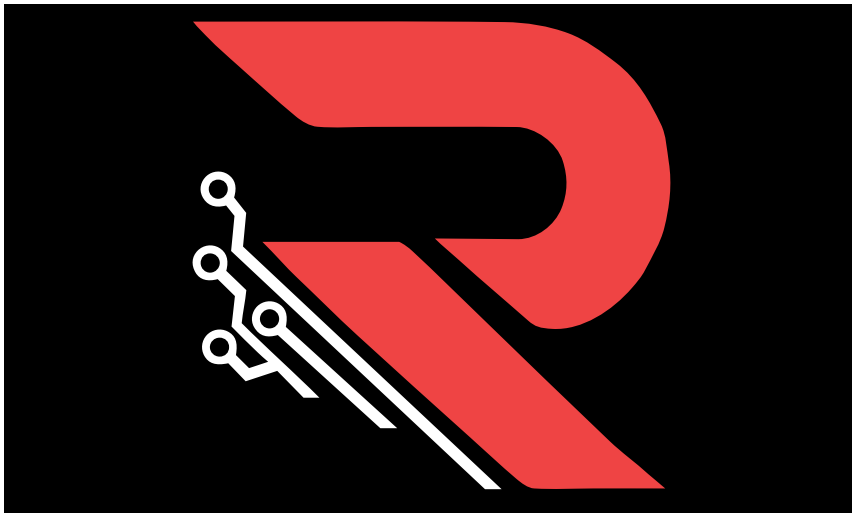
MaxAPY Security Review



Version 2.0

10.04.2025

Conducted by:

MaslarovK, Lead Security Researcher

radev-eth, Lead Security Researcher

Table of Contents

1 About MaslarovK

MaslarovK a Security Reseacher and Co-Founder of Rezolv Solutions.

2 About radev.eth

radev_eth a Security Reseacher and Co-Founder of Rezolv Solutions.

3 Disclaimer

Audits are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

4 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

4.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

4.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

5 Executive summary

Overview

Project Name	MaxAPY
Repository	https://github.com/VerisLabs/metavault
Commit hash	1257f512568eaf96ae14419fedbf2dcdcb767334
Resolution	20a546acd32d006ee2561dbb17c8da77bf40aaea
Documentation	https://devs.maxapy.io/
Methods	Manual review

Scope

src/MetaVault.sol
src/crosschain/modules/DivestSuperform.sol
src/crosschain/modules/LiquidateSuperform.sol
src/crosschain/ERC20Receiver.sol
src/modules/ERC7540Engine/ERC7540Engine.sol
src/modules/ERC7540Engine/ERC7540EngineSignatures.sol
src/modules/ERC7540Engine/common/ERC7540EngineBase.sol
src/modules/ERC7540Engine/common/ERC7540ProcessRedeemBase.sol

Issues Found

Critical risk	0
High risk	1
Medium risk	0
Low risk	1
Informational	1

6 Findings

6.1 High risk

6.1.1 No check if all the requests retrieved by a certain key are processed

Severity: *High risk*

Description: In `DivestSuperform.sol::notifyRefund` and `DivestSuperform.sol::notifyBatchRefund`, there is no check if the all the requests retrieved by a certain key are processed:

```
function notifyRefund(uint256 superformId, uint256 value) external {
        // Prevent bugs from superform
        if (value == 0) return;
        bytes32 key = ERC20Receiver(msg.sender).key();
        if (requests[key].receiverAddress != msg.sender) revert();
        RequestData memory req = requests[key];
        uint256 currentExpectedBalance = ERC20Receiver(msg.sender).
            minExpectedBalance();

        uint256 vaultIndex;
        for (uint256 i = 0; i < req.superformIds.length; ++i) {
            if (req.superformIds[i] == superformId) {
                vaultIndex = i;
                break;
            }
        }
        uint256 vaultRequestedAssets = req.requestedAssetsPerVault[vaultIndex];

        _handleRefund(key, superformId, value, vaultRequestedAssets);

        //@audit key is being removed even if there are unprocessed checks.
        _requestsQueue.remove(key); // We can only remove the request if it's a
            single vault otherwise we need to
            // confirm both succeeded
        ERC20Receiver(msg.sender).setMinExpectedBalance(_sub0(currentExpectedBalance
            , vaultRequestedAssets));
    }

function notifyBatchRefund(uint256[] calldata superformIds, uint256[] calldata
    values) external {
        bytes32 key = ERC20Receiver(msg.sender).key();
        if (requests[key].receiverAddress != msg.sender) revert();
        RequestData memory req = requests[key];
        uint256 currentExpectedBalance = ERC20Receiver(msg.sender).
            minExpectedBalance();
        uint256 totalVaultRequestedAssets;

        for (uint256 j = 0; j < superformIds.length; ++j) {
            uint256 vaultIndex;
            for (uint256 i = 0; i < req.superformIds.length; ++i) {
                if (req.superformIds[i] == superformIds[j]) {
                    vaultIndex = i;
                    break;
                }
            }
            uint256 vaultRequestedAssets = req.requestedAssetsPerVault[vaultIndex];
```

```
            totalVaultRequestedAssets += vaultRequestedAssets;

            _handleRefund(key, superformIds[j], values[j], vaultRequestedAssets);
        }
        //@audit key is being removed even if there are unprocessed checks.
        _requestsQueue.remove(key);
        ERC20Receiver(msg.sender).setMinExpectedBalance(_sub0(currentExpectedBalance
            , totalVaultRequestedAssets));
    }
```

This may lead to lost data and funds.

Recommendation: Implement a check to ensure all the superformIds in the queue have been processed and only if so - remove the key.

Resolution: Fixed.

6.2 Low risk

6.2.1 ChainId may be changed after the upcoming Ethereum hard fork

Severity: *Low risk*

Description: In the MetaVault.sol, the `THIS_CHAIN_ID` is set once in the contructor, but later it is compared to `block.chainid`. This is a standard practice, but in case of a hard fork, the chainId may be changed, which will result in DoS for all the functions depending on such check.

Recommendation: Introduce a restricted function to change the `THIS_CHAIN_ID` if needed.

Resolution: Aknowledged.

6.3 Informational

6.3.1 The `value` in `OnERC1155Received` and `OnERC1155BatchReceived` is used, no need to silence the compiler.

Severity: *Informational*

Resolution: Aknowledged.