International Journal of Computer Applications (0975 – 8887)
Volume 171 – No. 6, August 2017

Design and Implement a Hidden Processes Detector (HPD) based on Windows Prefetch Files

Zaid Abdulelah Mundher
Department of Computers Sciences
College of Computer Sciences and Mathematics
University of Mosul
Mosul, Iraq

ABSTRACT

Hidden processes threat, which is a technique that is used by malicious code to hide their activities, is a serious threat to the operating systems. Therefore, the security programs try to defeat this threat using different approaches. This paper presents a hidden processes detector (HPD) program to detect hidden processes on Windows-based systems. The proposed HPD program introduces a new approach based on the Windows Prefetch files. The proposed HPD program has been tested and the results have been mentioned in this paper.

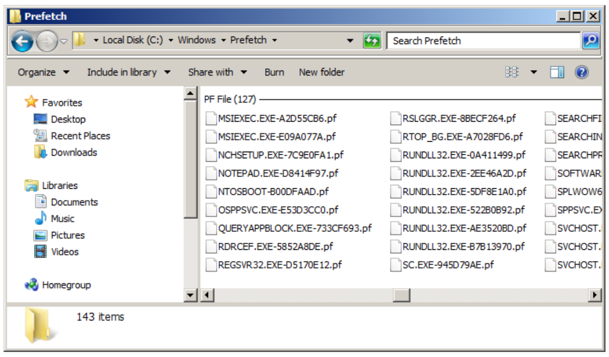
Keywords

Hidden-process, Windows Prefetch files, Rootkit.

1. INTRODUCTION

To improve performance, for each program that runs on Windows based systems (started from Windows XP), a Prefetch file is created or updated in the Prefetch folder which is found under C:\Windows directory [1][2]. Each file in the Prefetch folder includes information about the program that the file refers to. This information includes the name of the application, the DLL and files that are used by the application, the last run time of the application, and the total number of times that the application has been launched [13]. The name of the Prefetch files consists of the executable file name, followed by a dash, and then an eight character hash of the program's start location, with a ".pf" extension [3]. For example, the Prefetch file for notepad.exe program would be NOTEPAD.EXE-D8414F97.pf, where D8414F97 is a hash of the path of the notepad.exe file.

Prefetch directly is shown in Fig. 1.



```
PF File (127)
MSIEXEC.EXE-A2D55CB6.pf          RSLGGR.EXE-8BECF264.pf          SEARCHFI
MSIEXEC.EXE-E09A077A.pf          RTOP_BG.EXE-A7028FD6.pf         SEARCHIN
NCHSETUP.EXE-7C9E0FA1.pf         RUNDLL32.EXE-0A411499.pf        SEARCHPF
NOTEPAD.EXE-D8414F97.pf          RUNDLL32.EXE-2EE46A2D.pf        SOFTWAR
NTOSBOOT-B00DFAAD.pf             RUNDLL32.EXE-5DF8E1A0.pf        SPLWOW6
OSPPSVC.EXE-E53D3CC0.pf          RUNDLL32.EXE-522B0B92.pf        SPPSVC.E
QUERYAPPBLOCK.EXE-733CF693.pf    RUNDLL32.EXE-AE3520BD.pf        SVCHOST.
RDRCEF.EXE-5852A8DE.pf           RUNDLL32.EXE-B7B13970.pf        SVCHOST.
REGSVR32.EXE-D5170E12.pf         SC.EXE-945D79AE.pf              SVCHOST.
```

Fig.1: Prefetch Directory

On the other side, hide processes is a technique which is used by malicious software such as rootkits and backdoors, so they can run in background while anti-viruses and security programs are unware of [4]. There are different methods that could be used to hide processes, and these methods may be implemented either in user-mode or kernel-mode. Inline function hooking and Import Address Table (IAT) redirection are examples of user-mode hooking techniques. Moreover, to implement a user-hooking technique, a code injection method must be implemented. Windows message hooking and create remote thread are examples of code injection technique. On the other hade, a device driver must be implemented to use the kernel-mode hooking type. Modifying the System Service Descriptor Table (SSDT) is an example of kernel-mode hooking technique.

The major contribution of this work is to address the hidden processes detecting issue by introducing a new method that uses the information that could be extracted from the Prefetch files to detect hidden processes.

2. LITERATURE REVIEW

There are different ways to detect hidden processes. Traditionally, most of these ways are based on hooking techniques and kernel memory scanning [5][6][7]. There are already some academic works that try to detect hidden processes. In [8], the author describes some of these methods. In [4], the author introduced a new vm-based approach to detect hidden processes. The author of [9] introduced a new technique that depends on monitoring the parts most often modified by kernel-mode rootkit.

Each of these methods has its own advantages and disadvantages, but, in general, all of these techniques are complex and hard to implement. Moreover, by using one of these methods, find the path of the hidden processes is difficult [10]. The aim of this work is to introduce a simple yet efficient method to detect hidden processes on Windows-based systems. To the best of the researcher's knowledge, the proposed approach has never been introduced before.

3. PROPOSED METHODOLOGY

As was mentioned before, for each program that runs, there will be an entry in the C:\Windows\Prefetch folder [11]. In other words, by analyzing the Prefetch file data, the HPD program could determine if a program is run or not. Since that, the malicious software such as Rootkits and malwares, are programs as well, so Prefetch files will be created for them. The proposed HPD program finds names of the all processes that listed in the Prefetch folder. To extract the process's name from the Prefetch file's name, some string manipulations are needed to be done. The list of discovered processes is compared with a normal processes list that is gotten using standard enumeration functions. If a process has a Prefetch file, while it is not appearing using standard enumeration functions, this process is probably a hidden process. When the proposed HPD program detects a hidden process, it will take two actions: First, it adds the detected hidden process to the hidden-processes-list. Second, it searches inside Layout.ini file to get the full path of the detected hidden process. An overview of the proposed HPD program is shown in Fig.2

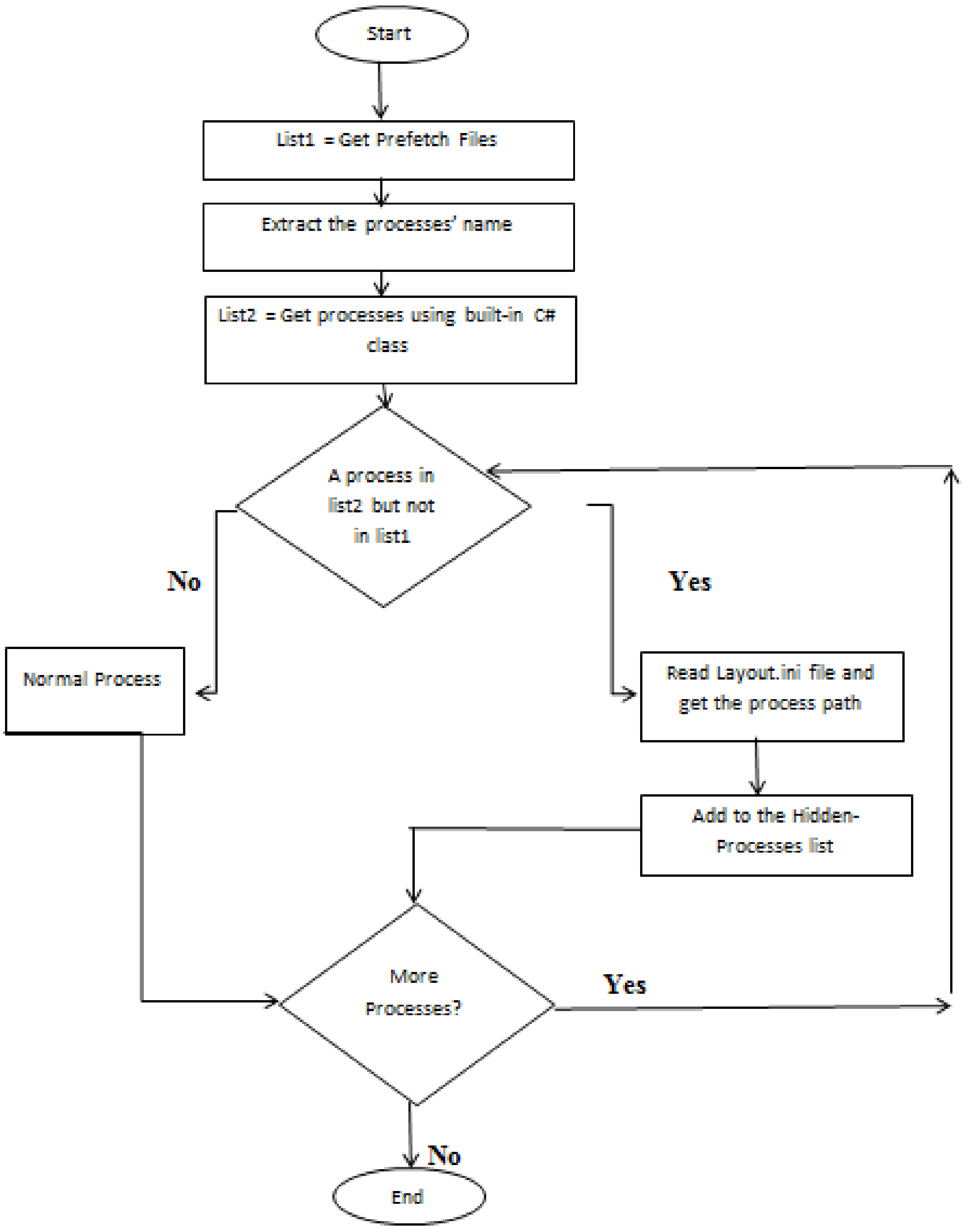

Start
List1 = Get Prefetch Files
Extract the processes' name
List2 = Get processes using built-in C# class
A process in list2 but not in list1
No
Yes
Normal Process
Read Layout.ini file and get the process path
Add to the Hidden-Processes list
More Processes?
Yes
No
End

Fig.2: An overview of the proposed system

4. IMPLEMENTATION AND RESULTS

C# language was used to develop the HPD program. The main window of the HPD is shown in Fig 3.

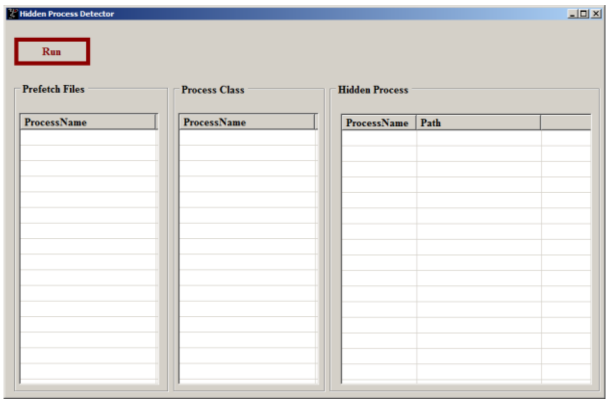
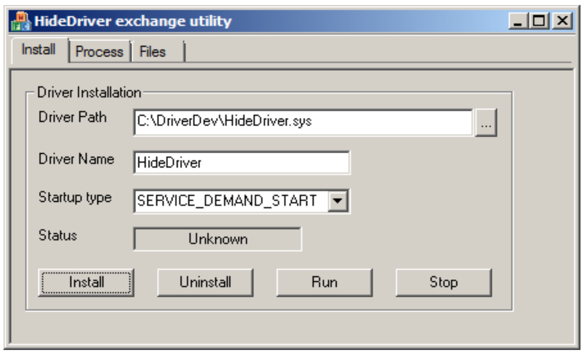


Fig.3: The Proposed HPD's window

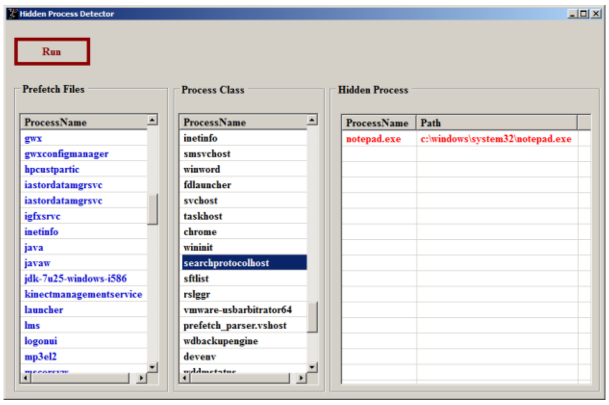
To test the activity of the proposed HPD program, a hidden process tool was used. The HideDriver exchange utility (Fig. 4), which is "a free, lightweight and portable process hiding and file hiding software developed by Sergey Popenko, it allows to hide any processes from almost any process monitors/viewers including Windows task manager."[12].



Driver Path C:\DriverDev\HideDriver.sys
Driver Name HideDriver
Startup type SERVICE_DEMAND_START
Status Unknown

Fig. 4: The HideDriver Exchange Utility

The test is performed in three phases. First, the HideDriver exchange utility is used to hide a selected process (notepad.exe was chosen to achieve the test). Then, to ensure that the tool works correctly, the Task Manager was run, and the results have showed the process (notepad.exe) was not appear in the processes tab of the Task Manager, which means that the notepad.exe became a hidden process. Finally, the HPD was executed, and the hidden process (notepad.exe) was successfully detected as shown in Fig 5.



ProcessName
gwx
gwxconfigmanager
hpcustpartic
iastordatamgrsvc
iastordatamgrsvc
igfxsrvc
inetinfo
java
javaw
jdk-7u25-windows-i586
kinectmanagementservice
launcher
lms
logonui
mp3el2

ProcessName
inetinfo
smsvchost
winword
fdlauncher
svchost
taskhost
chrome
wininit
searchprotocolhost
sftlist
rslggr
vmware-usbarbitrator64
prefetch_parser.vshost
wdbackupengine
devenv

ProcessName	Path
notepad.exe	c:\windows\system32\notepad.exe

Fig.5: Detecting a Hidden Process

The results have shown that the number of files in the Prefetch folder may be larger than the number of the process. However, the experimental results indicate that the proposed HPD program is able to detect any user/kernel hidden process with a simple, convenient, and efficient method.

5. CONCLUSION

Malicious programs use one or more approaches to cover their presence, and this behavior called hide processes. To detect hidden processes, special security programs should be used. This paper introduced a new way to detect hidden processes based on Prefetch files. The results clearly demonstrated the efficient of the proposed method.

The proposed HPD program uses only two useful pieces of information (name and path) of Prefetch files, but Prefetch files have more. For example, they contain the date that the application was last launched, how many times that a program has been launched, a list of DLL and files that are accessed by the program, and more. All of these information could be used to get more details about any hidden process.

6. REFERENCES

[1] Hale Ligh, M., Case, A. , Levy, J. , Walters, A. 2014, " The Art of Memory Forensics", John Wiley & Sons, Inc.

[2] Carvey, H., 2012, "Windows Forensic Analysis Toolkit", Syngress.

[3] Blunden, B. , 2013, "The Rootkit Arsenal", 2nd Edition,

[4] Wen, Y., Zhao, J. , Wang, H. ,Implicit Detection of Hidden Processes with a Local-Booted Virtual Machine, International Journal of Security and Its Applications vol. 2. No. 4, 2008

[5] Rutkowski, J., 2003, Advanced Windows 2000 Rootkit Detection.

[6] Oroszlany, M., 2008, Rootkits under Windows OS and methods of their detection

[7] ARNOLD, T. , 2011, A COMPARATIVE ANALYSIS OF ROOTKIT DETECTION TECHNIQUES.

[8] Bozagac, C., 2006,GHOSTWARE AND ROOTKIT DETECTION TECHNIQUES FOR WINDOWS.

[9] Bravo, P. , García, D. , PROACTIVE DETECTION OF KERNEL-MODE ROOTKITS

[10] Hoglund, G., Butler, J. , 2005, Rootkits: Subverting the Windows Kernel.

[11] Messier, R., 2016, OPERATING SYSTEM FORENSICS, Syngress.

[12] http://diggfreeware.com/incredible-free-and-open-source-process-hider-and-file-hider/]

[13] Garcia, L., 2011, BULK EXTRACTOR WINDOWS PREFETCH DECODER.