HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: BNPLPAY.io
Date: January 14th, 2022

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 BNPLPAY.io.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 tokens
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/cf/bnpl-contracts-v5
Commit	c4cdd3615b135644b3e363365f8660b3ddcdc512
Technical Documentation	YES
JS tests	YES
Website	bnplpay.io
Timeline	18 OCTOBER 2021 - 12 JANUARY 2022
Changelog	05 NOVEMBER 2021 - INITIAL AUDIT 07 DECEMBER 2021 - SECOND REVIEW 12 JANUARY 2022 - THIRD REVIEW

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 BNPLPAY.io (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between October 18th, 2021 - November 5th, 2021.

Second review conducted on December 7th, 2021.

Third review conducted on January 12th, 2022.

Scope

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

Repository:
https://github.com/cf/bnpl-contracts-v5/
Commit:
c4cdd3615b135644b3e363365f8660b3ddcdc512
Technical Documentation: Yes, https://github.com/cf/bnpl-contracts-v5/blob/main/README.md
JS tests: Yes, https://github.com/cf/bnpl-contracts-v5/tree/main/test
Contracts:
Aave/FakeAave/FakeAaveUSDToken.sol
Aave/FakeAave/IFakeAaveToken.sol
Aave/FakeAave/FakeUSDToken.sol
Aave/FakeAave/FakeAaveLendingPool.sol
Aave/IAaveLendingPool.sol
BankNode/IBNPLBankNode.sol
BankNode/StakingPool/BNPLStakingPool.sol
BankNode/StakingPool/IBNPLNodeStakingPool.sol
BankNode/StakingPool/UserTokenLockup.sol
BankNode/BNPLBankNode.sol
ERC20/PoolTokenUpgradable.sol
ERC20/ITokenInitializableV1.sol
ERC20/IGenericBurnableFrom.sol
ERC20/IGenericMintableTo.sol
ERC20/ERC20BurnableUpgradeable.sol
ERC20/BNPLToken.sol
ERC20/IMintableToken.sol
ERC20/IMintableBurnableTokenUpgradeable.sol
ERC20/MintableDebugToken.sol
Management/BankNodeManager.sol
Management/BNPLKYCStore.sol
Management/IBankNodeManager.sol
ProtocolDeploy/BNPLProtocolConfig.sol
ProtocolDeploy/BNPLFirstDeploySetup.sol
ProtocolDeploy/IBNPLProtocolConfig.sol
Rewards/PlatformRewards/BankNodeLendingRewards.sol
Rewards/PlatformRewards/BankNodeRewardSystem.sol
SwapMarket/IBNPLSwapMarket.sol
SwapMarket/IBNPLPriceOracle.sol
SwapMarket/BNPLSwapMarketExample.sol
Utils/Math/PRBMathUD60x18.sol

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

Utils/Math/PRBMathCommon.sol
Utils/TransferHelper.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
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.

Insecure	Poor secured	Secured	Well-secured

You are here

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

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. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **2** medium and **3** low severity issues.

After the second review security engineers found **3** low severity issues.

After the third review security engineers found that **all** issues were fixed.

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

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. A mutable function that is available for everyone

   Contracts: BankNodeManager.sol

   Function: createBondedBankNode

   A malicious user could create an unlimited number of nodes. While it may not affect the contract itself, some methods in contract BankNodeLendingRewards that iterate over all nodes could become inoperable.

   Recommendation: Add access control for this method.

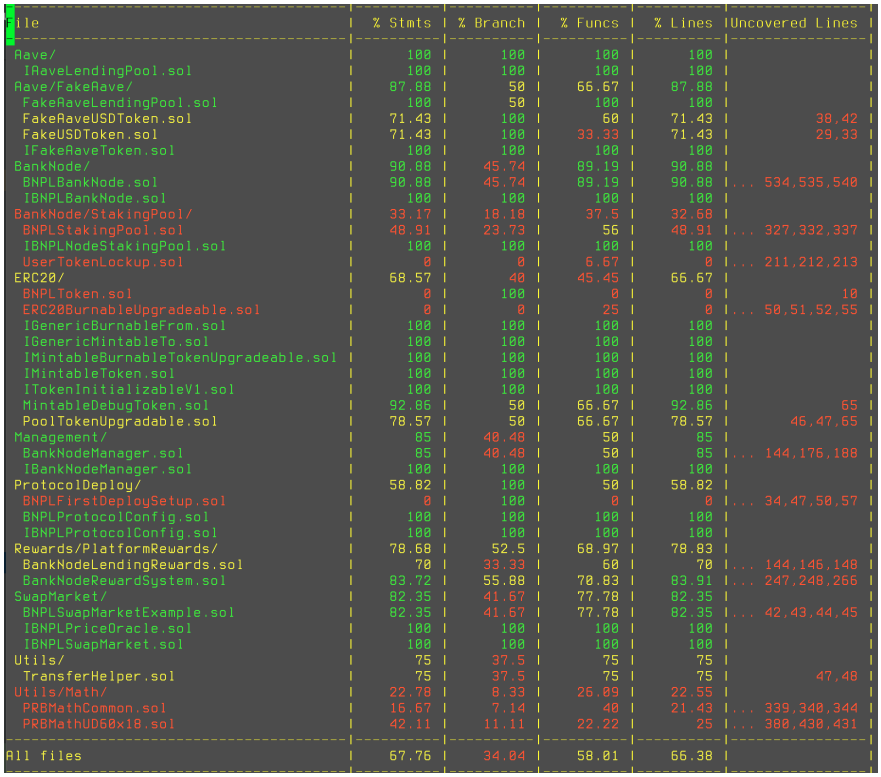
   Status: fixed

2. Too low test coverage

   Global test coverage is about 68% for statements and 34% for branches.

