UNIVERSITE DE YAOUNDE I
****
ECOLE NATIONALE SUPERIEURE POLYTECHNIQUE
****
DEPARTEMENT DE GENIE INFORMATIQUE

UNIVERSITY OF YAOUNDE I
****
NATIONAL ADVANCED SCHOOL OF ENGINEERING
****
DEPARTMENT OF COMPUTER ENGINEERING

Mise en place d'un dispositif de détection des fraudeurs à la Simbox au Cameroun

Rapport du Projet de Systèmes Distribués et Virtualisation

Membres du groupe :

Djizanne Toukem Joel
Fotsa Jounda Boris
Manfouo Kennedy Armel
Tchouatcha Deumaga Michel
Yuhala Peterson Jr. Dohbila (C)

Sous la supervision de :

Alain Tchana., PhD.

Année academique 2017-2018

Sigles et Abréviations

BTS Base Transceiver Station

CDR Call Detail Record

ELK Elasticsearch Logstash Kibana

IMEI International Mobile Equipment Identity

IMSI International Mobile Subscriber Identity

IP Internet Protocol

REGEX Regular Expression

GLOSSAIRE

CDR Un enregistrement produit par un équipement de télécommunication contenant les détails concernant une transaction de télécommunication.

Docker Un système de virtualisation qui permet d'embarquer une application dans un container virtuel qui pourrait s'executer sur n'importe quelle machine.

Pile ELK Une pile logicielle composée de Elasticsearch, Logstash et Kibana.

Simbox Un équipement qui transmet les appels par l'internet grâce au protocole IP.

ABSTRACT

Simbox fraud, or call forwarding, is a widespread phenomenon today in Africa and Asia, in which fraudsters trick subscribers into call forwarding their numbers to long distance numbers routed over the internet, so as to exploit the low retail call prices on the Internet.

This report presents the work done in the course of a Distributed Systems and Virtualization project, and the objective of the project is to develop an application which will enable telecommunication operators to detect SIMbox fraudsters.

Our solution is based on the ELK Stack, which is a very powerful open source tool for data analysis and visualization. The final version of the application will be deployed on Docker Hub.

Key Words : SIMbox, Fraudster, ELK Stack, Distributed Systems and Virtualization, Docker.

RÉSUMÉ

La fraude à la simbox, ou bypass téléphonique, est une phénomème très répandue aujourd'hui en Afrique et en Asie, dans lequel les fraudeurs contournent efficacement les points de recharge interurbains, pour exploiter le faible prix au détail des appels sur le réseau internet.

Le présent rapport a été rédigé dans le cadre d'un projet de Systèmes Distribués et Virtualisation et l'objectif du projet est de mettre en oeuvre un dispositif qui permet aux opérateurs de détecter les fraudeurs.

Notre solution est basée sur la Pile ELK, qui est un outil d'analyse et de visualisation de données, open source et très puissant. Le dispositif final sera déployé sur Docker Hub.

Mots-clés : Simbox, Fraudeur, Pile ELK, Système Distribués et Virtualisation, Docker.

TABLE DES FIGURES

SOMMAIRE

INTRODUCTION GÉNÉRALE

Contexte

D'après Cameroon-Info.net, entre Janvier et Août 2015, plus de 65 millions de minutes d'appels téléphoniques ont été détournées au Cameroun par des fraudeurs exploitant des boîtes à outils Simbox, faisant perdre à l'État et aux opérateurs de télécoms plus de 13 milliards de FCFA sur cette période. Cette pratique, connue sous le nom de fraude à la Simbox ou bypass téléphonique, est courante en Afrique et en Asie et consiste à faire passer les appels internationaux pour des appels locaux. Il est question dans ce projet de mettre sur pied un dispositif qui permet aux opérateurs de détecter les fraudeurs.

Objectif général du projet

Ce projet a pour objectif principal la mise en pratique par les étudiants de toutes les notions vues en cours de Systèmes distribués et Big Data. Il les confronte à un problème concret, celui de la fraude à la Simbox ici, qu'ils doivent résoudre en exploitant le Cloud et les technologies du Big Data.

Objectifs spécifiques

1. Etudier le comportement des utilisateurs et dresser les différents profils
2. Générer des CDR et un fichier de recharge simulant les comportements des différents profils et les prétraiter,
3. Déterminer les critères pour détecter les appels suspects,
4. Exploiter le Cloud pour optimiser le temps de calcul,
5. Exploiter les frameworks de Big Data pour analyser les données
6. Permettre la visualisation des résultats.

Périmètre de travail

L'outil de détection de fraude que nous allons développer sera basé sur le ELK Cluster, déployé en local sur nos machines physiques. Nous basons nos études dans le contexte camerounais.

Contraintes techniques

Pour l'implémentation de la solution, la pile ELK sera utilisée. Le dispositif final sera déployé dans un docker installé et mis dans une repository Docker Hub.

Pour simplifier le problème, on suppose qu'on veut détecter des fraudeurs « naïfs », c'est-à-dire, des fraudeurs qui utilisent des Simbox dans un emplacement fixe, qui ne changent pas d'IMEI et qui ne reçoivent jamais ni appel, ni sms, et qui ne surfent pas. Quant aux autres profils, on va considérer les suivants :

- Les jeunes de la ville
- Les jeunes de la campagne
- Les vieux de la ville
- Les vieux de la campagne
- Les entreprises
- Les callbox

Plan du Rapport

La suite de ce document est structurée comme suit :

- **Chapitre 1 : Description globale du système.** Ce chapitre présente une description globale du système à réaliser. Il présente une architecture globale du système ainsi qu'une description de chaque composant.
- **Chapitre 2 : Mise en place du système.** Ce chapitre est dédié à la présentation des différentes étapes de la mise en oeuvre de notre outil. Il présente les configurations faites dans le système ELK, le système de génération des CDRs ainsi que l'analyse des données générées.
- **Chapitre 3 : Manuel d'utilisation.** Ce chapitre est dédié à la présentation du manuel d'utilisation du dispositif. Il explique de façon assez claire les différents étapes pour déployer le dispositif sur une machine linux.

Nous terminons ce document par une conclusion présentant un bilan du travail effectué, les difficultés rencontrées, les limites liées à la solution ainsi que les perspectives qui en découlent.

CHAPITRE 1

DESCRIPTION GLOBALE DU SYSTÈME

Ce chapitre présente une description globale du système à réaliser. Il présente une architecture globale du système ainsi qu'une description de chaque composant. Ce chapitre est structuré comme suit :

Contents

1.1 Architecture globale du dispositif

Notre système est basé sur la pile ELK . Le système est donc constitué de ElasticSearch, Logstash et Kibana.[3]

1. **ElasticSearch** : moteur de stockage et d'indexation de documents et moteur de requêtes/-d'analyse de celles-ci.
2. **Logstash** : utilisé pour l'analyse, le filtrage et le découpage des logs pour les transformer en documents, parfaitement formatés notamment pour ElasticSearch.
3. **Kibana** : dashboard interactif et paramétrable permettant de visualiser les données stockées dans ElasticSearch.

Chaque composant de la Pile ELK a été mit dans une image docker que nous avons utilisée en locale. Le serveur de génération de CDRs est une machine Ubuntu. NB : Chaque image crée un container docker au lancement. Nous avons donc 4 containers en total, serveur de génération y compris. La figure suivante illustre l'architecture du dispositif :

