Token

Smart Contract Audit Report
Prepared for Aniverse

Date Issued:	Jun 30, 2022
Project ID:	AUDIT2022042
Version:	v1.0
Confidentiality Level:	Public

inspeX
CYBERSECURITY PROFESSIONAL SERVICE

Report Information

Project ID	AUDIT2022042
Version	v1.0
Client	Aniverse
Project	Token
Auditor(s)	Natsasit Jirathammanuwat
Author(s)	Natsasit Jirathammanuwat
Reviewer	Peeraphut Punsuwan
Confidentiality Level	Public

Version History

Version	Date	Description	Author(s)
1.0	Jun 30, 2022	Full report	Natsasit Jirathammanuwat

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 Aniverse, Inspex team conducted an audit to verify the security posture of the Token smart contracts on Jun 29, 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 1 very low, and 1 info-severity issues. With the project team's prompt response in resolving the issues found by Inspex, all issues were resolved in the reassessment. Therefore, Inspex trusts that Token smart contracts have high-level protections in place to be safe from most attacks.

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

Aniverse is a virtual world in which many people could have the opportunity to create their dreams in virtual reality, bringing you new experiences for living life, building a new business journey and determining your future that you have always dreamed of, and breaking the limitation in the real world.

ANIV token is the ERC-20 standard token with the burnable function. There are a total of 2,000,000,000 ANIV tokens with limited supply. The allocation of tokens is divided into the public/private sale, founder & team, partners, and marketing.

Scope Information:

Project Name	Token
Website	https://aniv.io
Smart Contract Type	Ethereum Smart Contract
Chain	Polygon
Programming Language	Solidity
Category	Token

Audit Information:

Audit Method	Whitebox
Audit Date	Jun 29, 2022
Reassessment Date	Jun 30, 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: 3cf5acea52bf4d67bcb4ea5278bff453108981e3)

Contract	Location (URL)
ANIV20	https://github.com/CREATIVE-DIGITAL-LIVING-CO-LTD/SC_ERC20/blob/3cf5acea52/contracts/ANIV20.sol

Reassessment: (Commit: c35474969003fcfc678706faccb2d95480ea8a0a)

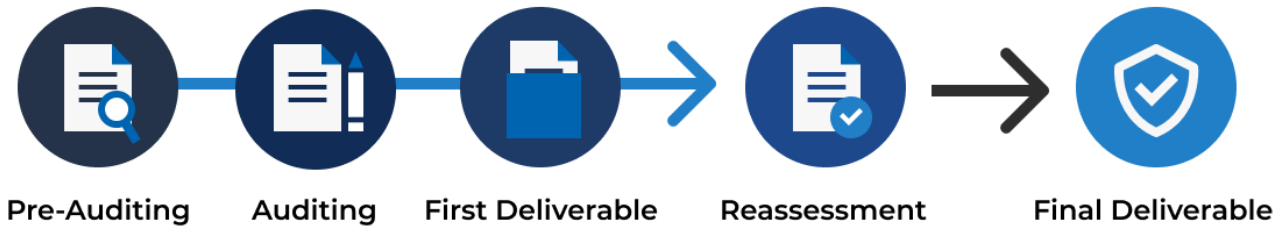
Contract	Location (URL)
ANIV20	https://github.com/CREATIVE-DIGITAL-LIVING-CO-LTD/SC_ERC20/blob/c354749690/contracts/ANIV20.sol

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 testing items checked are based on our Smart Contract Security Testing Guide (SCSTG) v1.0 (https://github.com/InspexCo/SCSTG/releases/download/v1.0/SCSTG_v1.0.pdf) which covers most prevalent risks in smart contracts. The latest version of the document can also be found at https://inspex.gitbook.io/testing-guide/.

The following audit items were checked during the auditing activity:

Testing Category	Testing Items
1. Architecture and Design	1.1. Proper measures should be used to control the modifications of smart contract logic 1.2. The latest stable compiler version should be used 1.3. The circuit breaker mechanism should not prevent users from withdrawing their funds 1.4. The smart contract source code should be publicly available 1.5. State variables should not be unfairly controlled by privileged accounts 1.6. Least privilege principle should be used for the rights of each role
2. Access Control	2.1. Contract self-destruct should not be done by unauthorized actors 2.2. Contract ownership should not be modifiable by unauthorized actors 2.3. Access control should be defined and enforced for each actor roles 2.4. Authentication measures must be able to correctly identify the user 2.5. Smart contract initialization should be done only once by an authorized party 2.6. tx.origin should not be used for authorization
3. Error Handling and Logging	3.1. Function return values should be checked to handle different results 3.2. Privileged functions or modifications of critical states should be logged 3.3. Modifier should not skip function execution without reverting
4. Business Logic	4.1. The business logic implementation should correspond to the business design 4.2. Measures should be implemented to prevent undesired effects from the ordering of transactions 4.3. msg.value should not be used in loop iteration
5. Blockchain Data	5.1. Result from random value generation should not be predictable 5.2. Spot price should not be used as a data source for price oracles 5.3. Timestamp should not be used to execute critical functions 5.4. Plain sensitive data should not be stored on-chain 5.5. Modification of array state should not be done by value 5.6. State variable should not be used without being initialized

Testing Category	Testing Items
6. External Components	6.1. Unknown external components should not be invoked 6.2. Funds should not be approved or transferred to unknown accounts 6.3. Reentrant calling should not negatively affect the contract states 6.4. Vulnerable or outdated components should not be used in the smart contract 6.5. Deprecated components that have no longer been supported should not be used in the smart contract 6.6. Delegatecall should not be used on untrusted contracts
7. Arithmetic	7.1. Values should be checked before performing arithmetic operations to prevent overflows and underflows 7.2. Explicit conversion of types should be checked to prevent unexpected results 7.3. Integer division should not be done before multiplication to prevent loss of precision
8. Denial of Services	8.1. State changing functions that loop over unbounded data structures should not be used 8.2. Unexpected revert should not make the whole smart contract unusable 8.3. Strict equalities should not cause the function to be unusable
9. Best Practices	9.1. State and function visibility should be explicitly labeled 9.2. Token implementation should comply with the standard specification 9.3. Floating pragma version should not be used 9.4. Builtin symbols should not be shadowed 9.5. Functions that are never called internally should not have public visibility 9.6. Assert statement should not be used for validating common conditions

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**.

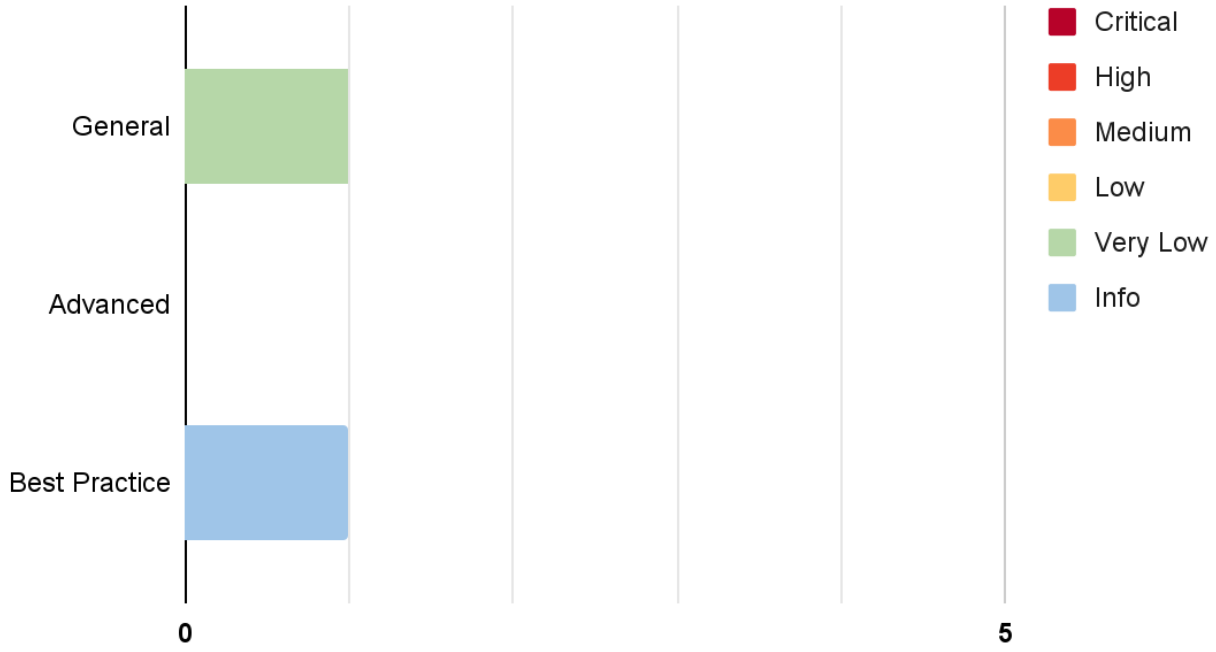
Severity is the overall risk of the issue. It can be categorized into five levels: **Very Low**, **Low**, **Medium**, **High**, and **Critical**. It is calculated from the combination of likelihood and impact factors using the matrix below. The severity of findings with no likelihood or impact would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

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

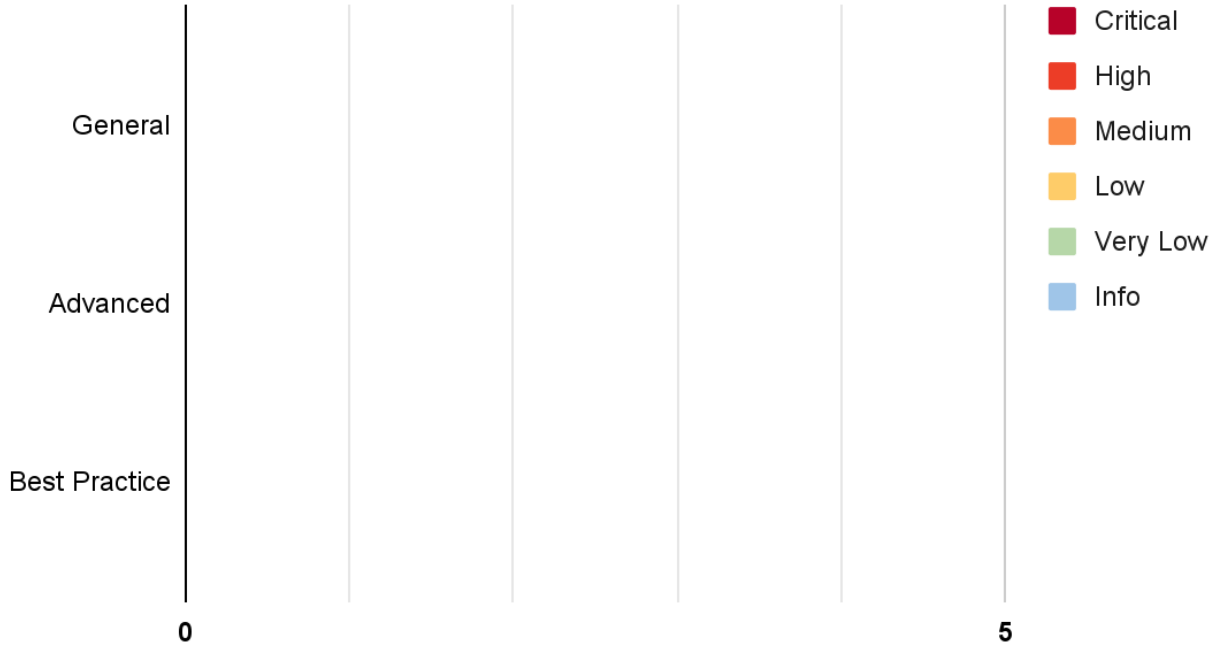
4. Summary of Findings

The following charts show the number of the issues found during the assessment and the issues acknowledged in the reassessment, categorized into three categories: **General**, **Advanced**, and **Best Practice**.

Assessment:



Reassessment:



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	Outdated Compiler Version	General	Very Low	Resolved
IDX-002	Inexplicit State Declaration	Best Practice	Info	Resolved

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

5. Detailed Findings Information

5.1. Outdated Compiler Version

ID	IDX-001
Target	ANIV20
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 the smart contract.
Status	**Resolved** Aniverse Team has resolved this issue by upgrading the Solidity compiler to the latest stable version.

5.1.1. Description

The Solidity compiler version specified in the smart contract was outdated. These versions have publicly known inherent bugs (https://docs.soliditylang.org/en/v0.8.15/bugs.html) that may potentially be used to cause damage to the smart contracts or the users of the smart contracts.

ANIV20.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity 0.8.0;
```

5.1.2. Remediation

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

At the time of the audit, the latest stable version of Solidity compiler in major 0.8 is v0.8.15.

5.2. Inexplicit State Declaration

ID	IDX-002
Target	ANIV20
Category	Smart Contract Best Practice
CWE	CWE-710: Improper Adherence to Coding Standards
Risk	**Severity:** Info **Impact: None** **Likelihood: None**
Status	**Resolved** Aniverse Team has resolved this issue by declaring the state variable visibilities as public.

5.2.1. Description

The state variable visibility should be explicitly declared otherwise it will be internal. This improves the readability of the contract, allowing clear distinction on the scoping of these states.

The following state variables are declared without explicit visibility.

ANIV20.sol

```
21 address MainAddress;
22 address TeamAddress;
23 address PartnerAddress;
24 address MarketingAddress;
```

5.2.2. Remediation

Inspex suggests properly defining the state variable visibilities according to the state variable purpose. For example, if these state variables are needed to be publicly accessible, these state variable visibilities should be masked as public.

ANIV20.sol

```
21 address public MainAddress;
22 address public TeamAddress;
23 address public PartnerAddress;
24 address public MarketingAddress;
```

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