HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: TenPointOne
Date: Mar 03, 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 TenPointOne
Approved By	Yevheniy Bezuhlyi | SC Audits Head at Hacken OU
Type	Factory
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://tenpoint.one/
Changelog	02.03.2023 - Initial Review 03.04.2023 - Second 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 TenPointOne (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.

Scope

The scope of the project is review and security analysis of smart contracts in the repository:

Initial review scope

Repository	https://github.com/hknio/RP-Backend-V2-060aa1ee41679dcacd1501ff0
Commit	a094e82dd1c704feddb29183429e560f0ad3753d
Whitepaper	https://drive.google.com/file/d/1Eo1hjlj4bsCDTYQsLm02HAmAivWT0yRe/view
Functional Requirements	https://github.com/hknio/RP-Backend-V2-060aa1ee41679dcacd1501ff0/blob/main/README.md
Technical Requirements	https://github.com/hknio/RP-Backend-V2-060aa1ee41679dcacd1501ff0/blob/main/README.md
Contracts	File: ./contracts/0_RCPU_P.sol SHA3:2f84d78b168965ca3854fdf468b22bae9283bb0d66fb7b84a391d49dca328682 File: ./contracts/1_vFact_v112.sol SHA3 e6c2423f0e2af405abe0984bb58117049db7124b731b7c8b91e61008784ea562 File: ./contracts/2_Vault_v112.sol SHA3:8ddc2b677b6f3c5928be050a7a2c5cfcd8f24ef3d1e01571633d07296c068079 File: ./contracts/3_sFact_v112.sol SHA3:95955270241bacdcf1f73c6f6a98abc72bffe4adad62a1c1fecc31102f90fae9 File: ./contracts/4_Sega_v112.sol SHA3: bec313bd61a8274c1d97789e52bbce465c4962bd4951f749f6c2f0acd03047e7 File: ./interfaces/ISega_v112.sol SHA3: 869654f0e28cd10ce8c109aa72456ffe601fbf8a0b3182efc960954b46e3b0f0 File: ./interfaces/IsFact_v112.sol SHA3: c609df55b9632d1ef2a74479892204d6405575083fe68e1f015a4530df7ebaa4 File: ./interfaces/IVaultFactP_v112.sol SHA3: 4bee6f12ad689ea5a4322a190516c72c299ea4cb2649bd1b7596337e06ef1abc

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

Second review scope

Repository	https://github.com/hknio/RP-Backend-V2-060aa1ee41679dcacd1501ff0
Commit	e5ccd88c227252310d5bc1d5e654df59b1745366
Whitepaper	https://drive.google.com/file/d/1Eo1hjlj4bsCDTYQsLm02HAmAivWT0yRe/view
Functional Requirements	https://github.com/hknio/RP-Backend-V2-060aa1ee41679dcacd1501ff0/blob/main/README.md
Technical Requirements	https://github.com/hknio/RP-Backend-V2-060aa1ee41679dcacd1501ff0/blob/main/README.md
Contracts	File: ./contracts/0_RCU_P.sol SHA3: 21992b8d3a8915f5f89fd7faf82ce551829c332e802c161502d63042b42ca814 File: ./contracts/1_vFact_v112.sol SHA3: 94910c170cf7f8cf555060f5c40ecca7d525a44d3ebd10b9ced6c216a7cfbed2 File: ./contracts/2_Vault_v112.sol SHA3: 384e5eaa1cff082008bbd29fe416c87c1882130d9c7c3c37172957c386062e4b File: ./contracts/3_sFact_v112.sol SHA3: e626feb60ab9a63c71bd79a278be217e89d85fc7a7442fdefdbc2bdc1136b6e2 File: ./contracts/4_Sega_v112.sol SHA3: 6445bf272e56994ef8d1bf38f6b79d53e8b313121697d7266d043f15954d2074 File: ./contracts/SafeMath.sol SHA3: ba40d3d3c14fc8c74c910a34decce98aa77a9456135469ec5ed6edbbf2d14d62 File: ./interfaces/ISega_v112.sol SHA3: 9919035d7180ba7b4f34254c80ecebefafafb8936d92256adcdd445bfc2f8869 File: ./interfaces/IsFact_v112.sol SHA3: 5203581218547876310bf0af73548fcf15e37aabf6daed75d6e6dd6f56050247 File: ./interfaces/IVaultFactP.sol SHA3: 6550ed726dbc714bacbb6b714de8826018bbfb8a61e8c148b30ec01f60abaec5 File: ./interfaces/IvFact_v112.sol SHA3: cfdad15d9b2a0a827db2c8f49509bf39987e2107e3b5fe5afc96ba62f760b43f

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 the loss of user funds or contract state manipulation by external or internal actors.
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 by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor gas optimization. These issues won't have a significant impact on code execution but affect code quality

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**.

- Project overview is detailed
- All roles in the system are described.
- Use cases are described and detailed.
- All interactions are described.
- Run instructions are provided.
- Technical specification is provided.
- NatSpec is sufficient.

Code quality

The total Code Quality score is **8** out of **10**.

- Solidity Style Guide violations.
- Best practices violations.

Test coverage

Code coverage of the project is **98.2%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Interactions with several users are tested.

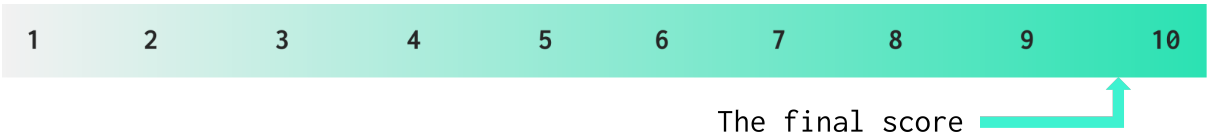
Security score

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: **9.6**.



