

# Linux, day 14



# Objectives covered

Objective	Summary	Boek
3.2	Given a scenario, configure and implement appropriate access and authentication methods.	16
3.3	Summarize security best practices in a Linux environment.	19
3.5	Given a scenario, implement and configure Linux firewalls.	18

# LABS: PAM

# Lab preparation!

- You will need a Debian-derivative VM.
  - If you don't have one yet, fire up Vagrant.

```
$ cd ~/Downloads;  
$ mkdir pam-test; cd pam-test  
$ vagrant init bento/debian-10  
$ vagrant up
```

# Lab preparation!

- You may need to install PAM modules.

```
$ sudo apt install -y libpam-modules-bin
```

# Let's add tallying

- Make a backup of `/etc/pam.d/common-auth`.
- Then edit the file.
  - Add this line, **above** `auth ... pam_unix.so`:

```
auth    required    pam_tally2.so deny=3 unlock_time=60
```

- Restart SSHD.

# Let's add tallying

- Test with the "*vagrant*" (or other) account:
  - Do three failed SSH logins block a further login?
  - Run: "*sudo pam\_tally2*" to check.
    - If this fails, there are not logged account locks.
  - Does the block reset after 60 seconds?
  - Can you login with SSH after the 60 seconds?

# Let's add password complexity

- Check that "pam\_pwquality.so" is on your system.
  - If not, install it:

```
$ sudo apt install libpam-pwquality
```



# Let's add password complexity

- Make a backup of *`/etc/pam.d/common-password`*.
- Then edit the file.
  - Add this line, **above** *`password ... pam_unix.so`*:

```
password    required    pam_pwquality.so minlen=10
```

- If missing **add** *`use_authok`* at the end of *`pam_unix.so`*.

# Let's add password complexity

- Test with the "*vagrant*" (or other) account:
  - Login with their current password.
  - Try changing the password with a 4-letter word.
  - Try other weak passwords.

# LABS: What will we do??



# The next three labs...

- All three labs follow the same pattern:
  - Set the firewall to block all traffic by default.
  - Start a service.
  - Prove that traffic is blocked.
  - Open the firewall.
  - Prove that traffic is now open.

# LAB: Uncomplicated Firewall



# Warning: prior work?

- If you already worked on this VM before,
  - And if you worked with *iptables* on it...
  - IPTables will fight your UFW.
  - Results will be weird!



# Lab preparation

- You will need two VMs, in the same network.
  - It's best if you use our Fedora and Ubuntu hosts.
  - Or rebuild it using the known *Vagrantfile*.

# Enabling UFW

- Once the VM is up, login. Then:

```
$ sudo ufw status
```

```
$ sudo ufw enable
```

```
$ sudo ufw app list
```



# Allowing SSH, before closing

- Let's not lock ourselves out of the VM.

```
$ sudo ufw allow openssh
```

```
$ sudo ufw default reject
```

```
$ sudo ufw status verbose
```

# Setting up a website

- Here's a quick test

```
$ sudo apt install lighttpd
```

```
$ sudo service lighttpd start
```

```
$ curl http://localhost:80
```

# Setting up a website

- Can you reach the site from your host OS?
  - e.g. "*curl http://ubuntu*" from your Fedora VM?
  - e.g. "*curl http://localhost*" on Ubuntu?

# Setting up a website

- Let's open the firewall!

```
$ sudo ufw app list
```

```
$ sudo ufw allow "Lighttpd Full"
```

- Can you reach the site from the other host now?

# LAB: firewallld

# Setup

- You should already have a **Fedora** box,
  - Plus another one inside the same "NATNetwork".
  - This will help us test "*httpd*" on the Fedora host.

