Lab 3

My first CTF: patchme

Patchme binary

```
$ ./patchme
Hello there! Can you patch me up to call my function?
```

This challenge suggests the presence of an hidden function that will give back the CTF flag… we want to reach such a function

Exploring the binary

```
$ cp ./patchme ./patchme_fix
$ r2 -Aw patchme_fix
[0x00400430]> afl
...
0x0040054a    1 21           main
...
[0x00400430]> s main
```

-Aw means open in write mode
And analyze symbols

```
[0x0040054a]> pdf
/ (fcn) main 21
|   main ();
|           ; DATA XREF from 0x0040044d (entry0)
|           0x0040054a      55             push rbp
|           0x0040054b      4889e5         mov rbp, rsp
|           0x0040054e      bf38106000     mov edi, str.Hello_...
|           0x00400553      e8a8feffff     call sym.imp.puts
|           0x00400558      90             nop
|           0x00400559      90             nop
|           0x0040055a      90             nop
|           0x0040055b      90             nop
|           0x0040055c      90             nop
|           0x0040055d      c9             leave
\           0x0040055e      c3             ret
[0x0040054a]>
```

Exploring the binary

```
$ cp ./patchme ./patchme_fix
$ r2 -Aw patchme_fix
[0x00400430]> afl
...
0x0040054a    1 21           main
...
[0x00400430]> s main
```

It seems to suggest we have to patch the "nop" pad by inserting a call to a hidden function

```
[0x0040054a]> pdf
/ (fcn) main 21
|   main ();
|           ; DATA XREF from 0x0040044d (entry0)
|           0x0040054a      55             push rbp
|           0x0040054b      4889e5         mov rbp, rsp
|           0x0040054e      bf38106000     mov edi, str.Hello_...
|           0x00400553      e8a8feffff     call sym.imp.puts
|           0x00400558      90             nop
|           0x00400559      90             nop
|           0x0040055a      90             nop
|           0x0040055b      90             nop
|           0x0040055c      90             nop
|           0x0040055d      c9             leave
\           0x0040055e      c3             ret
[0x0040054a]>
```

Looking for traces

- Let's look for strings within the data section (try i? and iz?)

```
[0x0040054a]> iz
000 0x00001038 0x00601038  53  54 (.data) ascii Hello there! Can you patch me up to call my function?
001 0x0000106e 0x0060106e  10  11 (.data) ascii Thank you!
```

- The "Thank you!" string is not referenced within the main function it could be used in the hidden function
- Let's check any reference to it…
  we find out one at 0x400534 it is marked as data (never referenced)
  - so let's seek at this address and switch to Visual mode

```
[0x0040054a]> axt 0x0060106e
(nofunc) 0x400534 [data] mov edi, str.Thank_you
[0x0040054a]> s 0x400534
[0x00400534]> Vp
```

Exposing the hidden function

- In visual mode we see that at the given reference we have some code

```
[0x00400534 15% 260 patchme_fix]> pd $r @ loc.callme+4 # 0x400534
            0x00400534      bf6e106000          mov edi, str.Thank_you
            0x00400539      e8c2feffff          call sym.imp.puts
            0x0040053e      bf5f054000          mov edi, loc.flag
            0x00400543      e8b8feffff          call sym.imp.puts
            0x00400548      c9                  leave
            0x00400549      c3                  ret
/ (fcn) main 21
|   main ();
|           ; DATA XREF from 0x0040044d (entry0)
...
```

- By switching to cursor mode (“c”) and moving up we see an epilogue

```
[0x00400530 15% 260 (0x0:-1=1)]> pd $r @ loc.callme
            ;-- callme:
            0x00400530    * 55              push rbp
            0x00400531      4889e5          mov rbp, rsp
            0x00400534      bf6e106000      mov edi, str.Thank_you
            0x00400539      e8c2feffff      call sym.imp.puts
            0x0040053e      bf5f054000      mov edi, loc.flag
            0x00400543      e8b8feffff      call sym.imp.puts
            0x00400548      c9              leave
            0x00400549      c3              ret
/ (fcn) main 21
|   main ();
|           ; DATA XREF from 0x0040044d (entry0)
```

