2021工业互联网决赛tprinter题wp

原创

sln_1550 于 2021-11-03 10:56:45 发布 38 收藏

分类专栏： 逆向 文章标签： 安全

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/sln_1550/article/details/121113889

版权

逆向 专栏收录该内容

13 篇文章 0 订阅

订阅专栏

这题比赛的时候没有做出来，赛后花了2天才搞定，又学到新知识了：
IDA中看不出代码的逻辑，只有print_chk函数：

IDA View-A

```
1  __int64 __fastcall main(int a1, char **a2, char **a3, double a4)
2  {
3    struct tm *v4; // rax
4    time_t v6; // [rsp+0h] [rbp-18h] BYREF
5    unsigned __int64 v7; // [rsp+8h] [rbp-10h]
6
7    v7 = __readfsqword(0x28u);
8    puts("==== Time Printer ====");
9    __isoc99_scanf("%100s", &unk_559544070420);
10   v6 = time(0LL);
11   v4 = localtime(&v6);
12   __printf_chk(1LL, "Time: %T", v4, a4);
13   __printf_chk(1LL, "# %F\n", a4);
14   return 0LL;
15 }
```

CSDN @sln_1550

然后看字符串，发现一堆奇怪的格式化字符串，也不知道干啥的。

```
.rodata:000055...  00000017  C  ==== Time Printer ====
.rodata:000055...  00000006  C  %100s
.rodata:000055...  00000009  C  Time: %T
.rodata:000055...  00000006  C  # %F\n
.eh_frame:0000...  00000006  C  ;*3$\"
.data:00005595...  00000007  C  %51C%s
.data:00005595...  0000002C  C  %3.0hM%3.2lO%+0.3lM%0.4LA%3.1lM%3.0lS%+7.3C
.data:00005595...  00000078  C  %0.1024hhM%1.0lM%0.707406378LN%1.1431655765LN%0.1LR%1.1LL%0.1lO%2.0lM%0.171LM%1.555LM%7C%0.171hhM%0.825110565LS%0171.0C
.data:00005595...  00000009  C  \nRSM}y<+[
.data:00005595...  0000000B  C  Vtx:/Y\ag}8+
```

CSDN @sln_1550

然后开始调试，慢慢梳理程序的逻辑，搞清楚原来程序把一些格式化字母注册了一个函数

```
2  {
3    register_printf_function('T', func, arginfo);
4    register_printf_function('F', sub_559543E6F2C0, arginfo);
5    register_printf_function('C', (printf_function *)sub_559543E6F1F0, sub_559543E6F1D0);
6    register_printf_function('M', calc_let, sub_559543E6F1D0);
7    register_printf_function('A', calc_add, sub_559543E6F1D0);
8    register_printf_function('S', calc_sub, sub_559543E6F1D0);
9    register_printf_function('X', calc_multiply, sub_559543E6F1D0);
10   register_printf_function('V', calc_divide, sub_559543E6F1D0);
11   register_printf_function('P', calc_mod, sub_559543E6F1D0);
12   register_printf_function('L', calc_shl, sub_559543E6F1D0);
13   register_printf_function('R', calc_shr, sub_559543E6F1D0);
14   register_printf_function('O', calc_xor, sub_559543E6F1D0);
15   register_printf_function('N', calc_and, sub_559543E6F1D0);
16   return register_printf_function('U', (printf_function *)calc_or, sub_559543E6F1D0);
17 }
```

CSDN @sln_1550

当碰到printf函数中格式化字符串中有这些字母的时候，就执行对应的函数，这是一种新型的分支引擎，整个代码类似于虚拟机一样的机制来实现加密和比较。

比如C这个字母对应的是嵌套调用printf，也就是call的意思；M是赋值，A是加，S是减等等。
这样整个虚机的逻辑就渐渐清晰了

```
.data:0000000000202020 25 35 31 43 25 73 00        a51cS           db '%51C%s',0           ; DATA XREF: calc_let:loc_B00↑o
.data:0000000000202020                                             ; calc_let:loc_B3B↑o ...
.data:0000000000202027 25 33 2E 30 68 4D 25 33+    a30hm32lo03lm04 db '%3.0hM%3.21O%+0.31M%0.4LA%3.11M%3.01S%+7.3C',0
.data:0000000000202053 25 30 2E 31 30 32 34 68+    a01024hhm101m07 db '%0.1024hhM%1.01M%0.707406378LN%1.1431655765LN%0.1LR%1.1LL%0.11O%2'
.data:0000000000202053 68 4D 25 31 2E 30 6C 4D+                    db '.01M%0.171LM%1.555LM%7C%0.171hhM%0.825110565LS%0171.0C',0
.data:00000000002020CB BC                                          db 0BCh
.data:00000000002020CC AC                                          db 0ACh
.data:00000000002020CD BC                                          db 0BCh
.data:00000000002020CE AA                                          db 0AAh
.data:00000000002020CF A9                                          db 0A9h
.data:00000000002020D0 AE                                          db 0AEh
.data:00000000002020D1 A6                                          db 0A6h
.data:00000000002020D2 D7                                          db 0D7h
.data:00000000002020D3 D4                                          db 0D4h
.data:00000000002020D4 B9                                          db 0B9h
.data:00000000002020D5 A3                                          db 0A3h
.data:00000000002020D6 B5                                          db 0B5h
.data:00000000002020D7 A8                                          db 0A8h
.data:00000000002020D8 AB                                          db 0ABh
.data:00000000002020D9 A3                                          db 0A3h
.data:00000000002020DA D7                                          db 0D7h
.data:00000000002020DB D4                                          db 0D4h
.data:00000000002020DC B9                                          db 0B9h
.data:00000000002020DD A0                                          db 0A0h
.data:00000000002020DE B5                                          db 0B5h
.data:00000000002020DF A9                                          db 0A9h
```

