Palminer Audit Report

February 10, 2026

Palminer Audit Report

Zero Drift

February 10, 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

Palminer implements the PalClaimKey smart contract, which manages daily claim limits for key distribution. The contract features an initialization pattern, ownership management, and configurable daily limits with tracking per user address.

Audit Scope

The scope consisted of a focused review of the PalClaimKey contract implementation.

- Files reviewed: `PalClaimKey.sol`

Executive Summary

Over the course of the review, the Zero Drift team examined the PalClaimKey functionality and associated patterns. We identified 1 Medium and 3 Informational issues. The project team has acknowledged all identified issues.

Summary

Project Name	Palminer
Audit Timeline	February 2026
Methods	Manual Review

Issues Found

	Count
Critical Risk	0
High Risk	0
Medium Risk	1
Low Risk	0
Informational	3
Total Issues	**4**

Summary of Findings

ID	Title	Severity	Status
M-1	Incorrect Default Value for dailyLimit defeats rate limiting	Medium	Acknowledged
I-1	Non-Standardized ERC20 Withdrawal Implementation	Informational	Acknowledged
I-2	No Minimum Value Protection for setDailyLimit	Informational	Acknowledged

ID	Title	Severity	Status
I-3	Missing Event in setDailyLimit Function	Informational	Acknowledged

Findings

Medium Risk

[M-01] Incorrect Default Value for dailyLimit

Severity

Impact: High

Likelihood: Medium

Status: Acknowledged

Description

The default value of `dailyLimit` is set to `1e18` (1,000,000,000,000,000,000) in the `initialize()` function at line 69. This extremely large number effectively defeats the purpose of the daily claim limit mechanism, allowing users to claim up to 10^18 times per day.

```
function initialize() external {
    require(!_initialized, "Already initialized");
    _initialized = true;

    _owner = msg.sender;
    dailyLimit = 1e18;  // VULNERABILITY: Incorrect default value

    emit Initialized(1);
    emit OwnershipTransferred(address(0), msg.sender);
}
```

Impact

- Users can claim up to 100 quintillion claims per day, rendering the limit meaningless
- The economic model designed for daily rate limiting is completely broken
- A single user can accumulate an unlimited number of keys
- The fundamental design intent of controlled distribution is undermined

Recommended Mitigation

Change the default value to a reasonable small number that aligns with the intended economic model:

```
dailyLimit = 1;  // Allow 1 claim per day
// OR
dailyLimit = 10; // Allow 10 claims per day, depending on tokenomics
```

Team Response

The development team acknowledges this finding and states: “dailyLimit should be fine. We have a setDailyLimit method that only the owner can call, and we will limit this daily claim amount accordingly.” The team plans to set the appropriate limit through the `setDailyLimit()` function after deployment.

Informational

[I-01] Non-Standardized ERC20 Withdrawal Implementation

Location: PalClaimKey.sol:147-155

Status: Acknowledged

Description

The ERC20 token withdrawal function uses raw `call` instead of the standardized IERC20 interface. This approach has several drawbacks:

- No proper ERC20 interface validation at compile time
- If `token` is a malicious contract, the call may fail silently
- Some non-standard tokens may not return data, making verification logic complex and potentially unreliable
- Reduced code readability and maintainability

```
(bool ok, bytes memory data) = token.call(
    abi.encodeWithSignature("balanceOf(address)", address(this))
);
require(ok && data.length >= 32, "balanceOf failed");
uint256 balance = abi.decode(data, (uint256));

(bool ok2, bytes memory data2) = token.call(
    abi.encodeWithSignature("transfer(address,uint256)", to, balance)
);
require(ok2 && (data2.length == 0 || abi.decode(data2, (bool))), "
   transfer failed");
```

Impact

- Withdrawal may fail without clear error messages
- Increased risk of interaction with non-compliant tokens
- Lower code quality and security guarantees

Recommended Mitigation

Use the IERC20 interface for better type safety and standardization:

```
interface IERC20 {
    function balanceOf(address account) external view returns (uint256);
    function transfer(address to, uint256 amount) external returns (bool);
}

function withdraw(address token, address to) external onlyOwner {
    require(to != address(0), "Zero address");
    if (token == address(0)) {
        (bool ok, ) = to.call{value: address(this).balance}("");
        require(ok, "ETH transfer failed");
    } else {
        uint256 balance = IERC20(token).balanceOf(address(this));
        require(IERC20(token).transfer(to, balance), "Transfer failed");
    }
}
```

[I-02] No Minimum Value Protection for setDailyLimit

Location: PalClaimKey.sol:135-137

Status: Acknowledged

Description

The `setDailyLimit` function lacks validation for minimum values. The owner can set `dailyLimit` to `0`, which would permanently disable the claim functionality for all users without any recovery mechanism beyond calling `setDailyLimit` again.

```
function setDailyLimit(uint256 newLimit) external onlyOwner {
    dailyLimit = newLimit;  // No minimum value validation
}
```

Impact

- Owner can accidentally or intentionally disable all claims
- No safeguard against operational errors

- Users would have no way to claim keys until the limit is reset

Recommended Mitigation

Add minimum value validation to prevent setting the limit to zero:

```
function setDailyLimit(uint256 newLimit) external onlyOwner {
    require(newLimit > 0, "Limit must be greater than zero");
    dailyLimit = newLimit;
}
```

[I-03] Missing Event in setDailyLimit Function

Location: PalClaimKey.sol:135-137

Status: Acknowledged

Description

The `setDailyLimit` function modifies a critical state variable but does not emit an event. This makes it difficult for off-chain applications, users, and monitoring systems to track changes to the daily claim limit in real-time.

```
function setDailyLimit(uint256 newLimit) external onlyOwner {
    dailyLimit = newLimit;
}
```

Impact

- Off-chain systems cannot easily monitor limit changes without polling
- Users are not notified when the daily limit changes
- Blockchain explorers and indexers cannot properly track this critical parameter
- Reduced transparency in governance actions

Recommended Mitigation

Add an event to emit when the daily limit is changed:

```
event DailyLimitChanged(uint256 oldLimit, uint256 newLimit);

function setDailyLimit(uint256 newLimit) external onlyOwner {
    uint256 oldLimit = dailyLimit;
    dailyLimit = newLimit;
    emit DailyLimitChanged(oldLimit, newLimit);
}
```