Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.12.03, the SlowMist security team received the team's security audit application for particle-universal-account-solana, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Reentrancy Vulnerability
- Replay Vulnerability
- Reordering Vulnerability
- Denial of Service Vulnerability
- Transaction Ordering Dependence Vulnerability
- Race Conditions Vulnerability
- Authority Control Vulnerability
- Integer Overflow and Underflow Vulnerability
- TimeStamp Dependence Vulnerability
- Unsafe External Call Audit
- Design Logic Audit
- Scoping and Declarations Audit
- Account substitution attack Audit
- Malicious Event Log Audit

3 Project Overview

3.1 Project Introduction

The L1 Unifying All Chains Through Universal Accounts.

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	`environment` parameter forgery	Authority Control Vulnerability Audit	Low	Acknowledged
N2	Missing signature expiration time in `update_metadata_validate_mode` function	Design Logic Audit	Medium	Fixed

4 Code Overview

4.1 Contracts Description

https://github.com/Particle-Network/particle-universal-account-solana

Audit commit: 63eeabe1f5528eb0ab69b0375cdce919fb934eba

Patch PRs: https://github.com/Particle-Network/particle-universal-account-solana/pull/1

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist security team analyzed the visibility of major contracts during the audit, the result as follows:

universal			
Function Name	Account check coverage	Auth Signer	Params Check
handle_user_operation	4/4	payer	7/7
update_metadata_validate_mode	3/3	payer	4/4

4.3 Vulnerability Summary

[N1] [Low] `environment` parameter forgery

Category: Authority Control Vulnerability Audit

Content

In the `handle` method, call permissions are controlled by checking the key of the payer.

However, the `args.account_index` parameter is used to indicate whether the key is used in the test environment or the production environment, then the test environment key can be invoked against the main network contract by modifying the `args.account_index` to zero.

```
pub fn handle(
        ctx: Context<'_, '_, '_, '_, UserOperation<'info>>,
        args: UserOperationArgs,
    ) -> Result<()> {
        //...
        check_control_key(&ctx.accounts.payer, metadata, args.account_index)?;
```

- particle-universal-account-solana/programs/particle-universal-account-solana/src/instructions/entry_point.rs

```
pub fn check_control_key<'info>(
    payer: &Signer<'info>,
    metadata: &mut Account<'info, Metadata>,
    account_index: u8,
) -> Result<()> {
    //...
    if account_index == 0 {
        require!(
            payer.key() == CONTROL_KEY_DEBUG,
            EntryPointError::InvalidControlKey
        );
    } else {
        require!(
            payer.key() == CONTROL_KEY_PRODUCTION,
            EntryPointError::InvalidControlKey
        );
    }
```

```
    Ok(())
}
```

Solution

Instead of using parameter passing, parameters are used at compile time to distinguish between test and production nets.

Status

Acknowledged

[N2] [Medium] Missing signature expiration time in `update_metadata_validate_mode` function

Category: Design Logic Audit

Content

In the `update_metadata_validate_mode` method, a secp256k1 signature is used to verify address ownership, but the content of this signature does not include an expiration time, which means that a signed transaction can be used at any time in the future with unintended consequences if it is not invoked successfully immediately.

- particle-universal-account-solana/programs/particle-universal-account-solana/src/instructions/entry_point.rs

```
pub struct UpdateMetadataValidateModeArgs {
    // evm address
    pub credential_id: [u8; 20],
    pub nonce: u64,
    pub validate_mode: u8,
    pub signature: Vec<u8>,
}
//...
    pub fn update_metadata_validate_mode(
        //...
        verify_ecdsa_signature(
            metadata_hash,
            full_signature,
            ctx.accounts.metadata.validate_mode,
            ctx.accounts.metadata.credential_id,
        )?;
```

Solution

Add the `expired_at` field and limit the timeframe to for.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002412060003	SlowMist Security Team	2024.12.03 - 2024.12.06	Low Risk

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 medium risk, 1 low risk vulnerabilities.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist