CERTIK

Security Assessment

zklink

Jun 29th, 2021

Table of Contents

Summary

This report has been prepared for zklink to discover issues and vulnerabilities in the source code of the zklink project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	zklink
Platform	Ethereum, BSC
Language	Solidity
Codebase	https://github.com/zkLinkProtocol/zklink-contracts
Commit	376423ebffae89967c4e1a22a6b56b00649ffdb5 545a9b918719dbc9d388e4c1b272c51268ec632c

Audit Summary

Delivery Date	Jun 29, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

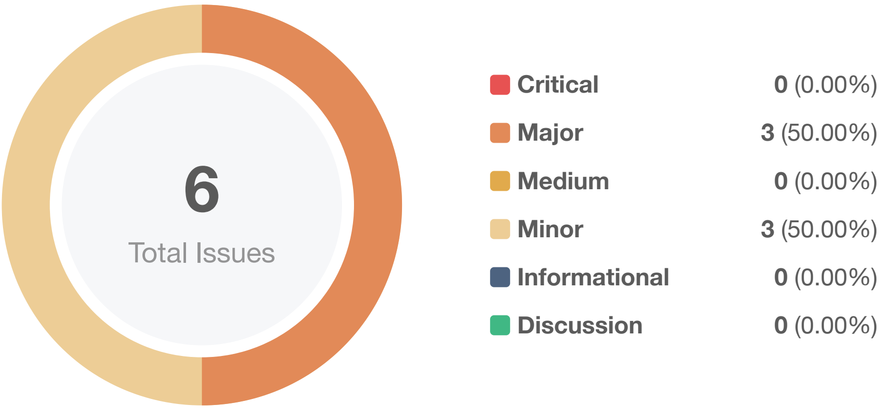
Vulnerability Level	Total Count	Pending	Partially Resolved	Resolved	Acknowledged	Declined
● Critical	0	0	0	0	0	0
● Major	3	0	0	2	1	0
● Medium	0	0	0	0	0	0
● Minor	3	0	0	3	0	0
● Informational	0	0	0	0	0	0
● Discussion	0	0	0	0	0	0

Audit Scope

ID	file	SHA256 Checksum
BCK	Bytes.sol	0a9301eeec89e3b24f82d145351b9eb848616aef1e8893723ddc4a2887756e06
CCK	Config.sol	921e5110e461cf55c76289b1ef02b2547e1818fda74b46a0a08c6fe46b6f0ba0
DFC	DeployFactory.sol	433fb1a74dfab71fabbc3b9f1d0e23d561b0cb641b74f4517a6daca482641645
ECK	Events.sol	fe12e52b2d4f72e67fe239f215e969564365e5e5cf5441e339fa1c3775cd4c72
GCK	Governance.sol	d4e32e55cfa7af711e857b771bc455c5b09ec98f87655c9886e1ca303fe7d72a
IER	IERC20.sol	32d2da4d516f3ad71619411f2dfe84c60d0f1f59627b68faff890f40f45456a3
ISC	IStrategy.sol	6e8a70a8f35dfea1c65002d8dfdf4634eb227a636ad46804a59693da0111d785
KWP	KeysWithPlonkVerifier.sol	fec093e4d62720cbe8152b8c3b92c9502552f8a46648cfcb2bf6217704c26dba
OCK	Operations.sol	d65f086d0cf06a13ae184c6219d8e1f1f1b3653f81ae15649ff8ae66897f1b22
OCP	Ownable.sol	91cf33acadc334e18d11a3900282dcf30b70cbed75c4dbef14f24f3d2090c27e
PTM	PairTokenManager.sol	fea51357e70ccdd1e66b886b03138658cf49621f91fa3d9633c0cc6ea2630fe2
PCC	PlonkCore.sol	99625d9aca5c8265a958738dffb43997cfaa06d6b063356e2b6e67d86812018f
PCK	Proxy.sol	f6ed65bbf1b6bb6de3f8eddd44bc6657ef12f34dc500ff94a1892450edadee94
RGC	ReentrancyGuard.sol	2ab736a9115321113b0200fbf3294d7cc3c3df7820b2e0d35aef0752a582dc5c
SCC	SafeCast.sol	918fac47cbc2c3cd918ab50297c2e4d46b7b75baaa379aff2a20a0f836ef1410
SMC	SafeMath.sol	33ef6bc4640ad3f611140e430bc5c22d91bf73c06b9dcbc20f664a72e3cd26a6
SMU	SafeMathUInt128.sol	7a5f67ab8cf3cd6a0406e7554fbe50b4dc195040f2ba714364fafd35b79f001c
SCK	Storage.sol	ff7cd1c8d553d7b0349d9ee4e6e5af8001b361e4bffd14d412f2b2b1687e4db0
TIC	TokenInit.sol	a0f3283cfbfe106653ff9b4d2dff45009b0c4c2e9ebac0c0c9f1adf5491d8207
UGC	UpgradeGatekeeper.sol	d4e9848f714eb285cbbef41652660700feebebe2af113239ea797200d596d64a
UCK	Upgradeable.sol	17934cf8c8fe588d512a82078e2dfeb02215e9c30069b4bf241f44f932921b49
UMC	UpgradeableMaster.sol	c28380536f2889ccb637ca84df23d75a50e4b256214289355b37a37e3ef8cdc1

