ZUNAMI SMART CONTRACTS SECURITY AUDIT REPORT

OXORIO

APRIL 30, 2024

CONTENTS

1 INTRO

1.1 DISCLAIMER

The audit makes no assertions or warranties about the utility of the code, its security, the suitability of the business model, investment advice, endorsement of the platform or its products, the regulatory regime for the business model, or any other statements about the fitness of the contracts for their intended purposes, or their bug-free status. The audit documentation is for discussion purposes only.

1.2 ABOUT OXORIO

Oxorio is a prominent audit and consulting firm in the blockchain industry, offering top-tier security audits and consulting to organizations worldwide. The company's expertise stems from its active involvement in designing and deploying multiple blockchain projects, wherein it developed and analyzed smart contracts.

With a team of more than six dedicated blockchain specialists, Oxorio maintains a strong commitment to excellence and client satisfaction. Its contributions to several blockchain projects reflect the company's innovation and influence in the industry. Oxorio's comprehensive approach and deep blockchain understanding make it a trusted partner for organizations in the sector.

Contact details:

- oxor.io
- ping@oxor.io
- Github
- Linkedin
- Twitter

1.3 SECURITY ASSESSMENT METHODOLOGY

Several auditors work on this audit, each independently checking the provided source code according to the security assessment methodology described below:

1. Project architecture review

The source code is manually reviewed to find errors and bugs.

2. Code check against known vulnerabilities list

The code is verified against a constantly updated list of known vulnerabilities maintained by the company.

3. Security model architecture and structure check

The project documentation is reviewed and compared with the code, including examining the comments and other technical papers.

4. Cross-check of results by different auditors

The project is typically reviewed by more than two auditors. This is followed by a mutual cross-check process of the audit results.

5. Report consolidation

The audited report is consolidated from multiple auditors.

6. Re-audit of new editions

After the client has reviewed and fixed the issues, these are double-checked. The results are included in a new version of the audit.

7. Final audit report publication

The final audit version is provided to the client and also published on the company's official website.

1.4 FINDINGS CLASSIFICATION

1.4.1 Severity Level Reference

The following severity levels were assigned to the issues described in the report:

- **CRITICAL**: A bug that could lead to asset theft, inaccessible locked funds, or any other fund loss due to unauthorized party transfers.
- **MAJOR**: A bug that could cause a contract failure, with recovery possible only through manual modification of the contract state or replacement.
- **WARNING**: A bug that could break the intended contract logic or expose it to DDoS attacks.
- **INFO**: A minor issue or recommendation reported to or acknowledged by the client's team.

1.4.2 Status Level Reference

Based on the client team's feedback regarding the list of findings discovered by the contractor, the following statuses were assigned to the findings:

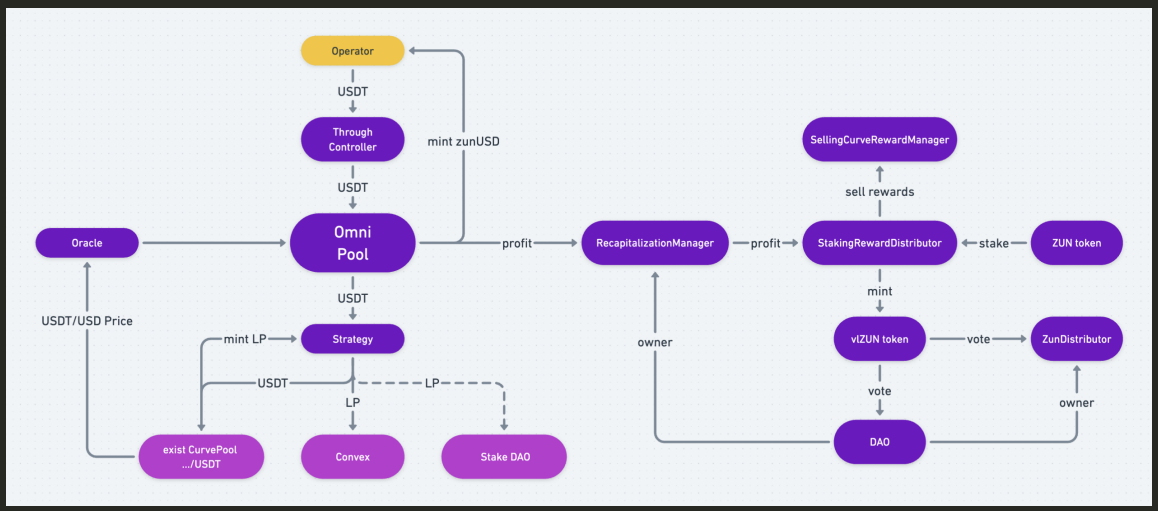
- **NEW**: Awaiting feedback from the project team.
- **FIXED**: The recommended fixes have been applied to the project code, and the identified issue no longer affects the project's security.
- **ACKNOWLEDGED**: The project team is aware of this finding. Fixes for this finding are planned. This finding does not affect the overall security of the project.
- **NO ISSUE**: The finding does not affect the overall security of the project and does not violate its operational logic.

