Vether3 Smart Contract Audit Report

This report summarizes the findings of a security audit performed on the Vether3 smart contract using Sli

1. Unchecked Low-Level Calls

- **Severity**: Medium
- **Description**: The contract uses `burnAddress.call.value(msg.value)("")` in the `receive()` and `burnEt
- **Impact**: If the `burnAddress` contract fails to execute the call, the funds will be lost.
- **Mitigation**: Replace `burnAddress.call.value(msg.value)("")` with the safer `burnAddress.transfer(msg

2. Missing Events for Arithmetic Operations

- **Severity**: Low
- **Description**: The `upgradeV1()` and `upgradeV2()` functions modify the `upgradedAmount` state var
- **Impact**: This lack of transparency could make it difficult to track the amount of tokens upgraded.
- **Mitigation**: Add an event to both `upgradeV1()` and `upgradeV2()` functions that emits the updated v

3. Timestamp-Based Vulnerability in `_updateEmission()` Function

- **Severity**: Medium
- **Description**: The `_updateEmission()` function relies on the `now` timestamp to determine if a new d
- **Impact**: Malicious actors could potentially manipulate the blockchain timestamp to skip ahead in time
- **Mitigation**: Use a block number-based mechanism to determine if a new day or era has started inste

4. Potential Denial-of-Service (DoS) Attacks

- **Severity**: Low
- **Description**: While the `_updateEmission()` function updates the emission state daily, a malicious ac
- **Impact**: A successful DoS attack could temporarily prevent the contract from functioning correctly.
- **Mitigation**: Implement measures to prevent excessive gas consumption within the `_updateEmission

5. Inconsistent Naming Conventions

- **Severity**: Low
- **Description**: Several variables and functions in the contract do not follow the Solidity naming conven
- **Impact**: Inconsistent naming conventions make the code harder to read and understand, potentially i
- **Mitigation**: Refactor the code to consistently follow Solidity naming conventions to enhance code rea

6. `purgeDeployer()` Function Vulnerability

- **Severity**: Medium
- **Description**: The `purgeDeployer()` function allows the deployer to set their address to the zero addr
- **Impact**: An attacker could exploit this vulnerability to gain control over the contract and its functionali
- **Mitigation**: Implement a check before executing the `purgeDeployer()` function to ensure that the de

7. Logic Flaw in `changeExcluded()` Function

- **Severity**: Medium
- **Description**: The `changeExcluded()` function doesn't check if the address is already excluded befo
- **Impact**: Users might be charged unnecessary fees, leading to financial losses.
- **Mitigation**: Implement a check to ensure that the `changeExcluded()` function only deducts the fee if

8. Outdated Solidity Version

- **Severity**: Medium
- **Description**: The contract uses Solidity version 0.6.4, which is no longer recommended for deploym
- **Impact**: The use of outdated Solidity versions could expose the contract to potential vulnerabilities th
- **Mitigation**: Upgrade the contract to a supported and more secure Solidity version.

9. Potentially Vulnerable Use of `call.value`

- **Severity**: Low
- **Description**: The contract uses `call.value` to send Ether, which could pose a security risk if the rece
- **Impact**: Funds might be lost if the call fails.
- **Mitigation**: Consider using the safer `transfer()` function instead of `call.value` to send Ether. This fu

Conclusion

Overall, the Vether3 smart contract exhibits several vulnerabilities, with some requiring immediate attentio