UNIVERSIDADE DA CORUÑA

Raúl Fernández del Blanco
3ºGEI

PRÁCTICA 3 LSI

1. Tomando como base de trabajo el SSH pruebe sus diversas utilidades:

a) Abra un shell remoto sobre SSH y analice el proceso que se realiza. Configure su fichero ssh_known_hosts para dar soporte a la clave pública del servidor.

Para obtener información sobre cómo funciona una conexión ssh tendremos que realizar el siguiente comando:

```
lsi@debian:~$ ssh -v lsi@10.11.49.116
```

1. Establecer conexión: el cliente lee los ficheros de configuración y establece conexión con el servidor. Se comprueba la compatibilidad de los protocolos SSH se mira el fichero sshconfig y se abre conexión al puerto 22.
2. Seleccionar versión a utilizar y autenticación: ambos extremos conversan para saber que algoritmo de cifrado conocen ambos y se ponen de acuerdo para utilizar uno, comprobando el cliente si tiene o no una serie de ficheros. También se selecciona la versión de ssh a utilizar (v1 o v2), en caso de v1 mejor no conectarse.
3. Negociación de parámetros de la conexión: El cliente recibe la clave pública del servidor, comprueba con esta la autenticidad de la máquina (en los ssh_known_hosts tanto de etc/ssh como del $HOME/.ssh/known_hosts) y cifra también con ella la clave de sesión. Esta clave de sesión cifrada será enviada al servidor más tarde (paso 5) ya que es el único que puede descifrarla al tener la clave privada. Se negocian los algoritmos que vamos a utilizar en la conexión, se sacan de los ficheros de configuración, cliente y servidor miran sus líneas y se ponen de acuerdo. (algoritmo de autenticación, algoritmo de intercambio de clave, algoritmo de cifrado , algoritmo de MAC para integridad y algoritmo de compresión).
4. Autenticación con el servidor: Se inicia el intercambio de la clave de sesión, preguntando si te fías de la clave en caso de ser la primera vez. (Utilizando ecdsa, algoritmo de curva elíptica).
5. Establecimiento de la clave de sesión: (utilizando la pública del servidor). Se genera una clave de sesión y se cifra con la pública del servidor y se envía. Dicha clave se utiliza en toda la sesión para cifrar las conexiones y se autentica el servidor, para tener la certeza de que no le están suplantando).
6. Autenticación del usuario: Primero este lo intenta mediante la clave pública, es decir, el servidor mira en el fichero $/home/ssh/authorized_keys (2) si está registrada la pública del cliente(puede probar con varias: dsa,rsa,ecdsa,ed25519...). En caso de no estar la clave registrada solicita contraseña(esto se puede cambiar en sshd_config, haciendo que solo se pueda entrar por clave pública para evitar ataques password guessing).
7. Termina debug: Una vez llegados a este punto la máquina ya está lista para usar.

Siguiendo con la configuración del fichero ssh_known_hosts:

Nuestra clave pública se encuentra en el fichero /etc/ssh/ssh_host_rsa_key.pub.

Esta clave se la enviaremos a nuestro compañero con el siguiente comando:

```
lsi@debian:~$ ssh-keyscan 10.11.49.48 >> /etc/ssh/ssh_known_hosts
```

Una vez hecho esto, comprobamos que todo haya ido correctamente de la siguiente forma:

1. Borramos el fichero /home/lsi/.ssh/known_hosts:

```
lsi@debian:~$ rm /home/lsi/.ssh/known_hosts
```

2. Iniciamos sesión en la otra máquina con ssh:

```
lsi@debian:~$ ssh -X lsi@10.11.49.48
```

De esta manera, ahora no va a asegurarse si realmente nuestro servidor es el que le indicamos, pues al tener su clave pública no desconfía.

b) Haga una copia remota de un fichero utilizando un algoritmo de cifrado determinado. Analice el proceso que se realiza.

Lo primero de todo es crear el fichero que enviaremos:

```
lsi@debian:~$ nano P3_1b
Esto es un ejemplo
```

Después de esto, haremos lo siguiente:

```
lsi@debian:~$ scp -v -c aes256-ctr P3_1b lsi@10.11.49.48:/home/lsi
```

Para comprobar los algoritmos de cifrado que hay en nuestra máquina, introducimos lo siguiente:

```
lsi@debian:~$ ssh -Q cipher
```

c) Configure su cliente y servidor para permitir conexiones basadas en un esquema de autenticación de usuario de clave pública

En el cliente (IMPORTANTE: en forma de usuario), cambiamos a la carpeta /.ssh y ejecutamos los siguientes comandos:

```
lsi@debian:~$ ssh-keygen -t rsa
lsi@debian:~$ ssh-keygen -t ecdsa
lsi@debian:~$ ssh-keygen -t ed25519
```

Esto, lo que hace es generar dos claves, una pública y una privada. Las frases de paso se utilizan para cifrar la clave privada. En nuestro caso, la vamos a dejar en blanco para no complicarnos la vida.

