Server hardening

What i am doing or what i want to do to harden my server.

Prerequisites

- A fresh Ubuntu 18.04 installation
- Root privileges

First minutes

The first steps after installing a new server to make sure nobody can capture it and use it in a way it was not intended. Make sure you work as fast and correct as possible until you reach *BREAKTIME*. That should not consume too much time and then you can think about what you want to install afterwards.

What we will do

- Create a new user with sudo rights
- Test login with the new user
- Configure SSH (deactivate root login, password login, [optional] change Port)
- Install fail2ban (Configure short-term ban and a long-term ban)
- Update (Just in case)
- Install and configure UFW (Just allow used ports like ssh)
- BREAKTIME

Create a new user with sudo rights

Create new User (at the server)

```
adduser sammy
```

Give sudo-rights (at the server)

```
usermod -aG sudo sammy
```

Generate SSH key (at own computer)

```
ssh-keygen
```

Copy the public key to server (at own computer)

```
ssh-copy-id sammy@your_server_ip
```

Test login

```
ssh sammy@serverip -p PORT
```

Configure SSH

```
sudo nano /etc/ssh/sshd_config
```

- Disable root login: `PermitRootLogin no`
- Disable password login: `ChallengeResponseAuthentication no` and `PasswordAuthentication no`
- Don't allow empty passwords: `PermitEmptyPasswords no`

change SSH Port

It's security by obscurity and not actually needed. It will reduce the amount of automated scans that reach your ssh-port but is not really something to secure the server. Just to keep your log files clear.

restart sshd

```
sudo systemctl restart sshd
```

Install fail2ban

```
sudo apt install fail2ban
```

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

and configure (1 short-lock for 24 hours, one for 1 week block)

- https://www.booleanworld.com/protecting-ssh-fail2ban/
- https://blog.shanock.com/fail2ban-increased-ban-times-for-repeat-offenders/

```
sudo nano /etc/fail2ban/jail.local
```

```
#
# JAILS
#

#
# SSH servers
#

[sshd]
```

```
# To use more aggressive sshd modes set filter parameter "mode" in
jail.local:
# normal (default), ddos, extra or aggressive (combines all).
# See "tests/files/logs/sshd" or "filter.d/sshd.conf" for usage example and
details.
mode    = normal
port    = ssh
logpath = %(sshd_log)s
backend = %(sshd_backend)s
# input by stefan
# one day
findtime = 5400 ;1.5 hours
maxretry = 5
bantime = 86400 ;1 day

# input by stefan, longterm ban
# 30 attempts over 3 days result in a 1 week ban
[sshlongterm2]
port      = ssh
logpath   = %(sshd_log)s
banaction = iptables-multiport
findtime  = 259200 ;3 days
maxretry  = 10
bantime   = 604800 ;1 week
enabled   = true
filter    = sshd

[sshlongterm3]
enabled = true
filter = sshd
findtime = 15552000 ;6 months
maxretry = 15
bantime = 2592000 ;1 month
logpath   = %(sshd_log)s
banaction = iptables-multiport
```

Check with tail

```
tail -10f /var/log/fail2ban.log
```

Update

```
sudo apt update && sudo apt upgrade
```

Install and configure UFW

Install und enable UFW and allow only SSH default [or Enable UFW and disable all inbound traffic from eth0 on all ports except SSH from my local IP (temporary, eventually I allow SSH globally due to

potential for IP changes) and disable all outbound traffic except for port 80.] and for hosted websites port 80 and if you intend to use letsencrypt or somethinglike that port 443 too. UFW does not play well with Docker, keep that in mind.

Important commands for UFW

```
ufw allow APPLICATION
ufw enable
ufw disable
ufw status
```

Important ports for the first use

```
ufw allow ssh
ufw allow http
ufw allow https
```

BREAKTIME

Pause, drink a cup of coffee, think about what you are going to do next and plan a little bit. The server now has some basic security.

After installing

Installing Tools

Tools

Install tools you want(vim, tmux, htop, nmap, sysstat, net-tools) or / and take look at https://github.com/n1trux/awesome-sysadmin

Mailserver

Install and configure mailserver (postfix with s-nail?) for automated messages from unattended upgrades or from other services.

Unattended Upgrades

Automatically just updates security-relevant updates. Can also update all updates, if you want. Can also send autmated messages if a mailserver is installed. A sugestion is to send on every update at first and change the setting later to “justOnError” in /etc/apt/apt.conf.d/50unattended-upgrades

(multiple recipients separated with a komma)

Logrotate

Configure logrotate to rotate with dates instead of rolling numbers (easier for archive/backup)
https://linoxide.com/linux-how-to/setup-log-rotation-logrotate-ubuntu/

Logwatch

daily mail set up

Time-Related

Configure time-related stuff (tzdata, install ntp, setting the time zone to UTC)

Disable unrequired services

Disable any and all services that are not required for the purpose of the box, bind others to localhost, unless they need to listen on public interfaces. This reduces attack vectors.

