

Rapport de laboratoire



Aysan Hakan

Table des matières

Laboratoire 2 : Ecosystème de logs	4
Linux :	4
Configuration LogHost:	4
Configuration de la machine Agent :	4
Windows :	5
Configuration de la machine agent :	5
Laboratoire 3 : Corrélations	6
3.1 Mode promiscuous	6
3.2 Tentatives de connexion [SSH / Session via écran]	8
3.3 Connexions multiples [SSH]	11
3.4 Disponibilité d'applications [Daemon]	13
3.5 Interdire l'administration à distance [SSH]	16
3.6 Interdire la connexion en cascade [SSH]	19
3.7 Faille Sudo	21
	25
Laboratoire 4 : SNMP	26
Introduction	26
OID surveillés :	26
1. Utilisation du processeur	26
3. Espace disque	29
4. Réseau	30
6. Processus	31
Analyse des données obtenues	32
- RAM	35
- Disques	36
- Interfaces	37
Laboratoire 5 : Netflow	40
Scénario n°1 : L'agent C réalise des "speed test" via google	40
Scénario n°2 : L'agent A télécharge via sftp les fichiers volumineux depuis l'agent B	
43	
Scénario n°3 : L'agent B regarde des vidéos HD ou 4K sur youtube	46
Scénario n°4 : L'agent A lance un appel + messages sur Discord	48
Identification IP/Port qui ont échangé le plus gros volume	50
Bibliographie	52

Laboratoire 2 : Ecosystème de logs

Linux :

Configuration LogHost:

Le loghost désigne le serveur qui est configuré pour recevoir les messages de journal provenant d'autres serveurs ou ordinateurs. La configuration de rsyslog se trouve dans le fichier /etc/rsyslog.conf.

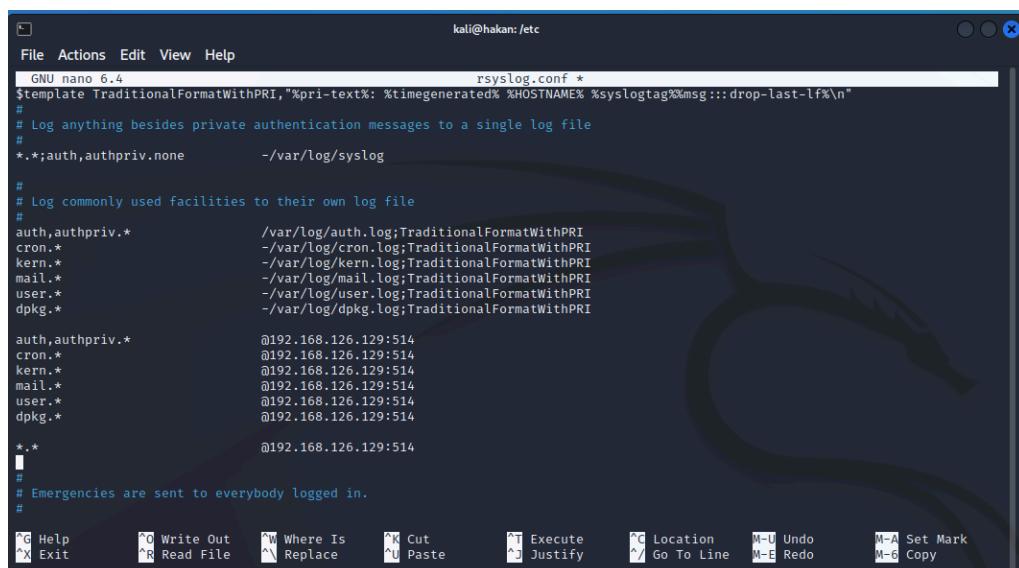
Il existe deux protocoles que vous pouvez utiliser pour envoyer/recevoir des fichiers journaux avec rsyslog : TCP et UDP.

Dans notre cas, nous allons utiliser le protocole UDP, pour se faire nous avons simplement à décommenter les lignes suivantes:

```
# provides UDP syslog reception
module(load="imudp")
input(type="imudp" port="514")
```

Configuration de la machine Agent :

L'agent est la machine qui envoie ses journaux à un serveur hôte de journaux distant. Pour le configurer, vous devez accéder au fichier situé dans /etc/rsyslog.conf et y ajouter les lignes suivantes. Ces lignes permettent de spécifier l'adresse IP de la machine distante vers laquelle les journaux doivent être envoyés, ainsi que le port associé.



```
File Actions Edit View Help
GNU nano 6.4                               rsyslog.conf *
$template TraditionalFormatWithPRI,"%pri-text% %timegenerated% %HOSTNAME% %syslogtag%%msg:::drop-last-lf%\n"
#
# Log anything besides private authentication messages to a single log file
#
*.*;auth,authpriv.none          -/var/log/syslog

#
# Log commonly used facilities to their own log file
#
auth,authpriv.*           /var/log/auth.log;TraditionalFormatWithPRI
cron.*                     -/var/log/cron.log;TraditionalFormatWithPRI
kern.*                     -/var/log/kern.log;TraditionalFormatWithPRI
mail.*                     -/var/log/mail.log;TraditionalFormatWithPRI
user.*                     -/var/log/user.log;TraditionalFormatWithPRI
dpkg.*                     -/var/log/dpkg.log;TraditionalFormatWithPRI

auth,authpriv.*           @192.168.126.129:514
cron.*                     @192.168.126.129:514
kern.*                     @192.168.126.129:514
mail.*                     @192.168.126.129:514
user.*                     @192.168.126.129:514
dpkg.*                     @192.168.126.129:514

*.*                         @192.168.126.129:514

#
# Emergencies are sent to everybody logged in.
#
^C Help      ^O Write Out    ^W Where Is     ^X Cut        ^E Execute   ^C Location   ^U Undo       ^A Set Mark
^X Exit      ^R Read File    ^P Replace     ^V Paste      ^J Justify   ^G Go To Line  ^E Redo       ^M-C Copy
```

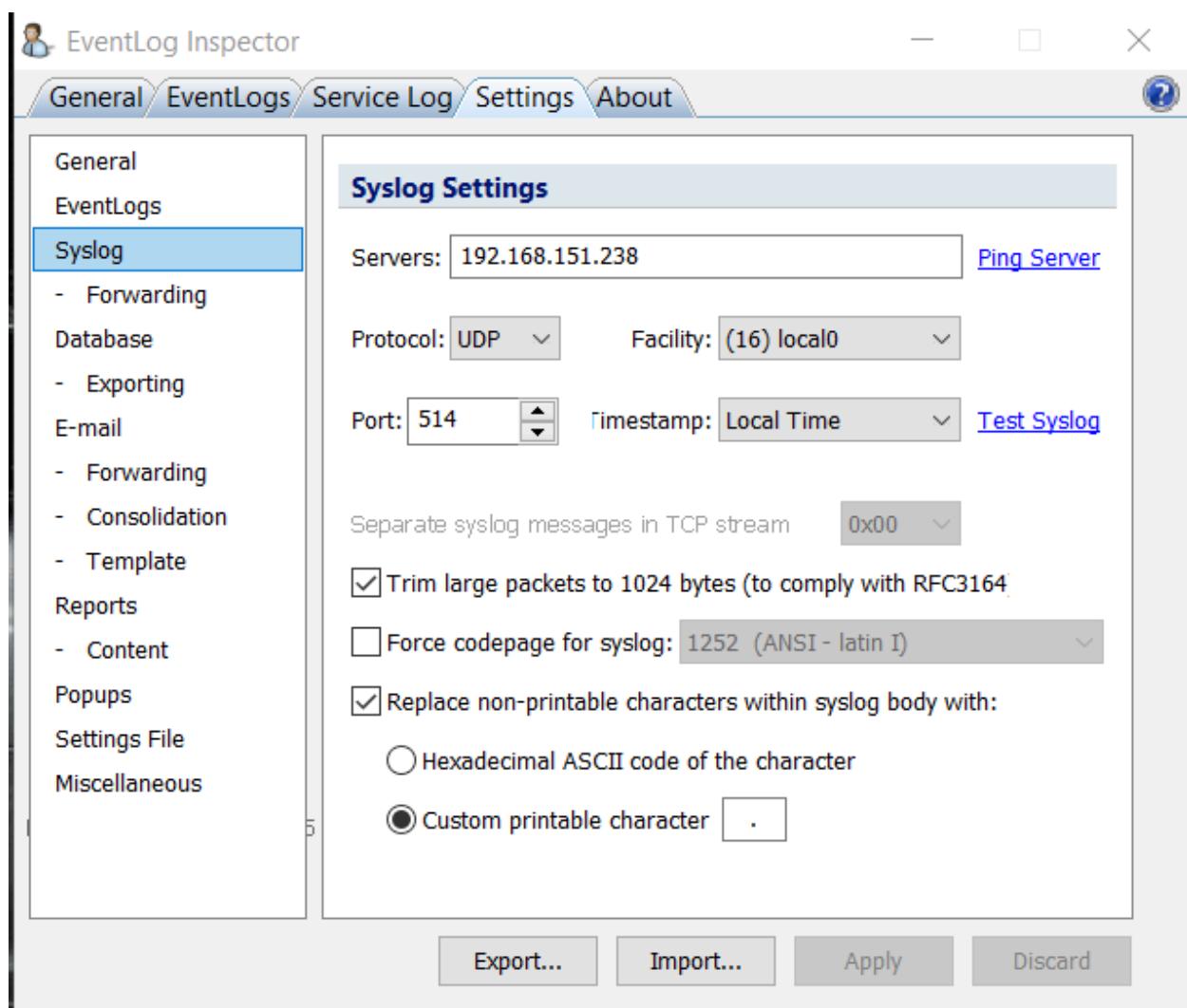
Windows :

Pour la centralisation des journaux sur Windows, nous avons utilisé l'application Event Log Inspector.

Pour la configuration du logHost, les mêmes manipulations que sous Linux ont été réalisées.

Configuration de la machine agent :

Pour configurer la machine agent, il nous suffisait de nous rendre dans les paramètres de syslog/settings et d'y saisir l'adresse IP du LogHost, le protocole utilisé ainsi que le port pour la centralisation des journaux.



Laboratoire 3 : Corrélations

3.1 Mode promiscuous

Dans ce scénario, notre objectif est de détecter quand une interface réseau passe en mode promiscuous, qui lui permet de recevoir tous les paquets du réseau, même ceux qui ne lui sont pas destinés (avec une adresse MAC différente). Ce mode est généralement utilisé par des outils de surveillance, mais en dehors de ces outils, il peut être associé à un sniffer qui analyse le trafic de votre réseau.

Comment simuler le promiscuous mode ?

Avant d'entamer la mise en place de la procédure de détection du mode promiscuous, nous allons activer le mode promiscuous sur la machine agent avec la commande :

```
└─(root㉿kali)-[~]
# ip link set eth0 promisc on
```

Lors de l'activation de ce mode, un message de log va être généré dans le fichier /var/log/kern.log nous alertant de l'activation du mode.

```
└─(kali㉿yas)-[~]
$ cat /var/log/kern.log
No    Time           Source          Destination       Protocol Length Info
kern.notice: Apr  5 17:44:57 yas kernel:[ 970.062374] cfg80211: Loading compiled-in X.509 certificates for regulatory database   0PR  269 M-SEARCH
kern.notice: Apr  5 17:44:57 yas kernel:[ 970.062576] cfg80211: Loaded X.509 cert 'benh@debian.org: 577e021cb980e0e820821ba7b54b4961b8b4fadf'
kern.notice: Apr  5 17:44:57 yas kernel:[ 970.062749] cfg80211: Loaded X.509 cert 'romain.perier@gmail.com: 3abbc6ec146e09d1b6016ab9d6cf71dd233f0328' ICMPv6 174 Router Ad
kern.notice: Apr  5 17:44:57 yas kernel:[ 970.062921] cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aeef9cea7' ICMPv6 139 Multicast
kern.info:  Apr  5 17:44:57 yas kernel:[ 970.066563] platform regulatory.0: firmware: direct-loading firmware regulatory.db SDP  269 M-SEARCH
kern.info:  Apr  5 17:44:57 yas kernel:[ 970.068123] platform regulatory.0: firmware: direct-loading firmware regulatory.db.p7s ICMPv6 130 Multicast
kern.info:  Apr  5 17:44:59 yas kernel:[ 971.587568] Bluetooth: Core ver 2.22
kern.info:  Apr  5 17:44:59 yas kernel:[ 971.587629] NET: Registered PF_BLUETOOTH protocol family 239.255.255.250 SSDP  269 M-SEARCH
kern.info:  Apr  5 17:44:59 yas kernel:[ 971.587632] Bluetooth: HCI device and connection manager initialized 1 ICMPv6 174 Router Ad
kern.info:  Apr  5 17:44:59 yas kernel:[ 971.587636] Bluetooth: HCI socket layer initialized 00:00:00:00:00:00 1 ICMPv6 130 Multicast
kern.info:  Apr  5 17:44:59 yas kernel:[ 971.587638] Bluetooth: L2CAP socket layer initialized 00:00:00:00:00:00 1 ICMPv6 130 Multicast
kern.info:  Apr  5 17:44:59 yas kernel:[ 971.587643] Bluetooth: SCO socket layer initialized 00:00:00:00:00:00 1 ICMPv6 130 Multicast
kern.info:  Apr  5 17:45:00 yas kernel:[ 973.488992] device eth0 entered promiscuous mode 00:0c:29:01:31:174 31 33 00 00 00 01 02 10 18 4F 65
```

C'est donc ce que nous allons tenter de détecter avec l'aide d'une règle SEC.

Création d'une règle SEC pour détecter le mode promiscuous:

Nous pouvons maintenant créer une règle SEC qui va détecter le mode promiscuous.

```
1 type=Single
2 ptype=RegExp
3 pattern=(\d+-\d+-\d+T\d+:\d+:\d+. \d+\+\d+:\d+ )(\w+) (\w+:\) \[ \d+. \d+] device (\w+) entered promiscuous mode
4 desc=$0
5 action=write /var/log/stock.log $4 entered promiscuous $2;
6
```

Cette règle SEC va utiliser une Regex qui va chercher les log correspondant au pattern fourni, une fois le log trouver, un message va être généré et sera envoyé dans le fichier stock.log avec le format spécifié dans la règle.

Une fois la règle créée , nous allons utiliser la commande pour vérifier que la règle SEC fonctionne correctement et retourne bien ce que nous désirons:

```
(kali㉿kali)-[~/Documents/sec]
└─$ sudo sec --conf=regex1 --input=/var/log/kern.log
SEC (Simple Event Correlator) 2.9.1
Reading configuration from regex1
1 rules loaded from regex1
No --bufsize command line option or --bufsize=0, setting --bufsize to 1
Opening input file /var/log/kern.log
Interactive process, SIGINT can't be used for changing the logging level
```

Cette commande va permettre d'analyser le fichier log, « /var/log/kern.log » à l'aide de l'expression régulière créer dans le fichier de configuration « regex1 ». Grâce à l'outil, Simple Event Correlator (SEC), qui va rechercher des occurrences de la regex définies dans les logs (/var/log/kern.log dans ce cas) .

Si la règle est opérationnelle, elle affichera le message "x rules loaded from..." pour indiquer le nombre de règles chargées à partir de l'emplacement spécifié. De plus, si l'agent est passé en mode promiscuous, un événement sera enregistré dans le fichier spécifié.

```
(kali㉿kali)-[~/Documents/sec]
└─$ sudo sec --conf=regex1 --input=/var/log/kern.log
SEC (Simple Event Correlator) 2.9.1
Reading configuration from regex1
1 rules loaded from regex1
No --bufsize command line option or --bufsize=0, setting --bufsize to 1
Opening input file /var/log/kern.log
Interactive process, SIGINT can't be used for changing the logging level
Writing event 'eth0 entered promiscuous kali' to file '/var/log/stock.log'
```

En quoi la règle évite-t-elle les faux-positifs?

La règle ne générera pas de faux positif car le pattern fourni pour la regex vise uniquement l'entrée en mode promiscuous d'un utilisateur.

3.2 Tentatives de connexion [SSH / Session via écran]

Ce scénario concerne la détection d'attaques de connexion SSH infructueuses ou de tentatives de session via écran. L'objectif est de détecter lorsque le nombre de tentatives de connexion infructueuses dépasse un seuil donné dans une fenêtre de temps spécifiée et de prendre une mesure de sécurité appropriée en conséquence.

Adaptation apportées à la configuration Rsyslog :

```
$template TentConTemp, "/var/log/hakan/connexion.log"  
:  
:msg, contains, "(lightdm:auth): authentication failure" ?TentConTemp  
:msg, contains, "(lightdm:session): session opened" ?TentConTemp
```

Ces règles spécifie que les messages de log contenant la chaîne de caractères "(lightdm:auth): authentication failure" et "(lightdm:session): session opened" seront enregistrés dans le fichier de log défini par le template "TentConTemp".

Comment simuler les tentatives de connexion (SSH/ session via écran) ?

Pour simuler ce scénario, il faudrait générer délibérément cinq tentatives de connexion infructueuses à un serveur SSH ou des tentatives de session via écran dans une fenêtre de temps de 30 secondes.

Création d'une règle SEC pour détecter les tentatives de connexion:

La détection du scénario repose sur la surveillance des logs de connexion. La règle basée sur une expression régulière recherche le motif "failure" dans les logs de connexion. Si ce motif est trouvé au moins cinq fois dans une fenêtre de temps de 30 secondes, l'action spécifiée sera déclenchée. L'action consiste à enregistrer un message dans le fichier /var/log/hakan/connexion.log, indiquant le nombre de tentatives de connexion infructueuses détectées.

```
GNU nano 6.4                               connexion.conf *
```

```
type=SingleWithThreshold
ptype=RegExp
desc=Test
pattern=failure
window=30
thresh=5
action=write /var/log/hakan/connexion.log 5 Tentatives de connexions échouées en 30 secondes !!
```

^G Help ^O Write Out ^W Where Is ^T Execute
^X Exit ^R Read File ^\ Replace ^K Cut ^J Justify ^C Location
^/ Go To Line

Le scénario utilise une règle de détection basée sur des expressions régulières. La règle est configurée comme suit :

Type : SingleWithThreshold, ce qui signifie qu'il s'agit d'une règle unique avec seuil.

Window : 30 secondes, la fenêtre de temps pendant laquelle les événements de log seront collectés pour la détection d'attaque.

Thresh : 5, le nombre minimum d'événements correspondant au motif ("failure") qui doivent être détectés dans la fenêtre de temps spécifiée pour déclencher l'action de sécurité.

