Competitive Security Assessment

zkSync

Dec 5th, 2022

Secure3

secure3.io

Summary

zkSync is a layer 2 scaling solution built on Ethereum. zkSync uses advanced cryptography called zero-knowledge proofs to enable high-speed transactions at low cost, without compromising security or decentralization. By solving the blockchain scalability trilemma, zkSync promises to be the endgame for scaling Ethereum, and accelerate the mass adoption of crypto for personal sovereignty.

This report has been prepared for the project to identify issues and vulnerabilities in the smart contract source code. A comprehensive examination with Static Analysis and Manual Review techniques has been performed by Secure3 team. Also, a group of NDA covered experienced security experts have participated in the Secure3's Audit Contest as well to provide extra auditing coverage and scrutiny of the code.

The examination and auditing scope includes:

- Cross checking contract implementation against functionalities described in the documents and white paper disclosed by the project owner.
- Contract Privilege Role Review to provide more clarity on smart contract roles and privilege.
- Using static scanner to analyze smart contracts against common known vulnerabilities patterns.
- Verify the code base is compliant with the most up-to-date industry standards and best practices.
- Comprehensive line-by-line manual code review of the entire codebase by industry experts.

The security assessment resulted in findings that are categorized in four severity levels: Informational, Low, Medium, Critical. For each of the findings we have provided recommendation of a fix or mitigation for security and best practices.

Overview

Project Detail

Project Name	zkSync
Platform & Language	Solidity
Codebase	• repo - https://github.com/miladpiri/zksync • audit commit - 8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
Audit Methodology	• Audit Contest • Business Logic and Code Review • Privileged Roles Review • Static Analysis

Code Vulnerability Review Summary

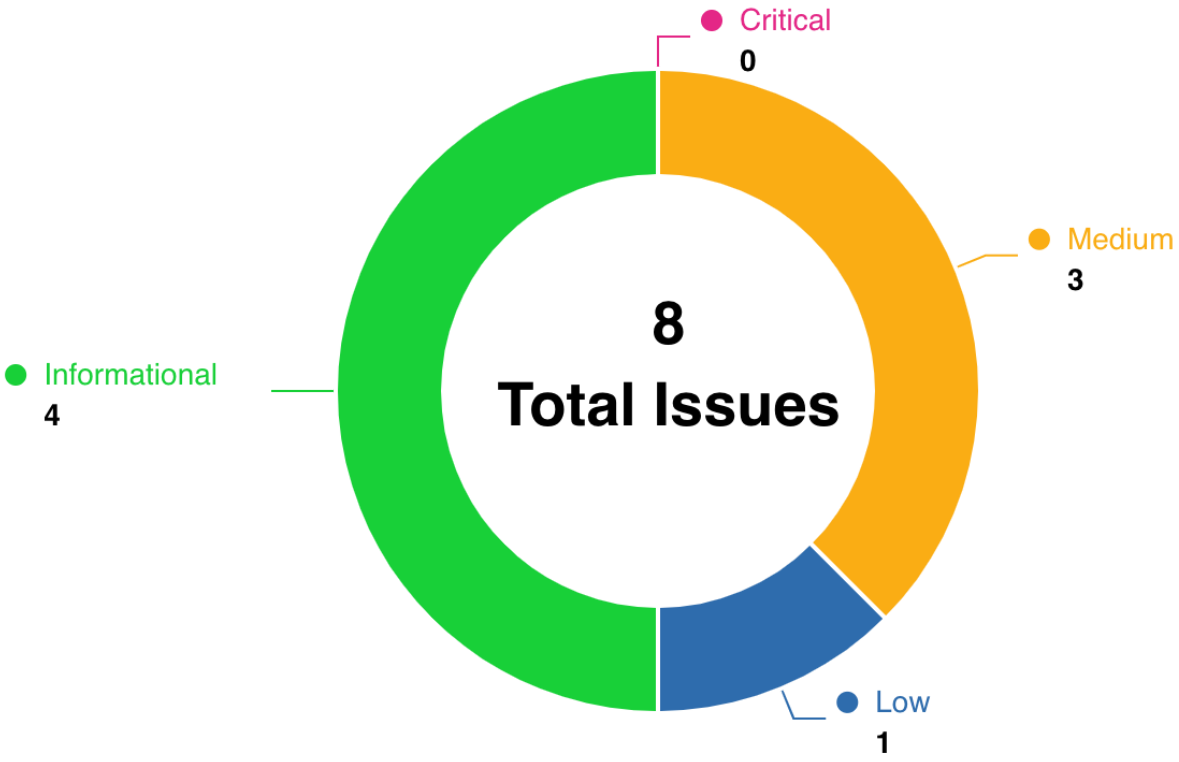
Vulnerability Level	Total	Reported	Acknowledged	Fixed	Mitigated	Declined
Critical	0	0	0	0	0	0
Medium	3	0	1	1	0	1
Low	1	0	0	0	0	1
Informational	4	0	2	2	0	0

