	Too Many Digits	Coding Style	Informational	Acknowledged
DZD-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
DZD-02	WETH locked in contract via the `swapAndLiquify` function	Logical Issue	Major	Acknowledged
DZD-03	Variable `_rOwned[account]` Not Updated in Function `includeInReward()`	Control Flow	Major	Acknowledged
DZD-04	Unnecessary for loop in function `includeInReward`	Gas Optimization, Control Flow	Minor	Acknowledged
DZD-05	Incorrect error message in function `includeInReward`	Logical Issue	Minor	Acknowledged
DZD-06	Pull-Over-Push Pattern	Logical Issue	Minor	Acknowledged
DZD-07	Variables Could be declared as `constant`	Gas Optimization	Informational	Acknowledged
DZD-08	Potential Resource Exhaustion	Gas Optimization, Control Flow	Informational	Acknowledged
DZZ-01	Missing Emit Events	Coding Style	Informational	Acknowledged

DZC-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	projects/DogeZillaCoin/DogeZilla.sol (5e0de5b): 5	Acknowledged

Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract 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 an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.6.2` the contract should contain the following line:

```
pragma solidity 0.6.12;
```

DZC-03 | Too Many Digits

Category	Severity	Location	Status
Coding Style	Informational	projects/DogeZillaCoin/DogeZilla.sol (5e0de5b): 699, 719~720	Acknowledged

Description

Use of numeric literals with too many digits in the definition of variables:

```
_tTotal = 690000000000000000000 * 10**9
```

```
_maxTxAmount = 690000000000000000000 * 10**9;
```

```
numTokensSellToAddToLiquidity = 690000000000000000000 * 10**9;
```

Recommendation

We recommend using scientific notation modifying as below:

```
variableName = 69 * 10**21 * 10**9;
```

It increases the code readability.

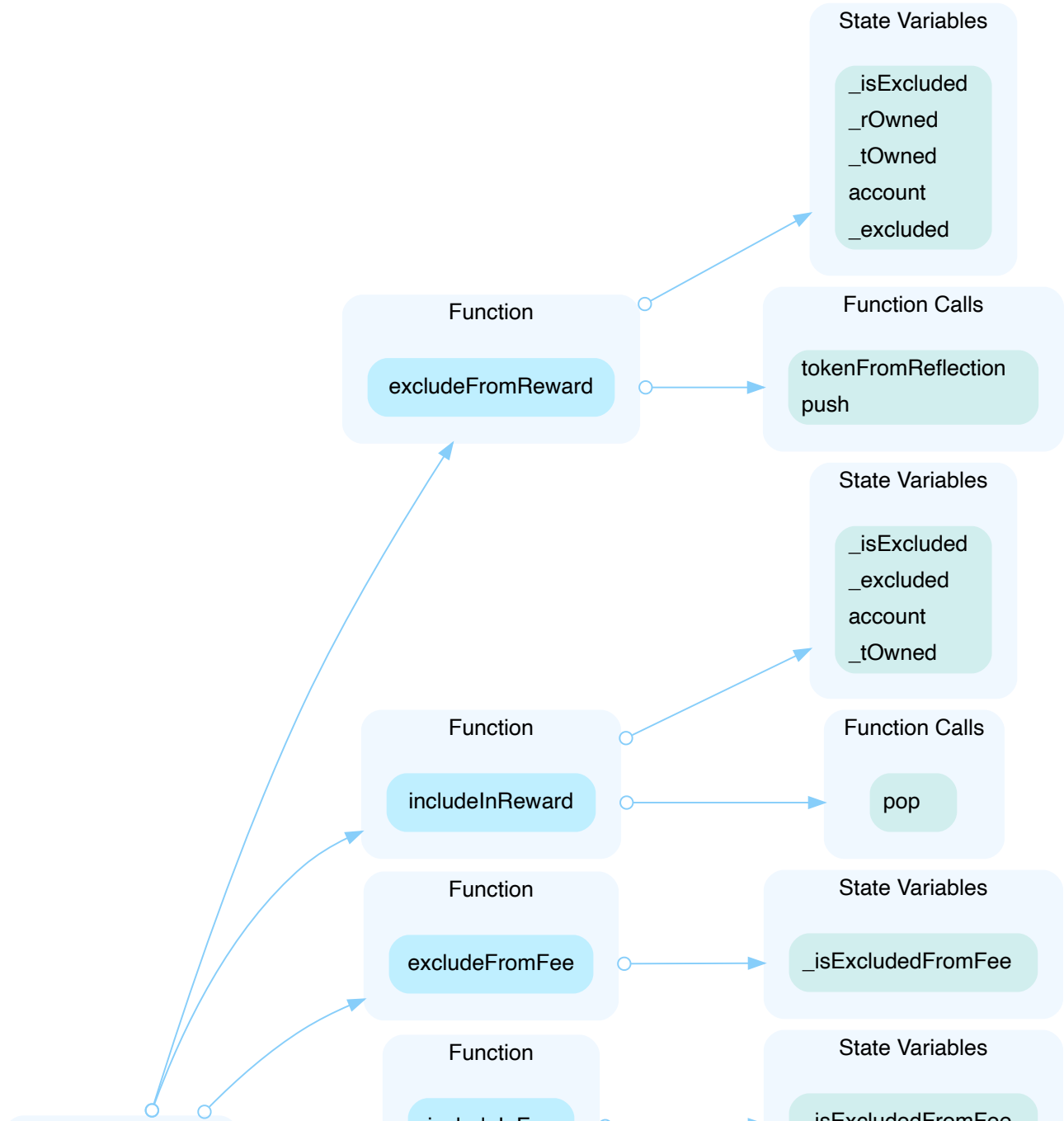
DZD-01 | Centralization Risk

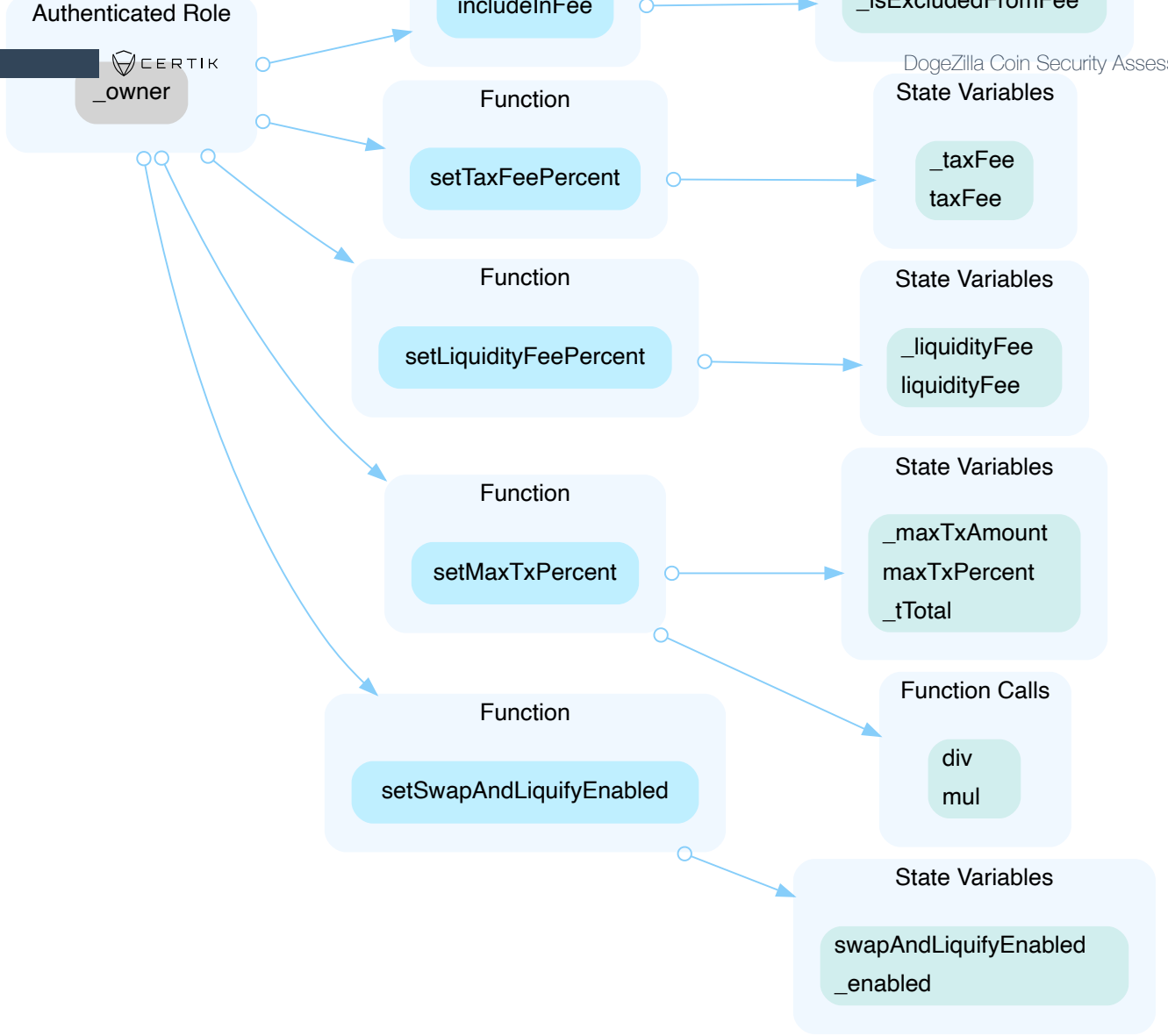
Category	Severity	Location	Status
Centralization / Privilege	Major	projects/DogeZillaCoin/DogeZilla.sol (5e0de5b): 839~847, 849~860, 872~874, 876~878, 880~882, 884~886, 888~892, 894~897, 435~438, 444~448, 455~460, 463~468	Acknowledged

Description

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

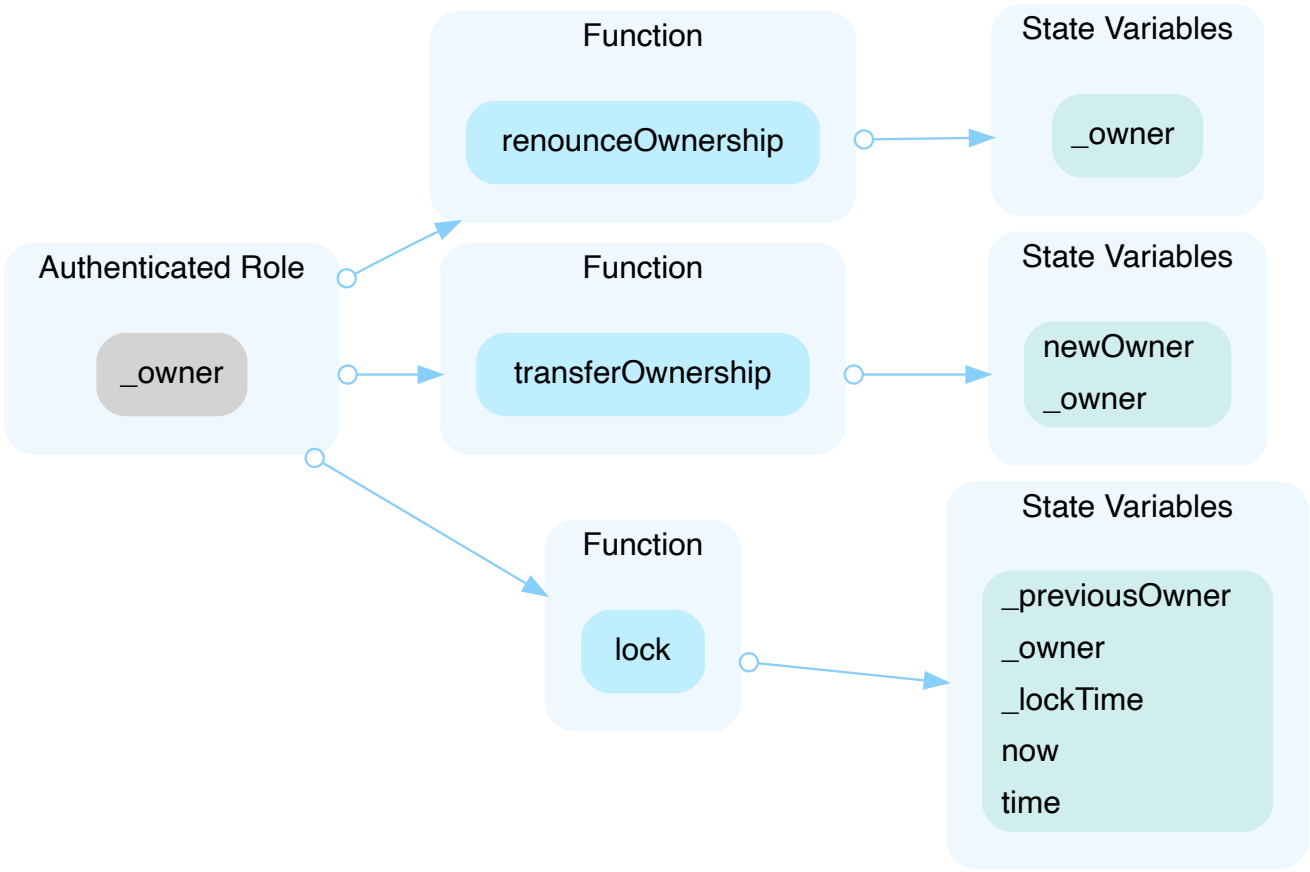
Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this.





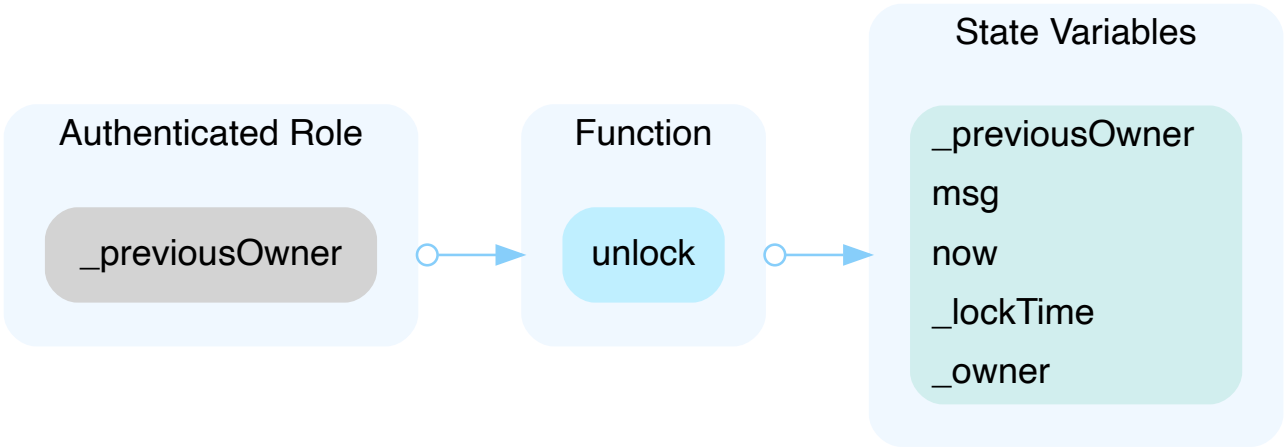
In the contract, `Ownable`, the role, `_owner`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this.



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

Any compromise to the privileged account which has access to `_previousOwner` may allow the hacker to take advantage of this.



Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

DZD-02 | WETH locked in contract via the `swapAndLiquify` function

Category	Severity	Location	Status
Logical Issue	Major	projects/DogeZillaCoin/DogeZilla.sol (5e0de5b): 1038~1059	Acknowledged

Description

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

Recommendation

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

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

DZD-03 | Variable `_rOwned[account]` Not Updated in Function `includeInReward()`

Category	Severity	Location	Status
Control Flow	Major	projects/DogeZillaCoin/DogeZilla.sol (5e0de5b): 849~860	Acknowledged

Description

The function below has a known bug.

```
    function includeInReward(address account) external onlyOwner() {
        require(_isExcluded[account], "Account is already excluded");
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_excluded[i] == account) {
                _excluded[i] = _excluded[_excluded.length - 1];
                _tOwned[account] = 0;
                _isExcluded[account] = false;
                _excluded.pop();
                break;
            }
        }
    }
```

Variable `_rOwned[account]` is not updated in the function `includeInReward()`, which will make the accounts included siphon off the tokens out of the balances of all token holders.

Details of this finding can be seen in this article from Pera Finance: Link

Recommendation

We recommend updating `_rOwned[account]` before setting `_tOwned[account]` to 0.

