看雪熊猫前辈——svchost进程的浅析

转载

BetaBin 于 2012-04-18 20:53:46 发布 2275 收藏
分类专栏： Windows 文章标签： service windows buffer linux内核 microsoft parameters


Windows 专栏收录该内容
40 篇文章 0 订阅
订阅专栏

/*

看雪熊猫前辈讲得太棒了，故转载过来作为学习资料存放。

看雪是个好论坛~~~

原文链接如下：http://bbs.pediy.com/showthread.php?t=127798

（仅做链接资料存放，CSDN格式复制不适合看，就当是图片预览，请移步到看雪。）

*/

标 题: 【原创】svchost进程的浅析
作 者: 熊猫正正
时 间: 2011-01-08,19:10:53
链 接: http://bbs.pediy.com/showthread.php?t=127798

大家好，先扯点闲话吧~~最近比较忙，很少给大家献上点东西，就像看雪上的一位大牛说过：现在我还只是个半瓶水，所以我会写一些“垃圾”放在论坛上，当我满的时候，就看不到我的身影了，确是，现在我还真没看到过那位大牛在写过什么了，也许这就是看雪上的一些元老们都不曾发言的原因吧~~~不管怎么说，我想每个人都是从一无所知，到成为某个领域的专家的，没有谁天生会什么，但我还是希望那些元老们，能在百忙之中，抽点时间出来写一两篇文章给后面也在奋斗的人们吧~~就像潘爱民先生window原理与实践书中前言的最后一句话说的：如果你在年轻的时候，能遇到像王选（像这样类型,我一直很尊敬像钱学森，华罗庚，王选，陈景润，还有很多，这样为自己喜欢的，不断追求，不断努力的人，他们真的值得尊敬）这样的人的指点，那将会能这位年轻人造成多大的影响！（原话我记不得了，反正就是这意思吧），其实真的有时候，也许因为你们的一句话，真的可以改变点什么，所以请不要吝啬你的言语，这只是后辈的一点建议，也没有指针什么人，什么事（就像记得某位牛人说过：当你问问题的时候，别人没有义务一定要回答你），这里我也要说一些后辈了，首先自己在努力一点，不要连1+1=？都要问问别人，很多问题，我想google都已经告诉你了，其实我知道那些牛人们都挺忙的，所以别人有时真的没时间去和你闲扯1+1=?这样的问题！

最后说明一点：咱，小菜一个，不是什么牛人，只是一直在向大牛努力，也许有一天，真会变成一头“牛”，而不做“鸟”了！同时也鼓励一下大家，没有什么技术是你学不会的，只要你想学，坚持学下去，你总有一天会和你曾经崇拜的牛人们一样的，趁我们还年轻，想做什么就坚持做吧！就像《orange's操作系统》的作者说的：当你有个想法时，就去努力做吧！学习贵在坚持，还有一句话我很喜欢：自助者天助，当你坚强勇敢的时候，不说大牛们，连老天爷也会帮你的~~因为我想大牛们也曾经是很坚强勇敢执着的为着自己的理想的人，所以他们更愿意帮助--志同道合的人~~~~~~

最近一直在研究windnow内核驱动，本来想买老毛的<windows内核情景分析>无懒，淘宝价也要一百三十多，买不起，其实买这个价应该还算公道，必竟别人花了三年时间去写，听说内容还不错，驱动开发确是一个需要花大量的时间和精力去研究的东西，相对于驱动开发，我认为一般rootkit之类的技术也不算什么，只是很巧炒的利用了HOOK技术吧了，真正能写出一个完整的操作系统才真的需要很多优秀的人努力才能做得到的，如果你对操作系统底层知识不了解，我敢说你写出来的rootkit基本上没有多大用处，只有对操作系统的底层有了一个很深的认识，才能写出优秀的rootkit，而不是搞几个很简单的Hook就算是rootkit了，所以我个人认为，要想写出一流的rootkit(这里不是鼓励大家编写木马和病毒啦)，你必须花大量的时间去研究操作系统底层的架构和原理，包括很底层的(GDT,LDT,中断等)，但这是需要花大量时间研究的，如果你浮燥，就请趁早放弃，找几本《21天搞定C++》《VC++实例编程》之类的书，花几个月学学，然后找个程序员（其实是外包工作者）的活做做算了！如果你想做一些比较伟大的事，如果你想真正掌屋window和linux，你一定要有足够的耐心和坚持不懈的努力去研究他们的后面的东西，这样你才能成为真正的程序员~~中国这样的人太少了！！！我只是小菜鸟一个，但我会努力朝着之个方向努力，我想也会

