Práctica 3 lsi - Ejercicios P3 lsi 2019-2020

Lexislación e Seguridade Informática (Universidade da Coruña)

**********EJERCICIO 1)**********

a) ssh -v lsi1@10.10.102.54

//ESTABLECER CONEXIÓN. Cliente lee los ficheros de configuracion y establece conexion con el servidor. Se comprueba la versión del protocolo SSH que utilizan ambos para compatibilidad. Cada parte mira su fichero de sshconfig y se abre conexion al puerto 22

OpenSSH_7.4p1 Debian-10+deb9u7, OpenSSL 1.0.2t 10 Sep 2019

debug1: Reading configuration data /etc/ssh/ssh_config

debug1: /etc/ssh/ssh_config line 19: Applying options for *

debug1: Connecting to 10.10.102.54 [10.10.102.54] port 22.

debug1: Connection established.

//SELECCIONAR VERSIÓN A UTILIZAR Y AUTENTICACION. 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 seleccion na versión de ssh a utilizar(v1 o v2), en caso de v1 mejor no conectarse.

debug1: permanently_set_uid: 0/0

debug1: key_load_public: No such file or directory

debug1: identity file /root/.ssh/id_rsa type -1

debug1: key_load_public: No such file or directory

debug1: identity file /root/.ssh/id_rsa-cert type -1

debug1: key_load_public: No such file or directory

debug1: identity file /root/.ssh/id_dsa type -1

debug1: key_load_public: No such file or directory

debug1: identity file /root/.ssh/id_dsa-cert type -1

debug1: key_load_public: No such file or directory

debug1: identity file /root/.ssh/id_ecdsa type -1

debug1: key_load_public: No such file or directory

debug1: identity file /root/.ssh/id_ecdsa-cert type -1

debug1: key_load_public: No such file or directory

debug1: identity file /root/.ssh/id_ed25519 type -1

debug1: key_load_public: No such file or directory

debug1: identity file /root/.ssh/id_ed25519-cert type -1

debug1: Enabling compatibility mode for protocol 2.0

debug1: Local version string SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u7

debug1: Remote protocol version 2.0, remote software version OpenSSH_7.4p1 Debian-10+deb9u7

debug1: match: OpenSSH_7.4p1 Debian-10+deb9u7 pat OpenSSH* compat 0x04000000

debug1: Authenticating to 10.10.102.54:22 as 'lsi'

//NEGOCIACION DE PARAMETROS DE LA CONEXION(Ofrecidos por el cliente, ya que normalmente este es el que propone). El cliente recibe la c.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 tambien 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 configuracion, cliente y servidor miran sus lineas y se ponen de acuerdo.

(algoritmo de autentificacion, algoritmo de intecambio de clave, algoritmo de cifrado , arlgoritmo de MAC para integridad y algoritmo de compresion)

debug1: SSH2_MSG_KEXINIT sent

debug1: SSH2_MSG_KEXINIT received

debug1: kex: algorithm: curve25519-sha256

debug1: kex: host key algorithm: ecdsa-sha2-nistp256

debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none

debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none

//AUTENTICACION DEL 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)

debug1: expecting SSH2_MSG_KEX_ECDH_REPLY

debug1: Server host key: ecdsa-sha2-nistp256 SHA256:9Vdaetzi2SjGlJppDJ+VQh6/exj55jTf138HQIXJBx4

debug1: Host '10.10.102.54' is known and matches the ECDSA host key.

debug1: Found key in /root/.ssh/known_hosts:1

debug1: rekey after 134217728 blocks

//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 clase 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)

debug1: SSH2_MSG_NEWKEYS sent

debug1: expecting SSH2_MSG_NEWKEYS

debug1: SSH2_MSG_NEWKEYS received

debug1: rekey after 134217728 blocks

debug1: SSH2_MSG_EXT_INFO received

debug1: kex_input_ext_info: server-sig-algs=<ssh-ed25519,ssh-rsa,ssh-dss,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521>

debug1: SSH2_MSG_SERVICE_ACCEPT received

debug1: Authentications that can continue: publickey,password

debug1: Next authentication method: publickey

debug1: Trying private key: /root/.ssh/id_rsa

debug1: Trying private key: /root/.ssh/id_dsa

debug1: Trying private key: /root/.ssh/id_ecdsa

debug1: Trying private key: /root/.ssh/id_ed25519

//AUTENTICACIÓN DE 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).

debug1: Next authentication method: password

lsi@10.10.102.54's password:

debug1: Authentication succeeded (password).

