Security Assessment of Engie Solar Crowdfunding

Energy Web AG on behalf of the Energy Web Community Fund

April 2022

Version: 2.1

Presented by:
BTblock LLC

Corporate Headquarters
BTblock LLC
PO Box 147044
Lakewood, CO 80214
United States

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

EXECUTIVE SUMMARY

Overview

Energy Web AG on behalf of Energy Web Community Fund engaged BTblock LLC to perform a Security Assessment of Engie Solar Crowdfunding.

The assessment was conducted remotely by the BTblock Security Team. Testing took place on March 01 - March 14, 2022, and focused on the following objectives:

- Provide the customer with an assessment of their overall security posture and any risks that were discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the result of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the BTblock Security Teams took to identify and validate each issue, as well as any applicable recommendations for remediation.

A re-review took place on March 31, 2022.

Key Findings

The following issues were identified during the testing period. These should be prioritized for remediation to reduce the risk they pose:

- BT-EW-01 - Insufficient Funding Restrictions
- BT-EW-02 - Reentrancy attacks
- BT-EW-03 - ERC20Burnable interface functions allow token exchange and burn
- BT-EW-04 - Funds can be locked
- BT-EW-05 - Gas Cost Expenditure

During the test, the following positive observations were noted regarding the scope of the engagement:

- The team was very supportive and open to discuss the design choices made

Based on formal verification we conclude that the reviewed code implements the documented functionality.

Scope and Rules of Engagement

BTblock performed a Security Assessment of Engie Solar Crowdfunding. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

The source code was supplied through a private repository at https://github.com/energywebfoundation/engie-solar-crowdfunding with the commit hash 5b49d828d526a9f83dcd423d437fc938ee42aa70. The code was audited for a second time at commit hash b5812045310d43cef61fdcab545c0cf4fb21b6e4.

Files included in the code review

```
engie-solar-crowdfunding/
├── apps/
│   ├── ew-crowdfunding/
│   │   ├── components/
│   │   │   ├── AppContainer/
│   │   │   │   ├── AppContainer.styles.ts
│   │   │   │   ├── AppContainer.tsx
│   │   │   │   └── index.ts
│   │   │   ├── ConnectCard/
│   │   │   │   ├── ConnectCard.effects.ts
│   │   │   │   ├── ConnectCard.tsx
│   │   │   │   └── index.ts
│   │   │   ├── ContributionItem/
│   │   │   │   ├── ContributionItem.styles.ts
│   │   │   │   ├── ContributionItem.tsx
│   │   │   │   └── index.ts
│   │   │   ├── ImageText/
│   │   │   │   ├── ImageText.styles.ts
│   │   │   │   ├── ImageText.tsx
│   │   │   │   └── index.ts
│   │   │   ├── InfoCard/
│   │   │   │   ├── InfoCard.styles.ts
│   │   │   │   ├── InfoCard.tsx
│   │   │   │   └── index.ts
│   │   │   ├── ListComponent/
│   │   │   │   ├── ListComponent.styles.ts
│   │   │   │   ├── ListComponent.tsx
│   │   │   │   └── index.ts
│   │   │   ├── MediaText/
│   │   │   │   ├── MediaText.styles.ts
│   │   │   │   ├── MediaText.tsx
│   │   │   │   └── index.ts
│   │   │   ├── ProgressBar/
│   │   │   │   ├── ProgressBar.styles.ts
│   │   │   │   ├── ProgressBar.tsx
│   │   │   │   └── index.ts
│   │   │   ├── SocialLinks/
│   │   │   │   ├── SocialLinks.effects.ts
│   │   │   │   ├── SocialLinks.styles.ts
│   │   │   │   ├── SocialLinks.tsx
│   │   │   │   └── index.ts
│   │   │   ├── WalletCard/
│   │   │   │   ├── WalletCard.styles.ts
│   │   │   │   ├── WalletCard.tsx
│   │   │   │   └── index.ts
│   │   │   ├── form/
│   │   │   │   ├── FormInputText/
│   │   │   │   │   ├── FormInputText.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   └── index.ts
│   │   │   ├── modals/
│   │   │   │   ├── DialogContainer/
│   │   │   │   │   ├── DialogContainer.styles.ts
```

```
│   │   │   │   │   ├── DialogContainer.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   ├── DialogTitle/
│   │   │   │   │   ├── DialogTitle.styles.ts
│   │   │   │   │   ├── DialogTitle.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   └── index.ts
│   │   │   └── index.ts
│   │   ├── containers/
│   │   │   ├── Carousel/
│   │   │   │   ├── Carousel.styles.ts
│   │   │   │   ├── Carousel.tsx
│   │   │   │   └── index.ts
│   │   │   ├── Contact/
│   │   │   │   ├── Contact.styles.ts
│   │   │   │   ├── Contact.tsx
│   │   │   │   └── index.ts
│   │   │   ├── Footer/
│   │   │   │   ├── Footer.styles.ts
│   │   │   │   ├── Footer.tsx
│   │   │   │   └── index.ts
│   │   │   ├── HowTo/
│   │   │   │   ├── HowTo.tsx
│   │   │   │   └── index.ts
│   │   │   ├── InfoContainer/
│   │   │   │   ├── InfoContainer.effects.ts
│   │   │   │   ├── InfoContainer.styles.ts
│   │   │   │   ├── InfoContainer.tsx
│   │   │   │   └── index.ts
│   │   │   ├── InfoPane/
│   │   │   │   ├── InfoPane.effects.ts
│   │   │   │   ├── InfoPane.styles.ts
│   │   │   │   ├── InfoPane.tsx
│   │   │   │   └── index.ts
│   │   │   ├── Lending/
│   │   │   │   ├── Lending.effects.ts
│   │   │   │   ├── Lending.styles.ts
│   │   │   │   ├── Lending.tsx
│   │   │   │   └── index.ts
│   │   │   ├── LendingDetails/
│   │   │   │   ├── LendingDetails.effects.ts
│   │   │   │   ├── LendingDetails.styles.ts
│   │   │   │   ├── LendingDetails.tsx
│   │   │   │   └── index.ts
│   │   │   ├── LendingStats/
│   │   │   │   ├── LendingStats.effects.ts
│   │   │   │   ├── LendingStats.styles.ts
│   │   │   │   ├── LendingStats.tsx
│   │   │   │   └── index.ts
