Déploiement de la solution SIEM

Graylog

de A à Z

Ershad Ramezani

Table des matières

SIEM

Pourquoi Graylog

- Sa conviviale interface lui permet de gérer une variété de formats de données.
- Une grande flexibilité est offerte en matière d'authentification et d'autorisations utilisateur.
- Il est également possible de le configurer pour recevoir des alertes par e-mail.
- Graylog est un logiciel open-source, donc il peut être utilisé gratuitement.

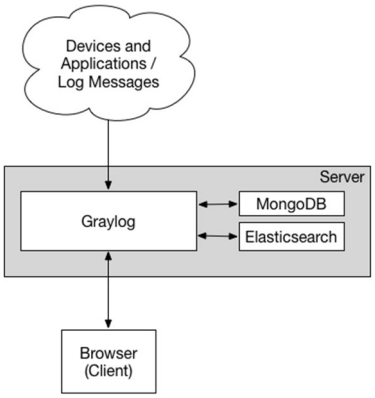
Un inconvénient est que le tableau de bord de gestion n'est pas assez convivial.

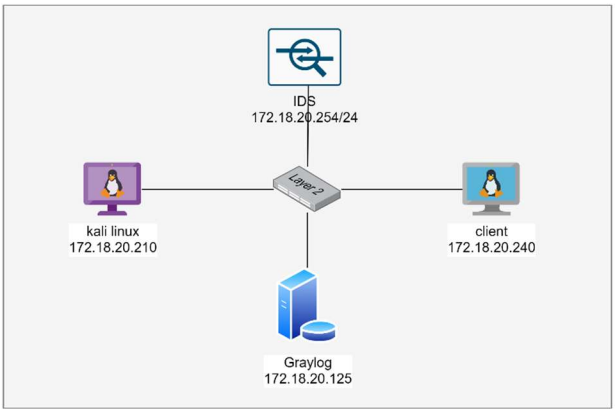
Les composants de Graylog sont les suivants :

- Le serveur Graylog : Il sert principalement à traiter les journaux.
- L'interface Web Graylog : Cette application de navigateur permet de visualiser les données et les journaux collectés.
- MongoDB : Il s'agit d'un serveur de base de données pour stocker les données de configuration.
- ElasticSearch : C'est un moteur de recherche et d'analyse gratuit et open-source qui analyse et indexe les données brutes provenant de diverses sources.

Configuration minimale

Il s'agit d'une configuration minimale de Graylog qui peut être utilisée pour des environnements plus petits, non-critiques ou pour des tests. Aucun des composants n'est en double, et leur installation est rapide et facile.





Installation de Graylog 5

Au moment de la réalisation de ce projet, la version la plus récente de Graylog était la version 5.0. Pour maintenir la compatibilité avec ses dépendances logicielles, Graylog 5.0 requiert les éléments suivants :

- OpenJDK 17 (JDK 17 est l'implémentation de référence open source de la version 17 de la plate-forme Java SE. Il est intégré à Graylog 5.0 et n'a pas besoin d'être installé séparément.)
- Elasticsearch 7.10.2
- MongoDB 5.x ou 6.x

J'ai installé Graylog sur une machine Ubuntu Server 22.04 en suivant un processus en plusieurs étapes. Pour de plus amples informations sur les prérequis et les étapes d'installation, veuillez suivre le lien suivant. J'ai suivi les étapes 1, 2 et 3 telles qu'indiquées dans la procédure, mais j'ai adapté les étapes 4 et 5 à mes besoins spécifiques.

https://computingforgeeks.com/install-graylog-on-ubuntu-with-lets-encrypt/

Avant de commencer l'installation, il est recommandé de vérifier que la machine Ubuntu dispose des éléments suivants :

- Un nom de domaine complet (FQDN) approprié pour le serveur : graylog.vironax.local
- Les lignes suivantes doivent être ajoutées au fichier /etc/hosts et vérifiées avec la commande "hostname -f"

```
127.0.0.1 localhost
172.18.20.125 graylog.vironax.local graylog
```

Il est également recommandé de vérifier le fuseau horaire et la date/heure de la machine Ubuntu. Vous pouvez vérifier le fuseau horaire actuel en exécutant la commande "date" dans le terminal. Pour configurer le fuseau horaire à "Europe/Paris", vous pouvez exécuter la commande suivante en tant qu'utilisateur root ou avec la commande "sudo" :

```
sudo timedatectl set-timezone Europe/Paris
```

Etape 1 : installer MongoDB

Les commandes suivantes permettent d'installer MongoDB :

```
wget http://archive.ubuntu.com/ubuntu/pool/main/o/openssl/libssl1.1_1.1.0g-2ubuntu4_amd64.deb

sudo dpkg -i libssl1.1_1.1.0g-2ubuntu4_amd64.deb

sudo apt install wget curl gnupg2 software-properties-common apt-transport-https ca-certificates lsb-release

curl -fsSL https://www.mongodb.org/static/pgp/server-6.0.asc|sudo gpg --dearmor -o
/etc/apt/trusted.gpg.d/mongodb-6.gpg

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu $(lsb_release -cs)/mongodb-org/6.0
multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list

sudo apt update

sudo apt install mongodb-org
```

La dernière étape consiste à activer MongoDB lors du démarrage du système d'exploitation et à vérifier qu'il est en cours d'exécution :

```
sudo systemctl daemon-reload

sudo systemctl enable mongod.service

sudo systemctl restart mongod.service

sudo systemctl --type=service --state=active | grep mongod
```

Etape 2 : installer Elasticsearch

Elasticsearch est l'outil utilisé pour stocker et analyser les journaux provenant de sources externes.

Téléchargez et installez la clé de signature Elasticsearch GPG :

```
curl  -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o
/etc/apt/trusted.gpg.d/elastic.gpg
```

Ajoutez le référentiel Elasticsearch à votre liste de sources :

```
echo "deb https://artifacts.elastic.co/packages/oss-7.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-
7.x.list
```

Installer Elasticsearch:

```
sudo apt update
sudo apt install elasticsearch-oss -y
```

Configurez le nom du cluster pour Graylog :

```
sudo vim /etc/elasticsearch/elasticsearch.yml
```

Modifiez le nom du cluster en **graylog** :

```
cluster.name: graylog
```

Ajoutez les informations suivantes dans le même fichier :

```
action.auto_create_index: false
```

Rechargez le démon et démarrez le service Elasticsearch. Avec la commande enable, il démarrera automatiquement au prochain redémarrage du système :

```
sudo systemctl daemon-reload
sudo systemctl restart elasticsearch
sudo systemctl enable elasticsearch
```

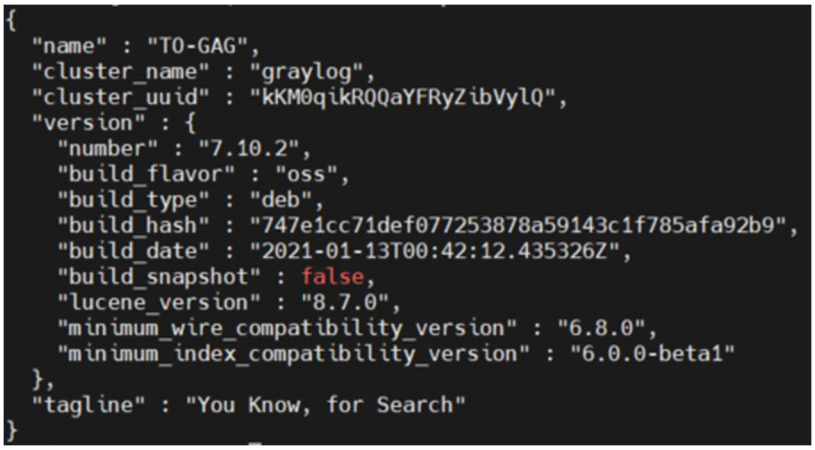
Vous pouvez vérifier l'état du service avec cette commande :

```
systemctl status elasticsearch
```

