

# 杀毒软件开发

全球第一本公开杀毒软件开发技术资料的书

王丽文 著

2013年1月1日

“这本所谓的书，是我在十多年前做杀毒软件开发时所写。以一个入门级杀毒软件案例为基础、以源码解读的方式分享了开发杀毒软件开发的一些技术，包括：特征码扫描、云杀毒、感染型病毒原理及查杀、启发式杀毒技术，R3层API Hook防护等…。当年，曾将此资料公开分享过。2024年末，我发现依然有人在找这份资料，虽然感觉当年写的有些差劲，更不够完善，但既然有人需要，emmm...再Share之，从很久不用的移动硬盘找到了这份底稿，上传到GitHub。”

JShaman.com w2sfot  
2024. 11. 28

## 前 言

本书将向各位读者展示如何开发杀毒软件。

在很多人思维中，特别是IT从业者、程序员看来，杀毒软件及其开发技术历来是一个颇为神秘不可及的领域。在市面上和网络中的各种文章、书集中，也鲜有涉及此方面的开发资料。正因如此，使得杀毒软件业成了一个稀缺、高门槛的行业，相关技术也似乎是高度机密的资料、只掌握在极少数人手中。

本书将从杀毒软件开发方案、功能结构设计、界面设置、代码编写、实际应用等各方面，逐步展示如何开发一款具功能完善的大众化杀毒软件。以此揭密杀毒软件开发各方面技术，引领对此领域有兴趣或有志于此行的读者朋友们入门杀毒软件开发！

杀毒领域技术博大精深，本人勤奋有余、智慧不足，所能研究领悟的也只是沧海一粟，尽数展示在本书中，权当抛砖引玉，以资先辈和大师们笑评。

# 目 录

1. 关于杀毒软件, 你需要知道的 .....	1
1.1. 杀毒软件的重要性 .....	1
1.1.1. 学习杀毒软件开发的重要性 .....	1
1.2. 杀毒软件技术难吗? .....	1
2. 杀毒软件功能设计 .....	2
2.1. 杀毒 .....	2
2.2. 防毒 .....	2
2.3. 升级 .....	3
2.4. 自我保护 .....	3
2.5. 黑白名单 .....	3
2.6. 设置 .....	4
2.7. 界面 .....	4
2.8. 其它 .....	4
3. 功能实现 .....	5
3.1. 杀毒实施方案及编码实现 .....	5
3.1.1. 病毒库简介 .....	10
3.1.2. 特征码 .....	10
3.1.3. 如何定义病毒名称 .....	10
3.1.4. 病毒库设计 .....	10
3.1.5. 优化病毒库 .....	11
3.1.6. 特征码加载 .....	12
3.2. 防毒实施方案实编码实现 .....	18
3.2.1. 云安全实施方案的引入 .....	19
3.2.2. 云安全服务端编程 .....	27
3.3. 升级实施方案及编码实现 .....	28
3.4. 自我保护设计方案及编码实现 .....	33
3.5. 黑白名单设计方案及编码实现 .....	36
3.6. 软件设置实施方案 .....	38
3.7. 杀毒软件界面设计 .....	40
3.7.1. 漂亮界面的设计技巧: 图片替换控件法 .....	40
3.7.2. 图片替换法的高级应用: 界面换肤 .....	41
3.7.3. 扫描自定义目录界面的实现 .....	42
3.8. 其它辅助功能设计 .....	47
3.8.1. 右键扫描杀毒 .....	47
3.8.2. 软件和病毒库自动升级 .....	50
3.8.3. 病毒隔离 .....	51
3.8.4. 提取文件特征码 .....	51
4. 杀毒软件功能设计总结 .....	51
5. 完整的杀毒软件代码剖析 .....	52
5.1. 杀毒软件工程构成 .....	52
5.2. 杀毒软件各窗体及模块功能概述 .....	53
5.3. 窗体文件说明及代码剖析 .....	55
5.3.1. 主动防御中拦截软件启动窗体 .....	55
5.3.2. 屏幕右下角弹出的消息提示窗体 .....	61
5.3.3. 扫描病毒发现病毒时的提示窗体 .....	66
5.3.4. 黑白名单文件管理窗体 .....	71
5.3.5. 隔离文件管理窗体 .....	79

5.3.6. 右键扫描结果窗体 .....	85
5.3.7. 杀毒软件主界面窗体 .....	90
5.3.8. 病毒扫描完成窗体 .....	163
5.3.9. 右键菜单设计窗体 .....	167
5.3.10. 病毒自定义扫描窗体 .....	172
5.3.11. 软件功能设置窗体 .....	182
5.3.12. 软件换肤窗体 .....	197
5.3.13. 软件及病毒库升级窗体 .....	202
5.4. 模块文件说明及代码剖析 .....	214
5.4.1. 主动防御模块 .....	214
5.4.2. 软件开机自启动实现模块 .....	216
5.4.3. 云安全防护实现模块 .....	217
5.4.4. 可执行文件图标获取模块 .....	219
5.4.5. 获取文件或文件节的哈希值模块 .....	220
5.4.6. Ini 文件读写操作模块 .....	224
5.4.7. 特征码加载和验证模块 .....	226
5.4.8. 特征码匹配检测模块 .....	233
5.4.9. PE 文件操作和文件扫描模块 .....	236
5.4.10. 进程优先级设置模块 .....	243
5.4.11. 公共函数模块 .....	244
5.4.12. 公共变量模块 .....	247
5.4.13. 右键菜单关联与解除模块 .....	248
5.4.14. 自我保护功能实现模块 .....	250
5.4.15. 窗体消息处理模块 .....	251
5.4.16. 添加和移除托盘图标模块 .....	262
5.5. 控件文件说明及代码剖析 .....	264
5.5.1. 软件及病毒库升级控件 .....	264
5.6. DLL 模块说明及代码剖析 .....	270
5.6.1. 主动防护 DLL 模块功能说明及代码剖析 .....	270
5.6.2. 自我保护 DLL 模块功能说明及代码剖析 .....	275
6. 感染型病毒的杀毒方案与编程实现详解 .....	277
6.1. “感染型病毒”源代码及功能说明: .....	278
6.2. “感染型病毒”运行测试及“感染文件”演示 .....	286
6.3. 编程清除“感染型病毒” .....	287
7. 启发式杀毒技术原理与编程实现 .....	292
7.1. 启发式杀毒技术原理 .....	292
7.2. 启发式杀毒编程实现 .....	293
8. 后记 .....	298

## 1. 关于杀毒软件，你需要知道的

### 1.1. 杀毒软件的重要性

电脑的使用已普及到社会生活的方方面面，无论是工作、娱乐、学习、购物等等都会接触到电脑。使用电脑实际上是使用电脑中的各种软件。

什么软件重要呢？对于不同的人、不同的目标，答案也不尽相同，但是无论做什么，安全自然是不容忽视的：网上购物需要确保交易安全、资金安全；聊天娱乐需要保护隐私；工作学习需要防止资料、机密泄漏，即便没有这些诸多考虑只是随意上网也要需要一个安全稳定的操作系统环境，这一切需要被人们担心、防范的又是什么呢？是木马、病毒、恶意软件。而杀毒软件正是用来对付这些威胁、保护电脑正常使用的必不可少的软件。

#### 1.1.1. 学习杀毒软件开发的重要性

对于大多数普通的电脑使用者来说并不重要。

绝大多数的电脑使用者并不需要学会开发杀毒软件，也不需要懂杀毒原理，会用就足够了。但对于某些人也许很重要，有的需要了解、有的需要熟悉，甚至有的需要掌握。

电脑行业相关从业者，譬如网管、空间服务商、系统管理员或其它从事系统安全维护人员等，有必要多了解杀毒软件相关知识，熟悉杀毒软件功能重点，以及防杀毒原理，以便在进行选用杀毒软件或部署防杀毒安全方案等操作时有更多好的选择。

因为杀毒软件的重要性所至，绝大多数的电脑都会安装杀毒软件，因此杀毒软件的目标用户群非常之大、市场非常广阔。也就意味着从业者的前景较为光明。有条件以此方向进行创业是个不错的选择，当然并不是指跟国内外几大杀毒巨头抗衡，现存的一些小众反病毒安全产品也有不错的生存空间。

在安全公司或开展安全相关技术项目的公司供职者，如果自己掌握着此方面技术，想必在工作中更会如鱼得水。

最后，从兴趣或是研究、学习角度讲，如果以广义的技术含量而论，杀毒技术无疑居于金字塔的顶端，是一个门槛级高的领域，掌握了其技术，就等于掌握一门稀缺的技艺，何乐而不为呢？

### 1.2. 杀毒软件技术难吗？

不难。

杀毒软件开发技术之所以在人们眼中高深莫测，是因为通常无法窥得其门禁所至。就好像要做一件从未做过的不一般事情一样，会觉的无从下手。

造成这种境况的原因源自相关资料的匮乏，就像在武侠小说中，常会有某门派高等功夫很厉害，却传男不传女、传内不传外，更没有印刷量为5000册的秘籍。结果造成江湖人士们天天听说，却见不得、习不到。感觉就是很好很强大，自己学不会。

杀毒软件开发技术便像是如此，掌握在全地球为数不多的几家或几十家企业手中，用户能听到的往往是某某杀毒引擎、启发式技术甲、主动式技术乙、八重防御系统、九层监测技术等这些貌似很厉害的字眼，用户无法判断其到底有多么强大，只能中这些晦涩神奇的词句中下意识的自我判断，书店中各种类的也更加使人们觉的杀毒是一个神秘不可及的领域。

而事实上，杀毒技术真的有那么难吗？

不可否认，任何领域的任何事情做到极致都是很难的，杀毒技术确实有很难的部分，但如果想入门、想掌握，并不会是想像中的那么困难。很多内容其实是已知的，但只是之前没有把它跟杀毒开发联想起来，只是以前没有想到杀毒软件是用如此的方法开发的，只是以前不知道如此技术有如此用法。

相信很多人读完本书后会有这样的感觉：原来如此，不难，我也可以掌握的。

## 2. 杀毒软件功能设计

进行杀毒软件开发之前，首先需要明确功能。

杀毒软件需要具备的基本功能分为：杀毒、防毒、升级、自我保护、设置以及其它辅助功能。

### 2.1. 杀毒

查毒杀毒，是杀毒软件的基本功能。

在实际的软件应用中，用户通常有不同的操作需求，比如有要要对内存和敏感系统区域进行快速扫描、有时需要进行全硬盘扫描、有时临时使用U盘，插入U盘后，需要对U盘进行单独扫描。

为了满足这些需求，通常在杀毒软件中提供三种扫描方式：快速扫描、全盘扫描、自定义扫描，也就是经典的三按钮式杀毒功能。国内外很多杀毒软件都采用了这种方式，并不是跟风，而是这样的设计确实实用、确实能满足用户的需求。

既然进行病毒扫描，理所当然可能会扫描到病毒。在扫描到病毒后，需要提供给用户相应的操作措施，如清除、自动清除、忽略、隔离。

### 2.2. 防毒

在一定意义上而言，防毒功能的重要性甚至大于杀毒功能。如果用户有忧患意识，在安装系统后及时安装好杀毒软件，做好病毒防护工作，时刻阻止病毒侵入系统，远比中毒后再杀毒效果要好的多，亡羊补牢在电脑使用过程中是常常会晚的。因此，防毒功能是杀毒软件另一个非常重要且不可缺的部分。

在这里要对云安全进行一些简单的介绍，因为本书中对杀毒软件的设计中，将要在防毒功能中引入云安全概念，这是一个与常见的主流杀毒软件不同的防毒理念。

云安全，是个新兴的技术，高度依赖互联网、依赖带宽、依赖用户量，是互联网高度发展衍生出的一种技术。在杀毒软件领域中，具有极高的实用性。目前主要的杀毒软件中，大都有云安全应用的身影，甚至有完全的云查杀

软件。使用云技术，利用云病毒库进行病毒的检测和查杀。

但本人认为，云安全技术在这一领域更适用来做防护，而不适用做杀毒。原因在于：云查杀，是在建立云服务器的基础上，将病毒库特征码置于服务器数据库中。用户在本地进行扫描时获取到本地文件特征码，然后需要连接到服务器，进行特征码匹配验证，以确定被扫描的文件是否是病毒。网络验证，不管当前的网速达到了多快，但其速度是永远无法与本地相比的，进行文件扫描查杀，用户需要速度，而此举降低了扫描速度。况且，存储于服务器中的病毒库，是完全可以以升级病毒库的方式下载到客户机器中，进行高速扫描，这是传统杀毒软件惯用的方式，没有必要，也没有理由非存放在服务器。再则，云的思想，原本是将工作量分散到大量的互联网终端执行，而现行的这种云查杀，反而是将大量的客户端工作量集中到服务器执行，根本有驳云的思路。因此，云查杀不是一种优势的做法。

而如果将它应用到防护方面，在大数量用户的基础上，可以做到自动、快速的病毒特征码提取，将大量互联网客户端的数据上交到服务器，并且可以将特征码即时放置在云服务器病毒库，在有用户查询时，又能做到即时的反馈，直接提高了对新出病毒的反映速度。而这种防护式的云检测，对速度要求远低于云查杀的强度。

介于以上原因，本书中设计的杀毒软件，将使用此种云安全防护技术。

### 2.3. 升级

杀毒软件，需要带有病毒库，病毒库是病毒特征码的集合，杀毒软件性能的强弱在很大程度上依赖于病毒库。病毒库是很庞大的，通常有几十MB，甚至上百MB，因为它要存放几十万上百万的病毒特征码。最新最全的病毒库放置在网络中的服务器端，为了实时的将病毒库更新到客户机器，便需要用到下载升级功能。这是杀毒软件的辅助的功能，却也是必不可少的。

另一方面，升级功能也用于更新杀毒软件的功能模块，以方便户在不必重新下载安装软件即可使用到最新最稳定的功能。

### 2.4. 自我保护

杀毒软件需要一定的自我保护能力甚至是绝对的自我保护能力，以防止被病毒或其它恶意软件强行将自身进程结束，换句话说，即是要防止没杀掉病毒反被病毒杀掉。

很多病毒木马为了提高生存率，会与杀毒软件对抗，反杀杀毒软件。比如某些病毒木马在运行时，会检测系统中是否已经安装了杀毒软件，如果检测到，会想尽办法终止杀毒软件，以方便进一步入侵系统。此时，杀毒软件必须有足够的能力自保，以确认能够击败病毒。

当然，杀毒软件的主要功能是保护系统的正常运行，而不是与病毒木马进行技术对抗，在使用自我保护功能时，需多方面考虑，即不过份影响系统性能和稳定性又能确保系统安全为最佳。

### 2.5. 黑白名单

黑白名单是一个简化的特征码库，辅助病毒库工作。

有了病毒库，为什么还要使用黑白名单？

病毒库中存放的是病毒木马的特征码，有某些情况下，除了病毒木马还有一些边缘软件，不属于病毒木马范畴，不会对系统造成危害，但它会对系统产生一些不好的影响，比如某些P2P在线影音软件的后台程序，会将机器做为一台种子机器，不停的上传下载电影，占用大量宝贵的带宽和流量、比如某些聊天软件登录时弹出的广告等等。这些软件是正常电脑使用不需要的，但它却是某些正常软件附加软件，出于多种原因的考虑，不能将其列入到病毒库，那么只能使用黑名单功能阻止其运行。

甚至是系统中存在的某些正常软件，因为一些个人原因想阻止它的启动，等等这些都可以使用黑名单功能。

至于白名单，与此恰恰相反。比如一些调试工具，往往被归为恶意工具的一种而被列入病毒库，如果出于工作或其它需要需要使用这些工具，则需要使用白名单功能使其能够运行而不被杀毒软件阻止。

归结起来，黑白名单的存在，是为了扩充病毒库的使用范围，和减少杀毒软件的误报率。

## 2.6. 设置

杀毒软件需要提供合理自由的设置选项，对软件功能进行配置，以符合用户的使用习惯。

比如是否使用右键菜单。文件及文件夹的右键菜单可以使扫描病毒等工具简单化，但有的用户却不喜欢在右键菜单中增加过多的内容而宁可使用自定义扫描。

比如是否要让软件在系统启动后自动运行、是否开启自动保护、是否启用云安全等等。

细节决定成败，用户使用软件的舒适度，决定着用户对软件的忠诚度。

## 2.7. 界面

除了功能性方面，软件界面也是不可忽视的一部分，对用户而言，基本功能相近的情况下，选择一款软件时，往往会根据对软件界面的第一感觉进行选择。

界面不单单是美工、设计的事，在编程中巧妙的利用技巧对界面进行美化、优化也会起到非常好的效果。比如可以做换肤、动态按钮、显示特效等等。

## 2.8. 其它

除上面介绍到的具体功能外，软件中还需要对版本号、病毒库日期等用户较为关心的相关信息在界面中合理的位置进行显示。在以上的设计理念中，多次提到了对用户的感受，之所以要多次提及，是因为不管设计任何软件，都是以用户使用、为用户解决问题为目的，设计杀毒软件当然也不例外，一方面设计功能，一方面也要考虑用户体验。

以上，对于功能的设计已经写完了，到此也许有疑惑：杀毒引擎呢？主动防御呢？为什么都没有说到。

诚然，在各种杀毒软件的推广宣传词中，常会见到各自引擎的名称，甚至是双引擎、四引擎，好似引擎越多功能越强大，当然在大多数用户认知当中也确实会是这样。那么杀毒引擎倒底是什么呢？它是检测和查杀病毒的方式，比如在扫描一个文件时，首先获取它的特征码，然后与病毒库中的病毒特征码进行比对，如果确认是病毒，对它进行查杀，如果是感染型病毒，需要对文件重写，还原文件入口点，等等这一系列操作组合整合起来就是杀毒软件的



引擎。对源程序而言它就是一部分代码。同理，主动防御也只是防毒功能而已。

是否有这样一种感觉：宣传中那样神秘莫测的杀毒引擎原来不过如此。

### 3. 功能实现

在上面的章节中，对软件的功能进行了初步设定，接下来将以编程的思路，将这些功能进行梳理、整合，确定模块化的编程实现方案，并编码实现。

本书中的编码中，会使用到VB、VC、ASP、汇编四种编程语言，主要功能由VB编写、某些功能需要使用DLL，DLL代码由VC编写，ASP语言用在云安全后台，汇编是在某些环节在DLL中内嵌使用。要深入理解本书内容，需要对此三种编程语言熟悉，最低要求是有一定的了解，能够读懂相关代码，本书非编程语言教程，不讲解编程基础知识。

也许有程序员会质疑：VB能写的了杀毒软件吗？

用事实说话往往是最有说服力的：微软发行的Microsoft AntiSpyware软件，便是用VB开发而成的。

规则约定：在本书代码中，对系统API函数，使用粗体字，复杂的API函数，会进行解释说明。对于重要的自定义的函数、变量，使用之处用加粗加大字体。

#### 3.1. 杀毒实现方案及编码实现

如上文所讲，本软件中将杀毒分为三种模式：快速扫描、全盘扫描、自定义扫描。

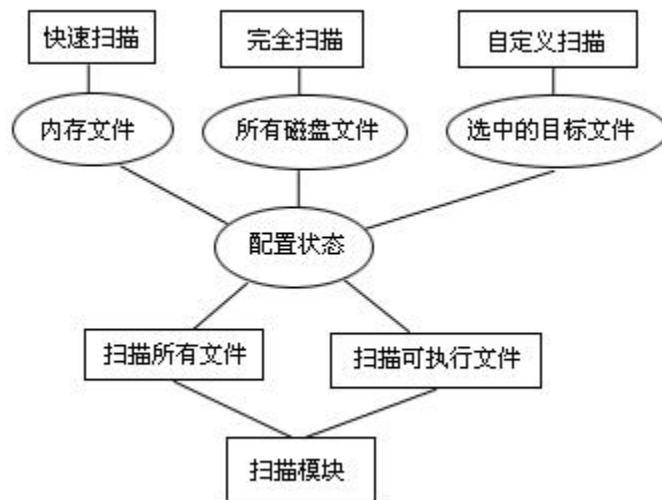
这三种扫描方式，展现在用户面前时是3种不同的操作，而在程序中的编码是非常相近的，不同在于扫描的目标文件对象不同。

快速扫描对加载在内存中的文件进行扫描；

全盘扫描对整个硬盘中的文件扫描；

自定义扫描对选中的目标文件进行扫描。

如下图所示：



在此，以快速扫描为例，进行编码说明。

快速扫描要扫描内存中的文件，如何确定哪些文件在内存中被使用呢？这里我们通过遍历系统活动进程及进程相关模块文件的方式来实现。

编码如下：

```

Dim lProcess As Long
lProcess = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0)

Dim uProcess As PROCESSENTRY32
uProcess.dwSize = Len(uProcess)

If Process32First(lProcess, uProcess) Then

    Dim lModule As Long
    lModule=CreateToolhelp32Snapshot(TH32CS_SNAPMODULE, uProcess.th32ProcessID)

    Dim ModEntry As MODULEENTRY32
    ModEntry.dwSize = LenB(ModEntry)
    ModEntry.szExePath = ""

    If Module32First(lModule, ModEntry) Then

        Dim sFile As String
        sFile = TrimNull(ModEntry.szExePath)

        Dim sScanResult As String
        sScanResult = ScanFile(sFile)

        Loop While Module32Next(lModule, ModEntry)

    CloseHandle lModule
    Loop While Process32Next(lProcess, uProcess)
  
```

**CloseHandle** lProcess

以上是遍历系统进程及相关模块的代码，也就是检测系统内存中加载着哪些文件。其中，使用API函数CreateToolhelp32Snapshot获取系统进程快照，然后使用Process32First和Process32Next函数获取进程列表、使用Module32First和Module32Next获取进程使用的各个文件，取得了文件列表就可以对文件逐个进行扫描了。

这里没有读取配置文件区分是否扫描执行文件或是全部文件，配置文件的使用功能放在后面部分单独讲述。

上面的代码中ScanFile函数的sFile参数变量中存放的是加载在内存中的各文件完整路径，在获取到文件路径后，使用ScanFile函数对文件进行扫描，ScanFile函数是实现文件病毒扫描的功能。编码如下：

```
Public Function ScanFile(sFile As String) As String
ScanFile = ""
Dim lFileHwnd As Long
lFileHwnd = CreateFile(sFile, ByVal &H80000000, 0, ByVal 0&, 3, 0, ByVal 0)

Dim bBuffer(12) As Byte
Dim lBytesRead As Long
Dim uDosHeader As IMAGE_DOS_HEADER
```

```
ReadFile lFileHwnd, uDosHeader, ByVal Len(uDosHeader), lBytesRead, ByVal 0&
CopyMemory bBuffer(0), uDosHeader.Magic, 2
```

检查PE文件标志

```
If (Chr(bBuffer(0)) & Chr(bBuffer(1))) = "MZ" Then
```

设置偏移量，指向PE头结构

```
SetFilePointer lFileHwnd, uDosHeader.lfanew, 0, 0
ReadFile lFileHwnd, bBuffer(0), 4, lBytesRead, ByVal 0&
```

再次检查是否是PE文件

```
If (Chr(bBuffer(0)) = "P") And (Chr(bBuffer(1)) = "E") And (bBuffer(2) = 0) And (bBuffer(3) = 0) Then
```

再进一步调用下级函数扫描文件是否是病毒，如果是病毒则返回病毒名，否则返回空值

```
Dim sScanSectionResult As String
sScanSectionResult = ScanSections(lFileHwnd)
```

如果返回不为空，表示检测到是病毒，则关闭文件，返回

```
If sScanSectionResult <> "" Then
ScanFile = sScanSectionResult
CloseHandle lFileHwnd
Exit Function
End If
End If
End If
CloseHandle lFileHwnd
End Function
```

上面已对部分关键代码做了注释，这里再对代码功能做具体介绍。此函数中，sFile是传入的待扫描文件名，先使用CreateFile函数打开文件，然后检测两处文件标识对文件类型进行判断，确定是PE文件(也就是可执行文件)后，再调用下级函数对文件进行扫描，因为病毒木马都是可执行文件，而不可能是文本文件，所以只有扫描可执行文件才有意义，只有可执行文件才可能是病毒木马。

也许有人注意到，到之前介绍扫描方式时，方案中设计了两种扫描方式，一种为扫描可执行文件，一种为扫描所有文件，那这里为何又要对文件格式进行判断只扫描可执行文件呢？这是因为：在windows系统中，文件的后缀名是可以自由更改的，前面所提到的是从表面上根据后缀名进行分类，这种分类较为直观，但不够严谨，这里再次进行判断是为了弥补根据后缀名判断的不足。

这部分代码中，要扫描PE文件，涉及了PE文件结构的知识，那么就需要对PE文件结构有一定的了解。

PE文件是Windows系统上的可执行文件，PE的全称是Portable Execute，即：可移植的执行体。常见的EXE、DLL、OCX、SYS、COM都文件都是PE文件。PE文件结构较为复杂，但在此只需了解部分相关知识，没有必要详尽说明，我们在程序中需用到的知识点是：判断一个文件是否是PE文件、获取PE文件的节内容，如节的大小、节的哈希值。

首先来看一下PE文件结构图：



PE文件是由多个节组成的，每个节都是一个相对独立的部分，相互之间又有着关联。在文件的开始部分，首先是三个头部分，分别是：DOS头、PE文件头、可选文件头，存储着PE文件的各种属性信息，比如文件的入口点、节的个数、节的属性、节的位置等等。后面跟着的是节头和各节的数据，比如代码节、资源节等。每个节头对应一个节，一个文件可以有多个节头和节，节头中保存着节的名称、地址等重要信息，根据这些信息可以定位到节的地址，进而可以对节的内容进行读取。在后面的内容中，会根据这些基本信息和思路获取文件节的内容、节的特征码。

要操作PE文件，就要在程序中定义与之相同的文件格式。首先在程序中需要定义的是DOS头，需如下编码：

```
Private Type IMAGE_DOS_HEADER
    Magic As Integer
    cblp As Integer
    cp As Integer
    crlc As Integer
    cparhdr As Integer
    minalloc As Integer
    maxalloc As Integer
    ss As Integer
    sp As Integer
```

```

csum As Integer
ip As Integer
cs As Integer
lfarlc As Integer
ovno As Integer
res(3) As Integer
oemid As Integer
oeminfo As Integer
res2(9) As Integer
lfanew As Long
End Type

```

DOS头中除了最后的lfanew和e\_magic两个变量，其它内容对我们都不重要。这个部分，我们关心的仅为lfanew，该值指向了PE头结构在文件中的偏移。假设该值为0x40，那么文件中0x40位置开始就是PE头的开始，标志为“PE”两个字符。判断文件是否是PE文件，就是依靠这个。

PE结构第二部分：文件头：

```

Private Type IMAGE_FILE_HEADER
Machine As Integer
NumberOfSections As Integer
TimeDateStamp As Long
PointerToSymbolTable As Long
NumberOfSymbols As Long
SizeOfOptionalHeader As Integer
Characteristics As Integer
End Type

```

这部分没有实际操作意义，但要定位到PE文件的相应位置，它在程序中编码时必须存在。

PE结构第三部分：可选文件头：

```

Private Type IMAGE_DATA_DIRECTORY
DataRVA As Long
DataSize As Long
End Type

```

DataRVA表示指向的地址是虚拟地址，就是程序加载后该地址在内存中相对于ImageBase的偏移，带Virtual的地址是这样：在文件中的偏移为：Addr - 对应Section的VirtualAddress + 对应Section的文件内起始偏移。

PE结构第四部分：节头：

```

Private Type IMAGE_SECTION_HEADER
SectionName(7) As Byte
VirtualSize As Long
VirtualAddress As Long
SizeOfRawData As Long
PointerToRawData As Long
PointerToRelocations As Long
PLineNums As Long
RelocCount As Integer

```

```
LineCount As Integer
Characteristics As Long
End Type
```

变量SectionName中存储的是节的名称，比如.Text、.Res，占8个字节；变量的VirtualAddress是节的RVA，即虚拟地址；变量SizeOfRawData是节的大小；变量PointerToRawData是节的文件偏移量。这几个变量很重要，我们要依靠它们来获取节的内容，进而取得特征码。

