

## 1. Check if Antivirus is Installed

### Explanation:

- **Microsoft.Win32:** Used for accessing the Windows Registry.
- **RegistryKey:** Represents a key-level node in the Windows Registry.
- **OpenSubKey():** Opens a subkey for read access.
- **.NET 4.6 Support:** Microsoft.Win32 is included in .NET 4.6, so no additional installation is required.

### C# Code:

```
csharp
CopyEdit
using System;
using Microsoft.Win32;

class AntivirusCheck
{
    static void Main()
    {
        string registryPath =
@"SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall";
        bool antivirusFound = false;

        using (RegistryKey key =
Registry.LocalMachine.OpenSubKey(registryPath))
        {
            if (key != null)
            {
                foreach (string subkeyName in key.GetSubKeyNames())
                {
                    using (RegistryKey subkey = key.OpenSubKey(subkeyName))
                    {
                        string displayName =
subkey?.GetValue("DisplayName") as string;
                        if (!string.IsNullOrEmpty(displayName) &&
displayName.ToLower().Contains("antivirus"))
                        {
                            antivirusFound = true;
                            break;
                        }
                    }
                }
            }
        }

        Console.WriteLine($"Antivirus Installed: {antivirusFound}");
    }
}
```

---

## 2. Check Last Update Date of Antivirus

### Explanation:

- **WMI (Windows Management Instrumentation):** Used to query antivirus update information.
- **System.Management:** Provides access to WMI data and events.
- **ManagementObjectSearcher:** Queries WMI for antivirus update data.
- **.NET 4.6 Support:** System.Management is **not included** by default in .NET 4.6.

### Offline Installation Steps:

1. Open **Visual Studio Installer**.
2. Select **.NET Framework 4.6 Development Tools**.
3. Enable **WMI Provider Host**.

### C# Code:

```
csharp
CopyEdit
using System;
using System.Management;

class AntivirusUpdateCheck
{
    static void Main()
    {
        string query = "SELECT * FROM AntiVirusProduct";
        using (var searcher = new
ManagementObjectSearcher(@"root\SecurityCenter2", query))
        {
            foreach (ManagementObject obj in searcher.Get())
            {
                string lastUpdated = obj["timestamp"]?.ToString();
                Console.WriteLine($"Last Update Date of Antivirus:
{lastUpdated}");
            }
        }
    }
}
```

---

I'll continue adding the rest of the checks in the same format. Let me know if you need any modifications before proceeding! 🚀

## 3. Check if Licensed Antivirus is Installed

### Explanation:

- **Microsoft.Win32:** Used to check Windows Registry for antivirus license details.
- **Registry.GetValue():** Retrieves the value of a specified registry key.
- **.NET 4.6 Support:** Microsoft.Win32 is included in .NET 4.6, so no additional installation is required.

### C# Code:

```
csharp
CopyEdit
```

```

using System;
using Microsoft.Win32;

class LicensedAntivirusCheck
{
    static void Main()
    {
        string registryPath =
@"SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall";
        bool licensedAntivirus = false;

        using (RegistryKey key =
Registry.LocalMachine.OpenSubKey(registryPath))
        {
            if (key != null)
            {
                foreach (string subkeyName in key.GetSubKeyNames())
                {
                    using (RegistryKey subkey = key.OpenSubKey(subkeyName))
                    {
                        string displayName =
subkey?.GetValue("DisplayName") as string;
                        string licenseStatus = subkey?.GetValue("License")
as string;

                        if (!string.IsNullOrEmpty(displayName) &&
displayName.ToLower().Contains("antivirus") &&
!string.IsNullOrEmpty(licenseStatus) &&
licenseStatus.ToLower().Contains("valid"))
                        {
                            licensedAntivirus = true;
                            break;
                        }
                    }
                }
            }
        }

        Console.WriteLine($"Licensed Antivirus Installed:
{licensedAntivirus}");
    }
}

```

---

## 4. Check if Antivirus Logs are Configured

### Explanation:

- **System.IO:** Used to check the existence of antivirus log files.
- **File.Exists():** Checks if a file exists in a specific location.
- **.NET 4.6 Support:** System.IO is included in .NET 4.6, so no additional installation is required.

