Report on Security Assessment for Oroundo Mobile GmbH smart contract

Version 1.0
06 May 2022

By: Ilya A. Shlyakhovoy

Table of Contents

1. Introduction 3

1.1 Audit details 4

1.2 Audit Goals 4

2. Source code audit Details 5

2.1 Risk analysis legend 5

2.2 No. of issue per severity 5

2.2 Audit findings 5

3. Source code testing 7

4. Test cases and coverage 10

5. Summary 10

6. About the auditor 11

1. Introduction

This audit report highlights the overall security of the CPTC ecosystem smart contracts. With this report, I have tried to ensure the reliability of the smart contract by completing the assessment of their system's architecture and smart contract codebase.

Auditing approach and Methodologies applied

In this audit, I consider the following crucial features of the code.

- Whether the implementation of protocol standards.
- Whether the code is secure.
- Whether the code meets the best coding practices.
- Whether the code meets the SWC Registry issue.

The audit has been performed according to the following procedure:

• Manual audit

1. Inspecting the code line by line and revert the initial algorithms of the protocol and then compare them with the specification
2. Manually analyzing the code for security vulnerabilities.
3. Assessing the overall project structure, complexity & quality.
4. Checking SWC Registry issues in the code.
5. Unit testing by writing custom unit testing for each function.
6. Checking whether all the libraries used in the code of the latest version.
7. Analysis of security on-chain data.
8. Analysis of the failure preparations to check how the smart contract performs in case of bugs and vulnerability.

• Automated analysis

1. Scanning the project's code base with Mythril, Slither, Echidna , Manticore other's.
2. Manually verifying (reject or confirm) all the issues found by tools.
3. Performing Unit testing.
4. Manual Security Testing (SWC-Registry, Overflow)
5. Running the tests and checking their coverage.

Report: All the gathered information is described in this report.

1.1 Audit details

Project Name: CPTC ecosystem with commit 6405da6

Token symbol:

Token Supply:

Language: Solidity

Platform and tools: Remix, VScode, securify and other tools mentioned in the automated analysis section.

1.2 Audit Goals

The focus of this audit was to verify whether the smart contract is secure, resilient, and working properly according to the specs. The audit activity can be grouped in three categories.

Security: Identifying the security-related issue within each contract and system of contracts.

Architecture: Evaluating the architect of a system through the lens of established smart contract best practice and general software practice.

Code correctness and quality: A full review of contract source code. The primary area of focus includes.

- Correctness.
- Section of code with high complexity.
- Readability.
- Quantity and quality of test coverage.

2. Source code audit Details

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

2.1 Risk analysis legend

Impact	Description
High	Issues mentioned here are critical to smart contract performance and functionality and should be fixed before moving to mainnet.
Medium	This could potentially bring the problem in the future and should be fixed.
Low	These are minor details and warnings that can remain unfixed but would be better if it got fixed in the future.

2.2 No. of issue per severity

Severity	High	Medium	Low
Open	0	0	6

2.3 Audit findings

Manual audit

Following are the reports from our manual analysis.

High severity issues

No High Severity Issue found.

Medium severity issues

No Medium Severity Issue found.

Low Severity Issues :

There were 6 low severity issues found in the contracts.

1. Compiler version

solc-0.8.13 is not recommended for deployment In the token cptcToken.sol

2. 0 Address for Mints & Burns

Although not technically part of the EIP20 specification, it is common practice to use the zero address as the source for all Transfer events after minting, and as the destination for Transfers upon burning tokens. The constructor of the contract use the provided address instead. Consider using the zero address instead or informing users and developers of this feature.

3. Source code testing

Automated test:

We have used multiple automated testing frameworks. This makes code more secure common attacks. The results are below.

Mythril:

Mythril automatically checks Smart Contracts for vulnerabilities and bad practices.

No vulnerabilities found.

