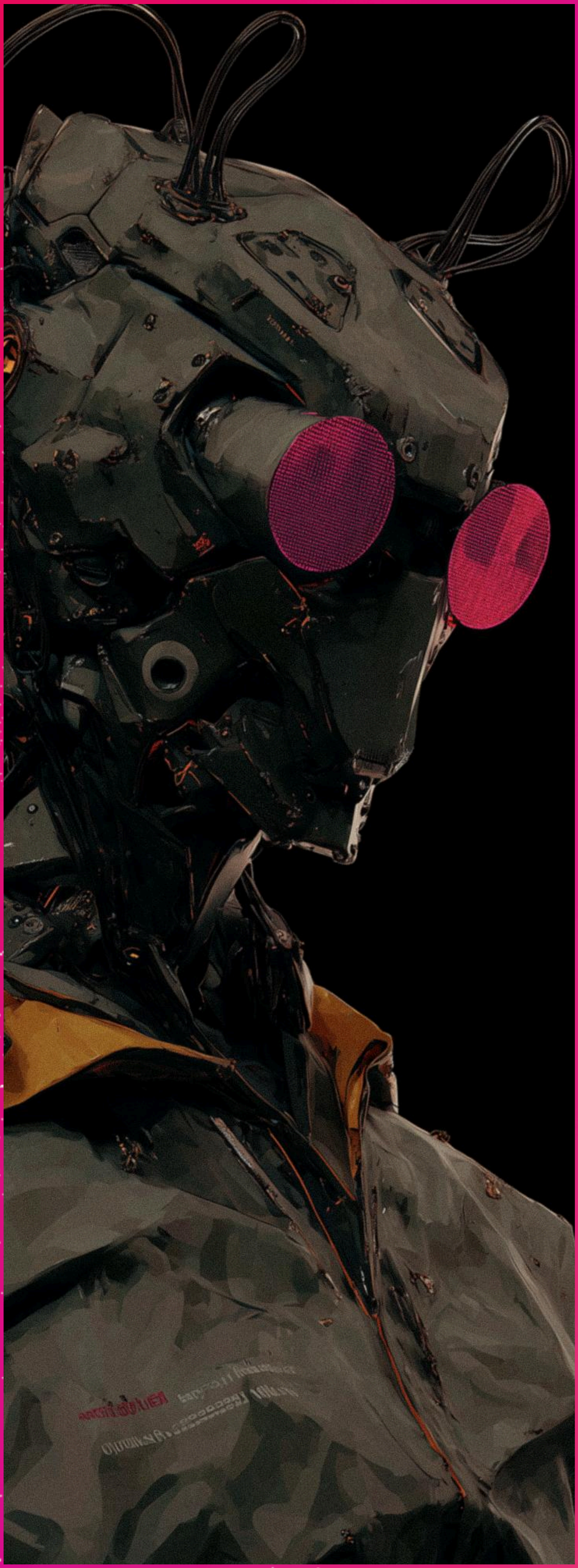
IMMUNEFI AUDIT

Immunefi / Helios Finance



DATE	June 11, 2025
AUDITOR	**qckhp**, Security Researcher
REPORT BY	**gmhacker**, Immunefi Head of Security

ABOUT IMMUNEFI

Immunefi is the leading onchain security platform, having directly prevented hacks worth more than $25 billion USD. Immunefi security researchers have earned over $120M USD for responsibly disclosing over 4,000 web2 and web3 vulnerabilities, more than the rest of the industry combined.

Through Magnus, Immunefi delivers a comprehensive suite of best-in-class security services through a single command center to more than 300 projects — including Sky (formerly MakerDAO), Optimism, Polygon, GMX, Reserve, Chainlink, TheGraph, Gnosis Chain, Lido, LayerZero, Arbitrum, StarkNet, EigenLayer, AAVE, ZKsync, Morpho, Ethena, USDT0, Stacks, Babylon, Fuel, Sei, Scroll, XION, Wormhole, Firedancer, Jito, Pyth, Eclipse, PancakeSwap and many more.

Magnus unifies SecOps across the entire onchain lifecycle, combining Immunefi's market leading products and community of elite security researchers with a curated set of the very best security products and technologies provided by top security firms — including Runtime Verification, Dedaub, Fuzzland, Nexus Mutual, Failsafe, OtterSec and others.

Magnus is powered by Immunefi's proprietary vulnerabilities dataset — the largest and most comprehensive in web3, ensuring that security leaders and teams have the best possible tools for identifying and mitigating life threats before they cause catastrophic harm, all while reducing operational overhead and complexity.

