Anchor V0.32

Security Assessment

September 24th, 2025 — Prepared by OtterSec

Jamie Hill-Daniel jamie@osec.io

Robert Chen r@osec.io

Table of Contents

01 — Executive Summary

Overview

Solana foundation engaged OtterSec to assess the `anchor-V0.32` program. This assessment was conducted between August 1st and September 21st, 2025.

Key Findings

We produced 2 findings throughout this audit engagement.

In particular, we highlighted certain engineering concerns regarding prematurely removing commands or enabling stable builds without MSRV enforcement, as it risks breaking user workflows and results in inconsistent builds (OS-ANC-SUG-00). We also suggested highlighting the behavioral change in the release notes when changing IDL upload to be automatic by default (OS-ANC-SUG-01).

Scope

The source code was delivered to us in a git repository at https://github.com/solana-foundation/anchor.

A brief description of the program is as follows:

Name	Description
anchor-V0.32	Review of the anchor-V0.32 program, including PRs that remove the anchor publish command, remove solang, enable building IDLs with stable Rust, and change the default Anchor behavior to automatically upload the IDL, which was previously optional.

02 — General Findings

Here, we present a discussion of general findings during our audit. While these findings do not present an immediate security impact, they represent anti-patterns and may result in security issues in the future.

ID	Description
OS-ANC-SUG-00	There are certain engineering concerns regarding prematurely removing commands or enabling stable builds without MSRV enforcement, as it risks breaking user workflows and results in inconsistent builds.
OS-ANC-SUG-01	**PR#3863** changes IDL upload to be automatic by default, potentially impacting existing workflows.

Potential Code Breaking Changes OS-ANC-SUG-00

Description

It will be appropriate to hold-off on merging PR#3795. While the change only removes dead code related to the `anchor publish` command, it is still a breaking change from a versioning perspective, and removing it before a major release may unintentionally break compatibility. Defer this removal until the Anchor version 1.0.

Similarly, refrain from merging PR#3824, which removes `solang` until version 1.0, as it is also a breaking change. Furthermore, PR#3842 enables IDL builds on stable Rust. While this is beneficial, without a declared Minimum Supported Rust Version (MSRV), users may attempt builds on arbitrary stable compilers. PR#3873 addresses this by locking the Rust template to a supported version, ensuring predictable behavior. Thus, ensure `PR#3842` is merged along with `PR#3873` to prevent exposing users to breakage across different compiler versions.

Remediation

Implement the suggestions mentioned above.

Code Documentation OS-ANC-SUG-01

Description

PR#3863 changes the default behavior of Anchor by automatically uploading the IDL, whereas previously this step was optional. This may result in unexpected uploads in existing workflows, if users are unaware of the change. To avoid confusion or unintended consequences, the new default should be clearly documented in the release notes, along with instructions on how to skip the upload when needed.

Remediation

Ensure to highlight the behavioral change in the release notes.

03 — Vulnerability Rating Scale

We rated our findings according to the following scale. Vulnerabilities have immediate security implications. Informational findings may be found in the General Findings.

CRITICAL

Vulnerabilities that immediately result in a loss of user funds with minimal preconditions.

Examples:

- Misconfigured authority or access control validation.
- Improperly designed economic incentives leading to loss of funds.

HIGH

Vulnerabilities that may result in a loss of user funds but are potentially difficult to exploit.

Examples:

- Loss of funds requiring specific victim interactions.
- Exploitation involving high capital requirement with respect to payout.

MEDIUM

Vulnerabilities that may result in denial of service scenarios or degraded usability.

Examples:

- Computational limit exhaustion through malicious input.
- Forced exceptions in the normal user flow.

LOW

Low probability vulnerabilities, which are still exploitable but require extenuating circumstances or undue risk.

Examples:

- Oracle manipulation with large capital requirements and multiple transactions.

INFO

Best practices to mitigate future security risks. These are classified as general findings.

Examples:

- Explicit assertion of critical internal invariants.
- Improved input validation.