

Sophie VALENTIN, Mathieu BIVERT

Configuration et sécurisation de services réseaux
5 février 2013

Professeur : Bruno MARTIN



Table des matières

1	Scénario	3
2	Topologie	3
3	Mise en place d'une passerelle et accès à Internet	3
3.1	Tests	3
4	Mise à disposition d'un accès distant sur la machine et sécurisation	4
4.1	Serveur Telnet et attaques de <i>sniffing</i>	4
4.2	Serveur SSH et optimisations	5
5	Configuration d'un serveur web et d'OpenSSL	6
5.1	Installation d'OpenSSL	6
5.2	Installation d'Apache2	6
5.3	Passage à https	7
5.4	Tests	7
6	Configuration serveur (smtp+imaps) et client (gpg) email	7
6.1	Envoi et transfert d'emails avec Postfix (MTA)	7
6.2	Récupération d'emails avec Dovecot (MDA) et Thunderbird (MUA)	7
6.3	Des messages chiffrés et signés avec GnuPG	7
6.3.1	Principe	7
6.3.2	Mise en pratique	8
6.4	Tests	8
7	Configuration VPN	8
7.1	Tests	8
8	OpenVAS & Metasploit	8

1 Scénario

Serveur « public » fournissant accès distant (VPN), compte email (postfix, dovecot, client email), espace web (apache).

Encryption autant que faire ce peut.

Logs.

Outils d'audit.

2 Topologie

3 Mise en place d'une passerelle et accès à Internet

Une passerelle (gateway) est un homme du milieu reliant deux réseaux distincts. Dans le cas présent, la machine *passerelle* doit faire communiquer les deux réseaux SLAN (192.168.1.0/24) et LAN Travaux Pratiques (192.168.2.0/24).

La passerelle doit être capable de transmettre des paquets IP :

```
(passerelle)# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Afin de maintenir l'*IP forwarding* après un reboot de la machine *passerelle*, on décommente dans le fichier */etc/sysctl.conf* la ligne suivante :

```
#net.ipv4.ip_forward=1
```

L'IP Masquerade (Network Address Translation) doit être activée. Cette fonctionnalité modifie les entêtes IPs du trafic passant par *passerelle* afin de rendre invisibles, au niveau IP, les machines de LAN Travaux Pratiques depuis l'extérieur. Ici on utilise du NAT dit de source utilisant la chaîne *POSTROUTING* puisqu'on modifie les adresses sources du paquet.

```
(passerelle)# iptables -t nat -A POSTROUTING -o eth0 -s 192.168.2.0/24 -j MASQUERADE
```

Note : L'interface *eth0* est connectée au réseau SLAN.

Enfin, syslogd doit être activé afin de logger les activités d'iptables

```
(passerelle)# apt-get install inetutils-syslogd
```

```
(passerelle)# edit /etc/syslog.conf # logs dans /var/log/kernel.log
```

```
(passerelle)# services syslog
```

3.1 Tests

On choisit un client, par exemple *client-bsd*. On lui retire l'interface réseau connectée à SLAN *em0*, et on s'assure que la machine est bien connectée sur LAN Travaux Pratiques via *em1*, et qu'elle peut communiquer avec la passerelle.

```
(client-bsd)# ifconfig em0 down
```

```
(client-bsd)# ifconfig em1
```

```
(client-bsd)# ping passerelle.cs.sr
```

Puis, on configure la table de routage du client afin que la route par défaut passe par *passerelle* et on vérifie :

```
(client-bsd)# netstat -r
```

Enfin, on s'assure qu'il est possible de contacter le serveur et d'atteindre Internet.

