

CAMPAGNE DE PHISHING – PROCEDURE MANUELLE

1) Contexte

L'idée de ce devoir est de mettre en place un environnement local de test de phishing à petite échelle afin d'envoyer des mails ciblés à des utilisateurs définis, suivre les potentiels clics sur les liens envoyés, récolter les informations des formulaires (email et mot de passe) et tracer les utilisateurs à partir de l'ouverture de pièces jointes piégées. Il doit également être possible de naviguer à partir d'une page d'accueil permettant de relancer la campagne mais aussi visualiser les résultats sur une page regroupant les statistiques par utilisateur.

2) Scénario

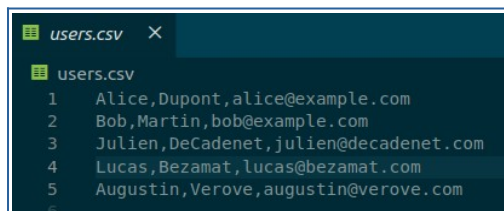
Pour cette procédure manuelle, j'ai décidé de me mettre dans la peau d'un membre haut placé d'une entreprise quelconque qui souhaiterait mener une campagne de phishing au sein de sa structure afin de s'assurer que tous les stagiaires en cours de contrat soient sensibilisés à ce risque. Les stagiaires étant les membres des équipes possédant souvent le moins d'expérience, ce sont les plus à même de se faire piéger croyant simplement avoir reçu un mail venant de plus haut. On parlera alors ici de social engineering et de mass phishing en fonction de la taille de l'entreprise et du nombre de stagiaires qu'elle possède. En ce qui concerne l'hameçon, il s'agira d'un mail indiquant un besoin urgent d'effectuer une mise à jour OS en exécutant un script fourni en pièce jointe ainsi qu'un formulaire d'identifiants à remplir afin de ne pas être importuné par une maintenance.

3) Mise en place de la procédure

3.1) Fichiers nécessaires au fonctionnement

Dans un premier temps, il est primordial de s'assurer que quelques fichiers soient bien mis en place pour pouvoir faire fonctionner l'outil :

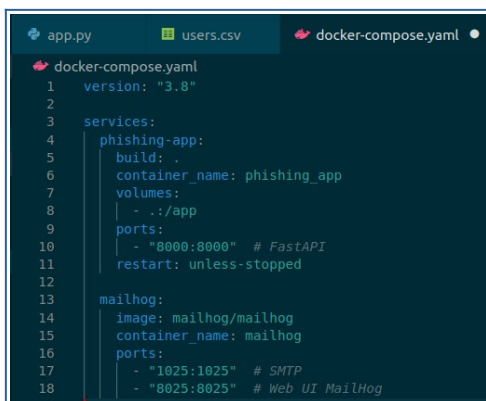
- **users.csv** : regroupe la liste des cibles avec une ligne pour chaque utilisateur contenant prénom, nom, e-mail (exemple ci-contre).



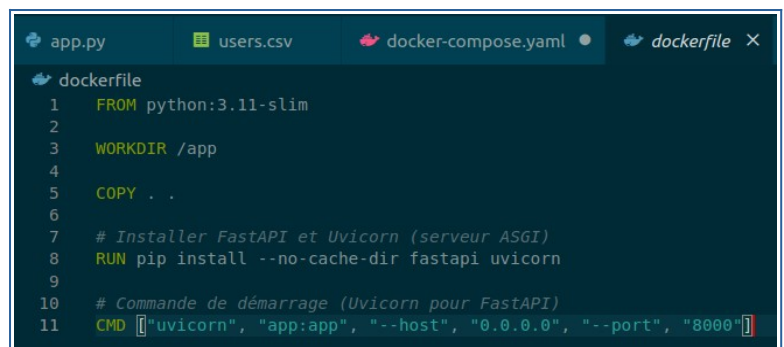
```
users.csv
1 Alice,Dupont,alice@example.com
2 Bob,Martin,bob@example.com
3 Julien,DeCadenet,julien@decadenet.com
4 Lucas,Bezamat,lucas@bezamat.com
5 Augustin,Verove,augustin@verove.com
```

- **app.py** : il s'agit du fichier FastAPI qui contient le corps de l'application web.

