October 08, 2021

Smart Contract Assessment – Optics-Core

Prepared by FTI Consulting

Overview

This report has been prepared for cLabs in review of Optics-core smart contracts to identify issues and vulnerabilities in the source code as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The assessment process pays particular attention to the following considerations:

- Testing smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards. Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase.

The security assessment did not result in findings that were critical, only medium, and low. FTI recommends addressing these findings to ensure security standards as well as enhancing general coding practices for better structure of source code as described in Detailed Findings.

Project Summary

Project Name	Optics-Core Smart Contracts
Description	The assessment is comprised of code review of all smart contracts within the Optics-core repository

Platform	EVM Compatible Chains
Language	Solidity
Scope and Codebase	optics-core-contracts: https://github.com/celo-org/optics-monorepo/tree/main/solidity/optics-core/contracts Commit: 3e255d5f7417049542ffaad2dfb78d52598368a1

Assessment Summary

Delivery Date	Oct 8, 2021
Assessment Methodology	Static Analysis, Manual Review

Vulnerability Summary

Total Issues	**10**
Medium	2
Low	8

FTI CONSULTING

Overview of All Contracts Assessed

Number	Contract	Name	General Assessment
1	CMN-01	Common.sol	No changes required
2	HM-01	Home.sol	Gas Optimization
3	MRK-01	Merkle.sol	No changes required
4	QUE-01	Queue.sol	No changes required
5	RPLA-01	Replica.sol	Gas Optimization and Logical Issue
6	UM-01	UpdateManager.sol	Gas Optimization and Logical Issue
7	V0-01	Version0.sol	No changes required
8	XACM-01	XAppConnectionManager.sol	No changes required
9	GM-01	GovernanceMessage.sol	Gas Optimization
10	GR-01	GovernanceRouter.sol	Coding Style, Volatile Code, Gas Optimization
11	UB-01	UpgradeBeacon.sol	Logical Issue and Gas Optimization
12	UBC-01	UpgradeBeaconController.sol	Volatile Code
13	UBP-01	UpgradeBeaconProxy.sol	Volatile Code

Findings – Vulnerability Summary

ID	Title	Category	Severity	Status
UM-01	contracts/UpdaterManager.sol	Logical Issue	Medium	Requires Attention
UB-01	contracts/upgrade/UpgradeBeacon.sol	Logical Issue	Medium	Requires Attention
GR-01	contracts/governance/GovernanceRouter.sol	Coding Style; Volatile Code; Gas Optimization	Low	Requires Attention
GM-01	contracts/governance/GovernanceMessage.sol	Gas Optimization	Low	Requires Attention
UB-02	contracts/upgrade/UpgradeBeacon.sol	Volatile Code	Low	Requires Attention
UBC-01	contracts/upgrade/UpgradeBeaconController.sol	Volatile Code	Low	Requires Attention
UBP-01	contracts/upgrade/UpgradeBeaconProxy.sol	Volatile Code	Low	Requires Attention

HM-01	contracts/Home.sol	Gas Optimization	Low	Requires Attention
UM-02	contracts/UpdaterManager.sol	Gas Optimization; Volatile Code	Low	Requires Attention
RPLA-01	contracts/Replica.sol	Gas Optimization; Volatile Code	Low	Requires Attention

Detailed Findings

Medium

1. UM-01 (UpdaterManager.sol)

ID	Title	Category	Severity	Status
UM-01	contracts/UpdaterManager.sol	Logical Issue	Medium	Requires Attention

Line 55-57 Function

constructor(address _updaterAddress) payable Ownable()

Issue

The constructor is payable but does not have a function to withdraw the ether which can lead to a locking of ether.

Recommendation

Confirm that the lack of withdrawal functionality is intentional. If locking of ether is a requirement then this is a non-issue.

2. UB-01 (UpgradeBeacon.sol)

ID	Title	Category	Severity	Status
UB-01	contracts/upgrade/UpgradeBeacon.sol	Logical Issue	Medium	Requires Attention

Line 44-47 Function

constructor(address _initialImplementation, address _controller) payable

Line 58-77 Function

fallback() external payable

Issue

The constructor is payable but does not have a function to withdraw the ether which can lead to a locking of ether.

Recommendation

Confirm that the lack of withdrawal functionality is intentional. If locking of ether is a requirement then this is a non-issue.

Low

1. GR-01 (GovernanceRouter.sol)

ID	Title	Category	Severity	Status
GR-01	contracts/governance/GovernanceRouter.sol	Coding Style, Volatile Code, Gas Optimization	Low	Requires Attention

Line 164-180 Function

GovernanceRouter.initialize(address,address)
recoveryManager = _recoveryManager (governance/GovernanceRouter.sol#173)

Issue

The code sets recoveryManager without emitting an event which is difficult to track off-chain.

Recommendation

Emit an event for critical parameter changes.

Zero address validation

Line 164-180 Function

GovernanceRouter.initialize(address,address)
recoveryManager = _recoveryManager (governance/GovernanceRouter.sol#173)

Line 281-287 Function

GovernanceRouter.transferRecoveryManager(address)._newRecoveryManager
recoveryManager = _newRecoveryManager (governance/GovernanceRouter.sol#286)

Issue

The function does not check for a zero-address.

Recommendation

Before setting the recoveryManager check for non-zero address.

Declare external

Line 164-180 Function

GovernanceRouter.initialize(address,address)

Issue

The function is declared with visibility mode as public instead of external. The function is never called by the contract.

Recommendation

For gas optimization use the external attribute for functions if they are not called from the contract but are used externally.