Authenticated to 10.10.102.54 ([10.10.102.54]:22).

debug1: channel 0: new [client-session]

debug1: Requesting no-more-sessions@openssh.com

debug1: Entering interactive session.

debug1: pledge: network

debug1: client_input_global_request: rtype hostkeys-00@openssh.com want_reply 0

debug1: Sending environment.

debug1: Sending env LANG = es_ES.UTF-8

//TERMINA DEBUG

Linux debian 4.9.0-11-amd64 #1 SMP Debian 4.9.189-3+deb9u2 (2019-11-11) x86_64

The programs included with the Debian GNU/Linux system are free software;

the exact distribution terms for each program are described in the

individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent

permitted by applicable law.

Configurar el fichero ssh_known_hosts (el cual por defecto no existe sino que hay que crearlo) para dar soporte a la clave pública del servidor.

En el archivo /etc/ssh/ssh_known_hosts guardamos las claves públicas a nivel de máquina (/home/lsi/.ssh/known_hosts las guarda a nivel de usuario, siendo lsi el usuario en este caso).

El fichero contiene 3 valores: el hash, el tipo de cifrado y la clave en sí

*>ssh-keyscan 10.10.102.43 >> /etc/ssh/ssh_known_hosts (Este comando devuelve las claves públicas del host indicado(cliente))

/etc/ssh/ssh_config (configuración cliente)

/etc/ssh/sshd_config (configuración servidor) -- X11forwarding debe estar a yes

De esta forma, utilizamos estas claves públicas para el intercambio de la clave de sesión, en lugar de que nos la remita. Así evitamos inyecciones de clave.

Comprobar borrando el contenido del $HOME/.ssh/known_hosts del usuario y conectado.

Lanzar una sesión X sobre una conexión ssh

*>ssh -X lsi@10.10.102.servidor-ssh (estando como lsi)

b)

-----Compañero1

Creamos un archivo para cifrar, en nuestro caso se llamará contraseñas.txt y lo editamos poniéndole el texto que queramos.

Cifrar un archivo:

>openssl aes-256-cbc -e -in contraseñas.txt -out contraseñasCifradas.txt.

Esto pedirá una clave con la que cifrar el archivo contraseñas.txt y generará un nuevo archivo contraseñasCifradas.txt completamente cifrado y seguro que podremos enviarnos a algún lugar donde recuperarlo en algún momento dado.

Está clave debe de saberla el compañero 2 para poder descifrar el archivo.

>openssl ciphers (Para ver todos los ciphers que tenemos disponibles con el comando openssl, de tal forma que, en vez de poner el cipher aes-256-cbc podemos poner cualquier otro)

(Lo que está puesto justo debajo es un comando entero)

>scp -c aes128-ctr contraseñasCifradas.txt lsi@10.10.102.220:/home/lsi/contraseñasCifradasCopia.txt (Se envía el fichero al compañero)

>ssh -Q cipher (Para ver todos los ciphers que tenemos disponibles con el comando scp, de tal forma que, en vez de poner el cipher aes-256-cbc podemos poner cualquier otro)

-----Compañero2

>cat contraseñasCifradasCopia.txt (Podemos ver como el archivo que nos envió el compañero 1 está completamente cifrado)

>openssl aes-256-cbc -d -in contraseñasCifradasCopia.txt -out contraseñasDescifradas.txt (Para descifrar el archivo que envió el compañero 1. Pedirá una clave para descifrar qué es la misma clave que puso el compañero 1)

>cat contraseñasDescifradas.txt (Podemos ver como el archivo descifrado es el mismo que envió el compañero 1)

c)

MAU CLIENTE _____ AD SERVIDOR

CLIENTE

En el fichero de configuración del cliente /etc/ssh/ssh_config, la variable RSAAuthentication debe estar a yes.

>ssh-keygen -t rsa (Dejar la frase de paso en blanco cuando nos la pida)

ssh-keygen genera, administra y convierte claves de autenticación para ssh. Puede crear claves para que las use el protocolo SSH.

*-t: para especificar el tipo de clave que queremos crear, usamos rsa.

Frase de paso: se puede dejar en blanco para que no nos la pida. Esta frase se utiliza para cifrar la clave privada del usuario. Se puede utilizar un agente para que no nos la pida. Se utiliza para cifrar la clave privada pero nos complica la configuración. Las claves privadas se securizan de dos formas:

- Restringiendo los permisos del fichero en el que se almacenan: de forma que solo el propietario y el root puedan leer las claves privadas.

- Cifrándolas: a la frase de paso se le aplica una función hash cuyo resultado (huella) sea de una longitud acorde al algoritmo que se utiliza para cifrarla.

Esto creará en $HOME/.ssh los ficheros id_rsa e id_rsa.pub.

Se puede hacer de igual forma para otros asimétricos como dsa, ecdsa o ed25519

Copiamos la clave pública al servidor (suponiendo que la ip 220 es el servidor):

>scp id_rsa.pub lsi@10.10.102.220:.ssh/id_rsa.pub

SERVIDOR

En el fichero de configuración del servidor /etc/ssh/sshd_config, la variable PubkeyAuthentication debe estar a yes.

>cat .ssh/id_rsa.pub >> /home/lsi/.ssh/authorized_keys (lo metemos en el fichero authorized_keys, Metemos su contenido en el fichero especificado en el archivo de configuración (/etc/ssh/authorized_keys), suele ser authorized_keys o authorized_keys2

El servidor ya puede borrar la clave pública ya que no le hace falta porque en el paso anterior ya la metió en el fichero authorized_keys:

>rm id_rsa.pub (La borramos del servidor, ya que no nos hace falta)

d)

-L --> [bind_address:]port:host:hostport

Permite crear un tunel seguro desde el puerto local hasta el puerto del otro host, de forma que el otro host está escuchando al puerto local, de forma que cuando una conexion se hace al puerto local, la conexion se retransmite a través del canal seguro y la conexion se hace al hostport del host especificado.

Realiza todo cliente, servidor solo ten funcionando o apache.

>ssh -L 10080:10.10.102.54:80 lsi@10.10.102.54

Con este comando establecemos un túnel desde el puerto 10080 del cliente al puerto 80 del servidor, de tal forma que para 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)

w3m http://localhost:10080/

e)

Para esto utilizaremos sshfs.

Primero crearemos el directorio en nuestro ordenador local donde montaremos el directorio remoto, por ejemplo en /mnt/lsi/

Despues de haber creado ese directorio, para montarlo simplemente ejecutamos el comando

#sudo sshfs lsi@10.10.102.242:/home/lsi /mnt/lsi

Ahi se montara el directorio en nuestro ordenador local, y cualquier cambio que hagamos en el, se vera reflejado en el servidor donde

se encuentra alojado ese sistema de archivos.

Para desmontarlo, lo desmontamos como hariamos con cualquier otro sistema de archivos utilizando umount.

f)

-Deshabilitar todos los servicios menos el ssh, tanto tcp, como udp. Deshabilitar ipv6.

-Deshabilitar icmp, portmap, etc.

-Editar el fichero /etc/sysctl.conf y añadir:

net.ipv4.icm_echo_ignore_all=1

>sysctl -p

-En servidor sshd en /etc/ssh/sshd_config poner:

PermitRootLogin no

-Limitar el acceso a un determinado usuario. Al final del /etc/ssh/sshd_config ponemos:

AllowUsers lsi-ssh

Si es necesario, configurar para que lsi pueda hacer >su

Se puede optar por dejar al usuario lsi-ssh sin password, bloqueado, para que únicamente se pueda autenticar por clave pública.

Se pueden permitir y denegar usuarios, así como grupos de usuarios.

-Cambiar el puerto de escucha a uno alto.

