QuillAudits

Audit Report
July, 2021

DCTDAO

Contents

Scope of Audit

The scope of this audit was to analyze and document the DCTDAO ERC20GenericNativeSwap smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	1	0	0	2
Closed	1	0	0	0

Introduction

During the period of June 08, 2021 to June 10, 2021 - QuillAudits Team performed a security audit for DCTDAO smart contracts.

The code for the audit was taken from following the official link:
https://github.com/DCTDAO/chainbridge-solidity/blob/master/contracts/handlers/ERC20GenericNativeSwap.sol
Commit hash: 72ff1b4

The ERC20GenericNativeSwap.sol was used for the **final review**
(MD5 (ERC20GenericNativeSwap.sol) =
45fa3b918bfddc970963fbf838e34f99)

https://drive.google.com/file/d/1h02KS-TCl8GgjQCmj3DW3lFdtcqGzUL_/view?usp=sharing

Issues Found – Code Review / Manual Testing

High severity issues

1. Integer Overflow and Underflow

Line	Code
144-210	function executeProposal(bytes32 resourceID, bytes calldata data) external override onlyBridge

Description
The overflow/underflow has been found in this contract which happens in the **executeProposal** function.

An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type. For instance, if a number is stored in the uint8 type, it means that the number is stored in an 8 bits unsigned number ranging from 0 to 2^8-1. In computer programming, an integer overflow occurs when an arithmetic operation attempts to create a numeric value that is outside of the range that can be represented with a given number of bits – either larger than the maximum or lower than the minimum representable value.

Remediation
It is recommended to use vetted safe math libraries for arithmetic operations consistently throughout the smart contract system.

References
Ethereum Smart Contract Best Practices - Integer Overflow and Underflow

Exploit POC
The overflow/underflow happens when an attacker can manipulate the “bytes calldata data” variable.

Overflow

The “bytes calldata data” parameter is controllable by any callers, therefore, a malicious user can manipulate this parameter to trigger the overflow. The following value for “byes calldata data” can be used to trigger the vulnerability.

```
0x0000000000000000000000000000000000000000000000000000000000000000
0000100000000000000000000000000000000000000000000000000000000000
0004E20800000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
000000000004
```

At line 54:

```
(lenFromServerGeneric, amount, swapPer, lenRecipientAddress) = abi.decode(data,
(uint256, uint256, uint256, uint256));
```

We can see that the above “bytes calldata data” represents the subsequence of those state variables at line 54, including lenFromServerGeneric, amount, swapPer, lenRecipientAddress. Therefore,

```
lenFromServerGeneric =
0x0000000000000000000000000000000000000000000000000000000000000001
```

```
Amount =
0x0000000000000000000000000000000000000000000000000000000000004E20
```

```
swapPer =
0x8000000000000000000000000000000000000000000000000000000000000000
```

```
lenRecipientAddress =
0x0000000000000000000000000000000000000000000000000000000000000004
```

At line 166, 167:

```
swapTokens = (amount/10000)*swapPer;
amount = amount - swapTokens;
```

Based on the values the attacker can insert above, the swapTokens is exposed to the overflow vulnerability since an attacker has inserted a very big value of the uint256 which is equivalent to 0x8000000000000000000000000000000000000000000000000000000000000000

This value triggers the overflow for the **swapTokens**

```
swapTokens = (20000/10000)*0x8000000000000000000000000000000000000000000000000000000000000000 = Zero
```

Note: 0x8000000000000000000000000000000000000000000000000000000000000000 is very big number of uint256

In the end, the **swapTokens** reaches the maximum size of uint256 and returns back to **Zero**.

Underflow

Similar to the overflow vulnerability, an attacker can manipulate the value of “bytes calldata data” and cause the underflow to the operation at line 167 in this case.

```
0x0000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000000000000000000004E2000000000000000000000000000000000000000000000000000000000004E200000000000000000000000000000000000000000000000000000000000004
```

