Security Assessment Report

Roosting v1.0.0

December 12th, 2022

Summary

The sec3 team (formerly Soteria) was engaged to do a thorough security analysis of the Roosting Solana smart contract program. The artifact of the audit was the source code of the following on-chain smart contract excluding tests in a private repository.

The audit was done on the following contract

- **Contract "roosting":**
    - Commit `6a02382bd5b50d2490493e11f92f02555cd99fa6`

The audit revealed 3 issues or questions. This report describes the findings and resolutions in detail.

Table of Contents

Methodology and Scope of Work 3

Result Overview 4

Findings in Detail 5

[I-1] The contract accepts arbitrary NFT 5

[I-2] Accounts mint, metadata and edition don't have to be mutable 6

[I-3] No need to save the PDA bump in the RoostingAccount 8

Methodology and Scope of Work

The sec3 (formerly Soteria) audit team, which consists of Computer Science professors and industrial researchers with extensive experience in Solana smart contract security, program analysis, testing and formal verification, performed a comprehensive manual code review, software static analysis and penetration testing.

Assisted by the sec3 Scanner developed in-house, the audit team particularly focused on the following work items:

- Check common security issues.
    - Missing ownership checks
    - Missing signer checks
    - Signed invocation of unverified programs
    - Solana account confusions
    - Arithmetic over- or underflows
    - Numerical precision errors
    - Loss of precision in calculation
    - Insufficient SPL-Token account verification
    - Missing rent exemption assertion
    - Casting truncation
    - Did not follow security best practices
    - Outdated dependencies
    - Redundant code
    - Unsafe Rust code
- Check program logic implementation against available design specifications.
- Check poor coding practices and unsafe behavior.
- The soundness of the economics design and algorithm is out of scope of this work

Result Overview

In total, the audit team found the following issues.

CONTRACT ROOSTING V1.0.0

Issue	Impact	Status
[I-1] The contract accepts arbitrary NFT	Informational	Resolved
[I-2] Accounts mint, metadata and edition don't have to be mutable	Informational	Resolved
[I-3] No need to save the PDA bump in the RoostingAccount	Informational	Resolved

Findings in Detail

IMPACT – INFO

[I-1] The contract accepts arbitrary NFT

The contract doesn't restrict the types of NFTs accepted by the contract. As it's not clear the benefits of being accepted by this contract (as the logic of that part is not in this contract), it's unclear if accepting garbage NFTs can introduce actual damages.

Depending on the usage scenarios, it's recommended to further validate NFTs in the backend or the contract referring to these NFTs to make sure they are acceptable.

Resolution

The team is aware of this issue and treats it as a feature:

> "While the official use for the contract is for our roosting rewards, we also want to in the future allow users to roost their other NFTs as a means of protection from generic wallet drainers. Regarding garbage NFTs, the portion of our backend services that handles roosting rewards will verify that the roosted NFT is a valid MechaFightClub NFT, so no money can be made from invalid NFTs."

IMPACT – INFO

[I-2] Accounts mint, metadata and edition don't have to be mutable

```
/* programs/roosting/src/instructions/roost_nft.rs */
074 | pub struct RoostNft<'info> {
078 |     #[account(mut, mint::decimals = 0)]
079 |     pub mint: Account<'info, Mint>,
089 |     #[account(mut,
094 |     )]
095 |     pub metadata: UncheckedAccount<'info>,
098 |     #[account(mut,
103 |     )]
104 |     pub edition: UncheckedAccount<'info>,
118 | }

/* programs/roosting/src/instructions/unroost_nft.rs */
057 | pub struct UnroostNft<'info> {
061 |     #[account(mut, mint::decimals = 0)]
062 |     pub mint: Account<'info, Mint>,
072 |     #[account(mut,
077 |     )]
078 |     pub metadata: UncheckedAccount<'info>,
081 |     #[account(mut,
086 |     )]
087 |     pub edition: UncheckedAccount<'info>,
103 | }
```

