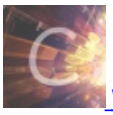
GKCTF X DASCTF应急挑战杯-Maple_root-Writeup

原创

「已注销」 于 2021-06-27 12:59:04 发布 283 收藏

分类专栏： WriteUp CTF DASCTF

本文链接：https://blog.csdn.net/OERR0R_/article/details/118272291


WriteUp 同时被 3 个专栏收录

3 篇文章 0 订阅

订阅专栏


CTF

18 篇文章 0 订阅

订阅专栏


DASCTF

1 篇文章 0 订阅

订阅专栏

部分题目WP包含图片，因为CSDN无法获取外部图片所以无法加入，如想观看完整版本欢迎点此访问

另：CSDN插入外链图片次次失败真的是太恶心人了，已经在考虑转站了

> 参赛队员：
> b4tteRy, x0r, f1oat
> 最终成绩：2285
> 最终排名：27

总结

经过最近几次类线下的演练，感觉慢慢对CTF有点上手了，这次终于不再爆0了，继续努力

MISC

签到

wireshark打开可知是shell流量，命令结果编码为hex+base64。观察前面几条`whoami/ls`等命令输出可知每行输出都是倒序输出。

故将`cat /f14g|base64`的结果每行倒序拼接解码后得到flag的编辑记录，去除双写得到flag。

你知道apng吗

将apng转换为gif后查看，发现部分关键帧内有二维码。

扫描结果顺序拼接得到flag。

银杏岛の奇妙冒险

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传

FireFox Forensics

下载下来解压以后有一个json文件和一个sql存储的db数据库文件，查看db数据库内容是加密的，又根据题目提示本题内容属于登录信息相关，应该是FireFox浏览器所存储的登录信息，在Github上已有相关的解密脚本，直接下载以后将题目给出的两个文件放入相同目录，使用`python3 firepwd.py`即可成功解密得到flag

excel骚操作

打开看到一行字，根据提示flag隐写在表格中。
依次查看单元格/解压查看数据可发现部分单元格值为1，查看单元格格式可发现格式为;;;，改变格式后正常显示出数字。
设置条件格式，使得所有值为1的单元格黑色填充，并拉伸所有数字范围内单元格至正方形，得到黑色单元格画成的汉信码。
使用中国编码app扫描得到flag。

Reverse

QQQQT

下载后使用PEID扫描是WIN32程序，拖入IDA32分析可看出程序显然是由Qt语言所编写，题目名称也印证了这一点，一开始在IDA的WinMain窗口中多次依照逻辑查找无果，之后又在OD中尝试进行动态调试，在分析出的字符串当中找到了flag相关的关键词已经其上下的文本地址
然后在IDA中根据以上地址找到这些字符串所在的位置，可以得到如下函数

```
int __thiscall sub_4012F0(_DWORD *this)
{
  int v1; // edi
  _BYTE *v2; // esi
  const char *v3; // edx
  _BYTE *v4; // esi
  int v5; // ecx
  int v6; // eax
  int v7; // ecx
  int v8; // edx
  int v9; // edi
  int v10; // esi
  _BYTE *v11; // ecx
  unsigned int v12; // ecx
  int v14; // [esp-8h] [ebp-A8h]
  char v16[4]; // [esp+10h] [ebp-90h] BYREF
  char v17[4]; // [esp+14h] [ebp-8Ch] BYREF
  _BYTE *v18; // [esp+18h] [ebp-88h]
  const char *v19; // [esp+1Ch] [ebp-84h]
  int v20; // [esp+20h] [ebp-80h]
  int v21; // [esp+24h] [ebp-7Ch] BYREF
  _BYTE *v22; // [esp+28h] [ebp-78h] BYREF
  char v23[60]; // [esp+2Ch] [ebp-74h] BYREF
  __int128 v24[2]; // [esp+68h] [ebp-38h] BYREF
  __int64 v25; // [esp+88h] [ebp-18h]
  int v26; // [esp+9Ch] [ebp-4h]

  MEMORY[0x5FF6](*(_DWORD *)(this[6] + 4), v16);
  v26 = 0;
  MEMORY[0x7C7C](v16, v17);
  LOBYTE(v26) = 1;
  v19 = (const char *)MEMORY[0x7C48](v17);
  v24[0] = 0i64;
  v24[1] = 0i64;
```

```
v25 = 0164;
strcpy(v23, "123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz");
v21 = 138 * strlen(v19) / 0x64;
v14 = v21 + 1;
v1 = 0;
v22 = (_BYTE *)MEMORY[0x8114](v21 + 1);
v2 = v22;
sub_402C08(v22, 0, v14);
v3 = v19;
v20 = (int)(v19 + 1);
if ( strlen(v19) )
{
  v4 = &v2[v21];
  v18 = v4;
  while ( 1 )
  {
    v20 = ((char)*v4 << 8) + v3[v1];
    v5 = v20 / 58;
    *v4 = v20 % 58;
    if ( v5 )
    {
      do
      {
        v6 = (char)*--v4;
        v7 = (v6 << 8) + v5;
        v20 = v7 / 58;
        *v4 = v7 % 58;
        v5 = v20;
      }
      while ( v20 );
      v4 = v18;
    }
    if ( ++v1 >= strlen(v19) )
      break;
    v3 = v19;
  }
  v2 = v22;
}
v8 = 0;
if ( !*v2 )
{
  do
    ++v8;
  while ( !v2[v8] );
}
v9 = v21;
if ( v8 <= v21 )
{
  v10 = v2 - (_BYTE *)v24;
  do
  {
    v11 = (char *)v24 + v8++;
    *v11 = v23[(char)v11[v10]];
  }
  while ( v8 <= v9 );
}
if ( !MEMORY[0x7C1A](v24, "56fkoP8KhwCf3v7CEz") )
{
  if ( v19 )
    v12 = strlen(v19);
```

```
    else
      v12 = -1;
    v22 = (_BYTE *)MEMORY[0x7CCC](v19, v12);
    LOBYTE(v26) = 2;
    v21 = MEMORY[0x7CCC]("flag", 4);
    LOBYTE(v26) = 3;
    MEMORY[0x6124](this, &v21, &v22, 1024, 0);
    MEMORY[0x7C66](&v21);
    MEMORY[0x7C66](&v22);
  }
  MEMORY[0x7C30](v17);
  return MEMORY[0x7C66]();
}
```

由上方字符串`123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz`推测为Base58编码，将字符串`56fkoP8KhwCf3v7CEz`以Base58解码后尝试提交即为flag

Crash

打开发现本程序是用Golang编写的，因此使用IDA7.6打开即可对应大部分的字符集，打开main_main看第18行为

```
if ( v1[1] == 43LL && *(_DWORD *)v0 == 1413696327 && *(_WORD *)(v0 + 4) == 31558 && *(_BYTE *)(v0 + 42) == 125
)
```

其中，各个等值式分别判定了`TCKG`(内存地址中倒序存储字符串),`{F`,`}`，因此推断这里应该就是在进行flag的判定，接着进入main_check函数

```
void __golang main_check(__int64 a1, unsigned __int64 a2)
{
  __int64 v2; // [rsp+10h] [rbp-68h]
  __int64 v3; // [rsp+10h] [rbp-68h]
  __int64 v4; // [rsp+10h] [rbp-68h]
  __int64 v5; // [rsp+18h] [rbp-60h]
  __int64 v6; // [rsp+18h] [rbp-60h]
  __int64 v7; // [rsp+18h] [rbp-60h]
  __int64 v8; // [rsp+18h] [rbp-60h]
  __int64 v9; // [rsp+18h] [rbp-60h]
  __int64 v10; // [rsp+18h] [rbp-60h]
  __int64 v11; // [rsp+20h] [rbp-58h]
  __int64 v12; // [rsp+20h] [rbp-58h]
  __int64 v13; // [rsp+20h] [rbp-58h]
  __int64 v14; // [rsp+20h] [rbp-58h]
  __int64 v15; // [rsp+20h] [rbp-58h]
  __int64 v16; // [rsp+20h] [rbp-58h]
  __int64 v17; // [rsp+28h] [rbp-50h]
  __int64 v18; // [rsp+28h] [rbp-50h]
  char v19[32]; // [rsp+30h] [rbp-48h] BYREF
  char v20[32]; // [rsp+50h] [rbp-28h] BYREF

  if ( a2 < 0x1E )
    runtime_panicSliceAlen();
  v2 = main_encrypto(a1 + 6, 24LL);
  if ( v5 == 44 )
  {
    v11 = runtime_memequal(v2, (__int64)"o/aWPjNNxMPZDnJlNp0zK5+NLPC4Tv6kqdJqjkL0XkA=", 44LL, 44);
    if ( (_BYTE)v5 )
    {
      if ( a2 < 0x22 )
        runtime_panicSliceAlen();
```

```
      v17 = runtime_stringtoslicebyte((__int64)v19, a1 + 30, 4LL, v5, v11);
      Encrypt_HashHex2(v6, v12, v17, v6, v12);
      if ( v13 == 64 )
      {
        v14 = runtime_memequal(
                v7,
                (__int64)"6e2b55c78937d63490b4b26ab3ac3cb54df4c5ca7d60012c13d2d1234a732b74",
                64LL,
                v7);
        if ( (_BYTE)v7 )
        {
          if ( a2 < 0x26 )
            runtime_panicSliceAlen();
          v18 = runtime_stringtoslicebyte((__int64)v20, a1 + 34, 4LL, v7, v14);
          Encrypt_HashHex5(v8, v15, v18, v8, v15);
          if ( v16 == 128 )
          {
            runtime_memequal(
              v9,
              (__int64)"6500fe72abcab63d87f213d2218b0ee086a1828188439ca485a1a40968fd272865d5ca4d5ef5a651270a52ff
952d955c9"
                        "b757caae1ecce804582ae78f87fa3c9",
              128LL,
              v9);
            if ( (_BYTE)v9 )
            {
              if ( a2 < 0x2A )
                runtime_panicSliceAlen();
              main_hash(a1 + 38, 4LL, v3, v9);
              if ( v10 == 32 )
                runtime_memequal(v4, (__int64)"ff6e2fd78aca4736037258f0ede4ecf0", 32LL, 32);
            }
          }
        }
      }
    }
  }
}
```

