Yookiterm – http://exploit.courses

Yookiterm: Exploit.courses
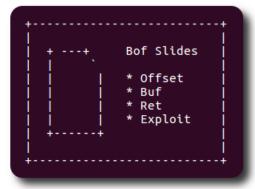
exploit.courses

What?

This website provides an interactive exploit development learning area. Every user has access to his own, personal Linux container. The container can be x32 and x64, with and without ASLR - and even 32 bit and 64 bit ARM.

```
+---------------------------+     +---------------------------+     +---------------------------+
|                           |     |                           |     |                           |
|   # Exploit Writup        |     | root@container:~# id      |     | + ---+      Bof Slides     |
|                           |     | uid=0(root) gid=0(root)   |     | |    |  `                 |
|   ## Step by step         |     |                           |     | |    |    |  * Offset       |
|                           |     | root@container:~# ls      |     | |    |    |  * Buf          |
|   ## Debugging            |     | challenges                |     | |    |    |  * Ret          |
|                           |     |                           |     | |    |    |  * Exploit      |
|   ## Exploit             |     | root@container:~# gdb     |     | +------+                   |
|                           |     | (gdb)                     |     |                           |
|                           |     |                           |     |                           |
+---------------------------+     +---------------------------+     +---------------------------+
```

You dont need to have anything else then a modern browser. Login, select Challenges, and start hacking!

Just want to play around? Start your container without a challenge in the Containers tab.
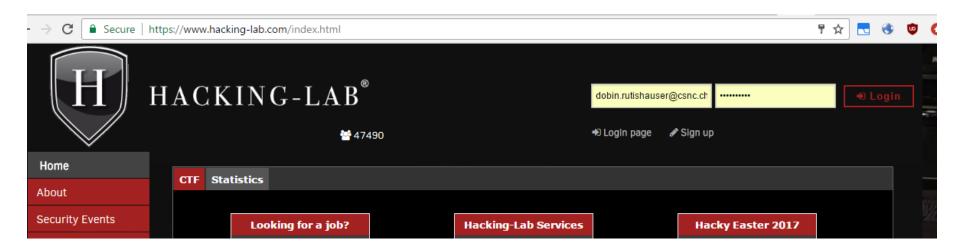
How?

In short: KVM + LXD.
Together with websockets and xterm.js.
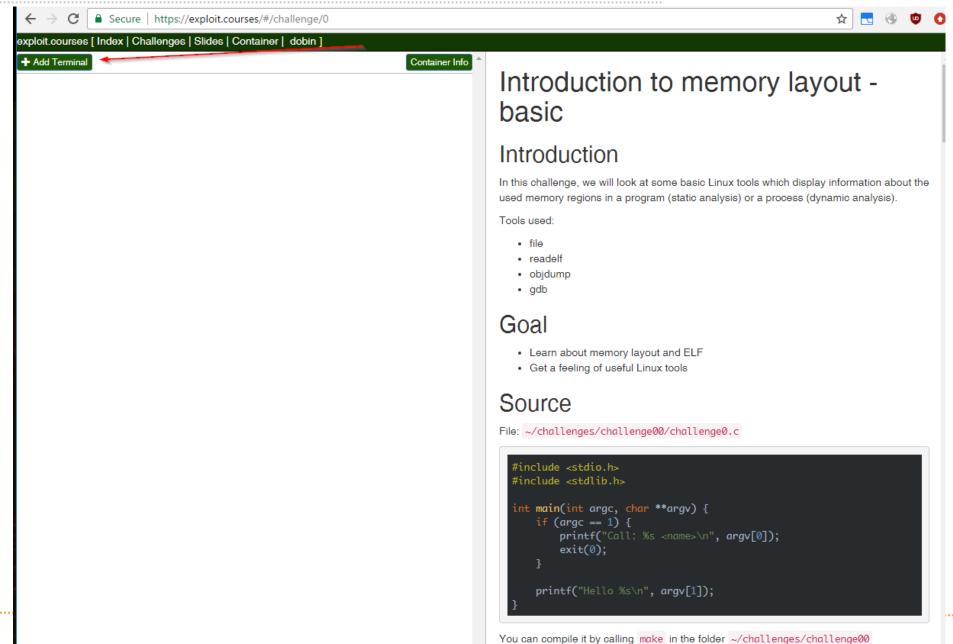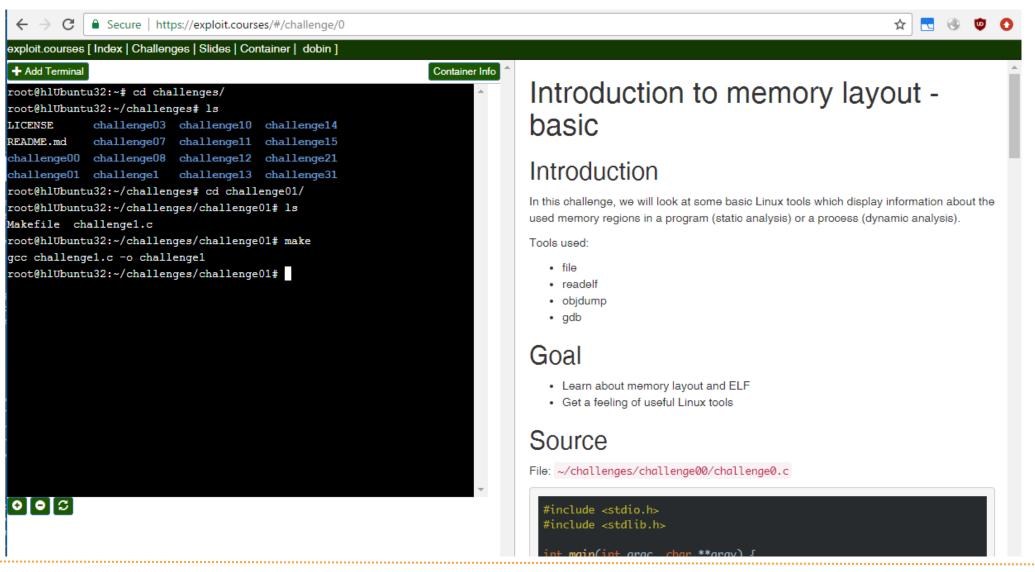Glued together with AngularJS and GO.

