**Project X**

**Question 1: In the above examples how to determine anomaly?**

The anomaly detection in this new project can be done using this proposed data model. This model operates in two modes, first mode is in training mode and second mode is in detection mode. The input to this model is the following values url’s, list of query parameter and its value (example: list of query parameters = [(q1, v1), (q2, v2) …]), list of headers and its value, signal type. Mentioned values will be extracted from each signal, which we receive from our data sources.

First, we create set of procedures to evaluate the query parameter and each procedure assigns a probability to the query parameter. If the parameter probability is low then it’s an anomaly. For each signal, we compute an anomaly score using weighted sum equation as shown below.

In the above equation ‘weight’ represents the weight we assign to the procedure, then we compute (1-prob of a parameter resulted from a procedure). In the above equation, we subtract “probability of a query parameter” from one because probability values which are close to 0 are abnormal so it should be treated with high priority. During training phase, we compute the threshold probabilities for each parameter and for each signal.

Now let’s consider small set of procedures we will be creating:

* Query parameter length, in web requests most of the parameters are fixed length like session id, tokens or short strings from user input. This procedure mostly concerned with the length of the parameters.
* Query parameter character distribution, unusual long parameters or repetitions of group of characters.
* In web requests the order of parameters received are maintained because typically client programs mention query parameters in specific order in requests, this procedure mostly concerned about the order of query parameters mentioned in the request.
* This procedure uses tokens, session id to validate the data source and assign probabilities.
* Procedure to validate uri’s received.
* Procedure to validate IP, port numbers, protocols.
* Procedure which checks for number of signals received in time interval for each signal type. If we detect any signal received more than normal frequency we assign low probability. So that anomaly score will be more for this type of signal.

Once the training phase is done, the model is ready to be put in production to detect anomaly. Using anomaly score we create one of these three type of alerts informational, warning and critical.

Typical anomaly detection system will check subset of query parameters or fields in the request and create alerts, which results in more number of alerts. Proposed model try to reduce number of alerts generated and try to detect anomalies in most cases. Additionally, new model is flexible to add new procedures in future.

**Question 2: Do any events raise concerns? If so why?**

Following are the issues I found during signal inspections.

In signal alpha 1 the query parameter “ctu” value is an url, but that url does not follow any syntax in the query parameters. I mean ctu query parameter not encoded properly and there are not proper delimeters for query parameters in this url. There are two possibilities in this case the url could be completely wrong or the application which creates an url could have some issues.

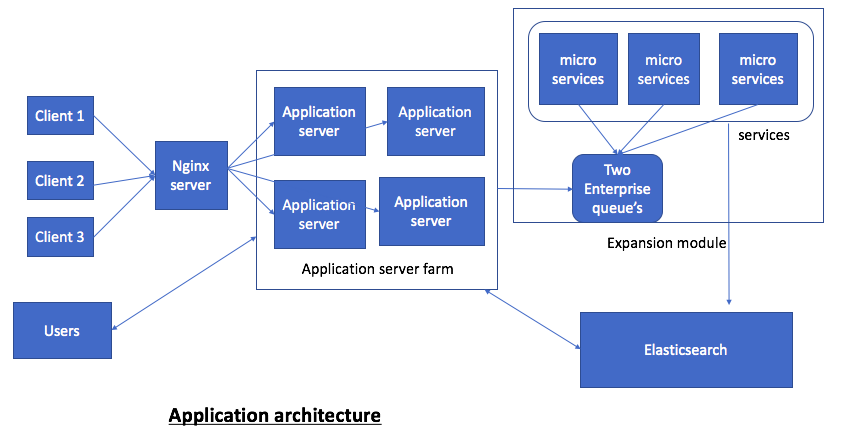
In signal “alpha2” the domain name mentioned as <http://testuser:testpassword@pqz.com>, I think the data source didn’t correctly create the url in this case as well.

In signal beta the format of the message is similar to json with key value separated using ‘=>’ delimiter. In this case as well, the client could use ‘=>’ to delimit key value pairs, if its not configured to create this format the client could have some issues as well.

**Question 3: How would and what would you normalize in the events above?**

The proposed data model considers all parameters before creating an alert, in contrast with traditional systems which raise alert even any one parameter is considered as an anomaly. This approach greatly reduces number of alerts generated. Additionally, we can suppress the number of alerts created by maintaining number of alerts of specific type in an interval, so at the end of an interval we can create only one alert by suppressing multiple alerts.

**Application Architecture:**

****

**Question 4 How would you go about building project x?**

Above architecture is the proposed architecture diagram for project x. First, we will expose api endpoints, which will accept signals from multiple data sources. The api endpoints after accepting the request, it creates a new packet with necessary values like urls, query parameters, key:value pairs from request. Once the packet is created, it assigns id for it and put that packet in enterprise queue (**RabbitMQ or Kafka**). There will be multiple micro services which accept packets from the queue, process the packet and create corresponding alerts in the elasticsearch.

**Question 5 What technologies would you use and why?**

API endpoints are exposed using Nginx, Flask and python as the backend programming language. Nginx is used for load balancing between flask servers. For enterprise queues, we could use either RabbitMQ or Kafka, both are open source, reliable, scalable and delivers high performance. Micro services are created using python and they interact with enterprise queues and elasticsearch. Elasticsearch(ES) is used in this project because we need to perform analytics on alerts created as well, ES provides full text search capabilities and it is based on popular information retrieval library lucene. Additionally, we can even store the signals we receive as json in ES, performing analysis on those documets would be easy and fast. ES is highly scalable.

**Question 6 How do you address the expectations of the product owners? Customers?**

The architecture of this project is designed in such way to support growing business needs or to withstand number of signals need to be processed in future. In our project to handle more signals from multiple clients we can add multiple application servers. Additionally, we have an expansion module, which contains enterprise queue and micro services. When number of signals need to be processed grows we can add expansion modules to process those new set of signals. Our architecture fully supports the requirements from product owners.

From the customers point of view, we need to detect anomalies respective to the applications and provide insights about how applications are performing. In our data model, we can add procedures which handle those specific requests to create those alerts.