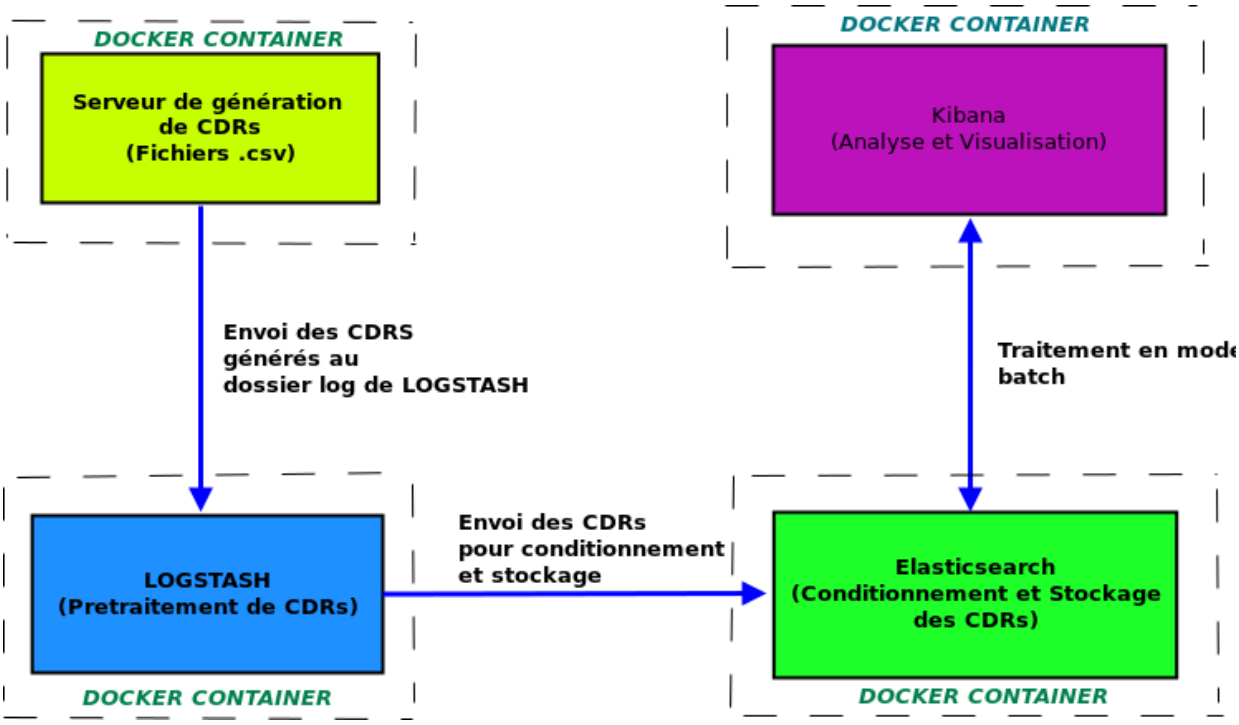

FIGURE 1.1: Architecture globale du système

1.2 Description des composants

1.2.1 Données CDR

Définition 1.2.1. *Un CDR (Call Detail Record) est un enregistrement produit par une équipement de télécommunication contenant les détails concernant une transaction télécom, par exemple un appel ou un message texte. L'enregistrement contient des attributs comme l'heure de l'appel, la durée, le numéro appélant, le numéro destinataire etc.*

Les CDRs générés seront de 03 types : appel vocal (voice), SMS, et surf et comprendront les informations suivants :

- IdCDR
- Date début
- Type
- Taille
- Numéro appelant
- Numéro destinataire
- Id appareil (IMEI)
- Id puce (IMSI)
- Opérateur
- Etat du BTS (en mouvement ou statique)

Etant donné que nous travaillons dans le contexte camerounais, nos opérateurs seront limités aux 04 opérateurs principals au Cameroun :

- MTN
- Orange
- Camtel
- Nextell

Nous avons choisi d'utiliser les CDRs dans le format csv car ils sont plus faciles à manipuler.

1.2.2 Docker

Docker est une plateforme qui va vous permettre d'exécuter votre code à l'intérieur d'un conteneur indépendamment de la machine sur laquelle vous êtes ! Un conteneur ressemble à une machine virtuelle sauf qu'il n'embarque pas tout un système d'exploitation avec lui, ce qui lui permet de s'exécuter en quelque secondes et d'être beaucoup plus léger.

Docker peut donc résoudre notre problème d'environnement, car quelle que soit la machine que nous utiliserons, le code s'exécutera de la même manière.

La plateforme Docker est composée de deux éléments :[2]

- Le démon Docker qui s'exécute en arrière-plan et qui s'occupe de gérer vos conteneurs
- Le client Docker qui vous permet d'interagir avec le démon par l'intermédiaire d'un outil en ligne de commande

La figure ci-dessous resume le système docker.

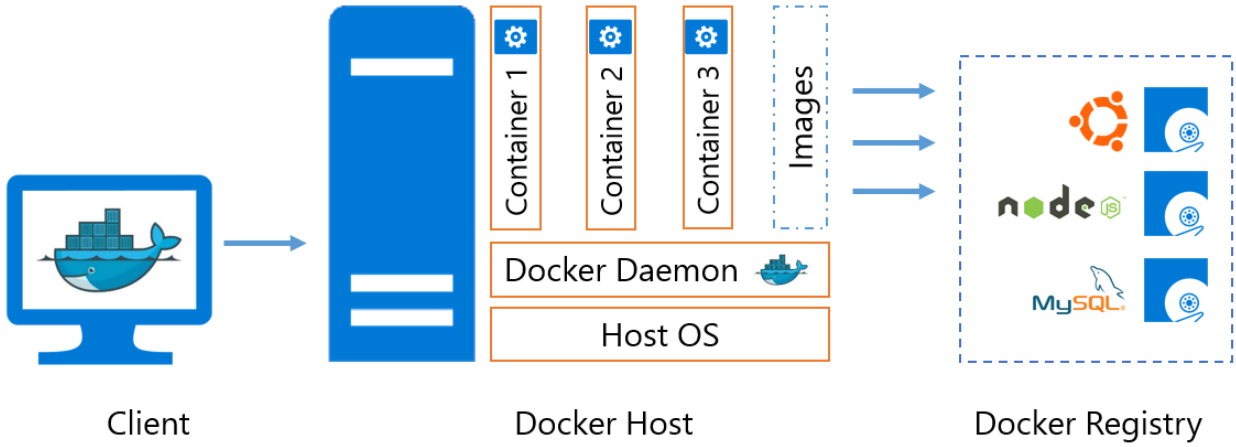

FIGURE 1.2: Système docker

Notre dispositif final sera déployé dans un container docker et mis sur le cloud.

CHAPITRE 2

MISE EN PLACE DU DISPOSITIF

Ce chapitre est dédié à la présentation des différents étapes de la mise en oeuvre de notre outil. Il présente les configurations faites dans le système ELK, le système de génération des CDRs ainsi que l'analyse des données générées.. Le chapitre se structure comme suit :

Contents

2.1 Génération des CDRs

2.1.1 Dressage de profils

Avant de générer les CDRs, nous avons pris la peine d'étudier le comportement des utilisateurs dans le contexte camerounais afin de faire un dressage de profils.

Après une étude preliminaire, nous avons sortis avec le tableau suivant qui résume le comportement des différents profils étudiés.

profils / paramètres	Jeunes de la ville	Jeunes de la campagne	Vieux de la ville	Vieux de la campagne	entreprise	call box	fraudeurs
Durée E moyenne/j	0 – 10 min	0 -10 min	5 – 20 min	5 – 20 min	0 – 5 min	> 20 min	> 30 min
Durée R moyenne/j	0 – 10 min	0 – 10 min	5 – 20 min	5 – 20 min	> 30 min	0 – 5 min	0
Tranche Horaire d'appels	18h – 00h	18h – 00h	7h – 21h	7h – 20 h	7h – 18h	7h – 22h	24 h /24 h
Nbre de SMS émis/j	> 30	> 30	0 - 10	0 - 5	0 – 10	0 – 5	0
Nbre de Mo /j	> 100	50	50	0	>2000	0	0
Mobilité	Oui	Oui	Oui	Oui	Non	Non	Non
Recharge mensuel moyenne	5000-10000	500 - 5000	5000-10000	300-5000	> 20000	> 15000	> 20000

FIGURE 2.1: Profils

Il est important de noter que ces résultats ne sont pas exacts et contiennent quelques approximations. Néamoins, ce tableau reflète la réalité d'une façon générale.

2.1.2 Processus de génération des CDRs

Pour une bonne simulation de la réalité et pour produire des résultats valables et raisonnables, il est primordiale d'avoir des données CDRs réalistes.

Nous avons donc mis en place un outil de génération de CDRs développé en Python. Ensuite, nous avons utilisé quelques scripts bash pour un traitement préliminaire de fichiers générés. L'outil de génération des CDRs est basé sur un outil opensource appelé **csvgen**. Cet outil peut être utilisé pour générer plusieurs types de données mais nous l'avons configuré pour répondre à nos besoins.

