HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: WorkoutApp
Date: 3 Nov, 2023

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for WorkoutApp
Approved By	Oleksii Zaiats | SC Audits Head at Hacken OÜ
Tags	ERC20 token;
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://workoutapp.io/
Changelog	01.08.2023 - Initial Review 22.09.2023 - Second Review 03.11.2023 - Third Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by WorkoutApp (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

System Overview

WorkoutApp is a staking protocol with the following contracts:

- *Token* – simple ERC-20 token that mints all initial supply to a deployer. Additional minting is not allowed.
  It has the following attributes:
  - Name: WorkoutApp
  - Symbol: WRT
  - Decimals: 18
  - Total supply: 15B tokens.
  - Initial distribution:
    - trainEarn1Address: 0xdDcee1328c102A1880f4664350547f7421AEc3Fe 1.500.000.000 tokens
    - trainEarn2Address: 0xD4dCe63A35F2570644538A7821d604195e83475D 1.500.000.000 tokens
    - trainEarn3Address: 0xEe7Fb5f3770709CBd8dEf09137985F09bEDDe544 1.500.000.000 tokens
    - liq1Address: 0xdB450cb548568F4FAa3D814d86c628056f765308 1.500.000.000 tokens
    - liq2Address: 0xB7b92f9E9E9e525e25D51767bF17a719E1Fe418b 1.500.000.000 tokens
    - marketing1Address: 0xb31a5b71aF940B03A224Ab33e0B6B34d1fEBa4d4 1.125.000.000 tokens
    - marketing2Address: 0x6E2B9EAB334EecE13Fbd8dAF6F096C07fBEF7828 1.125.000.000 tokens
    - publicSaleAddress: 0x7fDCb42386032a7410db83d97F47B10c7DD531d0 1.500.000.000 tokens
    - dev1Address: 0x64B7992949e383Ce6d4999D0E8eFEc66B5e9bE09 750.000.000 tokens

- dev2Address: 0x9c3cb850Fca46f6E247e49C0C7fb4B71D37F9989 750.000.000 tokens
- team1Address: 0xDA31c02ddD4543f835657564CE03b420C122C575 750.000.000 tokens
- team2Address: 0x06F65b1a13Fa387B2e461272c3cDDAe58e9F0A13 750.000.000 tokens
- advAddress: 0xAa41bbA8033CC1cFDC52240248381B4eefE3BD72 450.000.000 tokens
- privAddress 0x651F50890525d7A9F6AaFaE398Fa55977DDd47f8 300.000.000 tokens

Privileged roles

- 2 SuperOwner that can set the role of an owner or give the owner role to an address.
- Owners that can add or remove role from other owners.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional requirements are provided.
- Technical description is provided.
- The environment is configured.
- NatSpec is provided.

Code quality

The total Code Quality score is **9** out of **10**.

- The development environment is configured.
- Minor code quality issues identified in: L05, I02, I04

Test coverage

Code coverage of the project is **80%** (branch coverage).

Security score

As a result of the audit, the code contains **2** low severity issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.8**. The system users should acknowledge all the risks summed up in the risks section of the report.

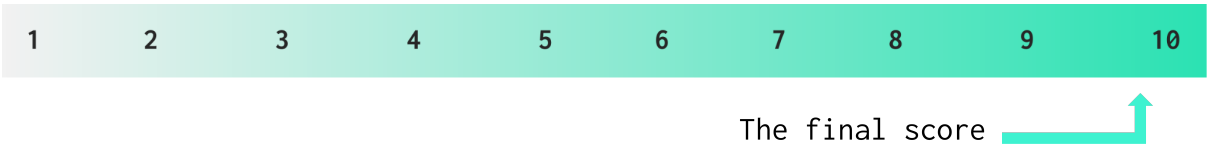

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
22 Aug 2023	5	2	1	0
22 Sep 2023	3	1	0	0
03 Nov 2023	2	0	0	0

