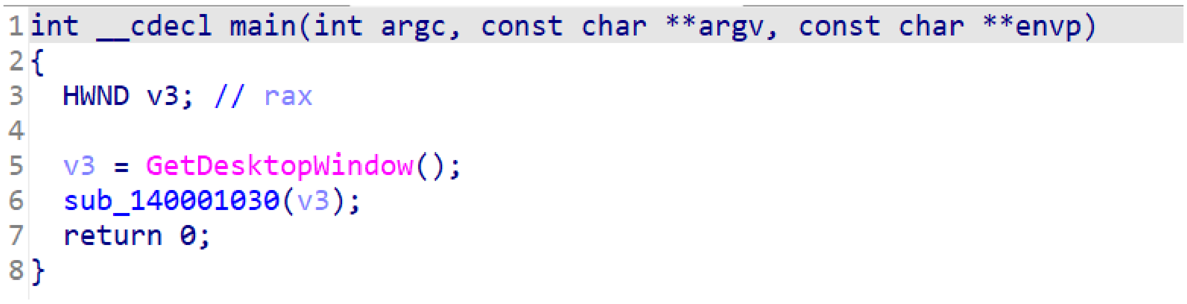
cap

我的评价是纯纯misc

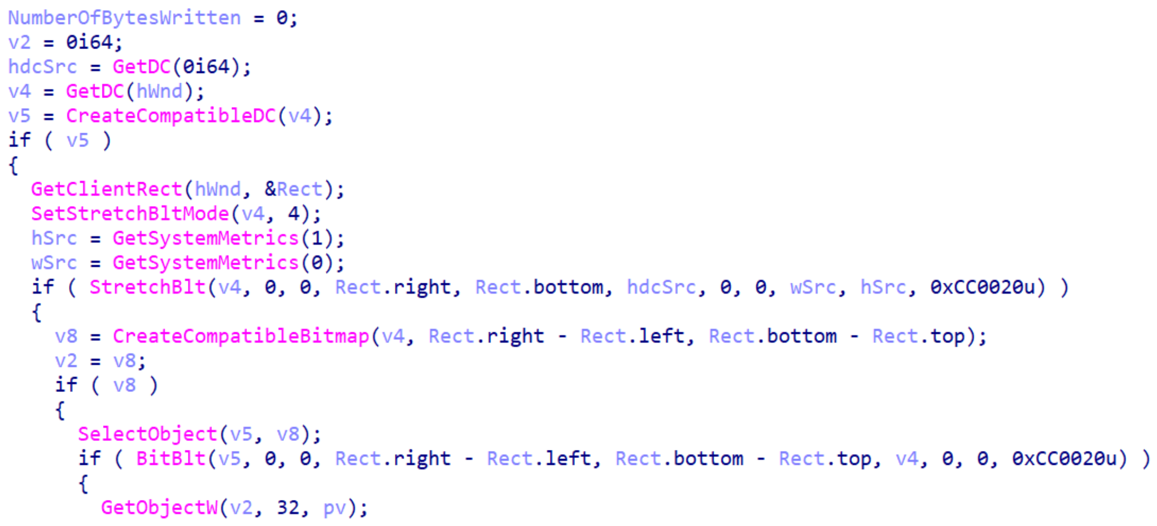


```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  HWND v3; // rax

  v3 = GetDesktopWindow();
  sub_140001030(v3);
  return 0;
}
```

