Bitlend Protocol Smart Contracts Security Audit Report

October 14, 2022

List of contents

1. Introduction 3
1.1. Disclaimer 3
1.2. Security Assessment Methodology 3
1.2.1 Severity Level Reference 4
1.2.2 Status Level Reference 4
1.3. Project overview 4
1.4. Audit Scope 4
2. Report 6
2.1. CRITICAL 6
2.2. MAJOR 6
2.3. WARNING 6
2.3.1 Difference between Compound and Band oracles 6
2.3.2 Forked compound version may be unaudited 7
2.4. INFO 8
2.4.1 Pragma is not locked 8
2.4.2 Named imports are not used 9
2.4.3 Centralization risk 9
2.4.4 More complex interface used for comparison 10
2.4.5 Unnecessary calculation 10
2.4.6 No way to remove asset symbol 11
2.4.7 No event for setAssetSymbol 12
2.4.8 Not all token types are supported 12
2.4.9 Expression can be simplified 13
3. Conclusion 15
4. About Oxorio 16

1 Introduction

1.1 Disclaimer

The audit makes no statements or warranties about the utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about the fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

1.2 Security Assessment Methodology

A group of auditors is involved in the work on this audit. Each of them checks the provided source code independently of each other in accordance with the security assessment methodology described below:

1. Project architecture review:

Manually code study of the architecture of the code based on the source code only to find out the errors and bugs.

2. Check the code against the list of known vulnerabilities

The verification process of the code against the constantly updated list of already known vulnerabilities maintained by the company.

3. Architecture and structure check of the security model

Study project documentation and its comparison against the code, including the study of the comments and other technical papers.

4. Result’s cross-check by different auditors

Normally the research of the project is made by more than two auditors. After that, there is a step of the mutual cross-check process of audit results between different task performers.

5. Report consolidation

Consolidation of the audited report from multiple auditors.

6. Reaudit of new editions

After the client’s review and fixes, the found issues are double-checked. The results are provided in the new audit version.

7. Final audit report publication

The final audit version is prepared and provided to the client and also published on the official website of the company.

1.2.1 Severity Level Reference

Findings discovered during the audit are classified as follows: Every issue in this report was assigned a severity level from the following:

- **CRITICAL**: A bug leading to assets theft, fund access locking, or any other loss of funds due to transfer to unauthorized parties.
- **MAJOR**: A bug that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
- **WARNING**: A bug that can break the intended contract logic or expose it to DDoS attacks.
- **INFO**: Minor issue or recommendation reported to / acknowledged by the client's team.

1.2.2 Status Level Reference

Based on the feedback received from the client's team regarding the list of findings discovered by the contractor, the following statuses were assigned to the findings:

- **NEW**: Waiting for the project team's feedback.
- **FIXED**: Recommended fixes have been made to the project code, and the identified issue no longer affects the project's security.
- **ACKNOWLEDGED**: The project team is aware of this finding. Recommended fixes for this finding are planned to be made. This finding does not affect the overall security of the project.
- **NO ISSUE**: Finding does not affect the overall security of the project and does not violate the logic of its work
- **DISMISSED**: The issue or recommendation was dismissed by the client.

1.3 Project overview

Bitlend Protocol is a fork of Compound V2.

1.4 Audit Scope

The scope of the audit includes the following smart contracts at:

- BandPriceOracle.sol

- IBandAggregator.sol

The audited commit identifier is dc2da0cd3e7dd43186a82b11a49a61bad96f726d

The re-audited commit identifiers are:

- f594682e1f3ef77888174cdf95951654a5b17e27
- 8fcdfe52a6fd02931deda4a388083fb0b45b41ce
- 50ea9069f4dc0a558d493fdc95515eade20a2ca9

2 Report

2.1 CRITICAL

No critical issues found

2.2 MAJOR

No major issues found

2.3 WARNING

2.3.1 Difference between Compound and Band oracles

Severity	WARNING
Status	FIXED

Description

It's possible that a Band oracle is misused if it's used without a proxy Compound docs says for `getUnderlyingPrice` function

> RETURNS: The price of the asset in USD as an unsigned integer scaled up by 10 ^ (36 - underlying asset decimals). E.g. WBTC has 8 decimal places, so the return value is scaled up by 1e28.

