Smart Contract Security Audit Report

For

Dora

Date Issued: Apr.15, 2024

Version: v2.0

Confidentiality Level: Public

Contents

1 Abstract

This report was prepared for Dora smart contract to identify issues and vulnerabilities in its smart contract source code. A thorough examination of Dora smart contracts was conducted through timely communication with Dora, static analysis using multiple audit tools and manual auditing of their smart contract source code.

The audit process paid particular attention to the following considerations.

- A thorough review of the smart contract logic flow
- Assessment of the code base to ensure compliance with current best practice and industry standards
- Ensured the contract logic met the client's specifications and intent
- Internal vulnerability scanning tools tested for common risks and writing errors
- Testing smart contracts for common attack vectors
- Test smart contracts for known vulnerability risks
- Conduct a thorough line-by-line manual review of the entire code base

As a result of the security assessment, issues ranging from critical to informational were identified. We recommend that these issues are addressed to ensure a high level of security standards and industry practice. The recommendations we made could have better served the project from a security perspective.

- Enhance general coding practices to improve the structure of the source code.
- Provide more comments for each function to improve readability.
- Provide more transparency of privileged activities once the agreement is in place.

2 Overview

2.1 Project Summary

Project Summary	Project Information
Name	Dora
Start date	Apr.4, 2024
End date	Apr.15, 2024
Contract type	Token ,DeFi
Language	Solidity
Code	Dora.sol

2.2 Report HASH

Platform	Address
Pego Network	https://scan.pego.network/address/0x332E657DE4383F9d17C4F09d12Bed8F835141319/contracts#address-tabs

3 Project contract details

3.1 Contract Overview

Ownable.sol

The code defines a basic "ownership model" that contains mechanisms for transferring and relinquishing ownership. Through the onlyOwner modifier, it ensures that only the current owner of the contract can call a specific function. This mode is useful when you need to restrict access to certain sensitive operations of the contract, such as when upgrading the contract or changing key parameters.

Dora.sol

This code defines a token contract Dora based on the ERC20 standard, which includes enhanced features such as automatic liquidity addition, transaction fee management (including marketing, team support and token destruction fees), and prevention of malicious address transaction hacking. List mechanism. The contract uses the Uniswap protocol to automatically handle liquidity issues, support project development and encourage token holdings by charging transaction fees, and also provides flexible permission control, allowing contract owners to perform key operations, such as adjusting fee structures, managing blacklists, and Liquidity pool strategies, etc.

4 Audit results

4.1 Key messages

ID	Title	Severity	Status
01	Unlimited handling fee size	Low	Fixed
02	There is a blacklist address and cannot be traded	Informational	Fixed
03	Be cautious when changing routing and trading pair contracts	Informational	Fixed
04	When the startTx variable is not updated, some addresses will not be able to trade	Informational	Fixed
05	Contract privileged roles can set multiple key variables in the contract	Low	Fixed
06	Transfer sender and recipient addresses can be the same	Low	Confirmed