│   │   │   ├── LendingTerms/
│   │   │   │   ├── LendingTerms.effects.ts
│   │   │   │   ├── LendingTerms.styles.ts
│   │   │   │   ├── LendingTerms.tsx
```

```
│   │   │   │   └── index.ts
│   │   │   ├── Navigation/
│   │   │   │   ├── Navigation.effects.ts
│   │   │   │   ├── Navigation.styles.ts
│   │   │   │   ├── Navigation.tsx
│   │   │   │   └── index.ts
│   │   │   ├── RoleEnrollment/
│   │   │   │   ├── ApprovedSynced/
│   │   │   │   │   ├── ApprovedSynced.styles.ts
│   │   │   │   │   ├── ApprovedSynced.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   ├── EmailVerification/
│   │   │   │   │   ├── EmailVerification.effects.ts
│   │   │   │   │   ├── EmailVerification.styles.ts
│   │   │   │   │   ├── EmailVerification.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   ├── NotApproved/
│   │   │   │   │   ├── NotApproved.effects.ts
│   │   │   │   │   ├── NotApproved.styles.ts
│   │   │   │   │   ├── NotApproved.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   ├── NotSynced/
│   │   │   │   │   ├── NotSynced.effects.ts
│   │   │   │   │   ├── NotSynced.styles.ts
│   │   │   │   │   ├── NotSynced.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   ├── RoleEnrollment.tsx
│   │   │   │   └── index.ts
│   │   │   ├── StakingTimeline/
│   │   │   │   ├── StakingTimeline.effects.ts
│   │   │   │   ├── StakingTimeline.styles.ts
│   │   │   │   ├── StakingTimeline.tsx
│   │   │   │   └── index.ts
│   │   │   ├── Welcome/
│   │   │   │   ├── Welcome.styles.ts
│   │   │   │   ├── Welcome.tsx
│   │   │   │   └── index.ts
│   │   │   ├── modals/
│   │   │   │   ├── Confirm/
│   │   │   │   │   ├── Confirm.effects.ts
│   │   │   │   │   ├── Confirm.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   ├── Congrats/
│   │   │   │   │   ├── Congrats.effects.ts
│   │   │   │   │   ├── Congrats.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   ├── Lend/
│   │   │   │   │   ├── Lend.effects.ts
│   │   │   │   │   ├── Lend.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   ├── Login/
│   │   │   │   │   ├── Login.effects.ts
│   │   │   │   │   ├── Login.styles.ts
```

```
│   │   │   │   │   ├── Login.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   ├── Redeem/
│   │   │   │   │   ├── Redeem.effects.ts
│   │   │   │   │   ├── Redeem.styles.ts
│   │   │   │   │   ├── Redeem.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   ├── Web3Notification/
│   │   │   │   │   ├── Web3Notification.effects.ts
│   │   │   │   │   ├── Web3Notification.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   ├── DSLAModalsCenter.tsx
│   │   │   │   └── index.ts
│   │   │   └── index.ts
│   │   ├── context/
│   │   │   ├── iam/
│   │   │   │   ├── getIamService.ts
│   │   │   │   ├── getSignerService.ts
│   │   │   │   ├── index.ts
│   │   │   │   └── setListeners.ts
│   │   │   ├── modals/
│   │   │   │   ├── index.ts
│   │   │   │   ├── provider.tsx
│   │   │   │   ├── reducer.ts
│   │   │   │   └── types.ts
│   │   │   └── index.ts
│   │   ├── dsla-theme/
│   │   │   ├── DSLAThemeProvider.tsx
│   │   │   ├── index.ts
│   │   │   └── theme.ts
│   │   ├── hooks/
│   │   │   ├── index.ts
│   │   │   ├── useContractStatus.ts
│   │   │   └── useFetching.ts
│   │   ├── pages/
│   │   │   ├── _app.tsx
│   │   │   ├── _document.tsx
│   │   │   ├── global.css
│   │   │   ├── index.tsx
│   │   │   ├── privacy-policy.tsx
│   │   │   ├── terms-and-conditions.tsx
