







The trouble with Microsoft's Troubleshooters

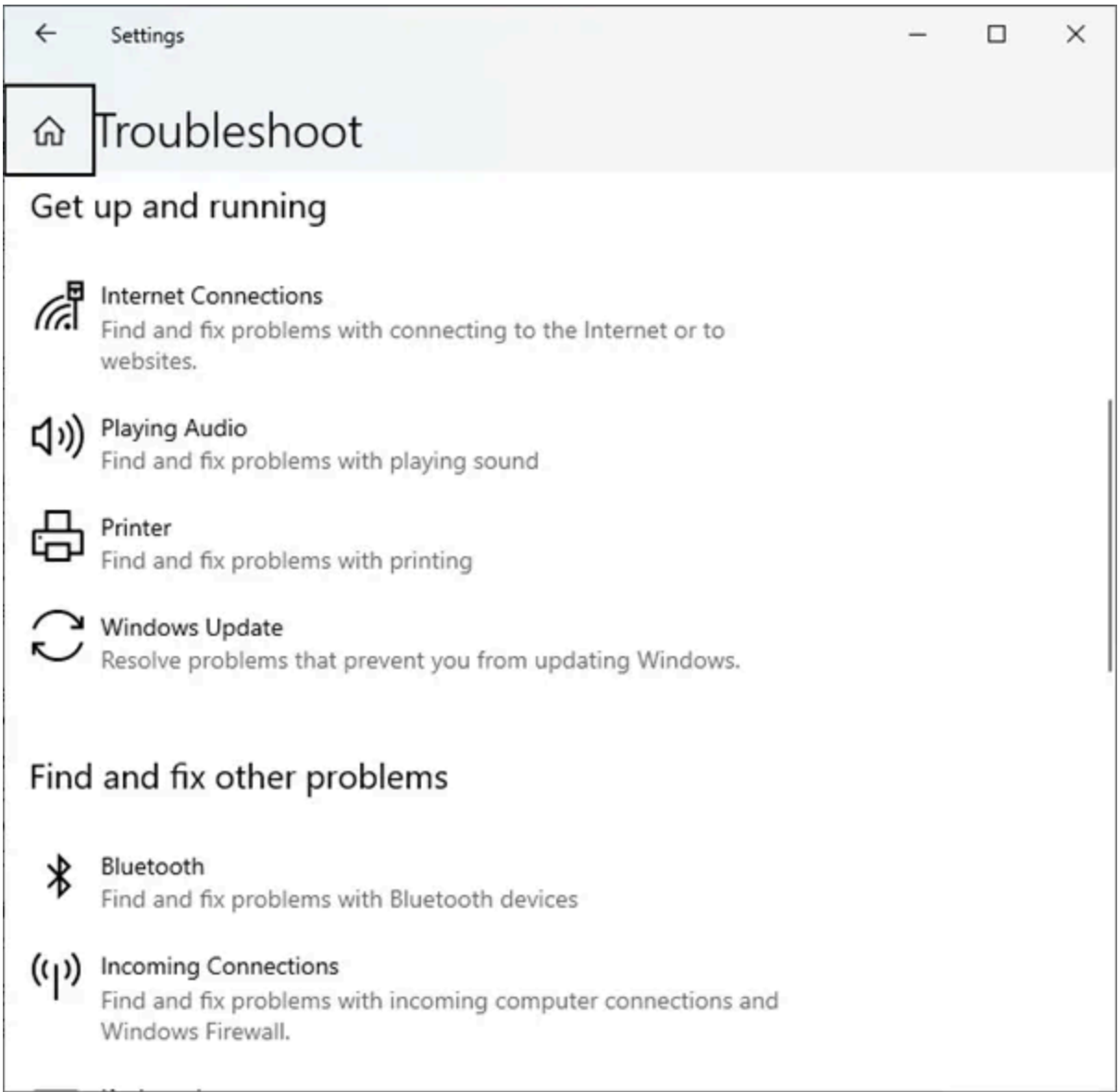
 Imre Rad · Follow
8 min read · Jan 15, 2020

 -- 

TL;DR

Recent versions of **Windows** are equipped with a “Troubleshooting” feature that can be found in the **Control Panel** or the new **Settings** application. You might find this tool useful if you have some trouble with your computer as it can quickly fix some common issues in a user-friendly way. This write up demonstrates an implementation flaw that can be used to compromise a computer where a crafted diagnostic package is opened. Microsoft [edit: initially] pushed back providing a fix for the current versions of **Windows** [edit: but decided two fix it two years later]; to avoid trouble, pay special attention to not open *.diagcab* files. Mail server operators are recommended to include this file type to their blacklists.



The Troubleshoot menu in the Settings app

Vulnerability

Windows is shipped with a bunch of Troubleshooters by default. The technology is flexible so additional diagnostic packages can be downloaded from the internet. Integrity of these packages are protected to avoid external tampering, if signature verification fails, the tool refuses to execute the package.

So where is the problem then? A flaw in the implementation allows attackers to save any files to any locations on the file system (in line with the permissions of the current user) and this takes place before the integrity of the package is checked. Even though overwriting of files is not possible via this vulnerability due to another security measure, an attacker could still gain code execution here by dropping a file to the Startup folder of **Windows**, which will be executed by the Operating System next time when the user logs in.

Threat actors usually distribute malicious files like this in email, via their shady websites or packaged in a torrent file. As in case of every vulnerability, there are several constraints of successful exploitation:

- The email attachments are monitored and scanned by decent email providers — so they could detect and reject messages with attachments like this
- Certain file extensions are blocked by the mail reader or web browser applications. According to Microsoft, **Outlook** and **Internet Explorer** are protected since *.diagcab* is blacklisted by these two softwares if the file is coming from the internet zone.
- User interaction to trigger processing of the diagnostic file by clicking on it
- The user must be using a Microsoft Windows operating system

During my testing, I concluded that neither **Gmail** nor **Outlook Live** blocked *.diagcab* files at all, so users of these services could be potential targets. I encountered the filtering mechanism of some **MS Exchange** based corporate servers blocking my attachments, however, by linking to a webdav share, I could circumvent this protection so the diagcab file could be executed in **Outlook**. But not even links like this can be used ultimately, they are deactivated by providers like **Gmail** or **Outlook Live** and blocked by other security measures of Internet Explorer.

Other popular products like **Google Chrome**, **Mozilla Firefox** or **Thunderbird** simply pop up the standard open and execute dialog box when they see *.diagcab* files. Actually, even **Microsoft Edge**. Using one of these browsers for reading emails on a web based platform is probably the biggest

attack vector of this vulnerability — especially given that users of those platforms are used to the high quality pre-filtering of the emails received.

There is a Man-in-the-Middle attack vector as well; if a *.diagcab* file is downloaded over a clear-text network channel (like via the plain http links found [here](#)), the same attack could be performed by operators of a hostile network (like public Wi-Fi or similar).

The table below summarizes the potential attack vectors that have been tested and found the flaw to be exploitable:

Attack vector	Software	Remark
Email along with a Webdav link sent to a victim using Microsoft Outlook	Microsoft Outlook	No warnings, requires a single click to open the folder then a double click to open the .diagcab file
Email along with direct .diagcab attachment sent to a victim using a desktop email client	Mozilla Thunderbird	Confirmation is needed via standard dialog box
Email along with direct .diagcab attachment sent to a victim accessing emails via a major web based platform (like Gmail or Outlook Live)	Google Chrome, Mozilla Firefox, Microsoft Edge	Confirmation is needed via standard dialog box
.diagcab file hosted on a website controlled by the attacker	Google Chrome, Mozilla Firefox, Microsoft Edge	Confirmation is needed via standard dialog box
.diagcab file distributed on Torrent	any	No warnings, requires a double click

Attack vectors exploiting the .diagcab flaw

Technical details

The software behind the Troubleshooters technology is called **Microsoft Support Diagnostics Tool** and located at `%WINDIR%\System32\msdt.exe`. It is associated to the following file types:

File extensions related to Microsoft Support Diagnostics Tool

Files with *.diagcfg* extension are simple XML files that hold reference to one or more diagnostic packages and provide meta information about them. They are packaged into **Microsoft cabinet** (.cab) file archives and saved with *.diagcab* extension. The XML content itself could look something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<PackageConfiguration
xmlns="http://www.microsoft.com/schemas/dcm/configuration/2008">
  <Execution>
    <Package Path="%windir%\diagnostics\system\Audio">
      <Answers Version="1.0">
```

```
<Interaction ID="IT_GetDeviceType">
  <Value>microphone/headset microphone</Value>
</Interaction>
</Answers>
</Package>
<Name>@%windir%\diagnostics\system\Audio\DiagPackage.dll,-33</Name>
<Description>@%windir%\diagnostics\system\Audio\DiagPackage.dll,-31</Description>
<Icon>@%windir%\diagnostics\system\Audio\DiagPackage.dll,-1003</Icon>
</Execution>
<Index>
<Id>AudioRecordingDiagnostic</Id>
<RequiresAdminPrivileges>true</RequiresAdminPrivileges>
<PrivacyUrl>http://go.microsoft.com/fwlink/?LinkId=190175</PrivacyUrl>
<Version>1.0</Version>
<PublisherName>Microsoft Corporation</PublisherName>
<Category>@%windir%\system32\DiagCpl.dll,-401</Category>
<Keyword>@%windir%\system32\DiagCpl.dll,-24</Keyword>
</Index>
</PackageConfiguration>
```

The Path attribute of the referenced packages point to a directory in the file system. Packages under %WINDIR%\Diagnostics folder are considered to be valid without any further checking, but anything else is subject to signature verification. This is implemented in the helper library *sdiageng.dll*. Before verifying the signature, the implementation makes a local copy of the referenced directory to a temporary random destination folder, something like “C:\Users\John Doe\AppData\Local\Temp\SDIAG_0636db01-fabd-49ed-bd1d-b3fbb5fd0ca”. Note this path has a static number of components which will make the attack deterministic.

The function responsibly for the file transfer is *SdpCopyDirectory* of *sdiageng.dll* and its skeleton code looks something like this:

```
TCHAR attackerControlledSourcePath[MAXPATH]; // this is the "user input"

TCHAR tempDirectory[MAXPATH];
GetTempPathW(MAXPATH, tempDirectory);

TCHAR allFilesFromAttackerControlledSourcePath[MAXPATH];
StringCchPrintfW(allFilesFromAttackerControlledSourcePath,
MAXPATH,
L"%s\\*.*", AttackerControlledSourcePath);
hFind = FindFirstFile(allFilesFromAttackerControlledSourcePath,
&FindFileData);

do {
  TCHAR srcFile[MAXPATH];
  TCHAR dstFile[MAXPATH];
```

```
StringCchPrintfW(srcFile, MAXPATH, L"%s\\%s",
attackerControlledSourcePath, FindFileData.cFileName);
StringCchPrintfW(dstFile, MAXPATH, L"%s\\%s", tempDirectory,
FindFileData.cFileName);

CopyFileW(srcFile, dstFile, TRUE);

while (FindNextFile(hFind, &FindFileData) != 0);FindClose(hFind);
```

After the copy is done, the *dll* verifies the integrity based on the file *DiagPackage.cat*. If everything is correct and the user proceeds with the wizard on the GUI, the embedded powershell scripts are executed under the hood.

Here comes the trick: since the data source is controlled by the attacker and network file systems are supported by the Windows OS natively, this pattern can be exploited before the integrity check had chance to detect the malformed package. A rogue, network attached file system can gain controlled write over the destination file system by returning *..|..* components in the file name.

For proof of concept, I chose to spin up a webdav server of my own. Webdav is a firewall friendly, lightweight protocol that is easy to script, but I believe the same attack could be mounted with CIFS as well. The UNC (Universal Naming Convention) path of a webdav share looks something like this: *\\localhost@80\\DavWWWRoot*. An example session of interaction with such a malicious webdav server from the command prompt:

```
C:\Projects\diagcab>dir \\127.0.0.1@80\\DavWWWRoot\package
Volume in drive \\127.0.0.1@80\\DavWWWRoot has no label.
Volume Serial Number is 0000-0000

Directory of \\127.0.0.1@80\\DavWWWRoot\package
2017. 07. 12.  11:10    <DIR>          .
2017. 07. 12.  11:10    <DIR>          ..

2017. 07. 12.  10:48                27 648
..\..\..\..\AppData\Roaming\Microsoft\Windows\Start
Menu\Programs\Startup\calc.exe

1 File(s)                27 648 bytes
2 Dir(s)  251 292 504 064 bytes free
```

Putting this altogether, the crafted *.diagcab* file has the package path set to the rogue webdav server. Once it is opened by the victim, a new file is saved under the *Startup* directory, and is executed by the operating system on the next startup.

Exploitation of the vulnerability described above was tested on *Windows 10 version 1903 (OS Build 18362.535)*, probably other versions are affected as well.

Remediation

I believe the correct remediation at MS side would be discarding file system entries that have a path separator character in their name. This could be done in the implementation of the FindFirstFile/FindNextFile API calls, which reside in kernel32.dll, so in the kernel space. This would protect all applications having a similar vulnerable pattern in their code, no matter which file system backend the nasty file system entries were returned by, providing sufficient protection for both local and remote file systems.

Since this won't happen for a while, I don't recommend opening .diagcab files anymore. Mail system administrators are advised to tweak the blacklist in their configurations.

Demo

Diagcab files and a rogue webdav PoC server is hosted online for demonstrational purposes. (This latter will go down once my free dyno hours are exhausted at Heroku.)

The exploit in work

If you wish, you can test the same via this [webpage hosted on Github.io](#).

You can inspect the webdav backend server by running the following command:

```
dir \\webdav-test.herokuapp.com@SSL\DavWWWRoot
```

If you execute the diagcab file hosted here, it will configure your Windows to launch the calculator at login. To revert the original status of your computer press CTRL+R, type shell:startup and remove the calc.exe file from the folder.

Sources of the webdav server exploit can be found on Github:

<https://github.com/irsl/microsoft-diagcab-rce-poc>

Timeline

12/22/2019: issue reported

12/23/2019: case is opened

6/1/2020: Microsoft responded to not fix the issue

9/1/2020: Draft of this write up shared with Microsoft as they requested

15/1/2020: Public advisory

7/x/2022: Microsoft block-listed .diagcab in Edge

7/29/2022: Google block-listed .diagcab in Chrome (CVE-2022-2622)

8/4/2022: Microsoft have reassessed the case “This email may come as a bit of a surprise, but this is a follow up to your case MSRC 55532 which was submitted back in 2019. We have reassessed the issue per our updated Windows bug bar (<https://aka.ms/windowsbugbar>) and determined that this issue meets our criteria for servicing with a security update.” (CVE-2022-34713)

Microsoft’s response

MS: Microsoft has decided that it will not be fixing this vulnerability in the current version and we are closing this case. ...

IR: Clicking on a non-executable file driven by an official Microsoft technology could save attacker controlled content to any attacker controlled location on the file system including shell:startup and thus accomplishing code execution. Please confirm Microsoft does not consider this being a vulnerability. ...

MS: There are a number of file types that can execute code in such a way but aren’t technically “executables”. And a number of these are considered unsafe for users to download/receive in email, even .diagcab is blocked by default in Outlook on the web and other places. This is noted a number of places online by Microsoft.

The issue is that to make use of this attack an attacker needs to create what amounts to a virus, convince a user to download the virus, and then run it. Yes, it doesn't end in .exe, but these days most viruses don't. Some protections are already put into place, such as standard files extensions to be blocked, of which this is one. We are also always seeking to improve these protections. But as written this wouldn't be considered a vulnerability. No security boundaries are being bypassed, the PoC doesn't escalate permissions in any way, or do anything the user couldn't do already.

Poll

Please vote

Credits

Imre Rad

- Security
- Cybersecurity
- Microsoft
- Vulnerability
- Webdav

--



Written by Imre Rad

104 Followers

Software developer daytime, security researcher in freetime

Follow

