Tu Lam

CS 373 (Defense Against the Dark Arts)

Dr. Bram Lewis

November 29th, 2021

Lab #5 (Hack the Box)

This time in lab No.5, focus on us using the website ***Hack the Box***, a website design for user o learn and try to reverse engineer their way of hacking a software, to learn and see throughout the whole course what can we learn from it. From the perspective of the website, we will also be taking a look at it and see if we could even figure out ourselves and see if there are any additional tools that we need to help us perform the task that we are ask to do.

First, I visited the ***Hack the Box*** website and was greeted with the interface displaying what the website has to offer. Then, I proceed from what the instruction of the lab told me and register for an account to start working on the lab. Once I got all that setup, I was able to start the lab right away. Below is an image of what I was greeted with when login for the first time into Hack the Box website.

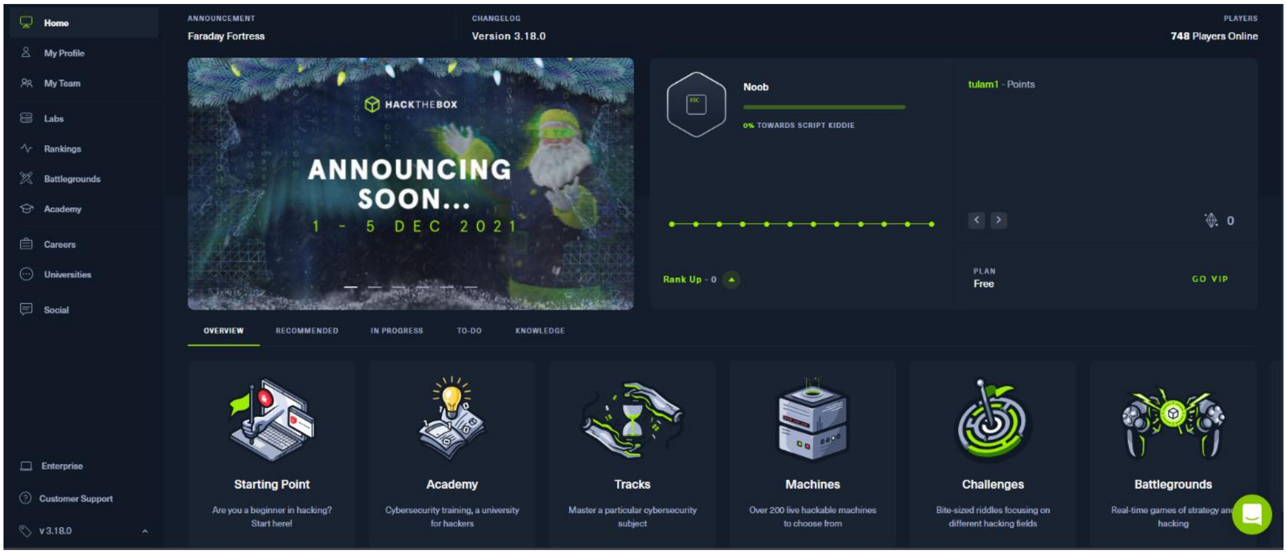
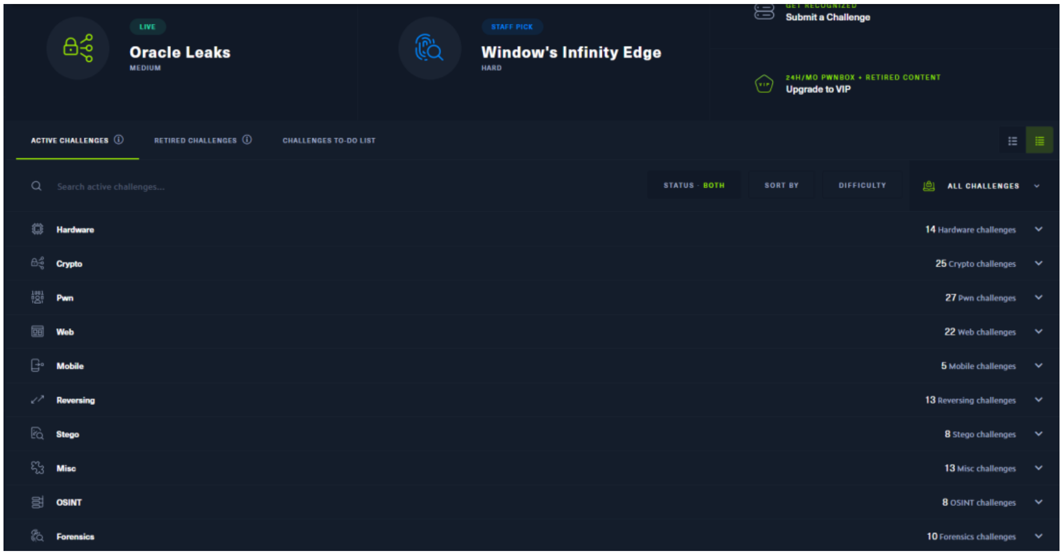


Figure #1: *The layout of the Hack the Box website from user's perspective*

From here, I took a couple of minutes to discover what there is on the website as this is my first time ever using the website. I see that they listed a couple different pages for the user to explore. I see the "***starting point***" which is like a place where it is your first time using the website, then you would go here to start off learning about the website, but also learn about hacking. Then you have "***academy***" is where you have training about the field of cybersecurity. Lastly, I saw the field of "***challenges***" which is a page where you go to do the challenges the website gave you and try and tested out to see if you are able to crack it. This is where I wanted to be as the task that was given to me in this lab is to try out the challenges and see if I was able to solve or not. Once I click on the "***challenges***" field, I was greeted to another page where it listed all the challenges they have on the website. Below is an image of what the page layout looks like. Besides that, as the picture pointed out below, you can see the challenges are divided by their subject of what you wanted to do and which challenges you wanted to tackle on. From here, I took a couple of minutes exploring these fields in the challenge section to see what they have.



LIVE
Oracle Leaks
MEDIUM

STAFF PICK
Window's Infinity Edge
HARD

GET RECOGNIZED
Submit a Challenge

24H/MO PWNBOX + RETIRED CONTENT
Upgrade to VIP

ACTIVE CHALLENGES | RETIRED CHALLENGES | CHALLENGES TO-DO LIST

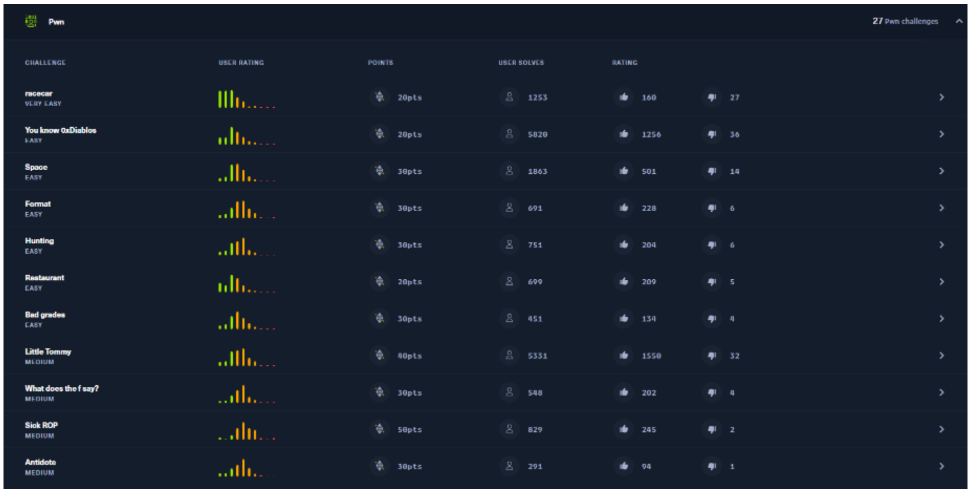
Search active challenges...

