Security Audit – Jito Merkle Distributor

Lead Auditor: Thomas Lambertz

December 5th 2023

Table of Contents

Executive Summary

Neodyme audited **Jito's Merkle-Distributor** on-chain program during early December 2023.

This is an abridged report only listing findings. According to Neodymes Rating Classification, **two informational issues** and **no security vulnerabilities** were found. In addition to these findings, Neodyme delivered the Jito team a list of nit-picks that are not part of this report. The number of findings identified throughout the audit, grouped by severity, can be seen in Figure 1.

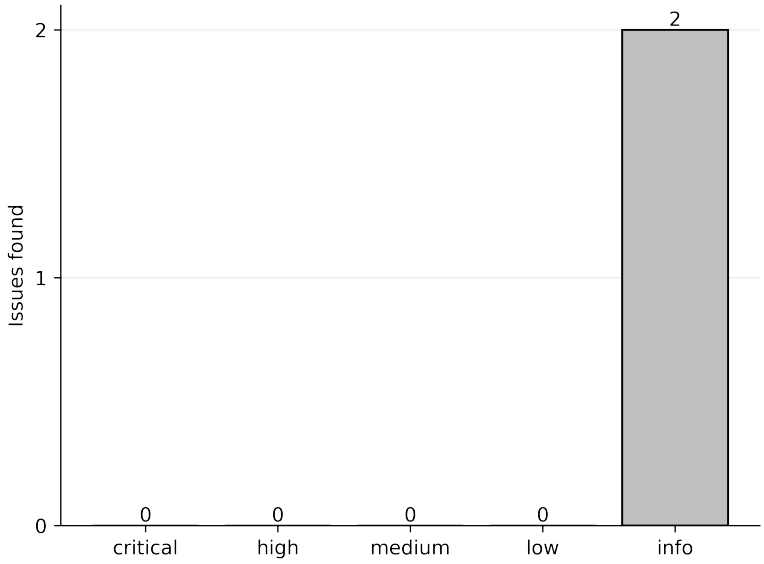

Figure 1: Overview of Findings

1 Introduction

In early December of 2023, Jito commissioned Neodyme to conduct a detailed security analysis of their new Merke-distributor contract. The audit was performed between November 30th and December 1st, 2023.

The audit mainly focused on the contract's technical security but also considered its design and potential social engineering attack vectors. This is an **abridged report** that only details findings.

Findings Summary

During the audit, **two informational** findings were identified. Jito remediated both findings before the contract's launch.

The first informational finding addresses a potential frontrunning vector against distributor initialization; the second one discusses that log-based Anchor events are fallible.

2 Scope

The contract audit's scope comprised of the on-chain components of the Merkle-distributor smart-contract developed by Jito. All of the source code, located at https://github.com/jito-foundation/distributor/tree/master/programs/merkle-distributor, including the `jito-merkle-verify` dependency is in scope of this audit. However, third-party dependencies are not.

Relevant source code revisions are:

- `0e66f1c52b6ecac6e39b82788e3c5326608168b4` • Start of the audit
- `48cf51e3cc117b44fcf6d29270d4bbc247818e84` • Commit including all remediations

3 | Findings

This section outlines all of our findings. They are classified into one of five severity levels, detailed in Appendix C. In addition to these findings, Neodyme delivered the Audius team a list of nit-picks and additional notes that are not part of this report.

All findings are listed in Table 1 and further described in the following sections.

Table 1: Findings

Identifier	Name	Severity	State
ND-JIT1-I1	Risk of frontrunning during distributor creation	Informational	Remediated
ND-JIT1-I2	Logs can be truncated, and events might be omitted	Informational	Remediated

ND-JIT1-I1 – Risk of frontrunning during distributor creation

Severity	Impact	Affected Component	Status
Info	Creator might use a distributor an attacker configured	Distributor Creation	Remediated

The design of `NewDistributor` allows for an unlikely frontrunning attack, because creation is permissionless:

- A legitimate user opens a new distribution
- An attacker observes this transaction before it is executed, for example, by being the RPC node, the current leader, or via MEV.
- An attacker front-runs the creation with the same version and mint, but a different admin and times.
- The user will see his original transaction fail, as the distributor has already been initialized.
- The user might simply blame unstable internet/RPC, and not check the parameters, and keep using this distributor.

