

GMX · SECURITY REVIEW

GMX Risk Oracle.

A security review of Risk Oracle.

CLIENT	PROJECT	AUDITORS	AUDIT WINDOW	FINAL REPORT
GMX	Risk Oracle	Andreas Zottl, Nicholas Chew	May 18 – May 25, 2026	May 25, 2026

§ 00

Contents.

25 pages: overview, confidence ranking, scope, 13 findings, disclaimer, and about Guardian.

§ 01	Overview	P. 03
§ 02	Confidence Ranking	P. 04
§ 03	Scope & Classifications	P. 05
§ 04	Findings	P. 06
	Main Review · 8 findings	P. 07
	Remediation Review · 5 findings	P. 17
§ 05	Disclaimer	P. 24
§ 06	About Guardian	P. 25

§ 01

Overview.

Guardian reviewed Risk Oracle across 2 review rounds. The final report includes 2 Medium, 4 Low, 7 Informational findings.

AUDIT FIRM	REPOSITORY
Guardian	https://github.com/GuardianOrg/gmx-synthetics-private-team1-1779120112532
CLIENT	METHODOLOGY
GMX	Smart contract · Manual · Main Review · Remediation Review
PROJECT	MAIN REVIEW COMMIT
Risk Oracle	c49ebdea0001
PREPARED BY	REMEDATION REVIEW COMMIT
Andreas Zottl, Nicholas Chew	8c2d75803a8c
AUDIT WINDOW	
May 18 – May 25, 2026	
FINAL REPORT	
May 25, 2026	

1.1 Confidence Ranking

5 / 5

Very High Confidence

Guardian assigns a Confidence Ranking of 5 out of 5, combining finding severity, remediation outcome, and the reviewer’s assessment of latent risk. The full scale is on the next page.

1	2	3	4	5
VERY LOW CONFIDENCE	LOW CONFIDENCE	MODERATE CONFIDENCE	HIGH CONFIDENCE	VERY HIGH CONFIDENCE

1.2 Findings at a Glance

SEVERITY	TOTAL	PENDING	ACK	PARTIAL	RESOLVED
Critical	0	0	0	0	0
High	0	0	0	0	0
Medium	2	0	0	0	2
Low	4	0	0	0	4
Informational	7	0	2	0	5
All severities	13	0	2	0	11

§ 02

Confidence Ranking.

Guardian assigns a project-level confidence score using the scale below. The ranking combines finding severity, remediation outcome, and the reviewer's assessment of latent risk.

CONFIDENCE RANKING	DEFINITION AND RECOMMENDATION	RISK PROFILE
5 Very High Confidence ASSIGNED	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. Recommendation: code is highly secure at time of audit.	0 High/Critical findings and few Low/Medium severity findings.
4 High Confidence	Code is clean, well-structured, and adheres to best practices. Design patterns are sound and test coverage is strong. Recommendation: suitable for deployment after remediations; consider periodic review with changes.	0-1 High/Critical findings per engagement week and little to no Medium severity findings.
3 Moderate Confidence	Code is functional but shows gaps in defensive programming or test coverage. Recommendation: remediate the findings and re-review the changes before deployment.	Some High/Critical findings, with various Medium severity findings.
2 Low Confidence	Recurring issues suggest systemic gaps rather than isolated mistakes. Recommendation: remediate, then commission a follow-up review at a frozen commit.	Over 6 High and Critical findings across the engagement.
1 Very Low Confidence	The volume and severity of findings indicate the codebase is not ready for production. Recommendation: substantial rework, then a full re-review.	Over 12 High and Critical findings across the engagement.

§ 03

Scope & Classifications.

Guardian reviewed Risk Oracle in <https://github.com/GuardianOrg/gmx-synthetics-private-team1-1779120112532>.

3.1 Severity Matrix

	IMP · HIGH	IMP · MED	IMP · LOW
LIK · HIGH	Critical	High	Medium
LIK · MED	High	Medium	Low
LIK · LOW	Medium	Low	Low

IMPACT · HIGH

Significant loss of assets, harm to a group of users, or core protocol functionality is disrupted.

IMPACT · MEDIUM

Small amount of funds can be lost, or ancillary functionality is affected. Users may experience reduced or delayed receipt.

IMPACT · LOW

Unexpected behavior in some functionality that is notable but does not meet the criteria for higher severity.

