Windows 自启动项查看和分析

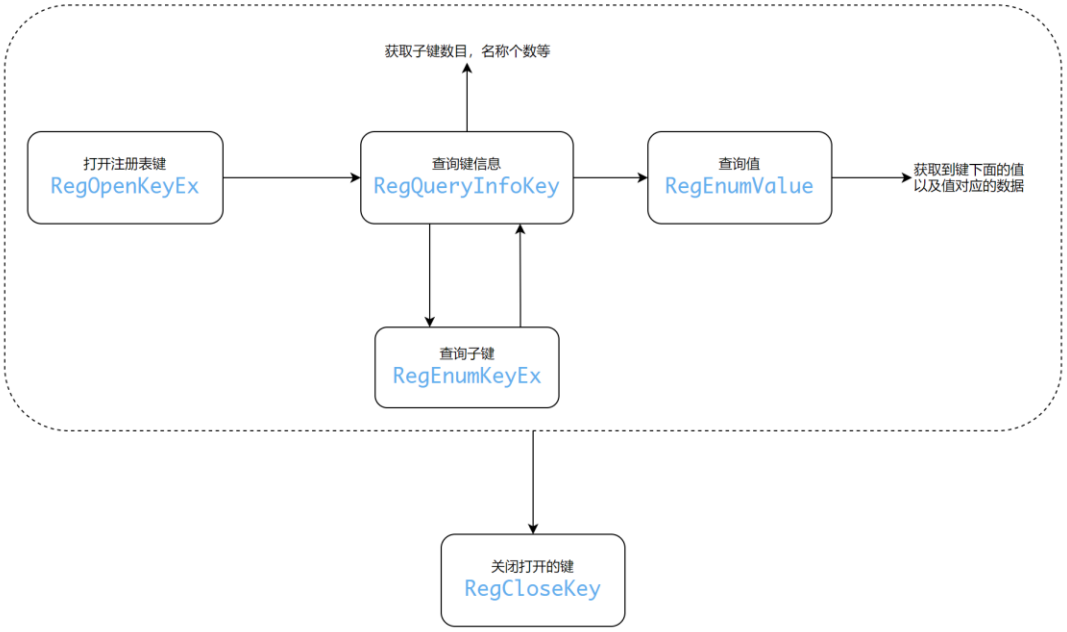
一、简介

此时实验中，我们主要需要掌握下面的内容

1.1 注册表读取

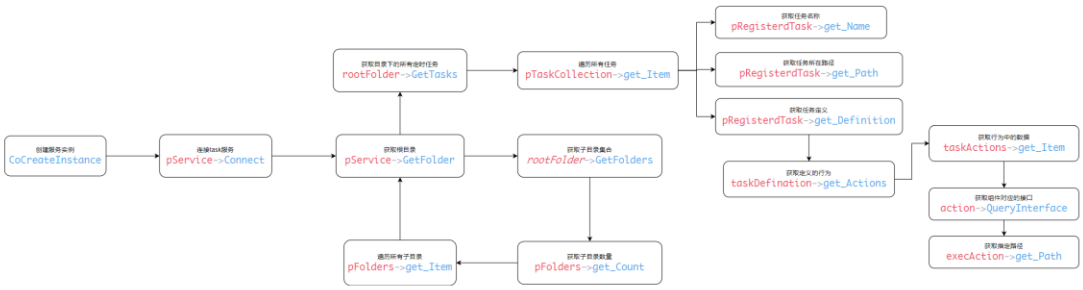
注册表的读写是本次实验的重点内容，主要的读取流程如下

1. 打开在注册表中的键
2. 查询打开键的数据信息，比如说子键的数量，键下值的个数，修改时间等等
3. 读取键下值的信息，以及对应的子键
4. 处理完成之后关闭打开的键



1.2 定时任务读取

读取定时任务的时候，我们需要使用另外一个不同的 windows API 接口，主要流程如下，过程相对而言比较复杂，这里不再赘诉，可以参考官方文档: Displaying Task Names and States (C++)



1.3 文件属性获取

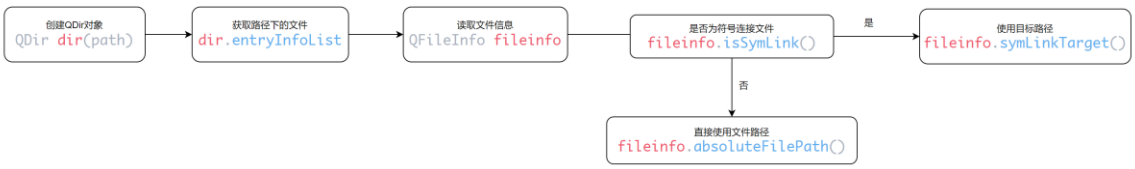
Windows 并没有提供一个直接获取文件版本属性的 API，我们需要调用多个 API 才能读取到我们需要的属性，比如说我们想要获取到文件的 `Description`，我们应该：

