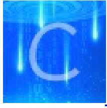
HITB_Binary_100_writeup

原创

Mi1k7ea 于 2018-08-26 15:10:26 发布 544 收藏
分类专栏： 逆向 CTF
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/SKI_12/article/details/82080147
版权



逆向 同时被 2 个专栏收录
20 篇文章 1 订阅
订阅专栏

CTF
1 篇文章 0 订阅
订阅专栏

HITB Binary 100是之前的HITB CTF的一道简单的逆向题，这里简单做一遍。

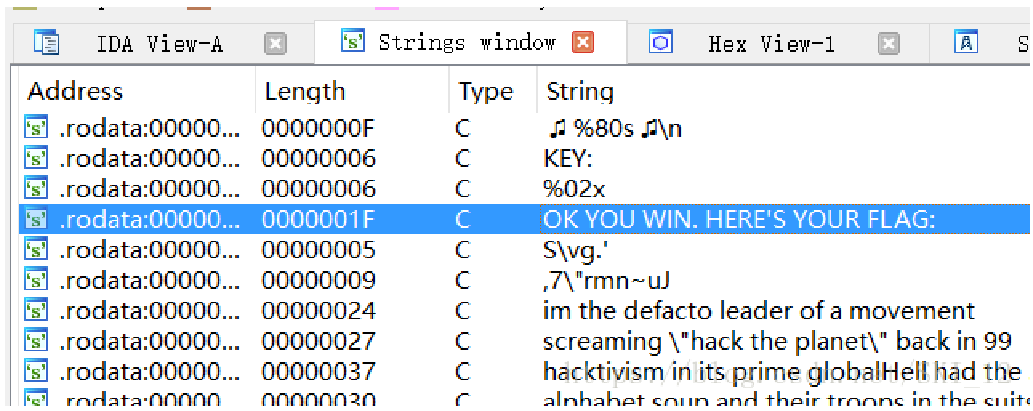
下载hitb-bin100.elf

将elf文件先跑一下，发现像歌词播放一样、每个一秒钟左右输出一句话，歌词好像没啥用，而且运行很久也还没停下来：

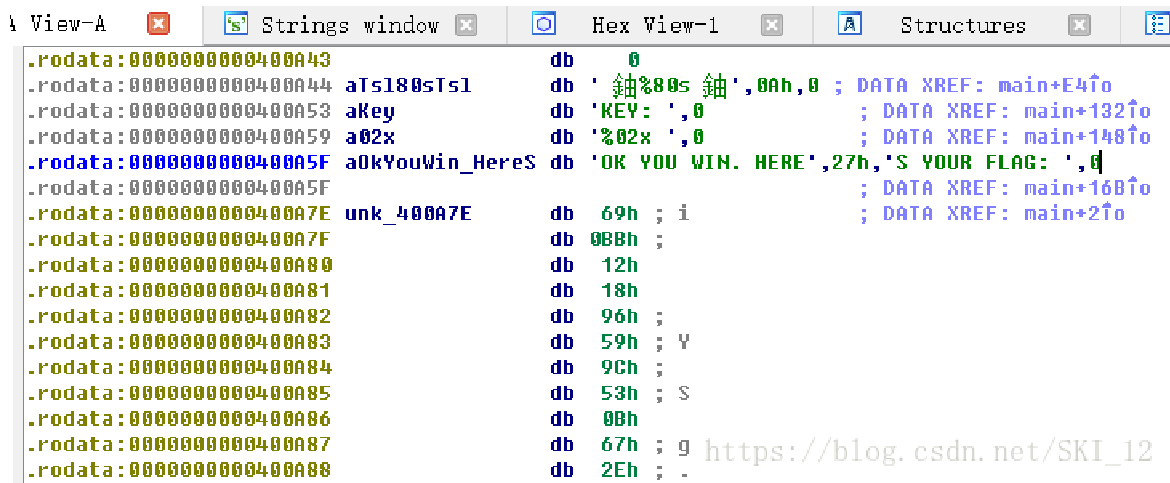
```
root@kali:~/Documents/CTF# chmod u+x hitb-bin100.elf
root@kali:~/Documents/CTF# ./hitb-bin100.elf
 ♫                                     iM tHE dEFACTO lEADER oF a mOVEMENT ♫
 ♫                                  sCREAMiNG "HACK tHE pLANET" bACK iN 99 ♫
 ♫                  hACKTiViSM iN iTS pRiME gLOBALHELL hAD tHE .MiL rOOTED ♫
 ♫                         aLPHABET sOUP aND tHEiR tROOPS iN tHE sUiTS kiD ♫
 ♫                             kiCKiNG dOWN dOORS aND sEiZiNG mY eQUiPMENT ♫
 ♫                     bLOCKiNG aLL mY sHiPMENTS siTTiNG oN tHEiR hiTLiST ♫
 ♫                                       0DAY rADiCAL eMPHATiC bEAT aDDiCT ♫
 ♫                   aND tHAT sTAB hiT i eNVELOPE tHE gAME cALL mE rABBiT ♫
 ♫                               hOP tO hOP i rUN tHE iNTERNET eQUiVOCALLY ♫
 ♫                           biTCH i bE hiT eM wiTH tHE bYTESTYLE sYMMETRY ♫
 ♫                            diGi g diGiTAL gANGSTER rEPPiNG tiLL iM dEAD ♫
 ♫                    sTEADY gREP aPACHE lOGS wHEN iM lOOKiNG fOR tHE fEDS ♫
 ♫                                 fAST fORWARD nOW tHE iNTERNET aNONYMOUS ♫
