COVER CORE V2 SMART CONTRACT AUDIT

February 25, 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 audited scope implements custom-token insurance protocol. The project have 3 logical modules: cover contract itself with statements to control insurance conditions, claim management for claims filed for cover pool, cover pool to manage covers for pool. Such project could be used as an insurance for funds.

1.5 PROJECT DASHBOARD

Client	Cover Protocol
Audit name	Cover-core-v2
Initial version	513f5e502a8e8a623729c2c3480fca4e80fdef39
Final version	845e33cca83d05bd907dec902f6942fcaa59f030
SLOC	624
Date	2021-01-15 - 2021-02-25
Auditors engaged	2 auditors

FILES LISTING

Cover.sol	Cover.sol
CoverPool.sol	CoverPool.sol
CoverPoolFactory.sol	CoverPoolFactory.sol
CoverERC20.sol	CoverERC20.sol
ClaimManagement.sol	ClaimManagement.sol
ClaimConfig.sol	ClaimConfig.sol
BasicProxyLib.sol	BasicProxyLib.sol
StringHelper.sol	StringHelper.sol
EIP712.sol	EIP712.sol
ERC20Permit.sol	ERC20Permit.sol
SafeERC20.sol	SafeERC20.sol
ERC20.sol	ERC20.sol
Ownable.sol	Ownable.sol
Address.sol	Address.sol
ReentrancyGuard.sol	ReentrancyGuard.sol
Initializable.sol	Initializable.sol
Create2.sol	Create2.sol
Proxy.sol	Proxy.sol
BaseUpgradeabilityProxy.sol	BaseUpgradeabilityPro...
BaseAdminUpgradeabilityProxy.sol	BaseAdminUpgradeabili...
InitializableAdminUpgradeabilityProxy.sol	InitializableAdminUpg...

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	2
Warning	5
Comment	9

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 or some misunderstanding, also several warnings and comments were found and discussed with the client. After working on the reported findings all of them were resolved or acknowledged (if the problem was not critical).Final commit identifier with all fixes: `845e33cca83d05bd907dec902f6942fcaa59f030`

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Incorrect check of `timeWindow`
File	ClaimConfig.sol
Severity	Major
Status	Fixed at 845e33cc

DESCRIPTION

At the line ClaimConfig.sol#L50 there is incorrect check of `_newTimeWindow`

RECOMMENDATION

Change to `require(_newTimeWindow >= 3 days, "CC: window too short");`

MJR-2	Lack of claim validation
File	ClaimManagement.sol
Severity	Major
Status	Fixed at 56123df7

DESCRIPTION

At lines:

- ClaimManagement.sol#L112
- ClaimManagement.sol#L145
- ClaimManagement.sol#L176

the claim is taken by `_coverPool, _nonce, _index`, however caller may send incorrect indexes to the method.

At the line ClaimManagement.sol#L114 even the flow with invalid claim will pass the require condition and go to

```
claim.state = ClaimState.Validated;
_resetCoverPoolClaimFee(_coverPool);
```

this is unexpected behavior and potentially can lead to the contract misfunctioning.

RECOMMENDATION

Add `require(_index < coverPoolClaims[_coverPool][_nonce].length, "bad indexes")`

2.3 WARNING

WRN-1	Lack of `onlyOwner` modifier in `Cover.deploy`
File	Cover.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At Cover.sol#L239 anyone may call deploy. It's not a big thing now (at least in current implementation), but is rather an unexpected permission. Adding of `onlyOwner` modifier will make the code robust.

RECOMMENDATION

Add `onlyOwner` modifier.

WRN-2	Too soft check in `addCover`
File	CoverPool.sol
Severity	Warning
Status	Fixed at 56123df7

DESCRIPTION

At CoverPool.sol#L123 it's not clear why do we require only `received>0` not `received==_amount`

RECOMMENDATION

Add `require(received==_amount)` or document and argue in code-comments and in the project's docs why it's so relaxed.

WRN-3	Unnecessary getter method
File	ClaimConfig.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

The method `getCVCList` at ClaimConfig.sol#L91 is not needed because cvcMap is already defined as public attribute and has default getter.

RECOMMENDATION

Remove custom getter to save gas and use default one.

WRN-4	Unused modifier
File	CoverPool.sol CoverPool.sol
Severity	Warning
Status	Fixed at 56123df7

DESCRIPTION

`CoverPool`'s modifier `onlyGov` defined at line CoverPool.sol#L67 is never used within the contract. In the same time there are methods having requires just within the methods:

```
require(msg.sender == _factory().governance(), "CoverPool: caller not governance");
```

Seems like the methods should have used the modifier:

- CoverPool.sol#L207
- CoverPool.sol#L220

RECOMMENDATION

Use the modifier instead of the `require` above.

WRN-5	Event is probably missing
File	CoverPoolFactory.sol
Severity	Warning
Status	Fixed at 56123df7

DESCRIPTION

At line CoverPoolFactory.sol#L89 `CoverPoolFactory`'s method `updateDeployGasMin` should probably emit an event as well as other methods for update (e.g. `updateCoverPoolImpl` or `updateCoverImpl`, etc.) do.

RECOMMENDATION

Create a suit event and emit the one in the method.

2.4 COMMENTS

CMT-1	Hard-coded DAI address
File	ClaimConfig.sol
Severity	Comment
Status	Fixed at 56123df7

DESCRIPTION

At ClaimConfig.sol#L14 the `DAI` address is hardcoded. But if the contract deployed to some other testnet it must be different.

RECOMMENDATION

Make the `DAI` address an argument to init method.

CMT-2	Magic hard-coded constants
File	ClaimConfig.sol Cover.sol
Severity	Comment
Status	**Fixed** at **56123df7**

DESCRIPTION

There are some magic constants in the middle of the code:

- ClaimConfig.sol#L50
- Cover.sol#L218
- Cover.sol#L282

RECOMMENDATION

Make them class-level named constants.

CMT-3	Debateable gas usage
File	ClaimConfig.sol Cover.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

In the method `removeCVCForPool` at ClaimConfig.sol#L104 the new memory structure is created and then filled with all elements `!= _cvc`. However it's not clear if it is really cheaper than just do.

```
function removeCVCForPool(address _coverPool, address _cvc, uint256 _cvt_index) public
override onlyOwner {
  require(cvcMap[_coverPool][_cvt_index] == _cvc, "incorrect index");
  cvcMap[_coverPool][_cvt_index] = cvcMap[_coverPool][cvcMap[_coverPool].length-1];
  delete cvcMap[_coverPool][cvcMap[_coverPool].length--];
}
```

Also at Cover.sol#L134 it's not clear if it is really cheaper than straight-forward approach.

RECOMMENDATION

Add gas performance tests or argue the optimal way as a comment in the code.

CMT-4	Governance cannot be creator of the contract
File	ClaimManagement.sol
Severity	Comment
Status	**Fixed** at **56123df7**

DESCRIPTION

At the line ClaimManagement.sol#L22 it's required that governance may not be creator of the contract. However it's not clear why.

RECOMMENDATION

Let the governance be creator of the contract or comment why it should not be like this.

CMT-5	if-not-return statements used instead of require
File	Cover.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At Cover.sol#L132 the check does nothing on fail, it just returns.
This is not a regular way to do this kind of checks, usually people just write `require` statement. It seems that these if-not-return statements are used to allow multi transactional initialization. It's debateable if it's the best way to do it. And it's better to explicitly document it in the method's docstring and usage.

RECOMMENDATION

Add more comments.

CMT-6	Unclear part of code which burns token after redeem
File	Cover.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

It's not really clear what is the business logic behind the code-block at
Cover.sol#L166
as far as I understood it just burns unused tokens caused by the rest of division.

This uncertainty and the fact that it's not obvious (at least for me) what does the block of code do, increases the chance of mistake and makes review harder.

RECOMMENDATION

Add more comments.

CMT-7	Unclear business logic behind `_futureToken`, `futureCovTokenMap`, `futureCovTokens`
File	Cover.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

It's not really clear what is the business logic behind the usage of `_futureToken`, `futureCovTokenMap`, `futureCovTokens` and the code-block at Cover.sol#L225 as far as I understood it switches to new claim tokens but it's not clear. Also there is no any explanation in the product docs.

This uncertainty and the fact that it's not obvious (at least for me) what does the block of code do, increases the chance of mistake and makes review harder.

RECOMMENDATION

Add more comments.

CMT-8	Lack of `require(len>0)` in `_handleLatestFutureToken`
File	Cover.sol
Severity	Comment
Status	**Fixed** at **845e33cc**

DESCRIPTION

At Cover.sol#L292 nothing will happen if `futureCovTokens` is empty, it's not revert. However this is unexpected and may cause misfunctioning in callers methods.

RECOMMENDATION

Use `require(len>0)` instead of `if` condition

CMT-9	Block timestamp type inconsistency
File	ClaimManagement.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

This comment is about multiple type shrinking of a `block.timestamp`'s `uint256` to `uint48` cases e.g. in here:

- ClaimManagement.sol#L39
- ClaimManagement.sol#L55

or in here:

- ClaimManagement.sol#L120
- ClaimManagement.sol#L171

Such a significant type shrink reduces the application's logic "time to live" duration.

RECOMMENDATION

It is recommended to keep the block timestamp being stored within the original `uint256` type. Since such recommendation requires quite a significant refactoring, this was made a comment, not a warning.

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