SOLIDProof

Blockchain Security | Smart Contract Audits

MADE IN GERMANY

Audit Passed

Security Assessment
14. June, 2021

For

WORLD

Disclaimer 3
Description 5
Project Engagement 5
Logo 5
Contract Link 5
Methodology 7
Used Code from other Frameworks/Smart Contracts (direct imports) 8
Source Lines 9
Risk Level 9
Capabilities 10
CallGraph 11
Source Units in Scope 11
Critical issue 13
no critical issues found 13
High issues 13
no high issues found 13
Medium issues 13
Low issues 14
Informational issues 16
SWC Attacks 17

Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing 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. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Overview

Network
Ethereum (ERC20)

Website
https://worldtoken.network/

Telegram
https://t.me/worldtokenofficial

Twitter
https://twitter.com/worldtoken_

Github
https://github.com/worldtoken/WORLD

Medium
https://worldtoken.medium.com/

Email
worldtoken.crypto@protonmail.com

Description

WORLD is a unique platform that combines the best tokenomics of currrent frictionless yield protocols for instant rewards with the additional benefits of staking in our upcoming marketplace. This way the best rewards can be guaranteed without any token inflation.

A 3% transaction tax goes to holders (later on merchants too), stakers, and a perpetual marketing and development fund. This project is built to keep going and continually expand further until it has its own ecosystem to call its own.

The $WORLD system guarantees token rewards to LP stakers on every block, regardless if there was a $WORLD transaction on it or not. Under the same system, rewards will scale as the project grows, whilst ensuring the rewards pool can never run out

Project Engagement

During the 8th of June, **Worldtoken Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. **Worldtoken Team** provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

https://etherscan.io/address/0xBF494F02EE3FdE1F20BEE6242bCe2d1ED0c15e47#code

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 aspossible.
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

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.

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 SolidProof 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 SolidProof 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.

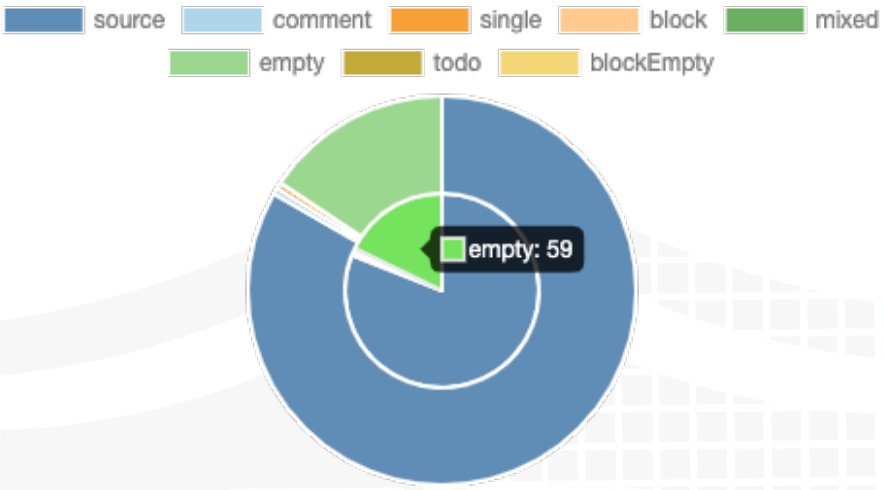
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

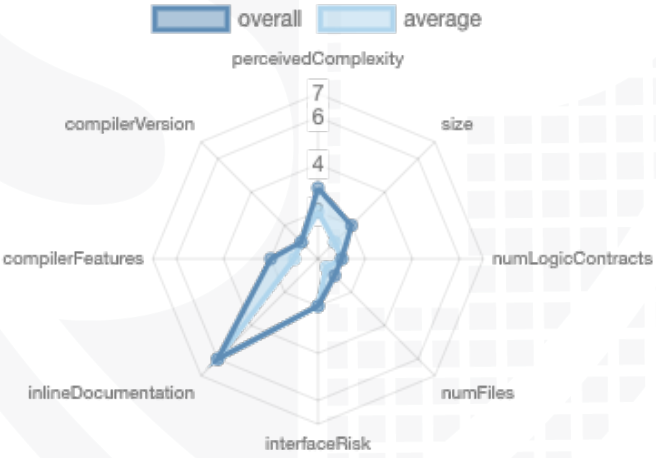
Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	1
@openzeppelin/contracts/token/ERC1155/IERC1155.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	1
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	1
@openzeppelin/contracts/token/ERC721/IERC721.sol	1
@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol	

Metrics

Source Lines



Risk Level

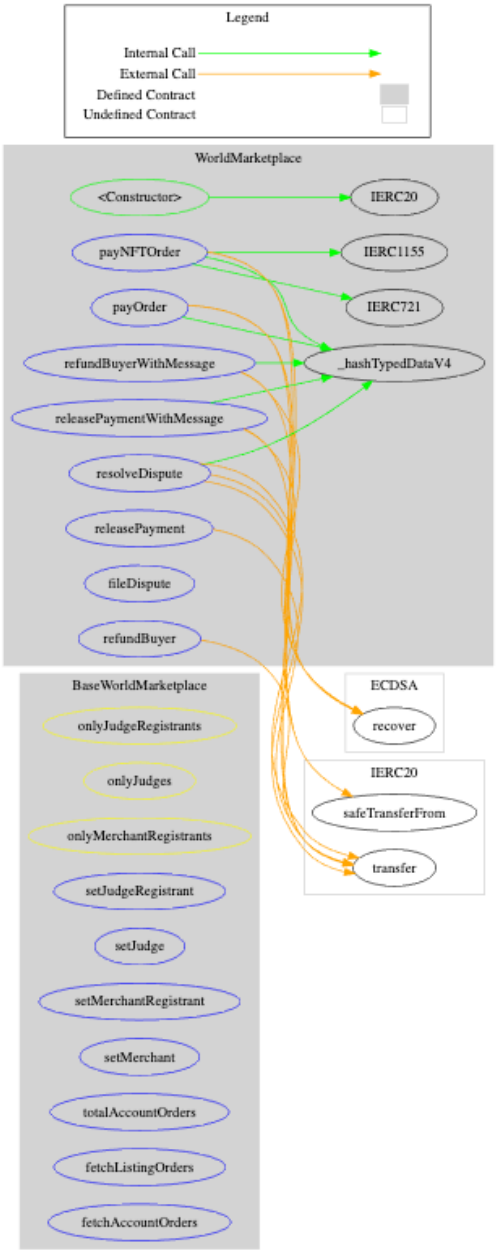


Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
0.8.0			**** (0 asm blocks)	

📩 Transfers ETH	⚡ Low-Level Calls	👥 Delegate Call	🧱 Uses Hash Functions	🏷 ECRecover	🌀 New/ Create/ Create2
Yes			Yes		

CallGraph



Source Units in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/BaseWorldMarketplace.sol	1	———	111	103	80	1	65	———
📝	contracts/WorldMarketplace.sol	1	———	264	226	188	1	114	📩🧱
📝	**Totals**	**2**	———	**375**	**329**	**268**	**2**	**179**	📩🧱

Contract Summary

File Name	SHA-1 Hash
contracts/BaseWorldMarketplace.sol	d4c9f563e1af1c108d8d4a0f90348ab2406d4b92
contracts/WorldMarketplace.sol	c61a67273cc69f1a1aad64f3b8df81c42d1cbe47

Contract	Type	Bases		
└	Function Name	Visibility	Mutability	Modifiers
BaseWorldMarketplace	Implementation	Ownable		
└	setJudgeRegistrant	External ❗	🛑	onlyOwner
└	setJudge	External ❗	🛑	onlyJudgeRegistrants
└	setMerchantRegistrant	External ❗	🛑	onlyOwner
└	setMerchant	External ❗	🛑	onlyMerchantRegistrants
└	totalAccountOrders	External ❗		NO ❗
└	fetchListingOrders	External ❗		NO ❗
└	fetchAccountOrders	External ❗		NO ❗
WorldMarketplace	Implementation	BaseWorldMarketplace, EIP712		
└		Public ❗	🛑	EIP712
└	payNFTOrder	External ❗	🛑	NO ❗
└	payOrder	External ❗	🛑	NO ❗
└	releasePayment	External ❗	🛑	NO ❗
└	releasePaymentWithMessage	External ❗	🛑	NO ❗
└	refundBuyer	External ❗	🛑	NO ❗
└	refundBuyerWithMessage	External ❗	🛑	NO ❗
└	fileDispute	External ❗	🛑	NO ❗
└	resolveDispute	External ❗	🛑	NO ❗

Legend:

🛑: Function can modify state

Audit Results

AUDIT PASSED

Critical issue

no critical issues found

High issues

no high issues found

Medium issues

Issue	File	Type	Line	Description
#1	WorldMarketplace.sol	Reentrancy vulnerabilities (no theft of ethers) (reentrancy-no-eth)	86-129	Reentrancy in WorldMarketplace.payOrder(string,uint256,uint256,address,uint256,uint256,uint256,bytes) (WorldMarketplace.sol:86-129): • WORLD.safeTransferFrom(msg.sender,address(this),_total (Line: 125) • order.indisputableTime = block.timestamp + _disputeWindow (Line: 126)

Low issues

Issue	File	Type	Line	Description
#1	WorldMarketplace.sol	Unchecked tokens transfer (unchecked-transfer)	181-191	WorldMarketplace.refundBuyer(uint256,uint256) ignores return value by WORLD.transfer(order.buyer,order.total)
#2	WorldMarketplace.sol	Unchecked tokens transfer (unchecked-transfer)	193-218	WorldMarketplace.refundBuyerWithMessage(string,uint256,uint256,bytes) ignores return value by WORLD.transfer(order.buyer,order.total) (Line: 217)
#3	WorldMarketplace.sol	Unchecked tokens transfer (unchecked-transfer)	131-152	WorldMarketplace.releasePayment(uint256,uint256) ignores return value by WORLD.transfer(order.seller,order.total) (Line: 151)
#4	WorldMarketplace.sol	Unchecked tokens transfer (unchecked-transfer)	154-179	WorldMarketplace.releasePaymentWithMessage(string,uint256,uint256,bytes) ignores return value by WORLD.transfer(order.seller,order.total) (Line: 178)
#5	WorldMarketplace.sol	Unchecked tokens transfer (unchecked-transfer)	229-263	WorldMarketplace.resolveDispute(string,uint256,uint256,bool,bytes) ignores return value by WORLD.transfer(order.buyer,order.total) (Line: 258)
#6	WorldMarketplace.sol	Unchecked tokens transfer (unchecked-transfer)	229-263	WorldMarketplace.resolveDispute(string,uint256,uint256,bool,bytes) ignores return value by WORLD.transfer(order.seller,order.total) (Line: 261)

#7	WorldMarketplace.sol	Dangerous usage of `block.timestamp` (timestamp)	220-227	WorldMarketplace.fileDispute(uint256,uint256) uses timestamp for comparisons • require(bool,string) (block.timestamp <= order.indisputableTime,Order is already indisputable) (Line: 224)
#8	WorldMarketplace.sol	Dangerous usage of `block.timestamp` (timestamp)	31-84	WorldMarketplace.payNFTOrder(string,uint256,uint256,address,uint256,address,uint256,uint256,bytes,bool,uint256,bytes) uses timestamp for comparisons • require(bool,string) (block.timestamp <= _paymentDueTime,Order can't be paid anymore as it's passed due) (Line: 46)
#9	WorldMarketplace.sol	Dangerous usage of `block.timestamp` (timestamp)	86-129	WorldMarketplace.payOrder(string,uint256,uint256,address,uint256,uint256,uint256,bytes) uses timestamp for comparisons • require(bool,string) (block.timestamp <= _paymentDueTime,Order can't be paid anymore as it's past due) (Line: 97) • require(bool,string) (order.status == OrderStatus.UNDEFINED,Order is already paid) (Line: 115)

#10	WorldMarketplace.sol	Dangerous usage of `block.timestamp` (timestamp)	131-152	WorldMarketplace.releasePayment(uint256,uint256) uses timestamp for comparisons • block.timestamp <= order.indisputableTime || order.status == OrderStatus.IN_DISPUTE (Line: 138)

Informational issues

Issue	File	Type	Line	Description
#1	WorldMarketplace.sol	Conformity to Solidity naming conventions (naming-convention)	77	Parameter BaseWorldMarketplace.fetchListingOrders(uint256[],uint256[])._listingIds is not in mixedCase
#2	WorldMarketplace.sol	Conformity to Solidity naming conventions (naming-convention)	77	Parameter BaseWorldMarketplace.fetchListingOrders(uint256[],uint256[])._offerIds is not in mixedCase
#3	WorldMarketplace.sol	Conformity to Solidity naming conventions (naming-convention)	61	Parameter BaseWorldMarketplace.setJudge(address,bool)._account is not in mixedCase
#4	WorldMarketplace.sol	Conformity to Solidity naming conventions (naming-convention)	156	Parameter WorldMarketplace.releasePaymentWithMessage(string,uint256,uint256,bytes)._listingId is not in mixedCase

SWC Attacks

ID	Title	Relationships	Status
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED

SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED

SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED

SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**

Blockchain Security | Smart Contract Audits

MADE IN GERMANY