SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Avara

Audit

Security Assessment

21. February, 2022

For

AVARA

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.

Version	Date	Description
1.0	17. February 2022	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	21. February 2022	Reaudit

Network
Ethereum (ERC20)

Website
https://avara.cc/

Telegram
https://t.me/avara_cc

Twitter
https://twitter.com/avara_cc

Facebook
https://www.facebook.com/AVARA-108154411726379

Github
https://github.com/avara-cc/AvaraETH

Reddit
https://www.reddit.com/r/AVARA/

Description

AVARA is a module based token on the Ethereum Network, offering multiple utilities to its users. AVARA HUB is the core of AVARA, where investors and users can access AVARA services, products, and utilities. We have many great plans to improve AVARA, and we are full-time working on it, to attract investors, and make partnerships. 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 holders.

Project Engagement

During the 14th of February 2022, **Avara 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)
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.

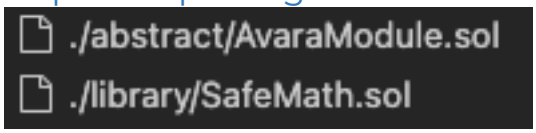
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

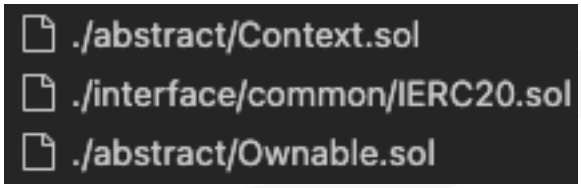
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

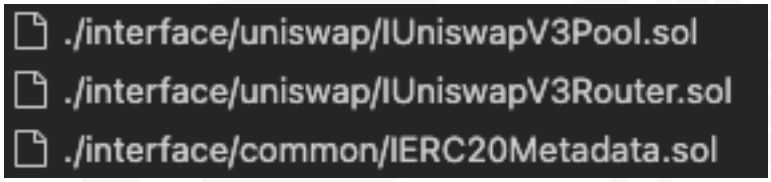
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

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

Imported packages:







Tested Contract Files

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

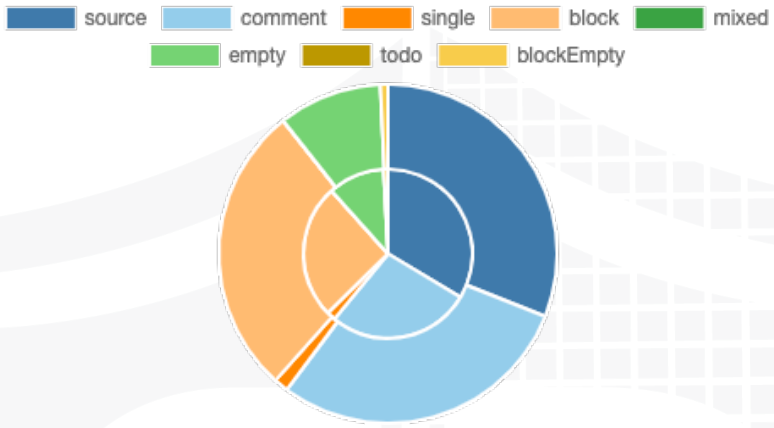
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

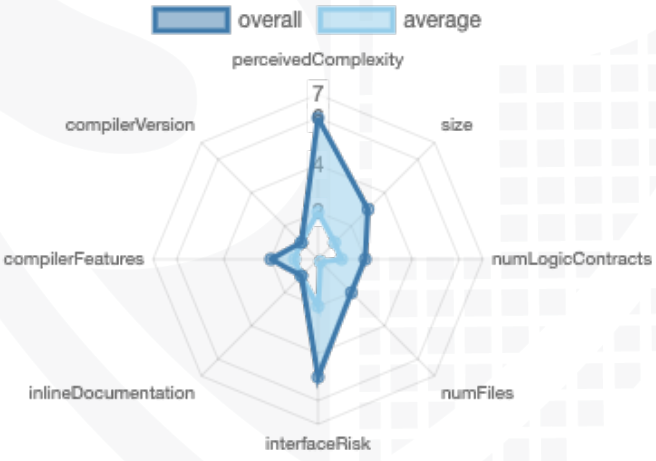
File Name	SHA-1 Hash
contracts/StakingModule.sol	c2de6bc121e615acf69355d1d223e9ebfa3299f2
contracts/AirDropModule.sol	29b78591e83a1e69e6dc85a6a68e1486d02bad20
contracts/BitDuelModule.sol	297be596958a0281b8fe838856c94c18b5b75302
contracts/library/SafeMath.sol	fc780aa608c43b6184763b42107d33c4d13acad2
contracts/Avara.sol	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	055964aedf9b0d02cf9f88ea65e637405f704aa4
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

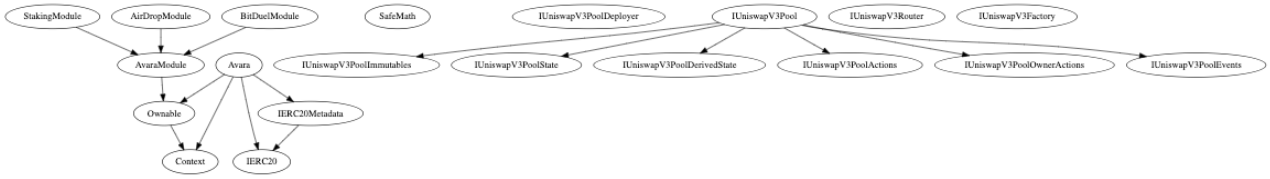
Version	Total	Public
1.0	45	18

Capabilities

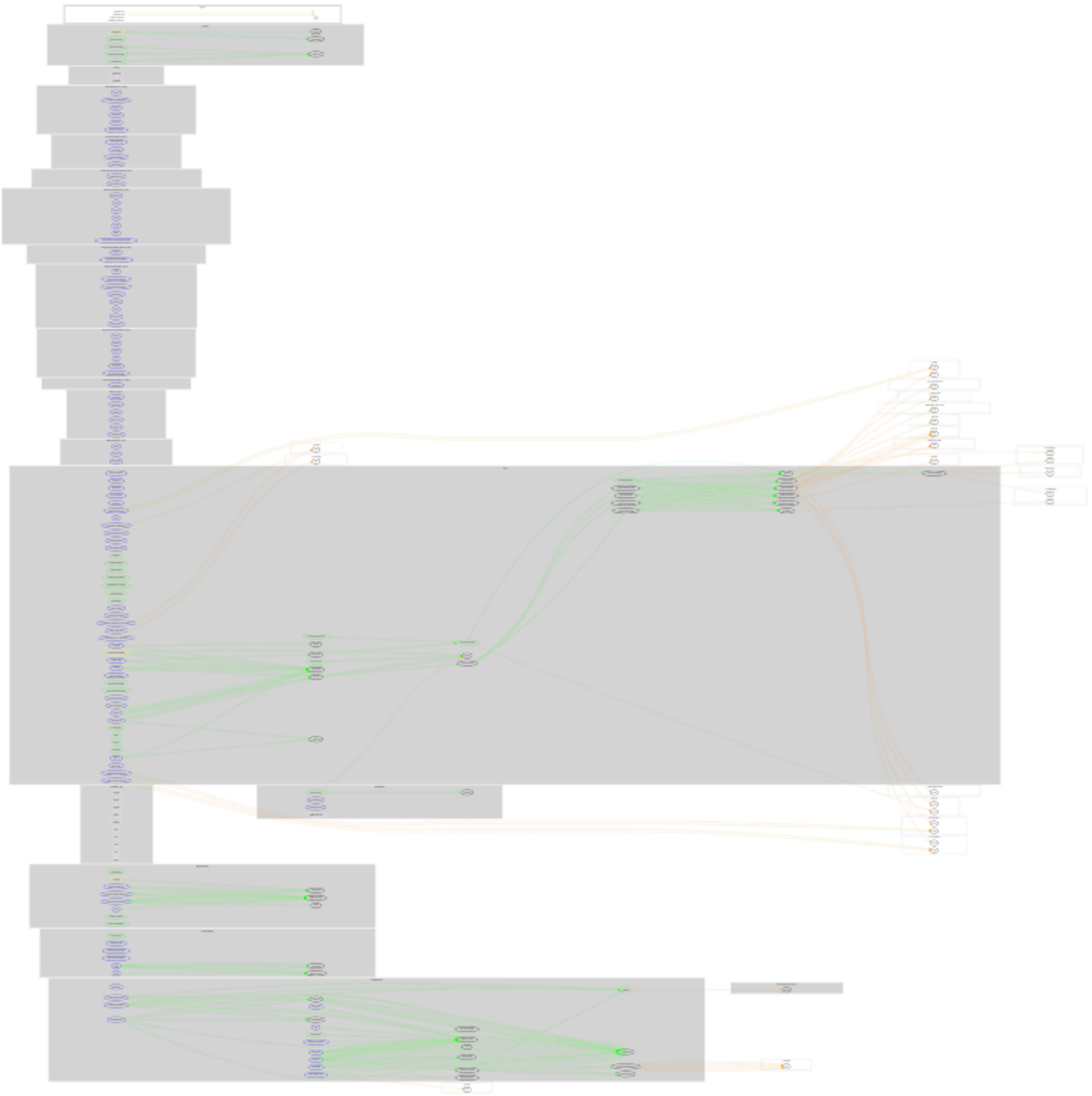
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`0.8.4`		`yes`		

