

Índice

[1. Instalación del script de enrequicimiento de datos 3](#_Toc179800970)

[1.1 Instalación de Python 3.6.12 3](#_Toc179800971)

[1.2 Creación del directorio 4](#_Toc179800972)

[1.3 Instalación de dependencias 4](#_Toc179800973)

[1.4 Creación de token y etiquetas 5](#_Toc179800974)

[1.5 Archivo config.ini 6](#_Toc179800975)

[1.6 Crontab 7](#_Toc179800976)

Malcolm: Additional Settings

Alejandro Jorge Poyuelo || Claudia Crespo Castillo ||   
José Manuel Candilejo Egea

# Instalación del script de enriquicimiento de datos

### 1.1 Instalación de Python 3.6.12

En primer lugar, debemos actualizar los paquetes que ya tenemos en nuestro sistema.

sudo apt update && sudo apt upgrade -y

Una vez actualizados, instalamos dependencias y descargamos el código fuente de Python 3.12.6 que ha sido la versión utilizada para el desarrollo de este código.

sudo apt install -y build-essential libssl-dev zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev libncursesw5-dev xz-utils tk-dev libffi-dev liblzma-dev python-openssl git

cd /usr/src

sudo wget <https://www.python.org/ftp/python/3.12.6/Python-3.12.6.tgz>

Extraemos el paquete del archivo comprimido y compilamos e instalamos Python.

sudo tar xzf Python-3.12.6.tgz

Configuramos el entorno de compilación:

sudo ./configure --enable-optimizations

Iniciamos el proceso de compilación:

sudo make -j$(nproc)

Finalmente, una vez completada la compilación, instalamos Python:

sudo make altinstall

Importante, una vez que haya finalizado la instalación comprobar que efectivamente Python esta instalado y con la versión que deseamos.

python3.12 --version

### 1.2 Creación del directorio

Creamos un directorio donde añadiremos los diferentes scripts.

mkdir enrichmentscrip

Y haciendo uso del remote desktop, añadiremos los documentos deseados al directorio creado.

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente con confianza media

Finalmente hemos otorgado permisos completos a los documentos para que puedan ser utilizados ejecutando el comando:

sudo chmod 777 \*

### 1.3 Instalación de dependencias

Para que podamos recopilar los datos necesarios que pasaremos a la API de Fingerbank, es necesario analizar distintos campos que obtenemos mediante capturas de Wireshark. Es por ello por lo que debemos instalar tshark.

sudo apt install tshark

En la primera opción a elegir en la instalación, seleccionaremos Yes, y en la segunda seleccionaremos la opción por defecto.

Para poder usar todas las librerías requeridas en nuestro script, que se encuentran en el archivo requirements.txt, es necesario instalar el paquete pip.

sudo apt install pip

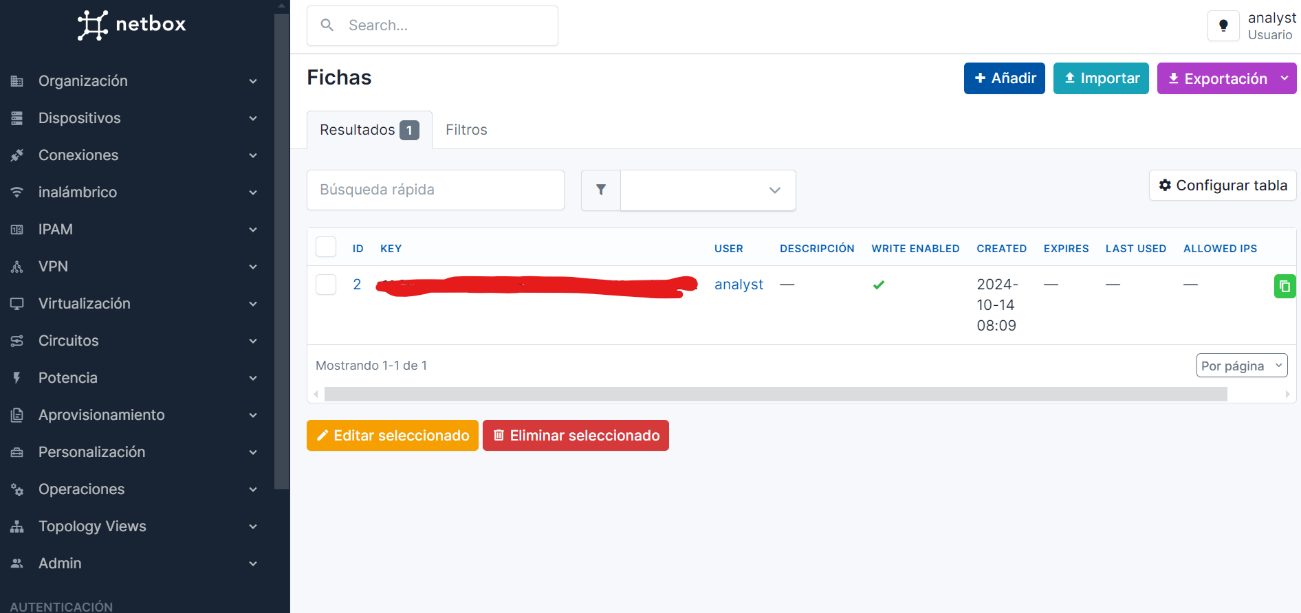
Una vez realizado este paso, instalaremos las configuraciones necesarias de nuestro archivo mediante el siguiente comando.

