



**SOLIDProof**  
*Bring trust into your projects*

**Blockchain Security | Smart Contract Audits | KYC**

MADE IN GERMANY

# Avara Audit

**Security Assessment**  
**17. February, 2022**

**For**



Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Inheritance Graph	12
CallGraph	13
Scope of Work/Verify Claims	14
Modifiers and public functions	20
Source Units in Scope	22
Critical issues	23
High issues	23
Medium issues	23
Low issues	23
Informational issues	24
Audit Comments	25
SWC Attacks	26

# 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 Uniswap, 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.

Version	Date	Description
1.0	17. February 2022	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>

## **Network**

Ethereum (ERC20)

## **Website**

<https://avara.cc/>

## **Telegram**

[https://t.me/avara\\_cc](https://t.me/avara_cc)

## **Twitter**

[https://twitter.com/avara\\_cc](https://twitter.com/avara_cc)

## **Facebook**

<https://www.facebook.com/AVARA-108154411726379>

## **Github**

<https://github.com/avara-cc/Avara>

## **Reddit**

<https://www.reddit.com/r/AVARA/>

## Description

AVARA is a token in the Binance Smart Chain (BSCBEP20) Network, offering multiple utilities to its users. AVARA HUB is the core of AVARA, where investors and users can access the services, products, and utilities AVARA offers. We have many great plans to improve the useability of AVARA, and we are fulltime working on it, to attract investors, and make partnerships to grow AVARA even further. We intend to keep investors included in the decision-making process throughout the life of the token and will take further suggestions for future growth, partnerships, brand ambassadorships, and more from our investors.

## Project Engagement

During the Date, **Teamname 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. They provided Solidproof.io with access to their code repository and whitepaper.

## Logo



## Contract Link

### v1.0

- Github
  - <https://github.com/avara-cc/AvaraETH>
  - Commit: c769f3053f2400793b65c5ad966739d07d07d501

# 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)
<b>Critical</b>	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.
<b>High</b>	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.
<b>Medium</b>	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.
<b>Low</b>	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.
<b>Informational</b>	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)

Imported packages:

📄 ./abstract/AvaraModule.sol

📄 ./library/SafeMath.sol

📄 ./abstract/Context.sol

📄 ./interface/common/IERC20.sol

📄 ./abstract/Ownable.sol

📄 ./interface/uniswap/IUniswapV3Pool.sol

📄 ./interface/uniswap/IUniswapV3Router.sol

📄 ./interface/common/IERC20Metadata.sol



## Tested Contract Files

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

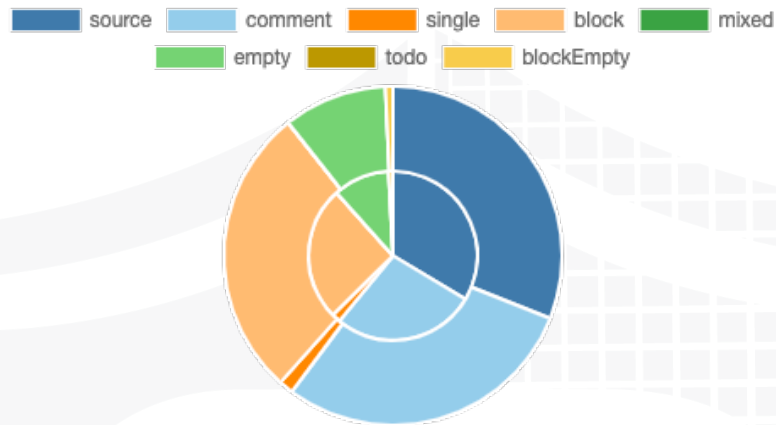
*A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.*

