Centro de Procesamiento de Datos

Práctica 9. Limitando el acceso al servidor

Objetivos

Evaluar herramientas de Gestión de Host IDS como DenyHosts, Fail2ban.

Alguno de los ejercicios propuestos se incluirán en el proyecto final.

Descripción

Para poder garantizar la seguridad en un servidor es importante contar con herramientas que detecten de forma automática intentos no autorizados de acceso.

Resulta conveniente que un servidor de esté conectado directamente a Internet tenga activo mecanismos de detección de intrusos.

Realización

En esta práctica analizamos diversos programas orientados al análisis de registros y realizar alguna contramedida para proteger al computador.

I) DenyHost

DenyHost es un programa Python que analiza los registros del servidor SSH (fichero */var/log/secure)* detectando los intentos de conexión (login) fallidos, los ataques basados en diccionario y los ataques de fuerza bruta. Si identifica un ataque bloquea las direcciones IP de origen. En versiones anteriores de denyhosts se añadía una entrada al fichero */etc/hosts.deny* en el servidor y evita que la dirección IP pueda acceder durante un tiempo. Las versiones recientes (como la que viene por defecto en ubuntu xenial64) se añade una regla en iptables.

Dispone de un modo de sincronización (a partir de la versión 2.0) que permite compartir datos a través de un servidor centralizado.

Opcionalmente puede enviar un correo informando del ataque detectado. Mantiene un historial de todos los inicios de sesión sospechosas encontrado que incluye los datos y el número de intentos fallidos de la conexión.

Disponible en múltiples distribuciones. Veamos una instalación basada en Yum

```
yum -y install epel-release
yum -y install denyhosts
```

Activar en el inicio

```
systemctl enable denyhosts.service
```

Puesta en marcha del servicio

```
systemctl start denyhosts.service
```

En ubuntu xenial64

```
sudo apt update
sudo apt-get install denyhosts
```

En ubuntu se activa el servicio y se pone en marcha automáticamente.

En ubuntu trusty64:

```
git clone https://github.com/denyhosts/denyhosts.git
cd denyhosts
sudo apt install python-ipaddr
sudo python setup.py install
```

Revisamos el fichero /etc/denyhosts.conf donde podemos controlar en número de intentos fallidos para considerar un ataque.

Podemos editar el fichero para indicar los ordenadores que tienen acceso.

- Ubuntu: */var/lib/denyhosts/allowed-warned-hosts*
- CentOS:*/var/lib/denyhosts/allowed-hosts*

En la carpeta */var/lib/denyhosts* también podemos encontrar los ficheros de información de accesos.

Enlaces relacionados: http://denyhosts.sourceforge.net/faq.html

Ejercicio 1. Crear un par de máquinas virtuales mediante Vagrant, una con CentOS y otra con Ubuntu. (Ver Anexo A).
Podemos pasar a superusuario en ambas máquinas con : *sudo su* -
- Fijar el nuevo password de ubuntu(usuario ubuntu): *sudo passwd ubuntu*
- Fijar el nuevo password de centos (usuario vagrant): sudo passwd vagrant
- En la máquina CentOS: activar el acceso con password en */etc/ssh/sshd_config*
  PasswordAuthentication yes
- Reiniciar el servicio ssh: *systemctl restart sshd.service*

Ejercicio 2. Instalar Denyhosts en la máquina CentOS y comprobar el acceso SSH correcto desde la máquina Ubuntu. Intentar varios accesos erróneos desde Ubuntu y comprobar cómo CentOS bloquea los accesos desde el otro nodo.
Fijar PURGE_DENY = 5m

sudo systemctl restart denyhosts.service

El bloqueo aparece en */etc/hosts.deny*. Si es necesario podemos 'purgar' la tabla con:
systemctl stop denyhosts.service
denyhosts.py --purge-all
systemctl start denyhosts.service

Ejercicio 3. Modificar el fichero */var/lib/denyhosts/allowed-hosts* para autorizar el nodo 192.168.56.12, a pesar de tener fallos de acceso.

