Smart Contract Audit Report

Date: September 18, 2024

Contract: ConvexToken

Contract Functionality:

The contract allows users to interact with a decentralized finance (DeFi) protocol by:

- Depositing or withdrawing tokens into a liquidity pool
- Borrowing or lending assets against collateral
- Participating in yield farming and other liquidity provision mechanisms
- Interacting with an on-chain oracle to determine certain parameters

Vulnerabilities:

1. **Visibility errors, including unrestricted action**
    - **Severity**: High
    - **Description**: The `mint` function can only be called by the `operator`, but there is no restriction on who can call the `updateOperator` function. This allows an attacker to potentially change the `operator` to their own address and then mint tokens at will.
    - **Impact**: An attacker could call `updateOperator` to set themselves as the new operator. Once they have control of the operator address, they can mint unlimited tokens, which could severely dilute the value of existing tokens and lead to financial losses for other users.
    - **Mitigation**: Implement access control for the `updateOperator` function to ensure it can only be called by a trusted entity or by a governance mechanism that requires approval from the token holders.
2. **Absence of code logic or sanity check**
    - **Severity**: Medium
    - **Description**: The `mint` function does not have proper checks for the `_amount` parameter. If `_amount` is set to an extremely high value, it could cause overflows or other unexpected behaviors.
    - **Impact**: An attacker could potentially call the `mint` function with a very large `_amount` parameter if they manage to become the operator, leading to issues with the total supply and potentially causing overflows.
    - **Mitigation**: Add checks to the `mint` function to validate the `_amount` parameter and prevent overflows.
3. **Direct call to untrusted contract**
    - **Severity**: High
    - **Description**: The `updateOperator` function fetches the operator from an external contract (`IStaker`). If the `vecrvProxy` address is set to a malicious contract, it could return an arbitrary address as the operator.
    - **Impact**: An attacker could manipulate the `vecrvProxy` to point to a malicious contract that returns their own address as the operator, thereby gaining control over the minting process.
    - **Mitigation**: Implement a trusted address verification mechanism for the `vecrvProxy` or use a trusted address registry to ensure that it always points to a trusted contract.

Conclusion:

The contract has several vulnerabilities that need to be addressed. These vulnerabilities can be exploited by an attacker to gain control over token minting, leading to potential financial damage to users.

Contract Status:

The contract is **not safe** at this time.

Recommendation:

The developers should address the identified vulnerabilities immediately to prevent potential security risks. The contract should be re-audited after the vulnerabilities have been addressed.