Token

Smart Contract Audit Report
Prepared for Bold Point

BOLD POINT

Date Issued:	Apr 4, 2022
Project ID:	AUDIT2022024
Version:	v1.0
Confidentiality Level:	Public

inspeX
CYBERSECURITY PROFESSIONAL SERVICE

Report Information

Project ID	AUDIT2022024
Version	v1.0
Client	Bold Point
Project	Token
Auditor(s)	Peeraphut Punsuwan Sorawish Laovakul
Author(s)	Peeraphut Punsuwan
Reviewer	Natsasit Jirathammanuwat
Confidentiality Level	Public

Version History

Version	Date	Description	Author(s)
1.0	Apr 4, 2022	Full report	Peeraphut Punsuwan

Contact Information

Company	Inspex
Phone	(+66) 90 888 7186
Telegram	t.me/inspexco
Email	audit@inspex.co

Table of Contents

1. Executive Summary

As requested by Bold Point, Inspex team conducted an audit to verify the security posture of the Token smart contracts on Mar 31, 2022. During the audit, Inspex team examined all smart contracts and the overall operation within the scope to understand the overview of Token smart contracts. Static code analysis, dynamic analysis, and manual review were done in conjunction to identify smart contract vulnerabilities together with technical & business logic flaws that may be exposed to the potential risk of the platform and the ecosystem. Practical recommendations are provided according to each vulnerability found and should be followed to remediate the issue.

1.1. Audit Result

In the initial audit, Inspex found 2 very low and 1 info-severity issues. With the project team's prompt response, 1 very low and 1 info-severity issues were resolved in the reassessment, while 1 very low-severity issue was acknowledged by the team. Therefore, Inspex trusts that Token smart contracts have sufficient protections to be safe for public use. However, in the long run, Inspex suggests resolving all issues found in this report.

1.2. Disclaimer

This security audit is not produced to supplant any other type of assessment and does not guarantee the discovery of all security vulnerabilities within the scope of the assessment. However, we warrant that this audit is conducted with goodwill, professional approach, and competence. Since an assessment from one single party cannot be confirmed to cover all possible issues within the smart contract(s), Inspex suggests conducting multiple independent assessments to minimize the risks. Lastly, nothing contained in this audit report should be considered as investment advice.

2. Project Overview

2.1. Project Introduction

BoldPoint is a fully decentralized NFT Marketplace designed for traders and NFT. The protocol participants can earn cryptocurrency through mining, farming, and trading in the game.

$BPT token is a standard BEP20 token with pausable and burnable mechanisms. The token is minted to the owner with a limited supply of 1 billion $BPT.

Scope Information:

Project Name	Token
Website	https://bp.yay.games
Smart Contract Type	Ethereum Smart Contract
Chain	Avalanche
Programming Language	Solidity
Category	Token

Audit Information:

Audit Method	Whitebox
Audit Date	Mar 31, 2022
Reassessment Date	Apr 4, 2022

The audit method can be categorized into two types depending on the assessment targets provided:

1. **Whitebox**: The complete source code of the smart contracts are provided for the assessment.
2. **Blackbox**: Only the bytecodes of the smart contracts are provided for the assessment.

2.2. Scope

The following smart contracts were audited and reassessed by Inspex in detail:

Initial Audit: (Commit: fdad0b795752f6c947a774e867972c507f0e6ac7)

Contract	Location (URL)
BoldPointToken	https://github.com/YAY-Games/bpt-token-contract/blob/fdad0b7957/contracts/BoldPointToken.sol

Reassessment: (Commit: 7f2c3a1589500a172fdabb8b3414f42d16a9457e)

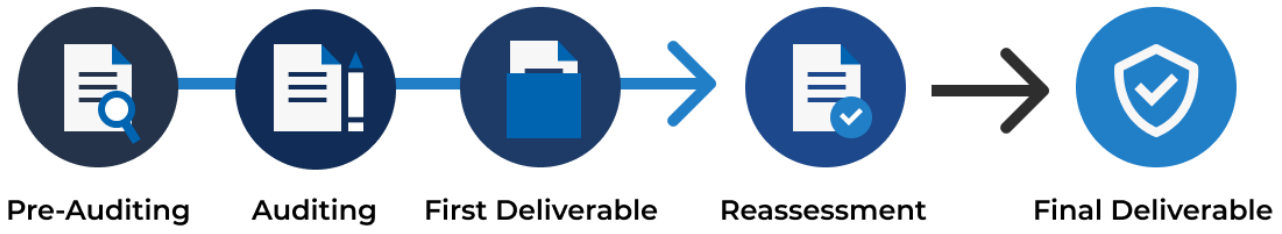
Contract	Location (URL)
BoldPointToken	https://github.com/YAY-Games/bpt-token-contract/blob/7f2c3a1589/contracts/BoldPointToken.sol

The assessment scope covers only the in-scope smart contracts and the smart contracts that they inherit from.