有很多志同道合的朋友在研究着，一起努力吧！

每次给大家写个什么东西，都会给大家讲讲大道理，其实不管我说的这些是对的还是错的，对的你就看看，错的你就当我是喝多了，在这瞎放屁~~~上面的话，不针对任何人，任何事，因为我想每个人的追求和理想是不同的，所以我也不好评价过多，（最主要的是我不想因为这篇文章而引起什么口水战之类的，我没时间和你口水战，如果我说错了，你就大人大量，别我计较好了，谢谢）

不知不觉就说了这么多，就当2010年新年献文吧，呵呵，好了，下面正式进入今天的主题吧~~~~

今天我就以为个人的能力去研究一下，一个重要的进程svchost，本人水平极其有限，而svchost又是微软的一个非常重要的进程，很强大，所以题目就叫svchost进程的浅析，如果

有高手，可以后面跟贴补充~~~不甚感激！

科普一下：
写过一些病毒和木马的人，也许一定不会对svchost进程感到陌生，中过木马的人也许也看到过这个进程，没错，svchost进程，曾几何时一度风靡病毒界，很多病毒木马制作者一

定会用到它，由于杀毒软件的发展，一些杀毒软件已经严密监控住了这个进程（只是部分），所以svchost进程也一时受到冷默，大部分人转向了rootkit技术，可是现在rootkit技

术也成为各大杀软的头号目标，什么高启发式，云查杀，主动防御等技术都会对一般的rootkit绝不留情的，呵呵~~不过，道高一尺，魔高一丈，聪明的病毒和木马制作者总是会想

出一些更高操作的技术.....

svchost.exe是NT核心系统的一个重要的进程，通过ctrl+alt+del打开任务管理器，就会在进程列表中看到进程中会出现几个svchost，这些svchost提供很多系统服务，如:rpcss服

务,dmserver服务,dhcp服务等，要想知道svchost进程，提供了哪些服务，很简单，(xp系统)在cmd中输入tasklist /svc即可，我们就会看到一大堆我们熟悉的服务:
RpcSs,Dhcp,Netman,srservice,swcsvc,winmgmt.....

下面借用网上一段话来详细讲解一下svchost进程!

Windows系统进程分为独立进程和共享进程两种，“Svchost.exe”文件存在于“%systemroot% system32”目录下，它属于共享进程。随着Windows系统服务不断增多，为了节省系统资源，微软把很多服务做成共享方式，交由 Svchost.exe进程来启动。

但Svchost进程只作为服务宿主，并不能实现任何服务功能，即它只能提供条件让其他服务在这里被启动，而它自己却不能给用户提供任何服务。那这些服务是如何实现的呢?

原来这些系统服务是以动态链接库(dll)形式实现的，它们把可执行程序指向 Svchost，由Svchost调用相应服务的动态链接库来启动服务。那Svchost又怎么知道某个系统服务该调用哪个动态链接库呢?这是通过系统服务在注册表中设置的参数来实现。

svchost.exe是一个属于微软Windows操作系统的系统程序，微软官方对它的解释是：Svchost.exe 是从动态链接库 (DLL) 中运行的服务的通用主机进程名称。这个程序对系统的正常运行是非常重要,而且是不能被结束的。

当系统启动时，Svchost将检查注册表中的服务部分，以构建需要加载的服务列表。Svchost的多个实例可以同时运行。每个Svchost会话可以包含一组服务，以便根据Svchost的启动方式和位置的不同运行不同的服务，这样可以更好地进行控制且更加便于调试。

Svchost组是由注册表[HKEY_LOCAL_MACHINE\ Software\Microsoft\Windows NT\CurrentVersion\Svchost]项来识别的。在这个注册表项下的每个值都代表单独的Svchost组，并在我们查看活动进程时作为单独的实例显示。这里的键值均为REG_MULTI_SZ类型的值，并且包含该Svchost组里运行的服务名称

总结一点：服务是靠Svchost来启动的（并不是所有的）

举例说明：

以windows xp为例，点击“开始”/“运行”，输入“services.msc”命令，弹出服务对话框，然后打开“remote procedure call”属性对话框，可以看到rpcss服务的可执行文件的路径为“c:\windows\system32\svchost -k rpcss”，这说明rpcss服务是依靠svchost调用“rpcss”参数来实现的，而参数的内容则是存放在系统注册表中的。

在运行对话框中输入“regedit.exe”后回车，打开注册表编辑器，找到[hkey_local_machine systemcurrentcontrolsetservicesrpcss]项，找到类型为“reg_expand_sz”的键“magepath”，其键值为“%systemroot%system32svchost -k rpcss”（这就是在服务窗口中看到的服务启动命令），另外在“parameters”子项中有个名为“servicedll”的键，其值为“% systemroot%system32rpcss.dll”，其中“rpcss.dll”就是rpcss服务要使用的动态链接库文件。这样 svchost进程通过读取“rpcss”服务注册表信息，就能启动该服务了。

上面介绍了svchost的基本原理，下面我通过分析svchost主要代码来说明svchost，代码参考开源操作系统ReactOS~~

还是先按调试的顺序，先从函数入口看起吧，哈哈~~~

代码如下：

```
int _tmain (int argc, LPTSTR argv [])
{
  DWORD NrOfServices;
  LPSERVICE_TABLE_ENTRY ServiceTable;

  if (argc < 3)
  {
    //当输入参数少于3个时，可以做一些你想要的事，这里直接返回
    return 0;
  }

  if (_tcscmp(argv[1], _T("-k")) != 0)  //检查第两个参数是不是-k
  {
    //如果不是也直接返回
    return 0;
  }

  NrOfServices = LoadServiceCategory(argv[2]);  //如果参数正确，则检查注册表中的服务部分，以构建需要加载的服务列表，并在其中加载所有服务

  DPRINT1("NrOfServices: %lu\n", NrOfServices);
  if (0 == NrOfServices)
    return 0;

  ServiceTable =(LPSERVICE_TABLE_ENTRY)HeapAlloc(GetProcessHeap(), 0, sizeof(SERVICE_TABLE_ENTRY) * (NrOfServices + 1));   //分配存放服务列表的堆栈，构建服务列表

  if (NULL != ServiceTable)
  {
    DWORD i;
    PSERVICE Service = FirstService;

    //填充服务列表，使用了数据结构中的链表进行操作
    for (i = 0; i < NrOfServices; ++i)
    {
      DPRINT1("Loading service: %s\n", Service->Name);
      ServiceTable[i].lpServiceName = Service->Name;
      ServiceTable[i].lpServiceProc = Service->ServiceMainFunc;
      Service = Service->Next;
    }

    //用一个空的服务，做为服务列表的结束
    ServiceTable[i].lpServiceName = NULL;
    ServiceTable[i].lpServiceProc = NULL;

    if (FALSE == StartServiceCtrlDispatcher(ServiceTable))      //启动服务的分发函数
      printf("Failed to start service control dispatcher, ErrorCode: %lu\n", GetLastError());
```

```
    HeapFree(GetProcessHeap(), 0, ServiceTable);        //释放分配的服务列表堆栈
  }
  else
  {
    DPRINT1("Not enough memory for the service table, trying to allocate %u bytes\n", sizeof(SERVICE_TABLE_ENTRY) * (NrOfServices + 1));
  }

  DPRINT1("Freeing services...\n");
  FreeServices();      //做最后的清理工作

    return 0;
}
```

代码很简单，主要用到了StartServiceCtrlDispatcher启动服务，其实这就是编写svchost启动的服务和编写一般服务的区别，一般的服务会写成一个EXE的文件，然后自己编写一个安装启动服务的程序，但是用svchost，其实svchost就帮我们做了这些工作，我们只需要在某个注册表下写入svchost的值，svchost会在像上面操作的一样，在系统启动时，通过svchost的参数，来检查注册表，然后建立一个服务列表，再通过遍历服务列表，一项一项的加启它们就OK了！

看了上面的一段话，我想大家都已经很清楚了svchost进程，下面我在进一步讲解一下~~上面用到了一个自定义函数LoadServiceCategory来检查注册表，我们来看看它做了哪些工作吧~~

```
DWORD LoadServiceCategory(LPCTSTR ServiceCategory)
{
  HKEY hServicesKey;
  DWORD KeyType;
  DWORD BufferSize = REG_MAX_DATA_SIZE;
  TCHAR Buffer[REG_MAX_DATA_SIZE];
  LPCTSTR ServiceName;
  DWORD BufferIndex = 0;
  DWORD NrOfServices = 0;

  //得到所有的服务
  //static LPCTSTR SVCHOST_REG_KEY = _T("SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\SvcHost")
  if (ERROR_SUCCESS != RegOpenKeyEx(HKEY_LOCAL_MACHINE, SVCHOST_REG_KEY, 0, KEY_READ, &hServicesKey))
  {
    DPRINT1("Could not open service category: %s\n", ServiceCategory);
    return 0;
  }

  if (ERROR_SUCCESS != RegQueryValueEx(hServicesKey, ServiceCategory, NULL, &KeyType, (LPBYTE)Buffer, &BufferSize))
  {
    DPRINT1("Could not open service category (2): %s\n", ServiceCategory);
    RegCloseKey(hServicesKey);
    return 0;
  }

  //对应上面的RegOpenKeyEx
  RegCloseKey(hServicesKey);

  //加载所有的服务
  ServiceName = Buffer;
  while (_T('\0') != ServiceName[0])
  {
    size_t Length;

    Length = _tcslen(ServiceName);
    if (0 == Length)
      break;

    if (TRUE == PrepareService(ServiceName))     //写入到注册表的相应位置，加载服务
      ++NrOfServices;

    BufferIndex += (Length + 1) * sizeof(TCHAR);

    ServiceName = &Buffer[BufferIndex];
  }

  return NrOfServices;
}
```

通过上面的这个函数，将SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\SvcHost键下的所有服务名，写入到注册表相应位置，并加载服务，这个函数很简单只进行了注册表的查寻，然后将得到的键值存入一个数组中，在从数组中取出相应值进行操作，逐个操作，并调用了一个自定义函数PrepareService~~再来分析一下PrepareService这个函数在为我们做哪些工作~~~

```
BOOL PrepareService(LPCTSTR ServiceName)
{
  HKEY hServiceKey;
  TCHAR ServiceKeyBuffer[MAX_PATH + 1];
  DWORD LeftOfBuffer = sizeof(ServiceKeyBuffer) / sizeof(ServiceKeyBuffer[0]);
  DWORD KeyType;
  PTSTR Buffer = NULL;
  DWORD BufferSize = MAX_PATH + 1;
  LONG RetVal;
```

```
  HINSTANCE hServiceDll;
  TCHAR DllPath[MAX_PATH + 2]; /* See MSDN on ExpandEnvironmentStrings() for ANSI strings for more details on + 2 */
  LPSERVICE_MAIN_FUNCTION ServiceMainFunc;
  PSERVICE Service;

  //构建服务在注册表中的一些参数
  //static LPCTSTR SERVICE_KEY = _T("SYSTEM\\CurrentControlSet\\Services\\")
  //static LPCTSTR PARAMETERS_KEY  = _T("\\Parameters");
  _tcsncpy(ServiceKeyBuffer, SERVICE_KEY, LeftOfBuffer);
  LeftOfBuffer -= _tcslen(SERVICE_KEY);
  _tcsncat(ServiceKeyBuffer, ServiceName, LeftOfBuffer);
  LeftOfBuffer -= _tcslen(ServiceName);
  _tcsncat(ServiceKeyBuffer, PARAMETERS_KEY, LeftOfBuffer);
  LeftOfBuffer -= _tcslen(PARAMETERS_KEY);

  if (LeftOfBuffer < 0)
  {
    DPRINT1("Buffer overflow for service name: '%s'\n", ServiceName);
    return FALSE;
  }

  //打开相应注册表项，查找要加载的DLL名
  if (ERROR_SUCCESS != RegOpenKeyEx(HKEY_LOCAL_MACHINE, ServiceKeyBuffer, 0, KEY_READ, &hServiceKey))
  {
    DPRINT1("Could not open service key (%s)\n", ServiceKeyBuffer);
    return FALSE;
  }

  do
  {
    if (Buffer)
      HeapFree(GetProcessHeap(), 0, Buffer);

    Buffer = (PTSTR)HeapAlloc(GetProcessHeap(), 0, BufferSize);
    if (NULL == Buffer)
    {
      DPRINT1("Not enough memory for service: %s\n", ServiceName);
      return FALSE;
    }

    RetVal = RegQueryValueEx(hServiceKey, _T("ServiceDll"), NULL, &KeyType, (LPBYTE)Buffer, &BufferSize);

  } while (ERROR_MORE_DATA == RetVal);


  RegCloseKey(hServiceKey);

  if (ERROR_SUCCESS != RetVal || 0 == BufferSize)
  {
    DPRINT1("Could not read 'ServiceDll' value from service: %s, ErrorCode: 0x%x\n", ServiceName, RetVal);
    HeapFree(GetProcessHeap(), 0, Buffer);
    return FALSE;
  }

  //传换成系统中的DLL的目录
  BufferSize = ExpandEnvironmentStrings(Buffer, DllPath, sizeof(DllPath));
  if (0 == BufferSize)
  {
    DPRINT1("Invalid ServiceDll path: %s\n", Buffer);
    HeapFree(GetProcessHeap(), 0, Buffer);
    return FALSE;
  }

  HeapFree(GetProcessHeap(), 0, Buffer);


  DPRINT1("Trying to load dll\n");
  hServiceDll = LoadLibrary(DllPath);  //加载服务的DLL

  if (NULL == hServiceDll)
  {
    DPRINT1("Unable to load ServiceDll: %s, ErrorCode: %u\n", DllPath, GetLastError());
    return FALSE;
  }

  ServiceMainFunc = (LPSERVICE_MAIN_FUNCTION)GetProcAddress(hServiceDll, "ServiceMain");  //查找DLL中存在的ServiceMain函数，现在知道为什么每个DLL服务中
必须包括一个ServiceMain函数了吧

  //分配一个服务列表中的一个服务
  Service =(PSERVICE)HeapAlloc(GetProcessHeap(), 0, sizeof(SERVICE));
  if (NULL == Service)
  {
    DPRINT1("Not enough memory for service: %s\n", ServiceName);
    return FALSE;
  }
```

```
  memset(Service, 0, sizeof(SERVICE));
  Service->Name =(PTSTR)HeapAlloc(GetProcessHeap(), 0, _tcslen(ServiceName) + sizeof(TCHAR));
  if (NULL == Service->Name)
  {
    DPRINT1("Not enough memory for service: %s\n", ServiceName);
    HeapFree(GetProcessHeap(), 0, Service);
    return FALSE;
  }

  //填充服务列表中的服务
  //服务列表的结构如下：
  /*
  typedef struct _SERVICE {
        PTSTR    Name;  //服务名
    HINSTANCE  hServiceDll;  //此服务所使用的DLL
        LPSERVICE_MAIN_FUNCTION    ServiceMainFunc;  //DLL中的ServiceMain函数
    struct _SERVICE  *Next;  //指向下一个服务
  } SERVICE, *PSERVICE;
  */
  _tcscpy(Service->Name, ServiceName);
  Service->hServiceDll = hServiceDll;
  Service->ServiceMainFunc = ServiceMainFunc;

  Service->Next = FirstService;
  FirstService = Service;

  return TRUE;
}
```

在上面的操作中我们为服务列表中的服务分配了堆栈，最后一定要记得释放~~于是在main函数的最后调用了一个自定义的函数FreeServices用来释放前面分配的堆栈空间，代码如

下：

```
VOID FreeServices(VOID)
{
  while (FirstService)
  {
    PSERVICE Service = FirstService;
    FirstService = Service->Next;

    FreeLibrary(Service->hServiceDll);  //释放前面加载的服务的DLL

    HeapFree(GetProcessHeap(), 0, Service->Name);  释放堆栈
    HeapFree(GetProcessHeap(), 0, Service);
  }
}
```

如果你能看到这里，基本上svchost的进程的原理你已经掌握了，如果你写过用svchost启动的服务，我想你一定知道为什么要那么写了~~~其实每个事物的存在都会有它的原因，我们只要肯花时间去研究，就没有什么不懂的！（至于，怎么样编写基于svchost的服务，我就不说了，大家自己google吧，以前有位很牛的前辈用Win32汇编写过一篇很详细的文章）

上面我们详细的分析了一下开源的操作系统ReactOS的Svchost的源代码，其实Windows的Svchost的实现方式和这是差不多的，只是其中还应用了一些用于处理多线程同步的代码，有人也许会问，windows不开布源代码，为何ReactOS组织的人会写出如此相近的源代码，其实，以我个人的想法，无非就是他们经过反汇编windows的svchost进程，而得到的，因为windows中的所有的DLL,EXE,SYS等等之类的东东，微软并没有加壳保护，所以只要反汇编牛一点的人，我想写出类似的代码也不是问题的！！呵呵，只是我的猜想~~有兴趣的人，可以去反汇编看看Windnows中的svchost进程（可以用IDA反汇编看看，还可以用OD跟踪调试，只是记住带参数，不然就直接返回了,我是直接把那句判断跳过，只想看看他的实现原理就可以）,你会发现，原来就是这样的！

好了，就写这么多吧，写多了，大家看着也累，我想svchost的基本原理和方法，大家一定都有了一个大概的了解，本人水平有限，听说老毛的《windows内核情景分析》中最后有一章讲到过svchost进程，我没看过，只看过目录上有，也不知道他老人家是怎么讲的，肯定没他老人家那等功夫，如果还看不懂，可以去看看他老人家的书，就是有点贵。~~不过说不定，一狠心我就买了，我喜欢拿着书看，并把它看成黑黑的，然后整理成薄薄的笔记，也许就是别人说的，把一本很厚的书看薄了，你就学会了~~再见，我会用一整年的时间（只是有个大概认识，呵呵，真正在理解可能要等以后工作中慢慢体会才能得到windows和linux的精华所在），好好研究一下windows的内核，因为打算以后就从事系统底层驱动开发这方面的工作了，也许过几年还会去研究linux内核，不过都是后话了，在这期间，如果偶有心得体会，我会及时将自己的一些学习休会与心得发到论坛上，希望能给更多喜欢研究内核的朋友们一点帮助，先把windows这个难缠的家伙搞定，学习windows内核不像linux内核（相对难一点，个人观点），linux内核源码网上有很多，但windows有时要反汇编才能明白其中的原理和精华~~其实我们只有掌握了一种学习能力，在加上坚持不懈的努力，最后都可以变成“牛”，学习能力的培养就在于你平时的日积月累了，呵呵，这个谁都帮不了你的！！(最后补充非常重要的一点：小弟前面的一些言论，可以不完全正确，如果实在看不下去的，可以直接跳过，我不想看到因为我的一些胡言乱语，又引起后面一些口水战之类的，再次重申：我的言论不针对任何个人，任何组织和公司，因为我没时间和你口水战，我想大家有时间，还是多花在时间在自己喜欢的值得为之努力奋斗的事情上)，但是对于文中的错误和对此有更深层的理解，欢迎指点，后面跟贴，以免我这小菜误导很多看到我这篇文章的人！。

最后，把我在vc6.0下编译通过的svchost源码附加上去，用vs2008同样编译通过！

上传的附件

	svchost.rar (1.15 MB, 431 次下载)	[谁下载?]

此帖于 2011-01-08 19:16:59 被 熊猫正正 最后编辑 原因: 上传附件