



# MythX Report

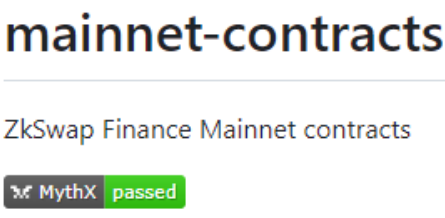
## Overview

Project Name	zkSwap Finance
Auditor	MythX.io
Source Code	<a href="https://github.com/ZkSwapFinance/zf-periphery/tree/main/contracts/paymaster">https://github.com/ZkSwapFinance/zf-periphery/tree/main/contracts/paymaster</a>
Mode	Deep
Time	Sat Dec 23 <sup>rd</sup> 2023
<b>DETECTED VULNERABILITIES</b>	<b>1</b>

## Summary

Done	Contract	High Risk Issues	Medium Risk Issues	Low Risk Issues
<input checked="" type="checkbox"/>	ZFPaymaster.sol	0	0	1

## Reference

ZFPaymaster.sol	<a href="https://github.com/ZkSwapFinance/Audit-Reports/blob/main/Original_MythX_ZFPaymaster.pdf">https://github.com/ZkSwapFinance/Audit-Reports/blob/main/Original_MythX_ZFPaymaster.pdf</a>
<p>MythX Passed Badge on Github</p>  <p>mainnet-contracts</p> <p>ZkSwap Finance Mainnet contracts</p> <p>MythX passed</p>	<a href="https://github.com/ZkSwapFinance/zf-periphery">https://github.com/ZkSwapFinance/zf-periphery</a>

REPORT 6586736FF1BCF1001A61D50C

Created	Sat Dec 23 2023 05:43:11 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	648fc02af4bf584372592643

## REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
<a href="#">8ff494f1-1d93-472f-a4cb-a1a795695443</a>	/paymaster/zfpaymaster.sol	1

Started	Sat Dec 23 2023 05:43:22 GMT+0000 (Coordinated Universal Time)
Finished	Sat Dec 23 2023 06:28:46 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Vscode-Extension
Main Source File	/Paymaster/Zfpaymaster.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

UNKNOWN Arithmetic operation "\*" discovered  
This plugin produces issues to support false positive discovery within MythX.  
SWC-101

Source file  
/paymaster/zfpaymaster.sol  
Locations

```
87 |  
88 | try  
89 | ERC20(token).transferFrom(address(uint160(_transaction.from)), address(this), priceForPayingFees)  
90 | {} catch (bytes memory revertReason) {  
91 | // If the revert reason is empty or represented by just a function selector,
```

UNKNOWN Arithmetic operation "/" discovered  
This plugin produces issues to support false positive discovery within MythX.  
SWC-101

Source file  
/paymaster/zfpaymaster.sol  
Locations

```
137 |  
138 | address NATIVE_TOKEN = 0x0000000000000000000000000000000000000000000000000000000000000000;  
139 | if (_token == NATIVE_TOKEN) {  
140 | bool success = payable(msg.sender).call{ value: _value }("");  
141 | require(success, "_safeTransferETH: failed");  
142 | }
```

## UNKNOWN Arithmetic operation "\*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

/paymaster/zfpaymaster.sol

Locations

```
137 |
138 | address NATIVE_TOKEN = 0x0000000000000000000000000000000000000000000000000000000000000000;
139 | if (_token == NATIVE_TOKEN) {
140 |     bool success = payable(msg.sender).call{ value: _value }("");
141 |     require(success, "_safeTransferETH: failed");
142 | }
```

## UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

/paymaster/zfpaymaster.sol

Locations

```
138 | address NATIVE_TOKEN = 0x0000000000000000000000000000000000000000000000000000000000000000;
139 | if (_token == NATIVE_TOKEN) {
140 |     (bool success, _) = payable(msg.sender).call{ value: _value }("");
141 |     require(success, "_safeTransferETH: failed");
142 | }
```

LOW

## A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

SWC-103

Source file

/paymaster/zfpaymaster.sol

Locations

```
1 | // SPDX-License-Identifier: MIT
2 | pragma solidity ^0.8.0;
3 |
4 | import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
```

## UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

/paymaster/zfpaymaster.sol

Locations

```
134 | }
135 |
136 | function withdrawFee(address _token, uint256 _value) external onlyOwner {
137 |
138 | address NATIVE_TOKEN = 0x0000000000000000000000000000000000000000000000000000000000000000;
139 | if (_token == NATIVE_TOKEN) {
140 | (bool success, ) = payable(msg.sender).call{ value: _value }("");
```

## UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

/paymaster/zfpaymaster.sol

Locations

```
136 | function withdrawFee(address _token, uint256 _value) external onlyOwner {
137 |
138 | address NATIVE_TOKEN = 0x0000000000000000000000000000000000000000000000000000000000000000;
139 | if (_token == NATIVE_TOKEN) {
140 | (bool success, ) = payable(msg.sender).call{ value: _value }("");
```

## UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

/paymaster/zfpaymaster.sol

Locations

```
138 | address NATIVE_TOKEN = 0x0000000000000000000000000000000000000000000000000000000000000000;
139 | if (_token == NATIVE_TOKEN) {
140 | (bool success, ) = payable(msg.sender).call{ value: _value }("");
141 | require(success, "_safeTransferETH: failed");
142 | }
143 | else {
```