Lab #8 : Reverse Engineering Lab

CSE3801 : Introduction to Cyber Operations

Name: K. Kelly

Overview

This lab required me to figure out how I could reverse engineer four servers in order to get the flag for each one. I chose to use binary ninja to analyze each file in order to decide how I would get each flag.

The following section will detail how I found each flag for the reverse engineering challenges:

Methodology

RE-100:
The below image shows where the input is being authenticated in re-100.

```
int64_t authenticate()

printf("Authenticate >>> ")
void var_28
__isoc99_scanf(&data_2023, &var_28)
if (strcmp(&var_28, "AdAstraPerExplotium") != 0)
    fail()
    noreturn
return display_flag("flag.txt")
```

As it can be seen in the authentication process, var_28 needs to be “AdAstraPerExploitium” in order for the flag to be displayed. In order to do this I created a python script that receives until the authenticate line and then sends “AdAstraPerExploitium” so that the flag will print out. The python script and output is shown below:

```
from pwn import *

r = remote("cse3801-re-100.chals.io", 443, ssl=True, sni="cse3801-re-100.chals.io")

r.recvuntil("Authenticate >>>")

r.sendline("AdAstraPerExplotium")

response = r.recvall().decode()

print(response)
```

```
{21:40}~/workspace:main × ➭ python3 re100.py
[+] Opening connection to cse3801-re-100.chals.io on port 443:
/root/workspace/re100.py:5: BytesWarning: Text is not bytes; a
#bytes
  r.recvuntil("Authenticate >>>")
/root/workspace/re100.py:7: BytesWarning: Text is not bytes; a
#bytes
  r.sendline("AdAstraPerExplotium")
[+] Receiving all data: Done (51B)
[*] Closed connection to cse3801-re-100.chals.io port 443
 <<< Congratulations: flag{ch01c3_1s_an_1llus10n}\n
```

RE-200:
The below image shows where the input is being authenticated in re-200.

```
int64_t authenticate()

printf("Authenticate >>> ")
int32_t var_10
int32_t var_c
__isoc99_scanf("%i %i", &var_c, &var_10)
if (var_10 + var_c != 0x7a69)
    fail()
    noreturn
return display_flag("flag.txt")
```

The above code shows that the user needs to input two variables that when added together equal 0x7a69, which is 31337 in decimal form. Therefore, the python script should wait until the authentication line is received. Then the numbers 31337 and 0 should be sent for var_10 and var_c. When these numbers are added they will equal 0x7a69 and the flag will be displayed. The python script used and the output are shown in the image below:

```
from pwn import *

r = remote("cse3801-re-200.chals.io", 443, ssl=True, sni="cse3801-re-200.chals.io")

r.recvuntil("Authenticate >>>")

r.sendline("31337 0")

response = r.recvall().decode()

print(response)
```

```
{21:45}~/workspace:main ✗ ➲ python3 re200.py
[+] Opening connection to cse3801-re-200.chals.io on port 443: Done
/root/workspace/re200.py:5: BytesWarning: Text is not bytes; assuming
#bytes
  r.recvuntil("Authenticate >>>")
/root/workspace/re200.py:7: BytesWarning: Text is not bytes; assuming
#bytes
  r.sendline("31337 0")
[+] Receiving all data: Done (67B)
[*] Closed connection to cse3801-re-200.chals.io port 443
 <<< Congratulations: flag{th3_b0dy_cann0t_l1v3_w1th0ut_th3_m1nd}\n
```

RE-300:
The below image shows where the input is being authenticated in re-300.

```
int64_t authenticate()

srand(time(nullptr))
int32_t rax_1 = rand()
printf("<<< Nonce %i\n", rax_1)
printf("Authenticate >>> ")
int32_t var_10
__isoc99_scanf(&data_207a, &var_10)
if (rax_1 + var_10 != 0x7a69)
    fail()
    noreturn
return display_flag("flag.txt")
```

The code above shows that in order to display the flag the sum of rax_1 and var_10 must equal 0x7a69 or 31337. We can see that rax_1 is a random number and we must send var_10. In the

python script it will wait until “<<< Nonce” and then it will strip rax_1 and convert it to an int. Then it will wait until authenticate and val_10 should equal 31337 - rax_1 and send val_10 as a string. Now rax_1 + var_10 will equal 0x7a69 or 31337. The python script and output are shown in the images below:

```python
from pwn import *

r = remote("cse3801-re-300.chals.io", 443, ssl=True, sni="cse3801-re-300.chals.io")

r.recvuntil("<<< Nonce")

rax_1 = int(r.recvline().decode().strip())

r.recvuntil("Authenticate >>>")

val_10 = 31337 - rax_1

r.sendline(str(val_10))

response = r.recvall().decode()

print(response)
```

```
{21:45}~/workspace:main × ➭ python3 re200.py
[+] Opening connection to cse3801-re-200.chals.io on port 443: Done
/root/workspace/re200.py:5: BytesWarning: Text is not bytes; assuming
#bytes
  r.recvuntil("Authenticate >>>")
/root/workspace/re200.py:7: BytesWarning: Text is not bytes; assuming
#bytes
  r.sendline("31337 0")
[+] Receiving all data: Done (67B)
[*] Closed connection to cse3801-re-200.chals.io port 443
 <<< Congratulations: flag{th3_b0dy_cann0t_l1v3_w1th0ut_th3_m1nd}\n
```

RE-400:

The below image shows where the input is being authenticated in re-400.

```
int64_t authenticate()

int32_t var_c = 0x539
int32_t var_10 = 0x7a69
printf("Authenticate >>> ")
int32_t var_14
__isoc99_scanf(&data_206c, &var_14)
int32_t temp2
int32_t temp3
temp2:temp3 = var_10
if (mods.dp.d(temp2:temp3, var_c) != var_14)
    fail()
    noreturn
return display_flag("flag.txt")
```

The code above shows that temp2:temp3, which is equal to 0x7a69 or 31337, modulo var_c needs to equal var_14 in order for the flag to be displayed. In the python script it will wait until authenticate, then val_14 is initialized to 586 which is 0x7a69 mod 0x539, or 31337 mod 1337, and send the value as a string. Now the if statement is satisfied and the flag will be displayed. The python script and output are shown in the images below.

```python
from pwn import *

r = remote("cse3801-re-400.chals.io", 443, ssl=True, sni="cse3801-re-400.chals.io")

r.recvuntil("Authenticate >>>")

val_14 = 586

r.sendline(str(val_14))

response = r.recvall().decode()

print(response)
```

```
{22:10}~/workspace:main × ➭ python3 re400.py
[+] Opening connection to cse3801-re-400.chals.io on port
/root/workspace/re400.py:5: BytesWarning: Text is not byte
#bytes
  r.recvuntil("Authenticate >>>")
/root/workspace/re400.py:9: BytesWarning: Text is not byte
#bytes
  r.sendline(str(val_14))
[+] Receiving all data: Done (54B)
[*] Closed connection to cse3801-re-400.chals.io port 443
 <<< Congratulations: flag{th3_Matr1x_1s_3v3rywh3r3}\n
```

Results

I was able to find all of the flags and complete all of the challenges. I could always improve how quickly I completed the challenges by being more prepared and researching more before starting the challenges. Overall, although I could have prepared more I was still able to complete all of the challenges.

References

[1] Reverse Engineering Slides
[2] I got the general idea of how my scripts may be set up from a script I created for web fuzzing.