COVER PROTOCOL PERIPHERAL SMART CONTRACT AUDIT

January 11, 2021

COVER

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 Yearn Finance (name of Client). 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 is a blockchain-based peer-to-peer coverage market for decentralized finance. The platform allows DeFi users to hedge against risks due to smart contracts' fallacies (especially useful when farming or staking).

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 contract represents an intermediate layer between Cover Protocol core contracts and end-users. The general purpose of the contract is routing requests to target pools, so contract provide a single place to interact with pools and create them (approach is very similar with the Uniswap router model).

1.5 PROJECT DASHBOARD

Client	Yearn Finance
Audit name	Cover Peripheral
Initial version	d5b37e34d47abec3252cdabd46e55e34a72421d4
Final version	cbf6f30cde5ca6830af0554f3fb5247ae3bdaf06
SLOC	700
Date	2020-12-25 - 2021-01-11
Auditors engaged	2 auditors

FILES LISTING

CoverRouter.sol	CoverRouter.sol
Rollover.sol	Rollover.sol
Address.sol	Address.sol
Ownable.sol	Ownable.sol
ReentrancyGuard.sol	ReentrancyGuard.sol
SafeERC20.sol	SafeERC20.sol
SafeMath.sol	SafeMath.sol
IBFactory.sol	IBFactory.sol
IBlacksmith.sol	IBlacksmith.sol
IBPool.sol	IBPool.sol
ICover.sol	ICover.sol
ICoverERC20.sol	ICoverERC20.sol
ICoverRouter.sol	ICoverRouter.sol
IERC20.sol	IERC20.sol
IProtocol.sol	IProtocol.sol
IRollover.sol	IRollover.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	2
Warning	4
Comment	1

CONCLUSION

Smart contracts were audited and several suspicious places were spotted. Two issues were marked as major since they potentially can break desired contract behavior. The other issues were marked as warnings and comments. After working on the reported findings all issues were fixed by the client's team or marked as acknowledged. So, the contracts are assumed as secure to use according to our security criteria.

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	It is possible to carry out attacks to manipulate pools within one transaction using a flash loan
File	CoverRouter.sol Rollover.sol
Severity	Major
Status	Fixed at 492741bc

DESCRIPTION

In contracts CoverRouter.sol and Rollover.sol, any user can exchange tokens with a contract. Any user can add and remove liquidity. An attacker can take a flash loan and perform multiple liquidity manipulations within a single transaction. These manipulations can lead to a loss of funds for other users.

RECOMMENDATION

It is recommended to add protection against token manipulation with flash loans. Here's some sample code:

```
mapping(address => uint256) private _lastSwapBlock;

function some() external {
   _preventSameTxOrigin();
   ....
   some logic
   ...
 }

function _preventSameTxOrigin() private {
   require(block.number > _lastSwapBlock[tx.origin], "SAME_TX_ORIGIN");
   _lastSwapBlock[tx.origin] = block.number;
 }
```

MJR-2	Possibility to steal tokens from the contract balance
File	CoverRouter.sol Rollover.sol
Severity	Major
Status	Fixed at 492741bc

DESCRIPTION

Method `addCoverAndCreatePools` defined at CoverRouter.sol#L128 accepts `_protocol` and `_collateral` addresses as arguments, then call `_addCover` that makes approve for `_protocol` an unlimited amount of `_collateral` tokens and call `_protocol.addCover`. There are no checks of `_protocol` and `_collateral` validity, so an attacker can pass malicious `_protocol` and get unlimited approval of `_collateral` tokens:

```
if (_token.allowance(address(this), _spender) < _amount) {
  _token.approve(_spender, uint256(-1));
}
```

According to the contract logic in an optimistic flow a contract shouldn't have any tokens in balance, but this invariant not fully checked, e.g. if `_protocol.addCover` at Rollover.sol#L75 fails and returns false, then the transaction will be executed successfully, but the user's funds will be left on the CoverRouter balance.

RECOMMENDATION

We recommend to not use unlimited approve and add particular checks to keep the zero balance invariant.

2.3 WARNING

WRN-1	No validation of the address parameter value in contract constructor
File	CoverRouter.sol
Severity	Warning
Status	Fixed at 492741bc

DESCRIPTION

The variable is assigned to the value of the constructor input parameter. But the parameter is not checked before this. If the value turns out to be zero, then it will be necessary to redeploy the contract since there is no other functionality to set this variable.

- At line CoverRouter.sol#L36 the `protocolFactory` variable is set to the value of the `_protocolFactory` input parameter.
- At line CoverRouter.sol#L37 the `bFactory` variable is set to the value of the `_bFactory` input parameter.

RECOMMENDATION

In all the cases, it is necessary to add a check of the input parameter to zero before initializing the variables.

WRN-2	It is possible to use a Front-Running attack
File	CoverRouter.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

Since all transactions are visible in the mempool for a little while before being executed, the observers of the network can see and react to an action before it is included in a block. An example of how this can be exploited is with a decentralized exchange where a buy order transaction can be seen, and a second order can be broadcast and executed before the first transaction is included.

At line CoverRouter.sol#L64, any user can execute the `rolloverAndAddLiquidityForAccount()` function. This function calls other functions and exchanges tokens between the contract and the `_account` address. Due to the fact that another user completes the transaction earlier, a profitable position in the trade may be lost and the user will lose his profit.

RECOMMENDATION

The best remediation is to remove the benefit of Front-Running in your application, mainly by removing the importance of transaction ordering or time. It will be possible to make use of the maximum or minimum acceptable price or amount range for the transaction, thereby limiting the price slippage.

WRN-3	Need to check the remaining tokens
File	CoverRouter.sol
Severity	Warning
Status	Fixed at 492741bc

DESCRIPTION

There is the `_transferRem` method in the contract. But in some cases, there are no checks:

- CoverRouter.sol#L45 (`addCoverAndAddLiquidity`)
- CoverRouter.sol#L115 (`addLiquidity`)
- CoverRouter.sol#L158 (`createNewPool`)

RECOMMENDATION

We recommend checking all tokens and transfer remaining balance after every external method of the contract.

WRN-4	Unlimited approval
File	Rollover.sol
Severity	Warning
Status	Fixed at 492741bc

DESCRIPTION

At this line `_token.approve(_spender, uint256(-1))` located at Rollover.sol#L64 we have an unlimited approval access from CoverRouter.

RECOMMENDATION

We recommend either to revoke the approval after the execution or to approve only the expected amount.

2.4 COMMENTS

CMT-1	No events for parameter changes
File	CoverRouter.sol
Severity	Comment
Status	Fixed at 492741bc

DESCRIPTION

Basic features for `onlyOwner` don't emit any events:

- CoverRouter.sol#L179 (`setSwapFee`)
- CoverRouter.sol#L185 (`setCovTokenWeights`)

RECOMMENDATION

We recommend to create events: `SwapFeeUpdate`, `CovTokenWeightUpdate`.

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