Una vez realizado esto, tenemos que pasarle nuestras claves públicas a nuestro compañero:

```
lsi@debian:~$ ssh-copy-id -i $HOME/.ssh/id_rsa lsi@10.11.49.48
lsi@debian:~$ ssh-copy-id -i $HOME/.ssh/id_ecdsa lsi@10.11.49.48
lsi@debian:~$ ssh-copy-id -i $HOME/.ssh/id_ed25519 lsi@10.11.49.48
```

Para probar que todo funciona correctamente, nos conectamos a la máquina de nuestro compañero y comprobamos que no nos pide la contraseña.

d) Mediante túneles SSH securice algún servicio no seguro.

Para crear un túnel seguro desde el puerto local hasta el puerto del otro host hacemos el siguiente comando:

```
lsi@debian:~$ ssh -P -L 10080:10.11.49.116:80 lsi@10.11.49.116
```

El otro host está escuchando al puerto local, así, cuando el puerto local reciba una conexión, esta se retransmite a través del canal seguro y la conexión se hace al hostport del host especificado. En nuestro caso, establecemos un túnel desde el puerto 10080 del cliente al puerto 80 del servidor. Para que el cliente se conecte al servidor, simplemente tendrá que conectarse a su puerto 10080. De esta forma, estamos accediendo al servidor http de forma segura mediante un túnel ssh.

Para comprobar, simplemente tenemos que ejecutar el siguiente comando:

```
lsi@debian:~$ lynx http://localhost:10080/
```

e) "Exporte" un directorio y "móntelo" de forma remota sobre un túnel SSH.

Primero, creamos un nuevo directorio:

```
lsi@debian:~$ mkdir /home/lsi/P3_1e
```

Luego, ejecutamos el siguiente comando:

```
lsi@debian:~$ sshfc lsi@10.11.49.48 :/home/lsi/ /home/lsi/P3_1e
```

De esta forma, si creamos algo en ese nuevo directorio, le aparecerá a nuestro compañero en su directorio /home/lsi.

Podemos desvincular los directorios con el siguiente comando:

```
lsi@debian:~$ fusermont -u /home/lsi/P3_1e
```

2. Tomando como base de trabajo el servidor Apache2

a) Configure una Autoridad Certificadora en su equipo.

Lo primero que tendremos que hacer es instalar el material necesario para crear nuestra AC:

```
lsi@debian:~$ apt install openssl
```

Una vez instalado, tendremos que dirigirnos al directorio /usr/lib/ssl/misc, donde realizaremos los siguientes comandos:

```
lsi@debian:~$ cd /usr/lib/ssl/misc
lsi@debian:~$ ./CA.pl -newca
Enter PEM pass phrase: lsi116
Verifying - Enter PEM pass phrase: lsi116
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:A Coruña
Locality Name (eg, city) []:Coruña
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UDC
Organizational Unit Name (eg, section) []:FIC
Common Name (e.g. server FQDN or YOUR name) []:web.lsi.com
Email Address []:web@lsi.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:lsi116
An optional company name []:UDC_GAL
```

Ya somos una autoridad certificadora, por lo que se nos han generado dos ficheros:

- /usr/lib/ssl/misc/demoCA/private/cakey.pem: este fichero contiene la clave privada de la autoridad certificadora, cifrada con la clave de paso introducida.
- /usr/lib/ssl/misc/demoCA/cacert.pem: este fichero contiene un certificado con la clave pública de la autoridad certificadora autofirmado.

b) Cree su propio certificado para ser firmado por la Autoridad Certificadora. Bueno, y fírmelo.

Lo primero que haremos será ejecutar el siguiente comando:

```
lsi@debian:~$ ./CA.pl -newreq-nodes
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:A Coruña
Locality Name (eg, city) []:Coruña
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UDC
Organizational Unit Name (eg, section) []:FIC
Common Name (e.g. server FQDN or YOUR name) []:web.lsi.com
Email Address []:web@lsi.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:lsi116
An optional company name []:UDC_GAL
```

De esta forma, se nos generan otros dos ficheros:

- /usr/lib/ssl/misc/newreq.pem: este fichero contiene el certificado y la clave pública.
- /usr/lib/ssl/misc/newkey.pem: este fichero contiene la clave privada.

