Hashcheck

Security Audit

Smart Contract Audit Report for

El Hippo [HIPP]

{ September 2023 }

01. Introduction

This document includes the results of the audit performed by the Hashcheck team on the HIPP Token project.

Token's Name:

El Hippo Token

Token's Symbol:

HIPP

Token's Precision:

18

Audited Source File's Address:

https://etherscan.io/token/0x7b744eEa1dECa2f1B7b31F15Ba036Fa1759452d7#code

The goal of this audit is to review HIPP’s token issuance function, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire codebase horizontally, which could improve its quality as a whole.

This audit only applies to the specified code, software or any materials supplied by the HIPP team for specified versions. Whenever the code, software, materials, settings, environment etc is changed, the comments of this audit will no longer apply.

— Disclaimer

Note that as of the date of publishing, the contents of this report reflect the current understanding of known security patterns and state of the art regarding system security. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. If the audited source files are smart contract files, risks or issues introduced by using data feeds from off-chain sources are not extended by this review either.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

— The HIPP Team's Consent/Acknowledgement:

The audited materials of the project including but not limited to the documents, home site, source code, etc are all developed, deployed, managed, and maintained outside Mainland CHINA.

The members of the team, the foundation, and all the organizations that participate in the audited project are not Mainland Chinese residents.

The audited project doesn’t provide services or products for Mainland Chinese residents.

— Methodology

The above files' code was studied in detail in order to acquire a clear impression of how the its specifications were implemented. The codebase was then subject to deep analysis and scrutiny, resulting in a series of observations. The problems and their potential solutions are discussed in this document and, whenever possible, we identify common sources for such problems and comment on them as well.

The Hashcheck auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Hashcheck to make sure we understand the size, scope, and functionality of the project's source code.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Hashcheck describe.

2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run the test cases. ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the source code to improve maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

— Structure of the document

This report contains a list of issues and comments on all the above source files. Each issue is assigned a severity level based on the potential impact of the issue and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

— Documentation

For this audit, we used the following source of

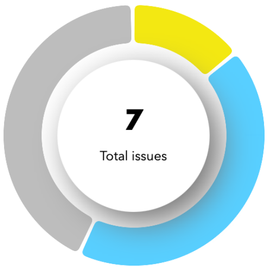
truth about how the token issuance should work:

https://etherscan.io/token/0x7b744eEa1dECa2f1B7b31F15Ba036Fa1759452d7#code

This was considered the specification.

Appendix A. Issues' severity classification

- **Critical.** Issues that may cause an unlimited loss of funds or entirely break the contract workflow. Malicious code (including malicious modification of libraries) is also treated as a critical severity issue. These issues must be fixed before deployments or fixed in already running projects as soon as possible.
- **High.** Issues that may lead to a limited loss of funds, break interaction with users, or other contracts under specific conditions. Also, issues in a smart contract, that allow a privileged account the ability to steal or block other users' funds.
- **Medium.** Issues that do not lead to a loss of funds directly, but break the contract logic. May lead to failures in contracts operation.
- **Low.** Issues that are of a non-optimal code character, for instance, gas optimization tips, unused variables, errors in messages.
- **Informational.** Issues that do not impact the contract operation. Usually, informational severity issues are related to code best practices, e.g. style guide.



● Critical	0	
● High	0	
● Medium	🟨	1
● Low	🟦	3
● Informational	▨	3

Summary:

The Hashcheck security team used its auto analysis tools and manual work to audit the project. During the audit, no issues were discovered.

