HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Wanderverse / Adactive Asia Pte Ltd
Date: September 19th, 2022

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for Wanderverse / Adactive Asia Pte Ltd
Approved By	Noah Jelich | SC Audits Department Head at Hacken OU
Type	ERC20 token
Platform	EVM
Network	Polygon, Matic
Language	Solidity
Methods	Manual Review, Automated Review, Architecture Review
Website	https://www.walk.com.sg/
Timeline	28.07.2022 - 19.09.2022
Changelog	01.08.2022 - Initial Review 23.08.2022 - Second Review 19.09.2022 - Third Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Wanderverse / Adactive Asia Pte Ltd (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 contracts.

Scope

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

Initial review scope

Repository:
https://github.com/adactiveasia/wanderverse-token-internal

Commit:
ac73c144aff20bc4304c3aef6cdb84746ff6e1a1

Technical Documentation:
Type: README
Link

Integration and Unit Tests: No

Deployed Contracts Addresses: No

Contracts:
File: ./src/contracts/WanderTokenPol.sol
SHA3: afec508107250e5b41b59574c31077533a89f9f9ee1162b97ae74a0588a57961

Second review scope

Repository:
https://github.com/adactiveasia/wanderverse-token-internal

Commit:
87383badf6fb7bdb9c1f5f4750755ed32aade9ce

Technical Documentation:
Type: README
Link

Integration and Unit Tests: Yes

Deployed Contracts Addresses: No

Contracts:
File: ./src/contracts/WanderTokenPol.sol
SHA3: 53fc7cdd24362251b6d2c282bf022abc2b92c4b369ab19e40fac3df1e617565b

Third review scope

Repository:
https://github.com/adactiveasia/wanderverse-token-internal

Commit:
45775771f50e9a045a7634fb04d9212e87515ccb

Technical Documentation:
Type: README
Link

Integration and Unit Tests: Yes

Deployed Contracts Addresses: No

Contracts:
File: ./src/contracts/WanderTokenPol.sol
SHA3: 868e387caf38ff33eb5a3b8f9730d22db4285c6828779c3bf0ade2fd41c5e15a

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 cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution.

Executive Summary

The score measurement details can be found in the corresponding section of the methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**. Detailed functional, technical requirements, and tokenomics were provided. A brief technical overview was provided in the README file of the project repository.

Code quality

The total CodeQuality score is **10** out of **10**. No unused, hardcoded, or redundant variable is detected. Unit tests are provided; they cover negative and positive test cases. **Test coverage is 100%.**

Architecture quality

The architecture quality score is **10** out of **10**. Truffle is used as a development environment, and instructors were provided in the README. Deployment scripts were provided, and the local development environment was documented strongly.

Security score

As a result of the audit, the code contains **1** low severity issue. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **10**.

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
15 September 2022	1	0	0	0

Checked Items

We have audited provided smart contracts for commonly known and more specific vulnerabilities. Here are some of the items that are considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Failed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Not Relevant
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless it is required.	Passed
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization	SWC-115	tx.origin should not be used for	Passed

through tx.origin		authorization.	
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Not Relevant
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifier should always be used. All parameters from the signature should be used in signer recovery	Not Relevant
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of unused variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP standards violation	EIP	EIP standards should not be violated.	Passed
Assets integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions.	Passed
User Balances manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed
Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Not Relevant
Token Supply manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Passed
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of	Passed

		data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	
Style guide violation	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Not Relevant
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be 100%, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed
Stable Imports	**Custom**	The code should not reference draft contracts, that may be changed in the future.	Passed

System Overview

Wanderverse is an ERC20 token with the following contract:

- *WanderTokenPol* – ERC-20 token that allows additional minting and allows administrators to execute restricted operations with a voting system.
  It has the following attributes:
  - Name: Wanderverse Token
  - Symbol: WANDER
  - Decimals: 18
  - Maximum supply: 1 billion tokens.

Privileged roles

- Specified operations are controlled by administrators defined in the contract. An administrator of the *WanderTokenPol can:*
  - *submit a request for operations below:*
    - *The addition and removal of an account's administrator rights.*
    - *Transferring 1,000,000 $WANDERtokens from the administrator's balance.*
    - *The minting any amount of $WANDER tokens.*
    - *Pausing or un-pausing the state of the smart contract.*
    - *Clearing up the request pool that holds upon the requests to execute the crucial functions mentioned above.*
  - *vote for a submitted request.*

Findings

■■■■ Critical

1. Requirements violation

Administrators' authorization is restricted via voting system not to mint more than 1000000 (1 million) tokens. However, the *mint* function in the contract allows minting if the amount is less than 1 million tokens.

Administrators can reach restricted amounts by calling the *mint* function repeatedly.

File: ./contracts/WanderTokenPol.sol

Contract: WanderTokenPol

Function: mint

Recommendation: Fix the logic issue.

Status: Fixed (Revised commit: 87383badf6fb7bdb9c1f5f4750755ed32aade9ce)

■■■ High

1. Requirements violation

Although it is specified in the documentation that pausing or unpausing is only done after an administrator gets a positive result from request voting, in the code implementation pause function can be called by any administrator without a confirmed pausing request.

File: ./contracts/WanderTokenPol.sol

Contract: WanderTokenPol

Functions: pause, unpause

Recommendation: Override the pause and unpause functions in ERC20PresetMinterPauser.sol to not be called without voting.

Status: Fixed (Revised commit: 87383badf6fb7bdb9c1f5f4750755ed32aade9ce)

2. Requirements violation

Granting a role must be done by only an administrator after submitting a request and getting a confirmation according to the docs.

grantRole function in *OpenZeppelin/AccessControlUpgradeable* contract is open to be called by administrators without positive voting.

File: ./contracts/WanderTokenPol.sol

Contract: WanderTokenPol

Function: grantRole

Recommendation: Override the grantRole function, so it is not callable without voting.

Status: Fixed (Revised commit: 87383badf6fb7bdb9c1f5f4750755ed32aade9ce)

3. Potential Out-of-Gas exception

If the number of administrators is large enough, the iterator on lines 150-155 may revert due to an Out-of-Gas exception.

File: ./contracts/WanderTokenPol.sol

Contract: WanderTokenPol

Function: executeRequest

Recommendation: Limit the number of administrators.

Status: Fixed (Revised commit: 87383badf6fb7bdb9c1f5f4750755ed32aade9ce)

Medium

1. Missing event-emitting

No event is emitted for internal state changes after a request is submitted or executed.

File: ./contracts/WanderTokenPol.sol

Contract: WanderTokenPol

Recommendation: Write “Event” and emit them for important state changes.

Status: Fixed (Revised commit: 45775771f50e9a045a7634fb04d9212e87515ccb)

Low

1. Floating pragma

Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

File: ./contracts/WanderTokenPol.sol

Contract: WanderTokenPol

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

Status: Reported (Revised commit: 45775771f50e9a045a7634fb04d9212e87515ccb)

2. Style guide violation

The provided contract does not follow the official guidelines.

File: ./contracts/WanderTokenPol.sol

Contract: WanderTokenPol

Recommendation: Follow the official Solidity guidelines. Official Style Guide

Status: Fixed (Revised commit: 45775771f50e9a045a7634fb04d9212e87515ccb)

3. Use of hard-coded values

MAXSUPPLY ,*NUMCONFIRMSREQUIRED* and *MAXADMINS* variables are constant, and there is no implementation to set them. Therefore, declaring them as constant variable saves Gas.

File: ./contracts/WanderTokenPol.sol

Contract: WanderTokenPol

Recommendation: Convert these variables into constants.

Status: Fixed (Revised commit: 45775771f50e9a045a7634fb04d9212e87515ccb)

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date of this report, with 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 report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted to and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

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.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

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, Consultant cannot guarantee the explicit security of the audited smart contracts.