PUBLIC

Code Assessment

of the Chief Migration Smart Contracts

April 30, 2025

Produced for

Sky

by

CHAINSECURITY

Contents

1 Executive Summary 3
2 Assessment Overview 5
3 Limitations and use of report 8
4 Terminology 9
5 Open Findings 10
6 Notes 11

1 Executive Summary

Dear all,

Thank you for trusting us to help Sky with this security audit. Our executive summary provides an overview of subjects covered in our audit of the latest reviewed contracts of Chief Migration according to Scope to support you in forming an opinion on their security risks.

Sky offers deployment and initialization libraries to migrate the existing `MCD_ADM` (the legacy Chief with MKR as governance token) to a new Chief that uses SKY as governance token.

The most critical subjects covered in our audit are functional correctness, frontrunning resistance, and integration with other contracts of the system.

Security regarding all the aforementioned subjects is high. Note that the Chief migration requires strong coordination and adherence to timelines to minimize the risk of governance attacks at different migration phases, see Migration Considerations for more details.

In summary, we find that the codebase provides a high level of security.

It is important to note that security audits are time-boxed and cannot uncover all vulnerabilities. They complement but don't replace other vital measures to secure a project.

The following sections will give an overview of the system, our methodology, the issues uncovered, and how they have been addressed. We are happy to receive questions and feedback to improve our service.

Sincerely yours,

ChainSecurity

1.1 Overview of the Findings

Below we provide a brief numerical overview of the findings and how they have been addressed.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	0

2 Assessment Overview

In this section, we briefly describe the overall structure and scope of the engagement, including the code commit which is referenced throughout this report.

2.1 Scope

The assessment was performed on the source code files inside the Chief Migration repository based on the documentation files. The table below indicates the code versions relevant to this report and when they were received.

V	Date	Commit Hash	Note
1	12 Apr 2025	722ff567592f6154cd4216033813db0ac401f610	Initial Version
2	24 Apr 2025	e4a820483694f015a2daf8b1dccc5548036d94d4	Updated Commit

For the solidity smart contracts, the compiler version `0.8.21` was chosen. The `evm_version` is set to `shanghai`.

The following files are in scope of this review:

```
deploy/
    MigrationDeploy.sol
    MigrationInit.sol
    MigrationInstance.sol
```

2.1.1 Excluded from scope

All files not listed above including the tests are out of scope. The to-be-deployed contracts (namely MkrSky, Chief, VoteDelegateFactory, LockStakeEngine, and StakingRewards) were covered in other reviews. The parameter selection is out of scope.

This review considers using the relevant Chainlog entries in the deployment and initialization libraries as of Chainlog V1.19.9. Changes to these entires prior to the migration are not in scope and may lead to wrong deployment / initialization or reverts.

In addition, in the initialization library, the bad debt auction (`flop`) and the Emergency Shutdown Module (`esm`) are disabled. **The economic implications of turning off these two mechanisms are out of scope.**

2.2 System Overview

This system overview describes the initially received version (Version 1) of the contracts as defined in the Assessment Overview.

Furthermore, in the findings section, we have added a version icon to each of the findings to increase the readability of the report.

Sky offers deployment and initialization libraries to migrate the existing `MCD_ADM` (the legacy Chief with MKR as governance token) to a new Chief that uses SKY as governance token.

2.2.1 Migration

The migration libraries aim to migrate governance from MKR to SKY. Note that for the governance system that includes the following migration and configuration changes:

- Chief: The new Chief is deployed with SKY as the governance token. All contracts with dependencies to DsChief will migrate their authority.
- VoteDelegateFactory v3: The new VoteDelegateFactory v3 is deployed to enable voting power delegations on the new Chief contract. Note that the prior polling contract can be reused.
- MkrSky: The new unidirectional MkrSky contract is deployed and used in all new contracts. The bidirectional MkrSky converter will eventually be deprecated but is configured to become unidirectional as well. Ultimately, only MKR to SKY conversions are possible.
- SkyOsm: An OSM contract for SKY, required for Lockstake v2 (see below) where SKY serves as the collateral, is deployed.
- Lockstake v2: The SKY-based Lockstake v2 module (LockstakeSky, LockstakeEngine v2, LockstakeClipper v2, ClipperCalc) will be deployed accordingly with the contracts above. Additionally, the LockstakeMigrator will be prepared to allow for migrating v1 positions to v2.
- StakingRewards: A farm for Lockstake v2's LockstakeSky will be configured. The reward token will be USDS and the staking token will be LockstakeSky accordingly.

