GUARDIAN

Synthetix

BFP Market #2

Security Assessment

October 28th, 2024

Summary

Audit Firm Guardian

Prepared By Daniel Gelfand, Owen Thurm, Mark Jonathas, Devtooligan, Kiki, Dogus Kose, Michael Lett

Client Firm Synthetix

Final Report Date October 28, 2024

Audit Summary

Synthetix engaged Guardian to review the security of its synthetix. From the 16th of June to the 1st of July, a team of 7 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Issues Detected Throughout the engagement 13 High/Critical issues were uncovered and promptly remediated by the Synthetix team. Several issues impacted the fundamental behavior of the protocol, following their remediation Guardian believes the protocol to uphold the functionality described for the lending protocol product.

For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Ethereum**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/synthetix-bfp-fuzzing

Table of Contents

Project Overview

Project Summary

Project Name	Synthetix
Language	Solidity
Codebase	https://github.com/Synthetixio/synthetix-v3
Commit(s)	Initial: 42ea139bff48b537b2ca85c02aa9bead84117407 Final: 7e9157e8eda2e9f3df84a9c6b65d701484bf3faa

Audit Summary

Delivery Date	October 28, 2024
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
🔴 Critical	2	0	0	0	0	2
🟠 High	10	0	0	2	0	8
🟡 Medium	19	0	0	14	0	5
🟢 Low	18	0	0	6	0	12

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	● Critical	● High	● Medium
Likelihood: *Medium*	● High	● Medium	● Low
Likelihood: *Low*	● Medium	● Low	● Low

Impact

High	Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.
Medium	A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.
Low	Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High	The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.
Medium	An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.
Low	Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Invariants Assessed

During Guardian's review of Synthetix, fuzz-testing with Echidna was performed on the protocol's main functionalities. Given the dynamic interactions and the potential for unforeseen edge cases in the protocol, fuzz-testing was imperative to verify the integrity of several system invariants.

Throughout the engagement the following invariants were assessed for a total of 140,000,000+ runs with a prepared Echidna fuzzing suite.

ID	Description	Tested	Passed	Remediation	Run Count
MGN-01	Position is never liquidatable after a successful margin withdraw	✅	✅	✅	140M+
MGN-02	A modify collateral call will always revert for an account with a flagged position	✅	✅	✅	140M+
MGN-03	A modify collateral call will always revert for an account that has a pending order	✅	✅	✅	140M+
MGN-04	If an account's collateral is 0, then the account's debt must also be 0	✅	✅	✅	10M+
MGN-05	depositedCollaterals array should be adjusted by amount of collateral modified	✅	✅	✅	10M+
MGN-06	If sUSD collateral modified, minimumCredit should be updated by that amount	✅	✅	✅	10M+
MGN-07	There should be no reportedDebt if all collateral has been withdrawn and skew=0	✅	✅	❌	10M+
MGN-08	sum of collateral token values should be the totalCollateralValueUsd stored in the market	✅	✅	✅	10M+
MGN-09	All accountMargin.collaterals should be 0 after call to withdrawAllCollateral	✅	✅	✅	10M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
MGN-10	All accountMargin.collaterals are 0 after call to withdrawAllCollateral()	✅	✅	✅	10M+
MGN-11	Market collateral should decrease by amount of collateral user had deposited before withdrawing all collateral	✅	✅	✅	10M+
MGN-12	User cannot withdraw more non-sUSD collateral than they deposited	✅	✅	✅	10M+
MGN-13	After call to payDebt , user collateral should decrease	✅	✅	✅	10M+
MGN-14	After call to payDebt, market totalTraderDebt should decrease	✅	✅	✅	10M+
ORD-01	If an account has an order committed, a subsequent commit order call will always revert	✅	✅	✅	140M+
ORD-02	The sizeDelta of an order is always 0 after a successful settle order call	✅	✅	✅	140M+
ORD-03	An order immediately after a successful settle order call, is never liquidatable	✅	✅	✅	140M+
ORD-04	If a user successfully settles an order, their sUSD balance is strictly increasing	✅	✅	✅	140M+
ORD-05	The sUSD balance of a user that successfully cancels an order for another user is strictly increasing	✅	✅	✅	50M+
ORD-06	The minimum credit requirement must be met after increase order settlement	✅	❌	✅	10M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
ORD-07	Utilization is between 0% and 100% before and after order settlement	✅	✅	✅	10M+
ORD-08	non-SUSD collateral should stay the same after profitably settling order	✅	✅	✅	10M+
ORD-09	Should always give premium when increasing skew and discount when decreasing skew	✅	✅	✅	10M+
ORD-10	market.currentUtilizationAccruedComputed only increases	✅	✅	✅	10M+
ORD-11	market.reportedDebt == positions.sum(p.collateralUsd + p.pricePnL + p.pendingFunding - p.pendingUtilization - p.debtUsd)	✅	❌	❌	<1M
ORD-12	An account should not be liquidatable by margin only after order settlement	✅	✅	✅	10M+
ORD-13	An account should not be liquidatable by margin only after order cancelled	✅	✅	✅	10M+
ORD-14	Market size should always be the sum of individual position sizes	✅	✅	✅	10M+
ORD-15	Position should not be liquidatable after committing an order	✅	✅	✅	10M+
ORD-16	Position should not be liquidatable after cancelling an order	✅	❌	✅	<1M
ORD-17	Position should not be liquidatable after cancelling a stale order	✅	✅	✅	10M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
LIQ-01	isPositionLiquidatable never reverts	✅	✅	✅	140M+
LIQ-02	If a position is flagged for liquidation before any function call, the position after is always either flagged for liquidation, or no longer exists	✅	✅	✅	50M+
LIQ-03	remainingLiquidatableSizeCapacity is strictly decreasing immediately after a successful liquidation	✅	✅	✅	50M+
LIQ-04	If a user gets successfully flagged, their collateral will always be 0	✅	✅	✅	50M+
LIQ-05	The sUSD balance of a user that successfully flags a position is strictly increasing	✅	✅	✅	50M+
LIQ-06	The sUSD balance of a user that successfully flags a position increases less or equal to maxKeeperFee	✅	✅	✅	50M+
LIQ-07	User should not be able to gain more in keeper fees than collateral lost in liquidatePosition	✅	✅	✅	10M+
LIQ-08	A user can be liquidated if minimum credit is not met	✅	✅	✅	10M+
LIQ-09	All account margin collateral should be removed after full liquidation	✅	✅	✅	10M+
LIQ-10	Flagged positions should be liquidated even if they have a health factor > 1	✅	✅	✅	10M+
LIQ-11	Market deposited collateral should decrease after full liquidation by the account margin that was liquidated	✅	✅	✅	10M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
LIQ-12	If a position has position.size == 0, flagger should be set to address(0)	✅	✅	✅	10M+
LIQ-13	After user position is flagged, user should have 0 collateral value	✅	✅	✅	10M+
LIQ-14	After user is flagged, market collateral value decreases by user’s collateral value	✅	✅	✅	10M+
LIQ-15	User should not be able to gain more in keeper fees than collateral lost in liquidateMarginOnly	✅	❌	✅	<1M
LIQ-16	Total market collateral value must decrease on successful liquidation	✅	✅	✅	10M+
ACT-01	Position should not be liquidatable by margin only after splitting	✅	❌	✅	<1M
ACT-02	Position should not be liquidatable after splitting	✅	✅	✅	10M+
ACT-03	Position should not be liquidatable by margin only after merging	✅	✅	✅	10M+
ACT-04	Position should not be liquidatable after merging	✅	✅	✅	10M+