— Comments from Auditor

Serial Number	Auditor	Audit Time	Result
202302091799	Hashcheck Security Team	Sep 2023	Passed

Project Summary

Project Name	ElHippo
Address	**0x7b744eEa1dECa2f1B7b31F15Ba036Fa1759452d7**
Network	1

Issue ID	183
Severity	**High**
Status	**Optimization**

Description Code	**string private** _nameFallback;
Location	EIP712._nameFallback (EIP712.sol#52) should be constant

Issue ID	183
Severity	**High**
Status	**Optimization**
Description Code	**string private** _versionFallback;
Location	EIP712._versionFallback (EIP712.sol#53) should be constant

Issue ID	183
Severity	**High**
Status	**Optimization**
Description Code	
Location	ERC20Permit._PERMIT_TYPEHASH_DEPRECATED_SLOT (ERC20Permit.sol#37) should be constant

Issue ID 103

Severity **High**

Status **Informational**

Description Code

```
pragma solidity ^0.8.0;
```

Location

Different versions of Solidity is used:
- Version used: ['^0.8.0', '^0.8.8', '^0.8.9']
- ^0.8.0 (AccessControl.sol#4)
- ^0.8.0 (IAccessControl.sol#4)
- ^0.8.0 (IERC5267.sol#4)
- ^0.8.0 (ERC20.sol#4)
- ^0.8.0 (IERC20.sol#4)
- ^0.8.0 (ERC20Burnable.sol#4)
- ^0.8.0 (ERC20Permit.sol#4)
- ^0.8.0 (ERC20Snapshot.sol#4)
- ^0.8.0 (IERC20Metadata.sol#4)
- ^0.8.0 (IERC20Permit.sol#4)
- ^0.8.0 (draft-ERC20Permit.sol#4)
- ^0.8.0 (Arrays.sol#4)
- ^0.8.0 (Context.sol#4)
- ^0.8.0 (Counters.sol#4)
- ^0.8.8 (ShortStrings.sol#4)
- ^0.8.0 (StorageSlot.sol#5)
- ^0.8.0 (Strings.sol#4)
- ^0.8.0 (ECDSA.sol#4)
- ^0.8.8 (EIP712.sol#4)
- ^0.8.0 (ERC165.sol#4)
- ^0.8.0 (IERC165.sol#4)
- ^0.8.0 (Math.sol#4)
- ^0.8.0 (SignedMath.sol#4)
- ^0.8.0 (AccessControl.sol#4)
- ^0.8.0 (Arrays.sol#4)
- ^0.8.0 (Context.sol#4)
- ^0.8.0 (Counters.sol#4)
- ^0.8.0 (ECDSA.sol#4)
- ^0.8.8 (EIP712.sol#4)
- ^0.8.0 (ERC165.sol#4)
- ^0.8.0 (ERC20.sol#4)
- ^0.8.0 (ERC20Burnable.sol#4)
- ^0.8.0 (ERC20Permit.sol#4)
- ^0.8.0 (ERC20Snapshot.sol#4)

Issue ID	156
Severity	**Medium**
Status	**Low**
Description Code	
Location	Math.mulDiv(uint256,uint256,uint256) (Math.sol#55- 134) performs a multiplication on the result of a division: -denominator = denominator / twos (Math.sol#101) -inverse = (3 * denominator) ^ 2 (Math.sol#116)

Issue ID	156
Severity	**Medium**
Status	**Low**
Description Code	
Location	Math.mulDiv(uint256,uint256,uint256) (Math.sol#55- 134) performs a multiplication on the result of a division: -denominator = denominator / twos (Math.sol#101) -inverse *= 2 - denominator * inverse (Math.sol#120)

Issue ID	156
Severity	**Medium**
Status	**Low**

Description Code	
Location	Math.mulDiv(uint256,uint256,uint256) (Math.sol#55- 134) performs a multiplication on the result of a division: -denominator = denominator / twos (Math.sol#101) -inverse *= 2 - denominator * inverse (Math.sol#121)

Issue ID	156
Severity	**Medium**
Status	**Low**
Description Code	
Location	Math.mulDiv(uint256,uint256,uint256) (Math.sol#55- 134) performs a multiplication on the result of a division: -denominator = denominator / twos (Math.sol#101) -inverse *= 2 - denominator * inverse (Math.sol#122)

