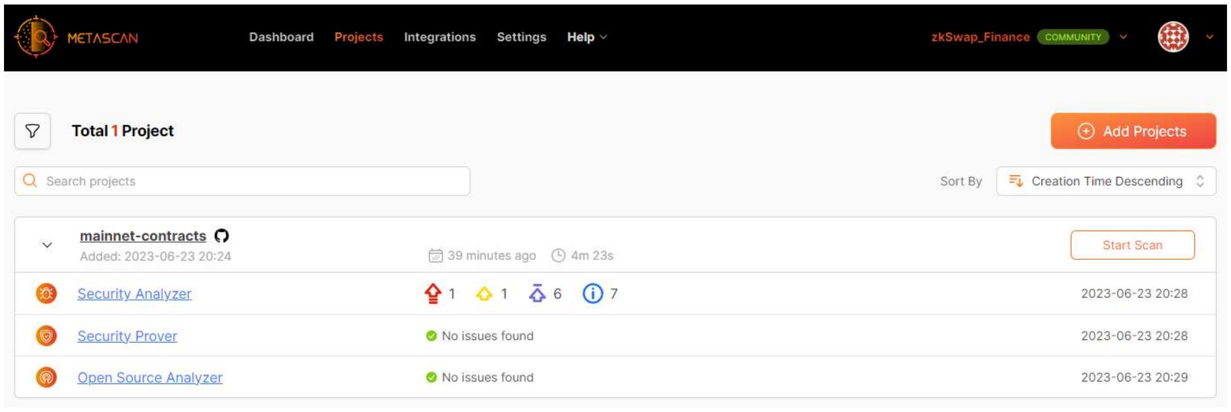
MetaTrust Report

Overview

Project Name	zkSwap Finance
Auditor	MetaTrust.io
Source Code	https://github.com/ZkSwapFinance/mainnet-contracts
Time	Fri Jun 23rd 2023

Summary

Scan Engine	Critical Issues	High Risk Issues	Medium Risk Issues	Low Risk Issues	Information Issues
Security Analyzer	1	0	1	6	7
Security Prover	0	0	0	0	0
Open-Source Analyzer	0	0	0	0	0



Total 1 Project

mainnet-contracts
Added: 2023-06-23 20:24
39 minutes ago 4m 23s

Security Analyzer	1 1 6 7	2023-06-23 20:28
Security Prover	No issues found	2023-06-23 20:28
Open Source Analyzer	No issues found	2023-06-23 20:29

METASCAN   Dashboard   Projects   Integrations   Settings   Help   zkSwap_Finance COMMUNITY

Projects > mainnet-contracts

mainnet-contracts

Start Scan   Scan Reports

Project added **2023/06/23 20:24**   Scan time **2023/06/23 20:25**   Scan ID **3793**

History   Scan Comparison   Project Settings

Vulnerability					
15	1	0	1	6	7
Total Issues	Critical	High	Medium	Low	Info

Code Duplication
-
Clone Rate

Code Quality			
-	-	-	-
Total Issues	Medium	Warning	Info

Completed
4m 23s
Total Duration

Security Analyzer (15)   Security Prover   Open Source Analyzer   IP Analyzer   Code Quality   AI Analyzer

Scan Duration	Solc Version	Node Version	Branch	Commit Hash
3m 11s	auto	16.15	main	3fe685b

Issues (15)   Scan Log

Search vulnerabilities ...   Expand All   Collapse All   Sort By   Severity Descending

MWE-099: Possibility of Price manipulation with Pool Reserves

Severity
Critical

File(s) Affected

contracts/libraries/protocol/ZFLibrary.sol   Report Inaccuracy   AI Suggestions   View on Github

```
        amounts = getAmountsOutUnchecked(factory, amountIn, path);
```

```
        amounts = getAmountsInUnchecked(factory, amountOut, path);
```

Description

Please check all child function call based on this expression or expression itself, the potential price manipulation risk may in it. for example, in some functions, certain variables used in `transfer` or `mint` or `return` procedures depend on another dangerous variable that derives its data from `balanceof`, `getReserve`, `totalSupply()` or `address(someAddress).balance` and is vulnerable to manipulation by flash loan.