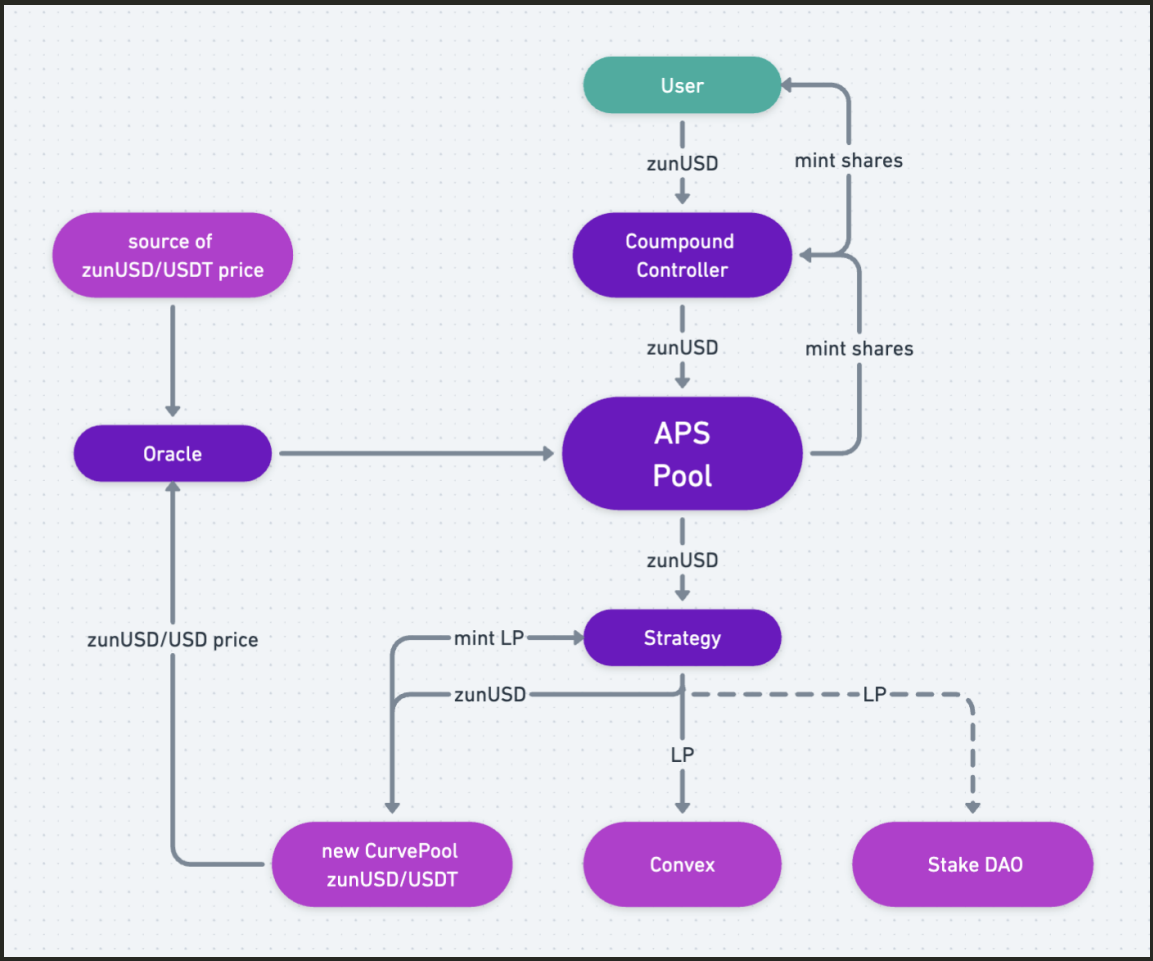
1.5 PROJECT OVERVIEW

Zunami is a protocol that aggregates stablecoin collateral to generate optimized yields.

Users deposit collateral to establish an omnipool. zunStables, representing shares of the omnipool, are issued to users. The omnipool provides liquidity across diverse yield-generating strategies. Profits are distributed among ZUN token stakers, who also hold governance rights. Stakers can vote to approve recapitalization.

System diagrams:





1.5.1 Documentation

For this audit, the following sources of truth about how the smart contracts should work were used:

- main GitHub repository of the project.

The sources were considered to be the specification. In the case of discrepancies with the actual code behavior, consultations were held directly with the client team.

1.6 AUDIT SCOPE

The audit scope covers the following smart contracts:

- ZunamiStratBase.sol
- ERC4626StratBase.sol
- StakeDaoERC4626StratBase.sol
- CrvUsdStakeDaoERC4626StratBase.sol
- LlamalendCrvUsdStakeDaoERC4626Strat.sol
- TokenConverter.sol
- SellingCurveRewardManager2.sol
- SdtOracle.sol
- ZunEthOracle.sol

The audited commit identifier is 16dcfab8b014d14c5f1b15792da2f859a85d08ec.
The identifier of the commit with fixes is 3851418c125703fdc3513a76672725de042a1ebb.
Additional fixes added in the commit 4e931f10026ac2d022c11f43d540c3394a0341eb.

2 FINDINGS REPORT

OXORIO

2.1 CRITICAL

C-01	Claiming rewards is blocked after liquidity removal in `ERC4626StratBase`
Severity	CRITICAL
Status	• FIXED

Location

File	Location	Line
ERC4626StratBase.sol	contract `ERC4626StratBase` > function `claimCollectedRewards`	68

Description

In the function `claimCollectedRewards` of the contract `ERC4626StratBase`, the `redeemableAssets` variable is compared to `depositedAssets` value, resulting in rewards claim in case the value of `redeemableAssets` is greater. The `depositedAssets` is increased upon a deposit in the function `depositLiquidity`, but it never diminished during withdrawal. Thus, a call to `removeLiquidity` will result in decrease in liquidity in the vault, resulting in a decrease of the `redeemableAssets` variable upon the call to `vault.previewRedeem()` in the `claimCollectedRewards` function, but the `depositedAssets` will not change, resulting in the `redeemableAssets > depositedAssets` inequality returning `false` and preventing the rewards from being claimed.

```
function claimCollectedRewards() internal virtual override {
    uint256 redeemableAssets = vault.previewRedeem(depositedLiquidity);
    if (redeemableAssets > depositedAssets) {
        uint256 withdrawnAssets = redeemableAssets - depositedAssets;
        uint256 withdrawnShares = vault.convertToShares(withdrawnAssets);
        uint256[5] memory minTokenAmounts;
        removeLiquidity(withdrawnShares, minTokenAmounts, false);
        depositedLiquidity -= withdrawnShares;
    }
}
```

The following test case demonstrates the issue:

```
it.only('should claim rewards', async () => {
    const { admin, alice, zunamiPool, zunamiPoolController, strategies} = await loadFixture(
        deployFixture
    );

    const llamaLandCrvUSD_vault_addr = '0xEdA215b7666936DEd834f76f3fBC6F323295110A';

    const strategy = strategies[0];
    await zunamiPool.addStrategy(strategy.address);

    await zunamiPoolController.setDefaultDepositSid(0);
    await zunamiPoolController.setDefaultWithdrawSid(0);

    const stableBefore = await zunamiPool.balanceOf(alice.getAddress());
    console.log('stable before deposit', stableBefore);

    await expect(
        zunamiPoolController
            .connect(alice)
            .deposit(getMinAmountZunUSD('1000'), alice.getAddress())
    ).to.emit(zunamiPool, 'Deposited');
    let stableAmount = BigNumber.from(await zunamiPool.balanceOf(alice.getAddress()));

    console.log('stable after deposit', stableAmount);
    console.log('deposited liquidity', await strategy.depositedLiquidity());

    console.log('vault CRVUSD balance before donation', await
crvUSD.balanceOf(llamaLandCrvUSD_vault_addr));

    await mintTokenTo(
        llamaLandCrvUSD_vault_addr, //  Llamalend vault controller
        admin,
        '0xf939E0A03FB07F59A73314E73794Be0E57ac1b4E', // CRVUSD
        '0xa920de414ea4ab66b97da1bfe9e6eca7d4219635', // CRVUSD Vault
        parseUnits('100000', 'ether')
    );

    console.log('vault CRVUSD balance after donation', await
crvUSD.balanceOf(llamaLandCrvUSD_vault_addr));

    await zunamiPool.connect(alice).approve(zunamiPoolController.address, stableAmount);
    console.log('deposited assets before withdraw', await strategy.depositedAssets());

    await expect(
```

```
        zunamiPoolController
            .connect(alice)
            .withdraw(ethers.utils.parseUnits('100', 'ether'), [0, 0, 0, 0, 0],
alice.getAddress())
    ).to.emit(zunamiPool, 'Withdrawn');

    stableAmount = BigNumber.from(await zunamiPool.balanceOf(alice.getAddress()));
    console.log('deposited assets after withdraw', await strategy.depositedAssets());
    console.log('stable after withdraw', stableAmount);

    let depositedLiquidityBeforeClaim = await strategy.depositedLiquidity();
    console.log('deposited liquidity before claim', depositedLiquidityBeforeClaim);

    await zunamiPoolController.claimRewards();

    let depositedLiquidityAfterClaim = await strategy.depositedLiquidity();
    console.log('deposited liquidity after claim', depositedLiquidityAfterClaim);

    await expect(depositedLiquidityBeforeClaim).to.be.gt(depositedLiquidityAfterClaim);
});
```

Output:

```
stable before deposit BigNumber { value: "0" }
stable after deposit BigNumber { value: "3003094966727196661254" }
deposited liquidity BigNumber { value: "2962999331045711925473538" }
vault CRVUSD balance before donation BigNumber { value: "45153143348167847377854O7" }
vault CRVUSD balance after  donation BigNumber { value: "46153143348167847377854O7" }
deposited assets before withdraw BigNumber { value: "3014670790067023793536" }
deposited assets after withdraw BigNumber { value: "3014670790067023793536" }
stable after withdraw BigNumber { value: "2903094966727196661254" }
deposited liquidity before claim BigNumber { value: "2865077463524275774038057" }
deposited liquidity after claim BigNumber { value: "2865077463524275774038057" }
```

Recommendation

We recommend updating the `depositedAssets` variable upon removing liquidity.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

2.2 MAJOR

M-01	Tokens' rate is ignored in `CrvUsdStakeDaoERC4626StratBase`
Severity	MAJOR
Status	• FIXED

Location

File	Location	Line
CrvUsdStakeDaoERC4626StratBase.sol	contract `CrvUsdStakeDaoERC4626StratBase` > function `convertAndApproveTokens`	75
CrvUsdStakeDaoERC4626StratBase.sol	contract `CrvUsdStakeDaoERC4626StratBase` > function `removeLiquidity`	96

Description

In the mentioned locations and in similar logic of other strategies, potential slippage in the converter is limited:

```
function convertAndApproveTokens(
  ...
  uint256[POOL_ASSETS] memory amounts
) internal override returns (uint256 amount) {
  ...
  converter.handle(
    ...
    applySlippage(amounts[i]) * tokenDecimalsMultipliers[i]
  );
```

However, the amount of the input tokens (`DAI/USDT/USDC` in the `convertAndApproveTokens` function) is used to limit slippage for the output token (`CRVUSD`) without taking into account the difference in their rates.

```
function applySlippage(uint256 amount) internal view returns (uint256) {
  return (amount * (SLIPPAGE_DENOMINATOR - slippage)) / SLIPPAGE_DENOMINATOR;
}
```

If the rate of input and output token deviates from `1`, the slippage mechanism works incorrectly. The following scenario is possible:

- The `slippage` parameter is set to `0.5%`
- The `USDT/CRVUSD` rate is `2`
- `convertAndApproveTokens` is called during deposit `1000` USDT
- `minAmountOut_` parameter for the `handle` function of `tokenConverter` is set to `995` `CRVUSD`. It means that the real slippage limit is `50.5%`.

Recommendation

We recommend using the input/output token rate in the slippage limit calculation.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

2.3 WARNING

W-01	Missing parameter validation
Severity	WARNING
Status	• FIXED

Location

File	Location	Line
ZunEthOracle.sol	contract `ZunEthOracle` > `constructor`	23
SdtOracle.sol	contract `SdtOracle` > `constructor`	22
SellingCurveRewardManager2.sol	contract `SellingCurveRewardManager2` > function `setDefaultSlippage`	35

Description

Parameter validation is not performed in the specified constructors and setter function. The lack of validation can lead to unpredictable behavior or the occurrence of panic errors.

Recommendation

We recommend implementing validation for parameters to ensure stable and predictable behavior.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

Oxorio's response

The parameter `defaultSlippage_` of the function setDefaultSlippage is not validated against the upper bound - the `SLIPPAGE_DENOMINATOR` value. If the value of the `defaultSlippage` is greater than `SLIPPAGE_DENOMINATOR`, the function `calcMinAmount` will revert, resulting in blocked execution of the `valuate` and `handle` functions.
We recommend adding a validation `defaultSlippage_ <= SLIPPAGE_DENOMINATOR` to the `setDefaultSlippage` function.

Client's response

Fixed in the commit `4e931f10026ac2d022c11f43d540c3394a0341eb`.

W-02	Missing `route` validation in `TokenConverter`
Severity	WARNING
Status	• FIXED

Location

File	Location	Line
TokenConverter.sol	contract `TokenConverter` > function `setRoute`	28-29

Description

In the `setRoute` function of the `TokenConverter` contract, there is no validation of `route.length` and `swapParams.length`. The values of these variables are used in the `_fillRoutes` and `_fillSwapParams` functions:

```
function _fillRoutes(
  ...
) internal view returns (address[11] memory routes_) {
  for (uint8 i; i < routes[tokenIn_][tokenOut_].route.length; ++i) {
    routes_[i] = routes[tokenIn_][tokenOut_].route[i];
  }
}

function _fillSwapParams(
  ...
) internal view returns (uint256[5][5] memory swapParams_) {
  for (uint8 i; i < routes[tokenIn_][tokenOut_].swapParams.length; ++i) {
    swapParams_[i] = routes[tokenIn_][tokenOut_].swapParams[i];
  }
}
```

These functions assume that `route.length` is less than or equal to `11` and `swapParams.length` is less than or equal to `5`. Otherwise, an out-of-range error occurs.

Recommendation

We recommend adding the following parameter validation checks in the `setRoute` function:

```
if (route.length > 11) revert WrongLength();
if (swapParams.length > 5) revert WrongLength();
```

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

W-03	Valuation result doesn't account for slippage in `TokenConverter`
Severity	WARNING
Status	• NO ISSUE

Location

File	Location	Line
TokenConverter.sol	contract `TokenConverter` > function `valuate`	82

Description

In the function `valuate` of the contract `TokenConverter`, the valuation of the input tokens is performed and the value of the `ICurveRouterV1(curveRouter).get_dy()` is returned. Thus, the caller of the `valuate` function is required to validate this value against a calculation that uses oracle prices of the valuated tokens. This workflow is different from the one used in other converters.
For example, the `FraxEthNativeConverter` checks the slippage inside the the `valuate` function and reverts in case the slippage is above treshold:

```
function valuate(bool buyToken, uint256 amount) public view returns (uint256 valuation) {
    if (amount == 0) return 0;
    int128 i = buyToken ? ETH_frxETH_POOL_WETH_ID : ETH_frxETH_POOL_frxETH_ID;
    int128 j = buyToken ? ETH_frxETH_POOL_frxETH_ID : ETH_frxETH_POOL_WETH_ID;
    valuation = fraxEthPool.get_dy(i, j, amount);

    if (valuation < applySlippage(amount, 0)) revert BrokenSlippage();
}
```

Similar approach is used in the `StableConverter`. The difference in the approach to checking slippage creates a missmatch in the responsibility of checking the slippage between the consumers of the converters.

Recommendation

We recommend using a unified approach to checking the slippage during the valuation call to prevent potential mistakes.

Update

Client's response

Other converters have become obsolete. With the new version of the TokenConverter, there is no need to use slippage.

2.4 INFO

I-01	`Require` is used for error handling
Severity	INFO
Status	• FIXED

Location

File	Location	Line
SdtOracle.sol	contract `SdtOracle` > function `getUSDPrice`	27
ZunEthOracle.sol	contract `ZunEthOracle` > function `getUSDPrice`	28
SellingCurveRewardManager2.sol	contract `SellingCurveRewardManager2` > `constructor`	29
SellingCurveRewardManager2.sol	contract `SellingCurveRewardManager2` > `constructor`	32
ZunamiStratBase.sol	contract `ZunamiStratBase` > function `setSlippage`	60
ZunamiStratBase.sol	contract `ZunamiStratBase` > function `getLiquidityAmountByRatio`	120

Description

In the mentioned locations and the other contracts outside of the current scope, `require` is used for the error handling.

Recommendation

We recommend using the `error` type instead of `require` to maintain the same code style throughout the entire project.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

I-02	Magic numbers
Severity	INFO
Status	• FIXED

Location

File	Location	Line
SdtOracle.sol	contract `SdtOracle` > function `getUSDPrice`	28
ERC4626StratBase.sol	contract `ERC4626StratBase` > function `getLiquidityTokenPrice`	31
ERC4626StratBase.sol	contract `ERC4626StratBase` > function `claimCollectedRewards`	71
TokenConverter.sol	contract `TokenConverter` > function `setRoute`	29
TokenConverter.sol	contract `TokenConverter` > function `setRoutes`	38
TokenConverter.sol	contract `TokenConverter` > function `valuate`	79-80

Description

In the mentioned locations literal values with unexplained meaning are used to perform calculations.

Recommendation

We recommend defining a constant for every magic number, giving it a clear and self-explanatory name.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

Oxorio's response

The constant `SDT_IDNEX` introduced in the SdtOracle.sol contract contains a typo, the name should be `SDT_INDEX`.

I-03	Inconsistent interface location in `SdtOracle`, `ZunEthOracle`
Severity	INFO
Status	• FIXED

Location

File	Location	Line
SdtOracle.sol	interface `ICurvePriceOracleNG`	7
ZunEthOracle.sol	interface `ICurvePriceOracleNG`	7

Description

In the mentioned locations, interfaces are declared in contract implementation files. However, there is a separate interfaces folder for interfaces in the project.

Recommendation

We recommend moving the interfaces to the interfaces folder.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

I-04	Redundant imports in `SellingCurveRewardManager2`, `TokenConverter`
Severity	INFO
Status	• FIXED

Location

File	Location	Line
SellingCurveRewardManager2.sol	None	7
SellingCurveRewardManager2.sol	None	8
SellingCurveRewardManager2.sol	None	9
TokenConverter.sol	None	7

Description

The imports in the mentioned locations are redundant and can be removed.

Recommendation

We recommend removing unused imports to keep the codebase clean.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

I-05	Functions visibility not optimal in `SellingCurveRewardManager2`, `TokenConverter`
Severity	INFO
Status	• FIXED

Location

File	Location	Line
SellingCurveRewardManager2.sol	contract `SellingCurveRewardManager2` > function `handle`	41
SellingCurveRewardManager2.sol	contract `SellingCurveRewardManager2` > function `valuate`	65
TokenConverter.sol	contract `TokenConverter` > function `handle`	54

Description

In the mentioned locations, functions are declared as `public`. However, they do not appear to be called from within the contracts in the codebase. Suppose a function is designed to be called by users and is not intended to be called by the other functions. In that case, it is better to declare them as `external` to reduce the gas cost associated with their execution.