Slither:

Slither is a Solidity static analysis framework which runs a suite of vulnerability detectors, prints visual information about contract details, and provides an API to write custom analyses quickly. Slither enables developers to find vulnerabilities, enhance their code comprehension, and promptly prototype custom analyses. Each solidity file and project together has been analyzed. We got a report with a few warnings and errors.

Below are the results.

```
Pragma version^0.8.0 (polygon/contracts/Migrations.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
(polygon/contracts/ERC20Permit.sol#37-78) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,ERC20Permit: expired deadline)
(polygon/contracts/ERC20Permit.sol#46)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should
be removed
Counters.decrement(Counters.Counter) (node_modules/@openzeppelin/contracts/utils/Counters.sol#32-38) is
never used and should be removed
Counters.reset(Counters.Counter) (node_modules/@openzeppelin/contracts/utils/Counters.sol#40-42) is never
used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#4) allows old versions
```

Pragma version^0.8.0 (polygon/contracts/CptcHub.sol#3) allows old versions
Pragma version^0.8.0 (polygon/contracts/ERC20Permit.sol#2) allows old versions
Pragma version^0.8.0 (polygon/contracts/IERC2612Permit.sol#2) allows old versions
Pragma version^0.8.0 (polygon/contracts/cptcToken.sol#3) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
symbol() should be declared external:
- ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
- ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
- ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
transfer(address,uint256) should be declared external:
- ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
- ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
- ERC20.transferFrom(address,address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
- ERC20.increaseAllowance(address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
- ERC20.decreaseAllowance(address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (polygon/contracts/CptcHub.sol#3) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
(polygon/contracts/ERC20Permit.sol#37-78) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp <= deadline,ERC20Permit: expired deadline)
(polygon/contracts/ERC20Permit.sol#46)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
Counters.decrement(Counters.Counter) (node_modules/@openzeppelin/contracts/utils/Counters.sol#32-38) is never used and should be removed
Counters.reset(Counters.Counter) (node_modules/@openzeppelin/contracts/utils/Counters.sol#40-42) is never used and should be removed
ERC20._burn(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#280-295) is never used and should be removed
ERC20._mint(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#257-267) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#4) allows old versions
Pragma version^0.8.0 (polygon/contracts/ERC20Permit.sol#2) allows old versions
Pragma version^0.8.0 (polygon/contracts/IERC2612Permit.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

symbol() should be declared external:
- ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
- ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
- ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
- ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
- ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
- ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
- ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
- ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
- ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

Pragma version^0.8.0 (polygon/contracts/IERC2612Permit.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

4. Test cases and coverage

Passed test cases: 20 of 20. Both positive and negative conditions are passed.

Manual analysis:

Manual analysis consists of code review and visual analysis of the functions. The code is well structured and clear.

CptcHub.sol

3. uint256 in mapping

Line 10: **mapping(address => uint256) contractAuthorisationList;**
Can be reduced to the uint8 can reduce the gas consumption without losing of the functionality.

4. zero address in the function call

Line 14: **unction setContractAddress(...**
Function allow to use 0x0 address, it can be unsafe.

5. packed value in call

Line 15, line 31: **bytes32 index = keccak256(abi.encodePacked(contractName));**
Using of the prepacked bytes32 instead of string can reduce the gas consumption.

cptcToken.sol

6. can be used bitwise operations

Line 32, line 38
Using of the '&' operator instead of comparisons will reduce the gas consumption.

5. Summary

The use of smart contracts is simple and the code is relatively small. Altogether, the code is written and demonstrates effective use of abstraction, separation of concern, and modularity. But there are a few issues/vulnerabilities to be tackled at various security levels, it is recommended to fix them before deploying the contract on the main network.

6. About the Auditor: Ilya A. Shlyakhovoy

Ilya is a technology expert with many years of expert experience building, managing, and automating systems at scale for blockchain, distributed systems, and storage projects.

Started a career as a web developer at 1998, last 5 years he work with Solidity language.

He has developed many smart contracts in financial and b2b areas