


DESCRIPTION D'UNE SITUATION PROFESSIONNELLE

PARCOURS	SISR <input checked="" type="checkbox"/>	SLAM <input type="checkbox"/>
-----------------	---	--------------------------------------

Lieu de réalisation	Entreprise	
Période de réalisation	Du : 06/06/25	Au : 19/06/25
Modalité de réalisation	SEUL <input checked="" type="checkbox"/>	EN EQUIPE <input type="checkbox"/>

Intitulé de la mission	Automatisation hebdomadaire de l'export des assets CrowdStrike vers SharePoint
Description du contexte de la mission	<p>Dans le cadre du suivi de la sécurité du parc informatique supervisé par CrowdStrike, un export des données des assets détectés est requis chaque semaine.</p> <p>Mon tuteur m'a confié la tâche d'automatiser ce processus via un job Jenkins hebdomadaire qui exécute un script Python et PowerShell.</p>

Contraintes & Résultat	Ressources fournies / contraintes techniques / Résultats attendu
	<p>Exporter les données des endpoints gérés par CrowdStrike sur les 7 derniers jours.</p> <p>Transférer automatiquement ce fichier sur un espace SharePoint sécurisé.</p>
Productions associées	Liste des documents produits et description
	<p>Script Python utilisant la librairie FalconPy pour interroger l'API CrowdStrike.</p> <p>Script PowerShell pour structurer le fichier de sortie et l'envoyer sur SharePoint via PnP PowerShell.</p> <p>Job Jenkins hebdomadaire planifié : Exploit-Export Asset MDR - Hebdo.</p>

Modalités d'accès aux productions	Identifiants, mots de passe, URL d'un espace de stockage et présentation de l'organisation du stockage
	Nom de fichier : assetMDR_<date du jour>.csv

Description détaillée de la situation professionnelle retenue et des productions réalisées

en mettant en évidence la démarche suivie, les méthodes et les techniques utilisées

La tâche m'a été confiée dans le but de fiabiliser et d'automatiser l'export hebdomadaire des données CrowdStrike.

J'ai d'abord modifié un script Python basé sur la librairie **FalconPy** pour extraire les données des hôtes sur les 7 derniers jours, en boucle sur plusieurs CIDs.

Les résultats sont formatés dans un fichier CSV.

J'ai ensuite automatisé cette tâche avec un script **PowerShell** :

- Définition des dates en variables d'environnement.
- Déplacement et renommage du fichier CSV exporté.
- Connexion à SharePoint via **PnP PowerShell** en utilisant les identifiants stockés dans les variables d'environnement.
- Envoi automatique du fichier dans le bon répertoire SharePoint.

Le tout est orchestré via **Jenkins**, programmé pour s'exécuter chaque semaine.

En cas d'erreur, des messages explicites sont générés et l'exécution s'arrête, garantissant un suivi fiable.

jenkins : Exploit-Export Asset MDR - Hebdo

Dates optionnelles (laisser vide pour mode automatique 7 derniers jours)FORMAT: YYYY-MM-DD

\$env:DATE_DEBUT = ""

\$env:DATE_FIN = "" #La date de fin est exclue

cd "\\fr\asp01\Jobs_Exploit\Exploit\crowdstrike"

python Export_Asset_MDR.py

if (\$LASTEXITCODE -ne 0) {

Write-Host "Erreur pendant l'exécution du script Python"

exit 1

}

\$today = Get-Date -Format "yyyy-MM-dd"

\$exportFolder = "C:\Temp"

\$exportFile = "assetMDR_\$today.csv"

```
if (!(Test-Path -Path $exportFolder)) {  
    New-Item -ItemType Directory -Path $exportFolder | Out-Null  
}  
  
Move-Item -Path "assetMDR.csv" -Destination (Join-Path $exportFolder $exportFile) -Force  
Write-Host "Export terminé : $exportFile"  
#####  
  
# Importer le module PnP PowerShell pour SharePoint  
Import-Module SharePointPnPPowerShellOnline  
  
# Définir les informations de connexion à SharePoint  
$siteURL = "https://lehub.sharepoint.com/sites/P353-GroupEDRRenewal-Run"  
  
# Récupérer les informations d'identification depuis les variables d'environnement  
$UserName = $ENV:O365user  
$Password = ConvertTo-SecureString $ENV:O365pass -AsPlainText -Force  
  
# Créer un objet PSCredential pour l'authentification  
$Credential = New-Object System.Management.Automation.PSCredential($UserName, $Password)  
  
# Se connecter à SharePoint via PnP PowerShell avec l'authentification  
Connect-PnPOnline -Url $siteURL -Credentials $Credential -WarningAction Ignore  
if (Get-PnPContext) {  
    Write-Host "Connexion a SharePoint reussie."  
} else {  
    Write-Error "Echec de la connexion a SharePoint."  
    exit 1  
}  
# Définir le dossier de destination sur SharePoint  
$sharePointFolder = "Shared Documents/Run/Asset"  
  
# Télécharger le fichier CSV vers SharePoint  
Add-PnPFile -Path "C:\Temp\$exportFile" -Folder $sharePointFolder
```