可以看出，总共分为了若干段，第一段有六个字符，且使用了DES加密（可直接爆破）第二段是使用了sha256加密，第三段是使用了sha512加密，最后一段则使用了md5加密，以上若干加密段皆可根据前面所截取提交内容内的长度推断该段加密前明文的长度，因此除第一个DES爆破解密已有相关网站外，其余皆写相同脚本爆破即可

第一段DES加密可以从`encoding_json_Unmarshal`内找到key和iv

```
    "key": "WelcomeToTheGKCTF2021XXX",
    "iv": "1Ssecret"
```

接着发现此处key共有24byte，因此推测为3des，因此进行相应的爆破即可得到第一段flag，后面几段的解题脚本如下：

sha256 -> 4位

```python
import hashlib
import string
import itertools

dateset = string.ascii_lowercase + string.digits

res = "6e2b55c78937d63490b4b26ab3ac3cb54df4c5ca7d60012c13d2d1234a732b74"

def generate_strings(length=4):
    chars = string.ascii_lowercase + string.digits
    for item in itertools.product(chars, repeat=length):
        tmp = "".join(item)
        aa = hashlib.sha256(tmp.encode('utf-8')).hexdigest()

        if aa.hexdigest() == res:
            print(tmp)
            exit(0)



generate_strings()
```

sha512 -> 4位

```python
import hashlib
import string
import itertools

dateset = string.ascii_lowercase + string.digits

res = "6500fe72abcab63d87f213d2218b0ee086a1828188439ca485a1a40968fd272865d5ca4d5ef5a651270a52ff952d955c9b757caae
1ecce804582ae78f87fa3c9"

def generate_strings(length=4):
    chars = string.ascii_lowercase + string.digits
    for item in itertools.product(chars, repeat=length):
        tmp = "".join(item)
        aa = hashlib.sha512(tmp.encode('utf-8')).hexdigest()

        if aa.hexdigest() == res:
            print(tmp)
            exit(0)



generate_strings()
```

md5 -> 4位

```python
import hashlib
import string
import itertools

dateset = string.ascii_lowercase + string.digits

flag = "ff6e2fd78aca4736037258f0ede4ecf0"

def generate_strings(length=4):
    chars = string.ascii_lowercase + string.digits
    for item in itertools.product(chars, repeat=length):
        md5 = hashlib.md5()

        tmp = "".join(item)
        md5.update(tmp.encode('utf-8'))

        if md5.hexdigest() == flag:
            print(tmp)
            exit(0)
```

将以上爆破出来的值按照顺序连在一起即为flag

Web

easycms

题目提示五位弱密码，尝试admin/12345后登陆成功。
尝试更改主题相关模板代码提示需要创建`/system/tmp/xxxx.txt`(貌似是动态的四个字母）。
在素材库上传文件可观察到路径是文件名直接拼接来的，故可利用目录穿越创建上述要求创建的文件。再向模板内插入shell后读flag即可。