The current CLI will panic on an `unwrap()` call, but doesn't explicitly check parameters anywhere.

In practice, this attack is quite unlikely due to the requirement to front-run the transaction, though there is a variant where an attacker can predict that a new distribution will be created for a known mint, in which case he would have more time to front-run.

Suggestion

There are two ways to handle this:

- A: Modify the program to not use a version, but a "seed" account that has to sign creation.
- B: Modify the CLI to throw an explanatory warning instead of a panic when the creation fails, and maybe add a way to view the current parameters so a user can double-check them.

Remediation

The Jito team implemented a version of B in `48cf51e3cc117b44fcf6d29270d4bbc247818e84`. The CLI now has a check that a potentially existing on-chain distributor is read back to ensure the parameters match with the one the user wants to create. In addition, documentation was added to

the source code of the contract, which also gets included in the IDL. This makes sure developers of frontends that might use this contract in the future are aware.

ND-JIT1-I2 – Logs can be truncated, and events might be omitted

Severity	Impact	Affected Component	Status
Info	Emitted events might be dropped and not observable	Event Logging	Remediated

The distributor uses Anchor's `emit!` event feature. This logs the event with `sol_log_data`, which is affected by log-truncation issues. Each validator or RPC node can freely choose how much logs it wants to keep. In practice, this is usually the default of 10k bytes. In this distributor's case, logs are fairly small. They can only overflow if included with other programs in the same instructions, which log a lot. This could be triggered by an attacker at will, but only on his own claims.

Suggestion

Anchor has an `emit-cpi` feature, which uses CPI instead of program logs to emit events, which are always fully stored, but also limited in size. This doesn't impact the security of the contract itself at all, but is important to keep in mind when designing backend-infrastructure that makes decisions based on the emitted events.

Remediation

The Jito team does not use the events for decisions in the backend and will keep this issue in mind. They added comments to all logs stating that they should not be relied on in `48cf51e3cc117b44fcf6d29270d4bbc247818e84`.

A | About Neodyme

Security is difficult.

To understand and break complex things, you need a certain type of people. People who thrive in complexity, who love to play around with code, and who don't stop exploring until they fully understand every aspect of it. That's us.

Our team never outsources audits. Having found over 80 High or Critical bugs in Solana's core code itself, we believe that Neodyme hosts the most qualified auditors for Solana programs. We've also found and disclosed critical vulnerabilities in many of Solana's top projects and have responsibly disclosed issues that could have resulted in the theft of over $10B in TVL on the Solana blockchain.

All of our team members have a background in competitive hacking. During such hacking competitions, called CTFs, we competed and collaborated while finding vulnerabilities, breaking encryption, reverse engineering complicated algorithms, and much more. Through the years, many of our team members have won national and international hacking competitions, and keep ranking highly among some of the hardest CTF events world-wide. In 2020, some of our members started experimenting with validators and became active members in the early Solana community. With the prospect of an interesting technical challenge and bug bounties, they quickly encouraged others from our CTF team to look for security issues in Solana. The result was so successful that after reporting several bugs, in 2021, the Solana Foundation contracted us for source code auditing. As a result, Neodyme was born.

B | Methodology

Neodyme prides itself on not being a checklist auditor. We adapt our approach to each audit, investing considerable time into understanding the program upfront and exploring its expected behavior, edge cases, invariants, and ways in which the latter could be violated. We use our uniquely deep knowledge of Solana internals, and our years-long experience in auditing Solana programs to even find bugs that others miss. We often extend our audit to cover off-chain components, in order to see how users could be tricked or the contract affected by bugs in those components.

Nonetheless, we also have a list of common vulnerability classes, which we always exhaustively look for. We provide a sample of this list below.

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

C | Vulnerability Severity Rating

Critical Vulnerabilities that will likely cause loss of funds. An attacker can trigger them with little or no preparation, 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 that may lead to other exploitable mechanisms, or that could be exploited to render the contract partially unusable.

Low Bugs that do not have a significant immediate impact and could be fixed easily after detection.

Info Bugs or inconsistencies that have little to no security impact.

Neodyme AG

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

https://neodyme.io