Pour générer des CDRs avec csvgen, nous créons d'abord un fichier de description dans lequel nous définissons les characteristiques d'un CDR. Le fichier suivant présente un exemple d'un fichier de description pour générer un type spécifique de CDRs.

```
row_number 180000 cdr_mtn_
date "01-01-2016 00:00:00" "12-31-2016 23:59:59" "%m-%d-%Y %H:%M:%S"
fixed voice
number 1 15 0
regex (65[0-4](^[0-9]{6}))|(67(^[0-9]{7}))
regex (2(^[0-9]{8}))|((65[5-9](^[0-9]{6}))|(69(^[0-9]{7})))|((65[0-4](^[0-9]{6}))
fixed mtn
regex ^[0-9]{15}
regex ^[0-9]{15}
number 0 1 0
number 5000 15000 0
```

- La première ligne décrit l'ID du CDR. Nous spécifions que l'ID aura une préfixe "cdr_mtn" et le compteur commencera au numero 180000
- La ligne suivante spécifie l'intervalle dans lequel la date du CDR sera sélectionnée, et le format de la date
- Ensuite, nous spécifions que le type du CDR est un appel (ie de type voice)
- Les deux lignes suivantes spécifient le numéro appelant et le numéro destinataire respectivement. Comme vous pouvez le constater, nous utilisons les expressions régulières. Ainsi, les numéros générés sont cohérents à 100 pourcent avec la convention de numération téléphonique camerounaise. Dans cet exemple, le numéro appelant est un numéro MTN et les destinataires sont soit MTN, Orange, Camtel ou Nextel
- La ligne suivante spécifie l'opérateur auquel les CDRs appartiennent.
- Dans les deux prochaines lignes, nous utilisons les regex (expressions regulières) pour générer l'IMEI et l'IMSI, qui sont les entiers à 15 chiffres de façon général.
- Ensuite, nous spécifions un booléen 0 ou 1 spécifiant si le BTS est statique ou en mouvement
- La dernière ligne spécifie la valeur dans le fichier de recharge correspondant au numéro appelant

Pour générer les CDRs à partir de ce fichier de description, nous utilisons la commande suivante :

```
python csvgen.py -i 10000 -o cdr.csv description.in
```

- L'option -i spécifie le nombre de CDRs à générer ie 10000
- L'option -o spécifie le nom du fichier dans lequel les CDRs seront générés.
- Le dernier argument représente le nom du fichier de description.

Nos fichiers de description pour la génération des CDRs dans le cadre de ce projet sont dans trois groupes : CDRs de type Voice, CDRs de type Surf et CDRs de types SMS. Nous avons généré ces différentes catégories pour les différents opérateurs tout en prenant en compte les réalités du terrain, c'est-à-dire, les pourcentages d'abonnés pour chaque opérateur.

Pour faciliter la tâche de génération, nous avons écrit un script bash dans lequel nous avons spécificié les commandes de génération et mergé les différents fichiers .csv générés en un seul fichier final.

Le nombre total de CDRs générés est de 310000, un nombre que nous trouvons suffisament grand pour faire nos analyses. NB : Après chaque lancement, le serveur de génération de CDRs génère quelques CDRs et les envoi dans le dossier `/var/log` du container Logstash. Vous pouvez facilement indexer ces fichiers .csv dans le container Elasticsearch à l'aide d'un fichier de configuration logstash.

Le code pour le générateur de CDRs se trouve dans le dossier : `elkdocker/cdr_generator/cdrGen`.
Pour tester la génération sur une machine linux, lancez le script `generator.sh`. Les CDRs générés se trouvent dans le dossier data.

2.2 Création des images docker

Pour créer nos images docker nous avons utilisé le système `docker-compose` qui facilite la tâche de création et lancement des containers. Le fichier `docker-compose.yml` utilisé dans le cadre de ce projet est le suivant :

```
version: '2'
services:
cdrgenerator:
image: pyuhala01/cdr_gen
volumes: ['../docker/logstash_conf:/home/yuhala/cdrGen/cdrGen/data']

elasticsearch:
image: pyuhala01/elasticsearch
ports:
- "9200:9200"
- "9300:9300"
volumes:
- ../esdata:/usr/share/elasticsearch/data
- ../docker/elasticsearch_conf/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.y

logstash:
image: pyuhala01/logstash
ports:
- "5044:5044"
links:
- elasticsearch:9200
volumes:
- ../docker/logstash_conf/rech.conf:/etc/logstash/conf.d/rech.conf
- ../docker/logstash_conf/recharge.csv:/var/log/rech.csv
command: -f /etc/logstash/conf.d/rech.conf
kibana:
image: pyuhala01/kibana
ports:
- "5601:5601"
links:
- elasticsearch:9200
volumes:
- ../docker/kibana_conf/kibana.yml:/usr/share/kibana/config/kibana.yml
```

2.3 Insertion des CDRs dans ElasticSearch avec Logstash

Après la génération des CDRs, nous sommes passé au prochaine étape, ie l'insertion de ces données dans ElasticSearch. Le composant utilisé pour ceci est Logstash. Il prend les CDRs générés en entrée afin de les transformer et les parser pour ensuite les stocker dans ElasticSearch.

Pour insérer les données générés dans ElasticSearch en utilisant Logstash, nous utilisons des fichiers de configuration Logstash (ie fichiers **.conf**). Un fichier de configuration Logstash est composé de trois parties principales : **input, filter, output**. Le fichier de configuration que nous avons créé pour ce projet est le suivant :

```
input {
file{path => "/var/log/finalcdr.csv"
start_position => "beginning" }

}


filter {
csv{columns => ["IdCDR","Time","Type","Size","CallingNum","ReceivingNum",
"Operator","IMEI","IMSI","BTS_movement"] separator => ","}

mutate{convert => {"Size" => "integer"}}

date{match => ["Time", "MM-DD-YYYY HH:mm:ss"]
target => "Time"}

}

output {
stdout{codec=>rubydebug}
elasticsearch {
action => "index"
hosts => "http://localhost:9200"
index => "cdr-%{+YYYY.MM.dd}"
user => elastic
password => changeme

}
}
```

- L'**input** correspond à l'entrée, c'est-à-dire les fichiers de logs que l'on donne à Logstash. Dans notre cas, nous indiquons le chemin vers notre fichier .csv contenant nos CDRs générés. Le start_position indique où dans le fichier Logstash doit commencer sa lecture. Dans notre cas, elle commence au début du fichier.

  Dans un contexte réel, nous pouvons recevoir les fichiers de logs stockés sur d'autres serveurs si ces derniers sont configurés avec Filebeat.

- Le **filter** est composé de plusiers filtres : csv, mutate, date etc.
  - Le filtre `csv{}` indique que le fichier log est un fichier .csv et spécifie les noms de colonnes ainsi que le séparateur.
  - Le filtre `mutate{}` permet de convertir les strings en integer,float ou boolean. Dans notre cas, nous convertissons les tailles en integer.
  - Le filtre `date{}` permet de spécifier le champ d'un objet CDR qui sera utilisé pour le timestamp, et le format de date utilisé dans le CDR. Ceci facilitera la tâche de recherche en ElasticSearch.

- Le **output** permet de spécifier où les logs seront envoyés. Dans notre cas, nous les envoyons dans ElasticSearch pour stockage et stdout pour le debug.

Pour importer les données dans Elasticsearch via Logstash, nous utilison la commande suivante :

```
./logstash -f /etc/logstash/conf.d/cdr.conf
```

Cette commande lit le fichier de configuration logstash `cdr.conf` et insert les données CDRs dans le fichier .csv spécifié dans Elasticsearch. Chaque CDR est importé dans Elasticsearch sous forme d'un objet json comme le montre la figure suivante :

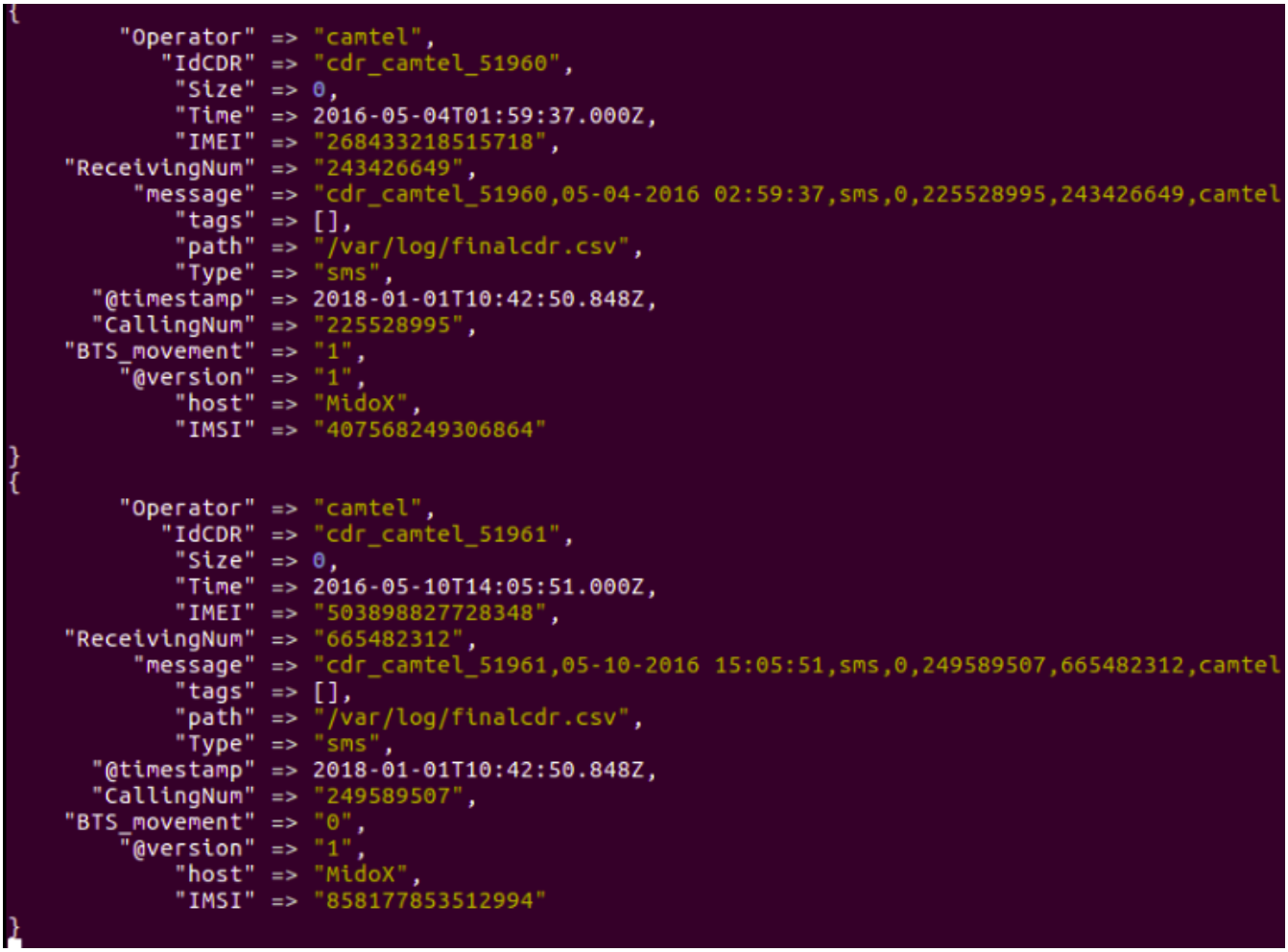

FIGURE 2.2: Les CDRs sous forme json

Après l'importation des CDRs dans Elasticsearch, l'index spécifié dans le fichier de configuration Logstash est créé dans Elasticsearch et nous pouvons visualiser nos CDRs dans Kibana.

2.4 Visualisation des CDRs dans Kibana

Pour analyser nos CDRs, nous devons d'abbord les indexer dans Kibana en créant un index ; Nous avons créé deux indexes `cdr-*` et `recharge-*` pour indexer les CDRs et le données liées au rechargement respectivement. Kibana utilise ces indexes pour facilement trouver les données dans Elasticsearch

2.5 Analyse de CDRs avec Kibana

Pour analyser nos CDRs, nous utilisons les requêtes Kibana dans la section **"Discover"**. Nous présentons ici quelques résultats des requêtes basiques[4] que nous avons faites dans Kibana :

1. **Afficher tous les données CDRs**

   La requête utilisée ici est : *

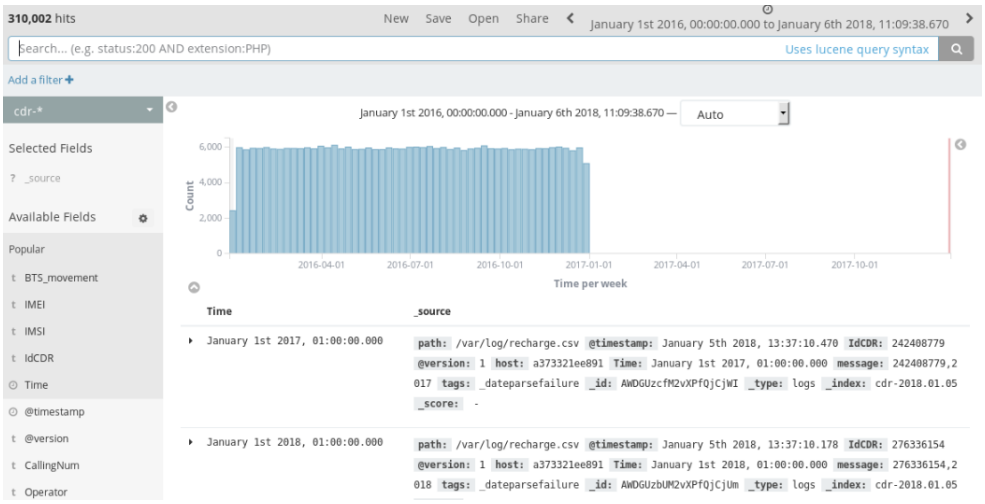

FIGURE 2.3: Ensemble de CDRs

Nous voyons bien que le résultat de cette requête retourne tous les 310002 CDRs. La date de chaque CDR est aussi indiqué ainsi que les valeurs de différents champs.

2. **Appels de très grande durée avec BTS qui ne change pas**

   La requête utilisée ici est : `Type:voice AND Size:>30 AND BTS_movement:0`

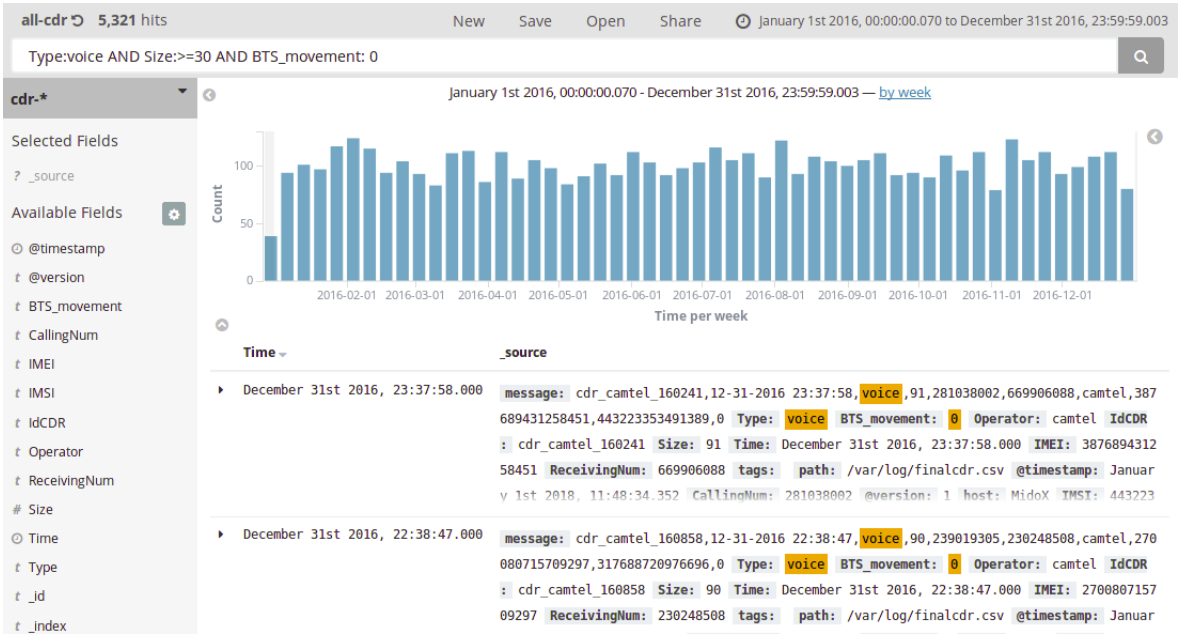

FIGURE 2.4: Appels suspects

2.6 Critères de détection des fraudeurs

De façon générale, on ne peut dire avec certitude si un CDR appartient à un fraudeur ou pas ; Néamoins, nous pouvons utiliser quelques critères pour détecter les CDRs suspects. Nos critères sont les suivants :

- Le temps moyen d'appels par IMEI.
- Nombre d'IMSI par IMEI.
- L'écart par rapport à la moyenne des durées d'appels par IMEI.
- Nombre d'appels total par numéro appelant.
- Le montant moyen de recharge par IMSI.
- Le mouvement de BTS moyen par appel d'un IMEI.
- La quantité moyenne d'unités surf par IMSI
- Le nombre de SMS envoyés par IMSI.

Pour le temps moyen d'appels, nombre d'IMSIs par IMEI, nombres total d'appels, l'écart type par rapport au montant moyen de recharge ou moyen des temps d'appel, les très grandes valeurs indiquent que ce profil est suspect.

Dans le cas contraire, un très petit nombre (par exemple 0) de sms envoyés par IMSI peut être vu comme un comportement suspect, dans le cas où ce IMSI envoi beaucoup d'appels et/ou ne surfe jamais.

NB : En utilisant ces critères, il devient un peut difficile de faire une distinction claire entre les fraudeurs et les call boxeurs, car ils se comportent presque de la même façon. Pour faire une distinction plus fine, on utilise plusieurs critères à la fois.[1]

Quelques visualisation liées au critères listés çi-déssus

Ces visualisations ont ete créées pour les 100 prémiers IMEIs uniques car au-delà, la puissance de la machine devient un problème.Les durées d'appel sont en minutes.