Action : write = écriture en console

Pour pouvoir autoriser la règle sec à écrire dans le fichier connexion.log on lui donne les droit d'écriture.

```
(kali㉿loghost)-[~/var/log/hakan]
$ sudo chmod a+w /var/log/hakan/connexion.log
```

Exécution et résultat de la règle:

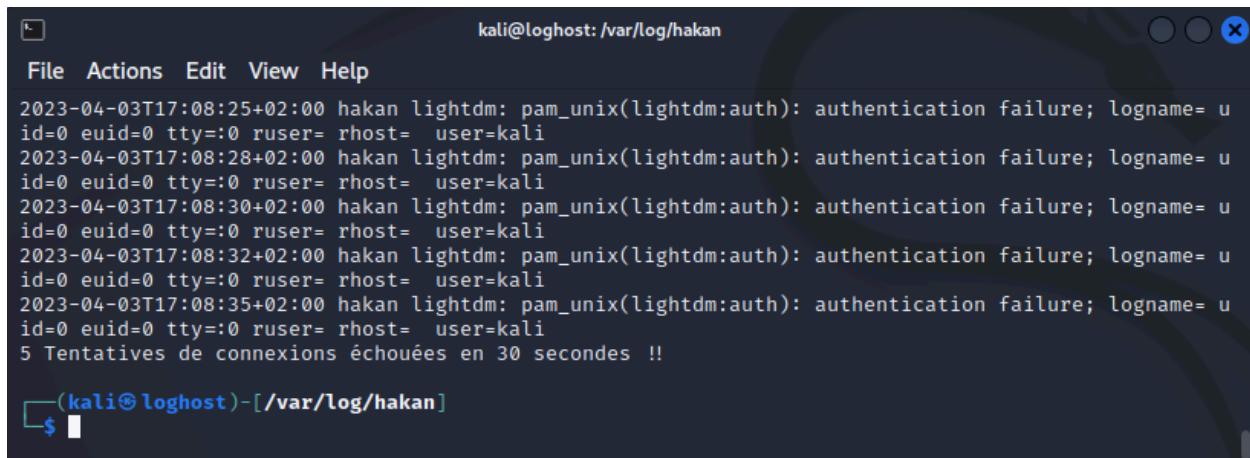
Pour lancer la surveillance:

```
(kali㉿loghost)-[~/var/log/hakan]
$ sec -conf=/etc/sec/connexion.conf --input=/var/log/hakan/connexion.log
```

Voici comment la règle réagit:

```
(kali㉿loghost)-[~/var/log/hakan]
$ sec -conf=/etc/sec/connexion.conf --input=/var/log/hakan/connexion.log
SEC (Simple Event Correlator) 2.9.1
Reading configuration from /etc/sec/connexion.conf
1 rules loaded from /etc/sec/connexion.conf
No --bufsize command line option or --bufsize=0, setting --bufsize to 1
Opening input file /var/log/hakan/connexion.log
Interactive process, SIGINT can't be used for changing the logging level
Writing event '5 Tentatives de connexions échouées en 30 secondes !!' to file '/var/log/hakan/connexion.log'
```

Et nous pouvons observer le résultat dans le fichier /var/log/hakan/connexion.log:



```
kali@loghost: /var/log/hakan
File Actions Edit View Help
2023-04-03T17:08:25+02:00 hakan lightdm: pam_unix(lightdm:auth): authentication failure; logname= u
id=0 euid=0 tty=:0 ruser= rhost=
2023-04-03T17:08:28+02:00 hakan lightdm: pam_unix(lightdm:auth): authentication failure; logname= u
id=0 euid=0 tty=:0 ruser= rhost=
2023-04-03T17:08:30+02:00 hakan lightdm: pam_unix(lightdm:auth): authentication failure; logname= u
id=0 euid=0 tty=:0 ruser= rhost=
2023-04-03T17:08:32+02:00 hakan lightdm: pam_unix(lightdm:auth): authentication failure; logname= u
id=0 euid=0 tty=:0 ruser= rhost=
2023-04-03T17:08:35+02:00 hakan lightdm: pam_unix(lightdm:auth): authentication failure; logname= u
id=0 euid=0 tty=:0 ruser= rhost=
5 Tentatives de connexions échouées en 30 secondes !!
(kali㉿loghost)-[~/var/log/hakan]
$
```

En quoi la règle évite-t-elle les faux-positifs?

La règle de détection basée sur l'expression régulière "failure" vise à filtrer les tentatives de connexion infructueuses spécifiquement. Cela limite les faux positifs, car seules les tentatives de connexion avec les chaînes de caractères "(lightdm:auth): authentication failure" et "(lightdm:session): session opened" qui sont des connexions de session via écran seront stocker dans notre fichier connexion.log.

3.3 Connexions multiples [SSH]

Dans ce scénario, l'objectif est de détecter si une machine se connecte plusieurs fois à une autre machine en utilisant le même protocole. Cette situation peut indiquer une attaque de type déni de service (DoS) ou une activité suspecte.

Adaptation apportées à la configuration Rsyslog :

Pour le cas d'une connexion multiples SSH, nous avons adaptés notre configuration rsyslog afin de rediriger log liés aux connexions SSH, y compris les connexions réussies avec mots de passe, les sessions fermées et les échecs d'authentification, vers un fichier de journalisation spécifique pour nous faciliter l'analyse de ces logs de connexions.

```
# Save remote logs with ssh connections
$template connexion_ssh, "/var/log/hakan/connexion_ssh.log"
:msg, regex,"Accepted password for .* from"?connexion_ssh
:msg, regex,"sshd:session.* session closed"?connexion_ssh
:msg, regex,"authentication failure"?connexion_ssh
```

Comment simuler ce scénario ?

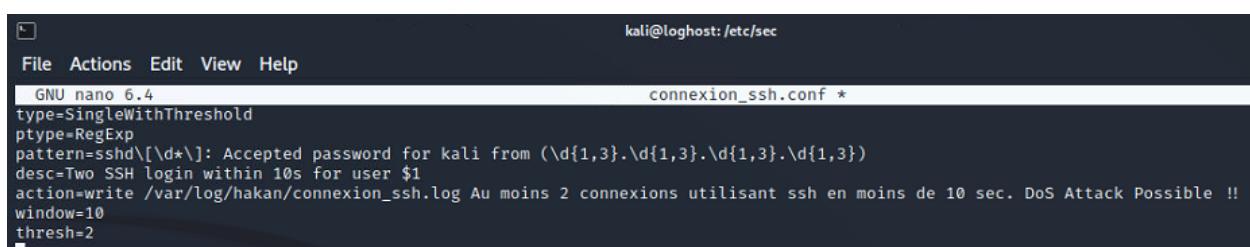
Pour simuler ce scénario il nous suffit d'exécuter cette commande plus d'une fois afin de réaliser une connexion multiple sur une ip cible:

```
ssh user@adresse ip cible
```

La trace de cette connexion se trouvera dans le fichier "connexion_ssh.log" déterminer dans la template plus haut.

Création d'une règle SEC pour détecter les connexions multiples:

Nous avons mis en place une règle SEC qui va alerter lorsqu'il identifie une connexion multiple (+/- 2 connexions).



```
kali@loghost: /etc/sec
File Actions Edit View Help
GNU nano 6.4
type=SingleWithThreshold
ptype=RegExp
pattern=sshd\[.\d*\]: Accepted password for kali from (\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3})
desc=Two SSH login within 10s for user $1
action=write /var/log/hakan/connexion_ssh.log Au moins 2 connexions utilisant ssh en moins de 10 sec. DoS Attack Possible !!
window=10
thresh=2
```

Explication de la règle :

type=SingleWithThreshold: détecter des occurrences uniques d'un modèle spécifique avec un seuil défini.

ptype=RegExp: Utilisation d'une expression régulière

action=write: écriture en console

window=10: Ce paramètre définit la fenêtre de temps en secondes pendant laquelle la règle recherche les correspondances du modèle

thresh=2: Ce paramètre spécifie le seuil, c'est-à-dire le nombre minimum de correspondances du modèle requis pour déclencher la règle.

Exécution et résultat de la règle:

Nous allons maintenant tester la règle et simuler une connexion multiple afin de voir si elle s'exécute correctement avec cette commande :

```
(kali㉿loghost)-[~/var/log/hakan]
$ sec -conf=/etc/sec/connexion_ssh.conf --input=/var/log/hakan/connexion_ssh.log
SEC (Simple Event Correlator) 2.9.1
Reading configuration from /etc/sec/connexion_ssh.conf
1 rules loaded from /etc/sec/connexion_ssh.conf
No --bufsize command line option or --bufsize=0, setting --bufsize to 1
Opening input file /var/log/hakan/connexion_ssh.log
Interactive process, SIGINT can't be used for changing the logging level
Writing event 'Au moins 2 connexions utilisant ssh en moins de 10 sec. DoS Attack Possible !!' to file '/var/log/hakan/connexion_ssh.log'
```

On peut apercevoir que la règle à bien été chargée et qu'elle a écrit un event dans le fichier connexion_ssh.log en détectant la connexion multiple ssh.

En quoi la règle évite-t-elle les faux-positifs?

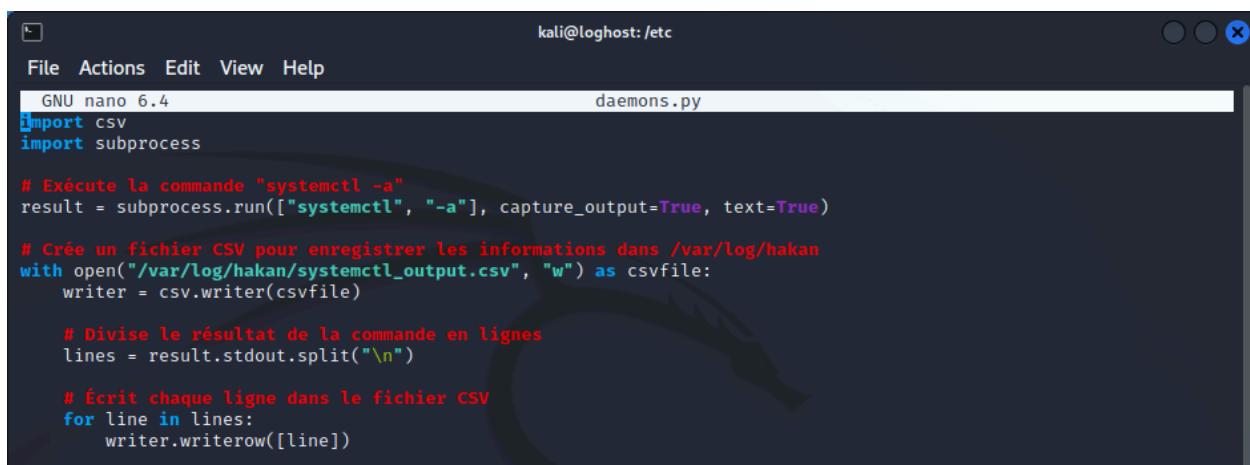
Utilisation d'une expression régulière : cible un log précis ce qui va rendre compliqué l'apparition de faux-positif.

Utilisation d'un seuil fixe : une seule paire de connexions pourrait déclencher la règle.

3.4 Disponibilité d'applications [Daemon]

Ce scénario vise à assurer la disponibilité des applications clés en surveillant leurs arrêts, démarrages et redémarrages. Il consiste à maintenir un fichier d'état des applications, indiquant si elles sont en cours d'exécution ou non, ainsi qu'un fichier d'historique des observations. Cette surveillance permet de prendre des mesures immédiates pour résoudre les problèmes et minimiser les interruptions de service, assurant ainsi la continuité des opérations et la satisfaction des utilisateurs finaux.

Création d'un script:



```
File Actions Edit View Help
GNU nano 6.4                               daemons.py
Import CSV
Import subprocess

# Exécute la commande "systemctl -a"
result = subprocess.run(["systemctl", "-a"], capture_output=True, text=True)

# Crée un fichier CSV pour enregistrer les informations dans /var/log/hakan
with open("/var/log/hakan/systemctl_output.csv", "w") as csvfile:
    writer = csv.writer(csvfile)

    # Divise le résultat de la commande en lignes
    lines = result.stdout.split("\n")

    # Écrit chaque ligne dans le fichier CSV
    for line in lines:
        writer.writerow([line])
```

Ce script Python utilise le module subprocess pour exécuter la commande systemctl -a dans le système. Cette commande retourne l'état de tous les services système sous Linux qui utilisent systemd comme système d'initialisation. Le résultat est ensuite écrit dans un fichier CSV.

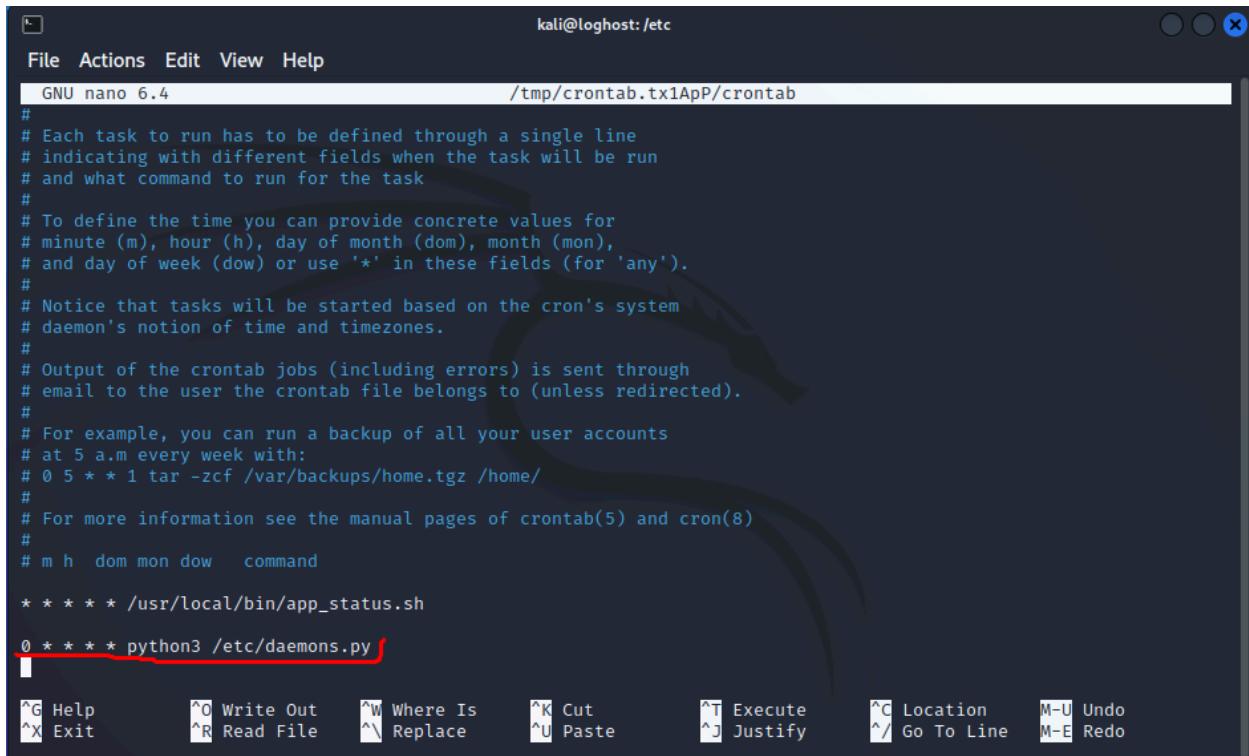
Voici le détail des opérations du script :

La commande systemctl -a est exécutée à l'aide de la méthode subprocess.run(). Les arguments capture_output=True et text=True permettent de capturer la sortie standard (stdout) de la commande sous forme de texte.

Le script crée un nouveau fichier CSV dans le dossier /var/log/hakan/ nommé systemctl_output.csv. Ce fichier est ouvert en mode écriture ("w").

Le script écrit ensuite chaque ligne de la sortie standard de la commande systemctl -a dans le fichier CSV. Pour cela, il divise d'abord la sortie de la commande en lignes avec la méthode split("\n").

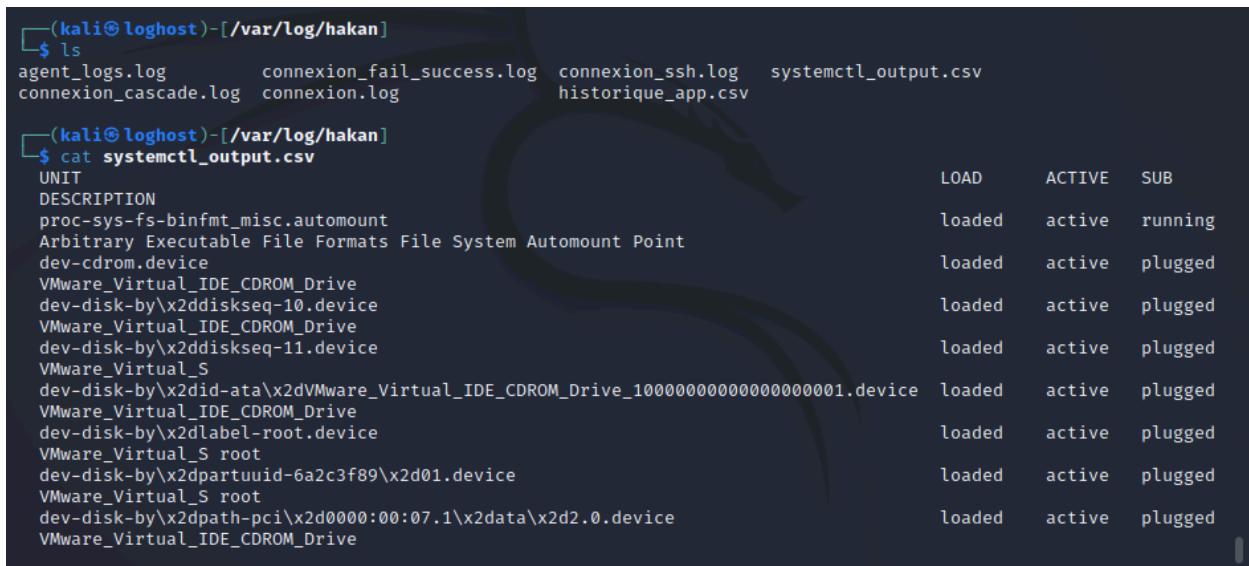
Chaque ligne est ensuite écrite dans le fichier CSV à l'aide de la méthode writerow(). Chaque ligne est mise dans une liste [line], car writerow() attend une liste d'éléments à écrire.



```
kali@loghost: /etc
File Actions Edit View Help
GNU nano 6.4
/tmp/crontab.tx1ApP/crontab
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
* * * * * /usr/local/bin/app_status.sh
0 * * * * python3 /etc/daemons.py
```

Cette règle crontab exécute le script Python nommé daemons.py qui se trouve dans le répertoire /etc/ à chaque début d'heure.