The `mint`, `metadata` and `edition` accounts don't require write permissions. Since they are only used in `freeze_delegated_account()` and `thaw_delegated_account()` in the contract, it's still safe.

```
/* mpl-token-metadata-1.6.1/src/instruction.rs */
1386 | pub fn freeze_delegated_account(
1387 |     program_id: Pubkey,
1388 |     delegate: Pubkey,
1389 |     token_account: Pubkey,
1390 |     edition: Pubkey,
1391 |     mint: Pubkey,
1392 | ) -> Instruction {
1393 |     Instruction {
1394 |         program_id,
1395 |         accounts: vec![
1396 |             AccountMeta::new(delegate, true),
1397 |             AccountMeta::new(token_account, false),
1398 |             AccountMeta::new_readonly(edition, false),
```

```
1399 |             AccountMeta::new_readonly(mint, false),
1400 |             AccountMeta::new_readonly(spl_token::id(), false),
1401 |         ],

1418 | pub fn thaw_delegated_account(
1419 |     program_id: Pubkey,
1420 |     delegate: Pubkey,
1421 |     token_account: Pubkey,
1422 |     edition: Pubkey,
1423 |     mint: Pubkey,
1424 | ) -> Instruction {
1425 |     Instruction {
1426 |         program_id,
1427 |         accounts: vec![
1428 |             AccountMeta::new(delegate, true),
1429 |             AccountMeta::new(token_account, false),
1430 |             AccountMeta::new_readonly(edition, false),
1431 |             AccountMeta::new_readonly(mint, false),
1432 |             AccountMeta::new_readonly(spl_token::id(), false),
1433 |         ],
```

Resolution

The team acknowledged this finding.

IMPACT – INFO

[I-3] No need to save the PDA bump in the RoostingAccount

```
/* programs/roosting/src/instructions/unroost_nft.rs */
057 | pub struct UnroostNft<'info> {
089 |     #[account(
090 |         mut,
091 |         seeds = [RoostingAccount::SEED, mint.key().as_ref()],
092 |         bump = roosting_account.bump,
097 |     )]
098 |     pub roosting_account: Account<'info, RoostingAccount>,
103 | }
```

The current approach is sound and safe. However, the contract doesn't have to save the bump in the PDA. If no bump is provided, Anchor will use the canonical bump.

```
#[account(
    mut,
    seeds = [RoostingAccount::SEED, mint.key().as_ref()],
    bump,
)]
pub roosting_account: Account<'info, RoostingAccount>,
```

Anchor will use `find_program_address()` to get the canonical PDA and bump.

```
let (__pda_address, __bump) = Pubkey::find_program_address(
    &[RoostingAccount::SEED, mint.key().as_ref()],
    &program_id,
);
__bumps.insert("roosting_account".to_string(), __bump);
if roosting_account.key() != __pda_address {
    return Err(
        anchor_lang::error::Error::from(
                anchor_lang::error::ErrorCode::ConstraintSeeds,
            )
            .with_account_name("roosting_account")
            .with_pubkeys((roosting_account.key(), __pda_address)),
    );
}
```

Resolution

The team acknowledged this finding.

DISCLAIMER

The instance report ("Report") was prepared pursuant to an agreement between Coderrect Inc. d/b/a sec3 (the "Company") and Irreverent Labs, Inc. (the "Client"). This Report solely includes the results of a technical assessment of a specific build and/or version of the Client's code specified in the Report ("Assessed Code") by the Company. The sole purpose of the Report is to provide the Client with the results of the technical assessment of the Assessed Code. The Report does not apply to any other version and/or build of the Assessed Code. Regardless of the contents of the Report, the Report does not (and should not be interpreted to) provide any warranty, representation or covenant that the Assessed Code: (i) is error and/or bug free, (ii) has no security vulnerabilities, and/or (iii) does not infringe any third-party rights. Moreover, the Report is not, and should not be considered, an endorsement by the Company of the Assessed Code and/or of the Client. Finally, the Report should not be considered investment advice or a recommendation to invest in the Assessed Code and/or the Client.

This Report is considered null and void if the Report (or any portion thereof) is altered in any manner.

ABOUT

Founded by leading academics in the field of software security and senior industrial veterans, sec3 (formerly Soteria) is a leading blockchain security company that currently focuses on Solana programs. We are also building sophisticated security tools that incorporate static analysis, penetration testing, and formal verification.

At sec3, we identify and eliminate security vulnerabilities through the most rigorous process and aided by the most advanced analysis tools.

For more information, check out our website and follow us on twitter.

sec3™