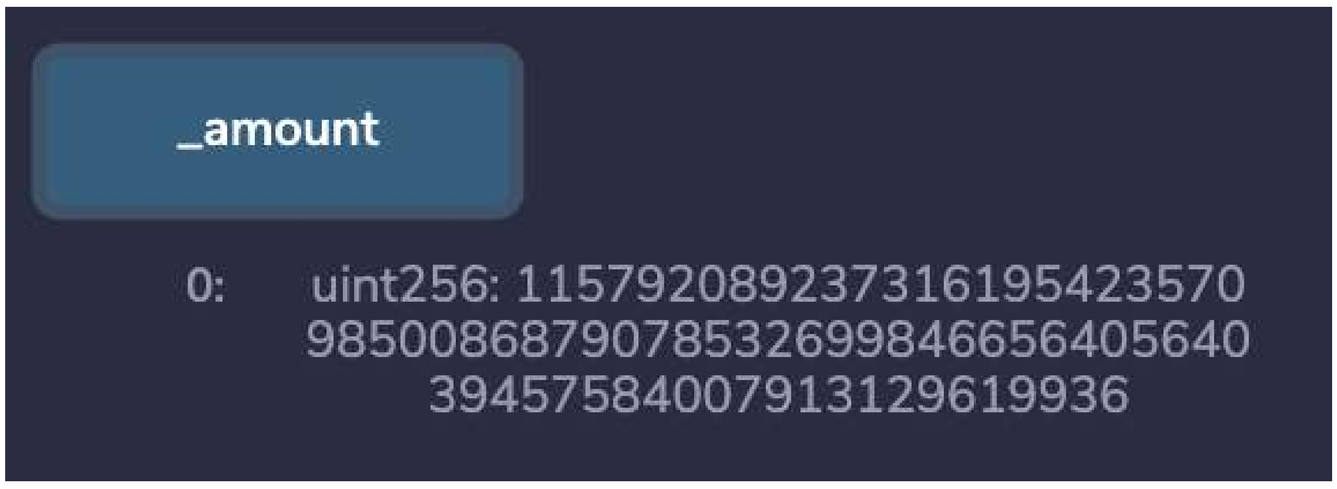
In this case, we can see the subtraction at line 167 does not use any type of check to prevent the underflow.

```
amount = amount - swapTokens;
```

If the swapTokens is bigger than the amount, the amount will reach the minimum size of the uin256 and become 2^256-1.

Based on the “bytes calldata data” inserted by an attacker and combine with the operation at line 166, we have:

```
swapTokens = (20000/10000)*20000 = 40000
amount = 20000-40000 = 2^256-1
```



The amount has reached the minimum size of the uint256 and returned to the maximum value of the uin256, which is 2^256-1.

Status: Fixed

2. Divide before multiply

Line	Code
166	swapTokens = (amount/10000)*swapPer;

Description

The swapTokens value is always Zero if the amount is less than 10000.

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.

Remediation

The amount should be checked before division.

Status: Acknowledged by the Auditee

Auditee asserted that this one is ok because their backend and frontend do not allow such small values.

Medium severity issues

No issues were found

Low level severity issues

No issues were found

Informational

3. Different pragma directives are used

Version used:
0.6.12
>=0.5.0
>=0.6.0
>=0.6.0<0.8.0
>=0.6.2
>=0.6.2<0.8.0

Description
It is detected when different Solidity versions are used.

Remediation
Use one Solidity version for all contracts. The pragma version 0.6.12 is safe to use for this contract.

Status: Acknowledged by the Auditee

4. Calling function is not initiated

Line	Code
196	try r.swapExactTokensForNATIVE(swapTokens,

Description
The **swapExactTokensForNATIVE** is not initiated in the IUniswapV2Router02 interface.

The scope of the audit would treat the third-party implementation at **IUniswapV2Router02** as a black box and assume its functional correctness. However, third parties may be compromised in the real world that leads to assets lost or stolen.

Remediation
Please verify that the IUniswapV2Router02 contract you are calling is compatible with the IUniswapV2Router02 interface.

Status: Acknowledged by the Auditee
Auditee asserted that they have modified IUniswapV2Router functions such that all namings containing ETH were renamed to NATIVE.

