

# Linux, day 14



# Objectives covered

Objective	Summary	Boek
<b>2.1</b>	Authentication	16,19
<b>2.1</b>	System hardening	10,16
<b>2.3</b>	Firewalls	18

# LABS: PAM

# Lab preparation!

- We will use our Ubuntu /Debian VM.
  - You will 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".`
- **Replace** the complete *pam\_unix.so* line, with

```
auth    required    pam_faillock.so preauth deny=3 unlock_time=60
auth    sufficient  pam_unix.so
auth    required    pam_faillock authfail deny=3 unlock_time=60
```

# Let's add tallying

- Make a backup of *`/etc/pam.d/common-account`*.
- Add this line, **at the bottom**:

```
account    required    pam_faillock.so
```

# Let's add tallying

- Test with a dummy account:
  - Do three failed SSH logins block a further login?
  - Run: "*sudo faillock*" to check.
  - 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
```

# Let's add password complexity

- Test with a dummy 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?

- We will use the Ubuntu VM to practice UFW.
- If you already worked with *iptables* on this VM before,
  - IPTables will fight your UFW.
  - Results will be weird!



# Lab preparation

- You will need two VMs, in the same network.
  - Ubuntu will be the server, with UFW,
  - Fedora will be the client.

# Enabling UFW

- Once the Ubuntu 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 systemctl start lighttpd
```

```
$ curl http://localhost:80 # This should work
```

# Setting up a website

- Can you reach the site from your Fedora VM?
  - e.g. "*curl http://ubuntu*" from your Fedora VM?
- The "localhost" connection from Ubuntu should work,
  - But the external connection from Fedora shouldn't.

# 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: firewall

# Lab preparation

- We will work on Fedora to learn FirewallD.
- You will have another VM, in the same network.
  - Fedora will be the server, with *firewalld*,
  - Ubuntu will be the client.

# 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 connections

- Can you reach the site from your Ubuntu VM?
  - e.g. "*curl http://fedora*" from your Ubuntu VM?
- The "*localhost*" connection from Fedora should work,
  - But the external connection from Ubuntu shouldn't.

# 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?
  - It should!

# 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 allows all pre-established connections.



# 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.

# What will we do today?

- ~~Recap~~
- ~~Pluggable Authentication Modules (PAM)~~
- ~~Firewalls~~
- Security best practices
- Extra labs
- Closing: Homework and Q&A

# 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: Time restricting SSH

# Case 2: Time restricting SSH

- Read the documentation for *pam\_time.so*.
- Setup a dummy test account on your VM.
- Configure the SSH daemon and PAM,
  - So this dummy user can only login with SSH,
  - On Monday through Friday, between 0600 and 1700.

# Closing

# Homework

- Reading (some of this are repeats):
  - Chapters 7, 11, 20, 21, 24
- 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](#)