pip install -r requirements.txt

### 1.4 Creación de token y herramientas

Antes de poner a funcionar nuestro servicio tendremos que configurarlo, y para ello necesitamos varias cosas.

* En primer lugar, un token de Fingerbank, que tendremos que obtener de su página web: <https://www.fingerbank.org/>
* Por otro lado, tendremos que generar un token API en Netbox, dentro de la suite de Malcolm. Para ello, entramos en Netbox y nos dirigimos a Admin->Tokens de API:



Aquí pulsamos sobre Añadir, elegimos analyst como usuario y generamos el token con escritura habilitada. Copiamos la clave para tenerla disponible cuando la necesitemos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Teams

Descripción generada automáticamente

* También tendremos que generar dos etiquetas que marcarán los datos enriquecidos y consultados a fingerbank. Para ello nos dirigimos en Netbox a Personalización->Tags, y creamos dos etiquetas con slug “enriquecido” y “fingerbank”:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

### 1.5 Archivo config.ini

Una vez realizado estos pasos, actualizaremos el archivo **config.ini**, donde añadiremos el API Token de netbox y la ruta donde se encuentra tshark. A continuación, se procede a explicar los distintos parámetros de este archivo.

**[GENERAL]  
MALCOLM\_IP**: La IP donde se encuentra desplegado el servicio de Malcolm.

**LOG\_LEVEL**: Número que indica el tipo de información mostrada en los logs.  
0-NOTSET, 1-DEBUG, 2-INFO, 3-WARNING, 4-ERROR, 5-CRITICAL.

**[NETBOX]**  
**TOKEN**: El token de netbox creado previamente.

**[NGINX]**  
Usuario y contraseña para el proxy NGINX. Este paso se realizó en la instalación de Malcolm.  
**USER**: analyst  
**PASSWORD**: M!G4ever

**[ARKIME]**  
**BACKTIME\_SEARCH** = -1. En este campo indicamos la antigüedad de los logs de Arkime que vamos a analizar. Con el valor -1, procedemos a analizar desde el más antiguo.  
**PCAP\_SESSIONS\_SIZE** = 50. Este campo indica el número de logs que obtenemos para analizar cada vez que se ejecuta el script.

**[tshark]**  
Indicamos la ruta donde se encuentra tshark instalado  
**tshark\_path** = /usr/bin/tshark

**[FINGERBANK]**  
**TOKEN** = El token de la API de fingerbank.

**[ENRICHMENT]**  
Este campo indica el número de activos que procedemos a analizar descartando aquellos que ya tengan la etiqueta Enriquecido y Fingerbank.  
**BATCH\_SIZE** = 30

Consta añadir que el número máximo de peticiones por hora que podemos realizar a la API de Fingerbank es de 300 peticiones.

### 1.6 Crontab

Por último, con toda la configuración realizada, solo quedaría añadir nuestro servicio al crontab, de forma que se ejecute automáticamente cada cierto tiempo, en nuestro caso hemos considerado ejecutarlo cada hora, para ello.

