CERTORA

Formal Verification of CompoundStrategy

Summary

This document describes the specification and verification of CompoundStrategy from SushiSwap using Certora Prover. The work started at an early development stage, before manual testing, and was undertaken from April 11 - April 25, 2021. The latest commit that was reviewed and run through the Certora Prover was **76d7238970726b576779bb2483f0fd8421d31652**.

The scope of our verification was the CompoundStrategy contract. The contract is an investing strategy for a particular token that is deposited in BentoBox and supported by Compound protocol. The strategy doesn't hold any tokens, all are invested using the Compound protocol.

The Certora Prover proved that the implementation of the CompoundStrategy is correct with respect to the formal rules written by the SushiSwap and the Certora teams. During the verification process, the Certora Prover discovered issues in the code listed in the table below. All issues were promptly corrected, and the fixes were verified to satisfy the specifications up to the limitations of the Certora Prover. The Certora development team is currently handling these limitations. The next section formally defines high-level specifications.

All the rules are publically available in a public github:
https://github.com/sushiswap/bentobox/tree/master/spec

Certora Prover verification results:

1. CompondStrategy Rules:
    - All rules
    - exitRevert (on a simplified version)

2. BentoBox Rules (verified on BentoBox with CompoundStrategy):
    - totalAssetsAfterFlashLoan
    - additivity
    - validDecreaseToBalanceOf
    - solvency
    - NoChangeToOthersBalances

List of Main Issues Prevented By Certora Prover

Severity: Critical **Status:** Fixed

Issue:	Loss of assets in BentoBox
Rules Broken:	allTokensAreInvested, integrityHarvest
Description:	When harvesting the profits, cTokens are transferred to BentoBox instead of the invested token.
Fix:	Transfer the invested token instead of cToken.

Severity: Critical **Status:** Fixed

Issue:	Loss of assets in BentoBox
Rules Broken:	allTokensAreInvested, integrityExit
Description:	When exiting the strategy, all the tokens are passed to the owner instead of to BentoBox.
Fix:	Transfer to BentoBox instead of the owner.

Disclaimer

The Certora Prover takes as input a contract and a specification and formally proves that the contract satisfies the specification in all scenarios. Importantly, the guarantees of the Certora Prover are scoped to the provided specification, and any cases not covered by the specification are not checked by the Certora Prover.

We hope that this information is useful, but provides no warranty of any kind, expressed or implied. The contents of this report should not be construed as a complete guarantee that the contract is secure in all dimensions. In no event shall Certora or any of its employees be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the results reported here.

Notations

1. ✔️ indicates the rule is formally verified on the latest commit.
2. ✔️* indicates that the rule is verified on a simplified version.
3. We use Hoare triples of the form {p} C {q}, which means that if the execution of program C starts in any state satisfying p, it will end in a state satisfying q. In Solidity, p is similar to require, and q is similar to assert.
4. The syntax {p} ($C_1 \sim C_2$) {q} is a generalization of Hoare rules, called relational properties. {p} is a requirement on the states before C_1 and C_2, and {q} describes the states after their executions. Notice that C_1 and C_2 result in different states. As a special case, $C_1 \sim$op C_2, where op is a getter, indicating that C_1 and C_2 result in states with the same value for op.

Verification of CompoundStrategy

The CompoundStrategy is associated with a particular token from the BentoBox in order to invest it, on behalf of BentoBox, in the Compound protocol**.**

- **BentoBox** interacts with the strategy to invest, withdraw, harvest, and exit the strategy.
- **Owner** is responsible to harvest profits in the form of Compound Governance Token (COMP). BentoBox owner can decide to exit the strategy, which gives the Strategy owner control over the contract.

The contract CompoundStrategy is verified against the following properties. In addition, it is also checked with the rules written by the Certora team and SushiSwap for BentoBox using the Certora Prover.

Data Structures/Fields

1. **token**
   The ERC20 token being investing in this strategy.
2. **cToken**
   The compound investment token used in this strategy.
3. **exited**
   A boolean flag that keeps track if the strategy is exited or not, or in other words, if the exit method is called on the strategy.

Functions

All functions return, in addition to other values, a boolean flag that is true when the function succeeded and false when it reverts.

1. **skim(uint256 amount) : bool**
   Invests all assets, called after a deposit is performed from BentoBox.

2. **withdraw(uint256 amount): uint256**
   Withdraws assets and transfers them to the BentoBox. Returns the actual amount transferred, which can differ from amount due to rounding errors.

3. **harvest(uint256 balance, address sender): int256**
   Transfers to BentoBox any positive profit made by the strategy on balance (from the caller's view).

   After a harvest operation, the strategyCurrentAssets must match the strategy balance passed in plus the returned difference. (The BentoBox will update the balance with the returned difference.)

4. **exit(uint256 balance): (int256, bool)**
   Withdraws all assets and transfers them to the BentoBox. Returns the difference between balance (from the BentoBox's view) and the actual amount transferred, and a flag indicating if reverted.

Properties

1. **Strategy has all tokens invested ✔️** (rule: allTokensAreInvested)
   At any given time, except during function execution, the CompoundStrategy contract doesn't hold any tokens, all are invested in the Compound protocol or passed back to BentoBox.

   ```
   { } op { token.balanceOf(CompoundStrategy) = 0 }
   ```

2. **No one except the owner can use the strategy after it has been exited**

   a. **The exited data field is never false once it is true ✔️** (rule: onceExitedIsTrueThenItIsNeverFalse)

   ```
   { exited = true } op { exited = true }
   ```

   b. **Methods revert if the strategy has been exited except if they are called by the owner ✔️** (rule: ifExitedIsTrueThenMethodsRevertExceptOwner)

   ```
   { exited = true ∧ u != owner} success = op_u { !success }

   where op_u is any operation performed by user u.
   ```

3. **Integrity of harvest ✔️** (rule: integrityHarvest)
   Transfers excess tokens above balance. On a positive profit, BentoBox's balance increases and the return value is the profit above balance. On a negative profit, the negative profit is returned.

   ```
   { balanceBefore = token.balanceOf(BentoBox) }

         amountAdded = harvest(balance, sender)

   { ((amountAdded > 0) ⇒
         (token.balanceOf(BentoBox) = balanceBefore + amountAdded)) ∧

   ((amountAdded ≤ 0) ⇒ (token.balanceOf(BentoBox) = balanceBefore)) }
   ```

4. **Integrity of withdraw ✔️** (rule: integrityWithdraw)
   A withdraw operation increases the BentoBox's balance by the withdrawn amount.

   ```
   { balanceBefore = token.balanceOf(BentoBox) }

         amountAdded = withdraw(amount)

   { token.balanceOf(BentoBox) = balanceBefore + amountAdded }
   ```

5. **Integrity of exit** ✔️ (rule: integrityExit)
   The exit operation transfers all of the strategy's assets to BentoBox.

```
{ balanceBefore = token.balanceOf(BentoBox) }

      amountAdded = exit(balance)

{ token.balanceOf(BentoBox) = balanceBefore + balance + amountAdded }
```

6. **Exit does not revert** ✔️* (rule: exitRevert)
   The BentoBox should be able to exit from the strategy and withdraw all possible assets.

```
{ } (x, reverted) = exit(b) { !reverted }
```

   * assuming that the Compound protocol methods don’t revert