│   │   │   └── wallet.tsx
│   │   ├── public/
│   │   │   ├── LandingCover2.jpg
│   │   │   └── MountMeru.jpg
│   │   ├── redux-store/
│   │   │   ├── smart-contract/
│   │   │   │   ├── actions.ts
│   │   │   │   ├── index.ts
│   │   │   │   ├── reducers.ts
│   │   │   │   ├── selectors.ts
│   │   │   │   └── types.ts
│   │   │   ├── web3/
```

```
│   │   │   │   ├── actions.ts
│   │   │   │   ├── index.ts
│   │   │   │   ├── reducers.ts
│   │   │   │   ├── selectors.ts
│   │   │   │   └── types.ts
│   │   │   ├── index.ts
│   │   │   ├── localStorage.ts
│   │   │   ├── root-reducer.ts
│   │   │   └── store.ts
│   │   ├── specs/
│   │   │   └── index.spec.tsx
│   │   ├── utils/
│   │   │   ├── StakingTimelineEnum.ts
│   │   │   ├── formatDate.ts
│   │   │   ├── formatUTCDate.ts
│   │   │   ├── formatUTCTimestamp.ts
│   │   │   ├── getStakingTimeline.ts
│   │   │   ├── index.ts
│   │   │   ├── propertyExists.ts
│   │   │   ├── shortenAddress.ts
│   │   │   └── shortenDid.ts
│   │   ├── index.d.ts
│   │   ├── jest.config.js
│   │   ├── next-env.d.ts
│   │   ├── next.config.js
│   │   ├── project.json
│   │   ├── theme.d.ts
│   │   ├── tsconfig.json
│   │   └── tsconfig.spec.json
│   └── ew-crowdfunding-e2e/
│       ├── src/
│       │   ├── fixtures/
│       │   │   └── example.json
│       │   ├── integration/
│       │   │   └── app.spec.ts
│       │   └── support/
│       │       ├── app.po.ts
│       │       ├── commands.ts
│       │       └── index.ts
│       ├── cypress.json
│       ├── project.json
│       └── tsconfig.json
├── libs/
│   └── ew-crowdfunding/
│       └── smart-contracts/
│           ├── contracts/
│           │   ├── interfaces/
│           │   │   └── IClaimManager.sol
│           │   └── Staking.sol
│           ├── ethers/
│           │   ├── factories/
│           │   │   ├── ERC20Burnable__factory.ts
│           │   │   ├── ERC20__factory.ts
```

```
│           │   │   ├── IClaimManager__factory.ts
│           │   │   ├── IERC20Metadata__factory.ts
│           │   │   ├── IERC20__factory.ts
│           │   │   └── Staking__factory.ts
│           │   ├── ERC20.d.ts
│           │   ├── ERC20Burnable.d.ts
│           │   ├── IClaimManager.d.ts
│           │   ├── IERC20.d.ts
│           │   ├── IERC20Metadata.d.ts
│           │   ├── Staking.d.ts
│           │   ├── common.d.ts
│           │   ├── hardhat.d.ts
│           │   └── index.ts
│           ├── scripts/
│           │   ├── deploy.ts
│           │   └── initialize.ts
│           ├── src/
│           │   ├── flattened_contracts/
│           │   │   └── Staking_flat.sol
│           │   ├── lib/
│           │   │   ├── deployedAddress.ts
│           │   │   ├── ew-crowdfunding-smart-contracts.spec.ts
│           │   │   ├── ew-crowdfunding-smart-contracts.ts
│           │   │   └── index.ts
│           │   └── index.ts
│           ├── test/
│           │   ├── index.spec.ts
│           │   └── rewards.spec.ts
│           ├── README.md
│           ├── Staking_flat.sol
│           ├── hardhat.config.ts
│           ├── jest.config.js
│           ├── package-lock.json
│           ├── package.json
│           ├── project.json
│           ├── tsconfig.json
│           ├── tsconfig.lib.json
│           └── tsconfig.spec.json
├── tools/
│   └── tsconfig.tools.json
├── README.md
├── babel.config.json
├── jest.config.js
├── jest.preset.js
├── nx.json
├── package-lock.json
├── package.json
├── tsconfig.base.json
└── workspace.json
```