3. Methodology

Inspex conducts the following procedure to enhance the security level of our clients' smart contracts:

1. **Pre-Auditing**: Getting to understand the overall operations of the related smart contracts, checking for readiness, and preparing for the auditing
2. **Auditing**: Inspecting the smart contracts using automated analysis tools and manual analysis by a team of professionals
3. **First Deliverable and Consulting**: Delivering a preliminary report on the findings with suggestions on how to remediate those issues and providing consultation
4. **Reassessment**: Verifying the status of the issues and whether there are any other complications in the fixes applied
5. **Final Deliverable**: Providing a full report with the detailed status of each issue



3.1. Test Categories

Inspex smart contract auditing methodology consists of both automated testing with scanning tools and manual testing by experienced testers. We have categorized the tests into 3 categories as follows:

1. **General Smart Contract Vulnerability (General)** - Smart contracts are analyzed automatically using static code analysis tools for general smart contract coding bugs, which are then verified manually to remove all false positives generated.
2. **Advanced Smart Contract Vulnerability (Advanced)** - The workflow, logic, and the actual behavior of the smart contracts are manually analyzed in-depth to determine any flaws that can cause technical or business damage to the smart contracts or the users of the smart contracts.
3. **Smart Contract Best Practice (Best Practice)** - The code of smart contracts is then analyzed from the development perspective, providing suggestions to improve the overall code quality using standardized best practices.

3.2. Audit Items

The following audit items were checked during the auditing activity.

General
Smart Contract with Unpublished Source Code
Reentrancy Attack
Integer Overflows and Underflows
Unchecked Return Values for Low-Level Calls
Bad Randomness
Transaction Ordering Dependence
Time Manipulation
Short Address Attack
Outdated Compiler Version
Use of Known Vulnerable Component
Deprecated Solidity Features
Use of Deprecated Component
Loop with High Gas Consumption
Unauthorized Self-destruct
Redundant Fallback Function
Insufficient Logging for Privileged Functions
Invoking of Unreliable Smart Contract
Use of Upgradable Contract Design
Centralized Control of State Variable
Advanced
Business Logic Flaw
Ownership Takeover
Broken Access Control

Broken Authentication
Improper Kill-Switch Mechanism
Improper Front-end Integration
Insecure Smart Contract Initiation
Denial of Service
Improper Oracle Usage
Memory Corruption
Best Practice
Use of Variadic Byte Array
Implicit Compiler Version
Implicit Visibility Level
Implicit Type Inference
Function Declaration Inconsistency
Token API Violation
Best Practices Violation

3.3. Risk Rating

OWASP Risk Rating Methodology (https://owasp.org/www-community/OWASP_Risk_Rating_Methodology) is used to determine the severity of each issue with the following criteria:

- **Likelihood**: a measure of how likely this vulnerability is to be uncovered and exploited by an attacker
- **Impact**: a measure of the damage caused by a successful attack