Audit Scope

File	Commit Hash
contracts/zksync/facets/Executor.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/libraries/PairingsBn254.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/bridge/L1ERC20Bridge.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/libraries/Diamond.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/facets/Mailbox.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/bridge/L1EthBridge.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/facets/Getters.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/common/AllowList.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/facets/DiamondCut.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/facets/Governance.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/interfaces/IMailbox.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/common/L2ContractHelper.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/interfaces/IExecutor.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/Storage.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/DiamondInit.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/libraries/PriorityQueue.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/dev-contracts/Multicall.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/common/interfaces/IAllowList.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/libraries/TranscriptLib.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/interfaces/IGetters.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/common/ReentrancyGuard.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/Config.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/bridge/interfaces/IL1Bridge.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/DiamondProxy.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/common/libraries/UnsafeBytes.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/libraries/Merkle.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e

contracts/zksync/DiamondUpgradeInit.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/interfaces/IDiamondCut.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/interfaces/IGovernance.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/dev-contracts/RevertTransferERC20.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/dev-contracts/TestnetERC20Token.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/bridge/interfaces/IL2Bridge.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/common/interfaces/IERC20.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/facets/Base.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/common/libraries/UncheckedMath.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/dev-contracts/RevertReceiveAccount.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/common/AllowListed.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/zksync/interfaces/IZkSync.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/dev-contracts/RevertFallback.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e
contracts/common/Dependencies.sol	8bc57b7273a61b04d9ca96b5d3443f5a8f0a150e

Code Assessment Findings



ID	Name	Category	Severity	Status	Contributor
ZKS-1	Contract with payable function but lack of withdraw function	Logical	Medium	Declined	BradMoonUESTC
ZKS-2	Potential Security Issue if revealing secret `X`	Logical	Informational	Acknowledged	lfzkoala
ZKS-3	Should check whether the transfer is successful	Logical	Medium	Fixed	lfzkoala, iczc
ZKS-4	Useless mappings should be deleted	Gas Optimization	Informational	Fixed	zircon

ZKS-5	`Executor._blockMetaParameters()` unused input `_block`	Code Style	Informational	Fixed	lfzkoala
ZKS-6	`IL2Bridge` function `l1TokenAddress` and `l1Bridge` implementation never defined	Logical	Informational	Acknowledged	lfzkoala
ZKS-7	`msg.value` should be restricted when depositing ERC20	Logical	Low	Declined	zircon
ZKS-8	claimFailedDeposit logic may leads to double spend risk	Logical	Medium	Acknowledged	iczc

ZKS-1:Contract with payable function but lack of withdraw function

Category	Severity	Code Reference	Status	Contributor
Logical	Medium	code/contracts/zksync/DiamondUpgradeInit.sol#L10	Declined	BradMoonUESTC

Code

```
10:contract DiamondUpgradeInit is MailboxFacet {
```

Description

BradMoonUESTC : In contract `DiamondUpgradeInit` has payable function `forceDeployL2Contract`, but it lacks of withdraw function or logic, this may lead to ETH locked in the contract.

Recommendation

BradMoonUESTC : Add `EmergencyWithdraw` or similar function to allow withtdraw.

Client Response

The function is used only via delegate call on the diamond initialization, so we do not consider the possibility of withdrawing funds from this contract, same as for facet/target/implementation contracts.

