HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: NeoNomad
Date: Jul 18th, 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 NeoNomad.
Approved By	Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type	Solana SPL token; Staking
Platform	Solana
Language	Rust
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://www.neonomad.finance/
Timeline	02.05.2022 - 18.07.2022
Changelog	13.05.2022 - Initial Review 13.06.2022 - Second Review 18.07.2022 - 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 NeoNomad 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/NeoNomadFinance/staking_contract
Commit:
b038f5fdd117cf94df8999c256540ce6c01fca51
Documentation: Yes
https://docs.neonomad.finance/neonomad-documentation/defi-tutorials/staking-step-by-step-guide
Technical Documentation: Yes
NNI_STAKING_CONTRACT-1.pdf
JS tests: Yes
Contracts: neonomad/src/*

Second review scope

Repository:
https://github.com/NeoNomadFinance/staking_contract
Commit:
d7206d9bc7c09ab32825c9658404e7dec4b558f0
Documentation: Yes
https://docs.neonomad.finance/neonomad-documentation/defi-tutorials/staking-step-by-step-guide
Technical Documentation: Yes
NNI_STAKING_CONTRACT-1.pdf
JS tests: Yes
Contracts: neonomad/src/*

Third review scope

Repository:
https://github.com/NeoNomadFinance/staking_contract
Commit:
08ad2d1ad6f2f4b0033ab9f069edf9d84c07e730
Documentation: Yes
https://docs.neonomad.finance/neonomad-documentation/defi-tutorials/staking-step-by-step-guide
Technical Documentation: Yes
NNI_STAKING_CONTRACT-1.pdf
JS tests: Yes
Contracts: neonomad/src/*

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 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 Customer provided functional and technical documentation. The total Documentation Quality score is **8** out of **10**. A few pages are not completed yet or lack detailed documentation.

Code quality

The total CodeQuality score is **5** out of **10**. No unit tests were provided for scripts. Logging messages are not simple and clear. Code is not covered by comments.

Architecture quality

The architecture quality score is **3** out of **10**. All the logic is implemented in one file. Some functions code could be moved to separate structs to be used from there to improve code readability.

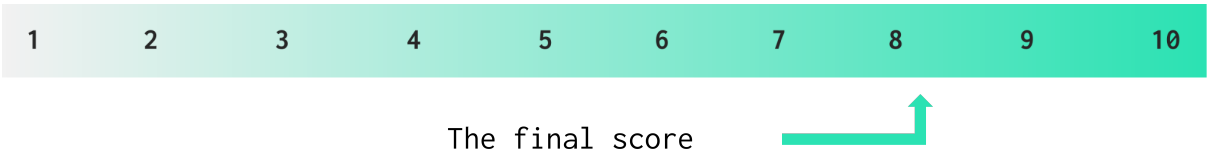
Security score

As a result of the third audit, the code contains **1** medium severity issue. The security score is **9** 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: **8.2**



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

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	Description	Status
Missing Signer Checks	Case when instruction should only be available to a restricted set of entities, but the program does not verify that the call has been signed by the appropriate entity (e.g., by checking AccountInfo::is_signer).	Passed
Missing Ownership Checks	For accounts that are not supposed to be fully user-controlled, the program does not check the AccountInfo::owner field.	Passed
Missing rent exemption checks	All Solana accounts holding an Account, Mint, or Multisig must contain enough SOL to be considered rent exempt. Otherwise, the accounts may fail to load.	Passed
Signed invocation of unverified programs	The program does not verify the pubkey of any program called via the invoke_signed() API.	Passed
Solana account confusions	The program fails to ensure that the account data has the type it expects to have.	Passed
Redeployment with cross-instance confusion	The program fails to ensure that the wasm code has the code it expects to have.	Passed
Arithmetic overflow/underflows	If an arithmetic operation results in a higher or lower value, the value will wrap around with two’s complement.	Passed
Numerical precision errors	Numeric calculations on floating point can cause precision errors, wich can accumulate.	Passed
Loss of precision in calculation	Numeric calculations on integer types such as division can loss precision.	Passed
Casting truncation	Potential truncation problem with a cast conversion.	Passed
Exponential complexity in calculation	Finding computational complexity in calculations.	Passed
Missing freeze authority checks	When freezing is enabled, but the program does not verify that the freezing account call has been signed by the appropriate freeze_authority.	Passed

