**完整性机制：**

完整性机制，是对windows安全授权的一个补充。

BOOL GetProcessIntegrityLevel(HANDLE hProcess, PDWORD pIntegrityLevel,

PDWORD pPolicy, PDWORD pResourceIntegrityLevel, PDWORD pResourcePolicy) {

HANDLE hToken = NULL;

if (!OpenProcessToken(hProcess, TOKEN\_READ, &hToken)) {

return(FALSE);

}

BOOL bReturn = FALSE;

// First, compute the size of the buffer to get the Integrity level

DWORD dwNeededSize = 0;

if (!GetTokenInformation(

hToken, TokenIntegrityLevel, NULL, 0, &dwNeededSize)) {

PTOKEN\_MANDATORY\_LABEL pTokenInfo = NULL;

if (GetLastError() == ERROR\_INSUFFICIENT\_BUFFER) {

// Second, allocate a memory block with the the required size

pTokenInfo = (PTOKEN\_MANDATORY\_LABEL)LocalAlloc(0, dwNeededSize);

if (pTokenInfo != NULL) {

// And finally, ask for the integrity level

if (GetTokenInformation(hToken, TokenIntegrityLevel, pTokenInfo,

dwNeededSize, &dwNeededSize)) {

\*pIntegrityLevel =

\*GetSidSubAuthority(

pTokenInfo->Label.Sid,

(\*GetSidSubAuthorityCount(pTokenInfo->Label.Sid)-1)

);

bReturn = TRUE;

}

// Don't forget to free the memory

LocalFree(pTokenInfo);

}

}

}

// Try to get the policy if the integrity level was available

if (bReturn) {

\*pPolicy = TOKEN\_MANDATORY\_POLICY\_OFF;

dwNeededSize = sizeof(DWORD);

GetTokenInformation(hToken, TokenMandatoryPolicy, pPolicy,

dwNeededSize, &dwNeededSize);

}

// Look for the resource policy

\*pResourceIntegrityLevel = 0; // 0 means none explicitely set

\*pResourcePolicy = 0;

PACL pSACL = NULL;

PSECURITY\_DESCRIPTOR pSD = NULL;

DWORD dwResult = ERROR\_SUCCESS;

// Look for the no-read-up/no-write-up policy in the SACL

if (hToken != NULL) {

dwResult =

GetSecurityInfo(

hProcess, SE\_KERNEL\_OBJECT,

LABEL\_SECURITY\_INFORMATION,

NULL, NULL, NULL,

&pSACL, &pSD

);

if (dwResult == ERROR\_SUCCESS) {

if (pSACL != NULL) {

SYSTEM\_MANDATORY\_LABEL\_ACE\* pACE = NULL;

if ((pSACL->AceCount > 0) && (GetAce(pSACL, 0, (PVOID\*)&pACE))) {

if (pACE != NULL) {

SID\* pSID = (SID\*)(&pACE->SidStart);

\*pResourceIntegrityLevel = pSID->SubAuthority[0];

\*pResourcePolicy = pACE->Mask;

}

}

}

}

// Cleanup memory allocated on our behalf

if (pSD != NULL) LocalFree(pSD);

}

// Don't forget to close the token handle.

CloseHandle(hToken);

return(bReturn);

}

|  |  |  |
| --- | --- | --- |
| **Value** | **Description** | **Symbol** |
| 0x0000 | Untrusted level | SECURITY\_MANDATORY\_UNTRUSTED\_RID |
| 0x1000 | Low integrity level | SECURITY\_MANDATORY\_LOW\_RID |
| 0x2000 | Medium integrity level | SECURITY\_MANDATORY\_MEDIUM\_RID |
| 0x3000 | High integrity level | SECURITY\_MANDATORY\_HIGH\_RID |
| 0x4000 | System integrity level | SECURITY\_MANDATORY\_SYSTEM\_RID |

|  |  |
| --- | --- |
| **Value** | **Meaning** |
| TOKEN\_MANDATORY\_POLICY\_OFF  0x0 | No mandatory integrity policy is enforced for the token. |
| TOKEN\_MANDATORY\_POLICY\_NO\_WRITE\_UP  0x1 | A process associated with the token cannot write to objects that have a greater mandatory integrity level. |
| TOKEN\_MANDATORY\_POLICY\_NEW\_PROCESS\_MIN  0x2 | A process created with the token has an integrity level that is the lesser of the parent-process integrity level and the executable-file integrity level. |
| TOKEN\_MANDATORY\_POLICY\_VALID\_MASK  0x3 | A combination of TOKEN\_MANDATORY\_POLICY\_NO\_WRITE\_UP and TOKEN\_MANDATORY\_POLICY\_NEW\_PROCESS\_MIN. |

**DLL简介以及DLL文件如何装入进程地址空间。**

**DLL-动态链接库。LIB-静态链接库。**

**LoadLibrary**