-Modificar el sistema ante fingerprinting. Modificar ssh ante fingerprinting. Validar con nmap, xprobe, etc.

Por ejemplo, poner en el fichero /etc/ssh/sshd_config la línea:

Banner /etc/issue.net

Y poner en el banner otro servicio diferente o mejor dejarlo vacío.

-Configurar iptables y wrapper.

Permitir conexiones entrantes al puerto en el que está nuestro ssh a todos o a las ips permitidas. O por ejemplo permitir únicamente direcciones de España.

Denegar todo lo demás.

-Configurar snort y ossec

-Configurar port knocking

-Correr lynis para chequear defectos de configuración:

>lynis -c (Chequea todo el sistema)

Ver ratio de hardening del sistema y forma de incrementarlo. A todos los niveles.

**********EJERCICIO 2)**********

Para crear la Autoridad Certificadora vamos a la ruta /usr/lib/ssl/misc y ejecutamos el siguiente comando

>./CA.pl -newca

*-newca: Crea el certificado de la Autoridad Certificadora que contiene la clave pública, y también crea la clave privada. Va a pedir frase de paso, la cual hay que meter ya que con ella se cifra la clave privada usando un algoritmo simétrico.

Después de haber hecho esto, se habrá creado en la misma carpeta donde está el script CA.pl la carpeta demoCA que contendrá dentro (entre otros archivos) el fichero cacert.pem y la carpeta private, que tendrá a su vez dentro el fichero cakey.pem.

En la misma ruta donde creamos la Autoridad Certificadora (/usr/lib/ssl/misc) hacemos lo siguiente:

Creamos nuestro propio certificado:

>./CA.pl -newreq-nodes

*-newreq-nodes: para generar la clave privada y el certificado con la clave pública para una entidad (que será nuestro compañero) y además para que no nos pida una frase de paso, lo cual no nos interesa en este paso, ya que la idea en este paso es no meterla. Además los campos “Organization Name” y “State or Province Name” deben ser iguales en el certificado de la Autoridad Certificadora y en los certificados emitidos. Es decir se debe meter los mismos campos “Organization Name” y “State or Province Name” tanto al hacer el comando >./CA.pl -newca en el paso anterior, como al hacer >./CA.pl -newreq-nodes en este paso.

Después de haber hecho esto, habremos creado los archivos newkey.pem y newreq.pem en la misma carpeta donde esta el script CA.pl y la carpeta demoCA.

Firmamos el certificado:

>./CA.pl -sign

*-sign: firma el certificado. Pedirá la frase de paso de la AC, la cual hay que introducir. La razón por la que pide la frase de paso es porque necesita usar la clave privada de la Autoridad Certificadora.

Una vez hecho esto, nos habremos creado el archivo newcert.pem en la misma carpeta donde esta el script CA.pl, la carpeta demoCA, y los archivos newkey.pem y newreq.pem.

Información de los ficheros que acabamos de crear ficheros:

cacert.pem: tiene el certificado con la clave pública de la Autoridad Certificadora.

private/cakey.pem: tiene la clave privada de la Autoridad Certificadora.

newcert.pem: tiene el certificado generado y firmado.

newkey.pem: tiene la clave privada del certificado.

newreq.pem: tiene la petición de certificado para firmar.

Los ficheros cacert.pem y private/cakey.pem contienen datos de la Autoridad Certificadora.

Los ficheros newcert.pem y newkey.pem son los que se enviarán al servidor apache.

A la Autoridad Certificadora sólo le queda mandar el newcert.pem y el newkey.pem al servidor Apache y enviar su propio certificado al cliente, es decir, enviar el cacert.pem, para que lo inserte en su navegador.