### v1.0

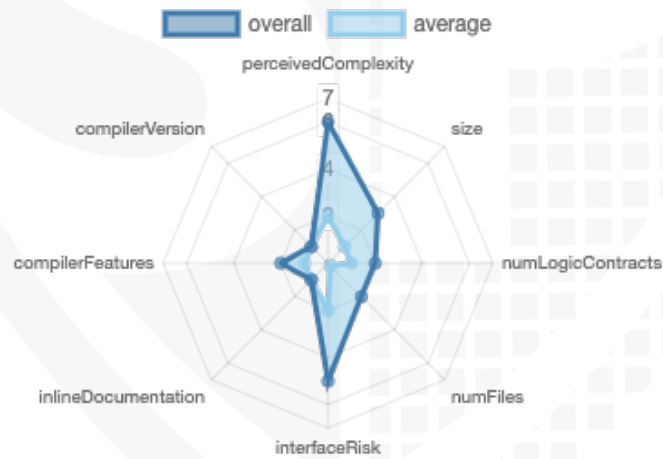
File Name	SHA-1 Hash
contracts/StakingModule.sol	c2de6bc121e615acf69355d1d223e9ebfa3299f2
contracts/AirDropModule.sol	29b78591e83a1e69e6dc85a6a68e1486d02bad20
contracts/BitDuelModule.sol	297be596958a0281b8fe838856c94c18b5b75302
contracts/library/SafeMath.sol	fc780aa608c43b6184763b42107d33c4d13acad2
contracts/Avara.sol	0103005324e2b668b7765737289b4c9ccac2588d
contracts/interface/common/IERC20Metadata.sol	3ca61103986b2dff51f9d3449c57274046cfca76
contracts/interface/common/IERC20.sol	61f4e94a2a1c8389e5cfb7856851992f417995a3
contracts/interface/uniswap/IUniswapV3PoolDeployer.sol	dee8fa2020f470313bfb9146bffcea0aaa3d3180
contracts/interface/uniswap/IUniswapV3Pool.sol	63c037dce1cc68e51dd3c238bf991baec92cc5d7
contracts/interface/uniswap/IUniswapV3Router.sol	14cecfdcf853b8c4592b4fa3c1a6f3060e7dc717
contracts/interface/uniswap/IUniswapV3Factory.sol	92fe2462609f68e9b113e76e510d144bc6c28837
contracts/abstract/Context.sol	055964aedef9b0d02cf9f88ea65e637405f704aa4
contracts/abstract/AvaraModule.sol	e13a49606fd63c7629c9a8377d39276e6d59736d
contracts/abstract/Ownable.sol	d7d4cbe8aae344500cc2e6118d513540e7da3da4

# Metrics

## Source Lines v1.0



## Risk Level v1.0



## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	4	1	12	3

### Exposed Functions

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

Version	Public	Payable
1.0	121	2

Version	External	Internal	Private	Pure	View
1.0	95	104	15	16	64

### State Variables

Version	Total	Public
1.0	45	18

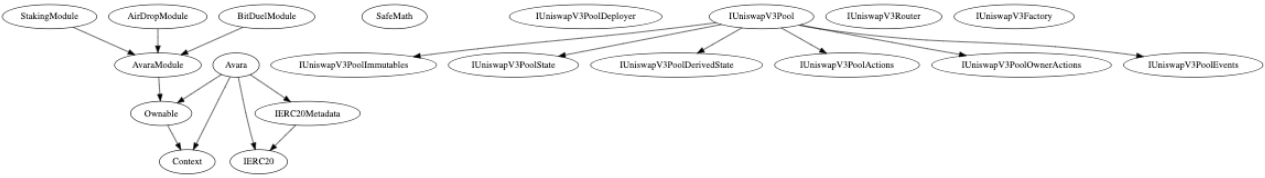
### Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	0.8.4		yes		

