SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

MechaChain

Audit

Security Assessment
11. February, 2022

For

MECHACHAIN

Disclaimer

Version	Date	Description
1.0	11. February 2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://mechachain.io/en/

Telegram
https://t.me/mechachain

Twitter
https://twitter.com/mechachain

Discord
https://discord.gg/kMJCNaWaNz

Description

MechaChain is a 3D play-to-earn video game about robot combat and space conquest. Each robot, called "Mecha", is a collection of NFT composed of robot parts, which can be purchased online with the game cryptocurrency called Mechanium, Ethereum, or by card. These parts once assembled give birth to a robot in a PvP fighting video game.

The player earns Mechanium by winning battles, and can trade and buy new parts to become the best MechaChain pilot.

Project Engagement

During the 9th of February 2022, **MechaChain 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/thibautvdu/MechaChain-Smart-Contracts/tree/develop/contracts
    - Commit: 678acde6e82a5c049c8e0707cb8095cfa8c3218e

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:

Dependency / Import Path	Count
@openzeppelin/contracts/access/AccessControl.sol	1
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	1

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/MechaniumVestingWallet.sol	007f9143ba862fdf36b492154d551760f29fc58b
contracts/IMechaniumVestingWallet.sol	44c81af9534c74f4d64ddf09fe768ee88e2782cf
contracts/MechaniumGrowthVestingWallet.sol	5be660fffeccd17bf46e803b1fdac01311a3be6e

Metrics

Source Lines
v1.0
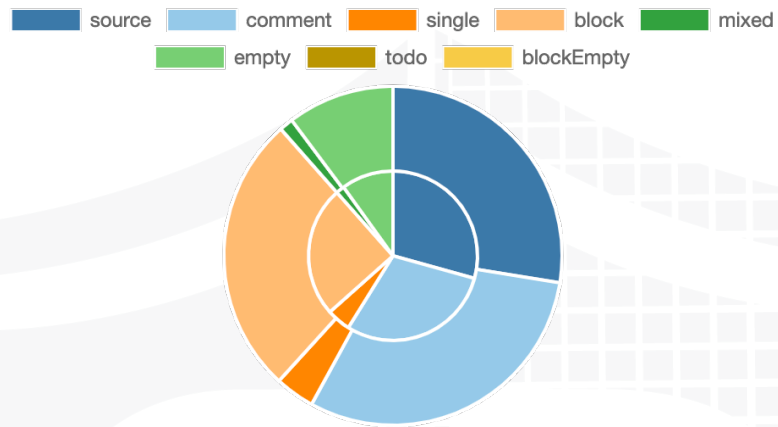
Risk Level
v1.0

Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	0	1	0

Exposed Functions

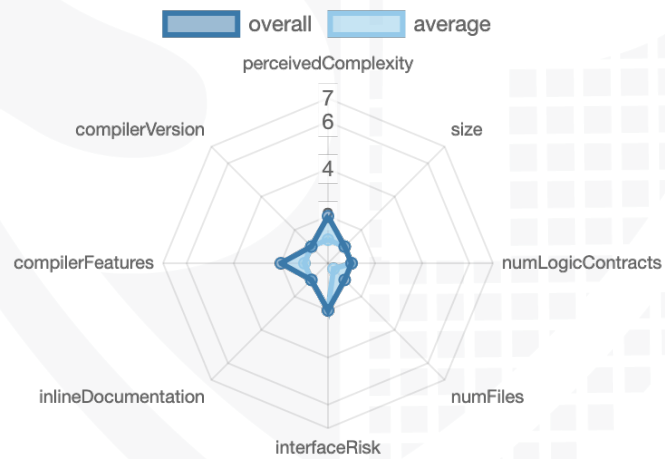*This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.*

Version	Public	Payable
1.0	20	0

Version	External	Internal	Private	Pure	View
1.0	10	4	0	0	18

State Variables

Version	Total	Public
1.0	7	1

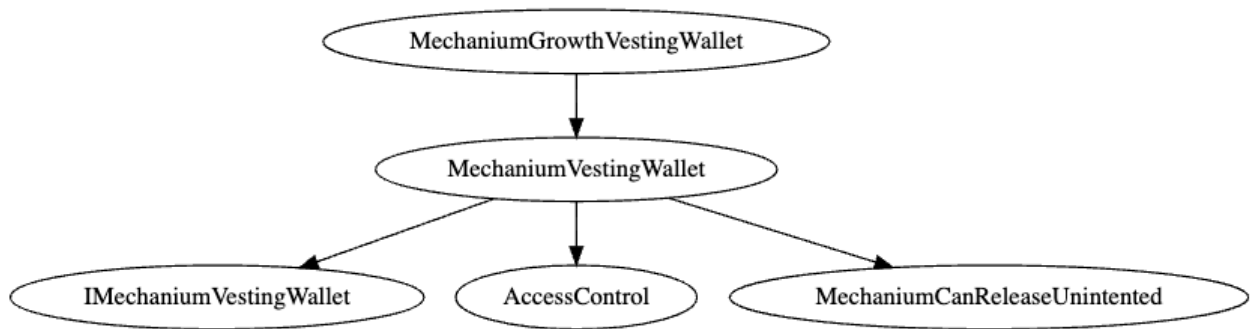## Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	^0.8.2				