Findings & Resolutions

ID	Title	Category	Severity	Status
C-01	Profitable Self Liquidations	Gaming	● Critical	Resolved
C-02	mergeAccount Abused For Profit	Gaming	● Critical	Resolved
H-01	Liquidity Providers Can Control Markets	Gaming	● High	Resolved
H-02	Split Order May Be Used To Liquidate Accounts	Gaming	● High	Resolved
H-03	Accounts Can Be Liquidated Upon Order Settlement	Validation	● High	Resolved
H-04	PnL Wiped Out On Breaking Even	Logical Error	● High	Resolved
H-05	DoS Via validateMinimumCredit	DoS	● High	Resolved
H-06	Liquidations Errantly Adjust Funding Correction	Logical Error	● High	Acknowledged
H-07	LPs Can Game Liquidations Via mintUsd	Gaming	● High	Acknowledged
H-08	Large Orders Can Exceed minimumCredit	Logical Error	● High	Resolved
H-09	IM Validation Incorrectly Attributes Price Impact	Logical Error	● High	Resolved
H-10	DebtCorrection Uses Incorrect Price For Funding	Logical Error	● High	Resolved
M-01	Reward Distributor Is Not Future-Proof	Logical Error	● Medium	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
M-02	Risk Free Trades Via Pay Debt	Gaming	● Medium	Resolved
M-03	Orders Can Be Censored By Front-running	Gaming	● Medium	Acknowledged
M-04	Liquidation Fees Not Covered By non-sUSD Collateral	Gaming	● Medium	Acknowledged
M-05	Negative New Margin Is Not Recognized	Unexpected Behavior	● Medium	Acknowledged
M-06	Account Can Be Made Liquidateable By Canceling	Unexpected Behavior	● Medium	Resolved
M-07	Debt With Zero Collateral Is Unliquidateable	Unexpected Behavior	● Medium	Acknowledged
M-08	Pending Fees Altered By setMarketConfiguration	Logical Error	● Medium	Acknowledged
M-09	DoS To Keeper Transactions	DoS	● Medium	Acknowledged
M-10	Value Extraction Via Trustless Keepers	Gaming	● Medium	Acknowledged
M-11	Unexpected Liquidations Caused By Block.basefee	Logical Error	● Medium	Acknowledged
M-12	payDebt Affects Utilization Without Update	Logical Error	● Medium	Resolved
M-13	BFP Market Liquidations May Liquidate LPs	Unexpected Behavior	● Medium	Acknowledged
M-14	Inadequate Estimated Liquidation Costs	Logical Error	● Medium	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
M-15	Attacker Profits From Block Stuffing Orders	Gaming	● Medium	Acknowledged
M-16	Setting SkewScale Disrupts Markets	Warning	● Medium	Resolved
M-17	Utilization Fees Are Not Accurate To Liquidity Updates	Unexpected Behavior	● Medium	Acknowledged
M-18	Debt Is Not Reflected In CreditCapacity Until Repaid	Logical Error	● Medium	Acknowledged
M-19	Liquidation Prevented By Zero Vault Delegation	DoS	● Medium	Resolved
L-01	mergeAccounts Used To Steal A User's Position	Validation	● Low	Acknowledged
L-02	Settlement Hook Executions Not Validated	Validation	● Low	Acknowledged
L-03	Redundant Funding Velocity Computations	Optimization	● Low	Resolved
L-04	Unnecessary newSupportedCollaterals Array	Optimization	● Low	Resolved
L-05	Lacking Margin Configuration Safety Validations	Validation	● Low	Resolved
L-06	Lacking Distributor Configuration Safety Validations	Validation	● Low	Resolved
L-07	getHealthData Divide By Zero Revert	DoS	● Low	Resolved
L-08	Reported Debt Stepwise Jump Risks	Gaming	● Low	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
L-09	sUSD Collateral maxAllowable Can Be Exceeded	Unexpected Behavior	● Low	Acknowledged
L-10	Margin Requirements Do Not Use The Max Fee	Best Practices	● Low	Resolved
L-11	Settlement Hooks May Be Repeated	Validation	● Low	Resolved
L-12	Unused Flag	Optimization	● Low	Acknowledged
L-13	Unused Mm Parameter	Optimization	● Low	Resolved
L-14	Liquidation Rewards Are Not Bounded By The Minimum	Unexpected Behavior	● Low	Resolved
L-15	Settlement And Cancellation Fees Lack Fixed Margin	Unexpected Behavior	● Low	Resolved
L-16	Typo	Typo	● Low	Resolved
L-17	Typo	Typo	● Low	Resolved
L-18	Immediately Cancelable Orders Are Permitted	Validation	● Low	Acknowledged

C-01 | Profitable Self Liquidations

Category	Severity	Location	Status
Gaming	● Critical	MarginModule.sol	Resolved

Description PoC

When a user deposits, there is no validation that the amount that they deposit is enough to cover the fee for liquidateMarginOnly(). This allows a user to deposit 1 wei, and liquidate themselves immediately to realize a profit.

Below are the calculations based off current ETH prices:

Cost of Deposit	: 0.001633216 ETH
Cost of Liquidation	: 0.001899712 ETH
Collateral Cost	: 0.000000000000000001 ETH
Total Cost ETH	: 0.003656128000000001 ETH
Total Cost USD	: $12.4097 = 0.003656128000000001 ETH * $3,394.23
Reward sUSD	: 24.0608
Profit	: $11.6511

Recommendation

Verify that if the user is depositing, the collateral amount must be equal or greater than the highest possible marginLiquidationOnly reward. Additionally, validate that the amount of collateral left after withdrawing is sufficient to cover the highest possible marginLiquidationOnly reward if it is nonzero.

Resolution

Synthetix Team: The issue was resolved in PR#2185.

C-02 | mergeAccount Abused For Profit

Category	Severity	Location	Status
Gaming	● Critical	PerpAccountModule: 469	Resolved

Description PoC

When calling mergeAccount() the check to verify the size of the fromPositon and toPosition calls sameSide(). sameSide() will always return true if either size is 0.

This will then set entry price to the toPosition to 0, since the fromPositon was deleted in settleOrder(). Now the toPosition instantly has a positive PnL if the position is a long.

Recommendation

Revert if the fromPositon has 0 size in mergeAccount().

Resolution

Synthetix Team: The issue was resolved in PR#2199.

H-01 | Liquidity Providers Can Control Markets

Category	Severity	Location	Status
Gaming	● High	Global	Resolved

Description

Liquidity providers can mintUsd and burnUsd without any restrictions with an atomic action which will rebalance collaterals for markets.

The problem with these functions is that while it is rebalancing collaterals for markets, it doesn't validate if there is enough creditCapacity to back the markets after the action.

