HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Router Protocol
Date: May 25th, 2022

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Router Protocol.
Approved by	Andrew Matiukhin | CTO Hacken OU Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type	Cross-chain exchange
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/router-protocol/router-bridge-contracts-v2/tree/audit/freeze https://github.com/router-protocol/path-finder-api/tree/develop https://github.com/router-protocol/router-aggregator/tree/development
Commit	45eca63ffb3bca752dfe3e98b1032ce9a7b18ca8 cd4e615727c90444955f7551e123bc87188c13e6 6843e264a01ca5083aa66b69e5f286840ecb3834
Technical Documentation	YES
JS tests	YES
Website	https://www.routerprotocol.com/
Timeline	24 JANUARY 2022 - 25 MAY 2022
Changelog	10 MARCH 2022 - Initial Audit 25 MAY 2022 - Second Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Router Protocol (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project is smart contracts in the repository:

Repository:

https://github.com/router-protocol/router-bridge-contracts-v2/tree/audit/freeze
https://github.com/router-protocol/path-finder-api/tree/develop
https://github.com/router-protocol/router-aggregator/tree/development

Commit:

45eca63ffb3bca752dfe3e98b1032ce9a7b18ca8
cd4e615727c90444955f7551e123bc87188c13e6
6843e264a01ca5083aa66b69e5f286840ecb3834

Technical Documentation: Yes, Router-Litepaper.pdf

JS tests: Yes, in the repository

Contracts:

