# COVER PROTOCOL COVER FLASHLOANS SMART CONTRACT AUDIT

March 10, 2021

MixBytes()

# CONTENTS

# 1. INTRODUCTION

## 1.1 DISCLAIMER

The audit makes no statements or warranties about 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 fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Cover Protocol. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

## 1.2 PROJECT OVERVIEW

Cover Protocol provides peer to peer coverage with fungible tokens. It lets the market set coverage prices as opposed to a bonding curve.

# 1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> Cross check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level
Provide the client with an interim report

05 Bug fixing & re-check.
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

## FINDINGS SEVERITY BREAKDOWN

| Level | Description | Required action |
| --- | --- | --- |
| Critical | Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party | Immediate action to fix issue |
| Major | Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement. | Implement fix as soon as possible |
| Warning | Bugs that can break the intended contract logic or expose it to DoS attacks | Take into consideration and implement fix in certain period |
| Comment | Other issues and recommendations reported to/acknowledged by the team | Take into consideration |

Based on the feedback received from the Customer's team regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

| Status | Description |
| --- | --- |
| Fixed | Recommended fixes have been made to the project code and no longer affect its security. |
| Acknowledged | The project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project. |
| No issue | Finding does not affect the overall safety of the project and does not violate the logic of its work. |

## 1.4 EXECUTIVE SUMMARY

The functionality of this project will well expand the functionality of the Cover protocol. Users will now be able to use flashloans. Support for the DAI token accelerates the work of this project. The main logic is located in the CoverFlashBorrower contract. For compatibility with other applications, the project was developed according to the EIP-3156 standard.

## 1.5 PROJECT DASHBOARD

| | |
|---|---|
| **Client** | Cover Protocol |
| **Audit name** | Cover Flashloans |
| **Initial version** | bea8cd444aa1791a97b2a81078dc887a835f7949<br>4f6cbd2659ae02c1fcd9d79da994684fa5c94409<br>b43bc15deae11db231af3fa4c0ea3843f9565db3<br>0a31e67acf52fafdeed42f366e88dc70c1b5592e<br>9cb6561f3d40f870252088d3c16c1eed24a93166 |
| **Final version** | 9cb6561f3d40f870252088d3c16c1eed24a93166 |
| **SLOC** | 245 |
| **Date** | 2021-02-15 - 2021-03-10 |
| **Auditors engaged** | 2 auditors |

## FILES LISTING

| | |
|---|---|
| CoverFlashBorrower.sol | CoverFlashBorrower.sol |
| IERC20.sol | IERC20.sol |
| IYERC20.sol | IYERC20.sol |
| SafeERC20.sol | SafeERC20.sol |
| IBPool.sol | IBPool.sol |
| ICover.sol | ICover.sol |
| IERC3156FlashBorrower.sol | IERC3156FlashBorrower.sol |
| IERC3156FlashLender.sol | IERC3156FlashLender.sol |
| IFlashBorrower.sol | IFlashBorrower.sol |
| IProtocol.sol | IProtocol.sol |
| Address.sol | Address.sol |
| Context.sol | Context.sol |
| Ownable.sol | Ownable.sol |

## FINDINGS SUMMARY

| Level | Amount |
|---|---|
| Critical | 0 |
| Major | 2 |
| Warning | 4 |
| Comment | 2 |

## CONCLUSION

Smart contracts have been audited and several suspicious places have been spotted. During the audit no critical issues were found, two issues were marked as major because they could lead to some undesired behavior, also several warnings and comments were found and fixed by the client. After working on the reported findings all of them were resolved or acknowledged.Findings list: Level Amount CRITICAL - MAJOR 1 WARNING 4 COMMENT 2 Final commit identifier with all fixes: `9cb6561f3d40f870252088d3c16c1eed24a93166`

# 2. FINDINGS REPORT

## 2.1 CRITICAL

Not Found

## 2.2 MAJOR

| MJR-1 | Error while calculating the value |
| --- | --- |
| File | CoverFlashBorrower.sol |
| Severity | Major |
| Status | Fixed at 0a31e67a |

### DESCRIPTION

The `getPricePerFullShare()` function for a yDAI token contract returns the value multiplied by a factor of 1e18.
But it doesn't divided by 1e18:

- CoverFlashBorrower.sol#L90
- CoverFlashBorrower.sol#L155
- CoverFlashBorrower.sol#L182

### RECOMMENDATION

It is recommended to fix the error.

| MJR-2 | Wrongly used `safeApprove` |
| --- | --- |
| File | CoverFlashBorrower.sol |
| Severity | Major |
| Status | |

## DESCRIPTION

At the line CoverFlashBorrower.sol#L351 contract call `safeApprove` method, however that method fails if account have remaining allowed tokens.

## RECOMMENDATION

We suggest to reset approval calling

```
ERC20(token).safeApprove(recipient, 0);
```

before setting new approval

## CLIENT'S COMMENTARY

> No comment received from the team

## 2.3 WARNING

| | |
|---|---|
| **WRN-1** | No processing of function execution results |
| **File** | CoverFlashBorrower.sol<br>SafeERC20.sol |
| **Severity** | Warning |
| **Status** | Fixed at 9cb6561f |

### DESCRIPTION

The CoverFlashBorrower.sol contract uses the `approve ()`, `transferFrom ()` and `transfer ()` functions to transfer tokens. According to the ERC20 standard, functions return boolean variables, but the result is not processed.
This can be seen in the following lines:

- CoverFlashBorrower.sol#L97
- CoverFlashBorrower.sol#L130
- CoverFlashBorrower.sol#L143
- CoverFlashBorrower.sol#L231
- CoverFlashBorrower.sol#L233
- CoverFlashBorrower.sol#L235
- CoverFlashBorrower.sol#L264
- CoverFlashBorrower.sol#L284
- CoverFlashBorrower.sol#L286
- CoverFlashBorrower.sol#L335

At the same time, the project has a library for safe work with tokens for the ERC20 standard. This library is located here: SafeERC20.sol.

### RECOMMENDATION

For operations with tokens, use functions from the SafeERC20.sol library.

| | |
|---|---|
| WRN-2 | Event is probably missing |
| File | CoverFlashBorrower.sol |
| Severity | Warning |
| Status | Fixed at 9cb6561f |

## DESCRIPTION

At the line CoverFlashBorrower.sol#L134-L137
`CoverFlashBorrower`'s method `setFlashLender()` should probably emit an event.

## RECOMMENDATION

It is recommended to create event: `NewFlashLender`.

| | |
|---|---|
| **WRN-3** | There is no check for the available amount of tokens |
| **File** | CoverFlashBorrower.sol |
| **Severity** | Warning |
| **Status** | Fixed at 4f6cbd26 |

## DESCRIPTION

Before taking an flashloan, you need to make sure that there are enough tokens.
But now this check is not being done.
The lines: CoverFlashBorrower.sol#L98 and CoverFlashBorrower.sol#L131 need to be checked:

```
require (amountDaiNeeded <= flashLender.maxFlashLoan (address (dai)),"  wrong amount of token
");
```

Additional checks keep the source code clean.
Additional checks do not require gas consumption when making a transaction.

## RECOMMENDATION

It is recommended to add checks for the available amount of tokens

| | |
|---|---|
| **WRN-4** | Should return zero. But instead the transaction crashes |
| **File** | CoverFlashBorrower.sol |
| **Severity** | Warning |
| **Status** | Fixed at **b43bc15d** |

## DESCRIPTION

Functions are used to calculate the number of tokens. But in some cases, an arithmetic overflow occurs and the transaction is reverted. This can be avoided by doing a check and returning zero.

- At the line CoverFlashBorrower.sol#L163 should be done like this:

```
if (amountDaiNeeded + flashFee < daiReceivedFromSwap) {
 totalCost = 0;
 } else {
 totalCost = amountDaiNeeded - daiReceivedFromSwap + flashFee;
 }
```

- At the line CoverFlashBorrower.sol#L186 should be done like this:

```
if (daiReceivedFromRedeem < amountDaiNeeded + flashFee) {
 totalReturn = 0;
 } else {
 totalReturn = daiReceivedFromRedeem - amountDaiNeeded - flashFee;
 }
```

## RECOMMENDATION

It is recommended to fix the code

## 2.4 COMMENTS

| | |
|---|---|
| CMT-1 | Using "magic" numbers/span> |
| File | CoverFlashBorrower.sol |
| Severity | Comment |
| Status | Acknowledged |

### DESCRIPTION

The use in the source code of some unknown where taken values impairs its understanding.
On line CoverFlashBorrower.sol#L185 the value is `10000`.

### RECOMMENDATION

It is recommended that you create constants with meaningful names for using numeric values.

### CLIENT'S COMMENTARY

> Comment is regarding the variable naming for a view function. We could redeploy, but it's only a clarity issue, not functional.

| | |
|---|---|
| CMT-2 | Hardcoded addresses/span> |
| File | CoverFlashBorrower.sol |
| Severity | Comment |
| Status | Fixed at 9cb6561f |

## DESCRIPTION

At the lines CoverFlashBorrower.sol#L23-L24
the DAI and yDAI addresses is hardcoded. But if the contract deployed to some other testnet it must be different.

## RECOMMENDATION

It is recommended to make these addresses as arguments to the initialization method.

# 3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build open-source solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

## BLOCKCHAINS

 Ethereum

 Cosmos

 EOS

 Substrate

## TECH STACK

 Python

 Solidity

 Rust

 C++

## CONTACTS

 https://github.com/mixbytes/audits_public

 https://mixbytes.io/

 hello@mixbytes.io

 https://t.me/MixBytes

 https://twitter.com/mixbytes