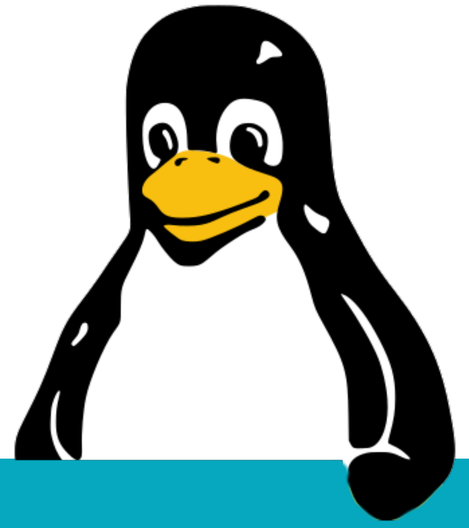
Linux, day 13



Objectives covered

Objective	Summary	Boek
3.1	Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.	15
3.2	Given a scenario, configure and implement appropriate access and authentication methods.	16

LAB: users and more

Sudo lab

- Create new users, "*pete*" and "*support*".
- Make *sudo* rules that allow:
  - The group "*support*" a Bash as root, **with** password.
  - The user "*pete*" passwordless "*ls*", "*cp*" and "*cat*".

Passwd / chage

- Change the "*pete*" user so:
  - He needs 1 day between each password change.
  - His password is valid for 60 days.
- Now expire Pete's password and "su" to his account.
- Now expire Pete's account and "su" to him again.
- Don't forget to UNexpire Pete's account again ;)

See - A passwd / chage reference

FACL

- Create the directory */tmp/demo/*.
  - And add a few test files in there (*test1, test2*).
- Use a **FACL** on the files to make sure that:
  - User "*pete*" can also read/write the files.
  - Group "support" can also read the files.
- Test this!! SU to the relevant users.

Ulimits (advanced)

- Using "*/etc/security/limits.conf*", change "pete".
  - Set hard limit for "*nproc*" to 50, soft to "10".
- Test these limits with a small shell script!
  - Try to start 11 "*sleep 60*" commands simultaneously.
  - Check with "*jobs*".
  - Then up the soft limit to 20 (search "*man bash*" - ulimit)
  - Try to start 21 sleeps at the same time.

SELinux lab

Preparation

- If your Fedora was built with Vagrant, run:

```
$ sudo yum install -y httpd \
  python3 \
  policycoreutils-python-utils

$ pip install http.server
```

The "before"

- Login (ssh) to your Fedora host **as "pete"** and check:

```
$ sestatus                # SELinux should be active
$ id -a; id -Z
$ ls -al /var/www/html
$ touch /tmp/bla
$ python -m http.server   # Exit with ^C
```

Setting an SELinux context

- Use **your own account** to reconfigure "*pete*".
  - Pete will lose their "*unconfined*" context.

```
$ sudo semanage login -a -s user_u pete
$ sudo semanage login -l

$ sudo restorecon -Fr /home/pete
$ sudo ls -alZ /home/pete
```

The "after"

- **Logout** and SSH back in **as user "pete"**.
- Check the following:

```
$ id -a; id -Z
$ ls -al /var/www/html
$ touch /tmp/bla2; ls -alZ /tmp/bla2
$ python -m http.server
```

Checking the logs

- Use **your own account** to check for alerts:

```
$ sudo journalctl -t audit \
    --since "10 minutes ago"
```

SELinux lab 2

Preparation

- Make sure that your VM has "*httpd*" installed.

The "before"

- Let's make a tiny, dynamic website.
  - Create "*/var/www/cgi-bin/test.pl*":

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "Hello, World.";
```

The "before"

- Make sure it works.

```
$ sudo chmod +x /var/www/cgi-bin/test.pl
$ sudo systemctl start httpd

$ curl http://localhost/cgi-bin/test.pl
```

Disabling dynamic sites

- Change the SELinux boolean for cgi.

```
$ sudo getsebool -a | grep ^httpd

$ sudo setsebool httpd_enable_cgi off
```

The "after"

- Check that the Perl script no longer executes.

```
$ curl http://localhost/cgi-bin/test.pl

$ sudo journalctl -t audit \
  --since "10 minutes ago" | grep -i avc
```

AppArmor lab

Setup

- You will need a Debian-derivative VM.
  - We can use Vagrant to quickly set one up!

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

Assignment

- Install the required AppArmor packages.
  - *apparmor-utils*
  - *apparmor-profiles*
  - *apparmor-profiles-extra*

Assignment

- Check the current status of AppArmor.
- Edit "*/etc/apparmor.d/usr.sbin.tcpdump*":
  - Put "*deny*" in front of the *network* lines.

- Reload the policy:
  - *sudo apparmor_parser -r /etc/apparmor.d/usr.sbin.ping*

Assignment

- *sudo aa-audit usr.sbin.tcpdump*
- *sudo aa-complain usr.bin.tcpdump*
- Run:
  - tcpdump
  - sudo tcpdump # exit with ctrl-C
- *sudo aa-enforce usr.bin.tcpdump*
  - Try *"sudo tcpdump*" again.

Assignment

- Check "*sudo aa-status*" for *tcpdump*.
- Check *journalctl* for recent AA messages.

- Restore the profile to its original state.
  - And reload with the parser.

SSH recap lab

Setup

- Ensure that you have two Linux VMs.
- And that you have an account on both.

Assignment 1: new key

- Double-check that SSHd runs on both servers.
- Generate a new key pair on one of the accounts.
  - Make it type ECDSA, with a password.
  - Setup its pub.key for authentication on the other VM.
  - Test your SSH key authentication.

Assignment 2: ssh-agent

- Start "*eval $(ssh-agent)*".
- Add / load the private key you generated into the running "*ssh-agent*", with the "ssh-add" command.
  - This should ask your password once.
- Try SSH-ing to the other VM again.
  - This should not ask your password.

Assignment 3: restrict access

- Reconfigure "*sshd_config*" on one of the VMs,
  - So it will only allow group "*sshusers*" to login.
  - Hint: *AllowUsers, AllowGroups*
- Give your own account the new group "*sshusers*"
- Restart the SSH daemon and test that you can login.
  - Also make sure that another user cannot.

Closing

Homework

- Reading:
  - Chapter 16
  - Chapter 18
  - Chapter 19

Homework

- Go do:
  - One or more CertDepot exercises.
  - Or the more advanced exercises (see day 11).
  - Practice exam questions.
  - Other practical fun!

Reference materials

Resources

- CertDepot.net has daily Linux tasks.
- Github project with RHCSA practice exams.
- The anatomy of Docker
- A passwd / chage reference
- PluralSight: access control models

Resources

- CentOS documentation for SELinux
- The full Redhat guide to SELinux (deep & awesome)
- Detailed SELinux example for Postgres
- SELinux troubleshooting example
- Understanding SELinux policies
- No! Don't turn off SELinux