II) Fail2Ban

Este programa permite controlar mas servicios que DenyHosts. Además de SSH puede controlar: Apache, Nginx, Lighthttp, Openwebmail, Horde, Mysql, Drupal, Squid, Proftpd, Vsftpd, Postfix, Sendmail, Squirrelmail, Asterix, …

Fail2ban establece "jaulas" (jails) para controlar cada aplicación, de forma que si descubre un posible ataque en el fichero establece una acción. Por ejemplo, en el caso de SSH, añade una regla al cortafuegos bloqueando la entrada de paquetes de la dirección IP durante un tiempo (bantime).

Para instalar fail2ban en Ubuntu:

```
sudo apt-get install fail2ban
```

En CentOS/RPM debemos tener instalado el repositorio EPEL.

```
yum install fail2ban fail2ban-systemd
systemclt enable fail2ban
systemctl start fail2ban
```

```
sudo fail2ban-client status
```

Por defecto se activa la jaula de ssh.

Podemos comprobar las reglas en el cortafuegos

```
sudo iptables -L -n
```

O bien:

```
sudo iptables -S
```

Si queremos desbloquear una IP que ha sido filtrada:

```
fail2ban-client set sshd unbanip <DIRECCION_IP>
```

O bien eliminamos la entrada de las iptables.

```
sudo iptables -D <regla a eliminar>
```

En CentOS/RetHat:

Copiamos el fichero */etc/fail2ban/jail.conf* en */etc/fail2ban/jail.local*

```
cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

Y editamos sólo la copia para modificar la configuración.

Podríamos cambiar:

bantime: Número de segundos que el nodo sospechoso quedará bloqueado.

maxretry: Número máximo de intentos fallidos.

findtime: tiempo en el que se consideran los intentos erróneos.

ignoreip: lista de direcciones IP que son excluidas del control . Nunca que bloquearán. Pueden ser direcciones IP, máscara CIDR o un nombre separadas por espacios. Ej: ignoreip = 192.168.10.0/24

Los distintos servicios los podemos encontrar en */etc/fail2ban/filter.d*

Para activar servicios debemos asignar *enabled = true* en el fichero de configuración.

Ejercicio 4. Instalamos fail2ban en la máquina Ubuntu. Comprobamos el control de acceso SSH como en el ejercicio.

Ejercicio 5. Instalar también Google Authenticator.

Supervision de NGINX mediante Fail2ban.

Supongamos que queremos supervisar un servidor NGINX. Para controlar intentos fallidos de autenticación, editamos /etc/fail2ban/jail.conf:

```
[nginx-http-auth]
enabled  = true
port     = http,https
logpath  = %(ngninx_error_log)s
```

Reiniciamos:

```
systemctl restart fail2ban
```

Ejercicio 6. Instalamos NGINX en la máquina ubuntu (*sudo apt install nginx*). Creamos una página con acceso mediante autenticación. Configurar fail2ban para bloquear el nodo que intenta acceder. Comprobamos entre nodos el bloqueo en los accesos a la página.

Creamos el fichero /etc/nginx/.httpasswd con las claves

```
sudo sh -c "echo -n 'antonio:' >> /etc/nginx/.htpasswd"
sudo sh -c "openssl passwd -apr1 >> /etc/nginx/.htpasswd"
```

Fichero /etc/nginx/sites-enabled/default

```
server {
    listen 80 default_server;
    listen [::]:80 default_server ipv6only=on;

    root /var/www//html;
    index index.html index.htm;

    server_name _;

    location / {
        try_files $uri $uri/ =404;
        auth_basic "Restricted Content";
        auth_basic_user_file /etc/nginx/.htpasswd;
    }
}
```

Reiniciamos el servicio

