P2 Lección-1

COSAS QUE DEBEMOS SABER

UN POCO DE SEGURIDAD

Se espera que usted sea capaz de instalar, configurar y gestionar la funcionalidad básica de fail2ban así como de ejecutar rkhunter y determinar si hay alguna amenaza en el sistema.

Para evitar ataques de fuerza bruta podemos usar **fail2ban** que mete en una lista negra (encarcela) las IPs que han intentado iniciar sesión de manera errónea varias veces seguidas en un intervalo de tiempo predefinido.

RootKit Hunter es un software que permite analizar el sistema para ver si hay software malicioso instalado en la máquina (*rootkits, backdoors,exploits, etc.*)

RootKit Hunter

Rkhunter no se ejecuta automáticamente en segundo plano sino que es tarea de los usuarios ejecutarlo a demanda según cuando queramos analizar nuestro sistema.

Lo primero que haremos será tomar una **muestra del estado actual de nuestro sistema operativo** para compararlo las próximas veces que ejecutemos el programa con ella. Para ello tecleamos.

```
sudo rkhunter – -propupd  (sin espacio entre los guiones)
```

Una vez creada la base de datos cada vez que queramos **analizar nuestro equipo** ejecutaremos el siguiente comando para ejecutar un análisis completo del sistema.

```
sudo rkhunter -c – -enable all  (sin espacio entre los guiones)
```

Una vez finalice podemos ver los resultados completos del análisis y saber así si nuestro sistema está posiblemente infectado o limpio de malware.

Para finalizar también podemos ejecutar una actualización de las bases de datos de virus y malware tecleando:

```
sudo rkhunter – -update  (sin espacio entre los guiones)
```

Fail2ban

Es una herramienta de seguridad escrita en Python fundamentalmente para cualquier servidor que preste servicios públicos. Su principal función **es securizar un servidor** del siguiente modo:

- Evitar accesos indeseados a nuestro equipo o servidor

- Evitando ataques de fuerza bruta para que un tercero averigüe nuestra contraseña o tumbe el servidor.

Por un lado fail2ban está monitorizando las autenticaciones que una determinada IP hace a un determinado/s puerto/s y servicio/s. Para ello, está consultando permanentemente los logs de autenticación de nuestro sistema como por ejemplo el /var/log/auth.log

Si fail2ban detecta un número determinado de intentos de conexión fallidos bloqueará la IP que está intentando acceder a nuestro equipo o el servicio. La **forma de bloquear la IP** será **introduciendo una regla en el Firewall de nuestro equipo** o servidor durante un tiempo determinado. Una vez transcurrido el tiempo, la IP que fue desbloqueada podrá intentar conectar de nuevo a nuestro servidor.

1. PARA CONFIGURAR FAIL2BAN:

La ubicación que contiene la totalidad de los filtros es la siguiente:
`/etc/fail2ban/filter.d`

Algunos de los filtros que podemos encontrar son los siguientes:

Archivo que contiene el filtro	Función
sshd.conf	Filtro para detectar autenticaciones erróneas a un **servidor SSH**
proftpd.cof	Filtro para detectar autenticaciones erróneas al **servidor ftp Proftp**
exim.conf	Filtro para detectar autenticaciones a un **servidor de correo Exim.**

La **función de los filtros** es definir las expresiones regulares para detectar autenticaciones erróneas o ataques a nuestro equipo o servicio.

Una vez definida una expresión regular se irá comprobando que esta expresión no aparezca en ninguno de los logs de autenticación de nuestros servicios. En el caso que la expresión regular aparezca en los logs se contabilizará un intento fallido de autenticación.

Si pretendemos usar los servicios estándares predeterminados de fail2ban no será necesario modificar ni crear ningún filtro

2. Configuración y consulta de las acciones

Ubicación: `/etc/fail2ban/action.d`

En esta ruta se guardan la totalidad de scripts que definen diferentes tipos de acciones a aplicar cuando se detecta un intento de ataque, se arranca alguna de las jaulas, etc.

En principio no tendremos que modificar ni configurar parámetro de este apartado. Fail2ban ya trae predefinidas multitud de acciones.

3. Archivo jail.conf

`jail.conf` es el **archivo de configuración más importante.** En este archivo es donde activamos, desactivamos y configuramos fail2ban para que proteja determinados servicios.

Realizamos las siguientes acciones:

1. Activamos desactivamos la protección de fail2ban para un servicio
2. Definimos el filtro y las acciones que se aplican a cada uno de los servicios.
3. Se define el puerto de funcionamiento de un servicio determinado para que fail2ban pueda funcionar de forma adecuada
4. Se establece el log a controlar para detectar los intentos de autenticación erróneos.
5. Se establece el número de intentos fallidos que permitimos antes que se aplique la acción para bloquear los intentos de autenticación erróneos.
6. etc

Para **acceder a la configuración de las reglas** ejecutamos el siguiente comando en la terminal:
`sudo nano /etc/fail2ban/jail.local`

El contenido del fichero jail.local está repleto de preconfiguraciones estándares para cada uno de los servicios que podemos proteger. Para activar y configurar las preconfiguraciones tan solo tenemos que comentar, descomentar y modificar los parámetros de cada una de la secciones.