Elasticsearch s'exécute sur le port 9200 et cela peut être vérifié par la commande curl :

```
curl -X GET http://localhost:9200
```

Vous devriez voir le nom de votre cluster dans la sortie :



Étape 3 : Installer le serveur Graylog

Installez la configuration du référentiel Graylog et Graylog lui-même avec les commandes suivantes :

```
wget https://packages.graylog2.org/repo/packages/graylog-5.0-repository_latest.deb
sudo dpkg -i graylog-5.0-repository_latest.deb
sudo apt-get update && sudo apt-get install graylog-server
```

Générer un secret pour sécuriser les mots de passe des utilisateurs à l'aide de la commande **pwgen :**

```
sudo apt install pwgen

pwgen -N 1 -s 96
```

La sortie devrait ressembler à :

```
os6Dl7duLpgjS9E4ktrnaUC33ApATdMr5EWYlTBJUxNDr4C12T2ZbMvu34ruOnmfPB0C78KJyMO01feqHShicS
fiHrPPAegr
```

Modifiez le fichier de configuration graylog pour ajouter le secret que nous venons de créer :

```
sudo vim /etc/graylog/server/server.conf
```

Localisez la ligne **password_secret =** et ajoutez le secret créé ci-dessus après :

```
password_secret =
os6Dl7duLpgjS9E4ktrnaUC33ApATdMr5EWYlTBJUxNDr4C12T2ZbMvu34ruOnmfPB0C78KJyMO01feqHShicSfiHrPPAegr
```

Si vous souhaitez utiliser l'interface Graylog avec l'adresse IP et le port du serveur, définissez **http_bind_address** sur l'adresse IP de la machine :

```
http_bind_address = 172.18.20.125:9000
```

Changez le timezone pour l'admin du graylog en changant la directive **root_timezone =**.

```
root_timezone = Europe/Paris
```

Sauvegardez les modifications et sortez de **vim** avec :**wq!**

L'étape suivante consiste à créer un mot de passe de hachage sha256 pour l'administrateur. Il s'agit du mot de passe dont vous aurez besoin pour vous connecter à l'interface Web :

```
echo -n "Enter Password: " && head -1 </dev/stdin | tr -d '\n' | sha256sum | cut -d" " -f1
```

Vous obtiendrez une sortie de ce type :

```
Enter Password: Ershad
417bc64503aaa98daaa038dbd0acf81e23d14ac5d01aa47641a08afc22af0f72
```

Modifiez le fichier **/etc/graylog/server/server.conf** puis placez le mot de passe de hachage à **root_password_sha2 =** :

```
sudo vim /etc/graylog/server/server.conf
root_password_sha2 = 417bc64503aaa98daaa038dbd0acf81e23d14ac5d01aa47641a08afc22af0f72
```

Graylog est maintenant configuré et prêt à être utilisé :

Démarrez le service Graylog :

```
sudo systemctl daemon-reload
sudo systemctl restart mongod graylog-server
sudo systemctl enable mongod graylog-server
```

Vous pouvez vérifier si le service a démarré avec succès à partir des journaux :

```
sudo tail -f /var/log/graylog-server/server.log
```

Note : Si vous recevez cette erreur dans les logs, par rapport au la taille mémoire max pour le journal, vous pouvez le modifier dans la configuration de graylog :

```
2022-10-02T13:18:41.725+02:00 ERROR [PreflightCheckService] Preflight check failed with error: Journal directory </var/lib/graylog-server/journal> has not enough free space (2449 MB) available. You need to provide additional 2670 MB to contain 'message_journal_max_size = 5120 MB'
```

Trouvez ce ligne **message_journal_max_size =** et mettez son valeur (par defaut sur 5Go) au-dessous d'espaces disponbile (2449Mo)

```
message_journal_max_size = 2gb
```

Redémarrer le service et revérifier les logs :

```
sudo systemctl restart mongod graylog-server
sudo tail -f /var/log/graylog-server/server.log
```

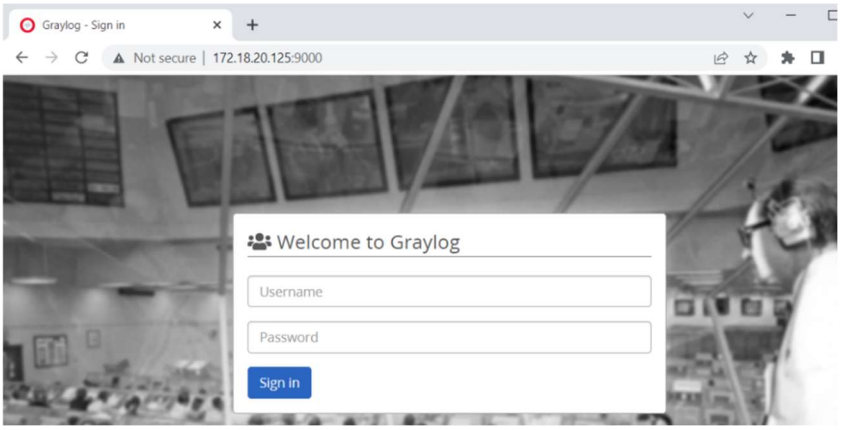
La sortie sera :

```
2023-02-08T23:18:41.226+01:00 INFO  [ServerBootstrap] Graylog server up and running.
2023-02-08T23:18:41.226+01:00 INFO  [ServiceManagerListener] Services are healthy
```

```
2022-10-02T13:23:22.464+02:00 INFO  [ServerBootstrap] Graylog server up and running.
2022-10-02T13:23:22.468+02:00 INFO  [ServiceManagerListener] Services are healthy
```

Vous pouvez ensuite accéder au tableau de bord Web graylog sur :

http://10.xx.xx.66:9000



Sécurisation du serveur Graylog

Il existe deux façons pour activer l'interface web et API du serveur Graylog en https :

Appliquer les configurations par rapport à TLS dans le fichier de configuration de Graylog :

Pour cela nous allons définir trois paramètres dans le fichier de configuration de Graylog :

- http_enable_tls=true
- http_tls_cert_file =
- http_tls_key_file =

Il est nécessaire que les fichiers de certificat (cert) et de clé privée (key) soient au format X.509 et PKCS#8 respectivement, et qu'ils soient tous les deux en format PEM. Cette exigence est due à la compatibilité requise avec Graylog, qui n'accepte que ces formats.

PKCS#8

Le standard de cryptographie publique PKCS#8 (Public-Key Cryptography Standards #8) a été développé pour définir le format de stockage des clés privées utilisées avec des algorithmes de cryptographie asymétrique. Ce format permet de stocker les clés privées de manière sécurisée pour leur utilisation ultérieure.

Le format de clé privée PKCS#8 est généralement considéré comme un standard largement accepté pour de nombreuses applications de cryptographie. Il est souvent utilisé en combinaison avec d'autres standards de cryptographie publique, tels que X.509 pour les certificats SSL/TLS et SSL.

X.509

Le standard X.509 est un standard international qui définit le format des certificats numériques utilisés dans la sécurité des communications sur Internet.

OpenSSL

OpenSSL est un logiciel de cryptographie open-source qui prend en charge une large gamme de protocoles de sécurité, tels que SSL, TLS, PGP, et autres. Il est largement utilisé sur les systèmes d'exploitation Unix et Linux pour gérer les clés et les certificats, générer des clés, signer des certificats, coder et décoder des données, ainsi que pour la gestion des certificats.