- **Durée totale d'appel par IMEI**
  Cette visualisation est faite sur les CDRs de type "voice" donc nous lançons d'abbord la requête : `Type:voice`

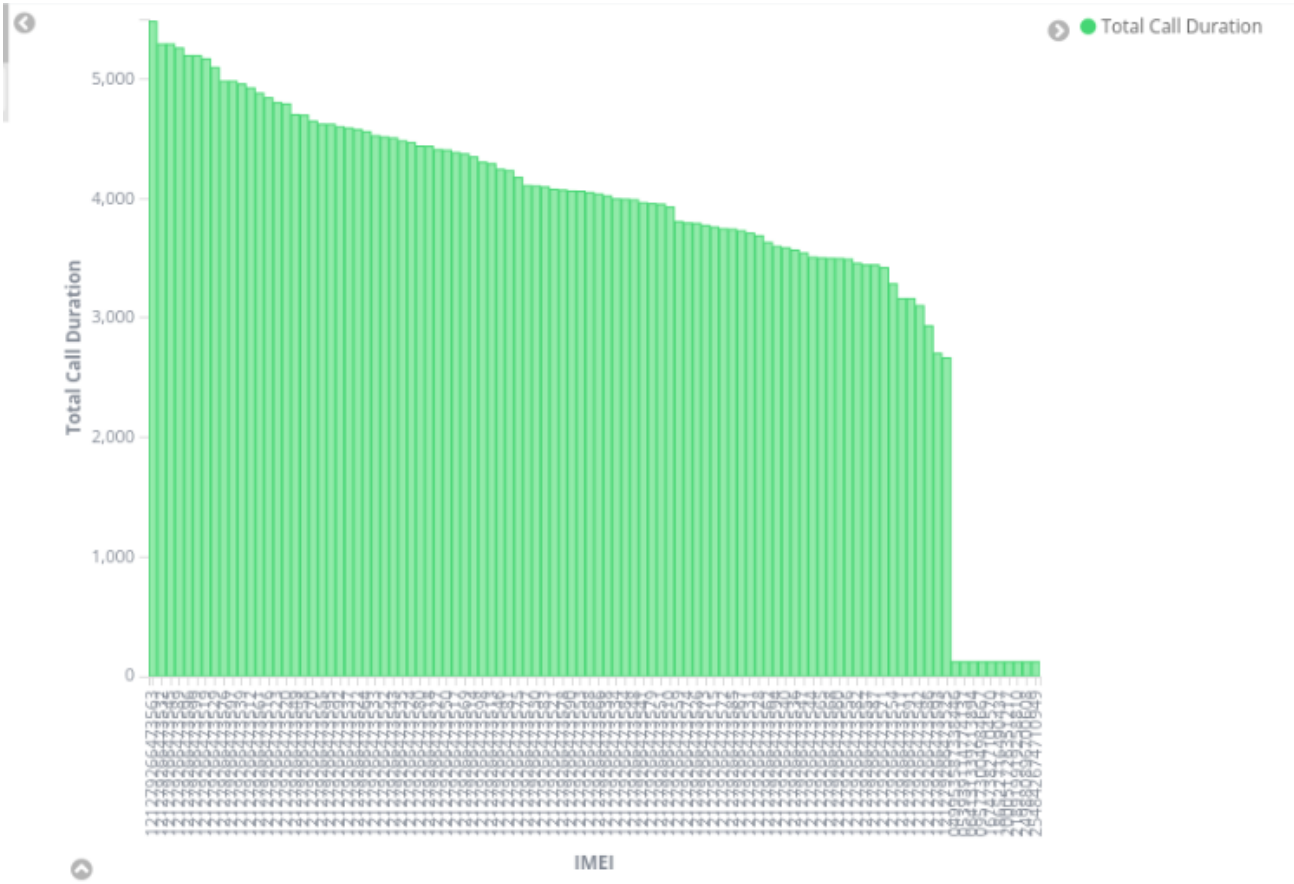

FIGURE 2.5: Valeurs très grandes suspectes

- **Ecart type de temps total d'appel par IMEI**
  Cette visualization est faite sur les CDRs de type "voice" donc nous lançons d'abbord la requête : `Type:voice`

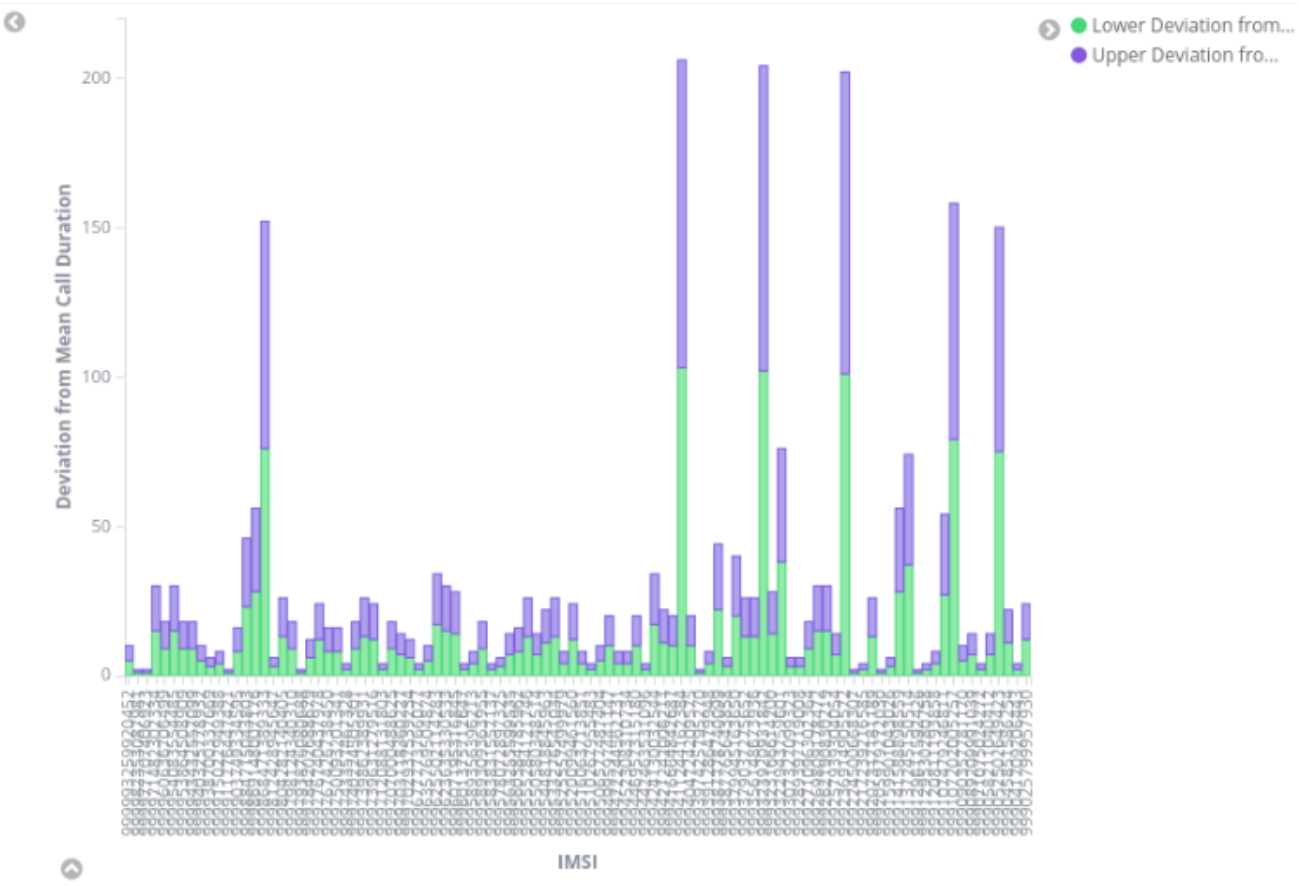

FIGURE 2.6: Deviations très grandes suspectes

- **Montant total de recharge par IMEI**
  Cette visualisation est faite sur tout les CDRs donc nous lançons d'abord la requête : *

