Worldcoin

WLD Token

SMART CONTRACT AUDIT

01.09.2023

Made in Germany by Softstack.io

Table of contents

1. Disclaimer

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Worldcoin Foundation. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (15.08.2023)	Layout
0.4 (17.08.2023)	Automated Security Testing Manual Security Testing
0.5 (20.08.2023)	Verify Claims and Test Deployment
0.6 (22.08.2023)	Testing SWC Checks
0.9 (25.08.2023)	Summary and Recommendation
1.0 (01.09.2023)	Final document

2. About the Project and Company

Company

Worldcoin Foundation
Suite 3119, 9 Forum Lane, Camana Bay
PO Box 144, George Town
Grand Cayman KY1-9006
Cayman Islands

Website: https://worldcoin.org

Twitter: https://twitter.com/worldcoin

LinkedIn: https://www.linkedin.com/company/worldcoinfoundation

Discord: https://worldcoin.org/discord

Telegram: https://t.me/worldcoin

Youtube: https://www.youtube.com/@worldcoinofficial

2.1 Project Overview

In 2023, the company Tools For Humanity, unveiled Worldcoin, positioning it as a vanguard in the crypto-sphere with its "proof of personhood" model. Through its WorldID system, Worldcoin leverages the unique biometric intricacies of an individual's iris, cementing its commitment to differentiating genuine human participants from bots in the digital space.

The core tech underpinning this endeavor is the Orb, a device fine-tuned for high-fidelity iris scans. Once an iris is captured, it's translated into a unique IrisCode, subsequently anchored onto a decentralized blockchain for unparalleled security. In an age where data privacy is paramount, Worldcoin ensures that the actual iris images are purged post-conversion, retaining only the essential coded data.

But Worldcoin isn't just about ID verification. The World App extends its footprint into the DeFi sector, serving both as a digital wallet and a bridge to various DeFi platforms. Given the rampant Sybil attacks and fake accounts in the crypto world, World App's insistence on identity assurance is poised to revolutionize peer-to-peer interactions within the DeFi ecosystem.

On the cryptocurrency front, Worldcoin has rolled out its native WLD token, already gaining traction on major exchanges. This token is part of a long-term release strategy, spanning 15 years, introducing a gradual but sustained influx into the market.

In the midst of the platform's high-tech offerings, Worldcoin remains adamant about user privacy. While it authenticates user uniqueness, it's committed to a non-invasive approach, refraining from exploiting personal biometric markers.

Worldcoin's debut saw a meteoric surge in user registrations, reflecting the crypto community's eager anticipation. By synergizing digital identity with cryptocurrency, Worldcoin stands at the frontier of a more authenticated, human-driven digital crypto realm.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i. Review of the specifications, sources, and instructions provided to softstack to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to softstack describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

5. Metrics

The metrics section should give the reader an overview on the size, quality, flows and capabilities of the codebase, without the knowledge to understand the actual code.

5.1 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

Optimism
Source: https://optimistic.etherscan.io/token/0xdc6ff44d5d932cbd77b52e5612ba0529dc6226f1#code

File	Fingerprint (MD5)
./contracts/universal/CrossDomainMessenger.sol	48c467ab137dbb86308c4451fd31e713
./contracts/universal/ProxyAdmin.sol	01019e05fa55de2cb4ae42fa037df0c7
./contracts/universal/IOptimismMintableERC721.sol	80bde815cb984076442da4751c9affd0
./contracts/universal/OptimismMintableERC721Factory.sol	9a4b9c9002dd1b6241774b8ad7932919
./contracts/universal/IOptimismMintableERC20.sol	206a8caa03bb7107bd56908bf448bc18
./contracts/universal/FeeVault.sol	dc389a51582e7c5fe00e31434a08243c
./contracts/universal/OptimismMintableERC721.sol	9d90c6e2d464e82be121c1338df39cce
./contracts/universal/Proxy.sol	6c32f3b1a541dca89935d432674f8f54
./contracts/universal/ERC721Bridge.sol	6b91e5e39116c8f18b8edf49c40ac1fa
./contracts/universal/StandardBridge.sol	b13add962e0fbba8566435c78e2a6f9f
./contracts/universal/Semver.sol	e4ac282d17a29c5bb4e1cbe674dd85e4
./contracts/universal/OptimismMintableERC20.sol	37f58d0200e3f3667d955d24d9fd7add
./contracts/universal/OptimismMintableERC20Factory.sol	c63ea40866b62745d5ef02da7c8d1b66

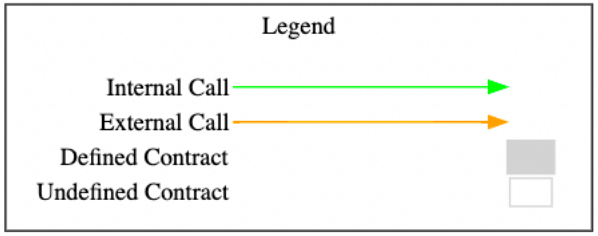
./contracts/L2/L2ToL1MessagePasser.sol	b5efc66fa782964d54c201fb4280b03e
./contracts/L2/L1FeeVault.sol	fc0878f2859e6733e0e8a5a5a3907b51
./contracts/L2/L1Block.sol	f823dd24866c0fa13f0c06c2f0bea126
./contracts/L2/CrossDomainOwnable.sol	038e310ad1062314ff08283b23e8f266
./contracts/L2/BaseFeeVault.sol	f18deef57f0b3fcd5b76be617630f3f0
./contracts/L2/L2CrossDomainMessenger.sol	f9e6c4844d681b33209372b7ddfbf277
./contracts/L2/GasPriceOracle.sol	36a62b0942804e6abfd304603bd1caee
./contracts/L2/SequencerFeeVault.sol	58e2bcf32af2d1fa2c5270bc5b7c83d6
./contracts/L2/CrossDomainOwnable2.sol	95e4a14321b5d4727c88b24ed99bd7d5
./contracts/L2/L2ERC721Bridge.sol	cfcdc1a4e051d995d2073a289b94d034
./contracts/L2/L2StandardBridge.sol	c255903ef1f7e43072a6f93dda4bd45d
./contracts/legacy/ResolvedDelegateProxy.sol	7387e99c938d354aa214d99d38abf4b7
./contracts/legacy/L1ChugSplashProxy.sol	0b304c5700a7b4649e7277a8fe50f850
./contracts/legacy/AddressManager.sol	7cd58b4fb4c4983ca5c421fd623c87b5
./contracts/legacy/LegacyMessagePasser.sol	9266e8bf736e60d6675c6371357bae2d
./contracts/legacy/DeployerWhitelist.sol	9db7ce952b76cbda897ad66cd70fd6cf
./contracts/legacy/LegacyERC20ETH.sol	7cfbd130861120aae230d4955a34362d
./contracts/legacy/L1BlockNumber.sol	0febdca95698ad7f11ec75e7ab337aaf
./contracts/libraries/Hashing.sol	be95795ee6673b0ea16165bc958afb2e
./contracts/libraries/Encoding.sol	a6c3292b192979ed9297f3221c8d2720
./contracts/libraries/Arithmetic.sol	e912ae2558a2066daa8c72180a9585a9
./contracts/libraries/SafeCall.sol	ca8aa1eb20aa58bf287da806f1ee201b
./contracts/libraries/rlp/RLPReader.sol	3503563d960787e9b2ea2f2cf15d6fd5
./contracts/libraries/rlp/RLPWriter.sol	4d7d36c5f6a97393fd4d62d9aa2a45d6
./contracts/libraries/Bytes.sol	75b16d2c1c705ddcb4f8455b9ded2f83
./contracts/libraries/trie/SecureMerkleTrie.sol	b0027c162452ab0b9e502d78e5bcad41
./contracts/libraries/trie/MerkleTrie.sol	0e1c3a7b4b54fc8a27dfd18b2b441640
./contracts/libraries/Constants.sol	b78aa03764ec4a6cf1704238f17413a8
./contracts/libraries/Predeploys.sol	f50e5d8dffb55cf19381fc5ffccc9f8c
./contracts/libraries/Burn.sol	9ebf66c29ae9ec7e048b87184c4236e1
./contracts/libraries/Types.sol	ffeeca5418798f845d0914ce71a7374b

./contracts/deployment/PortalSender.sol	06437bf67edb61a6ac90a574545f2d30
./contracts/deployment/SystemDictator.sol	8505d746bcea303a5238c3b1ca986c16
./contracts/L1/L2OutputOracle.sol	d483fa3559b883785fe8e5fee0e3e816
./contracts/L1/ResourceMetering.sol	5f0f9206accf098ee27fe735a46b7b8d
./contracts/L1/SystemConfig.sol	1682f8f993f8180de23bb64396c1ac08
./contracts/L1/L1ERC721Bridge.sol	b3f901f61813973501508430dafcdbe1
./contracts/L1/L1StandardBridge.sol	cd7116701f46b729fc3140db3cc00acf
./contracts/L1/OptimismPortal.sol	232e08670fde2da68ac42562962dd48a
./contracts/L1/L1CrossDomainMessenger.sol	d76e623d3ff06e67d2ee0e599755d23b
./contracts/vendor/AddressAliasHelper.sol	884a262b1fa2ff1daf3022c3a5ecf034

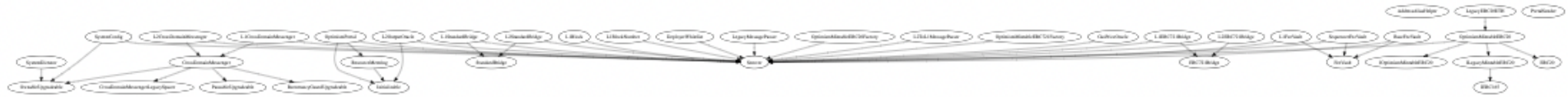
Ethereum
Source: https://etherscan.io/token/0x163f8c2467924be0ae7b5347228cabf260318753#code

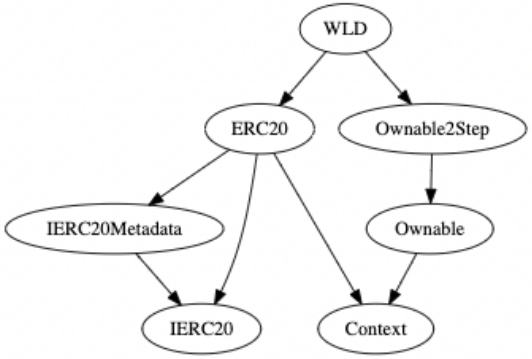
File	Fingerprint (MD5)
./contracts/src/WLD.sol	6d932bf87be058ba5f23af4238bfdd5d

5.2 CallGraph

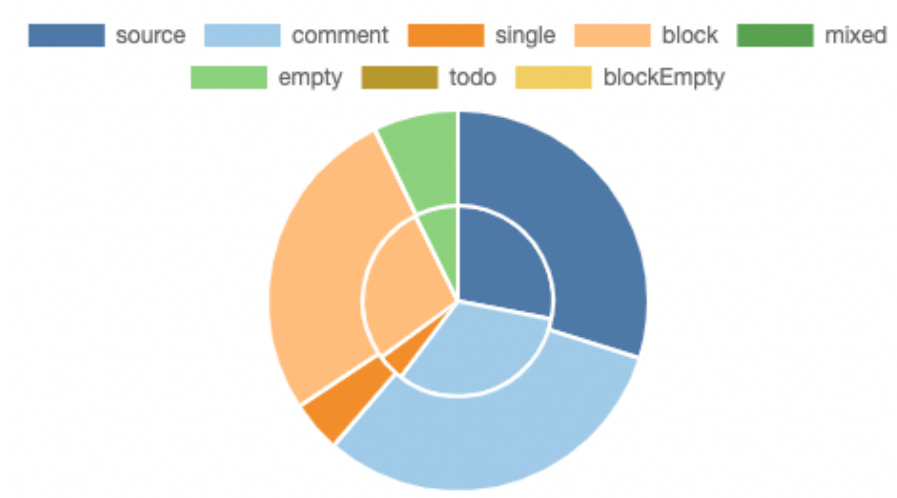


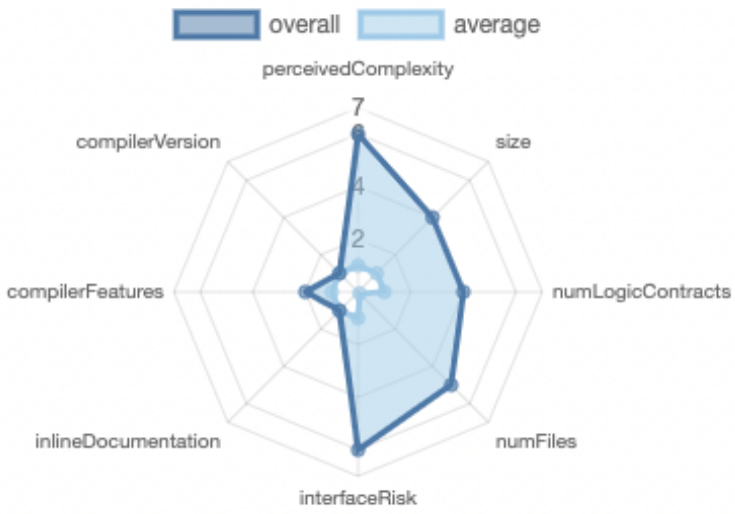
5.3 Inheritance Graph





5.4 Source Lines & Risk





5.5 Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts

`^0.8.19`
`0.8.15`
`^0.8.0`
`^0.8.9`
`^0.8.8`
		`yes`	`yes`
			(37 asm blocks)
				`yes`

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	✍️ ECRecover	🌀 New/Create/Create2
—	—	`yes`	`yes`	—	`yes` → `AssemblyCall:Name:create` → `NewContract:OptimismMintableERC20` → `NewContract:OptimismMintableERC721`

♻️ TryCatch	Σ Unchecked
`yes`	`yes`

Exposed Functions
This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
197	29

External	Internal	Private	Pure	View
129	252	11	52	90

StateVariables

Total	🌐Public
157	90

5.6 Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	mainnet/contracts/src/WLD.sol	1		231	228	103	88	64	
🎨	mainnet/contracts/lib/openzeppelin-contracts/contracts/utils/Context.sol	1		24	24	9	12	1	
🎨	mainnet/contracts/lib/openzeppelin-contracts/contracts/access/Ownable.sol	1		83	83	31	41	23	
🎨	mainnet/contracts/lib/openzeppelin-contracts/contracts/access/Ownable2Step.sol	1		57	57	22	27	18	

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	mainnet/contracts/lib/openzeppelin-contracts/contracts/token/ERC20/extensions/IERC20Metadata.sol		1	28	17	4	16	9	🔆
🎨	Optimism/contracts/L1/ResourceMetering.sol	1		168	168	69	77	44	
📝	Optimism/contracts/L1/SystemConfig.sol	1		202	195	77	96	61	🖥️🧱
📝	Optimism/contracts/L1/L2OutputOracle.sol	1		331	315	134	144	76	🖥️💰
📚	Optimism/contracts/vendor/AddressAliasHelper.sol	1		43	43	14	24	8	Σ
📝	Optimism/contracts/L1/L1CrossDomainMessenger.sol	1		70	65	28	30	20	
📝	Optimism/contracts/L1/OptimismPortal.sol	1		418	407	170	190	90	💰🧱
📝	Optimism/contracts/L1/L1StandardBridge.sol	1		293	251	85	151	48	💰
📝	Optimism/contracts/L1/L1ERC721Bridge.sol	1		107	92	38	42	26	
📝	mainnet/contracts/lib/openzeppelin-contracts/contracts/token/ERC20/ERC20.sol	1		320	320	112	180	77	Σ

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	Optimism/contracts/deployment/SystemDictator.sol	1		426	424	244	129	137	🧱♻️
📝	Optimism/contracts/deployment/PortalSender.sol	1		30	30	11	15	6	
📚	Optimism/contracts/libraries/Types.sol	1		84	84	33	47	1	
📝📚	Optimism/contracts/libraries/Burn.sol	2		42	42	18	21	22	💰💣
📚	Optimism/contracts/libraries/Predeploys.sol	1		107	107	23	67	18	
📚	Optimism/contracts/libraries/Constants.sol	1		27	27	5	20	5	
📝	Optimism/contracts/legacy/L1BlockNumber.sol	1		55	55	24	26	36	🖥️💰
📝	Optimism/contracts/legacy/LegacyERC20ETH.sol	1		97	93	32	51	29	
📝	Optimism/contracts/legacy/DeployerWhitelist.sol	1		115	115	36	66	25	
📝	Optimism/contracts/legacy/LegacyMessagePasser.sol	1		33	33	9	20	9	🧱
📝	Optimism/contracts/legacy/AddressManager.sol	1		64	64	18	39	14	🧱

