无痕HOOK方式=硬断+VEH

转载

YuHengZuo 于 2017-11-24 23:44:43 发布 8696 收藏 10
分类专栏： DLL注入


DLL注入 专栏收录该内容
2 篇文章 1 订阅
订阅专栏

无痕HOOK方式=硬断+VEH

```
BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        {
            AddVectoredExceptionHandler(1, (PVECTORED_EXCEPTION_HANDLER)ExceptionHandler);
            SetHwBreakpoint();
        }
    case DLL_THREAD_ATTACH:
        {
            SetHwBreakpoint();
        }
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}

void SetHwBreakpoint()
{
    CONTEXT ctx;
    ctx.ContextFlags = CONTEXT_ALL;
    GetThreadContext(GetCurrentThread(), &ctx);
    ctx.Dr0 = 0x6f3a20dd;
    ctx.Dr1 = 0x6f361f7b;
    ctx.Dr7 = 0x405;
    SetThreadContext(GetCurrentThread(), &ctx);
}

DWORD NTAPI ExceptionHandler(EXCEPTION_POINTERS * ExceptionInfo)
{
    if ((DWORD)ExceptionInfo->ExceptionRecord->ExceptionAddress == 0x6f3a20dd)
    {
        //直接改eip模拟jmp
        ExceptionInfo->ContextRecord->Eip += 0x34;
        return EXCEPTION_CONTINUE_EXECUTION;
    }
    else if ((DWORD)ExceptionInfo->ExceptionRecord->ExceptionAddress == 0x6f361f7b)
    {
        //直接设eax为零模拟mov eax,0
        ExceptionInfo->ContextRecord->Eax = 0;
        ExceptionInfo->ContextRecord->Eip += 5;
        return EXCEPTION_CONTINUE_EXECUTION;
    }
    else
    {
        //在异常handler里重设drx防止断点被意外清除
        ExceptionInfo->ContextRecord->Dr0 = 0x6f3a20dd;
        ExceptionInfo->ContextRecord->Dr1 = 0x6f361f7b;
        ExceptionInfo->ContextRecord->Dr7 = 0x405;
        return EXCEPTION_CONTINUE_SEARCH;
    }
}
```