GoldFinger Audit Report

January 2026

GoldFinger Audit Report

Zero Drift

January 2026

Prepared by: Zero Drift

Table of Contents

About Zero Drift

Zero Drift is a tech-driven Web3 auditing firm dedicated to zero-incident smart-contract assurance for mission-critical protocols.

Our team consists of several senior smart-contract auditors, each with 5+ years of experience and a perfect post-audit record across more than 80 successful reviews, collectively helping protect over $5 billion in on-chain assets.

Disclaimer

The Zero Drift team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the smart-contract implementation of the contracts.

Risk Classification

	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Protocol Overview

GoldFinger is a platform that integrates Real World Assets (RWA) with Decentralized Finance (DeFi). It focuses on bringing physical gold on-chain, allowing for the tokenization of gold to enable secure, transparent yield generation in Web3 applications.

The CheckIn contract implements a time-limited campaign check-in mechanism. Users can perform daily check-ins during a defined campaign period, earning rewards for consistent participation. The contract tracks user check-in history relative to a configurable `startTime` and enforces a maximum campaign duration (`CAMPAIGN_DAYS`).

Audit Scope

The scope consisted of a focused review of the CheckIn contract logic and campaign validation mechanisms.

- Files reviewed: `CheckIn.sol`

Executive Summary

Over the course of the review, the Zero Drift team examined the CheckIn functionality and associated patterns. We identified several issues across Medium and Low severities. The project team has addressed all identified issues; each finding below is marked Resolved.

Summary

Project Name	GoldFinger
Audit Timeline	Jan 2026
Methods	Manual Review

Issues Found

	Count
Critical Risk	0
High Risk	0
Medium Risk	1
Low Risk	2
Informational	0
Gas Optimizations	0
Total Issues	**3**

Summary of Findings

ID	Title	Severity	Status
M-1	Redundant validation in `campaignActive()` function	Medium	Resolved
L-1	Missing input validation in constructor	Low	Resolved
L-2	Missing use of `canCheckInToday()` in `checkIn()` function	Low	Resolved

Findings

Medium Risk

[M-01] Redundant Validation in `campaignActive()` Function

Description

The `campaignActive()` function contains a redundant check `d >= 1` that will always evaluate to true given the calculation method. The variable `d` is computed as `(block.timestamp - startTime) / 1 days + 1`, which guarantees `d >= 1` for any valid execution path (since the function returns early if `block.timestamp < startTime`).

```
function campaignActive() public view returns (bool) {
    if (block.timestamp < startTime) return false;
    uint256 d = (block.timestamp - startTime) / 1 days + 1;
    return d >= 1 && d <= CAMPAIGN_DAYS;  // d >= 1 is always true
}
```

Impact

Unnecessary gas consumption on every call to `campaignActive()`. The redundant condition reduces code clarity and may mislead future maintainers about the function's invariants.

Recommended Mitigation

Remove the redundant check to streamline the function:

```
function campaignActive() public view returns (bool) {
    if (block.timestamp < startTime) return false;
    uint256 d = (block.timestamp - startTime) / 1 days + 1;
    return d <= CAMPAIGN_DAYS;
}
```

Status: Resolved

Low Risk

[L-01] Missing Input Validation in Constructor

Description

The constructor accepts any `_startTime` value without validation. A start time in the past would make the campaign immediately active or potentially already completed at deployment time, depending on how far in the past the value is set.

Impact

Potential deployment with incorrect parameters leading to unexpected campaign state. If `_startTime` is set to a past timestamp, the campaign may already be partially or fully elapsed at the moment of deployment.

Recommended Mitigation

Add validation to ensure the start time is in the future:

```
constructor(uint256 _startTime) {
    require(_startTime > block.timestamp, "Start time must be in the
        future");
    startTime = _startTime;
}
```

Status: Resolved

[L-02] Missing Use of `canCheckInToday()` in `checkIn()` Function

Description

The `checkIn()` function uses `currentDay()` directly instead of leveraging the existing `canCheckInToday()` function for validation. The `canCheckInToday()` function provides a more comprehensive check that ensures both campaign validity and user eligibility.

Impact

Reduced code reusability and potential for inconsistent validation logic. Using dedicated validation functions improves maintainability and ensures consistent checks across the contract.

Recommended Mitigation

Refactor `checkIn()` to use `canCheckInToday()` for validation:

```
function canCheckInToday(address user) public view returns (bool) {
    // validation logic
}

function checkIn() external returns (uint256 earned) {
    require(canCheckInToday(msg.sender), "Cannot check in today");
    // ... rest of checkIn logic
}
```

Status: Resolved