




SANS Network Security: Las Vegas Sept 4-9.

 Handler on Duty: Guy Bruneau

Threat Level: Green

 Homepage

 Diaries


 Podcasts

 Jobs

 Data

 Tools


 Contact Us

 About Us

 Slack Channel

 Mastodon

 Bluesky

 X

previous

next

Desktop.ini as a post-exploitation tool

Published: 2020-03-16. Last Updated: 2020-03-16 11:15:21 UTC
by [Jan Kopriva](#) (Version: 1)



[2 comment\(s\)](#)

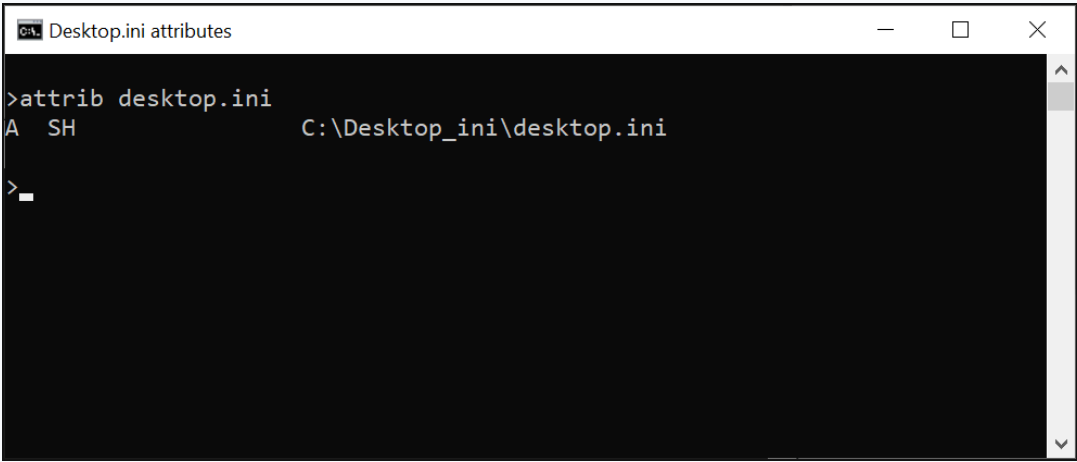
Desktop.ini files have been part of Windows operating systems for a long time. They provide users with the option to customize the appearance of specific folders in File Explorer, such as changing their icons^[1]. That is not all they are good for, however.

Couple of months back, I noticed a small weakness/vulnerability in the way desktop.ini files are interpreted by File Explorer, which may be used to hide files and folders from a user without a need to delete them or to substitute different file/folder for a valid one. As you can probably imagine, this might lead to users executing unintended code or reading from/writing to unintended folders or files and could therefore be quite useful for certain red teaming activities or malicious actions (as well as for pulling pranks on co-workers, but we'll leave that aside).

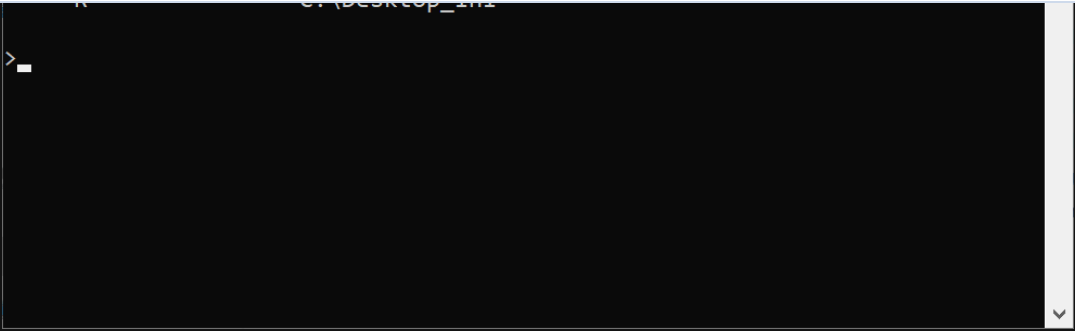
It is possible I wasn't the first one to notice the issue, but since I didn't find any write-ups of it anywhere else (and since Microsoft informed me that they will not be fixing the vulnerability in the current version of the OS and don't mind if I publish it), I thought I'd share it here.

UPDATE: It turns out that Adam, aka [@Hexacorn](#), did a write-up of the folder re-naming capability of desktop.ini all the way back in 2012. You may find it [here](#).

Before we get into the details of how it may be exploited, let's take a quick look at the conditions under which desktop.ini files are interpreted by File Explorer. When a desktop.ini file is created by the operating system, it has the Hidden (H) and System (S) attributes set. You may check this by having Windows create one (e.g. by going to Properties of a folder and changing its icon on the Customize tab) and then looking at its attributes using the attrib command.



Although these attributes are customary for desktop.ini files, they are not necessary in order for them to be interpreted by File Explorer. In fact, the file itself doesn't have to have any special attributes set, but the folder containing the desktop.ini file needs to be Read-only (i.e. the "R" attribute).



Besides that, of course, the desktop.ini file needs to have valid contents conforming to the INI format[2]. The following example shows a correctly formatted desktop.ini.

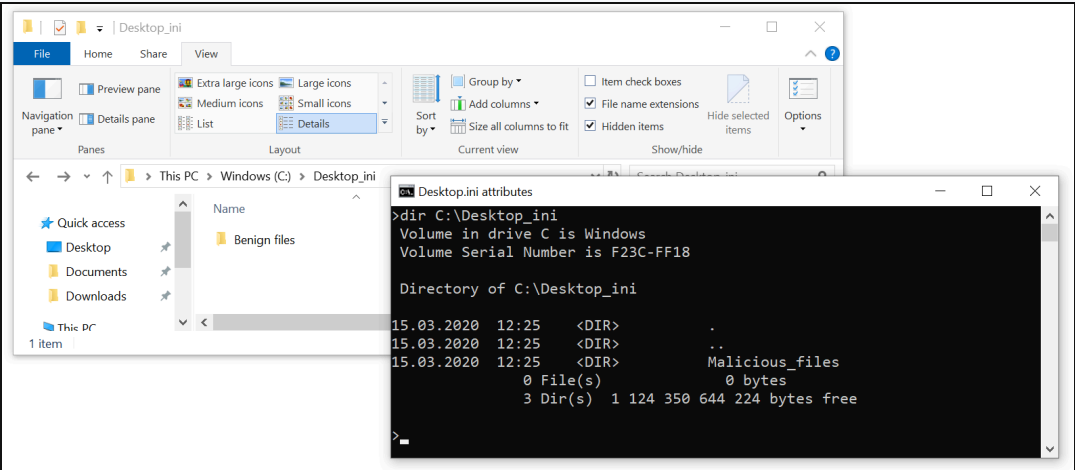
```
[.ShellClassInfo]
IconResource=C:\WINDOWS\System32\SHELL32.dll,8
```

So where is the vulnerability?

It lies in the fact that an icon of a folder isn’t the only thing which may be changed using desktop.ini. The name of the folder, as well as the names of all files it contains, may seemingly be changed as well. This may be done with the use of “LocalizedResourceName” key and the “LocalizedFileNames” section. If these are present in a desktop.ini file, the File Explorer (other file managers will still display files normally) will display names provided by them instead of the real ones.

The name of a folder may be changed by setting LocalizedResourceName in its desktop.ini file in the following way.

```
[.ShellClassInfo]
LocalizedResourceName=Benign files
```



As you may see, our folder “Malicious files” becomes “Benign files”. Using this mechanism, it is even possible to create multiple folders with the (seemingly) same name in the same path (more on that later).

Changing of file names using desktop.ini is similar – one only needs to use the LocalizedFileNames section, as the following example shows.

```
[LocalizedFileNames]
malicious_file.exe=bening_file
benign_file.exe=unimportant_file
```



Homepage

Diaries

Podcasts

Jobs

Data

Tools

Contact Us

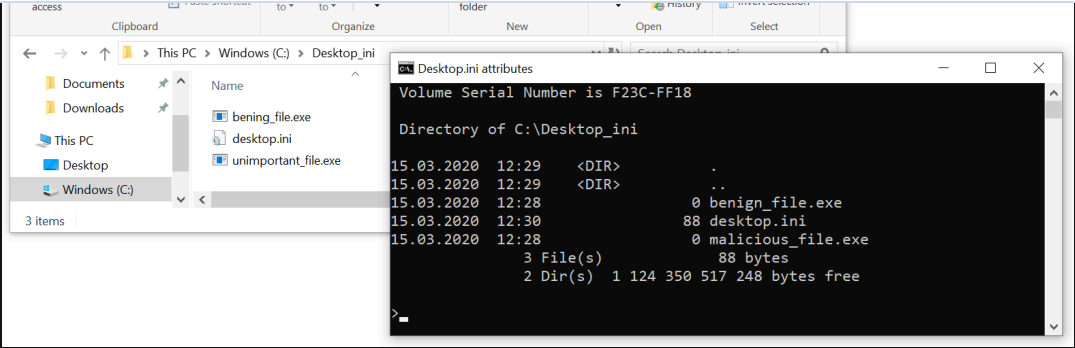
About Us

Slack Channel

Mastodon

Bluesky

X



As you may see, the *malicious_file.exe* is now displayed as *benign_file.exe* (at least in File Explorer), and the original *benign_file.exe* was seemingly re-named to *unimportant_file.exe*. If you’d like to try this out on your own system, you may download an archive containing the folder used in the example from https://untrustednetwork.net/files/ISC/2020/Desktop_ini.zip.

Since almost any string may be used for the “fake” file or folder name, there is quite a large potential for misuse of this behavior of File Explorer, as you can probably imagine.

Although a hypothetical attacker would have to have high enough privileges to write desktop.ini to any folder he would wish to use (and potentially to set the folder to be “read-only”), the fact that the behavior is the same on local drives as on remote shares makes the above-described technique viable for the post-exploitation phase of a red team engagement.

Two main scenarios for use of the vulnerability come to mind:

1. A red teamer/malicious actor could create a folder named “abc” and place it on a file system in the same path, where a folder, which is inaccessible to him and into which other users write confidential information (e.g. a “Salaries” folder), is placed. The malicious actor could then set the folder “abc” with a “read-only” attribute and place in it a desktop.ini with the following contents:

```
[.ShellClassInfo]
LocalizedResourceName=Salaries
```

This would result in two folders named “Salaries” seemingly existing in the same path and since File Explorer usually displays folders alphabetically, the fake “Salaries” folder would be displayed first (as it’s name really starts with “a”). A user could then easily make the mistake of saving a file into the fake “Salaries” folder instead of the legitimate one.

2. A red teamer/malicious actor could substitute one file for another and cause a legitimate user to open/modify/execute/delete the wrong one. An example might be substituting one executable (malicious) for another (benign), as was shown in one of the previous examples.

It should be noted that none of the above-mentioned activities would result in any detectable events in terms of re-naming of the files or folders themselves, given that no “re-naming” actually takes place. On the other hand, since new desktop.ini files are only seldom created, especially on network files shares, one potential detection mechanism for attempted use of these techniques could be to simply monitor any newly created files named desktop.ini on any sensitive file system.

I should add that I tested the behavior only on Windows 10 and Windows 7, but I wouldn’t be surprised if other Windows OSs behaved in the same manner.

As you may see, desktop.ini files may really be used for much more than just changing a folder icon. And although the techniques described above are mainly useful for (if anything) red teaming, either when interacting with a shared-computer environment or with network file shares, it is definitely good to be aware of them, whether one leans more toward the blue or red side of the infosec spectrum.



Homepage

Diaries

Podcasts

Jobs

Data

Tools

Contact Us

About Us

Slack Channel

Mastodon

Bluesky

X

[2] https://en.wikipedia.org/wiki/INI_file#Format

Jan Kopriva

@jk0pr

Alef Nula

Keywords: Vulnerability Windows

2 comment(s)

previous

next

Comments

> but the folder containing the desktop.ini file needs to be Read-only (i.e. the “R” attribute).

Huh? If the folder is tagged as "read-only", then all the files inside the folder inherit the "read-only" attribute.

Your example shows that the file, itself, has the "R" attribute.
Or, maybe your example shows that you have a folder named "desktop.ini" ???

Anonymous
Mar 18th 2020
4 years ago

The folder is named "Desktop_ini" - in hindsight, I probably should have named it differently...

Anonymous
Mar 18th 2020
4 years ago

Login here to join the discussion.

Top of page

Diary Archives

