SPEARBIT

Optimism Upgrade 16 Security Review

Auditors

Chris Smith, Lead Security Researcher

MiloTruck, Lead Security Researcher

Report prepared by: Lucas Goiriz

July 16, 2025

Contents

1 About Spearbit

Spearbit is a decentralized network of expert security engineers offering reviews and other security related services to Web3 projects with the goal of creating a stronger ecosystem. Our network has experience on every part of the blockchain technology stack, including but not limited to protocol design, smart contracts and the Solidity compiler. Spearbit brings in untapped security talent by enabling expert freelance auditors seeking flexibility to work on interesting projects together.

Learn more about us at spearbit.com

2 Introduction

Optimism is a fast, stable, and scalable L2 blockchain built by Ethereum developers, for Ethereum developers. Built as a minimal extension to existing Ethereum software, Optimism's EVM-equivalent architecture scales your Ethereum apps without surprises. If it works on Ethereum, it works on Optimism at a fraction of the cost.

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of optimism according to the specific commit. Any modifications to the code will require a new security review.

3 Risk classification

Severity level	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

3.1 Impact

- High - leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
- Medium - global losses <10% or losses to only a subset of users, but still unacceptable.
- Low - losses will be annoying but bearable--applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.

3.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align, or little-to-no incentive

3.3 Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

4 Executive Summary

Over the course of 4 days in total, Op Labs engaged with Spearbit to review the optimism protocol, specifically the `scripts/deploy/VerifyOPCM.s.sol` file at commit 731280...fd60 and that it verifies the `src/L1/OPContractsManager.sol` contract at commit 54c19f...4a3a6. In this period of time a total of **3** issues were found.

Summary

Project Name	Op Labs
Repository	optimism
Commits	731280...fd60 and 54c19f...4a3a6
Scope	`scripts/deploy/VerifyOPCM.s.sol` and `src/L1/OPContractsManager.sol`
Type of Project	Infrastructure, L2
Audit Timeline	Jul 9 to Jul 14

Issues Found

Severity	**Count**	**Fixed**	**Acknowledged**
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	1	0	1
Low Risk	1	0	1
Gas Optimizations	0	0	0
Informational	1	0	1
Total	**3**	**0**	**3**

5 Findings

5.1 Medium Risk

5.1.1 Blueprints and implementations are only checked for `OPContractsManagerDeployer`

Severity: Medium Risk

Context: VerifyOPCM.s.sol#L189-L199, OPContractsManager.sol#L1937-L1945

Description: There are four contracts alongside `OPContractsManager` itself:

- `OPContractsManagerGameTypeAdder`.
- `OPContractsManagerUpgrader`.
- `OPContractsManagerDeployer`.
- `OPContractsManagerInteropMigrator`.

Each contract has an immutable `contractsContainer` address that points to `OPContractsManagerContractsContainer`, which stores the `Blueprints` and `Implementations` struct.

The `VerifyOPCM.s.sol` script fetches the blueprint and implementation addresses in the `OPContractsManager` contract by calling its `blueprints()` and `implementations()` functions:

```
/// @notice Returns the blueprint contract addresses.
function blueprints() public view returns (Blueprints memory) {
    return opcmDeployer.blueprints();
}

/// @notice Returns the implementation contract addresses.
function implementations() public view returns (Implementations memory) {
    return opcmDeployer.implementations();
}
```

However, as seen from above, these two functions fetch the `Blueprints` and `Implementatiosn` struct from only `OPContractsManagerDeployer`. This means all subsequent checks in the script only check the blueprints and implementations for `OPContractsManager` and `OPContractsManagerDeployer`. The other three contracts could have a different `contractsContainer` address, resulting in different blueprint and implementation addresses, and the script would still pass.

Recommendation: The script should check the implementation and blueprint addresses for all four contracts instead. Alternatively, all four contracts could be checked to have the same `contractsContainer` address.

Optimism: Acknowledged, this will be fixed in a future upgrade. For Upgrade 16, the `contractsContainer` address will be manually checked to be the same for all four contracts.

Spearbit: Acknowledged.

5.2 Low Risk

5.2.1 Immutable variables are not checked

Severity: Low Risk

Context: VerifyOPCM.s.sol#L326-L333

Description: In the `VerifyOPCM.s.sol` script, the `_verifyOpcmContractRef()` function does not check the value of immutable variables to ensure they are correct.

As such, the OPCM's `superchainConfig`, `protocolVersions`, `superchainProxyAdmin` and `upgradeController` addresses could be wrong and the script would still pass.

Recommendation: Modify the script to ensure immutable variables are checked as well.

Optimism: Acknowledged. As part of our current process, immutable variables are manually checked to be correct.

Spearbit: Acknowledged.

5.3 Informational

5.3.1 `OPContractsManager` never upgrades the `ProtocolVersions` contract

Severity: Informational

Context: OPContractsManager.sol#L1710-L1713

Description: In `OPContractsManager`, the `Implementations` struct contains the implementation address for the `ProtocolVersions` contract:

```
/// @notice The latest implementation contracts for the OP Stack.
struct Implementations {
    address superchainConfigImpl;
    address protocolVersionsImpl;
```

However, the `ProtocolVersions` contract is never actually upgraded to the latest implementation address in the OPCM. This can be seen as `protocolVersionsImpl` is never used anywhere in the contract.

Recommendation: Similar to the `SuperchainConfig` contract, the OPCM should check the current implementation of `ProtocolVersions` and upgrade it if it isn't already upgraded.

Optimism: Acknowledged.

Spearbit: Acknowledged.