coinspect

You build, we defend.

Smart Contract Audit

Capyfi

June, 2025

Capyfi
Smart Contract Audit

Version: v250610 | Prepared for: Ripio | June 2025

Security Assessment

1. Executive Summary

In **May, 2025**, **Ripio** engaged Coinspect to perform a Smart Contract Audit of Capyfi.

The objective of the project was to evaluate the security of the smart contracts forked from Compound alongside the deployment scripts, parameters and actual deployment.

Capyfi is a decentralized lending protocol operating as a fork of Compound V2 on the Ethereum and LaChain blockchains. Its primary objective is to provide standard lending and borrowing functionalities while introducing a significant modification: an admin-managed whitelist that controls the minting of `cTokens`. This restricts which addresses can supply assets, offering a permissioned layer to the protocol.

✓ Solved	⚠ Caution Advised	✗ Resolution Pending
High 1	High 0	High 0
Medium 0	Medium 0	Medium 0
Low 0	Low 0	Low 0
No Risk 2	No Risk 0	No Risk 0
Total **3**	Total **0**	Total **0**

2. Summary of Findings

This section provides a concise overview of all the findings in the report grouped by remediation status and sorted by estimated total risk.

2.3 Solved issues & recommendations

These issues have been fully fixed or represent recommendations that could improve the long-term security posture of the project.

Id	Title	Risk
CAPY-01	Adversaries might drain the protocol as 1 `caUXD` will always be worth 1 USD	High
CAPY-02	Price oracle does not check for stale prices	None
CAPY-03	Hardcoded blocksPerYear could lead to interest rate miscalculations in future deployments	None

3. Scope

The scope was set to be the modifications made from the forked commit of Compound V2 up to Capyfi's commit `dad6d138d8ad7d955c3e5de3a49c73ff913945e2` of the https://github.com/LaChain/capyfi-sc repository.

Capyfi forked Compound's V2 state from the repository https://github.com/compound-finance/compound-protocol at commit `a3214f67b73310d547e00fc578e8355911c9d376` and commited this state to Capyfi's repository at commit `69714689cad05aa63845c9996c2b1837dfc0dd21`.

Coinspect reviewed the modifications to the base Compound V2 protocol between `69714689cad05aa63845c9996c2b1837dfc0dd21` and `dad6d138d8ad7d955c3e5de3a49c73ff913945e2` at Capyfi's repository.

Additionally, Coinspect reviewed the state of the deployed smart contracts at these Ethereum addresses:

Core

Component	Address
Unitroller	`0x0b9af1fd73885aD52680A1aeAa7A3f17AC702afA`
Comptroller	`0x00dc4965916e03A734190fA382633657c71f867E`
Oracle	`0xfbA2712d3bbcf32c6E0178a21955b61FE1FF424A`
Multisig	`0x6C15e4Bc44CC5674b1d7956D0e9596d2E509eD24`
Whitelist	`0x4B3535047C2B193Bc0c95ED2EF435A6BA6Dc2609`

Interest Rate Models

Model	Address
iRM_UXD_Updateable	`0xf5FA0EA9C6b7bE2da713F8BDec9D35AAE289E5c0`

iRM_WETH_Updateable	0x03c1cF154d621E0Fd7e2b88be3aE60CCf07Aca31
iRM_LAC_Updateable	0x254FCeeece1893c0A55bC7cF8A8a1C21cB05C29C
iRM_WBTC_Updateable	0xcA142dA9286D37211e7e04FEc59D1de5de86EF33
iRM_USDT_Updateable	0x5BeA6bE1DCcD0b5066842d42852e6302d7f668e8
iRM_USDC_Updateable	0xa6b02274f7B017C96d570bc2693119c90533E9c3

cTokens

CToken	Address
caUXD	0x98Ac8AC56d833bD69d34F909Ac15226772FAc9aa
caETH	0x37DE57183491Fa9745d8Fa5DCd950f0c3a4645c9
caLAC	0x0568F6cb5A0E84FACa107D02f81ddEB1803f3B50
caWBTC	0xDa5928d59ECE82808Af2cbBE4f2872FeA8E12CD6
caUSDT	0x0f864A3e50D1070adDE5100fd848446C0567362B
caUSDC	0xc3aD34De18B59A24BD0877e454Fb924181F09C8f

