DeHacker

# Code Security Assessment

CEEK

Aug 8th, 2023

# Contents

## Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.
Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

## Issue Categories

Every issue in this report was assigned a severity level from the following:

### Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

### Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

### Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

### Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

### Informational

A vulnerability that has informational character but is not affecting any of the code.

# Overview

## Project Summary

| | |
|---|---|
| **Project Name** | CEEK |
| **Platform** | BSC |
| **Website** | https://www.ceek.com/ |
| **Type** | NFT |
| **Language** | Solidity |

## Vulnerability Summary

| **Vulnerability Level** | **Total** | Pending | **Declined** | **Acknowledged** | **Partially Resolved** | **Resolved** |
|---|---|---|---|---|---|---|
| Critical | 0 | 0 | 0 | 0 | 0 | 0 |
| Major | 0 | 0 | 0 | 0 | 0 | 0 |
| Medium | 0 | 0 | 0 | 0 | 0 | 0 |
| Minor | 1 | 0 | 0 | 1 | 0 | 0 |
| Informational | 2 | 2 | 0 | 0 | 0 | 0 |
| Discussion | 0 | 0 | 0 | 0 | 0 | 0 |

## Audit scope

| ID | File | SHA256 Checksum |
|---|---|---|
| BRR | BaseRelayRecipient.sol | 0ceea6838f96595a3c9f6da9fb02f5445 24162baad3c69f00ace5a7df75b4982 |
| BRR | Ceek.sol | 8a76cd8ba95caab414b503e0096ec6c2b f087ea08190a2d146cc3a09adffcdf4 |
| CCP | Context.sol | 6a25312554a817075fbe85e3c57f5e0ec ad5b0bac4303bd3967ce9680e71a2af |
| ERC | ERC20.sol | 7e8926061cab72cff6ace1c2549d62662 8b26ca95118a644282631ce0e1e4d4b |
| ERB | ERC20Burnable.sol | 0d722f6afa35f9ef6a5f03866b3368060 b9e48f47c346fb2f72a5729347edee9 |
| ERK | ERCCapped.sol | 30854df7c668bbe1c6e6d8c0aecea2b65 1d870b8a403a8bce8707105f9e65191 |
| IER | IERC20.sol | 397534ef6f97a7fa11089b8f05cf538374 9067352d06c0e39b880fd08743489a |
| IRR | IRelayRecipient.sol | c365e6f665eebd9745c43416e41a715cd b292efa50a9b5a2dfb42dbf89e20bb4 |
| OCK | Ownable.sol | 9baa3fc584343ac5fc04a399c7d922998 8c8f77fdf4d5ce78cc328dbc36735f8 |
| SMC | SafeMath.sol | c12e80c632ffe9759da012ded8a255048 b87d4ee6d7156c7aa9f7e98e4b9a6b6 |

## Findings

| ID | Category | Severity | Status |
|---|---|---|---|
| CCK-01 | Gas Optimization | Informational | Pending |
| CCK-02 | Centralization /Privilege | Minor | Acknowledged |
| CCK-03 | Gas Optimization | Informational | Pending |

# INFORMATIONAL

## CCK-01 | Missing Emit Events

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization | Informational | Ceek.sol: 22, 18 | Pending |

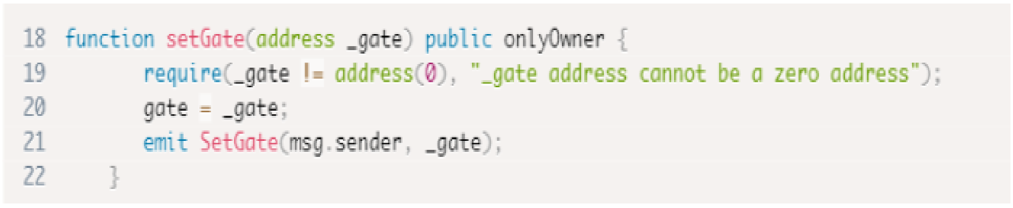
## Description

The sensitive functions' input parameters miss the sanitized check for zero address and emit the eventlogs.

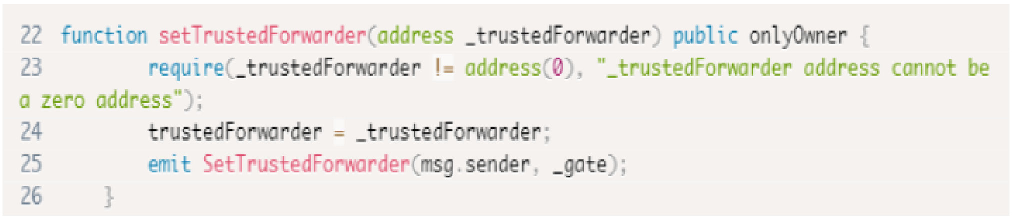
For example:
setGate()
setTrustedForwarder()

## Recommendation

We recommend adding zero address check and emit the event logs for functions setGate () ,setTrustedForwarder() .



```
18  function setGate(address _gate) public onlyOwner {
19          require(_gate != address(0), "_gate address cannot be a zero address");
20          gate = _gate;
21          emit SetGate(msg.sender, _gate);
22      }
```



```
22  function setTrustedForwarder(address _trustedForwarder) public onlyOwner {
23          require(_trustedForwarder != address(0), "_trustedForwarder address cannot be
a zero address");
24          trustedForwarder = _trustedForwarder;
25          emit SetTrustedForwarder(msg.sender, _gate);
26      }
```

## INFORMATIONAL

### CCK-02 | Centralized Risk

| Category | Severity | Location | Status |
|---|---|---|---|
| Centralization / Privilege | Minor | Ceek.sol: 18 | Acknowledged |

### Description

The account owner of onlyOwner role can assign any arbitrary gate address, which later the gateaddress uses for minting the arbitrary token amount. Any compromise to this account may allow thehacker to take advantage of these two functions and eventually drain all tokens from the contract.

### Recommendation

We advise the client to carefully manage the role onlyOwner and gate 's account private key and avoid anypotential risks of being hacked. In general, we strongly recommend centralized privileges or roles in theprotocol to be improved via a decentralized mechanism or via smart-contract-based accounts withenhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:
Time-lock with reasonable latency, i.e., 48 hours, for awareness on privileged operations;
Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to theprivate key;
Introduction of a DAO / governance/voting module to increase transparency and user involvement.

## INFORMATIONAL

### CCK-03 | Proper usage of public and external type

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization | Informational | Ceek.sol: 18, 22, 26 | Pending |

### Description

Public functions that are never called by the contract could be declared external. When the inputs arearrays, external functions are more efficient than public functions.
Example functions :

setGate() ;
setTrustedForwarder() ;
mint() ;

### Recommendation

We recommend using the external attribute for functions never called from the contract.

## Disclaimer

# Appendix

## Finding Categories

### Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

### Coding Style

Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

### Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

### Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

## Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

## About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

### BLOCKCHAIINS

| | |
|---|---|
| Ethereum | Cosmos |
| Eos | Substrate |

### TECH STACK

| | |
|---|---|
| Python | Solidity |
| Rust | c++ |

### CONTACTS

- https://dehacker.io
- https://twitter.com/dehackerio
- https://github.com/dehacker/audits_public
- https://t.me/dehackerio
- https://blog.dehacker.io/

DeHacker

Aug 2023