SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Unipump
Date: September 17th, 2020

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 Unipump
Type	Contract for creating and managing Unipump Groups.
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
File Name	UnipumpGroup.sol
SHA256 Checksum	12A4346A2976BDC86BD57DE2FAA845C439F517CF8C2586E2154DDE6950F9280B
Timeline	16TH SEP 2020 - 17TH SEP 2020
Changelog	17TH SEP 2020 - Initial Audit

Table of contents

Document....................2
Table of contents....................3
Introduction....................4
Scope....................4
Executive Summary....................6
Severity Definitions....................7
AS-IS overview....................8
Conclusion....................19
Disclaimers....................20

Introduction

Hacken OÜ (Consultant) was contracted by Unipump (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer`s smart contract and its code review conducted between September 16th, 2020 - September 17th, 2020.

Scope

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

File Name - UnipumpGroup.sol
SHA256 Checksum -
12A4346A2976BDC86BD57DE2FAA845C439F517CF8C2586E2154DDE6950F9280B

unipump-protocol-master/contracts/UnipumpGroup.sol	**In Scope of Review**
unipump-protocol-master/contracts/UnipumpGroupLibrary.sol	Not in Scope of Review
unipump-protocol-master/contracts/UnipumpStaking.sol	Not in Scope of Review
unipump-protocol-master/contracts/Unipump.sol	Not in Scope of Review
unipump-protocol-master/contracts/IUnipumpGroup.sol	Not in Scope of Review
unipump-protocol-master/contracts/uniswap/IUniswapV2Router01.sol	Not in Scope of Review
unipump-protocol-master/contracts/openzeppelin/ERC20.sol	Not in Scope of Review
unipump-protocol-master/contracts/UnipumpTokenListLibrary.sol	Not in Scope of Review
unipump-protocol-master/contracts/UnipumpEscrow.sol	Not in Scope of Review
unipump-protocol-master/contracts/weth/WETH9.sol	Not in Scope of Review
unipump-protocol-master/contracts/Multicall.sol	Not in Scope of Review
unipump-protocol-master/contracts/openzeppelin/Address.sol	Not in Scope of Review
unipump-protocol-master/contracts/uniswap/IUniswapV2Pair.sol	Not in Scope of Review
unipump-protocol-master/contracts/uniswap/IUniswapV2Router02.sol	Not in Scope of Review
unipump-protocol-master/contracts/openzeppelin/SafeMath.sol	Not in Scope of Review
unipump-protocol-master/contracts/UnipumpDrain.sol	Not in Scope of Review
unipump-protocol-master/contracts/UnipumpGroupData.sol	Not in Scope of Review
unipump-protocol-master/contracts/IUnipump.sol	Not in Scope of Review
unipump-protocol-master/contracts/IUnipumpStaking.sol	Not in Scope of Review
unipump-protocol-master/contracts/UnipumpContestProxy.sol	Not in Scope of Review
unipump-protocol-master/contracts/UnipumpErc20Helper.sol	Not in Scope of Review
unipump-protocol-master/contracts/UnipumpDefaults.sol	Not in Scope of Review
unipump-protocol-master/contracts/UnipumpTest.sol	Not in Scope of Review
unipump-protocol-master/contracts/uniswap/IUniswapV2Factory.sol	Not in Scope of Review
unipump-protocol-master/contracts/openzeppelin/IERC20.sol	Not in Scope of Review
unipump-protocol-master/contracts/test/ERC20Test.sol	Not in Scope of Review
unipump-protocol-master/contracts/openzeppelin/Context.sol	Not in Scope of Review

