Smart Contract Audit Report

Prepared by: Alice Zhang

Date: 2025-10-24

Contents

Chapter 1

Audit Details

1.1 Scope

```
src/
|-- RaiseBoxFaucet.sol
|-- DeployRaiseBoxFaucet.s.sol
```

1.2 Roles

1.2.1 Owner:

RESPONSIBILITIES:

- deploys contract,
- mint initial supply and any new token in future,
- can burn tokens,
- can adjust daily claim limit,
- can refill sepolia eth balance

LIMITATIONS:

- cannot claimfaucet tokens

1.2.2 Claimer:

RESPONSIBILITIES:

- can claim tokens by calling the claimFaucetTokens function of this contract.

LIMITATIONS:

- Doesn't have any owner defined rights above.

1.2.3 Donators:

RESPONSIBILITIES:

- can donate sepolia eth directly to contract

Chapter 2

Executive Summary

2.1 Issues found

Severity	Number of issues found
High	1
Medium	2
Low	1

Chapter 3

Findings

3.1 High

3.1.1 [H-1] Incorrect Daily ETH Drip Reset, the dailyDrips may not be correct

Description:

In the *./src/RaiseBoxFaucet.sol* claimFaucetTokens() function, the following code resets dailyDrips in the else branch:

```
//Line-211
if (!hasClaimedEth[faucetClaimer] && !sepEthDripsPaused) {
        ...
} else {
        dailyDrips = 0;
}
```

This executes every time a non-first-time claimer calls the faucet, which can unintensionally reset the dail ETH drip counter. As a result, the daily drip limit (dailySepEthCap) may be bypassed, allowing more ETH to be claimed than intended within a single day.

Impact

- The faucet can distribute more ETH than the intended daily cap.

Proof of Concepts

1. Assume dailySepEthCap = 5 ETH and dailyDrips = 4 ETH.
2. User A (first-time claimer) receives 1 ETH -> dailyDrips = 5 ETH.
3. User B (already claimed ETH) calls the faucet -> hits the else { dailyDrips = 0; } branch.
4. Now dailyDrips is reset to 0, allowing the faucet to drip ETH again, bypassing the cap.

Recommended mitigation

- Remove the else { dailyDrips = 0; } branch.

3.2 Medium

3.2.1 [M-1] Incorrect Faucet Token Burn Transfer, Faucet has no token after transfer

Description

In the *./src/RaiseBoxFaucet.sol* burnFaucetTokens() function, the following line:

```
//Line-132
_transfer(address(this), msg.sender, balanceOf(address(this)));
```

transfers the entire balance of faucet tokens from the contract to the owner (msg.sender). However, according to the function's comment: > "Transfers tokens to owner first, then burns from owner"

the intended behavior is to only transfer the amount of tokens that will be burned (amountToBurn).

As a result, the faucet ends up spending more tokens to the owner than intended, and the owner keeps any remaining tokens after the burn.

Impact

The current implementation may unintentionally transfer all faucet tokens to the owner, leaving the faucet empty while only a small portion (amountToBurn) is actually burned. This breaks the intended faucet logic and allows the owner to take more tokens than expected.

Proof of Concepts

1. Assume the contract holds 10000 faucet tokens.
2. Call burnFaucetTokens(1000) burns only 1000 tokens.
3. _transfer(address(this), msg.sender, balanceOf(address(this))) transfers all 10000 tokens to the owner.
4. _burn(msg.sender, 1000) burns only 1000 tokens.
5. Result: owner retains 9000 tokens, faucet balance is 0.
6. This violates the intended logic of burning only the specified amount

Recommended mitigation

- Modify the transfer to only send amountToBurn tokens before burning:

```
_transfer(address(this), msg.sender, amountToBurn)
```

3.2.2 [M-2] Incorrect lastFaucetDripDay set, dailyClaimCount may not be actually counted by a day

Description

In the *./src/RaiseBoxFaucet.sol* claimFaucetTokens() function, the following code resets dailyClaimCount:

```
//Line-220
if (block.timestamp > lastFaucetDrip) + 1 days) {
        lastFaucetDripDay = block.timestamp;
        dailyClaimCount = 0;
}
```

lastFaucetDripDay is assigned directly as block.timestamp, which represents the exact moment the function is called. This means the "daily" reset is not aligned to natural calender days, but instead depends on when the first user calls the faucet each day.

As a result, the 24-hour period for dailyClaimCount can start at any arbitrary time, leading to inaccurate daily statics and inconsistent enforcement of the daily claim limit.

Impact

- The faucet may allow users to claim more than the intended daily limit if the first claim occurs later in the day.

Recommended mitigation

- change the code

```
//old-version
lastFaucetDripDay = block.timestamp;

//new-version
lastFaucetDripDay = (block.timestamp / 1 days) * 1 days;
```

3.3 Low

3.3.1 [L-1] Misspelled SPDX License Identifier

Description

In the both *./script/DeployRaiseBoxFaucet.s.sol* and *./src/RaiseBoxFaucet.sol*, the SPDX license identifier is incorrectly written as:

```
//SPDX-Lincense-Identifier: MIT
```

Impact

- Trigger Solidity compiler waining.

Recommended mitigation

Correct the spelling of the SPDX identifier to the official format:

```
// SPDX-License-Identifier: MIT
```

Chapter 4

Informational

Chapter 5

Gas