Insufficient SPL-Token account verification	Finding extra checks that should not exist with the given type of accounts.	Passed
Over/under payment of loans	A loan overpayment is when paying extra towards a loan over and above the agreed monthly repayment. A loan underpayment is when paying less towards a loan over and below the agreed monthly repayment.	Passed
Anti-pattern instruction calls	Calling some anti-pattern instructions specific to Solana blockchain.	Passed
Unsafe Rust code	The Rust type system does not check the memory safety of unsafe Rust code. Thus, if a smart contract contains any unsafe Rust code, it may still suffer from memory corruptions such as buffer overflows, use after frees, uninitialized memory, etc.	Failed
Outdated dependencies	Rust/Cargo makes it easy to manage dependencies, but the dependencies can be outdated or contain known security vulnerabilities. cargo-outdated can be used to check outdated dependencies.	Failed
Redundant code	Repeated code or dead code that can be cleaned or simplified to reduce code complexity.	Passed
Do not follow security best practices	Failing to properly use assertions, check user errors, multisig, etc.	Passed
Project specification implementation check	Ensuring that the contract logic correctly implements the project specifications.	Passed
Contract-specific low-level vulnerabilities	Examining the code in detail for contract-specific low-level vulnerabilities.	Passed
Ruling out economic attacks	Economic rules that can be exploited to steal funds.	Passed
DoS (Denial of Service)	Execution of the code should never be blocked by a specific contract state unless it is required.	Passed

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

Front-running or sandwiching	Checking for instructions that allow front-running or sandwiching attacks.	Passed
Unsafe design vulnerabilities	Checking for unsafe design which might lead to common vulnerabilities being introduced in the future.	Passed
As-of-yet solana unknown classes of vulnerabilities	Checking for any other, as-of-yet unknown classes of vulnerabilities arising from the structure of the Solana blockchain.	Passed
Rug-pull mechanisms or hidden backdoors	Checking for rug-pull mechanisms or hidden backdoors.	Passed

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

System Overview

NeoNomad is staking – a contract that rewards users for staking their tokens. APY depends on the tokens provided by the owner and cannot be calculated before reward tokens are deposited.

Risks

- Anchor is in active development, so all APIs are subject to change. Anchor code is unaudited. The usage is risky.

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

Findings

Critical

No critical severity issues were found.

High

1. **Insufficient system type.**

   UncheckedAccount is not recommended for using as system_program type, it should be used carefully.`System_program` field should be a type of anchor_lang::Program<System>, but instead UncheckedAccount is used in CreateState, CreateFarmPool, CreateExtraRewardsConfigs, SetExtraRewardsConfigs, CreatePoolUser, CreateUserEtherAddress, SetUserEtherAddress, Stake, Harvest structs.

   Contract: programs/neonomad/src/lib.rs

   Functions: create_state, create_pool

   Recommendation: Consider changing system_program field type to Program<System>

   Status: Fixed

Medium

1. **Yanked package version.**

   Anchor-spl 0.16.2 is marked as yanked. This is usually done when the authors of a package have a compelling reason that a certain version of a package should not be used and strongly suggest that the package should not be used.

   Contract: programs/neonomad/

   Recommendation: Consider updating anchor-lang to the latest version. Not updating can potentially bring some security issues.

   Status: Reported

2. **Unsafe rust code.**

   `anchor-attribute-access-control` uses unsafe rust code. Crate dependencies anchor-syn 0.16.2 -> *bs58 0.3.1*, *anyhow 1.0.56*, *proc-macro2 1.0.36* contain unsafe code.

   Contract: programs/neonomad/

   Recommendation: Consider removing this package from the dependency list.

   Status: Mitigated (with customer notice).

3. **Using unsafe property.**

   Using ctx.remaining_accounts directly is not safe in the Context struct. They are not deserialized or validated.

Contract: programs/neonomad/src/lib.rs

Functions: change_tokens_per_second, create_pool, close_pool, change_pool_point, change_tokens_per_second

Recommendation: Consider removing this package from the dependency list.

Status: Fixed

Low

1. **Unneeded `return` statement.**

   A return statement that returns no value and occurs just before the function would have "fallen through" the bottom. These statements may be safely removed.

