HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Taikai
Date: Aug 31st, 2022

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 TAIKAI S.A.
Approved By	Evgeniy Bezuglyi | SC Audits Department Head at Hacken OU
Type	ERC721 token; Decentralized development
Platform	EVM
Network	Ethereum
Language	Solidity
Methods	Manual Review, Automated Review, Architecture Review
Website	https://taikai.network
Timeline	01.08.2022 - 29.08.2022
Changelog	08.08.2022 - Initial Review 15.08.2022 - Second Review 30.08.2022 - 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 Taikai S.A. (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 smart contracts in the repository:

Initial review scope

Repository:
https://github.com/taikai/dappkit
Commit:
adf31a6ff8d67081704611ba5528897ca5e88ba4
Technical Documentation:
Documentation

Integration and Unit Tests: Yes
Contracts:
File: ./contracts/bepro/Network_v2.sol
SHA3: 553345cbfb01e5239b896212391f1d9eee6566e47ba94d78a97a8b6334560482

File: ./contracts/bepro/Network_Registry.sol
SHA3: c6c56027fc3537dc3385417b4bf6e52a2a1554293bb2bc306942f340c2f68ff2

File: ./contracts/bepro/BountyToken.sol
SHA3: 06df7f126c630984fe7a142c93560b9e7bbb3432d07ea9f7d7797c83f713c3e6

Second review scope

Repository:
https://github.com/taikai/dappkit
Commit:
c8c52801a974c9368086181cdc31b2ad88cc723c
Technical Documentation:
Documentation

Integration and Unit Tests: Yes
Contracts:
File: ./contracts/bepro/NetworkV2.sol
SHA3: 35b5e698601ffd2ab2df2925f286e1410e18334df116081d38a95bf6e9ffc68f

File: ./contracts/bepro/NetworkRegistry.sol
SHA3: c752ef721f0c79e00dbd2d318271118ba7f06830cbe0cd7f99e8f1227bfb7730

File: ./contracts/bepro/BountyToken.sol
SHA3: 8e8d10a7f5b1a47f29ebfc51a9a1ad2b140519b0ead2786e5ab4936188931f32

Third review scope

Repository:
https://github.com/taikai/dappkit
Commit:
bb5a589f12c4996d8e01ee7c8ed8b59b081219b1
Technical Documentation:

Documentation

Integration and Unit Tests: Yes
Contracts:
File: ./contracts/bepro/NetworkV2.sol
SHA3: 9f91b4567d919fbfc9c456bfb8ccfa2741089eb352000c5b65e0dfba6e0bb8b7

File: ./contracts/bepro/NetworkRegistry.sol
SHA3: e2301dfd2ca8fbf7ad33869b735404b504d59d8ddc68fe112b49e33a320eeb83

File: ./contracts/bepro/BountyToken.sol
SHA3: 01fedd81cda376d57d4b9155be0b3bccf7db64dcce2091eedc2a8bfa7bdc5da1

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 cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution.

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 methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**. Functional requirements are provided and well-describes the product.

Code quality

The total CodeQuality score is **8** out of **10**. Basic user interactions are covered with tests. Test coverage is **25%** for *BountyToken*, **82%** for *NetworkRegistry* and *NetworkV2*.

Architecture quality

The architecture quality score is **10** out of **10**. Code is separated to different contracts, following the single responsibility principle. Development environment is well set-up.

Security score

As a result of the audit, the code contains **3** 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**.

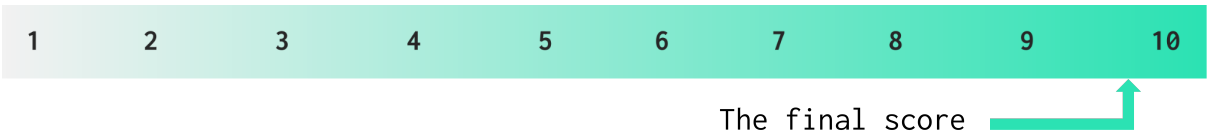

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
5 August 2022	10	5	3	0
15 August 2022	4	0	0	0
29 August 2022	3	0	0	0