### C# Code:

```

csharp
CopyEdit
using System;

```

```
using System.IO;

class AntivirusLogCheck
{
    static void Main()
    {
        string logPath = @"C:\ProgramData\Antivirus\Logs\log.txt";
        bool logsConfigured = File.Exists(logPath);

        Console.WriteLine($"Antivirus Logs Configured: {logsConfigured}");
    }
}
```

---

## 5. Check Last Update Date of Operating System

### Explanation:

- **System.Management:** Used to query WMI for OS update details.
- **ManagementObjectSearcher:** Executes a WMI query to fetch OS last update details.
- **.NET 4.6 Support:** System.Management is **not included** by default in .NET 4.6.

### Offline Installation Steps:

1. Open **Visual Studio Installer**.
2. Select **.NET Framework 4.6 Development Tools**.
3. Enable **WMI Provider Host**.

### C# Code:

```
csharp
CopyEdit
using System;
using System.Management;

class OSUpdateCheck
{
    static void Main()
    {
        string query = "SELECT * FROM Win32_OperatingSystem";
        using (var searcher = new ManagementObjectSearcher(query))
        {
            foreach (ManagementObject obj in searcher.Get())
            {
                string lastUpdated = obj["InstallDate"]?.ToString();
                Console.WriteLine($"Last Update Date of OS:
{lastUpdated}");
            }
        }
    }
}
```

---

I'll continue adding the remaining checks in the same format. Let me know if you need any modifications before proceeding! 🚀

## 6. Check if User is Not Given Root Access

### Explanation:

- **System.Security.Principal:** Used to check if the current user is an administrator.
- **WindowsIdentity:** Represents the Windows user account executing the code.
- **WindowsPrincipal:** Determines the user's role.
- **.NET 4.6 Support:** `System.Security.Principal` is included in .NET 4.6, so no additional installation is required.

### C# Code:

```
csharp
CopyEdit
using System;
using System.Security.Principal;

class RootAccessCheck
{
    static void Main()
    {
        WindowsIdentity identity = WindowsIdentity.GetCurrent();
        WindowsPrincipal principal = new WindowsPrincipal(identity);

        bool isAdmin =
principal.IsInRole(WindowsBuiltInRole.Administrator);

        Console.WriteLine($"User has root access: {isAdmin}");
    }
}
```

---

## 7. Check if Licensed OS is Installed

### Explanation:

- **System.Management:** Used to retrieve OS licensing information via WMI.
- **Win32\_OperatingSystem:** Retrieves OS licensing details.
- **.NET 4.6 Support:** `System.Management` is **not included** by default in .NET 4.6.

### Offline Installation Steps:

1. Open **Visual Studio Installer**.
2. Select **.NET Framework 4.6 Development Tools**.
3. Enable **WMI Provider Host**.

### C# Code:

```
csharp
CopyEdit
using System;
using System.Management;

class OSLicenseCheck
```

```

{
    static void Main()
    {
        string query = "SELECT * FROM Win32_OperatingSystem";
        using (var searcher = new ManagementObjectSearcher(query))
        {
            foreach (ManagementObject obj in searcher.Get())
            {
                string licenseStatus = obj["SerialNumber"]?.ToString();
                Console.WriteLine($"Licensed OS Installed:
{!string.IsNullOrEmpty(licenseStatus)}");
            }
        }
    }
}

```

---

## 8. Check if USB Storage Media is Blocked

### Explanation:

- **Microsoft.Win32:** Used to check Windows Registry settings for USB storage access.
- **Registry.GetValue():** Reads the value of a registry key.
- **.NET 4.6 Support:** Microsoft.Win32 is included in .NET 4.6, so no additional installation is required.

### C# Code:

```

csharp
CopyEdit
using System;
using Microsoft.Win32;

class USBBlockCheck
{
    static void Main()
    {
        string registryPath =
@"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\USBSTOR";
        string valueName = "Start";

        var value = Registry.GetValue(registryPath, valueName, null);
        bool usbBlocked = value != null && Convert.ToInt32(value) == 4;

        Console.WriteLine($"USB Storage Media Blocked: {usbBlocked}");
    }
}

```

---

## 9. Check if IP Tables are Used in the System

### Explanation:

- **System.Diagnostics:** Used to run system commands.
- **Process.Start():** Executes system commands (like checking firewall rules).

- **.NET 4.6 Support:** `System.Diagnostics` is included in .NET 4.6, so no additional installation is required.

