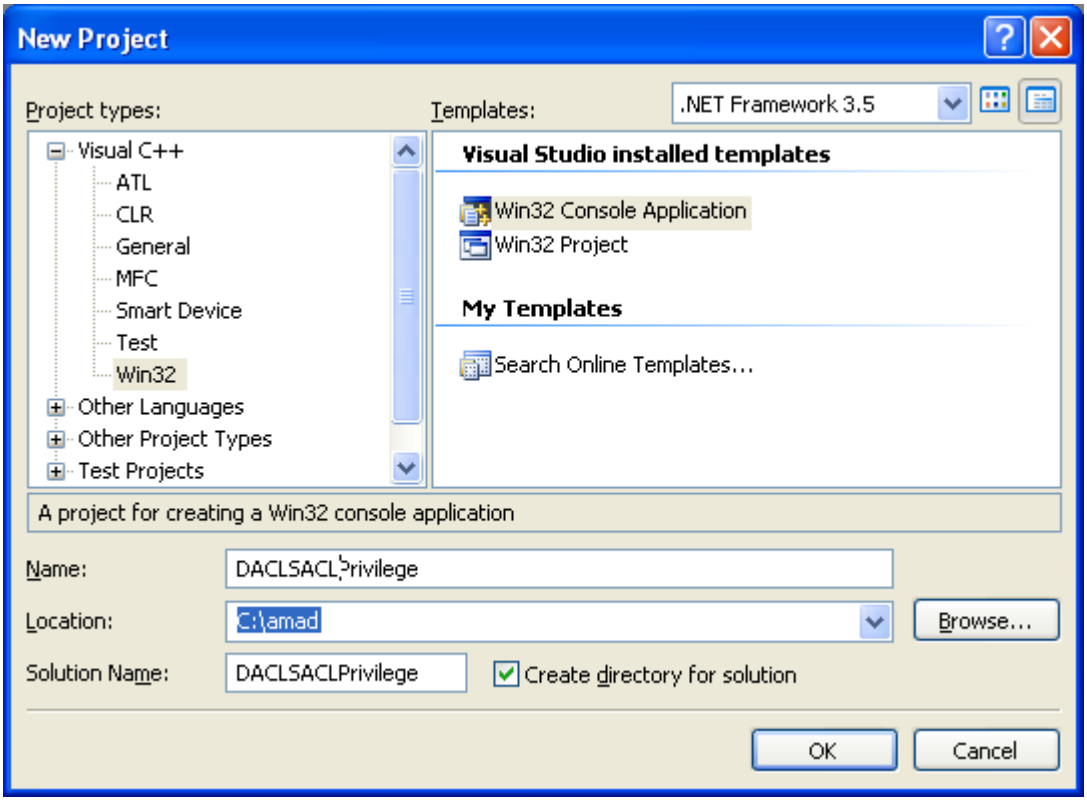
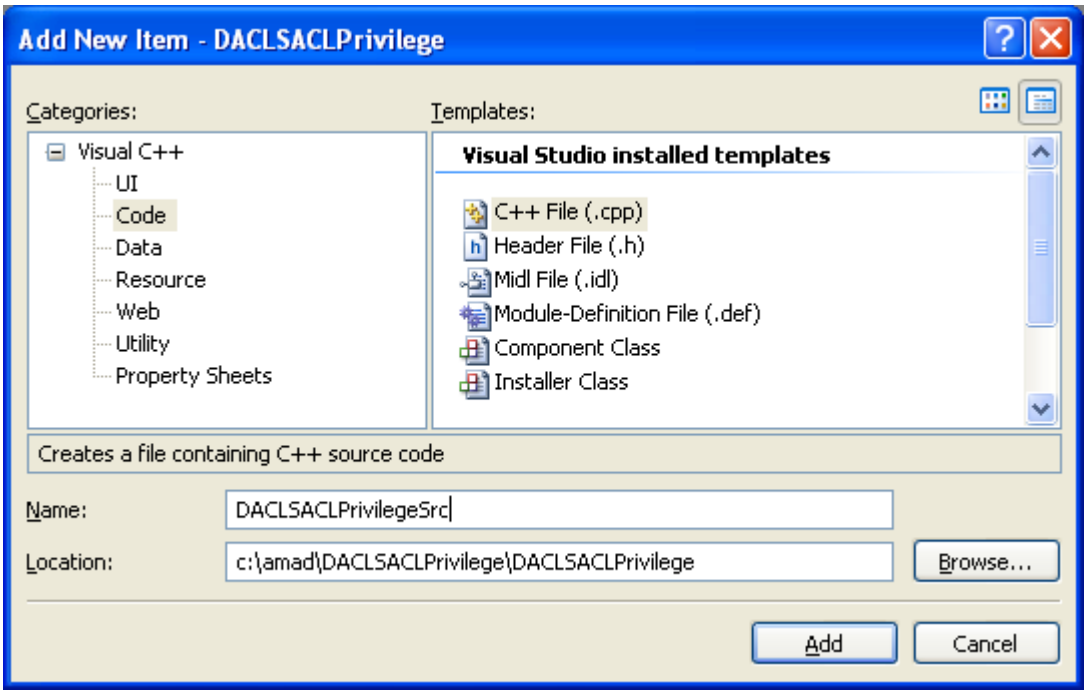
Creating DACL and SACL with the Privilege Program Example