   The recommended coverage is a minimum of 95% for branches, while it should be definitely 100% for the main logic contracts.

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



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
Aave/	100	100	100	100	
IAaveLendingPool.sol	100	100	100	100	
Aave/FakeAave/	87.88	50	66.67	87.88	
FakeAaveLendingPool.sol	100	50	100	100	
FakeAaveUSDToken.sol	71.43	100	60	71.43	38,42
FakeUSDToken.sol	71.43	100	33.33	71.43	29,33
IFakeAaveToken.sol	100	100	100	100	
BankNode/	90.88	45.74	89.19	90.88	
BNPLBankNode.sol	90.88	45.74	89.19	90.88	... 534,535,540
IBNPLBankNode.sol	100	100	100	100	
BankNode/StakingPool/	33.17	18.18	37.5	32.68	
BNPLStakingPool.sol	48.91	23.73	56	48.91	... 327,332,337
IBNPLNodeStakingPool.sol	100	100	100	100	
UserTokenLockup.sol	0	0	6.67	0	... 211,212,213
ERC20/	68.57	40	45.45	66.67	
BNPLToken.sol	0	100	0	0	10
ERC20BurnableUpgradeable.sol	0	0	25	0	... 50,51,52,55
IGenericBurnableFrom.sol	100	100	100	100	
IGenericMintableTo.sol	100	100	100	100	
IMintableBurnableTokenUpgradeable.sol	100	100	100	100	
IMintableToken.sol	100	100	100	100	
ITokenInitializableV1.sol	100	100	100	100	
MintableDebugToken.sol	92.86	50	66.67	92.86	65
PoolTokenUpgradable.sol	78.57	50	66.67	78.57	46,47,65
Management/	85	40.48	50	85	
BankNodeManager.sol	85	40.48	50	85	... 144,176,188
IBankNodeManager.sol	100	100	100	100	
ProtocolDeploy/	58.82	100	50	58.82	
BNPLFirstDeploySetup.sol	0	100	0	0	... 34,47,50,57
BNPLProtocolConfig.sol	100	100	100	100	
IBNPLProtocolConfig.sol	100	100	100	100	
Rewards/PlatformRewards/	78.68	52.5	68.97	78.83	
BankNodeLendingRewards.sol	70	33.33	60	70	... 144,146,148
BankNodeRewardSystem.sol	83.72	55.88	70.83	83.91	... 247,248,266
SwapMarket/	82.35	41.67	77.78	82.35	
BNPLSwapMarketExample.sol	82.35	41.67	77.78	82.35	... 42,43,44,45
IBNPLPriceOracle.sol	100	100	100	100	
IBNPLSwapMarket.sol	100	100	100	100	
Utils/	75	37.5	75	75	
TransferHelper.sol	75	37.5	75	75	47,48
Utils/Math/	22.78	8.33	26.09	22.55	
PRBMathCommon.sol	16.67	7.14	40	21.43	... 339,340,344
PRBMathUD60x18.sol	42.11	11.11	22.22	25	... 380,430,431
All files	67.76	34.04	58.01	66.38	

Status: fixed

Low

1. Boolean equality

   Boolean constants can be used directly and do not need to be compared to true or false.

   Contracts: UserTokenLockup.sol

   Function: _createTokenLockup

   Recommendation: remove the equality to the boolean constant.

   Status: fixed

2. Tautology or contradiction

   Contracts: BNPLSwapMarketExample.sol

   Functions: withdrawToken, withdrawBNPL

   Recommendation: remove tautological comparisons **tokenBalances[token] != 0** and **bnplBalance != 0**

   Status: fixed

3. A public function that could be declared external.

   public functions that are never called by the contract should be declared external to save gas.

Contracts: FakeAaveLendingPool.sol, FakeAaveUSDToken.sol, FakeUSDToken.sol, UserTokenLockup.sol, BNPLStakingPool.sol, BNPLBankNode.sol, BankNodeLendingRewards.sol, MintableDebugToken.sol, PoolTokenUpgradeable.sol, BNPLKYCStore.sol, BankNodeManager.sol, BankNodeRewardSystem.sol, BNPLFirstDeploySetup.sol

Functions: addAssetPair, addLendableToken, addLiquidity, approveLoanRequest, bankNodeIdExists, calculateSlashAmount, createBondedBankNode, decodeUserBankNodeKey, decodeVaultValue, denyLoanRequest, deployBNPLWithImplementations, deposit, distributeBNPLTokensToBankNodes, distributeBNPLTokensToBankNodes2, donate, encodeVaultValue, getBNPLTokenDistribution, getBankNodeContract, getBankNodeLendableToken, getBankNodeStakingPoolContract, getBankNodeStakingPoolToken, getBankNodeToken, getNextTokenLockupForUser, initialize, internalAaveBurnFor, internalAaveMintFor, makeLoanPayment, mint, poolTokensCirculating, removeLiquidity, reportOverdueLoan, requestLoan, setLendableTokenStatus, setMinimumBankNodeBondedAmount, withdraw, getDomainPermissions, orKYCStatusWithProof

Recommendation: Use the external attribute for functions never called from the contract.

Status: fixed

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.

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

As a result of the audit, security engineers found **2** medium and **3** low severity issues.

After the second review security engineers found **3** low severity issues.

After the third review security engineers found that **all** issues were fixed.

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 the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free 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 the security of smart contracts.

Technical Disclaimer

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