Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

2 Audit Methodology

3 Project Overview

3.1 Project Introduction

3.2 Vulnerability Information

4 Code Overview

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

5 Audit Result

6 Statement

1 Executive Summary

On 2022.04.04, the SlowMist security team received the Changer team's security audit application for Changer protocol, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

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

The vulnerability severity level information:

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

Level	Description
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit
		Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

Audit Version

```
https://github.com/changerio/aggregator-contracts
commit: 0a783d927fe9f5d4ae3fa6ea95472e1d82abe0be

Scope of the audit in the Aggregator Contracts section:
  actions/*.sol
  compensation/*.sol
  lib/*.sol
  IActionHandler.sol
  ActionHandler.sol
  ActionHandlerStorage.sol
  ActionTypes.sol
  ChangerSwapRouterV3.sol
```

```
https://github.com/changerio/cng-contracts
commit: 2daac700f717bda357d2c9540ebdfc89c704b557

Scope of the audit in the CNG Contracts section:
  all files in the contracts directory except the following four files:
  /swapper/NonLinearTimeLockSwapper.sol
  /swapper/NonLinearTimeLockSwapperV2.sol
  /swapper/NonLinearTimeLockSwapperV2Storage.sol
  /swapper/NonLinearTimeLockSwapperV2_0_0.sol
```

Fixed Version

https://github.com/changerio/aggregator-contracts/tree/3dee25726fb6704e3a791fa53e2b45ca3d1b9be4

https://github.com/changerio/cng-contracts/tree/19ba75d7bf5a3a25b3ecc93612885b340c55d1b9

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Coding specification optimization	Others	Suggestion	Fixed
N2	Business logic is not clear	Others	Low	Fixed
N3	from address missing permission check	Authority Control Vulnerability	High	Fixed
N4	Assets retained in the contract can be transferred	Authority Control Vulnerability	Low	Confirmed
N5	Slippage check is flawed	Design Logic Audit	Low	Confirmed
N6	Slippage check is missing	Design Logic Audit	Low	Confirmed
N7	Over-Approve issue	Design Logic Audit	Low	Confirmed
N8	Preemptive initialization	Race Conditions Vulnerability	Low	Confirmed
N9	Excessive authority issue	Authority Control Vulnerability	Medium	Fixed
N10	Arithmetic precision issue	Arithmetic Accuracy Deviation Vulnerability	Low	Fixed
N11	Token Compatibility Issues	Others	Suggestion	Confirmed

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

ActionHandler			
Function Name	Visibility	Mutability	Modifiers
run	External	Payable	-
_run	Public	Payable	-

ChangerSwapRouterV3			
Function Name	Visibility	Mutability	Modifiers
<Receive Ether>	External	Payable	-
swapWithCompensation	External	Payable	-
swap	External	Payable	-
_swap	Internal	Can Modify State	whenNotPaused
_getBalance	Internal	-	-
_transfer	Internal	Can Modify State	-
_initialize	External	Can Modify State	-
pause	External	Can Modify State	onlyOwner
unpause	External	Can Modify State	onlyOwner

BalancerV1Actions			
Function Name	Visibility	Mutability	Modifiers

BalancerV1Actions			
	Private Source Code		

BalancerV2Actions			
	Private Source Code		

BaseActions			
	Private Source Code		

CacheActions			
Function Name	Visibility	Mutability	Modifiers
	Private Source Code		

CurveV1Actions

CurveV1Actions			
Function Name	Visibility	Mutability	Modifiers
	Private Source Code		

CurveV2Actions			
Function Name	Visibility	Mutability	Modifiers
	Private Source Code		

ERC20Actions			
Function Name	Visibility	Mutability	Modifiers
	Private Source Code		

ERC20Actions			
	Private Source Code		

SushiswapActions			
Function Name	Visibility	Mutability	Modifiers
	Private Source Code		

UniswapV2Actions			
Function Name	Visibility	Mutability	Modifiers
	Private Source Code		