Worth checking out

zsh (instead of bash), glances, rsync, debug tools (lsof, gdb, iotop, slurm, strace), Enable Byobu, etckeeper (etckeeper init, etckeeper commit -m initial), vnstat, linuxbrew, git and checkout the dot files from git, learn Ansible (install python-minimal for ansible), chef bootstrap

Check server

- https://www.ssllabs.com/ssltest/analyze.html

Further hardening

AppArmor / SELinux

AppArmor (Ubuntu, Debian) or SeLinux (Fedora, CentOS, GenToo, OpenSUSE, possible to install on Ubuntu and Debian)

- https://help.ubuntu.com/community/AppArmor
- https://www.kuketz-blog.de/apparmor-linux-haerten-teil3/

Short introduction

Apparmor is one of the best tools you can possibly use for hardening security. Generally speaking “deny by default” security policies where you explicitely state what a process can or can't do (all in one simple file) work very well and have less chance for human error. So what's the point? AppArmor offers a reasonably high level of assurance that any unpublished or future exploits in your server software won't be catastrophic. For instance if someone finds a 0day in Apache, a good AppArmor profile will prevent them from defacing your websites or rooting your box.

AppArmor ships by default with Ubuntu (probably debian too) along with profiles that secure notoriously insecure daemons like BIND. What's even better is that you can run everything as root and not have to put up with all the headaches of user permissions, alternate users, and chrooting.

Here's a simple tutorial of the steps you can take to secure a system:

- See what profiles are installed and currently enabled: `sudo apparmor_status`
- Find a list of server processes running that are insecured: `sudo aa-unconfined`
- Let's say you want to secure the program foobar. You can start by generating an initial profile: `sudo aa-genprof /usr/bin/foobar`
- Set it to “complain” mode (security is not yet enforced: `sudo aa-complain /usr/bin/foobar`
- Next you would run foobar and get it to do a bunch of stuff. You can now run the following program to generate a security profile automatically! `sudo aa-logprof`
- You can also edit the new profile manually if you want: `sudo nano -w /etc/apparmor.d/usr.bin.foobar`
- Once you're confident your security profile is correct, set it to enforce mode: `sudo aa-enforce /usr/bin/foobar`

Enjoy.

Also when compiling programs you can set a bunch of security flags. On many systems like Ubuntu and Gentoo I think these are enabled by default. If not you can do the following:

```
./configure CFLAGS="-fstack-protector-all -D_FORTIFY_SOURCE=2 -Wformat -
Wformat-security" LDFLAGS="-rdynamic -Wl,-z,relro"
```

su and sudo

Deactivate sudo for your account, check if login for root via ssh is deactivated

As for people telling you to sudo everything, I'm inclined to say “nah” I don't enable sudo on internet-facing machines. I use a strong password for root and su over to the account when I need to. No reason to hand over root access to someone who got into my user account.

I don't see how you consider su over sudo to be lazy. The default, “add user to wheel, enable sudo” is pretty garbage. If your user credentials are compromised, so is root. That's like advocating to add a user to the docker group on a container host. No one aside from myself has advocated for privilege

separation or the use of strong ACLs/group policies.

The cult of “sudo all the things” is irresponsible. If you need root privileges, use root. Everything else should be covered under group policy.

Check file permissions

After this is done ensure your file permissions are correct by this command

```
find / -perm -2 ! -type l -ls
```

once this is done change permissions to remove world writable status from files that do no need this, see man chmod.

Check open ports

```
netstat -lntp
```

Stuff

To integrate later

- https://github.com/imthenachoman/How-To-Secure-A-Linux-Server
- DOD STIG checklists
- https://www.cisecurity.org/cis-benchmarks/
- https://www.nsa.gov/ia/mitigation_guidance/security_configuration_guides/ (Check for new link!)
- Centers for Internet Security config standards
- http://greenfly.org/talks/security/simple_hardening.html
- https://www.cyberciti.biz/tips/linux-security.html
- https://linux-audit.com/ubuntu-server-hardening-guide-quick-and-secure
- https://www.reddit.com/r/linuxadmin/comments/arx7xo/howtosecurealinuxserver_an_evolving_howto_guide/?sort=top
- https://github.com/imthenachoman/How-To-Secure-A-Linux-Server#custom-jails
- https://www.cisecurity.org/cis-benchmarks/