Recommendation

It is recommended to use the chainlink oracle to obtain data, or to avoid relying on easily manipulated variables, or to use the TWAP mechanism.

MWE-100: Possibility of Price manipulation with Balances in contracts/core/Multicall.sol

Severity
Medium

File(s) Affected

contracts/core/Multicall.sol   Report Inaccuracy   AI Suggestions   View on Github

```
        balance = addr.balance;
```

Description

Please check all child function call based on this expression or expression itself, the potential price manipulation risk may in it. for example, in some functions, certain variables used in `transfer` or `mint` or `return` procedures depend on another dangerous variable that derives its data from `balanceof`, `getReserve`, `totalSupply()` or `address(someAddress).balance` and is vulnerable to manipulation by flash loan.

Recommendation

It is recommended to use the chainlink oracle to obtain data, or to avoid relying on easily manipulated variables, or to use the TWAP mechanism.

MWE-084: Possible Usage of a Variable before Declaration

Severity
Low

File(s) Affected

contracts/libraries/cryptography/ECDSA.sol   Report Inaccuracy   View on Github

```
            // ecrecover takes the signature parameters, and the only way to get them
            // currently is to use assembly.
            assembly {
                r := mload(add(signature, 0x20))
                vs := mload(add(signature, 0x40))
            }
            return tryRecover(hash, r, vs);
```

```
            bytes32 r;
```

Description

Detects the possible usage of a variable before the declaration is stepped over (either because it is later declared or declared in another scope).

Recommendation

Move all variable declarations before any variable usage, and ensure that reaching a variable declaration does not depend on some conditional if used unconditionally.

MWE-110: Missing Event Setter in contracts/core/ZFFactory.sol

Severity
Low

File(s) Affected

contracts/core/ZFFactory.sol   Report Inaccuracy   View on Github

```
    function setFeeTo(address _feeTo) external override onlyFeeToSetter {
        feeTo = _feeTo;
    }
```

```
    function setSwapFee(uint16 newFee) external override onlyFeeToSetter {
        require(newFee <= 1000, "Swap fee point is too high"); // 10%
        swapFee = newFee;
    }
```

```
    function setProtocolFeeFactor(uint8 newFactor) external override onlyFeeToSetter {
        require(protocolFeeFactor > 1, "Protocol fee factor is too high");
        protocolFeeFactor = newFactor;
    }
```

```
    function setFeeToSetter(address _feeToSetter) external override onlyFeeToSetter {
        pendingFeeToSetter = _feeToSetter;
    }
```

```
    function setSwapFeeOverride(address _pair, uint16 _swapFeeOverride) external override onlyFeeToSetter {
        ZFPair(_pair).setSwapFeeOverride(_swapFeeOverride);
    }
```

Description

Setter-functions must emit events

Recommendation

Emit events in setter functions

MWE-110: Missing Event Setter

Severity
Low

File(s) Affected

contracts/core/ZFPair.sol   Report Inaccuracy   View on Github

```
    function setSwapFeeOverride(uint16 _swapFeeOverride) external override {
        require(msg.sender == factory, 'FORBIDDEN');
        require(_swapFeeOverride <= 1000 || _swapFeeOverride == SWAP_FEE_INHERIT, 'INVALID_FEE');
        swapFeeOverride = _swapFeeOverride;
    }
```

Description

Setter-functions must emit events

Recommendation

Emit events in setter functions

MWE-105: Missing Zero Address Check in contracts/core/ZFFactory.sol

Severity
Low

File(s) Affected

contracts/core/ZFFactory.sol

Report Inaccuracy View on Github

```
    constructor(address _feeToSetter) {
```

```
        feeToSetter = _feeToSetter;
```

```
    function setFeeTo(address _feeTo) external override onlyFeeToSetter {
```

```
        feeTo = _feeTo;
```

```
    function setFeeToSetter(address _feeToSetter) external override onlyFeeToSetter {
```

```
        pendingFeeToSetter = _feeToSetter;
```

Description

This Function is lack of zero address check in important operation, which may cause some unexpected result.