1. 读取文件版本信息内容的大小
2. 申请对应大小的缓冲区，将文件版本信息读取到缓冲区中
3. 调用 API 解析内容，获取需要的属性

ref: https://docs.microsoft.com/en-us/windows/win32/api/winver/

1.4 文件的遍历

在获取基于目录的自启动文件的时候，我们需要遍历一个目录下的文件。我们可以使用 Windows 提供的 API，实验时使用 Qt 进行展示，所以直接使用 Qt 提供的相比较更加简单，具体流程如下，获取到文件路径之后，我们便可以通过上述方式获取文件的属性



1.5 签名验证

Windows 中提供了相应的两个库 `wintrust.h`，`wincrypt.h`，我们使用其检查是否包含签名，签名是否可信任，具体流程文档中有详解，见 Example C Program: Verifying the Signature of a PE File

二、Logon：基于启动目录和注册表

2.1 原理

Windows 启动目录主要有两个：

- `%ProgramData%\Microsoft\Windows\Start Menu\Programs\Startup`
- `%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup`
  在启动目录的程序在系统启动的时候会自动加载并运行，我们只需要遍历其中的文件即可

在注册表中添加对应的数据，也可以实现自启动，主要的键有：

- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`
  通过读取键下面的值，我们也可以获取到自启动项

2.2 读取实现

对于启动目录，我们可以只要遍历这两个目录下的所有文件即可，可以使用 `Qt` 中提供的 API 进行实现，比如 `QDir`, `QFileinfo` 等，具体流程见[文件属性获取]

对于注册表中注册的所有自启动项，通过对注册表键的读取即可，具体流程见 [注册表读取]

2.3 隐秘性

对于在启动目录中的程序，会显式的显示在 Windows 菜单栏，容易被发觉
对于在注册表中定义的自启动项目，难以被发现

三、Services: 系统服务

3.1 原理

系统服务的自启动项位于下面的注册表键中：

- `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services`

3.2 读取实现

借助 `Autoruns` 工具，我们可以发现，对应注册表键下的子键为对应的服务名，子键中包含一些对应的值信息，比如说 `ImagePath`, `Type`，通过判断 `Type` 类型，我们便可以获取到自启动服务项，`ImagePath` 指定了启动服务的镜像文件，描述信息并

不是直接读取 ImagePath 中的文件描述，而是应该检查值 Description 中的 dll 文件的描述信息

Start	SERVICE_BOOT_START (0)	Winload预先加载该驱动程序，所以在引导过程中该驱动程序一直待在内存中。这些驱动程序恰好在SERVICE_SYSTEM_START驱动程序之前被初始化
	SERVICE_SYSTEM_START (1)	在内核初始化过程中，在SERVICE_BOOT_START驱动程序已初始化之后，该驱动程序被加载到内存中，并进行初始化
	SERVICE_AUTO_START (2)	在SCM进程（Services.exe）启动以后SCM启动该驱动程序或者服务
	SERVICE_DEMAND_START (3)	SCM根据需要启动该驱动程序或者服务
	SERVICE_DISABLED (4)	驱动程序或者服务不加载到内存中，也不初始化

在这里我们需要进行过滤，Start 小于 2 的才能视为自启动服务

3.3 隐秘性

系统服务自启动设置基于注册表，隐秘性较高，且有很多共享程序，均使用 svchost.exe，但是需要加载额外的 dll 文件，这样比较容易让恶意的 dll 文件隐藏，难以发现

四、Drivers：系统驱动程序

4.1 原理

系统驱动程序自启动项和系统服务所在的注册表位置一致，均在

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services
  虽然在同一个注册表键中，但是这两个还是有区别的，我们可以通过值中的 Type 加以区分，对于系统驱动程序而言，一般其对应的 Type 为 1,2,8

Type	SERVICE_KERNEL_DRIVER (1)	设备驱动程序
	SERVICE_FILE_SYSTEM_DRIVER (2)	内核模式的文件系统驱动程序
	SERVICE_ADAPTER(4)	已废弃
	SERVICE_RECOGNIZER_DRIVER (8)	文件系统识别器驱动程序

通过判断 Type 对应的值，我们便可以挑选出系统驱动程序

其实还有另外一个不同的地方，系统驱动程序 ImagePath 一般为 sys 文件，而系统服务一般都是可执行文件，以 exe 结尾，但是相比较而言，使用 Type 进行比较更加严谨

4.2 读取实现

与系统服务的读取一致

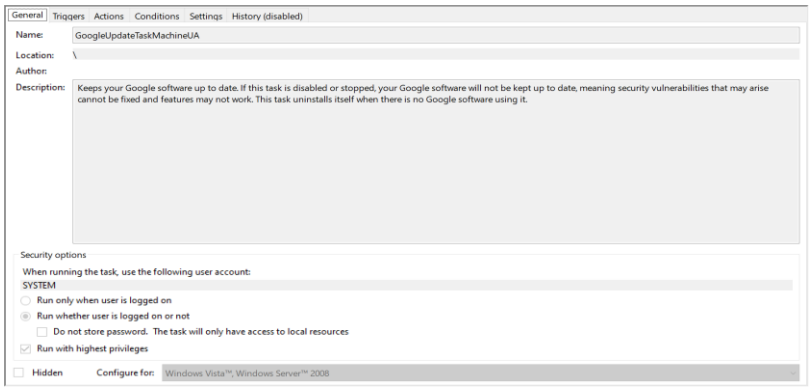
4.3 隐秘性

系统驱动程序自启动设置基于注册表，隐秘性较高，难以发觉，如果是与内核相关的系统驱动，系统级别高，对系统造成的危害更大

五、Scheduled Tasks： 计划任务

5.1 原理

计划任务是系统提供的一项常用的功能，我们可以在其中添加一个定时的任务，这样系统就可以在指定的时间运行该任务，通过 Task Scheduler 我们可以观察到下面的内容



其中 General 中保存一些基本的信息，Triggers 中指定对应的触发条件，Actions 表示触发之后需要执行的动作，比如打开一个服务，Conditions 常常用来配合 Triggers，指定在什么时候运行，如果 Conditions 中定义的条件不满足，那么不会运行任务

5.2 读取实现

具体流程见[定时任务读取]，官方提供了一个简单的示例，我们可以通过改写这个示例实现需要的功能。
因为我们需要遍历所有的任务，我这里使用的是递归实现

- 首先读取根目录，如果存在子目录，读取子目录
- 如果没有子目录，遍历读取目录中的任务
  大概代码框架如下

```
void walkFolders(ITaskFolder *rootFolder, HRESULT hr, vector<info> &inf
os)
{
  ITaskFolderCollection *pFolders = NULL;
  // 获取rootFolder 下面的目录
  hr = rootFolder->GetFolders(0, &pFolders);

  LONG numFolders = 0;
  // 获取子目录个数
  hr = pFolders->get_Count(&numFolders);

  for (LONG i = 0; i < numFolders; i++)
  {
    ITaskFolder *pRootFolder = NULL;
    // 遍历子目录
    hr = pFolders->get_Item(_variant_t(i + 1), &pRootFolder);

    // 递归遍历
    walkFolders(pRootFolder, hr, infos);

    // 遍历tasks
    IRegisteredTaskCollection *pTaskCollection = NULL;
    hr = pRootFolder->GetTasks(NULL, &pTaskCollection);
    // ....
  }
  }
```

5.3 隐秘性

任务计划程序隐秘性很好，一般用于定时任务

六、Known DLLs：知名动态链接库

6.1 原理

Known DLLs 是 Windows NT 内核的一种机制，可以用来缓存经常性使用的系统 dll 文件，如果一个可执行文件需要调用一个 dll 文件，那么首先会在这个注册表中找到对应的文件，如果找到，那么会直接使用这里面的文件

6.2 隐秘性

Known DLLs 位于注册表，隐秘性好，是保证用 LoadLibrary 装载系统 DLL 时只从特定的系统目录装载，防止装载错误的系统 DLL

6.3 读取实现

Known DLLs 保存在注册表中，读取内容方式与[Logon]等一致

七、Boot Execute：启动执行

7.1 原理

Boot Execute 所指定的内容会在系统开机的时候进行执行，其信息保存在 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager 下的 Boot Execute 值中，默认情况下为 autocheck autochk *

7.2 读取实现

首先读取到 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager 子键的信息，然后通过 RegQueryValueEx 或者遍历所有值的方式，获取到 Boot Execute 所对应的信息

7.3 隐秘性

位于注册表，隐秘性好，系统启动的时候执行

八、遇到的问题

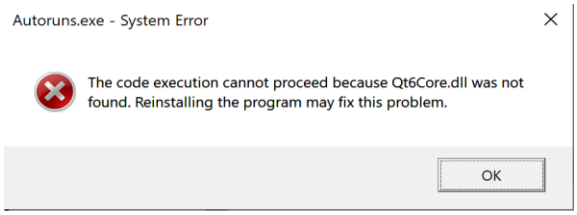
8.1 运行的时候提示库不存在

按照微软的官方文档，使用 comment 引入库文件，但是实际上运行的时候发现仍然无法引入

实际上本地使用的是 MinGW，在网上查找资料，发现应该通过.pro 添加，最后项目中的 LIBS 如下

```
LIBS += -lVersion \
        -ltaskschd \
        -lole32 \
        -luser32 \
        -loleaut32 \
        -lwintrust
```

8.2 release 的之后打开执行文件失败，提示找不到 dll 文件



默认情况下，Windows 下没有这些 dll 文件，在 Qt 提供的环境下编译的时候，会自动链接这些 dll 文件，我们发布的时候，不会将这些 dll 文件包含在可执行文件中，我们可以在 Qt 提供控制台的使用 windeployqt 命令进行部署

```
# 生成 Autoruns 需要的 dll 文件
windeployqt Autoruns.exe
```

8.3 运行程序之后有时候会 crash

运行 Qt 界面之后有时候会出现 crash 的现象，有时候是正常的，刚开始没有注意，后来到网上查了一下发现是可能访问了没有分配的地址空间。对照代码进行 review，发现有一处进行字符转换的的位置出现了问题，后来采用下面的转换方式成功解决该问题。

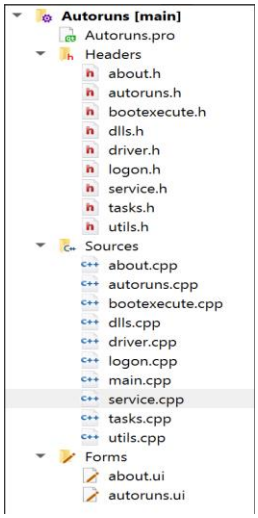
```
QString Byte2Qs(BYTE *str){
  return QString::fromWCharArray((LPCTSTR)str);
}
```

九、软件框架

实验中使用 `Qt6.1.0` 套件进行开发，使用 Qt Creator 作为主要的开发 IDE

9.1 代码结构

每一个项目使用一个类进行保存，向外提供一个 `setTableData` 函数用以设置表格的内容，内部各自实现对应的数据获取，代码结构如下

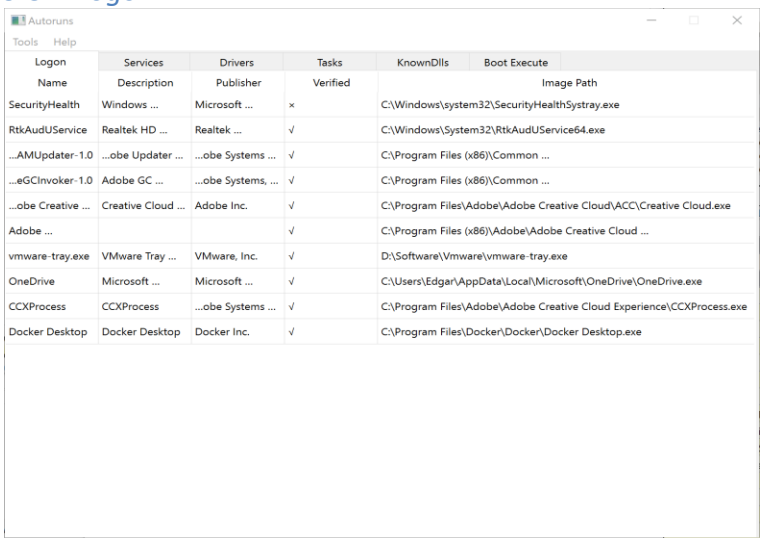


9.2 优化

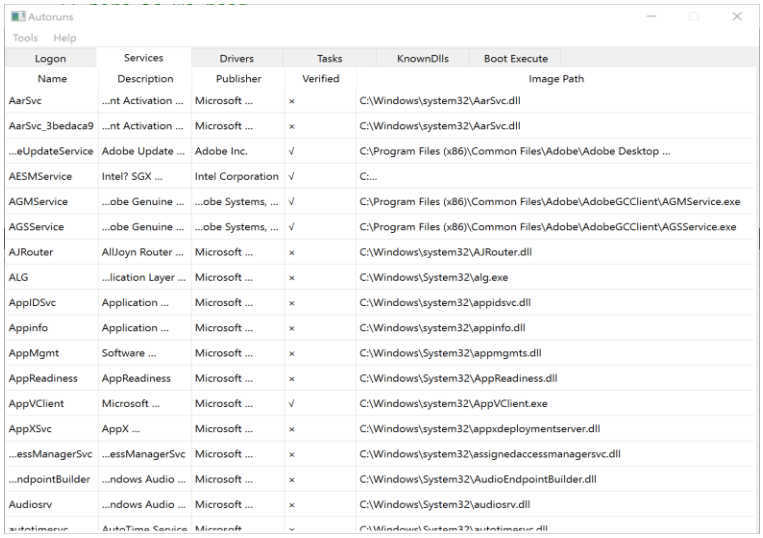
在代码中使用 `cache` 数组，标记已经设置数据的 `table`，下次访问的时候不用再去请求数据，使用上一次的数据，另外提供 `Tools > Refresh` 刷新当前标签页的数据

9.3 效果

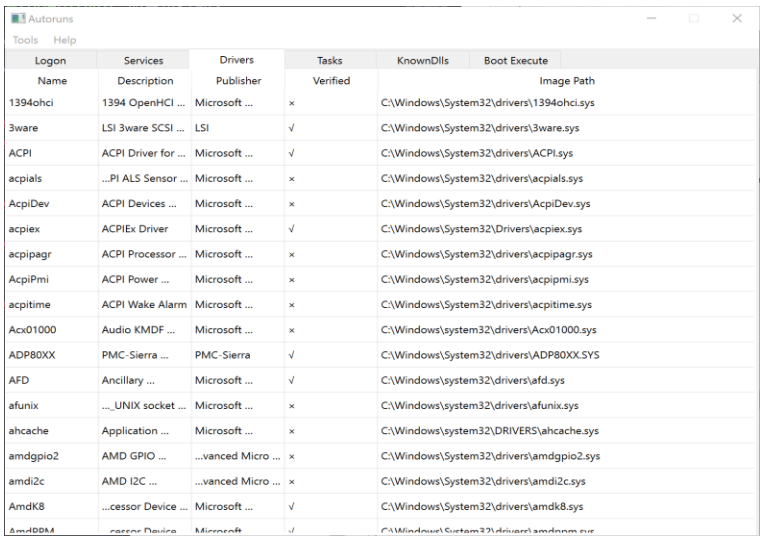
9.3.1 Logon



9.3.2 Service



9.3.3 Drivers



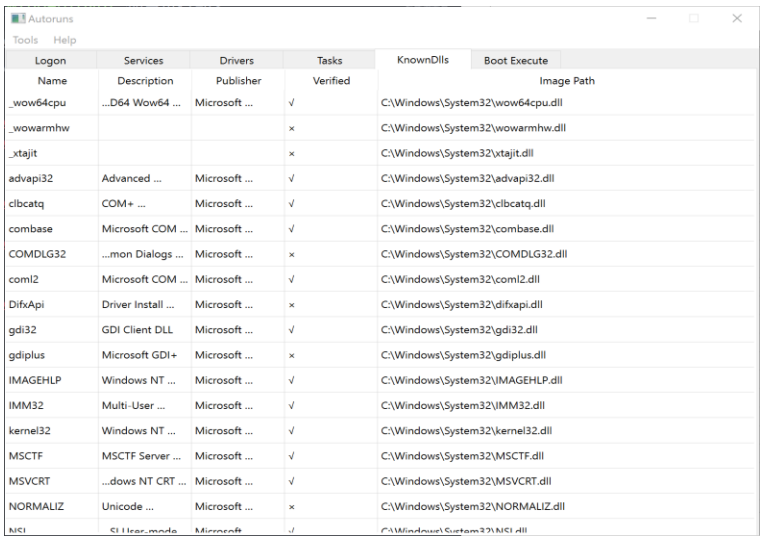
Name	Description	Publisher	Verified	Image Path
1394ohci	1394 OpenHCI ...	Microsoft ...	×	C:\Windows\System32\drivers\1394ohci.sys
3ware	LSI 3ware SCSI ...	LSI	√	C:\Windows\System32\drivers\3ware.sys
ACPI	ACPI Driver for ...	Microsoft ...	√	C:\Windows\System32\drivers\ACPI.sys
acpials	...PI ALS Sensor ...	Microsoft ...	×	C:\Windows\System32\drivers\acpials.sys
AcpiDev	ACPI Devices ...	Microsoft ...	×	C:\Windows\System32\drivers\AcpiDev.sys
acpiex	ACPIEx Driver	Microsoft ...	√	C:\Windows\System32\Drivers\acpiex.sys
acpipagr	ACPI Processor ...	Microsoft ...	×	C:\Windows\System32\drivers\acpipagr.sys
AcpiPmi	ACPI Power ...	Microsoft ...	×	C:\Windows\System32\drivers\acpipmi.sys
acpitime	ACPI Wake Alarm	Microsoft ...	×	C:\Windows\System32\drivers\acpitime.sys
Acx01000	Audio KMDF ...	Microsoft ...	×	C:\Windows\system32\drivers\Acx01000.sys
ADP80XX	PMC-Sierra ...	PMC-Sierra	√	C:\Windows\System32\drivers\ADP80XX.SYS
AFD	Ancillary ...	Microsoft ...	√	C:\Windows\system32\drivers\afd.sys
afunix	..._UNIX socket ...	Microsoft ...	×	C:\Windows\system32\drivers\afunix.sys
ahcache	Application ...	Microsoft ...	×	C:\Windows\system32\DRIVERS\ahcache.sys
amdgpio2	AMD GPIO ...	...vanced Micro ...	×	C:\Windows\System32\drivers\amdgpio2.sys
amdi2c	AMD I2C ...	...vanced Micro ...	×	C:\Windows\System32\drivers\amdi2c.sys
AmdK8	...cessor Device ...	Microsoft ...	√	C:\Windows\System32\drivers\amdk8.sys
AmdPPM	...cessor Device	Microsoft	√	C:\Windows\System32\drivers\amdppm.sys

9.3.4 Tasks

Logon Services Drivers Tasks KnowDlls Boot Execute

Name	Description	Publisher	Verified	Image Path
...ndows\Active ...			×	{CF2CF428-325B-48D3-8CA8-7633E36E5A32}
...ndows\Active ...			×	{BF5CB148-7C77-4D8A-A53E-D81C70CF743C}
...PolicyConverter	AppID Policy Converter Task	Microsoft Corporation	×	C:\Windows\system32\appidpolicyconverter.exe
...CertStoreCheck	AppID Certificate Store ...	Microsoft Corporation	×	C:\Windows\system32\appidcertstorecheck.exe
...s\Application ...	Microsoft Compatibility ...	Microsoft Corporation	√	C:\Windows\system32\compattelrunner.exe
...s\Application ...	Microsoft Compatibility ...	Microsoft Corporation	√	C:\Windows\system32\compattelrunner.exe
...s\Application ...	Windows host process ...	Microsoft Corporation	×	C:\Windows\system32\rundll32.exe
...puriverifierdaily	... Uri Handlers Registration ...	Microsoft Corporation	×	C:\Windows\system32\AppHostRegistrationVerifier.exe
...uriverifierinstall	... Uri Handlers Registration ...	Microsoft Corporation	×	C:\Windows\system32\AppHostRegistrationVerifier.exe
...TemporaryState	Windows host process ...	Microsoft Corporation	×	C:\Windows\system32\rundll32.exe
...\DsSvcCleanup	Data Sharing Service ...	Microsoft Corporation	×	C:\Windows\system32\dstokenclean.exe
...\Autochk\Proxy	Windows host process ...	Microsoft Corporation	×	C:\Windows\system32\rundll32.exe
...ker\BitLocker ...			×	{61BCD1B9-340C-40EC-9D41-D7F1C0632F05}
...ker\BitLocker ...			×	{61BCD1B9-340C-40EC-9D41-D7F1C0632F05}
...stallDeviceTask	Bluetooth Uninstall Device ...	Microsoft Corporation	×	C:\Windows\System32\BthUdTask.exe
...aintenanceTask			×	{E984D939-0E00-4DD9-AC3A-7ACA04745521}
...Client\UserTask			×	{58FB76B9-AC85-4E55-AC04-427593B1D060}
...ient\UserTask			×	{58FB76B9-AC85-4E55-AC04-427593B1D060}

9.3.5 Known dlls



9.3.6 Boot Execute