Step-by-step guide to Linux security for beginners

Clément Levallois

2017-04-03

Table of Contents

last modified: 2017-04-10

Ordering the server

- Server ordered on Hetzner.de (based in Germany, dirt cheap, but without management.)
- I use Debian, version 8.7 (why?)
- Vi is used as a text editor in the following
- we are logged as root first

Get the latest versions of all packages

Do:

```
sudo apt-get update && sudo apt-get upgrade
```

Because:

```
apt-get update
```

→ refreshes the repositories and fetches information about packages that are available online.

```
apt-get upgrade
```

→ downloads and installs updates for all installed packages - as long as it doesn't bother dependencies (install new packages, remove old ones or crosses a repo source (switch a package from one repo to another)).

(source)

Install this package to get the clock of your server right:

```
aptitude install ntp
```

Then define your time zone (the one where your server is located).

This step helps when your server needs to be synchronized with other servers.

Harden the kernel

Source: http://www.pontikis.net/blog/debian-wheezy-web-server-setup

The kernel is the software at the closest of the machine: it is provided by the Linux distribution you

use.

A configuration file offers parameters which tune the kernel to make things harder for an intruder. Here I rely exactly on the tutorial by Pontikis:

Create a new file, so as to preserver / not to mess up the original file:

```
vi /etc/sysctl.d/local.conf
```

- Paste the contents of this file:
- Close the file
- reboot the server

Forward root mail

Source: http://www.pontikis.net/blog/debian-wheezy-web-server-setup

```
vi /etc/aliases
```

Add this line if not already present:

root:youraddress@email.com

Then, rebuild aliases:

```
newaliases
```

Change the SSH port

By default, loggging to the server via SSH is done on the port 22. Knowing that, attackers scan the port 22. Changing the port to a different one makes the attacker's job more difficult. To do that:

```
vi /etc/ssh/sshd_config
```

Text to change in the file: change port SSH 22 by a new port (**let's say 1234**), write the new port down somewhere

```
service sshd restart
```

Creating a user and disabling logging for root

We should now disable root login via SSH. Why? Because attackers would know that a "root" user is available to log in, and it just remains to attack its password.

With the root user disabled at the SSH login step, the attackers must guess **both** the username and its password to access the connection, and that's much harder.

Of course, an attacker who aims at you or your server specifically (a "targeted" attack) would expect a series of usernames (in my case "seinecle", the name I use on all social media), so don't use it either.

So the logic is the following: we will create a user with much less priviledges than the root user. Only this weak user will have the right to connect to the server.

The user will be "enough" for regular tasks on the server. If we need the admin rights of root to install stuff or else, we will **temporarily** switch from this weak user to root to execute what we need, but then revert back to this weak user.

This way, we limit greatly the exposure of root privileges to the outside.

The steps:

1. making sure we have installed the "sudo" command that will allow us to switch from a weak user to root.
2. creating a weak user
3. giving rights to this user to establish a connection to the server (not just act on it once logged)
4. removing the rights of root to connect to the server.

1. Installing the sudo command:

apt-get install sudo

2. Adding a new user (let's call it "myUser")

```
adduser myUser -s /bin/bash
passwd myUser
vi /etc/sudoers
```

and place the following line:

```
myUser    ALL=(ALL)
```

Have a strong password ready. Other questions (email, phone...) can be left empty by just pressing

enter.

3. Enabling server connections via myUser

text to add still in the file sshd_config:

AllowUsers myUser

Then restart the SSH service:

```
service sshd restart
```

4. Disabling connection through root

```
vi /etc/ssh/sshd_config
```

Text to change in the file:

PermitRootLogin no

From there on, you cannot login to the server from root, only from myUser!

Let's try it. Create a new SSH session with myUser. Then:

Switch to root privileges:

```
su -
```

(you must enter the root password at this step)

Disabling password authentication, enabling SSH

Password authentication is less secure than SSH public key. A password transits through the Internet for the auhtentication, it can be hacked at this step.

A SSH private key is not transmitted on the wire. So, it can't be hacked this way.

A detailed explanation is available here.

How to generate a SSH key?

