Audius Contracts Audit

AUGUST 25, 2020 | IN SECURITY AUDITS | BY OPENZEPPELIN SECURITY

INTRODUCTION

CRITICAL

HIGH

MEDIUM

LOW

NOTES

C

0
N
C
L
U
S
I
O
N

Introduction

The
Audius
team
asked
us
to
review
and
audit
their
smart
contracts.
We
looked
at
the
code
and
now
publish
our
results.

We
audited
commit
6f3b31562b9d4c43cef91af0a011986a2580fba2
of
the
AudiusProject/audius-
protocol/
repository.
In
scope
are
all
the
smart
contracts
in
the

`eth-contracts/contracts` directory, except for the `test` contracts.

All external code and contract dependencies were assumed to work correctly. Additionally, we assumed that the administrators are available, honest, and not compromised during this audit.

Update: *The Audius team made some fixes and comments based on our recommendations. They fixed*

all the critical, high, and medium severity issues that we reported. Below we address those fixes, which were introduced in pull requests #564 and #657. The Audius team then merged these pull requests into master, resulting in commit dac9cb31f0a2df9c25083bd3a833d285a4d947ef, which now includes the smart contract code we originally audited and

the fixes that we reviewed as part of this engagement. Our analysis of the mitigations disregards any other changes to the code base.

About Audius

Audius is a decentralized community of artists, developers, and listeners whose mission is to give everyone the freedom to share, monetize, and listen

to any audio.

The system comprises service providers who maintain the availability of the content, index the content for discovery, handle authentication, monitor activity, and cache.

This system is backed by these main smart contracts deployed in the Ethereum blockchain:

- Registry: hub where all the contracts are registered.

- ServiceTypeManager: maintains the different types of service providers, their versioning, and stake requirements.
- ServiceProviderFactory: manages the registration of service endpoints and their delegate owner wallet.
- AudiusToken: ERC-20 token used for staking on service providers and rewards.
- Staking: stores staking balances for service providers.
- DelegateManager: manages delegation of stake

to service providers, slashing and claiming rewards.

- ClaimFactory: mints and allocates tokens, and manages claim rounds.
- Governance: manages changes to the protocol through staked voting.

All the Audius contracts use the OpenZeppelin proxy pattern and can be upgraded through the Governance contract by voting on proposals submitted by stakers

or through administrator action.

The system administrators manage the registry, which lists all the target contracts that can be proposed for Governance voting. In the Governance contract migration, the registry ownership is transferred to the Governance contract, so all contracts added to the registry are intended to go through

the governance system. In this audit we assumed that contracts are extensively tested and audited before being added to the registry.

The administrators also control the `guardianAddress`, which can veto proposals made by the stakers, and directly execute any protocol changes without voting. According to the Audius team, “These

safeguards will be slowly removed over time through contract upgrades to Governance itself." With pull request #616, the Audius team also allowed the `guardianAddress` to submit proposals without stake.

***Update**: While we were auditing this project, the Audius team found the following issue:*

The guardian can veto a proposal

at any time from its creation until its evaluation.

However, if the `guardianAddress` waits to veto a proposal until the voting period is almost over, by making use of the `vetoProposal` function, this transaction can be frontrun by any staker calling the `evaluateProposalOutcome` function, which will modify the proposal's

outcome and make the `vetoProposal` transaction fail when the requirement for an `InProgress` proposal is not met.

Also, depending on the vote count, the guardian could be tricked to not veto the proposal because the proposal does not reach the necessary votes to be executed. However, a malicious staker

could vote near the voting deadline, changing the proposal from a failure into a success, and executing it while it frontruns the `vetoProposal` call.

Consider adding a cooldown period after the voting ends to allow the guardian to veto successful but malicious proposals.

Update: *Fixed. An* `executionDelay` *is now*

enforced on every proposal to account for this vulnerability.

READ ALL THE ISSUES

Security Audits

- If you are interested in smart contract security, you can continue the discussion in our forum, or even better, join the team 🚀
- If you are building a project of

your own and would like to request a security audit, please do so here.

RELATED POSTS