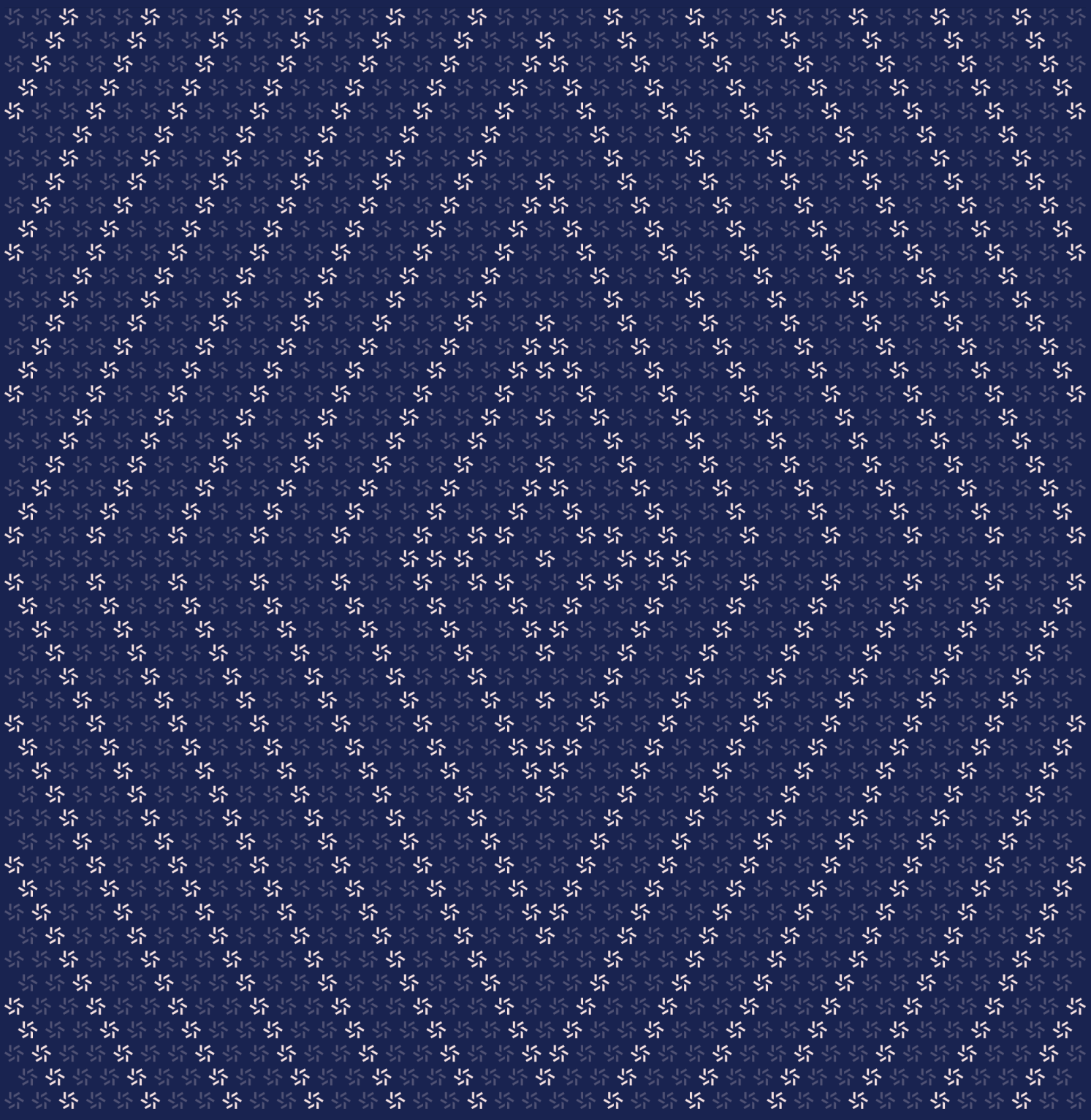
Zellic

Prepared for
Kester Wu
Sailor Finance

Prepared by
Dimitri Kamenski
Sylvain Pelissier
Zellic

February 5, 2025

Core and Periphery
Smart Contract Patch Review



Contents

About Zellic

Zellic is a vulnerability research firm with deep expertise in blockchain security. We specialize in EVM, Move (Aptos and Sui), and Solana as well as Cairo, NEAR, and Cosmos. We review L1s and L2s, cross-chain protocols, wallets and applied cryptography, zero-knowledge circuits, web applications, and more.

Prior to Zellic, we founded the #1 CTF (competitive hacking) team ↗ worldwide in 2020, 2021, and 2023. Our engineers bring a rich set of skills and backgrounds, including cryptography, web security, mobile security, low-level exploitation, and finance. Our background in traditional information security and competitive hacking has enabled us to consistently discover hidden vulnerabilities and develop novel security research, earning us the reputation as the go-to security firm for teams whose rate of innovation outpaces the existing security landscape.

For more on Zellic's ongoing security research initiatives, check out our website zellic.io ↗ and follow @zellic_io ↗ on Twitter. If you are interested in partnering with Zellic, contact us at hello@zellic.io ↗.

1. Overview

1.1. Executive Summary

Zellic conducted a security assessment for Sailor Finance from January 28th to January 29th, 2025. During this engagement, Zellic reviewed Core and Periphery's code for security vulnerabilities, design issues, and general weaknesses in security posture.

Core and Periphery is a fork of Uniswap V3, and we were asked to review minor patches, which changed the protocol-fee distribution. In section 4.1. ↗, we have provided an overview of the changes and the commits we used for the diffs.

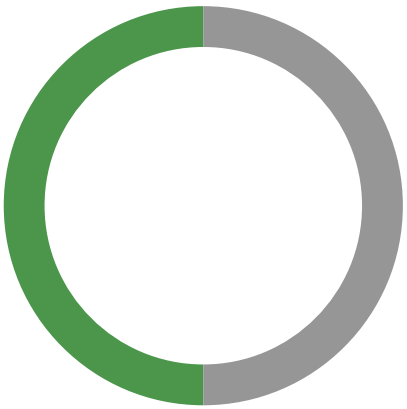
1.2. Results

During our assessment on the scoped Core and Periphery contracts, we discovered two findings. No critical issues were found. One finding was of low impact and the other finding was informational in nature.

Additionally, Zellic recorded its notes and observations from the assessment for the benefit of Sailor Finance in the Discussion section (4. ↗).

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	1
Informational	1



2. Introduction

2.1. About Core and Periphery

Sailor Finance contributed the following description of Core and Periphery:

> Sailor Finance is a native spot DEX built on Sei Network.

2.2. Scope

The engagement involved a review of the following targets:

Core and Periphery Contracts

Type	Solidity
Platform	EVM-compatible
Target	v3-core
Repository	https://github.com/capybaralabs-xyz/v3-core ↗
Version	`8064f9689124d5fafdc364b12b9b19ef12adacbf`

Programs	`UniswapV3Factory`
	`UniswapV3Pool`
	`IUniswapV3PoolEvents`
	`IUniswapV3PoolOwnerActions`
	`IUniswapV3PoolState`
	`TransferHelper`

Target	v3-periphery
Repository	https://github.com/capybaralabs-xyz/v3-periphery ↗
Version	`4b2709972be3f10f665e14c24781b0c20129d677`
Programs	`NonfungiblePositionManager` `NonfungibleTokenPositionDescriptor`

Contact Information

The following project managers were associated with the engagement:

Jacob Goreski
Engagement Manager
jacob@zellic.io ↗

Chad McDonald
Engagement Manager
chad@zellic.io ↗

The following consultants were engaged to conduct the assessment:

Dimitri Kamenski
Engineer
dimitri@zellic.io ↗

Sylvain Pelissier
Engineer
sylvain@zellic.io ↗

3. Detailed Findings

3.1. Incorrect chain-ID comparison

Target	NonfungibleTokenPositionDescriptor		
Category	Coding Mistakes	**Severity**	Low
Likelihood	Medium	**Impact**	Low

Description

The `tokenRatioPriority` function is used to build token URI for NFT positions. The goal of this function is to reorder the tokens in the URI to always keep some tokens at the first position. Sailor made the following change to the function:

```
function tokenRatioPriority(address token, uint256 chainId)
    public view returns (int256) {
    if (token == WETH9) {
        return TokenRatioSortOrder.DENOMINATOR;
        return TokenRatioSortOrder.NUMERATOR;
    }
    if (chainId == 1) {
        if (token == USDC) {
        if (token == USDT) {
            return TokenRatioSortOrder.NUMERATOR_MOST;
        } else if (token == USDT) {
        } else if (token == USDC) {
            return TokenRatioSortOrder.NUMERATOR_MORE;
        } else if (token == DAI) {
            return TokenRatioSortOrder.NUMERATOR;
        } else if (token == TBTC) {
        } else if (token == WETH) {
            return TokenRatioSortOrder.DENOMINATOR_MORE;
         } else if (token == WBTC) {
            return TokenRatioSortOrder.DENOMINATOR_MOST;
                    } else {
                return 0;
            }
        }
        return 0;
    }
}
```

For example, the USDT token will be placed first in the URI and the token WBTC always last. However, the changes did not update the `chainID` in the comparison. It should be compared to 1,329, which is the Sei network chain ID, and not 1, which is the Ethereum chain ID.

Impact

Since the chain-ID parameter is compared with 1, the reordering would never happen as expected for the WETH9 token. The token URI would not contain the token in the correct order, and it may lead to incorrect or duplicate URI creations.

Recommendations

We recommend updating the chain-ID comparison with the correct ID for the Sei network.

Remediation

This issue has been acknowledged by Sailor Finance.

3.2. Fee unit type exceeds the first slot

Target	UniswapV3Pool		
Category	Coding Mistakes	**Severity**	Informational
Likelihood	N/A	**Impact**	Informational

Description

The `Slot0` structure in Uniswap's UniswapV3Pool contract is intended to take only one storage slot. This ensures that, in reading and writing to variables in that slot, all of them are warmed, ensuring that there is minimal gas expenses for `SLOAD` and `SSTORE` operations. The total storage space taken in Uniswap is 241 bits.

The current struct configuration is as follows:

```
struct Slot0 {
    uint160 sqrtPriceX96;
    int24 tick;
    uint16 observationIndex;
    uint16 observationCardinality;
    uint16 observationCardinalityNext;
    uint32 feeProtocol;
    bool unlocked;
}
```

After the modifications, the structure is 265 bits and therefore expands beyond the 256 bits per slot. This will not only increase the cost of write operations during `setFeeProtocol()`, it will also add one `SLOAD` operation for any `Slot0` read. This will increase the cost of optimized functions, such as `_modifyPosition()` and `swap()`.

Impact

Changes described here will have impacts on gas, which could cost users unnecessarily over many swaps and position modifications.

Recommendations

If attempting to scale the fees, provide a fee multiplier that scales the `uint8 feeProtocol` up by a factor of 100–1,000. This will keep `slot0` as mostly one read or write operation on a warmed slot.

Remediation

This issue has been acknowledged by Sailor Finance.

4. Discussion

The purpose of this section is to document miscellaneous observations that we made during the assessment. These discussion notes are not necessarily security related and do not convey that we are suggesting a code change.

4.1. Patch review

This section documents notable and minor changes applied to the in-scope code from commit `d8b1c635` to commit `8064f968` in v3-core repository and from commit `d8b1c635` to commit `8064f968` in v3-periphery repository.

Notable changes

The main purpose of the changes is to have a protocol fee defined on 32 bits instead of the original 8 bits. With the new changes, the upper 16 bits of the protocol fee represent the conversion fee for token0-to-token1 conversion, and the lower 16 bits represent the fee for token1-to-token0 conversion. The initial value set by the `initialize` function is `104859200` instead of `0` previously, resulting in (1,600, 1,600) or 16% fees for each conversion. Both fees can be set to a minimum of 1,000 and a maximum of 4,000 by the `setFeeProtocol` function.

The TransferHelper contract uses the IERC20 contract from OpenZeppelin instead of the internal IERC20Minimal contract. It implements the new functions `safeApprove`, `safeTransferFrom`, and `safeTransferETH`.

Minor differences

There were minor changes to the codebase to make the contracts compatible with the Sei network, like token-address updates. The changes also add a tick spacing of 1 in the UniswapV3Factory contract.

4.2. Fee-protocol type could be adapted

Even if the fee protocols are checked to be in the correct range by the `setFeeProtocol` function, they could be declared as `uint16` instead of `uint32` to avoid confusion. Also, the following comment in the `Slot0` structure could be changed as follows:

```
// 2 uint32 values store in a uint32 variable (fee/PROTOCOL_FEE_DENOMINATOR)
// 2 uint16 values store in a uint32 variable (fee/PROTOCOL_FEE_DENOMINATOR)
uint32 feeProtocol;
```

4.3. Remove Etherscan API key from the configuration

An Etherscan API key is publicly listed in the hardhat.config.ts file at lines 62–64. We were not able to confirm if this was a valid API key, but this kind of credential should not be publicly available.

4.4. Original code linting action should be kept

Linting and code formatting have been removed, which increases the number of lines changed on a small amendment to battle-tested code. We advise, if forking the entire codebase and introducing small changes, to keep the linting and the code formatting in the GitHub Actions to avoid adding accidental errors.

4.5. Misleading `safeApprove` function name

In the TransferHelper contract, the newly included function `safeApprove` mimics the behavior of the `forceApprove` function from OpenZeppelin ↗:

```
function safeApprove(address token, address to, uint256 value) internal {
    (bool success, bytes memory data)
    = token.call(abi.encodeWithSelector(IERC20.approve.selector, to, value));
    require(success && (data.length == 0 || abi.decode(data, (bool))), 'SA');
}
```

However, OpenZeppelin previously received a `safeApprove` function, which has since been deprecated ↗. To avoid confusion, the `safeApprove` function of the TransferHelper contract may be renamed in the contract to `forceApprove`.

4.6. Security contacts are not correct

Both README.md files in Core and Periphery repositories are not up-to-date. Here is the **Bug bounty** section from the files:

```
------------
## Bug bounty
```

```
This repository is subject to the Uniswap V3 bug bounty program, per the terms
    defined [here](./bug-bounty.md).
```

It mentions the Uniswap bug bounty instead of Sailor's bug bounty program or security contacts. To help security researchers to report security vulnerabilities, clear instructions regarding the project should be available.

5. Assessment Results

At the time of our assessment, the reviewed code was deployed to Sei network.

During our assessment on the scoped Core and Periphery contracts, we discovered two findings. No critical issues were found. One finding was of low impact and the other finding was informational in nature.

5.1. Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any code added to the project after the version reviewed during our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code samples in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code. These recommendations are not exhaustive, and we encourage our partners to consider them as a starting point for further discussion. We are happy to provide additional guidance and advice as needed.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.