HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: BarnBridge
Date: March 1st, 2021

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for BarnBridge.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Yield farming
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	
Commit	
Deployed contract	
Timeline	23 FEB 2021 - 1 MAR 2021
Changelog	1 MAR 2021 - INITIAL AUDIT

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by BarnBridge (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted between February 23rd, 2021 - March 1st, 2021.

Scope

The scope of the project is smart contracts in the repository:

```
Contract deployment address:
Repository
File:
      SmartYield.sol
      SeniorBond.sol
      JuniorToken.sol
      JuniorBond.sol
      Governed.sol
      CompoundController.sol
      CompoundProvider.sol
      YieldOracle.sol
```

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Functional review	■ Business Logics Review ■ Functionality Checks ■ Access Control & Authorization ■ Escrow manipulation ■ Token Supply manipulation ■ Assets integrity ■ User Balances manipulation ■ Kill-Switch Mechanism ■ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are secure. Though we recommend optimizing the code.

Insecure	Poor secured	Secured	Well-secured

You are here 

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section, and all found issues can be found in the Audit overview section.

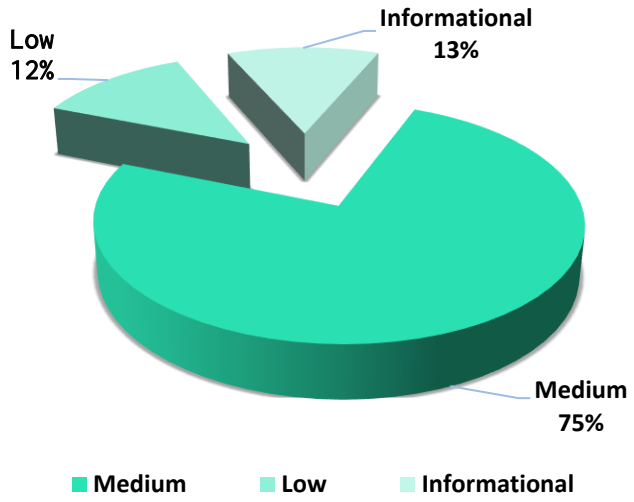
Security engineers found **6** medium, **1** low, and **1** informational issue during the audit.

Notice:

1. Logic of the CompoundProvider and the YieldOracle contratts is not described in the project documentation. We may not verify calculations correctness.
2. Contract owners may change all values in the IController without notifying users about those changes. We may not guarantee correctness of those values and that users will be secured after those changes.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Graph 1. The distribution of vulnerabilities after the first review.



Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

AS-IS overview

SmartYield.sol

Description

SmartYield is an ERC20 token contract that allows buying and selling tokens. Also provides Yield functionality for Senior Bonds and Junior Bonds.

Imports

SmartYield has following imports:

- openzeppelin/contracts/math/SafeMath.sol
- openzeppelin/contracts/token/ERC20/IERC20.sol
- ./lib/math/MathUtils.sol
- ./Governed.sol
- ./IController.sol
- ./oracle/IYieldOraclelizable.sol
- ./ISmartYield.sol
- ./IProvider.sol
- ./model/IBondModel.sol
- ./oracle/IYieldOracle.sol
- ./IBond.sol
- ./JuniorToken.sol

Inheritance

SmartYield is JuniorToken, ISmartYield.

Usages

SmartYield contract has following usages:

- SafeMath for uint256.

Structs

SmartYield contract has no custom data structures.

Enums

SmartYield contract has no enums.

Events

SmartYield contract has no custom events.

Modifiers
SmartYield has no custom modifiers.

Fields

SmartYield contract has following fields and constants:

- address public controller
- address public pool
- address public seniorBond
- address public juniorBond
- uint256 public underlyingLiquidatedJuniors
- uint256 public tokensInJuniorBonds
- uint256 public seniorBondId
- uint256 public juniorBondId
- uint256 public juniorBondsMaturitiesPrev
- uint256[] public juniorBondsMaturities
- mapping(uint256 => JuniorBondsAt) public juniorBondsMaturingAt
- mapping(uint256 => SeniorBond) public seniorBonds
- mapping(uint256 => JuniorBond) public juniorBonds
- SeniorBond public abond
- bool public _setup

Functions

SmartYield has following public functions:

- *constructor*
  Description
  Inits the contract and sets contract name and symbol.
  Visibility
  public
  Input parameters
  - string memory name_
  - string memory symbol_

  Constraints
  None
  Events emit
  None
  Output
  None
- *setup*
  Description
  Setting up addresses.
  Visibility

external
Input parameters
- address controller_
- address pool_
- address seniorBond_
- address juniorBond_

Constraints
- Can only be called once.

Events emit
None
Output
None

- *buyTokens*

Description
Buy at least *_minTokens* with *_underlyingAmount*, before *_deadline* passes.
Visibility
external
Input parameters
- uint256 underlyingAmount_
- uint256 minTokens_
- uint256 deadline_

Constraints
- The contract should not be paused.
- deadline_ should not pass yet.
- Amount of bought tokens should be at least minTokens_.

Events emit
None
Output
None

- *sellTokens*

Description
Sell *tokenAmount_* for at least *_minUnderlying*, before *_deadline* and forfeit potential future gains.
Visibility
external
Input parameters
- uint256 tokenAmount_
- uint256 minUnderlying_
- uint256 deadline_

Constraints
- deadline_ should not pass yet.
- Amount of underlying tokens should be at least minTokens_.

Events emit
None

Output
None

- *buyBond*
  Description
  Purchase a senior bond with *principalAmount_* underlying for *forDays_*. Buyer gets a bond with gain >= minGain_ or revert.
  Visibility
  external
  Input parameters
    - uint256 principalAmount_
    - uint256 minGain_
    - uint256 deadline_
    - uint16 forDays_

  Constraints
    - The contract should not be stopped.
    - deadline_ should not pass yet.
    - forDays_ should not exceed max bond life.
    - Gain should be at least minGaing_.
    - Gain should not be 0.
    - Gain should be less than underlyingLoanable..

  Events emit
  None
  Output
  None
- *buyJuniorBond*
  Description
  Buy a junior bond
  Visibility
  external
  Input parameters
    - uint256 tokenAmount_
    - uint256 maxMaturesAt_
    - uint256 deadline_

  Constraints
    - deadline_ should not pass yet.
    - The token should mature before the maxMaturesAt_.

  Events emit
  None
  Output
  None
- *redeemBond*
  Description
  Redeem a senior bond by id. Anyone can redeem but owner gets principal + gain
  Visibility

external
Input parameters
- uint256 bondId_

Constraints
- Token should be matured.

Events emit
None
Output
None

- *redeemJuniorBond*
Description
Redeem a junior bond by id.
Visibility
external
Input parameters
- uint256 jBondId_

Constraints
- Token should be matured.

Events emit
None
Output
None

- *providerRatePerDay, bondGain, currentTime, price, underlyingTotal, underlyingJuniors, underlyingLoanable, abondGain, abondPaid, abondDebt*
Description
View functions for calculation of corresponding values.

SeniorBond.sol, JuniorBond.sol

Description

Simple ERC721 tokens with minting and burning functionality limited to the SmartYield contract.

JuniorToken.sol

Description

Abstract ERC20 contract.

Governed.sol

Description

Contract that manages access for guardians and dao addresses.

CompoundController.sol

Description

Contract for storing the system parameters. Setter functions are governed.

CompoundProvider.sol

Description

CompoundProvider is the interlayer between the SmartYield and Compound.

Imports

CompoundProvider has following imports:

- openzeppelin/contracts/math/SafeMath.sol
- openzeppelin/contracts/token/ERC20/IERC20.sol
- ./../external-interfaces/uniswap/IUniswapV2Router.sol
- ./../external-interfaces/compound-finance/ICToken.sol
- ./../external-interfaces/compound-finance/IComptroller.sol
- ./../lib/math/MathUtils.sol
- ./CompoundController.sol
- ./../oracle/IYieldOracle.sol
- ./../IProvider.sol

Inheritance

CompoundProvider is IProvider.

Usages

CompoundProvider contract has following usages:

- SafeMath for uint256.

Structs

CompoundProvider contract has no custom data structures.

Enums

CompoundProvider contract has no enums.