Yookiterm: Exploit.courses

exploit.courses: Challenges

exploit.courses [Index | Challenges | Slides | Container | dobin]

ID	Title	Description	Arch	Bits	ASLR
0	Introduction to memory layout - basic	Static and dynamic analysis of an ELF binary with Linux command line tools	intel	32	✔
1	Introduction to memory layout - advanced	Research on where the different types of variables are stored in an ELF file	intel	32	✔
3	Introduction to shellcode development	Create a basic running shellcode (print to console)	intel	32	✖
7	Function Call Convention in x86 (32bit)	Analysis of function calling (gdb static and dynamic analysis)	intel	32	✖
8	C buffer analysis	Analysis of out-of-bound read with gdb	intel	32	✖
10	Simple Buffer overflow	Overwrite local variables on the stack to bypass authentication	intel	32	✖
11	Development of a buffer overflow exploit - 32 bit	How to create a simple buffer overflow exploit	intel	32	✖
12	Development of a buffer overflow exploit - 64 bit	How to create a simple buffer overflow exploit	intel	64	✖
13	Development of a remote buffer overflow exploit - 64 bit	How to create a buffer overflow exploit for a networked server	intel	64	✖
14	Stack canary brute force	How to brute force the stack canary in a remote server	intel	64	✖
15	Development of a remote buffer overflow exploit - 64 bit with ASLR	How to create a contemporary remote buffer overflow exploit	intel	64	✔
20	Introduction to ARM	Basics of the ARM architecture	arm	32	✔
21	Introduction to develop an ARM buffer overflow	Create a simple buffer overflow on ARM	arm	32	✔
31	Heap use-after-free analysis	Analyse a simple use-after-free bug in noteheap	intel	64	✔
50	Introduction to GDB	A reference and playground for GDB	intel	32	✖

exploit.courses: Challenges

exploit.courses [Index | Challenges | Slides | Container | dobin]

+ Add Terminal

Container Info

Introduction to memory layout - basic

Introduction

In this challenge, we will look at some basic Linux tools which display information about the used memory regions in a program (static analysis) or a process (dynamic analysis).

Tools used:

- file
- readelf
- objdump
- gdb

Goal

- Learn about memory layout and ELF
- Get a feeling of useful Linux tools

Source

File: `~/challenges/challenge00/challenge0.c`

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv) {
    if (argc == 1) {
        printf("Call: %s <name>\n", argv[0]);
        exit(0);
    }

    printf("Hello %s\n", argv[1]);
}
```

You can compile it by calling `make` in the folder `~/challenges/challenge00`

exploit.courses: Challenges

exploit.courses: Container

There are three different container you can access:

✦ ASLR (32, 64 bit)

✦ NO ASLR (32, 64 bit)

✦ ARM (32, 64 bit)

32 and 64 bit hosts are usually shared, including data

exploit.courses: Container

Container lifetime: 6 days

Container max lifetime: 12 days

Copy your files with SCP

exploit.courses: Container

The challenge files (vulnerable programs, exploits):

✦ https://github.com/dobin/yookiterm-challenges-files

The challenge writeups:

✦ https://github.com/dobin/yookiterm-challenges