Ntleas 代码说明

By littlewater

2014-04-26

前言

Ntleas 目前已经趋于稳定，但是由于 API 的数量之多以及各种游戏对 API 的使用各不相同，转区的维护等同于一个对 WINDOWS API 不断兼容修补工作，这不仅需要对 WINAPI 很熟悉，而且要对其各种手法都有办法照顾周全。

考虑到维护的成本，以及 HOOK 的技巧，NTLEAS 的作者个人认为开源还是有一定意义的，这一方面方便技术实现的交流，另外一方面也可以通过更多有热情有技术的朋友参与进来。

没有 NTLEA 的开源显然不会有 NTLEAS 的后继，现在啥名字都习惯套上一个 OPEN 表示开源，这里就祝愿 OPEN NTLEA 的将来光明和美好。

阅读注意

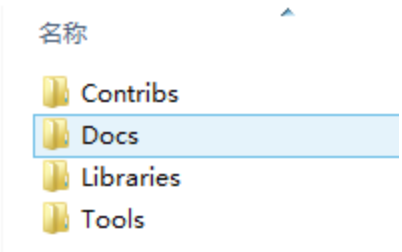
1. 这个文档主要是提供需要使用 ntleas 代码编译或者完善 ntleas 功能开发者使用，如果需要使用 ntleas 则不应该翻阅本文档而应该查阅 ntleas 的使用说明；
2. NTLEA 的全名为 NT LOCALE EMULATOR ADVACE，原内核作者为 LoveHina Advance（AVC），后续维护作者为 Magami（七夜真神），NTLEAS 为通过 AVC 和 MAGAMI 认可的 NTLEA 重制版本，新界面和新内核均为 littlewater（水水）所编写；
3. NTLEA 主站页面目前是：https://ntlea.codeplex.com/，欢迎和作者联系交流技术话题；

本文目的

鉴于 ntlea 原版本的文档资料基本没有，因此在改写使用 C 语言开发的基础上，通过本文一并提供 NTLEA 实现的大致思路以及调试方法。

编译说明

下载到源代码（或者解压缩）以后，目录应该为如下结构：



其中：

Contribs 和 Libraries 里面包含了第三方现成库或者从其他公开网站中调整得到的代码库，Docs 就是本文档所在位置，Tools 内主要就是 ntleas 的核心代码。
要进行编译，首先需要安装好 Microsoft VisualStudio 2013 Express 或者 2013 Professional 或者 2013 Ultimate 版本，然后直接进入 Tools 的 ntlea 目录下双击 ntleas.sln 打开进行编译即可，所需要的依赖库应该已经配齐。

整体组成

Ntlea 的第三方依赖主要包括两大部分，其一是 hook 库包括 mhook 和 minhook，其二就是辅助功能库，包括 locale 时区等的枚举库、崩溃信息库、win32 界面库、shortcut 辅助创建库；

Ntlea 的核心部分包括：
1) Ntdebug，打印调试信息用，基本上后期都没什么作用
2) Nthook，主要实现各种被 HOOK API 转码功能的库
3) Nthookfun，通用实现和调用 API HOOK 的代码，但是 ntleai 除外
4) Ntproc，实现子进程创建以及注入逻辑的代码

Ntlea 的实体部分包括：
1) Ntleai，基本遵循原 ntleah 的实现，是目前 X86 优先选择的版本
2) Ntleaj，基于 minhook 的实现，由于 minhook 兼容性不是太好，有一部分程序无法通过 minhook 正常转码运行；
3) Ntleak，基于 mhook 的实现，效果比 minhook 好不少，但是有一些游戏兼容性还是不好，由于完全为 C/C++实现，因此可以支持原生 X64 平台，在 X64 平台下 nleak 为默认选择的版本；
4) Ntleas，基本遵循原 ntleac 的实现，包括 X86 和 X64 版本，但是各个版本必须正确对应各自的平台使用；

Ntlea 的外围部分包括：
1) minicrt，只是用来实现 CRT 必须部分功能的代码，减少对 CRT 的依赖
2) ntleasCtx，仅仅是一个符合右键菜单规范的空壳，主要是考虑到升级通常 explorer 会占住 DLL 替换，因此大部分而言无需替换该组件，只需要替换功能 DLL 即可
3) ntleasCtxExt，在 ntleasCtx 的基础上实现右键功能的 DLL 扩展，更新内容只需要替换该 DLL 即可
4) ntleasig，配合程序调试用的开关小程序
5) ntprofile，配置单元的序列化和反序列化辅助库
6) ntleasWin，配置 Profile 的界面程序