Python :

```
from falconpy import Hosts
from datetime import datetime, timedelta, timezone
import os
import csv
```

```
# Format : YYYY-MM-DD (exemple : "2025-06-01")
DATE_DEBUT = os.getenv("DATE_DEBUT", "")
DATE_FIN = os.getenv("DATE_FIN", "") #DATE_FIN Exclue
```

```
if DATE_DEBUT and DATE_FIN:
```

```
    start_date = datetime.strptime(DATE_DEBUT, "%Y-%m-%d").replace(tzinfo=timezone.utc)
    end_date = datetime.strptime(DATE_FIN, "%Y-%m-%d").replace(tzinfo=timezone.utc)
    nb_days = (end_date - start_date).days
```

```
    if nb_days <= 0:
```

```
        raise ValueError("La date de fin doit être après la date de début.")
```

```
    date_range = [start_date + timedelta(days=i) for i in range(nb_days)]
```

```
else:
```

```
    date_range = [
```

```
        (datetime.now(timezone.utc) - timedelta(days=i + 1)).replace(hour=0, minute=0, second=0,
microsecond=0)
```

```
        for i in range(7)
```

```
    ]
```

```
# Chargement des credentials
```

```
CLIENT_ID = os.getenv('CLIENT_ID')
```

```
CLIENT_SECRET = os.getenv('CLIENT_SECRET')
```

```
# Dictionnaire des CIDs par nom
```

```
CIDS = {
```

```
    "2df75cebdbed46b59cd581b9efd1d4e3": "2df75cebdbed46b59cd581b9efd1d4e3",
```

```
    "9196f612c7b4c85a35638188b8601f6": "9196f612c7b4c85a35638188b8601f6",
```

```
    "6962f504682f4919a29c57acef12c05d": "6962f504682f4919a29c57acef12c05d",
```

```
    "5c9c9562a21b4922aa510c1a59c42ed7": "5c9c9562a21b4922aa510c1a59c42ed7",
```

```
    "0d7dea91d668446f9e73594137d5a9c9": "0d7dea91d668446f9e73594137d5a9c9",
```

```
"35253090b1aa4d5895b03d9a0902c1d8": "35253090b1aa4d5895b03d9a0902c1d8",  
"4fa86cdf7f9e4712975b203e0dc81100": "4fa86cdf7f9e4712975b203e0dc81100",  
"5860b8c5b47b46f5a5e123b0983d22dd": "5860b8c5b47b46f5a5e123b0983d22dd"  
}
```

```
results = []
```

```
for cid_name, cid_value in CIDS.items():  
    print(f"Traitement du CID {cid_name} ({cid_value})")
```

```
    falcon = Hosts(  
        client_id=CLIENT_ID,  
        client_secret=CLIENT_SECRET,  
        base_url="https://api.eu-1.crowdstrike.com",  
        cid=cid_value  
    )
```

```
    for day_start in date_range:  
        day_end = day_start + timedelta(days=1)
```

```
        start_str = day_start.strftime('%Y-%m-%dT%H:%M:%SZ')  
        end_str = day_end.strftime('%Y-%m-%dT%H:%M:%SZ')  
        filter_str = f"last_seen:>='{start_str}'+last_seen:<='{end_str}'"
```

```
        print(f" -> Requête pour le {day_start.date()}")
```

```
        device_ids = []  
        total_limit = 10000  
        limit = 100  
        offset = 0
```

```
        while True:  
            if offset + limit > total_limit:  
                limit = total_limit - offset
```

```
if limit <= 0:
    break

query = falcon.QueryDevicesByFilter(filter=filter_str, limit=limit, offset=offset)
if query['status_code'] != 200:
    print(f" Erreur lors de la récupération des devices pour {cid_name} ({start_str}) :", query)
    break

ids = query["body"]["resources"]
device_ids.extend(ids)

meta = query["body"]["meta"]
total = meta["pagination"]["total"]
offset += limit

if offset >= total or offset >= total_limit:
    break

print(f" {len(device_ids)} devices récupérés pour cette journée.")

# Par lots de 10
def chunks(lst, n):
    for i in range(0, len(lst), n):
        yield lst[i:i + n]

for batch in chunks(device_ids, 10):
    details = falcon.GetDeviceDetails(ids=batch)
    if details['status_code'] != 200:
        print(f" Erreur récupération détails pour {cid_name} :", details)
        continue

    for device in details["body"]["resources"]:
        results.append({
            "cid": cid_name,
```

```
"hostname": device.get("hostname", ""),  
"agent_version": device.get("agent_version", ""),  
"last_reboot": device.get("last_reboot", ""),  
"os_version": device.get("os_version", ""),  
"last_seen": device.get("last_seen", ""),  
"active_directory_domain": device.get("machine_domain", "")  
})
```

Export CSV

```
csv_file = "assetMDR.csv"
```

```
with open(csv_file, mode='w', newline="", encoding='utf-8') as f:
```

```
    writer = csv.DictWriter(f, fieldnames=["cid", "hostname", "agent_version", "last_reboot",  
"os_version", "last_seen", "active_directory_domain"])  
    writer.writeheader()  
    writer.writerows(results)
```

```
print(f"Export terminé dans le fichier {csv_file}")
```