LIKELIHOOD · HIGH

Possible with reasonable assumptions that mimic on-chain conditions; cost is low compared to amount gained.

LIKELIHOOD · MEDIUM

Possible only in uncommon cases or requires a large amount of capital relative to disruption caused.

LIKELIHOOD · LOW

Unlikely to ever occur in production.

§ 04

Findings.

Grouped by round. Visible IDs reset by severity within each round, so a finding is named the same way in the report as it was in the channel it was written in. Status is where each finding ended up; its own page says how.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
MAIN REVIEW					
M-01	Funding/Borrowing Update Influence Stale Markets	Configuration	MEDIUM	RESOLVED	p. 7
M-02	RiskOracleConfig is not granted CONTROLLER role	Configuration	MEDIUM	RESOLVED	p. 9
L-01	Missing Cross-Parameter Sanity Checks	Configuration	LOW	RESOLVED	p. 11
L-02	Risk Oracle Config Accepts Non-Market Addresses	Configuration	LOW	RESOLVED	p. 12
L-03	Unnecessary Setters Exposed	Warning	LOW	RESOLVED	p. 13
L-04	RISK_ORACLE Role Missing In Scripts	Warning	LOW	RESOLVED	p. 14
I-01	Unused Imports	Superfluous Code	INFORMATIONAL	RESOLVED	p. 15
I-02	_validateKey performs a redundant role check	Access Control	INFORMATIONAL	RESOLVED	p. 16
REMEDIATION REVIEW					
I-01	Coarser operational controls than ConfigSyncer	Best Practices	INFORMATIONAL	ACKNOWLEDGED	p. 17
I-02	Allowlist differs from previous ConfigSyncer	Warning	INFORMATIONAL	ACKNOWLEDGED	p. 19
I-03	Incorrect NatSpec	Documentation	INFORMATIONAL	RESOLVED	p. 21
I-04	Unused EventUtils	Superfluous Code	INFORMATIONAL	RESOLVED	p. 22
I-05	Stale updateMarketConfigUtils Script	Configuration	INFORMATIONAL	RESOLVED	p. 23

MEDIUM **M-01**

RESOLVED

Funding/Borrowing Update Influence Stale Markets

CATEGORY Configuration

LOCATION <https://github.com/GuardianOrg/gmx-synthetics-private-team1-1779120112532/blob/c49ebdea0001126908ad266dec1b870c254f2c6e/contracts/config/RiskOracleConfig.sol>

REVIEW Main Review

DESCRIPTION

`RiskOracleConfig` allows the `RISK_ORACLE` role to directly update funding and borrowing rate parameters without first settling the market's accrued funding and borrowing state. Since the next market update calculates elapsed time from the previous update timestamp using the current config, a newly set rate can be applied to the entire unsettled interval.

This can overcharge or undercharge open positions and distort claimable funding or borrowing fees and therefore lead to step-wise jumps in positions health. This can even lead to positions being suddenly liquidatable. The more stale a market is the bigger the impact of such config changes.

RECOMMENDATION

Consider to update the market state (accrue funding, borrowing, etc.) before changing configurations that have any influence on a stale market state.

RESOLUTION

Status: Resolved.

Guardian: The remediation for stale borrowing accrual only settles borrowing before updates to

`BASE_BORROWING_FACTOR` and `ABOVE_OPTIMAL_USAGE_BORROWING_FACTOR`.

However, `RiskOracleConfig` also allows the `RISK_ORACLE` role to update `OPTIMAL_USAGE_FACTOR`, which is part of the kink borrowing-rate formula. When it is non-zero, `MarketUtils.getBorrowingFactorPerSecond` uses it to route into `getKinkBorrowingFactor`:

```
```plain text
```

```
uint256 optimalUsageFactor = getOptimalUsageFactor(dataStore, market.marketToken, isLong);
```

```
if (optimalUsageFactor != 0) {
 return getKinkBorrowingFactor(
 dataStore,
 market,
```

**M-01** Funding/Borrowing Update Influence Stale Markets

CONTINUED