UniswapV3Actions			
Function Name	Visibility	Mutability	Modifiers
	Private Source Code		

WETHActions			
Function Name	Visibility	Mutability	Modifiers

WETHActions			
	Private Source Code		

CompensationVault			
Function Name	Visibility	Mutability	Modifiers
_initialize	Public	Can Modify State	-
updateLimits	External	Can Modify State	onlyOwner
reset	Public	Can Modify State	onlyOwner whenPaused
remainingCompensation	Public	-	-
getDailyLimit	Public	-	-
claimCompensation	External	Can Modify State	-
_claimCompensation	Internal	Can Modify State	-
hashParams	Public	-	-
addCompensation	External	Can Modify State	onlyRouter onlySigner
calcCompensationAmount	Public	-	-
_min	Private	-	-
setSigner	External	Can Modify State	onlyOwner

CompensationVault			
setRouter	External	Can Modify State	onlyOwner
chainID	Public	-	-
pause	External	Can Modify State	onlyOwner whenNotPaused
unpause	Public	Can Modify State	onlyOwner whenPaused
rescueTokens	External	Can Modify State	onlyOwner whenPaused
getCompensation	External	-	-

StorageSlotOwnable			
Function Name	Visibility	Mutability	Modifiers
_getOwner	Internal	-	-
_setOwner	Internal	Can Modify State	-
owner	Public	-	-
renounceOwnership	Public	Can Modify State	onlyOwner
transferOwnership	Public	Can Modify State	onlyOwner

StorageSlotPausable			
Function Name	Visibility	Mutability	Modifiers
_getPaused	Private	-	-
_setPaused	Private	Can Modify State	-
paused	Public	-	-
_pause	Internal	Can Modify State	whenNotPaused

StorageSlotPausable			
_unpause	Internal	Can Modify State	whenPaused

AirdropRegistry			
Function Name	Visibility	Mutability	Modifiers
chainid	External	-	-
implementationVersion	Public	-	-
_initializeKernel	Internal	Can Modify State	-
onApprove	External	Can Modify State	-
claim	Public	Can Modify State	-
claims	External	Can Modify State	-

AirdropRegistryVault			
Function Name	Visibility	Mutability	Modifiers
setAirdropRegistry	External	Can Modify State	onlyOwner
onApprove	External	Can Modify State	-
pause	External	Can Modify State	onlyOwner
unpause	External	Can Modify State	onlyOwner
claimToken	Public	Can Modify State	onlyOwner whenPaused
claimTokens	External	Can Modify State	onlyOwner whenPaused

AirdropRegistryVerify

AirdropRegistryVerify			
Function Name	Visibility	Mutability	Modifiers
hashAirdropInfo	Private	-	-
verifyAirdropInfo	Public	-	-

Kernel			
Function Name	Visibility	Mutability	Modifiers
implementationVersion	Public	-	-
initializeKernel	External	Can Modify State	-
_initializeKernel	Internal	Can Modify State	-
kernelInitialized	Public	-	-
kernelInitialized	Public	-	-
getVersionHash	Public	-	-
getVersionHash	Public	-	-

NonLinearTimeLockSwapperV2_0_2			
Function Name	Visibility	Mutability	Modifiers
implementationVersion	Public	-	-
_initializeKernel	Internal	Can Modify State	-
_initializeV2	Private	Can Modify State	onlyValidAddress onlyValidAddress onlyValidAddress
setTokenWallet	External	Can Modify State	onlyOwner onlyValidAddress

NonLinearTimeLockSwapperV2_0_2			
setInitialBalances	External	Can Modify State	onlyOwner onlyMigrationNotStopped
setClaimedAmounts	External	Can Modify State	onlyOwner onlyMigrationNotStopped
undeposit	Public	Can Modify State	onlyMigrationNotStopped
undeposits	External	Can Modify State	-
stopMigration	External	Can Modify State	onlyOwner
register	External	Can Modify State	onlyOwner
isRegistered	Public	-	-
getStepEndTimes	External	-	-
getAccStepRatio	External	-	-
onApprove	External	Can Modify State	-
deposit	Public	Can Modify State	onlyValidAddress
claim	Public	Can Modify State	onlyDeposit
claimTokens	External	Can Modify State	-
claimable	Public	-	-
claimableAt	Public	-	-
initialBalance	External	-	-
getStep	Public	-	-