```
sudo systemctl restart nginx
```

Si queremos que nos avise por correo, añadir en el fichero de configuración de fail2ban /etc/fail2ban/jail.conf:

```
mta = mail
destemail = tudirereccion@correo.com
sendername = AlertaFail2Ban
```

Modificamos en el fichero, en la entrad [DEFAULT]

action = $(action_)s lo cambiamos a: *action = $(action_mw)s*

Configuramos Postfix para que envíe correo SMTP

```
sudo apt-get install mailutils
```

Anexo A: Despliegue de la infraestructura con Vagrant

La página principal es: https://www.vagrantup.com/intro/index.html

Podemos encontrar Vagrant Boxes en: https://app.vagrantup.com/boxes/search

Órdenes básicas para Vagrant:

-vagrant init <nombre_box> : Crea un fichero Vagrant file con la configuración básica

-vagrant up: Inicia máquinas

- vagrant ssh: accede por ssh

-vagrant halt: para máquina

-vagrant global-status: ver máquinas

Cuando creamos una máquina virtual, el directorio /vagrant concide con el directorio donde tenemos el fichero Vagrantfile

Órdenes relacionadas con Boxes:

vagrant box list

vagrant box remove <box>:

Instalación de las máquinas virtuales CentOS y Ubuntu.

Creamos un directorio va1 y creamos el fichero Vagrantfile

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure(2) do |config|
 config.vm.define :centos do |centos_config|
    centos_config.vm.box = "centos/7"
    centos_config.vm.hostname = "centos.vm"
    centos_config.vm.network "private_network" , ip:"192.168.56.11"
    centos_config.vm.provider :virtualbox do |vb|
       vb.name = "centos"
       vb.customize ["modifyvm", :id, "--memory", "512"]
       vb.customize ["modifyvm", :id, "--cpus", "1"]
    end
 end

 config.vm.define :ubuntu do |ubuntu_config|
    ubuntu_config.vm.box = "ubuntu/xenial64"
    ubuntu_config.vm.hostname = "ubuntu.vm"
    ubuntu_config.vm.network "private_network" , ip:"192.168.56.12"
    ubuntu_config.vm.provider :virtualbox do |vb|
       vb.name = "ubuntu"
       vb.customize ["modifyvm", :id, "--memory", "512"]
       vb.customize ["modifyvm", :id, "--cpus", "1"]
    end
    ubuntu_config.vm.provision "shell", inline: <<-SHELL
      export DEBIAN_FRONTEND=noninteractive
      curl -s https://install.zerotier.com/ | bash
      zerotier-cli join a09acf02334c0cde
    SHELL
 end
```

Ejecutamos:

```
vagrant up
```

En la máquina *ubuntu* con xenial64 (ubuntu 16.04) el usuario es *ubuntu*.

Para cambiar el password

```
sudo passwd ubuntu
```

En la máquina centos, usuario *vagrant*, password *vagrant*

El acceso con password está desactivado por defecto en /etc/ssh/sshd_config

Para activarlo: *PasswordAuthentication no*

Anexo B. Instalación de Google Authenticator

Instalamos el módulo PAM

```
sudo apt-get install libpam-google-authenticator
```

En el caso de CentOS, instalamos el módulo PAM con:

```
git clone https://code.google.com/p/google-authenticator/
cd libpam
make
make install
```

Editamos /etc/pam.d/sshd y añadimos al comienzo del fichero

```
auth required pam_google_authenticator.so
```

Editamos /etc/sshd/sshd_config

```
ChallengeResponseAuthentication yes
```

Reiniciamos el servicio

```
sudo systemctl restart ssh.service
```

Instalamos qrencode

```
sudo apt install qrencode
```

Finalmente en cada cuenta ejecutamos

```
google-authenticator
```

En los clientes (móvil:Android, iOS) instalamos el Gauth o equivalente.

También podemos añadir el plugin de Authenticator para Chrome o Firefox.