OpenSSL offre une variété de fonctionnalités pour la gestion de la cryptographie, telles que:

- Création et gestion de clés : OpenSSL peut générer des paires de clés publiques et privées pour différents algorithmes de cryptographie, comme RSA, DSA, et ECC.
- Signature et vérification de certificats : OpenSSL peut signer des certificats numériques et vérifier leur signature pour garantir leur authenticité.
- Cryptage et décryptage : OpenSSL peut être utilisé pour crypter et décrypter des fichiers et des données en utilisant différents algorithmes de cryptographie.
- Conversion de format de fichier : OpenSSL permet de convertir des fichiers de différents formats, tels que PEM, DER, PFX, etc. en un autre format.

Nous allons suivre les étapes ci-dessous pour créer les fichiers cert et key au format requis :

1. Tout d'abord, nous allons créer un fichier de configuration contenant les paramètres nécessaires pour créer la paire de clés et de certificat. Ce fichier de configuration sera utilisé par l'outil OpenSSL. Nous le nommerons openssl-graylog.cnf :

```
[req]
distinguished_name = req_distinguished_name
x509_extensions = v3_req
prompt = no
[req_distinguished_name]
C = FR
ST = Occitanie
L = Toulouse
O = XXXX
OU = SI
CN = graylog.toulouse.xxxx.local
[v3_req]
keyUsage = keyEncipherment, dataEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names
[alt_names]
IP.1 = 10.xx.xx.56
DNS.1 = graylog.toulouse.xxxx.local
```

2. Ensuite, nous allons créer le fichier de certificat au format PEM et en utilisant le standard X.509 :

```
sudo openssl req -x509 -days 365 -nodes -newkey rsa:2048 -config openssl-graylog.cnf -keyout pkcs5-privatekey.pem -out graylog-certificate.pem
```

3. Cette commande crée deux fichiers :
    a. graylog-certificate.pem qui contiendra le certificat,
    b. pkcs5-privatekey.pem qui contiendra la clé privée.
4. Comme Graylog ne reconnaît que les clés privées au format PKCS#8, nous allons convertir la clé privée en ce format en utilisant la commande appropriée :

```
sudo openssl pkcs8 -in pkcs5-privatekey.pem -topk8 -nocrypt -out graylog-privatekey.pem
```

Une autre option aurait été de créer un nouveau fichier de certificat avec un mot de passe :

```
sudo openssl pkcs8 -in pkcs5-privatekey.pem -topk8 -out graylog-privatekey.pem -passout pass:secret
```

```
total 16
-rw-r--r-- 1 root     root 1419 févr.  8 23:28 graylog-certificate.pem
-rw------- 1 root     root 1704 févr.  8 23:28 graylog-privatekey.pem
-rw-r--r-- 1 admintls sudo  377 févr.  8 23:27 openssl-graylog.cnf
-rw------- 1 root     root 1704 févr.  8 23:28 pkcs5-privatekey.pem
```

Maintenant que nous avons créé la paire de clés/certificat, nous allons les ajouter au fichier de configuration de Graylog :

```
#### Enable HTTPS support for the HTTP interface
#
# This secures the communication with the HTTP interface with TLS to prevent request forgery and eavesdropping.
#
# Default: false
http_enable_tls = true

# The X.509 certificate chain file in PEM format to use for securing the HTTP interface.
http_tls_cert_file = /etc/graylog/server/graylog-certificate.pem

# The PKCS#8 private key file in PEM format to use for securing the HTTP interface.
http_tls_key_file = /etc/graylog/server/graylog-privatekey.pem

# The password to unlock the private key used for securing the HTTP interface.
#http_tls_key_password = secret
```

Mettre en place un proxy inverse :

La deuxième méthode consiste à configurer un proxy inverse qui redirige toutes les demandes HTTPS reçues vers le serveur Graylog configuré en http :

Créer le Nginx en http

Pour créer un serveur Nginx en HTTP, suivez les étapes ci-dessous :

Installez Nginx en utilisant les commandes suivantes :

```
sudo apt update
sudo apt install nginx
```

Créez un nouveau fichier de configuration virtualhost pour votre site Graylog en utilisant la commande suivante :

```
sudo vim /etc/nginx/sites-available/graylog.conf
```

Ajoutez les lignes suivantes au nouveau fichier de configuration :

```
server {
  listen 80;
  server_name       graylog.vironax.local;
  access_log         /var/log/nginx/graylog.vironax.local.access.log combined;
  error_log          /var/log/nginx/graylog.vironax.local.error.log;
}
```

Créez un lien symbolique pour le fichier de configuration en utilisant la commande suivante :

```
sudo ln -s /etc/nginx/sites-available/graylog.conf /etc/nginx/sites-enabled/
```

Vérifiez que la configuration de Nginx est correcte en utilisant la commande suivante :

```
sudo nginx -t
```

Si la configuration est correcte, vous devriez voir la sortie suivante :

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Enfin, créez un enregistrement A dans le DNS de votre routeur pfsense pour permettre à votre site Graylog d'être accessible via le nom de domaine que vous avez configuré. Dans **services/dns resolver/general settings :**

J'ai utilisé ShadowCA pour créer une autorité de certificat et ensuite le certificat nécessaire pour le nginx :

1. Installer shadowCA :

```
git clone https://github.com/graylog-labs/shadowCA.git
```

2. Générer le CA :

```
bash bin/create_ca_certificate.sh
```

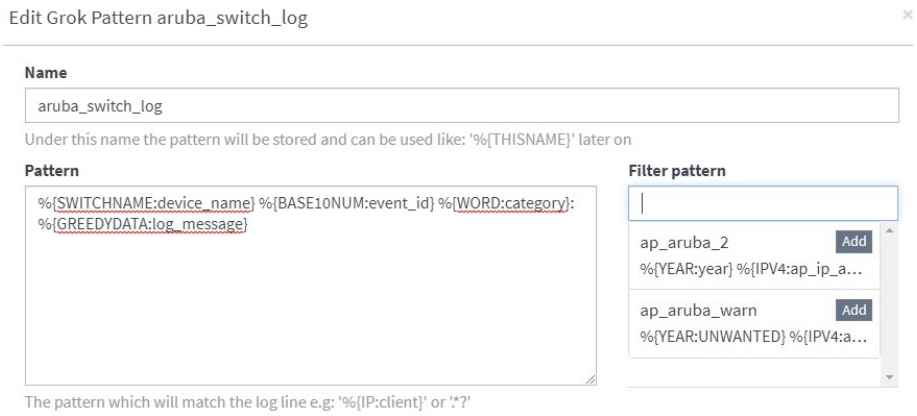
3. Générer la paire clé/certificat :

Vous n'avez pas besoin de sauvegarder cet extracteur, car on a un pipeline avec ce modèle grok. Copiez seulement ce modèle pour la création un modèle grok dans le menu system/grok patterns.

Dans le system/grok patterns créez un nouveau modèle, nommez-le aruba_switch_log (comme écrit dans la règle pipeline) et sauvegardez-le.

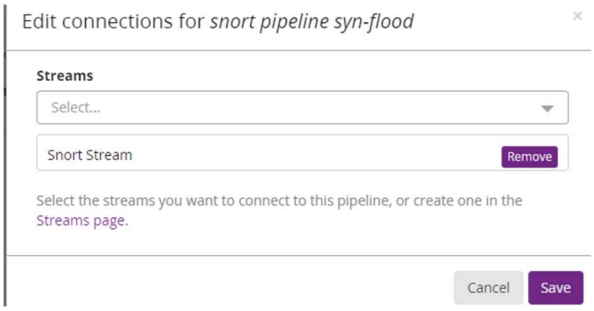
Note : Ce modèle peut être utilisé pour plusieurs pipeline.