ID	file	SHA256 Checksum
UCP	Utils.sol	55ebb83ebbcc2a24f7aa82d1d1152e8ef9833e97d44b9cc1306559c5f6f4e368
VCK	Vault.sol	4042af83c3c222ce4fe3bd4e365e95d83a93ecf3e7f4e288da269a4b43b3014f
VSC	VaultStorage.sol	d1d324c82f10cf8c99065a1bab5ead23b5f902f23ffbc42fe22147290a72a21f
VCP	Verifier.sol	c7ac907dbc435f9beb2d354b8a1e0df4b484467957d276cdf0bcfd75727138c4
ZSC	ZkSync.sol	2df6ab7105092b957f4096717a6efadbb8915c36fc2eeb5b6014c887c2f64d1f
ZSB	ZkSyncBase.sol	c0118acefc6f96c39ad34a6bf44305c908fd988aeb108fd69ec995da4e2cadf9
ZSK	ZkSyncBlock.sol	fbe5466cbabebf654e5592d243ee3c8e294fdaef9aa64b8d2c2b6c8f0e2aca78
UVE	uniswap/UniswapV2ERC20.sol	89b08a7622577092b370a47776ae6c7505686178f70f7f42660ab75c760b3eb3
UVF	uniswap/UniswapV2Factory.sol	ce0396990b7197c5225c12b15d63900139355d0ae42af6d06ba9393a582cb3a4
UVP	uniswap/UniswapV2Pair.sol	fdfa5c75d64cdb84472bb8cc1d42d068be45299a3a181e8823d01bff7a6a598e
IUN	uniswap/interfaces/IUNISWAPERC20.sol	3d3d515b9dbb6bab84e93ce043b38ddd521470b3466aab2d456ad2af394b97d8
IUV	uniswap/interfaces/IUniswapV2Callee.sol	4961b6738cefcb91594459c19092f3a9164e8876613f1674be82b263fcf636df
IUE	uniswap/interfaces/IUniswapV2ERC20.sol	ec3a050b2f0c2a0b1e5e23bff300bad14e2ca73d62ba7869fc9a73f0e4086c35
IUF	uniswap/interfaces/IUniswapV2Factory.sol	d528fe54456228c21306b70dca724979eaea785794fa72fc9990952e126a2919
IUP	uniswap/interfaces/IUniswapV2Pair.sol	6b8a911bd50f4c93259b2657eaf4fc0d3fded12dc52c66d375aa7d8a49e9ced6
MCK	uniswap/libraries/Math.sol	e4a9d451964a0689be2b244322a353de143ca4248d8736d91aca4ffadca4325f
UQC	uniswap/libraries/UQ112x112.sol	6633b57b0723b1d72e08cc3e8b29f0af838294e59863b6cdcce95a141ed02cdb
USM	uniswap/libraries/UniswapSafeMath.sol	2e9f5de7f01ab4ae9ce5d52d422d9ff5cbcec5ca702b8940894ab37ae397c633