- On Windows, use Puttygen.
- On Mac, use the Terminal
- On Linux, use the ssh-keygen command

How to disable password auth and enable SSH?

Logging through SSH rather than passwords can be hair rising because there are so many tiny details that can go wrong. There is a good chance that if you do it for the first time you will lock yourself outside the server.

So, do this before you can erase the server, of if you are confortable waiting that your provider will unlock it for you.

Steps:

1. Parameters to change in `/etc/ssh/sshd_config`:

ChallengeResponseAuthentication no

X11Forwarding no

UsePAM no

LogLevel DEBUG3 (this should be added, the parameter is not listed by default)

Save the file, then:

```
service sshd restart
```

2. Add your public key to `/home/myUser/.ssd/authorized_keys`
    - make sure you have put the public key in /home/myUser/.ssd/authorized_keys (not just in the root user folder)
    - make sure your key starts with "the "ssh-rsa" (with a space after it, check the first "s" might be missing ...)
    - triple check the key doesn't break in several lines
    - do `chmod 700 ~/.ssh` on the home folder

3. What will probably happen:

Your private key will probably not be recognized the first time because of some problems above not completely fixed.

Keep trying to log with your SSH key. To find the cause of your issues, inspect the log for auth operations:

```
tail -f /var/log/auth.log
```

Some useful answers to questions from developers lost in making SSH keys works:

- A recap of the steps: http://askubuntu.com/a/306832
- On debugging (saved my life): http://stackoverflow.com/a/20923212/798502

4. Finally, when the login via SSH keys work, only then can you disable login via passwords:

In `/etc/ssh/sshd_config`, you can disable password authentification:

PasswordAuthentication no

Do again: `service sshd restart`

Now only connecions via a public / private key is possible.

Setting up a firewall

A firewall gives you control on what can enter and leave your server.

ip tables

The rules for setting up ip tables are logical but quite complicated. Using an ip tables generator could help.

But there is an even easier alternative.

better: uncomplicated firewall

Following @mgilbir's advice, I'll use ufw: a linux package for "uncomplicated firewall". To install it:

```
apt-get install ufw
```

The firewall is now installed, but is is not active yet.

We add a rule to block all incoming traffic, except for SSH connections through the port we defined:

```
ufw default deny incoming
ufw allow 1234/tcp
```

Now, we can activate the firewall

```
ufw enable
```

Use anti-intrusion defenses and audit systems

Psad

INFO

this part builds on: http://www.pontikis.net/blog/psad-install-config-debian-wheezy

Psad is an app which bans users which scan ports. Before installing it, we need to make sure the firewall logs traffic:

```
iptables -A INPUT -j LOG
iptables -A FORWARD -j LOG
```

Then we install Psad:

```
apt-get install psad
```

Now we configure Psad by modifying this file:

```
vi /etc/psad/psad.conf
```

Possible values for some interesting parameters (and the source for this section), are here:

http://www.pontikis.net/blog/psad-install-config-debian-wheezy

Then we must edit this file to add the address of the server to the whitelist:

```
vi /etc/psad/auto_dl
```

where I put just 2 values:

```
127.0.0.1    0;  # localhost
xx.xx.xxx.xxx    0; # Server IP (replace xx.xx.xxx.xxx by your actual server IP)
```

fail2ban

This is an app which bans users which fail to login after a number of times - typically bots trying to break in.

fail2ban can actually scan logs from a list of apps you decide (MongoDB, Apache server, GlassFish, etc.) and ban ips mentioned in logs showing a failed access. You need to setup a regex rule specific for each log format, though.

I'll cover it later, when I'll have MongoDB and GlassFish installed.

Documentation on failtoban: http://www.pontikis.net/blog/fail2ban-install-config-debian-wheezy

Lynis

This is an application running on your machine, generating security audits and making suggestions.

Install it:

```
apt-get install lynis
```

Run it: (from any directory)

```
lynis audit system
```

The report will appear on screen (hit Enter to move on), and in this file:

```
/var/log/lynis-report.dat
```

the end

Author of this tutorial: Clement Levallois

All resources on linux security: https://seinecle.github.io/linux-security-tutorials/