DENY HOSTS LAB

Overview

This Labtainer exercise explores the use of the denyhosts utility on a SSH server to limit SSH login attempts from an IP address.

Performing the lab

The lab is started from the labtainer working directory on your linux host, e.g., a Linux VM. From there issue the command:

```
labtainer denyhost
```

the resulting virtual terminals include a display of these instructions, a terminal connected to a client and a terminal connected to a SSH server.

As with all Labtainer exercises, you can restart a fresh instance of the lab by using the "-r" option. Note however that will delete any of your previous results.

Tasks

Brute force attacks on SSH servers may attempt thousands of different user IDs and passwords, often using automated tools, or "bots". The `denyhosts` utility allows an administrator to deny access from IP addresses that are the source of multiple failed login attempts.

1. R**eview configuration files**

we will consider three key files associated with the `denyhosts` utility.

Key file #1

`auth.log`

The `/var/log/auth.log` file, reflects the success of login attempts. Watch that file using:

```
sudo tail –f /var/log/auth.log
```

Then ssh to `172.20.0.3` as the user "`hank`" with the password hank21

Note: when the system displays " `Are you sure you want to continue connecting (yes/no)`? Type "`yes`"

Observe what happens in the `/var/log/auth.log` file

Then, exit and again ssh to `172.20.0.3`, but this time type an incorrect password. Observe the result in the `/var/log/auth.log` file.

The denyhosts utility observes events in the `auth.log` to detect failed login attempts.

Key file #2

`denyhosts.conf`

review the content of `/etc/denyhosts.conf` :

```
sudo less /etc/denyhosts.conf
```

Note in particular the description and values for
"`DENY_THRESHOLD_INVALID` and `DENY_THRESHOLD_VALID`"

Key file #3

`hosts.deny`

This system file is used by networking functions to block connections from selected IP addresses. An entry in this file will block the connection prior to it reaching the intended service. Thus, the SSH daemon will never see connection attempts from these sources, thereby helping to avoid consuming system resources. The denyhosts utility adds entries to this file when the quantity of failed login attempts recorded in the `/var/log/auth.log` exceeds the thresholds defined in `/etc/denyhosts.conf`.

2. Lock out a valid user using a bot

the bot.py program in the client home directory is a crude bot that attempts a brute force ssh login by iterating through simple passwords based on the user ID. For Example, if given a user ID of "hank", it will try: `Hank1`, `hank2`, `hank3`, and so on until it gets the password right. Before you start the bot, go to the server and:

```
sudo tail the /var/log/auth.log file
```

On the client, make not of your IP addresses:

```
ifconfig
```

Then start the bot, giving a user of "`hank`"

```
./bot.py hank
```

Watch the output, the program will appear to hang once the invalid login threshold is met.
Based on your review of the client IP to be added to the `hosts.deny` file?

Did the connection get denied at the point you expected? If not consider the implications of the `DAEMON_SLEEP` value in the `denyhosts.conf` file.

3. R**estore login ability of the valid user**
View the `/etc/hosts.deny` file, and note the entry for your IP address. That will prevent anyone from that IP from connecting to the SSH server.

What if the login failures were an honest mistake?
Simply deleting the entry in `/etc/hosts.deny` file is generally not sufficient because `denyhosts` keeps other additional databases.

One option is to "whitelist" the client computer by adding an entry to the `/etc/hosts.allow` file on the server, e.g.,

```
ALL: 172.20.0.2
```

the offending IP address may have also been blocked using "`iptables`". On the server , you can see if this is that case using:

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

If the IP address appears in the output, you can remove the block using:

```
sudo iptables -D INPUT -s 172.20.0.2 -j DROP
```

Now confirm that you are able to SSH from the client

4. **Lock out an invalid user**

First, change the Ip address of the client so that our client is no longer in the `hosts.allow` whitelist.
On the client:

```
ifconfig eth0 172.20.0.9
```

Then try the bot.py again, but this time provide a different user e.g.,:

```
./bot.py tony
```

Did the connection get rejected after a similar number of attempts as when you ran the bot with the "`hank`" user? Why?

Consider why you may want a different threshold for attempts with valid user names as opposed to invalid user names.

Stop the labtainer

when the lab is completed, or you would like to stop working for a while, run:

```
stoplab denyhost
```

from the host labtainer working directory. You can always restart the labtainer to continue your work. When the labtainer is stopped, a zip file is created and copied to a location displayed by the stoplab command. When the lab is completed, send that zip file to your instructor.