* Creamos un script en shell que ejecute el programa de Python desde el directorio adecuado y que será el que ejecutemos desde cron. En nuestro caso lo llamaremos **exec.sh** y lo vamos a situar en la carpeta en la que se encuentran el resto de los scripts:

nano /home/mig/enrichmentscript/exec.sh

dentro del fichero escribimos:

cd /home/mig/enrichmentscript  
python3 enrich\_script.py

por último, le damos permisos de ejecución:

chmod +x exec.sh

* Finalmente abrimos crontab:

crontab -e

y añadimos la línea:

0 \* \* \* \* /bin/bash /home/mig/enrichmentscript/exec.sh >> /home/mig/enrichmentscript/enrichmentscript.log 2>&1

Con esto queda listo el despliegue y configuración del script de enriquecimiento, los logs que genera quedarán registrados (de acuerdo al nivel de log configurado) en /home/mig/enrichmentscript/enrichmentscript.log.

# Gestión del Disco Duro

Para que Malcolm pueda funcionar durante largos periodos de tiempos, es importante hacer una correcta gestión de los datos, de forma que los más antiguos se vayan borrando conforme dejan de ser necesarios.

Para ello, en la documentación de Malcolm se proponen una serie ajustes, que son los que vamos a realizar: <https://malcolm.fyi/docs/malcolm-config.html#DiskUsage>

La eliminación de archivos PCAP en Malcolm se configura a través de variables de entorno en el archivo **arkime.env**:

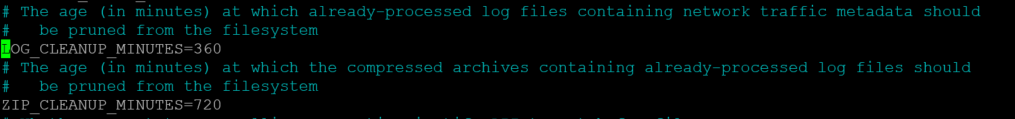
* **MANAGE\_PCAP\_FILES**: Si está en true, Arkime eliminará los archivos PCAP más antiguos cuando el espacio de almacenamiento sea insuficiente (por defecto está en false).
* **ARKIME\_FREESPACEG**: Define el porcentaje de espacio libre deseado antes de eliminar archivos PCAP, por ejemplo, 5% significa eliminar cuando el espacio libre cae por debajo de ese valor (por defecto 10%).

Texto

Descripción generada automáticamente

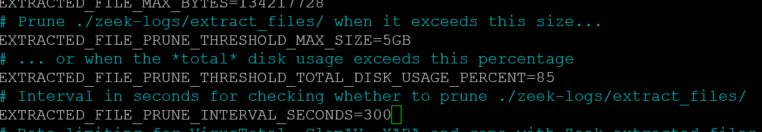
Los logs de Zeek y Suricata se almacenan temporalmente y se eliminan según estas variables en filebeat.env:

* **LOG\_CLEANUP\_MINUTES**: Tiempo en minutos para eliminar logs procesados.
* **ZIP\_CLEANUP\_MINUTES**: Tiempo en minutos para eliminar archivos comprimidos de logs procesados.



Los archivos extraídos por Zeek se eliminan según las siguientes variables en zeek.env:

* **EXTRACTED\_FILE\_PRUNE\_THRESHOLD\_MAX\_SIZE**: Tamaño máximo que puede tener la carpeta extract\_files/ antes de activar la eliminación.
* **EXTRACTED\_FILE\_PRUNE\_THRESHOLD\_TOTAL\_DISK\_USAGE\_PERCENT**: Porcentaje máximo de uso del disco que puede tener la carpeta antes de activar la eliminación.
* **EXTRACTED\_FILE\_PRUNE\_INTERVAL\_SECONDS**: Intervalo entre verificaciones de eliminación (por defecto 300 segundos).



La gestión de índices en OpenSearch o Elasticsearch puede controlarse con el plugin adecuado y la variable:

* **OPENSEARCH\_INDEX\_SIZE\_PRUNE\_LIMIT**: Define el tamaño máximo que pueden ocupar los índices antes de eliminar los más antiguos (por ejemplo, 500 GB o un porcentaje del disco).

Texto

Descripción generada automáticamente

Puedes convertir toda adversidad en algo provechoso para ti con sólo pensar. – Robert Jordan