```
isLong,
reservedUsd,
poolUsd,
optimalUsageFactor
);
}
`
```

Inside `getKinkBorrowingFactor`, the value controls the kink threshold and the divisor used to calculate the above-optimal portion of the borrowing rate. Despite this,

`_isRiskOracleBorrowingBaseKey` excludes `OPTIMAL_USAGE_FACTOR`.

As a result, `_settleBorrowingIfRequired` returns without settling accrued borrowing before the new `OPTIMAL_USAGE_FACTOR` is stored. The next borrowing update can then apply the new kink threshold to the full elapsed stale interval since `cumulativeBorrowingFactorUpdatedAt`.

This can overcharge or undercharge open positions for the stale interval and may affect position health and liquidation eligibility. The impact depends on market staleness, utilization, and the size of the optimal usage factor change.

**\*\*Recommendation\*\***

Include `OPTIMAL_USAGE_FACTOR` in `_isRiskOracleBorrowingBaseKey`.

MEDIUM **M-02**

RESOLVED

# RiskOracleConfig is not granted CONTROLLER role

CATEGORY Configuration  
LOCATION RiskOracleConfig.sol, deployRiskOracleConfig.ts  
REVIEW Main Review

## DESCRIPTION

RiskOracleConfig requires the CONTROLLER role to function because it writes through DataStore and emits through EventEmitter, both of which are onlyController-gated.

The deploy script deployRiskOracleConfig.ts attempts to grant CONTROLLER after deployment:

```
``typescript
afterDeploy: async ({ deployedContract }) => {
 await grantRoleIfNotGranted(deployedContract, "CONTROLLER");
},
`;
```

However, grantRoleIfNotGranted immediately returns on existing mainnet deployment networks:

```
``typescript
if (hre.gmx.isExistingMainnetDeployment) {
 return;
}
`;
```

Affected flows include:

- setRiskOracleMarketEnabled, because it calls DataStore.setBool.
- setBool, setAddress, setBytes32, setUint, and setInt, because they call the corresponding DataStore.set\* method.
- All successful setter flows, because they emit through EventEmitter, which also requires CONTROLLER.

## RECOMMENDATION

Add RiskOracleConfig to requiredRolesForContracts.CONTROLLER in config/roles.ts, and ensure the mainnet role update transaction grants the deployed RiskOracleConfig address the CONTROLLER role.

---

**M-02** RiskOracleConfig is not granted CONTROLLER roleCONTINUED

---

---

**— RESOLUTION**

Status: Resolved.

LOW **L-01**

RESOLVED

# Missing Cross-Parameter Sanity Checks

CATEGORY Configuration

LOCATION <https://github.com/GuardianOrg/gmx-synthetics-private-team1-1779120112532/blob/c49ebdea0001126908ad266dec1b870c254f2c6e/contracts/config/RiskOracleConfig.sol>

REVIEW Main Review

## DESCRIPTION

`RiskOracleConfig` allows the `RISK_ORACLE` role to directly update sensitive market parameters. Right now, values are mostly checked against broad per-key bounds, but important cross-parameter invariants are not enforced.

For example, `MIN_COLLATERAL_FACTOR` should remain greater than or equal to `MIN_COLLATERAL_FACTOR_FOR_LIQUIDATION`; otherwise users may be able to open or increase positions that are immediately liquidatable.

## RECOMMENDATION

Consider to add invariant checks before applying risk oracle updates.

## RESOLUTION

Status: Resolved.

LOW **L-02**

RESOLVED

# Risk Oracle Config Accepts Non-Market Addresses

CATEGORY Configuration

LOCATION <https://github.com/GuardianOrg/gmx-synthetics-private-team1-1779120112532/blob/c49ebdea0001126908ad266dec1b870c254f2c6e/contracts/config/RiskOracleConfig.sol#L57-L73>

REVIEW Main Review

## DESCRIPTION

`RiskOracleConfig.setRiskOracleMarketEnabled` only checks that the given `market != address(0)` before setting `RISK_ORACLE_MARKET_ENABLED`. It does not verify that the supplied address is a registered GMX market.

## RECOMMENDATION

Consider to validate that the given market exists in GMX.

## RESOLUTION

Status: Resolved.

LOW **L-03**

RESOLVED

# Unnecessary Setters Exposed

CATEGORY	Warning
LOCATION	RiskOracleConfig.sol
REVIEW	Main Review

## DESCRIPTION

`RiskOracleConfig` should only expose `setUint` for the currently allowlisted risk-oracle parameters, because those parameters are consumed by the protocol as `uint256` configs.

Instead, the contract exposes generic setters for multiple `DataStore` types:

- `setBool`
- `setAddress`
- `setBytes32`
- `setUint`
- `setInt`

Because the same base-key allowlist is shared by all setters, a `RISK_ORACLE` account can accidentally call the wrong setter for a critical risk parameter. The transaction can succeed and emit a config event, but the actual uint config used by the protocol remains unchanged.

## RECOMMENDATION

Remove the non-uint setters from `RiskOracleConfig` if the internal risk oracle only needs to update `uint` risk parameters.

## RESOLUTION

Status: Resolved.

LOW **L-04**

RESOLVED

## RISK\_ORACLE Role Missing In Scripts

CATEGORY Warning  
LOCATION GLOBAL  
REVIEW Main Review

### — DESCRIPTION

The `RISK_ORACLE` is missing in multiple scripts:

- `validateRolesUtils` only uses the `Role` library and not the new `Role2` library
- `printRoles` lacks the role in `knownRoles`
- `validateContractDeployment` lacks the role in `roleLabels`
- `validateDeploymentUtils` does not include it in the external-role allowlist

### — RECOMMENDATION

Consider to add the `RISK_ORACLE` to scripts wherever it is needed.

### — RESOLUTION

Status: Resolved.

INFORMATIONAL I-01

RESOLVED

# Unused Imports

CATEGORY	Superfluous Code
LOCATION	<a href="https://github.com/GuardianOrg/gmx-synthetics-private-team1-1779120112532/blob/c49ebdea0001126908ad266dec1b870c254f2c6e/contracts/config/RiskOracleConfig.sol">https://github.com/GuardianOrg/gmx-synthetics-private-team1-1779120112532/blob/c49ebdea0001126908ad266dec1b870c254f2c6e/contracts/config/RiskOracleConfig.sol</a>
REVIEW	Main Review

## DESCRIPTION

`RiskOracleConfig.sol` imports `Keys2`, `Precision`, `Cast`, and `MarketUtils`, but none of these are used.

## RECOMMENDATION

Consider to remove unused code.

## RESOLUTION

Status: Resolved.

## INFORMATIONAL I-02

RESOLVED

# validateKey performs a redundant role check

CATEGORY	Access Control
LOCATION	RiskOracleConfig.sol
REVIEW	Main Review

## DESCRIPTION

All public risk-oracle setters are already protected by the `onlyRiskOracle` modifier. However, `_validateKey` checks the same role again:

```
```plain text
if (roleStore.hasRole(msg.sender, Role2.RISK_ORACLE)) {
...
return;
}
```

```
revert Errors.InvalidBaseKey(baseKey);
``
```

This role check is redundant for the current setters. Additionally, if `_validateKey` is reused by a future function that is not protected by `onlyRiskOracle`, callers without the role would receive `InvalidBaseKey` instead of the more accurate `Unauthorized` error.

RECOMMENDATION

Remove the role check from `_validateKey` and rely on the external `onlyRiskOracle` modifier for authorization.

RESOLUTION

Status: Resolved.

INFORMATIONAL I-01

ACKNOWLEDGED

Coarser operational controls than ConfigSyncer

CATEGORY	Best Practices
LOCATION	RiskOracleConfig.sol
REVIEW	Remediation Review

— DESCRIPTION

The older Chaos Labs risk-oracle flow used `ConfigSyncer` as an adapter between the external Risk Oracle and GMX config writes. Before applying an update, `ConfigSyncer` checked multiple operational disable flags:

- global sync feature disable;
- per-market disable;
- per-parameter disable;
- per-market-parameter disable.

`RiskOracleConfig` uses a simpler direct-write model. A `CONFIG_KEEPER` can enable or disable all risk-oracle updates for a market through `setRiskOracleMarketEnabled`, and a `RISK_ORACLE` account can then write allowlisted parameters for enabled markets.

This is simpler and may be intentional, but it is also less granular. If the internal risk oracle has an issue with one parameter family, such as funding, borrowing, or caps, operators can only disable the whole market or revoke the `RISK_ORACLE` role. They cannot pause a single base key or a single market/base-key pair while allowing other risk-oracle updates to continue.

— RECOMMENDATION

Consider whether `RiskOracleConfig` should include finer emergency controls similar to `ConfigSyncer`, especially:

- a global risk-oracle config feature disable;
- a per-base-key disable;
- optionally, a per-market/base-key disable.

