Security Audit – Neon Governance

Neodyme AG

June 7th, 2023

Contents

Introduction

Neon engaged Neodyme to undertake multiple extensive security audits. This includes the launch infrastructure, governance contracts and multisig, and the primary Neon contract, which operates an Ethereum Virtual Machine (EVM) on the Solana platform. These elements together comprise all the on-chain components developed by the Neon team.

This report's focus is the project's governance infrastructure, whereas a separate report will cover the audit of the main Neon smart contract.

Neodyme conducted a comprehensive audit of the governance infrastructure from December 5th to 23rd, 2022. The audit did not uncover any significant security issues, identifying one low-severity issue, which has since been fixed.

This document first outlines the precise scope and methodology of the audit, provides an overview of the Neon governance architecture, and concludes with a presentation of all findings and implemented resolutions.

Project Overview

The main Neon contract implements a fully compatible Ethereum Virtual Machine (EVM). This will allow users to deploy solidity-based smart contracts on Solana while profiting from Solana's performance and quick finality and still interacting with the SPL-Token ecosystem. Neon will have its own version of the native Ethereum token, the NEON token. There will also be a treasury that collects protocol fees, which can, in turn, be used to develop the Neon ecosystem further.

Many of the components will have an owner. Someone needs to manage the treasury, hold the upgrade authority of the Neon program and do the initial distribution of NEON tokens.

For that, Neon will use a combination of a DAO, a multisig, and a dedicated maintenance contract. Contrary to many other protocols, Neon already has detailed plans and prepared launch scripts to pass complete control of Neon and supporting infrastructure into a DAO.

Scope

In addition to the main Neon contract, there are four contracts responsible for the management of Neon:

- The SPL Governance Program
- The Vestin-Addin Program
- The Multisig Program
- The Maintenance Program

There are also off-chain components in the form of launch-scripts, which set up the whole infrastructure and migrate it through the different phases of the launch.

Since the SPL Governance Program is identical to the upstream one and has already been audited there, it is out-of-scope for this audit. However, the Vesting-Addin, which provides the vote weights for vested Neon tokens, is specifically included. As well as the Maintenance Program, the Multisig, and the off-chain launch-scripts.

This report does specifically not include the main Neon contract, which is presented in a companion report. It also does not include a review of any specific DAO parameters, which among other things, also depend on the value and markets of the governance tokens, not just the contract implementation, making them only possible to review upfront with a detailed economic examination.

The relevant repositories are:

- https://github.com/neonlabsorg/neon-spl-governance
  - Contract: addin-vesting
  - Contract: maintenance contract
  - Initial Hash: `a5e6e0b2f321afdbaff17a2027ab35dc1c686ff9`
- https://github.com/neonlabsorg/launch-script
  - off-chain launch script
  - multisig
  - Initial Hash: `515b8b3a6e65d5159503e6933acc24c8baf811ec`
- Treasury Setup, in particular `collect_treasury.rs` and `create_main_treasury.rs` of `v0.14.1` of https://github.com/neonlabsorg/neon-evm/

Methodology

Neodyme's audit team performed a comprehensive examination of the three contracts. The team, which consists of security engineers with extensive experience in Solana smart contract security, reviewed and tested the code of the on-chain contracts, paying particular attention to the following:

- Ruling out common classes of Solana contract vulnerabilities, such as:
  - Missing ownership checks,
  - Missing signer checks,
  - Signed invocation of unverified programs,
  - Solana account confusions,
  - Re-initiation with cross-instance confusion,
  - Missing freeze authority checks,
  - Insufficient SPL token account verification,
  - Missing rent exemption assertion,
  - Casting truncation,
  - Arithmetic over- or underflows,
  - Numerical precision errors.
- Checking for unsafe design that might lead to common vulnerabilities being introduced in the future,
- Checking for any other, as-of-yet unknown classes of vulnerabilities arising from the structure of the Solana blockchain,
- Ensuring that the contract logic correctly implements the project specifications,
- Examining the code in detail for contract-specific low-level vulnerabilities,
- Ruling out denial-of-service attacks,
- Ruling out economic attacks,
- Checking for instructions that allow front-running or sandwiching attacks,
- Checking for rug-pull mechanisms or hidden backdoors.