NonLinearTimeLockSwapperV2_0_2			
getStepAt	Public	-	-

Swapper			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
register	External	Can Modify State	onlyOwner
isRegistered	Public	-	-
onApprove	External	Can Modify State	-
deposit	Public	Can Modify State	-
claim	External	Can Modify State	-
initialBalance	External	-	-
claimable	External	-	-
claimableAt	External	-	-
claimed	External	-	-

NonLinearTimeLock			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
init	External	Can Modify State	onlyOwner
onApprove	External	Can Modify State	-
addDeposit	External	Can Modify State	-

NonLinearTimeLock			
_addDeposit	Internal	Can Modify State	-
claim	External	Can Modify State	-
claimable	Public	-	-
claimableAt	Public	-	-
getStep	Public	-	-
getStepAt	Public	-	-

TimeLock			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
init	External	Can Modify State	onlyOwner
claim	External	Can Modify State	-
claimable	Public	-	-
claimableAt	Public	-	-

TimeLockFactory			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
onApprove	External	Can Modify State	-
deposit	Public	Can Modify State	-
claim	External	Can Modify State	-

TimeLockFactory			
initialBalance	External	-	-
claimable	External	-	-
claimableAt	External	-	-
claimed	External	-	-

ERC20MultiTransfer			
Function Name	Visibility	Mutability	Modifiers
transferToken	Public	Can Modify State	-
transferTokens	External	Can Modify State	-

OnApprove			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
onApprove	External	Can Modify State	-

Token			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20PresetMinterPauser ERC20Permit
changeAdminRole	External	Can Modify State	-
changeRole	Public	Can Modify State	-
_beforeTokenTransfer	Internal	Can Modify State	-

Token			
getChainId	External	-	-

SwapperVault			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	onlyValidAddress
setSwapper	External	Can Modify State	whenNotPaused onlyOwner onlyValidAddress
_approveToSwapper	Internal	Can Modify State	-
pause	External	Can Modify State	onlyOwner
unpause	External	Can Modify State	onlyOwner
claimTokens	External	Can Modify State	onlyOwner whenPaused

4.3 Vulnerability Summary

[N1] [Suggestion] Coding specification optimization

Category: Others

Content

The visibility of functions named starting with _ in contract code is internal , but _initialize is public.

- aggregator-contracts/contracts/ChangerSwapRouterV3.sol#L120-L122

```
function _initialize(address _owner) external {
    if (owner() == address(0)) _setOwner(_owner);
}
```

- aggregator-contracts/contracts/compensation/CompensationVault.sol#L56-L77

```
function _initialize(
        address _owner,
        address _token,
        uint256 _START_TIME,
        uint256 _MAX_RUNNING_DAYS,
        uint256 _TOTAL_ALLOCATED,
        uint256 _LIMIT_PER_TX,
        uint256 _LIMIT_PER_DAY
    ) public {
        require(!_initialized0, "IE"); // initialize error
        token = _token;

        _setOwner(_owner);

        START_TIME = uint64(_START_TIME);
        MAX_RUNNING_DAYS = uint64(_MAX_RUNNING_DAYS);
        TOTAL_ALLOCATED = uint128(_TOTAL_ALLOCATED);
        LIMIT_PER_TX = uint128(_LIMIT_PER_TX);
        LIMIT_PER_DAY = uint128(_LIMIT_PER_DAY);

        _initialized0 = true;
    }
```

Solution

It is recommended that public functions do not need to start with "_".

Status

Fixed; The issue has been fixed in this commit:

https://github.com/changerio/aggregator- contracts/commit/4ca609888eb08a0b39057dec63d3cd67117f0ad2.