Underlying Tokens

Token	Address
uxd	0x0f6011F7DBC40c17EcE894b1147f4ecfA712b600
lac	0x0Df3a853e4B604fC2ac0881E9Dc92db27fF7f51b
wbtc	0x2260FAC5E5542a773Aa44fBCfeDf7C193bc2C599
usdt	0xdAC17F958D2ee523a2206206994597C13D831ec7
usdc	0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48

4. Assessment

The Capyfi protocol largely mirrors Compound V2's functionality, enabling users to supply assets to earn interest and borrow assets against collateral. The core innovation is the Whitelist contract, which, managed by designated administrators, dictates who can mint `cTokens` and thus supply liquidity. This whitelist check is integrated into the `mintInternal()` function within `CToken`. The protocol uses an oracle for asset pricing and has adjusted its interest rate model parameters for Ethereum's current block times. Administrative control over the whitelist includes adding or removing authorized minters and activating/deactivating the whitelist system. The Whitelist contract is also designed to be an UUPS upgradeable proxy by its admin.

4.1 Security assumptions

For this security assessment, Coinspect made the following assumptions:

- Users understand that supplying assets (minting) is a permissioned action controlled by administrators via the whitelist.
- The administrative roles, including the protocol's multisig and the whitelist's `ADMIN_ROLE`, are managed securely and competently.
- The designated multi-sig address is secure, and its keyholders are honest, competent, and not compromised.
- Underlying `ERC20` tokens integrated into the protocol are well-behaved (e.g., conform to ERC20 standard, no malicious hooks in `transfer`/`transferFrom`).

4.2 Decentralization

Capyfi incorporates centralized elements through its administrative controls. The general protocol parameters, inherited from Compound V2, are expected to be managed by a multisig address (`0x6C15e4Bc44CC5674b1d7956D0e9596d2E509eD24`). More specifically, the new whitelist functionality is governed by an `ADMIN_ROLE` associated with the Whitelist contract. This admin has the power to add/remove addresses from the minting whitelist, enable/disable the whitelist checks, and upgrade the whitelist contract itself. Users must be aware of these significant powers vested in the admins, as they directly impact access to supply-side participation and can alter the whitelist logic. The security and operational practices of these admin entities are critical.

4.3 Testing

Capyfi employs Foundry for its testing environment and has developed new tests for the recently added whitelist functionality. However, Coinspect noted that the comprehensive test suite from the base Compound V2 protocol was discarded during the port to Foundry framework. This absence of foundational tests significantly elevates the protocol's risk profile.

Key risks include:

- Potential for undetected regressions or vulnerabilities within the forked Compound V2 core logic.
- Loss of coverage for numerous edge cases and complex interactions previously validated by the original tests. The lack of this broad test coverage for the underlying protocol increases the possibility of undiscovered issues in the well-established but now less-tested base components, and makes future codebase maintenance and modifications more challenging and risk-prone.

4.4 Code quality

Capyfi's approach to code quality involves minimizing modifications to the original Compound V2 code to maintain clarity on the changes introduced, primarily the whitelist system. The documentation provides a good overview of this new feature.

4.5 Deployment review

Overall, the parameters configured reflect the values from the main production configuration file `ProdConfig.sol` and proxies are initialized.