In addition, we created an overview of the planned governance and the respective permissions of each component.

Architecture Overview

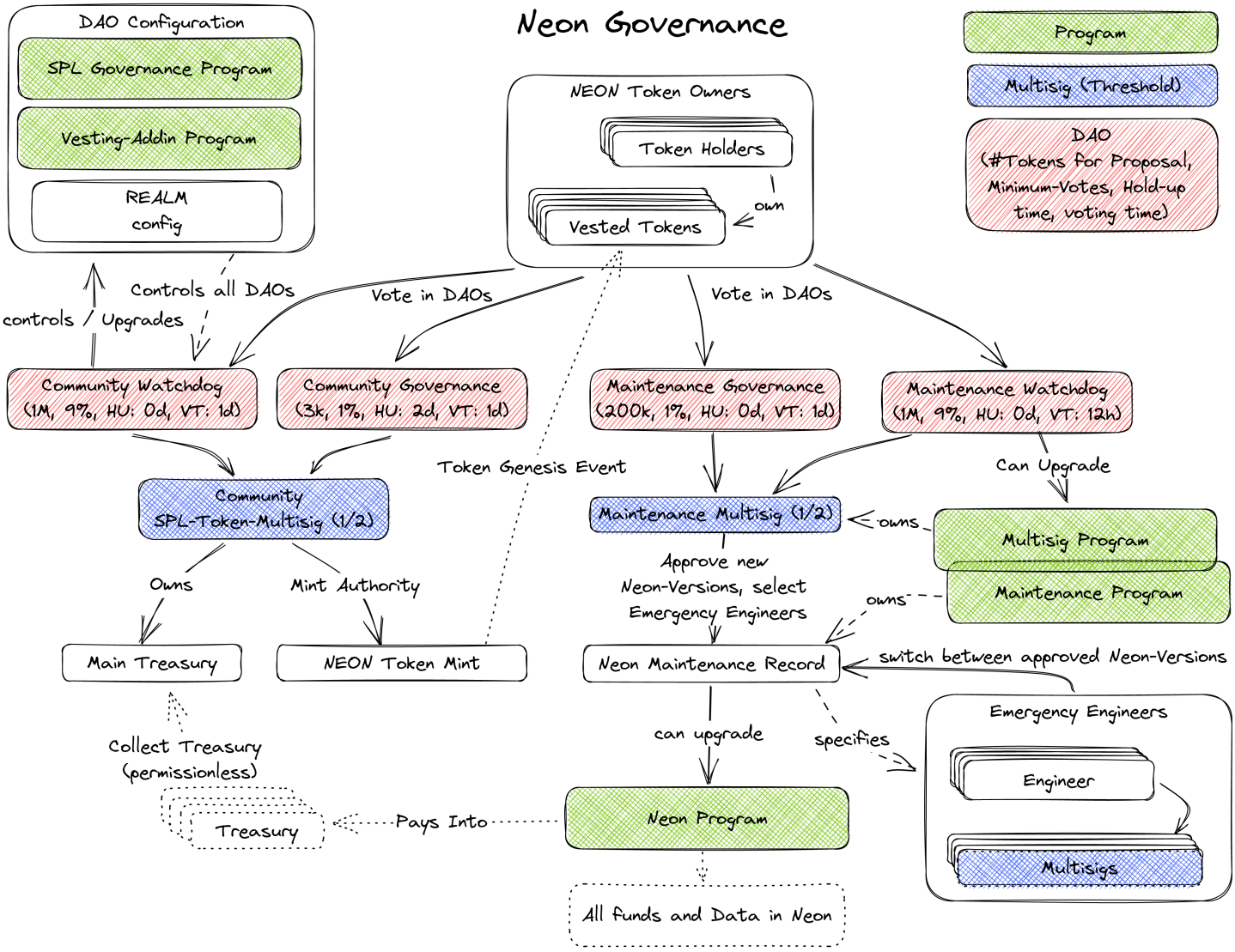

Figure 1: Neon Governance

The planned Neon governance system has a lot of moving parts. At first, to set everything up correctly, the launch scripts deploy all required contracts and mints. In multiple following phases, which can be separated by a time controlled by the neon team, control over everything is transferred over to a DAO system.

In this system, a governance token (NEON) will be used to vote on proposals. Those tokens are held by token owners, either in liquid or vested form. Vesting time and unlock are controlled by the vesting addin program.

We have audited all code required to run this system but have not evaluated the concrete parameters, vote thresholds or voting-times. DAOs are still fairly new territory, and parameter choice largely depends on the value of the governance tokens, market availability of the tokens, and how much funds they control. As such, these parameters might have to change over time.

See the flowgraph above to see the entire ownership chain in the Neon Governance as set-up by the final phase of the launch-script.

The Neon Program itself is under the control of a dedicated *Maintenance Program*. This maintenance program has a set of Engineers (which can be multisigs). Each Engineer can change the Neon program to a pre-determined set of approved versions. This allows for quick reactions to any potential issues when up/downgrading the program or if security issues are found. At the same time, it prevents an Engineer from upgrading to a potentially malicious version of Neon on his own.

The approved versions of the Neon contract are stored in a *Maintenance Record*. This record can only be changed by the *Maintenance Multisig*, which is in turn controlled by two DAOs.

The two DAOs are essentially the same and use the same Governance Token but can use different parameters. That allows having a “slow-moving” DAO, with lower requirements on the minimum number of votes for a proposal to pass, in exchange for longer voting times. This slower DAO is the “normal” governance. But there also is a watchdog governance, with higher token and vote requirements but faster voting time. This latter watchdog DAO also controls the *Maintenance Program* and the *Multisig Program* themselves.

Upgrading those to a malicious version would also lead to full control of the main Neon program, so it is required that they are also under DAO control.

Something similar happens with the treasury the Neon program pays fees into and the main Neon token mint. Both are owned by a Community Token Multisig, which in turn is controlled by two DAOs with the same quick/slow, normal/watchdog arrangement as the Maintenance DAOs.

Finally, there is the DAO program and configuration itself, which is under the control of the Community Watchdog DAO.

In total, there are 4 DAOs, each with potentially different parameters but all controlled by the same tokens.

Findings

All findings are classified in one of four severity levels:

- **Critical**: Bugs that will likely cause loss of funds. This means that an attacker can, with little or no preparation, trigger them, or they are expected to happen accidentally. Effects are difficult to undo after they are detected.
- **High**: Bugs that can be used to set up loss of funds in a more limited capacity, or to render the contract unusable.
- **Medium**: Bugs that do not cause direct loss of funds but lead to other exploitable mechanisms.
- **Low**: Bugs that do not have a significant immediate impact and could be fixed easily after detection.

We have found no significant vulnerabilities in all audited components, only one low-severity bug.

LOW: The last approver can steal lamports (rent) of proposal account [resolved]

Severity	Impact	Affected Component
Low	Proposer of a Transaction does not get Rent back	Executed Transaction Cleanup

When an approver causes a proposal to get executed, the proposal account is zero-filled and emptied afterward. The lamports in the proposal account will be transferred to the "protected account". However, there is no check that the approver-supplied key actually belongs to the protected account. This allows an approver to steal the lamports in the proposal account.

Neon could enforce a transfer back to the proposal author, since the author initially paid for the proposal account. Or verify that the protected account matches the expected account.

The impact of this bug is low because the proposal account will only have the required rent unless the proposal author transfers more than necessary.

Fix Fixed in commit `d4a8b523e5af7e118d2a6f63019df74b7da627f5` by transferring the rent always to the correct protected account.

Neodyme AG

Dirnismaning 55
Halle 13
85748 Garching
E-Mail: contact@neodyme.io

https://neodyme.io