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝🔍	Optimism/contracts/legacy/L1ChugSplashProxy.sol	1	1	289	273	110	143	147	🖥️💰👥🧱🌀🔆
📝	Optimism/contracts/legacy/ResolvedDelegateProxy.sol	1		64	64	26	30	41	🖥️💰👥
📚	Optimism/contracts/libraries/Bytes.sol	1		142	138	59	63	131	🖥️🧱Σ
📚	Optimism/contracts/libraries/trie/MerkleTrie.sol	1		288	275	130	117	85	🧱Σ
📚	Optimism/contracts/libraries/rlp/RLPWriter.sol	1		221	217	110	81	179	🖥️Σ
📚	Optimism/contracts/libraries/rlp/RLPReader.sol	1		359	347	188	107	204	🖥️
📚	Optimism/contracts/libraries/trie/SecureMerkleTrie.sol	1		64	55	15	36	7	🧱
📚	Optimism/contracts/libraries/SafeCall.sol	1		37	32	18	20	19	🖥️
📚	Optimism/contracts/libraries/Arithmetic.sol	1		48	40	13	24	4	
📚	Optimism/contracts/libraries/Encoding.sol	1		162	139	67	65	56	🖥️
📚	Optimism/contracts/libraries/Hashing.sol	1		172	137	55	74	22	🧱

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	mainnet/contracts/lib/openzeppelin-contracts/contracts/token/ERC20/IERC20.sol		1	78	38	16	58	13	🔆
🎨	Optimism/contracts/L2/CrossDomainOwnable.sol	1		25	25	11	12	6	
📝	Optimism/contracts/L2/BaseFeeVault.sol	1		20	20	6	12	7	
📝	Optimism/contracts/L2/L2CrossDomainMessenger.sol	1		75	70	31	32	20	
📝	Optimism/contracts/universal/OptimismMintableERC20.sol	1		114	104	50	40	37	
📝	Optimism/contracts/universal/Semver.sol	1		58	58	28	24	11	
📝	Optimism/contracts/universal/OptimismMintableERC20Factory.sol	1		106	98	30	57	26	🌀
📝	Optimism/contracts/L2/GasPriceOracle.sol	1		130	130	47	72	37	
🎨	Optimism/contracts/universal/StandardBridge.sol	1		467	420	173	210	97	💰
🎨	Optimism/contracts/universal/ERC721Bridge.sol	1		214	192	67	109	26	
📝	Optimism/contracts/L2/L1Block.sol	1		103	94	28	53	16	

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	Optimism/contracts/universal/Proxy.sol	1		216	211	83	105	111	🖥️💰👥
📝	Optimism/contracts/L2/L1FeeVault.sol	1		20	20	6	12	7	
📝	Optimism/contracts/universal/OptimismMintableERC721.sol	1		153	148	72	58	42	
📝	Optimism/contracts/L2/L2ToL1MessagePasser.sol	1		141	137	59	63	33	💰Σ
🎨	Optimism/contracts/universal/FeeVault.sol	1		71	71	28	32	19	💰
🔍	Optimism/contracts/universal/IOptimismMintableERC20.sol		2	34	14	5	13	18	
📝	Optimism/contracts/universal/OptimismMintableERC721Factory.sol	1		77	73	29	34	24	🌀
🔍	Optimism/contracts/universal/IOptimismMintableERC721.sol		1	76	37	8	49	19	
📝	Optimism/contracts/L2/SequencerFeeVault.sol	1		31	31	9	19	9	
📝🔍	Optimism/contracts/universal/ProxyAdmin.sol	1	2	254	230	101	121	98	💰🔆
📝🎨	Optimism/contracts/universal/CrossDomainMessenger.sol	2		447	408	163	219	92	💰Σ

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	Optimism/contracts/L2/L2StandardBridge.sol	1		179	154	53	90	40	💰
📝	Optimism/contracts/L2/L2ERC721Bridge.sol	1		126	111	49	48	34	
🎨	Optimism/contracts/L2/CrossDomainOwnable2.sol	1		36	36	19	13	9	
📝📚🔍🎨	**Totals**	**59**	**8**	**8552**	**7986**	**3311**	**3934**	**2613**	🖥️💰💣👥🧱🌀🔆♻️Σ

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

6. Scope of Work

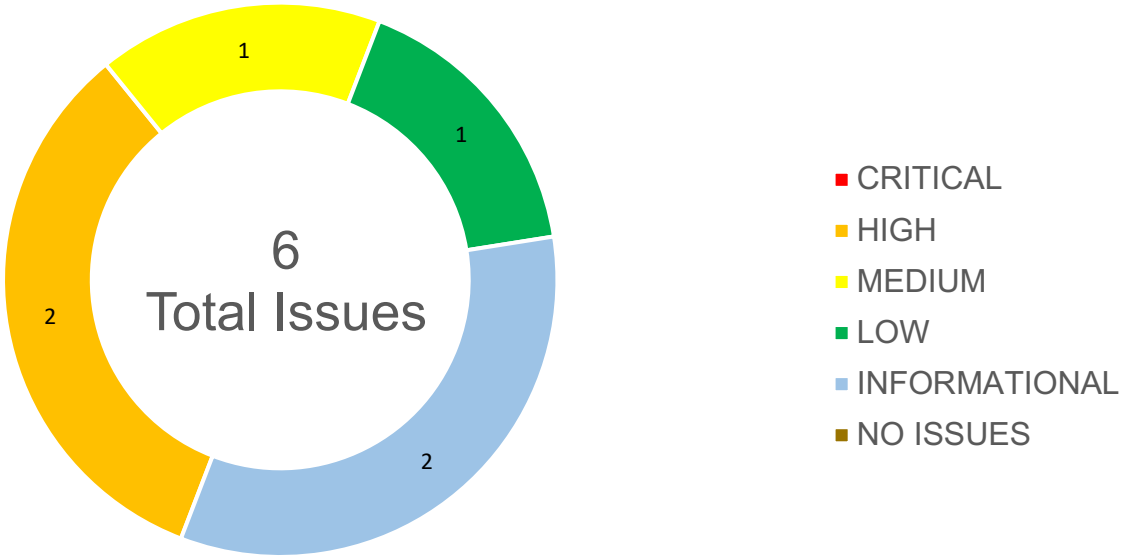
The Worldcoin Team provided us with the files that needs to be tested. The scope of the audit are the Optimism WLD Token Bridge (L2) and Ethereum WLD Token (L1) contracts

The team put forward the following assumptions regarding the security, usage of the contracts:

1. The security audit should meticulously examine the smart contract to ensure there are no vulnerabilities, potential exploits, or security flaws.

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.

6.1 Findings Overview



No	Title	Severity	Status
6.2.1	Potential Loss Of User's Funds On Optimism Due To Gas Limits	MEDIUM	ACKNOWLEDGED
6.2.2	Permanent Locking Of User's Withdrawals On Optimism	MEDIUM	ACKNOWLEDGED
6.2.3	Permanent Locking Of User's Tokens On Optimism	MEDIUM	ACKNOWLEDGED
6.2.4	Missing Error Messages	MEDIUM	ACKNOWLEDGED
6.2.5	Floating Pragma	LOW	ACKNOWLEDGED
6.2.6	Redundant Boolean Equality Check	INFORMATIONAL	ACKNOWLEDGED

6.2 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, softstack's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, softstack's experts found **2 High issues** in the code of the smart contract.

6.2.1 Potential Loss Of User's Funds On Optimism Due To Gas Limits

Severity: HIGH
Status: ACKNOWLEDGED
Code: NA
File(s) affected: OptimismPortal.sol

Attack / Description	Due to imprecise gas amount checks, malicious actors can lock user's funds in the Optimism Portal by setting a specific gas amount on calling *finalizeWithdrawalTransaction*. Because the call does not revert on failure and the withdrawal is marked as processed, the user cannot access it after calling the function within a gas range of 5122 gas.

Code	Line 313 – 329 (OptimismPortal.sol)
	
	`require(`
	`gasleft() >= _tx.gasLimit + FINALIZE_GAS_BUFFER,`
	`"OptimismPortal: insufficient gas to finalize withdrawal"`
	`);`
	
	`// Set the l2Sender so contracts know who triggered this withdrawal on L2.`
	`l2Sender = _tx.sender;`

	`// Trigger the call to the target contract. We use SafeCall because we don't`
	`// care about the returndata and we don't want target contracts to be able to force this`
	`// call to run out of gas via a returndata bomb.`
	`bool success = SafeCall.call(`
	`  _tx.target,`
	`  gasleft() - FINALIZE_GAS_BUFFER,`
	`  _tx.value,`
	`  _tx.data`
	`);`

Result/Recommendation	It is recommended to use two different values for calculating gas consumption within the function call to avoid the difference of 5122 gas. The values should be tracked before the require check and right before the function call. Ideally, the newest version of the Optimism bedrock smart contracts should be used where this issue has been fixed.

6.2.2 Permanent Locking Of User's Withdrawals On Optimism

Severity: HIGH
Status: ACKNOWLEDGED
Code: NA
File(s) affected: CrossDomainMessenger.sol

Attack / Description	An attacker can take advantage of the reentrancy guard protecting the *relayMessage* function to lock user's funds permanently. The attacker can create a contract, which will be called by *relayMessage*, calling *OptimismPortal.finalizeWithdrawalTransaction*, which marks the transaction as successful, before recalling the *relayMessage,* which will fail due to the reentrancy guard. As a result, the funds will be locked.
Code	Line 256 - 268 (CrossDomainMessenger.sol) `function relayMessage(`

	`uint256 _nonce,`
	`address _sender,`
	`address _target,`
	`uint256 _value,`
	`uint256 _minGasLimit,`
	`bytes calldata _message`
	`) external payable nonReentrant whenNotPaused {`
	`(, uint16 version) = Encoding.decodeVersionedNonce(_nonce);`
	`require(`
	`version < 2,`
	`"CrossDomainMessenger: only version 0 or 1 messages are supported at this time"`
	`);`

Result/Recommendation	It is recommended to revert the function call if an error is thrown by the reentrancy guard. Ideally, the newest version of the Optimism bedrock smart contracts should be used where this issue has been fixed.

MEDIUM ISSUES

During the audit, softstack's experts found **1 Medium issue** in the code of the smart contract.

6.2.3 Permanent Locking Of User's Tokens On Optimism

Severity: MEDIUM
Status: ACKNOWLEDGED
Code: NA
File(s) affected: CrossDomainMessenger.sol

Attack / Description	On sending WLD token from Ethereum to Optimism, funds will be deposited in *L1StandardBridge* on Ethereum and received by *L2StandardBridge* on Optimism. The funds are relayed by

	L2CrossDomainMessenger and if it is paused, they can't be relayed and will never reach the target. The funds will be stuck in the *L2CrossDomainMessenger* resulting in loss of funds.

Code	Line 256 - 268 (CrossDomainMessenger.sol)
	
	`function relayMessage(`
	`  uint256 _nonce,`
	`  address _sender,`
	`  address _target,`
	`  uint256 _value,`
	`  uint256 _minGasLimit,`
	`  bytes calldata _message`
	`  ) external payable nonReentrant whenNotPaused {`
	`  (, uint16 version) = Encoding.decodeVersionedNonce(_nonce);`
	`  require(`
	`  version < 2,`
	`  "CrossDomainMessenger: only version 0 or 1 messages are supported at this time"`
	`  );`

Result/Recommendation	It is recommended to remove the *whenNotPaused* modifier from *relayMessage* function or replay messages later on if the contract is paused at the time of *relayMessage* call. Ideally, the newest version of the Optimism bedrock smart contracts should be used where this issue has been fixed.

LOW ISSUES

During the audit, softstack's experts found **1 Low issue** in the code of the smart contract

6.2.4 Missing Error Messages

Severity: LOW

Status: ACKNOWLEDGED
Code: NA
File(s) affected: WLD.sol

Attack / Description	In the current implementation of the Worldcoin token contract, some functions hold several require checks without any error message on failure. This leads to undefined errors on transaction failures leaving the caller without any feedback why the transaction failed.
Code	Line 92: require(existingAmounts.length == existingHolders.length); Line 93: require(inflationCapPeriod_ != 0); Line 114: require(totalSupply() <= INITIAL_SUPPLY_CAP); Line 139: require(initialMintDone == false); Line 142: require(newHolders.length == newAmounts.length); Line 153: require(totalSupply() <= INITIAL_SUPPLY_CAP); Line 202: require(to != address(0)); Line 203: require(amount != 0); Line 206: require(msg.sender == minter); Line 209: require(block.timestamp >= inflationUnlockTime); Line 226: require(totalSupply() <= currentPeriodSupplyCap);
Result/Recommendation	It is recommended to use error messages in require statements to indicate why a transaction fails. It improves the user experience significantly by knowing what the caller may have done wrong causing a transaction failure.

INFORMATIONAL ISSUES

During the audit, softstack's experts found **2 Informational issues** in the code of the smart contract.

6.2.5 Floating Pragma

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: SWC-103
File(s) affected: WLD.sol

Attack / Description	The current pragma Solidity directive is ^0.8.19; 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.
Code	Line 2 (WLD.sol) pragma solidity ^0.8.19;
Result/Recommendation	It is recommended to follow the example (0.8.19), as future compiler versions may handle certain language constructions in a way the developer did not foresee. Not effecting the overall contract functionality.

6.2.6 Redundant Boolean Equality Check

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: NA
File(s) affected: WLD.sol

Attack / Description	In the current implementation of the *mintOnce* function, the boolean variable *initialMintDone* is checked for equality with a boolean constant. This equality check is redundant and consumes gas unnecessarily.

Code	Line 139 (WLD.sol) `require(initialMintDone == false);`
Result/Recommendation	It is recommended to remove the check for equality and read the constant value directly to save gas: !*initialMintDone.*

6.3 SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✅
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✅
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✅
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✅
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✅
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✅

ID	Title	Relationships	Test Result
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✅
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✅
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✅
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✅
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✅
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✅
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅

ID	Title	Relationships	Test Result
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✅
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✅
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✅
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✅
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✅
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✅
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✅
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✅

ID	Title	Relationships	Test Result
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✅
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✅
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	X
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✅
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✅
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅

7. Executive Summary

Two independent softstack experts performed an unbiased and isolated audit of the smart contract codebase provided by the Worldcoin Team. The main objective of the audit was to verify the security and functionality claims of the smart contract. The audit process involved a thorough manual code review and automated security testing.

Overall, the audit identified a total of 6 issues, classified as follows:
- No critical issues were found.
- Two high severity issues were found.
- One medium severity issues were found.
- One low severity issues were discovered
- Two informational issues were identified

The audit report provides detailed descriptions of each identified issue, including severity levels, CWE classifications, and recommendations for mitigation. We advise the Worldcoin team to implement the recommendations to further enhance the code's security and readability.

8. About the Auditor

Established in 2017 under the name Chainsulting, and rebranded as softstack GmbH in 2023, softstack has been a trusted name in Web3 Security space. Within the rapidly growing Web3 industry, softstack provides a comprehensive range of offerings that include software development, security, and consulting services. Softstack's competency extends across the security landscape of prominent blockchains like Solana, Tezos, Ethereum and Polygon. The company is widely recognized for conducting thorough code audits aimed at mitigating risk and promoting transparency.

The firm's proficiency lies particularly in assessing and fortifying smart contracts of leading DeFi projects, a testament to their commitment to maintaining the integrity of these innovative financial platforms. To date, softstack plays a crucial role in safeguarding over $100 billion worth of user funds in various DeFi protocols.

Underpinned by a team of industry veterans possessing robust technical knowledge in the Web3 domain, softstack offers industry-leading smart contract audit services. Committed to evolving with their clients' ever-changing business needs, softstack's approach is as dynamic and innovative as the industry it serves.

Check our website for further information: https://chainsulting.de