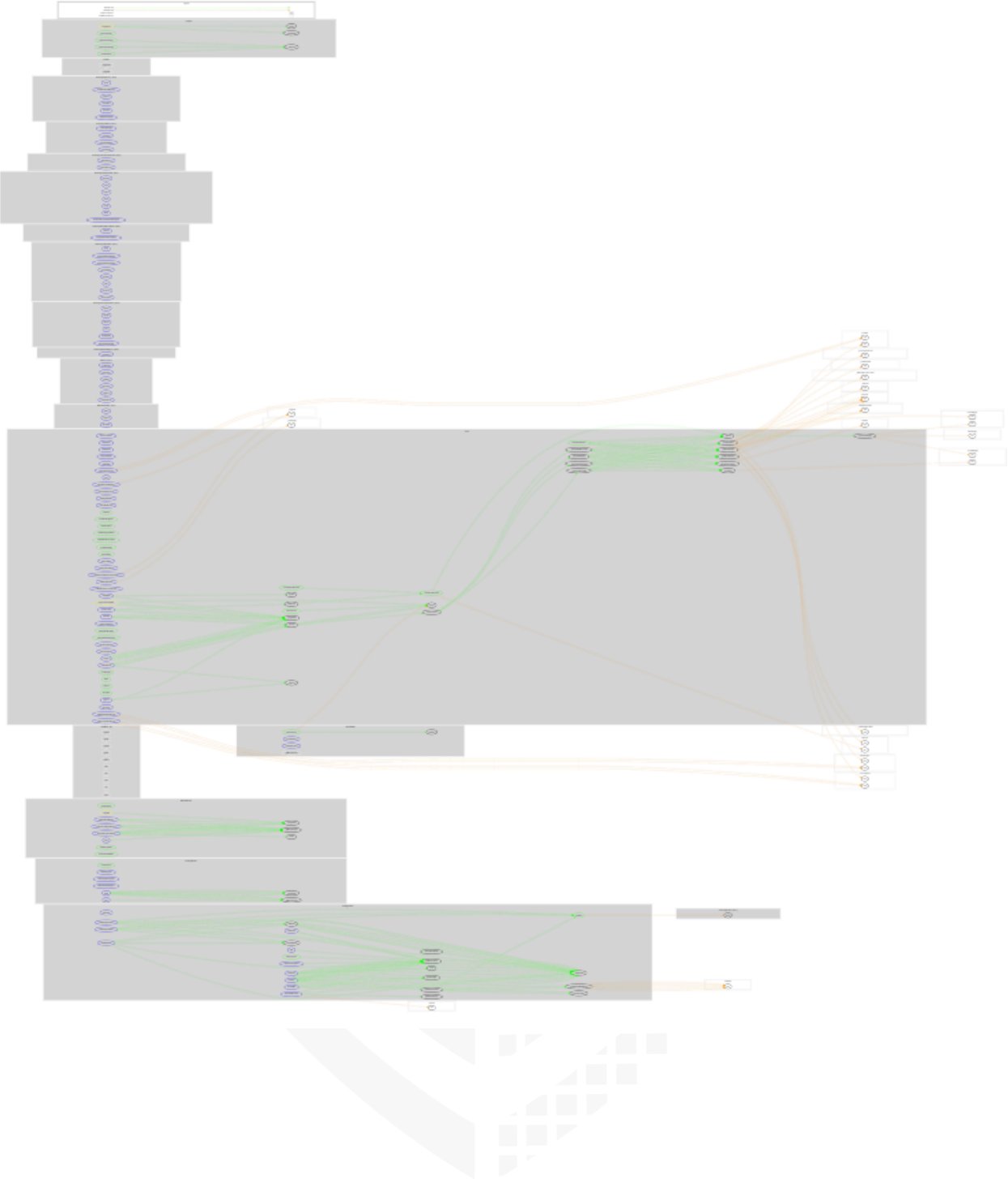
Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	New/Create/Create2
1.0	yes			yes		

# Inheritance Graph

## v1.0



# CallGraph 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. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

### Correct implementation of Token standard

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 contract v1.0

### BitDuelModule

addGameMaster
addToPlayerBalance
deductFromPlayerBalance
migratePlayerToAddress
removeGameMaster
renounceOwnership
transferOwnership

### AirdropModule

addParticipants
addUniqueParticipants
claim
renounceOwnership
transferOwnership

### StakingModule

addCombinedStake
addMultiplierStake
addTimeStake
distributeRewards
refillFunds
renounceOwnership
revertStake
transferOwnership
updateUniswapPool
useFunds
withdraw

## Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	—	—	—





## Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	✓	✓	✗
Deployer cannot burn	—	—	—

Comments:

### v1.0

- Deployer can lock user funds by
  - Setting `_maxTxAmount` to 0

## Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	—	—	—



## Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

### Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	🚩
Unverified / Not checked	✗
Not available	—

# Modifiers and public functions

## v1.0

### BitDuelModule

```

  ✓ 🔹 addToPlayerBalance
    ☹️ onlyGM
  ✓ 🔹 deductFromPlayerBalance
    ☹️ onlyGM
    🔹 migratePlayerToAddress
  ✓ 🔹 addGameMaster
    ☹️ onlyOwner
  ✓ 🔹 removeGameMaster
    ☹️ onlyOwner
  