Hence LPs can mintUsd such that market's creditCapacity will drop below their minimumCredit which in turn will lead to not executable orders. Since minting and burning are interest-free actions, liquidity providers can use this system to lock and unlock markets whenever they want.

This can be used in several ways such as:
1. Risk free trades via preventing settleOrder calls by other keepers and only executing the order when it is profitable
2. Preventing some order's execution while executing some other orders
Among many other things.
With these vectors LP's can extract a significant amount from the BFP market.

Recommendation

Add _verifyNotCapacityLocked() to mintUsd() similar to how delegateCollateral() incorporates it so that LP's can't mint USD if it will invalidate market's minimumCredit checks.

Additionally consider adding a buffer between the minimumCredit defined by a market and the creditCapacity delegation mintUsd is allowed to leave markets at.

This buffer would be useful as an LP would still be able to mintUsd up to the minimumCredit limit right before an order becomes executable to prevent it’s execution, which can also lead to risk free trades and extractable value.

Resolution

Synthetix Team: The issue was resolved in PR#2328.

H-02 | Split Order May Be Used To Liquidate Accounts

Category	Severity	Location	Status
Gaming	● High	PerpAccountModule.sol	Resolved

Description PoC

An account may be split and it may be put into a state that is immediately liquidatable with liquidateMarginOnly. This is accomplished through the use of accounts with a small position and the use of a small proportion which could result in zero size due to rounding.

This attack would result in profits of approximately the amount of the liquidation fee. However, this attack could be repeated many times in the same block and in subsequent blocks until the splitting account was removed from the whitelist.

It is because splitAccount is a whitelisted function that this is a High severity rather than a Critical.

Recommendation

At the end of splitAccount, revert if either the to or from accounts are liquidatable with liquidateMarginOnly.

Resolution

Synthetix Team: The issue was resolved in PR#2208.

H-03 | Accounts Can Be Liquidated Upon Order Settlement

Category	Severity	Location	Status
Validation	● High	Order.sol	Resolved

Description PoC

An account can be put into a liquidatable state immediately upon settling an order. This would cause the immediate loss to a user of all funds that they could have instead withdrawn through `modifyColatteral`.

This is made possible by the lack of a check for `isLiquidatable` at the end of the `settleOrder` function. Even if the advice is followed in finding H-XX to ensure there is enough margin to cover the settlement fees, the account can still be liquidated upon settlement if the market were to move unfavorably.

Recommendation

Consider adding logic to the `validateTrade` function which will revert in case the new position including fees causes the account to become liquidatable.

Resolution

Synthetix Team: The issue was resolved in PR#2210.

H-04 | PnL Wiped Out On Breaking Even

Category	Severity	Location	Status
Logical Error	● High	Margin.sol: 105	Resolved

Description PoC

In the realizeAccountPnlAndUpdate function when the amountDeltaUsd is 0 the function early returns. However this early return will errantly leave debt in the account's margin while the position's unrealized profit, which is perfectly offsetting the debt, is lost.

The PnL lost by the trader this way is credited to the LPs as it is removed from the reportedDebt.

Recommendation

Assign the account's debt to 0 in the amountDeltaUsd == 0 early return case and account for this debt update in the market.totalTraderDebtUsd.

Resolution

Synthetix Team: The issue was resolved in PR#2187.

H-05 | DoS Via validateMinimumCredit

Category	Severity	Location	Status
DoS	● High	Global	Resolved

Description

In the `validateMinimumCredit` function the `delegatedCollateralValueUsd` is compared to the `minimumCredit`, however the `minimumCredit` for the `PerpMarket` includes the market sUSD collateral as this amount is included in the `creditCapacity` which is ultimately validated against the `minimumCredit`.

Therefore the `delegatedCollateralValueUsd` cannot directly be compared against the market's `minimumCredit`, as the `delegatedCollateralValueUsd` does not include the trader deposited sUSD amount.

One can deposit big chunks of sUsd as a collateral from BFP side via `modifyCollateral` such that all position increase orders will fail. It can also happen naturally without requiring a malicious intent because of wrong accounting.

Recommendation

Update the validation to `delegatedCollateralValueUsd + market.depositedCollateral[SYNTHETIX_SUSD] < minimumCredit.toInt()`.

Resolution

Synthetix Team: The issue was resolved in PR#2180.

H-06 | Liquidations Errantly Adjust Funding Correction

Category	Severity	Location	Status
Logical Error	● High	LiquidationModule.sol	Acknowledged

Description PoC

When liquidating positions with the liquidatePosition function, the debtCorrection is updated to account for the position's size decrease.

However the funding which has accrued over the period from the time range [flagPosition, liquidatePosition] is accounted for in the debt correction update.

As the liquidated position will not settle the funding received or paid to it's margin, this debtCorrection adjustment for the funding realized is unnecessary and perturbs the reportedDebt accounting of the market.

Recommendation

Consider adding a parameter to the updateDebtCorrection function that would allow the totalPositionPnl to be ignored from the result. In the case of liquidations post-flagging, neither the price PnL or the funding PnL is relevant.

Resolution

Synthetix Team: Acknowledged.

H-07 | LPs Can Game Liquidations Via mintUsd

Category	Severity	Location	Status
Gaming	● High	Global	Acknowledged

Description PoC

When flagging a position for liquidation all of the account's collateral is seized which will immediately reduce the reportedDebt. However the position's unrealized losses are still present in the reportedDebt before the liquidatePosition function has been called.

The account's collateral that was seized is intended to cover the position's losses, however upon a debt distribution and rebalancing of the pool the backing LPs debt will be decreased by the collateral removed but not yet increased by the liquidation of the position that is in a loss.

SIP-366, Asynchronous Delegation, aims to address a scenario in which LPs would be able to specifically target this intermediate liquidation mis-accounting in order to undelegate while the debt of their position has been deflated.

This addresses the vector where LPs would undelegate from the pool, however LPs may still mintUsd while they have this artificially lowered debt. This way an LP could mintUsd up to the c-ratio while their debt is suppressed, and then once the BFP account is liquidated the LPs position would be immediately under the c-ratio by a stepwise, and potentially significant, amount.

In some cases LPs may be able to mint more sUSD than they would lose from liquidation. As a result, the LP forces others in the vault to take on socialized debt and can even potentially make keeper fee profits from triggering the BFP liquidation and their own V3 liquidation in the same block.

Additionally, even with SIP-366 implemented LPs could frontrun the processIntentToDelegateCollateral function call to trigger a flagging of a position. Or process their own intent before flagging a position in BFP. Though these actions are not as guaranteed.

Recommendation

Ensure that c-ratios in the V3 core system are assigned such that it would not be possible for an LP to gain a net profit from:
1. Flagging a large position in the BFP market, realizing the collateral + not-yet-reduced losses gain + flag reward
2. Liquidating the BFP position, realizing the liquidation reward
3. Liquidating their now unhealthy V3 core position + any other accounts in the same vault that would be made unhealthy, realizing the liquidation reward

Resolution

Synthetix Team: Acknowledged.