这里提到了特征码，在接下来进一步扫描文件时，我们使用特征码匹配技术对文件进行检测，判断是否是病毒木马。为了便于理解之后的代码，我们先对本书中设计的杀毒软件的特征码、病毒库规范进行一些讲解。

### 3.1.1. 病毒库简介

病毒库是杀毒软件很重要的组成部分，影响杀毒软件的查杀、防护性能，病毒库由病毒特征码组成，是特征码的集合。病毒库可以是一个单独的文件，也可由多个文件组成。

### 3.1.2. 特征码

病毒特征码是一组字符串，由特定的规则组成，用于标识某一病毒文件的特征，在编程中通过一定规则的特征码匹配校验，可以确定某文件是否是病毒。

特征码定义方式很有多种，比如根据文件特定位置的字节信息、根据PE文件格式使用的API组合、根据文件节信息等，甚至文件的MD5码也属于文件独有的特征码。

在本文中将使用“PE文件节大小”+“对应节的哈希值”做为特征码。

### 3.1.3. 如何定义病毒名称

在国际上，有一套病毒名称定义标准。对某病毒根据不同的性质偏向冠以不同的前缀加以详细区分。比如这些种类有：Virus（病毒）、Trojan（木马）、Worm（蠕虫）、AdWare（广告）、HackTool（黑客工具）、Exploit（溢出程序）、W32（Win32恶意文件）等。

比如我们获取到一个以远程控制为主要功能的病毒，我会可能会将它命名为:Trojan. a. b。

### 3.1.4. 病毒库设计

有了特征码、病毒名称，将大量的特征码和病毒名称依照一定的规则存储在一起，即形成了病毒库。

在本杀毒软件设计中，为了起到保密作用，防止特征码被他人非法使用，将前文的特征码字符串进行加密，加密后存储在本地病毒库中。

例如：某文件Test.exe中，.Text节的大小为1024，节内容的Hash值为1234567890abcdef，则其特征码为：“1024:1234567890abcdef”，再将此特征码字符串加密，得到加密的特征码：abcdef1234567890，再附上病毒名

称，就得到了最终特征码组合：abcdef1234567890Trojan. a. b，这时便可以进行存储了。

### 3.1.5. 优化病毒库

实现了特征码的存储，这还不够。还要考虑它的使用性能，如果特征码数据量达到一定级别，比如100万。那么逐一加载和匹配校验耗时就成了制约杀毒软件性能的一个瓶颈。诚然，数据量越大加载和匹配校验需要的时间就越长，这是一个根本的事实。但我们要尽可能的优化，加快这一过程。

#### 3.1.5.1. 病毒库加载速度优化

加载速度的优化，我们采用这种的方案来实现：给每个组合过的特征码字符串增加足够的长度，使所有字符串长度统一。

比如：

特征码1：“abcdef1234567890Trojan. a. b”

特征码2：“abcdef1234567890Virus. a. b”

为了让这两个特征码长度一至，我们给它的末尾增加空格，使其成为：

特征码1：“abcdef1234567890Trojan. a. b ”

特征码2：“abcdef1234567890Virus. a. b ”

然后，在加载时，不对每条数据进行逐一读取，而是一次性将整个病毒库文件读入内存，这个操作在速度上会比逐一读取特征码快很多。

因为每个特征码的长度是统一的，我们就能以固定长度分隔特征码，分隔后也就把整块的内存数据赋值给了程序中代表特征码的字符串数组。

#### 3.1.5.2. 特征码匹配检测速度优化

如果有100万条数据，每扫描一个文件时，都对这100万条数据进行一次匹配检查，显然是会非常耗时的。因此，特征码匹配检测速度优化是非常有必要。我们采用如下的办法：

在加密初始特定码时，使用Md5加密方法，不管加密前的特征码是哪种形式，加密后，将都是一个32位固定长度的字符串，且首字母将会是16个字符：0123456789abcdef中的任意一个。那么我们将有机会将原有扫描匹配速度提高16倍。

把特征码按字母的不同，分别存储在16个不同的文件中。软件加载时同样按首字母的不同赋值给16个不同的特征码字符串数组。

接下来，在进行文件扫描时，为了能进行匹配，我们也必须把被扫描的文件特征码转化为与病毒库中存储的数据相符的格式。转化之后进行匹配校验之前，先取特征码首字符进行比较。

比如：如果被扫描文件的特征码首字符是“a”，在程序中，只在“a”打头的特征码字符串数组中进行匹配，其它15种情况完全忽略，这样就极大的提高了扫描速度。

定义了16个病毒库文件，我们还是假设数据量为100万条，那么每个病毒库文件的体积都是较大的，可能有数MB大小。这对病毒库升级带来了一定的影响：下载的文件太大，每天可能需要下载数十MB的病毒库。对于病毒库服务器和用户来讲，都是一个不小的负担。为了解决这个问题，我们在16个病毒库文件之外，再增加一个X文件，它里面存储近期升级的少量的特征码。日常病毒库升级时，新的特征码都放置在其中。当积累到一数量时，再将特征码归类置入相应的文件，进行全面升级。

到此，我们了解了病毒库以及内部特征码的存储方式。但在进行特征码检测前，还有一项工程必须进行：特征码的加载。

### 3.1.6. 特征码加载

假设我们已经建产了病毒库文件，文件共17个，文件名分别为：0.sig、1.sig、2.sig、3.sig、4.sig、5.sig、6.sig、7.sig、8.sig、9.sig、a.sig、b.sig、c.sig、d.sig、e.sig、f.sig、g.sig。

文件后缀名sig，含意为signature，即特征码的意思。

文件内容格式为：

```
"072B68E60B25FED5336235962D9C86E0Virus.OnlineGame-1376      "  
"0F5473AA8A8402561C40F050A784F47EVirus.Spy-62220             "  
"0DC38A0DA3B57DBE5E14512DE7A67C72Virus.Spy-62252             "
```

特征码每个占一行，以回车换行结束。特征码在病毒库文件中不使用双引号“”。

程序中，读取之前，首先要定义一个自定义格式。用于存储单个特征码信息：

```
Private Type Signature  
sHash As String * 32  
sName As String * 32  
sVbCrLf As String * 2  
End Type
```

变量sHash是文件特征码，占32个字节；变量sName是病毒名称，占32个字节；此外，在病毒库中每行存储一个特征码，特征码后方还隐形的存在着一个回车换行符。回车换行符占两个字节，因此，在定义格式时，还需要定义一个代表回车换行的标志，上面的代码中变量sVbCrLf便是回车换行的意思，它占两个字节。

然后定义一组公共变量数组，用于存储所有病毒库特征码：

```
Public uSignature0() As Signature  
Public uSignature1() As Signature  
Public uSignature9() As Signature
```

中间省略

```
Public uSignatureA() As Signature  
Public uSignatureF() As Signature  
Public uSignatureG() As Signature
```

定义这些变量时，需要将他们放在公共模块中，并用Public标识，以便使它们可以在工程中的所有窗体和模块中被访问到，方便在其它函数中调用进行特征码匹配校验。



然后编程读取病毒库，将特征码内容填充进这些数组中：

```
Dim sSignatureFile As String
Dim lFile As Long
Dim lFileLen As Long
```

加载0~9.sig病毒库文件：

```
Dim i As Long
For i = 0 To 9
    sSignatureFile = sAppPath & i & ".sig"
```

从病毒库文件加载特征码：

```
Select Case i
Case "0"
    lFile = FreeFile
    Open sSignatureFile For Binary As #lFile
    lFileLen = LOF(lFile)
```

重新定义特征码数组大小，这里数字66的含意是：前面定义的特征码自定义格式中，sHash和sName变量各占32字节，sVbCrLf占2字节，即一个自定义特征码占66字节，因此，使用文件大小除以66，得到的是该病毒库文件中特征码的数量。

```
ReDim uSignature0(1 To lFileLen / 66) As Signature
```

使用get方法获取一次性获取文件所有内容内容，获取后uSignature0数组保已经得到了所有以字符0开头的病毒特征码，使用uSignature0(?)即可访问到具体的数据。

```
Get lFile, , uSignature0
Close lFile
```

接下来使用与上面类似的方法加载A~F.sig病毒库文件，数字65~70使用Chr()函数转化后可以得到字母A~F。

```
Dim j As Long
For j = 65 To 70
    sSignatureFile = sAppPath & Chr(j) & ".sig"
    Select Case UCase(Chr(j))
    Case "A"
        lFile = FreeFile
        Open sSignatureFile For Binary As #lFile
        lFileLen = LOF(lFile)
        If lFileLen <> 0 Then
            ReDim uSignatureA(1 To lFileLen / 66) As Signature
            Get lFile, , uSignatureA
        End If
        Close lFile
    End Select
Next
```

有了这些前期准备，现在可以正式介绍扫描病毒的部分了，这些部分即相当于概念上的杀毒引擎，比较重要，请读者朋友请仔细理解。

```
Private Function ScanSections(hFile As Long) As String
```

```
ScanSections = ""
```

```
Dim uDos As IMAGE_DOS_HEADER
```

```
Dim uFile As IMAGE_FILE_HEADER
```

```
Dim uOptional As IMAGE_OPTIONAL_HEADER
```

```
Dim uSections() As IMAGE_SECTION_HEADER
```

```
Dim lBytesRead As Long
```

```
Dim i As Long
```

```
Dim bTempMemory() As Byte
```

把文件内容读取指针移动到文件头，读取PE文件DOS头信息：

```
SetFilePointer hFile, 0, 0, 0
```

```
ReadFile hFile, uDos, Len(uDos), lBytesRead, ByVal 0&
```

根据DOS头的lfanew值，使文件内容读取指针指向PE头+4的地方(即文件头开始地址)：

```
SetFilePointer hFile, ByVal uDos.lfanew + 4, 0, 0
```

读取文件头内容：

```
ReadFile hFile, uFile, Len(uFile), lBytesRead, ByVal 0&
```

读取可选头内容：

```
ReadFile hFile, uOptional, Len(uOptional), lBytesRead, ByVal 0&
```

```
ReDim uSections(uFile.NumberOfSections - 1) As IMAGE_SECTION_HEADER
```

读取节的个数：

```
ReadFile hFile, uSections(0), Len(uSections(0)) * uFile.NumberOfSections, lBytesRead, ByVal 0&
```

遍历PE文件的每个节：

```
For i = 0 To UBound(uSections)
```

```
ReDim bTempMemory(1 To uSections(i).SizeOfRawData) As Byte
```

```
SetFilePointer hFile, ByVal uSections(i).PointerToRawData, 0, 0
```

```
ReadFile hFile, bTempMemory(1), uSections(i).SizeOfRawData, lBytesRead, ByVal 0&
```

```
Dim sTargetStr As String
```

构造特征码，特征码为：节的大小+节的哈希值：

```
sTargetStr=uSections(i).SizeOfRawData&"":&LCASE(HashFileStream(bTempMemory))
```

```
Dim bTempStrByte() As Byte
```

```
bTempStrByte = sTargetStr
```

用哈希算法进行加密，得到加密的最终特征码：

```
sTargetStr = HashFileStream(bTempStrByte)
```

使用特征码匹配检测，判断此特征码是否在病毒库中存在，如果存在，表示是病毒，则会返回病毒名称：

```
Dim sMatchResult As String
```

```
sMatchResult = MatchSignature(sTargetStr)
```

如果函数返回不为空表示检测到是病毒：

```

If sMatchResult <> "" Then
ScanSections = Trim(sMatchResult)
Exit Function
End If
Next i

End Function

```

这个函数的主要能两有两处：

- 一、分析PE文件，获取文件每个节的特征码；
- 二、再次调用下级函数对节特征码进行检测，判断是否是病毒。

也许有的朋友会问：这已经多次使用下级函数进行特征码判断了，为什么不把代码顺序写下去，那样代码会更为工整，逻辑会更清晰。这是因为以上调用的每个下级函数都是一个相较独立的功能，且会在代码的其它地方被调次使用，写为独立函数从全局角度看会使函数调用关系更为合理，这了是一个良好的编程习惯。

接下来且看HashFileStream和MatchSignature两个关键函数：

```

Public Function HashFileStream(ByteStream() As Byte) As String

HashFileStream = ""
Dim lCtx As Long
Dim lHash As Long
Dim lFile As Long
Dim lRes As Long
Dim lLen As Long
Dim lIdx As Long
Dim bHash() As Byte
Dim bBlock() As Byte

Dim bStream() As Byte
ReDim bStream(1 To UBound(ByteStream)) As Byte

CopyMemory bStream(1), ByteStream(1), UBound(ByteStream)

```

CryptAcquireContext函数功能：得到系统CS，即密码容器，为接下来使用哈希算法做准备。

参数说明：

参数1 lCtx ： 返回CSP句柄

参数2 ： 密码容器名, 为空连接缺省的CSP

参数3 ： 为空时使用默认CSP名(微软RSA Base Provider)

参数4 PROV\_RSA\_FULLCSP ： 类型

```
lRes = CryptAcquireContext(lCtx, vbNullString, vbNullString, PROV_RSA_FULL, 0)
```

```
If lRes <> 0 Then
```

CryptCreateHash函数功能：创哈希对象，以便可以进行哈希运算。

参数说明：

参数1: CSP句柄

参数2: 选择哈希算法，比如CALG\_MD5等

参数3: HMAC 和MAC算法时有用

参数4: 保留, 传入0即可

参数5: 返回hash句柄

```
lRes = CryptCreateHash(lCtx, ALGORITHM, 0, 0, lHash)
```

```
If lRes <> 0 Then
```

```
Const BLOCK_SIZE As Long = 32 * 1024&
```

```
ReDim bBlock(1 To BLOCK_SIZE) As Byte
```

```
Dim lCount As Long
```

```
Dim lBlocks As Long
```

```
Dim lLastBlock As Long
```

将传入的数据块分隔为多个部分:

```
lBlocks = UBound(ByteStream) \ BLOCK_SIZE
```

得到最后一个数据块:

```
lLastBlock = UBound(ByteStream) - lBlocks * BLOCK_SIZE
```

```
For lCount = 0 To lBlocks - 1
```

```
CopyMemory bBlock(1), bStream(1 + lCount * BLOCK_SIZE), BLOCK_SIZE
```

CryptHashData函数功能: 对数据进行哈希运算。

参数说明:

参数1: 哈希对象

参数2: 被哈希的数据

参数3: 数据的长度

参数4: 微软的CSP这个值会被忽略

```
lRes = CryptHashData(lHash, bBlock(1), BLOCK_SIZE, 0)
```

```
Next
```

```
If lLastBlock > 0 And lRes <> 0 Then
```

```
ReDim bBlock(1 To lLastBlock) As Byte
```

```
CopyMemory bBlock(1), bStream(1 + lCount * BLOCK_SIZE), lLastBlock
```

```
lRes = CryptHashData(lHash, bBlock(1), lLastBlock, 0)
```

```
End If
```

```
If lRes <> 0 Then
```

CryptGetHashParam函数功能: 获取哈希数据大小。

参数说明:

参数1: 哈希对象

参数2: 取哈希数据大小(HP\_HASHSIZE)

参数3: 返回哈希数据长度

```
lRes = CryptGetHashParam(lHash, HP_HASHSIZE, lLen, 4, 0)
```

```
If lRes <> 0 Then
```

```
ReDim bHash(0 To lLen - 1)
```

CryptGetHashParam函数功能: 取得哈希运算数据结果。

参数说明:

参数1: 哈希对象

参数2: 取哈希数据(值)(HP\_HASHVAL)

参数3: 哈希数组

```
lRes = CryptGetHashParam(lHash, HP_HASHVAL, bHash(0), lLen, 0)
If lRes <> 0 Then
For lIdx = 0 To UBound(bHash)
```

构造哈希值, 2位一组, 不足补0:

```
HashFileStream = HashFileStream & Right("0" & Hex(bHash(lIdx)), 2)
Next
End If
End If
End If
```

```
CryptDestroyHash lHash
End If
End If
CryptReleaseContext lCtx, 0
```

```
End Function
```

以上使用的是微软提供的一组用于哈希计算的API函数, 在我们的程序中用于算运算出可执行文件各节的哈希值, 用途是得到文件节的哈希码, 以进行特征码匹配, 检测扫描文件是否是病毒。而用于匹配算法的MatchSignature代码编码如下:

```
Public Function MatchSignature(ByVal sHash As String) As String
MatchSignature = ""
If sHash = "" Then
Exit Function
End If
```

取哈希码值的第一位为标志, 确定使用特征码类型:

```
Dim sFlagWord As String
sFlagWord = UCase(Left(sHash, 1))

Dim i As Long
Select Case sFlagWord
Case "0"
For i = 1 To UBound(uSignature0)
If uSignature0(i).sHash = sHash Then
```

匹配成功, 在病毒库中找到了该特征码, 返回病毒名称:

```
MatchSignature = uSignature0(i).sName
Exit Function
End If
Next i
```

```
Case "1"
For i = 1 To UBound(uSignature1)
If uSignature1(i).sHash = sHash Then
```

匹配成功, 在病毒库中找到了该特征码, 返回病毒名称:

```
MatchSignature = uSignature1(i).sName
```

```
Exit Function
End If
Next i

End Function
```

此处代码的功能是用于检测实现从PE文件中运算出的节特征码与病毒库中的特征码是否匹配，如果匹配则返回病毒名称。

到此，扫描病毒的功能的编码都已实现。

这里实现的仅仅是非感染型病毒木马的检测。在查杀时，仅需要简单的结束病毒进程、删除病毒文件即可完成。在近年来，绝大多数的病毒木马都是这种非感染型的，本世纪初及上世纪末DOS时代流行的感染型病毒已并不多见，当然并不是说完全不存在。对于感染型病毒的检测方式，与此大至相同，各位读者可以触类旁通使用类似的方法进行检测，只是在查杀方法上与此有所不同。并不能只是简单的进行文件删除操作，还需要对已感染的文件进行重写。

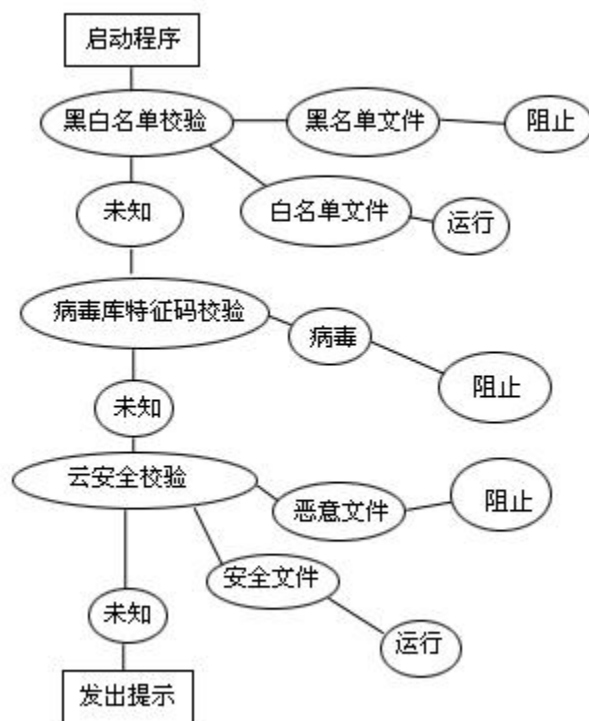
启发式病毒扫描技术是与此种特征码匹配检测完全不同的病毒木马检测技术。启发式也可以理解为行为分析，即根据病毒木马对系统入侵的行为进行检测，比如木马进入系统后，为了能够长期获得对系统的控制权，最基本的一点是要实现开机自启动，实现的方式可以通过写注册表自启动键值、关联文件类型、向系统添加服务、使用计划任务等等。木马的这些行为，会使系统配置、设置发生变化，在对系统相关信息熟悉的情况下，通过检查相关配置位置的内容，便可以发现木马入侵的蛛丝马迹，进而揪出木马本身。这一套操作，通过编程的方法进行实现便形成了启发式杀毒。当然，这只是实现启发式的一种方式，除此之外，还有其它多种实现方法，比如通过检测PE文件的函数导入列表中是否含有某种组织的函数、通过内置反汇编引擎判断PE文件的反汇编代码等等。

在近年来，有的病毒木马也会使用Rootkit技术，通过SSDT HOOK、IAT、API HOOK等技术隐藏自身进程、文件、注册表项。使用户在日常的操作和使用电脑过程中根本无法意识到病毒的存在。因此检测和清除Rootkit病毒的难度较大，好在魔与道总是并存的，有Rootkit也就有与之对应的Anti Rootkit工具的存在，专门用于检测和清除Rootkit，破掉了隐身外衣，其查杀方式就与普通病毒木马无异了。

### 3.2. 防毒实现方案实编码实现

防毒功能实现需要拦截进程启动行为，在一个进程启动前，先对其进行扫描，判断是否是病毒，如果是病毒则立刻进行阻止。这也就是许多杀毒软件宣传的主动防御功能。

为了实现拦截进程启动功能，我们这里要使用API Hook技术。拦截到进程时，获取到它的文件特征码，并在本地病毒库、黑名单、云病毒库中进行匹配校验，如果检测到是病毒，直接阻止。其程序流程如下图所示：



实现拦截进程启动，可以由多种技术手段实现。可以修改SSDT表通过驱动实现，这是较底层的方式；也可以在用户层进行全局DLL注入实现。两者各有优劣。驱动方式更加接近系统底层，控制权限更高，但容易对系统稳定性造成影响，并且杀毒软件大多采用驱动方式，这样便容易引发同类软件冲突、与同类软件不兼容等情况。DLL方式工作在用户层，对系统稳定性及性能影响小，并不易有同类软件兼容性问题。本杀毒软件设计中采用DLL注入方式。

大多数杀毒软件的防护功能，不仅仅会拦截进程启动，还会对注册表读写、文件读写等众多操作进行监管。而本设计中，只拦截启动行为，这样做的原因在于：防护的最核心最关键的就在于进程执行这一步，如果这一步被病毒通过，那么病毒可能会读写文件，可能会读写注册表，也可能会格式化磁盘，甚至会强行删除文件，一切都是不可预料的，做为杀毒软件，不可能接管系统所有的操作，也就是说，如果被病毒运行，等于防线已经崩溃了，系统已经被破坏了。而且，接管系统的各种操作，在系统中没有病毒的情况下，也会给系统本身带来很大的性能负担和运行风险。可能会使系统性能降低一半甚至更多，可能会使系统经常性的死机、蓝屏。用一个为了防止未知的风险，而付出如此多的代价是划不来的，用户也不愿意接受的。因此，在本软件中，仅最大化的加强执行防护这一个防毒门槛。带给用户安全的同时，又不影响用户使用电脑。

### 3.2.1. 云安全实现方案的引入

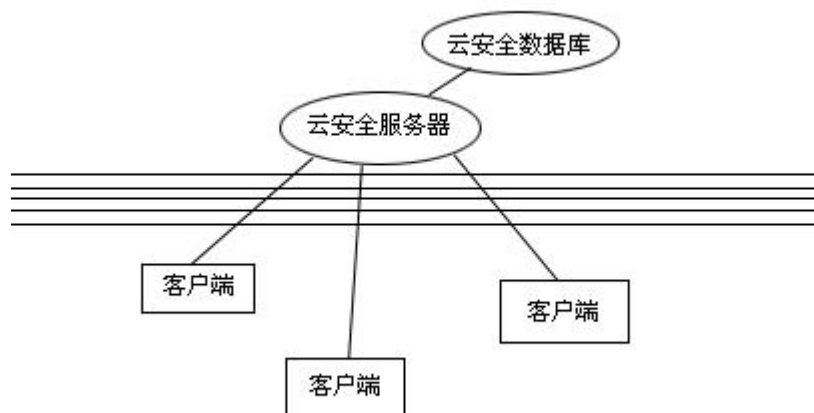
云安全的优势在于：在具有广泛云客户端的基础上，可以及时、实时的获取最新的病毒木马信息，并同时加入到云安全病毒库中。

为了进一步增强防护功能，在本杀毒软件的防毒功能设计中引入了云安全防毒功能。从理论上讲，云安全的实现，最少是需要一台服务器的，用于架设服务器、存储数据。架设服务器，需要云后台程序，存储数据，需要用安装数据库。

本设计中使用的云安全后台使用ASP编写。数据库可以用MS-SQL或Access。

后台程序实现两个功能。其一，接收用户端杀毒软件传送的数据，数据为用户执行程序时获取的文件特征码，这个行为将用户端视为云的客户端，实现了采集数据的功能；其二，根据客户端请求进行查询并反馈结果。请求中

附带的依然是特征码。根据云数据库中的数据，后台程序接收特征码进行校验查询，并反馈查询结果给客户端，这个操作用于防毒功能。即上文所讲的防毒方案中的查询云安全病毒库过程。接收到的数据存储在数据库中，可由维护人员操作，或由云后台根据数据情况按一定规则返回查询结果给客户端软件。



数据量和用户量都较小的时候，可以使用Access，服务器可以使用虚拟主机代替。

介绍了云安全的服务端设置，再来看拦截进程启动的编程实现：

系统在启动一个进程时，在用户层是通过调用API函数:CreateProcess实现的。拦截进程启动行为，我们要做的就是Hook这个API。要实现API Hook, 并在拦截后执行相应的操作，单纯VB是做不到的。这时需要用VC开发一个DLL文件供VB调用。软件运行时，DLL将被注入到所有系统进程中，接管各进程中的CreateProcess函数。这样，无论是哪个程序调用CreateProcess启动程序，DLL都会将暂停这个启动行为，并将被启动的文件信息发给杀毒软件，由杀毒软件判断安全性，根据判断结果决定是否允许程序启动。如果这时检测到是病毒在启动，杀毒软件会告诉DLL：这是病毒，不能让它启动。病毒的启动行为就被阻止了。

来看代码：

DLL部分：

Dll入口函数：

```
BOOL APIENTRY DllMain( HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)
{
    hMod = (HINSTANCE)hModule;
    if (ul_reason_for_call==DLL_PROCESS_ATTACH)
    {
        hProcess=OpenProcess(PROCESS_ALL_ACCESS,0, CurrentProcessId());
    }
}
```

获取我们设计的杀毒软件的标题，以便发送消息进行通讯：

```
hExe = FindWindow(NULL, "杀毒软件Demo");
```

DLL被加载时对createprocess函数进行Hook：

```
HookCreateProcess();
}
```

```
if (ul_reason_for_call==DLL_PROCESS_DETACH)
{
}
```



```
HookStatus(FALSE);
}
return TRUE;
}
```

Hookcreateprocess函数:

```
BOOL HookCreateProcess()
{
    ULONG JmpAddr=0;
```

获取CreateProcessW函数原始地址:

```
FunAddr=(ULONG)GetProcAddress(LoadLibrary("Kernel32.dll"), "CreateProcessW");
```

保存CreateProcessW函数原始地址, 以备恢复hook时使用:

```
memcpy(OldCode, (void *)FunAddr, 5);
```

构造汇编跳转指令:

```
NewCode[0] = 0xe9;
JmpAddr = (ULONG)CreateProcessWCallBack - FunAddr - 5;
```

用我们自定义的CreateProcessWCallBack函数取代CreateProcessW函数的地址, 这样当CreateProcessW系统函数被调用时, 我们自定义的函数可以接管相关的操作:

```
memcpy(&NewCode[1], &JmpAddr, 4);
```

启用API Hook, 此时, 当有启动进程操作时, 由CreateProcessWCallBack进行管理:

```
HookStatus(TRUE);
return TRUE;
}
```

CreateProcessW的替代函数, 拥有与CreateProcessW相同的参数:

```
BOOL WINAPI CreateProcessWCallBack
(
    LPCWSTR lpApplicationName,
    LPWSTR lpCommandLine,
    LPSECURITY_ATTRIBUTES lpProcessAttributes,
    LPSECURITY_ATTRIBUTES lpThreadAttributes,
    BOOL bInheritHandles,
    DWORD dwCreationFlags,
    LPVOID lpEnvironment,
    LPCWSTR lpCurrentDirectory,
    LPSTARTUPINFO lpStartupInfo,
    LPPROCESS_INFORMATION lpProcessInformation
)
{
    BOOL b=FALSE;
```

获取要启动的进程:

```
char AppName[256]={0};
WideCharToMultiByte(CP_ACP, 0, (const unsigned short *)lpApplicationName, -1, AppName, 256, NULL,
NULL);
char* CopyData="";
```

