DeHacker

Code Security Assessment

ZKSwap V2 - Smart Contracts

July 1 8 th, 2023

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.
Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	ZKSwap V2 - Smart Contracts
Platform	Ethereum
Website	https://zks.org/en
Type	Dex
Language	Solidity

Vulnerability Summary

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

Audit scope

ID	File	SHA256 Checksum
BYT	Bytes.sol	1523ced00bea3489f2e87e84fead6bf3636 c9fd34fd52c226e4c90dde10b37ca
CON	Config.sol	6348269a182330505977ff1e87569ce9a6 d8c058266a100339e94e389cf5962d
DFS	DeployFactory.sol	2c0918e1aad6c3fe03d5f94aae74c2443c 0c3be026746c69933e686984165b86
IER	contracts/IERC20.sol	215e6566be35c9700ee4d29c4738bf46cb 78b72b2a8ba1072a71a6a2ff44305e
EVE	Events.sol	6073eb508c0287a7a3082754a36ec0a64f bae599772e2602877ae16d4c360acc
GOV	Governance.sol	610aa5dbed14698215f1cf0336184630d0 5e14815cfc9aa2ead4e117ce69e879
IER	IERC20.sol	1aa5fd202b00ab7a51871d5bd505693c4e 51d75b313da893d9d6e4f266ef9a98
OPE	Operations.sol	c55d1eeef547bd9e31cff6f6c179ad5b3b88 516a284eda472b215b53936f5126
OWN	Ownable.sol	50506267a96160bc02ce8f75329d219d0 7fab98afa2d6aa86d0b42c7d55c6b16
PTM	PairTokenManager. sol	af016f86aef851076cafe9bc303f30ae01f0 fe41c22447c6bd18e61ca2f0513e
PAC	PlonkAggCore.sol	f59465b49841718fc94ba83c4cf2421e39 25b605fb1adc47309cc1cf795a54cb
PCL	PlonkCoreLib.sol	e74084711aef34eb5c3e603d04086b19d6 7b28525ad1eff6f385a6c92957198c
PSC	PlonkSingleCore.sol	3ebbc38d537dddb7908a89ce8d6447265d c322731cd4f4fc51cc4557f0617c49
PRO	Proxy.sol	bc44491a71f683ac669fc2f6a094fd2ea7a 7fcd649e9539f300d3c9aa4998632
RGS	ReentrancyGuard.sol	8f40b887441b83b3980aa6b50ed3d1138 1e0140239edff8f76fe48d04b70aef3
SCS	SafeCast.sol	61446aabf5c02273653e8614232932618 072d31823422544d1aa513ac7011f05
SMS	SafeMath.sol	ccbc65eddc0fe23db1360af754dfc534f2a b28ab1d2e79c1ca0cc9420a96dc58
STO	Storage.sol	b3983af63201e62c8fe125e7dfe47e74d1 6685362d319681f7ae648377ee661a
UGS	UpgradeGatekeeper.sol	cdfb76f8d6fc0960a79558ba724ebdf329 2ca6280d1191449b3f49a7dbd6498a
UPG	Upgradeable.sol	b69175810f522879888a0ca10e6996543 22edfb7ec3e725676b6599727b88f35
UMS	UpgradeableMaster.sol	a7258d3796bccfd22efa4ab8dc801632d 085bfbdfc3182caca966c1abb986b9c
UTI	Utils.sol	48ef0cffe611d71651702b05be1a058aa7 b7e3a476278b7b1bd9042f233061f7