- **docker-compose.yaml** + **dockerfile** : permettent de setup le container docker car nous dockerisons le devoir pour faciliter la re-cr  ation du container sur la VM Debian par la suite.



```
docker-compose.yaml
1 version: "3.8"
2
3 services:
4   phishing-app:
5     build: .
6     container_name: phishing_app
7     volumes:
8       - ./app
9     ports:
10      - "8000:8000" # FastAPI
11     restart: unless-stopped
12
13   mailhog:
14     image: mailhog/mailhog
15     container_name: mailhog
16     ports:
17       - "1025:1025" # SMTP
18       - "8025:8025" # Web UI MailHog
```



```
dockerfile
1 FROM python:3.11-slim
2
3 WORKDIR /app
4
5 COPY . .
6
7 # Installer FastAPI et Uvicorn (serveur ASGI)
8 RUN pip install --no-cache-dir fastapi uvicorn
9
10 # Commande de d  marrage (Uvicorn pour FastAPI)
11 CMD ["uvicorn", "app:app", "--host", "0.0.0.0", "--port", "8000"]
```

3.2) Déploiement de l'application

Avant de lancer l'outil, il faut penser à modifier le fichier `app.py` aux lignes **145** et **156** avec l'adresse locale qui peut différer selon le réseau sur lequel la machine est connectée et le port utilisé que l'on retrouve dans le fichier « ***docker-compose.yaml*** » ci-dessus.

```
145 curl http://192.168.1.47:8000/attachment-click/{idx}
146
147 """
148
149 # Création mail multipart
150 msg = MIMEMultipart()
151 msg['Subject'] = "Maintenance et Mise à jour OS"
152 msg['From'] = FROM_EMAIL
153 msg['To'] = email
154
155 # Corps du mail
156 link = f"http://192.168.1.47:8000/click/{idx}"
157 body = textwrap.dedent(f"""
```

Screenshot de la portion de code à modifier dans le fichier `app.py`.

Puis, pour lancer l'API, il suffit d'utiliser la commande « ***uvicorn app:app --host 0.0.0.0 --port 8000 --reload*** » puis de se rendre sur le navigateur à l'adresse « ***http://{IP_HÔTE}:8000*** » avec `IP_HÔTE` étant l'adresse du réseau local que l'on vient d'évoquer précédemment.

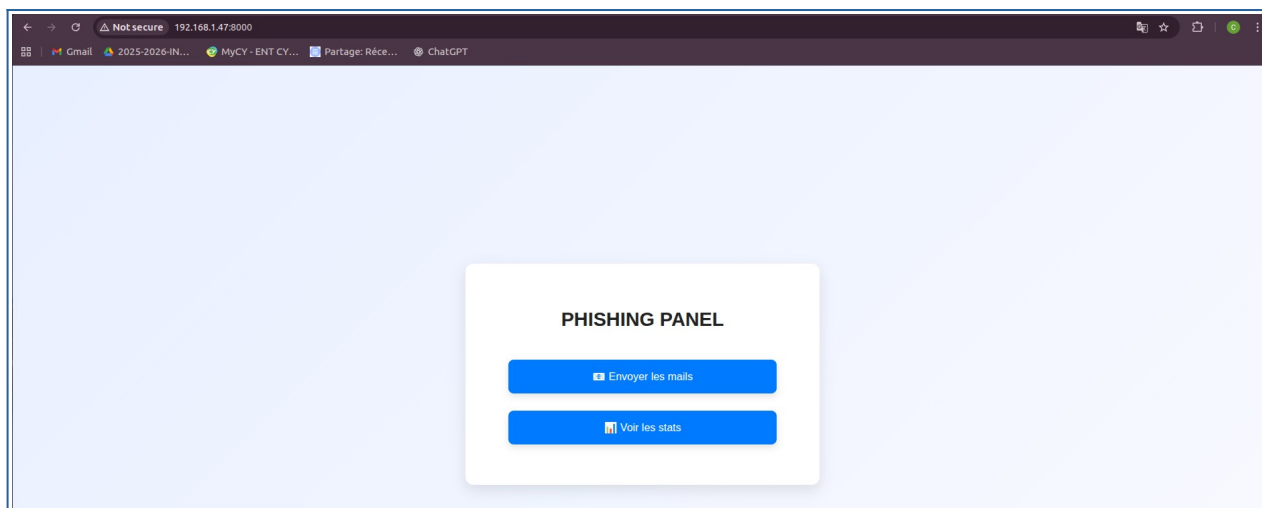
À propos des paramètres : « ***uvicorn*** » correspond au serveur ASGI permettant de lancer des applications web asynchrones comme FastAPI, le premier « ***app*** » désigne le module (`app.py`) et le deuxième l'objet que l'on définit dans ce même fichier, « ***--host 0.0.0.0*** » signifie que `uvicorn` doit écouter toutes les interfaces réseau du serveur, « ***--port 8000*** » définit le port TCP sur lequel le serveur écoute les requêtes et enfin « ***--reload*** » active le rechargement automatique.

4) Fonctionnement de l'application

4.1) Page d'accueil

Elle affiche deux boutons qui permettent :

- d'envoyer les mails en suivant la route « ***/send-mails*** » dans `app.py`
- voir les statistiques en redirigeant vers la route « ***/stats*** » dans `app.py`



Screenshot de la page d'accueil.

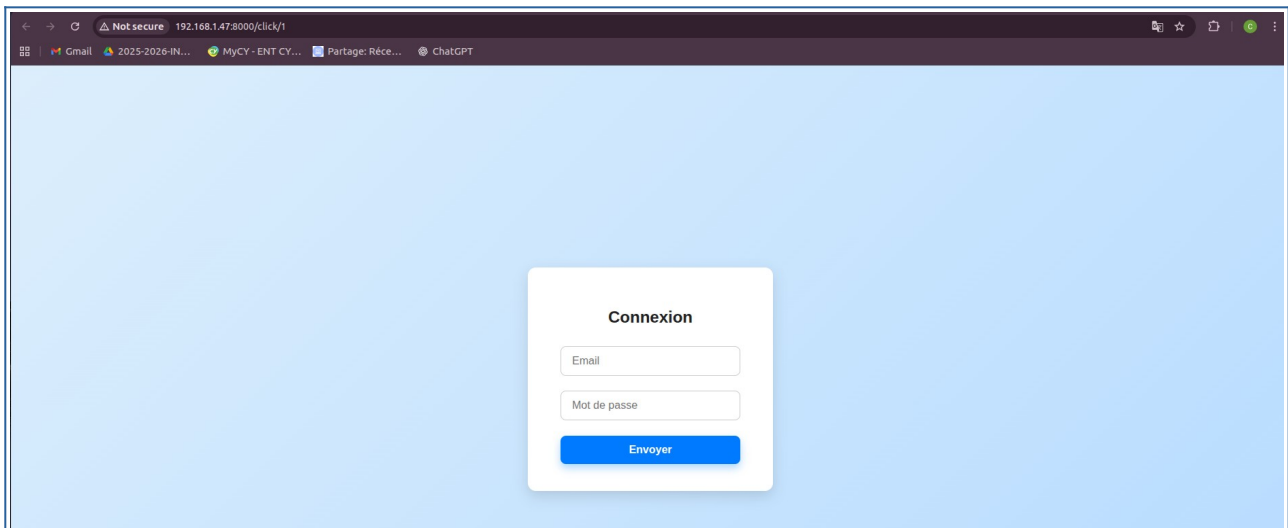
4.1.1) Envoi de mails

En cas de clic sur le bouton d'envoi de mails, l'application parcourt chaque ligne du « **users.csv** », génère un lien unique par utilisateur « **http://{IP_HÔTE}/click/{id}** », cela à l'aide de la méthode « **/send-mails** » et plus précisément à la ligne 141 qui crée un lien en fonction de l'id de l'utilisateur. Puis, elle envoie un email via SMTP (ici smtp.gmail.com → défini au début du fichier app.py) contenant ce lien ainsi qu'une pièce jointe unique. Enfin, elle redirige vers une page de confirmation qui renvoie vers l'accueil.



Screenshot du mail envoyé à chaque utilisateur cible.

Comme expliqué dans le scénario, lors du clic sur le lien, l'utilisateur est renvoyé vers une autre page contenant un faux formulaire de connexion dans lequel il est prié de renseigner ses identifiants.



Screenshot du faux formulaire de connexion.

Lors de la soumission du formulaire, l'application récupère les infos d'origine (prenom, nom, email_origine) depuis « **users.csv** » puis stocke dans « **clicks.csv** » id, email d'origine, date et heure du clic, adresse IP, email saisi dans le formulaire (email_form), mot de passe. Enfin, l'utilisateur est redirigé vers une fausse page qui indique que l'entreprise concernée le remercie de sa confiance et que le formulaire a été soumis avec succès.

À propos de la pièce jointe, celle-ci représente un fichier bash que la cible est priée d'exécuter. En réalité, il ne s'agit que d'un script comportant un curl vers l'adresse du réseau local afin de pouvoir récupérer l'information comme quoi celui-ci a été exécuté. En effet, cette procédure ayant été réalisée avec pour seul but de tracer la pièce jointe, c'est ici sa seule utilité, cependant, s'agissant de la procédure manuelle et possédant ainsi beaucoup moins de restrictions qu'une méthode automatisée, nous pourrions intégrer dans les mails bien plus de fichiers malveillants que d'ordinaire (reverse shell en .bat par exemple).

```

1 #!/bin/bash
2 curl http://192.168.1.47:8000/attachment-click/1
3

```

Screenshot du fichier bash que l'on transmet en pièce jointe.

4.1.2) Page de statistiques :

Premièrement, elle lit chaque ligne du fichier « **clicks.csv** » qui correspond aux résultats de la méthode de phishing avec formulaire, puis affiche un tableau HTML avec les informations pour chaque formulaire soumis.

Deuxièmement, elle lit chaque ligne du fichier « **opens.csv** » qui correspond aux résultats de la méthode de phishing avec pièce jointe, puis affiche un tableau HTML avec les informations pour chaque script exécuté. En ce qui concerne le remplissage de ce fichier dont nous n'avons pas encore parlé, à chaque exécution du script fourni en pièce jointe, il envoie une requête HTTP avec curl vers le serveur FastAPI en appelant la route « **/attachment-click/{user_id}** » qui s'occupe d'ajouter une ligne au .csv avec la date d'exécution et l'adresse IP.

Informations récupérées

Méthode formulaire :

User ID	Prénom	Nom	Email	Date du clic	IP	Email formulaire	Password
2	Clement	Durecu	clement.shoddox@gmail.com	2025-09-17T12:44:00.078098	172.20.10.7	test@test.test	password
1	Clement	Durecu	clement.shoddox@gmail.com	2025-09-17T13:09:04.445695	172.20.10.7	durecuclem@cy-tech.fr	test
1	Clement	Durecu	clement.shoddox@gmail.com	2025-09-17T13:14:44.649753	172.20.10.7	durecuclem@cy-tech.fr	tsewdt
2	Durecu	Clement	durecuclem@cy-tech.fr	2025-09-17T13:27:56.493149	172.20.10.7	durecuclem@cy-tech.fr	test
1	Clement	Durecu	clement.shoddox@gmail.com	2025-09-18T00:25:18.356212	192.168.1.47	test@test.test	aaaaa

Méthode pièce-jointe :

User ID	Date d'exécution	IP
3	2025-09-17T12:43:37.199678	172.20.10.7

[Retour au Phishing Panel](#)

Screenshot de la page de statistiques.