bridge-contracts/contracts/RouterERC20Upgradable.sol
bridge-contracts/contracts/CentrifugeAssetUpgradeable.sol
bridge-contracts/contracts/FeeManagerUpgradeable.sol
bridge-contracts/contracts/ERC20SafeUpgradeable.sol
bridge-contracts/contracts/ERC721SafeUpgradeable.sol
bridge-contracts/contracts/RouterERC721Upgradable.sol
bridge-contracts/contracts/utils/AccessControlUpgradeable.sol
bridge-contracts/contracts/utils/PausableUpgradeable.sol
bridge-contracts/contracts/utils/SafeMathUpgradeable.sol
bridge-contracts/contracts/utils/SafeCastUpgradeable.sol
bridge-contracts/contracts/utils/AddressUpgradeable.sol
bridge-contracts/contracts/BridgeUpgradeable.sol
bridge-contracts/contracts/VoterUpgradeable.sol
bridge-contracts/contracts/handlers/ERC20HandlerUpgradeable.sol
bridge-contracts/contracts/handlers/HandlerHelpersUpgradeable.sol
bridge-contracts/contracts/handlers/HandlerReserveUpgradeable.sol
bridge-contracts/contracts/interfaces/IHandlerReserve.sol
bridge-contracts/contracts/interfaces/IDepositExecute.sol
bridge-contracts/contracts/interfaces/IBridge.sol
bridge-contracts/contracts/interfaces/IWETH.sol
bridge-contracts/contracts/interfaces/IFeeManager.sol
bridge-contracts/contracts/interfaces/ILiquidityPool.sol
bridge-contracts/contracts/interfaces/IERCHandler.sol
bridge-contracts/contracts/interfaces/IOneSplitWrap.sol
bridge-contracts/contracts/interfaces/IGenericHandler.sol
bridge-contracts/contracts/interfaces/iRouterCrossTalk.sol
bridge-contracts/contracts/interfaces/iGBridge.sol
bridge-contracts/contracts/interfaces/IFeeManagerGeneric.sol
bridge-contracts/contracts/handlers/GenericHandlerUpgradeable.sol
bridge-contracts/contracts/FeeManagerGenericUpgradeable.sol
path-finder-api/contracts/IUniswapV2Factory.sol
path-finder-api/contracts/IUniswapV2Exchange.sol
path-finder-api/contracts/FetchLiquidity.sol
path-finder-api/contracts/UniversalERC20.sol
path-finder-api/forks/@dfyn/v2-periphery/contracts/UniswapV2Migrator.sol
path-finder-api/forks/@dfyn/v2-periphery/contracts/libraries/UniswapV2OracleLibrary.sol
path-finder-api/forks/@dfyn/v2-periphery/contracts/libraries/UniswapV2Library.sol
path-finder-api/forks/@dfyn/v2-periphery/contracts/libraries/SafeMath.sol
path-finder-api/forks/@dfyn/v2-periphery/contracts/UniswapV2Router01.sol
path-finder-api/forks/@dfyn/v2-periphery/contracts/examples/ExampleFlashSwap.sol
path-finder-api/forks/@dfyn/v2-periphery/contracts/examples/ExampleOracleSimple.sol
path-finder-api/forks/@dfyn/v2-periphery/contracts/examples/ExampleSwapToPrice.sol

```
        path-finder-api/forks/@dfyn/v2-periphery/contracts/examples/ExampleSlidingWindowOracle.sol
        path-finder-api/forks/@dfyn/v2-periphery/contracts/UniswapV2Router02.sol
        path-finder-api/forks/@dfyn/v2-periphery/contracts/interfaces/V1/IUniswapV1Factory.sol
        path-finder-api/forks/@dfyn/v2-periphery/contracts/interfaces/V1/IUniswapV1Exchange.sol
        path-finder-api/forks/@dfyn/v2-periphery/contracts/interfaces/IERC20.sol
        path-finder-api/forks/@dfyn/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol
        path-finder-api/forks/@dfyn/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol
        path-finder-api/forks/@dfyn/v2-periphery/contracts/interfaces/IWETH.sol
        path-finder-api/forks/@dfyn/v2-periphery/contracts/interfaces/IUniswapV2Migrator.sol
        path-finder-api/forks/@dfyn/v2-core/contracts/UniswapV2Factory.sol
        path-finder-api/forks/@dfyn/v2-core/contracts/UniswapV2ERC20.sol
        path-finder-api/forks/@dfyn/v2-core/contracts/libraries/SafeMath.sol
        path-finder-api/forks/@dfyn/v2-core/contracts/libraries/UQ112x112.sol
        path-finder-api/forks/@dfyn/v2-core/contracts/libraries/Math.sol
        path-finder-api/forks/@dfyn/v2-core/contracts/UniswapV2Pair.sol
        path-finder-api/forks/@dfyn/v2-core/contracts/interfaces/IERC20.sol
        path-finder-api/forks/@dfyn/v2-core/contracts/interfaces/IUniswapV2ERC20.sol
        path-finder-api/forks/@dfyn/v2-core/contracts/interfaces/IUniswapV2Factory.sol
        path-finder-api/forks/@dfyn/v2-core/contracts/interfaces/IUniswapV2Pair.sol
        path-finder-api/forks/@dfyn/v2-core/contracts/interfaces/IUniswapV2Callee.sol
        path-finder-api/forks/@dfyn/lib/contracts/libraries/SafeERC20Namer.sol
        path-finder-api/forks/@dfyn/lib/contracts/libraries/AddressStringUtil.sol
        path-finder-api/forks/@dfyn/lib/contracts/libraries/FixedPoint.sol
        path-finder-api/forks/@dfyn/lib/contracts/libraries/PairNamer.sol
        path-finder-api/forks/@dfyn/lib/contracts/libraries/Babylonian.sol
        path-finder-api/forks/@dfyn/lib/contracts/libraries/TransferHelper.sol
        router-aggregator/contracts/interface/IUniswapExchange.sol
        router-aggregator/contracts/interface/IWETH.sol
        router-aggregator/contracts/interface/IUniswapFactory.sol
        router-aggregator/contracts/interface/IUniswapV2Factory.sol
        router-aggregator/contracts/interface/IUniswapV2Exchange.sol
        router-aggregator/contracts/OneSplit.sol
        router-aggregator/contracts/libraries/TransferHelper.sol
        router-aggregator/contracts/IOneSplit.sol
        router-aggregator/contracts/UniversalERC20.sol
        router-aggregator/contracts/BEP20Token.sol
```

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

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ EIP standards violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency ▪ Kill-Switch Mechanism

Executive Summary

The score measurement details can be found in the corresponding section of the methodology.