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Risks

- There are two contracts out of scope from the third review: owner.sol and strings.sol. This audit is valid only for the code included in the workoutToken.sol contract, every other dependency is not included.
- There are two super owners of the contract that need to approve the initial distribution of workoutToken ERC20.
  To do so the contract needs to be deployed by one of the two super owners.
  The other one has to start the distribution and the deployer needs to confirm the distribution.
  If the super owner that deploys the contract also starts the distribution, there will be a Denial of Service.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

Item	Description	Status	Related Issues
Default Visibility	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Failed	L05
Integer Overflow and Underflow	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed	
Outdated Compiler Version	It is recommended to use a recent version of the Solidity compiler.	Passed	
Floating Pragma	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Failed	L01
Unchecked Call Return Value	The return value of a message call should be checked.	Passed	
Access Control & Authorization	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed	
SELFDESTRUCT Instruction	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant	
Check-Effect-Interaction	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed	
Assert Violation	Properly functioning code should never reach a failing assert statement.	Passed	
Deprecated Solidity Functions	Deprecated built-in functions should never be used.	Passed	
Delegatecall to Untrusted Callee	Delegatecalls should only be allowed to trusted addresses.	Not Relevant	
DoS (Denial of Service)	Execution of the code should never be blocked by a specific contract state unless required.	Passed	

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Race Conditions	Race Conditions and Transactions Order Dependency should not be possible.	Passed	
Authorization through tx.origin	tx.origin should not be used for authorization.	Passed	
Block values as a proxy for time	Block numbers should not be used for time calculations.	Not Relevant	
Signature Unique Id	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	Not Relevant	
Shadowing State Variable	State variables should not be shadowed.	Passed	
Weak Sources of Randomness	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant	
Incorrect Inheritance Order	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed	
Calls Only to Trusted Addresses	All external calls should be performed only to trusted addresses.	Not Relevant	
Presence of Unused Variables	The code should not contain unused variables if this is not justified by design.	Passed	
EIP Standards Violation	EIP standards should not be violated.	Passed	
Assets Integrity	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Not Relevant	
User Balances Manipulation	Contract owners or any other third party should not be able to access funds belonging to users.	Passed	
Data Consistency	Smart contract data should be consistent all over the data flow.	Passed	

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Flashloan Attack	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used. Contracts shouldn't rely on values that can be changed in the same transaction.	Not Relevant	
Token Supply Manipulation	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the Customer.	Passed	
Gas Limit and Loops	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed	
Style Guide Violation	Style guides and best practices should be followed.	Passed	
Requirements Compliance	The code should be compliant with the requirements provided by the Customer.	Passed	
Environment Consistency	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed	
Secure Oracles Usage	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Not Relevant	
Tests Coverage	The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed	
Stable Imports	The code should not reference draft contracts, which may be changed in the future.	Passed	

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Findings

■■■■ Critical

No critical severity issues were found.

■■■ High

H01. Contradiction

Impact	Medium
Likelihood	High

In the documentation, it is stated that the contract is used to buy and sell WorkoutApp tokens, this is not reflected in the code, the code represents a slightly modified version of the standard ERC20 token; it is not used to buy nor sell any tokens.

Path: ./contracts/workoutToken.sol

Recommendation: Change the documentation to match the code.

Found in: f0749b5

Status: Fixed (Revised commit: 34ac231)

■■ Medium

M01. Best Practice Violation

Impact	Low
Likelihood	High

Well-known contracts from projects like OpenZeppelin should be imported directly from source as the projects are in development and may update the contracts in future.

Path: ./contracts/workoutToken.sol : Context, IERC20, ERC20;

Recommendation: Import the contracts directly from source.

Found in: f0749b5

Status: Fixed (Revised commit: af25a86)

M02. Inconsistent Data

Impact	High
Likelihood	Low

deleteRole() could return true even when no action is performed, this happens when the *msg.sender* is allowed to delete a role, but the address passed in the function does not have the removed role.