```
(client-bsd)# ping -c3 google.fr
```

Internet:						
Destination	Gateway	Flags	Refs	Use	Netif	Expire
default	192.168.2.1	UGS	0	4	em1	
localhost	link#4	UH	0	160	lo0	
192.168.1.3	link#1	UHS	0	0	lo0	
192.168.2.0	link#2	U	0	7	em1	
obsd.localdomain	link#2	UHS	0	0	lo0	

FIGURE 1 – Table de routage de *client-bsd*

On vérifie les logs sur la passerelle :

```
(passerelle)# tail -f /var/log/kernel.log
```

nmap? Faire peut-être une partie "outils d'attaque et... d'audit" où on explique l'utilisation de nmap, metasploit, vas (TP de demain) et en quoi ils nous aident à sécuriser notre propre réseau.

4 Mise à disposition d'un accès distant sur la machine et sécurisation

Telnet, SSH, VPN Expliquer comment mettre en place TCP Wrapper avec Telnet mais pas sécurisé. essai ssh avec mitm? (normalement il affiche un message du genre "SOMEONE MAY BE ON THE CABLE!")

4.1 Serveur Telnet et attaques de *sniffing*

On commence par installer un serveur *telnet* afin d'offrir un premier accès distant à la machine *passerelle*.

On n'utilisera pas ici le démon classique de telnet mais un démon générique nommé *inetd* permettant de minimiser le nombre de processus lancés. Quand *inetd* reçoit une demande de connexion, il lance le processus serveur adéquat.

Pour ce faire, on installe le serveur telnet.

```
(passerelle)# apt-get install telnetd
```

Puis, on modifie le fichier */etc/inetd.conf* pour y ajouter le service *telnet* dans la section des services standard, de façon à ce qu'il soit démarré automatiquement en tant qu'utilisateur telnetd :

```
(passerelle)# grep '^telnet' /etc/inetd.conf
telnet      stream    tcp nowait telnetd /usr/sbin/tcpd /usr/sbin/in.telnetd
```

On va maintenant tenter d'intercepter le couple *login/passwd* lors d'une connexion *telnet* entre *client-bsd* et *passerelle*.

Pour opérer, on utilise l'outil *ettercap* depuis la machine BackTrack. Il faut le lancer ainsi :

```
(backtrack)# ettercap -G
```

1. Dans le menu, on clique tout d'abord sur *Sniff* et on sélectionne *Unified sniffing* avant de choisir l'interface *eth1* qui est sur LAN Travaux Pratiques.
2. Dans le menu *Hosts*, on lance un scan du réseau local puis depuis le même menu, on affiche la liste des hôtes.

On reconnaît les adresses IP de *passerelle* et *client-bsd*. Il suffit donc de spécifier que ce sont nos deux cibles avec les boutons *TARGET 1* et *TARGET 2*.

3. Comme il s'agit d'un réseau commuté, on doit empoisonner les tables ARP (*ARP poisoning*) avant de lancer le *sniffing* pour récupérer le trafic entre les deux cibles.

Dans le menu *Mitm*, on sélectionne donc *ARP Poisoning*.

4. Enfin, on sélectionne dans le menu *Start* le choix *Start Sniffing*.

La machine *client-bsd* se connecte avec *telnet*.

Et on constate qu'il est extrêmement facile d'obtenir le couple *login/password* en clair :

On ne veut donc pas proposer un service d'accès à distance non sécurisé, c'est-à-dire basé sur des communications transmises en clair sur le réseau.

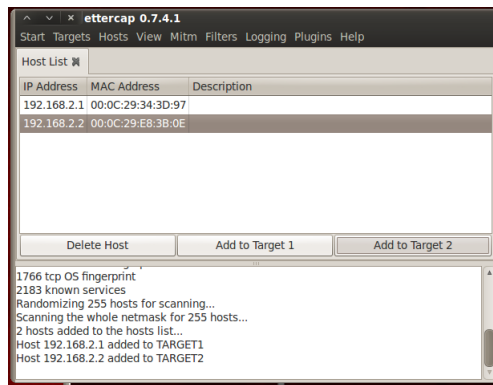


FIGURE 2 – Hôtes détectés par le scan

