Practica 3

ILLIA NECHESA | PABLO OSPINO

Seguridad en Sistemas Informáticos | 4º Curso | 06-11-2019

Certificados SSL y ataques de contraseña

Configuración

Primero procedemos a descargar la maquina virtual, y seguido abrimos VirtualBox, aplicando todas las configuraciones necesarias, indicadas en las instrucciones de las practicas.

En cuanto a la configuración de red, vamos a utilizar NAT cuando sea necesario conectarse a internet, y Host-Only cuando sea necesario tener conectividad entre varias máquinas virtuales.

Para conectar con ssh, ponemos las dos maquinas en host only, y escribimos

- ssh usuario@<ip> → En donde pone <ip> escribimos la IP donde esta corriendo el servidor (no poner <>) → Seguidamente nos pide una contraseña, que seria la del servidor al cual nos estamos conectando.

Añadir usuarios

Ahora vamos a hacer usuarios en una distribución de Unix.

- sudo useradd ssi29
- sudo passwd ssi29
- ssi29

Conexión

Para conectarnos entre las dos máquinas, lo primero es obtener la IP.

- ifconfig → 192.168.56.102
- ssh ssi29@192.168.56.102 → Seguidamente nos pide la contraseña
- ssi29 →Esta es la contraseña de la ip del servidor

Una vez hecho esto, ya hemos establecido conexión

Permisos de usuarios para el acceso ssh

Ahora, configuramos los permisos para unos usuarios, y para otros no.

Primero, modificamos el archivo /etc/ssh/sshd_config

```
Port 1234
PermitRootLogin no
AllowUsers ssi29
```

- sudo service ssh reload

Comprobamos que todo funcione correctamente

```
ssh ssi29@192.168.56.102
> ssi29
Connection
exit

ssh tlm@192.168.56.102
> tlm
Permission denied, please try again Connection
exit
```

Podemos ver que funciona correctamente, ya que nos deja conectarnos con el usuario que le hemos indicado en la configuración (ssi29), pero no con otro usuario.

Autentificación de usuario con clave pública

Comenzamos por generar el par de clave pública y clave privada.

```
> ssh-keygen -f key5
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in key5.
Your public key has been saved in key5.pub.
The key fingerprint is:
SHA256:KXu+E/SZtwhboareFqNrt1jZL9JuNRA/yxaSiHDZJOY
tlm@tlm
The key's randomart image is:
+---[RSA 2048]----+
|    o+.          |
|  .oo.. .        |
|   oE. . +       |
|    . . =.=      |
|      ..S* B     |
|      o=+ @ .    |
|     .+=oB + .   |
|    o+=o*.. .    |
|   o==o=++.      |
+----[SHA256]-----+.
```

Una vez tenemos generado el par de claves copiamos la clave pública con el comando `ssh-copy-id` al servidor de ssh.

```
ssh-copy-id ssi29@192.168.56.102
> ssi29 # Use the password
```

Una vez copiado ahora somos capaces de conectarnos al servidor sin que nos pida contraseña. La autenticación se realiza con el par de claves.

```
ssh ssi29@192.168.56.102
```

Ataques de login

Para ello, es necesario realizar un cambio de ip para la máquina de hydra.

```
> sudo ifconfig eth0 192.168.56.103, los otros tienen la 101 y 102.
```

Ahora vamos a instalar hyndra:

- ➔ NAT network configuration
- ➔ sudo apt-get update
- ➔ sudo apt-get install hydra

```
hydra -l root -p admin 192.168.1.105 -t 4 ssh
hydra -l user_name -p single_password ip -
t n_threads server_type_it_will_attack
```

```
# El usuario que vamos a atacar
- -l
# Listado de usuarios
- -L
# Contraseña con la que vamos a probar
- -p
# Lista de posibles contraseñas word list en vez de una sola contraseña
- -P
# Numero de hilos
- -t
```

```
hydra -l ssi29 -p dict.txt 192.168.56.102 -
t 4 ssh # Diccionario con todas las palabras para probar
crunch 3 3 > dict.txt # Fuerza bruta de todas conbinaciones de 3 caracteres
```

Defensas

```
- [x] Configure que solo algunos usuarios puedan entrar
- [x] Pruebe a cambiar el puerto del servidor de ssh
  - Search for port in nano -w /etc/ssh/ssh_config
  - Port 889, different
- [x] Deshabilitar root login
- [x] Passwordless login - with ssh-keygen
  - Disable empty password
- [x] Pruebe a cambiar los parámetros de ssh que afectan a la velocidad con
 que un atacante puede intentar contraseñas
```

Probamos a cambiar los parámetros de segundos que afectan a la velocidad con la que un atacante puede intentar contraseñas.

```
> sudo nano -w /etc/ssh/sshd_config
...
LoginGraceTime 120 # Bajado a 10 y da error
...
```

120 indica el número de segundos en que la pantalla de login estará disponible.

Bajamos el número de segundos para que no pueda estar una pantalla abierta.

Ahora vamos a instalar y configurar monitores de login que corten el acceso a quien haga varios intentos de acceso fallidos.

```
> sudo nano -w /etc/ssh/sshd_config
...
MaxAuthTries 2
...
```

Por último, vamos a investigar sobre el port-knocking.

- ➔ Se trata de un mecanismo para abrir puertos externamente en un firewall mediante una secuencia preestablecida de intentos de conexión a puertos que se encuentran cerrados. Una vez que el firewall recibe la secuencia de conexiones correcta, permite al host que realizó los intentos conectarse al puerto especifico.
- ➔ Instrucciones

- apt-get update -y
- apt-get upgrade -y

➔ Instalar y configurar tablas
- ufw disable → hay que deshabilitar ufw antes de instalar las tablas
- apt-get install iptables iptables-persistent → instalamos iptables
- iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT → permitir todas las conexiones por iptables
- iptables -A INPUT -p tcp --dport 22 -j REJECT → bloquear ssh de entrada por el puerto 22
- netfilter-persistent sabe → guardar las reglas del firewall
- netfilter-persisten reload

➔ Ahora podemos testear que puertos ssh están bloqueados
- nmap 192.168.0.190

➔ Ahora instalamos y configuramos Knockd
- apt-get install knockd -y
- nano /etc/default/knockd
  - Dentro, cambiamos el valor de START_KNOCKD de cero a uno.
- nano /etc/knockd.conf → Aquí cambiamos la secuencia tanto de openSSH como de close SSH por la que queramos

```
[options]

    logfile = /var/log/knockd.log



[openSSH]

    sequence = 10001,10002,10003

    seq_timeout = 20
```

```
    tcpflags = syn

    command = /sbin/iptables -I INPUT -s %IP% -p tcp --dport 22
-j ACCEPT




 [closeSSH]

    sequence = 10003,10002,10001

    seq_timeout = 20

    command = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 22
-j ACCEPT

    tcpflags = syn
```

- systemctl start knockd → lanzamos el servicio knock para aplicar los cambios

→ Ahora vamos a testea Knockd desde el cliente
  - apt-get install telnet -y

→ Lanzamos los siguientes comandos en orden

```
telnet 192.168.0.190 10001

telnet 192.168.0.190 10002

telnet 192.168.0.190 10003
```

Para ver el log introducimos el siguiente comando:

- tail -f /var/log/syslog

```
[2017-09-25 09:11] 192.168.0.191: openSSH: Stage 1

 [2017-09-25 09:12] 192.168.0.191: openSSH: Stage 2

 [2017-09-25 09:13] 192.168.0.191: openSSH: Stage 3

 [2017-09-25 09:13] 192.168.0.191: openSSH: OPEN SESAME

 [2017-09-25 09:13] openSSH: running command: /sbin/iptables -I
INPUT -s 192.168.0.191 -p tcp --dport 22 -j ACCEPT
```

