```
using System;
using System.Management;

class Program
{
    static void Main()
    {
        try
        {
            // Define the scope to target HP's custom WMI namespace
            ManagementScope scope = new
ManagementScope(@"\\.\root\hp\instrumentedBIOS");
            scope.Connect();

            // Define the WMI query
            ObjectQuery query = new ObjectQuery("SELECT Name, IsSet FROM
HP_BIOSPassword");

            // Execute the query
            using (ManagementObjectSearcher searcher = new
ManagementObjectSearcher(scope, query))
            {
                foreach (ManagementObject obj in searcher.Get())
                {
                    string name = obj["Name"]?.ToString();
                    string isSet = obj["IsSet"]?.ToString();

                    // Only show result for "Setup Password"
                    if (name == "Setup Password")
                    {
                        Console.WriteLine("BIOS Setup Password is " + (isSet == "1"
? "SET" : "NOT SET"));
                        return;
                    }
                }

                Console.WriteLine("Setup Password entry not found.");
            }
        }
        catch (ManagementException ex)
        {
            Console.WriteLine("WMI query failed: " + ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine("Unexpected error: " + ex.Message);
        }
    }
}
```

To know on hp

```
using System;
using System.Management;

namespace BiosPasswordChecker
{
    class Program
    {
        static void Main(string[] args)
        {
            string manufacturer = GetSystemManufacturer();
            Console.WriteLine("Detected Manufacturer: " + manufacturer);

            if (manufacturer != null && manufacturer.ToUpper().Contains("HP"))
            {
                CheckHpBiosPassword();
            }
            else if (manufacturer != null &&
manufacturer.ToUpper().Contains("DELL"))
            {
                CheckDellBiosPassword();
            }
            else if (manufacturer != null &&
manufacturer.ToUpper().Contains("LENOVO"))
            {
                CheckLenovoBiosPassword();
            }
            else
            {
                Console.WriteLine("Unsupported manufacturer or unknown system.");
            }

            Console.WriteLine("\nPress any key to exit...");
            Console.ReadKey();
        }

        static string GetSystemManufacturer()
        {
            try
            {
                using (ManagementObjectSearcher searcher = new
ManagementObjectSearcher("SELECT Manufacturer FROM Win32_ComputerSystem"))
                {
                    foreach (ManagementObject obj in searcher.Get())
                    {
                        return obj["Manufacturer"] != null ?
obj["Manufacturer"].ToString() : "Unknown";
                    }
                }
            }
            catch (Exception ex)
```

```
            {
                Console.WriteLine("Error fetching manufacturer: " + ex.Message);
            }
            return "Unknown";
        }

        static void CheckHpBiosPassword()
        {
            Console.WriteLine("\n[HP BIOS Check]");
            try
            {
                ManagementScope scope = new
ManagementScope(@"\\.\root\hp\instrumentedBIOS");
                scope.Connect();

                ObjectQuery query = new ObjectQuery("SELECT Name, IsSet FROM
HP_BIOSPassword");
                using (ManagementObjectSearcher searcher = new
ManagementObjectSearcher(scope, query))
                {
                    foreach (ManagementObject obj in searcher.Get())
                    {
                        string name = obj["Name"] != null ? obj["Name"].ToString() :
"";
                        string isSet = obj["IsSet"] != null ?
obj["IsSet"].ToString() : "";

                        if (name == "Setup Password")
                        {
                            Console.WriteLine("BIOS Setup Password is " + (isSet ==
"1" ? "SET" : "NOT SET"));
                            return;
                        }
                    }

                    Console.WriteLine("Setup Password entry not found.");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("HP BIOS WMI query failed: " + ex.Message);
            }
        }

        static void CheckDellBiosPassword()
        {
            Console.WriteLine("\n[Dell BIOS Check]");
            try
            {
                ManagementScope scope = new
ManagementScope(@"\\.\root\dcim\sysman");
                scope.Connect();

                ObjectQuery query = new ObjectQuery("SELECT * FROM
DCIM_BIOSPassword");
                using (ManagementObjectSearcher searcher = new
ManagementObjectSearcher(scope, query))
                {
```

```
                    foreach (ManagementObject obj in searcher.Get())
                    {
                        string passwordType = obj["PasswordType"] != null ?
obj["PasswordType"].ToString() : "";
                        string isSet = obj["IsPasswordSet"] != null ?
obj["IsPasswordSet"].ToString() : "";

                        if (passwordType == "Admin")
                        {
                            Console.WriteLine("Dell BIOS Admin Password is " +
(isSet.ToLower() == "true" ? "SET" : "NOT SET"));
                            return;
                        }
                    }

                    Console.WriteLine("Admin Password entry not found.");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Dell BIOS WMI query failed: " + ex.Message);
            }
        }

        static void CheckLenovoBiosPassword()
        {
            Console.WriteLine("\n[Lenovo BIOS Check]");
            try
            {
                ManagementScope scope = new ManagementScope(@"\\.\root\wmi");
                scope.Connect();

                ObjectQuery query = new ObjectQuery("SELECT * FROM
Lenovo_GetBiosSelections");
                using (ManagementObjectSearcher searcher = new
ManagementObjectSearcher(scope, query))
                {
                    bool found = false;
                    foreach (ManagementObject obj in searcher.Get())
                    {
                        string setting = obj["CurrentSetting"] != null ?
obj["CurrentSetting"].ToString() : "";
                        if (setting.ToLower().Contains("password"))
                        {
                            Console.WriteLine("Lenovo BIOS setting: " + setting);
                            found = true;
                        }
                    }

                    if (!found)
                        Console.WriteLine("No password-related settings found.");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Lenovo BIOS WMI query failed: " + ex.Message);
            }
        }
```

```
    }
}
```

1. HP (Hewlett-Packard)

powershell

CopyEdit

```
# HP BIOS Setup Password Status
Get-WmiObject -Namespace root\hp\instrumentedBIOS -Class HP_BIOSPassword |
   Where-Object { $_.Name -eq "Setup Password" } |
   Select-Object -Property Name, IsSet -Unique
```

✅ Output:

pgsql

CopyEdit

```
Name          IsSet
----          -----
Setup Password 0  ← NOT SET (1 = SET)
```

✅ **2. Dell**

Requires **Dell Command | Monitor** to be installed.

powershell

CopyEdit

```
# Dell BIOS Admin Password Status
Get-WmiObject -Namespace root\dcim\sysman -Class DCIM_BIOSPassword |
   Where-Object { $_.PasswordType -eq "Admin" } |
   Select-Object -Property PasswordType, IsPasswordSet -Unique
```

✅ Output:

pgsql

CopyEdit

```
PasswordType IsPasswordSet
------------ -------------
Admin        True  ← SET (False = NOT SET)
```

✅ **3. Lenovo**

Works mainly on **ThinkPad** and **commercial models** with BIOS WMI provider.

powershell

CopyEdit

```
# Lenovo BIOS Password or Lock-related settings (indirect)
Get-WmiObject -Namespace root\wmi -Class Lenovo_GetBiosSelections |
    Where-Object { $_.CurrentSetting -match "password|lock" } |
    Select-Object -Property CurrentSetting -Unique
```

✅ Output (example):

mathematica

CopyEdit

```
CurrentSetting
--------------
Power On Password, Enabled
BIOS Lock, Enabled
```