En sortie de notre script et notre règle crontab nous avons le résultat suivant :



UNIT	LOAD	ACTIVE	SUB
proc-sys-fs-binfmt_misc.automount	loaded	active	running
Arbitrary Executable File Formats File System Automount Point			
dev-cdrom.device	loaded	active	plugged
VMware_Virtual_IDE_CDROM_Drive	loaded	active	plugged
dev-disk-by\x2ddiskseq-10.device	loaded	active	plugged
VMware_Virtual_IDE_CDROM_Drive	loaded	active	plugged
dev-disk-by\x2ddiskseq-11.device	loaded	active	plugged
VMware_Virtual_S	loaded	active	plugged
dev-disk-by\x2did-ata\x2dVMware_Virtual_IDE_CDROM_Drive_10000000000000000000000000000001.device	loaded	active	plugged
VMware_Virtual_IDE_CDROM_Drive	loaded	active	plugged
dev-disk-by\x2dlabel-root.device	loaded	active	plugged
VMware_Virtual_S root	loaded	active	plugged
dev-disk-by\x2dpartition-6a2c3f89\x2d01.device	loaded	active	plugged
VMware_Virtual_S root	loaded	active	plugged
dev-disk-by\x2dpath-pci\x2d0000:00:07.1\x2data\x2d2.0.device	loaded	active	plugged
VMware_Virtual_IDE_CDROM_Drive	loaded	active	plugged

2ème partie - Détection de changement de statut



```
kali@loghost: /var/log/hakan
File Actions Edit View Help
GNU nano 6.4
import csv
from datetime import datetime
csv_file = "systemctl_output.csv"

# Lire le fichier CSV
with open(csv_file, "r") as csvfile:
    reader = csv.reader(csvfile)
    data = list(reader)

# Parcourir chaque ligne du fichier CSV
for row in data:
    # Récupérer les données de la ligne
    unit, load, active, sub, description = row

    # Lire l'historique des changements d'état à partir du fichier d'historique CSV
    with open("history.csv", "a+") as history_file:
        history_reader = csv.reader(history_file)
        rows = list(history_reader)
        rows.append(["Unit", "Date", "Heure", "Etat"])

        # Si le fichier d'historique est vide, ajoute une ligne d'en-tête
        if not rows:
            rows.append(["Unit", "Date", "Heure", "Etat"])

        # Récupère le dernier état enregistré dans l'historique pour cette unité
        last_status = None
        for row1 in rows[-1:-1]:
            if row1[0] == unit:
                last_status = row1[3]
                break

        # Si l'état actuel est différent de l'état précédent, enregistre le changement dans l'historique
        if active != last_status:
            date = datetime.now().strftime("%Y-%m-%d")
            heure = datetime.now().strftime("%H:%M:%S")
            rows.append([unit, date, heure, active])

        # Écrit les changements dans le fichier CSV
        history_writer = csv.writer(history_file)
        history_writer.writerows(rows[-1:])

        # Affiche un message d'information avec la date et l'heure du changement
        if active == "active":
            print(f'L\'unité {unit} est active à {heure} le {date}.')
        else:
            print(f'L\'unité {unit} n\'est pas active à {heure} le {date}.')
systemctl_final.py *
```

Ce script analyse chaque ligne du fichier CSV donné et suit l'état de chaque unité. S'il détecte un changement d'état, il enregistre le changement dans un fichier CSV d'historique et affiche un message.

Voici la règle crontab pour exécuter le script chaque minute: * * * * * python3 /var/log/hakan/systemctl_final.py

3.5 Interdire l'administration à distance [SSH]

Comment simuler ce scénario ?

Etape 1 : Connexion ssh sur une machine cible

```
ssh user@adresse_ip_cible
```

Etape 2: Executer des commandes en mode administrateurs

```
sudo su
```

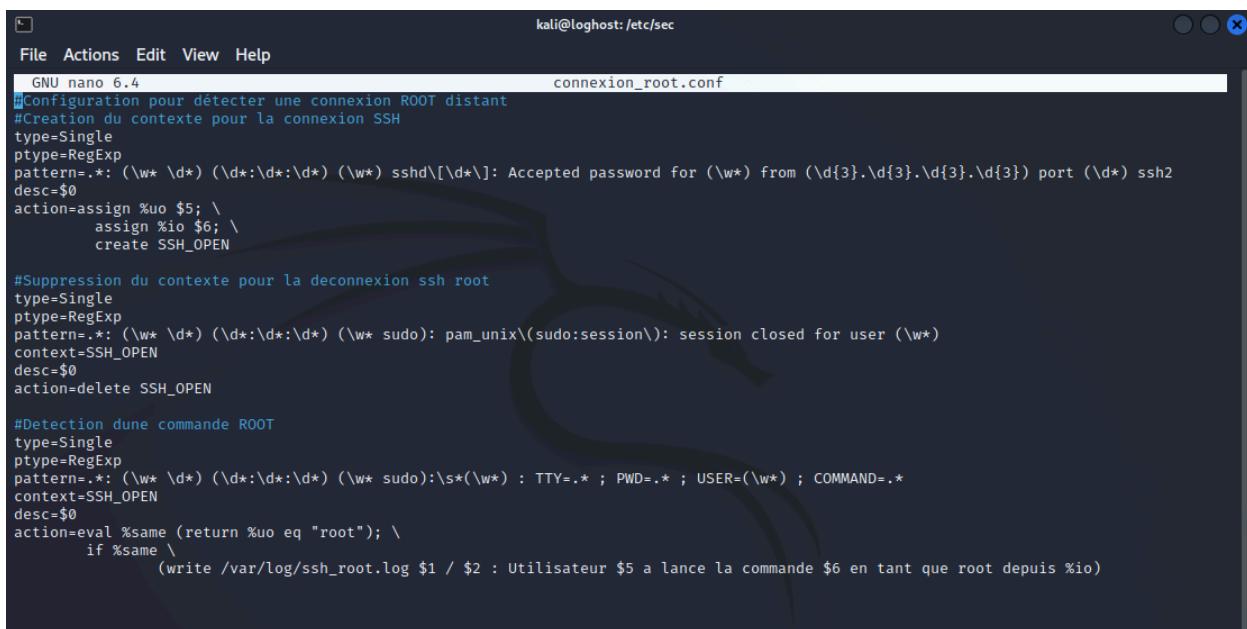
Voici les logs générés par ces actions:

```
auth.info: Apr 12 17:29:15 loghost sshd[119527]: Accepted password for kali from 192.168.126.128 port 56016 ssh2  
authpriv.info: Apr 12 17:29:15 loghost sshd[119527]: pam_unix(sshd:session): session opened for user kali(uid=1000) by (uid=0)
```

```
authpriv.notice: Apr 12 17:29:55 loghost sudo:      kali : TTY=pts/1 ; PWD=/home/kali ; USER=root ; COMMAND=/usr/bin/su  
authpriv.info: Apr 12 17:29:55 loghost sudo: pam_unix(sudo:session): session opened for user root(uid=0) by kali(uid=1000)  
auth.notice: Apr 12 17:29:55 loghost su[119760]: (to root) root on pts/2  
authpriv.info: Apr 12 17:29:55 loghost su[119760]: pam_unix(su:session): session opened for user root(uid=0) by kali(uid=0)
```

Création d'une règle SEC pour détecter l'administration à distance:

Nous allons donc créer une règle SEC sur base des logs qui ont été générés lors de la simulation.



```
File Actions Edit View Help
GNU nano 6.4                               connexion_root.conf
#Configuration pour détecter une connexion ROOT distant
#Creation du contexte pour la connexion SSH
type=Single
ptype=RegExp
pattern=.*: (\w* \d*) (\d*:\d*:\d*) (\w*) sshd\[.\d*\]: Accepted password for (\w*) from (\d{3}.\d{3}.\d{3}).\d{3}) port (\d*) ssh2
desc=$0
action=assign %uo $5; \
            assign %io $6; \
            create SSH_OPEN

#Suppression du contexte pour la deconnexion ssh root
type=Single
ptype=RegExp
pattern=.*: (\w* \d*) (\d*:\d*:\d*) (\w* sudo): pam_unix\(\sudo:session\): session closed for user (\w*)
context=SSH_OPEN
desc=$0
action=delete SSH_OPEN

#Detection d'une commande ROOT
type=Single
ptype=RegExp
pattern=.*: (\w* \d*) (\d*:\d*:\d*) (\w* sudo):\s*(\w*) : TTY=.* ; PWD=.* ; USER=(\w*) ; COMMAND=.*
context=SSH_OPEN
desc=$0
action=eval %same (return %uo eq "root"); \
            if %same \
                (write /var/log/ssh_root.log $1 / $2 : Utilisateur $5 a lancé la commande $6 en tant que root depuis %io)
```

Explication de la règle :

La règle est divisée en trois parties distinctes, chacune étant destinée à détecter un aspect spécifique des connexions SSH en tant que superutilisateur (root) :

1. Configuration pour détecter une connexion ROOT distante :

Utilisation d'une expression régulière.

Si le pattern correspond à un événement d'acceptation de mot de passe pour une connexion SSH en tant que root, la règle est déclenchée.

En utilisant les actions "assign" et "create", les valeurs capturées sont assignées à des variables ("%uo" pour le nom d'utilisateur et "%io" pour l'adresse IP source) et un contexte nommé "SSH_OPEN" est créé.

2. Suppression du contexte pour la déconnexion SSH root :

Elle utilise une expression régulière pour rechercher un motif correspondant à un événement de fermeture de session pour un utilisateur root.

Si le pattern correspond à un événement de fermeture de session pour un utilisateur root, la règle est déclenchée.

La règle utilise l'action delete pour supprimer le contexte "SSH_OPEN" créé lors de la connexion SSH root détectée précédemment.

3. Détection d'une commande ROOT :

Elle utilise une expression régulière pour rechercher un pattern correspondant à un événement de commande exécutée en tant que root.

Si le pattern correspond à un événement de commande exécutée en tant que root, la règle est déclenchée.

La règle utilise l'action eval pour évaluer une condition : si l'utilisateur qui a exécuté la commande est réellement root, alors la règle continue son exécution.

L'action write est utilisée pour enregistrer les informations de l'événement dans le fichier /var/log/ssh_root.log, y compris la date, l'heure, le nom d'utilisateur, la commande et l'adresse IP source.

Exécution et résultat de la règle:

Nous allons donc maintenant tester la règle, sur l'image on peut apercevoir que les 3 règles ont bien été chargée:

```
(kali㉿loghost)-[~/etc/sec]
$ sec -conf=/etc/sec/connexion_root.conf --input=/var/log/ssh_root.log
SEC (Simple Event Correlator) 2.9.1
Reading configuration from /etc/sec/connexion_root.conf
3 rules loaded from /etc/sec/connexion_root.conf
No --bufsize command line option or --bufsize=0, setting --bufsize to 1
Opening input file /var/log/ssh_root.log
Interactive process, SIGINT can't be used for changing the logging level
```

Lors d'une simulation du scénario, nous obtenons bien un message sur la console nous informant d'une administration à distance.

```
(kali㉿loghost)-[~/var/log]
$ cat ssh_root.log
Apr 13 / 00:37:52 : Utilisateur root a lancé la commande /usr/bin/su en tant que root depuis 192.168.126.128
```

En quoi la règle évite-t-elle les faux-positifs?

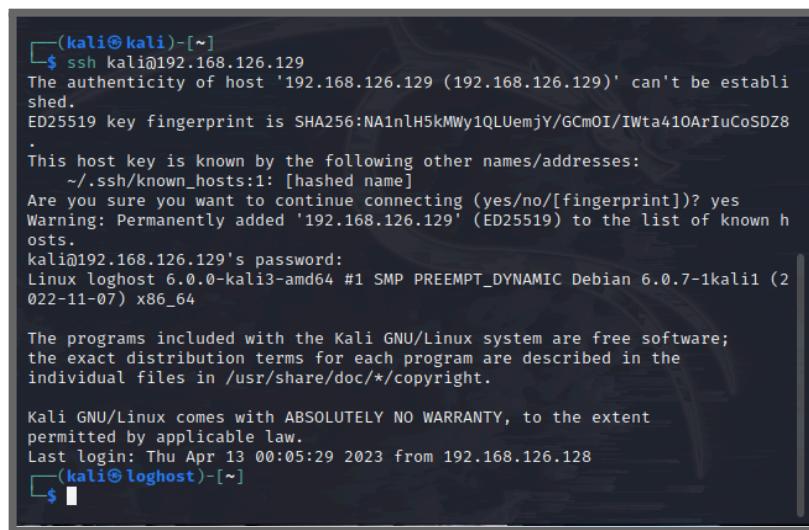
La règle utilise des contextes spécifiques et effectue des vérifications supplémentaires pour éviter les faux positifs. Les contextes restreignent l'application de la règle aux événements liés à une session SSH en tant que root, évitant ainsi les réactions inappropriées à d'autres événements. Les vérifications supplémentaires, telles que la vérification de l'utilisateur réel, permettent d'éviter l'enregistrement de commandes exécutées par d'autres utilisateurs en tant que root, réduisant ainsi les faux positifs.

3.6 Interdire la connexion en cascade [SSH]

Le scénario consiste à détecter et interdire les connexions en cascade dans le réseau informatique. Une connexion en cascade se produit lorsqu'un utilisateur tente de se connecter à un ordinateur distant en passant par un ordinateur intermédiaire. Cette pratique est considérée comme suspecte et peut indiquer une tentative d'accès non autorisé ou des intentions malveillantes.

Simulation du scénario :

Connexion ssh d'une machine temporaire [PC1] vers la machine LogHost[PC2]

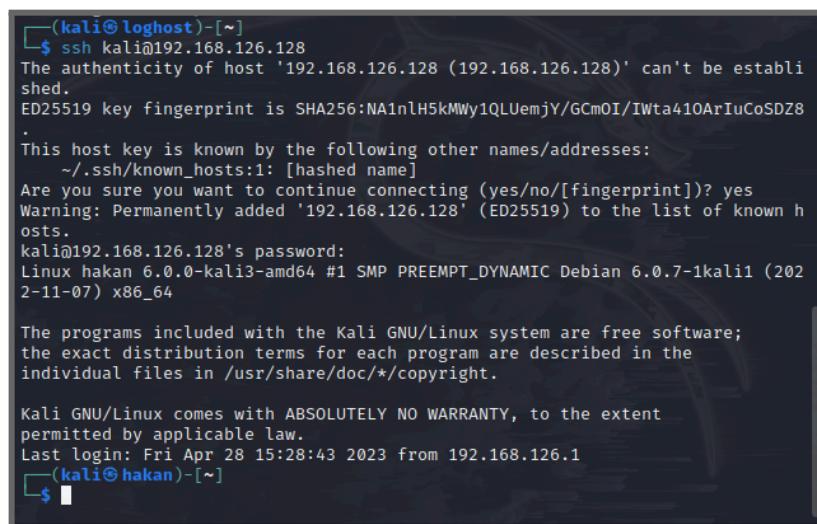


```
(kali㉿kali)-[~]
$ ssh kali@192.168.126.129
The authenticity of host '192.168.126.129 (192.168.126.129)' can't be established.
ED25519 key fingerprint is SHA256:NA1nlH5kMWy1QLUemjY/GCmOI/IWta410ArIuCoSDZ8
.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.126.129' (ED25519) to the list of known hosts.
kali@192.168.126.129's password:
Linux loghost 6.0.0-kali3-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.0.7-1kali1 (2
022-11-07) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Apr 13 00:05:29 2023 from 192.168.126.128
(kali㉿loghost)-[~]
$
```

Connexion ssh d'une machine temporaire vers une machine agent "hakan" [PC3] en passant par la machine LogHost [PC2]



```
(kali㉿loghost)-[~]
$ ssh kali@192.168.126.128
The authenticity of host '192.168.126.128 (192.168.126.128)' can't be established.
ED25519 key fingerprint is SHA256:NA1nlH5kMWy1QLUemjY/GCmOI/IWta410ArIuCoSDZ8
.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.126.128' (ED25519) to the list of known hosts.
kali@192.168.126.128's password:
Linux hakan 6.0.0-kali3-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.0.7-1kali1 (202
2-11-07) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Apr 28 15:28:43 2023 from 192.168.126.1
(kali㉿hakan)-[~]
$
```

Nous observons un log spécifique lorsqu'il y'a une connexion en cascade qui est le suivant:

```
(kali㉿loghost)-[~/var/log/hakan]
$ grep "deprecated" agent_logs.log
2023-04-28T15:28:43+02:00 hakan sshd[270013]: pam_env(sshd:session): deprecated reading of user environment enabled
2023-04-28T15:31:13+02:00 hakan sshd[270632]: pam_env(sshd:session): deprecated reading of user environment enabled
2023-04-28T15:36:08+02:00 hakan sshd[271850]: pam_env(sshd:session): deprecated reading of user environment enabled
```

Nous allons donc nous baser sur ce log pour la construction de la règle SEC.

Création d'une règle SEC pour détecter la connexion en cascade:

```
(kali㉿loghost)-[~/etc/sec]
$ cat connexion_cascade.conf
type=Single
ptype=RegExp
pattern="^(\d*\-\d*\-\d*).(\d*\:\d*\:\d*)\+\d*\:\d* (\w+) sshd\[.\]: pam_env(\(sshd:session\)): deprecated reading of user environment enabled
desc=Connexion en cascade sur machine agent hakan
action=write /var/log/hakan/connexion_cascade.log Une connexions en cascade utilisant ssh a été détectée !!
```

Lorsqu'un événement de connexion en cascade est repéré dans les journaux du service SSH sur la machine "agent hakan", cette règle enregistrera un message spécifique dans le fichier de journal "connexion_cascade.log" situé dans le répertoire /var/log/hakan. Le message indiquera qu'une connexion en cascade utilisant SSH a été détectée, fournissant ainsi une trace de l'incident pour des raisons de sécurité et de suivi ultérieur.

Exécution et résultat de la règle:

```
(kali㉿loghost)-[~/etc/sec]
$ sec -conf=/etc/sec/connexion_cascade.conf --input=/var/log/hakan/agent_logs.log
SEC (Simple Event Correlator) 2.9.1
Reading configuration from /etc/sec/connexion_cascade.conf
1 rules loaded from /etc/sec/connexion_cascade.conf
No --bufsize command line option or --bufsize=0, setting --bufsize to 1
Opening input file /var/log/hakan/agent_logs.log
Interactive process, SIGINT can't be used for changing the logging level
Writing event 'Une connexions en cascade utilisant ssh a été détectée !!' to file '/var/log/hakan/connexion_cascade.log'
```

Lors d'une simulation du scénario, nous obtenons bien un message sur la console nous informant d'une connexion en cascade.

```
(kali㉿loghost)-[~/etc/sec]
$ cat /var/log/hakan/connexion_cascade.log
Une connexions en cascade utilisant ssh a été détectée !!
```

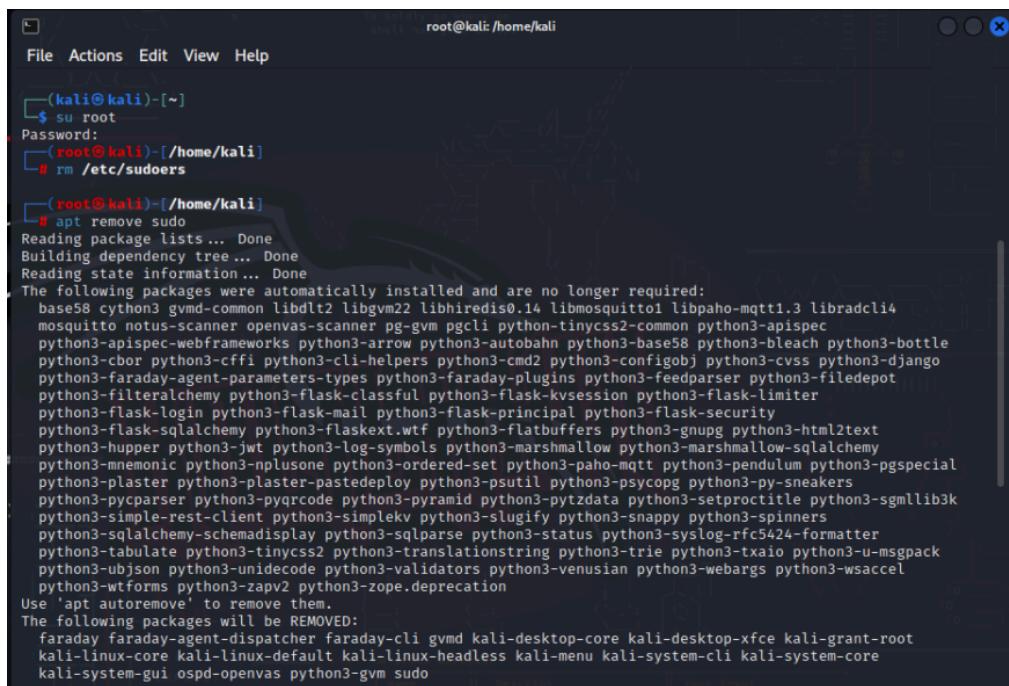
3.7 Faille Sudo

Installation sudo 1.8.23

Pour commencer ce scénario nous devons d'abord installer une version de sudo vulnérable à la faille . Nous avons décidé d'installer la version 1.8.23 de sudo . Pour se faire nous avons dû installer cette version depuis ce lien :

<https://www.sudo.ws/getting/download/#binary>.

Nous avons ensuite supprimer la version sudo ainsi que les fichiers de celui-ci de notre machine :



```
root@kali: /home/kali
File Actions Edit View Help
(kali㉿kali)-[~]
$ su root
Password:
(root㉿kali)-[~/home/kali]
# rm /etc/sudoers
(root㉿kali)-[~/home/kali]
# apt remove sudo
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
base58 cython3 gvmd-common libldlt2 libgvm2 libhiredis0.14 libmosquitto1 libpaho-mqtt1.3 libradcli4
mosquitto notus-scanner openvas-scanner pg-gvm pgcli python-tinycss2-common python3-apispec
python3-apispec-webframeworks python3-arrow python3-autobahn python3-base58 python3-bleach python3-bottle
python3-cbor python3-cffi python3-cli-helpers python3-cmd2 python3-configobj python3-cvss python3-django
python3-faraday-agent-parameters-types python3-faraday-plugins python3-feedparser python3-filedepot
python3-filteralchemy python3-flask-classful python3-flask-kvsession python3-flask-limiter
python3-flask-login python3-flask-mail python3-flask-principal python3-flask-security
python3-flask-sqlalchemy python3-flaskext.wtf python3-flatbuffers python3-gnupg python3-html2text
python3-hupper python3-jwt python3-log-symbols python3-marshmallow python3-marshmallow-sqlalchemy
python3-mnemonic python3-npluseone python3-ordered-set python3-paho-mqtt python3-pendulum python3-pgspecial
python3-plaster python3-plaster-pastedeploy python3-psutil python3-psycopg python3-py-sneakers
python3-pycparser python3-pyrcode python3-pyramid python3-pytzdata python3-setproctitle python3-sgmlib3k
python3-simple-rest-client python3-simplekv python3-slugify python3-snappy python3-spinners
python3 sqlalchemy-schemadisplay python3-sqlparse python3-status python3-syslog-rfc5424-formatter
python3-tabulate python3-tinycss2 python3-translationstring python3-trie python3-tx aio python3-u-msgpack
python3-ubjson python3-unicodecode python3Validators python3-venusian python3-webargs python3-wsaccel
python3-wtforms python3-zapv2 python3-zope.deprecation
Use 'apt autoremove' to remove them.
The following packages will be REMOVED:
faraday faraday-agent-dispatcher faraday-cli gvmd kali-desktop-core kali-desktop-xfce kali-grant-root
kali-linux-core kali-linux-default kali-linux-headless kali-menu kali-system-cli kali-system-core
kali-system-gui ospd-openvas python3-gvm sudo
```

Nous devons ensuite nous rendre dans l'emplacement du fichier sudo-1.8 afin de lancer l'installation de la version :

```

└──(root㉿kali)-[~/home/kali/Downloads/sudo-1.8.23]
  # ls
  ABOUT-NLS      Makefile.in  autogen.sh  configure.ac  init.d    m4          plugins
  ChangeLog     NEWS        config.guess  doc          install-sh  mkdep.pl   po
  INSTALL       README     config.h.in  examples     include     lib        mkinstalldirs  pp
  INSTALL.configure README.LDAP config.sub  indent.pro  log2cl.pl  mkpkg    pathnames.h.in  src
  MANIFEST      aclocal.m4  configure   ltmain.sh   sudo.pp
  # ./configure
  configure: Configuring Sudo version 1.8.23
  checking for gcc ... gcc
  checking whether the C compiler works ... yes
  checking for C compiler default output file name ... a.out
  checking for suffix of executables...
  checking whether we are cross compiling ... no
  # make
  for d in lib/util  plugins/group_file plugins/sudoers plugins/system_group src include doc examples; do \
    (cd $d && exec make all) && continue; \
    exit $?; \
done
make[1]: Entering directory '/home/kali/Downloads/sudo-1.8.23/lib/util'
/bin/bash ../../libtool --tag=disable-static --mode=compile gcc -c -o event.lo -I../../include -I.. -I..
-D_PATH_SUDO_CONF="/etc/sudo.conf" -D_FORTIFY_SOURCE=2 -g -O2 -fvisibility=hidden -fPIE -fstack-protect
or-strong ./event.c
make[1]: Leaving directory '/home/kali/Downloads/sudo-1.8.23/lib/util'
# make install
if test -d ./.hg && cd .; then \
  if hg log --style=changelog -b default > ChangeLog.tmp; then \
    mv -f ChangeLog.tmp ChangeLog; \
  else \
    rm -f ChangeLog.tmp; \
  fi; \
elif test -d ./git && cd .; then \
  ./log2cl.pl -b master > ChangeLog; \
else \
  echo "ChangeLog data not available" > ChangeLog; \
fi
for d in lib/util  plugins/group_file plugins/sudoers plugins/system_group src include doc examples; do \
  (cd $d && exec make pre-install) && continue; \
  exit $?; \
done
make[1]: Entering directory '/home/kali/Downloads/sudo-1.8.23/lib/util'
make[1]: Nothing to be done for 'pre-install'.
make[1]: Leaving directory '/home/kali/Downloads/sudo-1.8.23/lib/util'
make[1]: Entering directory '/home/kali/Downloads/sudo-1.8.23/plugins/group_file'
make[1]: Nothing to be done for 'pre-install'.

```

Une fois les commandes exécutées, nous pouvons apercevoir que nous avons dorénavant la version vulnérable:

```

└──(root㉿kali)-[~/home/kali/Downloads/sudo-1.8.23]
  # sudo -V
  Sudo version 1.8.23
  Configure options:
  Sudoers policy plugin version 1.8.23
  Sudoers file grammar version 46

```

Comment simuler ce scénario ?

Afin de tester la faille, j'ai créé un nouvel utilisateur nommé "testfaille".

On peut apercevoir que suite à une commande sudo, un message d'erreur est retourné informant à l'utilisateur qu'il n'est pas autorisé à exécuter ce genre de commande :

```
(kali㉿kali)-[~/etc]
$ su testfaille
Password:
(testfaille㉿kali)-[~/etc]
$ sudo su
Sorry, user testfaille is not allowed to execute '/usr/bin/su' as root on kali.
```

Nous allons donc utiliser la faille qui est de se connecter en sudo avec un id négatif :

```
(testfaille㉿kali)-[~/etc]
$ sudo -u#-1 -s
Password:
root@kali:/etc#
```

La commande est bien passée je peux maintenant naviguer sur la machine en tant que root.

Création d'une règle SEC pour détecter l'utilisation de la faille:

```
kali@kali: /etc/SEC
File Actions Edit View Help
GNU nano 7.2                                     faillesudo.conf *
type=Single
type=RegExp
pattern=(\d*-\d*-\d*)\w(\d*:\d*:\d*).*\d* (\w+) sudo: .* PWD=(.*); (USER=#-1); COMMAND=(.*)
desc= Utilisation faille SUDO
action=pipe '$1 : $2 User $4 a contourné les barrières de sécurité de la commande sudo sur $3' /usr/bin/mail -s 'Utilisation faille SUDO' kali
```

La règle va se baser sur une expression régulière qui a pour pattern le message de log généré lorsqu'une utilisation a été réalisée. On peut apercevoir le (USER=#-1) qui est l'utilisation d'un id négatif pour passer en mode root.

Comme action nous avons l'envoie d'un mail vers notre machine loghost "kali" nous informant qu'une utilisation de la faille a été détectée.

Configuration du serveur mail :

Nous avons dans un premier temps installé "Mailutils" qui est un ensemble d'utilitaires permettant la gestion des mails.

```
(kali㉿kali)-[~]
$ sudo apt install mailutils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
mailutils is already the newest version (1:3.15-4).
0 upgraded, 0 newly installed, 0 to remove and 16 not upgraded.
```

Nous allons ensuite ajouter un utilisateur au groupe “mail”, dans notre cas ce sera “kali” qui est notre LogHost.

```
(kali㉿kali)-[~]
$ sudo adduser kali mail
Adding user `kali' to group `mail' ...
Done.
```

Enfin, il nous reste à octroyer les droits sur le fichier mail de l'utilisateur

```
(kali㉿kali)-[~]
$ sudo chmod ug+rwx /var/mail/kali
```

Exécution et résultat de la règle:

Nous pouvons apercevoir ici que la règle a bien été chargée et qu'elle est prête à l'exécution.

```
(kali㉿kali)-[~/etc/SEC]
$ sudo sec -conf=/etc/SEC/faillesudo.conf --input=/var/log/agent_logs.log
SEC (Simple Event Correlator) 2.9.1
Reading configuration from /etc/SEC/faillesudo.conf
1 rules loaded from /etc/SEC/faillesudo.conf
No --bufsize command line option or --bufsize=0, setting --bufsize to 1
Opening input file /var/log/agent_logs.log
Interactive process, SIGINT can't be used for changing the logging level
Feeding event '2023-05-06 : 10:48:42 User testfaille a contourné les barrières de sécurité de la commande sudo sur kali' to shell command '/usr/bin/mail -s 'Utilisation faille SUDO' kali'
Child 14129 created for command '/usr/bin/mail -s 'Utilisation faille SUDO' kali'
Feeding event '2023-05-06 : 10:48:42 User testfaille a contourné les barrières de sécurité de la commande sudo sur kali' to shell command '/usr/bin/mail -s 'Utilisation faille SUDO' kali'
Child 14131 created for command '/usr/bin/mail -s 'Utilisation faille SUDO' kali'
```

Lors d'une simulation du scénario, nous recevons bien un mail sur notre loghost :

```
(kali㉿kali)-[~/etc/SEC]
$ mail
"/var/mail/kali": 8 messages 8 new
>N 1 Mail Delivery Syst Tue May  2 13:34  59/1682  Mail delivery failed: returning message to sender
N 2 Mail Delivery Syst Tue May  2 13:36  58/1655  Mail delivery failed: returning message to sender
N 3 Mail Delivery Syst Wed May  3 11:04  57/1646  Mail delivery failed: returning message to sender
N 4 kali㉿kali      Sat May  6 10:37  17/418   TEST2
N 5 root㉿kali     Sat May  6 10:44  17/484   Sudo exploit
N 6 root㉿kali     Sat May  6 10:44  17/484   Sudo exploit
N 7 root㉿kali     Sat May  6 10:48  16/531   Utilisation faille SUDO
N 8 root㉿kali     Sat May  6 10:48  16/531   Utilisation faille SUDO
```

Nous pouvons ensuite voir d'avantages d'informations sur le mail reçu :

```
Return-path: <root@kali>
Envelope-to: kali@kali
Delivery-date: Sat, 06 May 2023 10:48:45 -0400
Received: from root by kali with local (Exim 4.96)
  (envelope-from <root@kali>)
  id 1pvJDG-0003g0-1T
  for kali@kali;
  Sat, 06 May 2023 10:48:42 -0400
Subject: Utilisation faille SUDO
To: kali@kali
User-Agent: mail (GNU Mailutils 3.15)
Date: Sat, 6 May 2023 10:48:42 -0400
Message-Id: <E1pvJDG-0003g0-1T@kali>
From: root@kali

2023-05-06 : 10:48:42 User testfaille a contourné les barrières de sécurité de la commande sudo sur kali
```

En quoi la règle évite-elle les faux-positifs?

Utilisation d'un pattern identifiant uniquement les log contenant un id négatif qui fait référence à la faille SUDO.

Laboratoire 4 : SNMP

Introduction

Le protocole SNMP (Simple Network Management Protocol) est un standard Internet pour la gestion des périphériques sur les réseaux IP. Ce protocole facilite l'échange d'informations de gestion entre les dispositifs de réseau. Dans ce rapport, nous allons décrire l'utilisation des différents Object Identifier (OID) utilisés pour le monitoring d'un serveur. Ces OIDs sont extraits à l'aide de la commande `snmpwalk` et un script qui est une méthode pour récupérer les informations de gestion via SNMP.

Pour notre analyse, nous avons utilisé la commande `snmpwalk -v2c -c HELMpAllUser9465CmA 192.168.128.24`, avec des paramètres supplémentaires `-OUqv` pour faciliter le traitement numérique des données récupérées et leur utilisation ultérieure avec Grafana, un outil puissant pour la visualisation et l'analyse de données.

OID surveillés :

1. Utilisation du processeur

Les OIDs suivants sont utilisés pour surveiller l'utilisation du processeur. L'utilisation du CPU est l'un des indicateurs les plus importants de la performance d'un système. Ils indiquent la quantité de temps de processeur utilisée par rapport à la capacité totale du processeur pendant cette période. Surveiller cette valeur permet de détecter les pics d'utilisation du processeur et d'identifier les périodes où le processeur est fortement sollicité.

CPU	CPU_1MIN	CPU_5MIN	CPU_15MIN
Description	Charge du processeur sur 1 minute	Charge du processeur sur 5 minutes	Charge du processeur sur 15 minutes
OID	.1.3.6.1.4.1.2021.1 0.1.3.1	.1.3.6.1.4.1.2021.1 0.1.3.2	.1.3.6.1.4.1.2021.1 0.1.3.3

L'OID suivant indique la proportion d'utilisation du CPU par les processus utilisateurs par rapport à l'utilisation totale du CPU. Il mesure la charge de travail générée par les processus exécutés par les utilisateurs. Surveiller cette valeur permet de comprendre la contribution des processus utilisateurs à la charge du CPU et d'identifier les processus qui utilisent une part importante des ressources du CPU.

CPU_USER_PCT	
OID	.1.3.6.1.4.1.2021.11.9.0
Description	Utilisation du CPU par les processus utilisateurs
Unité	Pourcentage (%)

L'OID suivant fournit la proportion d'utilisation du CPU par le système d'exploitation lui-même. Cette valeur peut être utilisée pour des analyses plus avancées ou pour des besoins spécifiques liés à l'utilisation du CPU par le système.

CPU_SYSTEM_RAW	
OID	.1.3.6.1.4.1.2021.11.52.0
Description	Utilisation brute du CPU par le système
Unité	Pourcentage (%)

Le dernier OID représente le taux d'inactivité du CPU, c'est-à-dire la proportion du temps pendant lequel le CPU est inactif par rapport au temps total. Il indique la disponibilité du CPU pour traiter de nouvelles tâches. Surveiller cette valeur permet de vérifier si le CPU a suffisamment de capacité pour gérer les demandes supplémentaires.

CPU_IDLE_PCT	
OID	.1.3.6.1.4.1.2021.11.11.0
Description	Taux d'inactivité du CPU
Unité	Pourcentage (%)

2. Mémoire

TOT_RAM_USED	
OID	.1.3.6.1.4.1.2021.4.6.0
Description	Mémoire RAM utilisée
Unité	Kilooctets (Ko)

Cet OID représente la quantité de mémoire RAM actuellement utilisée sur la machine. Il indique la quantité de mémoire occupée par les processus en cours d'exécution et les données stockées.

TOT_RAM_LIBRE	
OID	.1.3.6.1.4.1.2021.4.11.0
Description	Mémoire RAM libre
Unité	Kilooctets (Ko)

Cet OID indique la quantité de mémoire RAM disponible et non utilisée sur la machine. Il représente la mémoire libre qui peut être utilisée pour de nouvelles opérations.

