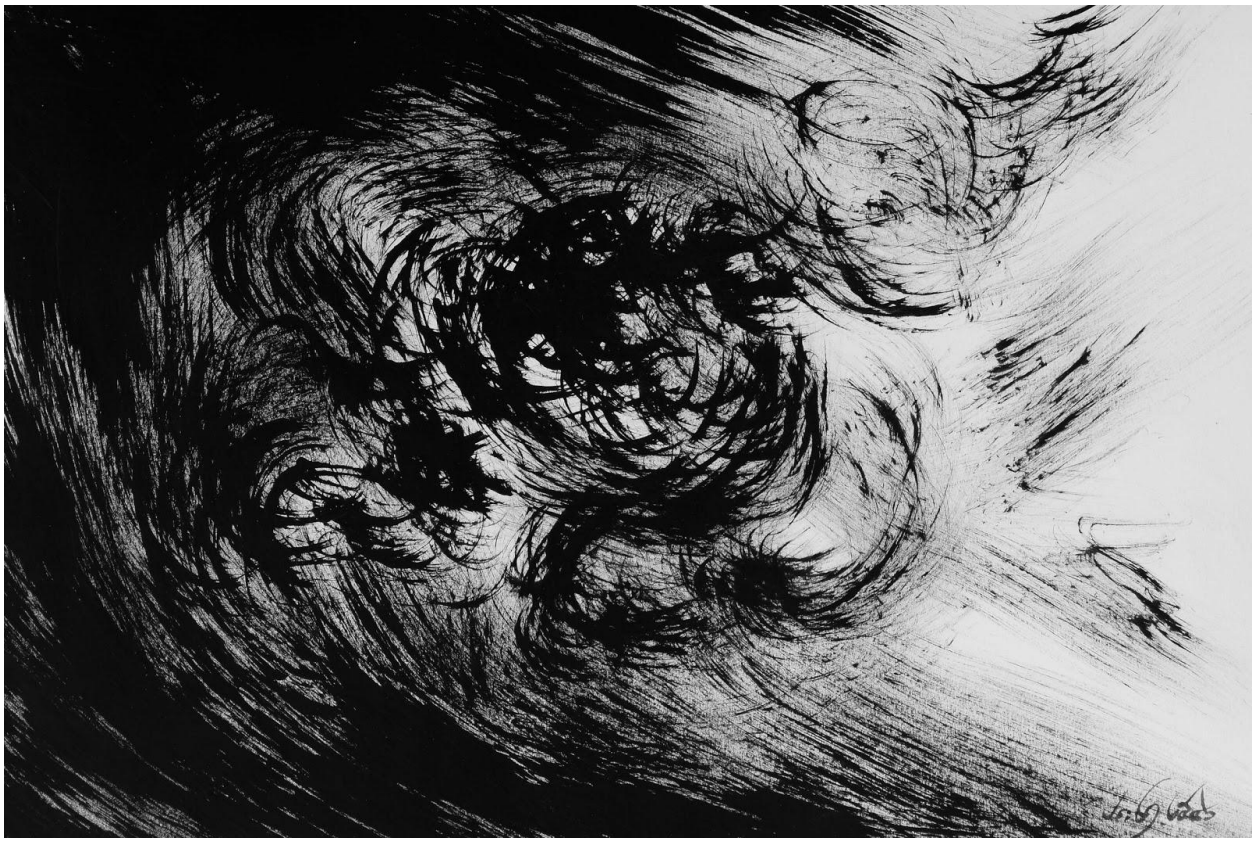
USB PORT DISABLER

by// Ethereal



Introduction:

The objective of the following code is to disable USB ports on a system. Getting the hardware id's for the devices is a multi-step round about process, which requires calling several of the SetupDixxx() functions provided by the setupapi.dll. The programmatic implementation will be written in C using the WINAPI.

The CODE:

1. First populate a list of plugged in USB devices using SetupDiGetClassDevsW by specifying DIGCF_PRESENT | DIGCF_DEVICEINTERFACE flags.

```
DeviceInfoSet = SetupDiGetClassDevsW((LPGUID)&GUID_DEVINTERFACE_USB_DEVICE, NULL, NULL,
DIGCF_PRESENT | DIGCF_DEVICEINTERFACE);

if (DeviceInfoSet == INVALID_HANDLE_VALUE)
{
   return FALSE;
}
```

2. Initialize an appropriate SP_DEVINFO_DATA structure to enumerate through the USB devices. We need this structure for SetupDiGetDeviceRegistryProperty().

```
ZeroMemory(&DeviceInfoData, sizeof(SP_DEVINFO_DATA));
DeviceInfoData.cbSize = sizeof(SP_DEVINFO_DATA);

while (true)
{
   if (!SetupDiEnumDeviceInfo(DeviceInfoSet, DeviceIndex, &DeviceInfoData))
       {
              dwError = GetLastError();
              if (dwError == ERROR_NO_MORE_ITEMS)
              {
                     break;
              }

              else
              {
                     goto FAILURE;
              }
       }
```

3. Query for the size of the hardware ID using SetupDiGetDeviceRegistryPropertyW to allocate appropriate buffer to store data.

```
SetupDiGetDeviceRegistryPropertyW(DeviceInfoSet, &DeviceInfoData, SPDRP_HARDWAREID, &dwRegType,
NULL, 0, &dwBuffersize);


devBuffer = (PBYTE)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, dwBuffersize);

if (devBuffer == NULL)
{
    goto FAILURE;
}
```

4. Retrieve the hardware IDs of the USB Devices enumerated using SetupDiGetDeviceRegistryPropertyW.

```
if (!SetupDiGetDeviceRegistryPropertyW(DeviceInfoSet, &DeviceInfoData,
SPDRP_HARDWAREID, &dwRegType, (PBYTE)devBuffer, dwBuffersize, 0))
{
   goto FAILURE;
}
```

5. Create SP_CLASSINTALL_HEADER which is the first parameter of any class install parameter structure and it contains the device installation request code. Further an SP_PROPCHANGE_PARAMS structure which corresponds to a DIF_PROPERTY_CHANGE installation request would be used to set .StateChange to Disable.

```
SP_CLASSINSTALL_HEADER header;
header.cbSize = sizeof(SP_CLASSINSTALL_HEADER);
header.InstallFunction = DIF_PROPERTYCHANGE;

SP_PROPCHANGE_PARAMS ChangeDeviceParams;
ChangeDeviceParams.ClassInstallHeader = header;
ChangeDeviceParams.StateChange = DICS_DISABLE;
ChangeDeviceParams.Scope = DICS_FLAG_GLOBAL;
ChangeDeviceParams.HwProfile = 0;
```

6. Use SetupDiSetClassInstallParamsW & SetupDiChangeState to change the state of enumerated USB Devices to disabled. Clean up the DeviceInfoSet structure and keep looping until all USB devices are found and disabled.

```
if (!SetupDiSetClassInstallParamsW(DeviceInfoSet, &DeviceInfoData,
(SP_CLASSINSTALL_HEADER*)&ChangeDeviceParams, sizeof(ChangeDeviceParams)))
{
   goto FAILURE;
}

if (!SetupDiChangeState(DeviceInfoSet, &DeviceInfoData))
{
   goto FAILURE;
}

SetupDiDestroyDeviceInfoList(DeviceInfoSet);
HeapFree(GetProcessHeap(), 0, devBuffer);
DeviceIndex++;
}
```

7. In the event any of the code fails we safely exit by jumping to our exit routine “FAILURE”. It closes the handle “DeviceInfoSet” , frees up the heap memory allocated.

```
FAILURE:

dwError = GetLastError();
HeapFree(GetProcessHeap(), 0, devBuffer);
SetupDiDestroyDeviceInfoList(DeviceInfoSet);

  return dwError;
}
```