CompoundV2 oracle returns scaled value for WBTC(0xc11b1268c1a384e55c48c2391d8d480264a3a7f4)

But Band Protocol's stdReference returns value scaled up to 1e18 value, see etherscan
Compound contracts that uses this function is out of scope so we have not checked what can it lead to. It may be critical.

Also Compound do price sanity checks with TWAP from Uniswap.

> The Compound Protocol uses a View contract (“Price Feed”) which verifies that reported prices fall within an acceptable bound of the time-weighted average price of the token/ETH pair on Uniswap v2, a sanity check referred to as the Anchor price.

It may be helpful if an oracle will stop supplying a current price as has happened with LUNA/USD on Chainlink and had lead to exploits in some lending protocols.

Also Band Protocol may revert if the price is not available StdReferenceBasic.sol#L100

```
/// @notice Returns the price data for the given base/quote pair.
Revert if not available.
```

StdReferenceBasic.sol#L116

```
require(refData.resolveTime > 0, "REFDATANOTAVAILABLE");
```

While Compound's PriceOracle documents that it will return 0 in that case PriceOracle.sol#L14

```
Zero means the price is unavailable.
```

Recommendation

Check the usage and make sure it won't lead to any issues. Consider using sanity checks in proxy if not used. Like TWAP and timestamp.

Update

Scaling is fixed in PR7 Sanity checks and reverts are acknowledged.

2.3.2 Forked compound version may be unaudited

Severity	**WARNING**
Status	**NO_ISSUE**

Description

The last audit we have found have been done in March 4, 2022. https://blog.openzeppelin.com/compound-comprehensive-protocol-audit/

But the forked version contains commits added after the audit.

Recommendation

Make sure that the forked version is audited. Consider forking from audited commit if it's not.

Update

Bitlend Protocol Team's Response

Here's a conversation on the changes last made and their audits by OZ

```
- https://www.comp.xyz/t/rfp12-implementation-ctoken-cleanup/2694/8
- https://drive.google.com/file/d/1EVWDDJDuwsrDuIRPDAWHmtX0_wjWYA0t/
view
```

Oxorio's Response

The pdf report has a date March 4th. The report says that it has audited PR152. But PR152 has commits that has been done after this date.

It's not 100% clear which changes are discussed as audited in the discussion thread because they never mention a commit hash. But most likely it's all the changes in PR152 made until March 31. It's even more likely if you take into account a proposal made on April 8th.

2.4 INFO

2.4.1 Pragma is not locked

Severity	INFO
Status	FIXED

Description

In all the audited contracts pragma is not locked.

Contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors. Also solc frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks.

Recommendation

Lock the pragma, e.g.

```
pragma solidity 0.8.10;
```

Update

Fixed in PR7

2.4.2 Named imports are not used

Severity	INFO
Status	FIXED

Description

BandPriceOracle.sol#L3-L4

```
import "../PriceOracle.sol";
import "./IBandAggregator.sol";
```

Using unnamed imports is not recommended because it unpredictably pollutes the namespace. See Solidity docs.

Recommendation

Consider using named import to make sure namespace is never polluted

Update

Fixed in PR7

2.4.3 Centralization risk

Severity	INFO
Status	ACKNOWLEDGED

Description

BandPriceOracle.sol#L43 BandPriceOracle.sol#L67 A compromise to the owner account may allow the hacker to manipulate the oracle prices by changing the price aggregator's address or asset symbols.

Recommendation

In general, we recommend to use decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets or DAOs.

2.4.4 More complex interface used for comparison

Severity	INFO
Status	FIXED

Description

Comparing interfaces may be a little bit harder to read than comparing addresses. BandPriceOracle.sol#L89

```
if (priceAggregator == IBandAggregator(address(0))) {
```

Recommendation

Consider using

```
if (address(priceAggregator) == address(0)) {
```

Update

Fixed in PR7

2.4.5 Unnecessary calculation

Severity	INFO
Status	FIXED

Description

It seems that the most frequent way to use `setAssetSymbol` is to set a new symbol.

In that case there is no need to calculate keccak256 because `oldSymbol` is not set BandPriceOracle.sol#L77. You can just check its' length.

SHA3 opcode cost 30 gas.
While MLOAD and GT cost only 3 gas each.