构造字符串，内容是：进程和命令行。用于发送给上层杀毒软件：

```
strcpy(CopyData, AppName);  
strcat(CopyData, "$");  
COPYDATASTRUCT cds={0};  
cds.lpData = CopyData;  
cds.cbData = 1024;
```

取得dll所在进程pid:

```
DWORD dwProcessId;  
dwProcessId=GetCurrentProcessId();
```

```
DWORD ret;
```

向杀毒软件发送消息，消息中包含着正要启动的进程的路径，并等待返回结果：

```
ret=SendMessage(hExe, WM_COPYDATA, dwProcessId, (LPARAM)&cds);
```

如果返回值是915，这个值是自定义的，这个值表明杀毒软件已经检验过被启动文件，并且是安全的，可以启动。

```
if (ret==915)  
{
```

设置拦截状态，暂不启用API Hook。如果此处不做修改，CreateProcessWCallBack会一直有效，形成错误的递归调用，任何程序都启动不了。

```
HookStatus(FALSE);
```

启动进程：

```
b = CreateProcessW  
(  
lpApplicationName,  
lpCommandLine,  
lpProcessAttributes,  
lpThreadAttributes,  
bInheritHandles,  
dwCreationFlags,  
lpEnvironment,  
lpCurrentDirectory,  
lpStartupInfo,  
lpProcessInformation  
);
```

启动后立刻启用API Hook，继续监控进程启动行为：

```
HookStatus(TRUE);  
return b;  
}  
else  
{
```

如果返回值不是915，表达在杀毒软件中检测到被启动文件是病毒或是其它恶意文件。直接返回，阻止程序启动。这时软件中也有相应的提示及操作。

```
return FALSE;  
}
```

```
return FALSE;
}
```

设置API Hook启用状态状态:

```
BOOL HookStatus(BOOL Status)
{
    BOOL ret=FALSE;
    if (Status)
    {
```

将CreateProcessW入口地址写为我们自己定义的函数，使API Hook启用:

```
ret = WriteProcessMemory(hProcess, (void *)FunAddr, NewCode, 5, 0);
if (ret) return TRUE;
}
else
{
```

将CreateProcessW入口地址写为原始函数CreateProcessW的地址，API Hook停止

```
ret = WriteProcessMemory(hProcess, (void *)FunAddr, OldCode, 5, 0);
if (ret) return TRUE;
}
return FALSE;
}
```

```
LRESULT CALLBACK HookProc(int nCode, WPARAM wParam, LPARAM lParam)
{
    return(CallNextHookEx(hHook, nCode, wParam, lParam)
);
}
```

这是输出函数，用于在应用程序中调用，关闭API HOOK:

```
BOOL WINAPI ActiveDefenseOFF()
{
    return(UnhookWindowsHookEx(hHook));
}
```

这是输出函数，用于在应用程序中调用，启用API HOOK:

```
BOOL WINAPI ActiveDefenseON()
{
    设置系统钩子，向各活动进程注入此DLL
    hHook = SetWindowsHookEx(WH_GETMESSAGE, (HOOKPROC)HookProc, hMod, 0);
    if (hHook)
    {
        return TRUE;
    }
    else
    {
        return FALSE;
    }
}
```

在此DLL中，使用SetWindowsHookEx函数向所有进程注入此DLL，DLL初注入后，接管各进程中的CreateProcessW系统函数，此函数可以控制进程启动操作，当检测到进程启动后，暂停启动行为，获取进程的完整路径，并传递给上层执行文件，也就是我们的杀毒软件主程序。传递到主程序后，主程序中对传来路径的文件进行扫描，此时DLL中还在等待上层主程序的处理结果，当在主程序中扫描完成后，反馈信息给DLL：如果是病毒，会中止此进程的启动行为，否则让程序完成启动。这样就完成了对病毒的主动防御。

在主程序中，为了与DLL呼应，完成相应的功能，需要两方面的操作，一方面接管窗口消息，以实现DLL发送消息的响应；另一方面，完成文件扫描并向DLL回应消息。

编码如下：

用SetWindowLong函数，将把程序的消息权交给SubClassMessage函数，这样就接管了程序的消息管理：

```
SetWindowLong hWnd, -4, AddressOf SubClassMessage
```

这样我们就可以在SubClassMessage函数中获取DLL发来的数据。如下：

```
Public Function SubClassMessage(ByVal lHwnd As Long, ByVal lMessage As Long, ByVal lParentPID As Long,
ByVal lParam As Long) As Long
```

```
...
Dim sTemp As String
Dim uCDS As COPYDATASTRUCT
```

判断是否是DLL传来的消息，上文中DLL发送消息时的参数WM\_COPYDATA值等于&H4A

```
If lMessage = &H4A Then
```

```
CopyMemory uCDS, ByVal lParam, Len(uCDS)
sTemp = Space(uCDS.cbData)
CopyMemory ByVal sTemp, ByVal uCDS.lpData, uCDS.cbData
```

获取拦截到的正要启动的程序

```
Dim sFile As String
sFile = Left(sTemp, InStr(1, sTemp, "$") - 1)
sTemp = Right(sTemp, Len(sTemp) - InStr(1, sTemp, "$"))
```

```
end if
End function
```

获取到要启动的程序后，便可使上前面介绍过的扫描病毒代码对文件进行检测。并根据扫描结果向DLL返回消息。编码如下：

```
Dim sScanResult As String
sScanResult = ScanFile(sFile)
If sScanResult = "" then
SubClassMessage =915
else
SubClassMessage = 0
End if
```

同样是调用前面讲过的ScanFile函数对文件进行扫描，如果返回为空说明没有检测到病毒，文件是安全的，则给SubClassMessage赋值915。这句代码代码的意思是：SubClassMessage函数是主程序负责消息处理的函数，DLL中将消息发送到主程序后，将由此函数负责处理，此函数获取赋值后，相当于是函数的返回值，也就是DLL中使用

SendMessage函数发送消息后要等待的消息返回结果。这里的赋值，会直接传递给DLL。而915这个数字我们在前面已经提过，是自定义的一个数值，如果DLL中接收到此值，表明文件没有扫描到病毒，文件是安全的。反之返回0的含意与此类似，只是功能相反。

这是防毒功能中使用传统特征码扫描识别的实现，我们前面讲过要在防毒中使用云安全技术。接下来进行相关介绍。

当本地病毒库特征码校验没有扫描到病毒后，此时可以使用云安全扫描，进行二次检测。编码如下：

```
sCloudAskRet=CloudMatch("http://xxx.com/ask.asp?Hash=" & HashFile(sFile))
```

我们约定：如果返回值不为空说明是病毒：

```
If sCloudAskRet <> "" Then  
Dim sCloudName As String
```

此处是获取返回值中对病毒文件描述，返回的数据中用“\$”符号做为判断分隔符：

```
sCloudName = Right(sCloudAskRet, Len(sCloudAskRet) - InStr(1, sCloudAskRet, "$"))
```

用与上面同样的方法向DLL回应消息：

```
SubClassMessage = 0  
End If
```

这里需要说明的是，使用云安全检测，同样是需要提取文件的特征码，并将特征码发送给云安全服务器。在这里我们使用不同于病毒库中的特征码格式，云安全中将文件整体的哈希值做为特征码。这样做是为了存储和查询的方便。上面代码中HashFile函数，用于获取文件的云安全特征码。编码如下：

```
Function HashFile(ByVal sFilename As String) As String
```

```
HashFile = ""  
Dim lCtx As Long  
Dim lHash As Long  
Dim lFile As Long  
Dim lRes As Long  
Dim lLen As Long  
Dim lIdx As Long  
Dim bHash() As Byte
```

```
If Len(Dir(sFilename)) = 0 Then  
Exit Function  
End If
```

```
lRes=CryptAcquireContext(lCtx,vbNullString,vbNullString, PROV_RSA_FULL, 0)  
If lRes <> 0 Then  
lRes = CryptCreateHash(lCtx, ALGORITHM, 0, 0, lHash)  
If lRes <> 0 Then  
lFile = FreeFile  
Open sFilename For Binary As #lFile  
Const BLOCK_SIZE As Long = 32 * 1024  
ReDim bBlock(1 To BLOCK_SIZE) As Byte  
Dim lCount As Long  
Dim lBlocks As Long  
Dim lLastBlock As Long
```

```

lBlocks = LOF(lFile) \ BLOCK_SIZE
lLastBlock = LOF(lFile) - lBlocks * BLOCK_SIZE
For lCount = 1 To lBlocks
    Get lFile, , bBlock
    lRes = CryptHashData(lHash, bBlock(1), BLOCK_SIZE, 0)
Next
If lLastBlock > 0 And lRes <> 0 Then
    ReDim bBlock(1 To lLastBlock) As Byte
    Get lFile, , bBlock
    lRes = CryptHashData(lHash, bBlock(1), lLastBlock, 0)
End If
Close lFile
If lRes <> 0 Then
    lRes = CryptGetHashParam(lHash, HP_HASHSIZE, lLen, 4, 0)
    If lRes <> 0 Then
        ReDim bHash(0 To lLen - 1)
        lRes = CryptGetHashParam(lHash, HP_HASHVAL, bHash(0), lLen, 0)
        If lRes <> 0 Then
            For lIdx = 0 To UBound(bHash)
                HashFile = HashFile & Right("0" & Hex(bHash(lIdx)), 2)
            Next
        End If
    End If
End If
CryptDestroyHash lHash
End If
End If
CryptReleaseContext lCtx, 0
End Function

```

此函数跟前面讲过的HashFileStream函数代码非常接近，上文中已对HashFileStream函数做过详细讲解，在此不再赘述。

上面的代码中获取了文件的云安全特征码后，使用CloudMatch函数进行云安全校验。CloudMatch函数中将连接云安全服务器，向云后台传递云安全特征码，通过调用云安全后台程序判断文件是否是病毒，编码代码如下：

```

Public Function CloudMatch(sUrl As String) As String

    Dim lInternetOpenUrl As Long
    Dim lInternetOpen As Long
    Dim sTemp As String * 1024
    Dim lInternetReadFile As Long
    Dim lSize As Long
    Dim sContent As String
    sContent = vbNullString

    lInternetOpen = InternetOpen("Cloud", 1, vbNullString, vbNullString, 0)

    If lInternetOpen Then

```

连接云后台，sUrl变量中存储的是云后台程序的网路地址：

```
lInternetOpenUrl = InternetOpenUrl(lInternetOpen, sUrl, vbNullString, 0, INTERNET_FLAG_NO_CACHE_WRITE, 0)
```

```
If lInternetOpenUrl Then
Do
```

读取云后台反馈的内容:

```
lInternetReadFile = InternetReadFile(lInternetOpenUrl, sTemp, 1024, lSize)
sContent = sContent & Mid(sTemp, 1, lSize)
Loop While (lSize <> 0)
End If
```

如果云安全后台检测到病毒，返回的内容格式为：Cloud\$(云安全标识)+病毒名称/病毒描述

```
If UCase(Left(sContent, 6)) = UCase("Cloud$") Then
    CloudMatch = Right(sContent, Len(sContent) - 6)
Else
    CloudMatch = ""
End If
End Function
```

### 3.2.2. 云安全服务端编程

上面的代码中执行云安全查询时，是这样调用的：“sCloudAskRet = CloudMatch("http://xxx.com/ask.asp?Hash=" & HashFile(sFile))”

不难看出，“http://xxx.com/ask.asp”此地址便是云安全的后台程序，在本例中，后台程序由asp写成，负责处理杀毒软件发起的查询，并返回云安全扫描结果。编码如下：

```
<%
取得杀毒软件中传递的文件特征码：
```

```
dim sHash
sHash=trim(request("Hash"))
```

```
dim conn
dim connstr
dim rs
dim sql
```

连接数据库，云安全的特征码存储在数据库，数据库可以使用MsSql、MySQL等各种数据库，甚至是Access，为了演示编码方便，这里我们采用最简单的Access数据库：

```
set conn=server.createobject("ADODB.CONNECTION")
connstr="DBQ="+server.mappath("cloud.mdb")+";DefaultDir=;DRIVER={Microsoft Access Driver (*.mdb)};"
conn.open connstr
set rs=createobject("adodb.recordset")
```

进行SQL查询，病毒库中是否有传特来的特征码：

```
sql="select * from Cloud where Hash=' " & sHash & "' "
rs.open sql,conn,1,3

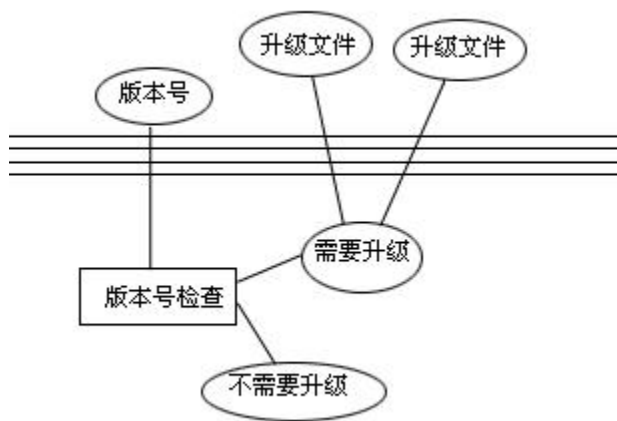
if rs.recordcount=1 then
response.write "Cloud$云安全检测到病毒"
end if

rs.close
set rs=nothing
set conn=nothing
%>
```

到此，杀毒和防毒两个最核心功能的基本讲解已完成。也许有读者会疑惑、会意犹未尽:为什么这么简单？是的，确实很简单，到此，本书使用较少量的代码已经完成了这两个最重要功能的介绍和编码实现演示，之所以使用能由如此少量的代码完成，原因在于省略掉了其它部分代码量非常大的会影响功能实现直观性的编码，只保留了相关的核心代码。这是一种模块化的功能演示编码，也只有这样，才能明了的完成这两个功能的讲解。其它相关部分，如黑白名单的使用、设置选项的关联、与界面的互交代码等等会单独进行讲解。本书后面章节中讲述其它功能时，也会采用类似的方法。

### 3.3. 升级实施方案及编码实现

杀毒软件需要及时升级病毒库，更新特征码，以保证能够识别最新爆发的病毒。为了完成这个操作，需要用到升级服务器（或者虚拟主机也可以满足基本需要）。将要升级的文件，连同新的版本号文件放置在服务器固定位置上。软件中以IP地址或域名的方式访问这个版本号文件，获取最新的版本号，然后跟保存在用户端本地的版本号进行对比。如果检测到服务器上的版本号更高，表明有新的文件需要升级。



升级其实是一个很简单的功能，主要实现的是文件下载功能，为了能够在程序中进行版本号判断以及替换正在执行的文件等操作，我们将此功能分为两个部分，一部分写在应用程序中，一部分写做一个单独的模块控件。模块中的编码如下：

主动事件，用于向用户显示下载进度：

```
Public Event DownloadProcess(lProgress As Long)
```



主动事件，下载完文件后触发此函数：

```
Public Event DownloadFinish()
```

主动事件，错误信息输出函数：

```
Public Event ErrorMessage(sDescription As String)
```

控件初始化函数：

```
Public Function Init() As Boolean  
bConnect = True
```

该函数是第一个由应用程序调用的 WinINET 函数。它告诉 Internet DLL 初始化内部数据结构并准备接收应用程序之后的其他调用。当应用程序结束使用Internet函数时，应调用 InternetCloseHandle 函数来释放与之相关的资源：

```
lOpen=InternetOpen(scUserAgent, INTERNET_OPEN_TYPE_PRECONFIG, vbNullString, vbNullString, 0)
```

通过一个完整的FTP、Gopher或HTTP网址打开一个资源：

```
lFile=InternetOpenUrl(lOpen, sUrl, vbNullString, 0, INTERNET_FLAG_RELOAD, 0)  
sBuffer = Space(1024)  
lBuffer = 1024
```

从服务器获取HTTP请求头信息：

```
bRetQueryInfo = HttpQueryInfo(lFile, 21, sBuffer, lBuffer, 0)  
sBuffer = Mid(sBuffer, 1, lBuffer)  
End Function
```

开始下载文件函数：

```
Public Function StartUpdate() As Boolean
```

```
Dim bBuffer(1 To 1024) As Byte  
Dim lRet As Long  
Dim lDownloadSize As Long  
Dim i As Long
```

```
Dim lFileNumber As Long  
lFileNumber = FreeFile()
```

下载保存文件方式:下载保存为file.exe.bak下载完成后删除file.exe，再重命名 file.exe.bak 为file.exe，这是为了防止file.exe在下载时出现网络错误文件只被写入一部分数据而引起错误：

```
Dim sTempDownloadFile As String  
sTempDownloadFile = Trim(sSaveFile) & ".bak"  
If Dir(sTempDownloadFile) <> "" Then  
Call DeleteFile(sTempDownloadFile)  
End If
```

打开文件，保存下载数据：

```
Open sTempDownloadFile For Binary Access Write As #lFileNumber  
Do
```

读取被下载文件内容:

```
InternetReadFile lFile, bBuffer(1), 1024, lRet  
If lRet = 1024 Then
```

将读取到的数据写入文件:

```
Put #1, , bBuffer  
Else  
For i = 1 To lRet  
Put #1, , bBuffer(i)  
doEvents  
Next i  
end If  
lDownloadSize = lDownloadSize + lRet
```

引发下载进度事件, 这是为了在下载文件时让用户了解下载进展情况, 防止用户以为程序假死:

```
RaiseEvent DownloadProcess(lDownloadSize)
```

```
Loop Until lRet < 1024  
Close #lFileNumber
```

检查已下载字节数与要下载文件大小是否一至, 一至说明下载完成了:

```
If lDownloadSize = CLng(GetHeader("Content-Length")) Then
```

下载完成后, 删除原文件, 重命名下载文件为原文件名:

```
If Dir(Trim(sSaveFile)) <> "" Then  
Call DeleteFile(sSaveFile)  
End If  
Sleep 50  
If Dir(sSaveFile) = "" Then
```

重命名下载的文件:

```
Name sTempDownloadFile As sSaveFile  
End If  
DoEvents  
End If
```

```
End Function
```

获取要下载文件的信息:

```
Public Function GetHeader(Optional hdrName As String) As String
```

```
Dim sTemp As Long  
Dim sTemp2 As String  
Select Case UCase(hdrName)
```

获取要下载的文件大小：

```
Case "CONTENT-LENGTH"
sTemp = InStr(sBuffer, "Content-Length")
sTemp2 = Mid(sBuffer, sTemp + 16, Len(sBuffer))
sTemp = InStr(sTemp2, Chr(0))
GetHeader = CStr(Mid(sTemp2, 1, sTemp - 1))
End Select

End Function
```

设置属性，即要下载的文件的网络地址：

```
Public Property Let URL(ByVal sInUrl As String)
    sUrl = sInUrl
End Property
```

以上是控件中的编码，之所以使用控件，是为了能够使用主动事件，方便的与界面互交，比如：即时的显示下载进度，使用户能看到下载进行了多少、下载完成后能做出相应的消息提示。

在程序更新病毒库或其它程序文件时，首先会进行如下的方式进行调用，获取网络服务器文件中的版本号，如果版本号高于本地版本号，则进行更新，如果与本地版本号一至，则说明本地已经是最新版本，无需更新：

```
Dim sUpdateFiles As String
sUpdateFiles=DetectUpdateFile("http://xxx.cn/update/update.txt")
```

DetectUpdateFile函数的代码如下：

```
Public Function DetectUpdateFile(sUrl As String) As String

    Dim lInternetOpenUrl As Long
    Dim lInternetOpen As Long
    Dim sTemp As String * 1024
    Dim lInternetReadFile As Long
    Dim lSize As Long
    Dim sContent As String
    sContent = vbNullString
    lInternetOpen = InternetOpen("update", 1, vbNullString, vbNullString, 0)
    lInternetOpenUrl=InternetOpenUrl(lInternetOpen, sUrl, vbNullString, 0,
INTERNET_FLAG_NO_CACHE_WRITE, 0)
    Do
        lInternetReadFile = InternetReadFile(lInternetOpenUrl, sTemp, 1024, lSize)
        sContent = sContent & Mid(sTemp, 1, lSize)
    Loop While (lSize <> 0)
    lInternetReadFile = InternetCloseHandle(lInternetOpenUrl)

    升级查询返回以“Update”开始为标识
    If UCase(Left(sContent, 6)) = UCase("Update") Then
        DetectUpdateFile = Right(sContent, Len(sContent) - 6)
    Else
        DetectUpdateFile = ""
    End If
End Function
```

```
End If
```

```
End Function
```

这部分代码与上面文件下载控件中的代码非常类似，都是用来下载文件的，不同在于此处下载文件后会读取文件内容。

文件内容是我们预设好的，本地设置文件中有版本号，升级时，从服务器下载版本号文件并从中读取内容获取最新整体版本号，与当前软件中的整体版本号进行比对，如果网络中的版本号更高，则进行更新，调用升级控件下载升级文件。每次升级把设置文件同时也下载，以同步版本号。

我们设定网络中版本号文件格式为：“Update”(6位，用于标识是升级校验文件)+版本号(6位)+待升级文件+“\$”(结束符)

获取到文件信息后，然后对比版本号，如果服务器中的版本号更高，则表示有文件需要下载更新，那么我们就需要下载文件。编码如下：

取得服务器文件中的版本号：

```
Dim lNewWholeVersion As Long
lNewWholeVersion = Left(sUpdateFiles, 6)
sUpdateFiles = Right(sUpdateFiles, Len(sUpdateFiles) - 6)
```

取得本地版本号，在这里Version.ini文件中存放着本地版本号：

```
Dim sVersionFile As String
sVersionFile = App.Path & "\Version.ini"
Dim lOldWholeVersion As Long
lOldWholeVersion = ReadIni(sVersionFile, "Version", "WholeVersion")
```

判断是否需要升级：

```
If lNewWholeVersion <= lOldWholeVersion Then
MsgBox "已是最新版本，无需升级。"
End If
```

如果有待更新内容，从返回内容中获取要升级的文件并下载：

```
Do While Not sUpdateFiles = ""
sNewFile = Left(sUpdateFiles, InStr(1, sUpdateFiles, "$") - 1)
sUpdateFiles = Right(sUpdateFiles, Len(sUpdateFiles) - InStr(1, sUpdateFiles, "$"))
```

如果是要下载exe文件，先用MoveFileEx函数去掉它的执行保护，这样就不会出现替换正在执行的文件而产生错误的问题：

```
If InStr(1, LCase(sNewFile), LCase(".exe")) Then
Call MoveFileEx(sNewFile, RndChr(6), 1)
End If
```

使用上面写好的升级控件下载文件，假设控件名为UserControlUpdate1：

```
With UserControlUpdate1
```

构造文件下载地址:

```
.URL = "http://xxx.cn/update/" & sNewFile  
.SaveFile = App.Path & "\" & sNewFile  
If .Init = True Then  
lCurrentFileSize = .GetHeader("Content-Length")
```

开始下载文件:

```
.StartUpdate  
End If  
End With
```

```
Loop
```

以上是升级功能的思现思路及编码方法,在实际软件的编码中,除了实现文件下载的功能,还需要考虑其它辅助因素,比如在下地时,要将文件显示在界面中,告诉用户正在下载哪个文件,以及显示下载进度,以给用户一个更为直观和亲切的使用感受。

### 3.4. 自我保护设计方案及编码实现

实现此功能,同样使用API Hook技术,与上面介绍过的病毒防护功能类似。只是这里不再是拦截CreateProcess,而是要接管OpenProcess函数。

也许有的程序员朋友会有疑惑,做我我保护,也就是要防止自己的进程被非法结束进程,在编程中,结束进程所使用API函数是TerminateProcess,而不是OpenProcess,这里为什么会说通过拦截OpenProcess实现保护进程呢?这是由于TerminateProcess函数结束进程时,需要使用进程句柄做为输入参数,结束传入的指定句柄的进程。但在不同进程间句柄是无法传递的,因为即便是同一个文件,在每个不同的进程中都会为它分配不同的句柄。好在进程ID是可以传递的,而在使用TerminateProcess函数结束进程前,会先调用OpenProcess打开进程并把句柄传递给TerminateProcess,而OpenProcess函数的输入参数恰好为进程ID。这也就是我们选择拦截OpenProcess的原因了。

防止被结束进程的原理很简单,将DLL注入其它进程后,接管进程的OpenProcess函数,在我们自己定义的新OpenProcess函数被调用时,判断是否要打开我们杀毒软件的进程,并且其操作是否是要将要进程结束进程操作,如果是,就终止这个行为。这样便实现了保护,做到了进程防杀。

且看编码实现,首先是DLL中拦截OpenProcess的部分。鉴于上文中又详细介绍过拦截CreateProcess函数的方法,这里就不再重复对代码做说明了,参考上文代码中的注释说明即可。

```
#include <windows.h>  
#include "stdio.h"  
#include "stdlib.h"  
  
#pragma data_seg(".Shared2")  
static DWORD g_PID2Protect=0;  
#pragma data_seg()  
#pragma comment(linker, "/section:.Shared2,rws")
```

```

HANDLE hProcess=0;
UCHAR OldCode2[5]={0}, NewCode2[5]={0};
ULONG FunAddr2=0;
HINSTANCE hMod=0;
HHOOK hHook=0;

BOOL HookStatus2(BOOL Status){
    BOOL ret2=FALSE;
    if (Status) {
        ret2 = WriteProcessMemory(hProcess, (void *)FunAddr2, NewCode2, 5, 0);
        if (ret2) return TRUE;}
    else {
        ret2 = WriteProcessMemory(hProcess, (void *)FunAddr2, OldCode2, 5, 0);
        if (ret2) return TRUE;}
    return FALSE;
}

HANDLE WINAPI OpenProcessCallBack(DWORD dwDesiredAccess, BOOL bInheritHandle, DWORD dwProcessId)
{
    HANDLE hProc=NULL;
    if(dwProcessId==g_PID2Protect && (dwDesiredAccess & PROCESS_TERMINATE!=0))
    {
        hProc=NULL;
    }else{
        HookStatus2(FALSE);
        hProc= OpenProcess(dwDesiredAccess,bInheritHandle,dwProcessId);
        HookStatus2(TRUE);
    }
    return hProc;
}

BOOL HookOpenProcess() {

    ULONG JumpAddr2=0;
    FunAddr2 = (ULONG)GetProcAddress(LoadLibrary("Kernel32.dll"), "OpenProcess");
    memcpy(OldCode2, (void *)FunAddr2, 5);
    NewCode2[0] = 0xe9;
    JumpAddr2 = (ULONG)OpenProcessCallBack - FunAddr2 - 5;
    memcpy(&NewCode2[1], &JumpAddr2, 4);
    HookStatus2(TRUE);
    return TRUE;
}

BOOL APIENTRY DllMain( HANDLE hModule,DWORD ul_reason_for_call, LPVOID lpReserved)
{
    hMod = (HINSTANCE)hModule;
    if (ul_reason_for_call==DLL_PROCESS_ATTACH)
    {
        hProcess = OpenProcess(PROCESS_ALL_ACCESS, 0, GetCurrentProcessId());
        HookOpenProcess();
    }
}

```

```

if (ul_reason_for_call==DLL_PROCESS_DETACH)
{
HookStatus2(FALSE);
}

return TRUE;
}

LRESULT CALLBACK HookProc(int nCode, WPARAM wParam, LPARAM lParam) {
return(CallNextHookEx(hHook, nCode, wParam, lParam));
}

BOOL WINAPI SafeGuardOFF() {
return(UnhookWindowsHookEx(hHook));
}

BOOL WINAPI SafeGuardON(DWORD uPID) {

g_PID2Protect=uPID;

hHook = SetWindowsHookEx(WH_GETMESSAGE, (HOOKPROC)HookProc, hMod, 0);
if (hHook)
{
return TRUE;
}else{
return FALSE;
}
}
}

```

由代码中可知，此DLL的导出函数有两个：SafeGuardON和SafeGuardOFF，SafeGuardON用于开启防护，参数是被保护进程的ID；SafeGuardOFF用于关闭防护。

在上层应用程序中，调用方法如下：

**函数声明：**

```

Private Declare Function ActiveDefenseON Lib "ActiveDefense.dll" (ByVal TargetHwnd As Long) As Boolean
Private Declare Function ActiveDefenseOFF Lib "ActiveDefense.dll" () As Boolean