Audit scope

ID	File	SHA256 Checksum
VER	Verifier.sol	155ef0a71d7e1c26ca2b06108f29a6e35c0fb41dd8d4092ee90014e5c1dd8286
VES	VerifierExit.sol	51e654a6a36559330e53458910f69a58a42a2c0bb72ef6696ed72b6f37828e52
ZSS	ZkSync.sol	46b2ab9ebe10b2ed9ee220521997d23b2cfbb9036c9702459fc3ecf062fb39e8
ZSC	ZkSyncCommitBlock.sol	a1d015ba566eddeffbb428a07a316814fdfcb15cec157aef383fe43f45632cdc
ZSE	ZkSyncExit.sol	e3e6b0ca9b057356b847ea7b473738abc8df7a368826ce1065f11a4979ae0316
UVE	uniswap/UniswapV2ERC20.sol	ed6d25cf7c76843edff274ed524d4b1579a7ec31df1175e7af9599270203f02d
UVF	uniswap/UniswapV2Factory.sol	ce0396990b7197c5225c12b15d63900139355d0ae42af6d06ba9393a582cb3a4
UVP	UniswapV2Pair.sol	fdfa5c75d64cdb84472bb8cc1d42d068be45299a3a181e8823d01bff7a6a598e
IUN	niswap/interfaces/IUNISWAPERC20.sol	ac32d44f783bc838b821d813456781c23e3aeb57098257521b5ffd37a0d94dac
IUV	uniswap/interfaces/IUniswapV2Callee.sol	ecfa434d468bb1888124d8c34b0b8483f4ea1e5cd63f0874c8b28aa4305c1dbc
IUE	uniswap/interfaces/IUniswapV2ERC20.sol	dafbc6526a52910782ff58fcf72a79f636dc64bb97bac3822863adf0c2229208
IUF	uniswap/interfaces/IUniswapV2Factory.sol	618f366e7eddbf712eedc2caf1e1dded3702761f3502320f7c09c105ba6b559e
IUP	uniswap/interfaces/IUniswapV2Pair.sol	462948fb6e974114dc9fa7407420a2f727c79738ab19e4555721079dc9da67e1
MAT	uniswap/libraries/Math.sol	e4a9d451964a0689be2b244322a353de143ca4248d8736d91aca4ffadca4325f
UQS	uniswap/libraries/UQ112x112.sol	6633b57b0723b1d72e08cc3e8b29f0af838294e59863b6cdcce95a141ed02cdb
USM	UniswapSafeMath.sol	2e9f5de7f01ab4ae9ce5d52d422d9ff5cbcec5ca702b8940894ab37ae397c633
ERC	uniswap/test/ERC20.sol	e11b370820584a64b1ef5b498a348349f311787ba77b3b0a632abc1110dcc7e2

Findings

ID	Category	Severity	Status
VER-01	Coding Style	Informational	Acknowledged
ZSS-01	Coding Style	Informational	Acknowledged
ZSS-02	Coding Style	Informational	Acknowledged

INFORMATIONAL

VER-01 | Redundant Variable Initialization

Category	Severity	Location	Status
Coding Style	Informational	Verifier.sol: 9	Acknowledged

Description

The aforementioned line redundantly initializes bool type variable to false as the default value of a booltype variable is false .

Recommendation

We advise that the linked initialization statement is removed from the codebase to increase legibility.

INFORMATIONAL

ZSS-01 | Redundant Variable Initialization

Category	Severity	Location	Status
Coding Style	Informational	ZkSync.sol: 349, 369	Acknowledged

Description

The aforementioned lines redundantly initialize uint16 type variable to 0 as the default value of a uint16type variable is 0 .

Recommendation

We advise that the linked initialization statement is removed from the codebase to increase legibility.

INFORMATIONAL

ZSS-02 | Reusability of code is not observed

Category	Severity	Location	Status
Coding Style	Informational	ZkSync.sol: 347, 366	Acknowledged

Description

The functions withdrawERC20 and withdrawERC20WithAddress on the aforementioned lines contain thesame functionality with the difference being one registers withdrawal on msg.sender and the otherregisters withdrawal on parameter _to .

Recommendation

We advise to introduce a private function containing functionality from withdrawERC20WithAddress that iscalled by both of the aforementioned functions to increase code legibility.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

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.

Coding Style

Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code

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

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.

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.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

Ethereum

Eos

Cosmos

Substrate

TECH STACK

Python

Rust

Solidity

c++

CONTACTS

https://dehacker.io

https://twitter.com/dehackerio

https://github.com/dehacker/audits_public

https://t.me/dehackerio

https://blog.dehacker.io/

DeHacker

July 2023