WhaleCTF_writeup

原创

钞sir 于 2019-04-05 22:22:54 发布 9899 收藏 1

分类专栏： CTF

钞sir

本文链接：https://blog.csdn.net/qq_40827990/article/details/89048473

Warmup

很简单，就是字符串的异或操作：

```
s = [0x4C, 0x44, 0x59, 0x56, 0x4C, 0x51, 0x4D, 0x5A, 0x48, 0x75,
  0x59, 0x3A, 0x7C, 0x63, 0x51, 0x5B, 0x5E, 0x51, 0x79, 0x6F,
  0x7C, 0x63, 0x51, 0x7B, 0x7E, 0x51, 0x59, 0x4F, 0x5C, 0x43,
  0x51, 0x5B, 0x5E, 0x2F, 0x73]
flag = ''
for i in range(0,len(s)):
    flag = flag + chr(s[i] ^ 0xE)

print("flag: " + flag)
```

app-release

字符串的异或操作：

```
key = 'PXEJPMQFTiS|v`"#vMDw`KMA3_b~w3o'
lenth = len(key)
print(lenth)
flag = ""
for i in range(0,lenth):
    flag = flag + chr(ord(key[i])^18)

print("flag: " + flag)
```

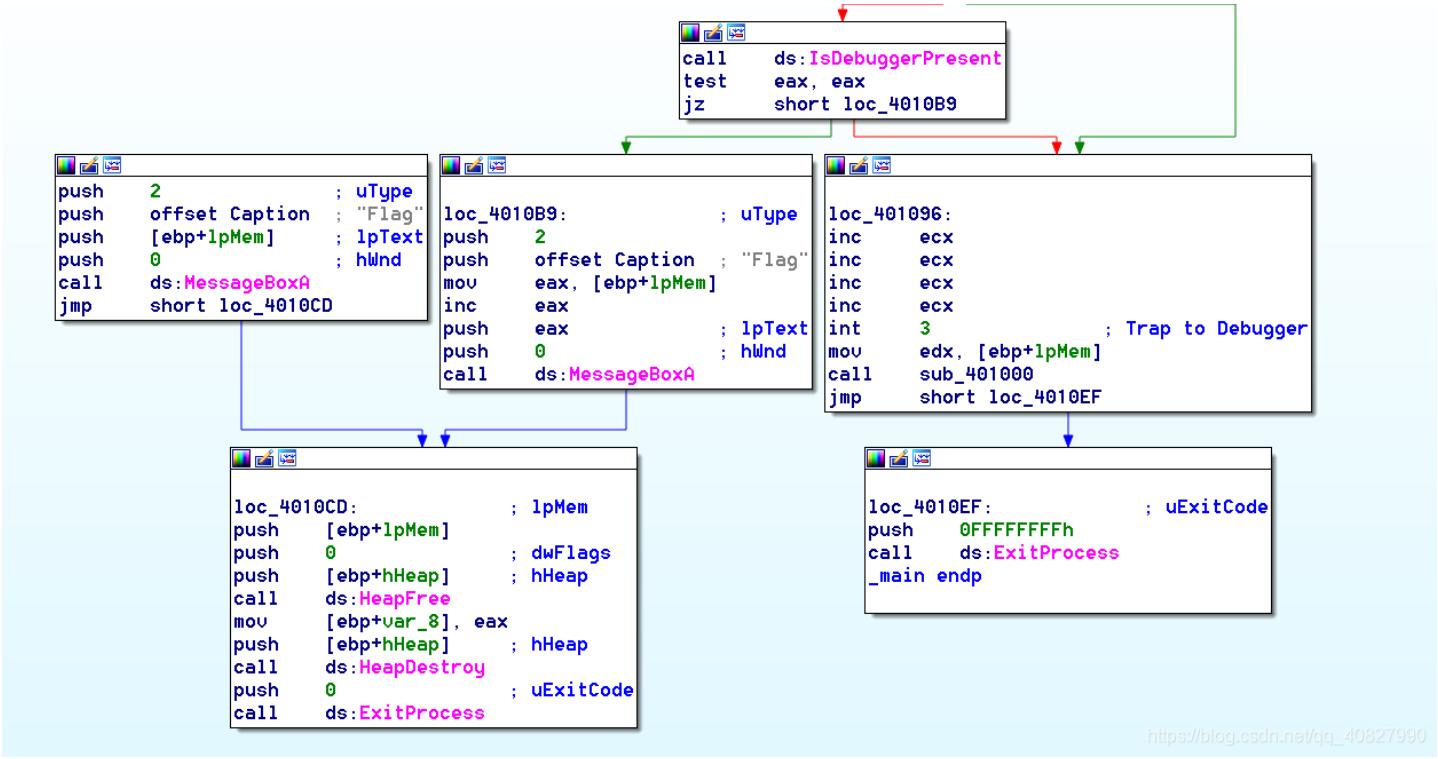
逆向练习

程序比较简单，取特定字符串中的不同位置的字符连接成flag,然后再末尾加上1024}:

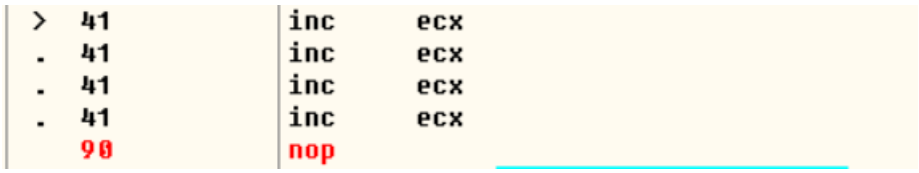
```c
#include<stdio.h>
#include<string.h>
int main()
{
 int esi,bl,ebx,i;
 char flag[0x12];
 char str[] = "sKfxEeft}f{gyrYgthtyhifsjei53UUrrr_t2cdsef66246087138\0087138";

 int num[] = {0x1,0x4,0xe,0xa,0x5,0x24,0x17,0x2a,0xd,0x13,0x1c,0xd,
     0x1b,0x27,0x30,0x29,0x2a};
 for(i=0;i<0x11;i++)
 {
  flag[i] = str[num[i]];
 }
 printf("flag: %s",flag);
// KEY{e2s6ry3r5s8f610241024}
 return 0;
}
```

rev2



主要函数在sub_401000处，这个函数在正常情况下是调用不到的，因为它上面有一个int 3断点，可以通过修改eip或将int 3指令nop掉：



然后就可以进入主要函数了，flag就会自己被异或出来：

crackme(ais3)

ida F5查看：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int result; // eax@2

  if ( argc == 2 )
  {
    if ( verify(argv[1]) )
      puts("Correct! that is the secret key!");
    else
      puts("I'm sorry, that's the wrong secret key!");
    result = 0;
  }
  else
  {
    puts("You need to enter the secret key!");
    result = -1;
  }
  return result;
}
```

关键函数verify():

```
__int64 __fastcall verify(__int64 key)
{
  int i; // [sp+14h] [bp-4h]@1

  for ( i = 0; *(i + key); ++i )
  {
    if ( encrypted[i] != (((*(i + key) ^ i) << ((i ^ 9) & 3)) | ((*(i + key) ^ i) >> (8 - ((i ^ 9) & 3)))) + 8 )
      return 0LL;
  }
  return i == 23;
}
```

就是将我们输入的字符串进行一些简单的计算，然后与encrypted数组比较；
注意最后比较的是al的值，所有要除以(16*16)取余：

```
movzx   eax, encrypted[rax]
cmp     al, [rbp+var_5]
jz      short loc_40059D
```

```c
#include<stdio.h>
#include<string.h>
int main()
{
 int esi,bl,ebx,i;
 int encrypted[24] = {0xca,0x70,0x93,0xc8,0x6,
 0x54,0xd2,0xd5,0xda,0x6a,0xd1,0x59,0xde,0x45,
 0xf9,0xb5,0xa6,0x87,0x19,0xa5,0x56,0x6e,0x63};
 char flag[24];
 for(i=0;i<23;i++)
 {
  ebx = 0;
  while(1)
  {
   bl = ((((ebx ^ i) << ((i ^ 9) & 3)) | ((ebx ^ i) >> (8 - ((i ^ 9) & 3)))) + 8)%(16*16);
   if(encrypted[i]==bl)
   {
    flag[i] = ebx;
    printf("ebx: 0x%x\n",ebx);
    break;
   }
   ebx ++;
  }
 }
 printf("flag: %s",flag);
 return 0;
}
```

这道题还一个用angr工具来做，参考：
https://blog.csdn.net/doudoudouzoule/article/details/79871979
https://www.freebuf.com/sectool/143056.html

trace(0CTF)

angr的提升用法,网上找了一个脚本：

```python
from __future__ import print_function
import struct
import angr

MAIN_START = 0x4009d4
MAIN_END = 0x00400c18

FLAG_LOCATION = 0x400D80
FLAG_PTR_LOCATION = 0x410EA0

def load_trace():
    res = []
    delay_slots = set()
    with open("./trace_8339a701aae26588966ad9efa0815a0a.log") as f:
        for line in f:
            if line.startswith('[INFO]'):
                addr = int(line[6:6+8], 16)
                res.append(addr)
                # every command like this is in delay slot
                # (in this particular binary)
                if ("move r1, r1" in line):
                    delay_slots.add(addr)
    return res, delay_slots
```

```
def main():
    trace_log, delay_slots = load_trace()
    # data.bin is simply the binary assembled from trace,
    # starting on 0x400770
    project = angr.Project("./data.bin", load_options={
        'main_opts': {
            'backend': 'blob',
            'custom_base_addr': 0x400770,
            'custom_arch': 'mipsel',
        },
    })

    state = project.factory.blank_state(addr=MAIN_START)
    state.memory.store(FLAG_LOCATION, state.solver.BVS("flag", 8*32))
    state.memory.store(FLAG_PTR_LOCATION, struct.pack("<I", FLAG_LOCATION))

    #sm = project.factory.simulation_manager(state) #why? not use it
    choices = [state]

    print("Tracing...")
    for i, addr in enumerate(trace_log):
        if addr in delay_slots:
            continue
        for s in choices:       #find the state of this address in the choices
            if s.addr == addr:
                break
        else:
            raise ValueError("couldn't advance to %08x, line %d" % (addr, i+1))

        if s.addr == MAIN_END:
            break

        # if command is a jump, it's followed by a delay slot
        # we need to advance by two instructions
        # https://github.com/angr/angr/issues/71
        if s.addr + 4 in delay_slots:
            choices = project.factory.successors(s, num_inst=2).successors
        else:
            choices = project.factory.successors(s, num_inst=1).successors

    state = s

    print("Running solver...")

    solution = state.solver.eval(state.memory.load(FLAG_LOCATION, 32), cast_to=str).rstrip(b'\0').decode('ascii'
)
    print("The flag is", solution)

    return solution

def test():
    assert main() == "0ctf{tr135m1k5l96551s9l5r}"

if __name__ == "__main__":
    main()
```

momo(0CTF)

还是angr:

```
import sys
import string
import angr
from angr.block import CapstoneInsn, CapstoneBlock

ins_char = 0x81fe6e0
flag_char = 0x81fe6e4
after_fgets = 0x08049653
mov_congrats = 0x0805356E

def main():
    p = angr.Project('./momo', load_options={'auto_load_libs': False}) #load the binary files
    addr = after_fgets
    size = mov_congrats - after_fgets
    # let's disasm with capstone to search targets
    insn_bytes = ''.join(
        p.loader.memory.read_bytes(addr, size))
    insns = []
    for cs_insn in p.arch.capstone.disasm(insn_bytes, addr):
        insns.append(CapstoneInsn(cs_insn))
    block = CapstoneBlock(addr, insns, 0, p.arch)
    targets = []

    # let's keep track of the state
    state = 0
    for ins in block.insns:
        if state == 0:
            if ins.op_str == 'edx, dword ptr [edx*4 + 0x81fe260]':
                state += 1
                continue
        if state == 1:
            if ins.op_str == 'al, byte ptr [0x81fe6e0]':
                state += 1
                continue
        if state == 2:
            if ins.op_str == 'dl, byte ptr [0x81fe6e4]':
                targets.append(ins.address + ins.size) #targets are addrs
                state = 0

    print "found {:d} targets".format(len(targets))
    assert len(targets) == 28

    flag_arr = ['0', 'c', 't', 'f', '{']

    for target in targets[5:]:
        print "\nexamining target {:#x}:".format(target)
        for trychar in string.printable:
            print trychar,
            sys.stdout.flush()
            flag = ''.join(flag_arr)+trychar
            state = p.factory.entry_state()
            state.posix.files[0].content.store(0, flag + "\n") #have no idea about posix.files

            e = p.surveyors.Explorer(start=state, find=(target,))#find target states from entry state
            e.run()

            assert len(e.found) == 1
            pp = e.found[0] #find the first state with target
```

```
            np = e.found[0] #find the first state with target

            while(True):
                nb_size = target - np.addr
                if nb_size <= 0:
                    break
                np = p.factory.successors(np, size=nb_size).flat_successors[0]
            assert nb_size == 0

            al = np.regs.eax[7:0]
            dl = np.regs.edx[7:0]
            al_val = al._model_concrete.value
            dl_val = dl._model_concrete.value

            if al_val == dl_val:
                flag_arr.append(trychar)
                break

    return ''.join(flag_arr)


def test():
    assert main() == '0ctf{m0V_I5_tUr1N9_c0P1Et3!}'

if __name__ == '__main__':
    print main()
```