```
cssr@client-bsd ~1$ telnet 192.168.2.1 23
Trying 192.168.2.1...
Connected to 192.168.2.1.
Escape character is '^I'.
Password:
Last login: Mon Jan 14 08:52:29 CET 2013 from 192.168.2.2 on pts/2
Welcome to Ubuntu 12.04.1 LTS (GNU/Linux 3.2.0-32-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/

cssr@passerelle:~$
```

FIGURE 3 – Connexion à *passerelle*

Telnet : 192.168.2.1:23 -> USER: cssr PASS: [REDACTED]

FIGURE 4 – Couple en clair

4.2 Serveur SSH et optimisations

SSH (Secure SHell) joue un rôle similaire à telnet, mais contrairement à ce dernier, il intègre des mécanismes de sécurité, comme l'encryption des communications.

On installe puis démarre un serveur ssh sur la plateforme :

```
(passerelle)# apt-get install openssh-server
(passerelle)# service ssh start
```

On teste une connexion avec mot de passe (l'option *-v* est utilisé pour montrer la différence avec une connexion à base de clefs, voir plus bas)

```
(client-bsd)$ ssh -v cssr@passerelle.cs.sr
<METTRE L'OUTPUT ICI>
```

Afin d'éviter d'avoir à entrer le mot de passe à chaque connexion, on peut utiliser un système à base de clefs. On génère un couple clef privée/publique de type RSA sur le client :

```
(client-bsd)$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/cssr/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cssr/.ssh/id_rsa
Your public key has been saved in /home/cssr/.ssh/id_rsa
The key fingerprint is:
f4:3e:54:9f:df:eb:72:cc:7a:da:29:a5:79:58:0c:44 cssr@client-bsd
The key's randomart image is:
+--[ RSA 2048 ]-----+
|           .E       |
```

```

|           .           |
|           .   o       |
|           . . . o .   |
|           S o   =     |
|           o       =.   |
|           o   0 o     |
|           . *. *o     |
|           oX=         |
+-----+

```

Deux fichiers sont alors générés :

id_rsa.pub Clé publique devant être connue du serveur ;

id_rsa Clé privée ne devant pas être dévoilée.

Côté serveur, on doit maintenant ajouter la clef publique sur le serveur. SSH cherche dans le répertoire `$HOME/.ssh/` les fichiers dont le nom commence par *authorized_keys*. Ceux-ci doivent contenir une clef par ligne.

On transmet ainsi la clé publique sur le serveur, par exemple avec *scp* :

```
(client-bsd)$ scp .ssh/id_rsa.pub cssr@passerelle.cs.sr:~/.ssh/authorized_keys2
```

On prend soin d'ajuster les permissions du fichier :

```
(passerelle)# chmod 700 authorized_keys2
```

Enfin, on s'assure que l'authentification par clefs marche bien, et qu'il est difficile de récupérer les informations sur la connexion avec une attaque MITM.

La configuration du serveur SSH se fait dans le fichier `/etc/ssh/sshd_config`. Il peut-être intéressant d'y faire les réglages suivants :

X11Forwarding no pour réduire le trafic X11, et donc réduire les risques de sécurité ;

PermitRootLogin no pour désactiver un accès distant direct au compte root ;

Port 42 , où tout autre port différent du port 22 pour éviter les tentatives de connexions lancées par des bots.

4.3 Accès distant par VPN

Alors que SSH permet l'accès sécurisé à une machine distante (le serveur SSH), un VPN permet d'accéder au serveur VPN et à son réseau local de façon sécurisée.

BLABLABLA

5 Configuration d'un serveur web et d'OpenSSL

5.1 Installation d'OpenSSL

```

(passerelle)# ./configure --prefix=/usr/local && make && make install
(passerelle)# cat >> /etc/ld.so.conf
/usr/local/openssl/lib
^D
(passerelle)# ldconfig

```

5.2 Installation d'Apache2

Après avoir récupéré les sources sur le site (et vérifié le hash du fichier), on configure Apache avec le support d'OpenSSL

```

(passerelle)# ./configure --option-du-chemin-ssl=/usr/local/openssl --prefix=/usr/local/apache2
(passerelle)# /usr/local/bin/apache2/bin/apachectl start
(passerelle)# wget http://localhost -O /dev/stdout && echo OK

```

5.3 Passage à https

5.4 Tests

6 Configuration serveur (smtp+imaps) et client (gpg) email

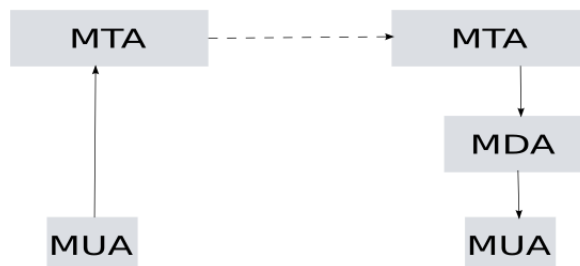


FIGURE 5 – Un MUA envoie un email à un MTA, qui le forward jusqu'à un MTA final, qui le transmet à un MDA. Enfin, le MUA du destinataire récupère l'email depuis ce MDA

La figure 5 donne un exemple simple de routage d'email, faisant intervenir 3 pièces logicielles :

MTA Mail Transfer Agent (eg. serveur SMTP), qui route les emails de domaines en domaines jusqu'à arriver à bonne destination ;

MDA Mail Delivery Agent (eg. serveur IMAP(s)), qui délivre les emails aux MUAs qui lui demandent ;

MUA Mail User Agent c'est le client email, dont le rôle principal est de récupérer les emails depuis un MDA, et d'en envoyer à un MTA ;

D'autres agents optionnels peuvent venir s'y greffer.

En pratique, on installe et configure postfix sur *passerelle*.

6.1 Envoi et transfert d'emails avec Postfix (MTA)

6.2 Récupération d'emails avec Dovecot (MDA) et Thunderbird (MUA)

6.3 Des messages chiffrés et signés avec GnuPG

6.3.1 Principe

GnuPG permet de chiffrer et signer des messages électroniques. Chaque individu génère sa propre paire de clés : une publique et une privée.

On envoie sa clé publique à ses correspondants. Ces derniers pourront à terme nous écrire en toute confidentialité car ils chiffreront le message avec notre clé publique et nous pourrons le lire grâce à notre clé privée. Cependant, nos correspondants ne font pas encore confiance à notre clé publique. Ils doivent nous contacter (par un autre moyen de communication) pour comparer l'empreinte de la clé publique qu'ils ont reçu et l'empreinte de notre clé publique. Une fois qu'ils ont confiance en la clé publique envoyée, ils la signent avec leur clé secrète.

Concernant la signature, nous appliquons notre clé privée sur le message que l'on souhaite signer et nos correspondants (qui connaissent notre clé publique) pourront vérifier que le message provient bel et bien de nous.

6.3.2 Mise en pratique

Pour simplifier l'utilisation de GnuPG, on utilise le plugin *Enigmail* pour le MUA *Thunderbird*. Et on crée une paire de clés (et un certificat de révocation) pour l'utilisateur *alice* mais aussi pour l'utilisateur *cssr*.

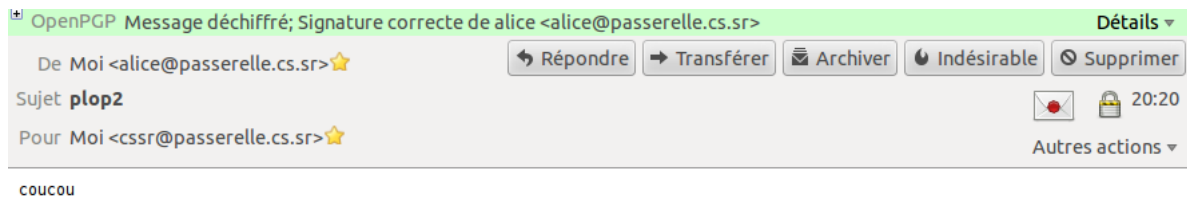


FIGURE 6 – L'utilisateur *cssr* reçoit un message d'Alice qui avait été signé et chiffré

6.4 Tests

7 Configuration VPN

7.1 Tests

8 OpenVAS & Metasploit