DeHacker

Code Security Assessment

MELD - ERC20 Audit

Aug 15th, 2023

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.
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	MELD - ERC20 Audit
Platform	Ethereum
Website	https://www.meld.com/
Type	Defi
Language	Solidity

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	2	0	0	1	0	1
Minor	1	0	0	0	0	1
Informational	1	0	0	0	0	1
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
BTM	contracts/BaseToken.sol	51de7bd64a2e66591afd4f4bd3e2585f2f24c61d3e9d0dc7216309cc095405c7
MTM	contracts/MeldToken.sol	7e8bb8b7b229b626913a1eb33a92cb8b0cd2b8bcccc9a9d25e39fac8817572b0
RTM	contracts/utils/RescueTokens.sol	29517b8dde830d8d574bc58a79651ee284c4ce8ee87a3000fb7429636d4cb4e6

Findings

ID	Category	Severity	Status
BTM-02	Control Flow	Medium	Resolved
BTM-02	Volatile Code	Medium	Acknowledged
MEL-01	Volatile Code	Minor	Resolved
BTM-02	Logical Issue	Informational	Resolved

INFORMATIONAL

CCK-01 | Missing Emit Events

Category	Severity	Location	Status
Medium	Medium	contracts/ BaseToken.sol	Resolved

Description

The contract BaseToken adopts the eip-2771 standard which overrides the function _msgSender() and uses the last 20bytes of the call data instead of msg.sender if the msg.sender is the forwarder. Meanwhile, the function _checkRole() inthe OpenZeppelin contract AccessControl is using the function _msgSender() for the role checks. Thus, the forwardercan forge the value of _msgSender() and effectively send transactions from privileged addresses.

Recommendation

We recommend adding require statement like below to block the forwarder to call the centralized functions likesetMintingAccount() , setTrustedForwarder() , pause() , unpause () , mint() , burn() , rescueERC20() ,rescueERC721() , rescueERC1155() , etc.

```
  require(!isTrustedForwarder(msg.sender), "EIP2771Recipient: meta transaction is
not allowed");
```

INFORMATIONAL

MEL-02|UNKNOW TRUSTED FORWARDER IMPLEMENTATION

Category	Severity	Location	Status
Volatile Code	Medium	contracts/ BaseToken.sol; contracts/ MeldToken.sol	Acknowledged

Description

The variable _trustedForwarder serves as a 3rd party forwarder to help the ERC2771Recipient identify the address ofthe transaction signer. BaseToken and MeldToken inherit ERC2771Recipient , which decodes the msg.sender frommsg.data when called from the forwarder by overriding the _msgSender() function. A malicious forwarder may forge thevalue of _msgSender() and effectively send transactions from any address. The scope of the audit treats these 3rd partyforwarders as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can becompromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severeimpacts.

Reference: EIP-2771

Recommendation

We understand that the business logic of BaseToken and MeldToken requires interaction with third-party forwardercontracts. We encourage the team to ensure their functional correctness and constantly monitor the statuses of 3rd parties tomitigate the side effects when unexpected activities are observed.

INFORMATIONAL

MEL-01|MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	contracts/BaseToken .sol: 38; contracts/ utils/RescueTokens. sol: 19, 29,38	Resolved

Description

The cited addresses input are missing a check that it is not address(0) .
For example:
_token in the RescueTokens.
_rescueERC20() , RescueTokens.
_rescueERC721() ,RescueTokens.
_rescueERC1155() ._defaultAdmin in the BaseToken.constructor() .
_account in the BaseToken.setMintingAccount() .
_forwarder in the BaseToken.setTrustedForwarder() .

Recommendation

We recommend adding a check the passed-in address is not address(0) to prevent unexpected errors.

INFORMATIONAL

BTM-03|LOGIC RELATED TO MINTING PERIOD

Category	Severity	Location	Status
Volatile Code	Minor	contracts/BaseToken .sol: 38; contracts/ utils/RescueTokens. sol: 19, 29,38	Resolved

Description

According to the following code snippet in the function mint() , each new minter will start a new minting period in the firstmint, and any further mint operation from this minter within the period cannot exceed the threshold limit. The first mintoperation after the period will clear the minted amount(currentMintingAmount[minter]) and start a new period, so theminter can mint more tokens below the threshold. If the mintingPeriodLength[minter] is set to 0, this logic is like amaximum mint limit for a single mint transaction.

```
113         if (block.timestamp > lastMintingPeriod[minter] +
mintingPeriodLength[minter]) {
114             currentMintingAmount[minter] = 0;
115             lastMintingPeriod[minter] = block.timestamp;
116         }
117         require(
118             currentMintingAmount[minter] + _amount <=
mintingAmountThreshold[minter],
119             "BaseToken: Minting amount exceeds threshold"
120         );
121
122         currentMintingAmount[minter] += _amount;
```

Recommendation

We would like to confirm with the client whether the current implementation aligns with the original project design.

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/

Aug 2023