CSDN @sln_1550

这里有两段VM代码，边调试边猜出来大概的逻辑就是把输入的前4个字节进行bit奇偶互换得到key，然后和0x2020cb的地方异或后判断是不是等于%25%30%2e%31（转成字符就是%0.1），然后会用key继续解密0x2020cb后面的部分，再调用printf执行。

```
.data:00005610BB45A020 25 35 31 43 25 73 00        a51cS           db '%51C%s',0           ; DATA XREF: calc_let:loc_5610BB258B00↑o
.data:00005610BB45A020                                             ; calc_let:loc_5610BB258B3B↑o ...
.data:00005610BB45A027 25 33 2E 30 68 4D 25 33+    a30hm32lo03lm04 db '%3.0hM%3.21O%+0.31M%0.4LA%3.11M%3.01S%+7.3C',0
.data:00005610BB45A053 25 30 2E 31 30 32 34 68+    a01024hhm101m07 db '%0.1024hhM%1.01M%0.707406378LN%1.1431655765LN%0.1LR%1.1LL%0.11O%2'
.data:00005610BB45A053 68 4D 25 31 2E 30 6C 4D+                    db '.01M%0.171LM%1.555LM%7C%0.171hhM%0.825110565LS%0171.0C',0
.data:00005610BB45A0CB 25 30 2E 31 30 32 34 4C+    a010241m1171lm2 db '%0.1024LM%1.171LM%2.0LM%275C%0.1024LM%1.01M%1.1LA%2.0LM%352C%0.10'
.data:00005610BB45A0CB 4D 25 31 2E 31 37 31 4C+                    db '24LM%1.1280LM%2.0LM%7.1LM%429C%+532.7C',0
.data:00005610BB45A133 25 34 2E 30 6C 4D 25 34+    a401m421a51lm52 db '%4.01M%4.21A%5.11M%5.21A%6.4hM%7.5hM%6.71O%+4.61M%2.4LA%4.21M%4.3'
.data:00005610BB45A133 2E 32 6C 41 25 35 2E 31+                    db '8LS%-275.4C',0
.data:00005610BB45A180 25 34 2E 30 68 4D 25 35+    a40hm51hm45lo04 db '%4.0hM%5.1hM%4.51O%+0.41M%0.1LA%1.1LA%2.1LA%3.21M%4.38LM%4.3LS%3.'
.data:00005610BB45A180 2E 31 68 4D 25 34 2E 35+                    db '41S%-352.3C',0
.data:00005610BB45A1CD 25 34 2E 30 68 4D 25 35+    a40hm51hm541s50 db '%4.0hM%5.1hM%5.41S%+503.5C%-503.5C%0.4LA%1.4LA%2.4LA%3.21M%3.38LS'
.data:00005610BB45A1CD 2E 31 68 4D 25 35 2E 34+                    db '%-429.3C',0
.data:00005610BB45A217 25 36 2E 31 30 32 34 4C+    a610241m6750038 db '%6.1024LM%+6.7500389LM%7.0LM',0
.data:00005610BB45A234 25 36 2E 31 30 32 34 4C+    a610241m6723595 db '%6.1024LM%+6.7235959LM',0
.data:00005610BB45A24B 00                                          db    0
.data:00005610BB45A24C 00                                          db    0
```

CSDN @sln_1550

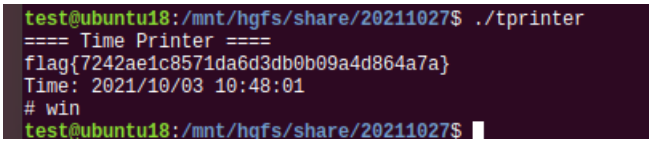
根据这个条件计算出输入的前4个字符正好是flag，解密以后的vm代码也很长，大致逻辑也就是把输入的flag内容和0x2020cb的字符串异或，然后每4个字节一组进行异或处理，加密以后再和下面这个固定的字符串比较，如果一致就输出win，否则输出err

```
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
1F 0A 52 53 4D 7D 79 3C  2B 5B 06 10 56 74 78 3A  ..RSM}y<+[..Vtx:
2F 59 07 67 7D 38 2B 12  07 2F 41 7E 0E 0D 4C 0F  /Y.g}8+../A~..L.
53 7D 79 0F 2C 69 31 2E  00 00 00 00 00 00 00 00  S}y.,i1.........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

最后逆向的脚本：

```
b=bytes.fromhex('25302E313032344C4D25312E3137314C4D25322E304C4D2532373543254325302E313032344C4D25312E306C4D25312E314C')
t=bytes.fromhex('1F0A52534D7D793C2B5B06105674783A2F5907677D382B12072F417E0E0D4C0F537D790F2C69312E')
t=list(t)
for i in range(34,-1,-1):
    x=t[i:i+5]
    t[i+3]^=x[4]
    t[i+2]^=t[i+3]
    t[i+1]^=t[i+2]
    t[i+0]^=t[i+1]
for i in range(len(t)):
    t[i]^=b[i]
print(''.join(chr(i) for i in t))
```

测试flag成功


test@ubuntu18:/mnt/hgfs/share/20211027$ ./tprinter
==== Time Printer ====
flag{7242ae1c8571da6d3db0b09a4d864a7a}
Time: 2021/10/03 10:48:01
win
test@ubuntu18:/mnt/hgfs/share/20211027$