certora

Formal Verification Final Report

Silo Leverage

July 2025

Prepared for Silo

Table of Contents

Project Summary

Project Scope

Project Name	Repository (link)	Initial Commit Hash	Final Commit Hash	Platform
Silo	https://github.com/silo-finance/silo-contracts-v2	af2ba96	66a08a3	EVM

Project Overview

This document describes the specification and verification of the **Silo Leverage Module** using the Certora Prover. The work was undertaken from **the 7th of July** to the **23rd of July**

The following contract list is included in our scope:

```
silo-core/contracts/leverage/modules/GeneralSwapModule.sol
silo-core/contracts/leverage/modules/LeverageTxState.sol
silo-core/contracts/leverage/modules/RevenueModule.sol
silo-core/contracts/leverage/LeverageUsingSiloFlashloan.sol
silo-core/contracts/leverage/LeverageUsingSiloFlashloanWithGeneralSwap.sol
```

The list of additional contracts after the fix:

```
silo-core/contracts/leverage/modules/RescueModule.sol
silo-core/contracts/leverage/LeverageRouter.sol
```

Protocol Overview

The Silo leverage module provides users with the ability to amplify their exposure to assets within Silo's lending markets through automated flashloan-based leverage operations. Users can open leveraged positions by borrowing additional capital, converting it to their desired asset, and using the combined position as collateral for the borrowed funds. Users can also close their leveraged positions by unwinding the collateral back to the borrowed asset and settling their debt. The module integrates with external DEX aggregators to facilitate efficient asset swaps.

Findings Summary

The table below summarizes the findings of the review, including type and severity details.

Severity	Discovered	Confirmed	Fixed
Critical	-	-	-
High	-	-	-
Medium	-	-	-
Low	-	-	-
Informational	-	-	-
Total	0		

Severity Matrix

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
			Likelihood	

Formal Verification

Verification Methodology {Solidity}

We performed verification of the **Silo Leverage Module** protocol using the Certora verification tool which is based on Satisfiability Modulo Theories (SMT). In short, the Certora verification tool works by compiling formal specifications written in the Certora Verification Language (CVL) and **Silo**'s implementation source code written in Solidity.
More information about Certora's tooling can be found in the Certora Technology Whitepaper.

If a property is verified with this methodology, it means the specification in CVL holds for all possible inputs. However, specifications must introduce assumptions to rule out situations that are impossible in realistic scenarios (e.g., to specify the valid range for an input parameter). Additionally, SMT-based verification is notoriously computationally difficult. As a result, we introduce overapproximations (replacing real computations with broader ranges of values) and underapproximations (replacing real computations with fewer values) to make verification feasible.

Rules: A rule is a verification task possibly containing assumptions, calls to the relevant functionality that are symbolically executed, and assertions that are verified on any resulting states from the computation.

Inductive Invariants: Inductive invariants are proved by induction on the structure of a smart contract. We use constructors as a base case, and consider all other (relevant) externally callable functions that can change the storage as step cases.
Specifically, to prove the base case, we show that a property holds in any resulting state after a symbolic call to the respective constructor. For proving step cases, we generally assume a state where the invariant holds (induction hypothesis), symbolically execute the functionality under investigation, and prove that after this computation, any resulting state satisfies the invariant.

Verification Notations

Formally Verified	The rule is verified for every state of the contract(s), under the assumptions of the scope/requirements in the rule.
Formally Verified After Fix	The rule was violated due to an issue in the code and was successfully verified after fixing the issue
Violated	A counter-example exists that violates one of the assertions of the rule.

General Assumptions and Simplifications

1. We assume a simplified version of Silo, CollateralToken, and DebtToken, based on the properties proven for the actual code.
2. We summarize the Clones.sol library of openzeppelin5 to reflect that function predictDeterministicAddress() returns the same value as cloneDeterministic().

Formal Verification Properties Overview

ID	Title	Impact	Status
P-01	Immutability of Leverage Contract	Ensures no loss of user assets	Verified
P-02	Usability of Leverage Contract	Ensures only the owner can use their Leverage contract	Verified
P-03	Integrity of calls to onFlashLoan	No unexpected calls to onFlashLoan()	Verified
P-04	Valid calls to onFlashLoan()	One can not create a corrupted call to onFlashLoan	Verified
P-05	Leverage contract can not have debt in Silo nor allowance to any EOA user	No bad account can be created is Silo on the leverage contract	Verified
P-06	Balance changed	No Silo not external balance change to other users	Verified
P-07	No asset-lose	Users can rescue redundant assets for their leverage contract	Verified

Detailed Properties

LeverageRouter.sol

Module Properties

P-01. Immutability of Leverage Contract.