 ♫                  aND cAPTAiNS oF tHE lULZBOAT rAiSE tHE mAST pROMiNENT ♫
 ♫                                  dOMiNANT hACKS - aNTiSEC oN tHAT nEW nEW ♫
 ♫                       dROPPiNG tABLES iN mYSQL liKE iT wAS sOME pOO pOO ♫
 ♫                          pOUND aNTiSEC - pOUNDiNG tHROUGH yOUR sPEAKERS ♫
 ♫                               pOUND aNTiSEC - pOUND iT tO tHE bLEACHERS ♫
 ♫                             pOUND aNTiSEC - iF yOURE siTTiNG bELOW dECK ♫
 ♫                    iN tHE lULZBOAT sALUTE biTCH aND sHOW sOME rESPECT ♫
 ♫                                 lULZSEC biTCH tHEY fOLD yA hACKED sONY ♫
 ♫                                     gOT tHAT mD5 wE rOCKED yA mACARONi ♫
 ♫                                cOOK cOKE cRACK tHEN bOiL liKE a nOODLE ♫
 ♫                                wHiLE hBGARY sTAY tOAST liKE a sTREUDEL ♫
 ♫                             rOOTSHELL oN yA bOOTSTRAP - nOW wHOS tHAT? ♫
 ♫                                 bOTNET mJOiN aND dROP yOUR wHOLE c cLASS ♫
 ♫                                     sEE cLASS? iT iS eViDENT wE fLOSSiNG ♫
 ♫                                     iON cANNON iN tHE pROC liST dDOSSiNG ♫
 ♫                              lEAN bACK biTCH wE bE sENDiNG aN iNJECTiON ♫
 ♫                                mAGiC qUOTES oFF jOiN tABLE iNTERSECTiON ♫
 ♫                           pUT iT uP oN pASTEBiN iT wONT gET eRASED tHEN ♫
 ♫                          20 miLLiON hiTS tO yOUR dOME liKE sOME cAVEMEN ♫
 ♫                                      aSK cNN - yOU wANT a iNTERViEW? ♫
```