```

### Avara

```

  ✓ 🔹 addModule
    ☹️ onlyOwner
  ✓ 🔹 removeModule
    ☹️ onlyOwner
    🔹 withdraw
  ✓ 🔹 setPlayerBalance
    ☹️ onlyOwnerOrModule
  ✓ 🔹 excludeFromReward
    ☹️ onlyOwner
  ✓ 🔹 includeInReward
    ☹️ onlyOwner
  ✓ 🔹 excludeFromFee
    ☹️ onlyOwner
  ✓ 🔹 includeInFee
    ☹️ onlyOwner
  ✓ 🔹 setDevWallet
    ☹️ onlyOwner
  ✓ 🔹 setPlayerPoolWallet
    ☹️ onlyOwner
  ✓ 🔹 setMarketingFeePercent
    ☹️ onlyOwner
  ✓ 🔹 setDeveloperFeePercent
    ☹️ onlyOwner
  ✓ 🔹 setBitDuelServiceFeePercent
    ☹️ onlyOwner
  ✓ 🔹 setEventFeePercent
    ☹️ onlyOwner
  ✓ 🔹 setSellPressureReductor
    ☹️ onlyOwner
  ✓ 🔹 setSellPressureReductorDecimals
    ☹️ onlyOwner
  ✓ 🔹 setMaxTxPercent
    ☹️ onlyOwner
  ✓ 🔹 setRewardEnabled
    ☹️ onlyOwner
  ✓ 🔹 setUniswapRouter
    ☹️ onlyOwner
  ✓ 🔹 setUniswapPool
    ☹️ onlyOwner
  ✓ 🔹 unstickEth
    ☹️ onlyOwner
  ✓ 🔹 unstickTokens
    ☹️ onlyOwner
    🔹 transfer
    🔹 approve
    🔹 transferFrom
  
```

### StakingModule

```

  ✓ 🔹 updateUniswapPool
    ☹️ onlyOwner
    🔹 addTimeStake
    🔹 addMultiplierStake
    🔹 addCombinedStake
    🔹 withdraw
  ✓ 🔹 useFunds
    ☹️ onlyOwner
  ✓ 🔹 revertStake
    ☹️ onlyOwner
    🔹 refillFunds
    🔹 distributeRewards
  
```

### AirDropModule

```

  ✓ 🔹 addParticipants
    ☹️ onlyOwner
  ✓ 🔹 addUniqueParticipants
    ☹️ onlyOwner
    🔹 claim
  
```

## Comments






















- Deployer can set following state variables
  - without any limitations

- `_airDropPool`
- `modules`
- `_playerPool`
- `_maxTxAmount`
- With a limitation of 20%
  - `_marketingFee`
  - `_developerFee`
  - `_bitDuelServiceFee`
  - `_eventFee`
  - `_sellPressureReductor`
  - `_sellPressureReductorDecimals`
- Deployer can enable/disable following state variables
  - `_isExcluded`
  - `_isExcludedFromFee`
  - `_rewardEnabled`
- Deployer can set following addresses
  - `_devWallet`
  - `_playerPoolWallet`
  - `_uniswapV3Router`
  - `_uniswapV3Pool`
- Deployer can transfer eth to own addresses
- Deployer can transfer tokens from contract to own address
- Deployer can send tokens to own address with `useFunds` function (StakingModukle, L372)
- Everybody can
  - `stake`
  - Refill funds

**Please check if an `OnlyOwner` or similar restrictive modifier has been forgotten.**

# Source Units in Scope

## v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/StakingModule.sol	1	————	426	426	251	112	220	
	contracts/AirDropModule.sol	1	————	89	89	38	32	49	————
	contracts/BitDuelModule.sol	1	————	110	110	50	37	59	————
	contracts/library/SafeMath.sol	1	————	206	206	69	122	10	
	contracts/Avara.sol	1	————	926	926	525	246	467	
	contracts/interface/common/IERC20Metadata.sol	————	1	42	31	4	29	9	
	contracts/interface/common/IERC20.sol	————	1	98	43	17	73	13	
	contracts/interface/uniswap/UniswapV3PoolDeployer.sol	————	1	49	39	3	32	3	————
	contracts/interface/uniswap/UniswapV3Pool.sol	————	7	542	227	72	327	69	————
	contracts/interface/uniswap/UniswapV3Router.sol	————	1	76	61	37	17	9	————
	contracts/interface/uniswap/UniswapV3Factory.sol	————	1	115	61	12	78	13	————
	contracts/abstract/Context.sol	1	————	41	41	10	27	1	————
	contracts/abstract/AvaraModule.sol	1	————	60	60	23	26	15	————
	contracts/abstract/Ownable.sol	1	————	93	93	32	49	25	————
	<b>Totals</b>	<b>8</b>	<b>12</b>	<b>2873</b>	<b>2413</b>	<b>1143</b>	<b>1207</b>	<b>962</b>	

## 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

# AUDIT PASSED

## Critical issues

**No critical issues**

## High issues

**No high issues**

## Medium issues

**No medium issues**

## Low issues

Issue	File	Type	Line	Description
#1	Main	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	-	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities
#2	Avara	Missing Zero Address Validation (missing-zero-check)	133, 134, 135, 408, 409, 420, 421	Check that the address is not zero
#3	Avara	Local variables shadowing	661, 662, 665, 666, 632, 633	Rename the local variables that shadow another component
#4	BitDuel Module	Owner can migrate to address	82	The owner can transfer tokens from an address to another without authorization