ZKS-2:Potential Security Issue if revealing secret `X`

Category	Severity	Code Reference	Status	Contributor
Logical	Informational	code/contracts/zksync/facets/Executor.sol#L334-L343	Acknowledged	lfzkoala

Code

```
334:         PairingsBn254.G2Point memory g2Gen = PairingsBn254.new_g2(
335:             [
336:                 0x198e9393920d483a7260bfb731fb5d25f1aa493335a9e71297e485b7aef312c2,
337:                 0x1800deef121f1e76426a00665e5c4479674322d4f75edadd46debd5cd992f6ed
338:             ],
339:             [
340:                 0x090689d0585ff075ec9e99ad690c3395bc4b313370b38ef355acdadcd122975b,
341:                 0x12c85ea5db8c6deb4aab71808dcb408fe3d1e7690c43d37b4ce6cc0166fa7daa
342:             ]
343:         );
```

Description

lfzkoala : The X value in Plonk is very important, if the X value is revealed then you reveal the trapdoor, then adversary can forge a valid recursive proof to cheat this verifier. Since I don't know how is this `g2X` is computed (hopefully it's securely computed) by `g2X = g2*X` and `X` is securely stored somewhere) and the implementation is in progress as shown in code/contracts/zksync/interfaces/IExecutor.sol#56, currently I have no idea how secure it is. At this moment I just would like to warn the zkSync team safely storing the trapdoor and generate the trapdoor securely and randomly.

Recommendation

lfzkoala : Make sure the security of the trapdoor X.

Client Response

Acknowledged. We are reusing trusted setup from v1

ZKS-3:Should check whether the transfer is successful

Category	Severity	Code Reference	Status	Contributor
Logical	Medium	code/contracts/bridge/L1ERC20Bridge.sol#L125 code/contracts/bridge/L1ERC20Bridge.sol#L248	Fixed	lfzkoala, iczc

Code

```
125:        _token.transferFrom(_from, address(this), _amount);

248:        _token.transfer(_to, _amount);
```

Description

lfzkoala : The transfer function defined in the interface seems no return value type, which contradicts to the IERC20 interface standard https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-Transfer-address-address-uint256- , in which `transfer` method returns a boolean value. Although it still emits a Transfer event, the function doesn't check whether the transfer is successful.
iczc : The `transferFrom` in the `_depositFunds()` does not use `safeTransferFrom` wrapper, this results the transfer status is not checked.

Recommendation

lfzkoala : Check whether the token transfer is successful.
iczc : Use SafeERC20 wrapper `using SafeERC20 for ERC20;`

Client Response

Fixed.

ZKS-4:Useless mappings should be deleted

Category	Severity	Code Reference	Status	Contributor
Gas Optimization	Informational	code/contracts/bridge/L1EthBridge.sol#L142 code/contracts/bridge/L1ERC20Bridge.sol#L182	Fixed	zircon

Code

```
142:        depositAmount[_depositSender][_l2TxHash] = 0;

182:        depositAmount[_depositSender][_l1Token][_l2TxHash] = 0;
```

Description

zircon : The special operator delete in Solidity is used to release space. To encourage active recycling of space, releasing space will return some gas.

```
// L1ERC20Bridge.sol
depositAmount[_depositSender][_l1Token][_l2TxHash] = 0;

// L1EthBridge.sol
depositAmount[_depositSender][_l2TxHash] = 0;
```

Recommendation

zircon :

```
// L1ERC20Bridge.sol
delete depositAmount[_depositSender][_l1Token][_l2TxHash];

// L1EthBridge.sol
delete depositAmount[_depositSender][_l2TxHash];
```

Client Response

Gas optimization, fixed.

ZKS-5: `Executor._blockMetaParameters()` unused input `_block`

Category	Severity	Code Reference	Status	Contributor
Code Style	Informational	code/contracts/zksync/facets/Executor.sol#L398	Fixed	lfzkoala

Code

```
398:     function _blockMetaParameters(CommitBlockInfo calldata _block) internal view returns (bytes memory) {
```

Description

lfzkoala : The function `_blockMetaParameters` doesn't use the input parameter calldata `_block`.

Recommendation

lfzkoala : Remove the input calldata `_block` and modify the function or use it if this parameter is needed in the logic.

Client Response

Fixed.