Learn how you can benefit too at immunefi.com.

TERMINOLOGY

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- **Likelihood** represents the likelihood of a finding to be triggered or exploited in practice
- **Impact** specifies the technical and business-related consequences of a finding
- **Severity** is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.

LIKELIHOOD	IMPACT		
	HIGH	MEDIUM	LOW
CRITICAL	Critical	Critical	High
HIGH	High	High	Medium
MEDIUM	Medium	Medium	Low
LOW	Low		
NONE	None		

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

EXECUTIVE SUMMARY

Over the course of 3 days in total, Helios Finance engaged with Immunefi to review the CollateralSwapperV1 repository. In this period of time a total of 5 issues were identified.

SUMMARY

Name	Helios Finance
Audit Repository	https://github.com/dhwndud408/helios-contract-v1
Audit Commit	e2eb69cc2bbca3f25a0197028b21d6c4d35de59a
Fix Commit	fb7a14999141cf463e03effa18a7d900ad583602
Type of Project	Lending
Audit Timeline	May 28th - May 30th
Fix Period	June 6th

ISSUES FOUND

Severity	Count	Fixed	Acknowledged
Critical	0	0	0
High	0	0	0
Medium	2	2	0
Low	2	2	0
Insights	1	1	0

CATEGORY BREAKDOWN

Bug	4
Gas Optimization	0
Informational	1

FINDINGS

IMM-MED-01

`swapCollateral` potentially could be sandwiched

Id	IMM-MED-01
Severity	Medium
Category	Bug
Status	Fixed

Description

In `CollateralSwapperV1's _swapTokens` function, the slippage tolerance is calculated on a chain using `uniswapRouter.getAmountsOut(amount, path)`. This exposes the user to sandwich attacks, as attackers can manipulate the Uniswap pool price to cause slippage, leading to loss of funds.

Recommendation

Not sure if it's possible to frontrun on MIDL network, but as its EVM based network, there is a possibility so I would Modify the swapCollateral to accept a user-provided `minAmountOut` parameter instead of calculating it on-chain.

IMM-MED-02

Potential loss of funds if `msg.value` exceeds `supplyAmount` in ChangeCollateral function

Id	IMM-MED-02
Severity	Medium
Category	Bug
Status	Fixed

Description

If in `changeCollateral` function the supplied asset is native BTC, and excess BTC (msg.value - supplyAmount) sent by the user, the excess amount is not refunded, causing it to be stuck in the contract.

Recommendation

Modify `changeCollateral` function to check if `msg.value` equals `supplyAmount` value to prevent excess native BTC from being sent.

IMM-LOW-01

Account for flash loan fee in `healthFactorBefore` check in swapCollateral function

Id	IMM-LOW-01
Severity	LOW
Category	Bug
Status	Fixed

Description

In the `CollateralSwapperV1` contract's swapCollateral function, the `require(healthFactorBefore >= HEALTH_FACTOR_MIN)` check ensures the user's health factor is at least 1.0 before the swap.

However, this check does not account for the flash loan fee, which is deducted from the user's collateral and potentially pushing the health factor below the safe threshold and causing a revert during the transaction.

Recommendation

Modify the `swapCollateral` function to deduct the flash loan fee (afaik 0.05%) from `healthFactorBefore` value when checking against `HEALTH_FACTOR_MIN`

IMM-LOW-02

`block.timestamp` should not be used for swap deadlines

Id	IMM-LOW-02
Severity	Low
Category	Bug
Status	Fixed

Description

The contract should accept a `user-provided` deadline parameter, passing it to Uniswap, which is reverting the transaction if the deadline is past the current `block.timestamp`.

Recommendation

Modify `swapCollateral` function to accept a user-provided deadline parameter instead of calculating it.

IMM-INSIGHT-01

`block.timestamp` should not be used for swap deadlines

Id	IMM-INSIGHT-01
Severity	INSIGHT
Category	Informational
Status	Fixed

Description

In `CollateralSwapperV1`'s executeOperation function, the initiator parameter from the pools flash loan callback is not checked against address(this). This could risk malicious reentrancy via malicious tokens via path param, where an attacker could re-trigger `executeOperation` via manipulated input parameters, causing double swaps of collateral or other undiscovered issues.

Recommendation

As there are no discovered direct issues, this issue is logged as insight, but I would add a check in executeOperation to ensure initiator equals address(this) to prevent any issues.