PERSISTENCE USING GLOBALFLAGS IN IMAGE FILE EXECUTION OPTIONS – HIDDEN FROM AUTORUNS.EXE

Posted on 10 Apr 2018

TL;DR

- Found a technique to execute any binary file after another application is closed without being detected by Autoruns.exe.
- Requires administrator rights and does not belong in userland.
- Can also be executed from alternate data streams
- Plant file on disk and run these commands to create persistence that triggers everytime someone closes notepad.exe:

```
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution O
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SilentProcessExit\note
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SilentProcessExit\note
```

Image File Execution Options

Another day with some unstructered research time. I must admit that it feels good every time. 🙄

Last time I found a way to execute DLL files and still hide from Autoruns.exe. This time I found some interesting stuff, that I have not found that much related information on and hopefully it will help people detect someone if they are using this technique. This adventure started out when I was looking for other ways to execute data from alternate streams. Somehow I ended up in Process monitor (big surprise) and started looking at the Image File Execution Options. Normally I would just pass by these, since I always assume that someone has probably already discovered all there is to discover. Again it turns out that assumptions is the mother of all fu**ups.

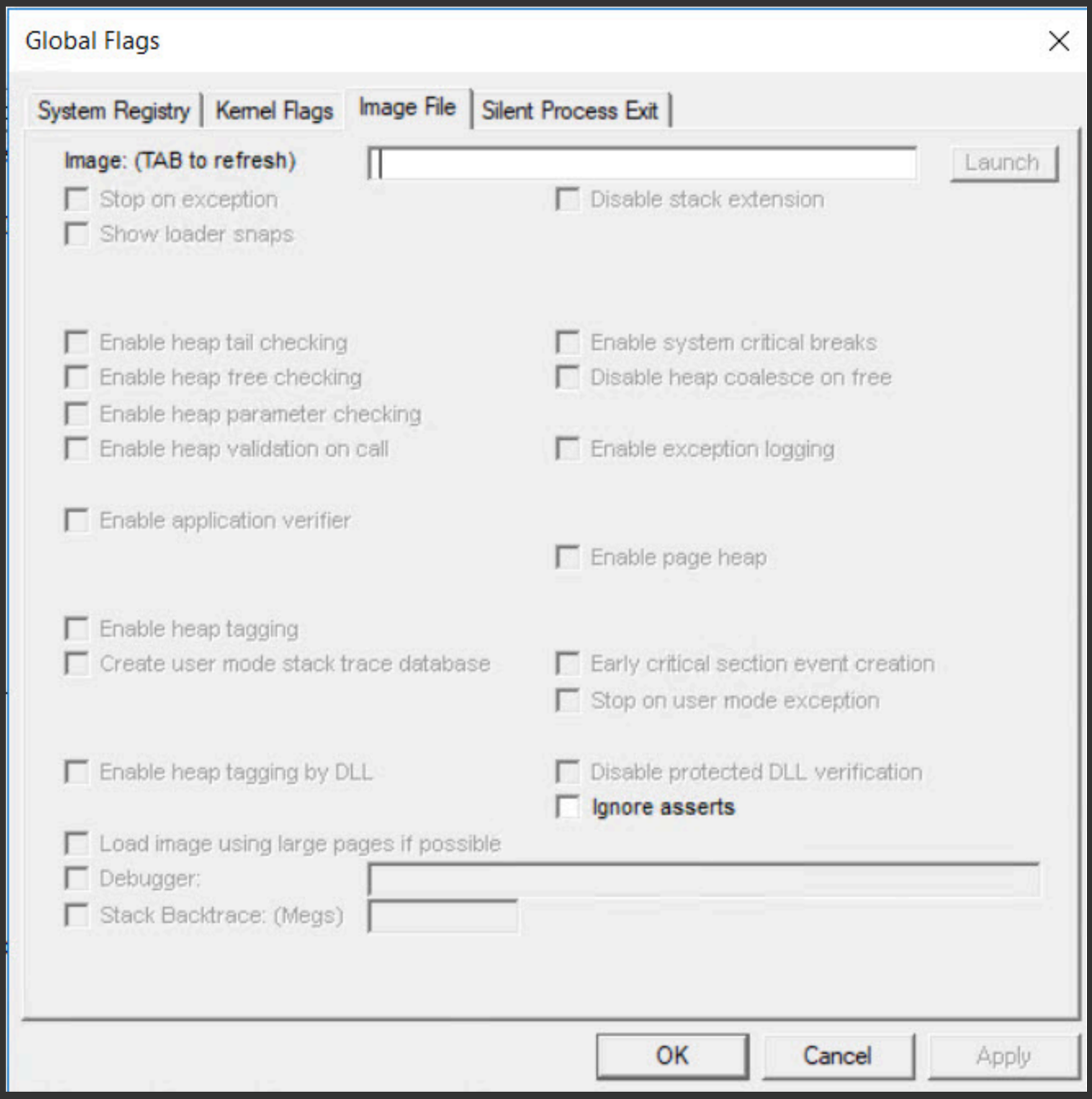
I started by Googling for information about the Image File Execution Options and especially the *ApplicationGoo* setting and I ended up here: https://blogs.msdn.microsoft.com/junfeng/2004/04/28/image-file-execution-options/

This blogpost also mentioned GlobalFlags and that cought my eye. After I was done Googling and searching for ApplicationGoo and what it did, I stumbled upon this and it turned out that you can add the ApplicationGoo in a special way to fake what operating system you are running to a process. I am not done researching the ApplicationGoo, so feel free to go on your own adventure. 😉

I returned to read some more details about the GlobalFlags, since that was more interesting. The MSDN blog stated the following (Thanks Microsoft):

> If you play with gflags.exe more, you will found more interesting registry values under Image File Execution Options.

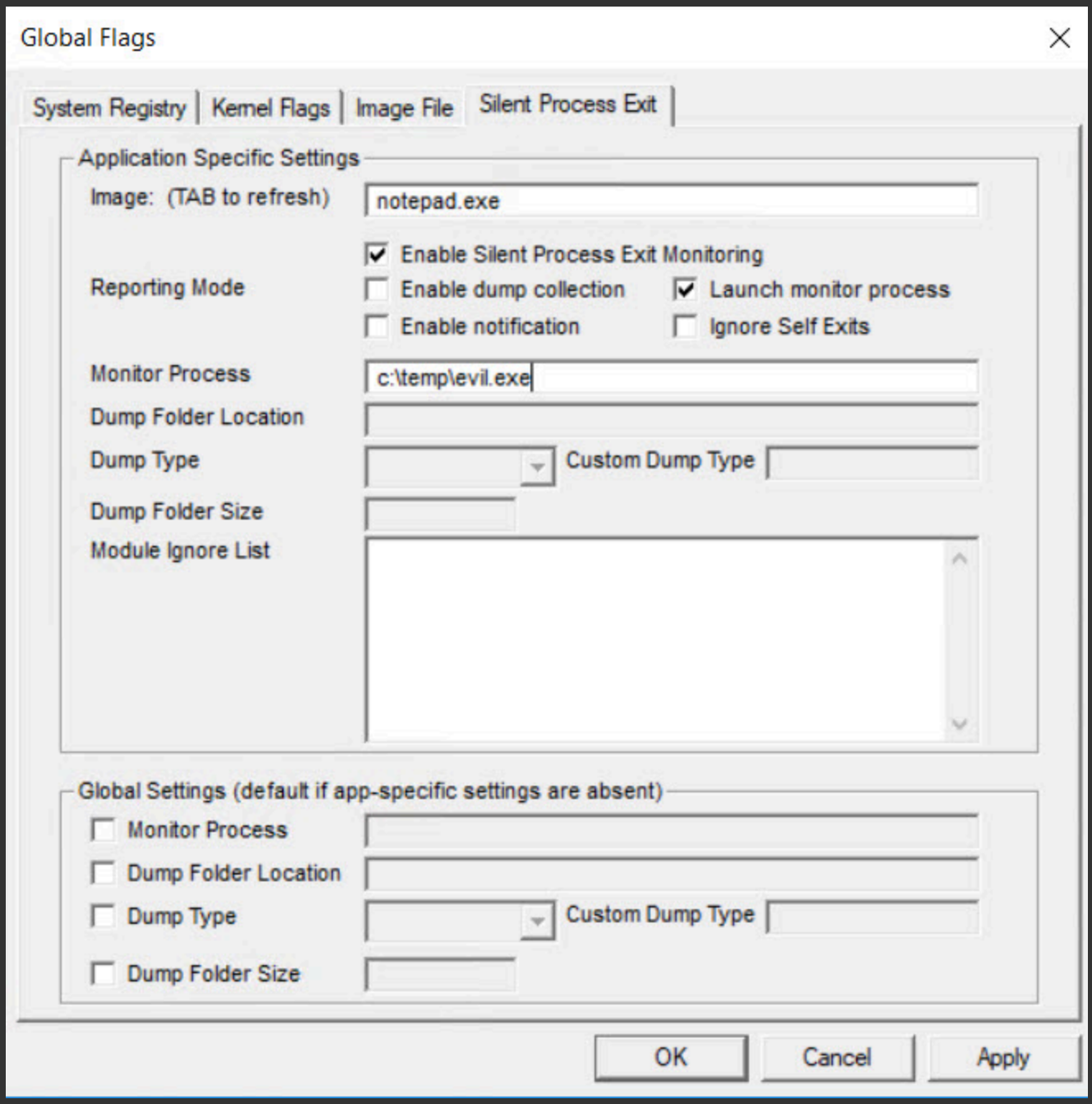
A quick search for gflags.exe and I found that this is a part of the Windows 10 SDK, and this binary was already present on my machine. I fired up the application and it looks like this:



This application can be used to change all the flags related to the execution of a binary. Here could also be more interesting stuff to dig into that I have not looked at yet.

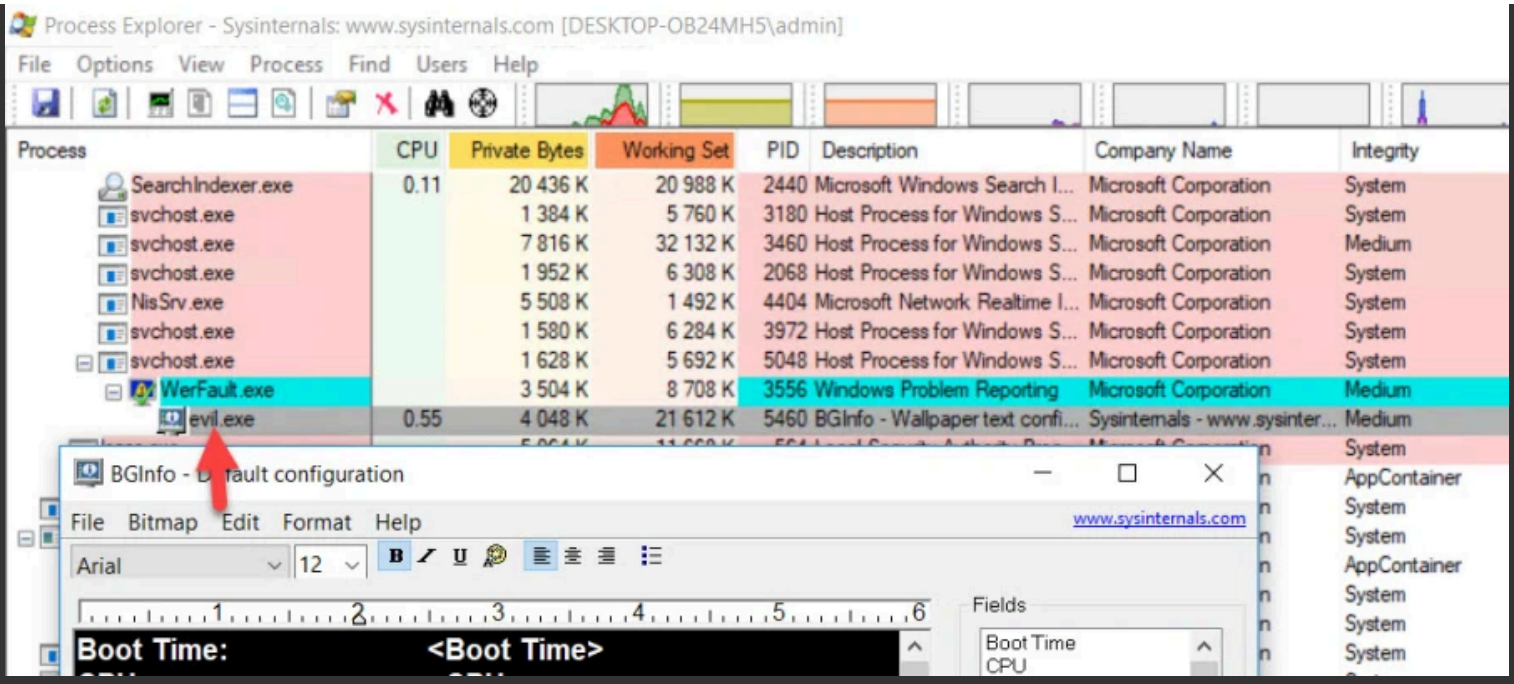
The first thing I tried was to check if this Application could work as a Device Guard bypass by leveraging the Launch command. This turned out to be negative. Based on my previous experience I

already knew what the debugger flag does so I did not care about that. What I however found out was that under the "Silent Process Exit" tab there was a lot of other interesting stuff to look at. 😈



As you can see, my evil plan here is to execute an evil binary every time notepad.exe is closed. After planting this I verified that it worked by running just a renamed version of bginfo.exe. The point here is not the payload I am running, more the technique.

After I close notepad.exe evil.exe is spawned like this:



So this was pretty awesome I thought. It also turns out that autoruns.exe does not detect this technique. (Sorry Mark, even more to do with autoruns.exe)

After a bit more reversing I also figured out that the registry keys that decides what to launch as a silent "monitor" resides in "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\SilentProcessExit\"

All that gflags.exe does is actually only write the registry keys necessary. To achieve the same with some simple commands you could simply run the following lines in cmd.

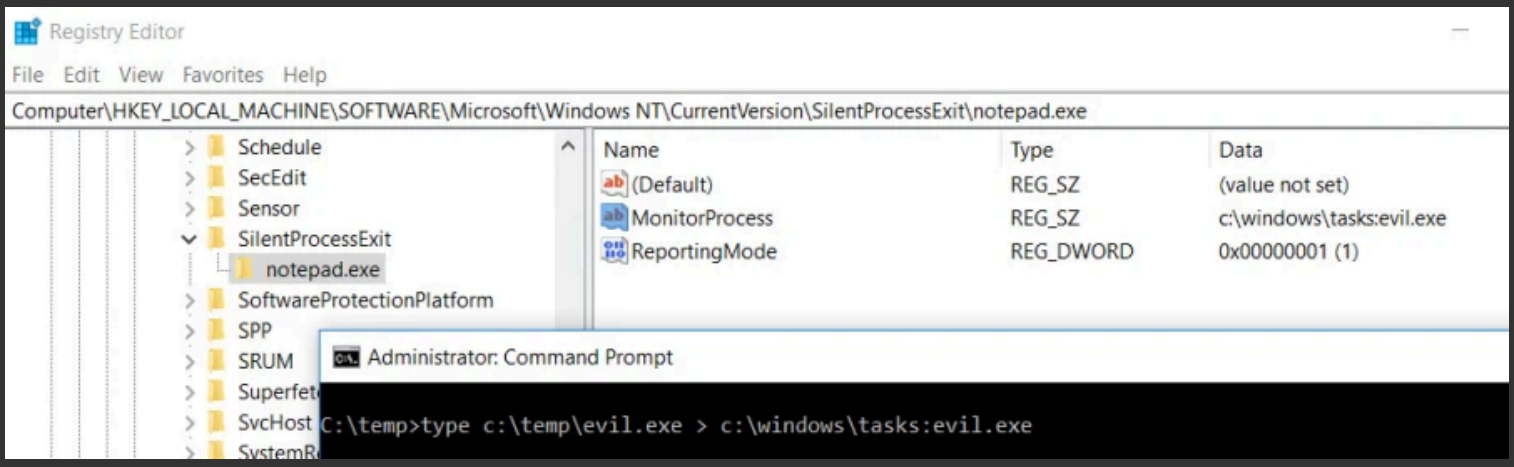
```
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution O
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SilentProcessExit\note
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SilentProcessExit\note
```

This is also pretty good documented at docs.microsoft.com.

BONUS – Execute with Alternate data streams

Also figured out that you can leverage alternate data streams as well. That means you can take the evil.exe and add it to for instance the tasks folder under C:\windows\ as an alternate stream. That can easily be done by changing the registry and using this command:

type c:\temp\evil.exe > c:\windows\tasks:evil.exe



After I close notepad, it now looks like this:

I've got asked by some people since my last post on why I disclose these things, and my attentions are pure. Many people fear that this is like giving away techniques to the bad guys, but I feel disclosing these things makes is possible to discover them in the wild and create good detection mechanisms and prevention. I have also seen a lot of discussions on Twitter lately about people not wanting to disclose their techniques since it makes their job more difficult (pentesters) and that makes me sad in some way even though I can understand and relate to the reasons.

My reasons for sharing things I discover is to make things more secure for everyone and hopefully it will also inspire others to start their own research and disclose new and unknown stuff to the public. Hope you enjoyed the post and as always, feedback is always welcome!

PREVIOUS POST
Persistence using RunOnceEx – Hidden from Autoruns.exe

NEXT POST
Putting data in Alternate data streams and how to execute it – part 2

12 THOUGHTS ON "PERSISTENCE USING GLOBALFLAGS IN IMAGE FILE EXECUTION OPTIONS – HIDDEN FROM AUTORUNS.EXE"

Dan says:
11 Apr 2018 at 1:16 pm
Interesting article. There is an additional method for persistence that I haven't seen anyone check for which also requires admin rights. The initial functionality is for process debugging and is enabled in a similar fashion. It basically allow you to run something prior the "debugged" process starting with the purpose of debugging attachment. I remember reading about the functionality at some point in a book and wondering why nobody uses this as a persistence method. I know I tested it and it worked like a charm.

I don't rememeber the details but I do believe it was something similar to:
https://support.microsoft.com/en-us/help/824344/how-to-debug-windows-services

Section:
Configure a service to start with the WinDbg debugger attached
method 2

Cheers!

★ Like
Reply

★ **Oddvar Moe [MVP]** says:
11 Apr 2018 at 2:23 pm
I think you are thinking about debugger that you can set in registry. This has been known for many years and malware use it all the time. Even process explorer uses this technique if you choose to change it to the default task manager.

★ Liked by 1 person
Reply
Pingback: Week 15 – 2018 – This Week In 4n6

Stratcat says:
27 Dec 2018 at 1:41 pm
This has been around for a long time and any developer worth a damn knows about it. I think you're safe in sharing and linking to the MSDN documents that describe its usage.

★ Like
Reply

Pingback: Persistence – Image File Execution Options Injection | Penetration Testing Lab

Pingback: RED TEAMING_Final Att&ck – B4cKD00₹

Pingback: 2 Crucial Registry Keys For DFIR – Intarna

Pingback: Sensitive Command Token - Bug Bounty - My Blog

Pingback: Sensitive Command Token – So much offense in my defense – Thinkst Thoughts

Pingback: Persistence Mechanisms – Cyber Elias Tube – Cyber Elias Tube

Pingback: Persistence Mechanisms | Cyber Elias Tube

Pingback: Monitoring Sensitive Windows Commands via CanaryTokens - Deploying Registry Entries via Group Policy | Cyber Elias Tube

LEAVE A COMMENT