Una vez hecho esto, tenemos que cifrar el certificado con mi clave privada:

```
lsi@debian:~$ ./CA.pl -sign
```

De esta forma, se nos generará un fichero /usr/lib/ssl/misc/newcert.pem, que contiene el certificado firmado por la autoridad certificadora.

c) Configure su Apache para que únicamente proporcione acceso a un determinado directorio del árbol web bajo la condición del uso de SSL. Considere que si su la clave privada está cifrada en el proceso de arranque su máquina le solicitará la correspondiente frase de paso, pudiendo dejarla inalcanzable para su sesión ssh de trabajo.

Primero, a nuestro compañero le tendremos que pasar el certificado firmado (newcert.pem) y su clave privada (newkey.pem).

```
lsi@debian:~$ scp -c aes128-ctr /usr/lib/ssl/misc/newcert.pem
lsi@10.11.49.48:/home/lsi/certs_to_copy/newcert.pem
lsi@debian:~$ scp -c aes128-ctr /usr/lib/ssl/misc/newkey.pem
lsi@10.11.49.48:/home/lsi/certs_to_copy/newkey.pem
```

Pasos a seguir por mi:

```
lsi@debian:~$ cp /usr/lib/ssl/misc/demoCA/cacert.pem
/usr/local/share/ca-certificates/cacert.crt
lsi@debian:~$ update-ca-certificates
```

Lo siguiente que haré, será buscar la página de mi compañero cuando él haya acabado.

Pasos a seguir por mi compañero:

Lo primero que hará será habilitar ssl para apache:

```
lsi@debian:~$ a2enmod ssl
lsi@debian:~$ a2ensite default-ssl
lsi@debian:~$ systemctl restart apache2
```

Ahora, tendremos que modificar el fichero de configuración de ssl para que utilice el certificado firmado por la AC que nos ha pasado nuestro compañero y su clave privada.

```
lsi@debian:~$ nano /etc/apache2/sites-enabled/default-ssl.conf

ServerAdmin webmaster@localhost
ServerName webraul
DocumentRoot /var/www/html

SSLCertificateFile      /home/lsi/certs_to_copy/newcert.pem
SSLCertificateKeyFile   /home/lsi/certs_to_copy/newkey.pem
```

INFORMACIÓN NECESARIA:

Hay que modificar los siguientes ficheros:

```
lsi@debian:~$ nano /etc/apache2/apache2.conf
ServerName webraul

//Mi máquina es la 116
lsi@debian:~$ nano /etc/hosts
10.11.49.116    webraul
10.11.49.48     web
```

Para asegurarnos que todo va bien, después de que nuestro compañero nos haya mandado el certificado y la clave privada hacemos lo siguiente:

```
lsi@debian:~$ openssl rsa -in newkey.pem -noout -modulus
lsi@debian:~$ openssl x509 -in newcert.pem -noout -modulus
```

En caso de que sean iguales, todo irá bien.

Información teórica:

Al conectarnos, el cliente nos va a enviar el certificado que le hicimos previamente, el navegador buscará en el repositorio web de autoridades certificadoras, buscará la autoridad correspondiente para obtener su pública y descifrar el certificado.

3. **Tomando como base de trabajo el openVPN deberá configurar una VPN entre dos equipos virtuales del laboratorio que garanticen la confidencialidad entre sus comunicaciones.**

Primero, debemos tener descargado lo siguiente:

```
lsi@debian:~$ apt install openssl
lsi@debian:~$ apt install openvpn
```

Para ver si el módulo tun está cargado ejecutamos lo siguiente:

```
lsi@debian:~$ lsmod | grep tun
//Si no está cargado ejecutamos:
lsi@debian:~$ modprobe tun
lsi@debian:~$ echo tun >> /etc/modules
```

Ahora generaremos una clave simétrica privada para cifrar y descifrar la paquetería VPN.

Ahora, tenemos que hacer la configuración, por lo que hacemos lo siguiente:

```
lsi@debian:~$ cd /etc/openvpn
lsi@debian:~$ openvpn –genkey secret clave.key
lsi@debian:~$ nano tunel.conf
local 10.11.49.116
remote 10.11.49.48
dev tun1
port 5555
comp-lzo
user nobody
ping 15
ifconfig 172.160.0.1 172.160.0.2
secret /etc/openvpn/clave.key
lsi@debian:~$ reboot
```

Por último, le pasamos el fichero a nuestro compañero:

```
lsi@debian:~$ scp clave.key lsi@10.11.49.48:/etc/openvpn
```

