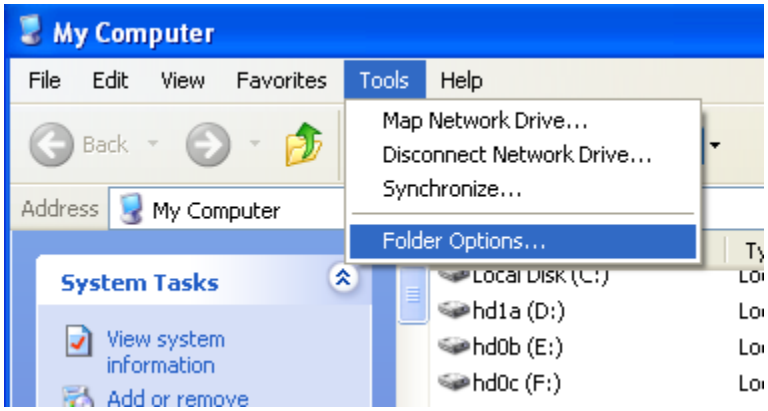
Creating a DACL from a scratch program example

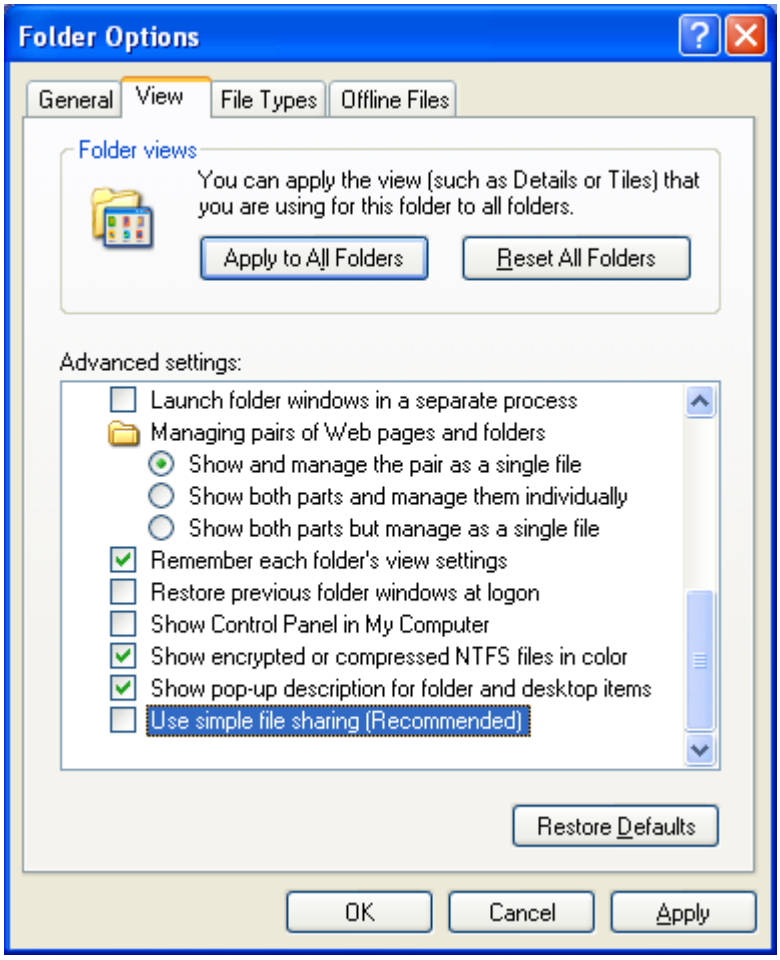
Before you begin, some notes for Windows XP:

For the default setting of the Win XP installation, to view the Security tab of the Windows object you have to disable the "Use simple file sharing" in the folder options setting as shown in the following steps.

Launch the Windows Explorer ⟶ Click the Tools menu ⟶ Select the Folder Options…



Then, click the View tab and uncheck the "Use simple file sharing (Recommended)" check box. Click the OK button.



