Coding Challenge

You are tasked with implementing an application that authorizes a transaction for a specific account following a set of predefined rules.

Packaging

Instructions on how to build and run your application must be present on a Readme file. Along with relevant a brief description on code design choices.
Building and running the application must be possible under Unix or Mac operating systems.
Dockerized builds are welcome.
Your program is going to be provided a json line as input and should provide a json line as output for each call to the authorization function.

Sample usage

$ cat operations

{ "transaction": { "id": 1, "consumer_id": 10, "score": 600, "income": 4000, "requested_value": 10000, "installments": 15, "time": "2019-02-13T10:00:00.000Z"}}
{ "transaction": { "id": 2, "consumer_id": 10, "score": 100, "income": 4000, "requested_value": 10000, "installments": 15, "time": "2019-03-13T10:00:00.000Z"}}
{ "transaction": { "id": 3, "consumer_id": 10, "score": 500, "income": 4000, "requested_value": 10000, "installments": 0, "time": "2019-04-13T10:00:00.000Z"}}

$ authorize < operations

{ "transaction": { "id": 1, "violatios": ["compromised-income"] }}
{ "transaction": { "id": 2, "violatios": ["low-score"] }}
{ "transaction": { "id": 3, "violatios": ["minimum-installments"] }}

State

The program should not rely on any external database. Internal state should be handled by an explicit in-memory structure. State is to be reset at application start.

Operations

The program handles just one operation:

1. Credit Transaction

Transaction authorization Input

Tries to authorize a transaction for a particular consumer, a requested value and profile for the account's state and last authorized transactions.
Output
The transaction id + any business logic violations.

Business logic violations
You should implement the following rules, keeping in mind new rules will appear in the future:

- When the value of installmets exceeds 30% of income: compromised-income
- When the score is lower than 200: low-score
- When the istallments value is lower than 6: minimum-installments
- When happens two transactions in the same 2 minutes: doubled-transactions

Examples
Given a score lower than 200:
input:
{ "transaction": { "id": 2, "consumer_id": 10, "score": 100, "income": 4000, "requested_value": 10000, "installments": 15, "time": "2019-03-13T10:00:00.000Z"}}
output:
{ "transaction": { "id": 2, "violations": ["low-score"] }}

Error handling

Our expectations
We at Serasa value simple, elegant, and working code. This exercise should reflect your understanding of it.
Your solution is expected to be production quality, maintainable and extensible. Hence, we will look for:

- Quality unit and integration tests;
- Documentation where needed;

- Instructions to run the code.
  General notes
- Please keep your test anonymous, paying attention to:
  the code itself, including tests and namespaces;
  version control author information;
  automatic comments your development environment may add.