Ahora, para el cliente:

```
lsi@debian:~$ cd /etc/openvpn
lsi@debian:~$ nano tunel.conf
local 10.11.49.48
remote 10.11.49.116
dev tun1
port 5555
comp-lzo
user nobody
ping 15
ifconfig 172.160.0.2 172.160.0.1
secret /etc/openvpn/clave.key
lsi@debian:~$ reboot
```

Después de esto, tendremos que activar el servicio openvpn y ejecutar un comando para configurar el túnel:

```
lsi@debian:~$ systemctl enable openvpn
lsi@debian:~$ systemctl start openvpn
lsi@debian:~$ openvpn –config /etc/openvpn/túnel.conf
```

Sabemos que openvpn funciona correctamente si hacemos un ifconfig tun1 y existe esa interfaz de red. Además, como comprobación se puede hacer un ping hacia la dirección 172.160.0.X y comprobaremos que esta funciona.

NOTA: En el caso del cliente, si ejecutamos el último comando, probablemente nos salga un error “IOCTL busy resource” o algo parecido. Aunque nos salga este error, el servicio de la vpn ya estará levantado.

7. Instale el SIEM splunk en su máquina. Sobre dicha plataforma haga los siguientes puntos.:

Antes de nada, tenemos que descargar splunk en nuestra máquina. Lo primero que haremos será introducir el fichero de descarga en nuestro equipo y luego simplemente tendremos que realizar los siguientes comandos:

```
lsi@debian:~$ apt install <paquete>.deb
lsi@debian:~$ /opt/splunk/bin/splunk enable boot-start
lsi@debian:~$ systemctl start splunk
```

De esta forma, ya tendremos splunk corriendo en nuestra máquina, por lo que ya podemos comenzar a hacer el ejercicio. Para utilizar la herramienta, simplemente tendremos que entrar en un navegador de nuestro ordenador y buscar: http://10.11.49.116:8000.

a) Genere una query que visualice los logs internos del splunk.

Para este apartado usaremos la app: Search & Reporting. En la barra de búsqueda introduciremos: index="_internal". La primera vez que lo hagamos nos saltará un error de memoria, por lo que (en nuestra máquina debian) haremos lo siguiente:

```
lsi@debian:~$ nano /opt/splunk/etc/system/default/server.conf
//Modificaremos lo siguiente:
minFreeSpace = 500

lsi@debian:~$ systemctl restart splunk
```

Ahora, si volvemos al buscador de splunk e introducimos lo mismo que antes, tendremos los logs internos del splunk que se encuentran en /opt/splunk/var/log/splunk.

b) Cargue el fichero /var/log/apache2/access.log y el journald del sistema y visualícelos.

En este caso iremos a Add Data. Una vez ahí, seleccionamos el método de Monitor para obtener los archivos de nuestra máquina virtual. Ahora, escogemos la opción Files & Directories y seleccionamos los archivos: /var/log/apache2/acces.log y /var/log/syslog. Le damos a next todo el rato. Ahora, podremos hacer búsquedas simplemente introduciendo el sourcetype, por ejemplo: “syslog” para los logs del sistema y “access_apache” para obtener información de los accesos a nuestro servidor apache.

- source="/var/log/syslog" host="debian" sourcetype="syslog"

c) Obtenga las IPs de los equipos que se han conectado a su servidor web (pruebe a generar algún tipo de gráfico de visualización), así como las IPs que se han conectado un determinado día de un determinado mes.

Si no lo tenemos, necesitaremos usar un campo llamado clientip. Para ello, tendremos que ir a: Extract New Fields y hacerlo.

Para obtener las IPs de los equipos que se han conectado a nuestro servidor web, tenemos que introducir la siguiente consulta:

- source="/var/log/apache2/access.log" host="debian" sourcetype="access_log" date_month="december" date_mday="8"
- source="/var/log/apache2/access.log" host="debian" sourcetype="access_log" date_month="december" date_mday="8" "10.11.49.48"
- source="/var/log/apache2/access.log" host="debian" sourcetype="access_log" | top clientip

d) Trate de obtener el país y región origen de las IPs que se han conectado a su servidor web y si posible sus coordenadas geográficas.

- source="/var/log/apache2/access.log" date_month="december" date_mday="8" | iplocation clientip |tail 20 | table clientip, status, City, Country | stats count by Country
- source="/var/log/apache2/access.log" clientip=* date_month="december" date_mday="8" | iplocation clientip | table clientip,lat,lon,Country,City | geostats latfield=lat longfield=lon count by Country

e) Obtenga los hosts origen, sources y sourcestypes.

Mirar las columnas de splunk.

f) ¿cómo podría hacer que splunk haga de servidor de log de su cliente?