ANALYSIS OF THE AmCACHE

Author  Blanche Lagny
Publication date  21/01/2019
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.
Introduction

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

Behavior of libraries originally packaged with Windows 8.0 and Server 2012

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

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

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

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

Conclusion

Appendix A Artifact location summary

Appendix B RecentFileCache.bcf

Appendix C AEINV_PREVIOUS structure

Appendix D AEINV_WER structure

Appendix E AEINV_AMI_WER structure

Appendix F PropCache.bin structure

Appendix G FullCompatReport structure

Bibliography
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 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 %SystemDrive%\%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.

---

4. https://github.com/keydet89/RegRipper2.8
Analysis of the AmCache

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 %SystemDrive%\%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 %SystemDrive%\%WinDir%\AppCompat\Programs. The format of this file is described in Appendix B. The service also stores path of the dependencies of the executed PE which need shimming.

Then, at 00:30 every night, a scheduled task, ProgramDataUpdater, is executed. This task launches "%SystemDrive%\%WinDir%\system32\rundll32.exe aepdu.dll,AePduRunUpdate", which flushes the RecentFileCache.bcf and stores all installed programs in %SystemDrive%\%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_WEB_{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\test, but the very same PE located in C:\Users\<username>\Documents\test, but the very same PE located in C:\Users\<username>\Documents\test 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](https://example.com/fig2.1.png)

```xml
<Log Version="6.1.7601.17514">
  <ProgramList>
    <Program>
      <Program Id="0000354384b22dbc26b2dc9dec22174df510000ffff" 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>
  </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: File 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.

```xml
<Log Version="6.1.7601.17514">
  <ProgramList>
    [...]
    <Program Id="00009056f81453a3569d36c40c2a6152d9640000904" 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.

Analysis of the AmCache

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

```xml
<Log Version="6.1.7601.17514">
[...]
<IEAddOnList>
<IEAddOn CLSID="{19916E01-B44E-4E31-94A4-4696DF46157B}" Name="InformationCardSigninHelper Class" Type="ActiveX" Publisher="Microsoft Corporation">
  <File Id="0000d8b095849b5172e07dffe1562bad89f37037bf951" 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 C.

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 D 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
Analysis of the AmCache 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}"/>
<br />
| 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 D 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
Analysis of the AmCache

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.

```
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.

```
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.

```
2.8: Extract of AEINV_WER_{49A35C5F-CCE9-48C7-B6EF-577A36E86135}_20181206_094337.xml:
Installed Program ShellIndicators
```
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 capifos.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 uniqueld="0x23" id="00009afddc213e845b1ed280a8d118317c363e807da5"/>
<directory uniqueld="0x65" id="0000c1920deeef6a4453b87d82e94b4d7cd7e34cfa"/>
<directory uniqueld="0x6b" id="0000f985ceec5256e326b80e61528e85f1d06039299"/>
<directory uniqueld="0x6d" id="00001835aab95f61091a75c2668c32f3acc6b6b39f3c"/>
<directory uniqueld="0x77" id="00002981edfd070ae25ff64b4632dd1d8e88bbaa3d3"/>
<directory uniqueld="0x7b" id="0000fa4622b722e71a460e2fc47d59bf7dceb30c5"/>
```
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.

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>
  <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">
    <Evidence Id="0x22" Id="000054384B2d9bc2f6b2d9dec22174dc51000fffc">
      <Indicators>
        [...]
        <Program>
          <Files>
            <File Name="capinfos.exe" Id="00005c5c87f4d4e6f50b186109b2c18b47f257365" ProductName="Capinfos" CompanyName="The Wireshark developer community" ProductVersion="2.6.5" VerLanguage="1033" SwitchBackContext="0x0100000000000600"FileVersion="2.6.5" Size="0x532a8" SizeOfImage="0x53000" PeHeaderHash="01012864b33151873a9ca2ddc0c5e28d87cfc0b23f03" PeChecksum="0x5f3e34" BinProductVersion="2.6.5.0" BinFileVersion="2.6.5.0">
                  <LongPathHash="000058d47d02b18994a27e38ea102c658e68e3b18ed3" UniqueId="0x24"/>
              </BinaryType>
            </FileDescription>
          </File>
          <File Name="comerr32.dll" Id="0001c2479e4491059fe4df5f737488104d9a9a4e62e2e2" ProductName="comerr32.dll" CompanyName="Massachusetts Institute of Technology." Version="1.6.3.16" kfw="--3.2.2" VerLanguage="1053" SwitchBackContext="0x0100000000000400" FileVersion="1.6.3.16">
            <FileDescription="COM_ERR – Common Error Handler for MIT Kerberos v5 / GSS" LinerVersion="6.0" LinkDate="01/18/2010 17:01:38">
                <LongPathHash="0000b3d0b9a5s5181478c8135b5afaddde664f683d1bed66" UniqueId="0x25"/>
              </BinaryType>
            </FileDescription>
          </File>
          <File Name="d3dcompiler_47.dll" Id="0000b29e74577085c41637b1cc7a14e1835264417a" ProductName="Microsoft® Windows® Operating System" CompanyName="Microsoft Corporation" ProductVersion="10.0.16299.15" VerLanguage="1033" ShortName="D3DCompiler-1.11">
            <SwitchBackContext="0x0100000000000600" FileVersion="10.0.16299.15">Windows\10.0.16299.15\WinBuild.160101.0800</FileVersion="10.0.16299.15">
              <Size="0x37d4a8" SizeofImage="0x0386000" PeHeaderHash="0101a7f2a4e9e1d735b13562316f872442f2a626053" PeChecksum="0x037e544" BinProductVersion="10.0.16299.15" BinFileVersion="10.0.16299.15" FileDescription="Direct3D HLSL Compiler for Redistribution" LinerVersion="14.10" LinkDate="10/19/2014 09:23:28">
                  <LongPathHash="0000ac76002f76c0ce9eb6dee7c392ad6b246256a0f" UniqueId="0x26"/>
                </BinaryType>
              </BinaryType>
            </SwitchBackContext>
          </File>
          [...]
        </Files>
      </Program>
    </Program>
  </Program>
</ProgramList>
```

2.11: Extract of AEINV_WER_{49A35C5F-CCCE9-48C7-B6EF-577A36E86135}_20181206_094337.xml: Installed Program Files

In addition to the structure of AEINV_WER, Appendix D 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 shimmering 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 file execute in a folder. However, an entry for Wireshark appeared in the Updated list with the following differences:

```
21/01/2019 Page 13 of 58
```
• 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

```
2.12: Extract of AEINV_WER_{49A35C5F-CCF9-4B5C-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.

```
2.13: Extract of AEINV_WER_{49A35C5F-CCF9-4B5C-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 fstat.exe and tree.com are shown in Listing 2.14.