The following program example try to create an ACL that contains both the DACL and SACL For the SACL we need to enable the SeSecurityPrivilege privilege by using the AdjustTokenPrivileges().
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.

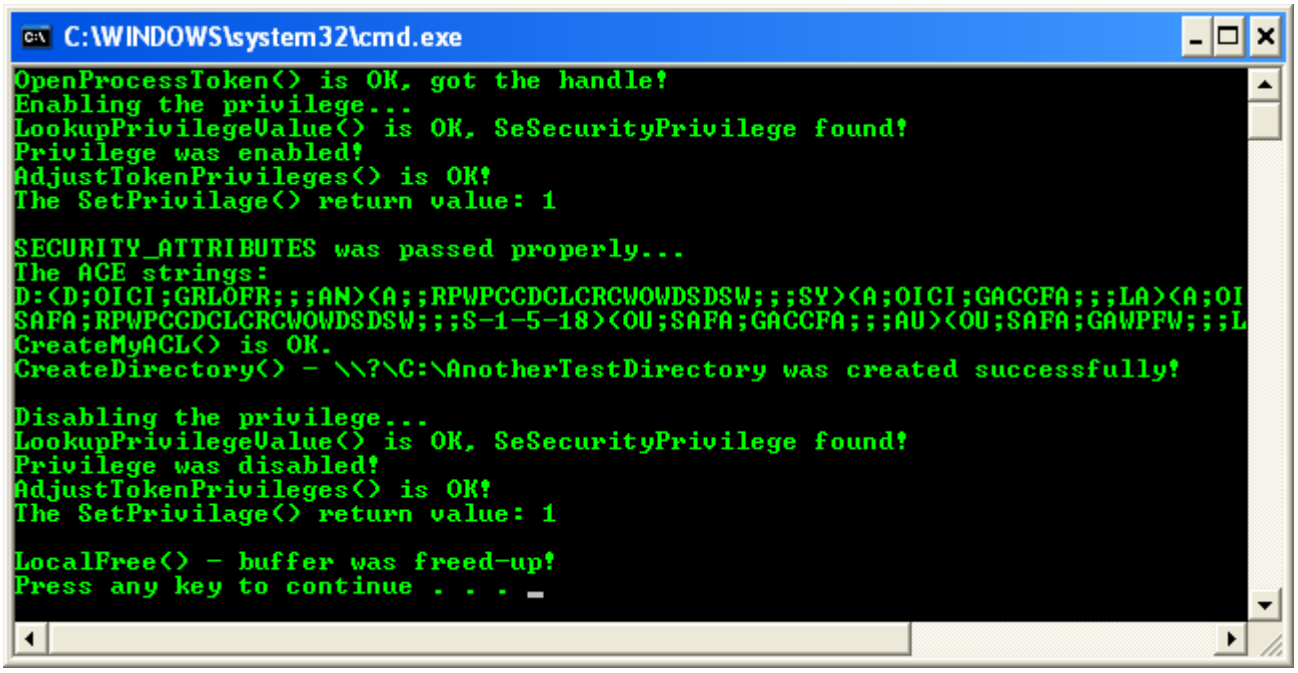
```
// Creating an ACL-DACL & SACL, need the required privilege for SACL
// The following #define must be the 1st statement
// #define _WIN32_WINNT 0x0500
#include <windows.h>
#include <sddl.h>
#include <stdio.h>

// ********** Enabling the privilege ***************
BOOL SetPrivilege(
    HANDLE hToken,          // access token handle
    LPCTSTR lpszPrivilege,  // name of privilege to enable/disable
    BOOL bEnablePrivilege   // to enable (or disable privilege)
    )
{
      TOKEN_PRIVILEGES tp;
      LUID luid;

      if(!LookupPrivilegeValue(
            NULL,            // lookup privilege on local system
            lpszPrivilege,   // privilege to lookup
            &luid))          // receives LUID of privilege
      {
            wprintf(L"LookupPrivilegeValue() failed, error: %u\n",
GetLastError());
            return FALSE;
      }
      else
            wprintf(L"LookupPrivilegeValue() is OK, %s found!\n",
lpszPrivilege);
```

```
	// the number of entries in the Privileges array
	tp.PrivilegeCount = 1;
	// an array of LUID_AND_ATTRIBUTES structures
	tp.Privileges[0].Luid = luid;

	// If TRUE
	if(bEnablePrivilege)
	{
		tp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
		wprintf(L"Privilege was enabled!\n");
	}
	else
	{
		tp.Privileges[0].Attributes = 0;
		wprintf(L"Privilege was disabled!\n");
	}

	// Enable the privilege (or disable all privileges)
	if(!AdjustTokenPrivileges(
		hToken,
		FALSE,	// If TRUE, function disables all privileges,
			// if FALSE the function modifies privileges based on
the tp
		&tp,
		sizeof(TOKEN_PRIVILEGES),
		(PTOKEN_PRIVILEGES) NULL,
		(PDWORD) NULL))
	{
		wprintf(L"AdjustTokenPrivileges() failed, error: %u\n",
GetLastError());
		return FALSE;
	}
	else
		wprintf(L"AdjustTokenPrivileges() is OK!\n");

	return TRUE;
}

//********************* Create the ACL ****************
// CreateMyACL() routine.
// The return value is FALSE if the address to the structure is NULL.
// Otherwise, this function returns the value from the
// ConvertStringSecurityDescriptorToSecurityDescriptor function.
BOOL CreateMyACL(SECURITY_ATTRIBUTES * pSA)
{
	// Define the SDDL for the DACL & SACL.
    WCHAR * szSD = L"D:"			// DACL
    L"(D;OICI;GRLOFR;;;AN)"			// Deny Anonymous some rights
    L"(A;;RPWPCCDCLCRCWOWDSDSW;;;SY)" // Allow System some rights
    L"(A;OICI;GACCFA;;;LA)"			// Allow Built-in Administrator some
rights
    L"(A;OICI;GACCFA;;;S-1-5-11)"	// Allow Authenticated user some rights
    L"S:"				// SACL
    L"(OU;SAFA;RPWPCCDCLCRCWOWDSDSW;;;S-1-5-18)" // Object audit
success/fail, Local systems, using a SID string
    L"(OU;SAFA;GACCFA;;;AU)"  // Object audit success/fail, Authenticated
users
```

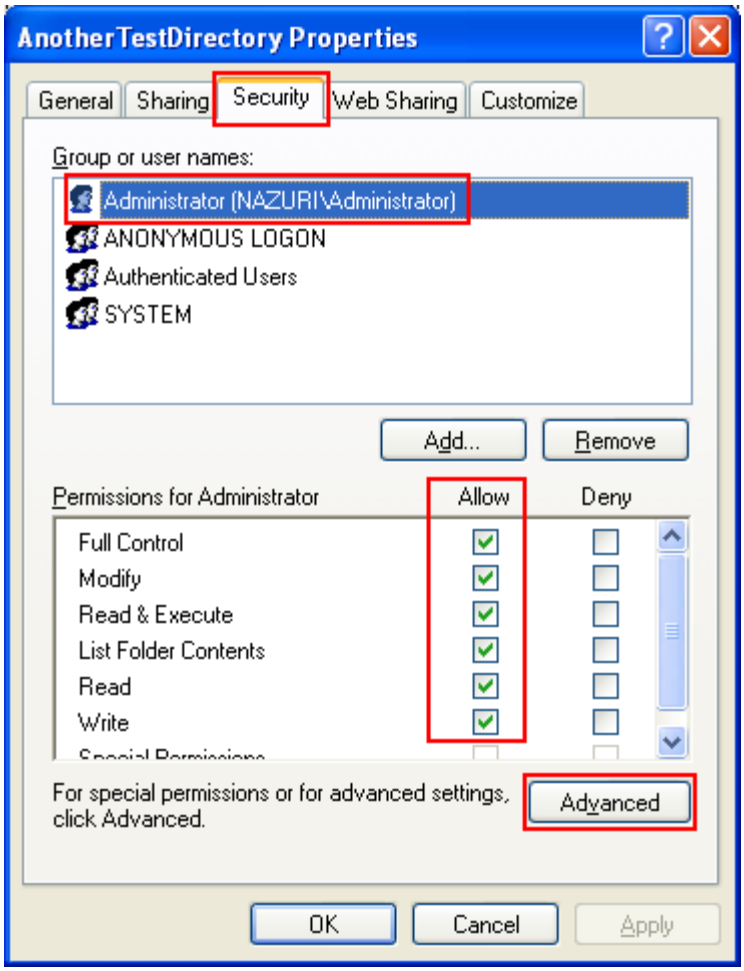
```
    L"(OU;SAFA;GAWPFW;;;LA)"; // Object audit success/fail, Built-in
Administrator

      // Verify
      if(pSA == NULL)
      {
            wprintf(L"SECURITY_ATTRIBUTES wasn\'t passed properly...\n");
            return FALSE;
      }
      else
            wprintf(L"SECURITY_ATTRIBUTES was passed properly...\n");

      // Do some verification
      wprintf(L"The ACE strings:\n%s \n", szSD);
      // Convert to 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
            NULL);                                                 // The size in
byte the converted security descriptor
}

//************ main ******************
int wmain(int argc, WCHAR **argv)
{
      WCHAR DirName[] = L"\\\\?\\C:\\AnotherTestDirectory";
      SECURITY_ATTRIBUTES  sa;

      LPCTSTR lpszPrivilege = L"SeSecurityPrivilege";
      // Initially we try to enable
      BOOL bEnablePrivilege = TRUE;
      // Handle to the running process that is this program
      HANDLE hToken;
      BOOL bTestRetVal = FALSE;

      //*************** Get the handle to the process ***************
      // Open a handle to the access token for the calling process.
      // That is this running program
      if(!OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES,
&hToken))
      {
            wprintf(L"OpenProcessToken() failed, error %u\n",
GetLastError());
            return FALSE;
      }
      else
            wprintf(L"OpenProcessToken() is OK, got the handle!\n");

      //************ Enabling privilege *****************
      // Call the user defined SetPrivilege() function to enable privilege
      wprintf(L"Enabling the privilege...\n");
      bTestRetVal = SetPrivilege(hToken, lpszPrivilege, bEnablePrivilege);
      // Verify
      wprintf(L"The SetPrivilage() return value: %d\n\n", bTestRetVal);
      //*************** End enabling privilege ********************
```

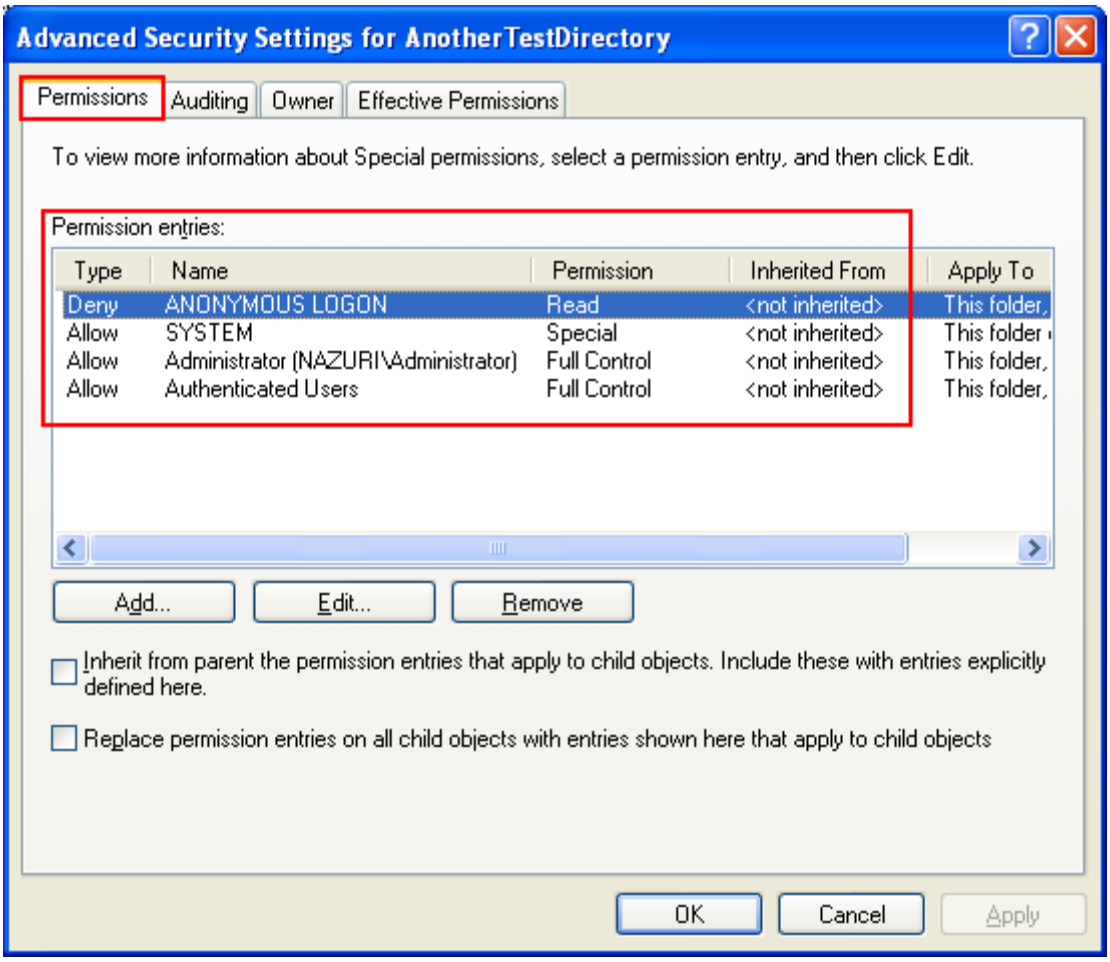
```
	// The SECURITY_ATTRIBUTE structure size
	sa.nLength = sizeof(SECURITY_ATTRIBUTES);
	// The return handle not inherited
	sa.bInheritHandle = FALSE;
	// Call CreateMyACL() function to set the DACL and SACL,
	// is set in the SECURITY_ATTRIBUTES lpSecurityDescriptor member
	if(!CreateMyACL(&sa))
	{
		// Error encountered; generate message and just exit.
		wprintf(L"CreateMyACL() failed, error %u.\n", GetLastError());
		exit(1);
	}
	else
		wprintf(L"CreateMyACL() is OK.\n");

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

	//************* Disable the privilege ***************
	wprintf(L"Disabling the privilege...\n");
	bEnablePrivilege = FALSE;
	SetPrivilege(hToken, lpszPrivilege, bEnablePrivilege);
	// Verify
	wprintf(L"The SetPrivilage() return value: %d\n\n", bTestRetVal);
	//************* End disabling the privilege ***************

	//************ Clean up ******************
	// Release the memory allocated for the SECURITY_DESCRIPTOR
	// This means release back the used memory to the system
	if(LocalFree(sa.lpSecurityDescriptor) != NULL)
	{
		// Error encountered; generate message and just exit
		wprintf(L"LocalFree() failed, error %u.\n", GetLastError());
		exit(1);
	}
	else
		wprintf(L"LocalFree() - buffer was freed-up!\n");
	return 0;
}
```

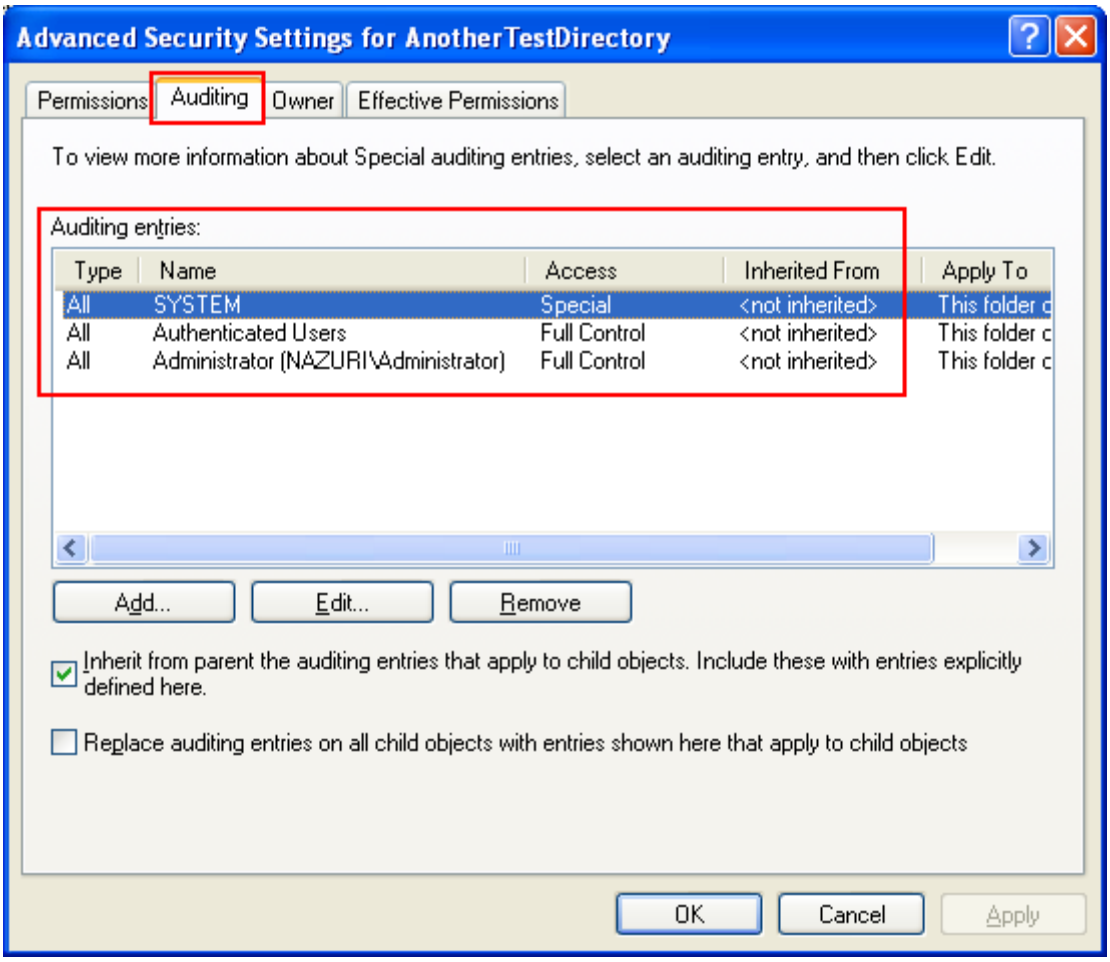
Build and run the project. The following screenshot is a sample output.



Let verify through the folder's property page.


AnotherTestDirectory Properties
General
Sharing
Security
Web Sharing
Customize
Group or user names:
Administrator (NAZURI\Administrator)
ANONYMOUS LOGON
Authenticated Users
SYSTEM
Add...
Remove
Permissions for Administrator
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

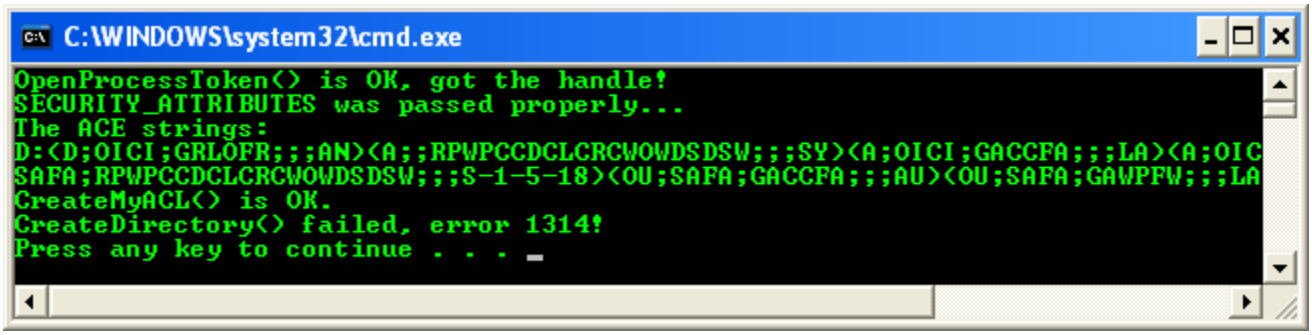

Advanced Security Settings for AnotherTestDirectory
Permissions
Auditing
Owner
Effective Permissions
To view more information about Special permissions, select a permission entry, and then click Edit.
Permission entries:
Type
Name
Permission
Inherited From
Apply To
Deny
ANONYMOUS LOGON
Read
<not inherited>
This folder,
Allow
SYSTEM
Special
<not inherited>
This folder
Allow
Administrator (NAZURI\Administrator)
Full Control
<not inherited>
This folder,
Allow
Authenticated Users
Full Control
<not inherited>
This folder,
Add...
Edit...
Remove
Inherit from parent the permission entries that apply to child objects. Include these with entries explicitly defined here.
Replace permission entries on all child objects with entries shown here that apply to child objects
OK
Cancel
Apply



Without enabling the required privilege, the CreateDirectory() will fail because we don't have privilege to create SACL for auditing. In order to test this, delete the created, C:\AnotherTestDirectory directory and comment out the following code portion:

```
...
/*
//************* Enabling privilege ******************
// Call the user defined SetPrivilege() function to enable privilege
wprintf(L"Enabling the privilege...\n");
bTestRetVal = SetPrivilege(hToken, lpszPrivilege, bEnablePrivilege);
// Verify
wprintf(L"The SetPrivilage() return value: %d\n\n", bTestRetVal);
//*************** End enabling privilege ********************
*/
...
```

Re-run the program, the following output should be expected.



The 1314 (0x522 - ERROR_PRIVILEGE_NOT_HELD) error constant means a required privilege is not held by the client. The attributes of a privilege can be a combination of the following values.

Value	Meaning
SE_PRIVILEGE_ENABLED	The privilege is enabled.
SE_PRIVILEGE_ENABLED_BY_DEFAULT	The privilege is enabled by default.
SE_PRIVILEGE_REMOVED	Used to remove a privilege.
SE_PRIVILEGE_USED_FOR_ACCESS	The privilege was used to gain access to an object or service. This flag is used to identify the relevant privileges in a set passed by a client application that may contain unnecessary privileges.