Analysis of the AmCache

v2

Author	Blanche Lagny

Publication date	25/07/2019
Keywords	amcache, forensic, program execution, recentfilecache, Windows

For all requests / reply linked to this document, please contact blanche.lagny@ssi.gouv.fr

Abstract

The AmCache is an artifact which stores metadata related to PE execution and program installation on Windows 7 and Server 2008 R2 and above.

Frequently overlooked and understudied, this database is rarely fully exploited when doing incident response. Indeed, its correct interpretation is complex: a lot of special cases can occur that have to be taken into account when performing an analysis. However, the information collected by the AmCache is extremely useful and the lack of awareness about this artifact makes it very valuable, since it is easily overlooked by attackers erasing their tracks.

The purpose of this paper is to restore the confidence in the AmCache among digital forensic examiners by providing an extensive reference of the conclusions that can be drawn when analyzing this artifact.

Relying on existing public research, this paper also depends heavily on tests performed in a controlled environment. Those tests were used to validate, rectify or refine the conclusions found in the scientific literature and to fill the gaps in previous researches.

For instance, traces left by the installation of a program in Windows 7 were not explored yet and several changes in the inner workings of the AmCache in Windows 8 and 10 needed to be documented.

1. Introduction

The Application Compatibility Infrastructure was introduced in Windows operating systems, starting with Windows XP. This infrastructure is described both in the Microsoft docs [5] and in an article by Alex Ionescu [2]. Put simply, it allows an application to run even if it is no longer fully compatible with the system it is running on, or if the version of a dependency has changed. This infrastructure, also called the Shim Infrastructure, provides two artifacts for the digital investigator: the Application Compatibility Cache (also called ShimCache) and, since Windows 7, the AmCache. Since the Shim Infrastructure is used when an application runs, we can expect these artifacts to store some information about executed applications and even installed programs. In this article, we provide an in-depth study of the information available in the AmCache on Windows systems.

The AmCache has currently been seen under two different file formats: a BCF file, called `RecentFileCache.bcf`, and a Registry hive, called `AmCache.hve`. Contrary to other artifacts, the format used does not depend on the version of the operating system but rather on the version of the libraries in charge of filling the cache. Indeed, Microsoft is repackaging the current libraries for each OS version, which means that the artifact has the same format on a Windows 10 and on a Windows 7, provided that both systems are up-to-date. To update those libraries, a user should apply the Windows Update KB2952664 on a Windows 7 and KB2976978 on a Windows 8 and 8.1. The libraries are stored in `%WinDir%\System32` and start with *ae* (probably for Application Experience):

- aecache.dll;
- aeevts.dll;
- aeinv.dll;
- aelupsvc.dll;
- aepdu.dll;
- aepic.dll.

It is worth noting that the versioning system is following the Windows Version Number [1], with the build number appended to it, and that the libraries have not been seen with a version number inferior to that of their host operating system. This implies that the `RecentFileCache.bcf` file, which is part of the version 6.1.* of the libraries, is not present on a Windows 10, which uses a Windows Version Number of 10.0*. At the time of this writing, the up-to-date version of the libraries is 10.0.17673.1003.

Previous research was already done on the AmCache: in [1], Corey Harrell studies `RecentFileCache.bcf` and shows that this file records the path and name of executed applications that need to be shimmed. He also explains that `RecentFileCache.bcf` is flushed every night by a scheduled task, `ProgramDataUpdater`, showing that the AmCache has a mode of operation in two steps. Furthermore, in [6], Maxim Suhanov points out that recent versions of the libraries come with a new scheduled task, called `Microsoft Compatibility Appraiser`, that seems to update `AmCache.hve`. Finally, in [3] and [4], Yogesh Khatri demonstrates that `AmCache.hve` can also be used to know which programs were installed on a system. As for related tools, `RecentFileCache.bcf` and `AmCache.hve` can be parsed respectively by RecentFileCacheParser[2] and AmcacheParser[3], both by Eric Zimmerman. There is also a Regripper[4] parser for `AmCache.hve`, created by Harlan Carvey. These are valuable first steps in the interpretation of AmCache, but this article shows how to dig further. In particular, we show how to tap into the wealth of information available in the files created when the scheduled tasks are executed. We also focus on which pieces of information are updated in `AmCache.hve` when the scheduled tasks are executed, in order to understand why the timestamp associated with this artifact cannot be reliably interpreted as the execution time of the application.

This paper describes the format of the AmCache according to the version of the libraries on the system. When relevant, formats presenting several similarities are regrouped. This report is split in six chapters, each explaining the inner workings of the AmCache when running a version of the Shim libraries originally shipped with a given Windows version. The first chapter explores the artifacts left by the Shim Infrastructure on Windows 7 SP0 and SP1 and on Windows Server 2008 R2, reviewing in details all the files related to the AmCache. The next chapters explore along the same lines the behavior of the AmCache on Windows 8, Windows 8.1 and several versions of Windows 10.

[1] https://docs.microsoft.com/en-us/windows/desktop/sysinfo/operating-system-version
[2] https://f001.backblazeb2.com/file/EricZimmermanTools/RecentFileCacheParser.zip
[3] https://f001.backblazeb2.com/file/EricZimmermanTools/AmcacheParser.zip
[4] https://github.com/keydet89/RegRipper2.8

2. Behavior of libraries originally packaged with Windows 7 and Windows Server 2008 R2

This chapter details the behavior of the versions 6.1.7600.16385 and 6.1.7601.17514 of the libraries, shipped with Windows 7 SP0 and SP1 ”out-of-the-box”.

2.1. General behavior

When executing a PE, the service `AeLookupSvc`, which executes `%WinDir%\system32\svchost.exe -k netsvcs`, checks whether the PE needs shimming. If it does, the service stores the filename, with its path, in a file named `RecentFileCache.bcf`, located under `%WinDir%\AppCompat\Programs`. The format of this file is described in Appendix C. The service also stores path of the dependencies of the executed PE which need shimming.

At 00:30 every night, a scheduled task, `ProgramDataUpdater`, is executed. This task launches ”`%WinDir%\system32\rundll32.exe aepdu.dll,AePduRunUpdate`”, which flushes the `RecentFileCache.bcf` and stores all installed programs in `%WinDir%\AppCompat\Programs\AEINV_CURRENT.xml`. It then renames this file as `AEINV_PREVIOUS.xml`, overwriting the previous file. On some systems, it also updates a file called `AEINV_WER_{MachineId}_YYYYMMDD_HHmmss.xml`, located under the same directory as `AEINV_PREVIOUS.xml`. This task is only executed if the computer has been in an idle state for at least 3 minutes. If it is not (or if it is turned off), this task tries to execute for the next 23 hours.

2.2. RecentFileCache.bcf

This file contains the path of binaries executed between the last execution date of `ProgramDataUpdater` and the current time, in lowercase. The order in which those paths are stored is not always chronological (i.e. the last path is not always the last executed PE).

As an experiment, Wireshark v2.6.5 was installed on a virtual machine of a Windows 7 Ultimate 32-bit. The previous `RecentFileCache.bcf` contained the following entries:

- c:\windows\system32\mobsync.exe
- c:\program files\oracle\virtualbox guest additions\vboxdrvinst.exe
- c:\windows\system32\vboxservice.exe
- c:\windows\system32\vboxtray.exe

After the execution of `C:\Users\User\Downloads\Wireshark-win32-2.6.5.exe`, the `RecentFileCache.bcf` had the following lines appended to it:

- c:\program files\wireshark\vcredist_x86.exe
- c:\windows\system32\wusa.exe
- c:\windows\system32\wuauclt.exe
- c:\windows\system32\msiexec.exe

And finally, after the first launch of the application, one path was added:

- c:\program files\wireshark\dumpcap.exe

As the experiment shows, `RecentFileCache.bcf` does not store every binary that was executed: for instance, the PE which started the installation of Wireshark is not present. The experiment also proves that the PE stored does not need to be manually executed by the user and can be executed as a consequence of another execution.

Further tests indicate that binaries executed from a USB drive or a network share are not stored, even for PEs that show in `RecentFileCache.bcf` when executed from the drive. Tests also highlight that the storage of executed PEs in `RecentFileCache.bcf` depends on where the PE file resides on the system. For example, a PE in need of shimming appears in `RecentFileCache.bcf` when located in `C:\Users\<username>\Documents\test`, but the very same PE located in `C:\Users\<username>\Documents` is not registered. Furthermore, occurrence in `RecentFileCache.bcf` depends on how long the PE has been on the system. For instance, if a PE is executed as soon as it appears in the system, it is stored in `RecentFileCache.bcf`, but not if the user waits several hours before executing it. This last behavior has only been noticed when the PE is located in a user’s directory.

2.3. AEINV_PREVIOUS.xml

This file contains details about installed programs at the time of the last execution of the `ProgramDataUpdater` scheduled tasks. Once again, these entries are not stored in chronological order.

The definition of "installed programs" is not clear, but it seems to be at least composed of the programs listed under the following registry keys:

- `SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall` (if the value `SystemComponent` of the subkey associated with the program does not have a value of 1);
- `SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall` (if the value `SystemComponent` of the subkey associated with the program does not have a value of 1);
- `SOFTWARE\Microsoft\Windows\CurrentVersion\Run`.

As a consequence of the previous experiment, the installation of Wireshark led to changes in the registry: the program is now registered under `SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall` as shown in Fig. 2.1. After the execution of `ProgramDataUpdater`, the `AEINV_PREVIOUS.xml` file contains information about the program, as shown in Listing 2.1. Both entries are shown below for comparison.

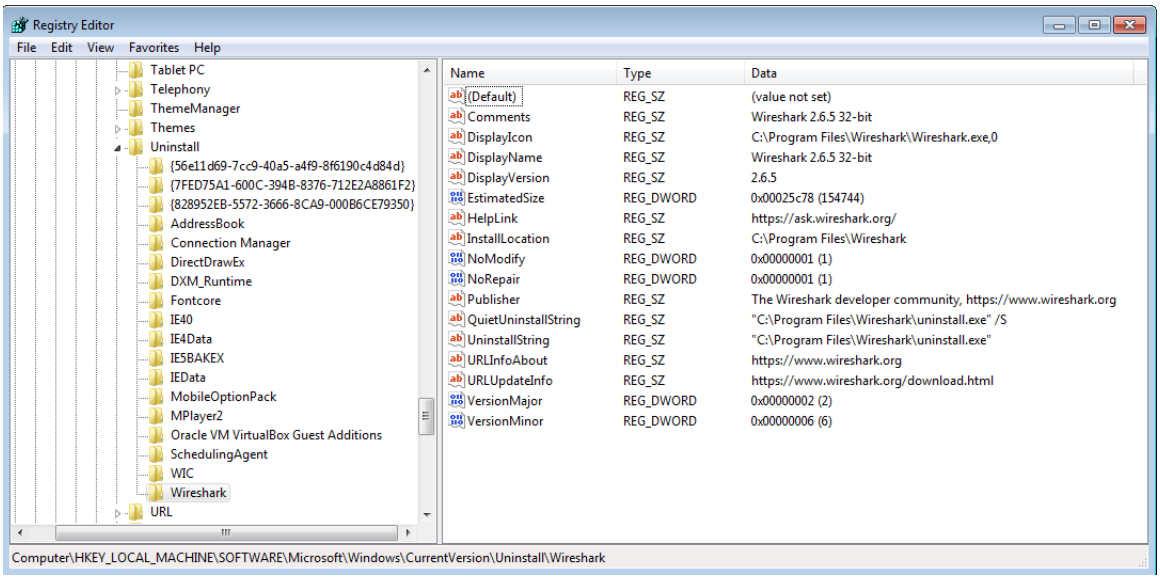


Fig. 2.1.: Content of `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Wireshark`

```
<Log Version="6.1.7601.17514">
  <ProgramList>
    [...]
    <Program Id="0000354384b2dbc2f6b2dc9dec22174dcf510000ffff" Name="Wireshark 2.6.5 32-bit" Publisher=
        "The Wireshark developer community, https://www.wireshark.org" Version="2.6.5" Source="
        AddRemoveProgram">
      <StaticProperties>
        <Files Id="00006ea5b5dae4e85c2b7a0ce4c0e609179961cd09fb"/>
      </StaticProperties>
    </Program>
    [...]
  </ProgramList>
  [...]
</Log>
```

2.1: Extract of AEINV_PREVIOUS.xml : Wireshark

In a nutshell, the majority of the information in this file is the same as the information found in the registry. `AEINV_PREVIOUS.xml` starts with an attribute `Log Version`, which is the version of the libraries used to populate this file. Then, the list `ProgramList` details every program installed on the machine. For Wireshark, the details consist of three fields that are also present in the Uninstall key: `Name`, `Publisher` and `Version`, and two that are not: `Id` and `Source`. The `Program Id` is not yet explained, although according to the Microsoft docs[1], it is supposed to be a hash of the `Name`, the `Version`, the `Publisher` and the `Language`. This is consistent with the fact that the `Program Id` is identical across different systems: the same version of a software installed on two different machines

[1] `https://docs.microsoft.com/en-us/windows/privacy/basic-level-windows-diagnostic-events-and-fields-1803#inventory-events`

results in the same `Program Id`. As for the `Source` attribute, its different values are detailed later, but in this case it is `AddRemoveProgram`, because the program is in an Uninstall key and was installed via an exe file. Then, there is an attribute called `StaticProperties` which only contains one attribute : `Files Id`. Much like the `Program Id`, this field is not explained but consistent across different machines, even if the program is installed in a different location on the drive.

Each program has an entry with the following information:

- Id;
- Name;
- Publisher;
- Version;
- Source = three possible values : `Msi`, `AddRemoveProgram` and `File`, which are explained below;
- MsiProductCode (if the program was installed via MSI);
- MsiPackageCode (if the program was installed via MSI) ;
- Language = the Microsoft's corresponding language identifier[2], in decimal (1033 for en-us).

The key `Source` is used to explain how the program was installed. If the key contains `Msi`, it means that the program was installed via MSI. If it contains `AddRemoveProgram`, it means that it was installed via an exe file and is in an Uninstall key. Finally, the `File` value appears to only be used to describe a PE that is listed in the Run key of the SOFTWARE hive. The other attributes for the `Source` key are extracted from the details of the PE file.

Another example is provided below: following the installation of the VirtualBox Guest Additions on the virtual machine, two entries are present in the XML file. The first one, which lists `AddRemoveProgram` as a `Source`, corresponds to the entry in the Uninstall key. The second one, which lists `File` as a `Source`, corresponds to the Run Key. This key is shown in Figure 2.2 with the details for the exe file to which it refers. The entry for Oracle VM VirtualBox Guest Additions in the XML file is shown in Listing 2.2. For the entry in `AEINV_PREVIOUS.xml`, the values are filled using the exe file properties : the `Name` is the `Product name`, the `Version` is the `Product version`... The `Publisher` is not listed in the details of VboxTray.exe, however the file is signed by "Oracle Corporation", which is probably where the information in the `Publisher` key came from.

```
<Log Version="6.1.7601.17514">
  <ProgramList>
    [...]
    <Program Id="00009056f81453a3569d36c40c2a6152d96400000904" Name="Oracle VM VirtualBox Guest
        Additions" Publisher="Oracle Corporation" Version="5.1.38.22592" Language="1033" Source="File">
    <StaticProperties>
      <Files Id="0000286ba9d8c8ff93b75d0cf3731d3bbd8b5f2db74e" />
    </StaticProperties>
    </Program>
    [...]
  </ProgramList>
  [...]
</Log>
```

2.2: Extract of AEINV_PREVIOUS.xml : Guest Additions

Another sublist, `IEAddOnList`, is present in `AEINV_PREVIOUS.xml`. It supposedly contains Internet Explorer add-ons. Since no way of enumerating all installed add-ons was found, the exhaustiveness of this list cannot be assumed. It contains the following information for each add-on:

- CLSID;
- Name;
- Type;
- Publisher;
- File Id (SHA1 of the file, preceded by '0000');
- File Name.

As an example, the entry for the add-on "InformationCardSigninHelper Class" is shown in Listing 2.3.

[2] https://docs.microsoft.com/en-us/windows/desktop/intl/language-identifier-constants-and-strings

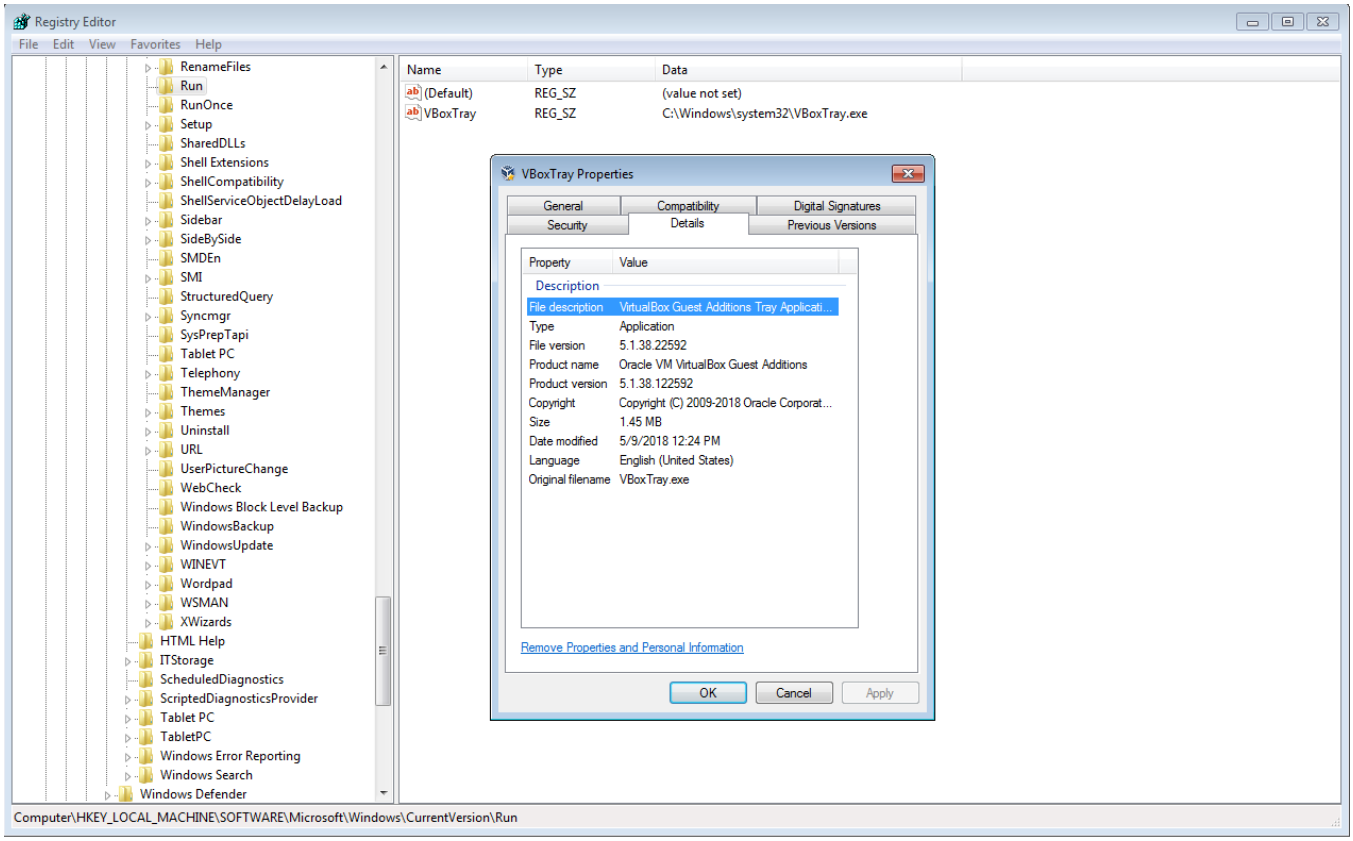


Fig. 2.2.: Content of HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run and details of VBoxTray.exe

```
<Log Version="6.1.7601.17514">
  [...]
  <IEAddOnList>
    <IEAddOn CLSID="{19916E01-B44E-4E31-94A4-4696DF46157B}" Name="InformationCardSigninHelper Class"
        Type="ActiveX" Publisher="Microsoft Corporation">
        <File Id="0000d8b095849b5172e07dff1562bad89f37037bf951" Name="icardie.dll" />
    </IEAddOn>
    [...]
  </IEAddOnList>
</Log>
```

2.3: Extract of AEINV_PREVIOUS.xml : IEAddOn

An exhaustive description of the format and content of `AEINV_PREVIOUS.xml` is provided in Appendix D.

2.4. AEINV_WER_{MachineId}_YYYYMMDD_HHmmss.xml

This file is not present on every system and the conditions of its presence are not yet explained. However, it has a real forensic value because it records every application that was installed, removed or updated and every PE file associated with the application. The meaning of "installed" is the same as the one in `AEINV_PREVIOUS.xml`.

The filename `AEINV_WER_{MachineId}_YYYYMMDD_HHmmss.xml` is composed of a field `MachineId` that is equal to the data contained in the value `ReportMachineId` of the key `SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\ClientTelemetry`. The filename also contains a timestamp, which is the date and time the report was created (in UTC). Since this report is updated every time the scheduled task `ProgramDataUpdater` is run, and not replaced by a new one, the timestamp does not change as the scheduled task is executed.

This XML file, which will be referenced as `AEINV_WER` for simplification, is composed of a header and three lists: System, ProgramList and IEAddOn which are detailed hereafter. The reader is invited to refer to Appendix E where the structure of the file is outlined. It can help follow detailed explanations given below.

2.4.1. Header

The `Report` key, which is the header of the XML file, is composed of a `Version`, a `TimeStamp`, a `SequenceNumber` and a `throttlingRuleSetGuid`, both of which are not yet explained. The timestamp corresponds to the time the

report was finished writing after the first execution of `ProgramDataUpdater`, in UTC. As a result, it usually refers to a few seconds later than the timestamp found in the filename. An example is shown in Listing 2.4.

```
<Report Version="1.3" TimeStamp="12/06/2018 09:43:40" SequenceNumber="1" ThrottlingRuleSetGuid="{
    F7D0E8C8-2DA8-4889-A910-3DE830B4148F}">
[...]
```

2.4: Extract of AEINV_WER_{49A35C5F-CCE9-48C7-B6EF-577A36E86135}_20181206_094337.xml: Header

2.4.2. System

In this field, information about the operating system is registered. An example of the `System` field from a Windows 7 Ultimate SP1 32-bit is shown in Listing 2.5.

```
<Report Version="1.3" TimeStamp="12/06/2018 09:43:40" SequenceNumber="1" ThrottlingRuleSetGuid="{
    F7D0E8C8-2DA8-4889-A910-3DE830B4148F}">
  <System MachineId="{49A35C5F-CCE9-48C7-B6EF-577A36E86135}" MajorVersion="6" MinorVersion="1"
      ServicePackMajor="1" ServicePackMinor="0" BuildNumber="7601" Sku="1" ProcessorArchitecture="1"
      OSPlatform="1" LocaleId="1033" GeoId="244"/>
[...]
```

2.5: Extract of AEINV_WER_{49A35C5F-CCE9-48C7-B6EF-577A36E86135}_20181206_094337.xml: System

Here are the meaning for the different fields :

- `MachineId` = the `ReportMachineId` mentioned previously;
- `MajorVersion` = the first part of the Windows Version Number;
- `MinorVersion` = the second part of the Windows Version Number;
- `ServicePackMajor`;
- `ServicePackMinor`;
- `BuildNumber`;
- `Sku` = integer that seems to reference the version of Windows installed as found in the OperatingSystemSKU Enum from PowerShell Core SDK[3] (tested for Ultimate and Enterprise Editions);
- `ProcessorArchitecture` (worth 1 for 32-bit and 2 for 64-bit);
- `OSPlatform` = this value could not be entirely clarified. The first hypothesis was that the value was supposed to identify the Windows Edition but in test, the values found were not consistent with the hypothesis: for Windows 7 Enterprise SP1 and Ultimate SP1 32-bit, the value is 1 whereas for a Windows 7 Enterprise SP1 64-bit, it is 2;
- `LocaleId` = decimal value of `LocaleName` in `HKCU\Control Panel\International`;
- `GeoId` = value `Nation` in `HKCU\Control Panel\International\Geo`.

2.4.3. ProgramList

This list keeps a record of every program installed on the system even if it does not need shimming to work on the system. The list is split into four sublists: Installed, which records programs that were installed on the system (even if they are not installed anymore), Orphan, which records executed files that do not belong to a program, Updated, which records some changes made to a program and Removed for uninstalled programs. We recall that the structure of the file is outlined in Appendix E and can be consulted while reading this section.

Several experiments were conducted to learn more about the behavior of those sublists. Each experiment is detailed below to help understand what can be found in each sublist, starting with `Installed`.

[3] `https://docs.microsoft.com/en-us/dotnet/api/microsoft.powershell.commands.operatingsystemsku?view=pscore-6.0.0`

Installed

The first experiment conducted was the same as for `AEINV_PREVIOUS.xml`: Wireshark v2.6.5 was installed on a virtual machine running Windows 7 Ultimate 32-bit. After the execution of `ProgramDataUpdater`, the information about the program installation was recorded in the `Installed` sublist of `AEINV_WER`. Extracts of the file are provided below to explain what can be found in the different keys.

The program header is shown in Listing 2.6. Just as in the `AEINV_PREVIOUS.xml` file, the information provided in the header of the program is the same as in the corresponding Uninstall key in the SOFTWARE hive. The only new attributes are `OnSystemDrive` and `EvidenceId`. The meaning of the former is not yet explained since it is always True, even when the program has been uninstalled and the files deleted. The latter is a value in hexadecimal that is explained later. For now, the reader is invited to note that in this example, the `EvidenceId` for Wireshark is 0x22.

```
[...]
<ProgramList>
  <Installed>
    <Program Name="Wireshark 2.6.5 32-bit" Type="Application" Source="AddRemoveProgram" Publisher="The
        Wireshark developer community, https://www.wireshark.org" Version="2.6.5" OnSystemDrive="true"
        EvidenceId="0x22" Id="0000354384b2dbc2f6b2dc9dec22174dcf510000ffff">
      [...]
    </Program>
    [...]
  </Installed>
  [...]
</ProgramList>
[...]
```

2.6: Extract of AEINV_WER_{49A35C5F-CCE9-48C7-B6EF-577A36E86135}_20181206_094337.xml: Installed Program header

The program has a list of different indicators that were not present in the `AEINV_PREVIOUS.xml` file, first of which, `AddRemoveProgramIndicators`, provided in Listing 2.7. It shows information about the Uninstall key with a `UniqueId` which is the `EvidenceId` previously mentioned. It also includes the name of the subkey in the registry.

```
[...]
<ProgramList>
  <Installed>
    <Program Name="Wireshark 2.6.5 32-bit" Type="Application" Source="AddRemoveProgram" Publisher="The
        Wireshark developer community, https://www.wireshark.org" Version="2.6.5" OnSystemDrive="true"
        EvidenceId="0x22" Id="0000354384b2dbc2f6b2dc9dec22174dcf510000ffff">
      <Indicators>
        <AddRemoveProgramIndicators>
          <AddRemoveProgram DisplayName="Wireshark 2.6.5 32-bit" CompanyName="The Wireshark developer
              community, https://www.wireshark.org" ProductVersion="2.6.5" RegistrySubKey="Wireshark"
              UniqueId="0x22" Id="00000773cfd2b58429384da8a9bea4a99e8bbef55402"/>
        </AddRemoveProgramIndicators>
        [...]
      </Indicators>
    </Program>
    [...]
  </Installed>
  [...]
</ProgramList>
[...]
```

2.7: Extract of AEINV_WER_{49A35C5F-CCE9-48C7-B6EF-577A36E86135}_20181206_094337.xml: Installed Program AddRemoveProgramIndicators

The next indicator, `ShellIndicators`, is shown in Listing 2.8. It contains information about what can be found in the Start Menu of the system. For Wireshark, there is an entry in the Start Menu called "Wireshark" which executes C:\Program Files\Wireshark\Wireshark.exe, as shown in Fig 2.3.

```
[...]
<Indicators>
  [...]
  <ShellIndicators>
    <Shell ShellName="Wireshark" TargetFileName="Wireshark.exe" UniqueId="0xa0" Id="00008
        f6fc717280228fa0fe0473fb0c23d38dd23f131"/>
  </ShellIndicators>
  [...]
</Indicators>
[...]
```

2.8: Extract of AEINV_WER_{49A35C5F-CCE9-48C7-B6EF-577A36E86135}_20181206_094337.xml: Installed Program ShellIndicators

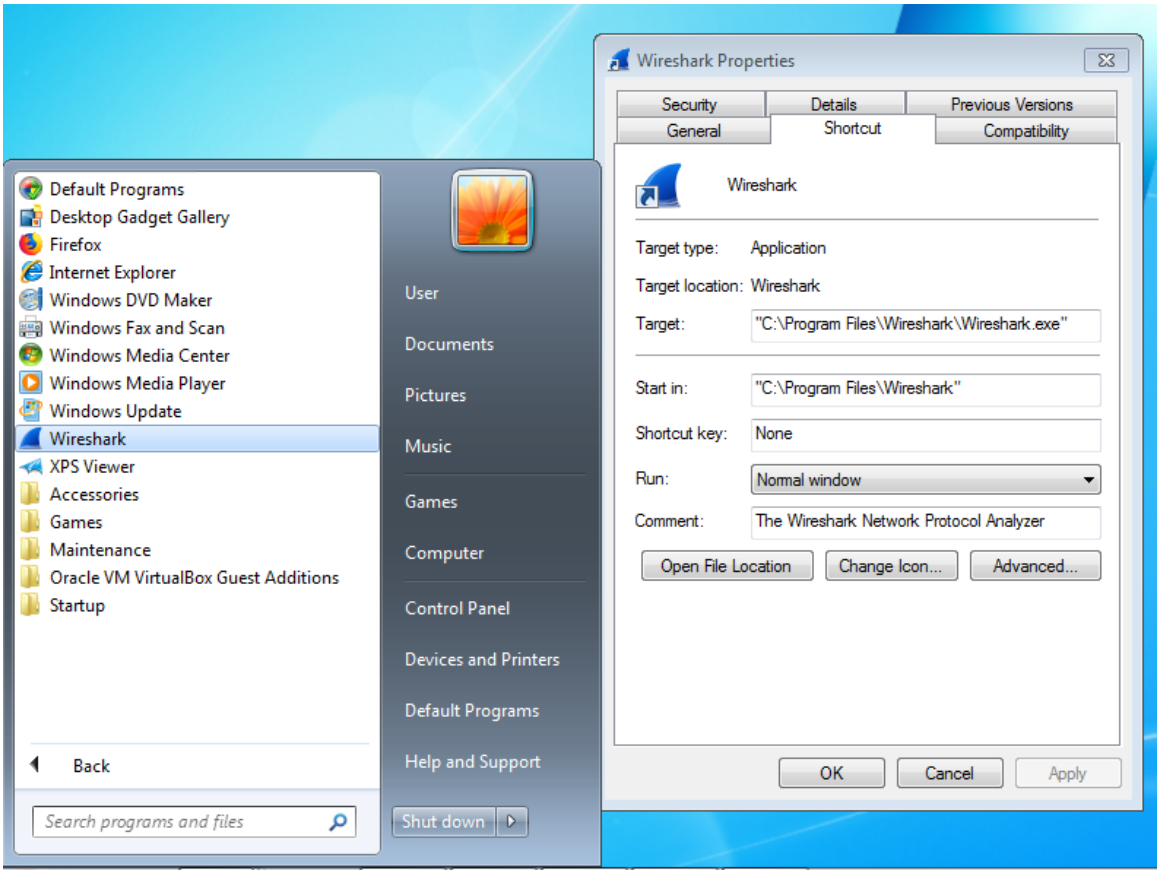


Fig. 2.3.: Start Menu and details of Wireshark.lnk

The `ShellIndicators` contains the entry name in the Start Menu (`ShellName`) and the filename of the PE it executes (`TargetFileName`). For this indicator, the meaning of the `UniqueId` has not yet been identified and is probably related to the lnk file and not the targeted exe file. Indeed, as shown later, the `UniqueId` is not the one associated with Wireshark.exe that is found in the `Files` sublist.

The `DirectoryIndicators` tag lists all the directories in the installation directory (itself included), which contains PE files. It is shown in Listing 2.9. Each entry in `DirectoryIndicators` has two keys: a `UniqueId` and an `Id`. The first key is used to know the location of the folder. For the example of Wireshark, the content of `C:\Program Files\Wireshark` is shown in Fig 2.4. Wireshark having an `EvidenceId` of 0x22, the first folder, which is the installation folder itself, has a `UniqueId` of 0x22+1=0x23. Since it contains PE files, it is listed in the `DirectoryIndicators`. Then, every PE or folder has a `UniqueId` associated with it, in alphabetical order, starting with the files. So capinfos.exe, the first PE file, is 0x24 and comerr32.dll, the second one, is 0x25. There are 62 (0x3E) PE files in the Wireshark folder, so the first folder, audio, has a `UniqueId` of 0x23+0x3E+0x1=0x62. This folder contains 2 DLLs, which means that it is listed in the `DirectoryIndicators` and that the next folder has a `UniqueId` of 0x62+0x2+0x1=0x65. As for the `Id`, its meaning has not been found yet. The first supposition was that it was the SHA-1 of the full path or of the name of the folder. Several encodings were tested : UTF-16LE, UTF-8, ASCII, but none matched with `Id`. Experiments were then made to see what could make the `Id` change. The first experiment was to install Wireshark in a different location: C:\Program Files\Wireshark64. This resulted in all the entries in `DirectoryIndicators` having the same `Directory Id` except the first one, so it is likely that the `Id` is linked to the name of the folder but not its path. The second experiment was to install Wireshark on a different system: once again, this resulted in all the entries in `DirectoryIndicators` having the same `Directory Id`. Finally, the 64-bit version of Wireshark was installed on a third system, resulting in a different `ProgramId` but still the same `Directory Id` if the directories were named the same, which was the case for all but 3 folders.

```
[...]
<Indicators>
  [...]
  <DirectoryIndicators>
    <Directory UniqueId="0x23" Id="00009afdcc213e845b1ed280a8d118317c363e807da5"/>
    <Directory UniqueId="0x62" Id="0000d02780464c90bf7bc1a299c4b9c9864aabc38041"/>
    <Directory UniqueId="0x65" Id="00000c1920ddeef6a4453b87d82e9e4bdd7cd7e34cfa"/>
    <Directory UniqueId="0x6b" Id="0000ff985ceec5256e32680e61528e85f1d606039299"/>
    <Directory UniqueId="0x6d" Id="00001835aab95f61091a75c2668c32fb3accb6b39f3c"/>
    <Directory UniqueId="0x77" Id="00002981edfd070ae25ff64b46362d1d48ee8bbaa3d3"/>
    <Directory UniqueId="0x7b" Id="0000fda4622bcc722e71a460e2fc47d59bf7dceb30c5"/>
```

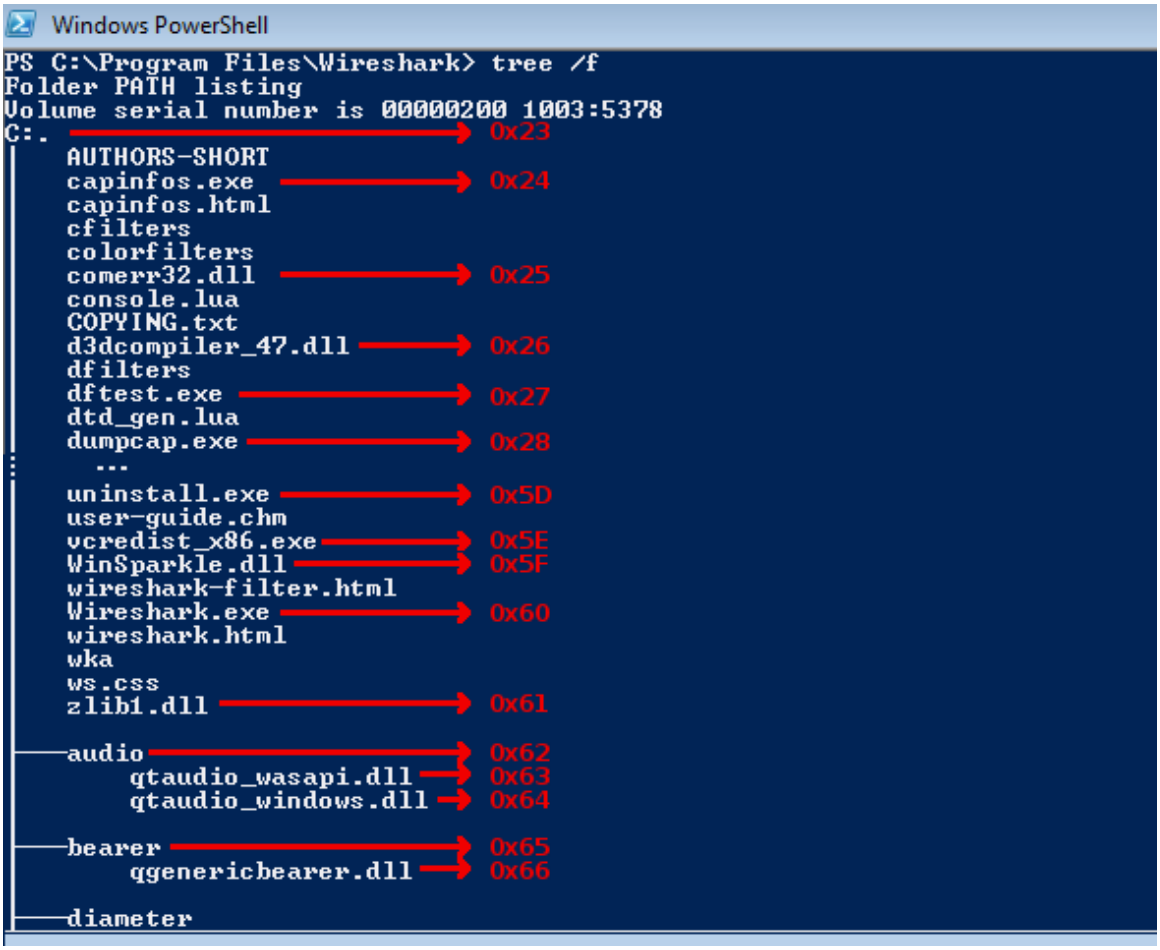


Fig. 2.4.: Partial content of C:\Program Files\Wireshark with the associated UniqueId

```
    <Directory UniqueId="0x7d" Id="00003e1b0e2dce63403b70b2b64d94406b25a6f9ecf3"/>
    <Directory UniqueId="0x7f" Id="0000cdff5a8b448d17782e1bc983e7f7abb756a41b34"/>
    <Directory UniqueId="0x80" Id="000089574cf70870d3b2c43857f917700e91afde8d86"/>
    <Directory UniqueId="0x81" Id="000036247d333a3c106b173c15a6892dc3613f478636"/>
    <Directory UniqueId="0x83" Id="000054e02e97208a5a43e9b4b63535fa0a6380512a87"/>
    <Directory UniqueId="0x90" Id="0000ab7ae667b5277a763ef1629b2c3e9b49c41c6a4f"/>
    <Directory UniqueId="0x92" Id="0000299fdc610f46b5395ac83f8ba9501cd4091d87bf"/>
  </DirectoryIndicators>
  [...]
</Indicators>
[...]
```

2.9: Extract of AEINV_WER_{49A35C5F-CCE9-48C7-B6EF-577A36E86135}_20181206_094337.xml: Installed Program DirectoryIndicators

The last indicator is the `FileExtIndicators` which is shown in Listing 2.10. It contains information about files that are opened with the program because of their extension. This information can also be found in the registry under `HKLM\SOFTWARE\Classes`.

```
[...]
<Indicators>
  <FileExtIndicators>
    <FileExtensionHandler Extension=".5vw" Name="wireshark-capture-file" File="Wireshark.exe" UniqueId=
        "0xa6" Id="0000f100f0a810d3369fb23078ccfccf2a9ae2342793"/>
    <FileExtensionHandler Extension=".acp" Name="wireshark-capture-file" File="Wireshark.exe" UniqueId=
        "0xa8" Id="000053fad4b5cd3a257f5356040dd85aed5dfc94a972"/>
    <FileExtensionHandler Extension=".apc" Name="wireshark-capture-file" File="Wireshark.exe" UniqueId=
        "0xaa" Id="0000e60a8211bfa17705f2b5b3e6f79acbe7e80e8078"/>
    [...]
  </FileExtIndicators>
</Indicators>
[...]
```

2.10: Extract of AEINV_WER_{49A35C5F-CCE9-48C7-B6EF-577A36E86135}_20181206_094337.xml: Installed Program FileExtIndicators

Finally, after all the different indicators, the last sublist consists of all the PE files that are in the program directories. An extract is shown in Listings 2.11. The files are listed in the order of their `UniqueId`, which is the same as in the `DirectoryIndicators`. For instance, for Wireshark, capinfos.exe, which was the first PE file in the first folder, has a `UniqueId` of 0x24. Various pieces of information are recorded about the file, the most important being its SHA-1.

```
[...]
<ProgramList>
  <Installed>
    <Program Name="Wireshark 2.6.5 32-bit" Type="Application" Source="AddRemoveProgram" Publisher="The
        Wireshark developer community, https://www.wireshark.org" Version="2.6.5" OnSystemDrive="true"
        EvidenceId="0x22" Id="0000354384b2dbc2f6b2dc9dec22174dcf510000ffff">
      <Indicators>
        [...]
      </Indicators>
      <StaticProperties>
        <Files Id="00006ea5b5dae4e85c2b7a0ce4c0e609179961cd09fb">
          <File Name="capinfos.exe" Id="00005c5ecbf7d4e969ff50b186109b2c18b47f257365" ProductName="
              Capinfos" CompanyName="The Wireshark developer community" ProductVersion="2.6.5"
              VerLanguage="1033" SwitchBackContext="0x0100000000000600" FileVersion="2.6.5" Size="0
              x532a8" SizeOfImage="0x53000" PeHeaderHash="01012864b33151873a9ca2d4c0c5e28d87cfb023f0f3"
               PeChecksum="0x5fe24" BinProductVersion="2.6.5.0" BinFileVersion="2.6.5.0"
              FileDescription="Capinfos" LinkerVersion="14.12" LinkDate="11/28/2018 18:23:59"
              BinaryType="32BIT" Created="11/28/2018 18:31:44" Modified="11/28/2018 18:31:44"
              LongPathHash="0000058d47d0b218994a27e38ea102effc68e3b18ed3" UniqueId="0x24"/>
          <File Name="comerr32.dll" Id="00001c24f9e44091059fe4df7f37488104d9a84e62e2" ProductName="
              comerr32.dll" CompanyName="Massachusetts Institute of Technology." ProductVersion="1.6-
              kfw-3.2.2" VerLanguage="1033" SwitchBackContext="0x0100000000000400" FileVersion="1.6-kfw
              -3.2.2" Size="0xa4a8" SizeOfImage="0x7000" PeHeaderHash="0101627
              e686207f162c390be2477d9b676b6591217bc" PeChecksum="0x180bd" BinProductVersion="1.6.3.16"
              BinFileVersion="1.6.3.16" FileDescription="COM_ERR - Common Error Handler for MIT
              Kerberos v5 / GSS distribution" LinkerVersion="6.0" LinkDate="01/18/2010 17:01:38"
              BinaryType="UNKNOWN" Created="11/28/2018 18:31:44" Modified="11/28/2018 18:31:44"
              LongPathHash="0000b3d0ba5a55811478c8135b5addde46f63d1bde66" UniqueId="0x25"/>
          <File Name="d3dcompiler_47.dll" Id="0000ba29e74577085c41637b1ce7a14ea1853264417a" ProductName
              ="Microsoft® Windows® Operating System" CompanyName="Microsoft Corporation"
              ProductVersion="10.0.16299.15" VerLanguage="1033" ShortName="D3DCOM~1.DLL"
              SwitchBackContext="0x0100000000000600" FileVersion="10.0.16299.15(WinBuild.160101.0800)"
              Size="0x37d4a8" SizeOfImage="0x386000" PeHeaderHash="0101
              a7f2a4e9e1d7375b13562316f87244c2fa626053" PeChecksum="0x37e544" BinProductVersion="
              10.0.16299.15" BinFileVersion="10.0.16299.15" FileDescription="Direct3D HLSL Compiler for
               Redistribution" LinkerVersion="14.10" LinkDate="10/19/2047 09:23:28" BinaryType="UNKNOWN
              " Created="11/28/2018 18:31:44" Modified="11/28/2018 18:31:44" LongPathHash="0000
              ac76002f76c0ce9e6bdee7c392a8d6b246256a0f" UniqueId="0x26"/>
          [...]
        </Files>
      </StaticProperties>
    </Program>
    [...]
  </Installed>
  [...]
</ProgramList>
[...]
```

2.11: Extract of AEINV_WER_{49A35C5F-CCE9-48C7-B6EF-577A36E86135}_20181206_094337.xml: Installed Program Files

In addition to the structure of `AEINV_WER`, Appendix E contains an exhaustive description of its contents, in particular, all observed attributes appearing in the different program indicators.

Updated

The recorded information in the `Updated` sublist is the same as in the `Installed` sublist. This sublist is populated when a change occurs in one of the indicators previously mentioned. So for instance, if a new file appears inside a directory it is recorded in this sublist. As previously mentioned, all the PE files in the program folders are recorded in the Files sublist, regardless of whether they need shimming or whether they were executed. This is not limited to binaries that came with the program installation.

To test this, an experiment was made where the following scenario was carried out: a PE, malware.exe (which was a renamed cmd.exe for the experiment), was placed inside `C:\Program Files\Wireshark\diameter`. Then at the next execution of `ProgramDataUpdater`, the change was recorded in `AEINV_WER`, even though Wireshark was not executed in the meantime. The entry in the `Installed` list did not change, although the `UniqueId` is now incorrect due to having one more exe file in a folder. However, an entry for Wireshark appeared in the `Updated` list with the following differences:

- The `EvidenceId` changed to 0x75;
- All the `UniqueId` values changed in accordance with the new `EvidenceId`;
- There was one more entry in `DirectoryIndicators`, since the folder `diameter` did not previously contain a PE file;
- There was a new entry in the `Files` list corresponding to a file named malware.exe. This entry can be found in Listing 2.12

```
[...]
<File Name="malware.exe" Id="0000ee8cbf12d87c4d388f09b4f69bed2e91682920b5" ProductName="Microsoft®
    Windows® Operating System" CompanyName="Microsoft Corporation" ProductVersion="6.1.7601.17514"
    VerLanguage="1033" SwitchBackContext="0x0100000000000601" FileVersion="6.1.7601.17514(win7sp1_rtm
    .101119-1850)" Size="0x49e00" SizeOfImage="0x4c000" PeHeaderHash="01013
    fb8cef24089e6b61cb1bf72c61e223aff261414" PeChecksum="0x57b3d" BinProductVersion="6.1.7601.17514"
    BinFileVersion="6.1.7601.17514" FileDescription="Windows Command Processor" LinkerVersion="9.0"
    LinkDate="11/20/2010 09:00:27" BinaryType="32BIT" Created="12/17/2018 09:43:41" Modified="
    11/20/2010 21:29:12" LongPathHash="00005580c7b910d3e448614e137f71c66fb7aed463de" UniqueId="0xbb"/>
[...]
```

2.12: Extract of AEINV_WER_{49A35C5F-CCE9-48C7-B6EF-577A36E86135}_20181206_094337.xml: malware.exe

As such, by comparing the `Files` of the `Updated` and `Installed` list, an analyst could pinpoint malware.exe, which is present in the former but not in the latter. This is especially useful, since there should not be a lot of modifications in the binaries under those folders.

It is worth noting that, if the version number of the program changes (during an upgrade for example), it is not considered an update but a removal of the previous program followed by an installation of the newer version. This leads to two entries in the `Installed` sublist (one for each version) and one in the `Removed` one.

Removed

The removed sublist only records the program id, name, publisher, version and source of the removed program. The list `StaticProperties` is also present but does not list every PE that used to be in the installation folder (however, this information can be retrieved in the program entry in the `Installed` sublist). An example is shown in Listing 2.13.

```
[...]
<Removed>
  <Program Id="0000354384b2dbc2f6b2dc9dec22174dcf510000ffff" Name="Wireshark 2.6.5 32-bit" Publisher="
      The Wireshark developer community, https://www.wireshark.org" Version="2.6.5" Source="
      AddRemoveProgram">
    <StaticProperties>
      <Files Id="0000ff88d3b106df1081a63003d3568f3fccf14c63cd"/>
    </StaticProperties>
  </Program>
</Removed>
[...]
```

2.13: Extract of AEINV_WER_{49A35C5F-CCE9-48C7-B6EF-577A36E86135}_20181206_094337.xml: Removed

Orphan

Finally, the `Orphan` sublist records executables that were listed in `RecentFileCache.bcf` but do not belong to a program, in the sense that they are not part of a program indicator. As an example, the entries for fsstat.exe and tree.com are shown in Listing 2.14.

```
[...]
<Program Name="fsstat.exe" Type="BOE" Source="File" OnSystemDrive="true" EvidenceId="0xa" Id="0003
    dbbf37cd35b9cda595baeb6211ae4bc60000ffff">
  <Indicators></Indicators>
  <StaticProperties>
    <Files Id="0000e88f46ad7600cdf0ccc84b810188cc03ccce4253">
      <File Name="api-ms-win-core-heap-l1-1-0.dll" Id="0000c7ce2330265d07d88ad15f80dd88473f3daafcd0"
          ProductName="Microsoft® Windows® Operating System" CompanyName="Microsoft Corporation"
          ProductVersion="10.0.10240.16384" VerLanguage="1033" ShortName="API-MS~1.DLL"
          SwitchBackContext="0x0100000000000A00" FileVersion="10.0.10240.16384(th1.150709-1700)" Size="
          0x4ac0" SizeOfImage="0x3000" PeHeaderHash="01014d5d81337e4e036a1047b8ac4196852ce574dc9c"
          PeChecksum="0x11bf5" BinProductVersion="10.0.10240.16384" BinFileVersion="10.0.10240.16384"
```

```
            FileDescription="ApiSet Stub DLL" LinkerVersion="12.10" LinkDate="07/10/2015 03:22:43"
            BinaryType="UNKNOWN" Created="12/10/2018 09:22:08" Modified="11/09/2018 09:57:18"
            LongPathHash="0000770c4f7e744ac041a8aea78bd42b74e1fa96ed96" UniqueId="0x13"/>
        [...]
        <File Name="fsstat.exe" Id="000089d756cdffbda5c9ce341c2d69a6edb87e9048f3" SwitchBackContext="0
            x0100000000000501" Size="0x7fc00" SizeOfImage="0x84000" PeHeaderHash="0101
            ed6e92b581121cb94845a4cf984586731cf526c2" PeChecksum="0x0" LinkerVersion="14.0" LinkDate="
            11/09/2018 15:57:11" BinaryType="32BIT" Created="12/10/2018 09:22:09" Modified="11/09/2018 09
            :57:18" LongPathHash="00005e25446d153b03778abf45fc1371bb2ec43b2a27" UniqueId="0xa"/>
      </Files>
    </StaticProperties>
</Program>
[...]
<Program Name="Microsoft Windows Operating System" Type="BOE" Source="File" Publisher="Microsoft
    Corporation" Version="0.0.0.0" Language="0" OnSystemDrive="true" EvidenceId="0xa" Id="0000
    f519feec486de87ed73cb92d3cac802400000000">
    <Indicators></Indicators>
    <StaticProperties>
      <Files Id="000099c3139497239b9f697cb73a65ce7d423a980bee">
        <File Name="tree.com" Id="00006b5d28546f358715844fd9946a8785db5df533ba" ProductName="Microsoft®
            Windows® Operating System" CompanyName="Microsoft Corporation" ProductVersion="6.1.7600.16385
            " VerLanguage="1033" SwitchBackContext="0x0100000000000601" FileVersion="6.1.7600.16385(
            win7_rtm.090713-1255)" Size="0x4000" SizeOfImage="0x7000" PeHeaderHash="01013089399
            f8d2893e8207a55cf81f51bb14a20c8f4" PeChecksum="0x13156" BinProductVersion="6.1.7600.16385"
            BinFileVersion="6.1.7600.16385" FileDescription="Tree Walk Utility" LinkerVersion="9.0"
            LinkDate="07/13/2009 23:15:24" BinaryType="32BIT" OsComponent="true" Created="07/13/2009 23
            :15:24" Modified="07/13/2009 23:15:24" LongPathHash="00002
            b75b5af4abe6066802a040875c5b8e4d0ae4408" UniqueId="0xa"/>
      </Files>
    </StaticProperties>
</Program>
[...]
```

2.14: Extract of AEINV_WER_{49A35C5F-CCE9-48C7-B6EF-577A36E86135}_20181206_094337.xml: Orphan

These entries follow the same structure as those in the other sublists except that the program name is either the `Product Name` of the executables if it has one (like tree.com), or the file name if it does not (like fsstat.exe). In this example, some DLLs are also registered in the `Files` sublist for fsstat.exe. The reason that some DLLs get recorded and others do not is not yet known. Indeed, the recorded DLLs are not in the import table of the PE and the sublist does not contain every DLL in the folder where the PE is. Moreover, the `StaticProperties` section records the PE (and associated DLLs) with the same information as in the `Installed` sublist (name, SHA-1, size, ...).

2.5. Examples of possible uses during a forensic investigation

During a forensic investigation, `RecentFileCache.bcf`, `AEINV_PREVIOUS.xml` and `AEINV_WER` can significantly help an analyst. The three files can be used to track executed binaries, installed and deleted programs and the content of an installation folder.

To prove that a binary was executed, an analyst can look at both `RecentFileCache.bcf` and `AEINV_WER`. If the binary is present in `RecentFileCache.bcf`, then it was first executed between the last run of `ProgramDataUpdater` and the current time. If the binary is present in the `Orphan` list of `AEINV_WER`, then it was first executed before the last run of `ProgramDataUpdater` and the analyst can also retrieve important information such as the SHA-1 and times of creation and modification of the binary, as shown in Section 2.4.3. These pieces of information, however, are recorded only once, after the *first* execution of a PE stored in a given path, which means that if an attacker replaces the PE with another one, `AEINV_WER` is not updated.

By studying `AEINV_PREVIOUS.xml` and `AEINV_WER`, it is possible to determine which programs were installed on the system when `ProgramDataUpdater` was last executed. The analyst can also retrieve which programs were uninstalled when inspecting the sublist `Removed` of `AEINV_WER`: those are programs removed before the last launch of `ProgramDataUpdater`.

Finally, `AEINV_WER` records new additions made to a program installation folder, as explained in Section 2.4.3. Indeed, PE files under an installation folder are recorded in the list `StaticProperties` of a program entry. When a new PE appears under one of those folders, it creates a new entry for the program in the `Updated` list but leaves the one originally in `Installed` untouched. By comparing the different `StaticProperties` of this program, the analyst can look for PEs that appeared under an installation folder after the installation and retrieve their SHA-1. This search is easily automated, and should not yield many false positives.

3. Behavior of libraries originally packaged with Windows 8.0 and Server 2012

This chapter describes the behavior of the version 6.2.9200.16384 of the libraries, shipped with Windows 8 "out of the box". This version comes with a major change : the file `RecentFileCache.bcf` no longer exists and the information it contained is now stored in `AmCache.hve`, a registry file. It is worth noting that if libraries in this version run on a machine as a result of an update of a system, the previous artifacts may still be found and operational on the new system. This entails that if the investigated system is a Windows 7, it is possible to have both the `AmCache.hve` and the `RecentFileCache.bcf` files.

3.1. General behavior

When executing a PE, the service `AeLookupSvc`, which executes "`%WinDir%\system32\svchost.exe -k netsvcs`", checks whether the PE needs shimming. If it does, the service stores information about the PE in a registry file named `AmCache.hve`, located under `%WinDir%\AppCompat\Programs`. However, if the executed PE is an installer for a program, it is handled by a different service : `PCASvc`, which executes `%WinDir%\system32\svchost.exe -k LocalSystemNetworkRestricted`. This service calls the following command : "`rundll32.exe aeinv.dll, UpdateSoftwareInventory`". This DLL creates a TXT file in `%WinDir%\AppCompat\Programs\Install` which is then rewritten into an XML file under the same directory. This XML file records the installation process. Moreover, it updates `AmCache.hve` with information about the newly installed program and the files the installation created.

Unlike the previous versions, the scheduled task `ProgramDataUpdater` is now a maintenance task, which means that it runs automatically when the computer is in idle state starting at 3AM. The settings of this task makes it run once every 3 days (parameter `Period = P3D`) with other maintenance tasks. If the task fails for 6 days (parameter `Deadline = P6D`), the user is notified or an emergency maintenance is performed. This task launches "`%WinDir%\system32\rundll32.exe aepdu.dll,AePduRunUpdate`" which deletes all the files under `%WinDir%\ AppCompat\Programs\Install\` and stores all installed programs in `%WinDir%\AppCompat\Programs\AEINV_CURRENT.xml`. It then renames this file as `AEINV_PREVIOUS.xml`, overwriting the previous file. `ProgramDataUpdater` also updates the information contained in `AmCache.hve`, `%WinDir%\AppCompat\Programs\AEINV_AMI_WER_{MachineId}_YYYYMMDD_HHmmss.xml` and, if needed (e.g. if a driver was installed), updates `%WinDir%\AppCompat\Programs\DevInvCache\PropCache.bin`.

Since `AEINV_PREVIOUS.xml` has the same structure detailed in Section 2.3, it is not described in further sections.

3.2. AmCache.hve

Starting with this version, Microsoft stores information about shimmed PEs and installed applications in a registry file. This implies that information described below may only be present in transaction logs and not yet in the registry file itself. At the root of this registry is a key called `Root`. This key contains four subkeys: `File`, `Programs`, `Orphan` and `Generic` and a value `Sync` which is a FILETIME timestamp and is the last date and time the `ProgramDataUpdater` has been launched, in UTC. The four subkeys are described in details below, starting with `File`.

3.2.1. File

This key is split into several subkeys, each representing a volume GUID. A volume GUID key contains subkeys that each represent a PE. For an NTFS volume, the name of the PE key is the MFT Sequence Number appended to the MFT Entry Number (prefix-padded to be 8 bytes long), both in hexadecimal, as found by Yogesh Khatri in [3]. He also found that for a FAT volume, the name of the PE key is the byte offset of the Directory Entry.

As an example, on an NTFS volume, the key `Root\File\b528e029-0e73-11e9-af9b-806e6f6e6963\50000f99c` describes Wireshark.exe and the record in the MFT for the same file is shown in Fig. 3.1. The Sequence Number and MFT Entry for Wireshark are respectively 5 (0x5) and 63900 (0xf99c). Since the MFT Entry must be padded to be 8-bytes long, it results in a FileID of 50000f99c.

Much like the `Files` sublist previously seen in the `AEINV_WER`, each PE key contains information about the PE file but the content seems to differ depending on whether the PE is part of a program. Indeed, if the PE is part of a program, meaning it is under the installation directory of a program, it usually contains about four or five values whereas if the PE is "orphan", it usually contains about twenty values. As an example, the entry for Wireshark.exe (which is part of a program) is shown in Fig. 3.2 and the entry for fsstat.exe (which is considered "orphan") is shown in Fig. 3.3.

The information found in those two keys is similar to what was found in `AEINV_WER` described in Section 2.4. The values have the same meaning whether the key exhibits four or twenty values and are as follows:

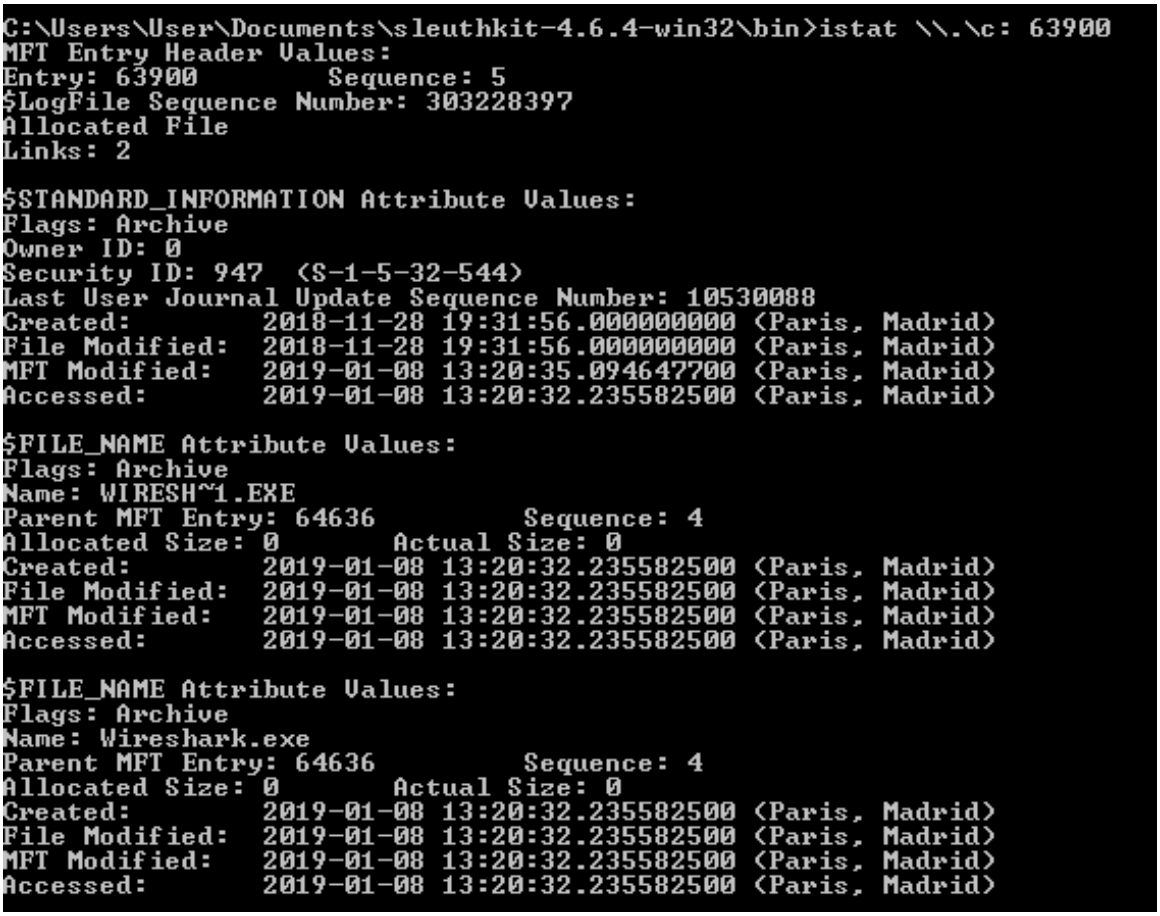
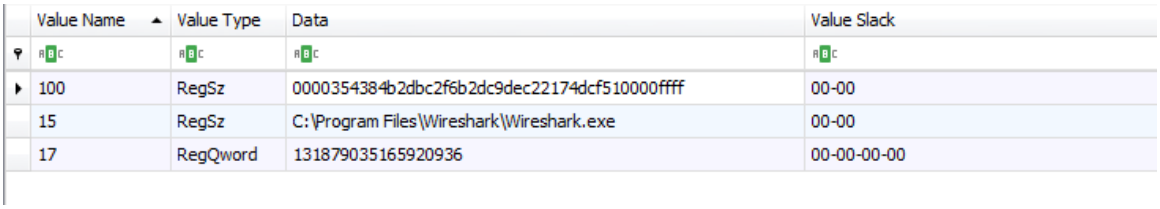


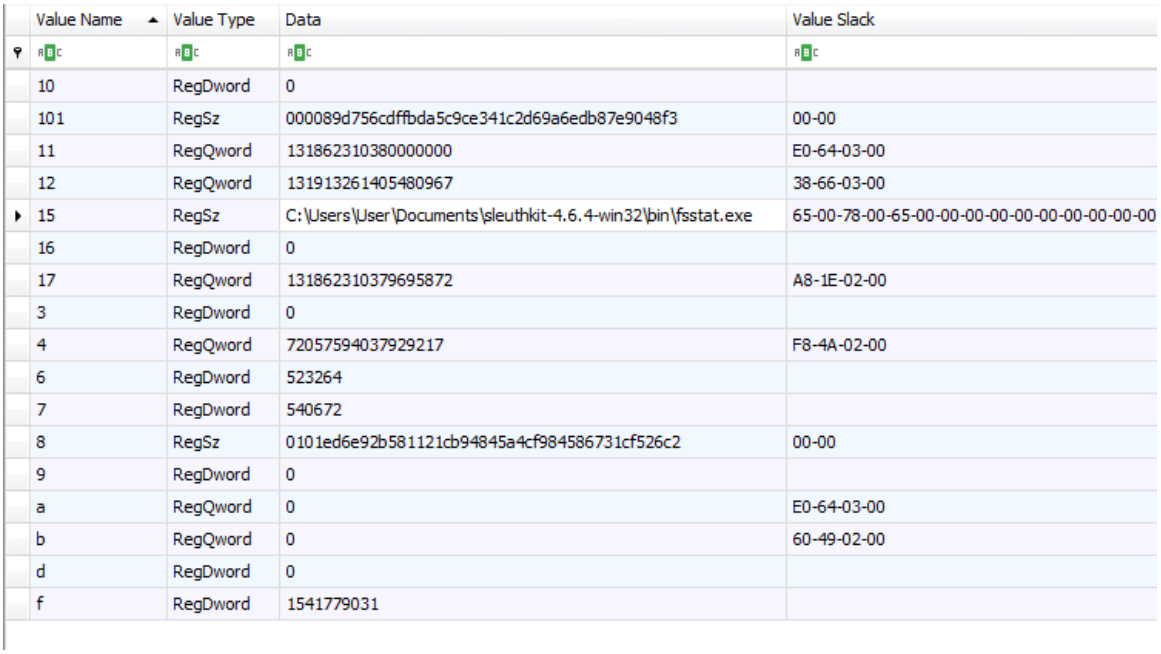
Fig. 3.1.: MFT Entry for C:\Program Files\Wireshark\Wireshark.exe



Value Name	Value Type	Data	Value Slack
100	RegSz	0000354384b2dbc2f6b2dc9dec22174dcf510000ffff	00-00
15	RegSz	C:\Program Files\Wireshark\Wireshark.exe	00-00
17	RegQword	131879035165920936	00-00-00-00

Fig. 3.2.: Content of `Root\File\b528e029-0e73-11e9-af9b-806e6f6e6963\50000f99c`

- 10 = Unknown;
- 100 = ProgramId. This information was previously found in the attribute `Id` of the `Program` the PE belonged to in `AEINV_WER`;
- 101 = SHA-1 preceded by '0000'. This information was previously found in the attribute `Id` of the PE in the list `Files` in `AEINV_WER`;
- 11 = FILETIME timestamp that seems to be either the date of modification or a few seconds after;
- 12 = The date of creation in the FILETIME timestamp format. This information was previously found in the attribute `Created` of the PE in the list `Files` in `AEINV_WER`;
- 15 = The full path of the PE;
- 16 = Unknown;
- 17 = The date of modification in the FILETIME timestamp format. This information was previously found in the attribute `Modified` of the PE in the list `Files` in `AEINV_WER`;
- 3 = Microsoft's corresponding Language Id, in decimal. This information was previously found in the attribute `VerLanguage` of the PE in the list `Files` in `AEINV_WER`;
- 4 = The SwitchBackContext. This information was previously found in the attribute `SwitchBackContext` of the PE in the list `Files` in `AEINV_WER`, only it was in hexadecimal;



Value Name	Value Type	Data	Value Slack
10	RegDword	0	
101	RegSz	000089d756cdffbda5c9ce341c2d69a6edb87e9048f3	00-00
11	RegQword	131862310380000000	E0-64-03-00
12	RegQword	131913261405480967	38-66-03-00
15	RegSz	C:\Users\User\Documents\sleuthkit-4.6.4-win32\bin\fsstat.exe	65-00-78-00-65-00-00-00-00-00-00-00-00-00-00-00
16	RegDword	0	
17	RegQword	131862310379695872	A8-1E-02-00
3	RegDword	0	
4	RegQword	72057594037929217	F8-4A-02-00
6	RegDword	523264	
7	RegDword	540672	
8	RegSz	0101ed6e92b581121cb94845a4cf984586731cf526c2	00-00
9	RegDword	0	
a	RegQword	0	E0-64-03-00
b	RegQword	0	60-49-02-00
d	RegDword	0	
f	RegDword	1541779031	

Fig. 3.3.: Content of `Root\File\b528e029-0e73-11e9-af9b-806e6f6e6963\10000fb80`

- 6 = The size. This information was previously found in the attribute `Size` of the PE in the list `Files` in `AEINV_WER`, only it was in hexadecimal;
- 7 = The SizeOfImage. This information was previously found in the attribute `SizeOfImage` of the PE in the list `Files` in `AEINV_WER`, only it was in hexadecimal;
- 8 = The PeHeaderHash. This information was previously found in the attribute `PeHeaderHash` of the PE in the list `Files` in `AEINV_WER`;
- 9 = The PE header checksum. This information was previously found in the attribute `PeChecksum` of the PE in the list `Files` in `AEINV_WER`, only it was in hexadecimal;
- a = Unknown, although when the value is present, it seems to be 0 for unsigned PE and something else for signed ones;
- b = Unknown, although when the value is present, it seems to be 0 for unsigned PE and something else for signed ones (usually the same as `a`);
- d = Concatenation of the `MajorImageVersion` and `MinorImageVersion` as found in the PE optional header and converted to decimal;
- f = Compilation date in the UNIX timestamp format. This information was previously found in the attribute `LinkDate` of the PE in the list `Files` in `AEINV_WER`.

Since several services interact with `AmCache.hve`, and especially with the `File` key, the meaning of the last write time of this key is difficult to interpret. During tests, `ProgramDataUpdater` seemed to only update keys to fill in the value 100 (`ProgramId`) and 101 (`SHA-1`) if they are empty. The first value is often missing for setup and orphan executables. The second value is always missing for PEs that are part of a program and that were not executed or did not need shimming when executed. The following algorithm comes from running multiple tests rather than code analysis and should not be considered as the immutable truth:

- if the PE is part of a program:
  - if the PE needed shimming and was executed before `ProgramDataUpdater` had a chance to run: the last write time seems to be the time of execution of the PE;
  - else, if `ProgramDataUpdater` was executed since the installation of the program: the last write time seems to be the time `ProgramDataUpdater` was first run after the execution of the PE;
  - finally, if neither of those cases apply, the last write time seems to be the time of installation of the program.
- if the PE is part of a setup for a program (for example, Wireshark-win32-2.6.5.exe):

  - if `ProgramDataUpdater` was launched since the execution of the PE: the last write time seems to be the time `ProgramDataUpdater` was first run after the execution of the PE;
  - else, the last write time seems to be the time the PE was executed.

- if the PE is part of the system (i.e. its `ProductName` is "Microsoft Windows Operating System"):
  - the last write time does not seem to correspond to anything: it is neither the first nor last time the PE was executed, it is not the time of a launch of `ProgramDataUpdater` and nothing in the event logs could help define what the time was.

- if neither of these cases apply:
  - if the PE had no value `100` associated with it and `ProgramDataUpdater` was launched since the execution of the PE: the last write time seems to be the time `ProgramDataUpdater` was first run after the execution of the PE;
  - else, the last write time seems to be the time the PE was executed.

Eventually, it is important to note that appearance in this subkey does not necessarily mean that the PE was executed since all PEs under an installation folder are present. Furthermore, if the execution is proven via another artifact or because the PE is orphaned, the last write time of the key associated with the PE should be considered as an upper bound to the execution time rather than the execution time itself.

3.2.2. Programs

This key contains every installed program only in this case, the definition of "installed program" is slightly different from the one described in Section 2.3: only the programs which have an entry under `SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall` or `SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall` are recorded, the `Run` key is no longer parsed. Each subkey corresponds to a `ProgramId`. The subkey corresponding to Wireshark 2.6.5 is shown in Fig 3.4.

Value Name	Value Type	Data	Value Slack
0	RegSz	Wireshark 2.6.5 32-bit	00-00-00-00-00-00
1	RegSz	2.6.5	
13	RegDword	0	
2	RegSz	The Wireshark developer community, https://www.wireshark.org	00-00
3	RegSz		
5	RegDword	256	
6	RegSz	AddRemoveProgram	00-00
7	RegMultiSz	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall\Wireshark	
a	RegQword	1546950057	50-43-02-00
b	RegQword	0	50-43-02-00
d	RegMultiSz	C:\Program Files\Wireshark C:\Program Files\Wireshark\audio C:\Program Files\Wireshark\bear...	35-00-32-00-38-00
Files	RegMultiSz	b528e029-0e73-11e9-af9b-806e6f6e6963@20000fbcf b528e029-0e73-11e9-af9b-806e6f6e6...	00-00-00-00-00-00

Fig. 3.4.: Content of `AmCache.hve\Root\Programs\0000354384b2dbc2f6b2dc9dec22174dcf510000ffff`

The information found in a subkey is similar to what was found in `AEINV_PREVIOUS.xml` and `AEINV_WER` in the previous version of the libraries. The values are as follows:

- 0 = The name of the program. This information was previously found in the attribute `Name` of the program header in `AEINV_WER`;
- 1 = The version of the program. This information was previously found in the attribute `Version` of the program header in `AEINV_WER`;
- 13 = Unknown;
- 2 = The publisher of the program. This information was previously found in the attribute `Publisher` of the program header in `AEINV_WER`;
- 3 = Unknown;
- 5 = Unknown;
- 6 = The installation method of the program, previously found in the attribute `Source` of the program header in `AEINV_WER`;

- 7 = The uninstall key of the program, previously found in the list `AddRemoveProgramIndicators` in `AEINV_WER`;
- a = The installation date of the program, in the Unix timestamp format;
- b = The uninstallation date of the program, in the Unix timestamp format, or `0` if the program is still installed;
- d = The installation folder of the program and its subfolders if they contain PEs. This information was previously found in the list `DirectoryIndicators` in `AEINV_WER`;
- Files = The PEs that were created following the installation of the program, meaning the PEs in the installation folder, but also for example drivers that were created in `C:\Windows\System32\Drivers`,... The structure of the data contained in this value is a list of `VolumeGUID@FileID`, where `VolumeGUID` and `FileID` are determined as described in Subsection 3.2.1 for the `File` key. Part of this information (only the PEs under the installation folder) was previously found in the list `StaticProperties` in `AEINV_WER`.

When installing an MSI program, four additional keys can also be present:

- 11 = MSI Product Code. This information was previously found in the attribute `MsiProductCode` of the program header in `AEINV_WER`;
- 12 = MSI Package Code. This information was previously found in the attribute `MsiPackageCode` of the program header in `AEINV_WER`;
- f = Product Code. In tests, this information always had the same value of `11`;
- 10 = Package Code. In tests, this information always had the same value of `12`.

3.2.3. Orphan

As in `AEINV_WER`, this key records executed PEs that are not part of a program. The format of the subkeys is `VolumeGUID@FileID`, where `VolumeGUID` and `FileID` are determined as described in 3.2.1 for the `File` key. Each subkey only contains one value, `c`, which is either 0 or 1. It seems that the value 0 means either that the associated `File` key does not have a `ProgramId` (value `100`) or that the entry has been added after the last execution of `ProgramDataUpdater`.

3.2.4. Generic

The Generic key contains one subkey named `0`, which in turn contains one subkey per driver installed on the system. Each of these subkeys is actually named as the SHA-1 of the driver it represents, preceded by '0000'. Under each of these subkeys, there seems to always be a value named `0` and worth `1`. An example is shown in Fig. 3.5, the entry for the SHA-1 of the driver named 1394ohci.sys.

	Value Name	Value Type	Data	Value Slack
▼	ABC	ABC	ABC	ABC
▶	0	RegDword	1	

Fig. 3.5.: Content of `AmCache.hve\Root\Generic\0\000002da97a4940b126c7710d13b431a6e74123f3cc0`

At the same level of the SHA-1 subkeys are keys with names that resemble a GUID and that also only have one value named `0` with associated data `1`. Those names are actually values of `DeviceModelId` of `DeviceContainers`. Since there are more details about `DeviceContainers` in `AEINV_AMI_WER`, they are explained in the corresponding section.

3.3. Install Directory

This folder contains an XML file for each program installed with an exe file. An example of the XML file for the installation of Wireshark 2.6.5 can be found in Listing 3.1.

```
<Installer CompletionState="1" CreatedArpEntries="1" StartTime="01/03/2019 14:34:28" StopTime="
    01/03/2019 14:36:31">
```

```
<InstallInfo Name="Wireshark-win64-2.6.5.exe" Path="C:\Users\User\Downloads" ShortName="WIRESH~1.EXE"
     OsComponent="false" Size="0x38c77a0" PeHeaderHash="010169294005c024647938d49d9afaf3ff93485269f7"
     SizeOfImage="0x7b000" PeChecksum="0x38cd75f" LinkDate="12/11/2016 21:50:45" LinkerVersion="6.0"
    BinFileVersion="2.6.5.0" BinProductVersion="2.6.5.0" BinaryType="32BIT" Created="01/03/2019 14
    :34:10" Modified="01/03/2019 14:31:17" VerLanguage="1033" Id="000015
    c2819075563b46e8a1a5cc49ee09daffcf85ce" SwitchBackContext="0x0100000006020400" SigPublisherName="
    Wireshark Foundation, Inc." FileVersion="2.6.5.0" CompanyName="Wireshark development team"
    FileDescription="Wireshark installer for 64-bit Windows" LegalCopyright="© Gerald Combs and many
    others" ProductName="Wireshark" LongPathHash="0000e114c123b71a9e0b27f56b4f9b974841b2961b43"/>
<DiscInfo/>
<ProgramIds>
  <ProgramId Id="0000a5c8d73a8a4913750a2b7678f38ef28a0000ffff"/>
  <ProgramId Id="0000c16b47f8ca21d3ca3f3ace1abb7c51e40000ffff"/>
</ProgramIds>
</Installer>
```

3.1: Content of INSTALL_ffff_6f6309c6-c56f-4e93-a6b1-b95cc246b8fb.xml

The `INSTALL` file starts with a header that indicates whether the installation was successful. The header contains 4 attributes :

- `CompletionState` = 1 if the installation was successful;
- `CreatedArpEntries` = 1 if the installation led to the creation of an `Uninstall` key in the `SOFTWARE` hive;
- `StartTime` = the timestamp, in UTC, of when the installation started. This could be interpreted as the time of execution of the setup binary;
- `StopTime` = the timestamp, in UTC, of when the installation process stopped, whether it succeeded or not.

Inside the `Installer` element, 3 sub-elements can be found. The first one, `InstallInfo`, stores information about the setup binary of the program. The different attributes are similar to what was previously recorded in other XML files such as `AEINV_WER`. The new attributes are :

- `Path` = The path of the file, case sensitive;
- `OsComponent` = Whether the PE is part of the OS;
- `SigPublisherName`;
- `LegalCopyright`.

The second sub-element, `DiscInfo`, contains information about the disc the setup binary was stored on, if there was one. Since there was none for Wireshark, the element is empty. The third and last sub-element is a list of `ProgramId` entries that stores the programs that were installed following the execution of the setup binary. In this example, the two entries are respectively Microsoft Visual C++ 2017 and Wireshark. In another test, Wireshark was installed along with WinPCAP, which is an option when installing Wireshark, resulting in an additional `ProgramId` entry in this list.

As an example of a program installed with a disc, the install file for the Virtual Box Guest Additions is shown in Listing 3.2.

```
<DiscInfo Name="VBOXADDITIONS_5." Id="0004021d62bcd80dc4a5ac67b8fbfdb91516395084b5" SetupScriptChecksum
    ="17231136449290210510" Size="58466304">
</DiscInfo>
```

3.2: Extract of INSTALL_ffff_6f6309c6-c56f-4e93-a6b1-b95cc246b8fb.xml

The attribute `Name` is the name of the disc. The `Size` is equal to the IpTotalNumberOfBytes parameter from the GetDiskFreeSpaceEx method of kernel32.dll.

3.4. AEINV_AMI_WER_{MachineId}_YYYYMMDD_HHmmss.xml

`AEINV_AMI_WER` contains eight sublists, three of which have been seen previously in `AEINV_WER` : `System`, `ProgramList` and `IEAddOn`. The only difference that could be found between `AEINV_WER` and `AEINV_AMI_WER` in those three sub-lists is that in `AEINV_AMI_WER` `ProgramList`, there never seems to be an `Updated` or `Removed` list: once the program is recorded in `Installed`, it is never removed or updated. As a consequence, only the new sublists are described below. The reader is invited to refer to Appendix F where the structure of the file is outlined. It can help follow detailed explanations given below.

3.4.1. InstallerList

This list contains the information in each `INSTALL` XML file, described in 3.3, word for word, so it is not redescribed in this section.

3.4.2. DeviceList

This list contains several entries named `DeviceContainer`. According to the Microsoft docs[1], a device container is an instance of a physical device that was plugged on a system. Since no relation could be found between this list and either PE execution or program installation, it was not studied in depth. Although and since it is valuable in a forensic examination, it is interesting to note that proof of USB usage could be found in this list, such as the example given in Listing 3.3.

```
<DeviceContainer DeviceModelId="{776d907e-05ed-7eb0-0ef7-6dff88ee1a34}" DeviceDataId="{1aedc93f-bfeb-9
    f36-826e-71bf7ee6fdfe}" ModelId="{2bda71a3-65a7-1c33-dd60-e2630bc8452b}" ModelName="USB DISK 2.0"
    IsMachineDevice="false" PrimaryCategory="storage">
  <Categories>
    <Category Id="storage"></Category>
  </Categories>
  [...]
  <Device DeviceId="{703078fc-7727-3141-ff51-e7c3fc5f5a21}" Enumerator="usb" DeviceOrder="0">
    <HardwareIds>
      <HardwareId Id="usb\vid_13fe&pid_4200&rev_0100" Order="0"></HardwareId>
      <HardwareId Id="usb\vid_13fe&pid_4200" Order="1"></HardwareId>
    </HardwareIds>
    <CompatibleIds>
      <CompatibleId Id="usb\class_08&subclass_06&prot_50" Order="0"></CompatibleId>
      <CompatibleId Id="usb\class_08&subclass_06" Order="1"></CompatibleId>
      <CompatibleId Id="usb\class_08" Order="2"></CompatibleId>
    </CompatibleIds>
  </Device>
  [...]
</DeviceContainer>
```

3.3: Extract of AEINV_AMI_WER_{0516712F-1ED3-44C1-A930-029F1AC8489F}_20180314_082618.xml

This `DeviceContainer` entry lists the same information as the different `Enum` registry keys related to the USB stick that was plugged in (`STORAGE`, `USB`, `USBSTORE`, `SWD`). The registry entry for `Enum\USB` is shown in Fig. 3.6 for comparison.

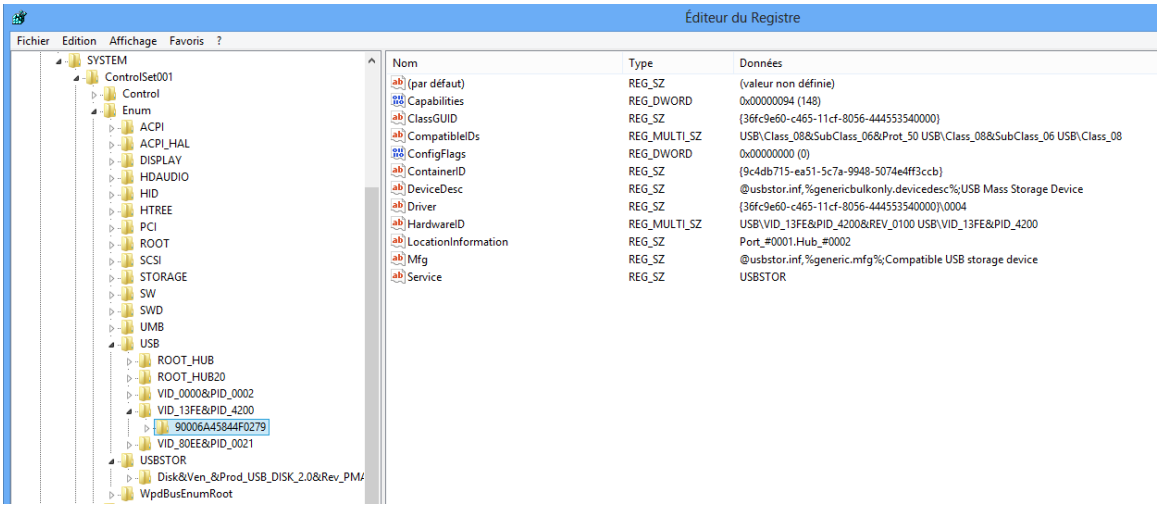


Fig. 3.6.: Content of `SYSTEM\ControlSet001\Enum\USB\vid_13fe&pid_4200`

3.4.3. DriverList

`DriverList` records the exhaustive list of installed drivers. An example for 1394ohci.sys is shown in Listing 3.4

```
<Driver DriverId="000002da97a4940b126c7710d13b431a6e74123f3cc0" Name="1394ohci.sys" Type="0x0004001a"
    Version="6.2.9200.16384" TimeStamp="0x5010aae6" CheckSum="0x00047021" ImageSize="0x0003d000"
    PagedSize="0x00000e00" Company="Microsoft Corporation" Product="Microsoft® Windows® Operating
    System" ProductVersion="6.2.9200.16384">
```

[1] `https://docs.microsoft.com/en-us/windows-hardware/drivers/install/container-ids`

```
</Driver>
```

3.4: Extract of AEINV_AMI_WER_{0516712F-1ED3-44C1-A930-029F1AC8489F}_20180314_082618.xml

The meaning of the different attributes are as follows:

- DriverId = SHA-1 of the driver, preceded by '0000';
- Name = Filename;
- Type = bitfield of driver attributes, explained in the Microsoft docs[2];
- Version;
- TimeStamp = Date of compilation in UNIX timestamp format, in hexadecimal;
- CheckSum;
- ImageSize;
- PagedSize;
- Company;
- Product;
- ProductVersion.

3.4.4. DriverPackageList and AitAnalysis

During tests, those two lists were always empty and it is not known if they sometimes contain information, and if so of what kind.

3.5. PropCache.bin

This file contains the same kind of information about drivers installed on the system as `DriverList`: Name, SHA-1, Version.... But it also contains information about the certificate used to sign the driver, such as its location on the system and the signer.

The structure of this file is described in Appendix G.

3.6. Examples of possible uses during a forensic investigation

On a system using version 6.2.9200.16384 of the libraries, files `AmCache.hve`, `AEINV_AMI_WER`, `AEINV_PREVIOUS`, `PropCache.bin` and `INSTALL` files available in `%WinDir%\AppCompat\Programs\Install` can be put to good use.

The appearance of a binary in the `File` key in `AmCache.hve` is not sufficient to prove binary execution but does prove the presence of the file on the system. Indeed, files related to a program are also listed in this key. However, when a binary is referenced under the `Orphan` key, it means that it was actually executed. In a similar manner, binaries listed under the `Orphan` list of `AEINV_AMI_WER` were executed. Conclusions about execution time are difficult to draw as explained in Subsection 3.2.1. As for the `AEINV_AMI_WER`, the referenced binaries were executed before the last run of `ProgramDataUpdater`.

Installed programs are indexed both in `AmCache.hve` and `AEINV_AMI_WER`. In the hive file, the date of installation appears in the value `a` of the program entry. In `AEINV_AMI_WER`, programs show up if they were installed before the last run of `ProgramDataUpdater`. If they were installed after, information about the program can be found in an `INSTALL` file.

Removed programs are only present in `AmCache.hve`, with the date of uninstallation in the value `b` of the program entry.

Although the `Updated` list no longer exists in `AEINV_AMI_WER`, pinpointing a new file inside an installation folder remains possible. Both `AmCache.hve` and `AEINV_AMI_WER` record the PE files present under an installation folder around the time of installation. Since this list is never updated, one can compare this list with the PEs currently in the folder.

Starting with this version, the AmCache can also be used to prove the presence of a driver on a system. The list of installed drivers and the information related to them can be found both in `AEINV_AMI_WER` and `PropCache.bin`, while only the SHA-1 of the drivers are present in `AmCache.hve`.

[2] https://docs.microsoft.com/en-us/windows/privacy/basic-level-diagnostic-events-and-fields-1709#microsoftwindowsinventorycoreinventorydriverbinaryadd

Eventually, it is worth noting that plugged-in devices (including USB sticks) are recorded in `AEINV_AMI_WER`, provided they were plugged in before the last run of `ProgramDataUpdater`.

4. Behavior of libraries originally packaged with Windows 8.1 and Server 2012 R2

For Windows 8.1, two versions of the DLLs were found. The first one, 6.3.9600.16384, exhibits no behavioral difference from the previous version described in Chapter 3, except that the `DriverPackageList` in `AEINV_AMI_WER` is not empty. Since this behavior is preserved by the next version of the library, 6.3.9600.17415, only changes introduced by this latter version are described here. This version comes with two changes : there is a new XML file and a new scheduled task. Otherwise, all the other files seen in Chapter 3 are still present on the system.

4.1. General behavior

When executing a PE, this version behaves almost like the previous one, whose behavior was described in Section 3.1. The difference is that there is a new scheduled task, `Microsoft Compatibility Appraiser`, which launches "`%WinDir%\system32\rundll32.exe aepdu.dll,AePduRunUpdate -nolegacy`" and is executed daily at 00:00 if a network connection is available. When executed, this task updates only one file : `FullCompatReport.xml`. The previous scheduled task, `ProgramDataUpdater`, is still present and performs the same actions as previously, on top of which it updates the new file, `FullCompatReport.xml`.

4.2. AEINV_AMI_WER_{MachineId}_YYYYMMDD_HHmmss.xml

The content of this file is the same as in version 6.2.9200.16384 described in 3.4, except for the list `DriverPackageList`, which was filled during tests. Whether this change occurred because of the new version of the DLLs or if something different happened on the system is not clear.

4.2.1. DriverPackageList

This list seems to record the drivers setup information file (INF). An example for the acpi.inf file is shown in Listing 4.1.

```
<DriverPackageList>
  <DriverPackage DriverPackageId="00000c1b98d2c5496ff45687caecb218f5f52e808bc8" Date="06/21/2006"
      Version="6.3.9600.17393" Class="{4d36e97d-e325-11ce-bfc1-08002be10318}" Provider="Microsoft">
  </DriverPackage>
  [...]
</DriverPackageList>
```

4.1: Extract of AEINV_AMI_WER_{A1990A22-112B-4D0F-BB3B-625E66C092E7}_20180524_083021.xml

The meaning of the different attributes are as follows:

- DriverPackageId = SHA-1, preceded by '0000', of the INF file;
- Date = Unknown;
- Version;
- Class = Unknown;
- Provider.

4.3. FullCompatReport.xml

This file contains data about the system and what is currently installed and/or running on it. `FullCompatReport.xml` contains mostly the same information as found in other XML files, such as the list of installed applications, the list of installed drivers and the list of plugged-in devices. However, two interesting new pieces of information appear: a `GeneralTelemetry` section that records the installed KB and a list of registered services, and a list that records the usage of EXE files on the system. The reader is invited to refer to Appendix H where the structure of the file is outlined. It can help follow detailed explanations given below.

4.3.1. GeneralTelemetry

The field `GeneralTelemetry` includes a list of installed hotfixes, or updates, with the date of installation. An example of the data found in this list, for KB2976978, is shown in Listing 4.2.

```
<InstalledHotfixesQuery>
  <InstalledHotfixesData HotFixID="KB2976978" InstalledOn="11/21/2014">
  </InstalledHotfixesData>
  [...]
</InstalledHotfixesQuery>
```

4.2: Extract of FullCompatReport.xml: InstalledHotfixesData

This list can be useful to help determine when the behavior of the AmCache changed on a system since it evolves by applying Windows Update KB2952664 or KB2976978 depending on what version of Windows is installed.

`GeneralTelemetry` then lists every service on the system, running or not, at the time `FullCompatReport.xml` was last updated. An example is shown in Listing 4.3.

```
<ServicesQuery>
  [...]
  <ServicesData Name="PcaSvc" State="Running" StartMode="Auto" PathName="svchost.exe -k
      LocalSystemNetworkRestricted" DisplayName="Service de l'Assistant Compatibilité des programmes">
      </ServicesData>
  [...]
</ServicesQuery>
```

4.3: Extract of FullCompatReport.xml: ServicesQuery

The attributes for this element provides the analyst with the name of the service (`Name`), but also which command it executes (`PathName`) and the current state of the service (`State`). Besides the obvious forensic utility, this could be used to determine if the AmCache was fully functional at the time of `FullCompatReport.xml` edition, since it is mainly controlled by two services : `AeLookupSvc` and `PCASvc`.

4.3.2. ProgramUseList

This list seems to record the execution count for every EXE file executed on the system, not just the shimmed ones. Data provided in this list seems reliable according to experiments, provided the analyst keeps in mind that it is compiled at the time of the last report edition. Everything occuring afterwards is not taken into account. Such worthy information is not featured in `AmCache.hve`. An example is shown in Listing 4.4 for cmd.exe, which is not shimmed and as such not present in `AmCache.hve`.

```
<ProgramUseList SnapshotTime="01/15/2019 10:02:35">
  [...]
  <ProgramUse Id="0000f519feec486de87ed73cb92d3cac802400000000">
    [...]
    <FileUse Name="CMD.EXE" Id="00007c3d7281e1151fe4127923f4b4c3cd36438e1a12">
      <LaunchInfo LaunchId="4A81B364" LaunchCount="12" FirstLaunchTime="05/24/2018 08:31:17"
          LastLaunchTime="01/15/2019 10:02:25">
      </LaunchInfo>
    </FileUse>
    [...]
  </ProgramUse>
  [...]
</ProgramUseList>
```

4.4: Extract of FullCompatReport.xml: ProgramUseList

In the example, the analyst can determine that at the time of the snaphost, cmd.exe was launched 12 times, the first time being on the 05/24/2018 at 08:31:17 (UTC) and the last time being on the 01/15/2019 at 10:02:25 (UTC).

4.4. Examples of possible uses during a forensic investigation

The artifacts created by this version of the libraries can be interpreted as detailed in Section 3.6.

The new file, `FullCompatReport.xml`, features essential data that was absent in previous version of the AmCache: an analyst can now determine when a hotfix was installed on the system, all registered services and, for every EXE file, the number of times it was executed, along with the first and last execution time. However, an investigator studying this file should be aware that `FullCompatReport.xml` is being updated by both `ProgramDataUpdater` and `Microsoft Compatibility Appraiser`, which implies that the information it contains pertains to the last run of one of the tasks.

5. Behavior of libraries originally packaged with Windows 10 version 1507 (Threshold 1)

This chapter describes the behavior of the version 10.0.10240.16384 of the libraries, as seen on Windows Threshold 1 "out-of-the-box". One change that appeared with this version is the disappearance of both aepdu.dll and the `AeLookupSvc` service. The first one seems to have been replaced by generaltel.dll, whereas the second one seems to have been replaced by the `DiagTrack` service. Moreover, two files are missing in this version: `FullCompatReport.xml` and `PropCache.bin`.

5.1. General behavior

When executing a PE, the service `DiagTrack`, which executes "`%WinDir%\system32\svchost.exe -k utcsvc`", checks whether the PE needs shimming. If it does, the service stores information about the PE in `AmCache.hve`. If the executed PE is an installer for a program, and whether it needs shimming or not, it is handled by a different service: `PCASvc`, which executes "`%WinDir%\system32\svchost.exe -k LocalSystemNetworkRestricted`". This service runs "`%WinDir%\system32\rundll32.exe aeinv.dll,UpdateSoftwareInventory`". This dll creates a TXT file in `%WinDir%\AppCompat\Programs\Install` which is then rewritten into an XML file in the same directory. This XML file records the installation process. Moreover, it updates `AmCache.hve` with information about the newly installed program and the files the installation created.

With the disappearance of `aepdu.dll`, the scheduled tasks `ProgramDataUpdater` and `Microsoft Compatibility Appraiser` respectively run "`%WinDir%\system32\rundll32.exe generaltel.dll,RunTelemetry -maintenance`" and "`%WinDir%\ system32\rundll32.exe generaltel.dll,RunTelemetryW`". While `Microsoft Compatibility Appraiser` does not seem to do anything related to the AmCache, the XML file it previously updated having disappeared, `ProgramDataUpdater` updates both `AmCache.hve` and `AEINV_AMI_WER`. Moreover, it deletes the content of the `Install` directory.

5.2. AmCache.hve

In this version, Microsoft added three keys in `AmCache.hve`: `Device`, `HwItem` and `Metadata`. In our tests, those three keys are always empty. The `Generic` key, which previously recorded the drivers, is also empty.

The most significant change made to the `File` key is that it no longer seems to store non-GUI PEs, except when they are part of a program or when they exhibit "Microsoft Operating System" as their `ProductName`. For the last case and as opposed to what was described in Subsection 3.2.1, the `File` key associated with PEs exhibiting "Microsoft Operating System" as their `ProductName` does here seem to have a last write time coinciding with their first execution time.

The `Program` key has five new possible values: `14`, `15`, `16`, `17` and `18`. The new entry for Wireshark is shown in Fig. 5.1.

	Value Name	Value Type	Data	Value Slack
▸	0	RegSz	Wireshark 3.0.1 64-bit	00-00-00-00-00-00
	1	RegSz	3.0.1	
	13	RegDword	0	
	14	RegDword	0	
	15	RegDword	0	
	16	RegBinary	46-41-44-44-00-00-00-00-00-00-00-00-00-00-00-01-00-30-00-30-00-30-00-30-00-30-00-64-00-61-00-33-00-39-00-61-...	00-00-00-00
	17	RegQword	2814749767116800	A0-EB-00-00
	18	RegDword	0	
	2	RegSz	The Wireshark developer community, https://www.wireshark.org	00-00
	3	RegSz		
	5	RegDword	257	
	6	RegSz	AddRemoveProgram	00-00
	7	RegMultiSz	HKEY_LOCAL_MACHINE\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\Wireshark	
	a	RegQword	1563207019	00-00-00-00
	b	RegQword	0	A0-EB-00-00
	d	RegMultiSz	c:\program files\wireshark C:\Program Files\Wireshark\audio C:\Program Files\Wireshark\bearer C:\Program Fi...	00-00-00-00-00-00
	Files	RegMultiSz	451037cc-0000-0000-0000-501f00000000@2000014d71 451037cc-0000-0000-0000-501f00000000@200001...	00-00

Fig. 5.1.: Content of `AmCache.hve\Root\Programs\0000921afeb3034fbdd2ab91b80731a65ab20000ffff`

None of the meanings of those values have been found yet. The `16` value seems to have different information in

it depending on the program. For Wireshark, the value contains a binary data that is shown in Fig. 5.2. It seems to contain a SHA-1 that could not be associated with any file on the system.

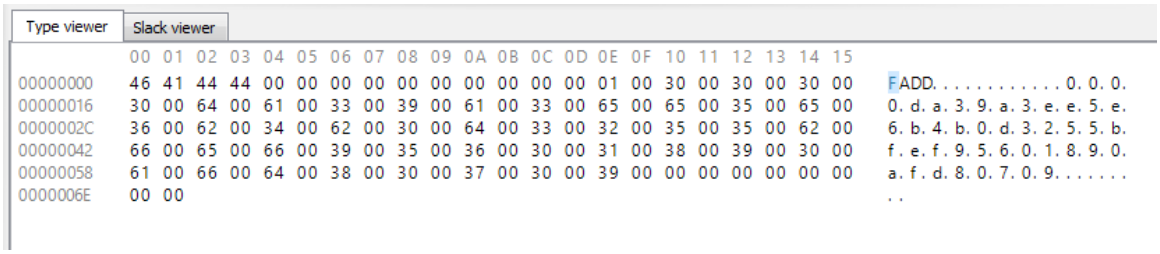


Fig. 5.2.: Content of the value 16 of `AmCache.hve\Root\Programs\0000921afeb3034fbdd2ab91b80731a65ab20000ffff\16`

For Microsoft Visual C++, the value contains the same SHA-1 along with a UTF-16 string representing the framework of the application, as shown in Fig. 5.3.

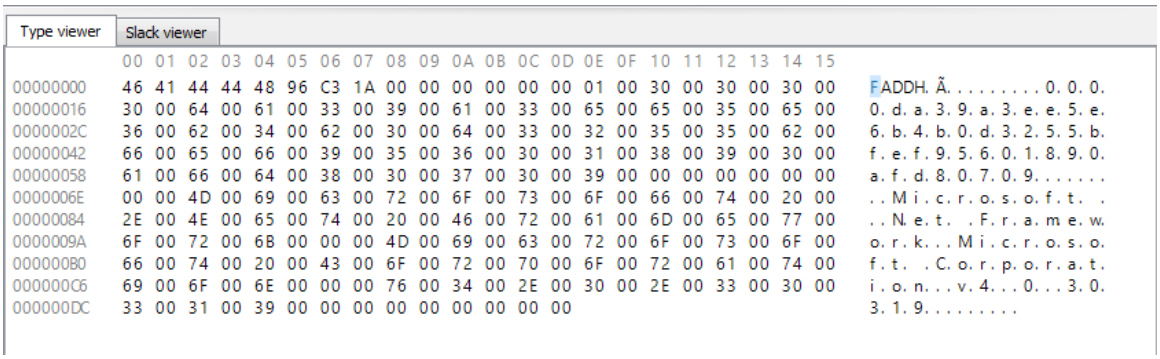


Fig. 5.3.: Content of the value 16 of `AmCache.hve\Root\Programs\0000d3fc44d32f67b84f3fb101f050fcdeac00000904`

5.3. Examples of possible uses during a forensic investigation

Like in the previous version, proof of the presence of a PE can be found in `AmCache.hve` under the `File` key. Furthermore, for the PE that is not part of a program, this is also a proof of execution. As for the last modification date of a registry `File` key, it corresponds with a run of `ProgramDataUpdater` more often than not. Indeed, and unlike what was described in previous versions, it seems that the only time when the modification date coincides with the execution date is if the PE was executed between the last run of `ProgramDataUpdater` and the retrieval of `AmCache.hve` or if the PE has a `16` value equal to 1. While not limited to those, this value is set to 1 for all PEs that are part of Microsoft Operating System. The meaning of this value remains unknown.

Installed programs are listed under the `Programs` key in `AmCache.hve` and can also be found in the `Install` directory, if `ProgramDataUpdater` has not run yet, and in `AEINV_AMI_WER`.

As for drivers, even though the `Generic` key inside `AmCache.hve` is empty, an analyst can rely on `AEINV_AMI_WER` to get the list of installed drivers.

6. Behavior of libraries originally packaged with Windows 10 version 1511 (Threshold 2)

This chapter describes the behavior of the version 10.0.10586.71 of the libraries, as seen on Windows Threshold 2 "out of the box". This version comes with two changes: `AEINV_AMI_WER` is no longer present on the system, leaving only the XML files inside the `Install` directory and `AmCache.hve`. The second change is the replacement of the dll launched by the scheduled tasks, `generaltel.dll`, with an exe file: `compattelrunner.exe`.

6.1. General behavior

When executing a PE, the service `DiagTrack` checks whether the PE needs shimming. If it does, the service stores information about the PE in `AmCache.hve`. If the executed PE is an installer for a program, and whether it needs shimming or not, it is handled by a different service: `PCASvc`. This service runs "`%WinDir%\system32\compattelrunner.exe -m:aeinv.dll -f:UpdateSoftwareInventory`". The DLL `aeinv.dll` updates `AmCache.hve`

and creates a TXT file in %WinDir%\AppCompat\Programs\Install which is then rewritten into an XML file in the same directory. This XML file records the installation process.

With the removal of generaltel.dll, the scheduled tasks ProgramDataUpdater and Microsoft Compatibility Appraiser now respectively launch "%WinDir%\system32\compattelrunner.exe -maintenance" and "%WinDir%\system32\compattelrunner.exe". Neither AmCache.hve nor any XML file seem to be updated by any of the scheduled tasks. They do not delete the content of the Install directory either.

6.2. AmCache.hve

In this version, there does not seem to be a difference in the content of AmCache.hve but there is one in its interpretation. Indeed, the fact that neither ProgramDataUpdater nor Microsoft Compatibility Appraiser update the subkeys in the File key has two important consequences. Firstly, the SHA-1 of the PEs that are part of a program is often missing because, as seen in Subsection 3.2.1, this value was most frequently filled by the scheduled tasks. Secondly, the last write time of the subkey coincides with either the first time of execution of the PE or the time of installation of the program.

The format of the Orphan key does not change and can still be used to determine if a PE is part of a program or not, as explained in Section 3.2.3: if it is referenced in this key, it should be considered as a standalone PE.

6.3. Examples of possible uses during a forensic investigation

In this version of the AmCache, all XML files except the ones under %WinDir%\AppCompat\Programs\Install disappeared. This implies that only the uses described in Section 3.6 involving AmCache.hve or the INSTALL files can apply. These are quickly recalled here. Evidence of the presence of a binary file can be found under the File key in AmCache.hve. Moreover if the PE is not part of a program, which can be checked with the Orphan key, it proves that it was executed, as explained in 6.2. AmCache.hve can be used to determine when a program was installed and when it was removed, the information being recorded in one of the values of the Programs key. Since this key retains the list of PE files under an installation folder, pinpointing a PE file that has been added to an installation folder is still possible by comparing the content of the key with the current content of the installation directory.

With the disappearance of PropCache.bin in the previous version and AEINV_AMI_WER in this one, no information pertaining to driver installation can be retrieved in this version.

Regarding the last write time of subkeys under the File key in AmCache.hve, it coincides with either the time of execution or the time of installation of the program, since the scheduled tasks no longer update AmCache.hve.

7. Behavior of libraries originally packaged with Windows 10 version 1607 (Redstone 1)

This chapter describes the behavior of the version 10.0.14913.1002 of the libraries, as seen on Windows Redstone 1 "out-of-the-box". This version comes with a major change in behavior for AmCache.hve and a new XML file, APPRAISER_FileInventory.xml.

7.1. General behavior

When executing a PE, the service DiagTrack checks whether the PE needs shimming. If it does, the service stores information about the PE in AmCache.hve. If the executed PE is an installer for a program and whether it needs shimming or not, it is handled by the service PCASvc. This service performs exactly the same actions as seen previously in 6.1.

Unlike the previous version, the two scheduled tasks ProgramDataUpdater and Microsoft Compatibility Appraiser both updates AmCache.hve. In addition, Microsoft Compatibility Appraiser updates a new file, APPRAISER_ FileInventory.xml, located under %WinDir%\appcompat\appraiser.

7.2. APPRAISER_FileInventory.xml

APPRAISER_FileInventory.xml contains information about EXE files that are under specific folders. In tests, the listed folders were always the same and are shown in Fig 7.1, but only two of them actually recorded EXE files: C:\Program Files and C:\Program Files (x86), even though the other folders did contain EXE files. It is interesting to note however, that the EXE files did not need to be executed to be listed in APPRAISER_FileInventory.xml.

```
<ScannedPaths>
  <PathEntry name="C:\ProgramData\Microsoft\Windows\Start Menu"/>
  <PathEntry name="C:\Users\Public\Desktop"/>
  <PathEntry name="C:\Program Files">
    [...]
    <File Name="Wireshark.exe" BinaryType="PE64_AMD64" Created="11/28/2018 18:40:02" Modified="
        11/28/2018 18:40:02" Size="0x0000000000754AA8" LowerCaseLongPath="c:\program files\wireshark\
        wireshark.exe" LongPathHash="0000cf4a8522cabda2c91c44e2510550f58b6983cdd5"/>
  </PathEntry>
  <PathEntry name="C:\Program Files (x86)">
  </PathEntry>
  <PathEntry name="C:\Windows\system32"/>
</ScannedPaths>
```

7.1: Content of `APPRAISER_FileInventory.xml`

7.3. AmCache.hve

`AmCache.hve` contains eight new keys:

- `InventoryDriverBinary`;
- `InventoryDriverPackage`;
- `DeviceCensus`;
- `InventoryDeviceMediaClass`;
- `InventoryDeviceContainer`;
- `InventoryDevicePnp`;
- `InventoryApplication`;
- `InventoryApplicationFile`.

Much of the new information could previously be found in `FullCompatReport.xml`, which no longer exists in this version, such as data about the OS version installed, recorded in `DeviceCensus`, and the devices that were plugged in on the system, recorded in `InventoryDeviceContainer` and `InventoryDevicePnp`.

Just as the previous version, described in Section 6.2, the keys `Device`, `HwItem`, `Metadata` and `Generic` are empty. However, the drivers, which were previously listed in `Generic`, are now under `InventoryDriverBinary` and are recorded by `Microsoft Compatibility Appraiser`. In this key, each entry is named after the SHA-1 of the driver, preceded by '0000', and the subkey representing the driver now contains the same data as previously seen in `AEINV_AMI_WER`, as shown in Fig. 7.1 for 1394ohci.sys.

Value Name	Value Type	Data	Value Slack
DriverName	RegSz	1394ohci.sys	02-00
Inf	RegSz		
DriverVersion	RegSz	10.0.14393.0	02-00
Product	RegSz	Microsoft® Windows® Operating System	00-00
ProductVersion	RegSz	10.0.14393.0	02-00
WdfVersion	RegSz		
DriverCompany	RegSz	Microsoft Corporation	
DriverPackageStrongName	RegSz		
Service	RegSz	1394ohci	02-00
DriverType	RegDword	8650778	
DriverTimeStamp	RegDword	1468635696	
DriverCheckSum	RegDword	285843	
ImageSize	RegDword	262144	

Fig. 7.1.: `AmCache.hve\Root\InventoryDriverBinary\0000895407cb018368e62fc360b972a8b0da7e729662`

Unlike previous versions of `AmCache.hve`, the values are self-explanatory, except for `DriverTimestamp`, which is the compilation date, in UNIX format.

One of the major changes undergone by `AmCache.hve` is the way it records both binaries and programs. Firstly, a new key, `InventoryApplicationFile`, only records EXE files that are part of a program. These files are also included under the `File` key. This last key is exclusively updated by `Microsoft Compatibility Appraiser`. Secondly, regarding program activity, new programs are added under the key `Programs` solely when `ProgramDataUpdater` runs, while it was previously updated by `PCASvc` at the time of installation of the program. `PCASvc` now records the installation of a program in the key `InventoryApplication`. This key also contains programs installed via an AppXPackage. As for the uninstallation of a program, the time of uninstallation is recorded in `Programs` in the `b` value, while the program key is just deleted in `InventoryApplication`.

As in the previous version of the libraries, the last write time of a key in `File` coincides with the execution time of a PE that is orphaned. For a PE that is part of a program, it coincides with either the installation time of the program or the first execution if the PE needed shimming. However, in `InventoryApplicationFile`, the last write time of the keys always coincides with an execution of `Microsoft Compatibility Appraiser`. For a program installation, the last write time of a key in `Programs` always coincides with an execution of `ProgramDataUpdater`, while the last write time of a key in `InventoryApplication` coincides with the installation time of the program.

The format of the two new keys is slightly different than `File` and `Programs`. Each EXE file is registered in `InventoryApplicationFile` under a key named after the SHA-1 of the full path of the binary (in lowercase and in UTF-16LE), preceded by '0000'. Like in `InventoryDriverBinary`, the meaning of the values describing a binary are straightforward, as shown in Fig. 7.2

Value Name	Value Type	Data	Value Slack
ProgramId	RegSz	0000c16b47f8ca21d3ca3f3ace1abb7c51e40000ffff	00-00
FileId	RegSz	00003c742e7d9ff40c291d5c1d2a9aa6c9d3b2023a34	00-00
LowerCaseLongPath	RegSz	c:\program files\wireshark\wireshark.exe	00-00
LongPathHash	RegSz	0000cf4a8522cabda2c91c44e2510550f58b6983cdd5	00-00
BinaryType	RegSz	PE64_AMD64	00-00-00-00-00-00
Size	RegSz	0x754aa8	00-00

Fig. 7.2.: `AmCache.hve\Root\InventoryApplicationFile\0000cf4a8522cabda2c91c44e2510550f58b6983cdd5`

As for `InventoryApplication`, each program entry is named after its `ProgramId`, and the values are once again easily understandable, as shown in Fig. 7.3.

Value Name	Value Type	Data	Value Slack
HiddenArp	RegSz		
InboxModernApp	RegSz		
InstallDate	RegSz	01/17/2019 18:27:06	00-00-00-00
InstallDateArpLastModified	RegMultiSz	01/17/2019 18:27:06	00-00
InstallDateFromLinkFile	RegMultiSz	01/17/2019 18:26:27	00-00
Language	RegSz		
MsiPackageCode	RegSz		
MsiProductCode	RegSz		
Name	RegSz	Wireshark 2.6.5 64-bit	00-00-00-00-00-00
OSVersionAtInstallTime	RegSz	10.0.0.14393	04-00
PackageFullName	RegSz		
ProgramId	RegSz	0000c16b47f8ca21d3ca3f3ace1abb7c51e40000ffff	00-00
ProgramInstanceId	RegSz	000040f36d871be737d86e602ed77ae1d43e5c8eb971	00-00
Publisher	RegSz	The Wireshark developer community, https://www....	00-00
RegistryKeyPath	RegSz	HKEY_LOCAL_MACHINE\Software\Wow6432Node\...	00-00
RootDirPath	RegSz	%programfiles%\wireshark	00-00
Source	RegSz	AddRemoveProgram	00-00
UninstallString	RegSz	"C:\Program Files\Wireshark\uninstall.exe"	00-00-00-00-00-00
Version	RegSz	2.6.5	

Fig. 7.3.: `AmCache.hve\Root\InventoryApplication\0000c16b47f8ca21d3ca3f3ace1abb7c51e40000ffff`

7.4. Examples of possible uses during a forensic investigation

Presence, execution and installation of PE files or programs can be ascertained exactly as for the previous version of the libraries, described in Section 6.3.

Fortunately for the forensic investigator, this version of `AmCache.hve` marks the return of the data that was missing in the previous chapter. Indeed, `AmCache.hve` records information about the system (OS version, devices plugged-in, ...) and about installed drivers. However, this data is only updated when `Microsoft Compatibility Appraiser` is run.

Up-to-date information about installed programs is now available in `InventoryApplication`, while the `Programs` key is only updated when `ProgramDataUpdater` runs.

Hunting for hidden binaries under `%SystemDrive%\Program Files` and `%SystemDrive%\Program Files (x86)` is eased by the exhaustive listing of EXE files stored in those folders in `APPRAISER_FileInventory.xml`. Such research in other installation folders still relies on the comparison between the list of binaries around the time of installation of a program (found in `AmCache.hve`) and the content of the same folder at the time of analysis of the system.

8. Behavior of libraries originally packaged with Windows 10 version 1709 (Redstone 3)

This chapter details the behavior of the version 10.0.16299.15 of the libraries, as seen on Windows Redstone 3 "out-of-the-box". Once again in this version, new keys have been added to `AmCache.hve` and a change in the behavior of the `Microsoft Compatibility Appraiser` occured.

8.1. General behavior

When executing a PE, the service `DiagTrack` checks whether the PE needs shimming. If it does, the service stores information about the PE in `AmCache.hve`. If the executed PE is an installer for a program and whether it needs shimming or not, it is handled by the service `PCASvc`. This service runs "`%WinDir%\system32\ compattelrunner.exe -m:aeinv.dll -f:UpdateSoftwareInventory`". The DLL `aeinv.dll` updates `AmCache.hve` but no longer writes information inside `%WinDir%\AppCompat\Programs\Install`.

The two scheduled tasks, `ProgramDataUpdater` and `Microsoft Compatibility Appraiser` are still present on the system. While `ProgramDataUpdater` does not seem to update any XML file nor the hive, the second task, `Microsoft Compatibility Appraiser`, exhibits several changes in its behavior. First, the task does not update `APPRAISER_FileInventory.xml` every time it runs, rendering the XML file unreliable. However, the information previously contained in the XML file, i.e. the list of PEs under specific directories, is not lost since the task now adds the binaries directly in `AmCache.hve`. From the list of "`ScannedPaths`" present in `APPRAISER_FileInventory.xml`, only the user's `Desktop` folder, `C:\Program Files` and `C:\Program Files (x86)` have their EXE files recorded. As for the last path previously stored in `ScannedPaths`, `C:\ProgramData\Microsoft\Windows\Start Menu`, it is scanned only for LNK files. Those files are added in a new key in `AmCache.hve`: `InventoryApplicationShortcut`.

In addition, `Microsoft Compatibility Appraiser` updates the key `InventoryApplication`, which contains every installed application, by rewriting all the entry in the key every time it runs. Finally, if a driver has been installed on the system since the last run, the task updates an XML file, `APPRAISER_TelemetryBaseline_UNV.bin`.

8.2. AmCache.hve

`AmCache.hve` contains five new keys:

- `DriverPackageExtended`, which only contains two values: `ProviderSyncId` and `ProviderVersion`;
- `InventoryDeviceInterface`, which contains information about sensors found on the computer (accelerometer, orientation,...);
- `InventoryDeviceUsbHubClass`, which contains the number of USB slots on the computer;
- `InventoryApplicationShortcut`, which contains information about LNK files found on the computer in the `Start Menu` directory;
- `InventoryApplicationFramework`, which lists the framework a specific application relies on.

The four keys from the first version of `AmCache.hve`, `File`, `Programs`, `Generic` and `Orphan`, are all empty. The information contained in those keys is respectively inside `InventoryApplicationFile`, `InventoryApplication`

and `InventoryDriverBinary`. As for `Orphan`, the content of this key is not mapped anywhere in the version of `AmCache.hve`.

Some changes occurred in the naming of the keys. Indeed, the subkeys under `InventoryApplicationFile` are now of the form filename and a hash, separated by the character '|'. The algorithm for the hash could not be found but seems to be based on at least the filename and the path of the binary. Indeed, on two different systems, two different versions of a binary with the same filename and path results in the same hash. The name of the keys under `InventoryDriverBinary` have also changed and is now the full path of the installed driver instead of the SHA-1 of the driver. Luckily, that information is still inside the hive under the value `DriverId`, as shown in Fig 8.1.

Value Name	Value Type	Data	Value Slack
DriverName	RegSz	1394ohci.sys	05-00
Inf	RegSz		
DriverVersion	RegSz	10.0.16299.15	
Product	RegSz	Microsoft® Windows® Operating System	06-00
ProductVersion	RegSz	10.0.16299.15	
WdfVersion	RegSz		
DriverCompany	RegSz	Microsoft Corporation	
DriverPackageStrongName	RegSz		
Service	RegSz	1394ohci	04-00
DriverInBox	RegSz	1	
DriverSigned	RegSz	1	
DriverIsKernelMode	RegSz	1	
DriverId	RegSz	00000a187cfe3469f21e6e1c050707de5b704f2deec6	00-00
DriverLastWriteTime	RegSz	09/29/2017 11:49:09	78-D8-06-00
DriverType	RegDword	8454170	
DriverTimeStamp	RegDword	65870087	
DriverCheckSum	RegDword	200635	
ImageSize	RegDword	188416	

Fig. 8.1.: `AmCache.hve\Root\InventoryDriverBinary\c:/windows/system32/drivers/1394ohci.sys`

In addition, several new values have been added to binaries under `InventoryApplicationFile`, as seen in Fig. 8.2

- Size is now stored as a 64-bit number (REG_QWORD) integer instead of a string representing its hexadecimal value;
- Name, the filename of the binary;
- Publisher;
- Version;
- BinFileVersion;
- ProductName;
- ProductVersion;
- LinkDate;
- BinProductVersion;
- Language;
- IsPeFile;
- IsOsComponent.

LNK files are now listed in `InventoryApplicationShortcut`, although the only information contained in each subkeys is the full path of the LNK. An example is shown in Fig 8.3.

Value Name	Value Type	Data	Value Slack
ProgramId	RegSz	00007d489e961eaa79515c920dd37a1be82a0000ffff	00-00
FileId	RegSz	0000bee45dd8b4b6d9950651f3ac3e0aeea01cb0ff0d	00-00
LowerCaseLongPath	RegSz	c:\program files\wireshark\wireshark.exe	00-00
LongPathHash	RegSz	wireshark.exe|8f0f02f3	
Name	RegSz	wireshark.exe	
Publisher	RegSz	the wireshark developer community, http://www.wireshark.org/	00-00
Version	RegSz	3.0.1	
BinFileVersion	RegSz	3.0.1.0	78-56-10-00
BinaryType	RegSz	pe32_i386	
ProductName	RegSz	wireshark	
ProductVersion	RegSz	3.0.1	
LinkDate	RegSz	04/08/2019 18:54:34	00-00-00-00
BinProductVersion	RegSz	3.0.1.0	00-00-08-00
Size	RegQword	7314088	90-69-08-00
Language	RegDword	1033	
IsPeFile	RegDword	1	
IsOsComponent	RegDword	0	

Fig. 8.2.: `AmCache.hve\Root\InventoryApplicationFile\wireshark.exe|8f0f02f3`

Value Name	Value Type	Data	Value Slack
ShortcutPath	RegSz	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Wireshark.lnk	00-00-76-6B-00-00

Fig. 8.3.: `AmCache.hve\Root\InventoryApplicationShortcut\wireshark.lnk|ee4ba020`

8.3. APPRAISER_Telemetry_UNV.bin

The format of this file is not fully understood yet. The file seems to list every installed driver along with information about the driver like its description, its provider... It does not seem to contain information not already found in `AmCache.hve`, which is why the study of this file has not been pushed further.

8.4. Examples of possible uses during a forensic investigation

In this version of the AmCache, all the useful information can be found in one file: `AmCache.hve`. Since the list of installed programs and the list of binaries are updated in two very different ways, the usage is slightly different from previous versions.

The list of installed programs can be found under `InventoryApplication`. This key is updated every time `Microsoft Compatibility Appraiser` runs, which implies that the last modification date of the registry key is not the date of installation of the program. However, this information can be found inside the hive, in a value called `InstallDate`, although its precision is only up to the day. Moreover, deleted programs no longer appear in AmCache.hve, which implies that all the programs listed were installed at the moment `Microsoft Compatibility Appraiser` ran.

The key containing the information related to PEs is `InventoryApplicationFile`. It seems to list three categories of PEs: executed shimmed EXE files with a GUI, EXE or SYS files that come with the installation of a program and EXE files that are present in one of the directories scanned by `Microsoft Compatibility Appraiser` (`Program Files`, `Program Files x86` and `Desktop`). The execution of a PE appearing under this key can only be ascertained if the PE is in the first category. For these, the last write time of the subkeys corresponds to the first execution date. For the other PEs, the last write time of the subkeys is either the time of execution or the date of the first run of `Microsoft Compatibility Appraiser` after the PE appeared, whichever comes first.

Hunting for illegitimate EXE or SYS files inside `Program Files` and `Program Files (x86)` is easier in this version. An analyst would have to put together all the `InventoryApplicationFile` entries for binaries under the same program directory and compare the last modification time of the registry keys to see if one is different.

Finally, information about installed drivers can be found under `InventoryDriverBinary`. This information is only updated by `Microsoft Compatibility Appraiser`.

9. Behavior of libraries originally packaged with Windows 10 version 1803 (Redstone 4) and Windows 10 version 1807 (Redstone 5)

This chapter details the behavior of the version 10.0.17134.1 of the libraries, as seen on Windows Redstone 4 "out-of-the-box". This version ends the transition between the first format of `AmCache.hve`, with the `File` and `Programs` keys and the new one, with `InventoryApplicationFile` and `InventoryApplication`. It also marks the return of the `Install` directory.

This version shares exactly the same behavior as the next one: 10.0.17763.1, present on Windows Redstone 5 "out-of-the-box".

9.1. General behavior

When executing a PE, the service `DiagTrack` checks whether the PE needs shimming. If it does, the service stores information about the PE in `AmCache.hve`. If the executed PE is an installer for a program and whether it needs shimming or not, it is handled by the service `PCASvc`. This service runs "`%WinDir%\system32\ compattelrunner.exe -m:aeinv.dll -f:UpdateSoftwareInventory`". The DLL `aeinv.dll` updates `AmCache.hve` and records the installation process in a TXT file located under `%WinDir%\AppCompat\Programs\Install`.

The two scheduled tasks, `ProgramDataUpdater` and `Microsoft Compatibility Appraiser`, exhibit exactly the same behavior as in Section 8.1.

9.2. AmCache.hve

In this version, the four keys from the first AmCache.hve, `File`, `Programs`, `Orphan` and `Generic` have been deleted, marking the end of the transition to their new counterparts: `InventoryApplicationFile`, `InventoryApplication` and `InventoryDriverBinary`.

Eleven new keys have appeared in this version:

- `InventoryApplicationAppV`;
- `InventoryApplicationDriver`;
- `InventoryMiscellaneousOfficeAddIn`;
- `InventoryMiscellaneousOfficeIdentifiers`;
- `InventoryMiscellaneousOfficeIESettings`;
- `InventoryMiscellaneousOfficeInsights`;
- `InventoryMiscellaneousOfficeProducts`;
- `InventoryMiscellaneousOfficeSettings`;
- `InventoryMiscellaneousOfficeVBA`;
- `InventoryMiscellaneousOfficeVBARuleViolations`;
- `InventoryMiscellaneousUUPInfo`.

The `InventoryApplicationAppV` has always been seen empty.

The `InventoryApplicationDriver` key lists every driver specifically installed by an application. It contains two values: `DriverServiceName`, which is the name of the service the driver installed, and `ProgramIds`, which is a list of every program id (the key name under `InventoryApplication`) that installed this driver. For example, the installation of Wireshark triggers the installation of the npcap driver. The view of the key related to this driver is shown in Fig 9.1. The value `ProgramIds` referenced two different programs: Wireshark and Microsoft Visual C++ 2017 Redistributable (x64) that was installed along with Wireshark.

The other keys are for storing information about the Office Suite (the installed add-in, the Office related Internet Explorer features...) or the Unified Update Platform. As it seems outside the scope of this research, those keys and the information they contained have not been researched in depth. Some information pertaining the content of those keys can be found in the Microsoft documentation[1].

[1] https://docs.microsoft.com/en-us/windows/privacy/basic-level-windows-diagnostic-events-and-fields-1803#inventory-events

Value Name	Value Type	Data	Value Slack
DriverServiceName	RegSz	npcap	
ProgramIds	RegSz	0000a5c8d73a8a4913750a2b7678f38ef28a0000ffff,0000921afeb3034fbdd2ab91b80731a65ab20000ffff,	00-00-00-00-00-00

Fig. 9.1.: `AmCache.hve\Root\InventoryApplicationDriver\npcap`

9.3. Install Directory

This folder contains TXT files that records the installation of some programs. The reason why some installations gets recorded and some not has not been found yet: tests in a controlled environment did not allow to see a difference between a program installation that gets recorded (Wireshark, 7Zip) and one that does not (Virtual Box Guest Additions), but the behavior seems to be consistent across different machines.

For each recorded installation, there seems to be a changing number of files created depending on if the installation results in one or more `Uninstall` registry keys. For instance, the installation of Wireshark triggers the creation of three programs (three `Uninstall` keys): Wireshark, Npcap and C++. This causes the creation of four TXT files: one for each newly installed program, whose filenames respectively starts with `INSTALL_0000`, `INSTALL_0001` and `INSTALL_0003` and one to gather everything, whose filename starts with `INSTALL_ffff`.

The files are plain text files with key-value pairs. In the previous versions where those files existed, although it was XML files, they contained information about the installer (its path, SHA-1,...), the time the installation process started and stopped, and the program identifier. Now, the TXT version of those files contain the same information along with the list of every PE file created on the system related to the installation and the path of the `Uninstall` registry key of the program. An example is shown in 9.1. Some keys being reused, the position of the key inside the file is important: for instance, there are three `Id` keys in the example, the first being the SHA-1 of the installer and the other two being undefined.

```
StartTime=07/01/2019 23:49:22
Name=Wireshark-win64-3.0.1.exe
Path=C:\Users\Lambda\Documents
Size=0x38c5e90
Magic=0x10b
SizeOfImage=0x8c000
PeChecksum=0x38cd22d
LinkDate=01/30/2018 03:57:38
LinkerVersion=6.0
BinFileVersion=3.0.1.0
BinProductVersion=3.0.1.0
BinaryType=PE32_I386
Created=05/09/2019 12:45:21
Modified=05/09/2019 12:45:21
LastAccessed=07/01/2019 16:44:12
VerLanguage=1033
Id=0000a067b99a6acc5dc5b1a6f25ebc3ceabe03d6f0fb
FileVersion=3.0.1.0
CompanyName=Wireshark development team
FileDescription=Wireshark installer for 64-bit Windows
LegalCopyright=© Gerald Combs and many others
ProductName=Wireshark
ProductVersion=3.0.1.0
PeImageType=0x14c
PeSubsystem=2
CrcChecksum=0x4fd2618
FileSize=0x00000000038C5E90
LongName=Wireshark-win64-3.0.1.exe
LongName=Wireshark-win64-3.0.1.exe
Id=0000000000000000000000000000000000000000
MsiProductCode={C99E2ADC-0347-336E-A603-F1992B09D582}
MsiPackageCode={1C423F21-E891-44F3-8FE9-E37D44470EF1}
LongName=Wireshark-win64-3.0.1.exe
Id=0000000000000000000000000000000000000000
MsiProductCode={2CD849A7-86A1-34A6-B8F9-D72F5B21A9AE}
MsiPackageCode={6D0A1ACD-F1C9-464F-8C70-F10295482CBE}
MsiDetected=1
FileCreate=C:\Program Files\Wireshark\uninstall.exe
FileCreate=C:\Program Files\Wireshark\libwiretap.dll
[...]
FileCreate=C:\Program Files\Wireshark\rawshark.exe
FileCreate=C:\Program Files\Wireshark\mmdbresolve.exe
ArpCreate=SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\{e2ee15e2-a480-4bc5-bfb7-e9803d1d9823}
ArpCreate=SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\Wireshark
ArpCreate=SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\NpcapInst
```

```
StopTime=07/01/2019 23:50:44
```

9.1: Content of INSTALL_ffff_f2fb886d-3422-427b-ab5d-58d0648f6d80.txt

9.4. Examples of possible uses during a forensic investigation

All the uses detailed in the previous version remains effective in this one. They are described in 8.4. In addition, one of the new key, `InventoryApplicationDriver`, can provide context on a driver installation and be used to help alleviate doubts on a suspicious driver.

The fact that the Install files are available once again implies that this version provides a more precise installation date and information about deleted programs. In those TXT files, the list of every new file added by a program installation includes DLLs which are not recorded in the key `InventoryApplicationFile` in `AmCache.hve`. This information can help pinpoint malicious DLLs added inside a program install directory by comparing the list to the current content of the directory.

10. Conclusion

This article is aimed at providing means for an analyst to reliably interpret the AmCache. To do so, it explores in details the various files left behind by either a service (`AeLookupService`, `PCASvc` or `DiagTrack`) or a scheduled task (`ProgramDataUpdater` or `Microsoft Compatibility Appraiser`). The majority of those files were never publicly researched even though they do not only show proof of execution, but also of program installation or removal, and of driver installation. They can even sometimes allow to pinpoint an unusual PE hidden in an installation folder. Furthermore, the examination of these files provides a mean to retrieve more information than when only looking at the Amcache hive - which appears with Windows 8. For instance, the list of installed programs was recorded starting with the first version of `AmCache.hve`, in Windows 8, but was available before, in `AEINV_PREVIOUS.xml` or `AEINV_WER`. The same goes for the drivers, that were recorded in the `AmCache.hve` starting with Windows 10, while their list was already present in Windows 8 and 8.1, in `AEINV_AMI_WER`.

This article also shows that it is important to keep in mind that the behavior of the AmCache is dictated by versions of libraries and not by the OS version of the system. This is especially relevant for two reasons. The first is that, when investigating an older system that could have undergone several upgrades, there could still be traces of the files of previous AmCache versions. The second is that Microsoft keeps changing the behavior of the AmCache, and while some changes are minor (like a change in a key name), some have bigger repercussions, like not storing the non-GUI executables or DLLs anymore. They can also have consequences on the interpretation of the information: for instance, the last write date of the keys in `AmCache.hve`. In some versions of the `AmCache`, it almost never corresponds to the date of execution of the PE, whereas in recent versions, it does more frequently. This paper highlights the extreme complexity of the inner workings of the Shim infrastructure, and the difficulties it yields for a forensic examiner to interpret artifacts in a sound manner. A lot of experiments have been carried out to confirm the meaning of the presence of an element in a file or a registry key. However, it is important that the reader keeps in mind two facts. Firstly, they remain experiments rather than source code analysis. Secondly, as is often the case in forensics, only the presence of an element should be relied on to draw a conclusion according to these artifacts. Reasoning on the absence of an element seems beyond the scope of the tests performed for this work.

Finally, as more and more information is stored in recent versions of the AmCache, it no longer only stores information about executed PE or installed programs. Hence, it seems relevant to continue studying this artifact. This will certainly prove quite beneficial to the digital forensics community.

A. Artifact location summary

Table A.1.: Artifacts

Version	Artifacts
6.1.7600 and 6.1.7601	%WinDir%\AppCompat\Programs\RecentFileCache.bcf
	%WinDir%\AppCompat\Programs\AEINV_PREVIOUS.xml
	%WinDir%\AppCompat\Programs\AEINV_WER_{MachineId}_YYYYMMDD_HHmmss.xml
6.2.9200 and 6.3.9600.16384	%WinDir%\AppCompat\Programs\AmCache.hve
	%WinDir%\AppCompat\Programs\AEINV_PREVIOUS.xml
	%WinDir%\AppCompat\Programs\AEINV_AMI_WER_{MachineId}_YYYYMMDD_HHmmss.xml
	%WinDir%\AppCompat\Programs\Install\INSTALL_ffff_*.xml
	%WinDir%\AppCompat\Programs\DevInvCache\PropCache.bin
6.3.9600.17415	%WinDir%\AppCompat\Programs\AmCache.hve
	%WinDir%\AppCompat\Programs\AEINV_PREVIOUS.xml
	%WinDir%\AppCompat\Programs\AEINV_AMI_WER_{MachineId}_YYYYMMDD_HHmmss.xml
	%WinDir%\AppCompat\Programs\FullCompatReport.xml
	%WinDir%\AppCompat\Programs\Install\INSTALL_ffff_*.xml
	%WinDir%\AppCompat\Programs\DevInvCache\PropCache.bin
10.0.10240	%WinDir%\AppCompat\Programs\AmCache.hve
	%WinDir%\AppCompat\Programs\AEINV_AMI_WER_{MachineId}_YYYYMMDD_HHmmss.xml
	%WinDir%\AppCompat\Programs\Install\INSTALL_ffff_*.xml
10.0.10586	%WinDir%\AppCompat\Programs\AmCache.hve
	%WinDir%\AppCompat\Programs\Install\INSTALL_ffff_*.xml
10.0.14913	%WinDir%\AppCompat\Programs\AmCache.hve
	%WinDir%\AppCompat\Programs\Install\INSTALL_ffff_*.xml
	%WinDir%\AppCompat\Programs\appraiser\APPRAISER_FileInventory.xml
10.0.16299	%WinDir%\AppCompat\Programs\AmCache.hve
	%WinDir%\AppCompat\Programs\appraiser\APPRAISER_FileInventory.xml
	%WinDir%\AppCompat\Programs\appraiser\APPRAISER_TelemetryBaseline_UNV.bin
10.0.17134	%WinDir%\AppCompat\Programs\AmCache.hve
	%WinDir%\AppCompat\Programs\Install\INSTALL_*.txt
	%WinDir%\AppCompat\Programs\appraiser\APPRAISER_TelemetryBaseline_UNV.bin
10.0.17763	%WinDir%\AppCompat\Programs\AmCache.hve
	%WinDir%\AppCompat\Programs\Install\INSTALL_*.txt
	%WinDir%\AppCompat\Programs\appraiser\APPRAISER_TelemetryBaseline_UNV.bin

B. AmCache.hve registry keys summary

We recall that in the following table, an *installed* program is an application that should have either an `Uninstall` or a `Run` key in the `SOFTWARE` hive.

Table B.1.: AmCache.hve registry keys

Registry key	Content	Appears with	Changes with
File	Metadata about PEs if they are: shimmed and executed or the installer of a program or created on the system following a program installation. Described in 3.2.1	6.2.9200.16384	Emptied with 10.0.16299 Deleted with 10.0.17134
Programs	Metadata about installed (and uninstalled) programs Described in 3.4 and in 7.3	6.2.9200.16384	Changed with 10.0.14913 Emptied with 10.0.16299 Deleted with 10.0.17134
Orphan	References the executed PEs that are recorded in `File` but not part of a program. Described in 3.2.3	6.2.9200.16384	Emptied with 10.0.16299 Deleted with 10.0.17134
Generic	SHA-1 of installed driver, without its name. Described in 3.2.4	6.2.9200.16384	Emptied with 10.0.16299 Deleted with 10.0.17134
Device	Always seen empty	10.0.10240	Deleted with 10.0.17134
HwItem	Always seen empty	10.0.10240	Deleted with 10.0.17134
Metadata	Always seen empty	10.0.10240	Deleted with 10.0.17134
InventoryDriverBinary	Metadata about installed drivers. Described in 7.3	10.0.14913	
InventoryDriverPackage		10.0.14913	
DeviceCensus	Data about the OS	10.0.14913	
InventoryDeviceMediaClass		10.0.14913	
InventoryDeviceContainer		10.0.14913	
InventoryDevicePnp	Devices plugged in on the system	10.0.14913	
InventoryApplication	Content is the same as `Programs` key. Described in 7.3 and in 8.1	10.0.14913	Changed in 10.0.16299
InventoryApplicationFile	Metadata about EXEs if they are shimmed, executed and have a GUI or if they are in scanned directories. It also records metadata about EXE and SYS files if they were created on the system following a program installation. Described in 7.3 and in 8.2	10.0.14913	Changed with 10.0.16299
DriverPackageExtended		10.0.16299	
InventoryDeviceInterface	Information about sensors found on the computer	10.0.16299	
InventoryDeviceUsbHubClass	Lists the USB slots on the computer	10.0.16299	
InventoryApplicationShortcut	Lists LNK files found on the computer	10.0.16299	
InventoryApplicationFramework	Lists the frameworks an application relies on	10.0.16299	
InventoryApplicationAppV		10.0.17134	
InventoryApplicationDriver	Links a driver with the program with which it was installed. Described in 9.2	10.0.17134	
InventoryMiscellaneousOffice AddIn		10.0.17134	

InventoryMiscellaneousOffice Identifiers		10.0.17134	
InventoryMiscellaneousOffice IESettings		10.0.17134	
InventoryMiscellaneousOffice Insights		10.0.17134	
InventoryMiscellaneousOffice Products		10.0.17134	
InventoryMiscellaneousOffice Settings		10.0.17134	
InventoryMiscellaneousOffice VBA		10.0.17134	
InventoryMiscellaneousOffice VBARuleViolations		10.0.17134	
InventoryMiscellaneousUUPInfo		10.0.17134	

C. RecentFileCache.bcf structure

The general structure of the file is described in Fig. C.1. The `Fixed` field is always ”0xFEFFEEFF 11220000 03000000 01000000”. The path are in UTF-16 LE and `Size` is the number of characters (so twice the number of bytes), not counting the ending ’\00’. This goes on until the end of file, but the number of paths is not present in the header.

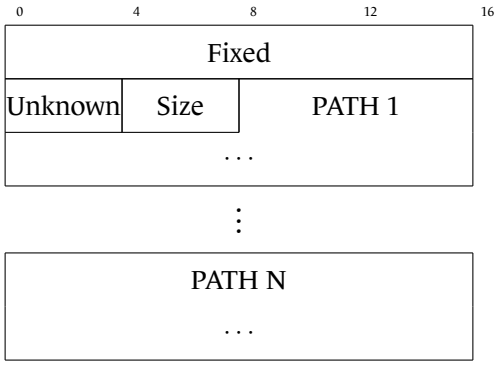


Fig. C.1.: RecentFileCache.bcf byte structure

D. AEINV_PREVIOUS.xml structure

```
<Log>          <!-- Version of the ae<...>.dll libraries -->
  <ProgramList>     <!-- List of installed programs -->
    <Program>      <!-- One entry by program. Attributes are described in Table C.1 -->
      <StaticProperties>
        <Files />     <!-- Unknown -->
      </StaticProperties>
    </Program>
    [...]
  </ProgramList>
  <IEAddOnList>     <!-- List of installed add-ons for Internet Explorer -->
    <IEAddOn>      <!-- One entry by add-on. Attributes are described in Table C.2 -->
      <File />     <!-- Information about the PE that provides the add-on. Attributes are described in Table C.3 -->
    </IEAddOn>
    [...]
  </IEAddOnList>
</Log>
```

D.1: Generic structure of AEINV_PREVIOUS.xml

Table D.1.: Program attributes

Attribute	Description	Example
Id	Unknown	Id="0000354384b2dbc2f6b2dc9dec22174dcf510000ffff"
MsiProductCode		MsiProductCode="{C3CC4DF5−39A5−4027−B136−2B3E1F5AB6E2}"
MsiPackageCode		MsiPackageCode="{AE5CF7E6−1FAD−47DF−A41F−3261FBF6B305}"
Name		Name="Wireshark 2.6.5 32−bit"
Publisher		Publisher="The Wireshark developer community, https://www.wireshark.org"
Version		Version="2.6.5"
Language	Microsoft Language Id, in decimal (1033 for en-us)[1]	Language="1033"
Source	`Msi` or `AddRemoveProgram`, depending on whether the program was installed by executing an MSI or a PE.	Source="AddRemoveProgram"

Table D.2.: IEAddOn attributes

Attribute	Description	Example
CLSID		CLSID="{19916E01−B44E−4E31−94A4−4696DF46157B}"
Name		Name="InformationCardSigninHelper Class"
Type	Observed values: `ActiveX`, `BrowserHelperObject` and `BrowserExtension`	Type="ActiveX"
Publisher		Publisher="Microsoft Corporation"

Table D.3.: File attributes

Attribute	Description	Example

[1] `https://docs.microsoft.com/en-us/windows/desktop/intl/language-identifier-constants-and-strings`

Id	SHA-1 preceded by '0000'	Id="0000d8b095849b5172e07dff1562bad89f37037bf951"
Name		Name="icardie.dll"

E. AEINV_WER structure

```
<Report>    <!-- Information about the file. Attributes are described in Table D.1 -->
  <System />   <!-- Information about the system. Attributes are described in Table D.2 -->
  <ProgramList>
    <Installed>       <!-- List of installed programs -->
      <Program>       <!-- One entry by program. Attributes are described in Table D.3 -->
        <Indicators>     <!-- List of installed programs -->
          <RegistryIndicators> <!-- List of Run keys associated with the program -->
            <Registry />   <!-- Information about a Run Key. Attributes are described in Table D.4 -->
          </RegistryIndicators>
          <AddRemoveProgramIndicators> <!-- List of Uninstall keys associated with the program -->
            <AddRemoveProgram /> <!-- Information about an Uninstall Key. Attributes are described in Table D.5 -->
          </AddRemoveProgramIndicators>
          <ShellIndicators>  <!-- List of exe files listed in the Start Menu -->
            <Shell />     <!-- Information about PE. Attributes are described in Table D.6 -->
          </ShellIndicators>
          <MsiIndicators>   <!-- Information about the MSI file used to install the program -->
            <Msi />      <!-- Information about MSI. Attributes are described in Table D.7 -->
          </MsiIndicators>
          <FileExtIndicators>  <!-- File extensions that are opened by the program -->
            <FileExtensionHandler /> <!-- Extension. Attributes are described in Table D.8 -->
            [...]
          </FileExtIndicators>
          <DirectoryIndicators>  <!-- Installation folder and sub-folder containing PE file -->
            <Directory />   <!-- Folder. Attributes are described in Table D.9 -->
            [...]
          </DirectoryIndicators>
        </Indicators>
        <StaticProperties>
          <Files>       <!-- List of PEs under the installation folder and sub-folder. Only contains one attribute: Id -->
            <File />     <!-- PE. Attributes are described in Table D.10 -->
            [...]
          </Files>
        </StaticProperties>
      </Program>
      [...]
    </Installed>
    <Updated>        <!-- List of programs where a change occurred in one of its indicators -->
      <Program>
        <Indicators>
          <RegistryIndicators>
            <Registry />
          </RegistryIndicators>
          <AddRemoveProgramIndicators>
            <AddRemoveProgram />
          </AddRemoveProgramIndicators>
          <ShellIndicators>
            <Shell />
          </ShellIndicators>
          <MsiIndicators>
            <Msi />
          </MsiIndicators>
          <FileExtIndicators>
            <FileExtensionHandler />
          </FileExtIndicators>
          <DirectoryIndicators>
            <Directory />
          </DirectoryIndicators>
        </Indicators>
        <StaticProperties>
          <Files>
```

```
            <File />
            [...]
          </Files>
        </StaticProperties>
      </Program>
      [...]
    </Updated>
    <Removed>       <!-- List of deleted programs -->
    </Removed>
    <Orphan>       <!-- List of executed PE not in a program install directory -->
      <Program>
        <Indicators></Indicators>
        <StaticProperties>
          <Files>
            <File />
            [...]
          </Files>
        </StaticProperties>
      </Program>
    </Orphan>
  </ProgramList>
  <IEAddOnList> <!-- Only one attribute : InstanceVersion, which is the version of Internet Explorer installed -->
    <Installed> <!-- List of installed Internet Explorer add-ons -->
      <IEAddOn>       <!-- One entry by add-on. Attributes are described in Table D.11 -->
        <File />      <!-- Information about the PE that provides the add-on. Attributes are described in Table D.12 -->
      </IEAddOn>
      [...]
    </Installed>
  </IEAddOnList>
  <Installations /> <!-- Always seen empty -->
</Log>
```

E.1: Generic structure of AEINV_WER

Table E.1.: Report attributes

Attribute	Description	Example
Version	Unknown	Version="1.3"
Timestamp	Finished writing time of the report after the first execution of `ProgramDataUpdater` in UTC	Timestamp="12/06/2018 09:43:40"
SequenceNumber	Unknown	SequenceNumber="1"
ThrottlingRuleSetGuid	Unknown	ThrottlingRuleSetGuid="{F7D0E8C8-2DA8-4889-A910-3DE830B4148F}"

Table E.2.: System attributes

Attribute	Description	Example
MachineId	Same ID that the one in the filename	MachineId="{49A35C5F-CCE9-48C7-B6EF-577A36E86135}"
MajorVersion	First part of the Windows Version Number	MajorVersion="6"

MinorVersion	Second part of the Windows Version Number	MinorVersion="1"
ServicePackMajor		ServicePackMajor="1"
ServicePackMinor		ServicePackMinor="0"
BuildNumber		BuildNumber="7601"
Sku	Version of Windows installed as found in the OperatingSystemSKU Enum	Sku="1"
ProcessorArchitecture	1 for 32-bit, 2 for 64-bit	ProcessorArchitecture="1"
OSPlatform	Unknown	OSPlatform="1"
LocaleId	decimal value of LocalName	LocaleId="1033"
GeoId		GeoId="244"

Table E.3.: Program attributes

Attribute	**Description**	**Example**
Name		Name="Wireshark 2.6.5 32−bit"
Type	Only value seen: "Application"	Type="Application"
Source	`Msi` or `AddRemoveProgram`, depending on whether the program was installed by executing an MSI or a PE	Source="AddRemoveProgram"
Publisher		Publisher="The Wireshark developer community, https://www.wireshark.org"
Version		Version="2.6.5"

OnSystemDrive	Unknown	OnSystemDrive="True"
EvidenceId	Starting point for indicators described below	EvidenceId="0x22"
Id	Unknown	Id="0000354384b2dbc2f6b2dc9dec22174dcf510000ffff"
InstallDate	Date of installation. Only present for MSI programs and the time is always 00:00:00	InstallDate="10/27/2015 00:00:00"
MsiPackageCode		MsiPackageCode="{AE5CF7E6−1FAD−47DF−A41F−3261FBF6B305}"
MsiProductCode		MsiProductCode="{C3CC4DF5−39A5−4027−B136−2B3E1F5AB6E2}"

Table E.4.: Registry attributes

Attribute	Description	Example
Name	Value of the `Run` key	Name="VBoxTray"
File	Filename contained in the data of the value <Name>	File="VBoxTray.exe"
RegistryRun	Location of the autostart entry: Run, RunOnce, RunOnceEx	RegistryRun="Run"
UniqueId	Unknown	UniqueId="0x2e"
Id	Unknown	Id="000000e4ecea2abfce5ca7602d5815f5fe8809e1e59d"

Table E.5.: AddRemoveProgram attributes

Attribute	Description	Example
DisplayName	Data contained in the `DisplayName` value of the `Uninstall` key	DisplayName="Wireshark 2.6.5 32−bit"
CompanyName	Data contained in the `Publisher` value of the `Uninstall` key	CompanyName="The Wireshark developer community, https://www.wireshark.org"

ProductVersion	Data contained in the `DisplayVersion` value of the `Uninstall` key	ProductVersion="2.6.5"
RegistrySubKey	Name of the `Uninstall` key	RegistrySubKey="Wireshark"
UniqueId	Unknown but same data as the `UniqueId` in the `Program` attribute	UniqueId="0x22"
Id	Unknown	Id="00000773cfd2b58429384da8a9bea4a99e8bbef55402"

Table E.6.: Shell attributes

Attribute	Description	Example
ShellName	Name displayed in the Start Menu	ShellName="Wireshark"
TargetFileName	Filename of the file executed	TargetFileName="Wireshark.exe"
UniqueId	Unknown	UniqueId="0xa0"
Id	Unknown	Id="00008f6fc717280228fa0fe0473fb0c23d38dd23f131"

Table E.7.: MSI attributes

Attribute	Description	Example
ProductName		ProductName="Python 2.7.6"
CompanyName		CompanyName="Python Software Foundation"
ProductVersion		ProductVersion="2.7.6150"
Language	Microsoft Language Id, in decimal (1033 for en-us)	Language="1033"
ProductCode		ProductCode="{C3CC4DF5−39A5−4027−B136−2B3E1F5AB6E2}"

PackageCode		PackageCode="{AE5CF7E6−1FAD−47DF−A41F−3261FBF6B305}"
InstallDate	Installation date, but time is always 00:00:00	InstallDate="10/27/2015 00:00:00"
UniqueId	Unknown	UniqueId="0xa"
Id	Unknown (not the SHA-1 of the MSI)	Id="0000f58476f702201e0706cb40cd350aad5cc387c133"

Table E.8.: FileExtensionHandler attributes

Attribute	Description	Example
Extension		Extension=".5vw"
Name	Data in the default value of `HKCR\Classes\<Extension>`	Name="wireshark−capture−file"
File	Filename of the binary that reads the files with this extension	File="Wireshark.exe"
UniqueId	Unknown	UniqueId="0xa6"
Id	Unknown	Id="0000f100f0a810d3369fb23078ccfccf2a9ae2342793"

Table E.9.: Directory attributes

Attribute	Description	Example
UniqueId	Records where the folder is located in the installation directory, starting with the installation directory itself	UniqueId="0x23"
Id	Unknown	Id="00009afdcc213e845b1ed280a8d118317c363e807da5"

Table E.10.: File attributes in the StaticProperties list

Attribute	Description	Example

Name	Filename	Name=”capinfos.exe”
Id	SHA-1 of the PE, preceded by ’0000’	Id=”00005c5ecbf7d4e969ff50b186109b2c18b47f257365”
ProductName	The ”Product name” field from the file metadata	ProductName=”Capinfos”
CompanyName		CompanyName=”The Wireshark developer community”
ProductVersion	The ”Product version” field from the file metadata	ProductVersion=”2.6.5”
VerLanguage	Microsoft Language Id, in decimal (1033 for en-us)	VerLanguage=”1033”
ShortName	Short name as found in the MFT	ShortName=”API—MS~1.DLL”
SwitchBackContext	Unknown	SwitchBackContext=”0x0100000000000600”
FileVersion		FileVersion=”2.6.5”
Size	Size of the PE in bytes	Size=”0x532a8”
SizeOfImage	The SizeOfImage field from the optional header of the PE	SizeOfImage=”0x53000”
PeHeaderHash		PeHeaderHash=”01012864b33151873a9ca2d4c0c5e28d87cfb023f0f3”
PeChecksum	The Checksum field from the optional header of the PE	PeChecksum=”0x5fe24”
BinProductVersion		BinProductVersion=”2.6.5.0”
BinFileVersion		BinFileVersion=”2.6.5.0”
FileDescription	The ”Description” field from the file metadata	FileDescription=”Capinfos”

LinkerVersion	The ”MajorLinkerVersion” and ”MinorLinkerVersion” fields combine from the optional header of the PE	LinkerVersion=”14.12”
LinkDate	Compile date in UTC	LinkDate=”11/28/2018 18:23:59”
BinaryType	`32BIT` or `64BIT`	BinaryType=”32BIT”
Created	Creation date in UTC	Created=”11/28/2018 18:31:44”
Modified	Modification date in UTC	Modified=”11/28/2018 18:31:44”
LongPathHash	SHA-1 of the full path in lowercase, encoded in UTF-16LE	LongPathHash=”0000058d47d0b218994a27e38ea102effc68e3b18ed3”
UniqueId	Records where the file is located in the installation directory	UniqueId=”0x24”

Table E.11.: IEAddOn attributes

Attribute	Description	Example
Name		Name=”XSL Template 3.0”
Type	Values seen : ”ActiveX”, ”BrowserHelperObject”, ”BrowserExtension”, ”Toolbar”	Type=”ActiveX”
Publisher		Publisher=”Microsoft Corporation”
CLSID		CLSID=”{f5078f36−c551−11d3−89b9−0000f81fe221}”
UniqueId	Unknown	UniqueId=”0x35”

The attributes for the `File` element in the `IEAddOn` list are the same as those found in the StaticProperties and described in Table E.10. The only additional attributes is described in Table E.12

Table E.12.: File attributes in the IEAddOn list

Attribute	Description	Example

OsComponent		OsComponent="true"

F. AEINV_AMI_WER structure

```
<Report>    <!-- Information about the file. Attributes are described in Table E.1 -->
  <System />   <!-- Information about the system. Attributes are described in Table E.2 -->
  <ProgramList>
    <Installed>       <!-- List of installed programs -->
      <Program>       <!-- One entry by program. Attributes described previously in Table D.3 -->
        <Indicators>     <!-- List of installed programs -->
          <RegistryIndicators> <!-- List of Run keys associated with the program -->
            <Registry />   <!-- Information about a Run Key. Attributes described previously in Table D.4 -->
          </RegistryIndicators>
          <AddRemoveProgramIndicators> <!-- List of Uninstall keys associated with the program -->
            <AddRemoveProgram /> <!-- Information about an Uninstall Key. Attributes described previously in Table D.5 -->
          </AddRemoveProgramIndicators>
          <ShellIndicators>  <!-- List of exe files listed in the Start Menu -->
            <Shell />     <!-- Information about PE. Attributes described previously in Table D.6 -->
          </ShellIndicators>
          <MsiIndicators>   <!-- Information about the MSI file used to install the program -->
            <Msi />      <!-- Information about MSI. Attributes described previsouly in Table D.7 -->
          </MsiIndicators>
          <FileExtIndicators>  <!-- File extensions that are opened by the program -->
            <FileExtensionHandler /> <!-- Extension. Attributes described previsouly in Table D.8 -->
            [...]
          </FileExtIndicators>
          <DirectoryIndicators> <!-- Installation folder and sub-folder containg PE file -->
            <Directory />   <!-- Folder. Attributes are described in Table D.9 -->
            [...]
          </DirectoryIndicators>
        </Indicators>
        <StaticProperties>
          <Files>       <!-- List of PEs under the installation folder and sub-folder. Only contains one attribute: Id -->
            <File />     <!-- PE. Attributes described previously in Table E.3 -->
            [...]
          </Files>
        </StaticProperties>
      </Program>
      [...]
    </Installed>
    <Removed>        <!-- List of deleted programs -->
    </Removed>
    <Orphan>        <!-- List of executed PE not in a program install directory -->
      <Program>
        <Indicators></Indicators>
        <StaticProperties>
          <Files>
            <File />
            [...]
          </Files>
        </StaticProperties>
      </Program>
    </Orphan>
  </ProgramList>
  <IEAddOnList>
    <Installed>       <!-- List of installed add-ons for Internet Explorer -->
      <IEAddOn>       <!-- One entry by add-on. Attributes are described in Table D.11 -->
        <File />      <!-- Information about the PE that provides the add-on. Attributes described previously in Table D.12 -->
      </IEAddOn>
      [...]
    </Installed>
  </IEAddOnList>
  <InstallerList> <!-- Installation process for each program -->
    <Installer> <!-- Date and time the installation started and finished. Attributes are described in Table E.4 -->
      <InstallInfo> <!-- Information about the setup exe. Attributes are described in Table E.5 -->
      </InstallInfo>
      <DiscInfo> <!-- Information about the disc the setup exe was on, if any. Attributes are described in Table E.6 -->
      </DiscInfo>
      <ProgramIds> <!--List the programs that were installed by the setup exe -->
```

```
        <ProgramId> <!-- Program installed by the setup exe. Only contains one attribute: Id -->
        </ProgramId>
      </ProgramIds>
    </Installer>
    [...]
  </InstallerList>
  <DeviceList>
    <DeviceContainer> <!-- List of physical devices that were plugged on the system. -->
      <Categories>
        <Category>
        </Category>
      </Categories>
      <Device>
        <HardwareIds>
          <HardwareId>
          </HardwareId>
          [...]
        </HardwareIds>
        <CompatibleIds>
          <CompatibleId>
          </CompatibleId>
          [...]
        </CompatibleIds>
      </Device>
      [...]
    </DeviceContainer>
    [...]
  </DeviceList>
  <DriverList> <!-- List of drivers installed on the system -->
    <Driver> <!-- Information about the driver. Attributes are described in Table E.7 -->
    </Driver>
    [...]
  </DriverList>
  <DriverPackageList> <!-- Always seen empty in tests -->
  </DriverPackageList>
  <AitAnalysis> <!-- Always seen empty in tests -->
  </AitAnalysis>
</Log>
```

F.1: Generic structure of AEINV_AMI_WER

The attributes for the Report element are the same as those found in `AEINV_WER` and described in Table E.1. The only additional attribute is described in Table F.1.

Table F.1.: Report attribute

Attribute	Description	Example
ClientVersion	Unknown, seems similar to the `Version`	ClientVersion="1.12.0"

The attributes for the System element are the same as those found in `AEINV_AMI` and described in Table E.2. The two additional attributes are described in Table F.2.

Table F.2.: System attributes

Attribute	Description	Example
VirtualMachine	Unknown: the tests on virtual machine all had `false` for this value.	VirtualMachine="false"
PortableWorkSpace	WindowsToGo or other USB booted environment	PortableWorkSpace="false"

The attributes for the File element are the same as those found in `AEINV_AMI` and described in Table E.10. The three additional attributes are described in Table F.3.

Table F.3.: File attributes

Attribute	Description	Example
CrcChecksum	Unknown	CrcChecksum="0xcb05168e"
PeImageType	Unknown	PeImageType="0x8664"
PeSubsystem	Unknown	PeSubsystem="2"

Table F.4.: Installer attributes

Attribute	Description	Example
CompletionState	`1` if the install was successful, `0` if not	CompletionState="1"
CreatedArpEntries	Always `1`	CreatedArpEntries="1"
StartTime	Timestamp, in UTC, the installation started	StartTime="08/21/2018 12:57:00"
StopTime	Timestamp, in UTC, the installation finished	StopTime="08/21/2018 12:58:47"

The attributes for the InstallInfo element are similar to those found in `AEINV_WER` and described in Table E.10 except that there is no `UniqueId` and two additional attributes, described in Table F.5.

Table F.5.: InstallInfo attributes

Attribute	Description	Example
OsComponent	Unknown	OsComponent="false"
SigPublisherName	The signer of the PE certificate	SigPublisherName="Wireshark Foundation, Inc"

Table F.6.: DiscInfo attributes

Attribute	Description	Example
Name	Name of the Disc	Name="VBOXADDITIONS_5."
Id	Unknown	Id="0004021d62bcd80dc4a5ac67b8fbfdb91516395084b5"
SetupScriptChecksum	CRC64 for INF, INI, ISS and OSD files content from the root of an installation media	SetupScriptChecksum="17231136449290210510"
Size	IpTotalNumberOfBytes from the GetDiskFreeSpaceEx method of kernel32.dll	Size="58466304"

Table F.7.: Driver attributes

Attribute	Description	Example
DriverId	SHA-1 preceded by '0000' of the driver	DriverId="000002da97a4940b126c7710d13b431a6e74123f3cc0"
Name	Filename	Name="1394ohci.sys"
Type	Bitfield of driver attributes[1]	Type="0x0004001a"
Version		Version="6.2.9200.16384"
TimeStamp	Compilation date in UNIX timestamp format	TimeStamp="0x5010aae6"
CheckSum		CheckSum="0x00047021"
ImageSize		ImageSize="0x0003d000"
PagedSize		PagedSize="0x00000e00"
Company		Company="Microsoft Corporation"
Product		Product="Microsoft® Windows® Operating System"

[1] https://docs.microsoft.com/en-us/windows/privacy/basic-level-diagnostic-events-and-fields-1709#microsoftwindowsinventorycoreinventorydriverbinaryadd

ProductVersion		ProductVersion="6.2.9200.16384"

G. PropCache.bin structure

The general structure of the file is described in Fig. G.1. The `Size` field is the size of the file, in bytes. `I` is the number of path listed in `PropCache.bin`.

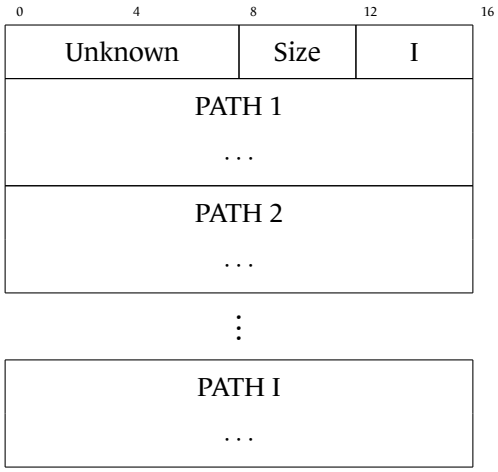


Fig. G.1.: PropCache.bin byte structure

The structure of a `PATH` field is described in Fig. G.2. The `Size` field is the size of the `PATH` field, in bytes. `J` is the number of driver listed in `PropCache.bin`, which are drivers located under the `PATH` previously identified. `str_len` is the number of bytes in `folder_name` (which is in UTF-16 LE), including the final "\00".

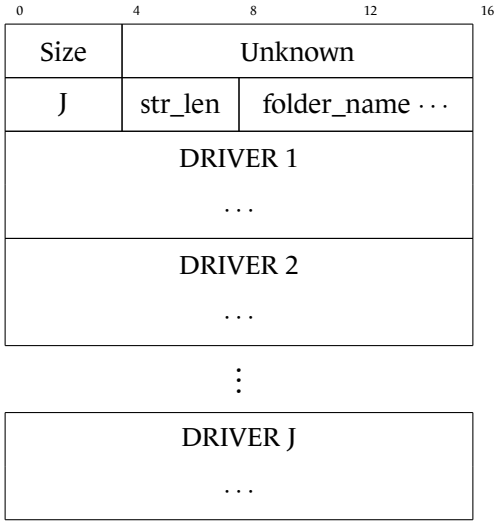


Fig. G.2.: PATH byte structure

The structure of a `DRIVER` field is described in Fig. G.3. The `Size` field is the size of the `DRIVER` field, in bytes. `K` is the number of `INFO` field.

The structure of an `INFO` field is described in Fig. G.4. The meaning of the first four bytes is unclear but seems to indicate the data type of the information: for a string it is always equal to '0', for a FILETIME timestamp it is always equal to '2' and for every other data type, it is '1' (int, UNIX timestamp, ...). The `info_type` indicates what kind of information about the driver follows. The `size_info` field is the size of `driver_info`, in bytes.

The different value for `info_type` are as follows:

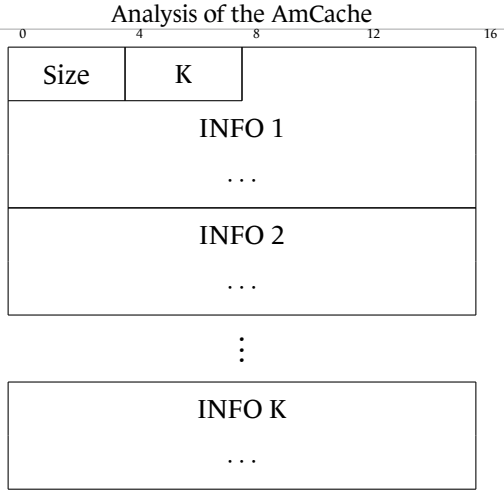


Fig. G.3.: DRIVER byte structure

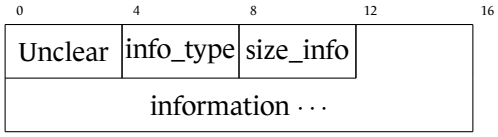


Fig. G.4.: INFO byte structure

- 0x0 = Filename in lowercase;
- 0x1 = SHA-1, preceded by ’0000’;
- 0x2 = Windows Driver Framework version;
- 0x3 = Version;
- 0x4 = Company;
- 0x5 = ProductName;
- 0x6 = ProductVersion;
- 0x7 = Full path of the certificate on the system;
- 0x8 = Signer;
- 0x9 = MajorImageVersion appended to the MinorImageVersion;
- 0xa = Compilation date in Unix format;
- 0xb = Checksum found in the Optional Header of the driver;
- 0xc = Size of image;
- 0xd = Paged size;
- 0xe = Architecture type (first three bits of the Type field in Table F.7);
- 0xf = Djb2 hash of the filename;
- 0x10 = Name of the associated service;
- 0x11 = Date of Modification, in FILETIME format;
- 0x12 = State;
- 0x13 = Driver Type (bits 17 to 23 of the Type field in Table F.7);
- 0x14 = Signature type (bits 4 to 16 and 24 of the Type field in Table F.7)

H. FullCompatReport structure

```
<CompatReport> <!-- Information about the report. Attributes are described in Table G.1 -->
  <System> <!-- Information about the system. Attributes are described in Table G.2 -->
    <Version> <!-- Information about the OS Version. Attributes are described in Table G.3 -->
    </Version>
    <Machine>
    </Machine>
    <SdbInfo> <!-- Information about the sdb files. For each file, attributes are FileSize and CreationDateTime -->
      <SysMain32Sdb>
      </SysMain32Sdb>
      <SysMain64Sdb>
      </SysMain64Sdb>
      <DrvMainSdb>
      </DrvMainSdb>
      <DrvMainSdb64>
      </DrvMainSdb64>
      <DrvMainSdArm>
      </DrvMainSdbArm>
    </SdbInfo>
  </System>
  <Hardware> <!-- Information about the hardware (processor type, vendor, ...). -->
    <HardwareItem>
      <CompatibilityInfo>
      </CompatibilityInfo>
    </HardwareItem>
    [...]
  </Hardware>
  <Plugins> <!-- Plugin for the system (SecureBoot for example) -->
    <Plugin>
      <CompatibilityInfo>
      </CompatibilityInfo>
    </Plugin>
    [...]
  </Plugins>
  <Devices> <!-- List of physical devices that were plugged on the system. -->
    <DeviceInventoryPerfData>
    </DeviceInventoryPerfData>
    <Device>
      <HardwareIds>
        <HwId>
        </HwId>
        [...]
      </HardwareIds>
      <CompatibleIds>
      </CompatibleIds>
      <InstalledDriver>
      </InstalledDriver>
      <CompatibilityInfo>
      </CompatibilityInfo>
    </Device>
    [...]
  </Devices>
  <Programs> <!-- Information about the installed programs. -->
    <Program> <!-- Information about the program. Attributes are described in Table G.4 -->
      <CompatibilityInfo> <!-- Supposedly information about compatibility for the program. No significant data found in tests -->
      </CompatibilityInfo>
      <ClrVersionsFound> <!-- Always empty in tests -->
      </ClrVersionsFound>
    </Program>
    [...]
  </Programs>
  <Usage>
  </Usage>
  <Performance>
  </Performance>
  <ProgramBlockList>
  </ProgramBlockList>
  <DeviceBiosBlockList>
  </DeviceBiosBlockList>
  <Drivers> <!-- List of drivers installed on the system -->
    <Driver> <!-- Information about the installed driver. Attributes are described in Table G.5 -->
    </Driver>
    [...]
  </Drivers>
  <DeviceContainers> <!-- List of physical devices that were plugged on the system. -->
    <Container>
    </Container>
    [...]
```

```
</DeviceContainers>
<IEAddOnList> <!-- List of Internet explorer add-ons -->
  <IEAddOn> <!-- Information about the installed add-on. Attributes are described in Table D.11 -->
    <File> <!-- Information about the PE that provides the add-on. Attributes are described in Table D.12 -->
    </File>
  </IEAddOn>
  [...]
</IEAddOnList>
<DriverPackages> <!-- List of drivers installation files -->
  <DriverPackage> <!-- Information about the installation file. -->
    <InfSection> <!-- Optional. Information about the service that uses the driver. -->
    </InfSection>
    [...]
  </DriverPackage>
  [...]
</DriverPackages>
<GeneralTelemetry> <!-- Telemetry information -->
  <AdvertisingID>
    <TelemetryData>
    </TelemetryData>
  </AdvertisingID>
  <ChromeOSLaunchMode>
    <TelemetryData>
    </TelemetryData>
  </ChromeOSLaunchMode>
  <DVDTelemetrySessionStartDate>
    <TelemetryData>
    </TelemetryData>
  </DVDTelemetrySessionStartDate>
  <OTHER-CDROM-DVDTelemetrySessionCount>
    <TelemetryData>
    </TelemetryData>
  </OTHER-CDROM-DVDTelemetrySessionCount>
  <OTHER-CDROM-DVDTelemetrySessionDuration>
    <TelemetryData>
    </TelemetryData>
  </OTHER-CDROM-DVDTelemetrySessionDuration>
  <OTHER-DISK-DVDTelemetrySessionCount>
    <TelemetryData>
    </TelemetryData>
  </OTHER-DISK-DVDTelemetrySessionCount>
  <OTHER-DISK-DVDTelemetrySessionDuration>
    <TelemetryData>
    </TelemetryData>
  </OTHER-DISK-DVDTelemetrySessionDuration>
  <TelemetryData>
  </TelemetryData>
  [...]
  <UserDefaultBrowser>
    <TelemetryData>
    </TelemetryData>
  </UserDefaultBrowser>
  <UserHttpHandler>
    <TelemetryData>
    </TelemetryData>
  </UserHttpHandler>
  <UserUILanguages>
    <TelemetryData>
    </TelemetryData>
  </UserUILanguages>
  <WMC-CDROM-DVDTelemetrySessionCount>
    <TelemetryData>
    </TelemetryData>
  </WMC-CDROM-DVDTelemetrySessionCount>
  <WMC-CDROM-DVDTelemetrySessionDuration>
    <TelemetryData>
    </TelemetryData>
  </WMC-CDROM-DVDTelemetrySessionDuration>
  <WMC-DISK-DVDTelemetrySessionCount>
    <TelemetryData>
    </TelemetryData>
  </WMC-DISK-DVDTelemetrySessionCount>
  <WMC-DISK-DVDTelemetrySessionDuration>
    <TelemetryData>
    </TelemetryData>
  </WMC-DISK-DVDTelemetrySessionDuration>
  <WMP-CDROM-DVDTelemetrySessionCount>
    <TelemetryData>
```

```
    </TelemetryData>
</WMP−CDROM−DVDTelemetrySessionCount>
<WMP−CDROM−DVDTelemetrySessionDuration>
    <TelemetryData>
    </TelemetryData>
</WMP−CDROM−DVDTelemetrySessionDuration>
<WMP−DISK−DVDTelemetrySessionCount>
    <TelemetryData>
    </TelemetryData>
</WMP−DISK−DVDTelemetrySessionCount>
<WMP−DISK−DVDTelemetrySessionDuration>
    <TelemetryData>
    </TelemetryData>
</WMP−DISK−DVDTelemetrySessionDuration>
<InstalledHotfixesQuery> <!−− Information about the installed hotfixes. −−>
    <InstalledHotfixesData> <!−− Attributes are HotFixID and date of installation (InstalledOn) −−>
    </InstalledHotfixesData>
    [...]
</InstalledHotfixesQuery>
<ServicesQuery> <!−− List of services −−>
    <ServicesData> <!−− Information about the service. Attributes are described in Table G.6 −−>
    </ServicesData>
    [...]
</ServicesQuery>
<DiskInfoQuery>
    <DiskInfoData>
    </DiskInfoData>
</DiskInfoQuery>
<VolumeInfoQuery> <!−− Information about the mounted volumes (drive letter, space, ...). −−>
    <VolumeInfoData>
    </VolumeInfoData>
    [...]
</VolumeInfoQuery>
<DiskPartitionInfoQuery>
    <DiskPartitionInfoData>
    </DiskPartitionInfoData>
    [...]
</DiskPartitionInfoQuery>
<PhysicalDiskInfoQuery>
    <PhysicalDiskInfoData>
    </PhysicalDiskInfoData>
    [...]
</PhysicalDiskInfoQuery>
<PrimaryMonitorQuery>
    <PrimaryMonitorData>
    </PrimaryMonitorData>
</PrimaryMonitorQuery>
<VolumeLicenseQuery>
</VolumeLicenseQuery>
<ProcessorInformationQuery>
    <ProcessorInformationData>
    </ProcessorInformationData>
</ProcessorInformationQuery>
<PCSystemTypeQuery>
    <PCSystemTypeData>
    </PCSystemTypeData>
</PCSystemTypeQuery>
<CurrentPowerPolicyQuery>
    <CurrentPowerPolicyData>
    </CurrentPowerPolicyData>
</CurrentPowerPolicyQuery>
<WifiTelemetryDaya>
</WifiTelemetryData>
<InstalledUILanguages>
    <UILanguage>
    </UILanguage>
    [...]
</InstalledUILanguages>
<WindowsGenuineTelemetryData>
</WindowsGenuineTelemetryData>
<BootConfig>
    <BootEntry>
    </BootEntry>
    [...]
</BootConfig>
<SupportedGraphicsDXVersion>
</SupportedGraphicsDXVersion>
<CpuIdData>
```

```
</CpuIdData>
<UserBrowserSearchSettings>
  <TelemetryData>
  </TelemetryData>
</UserBrowserSearchSettings>
<UserBrowserHomepage>
  <TelemetryData>
  </TelemetryData>
</UserBrowserHomepage>
<WiDiConnection>
</WiDiConnection>
<LastSyncTimeItems>
</LastSyncTimeItems>
<RedirectedProfiles>
  <Directory>
  </Directory>
  [...]
</RedirectedProfiles>
<ChromeApps>
</ChromeApps>
<StartupApplications> <!-- List of startup applications -->
  <Application>
  </Application>
  [...]
</StartupApplications>
<FirmwareTypeData>
</FirmwareTypeData>
<WinSAT>
  <Metrics>
    <CPUMetrics>
      <CompressionMetric>
      </CompressionMetric>
      <EncryptionMetric>
      </EncryptionMetric>
      <CPUCompression2Metric>
      </CPUCompression2Metric>
      <Encryption2Metric>
      </Encryption2Metric>
      <CompressionMetricUP>
      </CompressionMetricUP>
      <EncryptionMetricUP>
      </EncryptionMetricUP>
      <CPUCompression2MetricUP>
      </CPUCompression2MetricUP>
      <Encryption2MetricUP>
      </Encryption2MetricUP>
      <DshowEncodeTime>
      </DshowEncodeTime>
    </CPUMetrics>
    <MemoryMetrics>
      <Bandwidth>
      </Bandwidth>
    </MemoryMetrics>
    <GamingMetrics>
      <BatchFps>
      </BatchFps>
      [...]
      <AlphaFps>
      </AlphaFps>
      [...]
      <TexFps>
      </TexFps>
      [...]
      <ALUFps>
      </ALUFps>
      [...]
      <GeomF4>
      </GeomF4>
      <GeomV8>
      </GeomV8>
      <CBuffer>
      </CBuffer>
    </GamingMetrics>
    <GraphicsMetrics>
      <DWMFps>
      </DWMFps>
      <VideoMemBandwidth>
      </VideoMemBandwidth>
```

```
            <MFVideoDecodeDur>
            </MFVideoDecodeDur>
          </GraphicsMetrics>
          <VideoDecodeMetrics>
            <DecodeFrameCount>
            </DecodeFrameCount>
            [...]
          </VideoDecodeMetrics>
          <DiskMetrics>
            <AvgThroughput>
            </AvgThroughput>
            [...]
          </DiskMetrics>
        </Metrics>
      </WinSAT>
      <WindowsLicensing>
      </WindowsLicensing>
      <SleepStatesSupported>
      </SleepStatesSupported>
      <TelemetryData>
      </TelemetryData>
      <ChromeRlz>
        <Rlz>
        </Rlz>
      </ChromeRlz>
      <MicrophoneInfo>
      </MicrophoneInfo>
      <CBSErrorInfo>
      </CBSErrorInfo>
      <PreviousUpgradesInfo>
      </PreviousUpgradesInfo>
      <CompatibilityImpactData> <!-- Information about programs that needed compatibility fixes -->
        <CITRecord>
          <SystemData>
          </SystemData>
          <ProgramData>
            <ProgramImpact>
              <FileImpact> <!-- Information about the exe file. Attributes are described in Table G.7 -->
              </FileImpact>
              [...]
            </ProgramImpact>
            [...]
          </ProgramData>
        </CITRecord>
        [...]
      </CompatibilityImpactData>
    </GeneralTelemetry>
    <ProgramUseList> <!-- Information about programs usage. Only attribute is SnapshotTime -->
      <ProgramUse> <!-- Information about the program. Only attribute is Id. -->
        <FileUse> <!-- Information about the exe file launched. Attributes are described in Table G.8 -->
          <LaunchInfo> <!-- Information about the launches. Attributes are described in Table G.9 -->
          </LaunchInfo>
        </FileUse>
        [...]
      </ProgramUse>
      [...]
    </ProgramUseList>
</CompatReport>
```

H.1: Generic structure of FullCompatReport.xml

Table H.1.: CompatReport attributes

Attribute	Description	Example
MID	Unknown	MID="A1990A22-112B-4D0F-BB3B-625E66C092E7"
ReportScenario	Unknown	ReportScenario="PDU_WICA"

CensusId	Unknown	CensusId="{BB91F828-924E-4EBF-9EF3-63D97DF630EF}"
Version	Unknown	Version="1.6"
UpgradeEligible	Unknown	UpgradeEligible="1"
OfflineScan	Unknown	OfflineScan="1"
IeVersion	Version of installed Internet Explorer	IeVersion="9.11.9600.17416"
SqmId	Unknown	SqmId="{C70723D9-91ED-4AA4-9EF6-E0FB9035C335}"
RacId	Unknown	RacId="{819CA618-5513-405F-99BD-880AAF707895}"
WuId	Unknown	WuId="dd0fd1d3-1288-4914-bb3e-e17c8b03613d"
GeoId		GeoId="84"

Table H.2.: System attributes

Attribute	Description	Example
X64Capable		X64Capable="True"
X64Running		X64Running="True"
KnownWorkingCount	Unknown	KnownWorkingCount="54"
WontWorkIssueCount	Unknown	WontWorkIssueCount="0"
RequireActionIssueCount	Unknown	RequireActionIssueCount="0"
ComplianceIssuesCount	Unknown	ComplianceIssuesCount="0"

BlockUpgradeIssueCount	Unknown	BlockUpgradeIssueCount="0"
BlockUpgradeCanReinstallCount	Unknown	BlockUpgradeCanReinstallCount="0"
BlockUpgradeUntilUpdateCount	Unknown	BlockUpgradeUntilUpdateCount="0"
DismissableIssueCount	Unknown	DismissableIssueCount="0"
HardBlockedDevicesCount	Unknown	HardBlockedDevicesCount="0"
TotalIssueCount	Unknown	TotalIssueCount="0"
TotalAppCount	Number of installed programs	TotalAppCount="4"
TotalDeviceCount		TotalDeviceCount="57"
SocBlock	Unknown	SocBlock="false"
OSArch		OSArch="x64"
UserLocale	User settings for dates, times,... based on the language pack identifier[1]	UserLocale="1036"
TargetBuild		TargetBuild="9600"
Edition		Edition="Microsoft Windows 8.1 Professionnel"
Manufacturer		Manufacturer="innotek GmbH"
Model		Model="VirtualBox"

[1] https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-8.1-and-8/hh825678(v=win.10)#language-packs

Table H.3.: Version attributes

Attribute	Description	Example
Major		Major="6"
Minor		Minor="3"
ServicePackMajor		ServicePackMajor="0"
ServicePackMinor		ServicePackMinor="0"
Build		Build="9600"

Table H.4.: Program attributes

Attribute	Description	Example
Name		Name="Wireshark 2.6.5 64—bit"
Version		Version="2.6.5"
Publisher		Publisher="The Wireshark developer community, https://www.wireshark.org"
Type	Only value seen: "Application"	Type="Application"
Source	"AddRemoveProgram" or "Msi"	Source="AddRemoveProgram"
Id	ProgramId	Id="0000c16b47f8ca21d3ca3f3ace1abb7c51e40000ffff"

Table H.5.: Driver attributes

Attribute	Description	Example
DriverName		DriverName="1394ohci.sys"

DriverCompany		DriverCompany="Microsoft Corporation"
DriverId	SHA-1 of the driver, preceded by '0000'	DriverId="0000f000843ae742b251f0f3b2dd3629fd4803d1609b"
DriverCheckSum		DriverCheckSum="294388"
DriverTimeStamp	Compilation date, in UNIX format	DriverTimeStamp="1377171494"
DriverType	Unknown	DriverType="8650778"
DriverVersion		DriverVersion="6.3.9600.16384"

Table H.6.: ServicesData attributes

Attribute	Description	Example
Name		Name="AeLookupSvc"
State	`Running` or `Stopped`	State="Running"
StartMode		StartMode="Manual"
PathName		PathName="svchost.exe −k netsvcs"
DisplayName	Display name in the User-Locale language	DisplayName="Expérience 'dapplication"

Table H.7.: FileImpact attributes

Attribute	Description	Example
Name	File name in upper case	Name="WIRESHARK.EXE"
Id	SHA-1, preceded by '0000'	Id="00003c742e7d9ff40c291d5c1d2a9aa6c9d3b2023a34"

TimeStamp	Date of compilation in UNIX format and in hexadecimal	TimeStamp="5bfee034"
Checksum		Checksum="75a2e7"
Type	Unknown	Type="0"
ImpactData1	Unknown	ImpactData1="AAAAAAQAAAAAAAAAAAAEAAEAAAAAAAAA"
ImpactData2	Unknown	ImpactData2="AQAAAG51AAABAAAA+0 EAAAAAAAD7QQAAAAAAAPtBAAAAAAAAAAAAAAAEAAADGQgAA"
ImpactData3	Unknown	ImpactData3="AAAAAAAAAAAAAAAAAQAAAAAAAA"

Table H.8.: FileUse attributes

Attribute	Description	Example
Name	File name, in uppercase	Name="ICAT.EXE"
Id	SHA-1, preceded by '0000'	Id="00006b0c143e12d71685e752d8119219632281d3194b"

Table H.9.: LaunchInfo attributes

Attribute	Description	Example
LaunchId	Unknown	LaunchId="474BABE2"
LaunchCount		LaunchCount="14"
FirstLaunchTime		FirstLaunchTime="01/11/2019 09:14:59"
LastLaunchTime		LastLaunchTime="01/15/2019 09:58:32"

Bibliography

[1] Harrell, C (2013). *Revealing the RecentFileCache.bcf File*, http://journeyintoir.blogspot.com/2013/12/revealing-recentfilecachebcf-file.html.

[2] Ionescu, A (2007). *Secrets of the Application Compatibility Database (SDB) - Part 1*, http://www.alex-ionescu.com/?p=39.

[3] Khatri, Y (2013). *Amcache.hve in Windows 8 - Goldmine for malware hunters*, http://www.swiftforensics.com/2013/12/amcachehve-in-windows-8-goldmine-for.html.

[4] Khatri, Y (2013). *Amcache.hve - Part 2*, http://www.swiftforensics.com/2013/12/amcachehve-part-2.html.

[5] Microsoft (2012). *Understanding Shims*, https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-7/dd837644(v=ws.10).

[6] Suhanov, M (2018). *The CIT database and the Syscache hive*, https://dfir.ru/2018/12/02/the-cit-database-and-the-syscache-hive/.