Documentation quality

The Customer provided a litepaper that describes functional requirements and a high-level structure overview but no technical requirements. Some contracts are covered with autodoc comments. The total Documentation Quality score is **6** out of **10**.

Code quality

The total Code Quality score is **9** out of **10**. The style guide is mostly followed.

Architecture quality

The architecture quality score is **9** out of **10**.

Security score

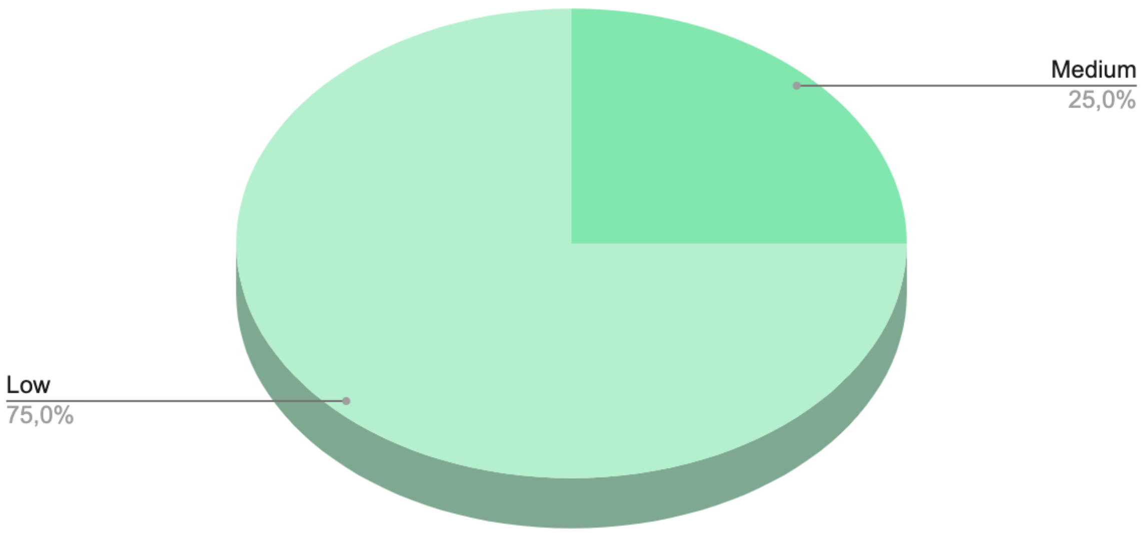
As a result of the audit, security engineers found **1** medium, and **2** low severity issues. The security score is **8** out of **10**. All found issues are displayed in the “Issues overview” section.

Summary

According to the assessment, the Customer's smart has the following score: **8.0**

1
2
3
4
5
6
7
8
9
10
You are here

Graph 1. The distribution of vulnerabilities after the audit.



AS-IS overview

FetchLiquidity.sol

Description

The contract provides view functions to get reserves for different DEXes that are used by path-finder-api

Imports

FetchLiquidity contract has the following import statements:

- ./IUniswapV2Factory.sol
- @openzeppelin/contracts/math/SafeMath.sol
- @openzeppelin/contracts/token/ERC20/IERC20.sol

Inheritance

FetchLiquidity contract has no inherited contracts.

Usages

FetchLiquidity contract has no custom usages.

Structs

FetchLiquidity contract has no data structures.

Enums

FetchLiquidity contract has no enums.

Events

FetchLiquidity contract has no events.

Modifiers

FetchLiquidity contract has no modifiers.