[N2] [Low] Business logic is not clear

Category: Others

Content

There are some annotations in the balancerV2Swap function, which seems to be necessary codes, and the business logic here needs to be communicated with developers.

- aggregator-contracts/contracts/actions/BalancerV2Actions.sol#L23-L52

Private Source Code

The code of AirdropRegistryMerkleProof is commented out,

- cng-contracts/contracts/airdrop/AirdropRegistry.sol#L14

```
import { SafeERC20 } from "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";

import { StorageSlotOwnable } from "../lib/StorageSlotOwnable.sol";
import { OnApprove } from "../token/ERC20OnApprove.sol";

import { AirdropRegistryStorage } from "./AirdropRegistryStorage.sol";
import { AirdropRegistryVerify } from "./AirdropRegistryVerify.sol";

// import { AirdropRegistryMerkleProof } from "./AirdropRegistryMerkleProof.sol";
```

Solution

It is recommended to clarify whether the code is required. If they are redundant codes, it is recommended to delete redundant codes.

Status

Fixed; The issue has been fixed in commits:

https://github.com/changerio/aggregator-contracts/commit/1db39d2ef20e79b3b3c89119a3c9c37e531732c7.

https://github.com/changerio/cng-contracts/commit/1d57d4fe0e2cb4bfb886b2dd1c18ceca8a3fc614.

[N3] [High] from address missing permission check

Category: Authority Control Vulnerability

Content

If the user approves the contract for transfer, if the allowence is not used up, there will be an issue of arbitrarily transferring user funds.

- aggregator-contracts/contracts/actions/ERC20Actions.sol#L60-L76

Private Source Code

- aggregator-contracts/contracts/ActionHandler.sol#L44-L95

Private Source Code

Private Source Code

```
    }
```

Solution

It is recommended to add a permission check to the from address to ensure that only addresses authorized by the user can transfer funds

Status

Fixed; The issue has been fixed in this commit: https://github.com/changerio/aggregator-contracts/commit/c7ffaac9de68cc37dea5fc5574dcd0589c395b5e.

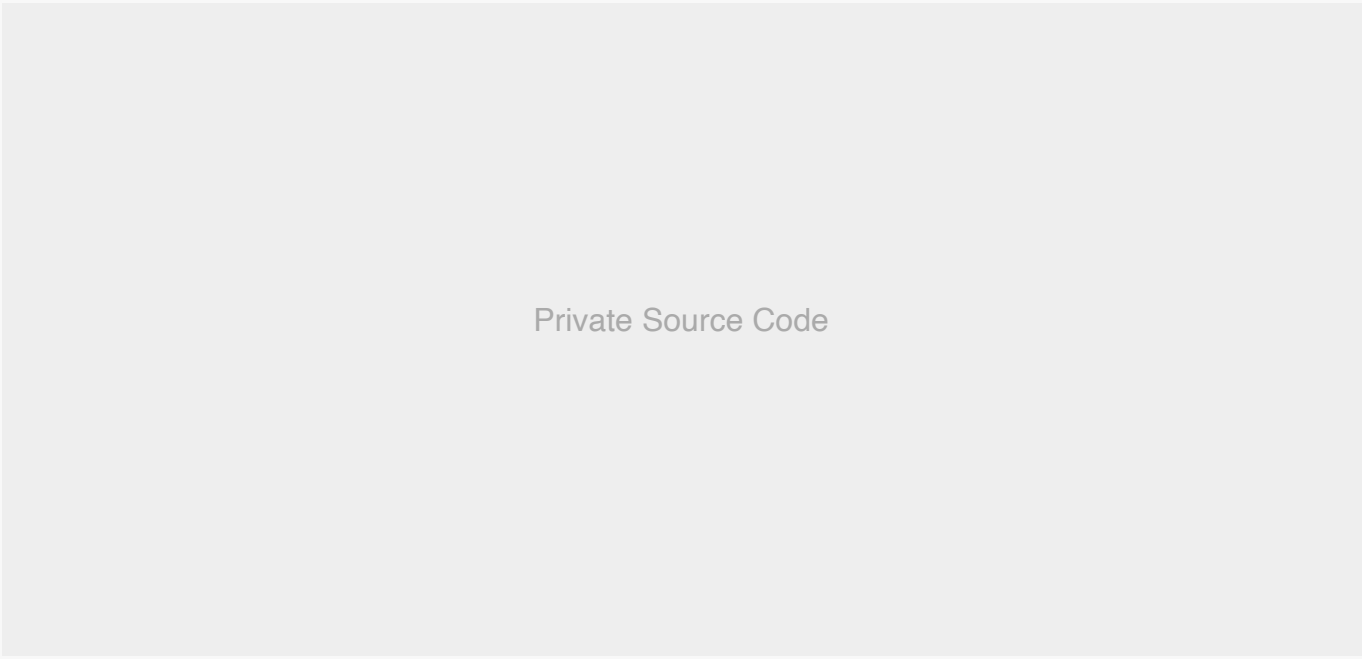
[N4] [Low] Assets retained in the contract can be transferred