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

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
02 Mar 2023	6	2	0	0
03 Apr 2023	2	0	0	0

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

System Overview

TenPointOne is a contract factory system with the following contracts:

- *DeployRCU_P* – is a head smart contract that stores information about smart contracts factories in the system.
- *vFact_v112* – is a smart contract factory for *Vault_v112*.
- *Vault_v112* – is a head smart contract over *SEGA_v112* that manages assets between *SEGA_v112* smart contracts.
- *sFact_v112* – is a smart contract factory for *SEGA_v112*.
- *SEGA_v112* – is a smart contract that is managed by *Vault_v112* and available to call third party methods and send assets.

Privileged roles

- The owner of the *DeployRCU_P* contract can arbitrarily change the owner, and addresses of factories.
- The owner of the *vFact_v112* contract can change the field *address RCU*.
- The owner of the *Vault_v112* contract can arbitrarily set a backup account, unlock period, transfer native and token assets. Moreover, create controlled *Sega_v112* smart contracts and manage them.
- The *Vault_v112* smart contract has a *backupAccount* role. This role can become the owner of the smart contract.
- The *SEGA_v112* smart contract has a *trader* role. This role can call external functions, transfer native and token assets.

Risks

- The view function returns a dynamic array of unlimited size. It is theoretically possible that the Gas cost for executing this function will exceed the Gas limit set in the node.
- Functions that are available to call external functions ignore returned data.

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	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Not Relevant
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	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	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Not Relevant
DoS (Denial of Service)	SWC-113 SWC-128	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	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Not Relevant
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Not Relevant
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155 EIP-712	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	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Not Relevant
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of Unused Variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP Standards Violation	EIP	EIP standards should not be violated.	Not Relevant
Assets Integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed
User Balances Manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Not Relevant
Data Consistency	**Custom**	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	**Custom**	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.	Not Relevant
Token Supply Manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Not Relevant
Gas Limit and Loops	**Custom**	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	**Custom**	Style guides and best practices should be followed.	Failed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	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	**Custom**	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	**Custom**	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	**Custom**	The code should not reference draft contracts, which may be changed in the future.	Not Relevant

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

Findings

Critical

No critical issues were found.

High

No high severity issues were found.

Medium

M01. Gas Limit and Loops

Functions `getVaultList(),getSegaList()` return a value depending on the amount of data stored in the smart contract. It is possible to increase the cost of the transaction so that it is not executed.

Path: ./contracts/0_RCU_P.sol, ./contracts/1_vFact_v112.sol : getVaultList(), ./contracts/3_sFact_v112.sol : getSegaList()

Recommendation: Return constant size data.

Found in: a094e82dd1c704feddb29183429e560f0ad3753d

Status: Fixed

M02. Ignores return values.

The function `callSC(address,bytes,uint),callSC(address,bytes,uint)` performs `Address.functionCallWithValue(address,bytes,uint,string)` but ignores the return value.

Path: ./contracts/4_Sega_v112.sol : callSC(address,bytes), callSC(address,bytes,uint).

Recommendation: implement a return value check.

Found in: a094e82dd1c704feddb29183429e560f0ad3753d

Status: Fixed

Low

L01. Floating Pragma

The smart contract uses floating pragma `^0.8.0`.

Path: ./contracts/0_RCU_P.sol, ./contracts/1_vFact_v112.sol, ./contracts/2_Vault_v112.sol, ./contracts/3_sFact_v112.sol, ./contracts/4_Sega_v112.sol, ./interfaces/ISega_v112.sol, ./interfaces/IsFact_v112.sol, ./interfaces/IVaultFactP.sol

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

Found in: a094e82dd1c704feddb29183429e560f0ad3753d

Status: Fixed

L02. Unused arguments

The function *createSega(address,address,address)* has an unused argument.

Path: ./contracts/3_sFact_v112.sol : createSega(address,address, address)

Recommendation:Rename redundant arguments.

Found in: a094e82dd1c704feddb29183429e560f0ad3753d

Status: Fixed

L03. Solidity Code Style Guide Violations

SegaMap and *SegaList* state variables of the *Vault_v112* contract violate the naming convention. Local and state variables should all be in mixedCase.

The layouts of the *SEGA_v112* and *Vault_v112* contracts violate the order of functions convention.

Path: ./contracts/*

Recommendation: follow the official solidity code style guide.

Found in: a094e82dd1c704feddb29183429e560f0ad3753d

Status: Reported (naming convention is still violated)

L04. State variables default visibility

The contract should specify a visibility level for all functions and state variables.

Path: ./contracts/*

Recommendation: Specify variables as public, internal, or private. Explicitly define visibility for all state variables.

Found in: a094e82dd1c704feddb29183429e560f0ad3753d

Status: Fixed

L05. Missing Zero Address Validation

Address parameters are used without checking against the possibility of 0x0. This issue is found in constructors and set methods of every file in the audit scope.

Path: ./contracts/*

Recommendation: Implement zero address checks.

Found in: a094e82dd1c704feddb29183429e560f0ad3753d

Status: Reported

L06. Unnecessary state variable

Mapping `(address => uint)SegaMap` and the dynamic array `uint[] SegaList` are shared data and can be merged.

Path: ./contracts/2_Vault_v112.sol

Recommendation: Replace two state variables into one mapping `(address => address)`.

Found in: a094e82dd1c704feddb29183429e560f0ad3753d

Status: Fixed

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.