Basic Artifacts

Table of Contents

Introduction

This is a tutorial on finding artifacts in a Windows system related to applications. There’s so much more than just looking at installed programs. As a forensic analyst, you’ll need to be aware of these different sources. Information from multiple sources will strengthen your investigation.

I’ve avoided directly answering the questions listed in the Data Leakage Case because I want people to try them on their own. This tutorial aims to help them understand the concepts of these sources of artifacts, how to understand the output, and includes some quick tips about how to think during an investigation.

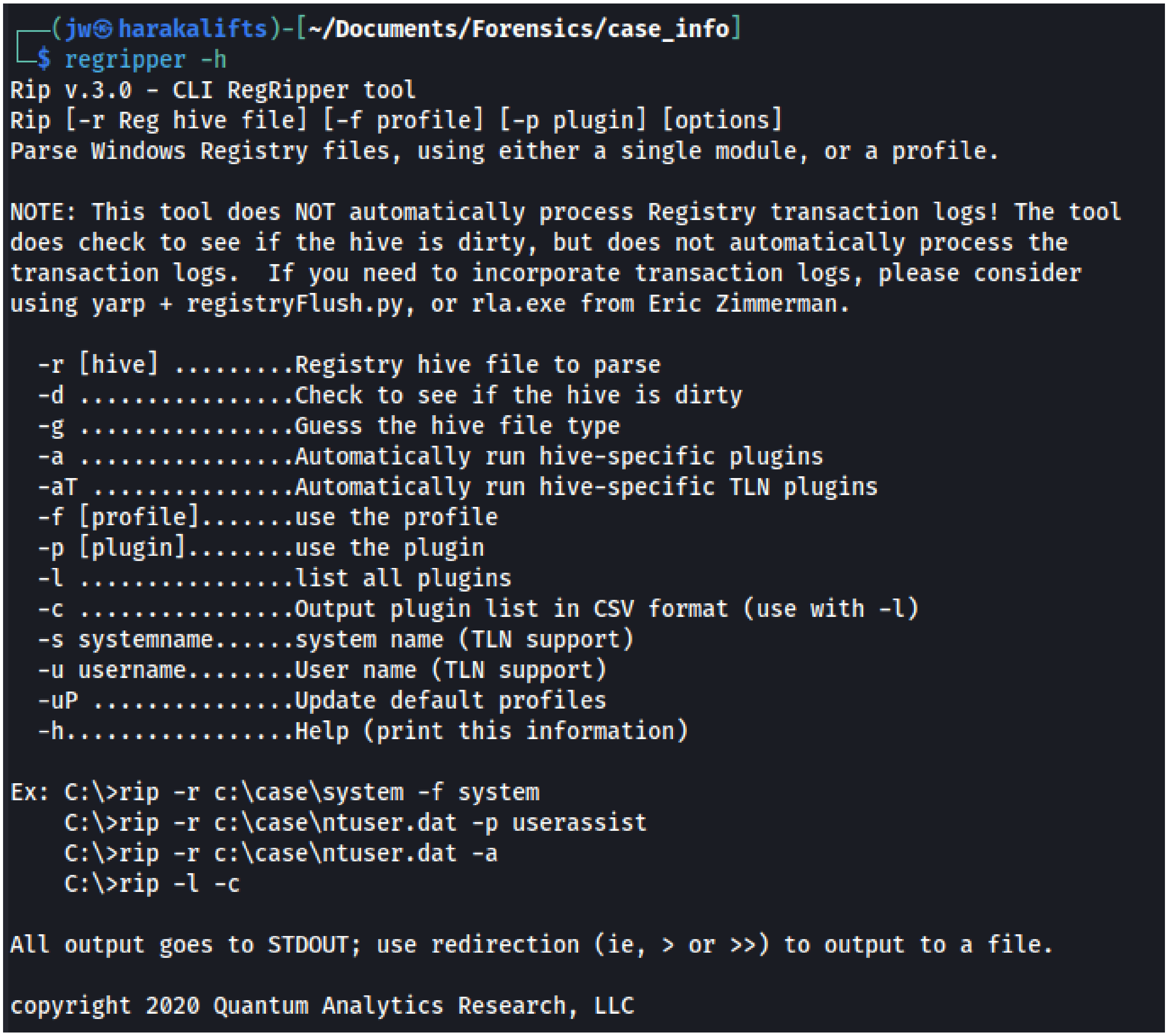
RegRipper

Reading:

- https://github.com/keydet89/RegRipper3.0

RegRipper is one of the most useful tools you’ll use in forensics, designed by Harlan Carvey. It makes it easy to extract data from the Windows registry. There are ways to manually parse the registry, but this requires researching the specific keys you need, which gets slow and boring very quickly. RegRipper takes all of that out of the equation. Let’s dive in.

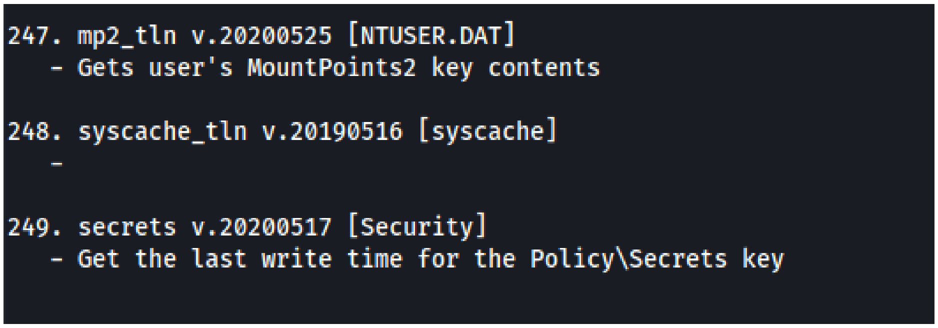
Before running anything, let’s take a look at the options:



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ regripper -h
Rip v.3.0 - CLI RegRipper tool
Rip [-r Reg hive file] [-f profile] [-p plugin] [options]
Parse Windows Registry files, using either a single module, or a profile.

NOTE: This tool does NOT automatically process Registry transaction logs! The tool
does check to see if the hive is dirty, but does not automatically process the
transaction logs.  If you need to incorporate transaction logs, please consider
using yarp + registryFlush.py, or rla.exe from Eric Zimmerman.

  -r [hive] .........Registry hive file to parse
  -d ................Check to see if the hive is dirty
  -g ................Guess the hive file type
  -a ................Automatically run hive-specific plugins
  -aT ...............Automatically run hive-specific TLN plugins
  -f [profile].......use the profile
  -p [plugin]........use the plugin
  -l ................list all plugins
  -c ................Output plugin list in CSV format (use with -l)
  -s systemname......system name (TLN support)
  -u username........User name (TLN support)
  -uP ...............Update default profiles
  -h.................Help (print this information)

Ex: C:\>rip -r c:\case\system -f system
    C:\>rip -r c:\case\ntuser.dat -p userassist
    C:\>rip -r c:\case\ntuser.dat -a
    C:\>rip -l -c

All output goes to STDOUT; use redirection (ie, > or >>) to output to a file.

copyright 2020 Quantum Analytics Research, LLC
```

For this tutorial, we’re mostly interested in the **-r** and **-p** options.

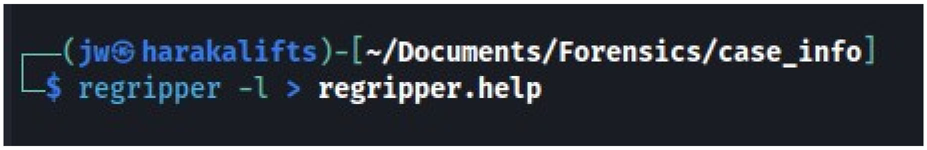
To get a list of all supported plugins, run **regripper -l**.



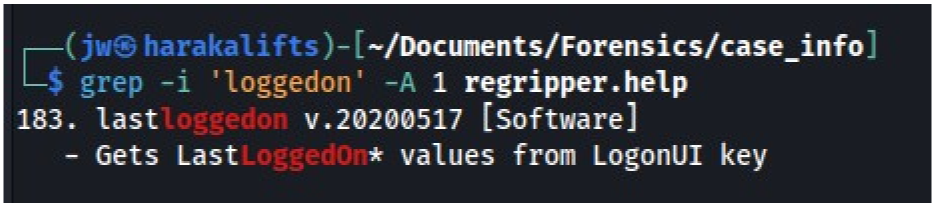
```
247. mp2_tln v.20200525 [NTUSER.DAT]
   - Gets user's MountPoints2 key contents