However, Coinspect identified several concerns that should be addressed/considered:

1. The multisig `0x6C15e4Bc44CC5674b1d7956D0e9596d2E509eD24` that manages the protocol is a 2 out of 5 wallet with the following owners:

- `0x5CA3F8EEBa12D83408fc097c2dAd79212456F20F`,

- `0x23ceC92F92bde95e401f0a2b50b072A6069dFBd5`
- `0x5b72e13f78FEB8f5b44392f2e32940D4f37FA313`
- `0x9850b4F631F1cae37bb1C42C8004ffc2Cd31DcBe`
- `0x00A74411DDBC50C04353543d5D3f4296936DA645`

This means that attackers have to compromise less than the 50% of the accounts to take control of the whole protocol. It is suggested to increase the multisig's threshold to represent at least a 50%.

2. The only admin of the Whitelist smart contract is not the multisig

Coinspect observed that the account in control of the Whitelist smart contract is `0x6a138bd6d69feb3c2f5426549e60e644778ad04c`. Also, the multisig has no administrator privileges on this smart contract. This account has multiple interactions with other protocols such as ZkSync `Bridgehub` at `0x303a465B659cBB0ab36eE643eA362c509EEb5213`. By analyzing previous transactions from this account, it is likely that this account belongs to the deployer used on the Foundry suite. Coinspect strongly recommends to grant these privileges to the multisig and remove any ownership granted to this account.

3. Whitelist smart contract is not set for any cToken

Coinspect identified that currently, there are is no whitelist address specified on any cToken. This means that the newly added feature is currently unused. By the time of this review, caWBTC has issued nearly $6.5MM in value.

5. Detailed Findings

CAPY-01

Adversaries might drain the protocol as 1 `caUXD` will always be worth 1 USD

Status
Solved

✓

Resolution
Fixed

Risk
High

🟥🟥🟥

Impact
High

Likelihood
High

Location

```
script/config/ProdConfig.sol
src/contracts/SimplePriceOracle.sol
```

Description

Adversaries might drain the protocol by arbitraging `caUXD` against different markets since this asset has a fixed price.

The Capyfi protocol's deployment configuration, specifically within `ProdConfig.sol`, sets a hardcoded `fixedPrice` representing $1.00 for the `caUXD` cToken. This configuration is intended to maintain a 1:1 peg with the US dollar.

```
config: CtokenConfig({
    collateralFactor: 0,
    underlyingPrice: 1_000_000_000_000_000_000,
    reserveFactor: 0.075e18,
    chainlinkOracleConfig: chainlinkOracleConfig({
        underlyingAssetDecimals: 18,
        priceFeed: 0x0000000000000000000000000000000000000000,
        fixedPrice: 1000000000000000000
    })
})
```

This static valuation, however, renders the protocol vulnerable if the UXD stablecoin deviates from this peg. The price oracle will persistently report $1.00 for `caUXD` irrespective of its actual market value, unless directly updated.

Should UXD depeg:

- Below $1.00: Malicious attacker could deposit undervalued UXD into Capyfi, where it's treated as $1.00, enabling them to borrow assets exceeding their collateral's true market worth. The protocol might also fail to liquidate undercollateralized UXD positions due to overvaluing the collateral.
- Above $1.00: Borrowers using UXD as collateral could face unfair liquidations if other asset prices decline, as their UXD collateral would be undervalued by the protocol.

Coinspect considers the impact to be high because a UXD market price deviation, combined with this hardcoded price, could lead to direct financial losses for users, drainage and insolvency of the protocol. As for the likelihood, it is assessed to be high as all fungible assets fluctuate with market price.

Past events have shown that even major stablecoins can depeg, for instance, USDC dropped to $0.87.

Recommendation

Fetch the UXD price directly from a reliable oracle feed (e.g., Chainlink) rather than relying on a hardcoded 1:1 assumption.

Status

Fixed on commit `31f08fdb42be757cc73afe32b90dbdfa5924c771`.

The Capyfi team addressed the issue by replacing the hardcoded price with a custom, updatable oracle contract, `CapyfiAggregatorV3`.

The security depends on the reliability and security of the off-chain process and the accounts authorized to push price updates. These off-chain components were not part of the audit's scope.

