Optimism u17
Security Review

Cantina Managed review by:

MiloTruck, Lead Security Researcher
R0bert, Lead Security Researcher

October 27, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

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

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings are a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

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.

From Oct 15th to Oct 21st the Cantina team conducted a review of optimism on commit hash aeed7033.

In addition, PR 17998 was reviewed by the researchers.

The team identified a total of **3** issues:

Issues Found

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

3 Findings

3.1 Informational

3.1.1 `DEPLOY_V2_DISPUTE_GAMES` rollback leaves V2 args on legacy game

Severity: Informational

Context: OPContractsManager.sol#L1232-L1238

Description: When `DEPLOY_V2_DISPUTE_GAMES` is enabled, `OPContractsManager.registerPermissionedGameV2` writes the 124-byte constructor payload into the factory via `setImplementation(gameType, impl, gameArgs)`. Later, switching the feature flag off falls back to the three-argument helper, which only invokes `setImplementation(gameType, impl)` and never clears `gameArgs[gameType]`. Because `DisputeGameFactory.create` always concatenates the stored constructor blob, the stale V2 payload is still appended when cloning the legacy `FaultDisputeGame`. That legacy initializer rejects any calldata length other than 122 bytes and reverts with `BadExtraData()`, so every rollback deployment bricked fault-proof creation once V2 had been enabled. Impact: toggling off the feature via the documented governance flow leaves the system unable to produce fault proofs until an operator manually deletes the stored args.

Recommendation: When reinstalling the legacy implementation, explicitly clear the cached constructor blob (for example `delete gameArgs[_gameType];` before calling the two-argument setter).

Optimism: Acknowledged. The `DevFeatures` are not intended to be deployed to production. We have the check VerifyOPCM.s.sol#L965-L978 which enforces that the feature flags bitmap is empty for mainnet release. The idea is, we will remove the feature flags and guards altogether (deleting the legacy code) ahead of deployment - we will not support toggling these flags in production.

Cantina Managed: Acknowledged.

3.1.2 Permissioned bond starts at 0 until `addGameType`

Severity: Informational

Context: OPContractsManager.sol#L1228-L1231

Description: `_registerPermissionedGameV2` registers the v2 permissioned game without touching `DisputeGameFactory.setInitBond`, so a fresh factory keeps the default zero bond. The team clarified that new chains intentionally launch with only the permissioned game and no bond; once permissionless proofs are enabled via `opcm.addGameType`, that upgrade also sets the init bond for both permissioned and permissionless games. This sequencing is expected and should be documented so the zero bond at genesis isn't mistaken for a misconfiguration.

Recommendation: No code change required. Document the deployment sequence to clarify that the permissioned game bond is intentionally zero at genesis and will be aligned with the permissionless bond when `opcm.addGameType` runs.

Optimism: Acknowledged.

Cantina Managed: Acknowledged.

3.1.3 Fork-Based Upgrade Tests Summary

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: We invested effort in building manual, fork-based tests on an Optimism fork so we could observe the real contract-level behavior and gas costs of the u17 upgrade and deployment flows. Running them against live chain state ensures what we see mirrors actual production transactions.

The fork tests can be located in the gist aa84c459.

- The `test_upgrade_u16_to_u17_fork` scenario starts from a freshly deployed U16 rollup configuration and delegates to `OPContractsManagerUpgrader.upgrade`. The delegatecall itself costs about 4.99 M gas and proves that the non-interop upgrade path is sound: every critical proxy: `SystemConfig`, `OptimismPortal2`, L1 bridges, `L1CrossDomainMessenger`, `DisputeGameFactory`, `AnchorStateRegistry` and `DelayedWETH` lands on the U17 implementation while ownership remains with the configured

proxy-admin owner. Because no dev features are enabled, the legacy `ETHLockbox` pointer is left untouched, the portal retains its ETH and the permissioned dispute game still swaps via the two-argument setter (so any stale V2 constructor blob could still block a rollback).

- The `test_deploy_u17_chain_end_to_end` path invokes `u17Deployer.deploy` once, at a cost of roughly 12.94 M gas, to stand up a fully interop-enabled U17 stack. With `DevFeatures.OPTIMISM_PORTAL_INTEROP` baked into the contracts container, the call instantiates `SystemConfig`, `OptimismPortalInterop`, `ETHLockbox`, all three L1 bridges, the L1 messenger, `OptimismMintableERC20Factory`, `DisputeGameFactory`, `AnchorStateRegistry`, `DelayedWETH` and the legacy permissioned dispute game. The test then inspects every initializer outcome (owners, batcher hash, unsafe signer, PDG constructor params, start anchor, lockbox authorization, ASR/DGF wiring) to demonstrate that a single transaction yields an immediately usable, interop-ready rollup. This effectively serves as the reference deploy script for new chains.
- Finally, the `test_upgrade_u16_to_u17_interop_migrates_lockbox` covers the interop-enabled upgrade. After seeding the legacy portal with 5 ETH, it runs `OPContractsManagerUpgrader.upgrade` (gas ~5.00 M) with the interop dev feature enabled, then, still acting as the proxy-admin owner, calls `SystemConfig.setFeature("ETH_LOCKBOX", true)` followed by `OptimismPortalInterop.migrateLiquidity`. This sequence is essential because the upgrader itself neither flips the feature flag nor sweeps funds. The test confirms the portal still points to the existing lockbox proxy (the upgrade reuses it), the manual migration drains the portal balance to zero while crediting the lockbox with exactly 5 ETH, and the portal remains authorized. In effect it documents the post-upgrade runbook: toggle the feature, run the migration and verify that the lockbox holds the ETH.