`MigrationDeploy.deployMigration()` deploys the new Chief contracts in the order above. Note that the underlying oracle of the SkyOsm will be an input oracle that should correspond to a Scribe oracle. The ClipperCal will be deployed with the factory `CALC_FAB`. Last, the

`MigrationInit.initMigration`, expected to be part of a governance spell and thus executed in the context of `MCD_PAUSE_PROXY`, initializes the deployed contracts accordingly. More specifically, it performs the following operations:

1. Various sanity checks are performed.
2. The migration from DsChief to Chief is performed. The contracts with DsChief as the authority are updated so that the new Chief becomes the authority. This mainly includes the `MCD_PAUSE` and the mom contracts. Namely, that includes `SPLITTER_MOM`, `OSM_MOM`, `CLIPPER_MOM`, `DIRECT_MOM`, `STARKNET_ESCROW_MOM`, `LINE_MOM`, `LITE_PSM_MOM`, and `SPBEAM_MOM`. Please consult Authority in Contracts of SubDAOs for further considerations. Further, `MCD_ADM` is updated to be the new Chief.
3. The `VOTE_DELEGATE_FACTORY` and `VOTE_DELEGATE_FACTORY_LEGACY` entries are updated in chainlog to v3 and v2, respectively. Note that this removes the v1 factory from the chainlog.
4. The new MkrSky converter is initialized by pausing the SKY to MKR flow on the old converter, pre-mint SKY tokens to the new converter, and updating the chainlog. Note that `SkyInit.updateMkrSky()` is used. See the respective audit report for more details.
5. The oracles are configured. Namely, the `pip` of flapper `MCD_FLAP` of `MCD_SPLIT` (flapper in Vow) is updated to be the source oracle of the SkyOsm. The underlying oracle is published to `FLAP_SKY_ORACLE` in the chainlog to replace the previous value. Additionally, the SkyOsm is published to the chainlog with the key `PIP_SKY`.
6. The StakingRewards contract for staking LockstakeSky with USDS rewards is wired with `MCD_SPLIT` to enable receiving rewards. Additionally, the reward duration is set. Note that the LockstakeSky farm with USDS rewards is registered as `REWARDS_LSSKY_USDS` within the chainlog while the `REWARDS_LSMKR_USDS` entry is migrated to `REWARDS_LSMKR_USDS_LEGACY`.
7. The LockstakeEngine is initialized with its peripheral contracts using `LockstakeInit.initLockstake()`. Note that the oracle set for the ilk will be the SkyOsm (`PIP_SKY`). Please consult the respective audit report for more details.

In addition to the Chief migration:

- The MKR flops are turned off by setting the sump to `type(uint256).max`, namely bad debt auction will be disabled.
- The ESM is disabled by setting `min` to `type(uint256).max`. Consequently, emergency shutdown can no longer be triggered by burning MKR tokens.

2.2.2 Changelog

In Version 2, the authority of the newly deployed `SPBEAM_MOM` contract is also switched to the new Chief in `MigrationInit` library.

2.3 Roles and Trust Model

Deployers are supposed to deploy the contracts as specified by the reviewed script. Deployers are, however, EOAs that could perform undesired actions besides simple deployment, such as changing the settings of the system or granting themselves special privileges. It is important that after deployment, concerned parties thoroughly check the state of the deployed contracts to ensure that no unexpected action has been taken on them during deployment.

The initialization, expected to be plotted and cast with a governance spell in the context of `MCD_PAUSE_PROXY`, assumes an honest governance that properly inspects the parameters to be passed and takes swift actions in case of governance attacks during different phases of the Chief migration.

3 Limitations and use of report

Security assessments cannot uncover all existing vulnerabilities; even an assessment in which no vulnerabilities are found is not a guarantee of a secure system. However, code assessments enable the discovery of vulnerabilities that were overlooked during development and areas where additional security measures are necessary. In most cases, applications are either fully protected against a certain type of attack, or they are completely unprotected against it. Some of the issues may affect the entire application, while some lack protection only in certain areas. This is why we carry out a source code assessment aimed at determining all locations that need to be fixed. Within the customer-determined time frame, ChainSecurity has performed an assessment in order to discover as many vulnerabilities as possible.

The focus of our assessment was limited to the code parts defined in the engagement letter. We assessed whether the project follows the provided specifications. These assessments are based on the provided threat model and trust assumptions. We draw attention to the fact that due to inherent limitations in any software development process and software product, an inherent risk exists that even major failures or malfunctions can remain undetected. Further uncertainties exist in any software product or application used during the development, which itself cannot be free from any error or failures. These preconditions can have an impact on the system's code and/or functions and/or operation. We did not assess the underlying third-party infrastructure which adds further inherent risks as we rely on the correct execution of the included third-party technology stack itself. Report readers should also take into account that over the life cycle of any software, changes to the product itself or to the environment in which it is operated can have an impact leading to operational behaviors other than those initially determined in the business specification.

4 Terminology

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

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

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

Likelihood	Impact		
	High	Medium	Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

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

5 Open Findings

In this section, we describe our findings. The findings are split into these different categories:

Below we provide a numerical overview of the identified findings, split up by their severity.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	0

6 Notes

We leverage this section to highlight further findings that are not necessarily issues. The mentioned topics serve to clarify or support the report, but do not require an immediate modification inside the project. Instead, they should raise awareness in order to improve the overall understanding.