- nmap 192.168.0.190 → Ahora ya somos capaces de conectar nuestro servidor ssh desde el cliente.

```
Starting Nmap 6.40 ( http://nmap.org ) at 2017-09-25 9:20 IST

 Nmap scan report for 192.168.0.190

 Host is up (0.00037s latency).

 Not shown: 996 closed ports

 PORT STATE SERVICE

 22/tcp open ssh

 80/tcp open http

 443/tcp open https

 3306/tcp open mysql
```

```
 MAC Address: 08:00:27:7C:5B:40 (Cadmus Computer Systems)




 Nmap done: 1 IP address (1 host up) scanned in 1.69 seconds
```

Por último, para cerrar la conexión, introducimos los siguientes comandos en orden secuencial.

```
telnet 192.168.0.190 10003


telnet 192.168.0.190 10002


telnet 192.168.0.190 10001
```

**

Todas estas pruebas son suponiendo que la IP del servidor es `192.168.0.190` y la del cliente `192.168.0.191`

**

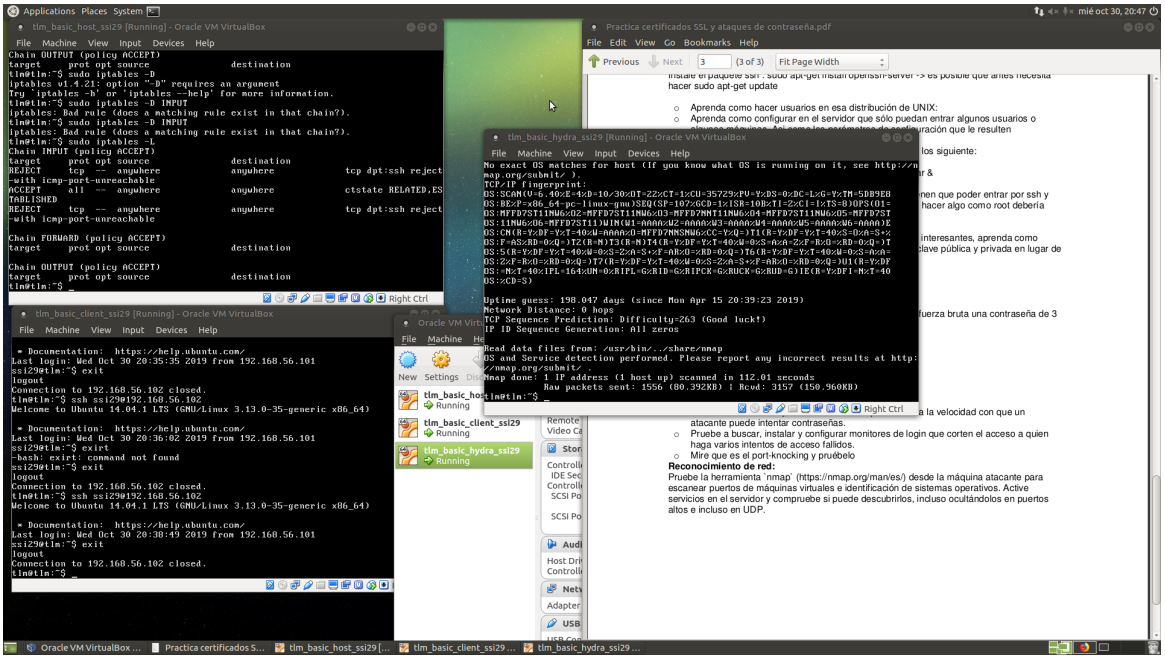
Aquí dejo los screenshots con nuestras pruebas

NMAP:

Según la página oficial de nmap, esta herramienta es definida como un mapeardor de reades (network mapper). Se diseñó con el fin de la auditoria para analizar rápidamente grandes redes. Sin embargo, esta potente herramienta puede hacer muchas mas tareas.

Nmap permite analizar un sistema individual para conocer cuales son los puertos que tiene abiertos. Tanto podemos pasar la ip o el dominio. Por ejemplo `nmap unavarra.es` o `nmap 192.168.0.101`. La opción -v nos da información mas detallada. Si concatenamos mas ips va a analizar cada una de ellas. Además podemos optar por analizar todas las maquinas que pertenezcan a una red colocando un asterisco al final: 192.168.0.*.

También podemos obtener información del sistema operativo. Si ejecutamos el comando con las opción -A seguido de la ip o también -O. Hemos probado este comando con las maquinas virtuales y este es su output.



Si queremos obtener cuales son los hosts encendidos en la red podemos ejecutar el comando nmap con la opción de -sP, de esta forma ahorramos la detección de puertos y demás, solo se centra en conocer cuales hosts están activos. Este scaneo lo hace realizando pings a cada una de las máquinas. También existen opciones como:

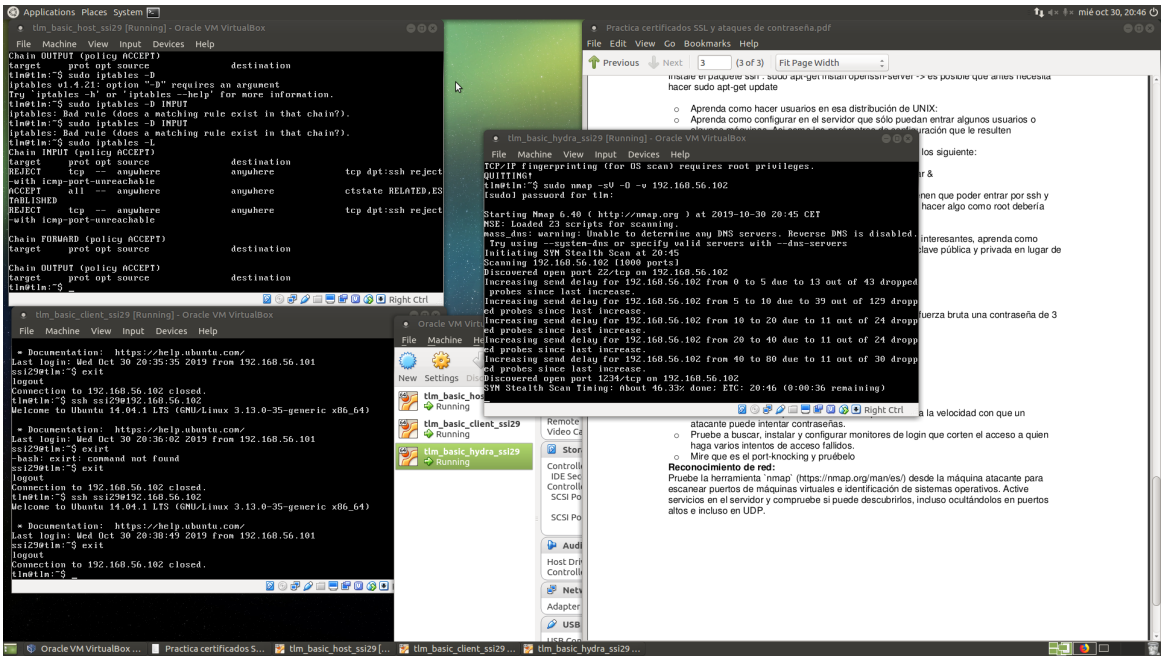
-sT scaneo con TCPs

-sS scaneo con paquetes SYN

-sF scaneo con FIN

-sX scaneo con un paquete Xmastree

-sN scaneo con un paquete null.



Sin embargo, estos scaneos no funcionan con puertos UDP. Para estos existe una opción que es -sU. De esta forma escaneamos los puertos si están activos y son UDP.

Como vemos nmap es una potente herramienta de escaneo y de exploración de la red. Tanto como para fines malignos como para auditoria y control de una red de ordenadores.