Fields

FetchLiquidity contract has following fields and constants:

- using SafeMath for uint256;
- using DisableFlags for uint256;
- uint256 internal constant DEXES_COUNT = 2;
- int256 internal constant VERY_NEGATIVE_VALUE = -1e72;
- uint256 internal constant FLAG_DISABLE_ALL_SPLIT_SOURCES = 0x20000000;
- IUniswapV2Factory internal constant uniswapV2 = IUniswapV2Factory(0x5C69bEe701ef814a2B6a3EDD4B1652CB9cc5aA6f);
- IUniswapV2Factory internal constant dfynExchange = IUniswapV2Factory(0xE7Fb3e833eFE5F9c441105EB65Ef8b261266423B);
- IUniswapV2Factory internal constant pancakeSwap = IUniswapV2Factory(0xcA143Ce32Fe78f1f7019d7d551a6402fC5350c73);
- uint256 internal constant FLAG_ENABLE_DFYN = 1;
- uint256 internal constant FLAG_ENABLE_UNISWAP_V2= 2;
- uint256 internal constant FLAG_ENABLE_PANCAKESWAP= 3;

Functions

FetchLiquidity has following external/public functions:

- getLiquidityReserves(IERC20[2][] calldata tokensIn, uint256 exchangeId)

BridgeUpgradeable.sol

Description

Facilitates deposits, creation and voting of deposit proposals, and deposit executions.

Imports

BridgeUpgradeable contract has the following import statements:

- @openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol
- @openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol
- @openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol
- @openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol
- @openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol
- ./interfaces/IVoterUpgradeable.sol
- ./interfaces/IERC20Upgradeable.sol
- ./interfaces/IDepositExecute.sol
- ./interfaces/ILiquidityPool.sol
- ./interfaces/IERCHandler.sol
- ./interfaces/IERCHandlerDecimals.sol
- ./interfaces/IGenericHandler.sol
- ./interfaces/IWETH.sol

Inheritance

BridgeUpgradeable contract has the following inheritance:

- Initializable
- PausableUpgradeable
- AccessControlUpgradeable
- UUPSUpgradeable
- ReentrancyGuardUpgradeable

Usages

BridgeUpgradeable contract has no custom usages.

Structs

BridgeUpgradeable contract has has the following data structures:

```
    struct proposalStruct {
        uint8 chainID;
        uint64 depositNonce;
        bytes32 dataHash;
        bytes32 resourceID;
    }
```

Enums

BridgeUpgradeable contract has no enums.

Events

BridgeUpgradeable contract has following events:

- event quorumChanged(uint64 quorum);
- event expiryChanged(uint256 expiry);
- event ProposalEvent(

```
        uint8 originChainID,
        uint64 depositNonce,
        IVoterUpgradeable.ProposalStatus status,
        bytes32 dataHash
    );
```

- event ProposalVote(
```
        uint8 originChainID,
        uint64 depositNonce,
        IVoterUpgradeable.ProposalStatus status,
        bytes32 dataHash
    );
```
- event Deposit(uint8 indexed destinationChainID, bytes32 indexed resourceID, uint64 indexed depositNonce);
- event Stake(address indexed staker, uint256 amount, address pool);
- event Unstake(address indexed unstaker, uint256 amount, address pool);
- event FeeSetterAdded(address feeSetter);
- event FeeSetterRemoved(address feeSetter);
- event AddedWhitelist(address whitelistAddress);
- event RemovedWhitelist(address whitelistAddress);
- event WhitelistingSetting(bool status);
- event AdminWithdraw(address handler, address tokenAddress, address recipient, uint256 amountOrTokenID);
- event Settlement(
```
        uint8 indexed originChainID,
        uint64 indexed depositNonce,
        address settlementToken,
        uint256 settlementAmount,
        IVoterUpgradeable.ProposalStatus status
    );
```
- event RelayerAdded(address relayer);
- event RelayerRemoved(address relayer);

Modifiers

BridgeUpgradeable contract has following modifiers:

- modifier onlyAdminOrRelayer()
- modifier isWhitelisted()
- modifier isWhitelistEnabled()
- modifier isResourceID(bytes32 _id)
- modifier isProposalExists(uint8 chainID,uint64 depositNonce, bytes32 dataHash)

Fields

BridgeUpgradeable contract has following fields and constants:

- uint8 private _chainID;
- uint256 private _expiry;
- bool private _whitelistEnabled;