```

**DLL中的函数再次封装为新函数：**

```

Public Function ActiveDefense(ByVal bOperation As Boolean) As Boolean

Dim lPid As Long

```

**获取当前进程PID，也就是我们杀毒软件的进程ID：**

```

lPid = GetCurrentProcessId

Dim bRet As Boolean

```

```
If bOperation = True Then
```

开启主动防御:

```
bRet = ActiveDefenseON(lPid)
If bRet Then
ActiveDefense = True
Else
ActiveDefense = False
End If
Else
```

关闭主动防御:

```
ActiveDefenseOFF
End If

End Function
```

有此简单封装过的函数，当杀毒软件启动时调用，传入参数true便可开启防护，关闭程序时调用传入false便可停止防护。使用非常简单。

### 3.5. 黑白名单设计方案及编码实现

黑白名单功能做为病毒库的辅助，主要用于病毒防护、减少误报、阻止非病毒类的边缘灰色文件。

在查杀病毒时，并不使用黑白名单，而在防毒的主动防御中我们使用它。前面我们已经讲过：当进程要启动时，会先使用病毒病毒库以特征码匹配的方法对其进行扫描，如果没有检测到病毒，会次使用云安全进行识别。黑白名单的引入，可以更进一步提升防毒效果，增强整体软件的防护功能。

我们进行如此设计：黑名白名数据文件，在本地存储时使用ini文件格式。ini文件内容由节和参数组成，可以方便进行少量数据的存储、修改、添加、删除。

黑白名单文件格式使用如下例：

```
[Count]
MaxID=0
[File]
00001=C:\WINDOWS\system32\cmd.exe
[Signature]
00001=83BA7E22BF529858A345F483D7E94C16
[Description]
00001=安全的文件
[DefenseFlag]
00001=1
```

Count节中，MaxID参数指明共有多少条数据。

File节中，序号对应的是文件。

Signature节中，序号对应的的是文件的特征码。



Description节中，序号对应的是对文件的描述。

DefenseFlag节中，序号对应的是处理规则。数字1代表安全的文件，执行放行操作；数字0代表是恶意文件，执行阻止操作。

程序编码时，相关操作代码置于上文中介绍过的SubClassMessage函数中，当获取到DLL专来的启动程序信息后，在黑白名单文件中进行匹配校验，如果检测到是安全的文件，则直接放行；如果检测到是恶意文件，直接阻止。且看代码：

获取黑白名单文件名,假设黑白名单文件名为ActiveDefense.ini

```
Dim sDefenseIniFile As String
sDefenseIniFile = App.Path & "\ActiveDefense.ini"
End If
```

读取黑白名单文件中的记录数据量

```
Dim lMaxID As Long
lMaxID = ReadIni(sDefenseIniFile, "Count", "MaxID")
```

规则对应文件

```
Dim sDefenseRuleFile As String
```

规则对应文件特征码

```
Dim sDefenseRuleFileSignature As String
```

规则标志，1为放行，0为禁止

```
Dim sDefenseRuleFileFlag As String
```

循环匹配黑白名单中的所有数据

```
Dim i As Long
For i = 1 To lMaxID
```

规则文件

```
sDefenseRuleFile = ReadIni(sDefenseIniFile, "File", Format(i, "00000"))
```

文件特征码

```
sDefenseRuleFileSignature = ReadIni(sDefenseIniFile, "Signature", Format(i, "00000"))
```

处理标志

```
sDefenseRuleFileFlag = ReadIni(sDefenseIniFile, "DefenseFlag", Format(i, "00000"))
```

这里假设sFile变量中保存的是DLL中传来的要启动的文件，获取文件的特征码，然后跟名单中所有记录进行匹配

```
If (sDefenseRuleFileSignature = HashFile(sFile)) And (Len(sDefenseRuleFileSignature) = 32) And
(Len(HashFile(sFile)) = 32) Then
```

```
If CLng(sDefenseRuleFileFlag) = 1 Then
```

标志为1，执行放行操作。SubClassMessage函数在上面介绍过，是用来接收DLL消息的函数。给它赋值915，也就是令SubClassMessage函数的返回值为915。在讲前面讲的DLL编程中，DLL中使用“ret=SendMessage(hExe, WM\_COPYDATA, dwProcessId, (LPARAM)&cds)”向程序发来了启动文件信息，同时在等待消息的返回。如果返回915，DLL中就会放行这个试图启动的文件。

```
SubClassMessage = 915
Exit Function
ElseIf CLng(sDefenseRuleFileFlag) = 0 Then
```

标志为0，执行阻止操作。与返回915的行为刚好相反。

```
SubClassMessage = 0
Exit Function
End If
End If
Next
```

代码中用到的HashFile函数即是上文中进行云安全扫描前用于获取文件特征码的函数。在黑白名单中使用的特征码，与云安全使用相同的特征码。

### 3.6. 软件设置实现方案

软件设置听起来貌似是一个杂项，似乎并不重要。但对用户而言，合理而便捷的设置会使软件操作更加舒适。而且在实际的软件应用中，各种设置功能会贯穿于编码的各个部分。上文中介绍各种功能时，为了使代码逻辑条理、便于理解，才没有考虑各种外部设置代来的影响，在编码中也没有引入设置功能。如果我们要完成一个功能完善的软件，是必须要时时刻刻考虑到设置选项功能的。

比如在软件启动时，要判断是正常启动，还是由右键扫描菜单引发的启动，如果是后者，则要执行不同的代码，进接进入病毒扫描环节；

比如在启动软件后，要判断设置选项中是否开启着片我保护，以决定是否要向其它进程中注入具有防杀功能的DLL文件；

比如在扫描文件时，要从设置选项中判断是否只是对可执行文件进行扫描；

比如在软件执行防护功能时，要从设置项目中判断是否开启了云安全防护功能，以决定是否要连接云安全服务器；

比如在关闭软件时，要从设置项中判断是要直接退出程序，还是仅仅要转为后台运行状态。等等。

编码时，要在庞大的代码中多次重复引入这些设置选项，因此不得不说合理的设置功能的设计对于此软件工程而言是不可轻视。

各种设置选项以数值的形式保存在配置ini文件中，以方便重复的读取和写入。Ini文件格式的字段和项属性非常便于准确定位和操作。因此，在本程序中被多次使用，前面介绍的黑白名单设计中已经使用，后面会讲到的右键扫描功能的实现中也将要使用。

假设设置文件Settings.ini内容如下：

```
[Normal]
SafeGuard=1
AutoRun=1
```

对应的读取此文件的编码为：

```
Dim sSettingsFile As String
sSettingsFile = App.Path & "\Settings.ini"
```

写入自我保护设置信息，CheckSafeGuard是界面中的复选框控件名

```
Call WriteIni(sSettingsFile, "Normal", "SafeGuard", CheckSafeGuard.value)
```

写入软件开机自启动设置信息，CheckAutoRun界面中的复选框控件名

```
Call WriteIni(sSettingsFile, "Normal", "AutoRun", CheckAutoRun.value)
```

WriteIni函数是经简单封装使用使用的ini操作函数，如下所示：

```
Public Function WriteIni(ByVal sFile As String, ByVal sSection As String, ByVal sKeyName As String,
ByVal sKeyValue As String) As Boolean
Dim lRet As Long
```

调用API函数WritePrivateProfileString进行写操作：

```
lRet = WritePrivateProfileString(sSection, sKeyName, sKeyValue, sFile)
If lRet = 0 Then
WriteIni = False
Exit Function
Else
WriteIni = True
Exit Function
End If
End Function
```

进行读取操作时：

读取自我保护设置信息，CheckSafeGuard是界面中的复选框控件名：

```
CheckSafeGuard.value = ReadIni(sSettingsFile, "Normal", "SafeGuard")
```

读取软件开机自启动设置信息，CheckAutoRun界面中的复选框控件名：

```
CheckAutoRun.value = ReadIni(sSettingsFile, "Normal", "AutoRun")
```

使用ReadIni函数获取配置信息，直接赋值给界面中的控件。

ReadIni函数与WriteIni类似，如下：

```
Public Function ReadIni(ByVal sFile As String, ByVal sSection As String, ByVal sKeyName As String)
As String

Dim lRet As Long
Dim sTemp As String * 255
lRet = GetPrivateProfileString(sSection, sKeyName, "", sTemp, 255, sFile)
If lRet = 0 Then
ReadIni = ""
Exit Function
Else
ReadIni = TrimNull(sTemp)
End If
End Function
```

### 3.7. 杀毒软件界面设计

从某种角色而言，软件界面比功能更为重要，当然这是针对用户的。界面是软件的门面，会带给用户第一使用感受，如果软件界面做的很糟糕，用户用过一次便不想再看，那么即便是软件功能再强大，用户也不会想使用。

在当下流行的各种开发工具中，都会提供各种功能控件，但这些控件风格都比较陈旧，按钮永远都是一个单色的小方块、文本框永远都有丑陋的下陷边框等等。当然是有第三方的界面控件可以使用，但其对开发环境的要求往往比较苛刻、会增加一定额外的代码量、控件内部功能对开发者不透明、会引起软件意外的错误、不能完全满足开发时随心所欲的需求等等。那么在这一章，我们将解决这一问题，介绍一种软件界面设计方面的技巧和相关的编程方法，提供给大家一种全新的界面设计思路。

#### 3.7.1. 漂亮界面的设计技巧：图片替换控件法

说到全新，其实也不能称的上是全新，只是一种技巧，只大多数人通常在不经意见忽视了它，没有发现它的价值。简单的说，就是大量使用图片来取代和装饰原来控件。

比如：去掉界面的边框，使用一张背景图来做为新的界面，在此图的基础上再次放置图片来替代界面固有的最大化、最小化、退出按钮。为了实现按钮在移动和按下的效果，可以做三组不同的图片，分别用来显示按钮在普通状态下的效果、鼠标移动到上面时的效果和鼠标按下后的效果。再在代码中图片感应到的事件函数中对图片做不同的显现控制、位置控件，这样就可以实现完美替换原有按钮的功能，又能达一美化界面的功效。

下面付上一段对界面中关闭按钮进行图片替代的编码方法：

首先在界面中放置三个图片，做了编码方法，使用数组的方式对其进行命名，分别为：ImageExit(0)、ImageExit(1)、ImageExit(2)

ImageExit(0)中贴入按钮普通状态下图片，ImageExit(1)中贴入鼠标移动到上面后的图片，ImageExit(2)中贴入按钮被按下后的状态图片。

然后进行编码：

在窗体加载函数中：

```
Private Sub Form_Load()  
Dim i As Long  
For i = 0 To 2  
    初始化关闭按钮图片位置  
    With ImageExit(i)  
        .Left = 3480  
        .Top = 0  
    End With  
Next
```

设置各图片的可见状态

```
ImageExit(0).Visible = True  
ImageExit(1).Visible = False  
ImageExit(2).Visible = False  
End Sub
```

这样，窗体加载后，使三个图片的位置重叠在一起，而且控制隐藏掉两个图片，只显示一个普通状态下的图片。在用户看起来，就会只是一个按钮。

然后在其它图片事件中，再对按钮的显现做调整，以模拟正常关闭按钮对鼠标的响应：

鼠标在图片上移动时：

```
Private Sub ImageExit_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    ImageExit(0).Visible = False
    ImageExit(1).Visible = True
    ImageExit(2).Visible = False
End Sub
```

鼠标按下时：

```
Private Sub ImageExit_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    ImageExit(0).Visible = False
    ImageExit(1).Visible = False
    ImageExit(2).Visible = True
End Sub
```

鼠标抬起时，执行关闭操作

```
Private Sub ImageExit_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    Unload Me
End Sub
```

由于是文字介绍，这部分功能无法直接的效果显现给读者，需仔细思考加实践方可理解此法之妙、效果之好。

这里只是演示了关闭按钮的图片替代法，其它各种功能控件，也可以使用类似的方法进行美化、替换。

### 3.7.2. 图片替换法的高级应用：界面换肤

因为图片用图片代替了控件，而使用的图片恰恰是可以控制的，那么在此技术的基础上，我们可以实现很多人梦寐以求的高级界面技术：界面换肤。

在图片替换法的基础上，界面换肤的实现非常简单：在上面的介绍中，关闭按钮的图片是事先贴好在图片中的，在此我们只需动态的替换图片中的图片源，即可实现界面换肤。同样以关闭按钮为例来演示：

在窗体加载函数中：

```
Private Sub Form_Load()
    ImageExit(0).Picture = LoadPicture(App.Path & "\Skin_01\Exit0.bmp")
    ImageExit(1).Picture = LoadPicture(App.Path & "\Skin_01\Exit1.bmp")
    ImageExit(2).Picture = LoadPicture(App.Path & "\Skin_01\Exit2.bmp")

    Dim i As Long
    For i = 0 To 2
        初始化关闭按钮图片位置
    Next i
End Sub
```

```

        With ImageExit(i)
            .Left = 3480
            .Top = 0
        End With
    Next

```

设置各图片的可见状态

```

ImageExit(0).Visible = True
ImageExit(1).Visible = False
ImageExit(2).Visible = False
End Sub

```

同样是在窗体加载函数中，只需新增三行代码，不严格的说，只需一行：

```
ImageExit(0).Picture = LoadPicture(App.Path & "\Skin_01\Exit0.bmp")
```

即可实现肤换，其它图片对事件的响应代码不变，依然是使用上面的代码。

这行代码的功能是当窗体加载时，对图片控件中的图片进行加载，在这里是使用了文件：  
App.Path & "\Skin\_01\Exit0.bmp"，也就是软件目录中Skin\_01目录下的Exit0.bmp文件，

要换肤时，只需更改此文件的路径即可，比如使用另一目录下的文件：

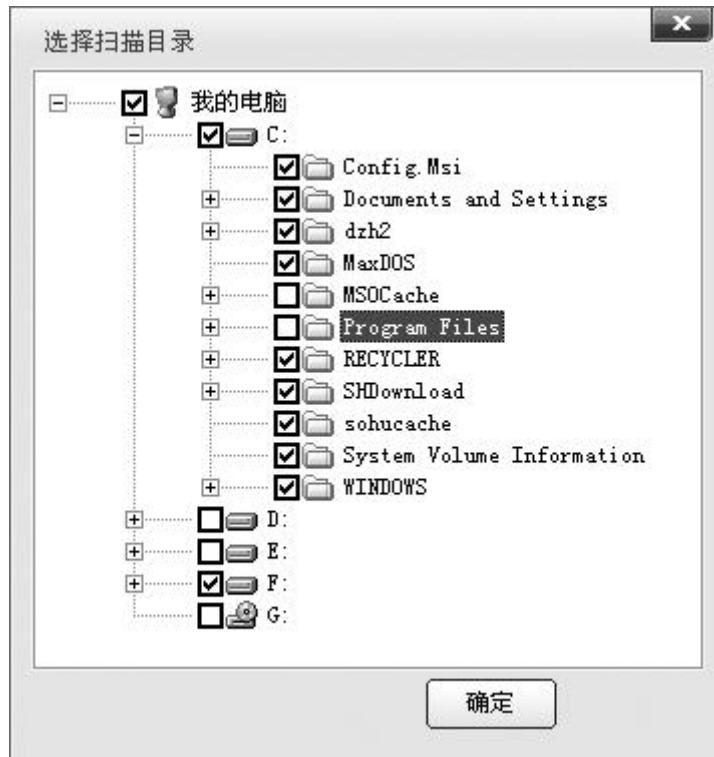
```
App.Path & "\Skin_02\Exit0.bmp"
```

然后显现在图片中的将会是另一张图，对用户而言，将会是另一个软件界面。

### 3.7.3. 扫描自定义目录界面的实现

这个功能是杀毒软件所必备的，进行自定义扫描时，必须提供给用户选择不同驱动器、目录的选择权。实现这个功能，我们使用TreeView树型控件，但TreeView控件原本并没有这个功能，为了达到预期的需求，这里需要使用众多的技巧，这一部分功能代码用于演示使用编程技巧扩充控件原有功能的实现。

我们的预期目标是：在TreeView控件中，显示出计算机上所有的驱动器；在打开不同的驱动器及目录时，能够显示它的下级目录；在选中或取消选中某一节点时，控件能自动选中或取消选中该节点的子节点，但不影响父节点。界面效果如下：



假设已经在窗体中使用了TreeView控件，名称为：TreeViewScanTarget，功能实现代码如下：

```
Private Sub Form_Load()
```

窗体加载时，获得系统所有盘符并添加到控件中进行显示

```
TreeViewScanTarget.Nodes.Add , , "Lord", "我的电脑", 1
```

```
Dim sDrive As String
```

```
sDrive = String(256, Chr(0))
```

```
Dim sDriveID As String
```

```
Call GetLogicalDriveStrings(256, sDrive)
```

```
Dim i As Long
```

```
For i = 1 To 100 Step 4
```

```
sDriveID = Mid(sDrive, i, 3)
```

```
TreeViewScanTarget.Nodes.Add "Lord", tvwChild, sDriveID, Left(sDriveID, 2), 2
```

```
Next i
```

设置根节为选中状态

```
TreeViewScanTarget.Nodes(1).Checked = True
```

设置根节为展开状态

```
TreeViewScanTarget.Nodes(1).Expanded = True
```

设置子节选中状态

```
CheckChildNodes TreeViewScanTarget.Nodes(1)
```

```
CheckParentNodes TreeViewScanTarget.Nodes(1)
```

```
End Sub
```

Treeview控件单击事件处理函数

```
Private Sub TreeViewScanTarget_Click()  
    CheckChildNodes TreeViewScanTarget.SelectedItem  
    CheckParentNodes TreeViewScanTarget.SelectedItem  
End Sub
```

Treeview控件节点单击事件处理函数

```
Private Sub TreeViewScanTarget_NodeClick(ByVal Node As MSComctlLib.Node)
```

转移Treeview中节点的焦点

```
    CheckChildNodes TreeViewScanTarget.SelectedItem  
    CheckParentNodes TreeViewScanTarget.SelectedItem  
End Sub
```

Treeview控件节点展开事件处理函数，功能是：获取节点所对应的子目录，增加子节点

```
Private Sub TreeViewScanTarget_Expand(ByVal Node As MSComctlLib.Node)
```

如果Treeview无内容则退出

```
If TreeViewScanTarget.Nodes.Count = 0 Then Exit Sub
```

```
Dim i As Long
```

如果选中的节点无子节点

```
If Node.Children = 0 Then  
    Call GetSubFolders(TreeViewScanTarget, Node)
```

如果有子节点

```
Else  
    Dim uNode As Node
```

节点的第一子节点

```
Set uNode = TreeViewScanTarget.Nodes(Node.Index).Child
```

遍历所有子节点

```
For i = 1 To Node.Children
```

```
DoEvents
```

如果子节点没有子节点，则取子节点的下级目录，增加子节点

```
If uNode.Children = 0 Then  
    Call GetSubFolders(TreeViewScanTarget, uNode)  
End If  
Set uNode = uNode.Next
```

```
Next
```

```
End If
```

```
End Sub
```



#### 处理子节点选中状态

```
Private Function CheckChildNodes(ByVal Node As MSComctlLib.Node)
    Dim i As Long
    Dim uChildNode As Node

    如果没有子节点，则退出
    If Node.Children = 0 Then
        Exit Function
    End If

    If Node.Checked = True Then

        选中
        Set uChildNode = Node.Child
        For i = 1 To Node.Children
            uChildNode.Checked = True
            If uChildNode.Children <> 0 Then
                CheckChildNodes uChildNode
            End If
            Set uChildNode = uChildNode.Next
        Next

    Else
        Set uChildNode = Node.Child
        For i = 1 To Node.Children
            uChildNode.Checked = False
            If uChildNode.Children <> 0 Then
                CheckChildNodes uChildNode
            End If
            Set uChildNode = uChildNode.Next
        Next
    End If
End Function
```

#### 处理父节点选中状态

```
Private Function CheckParentNodes(ByVal Node As MSComctlLib.Node)
    Dim i As Long
    Dim uParentNode As Node

    同层node个数
    Dim lSameLevelNodes As Long
    Dim uSameLevelNode As Node

    如果没有子节点，则退出
    If Node.Root = Node Then
        Exit Function
    End If

    If Node.Checked = True Then
```

```

Set uParentNode = Node.Parent
uParentNode.Checked = True

Do While uParentNode <> Node.Root
Set uParentNode = uParentNode.Parent
uParentNode.Checked = True
Loop
Else

```

检查是否同级节点都是不选中状态

```

If Node.Parent.Children > 0 Then
lSameLevelNodes = Node.Parent.Children
Set uSameLevelNode = Node.Parent.Child
For i = 1 To lSameLevelNodes
If uSameLevelNode.Checked = True Then
Exit Function
End If
Set uSameLevelNode = uSameLevelNode.Next
Next
End If

```

不选中

```

Set uParentNode = Node.Parent
uParentNode.Checked = False

Do While uParentNode <> Node.Root
Set uParentNode = uParentNode.Parent
uParentNode.Checked = False
Loop
End If
End Function

```

Treeview控件中节点选中及反选事件

```
Private Sub TreeViewScanTarget_NodeCheck(ByVal Node As MSComctlLib.Node)
```

转移Treeview中节点焦点

```
TreeViewScanTarget.SelectedItem = Node
```

处理节点选中状态

```

CheckChildNodes Node
CheckParentNodes Node
End Sub

```

取得目录或驱动器下子目录，参数：uTreeView，放置目录的Treeview控件；uParentNode：Treeview选中节点，包含文件路径，本函数即取该目录下的子目录

```
Public Function GetSubFolders(uTreeView As Control, ByVal uParentNode As Node)
```

```
Dim sPath As String
```

检查最后字母是否是“\”, Treeview以“\”分隔每个节点, 所以: “我的电脑” + “\” 长度为5

```
If Right(uParentNode.FullPath, 1) <> "\" Then
sPath = Right(uParentNode.FullPath, Len(uParentNode.FullPath) - 5) & "\"
Else
sPath = Right(uParentNode.FullPath, Len(uParentNode.FullPath) - 5)
End If
```

```
Dim lRet As Long
Dim uWFD As WIN32_FIND_DATA
```

查找目录下文件

```
lRet = FindFirstFile(sPath & "*. *" & Chr(0), uWFD)
If lRet <> -1 Then
```

不为. 或.. 且是目录

```
If (TrimNull(uWFD.cFileName) <> ".") And (TrimNull(uWFD.cFileName) <> "..") And
(uWFD.dwFileAttributes And vbDirectory) Then
```

增加子节点, 添加目录

```
Call uTreeView.Nodes.Add(uParentNode.Key, tvwChild, sPath & TrimNull(uWFD.cFileName),
TrimNull(uWFD.cFileName), 7)
End If
```

```
While FindNextFile(lRet, uWFD)
```

不为. 或.. 且是目录

```
If (TrimNull(uWFD.cFileName) <> ".") And (TrimNull(uWFD.cFileName) <> "..") And
(uWFD.dwFileAttributes And vbDirectory) Then
```

增加子节点, 添加目录

```
Call uTreeView.Nodes.Add(uParentNode.Key, tvwChild, sPath & TrimNull(uWFD.cFileName),
TrimNull(uWFD.cFileName), 7)
End If
wend
End If
Call FindClose(lRet)
End Function
```

### 3.8. 其它辅助功能设计

之前几章中, 出条理和易懂的目的, 对各内容的介绍是以功能为模块进行分类单独讲述的, 因此有许多本该穿插在其中功能未做到讲解。在本章中将对这些重要的未讲解到的功能再做单独介绍。

#### 3.8.1. 右键扫描杀毒

右键扫描是杀毒软件常见的功能，当下载了软件或从U盘复制了软件，想要进行扫描时，使用右键菜单扫描非常便捷。

我们把右键扫描功能分解成两个部分：右键菜单的关联，也就是如果让用户在文件或文件夹上点击右键时弹出的右键菜单中能调用我们的杀毒软件；其次是如何能在右键菜单中调用杀毒软件进行扫描。

且看第一部分右键菜单关联相关代码：

在文件的右键菜单中增加杀毒软件菜单项：

```
Public Function AddFileRightClickMenu() As Boolean
Dim hKey As Long, ret As Long
RegCreateKey HKEY_CLASSES_ROOT, "*\shell\杀毒软件\command", hKey
ret = RegSetValueEx(hKey, "", 0, REG_SZ, ByVal App.Path & "\" & App.EXENAME & ".exe %1", ByVal
LenB(StrConv(App.Path & "\" & App.EXENAME & ".exe %1", vbFromUnicode)) + 1)
RegCloseKey hKey
End Function
```

在文件夹的右键菜单中增加杀毒软件菜单项：

```
Public Function AddFolderRightClickMenu() As Boolean
Dim hKey As Long, ret As Long
RegCreateKey HKEY_CLASSES_ROOT, "Directory\shell\杀毒软件\command", hKey
ret = RegSetValueEx(hKey, "", 0, REG_SZ, ByVal App.Path & "\" & App.EXENAME & ".exe %1", ByVal
LenB(StrConv(App.Path & "\" & App.EXENAME & ".exe %1", vbFromUnicode)) + 1)
RegCloseKey hKey
End Function
```

由代码可见，实现的方向是调用注册表读写函数，对注册表中文件和文件夹右键相关键值进行读写，增加我们杀毒软件的信息。

接下来，是具体实现右键扫描文件的部分，在这里我们借助一个配置文件，通过实时向该配置文件写入待扫描文件信息来实现我们的需求。

假设配置文件文件名为：RCS.ini，其格式为：

```
[Normal]
WRFlag=0
RC=1
00001=c:\test.exe
```

WRFlag为读写标识，程序读写此配置文件时使用；RC是待扫描文件数量；序号对应的是待扫描文件路径。程序中编码如下：

```
Private Sub Form_Load()
```

判断是否有命令行参数

```
If Command <> "" Then
```

如果参数是文件并且存在,说明是右键扫描文件

```
If Dir(Command) <> "" Or Dir(Command, vbDirectory) <> "" Then
```

```
Dim sRightClickScanFile As String
```

右键扫描记录文件，用于记录即将要扫描的文件  
sRightClickScanFile = App.Path & "\RCS.ini"

WaitForWrite:

此标识用于防止该文件被同时读写

```
Dim lWriteReadFlag As Long
```

```
lWriteReadFlag = ReadIni(sRightClickScanFile, "Normal", "WRFlag")
```

如果标识为0，表示此时可以对此文件进行写操作

```
If lWriteReadFlag = 0 Then
```

接下来要进行写操作，修改防止同时读写的标识为1

```
Call WriteIni(sRightClickScanFile, "Normal", "WRFlag", 1)
```

要扫描的文件数量

```
Dim lPreCount As Long
```

```
lPreCount = ReadIni(sRightClickScanFile, "Normal", "RC")
```

增加记录数量

```
Call WriteIni(sRightClickScanFile, "Normal", "RC", lPreCount + 1)
```

增加记录，也就是要扫描的文件

```
Call WriteIni(sRightClickScanFile, "Normal", lPreCount + 1, Command)
```

此时已完成写入操作，修改防止读写标识为0

```
Call WriteIni(sRightClickScanFile, "Normal", "WRFlag", 0)
```

```
Else
```

执行到此else分枝说明程序正在进行写入操作，为避免冲突，此时要等待标识改变

```
DoEvents
```

```
Sleep 1
```

跳转到上面循环标识处

```
GoTo WaitForWrite
```

```
End If
```

```
End If
```

```
End Sub
```

这段代码的功能是：当程序启动时，通过获取启动参数判断是否是由右键扫描引发的启动。如果是，则获取待进行右键扫描的文件或文件夹，并将文件信息写入配置文件。

写入后，即要开始读取和扫描操作：

```
Private Sub RightClickScan()
```

```
Dim sRightClickScanFile As String
```

```
sRightClickScanFile = App.Path & "\RCS.ini"
```

```

WaitForWrite:
Dim lWriteReadFlag As Long
lWriteReadFlag = ReadIni(sRightClickScanFile, "Normal", "WRFlag")

If lWriteReadFlag = 0 Then
Call WriteIni(sRightClickScanFile, "Normal", "WRFlag", 1)
Dim lPreCount As Long
lPreCount = ReadIni(sRightClickScanFile, "Normal", "RC")

Dim i As Long
Dim sRightScanFile As String

For i = 1 To lPreCount
sRightScanFile = ReadIni(sRightClickScanFile, "Normal", i)

If GetAttr(sRightScanFile) And vbDirectory Then

    扫描目录
    RightClickScanCall sRightScanFile
Else

    扫描文件
    RightClickScanFile sRightScanFile
End If
Next i
Call WriteIni(sRightClickScanFile, "Normal", "RC", 0)
Call WriteIni(sRightClickScanFile, "Normal", "WRFlag", 0)
Else
DoEvents
GoTo WaitForWrite
End If

End Sub