H-08 | Large Orders Can Exceed minimumCredit

Category	Severity	Location	Status
Logical Error	● High	Position.sol	Resolved

Description

In the validateTrade function the minimumCredit is validated before accounting for the position's size in the market. Therefore positions which push the market over the minimumCredit, potentially by a large amount, are allowed to be settled.

Neither Trader's Nor LPs should be able to put the market in a state where the minimumCredit is exceeded, therefore the minimumCredit validation should be performed after taking the position's sizeDelta into account for the market.

Recommendation

Validate the minimumCredit after the market size has been updated in settleOrder. Consider leaving the current minimumCredit validation where it is now in order to offer some preemptive validation for order commitments.

Resolution

Synthetix Team: The issue was resolved in PR#2201.

H-09 | IM Validation Incorrectly Attributes Price Impact

Category	Severity	Location	Status
Logical Error	● High	Position.sol: 185	Resolved

Description

In the validateNextPositionEnoughMargin function the fillPremium is computed as if the fill price affected the entire position, when in fact the fillPrice only causes a net change in the position's margin for the newly added size from the current order.

For example:
• Current WETH price is $5,000
• Bob has a long position with size 10 WETH at an entry price of $5,000
• Bob opens an increase long for a size of 1 WETH and is negatively impacted to receive $5,100 as a fill price
• The computed fillPremium is 11 * -100 = -$1,100
• However Bob only received a negative impact of $100

As a result orders which build on top of existing positions will have their negative impact errantly accounted for the existing position size in the initial margin validation, preventing valid orders from being executed.

Additionally, orders which flip the side of the position will not be validated for the entire negative price impact that they experience, as only the net position on the opposite side remains.

Therefore positions can be opened where they are in fact below the IM since the entire negative price impact has not been accounted for.

Recommendation

Compute the fillPremium based upon the size delta of the order which is currently being executed instead of the entire size of the new position.

Resolution

Synthetix Team: The issue was resolved in PR#2200.

H-10 | DebtCorrection Uses Incorrect Price For Funding

Category	Severity	Location	Status
Logical Error	● High	PerpMarket.sol: 143	Resolved

Description

In the updateDebtCorrection function the pnl realized from funding is added to offset the funding gains which were realized to the account's margin.

However the funding gains which are transferred to the account's margin are based upon the oraclePrice, whereas the funding gains accounted for in the debt correction are based upon the new position's entry price.

These funding amounts will disagree due to the price differential and will cause a potentially large divergence between the actual debt of the market and the reported debt over time.

Recommendation

Use the oraclePrice to compute the settled funding amount in the debt correction.

Resolution

Synthetix Team: The issue was resolved in PR#2217.

M-01 | Reward Distributor Is Not Future-Proof

Category	Severity	Location	Status
Logical Error	● Medium	PerpRewardDistributor.sol	Acknowledged

Description

We have one reward distributor per collateral as specified with this:

`globalMarginConfig.supported[runtime.collateralAddress].rewardDistributor` and a `PerpRewardDistributor` instances can be linked to only one pool as we can see from their `_poolId` variable.

During liquidation, nonUsd collaterals are distributed only to that pool distributor is connected with. Although currently pool creation is done by governance only, this will create problems when the pool creation became permissionless.

]After creating pools became permissionless, when a pool owner choose to back a BFP-Market instance that they don't own, they won't be able to get something from liquidations (at least from `nonUsd` collaterals).

This will either disincentivize pool creators to back any other market that they don't own (which is against SNX's vision), or a pool owner that is not aware of this situation will back those markets and LP's will lose funds.

Recommendation

Change how rewards are distributed and allow multiple pool Id's. While distributing skim all pool's collaterals and distribute accordingly.

It is important to be extra cautious while implementing this because it can create new attack vectors. Pool addition should be done timely to capture reward distributions correctly.

Resolution

Synthetix Team: Acknowledged.

M-02 | Risk Free Trades Via Pay Debt

Category	Severity	Location	Status
Gaming	● Medium	MarginModule.sol	Resolved

Description PoC

An order can be committed that would attempt to merge with another account upon settlement but would then revert due to not meeting the Initial Margin Requirements.

That same order could then be "enabled" by paying down some debt in the account which would allow for the final merged account to meet IMR.

A trader could use this to create a risk-free trade whereby if the actual price moved enough from the Pyth price at the beginning of the settlement window, the trader could enable the trade and then immediately sell for a profit.

Recommendation

Consider restricting calls to `payDebt` when there is a pending order.

Resolution

Synthetix Team: The issue was resolved in PR#2216.

M-03 | Orders Can Be Censored By Front-running

Category	Severity	Location	Status
Gaming	● Medium	OrderModule.sol	Acknowledged

Description PoC

Any one can control whether or not their order is settled by front-running keeper bots. A user may commit an Order A that has a very tight price limit during block 0.

To prevent Order A from settling, in the same block 0, through a different account, they could also commit an Order B that would move the market slightly so that the slippage guard would prevent settlement of Order A.

In the next block 1 they could pay higher gas on an call to `settleOrder` for Order B which would "disable" Order A. The keeper would notice that Order A was outside of price tolerance and during the next block would call `cancelOrder`.

In anticipation of this, the user could, through yet another account, create an order to move the market back so that Order A could not be canceled. Back and forth they dance, until the user decides to let the order be canceled or settled.

This ability to censor trades at-will would give the user an opportunity to perform risk-free trades. Profit on a large risk-free trade could easily cover the gas for the multiple orders needed to manipulate the order's settleability.

Recommendation

Consider restricting calls to `settleOrder` to only the Keepers and apply a random execution order.

Resolution

Synthetix Team: Acknowledged.

M-04 | Liquidation Fees Not Covered By non-sUSD Collateral

Category	Severity	Location	Status
Gaming	● Medium	LiquidationModule.sol: 296	Acknowledged

Description

When a liquidation is flagged, the liquidator is paid out in sUSD regardless of the collateral type. `liquidateCollateral()` will decrement the amount of collateral and send the amount to the distributor. After, `withdrawMarketUsd()` is called, sending the liquidator sUSD. This will lower the credit capacity.

Credit capacity is primarily affected through withdrawals and deposits, which will essentially cancel out over time (although it is expected for there to be more deposits than withdrawals). However, the negative impact on credit capacity from liquidation rewards will not be balanced out in the long run.

This will cause accrual of negative credit capacity. Once the credit capacity decreases to a large enough state, `withdrawMarketCollateral()` will revert since it verifies that the credit capacity plus the deposited collateral must be greater than 0.

Recommendation

Pay out the liquidator in the collateral asset that has been liquidated. Make sure that reward is deducted from the distribution amount.

Resolution

Synthetix Team: Acknowledged.

M-05 | Negative New Margin Is Not Recognized

Category	Severity	Location	Status
Unexpected Behavior	● Medium	Position.sol	Acknowledged

Description PoC

When an order is impacted positively or negatively the price impact is applied to the rest of the existing position when computing the `newMarginUsd`. Because the `newMarginUsd` cannot go below 0, this behavior can potentially allow traders to erase losses from their position.

In the case of lower minimum margin configurations, the effect from the new entry price based on the fill price could create a situation which zeroes out sUSD collateral and then should create a new `debtUSD` amount but it does not because the `newMarginUSD` is not allowed to be negative in this case.