- bytes32 public constant FEE_SETTER_ROLE = keccak256("FEE_SETTER_ROLE");
- bytes32 public constant RELAYER_ROLE = keccak256("RELAYER_ROLE");
- bytes32 public constant PAUSER_ROLE = keccak256("PAUSER_ROLE");
- bytes32 public constant RESOURCE_SETTER = keccak256("RESOURCE_SETTER");
- bytes32 public constant EMERGENCY_ROLE = keccak256("EMERGENCY_ROLE");
- uint256 public totalRelayers;
- uint64 public _quorum;
- IVoterUpgradeable public _voter;
- mapping(uint8 => uint64) private _depositCounts;
- mapping(bytes32 => address) private _resourceIDToHandlerAddress;
- mapping(bytes32 => uint256) private _proposals;
- mapping(address => bool) private _whitelist;
- mapping(uint256 => proposalStruct) private _proposalDetails;

Functions

BridgeUpgradeable has following external/public functions:

-
  ```
  function initialize(
    uint8 chainID,
    uint256 quorum,
    uint256 expiry,
    address voter
  ) external initializer
  ```

-
  ```
  function grantRole(bytes32 role, address account)
    public
    virtual
    override
    nonReentrant
    onlyRole(getRoleAdmin(role))
  ```

-
  ```
  function revokeRole(bytes32 role, address account)
    public
    virtual
    override
    nonReentrant
    onlyRole(getRoleAdmin(role))
  ```
- function addToWhitelist(address _beneficiary) public virtual onlyRole(DEFAULT_ADMIN_ROLE) isWhitelistEnabled
- function removeFromWhitelist(address _beneficiary) public virtual onlyRole(DEFAULT_ADMIN_ROLE) isWhitelistEnabled
- function setWhitelisting(bool value) public virtual onlyRole(DEFAULT_ADMIN_ROLE)
- function pause() public virtual onlyRole(PAUSER_ROLE) whenNotPaused
- function unpause() public virtual onlyRole(PAUSER_ROLE) whenPaused
- function adminChangeQuorum(uint256 newQuorum) public virtual onlyRole(DEFAULT_ADMIN_ROLE)

- function adminChangeExpiry(uint256 newExpiry) public virtual onlyRole(DEFAULT_ADMIN_ROLE)

-
  ```
  function adminSetResource(
    address handlerAddress,
    bytes32 resourceID,
    address tokenAddress
  ) public virtual onlyRole(RESOURCE_SETTER)
  ```

-
  ```
  function adminSetTokenDecimals(
    address handlerAddress,
    address tokenAddress,
    uint8 destinationChainID,
    uint8 decimals
  ) public virtual onlyRole(RESOURCE_SETTER)
  ```

-
  ```
  function adminSetOneSplitAddress(address handlerAddress, address
  contractAddress)
    public
    virtual
    onlyRole(DEFAULT_ADMIN_ROLE)
  ```

-
  ```
  function adminSetLiquidityPool(
    string memory name,
    string memory symbol,
    uint8 decimals,
    address handlerAddress,
    address tokenAddress,
    address lpAddress
  ) public virtual onlyRole(DEFAULT_ADMIN_ROLE)
  ```

-
  ```
  function adminSetLiquidityPoolOwner(
    address handlerAddress,
    address newOwner,
    address tokenAddress,
    address lpAddress
  ) public virtual onlyRole(DEFAULT_ADMIN_ROLE)
  ```

-
  ```
  function adminSetGenericResource(
    address handlerAddress,
    bytes32 resourceID,
    address contractAddress,
    bytes4 depositFunctionSig,
    uint256 depositFunctionDepositerOffset,
    bytes4 executeFunctionSig
  ) public virtual onlyRole(RESOURCE_SETTER)
  ```

-
  ```
  function adminSetBurnable(
    address handlerAddress,
    address tokenAddress,
    bool status
  ) public virtual onlyRole(DEFAULT_ADMIN_ROLE)
  ```

-
  ```
  function adminWithdraw(
    address handlerAddress,
    address tokenAddress,
  ```

```
  address recipient,
  uint256 amount
) public virtual nonReentrant onlyRole(EMERGENCY_ROLE)
```

