DeHacker

Code Security Assessment

MILC

Aug3 rd, 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	MILC
Platform	BSC
Website	https://www.milc.global/
Type	meme
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	0	0	0	0	0	0
Minor	1	1	0	0	0	0
Informational	3	3	0	0	0	0
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
MLT	MLT.sol	c04f47b542f2fd6ff8118004e63d664cbd3b8f512a931077c4445610e596d9dc

Findings

ID	Category	Severity	Status
MLT-01	Logical Issue	Informational	Pending
MLT-02	Gas Optimization	Informational	Pending
MLT-03	Gas Optimization	Informational	Pending
MLT-04	Centralization / Privilege	Minor	Pending

INFORMATIONAL

MLT-01 | Inconsistent Comment Implementation

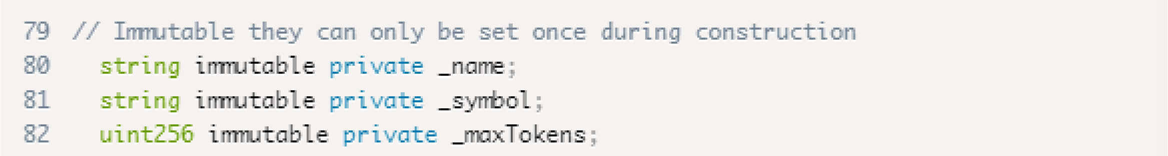
Category	Severity	Location	Status
Logical Issue	Informational	MLT.sol: 79~82	Pending

Description

The aforementioned line's comment described the variables _name , _symbol , _maxTokens should bedeclared as immutable for only be set once during construction

Recommendation

We recommend declaring the variables _name , _symbol , _maxTokens as immutable state to match thecomment description.



```
79  // Immutable they can only be set once during construction
80    string immutable private _name;
81    string immutable private _symbol;
82    uint256 immutable private _maxTokens;
```

INFORMATIONAL

MLT-02 | Proper usage of public and external type

Category	Severity	Location	Status
Gas Optimization	Informational	MLT.sol: 102, 107, 112, 117, 122, 127, 132, 141, 147, 155, 161, 172, 178, 191, 195, 199	Pending

Description

Public functions that are never called by the contract could be declared external . When the inputs arearrays external functions are more efficient than public functions.

Example functions :
name() ;
symbol() ;
decimals() ;
totalSupply() ;
balanceOf(address) ;
allowance(address,address) ;
isBlocked() ;
transfer(address,uint256)er ;
transferArray(address[],uint256[])er ;
transferFrom(address,address,uint256)er ;
approve(address,uint256) ;
increaseAllowance(address,uint256) ;
decreaseAllowance(address,uint256) ;
burn(uint256) ;blockAddress(address) ;
_block (address,bool) ;

Recommendation

We recommend using the external attribute for functions never called from the contract.

INFORMATIONAL

MLT-03 | Lack of Input Validation

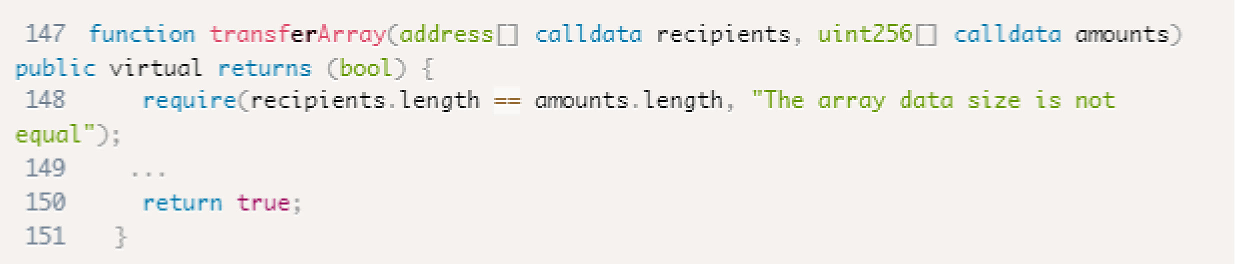
Category	Severity	Location	Status
Gas Optimization	Informational	MLT.sol: 147	Pending

Description

In aforementioned function transferArray() , the input parameters recipients && amounts are not sanitized with the array data size.

Recommendation

We recommend adding the data size validation recipients && amounts for ensuring the data length.



```
147  function transferArray(address[] calldata recipients, uint256[] calldata amounts) public virtual returns (bool) {
148      require(recipients.length == amounts.length, "The array data size is not equal");
149     ...
150      return true;
151    }
```

Minor

MLT-04 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	Minor	MLT.sol: 191	Pending

Description

The account owner of only owner role has authority to the functionser burn() and blockAddress() ,and unblockAddress()`. Any compromise to this account may allow the hacker to take advantage of these two functions and eventually drain all tokens from the contract.
burn() : The owner can burn an arbitrary amount of token.
blockAddress() : The owner can block an arbitrary address.
unblockAddress() : The owner can unblock an arbitrary address.

Recommendation

We advise the client to carefully manage the role onlyOwnerer 's account private key and avoid any potentialrisks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol toerbe improved via a decentralized mechanism or via smart-contract-based accounts with enhanced securitypractices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

Time-lock with reasonable latency, i.e., 48 hours, for awareness on privileged operations;
Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to theprivate key;

Introduction of a DAO / governance/voting module to increase transparency and user involvement.

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

Eos

Cosmos

Substrate

TECH STACK

Python

Rust

Solidity

c++

CONTACTS

https://dehacker.io

https://twitter.com/dehackerio

https://github.com/dehacker/audits_public

https://t.me/dehackerio

https://blog.dehacker.io/

DeHacker

Aug 2023