Recommendation

Consider reverting when the `marginUsdForNextMarginUsd` is negative or less than the `orderFee + keeperFee`.

Otherwise, to allow orders that would reach this edge case to be settled, consider refactoring the settlement logic such that the the PnL of the entire new position relative to the current market price is realized instead of just the existing position to the `fillPrice`.

Such a refactor would also require updates to the way the `debtCorrection` is handled as well.

Resolution

Synthetix Team: Acknowledged.

M-06 | Account Can Be Made Liquidatable By Cancelling

Category	Severity	Location	Status
Unexpected Behavior	● Medium	OrderModule.sol	Resolved

Description PoC

An account can be put into a liquidatable state when they are canceled. This would cause the immediate loss to a user of all funds that they could have instead withdrawn through `modifyColatteral`. This is made possible by the lack of a check for `isLiquidatable` at the end of the `cancelOrder` function.

As mentioned in finding M-03, an order's cancel-ability can be controlled by manipulating the skew. In addition to the fee for cancellation, the liquidation fees provide an added incentive to cancel the order which could even cover the costs of the skew manipulation.

Recommendation

This may be greatly mitigated by the recommendation in H-02 of adding logic to `validateTrade` which reverts if the new position, including fees, would cause the account to become liquidatable.

Resolution

Synthetix Team: Resolved.

M-07 | Debt With Zero Collateral Is Unliquidateable

Category	Severity	Location	Status
Unexpected Behavior	● Medium	LiquidationModule.sol	Acknowledged

Description

An account that has debt but no collateral and no position cannot be liquidated. This is due to a check within the `liquidateMarginOnly` function which reverts with `CannotLiquidateMargin` in the case there is no collateral.

This would effectively lock the loss in the system and never recognize it which would mean credit capacity would be overstated since the total `debtUSD` is added back to the final calculation.

The severity is reduced due to the difficulty since were were unable to find a way to put an account in this state with any significant debt value.

Currently the only identified means to achieve such a state is due to rounding error when splitting a position, such that a position can be left with 0 collateral and a few wei of debt. However, depending on how M-07 is resolved, this may become more likely and impactful in the future.

Recommendation

Consider removing the constraint that an account cannot have zero collateral when calling `liquidateMarginOnly`.

Resolution

Synthetix Team: Acknowledged.

M-08 | Missing Staleness Check For Chainlink Oracles

Category	Severity	Location	Status
Logical Error	● Medium	Global	Out of scope

Description

When the price is fetched in functions like `flagPosition`, the function will query Chainlink for the said price by calling the `latestRoundData` function. At this point, the `price` as well as the `updatedAt` are returned.

The issue, however, is that the freshness of the price is not confirmed, which can lead to stale prices being used to determine if a price is liquidatable or not. This can result in a position being wrongly liquidated using this function, and can lead to inaccurate profit/losses when closing a position.

Recommendation

Check and confirm that the price is not stale when fetching from Chainlink.

Resolution

Synthetix Team: Out of scope.

M-09 | Pending Fees Altered By setMarketConfiguration

Category	Severity	Location	Status
Logical Error	● Medium	PerpMarketConfiguration.sol	Acknowledged

Description

When the setMarketConfiguration function is called, the admin can change the utilizationBreakpointPercent, lowUtilizationSlopePercent, and highUtilizationSlopePercent.

Once these values are adjusted, all pending utilization fees will be updated according to the new slope, rather than the slope that was initially set when the fees were accruing.

This adjustment will result in a sudden increase in fees owed by users, potentially putting them in a liquidatable state due to the past utilization incorrectly increasing based on the new slope.

Recommendation

Consider updating the pending utilization fees in the setMarketConfiguration function.

Resolution

Synthetix Team: Acknowledged.

M-10 | DoS To Keeper Transactions

Category	Severity	Location	Status
DoS	● Medium	Global	Acknowledged

Description

Market's atomic actions that are not executed with a delay(`modifyCollateral` and `payDebt`) can be used to DOS keepers' liquidation calls.

Benign keeper who possibly simulated the transaction before executing it and saw that it is executable will see that their orders reverted because a user front-runned the keeper execution. Which in turn keeper will lose gas.

Attack Vector:
When someone sees a liquidation call for themselves (or flagging), they can front-run this call and either `modifyCollateral` or `payDebt` just enough that liquidation call will revert.

Although it’s not a problem system-wise considering the trader will be above liquidation threshold in the end, it is an effective way to DOS keepers' liquidation attempts. In the end, user can be stay at the liquidation threshold as long as they want and this can lead to fund loss for liquidators for their every liquidation attempt.

Recommendation

Document this so that keepers can be aware of it and they can use flash-bots to avoid such griefing vectors.

Resolution

Synthetix Team: Acknowledged.

M-11 | Value Extraction Via Trustless Keepers

Category	Severity	Location	Status
Gaming	● Medium	Global	Acknowledged

Description

Orders may be executed by any actor as soon as they are ready which allows for several potential value extraction or griefing mechanisms.

For example:
1. An attacker creates two accounts
2. The attacker watches the mempool and when they see that a trader will be positively impacted for an order that is going to be committed, the attacker creates two orders with these two accounts in order to sandwich the trader. All 3 orders are committed in the same block.
3. When all 3 orders are ready, the attacker can sandwich the trader with their two orders and settle all 3 of these orders by calling the `settleOrder` function.
4. The attacker gets positively impacted by their two orders via negatively impacting user who was expecting positive impact

Although the limit price of the order will limit how much can be stolen, users are not likely to set their `priceLimit` ahead of the current market price to lock in any positive impact.

Recommendation

The ultimate prevention for malicious keepers would be to use either a centralized, or semi-centralized but punishable keeper system.

However, if the current keeper setup is maintained, users should be informed that they must set the price limit such that they can capture any positive price impact that they expect to receive.

But this also comes with a caveat such that their order may not get filled if the skew changes naturally in the next 12 seconds before their order becomes executable.

Resolution

Synthetix Team: Acknowledged.

M-12 | Unexpected Liquidations Caused By Block.basefee

Category	Severity	Location	Status
Logical Error	● Medium	Position.sol: 427	Acknowledged

Description

To ensure that liquidators are adequately compensated for liquidating unhealthy positions, the liquidation fee is partially based on block.basefee. This differs from most liquidation functionality that does not base the fee on network activity.

The issue here is that users may get liquidated due to factors beyond their position's actual health. A seemingly healthy position could become unhealthy simply because of an increase in network activity. The block.basefee can increase by as much as 12.5% from one block to another.

This could potentially incentivize attackers to manipulate block.basefee, or it could occur naturally as the network experiences spikes in activity. All of this would be unexpected for users and could result in a loss of collateral for those affected.

Recommendation

Consider implementing a flat liquidation fee. Alternatively, document to users that network activity can impact their liquidation status.

Resolution

Synthetix Team: Acknowledged.

M-13 | payDebt Affects Utilization Without Update

Category	Severity	Location	Status
Logical Error	● Medium	MarginModule.sol: 531	Resolved

Description

In the payDebt function when the caller provides sUSD to repay an account's debt the creditCapacity of the market will be immediately incremented by the amountToBurn.