https://blog.csdn.net/SKI_12

根据字符串检索法，扔到IDA的String窗口查看是否存在关键字符串，可以看到有“FLAG”相关的字符串：



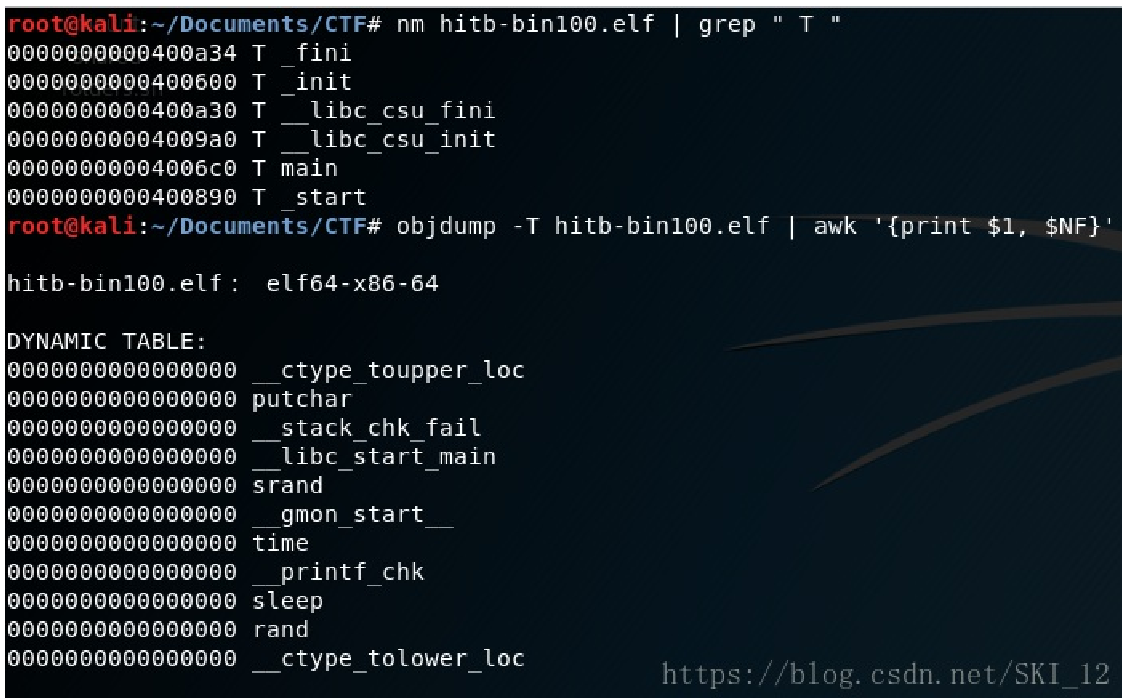
点击进去查看，发现这段代码是在main()函数中的：



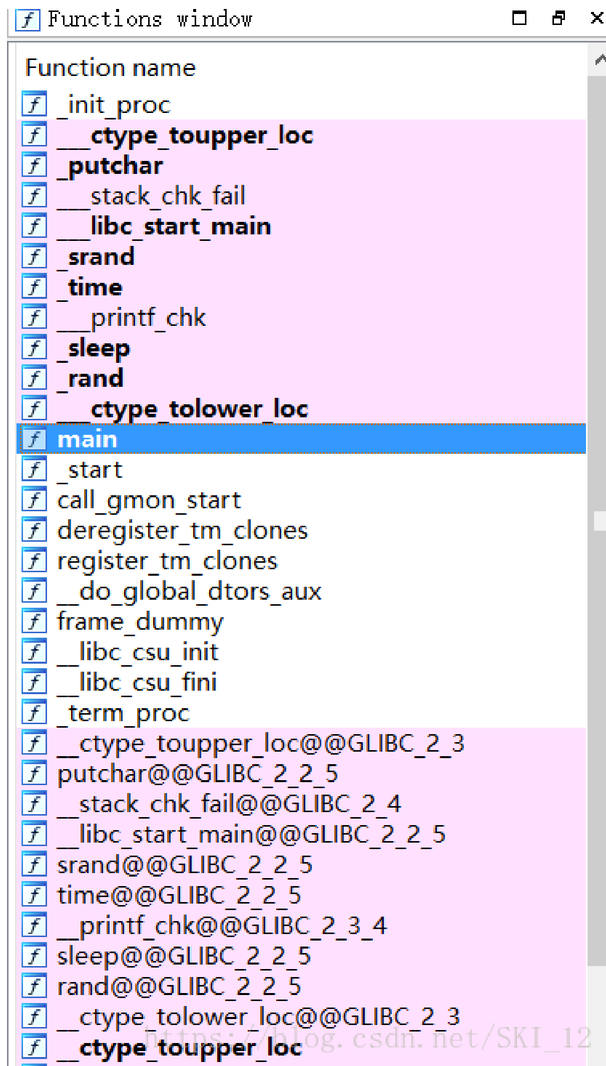
接着查看一下该二进制文件的函数等信息：

nm命令被用于显示二进制目标文件的符号表，如函数和全局变量等。

objdump查看目标文件或者可执行的目标文件的构成的gcc工具。-t，--syms，显示文件的符号表入口。类似于nm -s提供的信息。objdump -t obj：输出目标文件的符号表。