ZKS-6: `IL2Bridge` function `l1TokenAddress` and `l1Bridge` implementation never defined

Category	Severity	Code Reference	Status	Contributor
Logical	Informational	code/contracts/bridge/interfaces/IL2Bridge.sol#21 code/contracts/bridge/interfaces/IL2Bridge.sol#25	Acknowledged	lfzkoala

Code

```
21:    function l1TokenAddress(address _l2Token) external view returns (address);

25:    function l1Bridge() external view returns (address);
```

Description

lfzkoala : The `l1TokenAddress` and `l1Bridge` functions are undefined but their interfaces are defined.

Recommendation

lfzkoala : Make sure these interfaces are needed. If these methods are needed, add the corresponding function implementations.

Client Response

Implementation of these 2 functions was not included into scope as it is a part of L2 of the protocol.

ZKS-7: `msg.value` should be restricted when depositing ERC20

Category	Severity	Code Reference	Status	Contributor
Logical	Low	code/contracts/bridge/L1ERC20Bridge.sol#97-117	Declined	zircon

Code

```
97:    function deposit(
98:        address _l2Receiver,
99:        address _l1Token,
100:        uint256 _amount
101:    ) external payable nonReentrant senderCanCallFunction(allowList) returns (bytes32 txHash) {
102:        uint256 amount = _depositFunds(msg.sender, IERC20(_l1Token), _amount);
103:        require(amount > 0, "1T"); // empty deposit amount
104:
105:        bytes memory l2TxCalldata = _getDepositL2Calldata(msg.sender, _l2Receiver, _l1Token,
amount);
106:        txHash = zkSyncMailbox.requestL2Transaction{value: msg.value}(
107:            l2Bridge,
108:            0,
109:            l2TxCalldata,
110:            DEPOSIT_ERGS_LIMIT,
111:            new bytes[](0)
112:        );
113:
114:        depositAmount[msg.sender][_l1Token][txHash] = amount;
115:
116:        emit DepositInitiated(msg.sender, _l2Receiver, _l1Token, _amount);
117:    }
```

Description

zircon : When users deposit ERC20, they may attach many Ether coins due to operational errors, with no way to retrieve them, which will result in the loss of Ether coins.

Recommendation

zircon : Limit the maximum and minimum value of Ether that a user can attach.

Client Response

This is by design, because ETH is used to pay the fees, at line 106 the value is sent to the Mailbox which will keep ETH as a fee and deposit ERC20.

ZKS-8:claimFailedDeposit logic may leads to double spend risk

Category	Severity	Code Reference	Status	Contributor
Logical	Medium	code/contracts/bridge/L1EthBridge.sol#L115 code/contracts/bridge/L1ERC20Bridge.sol#L159	Acknowledged	iczc

Code

```
115:    function claimFailedDeposit(

159:    function claimFailedDeposit(
```

Description

iczc : The L1 bridge is designed to claim token assets that failed crossing the chain on L2, it only requires proving a specified log sent in a specific L2 block. However, when an asset is already crossed from L1 to L2 and the L2's `finalizeDeposit()` function also has been successfully executed, there is possibility that the asset is still claimable in L1 via the `claimFailedDeposit` function. However there is no code provided for L2 so it needs the project owner to confirm and prevent the potential double spend attack.

Recommendation

iczc : On the L1 claimFailedDeposit logic, it should check L2 asset status before claim failed deposit to prevent double spend attack.

Client Response

This is covered by the L2 implementation which was not in scope for this audit.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Invoices, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Invoice. This report provided in connection with the services set forth in the Invoices shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Invoice. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Secure3’s prior written consent in each instance.

This report is not an “endorsement” or “disapproval” of any particular project or team. This report is not an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Secure3 to perform a security assessment. This report does not provide any warranty or guarantee of free of bug of codes analyzed, nor do they provide any indication of the technologies, business model or legal compliancy.

This report should not be used in any way to make decisions around investment or involvement with any particular project. Instead, it represents an extensive assessing process intending to help our customers increase the quality of their code and high-level consistency of implementation and business model, while reducing the risk presented by cryptographic tokens and blockchain technology.

Secure3’s position on the final decisions over blockchain technologies and corresponding associated transactions is that each company and individual are responsible for their own due diligence and continuous security.

The assessment services provided by Secure3 is subject to dependencies and under continuing development. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.