4.2 Audit details

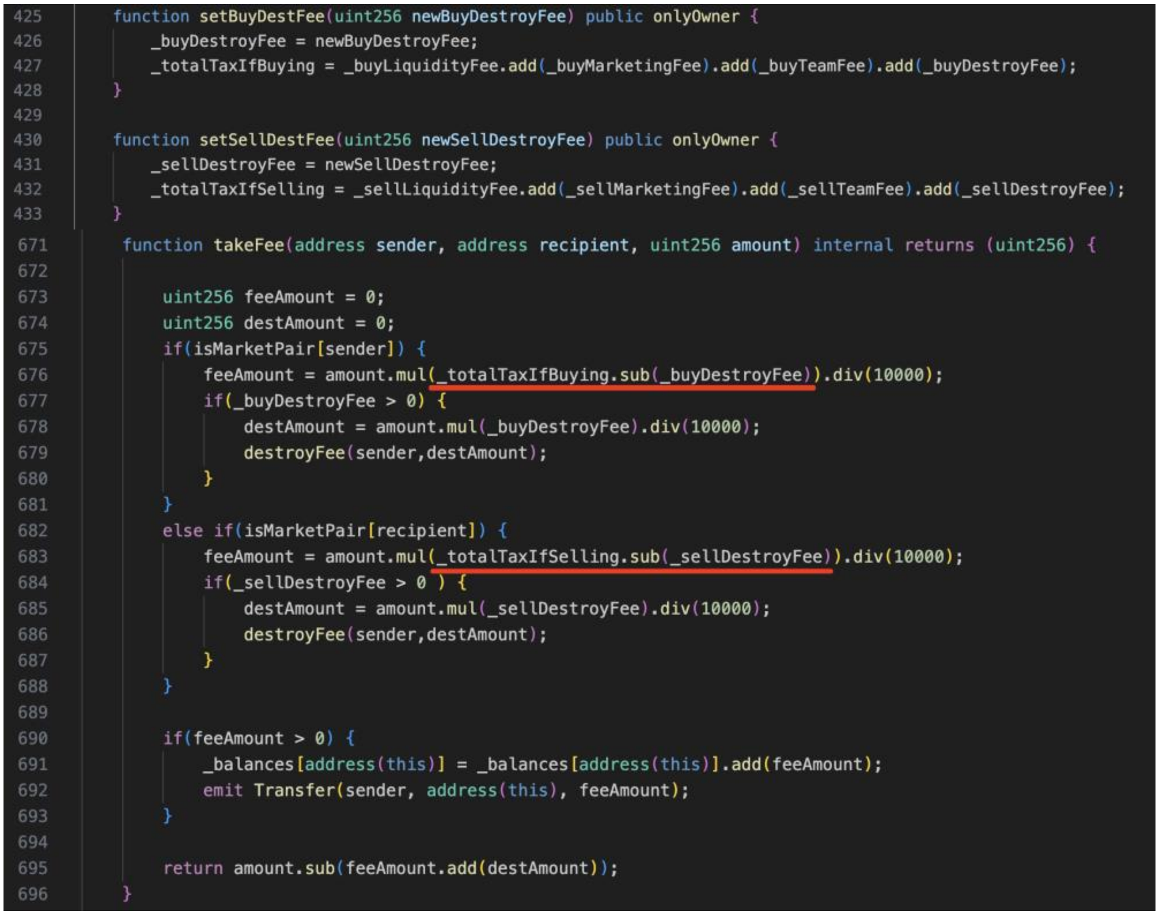
4.2.1 Unlimited handling fee size

ID	Severity	Location	Status
01	Low	Dora.sol: 425, 433;671,696	Confirmed

Description

There is no limit on the handling fees of _buyDestroyFee and _totalTaxIfBuying. If the handling fee rate is greater than 1, it will cause the user's transfer to exceed the limit. In addition, if the handling fee rate is set too large, it will also cause the user's principal loss. The current rate is set by the privileged role. Privileges The role is EOA.

Code location:



```
425     function setBuyDestFee(uint256 newBuyDestroyFee) public onlyOwner {
426         _buyDestroyFee = newBuyDestroyFee;
427         _totalTaxIfBuying = _buyLiquidityFee.add(_buyMarketingFee).add(_buyTeamFee).add(_buyDestroyFee);
428     }
429
430     function setSellDestFee(uint256 newSellDestroyFee) public onlyOwner {
431         _sellDestroyFee = newSellDestroyFee;
432         _totalTaxIfSelling = _sellLiquidityFee.add(_sellMarketingFee).add(_sellTeamFee).add(_sellDestroyFee);
433     }
671     function takeFee(address sender, address recipient, uint256 amount) internal returns (uint256) {
672
673         uint256 feeAmount = 0;
674         uint256 destAmount = 0;
675         if(isMarketPair[sender]) {
676             feeAmount = amount.mul(_totalTaxIfBuying.sub(_buyDestroyFee)).div(10000);
677             if(_buyDestroyFee > 0) {
678                 destAmount = amount.mul(_buyDestroyFee).div(10000);
679                 destroyFee(sender,destAmount);
680             }
681         }
682         else if(isMarketPair[recipient]) {
683             feeAmount = amount.mul(_totalTaxIfSelling.sub(_sellDestroyFee)).div(10000);
684             if(_sellDestroyFee > 0 ) {
685                 destAmount = amount.mul(_sellDestroyFee).div(10000);
686                 destroyFee(sender,destAmount);
687             }
688         }
689
690         if(feeAmount > 0) {
691             _balances[address(this)] = _balances[address(this)].add(feeAmount);
692             emit Transfer(sender, address(this), feeAmount);
693         }
694
695         return amount.sub(feeAmount.add(destAmount));
696     }
```

Recommendation

It is recommended to set a fee rate limit to avoid accidents caused by manipulation of privileged roles.