在Function Window也可以看到，除了调用一些标准库的函数，整个程序几乎都编译到了main函数中：



查看main函数的汇编代码，可以看到有两个地址值得研究，即4007e8和400803，因为其中都是会调用printf函数输出内容：

```
.text:00000000004007CC                 jnz     loc_400719
.text:00000000004007D2                 dec     [rsp+198h+var_190]
.text:00000000004007D6                 jz      short loc_4007E8
.text:00000000004007D8
.text:00000000004007D8 loc_4007D8:                             ; CODE XREF: main+54↑j
.text:00000000004007D8                 mov     ebp, 0DEFACEDh
.text:00000000004007DD                 xor     ebx, ebx
.text:00000000004007DF                 sub     ebp, [rsp+198h+var_18C]
.text:00000000004007E3                 jmp     loc_400719
.text:00000000004007E8 ; ---------------------------------------------------------------------------
.text:00000000004007E8
.text:00000000004007E8 loc_4007E8:                             ; CODE XREF: main+116↑j
.text:00000000004007E8                 lea     rbx, [rsp+198h+var_188]
.text:00000000004007ED                 lea     rbp, [rsp+198h+var_164]
.text:00000000004007F2                 mov     esi, offset aKey ; "KEY: "
.text:00000000004007F7                 mov     edi, 1
.text:00000000004007FC                 xor     eax, eax
.text:00000000004007FE                 call    ___printf_chk
.text:0000000000400803
.text:0000000000400803 loc_400803:                             ; CODE XREF: main+15D↓j
.text:0000000000400803                 movzx   edx, byte ptr [rbx]
.text:0000000000400806                 xor     eax, eax
.text:0000000000400808                 mov     esi, offset a02x ; "%02x "
.text:000000000040080D                 mov     edi, 1
.text:0000000000400812                 inc     rbx
.text:0000000000400815                 call    ___printf_chk
.text:000000000040081A                 cmp     rbx, rbp
.text:000000000040081D                 jnz     short loc_400803
.text:000000000040081F                 mov     edi, 0Ah        ; c
.text:0000000000400824                 xor     ebx, ebx
.text:0000000000400826                 call    _putchar
.text:000000000040082B                 mov     esi, offset aOkYouWin_HereS ; "OK YOU WIN. HERE'S YOUR FLAG: "
.text:0000000000400830                 mov     edi, 1
.text:0000000000400835                 xor     eax, eax
.text:0000000000400837                 call    ___printf_chk
.text:000000000040083C
```

https://blog.csdn.net/SKI_12

直接在GDB上运行并jump到相应的地址查看：

```
root@kali:~/Documents/CTF# gdb --quiet ./hitb-bin100.elf
Reading symbols from ./hitb-bin100.elf...(no debugging symbols found)...done.
(gdb) run
Starting program: /root/Documents/CTF/hitb-bin100.elf
 ♫                                                 iM tHE dEFACTO lEADER oF a mOVEMENT ♫
^C
Program received signal SIGINT, Interrupt.
0x00007ffff7af4260 in __nanosleep_nocancel () at ../sysdeps/unix/syscall-template.S:84
84      ../sysdeps/unix/syscall-template.S: 没有那个文件或目录.
(gdb) jump *0x4007e8
Continuing at 0x4007e8.
KEY: 2b e1 0f 24 00 00 00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 ed 6e 6d b2 00 00 00 00 c8 07 40 00
OK YOU WIN. HERE'S YOUR FLAG: +�$���[t�nm��@
*** stack smashing detected ***: /root/Documents/CTF/hitb-bin100.elf terminated

Program received signal SIGSEGV, Segmentation fault.
0x00007ffff7834874 in ?? () from /lib/x86_64-linux-gnu/libgcc_s.so.1
(gdb)
```

https://blog.csdn.net/SKI_12

可以看到，KEY是正常的输出出来了，但是FLAG输出的是一堆乱码，然后就出现“stack smashing detected”的错误而终止运行了。跳过去直接输出Flag是不可能的了。