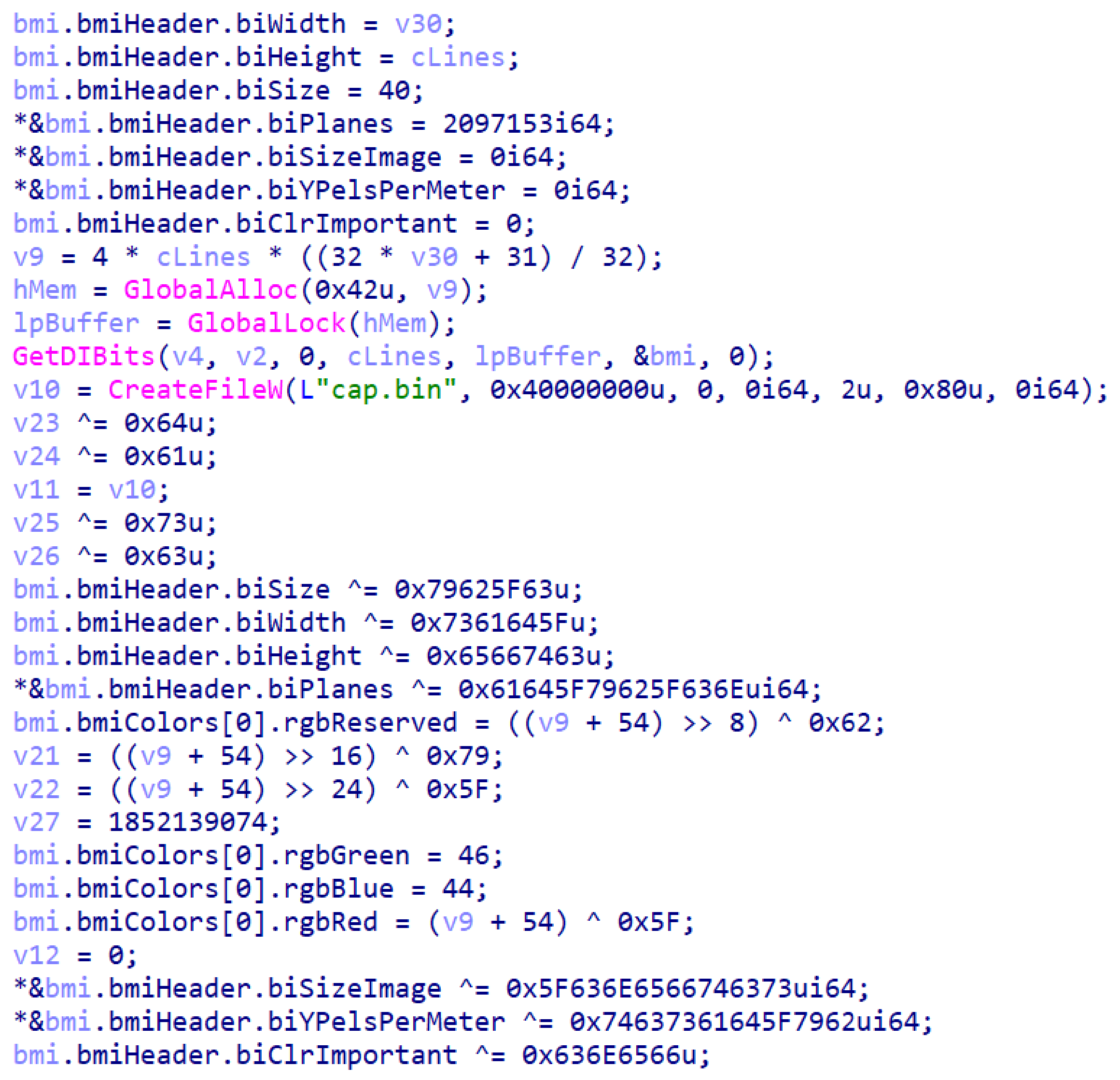
里面是获取当前窗口的句柄

跟进sub_14001030



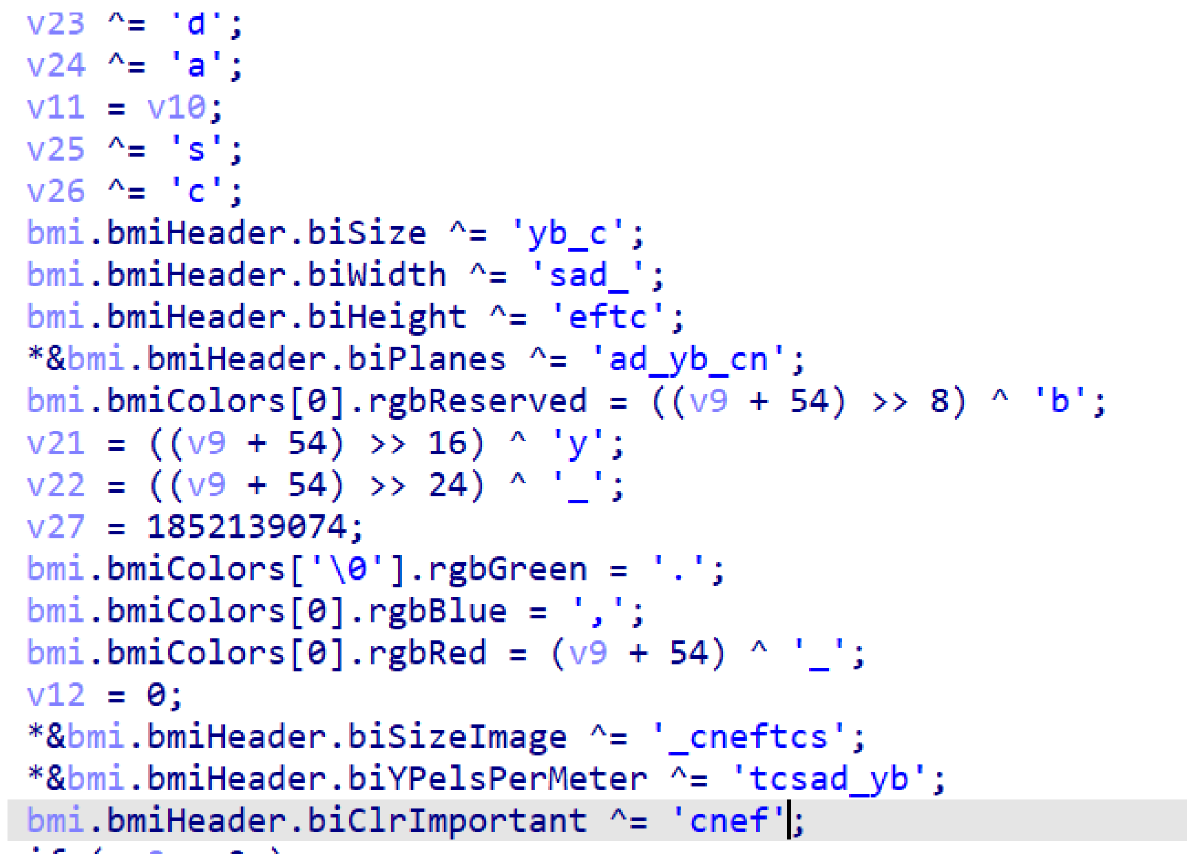
```
NumberOfBytesWritten = 0;
v2 = 0i64;
hdcSrc = GetDC(0i64);
v4 = GetDC(hWnd);
v5 = CreateCompatibleDC(v4);
if ( v5 )
{
  GetClientRect(hWnd, &Rect);
  SetStretchBltMode(v4, 4);
  hSrc = GetSystemMetrics(1);
  wSrc = GetSystemMetrics(0);
  if ( StretchBlt(v4, 0, 0, Rect.right, Rect.bottom, hdcSrc, 0, 0, wSrc, hSrc, 0xCC0020u) )
  {
    v8 = CreateCompatibleBitmap(v4, Rect.right - Rect.left, Rect.bottom - Rect.top);
    v2 = v8;
    if ( v8 )
    {
      SelectObject(v5, v8);
      if ( BitBlt(v5, 0, 0, Rect.right - Rect.left, Rect.bottom - Rect.top, v4, 0, 0, 0xCC0020u) )
      {
        GetObjectW(v2, 32, pv);
```

一大堆系统调用，实现的是读取之前传入的句柄，然后创建一张位图，获取位图的宽高存进结构体里。



```
bmi.bmiHeader.biWidth = v30;
bmi.bmiHeader.biHeight = cLines;
bmi.bmiHeader.biSize = 40;
*&bmi.bmiHeader.biPlanes = 2097153i64;
*&bmi.bmiHeader.biSizeImage = 0i64;
*&bmi.bmiHeader.biYPelsPerMeter = 0i64;
bmi.bmiHeader.biClrImportant = 0;
v9 = 4 * cLines * ((32 * v30 + 31) / 32);
hMem = GlobalAlloc(0x42u, v9);
lpBuffer = GlobalLock(hMem);
GetDIBits(v4, v2, 0, cLines, lpBuffer, &bmi, 0);
v10 = CreateFileW(L"cap.bin", 0x40000000u, 0, 0i64, 2u, 0x80u, 0i64);
v23 ^= 0x64u;
v24 ^= 0x61u;
v11 = v10;
v25 ^= 0x73u;
v26 ^= 0x63u;
bmi.bmiHeader.biSize ^= 0x79625F63u;
bmi.bmiHeader.biWidth ^= 0x7361645Fu;
bmi.bmiHeader.biHeight ^= 0x65667463u;
*&bmi.bmiHeader.biPlanes ^= 0x61645F79625F636Eui64;
bmi.bmiColors[0].rgbReserved = ((v9 + 54) >> 8) ^ 0x62;
v21 = ((v9 + 54) >> 16) ^ 0x79;
v22 = ((v9 + 54) >> 24) ^ 0x5F;
v27 = 1852139074;
bmi.bmiColors[0].rgbGreen = 46;
bmi.bmiColors[0].rgbBlue = 44;
bmi.bmiColors[0].rgbRed = (v9 + 54) ^ 0x5F;
v12 = 0;
*&bmi.bmiHeader.biSizeImage ^= 0x5F636E6566746373ui64;
*&bmi.bmiHeader.biYPelsPerMeter ^= 0x74637361645F7962ui64;
bmi.bmiHeader.biClrImportant ^= 0x636E6566u;
```