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	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:

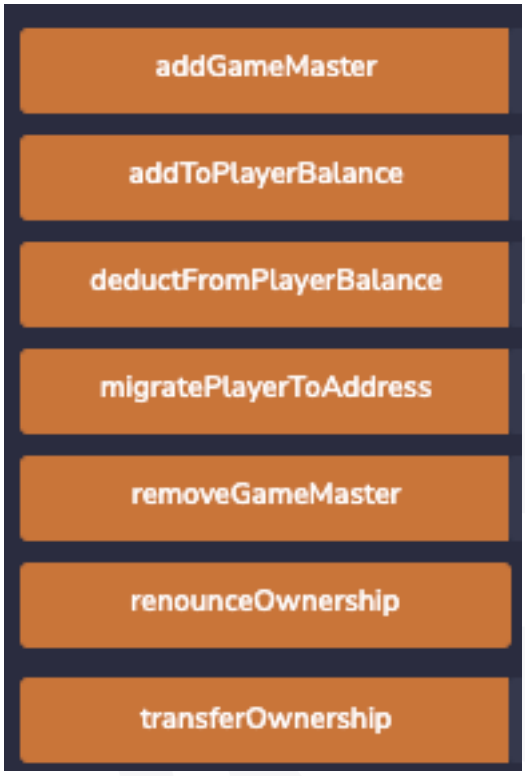
1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

