Lab #8: Reverse Engineering Lab

CSE 3801 : Introduction to Cyber Operations

Team: Tyler Dionne

Overview

The goal of this lab was to complete the reverse engineering challenges posted on cse3801.ctfd.io. For each challenge a unique binary file was provided and the objective was to reverse engineer the binary file to find the input that would result in displaying the flag.

Methodology

The methodology used in the lab was the following. For each challenge I loaded the binary file into a new session on binary ninja cloud. Once loaded I navigated to the linear disassembly tab. Once here I scrolled to the relevant methods shown on the left hand side. In this case each file had an int64_t authenticate() method which after analyzing called a display_flag method if a certain condition was met. The relevant piece of the disassembly along with the solution and explanation for each challenge is shown below.

<u>RE-100</u>

```
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")
```

After analyzing the disassembly of the binary shown above we can see that the display_flag method is called if the variable (var_28) inputted by the user is equal to “AdAstraPerExplotium”.

The pwn script that returned the flag is shown below:

```
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)
```

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")
```

After analyzing the disassembly of the binary shown above we can see that the display_flag function is called if the two variables inputted by the user (var_10 and var_c) are equal to 0x7a69 (which is equal to 31337 in decimal).
The pwn script that returned the flag is shown below:

```
from pwn import *
r = remote("cse3801-re-200.chals.io", 443, ssl=True,
sni="cse3801-re-200.chals.io")
r.recvuntil("Authenticate >>> ")
# want sum of var_10 and var_c to be 0x7a69
r.sendline("31337 0")
response = r.recvall().decode()
print(response)
```

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")
```

After analyzing the disassembly of the binary shown above we can see that the dispaly_flag function is called if the value of rax_1 and the variable inputted by the user (var_10) is equal to 31337. We can also see that rax_1 is assigned a random variable each time so we must take in the value of rax_1 printed each time by the program after <<< Nonce. Once this value is found we can calculate the value we need to input by subtracting the value of rax from 31337.
The pwn script that returned the flag is shown below:

```
from pwn import *
r = remote("cse3801-re-300.chals.io", 443, ssl=True,
sni="cse3801-re-300.chals.io")
r.recvuntil("<<< Nonce ")
# prints random value for rax each time take it in
rax = int(r.recvline().decode().strip())
# want to send a value for var 10 so that (var_10 + rax = 0x7a69)
var_10 = 31337 - rax
r.recvuntil("Authenticate >>> ")
r.sendline(str(var_10))
response = r.recvall().decode()
print(response)
```

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")
```

After analyzing the disassembly of the binary shown above we can see that the display_flag function is called if the value of temp2:temp3 mod var_c is equal to the variable inputted by the

user (var_14). Therefore to find the value we need to input we can simply calculate 31337 mod 1337 which is equal to 586.
The pwn script that returned the flag is shown below:

```
from pwn import *
r = remote("cse3801-re-400.chals.io", 443, ssl=True,
sni="cse3801-re-400.chals.io")
# 586 = var_10 mod var_c
var_14 = 586
r.recvuntil("Authenticate >>> ")
r.sendline(str(var_14))
response = r.recvall().decode()
print(response)
```

Results

In the final result I was able to find the flags and complete all four of the reverse engineering challenges posted on cse3801.ctfd.io.

Student Efforts

1. 100% - Tyler Dionne

References

N/A