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

Inheritance Graph
v1.0

CallGraph
v1.0

Legend

Internal Call
External Call
Defined Contract
Undefined Contract

IMechaniumVestingWallet (iface)

transfer
unlockableTokens
token
tokenBalance
totalSupply
totalReleasedTokens
vestingPerClock
vestingClockTime
initialVesting
startTime

MechaniumVestingWallet

<Constructor>
transfer
token
vestingPerClock
vestingClockTime
initialVesting
startTime

_addLockedToken
_setupRole
totalReleasedTokens
unlockableTokens
totalSupply
tokenBalance

diff
div

clockNumber
mul

IERC20
balanceOf
safeTransfer

_totalReleasedTokens
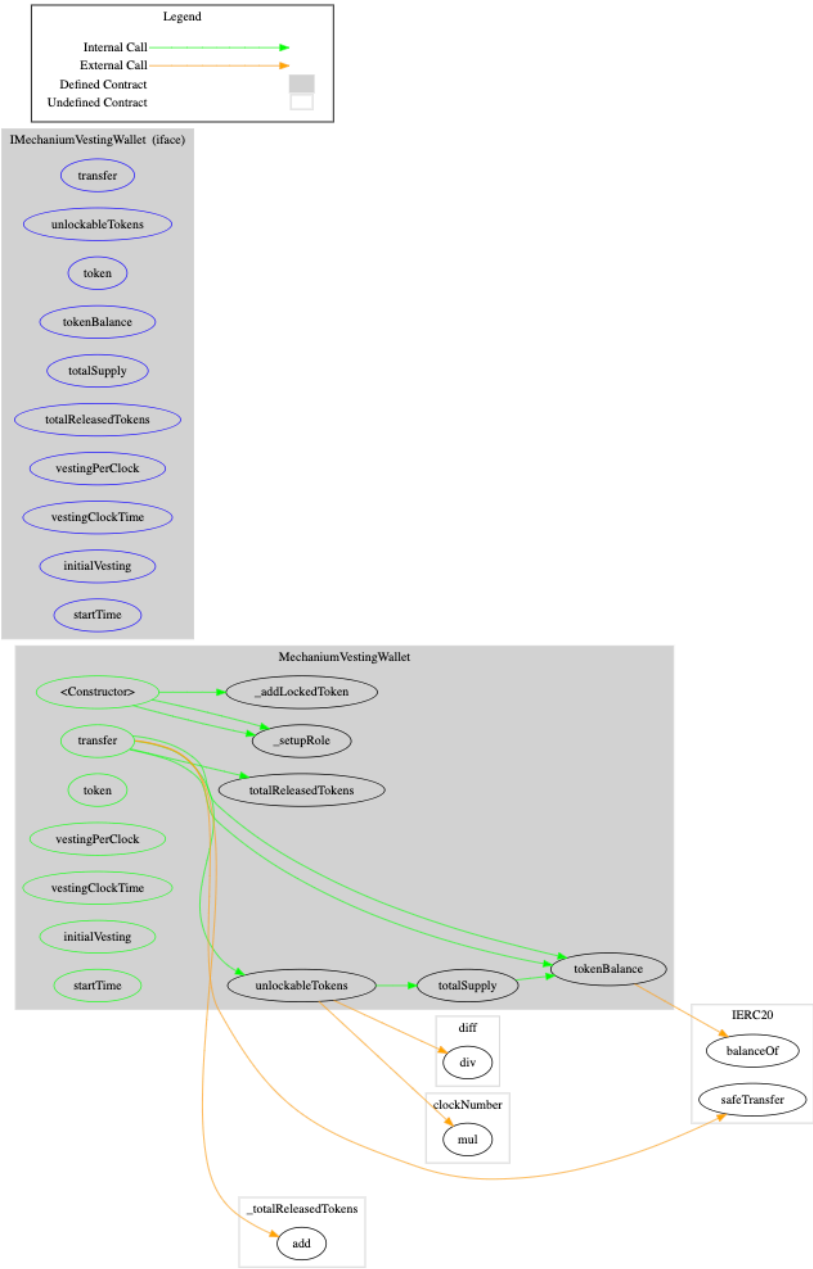add

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. Overall checkup (Smart Contract Security)