Se envían los ficheros newcert.pem y newkey.pem al servidor apache, es decir, en vez de usar la ip 220 se usa la ip del compañero que tenga el servidor apache

>chmod +r newkey.pem (el newkey.pem necesita permisos de lectura)

>scp newkey.pem lsi@10.10.102.220:/home/lsi/newkey.pem

>scp newcert.pem lsi@10.10.102.220:/home/lsi/newcert.pem

>scp demoCA/cacert.pem lsi@10.10.102.221:/home/lsi/cacert.pem

----------SERVIDOR APACHE

En /etc/apache2/sites-available hay dos ficheros: 000-default.conf y default-ssl.conf

Denegamos el acceso no seguro al directorio webssl, es decir, denegamos el acceso por HTTP, es decir, para que no se pueda acceder por el 80, añadiendo justo antes de </VirtualHost> en el archivo /etc/apache2/sites-available/000-default.conf:

<Directory “/var/www/webssl”>

Options FollowSymLinks

AllowOverride None

Order deny,allow

Deny from all

</Directory>

Creamos un servidor virtual. En /etc/apache2/sites-available/default-ssl.conf hay mucha información sobre las opciones de configuración disponibles, editamos el archivo de la siguiente forma:

```
<IfModule mod_ssl.c>
	<VirtualHost *:443>

	# Información (que coincida con el certificado)
	ServerAdmin lsi@udc.es
	ServerName 10.10.102.30
	DocumentRoot /var/www/

	# Opciones para los mensajes de log generados
	LogLevel warn
	ErrorLog ${APACHE_LOG_DIR}/error.log
	CustomLog ${APACHE_LOG_DIR}/access.log combined

	# Enable/Disable SSL for this virtual host.
	SSLEngine on

	# Le indicamos el certificado y la clave que generamos
	SSLCertificateFile	/etc/ssl/certs/Mi_Certificado_Digital.crt
	SSLCertificateKeyFile	/etc/ssl/private/Mi_Clave_Privada.key

	# Que no se requiera certificado a los clientes
	SSLVerifyClient none

	# Para los ficheros del directorio 'privado' requiere autenticación
	<Directory "/var/www/privado">
		AuthName "Ficheros privados"
		AuthType Basic
		AuthUserFile /etc/apache2/.htpasswd
		Require valid-user
		SSLRequireSSL
	 </Directory>
```

</VirtualHost>

</IfModule>

mv newkey y newcert a rutas correspondientes***

Con la primera línea del fichero /etc/apache2/sites-available/default-ssl.conf ya sabemos que tenemos que habilitar el módulo 'ssl'. Con la segunda ya sabemos que el puerto será el 443.

>a2enmod ssl (carga el módulo de ssl)

>a2ensite default-ssl (Habilitamos el site. Crea un link simbólico de sites-available a sites-enabled)

>systemctl restart apache2 (Reiniciamos el servicio para hacer efectivos todos los cambios)

----------COMPROBAR

>openssl verify /etc/ssl/certs/Mi_Certificado_Digital.crt (de esta forma openssl es capaz de encontrar el certificado raíz y con él comprueba la autenticidad del certificado emitido)

w3m

Abrir sesion ssh -X e abrir firefox, visitamos https://lsi ou 10.10.102.174, logo añadir /privado e vemos que nos pide a contraseña

**********EJERCICIO 3)**********

OpenVPN es una aplicación que implementa la Red Privada Virtual (VPN) para la creación de conexiones seguras punto a punto o sitio a sitio

https://community.openvpn.net/openvpn/wiki/HOWTO OPCIONES

https://openvpn.net/community-resources/reference-manual-for-openvpn-2-4/

https://serverfault.com/questions/392387/multiple-openvpn-instances-tun1-fails HECHO

apt-get install openvpn

apt-get install openssl

A SERVIDOR(0.1)

M CLIENTE(0.2)

*UNO DE LOS EXTREMOS GENERA LA CLAVE**

cd /etc/openvpn

openvpn --genkey --secret clave.key

nano tunel.conf

---local 10.10.102.174

remote 10.10.102.54

dev tun1 ---->Interfaz

port 5555 -----> Puerto

comp-lzo ---> Hace que solo se permita el algoritmo de compresión LZO

user nobody --->Reducir privilegios, medida de seguridad

ping 15 -----> Ping remote over the TCP/UDP control channel if no packets have been sent for at least n seconds

ifconfig 172.160.0.1 172.160.0.2

secret /etc/openvpn/clave.key ---->Indica donde está a clave (OpenVPN's secure modes)