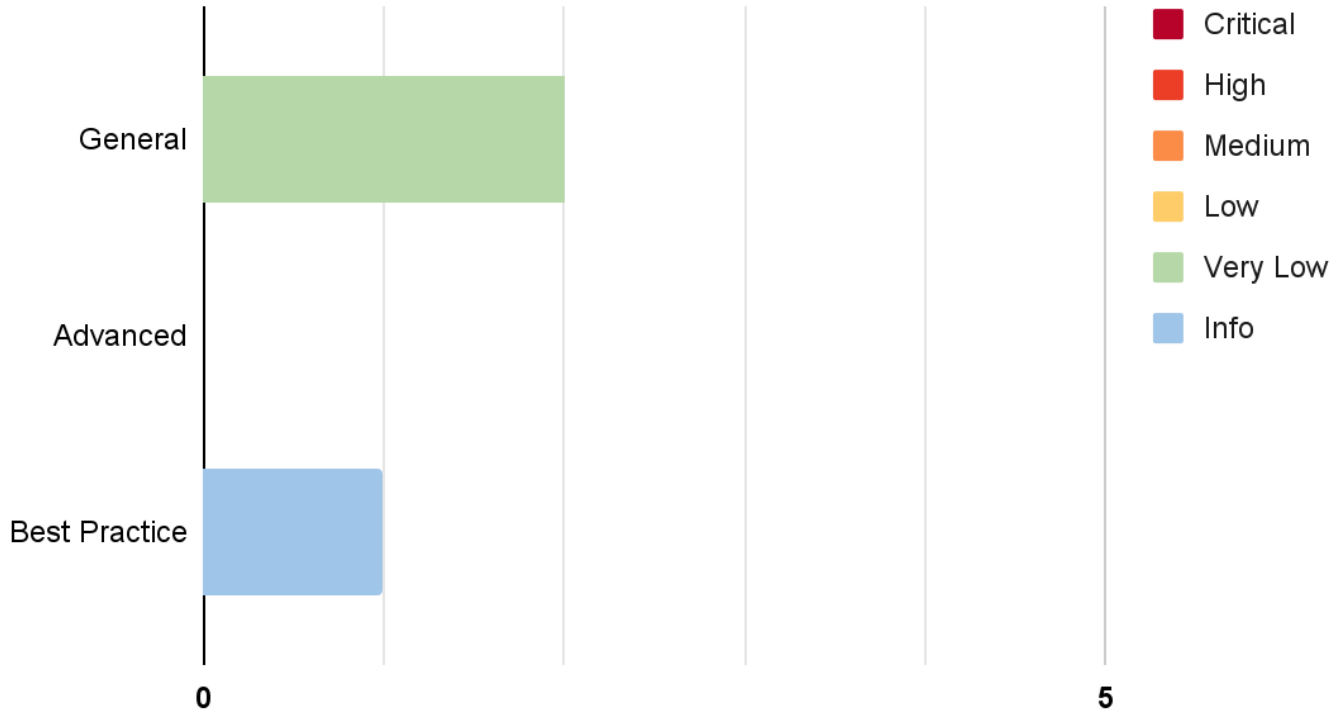
Both likelihood and impact can be categorized into three levels: **Low**, **Medium**, and **High**.

Severity is the overall risk of the issue. It can be categorized into five levels: **Very Low**, **Low**, **Medium**, **High**, and **Critical**. It is calculated from the combination of likelihood and impact factors using the matrix below. The severity of findings with no likelihood or impact would be categorized as **Info**.

Impact \ Likelihood	Low	Medium	High
Low	Very Low	Low	Medium
Medium	Low	Medium	High
High	Medium	High	Critical

4. Summary of Findings

From the assessments, Inspex has found 3 issues in two categories. The following chart shows the number of the issues categorized into three categories: **General**, **Advanced**, and **Best Practice**.



The statuses of the issues are defined as follows:

Status	Description
Resolved	The issue has been resolved and has no further complications.
Resolved *	The issue has been resolved with mitigations and clarifications. For the clarification or mitigation detail, please refer to Chapter 5.
Acknowledged	The issue's risk has been acknowledged and accepted.
No Security Impact	The best practice recommendation has been acknowledged.

The information and status of each issue can be found in the following table:

ID	Title	Category	Severity	Status
IDX-001	Untransferable Tokens via Contract Pausing	General	**Very Low**	**Acknowledged**
IDX-002	Outdated Compiler Version	General	**Very Low**	**Resolved**
IDX-003	Improper Function Visibility	Best Practice	**Info**	**Resolved**

* The mitigations or clarifications by Bold Point can be found in Chapter 5.

5. Detailed Findings Information

5.1. Untransferable Tokens via Contract Pausing

ID	IDX-001
Target	BoldPointToken
Category	General Smart Contract Vulnerability
CWE	CWE-284: Improper Access Control
Risk	**Severity:** Very Low **Impact:** Low The owner can control the token transfer flows, resulting in unfairness for the users. However, the owner will be unable to transfer the token when the `pause()` function is called. **Likelihood:** Low There is very low motivation to use the `pause()` function to gain profit from the users. Moreover, while the contract is pausing no one can transfer the $BPT.
Status	**Acknowledged** The BoldPoint team has clarified that the pausable mechanism is still required because of the community asset's safety and the overall security of the project. Moreover, they can be used in emergency cases of problems and abuses at the early stages of the project. However, the ownership can be renounced in the future. At the time of the reassessment, the contracts are not deployed yet, so the ownership is not confirmed to be renounced. For the platform users, please verify that the ownership has been renounced before using this platform.