Write functions of contract v1.0

grantRole
releaseUninte...
renounceRole
revokeRole
transfer

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

- ⌄ ♦ transfer
  - ◎ onlyRole

- ⌄ ♦ grantRole
  - ◎ onlyRole
- ⌄ ♦ revokeRole
  - ◎ onlyRole
  - ♦ renounceRole

- ⌄ ♦ releaseUnintented
  - ◎ onlyRole

Comments

- Only addresses with
  - TRANSFER_ROLE
    - can use transfer function
  - DEFAULT_ADMIN_ROLE
    - Can use releaseUnintented function

Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/MechaniumVestingWallet.sol	1	———	214	209	94	86	67	▦
🔍	contracts/IMechaniumVestingWallet.sol	———	1	61	15	3	37	21	———
📝	contracts/MechaniumGrowthVestingWallet.sol	1	———	26	26	12	14	4	———
📝 🔍	**Totals**	**2**	**1**	**301**	**250**	**109**	**137**	**92**	▦

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	IMechaniumCanReleaseUnintented	A floating pragma is set	2	The current pragma Solidity directive is „"^0.8.2"".
#2	IMechaniumVestingWallet	A floating pragma is set	2	The current pragma Solidity directive is „"^0.8.2"".
#3	MechaniumCanReleaseUnintented	A floating pragma is set	2	The current pragma Solidity directive is „"^0.8.2"".
#4	MechaniumGrowthVestingWallet	A floating pragma is set	2	The current pragma Solidity directive is „"^0.8.2"".

Issue	File	Type	Line	Description
#5	Mechanium Vesting Wallet	A floating pragma is set	2	The current pragma Solidity directive is „"^0.8.2"".

Informational issues

Issue	File	Type	Line	Description
#1	IMechaniumVestingWallet	Misspelling	See description	Change following: - Mechanim to Mechanium L5 - MechaniumCanReleaseUnintented to MechaniumCanReleaseUnintended L19, L8 - transfered to transferred L74 Change variables/functions/ interfaces/imports etc. everywhere else
#2	IMechaniumCanReleaseUnintented	MIsspelling	See description	Change following: - Mechanim to Mechanium L5 - IMechaniumCanReleaseUnintented to IMechaniumCanReleaseUnintended L10 - unintented to unintended L12, L12, L13 - releaseUnintented to releaseUnintended L18 Change variables/functions/ interfaces/imports etc. everywhere else

#3	Mechanium CanRelease Uninten ted	MIsspelling	See description	Change following: - IMechaniumCanReleaseUnintented to IMechaniumCanReleaseUnintended L6, L14 - ReleaseUintentedTokens to ReleaseUnintendedTokens L23, L83 - unintented to unintended L21, L47 - releaseUnintented to releaseUnintended L52 Change variables/functions/ interfaces etc. everywhere else
#4	Mechan iumVest ingWall et	MIsspelling	See description	Change following: - IMechaniumCanReleaseUnintented to IMechaniumCanReleaseUnintended L8, L19 - transfered to transferred L74 Change variables/functions/ interfaces etc. everywhere else

Testing Protocol
MechaniumGrowthVestingWallet

✓ Smart contract should be deployed (153ms)
✓ DAO account should not have TRANSFER_ROLE yet
✓ Admin should be able to set TRANSFER_ROLE
6) Random user should not be able to set TRANSFER_ROLE
> No events were emitted
7) Admin should not be able to release unintented $MECHA
> No events were emitted
✓ Admin should be able to release unintented $MECHABIS
8) Admin should not be able to release unintented ETH (Reason: insufficient balance)
> No events were emitted
✓ Admin should be able to release unintented ETH
9) DAO account should not be able to set TRANSFER_ROLE
> No events were emitted

10) DAO account should not be able to set DEFAULT_ADMIN_ROLE
> No events were emitted
✓ Anyone can see the current balance (equal to 8 000 000 after deployment)
✓ Anyone can see the vesting Per Clock (15%)
✓ Anyone can see the vesting Clock Time (6 months)
✓ Anyone can see the initial vesting (40%)
✓ Anyone can see the start Time (days of deployment)
✓ Anyone can see the number of tokens that have been transferred (0 for now)
✓ Anyone can see the current balance of unlocked tokens (3 200 000 after deployment)
11) Random user should not be able to transfer tokens
> No events were emitted
12) DAO can't transfer more tokens than the schedule allows
> No events were emitted
✓ DAO can transfer 40% of the tokens after deployment on multiple transactions and to any address
✓ DAO can transfer +15% of the tokens (after 6 months)
✓ DAO can transfer 100% of the tokens (after 18 new months)
13) DAO can't transfer more tokens now
> No events were emitted
✓ Anyone can see the total supply (always equal to 8 000 000)
✓ [BONUS] The owner can add new tokens (+1 000 $MECHA)
✓ [BONUS] can transfer the new tokens (related to the first calendar)

Audit Comments
11. February 2022:
• 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](#)	**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**

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY