verichains

SECURITY AUDIT OF

HERA

SMART CONTRACT

PUBLIC REPORT

September 24, 2021

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ACRONYMS AND ABBREVIATIONS

NAME	DESCRIPTION
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.

EXECUTIVE SUMMARY

This Security Audit Report prepared by Verichains Lab on Sep 17, 2021. We would like to thank the Hero Arena team for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the HERA Contract. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

The assessment did not identify any vulnerability issue in HERA smart contract code.

Overall, the code reviewed is of good quality, written with the awareness of smart contract development best practices.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Hero Arena and HERA

Hero Arena is a Metaverse RPG Gaming developed on Binance Smart Chain and Polygon.

HERA (HERA TOKEN) is the BEP-20 token of Hero Arena, with an initial total supply of 100 million.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the HERA Contract.

The audited contract is the HERA Contract that was deployed on Binance Smart Chain Mainnet at address *0x49C7295ff86EaBf5bf58C6eBC858DB4805738c01*. The details of the deployed smart contract are listed in Table 1.

FIELD	VALUE
Contract Name	HERA
Contract Address	0x49C7295ff86EaBf5bf58C6eBC858DB4805738c01
Compiler Version	v0.8.6+commit.11564f7e
Optimization Enabled	No with 200 runs
Explorer	https://bscscan.com/address/0x49c7295ff86eabf5bf58c6ebc858db4805738c01

Table 1: The deployed smart contract details

1.3. Audit methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Integer Overflow and Underflow.

- Timestamp Dependence.
- Race Conditions.
- Transaction-Ordering Dependence.
- DoS with (Unexpected) revert.
- DoS with Block Gas Limit.
- Gas Usage, Gas Limit and Loops.
- Redundant fallback function.
- Unsafe type Inference.
- Reentrancy.
- Explicit visibility of functions state variables (external, internal, private and public).
- Logic Flaws.

For vulnerabilities, we categorize the findings into categories as listed in Table 2, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 2: Vulnerability severity levels

1.4. Disclaimer

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

The HERA Contract is an BEP-20 Token Contract[1], which implements a standard interface for token as defined in IBEP-20[2]. This standard provides basic functionality to transfer tokens, as well as allow tokens to be approved so they can be spent by another on-chain third party.

Table 3 lists some properties of the audited HERA Contract (as of the report writing time).

PROPERTY	VALUE
Name	HERA TOKEN
Symbol	HERA
Decimals	18
Owner	0x3fa6C7f57398C6933E8fEbe0a670e472361e04Ea
Total Supply	100,000,000 ($\times 10^{18}$) Note: the number of decimals is 18, so the total representation tokens will be 100,000,000, or 100 million.

Table 3: The HERA Token properties

Besides the standard interface of an BEP-20 token, the HERA contract also implements some additional functional logics by extending the following abstract contracts:

- Context: provides information about the current execution context, including the sender of the transaction and its data. While these are generally available via msg.sender and msg.data, they should not be accessed in such a direct manner, since when dealing with GSN meta-transactions the account sending and paying for execution may not be the actual sender (as far as an application is concerned).
- Ownable: this contract has a basic access control mechanism, where there is an account (an owner) that can be granted exclusive access to specific functions. By default, the owner account will be the one that deploys the contract. The current owner can renounce or transfer ownership to a new owner.

[1] https://docs.binance.org/smart-chain/developer/BEP20.html
[2] https://docs.binance.org/smart-chain/developer/IBEP20.sol

2.2. Contract codes

The HERA Contract was written in Solidity language[3], with the required version to be 0.8.6.

The source codes consist of one contract, two abstract contracts, one interface and one library. Almost all source codes in the HERA Contract are imported from Binance's implementation template of BEP20-related contracts.

2.2.1. IBEP20 interface

This is the interface of the BEP-20 token standard. The source code is referenced from the official Binance's documentation.

2.2.2. Context abstract contract

This abstract contract provides information about the current execution context, including the sender of the transaction and its data. The source code is referenced from OpenZeppelin's implementation.

2.2.3. Ownable abstract contract

This abstract contract makes the HERA Contract ownable which extends `Context` abstract contract. The source code is referenced from OpenZeppelin's implementation.

2.2.4. SafeMath library

This library provides wrappers over Solidity's arithmetic operations with added overflow checks. The source code is referenced from OpenZeppelin's implementation.

2.2.5. HERA contract

This is the main contract in the HERA contract which extends the `IBEP20` and `Ownable` abstract contract. Below is a summary of some important functions in this contract:

- `constructor()`: constructor set the name, symbol, decimal for the contract.
- `getOwner()`: returns the bep token owner.
- `decimas()`: returns the decimals.
- `symbol()`: returns the token symbol.
- `name()`: returns the token name.
- `totalSupply()`: returns the total supply.
- `balanceOf(address account)`: returns the balance of the input account.

