

一、 课程设计要求

- 1、结合课堂所讲授的内容，围绕着“Windows 自启动项的查看和分析”主题，查阅课外资料，编写验证代码，动手进行实验，撰写一份技术研究和设计开发报告，要求有一定深度的分析和实践步骤，主题明确，条理清晰，文字流畅，图文并茂，字数不限。此外，还需一并附上可执行软件和源代码。
- 2、可以参考 SysinternalsSuite 工具集中的 Autoruns 软件，了解在 Windows 系统中有哪些可以实现自启动的技术方法，然后分析它们各自的技术原理、实现细节和隐蔽性状况，撰写到课程报告之中。
- 3、编写自己的 Windows 自启动项查看软件。

二、 Windows 自启动项的分析

2.1 相关简介

2.1.1 Windows 自启动项

从 DOS 时代最简陋的 Autoexec.bat 启动配置文件开始，操作系统的自启动功能给我们日常使用电脑带来很多方便。然而，系统的自启动也常常被病毒、木马等恶意程序利用进行相应的恶意操作。所以需要时刻对操作系统的自启动项进行管理和监控。经过理论的学习，Windows 的自启动项包括以下几项。

2.1.2 注册表（regedit）

注册表在 Windows 系统的配置和控制方面扮演了非常关键的角色。它既是系统全局设置的存储仓库，也是每个用户的设置信息的存储仓库。它包含操作系统和其他软件的所有设置和配置相关数据的目录，也是 Windows 执行体和内核所维护的各种内存中数据结构的一个窗口。

注册表的配置数据在以下四个时间点被读取：初始的引导过程、内核引导过程、登录过程、应用程序启动过程。在以下情况下被修改，Windows 系统安装、创建默认应用程序设置、驱动程序的安装、改变应用程序或者系统设置。

注册表的值存储了不同种类的数据，共有 12 中，本次课程设计中主要用到了三类注册表值，列表如下：

数据类型	说明
REG_DWORD	存储整数或者布尔值，此次设计读取的 Type,Start 等数据就是此类型
REG_BINARY	存储超过 32 位的整数值
REG_SZ	存储字符串（Unicode）,此次设计读取的 ImagePath、Description、ObjectName 等字段就是此类型

2.2 基于 Windows 启动目录的自启动

在 Windows Vista，即内核 Windows NT 6.0 之前的所有 Windows 版本的启动目录都为：

```
C:\Documents and Settings\All Users\「开始」菜单\程序\启动
```

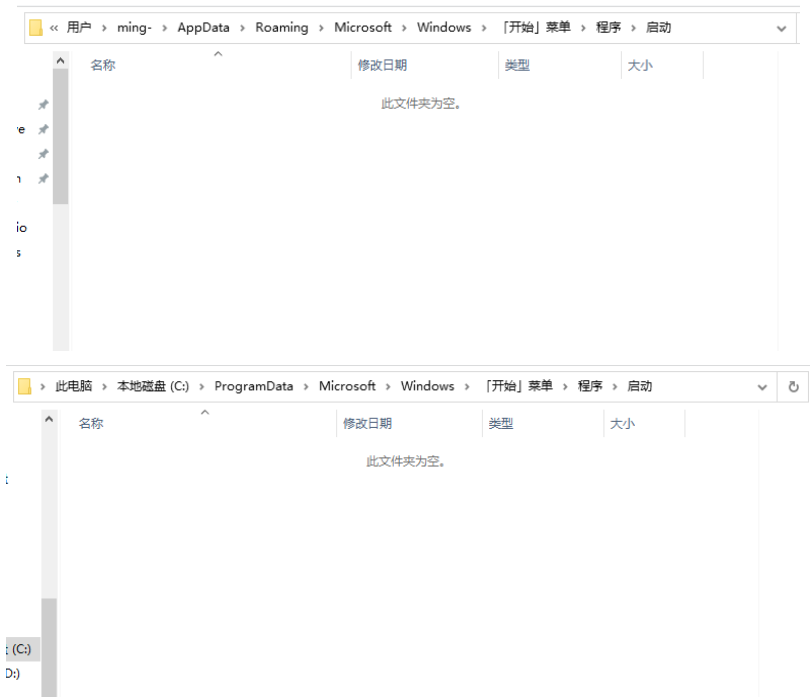
而在 Windows Vista，即内核 Windows NT 6.0 之后的所有 Windows 版本的启动目录为：

```
%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
%ProgramData%\Microsoft\Windows\Start Menu\Programs\Startup
```

本机中相应文件夹的截图如下。“开始->程序->启动”中的程序和快捷方式都会在系统

启动时自动运行。相应文件夹中的内容均为空，表明本机未有自启动项目通过 Windows 启动目录进行自启动。

这两个路径下的自启动项目在生效的时间和相应的更改权限两方面有较大区别。第一个目录是当前用户配置目录，生效的时间为相应用户登录时，其他用户登录条件下不生效。修改权限也是当前用户，其他用户登录时不生效。第二个目录是系统程序数据目录，任意用户登录都可生效，修改权限也规定为管理员及以上权限。



2.3 基于注册表（Registry）的自启动

通过此方法实现的自启动项最多，也是木马、病毒等的重灾区。自启动项的键主要位于 HKEY_CURRENT_USER (HKCU) 和 HKEY_LOCAL_MACHINE (HKLM) 两类根键下。两类自启动项目的区别仍体现在自启动的生效时间和写入的权限问题。HKLM 下键值的自启动的生效时间为计算机启动的时候，只有管理员以上权限可以写入。而 HKCU 下键值的自启动的生效时间为指定用户的登录时间内，也只有当前用户能够写入。很多键值记录了自启动项，部分列举如下：