Notes & potential follow-ups: It would be helpful adding a path that clears stale dispute-game constructor blobs to remove the lingering rollback hazard. Also, another thing to remark is that the u17 deployment test consumes ~12.9M gas, which should be acceptable operationally but is noteworthy for planning and cost predictability.

Optimism: Acknowledged.

Cantina Managed: Acknowledged.

4 Appendix

4.1 File hashes

The following files were reviewed in the present engagement:

File	Keccak256 Hash
L1/L1CrossDomainMessenger.sol	0xb04e48847710a27440c4354068fdc534 109f64390743737e19a451c553a7e7a5
L1/L1ERC721Bridge.sol	0xe942b2d5dcdc09d4b3503bba061e1093 e486c02e2e0896bb91a2f5f56ad488c7
L1/L1StandardBridge.sol	0xf1fe7c2981154c7d5f4fe0bd8590690a 28a9d260129fd59f21e174d8bd899771
L1/OPContractsManager.sol	0x718a01d43d47edb1ea9991a8e6effb0b 667c0c81ca57240869659b946b8f3f91
L1/OptimismPortal2.sol	0x3b924b5bae92522dcb96a606977267db 229d5b9cadf98e13b9b60d5d71a89c33
L1/SuperchainConfig.sol	0xd5b15c1598fdab0314a8857fa52c78a7 a8bb201d6c4a245a3cfc8859c236f174
L1/SystemConfig.sol	0x0f824e6de6260912785833058aeb5ff4 4bdea9e1a3ae824d786a2f22bbfae35d
L2/GasPriceOracle.sol	0xc4e521ae63672a90ae4a7e760238e714 7c95d5c620e21dcbe0aad3c5bfd57a6e
L2/L1Block.sol	0x47628b552a59fe73111a351c57abc333 a37acd66ff42eb1aa658aa50c7558166
cannon/MIPS64.sol	0xa5b60a99218a88dbbaf3b4b5bbb44744 66b089bcd17e3ee5bd93b3d115f95199
cannon/libraries/MIPS64Instructions.sol	0x2a6cde1bddaa3c72a2da64525bc6b52b fa3e65c555f250fa55c78580c649d81a
cannon/libraries/MIPS64State.sol	0x1d823d997acfea33cb73a85041943715 d50fc2d8bf767dcf93f4f70116feedb3
dispute/DisputeGameFactory.sol	0x637c57a469a65fdb9a44e20544769643 c34fc44cc59c3b6864a72b35a56f053e
dispute/FaultDisputeGame.sol	0x9d1b8465180a9aa0c9c2703187889b30 40eea7032a0bee7d8d1e5d85defd2d97
dispute/PermissionedDisputeGame.sol	0x1613293bc0d9f39c12a50530404bae06 a4e4236de8d9c04b679f859a72f47a66
dispute/lib/Types.sol	0x468fccc8d6ce3ac5b87bdd5002f84b31 5ac861be7655ceef87a018f596c410a1
libraries/DevFeatures.sol	0x3b587a1b2a4b657d26cd28be2e96ab68 e1582c37b45f3b717341d033e21a401f
libraries/Encoding.sol	0x7b1547dc1e1a745856fee295e43fa5d3 274466fade0fbdc5f1552ff940fb0406
universal/OptimismMintableERC20.sol	0x43fb0a13902050317935b4e9f0440358 e6d74eaab00e5e7418df9d464308c0e5
universal/OptimismMintableERC20Factory.sol	0xc9674fb62bee22344ceffdf5f246971d 75e089807a97db13d9360e273fefce5e

4.2 Python Script to Generate the File Hashes

Copy the contracts in scope into a directory called `relevant_contracts`.

```
import os
import sys
import sha3

def keccak256(data):
    return sha3.keccak_256(data).hexdigest()
```

```python
def hash_file(filepath):
    """Calculate hash of a file and return the result."""
    try:
        with open(filepath, 'rb') as f:
            data = f.read()
        hash_value = keccak256(data)
        filename = os.path.basename(filepath)
        print(f"{hash_value}  {filename}")
        return hash_value
    except Exception as e:
        print(f"Error reading {filepath}: {e}", file=sys.stderr)
        return None

def main():
    contracts_dir = './relevant_contracts'

    if not os.path.exists(contracts_dir):
        print(f"Directory {contracts_dir} does not exist", file=sys.stderr)
        sys.exit(1)

    # Get all .sol files and sort them
    files = [f for f in os.listdir(contracts_dir) if f.endswith('.sol')]
    files.sort()

    if not files:
        print("No .sol files found in relevant_contracts directory", file=sys.stderr)
        sys.exit(1)

    print("Calculating hashes for Solidity files:")
    print("=====================================")

    for file in files:
        filepath = os.path.join(contracts_dir, file)
        hash_file(filepath)

if __name__ == '__main__':
    main()
```