Palmera Audit Report

Version 1.0

Mahdi Rostami

September 17, 2024

Palmera Audit Report

Mahdi Rostami

September 17, 2024

Prepared by: Mahdi Rostami [Twitter]

Table of Contents

Palmera Security Review

A security review of the Palmera protocol was done by mahdi rostami.
This audit report includes all the vulnerabilities, issues and code improvements found during the security review.

Palmera Overview

Palmera streamlines your Safes operations and treasury management across multiple chains all from a single dashboard.

Disclaimer

"Audits are a time, resource and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to find as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**."

- Secureum

Impact

- **High** Issues that lead to the loss of user funds. Such issues include:
  - Direct theft of any user funds, whether at rest or in motion.
  - Long-term freezing of user funds. Theft or long term freezing of unclaimed yield or other assets.
  - Protocol insolvency
- **Medium** Issues that lead to an economic loss but do not lead to direct loss of on-chain assets. Examples are:
  - Gas griefing attacks (make users overpay for gas)
  - Attacks that make essential functionality of the contracts temporarily unusable or inaccessible.
  - Short-term freezing of user funds.
- **Low** Issues where the behavior of the contracts differs from the intended behavior (as described in the docs and by common sense), but no funds are at risk.

Actions required by severity level

- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

Executive summary

Overview

Project Name	PossumCore
Repository	Link
Commit Hash	d766a22
Docs	Link
Methods	Manual Review

Scope

File
All files in repo

Compatibilities

- Solc Version: 0.8.23

Known Issues

None

Issues found

Severity	Count
High	0
Medium	2
Low	1
Info/Gas	1

Findings

Medium Severity

[M-1] Unbonded orgHash Could Result in Denial of Service (DOS)

Description:

Unbonded orgHash could result in a denial of service (DOS) in several functions, potentially leading to serious issues. The affected functions are: - removeOrg and every function that uses removeOrg: 1. removeWholeTree 2. disconnectSafe

Impact:

Denial of service in core functions, potentially affecting the integrity and usability of the contract. A DOS attack in the removeOrg function can also cause issues when attempting to remove an organization from the list of organization hashes.

Scenario:

An attacker can exploit unbonded orgHash to cause these functions to fail, preventing legitimate users from interacting with the contract. For example:

An attacker creates lots of orgs.. The contract becomes unable to process legitimate orgHash values due to the presence of unbonded values, leading to DOS in the mentioned functions.

Remark:

Fixed. pull39

[M-2] Denial of Service (DoS) Vulnerability in Lead Role Disablement

Description:

When a user attempts to disable the lead role, the system iterates through the `_safeIds` array to remove the associated safe. An attacker can exploit this by increasing the size of the `_safeIds` array to cause a Denial of Service (DoS) during role disablement. This vulnerability stems from the lack of restrictions on adding safes to a user, which could result in extremely large arrays, making it impractical to remove the lead role in a timely manner.

Impact:

An attacker can prevent a target user from disabling their lead roles by inflating the size of the `_safeIds` array, leading to a Denial of Service (DoS). This makes role management difficult and can compromise the governance of an organization.

Scenario:

1. A malicious user creates an organization and adds a large number of safes to it.
2. The malicious user sets a target user as the lead of all these safes.
3. The target user tries to disable the lead role, but the loop over the large `_safeIds` array causes a DoS, making it nearly impossible for the user to remove their lead role.

The vulnerability arises in the following code:

```
if (doesUserHaveRole(safeId, user, role)) {
    currentRole &= ~(bytes32(1 << role));
    _removeElement(_safeIds, safeId); //@audit DOS
    _emit = true;
}
```

The function `_removeElement` loops over the entire array to find and remove the `safeId`:

```
function _removeElement(uint256[] storage array, uint256 element)
   private {
    for (uint256 i; i < array.length;) {
        if (array[i] == element) {
            array[i] = array[array.length - 1];
            array.pop();
            break;
        }
        unchecked {
            ++i;
        }
    }
}
```

So if the attacker makes an array of `_safeIds` for that particular user and role `uint256[] storage _safeIds = safeIdsAndRolesByUser[user][role];` so huge, it creates dos in looping over the array.

Mitigation:

Introduce a function that allows users to approve or restrict who can assign them as a lead for specific safes. This will prevent the array from growing uncontrollably and reduce the risk of DoS attacks. Additionally, consider imposing limits on the size of the `_safeIds` array or using a more efficient data structure for safe role management.

Remark:

Fixed. pull42

Low Severity

[L-1] execTransactionOnBehalf Function Incorrectly Marked as Payable

Description:

The `execTransactionOnBehalf` function is marked as `payable`, which means it can accept Ether. However, the contract is not designed to handle Ether transactions, nor is there any logic in the function to utilize `msg.value`. This could lead to Ether being inadvertently sent to the contract with no way to withdraw it, causing potential loss of funds.