- `function adminWithdrawFees(`
  ```
    address handlerAddress,
    address tokenAddress,
    address recipient,
    uint256 amount
  ) public virtual nonReentrant onlyRole(EMERGENCY_ROLE)
  ```
- `function adminSetFeeStatus(bytes32 resourceID, bool status) public virtual onlyRole(DEFAULT_ADMIN_ROLE)`
- `function setBridgeFee(`
  ```
    bytes32 resourceID,
    uint8 destinationChainID,
    address feeTokenAddress,
    uint256 transferFee,
    uint256 exchangeFee,
    bool accepted
  ) public virtual onlyRole(FEE_SETTER_ROLE)
  ```
- `function getBridgeFee(`
  ```
    bytes32 resourceID,
    uint8 destChainID,
    address feeTokenAddress
  ) public view returns (uint256, uint256)
  ```
- `function deposit(`
  ```
    uint8 destinationChainID,
    bytes32 resourceID,
    bytes calldata data,
    uint256[] memory distribution,
    uint256[] memory flags,
    address[] memory path,
    address feeTokenAddress
  ) public virtual nonReentrant whenNotPaused isWhitelisted
  ```
- `function stake(`
  ```
    address handler,
    address tokenAddress,
    uint256 amount
  ) public virtual whenNotPaused
  ```
- `function stakeETH(`
  ```
    address handler
  ) public payable virtual nonReentrant whenNotPaused
  ```
- `function unstake(`
  ```
    address handler,
    address tokenAddress,
    uint256 amount
  ) public virtual whenNotPaused
  ```
- `function unstakeETH(`
  ```
    address handler,
  ```

```
    uint256 amount
  ) public virtual nonReentrant whenNotPaused
```

-
  ```
  function getProposal(
    uint8 originChainID,
    uint64 depositNonce,
    bytes32 dataHash
  ) public view virtual
  ```

-
  ```
  function voteProposal(
    uint8 chainID,
    uint64 depositNonce,
    bytes32 resourceID,
    bytes32 dataHash
  ) public virtual isResourceID(resourceID) onlyRole(RELAYER_ROLE)
  whenNotPaused
  ```

-
  ```
  function cancelProposal(
    uint8 chainID,
    uint64 depositNonce,
    bytes32 dataHash
  ) public onlyAdminOrRelayer whenNotPaused
  ```

-
  ```
  function executeProposal(
    uint8 chainID,
    uint64 depositNonce,
    bytes calldata data,
    bytes32 resourceID,
    uint256[] memory distribution,
    uint256[] memory flags,
    address[] memory path
  ) public virtual onlyRole(RELAYER_ROLE) whenNotPaused
  ```

OneSplit.sol

Description

Perform swaps and calculations using external contracts.

Imports

OneSplit contract has the following import statements:

- @openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol
- @openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol
- @openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol
- @openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol
- @openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol
- @openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol
- ./interface/IUniswapFactory.sol
- ./interface/IUniswapV2Factory.sol
- ./interface/IHandlerReserve.sol
- ./interface/IEthHandler.sol
- ./interface/IBridge.sol
- ./IOneSplit.sol

- ./UniversalERC20.sol
- ./interface/IWETH.sol
- ./libraries/TransferHelper.sol
- hardhat/console.sol

Inheritance

OneSplit contract has the following inheritance:

- Initializable
- IOneSplit
- OneSplitRoot
- UUPSUpgradeable
- AccessControlUpgradeable

Usages

OneSplit contract has the following custom usages:

- using UniversalERC20 for IERC20Upgradeable;
- using SafeMathUpgradeable for uint256;
- using DisableFlags for uint256;
- using UniswapV2ExchangeLib for IUniswapV2Exchange;

Structs

OneSplit contract has no data structures.

Enums

OneSplit contract has no enums.

Events

OneSplit contract has no events.