Note : un pipeline peut avoir plusieurs modèles grok (nous ajouterons les autres modèles pour les autres logs dans la même règle pipeline.)



Connecter le pipeline au flux ARUBA-SWITCH

Revenez dans le *system/pipelines/edit* et cliquez sur *Edit connections* et ajoutez le flux *ARUBA-SWITCH*.

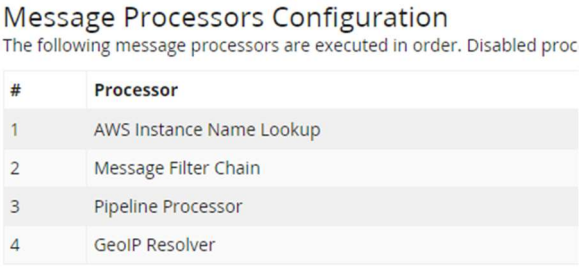


Configurer le Stage

Modifiez le *Stage* par défaut avec la *priorité 0*. Choisissez « *au moins une des règles sur ce Stage s'accorde avec le message* ». Ajoutez la règle *aruba_switch_rule* et sauvegardez les changements.

Changer l'ordre des processeurs de messages

Avant de commencer à utiliser les pipelines, vous devez vous assurer que le processeur de messages *Pipeline Processor* est activé et correctement configuré. Vous pouvez le faire en accédant à la page *Système/Configurations* et en vérifiant la configuration dans la section Configuration des processeurs de messages.



Sur la page *Configurations*, vous devez activer le processeur de messages pipeline et, si vous souhaitez que vos pipelines aient accès aux champs statiques définis sur les Inputs, mettez le processeur de pipeline après le processeur *Message Filter Chain*. Ce qui est notre cas :

Simulator

Après avoir fini la configuration de normalisation des logs (créer le pipeline, créer le modèle Grok, créer la règle pipeline, connecter le pipeline au flux Snort), nous pouvons la tester contre les logs sans vraiment besoins de déclencher les nouveaux alertes grâce à **Simulator** disponible dans la page *Pipelines overview*. Il suffit copier le champ message, choisir le flux et le codec du log. Ensuite en cliquant sur *Load message* on verra le résultat (sans devoir appliquer ces tests sur les vrais logs).

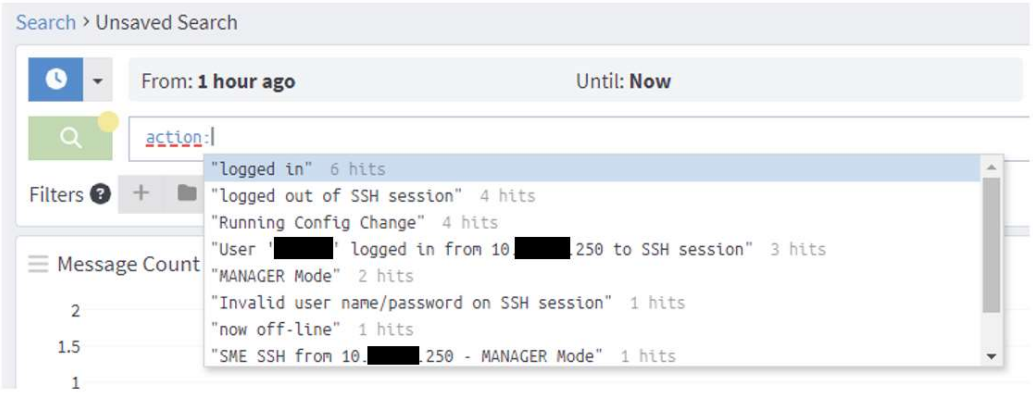
C'est la liste des différents grok pattern que j'ai créé pour les logs importants des switchs Aruba :

Name	Pattern	Actions
aruba_sw_config_changed 1	%{SWITCHNAME:device_name} %{WORD:category}: %{WORD:category_type}-Type='%{DATA:action}',Event-ID='%{INT:event_id}',Config-Method='%{WORD:config_method}',Device-Name='%{SWITCHNAME:device_name}',User-Name='%{WORD:username}',Remote-IP-Address='%{IPV4:ip_address}'	Edit Delete
aruba_sw_config_changed 2	%{SWITCHNAME:device_name} %{BASE10NUM:event_id} %{WORD:category}: %{DATA:action} by CLI. %{GREEDYDATA:UNWANTED}	Edit Delete
aruba_sw_invalid_userpass	%{SWITCHNAME:device_name} %{BASE10NUM:event_id} %{WORD:category}: %{DATA:action} User '%{WORD:username}' is trying to login from %{IPV4:ip_address}	Edit Delete
aruba_sw_login_logout	%{SWITCHNAME:device_name} %{BASE10NUM:event_id} %{WORD:category}: User '%{WORD:username}' %{DATA:action} from %{IPV4:ip_address}	Edit Delete
aruba_sw_others	%{SWITCHNAME:device_name} %{BASE10NUM:event_id} %{WORD:category}: %{GREEDYDATA:log_data}	Edit Delete
aruba_sw_port_state1	%{SWITCHNAME:device_name} %{BASE10NUM:event_id} %{WORD:category}: port %{BASE10NUM:port} is %{GREEDYDATA:action}	Edit Delete
aruba_sw_port_state2	%{SWITCHNAME:device_name} %{BASE10NUM:event_id} %{WORD:category}: ST%{INT:UNWANTED}-CMDR: port %{BASE10NUM:port}/%{BASE10NUM:port} is %{GREEDYDATA:action}	Edit Delete
aruba_sw_user_mode	%{SWITCHNAME:device_name} %{BASE10NUM:event_id} %{WORD:category}: %{DATA:UNWANTED} %{IPV4:ip_address} - %{GREEDYDATA:action}	Edit Delete

Et voici la règle pipeline pour ces logs :