TOT_RAM_SHARED	
OID	.1.3.6.1.4.1.2021.4.13.0
Description	Mémoire RAM partagée
Unité	Kilooctets (Ko)

Cet OID représente la quantité de mémoire RAM utilisée pour le partage entre plusieurs processus. La mémoire partagée est utilisée lorsque plusieurs processus ont besoin d'accéder aux mêmes données en mémoire.

TOT_MEM_CACHE	
OID	.1.3.6.1.4.1.2021.4.15.0
Description	Mémoire cache totale
Unité	Kilooctets (Ko)

Cet OID représente la taille totale de la mémoire cache sur la machine. La mémoire cache est une mémoire plus rapide que la mémoire principale (RAM) et est utilisée pour stocker les données récemment utilisées afin d'améliorer les performances d'accès aux données. Cette valeur indique la capacité totale de la mémoire cache disponible sur la machine.

3. Espace disque

Les OID suivants sont utilisés pour surveiller l'espace disque. L'espace disque disponible est crucial pour le bon fonctionnement d'un système, en particulier pour les serveurs gérant de grandes quantités de données.

TAILLE_TOT_DISK	
OID	1.3.6.1.2.1.25.2.3.1.5
Description	Taille totale du disque
Unité	Octets (bytes)

Cet OID fait partie de la MIB-II et représente la taille totale de l'espace disque sur la machine. Il indique la capacité totale du disque, c'est-à-dire la quantité maximale de données pouvant être stockées sur le disque. Surveiller cette valeur permet de connaître la capacité de stockage disponible sur le disque.

USED_SPACE_DISK	
OID	1.3.6.1.2.1.25.2.3.1.6
Description	Espace disque utilisé
Unité	Octets (bytes)

Cet OID également présent dans la MIB-II indique l'espace disque actuellement utilisé sur la machine. Il fournit la quantité de données stockées sur le disque. Surveiller cette valeur permet de connaître l'utilisation effective de l'espace disque et d'évaluer l'espace restant disponible.

4. Réseau

Les OIDs suivants sont utilisés pour surveiller l'activité réseau.

LIST_INTERFACES	
OID	.1.3.6.1.2.1.2.2.1.2
Description	Liste des interfaces réseau

Cet OID fait partie de la MIB-II et retourne une liste des noms des interfaces réseau présentes sur l'hôte. Les valeurs renvoyées sont des chaînes de caractères qui identifient de manière unique chaque interface réseau. Surveiller cette valeur permet d'obtenir la liste des interfaces réseau disponibles sur le système.

OCTETS_IN	
OID	.1.3.6.1.2.1.2.2.1.10
Description	Nombre total d'octets entrants
Unité	Octets (bytes)

Ce OID représente le nombre total d'octets entrants sur une interface réseau spécifique. Il indique la quantité totale de données qui ont été reçues par l'interface depuis son dernier démarrage ou depuis le dernier comptage de ces statistiques. Surveiller cette valeur permet de mesurer le trafic entrant sur une interface spécifique, ce qui est utile pour évaluer la charge réseau et détecter les problèmes de congestion ou de saturation.

OCTETS_OUT	
OID	.1.3.6.1.2.1.2.2.1.16
Description	Nombre total d'octets sortants
Unité	Octets (bytes)

Ce OID représente le nombre total d'octets sortants sur une interface réseau spécifique. Il indique la quantité totale de données qui ont été envoyées par l'interface depuis son dernier démarrage ou depuis le dernier comptage de ces statistiques. Surveiller cette valeur permet de mesurer le trafic sortant sur une interface spécifique, ce qui est utile pour évaluer la charge réseau et détecter les problèmes de congestion ou de saturation.

6. Processus

Processus	HR_PROCESS_NAME	HR_PROCESS_CPU	HR_PROCESS_MEM
Description	Donne le nom de chaque processus.	Donne l'utilisation du CPU par chaque processus.	Donne l'utilisation de la mémoire par chaque processus.
OID	.1.3.6.1.2.1.25.4.2.1.2	.1.3.6.1.2.1.25.5.1.1 .1	.1.3.6.1.2.1.25.5.1.1 .2

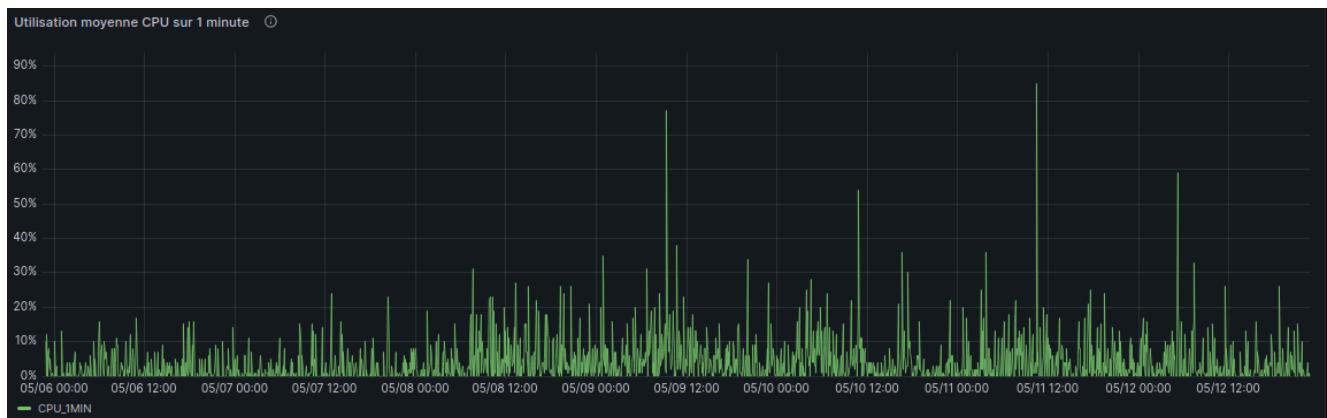
Les OIDs cités sont utilisés pour surveiller les processus en cours d'exécution sur l'hôte. Le monitoring des processus peut aider à identifier les processus qui consomment une quantité excessive de ressources.

Analyse des données obtenues

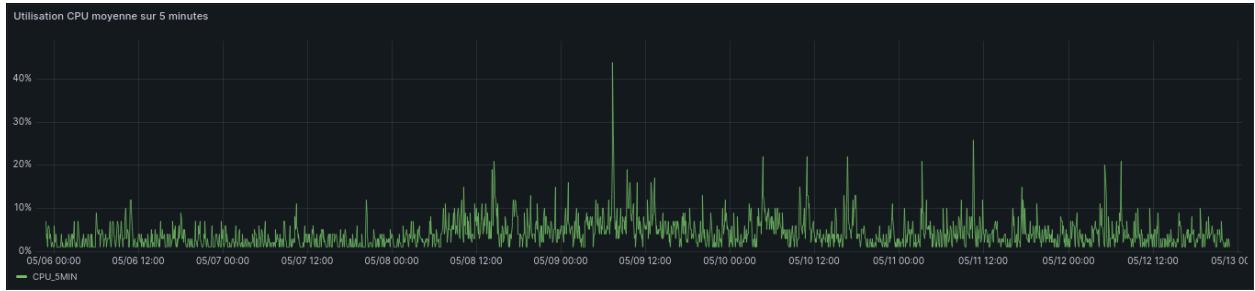
Nous avons laissé notre script collecter des informations sur nos différents composants pendant une semaine en utilisant un script (voir en annexe). Ensuite, nous avons obtenu des données brutes. Pour les exploiter, nous avons transféré les fichiers CSV sur Grafana afin de créer des graphiques.

L'analyse des graphiques de surveillance est une partie essentielle de la gestion des systèmes. Elle permet de comprendre visuellement les tendances, les pics et les anomalies dans le comportement d'un système. Grâce à ces graphiques, nous pouvons observer et interpréter les données collectées pour prendre des décisions éclairées et optimiser les performances du système.

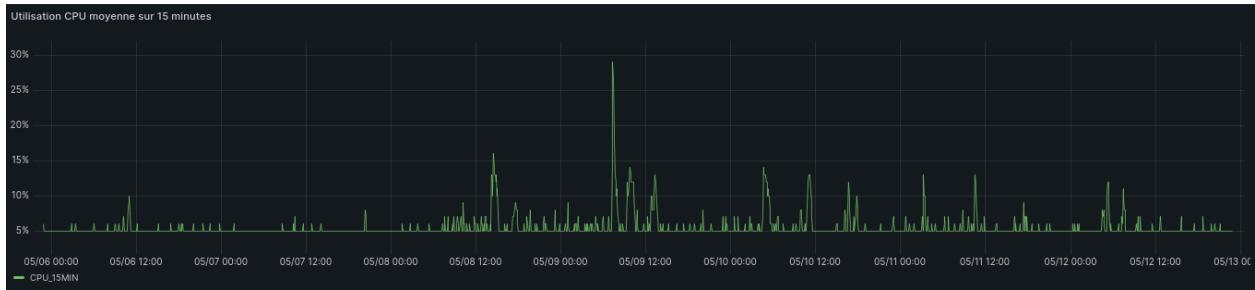
- CPU



Comme nous pouvons l'observer, l'utilisation du CPU est généralement faible, sauf lors de pics d'activité. En particulier, le 09/05, nous avons constaté une utilisation atteignant près de 80%, et le 11/05, elle a atteint près de 85%. Pour le reste de la semaine, l'utilisation du CPU est relativement stable, avec des variations entre 5% et 30%. Le taux d'utilisation minimum a été enregistré à presque 0% le 06/05 à 23h02, tandis que le taux d'utilisation maximum a atteint 85% le 11/05 à 14h30.



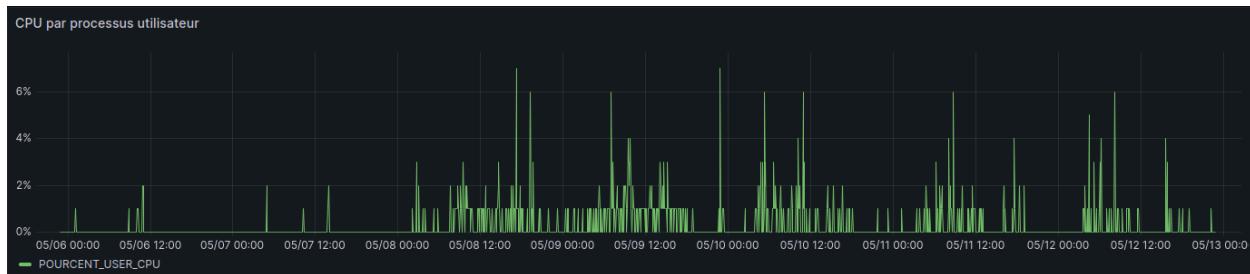
Avec ce deuxième graphique, nous observons une moyenne d'utilisation sur 5 minutes. Lorsque nous le comparons au premier graphique qui présente une moyenne sur une minute, nous pouvons constater que le pic maximum qui s'est produit le 11/05 n'a pas duré à long terme, car il n'est plus remarquable sur la période de 5 minutes. En revanche, nous pouvons toujours observer le premier pic du 09/05 et conclure que les tâches exécutées à ce moment-là ont pris plus de temps à se réaliser. Ainsi, nous pouvons constater que le pic du 09/05 a sollicité davantage le CPU à long terme.



Sur notre graphique de moyenne sur 15 minutes, nous pouvons clairement constater que le CPU a été beaucoup plus sollicité le 09/05 par rapport aux autres jours de la semaine. De plus, nous pouvons observer que l'utilisation du CPU est généralement stable (sauf le 09/05), avec de petits pics atteignant environ 10%.



Ce graphique représente la partie inactive du CPU, c'est-à-dire la proportion de temps pendant laquelle le CPU n'est pas sollicité. Il est important de noter que ces pourcentages ne sont pas calculés en moyenne sur une période de temps déterminée, mais sont instantanés au moment où le script est exécuté. Nous pouvons observer que le CPU fonctionne en moyenne entre 1% et 5% d'inactivité, sauf en cas de pics d'activité tels que celui qui s'est produit la nuit du 05/12.



Ce graphique représente le pourcentage d'utilisation du CPU par les processus utilisateur, ce qui correspond à la part des ressources CPU utilisée pour exécuter des tâches initiées par les utilisateurs du système, par opposition aux tâches du système elles-mêmes. Ces données sont pertinentes lors de l'analyse de l'utilisation du CPU par chaque utilisateur afin de comparer les données et d'évaluer leur contribution respective à la charge du CPU.



Ce dernier graphique représente la quantité de capacité de traitement du CPU utilisée pour gérer les tâches du système d'exploitation lui-même, par opposition au graphique précédent qui observait les tâches initiées par les utilisateurs. Il met en évidence la part des ressources du CPU consacrée aux processus et aux opérations internes du système d'exploitation, ce qui peut être utile pour analyser l'efficacité et la charge du système lui-même.

Concernant l'utilisation du CPU, il est normal de constater des pics d'activité à certains moments. Ces pics peuvent être causés par plusieurs facteurs. Voici quelques raisons possibles expliquant pourquoi le CPU d'un serveur peut connaître des pics d'utilisation à des moments spécifiques :

Tâches planifiées : Certaines tâches sont programmées pour s'exécuter à des moments spécifiques, ce qui peut entraîner une augmentation temporaire de l'utilisation du CPU pendant leur exécution.

Activité des utilisateurs : Si le serveur est utilisé pour héberger un site web ou une application, il peut y avoir des pics d'utilisation du CPU lorsque le nombre d'utilisateurs ou l'intensité de l'activité augmente.

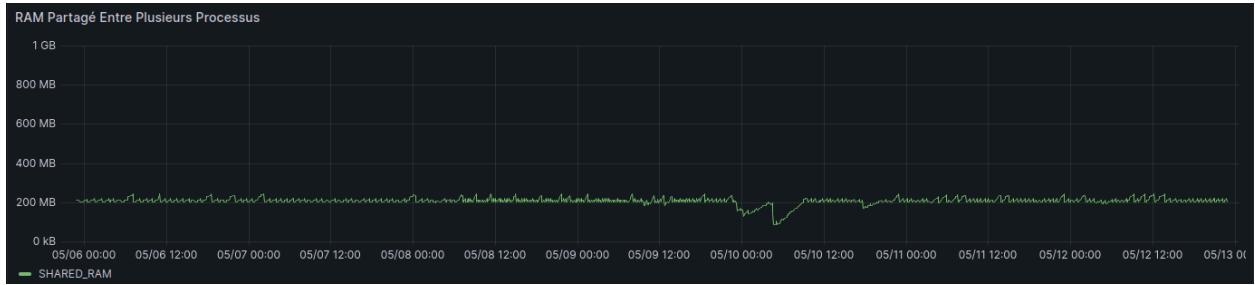
Processus en arrière-plan : Certains processus en arrière-plan s'exécutent régulièrement à des intervalles définis, tels que des tâches de maintenance, des sauvegardes ou des mises à jour. Ces processus peuvent utiliser une quantité significative de ressources CPU pendant leur exécution, ce qui peut se traduire par des pics d'utilisation.

Il est important de surveiller ces pics d'utilisation du CPU afin de comprendre leur origine et d'optimiser les performances du système, en veillant à ce que le CPU dispose de suffisamment de capacité pour répondre aux besoins des utilisateurs et des processus essentiels.

- RAM



Sur ce graphique, nous pouvons observer l'utilisation de la RAM en GB sur une période d'une semaine. Nous constatons que la RAM reste relativement stable, aux alentours de 500 MB, sauf entre le 05/08 et le 05/11. Ces pics d'utilisation plus élevée de la RAM sont considérés comme normaux, car la RAM stocke des données temporaires et peut fluctuer en fonction des besoins du système..



Ce graphique représente le partage de la RAM entre les processus. Comme on peut l'observer, si la RAM partagée à une moyenne de 200 Mo, cela signifie que cette quantité de mémoire est généralement utilisée par plusieurs processus pour partager leurs données. Il s'agit d'une utilisation normale et qui ne pose généralement pas de problème.

Cependant, nous avons constaté un pic négatif pendant la nuit du 10/05 entre minuit et 3 heures. Un pic négatif signifie que la quantité de RAM partagée a diminué pendant cette période. Plusieurs raisons peuvent expliquer cela, comme une diminution de l'activité sur le serveur, des redémarrages ou une maintenance du serveur, etc. Ces variations ponctuelles peuvent être liées à des événements spécifiques, mais tant qu'elles ne persistent pas de manière continue, elles ne sont généralement pas préoccupantes.

- Disques



Avant d'analyser ce graphique, il est important de préciser que nous avons collecté des données à partir de plusieurs disques (fichiers disponibles en annexe), mais le graphique présenté ici concerne le disque 1. Les valeurs affichées sont exprimées en octets (bytes).

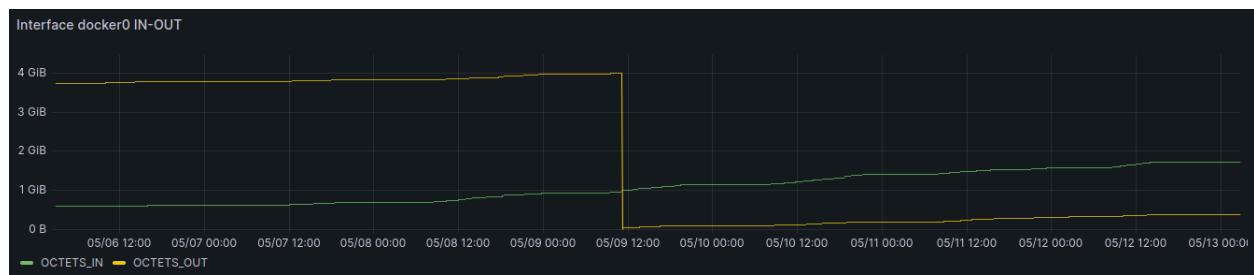
En examinant le graphique, nous pouvons observer que lorsque le disque est fortement sollicité, il est nécessaire de prendre des mesures appropriées, telles que décharger des fichiers ou ajouter de l'espace de stockage.

Cela contribue à garantir le bon fonctionnement du système et à éviter les problèmes liés à un espace disque insuffisant. En surveillant régulièrement l'utilisation du disque et en prenant des mesures proactives lorsque nécessaire, nous pouvons maintenir une bonne performance du système et éviter les éventuelles contraintes liées au stockage.

- Interfaces

Les OIDs suivants sont utilisés pour surveiller l'activité réseau. Un monitoring efficace du trafic réseau aide à identifier les éventuels goulets d'étranglement et à garantir une performance optimale du réseau.

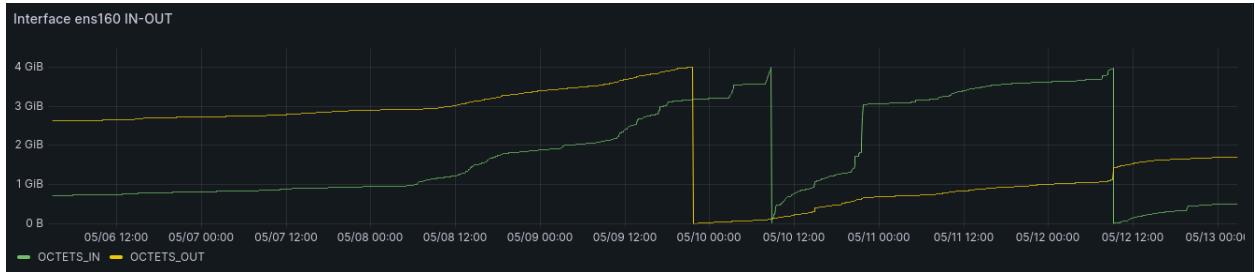
Nous avons analysé graphiquement les trois interfaces les plus utilisées. Les autres interfaces sont répertoriées dans un fichier joint à cet effet.



L'interface docker0 est une interface de réseau virtuel utilisée par Docker, qui est un logiciel de virtualisation de niveau système d'exploitation. Les conteneurs Docker utilisent cette interface pour communiquer avec l'hôte et entre eux.

Pour l'interface docker0, nous avons observé une diminution significative du trafic sortant (OCTETS_OUT) le 9 à midi, suivie d'une reprise progressive jusqu'au 13. Cela pourrait indiquer que les conteneurs Docker actifs sur votre serveur étaient moins actifs ou ont été arrêtés à ce moment-là, ce qui a entraîné une réduction de la quantité de trafic sortant.

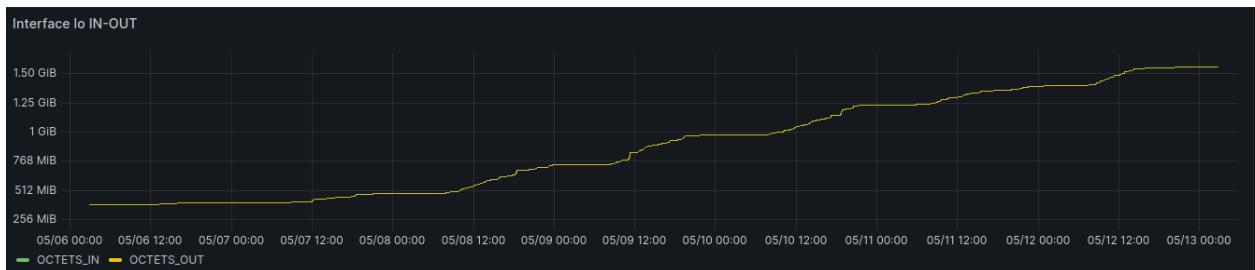
En ce qui concerne l'augmentation constante du trafic entrant (OCTETS_IN), cela pourrait indiquer que votre serveur recevait de plus en plus de données. Cela peut être dû à une activité accrue, telle qu'une augmentation du nombre de requêtes reçues par un serveur web, une augmentation du volume de téléchargement de données ou une augmentation de la communication entre les conteneurs.



L'augmentation progressive de la quantité de trafic sortant (OCTETS_OUT) que nous observons peut indiquer une activité croissante sur le serveur, telle qu'une utilisation accrue des services réseau ou un transfert accru de données.

La chute à zéro le 10 à minuit pourrait être due à plusieurs raisons, telles qu'une opération de maintenance planifiée ou une mise à jour nécessitant l'arrêt du réseau.

En ce qui concerne le trafic entrant (OCTETS_IN), nous observons deux chutes vers midi le 10 et le 12. Ces chutes pourraient être liées à des opérations de maintenance similaires ou à des problèmes de réseau. Il est intéressant de noter que ces chutes semblent se produire lorsque le trafic entrant ou sortant atteint 4 GiB. Cela pourrait suggérer un problème potentiel avec le serveur ou le réseau, comme un débordement de tampon ou une limite de capacité.



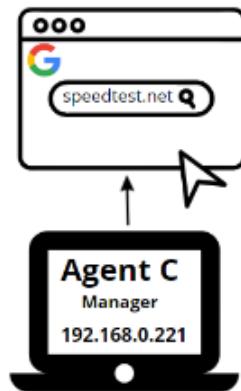
L'interface lo est l'interface de bouclage (ou loopback) du serveur SALTO. Elle est principalement utilisée pour le trafic réseau interne à la machine hôte. L'interface loopback a une adresse IP typique de 127.0.0.1 et est généralement désignée sous le nom de "localhost".

La nature de l'interface loopback signifie que tout ce qui est envoyé à cette interface est immédiatement reçu. C'est pourquoi nous observons que les valeurs OCTETS_IN et OCTETS_OUT sont identiques : chaque bit de données "envoyé" est également "reçu".

Une augmentation progressive de l'utilisation de l'interface de bouclage pourrait indiquer une activité accrue des services qui communiquent via cette interface. Par exemple, cela pourrait être dû à une application ou un service sur le serveur qui utilise de plus en plus les communications réseau internes.

Laboratoire 5 : Netflow

Scénario n°1 : L'agent C réalise des "speed test" via google



Identification des flux correspondant à ce scénario :

Pour ce faire, nous avons utilisé la commande

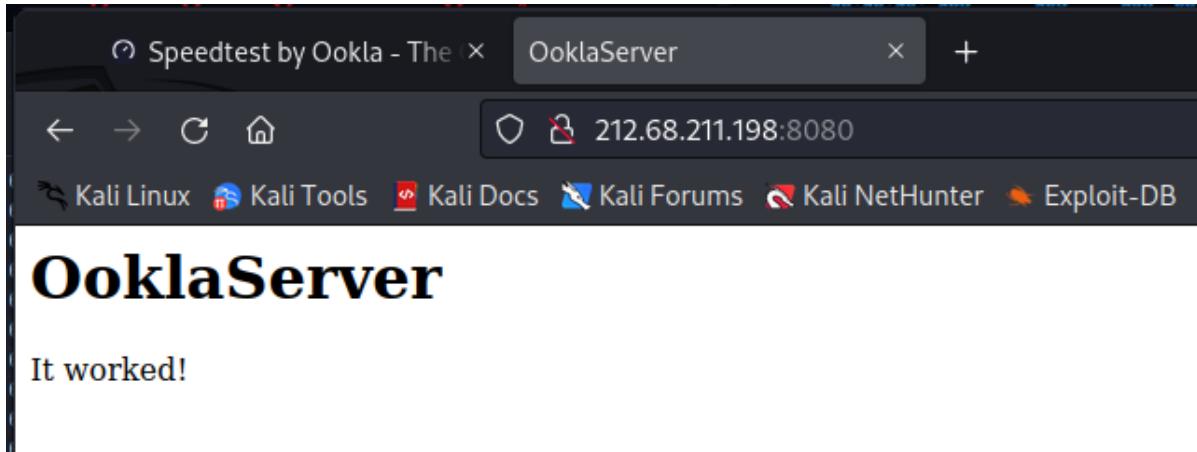
```
[kali㉿kali)-[~]
$ nfdump -R /var/cache/nfdump -A srcIp,srcPort,dstIp,dstPort,proto -O tstart
```

Voici les flux que nous avons obtenus :

2023-05-18	16:32:39.534	00:00:00.155	2600:90..c0:93a1	443	2a02:27..921:8cb	50584	TCP	11	7069	364851	642	1
2023-05-18	16:32:39.534	00:00:00.155	2a02:27..921:8cb	50584	2600:90..c0:93a1	443	TCP	11	1582	81651	143	1
2023-05-18	16:32:39.577	00:00:00.168	2600:90..c0:93a1	443	2a02:27..921:8cb	50594	TCP	11	7057	336047	641	1
2023-05-18	16:32:39.577	00:00:00.168	2a02:27..921:8cb	50594	2600:90..c0:93a1	443	TCP	12	1652	78666	137	1
2023-05-18	16:32:40.042	00:00:00.166	104.18.10.47	443	192.168.0.221	39454	TCP	6	3736	180048	622	1
2023-05-18	16:32:40.042	00:00:00.166	192.168.0.221	39454	104.18.10.47	443	TCP	7	1154	55614	164	1
2023-05-18	16:32:40.787	00:00:15.615	2a02:27..921:8cb	41224	2a02:a0..:2ff::2	8080	TCP	1575	119964	61460	76	1
2023-05-18	16:32:40.787	00:00:15.615	2a02:a0..:2ff::2	8080	2a02:27..921:8cb	41224	TCP	1882	36.6 M	18.8 M	19447	1
2023-05-18	16:32:40.829	00:00:15.574	2a02:27..921:8cb	57186	2a00:15..0:15::2	8080	TCP	2285	182488	93739	79	1
2023-05-18	16:32:40.829	00:00:15.574	2a00:15..0:15::2	8080	2a02:27..921:8cb	57186	TCP	5126	17.3 M	8.9 M	3382	1
2023-05-18	16:32:40.846	00:00:15.561	192.168.0.221	38734	212.68.211.198	8080	TCP	623	35098	18044	56	1
2023-05-18	16:32:40.846	00:00:15.561	212.68.211.198	8080	192.168.0.221	38734	TCP	692	8.2 M	4.2 M	11885	1
2023-05-18	16:32:40.927	00:00:15.480	2a02:b5..219::19	8080	2a02:27..921:8cb	57540	TCP	2703	8.3 M	4.3 M	3056	1
2023-05-18	16:32:40.927	00:00:15.480	2a02:27..921:8cb	57540	2a02:b5..219::19	8080	TCP	1263	95760	49488	75	1
2023-05-18	16:32:41.076	00:00:15.349	2a02:27..921:8cb	41240	2a02:a0..:2ff::2	8080	TCP	234	26790	13963	114	1
2023-05-18	16:32:41.076	00:00:15.349	2a02:a0..:2ff::2	8080	2a02:27..921:8cb	41240	TCP	228	26181	13645	114	1
2023-05-18	16:32:42.038	00:00:00.000	192.168.0.145	54876	99.181.79.12	443	TCP	2	1735	0	867	1
2023-05-18	16:32:46.232	00:00:04.061	192.168.0.145	54877	99.181.79.12	443	TCP	4	3470	6835	867	1
2023-05-18	16:32:47.314	00:00:09.093	2a02:27..921:8cb	41252	2a02:a0..:2ff::2	8080	TCP	867	65880	57961	75	1
2023-05-18	16:32:47.314	00:00:09.093	2a02:a0..:2ff::2	8080	2a02:27..921:8cb	41252	TCP	1006	14.7 M	12.9 M	14619	1
2023-05-18	16:32:50.563	00:00:03.890	192.168.0.145	54878	99.181.79.12	443	TCP	4	3470	7136	867	1
2023-05-18	16:32:57.588	00:00:16.821	2a02:27..921:8cb	40278	2a02:a0..:2ff::2	8080	TCP	2166	4.9 M	2.3 M	2275	1
2023-05-18	16:32:57.588	00:00:16.821	2a02:a0..:2ff::2	8080	2a02:27..921:8cb	40278	TCP	1791	136619	65070	76	1
2023-05-18	16:32:57.591	00:00:16.921	2a02:a0..:2ff::2	8080	2a02:27..921:8cb	40292	TCP	2109	160001	75646	75	1
2023-05-18	16:32:57.591	00:00:16.921	2a02:27..921:8cb	40292	2a02:a0..:2ff::2	8080	TCP	2544	5.8 M	2.8 M	2292	1
2023-05-18	16:32:57.592	00:00:16.251	2a02:a0..:2ff::2	8080	2a02:27..921:8cb	40294	TCP	312	42224	20785	135	1
2023-05-18	16:32:57.592	00:00:16.251	2a02:27..921:8cb	40294	2a02:a0..:2ff::2	8080	TCP	312	23928	11779	76	1
2023-05-18	16:32:57.594	00:00:17.044	2a02:a0..:2ff::2	8080	2a02:27..921:8cb	40304	TCP	1816	140082	65750	77	1
2023-05-18	16:32:57.594	00:00:17.044	2a02:27..921:8cb	40304	2a02:a0..:2ff::2	8080	TCP	2188	5.0 M	2.3 M	2270	1
2023-05-18	16:32:57.596	00:00:16.675	2a02:27..921:8cb	40314	2a02:a0..:2ff::2	8080	TCP	2347	5.4 M	2.6 M	2295	1
2023-05-18	16:32:57.596	00:00:16.675	2a02:a0..:2ff::2	8080	2a02:27..921:8cb	40314	TCP	1954	149702	71821	76	1
2023-05-18	16:32:57.714	00:00:16.253	2a02:27..921:8cb	40320	2a02:a0..:2ff::2	8080	TCP	357	40096	19735	112	1
2023-05-18	16:32:57.714	00:00:16.253	2a02:a0..:2ff::2	8080	2a02:27..921:8cb	40320	TCP	354	40972	20167	115	1

Comment être sûr que ce sont bien les flux du scénario ?

Nous avons utilisé un navigateur pour accéder à l'adresse IP obtenue en spécifiant le port 8080. Voici la réponse obtenue de cette adresse IP :



Nous tombons bien sur le serveur du speedtest.

Utilisation des protocoles pendant le scénario:

Protocoles	UDP	TCP
Bytes	926898	216246715
Pourcentages	0,4 %	99,6 %

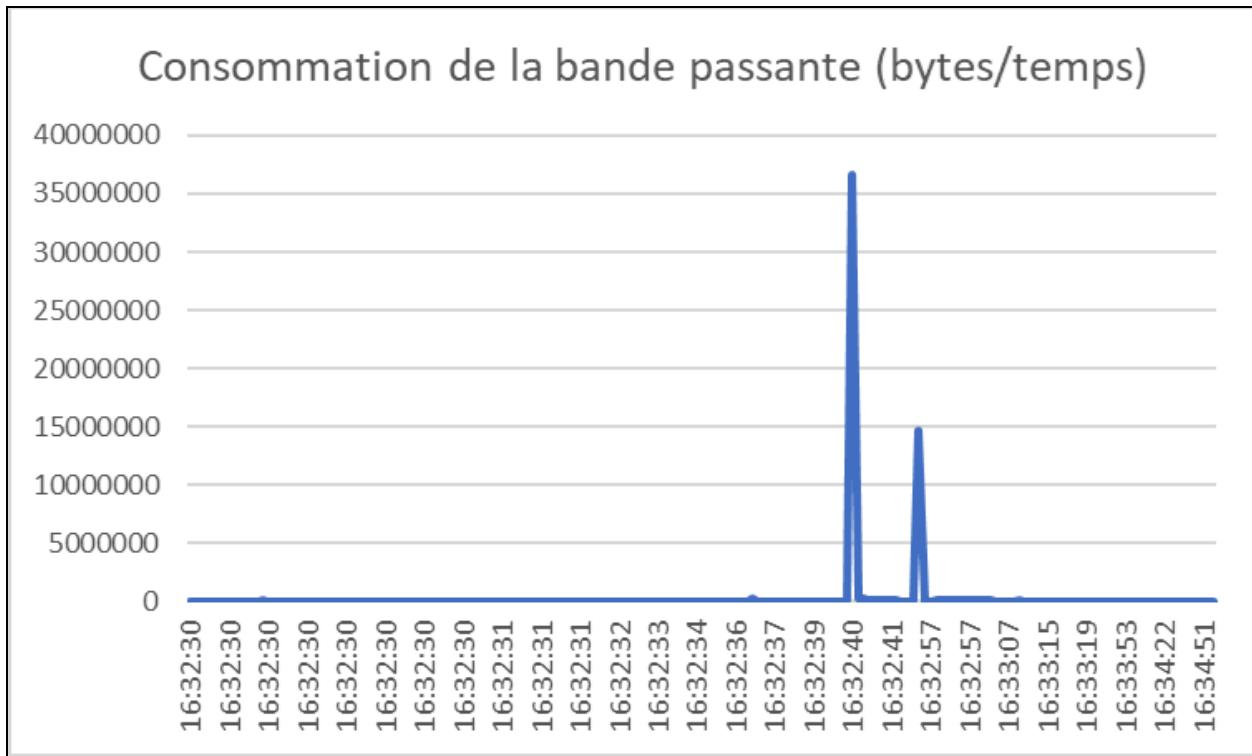
Les résultats obtenus révèlent que la grande majorité des flux (99,6%) sont basés sur le protocole TCP, tandis qu'une petite proportion (0,4%) utilise le protocole UDP.

Cette observation s'explique par la nature même du processus de speedtest, qui nécessite l'exécution de nombreuses requêtes TCP pour effectuer les différents tests tels que le téléchargement, le téléversement, etc. Ces tests exigent une transmission fiable des données, ce qui explique le pourcentage élevé de requêtes TCP dans les résultats. En d'autres termes, la prédominance des requêtes TCP découle des caractéristiques spécifiques des tests réalisés, qui requièrent une transmission de données stable et cohérente.

C'est pourquoi, dans ce scénario, les requêtes UDP sont très limitées, voire inexistantes. Les tests de speedtest mettent davantage l'accent sur la fiabilité des données, ce qui

rend le protocole TCP plus adapté à cet objectif, tandis que le protocole UDP, étant plus léger et moins fiable, est moins fréquemment utilisé dans ce contexte.

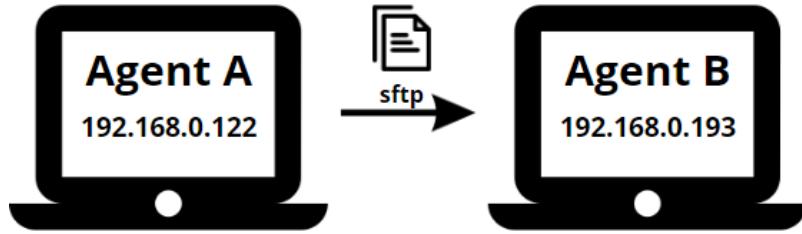
Graphique représentant le flux du scénario:



Analyse du graphique:

Dans ce graphique, nous pouvons apercevoir 2 pics importants qui correspondent aux 2 parties du speedtest, l'un pour le calcul de l'upload et l'autre du download.

Scénario n°2 : L'agent A télécharge via sftp les fichiers volumineux depuis l'agent B



Pour ce scénario, j'ai téléchargé au préalable des fichiers iso (19.3 GB) sur la machine B afin de la transférer sur la machine B.

Transfert de fichiers :

Connexion via sftp sur l'agent A :

```
(kali㉿agentA) [~/pmacct]
$ sftp kali@192.168.0.193

The authenticity of host '192.168.0.193 (192.168.0.193)' can't be established.
ED25519 key fingerprint is SHA256:4MTnMuMqUbILSF5vcp3sQmx0h3xUPzW+Znce4KZe58.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.0.193' (ED25519) to the list of known hosts.
kali@192.168.0.193's password:
Connected to 192.168.0.193.
sftp> cd
sftp> ls
Desktop    Documents    Downloads    Music        Pictures    Public      Templates    Videos     pmacct     src
```

Transfert du dossier contenant des iso :

Identification des flux correspondant à ce scénario :

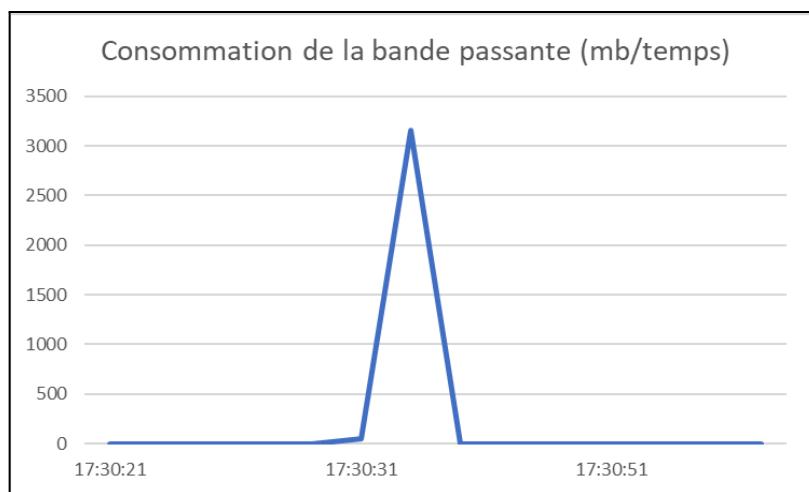
2023-05-16 17:30:21.008	4d	01:13:31.428	192.168.0.145	63095	224.0.0.252	5355 UDP	6	366	0	61	3
2023-05-16 17:30:25.004		00:00:00.416	192.168.0.145	59879	224.0.0.252	5355 UDP	4	244	4692	61	2
2023-05-16 17:30:25.004		00:00:00.416	fe80::2..df:1957	59879	ff02::1:3	5355 UDP	4	324	6230	81	2
2023-05-16 17:30:28.002		00:00:00.416	192.168.0.145	52215	224.0.0.252	5355 UDP	4	244	4692	61	2
2023-05-16 17:30:28.002		00:00:00.416	fe80::2..df:1957	52215	ff02::1:3	5355 UDP	4	324	6230	81	2
2023-05-16 17:30:31.802		00:52:41.370	192.168.0.122	34400	192.168.0.193	22 TCP	708748	51.7 M	130731	72	20
2023-05-16 17:30:31.802		00:52:41.370	192.168.0.193	22	192.168.0.122	34400 TCP	1.0 M	30.8 G	77.9 M	30181	20
2023-05-16 17:30:39.006		00:00:00.415	192.168.0.145	61968	224.0.0.252	5355 UDP	4	244	4703	61	2
2023-05-16 17:30:39.006	4d	21:43:49.422	fe80::2..df:1957	61968	ff02::1:3	5355 UDP	8	648	0	81	4
2023-05-16 17:30:42.004		00:00:00.414	fe80::2..df:1957	53752	ff02::1:3	5355 UDP	4	316	6106	79	2
2023-05-16 17:30:42.004		00:00:00.414	192.168.0.145	53752	224.0.0.252	5355 UDP	4	236	4560	59	2
2023-05-16 17:30:42.005	1d	22:50:04.421	fe80::2..df:1957	56482	ff02::1:3	5355 UDP	10	810	0	81	5

La commande indique l'adresse IP source, qui correspond à celle de l'agent A, ainsi que l'adresse IP de destination de l'agent B et le port 22 utilisé pour le transfert de fichiers.

Protocoles	UDP	TCP
Bytes	18092	33123041529
Pourcentages	0,000001 %	99,99 %

Dans ce scénario, seul le protocole TCP est utilisé pour le transfert de fichiers, car on utilise SFTP (SSH File Transfer Protocol), qui est un protocole sécurisé basé sur SSH (Secure Shell). SFTP garantit la confidentialité et l'intégrité des données lors du transfert en utilisant le protocole TCP pour assurer une transmission fiable et sécurisée. Par conséquent, l'utilisation exclusive du protocole TCP est justifiée par la nécessité d'une transmission sécurisée et fiable des fichiers via SFTP.

Graphique représentant le flux du scénario:



“Compression des données : Si les données que vous transférez sont compressées sur la machine A et décompressées sur la machine B, cela peut entraîner une réduction de

la taille des données pendant le transfert. Par conséquent, si les données compressées ont une taille initiale de 20 Go, elles peuvent être réduites à une taille inférieure lorsqu'elles sont transférées, par exemple 10 Go, mais lorsqu'elles sont décompressées sur la machine B, elles retrouvent leur taille d'origine de 20 Go. Ainsi, le flux de transfert pourrait afficher une taille totale de 40 Go."

Sur ce graphique, on peut apercevoir un pic qui correspond au transfert de fichiers de la machine B vers la machine A.

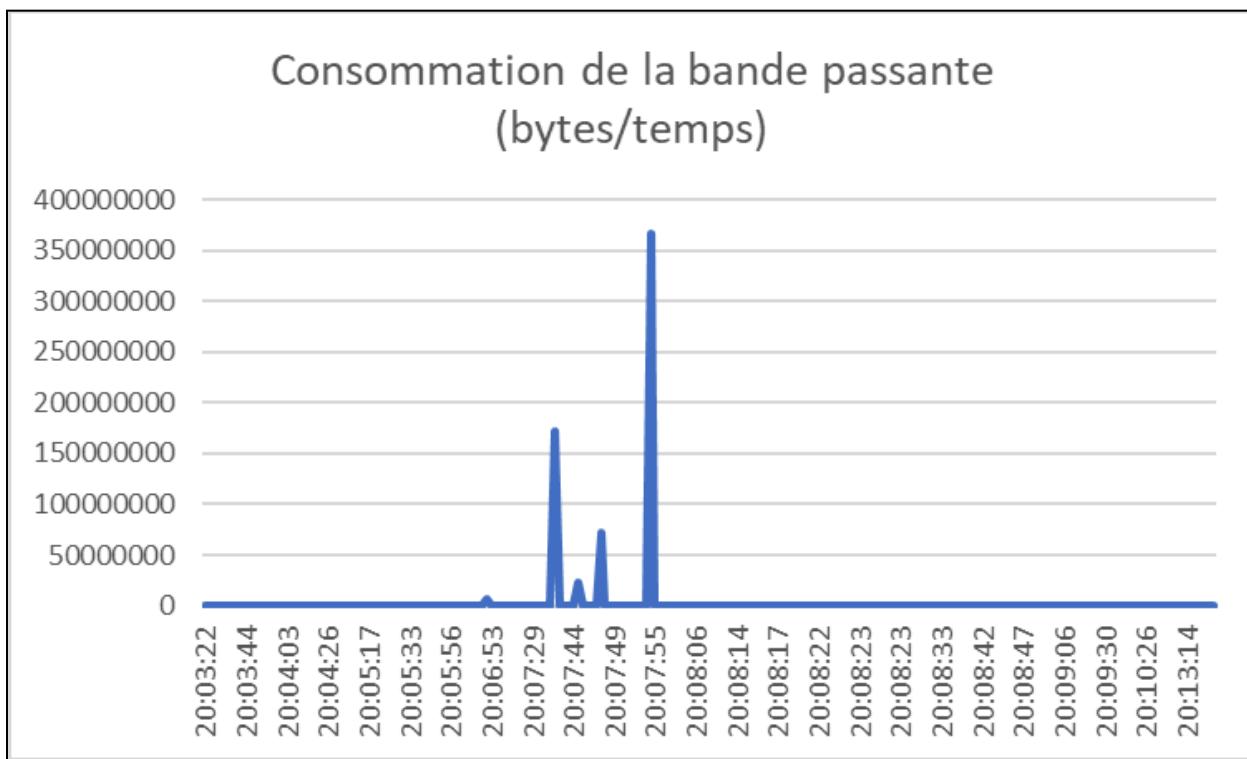
Scénario n°3 : L'agent B regarde des vidéos HD ou 4K sur youtube

Vidéo visualisée sur l'agent B : <https://www.youtube.com/watch?v=LXb3EKWslnQ>

Identification des flux correspondant à ce scénario :

2023-05-16 18:08:01.134	00:00:00.018	192.168.0.193	51812	62.197.111.140	53	UDP	4	296	131555	74	2
2023-05-16 18:08:01.134	00:00:00.018	62.197.111.140	53	192.168.0.193	51812	UDP	4	1116	496000	279	2
2023-05-16 18:08:01.154	00:02:51.319	34.117.237.239	443	192.168.0.193	37016	TCP	36	13136	613	364	8
2023-05-16 18:08:01.154	00:02:51.319	192.168.0.193	37016	34.117.237.239	443	TCP	42	4590	214	109	8
2023-05-16 18:08:01.437	00:00:00.025	62.197.111.140	53	192.168.0.193	36111	UDP	4	684	218880	171	2
2023-05-16 18:08:01.437	00:00:00.025	192.168.0.193	36111	62.197.111.140	53	UDP	4	240	76800	60	2
2023-05-16 18:08:01.445	00:00:00.017	192.168.0.193	57179	62.197.111.140	53	UDP	4	324	152470	81	2
2023-05-16 18:08:01.445	00:00:00.017	62.197.111.140	53	192.168.0.193	57179	UDP	4	1502	706823	375	2
2023-05-16 18:08:01.465	00:01:56.974	2a02:27..05:710d	43822	2a02:26..11:c5e2	80	TCP	50	7766	531	155	4
2023-05-16 18:08:01.465	00:01:56.974	2a02:26..11:c5e2	80	2a02:27..05:710d	43822	TCP	44	12070	825	274	4
2023-05-16 18:08:01.465	00:05:19.746	2a02:27..05:710d	38740	2600:19..:92a9::	443	TCP	66	7348	183	111	12
2023-05-16 18:08:01.465	00:05:19.746	2600:19..:92a9::	443	2a02:27..05:710d	38740	TCP	64	37986	950	593	12
2023-05-16 18:08:02.140	00:00:00.026	192.168.0.193	52509	62.197.111.140	53	UDP	4	284	87384	71	2
2023-05-16 18:08:02.140	00:00:00.026	62.197.111.140	53	192.168.0.193	52509	UDP	4	1252	385230	313	2
2023-05-16 18:08:02.157	00:00:00.020	62.197.111.140	53	192.168.0.193	60297	UDP	4	380	152000	95	2
2023-05-16 18:08:02.157	00:00:00.020	192.168.0.193	60297	62.197.111.140	53	UDP	4	292	116800	73	2
2023-05-16 18:08:02.171	00:00:00.020	192.168.0.193	60924	62.197.111.140	53	UDP	4	284	113600	71	2
2023-05-16 18:08:02.171	00:00:00.020	62.197.111.140	53	192.168.0.193	60924	UDP	4	632	252800	158	2
2023-05-16 18:08:02.178	00:00:00.485	2a02:27..05:710d	56704	2a00:14..d::200a	443	TCP	56	6956	114738	124	2
2023-05-16 18:08:02.178	00:00:00.485	2a00:14..d::200a	443	2a02:27..05:710d	56704	TCP	62	97494	1.6 M	1572	2
2023-05-16 18:08:02.178	00:00:00.485	34.117.65.55	443	192.168.0.193	41236	TCP	26	12662	175	487	6
2023-05-16 18:08:02.192	00:09:37.902	192.168.0.193	41236	34.117.65.55	443	TCP	30	4360	60	145	6
2023-05-16 18:08:02.242	00:00:00.023	192.168.0.193	35085	62.197.111.140	53	UDP	4	236	82086	59	2
2023-05-16 18:08:02.242	00:00:00.023	62.197.111.140	53	192.168.0.193	35085	UDP	4	960	333913	240	2
2023-05-16 18:08:02.257	00:00:00.032	62.197.111.140	53	192.168.0.193	55536	UDP	4	1158	289500	289	2
2023-05-16 18:08:02.257	00:00:00.032	192.168.0.193	55536	62.197.111.140	53	UDP	4	332	83000	83	2
2023-05-16 18:08:02.266	00:01:55.177	2a02:27..05:710d	50848	2a00:14..0::2003	80	TCP	34	3302	229	97	4
2023-05-16 18:08:02.266	00:01:55.177	2a00:14..0::2003	80	2a02:27..05:710d	50848	TCP	32	3724	258	116	4
2023-05-16 18:08:02.292	00:00:00.269	34.149.100.209	443	192.168.0.193	53089	UDP	24	13550	402973	564	2
2023-05-16 18:08:02.292	00:00:00.269	192.168.0.193	53089	34.149.100.209	443	UDP	18	4678	139122	259	2
2023-05-16 18:08:02.378	00:02:51.095	192.168.0.193	48648	34.149.100.209	443	TCP	36	3734	174	103	8
2023-05-16 18:08:02.378	00:02:51.095	34.149.100.209	443	192.168.0.193	48648	TCP	28	12134	567	433	8
2023-05-16 18:08:02.705	00:00:00.120	192.168.0.193	48652	34.149.100.209	443	TCP	20	2080	138666	104	2
2023-05-16 18:08:02.705	00:00:00.120	34.149.100.209	443	192.168.0.193	48652	TCP	16	10714	714266	669	2
2023-05-16 18:08:09.003	00:00:00.413	192.168.0.145	65207	224.0.0.252	5355	UDP	4	244	4726	61	2
2023-05-16 18:08:09.003	00:00:00.413	fe80::2..df:1957	65207	ff02::1:3	5355	UDP	4	324	6276	81	2

Graphique représentant le flux du scénario:



Selon les observations du graphique, il est notable que le débit n'est pas constant, présentant plutôt des pics qui correspondent aux moments où la vidéo a été préchargée.

Utilisation des protocoles pendant le scénario:

Protocoles	UDP	TCP
Total Bytes	469451876	172763176
Pourcentage	73,1 %	26,9%

Dans ce scénario de lecture d'une vidéo YouTube en 4K, le protocole UDP est largement utilisé pour garantir un flux multimédia en continu, offrant une haute capacité de débit et une faible latence, ce qui assure une expérience de visionnage fluide et ininterrompue. C'est pourquoi le pourcentage d'utilisation d'UDP est plus élevé.

Parallèlement, le protocole TCP est employé pour assurer la fiabilité des transferts de données, gérer les contrôles de flux et permettre la retransmission des paquets en cas de perte. Dans ce contexte, bien que le protocole UDP prédomine, le protocole TCP est utilisé pour les aspects nécessitant une transmission fiable, tels que les contrôles de lecture.

Scénario n°4 : L'agent A lance un appel + messages sur Discord

Identification des flux correspondant à ce scénario :