流程描述

ntleas 是代码的启动入口，通过 ntleas 配置好转区信息到一个内存映射保存，之后启动目标进程并使其暂停（参考 ntproc 部分），对于目标进程不为 EXE 通过查询对应的 EXE 来启动关联程序，在目标进程启动后，通过远程注入的方式把 ntleai（或者其他组件）注入到目

标进程中，此时目标进程会自动执行 DLL 初始化代码，即实际调用 nthook 的 InitUnicodeLayer，在 InitUnicodeLayer 中完成 API 的接管（mhook 和 minhook 或者以后的其他库，均通过 nthookfun 来完成接管，ntleai 则按照 ntlea 原来的方式进行 hook）。随后通知 ntleas 已经完成，ntleas 收尾并且恢复目标进程的执行。之后所有被 HOOK 的 API 即可以在运行期实现接管转区功能。

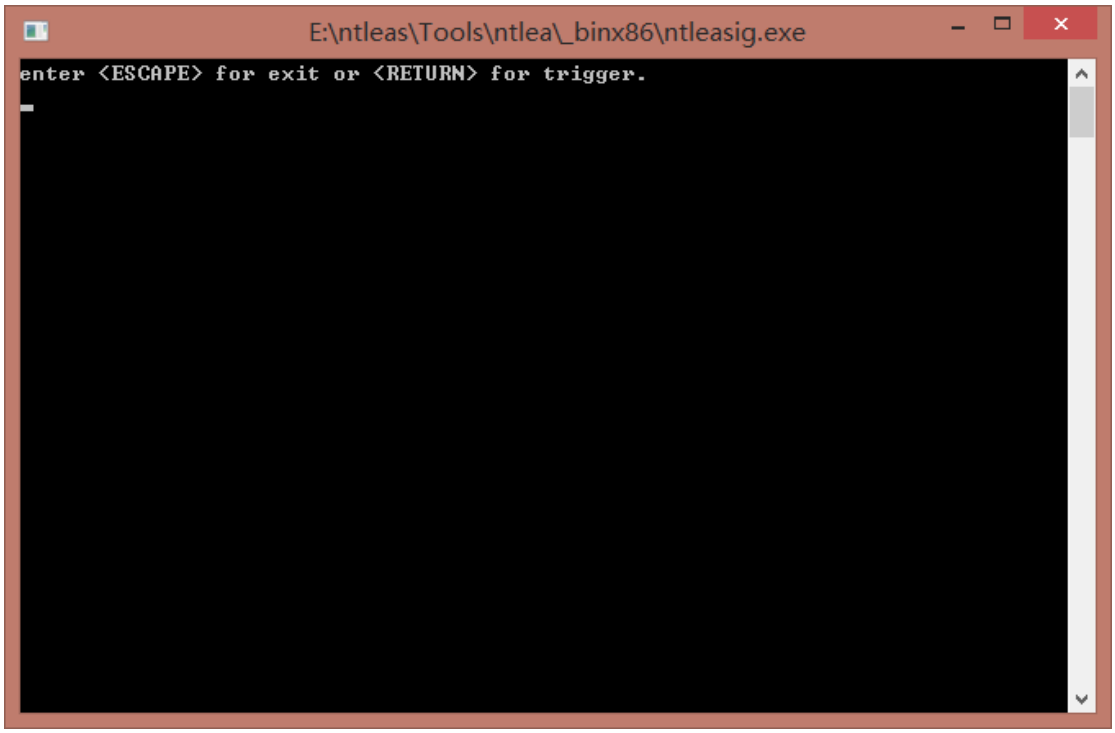
这里就不特别另配代码阅读说明了。

如何调试

Ntleas 可以通过 VisualStudio 进行源码调试（当然习惯了 windbg 或者 OD 的请随意=w=），其实外围部分无所谓，主要是针对 ntleas 以及相应库 ntleai 为主的兼容性调整，一般而言，改动代码主要发生在 nthook 中。

这里以 windows 自带的 notepad.exe 为例子，总结一下调试步骤：

1) 开启 ntleasig，ntleasig 会打开调试事件，在 ntleas 开启后会检测是否有特定 ntlea 的事件，如果存在则会一直保持等待，通过 ntleasig 按下回车可以让 ntleas 继续运行（注意第一次是让 ntleas 走完注入流程，再一次才是让 ntleas 结束退出）；



