Audit Report

BrainTrust

Aug 9, 2021

Table of Contents

License

Disclaimer

Oak Security

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of this Report

Oak Security has been engaged by FREELANCE LABS, INC., to perform a security audit of the BrainTrust Web application and blockchain integration.

The objectives of the audit are as follows:

1. Determine the correct functioning of the system, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract integrations bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following source code repositories:

https://git.hexocean.com/mb/braintrust

Commit hash: `96412c2d3bfa8864254cb6c25684a577b0e43ad4`

https://git.hexocean.com/braintrust-token-api/token-api-backend

Commit hash: 03c14ccca6c518809aa0c414834aebb39b13054c

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line by line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

The provided codebase implements the BrainTrust application for hiring freelance professionals and the reward token integration that connects to an Ethereum token smart contract.

How to read this Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending, Acknowledged** or **Resolved**. Informational notes do not have a status, since we consider them optional recommendations.

Note, that audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note, that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Summary of Findings

Token API Integration			
No	**Description**	**Severity**	**Status**
1	Decimal places inconsistency between deposit and withdrawals	Minor	Pending
2	Number of blocks required for deposits to be confirmed is relatively small	Minor	Pending
3	Ethereum address fields can be longer than required	Informational	Pending
BrainTrust Application			
No	**Description**	**Severity**	**Status**
4	Leaked API keys in source code	Minor	Pending
5	Websocket connection can be initiated by anyone	Minor	Pending
6	backend/apps/common/zoom/client.py: Meeting duration always calculated as zero	Informational	Pending
7	backend/apps/pages/views.py: context key INVOICE_STATUS set twice	Informational	Pending

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	High	-
Level of Documentation	Medium	-
Test Coverage	High	-

Detailed Findings

Token Integration API

1. Decimal places inconsistency between deposit and withdrawals

Severity: Minor

Amounts are interpreted differently in deposit and withdrawal operations. The underlying token contract uses 18 decimal places. However, deposit and withdrawal requests use 0 and 18 decimal places for the amounts specified. This may lead to confusion for integrators which may cause bugs.

In addition, the total amount is limited to 50 digits, whilst the underlying token contract technically allows for 78 digit numbers (in decimal representation). This is unlikely to cause any issue in practice since such large numbers will not occur.

Recommendation

Consider using the same number of decimals in the API requests to avoid confusion.

2. Number of blocks required for deposits to be confirmed is relatively small

Severity: Minor

The smart contract integration confirms events through event filtering transfer events. The constant `REQUIRED_NUMBER_OF_BLOCK_CONFIRMATIONS` is used to configure a limit of block confirmations that need to have occurred before such a deposit is considered confirmed. By default this is set to 6 block confirmations, This number is relatively small and corresponds to the number of confirmations usually used in Bitcoin. In Ethereum higher numbers are usually recommended, due to the lower block time. Major exchanges require 20 or even 50 blocks.

Recommendation

Consider using at least 10 or even 20 block confirmations.

Status: Pending

3. Ethereum address fields can be longer than required

Severity: Informational

The character limit specified for Ethereum addresses in `ethereum/models` is 200 characters. However, an Ethereum address expressed as a string in its standard format is 42 characters long (including leading ‘0x). Whilst this is not a security list, allowing the API to store and receive longer addresses could cause confusion or make the system more error-prone by allowing extra data to be received.

Recommendation

Consider reducing the character limit for Ethereum addresses to 42 characters to enforce standard representation.

Status: Pending

Braintrust Application

4. Leaked API keys in source code

Severity: Minor

Several files are found to contain API keys and secrets, namely:

- *.gitlab/gitlab_runner/runner1-config/config-template.toml*
- *avbox/app/routes.py*
- *backend/config/settings/common.py*
- *backend/config/settings/dev.py*

Depending on the API key usage and validity, a malicious user can abuse a leaked API key to cause damage to the overall project.

Recommendation

Consider verifying that the API keys are deactivated and rotated.

Status: Pending

5. Websocket connection can be initiated by anyone

Severity: Minor

When initiating a WebSocket connection, there is no CSRF protection implemented. A malicious user can start a WebSocket connection and send arbitrary WebSocket messages as the victim. The impact of this issue depends on the implementation of the WebSocket itself. This is unlikely to be exploited since the SameSite Cookies which are set to Lax are preventing it. However, there's still a risk for users that use outdated browsers.

Recommendation

Consider validating the Origin header when initiating a WebSocket connection.

Status: Pending

6. `backend/apps/common/zoom/client.py`: Meeting duration always calculated as zero

Severity: Informational

In line 61 the following expression is used to calculate a meeting duration:

```
duration = event.time_slot.end_time - event.time_slot.end_time
```

This will always result in zero.

Recommendation

Subtract the start time from the end time.

Status: Pending

7. `backend/apps/pages/views.py`: context key `INVOICE_STATUS` set twice

Severity: Informational

The field is set twice (line 157 and line 188).

Recommendation

Remove duplicate code.

Status: Pending