As a follow-up recommendation, since the `fixedPrice` functionality within the main price oracle is now obsolete, Coinspect recommends removing this logic from the implementation.

CAPY-02

Price oracle does not check for stale prices

Status
Solved

✓

Resolution
Acknowledged

Risk
None

Impact
Recommendation

Likelihood
-

Location

`/src/contracts/PriceOracle/ChainlinkPriceOracle.sol`

Description

The `ChainlinkPriceOracle` contract, responsible for providing asset prices to the Capyfi protocol, fails to verify the freshness of data obtained from Chainlink price feeds. This omission means the oracle can return stale (outdated) prices, leading to inaccurate asset valuations within the protocol. Such inaccuracies can result in incorrect liquidations, unfair loan-to-value calculations, etc.

The `getUnderlyingPrice` function retrieves price data by calling `latestRoundData` on the configured Chainlink aggregator:

```
// Retrieve price from feed
(
    /* uint80 roundID */,
    int256 answer,
    /*uint256 startedAt*/,
    /*uint256 updatedAt*/,
    /*uint80 answeredInRound*/
) = priceFeed.latestRoundData();
// Invalid price returned by feed. Comptroller expects 0 price on
error.
if (answer <= 0) return 0;
uint256 price = uint256(answer);
```

However, it only utilizes the price (answer) and the feed's decimals, while ignoring the `updatedAt` timestamp. This timestamp indicates when the data was last updated on-chain and is essential for assessing data freshness.

This issue is considered to have no overall risk as most Chainlink feeds are generally considered reliable and robust. However, if a less reliable price feed is configured, this issue could pose a significant risk to the protocol.

Recommendation

The `ChainlinkPriceOracle` contract should be modified to incorporate stale price checks for each feed it consumes. This ensures that the prices supplied to the protocol are recent and reflect current market conditions. Alternatively, add to the price feed a fallback system to allow the protocol to keep operating under these circumstances.

Status

Acknowledged.

The Capyfi Team stated that they will consider this warning for future deployments.

CAPY-03

Hardcoded blocksPerYear could lead to interest rate miscalculations in future deployments

Status

Solved

✓

Resolution

Acknowledged

Risk

None

Impact

Recommendation

Likelihood

–

Location

`/src/contracts/BaseJumpRateModelV2.sol:24`

Description

Future deployments of the protocol on chains with a different or variable amount of blocks per year lead to miscalculations in the interest rates.

The `blocksPerYear` variable is hardcoded at the `BaseJumpRateModelV2`, which was modified to match Ethereum times.

```
* @notice The approximate number of blocks per year that is assumed by
the interest rate model (assuming 12s blocks)
*/
uint public constant blocksPerYear = 2628000;
```

Coinspect verified the deployments on LaChain and observed that this value was changed to match LaChain's expected amount of blocks per year. However, since this parameter is hardcoded in the smart contract instead of being an immutable variable set on deployment, future deployments could face the risk of using an outdated value if remain unchanged.

Recommendation

Make `blocksPerYear` an immutable variable. Alternatively, include this warning on the protocol's documentation.

Status

Acknowledged.

The Capyfi Team stated that they will consider this warning for future deployments.

6. Disclaimer

The contents of this report are provided "as is" without warranty of any kind. Coinspect is not responsible for any consequences of using the information contained herein.

This report represents a point-in-time and time-boxed evaluation conducted within a specific timeframe and scope agreed upon with the client. The assessment's findings and recommendations are based on the information, source code, and systems access provided by the client during the review period.

The assessment's findings should not be considered an exhaustive list of all potential security issues. This report does not cover out-of-scope components that may interact with the analyzed system, nor does it assess the operational security of the organization that developed and deployed the system.

This report does not imply ongoing security monitoring or guaranteeing the current security status of the assessed system. Due to the dynamic nature of information security threats, new vulnerabilities may emerge after the assessment period.

This report should not be considered an endorsement or disapproval of any project or team. It does not provide investment advice and should not be used to make investment decisions.