Recommendation

Add check of zero address in important operation.

MWE-105: Missing Zero Address Check

Severity
Low

File(s) Affected

contracts/core/ZFPair.sol

Report Inaccuracy View on Github

```
    constructor(address _token0, address _token1) {
```

```
        token0 = _token0;
```

Description

This Function is lack of zero address check in important operation, which may cause some unexpected result.

Recommendation

Add check of zero address in important operation.

MWE-105: Missing Zero Address Check in contracts/core/ZFRouter.sol

Severity
Low

File(s) Affected

contracts/core/ZFRouter.sol

Report Inaccuracy View on Github

```
    constructor(address _factory, address _WETH) {
```

```
        factory = _factory;
```

Description

This Function is lack of zero address check in important operation, which may cause some unexpected result.

Recommendation

Add check of zero address in important operation.

MWE-108: Centralized Risk With Other in contracts/libraries/access/Ownable.sol

Severity
Info

File(s) Affected

contracts/libraries/access/Ownable.sol

Report Inaccuracy View on Github

```
    function renounceOwnership() public virtual onlyOwner {
        _transferOwnership(address(0));
    }
```

```
    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        _transferOwnership(newOwner);
    }
```

Description

The contract has a centralized risk, which means that the contract is controlled by a single address. If the address is compromised, the contract will be compromised.

Recommendation

Avoid using centralized risk contracts.

MWE-086: Unused Internal Functions in contracts/core/ZFRouterInternal.sol

Severity
Info

File(s) Affected

contracts/libraries/utils/Context.sol

Report Inaccuracy View on Github

```
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }
```

```
    function _msgData() internal view virtual returns (bytes calldata) {
        return msg.data;
    }
```

contracts/core/ZFRouterInternal.sol

contracts/libraries/token/ERC20/ERC20Readonly.sol

Description

Presence of internal functions that are defined but never used in the contract. Such functions may introduce unnecessary gas consumption and make the code's review more difficult.

Recommendation

Remove unused functions to save gas and improve code readability.

METASCAN Dashboard Projects Integrations Settings Help

zkSwap_Finance COMMUNITY

Projects > mainnet-contracts

mainnet-contracts

Start Scan | Scan Reports

Project added **2023/06/23 20:24** Scan time **2023/06/23 20:25** Scan ID **3793**

History | Scan Comparison | Project Settings

Vulnerability					
15	1	0	1	6	7
Total Issues	Critical	High	Medium	Low	Info

Code Duplication
-
Clone Rate

Code Quality			
-	-	-	-
Total Issues	Medium	Warning	Info

Completed
4m 23s
Total Duration

Security Analyzer (15) | Security Prover | Open Source Analyzer | IP Analyzer | Code Quality | AI Analyzer

Scan Duration	Solc Version	Node Version	Branch	Commit Hash
3m 11s	auto	16.15	main	3fe685b

Issues (15) | Scan Log

Search vulnerabilities ... | Expand All | Collapse All | Sort By: Severity Descending

MWE-076: DoS with Block Gas Limit

Severity
Info

File(s) Affected

contracts/core/Multicall.sol

Report Inaccuracy | View on Github

```
    function aggregate(Call[] calldata calls) public returns (uint256 blockNumber, bytes[] memory returnData) {
        blockNumber = block.number;
        returnData = new bytes[](calls.length);
        for(uint256 i = 0; i < calls.length; i++) {
            (bool success, bytes memory ret) = calls[i].target.call(calls[i].callData);
            require(success);
            returnData[i] = ret;
        }
    }
    // Helper functions
    function getEthBalance(address addr) public view returns (uint256 balance) {
```

contracts/core/ZFRouterInternal.sol

contracts/core/ZFRouter.sol

Description

When smart contracts are deployed, functions inside them are called, and the execution of these actions always requires a certain amount of gas based on how much computation is needed to complete them. The Ethereum network specifies a block gas limit, and the sum of all transactions included in a block can not exceed the threshold.\nProgramming patterns that are harmless in centralized applications can lead to Denial of Service conditions in smart contracts when the cost of executing a function exceeds the block gas limit. For example, modifying an array of unknown size that increases over time can lead to a Denial of Service condition.