Run：所有程序每次启动时都按照顺序自动执行
HKCU\HKLM\Software\Microsoft\Windows\CurrentVersion\Run
HKCU\HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run

RunOnce：程序仅会被自动执行一次
HKLM\HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce\

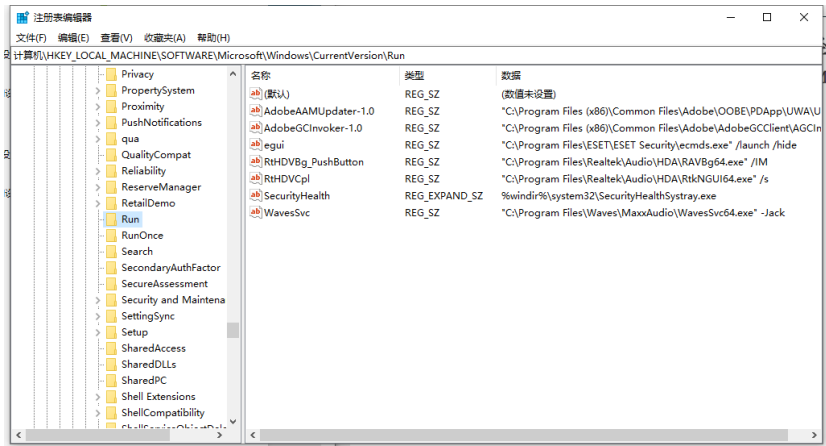
RunServicesOnce：程序会在系统加载时自动启动执行一次
HKCU\HKLM\Software\Microsoft\Windows\CurrentVersionRunServicesOnce

RunServices:RunServicesOnce后启动的程序
HKCU\HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices

RunOnceEx: Windows XP/03特有
HKCU\HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx

load
HKCU\HKLM\Software\Microsoft\WindowsNT\CurrentVersion\Windows

以 run 键为例,所包含的键值截图如下,包含的信息有相应的启动项名称和 imagepath。



此外，在 64 位系统中还存在重定向的注册表路径：

HKLM\HKCU\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run

可以通过调用相应的 API 来关闭和打开重定向，常用的函数包括 Wow64DisableWow64FsRedirection（关闭系统转向），Wow64RevertWow64FsRedirection（打开系统转向），Wow64EnableWow64FsRedirection（打开系统转向）。

2.4 基于服务程序的自启动

服务程序是通过 Windows 的服务管理机制所启动的一系列系统及网络服务，是后台运行的进程，常用来执行特定的任务，不需要和用户进行交互。比如自动更新服务、计划任务服务、后台智能传输服务等等。Windows 服务是由三个组件构成的：服务应用、服务控制程序 (SCP: service control program)，以及服务控制管理器 (SCM, service control manager)。服务程序的配置数据位于注册表如下键值，

"HKLM\System\CurrentControlSet\Services"

需要注意的是，驱动程序的配置数据也位于注册表此键值下，所以遍历所有子键的同时需要根据 type 和 start 等参数的不同来进行判断和筛选，阅读《深入解析 Windows 操作系统》第四章后对相关知识有了进一步了解，相关注册表键值列表如下。

Start:

值的名称	说明
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。
SERVICE_SYSTEM_START(1)	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

TYPE:

值的名称	说明
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服务
SERVICE_ADAPTER(4)	已废弃
SERVICE_RECOGNIZE_DRIVER(8)	文件系统识别器驱动程序
SERVICE_WIN32_OWN_PROCESS(16)	该服务运行在一个只能容纳一个服务的进程中
SERVICE_WIN32_SHARE_PROCESS(32)	该服务运行在一个可容纳多个服务的进程中
SERVICE_WIN32_INTERACTIVE_PROCESS(256)	允许该服务在控制台上显示窗口，并且接收用户的输入，但是仅限于在控制台绘画上，防止与其他用户/控制台应用程序进行交互

通过上述分析，对 Type 进行分类，认为 Type 值小于等于 8 的为驱动程序，而大于等于 16 的为服务。并获取 DisplayName, Description, ObjectName 等键值信息，分别表示服务名称、服务说明、运行账户名称。

2.4 基于驱动程序的驱动

同样地，驱动程序通常使用一组系统定义的注册表项来存储或访问驱动程序特定的信息或特定于设备的信息。此部分与 2.3 节相似，驱动程序的配置信息位于注册表如下键值：

"HKLM\System\CurrentControlSet\Services"

通过 2.3 节中的分析，对 Type 进行分类，认为 Type 值小于等于 8 的为驱动程序，而大于等于 16 的为服务。并获取 DisplayName, Description, ObjectName 等键值信息做打印及分析处理。

2.5 基于计划任务（Scheduled Tasks）的自启动

此功能类似于 Linux 系统中的 cronjob, 用户可以通过设置让系统定时运行程序和脚本，需要管理员以上的权限进行修改。此部分自启动项的分析不再通过注册表进行，二十通过分

析本地 system32 文件下的 Tasks 目录，具体路径为，文件夹下文件截图如下所示：

C:\Windows\System32\Tasks

名称	日期	大小
Microsoft	2020/4/26 10:47	文件夹
S-1-5-21-3122445638-2312831479-466466543-1001	2020/3/12 14:51	文件夹
Adobe Acrobat Update Task	2020/3/21 21:48	文件 5 KB
AdobeAAMUpdater-1.0-MicrosoftAccount-ming-guo99@outlook.com	2019/12/13 15:10	MS-DOS 应用程序 4 KB
AdobeGCIInvoker-1.0	2020/6/6 22:09	0 文件 4 KB
AliUpdater(AE069482-6D1A-4E0E-9F7D-67FC017A4180)	2020/3/11 11:36	文件 4 KB
GoogleUpdateTaskMachineCore	2020/3/21 10:18	文件 3 KB
GoogleUpdateTaskMachineUA	2020/3/21 10:18	文件 4 KB
MATLAB R2019a 启动加速器	2019/11/25 17:19	文件 4 KB
npcapwatchdog	2019/12/4 9:15	文件 4 KB
OneDrive Standalone Update Task-S-1-5-21-3122445638-2312831479-466...	2020/6/2 10:46	文件 4 KB

此目录下用 xml 的格式记录了自启动文件的 Description,version,command 等信息。以文件 GoogleUpdateTaskMachineCore 为例，其所包含的信息如下：

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.0" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Version>1.3.35.452</Version>
    <Description>请确保使用最新版的 Google 软件。如果停用或中断此任务，则您的 Google 软件就无法及时更新，这意味着无法修复潜在的安全漏洞，同时某些功能
    <URI>GoogleUpdateTaskMachineCore</URI>
  </RegistrationInfo>
  <Triggers>
    <LogonTrigger>
      <Enabled>true</Enabled>
    </LogonTrigger>
    <CalendarTrigger>
      <StartBoundary>2020-03-21T10:23:33</StartBoundary>
      <ScheduleByDay>
        <DaysInterval>1</DaysInterval>
      </ScheduleByDay>
    </CalendarTrigger>
  </Triggers>
  <Principal>
    <Principal id="Author">
      <UserId>S-1-5-18</UserId>
      <RunLevel>HighestAvailable</RunLevel>
    </Principal>
  </Principal>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StartWhenAvailable>true</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <Enabled>true</Enabled>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>C:\Program Files (x86)\Google\Update\GoogleUpdate.exe</Command>
      <Arguments>/c</Arguments>
    </Exec>
  </Actions>
</Task>
```

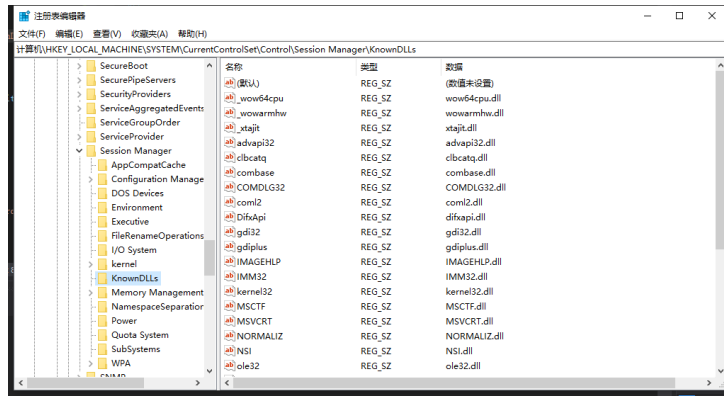
2.6 ActiveX

此类自启动较为少见，相应的注册表键值路径为，自启动就是在该键值下新建子键，用 GUID 命名，子键中新建 StubPath 的值项，内容为启动的文件名

HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components\ + GUID (全局标识符)

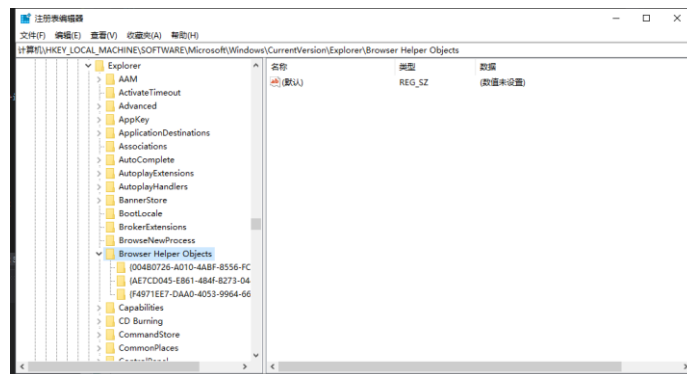
2.7 Known DDLs

Known DDLs 即为知名动态链接库，此部分自启动项的查询也是可以通过注册表信息获取，键的路径为 HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\KnownDlls，与基于注册表的自启动项类似，所有键值均为 REG_SZ 类型，遍历查询相应的值即可得到目标 dll 文件，同时这些文件均在文件夹“C:\windows\syswow64”下，所以可以得到相应 dll 文件的绝对路径，进而得到 Publisher 和 TimeStamp 的信息。



4.6 IE_BHO

BHO 是 IE 的一个插件技术，可以获取表单密码等，具有一定信息泄露的风险，此部分的自启动项分析依然通过注册表技术实现，获取键 HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects 下的所有子项，并获取相应的值，子键与 ActiveX 相同，也为全局标识符，



4.7 Image Hijacks

即为映像劫持，是微软提供给软件开发者调试使用的在注册表项，能够替换目标进程执行。但如果被病毒木马利用，便会成为触发式自启动的绝佳方式，所以修改映像劫持的操作行为也被反病毒软件列为极其危险的行为之一。

实现映像劫持修改的注册表项为：

"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Image File Execution Options"

三、设计实现

3.1 实验环境

系统	Windows10 专业版
编译器	Visual Studio 2019
开发语言	C++
界面制作	QT5.14.2

3.2 设计概述

在此课程设计中我学习了 Windows 自启动项以及注册表读写等相关知识后仿照 SysinternalsSuite 工具集中的 Autoruns 软件，借助 C++ 和 QT 进行相应的设计实现，最终完成的项目包括：

- Logon:启动项目，基于注册表启动；
- Services：系统服务；
- Drivers：系统驱动程序；
- Scheduled Tasks：计划任务；
- Internet Explorer：IE浏览器的 BHO 对象；
- Known DLLs：知名动态链接库；

3.3 注册表的读取

很多自启动项的查看需要用到注册表的读取，具体的读取注册表过程我分为两个函数去实现，分别为 `map<int, char*> read_subitem(HKEY aim_rootkey, LPCTSTR aim_key)` 和 `map<char*, LPBYTE> read_value(HKEY aim_rootkey, LPCTSTR key_data)`。

函数 `read_subitem` 实现的功能为读取某键的所有子键，输入的参数为 `HKEY` 类型的变量 `aim_rootkey` 和 `LPCWSTR` 类型的变量 `key_data`，分别为待读取的根键和键的路径，通过调用相关的注册表 API 函数打开指定的注册表、读取键值并返回此键下所有的子键名称。在此函数中用字典的数据结构存储所有的子键信息。函数 `read_value` 实现的功能为读取某键的所有键值，并可以返回指定键值，输入信息与上一个函数相同，为 `HKEY` 类型的变量 `aim_rootkey` 和 `LPCWSTR` 类型的变量 `key_data`，分别为待读取的根键和键的路径，同样地，通过调用相关的注册表 API 函数打开指定的注册表获取此键对应的键值，若无指定则返回此键的所有键值，用字典存储键值信息，`first` 项为名称，`second` 项为数据。若指定名称获取数据，则用 `LPBYTE` 类型变量存储数据并返回。具体函数如下：

```

1. map<char*, LPBYTE> read_value(HKEY aim_rootkey, LPCTSTR key_data)
2. {
3.     HKEY cpp_key;
4.     map<char*, LPBYTE> mymap;
5.
6.     if (ERROR_SUCCESS == RegOpenKeyEx(aim_rootkey, key_data, 0, KEY_READ, &cpp_key)) {
7.         DWORD dwIndex = 0, NameSize, NameCnt, NameMaxLen, Type;
8.         DWORD KeySize, KeyCnt, KeyMaxLen, DataSize, MaxDateLen;
9.         if (ERROR_SUCCESS == RegQueryInfoKey(cpp_key, NULL, NULL, 0, &KeyCnt,
            &KeyMaxLen, NULL, &NameCnt, &NameMaxLen, &MaxDateLen, NULL, NULL)) {
10.             //cout << "共有" << NameCnt << "个键值" << endl;
11.             for (DWORD dwIndex = 0; dwIndex < NameCnt; dwIndex++)
12.             {
13.                 DateSize = MaxDateLen + 1;
14.                 NameSize = NameMaxLen + 1;
15.                 char* szValueName = (char*)malloc(NameSize);
16.                 LPBYTE szValueDate = (LPBYTE)malloc(DateSize);
17.                 RegEnumValue(cpp_key, dwIndex, szValueName, &NameSize, NULL,
                    &Type, szValueDate, &DateSize);
18.                 //cout << "类型: " << Type << " 名称: "
                    << szValueName << " 数据: " << szValueDate << endl;
19.                 mymap[szValueName] = szValueDate;
20.             }
        }
    }
}

```

```

21.         }
22.
23.     }
24.     else {
25.         cout << "读取子键失败！" << endl;
26.     }
27. }
28. else {
29.     cout << "打开注册表失败！" << endl;
30. }
31.
32. return mymap;
33. }

```

```

1.  map<int, char*> read_subitem(HKEY aim_rootkey, LPCTSTR aim_key)
2.
3.  HKEY cpp_key;
4.  map<int, char*> subitem;
5.  int len = 0;
6.  //cout << "---读取子键---" << endl;
7.  if (ERROR_SUCCESS == RegOpenKeyEx(aim_rootkey, aim_key, 0, KEY_READ, &cpp_key)) {
8.      DWORD dwIndex = 0, NameSize, NameCnt, NameMaxLen, Type;
9.      DWORD KeySize, KeyCnt, KeyMaxLen, DateSize, MaxDateLen;
10.     if (ERROR_SUCCESS == RegQueryInfoKey(cpp_key, NULL, NULL, 0, &KeyCnt, &KeyMaxLen, NULL, &NameCnt, &NameMaxLen, &MaxDateLen, NULL, NULL)) {
11.         //cout << "共有" << KeyCnt << "个子键" << endl;
12.         for (DWORD dwIndex = 0; dwIndex < KeyCnt; dwIndex++)
13.         {
14.             KeySize = KeyMaxLen + 1;
15.             char* szKeyName = (char*)malloc(KeySize);
16.             RegEnumKeyEx(cpp_key, dwIndex, szKeyName, &KeySize, NULL, NULL, NULL, NULL);
17.             //cout << szKeyName << endl;
18.
19.             subitem[len] = szKeyName;
20.             len++;
21.         }
22.     }
23.     else {
24.         cout << "读取子键失败！" << endl;
25.     }
26. }

```



```

27. else {
28.     cout << "打开注册表失败！" << endl;
29. }
30. RegCloseKey(cpp_key); //关闭句柄
31. //cout << "---读取子键结束---" << endl;
32.
33. return subitem;

```

在此过程中调用了相应的 Windows API 函数如下：

(1) `LONG RegOpenKeyEx(HKEY hKey, LPCWSTR lpSubKey, DWORD ulOptions, REGSAM samDesired, PHKEY phkResult);`

此函数可用于打开指定的注册表项，hkey 为处理当前打开的键或任何预定义的根键，lpSubKey：指向一个空终止字符串的指针，该字符串包含要打开的子键的名称。如果该参数为 NULL 或指向空字符串的指针，则函数将由 hKey 参数标识的键打开一个新句柄。在这种情况下，函数将不会关闭先前打开的句柄。ulOptions 保留，设置为零。samDesired：作为输入，是一个访问掩码，设置为零。phkResult 指向接收已打开键的句柄的变量的指针。

(2) `RegQueryInfoKey`

此函数可以用于检索指定注册表项的信息，read_subitem 函数中调用此函数以获取目标键中的子键个数。Read_value 函数中调用此函数以获取目标键的所有信息。此函数的使用需要在调用 RegOpenKeyEx 函数打开目标注册表后使用。

(3) `LSTATUS RegEnumKeyExA(HKEY hKey, DWORD dwIndex, LPSTR lpName, LPDWORD lpcchName, LPDWORD lpReserved, LPSTR lpClass, LPDWORD lpcchClass, PFILETIME lpftLastWriteTime);`

此函数可用于枚举指定的已经打开的注册表项的子项，read_subitem 函数中调用此函数枚举子键的名称，read_value 函数中调用此函数用于枚举具体的数值。

(4) `LSTATUS RegCloseKey(HKEY hKey);`

此函数可用于关闭指定注册表项的句柄。

3.4 文件的读取

此部分功能主要用于 Scheduled Tasks 自启动项获取，需要获取“C:\Windows\System32\Tasks”目录下的所有文件并通过解析 xml 其信息。我一开始使用 API 函数 findfirst 和 findnex 调取文件夹的所有文件目录并通过文件属性来判断相应文件类型。但是完成的相应函数对普通目录正常运行，但是对 system32 下的文件夹无效，通过查阅资料认为是此文件夹需要管理员权限，但是提权后仅能识别 Tasks 目录下的 Microsoft 文件夹，无法识别其他文件。

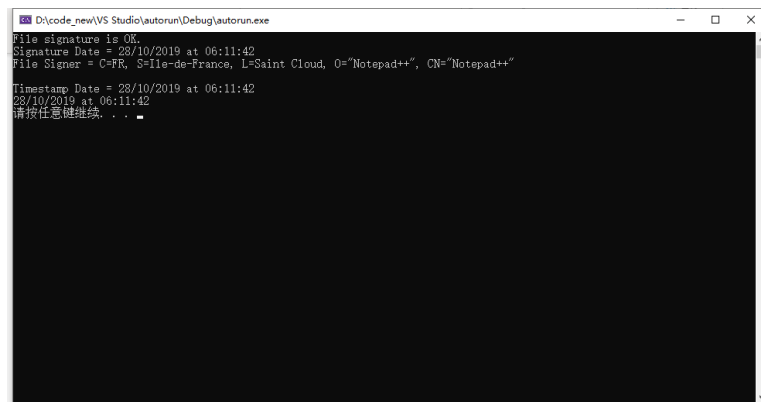
在完成获取 Publisher 和 TimeStamp 功能的时候调用了<Wintrust.lib>以提高权限来获取相应信息，所以获取 Scheduled Tasks 自启动项获取时决定转换思路，首先完成提权，以管理员以上的权限读取 Tasks 目录下的所有文件，并判断文件属性，若为文件夹则进入文件夹继续遍历直至结束返回上一文件夹，若为文件则解析信息得到 ImagePath。

3.5 证书和时间戳的获取

Publisher 和 TimeStamp 的获取通过函数 get_timestamp()和 get_publisher()实现，这两个函数的输入参数均为目标程序的路径，并用 LPCWSTR 变量传递，获取 Publisher 等信息

通过调用 CertGetNameString 等函数实现，并通过 WinVerifyTrust 来实现权限提升等。而时间戳的获取主要通过调用函数 FileTimeToSystemTime 并经过处理转换来获得。

以 NOTEPAD++.exe 为例，此函数的运行结果，包含证书信息、时间戳信息等，在实际运行中通过指定数据的获得 Publisher 和 TimeStamp 信息。



```
D:\code_new\VS Studio\autorun\Debug\autorun.exe
File signature is OK.
Signature Date = 28/10/2019 at 06:11:42
File Signer = C=FR, S=Ile-de-France, L=Saint Cloud, O=Notepad++, CN=Notepad++
Timestamp Date = 28/10/2019 at 06:11:42
28/10/2019 at 06:11:42
请按任意键继续...
```

3.6 数据类型的转换

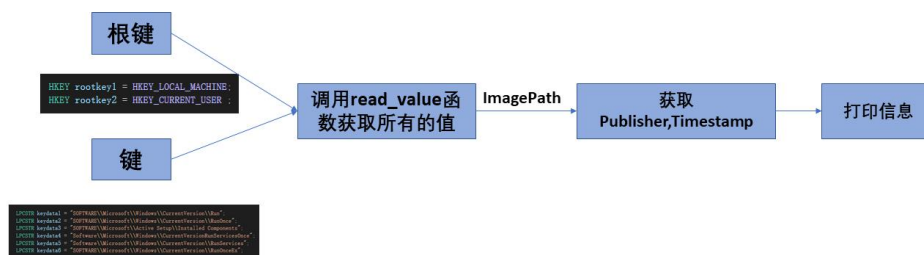
在全部的编程实现中，调用了很多并没有用过的 API 函数，这些函数对参量的类型有较高要求，用的数据类型大多是 LPBYTE, LPCSTR, LPCWSTR 等等，就需要很多的转换函数来实现与我们常用的 CHAT.TCHAR*, STRING 等数据类型的转换，包括为了实现图形界面转换为 QString 等类型，这是我全过程最头疼的点。起初通过修改项目属性，通过更改字符集等实现，但是变为 QT 项目后，编码方式被改变，需要完全重新转换。所以我在头文件中将各个转换函数列举，这些转换函数大多调用 WideCharToMultiByte 等字符转换函数实现转换功能。具体各转换函数可参见 merge.h 文件。

四、 结果展示与分析

本部分分功能进行相应原理简述、结果展示以及部分结果分析，因为原理已经在第二章的探究部分进行阐释，所以本部分着重阐述编程实现。在具体的项目实现中，我即完成了命令行输出的项目编写以及 QT 界面的编写，我首先是编写相应的功能函数，但是因为第一次用 QT 编写界面，在编写函数的过程中并没有注意数据类型的统一等，后来在实际界面编写过程中，在数据类型转换方面很是挣扎，所以界面只完成了必选功能的展示，命令行版本项目有完整的功能演示。

4.1 Logon

此部分功能运行的流程图如下所示，通过上一节所述，首先列举此部分基于注册表自启动项的具体键路径，调用 read_value 函数来获取所有的自启动项值，此时得到的就是 ImagePath,调用 get_publisher 和 get_timestamp 来获取发布者、时间戳等信息，最后在命令行界面打印信息，或者转为 QString 字符串在 QT 的 table widget 控件输出信息。



列举键值如下：

```

HKEY rootkey1 = HKEY_LOCAL_MACHINE;
HKEY rootkey2 = HKEY_CURRENT_USER ;

LPCSTR keydata1 = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run";
LPCSTR keydata2 = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\RunOnce";
LPCSTR keydata3 = "SOFTWARE\\Microsoft\\Active Setup\\Installed Components";
LPCSTR keydata4 = "Software\\Microsoft\\Windows\\CurrentVersion\\RunServicesOnce";
LPCSTR keydata5 = "Software\\Microsoft\\Windows\\CurrentVersion\\RunServices";
LPCSTR keydata6 = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\RunOnceEx";
  
```

相应的输出如下：

图形界面：

Logon	Services	Drivers	Scheduled Tasks	Known DLLs	Internet Explorer	Winlogon
Autorun Entry						
	Acrobat Assistant 8.0	AcroTray	Adobe Systems Inc.	Image Path		
	Adobe Creative Cloud	Adobe Creative Cloud	Adobe Systems Incorpo...	C:\Program Files (x86)\Ado...		
	vmware-tray.exe	VMware Tray Process	VMware, Inc.	C:\Program Files (x86)\VMw...		
	Intel Driver & Support Assist...	Intel Driver & Support ...	Intel	C:\Program Files (x86)\Inte...		
	Acrobat Assistant 8.0	AcroTray	Adobe Systems Inc.	C:\Program Files (x86)\Ado...		
	Adobe Creative Cloud	Adobe Creative Cloud	Adobe Systems Incorpo...	C:\Program Files (x86)\Ado...		
	vmware-tray.exe	VMware Tray Process	VMware, Inc.	C:\Program Files (x86)\VMw...		
	Intel Driver & Support Assist...	Intel Driver & Support ...	Intel	C:\Program Files (x86)\Inte...		
	OneDrive	Microsoft OneDrive	Microsoft Corporation	C:\Users\11920\AppData\L...		
	QQ2009	腾讯QQ	Tencent	C:\Program Files (x86)\Tenc...		
	qbclipboard	QQBrowse	Tencent	C:\Program Files (x86)\Tenc...		
	JingYeQian	敬业签	河南礼佑网络科技有限公司	C:\Program Files (x86)\jingy...		
	ctfmon	CTF 加载程序	Microsoft Corporation	C:\WINDOWS\system32\ctf...		
	BaiduYunGuanjia	BaiduNetdisk	Microsoft Corporation	C:\Users\11920\AppData\R...		
	BaiduYunDetect	BaiduNetdisk	Microsoft Corporation	C:\Users\11920\AppData\R...		
	AlternateShell	Windows 命令处理程序	Microsoft Corporation	cmd.exe		

命令行界面：

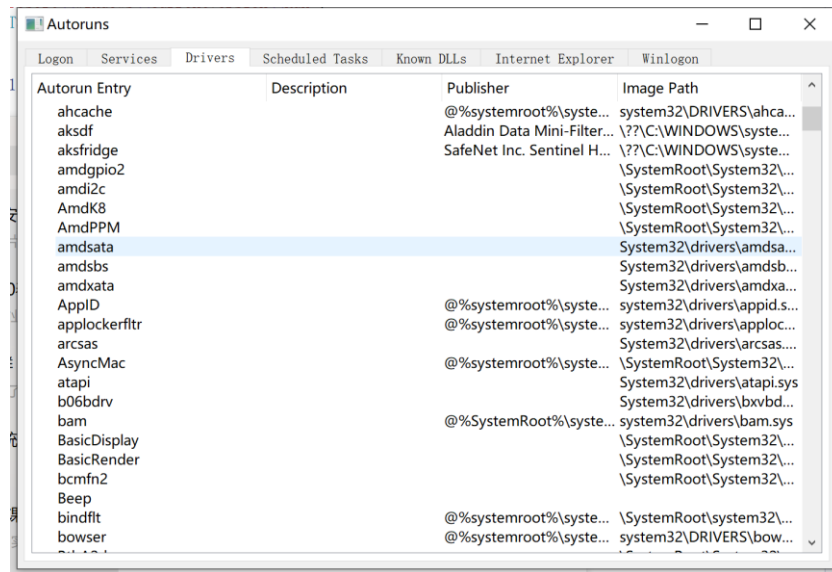
```

Please choose the mode number
1----Logon
2----Services
3----Drivers
4----Scheduled Tasks
1
Logon : Autorun entry ; Image Path ; Description ; Publisher ; TimeStamp :

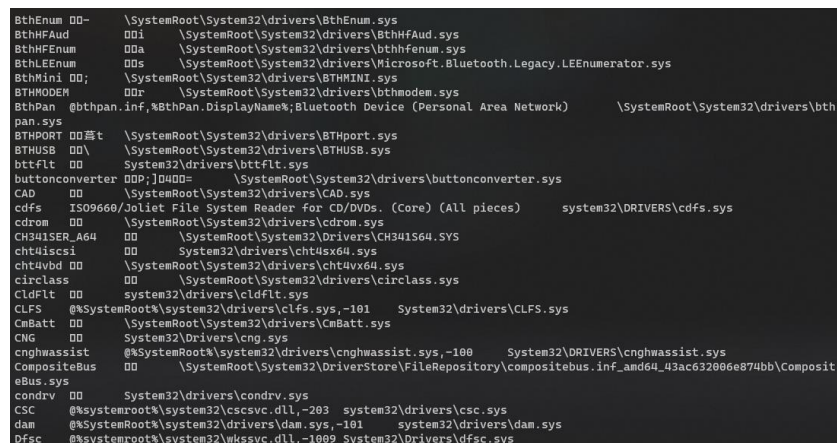
vmware-tray.exe "C:\Program Files (x86)\VMware\VMware Workstation\vmware-tray.exe"
Adobe Creative Cloud "C:\Program Files (x86)\Adobe\Adobe Creative Cloud\ACC\Creative Cloud.exe" --showwindow=false --
AcroStartup-tray "C:\Program Files (x86)\Adobe\Acrobat DC\Acrobat\Acrotray.exe"
OneDrive "C:\Users\ming\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background
JetBrains Toolbox "C:\Users\ming\AppData\Local\JetBrains\Toolbox\bin\jetbrains-toolbox.exe" --minimize
CCXProcess "C:\Program Files (x86)\Adobe\Adobe Creative Cloud\CCXProcess\CCXProcess.exe"
BaiduYunDetect "C:\Users\ming\AppData\Roaming\Baidu\BaiduNetdisk\YunDetectService.exe"
quit?
  
```

4.2 Services

此功能的实现与 Drivers 自启动项查看的实现类似，故相应操作流程进行一起叙述，首先定义根键“HKEY_LOCAL_MACHINE”，并用 LPCSTR 变量存储键的路径，首先调用 read_subitem 函数得到所有的子键名称，通过字符串的合并获得子键路径后调用 read_type 函数获取 Type 值，若大于等于 16，则判定为服务，继续进行读取 Description,ImagePath,继而得到 Publisher,TimeStamp 等信息，否则进行下一个循环，而驱动则相反，Type 值需小于

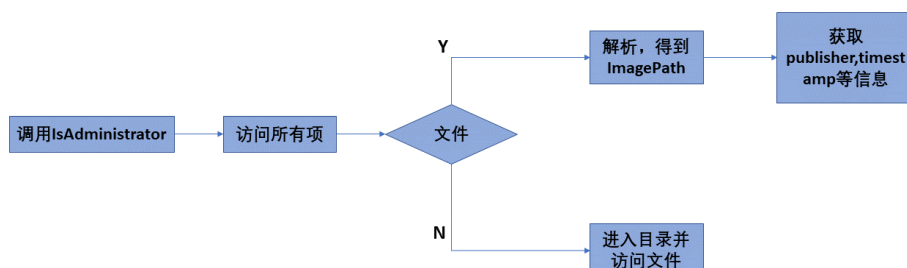


命令行界面:

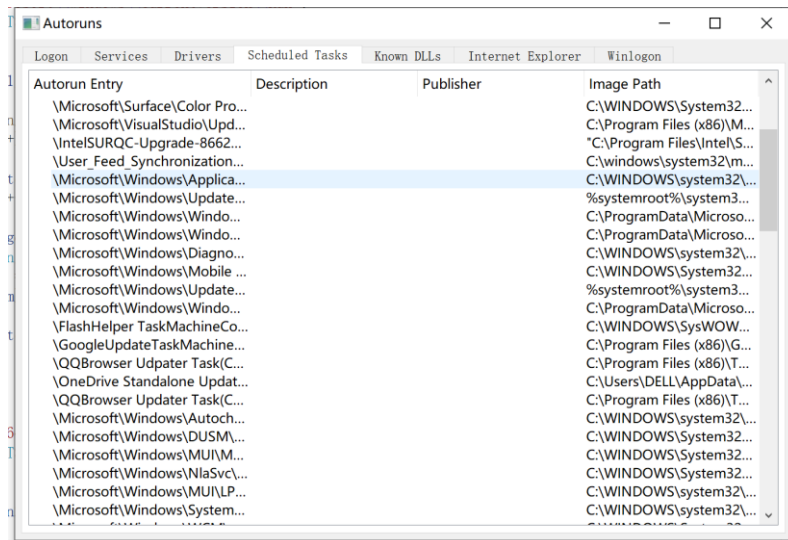


4.4 Scheduled Tasks

此部分功能的实现首先调用 IsAdministrator 提升权限, 然后扫描“C:\Windows\System32\Tasks”文件夹下的所有文件, 如果是目录则访问目录下的所有文件如此递归, 对读取到的文件进行解析获取其中<Command>标记的 ImagePath, 通过 ImagePath 获取 Publisher 和时间戳等信息。流程示意图如下所示:



图形界面:



命令行界面：

```
D:\code_new\VS_Studio\autorun\Debug>autorun.exe
please choose the mode number
1----Logon
2----Services
3----Drivers
4----Scheduled Tasks
5----DDLs
6----IE_BHO
4
Tasks : Autorun entry ; Image Path ; Description ; Publisher ; TimeStamp :

[>] Parsing tasks ...
\Adobe Acrobat Update Task C:\Program Files (x86)\Common Files\Adobe\ARM\1.0\AdobeARM.exe
\AdobeARUpdater-1.0-MicrosoftAccount-ming-quan@outlook.com C:\Program Files (x86)\Common Files\Adobe\AdobeGCClient\AdobeUpdaterStartupUtility.exe
\AdobeGCInvoker-1.0 C:\Program Files (x86)\Common Files\Adobe\AdobeGCClient\AGCInvokerUtility.exe
\AliUpdater(AE069482-6D1A-4E0E-9F7D-67FC017A4180) C:\Program Files (x86)\AliWangWang\AliTask.exe
\OneDrive Standalone Update Task-5-1-5-21-312245638-2312831479-466466543-1001 C:\Users\DELL\AppData\Local\Microsoft\OneDrive\OneDriveStandaloneUpdater.exe
\Microsoft\Office\Office Automatic Updates 2.0 C:\Program Files\Common Files\Microsoft Shared\ClickToRun\OfficeC2MClient.exe
\Microsoft\Office\Office ClickToRun Service Monitor C:\Program Files\Common Files\Microsoft Shared\ClickToRun\OfficeC2MClient.exe
```

4.5 DDLs

此部分与基于注册表的自启动项部分类似，读取键

HKEY_LOCAL_MACHINE\System\\CurrentControlSet\\Control\\Session Manager\\KnownDlls 下的所有键值，此键下的值均为 REG_SZ 类型，名称为相应的 dll 名称，数据即为相应的 dll 文件，这些文件均在文件夹“C:\windows\syswow64”下，受时间限制，未作图形化界面，命令行界面展示截图如下：

```
please choose the mode number
1----Logon
2----Services
3----Drivers
4----Scheduled Tasks
5----DDLs
6----IE_BHO
5
DDLs : Autorun entry ; Image Path ; Description ; Publisher ; TimeStamp :

SHELL32 SHELL32.dll
wowarmhw wowarmhw.dll
coml2 coml2.dll
IMM32 IMM32.dll
gdi32 gdi32.dll
MSCTF MSCTF.dll
MSVCRT MSVCRT.dll
NORMALIZ NORMALIZ.dll
DifxApi difxapi.dll
PSAPI PSAPI.DLL
rpcrt4 rpcrt4.dll
advapi32 advapi32.dll
kernel32 kernel32.dll
```

五、 心得体会

此次大作业难度很高，我在完成的过程中遇到很多困难，一方面是因为很久不用 C++，对其中的很多数据结构、函数传递等基本操作遗忘较多。此外，本次课程设计调用了大量的 Windows API 函数，这些函数的返回值和传递参数的数据类型都有相应限制，大多都要用 LPCSTR, LPCWSTR, LPCWSTR 等变量类型传递、存储相应的数据等，而 C++ 对数据类型的要求又很高，很多次都栽在了数据类型的转换上，很长一段时间我挣扎于种种数据变量的转换。但是又因为第一次使用 QT，不知道其中打印信息需要转换为 QString 等数据变量，我是先写的功能函数，再开始进行界面的制作，开始制作界面的时候很崩溃，因为原先的功能函数我只用输出就可以，根本没有考虑到数据类型的统一，我虽然进行了封装，也大多用字典的数据结构进行传递，但是很难和 QT 兼容，经过一段时间的学习我知道 CHAR* 和 string 的数据类型对 QT 的数据传输最友好，所以又开始大幅修改原函数，很多函数需要大改，而且因为 QT widget application 和我原先创的 C++ 空项目字符编码等等的不同，很多奇妙的错误需要去解决，最后对原函数经过了大量的修改终于制作好了图形界面并完成了所有功能和部分可选功能的制作。

虽然困难重重，但是最后看到自己制作成功的 autorun 还是很有成就感的。因为网上资料较少，在完成课程设计的过程中我又认真观看了课程回放、PPT 等，为了获取相关注册表知识，还阅读了《深入解析 Windows 操作系统》的部分章节，对课堂所学知识有了更深的理解，同时大大加强了 C++ 编程能力以及界面制作的能力。总的来说，这次课程设计使我获益良多，感谢王老师给我此次课程设计的机会和课堂上的认真指导。