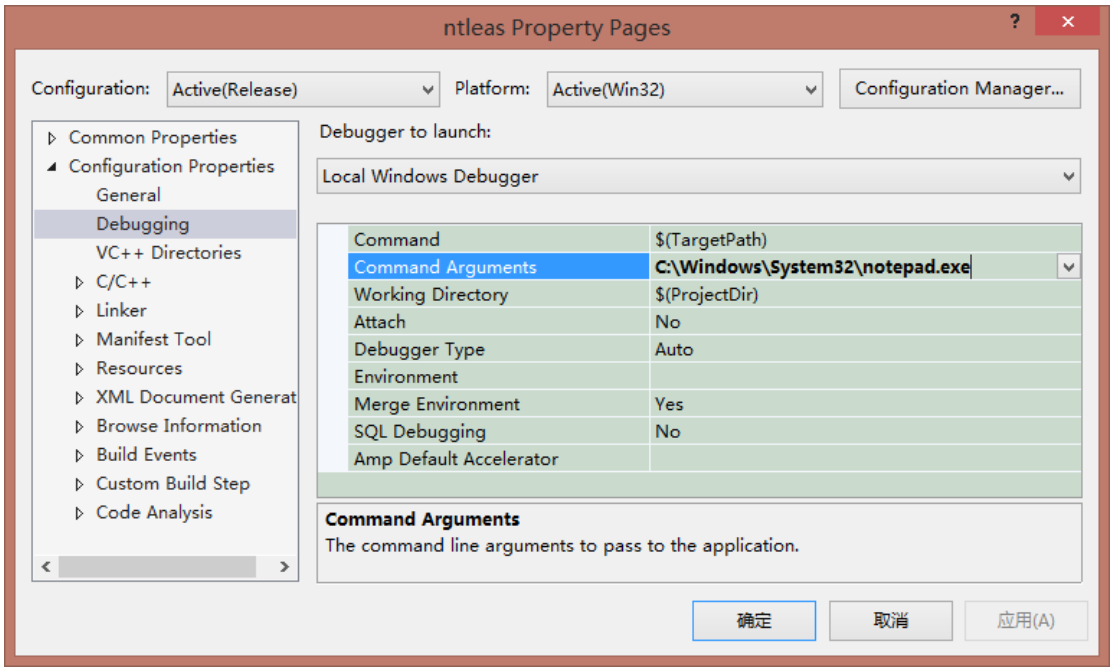
图表 1 ntlea 信号控制工具

2) 设置好需要调试的断点，ntlea 的 RPCHOOK 插件，主要是两个部分，其一为初始化即 API HOOK 部分，其二为 HOOK 的逻辑实现部分，ntleasig 也是根据这两个对子进程会有两次中断，可以根据调试的需要进行断点设置；

```
    HookDllFunc((LPCSTR)(DWORD_PTR)GetMenuStringA, (LPVOID)(DWORD_PTR)HookGetMenuString, NULL);
#ifdef Testing
    HookDllFunc((LPCSTR)(DWORD_PTR)GetMenuItemInfoA, (LPVOID)(DWORD_PTR)HookGetMenuItemInfo, NULL);
    HookDllFunc((LPCSTR)(DWORD_PTR)SetMenuItemInfoA, (LPVOID)(DWORD_PTR)HookSetMenuItemInfo, NULL);
#endif

    HookDllFunc((LPCSTR)(DWORD_PTR)IsDBCSLeadByte, (LPVOID)(DWORD_PTR)HookIsDBCSLeadByte, NULL);
    HookDllFunc((LPCSTR)(DWORD_PTR)CharPrevA, (LPVOID)(DWORD_PTR)HookCharPrev, NULL);
    HookDllFunc((LPCSTR)(DWORD_PTR)CharNextA, (LPVOID)(DWORD_PTR)HookCharNext, NULL);
    addresses.lpEnumFontFamiliesExA = (LPBYTE)HookDllFunc((LPCSTR)(DWORD_PTR)EnumFontFamiliesExA, (LPVOID)(DWORD_PTR)HookEnumFontFam
        /*+ (psettings->nOSVer == VER_WINXP_SP2_OR_ABOVE ? 0 : 1)*/;
    addresses.lpEnumFontFamiliesExW = (LPBYTE)HookDllFunc((LPCSTR)(DWORD_PTR)EnumFontFamiliesExW, (LPVOID)(DWORD_PTR)HookEnumFontFam

    addresses.lpEnumFontsA = HookDllFunc((LPCSTR)(DWORD_PTR)EnumFontsA, (LPVOID)(DWORD_PTR)HookEnumFontsA, NULL);
    addresses.lpEnumFontsW = HookDllFunc((LPCSTR)(DWORD_PTR)EnumFontsW, (LPVOID)(DWORD_PTR)HookEnumFontsW, NULL);
    HookDllFunc((LPCSTR)(DWORD_PTR)EnumFontFamiliesA, (LPVOID)(DWORD_PTR)HookEnumFontFamiliesA, NULL);
    HookDllFunc((LPCSTR)(DWORD_PTR)EnumFontFamiliesW, (LPVOID)(DWORD_PTR)HookEnumFontFamiliesW, NULL);

    HookDllFunc((LPCSTR)(DWORD_PTR)CreateFontIndirectA, (LPVOID)(DWORD_PTR)HookCreateFontIndirect, NULL);
    addresses.lpGetStockObjectAddress = (LPBYTE)HookDllFunc((LPCSTR)(DWORD_PTR)GetStockObject, (LPVOID)(DWORD_PTR)HookGetStockObject
}
```

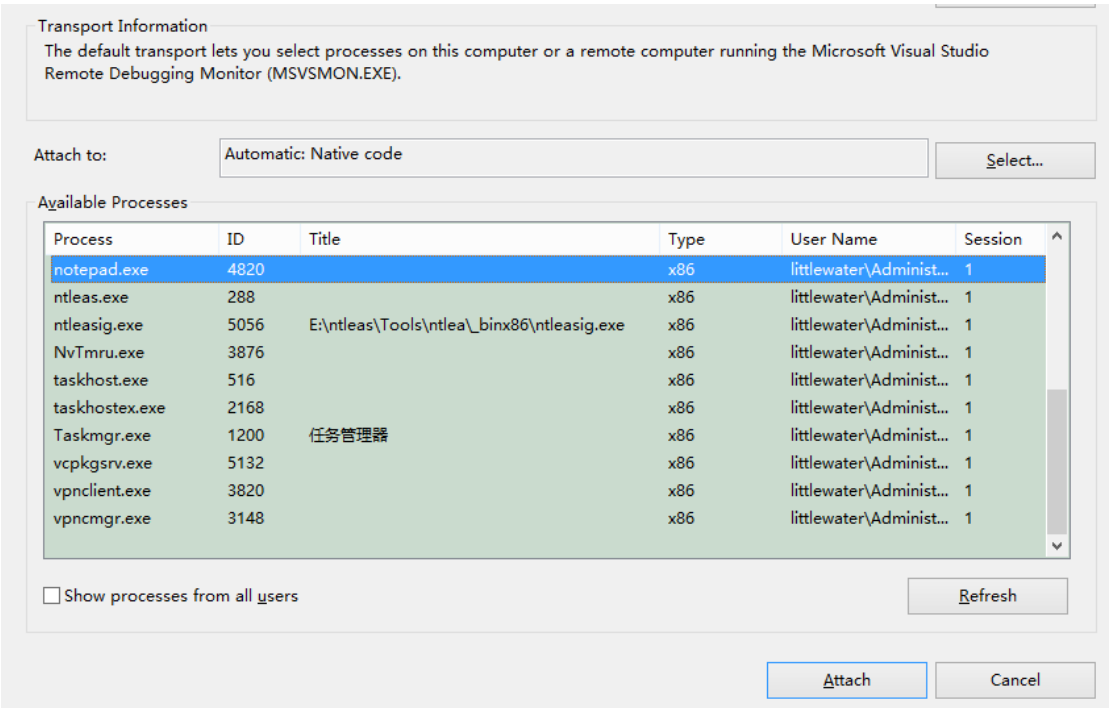
图表 2 设置断点

3) 开启 VisualStudio，选择 Debug 进行调试（release 也可以，但是 debug 会弹出必要调试信息控制台窗口，而 release 被设置为窗口模式启动，不会弹出控制台对话框），配置好启动参数后，直接运行（不是调试运行），主要是 ntlea 先通过 ntleas 的进程将 DLL 组件远程注入到目标进程中，主要调试的是目标进程；