Category: Authority Control Vulnerability

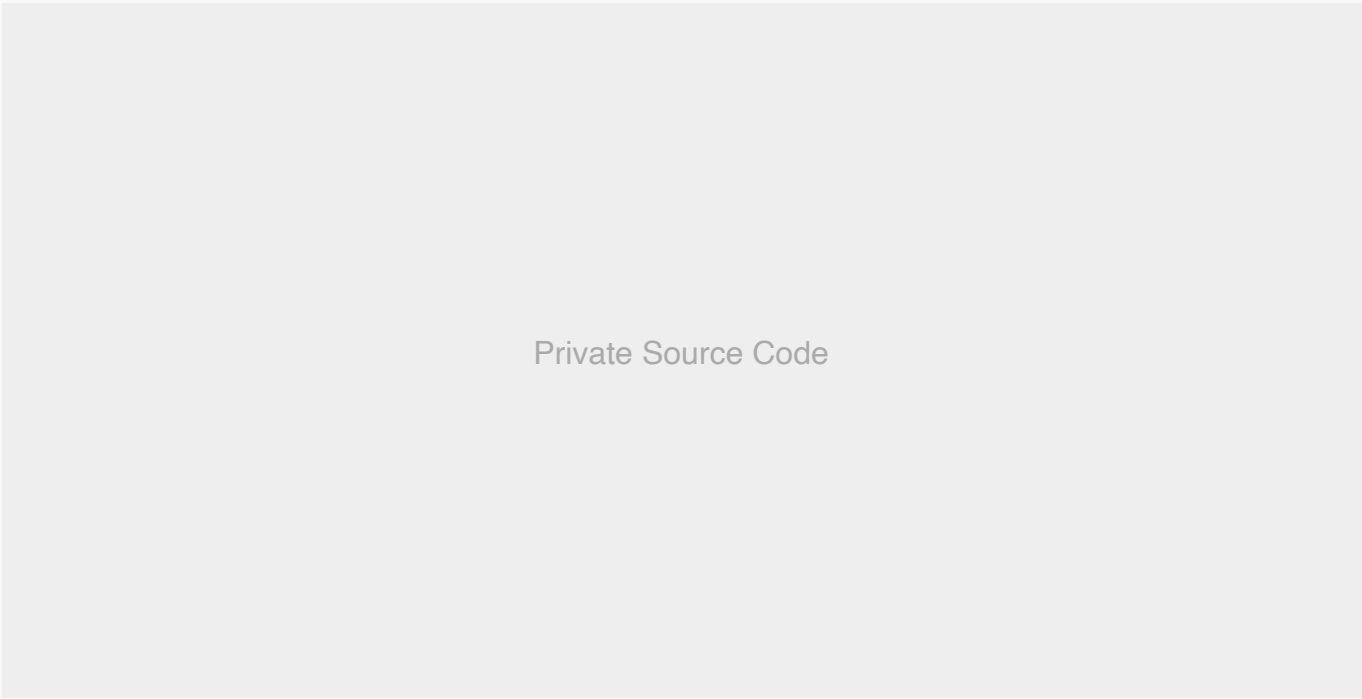
Content

The funds in the contract can be approved to any address. After the user runs the action, there are reserved funds in the contract. The attacker can use this issue to transfer the funds in the contract.

