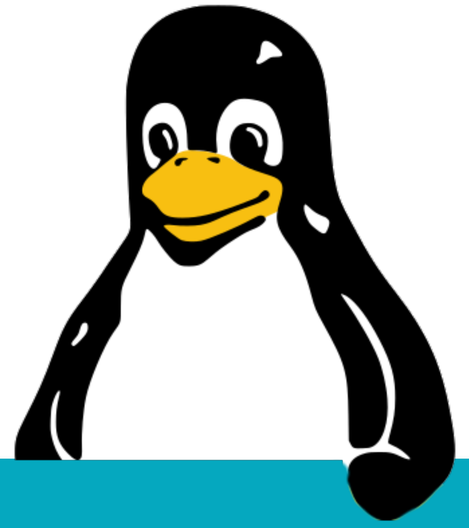
Linux, day 15



Objectives covered

Objective	Summary	Boek
4.1	Given a scenario, analyze system properties and remediate accordingly.	20
4.2	Given a scenario, analyze system processes in order to optimize performance.	21

LAB: PKill

Assignment

- Write a shell script to start 15x "*sleep 120*".
- Use "*pkill*" to kill all of them in one go.

LAB: Swap space

Preparations

- We made extra disk devices for Fedora a while back.
  - If you still have those, unused, use one of those.
  - If they are in use, make a new device of 100MB.

- The next slide assumes "*/dev/sdc*".
  - Adjust for your situation!

Checking and expanding swap

- Make the (new) device into swap space.
  - Use "*mkswap*" and "*swapon"*.
  - Verify swap space with "*free"* before and after.
  - Add the swap device to "*/etc/fstab*" and reboot.

- After the lab, remove from fstab again,
  - And remove with *"swapoff*".

LAB: System performance

Installing sar

- Install the "*sysstat*" package.
- Check where the "*sa1*" and "*sa2*" scrips were added.
  - You will need the path, to add into *cron*.
  - So far I've seen:
    - */usr/lib/sa/, /usr/lib/sysstat/, /usr/lib64/sa/*

Checking sar cronjobs

- Check */etc/cron.d*.
  - Are there config files for "*sysstat*"?
  - Any other job files that have "*sa1*"?

- **<u>If these cron-jobs exist, skip the next slide.</u>**

Manually creating cron jobs

- Edit the "root" crontab (*sudo crontab -e*).
  - Add *sa1* and *sa2*. For example (check the path):

```
# Collect measurements at 10-minute intervals
*/10 * * * * /usr/lib/sysstat/sa1

# Create daily reports and purge old files
0 0 * * * /usr/lib/sysstat/sa2 -A
```

Manual testing

- Run *sa1* a few times, manually.
- Run *sa2* once.
- Go to "*/var/log/sa/*" (Fedora). Check the files there.
  - On Ubuntu the path is "*/var/log/sysstat*".
  - You should at least have one report.

Querying reports

- Query the "sar" file, for example (check the name).

```
# sar -u -f /var/log/sa/sa03

# sar -r -f /var/log/sa/sa03

# sar -d -f /var/log/sa/sa03
```

LAB: Storage issues

Assignment 1: lsof

- Can you find:
  - Which processes currently access "*system.journal*"?
  - Which processes use files from "*/etc/*"?
  - Which files and resources are used by "*sshd*"?
    - Can you spot port 22?

Assignment 2: iostat

- Choose one of your throw-away disk devices.
  - Like "*/dev/sdc*" that was used for swap space.
- Open two terminals.
  - In one keep a running "*iostat /dev/sdc 1*"
  - In the other we will try a few "*dd"* tests.

Assignment 2: Raw devices (Fedora)

- Run:

```
$ sudo raw /dev/raw/raw1 /dev/sdc
```

- This creates a "raw" device to access the disk.
  - What does that mean? Do you remember?

Assignment 2: Raw devices (Ubuntu)

- Run:

```
$ echo "raw1:sdc" | sudo tee -a /etc/raw

$ sudo mkdir /dev/raw; cd /dev/raw
$ sudo mknod raw1 c 162 1
$ sudo modprobe raw

$ sudo raw /dev/raw/raw1 /dev/sdc
```

Assignment 2: iostat

- The following need "*root*" (use *sudo* or *su*).
- Compare speed, throughput and iostat activity.

```
# dd if=/dev/urandom of=/dev/sdc bs=1M count=50
# dd if=/dev/urandom of=/dev/raw/raw1 bs=1M count=50
# dd if=/dev/urandom of=/dev/sdc bs=1024 count=50000
# dd if=/dev/urandom of=/dev/raw/raw1 bs=1024 count=50000
```

- Can you tell what is happening and why?

LAB: Network issues

Assignment 1

- Start a *netcat* listener on port 8080.
- Use *netcat* in a second shell to connect to port 8080.
  - Type a bit of text. Does it arrive?

- Try a file transfer with netcat (example here).
- Try to access a "bind" shell with netcat (example here).

Assignment 2

- Let's see how you can connect to www.google.com.
  - Compare "*getent hosts*" and "*nslookup".*
  - Can you ping the host?
  - Do a *traceroute.* Can you see where you leave ITVitae?
  - What does "*mtr --report www.google.com*" tell you?

Assignment 3

- Start a web server on your VM.
- Start a *netcat* listener on port 3389.
- Run an *nmap* scan from your other VM.
  - What services do you find?
  - Can you access both services?
  - Are there any firewalls in the way?

LAB: Extra work, DVWA

Introduction

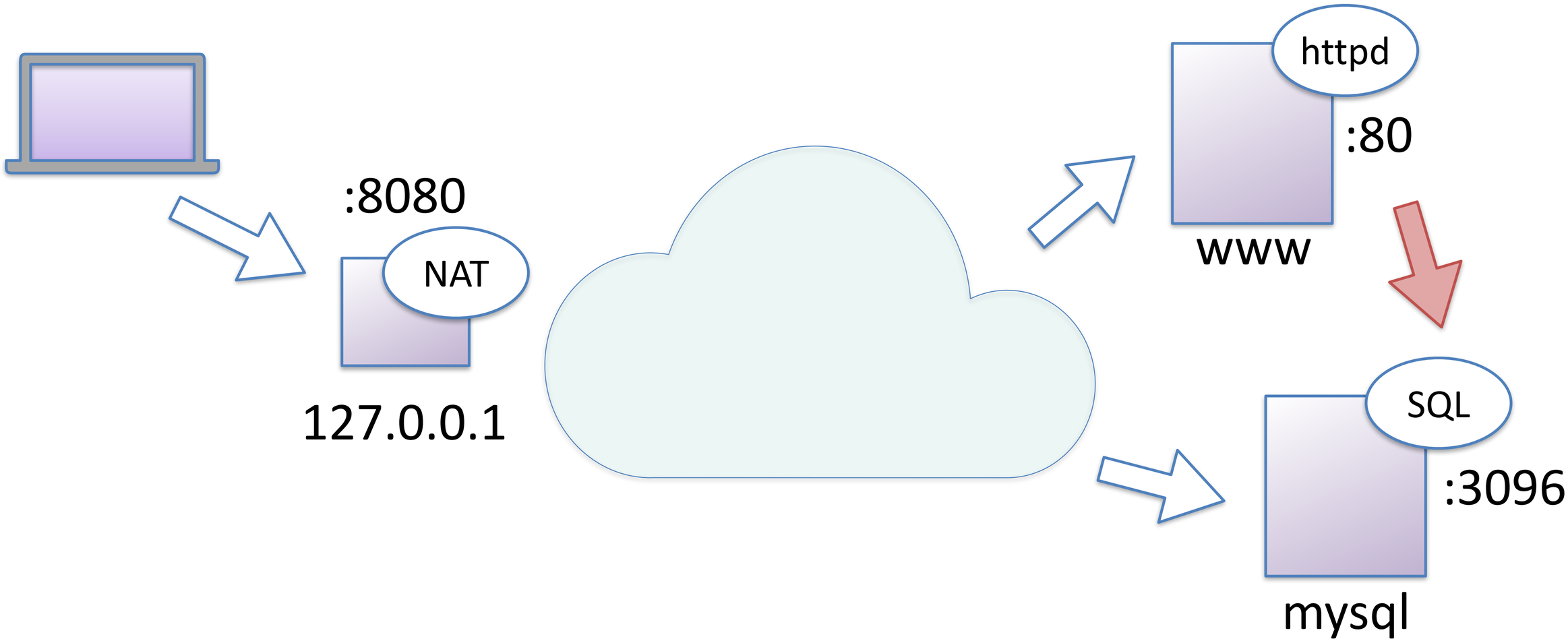
- *D*mn Vulnerable Web App*
  - A learning tool for InfoSec students.
  - A highly vulnerable web application to exploit.

