QuillAudits

Audit Report
August, 2022

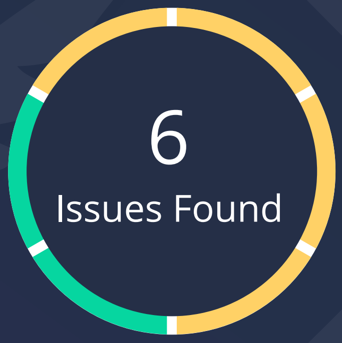
For

ASTRO
gallery

Table of Content

Executive Summary

Project Name	Astrobabies
Overview	Astrobabies offers a wide variety of the NFTs to its users and a lot of benefits tied with it. While its major offering is the marketplace where different NFTs can be sold and bought.
Timeline	20 July ,2022 - 23 August, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse the Astrobabies solana program's codebase for quality, security, and correctness. *https://gitlab.com/roots-projects/blockchain/astrogallery/astro-marketplace-contract.git* Branch: Development Commit: fb75cf05608cccfd450501164dee9eef76fe1f93
Fixed In	*https://gitlab.com/roots-projects/blockchain/astrogallery/astro-marketplace-contract/-/tree/developer/programs/Astro_Marketplace/src* Branch: Development Commit: e17c00c1e82adf326b53e4108229e01c8487dddd



🟥 High 🟨 Medium

🟩 Low 🟪 Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	**4**	**2**	0

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls
- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Using block.timestamp
- ✓ Multiple Sends
- ✓ Using SHA3
- ✓ Using suicide
- ✓ Using throw
- ✓ Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

High Severity Issues

No issues found

Medium Severity Issues

1. Parametrise bump would increase the chances of address confusion

File: programs/Astro_Marketplace/src/lib.rs

Description

Throughout the program bumps are provided as parameter for the different instructions, While it create a lot of dependency on the frontend system to store the canonical bump and provide it everytime. Frontend system is out of the scope so we can't assume that it always provide the canonical bump, If any other bump get provided then the storage of the account would get messed up.

Current architect doesn't work seamless for solana program composibility as bump calculation need to be performed onchain during program to program invocation.

Recommendation

Always recommended to store the canonical bump during the initialization of the account. For more details please refer *this*.

Status

Resolved

2. Missing zero fee validation

File: programs/Astro_Marketplace/src/lib.rs[#L56]

Description

At [#L56] sol_fee & market_fee can be set to zero as there is no validation exists in the instruction to make sure that sol_fee and market_fee can't be set zero.

Recommendation

It is recommended to have zero value check for sol_fee and market_fee.

Status

Resolved

3. Missing zero price check for NFTs during listing

File: programs/Astro_Marketplace/src/lib.rs[#L239]

Description

At [#L239-240] price of the NFT get set but there is no validation on its value as it can be set to 0 or can be set as high as 100K SOL. Although upper bound is not necessary while sometimes lower bound is necessary when market is charging fees to facilitate the listing.

Recommendation

It is recommended to add a zero value validation of the price_sol and price_token so values should be greater than zero.

Status

Resolved

4. Missing essential sanity checks during creation of auction

File: programs/Astro_Marketplace/src/lib.rs[#L1251]

Description

At [#L1251] min_increase_amount get directly set without making sure the value should be greater than zero. If it gets sets to zero then the buyer wouldn’t be able to create the bid as the bid value should be greater than the set value.

Similarly at [#L1253] end_date gets set directly without making sure that its value should be greater than the current timestamp. Otherwise create auction would get expired without receiving any bid from the buyers.

Recommendation

It is recommended to add below checks.

```
require!(min_increase > 0, MarketplaceError:InvalidMinimumIncreasevalue);
require!(end_date > Clock::get()?.unix_timestamp, MarketplaceError:InvalidEndDate);
```

Status

Resolved

Low Severity Issues

5. Junk values remain in the account

File: programs/Astro_Marketplace/src/lib.rs[#L149]

Description

At [L#141-L#153] team_treasury and treasury_rate gets removed while decrementing the team count so whenever the program access the different team members values then it will only read till the given count while the team_treasury and treasury_rate still have those junk values. This behaviour may create confusion to dApps or data collectors as they may not know that they need to validate the team_count first because the arrays are fixed size.

Recommendation

It is recommended to remove those junk values and set it to there default values.

Status

Resolved

6. Inefficient use of require! macro

File: programs/Astro_Marketplace/src/lib.rs

Description

Throughout the contract require! macro get used to do sanity checks even for comparing the pubKey, Although it is expensive in comparison to other variants of the require! macro.

Recommendation

It is recommended to use require_keys_eq! macro while comparing the pubKeys and use require_eq! for the equal operator.

Status

Resolved

7. Typographical error

File: programs/Astro_Marketplace/src/lib.rs[#L1784]

Description

A typographical mistake exists in the SetTreshold struct name.

Recommendation

Replace it with SetThreshold.

Status

Resolved

Functional Testing

Some of the tests performed are mentioned below

- ✓ should be able to initalize the global authority once.
- ✓ should be able to update fee by super admin.
- ✓ should be able to add and remove team from treasury.
- ✓ should be able to correctly init user pool.
- ✓ should be able to correctly init sell data.
- ✓ should be able to list nft for sale.
- ✓ should be able to delist nft from marketplace.
- ✓ should be able to purchase the nft from the listing.
- ✓ should be able to deposit the escrow.
- ✓ should be able to withdraw from escrow.
- ✓ should be able to init offer data.
- ✓ should be able to make offer.
- ✓ should be able to cancel offer.
- ✓ should be able to accept offer.
- ✓ should be able to init auction data.
- ✓ should be able to create auction.
- ✓ should be able to palace bid on running auction.
- ✓ should be able to claim auction.
- ✓ should be able to cancel auction.

Closing Summary

Some issues of Medium severity and Low severity were found. Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Astrobabies platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Astrobabies Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

500+
Audits Completed

$15B
Secured

500K
Lines of Code Audited

Follow Our Journey

Audit Report
August, 2022

For

ASTRO
gallery

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com