2. GM-01 (GovernanceMessage.sol)

ID	Title	Category	Severity	Status
GM-01	contracts/governance/GovernanceMessage.sol	Gas Optimization	Low	Requires Attention

Line 191-193 Function

GovernanceMessage.isCall(bytes29)

Line 265-267 Function

GovernanceMessage.isSetRouter(bytes29)

Line 225-229 Function

GovernanceMessage.isTransferGovernor(bytes29)

Line 134-136 Function

GovernanceMessage.messageType(bytes29)

Line 202-204 Function

GovernanceMessage.mustBeCalls(bytes29)

Issue

The above functions are not used in the contract and are considered dead code.

Recommendation

Remove unused functions for better gas optimization.

3. UB-02 (UpgradeBeacon.sol)

ID	Title	Category	Severity	Status
UB-02	contracts/upgrade/UpgradeBeacon.sol	Volatile Code	Low	Requires Attention

Zero address validation

Line 44-47 Function

constructor(address _initialImplementation, address _controller) payable
controller = _controller (upgrade/UpgradeBeacon.sol#46)

Issue

The function does not check for a zero-address.

Recommendation

Before using address _controller check for non-zero address.

Inline Assembly

Line 58-77 Function

Inline assembly (function line : 63-66)
Inline Assembly (upgrade/UpgradeBeacon.sol#63-66)

Inline assembly (function line : 71-73)
Inline Assembly (upgrade/UpgradeBeacon.sol#71-73)

Issue

The use of assembly is error-prone and uses excess gas.

Recommendation

Avoid EVM assembly and use standard solidity instead.

4. UBC-01 (UpgradeBeaconController.sol)

ID	Title	Category	Severity	Status
UBC-01	contracts/upgrade/UpgradeBeaconController.sol	Volatile Code	Low	Requires Attention

Zero address validation

Line 31-47 Function

UpgradeBeaconController.upgrade(address _beacon, address _implementation)
(_success) = _beacon.call(abi.encode(_implementation))
(UpgradeBeaconController.sol#38)

Issue

The function does not check for a zero-address.

Recommendation

Before using address _implementation check for non-zero address.

Inline Assembly

Line 31-47 Function

Inline assembly (function line : 41-44)

Issue

The use of assembly is error-prone and uses excess gas.

Recommendation

Avoid EVM assembly and use standard solidity instead.

5. UBP-01 (UpgradeBeaconProxy.sol)

ID	Title	Category	Severity	Status
UBP-01	contracts/upgrade/UpgradeBeaconProxy.sol	Volatile Code	Low	Requires Attention

Inline Assembly

Line 85-98 Function

UpgradeBeaconProxy._initialize(address,bytes)
Inline assembly (function line : 93-96)

```
assembly {
            returndatacopy(0, 0, returndatasize())
            revert(0, returndatasize())
        }
```

Line 115-142 Function

UpgradeBeaconProxy._delegate(address)
Inline assembly (function line : 116-141)

Issue

The use of assembly is error-prone and uses excess gas.

Recommendation

Avoid EVM assembly and use standard solidity instead.

6. HM-01 (Home.sol)

ID	Title	Category	Severity	Status
HM-01	contracts/Home.sol	Gas Optimization	Low	Requires Attention

Line 111-120 Function

initialize(IUpdaterManager _updaterManager) public initializer (Home.sol#111-120)

Issue

The function is declared with visibility mode as public instead of external. The function is never called by the contract.

Recommendation

For gas optimization use the external attribute for functions if they are not called from the contract but are used externally.

7. UM-02 (UpdaterManager.sol)

ID	Title	Category	Severity	Status
UM-02	contracts/UpdaterManager.sol	Volatile Code	Low	Requires Attention

Zero address validation

Line 55-57 Function

UpdaterManager.constructor(address _updaterAddress)
_updater = _updaterAddress (UpdaterManager.sol#56)

Issue

The function does not check for a zero-address.

Recommendation

Before using address _updaterAddress check whether it is a non-zero address or not

8. RPLA-01 (Replica.sol)

ID	Title	Category	Severity	Status
RPLA-01	contracts/Replica.sol	Gas Optimization, Volatile Code	Low	Requires Attention

Line 94-106 Function

Replica.initialize(uint32,address,bytes32,uint256) (Replica.sol#94-106)

Issue

The function is declared with visibility mode as public instead of external. The function is never called by the contract.

Recommendation

For gas optimization use the external attribute for functions if they are not called from the contract but are used externally.

Inline Assembly

Line 168-230 Function

Replica.process(bytes) (Replica.sol#168-230)
Inline assembly (function line : 206-225)V

Issue

The use of assembly is error-prone and uses excess gas.

Recommendation

Avoid EVM assembly and use standard solidity instead.

Appendix - Category Descriptions

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

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.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e., incorrect usage of private or delete.

Coding Style

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

Mathematical Operations

Mathematical operation findings related to mishandling of math formulas, such as overflows, incorrect operations etc.

Disclaimer

This report is based on the documents and materials provided to FTI as of the time of this report. The report is for informational purposes only and should not be used as the basis for any determinations or decisions by the recipient. This report is not an endorsement or disapproval of any particular project or team nor is it an indication of the economic value of any product or asset created by any team or project. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice. FTI shall not bear any responsibility for the consequences of the relevant decisions adopted by the recipient.

This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed and should not be considered a perfect representation of the risks threatening the analyzed system. Source code reviews are a “point-in-time” analysis and it is possible that the code and overall smart contract functionality could have changed since the review in this report was conducted.

In consideration of FTI allowing the recipient access to the report and, the recipient agrees that it does not acquire any rights as a result of such access that it would not otherwise have had and acknowledges that FTI does not assume any duties or obligations to the recipient in connection with such access.

The recipient agrees to release FTI and its personnel from any claim by the recipient that arises as a result of FTI permitting the recipient access to the report. Without our written consent, the report shall not be disclosed or provided to any other party or used for any other purposes.