Correct implementation of Token standard

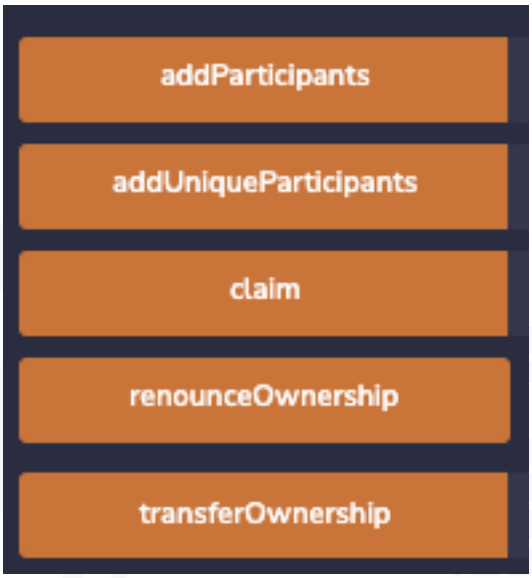
Function	Description	Exist	Tested	Verified
TotalSupply	provides information about the total token supply	✓	✓	✓
BalanceOf	provides account balance of the owner's account	✓	✓	✓
Transfer	executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	executes transfers of a 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

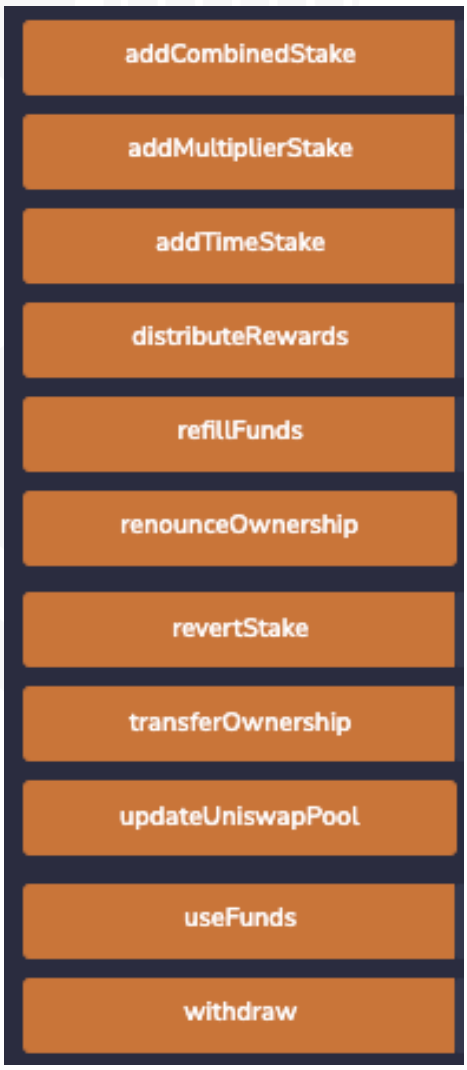
BitDuelModule



AirdropModule



StakingModule



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
✓	✓

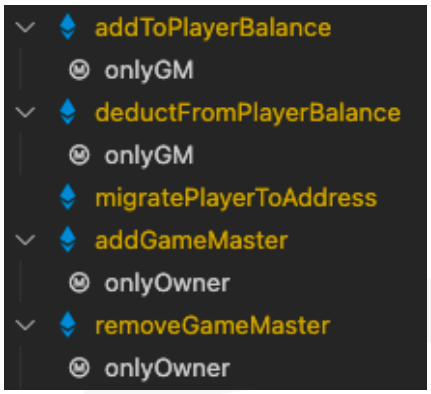
Legend

Attribute	Symbol
Verfified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	-

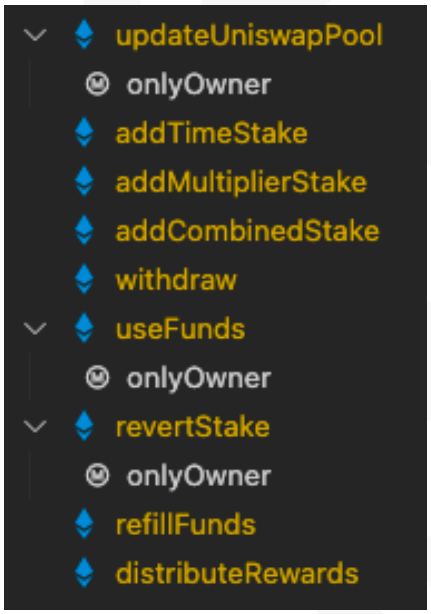
Modifiers and public functions

v1.0

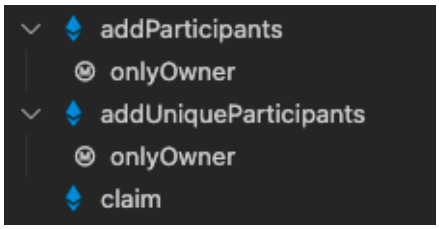
BitDuelModule



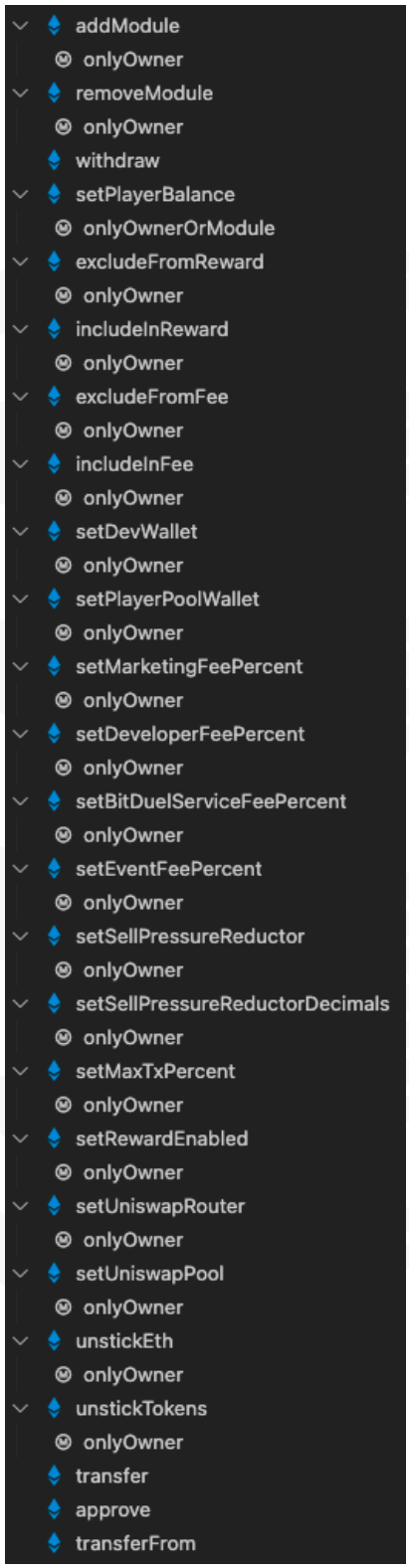
StakingModule



AirDropModule



Avara



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/IUniswapV3PoolDeployer.sol	———	1	49	39	3	32	3	———
🔍	contracts/interface/uniswap/IUniswapV3Pool.sol	———	7	542	227	72	327	69	———
🔍	contracts/interface/uniswap/IUniswapV3Router.sol	———	1	76	61	37	17	9	———
🔍	contracts/interface/uniswap/IUniswapV3Factory.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	———
📝📚🔍🎨	**Totals**	**8**	**12**	**2873**	**2413**	**1143**	**1207**	**962**	💰💠🧱🔆⏳

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	BitDuel Module	Owner can migrate to address	82	The owner can transfer tokens from an address to another without authorization
#3	Avara	Player balance can be changed	272	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	Avara	Naming convention	55, 56, 57, 89. 60, 63, 62	Constants are not in UPPER_CASE_WITH_UNDERSCORES Example: _decimals to DECIMALS Make sure to change it everywhere else if you want to modify those variables
#3	Avara	Old owner excluding from fee	L141	Owner is added in the constructor to _isExcludingFromFee. The old owner is still excluded from fee after renouncing/transferring the ownership

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

21. February 2022:

- Reaudit
- Read whole report 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-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 | KYC

MADE IN GERMANY