**C2-level Security**

C2安全等级的重要要求列表，由美国防御部提供：

可以对个人或者组对一个资源的访问进行控制，例如允许访问或者被拒绝。

当一个进程释放掉一段内存后，这段内存的内容不能够再被访问，相似的，文件系统也必须保证，文件在删除后，不能被读取。

当用户登陆的时候，必须能够使用一种唯一的方式，进行认证。用户的所有活动，必须得到确认。

系统管理员必须能够审查安全相关事件。对于安全相关事件审查数据的访问权限，只能授予系统管理员。

系统必须有能够防止外来的恶意攻击，例如在系统在运行中被修改，或者文件被修改。

**访问控制模块**

访问控制模块能够控制进程对安全特性对象的访问，或者进程执行的系统管理任务。

**内核对象的命名空间：**

一个远程桌面的服务可以为自己的命名内核对象设置多个命名空间，这些内核对象包括：event，semaphores，mutexes，waitable timers，file-mapping，job object。在client/server类型的应用程序中，服务都有一个全局命名空间global namespace。除此之外，每个客户会话都可以有自己单独的命名空间，来放置自己独有的内核对象。

独立的客户会话命名空间能够让多个客户运行同一个应用程序而不互相干扰。在一个会话中使用使用会话名称作为内核对象的缺省命名空间，另外，进程还可以使用全局命名空间，使用的方法是在内核对象的名字前，加上“Global\”前缀。例如，下面的代码调用CreateEven在全局命名空间中创建一个CSAPP命名对象。

CreateEvent(NULL,FALSE,FALSE,”Global\\CSAPP”);

远程桌面环境中的服务应用程序将全局命名空间Global namespace作为缺省命名空间。

会话0（session 0）只被主机服务使用，他没有控制台会话。

全局命名空间能够让多个客户会话和服务应用程序进行交互。例如，一个client/server应用程序可能会使用一个互斥对象mutex object来同步自己的线程。服务组件能够在全局命名空间中创建互斥对象。一个客户会话能够使用“Global\”前缀，来打开互斥对象。

全局命名空间global namespace还有一个用处，就是应用程序使用命名对象，来检查是否已经有一个应用程序的实例已经在系统中的某个会话下运行。这个命名对象必须在全局命名空间中创建，而不能在某个会话的命名空间中创建。更普遍的是检查某个会话下，是否有多个程序的实例在运行。

如果要在本会话下创建一个命名对象，也就是在会话的命名空间中，创建内核对象，那么必须在内核对象前加上”Local\”前缀！注意这个关键字是大小写敏感的。

CreateMutex(NULL,FALSE,L”Local\\ydm”);

“Session\”前缀被系统保留，不要使用这个前缀来创建命名内核对象。

windows快速用户切换，是通过远程桌面服务会话来实现的。第一个用户登陆后，其会话为会话1（Session 1），第二个用户的会话为会话2，以此类推。内核对象的名字必须遵守远程桌面服务的准则，否则应用程序不能支持多用户。

创建全局的file-mapping命名对象，有一些其他的要求。如果在非session 0的会话中，创建file-mapping命名对象的话，必须要有一定的特权。因此，一个运行在任意远程桌面上的应用程序必须有SeCreateGlobalPrivilege权限，才能创建全局的file-mapping对象，所幸的是，只有file-mapping命名对象有这个要求。file-mapping命名对象的打开，没有这个要求。

**HANDLE WINAPI CreateBoundaryDescriptor(**

**\_\_in LPCTSTR Name,**

**\_\_in ULONG Flags**

**);**

创建一个边界描述符：

Name：边界描述符的名字，这个名字由你自己来取；

Flags：保留，=NULL。

返回值：

如果函数成功，那么返回值就是一个边界描述符的句柄。

如果函数失败，返回值NULL。

备注：

一个新的边界描述符至少要有一个安全描述符SID。可以使用AddSIDToBoundaryDescriptor函数，为边界描述符增加SID。

**BOOL WINAPI AddSIDToBoundaryDescriptor(**

**\_\_inout HANDLE \*BoundaryDescriptor,**

**\_\_in PSID RequiredSid**

**);**

将一个安全描述符SID添加到指定的边界描述符中

BoundaryDescriptor：一个指向边界描述符的句柄。CreateBoundaryDescriptor函数返回这个句柄。

RequiredSid：一个指向安全描述符Sid的指针。

返回值：

如果函数成功，返回值是非零。

如果函数失败，返回零。

备注：

AddSIDToBoundaryDescriptor函数必须为每个SID调用一次，将其增加到边界描述符中。

**BOOL WINAPI CreateWellKnownSid(**

**\_\_in WELL\_KNOWN\_SID\_TYPE WellKnownSidType,**

**\_\_in\_opt PSID DomainSid,**

**\_\_out\_opt PSID pSid,**

**\_\_inout DWORD \*cbSid**

**);**

创建一个有预定义别名的SID

WellKnownSidType：WELL\_KNOWN\_SID\_TYPE列举的成员，用这个成员标识要获取的SID。

DomainSid：一个指向SID的指针，这个SID代表了创建SID所使用的域控制安全描述符，如果是本地计算机，那么使用NULL。

pSid：一个指向内存块的指针，CreateWellKnownSid函数会将创建好的新SID保存到这个内存块中。

cbSid：一个DWORD类型的指针。代表了生成的pSid的大小。

返回值：

如果函数成功，返回非零，否则返回零。

**HANDLE WINAPI CreatePrivateNamespace(**

**\_\_in\_opt LPSECURITY\_ATTRIBUTES lpPrivateNamespaceAttributes,**

**\_\_in LPVOID lpBoundaryDescriptor,**

**\_\_in LPCTSTR lpAliasPrefix**

**);**

创建一个私有命名空间

lpPrivateNamespaceAttributes：一个指向SECURITY\_ATTRIBUTES结构的指针，这个结构定义了命名空间的安全属性。

lpBoundaryDescriptor：一个描述符，定义了命名空间如何被隔离。调用者必须在这个边界内。函数CreateBoundaryDescriptor函数创建一个边界描述符。

lpAliasPrefix：命名空间的前缀。为了在这个命名空间中创建命名对象，定义对象名称的前缀，形式为”pre fix\”对象名称。系统支持不同边界中，可以有相同的私有命名空间。

返回值：

如果函数成功，返回一个指向新命名空间的句柄，否则返回NULL。

备注：

其他的应用程序，可以使用OpenPrivateNamespac函数，来打开一个已经存在的命名空间。创建命名控件的程序，使用ClosePrivateNamespace函数来关闭指向命名控件的句柄。当创建这个命名空间的进程退出时，被创建的命名空间也会被关闭，那么此后在使用OpenPrivateNamespace函数，就会失败，但这并不影响对该命名控件中的对象的操作。

**HANDLE WINAPI OpenPrivateNamespace(**

**\_\_in LPVOID lpBoundaryDescriptor,**

**\_\_in LPCTSTR lpAliasPrefix**

**);**

打开一个私有命名空间。

lpBoundaryDescriptor：定义私有命名空间如何被隔离的描述符。调用这个函数的程序必须在这个边界之内。函数CreateBoundaryDescriptor创建一个边界描述符。

lpAliasPrefix：命名空间前缀。为了在这个私有命名空间中创建一个命名对象，那么这个对象名称的前缀必须是lpAliasPrefix\。系统支持多个私有命名空间具有相同的别名，只要边界描述符不同就可以了。

返回值：

函数返回一个已经存在的私有命名空间的句柄。

1. 创建一个边界标识符BoundaryDescriptor；

**CreateBoundaryDescriptor;**

1. 获取一个SID；

**CreateWellKnownSid**

1. 将这个SID放到边界标识符中，只有具有这个SID的程序，才能打开这个边界标识符；

**AddSidToBoundaryDescriptor**

**//**

**你可以将这个函数理解为，为边界描述符创建边界！！！**

**//**

1. 创建一个安全描述符（允许访问安全描述符）；

**ConvertStringSecurityDescriptorToSecurityDescriptor**

1. 用这个边界描述符和安全描述符，来创建一个私有命名空间，同时给这个私有命名空间一个别名，例如:”ydm”;

**CreatePrivateNamespace(Boundary\_Handle，&Name\_Handle,L”ydm”);**

**OpenPrivateNamespace;**

1. 要在这个私有命名空间中，创建一个内核对象，例如一个Mutex，那么如下就可以了：

**CreatetMutex(NULL,FALSE,L”ydm\\object\_name”);**