Status

Fixed.

It has been officially confirmed and the privileged role has been destroyed.

https://scan.pego.network/tx/0x4cebcd10dcdacf39c1b8b713da67e339e550d884001d145e753fca294a82a29a

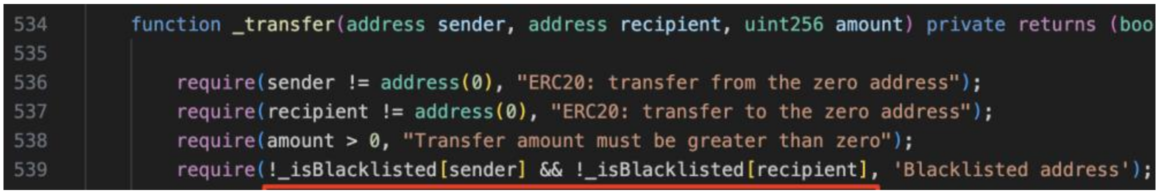
4.2.2 There is a blacklist address and cannot be traded

ID	Severity	Location	Status
02	Informational	Dora.sol: 534, 539	Confirmed

Description

The _isBlacklisted attribute is a blacklist address, which is set by the EOA privileged role. If the privileged role is maliciously manipulated, the specified address will not be able to trade.

Code location:



```
534     function _transfer(address sender, address recipient, uint256 amount) private returns (boo
535
536         require(sender != address(0), "ERC20: transfer from the zero address");
537         require(recipient != address(0), "ERC20: transfer to the zero address");
538         require(amount > 0, "Transfer amount must be greater than zero");
539         require(!_isBlacklisted[sender] && !_isBlacklisted[recipient], 'Blacklisted address');
```

Recommendation

It is recommended that privileged roles be managed using multi-signature.

Status

Fixed.

It has been officially confirmed and the privileged role has been destroyed.

https://scan.pego.network/tx/0x4cebcd10dcdacf39c1b8b713da67e339e550d884001d145e753fca294a82a29a

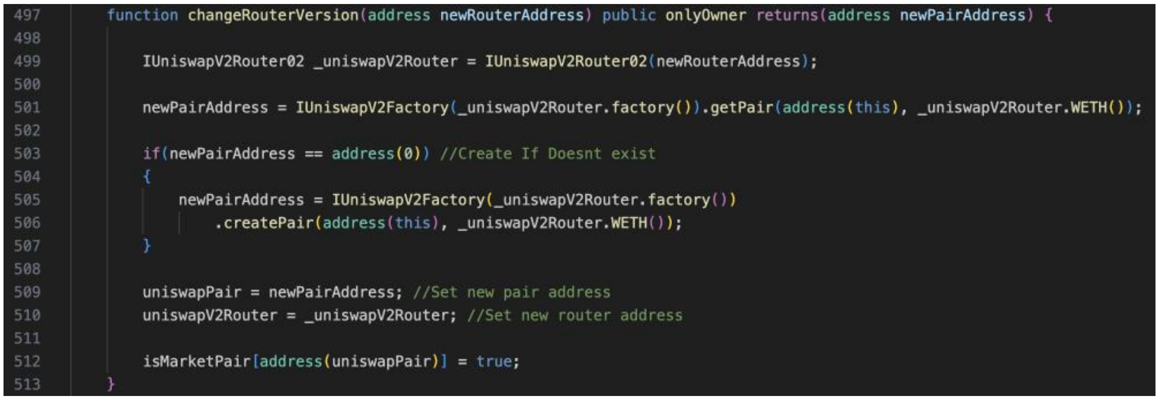
4.2.3 Be cautious when changing routing and trading pair contracts

ID	Severity	Location	Status
03	Informational	Dora.sol: 497,513	Confirmed

Description

Replacing routers and trading pair contracts may have an impact on existing liquidity. If there is insufficient liquidity in the new pairing contract, it may have a negative impact on the market performance of the token.

