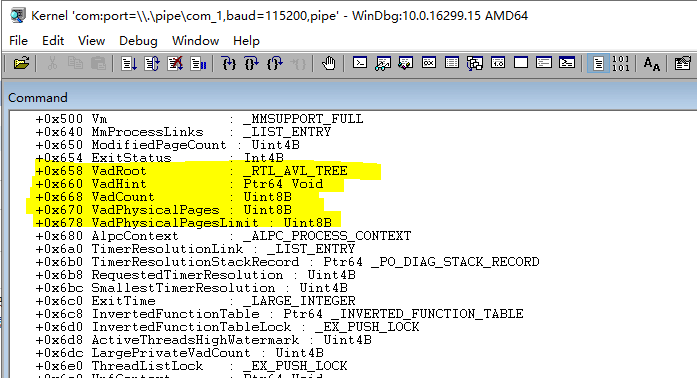
在进程的 \_EPROCESS 中有一个 \_RTL\_AVL\_TREE 类型的 VadRoot 成员，它是一个存放进程内存块的二叉树结构，如果我们找到了这个二叉树中我们想要隐藏的内存，直接将这个内存在二叉树中 抹去 ，其实是让上一个节点的 EndingVpn 指向下个节点的 EndingVpn ，类似于摘链隐藏进程，就可以达到隐藏的效果。

通过 dt \_EPROCESS 得到EProcess结构 VadRoot 如下:



例如当调用 VirtualAlloc 分配内存空间。



#include <iostream> #include <Windows.h>

int main(int argc, char argv[])

{

LPVOID p1 = VirtualAlloc(NULL, 0x10000, MEM\_COMMIT, PAGE\_READWRITE);

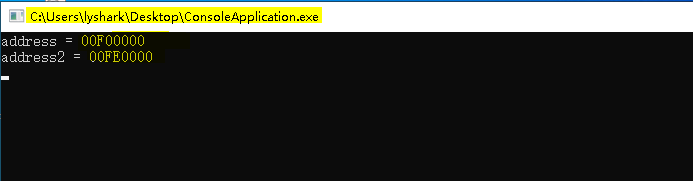
LPVOID p2 = VirtualAlloc(NULL, 0x10000, MEM\_COMMIT, PAGE\_EXECUTE\_READWRITE);

std::cout << "address = " << p1 << std::endl; std::cout << "address2 = " << p2 << std::endl;

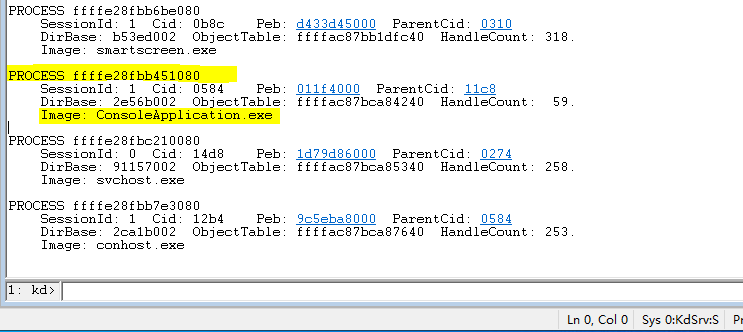
getchar(); return 0;

}

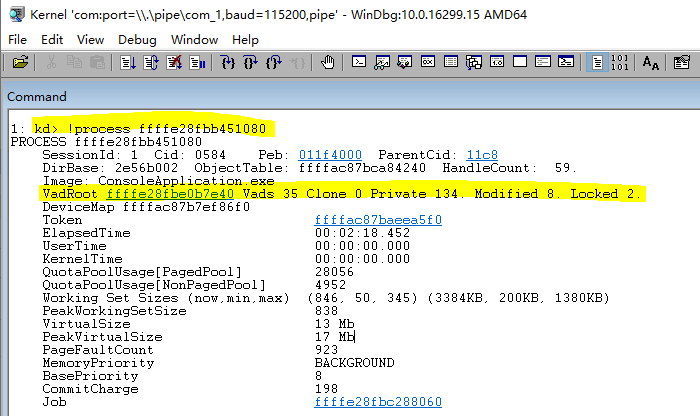
运行程序得到两个内存地址 0xf00000 和 0xfe0000



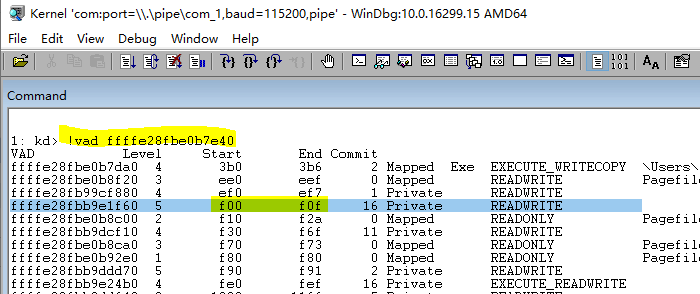
通过 !process 0 0 枚举所有进程，并得到我们所需进程的EProcess地址。



检查进程 !process ffffe28fbb451080 得到 VAD 地址 ffffe28fbe0b7e40



此处以 0xf00000 为例，这里我们看到 windbg 中的值和进程中分配的内存地址并不完全一样，这是因为 x86 cpu 默认内存页大小 4k 也就是 0x1000 ，所以这里还要再乘以 0x1000 才是真正的内存地址。



所以计算结果刚好等于 0xf00000



而隐藏进程内特定内存段核心代码在于 p1->EndingVpn = p2->EndingVpn; 将VAD前后节点连接。

PMMVAD p1 = vad\_enum((PMMVAD)VadRoot, 0x3a0); // 遍历第一个结点PMMVAD p2 = vad\_enum((PMMVAD)VadRoot, 0x3b0); // 遍历找到第二个结点if (p1 && p2)

{

p1->EndingVpn = p2->EndingVpn; // 将第二个结点完全隐藏起来

}

本书作者： 王瑞 (LyShark)

作者邮箱： m [e@lyshark.com](mailto:e@lyshark.com)

作者博客： h ttps://lyshark.cnblogs.com

团队首页： w ww.lyshark.com