Polygon zkEVM Security Review

Calldata bugfix review

Auditors

Andrei Maiboroda, Lead Security Researcher

Report prepared by: Pablo Misirov

August 21, 2023

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

Polygon zkEVM is a new zk-rollup that provides Ethereum Virtual Machine (EVM) equivalence (opcode-level compatibility) for a transparent user experience and existing Ethereum ecosystem and tooling compatibility.

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of PR 23 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 2 days in total, Polygon engaged with Spearbit to review the zkevm-rom-internal protocol. In this period of time a total of **6** issues were found.

Summary

Project Name	Polygon
Repository	zkevm-rom-internal
Commit	PR 23
Type of Project	Assembly, zkEVM
Audit Timeline	August 8 - August 10

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	1	1	0
Gas Optimizations	0	0	0
Informational	5	4	0
Total	**6**	**5**	**0**

5 Findings

5.1 Low Severity

5.1.1 CREATE2 sets `txCalldataLen` after saving calldata pointer

Severity: *Low Risk*

Context create-terminate-context.zkasm#L746-L748

Description: `CALL*` opcodes and `CREATE` have a common pattern: first set `txCalldataLen` variable to `argsLength-Call` argument, then call `saveCalldataPointer`. `CREATE2` implementation has the order of these two actions reversed. It doesn't affect the outcome of the helper, but for consistency it would be better to have the same order as other opcodes.

Recommendation: First set `txCalldataLen`, then call `saveCalldataPointer` in `CREATE2`.

Polygon zkEVM: Fixed in PR 23.

Spearbit: Fixed.

5.2 Informational

5.2.1 Constant has confusing name

Severity: *Informational*

Context: stack-operations.zkasm#L13

Description: `%CALLDATA_CTX` constant's name can be confusing because there is also a variable `calldataCTX`.

Recommendation: Rename the constant to `%CALLDATA_RESERVED_CTX`.

Polygon zkEVM: Fixed in PR 23.

Spearbit: Fixed.

5.2.2 Incorrect output variable in helper call comment

Severity: *Informational*

Context: calldata-returndata-code.zkasm#L27 calldata-returndata-code.zkasm#L134 calldata-returndata-code.zkasm#L161

Description: Comment states that output variable of `readFromCalldataOffset` helper is `readXFromCalldataOffset`, but it is actually `readXFromCalldataResult`:

```
A               :MSTORE(readXFromCalldataOffset), CALL(readFromCalldataOffset); in:
↪  [readXFromCalldataOffset: offset value, readXFromCalldataLength: length value], out:
↪  [readXFromCalldataOffset: result value]
```

Recommendation: Change output in comment to `readXFromCalldataResult`.

5.2.3 No need to save D register in `readFromCalldataOffset` helper

Severity: *Informational*

Context utils.zkasm#L1540

Description: `D` register is not modified in `readFromCalldataOffset`, so it's not necessary to save its previous value and restore in the end.

Recommendation: Remove `tmpVarDReadXFromOffset` variable and saving/restoring of `D`

Polygon zkEVM: Fixed in PR 23.

Spearbit: Fixed.

5.2.4 Outdated comment

Severity: *Informational*

Context: create-terminate-context.zkasm#L388 create-terminate-context.zkasm#L625 create-terminate-context.zkasm#L744 create-terminate-context.zkasm#L830

Description: Comment in CALL and CREATE opcodes describes context copying as it was done before the fix:

```
    ; copy calldata from origin CTX to current CTX
```

Recommendation: Rewrite the comment to reflect new approach.

Polygon zkEVM: Fixed in PR 23.

Spearbit: Fixed.

5.2.5 Unneeded label

Severity: *Informational*

Context utils.zkasm#L1142

Description: Label `addBatchHashByteByByteEnd` is added by the PR, but it is not used and is not relevant to the fix.

Recommendation: Label can be removed.

Polygon zkEVM: Fixed in PR 23.

Spearbit: Fixed.