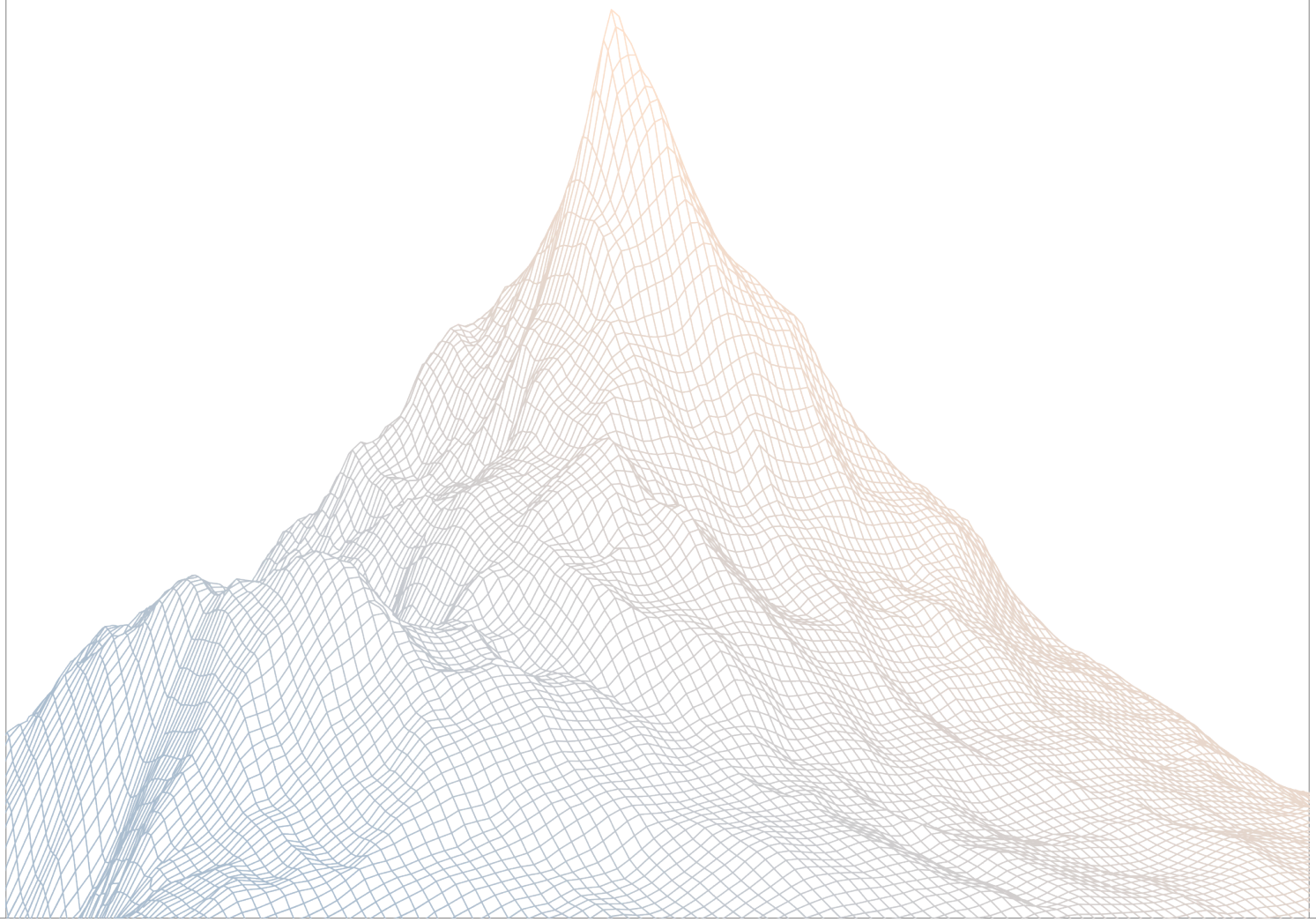
Zenith

LootGO

Smart Contract Security Assessment

VERSION 1.1



AUDIT DATES: November 7th to November 10th, 2025

AUDITED BY: adriro
montecristo

Contents

1 Introduction

1.1 About Zenith

Zenith assembles auditors with proven track records: finding critical vulnerabilities in public audit competitions.

Our audits are carried out by a curated team of the industry's top-performing security researchers, selected for your specific codebase, security needs, and budget.

Learn more about us at https://zenith.security.

1.2 Disclaimer

This report reflects an analysis conducted within a defined scope and time frame, based on provided materials and documentation. It does not encompass all possible vulnerabilities and should not be considered exhaustive.

The review and accompanying report are presented on an "as-is" and "as-available" basis, without any express or implied warranties.

Furthermore, this report neither endorses any specific project or team nor assures the complete security of the project.

1.3 Risk Classification

SEVERITY LEVEL	IMPACT: HIGH	IMPACT: MEDIUM	IMPACT: LOW
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

2

Executive Summary

2.1 About LootGO

LootGO is a walk-to-earn mobile game that turns your steps into a real-world crypto treasure hunt.

Think: Pokémon GO, but with crypto rewards.

2.2 Scope

The engagement involved a review of the following targets:

Target	lootgo-contracts
Repository	https://github.com/Loot-Go/lootgo-contracts
Commit Hash	`66ed9a974d05a4e3f74a51856a814c890acf006e`
Files	`contracts/**/*.sol`

2.3 Audit Timeline

November 7, 2025	Audit start
November 10, 2025	Audit end
November 10, 2025	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	1
Informational	4
Total Issues	**5**

3

Findings Summary

ID	Description	Status
L-1	fallback() function is not needed	Resolved
I-1	Introduce a pauser role	Resolved
I-2	Validate duplicate recipients	Resolved
I-3	Unnecessary payable casts	Resolved
I-4	Use custom errors instead of string error messages	Resolved

4

Findings

4.1 Low Risk

A total of 1 low risk findings were identified.

[L-1] `fallback()` function is not needed

SEVERITY: Low	IMPACT: Low
STATUS: Resolved	LIKELIHOOD: Low

Target

- LootDrop.sol

Description:

The `fallback()` function is not needed, as `receive()` allows receiving native payments, and could lead to potential accidents since the contract wouldn't revert to a call with unrecognized calldata.

Recommendations:

Remove the `fallback()` function.

LootGo: Resolved with `@340c1ca112...`.

Zenith: Verified.

4.2 Informational

A total of 4 informational findings were identified.

[I-1] Introduce a pauser role

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- LootDrop.sol

Description:

The `pause()` function is currently restricted to the `DEFAULT_ADMIN_ROLE`. It would be convenient to split this functionality into a separate role that can monitor and act in case of an emergency, separate from an account that has full admin privileges.

Recommendations:

Consider adding a pauser role that can call `pause()` from an account without root-level access.

LootGo: Resolved with `@340c1ca112...`.

Zenith: Verified.

[I-2] Validate duplicate recipients

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- LootDrop.sol

Description:

The `batchTransferSingleToken()` and `batchSendETH()` functions execute a multi-send to different receivers whose input is an array that may contain duplicate addresses.

Recommendations:

Validate there are no duplicates in the `recipients` array to prevent accidental transfers. This can be done efficiently by sending the array sorted, and validating on-chain that `recipients[i-1] < recipients[i]` during the iteration.

LootGo: Resolved with `@340c1ca112 ...`.

Zenith: Verified.

[I-3] Unnecessary payable casts

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- LootDrop.sol

Description:

Following `payable` casts are unnecessary and can be removed:

File: contracts/LootDrop.sol

```
174:     function sendETH(address payable to, uint256 amount)
...
215:             (bool success, ) = payable(recipients[i]).call{value:
    amounts[i]}("");
...
250:     function adminWithdrawETH(address payable to, uint256 amount)
```

According to forge testing, deploy cost can be reduced from `2780093` to `2768275` by removing `payable` casts.

Recommendations:

Remove unncessary `payable` conversions.

LootGo: Resolved with `@340c1ca112 ...`.

Zenith: Verified.

[I-4] Use custom errors instead of string error messages

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- LootDrop.sol

Description:

Throughout the codebase, strings are used for revert reason.

However, string errors are rather expensive in terms of deploy cost, and it is difficult to pass dynamic information:

File: contracts/LootDrop.sol

```
159:             require(bal >= amt, string(abi.encodePacked("LootDrop:
    insufficient balance for token ", tokenAddr.toHexString())));
 ```


It'll be more clearer and efficient, if we do the following instead:

```solidity
require(bal >= amount, InsufficientBalance(tokenAddr));
```

Recommendations:

Use custom errors instead of strings.

LootGO: Resolved with @340c1ca112 ... .

Zenith: Verified.