Issue ID	156
Severity	**Medium**
Status	**Low**
Description Code	

Location	Math.mulDiv(uint256,uint256,uint256) (Math.sol#55- 134) performs a multiplication on the result of a division: -denominator = denominator / twos (Math.sol#101) -inverse *= 2 - denominator * inverse (Math.sol#123)

Issue ID	156
Severity	**Medium**
Status	**Low**
Description Code	
Location	Math.mulDiv(uint256,uint256,uint256) (Math.sol#55- 134) performs a multiplication on the result of a division: -denominator = denominator / twos (Math.sol#101) -inverse *= 2 - denominator * inverse (Math.sol#124)

Issue ID	156
Severity	**Medium**
Status	**Low**
Description Code	

Location	Math.mulDiv(uint256,uint256,uint256) (Math.sol#55- 134) performs a multiplication on the result of a division: -denominator = denominator / twos (Math.sol#101) -inverse *= 2 - denominator * inverse (Math.sol#125)

Issue ID	156
Severity	**Medium**
Status	**Low**
Description Code	
Location	Math.mulDiv(uint256,uint256,uint256) (Math.sol#55- 134) performs a multiplication on the result of a division: -prod0 = prod0 / twos (Math.sol#104) -result = prod0 * inverse (Math.sol#131)

Issue ID	184
Severity	**High**
Status	**Optimization**

Description Code	**function** **snapshot**() **public** **onlyRole**(SNAPSHOT_ROLE) {
	_snapshot();
	}

Location	snapshot() should be declared external: - ElHippo.snapshot() (contract-9c6892d960.sol#19-21)

Issue ID	177
Severity	**High**
Status	**Informational**
Description Code	**pragma solidity** ^0.8.0;
Location	Pragma version^0.8.0 (AccessControl.sol#4) allows old versions

Issue ID	177
Severity	**High**
Status	**Informational**
Description Code	**pragma solidity** ^0.8.8;
Location	Pragma version^0.8.8 (ShortStrings.sol#4) is known to contain severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)

Issue ID	177

Severity	**High**
Status	**Informational**
Description Code	**pragma solidity** ^0.8.9;
Location	Pragma version^0.8.9 (contract-9c6892d960.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7

Issue ID	177
Severity	**High**
Status	**Informational**
Description Code	
Location	solc-0.8.18 is not recommended for deployment

Issue ID	210-b
Severity	**High**
Status	**Medium**
Description Code	

Location	Possible Honeypot: - Function: ERC20.transfer(address,uint256) (ERC20.sol#113-117) - External call: snapshots.values.push(currentValue) - External call: snapshots.ids.push(currentId)

Issue ID	210-b
Severity	**High**
Status	**Medium**
Description Code	
Location	Possible Honeypot: - Function: ERC20.transferFrom(address,address,uint2 56) (ERC20.sol#158-163) - External call: snapshots.values.push(currentValue) - External call: snapshots.ids.push(currentId)

Issue ID	182
Severity	**Medium**
Status	**Informational**
Description Code	**bytes32 private constant** _FALLBACK_SENTINEL = 0x0000000000000000000000000000000000000000 000 0000000000000000000FF;

Location	Contract ShortStrings uses literals with too many digits: - _FALLBACK_SENTINEL = 0x0000000000000000000000000000000000000000000 00000000000000000000FF (ShortStrings.sol#42)

Issue ID	182
Severity	**Medium**
Status	**Informational**

Description Code	**constructor**() **ERC20**("El Hippo", "HIPP")
	ERC20Permit("El Hippo") {
	_grantRole(DEFAULT_ADMIN_ROLE, msg.sender); _grantRole(SNAPSHOT_ROLE, msg.sender);
	_mint(msg.sender, 777000000000000 * 10 ** decimals());
	}

Location	ElHippo.constructor() (contract-9c6892d960.sol#13- 17) uses literals with too many digits: - _mint(msg.sender,777000000000000 * 10 ** decimals()) (contract-9c6892d960.sol#16)