Checked Items

We have audited provided smart contracts for commonly known and more specific vulnerabilities. Here are some of the items that are 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.	Passed
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.	Not Relevant
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 it is required.	Passed
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization	SWC-115	tx.origin should not be used for	Passed

through tx.origin		authorization.	
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Passed
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifier should always be used. All parameters from the signature should be used in signer recovery	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.	Passed
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.	Passed
Assets integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions.	Passed
User Balances manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed
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.	Passed
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of	Passed

		data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	
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 100%, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Failed
Stable Imports	**Custom**	The code should not reference draft contracts, that may be changed in the future.	Passed

System Overview

Taikai is a mixed-purpose system with the following contracts:

- *BountyToken* — a simple ERC-721 token that allows minting only to the contract owner. Transfers for token are disabled.
- *Network_v2* — entry point to the system that allows creating a bounty, creating a pull request, etc.
- *Network_Registry* - a management contract for networks that allow creating/removing networks.

Privileged roles

- The dispatcher role of the *BountyToken* can mint tokens.
- The owner of the *BountyToken* can assign a dispatcher role.
- The owner of the *NetworkRegistry* contract can change the network creation minimum lock amount.
- The owner of the *NetworkRegistry* contract can change the network registration fee. The fee could not be bigger than 10% of the locked amount required for network creation.
- The owner of the *NetworkRegistry* contract can change, cancel and close fees. Fees could be bigger than 100%.
- The owner of the *NetworkV2* contract can change the following parameters: councilAmount, draftTime, disputableTime, percentageNeededForDispute, mergeCreatorFeeShare, proposerFeeShare, oracleExchangeRate, cancelableTime.
- The owner of the *NetworkRegistry* contract can add or remove trusted token addresses.

Risks

- The repository contains contracts that are out of the audit scope.

Findings

Critical

No critical severity issues were found.

High

1. **Incorrect validation**

   getBountyToken validation is incorrect and allows access only to indexes out of array bound.

   Path: ./contracts/bepro/BountyToken.sol : getBountyToken()

   Recommendation: Use *tokenIds.length > id* validation instead.

   Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

2. **Checks-Effects-Interactions pattern violation**

   State update is done after the token transfer, which violates the CEI pattern.

   Paths: ./contracts/bepro/Network_Registry.sol : lock(), unlock(), registerNetwork()

   ./contracts/bepro/Network_v2.sol : _cancelFundingRequest(), updateBountyAmount(), retractFunds(), closeBounty()

   Recommendation: Update state only after external call.

   Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

3. **Unauthorized access**

   addAllowedTokens and *removeAllowedTokens* functions are available for any user, which could lead to unauthorized adding/removing tokens, so anyone could affect *Network_v2* contract functionality.

   Path: ./contracts/bepro/Network_Registry.sol : addAllowedTokens(), removeAllowedTokens()

   Recommendation: Restrict access to these functions.

   Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

Medium

1. **Missing validation**

   Max fees are not validated during contract creation, while *changeLockPercentageFee* and *changeGlobalFees* have logic to validate max fee.

   Path: ./contracts/bepro/Network_Registry.sol : constructor()

   Recommendation: Validate fees during contract creation.

Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

2. **Denial of Service vulnerability**

   getAllowedTokens function loops over all *_transactionalTokens* and *_rewardTokens* arrays. The size of these arrays is not limited, large iterations may cause failed transactions due to exceeding Gas.

   Path: ./contracts/bepro/Network_Registry.sol: getAllowedTokens()

   Recommendation: Fix the logic not to rely on arrays length.

   Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

3. **Denial of Service vulnerability**

   cancelPullRequest function loops over *proposals* array. The size of this array is not limited, large iterations may cause failed transactions due to exceeding Gas.

   Path: ./contracts/bepro/Network_v2.sol: cancelPullRequest()

   Recommendation: Fix the logic not to rely on arrays length.

   Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

4. **Denial of Service vulnerability**

   closeBounty function loops over *proposal.details* array. The size of this array is not limited, large iterations may cause failed transactions due to exceeding Gas.

   Path: ./contracts/bepro/Network_v2.sol: closeBounty()

   Recommendation: Fix the logic not to rely on arrays length.

   Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

5. **Non-final implementation**

   The contract contains an empty function.

   The implementation is non-final and unusable.

   Path: ./contracts/bepro/BountyToken.sol

   Recommendation: Implement missing functionality, throw an error if functions should never be called, or declare the contract as *abstract*.

   Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

Low

1. **Public functions instead of external**

   Some functions are declared as public, although they are not called internally in the related contract.

Public function visibility consumes more Gas than external visibility.

Paths: ./contracts/bepro/BountyToken.sol : getBountyToken(), getNextId(), setDispatcher()

./contracts/bepro/Network_Registry.sol : isAllowedToken(), amountOfNetworks(), lock(), unlock(), registerNetwork(), changeAmountForNetworkCreation(), changeLockPercentageFee(), changeGlobalFees(), getAllowedTokens(), awardBounty()

./contracts/bepro/Network_v2.sol : cancelPullRequest(), markPullRequestReadyForReview(), createBountyProposal(), changeNetworkParameter()

Recommendation: Change public visibility with external.

Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

2. **Tautology**

uint is always bigger or equals than zero, so *_cancelFee >= 0*, *_closeFee >= 0*, *_value >= 0* and *newAmount >= 0* are redundant.

After the second review issue still exists in the *changeNetworkParameter* function.

Paths: ./contracts/bepro/Network_Registry.sol : changeLockPercentageFee(), changeGlobalFees()

./contracts/bepro/Network_v2.sol : changeNetworkParameter()

Recommendation: Remove redundant checks.

Status: Fixed (bb5a589f12c4996d8e01ee7c8ed8b59b081219b1)

3. **Redundant import**

The use of unnecessary imports will increase the Gas consumption of the code. Thus they should be removed from the code.

Paths: ./contracts/bepro/Network_Registry.sol, ./contracts/bepro/Network_v2.sol

Recommendation: Remove *./math/SafePercentMath.sol* import.

Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

4. **Missing event emitting**

Events for critical state changes should be emitted for tracking things off-chain.

Path: ./contracts/bepro/Network_Registry.sol : addAllowedTokens(), removeAllowedTokens()

Recommendation: Emit *ChangeAllowedTokens* event after tokens change.

Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

5. **Missing state variable update**

   During the network close, *networksArray* is not updated, which leads to an incorrect value returned by *amountOfNetworks* function.

   Path: ./contracts/bepro/Network_Registry.sol : unlock()

   Recommendation: Removed closed network from the *networksArray*.

   Status: Reported

6. **Style guide violation**

   The provided projects should follow the official guidelines.

   After the second review, issues still exist, as “Order of Functions” is violated. Consider moving internal functions at the end of the contract.

   Path: ./contracts/bepro/Network_v2.sol

   Recommendation: Follow official recommendations related to “Order of Layout”. The contract should be named using CapWords style.

   Status: Reported

7. **Missing event emitting**

   Events for critical state changes should be emitted for tracking things off-chain.

   Path: ./contracts/bepro/Network_v2.sol : delegateOracles(), takeBackOracles()

   Recommendation: Emit state changes.

   Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

8. **Redundant require statement**

   Some functions perform the same check twice. To save Gas and simplify code, one validation should be removed.

   After the second review issue still exists. *fundBounty* function checks *_isFunded(id, false);* and *require(bounty.funded == false, "1");*

   Path: ./contracts/bepro/Network_v2.sol : fundBounty()

   Recommendation: Remove redundant check.

   Status: Reported

9. **Redundant inheritance**

A contract that extends *ReentrancyGuardOptimized* or *ReentrancyGuard* never uses functionality of the guards, so they can be removed.

Paths: ./contracts/bepro/Network_v2.sol,

./contracts/bepro/Network_Registry.sol

Recommendation: Remove redundant check.

Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

10. Zero valued transactions

Multiple functions allow transactions with zero tokens amount, which does not make sense and should be reverted.

Path: ./contracts/bepro/Network_v2.sol : manageOracles(), delegateOracles(),

Recommendation: Validate if the amount is not zero.

Status: Fixed (c8c52801a974c9368086181cdc31b2ad88cc723c)

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date 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 to 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, Consultant cannot guarantee the explicit security of the audited smart contracts.