Data Engineering Challenge - Crowdsource Karma Points

Description

Detectify offers the capability for customers to scan their websites for vulnerabilities.
This is done in an automated fashion by a function referred to as The scanner which scans websites for different vulnerabilities.

Some of these vulnerabilities are crowdsourced and published to Detectify by external security researchers aka. crowdsourcers.
To keep the crowdsourcers engaged, they receive a compensation (a payout) in the form of karma points every time one of their vulnerabilities is found by the scanner.

The publishing and tracking of these payouts is managed through a portal available to the crowdsourcers.
Currently, the portal is under development and one of the requested features is an analytical tool where crowdsourcers can see how many karma points their vulnerabilities have accumulated aggregated over vulnerability severity levels.

The data side of this feature is the basis for this challenge.

Challenge

The solution for the challenge should be:

1. A data pipeline that produces the requested data set. Schema for the data set is found under section *Data input*. The pipeline should be packaged in a way that requires maximum setup time of 5 minutes. The abstraction level for the pipeline logic should be code, other than that, feel free to use any language/infrastructure.

2. A documentation on how to run the pipeline along with answers to the *Discussion points* below. The challenge might contain uncertainties - make sure that all assumptions are well documented.

The challenge should not take more than 8 hours - feel free to mock and simply document parts of the pipeline if time is running out.
If you have any questions related to the challenge, don't hesitate to reach out to your recruiter.

Discussion points

- How does your solution handle outages in its different parts?
- How does your solution scale with a large increase in input?
- How would the feature request along the lines of: "*I want to see the distribution as it was last wednesday*" affect the solution?
- How would the feature request of a *karma_average* field in the output data set affect the solution?
- What are the trade offs with a denormalized output data set in the case of this challenge?

Data input

The four data sources for this challenges are described below.

Finding

Description:
This data set contains the actual vulnerability findigs that is emitted from the scanner.
The events are streamed to a file and simply holds the time of the finding along with the id of the found vulnerability.

Type:
Real-time streamed data.
The stream of data is mocked and generated by running:

```
cd $PROJ_ROOT/log_gen && python start_finding_stream.py <INT|FLOAT>
```

where the first and only argument option specifies the time scaling of the data generation as "length" of a day in seconds.

Location:
input/finding.csv (empty)

Schema:
id,time,vuln_id
1,2019-02-01 00:00:00,85
2,2019-02-01 00:55:37,36
3,2019-02-01 00:55:37,72

Vulnerability

Description:
This data set holds all the current vulnerabilities that are active in the scanner and keeps track of its severity and its publisher (crowdsourcer).

Type:
Materialized view based off production master data.

Location:
input/vulnerability.csv

Schema:
id,cs_id,sev_id
1,15,2
2,10,4
3,4,4

Crowdsourcer

Description:
This data set contains the crowdsourcers that have published vulnerabilities to us.
This source exist purly for denormalization purposes and only holds a id->name mapping of the crowdsourcer.

Type:
Materialized view based off production master data.

Location:
input/crowdsourcer.csv

Schema:
id,name
1,0BL1V10N
2,F474 M0R64N4
3,4N0M4LY

Severity

Description:
This data set contains the crowdsourcers that have published vulnerabilities to us.
This source exist purely for denormalization purposes and simply holds an id->name mapping of the crowdsourcer.

Type:
Materialized view based off production master data.

Location:
input/severity.csv

Schema:
id,severity,karma
1,minor,50
2,medium,100
3,high,500

Data output

Crowdsourcer_karma_distribution

Description:
This data set contains accumulating karma points statistics of crowdsourcers.
The data set should always hold a correct model of the karma statistics due to it being streamed in real time.

The aggregations in the data set is accumulated karma per severity level. I.e, the field minor_sum should hold the sum of all karma accumulated from minor vulnerabilities for the given crowdsourcer.

Type:
An chronologically accumulating data set in a suitable format.

Location:
Somewhere nice

Schema:
cs_id,cs_name,minor_sum,medium_sum,high_sum,critical_sum

Send in

Use the structure below for the documentation and source code.

```
.
├── input
│   ├── crowdsourcer.csv
│   ├── finding.csv
│   ├── severity.csv
│   └── vulnerability.csv
├── log_gen
│   └── start_finding_stream.py
├── README.pdf
└── solution
    ├── docs
    │   └── PUT_DOCUMENTATION_HERE
    └── src
        └── PUT_SOURCE_HERE
```

When sending in, please do not use a publically reachable version control (e.g github). Instead archive the tree above, name the archive de_challenge_firstname_lastname and send in over email.

Good luck!