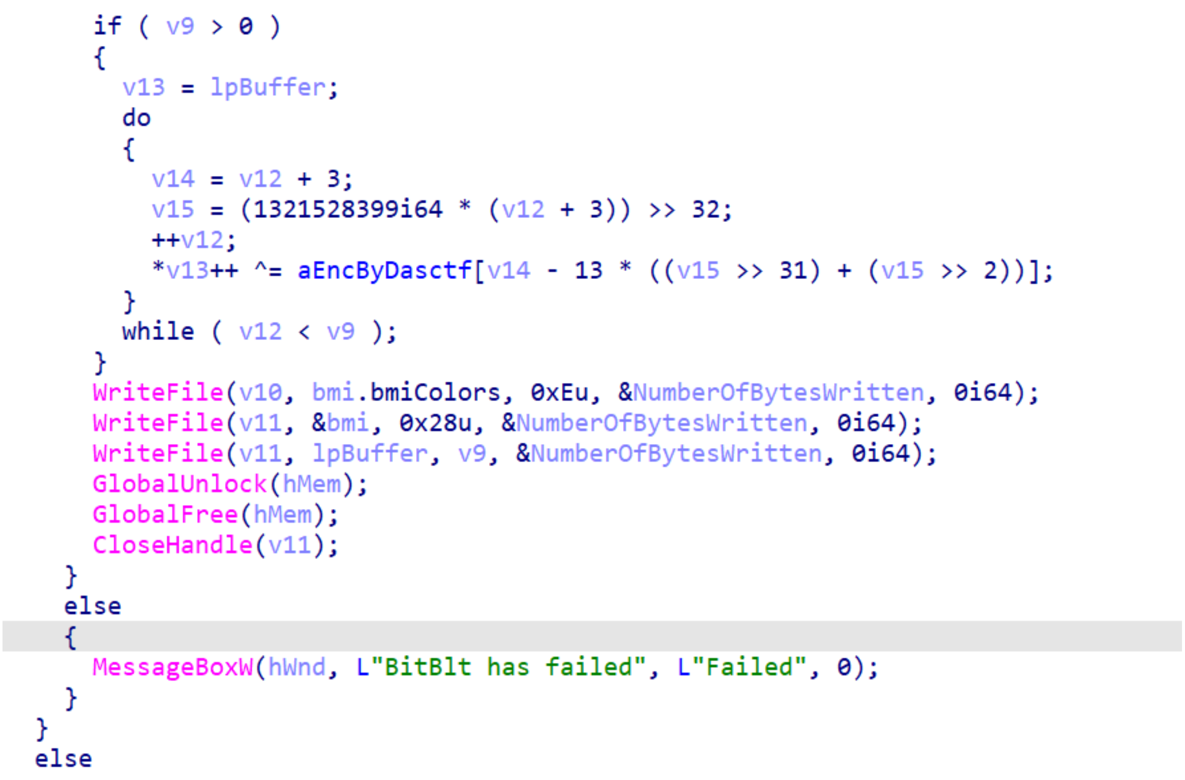
这里就是对对应数据的加密



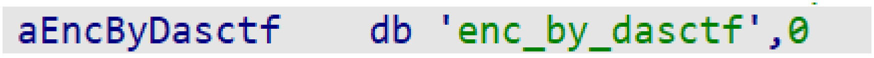
```
v23 ^= 'd';
v24 ^= 'a';
v11 = v10;
v25 ^= 's';
v26 ^= 'c';
bmi.bmiHeader.biSize ^= 'yb_c';
bmi.bmiHeader.biWidth ^= 'sad_';
bmi.bmiHeader.biHeight ^= 'eftc';
*&bmi.bmiHeader.biPlanes ^= 'ad_yb_cn';
bmi.bmiColors[0].rgbReserved = ((v9 + 54) >> 8) ^ 'b';
v21 = ((v9 + 54) >> 16) ^ 'y';
v22 = ((v9 + 54) >> 24) ^ '_';
v27 = 1852139074;
bmi.bmiColors['\0'].rgbGreen = '.';
bmi.bmiColors[0].rgbBlue = ',';
bmi.bmiColors[0].rgbRed = (v9 + 54) ^ '_';
v12 = 0;
*&bmi.bmiHeader.biSizeImage ^= '_cneftcs';
*&bmi.bmiHeader.biYPelsPerMeter ^= 'tcsad_yb';
bmi.bmiHeader.biClrImportant ^= 'cnef';
```

全部R一遍，就能发现其实是后面的key值(aEncByDasctf)

而且是按顺序的(处理一下大小端序和bmp文件头的偏移量)



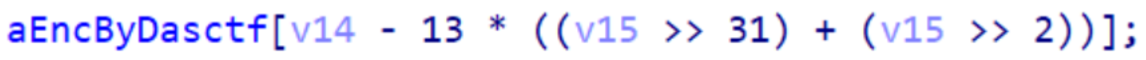
```
      if ( v9 > 0 )
      {
        v13 = lpBuffer;
        do
        {
          v14 = v12 + 3;
          v15 = (1321528399i64 * (v12 + 3)) >> 32;
          ++v12;
          *v13++ ^= aEncByDasctf[v14 - 13 * ((v15 >> 31) + (v15 >> 2))];
        }
        while ( v12 < v9 );
      }
      WriteFile(v10, bmi.bmiColors, 0xEu, &NumberOfBytesWritten, 0i64);
      WriteFile(v11, &bmi, 0x28u, &NumberOfBytesWritten, 0i64);
      WriteFile(v11, lpBuffer, v9, &NumberOfBytesWritten, 0i64);
      GlobalUnlock(hMem);
      GlobalFree(hMem);
      CloseHandle(v11);
    }
    else
    {
      MessageBoxW(hWnd, L"BitBlt has failed", L"Failed", 0);
    }
  }
  else
```



```
aEncByDasctf    db 'enc_by_dasctf',0
```

后面的 do循环就是对图像数据的加密

里面的



```
aEncByDasctf[v14 - 13 * ((v15 >> 31) + (v15 >> 2))];
```

对于key值下标的index处理其实是一个混淆，如果你把数据放下来跑一下就会发现是一个

5-12 0-12 ·········0-12 的循环

后面就是把加密的数据写到生成文件cap.bin里

所以整体的逻辑就是截屏然后生成一张加密的bmp

把文件异或回去就好了

```
key = b"enc_by_dasctf"
f=open('cap.bin', 'rb')
cry_data=f.read()
cnt=1
data=[]
for i in cry_data:
    data.append(i^key[cnt%13])
    cnt+=1
fp=open("flag.bmp",'wb')
fp.write(bytes(data))
```