248. syscache_tln v.20190516 [syscache]
   -

249. secrets v.20200517 [Security]
   - Get the last write time for the Policy\Secrets key
```

That’s 249 plugins it supports. How do you find which plugins you need?

Pro tip: save the plugin list to a file so you can search it later.

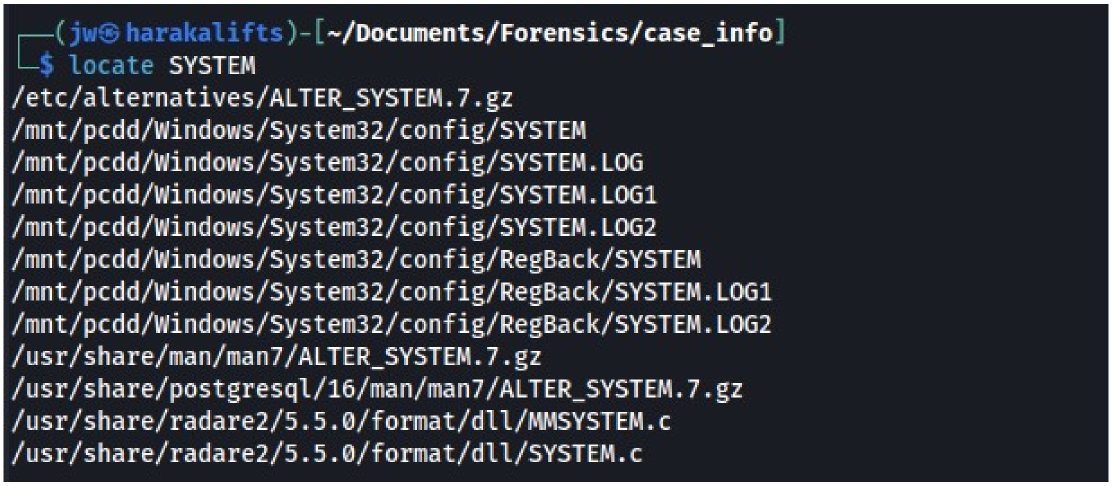


```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ regripper -l > regripper.help
```



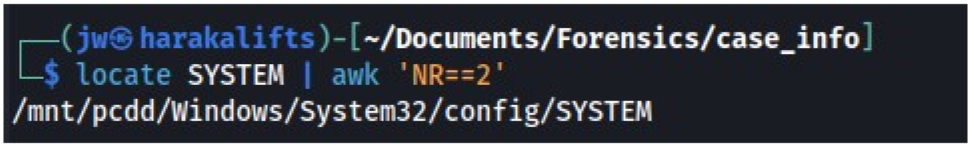
```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ grep -i 'loggedon' -A 1 regripper.help
183. lastloggedon v.20200517 [Software]
   - Gets LastLoggedOn* values from LogonUI key
```

Every plugin lists the registry hive that it operates on. You need to supply the full path to the hive. With just a little research, you’ll find these hives in a few different locations. To find them on my system, I used **locate**:



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ locate SYSTEM
/etc/alternatives/ALTER_SYSTEM.7.gz
/mnt/pcdd/Windows/System32/config/SYSTEM
/mnt/pcdd/Windows/System32/config/SYSTEM.LOG
/mnt/pcdd/Windows/System32/config/SYSTEM.LOG1
/mnt/pcdd/Windows/System32/config/SYSTEM.LOG2
/mnt/pcdd/Windows/System32/config/RegBack/SYSTEM
/mnt/pcdd/Windows/System32/config/RegBack/SYSTEM.LOG1
/mnt/pcdd/Windows/System32/config/RegBack/SYSTEM.LOG2
/usr/share/man/man7/ALTER_SYSTEM.7.gz
/usr/share/postgresql/16/man/man7/ALTER_SYSTEM.7.gz
/usr/share/radare2/5.5.0/format/dll/MMSYSTEM.c
/usr/share/radare2/5.5.0/format/dll/SYSTEM.c
```

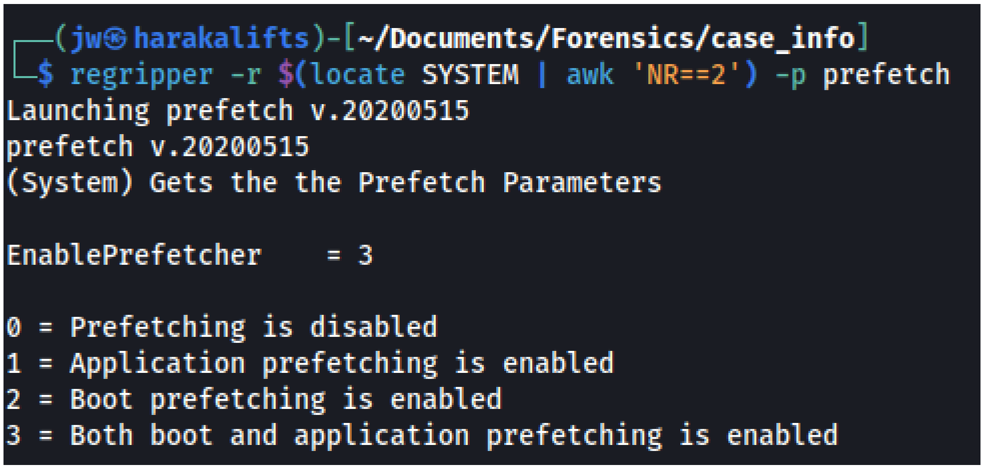
You can manually copy or write the path out, but because I want to impress people who use Linux, I used a fun trick with **awk**:



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ locate SYSTEM | awk 'NR==2'
/mnt/pcdd/Windows/System32/config/SYSTEM
```

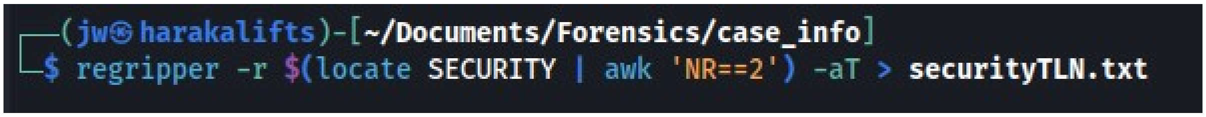
Figure 1: NR = Number of Record.

Once you have the path to the registry hive and the desired plugin, running RegRipper looks like this:



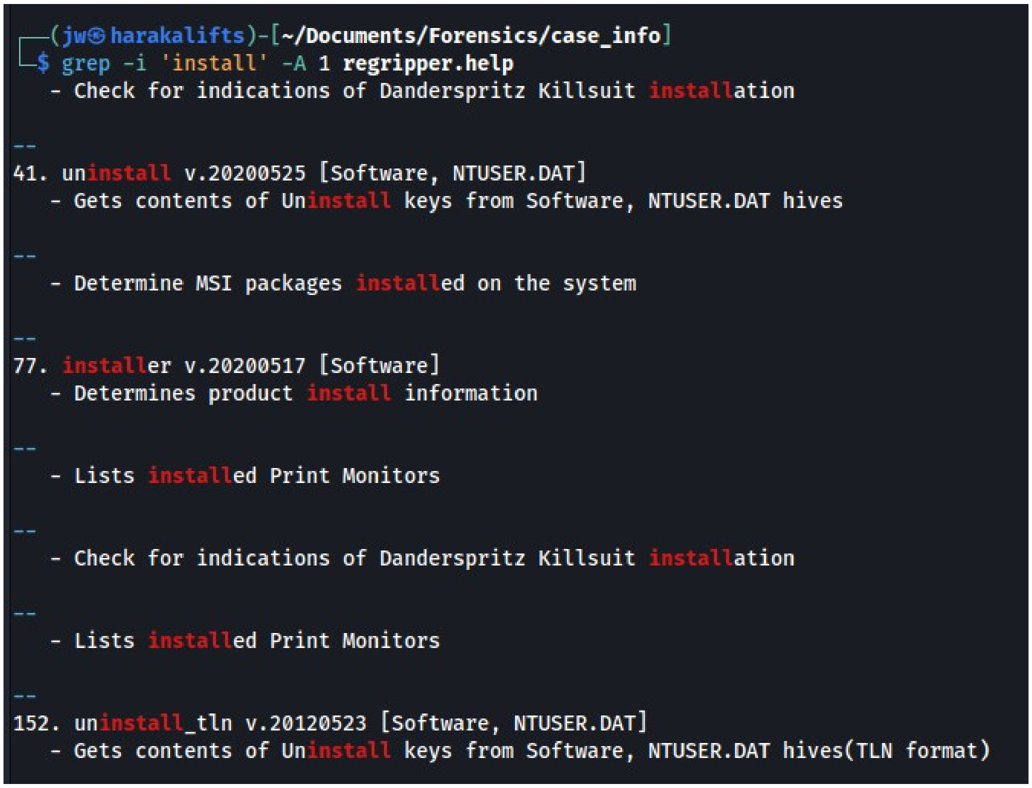
```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ regripper -r $(locate SYSTEM | awk 'NR==2') -p prefetch
Launching prefetch v.20200515
prefetch v.20200515
(System) Gets the the Prefetch Parameters

EnablePrefetcher    = 3

0 = Prefetching is disabled
1 = Application prefetching is enabled
2 = Boot prefetching is enabled
3 = Both boot and application prefetching is enabled
```

After confirming the output, I saved it to a file. Regripper can also output in TLN format, or **timeline** format. This will be useful later for crafting a timeline.



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ regripper -r $(locate SECURITY | awk 'NR==2') -aT > securityTLN.txt
```

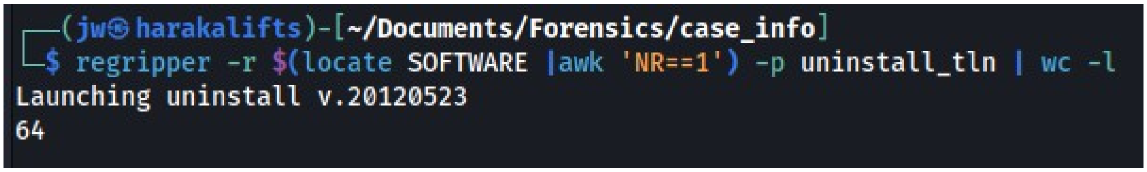
The Data Leakage Case asks you to find a list of installed programs. Let's start by searching through our help file for possible plugins.



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ grep -i 'install' -A 1 regripper.help
   - Check for indications of Danderspritz Killsuit installation

--
41. uninstall v.20200525 [Software, NTUSER.DAT]
   - Gets contents of Uninstall keys from Software, NTUSER.DAT hives

--
   - Determine MSI packages installed on the system

--
77. installer v.20200517 [Software]
   - Determines product install information

--
   - Lists installed Print Monitors

--
   - Check for indications of Danderspritz Killsuit installation

--
   - Lists installed Print Monitors

--
152. uninstall_tln v.20120523 [Software, NTUSER.DAT]
   - Gets contents of Uninstall keys from Software, NTUSER.DAT hives(TLN format)
```

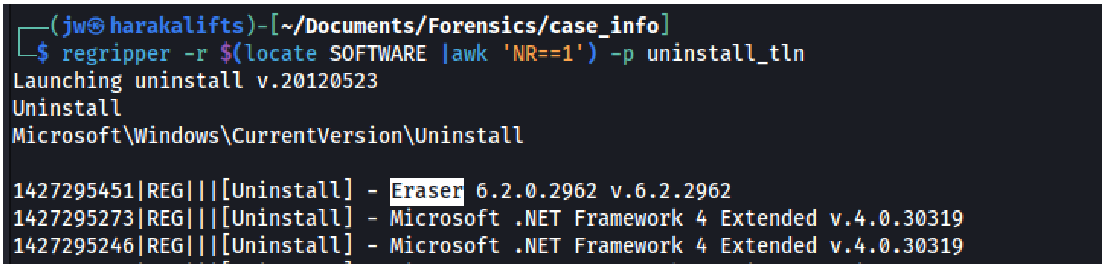
Two plugins immediately stand out: **uninstall**, and its TLN variant.

I like to count the lines before parsing just to see if I should redirect it to a file later. With **wc -l**, we find only 64 lines, which is very easy to parse directly on the command line:



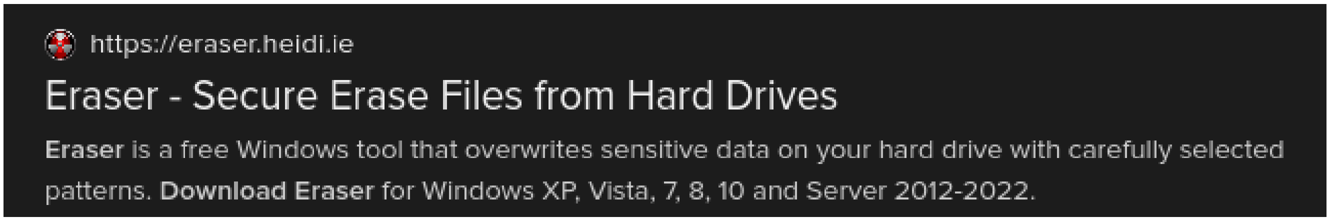
```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ regripper -r $(locate SOFTWARE |awk 'NR==1') -p uninstall_tln | wc -l
Launching uninstall v.20120523
64
```

There's one very notable entry I want to direct your attention to:



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ regripper -r $(locate SOFTWARE |awk 'NR==1') -p uninstall_tln
Launching uninstall v.20120523
Uninstall
Microsoft\Windows\CurrentVersion\Uninstall

1427295451|REG|||[Uninstall] - Eraser 6.2.0.2962 v.6.2.2962
1427295273|REG|||[Uninstall] - Microsoft .NET Framework 4 Extended v.4.0.30319
1427295246|REG|||[Uninstall] - Microsoft .NET Framework 4 Extended v.4.0.30319
```

A quick web search for Eraser brings up this:



https://eraser.heidi.ie

Eraser - Secure Erase Files from Hard Drives

Eraser is a free Windows tool that overwrites sensitive data on your hard drive with carefully selected patterns. **Download Eraser** for Windows XP, Vista, 7, 8, 10 and Server 2012-2022.

Remember the context of this exercise. We are investigating an employee for leaking company information. Eraser isn't a program installed by default on every Windows system, and most people won't need to securely erase every file they want to delete. The presence of this program may be a clue for the investigation.

However, this is not definitive evidence. This is only a list of **installed programs** on the system, nothing more. Keep this application in mind as you continue the exercise.

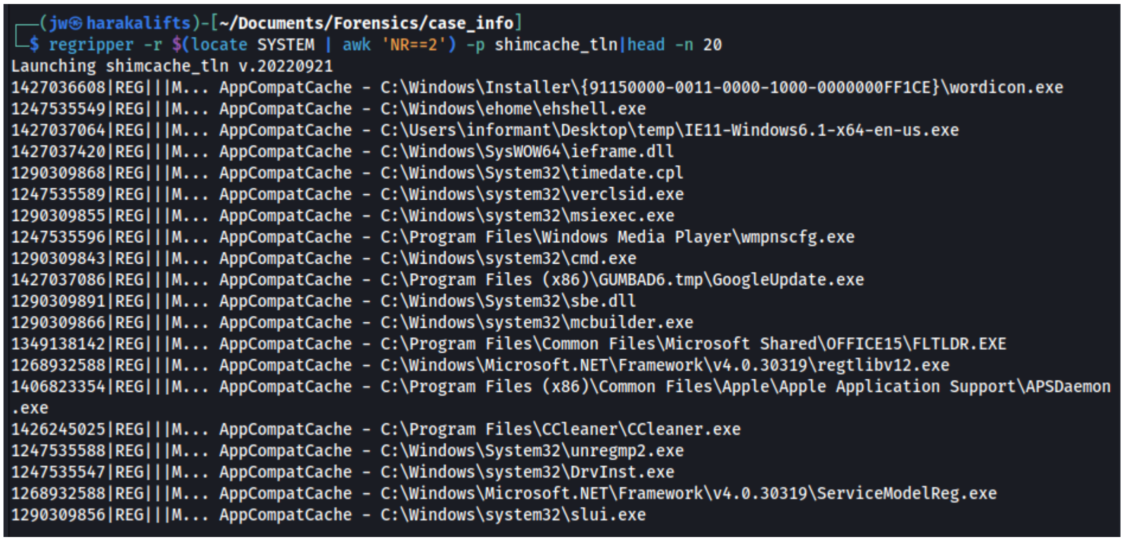
Shimcache

Reading:

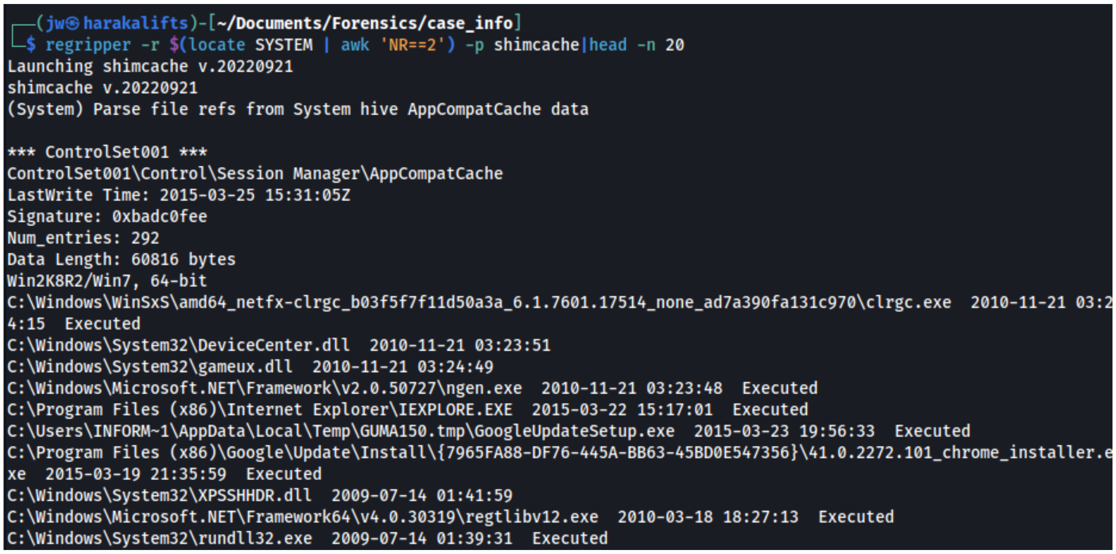
- https://cloud.google.com/blog/topics/threat-intelligence/caching-out-the-val/

So we found a suspicious file on the target machine. How do we prove it was actually used? Windows' shimcache is a solid answer. Essentially, it stores information related to file execution. In fact, a simple web search brings up a ton of forensic articles. Knowing how to extract data from the shimcache is a vital skill.

RegRipper provides a shimcache plugin. I've noticed the TLN output may be missing some key information. Notice the difference between the following two screenshots:



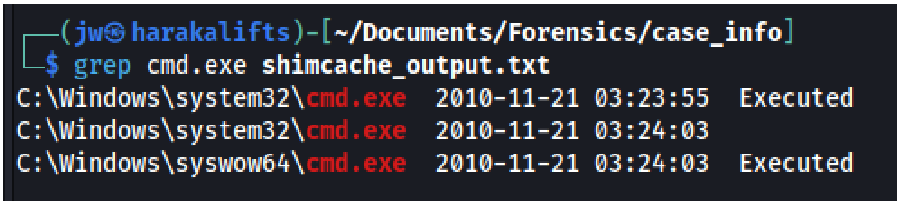
```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ regripper -r $(locate SYSTEM | awk 'NR==2') -p shimcache_tln|head -n 20
Launching shimcache_tln v.20220921
1427036608|REG|||M... AppCompatCache - C:\Windows\Installer\{91150000-0011-0000-1000-0000000FF1CE}\wordicon.exe
1247535549|REG|||M... AppCompatCache - C:\Windows\ehome\ehshell.exe
1427037064|REG|||M... AppCompatCache - C:\Users\informant\Desktop\temp\IE11-Windows6.1-x64-en-us.exe
1427037420|REG|||M... AppCompatCache - C:\Windows\SysWOW64\ieframe.dll
1290309868|REG|||M... AppCompatCache - C:\Windows\System32\timedate.cpl
1247535589|REG|||M... AppCompatCache - C:\Windows\system32\verclsid.exe
1290309855|REG|||M... AppCompatCache - C:\Windows\system32\msiexec.exe
1247535596|REG|||M... AppCompatCache - C:\Program Files\Windows Media Player\wmpnscfg.exe
1290309843|REG|||M... AppCompatCache - C:\Windows\system32\cmd.exe
1427037086|REG|||M... AppCompatCache - C:\Program Files (x86)\GUMBAD6.tmp\GoogleUpdate.exe
1290309891|REG|||M... AppCompatCache - C:\Windows\System32\sbe.dll
1290309866|REG|||M... AppCompatCache - C:\Windows\system32\mcbuilder.exe
1349138142|REG|||M... AppCompatCache - C:\Program Files\Common Files\Microsoft Shared\OFFICE15\FLTLDR.EXE
1268932588|REG|||M... AppCompatCache - C:\Windows\Microsoft.NET\Framework\v4.0.30319\regtlibv12.exe
1406823354|REG|||M... AppCompatCache - C:\Program Files (x86)\Common Files\Apple\Apple Application Support\APSDaemon
.exe
1426245025|REG|||M... AppCompatCache - C:\Program Files\CCleaner\CCleaner.exe
1247535588|REG|||M... AppCompatCache - C:\Windows\System32\unregmp2.exe
1247535547|REG|||M... AppCompatCache - C:\Windows\system32\DrvInst.exe
1268932588|REG|||M... AppCompatCache - C:\Windows\Microsoft.NET\Framework\v4.0.30319\ServiceModelReg.exe
1290309856|REG|||M... AppCompatCache - C:\Windows\system32\slui.exe
```



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ regripper -r $(locate SYSTEM | awk 'NR==2') -p shimcache|head -n 20
Launching shimcache v.20220921
shimcache v.20220921
(System) Parse file refs from System hive AppCompatCache data

*** ControlSet001 ***
ControlSet001\Control\Session Manager\AppCompatCache
LastWrite Time: 2015-03-25 15:31:05Z
Signature: 0xbadc0fee
Num_entries: 292
Data Length: 60816 bytes
Win2K8R2/Win7, 64-bit
C:\Windows\WinSxS\amd64_netfx-clrgc_b03f5f7f11d50a3a_6.1.7601.17514_none_ad7a390fa131c970\clrgc.exe  2010-11-21 03:2
4:15  Executed
C:\Windows\System32\DeviceCenter.dll  2010-11-21 03:23:51
C:\Windows\System32\gameux.dll  2010-11-21 03:24:49
C:\Windows\Microsoft.NET\Framework\v2.0.50727\ngen.exe  2010-11-21 03:23:48  Executed
C:\Program Files (x86)\Internet Explorer\IEXPLORE.EXE  2015-03-22 15:17:01  Executed
C:\Users\INFORM~1\AppData\Local\Temp\GUMA150.tmp\GoogleUpdateSetup.exe  2015-03-23 19:56:33  Executed
C:\Program Files (x86)\Google\Update\Install\{7965FA88-DF76-445A-BB63-45BD0E547356}\41.0.2272.101_chrome_installer.e
xe  2015-03-19 21:35:59  Executed
C:\Windows\System32\XPSSHHDR.dll  2009-07-14 01:41:59
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\regtlibv12.exe  2010-03-18 18:27:13  Executed
C:\Windows\System32\rundll32.exe  2009-07-14 01:39:31  Executed
```

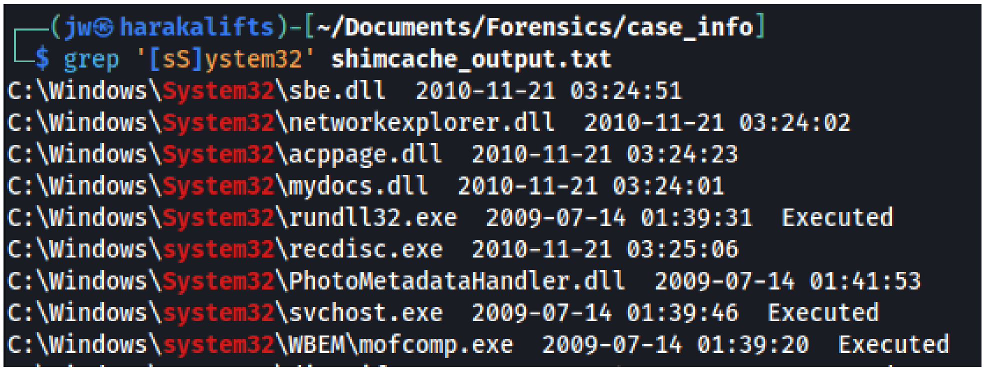
In the second screenshot, some of the entries are appended with “Executed”. This is a very important distinction the shimcache records. It records files that either execute *or* edit system configuration. Not all files are executable, but knowing which ones *did* execute is important. I’m going to focus on the non-TLN output for this writeup.

The output is formatted with full paths, execution time and whether or not the file was executed:



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ grep cmd.exe shimcache_output.txt
C:\Windows\system32\cmd.exe  2010-11-21 03:23:55  Executed
C:\Windows\system32\cmd.exe  2010-11-21 03:24:03
C:\Windows\syswow64\cmd.exe  2010-11-21 03:24:03  Executed
```

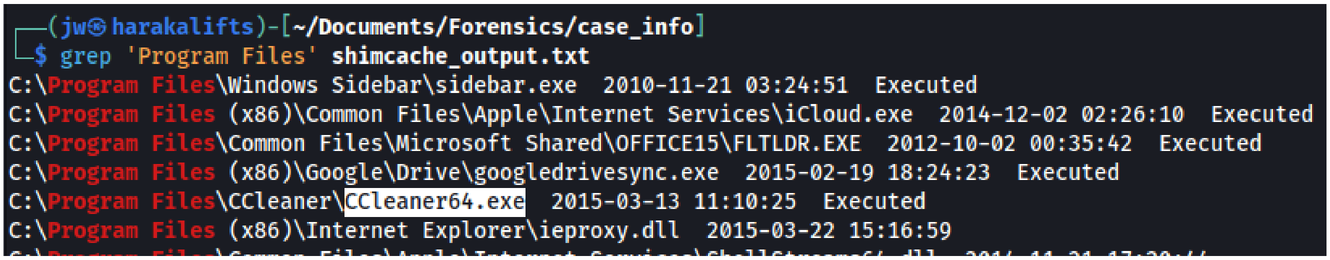
I’m not sure why, but some of the directories aren’t properly named. System32 can be found both capitalized and not. You may want to keep this in mind while parsing this output.



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ grep '[sS]ystem32' shimcache_output.txt
C:\Windows\System32\sbe.dll  2010-11-21 03:24:51
C:\Windows\System32\networkexplorer.dll  2010-11-21 03:24:02
C:\Windows\System32\acppage.dll  2010-11-21 03:24:23
C:\Windows\System32\mydocs.dll  2010-11-21 03:24:01
C:\Windows\System32\rundll32.exe  2009-07-14 01:39:31  Executed
C:\Windows\system32\recdisc.exe  2010-11-21 03:25:06
C:\Windows\System32\PhotoMetadataHandler.dll  2009-07-14 01:41:53
C:\Windows\system32\svchost.exe  2009-07-14 01:39:46  Executed
C:\Windows\system32\WBEM\mofcomp.exe  2009-07-14 01:39:20  Executed
```

Regardless, the actual files seem to be organized properly. Windows stores critical files under the System32 and SysWOW64 directories. It would be very unusual for an ordinary user to store personal files in these directories, so I’ll filter them out. Let’s look under the Program Files folders.

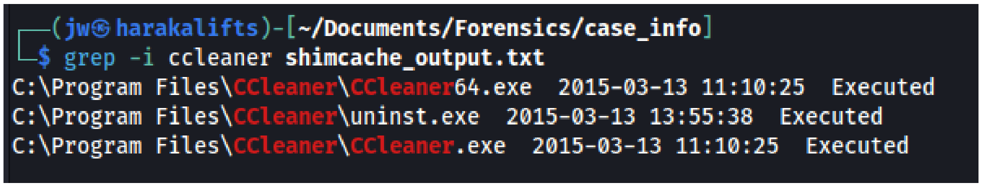
Almost immediately we find an unusual file:



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ grep 'Program Files' shimcache_output.txt
C:\Program Files\Windows Sidebar\sidebar.exe  2010-11-21 03:24:51  Executed
C:\Program Files (x86)\Common Files\Apple\Internet Services\iCloud.exe  2014-12-02 02:26:10  Executed
C:\Program Files\Common Files\Microsoft Shared\OFFICE15\FLTLDR.EXE  2012-10-02 00:35:42  Executed
C:\Program Files (x86)\Google\Drive\googledrivesync.exe  2015-02-19 18:24:23  Executed
C:\Program Files\CCleaner\CCleaner64.exe  2015-03-13 11:10:25  Executed
C:\Program Files (x86)\Internet Explorer\ieproxy.dll  2015-03-22 15:16:59
```

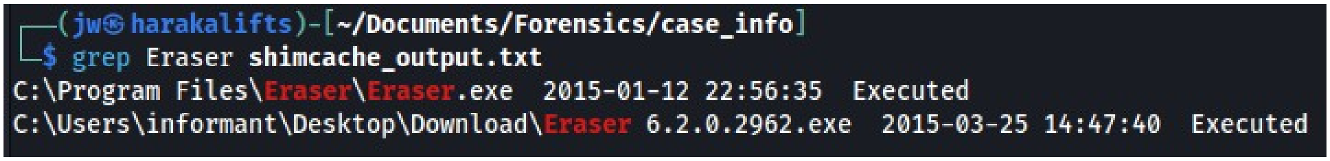
Just like with Eraser, if you search it up, you’ll find CCleaner is a legitimate program to securely delete files. It’s presence isn’t inherently inculpatory, but now we have a time and date for the execution of this program.

If you search for CCleaner in the shimcache output, you'll also find an uninstaller. This is why it didn't show up in the earlier example. I later saved all executed programs to another file.



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ grep -i ccleaner shimcache_output.txt
C:\Program Files\CCleaner\CCleaner64.exe  2015-03-13 11:10:25  Executed
C:\Program Files\CCleaner\uninst.exe  2015-03-13 13:55:38  Executed
C:\Program Files\CCleaner\CCleaner.exe  2015-03-13 11:10:25  Executed
```

Speaking of Eraser, it also shows up in the shimcache:



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ grep Eraser shimcache_output.txt
C:\Program Files\Eraser\Eraser.exe  2015-01-12 22:56:35  Executed
C:\Users\informant\Desktop\Download\Eraser 6.2.0.2962.exe  2015-03-25 14:47:40  Executed
```

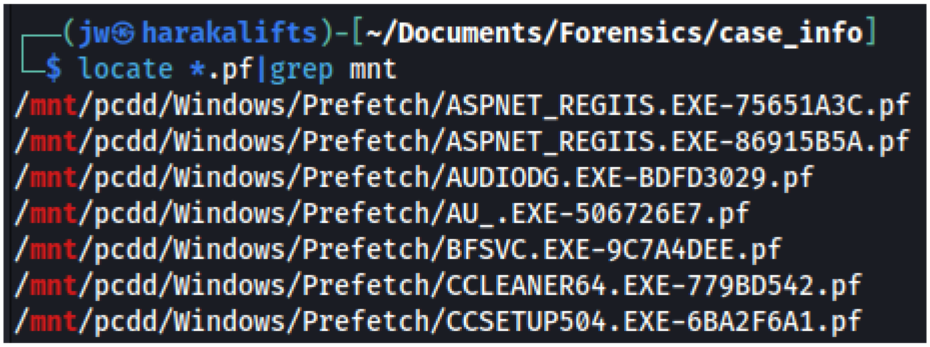
Prefetch

Reading:

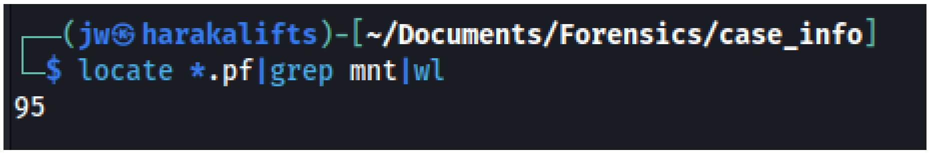
- https://forensics.wiki/prefetch/
- https://github.com/no-sec/prefetchkit

Prefetch files are used by Windows to improve performance. They are common (and valuable) artifacts left over whenever an application runs. Notably, they keep track of *how many times* an application was run.

Let’s start by looking for prefetch files on the Windows system;



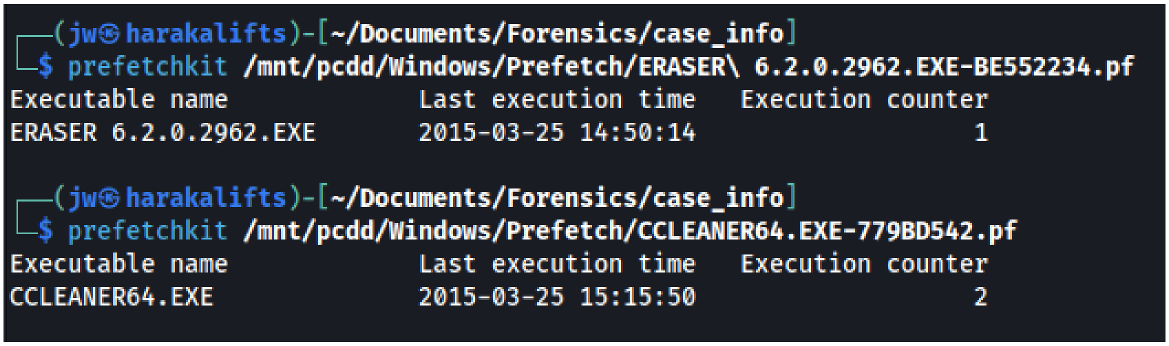
```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ locate *.pf|grep mnt
/mnt/pcdd/Windows/Prefetch/ASPNET_REGIIS.EXE-75651A3C.pf
/mnt/pcdd/Windows/Prefetch/ASPNET_REGIIS.EXE-86915B5A.pf
/mnt/pcdd/Windows/Prefetch/AUDIODG.EXE-BDFD3029.pf
/mnt/pcdd/Windows/Prefetch/AU_.EXE-506726E7.pf
/mnt/pcdd/Windows/Prefetch/BFSVC.EXE-9C7A4DEE.pf
/mnt/pcdd/Windows/Prefetch/CCLEANER64.EXE-779BD542.pf
/mnt/pcdd/Windows/Prefetch/CCSETUP504.EXE-6BA2F6A1.pf
```



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ locate *.pf|grep mnt|wl
95
```

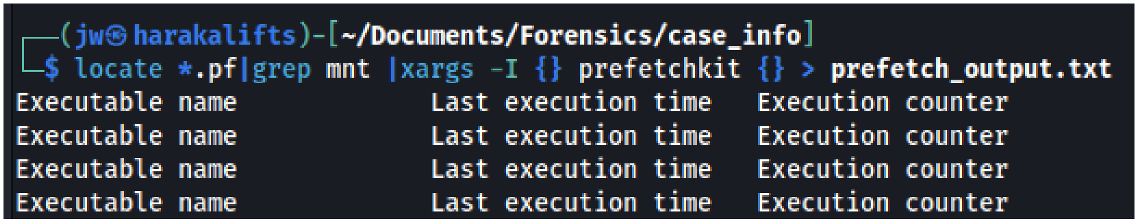
Figure 2: Since I use 'wc -l' so much, I made an alias for it.

For this section, I’m using a tool called **prefetchkit**, available on GitHub and as a Rust package. This is a remarkably simple tool. All you need to do is supply the prefetch file:

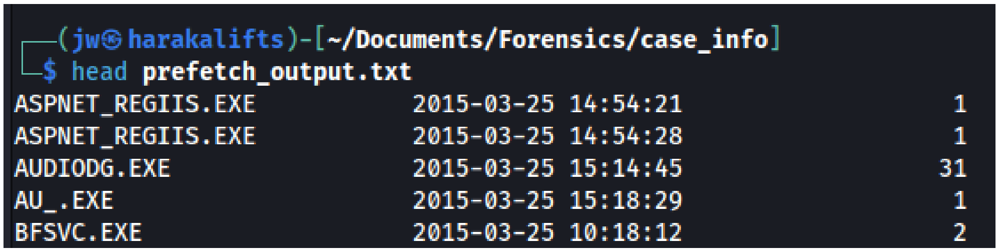


```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ prefetchkit /mnt/pcdd/Windows/Prefetch/ERASER\ 6.2.0.2962.EXE-BE552234.pf
Executable name             Last execution time   Execution counter
ERASER 6.2.0.2962.EXE       2015-03-25 14:50:14                   1

┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ prefetchkit /mnt/pcdd/Windows/Prefetch/CCLEANER64.EXE-779BD542.pf
Executable name             Last execution time   Execution counter
CCLEANER64.EXE              2015-03-25 15:15:50                   2
```

The only caveats are that it only takes one prefetch file at a time, and redirection is a little unusual:

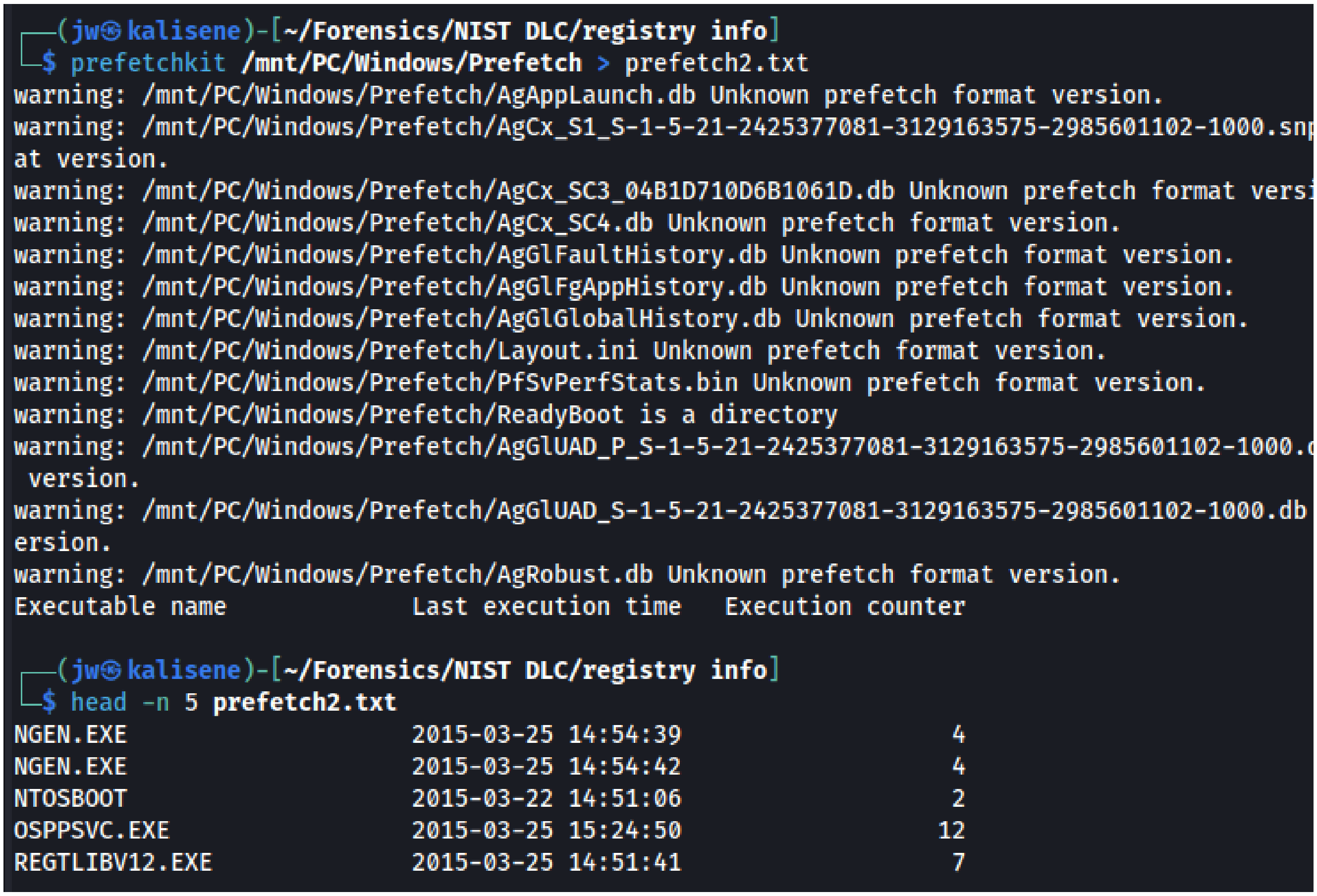


```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ locate *.pf|grep mnt |xargs -I {} prefetchkit {} > prefetch_output.txt
Executable name             Last execution time    Execution counter
Executable name             Last execution time    Execution counter
Executable name             Last execution time    Execution counter
Executable name             Last execution time    Execution counter
```



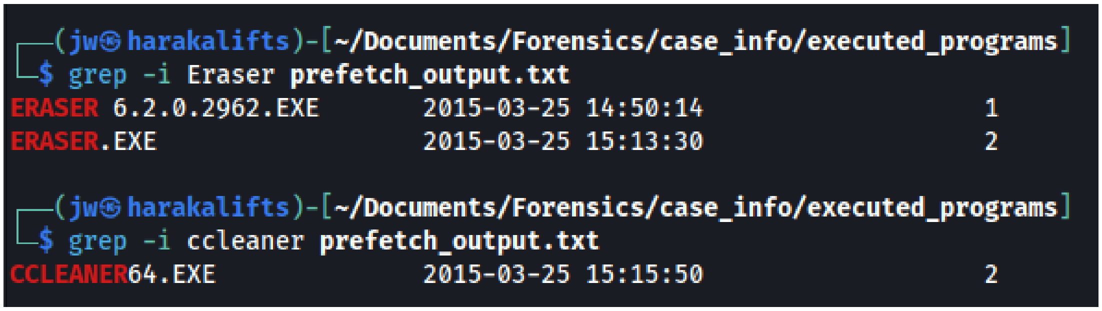
```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ head prefetch_output.txt
ASPNET_REGIIS.EXE           2015-03-25 14:54:21                 1
ASPNET_REGIIS.EXE           2015-03-25 14:54:28                 1
AUDIODG.EXE                 2015-03-25 15:14:45                31
AU_.EXE                     2015-03-25 15:18:29                 1
BFSVC.EXE                   2015-03-25 10:18:12                 2
```

Edit: I learned later that **prefetchkit** can also operate on an entire directory, which makes my previous solution look way more complicated.



```
┌──(jw㉿kalisene)-[~/Forensics/NIST DLC/registry info]
└─$ prefetchkit /mnt/PC/Windows/Prefetch > prefetch2.txt
warning: /mnt/PC/Windows/Prefetch/AgAppLaunch.db Unknown prefetch format version.
warning: /mnt/PC/Windows/Prefetch/AgCx_S1_S-1-5-21-2425377081-3129163575-2985601102-1000.snp
at version.
warning: /mnt/PC/Windows/Prefetch/AgCx_SC3_04B1D710D6B1061D.db Unknown prefetch format versi
warning: /mnt/PC/Windows/Prefetch/AgCx_SC4.db Unknown prefetch format version.
warning: /mnt/PC/Windows/Prefetch/AgGlFaultHistory.db Unknown prefetch format version.
warning: /mnt/PC/Windows/Prefetch/AgGlFgAppHistory.db Unknown prefetch format version.
warning: /mnt/PC/Windows/Prefetch/AgGlGlobalHistory.db Unknown prefetch format version.
warning: /mnt/PC/Windows/Prefetch/Layout.ini Unknown prefetch format version.
warning: /mnt/PC/Windows/Prefetch/PfSvPerfStats.bin Unknown prefetch format version.
warning: /mnt/PC/Windows/Prefetch/ReadyBoot is a directory
warning: /mnt/PC/Windows/Prefetch/AgGlUAD_P_S-1-5-21-2425377081-3129163575-2985601102-1000.d
 version.
warning: /mnt/PC/Windows/Prefetch/AgGlUAD_S-1-5-21-2425377081-3129163575-2985601102-1000.db
ersion.
warning: /mnt/PC/Windows/Prefetch/AgRobust.db Unknown prefetch format version.
Executable name             Last execution time    Execution counter

┌──(jw㉿kalisene)-[~/Forensics/NIST DLC/registry info]
└─$ head -n 5 prefetch2.txt
NGEN.EXE                    2015-03-25 14:54:39                 4
NGEN.EXE                    2015-03-25 14:54:42                 4
NTOSBOOT                    2015-03-22 14:51:06                 2
OSPPSVC.EXE                 2015-03-25 15:24:50                12
REGTLIBV12.EXE              2015-03-25 14:51:41                 7
```

So we know of two applications that are, or were, present on the target machine—Eraser and CCleaner. Let’s try searching for them in the output:



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info/executed_programs]
└─$ grep -i Eraser prefetch_output.txt
ERASER 6.2.0.2962.EXE       2015-03-25 14:50:14                   1
ERASER.EXE                  2015-03-25 15:13:30                   2

┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info/executed_programs]
└─$ grep -i ccleaner prefetch_output.txt
CCLEANER64.EXE              2015-03-25 15:15:50                   2
```

Now we have two suspicious applications, their last execution time and how many times they executed on the target machine. In your final report, you’ll want to exclude unnecessary findings. DLLHOST.EXE appears multiple times in the prefetch output, but will usually be a legitimate entry (unless you are specifically searching for malicious DLLHOST entries, of course).

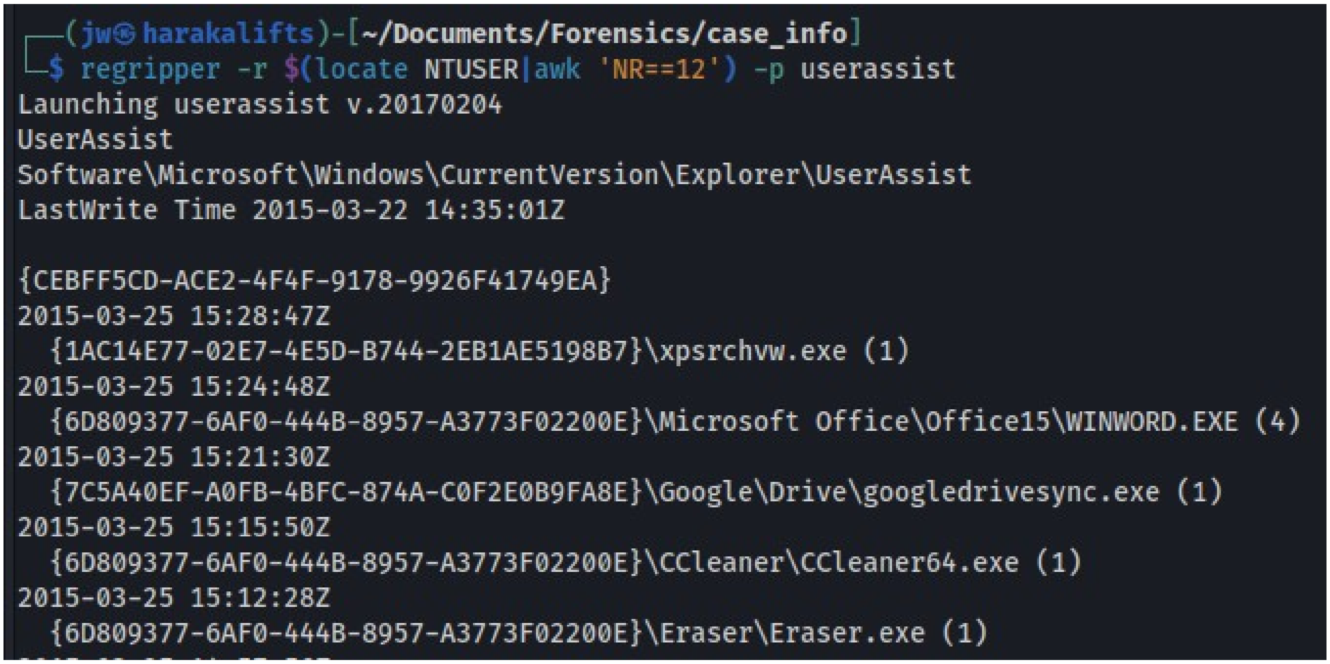
Userassist

Reading:

- https://blog.didierstevens.com/programs/userassist/

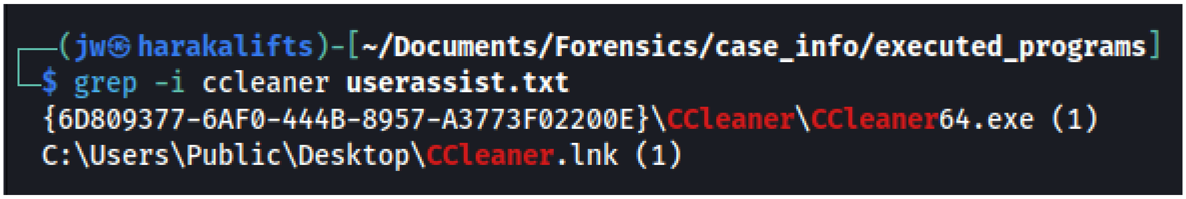
In addition to prefetch files, Windows also uses the UserAssist registry keys to track execution time/count of applications. Conceptually, the two are very similar.

You'll find a UserAssist plugin with RegRipper. If we run it:



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info]
└─$ regripper -r $(locate NTUSER|awk 'NR==12') -p userassist
Launching userassist v.20170204
UserAssist
Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist
LastWrite Time 2015-03-22 14:35:01Z

{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}
2015-03-25 15:28:47Z
  {1AC14E77-02E7-4E5D-B744-2EB1AE5198B7}\xpsrchvw.exe (1)
2015-03-25 15:24:48Z
  {6D809377-6AF0-444B-8957-A3773F02200E}\Microsoft Office\Office15\WINWORD.EXE (4)
2015-03-25 15:21:30Z
  {7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E}\Google\Drive\googledrivesync.exe (1)
2015-03-25 15:15:50Z
  {6D809377-6AF0-444B-8957-A3773F02200E}\CCleaner\CCleaner64.exe (1)
2015-03-25 15:12:28Z
  {6D809377-6AF0-444B-8957-A3773F02200E}\Eraser\Eraser.exe (1)
```

Again we find Eraser and CCleaner with execution times and counts, but notice the count on CCleaner. The prefetch output counted two instances of CCleaner, but here we see only one. If we try searching for it:



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info/executed_programs]
└─$ grep -i ccleaner userassist.txt
  {6D809377-6AF0-444B-8957-A3773F02200E}\CCleaner\CCleaner64.exe (1)
  C:\Users\Public\Desktop\CCleaner.lnk (1)
```

That **.lnk** file is a shortcut on Windows systems. You won't find it in the prefetch output. This is a unique distinction of the UserAssist keys.

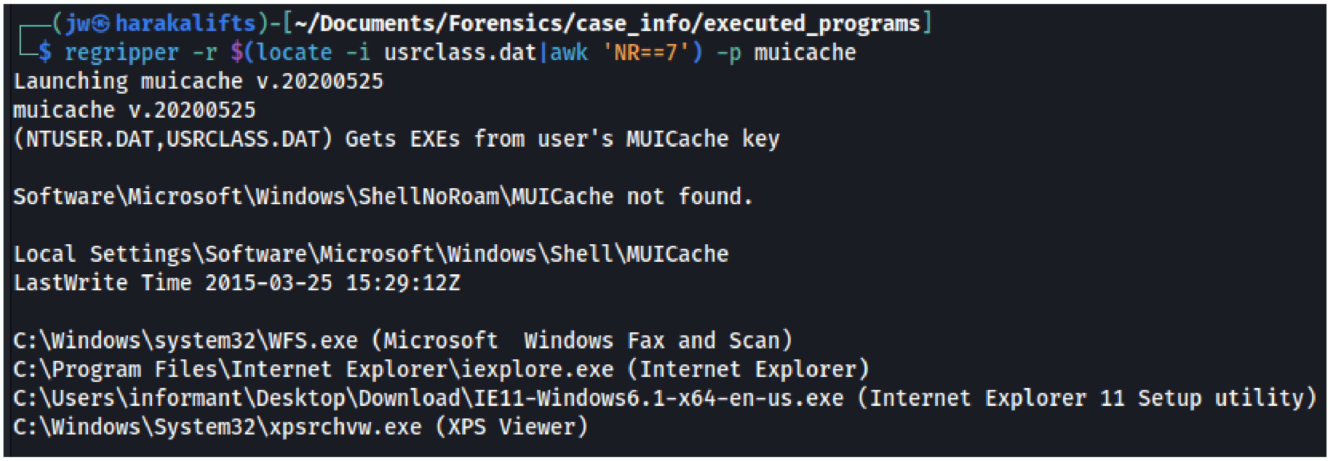
MUICache

Reading:

- https://learn.microsoft.com/en-us/windows/win32/intl/overview-of-mui
- https://www.magnetforensics.com/blog/forensic-analysis-of-muicache-files-in-windows/
- https://www.howtogeek.com/148499/what-is-an-xps-file-and-why-does-windows-want-me-to-print-to-one/

The MUICache, or Multilingual User Interface service, assists Windows with localization issues. As a forensic analyst, this doesn’t have a ton of information, but can still be helpful. The MUICache stores a little bit of information about application use.

With the RegRipper plugin:



```
┌──(jw㉿harakalifts)-[~/Documents/Forensics/case_info/executed_programs]
└─$ regripper -r $(locate -i usrclass.dat|awk 'NR==7') -p muicache
Launching muicache v.20200525
muicache v.20200525
(NTUSER.DAT,USRCLASS.DAT) Gets EXEs from user's MUICache key

Software\Microsoft\Windows\ShellNoRoam\MUICache not found.

Local Settings\Software\Microsoft\Windows\Shell\MUICache
LastWrite Time 2015-03-25 15:29:12Z

C:\Windows\system32\WFS.exe (Microsoft  Windows Fax and Scan)
C:\Program Files\Internet Explorer\iexplore.exe (Internet Explorer)
C:\Users\informant\Desktop\Download\IE11-Windows6.1-x64-en-us.exe (Internet Explorer 11 Setup utility)
C:\Windows\System32\xpsrchvw.exe (XPS Viewer)
```

There are only four entries. What this tells us is the subject of investigation *used these applications* around a certain time. The most notable entries are the Windows Fax and Scan and the XPS Viewer programs. An XPS file is a printable file format, kind of like PDFs. With the Fax and Scan program, this may indicate the subject intended to send a file via fax. This is not conclusive evidence, however. This is just something to keep in mind throughout the rest of the investigation.

Conclusion

This tutorial covered several locations of executable artifacts in a Windows system and how to extract information from them. The shimcache, prefetch files and UserAssist registry keys provide lots of valuable information about applications. The MUICache provides much less information, but may be contextually useful later. RegRipper is an extremely useful tool, but if it doesn’t have a plugin to parse something, you can quickly find another tool to accomplish that task. As a forensic analyst, you need to know how to find evidence in these locations and how to make sense of it.