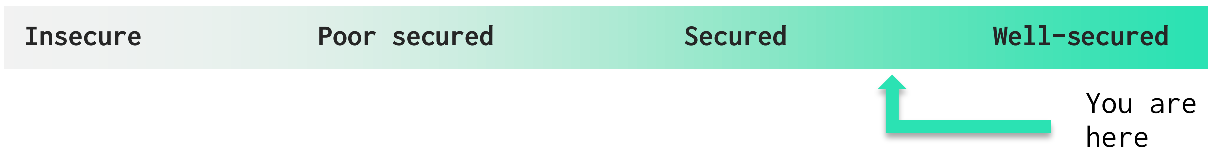
unipump-protocol-master/contracts/test/UnipumpDrainTest.sol	Not in Scope of Review
unipump-protocol-master/contracts/IUnipumpEscrow.sol	Not in Scope of Review
unipump-protocol-master/contracts/UnipumpGroupManagerProxy.sol	Not in Scope of Review
unipump-protocol-master/contracts/weth/IWETH.sol	Not in Scope of Review
unipump-protocol-master/contracts/IUnipumpDrain.sol	Not in Scope of Review
unipump-protocol-master/contracts/UnipumpContest.sol	Not in Scope of Review
unipump-protocol-master/contracts/IUnipumpContest.sol	Not in Scope of Review

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 ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

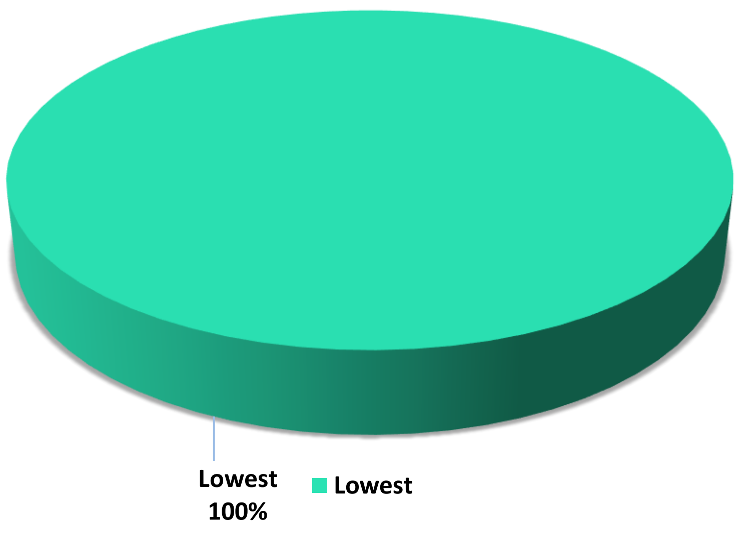
According to the assessment, the Customer's smart contract does not have high vulnerabilities and can be considered secure. But in this contract, most of the logic is imported from other contracts, so we cannot be sure that the entire system is secure.



Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section and all found issues can be found in the Audit overview section.

Security engineers found 1 lowest severity issues during audit.

Graph 1. The distribution of vulnerabilities.



Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets lose or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets lose or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

AS-IS overview

UnipumpGroup.sol

Description

UnipumpGroup is a contract for creating and managing Unipump Groups.

Imports

UnipumpGroup contract has 9 imports:

- *IERC20* – from OpenZeppelin;
- *UnipumpGroupData* – from project files;
- *UnipumpGroupDataMappings* – from project files;
- *UnipumpTokenList* – from project files;
- *UnipumpGroupLibrary* – from project files;
- *UnipumpTokenListLibrary* – from project files;
- *IUniswapV2Router02* – from project files;
- *IUniswapV2Router02* – from project files;
- *IUnipumpGroup* – from project files;

Inheritance

UnipumpGroup contract inherits *IUnipumpGroup*.

Usings

UnipumpGroup contract use:

- *UnipumpGroupLibrary* for *UnipumpGroupData*;
- *UnipumpTokenListLibrary* for *UnipumpTokenList*;

Structs

UnipumpGroup contract does not have data structs.