```

这段代码，与上面的类似，因此没有做多余的注解，它用来读取配置文件中的待扫描文件或文件夹，并调用下级扫描函数RightClickScanFolder和RightClickScanFile进行对文件的扫描，这样就完整的实现了文件的右键扫描功能。在前面介绍扫描病毒的章节中已对扫描文件进行过详细的介绍，这里不再重复进行扫描文件编码介绍，读者朋友可以参考前面的代码。

接下来的再对几个其它方面的内容做介绍和编程实现原理，不给出示代码了，希望读者朋友尝试开动思维自我完成这些功能，一方面可以提高自己的实践能力，另一方面也可以考验一下自己的学习成果。当然，仅是希望尝试，如果目前还无法完成，不要但心，还可以参考后面附录中的部分，附录中会提供一个完整杀毒软件的源码，并在代码中备注有详尽的代码注释说明，会包含本书讲到的所有知识点。

### 3.8.2. 软件和病毒库自动升级

自动升级功能的实现很简单，前面我们介绍过了升级功能，自动升级，只需在程序中使用一个定时器，定时调

用升级功能即可。

### 3.8.3. 病毒隔离

病毒隔离是个概念型的实用功能，即在扫描到病毒后，给用户一个可选择的操作，如果用户选择隔离，则把文件复制到自己的目录中，以一定的规则给文件改个名称，建议去除后缀名，以防止误运行病毒文件。

### 3.8.4. 提取文件特征码

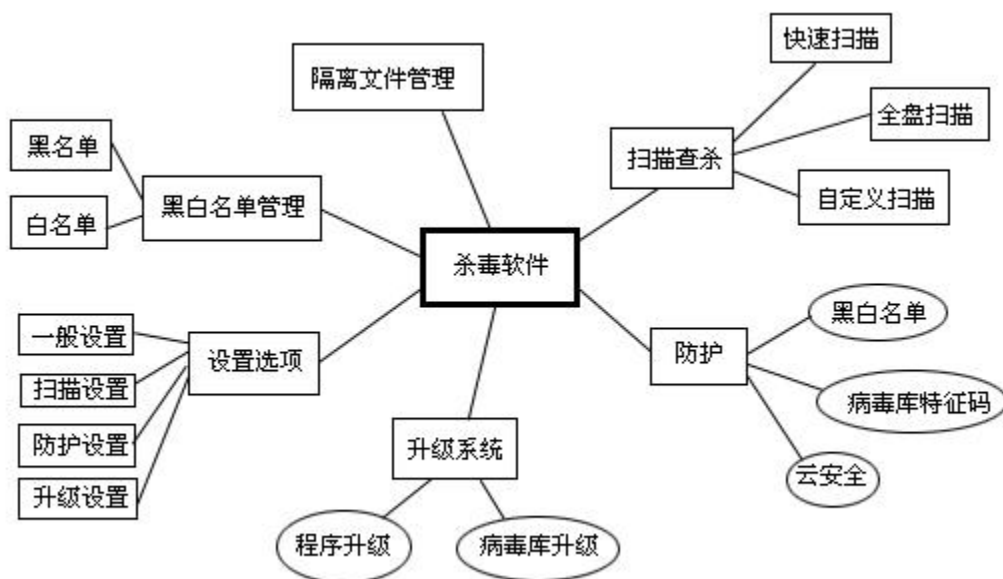
杀毒软件必须有病毒库的支持才能发挥其功能，因此病毒的日常维护是不可少的，随着新病毒的暴发旧病毒的消亡，病毒库也需要与时俱进。

病毒库的维护分为两个部分，本地病毒库和云安全病毒库。

本地病毒库提取特征码时，可使用上面讲过的扫描病毒时的HashFileStream函数中的代码获取文件特征码。云安全病毒库提取文件特征码时，使用上面讲过的HashFile函数的代码获取。

## 4. 杀毒软件功能设计总结

到此，一个具有基本功能的杀毒软件各部分功能的实现思路及编程方法已讲述完毕。在实际的杀毒软件编程中，需要整合以上所有功能，实现一个不可分隔的整体，各功能模块在很多方面是互有关联、相互影响的，代码也将会有条理的交织在一起。最终完成的将会是如下图所示的各功能的集合，也就是我们要实现的杀毒软件，当实现这一目标时，相信各位已经入门杀毒软件开发了。



本书所设计和讲述的杀毒软件开发，实现了传统杀毒软件的基础、主要功能，实现了杀毒防毒功能，从实用角度而言，已经具有确确实实的防毒杀毒能力。

但从功能及性能强度方面而言，还仅属于小型杀毒软件，与知名的大型杀毒软件相比有很大的差距。也有用户对如此设计软件的性能有怀疑，认为没有使用驱动，软件不够强大；在用户层做保护，安全防护性没达到最高，等等。这里我想说的是：我们开发杀毒软件，不是进行技术对抗、不是为了验证到底是魔高一丈还是道高一尺、也不是为了研究和模拟同类产品的功能，而是为了带给使用者实实在在的使用效果，保护用户的系统安全。

也许到此有的读者还没有读懂某些理论或未能理解某些编码，不必苦恼，古人云：书读百遍，其意自现，此读书法放到现代依旧实用。

本书到此并未完结，后面才是更加重量级的内容：一款商业杀毒软件完整源码，并加详细解读说明。相信各位读者在认真研读后，会有更大更实用的收获。

## 5. 完整的杀毒软件代码剖析

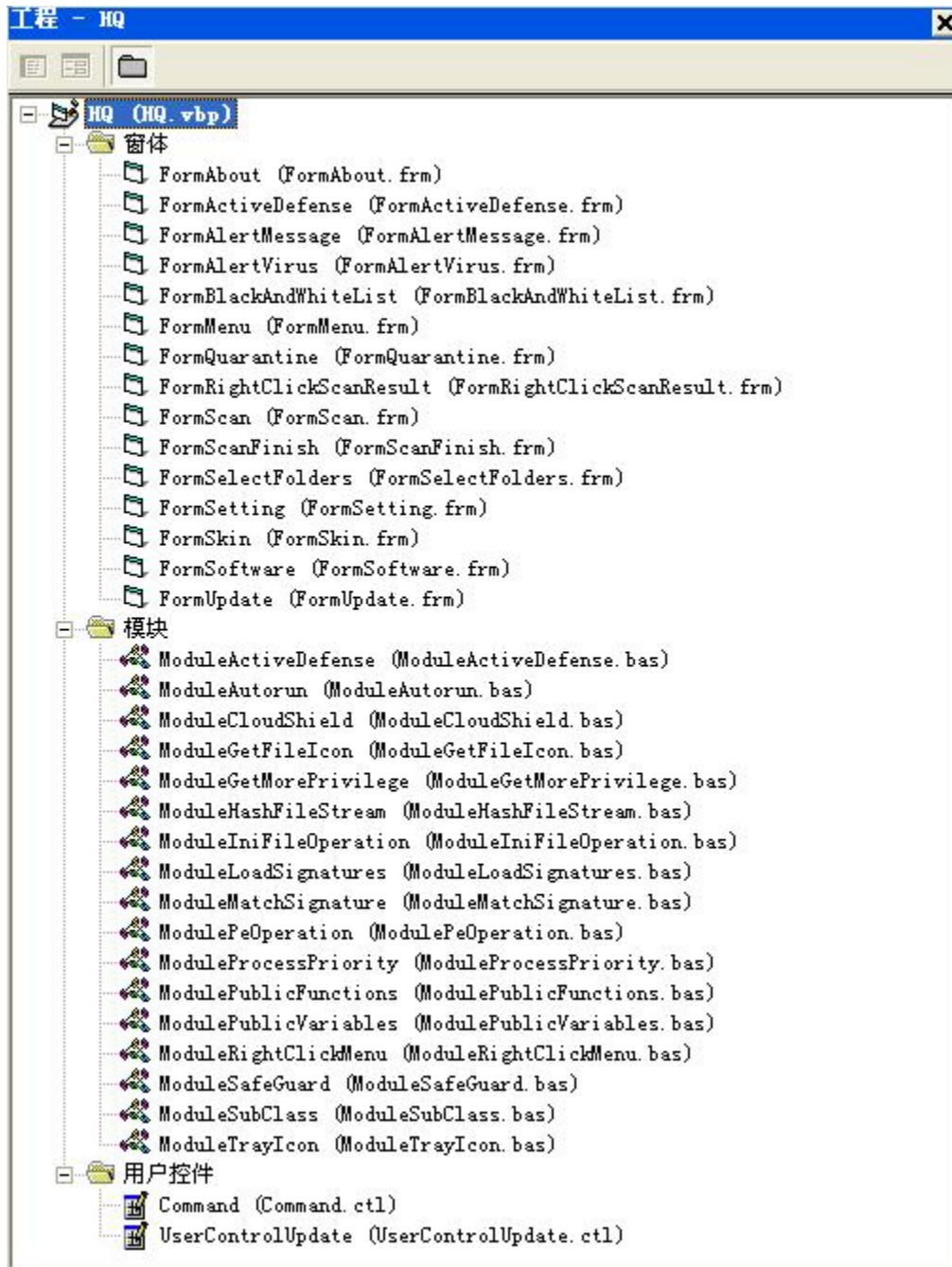
本书的前面部分，已经介绍了杀毒软件各功能的原理和实现，读到这里，应该对如何开发杀毒软件有了理论、方法上的认知，为了更进一步掌握具体的细节和实际编码方法，接下来将展示给大家的一个完整的杀毒软件源码，并加入了详细、具体的注解，所使用的理论和方法都采用的是前面讲过的内容。相信各位在认真读完后，会对理论有更深的体会和对杀毒软件入门级开发编程有更精准的把握。

在接下来的源代码展示、解说中，会出现重复使用的代码、类似的函数功能等看起来可能有些重复的内容，这部分没有做删除或精减，为的是展现完整的软件工程、呈现源代码的全貌。也只有这样才能使各位读者最精准的了解这个杀毒软件的真正全部内容。

### 5.1. 杀毒软件工程构成



整个工程主要由 13 个窗体文件、16 个模块文件、1 个控件文件构成，如下图所示：



## 5.2. 杀毒软件各窗体及模块功能概述

名称	类别	说明
FormActiveDefense	窗体	系统防护功能相关，当某程序启动时，如果该程序未检测到程序是病毒或恶意软件，但在软件设置中设置为启动程序时提示时，弹出此窗口。
FormAlertMessage	窗体	自动放行白名单文件、自动阻止病毒文件、升级完成时，如果设置中设置为进行提示则弹出此窗口提示用户。
FormAlertVirus	窗体	扫描病毒时，如果发现病毒，并且在设置中设置的是扫描到病毒后提示用户处理则弹出此窗口。
FormBlackAndWhiteList	窗体	黑白名单管理。
FormMenu	窗体	提供软件中各种使用的菜单，共三个菜单：托盘图标右键菜单、主界面中扫描完成后病毒列表中的右键菜单、主界面防护记录中的右键菜单。
FormQuarantine	窗体	隔离文件管理，这里可以对隔离文件进行删除或恢复。
FormRightClickScanResult	窗体	显示右键扫描结果。
FormScan	窗体	程序主界面。包含大量功能和操作，比如：进行快速扫描、全盘扫描、自定义扫描；显示软件版本号、病毒库版本号；显示和种设置状态，比如：是否启用着右键扫描、是否开启了云安全防护功能、是否开启着自动升级等等。
FormScanFinish	窗体	扫描完成后提示用户的窗体。
FormSelectFolders	窗体	进行自定义扫描时，让用户选择扫描目标的窗体，比如可以选择任意驱动器、目录。
FormSetting	窗体	对程序中各种功能进行配置和设置的窗体。
FormSkin	窗体	实现换肤功能。
FormUpdate	窗体	实现升级功能。
ModuleActiveDefense	模块	主动防护函数实现模块。
ModuleAutorun	模块	程序自启动实现模块。
ModuleCloudShield	模块	云安全防护实现模块。
ModuleGetFileIcon	模块	获取文件图标。
ModuleHashFileStream	模块	获取文件或文件节的哈希值。
ModuleIniFileOperation	模块	读写 Ini 文件。
ModuleLoadSignatures	模块	加载和验证特征码。
ModuleMatchSignature	模块	特征码匹配检测。
ModulePeOperation	模块	PE 文件操作各文件扫描。
ModuleProcessPriority	模块	设置进程优化级。
ModulePublicFunctions	模块	公共函数，包含了在程序中多次使用的全局函数。
ModulePublicVariables	模块	公共变量，这里定义的变量可以在所有窗体和模块中被访问。
ModuleRightClickMenu	模块	右键菜单关联与解除。
ModuleSafeGuard	模块	自我保护功能实现模块。
ModuleSubClass	模块	处理窗体消息，这其中会有大量操作，比如接收程序启动消息、弹出右键菜单、重建托盘图标等。
ModuleTrayIcon	模块	添加和移除托盘图标。
UserControlUpdate	控件	升级功能控件。

5.3. 窗体文件说明及代码剖析

5.3.1. 主动防御中拦截软件启动窗体

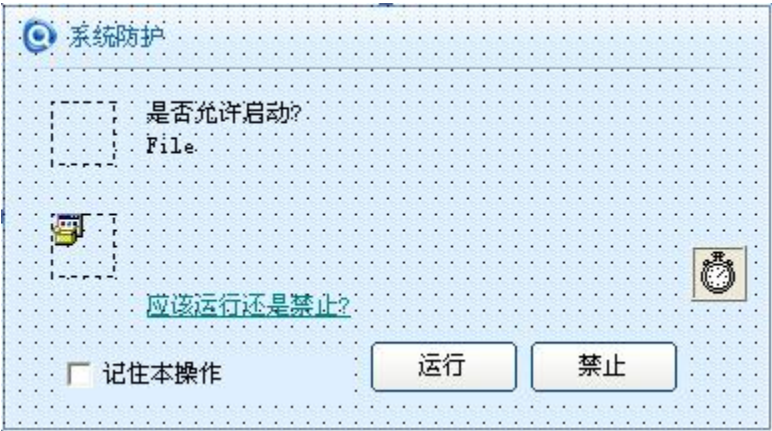
窗体：  
FormActiveDefense

功能：  
系统防护功能相关，当某程序启动时，如果该程序未检测到程序是病毒或恶意软件，但在软件设置中设置为启动程序时提示时，弹出此窗口。

在此窗口中，会显示出即将启动的程序的图标、路径。

用户可以选择放行或禁止运行该程序。

界面设计：



窗体中包含的控件及功能说明：

控件名称	类别	功能
ImageFileIcon	图片控件	用于显示将要启动的文件的图标，便于用户识别文件。
CheckRemember	复选框控件	“记住本操作”，再次遇到此文件启动时，不再提示，直接执行本次进行的运行或禁止操作。使用过此操作的文件，文件会被列入黑名单或白名单中。从黑白名单管理中可以看到相应的记录。
CommandRun	按钮控件	运行按钮。
CommandBlock	按钮控件	禁止按钮。

代码：  
  
' 强制变量声明，这样在程序中使用任何变量之前都必须事先声明，如果不声明，编译时会报错  
Option Explicit

' API 声明

' 函数功能：运行指定的程序

```
Private Declare Function WinExec Lib "kernel32" (ByVal lpCmdLine As String, ByVal nCmdShow As Long) As Long
```

' 函数功能：该函数将创建指定窗口的线程设置到前台，并且激活该窗口。键盘输入转向该窗口，并为用户改各种可视的记号。系统给创建前台窗口的线程分配的权限稍高于其他线程。

```
Private Declare Function SetForegroundWindow Lib "user32" (ByVal hWnd As Long) As Long
```

' 函数功能：该函数改变一个子窗口，弹出式窗口或顶层窗口的尺寸，位置和 Z 序。子窗口，弹出式窗口，及顶层窗口根据它们在屏幕上出现的顺序排序、顶层窗口设置的级别最高，并且被设置为 Z 序的第一个窗口。

```
Private Declare Function SetWindowPos Lib "user32" (ByVal hWnd As Long, ByVal hWndInsertAfter As Long, ByVal X As Long, ByVal Y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long
```

' 函数功能：播放声音

```
Private Declare Function PlaySound Lib "winmm.dll" Alias "PlaySoundA" (ByVal lpszName As String, ByVal hModule As Long, ByVal dwFlags As Long) As Long
```

' 函数功能：运行指定的程序

```
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hWnd As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
```

' 点击运行按钮

```
Private Sub CommandRun_Click()
```

' 把按钮不可操作化

```
CommandRun.Enabled = False
```

```
CommandBlock.Enabled = False
```

' 判断是否取得了文件图片

```
If ImageFileIcon = 0 Then
```

```
ImageFileIcon.Picture = ImageDefault16x16Icon.Picture
```

```
End If
```

```
If CheckRemember.value = 1 Then
```

' 主动防御规则文件

```
Dim sDefenseRuleFile As String
```

```
If Right(App.Path, 1) = "\" Then
```

```
sDefenseRuleFile = App.Path & "ActiveDefense.ini"
```

```
Else
```

```

sDefenseRuleFile = App.Path & "\ActiveDefense.ini"
End If

' 获取主动防御规则数
Dim lMaxID As Long
lMaxID = ReadIni(sDefenseRuleFile, "Count", "MaxID")

' 向 ini 文件中写入主动防御文件内容
' 更新最大 ID 值
Call WriteIni(sDefenseRuleFile, "Count", "MaxID", lMaxID + 1)

' 写文件路径
Call WriteIni(sDefenseRuleFile, "File", Format(lMaxID + 1, "00000"), LabelFile.Caption)

' 写文件特征码(文件 hash 操作获得的 md5 值)
Call WriteIni(sDefenseRuleFile, "Signature", Format(lMaxID + 1, "00000"), LabelSignature.Caption)

' 文件描述
Call WriteIni(sDefenseRuleFile, "Description", Format(lMaxID + 1, "00000"), "")

' 防御标志(放行还是阻止)
Call WriteIni(sDefenseRuleFile, "DefenseFlag", Format(lMaxID + 1, "00000"), 1)
End If

' 设置允许标志
lUserAction = 1
DoEvents

' 向软件界面控件中添加记录
With FormScan
.ListViewShieldLog.ListItems.Add , , .ListViewShieldLog.ListItems.Count + 1
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(1) = Date & " " & Time
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(2) = LabelFile.Caption
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(3) = "放行"
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(4) = "手动操作"

' 取文件图标
.ImageListListviewShieldLog.ListImages.Add, "Ico".ListViewShieldLog.ListItems.Count,
ImageFileIcon.Picture

' 设置 listview 控件行图标
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SmallIcon=. ImageListListviewShieldLog.ListImages.Item("Ico" .ListViewShieldLog.ListItems.Count).Key
End With
Dim sSettingsFile As String

' 软件设置的记录文件
If Right(App.Path, 1) = "\" Then
sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\Settings.ini"

```

End If

’ 读取云安全防护系统开启状态

Dim lCloudSecurty As Long

lCloudSecurty = ReadIni(sSettingsFile, "Shield", "CloudSecurty")

If lCloudSecurty = 1 Then

’ 如果云安全功能开启着，则向云安全服务器发送记录

If CheckRemember.value = 1 Then

Call CloudMatch("http://www.haiqi.cn/cloudsafe/answer.asp?Hash=" & LabelSignature.Caption & "&File=" & LabelFile.Caption & "&Flag=A" & "&Value=10")

Else

Call CloudMatch("http://www.haiqi.cn/cloudsafe/answer.asp?Hash=" & LabelSignature.Caption & "&File=" & LabelFile.Caption & "&Flag=A" & "&Value=1")

End If

End If

DoEvents

Unload Me

End Sub

**点击阻止按钮**

Private Sub CommandBlock\_Click()

’ 把按钮不可操作化

CommandRun.Enabled = False

CommandBlock.Enabled = False

’ 判断是否取得了文件图标

If ImageFileIcon = 0 Then

ImageFileIcon.Picture = ImageDefault16x16Icon.Picture

End If

If CheckRemember.value = 1 Then

’ 主动防御规则文件

Dim sDefenseRuleFile As String

If Right(App.Path, 1) = "\" Then

sDefenseRuleFile = App.Path & "ActiveDefense.ini"

Else

sDefenseRuleFile = App.Path & "\ActiveDefense.ini"

End If

’ 获取主动防御规则数量

Dim lMaxID As Long

lMaxID = ReadIni(sDefenseRuleFile, "Count", "MaxID")

’ 向 ini 文件中写入主动防御文件内容

’ 更新最大 ID 值

Call WriteIni(sDefenseRuleFile, "Count", "MaxID", lMaxID + 1)

```

' 写文件路径
Call WriteIni(sDefenseRuleFile, "File", Format(lMaxID + 1, "00000"), LabelFile.Caption)

' 写文件特征码(文件 hash 获得的 md5 值)
Call WriteIni(sDefenseRuleFile, "Signature", Format(lMaxID + 1, "00000"), LabelSignature.Caption)

' 文件描述
Call WriteIni(sDefenseRuleFile, "Description", Format(lMaxID + 1, "00000"), "")

' 防御标志(放行还是阻止)
Call WriteIni(sDefenseRuleFile, "DefenseFlag", Format(lMaxID + 1, "00000"), 0)
End If

' 设置阻止标志
lUserAction = 2

' 添加记录
With FormScan
.ListViewShieldLog.ListItems.Add , , .ListViewShieldLog.ListItems.Count + 1
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(1) = Date & " " & Time
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(2) = LabelFile.Caption
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(3) = "阻止"
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(4) = "手动操作"

' 取文件图标
.ImageListListviewShieldLog.ListImages.Add , "Ico" & .ListViewShieldLog.ListItems.Count,
ImageFileIcon.Picture

' 设置 listview 行图标
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SmallIcon
=. ImageListListviewShieldLog.ListImages.Item("Ico" .ListViewShieldLog.ListItems.Count).Key
End With
Dim sSettingsFile As String

' 软件设置记录文件
If Right(App.Path, 1) = "\" Then
sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\Settings.ini"
End If

' 读取云安全防护系统开启状态
Dim lCloudSecurty As Long
lCloudSecurty = ReadIni(sSettingsFile, "Shield", "CloudSecurty")
If lCloudSecurty = 1 Then

' 向云安全服务器发送记录
If CheckRemember.value = 1 Then
Call CloudMatch("http://www.haiqi.cn/cloudsafe/answer.asp?Hash=" & LabelSignature.Caption & "&File=" &
LabelFile.Caption & "&Flag=B" & "&Value=10")

```

```

Else
Call CloudMatch("http://www.haiqi.cn/cloudsafe/answer.asp?Hash=" & LabelSignature.Caption & "&File=" &
LabelFile.Caption & "&Flag=B" & "&Value=1")
End If
End If
DoEvents
Unload Me
End Sub

```

#### ’ 激活窗体时

```

Private Sub Form_Activate()
SetForegroundWindow Me.hWnd
End Sub

```

#### ’ 窗体启动函数

```

Private Sub Form_Load()
SetForegroundWindow Me.hWnd

```

```

’ 加载窗体的界面皮肤
ReSkinMe
End Sub

```

#### ’ 用定时器控制让窗体置于最前

```

Private Sub TimerBringToTop_Timer()

’ 将窗体置于屏幕最前
SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2
DoEvents

’ 判断窗体提示音文件是否存在
If Dir(App.Path & "\alert.wav") <> "" Then
PlaySound App.Path & "\alert.wav", 1, 1
End If

’ 恢复窗体位置
SetWindowPos Me.hWnd, -2, 0, 0, 0, 0, &H1 Or &H2
TimerBringToTop.Enabled = False
End Sub

```

#### ’ 软件换肤

```

Public Function ReSkinMe()
With Me
.Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\ActiveDefense.bmp")

’ 这里加载的是软件界面的背景图片，图片如下所示：

```





```
End With
End Function
```

5.3.2. 屏幕右下角弹出的消息提示窗体

窗体：  
FormAlertMessage

功能：  
自动放行白名单文件、自动阻止病毒文件、升级完成时，如果设置中设置为进行提示则弹出此窗口提示用户。

界面设计：



窗体中包含的控件及功能说明：

控件名称	类别	功能
LabelInfo	文本框控件	显示提示给用户的信息。
LabelAutoCloseAlertInterval	文本框控件	如果在设置中开启“自动关闭提示消息窗口”时，此处是用于显示多少秒后自动关闭窗口。
TimerAutoUnload	定时器控件	用于实现自动关闭窗口。
TimerBringToTop	定时器控件	用于将窗口放置到其它所有桌面窗口之前。
ImageExit	图片控件	用图片模拟的关闭按钮。

代码:

Option Explicit

' api 声明

```
Private Declare Function SetWindowPos Lib "user32" (ByVal hWnd As Long, ByVal hWndInsertAfter As Long,
ByVal X As Long, ByVal Y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long
```

' 函数功能: 该函数从当前线程中的窗口释放鼠标捕获, 并恢复通常的鼠标输入处理。捕获鼠标的窗口接收所有的鼠标输入 (无论光标的位置在哪里), 除非点击鼠标键时, 光标热点在另一个线程的窗口中。

```
Private Declare Function ReleaseCapture Lib "user32" () As Long
```

' 函数功能: 该函数将指定的消息发送到一个或多个窗口。此函数为指定的窗口调用窗口程序, 直到窗口程序处理完消息再返回。

```
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long, ByVal lParam As Any) As Long
```

```
Private Declare Function SetForegroundWindow Lib "user32" (ByVal hWnd As Long) As Long
```

```
Private Declare Function PlaySound Lib "winmm.dll" Alias "PlaySoundA" (ByVal lpszName As String, ByVal hModule As Long, ByVal dwFlags As Long) As Long
```

‘窗体激活时调用此函数

```
Private Sub Form_Activate()
SetForegroundWindow Me.hWnd
ReSkinMe
End Sub
```

\*窗体启动函数

```
Private Sub Form_Load()
SetForegroundWindow Me.hWnd
End Sub
```

‘鼠标在窗体按下时调用此函数

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

' 判断按下的是否是鼠标左键

```
If Button = vbLeftButton Then
```

' 为当前的应用程序释放鼠标捕获

```
ReleaseCapture
```

' 移动窗体

```
SendMessage Me.hWnd, &H1, 2, 0
```

```
End If
```

```
End Sub
```

‘‘鼠标在窗体按移动时调用此函数

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
’ 设置关闭按钮状态
```

```
ImageExit(0).Visible = True
```

```
ImageExit(1).Visible = False
```

```
ImageExit(2).Visible = False
```

```
End Sub
```

#### 退出窗体

```
Private Sub ImageExit_Click(Index As Integer)
```

```
’ 卸载本窗体
```

```
Unload Me
```

```
End Sub
```

#### 鼠标在退出窗体图片模拟的按钮上移动

```
Private Sub ImageExit_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
’ 退出按钮状态
```

```
ImageExit(0).Visible = False
```

```
ImageExit(1).Visible = True
```

```
ImageExit(2).Visible = False
```

```
End Sub
```

#### 鼠标在退出窗体图片模拟的按钮上按下

```
Private Sub ImageExit_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
’ 退出按钮状态
```

```
ImageExit(0).Visible = False
```

```
ImageExit(1).Visible = False
```

```
ImageExit(2).Visible = True
```

```
End Sub
```

#### 鼠标在退出窗体图片模拟的按钮上抬起

```
Private Sub ImageExit_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
’ 退出点击按钮
```

```
Unload Me
```

```
End Sub
```

#### 用定时器控制让窗体自动退出

```

Private Sub TimerAutoUnload_Timer()
If LabelAutoCloseAlertInterval <> 0 Then
LabelAutoCloseAlertInterval.Caption = CLng(LabelAutoCloseAlertInterval.Caption) - 1
LabelAutoCloseAlertInterval.Refresh
Else
Unload Me
End If
End Sub

```

#### 用定时器控制让窗体置于最前