As a result the utilization of the market will decrease, as the delegatedCollateralValueUsd which relies on the creditCapacity will increase.

The utilization of the market will also be affected by the updateDebtAndCollateral function call for sUSD collateral repayments from the account.

However the utilization rate is not recomputed in the payDebt function so this utilization change is not reflected in the resulting utilization charged over any period where payDebt occurs.

Recommendation

Always recompute the utilization of the market after calling depositMarketUsd at the end of the payDebt function.

Resolution

Synthetix Team: The issue was resolved in PR#2196.

M-14 | BFP Market Liquidations May Liquidate LPs

Category	Severity	Location	Status
Unexpected Behavior	● Medium	Global	Acknowledged

Description

During the flagging of a position in the BFP market, the non-sUSD collateral of the flagged account is removed from the market deposited collaterals and credited to the LPs through a reward distributor.

However this action will remove the LPs collateral value from the totalDebt and re-allocate it to the distributor rewards which do not contribute to a V3 account's health.

Consider the following scenario:
- A BFP position holds 2 wETH, valued at $5,000 each
- The position has a $7,000 loss and is liquidatable
- Before the liquidation totalDebt = reportedDebt - marketDepositedCollateral = $3,000 - $10,000 = -$7,000
- After the liquidation totalDebt = 0 - 0 = 0, but the LPs have received $10,000 through the reward distributor

This reallocation of debt to the distributor may cause some v3 positions to unexpectedly become immediately unhealthy as they no longer are credited with the negative debt of the BFP position that was liquidated. This may cause loss of assets for those V3 accounts as they are liquidated even though they received collateral from the liquidation that would have made their position healthy.

Their debt and collateral would then be socialized amongst the other v3 positions, however these positions would not receive the BFP collateral rewards that would have gone to the liquidated account through the distributor this way.

Unexpected liquidations in V3 could be triggered by the wiping of a position's debt upon flagging as well. This is most likely to occur when the liquidated BFP position holds little or no collateral and is supported by position profit.

Recommendation

Consider restructuring the method by which market deposited collateral is distributed to LPs upon liquidation.

One potential approach would be to keep the position's equivalent loss amount of market deposited collateral locked, but not accounted for in the market deposited amounts, and to slowly stream this amount to the rewardDistributor over time to avoid instantaneous decreases in V3 position health.

Or keep this portion of the collateral locked until an arbitrary caller exchanges it with sUSD, for a fee reward. Otherwise be aware of this risk and clearly document it for V3 pool delegators.

Resolution

Synthetix Team: Acknowledged.

M-15 | Inadequate Estimated Liquidation Costs

Category	Severity	Location	Status
Logical Error	● Medium	Position.sol: 546	Acknowledged

Description

In the getLiquidationKeeperFee function the resulting estimated keeper fee is restricted to the maxKeeperFeeUsd. However when estimating the initial margin and maintenance margin values necessary for a position the estimated keeper fee may apply to multiple liquidatePosition transactions.

Therefore a single maxKeeperFeeUsd is insufficient to cover these multiple transactions. As a result, positions which would require multiple liquidatePosition calls are not required to hold the estimated keeper fee for each transaction.

It is unlikely that positions large enough to require multiple liquidatePosition calls would not be able to cover the keeper fees due to increased maintenance margin requirements. However out of an abundance of caution, the estimated keeper fees should allow multiple liquidate position calls to be accounted for.

Recommendation

Consider multiplying the globalConfig.maxKeeperFeeUsd by the iterations amount. This way liquidation keeper fee estimations for MM and IM will include the fees necessary for multiple high cost liquidatePosition calls.

```
return MathUtil.min(liquidationFeeInUsd * iterations, globalConfig.maxKeeperFeeUsd * iterations);
```

Resolution

Synthetix Team: Acknowledged.

M-16 | Attacker Profits From Block Stuffing Orders

Category	Severity	Location	Status
Gaming	● Medium	Global	Acknowledged

Description

Because orders only have about 5 blocks to have their order be settled before they become stale an attacker could congest the network so that attempts to call settleOrder fail due to all the gas for that block being used.

If an attacker were to stuff the blocks with their own transactions so that settleOrder will fail for 60 seconds, the attacker would then be able to cancel all of the pending orders and collect the cancelation reward for each one.

The lower the block.baseFee is and the more pending orders there are the more profitable this attack becomes.

Recommendation

Consider carefully balancing the profit made from cancels and document the risk for users.

Resolution

Synthetix Team: Acknowledged.

M-17 | Setting SkewScale Disrupts Markets

Category	Severity	Location	Status
Warning	● Medium	MarketConfigurationModule.sol	Resolved

Description

Setting the skewSkale with setMarketConfigurationById will cause unexpected results the next time recomputeFunding is called. The logic in recomputeFunding expects that the skew has not changed since the last time it was called which is respected in the current codebase.

However, it is also assumed that the skewScale has not been changed and therefore setting it in between will result in inaccurate funding numbers.

Recommendation

Call recomputeFunding at the beginning of setMarketConfigurationById.

Resolution

Synthetix Team: Resolved.

M-18 | Utilization Fees Are Not Accurate To Liquidity Updates

Category	Severity	Location	Status
Unexpected Behavior	● Medium	Global	Acknowledged

Description

The utilization fees in the BFP market are dependent on the amount of credit capacity delegated to the market by the V3 core system, however the utilization rate does not update when the amount of backing liquidity changes.

For instance, a delegator can undelegate from a vault which provides credit capacity to the BFP market and increase the utilization ratio. However this utilization update is not reflected in the utilization rate until a `recomputeUtilization` is triggered on the BFP side.

Therefore the utilization fees that are charged can be misrepresentative of the actual amount of liquidity utilized during these periods before a utilization recompilation is triggered.

A malicious LP could abuse this by minting sUSD directly before a utilization update in the BFP market, this way increasing the utilization ratio and forcing the the utilization fees for all traders to be higher over the next period. The LP may then backrun the utilization update and burn their sUSD that was minted.

Recommendation

Consider implementing callbacks for markets, whereby they can update crucial pieces of their state based upon credit capacity changes which occur through the V3 system.

Otherwise implement more restrictions to the `mintUsd` functionality such that LPs cannot extract more value from traders than they ought to.

Resolution

Synthetix Team: Acknowledged.

M-19 | Debt Is Not Reflected In CreditCapacity Until Repaid

Category	Severity	Location	Status
Logical Error	● Medium	MarginModule.sol: 530	Acknowledged

Description

In the payDebt function users may pay their existing debt with sUSD from outside of the BFP market system. This sUSD is deposited for the market with the depositMarketUsd function on the Synthetix V3 core system.

The depositMarketUsd function will increase the creditCapacity of the market by the repaid amount. This net increase in the market's creditCapacity counts towards the minimumCredit validation for the market, however it is not added until the trader repays their debt.

Regardless of when the trader pays their debt the LPs receive the value lost by the trader as a delta reduction in their debt. This debt reduction may not be fully utilizable by the LPs until the trader's debt is repaid as the minimumCredit validation would fail because the debt has not been added to the market's creditCapacity.

