eng.bce_events_flow

Alexandr Kirilov (https://github.com/alexandrkirilov)

Blockchain example. Events flow implementation.

This example came from the real projects that been developing in last 5 years and based on article "Event Driven". Need to be noticed that this article is only architectural description, there is nothing about application realisation. The releasing version of block chained application depend on huge list of factors:

- What kind of DB Engine you are going to use? SQL or noSQL?
- What exactly DB Engine you are going to use? MySQL, Oracle, Postgre, CouchDB or any other?
- What kind of language you are about to use for developing your application and what kind of resource language platform might to offer to you in this case?
- What kind of OS you are going to use? Windows, Linux-based, FreeBSD-based or something special?
- And other ...

The global question that need to be answered: what and against whom you are going to defend? You have to define "the idea" of the application.

There is given issue

The events flow that is under term "event" containing different kind of data types and structures that is strictly positioned on timeline within strict fixation of events, where the time-related events - the chain that has to be blocked. The need to be excluded ability to change the time or body of any event that is stored in DB by. The examples of event flow that might be covered by this solution:

- any kind of system logbook
- personal profile of the student, where mentioning score of examinations, passing certification and etc
- personal card of any patient in any medical facility, where diagnosis determining - event within critical value time, arriving to hospital - event, releasing from hospital - event and other
- there might be any kind of event-driven system

Objective

To avoid ability manipulate the events timeline by changing time of event or changing the body of event for whole life-line of storing data about any events in a flow.

Solution

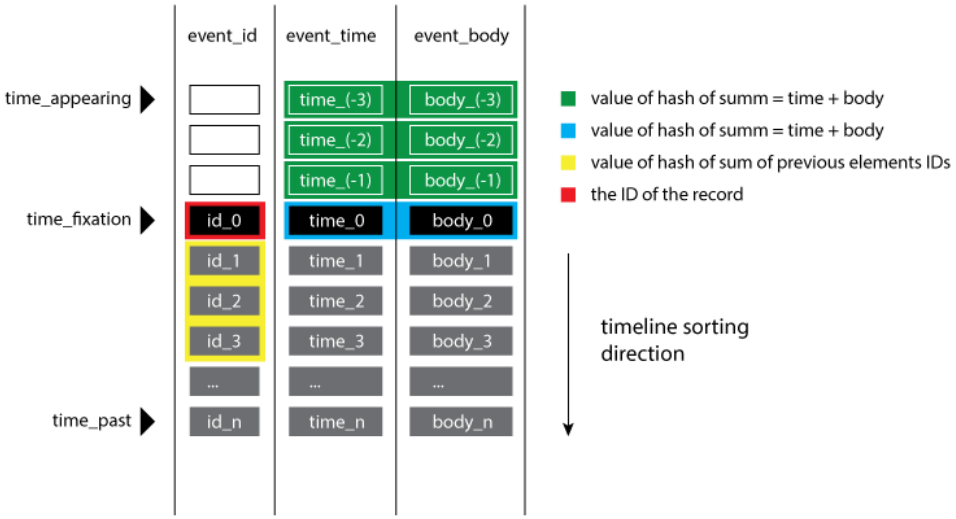
First of all need to clarify the key-points for defending against fraud or manipulation. Need to define the "chain" that need to "block" (based on "3W" event model):

- When? - the time of event. The position on timeline is very critical. We have to fix it and avoid changing it.
- What? and Where? - the body of event. We are going to fix it entirely. There standalone parts of event do not have any sense in case of storing event-body.

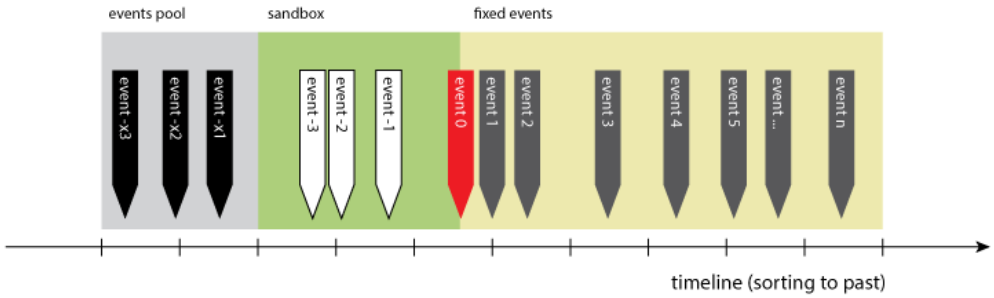
The next step - is defining what is unique for every key-point:

- for the time - most likely it will positive integer of UNIX-timestamp
- for the event-body going to be md5 (for example) from event-body
- the unique for entire event the md5 from sum of time and body

The main solution for blocking this chain of events is the manner of generating ID for storing it in DB



The unique ID for event is result of hash from sum ID's from sum of 3 previous record and the sum of 3 future events unique values. This kind of solution is making almost impossible to change any piece of data without attention from security side. Any attempt of changing, at least will rise huge load for servers that might be triggered by any simple monitoring system.



The next important point of this approach - method of writing and fixing data in DB. It is containing 3 part:

- event pool: the place of appearing of any event with time limited ability to change the body or delete it from flow
- sand-box: the specially defined positions in the table where you are going to store events, the events written to DB but not fixed or fixed by special kind of ID
- fixed-events: the events that already been blocked in the chain by described algorithm

In case of using BSD License based DB engine, you might to improve the DB engine and add the trigger for maximum time for transaction and if time exited it will drop attempt of changing out.

This solution do not totally excluding the ability of fraud. It's getting you more time for acting against it in case of necessity.

Follow author updates on **Linkedin**

Follow AR|BO|RE|US updates on **Twitter** and **Linkedin**