   Contract: programs/neonomad/src/lib.rs:611:9

   Function: get_extra_reward_percentage

   Recommendation: Remove the return statement. The default value of u64 will be returned.

   Status: Fixed

2. **Extra unused lifetimes**.

   The additional lifetimes make the code look more complicated, while there is nothing out of the ordinary going on. Removing them leads to more readable code.

   Contract:programs/neonomad/src/lib.rs:579:17

   Function: validate<'info>

   Recommendation: Remove unused lifetime.

   Status: Fixed

3. **Redundant code.**

   Useless conversion to the same type: `u128`.

   Contract: programs/neonomad/src/lib.rs:696:26

   Function: calculate_reward_amount

   Recommendation: Consider removing `u128::from()`: `self.reward_debt`

   Status: Fixed

4. **Redundant code.**

   Useless conversion to the same type: `u128`.

   Contract: programs/neonomad/src/lib.rs:699:34

   Function: calculate_reward_amount

Recommendation: Consider removing `u128::from()`: `pending_amount`

Status: Fixed

5. **Extra unused lifetimes.**

   The additional lifetimes make the code look more complicated, while there is nothing out of the ordinary going on.
   Removing them leads to more readable code.

   Contract: programs/neonomad/src/lib.rs:707:30

   Function: calculate_reward_debt

   Recommendation: Remove unused lifetime.

   Status: Fixed

6. **Needless borrowing.**

   This expression borrows a reference (`&anchor_lang::prelude::Pubkey`) immediately dereferenced by the compiler.

   Contract:programs/neonomad/src/lib.rs

   Function:change_tokens_per_second

   Recommendation: Consider changing this &_ctx.program_id to: `_ctx.program_id`.
   Suggests that the receiver of the expression borrows the expression.

   Status: Fixed

7. **Needless borrowing.**

   This expression borrows a reference (`&anchor_lang::prelude::Pubkey`) immediately dereferenced by the compiler.

   Contract: programs/neonomad/src/lib.rs

   Function: change_tokens_per_second

   Recommendation: Consider changing this &provided_token_accountinfo to: `provided_token_accountinfo`.
   Suggests that the receiver of the expression borrows the expression.

   Status: Fixed

8. **Needless borrowing.**

   This expression borrows a reference (`&anchor_lang::prelude::Pubkey`) immediately dereferenced by the compiler.

   Contract: programs/neonomad/src/lib.rs

   Function: close_pool

Recommendation: Consider changing this &provided_token_accountinfo to: `provided_token_accountinfo`.
Suggests that the receiver of the expression borrows the expression.

Status: Fixed

9. **Needless borrowing.**

This expression borrows a reference (`&anchor_lang::prelude::Pubkey`) immediately dereferenced by the compiler.

Contract: programs/neonomad/src/lib.rs

Function: close_pool

Recommendation: Consider changing this &_ctx.program_id to: `_ctx.program_id`.
Suggests that the receiver of the expression borrows the expression.

Status: Fixed

10. **Needless borrowing.**

This expression borrows a reference (`&anchor_lang::prelude::Pubkey`) immediately dereferenced by the compiler.

Contract: programs/neonomad/src/lib.rs

Function: close_pool_point

Recommendation: Consider changing this &_ctx.program_id to: `_ctx.program_id`.
Suggests that the receiver of the expression borrows the expression.

Status: Fixed

11. **Needless borrowing.**

This expression borrows a reference (`&anchor_lang::prelude::Pubkey`) immediately dereferenced by the compiler.

Contract: programs/neonomad/src/lib.rs

Function: change_pool_point

Recommendation: Consider changing this &_ctx.program_id to: `_ctx.program_id`.
Suggests that the receiver of the expression borrows the expression.

Status: Fixed

12. **Needless borrowing.**

This expression borrows a reference (`&anchor_lang::prelude::Pubkey`) immediately dereferenced by the compiler.

Contract: programs/neonomad/src/lib.rs

Function: change_pool_point

Recommendation: Consider changing this &provided_token_accountinfo to: `provided_token_accountinfo`.
Suggests that the receiver of the expression borrows the expression.

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 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 audit makes no statements or warranties on the security of the code. It also cannot be considered a 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.

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 cannot guarantee the explicit security of the audited smart contracts.