STATUS · BOTH | SORT BY | DIFFICULTY | ALL CHALLENGES

Hardware	14 Hardware challenges
Crypto	25 Crypto challenges
Pwn	27 Pwn challenges
Web	22 Web challenges
Mobile	5 Mobile challenges
Reversing	13 Reversing challenges
Stego	8 Stego challenges
Misc	13 Misc challenges
OSINT	8 OSINT challenges
Forensics	10 Forensics challenges

Figure #2: *The challenges page layout*

When going through the ***challenges***, I see that most of them are asking us to download a file to start the challenges while other offer the method of doing it instantly. Since it was my first time using the website, I have to do offsite search to see what should I do to tackle on the problem of using the website. As I look up on how to start the challenge, I need to do first is to download the file and then put it in a coding language software, run it, and identify where the problem is and try to fix it if I can with my effort. From there, I chose the section of "***Pwn***" to start off my first challenges with this topic in mind. In the "***Pwn***" section, there is a layout of all the challenges that offer under this section and below would be an image of what it looks like. They are range from very easy to hard depending on the challenge. I chose the challenge "***racecar***" as it was on a very easy difficulty and it would be a good starting place to try out a challenge.

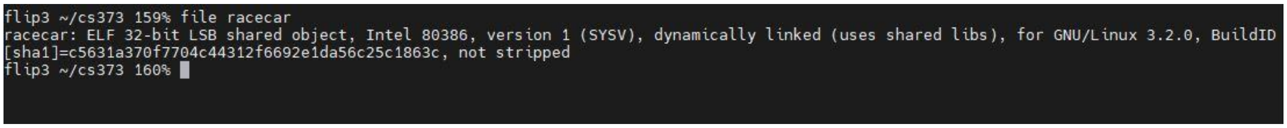


Pwn | 27 Pwn challenges

CHALLENGE	USER RATING	POINTS	USER SOLVES	RATING	
racecar VERY EASY		20pts	1253	160	27
You know 0xDiablos EASY		20pts	5820	1256	36
Space EASY		30pts	1863	501	14
Format EASY		30pts	691	228	6
Hunting EASY		30pts	751	204	6
Restaurant EASY		20pts	699	209	5
Bad grades EASY		30pts	451	134	4
Little Tommy MEDIUM		40pts	5331	1550	32
What does the f say? MEDIUM		30pts	548	202	4
Sick ROP MEDIUM		50pts	829	245	2
Antidote MEDIUM		30pts	291	94	1

Figure #3: *The challenges offer under the Pwn section*

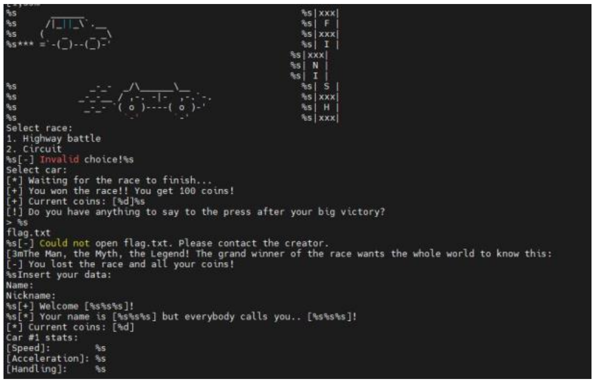
To start, I click on the "***racecar***" and it prompt me a screen asking me to download the file to get started, which I did, and from there, I extracted the file to be use later for decoding or hacking the file. To help me investigate this file, I will be using ***MobaXterm*** application to help me look, decode, and debug the program to see what's happening with the program. When I first drop the file into ***MobaXterm***, I check what type of file it is by doing the command "***file racecar"*** to see the file in detail. Below is an image of what information was printed out on the command line.



```
flip3 ~/cs373 159% file racecar
racecar: ELF 32-bit LSB shared object, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 3.2.0, BuildID
[sha1]=c5631a370f7704c44312f6692e1da56c25c1863c, not stripped
flip3 ~/cs373 160% ▋
```

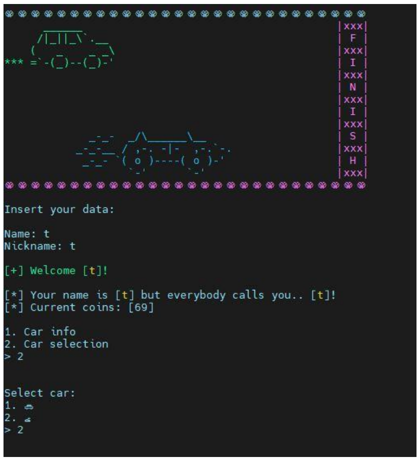
Figure #4: *Using the command "file racecar" to see what type of file racecar is*

From there, I decided to run the program and see what I can get as a result of the file. Since I didn't know how to start the file and run it yet, I decided to use the command "strings racecar" to see the content of the binary file that was given and below will display the figure of what the content have in there. Then, I was able to figure out how to run the program as the program was given me a permission denied and I unlock it by doing "***chmod +x filename***" to give myself the executable performance. From there, I run the program and the program run normally and I follow the program step by step and every time I play it, I always loses. Below is an image of what the program looks like once it is run.



```
%s      _______                                          %s|xxx|
%s     /|_||_\`.__                                       %s| F |
%s    (   _    _ _\                                      %s|xxx|
%s*** =`-(_)--(_)-'                                      %s| I |
                                                       %s|xxx|
                                                       %s| N |
                                                       %s| I |
%s            _-_-  _/\______\__                         %s| S |
%s          _-_-__ / ,-. -|-  ,-.`-.                     %s|xxx|
%s           _-_- `( o )----( o )-'                      %s| H |
%s                  `-'      `-'                         %s|xxx|
Select race:
1. Highway battle
2. Circuit
%s[-] Invalid choice!%s
Select car:
[*] Waiting for the race to finish...
[+] You won the race!! You get 100 coins!
[+] Current coins: [%d]%s
[!] Do you have anything to say to the press after your big victory?
> %s
flag.txt
%s[-] Could not open flag.txt. Please contact the creator.
[3mThe Man, the Myth, the Legend! The grand winner of the race wants the whole world to know this:
[-] You lost the race and all your coins!
%sInsert your data:
Name:
Nickname:
%s[+] Welcome [%s%s%s]!
%s[*] Your name is [%s%s%s] but everybody calls you.. [%s%s%s]!
[*] Current coins: [%d]
Car #1 stats:
[Speed]:        %s
[Acceleration]: %s
[Handling]:     %s
```

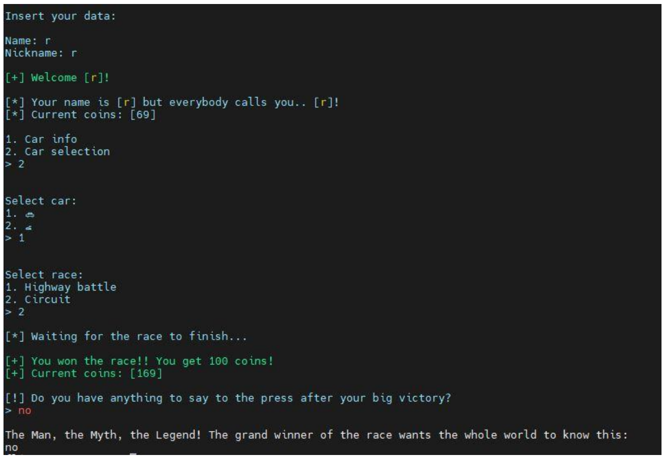
Figure #5: *The racecar file format using the string command*



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

Insert your data:

Name: t
Nickname: t

[+] Welcome [t]!

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

1. Car info
2. Car selection
> 2


Select car:
1. 🚘
2. 🏎
> 2
```

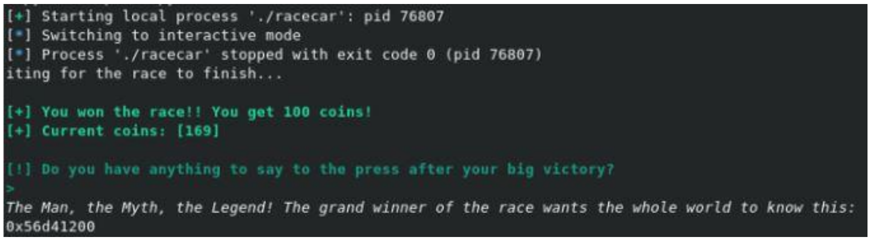
Figure #6: The program running under the racecar file

Next, I then try a different method from playing the racecar file and see if I could win. From the image below, it does show that you can win from just playing with the option below. From there, I went onto debugging the file to see how each information is access at which address. When I went to debugging, there was nothing wrong with it, so I decided to change it up. I created a python file to find out the leaked pointer address. From the file, I find that the address is located at ***0x56d41200***. From there, I then created a txt file to help me located where the pointer is sitting in the code.



```
Insert your data:

Name: r
Nickname: r

[+] Welcome [r]!

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

1. Car info
2. Car selection
> 2


Select car:
1. 🚗
2. 🏎
> 1


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

[*] 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?
> no

The Man, the Myth, the Legend! The grand winner of the race wants the whole world to know this:
no
```

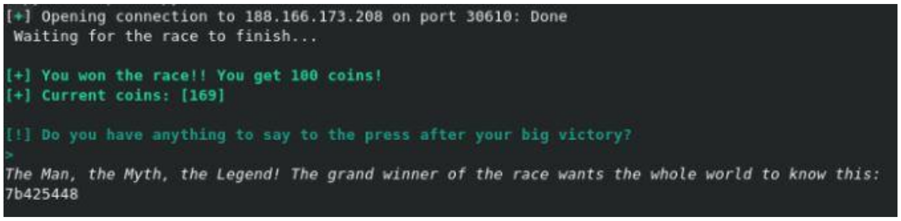
<u>Figure #7</u>: *Option from the racecar file that you could win*



```
[+] Starting local process './racecar': pid 76807
[*] Switching to interactive mode
[*] Process './racecar' stopped with exit code 0 (pid 76807)
iting 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?
>
The Man, the Myth, the Legend! The grand winner of the race wants the whole world to know this:
0x56d41200
```

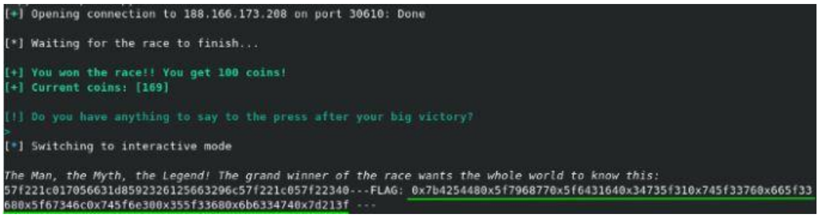
<u>Figure #8</u>: *Writing a python file to figure out the leaked pointer*

Then, using that txt file, I determine the leaked code sit in the 12th position in the code and from there, I modify the python code a bit to move it to the 12th place and printed out what content is in that position. Below is an image of what content being in that position and the hex number was ***7b425448***. From here, we could translate those hexes to each individual characters. So, we got "***{, B, T, H***" from that content address. Then, I reverse engineer the bytes that starting at 0x7b and keep going on from there and got this result in my flag. The flag content is "***0x7b4254480x5f7968770x5f6431640x34735f310x745f33760x665f33680x5f67346c0x745f6e30 0x355f33680x6b6334740x7d213f***". This could be seen below in the figure.



```
[+] Opening connection to 188.166.173.208 on port 30610: Done
 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?
>
The Man, the Myth, the Legend! The grand winner of the race wants the whole world to know this:
7b425448
```

<u>Figure #9</u>: *Modify the file to find the content of where the txt file content appears*



```
[+] Opening connection to 188.166.173.208 on port 30610: Done

[*] 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?
>
[*] Switching to interactive mode

The Man, the Myth, the Legend! The grand winner of the race wants the whole world to know this:
57f221c017056631d8592326125663296c57f221c057f22340---FLAG: 0x7b4254480x5f7968770x5f6431640x34735f310x745f33760x665f33
680x5f67346c0x745f6e300x355f33680x6b6334740x7d213f ---
```

Figure #10: *Reverse byte of the content at 12th position*

Finally, using the flag content, we have discovered the error in the code even though it is running fine at first. Overall, throughout the process of going through Hack the Box website, it was a good way to explore the hacking in cybersecurity throughout the knowledge that we learn from the beginning of the term. With this in mind, I believe this lab was a good taste of what cybersecurity offer as we are heading to the end of the term and be done with the class.