ENVIAMOS A CLAVE A COMPAÑEIRO

scp clave.key lsi@10.10.102.54:~

systemctl restart

ifconfig -a

CLIENTE

mv /home/lsi/clave.key /etc/openvpn/clave.key

nano tunel.conf(Igual que anterior invertidno ip)

openvpn --verb 5 --config /etc/openvpn/tunel.conf

Comprobamos facendo pings aos extremos

ping 172.160.0.2

**********EJERCICIO 4)**********

Vamos a /etc

Generamos la clave

ntp-keygen -M (Generate a new symmetric keys file containing 10 MD5 keys, and if OpenSSL is available, 10 SHA keys)

Eliminamos enlace

rm ntp.keys

Copiamos archivo al que apuntaba a ntp.key

cp ntpkey_MD5key_debian..... ntp.keys

Eliminamos el archivo primeramente creado

rm ntpkey_MD5key....

Cambiamos permisos ntp.keys

chmod -R 640 ntp.keys

Poner como propietario del archivo a ntp

chown ntp ntp.keys

Poner como grupo root

chgrp root ntp.keys

Modificar el ntp.conf poniendo:

keys /etc/ntp.keys (indica donde tenemos la lista de claves)

trustedkey 1 (indica las claves que van a poder usarse, para que si nos roban por ejemplo la 7 pero no la tenemos aquí metida no funcione)

Pasamos por scp el ntp.keys al cliente

scp ntp.keys lsi@10.10.102.54:/home/lsi/ntp.keys

ntpq

-as

**CLLIENTE

Modifica ntp.conf poniendo que clave usará

server x.x.x.x key 1 (escoller a clave, non ten porque ser a 1)

....

trustedkey 1

keys /etc/ntp.key

Ponerle al ntp.keys los mismos permisos y propietarios que en el servidor.

**********EJERCICIO 6)**********

INSTALACION

nano /etc/apt/sources.list

deb https://http.kali.org/kali kali-rolling main non-free contrib

deb-src https://http.kali.org/kali kali-rolling main non-free contrib

apt update

apt upgrade

apt dist-upgrade

apt get install openvas

openvas-setup

Esto te saca una contraseña en la ultim linea. Como es imposible recordarla:

openvasmd --user=admin --new-password=contraseña

openvas-start

Username:admin

Password:contraseña

SCANEO

creo un nueva target como lsi

Desactivamos el firewall tanto privado como publico

y un task "ejemplo" con escaneo muy profundo

scans -> reports

una vez realizado el escaneo me encuentro una vulnerabilidad de rango medio, veo detalles

se trata de un exploit muy conocido de sistemas Windows, por culpa de tenerlo

un atacante puede obtener mas informacion sobre el host remoto

RPC es un programa que utiliza una computadora para ejecutar código en otra máquina remota

sin tener que preocuparse por las comunicaciones entre ambas.

DCE/RPC es un sistema de llamada a procedimiento remoto del conjunto de software,

es el sistema de llamada a procedimiento remoto desarrollado para el entorno de la informática distribuida.

Este sistema permite a los programadores escribir software distribuido como si fueran

todos los que trabajan en el mismo equipo, sin tener que preocuparse por el código de red subyacente

MSRPC es una version modificada de DCE/RPC

**********EJERCICIO 7)**********

https://www.redeszone.net/gnu-linux/iptables-configuracion-del-firewall-en-linux-con-iptables/

sh firewall.sh start

sh firewall.sh stop

iptables -L -v -n

---herramienta avanzada de filtrado de paquetes en Linux

---analizar cada uno de los paquetes del tráfico de red entra en una máquina y decidir, en función de un conjunto de reglas, qué hacer con ese paquete

-A –append → agrega una regla a una cadena.

-P –policy → explica al kernel qué hacer con los paquetes que no coincidan con ninguna regla.

CONDICIONES

-p –protocol → la regla se aplica a un protocolo.

-s –src –source → la regla se aplica a una IP de origen.

-d –dst –destination → la regla se aplica a una Ip de destino.

-i –in-interface → la regla de aplica a una interfaz de origen, como eth0.

-o –out-interface → la regla se aplica a una interfaz de destino.