Recommendation

Caution is advised when you expect to have large arrays that grow over time. Actions that require looping across the entire data structure should be avoided.\n If you absolutely must loop over an array of unknown size, then you should plan for it to take multiple blocks and potentially require multiple transactions.

MWE-107: Centralized Risk With Key Variable Setting in contracts/core/ZFFactory.sol

Severity
Info

File(s) Affected

contracts/core/ZFFactory.sol

Report Inaccuracy | View on Github

```
    function setFeeTo(address _feeTo) external override onlyFeeToSetter {
        feeTo = _feeTo;
    }
```

```
    function setSwapFee(uint16 newFee) external override onlyFeeToSetter {
        require(newFee <= 1000, "Swap fee point is too high"); // 10%
        swapFee = newFee;
    }
```

```
    function setProtocolFeeFactor(uint8 newFactor) external override onlyFeeToSetter {
        require(protocolFeeFactor > 1, "Protocol fee factor is too high");
        protocolFeeFactor = newFactor;
    }
```

```
    function setFeeToSetter(address _feeToSetter) external override onlyFeeToSetter {
        pendingFeeToSetter = _feeToSetter;
    }
```

```
    function acceptFeeToSetter() external override {
        require(msg.sender == pendingFeeToSetter, 'FORBIDDEN');
        feeToSetter = pendingFeeToSetter;
    }
```

Description

The contract has a centralized risk, which means that the contract is controlled by a single address. If the address is compromised, the contract will be compromised.

Recommendation

Avoid using centralized risk contracts.

MWE-087: Uninitialized Local Variables

Severity
Info

File(s) Affected

contracts/libraries/protocol/ZFLibrary.sol

Report Inaccuracy | View on Github

```
        for (uint i; i < path.length - 1; ) {
```

Description

A local variable is either never initialized or is initialized only under certain conditions, while the variable will be used regardless of its initialization. As a result, the default zero value is used, which is not desired.

Recommendation

Initialize the local variable to a reasonable value. Explicitly setting it to zero if it is meant to be initialized to zero.

MWE-087: Uninitialized Local Variables in contracts/core/ZFRouter.sol

Severity
Info

File(s) Affected

contracts/core/ZFRouter.sol

Report Inaccuracy | View on Github

```
        for (uint i; i < path.length - 1; ) {
```

Description

A local variable is either never initialized or is initialized only under certain conditions, while the variable will be used regardless of its initialization. As a result, the default zero value is used, which is not desired.

Recommendation

Initialize the local variable to a reasonable value. Explicitly setting it to zero if it is meant to be initialized to zero.

MWE-087: Uninitialized Local Variables

Severity
Info

File(s) Affected

contracts/core/ZFRouterInternal.sol

Report Inaccuracy View on Github

```
        for (uint i; i < path.length - 1; ) {
```