#5	Avara	Player balance can be changed	270	Only owner or module are allowed to change _playerPool of any addresses without authorization.
----	-------	-------------------------------	-----	---

## Informational issues

Issue	File	Type	Line	Description
#1	Context	Functions that are not used	37	Remove unused functions
#2	SafeMath	Functions that are not used	165, 181, 29, 63, 73, 50, 40	Remove unused functions
#3	Ownable	State variable is not used	See description	<p>The state variable _previousOwner is not used in any logic of the project.</p> <p>We recommend you to remove following:</p> <ul style="list-style-type: none"> <li>- function previousOwner() L54</li> <li>- State variable _previousOwner L33</li> <li>- _previousOwner initializing L89</li> </ul> <p>Replace it instead (L89) with a local variable for the oldOwner which is needed in the OwnershipTransferred event like the following</p> <pre>address _oldOwner = _owner;</pre> <p>and pass this to the event</p>



#4	SafeMath	Unnecessary library	See description	<p>SafeMath is not necessary anymore in pragma version above 0.8.x because it is automatically implemented in those versions.</p> <p>If you want to remove SafeMath library make sure to replace every operator functions from library (add, sub etc.) with raw mathematical operations</p>
#5	Avara	Naming convention	59, 57, 58, 91, 65	<p>Constants are not in UPPER_CASE_WITH_UNDERSCORES</p> <p>Make sure to change it everywhere else if you want to modify those variables</p>
#6	Avara	Old owner excluding from fee	L141	<p>Owner is added in the constructor to <code>_isExcludingFromFee</code>.</p> <p>The old owner is still excluded from fee after renouncing/transferring the ownership</p>

## 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.

### 17. February 2022:

- Read whole report for more information

## SWC Attacks

ID	Title	Relationships	Status
<a href="#">SW C-1 36</a>	Unencrypted Private Data On-Chain	<a href="#">CWE-767: Access to Critical Private Variable via Public Method</a>	PASSED
<a href="#">SW C-1 35</a>	Code With No Effects	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 34</a>	Message call with hardcoded gas amount	<a href="#">CWE-655: Improper Initialization</a>	PASSED
<a href="#">SW C-1 33</a>	Hash Collisions With Multiple Variable Length Arguments	<a href="#">CWE-294: Authentication Bypass by Capture-replay</a>	PASSED
<a href="#">SW C-1 32</a>	Unexpected Ether balance	<a href="#">CWE-667: Improper Locking</a>	PASSED
<a href="#">SW C-1 31</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 30</a>	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	PASSED
<a href="#">SW C-1 29</a>	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	PASSED
<a href="#">SW C-1 28</a>	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	PASSED

<a href="#">SW C-1 27</a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	<b>PASSED</b>
<a href="#">SW C-1 25</a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	<b>PASSED</b>
<a href="#">SW C-1 24</a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	<b>PASSED</b>
<a href="#">SW C-1 23</a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	<b>PASSED</b>
<a href="#">SW C-1 22</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	<b>PASSED</b>
<a href="#">SW C-1 21</a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>
<a href="#">SW C-1 20</a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	<b>PASSED</b>
<a href="#">SW C-11 9</a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>NOT PASSED</b>
<a href="#">SW C-11 8</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	<b>PASSED</b>
<a href="#">SW C-11 7</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>

<a href="#">SW C-11 6</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 5</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 4</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	<b>PASSED</b>
<a href="#">SW C-11 3</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	<b>PASSED</b>
<a href="#">SW C-11 2</a>	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 1</a>	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 0</a>	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	<b>PASSED</b>
<a href="#">SW C-1 09</a>	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	<b>PASSED</b>
<a href="#">SW C-1 08</a>	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW C-1 07</a>	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	<b>PASSED</b>
<a href="#">SW C-1 06</a>	Unprotected SELFDESTRUCT Instruction	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>

<a href="#">SW</a> <a href="#">C-1</a> <a href="#">05</a>	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">04</a>	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">03</a>	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">02</a>	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">01</a>	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">00</a>	Function Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>

The logo features the words "SolidProofed" in a white, handwritten-style script. The "P" is particularly large and stylized, with a long horizontal stroke that extends to the left. The background is a solid blue color with a faint, large shield emblem. The shield has a blue-to-white gradient and a grid-like pattern on its right side.

SolidProofed

**Blockchain Security | Smart Contract Audits | KYC**

A small horizontal bar representing the German flag, with black, red, and gold stripes.

MADE IN GERMANY