### C# Code:

```
csharp
CopyEdit
using System;
using System.Diagnostics;

class IPTablesCheck
{
    static void Main()
    {
        ProcessStartInfo psi = new ProcessStartInfo
        {
            FileName = "cmd.exe",
            Arguments = "/c netsh advfirewall show allprofiles",
            RedirectStandardOutput = true,
            UseShellExecute = false,
            CreateNoWindow = true
        };

        Process process = Process.Start(psi);
        string output = process.StandardOutput.ReadToEnd();
        process.WaitForExit();

        bool ipTablesUsed = output.ToLower().Contains("firewall policy");

        Console.WriteLine($"IP Tables Used: {ipTablesUsed}");
    }
}
```

---

I'll continue with the remaining checks. Let me know if you need any modifications before proceeding! 🚀

## 10. Check Number of USB Pen Drives Connected

### Explanation:

- **System.Management:** Used to query USB devices.
- **Win32\_DiskDrive:** Retrieves storage device details.
- **.NET 4.6 Support:** `System.Management` is **not included** by default.

### Offline Installation Steps:

1. Open **Visual Studio Installer**.
2. Select **.NET Framework 4.6 Development Tools**.
3. Enable **WMI Provider Host**.

### C# Code:

```
csharp
CopyEdit
```

```

using System;
using System.Management;

class USBCountCheck
{
    static void Main()
    {
        int usbCount = 0;
        string query = "SELECT * FROM Win32_DiskDrive WHERE
InterfaceType='USB'";

        using (var searcher = new ManagementObjectSearcher(query))
        {
            foreach (var obj in searcher.Get())
            {
                usbCount++;
            }
        }

        Console.WriteLine($"Number of USB Pen Drives Connected:
{usbCount}");
    }
}

```

---

## 11. Check if SE Linux is Installed

### Explanation:

- **System.IO:** Used to check the presence of SELinux configuration files.
- **.NET 4.6 Support:** System.IO is included in .NET 4.6, so no additional installation is required.

### C# Code:

```

csharp
CopyEdit
using System;
using System.IO;

class SELinuxCheck
{
    static void Main()
    {
        bool seLinuxInstalled = File.Exists("/etc/selinux/config");
        Console.WriteLine($"SE Linux Installed: {seLinuxInstalled}");
    }
}

```

---

## 12. Check if No Traces of Smartphone/Dongle are Found

### Explanation:

- **System.Management:** Used to list connected USB devices.
- **Win32\_PnPEntity:** Queries connected hardware.



- **.NET 4.6 Support:** `System.Management` is **not included** by default.

## Offline Installation Steps:

1. Install **WMI Provider Host** via Visual Studio Installer.

### C# Code:

```
csharp
CopyEdit
using System;
using System.Management;

class SmartphoneCheck
{
    static void Main()
    {
        string query = "SELECT * FROM Win32_PnPEntity WHERE Name LIKE '%Android%' OR Name LIKE '%iPhone%' OR Name LIKE '%Modem%'";

        using (var searcher = new ManagementObjectSearcher(query))
        {
            bool found = searcher.Get().Count > 0;
            Console.WriteLine($"Traces of Smartphone/Dongle Found: {found}");
        }
    }
}
```

---

## 13. Check if Telnet is Not Installed

### Explanation:

- **System.Diagnostics:** Used to check installed features.
- **.NET 4.6 Support:** `System.Diagnostics` is included in .NET 4.6, no installation required.

### C# Code:

```
csharp
CopyEdit
using System;
using System.Diagnostics;

class TelnetCheck
{
    static void Main()
    {
        ProcessStartInfo psi = new ProcessStartInfo("cmd.exe", "/c telnet")
        {
            RedirectStandardError = true,
            UseShellExecute = false,
            CreateNoWindow = true
        };
    }
}
```

```

        Process process = Process.Start(psi);
        string errorOutput = process.StandardError.ReadToEnd();
        process.WaitForExit();

        bool telnetNotInstalled = errorOutput.Contains("not recognized");
        Console.WriteLine($"Telnet Installed: {!telnetNotInstalled}");
    }
}

```

---

## 14. Check if Autoplay is Disabled

### Explanation:

- **Microsoft.Win32:** Used to check Windows Registry settings.
- **.NET 4.6 Support:** Microsoft.Win32 is included in .NET 4.6.

### C# Code:

```

csharp
CopyEdit
using System;
using Microsoft.Win32;

class AutoplayCheck
{
    static void Main()
    {
        string registryPath =
@"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\Exp
lorer";
        string valueName = "NoDriveTypeAutoRun";

        var value = Registry.GetValue(registryPath, valueName, null);
        bool autoplayDisabled = value != null && Convert.ToInt32(value) ==
255;

        Console.WriteLine($"Autoplay Disabled: {autoplayDisabled}");
    }
}

```

---

## 15. Check if NTP is Installed

### Explanation:

- **System.Diagnostics:** Used to check Windows services.
- **.NET 4.6 Support:** System.Diagnostics is included in .NET 4.6.

### C# Code:

```

csharp
CopyEdit
using System;
using System.Diagnostics;

```

```

class NTPCheck
{
    static void Main()
    {
        ProcessStartInfo psi = new ProcessStartInfo("cmd.exe", "/c w32tm
/query /status")
        {
            RedirectStandardOutput = true,
            UseShellExecute = false,
            CreateNoWindow = true
        };

        Process process = Process.Start(psi);
        string output = process.StandardOutput.ReadToEnd();
        process.WaitForExit();

        bool ntpInstalled = output.ToLower().Contains("source");
        Console.WriteLine($"NTP Installed: {ntpInstalled}");
    }
}

```

---

## 16. Check if Wi-Fi is Disabled

### Explanation:

- **System.Management:** Queries network adapter details.
- **.NET 4.6 Support:** `System.Management` is **not included** by default.

### Offline Installation Steps:

1. Install **WMI Provider Host** via Visual Studio Installer.

### C# Code:

```

csharp
CopyEdit
using System;
using System.Management;

class WiFiCheck
{
    static void Main()
    {
        using (var searcher = new ManagementObjectSearcher("SELECT * FROM
Win32_NetworkAdapter WHERE NetConnectionID != NULL"))
        {
            foreach (ManagementObject obj in searcher.Get())
            {
                string name = obj["Name"].ToString().ToLower();
                if (name.Contains("wi-fi") || name.Contains("wireless"))
                {
                    Console.WriteLine($"Wi-Fi Enabled:
{Convert.ToBoolean(obj["NetEnabled"])}");
                    return;
                }
            }
        }
    }
}

```

```
    }
    Console.WriteLine("Wi-Fi Disabled");
}
}
```

---

## 17. Check if Bluetooth is Disabled

### Explanation:

- **System.Management:** Used to check Bluetooth status.
- **.NET 4.6 Support:** `System.Management` is **not included** by default.

### Offline Installation Steps:

1. Install **WMI Provider Host** via Visual Studio Installer.

### C# Code:

```
csharp
CopyEdit
using System;
using System.Management;

class BluetoothCheck
{
    static void Main()
    {
        using (var searcher = new ManagementObjectSearcher("SELECT * FROM Win32_PnPEntity WHERE Name LIKE '%Bluetooth%'"))
        {
            bool found = searcher.Get().Count > 0;
            Console.WriteLine($"Bluetooth Enabled: {found}");
        }
    }
}
```

---

This completes all your system security checks with separate code, explanations, and offline installation steps. 🚀 Let me know if you need modifications!

## 26. Check if USB Storage Media Access is Blocked

### Explanation:

- `Microsoft.Win32:` Provides access to the Windows Registry.
- `Registry.GetValue():` Fetches a registry key's value.
- **.NET 4.6 Support:** `Microsoft.Win32` is included in .NET 4.6; no installation required.

```
csharp
CopyEdit
using System;
using Microsoft.Win32;
```

```

class USBStorageCheck
{
    static void Main()
    {
        string registryPath =
@"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\UsbStor";
        string valueName = "Start";

        object value = Registry.GetValue(registryPath, valueName, null);
        bool isBlocked = value != null && Convert.ToInt32(value) == 4;

        Console.WriteLine($"USB storage media access blocked:
{isBlocked}");
    }
}

```

---

## 27. Check if Shared Folders are Configured

### Explanation:

- `System.IO`: Used for file and directory operations.
- `Directory.GetDirectories(path)`: Lists shared folders.
- **.NET 4.6 Support:** `System.IO` is included in .NET 4.6; no installation required.

```

csharp
CopyEdit
using System;
using System.IO;

class SharedFoldersCheck
{
    static void Main()
    {
        string sharedFolderPath = @"C:\Users\Public";
        string[] sharedFolders =
Directory.GetDirectories(sharedFolderPath);

        Console.WriteLine($"Shared folders configured:
{sharedFolders.Length > 0}");
    }
}

```

---

## 28. Check if Unwanted Browsing History is Present

### Explanation:

- `System.IO`: Used to check if browser history files exist.
- **.NET 4.6 Support:** `System.IO` is included in .NET 4.6; no installation required.

```

csharp
CopyEdit
using System;
using System.IO;

```

```

class BrowsingHistoryCheck
{
    static void Main()
    {
        string historyPath =
@"C:\Users\%USERNAME%\AppData\Local\Microsoft\Windows\History";
        bool historyExists = Directory.Exists(historyPath) &&
Directory.GetFiles(historyPath).Length > 0;

        Console.WriteLine($"Unwanted browsing history present:
{historyExists}");
    }
}

```

---

## 29. Check if Traces of Connecting Smartphone/Dongle Exist

### Explanation:

- `System.Management`: Used for querying hardware devices via WMI.
- `ManagementObjectSearcher`: Executes a WMI query.
- **.NET 4.6 Support**: `System.Management` is **not included** by default. Install it offline using:
  1. Open **Visual Studio Installer**.
  2. Select **.NET Framework 4.6 Development Tools**.
  3. Enable **WMI Provider Host** (Windows Management Instrumentation).

```

csharp
CopyEdit
using System;
using System.Management;

class SmartphoneDongleCheck
{
    static void Main()
    {
        using (var searcher = new ManagementObjectSearcher("SELECT * FROM
Win32_PnPEntity WHERE Name LIKE '%Mobile%' OR Name LIKE '%Modem%'"))
        {
            bool found = false;
            foreach (ManagementObject obj in searcher.Get())
            {
                found = true;
                break;
            }
            Console.WriteLine($"Traces of smartphone/dongle connection:
{found}");
        }
    }
}

```

---

## 30. Check if Prohibited Software is Installed

### Explanation:

- `Microsoft.Win32`: Provides access to the Windows Registry.
- `RegistryKey.OpenSubKey()`: Opens a registry key for reading installed software.
- **.NET 4.6 Support:** `Microsoft.Win32` is included in .NET 4.6; no installation required.

```
csharp
CopyEdit
using System;
using Microsoft.Win32;

class ProhibitedSoftwareCheck
{
    static void Main()
    {
        string[] prohibitedApps = { "uTorrent", "VPN Client", "Crack Tool"
};
        string registryPath =
@"SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall";

        using (RegistryKey key =
Registry.LocalMachine.OpenSubKey(registryPath))
        {
            foreach (string subKeyName in key.GetSubKeyNames())
            {
                using (RegistryKey subKey = key.OpenSubKey(subKeyName))
                {
                    string displayName = subKey?.GetValue("DisplayName") as
string;
                    if (!string.IsNullOrEmpty(displayName))
                    {
                        foreach (string app in prohibitedApps)
                        {
                            if (displayName.Contains(app,
StringComparison.OrdinalIgnoreCase))
                            {
                                Console.WriteLine($"Prohibited software
found: {displayName}");
                                return;
                            }
                        }
                    }
                }
            }
        }
        Console.WriteLine("No prohibited software found.");
    }
}
```

---

## 31. Check if Wi-Fi is Disabled

\*\*

**You said:**

continue

**ChatGPT said:**

```

csharp
CopyEdit
using System;
using System.Management;

class WiFiCheck
{
    static void Main()
    {
        using (var searcher = new ManagementObjectSearcher("SELECT * FROM Win32_NetworkAdapter WHERE NetConnectionID != NULL"))
        {
            foreach (ManagementObject obj in searcher.Get())
            {
                string name = obj["Name"].ToString().ToLower();
                if (name.Contains("wi-fi") || name.Contains("wireless"))
                {
                    Console.WriteLine($"Wi-Fi Enabled: {Convert.ToBoolean(obj["NetEnabled"])}");
                    return;
                }
            }
            Console.WriteLine("Wi-Fi Disabled");
        }
    }
}

```

---

## 32. Check if Bluetooth is Disabled

### Explanation:

- **System.Management:** Used for querying system devices via WMI.
- **Win32\_PnpEntity:** Lists installed hardware components.
- **.NET 4.6 Support:** **System.Management** is **not included** by default. Install it offline using:
  1. Open **Visual Studio Installer**.
  2. Select **.NET Framework 4.6 Development Tools**.
  3. Enable **WMI Provider Host** (Windows Management Instrumentation).

```

csharp
CopyEdit
using System;
using System.Management;

class BluetoothCheck
{
    static void Main()
    {
        using (var searcher = new ManagementObjectSearcher("SELECT * FROM Win32_PnpEntity WHERE Name LIKE '%Bluetooth%'"))
        {
            bool isBluetoothEnabled = false;
            foreach (ManagementObject obj in searcher.Get())
            {
                isBluetoothEnabled = true;
                break;
            }
        }
    }
}

```



```
        Console.WriteLine($"Bluetooth Enabled: {isBluetoothEnabled}");  
    }  
}  
}
```

---