

Linux, day 14



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
```

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.
 - 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=6
```

- Also 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

- If you don't already have a Ubuntu VM,
 - Set one up with *"vagrant init ubuntu/trusty64"*
- Edit the *"Vagrantfile"* and add:
 - *config.vm.network "forwarded_port", guest: 80, host: 8000*
 - Remember the examples in *"008-Vagrantfile"*.
 - Or make the port forward manually once it's up.

Enabling UFW

- Once the VM is up, login with "*vagrant ssh*". 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
```

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?
 - *"curl http://localhost:8000"*
 - Or use your web browser.
- Or from your Fedora VM?
 - *"curl http://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 hosts now?

LAB: firewall

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.
- If you don't have that second VM anymore,
 - Make a quick port forward (like in the last lab).
 - Host 8000, to guest 80.

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
```

Starting the web server

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

```
# Most RHEL-derivatives come with Apache
$ sudo systemctl start httpd

$ curl http://localhost
```

Testing remotely

- From the other VM, or your host OS:

```
$ 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
```

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

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*".

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](#)