Table 1: Scope

TECHNICAL ANALYSES AND FINDINGS

During the Security Assessment of Engie Solar Crowdfunding, we discovered:

- 2 findings with HIGH severity rating.
- 2 findings with MEDIUM severity rating.
- 1 finding with LOW severity rating.

The following chart displays the findings by severity.

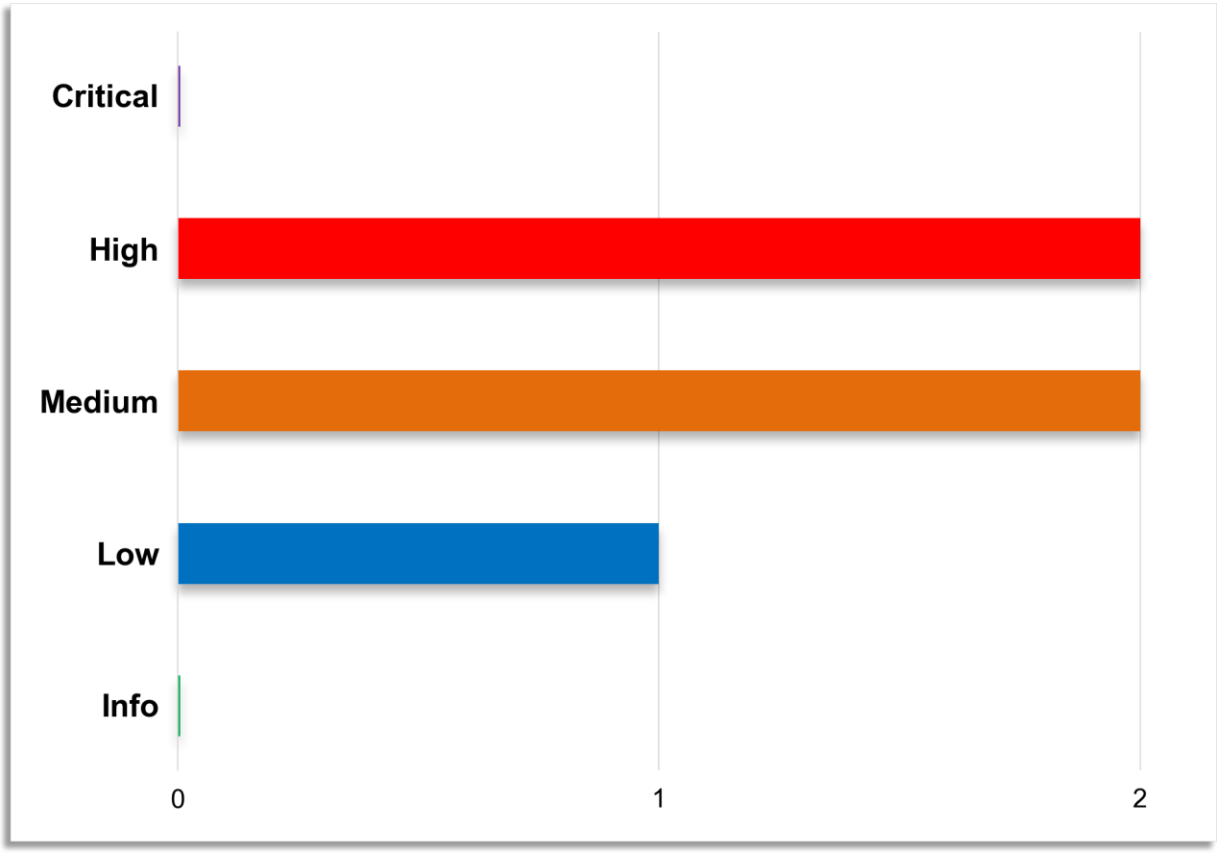

Figure 1: Findings by Severity

Findings

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

#	Severity	Description
BT-EW-01	**High**	Insufficient Funding Restrictions
BT-EW-02	**High**	Reentrancy attacks
BT-EW-03	**Medium**	ERC20Burnable interface functions allow token exchange and burn
BT-EW-04	**Medium**	Funds can be locked
BT-EW-05	**Low**	Gas Cost Expenditure

Table 2: Findings Overview

Technical Analyses

Based on the source code, the validity of the code was verified and confirmed that the intended functionality was implemented correctly and to the extent that the state of the repository allowed, unless otherwise stated.

Conclusion

Based on formal verification we conclude that the code implements the documented functionality to the extent of the reviewed code.

Technical Findings

General Observations

Engie Solar Crowdfunding provides a small contract allowing a staking mechanism. Despite its size, there were a few findings that could be abused both by users and governing bodies. When notified, the development team was quick to understand and address the issues. Additionally, the link to Energy Web's infrastructure contracts enhances the contract's security and reliability.

Insufficient Funding Restrictions

Finding ID: BT-EW-01
Severity: **High**
Status: **Remediated**

Description

The lifetime of the contract includes a funding period, which should take place after staking is completed. During that period, stakes are not allowed to be deposited. Withdrawals can happen at any time. A reward provider can provide an amount recorded on the contract's `totalRewards`. After the funding period, users can withdraw their stakes with 10% interest. Additionally, if the contract is terminated by the owner, the full amount of `totalRewards` initially deposited is returned to the rewards' provider.