- aggregator-contracts/contracts/actions/ERC20Actions.sol#L19-L31



- aggregator-contracts/contracts/actions/ERC20Actions.sol#L39-L52



- aggregator-contracts/contracts/ActionHandler.sol#L67-L69

Private Source Code

Solution

It is recommended to remind users that they cannot keep funds in the contract.

Status

Confirmed; The project team response: As explained in Further explanation, Our service executes a routing plan that combines several actions. Therefore, there is no case where tokens remain in ActionHandler. If the user accidentally sends the token to the contract, there is a risk of losing the token due to the above cases, so we will warn users not to use the contract in the wrong way.

[N5] [Low] Slippage check is flawed

Category: Design Logic Audit

Content

The slippage is calculated by the UniswapV2_getMaxAmountOut function, but this function is calculated by reserve, which has the issue of sandwich attack.

- aggregator-contracts/contracts/actions/UniswapV2Actions.sol#L27-L70

Private Source Code

Private Source Code

- aggregator-contracts/contracts/actions/SushiswapActions.sol#L24-L66

Private Source Code

Private Source Code

Solution

It is recommended to use price oracles or let users input slippage parameters to protect against sandwich attacks.

Status

Confirmed; The project team response: As explained in Further explanation, the slippage is finally checked on the ChangerSwapRouterV3 side. Therefore, the slippage check is omitted for the intermediate step uniswap v2 single dex type.

[N6] [Low] Slippage check is missing

Category: Design Logic Audit

Content

There is no limit on slippage when using uniswapV3 for swap.

- aggregator-contracts/contracts/actions/UniswapV3Actions.sol#L28-L50

Private Source Code

- aggregator-contracts/contracts/actions/BalancerV2Actions.sol#L23-L46

Private Source Code

Private Source Code

Solution

It is recommended to add a check for amountOutMinimum/amountInMaximum Or use uniswapV3's router contract.

add a check for limit in _vault.swap.

reference:

https://docs.uniswap.org/protocol/guides/swaps/single-swaps#swap-input-parameters

https://etherscan.io/address/0xBA12222222228d8Ba445958a75a0704d566BF2C8#code#F6#L59

Status

Confirmed; The project team response: For the same reason as N5, slippage check is omitted in the intermediate stage Uniswap V3 router.

The SlowMist team response: Since the run function in the aggregator-contracts/contracts/ActionHandler.sol contract does not use the onlyRouter decorator, there is no slippage protection when the user uses the run function directly. It is recommended to add the onlyRouter decorator or remind the user to implement the router contract to ensure that the slippage is controllable.

[N7] [Low] Over-Approve issue

Category: Design Logic Audit

Content

safeApprove is not allocated as needed, but is set by the owner, and the set allowance is the max value. and the owner can arbitrarily set the spender address, a malicious spender can steal the assets in the contract.

- cng-contracts/contracts/airdrop/AirdropRegistryVault.sol#L32

```
    function setAirdropRegistry(address airdropRegistry, address[] calldata tokens)
external onlyOwner {
        for (uint256 i = 0; i < tokens.length; i++) {
            IERC20(tokens[i]).safeApprove(airdropRegistry, type(uint256).max);
        }
    }
```

_approveToSwapper is not allocated as needed, and the set allowance is the max value.

- cng-contracts/contracts/vault/SwapperVault.sol#L39-L44

```
    function setSwapper(address swapper_) external whenNotPaused onlyOwner
onlyValidAddress(swapper_) {
        address previousSwapper = swapper;
        swapper = swapper_;
        _approveToSwapper(previousSwapper, swapper_);
        emit SwapperChanged(swapper_);
    }
```

- cng-contracts/contracts/vault/SwapperVault.sol#L50

```
    function _approveToSwapper(address previousSwapper, address newSwapper) internal
{
        if (previousSwapper != address(0)) {
            token.approve(previousSwapper, 0);
        }
        token.approve(newSwapper, type(uint256).max);
    }
```

Solution

It is recommended to allowance to use on-demand approval, And the Owner needs to use multi-sign contract for

management.

Status

Confirmed; The project team response: Regarding the Airdrop contract, only the correct amount of tokens to be used at the time of the airdrop is put into the Vault and used. As suggested, the owner of AirdropRegistryVault will use a multisig contract. Regarding the swapper, only the owner of swapperVault can set up the swapper. (This owner will use a multisig contract, also) Currently, only one timelock swapper is being used. There is no limit on the quantity of this swapper because tokens are continuously unlocked in proportion to time.

[N8] [Low] Preemptive initialization

Category: Race Conditions Vulnerability

Content

The initial preemption issue will affect the initialization parameters of the contract.

- aggregator-contracts/contracts/ChangerSwapRouterV3.sol#L121

```
    function _initialize(address _owner) external {
        if (owner() == address(0)) _setOwner(_owner);
    }
```

- cng-contracts/contracts/proxy/Kernel.sol#L28-L37

```
    function initializeKernel(bytes calldata data) external returns (bool) {
        require(!kernelInitialized(), "Kernel: already-init");

        bytes32 h = getVersionHash();
        _initialized[h] = true;

        _initializeKernel(data);

        return true;
    }
```

Solution

It is recommended to use contract to deploy contract, and use the contract to call the initialization function after deployment.

Status

Confirmed; The project team response: Contracts with this issue will be distributed as proxy contracts(TransparentUpgradeableProxy, ERC1967Proxy), and scripts will be used.

[N9] [Medium] Excessive authority issue

Category: Authority Control Vulnerability

Content

The owner can change the address of the tokenWallet, if the new tokenWallet is approved by the user to the contract, there may be have risks. and setTokenWallet function is missing event logging.

- cng-contracts/contracts/swapper/NonLinearTimeLockSwapperV2_0_2.sol#L85

```
    function setTokenWallet(address tokenWallet_) external onlyOwner
onlyValidAddress(tokenWallet_) {
        tokenWallet = tokenWallet_;
    }
```

- cng-contracts/contracts/swapper/NonLinearTimeLockSwapperV2_0_2.sol#L284

```
    function claim(address sourceToken) public onlyDeposit(sourceToken, msg.sender) {
        uint256 amount = claimable(sourceToken, msg.sender);
        require(amount > 0, "invalid-amount");

        claimedAmounts[sourceToken][msg.sender] = claimedAmounts[sourceToken]
[msg.sender].add(amount);
        token.safeTransferFrom(tokenWallet, msg.sender, amount);

        emit Claimed(sourceToken, msg.sender, amount);
    }
```

Owner can arbitrarily set depositAmounts, There is an issue with excessive authority.

- cng-contracts/contracts/swapper/NonLinearTimeLockSwapperV2_0_2.sol#L104

```
    function setInitialBalances(
        address sourceToken,
        address[] calldata beneficiaries,
        uint256[] calldata amounts
    ) external onlyOwner onlyMigrationNotStopped {
        require(beneficiaries.length == amounts.length, "invalid-length");
        for (uint256 i = 0; i < amounts.length; i++) {
            depositAmounts[sourceToken][beneficiaries[i]] = amounts[i];
        }
    }
```

Owner can arbitrarily set claimedAmounts, There is an issue with excessive authority.

- cng-contracts/contracts/swapper/NonLinearTimeLockSwapperV2_0_2.sol#L118

```
    function setClaimedAmounts(
        address sourceToken,
        address[] calldata beneficiaries,
        uint256[] calldata amounts
    ) external onlyOwner onlyMigrationNotStopped {
        require(beneficiaries.length == amounts.length, "invalid-length");
        for (uint256 i = 0; i < amounts.length; i++) {
            claimedAmounts[sourceToken][beneficiaries[i]] = amounts[i];
        }
    }
```

Owner can withdraw the funds in the contract, and receiver can be set arbitrarily.

