追根溯源DLL木马进程内幕大揭密 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/460/2021_2022__E8_BF_BD_E6_A0_B9_E6_BA_AF_E6_c98_460941.htm 如果是位经常玩木马的朋友，那么一般情况下都会或多或少掌握一些木马的特性，然而，很多朋友还是不知道“ DLL木马 ”是什么东东。那到底什么是“ DLL木马 ”呢?它与一般的木马又有什么不同?带着这些疑问，一起开始这次揭密之旅吧! 一、追根溯源 从DLL说起 要了解什么是“ DLL木马 ”，就必须知道“ DLL ”是什么意思!说起DLL，就不能不涉及到久远的DOS时代。在DOS大行其道的时代，写程序是一件繁琐的事情，因为每个程序的代码都是需要独立的，这时为了实现一个普通的功能，甚至都要为此编写很多代码。后来随着编程技术发展与进步，程序员们开始把很多常用的代码集合(也就是通用代码)放进一个独立的文件里，并把这个文件称为“ 库 ”(Library)。在写程序的时候，把这个库文件加入编译器，就能使用这个库包含的所有功能而不必自己再去写一大堆代码，这个技术被称为“ 静态链接 ”(Static Link)。静态链接技术让劳累的程序员松了口气，一切似乎都很美好。然而静态链接技术的最大缺陷就是极度消耗和浪费资源，当一个程序只想用到一个库文件包含的某个图形效果时，系统将把这个库文件携带的所有的图形效果都加入程序，这样就使得程序非常臃肿。虽然这并不重要，可是这些臃肿的程序却把道路都阻塞了静态链接技术让最终的程序成了大块头，因为编译器把整个库文件都加载进去了。 技术永远是在发展的，静态链接技术由于无法避免的弊端，不能满足程序员和编程的需要

，人们开始寻找一种更好的方法来解决代码重复的难题。随着Windows系统的出现， Windows系统使用一种被称为“ 动态链接库 ”(Dynamic Link Library)的新技术，它同样也是使用库文件，DLL的名字就是这样来的。动态链接本身和静态链接没什么区别，也是把通用代码写进一些独立文件里，但是在编译方面，微软把库文件做成已经编译好的程序文件，给它们开发一个交换数据的接口。程序员编写程序的时候，一旦要使用某个库文件的一个功能函数，系统就把这个库文件调入内存，连接上这个程序占有的任务进程，然后执行程序要用的功能函数，并把结果返回给程序显示出来。完成需要的功能后，这个DLL停止运行，整个调用过程结束。微软让这些库文件能被多个程序调用，实现了比较完美的共享，程序员无论要写什么程序，只要在代码里加入对相关DLL的调用声明就能使用它的全部功能。这样，写出来的程序就不能再携带一大堆无用的垃圾了。 DLL技术的诞生，使编写程序变成一件简单的事情，Windows为我们提供了几千个函数接口，足以满足大多数程序员的需要。而且，Windows系统自身就是由几千个DLL文件组成，这些DLL相互扶持，组成了庞大的Windows系统。如果Windows依然使用静态链接技术，那将是不可想象的。 二、什么是API 在前面提到的“ 接口 ”又是什么呢?因为DLL不能像静态库文件那样塞进程序里，如何让程序知道实现功能的代码和文件成了问题，微软就为DLL技术做了标准规范，为每个DLL文件都明确地标注好它的功能名称，程序只要根据标准规范找到相关的名称进行调用就行了，这就是API(Application Programming Interface)应用程序接口，每个DLL带的接口都不尽相同，最大限度地减少了程

序代码的重复。在Windows里，最基本的3个DLL文件是kernel32.dll、user32.dll、gdi32.dll。它们共同构成了基本的系统框架。 三、DLL与木马 DLL是编译好的代码，与一般程序没什么大差别，只是它不能独立运行，需要程序调用。那么，DLL与木马能扯上什么关系呢?如果你学过编程并且写过DLL，就会发现，其实DLL的代码和其他程序几乎没什么两样，仅仅是接口和启动模式不同，只要改动一下代码入口，DLL就变成一个独立的程序了。 当然，DLL文件是没有程序逻辑的，其实DLL并不等于EXE。不过，依然可以把DLL看做缺少了main入口的程序，DLL带的各个功能函数可以看作一个程序的几个函数模块。DLL木马就是把一个实现了木马功能的代码，加上一些特殊代码写成DLL文件，导出相关的API，在别人看来，这只是一个普通的DLL，但是这个DLL却携带了完整的木马功能，这就是DLL木马的概念。也许有人会问，既然同样的代码就可以实现木马功能，那么直接做程序就可以，为什么还要多此一举写成DLL呢?这是为了隐藏，因为DLL运行时是直接挂在调用它的程序的进程里的，并不会另外产生进程，所以相对于传统EXE木马来说，它很难被查到。 四、DLL的运行 虽然DLL不能自己运行，可是Windows在加载DLL的时候，需要一个入口函数，就如同EXE的main一样，否则系统无法引用DLL。所以根据编写规范，Windows必须查找并执行DLL里的一个函数DllMain作为加载DLL的依据，这个函数不作为API导出，而是内部函数。DllMain函数使DLL得以保留在内存里，有的DLL里面没有DllMain函数，可是依然能使用，这是因为Windows在找不到DllMain的时候，会从其它运行库中找一个不做任何操作的

缺省DllMain函数启动这个DLL使它能被载入，并不是说DLL可以放弃DllMain 