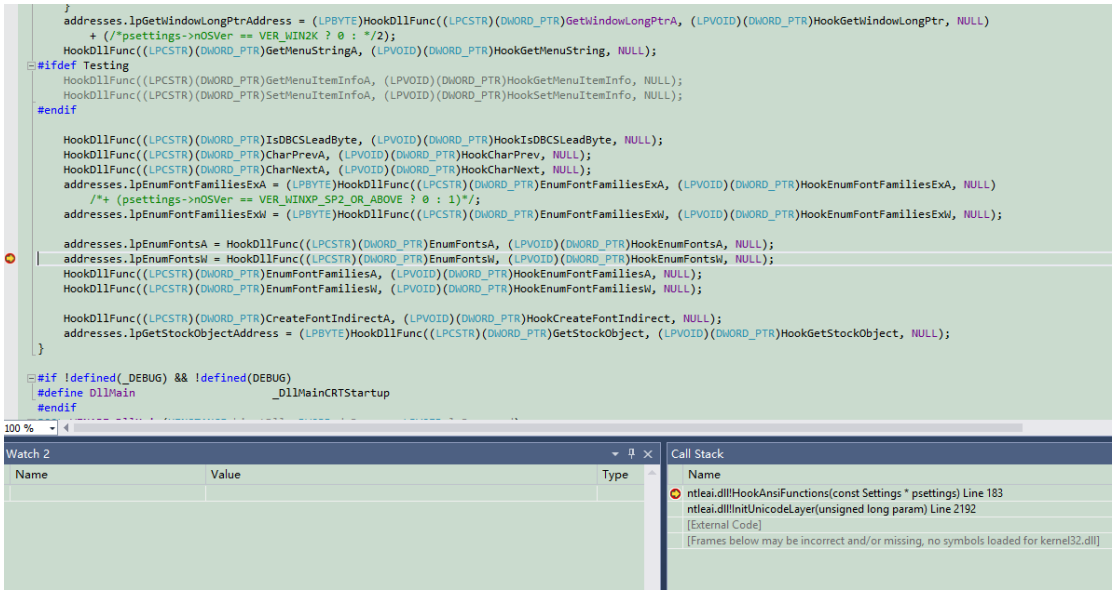
图表 3 参数配置 release 和 debug 均可

4) 通过 VisualStudio 的进程关联列表，选择需要调试的进程，这里选择 notepad.exe

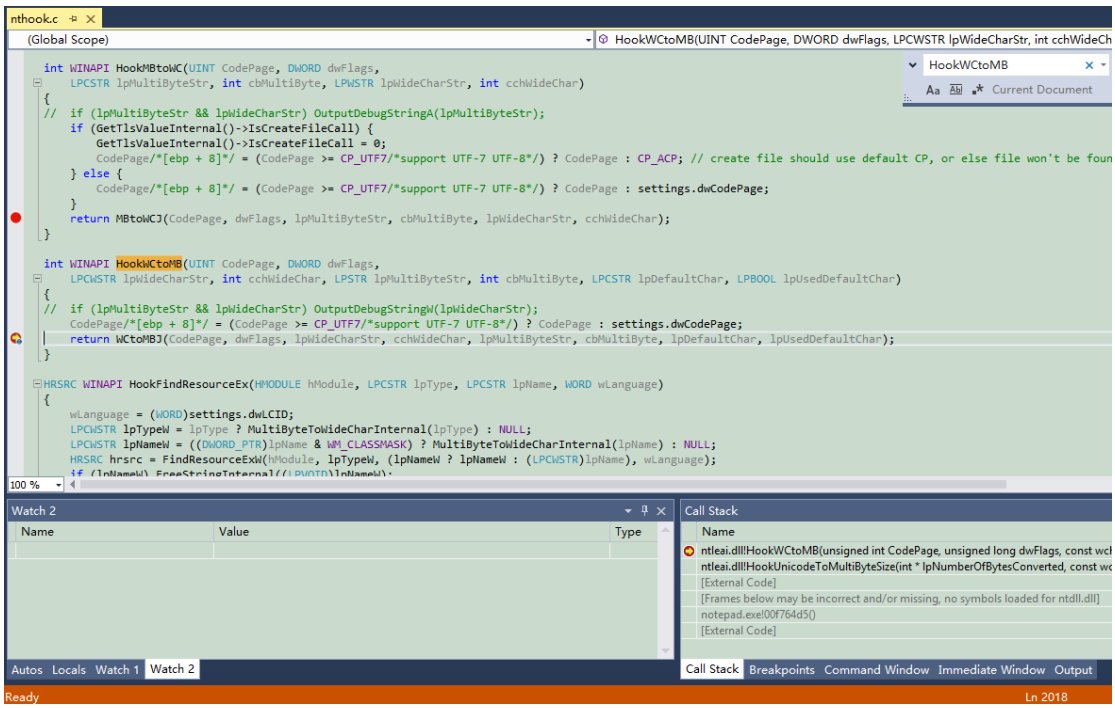


图表 4 选择等待 hook 的进程

5) 之后只需要根据调试的需要对 ntleas 拍回车即可，如果需要调试初始化部分（参考 nthook 的 InitUnicodeLayer 函数，在拍回车之后即会运行，并且停止在进程恢复之前，再拍一次回车就会恢复进程运行，此后各个 hook 会被调用；

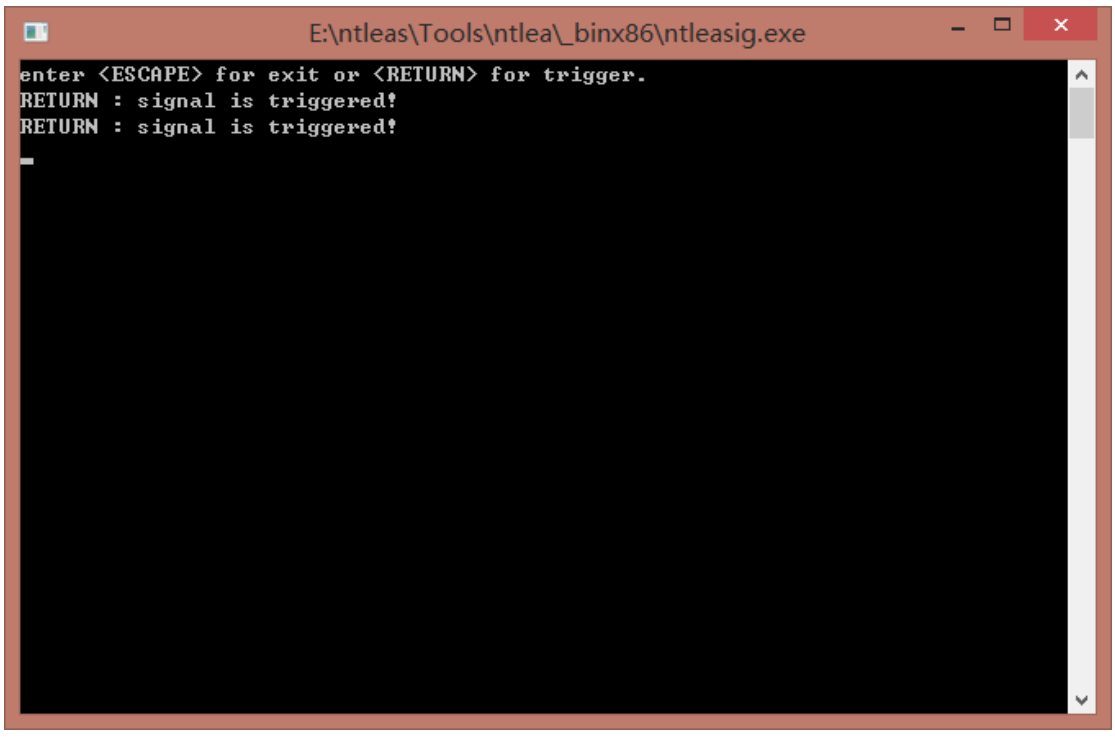


图表 5 在初始化部分断下



图表 6 在 HOOK 处理逻辑中断下

6) Ntleasig 可以一直开着复用，由于是自动事件实现，任何多次连续拍回车仅限一次触发，触发后一次在被充值前保持有效，退出使用 ESC 离开；



图表 7 两次回车即恢复进程运行

说明事项

- Ntleasig 不慎关闭，而进程没有开启，可以再次打开 ntleasig 将其启动（当然也可以通过任务管理器直接关闭）；

- 由于目标进程可能创建子进程，子进程也会受到 ntleasig 的影响，调试时应该注意；
- NTLEAS 未来如果有分支版本，建议通过主站和作者进行联系以免版本过多混乱；
- 不建议在 NTLEAS 上使用.NETFRAMEWORK 组件（当然可以也是可以的）；
- 多语言版本功能后期可能会考虑使用 gettext 进行实现；
- ntleai.dll 能够实现其对应的 X64 版本，主要是一来 VC 不支持 X64 内联，二来对 X64 汇编又不熟悉；