DeHacker

Code Security Assessment

Map Protocol Swap

November 15th, 2024

Contents

CONTENTS 1
SUMMARY 2
ISSUE CATEGORIES 3
OVERVIEW 4
PROJECT SUMMARY 4
VULNERABILITY SUMMARY 4
AUDIT SCOPE 5
FINDINGS 6
MAJOR 7
GLOBAL-01 : Centralization Risk **7**
DESCRIPTION 7
RECOMMENDATION 8
INFORMATIONAL 9
DKE-01 : Optimal Gas Optimization **9**
DESCRIPTION 9
RECOMMENDATION 10
INFORMATIONAL 11
OFD-01 : High Complexity **11**
DESCRIPTION 11
RECOMMENDATION 11
INFORMATIONAL 12
OFD-02 : Uninitialized Local Variables **12**
DESCRIPTION 12
RECOMMENDATION 14
DISCLAIMER 15
APPENDIX 16
ABOUT 17

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best prac tices. Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	Map Protocol Swap
Platform	Mapo
Website	https://www.mapprotocol.io/
Type	Bridge
Language	Func, Rust, Solidity
Codebase	https://github.com/mapprotocol/ton-router-contracts https://github.com/mapprotocol/router-contracts https://github.com/mapprotocol/solana-router-contracts

Vulnerability Summary

Vulnerability Level	Total	Mitigated	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	1	0	0	0	0	1
Medium	0	0	0	0	0	0
Minor	0	0	0	0	0	0
Informational	3	0	0	0	0	3
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
DKE	ton-router-contracts/ton_router_v2.fc	69d5234fe5a8c2917dbebdb1f06781fe374860182a11862426fd5ad8bd6dec6d
OFD	router-contracts/RouterPlusV2.sol	28bb6b7f3fd7e0a8bfa030eb216b57b33ee94e0954c3e8a08386d58ed8312c17
WEF	solana-router-contracts/router.rs	6b08a20072cbf93a9853bb27b65cd3516496431441a86800d831e1182e20658b

Findings

ID	Title	Severity	Status
GLOBAL-01	Centralization Related Risks	Major	Resolved
DKE-1	Optional Gas Optimization	Informational	Resolved
OFD-1	High Complexity	Informational	Resolved
OFD-2	Uninitialized Local Variables	Informational	Resolved

MAJOR

GLOBAL-01 | Centralization Related Risks

Issue	Severity	Location	Status
Centralization Related Risks	Major		Resolved

Description

The owner of the ton-router-contracts/ton_router_V2 contract has the following permissions:
- update_address
- update_owner

The owner will have the ability to influence the operation results of the protocol.
The owner has the authority to set the withdrawer, who has the authority to withdraw funds from the contract.

The owner of the router-contracts/RouterPlusV2 contract has the following permissions:
- function setAuthorization()
- function setGasForReFund()
- function setBridgeAddress()
- function setWToken()
- function editFuncBlackList()
- function setFee()
- function rescueFunds()

The owner will have the ability to influence the operation results of the protocol.

Recommendation

This finding describes the level of decentralization of the project, and it is recommended to strengthen security and improve the degree of decentralization from the following aspects:

- It is recommended that privileged addresses use multi-signature wallet addresses.
- For modification operations that affect protocol operation stability and key business parameters, it is recommended to implement time locks.

INFORMATIONAL

DKE-1 | Optional Gas Optimization

Issue	Severity	Location	Status
Optional Gas Optimization	Informational	ton-router-contracts/ton_router_v2.fc	Resolved

Description

Line 41, 43

```
() recv_internal(int my_balance, int msg_value, cell in_msg_full, slice in_msg_body) impure {
    if (in_msg_body.slice_empty?()) { ;; ignore all empty messages
        return ();
    }

    slice cs = in_msg_full.begin_parse();
    int flags = cs~load_uint(4);
    if (flags & 1) { ;; ignore all bounced messages
        return ();
    }

    slice sender_address = cs~load_msg_addr();
    int op = in_msg_body~load_uint(32); ;; by convention, the first 32 bits of incoming message is the op
    int query_id = in_msg_body~load_uint(64); ;; also by convention, the next 64 bits contain the "query id", although this is not always the case

    (int order_id, int swap_id, slice owner, slice withdrawer, slice bridger, slice bridge_token_address) = load_data();
```

