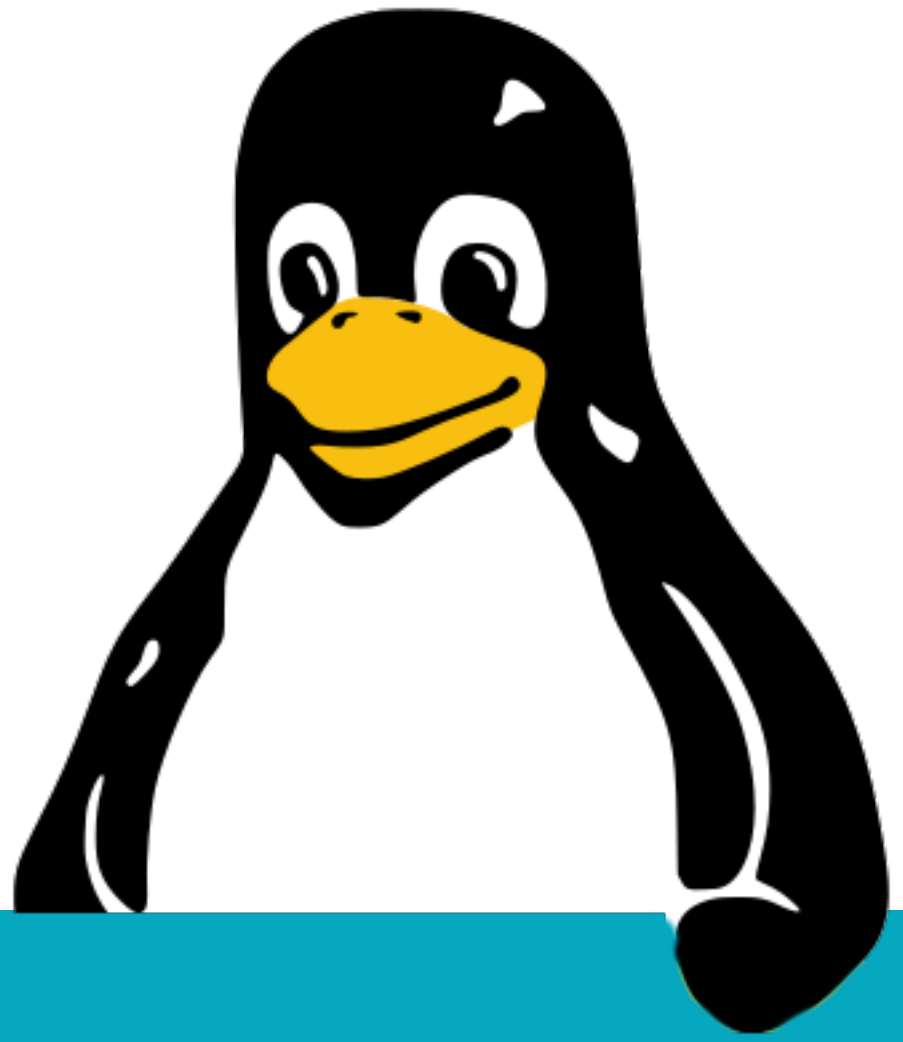
Linux, day 14



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.so 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 bad SSH logins with a wrong password.
  - Run: "*sudo faillock*" to check for the lock.
  - 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
```

- If asked to override your PAM changes, refuse.

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    requisite     pam_pwquality.so minlen=10
```

- If a *pwquality* line exist, don't add, but edit.

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.

Let's add password history

- Also in: "*/etc/pam.d/common-password*".
- Add this line, **above** *password … pam_unix.so*:

```
password    requisite     pam_pwhistory.so remember=5
```

- If a *pwhistory* line exist, don't add, but edit.

Let's add password history

- You already changed the password for dummy.
  - Check */etc/security/opasswd*
  - This should have dummy's old hash.

Bonus LAB: Apache and PAM

Bonus!

- This is extra work.
  - It goes beyond Linux+ exam content.
  - It probably doesn't fit into our class time.

Lab preparation

- We will work on Ubuntu Linux again.
  - Let's install the required packages.

```
$ sudo apt install -y apache2 \
libapache2-mod-authnz-pam
```

Enable PAM for Apache

- We have installed the module.
  - Let's enable it:

```
$ sudo a2enmod authnz_pam
$ sudo usermod -aG shadow www-data
$ sudo systemctl restart apache2
```

Make a private area

- Let's make a protected directory.

```
$ sudo mkdir /var/www/html/private

$ echo "Secret." | sudo tee \
/var/www/html/private/index.html
```

Apache configuration

```
$ sudo vi /etc/apache2/conf-available/
protected.conf

# The contents are on the NEXT slide
```

Contents for protected.conf

```
<Directory "/var/www/html/protected">
   AuthType Basic
   AuthName "Restricted"
   AuthBasicProvider PAM
   AuthPamService apache2
   Require valid-user
</Directory>
```

Enable the new config

```
$ sudo ln -s /etc/apache2/conf-available/
protected.conf /etc/apache2/conf-enabled/
protected.conf

$ sudo systemctl reload apache2
```

PAM configuration

```
$ sudo vi /etc/pam.d/apache2

# The contents of the file should be:

@include common-auth
@include common-account
```

Testing: anonymous

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

$ journalctl --since "2 minutes ago"

# Curl should get a HTTP 401.
# Journal should show failed access.
```

Testing: failed login

```
# Use your own name and a bad password

$ curl --basic -u tess \
http://localhost/private

$ journalctl --since "2 minutes ago"
```

Testing: successful login

```
# Use your own name and real password

$ curl --basic -u tess \
http://localhost/private

$ journalctl --since "2 minutes ago"
```

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?

How to define a service

- You can make a service for Chrony: */etc/ufw/applications.d/chrony-server*

```
[Chrony]
title=Chrony NTP Server
description=Chrony NTP Server
ports=123/udp
```

LAB: firewalld

Lab preparation

- We will work on Fedora to learn Firewalld.

- This time, 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 <u>Ubuntu VM</u>?
  - 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 firewalld

- 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.

Recap: firewalls

- UFW is very user friendly.
- Firewalld offers flexibility and friendliness.
- IPTables has you hand-writing rules.

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*" and use the "*lpeers*" 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

- Read:
  - Chapter 7, review entirely
  - Chapter 11, review entirely
  - Chapter 20, 573-593
  - Chapter 24, 667-673

Homework

- Go do:
  - One or more exercises from CertDepot exams.
  - 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
- CIS Benchmarks
- Ansible Lockdown (CIS Benchmark automation)