Findings



ID	Title	Category	Severity	Status
UVF-01	Centralization Risk	**Centralization / Privilege**	● **Major**	✓ **Resolved**
UVP-01	Centralization Risk	**Centralization / Privilege**	● **Major**	✓ **Resolved**
VCK-01	Lack Of Access Restriction	Control Flow	● Minor	✓ Resolved
VCK-02	Lack Of Protection For Duplicate Initialization	Logical Issue	● Major	ⓘ Acknowledged
VCK-03	Lack of Sanity Check	Volatile Code	● Minor	✓ Resolved
ZSK-01	Lack of Sanity Check	Volatile Code	● Minor	✓ Resolved

UVF-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	uniswap/UniswapV2Factory.sol: 53, 58	✓ Resolved

Description

The address `zkSyncAddress` has the authority to call `mint()` and `burn` to mint/burn any amount of token to any address. Any compromise to the `owner` account may allow the hacker to take advantage of this and manipulate the economic system of the project.

Recommendation

We advise the client to carefully manage the `zkSyncAddress`'s private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO / governance/voting module to increase transparency and user involvement.

Alleviation

`[zklink]`: `zkSyncAddress` is the address of ZkSync contract and can only be set once at the initialization of the protocol

UVP-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	uniswap/UniswapV2Pair.sol: 38, 43	✓ Resolved

Description

The address `factory` has the authority to call `mint()` and `burn` to mint/burn any amount of token to any address. Any compromise to the `owner` account may allow the hacker to take advantage of this and manipulate the economic system of the project.

Recommendation

We advise the client to carefully manage the `factory`'s private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO / governance/voting module to increase transparency and user involvement.

Alleviation

`[zklink]`: only factory can mint or burn pair token, factory can not be set once at the initialization of the protocol.

VCK-01 | Lack Of Access Restriction

Category	Severity	Location	Status
Control Flow	● Minor	Vault.sol: 220	✓ Resolved

Description

`migrateStrategy()` function is designed to migrate a specific token's current strategy to its prepared strategy. Currently, any external caller can call this function to execute the strategy migration once the take-effect time reached.

Recommendation

We advise the client to add the `onlyNetworkGovernor` modifier to the function `migrateStrategy()` to only allow the `governor` to execute the strategy migration.

Alleviation

`[zklink]`: The client heeded the advice and added the `onlyNetworkGovernor` modifier to the function `migrateStrategy()` in the commit 545a9b918719dbc9d388e4c1b272c51268ec632c

VCK-02 | Lack Of Protection For Duplicate Initialization

Category	Severity	Location	Status
Logical Issue	● Major	Vault.sol: 46	ⓘ Acknowledged

Description

`initialize()` function is used to initialize the sensitive variable `governance`. This function can be called multiple times and can be taken advantage of by the hacker to gain the `governance` role and manipulate the project.

Recommendation

We advise the client to add an openzeppelin `initializer` modifier to the function `initialize()`.

Reference: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/proxy/utils/Initializable.sol

VCK-03 | Lack of Sanity Check

Category	Severity	Location	Status
Volatile Code	● Minor	Vault.sol: 187	✓ Resolved

Description

There's no check for the validation of `tokenId`.

Recommendation

We advise the client to check the validation of the `tokenId` in the function `upgradeStrategy()`:

```
function upgradeStrategy(address strategy) onlyNetworkGovernor external {
    ...
    uint16 tokenId = IStrategy(strategy).want();
    _validateToken(tokenId);
    TokenVault storage tv = tokenVaults[tokenId];
    ...
}
```

Alleviation

`[zklink]`: The client heeded the advice and added the `tokenId` validation in the commit 545a9b918719dbc9d388e4c1b272c51268ec632c

ZSK-01 | Lack of Sanity Check

Category	Severity	Location	Status
Volatile Code	● Minor	ZkSyncBlock.sol: 295	✓ Resolved

Description

There's no check for the validation of `tokenId`.

Recommendation

We advise the client to check the validation of the `tokenId` in the function `withdrawOrStore()`:

```
function withdrawOrStore(
    uint16 _tokenId,
    address _recipient,
    uint128 _amount
) internal {
    ...
    if (_tokenId >= PAIR_TOKEN_START_ID) {
        require(governance.tokenAddresses(tokenId) != address(0), 'ZkSyncBlock: token not
exist');
        address _token = tokenAddresses[_tokenId];
    ...
}
```

Alleviation

`[zklink]`: The client heeded the advice and added the validation in the commit 545a9b918719dbc9d388e4c1b272c51268ec632c

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

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 Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.