-j Para aplicar una política

COND. TCP/UDP

-sport –source-port → selecciona o excluye puertos de un determinado puerto de origen.

-dport –destination-port → selecciona o excluye puertos de un determinado puerto de destino.

```
#! /bin/bash

### BEGIN INIT INFO
# Provides:     fw
# Required-Start:   $all
# Required-Stop:    $all
# Default-Start:    2 3 4 5
# Default-Stop:     0 1 6
# Short-Description:    Carga Reglas de IPTABLES
# Description:      firewall de estado
### END INIT INFO

#Variables
MI_IP_0="10.10.102.221"
MI_IP_1="10.10.150.221"
IP_COMP_0="10.10.102.220"
IP_COMP_1="10.10.150.220"
RANGO_WIFI="10.20.32.0/21"
RANGO_VPN="10.30.8.0/21"
IP_DNS="10.10.102.27"
TIME="120"

start() {
   echo "CONFIGURANDO FIREWALL DE ESTADO"

   # Limpiar reglas
```

```
clear


# Politica dura
echo "[+] Definiendo politica dura..."
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP


# Reglas de entrada
echo "[+] Configurando cadenas de ENTRADA..."
iptables -A INPUT -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT # http hacia mi servidor
iptables -A INPUT -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT # http desde fuera
iptables -A INPUT -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT # http ssl hacia mi server
iptables -A INPUT -p tcp --sport 443 -m state --state ESTABLISHED -j ACCEPT # http ssl hacia afuera
iptables -A INPUT -i eth1 -s $IP_COMP_1 -d $MI_IP_1 -p udp --dport 123 -j ACCEPT # ntp (cliente)
iptables -A INPUT -i eth0 -s $IP_COMP_0 -d $MI_IP_0 -p tcp --sport 514 -j ACCEPT # rsyslog (cliente)
iptables -A INPUT -i eth0 -s $IP_COMP_0 -d $MI_IP_0 -p tcp --dport 22 -j ACCEPT # ssh yo como host
iptables -A INPUT -i eth0 -s $IP_COMP_0 -d $MI_IP_0 -p tcp --sport 22 -j ACCEPT # ssh contestaciones de su servidor
iptables -A INPUT -i eth0 -p tcp -s $RANGO_WIFI --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT # wifi
iptables -A INPUT -i eth0 -p tcp -s $RANGO_VPN --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT # vpn
iptables -A INPUT -i lo -j ACCEPT # loopback
iptables -A INPUT -p udp -i eth0 -s $IP_DNS --sport 53 -j ACCEPT # dns
iptables -A INPUT -p tcp -i eth0 -j REJECT --reject-with tcp-reset # rechazamos conexiones TCP con paquetes TCP RST
iptables -A INPUT -p udp -i eth0 -j REJECT --reject-with icmp-port-unreachable # rechazamos conexiones UDP con mensajes ICMP de puesto inalcanzable si no estan abiertos
```

```
    iptables -A INPUT -i eth0 -j REJECT --reject-with icmp-proto-unreachable # rechazamos todo el trafico entrante para otros protocolos con mensajes de ICMP inalcanzable

    iptables -A INPUT -i eth1 -p tcp -s $IP_COMP_1 --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT # ssh eth1

    #iptables -A INPUT -i tun6to4 -j ACCEPT


    # Reglas de salida

    echo "[+] Configurando cadenas de SALIDA..."

    iptables -A OUTPUT -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT # http respuestas de mi server

    iptables -A OUTPUT -p tcp --sport 443 -m state --state ESTABLISHED -j ACCEPT # http ssl respuestas se mi server

    iptables -A OUTPUT -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT # http hacia afuera

    iptables -A OUTPUT -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT # http ssl hacia afuera

    iptables -A OUTPUT -o eth0 -s $MI_IP_0 -d $IP_COMP_0 -p tcp --dport 22 -j ACCEPT # ssh peticiones comp como host

    iptables -A OUTPUT -o eth0 -s $MI_IP_0 -d $IP_COMP_0 -p tcp --sport 22 -j ACCEPT # ssh contestaciones yo como host

    iptables -A OUTPUT -o eth1 -s $MI_IP_1 -d $IP_COMP_1 -p udp --sport 123 -j ACCEPT # ntp (servidor)

    iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT # ssh

    iptables -A OUTPUT -o eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT # ssh

    iptables -A OUTPUT -o lo -j ACCEPT # loopback

    iptables -A OUTPUT -p udp -o eth0 -d $IP_DNS --dport 53 -j ACCEPT # dns

    #iptables -T mangle -I OUTPUT -j TTL --ttl-set 148 # modificando ttl

    iptables -A OUTPUT -o eth1 -s $MI_IP_1 -d $IP_COMP_1 -p tcp --sport 22 -j ACCEPT # ssh eth1

    #iptables -A OUTPUT -o tun+ -j ACCEPT


    # Reglas de forward

    #echo "[+] Configurando cadenas de FORWARD..."

    #iptables -A FORWARD -i tun+ -j ACCEPT
```

```
    #iptables -A FORWARD -i tun+ -o eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT
    #iptables -A FORWARD -i eth1 -o tun+ -m state --state RELATED,ESTABLISHED -j ACCEPT

    # Durmiendo para tener tiempo de solucionar fallos
    sleeping

    # Si algo fue mal, reiniciaremos la maquina e iptables quedará limpio
    rebota
}

stop() {
    # Limpiar reglas
    clear

    # Politica de aceptar todo
    echo "[+] Quitando politica dura..."
    iptables -P INPUT ACCEPT
    iptables -P OUTPUT ACCEPT
    iptables -P FORWARD ACCEPT
}

clear() {
    # Limpiar reglas
    echo "[+] Limpiando reglas..."
      iptables -F
      iptables -X
      iptables -t nat -F
      iptables -t nat -X
      iptables -t mangle -F
      iptables -t mangle -X
}
```

```
sleeping() {
   # Durmiendo para tener tiempo de solucionar fallos
      echo "[+] Durmiendo... (tienes $TIME segundos para comprobar que todo vaya bien y no quedar mangao)"
      sleep $TIME
}

rebota() {
   # Si algo fue mal, reiniciaremos la maquina e iptables quedara limpio
      echo "[+] Reiniciando maquina..."
      reboot
}

# En un case gestionamos start|stop|restart
case "$1" in
   start)
      start
      exit 0
   ;;
   stop)
      stop
      exit 0
   ;;
   restart)
      stop
      start
      exit 0
   ;;
   **)
      echo "Uso: $0 {start|stop|restart}" 1>&2
```

```
        exit 1
    ;;
esac
```