Description

```
if (op == op::swap_from_ton) {
handle_swap_from_ton(msg_value, in_msg_body);
} elseif (op == op::withdrawer::withdraw_ton) {
handle_withdraw_ton(withdrawer, sender_address);
} elseif (op == op::withdrawer::withdraw_jetton) {
handle_withdraw_jetton(withdrawer, sender_address, query_id, in_msg_body);
} elseif (op == op::bridger::bridge_in) {
handle_bridge_in(sender_address, bridger, in_msg_body);
} elseif (op == op::admin::update_address) {
handle_update_address(sender_address, in_msg_body);
} elseif (op == op::admin::update_owner) {
handle_update_owner(sender_address, in_msg_body);
} elseif (op == op::transfer_notification) {
handle_transfer_notification(sender_address, in_msg_body);
} elseif (op == op::dedust::payout) {
handle_dedust_payout(sender_address, in_msg_body, msg_value);
} else {
throw(0xffff); ;; if the message contains an op that is not known to this contract,
we throw
}
}
```

When executing handle_update_address and handle_update_owner, internally using load_data to fetch and parse the stored data again increases some additional gas consumption.

Recommendation

This is an optional suggestion, as limited by the mechanisms of Ton and Func, although it is possible to directly pass in already parsed data from the outside, doing so would reduce the maintainability of the code. It is not recommended to do this in the early stages of contract development. If the contract has been finalized and there is a clear need to optimize the execution efficiency and cost at this point, consideration can be given to optimizing this option.

INFORMATIONAL

OFD-01 | High Complexity

Issue	Severity	Location	Status
High Complexity	Informational	router-contracts/RouterPlusV2.sol	Resolved

Description

Line 221

```solidity
function onReceived(
    bytes32 _orderId,
    address _srcToken,
    uint256 _amount,
    uint256 _fromChain,
    bytes calldata _from,
    bytes calldata _swapAndCall
) external override nonReentrant {
    ...
}
```

The `onReceived` method has a high complexity.

Recommendation

Minimize conditional branches and complex logic in functions, break down large functions, and follow the Single Responsibility Principle.

Informational

OFD-2 | Uninitialized Local Variables

Issue	Severity	Location	Status
Uninitialized Local Variables	Informational	router-contracts/RouterPlusV2.sol	Resolved

Description

Line 100
```solidity
function swapAndBridge(
    address _srcToken,
    uint256 _amount,
    bytes calldata _swapData,
    bytes calldata _bridgeData,
    bytes calldata _permitData
) external payable nonReentrant transferIn(_srcToken, _amount, _permitData) {
    require(_swapData.length + _bridgeData.length > 0,
ErrorMessage.DATA_EMPTY);
    SwapTemp memory swapTemp;
    ...
}
```

Line 162
```
function swapAndCall(
    bytes32 _transferId,
    address _initiator, // initiator address
    address _srcToken,
    uint256 _amount,
```

Description

```
 bytes calldata _swapData,
 bytes calldata _callbackData,
 bytes calldata _permitData,
 bool _feeBeforeSwap
) external payable override nonReentrant {
 SwapTemp memory swapTemp;
 swapTemp.initiator = _initiator;
 swapTemp.srcToken = _srcToken;
 swapTemp.srcAmount = _amount;
 swapTemp.transferId = _transferId;
 (swapTemp.nativeBalance, swapTemp.inputBalance) = _transferIn(
 swapTemp.srcToken,
 swapTemp.srcAmount,
 _permitData
 );
 if ((_swapData.length + _callbackData.length) == 0) revert Errors.DATA_EMPTY();

    (
        swapTemp.receiver,
        swapTemp.target,
        swapTemp.swapToken,
        swapTemp.swapAmount,
        swapTemp.callAmount
    ) = _doSwapAndCall(
        swapTemp.transferId,
        swapTemp.srcToken,
        swapTemp.srcAmount,
        swapTemp.inputBalance,
        _swapData,
        _callbackData,
        _feeBeforeSwap
    );

    if (swapTemp.swapAmount > swapTemp.callAmount) {
        _transfer(swapTemp.swapToken, swapTemp.receiver,
(swapTemp.swapAmount - swapTemp.callAmount));
    }
```

```
    emit SwapAndCall(
        TW_REFERRER,
        swapTemp.initiator,
        msg.sender,
        swapTemp.transferId,
        swapTemp.srcToken,
        swapTemp.swapToken,
        swapTemp.srcAmount,
        swapTemp.swapAmount,
        swapTemp.receiver,
        swapTemp.target,
        swapTemp.callAmount
    );
    _afterCheck(swapTemp.nativeBalance);
}
```

Uninitialized local variables.

Recommendation

Initialize all the variables.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege
Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style
Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code
Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue
Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

- Ethereum
- Cosmos
- Eos
- Substrate

TECH STACK

- Python
- Solidity
- Rust
- c++

CONTACTS

- https://dehacker.io
- https://twitter.com/dehackerio
- https://github.com/dehacker/audits_public
- https://t.me/dehackerio
- https://blog.dehacker.io/

November 15th 2024