Creating a DACL From a Scratch Example

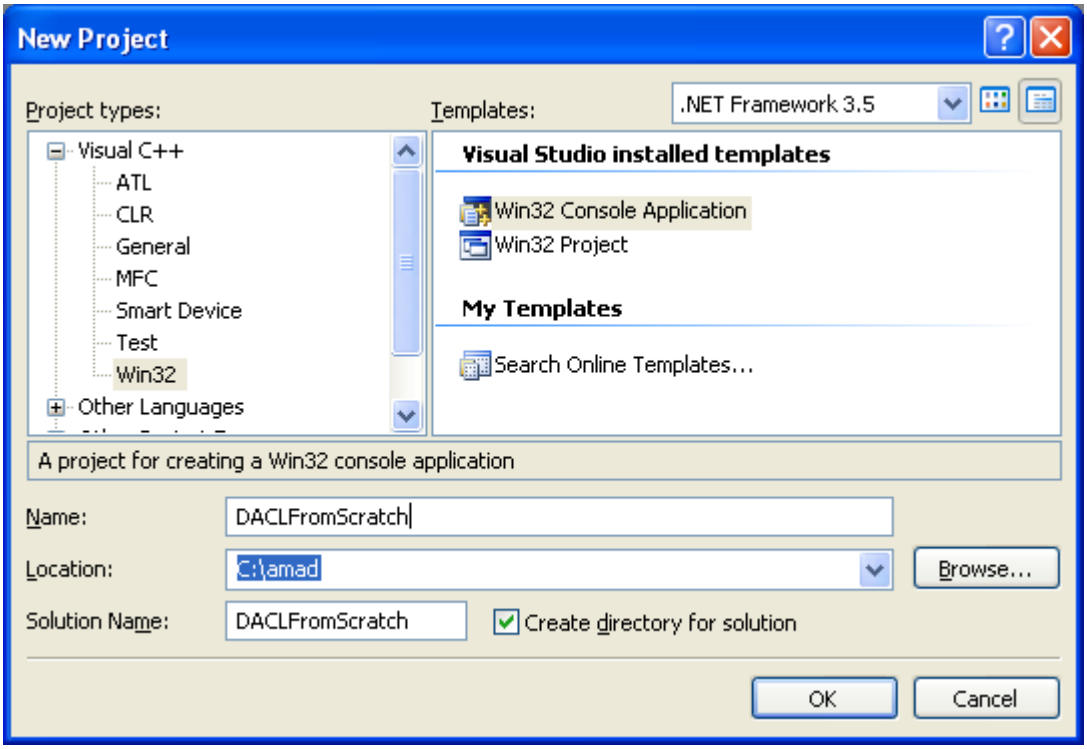
The following example shows how to properly construct a DACL during the Windows's object creation (Windows directory). The example contains a function, CreateMyDACL(), that uses the security descriptor definition language (sddl) to define the granted and denied access control in a DACL and then set the access control on the newly created directory. To provide different access for your application's objects, modify the CreateMyDACL() function as needed. In the example:

1. The wmain() function passes an address of a SECURITY_ATTRIBUTES structure to the CreateMyDACL() function. The CreateMyDACL() function uses SDDL strings to:

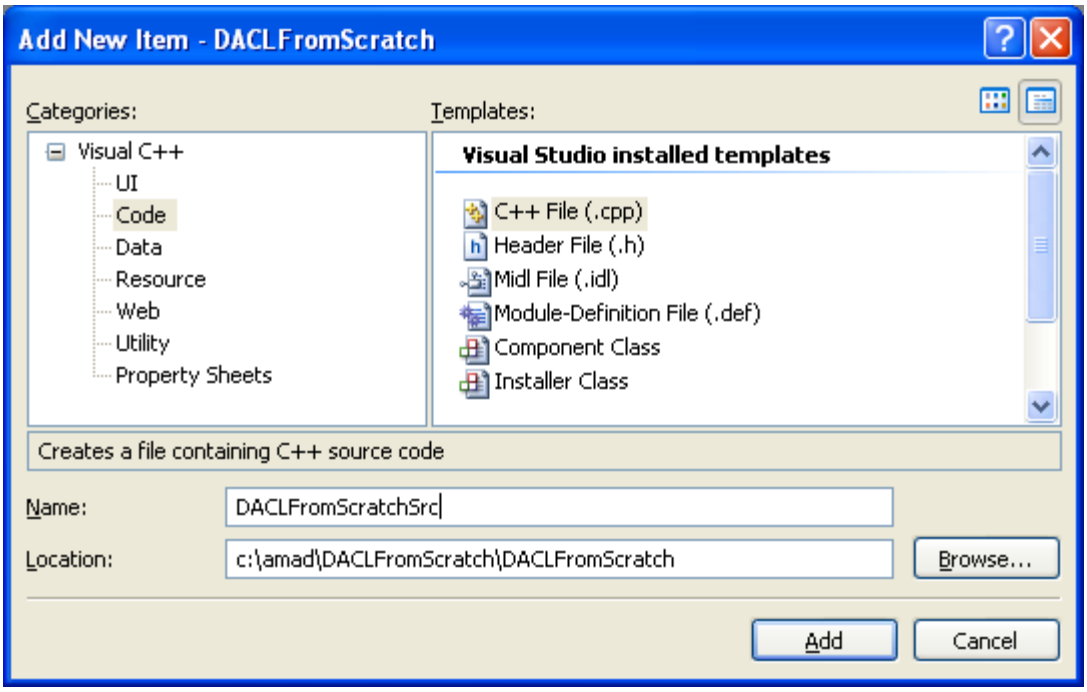
    a. Deny all access to built-in Administrators group.
    b. Allow read/write/execute access to authenticated users.
    c. Allow full control to anonymous.
    d. Allow full control to built-in guest.

2. The CreateMyDACL() function calls the ConvertStringSecurityDescriptorToSecurityDescriptor() function to convert the SDDL strings to a security descriptor. The security descriptor is pointed to by the lpSecurityDescriptor member of the SECURITY_ATTRIBUTES structure. CreateMyDACL() sends the return value from ConvertStringSecurityDescriptorToSecurityDescriptor() to the main function.
3. The wmain() function uses the updated SECURITY_ATTRIBUTES structure to specify the DACL for a new folder that is created by the CreateDirectory() function.
4. When the wmain() function is finished using the SECURITY_ATTRIBUTES structure, the wmain() function frees the memory allocated for the lpSecurityDescriptor member by calling the LocalFree() function.

To successfully compile SDDL functions such as ConvertStringSecurityDescriptorToSecurityDescriptor(), you must define the _WIN32_WINNT constant as 0x0500 or greater (please refer to _WIN32_WINNT constant for more detail). Create a new empty Win32 console application project. Give a suitable project name and change the project location if needed.



Then, add the source file and give it a suitable name.



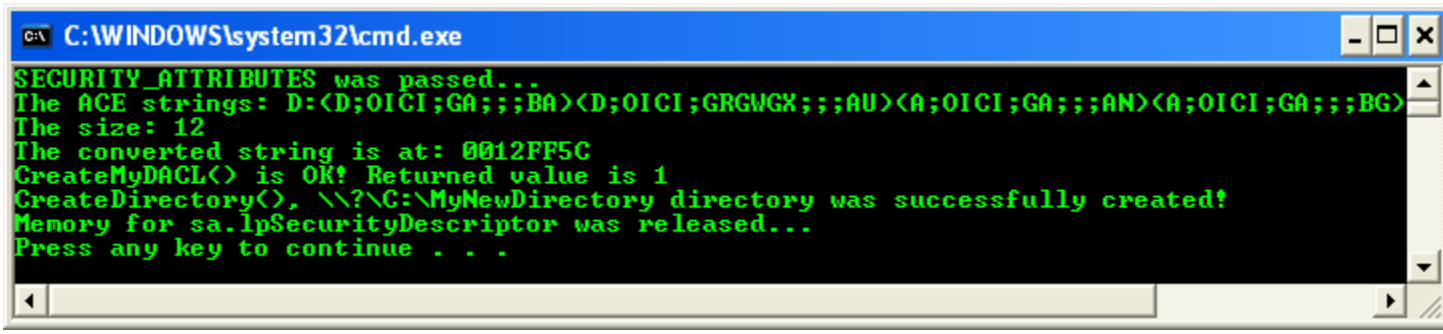
Next, add the following source code.

```
// #define _WIN32_WINNT 0x0500
#include <windows.h>
#include <sddl.h>
#include <stdio.h>

// Prototype
BOOL CreateMyDACL(SECURITY_ATTRIBUTES *);

// A directory, a securable object
WCHAR DirName[] = L"\\\\?\\C:\\MyNewDirectory";

int wmain(int argc, WCHAR **argv)
{
	SECURITY_ATTRIBUTES  sa;
	BOOL RetVal;

	// The SECURITY_ATTRIBUTE structure size
	sa.nLength = sizeof(SECURITY_ATTRIBUTES);
	// The return handle not inherited
	sa.bInheritHandle = FALSE;
	// Call CreateMyDACL() function to set the DACL.
	// The DACL is set in the SECURITY_ATTRIBUTES lpSecurityDescriptor
member.

	RetVal = CreateMyDACL(&sa);

	if(!RetVal)
	{
		// Error encountered; generate message and just exit.
		wprintf(L"Failed CreateMyDACL(), error %d\n", GetLastError());
```

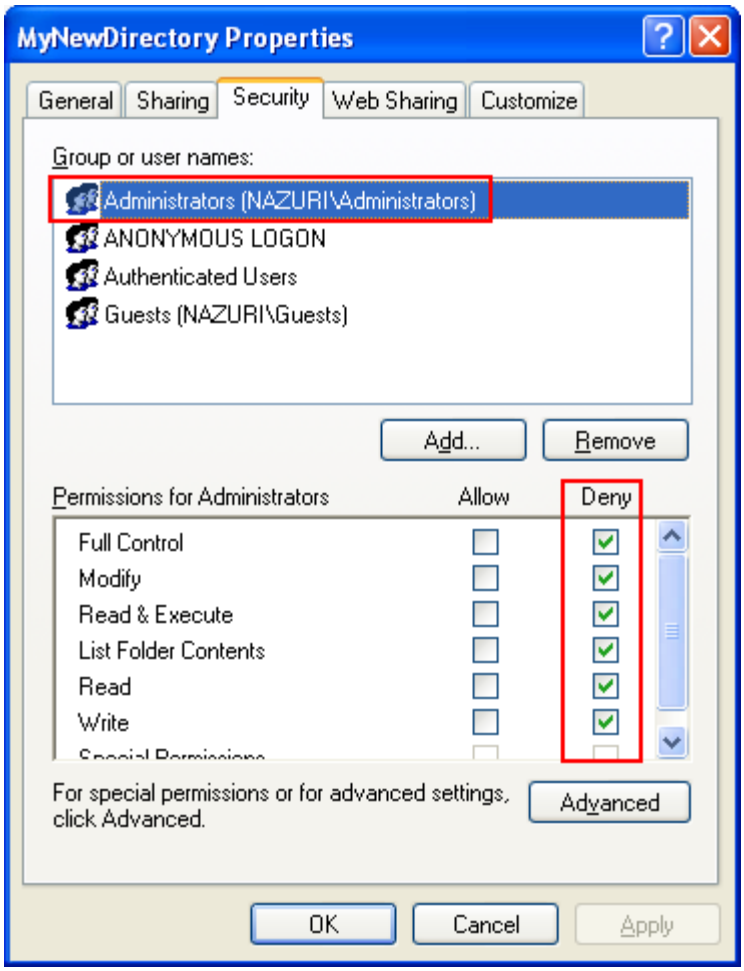
```
            exit(1);
      }
      else
            wprintf(L"CreateMyDACL() is OK! Returned value is %d\n", RetVal);

      // Use the updated SECURITY_ATTRIBUTES to specify security attributes
for securable objects.
      // This example uses security attributes during creation of a new
directory.
      if(CreateDirectory(DirName, &sa) == 0)
      {
            // Error encountered; generate message and exit.
            wprintf(L"CreateDirectory() failed lol! Error %d\n",
GetLastError());
            exit(1);
      }
      else
            wprintf(L"CreateDirectory(), %s directory was successfully
created!\n", DirName);

      // Release the memory allocated for the SECURITY_DESCRIPTOR.
      if(LocalFree(sa.lpSecurityDescriptor) != NULL)
      {
            // Error encountered; generate message and exit.
            wprintf(L"LocalFree() failed, error %d.\n", GetLastError());
            exit(1);
      }
      else
            wprintf(L"Memory for sa.lpSecurityDescriptor was released...\n");
      return 0;
}

// CreateMyDACL()
//    Create a security descriptor that contains the DACL you want.
//    This function uses SDDL to make Deny and Allow ACEs.
//
// Parameter:
//    SECURITY_ATTRIBUTES * pSA
//    Pointer to a SECURITY_ATTRIBUTES structure. It is the caller's
//    responsibility to properly initialize the structure and to free the
structure's
//      lpSecurityDescriptor member when the caller has finished using it.
//    To free the structure's lpSecurityDescriptor member, call the LocalFree
function.
//
// Return value:
//    FALSE if the address to the structure is NULL. Otherwise, this function
returns the value from the
//    ConvertStringSecurityDescriptorToSecurityDescriptor() function.
BOOL CreateMyDACL(SECURITY_ATTRIBUTES * pSA)
{
      PULONG nSize = 0;

      // Define the SDDL for the DACL. This example sets the following
access:
      // Deny all for built-in Administrators group Allow read/write/execute
to Authenticated users
```

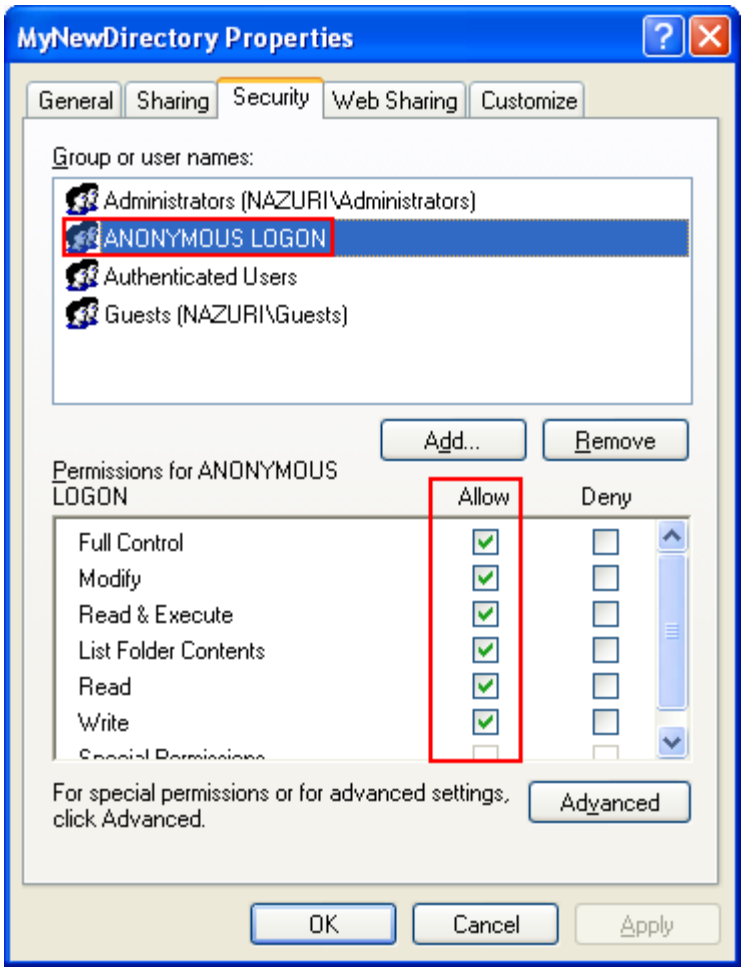
```
	// Allow all to anonymous logon Allow all to built-in guests. This is
not a
	//proper setting, how come you deny the Administrator lol!!!
	// But this example just for fun...
	WCHAR * szSD = L"D:"    // Discretionary ACL
		L"(D;OICI;GA;;;BA)"     // Deny all for built-in Administrators
group :o)
		L"(D;OICI;GRGWGX;;;AU)" // Allow read/write/execute to
Authenticated users
		L"(A;OICI;GA;;;AN)"     // Allow all to anonymous logon
		L"(A;OICI;GA;;;BG)";    // Allow all to built-in guests

	if(pSA == NULL)
		return FALSE;
	else
		wprintf(L"SECURITY_ATTRIBUTES was passed...\n");

	// Do some verifications
	wprintf(L"The ACE strings: %s \n", szSD);
	wprintf(L"The size: %d \n", pSA->nLength);
	wprintf(L"The converted string is at: %p \n", &(pSA-
>lpSecurityDescriptor));

	// Convert the string to the security descriptor binary and return
	return ConvertStringSecurityDescriptorToSecurityDescriptor(
		szSD,                           // The ACE strings
		SDDL_REVISION_1,                // Standard revision level
		&(pSA->lpSecurityDescriptor), // Pointer to the converted
security descriptor
		nSize);                 // The size in byte the converted security
descriptor
}
```

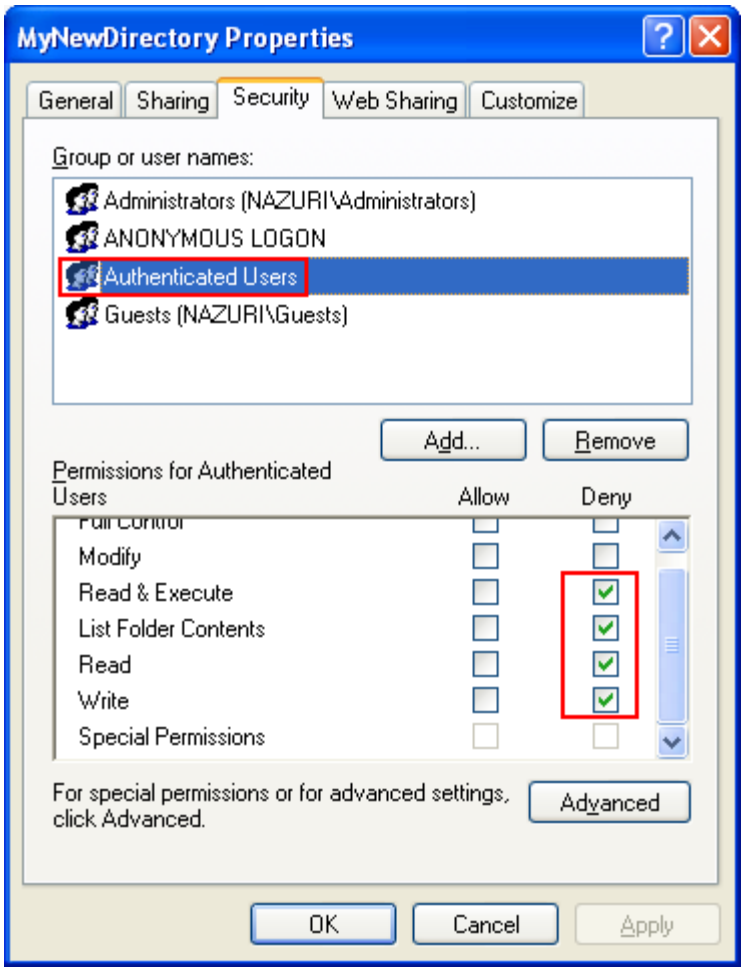
Build and run the project. The following screenshot is a sample output.

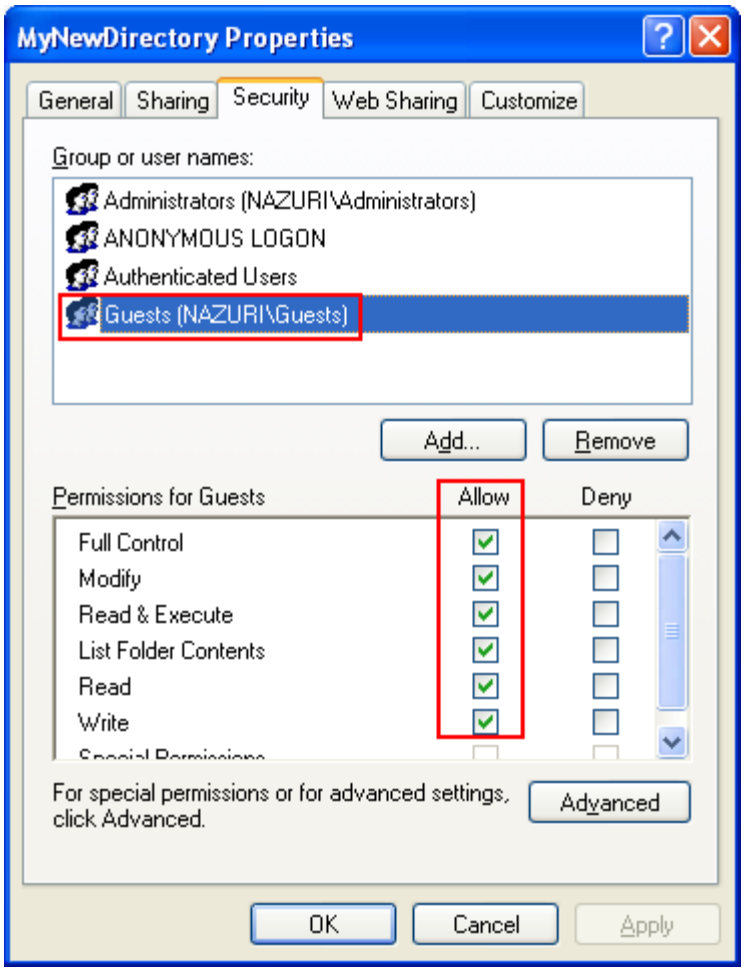


Next, verify through the MyNewDirectory property's pages.

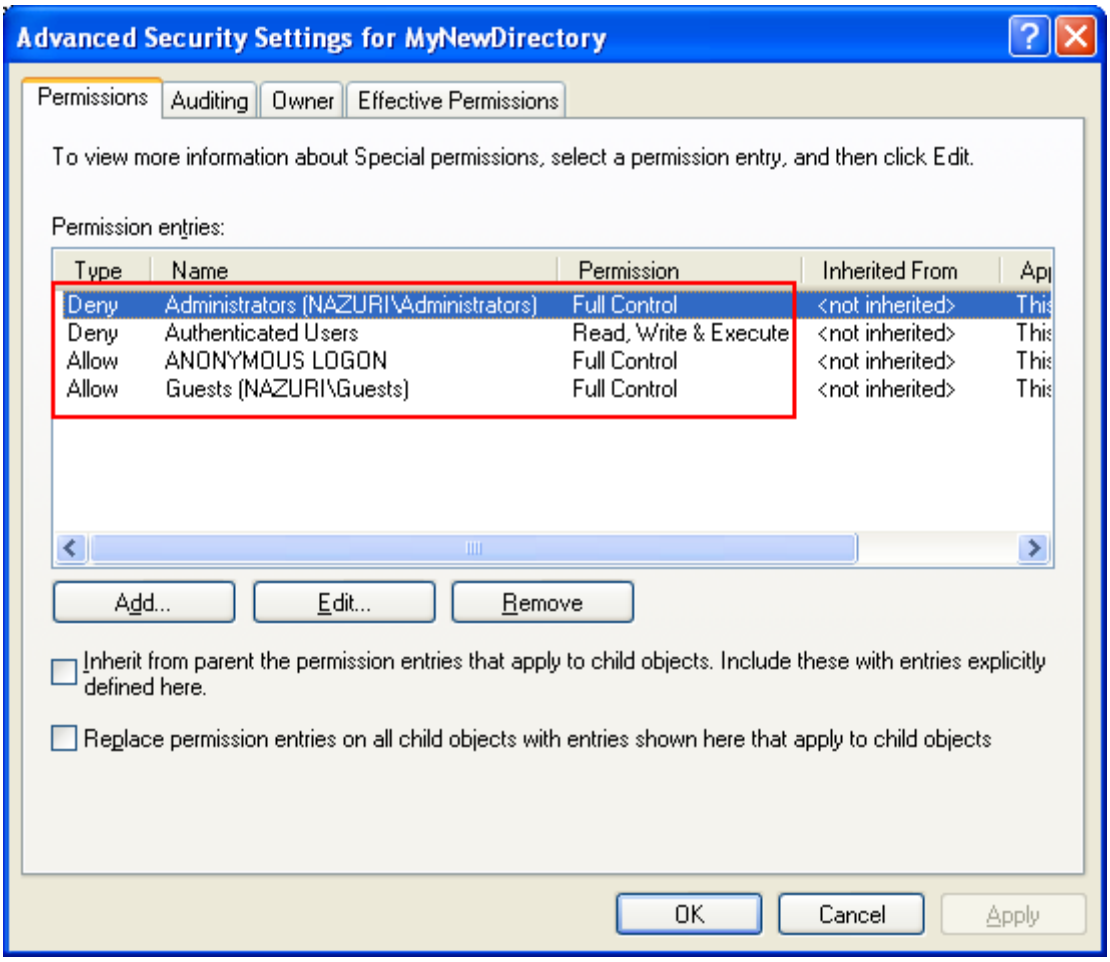

MyNewDirectory Properties
General
Sharing
Security
Web Sharing
Customize
Group or user names:
Administrators (NAZURI\Administrators)
ANONYMOUS LOGON
Authenticated Users
Guests (NAZURI\Guests)
Add...
Remove
Permissions for Administrators
Allow
Deny
Full Control
Modify
Read & Execute
List Folder Contents
Read
Write
For special permissions or for advanced settings, click Advanced.
Advanced
OK
Cancel
Apply


MyNewDirectory Properties
General
Sharing
Security
Web Sharing
Customize
Group or user names:
Administrators (NAZURI\Administrators)
ANONYMOUS LOGON
Authenticated Users
Guests (NAZURI\Guests)
Add...
Remove
Permissions for ANONYMOUS LOGON
Allow
Deny
Full Control
Modify
Read & Execute
List Folder Contents
Read
Write
For special permissions or for advanced settings, click Advanced.
Advanced
OK
Cancel
Apply

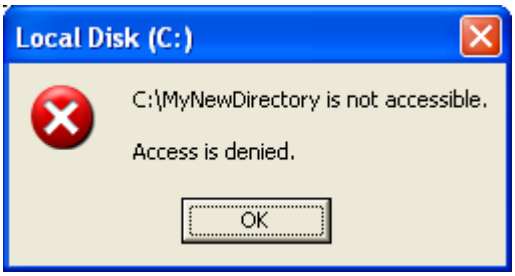

MyNewDirectory Properties
General
Sharing
Security
Web Sharing
Customize
Group or user names:
Administrators (NAZURI\Administrators)
ANONYMOUS LOGON
Authenticated Users
Guests (NAZURI\Guests)
Add...
Remove
Permissions for Authenticated Users
Allow
Deny
Modify
Read & Execute
List Folder Contents
Read
Write
Special Permissions
For special permissions or for advanced settings, click Advanced.
Advanced
OK
Cancel
Apply



Then click the Advanced button. The following Advanced Security Settings is shown. It shows the permission entries in more detail.



Try opening the directory and the access should be denied.



Try deleting the MyNewDirectory directory. Because the DENY take precedence and the ACEs is based on the "First found first served and forget the rest", you cannot delete the directory in this case.
However, in this case the computer used to run this program is logged by user Mike spoon, who is a member of the Administrators group. So he still has permission to modify the Allow, Deny Access permission.

Open the MyNewDirectory properties page. Tick the Allow tick box under the Full Control of Permissions for Administrators to enable the MyNewDirectory deletion and click OK.
It looks funny isn't it? You can't delete but you can change the permission so that you can delete!