Para probar que funciona, lo ejecutamos con el comando >sh miFirewall.sh start y tenemos 120 segundos para, mientras tenemos la terminal abierta con el firewall corriendo, abrir otra terminal y probar si nos deja entrar. Aquí hay dos escenarios posibles:

Si nos deja entrar, es que el firewall está bien, de tal forma que hacemos CTRL+C y ya se queda ahí corriendo, podemos hacer el comando >iptables -L para ver las reglas aplicadas. Podemos limpiar todas las reglas simplemente parando el firewall con el comando >sh miFirewall.sh stop.

Si no nos deja entrar, es que el firewall está mal, de tal forma que hacemos CTRL+C y hacemos el comando >sh miFirewall.sh stop. para pararlo. De todas formas si en 120 segundos no le damos a CTRL+C automáticamente se reiniciará el equipo de forma que se limpiarán todas las reglas automáticamente.

*-i: Interfaces de input

*-o: Interfaz de output

*-p: Protocolo

*--sport: Selecciona o excluye puertos de un determinado origen

*--dport: Selecciona o excluye puertos de un determinado destino

*-j: Para aplicar una política

PREGUNTAAAS.

- Borrar contenido known_hosts e conectarse ao compañeiro.

- Entrar por ssh sin que pida contraseña(authorized_keys)

- Crear tunel ssh para securizar servicio

- sshfs

- Mostrar certificado insertado no navegador(Autoridad certificadora)

- Mostrar que funciona o cliente-servidor vpn

- Mostrar o que obtuvemos con openvas

- Mostrar que funciona servidor ntp con autenticación

- Firewall