Exposing the hidden function

- We can define a function by hitting `d` (define current block as) and then `f` (function)
- A name is given to the new function (we can change it by usind `dr` –rename)
  - `dr hidden`
- Now we can go to substitute the “nop” pad with a call to the new function…
  - Move to the first `nop` and hit A to edit the assembly
  - Then insert call hidden hit enter twice and than q twice to exit radare2

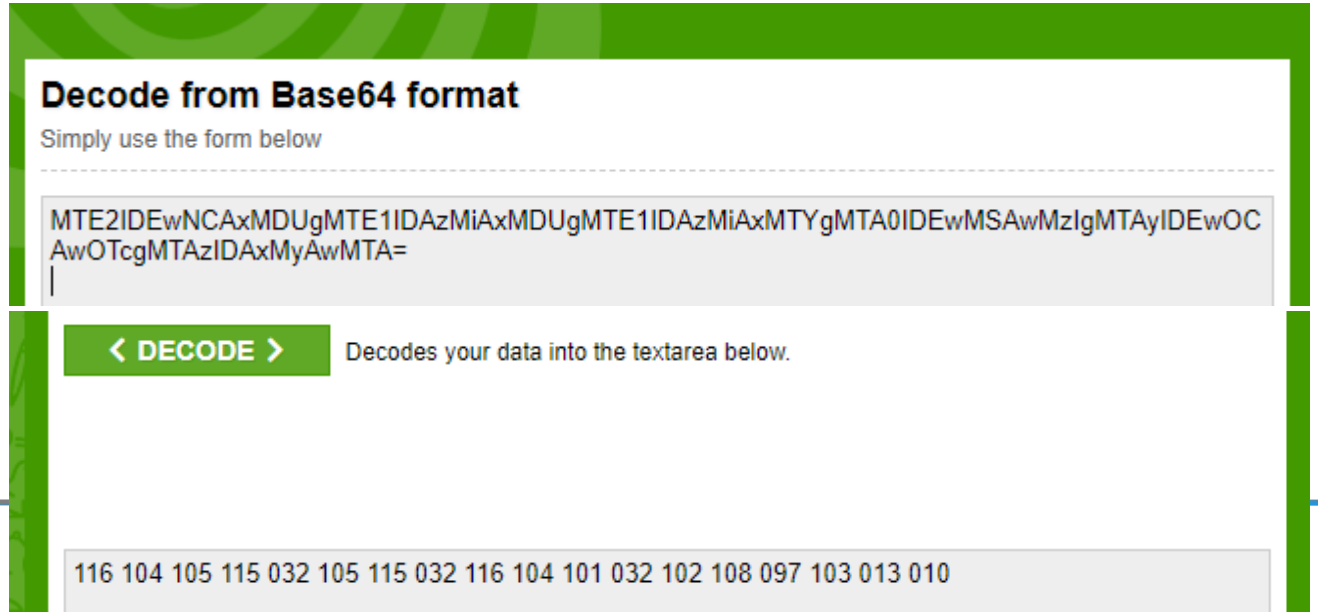
```
5> call hidden
* e8d3ffffff

|           0x00400558      e8d3ffffff      call hidden                ;[1]
|           0x0040055d      c9              leave
\           0x0040055e      c3              ret
            ;-- flag:
            ; DATA XREF from 0x0040053e (hidden)
            0x0040055f      4d54            push r12
            0x00400561      45324944        xor r9b, byte [r9 + 0x44]
```

Let's retrieve the flag

- Launch the patched binary

```
$ ./patchme_fix
Hello there! Can you patch me up to call my function?
Thank you!
MTE2IDEwNCAxMDUgMTE1IDAzMiAxMDUgMTE1IDAzMiAxMTYgMTA0IDEwMSAwMzIgMTAy
IDEwOCAwOTcgMTAzIDAxMyAwMTA=
```

- We reached the hidden function and we were prompted with a strange string
- Let's decode from Base64 (www.base64decode.org)



Here it is!

- We get a sequence of numbers… suppose it is an ASCII encoding

```
116 104 105 115 032 105 115 032 116 104 101 032 102 108 097 103 013 010
```

- We can try to decode the sequence (http://www.unit-conversion.info/texttools/ascii/)

ASCII to text converter

Input data

```
116 104 105 115 032 105 115 032 116 104 101 032 102 108 097 103
013 010
```

Convert: ASCII numbers to text

Output:

```
this is the flag
```