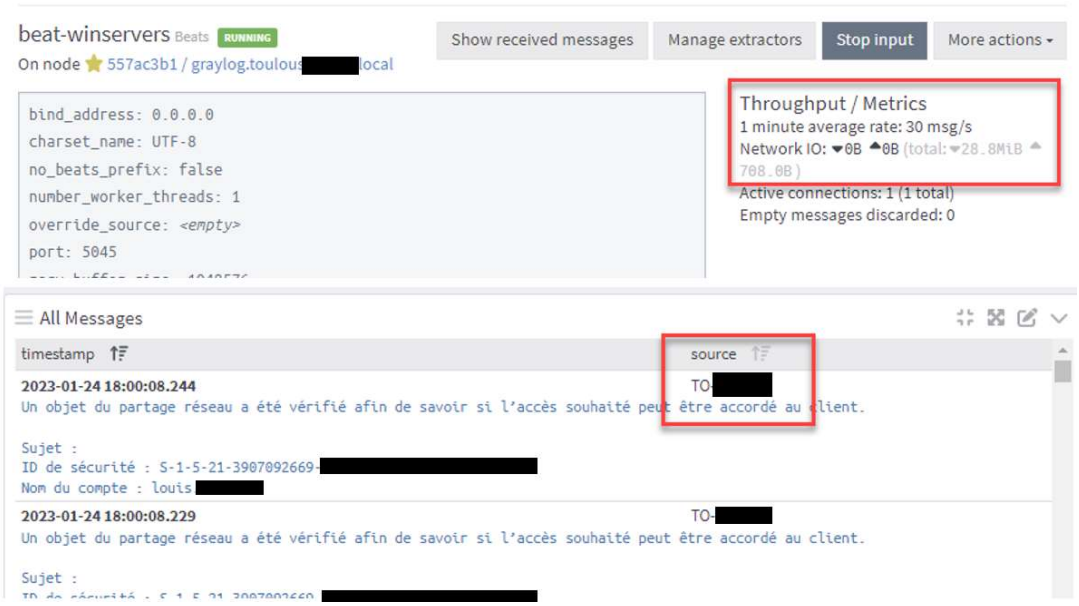
```
// le nom de la règle
rule "aruba_sw_rule"
// la condition
when
// quand le log a un champ nommé message
    has_field("message")
//execute les fonctions suivantes
then
//convertir le champ message en une chaine des caractère (string)
//utiliser le grok pattern nommé aruba_switch_log pour pour analyser le champ message
//mettre le résultat de la fonction grok dans le variable analyseur1
    let analyse1 = grok("%{aruba_sw_login_logout}", to_string($message.message), true);
    let analyse2 = grok("%{aruba_sw_port_state1}", to_string($message.message), true);
    let analyse3 = grok("%{aruba_sw_port_state2}", to_string($message.message), true);
    let analyse4 = grok("%{aruba_sw_invalid_userpass}", to_string($message.message), true);
    let analyse5 = grok("%{aruba_sw_config_changed1}", to_string($message.message), true);
    let analyse6 = grok("%{aruba_sw_config_changed2}", to_string($message.message), true);
    let analyse7 = grok("%{aruba_sw_user_mode}", to_string($message.message), true);
    let analyse8 = grok("%{aruba_sw_others}", to_string($message.message), true);
//créer les sous-champs à partir du variable analyseur
    set_fields(analyse1);
    set_fields(analyse2);
    set_fields(analyse3);
    set_fields(analyse4);
    set_fields(analyse5);
    set_fields(analyse6);
    set_fields(analyse7);
    set_fields(analyse8);

end
```

Le champ action est créé pour l'ensemble de ces logs et on peut filtrer les logs selon les actions :



Et voici les grok patterns, la règle pipeline et le champ action pour les switchs FlexFabric :

Vérifiez l'input et on voit bien qu'il reçoit des logs :



Nous pouvons voir le log de Sidecar sur la cible pour voir comment il a reçu les commandes depuis le serveur Graylog :

```
time="2023-02-03T13:47:09+01:00" level=info msg="No configurations assigned to this instance.
Skipping configuration request."
time="2023-02-03T14:06:43+01:00" level=info msg="Adding process runner for: winlogbeat-
63dce7b9fa7f156d726bf54d"
time="2023-02-03T14:06:43+01:00" level=info msg="[winlogbeat-63dce7b9fa7f156d726bf54d]
Configuration change detected, rewriting configuration file."
time="2023-02-03T14:06:43+01:00" level=info msg="Trying to create directory for: C:\\Program
Files\\Graylog\\sidecar\\generated\\63dce7b9fa7f156d726bf54d\\winlogbeat.conf"
time="2023-02-03T14:07:24+01:00" level=info msg="[winlogbeat-63dce7b9fa7f156d726bf54d] Starting
(svc driver)"
```

Nous pouvons modifier le fichier de la configuration de winlogbeat pour définir quel catégorie des logs de Windows doit être envoyé au serveur Graylog. Dans la configuration suivante j'ai marqué de seulement envoyer les évènements par rapport a la suppression des fichiers sur le serveur to-data-01 et l'envoie se fait seulement pour les évènements reçus dans la dernières 30 minutes.

Note : il faut activer le système d'audit pour les fichiers et dossiers concernés sur le serveur pour que le serveur journalise la suppression.

```
# Needed for Graylog
fields_under_root: true
fields.collector_node_id: ${sidecar.nodeName}
fields.gl2_source_collector: ${sidecar.nodeId}

output.logstash:
   hosts: ["10.[redacted].56:5045"]
path:
  data: C:\Program Files\Graylog\sidecar\cache\winlogbeat\data
  logs: C:\Program Files\Graylog\sidecar\logs
tags:
 - windows
winlogbeat:
  event_logs:
   - name: Security
     ignore_older: 30m
     event_id: 4663, 4660, 4659
```

Pour le serveur AD plusieurs événements sont en relation avec des objets active directory sont envoyés à Graylog :

Configuration

```
# Needed for Graylog
fields_under_root: true
fields.collector_node_id: ${sidecar.nodeName}
fields.gl2_source_collector: ${sidecar.nodeId}

output.logstash:
   hosts: ["10.[redacted].56:5045"]
path:
  data: C:\Program Files\Graylog\sidecar\cache\winlogbeat\data
  logs: C:\Program Files\Graylog\sidecar\logs
tags:
 - windows
winlogbeat:
  event_logs:
# account creations:
   - name: Security
     ignore_older: 30m
     event_id: 624, 631, 635, 658, 4720, 4727, 4731, 4754
# account deletions:
   - name: Security
     ignore_older: 30m
     event_id: 630, 4726
# account lockouts, adn unlocks
   - name: Security
     ignore_older: 30m
     event_id: 4740, 4767
# computer object created, deleted, modified
   - name: Security
     ignore_older: 30m
     event_id: 4741-4743
# group deleted
   - name: Security
     ignore_older: 30m
     event_id: 634,638,662,4730,4734,4758
# A user password has been reset - account activated
   - name: Security
     ignore_older: 30m
     event_id: 4724, 4722
# Group membership changes
   - name: Security
     ignore_older: 30m
     event_id: 4728,4729,4732,4733,4756,4757
# Group Modifications
   - name: Security
     ignore_older: 30m
     event_id: 4764,4735,4737,4755
# Failed Authentication Attempts, Interactive Logons, Successful Authentication Attempts
   - name: Security
     ignore_older: 30m
     event_id: 2, 10, 11, 4624, 4625
```