Events

CompoundProvider contract has no custom events.

Modifiers

CompoundProvider has following modifiers:

- accountYield - performs updates.

Fields

CompoundProvider contract has following fields and constants:

- address public uToken
- address public cToken
- address public comptrolle
- address public rewardCToken
- uint256 public cTokenBalance
- uint256 public compSupplierIndexLast
- uint256 public cumulativeUnderlyingBalanceHarvestedLast;
- uint256 public harvestedLas
- bool public _setup

Functions

CompoundProvider has following public functions:

- *setup*
  Description
  Sets up addresses.
  Visibility
  external
  Input parameters
    - address smartYield_
    - address controller_
    - address cToken_

  Constraints
    - Can only be called once.

  Events emit
  None
  Output
  None
- *_takeUnderlying*
  Description
  Transfers underlying tokens from the _from address to the contract address.
  Visibility
  external
  Input parameters
    - address from_
    - uint256 underlyingAmount_

  Constraints

- Transfer should be successful.

Events emit
None
Output
None

- *_sendUnderlying*
  Description
  Send underlying tokens to *to_* address.
  Visibility
  external
  Input parameters
  - uint256 underlyingAmount_
  - uint256 takeFees_

  Constraints
  None
  Events emit
  None
  Output
  None
- *_withdrawProvider*
  Description
  Redeems underlying cToken.
  Visibility
  external
  Input parameters
  - uint256 underlyingAmount_
  - uint256 takeFees_

  Constraints
  None
  Events emit
  None
  Output
  None
- *harvest*
  Description
  Harvest rewards. Can be called by anyone. A caller receives rewards in uToken.
  Visibility
  external
  Input parameters
  None
  Constraints
  - Can only be called once per block.

  Events emit
  None
  Output

None

- *transferFees*
  Description
  Transfer fees to the fee owner.
  Visibility
  external
  Input parameters
  None
  Constraints
  - Can only be called from the SmartYield contract.

  Events emit
  None
  Output
  None
- *cumulatives*
  Description
  Updates cumulative and underlying balances. Returns cumulative state.
  Visibility
  external
  Input parameters
  None
  Constraints
  - Can only be called from the SmartYield contract.

  Events emit
  None
  Output
  - uint256 cumulativeSecondlyYield
  - uint256 cumulativeUnderlyingBalance
- *underlyingBalance, currentTime, compRewardExpected*
  Description
  View functions for calculation of the corresponding values.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Audit overview

Critical

No critical issues were found.

High

No high severity issues were found.

Medium

1. The *SmartYield* contract ubiquitously using external calls to its own functions. Such behavior increases gas consumption.
2. *JuniorBond* and *SeniorBond* codes are identical. One of them may be removed.
3. The *_takeUnderlying* function of the *CompoundProvider* has redundant allowance validation. It is a part of the *transferFrom* function.
4. Unsafe math operations are used all over the code.

   We recommend changing them to safe operations or to write more tests that covers edge cases to prevent possible overflow issues.
5. *YieldOracle* is never used except in the *providerRatePerDay* function that is only the view function.
6. The *_beforeProviderOp* function of the *SmartYield* contract may fail due to block gas limit in a case when the gap between *juniorBondsMaturitiesPrev* and *juniorBondsMaturities.length* is big enough and enough junior bonds become mature. The gap may be increased by calling the *buyJuniorBond* function. The case is hardly exploitable.

Low

1. The *_burnJuniorBond* function of the *SmartYield* contract contains commented out code.

Lowest / Code style / Best Practice

1. Some code style issues were found by the static code analyzers.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, high-level description of functionality was presented in As-Is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found **6** medium, **1** low, and **1** informational issue during the audit.

Notice:

1. Logic of the CompoundProvider and the YieldOracle contratts is not described in the project documentation. We may not verify calculations correctness.
2. Contract owners may change all values int the IController without notifying users about those changes.

Violations in the following categories were found and addressed to Customer:

Category	Check Item	Comments
Code review	▪ Gas usage	▪ Gas usage of contracts may be optimized.
	▪ Unused code	▪ The project contains unused code.
	▪ Assets integrity	▪ In rare and complex case funds may be locked.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.