Functional test

Function Names	Testing results
Deposit	Passed
setRouter	Passed
withdraw	Passed
releaseTokens	Passed
executeProposal	Failed
deposit	Passed
getDepositRecord	Passed

Automated Testing

Slither

```
INFO:Detectors:
ERC20GenericNativeSwap.executeProposal(bytes32,bytes) (ERC20GenericNativeSwap.sol#144-210) ignores return value by IERC20(tokenAddress).transfer(address(recipientAddress),swap
Tokens) (ERC20GenericNativeSwap.sol#194)
ERC20GenericNativeSwap.executeProposal(bytes32,bytes) (ERC20GenericNativeSwap.sol#144-210) ignores return value by IERC20(tokenAddress).transfer(address(recipientAddress),swap
Tokens) (ERC20GenericNativeSwap.sol#197)
ERC20GenericNativeSwap.executeProposal(bytes32,bytes) (ERC20GenericNativeSwap.sol#144-210) ignores return value by IERC20(tokenAddress).transfer(address(recipientAddress),swap
Tokens) (ERC20GenericNativeSwap.sol#202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
ERC20GenericNativeSwap.executeProposal(bytes32,bytes) (ERC20GenericNativeSwap.sol#144-210) performs a multiplication on the result of a division:
        -swapTokens = (amount / 10000) * swapPer (ERC20GenericNativeSwap.sol#166)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
ERC20GenericNativeSwap.executeProposal(bytes32,bytes) (ERC20GenericNativeSwap.sol#144-210) ignores return value by IERC20(tokenAddress).approve(router,swapTokens) (ERC20Generi
cNativeSwap.sol#179)
ERC20GenericNativeSwap.executeProposal(bytes32,bytes) (ERC20GenericNativeSwap.sol#144-210) ignores return value by r.swapExactTokensForNATIVE(swapTokens,0,path,address(recipie
ntAddress),now + 30 * 60) (ERC20GenericNativeSwap.sol#186-199)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
ERC20PresetMinterPauser.constructor(string,string).name (openzeppelin/contracts/presets/ERC20PresetMinterPauser.sol#35) shadows:
        - ERC20.name() (openzeppelin/contracts/token/ERC20/ERC20.sol#64-66) (function)
ERC20PresetMinterPauser.constructor(string,string).symbol (openzeppelin/contracts/presets/ERC20PresetMinterPauser.sol#35) shadows:
        - ERC20.symbol() (openzeppelin/contracts/token/ERC20/ERC20.sol#72-74) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
ERC20GenericNativeSwap.constructor(address,address,bytes32[],address[],address[]).bridgeAddress (ERC20GenericNativeSwap.sol#62) lacks a zero-check on :
                - _bridgeAddress = bridgeAddress (ERC20GenericNativeSwap.sol#71)
ERC20GenericNativeSwap.constructor(address,address,bytes32[],address[],address[]).WNATIVE (ERC20GenericNativeSwap.sol#63) lacks a zero-check on :
                - _WNATIVE = WNATIVE (ERC20GenericNativeSwap.sol#72)
ERC20GenericNativeSwap.setRouter(address)._router (ERC20GenericNativeSwap.sol#241) lacks a zero-check on :
                - router = _router (ERC20GenericNativeSwap.sol#242)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'ERC20GenericNativeSwap.executeProposal(bytes32,bytes).amounts (ERC20GenericNativeSwap.sol#191)' in ERC20GenericNativeSwap.executeProposal(bytes32,bytes) (ERC20Generi
cNativeSwap.sol#144-210) potentially used before declaration: swapSuccessful(address(recipientAddress),amounts) (ERC20GenericNativeSwap.sol#192)
Variable 'ERC20GenericNativeSwap.executeProposal(bytes32,bytes)._err (ERC20GenericNativeSwap.sol#193)' in ERC20GenericNativeSwap.executeProposal(bytes32,bytes) (ERC20GenericNa
tiveSwap.sol#144-210) potentially used before declaration: swapFailure(address(recipientAddress),_err,bytes()) (ERC20GenericNativeSwap.sol#195)
Variable 'ERC20GenericNativeSwap.executeProposal(bytes32,bytes)._err_scope_0 (ERC20GenericNativeSwap.sol#196)' in ERC20GenericNativeSwap.executeProposal(bytes32,bytes) (ERC20G
enericNativeSwap.sol#144-210) potentially used before declaration: swapFailure(address(recipientAddress),,_err_scope_0) (ERC20GenericNativeSwap.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
fundERC20(address,address,uint256) should be declared external:
        - ERC20Safe.fundERC20(address,address,uint256) (ERC20Safe.sol#22-25)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControl.getRoleMemberCount(bytes32) (openzeppelin/contracts/access/AccessControl.sol#95-97)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControl.getRoleMember(bytes32,uint256) (openzeppelin/contracts/access/AccessControl.sol#111-113)
getRoleAdmin(bytes32) should be declared external:
        - AccessControl.getRoleAdmin(bytes32) (openzeppelin/contracts/access/AccessControl.sol#121-123)
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (openzeppelin/contracts/access/AccessControl.sol#135-139)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (openzeppelin/contracts/access/AccessControl.sol#150-154)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (openzeppelin/contracts/access/AccessControl.sol#170-174)
owner() should be declared external:
        - Ownable.owner() (openzeppelin/contracts/access/Ownable.sol#35-37)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (openzeppelin/contracts/access/Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (openzeppelin/contracts/access/Ownable.sol#63-67)
mint(address,uint256) should be declared external:
        - ERC20PresetMinterPauser.mint(address,uint256) (openzeppelin/contracts/presets/ERC20PresetMinterPauser.sol#51-54)
pause() should be declared external:
        - ERC20PresetMinterPauser.pause() (openzeppelin/contracts/presets/ERC20PresetMinterPauser.sol#65-68)
unpause() should be declared external:
        - ERC20PresetMinterPauser.unpause() (openzeppelin/contracts/presets/ERC20PresetMinterPauser.sol#79-82)
name() should be declared external:
        - ERC20.name() (openzeppelin/contracts/token/ERC20/ERC20.sol#64-66)
symbol() should be declared external:
        - ERC20.symbol() (openzeppelin/contracts/token/ERC20/ERC20.sol#72-74)
decimals() should be declared external:
        - ERC20.decimals() (openzeppelin/contracts/token/ERC20/ERC20.sol#89-91)
totalSupply() should be declared external:
        - ERC20.totalSupply() (openzeppelin/contracts/token/ERC20/ERC20.sol#96-98)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (openzeppelin/contracts/token/ERC20/ERC20.sol#103-105)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#115-118)
approve(address,uint256) should be declared external:
```

```
Pragma version>=0.6.0<0.8.0 (openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version>=0.6.2<0.8.0 (openzeppelin/contracts/utils/Address.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (openzeppelin/contracts/utils/EnumerableSet.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (openzeppelin/contracts/utils/Pausable.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in ERC20Safe._safeCall(IERC20,bytes) (ERC20Safe.sol#100-108):
        - (success,returndata) = address(token).call(data) (ERC20Safe.sol#101)
Low level call in TransferHelper.safeApprove(address,address,uint256) (TransferHelper.sol#7-11):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (TransferHelper.sol#9)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (TransferHelper.sol#13-17):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (TransferHelper.sol#15)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (TransferHelper.sol#19-23):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (TransferHelper.sol#21)
Low level call in TransferHelper.safeTransferNATIVE(address,uint256) (TransferHelper.sol#25-28):
        - (success) = to.call{value: value}(new bytes(0)) (TransferHelper.sol#26)
Low level call in Address.sendValue(address,uint256) (openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (openzeppelin/contracts/utils/Address.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Event ERC20GenericNativeSwapswapFailure(address,string,bytes) (ERC20GenericNativeSwap.sol#38) is not in CapWords
Event ERC20GenericNativeSwapswapSuccessful(address,uint256[]) (ERC20GenericNativeSwap.sol#39) is not in CapWords
Parameter ERC20GenericNativeSwap.setRouter(address)._router (ERC20GenericNativeSwap.sol#241) is not in mixedCase
Variable ERC20GenericNativeSwap._WNATIVE (ERC20GenericNativeSwap.sol#36) is not in mixedCase
Variable ERC20GenericNativeSwap._depositRecords (ERC20GenericNativeSwap.sol#43) is not in mixedCase
Variable HandlerHelpers._bridgeAddress (HandlerHelpers.sol#11) is not in mixedCase
Variable HandlerHelpers._resourceIDToTokenContractAddress (HandlerHelpers.sol#14) is not in mixedCase
Variable HandlerHelpers._tokenContractAddressToResourceID (HandlerHelpers.sol#17) is not in mixedCase
Variable HandlerHelpers._contractWhitelist (HandlerHelpers.sol#20) is not in mixedCase
Variable HandlerHelpers._burnList (HandlerHelpers.sol#23) is not in mixedCase
Function IUniswapV2Router01.WNATIVE() (IUniswapV2Router01.sol#7) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (openzeppelin/contracts/GSN/Context.sol#21)" inContext (openzeppelin/contracts/GSN/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (IUniswapV2Router01.sol#12) is too similar to IUniswap
V2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (IUniswapV2Router01.sol#13)
```

```
Address.functionCallWithValue(address,bytes,uint256) (openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (openzeppelin/contracts/utils/Address.sol#114-121) is never used and should be removed
Address.functionStaticCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#139-145) is never used and should be removed
Address.isContract(address) (openzeppelin/contracts/utils/Address.sol#26-35) is never used and should be removed
Address.sendValue(address,uint256) (openzeppelin/contracts/utils/Address.sol#53-59) is never used and should be removed
Context._msgData() (openzeppelin/contracts/GSN/Context.sol#20-23) is never used and should be removed
ERC20._setupDecimals(uint8) (openzeppelin/contracts/token/ERC20/ERC20.sol#287-289) is never used and should be removed