Modifiers

OneSplit contract has the following modifiers:

- modifier onlyHandler()

Fields

OneSplit contract has following fields and constants:

- IOneSplitView public oneSplitView;
- address public handlerAddress;
- IHandlerReserve public reserveInstance;
- IBridge public bridgeInstance;
- IEthHandler private _ethHandler;

Functions

OneSplit has following external/public functions:

-
  ```
  function initialize(
      IOneSplitView _oneSplitView,
      address _handlerAddress,
      address _reserveAddress,
      address _bridgeAddress
  ) public initializer
  ```

-
  ```
  function getExpectedReturn(
      IERC20Upgradeable fromToken,
      IERC20Upgradeable destToken,
      uint256 amount,
      uint256 parts,
  ```

```
        uint256 flags
    ) public view override
```
- function getExpectedReturnETH(
```
        IERC20Upgradeable srcStableFromToken,
        uint256 srcStableFromTokenAmount,
        uint256 parts,
        uint256 flags
    ) public view override
```
- function getExpectedReturnWithGas(
```
        IERC20Upgradeable fromToken,
        IERC20Upgradeable destToken,
        uint256 amount,
        uint256 parts,
        uint256 flags,
        uint256 destTokenEthPriceTimesGasPrice
    ) public view override
```
- function getExpectedReturnWithGasMulti(
```
        IERC20Upgradeable[] memory tokens,
        uint256 amount,
        uint256[] memory parts,
        uint256[] memory flags,
        uint256[] memory destTokenEthPriceTimesGasPrices
    ) public view override
```
- function setHandlerAddress(address _handlerAddress) public override onlyRole(DEFAULT_ADMIN_ROLE)
- function setReserveAddress(address _reserveAddress) public override onlyRole(DEFAULT_ADMIN_ROLE)
- function setBridgeAddress(address _bridgeAddress) public override onlyRole(DEFAULT_ADMIN_ROLE)
- function setEthHandler(IEthHandler ethHandler) external onlyRole(DEFAULT_ADMIN_ROLE)
- function withdraw(
```
        address tokenAddress,
        address recipient,
        uint256 amount
    ) public payable override onlyHandler
```
- function swap(
```
        IERC20Upgradeable fromToken,
        IERC20Upgradeable destToken,
        uint256 amount,
        uint256 minReturn,
        uint256[] memory distribution,
        uint256 flags,
        bool isWrapper
    ) public payable override
```
- function swapMulti(
```
        IERC20Upgradeable[] memory tokens,
        uint256 amount,
        uint256 minReturn,
        uint256[] memory distribution,
        uint256[] memory flags,
        bool isWrapper
    ) public payable override
```

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution

Audit overview

🟥🟥🟥🟥 Critical

No critical issues were found.

🟨🟨🟨 High

No high severity issues were found.

🟩🟩 Medium

1. **Contracts that lock Ether.**

   The contract has a payable function but without a withdrawal capacity.

   Contracts: ERC20HandlerUpgradeable.sol

   Function: receive

   Recommendation: While this function is needed for swap to work, we would recommend adding a check inside to validate whether a swap router sent Ethers or not. Otherwise, add an eth-balance-recovery function just in case.

   Status: Reported

🟦 Low

1. **Missing events**

   Contracts: OneSplit.sol, FeeManagerUpgradeable.sol, VoterUpgradeable.sol, HandlerReserveUpgradeable.sol, HandlerHelpersUpgradeable.sol, ERC20HandlerUpgradeable.sol

   Functions: setHandlerAddress, setBridge, _setLiquidityPoolOwner, _setLiquidityPool, _setOneSplitAddress, _setBurnable, _setResource, setReserve, toggleFeeStatus, setFeeManager

   Changing critical values should be followed by the event emitting for better tracking off-chain.

   Recommendation: Emit events on the critical values changing.

   Status: Reported

2. **No License header**

   Each Solidity source file should consist SPDX-License-Identifier header.

   Contracts: OneSplit.sol, IOneSplit.sol, UniversalERC20.sol, IUniswapFactory.sol, IUniswapV2Factory.sol, IWETH.sol, IUniswapExchange.sol, IUniswapV2Exchange.sol,

   Recommendation: Add SPDX-License-Identifier to all Solidity files.