- Vulnhub has plenty more like these.
  - But we're going for a challenge.

What we'll do

- Normally DVWA runs on one host / VM.
- We will use one *Vagrantfile* to define **two** VMs.
  - In a shared network segment, with fixed IPs.
  - One will run MySQL / MariaDB,
  - The other Apache and the web app.

The end result



Don't worry!

- Take things step by step! Don't get flustered. :)
- Remember!
  - Disable the auto-update of vbox guest additions.
  - If Vagrant provisioning fails, no need to rebuild.
  - You can re-run "*vagrant provision*".
  - If you're blocked, open the firewall.

Resources

- *"015-Vagrantfile-start-here"* is your **starting point**.
  - Yes, there is also "*015-Vagrantfile-spoilers*".

- The DVWA Github page has setup guides and wiki.
- MySQL have a getting-started guide.

Step 1: Vagrant

- Adjust the example configuration:
  - Run generic/centos7
  - Just make sure networking works.
- One VM should install MySQL
- The other should install Apache and git-clone DVWA.
- The DVWA Github page lists all required packages.

Step 1: Vagrant

- One VM should install MySQL
- The other should install Apache and git-clone DVWA.
  - The DVWA Github page lists all required packages.
  - Don't forget the port forward for 80.

Step 2: Basic networking

- Boot both VMs and login to both.
- Can you confirm that they can communicate?
  - Can you SSH between the two hosts?
  - Can you connect between the hosts with netcat?
- If it doesn't work, let's troubleshoot!

Step 3: MySQL basic checks

- On the database host, verify that MySQL was installed.
  - Or MariaDB of course... depending on your distro.
- Make sure the DB software starts (also at reboot).
  - Can you connect to the database?
  - Check the MySQL getting-started guide.

Step 4: Setup MySQL database

- Follow the DVWA database setup instructions.
- You will need to create:
  - A database
  - A user (don't use"*@localhost*", use "@'%'")
  - Access privileges for the user.
    - From both localhost AND the webserver host!

Step 5: Testing MySQL

- Can you login to the DB with the new account?
  - Localhost: *mysql -u dvwa -p -h localhost dvwa*
  - From webserver: *mysql -u dvwa -p -h mysql dvwa*

- If the first fails, we troubleshoot MySQL.
- If the second fails, let's check networking!

Step 6: Apache setup

- Make sure that Apache (httpd) starts at boot.
- After starting, can you pull http://localhost with curl?
  - Can you also access the page from your host OS?
    - http://localhost:8080
    - If not, did you set up the port forward?
  - And from mysql? http://www/login.php

Step 7: DVWA install

- Clone the DVWA Git repo to "*/var/www/html/".*
- Edit "*/var/www/html/config/config.php*".
  - Follow the setup guide on Github.
  - At least you should setup the DB connection info.

Closing

Homework

- Reading:
  - Chapter 22
  - Chapter 24

Homework

- Go do:
  - Get DVWA up and running on two VMs.
  - The lab describes all the steps needed.
  - You can first try each step separately or manually.

Reference materials

Resources

- Linux load averages, solving the mystery
- Troubleshooting high load averages
- System Activity Reporter (sar)
- Quick sar explanation
- IOPing tutorial
- In-depth: redis.io performance improvements

Resources

- File transfer with netcat
- Netcat shells, bind and reverse
- Netcat cheatsheet
- CloudFlare: what is MTR?
- Linux ate my RAM!
- Deepdive: Interpreting iostat output