```
2.14: Extract of AEINV_WER_{49A35C5F-CCF9-4B5C-B6EF-577A36E86135}_20181206_094337.xml:
Orphan
```

21/01/2019
A new PE appears under one of those folders, it creates a new entry for the program in the
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 "$\%SystemDrive\%\%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 %SystemDrive%\%WinDir\%AppCompat\Programs. However, if the executed PE is an installer for a program, it is handled by a different service: PCASvc, which executes %SystemDrive%\%WinDir%\%system32\svchost.exe -k LocalSystemNetworkRestricted. This service calls the following command: "rundll32.exe aeinv.dll, UpdateSoftwareInventory". This DLL creates a TXT file in %SystemDrive%\%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 "%SystemDrive%\%WinDir%\%system32\rundll32.exe aepdu.dll, AepduRunUpdate" which deletes the files under %SystemDrive%\%WinDir%\%AppCompat\Programs\Install and stores all installed programs in %SystemDrive%\%WinDir%\%AppCompat\Programs\AEINV_CURRENT.xml. It then renames this file as AEINV_PREVIOUS.xml, overwriting the previous file.

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 File sublist previously seen in the AEINV_WER, each 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:

- **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;
Analysis of the AmCache

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

- 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;

- 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.

<table>
<thead>
<tr>
<th>Value Name</th>
<th>Value Type</th>
<th>Data</th>
<th>Value Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RegStr</td>
<td>Wireshark 2.6.5 32bit</td>
<td>00-00-00-00-00-00</td>
</tr>
<tr>
<td>1</td>
<td>RegDword</td>
<td>2.6.5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RegDword</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>RegDword</td>
<td>The Wireshark developer community, <a href="https://www.wireshark.org">https://www.wireshark.org</a></td>
<td>00-00</td>
</tr>
<tr>
<td>4</td>
<td>RegDword</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>RegDword</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>RegDword</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>RegOrg</td>
<td>AddRemoveProgram</td>
<td>00-00</td>
</tr>
<tr>
<td>8</td>
<td>RegOrg</td>
<td>1594850037</td>
<td>50-43-02-00</td>
</tr>
<tr>
<td>9</td>
<td>RegOrg</td>
<td>0</td>
<td>50-43-02-00</td>
</tr>
<tr>
<td>d</td>
<td>RegOrg</td>
<td>C:\Program Files\Wireshark\C:Program Files\Wireshark\audio\C:\Program Files\Wireshark\be...</td>
<td>35-02-02-02-39-03</td>
</tr>
<tr>
<td>Files</td>
<td>RegOrg</td>
<td>1b28a002-0b75-11a6-a9b2-806456464e53</td>
<td>00-00-00-00-00-00</td>
</tr>
</tbody>
</table>

Fig. 3.4.: Content of AmCache.hve\Root\Programs\0000354384b2dbd2f6b2c9dec22174dc510000fff

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.

![Content of AmCache.hve\Root\Generic\0\000002da97a4940b126c7710d13b431a6e74123f3cc0](image)

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.
Analysis of the AmCache

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="0004021d62bcd80dc4a5ac67b8fbfd91516395084b5" SetupScriptChecksum="1723113644929021051" 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 meaning of all the other attributes are unknown, Size not being the size of neither the full disc nor the used space.

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 E 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, 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-05cd-7eb0-0ef7-6df88e1a34f}"
DeviceDataId="{taedc93f-bfeb-9f36-826e-71bf7ee6fde1}"
ModelId="{2bda71a3-65a7-1c33-dd60-e2630bc8452b}"
ModelName="USB_DISK 2.0"
IsMachineDevice="false"
PrimaryCategory="storage">
<Categories>
  <Category Id="storage"/>
</Categories>
<Device DeviceId="{703078f-7727-31f1-7c73-ef76f510ef76f5161a34f}"
  Enumerator="usb"
  DeviceOrder="0">
  <HardwareIds>
    <HardwareId Id="vid_13fe&pid_4200&rev_0100"
      Order="0"></HardwareId>
    <HardwareId Id="vid_13fe&pid_4200"
      Order="1"></HardwareId>
  </HardwareIds>
  <CompatibleIds>
    <CompatibleId Id="class_08&subclass_06&prot_50"
      Order="0"></CompatibleId>
    <CompatibleId Id="class_08&subclass_06"
      Order="1"></CompatibleId>
    <CompatibleId Id="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](Image)

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="000002da97a4940b126c771d13b431a6e74123f3c00"
  Name="1394ohci.sys"
  Type="0x0004001a"
  Version="6.2.9200.16384"
  Timestamp="0x5010aad6"
  CheckSum="0x00047021"
  ImageSize="0x0003d000"
  PagedSize="0x0000e000"
  Company="Microsoft Corporation"
  Product="Microsoft® Windows® Operating System"
  ProductVersion="6.2.9200.16384">
```

1https://docs.microsoft.com/en-us/windows-hardware/drivers/install/container-ids
The meaning of the different attributes are as follows:

- **DriverId**: SHA-1 of the driver, preceded by '0000';
- **Name**: Filename;
- **Type**: Unknown;
- **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 F.

### 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 %SystemDrive%\%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**.
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 "%SystemDrive%\%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 does not change between the two versions, 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.

```xml
<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 G 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 services 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 occurring 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="0000f519f0ecc486de87ed73cb92d3cac802400000000">
    [...]
    <FileUse Name="CMD.EXE" Id="00007c3d7281e1151fe4127923f4b4c3d36438e1a12">
      <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 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 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 a major change: the only XML files left are the one inside the Install directory. Another major change is the disappearance of both aepdu.dll and the AeLookupSvc service. The first one seems to have been replaced by compattelrunner.exe, whereas the second one seems to have been replaced by the DiagTrack service.

5.1. General behavior

When executing a PE, the service DiagTrack, which executes "%SystemDrive%\%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. However, if the executed PE is an installer for a program, it is handled by the service PCASvc, which executes "%SystemDrive%\%WinDir%\system32\svchost.exe -k LocalSystemNetworkRestricted". This service runs "%SystemDrive%\%WinDir%\system32\compattelrunner.exe -m:aeinv.dll -f:UpdateSoftwareInventory". The DLL aeinv.dll creates a TXT file in %SystemDrive%\%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 disappearance of aepdu.dll, the two scheduled tasks, ProgramDataUpdater and Microsoft Compatibility Appraiser, respectively launch "%SystemDrive%\%WinDir%\system32\compattelrunner.exe -maintenance" and "%SystemDrive%\%WinDir%\system32\compattelrunner.exe". Those tasks seem to update neither AmCache.hve nor any XML file. They do not delete the content of the Install directory either.

5.2. AmCache.hve

In this version, Microsoft added three keys in AmCache.hve: Device, HwItem and Metadata. In 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. Moreover, 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. As opposed to what was described in Subsection 3.2.1, the key associated with "internal" PEs does here seem to have a last write time coinciding with their first execution time.

In this version, 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.

![Fig. 5.1: Content of AmCache.hve\Root\Programs\0000354384b2dbfdb2f6b2dc9dec222174df510000ffff](image)

None of the meanings of those values have been found yet. The 16 value only appeared twice during tests, for Microsoft Visual C++. 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.
5.3. Examples of possible uses during a forensic investigation

In this version of the AmCache, all XML files except the ones under %SystemDrive%\%WinDir%\AppCompat\Programs\Install disappeared. This implies that only the uses described in Section 3.6 that involve AmCache.hve or the INSTALL files can apply. Evidence of the presence of a binary file can be found under the File key in AmCache.hve. Moreover if the PE is referenced under the Orphan key, it proves that it was executed. 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. Nevertheless, the Generic key being empty, no information pertaining to driver installation can be retrieved from AmCache.hve anymore.

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.

6. 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 of 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.

6.1. General behavior

When executing a PE, the service DiagTrack, which executes "%SystemDrive%\%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. However, if the executed PE is an installer for a program, it is handled by the service PCASvc, which executes "%SystemDrive%\%WinDir%\system32\svchost.exe -k LocalSystemNetworkRestricted". This service runs "%SystemDrive%\%WinDir%\%system32\compattelrunner.exe -m:aeinv.dll -f:UpdateSoftwareInventory".

The two scheduled tasks behave similarly to what was observed in Section 5.1. However, Microsoft Compatibility Appraiser updates a new file, APPRAISER_FileInventory.xml, located under %SystemDrive%\%WinDir%\appcompat\appraiser.

6.2. 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 6.1, 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.
6.3. 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 5.2, 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. 6.1 for 1394ohci.sys.

![Fig. 6.1: AmCache.hve\Root\InventoryDriverBinary\0000895407cb018368e62fc360b972a8b0da7e729662](image)

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, records EXE files that are part of a program. This last key is exclusively updated by Microsoft Compatibility Appraiser. Secondly, regarding program activity, the key Programs is solely updated when ProgramDataUpdater runs, while it was previously updated by PCASvc at the time of installation of the program. PCASvc now records the installation or removal of a program in the key InventoryApplication. This key also contains programs installed via an AppXPackage.
Analysis of the AmCache

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 those two new keys are 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. 6.2.

![Fig. 6.2: AmCache.hve\Root\InventoryApplicationFile\0000cf4a8522cabda2c91c44e2510550f58b6983cdd5](image)

As for InventoryApplication, each program entry is named after its ProgramId, and the values are once again easily understandable, as shown in Fig. 6.3.

![Fig. 6.3: AmCache.hve\Root\InventoryApplication\0000c16b47f8ca21d3ca3f3ace1abb7c51e40000fffe](image)

6.4. 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 5.3.

Fortunately for the forensic investigator, this version of AmCache.hve marks the return of the data that was missing in the previous chapter, due to the elimination of the XML files. 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.
Analysis of the AmCache

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.

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

This chapter is currently under review and will be described in a future version of this paper.

8. 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, e.g. concerning USB usage, it seems relevant to further study this artifact. This will certainly prove quite beneficial to the digital forensics community.

21/01/2019 Page 30 of 58
A. Artifact location summary

Table A.1: Artifacts

<table>
<thead>
<tr>
<th>Version</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1.7600 and 6.1.7601</td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\RecentFileCache.bcf</td>
</tr>
<tr>
<td></td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\AEINV_PREVIOUS.xml</td>
</tr>
<tr>
<td></td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\AEINV_AMI_WER_{MachineId}_YYYYMMDD_HHmmss.xml</td>
</tr>
<tr>
<td>6.2.9200 and 6.3.9600.16384</td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\AmCache.hve</td>
</tr>
<tr>
<td></td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\AEINV_PREVIOUS.xml</td>
</tr>
<tr>
<td></td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\AEINV_AMI_WER_{MachineId}_YYYYMMDD_HHmmss.xml</td>
</tr>
<tr>
<td></td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\DevInvCache\PropCache.bin</td>
</tr>
<tr>
<td>6.3.9600.17415</td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\AmCache.hve</td>
</tr>
<tr>
<td></td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\AEINV_PREVIOUS.xml</td>
</tr>
<tr>
<td></td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\AEINV_AMI_WER_{MachineId}_YYYYMMDD_HHmmss.xml</td>
</tr>
<tr>
<td></td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\DevInvCache\PropCache.bin</td>
</tr>
<tr>
<td>10.0.10586</td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\AmCache.hve</td>
</tr>
<tr>
<td></td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\Install\INSTALL_ffff_*_.xml</td>
</tr>
<tr>
<td>10.0.14913</td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\AmCache.hve</td>
</tr>
<tr>
<td></td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\Install\INSTALL_ffff_*_.xml</td>
</tr>
<tr>
<td></td>
<td>APPRAISER\Programs\appraiser\APPRAISER\FileInventory.xml</td>
</tr>
<tr>
<td>10.0.16299</td>
<td>%SystemDrive%%WinDir%\AppCompat\Programs\AmCache.hve</td>
</tr>
<tr>
<td></td>
<td>APPRAISER\Programs\appraiser\APPRAISER\FileInventory.xml</td>
</tr>
<tr>
<td></td>
<td>APPRAISER\Programs\appraiser\APPRAISER\TelemetryBaseline_UNV.bin</td>
</tr>
</tbody>
</table>

B. RecentFileCache.bcf

The general structure of the file is described in Fig. B.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.

```
0 4 8 12 16
Fixed
Unknown Size PATH 1
...

PATH N
...
```

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Unknown</td>
<td>Id=&quot;0000354384b2dbc2f6b2dec9dec22174dc510000ffff&quot;</td>
</tr>
<tr>
<td>MsiproductCode</td>
<td></td>
<td>MsiproductCode=&quot;{C3CC4DF5-39A5-4027-B136-2B3E1F5AB6E2}&quot;</td>
</tr>
<tr>
<td>MsipackageCode</td>
<td></td>
<td>MsipackageCode=&quot;{AE5CF7E6-1FAD-47DF-A41F-3261FBF6B305}&quot;</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>Name=&quot;Wireshark 2.6.5 32-bit&quot;</td>
</tr>
<tr>
<td>Publisher</td>
<td></td>
<td>Publisher=&quot;The Wireshark developer community, <a href="https://www.wireshark.org">https://www.wireshark.org</a>&quot;</td>
</tr>
<tr>
<td>Version</td>
<td></td>
<td>Version=&quot;2.6.5&quot;</td>
</tr>
<tr>
<td>Language</td>
<td>Microsoft Language Id, in decimal (1033 for en-us)(^1)</td>
<td>Language=&quot;1033&quot;</td>
</tr>
<tr>
<td>Source</td>
<td>Ms1 or AddRemoveProgram, depending on whether the program was installed by executing an MSI or a PE.</td>
<td>Source=&quot;AddRemoveProgram&quot;</td>
</tr>
</tbody>
</table>

Table C.2: IEAddOn attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
### Analysis of the AmCache

<table>
<thead>
<tr>
<th>CLSID</th>
<th>CLSID=&quot;{19916E01-B44E-4E31-9A44-469DF46157B}&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name=&quot;InformationCardSigninHelper Class&quot;</td>
</tr>
<tr>
<td>Type</td>
<td>Observed values: ActiveX, BrowserHelperObject and BrowserExtension</td>
</tr>
<tr>
<td>Publisher</td>
<td>Publisher=&quot;Microsoft Corporation&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>SHA-1 preceded by '0000'</td>
<td>Id=&quot;0000d8b095849b5172e07ddf1562bad89f37037bf951&quot;</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>Name=&quot;icardie.dll&quot;</td>
</tr>
</tbody>
</table>

### Table C.3: File attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>SHA-1 preceded by '0000'</td>
<td>Id=&quot;0000d8b095849b5172e07ddf1562bad89f37037bf951&quot;</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>Name=&quot;icardie.dll&quot;</td>
</tr>
</tbody>
</table>

### D. AEINV_WER structure

```xml
<Report> <!-- Information about the file. Attributes are described in Table D.1 -->
  <ProgramList> <!-- Information about the system. Attributes are described in Table D.2 -->
    <Installed> <!-- List of installed programs -->
      <Program> <!-- One entry by program. Attributes are described in Table D.3 -->
        <RegistryIndicators> <!-- List of installed programs -->
          <Registry /> <!-- Information about a Run Key. Attributes are described in Table D.4 -->
        <!--/RegistryIndicators>
      <!--/Program>
    <!--/Installed>
  <!--/ProgramList>
</Report>
```
Analysis of the AmCache

...<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>

...<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>

...<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>

...<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>

D.1: Generic structure of AEINV_WER

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Unknown</td>
<td>Version=&quot;1.3&quot;</td>
</tr>
</tbody>
</table>
### Analysis of the AmCache

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>Finished writing time of the report after the first execution of ProgramDataUpdater in UTC</td>
<td>Timestamp=&quot;12/06/2018 09:43:40&quot;</td>
</tr>
<tr>
<td>SequenceNumber</td>
<td>Unknown</td>
<td>SequenceNumber=&quot;1&quot;</td>
</tr>
<tr>
<td>ThrottlingRuleSetGuid</td>
<td>Unknown</td>
<td>ThrottlingRuleSetGuid=&quot;{F7D0E8C8-2DA8-4889-A910-3DE830B4148F}&quot;</td>
</tr>
</tbody>
</table>

#### Table D.2.: System attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>MachineId</td>
<td>Same ID that the one in the filename</td>
<td>MachineId=&quot;{49A35C5F-CCE9-48C7-B6EF-577A36E86135}&quot;</td>
</tr>
<tr>
<td>MajorVersion</td>
<td>First part of the Windows Version Number</td>
<td>MajorVersion=&quot;6&quot;</td>
</tr>
<tr>
<td>MinorVersion</td>
<td>Second part of the Windows Version Number</td>
<td>MinorVersion=&quot;1&quot;</td>
</tr>
<tr>
<td>ServicePackMajor</td>
<td></td>
<td>ServicePackMajor=&quot;1&quot;</td>
</tr>
<tr>
<td>ServicePackMinor</td>
<td></td>
<td>ServicePackMinor=&quot;0&quot;</td>
</tr>
<tr>
<td>BuildNumber</td>
<td></td>
<td>BuildNumber=&quot;7601&quot;</td>
</tr>
<tr>
<td>Sku</td>
<td>Version of Windows installed as found in the OperatingSystemSKU Enum</td>
<td>Sku=&quot;1&quot;</td>
</tr>
<tr>
<td>ProcessorArchitecture</td>
<td></td>
<td>ProcessorArchitecture=&quot;1&quot;</td>
</tr>
<tr>
<td>OSPlatform</td>
<td>Unknown</td>
<td>OSPlatform=&quot;1&quot;</td>
</tr>
<tr>
<td>LocaleId</td>
<td>decimal value of LocalName</td>
<td>LocaleId=&quot;1033&quot;</td>
</tr>
<tr>
<td>GeoId</td>
<td></td>
<td>GeoId=&quot;244&quot;</td>
</tr>
</tbody>
</table>
### Table D.3.: Program attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td>Name=&quot;Wireshark 2.6.5 32-bit&quot;</td>
</tr>
<tr>
<td>Type</td>
<td>Only value seen: &quot;Application&quot;</td>
<td>Type=&quot;Application&quot;</td>
</tr>
<tr>
<td>Source</td>
<td>Msi or AddRemoveProgram, depending on whether the program was installed by executing an MSI or a PE</td>
<td>Source=&quot;AddRemoveProgram&quot;</td>
</tr>
<tr>
<td>Publisher</td>
<td></td>
<td>Publisher=&quot;The Wireshark developer community, <a href="https://www.wireshark.org">https://www.wireshark.org</a>&quot;</td>
</tr>
<tr>
<td>Version</td>
<td></td>
<td>Version=&quot;2.6.5&quot;</td>
</tr>
<tr>
<td>OnSystemDrive</td>
<td>Unknown</td>
<td>OnSystemDrive=&quot;True&quot;</td>
</tr>
<tr>
<td>EvidenceId</td>
<td>Starting point for indicators described below</td>
<td>EvidenceId=&quot;0x22&quot;</td>
</tr>
<tr>
<td>Id</td>
<td>Unknown</td>
<td>Id=&quot;0000354384b2dbc2f6b2dc9dec22174df510000fff&quot;</td>
</tr>
<tr>
<td>InstallDate</td>
<td>Date of installation. Only present for MSI programs and the time is always 00:00:00</td>
<td>InstallDate=&quot;10/27/2015 00:00:00&quot;</td>
</tr>
<tr>
<td>MsiPackageCode</td>
<td></td>
<td>MsiPackageCode=&quot;{AE5CF7E6-1FAD-47DF-A41F-3261FBF6B305}&quot;</td>
</tr>
<tr>
<td>MsiProductCode</td>
<td></td>
<td>MsiProductCode=&quot;{C3CC4DF5-39A5-4027-B136-2B3E1F5AB6E2}&quot;</td>
</tr>
</tbody>
</table>

### Table D.4.: Registry attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Value of the Run key</td>
<td>Name=&quot;VBoxTray&quot;</td>
</tr>
<tr>
<td>File</td>
<td>Filename contained in the data of the value &lt;Name&gt;</td>
<td>File=&quot;VBoxTray.exe&quot;</td>
</tr>
<tr>
<td>RegistryRun</td>
<td>Unknown</td>
<td>RegistryRun=&quot;Run&quot;</td>
</tr>
</tbody>
</table>
Analysis of the AmCache

Table D.5: AddRemoveProgram attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisplayName</td>
<td>Data contained in the DisplayName value of the Uninstall key</td>
<td>DisplayName=&quot;Wireshark 2.6.5 32-bit&quot;</td>
</tr>
<tr>
<td>CompanyName</td>
<td>Data contained in the Publisher value of the Uninstall key</td>
<td>CompanyName=&quot;The Wireshark developer community, <a href="https://www.wireshark.org">https://www.wireshark.org</a>&quot;</td>
</tr>
<tr>
<td>ProductVersion</td>
<td>Data contained in the DisplayVersion value of the Uninstall key</td>
<td>ProductVersion=&quot;2.6.5&quot;</td>
</tr>
<tr>
<td>RegistrySubKey</td>
<td>Name of the Uninstall key</td>
<td>RegistrySubKey=&quot;Wireshark&quot;</td>
</tr>
<tr>
<td>UniqueId</td>
<td>Unknown but same data as the UniqueId in the Program attribute</td>
<td>UniqueId=&quot;0x22&quot;</td>
</tr>
<tr>
<td>Id</td>
<td>Unknown</td>
<td>Id=&quot;000000e4ecea2abf9c57a7602d5815f5e8809e1e59d&quot;</td>
</tr>
</tbody>
</table>

Table D.6: Shell attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShellName</td>
<td>Name displayed in the Start Menu</td>
<td>ShellName=&quot;Wireshark&quot;</td>
</tr>
<tr>
<td>TargetFileName</td>
<td>Filename of the file executed</td>
<td>TargetFileName=&quot;Wireshark.exe&quot;</td>
</tr>
<tr>
<td>UniqueId</td>
<td>Unknown</td>
<td>UniqueId=&quot;0xa0&quot;</td>
</tr>
<tr>
<td>Id</td>
<td>Unknown</td>
<td>Id=&quot;000008f6fc717280228fa0f0473f0c23d38d23f31f31&quot;</td>
</tr>
</tbody>
</table>
### Table D.7.: MSI attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductName</td>
<td>ProductName=“Python 2.7.6”</td>
<td></td>
</tr>
<tr>
<td>CompanyName</td>
<td>CompanyName=“Python Software Foundation”</td>
<td></td>
</tr>
<tr>
<td>ProductVersion</td>
<td>ProductVersion=“2.7.6150”</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Microsoft Language Id, in decimal (1033 for en-us)</td>
<td>Language=“1033”</td>
</tr>
<tr>
<td>PackageCode</td>
<td>PackageCode=“{AESC7E6-1FAD-47DF-A41F-3261FBF6B305}”</td>
<td></td>
</tr>
<tr>
<td>InstallDate</td>
<td>InstallDate=“10/27/2015 00:00:00”</td>
<td></td>
</tr>
<tr>
<td>UniqeId</td>
<td>Unknown</td>
<td>UniqeId=“0xa”</td>
</tr>
<tr>
<td>Id</td>
<td>Unknown (not the SHA-1 of the MSI)</td>
<td>Id=“0000f58476f702201e0706cb40cd350aad5cc387c133”</td>
</tr>
</tbody>
</table>

### Table D.8.: FileExtensionHandler attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extension</td>
<td>Extension=“.5vw”</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Name=“wireshark-capture-file”</td>
<td></td>
</tr>
<tr>
<td>File</td>
<td>File=“Wireshark.exe”</td>
<td></td>
</tr>
<tr>
<td>UniqeId</td>
<td>Unknown</td>
<td>UniqeId=“0xa6”</td>
</tr>
<tr>
<td>Id</td>
<td>Unknown</td>
<td>Id=“0000f100f0a810d3369f2b3078c6cc2f2a9ae2342793”</td>
</tr>
</tbody>
</table>
### Table D.9: Directory attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>UniqueId</td>
<td>Records where the folder is located in the installation directory, starting with the installation directory itself</td>
<td></td>
</tr>
</tbody>
</table>
UniqueId="0x23"

| Id | Unknown | Id="00009afdcc213e845b1ed280a8d118317c363e807da5"

### Table D.10: File attributes in the StaticProperties list

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Filename</td>
<td>Name=&quot;capinfos.exe&quot;</td>
</tr>
<tr>
<td>Id</td>
<td>SHA-1 of the PE, preceded by '0000'</td>
<td>Id=&quot;00005c5ebf7d4e969ff50b186109b2c18b47f257365&quot;</td>
</tr>
<tr>
<td>ProductName</td>
<td>The &quot;Product name&quot; field from the file metadata</td>
<td>ProductName=&quot;Capinfos&quot;</td>
</tr>
<tr>
<td>CompanyName</td>
<td></td>
<td>CompanyName=&quot;The Wireshark developer community&quot;</td>
</tr>
<tr>
<td>ProductVersion</td>
<td>The &quot;Product version&quot; field from the file metadata</td>
<td>ProductVersion=&quot;2.6.5&quot;</td>
</tr>
<tr>
<td>VerLanguage</td>
<td>Microsoft Language Id, in decimal (1033 for en-us)</td>
<td>VerLanguage=&quot;1033&quot;</td>
</tr>
<tr>
<td>ShortName</td>
<td>Short name as found in the MFT</td>
<td>ShortName=&quot;API-MS-1.DLL&quot;</td>
</tr>
<tr>
<td>SwitchBackContext</td>
<td>Unknown</td>
<td>SwitchBackContext=&quot;0x0100000000000600&quot;</td>
</tr>
<tr>
<td>FileVersion</td>
<td></td>
<td>FileVersion=&quot;2.6.5&quot;</td>
</tr>
<tr>
<td>Size</td>
<td>Size of the PE in bytes</td>
<td>Size=&quot;0x532a8&quot;</td>
</tr>
<tr>
<td>SizeOfImage</td>
<td>The SizeOfImage field from the optional header of the PE</td>
<td>SizeOfImage=&quot;0x53000&quot;</td>
</tr>
</tbody>
</table>

21/01/2019
Analysis of the AmCache

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>PeHeaderHash</td>
<td>The Checksum field from the optional header of the PE</td>
<td>PeHeaderHash=&quot;01012864b33151873a9ca2d4c05e28d87cfb023f0f3&quot;</td>
</tr>
<tr>
<td>PeChecksum</td>
<td>The Checksum field from the optional header of the PE</td>
<td>PeChecksum=&quot;0x5fe24&quot;</td>
</tr>
<tr>
<td>BinProductVersion</td>
<td>The &quot;Description&quot; field from the file metadata</td>
<td>BinProductVersion=&quot;2.6.5.0&quot;</td>
</tr>
<tr>
<td>BinFileVersion</td>
<td>The &quot;Description&quot; field from the file metadata</td>
<td>BinFileVersion=&quot;2.6.5.0&quot;</td>
</tr>
<tr>
<td>FileDescription</td>
<td>The &quot;Description&quot; field from the file metadata</td>
<td>FileDescription=&quot;Capinfos&quot;</td>
</tr>
<tr>
<td>LinkerVersion</td>
<td>The &quot;MajorLinkerVersion&quot; and &quot;MinorLinkerVersion&quot; fields combine from the optional header of the PE</td>
<td>LinkerVersion=&quot;14.12&quot;</td>
</tr>
<tr>
<td>LinkDate</td>
<td>Compile date in UTC</td>
<td>LinkDate=&quot;11/28/2018 18:23:59&quot;</td>
</tr>
<tr>
<td>BinaryType</td>
<td>32BIT or 64BIT</td>
<td>BinaryType=&quot;32BIT&quot;</td>
</tr>
<tr>
<td>Created</td>
<td>Creation date in UTC</td>
<td>Created=&quot;11/28/2018 18:31:44&quot;</td>
</tr>
<tr>
<td>Modified</td>
<td>Modification date in UTC</td>
<td>Modified=&quot;11/28/2018 18:31:44&quot;</td>
</tr>
<tr>
<td>LongPathHash</td>
<td>SHA-1 of the full path in lowercase, encoded in UTF-16LE</td>
<td>LongPathHash=&quot;00000584d7d0b21899a27a38e102ef668e3b18ed3&quot;</td>
</tr>
<tr>
<td>UniqueId</td>
<td>Records where the file is located in the installation directory</td>
<td>UniqueId=&quot;0x24&quot;</td>
</tr>
</tbody>
</table>

**Table D.11.: IEAddOn attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td>Name=&quot;XSL Template 3.0&quot;</td>
</tr>
<tr>
<td>Type</td>
<td>Values seen : &quot;ActiveX&quot;, &quot;BrowserHelperObject&quot;, &quot;BrowserExtension&quot;</td>
<td>Type=&quot;ActiveX&quot;</td>
</tr>
</tbody>
</table>
Analysis of the AmCache

Publisher

Publisher="Microsoft Corporation"

CLSID

CLSID="{5078f36–c551–11d3–89b9–0000f81e221}"  

UniqueId

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 D.10. The only additional attributes is described in Table D.12:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>OsComponent</td>
<td></td>
<td>OsComponent=&quot;true&quot;</td>
</tr>
</tbody>
</table>

### E. AEINV_AMI_WER structure

```xml
<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 -->
      </Program>
    </Installed>
    <Indicators>
      <!-- List of Run keys associated with the program -->
      <Registry />
        <!-- Information about a Run Key. Attributes described previously in Table D.4 -->
      </Registry>
    </Indicators>
    <AddRemoveProgramIndicators>
      <!-- List of Uninstall keys associated with the program -->
      <AddRemoveProgram />
        <!-- Information about an Uninstall Key. Attributes described previously in Table D.5 -->
      </AddRemoveProgram>
    </AddRemoveProgramIndicators>
    <ShellIndicators>
      <!-- List of exe files listed in the Start Menu -->
      <Shell />
        <!-- Information about PE. Attributes described previously in Table D.6 -->
      </Shell>
    </ShellIndicators>
    <MsiIndicators>
      <!-- Information about the MSI file used to install the program -->
      <Msi />
        <!-- Information about MSI. Attributes described previously in Table D.7 -->
      </Msi>
    </MsiIndicators>
    <FileExtIndicators>
      <!-- File extensions that are opened by the program -->
      <FileExtensionHandler />
        <!-- Extension. Attributes described previously in Table D.8 -->
      </FileExtensionHandler></FileExtIndicators>
    <DirectoryIndicators>
      <!-- Installation folder and sub–folder containing PE file -->
      <Directory />
        <!-- Folder. Attributes are described in Table D.9 -->
      </Directory>
    </DirectoryIndicators>
  </Indicators>
</ProgramList>
<StaticProperties>
  <!-- List of PEs under the installation folder and sub–folder. Only contains one attribute: Id -->
  <Files>
    <File />
      <!-- PE. Attributes described previously in Table E.3 -->
    </File>
  </Files>
</StaticProperties>
```
The attributes for the Report element are the same as those found in AEINV_WER and described in Table D.1.
only additional attribute is described in Table E.1.

### Table E.1.: Report attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClientVersion</td>
<td>Unknown, seems similar to the Version</td>
<td>ClientVersion=&quot;1.12.0&quot;</td>
</tr>
</tbody>
</table>

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

### Table E.2.: System attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualMachine</td>
<td>Unknown: the tests on virtual machine all had false for this value.</td>
<td>VirtualMachine=&quot;false&quot;</td>
</tr>
<tr>
<td>PortableWorkspace</td>
<td>Unknown</td>
<td>PortableWorkspace=&quot;false&quot;</td>
</tr>
</tbody>
</table>

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

### Table E.3.: File attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CrcChecksum</td>
<td>Unknown</td>
<td>CrcChecksum=&quot;0xcb05168e&quot;</td>
</tr>
<tr>
<td>PeImageType</td>
<td>Unknown</td>
<td>PeImageType=&quot;0x8664&quot;</td>
</tr>
<tr>
<td>PeSubsystem</td>
<td>Unknown</td>
<td>PeSubsystem=&quot;2&quot;</td>
</tr>
</tbody>
</table>

### Table E.4.: Installer attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompletionState</td>
<td>1 if the install was successful, 0 if not</td>
<td>CompletionState=&quot;1&quot;</td>
</tr>
<tr>
<td>CreatedArpEntries</td>
<td>Always 1</td>
<td>CreatedArpEntries=&quot;1&quot;</td>
</tr>
</tbody>
</table>

21/01/2019
Analysis of the AmCache

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartTime</td>
<td>Timestamp, in UTC, the install-</td>
<td>StartTime=&quot;08/21/2018 12:57:00&quot;</td>
</tr>
<tr>
<td>StopTime</td>
<td>Timestamp, in UTC, the install-</td>
<td>StopTime=&quot;08/21/2018 12:58:47&quot;</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>OsComponent</td>
<td>Unknown</td>
<td>OsComponent=&quot;false&quot;</td>
</tr>
<tr>
<td>SigPublisherName</td>
<td>The signer of the PE certificate</td>
<td>SigPublisherName=&quot;Wireshark Foundation, Inc&quot;</td>
</tr>
</tbody>
</table>

Table E.6.: DiscInfo attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the Disc</td>
<td>Name=&quot;VBOXADDITIONS.5.&quot;</td>
</tr>
<tr>
<td>Id</td>
<td>Unknown</td>
<td>Id=&quot;0004021d62bcd80dc4a5ac67bfdb91516395084b5&quot;</td>
</tr>
<tr>
<td>SetupScriptChecksum</td>
<td>Unknown</td>
<td>SetupScriptChecksum=&quot;17231136449290210510&quot;</td>
</tr>
<tr>
<td>Size</td>
<td>Unknown</td>
<td>Size=&quot;58466304&quot;</td>
</tr>
</tbody>
</table>

Table E.7.: Driver attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DriverId</td>
<td>SHA-1 preceded by '0000' of the driver</td>
<td>DriverId=&quot;000002da97a490b126c7710d13b431a6e74123f3cc0&quot;</td>
</tr>
<tr>
<td>Name</td>
<td>Filename</td>
<td>Name=&quot;1394ohci.sys&quot;</td>
</tr>
</tbody>
</table>
### F. PropCache.bin structure

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

![PropCache.bin byte structure](image)

**Fig. F.1.:** PropCache.bin byte structure

The structure of a `PATH` field is described in Fig. F.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. \( \text{str}\_\text{len} \) is the number of bytes in \text{folder}\_\text{name} (which is in UTF-16 LE), including the final "\00".

<table>
<thead>
<tr>
<th>Size</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>( \text{str}_\text{len} ) folder_\text{name} ...</td>
</tr>
</tbody>
</table>

DRIVER 1
...

DRIVER 2
...

...

DRIVER J
...

Fig. F.2.: PATH byte structure

The structure of a DRIVER field is described in Fig. F.3. The Size field is the size of the DRIVER field, in bytes. \( K \) is the number of INFO field.

<table>
<thead>
<tr>
<th>Size</th>
<th>( K )</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO 1</td>
<td>...</td>
</tr>
</tbody>
</table>

INFO 2
...

...

INFO \( K \)
...

Fig. F.3.: DRIVER byte structure

The structure of a INFO field is described in Fig. F.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.

| Unclear | info\_type | size\_info | information ... |

Fig. F.4.: INFO byte structure

The different value for info\_type are as follows:
- 0x0 = Filename in lowercase;
- 0x1 = SHA-1, preceded by '0000';
- 0x2 = Unknown;
- 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 = Unknown;
• 0xe = Size of the PE header;
• 0xf = Unknown;
• 0x10 = Unknown;
• 0x11 = Date of Modification, in FILETIME format.

G. 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>
</System>
</HardwareItem>
</CompatibilityInfo>
</HardwareItem>
[…]
</Hardware>
</Plugins> <!-- Plugin for the system (SecureBoot for example) -->
</Plugin>
</CompatibilityInfo>
</Plugin>
[…]
</Plugins>
</Devices> <!-- List of physical devices that were plugged on the system. -->
</DeviceInventoryPerfData>
</Device>
</HardwareIds>
</HwId>
[…]
</HardwareIds>
</CompatibleIds>
</CompatibilityIds>
</DeviceInfo>
Analysis of the AmCache

<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>
</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>
</AdvertisingID>
</TelemetryData>
</AdvertisingID>
</ChromeOSLaunchMode>
</TelemetryData>
</ChromeOSLaunchMode>
</DVDTelemetrySessionStartDate>
</TelemetryData>
</DVDTelemetrySessionStartDate>
</ OTHER-CORDOM-DVDTelemetrySessionCount>
</TelemetryData>
</OTHER-CORDOM-DVDTelemetrySessionCount>
</OTHER-CORDOM-DVDTelemetrySessionData>
</TelemetryData>
</OTHER-CORDOM-DVDTelemetrySessionData>
</OTHER-DISK-DVDTelemetrySessionCount>
</TelemetryData>
</OTHER-DISK-DVDTelemetrySessionCount>
</TelemetryData>
</OTHER-DISK-DVDTelemetrySessionData>
</TelemetryData>
</OTHER-DISK-DVDTelemetrySessionData>
Analysis of the AmCache
Analysis of the AmCache

<PrimaryMonitorQuery>
<PrimaryMonitorData>
</PrimaryMonitorData>
</PrimaryMonitorQuery>
<VolumeLicenseQuery>
</VolumeLicenseQuery>
<ProcessorInformationQuery>
<ProcessorInformationData>
</ProcessorInformationData>
</ProcessorInformationQuery>
<PCSystemTypeQuery>
<PCSystemTypeData>
</PCSystemTypeData>
</PCSystemTypeQuery>
<CurrentPowerPolicyQuery>
<CurrentPowerPolicyData>
</CurrentPowerPolicyData>
</CurrentPowerPolicyQuery>
</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>
[...]
</RedirectedProfiles>
</ChromeApps>
</ChromeApps>
<StartupApplications> <!-- List of startup applications -->
<Application>
</Application>
[...]
</StartupApplications>
</FirmwareTypeData>
</FirmwareTypeData>
</WinSAT>
</Metrics>
<CPUMetrics>
<CompressionMetric>
</CompressionMetric>
<EncryptionMetric>
</EncryptionMetric>
</CPUCompression2Metric>
</CPUCompression2Metric>
<Encryption2Metric>
</Encryption2Metric>
</Encryption2Metric>
</CompressionMetricUP>
</CompressionMetricUP>
</EncryptionMetricUP>
</EncryptionMetricUP>
</CPUCompression2MetricUP>
</CPUCompression2MetricUP>
Analysis of the AmCache

</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>
</WinSAT>
</WindowsLicensing>
</WindowsLicensing>
</SleepStatesSupported>
</SleepStatesSupported>
</TelemetryData>
</TelemetryData>
</ChromeRlz>
</Rlz>
</ChromeRlz>
</MicrophoneInfo>
</MicrophoneInfo>
</CBSErrorInfo>
</CBSErrorInfo>
</PreviousUpgradesInfo>
</PreviousUpgradesInfo>
</CompatiblityImpactData> <!-- 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>
[...]

21/01/2019 Page 51 of 58
Analysis of the AmCache

```xml
<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>
```

G.1: Generic structure of FullCompatReport.xml

**Table G.1.: CompatReport attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>MID</td>
<td>Unknown</td>
<td>MID=&quot;A1990A22-112B-4D0F-BB3B-625E66C092E7&quot;</td>
</tr>
<tr>
<td>ReportScenario</td>
<td>Unknown</td>
<td>ReportScenario=&quot;PDU_WICA&quot;</td>
</tr>
<tr>
<td>CensusId</td>
<td>Unknown</td>
<td>CensusId=&quot;{BB91F828-92A8-4EBF-9EF3-6397DF630EF}&quot;</td>
</tr>
<tr>
<td>Version</td>
<td>Unknown</td>
<td>Version=&quot;1.6&quot;</td>
</tr>
<tr>
<td>UpgradeEligible</td>
<td>Unknown</td>
<td>UpgradeEligible=&quot;1&quot;</td>
</tr>
<tr>
<td>OfflineScan</td>
<td>Unknown</td>
<td>OfflineScan=&quot;1&quot;</td>
</tr>
<tr>
<td>IeVersion</td>
<td>Version of installed Internet Explorer</td>
<td>IeVersion=&quot;9.11.9600.17416&quot;</td>
</tr>
<tr>
<td>Sqmid</td>
<td>Unknown</td>
<td>Sqmid=&quot;{C70723D9-91ED-4AA4-9EF6-E0FB9035C335}&quot;</td>
</tr>
<tr>
<td>Racid</td>
<td>Unknown</td>
<td>RacId=&quot;{819CA618-3513-405F-99BD-880A6F707895}&quot;</td>
</tr>
<tr>
<td>WuId</td>
<td>Unknown</td>
<td>WuId=&quot;dd0fd1d3-12B8-4994-bb3e-e17c8b03613d&quot;</td>
</tr>
<tr>
<td>GeoId</td>
<td></td>
<td>GeoId=&quot;84&quot;</td>
</tr>
</tbody>
</table>
### Table G.2.: System attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>X64Capable</td>
<td>X64Capable</td>
<td>X64Capable=&quot;True&quot;</td>
</tr>
<tr>
<td>X64Running</td>
<td>X64Running</td>
<td>X64Running=&quot;True&quot;</td>
</tr>
<tr>
<td>KnownWorkingCount</td>
<td>Unknown</td>
<td>KnownWorkingCount=&quot;54&quot;</td>
</tr>
<tr>
<td>WontWorkIssueCount</td>
<td>Unknown</td>
<td>WontWorkIssueCount=&quot;0&quot;</td>
</tr>
<tr>
<td>RequireActionIssueCount</td>
<td>Unknown</td>
<td>RequireActionIssueCount=&quot;0&quot;</td>
</tr>
<tr>
<td>ComplianceIssuesCount</td>
<td>Unknown</td>
<td>ComplianceIssuesCount=&quot;0&quot;</td>
</tr>
<tr>
<td>BlockUpgradeIssueCount</td>
<td>Unknown</td>
<td>BlockUpgradeIssueCount=&quot;0&quot;</td>
</tr>
<tr>
<td>BlockUpgradeCanReinstallCount</td>
<td>Unknown</td>
<td>BlockUpgradeCanReinstallCount=&quot;0&quot;</td>
</tr>
<tr>
<td>BlockUpgradeUntilUpdateCount</td>
<td>Unknown</td>
<td>BlockUpgradeUntilUpdateCount=&quot;0&quot;</td>
</tr>
<tr>
<td>DismissibleIssueCount</td>
<td>Unknown</td>
<td>DismissibleIssueCount=&quot;0&quot;</td>
</tr>
<tr>
<td>HardBlockedDevicesCount</td>
<td>Unknown</td>
<td>HardBlockedDevicesCount=&quot;0&quot;</td>
</tr>
<tr>
<td>TotalIssueCount</td>
<td>Unknown</td>
<td>TotalIssueCount=&quot;0&quot;</td>
</tr>
<tr>
<td>TotalAppCount</td>
<td>Number of installed programs</td>
<td>TotalAppCount=&quot;4&quot;</td>
</tr>
<tr>
<td>TotalDeviceCount</td>
<td></td>
<td>TotalDeviceCount=&quot;57&quot;</td>
</tr>
<tr>
<td>SocBlock</td>
<td>Unknown</td>
<td>SocBlock=&quot;false&quot;</td>
</tr>
</tbody>
</table>
## Analysis of the AmCache

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSArch</td>
<td>User settings for dates, times,...</td>
<td>OSArch=&quot;x64&quot;</td>
</tr>
<tr>
<td>UserLocale</td>
<td>based on the language pack identifier(^1)</td>
<td>UserLocale=&quot;1036&quot;</td>
</tr>
<tr>
<td>TargetBuild</td>
<td></td>
<td>TargetBuild=&quot;9600&quot;</td>
</tr>
<tr>
<td>Edition</td>
<td></td>
<td>Edition=&quot;Microsoft Windows 8.1 Professionnel&quot;</td>
</tr>
<tr>
<td>Manufacturer</td>
<td></td>
<td>Manufacturer=&quot;innotek GmbH&quot;</td>
</tr>
<tr>
<td>Model</td>
<td></td>
<td>Model=&quot;VirtualBox&quot;</td>
</tr>
</tbody>
</table>

### Table G.3.: Version attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major</td>
<td></td>
<td>Major=&quot;6&quot;</td>
</tr>
<tr>
<td>Minor</td>
<td></td>
<td>Minor=&quot;3&quot;</td>
</tr>
<tr>
<td>ServicePackMajor</td>
<td></td>
<td>ServicePackMajor=&quot;0&quot;</td>
</tr>
<tr>
<td>ServicePackMinor</td>
<td></td>
<td>ServicePackMinor=&quot;0&quot;</td>
</tr>
<tr>
<td>Build</td>
<td></td>
<td>Build=&quot;9600&quot;</td>
</tr>
</tbody>
</table>

### Table G.4.: Program attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td>Name=&quot;Wireshark 2.6.5 64-bit&quot;</td>
</tr>
</tbody>
</table>

\(^1\)https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-8.1-and-8/hh825678(v=win.10)\#language-packs
Analysis of the AmCache

<table>
<thead>
<tr>
<th>Version</th>
<th>Version=&quot;2.6.5&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publisher</td>
<td>Publisher=&quot;The Wireshark developer community, <a href="https://www.wireshark.org">https://www.wireshark.org</a>&quot;</td>
</tr>
<tr>
<td>Type</td>
<td>Type=&quot;Application&quot;</td>
</tr>
<tr>
<td>Source</td>
<td>Source=&quot;AddRemoveProgram&quot;</td>
</tr>
<tr>
<td>Id</td>
<td>Id=&quot;0000c16b47f8ca21d3ca3f3ace1abb7c51e40000fff&quot;</td>
</tr>
</tbody>
</table>

Table G.5.: Driver attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DriverName</td>
<td></td>
<td>DriverName=&quot;1394ohci.sys&quot;</td>
</tr>
<tr>
<td>DriverCompany</td>
<td></td>
<td>DriverCompany=&quot;Microsoft Corporation&quot;</td>
</tr>
<tr>
<td>DriverId</td>
<td>SHA-1 of the driver, preceded by '0000'</td>
<td>DriverId=&quot;0000f000843ae742b251f0f3b2dd36296d4803d1609b&quot;</td>
</tr>
<tr>
<td>DriverCheckSum</td>
<td></td>
<td>DriverCheckSum=&quot;294388&quot;</td>
</tr>
<tr>
<td>DriverTimeStamp</td>
<td>Compilation date, in UNIX format</td>
<td>DriverTimeStamp=&quot;1377173494&quot;</td>
</tr>
<tr>
<td>DriverType</td>
<td>Unknown</td>
<td>DriverType=&quot;8650778&quot;</td>
</tr>
<tr>
<td>DriverVersion</td>
<td></td>
<td>DriverVersion=&quot;6.3.9600.16384&quot;</td>
</tr>
</tbody>
</table>

Table G.6.: ServicesData attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td>Name=&quot;AeLookupSvc&quot;</td>
</tr>
</tbody>
</table>
## Analysis of the AmCache

<table>
<thead>
<tr>
<th>State</th>
<th>Running or Stopped</th>
<th>State=&quot;Running&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartMode</td>
<td></td>
<td>StartMode=&quot;Manual&quot;</td>
</tr>
<tr>
<td>PathName</td>
<td></td>
<td>PathName=&quot;svchost.exe -k netsvcs&quot;</td>
</tr>
<tr>
<td>DisplayName</td>
<td>Display name in the User-Locale language</td>
<td>DisplayName=&quot;Expérience 'dapplication&quot;</td>
</tr>
</tbody>
</table>

### Table G.7: FileImpact attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>File name in upper case</td>
<td>Name=&quot;WIRESHARK.EXE&quot;</td>
</tr>
<tr>
<td>Id</td>
<td>SHA-1, preceded by '0000'</td>
<td>Id=&quot;00003c742e7d9ff40c291d5c1d2a9aa6c93d023a34&quot;</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>Date of compilation in UNIX format and in hexadecimal</td>
<td>TimeStamp=&quot;5bfee034&quot;</td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
<td>Checksum=&quot;75a2e7&quot;</td>
</tr>
<tr>
<td>Type</td>
<td>Unknown</td>
<td>Type=&quot;0&quot;</td>
</tr>
<tr>
<td>ImpactData1</td>
<td>Unknown</td>
<td>ImpactData1=&quot;AAAAAAQaaaaaaaaaaaaaaaEAAAAAAAAAAAA&quot;</td>
</tr>
<tr>
<td>ImpactData2</td>
<td>Unknown</td>
<td>ImpactData2=&quot;AQAAAG5JAABAAAa00 EAAAAAAAAA0QAAAAAAAAAPBAAAAAAAAAAAAEAEADGQgAA&quot;</td>
</tr>
<tr>
<td>ImpactData3</td>
<td>Unknown</td>
<td>ImpactData3=&quot;AAAAAAAAAAAAAAAAAQAAAAAAA&quot;</td>
</tr>
</tbody>
</table>

### Table G.8: FileUse attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>File name, in uppercase</td>
<td>Name=&quot;ICAT.EXE&quot;</td>
</tr>
</tbody>
</table>
### Id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>SHA-1, preceded by '0000'</td>
<td>Id=&quot;00006b0c143e12d71685e752d8119219632281d3194b&quot;</td>
</tr>
</tbody>
</table>

#### Table G.9.: LaunchInfo attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaunchId</td>
<td>Unknown</td>
<td>LaunchId=&quot;474BABE2&quot;</td>
</tr>
<tr>
<td>LaunchCount</td>
<td></td>
<td>LaunchCount=&quot;14&quot;</td>
</tr>
<tr>
<td>FirstLaunchTime</td>
<td></td>
<td>FirstLaunchTime=&quot;01/11/2019 09:14:59&quot;</td>
</tr>
<tr>
<td>LastLaunchTime</td>
<td></td>
<td>LastLaunchTime=&quot;01/15/2019 09:58:32&quot;</td>
</tr>
</tbody>
</table>
Bibliography


