HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Talent Protocol
Date: October 04th, 2021

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 Talent Protocol.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token; Staking
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/talentprotocol/contracts
Commit	70acb4924503fad6416ffa44b117ab51bef3fd21
Technical Documentation	YES
JS tests	YES
Timeline	29 SEPTEMBER 2021 - 04 OCTOBER 2021
Changelog	04 OCTOBER 2021 - INITIAL AUDIT

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 Talent Protocol (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 contract and its code review conducted between September 29th, 2021 - October 4th, 2021.

Scope

The scope of the project is smart contracts in the repository:

Repository:
https://github.com/talentprotocol/contracts
Commit:
70acb4924503fad6416ffa44b117ab51bef3fd21
Technical Documentation: Yes
JS tests: Yes
Contracts:
TalentToken.sol
TalentFactory.sol
TalentProtocol.sol
Staking.sol
staking/StableThenToken.sol
staking/RewardCalculator.sol
tokens/ERC1363Upgradeable.sol
tokens/ERC1363.sol

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency

	▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.

Insecure	Poor secured	Secured	Well-secured

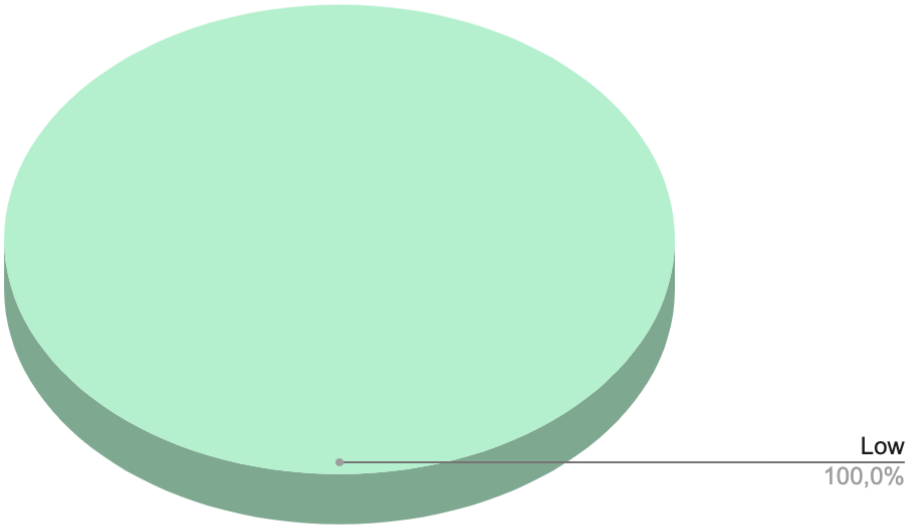
You are here 

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **3** low severity issues.

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

Graph 1. The distribution of vulnerabilities after the audit.



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

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution

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

Audit overview

Critical

No critical issues were found.

High

No high severity issues were found.

Medium

No medium severity issues were found.

Low

1. **Vulnerability**: Block timestamp

   Dangerous usage of block.timestamp. block.timestamp can be manipulated by miners. Some contracts are fully related on the block.timestamp.

   Recommendation: Please consider relying on the block.number instead.

2. **Vulnerability**: Missing address zero-check

   Method TalentToken.initialize accept address parameter _talent, method TalentToken.transferTalentWallet accept address parameter _newTalent, constructor for class Staking accept address parameter _factory all without checking for zero-address. That could cause loss of tokens or affect contract operability.

   Recommendation: Please add the zero-address check.

3. **Vulnerability**: A public function that could be declared external

   public functions that are never called by the contract should be declared **external** to save gas.

   Recommendation: Use the **external** attribute for functions never called from the contract.

Lines: TalentFactory.sol#73

```
function initialize() public virtual initializer {
```

Lines: TalentFactory.sol#85

```
function setMinter(address _minter) public onlyRole(DEFAULT_ADMIN_ROLE) {
```

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

Lines: TalentFactory.sol#97

```
function createTalent(
   address _talent,
   string memory _name,
   string memory _symbol
) public returns (address) {
```

Lines: TalentFactory.sol#159

```
function isTalentToken(address addr) public view override(ITalentFactory) returns
(bool) {
```

Lines: TalentToken.sol#77

```
function initialize(
   string memory _name,
   string memory _symbol,
   uint256 _initialSupply,
   address _talent,
   address _minter,
   address _admin
) public initializer {
```

Lines: TalentToken.sol#114

```
function mint(address _to, uint256 _amount) public override(ITalentToken)
onlyRole(ROLE_MINTER) {
```

Lines: TalentToken.sol#131

```
function burn(address _from, uint256 _amount) public override(ITalentToken)
onlyRole(ROLE_MINTER) {
```

Lines: TalentToken.sol#150

```
function transferTalentWallet(address _newTalent) public {
```

Lines: ERC1363Upgradeable.sol#56

```
function transferAndCall(address recipient, uint256 amount) public virtual
override returns (bool) {
```

Lines: ERC1363Upgradeable.sol#84

```
function transferFromAndCall(
   address sender,
   address recipient,
   uint256 amount
) public virtual override returns (bool) {
```

Lines: ERC1363Upgradeable.sol#117

```
function approveAndCall(address spender, uint256 amount) public virtual override
returns (bool) {
```

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

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found **3** low severity issues.

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 in accordance with the best industry practices at the date of this report, in relation to 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 audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. 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.

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 audit can't guarantee the explicit security of the audited smart contracts.