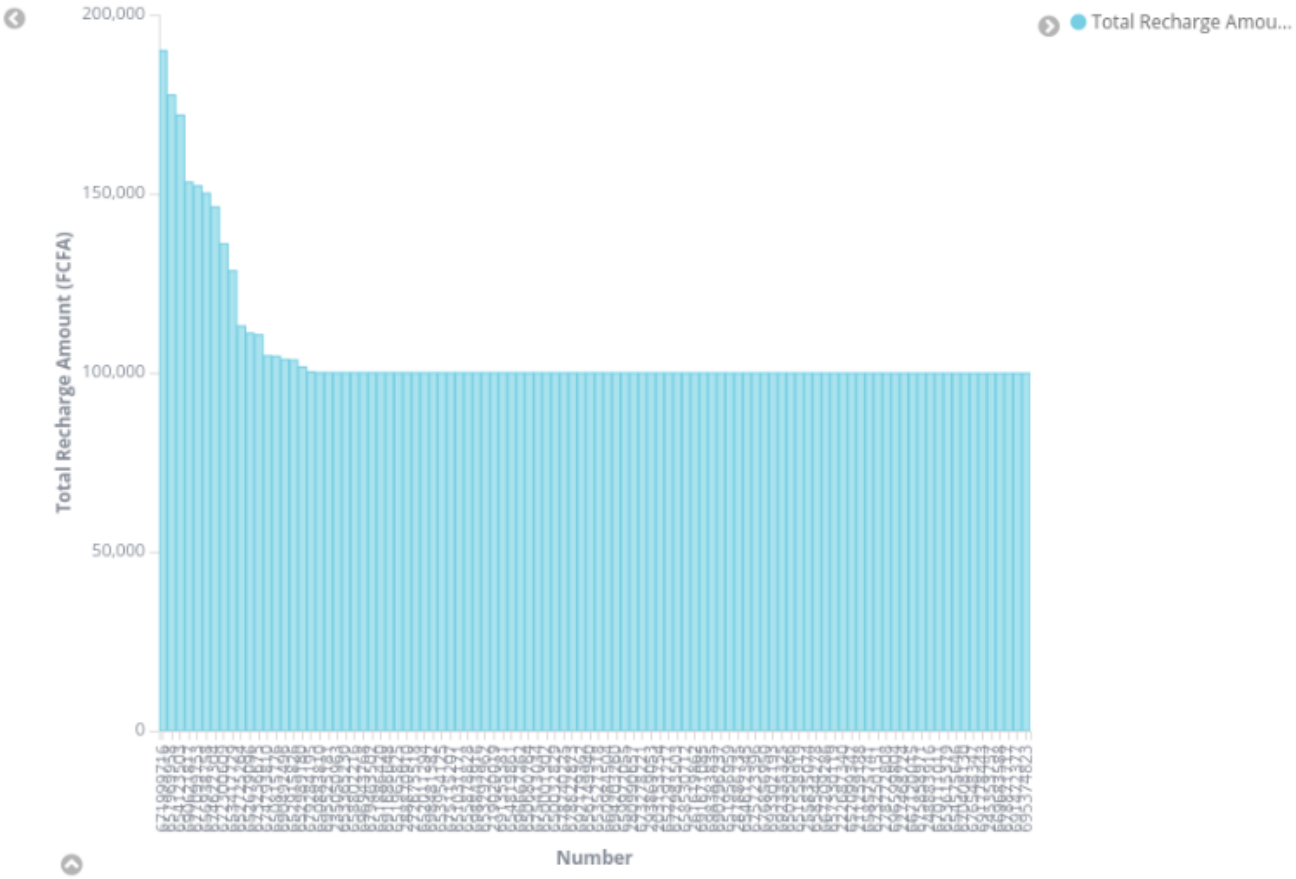

FIGURE 2.7: Valeurs très grandes suspectes

- **Durée minimum d'appel par IMEI**
  Cette visualisation est faite sur les CDRs de type "voice" donc nous lançons d'abord la requête : `Type:voice`

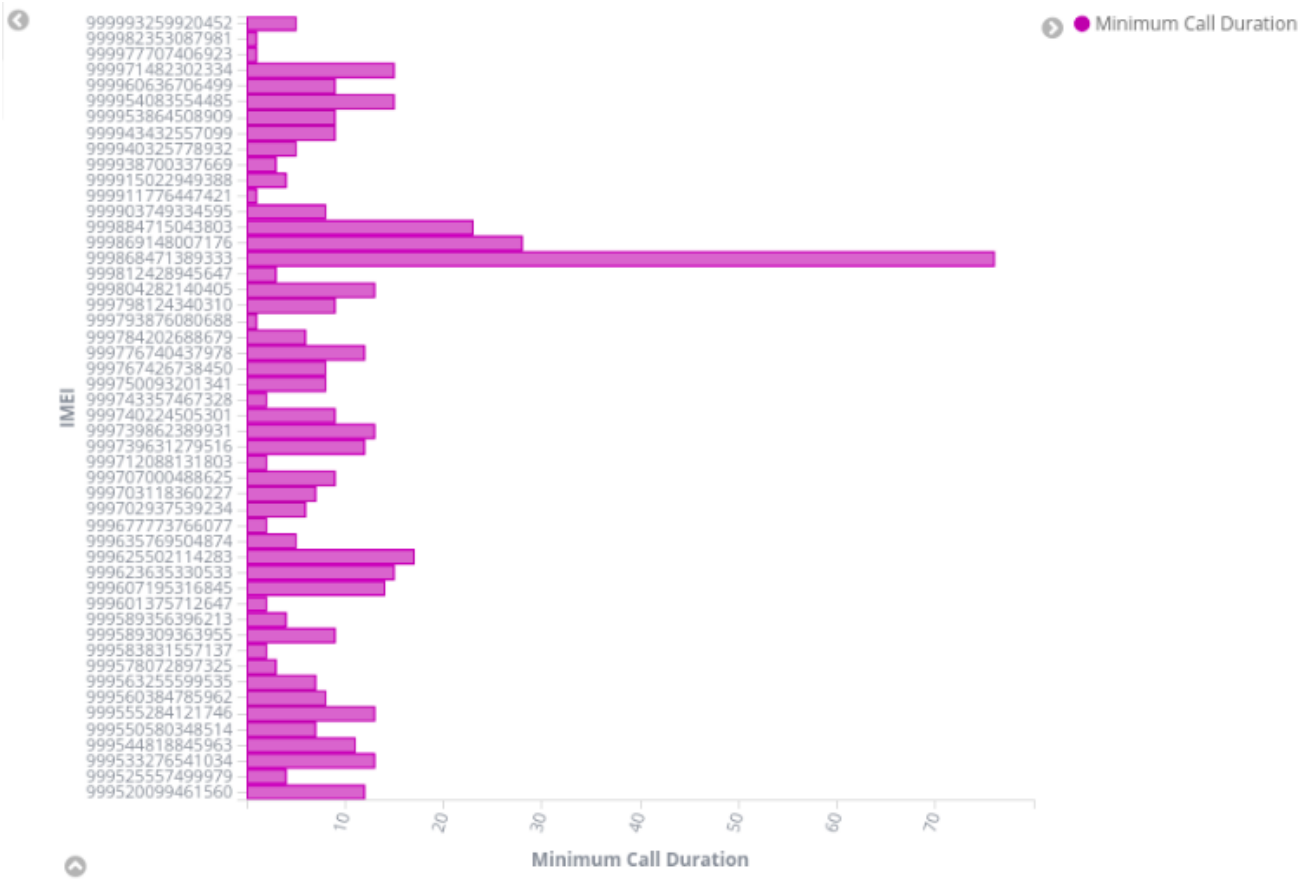

FIGURE 2.8: Valeurs très grandes suspectes

- **Ecart type de la quantité de surf total (en Mega) par IMEI**
  Cette visualisation est faite sur les CDRs de type "surf" donc nous lançons d'abord la requête : `Type:surf`