Status: Verified	**Leverage contract can not change**

Rule Name	Status	Description	Link to rule report
userLeverageContractImmutable	Verified	*Once a leverage contract is set for a user, it can not be changed*	*Report*
predictIntegrity	Verified	*Function predictUserLeverageContract() is the actual userLeverageContract address stored (if not zero)*	*Report*

P-02. Usability of Leverage Contract.

Status: Verified	**Only a specific user can use a leverage contract. This is proved by summarizing the calls to leverage and collecting the msgSender argument**

Rule Name	Status	Description	Link to rule report
calledWithMsgSender	Verified	*Only the owner can use his leverage*	*Report*
uniqueness	Verified	*Each user has a distinct leverage contract address*	*Report*
predictRevert	Verified	*Function predictUserLeverageContract() should not revert*	*Report*

LeverageUsingSiloFlashloanWithGeneralSwap.sol

Module Properties

P-03. Integrity of calls to onFlashLoan	
Status: Verified	**Calls to onFalshLoan() are restricted to specific use in specific states. This property is checked on all functions in the scene (silo, tokens, generalSwapModule) with a summarization to onFalshLoan() to track the calling context.**

Rule Name	Status	Description	Link to rule report
noCallsToFlashLoan	Verified	*Only specific top-level functions from the can call onFlahsLoan()*	*Report*
onFlashLoanReverts	Verified	*Function onFlahsLoan() reverts if called directly as a top-level function*	*Report*
siloFlashLoanReverts	Verified	*Function Silo.FlashLoan() reverts if called as top-level and attempts to call Leverage.onFlashLoan*	*Report*

P-04. Valid calls to onFlashLoan()

Status: Verified	**Function onFlashLoan() is reached on valid states only. This property is checked on all functions in the scene (silo, tokens, generalSwapModule) with a summarization to onFalshLoan() to track the calling context.**

Rule Name	Status	Description	Link to rule report
validCallsToFlashLoan_close	Verified	*Function closeLeveragePosition() reaches onFalshLoan() when:* *- msg.sender is Router* *- varaible `_txMsgSender` is the msgSender argument to closeLeveragePosition()* *- the msg.sender to onFalshLoan() is the expected silo according to the `closeArgs` arguement*	*Report*
validCallsToFlashLoan_open	Verified	*Function openLeveragePosition() reaches onFalshLoan() when:* *- msg.sender is Router* *- varaible `_txMsgSender` is the msgSender argument to openLeveragePosition()* *- the msg.sender to onFalshLoan() is the expected silo according to the `flashArgs` arguement*	

P-05. Leverage contract can not have debt in Silo nor allowance to any EOA user

Status: Verified	**No debt on the leverage contract. This property is checked on all functions in the scene (silo, tokens, generalSwapModule) with a summarization to onFalshLoan() to track the calling context.**

Rule Name	Status	Description	Link to rule report
noDebtOnLeverage	Verified	*Leverage contract can not have debt in Silo nor allowance to any EOA user*	*Report*

P-06. Balance changed

Status: Verified	**By using a leverage contract, others' balances can not change. This property is checked on all functions of LeverageUsingSiloFlashloanWithGeneralSwap.**

Rule Name	Status	Description	Link to rule report
noChangeToBalances_functions	Verified	*Only specific top-level functions can change balance in tokens*	*Report*
noLeverageDebtInSilo	Verified	*No debt on the account of a Leverage contract*	*Report*
balanceOfOther_close_open	Verified	*Function closeLeveragePosition() and openLeveragePosition() do not decrease assets in token or collateral token, nor increase debt of other users than the msgSender*	*Report*
integrityOpenLeverage	Verified	*Function openLeveragePosition() can increase the debt and collateral of the user*	*Report*

integrityCloseLeverage	Verified	*Function closeLeveragePosition() can decrease the debt and the collateral of the user*	*Report*

P-07. No asset-lose

Status: Verified	**User's assets are not lost.**

Rule Name	Status	Description	Link to rule report
ownerAndOnlyOwnerCanRescue	Verified	*Rescue tokens can be executed, but onlyby the leverageUSer. On a successful call, all tokens are transferred to the leverageUser*	*Report*
ownerAndOnlyOwnerCanRescueEth	Verified	*Rescue Eth can be executed but only by the leverageUSer. On successful call all eth are transferred to the leverageUser*	*Report*

Disclaimer

Even though we hope this information is helpful, we provide no warranty of any kind, explicit 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.

About Certora

Certora is a Web3 security company that provides industry-leading formal verification tools and smart contract audits. Certora's flagship security product, Certora Prover, is a unique SaaS product that automatically locates even the most rare & hard-to-find bugs on your smart contracts or mathematically proves their absence. The Certora Prover plugs into your standard deployment pipeline. It is helpful for smart contract developers and security researchers during auditing and bug bounties.

Certora also provides services such as auditing, formal verification projects, and incident response.