   Status: Reported

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit cannot guarantee the explicit security of the audited smart contracts.

Appendix A. Smart Contracts Security Issues

Category	Test Name	Result	Details
SWC-136	Unencrypted Private Data On-Chain	Passed	
SWC-135	Code With No Effects	Passed	
SWC-134	Message call with hardcoded gas amount	Passed	
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed	
SWC-132	Unexpected Ether balance	Passed	
SWC-131	Presence of unused variables	Passed	
SWC-130	Right-To-Left-Override control character (U+202E)	Passed	
SWC-129	Typographical Error	Passed	
SWC-128	DoS With Block Gas Limit	Passed	Not applicable in Solidity 0.8.x
SWC-127	Arbitrary Jump with Function Type Variable	Passed	
SWC-126	Insufficient Gas Griefing	Passed	Not applicable in Solidity 0.8.x
SWC-125	Incorrect Inheritance Order	Passed	
SWC-124	Write to Arbitrary Storage Location	Passed	
SWC-123	Requirement Violation	Passed	
SWC-122	Lack of Proper Signature Verification	Passed	
SWC-121	Missing Protection against Signature Replay Attacks	Passed	
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed	
SWC-119	Shadowing State Variables Function Default Visibility	Passed	
SWC-118	Incorrect Constructor Name	Passed	
SWC-117	Signature Malleability	Passed	
SWC-116	Block values as a proxy for time	Passed	
SWC-115	Authorization through tx.origin	Passed	
SWC-114	Transaction Order Dependence	Passed	
SWC-113	DoS with Failed Call	Passed	

SWC-112	Delegatecall to Untrusted Callee	Passed	
SWC-111	Use of Deprecated Solidity Functions	Passed	
SWC-110	Assert Violation	Passed	
SWC-109	Uninitialized Storage Pointer	Passed	Not applicable in Solidity 0.8.x
SWC-108	State Variable Default Visibility	Passed	
SWC-107	Reentrancy	Passed	
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed	
SWC-105	Unprotected Ether Withdrawal	Passed	
SWC-104	Unchecked Call Return Value	Passed	
SWC-103	Floating Pragma	Not passed	In Solidity 0.8.x floating pragma is not considered as a vulnerability
SWC-102	Outdated Compiler Version	Not passed	Version 0.8.2 has known bugs that do not affect the project but using version 0.8.9+ is recommended.
SWC-101	Integer Overflow and Underflow	Passed	Not applicable in Solidity 0.8.x
SWC-100	Function Default Visibility	Passed	

Appendix B. Other contracts attribute

Change Management – AccessControlUpgradeable is used

Source of Randomness – Not related. Contracts don't have randomness.

Timestamp Dependence – Contracts relate to the timestamp but nothing critical. Only used for interacting with external contracts when it’s necessary.

Hash Collisions – Not related. There are no such hash checks in contracts.

Deprecated Functions and Outdated Compiler Versions – Partially related. The compiler version used for most code is 0.8.2 which is not the latest but not outdated one, the other is 0.5.x which is outdated but it uses only in path-finder-api.

Reentrancy – Not related. ReentrancyGuardUpgradeable is used.

Race Conditions – No race conditions determined.

Delegate Call Injection – No delegate calls.

Denial of Service with Unexpected Revert – Not related.

Call Exception Management – Not related.

Smart Contract Owner Compromise – Not related. Separate roles are implemented to prevent the necessity of using owner permission.

Third-Party External Inputs – Only well-known external DEXes uses. No third-party Oracles were used.

Frozen Funds – Not related.

Manipulated Balance – Not related.

Erroneous Function Labeling – No such functions.

Caller Identity Validation – Every mutable function that changes contracts state/behavior has identity validation

Unexpected Throws – Not related.

Fee Limits – Not related.

Conflicting Transactions – Not related.

Transaction Order/Front Running – Not related.

Signature Verifications – Signature validation and verification is OK.

Malicious Transaction Relayer – Not related.