{P()LY BL[]CK AUDITS}

Security Assessment

PUNK KNIGHTS

July 19th, 2022

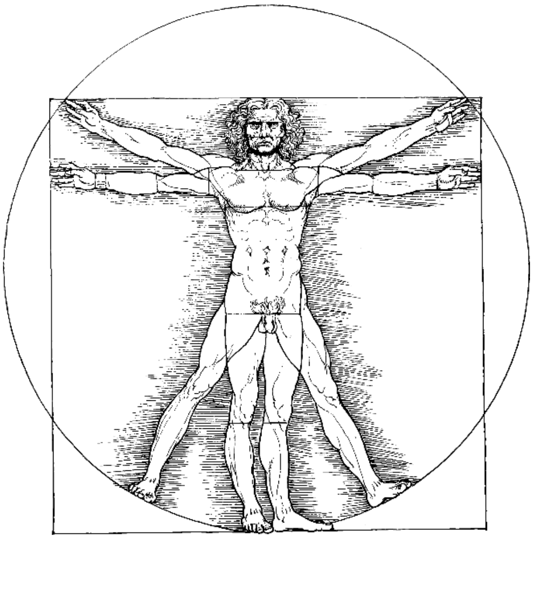

Table of Contents

.... 0

DISCLAIMER 2

Summary 3

Audit Details 4

Contract Overview 5

Contract Functions 6

Security Issues 9

Smart Contract Owner Functions 10

Conclusion 11

DISCLAIMER

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and P()LY BL[]CK AUDITS and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (P()LY BL[]CK AUDITS) owe no duty of care towards you or any other person, nor does P()LY BL[]CK AUDITS make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and P()LY BL[]CK AUDITS hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, P()LY BL[]CK AUDITS hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against P()LY BL[]CK AUDITS, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Summary

This report has been prepared for the Punk Knights smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

- The auditing process pays special attention to the following considerations:
- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the code base to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client. Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Audit Details

Audited Project:	PUNK KNIGHTS SMART CONTRACT
Audit Methodology:	Static Analysis, Manual Review, Testnet Deployment
Project Component:	PunkKnights.sol
Project Deployer:	0x9b5eb92e35c0f87beba27cee0465b1f6ec3af6cc
Project Contract:	0x9caAf7E8268512aD8aA8d89bCb0DC84E8Ec22710
Project Blockchain:	Binance Smart Chain
Project Website:	www.punkknights.com

Contract Overview

Project Name:	PUNK KNIGHTS SMART CONTRACT
Contract Address :	0x9caAf7E8268512aD8aA8d89bCb0DC84E8Ec22710
Total Supply:	1,125 NFKs
Symbol:	KNIGHT
Current Holders:	0
Current Transactions:	0
Top Holder 100 Dominance:	0%
Contract Deployer Address:	0x9b5eb92e35c0f87beba27cee0465b1f6ec3af6cc
Current Contract Owner:	0x9b5eb92e35c0f87beba27cee0465b1f6ec3af6cc
Language:	Solidity
Version:	0.8.15

Contract Functions

Contract ReentrancyGuard {
[Internal] <constructor>
}

Contract Context {
[Internal] _msgSender
[Internal] _msgData
}

Contract Ownable {
[Internal] <Constructor>
[Public] owner
[Public] renounceOwnership
[Public] transferOwnership
[private] _setOwner
}

Interface IBEP165 {
[External] supportsInterface
}

Interface BEP165 {
[Public] supportsInterface
}

Interface IBEP721 {
[External] balanceOf
[External] ownerOf
[External] safeTransferFrom
[External] transferFrom
[External] approve
[External] getApproved
[External] setApprovalForAll
[External] isApprovedForAll
}

Interface IBEP721Enumerable {
[External] totalSupply
[External] tokenOfOwnerByIndex
[External] tokenByIndex
}

Interface IBEP721Metadata {
[External] name
[External] symbol
[External] tokenURI
}

Interface IBEP721Receiver {
[External] onBEP721Received
}

Library Address {
[Internal] isContract
[Internal] sendValue
[Internal] functionCall
[Internal] functionCall
[Internal] functionCallWithValue
[Internal] functionCallWithValue
[Internal] functionStaticCall
[Internal] functionStaticCall
[Internal] functionDelegateCall
[Internal] functionDelegateCall
[Internal] verifyCallResult
}