Path: ./contracts/workoutToken.sol : deleteRole();

Recommendation: Check that the values of the *active* variable are actually changed before returning true.

Found in: f0749b5

Status: Fixed (Revised commit: 34ac231)

Low

L01. Floating Pragma

Impact	Low
Likelihood	Low

The project uses floating pragmas *^0.8.0*.

This may result in the contracts being deployed using the wrong pragma version, which is different from the one they were tested with. For example, they might be deployed using an outdated pragma version, which may include bugs that affect the system negatively.

Path: ./contracts/workoutToken.sol

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment. Consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Found in: f0749b5

Status: Reported (Revised commit: 34ac231)

L02. Style Guide Violation

Impact	Low
Likelihood	Low

The provided projects should follow the official guidelines.

Path: ./contracts/workoutToken.sol

Recommendation: Follow the official Solidity guidelines.

Found in: f0749b5

Status: Fixed (Revised commit: 34ac231)

L03. Unused Functions; Unused Contract

Impact	Low
Likelihood	Low

The libraries/imports/functions are unnecessary for the contract.

Most functions of the *Owner* contract are redundant and redundant and only *isSuperOwner* modifier with *superOwner* and *superOwner2* addresses are required.

Path: ./contracts/workoutToken.sol : Owner;

Recommendation: Clean-up the *Owner* contract.

Found in: f0749b5

Status: Fixed (Revised commit: af25a86)

L04. Redundant SafeMath

Impact	Low
Likelihood	Low

Using the *SafeMath* library on a contract that uses Solidity 0.8.0 and higher is an inefficient Gas model because the compiler already handles underflows and overflows.

Path: ./contracts/workoutToken.sol : SafeMath

Recommendation: Remove the library.

Found in: f0749b5

Status: Fixed (Revised commit: 34ac231)

L05. Functions That Should Be Declared External

Impact	Low
Likelihood	Low

Functions that are only called from outside the contract should be defined as external. External functions are much more Gas efficient compared to public functions.

Path: ./contracts/workoutToken.sol : startDistribution(), getDistributionStatus();

Recommendation: Change function visibility to external.

Found in: f0749b5

Status: Reported (Revised commit: 34ac231)

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Informational

I01. Function Name - Functionality Mismatch

The function's name *hasOwner()* and its functionality does not match. The functionality is to check if it is an owner, not if it has an owner.

Path: ./contracts/workoutToken.sol **:** hasOwner();

Recommendation: Rename the function.

Found in: f0749b5

Status: Fixed (Revised commit: af25a86)

I02. Variables That Should Be Declared Constant

State variables that do not change their value should be declared constant to save Gas.

Path: ./contracts/workoutToken.sol : superOwner, superOwner2, trainEarn1Address, trainEarn2Address, trainEarn3Address, liq1Address, liq2Address, marketing1Address, marketing2Address, publicSaleAddress, dev1Address, dev2Address, team1Address, team2Address, advAddress, privAddress;

Recommendation: Declare the above-mentioned variables as constants.

Found in: f0749b5

Status: Reported (Revised commit: 34ac231)

I03. Unindexed Events

Having indexed parameters in the events makes it easier to search for these events using *indexed* parameters as filters.

Path: ./contracts/workoutToken.sol : ListRolesForAddress(), RoleAdded(), RoleDeleted(), AddressDeleted(), WaitingForConfirmation(), AddressAdded();

Recommendation: Use the *"indexed"* keyword to the event parameters.

Found in: f0749b5

Status: Fixed (Revised commit: af25a86)

I04. Long Uint Literals

In the WorkoutApp *ERC20* contract, there are various uint with long literals that are not properly separated to aid readability.

Path: ./contracts/workoutToken.sol

Recommendation: Rewrite the long literals.

Found in: f0749b5

Status: Reported (Revised commit: 34ac231)

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Appendix 1. Severity Definitions

When auditing smart contracts Hacken is using a risk-based approach that considers the potential impact of any vulnerabilities and the likelihood of them being exploited. The matrix of impact and likelihood is a commonly used tool in risk management to help assess and prioritize risks.

The impact of a vulnerability refers to the potential harm that could result if it were to be exploited. For smart contracts, this could include the loss of funds or assets, unauthorized access or control, or reputational damage.

The likelihood of a vulnerability being exploited is determined by considering the likelihood of an attack occurring, the level of skill or resources required to exploit the vulnerability, and the presence of any mitigating controls that could reduce the likelihood of exploitation.

Risk Level	High Impact	Medium Impact	Low Impact
High Likelihood	**Critical**	**High**	**Medium**
Medium Likelihood	**High**	**Medium**	**Low**
Low Likelihood	**Medium**	**Low**	**Low**

Risk Levels

Critical: Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.

High: High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.

Medium: Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.

Low: Major deviations from best practices or major Gas inefficiency. These issues won't have a significant impact on code execution, don’t affect security score but can affect code quality score.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Impact Levels

High Impact: Risks that have a high impact are associated with financial losses, reputational damage, or major alterations to contract state. High impact issues typically involve invalid calculations, denial of service, token supply manipulation, and data consistency, but are not limited to those categories.

Medium Impact: Risks that have a medium impact could result in financial losses, reputational damage, or minor contract state manipulation. These risks can also be associated with undocumented behavior or violations of requirements.

Low Impact: Risks that have a low impact cannot lead to financial losses or state manipulation. These risks are typically related to unscalable functionality, contradictions, inconsistent data, or major violations of best practices.

Likelihood Levels

High Likelihood: Risks that have a high likelihood are those that are expected to occur frequently or are very likely to occur. These risks could be the result of known vulnerabilities or weaknesses in the contract, or could be the result of external factors such as attacks or exploits targeting similar contracts.

Medium Likelihood: Risks that have a medium likelihood are those that are possible but not as likely to occur as those in the high likelihood category. These risks could be the result of less severe vulnerabilities or weaknesses in the contract, or could be the result of less targeted attacks or exploits.

Low Likelihood: Risks that have a low likelihood are those that are unlikely to occur, but still possible. These risks could be the result of very specific or complex vulnerabilities or weaknesses in the contract, or could be the result of highly targeted attacks or exploits.

Informational

Informational issues are mostly connected to violations of best practices, typos in code, violations of code style, and dead or redundant code.

Informational issues are not affecting the score, but addressing them will be beneficial for the project.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Initial review scope

Repository	https://github.com/workouttoken/workout/tree/master/contracts
Commit	f0749b5158507d78819907cbd6e427af039974fc
Whitepaper	https://workoutapp.io/assets/docs/WhitepaperWorkoutApp-v2.pdf
Requirements	https://workoutapp.io/assets/docs/LitepaperWorkoutApp-v2.pdf
Contracts	File: workoutToken.sol SHA3: 71cd28451ce60738209a41e3cc46904f8a716ba61831156fb2c0ee44203f641f

Second review scope

Repository	https://github.com/workouttoken/workout/tree/master/contracts
Commit	34ac231dde37174e1fff4fdff73b6a83aea81760
Whitepaper	https://workoutapp.io/assets/docs/WhitepaperWorkoutApp-v2.pdf
Requirements	https://workoutapp.io/assets/docs/LitepaperWorkoutApp-v2.pdf
Contracts	File: workoutToken.sol SHA3: 6388b8d2f3dafd7564a1d8b2d13e44a1d89ac8d5b4855cc8b868eb0741b983e2

Third review scope

Repository	https://github.com/workouttoken/workout/tree/master/contracts
Commit	af25a863868b6256d493ccb0f593878b7424510f
Functional Requirements	Workout token update.pdf
Technical Requirements	Workout token update.pdf
Whitepaper	https://workoutapp.io/assets/docs/WhitepaperWorkoutApp-v2.pdf
Contracts	File: workoutToken.sol SHA3: d699462e391ae89b3fc970d1aa3be0d28d8b35bcc101a8e6119a2e7aac1f4841