# Starting the web server

- After starting the server, can you reach it locally?

```
$ sudo yum install -y httpd
```

```
$ sudo systemctl start httpd
```

```
$ curl http://localhost
```

# Checking on firewalld

```
$ sudo systemctl list-unit-files \
  | grep firewall
```

# Not running? Start it :) Then continue:

```
$ sudo firewall-cmd --state
```

```
$ sudo firewall-cmd --get-active-zones
```



# Enabling some block rules

```
$ sudo firewall-cmd --set-default-zone public
```

```
$ sudo firewall-cmd --get-services
```

```
$ sudo firewall-cmd --list-services
```

# Testing remotely

- From the other VM:

```
$ curl http://${FedoraIP}
```

- Test whether you can load the test-site on Fedora.

# Opening up the firewall

- Back on Fedora, open the firewall.

```
$ sudo firewall-cmd --add-service=http \
  --zone=public --permanent
```

```
$ sudo firewall-cmd --reload
```

```
$ sudo nft list ruleset
```

# Testing remotely

- From the other VM:

```
$ curl http://${FedoraIP}
```

- Does it work now?

# LAB: iptables

# Setup

- We will continue on the same **Fedora** box.
  - And we'll use the same test host.
- Make a snapshot first!
  - Just so you can easily go back.

# Disabling firewall

- IPtables and firewalld cannot co-exist.

```
$ sudo firewall-cmd --remove-service=http \
  --zone=public --permanent
```

```
$ sudo systemctl stop firewalld
```

- You should now be able to reach the website.

# Checking on IPTables

- We should have a fresh start!

```
$ sudo iptables -L
```

```
$ sudo iptables -A INPUT -m state \  
--state ESTABLISHED -j ACCEPT
```

- This rule makes sure established connections, both in- and outgoing, are allowed.



# Closing things down

- Again, let's allow only SSH

```
$ sudo iptables -A INPUT -p tcp --dport 22 \  
-m state --state NEW,ESTABLISHED -j ACCEPT
```

```
$ sudo iptables -P INPUT DROP
```

```
$ sudo iptables -L
```

# Starting the web server

- It should already be running.

```
$ sudo systemctl start httpd
```

```
$ curl http://localhost
```

- It still works locally, right?

# Testing remotely

- From the other VM, or your host OS:

```
$ curl http://${FedoraIP}
```

- Test whether you can load the test-site on Fedora.
  - Again, this should not work.

# Opening up the firewall

- Back on Fedora, open the firewall.

```
$ sudo iptables -A INPUT -p tcp --dport 80 \
-m state --state NEW,ESTABLISHED -j ACCEPT
```

- Then test again from the outside. Can you get in?

# Saving your changes

- With IPtables, we need to save our current config.

```
$ sudo iptables-save | \
sudo tee /etc/sysconfig/iptables
```

- On Fedora >20, the "*iptables*" service is not installed.
  - So on a reboot it won't load these rules.
  - For our lab, that's fine. Good enough.

# Case 1: NTP server

# Case 1: NTP server

- Assume a company network. You're asked to build an NTP server.
- Build and configuration:
  - On RHEL-derivatives you may need Chrony instead of NTPd.
  - Assume that the NTP server uses the default NTP pool on the Internet.
  - Open the NTP server to the internal network. Also open the firewall.
- Test it as well, from another system.
  - For example "*ntpq \${ServerIP}*" and use the "*/peers*" command.

# Case 2: NFS server



# Case 2: NFS file share

- Add a 100MB storage device to your VM.
- Integrate the disk into LVM. Make
  - A volume group "*data-vg*".
  - And a logical volume "*data-lv*".
- Mount the "*data-lv*" device as "*/data-share*".
- Install NFS server software.
- Share "*/data-share*" through NFS as "*data-share*".

# Closing

# Homework

- Reading:
  - Chapter 20
  - Chapter 21
- Go do:
  - One or more CertDepot "daily tasks".
  - Or the more advanced exercises (see day 11).

# Reference materials

# Resources

- [RedHat's introduction to PAM](#)
- [LinuxJournal's 1997 coverage of PAM](#)
- [Using PAM, NSS and SSSD for LDAP](#) (advanced)
- [Allowing routing/forwarding with UFW](#)