Fields

UnipumpGroup contract has 12 fields:

- *address immutable owner* – an owner address;
- *address immutable unipump* – an unipump address;
- *address immutable weth* – an wETH address;

- *IUniswapV2Router02 immutable uniswapV2Router* – a Router02 interface;
- *UnipumpGroupData[] groupData* – a list of *UnipumpGroupData*;
- *UnipumpGroupDataMappings[] groupDataMappings* – a list of *UnipumpGroupDataMappings*;
- *mapping (address => uint64[]) groupIdsByLeader* – a list of group IDs by leader address;
- *address[] public override groupLeaders* – a list of group leaders;
- *uint64[] public override groupIdsBeingStarted* – a list of group IDs;
- *uint64[] public override groupIdsRunning* – a list of group IDs;
- *mapping (uint64 => uint256) groupIdsIndex* – a list of indices by group IDs;
- *UnipumpTokenList[] tokenLists* – a list of *UnipumpTokenList*;

Modifiers

UnipumpGroup contract does not have data modifiers.

Informational functions

UnipumpGroup has 17 public view functions that are used to get data from contract fields:

- *tokenListCount()*
- *tokenListTokenCount(uint64 tokenListId)*
- *tokenListLocked(uint64 tokenListId)*
- *tokenListToken(uint64 tokenListId, uint256 at)*
- *group(uint64 groupId)*
- *groupCount()*
- *groupLeaderCount()*
- *groupIdsBeingStartedCount()*
- *groupIdsRunningCount()*
- *groupIdsByLeaderCount(address leader)*
- *groupIdByLeader(address leader, uint256 at)*
- *groupMemberCount(uint64 groupId)*
- *groupMember(uint64 groupId, uint256 at)*
- *groupContribution(uint64 groupId, address member)*
- *groupBalance(uint64 groupId, address token)*
- *groupMemberWithdrawal(uint64 groupId, address member, address token)*
- *authorizedTrader(uint64 groupId, address trader)*

Wrapper functions

UnipumpGroup has 10 functions that are wrappers for imported functions and do not provide any additional logic:

- *addTokensToList(uint64 tokenListId, address[] memory tokens)* – a public function that wraps UnipumpTokenList's *add(tokens)* function;
- *removeTokensFromList(uint64 tokenListId, address[] memory tokens)* – a public function that wraps UnipumpTokenList's *remove(tokens)* function;
- *tokenExistsInList(uint64 tokenListId, address token)* – a public function that wraps UnipumpTokenList's *exists(token)* function;
- *swapExactTokensForTokens(uint64 groupId, uint256 amountIn, uint256 amountOutMin, address[] memory path, uint256 deadline)* – a public function that wraps UnipumpGroupData's *swapExactTokensForTokens* function;
- *swapTokensForExactTokens(uint64 groupId, uint256 amountOut, uint256 amountInMax, address[] memory path, uint256 deadline)* – a public function that wraps UnipumpGroupData's *swapExactTokensForTokens* function;
- *swapExactTokensForTokensSupportingFeeOnTransferTokens(uint64 groupId, uint256 amountIn, uint256 amountOutMin, address[] memory path, uint256 deadline)* – a public function that wraps UnipumpGroupData's *swapExactTokensForTokensSupportingFeeOnTransferTokens* function;
- *addAuthorizedTrader(uint64 groupId, address trader)* – a public function that wraps UnipumpGroupData's *addAuthorizedTrader* function;
- *removeAuthorizedTrader(uint64 groupId, address trader)* – a public function that wraps UnipumpGroupData's *removeAuthorizedTrader* function;
- *withdraw(uint64 groupId, address token)* – a public function that wraps UnipumpGroupData's *withdraw* function;
- *withdrawMany(uint64 groupId, address[] memory tokens)* – a public function that wraps UnipumpGroupData's *withdrawMany* function;

Other functions

UnipumpGroup has 10 functions that are used to work with contract data:

- ***constructor***

  Description

Initializes contract.
Sets *owner*, *unipump*, *weth*, *uniswapV2Router* fields and owner of first *tokenLists* item.

Visibility

Default

Input parameters

- *address _unipump* – an unipump address;
- *address _weth* – a weth address;
- *IUniswapV2Router02 _uniswapV2Router* – a Router02

Constraints

- An address of unipump cannot be zero.
- An address of weth cannot be zero.
- An address of Router02 cannot be zero.

Events emit

None

Output

None

- ***createTokenList***

Description

Creates a token list.