Recommendation

Consider adding

```
if(bytes(oldSymbol).length > 0)
```

if this gas optimization is valuable before BandPriceOracle.sol#L77.

Update

We proposed to add a line before the statement to skip keccak256 calculation in case of `bytes(oldSymbol).length == 0`. Like

```
string memory oldSymbol = assetSymbols[asset];
if(bytes(oldSymbol).length > 0){
    if(keccak256(abi.encodePacked(oldSymbol)) ==
keccak256(abi.encodePacked(newSymbol))){
        revert BadAssetSymbol();
    }
}
```

Or even better to merge them

```
if(bytes(oldSymbol).length > 0 &&
keccak256(abi.encodePacked(oldSymbol)) ==
keccak256(abi.encodePacked(newSymbol))){
    revert BadAssetSymbol();
}
```

However removing this check is also ok. It will save even more gas if you don't consider this check necessary.
So it can be considered fixed in PR7

2.4.6 No way to remove asset symbol

Severity	**INFO**
Status	**FIXED**

Description

There is no way to remove an already added asset from the `assetSymbols` mapping, only to map another symbol onto it.

Recommendation

Fixed in PR7

2.4.7 No event for `setAssetSymbol`

Severity	INFO
Status	FIXED

Description

Subscribing to an asset symbol change may be useful for the protocol users or for monitoring.

Recommendation

Consider emitting an event on asset symbol change.

Update

Fixed in PR7

2.4.8 Not all token types are supported

Severity	INFO
Status	FIXED

Description

`decimals` function in ERC20 standard is not mandatory. However in practice, these functions are usually just assumed to be implemented. Most if not all well-known tokens include them.

Some tokens may have decimals more than 18 and this tokens are not supported because of BandPriceOracle.sol#L169

```
return _price * (10**(18 - tokenDecimals));
```

Recommendation

Make sure to check decimals as part of token integration checklist.

Update

Bitlend's response

Fix: PR11

Oxorio's response

Correctness of the line depends on supporting tokens with decimals more than 18.

If you want to support tokens with decimals more than 18, you should take into account that the expression from BandPriceOracle.sol#L169

```
(_price / (10**(tokenDecimals - 18)))
```

will return zero when `_price` will have a small value (for example, if `_price=5` and `tokenDecimals=20`). We propose to document that behaviour and consider reverting of execution in that cases.

If you don't want to support tokens with decimals more than 18, we propose to add that restriction of decimals as part of token integration checklist.

Bitlend's response

Fix: PR12

This pr fixes the INFO 2.4.8: Not all token types are supported

We decided to not not whitelist tokens with decimals > 18.

2.4.9 Expression can be simplified

Severity	INFO
Status	ACKNOWLEDGED

Description

Token on `BTT_ADDRESS` (0x0000000000000000000000000000000000001010) has `decimals` function. It returns 18.

The contract is not verified but it looks that the source code can be found here
So the expression BandPriceOracle.sol#L167

```
uint8 tokenDecimals = _underlying == BTT_ADDRESS ? 18 :
EIP20Interface(_underlying).decimals();
```

can be simplified to

```
uint8 tokenDecimals = EIP20Interface(_underlying).decimals();
```

Recommendation

Consider simplifying the expression

3 Conclusion

The following table contains the total number of issues that were found during audit:

Level	Amount
CRITICAL	0
MAJOR	0
WARNING	2
INFO	9
Total	11

Smart contracts have been audited and no critical or major issues were found. Also several recommendations were marked as warning and informational. Some changes were proposed to follow best practices, reduce potential attack surface, simplify code maintenance and increase its readability.

Because the scope of this audit was relatively narrow influence of `Difference between Compound and Band oracles` issue on Compound code was not checked. Band Oracle implementation have not been checked for the same reason.

After the re-audit all the important issue were either fixed, acknowledged or marked as not an issue. Some minor recommendations that are not security issues are in a discussion right now.

4 About Oxorio

Oxorio is a young but rapidly growing audit and consulting company in the field of the blockchain industry, providing consulting and security audits for organizations from all over the world. Oxorio has participated in multiple blockchain projects where smart contract systems were designed and deployed by the company.

Oxorio is the creator, maintainer, and major contributor of several blockchain projects and employs more than 5 blockchain specialists to analyze and develop smart contracts.

Contacts:

- oxor.io
- ping@oxor.io
- github
- linkedin