Code location:



```
497     function changeRouterVersion(address newRouterAddress) public onlyOwner returns(address newPairAddress) {
498
499         IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(newRouterAddress);
500
501         newPairAddress = IUniswapV2Factory(_uniswapV2Router.factory()).getPair(address(this), _uniswapV2Router.WETH());
502
503         if(newPairAddress == address(0)) //Create If Doesnt exist
504         {
505             newPairAddress = IUniswapV2Factory(_uniswapV2Router.factory())
506                 .createPair(address(this), _uniswapV2Router.WETH());
507         }
508
509         uniswapPair = newPairAddress; //Set new pair address
510         uniswapV2Router = _uniswapV2Router; //Set new router address
511
512         isMarketPair[address(uniswapPair)] = true;
513     }
```

Recommendation

It is recommended that privileged roles use multi-signature management, and privileged roles need to be cautious when performing operations.

Status

Fixed.

It has been officially confirmed and the privileged role has been destroyed.

https://scan.pego.network/tx/0x4cebcd10dcdacf39c1b8b713da67e339e550d884001d145e753fca294a82a29a

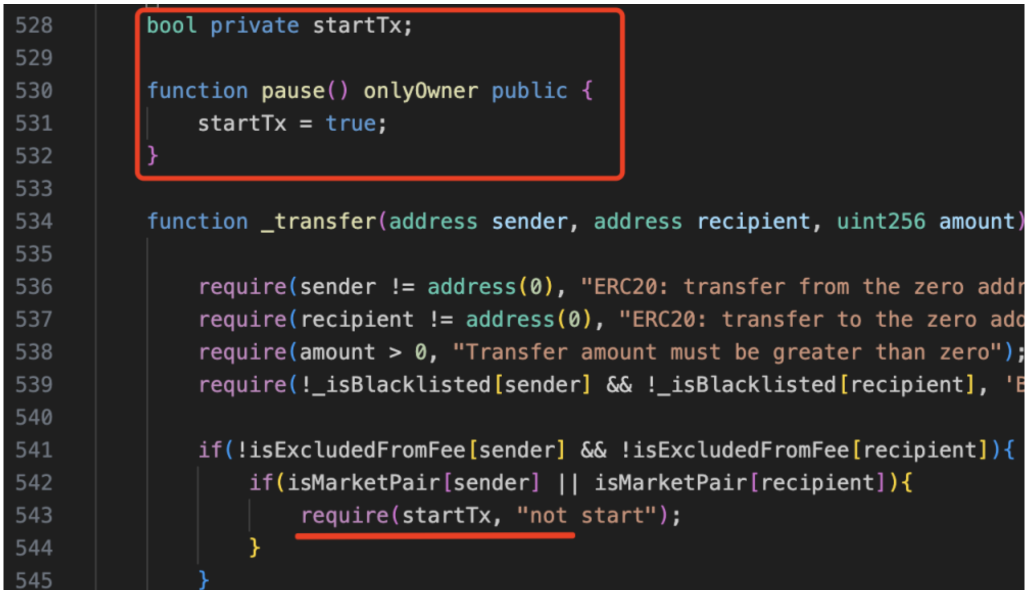
4.2.4 When the startTx variable is not updated, some addresses will not be able to trade

ID	Severity	Location	Status
04	Informational	Dora.sol: 528, 545	Confirmed

Description

When transferring money using the _transfer method, if both the sender and receiver addresses are exempt from handling fees, and any address is a trading pair, no transaction can be made without startTx. The exemption of handling fees and trading pairs are privileged role settings.

Code location:



```
528        bool private startTx;
529
530        function pause() onlyOwner public {
531            startTx = true;
532        }
533
534        function _transfer(address sender, address recipient, uint256 amount)
535
536            require(sender != address(0), "ERC20: transfer from the zero addr
537            require(recipient != address(0), "ERC20: transfer to the zero add
538            require(amount > 0, "Transfer amount must be greater than zero");
539            require(!_isBlacklisted[sender] && !_isBlacklisted[recipient], 'E
540
541            if(!isExcludedFromFee[sender] && !isExcludedFromFee[recipient]){
542                if(isMarketPair[sender] || isMarketPair[recipient]){
543                    require(startTx, "not start");
544                }
545            }
```

Recommendation

It is recommended that privileged roles be managed using multi-signature.

Status

Fixed.

It has been officially confirmed and the privileged role has been destroyed.

https://scan.pego.network/tx/0x4cebcd10dcdacf39c1b8b713da67e339e550d884001d145e753fca294a82a29a

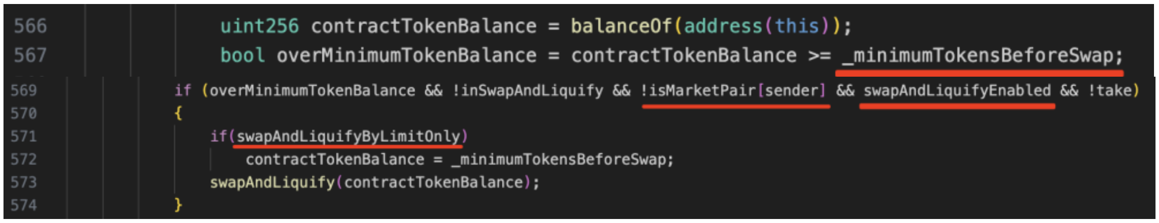
4.2.5 Contract privileged roles can set multiple key variables in the contract

ID	Severity	Location	Status
05	Low	Dora.sol: 566,574	Confirmed

Description

When transferring money using the _transfer method, multiple key variables used in the logic can be changed by privileged roles.

Code location:



```
566            uint256 contractTokenBalance = balanceOf(address(this));
567            bool overMinimumTokenBalance = contractTokenBalance >= _minimumTokensBeforeSwap;
569            if (overMinimumTokenBalance && !inSwapAndLiquify && !isMarketPair[sender] && swapAndLiquifyEnabled && !take)
570            {
571                if(swapAndLiquifyByLimitOnly)
572                    contractTokenBalance = _minimumTokensBeforeSwap;
573                swapAndLiquify(contractTokenBalance);
574            }
```

Recommendation

It is recommended that privileged roles be managed using multi-signatures to prevent EOA addresses from being manipulated and affecting the market.

Status

Fixed.

It has been officially confirmed and the privileged role has been destroyed.

https://scan.pego.network/tx/0x4cebcd10dcdacf39c1b8b713da67e339e550d884001d145e753fca294a82a29a

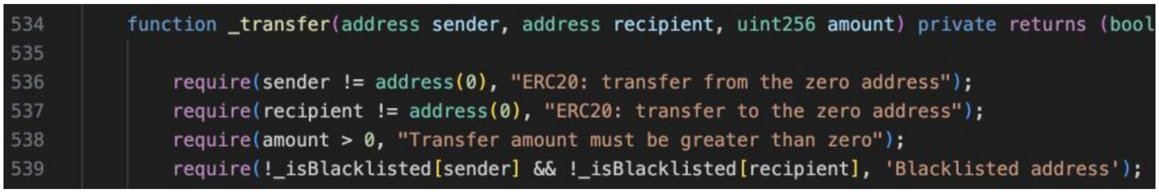
4.2.6 Transfer sender and recipient addresses can be the same

ID	Severity	Location	Status
06	Low	Dora.sol: 534, 539	Confirmed

Description

Since the _transfer method will make multiple different and identical judgments on the sender and recipient when transferring money, if the two addresses are consistent, this will cause some restrictions to be bypassed.

Code location:



```
534     function _transfer(address sender, address recipient, uint256 amount) private returns (bool
535
536         require(sender != address(0), "ERC20: transfer from the zero address");
537         require(recipient != address(0), "ERC20: transfer to the zero address");
538         require(amount > 0, "Transfer amount must be greater than zero");
539         require(!_isBlacklisted[sender] && !_isBlacklisted[recipient], 'Blacklisted address');
```

Recommendation

It is recommended that in the _transfer method, determine whether the sender and receiver have different addresses.

Status

Confirmed.

5 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.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

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.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in-storage one.

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.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Disclaimer

This report is issued in response to facts that occurred or existed prior to the issuance of this report, and liability is assumed only on that basis.
Shield Security cannot determine the security status of this program and assumes no responsibility for facts occurring or existing after the date of this report. The security audit analysis and other content in this report is based on documents and materials provided to Shield Security by the information provider through the date of the insurance report. in Shield Security's opinion. The information provided is not missing, falsified, deleted or concealed. If the information provided is missing, altered, deleted, concealed or not in accordance with the actual circumstances, Shield Security shall not be liable for any loss or adverse effect resulting therefrom. shield Security will only carry out the agreed security audit of the security status of the project and issue this report. shield Security is not responsible for the background and other circumstances of the project. Shield Security is not responsible for the background and other circumstances of the project.