ERC20Pausable._beforeTokenTransfer(address,address,uint256) (openzeppelin/contracts/token/ERC20/ERC20Pausable.sol#23-27) is never used and should be removed
ERC20PresetMinterPauser._beforeTokenTransfer(address,address,uint256) (openzeppelin/contracts/presets/ERC20PresetMinterPauser.sol#84-86) is never used and should be removed
EnumerableSet.add(EnumerableSet.Bytes32Set,bytes32) (openzeppelin/contracts/utils/EnumerableSet.sol#147-149) is never used and should be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256) (openzeppelin/contracts/utils/EnumerableSet.sol#256-258) is never used and should be removed
EnumerableSet.at(EnumerableSet.Bytes32Set,uint256) (openzeppelin/contracts/utils/EnumerableSet.sol#185-187) is never used and should be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256) (openzeppelin/contracts/utils/EnumerableSet.sol#294-296) is never used and should be removed
EnumerableSet.contains(EnumerableSet.Bytes32Set,bytes32) (openzeppelin/contracts/utils/EnumerableSet.sol#164-166) is never used and should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (openzeppelin/contracts/utils/EnumerableSet.sol#273-275) is never used and should be removed
EnumerableSet.length(EnumerableSet.Bytes32Set) (openzeppelin/contracts/utils/EnumerableSet.sol#171-173) is never used and should be removed
EnumerableSet.length(EnumerableSet.UintSet) (openzeppelin/contracts/utils/EnumerableSet.sol#280-282) is never used and should be removed
EnumerableSet.remove(EnumerableSet.Bytes32Set,bytes32) (openzeppelin/contracts/utils/EnumerableSet.sol#157-159) is never used and should be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256) (openzeppelin/contracts/utils/EnumerableSet.sol#266-268) is never used and should be removed
SafeMath.div(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#103-105) is never used and should be removed
SafeMath.div(uint256,uint256,string) (openzeppelin/contracts/math/SafeMath.sol#119-125) is never used and should be removed
SafeMath.mod(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#139-141) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (openzeppelin/contracts/math/SafeMath.sol#155-158) is never used and should be removed
SafeMath.mul(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#77-89) is never used and should be removed
TransferHelper.safeApprove(address,address,uint256) (TransferHelper.sol#7-11) is never used and should be removed
TransferHelper.safeTransfer(address,address,uint256) (TransferHelper.sol#13-17) is never used and should be removed
TransferHelper.safeTransferFrom(address,address,address,uint256) (TransferHelper.sol#19-23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.2 (IUniswapV2Router01.sol#3) allows old versions
Pragma version>=0.6.2 (IUniswapV2Router02.sol#3) allows old versions
Pragma version>=0.5.0 (IWNATIVE.sol#3) allows old versions
Pragma version>=0.6.0 (TransferHelper.sol#3) allows old versions
Pragma version>=0.6.0<0.8.0 (openzeppelin/contracts/GSN/Context.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (openzeppelin/contracts/access/AccessControl.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (openzeppelin/contracts/access/Ownable.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (openzeppelin/contracts/math/SafeMath.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (openzeppelin/contracts/presets/ERC20PresetMinterPauser.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (openzeppelin/contracts/token/ERC20/ERC20.sol#3) is too complex
```

```
INFO:Detectors:
ERC20GenericNativeSwap.executeProposal(bytes32,bytes) (ERC20GenericNativeSwap.sol#144-210) uses assembly
        - INLINE ASM (ERC20GenericNativeSwap.sol#161-163)
Address.isContract(address) (openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (openzeppelin/contracts/utils/Address.sol#147-164) uses assembly
        - INLINE ASM (openzeppelin/contracts/utils/Address.sol#156-159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.6.12', '>=0.5.0', '>=0.6.0', '>=0.6.0<0.8.0', '>=0.6.2', '>=0.6.2<0.8.0']
        - 0.6.12 (ERC20GenericNativeSwap.sol#1)
        - ABIEncoderV2 (ERC20GenericNativeSwap.sol#2)
        - 0.6.12 (ERC20Safe.sol#1)
        - 0.6.12 (HandlerHelpers.sol#1)
        - 0.6.12 (IDepositExecute.sol#1)
        - 0.6.12 (IERCHandler.sol#1)
        - >=0.6.2 (IUniswapV2Router01.sol#3)
        - >=0.6.2 (IUniswapV2Router02.sol#3)
        - >=0.5.0 (IWNATIVE.sol#3)
        - >=0.6.0 (TransferHelper.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/GSN/Context.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/access/AccessControl.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/access/Ownable.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/math/SafeMath.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/presets/ERC20PresetMinterPauser.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/token/ERC20/ERC20.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/token/ERC20/ERC20Burnable.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/token/ERC20/ERC20Pausable.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >=0.6.2<0.8.0 (openzeppelin/contracts/utils/Address.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/utils/EnumerableSet.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/utils/Pausable.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