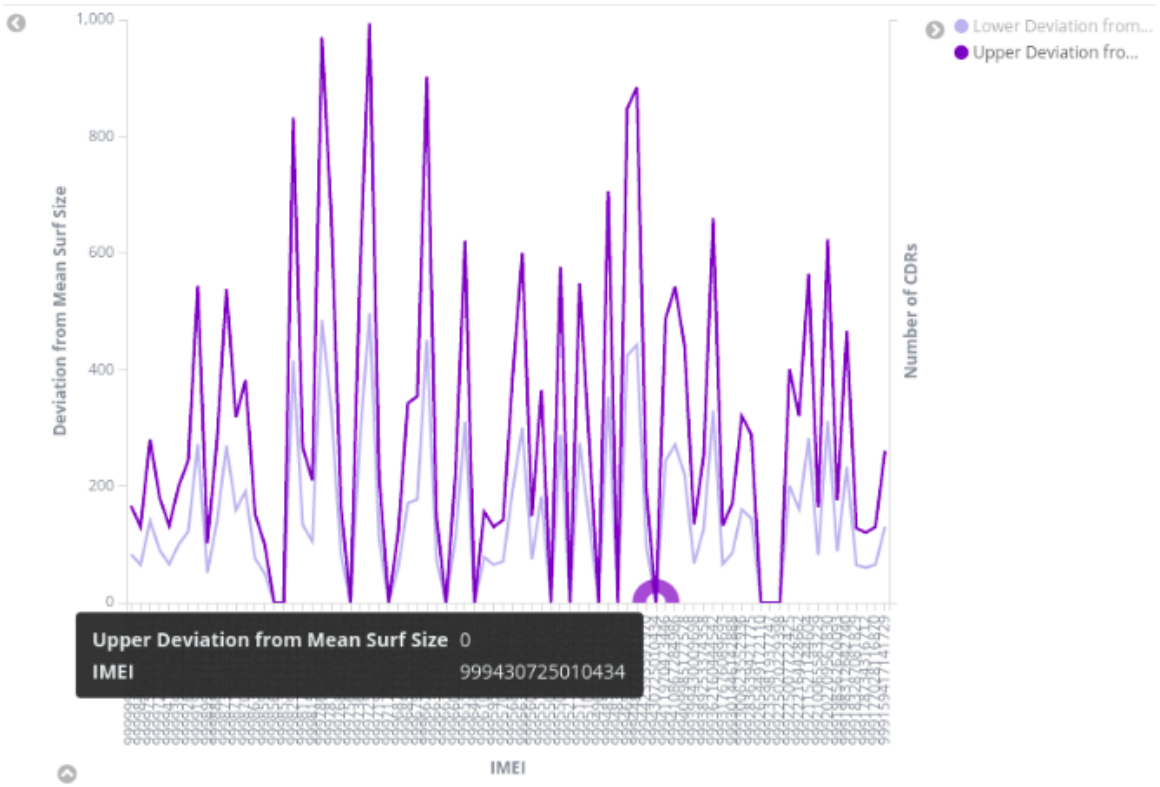

FIGURE 2.9: Valeurs très petites suspectes pour les IMEIs qui font beaucoup d'appels

NB : Il est important de noter que dans une scénario réelle, on ne peut dire avec certitude qu'un CDR appartient à un fraudeur ou pas juste en regardant la visualisation. Notre dispositif nous permet de détecter d'abbord les profils suspects et de suivre leurs comportements pendant une période, après laquelle nous pouvons conclure si c'est un fraudeur ou pas.

CHAPITRE 3

MANUEL D'UTILISATION

Ce chapitre est dédié à la présentation du manuel d'utilisation du dispositif. Il explique de façon assez claire les différents étapes pour déployer le dispositif et tester sur une machine linux. Le chapitre se structure comme suit :

Contents

3.1 Préparation de l'environnement

3.1.1 Installation de docker et docker-compose

Si vous n'avez pas déjà installé docker et docker compose, ouvrez votre terminal et entrez les commandes suivantes :

```
sudo apt-get update
sudo apt-get install curl
curl -sSL https://get.docker.com | sh

#Installer docker-compose
sudo curl -L https://github.com/docker/compose/releases/download/1.18.0/
docker-compose-‘uname -s‘-‘uname -m‘ -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose
#Tester l'installation avec la commande suivante
docker-compose --version
```

3.1.2 Création des dossiers de configurations

- Télécharger et dézipper le fichier `elkdocker.zip` : `https://drive.google.com/open?id=1vyJL5DpNj-cghS1GvJlJyDlzhBTVlDH1`.(Ce lien est donné aussi dans le mail envoyé.). Le dossier `elkdocker` contient tous les fichiers de configuration du système ELK lié au projet

3.2 Installation et lancement des images docker

- Arretez tous les services tournant sur les ports : 9200, 9300, 5044 et 5601.
- Lancer le service docker : `service docker start`
- Entrez dans terminal et téléchargez les images docker du projet :

  ```
  docker pull pyuhala01/cdr_gen
  docker pull pyuhala01/elasticsearch
  docker pull pyuhala01/logstash
  docker pull pyuhala01/kibana
  ```

- Déplacer vous dans le dossier `elkdocker/docker-compose/` et ouvrez le terminal. Tapez : `sudo docker-compose up` pour lancez les images.

Cette commande lancera tout les 4 containers du projet. Quelques données générées par le générateur de CDRs seront inserez dans Elasticsearch. Attendez quelques secondes pour que la Pile ELK soit prête.

3.3 Utilisation du dispositif

1. Pour tester si le service elasticsearch est lancé, ouvrez votre navigateur et entrez : `http://localhost:9200` . L'image suivante indique que Elasticsearch tourne sans problème :

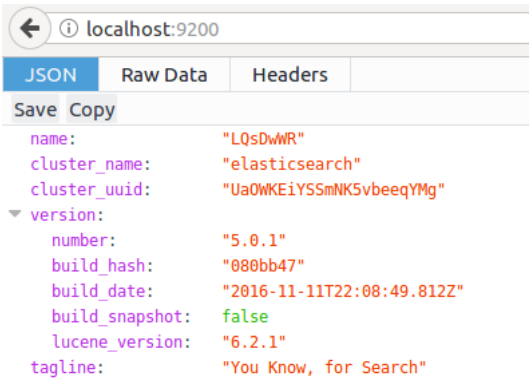

FIGURE 3.1: Interface JSON du service elasticsearch

2. Accéder à Kibana via le lien : `http://localhost:5601` . Vous devez voire l'interface suivante :

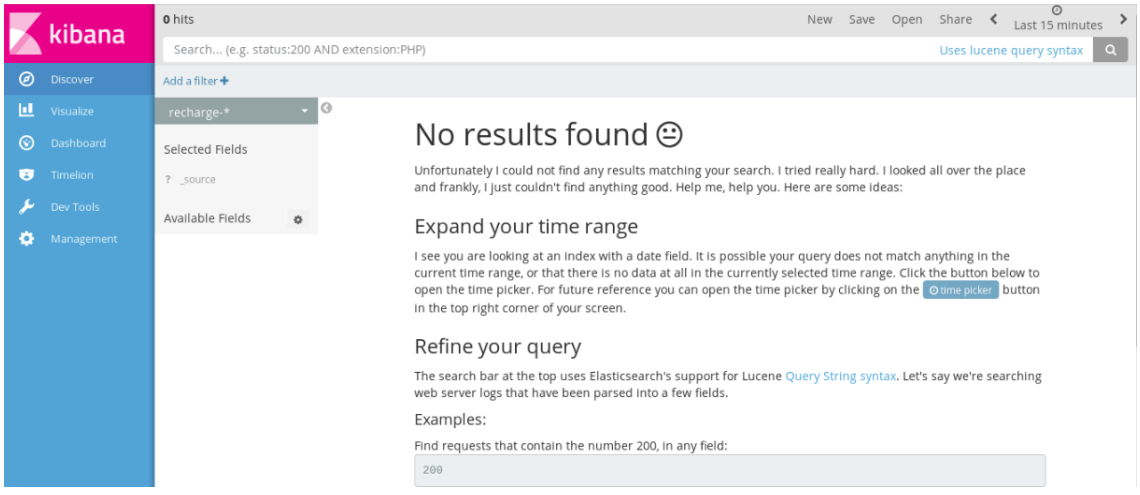

FIGURE 3.2: Interface Kibana

3. Dans un premier temps, vous n'allez probablement voir aucune donnée sur cette page parce que nos CDRs ont les Timestamps entre 01 janvier 2016 et 31 décembre 2016. Pour voire nos données, cliquez sur la petite horloge en haut à droite et dans le **Time Range** choissisez **Absolute**. Changez le **From** time en `2016-01-01 00:00:00.00`, comme indiquez dans la figure suivante :

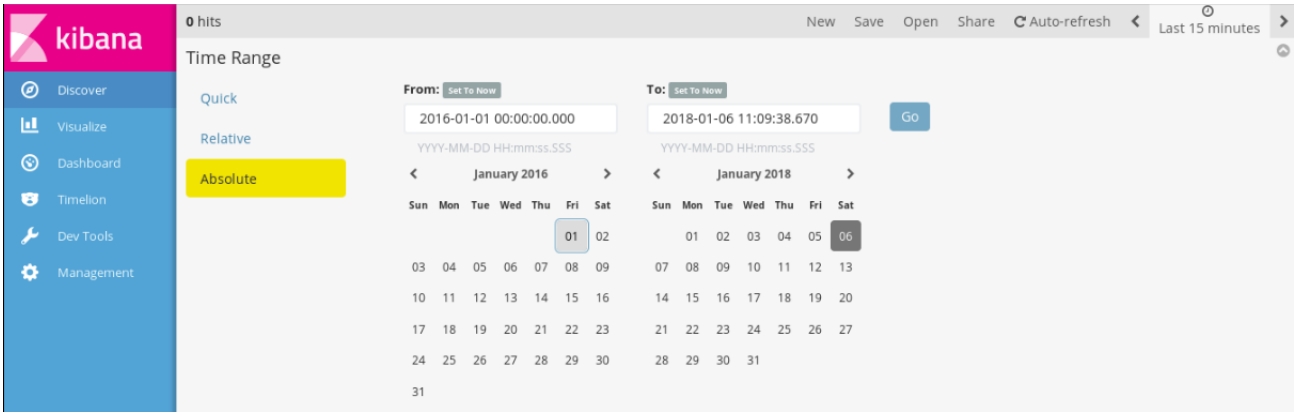

FIGURE 3.3: Kibana Time Range

4. Après avoir faite celà, cliquez sur l'onglet "Discover" et choissisez l'index **cdr-***. Kibana prendra quelques secondes pour afficher les résultats. Après le chargement, vous aurez le résultat suivant :
Ces documents peuvent être filtrés en réalisant des requêtes dans la barre de recherche.