Mitigation:

Remove the `payable` keyword from the `execTransactionOnBehalf` function.

Remark:

Fixed. pull38

Info Gas

- execTransactionOnBehalf doesn't have `requiresAuth` modifier, so assigning this capability to roles is redundant

https://github.com/keyper-labs/priv-PalmeraModule/blob/d766a2293634409f5a45896bc36682dd5eb1a7ac/src/Palme
L66 https://github.com/keyper-labs/priv-PalmeraModule/blob/d766a2293634409f5a45896bc36682dd5eb1a7ac/src/Pa
https://github.com/keyper-labs/priv-PalmeraModule/blob/d766a2293634409f5a45896bc36682dd5eb1a7ac/src/Palme

Fixed. pull40

- function catch msg.sender as a caller, but use msg.sender instead of caller

```
function checkAfterExecution(bytes32, bool) external view {
        address caller = msg.sender;
        // Check if the Palmera Module is the first module enabled, if
           not revert
        // Check if the All External Modules are listed in the
            Whitelist
        if (ISafe(msg.sender).isModuleEnabled(address(palmeraModule)))
            {
             (address[] memory array,) = ISafe(msg.sender).
                 getModulesPaginated(
                  address(Constants.SENTINEL_ADDRESS), 100
             );
```

Fixed. pull43

- check conditions first

1. updateDepthLimits:

```
function updateDepthLimits(uint256 newDeepLimit, uint256 newWidthLimit)
        external
        IsRootSafe(msg.sender)
        requiresAuth
    {
+         if (newDeepLimit > maxDepthLimit) revert Errors.InvalidLimit()
    ; //@audit gas
+         if (newWidthLimit > maxDepthLimit) revert Errors.InvalidLimit
    ();
        bytes32 org = getOrgHashBySafe(caller);
        uint256 currentDepthLimit = depthTreeLimit[org];
        uint256 currentWidthLimit = depthWidthLimit[org];
        // we change the approach in the use case of only wanna change
            one of the limits
        if (newDeepLimit < currentDepthLimit) revert Errors.
            InvalidLimit();
        // we change the approach in the use case of only wanna change
            one of the limits
        if (newWidthLimit < currentWidthLimit) revert Errors.
            InvalidLimit();
 +      address caller = msg.sender;
        emit Events.NewLimitLevel(
            org,
            getSafeIdBySafe(org, caller),
            caller,
            currentDepthLimit,
            newDeepLimit,
            currentWidthLimit,
            newWidthLimit
        );
```

2. isLimitReached:

```
        if (uint8(superSafe.tier) > uint8(1)) {
            revert Errors.SafeAlreadyRemoved();
        }
+       if (superSafe.child.length >= depthWidthLimit[org]) return true
    ;

        (, uint256 level,) = _seekMember(indexId + 1, superSafeId);
        return level >= depthTreeLimit[org];
```

- catch array length before for loop

1. addToList:

```
    function addToList(address[] calldata users)
        external
        IsRootSafe(msg.sender)
        requiresAuth
    {
+   uint265 usersLength = users.length;
+        if (usersLength == 0) revert Errors.ZeroAddressProvided();
        bytes32 org = getOrgHashBySafe(msg.sender);
        _isDisableHelpers(org);
        address currentWallet = Constants.SENTINEL_ADDRESS;
+        for (uint256 i; i < usersLength;) {

.
.
.
        unchecked {
+            listCount[org] += usersLength;
        }
```

- More efficent isSafeLead

```
function isSafeLead(
        uint256 safeId,
        address user,
        bool checkModifyOwners,
        bool checkExecOnBehalf
    ) public view returns (bool) {
        bytes32 org = getOrgBySafe(safeId);
        DataTypes.Safe memory _safe = safesInfoByOrg[org][safeId];
        if (_safe.safe == address(0)) return false;

        /// Check if the user is the lead of the safe
        if (_safe.lead != user) return false;

        /// if the user have the role the method response is true is
            _safe.Lead is the user
        if (doesUserHaveRole(safeId, user, uint8(DataTypes.Role.
            SAFE_LEAD))) return true;
        if (checkModifyOwners) {
            if (
                doesUserHaveRole(
                    safeId,
                    user,
                    uint8(DataTypes.Role.SAFE_LEAD_MODIFY_OWNERS_ONLY)
                )
            ) {
                return true;
            }
        }
        if (checkExecOnBehalf) {
            if (
                doesUserHaveRole(
                    safeId,
                    user,
                    uint8(DataTypes.Role.SAFE_LEAD_EXEC_ON_BEHALF_ONLY)
                )
            ) {
                return true;
            }
        }
        return false;
    }
```

Fixed. pull41