Trader's can therefore unjustly lock a portion of the LPs rightful gains until their debt is repaid. Notice that this will only occur when a market's creditCapacity is close to the minimumCredit and will not always have a noticeable locking effect on the LPs gains.

Recommendation

Consider offsetting this creditCapacity payDebt time mismatch by reducing the minimumCredit by the amount of outstanding debt that has yet to be paid. This way the LPs can mint using their realized debt reduction before trader's repay their debt.

Resolution

Synthetix Team: Acknowledged.

M-20 | Liquidation Prevented By Zero Vault Delegation

Category	Severity	Location	Status
DoS	● Medium	LiquidationModule.sol: 121	Resolved

Description PoC

When a PerpRewardDistributor is deployed it is assigned a list of poolCollateralTypes which it distributes rewards to. In the liquidateCollateral function, the BFP market distributes liquidated collateral to each poolCollateralType in the backing pool according to the value of collateral in each corresponding vault.

However if one of the vaults holds zero collateral then the distributeRewards call will attempt to distribute 0 rewards to 0 vault shares. This action reverts in the RewardDistribution.distribute function, where the totalSharesD18 value is validated to be nonzero.

As a result liquidation flagging or liquidation by margin only where the account holds the collateral token which would be distributed to the empty vault will be DoS'd.

Recommendation

If the vault collateral or the reward amount is equal to 0, skip distributing to that vault.

Resolution

Synthetix Team: The issue was resolved in PR#2317.

L-01 | mergeAccounts Used To Steal A User's Position

Category	Severity	Location	Status
Validation	● Low	PerpAccountModule.sol	Acknowledged

Description

mergeAccounts() validates that it is being called by a settlement hook, but does not verify the hook has access to the accounts it is trying to merge.

A malicious hook provider could either use an upgradable contract or a smart contract that has the same function selectors as a verified contract but performs different operations inside the function itself to get a malicious hook contract whitelisted.

Then, the malicious hook can merge an account with a large PnL to their own account to steal the profits from another user. Using MergeAccountSettlementHook as an example.

The malicious actor can set the first parameter for mergeAccounts() to the vaultAccountId and the second parameter to the accountId which initiates the hook.

They can then set vaultAccountId to the victim's account, and create an order that calls the hook. This will merge the victim's account to the malicious actor's account.

Recommendation

Be aware of the power granted to settlement hook contracts and carefully ensure that the owner of any hook contract cannot manipulate the from and to accounts such that trader positions are stolen.

Alternatively to remove this power from settlement hooks, revert to validating that the msg.sender holds the correct permissions from both accounts in the mergeAccounts function.

In conjunction with this, consider addressing the risk free trade vector previously described by reducing the valid order execution window or refactoring the pyth price selection as described in issues M-X and M-X.

Resolution

Synthetix Team: Acknowledged.

L-02 | Settlement Hook Executions Not Validated

Category	Severity	Location	Status
Validation	● Low	OrderModule.sol	Acknowledged

Description

In the report, it was noted that there is a potential vulnerability in the `commitOrder` function where no check is performed to determine if the settlement hook will be successful or not. This could result in accidental submission of a reverting hook, leading to wasted fees and gas.

Recommendation

To mitigate these risks, it is recommended to consider simulating the settlement hooks during the `commitOrder` process.

The call to the hook can be wrapped in a try/catch block to handle any potential errors and revert if necessary, providing valuable information about the call.

Resolution

Synthetix Team: Acknowledged.

L-03 | Redundant Funding Velocity Computations

Category	Severity	Location	Status
Optimization	● Low	Global	Resolved

Description

Throughout the BFP codebase, when the funding rate is recomputed the current funding velocity is computed and emitted with the `FundingRecomputed` event.

However the funding velocity has already been recomputed in the `getCurrentFundingRate` function, therefore this recomputation is potentially redundant and inefficient.

Recommendation

Consider returning the computed velocity from the `getCurrentFundingRate` function and as a result from the `getUnrecordedFundingWithRate` and `market.recomputeFunding` functions as well.

Resolution

Synthetix Team: The issue was resolved in commit e87976f.

L-04 | Unnecessary newSupportedCollaterals Array

Category	Severity	Location	Status
Optimization	● Low	MarginModule.sol: 427	Resolved

Description

In the `setMarginCollateralConfiguration` function the `newSupportedCollaterals` is declared to track the collateral addresses that are being set as the supported collaterals. However these addresses are already recorded in the same order in the `collateralAddresses` array.

Recommendation

Remove the `newSupportedCollaterals` array and assign the `globalMarginConfig.supportedCollaterals` to the `collateralAddresses` array.

Resolution

Synthetix Team: The issue was resolved in PR#2209.

L-05 | Lacking Margin Configuration Safety Validations

Category	Severity	Location	Status
Validation	● Low	MarginModule.sol: 379	Resolved

Description

In the setMarginCollateralConfiguration function there is no validation that the collateralAddresses array is filled with unique collateral addresses. In the event of an admin error, multiple entries for the same collateral may exist in the supportedCollaterals array.

This would double count the account and market value for these collaterals.

Recommendation

Consider adding validation to the setMarginCollateralConfiguration function to eliminate the possibility of a potentially catastrophic configuration error.

Resolution

Synthetix Team: The issue was resolved in PR#2209.

L-06 | Lacking Distributor Configuration Safety Validations

Category	Severity	Location	Status
Validation	● Low	PerpRewardDistributorFactoryModule.sol: 76	Resolved

Description

When creating a distributor with the `createRewardDistributor` function the `data.collateralTypes` is not validated to have unique entries.

In the event of an admin error, multiple entries for the same collateral would cause liquidation rewards to be distributed multiple times to the same vault delegators.

Recommendation

Ideally it is not possible to misconfigured the distributor upon creation. Consider implementing validation such that the `data.collateralTypes` list contains unique entries.

Resolution

Synthetix Team: The issue was resolved in PR#2209.

L-07 | getHealthData Divide By Zero Revert

Category	Severity	Location	Status
DoS	● Low	Position.sol: 615	Resolved

Description

The external getHealthFactor view function in the LiquidationModule relies on the does not gracefully handle the case where the queried position has zero size. As a result the getHealthFactor internal function in the Positions.sol file reverts as it attempts to divide by 0.

This is different from the behavior of the Synthetix V3 core system which reports a health of the max uint256 when an account's debt is negative. As a result this may cause unexpected reverts or difficult to handle edge cases for integrators who would interface with the getHealthFactor function.

Recommendation

Consider handling the position size zero case in the external getHealthFactor function in the LiquidationModule, or returning the max uint256 from the internal getHealthFactor function in the Positions.sol file when the maintenance margin is 0.

Resolution

Synthetix Team: The issue was resolved in PR#2209.

L-08 | Reported Debt Stepwise Jump Risks

Category	Severity	Location	Status
Gaming	● Low	Global	Acknowledged

Description

The BFP market integration with backing Synthetix V3 pools creates many opportunities for stepwise jumps in the debt reported to LPs. Positive stepwise jumps pose a risk of value extraction for LPs which would seek to delegate to a vault right before a `reportedDebt` decrease from the BFP market.