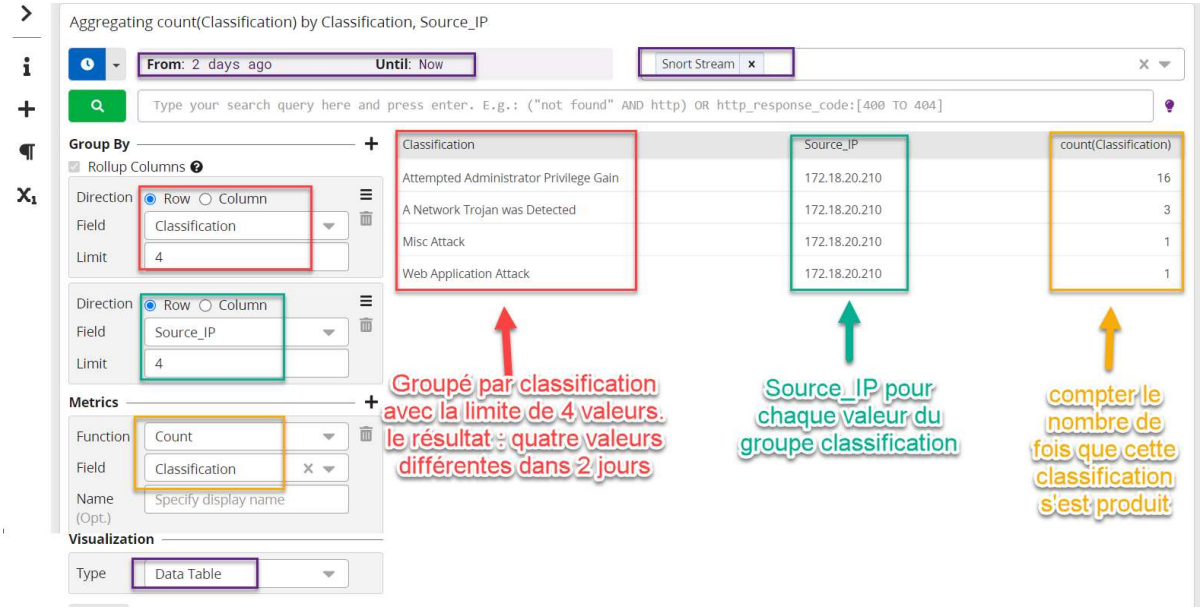
Sécurisation de Beats Input par TLS

Pour cela nous avons besoin un CA (je vais utiliser le même CA créer par shadowCA précédemment). Cela va être utilisé pour créer tous les certificats. Le certificat CA doit être importé dans tous les systèmes en relation avec Graylog beats input. Selon le navigateur utilisé, nous avons peut être besoin d'importer le CA en format *.der* dans le truststore du navigateur. En plus il faudrait ajouter le CA en format .der dans le Java (KVM keystore) qui est utilisé par Graylog.

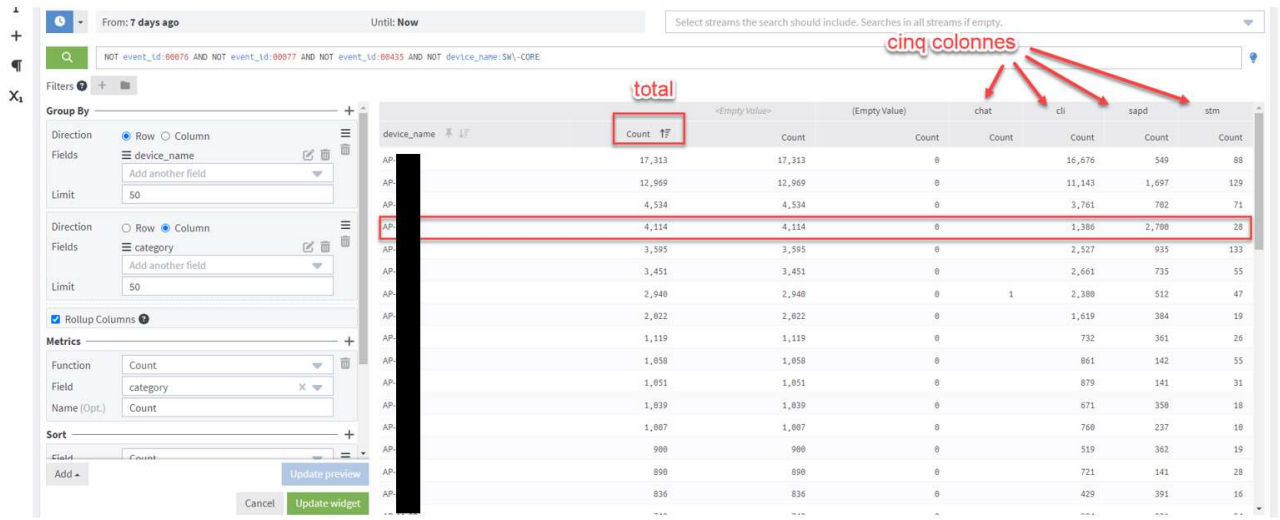
Graylog doit connaître l'autorité de certification utilisée pour vérifier les certificats. Le principal avantage est qu'il n'a besoin que du certificat CA et non de tous les certificats auto-signés connus dans la configuration :

a. Agrégation : l'option d'agrégation générique. Cette option vous permet de créer des différents graphiques personnalisés selon vos besoins et de combiner différents types de données dans un seul graphique.
b. Message Count : une agrégation prédéfinie pour compter le nombre des logs.
c. Message Table : une agrégation prédéfinie pour virtualiser les logs selon leurs horodatages.

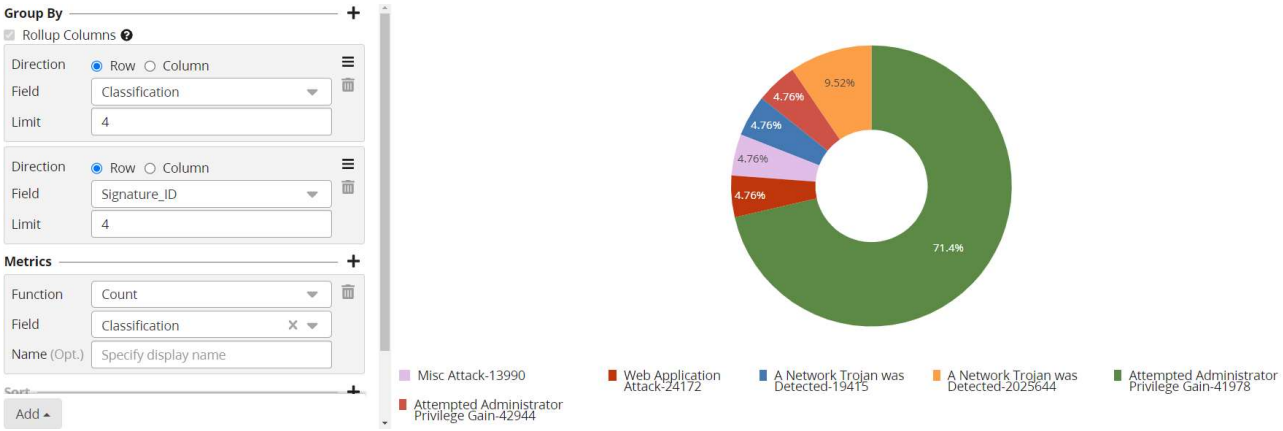
Agrégation



5. Cliquez sur agrégation pour créer une agrégation vide. Ensuite appuyez sur Edit. Il y a quatre options pour définir les paramètres du diagramme.
   a. **Group By** : Cette option vous permet de "grouper" votre graphique par lignes et colonnes. Lorsque vous créez un nouveau groupe à l'aide de Group By, les valeurs du champ sélectionnées sont cumulées dans le résultat. Dans la photo ci-dessus (exemple 1), d'abord j'ai regroupé les logs par le champ *Classification* et avec limite de 4 valeurs différentes. Ensuite j'ai ajouté un deuxième groupe pour des adresses IP sources de chaque valeur dans le groupe *classification*.
   b. **Metrics** : Ce sont un ensemble de fonctions permettant d'agréger des valeurs des champs. Le résultat de l'agrégation dépend du regroupement des lignes et/ou des colonnes.
   c. **Virtualization** : changer le type de diagramme : *Area Chart, Line Chart, Pie Chart*, etc. dans l'exemple ci-dessus, j'ai choisi *Data Table*.
   d. **Sort** : L'ordre des résultats peut être configuré.
6. Dans la photo ci-dessous (exemple 2), j'ai mis *Source_IP* par lignes et en premier, et *Classification* par colonnes et en deuxième :



7. Dans exemple ci-dessous, j'ai présenté le résultat de l'exemple 1 mais avec le *SID* au lieu de *Source_ID* et avec *Pie Chart* :



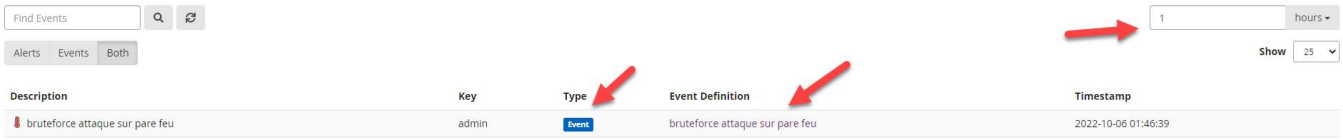
Suivez ce lien pour plus d'informations sur les widgets :
https://docs.graylog.org/docs/widgets

Agrégation prédéfinie

8. Le Message Table affiche les logs et leurs champs. Le Message Table peut être configuré pour afficher les champs et le log lui-même. Le log lui-même est rendu en police bleue sous les champs. Cliquer sur une ligne de message ouvre la vue détaillée d'un message avec tous ses champs.

Alertes

Graylog donne la possibilité de réagir à certaines conditions et d'envoyer des alertes à l'aide de notifications. Ces notifications peuvent être par e-mail.

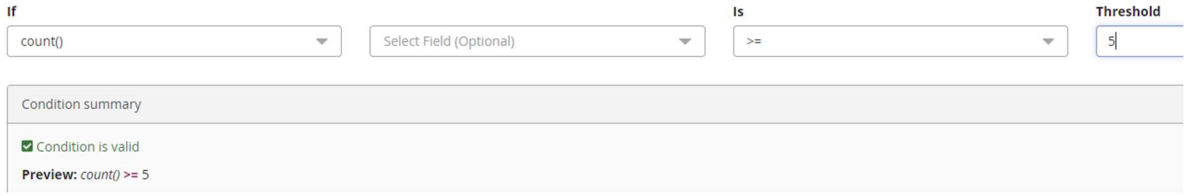


Agrégation : vous pouvez définir une condition et si les nombres des logs retournés dans le filtre arrivent à un seuil spécifique, créer un évènement (au lieu d'un évènement par log trouvé).

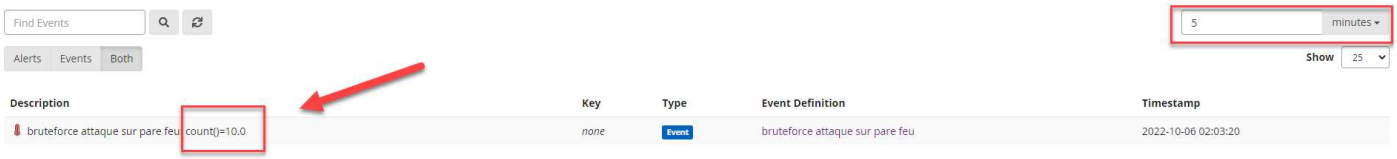
Par exemple, lancez plusieurs échecs de connexion dans une minute et vérifiez les évènements qui sont créés :

Description	Key	Type	Event Definition	Timestamp
bruteforce attaque sur pare feu	*none*	Event	bruteforce attaque sur pare feu	2022-10-06 01:54:19
bruteforce attaque sur pare feu	*none*	Event	bruteforce attaque sur pare feu	2022-10-06 01:54:17
bruteforce attaque sur pare feu	*none*	Event	bruteforce attaque sur pare feu	2022-10-06 01:54:15
bruteforce attaque sur pare feu	*none*	Event	bruteforce attaque sur pare feu	2022-10-06 01:54:13
bruteforce attaque sur pare feu	*none*	Event	bruteforce attaque sur pare feu	2022-10-06 01:54:10
bruteforce attaque sur pare feu	*none*	Event	bruteforce attaque sur pare feu	2022-10-06 01:54:08
bruteforce attaque sur pare feu	*none*	Event	bruteforce attaque sur pare feu	2022-10-06 01:54:06
bruteforce attaque sur pare feu	*none*	Event	bruteforce attaque sur pare feu	2022-10-06 01:54:05

Avec l'agrégation on peut éviter la création d'un évènement par log. Revenez dans le menu *filtre & agrégation*. Choisissez *aggregation of results reaches a threshold* et dans *Create events for definition* définissez cette condition :



Si le nombre des logs trouvés par le filtre est équivalent ou plus de 5 logs, il crée un évènement. Sauvegardez les modifications et faites un autre teste mais cette fois 5 échecs de connexion :



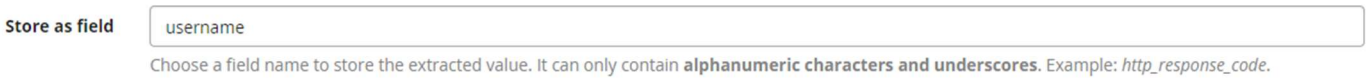
Cette fois un évènement est créé mais pour 10 échecs de connexion.

Group by Field(s) : dans notre exemple seulement le compte admin était sous attaque brute force. Et dans notre définition, on n'a pas défini quel utilisateur été la cible d'attaque. Mais cela est possible. On peut regrouper les logs trouvés par le filtre selon le champ utilisateur et créer un évènement par utilisateur qui était la cible d'attaque brute force.

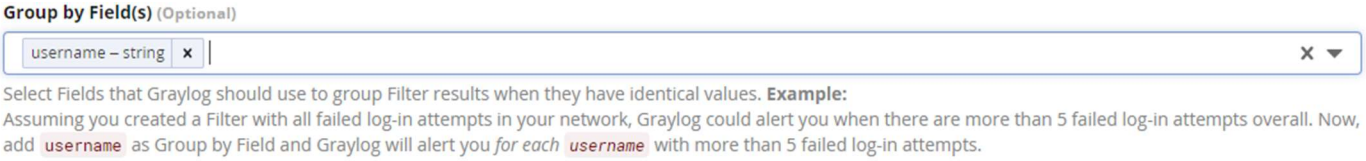
La première étape est de définir le champ utilisateur. J'utilise des extracteurs pour extraire le nom d'utilisateur.

1. Revenez dans le flux *Snort* et cliquez sur le champ *message*, ensuite choisissez *Create extractor*.
2. Pour le type d'extracteur, choisissez *Regular Expression*.
3. Dans la nouvelle page qui s'ouvre, créer un *regex* qui va extraire le nom l'utilisateur du log.

4. Définissez un nom pour le nouveau champ :



5. Revenez dans le *filtre & agrégation* et ajoutez le champ *username* dans le *group by fields* et sauvegardez les modifications.



Notifications

Les notifications vous alertent de tout événement configuré lorsqu'il se produit. Graylog peut vous envoyer des notifications par email. Les notifications peuvent être créées en sélectionnant le bouton Notifications sous l'onglet Alertes ou en les définissant dans la page de la création d'événement.

Email transport

On commence par la configuration d'email dans le fichier server.conf. Ajoutez ces lignes dans le fichier server.conf :

- Ici on indique au gaylog d'envoyer les emails au serveur smtp du google sur le port 25
- On lui donne l'email et le mot de passe d'un compte à utiliser.
- *From email* indique l'adresse email qui va apparaitre dans l'alerte

```
# Email transport
transport_email_enabled = true
transport_email_protocol = smtp
transport_email_hostname = smtp.gmail.com
transport_email_port = 25
transport_email_use_auth = true
transport_email_use_tls = true
transport_email_use_ssl = false
transport_email_auth_username = ershad.ra@gmail.com
transport_email_auth_password = [redacted]wx
transport_email_subject_prefix = [graylog]
transport_email_from_email = ershad.ra@gmail.com
transport_email_web_interface_url = https://172.18.20.125:443
```

Note : Certains FAI bloque le port smtp en sortant sur leur box internet. Par exemple chez *Orange* le port 25 est bloqué en sortant. Mais chez *Bouygues* ce port est ouvert. Pour vérifier l'ouverture du port faites un telnet sur le serveur smtp et vérifier la connexion : **telnet smtp.gmail.com 25**

```
ershad@graylog:~$ telnet smtp.gmail.com 25
Trying 74.125.71.108...
Connected to smtp.gmail.com.
Escape character is '^]'.
```

New Notification

1. Dans le menu *Alerts/Notifications*, créez une nouvelle notification en lui donnant un titre et une description. Choisissez Email comme le type de la notification.
2. Vous pouvez choisir l'adresse e-mail qui doit être utilisée comme expéditeur de la notification. Laissez-le vide pour utiliser l'adresse d'expéditeur par défaut.
3. Ajoutez les adresses e-mail qui recevront cette notification. J'ai mis ershad.ra@gmail.com pour le test. Alors l'expéditeur et destinateur sont le même.

/etc/graylog/server/GeoLite2-City.mmdb

Créer la table de recherche

4. Ensuite, nous devons configurer la table de recherche pour lire la base de données. Nous allons d'abord créer l'adaptateur de données depuis le menu */system/Lookup Tables/Data Adapters*. Pour *Data Adapter Type* choisissez **Geo IP – MaxMind™ Databases** et donnez-lui un titre, une description et un nom. Definissez le chemin du fichier .mmdb mettez le *Database type* sur **City database** et créer l'adaptateur.
5. Ensuite, nous devons créer un cache dans l'onglet *Caches*.
6. Dans la dernière étape de la table de recherche, nous devons créer la table elle-même, en utilisant l'adaptateur de données et le cache des deux étapes précédentes. Maintenant on a deux tables de recherches sur notre Graylog :

Title	Description	Name
src_name_lookup	src_name_lookup	src_name_lookup
geoip-city-lookup	geoip-city-lookup	geoip-city-lookup

Créer une règle pipeline pour la géolocalisation

Maintenant que nous avons créé la table de recherche et qu'elle est prête à être utilisée, nous devons créer une règle de pipeline pour l'utiliser et ajouter les métadonnées à chaque log avec une adresse IP.

Allez dans (Système -> Pipelines) et sous "Gérer les règles", nous devrons créer une nouvelle règle. Donnez-lui une description afin que vous puissiez vous en souvenir, et dans la source de la règle, mettez :

```
1  //le nom pour la règle
2  rule "geolocalisation"
3
4  //s'il y a un champ qui s'appelle "Source_IP"
5  when
6  has_field ("Source_IP")
7
8  //puis
9  then
10
11 //convertis-le en une chaîne des caractères
12 //et traduit-le avec geoip-city-lookup
13 //et mets le résultat dans le variable geo
14 let geo = lookup("geoip-city-lookup", to_string($message.Source_IP));
15
16
17 //crée les champs suivants à l'aide du variable geo :
18 set_field("src_ip_geolocation", geo["coordinates"]);
19
20 set_field("src_ip_geo_country_code", geo["country"].iso_code);
21
22 set_field("src_ip_geo_country_name", geo["country"].names.en);
23
24 set_field("src_ip_geo_city_name", geo["city"].names.en);
25
26 end
```

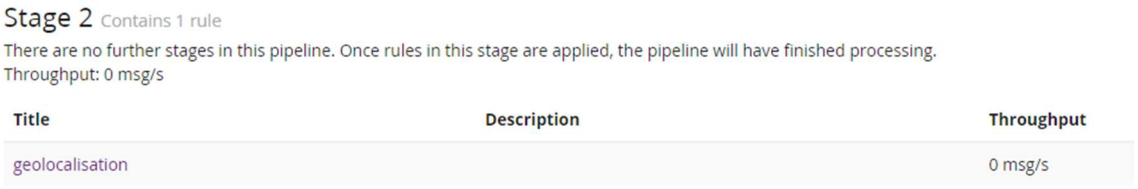
La condition *when* n'autorise le traitement de la règle que lorsque le journal contient le champ *Source_IP*. Une fois qu'il a trouvé ces journaux, il exécute la recherche sur notre table de recherche "*geoip-city-lookup*" avec les données dans le champ *Source_IP*, puis ajoute l'emplacement, le pays et la ville.

Notez que cette règle ne s'applique qu'à l'adresse IP source. Si l'adresse de destination doit également être recherchée, ajoutez des lignes supplémentaires à cette règle ou créez une deuxième règle pour les journaux avec des adresses IP de la destination.

Ajouter la nouvelle règle au pipeline

Après avoir créé les règles, nous les ajouterons à un *Stage* dans le pipeline *Snort*. Il y a déjà deux *Stage* dans notre pipeline. On ajoute un troisième.

REMARQUE IMPORTANT : cette règle cherchera le champ *Source_IP*. Le champ *Source_IP* sera créé pendant le premier Stage (stage 0) du pipeline. Donc il faut créer un Stage avec le numéro de la priorité plus haut.



Tester le résultat

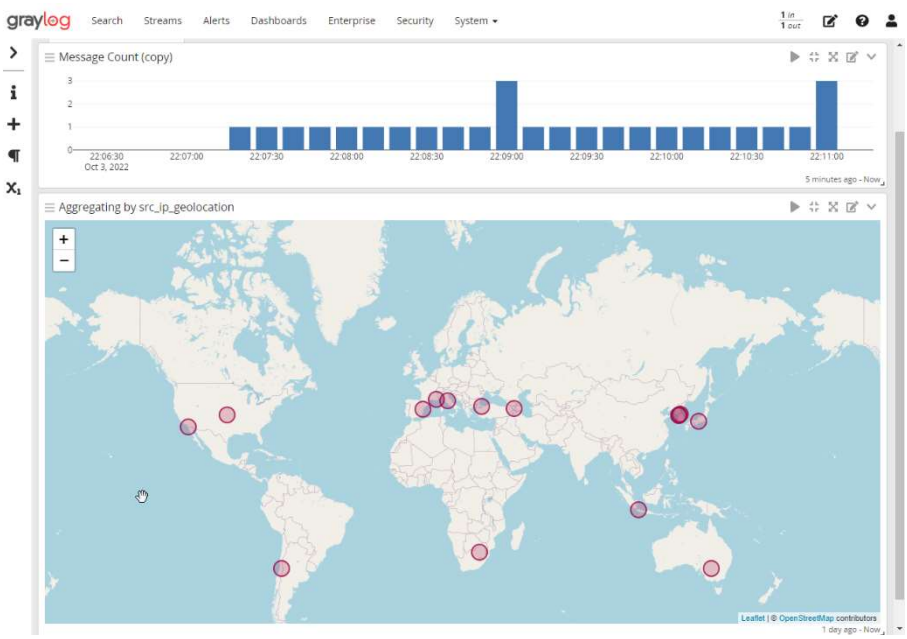
Et une fois que de nouveaux journaux arrivent dans le pipeline, vous verrez les champs par rapport à la géolocalisation.

Vous pouvez tester le résultat par le Simulator :

Vous pouvez lancer un agrégat de recherche sur "src_ip_geo_location" et mettez le type de table en "World Map", pour avoir une carte du mode. (Voir la section suivante, Dashboards)

World Map

World Map est un élément intéressant que l'on peut ajouter dans le tableau de bord. Une carte du monde a besoin de points géographiques sous forme de latitude, longitude. Comme nous avons déjà configuré la géolocalisation donc on va créer une carte du monde pour cela.



1. Créer un *Group By*. Mettez-le en Row. Pour le champ mettez src_ip_geolocation et avec la limite par défaut.
2. Pour la virtualisation choisissez World Map et sauvegarder les modifications. Voici le résultat :

La version complète est disponible aussi mais protéger par un mot de passe.

Merci de me contacter par email :

Ershad.ra@gmail.com