Library Address {
[Internal] isContract
[Internal] sendValue
[Internal] functionCall
[Internal] functionCall
[Internal] functionCallWithValue
[Internal] functionCallWithValue
[Internal] functionStaticCall
[Internal] functionStaticCall
[Internal] functionDelegateCall
[Internal] functionDelegateCall
[Internal] verifyCallResult
}

Library Strings {
[Internal] toString
[Internal] toHexString
[Internal] toHexString
}

Contract WithLimitedSupply{
[Public] maxAvailableSupply
[Public] tokenCount
[Public] availableTokenCount
[Internal] nextToken
}

Contract RandomlyAssigned {
[Internal] nextToken
}

Contract BEP721A {
[Internal] _startTokenId
[Public] totalSupply
[Internal] _totalMinted
[Public] supportsInterface
[Public] balanceOf
[Internal] _numberMinted
[Internal] _numberBurned
[Internal] _getAux
[Internal] _setAux
[Internal] ownershipOf
[Public] ownerOf
[Public] name
[Public] symbol
[Public] tokenURI
[Internal] _baseURI
[Public] approve
[Public] getApproved
[Public] setApprovalForAll
[Public] isApprovedForAll
[Public] transferFrom
[Public] safeTransferFrom
[Internal] _exists
[Internal] _safeMint
[Internal] _safeMint
[Internal] _mint
[Private] _transfer
[Internal] _burn
[Private] _approve
[Private] _checkContractOnBEP721Received
[Internal] _beforeTokenTranfers
[Internal] _afterTokenTransfers

}

Contract IBEP2981 {
[External] royaltyInfo
}

Contract PunkKnights {
[Public] <Constructor>
[Public] baseURI
[Public] mintKnight
[Public] reserveKnights
[Public] walletOfOwner
[Public] tokenURI
[Public] revealUnbornKnights
[Public] setPublicPrice
[Public] setWhitelistPrice
[Public] setUnrevealedURI
[Public] setBaseURI
[Public] setBaseExtension
[Public] pauseNfkMint
[Public] addKnightToWhitelist
[External] addKnightsToWhitelist
[Public] removeWhitelistedKnight
[Public] withdraw
[External] withdrawTreasure
[External] setRoyaltyInfo
[External] royaltyInfo
}

Security Issues Checking Status

NO	Issue description	Status
01	Compiler errors.	Passed
02	Race conditions and Reentrancy. Cross-function race conditions.	Passed
03	Possible delays in data delivery.	Passed
04	Oracle calls.	Passed
05	Front running.	Passed
06	Timestamp dependence.	Passed
07	Integer Overflow and Underflow.	Passed
08	DoS with Revert.	Passed
09	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Economy model of the contract.	Passed
12	The impact of the exchange rate on the logic.	Passed
13	Private user data leaks.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Uninitialized storage pointers.	Passed
17	Arithmetic accuracy.	Passed
18	Design Logic.	Passed
19	Cross-function race conditions.	Passed
20	Fallback function security.	Passed

Security Issues

Extreme Issues

NO EXTREME ISSUES FOUND

High Issues

NO HIGH ISSUES FOUND

Medium Issues

NO MEDIUM ISSUES FOUND

Low Issues

NO LOW ISSUES FOUND

Informational Issues

NO INFORMATIONAL ISSUES FOUND

Smart Contract Owner Functions

In the contract PunkKnights.sol, the role Owner has the authority over the following function:

- withdrawTreasure
- addKnightToWhitelist
- addKnightsToWhitelist
- removeWhitelistedKnight
- mintKnight
- pauseNfkMint
- revealUnbornKnights
- setApprovalForAll
- setKnightsExtension
- setBornKnightsURI
- setMaxKnightsInWallet
- setMaxMintAmount
- setPublicPrice
- setRoyaltyInfo
- setUnbornKnightsURI
- setWhitelistPrice
- transferOwnership

Conclusion

The smart contract contains no security issues. Please read our disclaimer above!

Audited on:	07-19-2022
Audited by:	Polymathist, the auditor