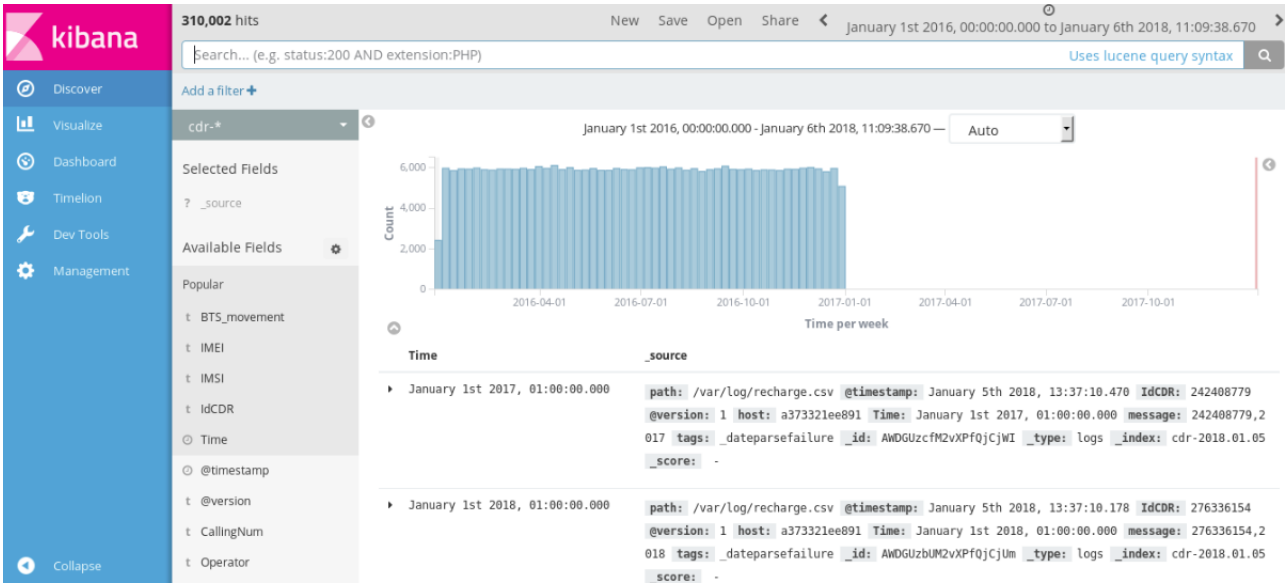

FIGURE 3.4: Données CDRs

Les données de recharge peuvent être visualisées en changeant l'index à : **recharge-***

5. **Quelques Requêtes Kibana** : Pour les récherches globale, nous utilisons les requêtes kibana. Pour éffectuez une requête, entrez-le dans la barre de récherche. Une liste de requêtes que nous avons faites est la suivante :
   • Tous les CDRs de type voice : `Type:voice`
   • Les CDRs de type voice avec BTS qui ne change pas : `Type:voice AND BTS_movement:0`
   • Les CDRs de type voice avec une durée d'appel supérieur à 30 minutes et un BTS qui ne change pas : `Type:voice AND Size:>30 AND BTS_movement:0`
   • Les profils qui ne surfent jamais : `Type:surf AND Size:0`
   • Les CDRs avec une durée d'appel entre 5 et 20 minutes :
   `Type:voice AND Size:[5 TO 20]`
   • Les SMS de l'opérateur MTN : `Type:sms AND Operator:mtn`

6. **Les Visualisations en Kibana** :
   La visualisation est au cœur de Kibana, car elle permet d'agréger nos données et d'afficher les résultats dans des diagrammes. Pour voir la liste des différents visualisations que nous avons créés, cliquez sur l'onglet "Visualize". Cliquez sur le nom d'une visualisation pour le

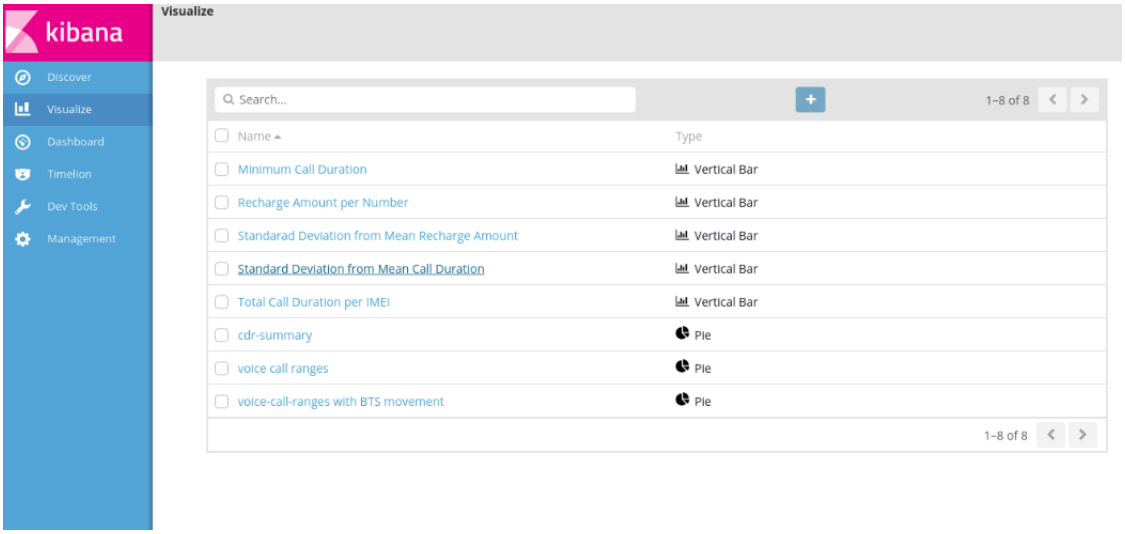

FIGURE 3.5: Les Visualisations

voir.

7. **Le Dashboard :**
   Kibana vous permet de juxtaposer les visualisations que vous avez créées dans un dashboard (tableau de bord) : un dashboard vous permet de visualiser en un coup d'œil les informations les plus importantes que vous pouvez tirer de vos données. Pour voir notre dashboard, cliquez sur l'onglet "Dashboard" et ouvrez `CDR-Dashboard`. La figure suivante donne une vue partielle de notre dashboard :

FIGURE 3.6: Le tableau de bord pour les CDRs

CONCLUSION GÉNÉRALE

Rappel du Problème

Ce travail a été effectué dans le cadre d'un projet académique du cours Systèmes Distribués et Virtualisation. Il était question ici de concevoir et mettre en oeuvre un dispositif pour la détection des fraudeurs à la Simbox au Cameroun.

Synthèse de la Solution

En termes de productivité, la plupart des objectifs fixés ont été atteints. La solution implémentée nous aura fait étudier bien des concepts de systèmes distribués, de virtualisation et de l'analyse des données à l'aide des outils Big Data comme la Pile ELK.

Difficultés rencontrés

A côté de tout ceci nous nous sommes confrontés à quelques problèmes :

- Les problèmes liés au permissions dans les images docker.
- La difficulté au niveau de dressage de profils ; C'est-à-dire, trouvez des caractéristiques très proches à la réalité.
- La prise en main des requêtes complexes dans Kibana.
- La génération de CDRs très réalistes.

Perspectives

Comme perspectives nous pouvons citer :

- L'amélioration dans la conception du système en gros, et surtout sur le module de génération de CDRs.
- La mise en place d'un cluster Elasticsearch (2 ou 3 containers Elasticsearch) pour améliorer la robustesse du système.
- L'introduction d'un plugin de Machine Learning dans Kibana, qui utilisera nos critères pour détecter les profils fraudeurs.

RÉFÉRENCES

[1] Alae-Eddine CHARRADI et al. *Projet Long final report :SIMbox fraud detection*. 2017.

[2] DOCKERDOCS. *Docker Official Documentation*. URL : https://docs.docker.com/.

[3] ELASTIC. *Elastic Official Documentation*. URL : https://www.elastic.io.

[4] Tim ROES. "Elasticsearch/Kibana Queries - In Depth Tutorial". In : (2016).

*