A modo de ejemplo localizamos la sección destinada a proteger nuestro servidor SSH contra ataques DdoS:

```
[ssh-ddos]
enabled = false ----> Vemos que la protección está desactivada
port = ssh
filter = sshd-ddos
logpath = /var/log/auth.log
maxretry = 6
```

El significado de los parámetros que nos podemos encontrar dentro de una sección es el siguiente:

Parámetro	Función
enabled =	Con los valores **true** y **false** activamos y desactivamos la protección que ofrece fail2ban para un determinado servicio.
port =	Definición de los puertos en que están trabajando los servicios que queremos proteger.
filter=	Se define el nombre del filtro a aplicar para detectar intentos de autenticación fallidos. Para ver la totalidad de filtros disponibles se puede visitar la ubicación **/etc/fail2ban/filter.d**

logpath =	Definición del log a monitorizar para detectar los intentos fallidos de autenticación.
maxrety =	Número de intentos de autenticación máximos fallidos antes de aplicar una acción de bloqueo.
action =	Para definir las acciones de bloqueo que se aplicarán en cada uno de los servicios que queremos proteger. La totalidad de acciones disponibles se hallan en la ubicación **/etc/fail2ban/action.d**
findtime =	Definimos el tiempo ha transcurrir para que el contador de intentos fallidos de una determinada IP se resetee.
bantime =	Definimos el tiempo en segundos que queremos bloquear una determinada IP. Normalmente un valor de **600** segundos es más que apropiado.

Para que **los cambios se hagan efectivos** ejecutamos el siguiente comando para reiniciar fail2ban:
`sudo service fail2ban restart`

5. Servicios con los que podemos usar fail2ban

1. SSH
2. Servidores web como por ejemplo lighttpd, Apache y nginx.
3. En servidores de email
4. Servicios de proxy como Squid
5. Otros servicios como Asterisk, FreeWITCH, Druotal, WordPress,etc.
6. etc.

GESTORES DE PAQUETES

Debe ser consciente de que algunas distribuciones como Fedora ya usan una de estas alternativas.

DNF

Es una reescritura de YUM que utiliza las características de ZYpp, más en particular, la dependencia para la resolución de capacidades. DNF es el gestor de paquetes **por defecto de Fedora 22 .**

¿Cómo utilizarlo?

Para **instalar** el paquete <paquetillo> :

```
sudo dnf install <paquetillo>
```

Para **eliminar** el paquete <paquetillo> :

```
sudo dnf remove <paquetillo>
```

Para **actualizar la instalación de tu servidor Fedora:**

```
sudo dnf update
```

Snap

Los paquetes snap son sistemas de archivos comprimidos de solo lectura squasFS, que contienen el código de tu aplicación y un archivo snap. yaml que contiene información específica del paquete. Se trata de sistemas de archivos de solo lectura, que una vez instalados disponen de un área que se puede escribir.

¿Cómo utilizarlo?

Para **instalar** un paquete snap llamado <paquetillo>:

```
sudo snap install <paquetillo>
```

Para **eliminar** el paquete <paquetillo> :

```
sudo snap remove <paquetillo>
```

Para **comprobar si tenemos instalado** el paquete <paquetillo> :

```
snap find <paquetillo>
```

Otros comandos que nos ayudan a mejorar las instalaciones:

`help` Nos muestra todos los parámetros que podemos usar

`list` Muestra todos los paquetes snap instalados y sus nombres

GESTIONANDO EL CORTAFUEGOS

Se espera que usted sea capaz de abrir y cerrar puertos así como de comprobar su estado. Para ello, no solo usaremos las opciones de estos comandos sino que haremos un escaneo de puertos con el comando nmap.

El comando nmap

Lanzando un escaneo de red con Nmap podemos visualizar una gran cantidad de información: hosts activos en la red, sistema operativo que están ejecutando, puertos y servicios abiertos a través de la red, tipos de firewall que están utilizando...

¿Cómo utilizarlo?

Para ver las opciones generales lanzamos el comando sin parámetros:

```
nmap
Nmap 6.40 ( http://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL : Input from list of hosts/networks
  -iR : Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile : Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver

[...]
```

Para **escanear un host o red completa :**

```
nmap <ip del host>
Starting Nmap 6.40 ( http://nmap.org ) at 2014-09-26 18:12 CEST
Nmap scan report for centos7 (192.168.1.100)
Host is up (0.000023s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
Nmap done: 1 IP address (1 host up) scanned in 2.56 seconds
```

Como podemos ver, este escaneo sencillo nos muestra **el sistema operativo del host, los puertos abiertos/cerrados/filtrados y los servicios asociados**. Esto mismo lo podríamos hacer para un segmento de red completo y ejecutaría el escaneo en todas las máquinas de **la red:**

```
nmap 192.168.1.0/24
```

Si queremos **excluir algún host de la red,** los especificamos con el parámetro «–exclude» seguido de los hosts separados por comas:

```
nmap 192.168.1.0/24 --exclude 192.168.1.1,192.168.1.100
```

Podemos leer los hosts a incluir y excluir en el escaneo directamente de una lista en archivo de texto.

Escanear hosts de un archivo: `nmap -iL archivo_de_entrada`

Excluir hosts de un archivo: `nmap --excludefile archivo_de_entrada`

Encontrar hosts de una red

Con los ejemplos anteriores, además de buscar hosts en una red también se escaneaban todos los puertos y servicios de cada host. Esto llevará bastante tiempo, así que si lo que queremos es únicamente hacer un rastreo rápido de la red en busca de los hosts que están online tenemos los siguientes ejemplos:

Mostrar una lista de los hosts de la red sin mandar ningún tipo de paquete al host, únicamente ejecuta un Reverse DNS para encontrar el hostname:

```
nmap -sL 192.168.1.0/24
```

Buscar hosts pero sin efectuar escaneo de puertos. Básicamente es un «ping scan», sólo lanza un ping al host y si responde lo lista por pantalla:

```
nmap -sn 192.168.1.0/24
```

Buscar hosts mediante ping a los hosts:

```
nmap -sP 192.168.1.0/24
```

Escaneo de puertos (Esto es lo más importante para el examen)

Con el primer comando que hemos ejecutado ya se estaban escaneando los puertos de cada host que se consultaba. Concretamente sin parámetros se escanean 1000 puertos TCP. El estado de los puertos se divide en los siguientes, y así es como nos lo mostrará en la salida del comando: **open, closed, filtered, unfiltered, open|filtered, y closed|filtered**. Básicamente, puerto abierto, cerrado, filtrado, no filtrado, abierto|filtrado y cerrado|filtrado.

Escanear **todos los puertos** de un host **con información extendida** (verbose):

```
nmap -v 192.168.1.100
```

Escanear **un único puerto** en un host:

```
nmap -p 80 192.168.1.100
```

Buscar **puertos TCP abiertos** en una máquina:

```
nmap -sT 192.168.1.100
```

Buscar **puertos UDP abiertos** en una máquina:

```
nmap -sU 192.168.1.100
```

Escaneo de **protocolos de un host** (además de TCP,UDP, muestra disponibilidad de ICMP, IGMP...):

```
nmap -sO 192.168.1.100
```

Escanear un **rango de puertos:**

```
nmap -p 80-200 192.168.1.100
```

Escanear un **rango de puertos y unos puertos específicos**, tanto TCP como UDP:

```
nmap -p U:53,161,8888,T:1000-2000,80,25,8888,8080 192.168.1.100
```

El comando screen

Usted deberá saber hacer uso de screen usando la opción más común como es la de recuperar una "ventana" y "desmarcarla" de otro sitio.

De cara a abrir una sesión con una shell y poder cerrar la conexión sin que la shell termine, podemos usar los comandos **screen y tmux.** Gracias a estos comandos, podemos tener varias sesiones abiertas y reconectarnos a ellas aunque la conexión se haya perdido. Por otra parte, **terminator** es una consola que en modo gráfico (tmux para modo texto) permite tener varias terminarles abiertas en una única ventana así como hacer operaciones de broadcast, es decir, teclear en una ventana y que el texto aparezca en otras.

Uso de screen

Si nos queremos **desvincular de una sesión** hemos de utilizar la siguiente combinación de teclas:

```
Ctrl-a d
```

Para **reconectar a una sesión** que ha sido desligada

```
screen -r
```

Si hay más de una sesión disponible necesitas proporcionar el identificador de la sesión a la que quieres conectarte, puedes obtener una lista de las sesiones con:

```
screen -ls
```

Y luego reconectar usando **screen -r seguido de la sesión** deseada:

Una vez has terminado de trabajar puedes **cerrar screen** de dos manera, la primera usando el comando `exit` para cerrar la terminal hasta que te aparezca el mensaje **[screen is terminating]**. La alternativa es usar la combinación de teclas `"Ctrl-a" "x",` que debería mostrarte un mensaje de confirmación en el que debes pulsar y.

WEBMIN

Usted debe saber instalar y dejar en funcionamiento Webmin además de razonar ventajas y desventajas de usar Webmin frente a SSH.

Webmin es una interfaz web que nos permite administrar el sistema.

¿Cómo instalarlo y configurarlo?

https://www.haulmer.com/docs/untitled-6/

Ventajas y desventajas frente a SSH