5.1.1. Description

The $BPT token is implemented as an `ERC20Pausable` token. It includes the pausable mechanism to prevent all token transfers when the contract owner decides to pause it.

BoldPointToken.sol

```
20 contract BoldPointToken is ERC20, ERC20Burnable, ERC20Permit, ERC20Pausable,
   Ownable {
21     constructor() ERC20("Bold Point Token", "BPT") ERC20Permit("Bold Point
   Token") {
22         _mint(msg.sender, 1_000_000_000 * 10 ** decimals());
23     }
24
25     function pause() external onlyOwner {
26         _pause();
```

```
27      }
28
29      function unpause() external onlyOwner {
30          _unpause();
31      }
32
33      function _beforeTokenTransfer(address from, address to, uint256 amount)
34          internal
35          override(ERC20, ERC20Pausable)
36      {
37          super._beforeTokenTransfer(from, to, amount);
38      }
39  }
```

5.1.2. Remediation

Inspex suggests removing the pausable mechanism from the contract to ensure that the tokens can be transferred freely.

However, if the pausable mechanism is needed to prevent any unexpected issue on the platform, it is recommended to implement this feature to the other contract logic instead since the users' tokens should be transferred freely at any cost.

5.2. Outdated Compiler Version

ID	IDX-002
Target	BoldPointToken
Category	General Smart Contract Vulnerability
CWE	CWE-1104: Use of Unmaintained Third Party Components
Risk	**Severity:** Very Low **Impact:** Low From the list of known Solidity bugs, direct impact cannot be caused from those bugs themselves. **Likelihood:** Low From the list of known Solidity bugs, it is very unlikely that those bugs would affect these smart contracts.
Status	Resolved The BoldPoint team has resolved this by upgrading the Solidity compiler version to v0.8.13 which currently does not have any known vulnerability. The issue has resolved in commit: `7f2c3a1589500a172fdabb8b3414f42d16a9457e`

5.2.1. Description

The Solidity compiler versions specified in the smart contracts were outdated. These versions have publicly known inherent bugs (https://docs.soliditylang.org/en/v0.8.13/bugs.html) that may potentially be used to cause damage to the smart contracts or the users of the smart contracts.

BoldPointToken.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity 0.8.10;
```

5.2.2. Remediation

Inspex suggests upgrading the Solidity compiler to the latest stable version (https://github.com/ethereum/solidity/releases).

During the audit activity, the latest stable version of Solidity compiler in major v0.8 is v0.8.13.

BoldPointToken.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity 0.8.13;
```

5.3. Improper Function Visibility

ID	IDX-003
Target	BoldPointToken
Category	Smart Contract Best Practice
CWE	CWE-710: Improper Adherence to Coding Standards
Risk	**Severity:** Info **Impact: None** **Likelihood: None**
Status	Resolved The BoldPoint team has resolved this issue by changing all affected functions' visibility to external. The issue has resolved in commit: `7f2c3a1589500a172fdabb8b3414f42d16a9457e`

5.3.1. Description

Functions with public visibility copy calldata to memory when being executed, while external functions can read directly from calldata. Memory allocation uses more resources (gas) than reading directly from calldata.

BoldPointToken.sol

```
25 function pause() public onlyOwner {
26     _pause();
27 }
```

The following table contains all functions that have public visibility and are never called from any internal function.

Target	Function
BoldPointToken (L:25)	pause()
BoldPointToken (L:29)	unpause()

5.3.2. Remediation

Inspex suggests changing all functions' visibility to external if they are not called from any internal function as shown in the following example:

BoldPointToken.sol

```
25 function pause() external onlyOwner {
26     _pause();
```

```
27 }
```

6. Appendix

6.1. About Inspex

Inspex is formed by a team of cybersecurity experts highly experienced in various fields of cybersecurity. We provide blockchain and smart contract professional services at the highest quality to enhance the security of our clients and the overall blockchain ecosystem.

Follow Us On:

Website	https://inspex.co
Twitter	@InspexCo
Facebook	https://www.facebook.com/InspexCo
Telegram	@inspex_announcement

inspeX

CYBERSECURITY PROFESSIONAL SERVICE