F5看一下伪代码吧，看到在输出flag前一段主要的代码：

```
35  v20 = 201527;
36  v21 = time(0LL);
37  do
38  {
39    v11 = 0LL;
40    do
41    {
42      v5 = 0LL;
43      v6 = time(0LL);
44      srand(233811181 - v21 + v6);
45      v7 = v22[v11];
46      v22[v11] = rand() ^ v7;
47      v8 = (&funny)[8 * v11];
48      while ( v5 < strlen(v8) )
49      {
50        v9 = v8[v5];
51        if ( (_BYTE)v9 == 105 )
52        {
53          v24[(signed int)v5] = 105;
54        }
55        else
56        {
57          if ( (_DWORD)v5 && v8[v5 - 1] != 32 )
58            v10 = __ctype_toupper_loc();
59          else
60            v10 = __ctype_tolower_loc();
61          v24[(signed int)v5] = (*v10)[v9];
62        }
63        ++v5;
64      }
65      v24[(signed int)v5] = 0;
66      ++v11;
67      __printf_chk(1LL, " 鉮%80s 鉮玕n", v24);
68      sleep(1u);
69    }
70    while ( v11 != 36 );
71    --v20;
72  }
73  while ( v20 );
74  v13 = v22;
75  __printf_chk(1LL, "KEY: ", v12);
```

https://blog.csdn.net/SKI_12

分析可知，v20变量存放的是该循环执行的次数、无法改变；v21变量调用time函数生成一个时间戳，用于后续与v6变量的相减形成随机种子值；v11变量用于内循环执行的次数，从0-36；v6变量同样调用time函数生成一个时间戳，与v21变量不同在于每次内循环结束后重新生成时间戳时都会先sleep 1s后再生成；在内循环的最后是一个sleep()函数，调用来休眠1s；其余的代码都是用于生成输出的字符串。

这里如果想直接nop掉sleep()函数是行不通的，因为整段代码的执行依赖于该sleep()函数，v6与v21的差值在每次内循环结束的时候都会通过sleep()函数实现增一，如果不这样正常处理，将得不到正常的flag。

换个思路，这里关键的两个函数是sleep()和time()，我们可以重新定义该两个函数，通过LD_PRELOAD预先加载这两个函数而不是调用系统自己的从而实现欺骗程序该代码确实是睡眠了1s了，也就是说，自定义的sleep()函数直接对时间变量t自增1而不是真的等待了一秒才运行结束，从而可以将程序的休眠时间去掉：

```
root@kali:~/Documents/CTF# cat time.c
static int t = 0;

int sleep(int sec){
        t += sec;
}

int time(){
        return t;
}
root@kali:~/Documents/CTF#
```

https://blog.csdn.net/SKI_12

gcc编译为so文件（-fpic参数指定为位置无关、即使用相对地址，-shared参数指定为共享库），再通过LD_PRELOAD在该elf程序运行前优先加载生成的动态链接库time.so并管道输出到一个文件中查看，可以看到输出了flag：

```
root@kali:~/Documents/CTF# gcc -fpic -shared -o time.so time.c
root@kali:~/Documents/CTF# LD_PRELOAD=./time.so ./hitb-bin100.elf > result.txt
root@kali:~/Documents/CTF# tail result.txt
 ♫                                        lEAN bACK biTCH wE bE sENDiNG aN iNJECTiON ♫
 ♫                                          mAGiC qUOTES oFF jOIN tABLE iNTERSECTiON ♫
 ♫                                     pUT iT uP oN pASTEBiN iT wONT gET eRASED tHEN ♫
 ♫                                    20 miLLiON hiTS tO yOUR dOME liKE sOME cAVEMEN ♫
 ♫                                                     aSK cNN - yOU wANT a iNTERViEW? ♫
 ♫                                                 sEND a pRIVMSG tO tHE niCKNAME sABU ♫
 ♫                                                           oN iRC - mAN wE cONVENiNG ♫
 ♫                                     tHiS sOME 99 tHROWBACK sHiT tHAT iM sCREAMiNG ♫
KEY: 19 8f 67 74 c9 68 e6 0c 6f 54 1a 43 af 7b 5f b3 5c 01 98 58 68 56 1a 5e 31 0c 46 29 b8 a8 93 fc bf f9 70 5e
OK YOU WIN. HERE'S YOUR FLAG: p4ul_1z_d34d_1z_wh4t_th3_r3c0rd_s4ys
root@kali:~/Documents/CTF#
```

https://blog.csdn.net/SKI_12