Security Audit Report

Hatom ush-isolated-lending (2)

MultiversX smart contract

by ARDA

on April 29, 2025

Table of Contents

Disclaimer

The report makes no statements or warranties, either expressed or implied, regarding the security of the code, the information herein or its usage. It also cannot be considered as a sufficient assessment regarding the utility, safety and bugfree status of the code, or any other statements.

This report does not constitute legal or investment advice. It is for informational purposes only and is provided on an "as-is" basis. You acknowledge that any use of this report and the information contained herein is at your own risk. The authors of this report shall not be liable to you or any third parties for any acts or omissions undertaken by you or any third parties based on the information contained herein.

Terminology

Code: The code with which users interact.

Inherent risk: A risk for users that comes from a behavior inherent to the code's design.

Inherent risks only represent the risks inherent to the code's design, which are a subset of all the possible risks. **No inherent risk doesn’t mean no risk.** It only means that no risk inherent to the code's design has been identified. Other kind of risks could still be present. For example, the issues not fixed incur risks for the users, or the upgradability of the code might also incur risks for the users.

Issue: A behavior unexpected by the users or by the project, or a practice that increases the chances of unexpected behaviors to appear.

Critical issue: An issue intolerable for the users or the project, that must be addressed.

Major issue: An issue undesirable for the users or the project, that we strongly recommend to address.

Medium issue: An issue uncomfortable for the users or the project, that we recommend to address.

Minor issue: An issue imperceptible for the users or the project, that we advise to address for the overall project security.

Objective

Our objective is to share everything we have found that would help assessing and improving the safety of the code:

1. The **inherent risks** of the code, labelled R1, R2, etc.
2. The **issues** in the **code**, labelled C1, C2, etc.
3. The **issues** in the **testing** of the code, labelled T1, T2, etc.
4. The **issues** in the **other** parts related to the code, labelled O1, O2, etc.
5. The **recommendations** to address each issue.

Audit Summary

Initial scope

- **Repository:** https://github.com/HatomProtocol/hatom-isolated-lending-protocol
- **Commit:** e729384574f44cee2ad62309d02692598b6a7575
- **MultiversX smart contract path:** ./isolated-lending-protocol/

Final scope

- **Repository:** https://github.com/HatomProtocol/hatom-isolated-lending-protocol
- **Commit:** 400e50d64f2ad5dd9d78d8ff40670ff30636547e
- **MultiversX smart contract path:** ./isolated-lending-protocol/

7 inherent risks in the final scope

0 issue in the final scope

4 issues reported in the initial scope and 0 remaining in the final scope:

Severity	Reported			Remaining		
	Code	Test	Other	Code	Test	Other
Critical	0	0	0	0	0	0
Major	2	0	0	0	0	0
Medium	2	0	0	0	0	0
Minor	0	0	0	0	0	0

Inherent Risks

R1: 1 USH minted by an USH isolated lending module might be backed by less than $1 worth of collateral.

This is because there is a trust that the oracles providing prices and that the liquidation bots are active and work properly:

- If for any reason the prices returned by the oracles are erroneous, then the real dollar value of the collateral of USH borrowers might be smaller than the amount of borrowed USH.
- If for any reason, while some users are insolvent, there are no sufficiently active liquidators to execute liquidations or liquidations fail to be executed (e.g. because prices fail to be obtained from the oracles), then the amount of borrowed USH might continue to increase and exceed the dollar value of the collateral of USH borrowers.

R2: The solvency of a user might be incorrectly assessed, possibly leading to bad debt or to the liquidations of solvent users.

This is because the solvency of a user depends on the value of his collateral relative to the value of his debt, and the prices of these tokens are obtained from external oracles which might make mistake and return incorrect prices. Consequently:

- Insolvent users might be deemed solvent: This would prevent the liquidations of these users, and would also allow them to borrow assets or withdraw collateral. This could then further lead to bad debt, i.e. a situation where USH is not sufficiently backed by collateral, increasing the risk that the dollar value of USH drops below 1.
- Solvent users might be deemed insolvent: This could result in unexpected liquidations, possibly making borrowers lose funds.

R3: Even if a user is solvent, his collateral might be seized by an external account who repays the user's debt.

Namely, in the USH isolated lending modules, there is a so-called "redemption mechanism" allowing anyone to seize the collateral of solvent users by repaying their debt.

Unlike in liquidations, in a redemption the user's loan-to-value is supposed to decrease: the amount of USH debt repaid to him is greater or equal to the dollar value of the seized collateral. However, this is not guaranteed because the dollar value of the collateral is obtained from external oracles which might make mistakes.

Finally, redemptions can be triggered anytime even if the dollar value of USH is above 1, although in this case they should not be profitable to redeemers.

R4: Users might have to pay a borrowing fee when borrowing USH.

This is because, at the time a user borrows some USH, his debt is increased not only by the amount being borrowed, but also by an extra borrowing fee. Therefore, in order to later withdraw all his collateral, the user must reimburse the amount borrowed and the borrowing fee.

The borrowing fee depends on the total amount of USH recently redeemed in the smart contract, and is capped to a maximum of 5%.

Example: Let's say significant redemptions occurred and the borrowing fee of 5%. Alice borrows 100 USH, and her debt is then $100 + 5\% \times 100 = 105$ USH. If later she wants to get back her collateral, she will have to reimburse the 100 USH she borrowed, and the additional 5 USH from the borrowing fee.

R5: Users might not be able to acquire the USH needed for repaying their debt.

This is because borrowers must repay their whole debt in USH, which includes the USH minted for them as well as the borrowing fee which was kept in the

smart contract. Therefore, in order to repay their debt, they must acquire USH, and for this they have two options:

1. Minting new USH: At the time of this audit, users can only mint new USH by making new borrows. However, by making a new borrow, users would increase their total debt, so this approach does not let them repay their current debt.
2. Finding USH in the circulating supply: However, the part of the circulating supply that users can acquire might be smaller than their total debt, in particular if the borrowing fees were not withdrawn by the admin or not made acquirable to users. In turn, it might be impossible for all users to repay their debt unless other users borrow USH and make it acquirable.

R6: Users who deposit liquid staking tokens or HTokens as collateral stop earning interests from these tokens.

This is because, when users deposit liquid staking tokens (sEGLD or sTAO) or HTokens (HsEGLD or HsTAO) as collateral in the smart contract, the staking interests and lending interests are redirected to the Hatom protocol.

R7: Users might not be able to withdraw collateral at all or to withdraw collateral in the deposited form.

This is because:

1) When a user deposits collateral in base tokens (EGLD or TAO) or liquid staking tokens (sEGLD or sTAO), his tokens are converted to liquid staking tokens and then to supply tokens in Hatom money markets (HsEGLD or HsTAO).

2) When a user withdraws collateral, he has the following possibilities:

- He can decide to withdraw collateral in supply tokens, which is always possible.
- He can decide to withdraw collateral in liquid staking tokens, but this might be impossible if there is an insufficient supply of liquid staking tokens in the

money market.

- He can decide to withdraw collateral in base tokens, but this requires first converting supply tokens to liquid staking tokens, which might be impossible if there is an insufficient supply of liquid staking tokens in the money market. Moreover, in the case of EGLD collateral, the user would not receive EGLD directly, but rather an unstaking NFT from the liquid staking protocol, which can be converted to EGLD only after waiting an unbonding period.

Code Issues & Recommendations

Since the code is not open-source, only the remaining issues are published.