Visibility

public

Input parameters

None

Constraints

None

Events emit

None

Output

Token list ID.

- ***lockTokenList***

  Description

  Wraps UnipumpTokenList's *lock()* function.

  Visibility

  public

  Input parameters

  - *uint64 tokenListId* – token list ID;

  Constraints

  - Token list with zero ID cannot be locked.

  Events emit

  None

  Output

  None

- ***create***

  Description

  Creates a group.

  Visibility

  public

  Input parameters

  - *uint64 tokenListId* – a token list ID;

- *uint16 maxRunTimeHours* – an amount of hours;
- *uint16 startTimeoutHours* – an amount of hours;
- *uint16 leaderProfitShareOutOf10000* – a profit that is distributed in favor of the leader;
- *uint256 leaderUppCollateral* – a leader collateral;
- *uint256 requiredMemberUppFee* – a member required fee;
- *uint256 minEthToJoin* – minimal amount of ETH to join;
- *uint256 minEthToStart* – minimal amount of ETH to start;
- *uint256 maxEthAcceptable* – maximum amount of ETH that can be accepted;

Constraints

- Token list ID must be exists.
- Token list must be locked or must have 0 ID.
- Maximum hours value must be greater than 0 and do not exceed 90 days.
- Start timeout hours value must be greater than 0 and do not exceed 30 days.
- Leader profit must be greater than 0 and do not exceed 10000.
- Required member fee must be greater or equal 0.01 ETH.
- Minimal ETH to join must be greater or equal 0.01 ETH.
- Minimal ETH to start must be greater or equal 1 ETH and must be greater or equal than minimal ETH to join.
- Maximum accepted ETH must be greater or equal than minimal ETH to start.
- Leader collateral must be greater or equal 1 ETH.

Events emit

- *GroupCreated(groupId)*

Output

Group ID.

- ***receive***

Description

Receives ETH.

Visibility

external payable

Input parameters

None

Constraints

- Caller must be weth.

Events emit

None

Output

None

- *join*

Description

Wraps UnipumpGroupData's *joinGroup* function.

Visibility

public payable

Input parameters

- *uint64 groupId* – group ID.

Constraints

None

Events emit

- *GroupContribution(groupId, msg.sender)*

Output

None

- *abort*

Description

Aborts group.

Visibility

public

Input parameters

- *uint64 groupId* – group ID.

Constraints

None

Events emit

- *GroupAborted(groupId)*

Output

None

- ***startPumping***

Description

Starts group pumping.

Visibility

public

Input parameters

- *uint64 groupId* – group ID.

Constraints

None

Events emit

- *GroupStarted(groupId)*

Output

None

- ***finish***

Description

Finish group.

Visibility

public

Input parameters

- *uint64 groupId* – group ID.

Constraints

None

Events emit

- *GroupFinished(groupId)*

Output

None

- ***emergencyWithdrawal***

Description

Emergency withdraw.

Visibility

public

Input parameters

- *uint64 groupId* – group ID;
- *address member* – an address of member;
- *address[] memory tokens* – a list of tokens addresses;

Constraints

- Caller must be owner.

Events emit

None

Output

None

Audit overview

🟥 🟥 🟥 🟥 Critical

No critical severity issues were found.

🟨 🟨 🟨 High

No high severity issues were found.

🟩 🟩 Medium

No medium severity issues were found.

🟦 Low

No low severity issues were found.

🟦 Lowest / Code style / Best Practice

1. According to the best practices, it is necessary to check if the array index is out of bounds.

Conclusion

Smart contracts within the scope was manually reviewed and analyzed with static analysis tools. For the contract high level description of functionality was presented in As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Violations in following categories were found and addressed to Customer:

Category	Check Item	Comments
Code review	■ Style guide violation	To avoid runtime errors, you need to check if the array index is out of bounds.

Security engineers found 1 lowest severity issues during audit, so this contract can be considered secure. But in this contract, most of the logic is imported from other contracts, so we cannot be sure that the entire system is secure.

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

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to: 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 security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. 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 security of smart contracts.

Technical Disclaimer

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