Launchpad & Staking

Smart Contract Audit Report
Prepared for Ancient8

Date Issued:	Mar 23, 2022
Project ID:	AUDIT2022016
Version:	v1.0
Confidentiality Level:	Public

inspex

CYBERSECURITY PROFESSIONAL SERVICE

Report Information

Project ID	AUDIT2022016
Version	v1.0
Client	Ancient8
Project	Launchpad & Staking
Auditor(s)	Patipon Suwanbol Peeraphut Punsuwan Puttimet Thammasaeng
Author(s)	Patipon Suwanbol
Reviewer	Weerawat Pawanawiwat
Confidentiality Level	Public

Version History

Version	Date	Description	Author(s)
1.0	Mar 23, 2022	Full report	Patipon Suwanbol

Contact Information

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

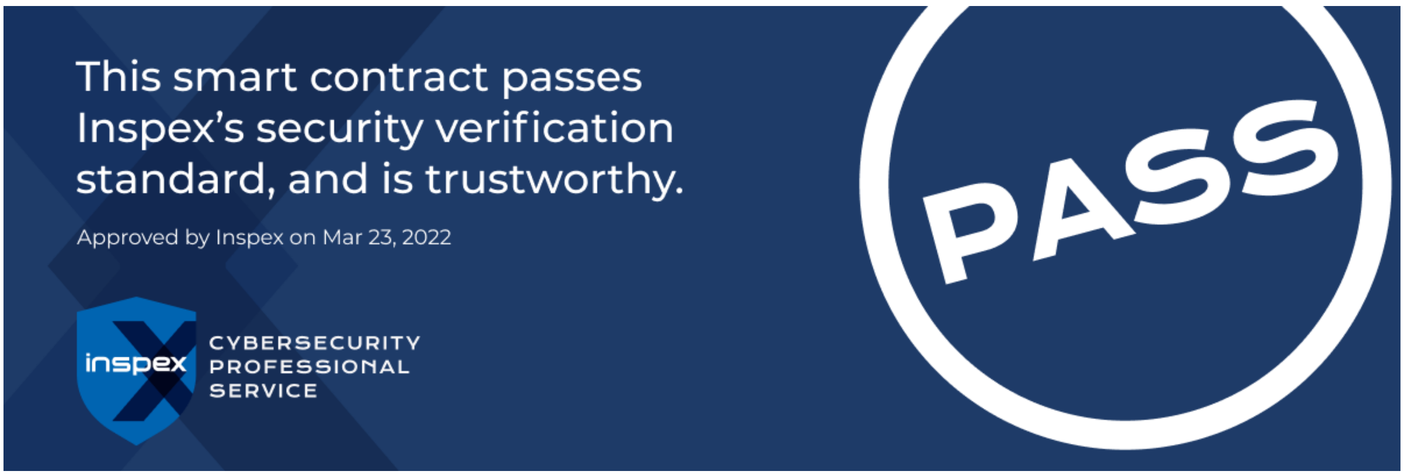
Table of Contents

1. Executive Summary

As requested by Ancient8, Inspex team conducted an audit to verify the security posture of the Launchpad & Staking smart contracts between Mar 8, 2022 and Mar 11, 2022. During the audit, Inspex team examined all smart contracts and the overall operation within the scope to understand the overview of Launchpad & Staking 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 1 high, 2 low, and 1 very low-severity issues. With the project team's prompt response, 1 high, 1 low and 1 very-low-severity issues were resolved or mitigated in the reassessment, while 1 low-severity issue was acknowledged by the team. Therefore, Inspex trusts that Launchpad & Staking smart contracts have sufficient protections to be safe for public use. However, as the source code is not publicly available, the bytecode of the smart contracts deployed should be compared with the bytecode of the smart contracts audited before interacting with them. In the long run, Inspex suggests resolving all issues found in this report.


This smart contract passes Inspex's security verification standard, and is trustworthy.

Approved by Inspex on Mar 23, 2022

inspex
CYBERSECURITY PROFESSIONAL SERVICE
PASS

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

Ancient8 is building a DAO that develops a community and software platform to enable everyone to play and build the Metaverse while earning rewards. As Vietnam's largest blockchain gaming guild, Ancient8 has helped tens of thousands of blockchain gamers and enthusiasts by providing scholarship and educational opportunities, community, and blockchain and software products. Ancient8's vision is to democratize social and financial access in the Metaverse, and is on a mission to reach, educate, and empower the next 100 million Metaverse citizens through the blockchain. Ancient8 is backed by leading investors including Dragonfly Capital, Pantera Capital, Hashed, Mechanism Capital, Coinbase Ventures, Alameda Research, Animoca Brands, among others.

Launchpad and Staking are programs that allow users of the platform to purchase interesting GameFi tokens that have been verified by Ancient8. This will allow the potential gem of the GameFi project to raise funds and contribute their part to improving the GameFi experience for the betterment of the GameFi ecosystem.

Scope Information:

Project Name	Launchpad & Staking
Website	https://ancient8.gg
Smart Contract Type	Solana Program
Chain	Solana
Programming Language	Rust
Category	Launchpad

Audit Information:

Audit Method	Whitebox
Audit Date	Mar 8, 2022 - Mar 11, 2022
Reassessment Date	Mar 21, 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 smart contracts with the following bytecodes were audited and reassessed by Inspex in detail:

Initial Audit:

Contract	Bytecode SHA256 Hash
Stake	88a80e4f168b6185544e06d9b8721ccdf4528623a6dc8b203e787be98dddf6b7
Launchpad	3e09ab049f7f3af0ce6b8c068a9daf4fbeb1aa2c1424127091efb1bd09193938

Reassessment Audit:

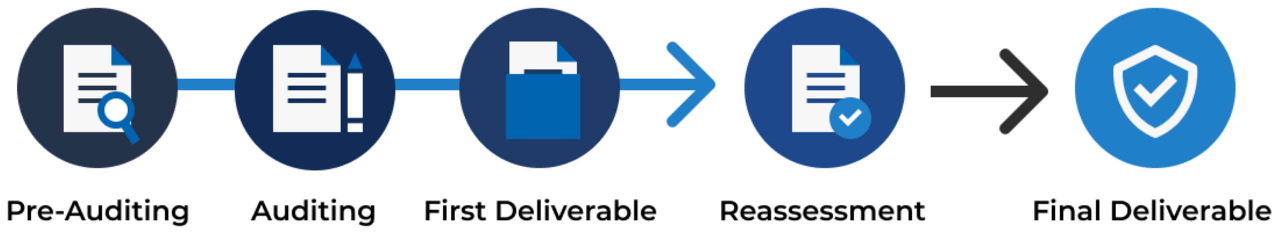
Contract	Bytecode SHA256 Hash
Stake	bfaff774d229c5f7d1c348235f1659fca5441dcac7fd04f71467e3c66aa46d67
Launchpad	c490d0bd992dd49cba2eabdfb407be547d21e08b5da2a33980416c933b3c4a1c

As the Ancient8 team has decided not to publish the source code to protect their intellectual property, the users should compare the bytecode hashes with the smart contracts deployed before interacting with them to make sure that they are the same with the contracts audited.

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
Program with Unpublished Source Code
Integer Overflows and Underflows
Bad Randomness
Use of Known Vulnerable Component
Use of Deprecated Component
Solana Account Confusions
Missing Rent Exemption Checking
Advanced
Business Logic Flaw
Ownership Takeover
Broken Access Control
Broken Authentication
Denial of Service
Improper Oracle Usage
Memory Corruption
Best Practice
Implicit Type Inference
Function Declaration Inconsistency
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

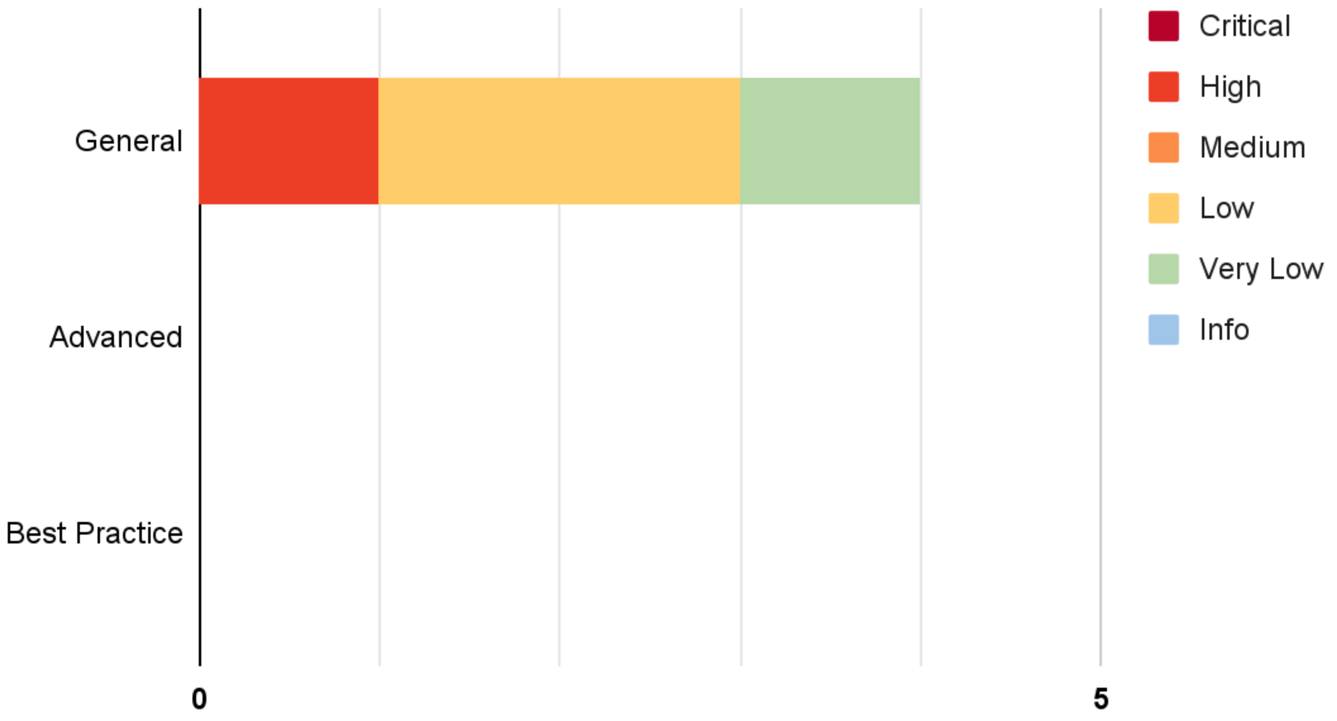
Both likelihood and impact can be categorized into three levels: **Low**, **Medium**, and **High**.

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**.

Likelihood Impact	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 4 issues in three 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	Upgradability of Solana Program	General	High	Resolved *
IDX-002	Centralized Control of State Variable	General	Low	Resolved *
IDX-003	Smart Contract with Unpublished Source Code	General	Low	Acknowledged
IDX-004	Insufficient Logging for Privileged Functions	General	Very Low	Resolved

* The mitigations or clarifications by Ancient8 can be found in Chapter 5.

5. Detailed Findings Information

5.1. Upgradability of Solana Program

ID	IDX-001
Target	stake launchpad
Category	General Smart Contract Vulnerability
CWE	CWE-284: Improper Access Control
Risk	**Severity:** High **Impact:** High The logic of the affected programs can be arbitrarily changed. This allows the upgrade authority to change the logic of the program in favor of the platform, e.g., transferring the users' funds to the platform owner's account. **Likelihood:** Medium Only the program upgrade authority can redeploy the program to the same program address; however, there is no restriction to prevent the authority from inserting malicious logic.
Status	Resolved * Ancient8 team has confirmed that they will apply the Goki protocol as the multisig account with timelock after it is fully released to the public. This will be controlled by multiple trusted parties to ensure the transparency of the platform.

5.1.1. Description

Programs on Solana can be deployed through the upgradable BPF loader to make them upgradable, allowing the program's upgrade authority to redeploy the program with the new logic, bug fixes, or upgrades to the same program address.

However, there is no restriction on how and when the program will be upgraded. This opens up an attack surface on the program, allowing the upgrade authority to redeploy the program with malicious logic and gain unfair benefits from the users. For example, transferring funds out from the users' accounts.

5.1.2. Remediation

Inspex suggests deploying the program as an immutable program to prevent the program logic from being modified.

However, if upgradability is needed, Inspex suggests mitigating this issue by the following options:

- Using a multisig account controlled by multiple trusted parties as the upgrade authority
- Implementing a community-run governance to control the redeployment of the program

5.2. Centralized Control of State Variable

ID	IDX-002
Target	stake
Category	General Smart Contract Vulnerability
CWE	CWE-284: Improper Access Control
Risk	**Severity:** Low **Impact:** Low The controlling authorities can change the critical state variables, thereby causing unfairness to the other users. However, the impact is limited to the user's opportunities for joining the launchpad pool and withdrawing the capital early penalty, which do not affect the monetary loss for those who have already performed that action. **Likelihood:** Medium There is nothing to restrict the changes from being done by the owner. However, only some owner roles can call these functions to change the states.
Status	Resolved * Ancient8 team has confirmed that they will apply the Goki protocol as the multisig account with timelock after it is fully released to the public. This will be controlled by multiple trusted parties to ensure that the critical changes will be publicly known first before it has the effect.

5.2.1. Description

Critical state variables can be updated any time by the controlling authorities. Changes in these variables can cause impacts to the users, so the users should accept or be notified before these changes are effective.

However, there is currently no constraint to prevent the authorities from modifying these variables without notifying the users.

The controllable privileged state update functions are as follows:

File	Program	Function
programs/stake/src/action/update_tiers.rs (L:23)	stake	process()
programs/launchpad/src/action/update_ticket_pool.rs (L:25)	launchpad	process()

5.2.2. Remediation

In the ideal case, the critical state variables should not be modifiable to keep the integrity of the program. However, if modifications are needed, Inspex suggests limiting the use of these functions by the following options:

- Using a multisig account controlled by multiple trusted parties to ensure that the changes of critical states are well prepared
- Implementing a community-run governance to control the use of these functions

5.3. Smart Contract with Unpublished Source Code

ID	IDX-003
Target	stake launchpad
Category	General Smart Contract Vulnerability
CWE	CWE-1006: Bad Coding Practices
Risk	**Severity:** Low **Impact:** Medium The logic of the smart contract may not align with the user's understanding, causing undesired actions to be taken when the user interacts with the smart contract. **Likelihood:** Low The possibility for the users to misunderstand the functionalities of the contract is not very high with the help of the documentation and user interface.
Status	**Acknowledged** Ancient8 team has acknowledged this issue and decided not to publish the source code because the team wants to protect their intellectual property.

5.3.1. Description

The smart contract source code is not publicly published, so the users will not be able to easily verify the correctness of the functionalities and the logic of the smart contract by themselves. Therefore, it is possible that the user's understanding of the smart contract does not align with the actual implementation, leading to undesired actions on interacting with the smart contract.

5.3.2. Recommendation

Inspex suggests publishing the contract source code through a public code repository or verifying the smart contract source code on the blockchain explorer so that the users can easily read and understand the logic of the smart contract by themselves.

5.4. Insufficient Logging for Privileged Functions

ID	IDX-004
Target	stake launchpad
Category	General Smart Contract Vulnerability
CWE	CWE-778: Insufficient Logging
Risk	**Severity:** Very Low **Impact:** Low Privileged functions' executions cannot be monitored easily by the users. **Likelihood:** Low It is unlikely that the execution of the privileged functions will be a malicious action.
Status	Resolved Anceint8 team has resolved this issue by emitting events as suggested.

5.4.1. Description

Privileged functions that are executable by the controlling parties are not logged properly by emitting events. Without events, it is not easy for the public to monitor the execution of those privileged functions, allowing the controlling parties to perform actions that cause big impacts on the platform.

The privileged functions with insufficient logging are as follows:

File	Program	Function
programs/stake/src/action/update_tiers.rs (L:23)	stake	process()
programs/launchpad/src/action/update_ticket_pool.rs (L:25)	launchpad	process()

5.4.2. Remediation

Inspex suggests emitting events for the execution of privileged functions, for example:

programs/stake/src/event.rs

```
#[event]
pub struct UpdateTierEvent {
    #[index]
    pub sender: Pubkey,
    pub penalty_withdraw: u32,
    pub min_days_stake: i64,
    pub stake_fee_rate: u32
}
```

programs/stake/src/action/update_tiers.rs

```
pub fn process(&mut self, params: UpdateTiersParams) -> ProgramResult {
    invariant!(params.penalty_withdraw <= 100, ErrorCode::InvalidData);
    invariant!(params.stake_fee_rate <= 100, ErrorCode::InvalidData);

    self.tiers_account.penalty_withdraw = params.penalty_withdraw;
    self.tiers_account.min_days_stake = params.min_days_stake;
    self.tiers_account.stake_fee_rate = params.stake_fee_rate;
    stake_emit!(UpdateTierEvent {
            sender: self.sender.key(),
            penalty_withdraw: params.penalty_withdraw,
            min_days_stake: params.min_days_stake,
            stake_fee_rate: params.stake_fee_rate
    });
    Ok(())
}
```

6. Appendix

6.1. About Inspex

inspex CYBERSECURITY PROFESSIONAL SERVICE

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