The problem arises in two places. The first is that since `totalRewards` is not reduced after withdrawing an amount with interest, some users could have their stakes locked. E.g. in a case where three users stake 10 ETH, while the rewards provider provides only 1 ETH; if two users withdraw 10 + 1 ETH each, there will be 9 ETH left in the pool, which is less than the final user's initial stake. The transfer of the amount with interest would thus fail.

The second scenario, which results in the same effect, occurs as termination is not restricted to happen before the end of the funding period. Even if the correct amount of `totalRewards` is initially provided, after termination the full amount is returned to the provider. This could result in stakes being locked. E.g. in a case where two users stake 10 ETH, while the rewards provider provides 2 ETH; after a user withdraws 10 + 1 ETH, if the owner terminates the contract, 2 ETH will be returned to the provider leaving the contract with 9 ETH.

Proof of Issue

File name: Staking.sol
Line number: 113

```
    function depositRewards() external payable notAborted activated
notfunded {
        require(msg.value > 0, "Not rewards provided");
        require(hasRole(msg.sender, serviceRole) || (msg.sender == owner),
"Not enrolled as service provider");
        totalRewards += msg.value;
        rewardProvider = msg.sender;
        contractFunded = true;
        emit RewardSent(msg.sender, msg.value, block.timestamp);
    }
```

Where `totalRewards` and `rewardProvider` are recorded without checks that the amount suffices the total amount of stakes.

Line number: 169

```
    function terminate() external onlyOwner {
        require(aborted == false , "Already terminated");
        uint256 payout = totalRewards;
        if (payout != 0){
            payable(rewardProvider).transfer(payout);
        }
```

Where `terminate` can be executed after the funding period.

Line number: 248

```
    function _getRewards(uint256 _amount) internal
sufficientBalance(_amount) view returns(uint256 reward){
        if (!aborted && totalRewards != 0){
            uint256 interests = _amount * 1e2;
            reward = interests / 1e3 + _amount;
        } else {
            reward = _amount;
        }
    }
```

Where `totalRewards` are only checked for a non-zero value and are not updated.

Severity and Impact Summary

Users' stakes might be locked due to missing restrictions during reward deposition and termination.

Recommendation

The reward's deposit should be verified to be sufficient, i.e. `>= totalStaked * 10`. Termination should happen only before `endDate`.

Another approach could include reducing the `totalRewards` after part of the reward total is withdrawn. However, this alone, would not guarantee that all users will receive a 10% interest.

Remediation

Termination is now checked with `require(block.timestamp < endDate` while `withdrawsAllowed` allows `block.timestamp >= endDate`.

Reentrancy attacks

Finding ID: BT-EW-02
Severity: **High**
Status: **Remediated**

Description

Reentrancy attacks occur in functions where part of the state changes after, rather than before, an ETH transfer. If such a transfer is towards a malicious contract that overrides the `fallback` function, it can call the contract's original function which, since its state has not been updated, might re-initiate the transfer.

In the case of Staking.sol, this happens in `redeem` and `stake`, where `totalStaked` is updated after the transfer of ETH.

Proof of Issue

File name: Staking.sol
Line numer: 169

```
    function terminate() external onlyOwner {
        require(aborted == false , "Already terminated");
    uint256 payout = totalRewards;
        if (payout != 0){
            payable(rewardProvider).transfer(payout);
        }
        deleteParameters();
        aborted = true;
```

Line number: 198

```
                refund(overFlow_limit + overFlow_hardCap);
                totalStaked += finalMint;
```

Line number: 204

```
                refund(overFlow_limit);
                totalStaked += toMint_limit;
```

Line number: 237

```
        payable(msg.sender).transfer(toWithdraw);
        emit Withdrawn(msg.sender, toWithdraw, block.timestamp);
        emit TokenBurnt(msg.sender, _amount, block.timestamp);
        totalStaked -= _amount;
```

The first case of reentrancy can happen in `terminate`. If `rewardProvider` and `owner` are malicious contracts, they can coordinate to transfer `payout` multiple times.