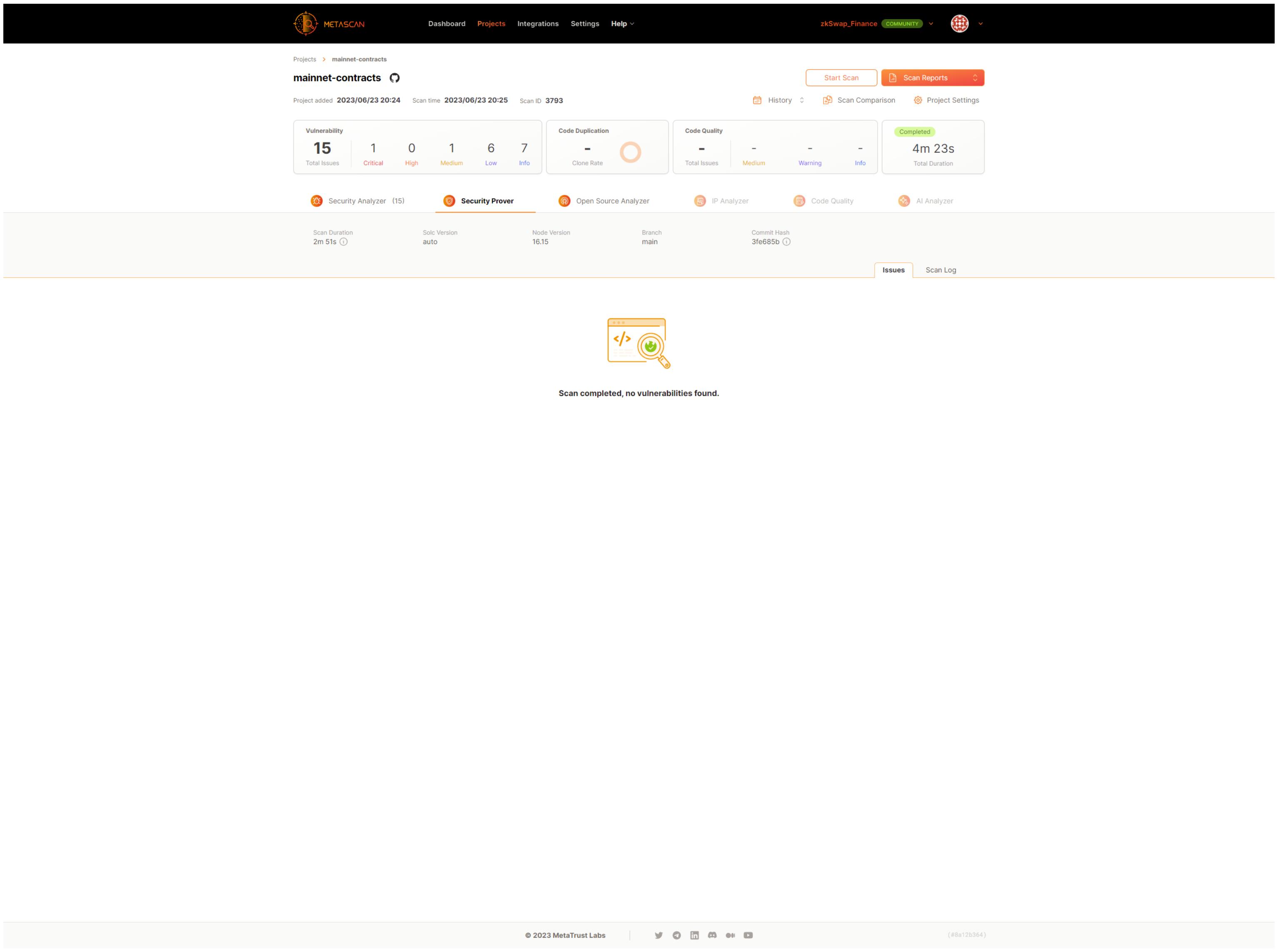
Description

A local variable is either never initialized or is initialized only under certain conditions, while the variable will be used regardless of its initialization. As a result, the default zero value is used, which is not desired.

Recommendation

Initialize the local variable to a reasonable value. Explicitly setting it to zero if it is meant to be initialized to zero.