AccessControl._setRoleAdmin(bytes32,bytes32) (openzeppelin/contracts/access/AccessControl.sol#201-204) is never used and should be removed
Address._verifyCallResult(bool,bytes,string) (openzeppelin/contracts/utils/Address.sol#147-164) is never used and should be removed
Address.functionCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#89-91) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (openzeppelin/contracts/utils/Address.sol#114-121) is never used and should be removed
```

```
INFO:Detectors:
Reentrancy in ERC20GenericNativeSwap.deposit(bytes32,uint8,uint64,address,bytes) (ERC20GenericNativeSwap.sol#101-133):
        External calls:
        - burnERC20(tokenAddress,depositer,amount) (ERC20GenericNativeSwap.sol#122)
                - erc20.burnFrom(owner,amount) (ERC20Safe.sol#70)
        - lockERC20(tokenAddress,depositer,address(this),amount) (ERC20GenericNativeSwap.sol#124)
                - (success,returndata) = address(token).call(data) (ERC20Safe.sol#101)
        State variables written after the call(s):
        - _depositRecords[destinationChainID][depositNonce] = DepositRecord(destinationChainID,depositer,resourceID,data) (ERC20GenericNativeSwap.sol#127-132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in ERC20GenericNativeSwap.executeProposal(bytes32,bytes) (ERC20GenericNativeSwap.sol#144-210):
        External calls:
        - releaseTokens(tokenAddress,address(this),swapTokens) (ERC20GenericNativeSwap.sol#170)
                - erc20.mint(recipient,amount) (ERC20Safe.sol#58)
                - (success,returndata) = address(token).call(data) (ERC20Safe.sol#101)
        - IERC20(tokenAddress).approve(router,swapTokens) (ERC20GenericNativeSwap.sol#179)
        - r.swapExactTokensForNATIVE(swapTokens,0,path,address(recipientAddress),now + 30 * 60) (ERC20GenericNativeSwap.sol#186-199)
        Event emitted after the call(s):
        - swapSuccessful(address(recipientAddress),amounts) (ERC20GenericNativeSwap.sol#192)
Reentrancy in ERC20GenericNativeSwap.executeProposal(bytes32,bytes) (ERC20GenericNativeSwap.sol#144-210):
        External calls:
        - releaseTokens(tokenAddress,address(this),swapTokens) (ERC20GenericNativeSwap.sol#170)
                - erc20.mint(recipient,amount) (ERC20Safe.sol#58)
                - (success,returndata) = address(token).call(data) (ERC20Safe.sol#101)
        - IERC20(tokenAddress).approve(router,swapTokens) (ERC20GenericNativeSwap.sol#179)
        - r.swapExactTokensForNATIVE(swapTokens,0,path,address(recipientAddress),now + 30 * 60) (ERC20GenericNativeSwap.sol#186-199)
        - IERC20(tokenAddress).transfer(address(recipientAddress),swapTokens) (ERC20GenericNativeSwap.sol#194)
        Event emitted after the call(s):
        - swapFailure(address(recipientAddress),_err,bytes()) (ERC20GenericNativeSwap.sol#195)
Reentrancy in ERC20GenericNativeSwap.executeProposal(bytes32,bytes) (ERC20GenericNativeSwap.sol#144-210):
        External calls:
        - releaseTokens(tokenAddress,address(this),swapTokens) (ERC20GenericNativeSwap.sol#170)
                - erc20.mint(recipient,amount) (ERC20Safe.sol#58)
                - (success,returndata) = address(token).call(data) (ERC20Safe.sol#101)
        - IERC20(tokenAddress).approve(router,swapTokens) (ERC20GenericNativeSwap.sol#179)
        - r.swapExactTokensForNATIVE(swapTokens,0,path,address(recipientAddress),now + 30 * 60) (ERC20GenericNativeSwap.sol#186-199)
        - IERC20(tokenAddress).transfer(address(recipientAddress),swapTokens) (ERC20GenericNativeSwap.sol#197)
        Event emitted after the call(s):
        - swapFailure(address(recipientAddress),,_err_scope_0) (ERC20GenericNativeSwap.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
```

```
totalSupply() should be declared external:
        - ERC20.totalSupply() (openzeppelin/contracts/token/ERC20/ERC20.sol#96-98)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (openzeppelin/contracts/token/ERC20/ERC20.sol#103-105)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#115-118)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#134-137)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#152-156)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#170-173)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#189-192)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (openzeppelin/contracts/token/ERC20/ERC20Burnable.sol#21-23)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20Burnable.sol#36-41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Mythril