The case is also concerning if a malicious patron contract is registered. When redeeming, `totalStaked` is reduced which shouldn't have a negative impact. However, during staking, a

malicious contract can use the attack to circumvent the `belowLimit` constraint. This would allow an attacker to stake more than allowed and potentially gain more rewards than expected.

Severity and Impact Summary

An attacker can steal stakes during termination using a malicious owner contract and circumvent the `hardCap` limitation on malicious patron contracts.

Recommendation

Delete parameters before the transfer during termination. Update `totalStaked` before transferring ETH, during staking, and when redeeming.

Remediation

All transfers are now performed after changing the state.

ERC20Burnable interface functions allow token exchange and burn

Finding ID: BT-EW-03
Severity: **Medium**
Status: **Remediated**

Description

The `Staking.sol` contract extends OpenZeppelin's `ERC20Burnable.sol` which provides implementation for the following functions:

- `burnFrom`
- `approve`
- `transferFrom`
- `increaseAllowance`
- `decreaseAllowance`

By using combinations of the above functions, stakers could exchange `Staking` tokens without the internal state of the contract being updated. This includes the `stakes` variable which maps users' deposits. If not updated, this could allow for users holding stakes above the contract's limits. In addition, `burnFrom` would allow stakers to burn their stakes without redeeming them, deviating from the contract's implementation of `burn`.

> Note
>
> This finding was not included in the initial versions of the report but was added after discussions with the development team.

Severity and Impact Summary

Stakers can use `ERC20Burnable` functions which are not overwritten to exchange and burn tokens.

Recommendation

These functions could be prevented by providing an overwriting implementation containing `revert` functionality. Alternatively, the functions could be implemented to correctly reflect changes onto the contract's internal state.

Remediation

Functions `transfer`, `transferFrom` were implemented to reflect the changes in the contract's internal state. A `revert` implementation for `burnFrom` has also been provided.

Funds can be locked

Finding ID: BT-EW-03
Severity: **Medium**
Status: **Remediated**

Description

As per design, reward providers can fund the contract after the staking and before the redeeming period. Given the interest rate is fixed, the rewards provided might exceed the funds needed. In this case, after all patrons have redeemed their rewards, there is no way for the reward provider to reclaim the excess rewards, which can thus be locked in the contract.

Severity and Impact Summary

Excess rewards provided can be locked into the contract.

Recommendation

This issue was pointed out by the development team and its fixes will be provided for re-review. Our recommendation is to refund the excess amount during `depositRewards`.

Remediation

The team introduced a `sweep` function allowing the reward provider to be paid remaining rewards. This can only occur after a `fullStopDate`, which however, now allows for locking of user funds by restricting withdrawals to be before the `fullStopDate`. This action is intentional and will restrict refunding of remaining funds to be done manually by the team only.

Gas Cost Expenditure

Finding ID: BT-EW-04
Severity: **Low**
Status: **Open**

Description

Since gas costs are expensive, we list the following places where calculations could be avoided:

Proof of Issue

File name: Staking.sol
Line number: 252

```
uint256 interests = _amount * 1e2;
reward = interests / 1e3 + _amount;
```

could be replaced with `reward = _amount / 10 + _amount` without loss of precision.

Line numbers: 188-189, 192-193, 210-211, include pattern:

```
uint256 Limit = Value - (Cap - Accumulated);
uint256 Mint = Value - Limit;
```

where variable names serve as placeholders. This could be replaced by

```
uint256 Mint = Cap - Accumulated;
uint256 Limit = Value - Mint;
```

saving one operation.

Line numbers: 187, 191, 208, include pattern

```
if (Accumulated >= Cap)
```

where variable names serve as placeholders. If `Accumulated == Cap`, the refunds result to 0 and thus calculations could be avoided by using a > comparison.

Severity and Impact Summary

Gas costs could be avoided during staking and redeeming.

Recommendation

Alternatives for each case are provided in proof of issue above.

Remediation

The issue was partially addressed. In the order above, the first and last sets of unnecessary calculations were correctly replaced. Overflow calculations have not been optimized. Due to partial remediation, this issue is still open.