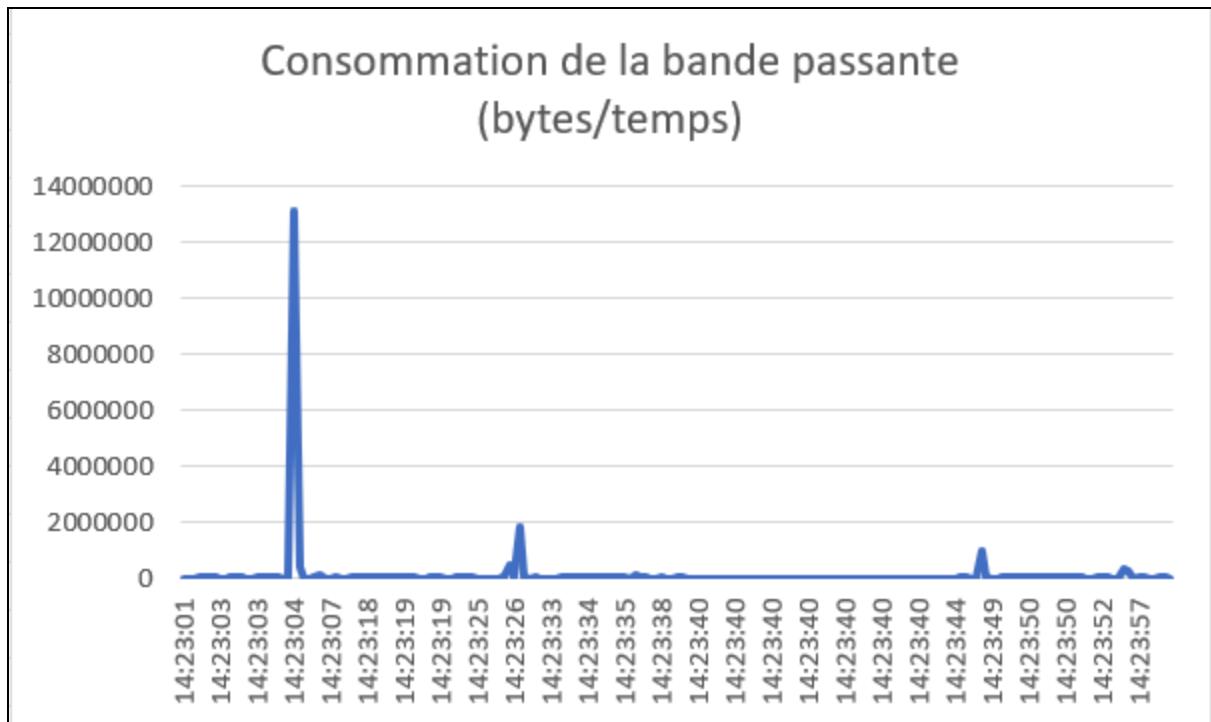
2023-05-21 14:23:46.137	00:00:00.204	192.168.0.1	53	192.168.0.122	37555	UDP	2	292	11450	146	2
2023-05-21 14:23:46.137	00:00:00.204	192.168.0.122	37555	192.168.0.1	53	UDP	2	132	5176	66	2
2023-05-21 14:23:46.164	00:05:21.732	162.159.129.232	443	192.168.0.122	57904	TCP	258	930902	23147	3608	16
2023-05-21 14:23:46.164	00:05:21.732	192.168.0.122	57904	162.159.129.232	443	TCP	220	15012	373	68	16
2023-05-21 14:23:48.837	00:00:00.599	fe80::2..df:1957	56719	ff02::1:3	5355	UDP	4	324	4327	81	2
2023-05-21 14:23:48.837	00:00:00.599	192.168.0.198	56719	224.0.0.252	5355	UDP	4	244	3258	61	2
2023-05-21 14:23:49.867	00:00:00.185	192.168.5.203	53538	239.255.255.250	1900	UDP	6	2994	129470	499	2
2023-05-21 14:23:49.886	00:00:00.175	192.168.5.203	44303	239.255.255.250	1900	UDP	2	1076	49188	538	2
2023-05-21 14:23:49.972	00:00:00.174	192.168.5.203	45181	239.255.255.250	1900	UDP	2	1076	49471	538	2
2023-05-21 14:23:49.976	00:00:00.174	192.168.5.203	59459	239.255.255.250	1900	UDP	2	1080	49655	540	2
2023-05-21 14:23:50.075	00:00:00.174	192.168.5.203	51044	239.255.255.250	1900	UDP	2	1076	49471	538	2
2023-05-21 14:23:50.081	00:00:00.173	192.168.5.203	59361	239.255.255.250	1900	UDP	2	1068	49387	534	2
2023-05-21 14:23:50.083	00:00:00.174	192.168.5.203	35978	239.255.255.250	1900	UDP	2	1088	50022	544	2
2023-05-21 14:23:50.090	00:00:00.266	192.168.5.203	39442	239.255.255.250	1900	UDP	4	2032	61112	508	2
2023-05-21 14:23:50.185	00:00:00.173	192.168.5.203	58501	239.255.255.250	1900	UDP	2	1076	49757	538	2
2023-05-21 14:23:50.279	00:00:00.174	192.168.5.203	56477	239.255.255.250	1900	UDP	2	1076	49471	538	2
2023-05-21 14:23:50.387	00:00:00.173	192.168.5.203	56648	239.255.255.250	1900	UDP	2	1088	50312	544	2
2023-05-21 14:23:50.393	00:00:00.177	192.168.5.203	55700	239.255.255.250	1900	UDP	4	1898	85785	474	2
2023-05-21 14:23:50.489	00:00:00.173	192.168.5.203	57157	239.255.255.250	1900	UDP	2	1076	49757	538	2
2023-05-21 14:23:50.588	00:00:00.172	192.168.5.203	52268	239.255.255.250	1900	UDP	2	1076	50046	538	2
2023-05-21 14:23:50.600	00:00:00.173	192.168.5.203	59213	239.255.255.250	1900	UDP	2	1076	49757	538	2
2023-05-21 14:23:50.601	00:00:00.172	192.168.5.203	55022	239.255.255.250	1900	UDP	2	1068	49674	534	2
2023-05-21 14:23:52.459	00:00:00.188	192.168.0.122	49965	192.168.0.1	53	UDP	2	146	6212	73	2
2023-05-21 14:23:52.459	00:00:00.188	192.168.0.1	53	192.168.0.122	49965	UDP	2	306	13021	153	2
2023-05-21 14:23:52.489	00:01:01.860	162.159.128.235	443	192.168.0.122	54236	TCP	108	17142	2216	158	2
2023-05-21 14:23:52.489	00:01:01.860	192.168.0.122	54236	162.159.128.235	443	TCP	115	17034	2202	148	2
2023-05-21 14:23:52.753	00:00:00.166	192.168.0.198	60978	99.181.79.12	443	TCP	2	3396	163662	1698	2
2023-05-21 14:23:53.484	00:00:00.223	66.22.199.229	50008	192.168.0.122	40674	UDP	2	204	7318	102	2
2023-05-21 14:23:53.484	00:00:00.223	192.168.0.122	40674	66.22.199.229	50008	UDP	2	204	7318	102	2
2023-05-21 14:23:53.484	00:01:00.188	66.22.199.229	50008	192.168.0.122	47660	UDP	2064	348288	46293	168	2
2023-05-21 14:23:53.484	00:01:00.188	192.168.0.122	47660	66.22.199.229	50008	UDP	1546	266312	35297	172	2
2023-05-21 14:23:54.850	00:00:00.584	192.168.0.198	65349	224.0.0.252	5355	UDP	4	244	3342	61	2
2023-05-21 14:23:56.892	00:00:00.159	192.168.0.198	60980	99.181.79.12	443	TCP	2	3396	170867	1698	2
2023-05-21 14:23:56.974	00:00:04.295	192.168.0.198	60983	52.223.192.54	443	TCP	8	6248	11637	781	2
2023-05-21 14:23:57.853	00:00:00.572	fe80::2..df:1957	64108	ff02::1:3	5355	UDP	4	324	4531	81	2
2023-05-21 14:23:57.853	00:00:00.572	192.168.0.198	64108	224.0.0.252	5355	UDP	4	244	3412	61	2
2023-05-21 14:23:57.884	00:00:53.602	192.168.0.198	60984	13.69.239.73	443	TCP	8	16098	2402	2012	4
2023-05-21 14:23:58.073	00:00:03.129	192.168.0.70	58679	239.255.255.250	1900	UDP	8	1568	4008	196	2
2023-05-21 14:23:59.852	00:00:00.566	192.168.0.198	49906	224.0.0.252	5355	UDP	4	224	3166	56	2
2023-05-21 14:24:00.863	00:00:00.575	192.168.0.198	53405	224.0.0.252	5355	UDP	4	236	3283	59	2
2023-05-21 14:24:01.081	00:00:00.154	192.168.0.198	60985	99.181.79.12	443	TCP	2	3396	176415	1698	2
2023-05-21 14:24:02.863	00:00:00.565	192.168.0.198	63605	224.0.0.252	5355	UDP	4	244	3454	61	2
2023-05-21 14:24:03.857	00:00:00.572	192.168.0.198	52273	224.0.0.252	5355	UDP	4	244	3412	61	2
2023-05-21 14:24:03.857	00:00:00.572	fe80::2..df:1957	52273	ff02::1:3	5355	UDP	4	324	4531	81	2
2023-05-21 14:24:04.869	00:00:00.569	192.168.0.198	53879	224.0.0.252	5355	UDP	4	204	2868	51	2
2023-05-21 14:24:05.230	00:00:00.149	192.168.0.198	60989	99.181.79.12	443	TCP	2	3396	182335	1698	2
2023-05-21 14:24:05.356	00:00:04.233	192.168.0.198	60990	52.223.192.54	443	TCP	8	6248	11808	781	2
2023-05-21 14:24:05.673	00:00:00.260	192.168.5.203	32950	239.255.255.250	1900	UDP	6	2994	92123	499	2
2023-05-21 14:24:05.785	00:00:00.148	192.168.5.203	38815	239.255.255.250	1900	UDP	2	1076	58162	538	2

Comment être sûr que ce sont bien les flux du scénario ?

Lors d'un appel sur Discord, il est possible d'obtenir l'adresse IP du serveur en effectuant un ping vers la région attribuée à l'appel. En réalisant cette opération, nous pouvons voir l'adresse IP du serveur sur lequel nous effectuons l'appel vocal. Par conséquent, cette adresse IP sera présente dans le flux de ce scénario.

```
(kali㉿agentB)-[~/pmacct]
└─$ ping rotterdam2989.discord.gg
PING rotterdam2989.discord.gg (66.22.199.229) 56(84) bytes of data.
64 bytes from 66.22.199.229 (66.22.199.229): icmp_seq=1 ttl=55 time=71.4 ms
64 bytes from 66.22.199.229 (66.22.199.229): icmp_seq=2 ttl=55 time=35.5 ms
64 bytes from 66.22.199.229 (66.22.199.229): icmp_seq=3 ttl=55 time=30.1 ms
^C
— rotterdam2989.discord.gg ping statistics —
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 30.069/45.672/71.442/18.356 ms
```

Graphique représentant le flux du scénario:



En examinant le graphique, on peut clairement identifier un pic de débit élevé, qui correspondrait dans notre contexte au moment où l'appel Discord est lancé. Une fois la connexion établie, le débit se stabilise, suggérant ainsi une communication fluide et sans problème. Les pics qui vont suivre seront ceux des messages envoyés depuis Discord.

Utilisation des protocoles pendant le scénario:

Protocoles	UDP	TCP
Bytes	14900000	1798850
Pourcentage	89,2 %	10,8 %

Dans le contexte des appels sur Discord, le protocole UDP est utilisé. Grâce à sa faible latence et à sa capacité à transmettre rapidement les données, il est idéal pour assurer une communication fluide et en temps réel pendant les appels sur Discord. Étant donné que les appels vocaux et vidéo nécessitent une transmission rapide des données, une certaine perte de paquets peut être tolérée sans compromettre la qualité de l'appel.

En revanche, pour les messages envoyés depuis Discord, le protocole TCP est utilisé pour garantir la transmission fiable des données. TCP assure que les données sont reçues dans l'ordre et sans perte, ce qui est essentiel pour la fiabilité des messages et des autres informations partagées dans Discord.

Identifier l'IP source qui a échangé le plus gros volume :

→ 192.168.0.193

Il s'agit de l'adresse IP de l'agent B (volume échangé lors du scénario SFTP)

Top 10 Src IP Addr ordered by bytes:										
Date first seen	Duration	Proto	Src IP Addr	Flows(%)	Packets(%)	Bytes(%)	pps	bps	bpp	
2023-05-10 19:33:59.095	7d 21:03:11.307	any	192.168.0.193	3620(2.5)	2.1 M(4.1)	61.6 G(48.1)	3	723745	29691	
2023-05-15 17:08:57.970	2d 23:03:37.110	any	2a02:27..f0:f::e	510(0.4)	40.6 M(80.5)	56.2 G(43.9)	158	1.8 M	1383	
2023-05-15 16:34:48.828	2d 23:58:25.810	any	2a02:a0..:2ff::2	103(0.1)	148167(0.3)	2.0 G(1.5)	0	60232	13166	
2023-05-16 18:08:38.298	00:05:42.972	any	2a00:14..0e:7::7	16(0.0)	1.1 M(2.1)	1.5 G(1.2)	3118	34.8 M	1393	
2023-05-16 18:14:28.737	1d 22:19:29.971	any	2a02:27..f0:f::c	62(0.0)	958501(1.9)	1.3 G(1.0)	5	63564	1382	
2023-05-15 16:34:48.759	2d 23:57:42.916	any	2a0b:5e..11:c4b	102(0.1)	113309(0.2)	1.2 G(0.9)	0	35682	10197	
2023-05-16 15:37:38.247	00:03:31.301	any	2620:0:..1a::2:b	32(0.0)	58678(0.1)	666.0 M(0.5)	277	25.2 M	11349	
2023-05-15 16:34:48.828	2d 23:58:07.575	any	2a00:15..0:15::2	225(0.2)	309744(0.6)	494.7 M(0.4)	1	15276	1597	
2023-05-16 18:33:07.332	00:14:21.875	any	99.181.67.23	30(0.0)	189407(0.4)	478.1 M(0.4)	219	4.4 M	2524	
2023-05-15 16:34:48.746	2d 23:58:07.661	any	212.68.211.198	91(0.1)	44995(0.1)	464.4 M(0.4)	0	14338	10320	

Identifier l'IP destination qui a échangé le plus gros volume :

→ 192.168.0.122

Il s'agit de l'adresse IP de l'agent A (volume échangé lors du scénario SFTP)

Top 10 Dst IP Addr ordered by bytes:										
Date first seen	Duration	Proto	Dst IP Addr	Flows(%)	Packets(%)	Bytes(%)	pps	bps	bpp	
2023-05-10 18:47:05.872	6d 01:01:06.488	any	192.168.0.122	3627(2.5)	2.4 M(4.7)	62.9 G(49.2)	4	963716	26371	
2023-05-15 17:08:36.529	03:05:26.144	any	2a02:27..41:69c6	1606(1.1)	40.6 M(80.6)	56.2 G(43.9)	3650	40.4 M	1383	
2023-05-16 15:23:50.821	2d 01:11:01.825	any	2a02:27..05:710d	845(0.6)	2.1 M(4.2)	3.2 G(2.5)	12	146838	1525	
2023-05-10 18:46:57.861	4d 23:52:18.900	any	2a02:27..ee:6564	1253(0.9)	544148(1.1)	3.2 G(2.5)	1	59435	5891	
2023-05-15 18:38:56.503	2d 21:54:35.541	any	2a02:27..921:8cb	1240(0.9)	145350(0.3)	671.0 M(0.5)	0	21328	4616	
2023-05-16 15:37:22.897	03:30:57.150	any	2a02:27..e7:70f4	849(0.6)	48911(0.1)	370.0 M(0.3)	3	233831	7563	
2023-05-15 16:34:48.828	2d 23:58:07.575	any	2a00:15..0:15::2	225(0.2)	326701(0.6)	362.1 M(0.3)	1	11181	1108	
2023-05-10 18:47:55.374	7d 21:49:15.028	any	192.168.0.221	5615(3.9)	144882(0.3)	249.5 M(0.2)	0	2920	1721	
2023-05-10 19:35:41.297	7d 20:58:33.547	any	192.168.0.193	2226(1.6)	1.4 M(2.9)	114.0 M(0.1)	2	1340	79	
2023-05-10 18:46:54.018	7d 21:49:59.993	any	ff02::fb	8757(6.2)	401934(0.8)	104.5 M(0.1)	0	1223	260	

Identifier le port source qui a échangé le plus gros volume :

→ port 22

Volume échangé lors du scénario SFTP

Top 10 Src Port ordered by bytes:										
Date first seen	Duration	Proto	Src Port	Flows(%)	Packets(%)	Bytes(%)	pps	bps	bpp	
2023-05-16 17:27:59.422	01:58:48.212	any	22	36(0.0)	2.0 M(4.0)	61.6 G(48.1)	285	69.1 M	30239	
2023-05-10 18:46:57.861	7d 21:47:16.983	any	443	8912(6.3)	43.2 M(85.6)	61.0 G(47.7)	63	713807	1412	
2023-05-15 16:34:48.746	2d 23:58:25.892	any	8080	619(0.4)	711954(1.4)	4.3 G(3.3)	2	131547	5984	
2023-05-10 18:46:54.017	7d 21:49:59.994	any	5353	17640(12.4)	817538(1.4)	205.0 M(0.2)	1	2399	250	
2023-05-16 19:18:18.808	00:08:28.826	any	37074	14(0.0)	719613(1.4)	52.3 M(0.0)	1414	822317	72	
2023-05-15 17:08:58.083	02:23:02.613	any	45366	377(0.3)	483570(1.0)	52.3 M(0.0)	56	48706	108	
2023-05-16 17:30:31.802	1d 23:01:59.892	any	34400	21(0.0)	708774(1.4)	51.7 M(0.0)	4	2441	72	
2023-05-15 11:19:26.128	06:00:43.613	any	33888	4(0.0)	10844(0.0)	18.4 M(0.0)	0	6797	1695	
2023-05-10 18:46:54.659	7d 21:50:15.461	any	0	14434(10.1)	176570(0.4)	17.8 M(0.0)	0	208	100	
2023-05-15 19:41:25.653	00:32:37.020	any	60134	106(0.1)	170735(0.3)	17.6 M(0.0)	87	71747	102	

Identifier le port source qui a échangé le plus gros volume :

→ port 45266

Top 10 Dst Port ordered by bytes:										
Date first seen	Duration	Proto	Dst Port	Flows(%)	Packets(%)	Bytes(%)	pps	bps	bpp	
2023-05-15 17:08:58.083	02:23:02.613	any	45366	377(0.3)	31.8 M(63.0)	44.0 G(34.4)	3701	41.0 M	1383	
2023-05-16 19:18:18.808	00:08:28.826	any	37074	14(0.0)	1.0 M(2.0)	30.8 G(24.1)	1996	483.9 M	30298	
2023-05-16 17:30:31.802	1d 23:01:59.892	any	34400	21(0.0)	1.0 M(2.0)	30.8 G(24.1)	6	1.5 M	30180	
2023-05-15 19:41:25.653	00:32:37.020	any	60134	106(0.1)	8.8 M(17.5)	12.2 G(9.5)	4497	49.8 M	1383	
2023-05-16 18:08:38.438	00:05:42.832	any	56022	12(0.0)	1.1 M(2.1)	1.5 G(1.2)	3116	34.5 M	1383	
2023-05-16 18:14:28.825	00:03:09.360	any	55454	8(0.0)	808931(1.6)	1.1 G(0.9)	4271	47.3 M	1383	
2023-05-15 16:34:48.746	2d 23:58:25.892	any	8080	619(0.4)	655912(1.3)	507.5 M(0.4)	2	15669	773	
2023-05-15 19:10:35.605	20:30:33.943	any	34128	12(0.0)	29550(0.1)	334.2 M(0.3)	0	36207	11308	
2023-05-16 18:33:07.332	00:08:22.160	any	40592	18(0.0)	128473(0.3)	332.3 M(0.3)	255	5.3 M	2586	
2023-05-16 15:37:38.248	00:02:50.237	any	36414	6(0.0)	28966(0.1)	331.7 M(0.3)	170	15.6 M	11451	

Identifier le couple d'IP qui a échangé le plus gros volume:

→ 192.168.0.122 - 192.168.0.183

Ces adresses IP sont significatives car elles font partie des adresses IP qui ont échangé le plus gros volumes de données en tant que IP source et destination.

Bibliographie

RSYSLOG:

<https://www.makeuseof.com/set-up-linux-remote-logging-using-rsyslog/>

https://www.rsyslog.com/doc/master/tutorials/recording_pri.html

<https://sematext.com/blog/linux-logs/>

<https://installati.one/install-rsyslog-kalilinux/>

SEC:

<https://www.youtube.com/watch?v=3ghNtB7yg1Q>

<https://simple-evcorr.github.io/>

<https://simple-evcorr.github.io/man.html>

SNMP:

<https://www.poftut.com/how-to-install-and-use-snmp-on-linux-tutorial-with-examples/>

<http://www.linux-admins.net/2012/02/linux-snmp-oids-for-cpumemory-and-disk.html>

<https://grafana.com/>

NETFLOW:

<https://www.youtube.com/watch?v=LXb3EKWsInQ>

<https://nfdump.sourceforge.net/>

<https://github.com/pmacct/pmacct/tree/1.7.8>

<https://mattjhayes.com/2018/08/19/collecting-netflow-with-nfcapd-and-nfdump/>

<https://mattjhayes.com/2018/08/22/netflow-ipfix-exporting-with-pmacct/>