Sample code:

```
    function includeInReward(address account) external onlyOwner() {
        require(_isExcluded[account], "Account is already excluded");
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_excluded[i] == account) {
                _excluded[i] = _excluded[_excluded.length - 1];
                _rOwned[account] = _tOwned[account].mul(_getRate());  // update
_rOwned[account]
                _tOwned[account] = 0;
                _isExcluded[account] = false;
                _excluded.pop();
                break;
```

```
            }
        }
    }
```

DZD-04 | Unnecessary for loop in function `includeInReward`

Category	Severity	Location	Status
Gas Optimization, Control Flow	Minor	projects/DogeZillaCoin/DogeZilla.sol (5e0de5b): 851~858	Acknowledged

Description

The variable `_isExcluded` stores the mapping from an address to a boolean variable that indicates if the address is excluded. As a result, it is unnecessary to loop through the _excluded array to see if a particular address is in the array. The require statement already guarantees that the account is excluded.

```
require(_isExcluded[account], "Account is already excluded");
for (uint256 i = 0; i < _excluded.length; i++) {
    if (_excluded[i] == account) {
        _excluded[i] = _excluded[_excluded.length - 1];
        _tOwned[account] = 0;
        _isExcluded[account] = false;
        _excluded.pop();
        break;
    }
}
```

Recommendation

Remove the unnecessary for loop to save gas, and update the values of _tOwned[account], _rOwned[account], and _isExcluded[account] directly.

DZD-05 | Incorrect error message in function `includeInReward`

Category	Severity	Location	Status
Logical Issue	Minor	projects/DogeZillaCoin/DogeZilla.sol (5e0de5b): 850	Acknowledged

Description

The error message in `require(_isExcluded[account], "Account is already excluded")` does not describe the error correctly.

Recommendation

The message "Account is already excluded" can be changed to "Account is not excluded" .

DZD-06 | Pull-Over-Push Pattern

Category	Severity	Location	Status
Logical Issue	Minor	projects/DogeZillaCoin/DogeZilla.sol (5e0de5b): 444~448	Acknowledged

Description

The change of `owner` by function `transferOwnership()` overrides the previously set `owner` with the new one without guaranteeing the new `owner` is able to actuate transactions on-chain.

Recommendation

We advise the pull-over-push pattern to be applied here whereby a new `owner` is first proposed and consequently needs to accept the `owner` status ensuring that the account can actuate transactions on-chain.

The following code snippet can be taken as a reference:

```
address public potentialAdmin;

function transferAdmin(address pendingAdmin) external onlyAdmin {
    require(pendingAdmin != address(0), "potential admin can not be the zero address.")
    potentialAdmin = pendingAdmin;
    emit AdminNominated(pendingAdmin);
}

function acceptAdmin() external {
    require(msg.sender == potentialAdmin, 'You must be nominated as potential admin
before you can accept administer role');
    admin = potentialAdmin;
    potentialAdmin = address(0);
    emit AdminChanged(admin)
}
```

DZD-07 | Variables Could be declared as `constant`

Category	Severity	Location	Status
Gas Optimization	Informational	projects/DogeZillaCoin/DogeZilla.sol (5e0de5b): 699	Acknowledged

Description

The linked variables could be declared as `constant` since this state variables is never modified.

Recommendation

We recommend to declare these variables as `constant`.

DZD-08 | Potential Resource Exhaustion

Category	Severity	Location	Status
Gas Optimization, Control Flow	Informational	projects/DogeZillaCoin/DogeZilla.sol (5e0de5b): 933~943	Acknowledged

Description

There is no explicit limit on the possible sizes of the `excluded` array.

If this array is too big, calling `_getCurrentSupply()` might consume all the possible gas, leading to an out-of-gas error.

In this case, the caller has to manually adjust the array size and execute the function again.

Recommendation

It is recommended to setup a max size restriction for the `_excluded` array. Alternatively, is possible to set a start and stop limit on the number of iteration the function `_getCurrentSupply()` is allow to do, in order to make possible to split an eventual call which would require too much gas if exectued in a single instance.

DZZ-01 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	projects/DogeZillaCoin/DogeZilla.sol (5e0de5b): 880~882, 881, 884~886, 888~892	Acknowledged

Description

The functions that affects the status of sensitive variables should be able to emit events as notifications.

Recommendation

Emit an event for critical parameter changes.

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.

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.