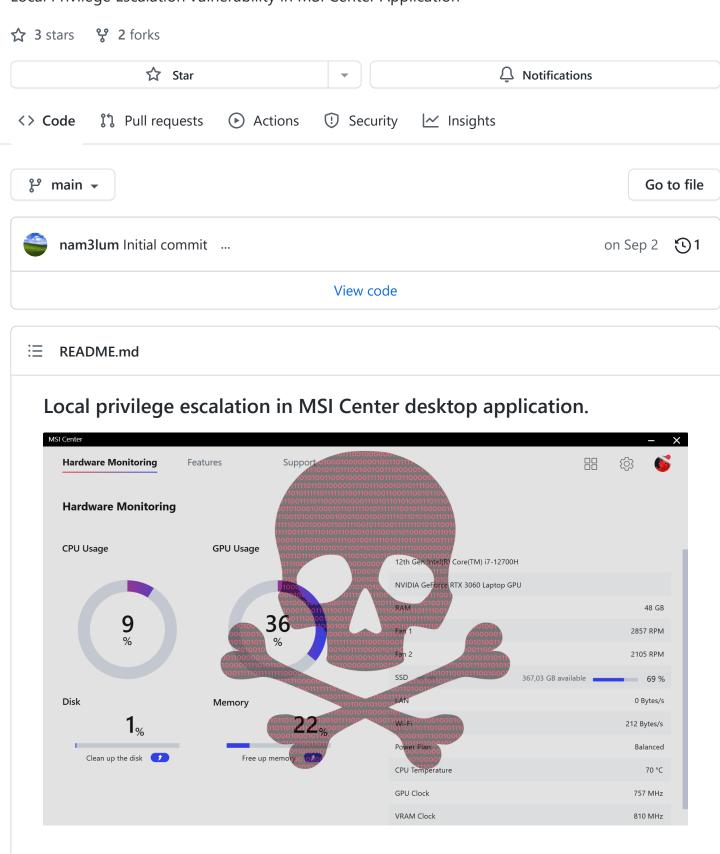


Local Privilege Escalation vulnerability in MSI Center Application



The vulnerability exist in "C\_Features" of MSI.CentralServer.exe. MSI.CentralServer.exe is an application that gathers information about your system, it collaborates with MSI.TerminalServer.exe. The ExecuteTask function which we can call it in "CMD\_AutoUpdateSDK" gives us a chance to run an exectable with custom parameters under Administrative privileges. You can see the related port only from localhost.

```
PS C:\Windows\system32> Get-Process -Id (Get-NetTCPConnection -LocalPort 32682).OwningProcess
Handles NPM(K)
                 PM(K)
                           WS(K)
                                     CPU(s)
                                               Id SI ProcessName
                                                0 0 Idle
     0
           0
                    60
                              8
                                     1,89 13848 4 MSI.CentralServer
   914
           52
                 38740
                           61660
```

## The vulnerability

You can easily disassemble the MSI.CentralServer.exe using any .NET disassembler. Central Server itself listens on 32682 port from localhost, we can find the source code of the handler in "C\_Features". Just look at the CMD\_AutoUpdateSDK feature to see the vulnerability. We abuse this feature (it is automatic updater of MSI Center). It receives the user-given payload, splits it into multiple parts to execute the command with custom parameters.

```
ş- 🚂 🗦 🥦 🗸 🐎 🕞 🔒 🔒
                                                                                              numArray = new byte[1]{ (byte) 1 };
return numArray;
                                                                                            r
if (C API.CompareBytes(RequestData.Data, C Features.CMD AutoUpdateSDK, 0, 8) == 0)

    CS_CommonAPI (3.2022.516.1, msil, .Net Framework v4.6)

                                                                                             byte[] numArray = new byte[RequestData.Data.Length - 8];
Array.Copy((Array) RequestData.Data, 8, (Array) numArray,
string strl = Encoding.UTF8.GetString(numArray);
if (!string.IsNullDrwhiteSpace(strl))
  License (file was re
                                                                                                                                                           numArray, 0, numArray.Length);
 mscorlib (4.0.0.0, x64)
▲ • MSI.CentralServer (3.2022.518.1, msil 32 bit pref, .Net Fra
  ▶ Netadata
                                                                                                 string[] strArray = str1.Split(',');
if (strArray.Length > 1)
  D ■ References
  ▶ ■ Resources
                                                                                                   string str2 = strArray[0];
string RunArguments = strArray[1];
if (!string.IsNullOrWhiteSpace(str2) && File.Exists(str2))

▲ ♦ MSI.CentralServer

    string str3 = Environment.GetEnvironmentVariable("Temp") + "\\MSI Center SDK.exe";
C_Log.Print("Auto update SDK (" + str3 + ").");
    ▶ <sup>♠</sup> C_DynLoad_SDK
  File.Copy(str2, str3, true);

EX_Task.ExecuteTask(str3, RunArguments, SetupType: 2);

return new byte[1]{ (byte) 1 };
    ► SS C_NB_Features
► SS C_WatcherHandler
    atch (Exception ex)
     C_Log.Print("Auto update SDK Error : " + ex.Message);
return new byte[1]{ (byte) 3 };
     ▶ MR Define LU Type
     return new byte[1]{ (byte) 2 };
     ▶ ♦ Define_UpdateDat
                                                                                           if (C_API.CompareBytes(RequestData.Data, C_Features.CMD_Uninstall) == 0)
```

This is main function which our feature uses it to execute given PE with custom arguments:

```
public static int ExecuteTask(
    string RunExePath,
    string RunArguments,
    bool IsSupervisor = true,
    int SetupType = 0)
{
    return EX_Task.ExecuteTask(RunExePath, RunArguments, "", IsSupervisor, SetupType);
}
```

## The port which MSI Central Server listens is updated in 1.0.59.0 version. It is 32683.

## POC

You can generate your own payload, hex it and run the script in the local computer. The POC creates hacker user with "hacker123" password and adds it to the Administrators group.

**Proof-of-Concept video:** https://user-images.githubusercontent.com/64528432/188067866-f30fe089-db76-4cc0-81ce-f74871769b33.mp4

## Languages

• Python 100.0%