1 2



Projects > mainnet-contracts

mainnet-contracts

Project added 2023/06/23 20:24 Scan time 2023/06/23 20:25 Scan ID 3793

Vulnerability

15	1	0	1	6	7
Total Issues	Critical	High	Medium	Low	Info

Code Duplication

-
Clone Rate

Code Quality

-	-	-	-
Total Issues	Medium	Warning	Info

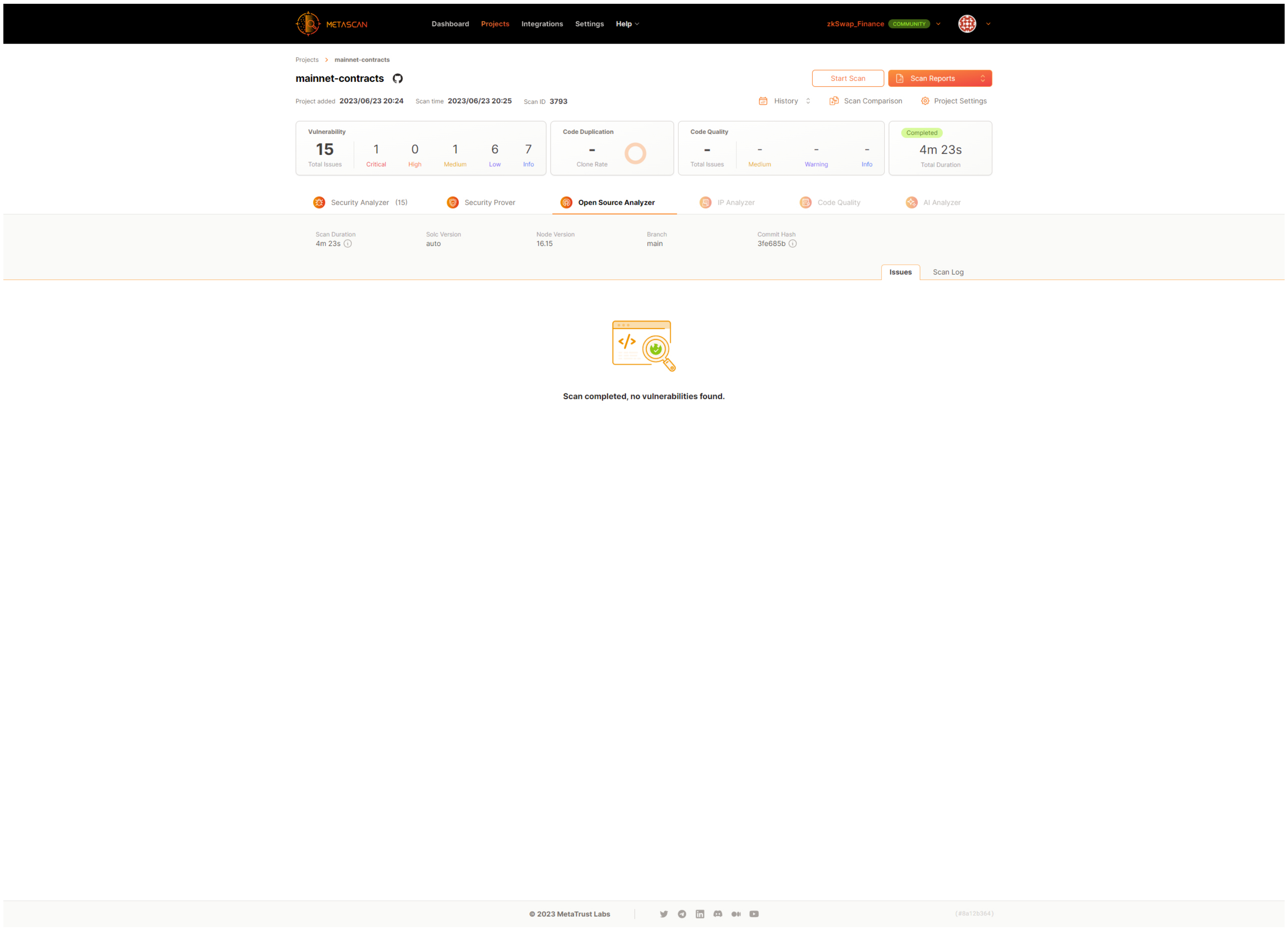
Completed

4m 23s
Total Duration

Scan Duration	Solc Version	Node Version	Branch	Commit Hash
2m 51s	auto	16.15	main	3fe685b

Scan completed, no vulnerabilities found.

(#8a12b364)



mainnet-contracts

Project added 2023/06/23 20:24 Scan time 2023/06/23 20:25 Scan ID 3793

Vulnerability					
15	1	0	1	6	7
Total Issues	Critical	High	Medium	Low	Info

Code Duplication
-
Clone Rate

Code Quality			
-	-	-	-
Total Issues	Medium	Warning	Info

Completed
4m 23s
Total Duration

Scan Duration	Solc Version	Node Version	Branch	Commit Hash
4m 23s	auto	16.15	main	3fe685b

Scan completed, no vulnerabilities found.

© 2023 MetaTrust Labs

(#8a126364)