```
enderphan@enderphan contracts % myth a ERC20GenericNativeSwap.sol
The analysis was completed successfully. No issues were detected.
```

Theo

```
enderphan@enderphan ~ % theo --rpc-http HTTP://127.0.0.1:7545

The account's private key (input hidden)
>
Contract to interact with
> 0x4F882c9Daf1868e44af541f6a561F65085A71fBE
Scanning for exploits in contract: 0x4F882c9Daf1868e44af541f6a561F65085A71fBE
Connecting to HTTP: HTTP://127.0.0.1:7545.
No exploits found. You're going to need to load some exploits.

Tools available in the console:
- `exploits` is an array of loaded exploits found by Mythril or read from a file
- `w3` an initialized instance of web3py for the provided HTTP RPC endpoint
- `dump()` writing a json representation of an object to a local file

Check the readme for more info:
https://github.com/cleanunicorn/theo

Theo version v0.8.2.
```

Solhint Linter

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read property 'name' of undefined
Pos: not available

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 33:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 180:16:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 167:8:

Block timestamp:

Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp".
"block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 200:21:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 57:27:

Solidity Static Analysis

contracts/handlers/ERC20GenericNativeSwap.sol:1:1: Error: Compiler version 0.6.12 does not satisfy the r semver requirement

contracts/handlers/ERC20GenericNativeSwap.sol:36:20: Error: Variable name must be in mixedCase

contracts/handlers/ERC20GenericNativeSwap.sol:38:5: Error: Event name must be in CamelCase

contracts/handlers/ERC20GenericNativeSwap.sol:39:5: Error: Event name must be in CamelCase

contracts/handlers/ERC20GenericNativeSwap.sol:63:6: Error: Variable name must be in mixedCase

contracts/handlers/ERC20GenericNativeSwap.sol:108:2: Error: Variable "destinationRecipientAddress" is unused

contracts/handlers/ERC20GenericNativeSwap.sol:167:9: Error: Avoid using inline assembly. It is acceptable only in rare cases.

contracts/handlers/ERC20GenericNativeSwap.sol:200:22: Error: Avoid to make time-based decisions in your business logic

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the DCTDAO platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the DCTDAO Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Re-entrancy or Back-Door Entry were found in the contract but Integer Overflow and Underflow vulnerabilities.

The audit showed several high and informational severity issues. In the end, one high severity issue was fixed by the Auditee. Other informational issues remained unfixed as the internal team needs further discussion.

DCTDAO

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com