[3] https://docs.soliditylang.org/en/latest

- *transfer(address recipient, uint256 amount):* transfers `amount` tokens from the sender to `recipient`.
- *allowance(address owner, address spender):* returns the `amount` which `spender` is still allowed to withdraw from `owner`.
- *approve(address spender, uint256 amount):* allows `spender` to withdraw from the caller's account multiple times, up to the `amount`. If this function is called again it overwrites the current allowance with `amount`.
- *transferFrom(address sender, address recipient, uint256 amount):* transfers `amount` of tokens from address `sender` to address `recipient`.
- *increaseAllowance(address spender, uint256 addedValue):* increases the allowance amount for `spender` by `addedValue`.
- *decreaseAllowance(address spender, uint256 subtractedValue):* decreases the allowance amount for `spender` by `subtractedValue`.

2.3. Findings

During the audit process, the audit team did not discover any security vulnerability issue in the HERA Contract.

2.4. Additional notes and recommendations

2.4.1. Unused SafeMath library

The library `SafeMath` is unused in this contract.

RECOMMENDATION

Remove the `SafeMath` library.

UPDATES

This finding has been acknowledged by Hero Arena team.

3. VERSION HISTORY

VERSION	DATE	STATUS/CHANGES	CREATED BY
1.0	Sep 24, 2021	Initial private report	Verichains Lab
1.1	Sep 24, 2021	Public report	Verichains Lab

APPENDIX A: FUNCTION CALL GRAPH

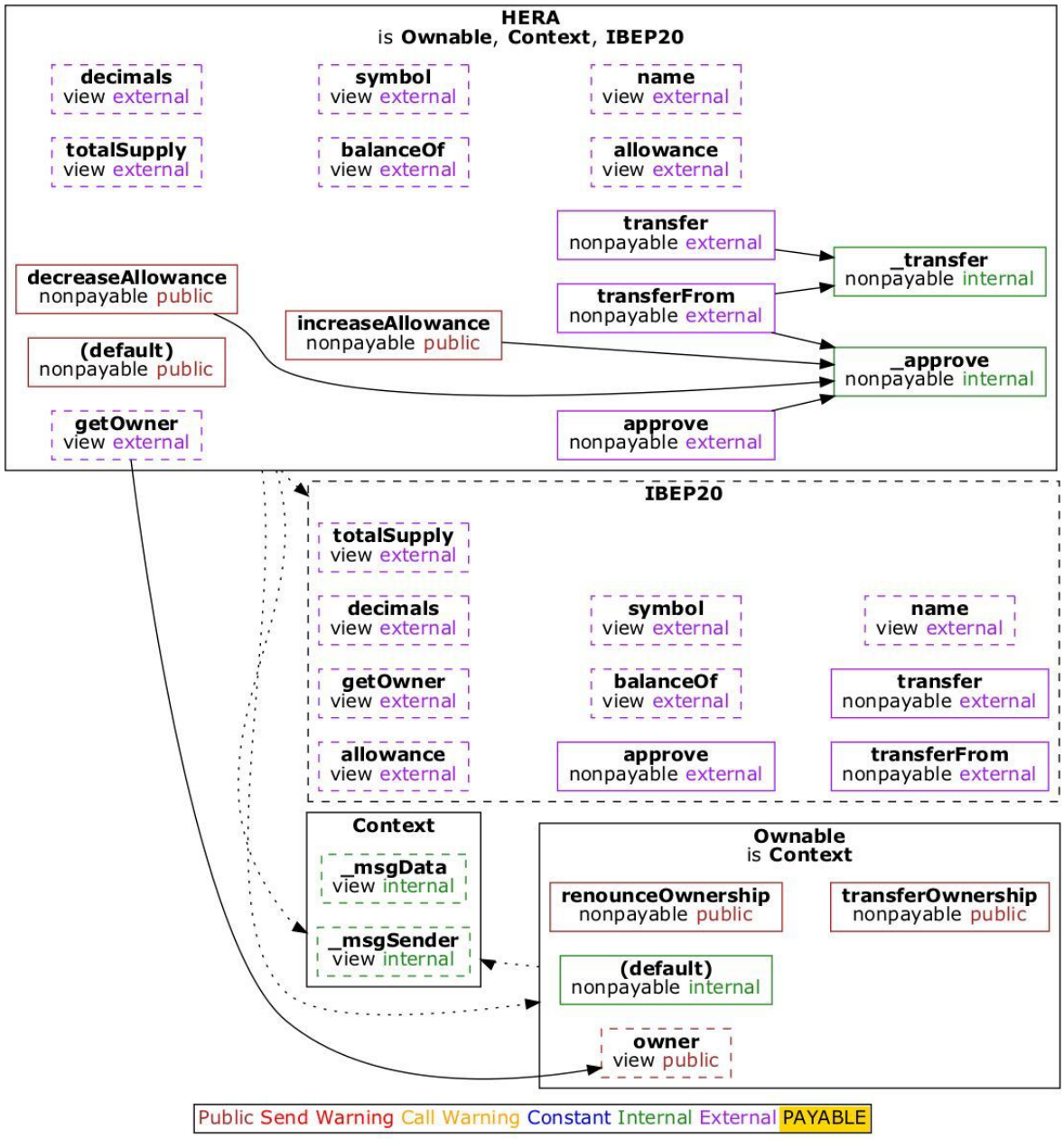

Figure 1: The function call graph of HERA smart contract