Negative stepwise jumps pose a risk of net profitable self liquidations in the Synthetix V3 system. As well as creating “scam” `totalDebt` wicks which can instantaneously cause Synthetix V3 positions to be liquidatable when a BFP liquidation is in an intermediate state.

An example of a “scam” wick V3 liquidation:
• Bob’s position is in significant profit, he removes all of his collateral since it is not needed to maintain mm.
• Bob commits orders which are unexecutable due to the priceLimit and cancels them.
• After creating a large account `debtUsd` from cancellations, Bob cancels an order which puts him below the mm limit.
• Bob flags his own BFP position, wiping his debt and thereby increasing the BFP `reportedDebt` in a stepwise fashion.
• Bob can now liquidate many V3 positions in the same transaction since he has caused this stepwise increase in the debt of V3 positions.

Additionally, stepwise jumps will occur upon order settlement when an order realizes an immediate gain or loss due to price impact, which may lead to similar gaming or unexpected behavior.

Recommendation

Async delegation may be sufficient to limit the risk of gaming. However the risk of causing LPs to be liquidated for a profit or even for griefing is still a concern, one such situation is described in H-02.

Ensure async delegation is implemented before connecting the BFP market to a Synthetix V3 pool, and consider smoothing out the `reportedDebt` jumps through a refactoring of the accounting for the liquidation process and price impact accounting.

Resolution

Synthetix Team: Acknowledged.

L-09 | sUSD Collateral maxAllowable Can Be Exceeded

Category	Severity	Location	Status
Unexpected Behavior	● Low	Margin.sol: 115	Acknowledged

Description

Each collateral including sUSD is configured with a maxAllowable amount which cannot be bypassed on deposits, however when realizing account profits this collateral cap is ignored when the sUSD account collateral is incremented.

As a result it is possible for the sUSD collateral amount for the market to exceed the maxAllowable when profit is realized.

Recommendation

Be aware of this unexpected behavior and consider if the maxAllowable should be exceeded.

Resolution

Synthetix Team: Acknowledged.

L-10 | Margin Requirements Do Not Use The Max Fee

Category	Severity	Location	Status
Best Practices	● Low	Position.sol: 500	Resolved

Description

In the getLiquidationMarginUsd function the marginAdjustment includes the estimated liquidation flag fee and liquidation fees. These fees are a best estimate of what the real fees may be in the event that the position were to be liquidated.

However as the Initial margin aims to provide a safety buffer it may be prudent to use the maximum that these liquidation costs can be in the estimate of the margin requirements. This way any potential under allocation of the margin requirements with respect to liquidation fees is avoided.

Recommendation

Consider using the maxKeeperFeeUsd in the marginAdjustment for the liquidation flag fee and the maxKeeperFeeUsd * iterations for the liqKeeperFee instead of the estimated amounts, which are based upon the current block base fee.

Resolution

Synthetix Team: The issue was resolved in PR#2321.

L-11 | Settlement Hooks May Be Repeated

Category	Severity	Location	Status
Validation	● Low	Global	Resolved

Description

When performing validation on the settlement hooks associated with an order, the settlement hook array is not validated to include only unique hook addresses. As a result a user may provide the same hook address multiple times in the same hooks array.

This may be leveraged to exploit settlement hook contracts, or even the BFP market itself depending on the logic in the settlement hook.

Recommendation

Validate that settlement hook addresses are not repeated in the hooks array in the `validateOrderHooks` function.

Resolution

Synthetix Team: The issue was resolved in PR#2253.

L-12 | Unused Flag

Category	Severity	Location	Status
Optimization	● Low	Flags.sol	Acknowledged

Description

In the Flags file there is an unused CREATE_ACCOUNT feature flag which is not used for any actions within the BFP market.

Recommendation

Consider removing the CREATE_ACCOUNT flag from the BFP market.

Resolution

Synthetix Team: Acknowledged.

L-13 | Unused Mm Parameter

Category	Severity	Location	Status
Optimization	● Low	Position.sol: 174	Resolved

Description

In the `validateNextPositionEnoughMargin` function the `mm` parameter value is no longer used.

Recommendation

Remove the `mm` parameter from the `validateNextPositionEnoughMargin` function.

Resolution

Synthetix Team: Resolved.

L-14 | Liquidation Rewards Are Not Bounded By The Minimum

Category	Severity	Location	Status
Unexpected Behavior	● Low	Global	Resolved

Description

The getMarginLiquidationOnlyReward, getLiquidationFlagReward, and getLiquidationKeeperFee functions do not apply the minKeeperFeeUsd to the resulting keeper reward like the getCancellationKeeperFee and getSettlementKeeperFee functions do.

As a result it may be possible for the liquidation rewards to fall below this minimum threshold depending on the configuration of the keeperProfitMarginUsd and liquidationRewardPercent.

Recommendation

Consider bounding the liquidation keeper fees such that they are at least the minKeeperFeeUsd.

Resolution

Synthetix Team: The issue was resolved in PR#2322.

L-15 | Settlement And Cancellation Fees Lack Fixed Margin

Category	Severity	Location	Status
Unexpected Behavior	● Low	Global	Resolved

Description

In the getCancellationKeeperFee and getSettlementKeeperFee functions the keeperProfitMarginUsd is not included in the consideration for the computed baseKeeperFeePlusProfitUsd.

This may be unexpected as the getLiquidationFlagReward, getLiquidationKeeperFee and getMarginLiquidationOnlyReward keeper fees include this static adjustment.

In cases where the baseKeeperFeeUsd is small relative to the priority fee necessary to get the transaction recorded this can result in latent settlements and executions as the keeperProfitMarginPercent would not contribute as much as the static keeperProfitMarginUsd would have.

Recommendation

Consider including the keeperProfitMarginUsd when considering what the resulting baseKeeperFeePlusProfitUsd ought to be, similar to the liquidation reward calculations.

Resolution

Synthetix Team: The issue was resolved in PR#2323.

L-16 | Typo

Category	Severity	Location	Status
Typo	● Low	MarginModule.sol: 395	Resolved

Description

In the `setMarginCollateralConfiguration` function the word maximum is misspelled as maxmium.

Recommendation

Correct maxmium to maximum.

Resolution

Synthetix Team: Resolved.

L-17 | Typo

Category	Severity	Location	Status
Typo	● Low	MarketConfigurationModule.sol: 51	Resolved

Description

In the `setMarketConfigurationById` function the comment on line 51 misspells `perp` as `per`.

Recommendation

Correct it to `perp`.

Resolution

Synthetix Team: Resolved.

L-18 | Immediately Cancelable Orders Are Permitted

Category	Severity	Location	Status
Validation	● Low	OrderModule.sol	Acknowledged

Description

Orders with a limitPrice outside of the current fill price can be entered and may be canceled in at the beginning of the settlement window.

It is already relatively simple to cancel an order with a tight limitPrice and users entering orders with a limitPrice that is currently invalid only exacerbates the situation.

Recommendation

Consider reverting if the limit price on the order exceeds the current price tolerance.

Resolution

Synthetix Team: Acknowledged.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits