AstraSec

BlasterSwap

Security Audit Report

February 29, 2024

Contents

1 Introduction

1.1 About BlasterSwap

`BlasterSwap` is a native `Blast` L2 DEX, offering v2-style full-range liquidity. It eliminates trusted intermediaries and unnecessary forms of rent extraction, allowing for fast, efficient trading, besides leveraging `Blast` L2 native features such as gas refund and liquidity autorebase.

1.2 Source Code

The following source code was reviewed during the audit:

- https://github.com/blasterswap/core-v2/
- Commit ID: c5ca4f7

And this is the final version representing all fixes implemented for the issues identified in the audit:

- https://github.com/blasterswap/blasterswap-core-v2/
- Commit ID: f256ce4

2 | Overall Assessment

This report has been compiled to identify issues and vulnerabilities within the `BlasterSwap` project. Throughout this audit, we identified a total of 2 issues spanning various severity levels. By employing auxiliary tool techniques to supplement our thorough manual code review, we have discovered the following findings.

Severity	Count	Acknowledged	Won't Do	Addressed
Critical	-	-	-	-
High	-	-	-	-
Medium	-	-	-	-
Low	-	-	-	-
Informational	2	-	-	2
Undetermined	-	-	-	-

3 Vulnerability Summary

3.1 Overview

Click on an issue to jump to it, or scroll down to see them all.

~~I-1~~ Suggested Constant BLAST Address for Gas Saving

~~I-2~~ Improved Validation for Governor in BlasterswapV2Factory

3.2 Security Level Reference

In web3 smart contract audits, vulnerabilities are typically classified into different severity levels based on the potential impact they can have on the security and functionality of the contract. Here are the definitions for critical-severity, high-severity, medium-severity, and low-severity vulnerabilities:

Severity	Description
C-X (Critical)	A severe security flaw with immediate and significant negative consequences. It poses high risks, such as unauthorized access, financial losses, or complete disruption of functionality. Requires immediate attention and remediation.
H-X (High)	Significant security issues that can lead to substantial risks. Although not as severe as critical vulnerabilities, they can still result in unauthorized access, manipulation of contract state, or financial losses. Prompt remediation is necessary.
M-X (Medium)	Moderately impactful security weaknesses that require attention and remediation. They may lead to limited unauthorized access, minor financial losses, or potential disruptions to functionality.
L-X (Low)	Minor security issues with limited impact. While they may not pose significant risks, it is still recommended to address them to maintain a robust and secure smart contract.
I-X (Informational)	Warnings and things to keep in mind when operating the protocol. No immediate action required.
U-X (Undetermined)	Identified security flaw requiring further investigation. Severity and impact need to be determined. Additional assessment and analysis are necessary.

3.3 Vulnerability Details

[I-1] Suggested Constant BLAST Address for Gas Saving

Target	Category	IMPACT	LIKELIHOOD	STATUS
Multiple Contracts	Coding Practices	NA	NA	🔗Addressed

The contracts in `BlasterSwap` protocol interact with the `Blast` contract located at $0x430...002$ to change their gas mode. The current implementation directly uses address $0x430...002$ everywhere it needs to interact with the `Blast` contract. As a good programming practice, we recommend defining the address $0x430...002$ as a constant variable (e.g., `BLAST`) so that this constant variable can be used wherever needed.

BlasterswapV2Factory::constructor()

```
constructor(address _feeToSetter, address _governor) public {
    feeToSetter = _feeToSetter;
    governor = _governor;

    BLAST = IBlast(0x4300000000000000000000000000000000000002);

    BLAST.configureClaimableGas();
    BLAST.configureGovernor(_governor);
}
```

Note the same suggestion can also be applied to `BlasterswapV2Pair`, `BlasterswapV2Router02` contracts.

Remediation Properly define the address of `Blast` contract as a constant variable.

[I-2] Improved Validation for Governor in BlasterswapV2Factory

Target	Category	IMPACT	LIKELIHOOD	STATUS
BlasterswapV2Factory.sol	Coding Practices	NA	NA	🔗Addressed

`BlasterSwap` protocol calls the `configureGovernor()` function of the `Blast` contract to set its `governor` address. After that, the `governor` can set the gas mode, claim gas fees and reconfigure the governor. However, we notice that the current implementation does not properly validate the input `governor` argument. As a result, if the input `governor` is `address(0)`, the `governor` of the `BlasterSwap` protocol will not be changed at all. Based on this, we suggest to add proper validation for the input `governor` argument and ensure it is a valid address (i.e., `!address(0)`).

BlasterswapV2Factory::constructor()

```
constructor(address _feeToSetter, address _governor) public {
    feeToSetter = _feeToSetter;
    governor = _governor;

    BLAST = IBlast(0x4300000000000000000000000000000000000002);

    BLAST.configureClaimableGas();
    BLAST.configureGovernor(_governor);
}
```

Remediation Add proper validation for the `governor` argument and ensure it's not `address(0)`.

4 Appendix

4.1 About AstraSec

AstraSec is a blockchain security company that serves to provide high-quality auditing services for blockchain-based protocols. With a team of blockchain specialists, AstraSec maintains a strong commitment to excellence and client satisfaction. The audit team members have extensive audit experience for various famous DeFi projects. AstraSec's comprehensive approach and deep blockchain understanding make it a trusted partner for the clients.

4.2 Disclaimer

4.3 Contact

Name	AstraSec Team
Phone	+86 156 0639 2692
Email	contact@astrasec.ai
Twitter	https://twitter.com/AstraSecAI