```

Private Sub TimerBringToTop_Timer()

' 将窗体置于最前
SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2
DoEvents

' 窗体提示音
If Dir(App.Path & "\notify.wav") <> "" Then
PlaySound App.Path & "\notify.wav", 1, 1
End If

' 恢复窗体位置
SetWindowPos Me.hWnd, -2, 0, 0, 0, 0, &H1 Or &H2
TimerBringToTop.Enabled = False

' 软件设置记录文件
Dim sSettingsFile As String
If Right(App.Path, 1) = "\" Then
sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\Settings.ini"
End If

' 读取防护提醒开启状态
Dim lAutoCloseAlert As Long
lAutoCloseAlert = ReadIni(sSettingsFile, "Shield", "AutoCloseAlertMessage")
If lAutoCloseAlert = 1 Then

' 自动关闭提示窗口频率
Dim lAutoCloseAlertInterval As Long
lAutoCloseAlertInterval = ReadIni(sSettingsFile, "Shield", "AutoCloseAlertMessageInterval")

' 设置时间
LabelAutoCloseAlertInterval.Caption = lAutoCloseAlertInterval

' 激活自动关闭
TimerAutoUnload = True
Else
TimerAutoUnload = False

```

```

LabelAutoCloseAlertInterval.Visible = False
LabelLabelAutoTip.Visible = False
End If
DoEvents
Dim i As Long
For i = 0 To 2

' 初始化关闭按钮位置
With ImageExit(i)
.Left = 3480
.Top = 0
End With
Next

' 图片模拟的关闭按钮可见状态
ImageExit(0).Visible = True
ImageExit(1).Visible = False
ImageExit(2).Visible = False
End Sub

```

## 软件换肤

```


Public Function ReSkinMe()
With Me
.Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\ActiveDefense.bmp")


```


’ 上面是加载窗体的背景图片，图片如下：



’ 以下是加载的是界面右上角用图片模拟的退出按钮，之所以使用三张图片，是因为要实现按钮的三态效果，即：普通状态下、鼠标移动时、鼠标按下时三种不同状态时的效果。这里使用的三张图片分别是：

’ 正常状态下：

’ 鼠标移动时：

’ 鼠标按下时：

’ 在本程序中的其它地方，当用图片模拟退出按钮时，使用的都是同种方法，后文将不再重复说明述。

```

.ImageExit(0).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit0.bmp")
.ImageExit(1).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit1.bmp")
.ImageExit(2).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit2.bmp")

```

```
End With
End Function
```

5.3.3. 扫描病毒发现病毒时的提示窗体

窗体：  
FormAlertVirus

功能：  
扫描病毒时，如果发现病毒，并且在设置中设置的是扫描到病毒后提示用户处理则弹出此窗口。

界面设计：



窗体中包含的控件及功能说明：

控件名称	类别	功能
LabelVirusName	文本框控件	用于显示病毒名称，比如：Trojan.a.b。
LabelFile	文本框控件	用于显示文件路径。
CommandClear	按钮控件	清除按钮。
CommandQuarantine	按钮控件	隔离按钮。
CommandIgnore	按钮控件	忽略按钮。
ImageExit	图片控件	用图片模拟的退出按钮。

代码：

```
Option Explicit

' api 声明

' 函数功能：删除文件
Private Declare Function DeleteFile Lib "kernel32" Alias "DeleteFileA" (ByVal lpFileName As String) As
```

Long

’ 函数功能：复制文件

```
Private Declare Function CopyFile Lib "kernel32" Alias "CopyFileA" (ByVal lpExistingFileName As String,
ByVal lpNewFileName As String, ByVal bFailIfExists As Long) As Long
Private Declare Function ReleaseCapture Lib "user32" () As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long, ByVal lParam As Any) As Long
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hWnd As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As String,
ByVal nShowCmd As Long) As Long
```

‘点击清除按钮

```
Private Sub CommandClear_Click()
```

’ 清除病毒

```
Dim lDeleteFileReturn As Long
lDeleteFileReturn = DeleteFile(LabelFile.Caption)
If lDeleteFileReturn <> 0 Then
```

’ 清除成功

```
With FormScan.ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = LabelFile.Caption
.ListItems(.ListItems.Count).SubItems(2) = LabelVirusName.Caption
.ListItems(.ListItems.Count).SubItems(3) = "已清除"
End With
Else
```

’ 清除失败

```
With FormScan.ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = LabelFile.Caption
.ListItems(.ListItems.Count).SubItems(2) = LabelVirusName.Caption
.ListItems(.ListItems.Count).SubItems(3) = "清除失败"
End With
End If
Unload Me
End Sub
```

‘点击忽略按钮

```
Private Sub CommandIgnore_Click()
```

’ 向主窗体扫描结果控件中添加记录

```
With FormScan.ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = LabelFile.Caption
.ListItems(.ListItems.Count).SubItems(2) = LabelVirusName.Caption
```

```
.ListItems(.ListItems.Count).SubItems(3) = "未清除"
End With
Unload Me
End Sub
```

## 隔离文件

```
Private Sub CommandQuarantine_Click()

' 确保当前路径包含"\
Dim sTempAppPath As String
If Right(App.Path, 1) = "\" Then
sTempAppPath = App.Path
Else
sTempAppPath = App.Path & "\"
End If

' 目标复制文件名
Dim sTargetFileName As String
sTargetFileName = sTempAppPath & "quarantine\" & RndChr(8) & ".tmp"

' 复制文件
Dim lCopyFileReturn As Long
lCopyFileReturn = CopyFile(LabelFile.Caption, sTargetFileName, 0)

' 复制成功
If lCopyFileReturn <> 0 Then

' 清除病毒
Dim lDeleteFileReturn As Long
lDeleteFileReturn = DeleteFile(LabelFile.Caption)
If lDeleteFileReturn <> 0 Then

' 向主界面扫描结果窗体中添加记录
With FormScan.ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = LabelFile.Caption
.ListItems(.ListItems.Count).SubItems(2) = LabelVirusName.Caption
.ListItems(.ListItems.Count).SubItems(3) = "已隔离"
End With

' 隔离记录文件
Call WriteQuarantineFile(LabelFile.Caption, sTargetFileName)
Else

' 向主界面扫描结果窗体中添加记录
With FormScan.ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = LabelFile.Caption
.ListItems(.ListItems.Count).SubItems(2) = LabelVirusName.Caption
```



```

.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If

' 复制失败，隔离失败
Else

' 向主界面扫描结果窗体中添加记录
With FormScan.ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = LabelFile.Caption
.ListItems(.ListItems.Count).SubItems(2) = LabelVirusName.Caption
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If
Unload Me
End Sub

```

#### \*窗体启动函数

```

Private Sub Form_Load()

' 加载本窗体的界面皮肤
ReSkinMe
Dim i As Long
For i = 0 To 2

' 初始化关闭按钮位置
With ImageExit(i)
.Left = 7080
.Top = 0
End With
Next

' 关闭按钮
ImageExit(0).Visible = True
ImageExit(1).Visible = False
ImageExit(2).Visible = False
End Sub

```

#### \*鼠标在窗体中按下

```

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

' 按下鼠标左键
If Button = vbLeftButton Then

' 为当前的应用程序释放鼠标捕获
ReleaseCapture

```

```
' 移动窗体
SendMessage Me.hWnd, &HA1, 2, 0
End If
End Sub
```

#### '鼠标在窗体中移动

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
' 关闭按钮
ImageExit(0).Visible = True
ImageExit(1).Visible = False
ImageExit(2).Visible = False
End Sub
```

#### '退出窗体

```
Private Sub ImageExit_Click(Index As Integer)
CommandIgnore_Click
End Sub
```

#### '鼠标在退出按钮上移动

```
Private Sub ImageExit_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
' 退出按钮状态
ImageExit(0).Visible = False
ImageExit(1).Visible = True
ImageExit(2).Visible = False
End Sub
```

#### '鼠标在退出按钮上按下

```
Private Sub ImageExit_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
' 退出按钮状态
ImageExit(0).Visible = False
ImageExit(1).Visible = False
ImageExit(2).Visible = True
End Sub
```

#### '鼠标在退出按钮上抬起

```
Private Sub ImageExit_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
' 退出点击按钮
CommandIgnore_Click
```

End Sub

## 换肤

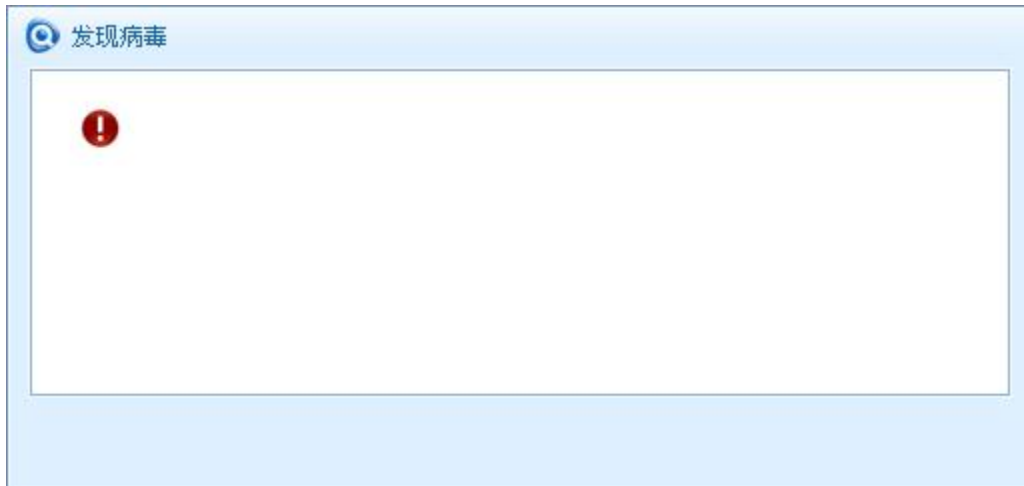
```
Public Function ReSkinMe()
```

```
With Me
```

```
’ 更换软件界面背景图片
```

```
.Picture = LoadPicture(App.Path & “\Skin\” & sSkin & “\AlertVirus.bmp”)
```

```
’ 被加载的背景图片为：
```



```
’ 更换界面右上角用图片模拟的退出按钮的图片
```

```
.ImageExit(0).Picture = LoadPicture(App.Path & “\Skin\” & sSkin & “\Exit0.bmp”)
```

```
.ImageExit(1).Picture = LoadPicture(App.Path & “\Skin\” & sSkin & “\Exit1.bmp”)
```

```
.ImageExit(2).Picture = LoadPicture(App.Path & “\Skin\” & sSkin & “\Exit2.bmp”)
```

```
End With
```

```
End Function
```

### 5.3.4. 黑白名单文件管理窗体

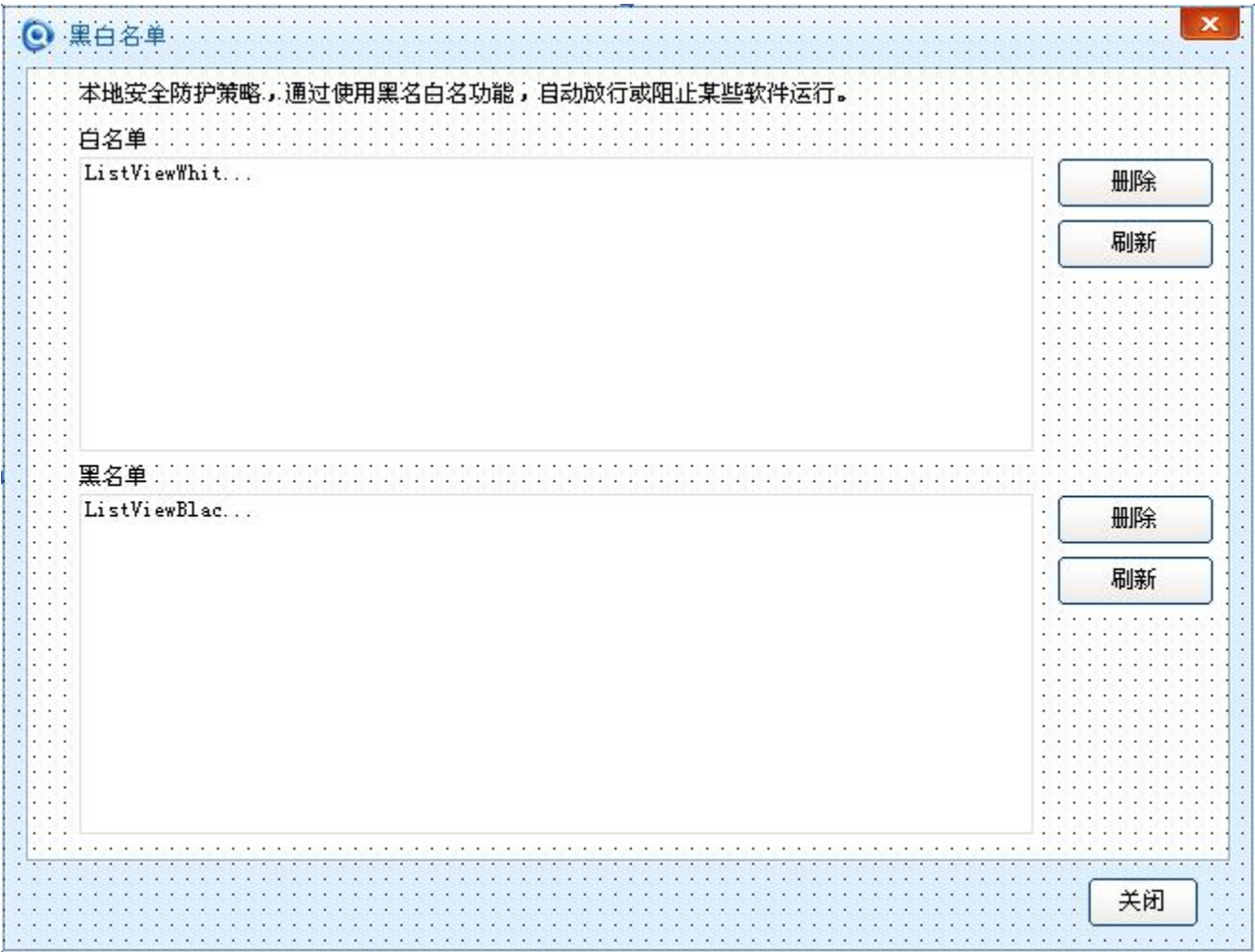
窗体：

FormBlackAndWhiteList

功能：

黑白名单管理。

界面设计：



窗体中包含的控件及功能说明：

控件名称	类别	功能
ListViewWhiteList	网格控件	放置白名单文件列表的网格控件。
ListViewBlackList	网格控件	放置黑名单文件列表的网格控件。
CommandDeleteWhiteListFile	按钮控件	删除白名单按钮。
CommandRefreshWhiteList	按钮控件	刷新白名单列表按钮。
CommandDeleteBlackListFile	按钮控件	删除黑名单按钮。
CommandRefreshBlackList	按钮控件	刷新黑名单列表按钮。
ImageExit	图片控件	用图片模拟的退出按钮。
CommandClose	按钮控件	关闭按钮。

代码：

```
Option Explicit
'api 声明
Private Declare Function ReleaseCapture Lib "user32" () As Long
```

```
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long, ByVal lParam As Any) As Long
```

#### ·点击关闭按钮

```
Private Sub CommandClose_Click()
```

```
    ' 卸载窗体  
    Unload Me  
End Sub
```

#### ·点击刷白名单按钮

```
Private Sub CommandRefreshWhiteList_Click()
```

```
    ' 清空白名单列表中原有内容  
    ListViewWhiteList.ListItems.Clear  
  
    ' 主动防御规则记录文件  
    Dim sDefenseIniFile As String  
    If Right(App.Path, 1) = "\" Then  
        sDefenseIniFile = App.Path & "ActiveDefense.ini"  
    Else  
        sDefenseIniFile = App.Path & "\ActiveDefense.ini"  
    End If  
  
    ' 获取主动防御规则数，也就是存在多少条黑白名单记录，每条记录都对应一个文件  
    Dim lMaxID As Long  
    lMaxID = ReadIni(sDefenseIniFile, "Count", "MaxID")  
  
    ' 规则对应文件  
    Dim sDefenseRuleFile As String  
  
    ' 规则标志，1 为放行，归为白名单文件，0 为禁止，归为黑名文件  
    Dim sDefenseRuleFileFlag As String  
    Dim i As Long  
    For i = 1 To lMaxID  
  
        ' 文件  
        sDefenseRuleFile = ReadIni(sDefenseIniFile, "File", Format(i, "00000"))  
  
        ' 取得放行文件，即白名单文件，它的标识为 1  
        If CLng(sDefenseRuleFileFlag) = 1 Then  
            If Trim(sDefenseRuleFile) <> "" Then  
  
                ' 向控件中添加文件记录  
                With FormBlackAndWhiteList  
  
                    ' ListView 的第一列设置为编号  
                    .ListViewWhiteList.ListItems.Add , , Format(i, "00000")
```

```

' Listview 的第二列显示文件名
.ListViewWhiteList.ListItems(.ListViewWhiteList.ListItems.Count).SubItems(1) = sDefenseRuleFile
End With
End If
End If

Next
End Sub

```

#### 点击刷新黑名单按钮

```

Private Sub CommandRefreshBlackList_Click()

' 清空控件中原有信息
ListViewBlackList.ListItems.Clear

' 主动防御规则记录文件
Dim sDefenseIniFile As String
If Right(App.Path, 1) = "\" Then
sDefenseIniFile = App.Path & "ActiveDefense.ini"
Else
sDefenseIniFile = App.Path & "\ActiveDefense.ini"
End If

' 获取主动防御规则数，与白名单文件的操作相似
Dim lMaxID As Long
lMaxID = ReadIni(sDefenseIniFile, "Count", "MaxID")

' 规则对应文件
Dim sDefenseRuleFile As String

' 规则标志，1 为放行，0 为禁止
Dim sDefenseRuleFileFlag As String
Dim i As Long
For i = 1 To lMaxID

' 文件
sDefenseRuleFile = ReadIni(sDefenseIniFile, "File", Format(i, "00000"))

' 标志
sDefenseRuleFileFlag = ReadIni(sDefenseIniFile, "DefenseFlag", Format(i, "00000"))

' 根据标识取得阻止记录文件，即黑名单文件
If CLng(sDefenseRuleFileFlag) = 0 Then
If Trim(sDefenseRuleFile) <> "" Then

' 向控件中添加黑名单记录
With FormBlackAndWhiteList

' Listview 控件第一列设置为序号

```

```
.ListViewBlackList.ListItems.Add , , Format(i, "00000")
```

```
' Listview 第二列显示黑名单文件
```

```
.ListViewBlackList.ListItems(.ListViewBlackList.ListItems.Count).SubItems(1) = sDefenseRuleFile
```

```
End With
```

```
End If
```

```
End If
```

```
Next
```

```
End Sub
```

#### **·点击删除黑名单文件按钮**

```
Private Sub CommandDeleteBlackListFile_Click()
```

```
' 判断控件中是否存在内容，如果没有内容则退出函数
```

```
If ListViewBlackList.ListItems.Count = 0 Then
```

```
Exit Sub
```

```
End If
```

```
' 主动防御规则记录文件
```

```
Dim sDefenseIniFile As String
```

```
If Right(App.Path, 1) = "\" Then
```

```
sDefenseIniFile = App.Path & "ActiveDefense.ini"
```

```
Else
```

```
sDefenseIniFile = App.Path & "\ActiveDefense.ini"
```

```
End If
```

```
' 弹出确认对话框
```

```
If MsgBox("确定要删除: " & ListViewBlackList.SelectedItem.SubItems(1) & "吗?", vbYesNo) = vbYes Then
```

```
' File 字段记录的是文件中，Signature 字段记录的是文件的特征码，在删除规则时，将这两项对应的内容清空即可
```

```
Call WriteIni(sDefenseIniFile, "File", ListViewBlackList.SelectedItem.Text, "")
```

```
Call WriteIni(sDefenseIniFile, "Signature", ListViewBlackList.SelectedItem.Text, "")
```

```
End If
```

```
' 删除后重新获取黑名单列表
```

```
CommandRefreshBlackList_Click
```

```
End Sub
```

#### **·点击删除白名单文件按钮**

```
Private Sub CommandDeleteWhiteListFile_Click()
```

```
' 判断控件中是否存在内容，如果没有内容则退出函数
```

```
If ListViewWhiteList.ListItems.Count = 0 Then
```

```
Exit Sub
```

```
End If
```

```
' 主动防御规则记录文件
```

```
Dim sDefenseIniFile As String
```

```

If Right(App.Path, 1) = "\" Then
sDefenseIniFile = App.Path & "ActiveDefense.ini"
Else
sDefenseIniFile = App.Path & "\ActiveDefense.ini"
End If

' 弹出确认对话框
If MsgBox("确定要删除: " & ListViewWhiteList.SelectedItem.SubItems(1) & "吗?", vbYesNo) = vbYes Then

' 删除操作是将 ini 文件中对应的记录清空
Call WriteIni(sDefenseIniFile, "File", ListViewWhiteList.SelectedItem.Text, "")
Call WriteIni(sDefenseIniFile, "Signature", ListViewWhiteList.SelectedItem.Text, "")
End If

' 刷新
CommandRefreshWhiteList_Click
End Sub

```

#### ' 鼠标在界面窗体中按下

```

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

' 判断是否按下鼠标左键
If Button = vbLeftButton Then

' 为当前的应用程序释放鼠标捕获
ReleaseCapture

' 移动窗体到鼠标拖拽位置
SendMessage Me.hWnd, &HAI1, 2, 0
End If
End Sub

```

#### '窗体启动函数

```

Private Sub Form_Load()

' 加载本窗体界面皮肤
ReSkinMe
Dim j As Long
For j = 0 To 2

' 初始化关闭按钮位置，设置在指定位置，且三个图片重叠在一起，这样可以模拟出按钮效果
With ImageExit(j)
.Left = 9240
.Top = 0
End With
Next

' 设置关闭按钮状态，设置第一张图片（也就是正常状态下按钮的形态）可见，其它两张图片（鼠标移动时的形态和

```



鼠标按下的形态) 不可见

```
ImageExit(0).Visible = True  
ImageExit(1).Visible = False  
ImageExit(2).Visible = False
```

’ 加载黑白名单文件

```
CommandRefreshWhiteList_Click  
CommandRefreshBlackList_Click  
End Sub
```

#### 鼠标在窗体界面中移动

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

’ 设置关闭按钮显示状态

```
ImageExit(0).Visible = True  
ImageExit(1).Visible = False  
ImageExit(2).Visible = False  
End Sub
```

#### 按下退出按钮

```
Private Sub ImageExit_Click(Index As Integer)
```

’ 卸载本窗体

```
Unload Me  
End Sub
```

#### 鼠标在退出按钮上移动

```
Private Sub ImageExit_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y  
As Single)
```

’ 设置退出按钮状态, 例正常状态下形态图片不可见、鼠标移动时的图片可见、鼠标按下时的图片不可见

```
ImageExit(0).Visible = False  
ImageExit(1).Visible = True  
ImageExit(2).Visible = False  
End Sub
```

#### 鼠标在退出按钮上按下

```
Private Sub ImageExit_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y  
As Single)
```

’ 设置退出按钮状态, 例正常状态下形态图片不可见、鼠标移动时的图片不可见、鼠标按下时的图片可见

```
ImageExit(0).Visible = False  
ImageExit(1).Visible = False  
ImageExit(2).Visible = True  
End Sub
```

### 鼠标在退出按钮上抬起

```
Private Sub ImageExit_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
' 卸载本窗体
```

```
Unload Me
```

```
End Sub
```

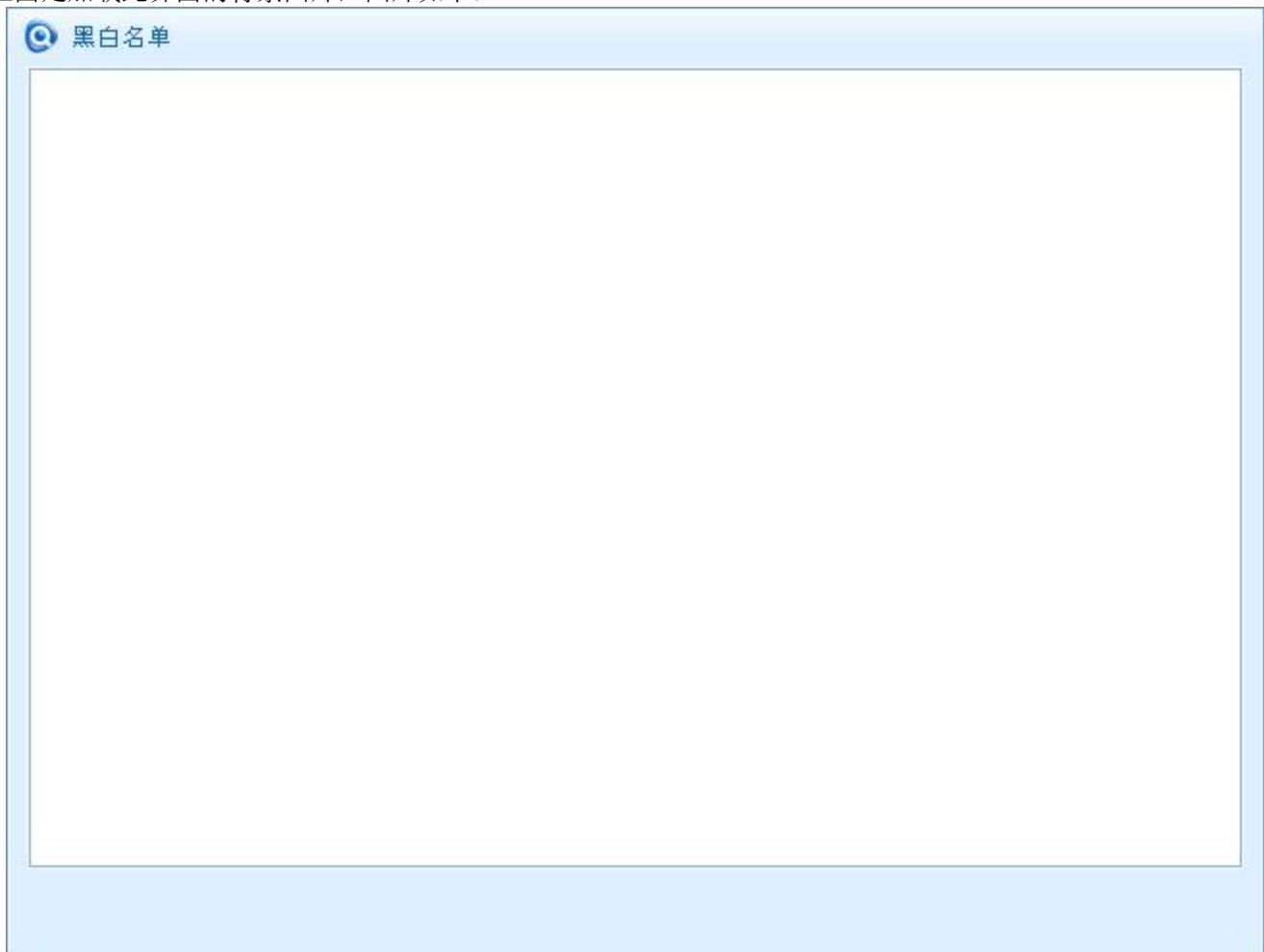
### 本窗体的换肤和加载皮肤函数

```
Public Function ReSkinMe()
```

```
With Me
```

```
.Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\BlackAndWhiteList.bmp")
```

```
' 上面是加载此界面的背景图片，图片如下：
```



```
' 加载图片模拟的三态退出按钮图片
```

```
.ImageExit(0).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit0.bmp")
```

```
.ImageExit(1).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit1.bmp")
```

```
.ImageExit(2).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit2.bmp")
```

```
End With
```

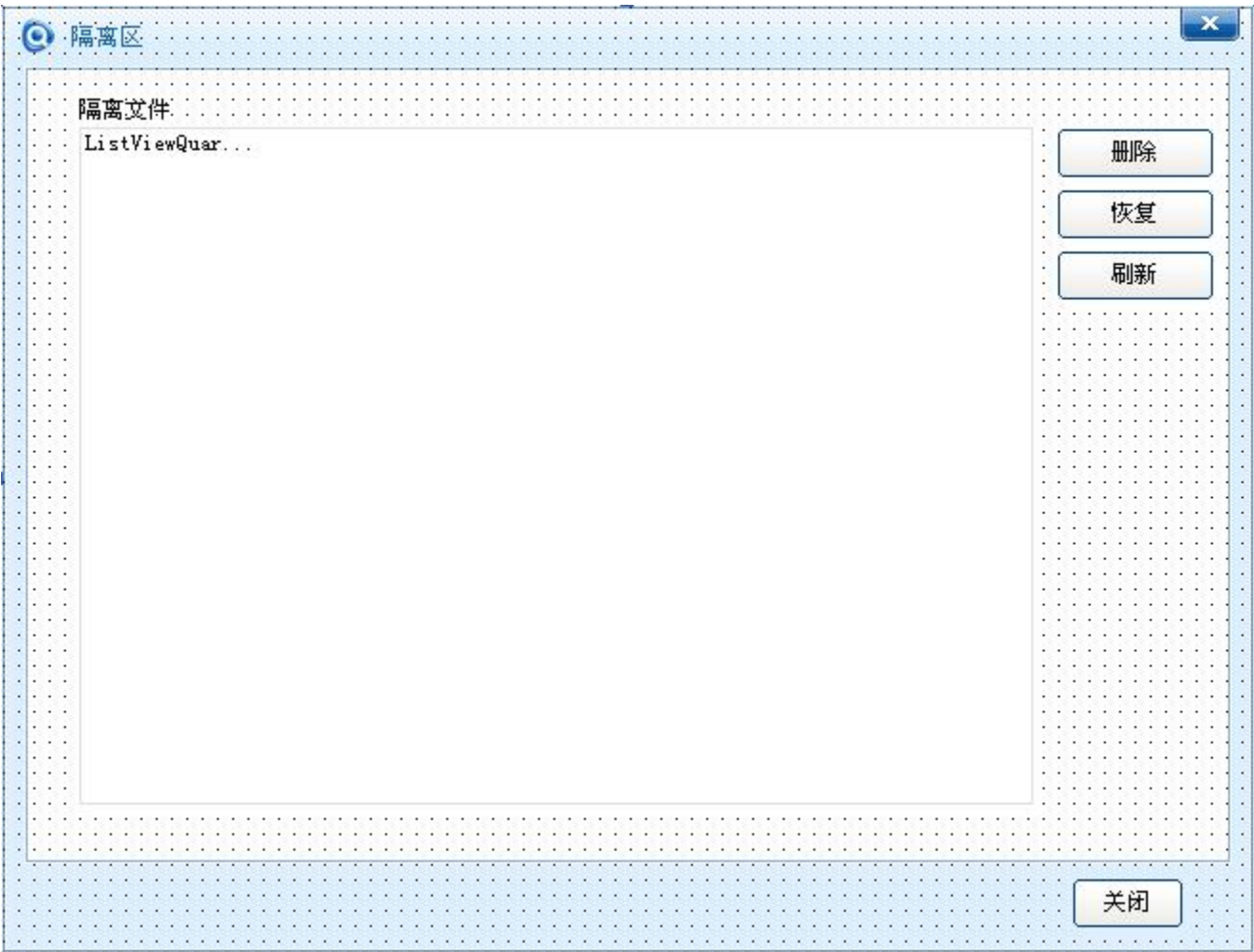
```
End Function
```

5.3.5. 隔离文件管理窗体

窗体：  
FormQuarantine

功能：  
隔离文件管理，这里可以对隔离文件进行删除或恢复。

界面设计：



窗体中包含的控件及功能说明：

控件名称	类别	功能
ListViewQuarantine	网格控件	放置隔离文件列表的网格控件。
CommandDelete	按钮控件	删除按钮。
CommandRestore	按钮控件	恢复按钮。
CommandRefresh	按钮控件	刷新按钮。
CommandClose	按钮控件	关按按钮。

ImageExit	图片按钮	用图片模拟的退出按钮。
-----------	------	-------------

隔离记录文件说明：  
隔离记录文件是用来记录哪些文件被隔离处理了，在本程序中使用 ini 文件格式，文件名为：quarantine.ini，其内容格式如下：

```
[Count]
MaxID=1

[SourceFile]
0001=c:\test\test.exe

[QuarantineFile]
0001=d:\software\hq\62A87C.exe
```

Count 字段中的 MaxID 记录共有多少个隔离文件；SourceFile 字段是记录隔离前文件的原始位置；QuarantineFile 字段是记录被隔离文件的隔离位置。这两个字段的存在，是为了使被隔离文件恢复到原始隔离前位置。

代码：

```
Option Explicit

' api 声明，上面说明过的 API 在下文将不再重复说明
Private Declare Function ReleaseCapture Lib "user32" () As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long, ByVal lParam As Any) As Long
Private Declare Function DeleteFile Lib "kernel32" Alias "DeleteFileA" (ByVal lpFileName As String) As Long
Private Declare Function CopyFile Lib "kernel32" Alias "CopyFileA" (ByVal lpExistingFileName As String, ByVal lpNewFileName As String, ByVal bFailIfExists As Long) As Long
```

•点击关闭按钮

```
Private Sub CommandClose_Click()
Unload Me
End Sub
```

删除隔离文件

```
Private Sub CommandDelete_Click()

' 判断 listview 控件中是否有内容，如果没有内容，则直接退出函数
If ListViewQuarantine.ListItems.Count = 0 Then
Exit Sub
End If

' 隔离记录文件
Dim sQuarantineIniFile As String
```

在不同的系统中，用 Vb 内置方法 App.Path 获取的程序自身路径会有差别，有的会带有 “\”，有的却没有。例如：“C:\test” 和 “C:\test\” 的不同。这里对最后一位符号进行判断，根据最后一位是不是 “\” 符号，连接文件名

时给予不同的操作。

```
If Right(App.Path, 1) = "\" Then
sQuarantineIniFile = App.Path & "Quarantine.ini"
Else
sQuarantineIniFile = App.Path & "\"Quarantine.ini"
End If

' 弹出确认对话框
If MsgBox("确定要删除: " & ListViewQuarantine.SelectedItem.SubItems(1) & "吗?", vbYesNo) = vbYes Then

' 获取隔离文件
Dim sQuarantineFile As String
sQuarantineFile = ReadIni(sQuarantineIniFile, "QuarantineFile",
Format(ListViewQuarantine.SelectedItem.Text, "00000"))

' 从隔离区删除文件
Call DeleteFile(sQuarantineFile)

' 清空文件原位置记录
Call WriteIni(sQuarantineIniFile, "SourceFile", ListViewQuarantine.SelectedItem.Text, "")

' 清空隔离文件位置记录
Call WriteIni(sQuarantineIniFile, "QuarantineFile", ListViewQuarantine.SelectedItem.Text, "")
End If

' 刷新隔离文件记录
CommandRefresh_Click
End Sub
```

### 刷新隔离文件

```
Private Sub CommandRefresh_Click()
ListViewQuarantine.ListItems.Clear

' 隔离记录文件
Dim sQuarantineIniFile As String
If Right(App.Path, 1) = "\" Then
sQuarantineIniFile = App.Path & "Quarantine.ini"
Else
sQuarantineIniFile = App.Path & "\"Quarantine.ini"
End If

' 获取隔离文件数量
Dim lMaxID As Long
lMaxID = ReadIni(sQuarantineIniFile, "Count", "MaxID")

' 隔离前的文件地址
Dim sSourceFile As String
Dim i As Long
For i = 1 To lMaxID
```

```

' 使用 For 循环，读取隔离文件数量条记录
sSourceFile = ReadIni(sQuarantineIniFile, "SourceFile", Format(i, "00000"))
If Trim(sSourceFile) <> "" Then

' 向控件中添加记录
With ListViewQuarantine
.ListItems.Add , , Format(i, "00000")

' 向 Listview 控件中添加隔离文件记录时，只添加文件原地址，不对目标地址（也就是隔离后文件存放的地址，在
本软件中，被隔离文件存放在软件目录中的 quarantine 目录下）进行添加。
.ListItems(.ListItems.Count).SubItems(1) = sSourceFile
End With
End If
Next
End Sub

```

### 将被隔离文件恢复

```

Private Sub CommandRestore_Click()
If ListViewQuarantine.ListItems.Count = 0 Then
Exit Sub
End If

' 隔离记录文件
Dim sQuarantineIniFile As String
If Right(App.Path, 1) = "\" Then
sQuarantineIniFile = App.Path & "Quarantine.ini"
Else
sQuarantineIniFile = App.Path & "\Quarantine.ini"
End If

' 弹出确认对话框
If MsgBox("确定要恢复: " & ListViewQuarantine.SelectedItem.SubItems(1) & "吗?", vbYesNo) = vbYes Then

' 隔离文件，也就是隔离后文件存放的位置
Dim sQuarantineFile As String
sQuarantineFile=ReadIni(sQuarantineIniFile,"QuarantineFile",
Format(ListViewQuarantine.SelectedItem.Text, "00000"))

' 恢复文件时，将文件从隔离位置复制回原始位置，原始位置是在文件被隔离时记录到隔离记录文件中的
Call CopyFile(sQuarantineFile, ListViewQuarantine.SelectedItem.SubItems(1), 0)
DoEvents

' 从隔离区删除文件
Call DeleteFile(sQuarantineFile)

' 清空文件原位置记录
Call WriteIni(sQuarantineIniFile,"SourceFile",ListViewQuarantine.SelectedItem.Text, "")

```

```

' 清空隔离文件位置记录，清空这两项记录是因为文件已经被恢复了，不能再在隔离管理器中显示这条记录
Call WriteIni(sQuarantineIniFile, "QuarantineFile", ListViewQuarantine.SelectedItem.Text, "")
End If

' 刷新隔离文件
CommandRefresh_Click
End Sub

```

#### 在窗体界面中移动鼠标

```

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

' 判断是否按下鼠标左键
If Button = vbLeftButton Then

' 为当前的应用程序释放鼠标捕获
ReleaseCapture

' 移动窗体
SendMessage Me.hWnd, &HA1, 2, 0
End If
End Sub

```

#### 窗体启动函数

```

Private Sub Form_Load()

' 为本窗体加载界面皮肤
ReSkinMe
Dim j As Long
For j = 0 To 2

' 初始化关闭按钮位置
With ImageExit(j)
.Left = 9240
.Top = 0
End With
Next

' 关闭按钮
ImageExit(0).Visible = True
ImageExit(1).Visible = False
ImageExit(2).Visible = False

' 显示隔离名单
CommandRefresh_Click
End Sub

```

#### 鼠标在窗体界面中移动

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
’ 关闭按钮
```

```
ImageExit(0).Visible = True  
ImageExit(1).Visible = False  
ImageExit(2).Visible = False  
End Sub
```

#### 点击退出按钮

```
Private Sub ImageExit_Click(Index As Integer)  
Unload Me  
End Sub
```

#### 鼠标在退出按钮上移动

```
Private Sub ImageExit_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y  
As Single)  
  
’ 退出按钮状态  
ImageExit(0).Visible = False  
ImageExit(1).Visible = True  
ImageExit(2).Visible = False  
End Sub
```

#### 鼠标在退出按钮上按下

```
Private Sub ImageExit_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y  
As Single)  
  
’ 退出按钮状态  
ImageExit(0).Visible = False  
ImageExit(1).Visible = False  
ImageExit(2).Visible = True  
End Sub
```

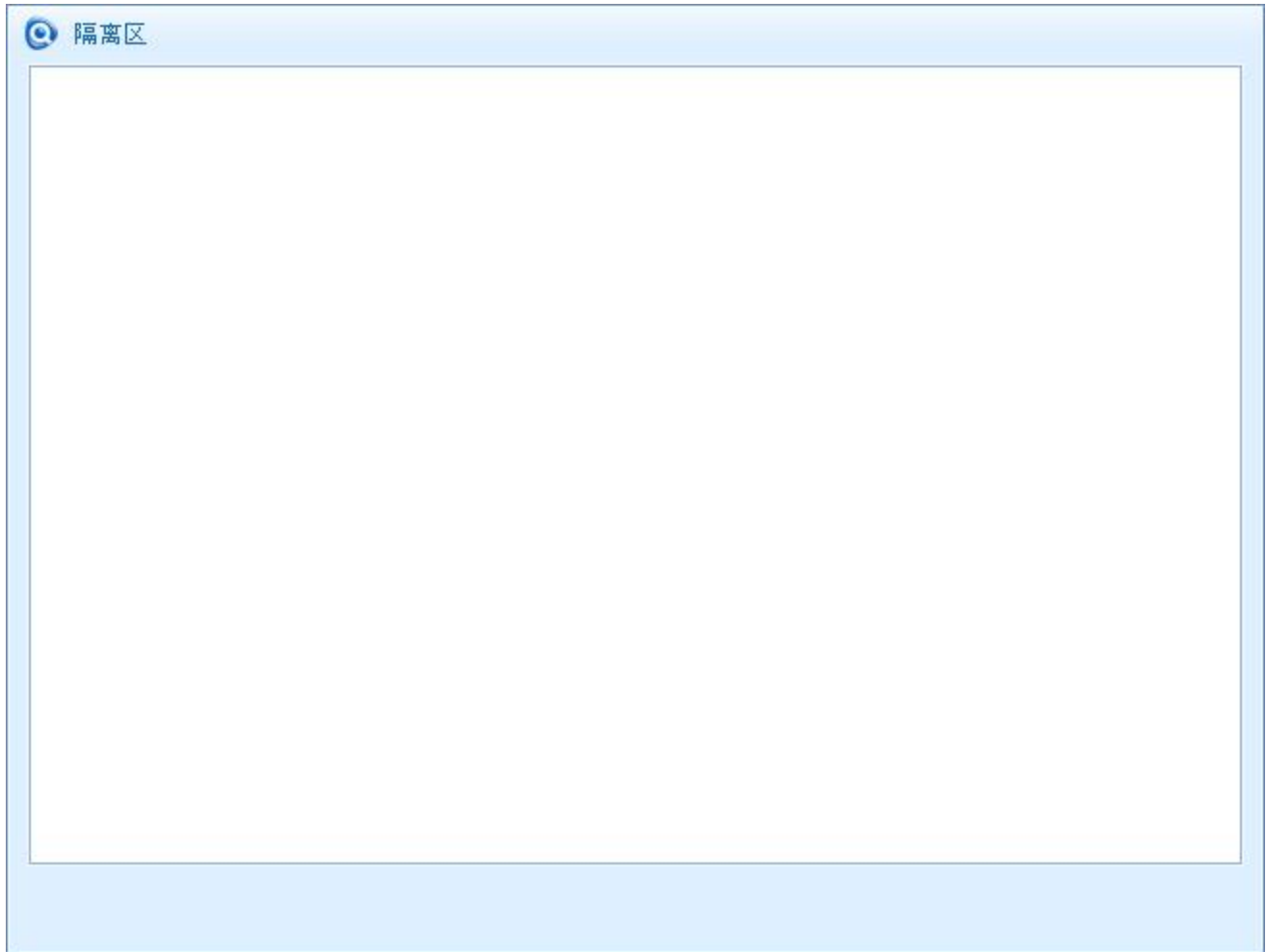
#### 鼠标在退出按钮上抬起

```
Private Sub ImageExit_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As  
Single)  
  
’ 退出点击按钮  
Unload Me  
End Sub
```

#### ’ 本窗体的换肤函数

```
Public Function ReSkinMe()  
With Me  
.Picture = LoadPicture(App.Path & “\Skin\” & sSkin & “\Quarantine.bmp”)  
  
’ 以上是加载窗体的背景图片，图片如下：
```





’ 加载图片模拟的三态退出按钮图片

```
.ImageExit(0).Picture = LoadPicture(App.Path & "\\Skin\\" & sSkin & "\\Exit0.bmp")  
.ImageExit(1).Picture = LoadPicture(App.Path & "\\Skin\\" & sSkin & "\\Exit1.bmp")  
.ImageExit(2).Picture = LoadPicture(App.Path & "\\Skin\\" & sSkin & "\\Exit2.bmp")  
End With  
End Function
```

### 5.3.6. 右键扫描结果窗体

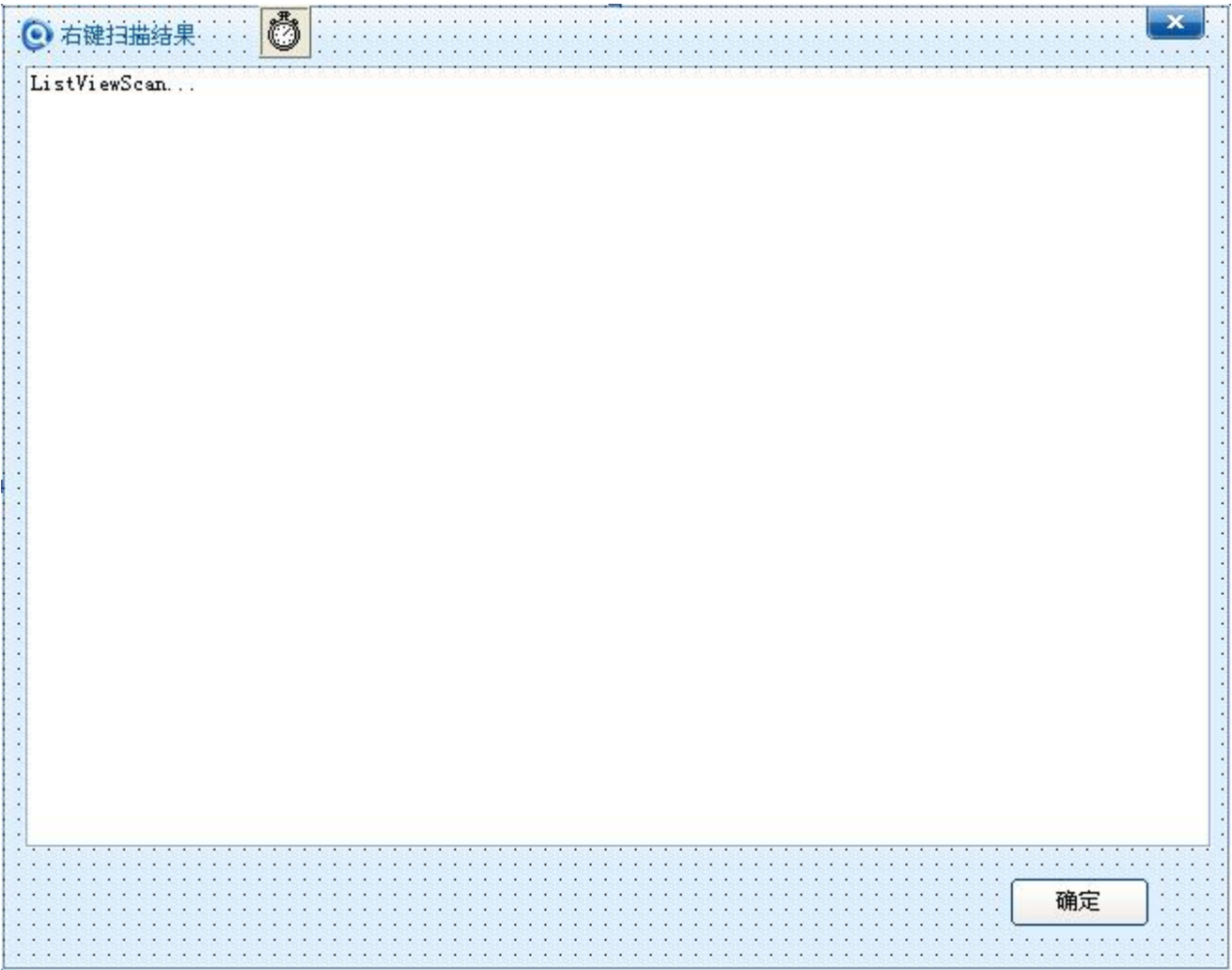
窗体:

FormRightClickScanResult

功能:

显示右键扫描结果。

界面设计:



窗体中包含的控件及功能说明：

控件名称	类别	功能
ListViewScanResult	网格控件	放置扫描结果的网格控件。
CommandUnload	按钮控件	确定按钮。
ImageExit	图片控件	用图片模拟的退出按钮。
TimerBringToTop	定时器控件	用于将窗口置于所有桌面窗口最前面的定时器控件。

代码：

```
Option Explicit
'api 声明
Private Declare Function SetWindowPos Lib "user32" (ByVal hWnd As Long, ByVal hWndInsertAfter As Long,
ByVal X As Long, ByVal Y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long
Private Declare Function ReleaseCapture Lib "user32" () As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long,
ByVal wParam As Long, lParam As Any) As Long
Private Declare Function SetForegroundWindow Lib "user32" (ByVal hWnd As Long) As Long
```

```
Private Declare Function PlaySound Lib "winmm.dll" Alias "PlaySoundA" (ByVal lpszName As String, ByVal hModule As Long, ByVal dwFlags As Long) As Long
```

#### 退出本窗体

```
Private Sub CommandUnload_Click()  
Unload Me  
End Sub
```

#### 窗体活动时调用此函数

```
Private Sub Form_Activate()  
  
' 为本窗体加载界面皮肤  
ReSkinMe  
  
' 将本窗体设置在桌面最新方，因为此窗体显示是正是右键扫描完成时，需要将扫描结果显示给用户  
SetForegroundWindow Me.hWnd  
End Sub
```

#### 窗体启动函数

```
Private Sub Form_Load()  
  
' 设置本窗体显示在屏幕最新端  
SetForegroundWindow Me.hWnd  
End Sub
```

#### 鼠标在窗体上按下

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)  
  
' 判断是否按下鼠标左键  
If Button = vbLeftButton Then  
  
' 为当前的应用程序释放鼠标捕获  
ReleaseCapture  
  
' 移动窗体  
SendMessage Me.hWnd, &HA1, 2, 0  
End If  
End Sub
```

#### 鼠标在窗体上移动

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)  
  
' 关闭按钮  
ImageExit(0).Visible = True  
ImageExit(1).Visible = False
```

```
ImageExit(2).Visible = False  
End Sub
```

#### 退出按钮

```
Private Sub ImageExit_Click(Index As Integer)  
Unload Me  
End Sub
```

#### 鼠标在退出按钮上移动

```
Private Sub ImageExit_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y  
As Single)  
  
' 退出按钮状态  
ImageExit(0).Visible = False  
ImageExit(1).Visible = True  
ImageExit(2).Visible = False  
End Sub
```

#### 鼠标在退出按钮上按下

```
Private Sub ImageExit_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y  
As Single)  
  
' 退出按钮状态  
ImageExit(0).Visible = False  
ImageExit(1).Visible = False  
ImageExit(2).Visible = True  
End Sub
```

#### 鼠标在退出按钮上抬起

```
Private Sub ImageExit_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As  
Single)  
  
' 卸载本窗体  
Unload Me  
End Sub
```

#### 定时器控制让窗体置于最前

```
Private Sub TimerBringToTop_Timer()  
  
' 将窗体置于最前  
SetWindowPos Me.hWnd, -1, 0, 0, 0, 0, &H1 Or &H2  
DoEvents  
  
' 窗体提示音  
If Dir(App.Path & "\notify.wav") <> "" Then
```

```

PlaySound App.Path & "\notify.wav", 1, 1
End If

' 恢复窗体位置
SetWindowPos Me.hWnd, -2, 0, 0, 0, 0, &H1 Or &H2
TimerBringToTop.Enabled = False

' 软件设置记录文件
Dim sSettingsFile As String
If Right(App.Path, 1) = "\" Then
sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\Settings.ini"
End If
DoEvents
Dim i As Long
For i = 0 To 2

' 初始化关闭按钮位置
With ImageExit(i)

' 距左侧距离
.Left = 9120

' 距上方距离
.Top = 0
End With
Next

' 关闭按钮
ImageExit(0).Visible = True
ImageExit(1).Visible = False
ImageExit(2).Visible = False
End Sub

```

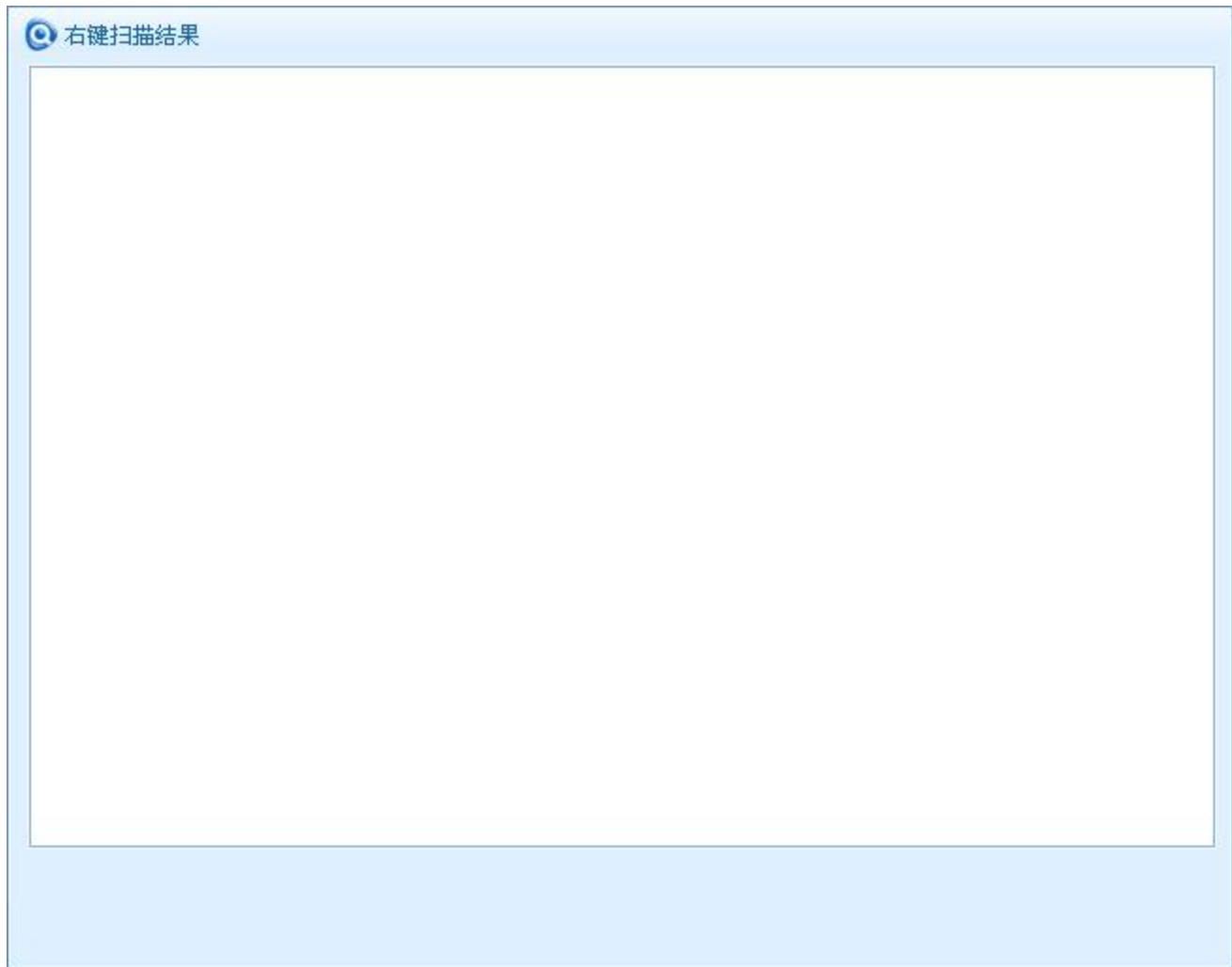
#### 本窗体的换肤函数

```

Public Function ReSkinMe()
With Me
.Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\RightClickScanResult.bmp")