6.1 Authority in Contracts of SubDAOs

Note Version 1

Contracts of SubDAOs that are dependent on the Chief as its authority will not have their authority updated in `initMigration()`. Note that the authority of those should be changed in a sub-spell. At the time of writing, the only relevant contract is Spark's Freezer Mom. As suggested in the comments, the authority of Freezer Mom will be changed in a Spark sub-spell.

6.2 Chainlog Considerations

Note Version 1

Governance and users should be aware that some items will be fully removed from the Chainlog. For example:

- DSChief, the previous `MCD_ADM`, will not be mapped to a new key.
- The vote delegate factory `v1`, mapped to `VOTE_DELEGATE_FACTORY_LEGACY`, will not be mapped to a new value and thus the address will be removed from the mapping.
- `MCD_GOV_ACTIONS` is removed from the Chainlog as part of a cleanup.

6.3 Migration Considerations

Note Version 1

Users, governance and all other involved parties should be aware that the migration requires strong coordination and adherence to timelines as otherwise governance could be left vulnerable to attacks.

The migration of voting power should occur only after the spell has been executed (migration to Chief). In case of early migration, voting power in the still active DsChief will be low, allowing for executing emergency actions with low voting power (e.g. dropping spells). Additionally, spells could be plotted, forcing SKY to be converted back to MKR to drop the spell.

The migration of voting power should occur as quickly as possible after the spell execution. Note that in case a malicious spell is scheduled right before the migration spell execution, MKR governance will have no time to drop the spell. Thus, the new SKY governance could be forced to drop the malicious spell. However, if shareholders are not migrating immediately, that might reduce the time to react significantly as the new governance might not be launched quickly enough due to the launch threshold. Thus, governance could be left in a vulnerable state where it cannot react in emergency situations.

Launch threshold must be carefully assessed. Consider the following

- The launch threshold defines the initial resilience of the governance. A low threshold might enable governance attacks.
- The launch threshold, to a certain degree, defines the time needed to migrate sufficient voting power to launch the new governance. In case, the launch threshold is set very high, the launch

might be delayed for too long, leaving the governance vulnerable to attacks performed as outlined above. In the worst case, governance would not be able to launch at all.

- Similarly, note that large shareholders could potentially artificially delay the launch if they plan to attack. Similarly, if the vote does not pass with a very strong majority, it could be that the minority could similarly delay the launch to launch an attack (if coordinated minority).

Other potential delays are possible during the migration of governance voting power through LockstakeEngine:

- The position migration requires several steps including manually selecting vote delegate again.
- The position migration will be blocked until the Osm has a valid current price, see Osm Updates May Delay Lockstake Positions Migration.

Preparations should be taken. Optimally, the vote delegates and similar should be deployed prior to migration so that the migration can take place efficiently. Similarly, frontends should be prepared accordingly and other preparations should be taken to allow running the migration as efficiently and smoothly as possible.

6.4 Osm Updates May Delay Lockstake Positions Migration

Note Version 1

To populate the osm with valid current price, two consecutive `poke()` need to be called respecting the `hop` (one hour) delay.

Given that the following timeline is unclear:

1. When `skyOracle` grants buds role to `skyOsm`.
2. When first two calls to `skyOsm.poke()` are executed.

It is not guaranteed that the osm already has a valid price during the initialization (`initMigration`). Hence, calling `spotter.poke(cfg.ilk)` in `initLockstake` does not guarantee to file a valid price. Consequently, the Lockstake position migration could potentially be blocked before the first two osm updates.

6.5 Remaining Authorizations

Note Version 1

Note that it is intended that several authorizations and configurations remain present in the system:

- MKR authorization for the flopper and DssVestMintable are expected to remain. Eventually, the authorization can be revoked as part of a clean-up.
- The End contract is authorized on a wide range of contracts. However, since the ESM will become disabled it is unclear what the future use of the End will be (e.g. in a shutdown initiated by governance itself). Thus, the End contract remains for the time being.
- Last, other potential configuration changes are potentially not applied (e.g. pausing the `REWARDS_LSMKR_USDS` to prevent unnecessary staking) as they are not strictly required and could potentially lead to unforeseen consequences.

6.6 Users Staking Rewards in the Last Period May Be Lost

Note Version 1

Function `initMigration()` will switch the `splitter.farm` from the legacy one (`REWARDS_LSMKR_USDS_LEGACY`) to the new one (`REWARDS_LSSKY_USDS`). Consequently, the rewards that are not kicked in the last period may directly go into the new staking contract instead of the legacy one.

In theory, `vow.flap()` should be called before the migration to distribute the rewards fairly (separately from the spell due to DoS possibilities). Though, by the time of this review (April 2025), the burn ratio in the splitter is set to `100%`.