Recommendation

We recommend declaring functions designed to be called externally as `external`.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

I-06	Wrong naming in `ZunEthOracle`
Severity	INFO
Status	• FIXED

Location

File	Location	Line
ZunEthOracle.sol	contract `ZunEthOracle` > function `_getZunUSDPriceForCurvePool`	36

Description

In the function `_getZunUSDPriceForCurvePool` of the contract `ZunEthOracle`, there is a function naming error: the name `_getZunUSDPriceForCurvePool` is used instead of `_getZunETHPriceForCurvePool`.

Recommendation

We recommend fixing the issue by replacing names of the function `_getZunUSDPriceForCurvePool` with `_getZunETHPriceForCurvePool`.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

I-07	`Constant` type name style in `CrvUsdStakeDaoERC4626StratBase`
Severity	INFO
Status	• FIXED

Location

File	Location	Line
CrvUsdStakeDaoERC4626StratBase.sol	contract `CrvUsdStakeDaoERC4626StratBase`	15

Description

In the contract CrvUsdStakeDaoERC4626StratBase.sol, the `constant` type is used. However, its name is not in `UPPER_CASE_WITH_UNDERSCORES` style as recommended in the Solidity documentation.

Recommendation

We recommend using the `UPPER_CASE_WITH_UNDERSCORES` name style for the `constant` type.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

I-08	Redundant variable in `SdtOracle`
Severity	INFO
Status	• FIXED

Location

File	Location	Line
SdtOracle.sol	contract `SdtOracle`	20
SdtOracle.sol	contract `SdtOracle`	22
SdtOracle.sol	contract `SdtOracle`	23

Description

In the contract `SdtOracle` the variable `_genericOracle` is redundant and can be removed.

Recommendation

We recommend removing the redundant variable.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

I-09	Confusing naming in `SellingCurveRewardManager2`
Severity	INFO
Status	• FIXED

Location

File	Location	Line
SellingCurveRewardManager2.sol	contract `SellingCurveRewardManager2` > function `handle`	57

Description

In the function `handle` of the contract `SellingCurveRewardManager2`, the variable `feeTokenAmount` is not participating in any fee-related logic, the associated token is named `receivingToken` according to the function parameter.

Recommendation

We recommend renaming the variable to `receivingTokenAmount`.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

I-10	Redundant cast in `SellingCurveRewardManager2`
Severity	INFO
Status	• FIXED

Location

File	Location	Line
SellingCurveRewardManager2.sol	contract `SellingCurveRewardManager2` > function `handle`	58

Description

In the function `handle` of the contract `SellingCurveRewardManager2`, the cast `address(msg.sender)` is redunant.

Recommendation

We recommend removing the redundant cast.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

I-11	Cache array length outside of loop in `TokenConverter`
Severity	INFO
Status	• FIXED

Location

File	Location	Line
TokenConverter.sol	contract `TokenConverter` > function `setRoutes`	44
TokenConverter.sol	contract `TokenConverter` > function `_fillRoutes`	89
TokenConverter.sol	contract `TokenConverter` > function `_fillSwapParams`	98

Description

In the mentioned locations, the array length is not stored in a variable before the `for` loop is executed. If the length is not cached, the Solidity compiler will read the array length every time during each iteration.

Therefore, if it is a `storage` array, this leads to an additional `sload` operation, and if it is a `memory` array, then an additional `mload` operation.

Recommendation

We recommend caching the array length in the additional variable.

Update

Client's response

Fixed in the commit `3851418c125703fdc3513a76672725de042a1ebb`.

3 CONCLUSION

The Zunami protocol security audit identified a range of vulnerabilities classified as critical, major, warning, and informational. The test coverage of the reward accounting logic was found insufficient to guarantee the correct performance of the protocol.

The suggested corrections were introduced by the procotol team, the final commit with fixes is `3851418c125703fdc3513a76672725de042a1ebb`.

The following table contains all the findings identified during the audit, grouped by statuses and severity levels:

Severity	FIXED	NO ISSUE	Total
CRITICAL	1	0	1
MAJOR	1	0	1
WARNING	2	1	3
INFO	11	0	11
Total	15	1	16

THANK YOU FOR CHOOSING