- cng-contracts/contracts/swapper/NonLinearTimeLockSwapperV2_0_2.sol#L125-148

```
    function undeposit(
        address sourceToken,
        address beneficiary,
        uint256 amount,
        address receiver
```

```
) public onlyMigrationNotStopped {
    require(msg.sender == beneficiary || msg.sender == owner(), "no-auth");

    uint256 depositAmount = depositAmounts[sourceToken][beneficiary];
    require(depositAmount > 0, "no-deposit");
    require(claimedAmounts[sourceToken][beneficiary] == 0, "already-claimed");

    if (amount == 0) {
        amount = depositAmount;
    }

    require(depositAmount >= amount, "insufficient-deposits");

    depositAmounts[sourceToken][beneficiary] = depositAmount - amount;

    IERC20(sourceToken).safeTransfer(receiver, amount);

    emit Undeposited(sourceToken, beneficiary, amount, receiver);
}
```

- cng-contracts/contracts/swapper/NonLinearTimeLockSwapperV2_0_2.sol#L150-L164

```
function undeposits(
    address[] calldata sourceToken,
    address[] calldata beneficiary,
    uint256[] calldata amount,
    address[] calldata receiver
) external {
    uint256 n = sourceToken.length;
    require(beneficiary.length == n, "invalid-length");
    require(amount.length == n, "invalid-length");
    require(receiver.length == n, "invalid-length");

    for (uint256 i = 0; i < n; i++) {
        undeposit(sourceToken[i], beneficiary[i], amount[i], receiver[i]);
    }
}
```

Solution

It is recommended to set the owner address as a timelock contract or governance contract for management or multi-

sign contract, and add an event for the setTokenWallet function to log

Status

Fixed; The project team response: As the owner, we plan to use a multisig contract.

[N10] [Low] Arithmetic precision issue

Category: Arithmetic Accuracy Deviation Vulnerability

Content

The code here will cause a loss of precision because of the div.

`timestamp.sub(startTime).div(duration.div(nSteps))`

Why not use mul instead?

`timestamp.sub(startTime).mul(nSteps)div(duration)`

- cng-contracts/contracts/timelock/TimeLock.sol#L87

```
    function claimableAt(uint256 timestamp) public view returns (uint256) {
        require(block.timestamp <= timestamp, "invalid-timestamp");

        if (timestamp < startTime) return 0;
        if (timestamp >= endTime) return initialBalance.sub(claimed);

        uint256 duration = endTime.sub(startTime);
        uint256 step = timestamp.sub(startTime).div(duration.div(nSteps));

        if (step == 0) return 0;
        return initialBalance.mul(step).div(nSteps).sub(claimed);
    }
```

Solution

It is recommended to multiply and then divide to reduce the loss of precision.

Status

Fixed; The project team response: The file was deleted because it is not currently in use in the following commit.

https://github.com/changerio/cng -contracts/commit/7228541bb57419dd2df9292e78e1972483a360cf.

[N11] [Suggestion] Token Compatibility Issues

Category: Others

Content

The contract bookkeeping is recorded by the amount transferred, which is not compatible with deflationary or inflationary tokens

Solution

It is recommended to avoid accessing deflationary or inflationary tokens when docking tokens.

Status

Confirmed; The project team response: In cng-contracts, we are managing the token list, and only tokens that do not have this issue are registered and used in the swapper, and since we are aware of this issue, we expect there will be no problem.

In aggregator-contracts, we will guide users to be fully aware of these issues when swapping deflation or inflation tokens.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002204150002	SlowMist Security Team	2022.04.04 - 2022.04.15	Low Risk

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 high risk, 1 medium risk, 7 low risk vulnerabilities and 2 suggestions. The code was not deployed to the mainnet.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist