SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

DopewarZ

Audit

Security Assessment

06.September,2022

For

Disclaimer

Version	Date	Description
1.0	05.September,2022	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	06. September,2022	• Reaudit

Network

Binance (BSC)

Website
http://dopewarz.io/

Twitter
https://twitter.com/DopeWarz_ofcl

Telegram
https://t.me/dopewarzann

Instagram
https://www.instagram.com/dopewarz/?hl=en

Discord
https://discord.com/invite/dopewarz

Blog
https://blog.dopewarz.io/

Description

The game takes inspiration from the original **DopeWars (1998)**. Where through buying and selling drugs you make use of the market arbitrage present in different districts, to maximize profits. Essentially, buying DrugZ at a low price and selling them at a higher price.
DopeWarZ.io, will build upon and take inspiration from the original 1990s DopeWars gameplay but adds an additional layer of an MMO environment and the blockchain.
As you go about dealing drugs, there are various factors that impact your profits, such as law enforcement (probability of getting caught) and supply/ demand of different drugs.

Project Engagement

During the 05th of September 2022, **DopewarZ** team engaged Solidproof.io to audit the smart contracts that they created. The engagement was technical in nature and focused on identifying the security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Links
v1.0
   Provided as files

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

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 analyzing 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)

Imported packages:

```
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol
@openzeppelin/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts/utils/math/SafeMath.sol
```

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

File Name	SHA-1 Hash
contracts/Staking.sol	8ccd59d303a5893c50fcb4e25e7efafeb4e83e64

Metrics

Source Lines
v1.0

Risk Level
v1.0
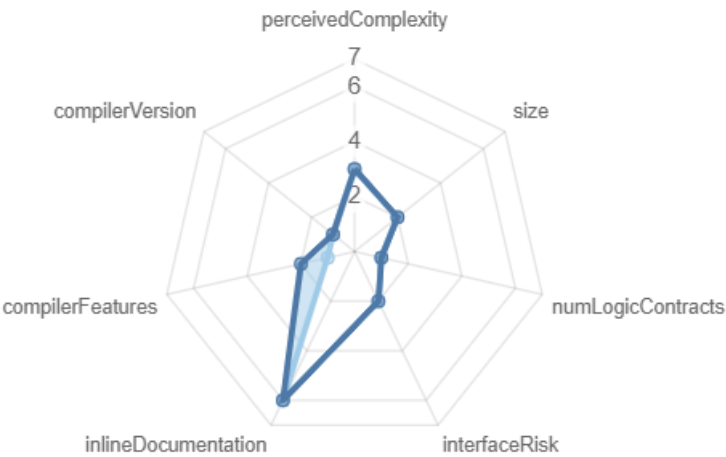
Capabilities

v1.0

Components

📝 Contracts	📚 Libraries	🔍 Interfaces	🎨 Abstract
1	0	0	0

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
11	0

External	Internal	Private	Pure	View
4	11	0	0	1

StateVariables

Total	🌐 Public
17	17

Capabilities

Solidity Versions observed	🎨 Experimental Features	💰 Can Receive Funds	🗄 Uses Assembly	🔮 Has Destroyable Contracts
^0.8.0				

🛢 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🗂 Uses Hash Functions	⚡ ECRecover	🌀 New/Create/Create2

♻️ TryCatch	Σ Unchecked

Inheritance Graph
v1.0
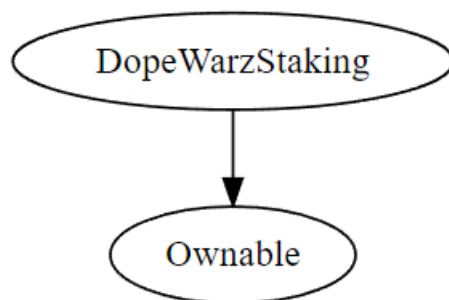
Call Graph
v1.0
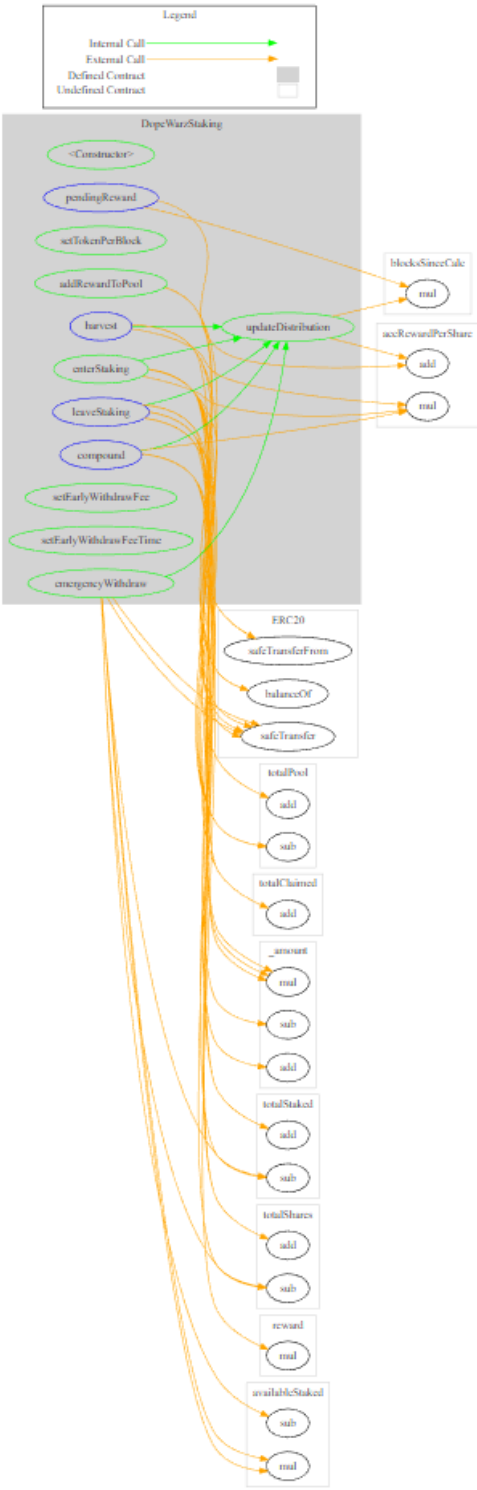
Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:
1. Is contract an upgradeable
2. Correct implementation of Token standard
3. Deployer cannot mint any new tokens
4. Deployer cannot burn or lock user funds
5. Deployer cannot pause the contract
6. Deployer can set fees
7. Deployer can blacklist/antisnipe address
8. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	**No**

Correct implementation of Token standard

ERC20				16
Function	**Description**	**Exist**	**Tested**	**Verified**
totalSupply	Provides information about the total token supply			
balanceOf	Provides account balance of the owner's account			
transfer	Executes transfers of a specified number of tokens to a specified address			
transferFrom	Executes transfers of a specified number of tokens from a specified address			
approve	Allow a spender to withdraw a set number of tokens from a specified account			
allowance	Returns a set number of tokens from a spender to the owner			

Write functions of contracts
v1.0

```
♦ addRewardToPool
♦ updateDistribution
♦ setTokenPerBlock
♦ enterStaking
♦ leaveStaking
♦ harvest
♦ compound
♦ setEarlyWithdrawFee
♦ setEarlyWithdrawFeeTime
♦ emergencyWithdraw
```

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint			
Max / Total Supply	N/A		

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock			
Deployer cannot burn			

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause			

Deployer can set fees

Name	Exist	Tested	Status
Deployer can set fees over 25%			
Deployer can set fees to nearly 100% or more			

Comments:

- The fees can be set to a maximum of 10%

Deployer cannot blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer can blacklist/antisnipe addresses			

Overall checkup (Smart Contract Security)
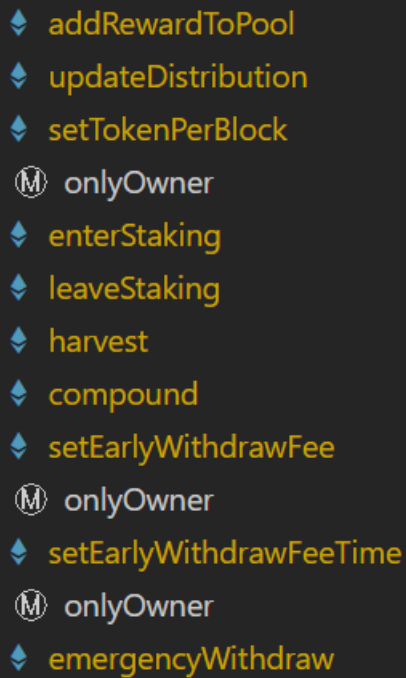
Tested	Verified

Legend

Attribute	Symbol
Verified / Checked	
Partly Verified	
Unverified / Not checked	
Not available	

Modifiers and public functions

- ◆ addRewardToPool
- ◆ updateDistribution
- ◆ setTokenPerBlock
- Ⓜ onlyOwner
- ◆ enterStaking
- ◆ leaveStaking
- ◆ harvest
- ◆ compound
- ◆ setEarlyWithdrawFee
- Ⓜ onlyOwner
- ◆ setEarlyWithdrawFeeTime
- Ⓜ onlyOwner
- ◆ emergencyWithdraw

Comments:

- The owner can set token per block which will be used in calculating rewards but not more than 10
- The owner can set early withdraw fee time but not to zero. Although, it can be set to any value greater than zero and up to max value of uint.

Source Units in Scope

v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/Staking.sol	1	————	289	289	210	41	167
Totals	**1**	————	**289**	**289**	**210**	**41**	**167**

Legend

Attribute	Description
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, ...)

