GMX INTEGRATION AUDIT

12th May 2023

Table of Contents

Scope

The scope of the audit is https://github.com/struct-defi/struct-core/, with the commit hash 4904146a154ccf5aaa837b4ebbcac827e8884ad4.

FEYGMXFactory.sol
FEYGMXProduct.sol
GMXYieldSource.sol

Summary of Findings

In performing a security audit of Struct GMX Integration, several issues of concern were found. For each finding, a summary of the issue is documented, along with any other finer details regarding the issue. Security recommendations are also provided where applicable.

The table below shows a breakdown of security findings found categorized by severity or risk and impact. A finding that has been reported is listed as pending, and if that finding is satisfactorily mitigated, it will be categorized as resolved.

Severity	**Resolved**	**Unresolved**	**Total**
Critical	0	0	0
High	0	0	0
Medium	2	0	2
Low	2	1	3
Info	8	0	8

Issues

STR-001: Lack of value checks in setMaximumTrancheDuration and setMinimumTrancheDuration

Severity: Low
Status: Resolved
In the setMaximumTrancheDuration and setMinimumTrancheDuration functions, it is possible to set a new trancheDurationMin that is larger than the trancheDurationMax, or a new trancheDurationMax than the trancheDurationMin. Such behavior can result in the Factory contract not functioning properly due to _validateProductConfig always failing.

Recommendations
In setMaximumTrancheDuration, there should be a check that _trancheDurationMin < than trancheDurationMax.
In setMinimumTrancheDuration, there should be a check that _trancheDurationMax > trancheDurationMin.

Resolution
The recommended checks have been added.

STR-002: Management Fee and Performance fee can be changed by governance with no upper limit

Severity: Low
Status: Acknowledged
As managementFee and performanceFee can be changed by governance with no upper limit, those parameters should be passed in by the user during createProduct, and must match the values in the state variable.

This serves as a sanity check to ensure that the values on the frontend match the smart contract, as well as to prevent any frontrunning to increase the values of the fees by governance.

Recommendations
Add managementFee and performanceFee as part of the parameters when creating a product, and match those with the existing values in the Factory. Additionally, set an upper limit for management and performance fees.

Resolution
Acknowledged

STR-003: Invest is not callable when GMXYieldSource is set with isEmergencyLockActive as true

Severity: Medium
Status: Resolved
If GMXYieldSource is set with isEmergencyLockActive as true, invest will not be callable as it will revert during _depositToLP. In such a case, the state be changed to WITHDRAWN, to allow users to withdraw the underlying.

Recommendations
Verify what the expected behavior for such a situation should be. It was mentioned by the team that emergencyLock is not required for GMXYieldSource, so verify if that portion of the code can be removed.

Resolution
The emergencyLock feature was removed.

STR-004: Identical Senior and Junior token flag can be stored in storage

Severity: Info
Status: Resolved
Consider storing a field that shows if the SR and JR tokens are the same token. This flag can then be used instead of needing to compare each time. The flag can also be passed as a param when calling GMXYieldSource functions.

Recommendations
Add a bool field that stores if SR and JR tokens are the same.

Resolution
The recommendation has been implemented.

STR-005: Unused mappings in removeFundsFromLP

Severity: Info
Status: Resolved
In removeFundsFromLP, the TrancheConfig is loaded from storage, but never used.

```
    DataTypes.TrancheConfig storage _trancheConfigSr =
trancheConfig[DataTypes.Tranche.Senior];
    DataTypes.TrancheConfig storage _trancheConfigJr =
trancheConfig[DataTypes.Tranche.Junior];
```

Recommendations

Remove the unused code.

Resolution
The unused code has been removed.

STR-006: Unused arrays in initialize

Severity: Info
Status: Resolved
In initialize, there is no longer a need to set the swap paths for jr<>sr, only jr>native and sr>native, so the following state variables can be removed.

```
address[] internal seniorToJunior;
address[] internal juniorToSenior;
```

Recommendations
Remove the unused code.

Resolution
The unused code has been removed.

STR-007: Inconsistency between the comment and actual calculation

Severity: Info
Status: Resolved
There is an inconsistency between the comment and actual calculation in the code when calculating the _srFrFactor.

FEYGMXProduct: L627~

```
/// Calculate the amount of senior tokens the senior tranche investors expect at maturity
/// The simplified formula is: (tokensInvestable * (1 + fixedRate * trancheDuration / 1 year))
uint256 _srFrFactor = (trancheInfo[DataTypes.Tranche.Senior].tokensInvestable *
    Constants.DECIMAL_FACTOR *
    Constants.YEAR_IN_SECONDS +
    productConfig.fixedRate *
    _trancheDuration) / (Constants.DECIMAL_FACTOR * Constants.YEAR_IN_SECONDS);
```

Recommendations
Verify if the code or the comment's calculation are correct, and fix the inconsistency by updating the contradiction.

Resolution
The comment has been updated to reflect the same calculation as the code.

STR-008: Lack of non-zero check for shares issued

Severity: Info
Status: Resolved
In GMXYieldSource's _tokenToShares, there could be a possibility that shares issues are 0 even if the amount deposited into is non-zero, due to loss of precision during division.

Recommendations
Add a check that shares are not 0.

Resolution
A non-zero check has been added for the share amount to be issued.

STR-009: recompoundRewards can be called by anyone

Severity: Info
Status: Resolved
Currently, recompoundRewards is public and is called on supplyTokens and redeemTokens, and can be called by anyone externally. This could allow some sandwiching attacks within the same transaction if called by a malicious contract.

Recommendations
As it does not seem necessary for an external contract to call this function directly, you can put the current recompoundRewards as an internal function called by the contract itself, and then have an external function calling the internal function, and restrict that function to only be callable by EOA.

Resolution
Recompound is only callable internally during supplyTokens and redeemTokens, as well externally by the KEEPER role.

STR-010: isEmergencyLockActive cannot be set to true

Severity: Medium
Status: Resolved
_setEmergencyLock is internal and not able to be called to set isEmergencyLockActive to true.

Recommendations

Make _setEmergencyLock callable externally, like deactivateEmergencyLock, and onlyRole(GOVERNANCE).

Resolution
The emergencyLock feature has been removed as the GMXYieldSource does not require it.

STR-011: emergencyRedeemTokens is an unused function

Severity: Info
Status: Resolved
emergencyRedeemTokens is an unused function and can be removed.

Recommendations
Remove the unused code.

Resolution
The unused code has been removed.

STR-012: Unnecessary calculation

Severity: Info
Status: Resolved
In _getFeeBps, 10**30 can be made 1e30 instead to avoid the unnecessary arithmetic each time the function is called.

Recommendations
Use 1e30 instead of calculating it each time.

Resolution
The recommendation has been implemented.

STR-013: Phishing possible in depositFor

Severity: Low
Status: Resolved
In a GMXProduct's depositFor, the caller of the function can deposit funds which will be credited to the _onBehalfOf instead of msg.sender. In the case where a user has granted token allowance to a GMXProduct and accesses a phishing site, it would be possible for the phishing site to make the user call depositFor with _onBehalfOf as the phishing address.

Recommendations

Limit depositFor to be called by the FEYGMXFactory.

Resolution
The recommendation has been implemented.