If this coarser control model is intentional, document the operational runbook for pausing faulty risk-oracle updates, including whether operators should disable affected markets or revoke the `RISK_ORACLE` role.

I-01 Coarser operational controls than ConfigSyncerCONTINUED

— RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-02

ACKNOWLEDGED

Allowlist differs from previous ConfigSyncer

CATEGORY	Warning
LOCATION	RiskOracleConfig.sol
REVIEW	Remediation Review

DESCRIPTION

The new `RiskOracleConfig` allowlist is not a direct replacement for the previous Chaos Labs `ConfigSyncer` risk-oracle surface.

`ConfigSyncer` can apply updates for the following base keys that `RiskOracleConfig` does not allow:

- `SWAP_IMPACT_FACTOR`
- `SWAP_IMPACT_EXPONENT_FACTOR`
- `BORROWING_FACTOR`
- `BORROWING_EXPONENT_FACTOR`
- `RESERVE_FACTOR`
- `OPEN_INTEREST_RESERVE_FACTOR`

Conversely, `RiskOracleConfig` adds support for some keys that `ConfigSyncer` did not support, including:

- `MAX_POSITION_IMPACT_FACTOR`
- `GLV_MAX_MARKET_TOKEN_BALANCE_USD`
- `GLV_MAX_MARKET_TOKEN_BALANCE_AMOUNT`
- `FUNDING_FACTOR`
- `FUNDING_EXPONENT_FACTOR`
- `THRESHOLD_FOR_STABLE_FUNDING`
- `THRESHOLD_FOR_DECREASE_FUNDING`
- `MIN_COLLATERAL_FACTOR`
- `MIN_COLLATERAL_FACTOR_FOR_OPEN_INTEREST_MULTIPLIER`
- `MIN_COLLATERAL_FACTOR_FOR LIQUIDATION`

This may be intentional if the internal Risk Oracle is meant to manage a different parameter set than the Chaos Labs integration. If not, the migration could silently remove automated risk-oracle management for swap impact, legacy borrowing, and reserve-factor parameters.

RECOMMENDATION

Confirm whether the new allowlist intentionally drops the `ConfigSyncer`-only keys.

I-02 Allowlist differs from previous ConfigSyncerCONTINUED

— RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-03

RESOLVED

Incorrect NatSpec

CATEGORY Documentation

LOCATION <https://github.com/GuardianOrg/gmx-synthetics-private-team1-1779120112532/blob/8c2d75803a8c843ed2a72d8a3503981a70609527/contracts/config/RiskOracleConfig.sol#L214-L228>

REVIEW Remediation Review

DESCRIPTION

The NatSpec of the `_validateKeyAndMarket` function states out that the function returns an address but it returns a market now.

RECOMMENDATION

Consider to update the NatSpec.

RESOLUTION

Status: Resolved.

INFORMATIONAL I-04

RESOLVED

Unused EventUtils

CATEGORY	Superfluous Code
LOCATION	https://github.com/GuardianOrg/gmx-synthetics-private-team1-1779120112532/blob/8c2d75803a8c843ed2a72d8a3503981a70609527/contracts/config/RiskOracleConfig.sol#L25
REVIEW	Remediation Review

DESCRIPTION

`EventUtils.IntItems` is unused now.

RECOMMENDATION

Consider to remove unused code.

RESOLUTION

Status: Resolved.

INFORMATIONAL I-05

RESOLVED

Stale updateMarketConfigUtils Script

CATEGORY Configuration

LOCATION <https://github.com/GuardianOrg/gmx-synthetics-private-team1-1779120112532/blob/8c2d75803a8c843ed2a72d8a3503981a70609527/scripts/updateMarketConfigUtils.ts#L8-L12>

REVIEW Remediation Review

DESCRIPTION

The list of RISK_ORACLE managed keys in the `updateMarketConfigUtils.ts` script is stale.

RECOMMENDATION

Consider to update this script.

RESOLUTION

Status: Resolved.

§ 05

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.

§ 06

About Guardian.

Guardian is a smart contract security firm. Every engagement is worked by two independent teams competing on the same scope, each with its own tooling and methods, so no single reviewer's blind spot survives.

\$45B+DIGITAL ASSETS UNDER
REVIEW**200+**

ENGAGEMENTS COMPLETED

500+CRITICAL & HIGH FINDINGS
REPORTED

Reach us at audits@guardianaudits.com.