Audit Results

<div style="background-color:#66ff33; color:green; text-align:center; font-size:3em;">

AUDIT PASSED

</div>

Critical issues

<div style="background-color:#66ff33; color:green; text-align:center;">No critical issues</div>

High issues

<div style="background-color:#66ff33; color:green; text-align:center;">No high issues</div>

Medium issues

<div style="background-color:#66ff33; color:green; text-align:center;">No medium issues</div>

Low issues

| Issue	File	Type	Line	Description
#1	Main	Missing Events	116,160,194, 214,242,282	Emit an event for critical parameter changes. There are no events in any write function of the contract.

Informational issues

Issue	File	Type	Line	Description
#1	All	NatSpec documentation missing	–	If you started to comment your code, also comment all other functions, variables etc.

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information https://docs.soliditylang.org/en/v0.5.10/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

06. September, 2022:

- There is still an owner (Owner still has not renounced ownership)
- Read the whole report and modifiers section for more information.
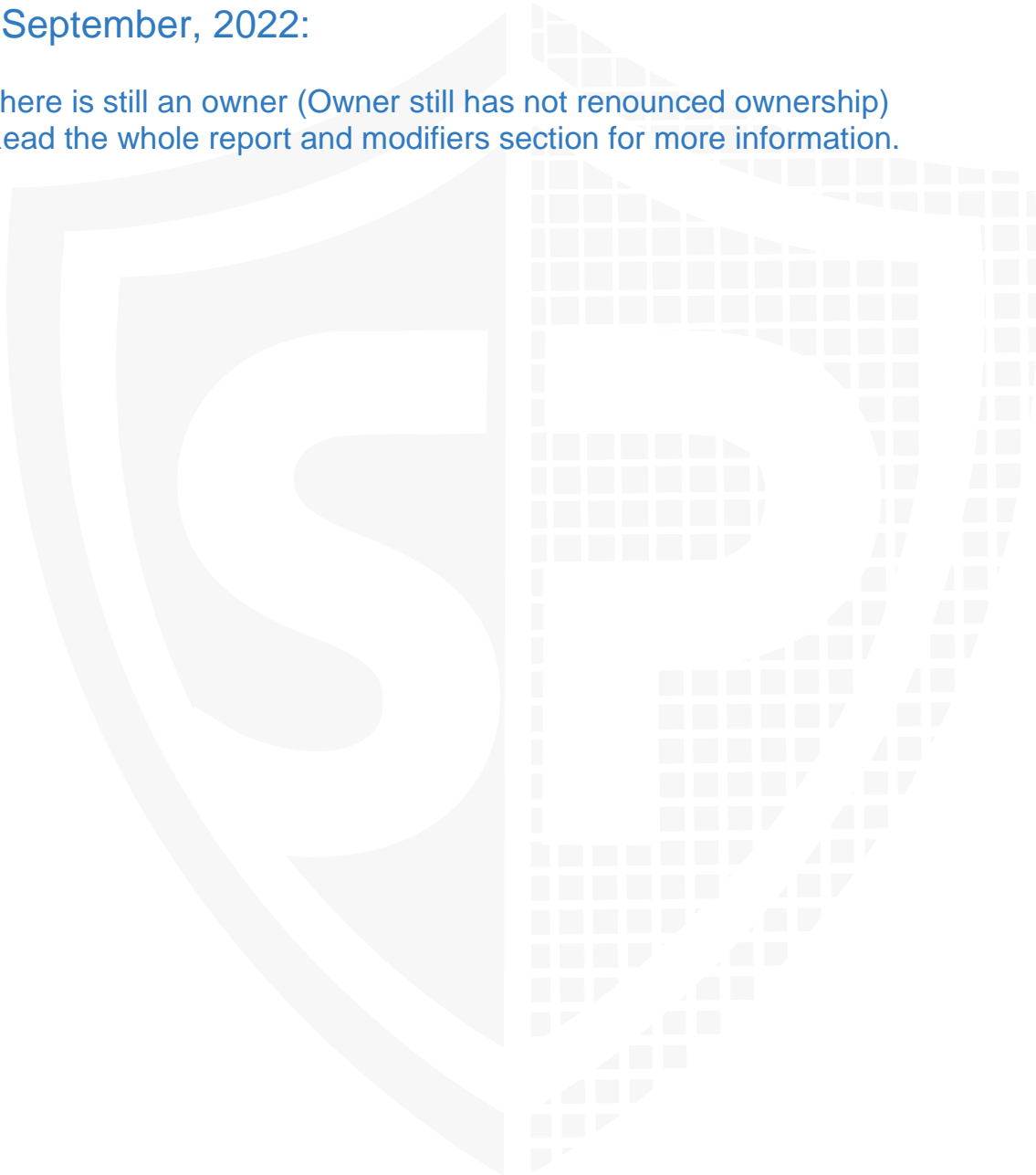
SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	**PASSED**
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	**PASSED**
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	**PASSED**
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	**PASSED**
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	**PASSED**
SWC-	Presence of unused variables	CWE-1164: Irrelevant Code	**PASSED**

1 3 1			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-	Write to Arbitrary	CWE-123: Write-what-where Condition	PASSED

[1 2 4](#)	Storage Location		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	**NOT 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 | KYC

MADE IN GERMANY