' 以上是加载窗体背景图片，图片为：

```



’ 加载图片模拟的三态退出按钮图片

```
.ImageExit(0).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit0.bmp")  
.ImageExit(1).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit1.bmp")  
.ImageExit(2).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit2.bmp")  
End With  
End Function
```

### 5.3.7. 杀毒软件主界面窗体

窗体:

FormScan

功能:

程序主界面。包含大量功能和操作，比如：进行快速扫描、全盘扫描、自定义扫描；显示软件版本号、病毒库版本号；显示和种设置状态，比如：是否启用着右键扫描、是否开启了云安全防毒功能、是否开启着自动升级等等。

界面设计:



窗体中包含的控件及功能说明：

控件名称	类别	功能
ImageSkin	图片控件	用图片模拟的右上角的换肤按钮。
ImageMin	图片控件	用图片模拟的最小化按钮。
ImageExit	图片控件	用图片模拟的退出按钮。
LabelLastScanUsedTime	文本框控件	用于显示扫描使用的时间。
LabelLastScanFiles	文本框控件	用于显示扫描文件数量。
LabelLastDetectedSpywares	文本框控件	用于显示检测到的病毒数量。
ImageMemoryScan	图片控件	图片控件模拟的快速扫描按钮。
ImageFullDiskScan	图片控件	图片控件模拟的全盘扫描按钮。
ImageCustomerScan	图片控件	图片控件模拟的自定义扫描按钮。
ImageSecurityState	图片控件	图片控件模拟的安装状态选项卡。

ImageScanResult	图片控件	图片控件模拟的扫描结果选项卡。
ImageShield	图片控件	图片控件模拟的防护状态选项卡。
LabelSoftwareVersion	文本框控件	用于显示软件版本号。
LabelSignatureVersion	文本框控件	用于显示病毒库版本号。
LabelAutorun	文本框控件	用于显示软件是否开机自启动。
LabelAutoMin	文本框控件	用于显示软件是否启动后最小化。
LabelShieldSelf	文本框控件	用于显示软件的防护状态。
LabelAutoUpdateState	文本框控件	用于显示是否开启了自动升级。
LabelCloud	文本框控件	用于显示是否启用云安全。
LabelRightClickMenu	文本框控件	用于显示是否关联了文件或文件夹右键扫描菜单。
LabelActionWhenDetectVirus	文本框控件	用于显示检测到病毒的处理方式：自动清除、提示或忽略。
LabelShieldLevel	文本框控件	用于显示防护级别。
ImageUpdate	图片控件	用图片控件模拟的升级按钮。
ImageSetting	图片控件	用图片控件模拟的设置按钮。
ImageListListViewShieldLog	图片列表控件	实时存放文件图标，用于增加防护记录时，增加记录中的文件图标。
TimerRightClickScan	定时器控件	用于右键扫描。
TimerUpdateRefresh	定时器控件	用于实现自动升级。
ListViewShieldLog	网格控件	用于显示防护记录的网格控件。
ListViewScanResult	网格控件	用于显示扫描结果的网格控件。
ListProcess	列表控件	用于存放进程列表，此列表在进行快速扫描时使用。此控件不可见。
ListFolders	列表控件	用于存放文件列表，此列表在进行自定义扫描或全盘扫描时使用。此控件不可见。

代码：

Option Explicit

’ API 声明

’ 函数功能：获取一个字串，其中包含了当前所有逻辑驱动器的根驱动器路径

```
Private Declare Function GetLogicalDriveStrings Lib "kernel32" Alias "GetLogicalDriveStringsA" (ByVal nBufferLength As Long, ByVal lpBuffer As String) As Long
```

’ 函数功能：判断一个磁盘驱动器的类型

```
Private Declare Function GetDriveType Lib "kernel32" Alias "GetDriveTypeA" (ByVal nDrive As String) As Long
```



’ 函数功能：根据文件名查找文件

```
Private Declare Function FindFirstFile Lib "kernel32" Alias "FindFirstFileA" (ByVal lpFileName As String, lpFindFileData As WIN32_FIND_DATA) As Long
```

’ 函数功能：根据调用 FindFirstFile 函数时指定的一个文件名查找下一个文件

```
Private Declare Function FindNextFile Lib "kernel32" Alias "FindNextFileA" (ByVal hFindFile As Long, lpFindFileData As WIN32_FIND_DATA) As Long
```

’ 函数功能：关闭由 FindFirstFile 函数创建的一个搜索句柄

```
Private Declare Function FindClose Lib "kernel32" (ByVal hFindFile As Long) As Long
Private Declare Function DeleteFile Lib "kernel32" Alias "DeleteFileA" (ByVal lpFileName As String) As Long
Private Declare Function CopyFile Lib "kernel32" Alias "CopyFileA" (ByVal lpExistingFileName As String, ByVal lpNewFileName As String, ByVal bFailIfExists As Long) As Long
Private Declare Function ReleaseCapture Lib "user32" () As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long, ByVal wMsg As Long, ByVal lParam As Any) As Long
```

’ process32First 是一个进程获取函数,当我们利用函数 CreateToolhelp32Snapshot()获得当前运行进程的快照后,我们可以利用 process32First 函数来获得第一个进程的句柄

```
Private Declare Function Process32First Lib "kernel32" (ByVal hSnapshot As Long, lppe As PROCESSENTRY32) As Long
```

函数功能：Process32Next 是一个进程获取函数,当我们利用函数 CreateToolhelp32Snapshot()获得当前运行进程的快照后,我们可以利用 Process32Next 函数来获得下一个进程的句柄

```
Private Declare Function Process32Next Lib "kernel32" (ByVal hSnapshot As Long, lppe As PROCESSENTRY32) As Long
```

’ 函数功能：CreateToolhelp32Snapshot 函数通过获取进程信息为指定的进程、进程使用的堆[HEAP]、模块[MODULE]、线程[THREAD]建立一个快照[snapshot]。

```
Private Declare Function CreateToolhelp32Snapshot Lib "kernel32" (ByVal dwFlags As Long, ByVal th32ProcessID As Long) As Long
```

’ 函数功能：此函数检索与进程相关联的第一个模块的信息

```
Private Declare Function Module32First Lib "kernel32" (ByVal hSnapshot As Long, lpme As MODULEENTRY32) As Long
```

’ 函数功能：此函数与 Module32First 函数结合使用，遍历进程的各个模块

```
Private Declare Function Module32Next Lib "kernel32" (ByVal hSnapshot As Long, lpme As MODULEENTRY32) As Long
```

’ 函数功能：CloseHandle 包括文件、文件映射、进程、线程、安全和同步对象等。涉及文件处理时，这个函数通常与 vb 的 close 命令相似。应尽可能的使用 close，因为它支持 vb 的差错控制。

```
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long
```

’ 函数功能：GetWindowLong 是一个 Windows API 函数。该函数获得有关指定窗口的信息，函数也获得在额外窗口内存中指定偏移位地址的 32 位度整型值。

```
Private Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" (ByVal hWnd As Long, ByVal nIndex As Long) As Long
```

’ 函数功能：SetWindowLong 是一个 Windows API 函数。该函数用来改变指定窗口的属性。函数也将指定的一个 32

位值设置在窗口的额外存储空间的指定偏移位置。

```
Private Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hWnd As Long, ByVal  
nIndex As Long, ByVal dwNewLong As Long) As Long
```

```
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hWnd As Long, ByVal  
lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As String,  
ByVal nShowCmd As Long) As Long
```

```
Private Declare Function SetForegroundWindow Lib "user32" (ByVal hWnd As Long) As Long
```

’ 函数说明：函数说明：**RegisterWindowMessage** 函数定义一个新的窗口消息，保证该消息在系统范围内是唯一的。通常调用 **SendMessage** 或者 **PostMessage** 函数时，可以使用该函数返回的消息值。

```
Private Declare Function RegisterWindowMessage Lib "user32" Alias "RegisterWindowMessageA" (ByVal  
lpString As String) As Long
```

’ 函数功能：执行挂起一段时间

```
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
```

’ 函数说明：**FindWindow** 这个函数检索处理顶级窗口的类名和窗口名称匹配指定的字符串。这个函数不搜索子窗口。**lpClassName** 参数指向类名，**lpWindowName** 指向窗口名，如果有指定的类名和窗口的名字则表示成功返回一个窗口的句柄。否则返回零。

```
Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName As String, ByVal  
lpWindowName As String) As Long
```

’ 函数功能：获取指定文件路径的长路径形式。该 API 不适合对畸形文件夹进行操作。

```
Private Declare Function GetLongPathName Lib "kernel32" Alias "GetLongPathNameA" (ByVal lpszShortPath  
As String, ByVal lpszLongPath As String, ByVal cchBuffer As Long) As Long
```

’ 函数说明：函数 **GetShortPathName** 获取指定路径的短路径形式。该 API 不适合对畸形文件夹进行操作。

```
Private Declare Function GetShortPathName Lib "kernel32" (ByVal lpszLongPath As String, ByVal  
lpszShortPath As String, ByVal cchBuffer As Long) As Long
```

’ 常量定义

```
Private Const MAX_PATH As Long = 260
```

```
Private Const DRIVE_REMOVABLE As Long = 2
```

```
Private Const DRIVE_FIXED As Long = 3
```

```
Private Const DRIVE_REMOTE As Long = 4
```

```
Private Const DRIVE_CDROM As Long = 5
```

```
Private Const DRIVE_RAMDISK As Long = 6
```

```
Private Const TH32CS_SNAPPROCESS = &H2&
```

```
Private Const TH32CS_SNAPMODULE = &H8
```

```
Private Const TH32CS_SNAPHEAPLIST = &H1
```

```
Private Const TH32CS_SNAPTHREAD = &H4
```

```
Private Const TH32CS_SNAPALL = (TH32CS_SNAPHEAPLIST Or TH32CS_SNAPPROCESS Or TH32CS_SNAPTHREAD Or  
TH32CS_SNAPMODULE)
```

```
Private Const FILE_ATTRIBUTE_DIRECTORY = &H10
```

’ 自定义类型

```
Private Type FILETIME
```

```
dwLowDateTime As Long
```

```
dwHighDateTime As Long
```

```
End Type
```

```

Private Type WIN32_FIND_DATA
dwFileAttributes As Long
ftCreationTime As FILETIME
ftLastAccessTime As FILETIME
ftLastWriteTime As FILETIME
nFileSizeHigh As Long
nFileSizeLow As Long
dwReserved0 As Long
dwReserved1 As Long
cFileName As String * MAX_PATH
cAlternate As String * 14
End Type

```

’ 进程信息自定义类型

```

Private Type PROCESSENTRY32
    dwSize As Long
    cntUsage As Long
    th32ProcessID As Long
    th32DefaultHeapID As Long
    th32ModuleID As Long
    cntThreads As Long
    th32ParentProcessID As Long
    pcPriClassBase As Long
    dwFlags As Long
    szExeFile As String * 260
End Type

```

’ 模块信息自定义类型

```

Private Type MODULEENTRY32
dwSize As Long
th32ModuleID As Long
th32ProcessID As Long
GblcntUsage As Long
ProccntUsage As Long
modBaseAddr As Long
modBaseSize As Long
hModule As Long
szModule As String * 255
szExePath As String * 255
End Type

```

’ 消息复制自定义类型

```

Private Type COPYDATASTRUCT
dwData As Long
cbData As Long
lpData As Long
End Type

```

’ 停止扫描标志

```

Dim bStopScanFlag As Boolean

```

```
Dim lScannedFiles As Long
Dim lTrojanFiles As Long
Dim dScanStartTime As Date
```

#### 停止扫描

```
Private Sub CommandStop_Click()

' 停止扫描标志
bStopScanFlag = True
End Sub
```

#### 窗体启动函数

```
Private Sub Form_Load()

' 右键扫描功能先关闭
TimerRightClickScan.Enabled = False

' 禁用 OLE 拖放
ImageCustomerScan(0).OLEDropMode = 0
ImageCustomerScan(1).OLEDropMode = 0

' 判断是否有命令行参数
If Command <> "" Then

' 如果是文件并且存在, 说明是右键扫描文件
If Dir(Command) <> "" Or Dir(Command, vbDirectory) <> "" Then
Dim sRightClickScanFile As String

' 右键扫描记录文件
If Right(App.Path, 1) = "\" Then
sRightClickScanFile = App.Path & "RCS.ini"
Else
sRightClickScanFile = App.Path & "\RCS.ini"
End If
WaitForWrite:

' 同时读写的标识
Dim lWriteReadFlag As Long
lWriteReadFlag = ReadIni(sRightClickScanFile, "Normal", "WRFlag")
If lWriteReadFlag = 0 Then

' 防止同时读写的标识
Call WriteIni(sRightClickScanFile, "Normal", "WRFlag", 1)

' 记录数
Dim lPreCount As Long
lPreCount = ReadIni(sRightClickScanFile, "Normal", "RC")
```

```

' 增加记录数
Call WriteIni(sRightClickScanFile, "Normal", "RC", lPreCount + 1)

' 增加记录
Call WriteIni(sRightClickScanFile, "Normal", lPreCount + 1, Command)

' 防止同时读写的标识
Call WriteIni(sRightClickScanFile, "Normal", "WRFlag", 0)
Else

' 等待变为可写
DoEvents
Sleep 1
GoTo WaitForWrite
End If
End If
End If

' 防止软件自身重复运行
If App.PrevInstance Then
End
End If
Dim sSettingsFile As String

' 软件设置记录文件
If Right(App.Path, 1) = "\" Then
sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\"Settings.ini"
End If
DoEvents

' 设置是否允许关闭窗体，此时此设置为 False，也就是此时（初始化阶段）不允许退出窗体
bEnableUnloadForm = False

' 读取皮肤设置字符串，在各窗体加载界面皮肤时，都会使用如下的方式：
'.Picture = LoadPicture(App.Path & "\"Skin\" & sSkin & "\"RightClickScanResult.bmp")
' 在上面一行代码中，sSkin 变量保存的是皮肤路径变量，有了此路径变量，才能合成完整的皮肤文件路径
sSkin = ReadIni(sSettingsFile, "Normal", "Skin")
DoEvents

' 初始载入各界面的皮肤
ReSkinAll

' 读取自我保护设置
Dim lSafeGuard As Long
lSafeGuard = ReadIni(sSettingsFile, "Normal", "SafeGuard")
DoEvents
If lSafeGuard = 1 Then

' 如果自我保护设置开启，则开启自我保护功能，防止程序被非法结束，SafeGuard 函数会在后面的章节中介绍

```

```

Call SafeGuard(True)
End If
DoEvents

' 自启动设置
Dim lAutorun As Long
lAutorun = ReadIni(sSettingsFile, "Normal", "AutoRun")
If lAutorun = 1 Then
SetAutoRun
Else
StopAutoRun
End If
DoEvents

' 在任务管理器应用程序中不可见
App.TaskVisible = False

' 提升权限
GetMorePrivilege
DoEvents

' 向系统托盘区添加托盘图标
AddTrayIcon
DoEvents

' 注册重建托盘图标消息，注册此消息后，当系统发生错误任务栏重建时可以自由重新添加图标到任务栏托盘区
WM_TASKBARCREATED = RegisterWindowMessage("TaskbarCreated")

' 使软件使用最少内存
MiniUseMemory
DoEvents
Dim i As Long
For i = 0 To 2

' 初始化最小化按钮位置
With ImageMin(i)
.Left = 8640
.Top = 0
End With

' 初始化关闭按钮位置
With ImageExit(i)
.Left = 9105
.Top = 0
End With

' 初始化换肤按钮位置
With ImageSkin(i)
.Left = 8040
.Top = 0
End With

```

Next

' 设置用图片模拟的选项卡控件：安全状态的可见状态

```
ImageSecurityState(0).Visible = True  
ImageSecurityState(1).Visible = False
```

' 设置用图片模拟的选项卡控件：扫描结果的可见状态

```
ImageScanResult(0).Visible = False  
ImageScanResult(1).Visible = True
```

' 设置用图片模拟的选项卡控件：防护记录的可见状态

```
ImageShield(0).Visible = False  
ImageShield(1).Visible = True
```

' 设置用图片模拟的大扫描按钮：快速扫描的可见状态

```
ImageMemoryScan(0).Visible = True  
ImageMemoryScan(1).Visible = False
```

' 设置用图片模拟的大扫描按钮：全盘扫描的可见状态

```
ImageFullDiskScan(0).Visible = True  
ImageFullDiskScan(1).Visible = False
```

' 设置用图片模拟的大扫描按钮：自定义扫描的可见状态

```
ImageCustomerScan(0).Visible = True  
ImageCustomerScan(1).Visible = False
```

' 设置最小化按钮可见状态

```
ImageMin(0).Visible = True  
ImageMin(1).Visible = False  
ImageMin(2).Visible = False
```

' 设置关闭按钮可见状态

```
ImageExit(0).Visible = True  
ImageExit(1).Visible = False  
ImageExit(2).Visible = False
```

' 设置换肤按钮可见状态

```
ImageSkin(0).Visible = True  
ImageSkin(1).Visible = False  
ImageSkin(2).Visible = False
```

' 扫描按钮与状态背景 Picture 控件可见状态

```
PictureScanBottoms.Visible = True  
PictureScanState.Visible = False
```

' 上次扫描用时变量

```
Dim sLascScanUsedTime As String
```

' 上次扫描用时

```
sLascScanUsedTime = ReadIni(sSettingsFile, "History", "LastScanUsedTime")
```

```

' 显示上次扫描时间
LabelLastScanUsedTime.Caption = sLastScanUsedTime

' 上次扫描的文件数量
LabelLastScanFiles.Caption = ReadIni(sSettingsFile, "History", "LastScanFiles") & "个"

' 上次扫描检测到的病毒数量
LabelLastDetectedSpywares.Caption = ReadIni(sSettingsFile, "History", "LastDetectedSpywares") & "个"

' 控制上面几个显示内容的位置
LabelLastScanUsedTimeTitle.Left = 120
LabelLastScanUsedTime.Left = LabelLastScanUsedTimeTitle.Left + LabelLastScanUsedTimeTitle.Width + 20
LabelLastScanFilesTitle.Left = LabelLastScanUsedTime.Left + LabelLastScanUsedTime.Width + 200
LabelLastScanFiles.Left = LabelLastScanFilesTitle.Left + LabelLastScanFilesTitle.Width + 20
LabelLastDetectedSpywaresTitle.Left = LabelLastScanFiles.Left + LabelLastScanFiles.Width + 200
LabelLastDetectedSpywares.Left = LabelLastDetectedSpywaresTitle.Left + LabelLastDetectedSpywaresTitle.Width + 20

' 读取升级设置
Dim lAutoUpdate As Long
lAutoUpdate = ReadIni(sSettingsFile, "Update", "AutoUpdate")

' 判断自动更新是否开启
If lAutoUpdate = 1 Then
LabelAutoUpdateState.Caption = "自动升级：已启用"
ImageAutoUpdate(0).Visible = False
ImageAutoUpdate(1).Visible = True
Else
LabelAutoUpdateState.Caption = "自动升级：已禁用"
ImageAutoUpdate(0).Visible = True
ImageAutoUpdate(1).Visible = False
End If

' 云安全启用状态
Dim lCloudSecurity As Long
lCloudSecurity = ReadIni(sSettingsFile, "Shield", "CloudSecurity")
If lCloudSecurity = 1 Then
LabelCloud.Caption = "云安全防护：已启用"
ImageCloud(0).Visible = False
ImageCloud(1).Visible = True
Else
LabelCloud.Caption = "云安全防护：未启用"
ImageCloud(0).Visible = True
ImageCloud(1).Visible = False
End If

' 防护级别
Dim lShieldLevel As Boolean
lShieldLevel = ReadIni(sSettingsFile, "Shield", "NormalLevel")
If lShieldLevel = True Then
LabelShieldLevel.Caption = "系统防护级别：正常"

```



```

ImageShieldLevel(0).Visible = False
ImageShieldLevel(1).Visible = True
Else
LabelShieldLevel.Caption = "系统防护级别：严格"
ImageShieldLevel(0).Visible = True
ImageShieldLevel(1).Visible = False
End If

' 开机自启动
If lAutorun = 1 Then
LabelAutorun.Caption = "开机自启动：已启用"
ImageAutoRun(0).Visible = False
ImageAutoRun(1).Visible = True
Else
LabelAutorun.Caption = "开机自启动：未启用"
ImageAutoRun(0).Visible = True
ImageAutoRun(1).Visible = False
End If

' 自动最小化
Dim lAutomin As Long
lAutomin = ReadIni(sSettingsFile, "Normal", "AutoTray")
If lAutomin = 1 Then
LabelAutoMin.Caption = "开机最小化：已启用"
ImageAutoMin(0).Visible = False
ImageAutoMin(1).Visible = True
Else
LabelAutoMin.Caption = "开机最小化：未启用"
ImageAutoMin(0).Visible = True
ImageAutoMin(1).Visible = False
End If

' 自我保护
If lSafeGuard = 1 Then
LabelShieldSelf.Caption = "自我保护：已启用"
ImageShieldSelf(0).Visible = False
ImageShieldSelf(1).Visible = True
Else
LabelShieldSelf.Caption = "自我保护：未启用"
ImageShieldSelf(0).Visible = True
ImageShieldSelf(1).Visible = False
End If

' 右键菜单
Dim lRightClickMenu As Long
lRightClickMenu = ReadIni(sSettingsFile, "Normal", "RightClickMenu")
If lRightClickMenu = 1 Then
LabelRightClickMenu.Caption = "关联右键菜单：已启用"
ImageRightClickMenu(0).Visible = False
ImageRightClickMenu(1).Visible = True
Else

```

```

LabelRightClickMenu.Caption = "关联右键菜单：未启用"
ImageRightClickMenu(0).Visible = True
ImageRightClickMenu(1).Visible = False
End If

'发现病毒时自动清除
Dim bClearVirus As Boolean
bClearVirus = ReadIni(sSettingsFile, "Scan", "ClearVirus")

'发现病毒时提示
Dim bAlertVirus As Boolean
bAlertVirus = ReadIni(sSettingsFile, "Scan", "AlertVirus")

'发现病毒时忽略
Dim bIgnoreVirus As Boolean
bIgnoreVirus = ReadIni(sSettingsFile, "Scan", "IgnoreVirus ")

'发现病毒时隔离
Dim bQuarantineVirus As Boolean
bQuarantineVirus = ReadIni(sSettingsFile, "Scan", "QuarantineVirus")
If bClearVirus = True Then
LabelActionWhenDetectVirus.Caption = "检测到病毒处理方式：自动清除"
End If
If bAlertVirus = True Then
LabelActionWhenDetectVirus.Caption = "检测到病毒处理方式：询问"
End If
If bIgnoreVirus = True Then
LabelActionWhenDetectVirus.Caption = "检测到病毒处理方式：忽略"
End If
If bQuarantineVirus = True Then
LabelActionWhenDetectVirus.Caption = "检测到病毒处理方式：隔离"
End If
PictureScanResult.Left = PictureSecurityState.Left
PictureScanResult.Top = PictureSecurityState.Top
PictureShield.Left = PictureSecurityState.Left
PictureShield.Top = PictureSecurityState.Top

'安全状态
PictureSecurityState.Visible = True

'扫描结果
PictureScanResult.Visible = False

'防护记录
PictureShield.Visible = False

'子类化窗体消息
lSubClassOldAddress = GetWindowLong(FormScan.hWnd, -4)
SetWindowLong hWnd, -4, AddressOf SubClassMessage

'自动最小化

```

```

Dim lAutoTray As Long
lAutoTray = ReadIni(sSettingsFile, "Normal", "AutoTray")
If lAutoTray = 1 Then
Me.Hide
Else

' 初始化位置
With Me
.Top = (Screen.Height - .Height) / 2
.Left = (Screen.Width - .Width) / 2
.Show
End With
End If

' 设置鼠标形状为漏斗形状，因为接下来要加载病毒库，消耗时间较长，用漏斗形表示后台正在工作
MousePointer = 11

' 令扫描功能暂时不可用，同因是因为要加载病毒库，病毒库未加载完成时，病毒特征码尚未加载到内容，不能进行扫描操作
ImageMemoryScan(0).Enabled = False
ImageMemoryScan(1).Enabled = False
ImageFullDiskScan(0).Enabled = False
ImageFullDiskScan(1).Enabled = False
ImageCustomerScan(0).Enabled = False
ImageCustomerScan(1).Enabled = False

' 加载特征码
LoadSignatures

' 病毒库加载完成后，将鼠标形状设置回正常状态
MousePointer = 0

' 令三个扫描功能都可以使用
ImageMemoryScan(0).Enabled = True
ImageMemoryScan(1).Enabled = True
ImageFullDiskScan(0).Enabled = True
ImageFullDiskScan(1).Enabled = True
ImageCustomerScan(0).Enabled = True
ImageCustomerScan(1).Enabled = True
DoEvents

' 启动主动防御
ActiveDefense (True)
DoEvents

' 右键扫描功能开启
TimerRightClickScan.Enabled = True

' 启用 OLE 拖放，OLE 拖放设置到了自定义扫描按钮上，这样当将文件或文件夹选中并拖放到界面中的自定义按钮上时，可以快捷的扫描被选中的文件或文件夹
ImageCustomerScan(0).OLEDropMode = 1

```

```
ImageCustomerScan(1).OLEDropMode = 1
End Sub
```

#### 鼠标在窗体中按下

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

' 判断是否按下鼠标左键
If Button = vbLeftButton Then

' 为当前的应用程序释放鼠标捕获
ReleaseCapture

' 移动窗体
SendMessage Me.hWnd, &HA1, 2, 0
End If
End Sub
```

#### 鼠标在窗体中移动

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

' 最小化按钮
ImageMin(0).Visible = True
ImageMin(1).Visible = False
ImageMin(2).Visible = False

' 关闭按钮
ImageExit(0).Visible = True
ImageExit(1).Visible = False
ImageExit(2).Visible = False

' 换肤按钮
ImageSkin(0).Visible = True
ImageSkin(1).Visible = False
ImageSkin(2).Visible = False

' 快速扫描按钮状态
ImageMemoryScan(0).Visible = True
ImageMemoryScan(1).Visible = False

' 全局扫描按钮状态
ImageFullDiskScan(0).Visible = True
ImageFullDiskScan(1).Visible = False

' 自定义扫描按钮状态
ImageCustomerScan(0).Visible = True
ImageCustomerScan(1).Visible = False
End Sub
```

## 窗体卸载函数

```
Private Sub Form_Unload(Cancel As Integer)
```

’判断是否可以退出，bEnableUnloadForm 变量是是否可以退出程序的标识，为 True 时可以退出程序，False 时不能退出。一般操作下，点击右方角的退出按钮时，并不退出程序，只是隐藏到后台运行，这是为了防止误操作不小心退出杀毒软件。当点击托盘区的右键菜单中的“退出”菜单时，会改变此标识变量，使程序真正退出。

```
If bEnableUnloadForm = False Then
```

’取消窗体卸载

```
Cancel = 1
```

’隐藏，使窗体不可见

```
Me.Hide
```

```
End If
```

```
End Sub
```

## 自定义扫描

```
Private Sub ImageCustomerScan_Click(Index As Integer)
```

’退出菜单不可用（扫描过程中不能退出程序）

```
FormMenu.m_Exit.Enabled = False
```

’选择目录后是否进行扫描

```
bDoCustmerScan = True
```

’选择目录

```
FormSelectFolders.Show vbModal
```

’为 False 表示用户选择目录时进行了关闭，不扫描，直接退出

```
If bDoCustmerScan = False Then
```

’退出菜单可用

```
FormMenu.m_Exit.Enabled = True
```

```
Exit Sub
```

```
End If
```

’病毒列表显示，因为是进行扫描，所以相应的要将界面下方选项卡切换为扫描结果

```
ImageScanResult_Click (0)
```

’扫描按钮与状态可见状态

```
PictureScanBottoms.Visible = False
```

```
PictureScanState.Visible = True
```

```
Dim sSettingsFile As String
```

’软件设置记录文件

```
If Right(App.Path, 1) = "\" Then
```

```
sSettingsFile = App.Path & "Settings.ini"
```

```
Else
```

```
sSettingsFile = App.Path & "\\Settings.ini"
```

```

End If

' 读取扫描时减少 CPU 占用设置
Dim lLowCPU As Long
lLowCPU = ReadIni(sSettingsFile, "Scan", "CheckLowCPU")

' 判断设置值是否为 1，如果为 1 则要设置较低的 CPU 优先级
If lLowCPU = 1 Then

' 设置当前进程优先级为低于标准，12384 为低于标准的常量。
' 为什么要设置进程优先级为低于标准？这是因为，在进行病毒扫描时，会占用大量的系统资源，如果程序合用正常
CPU，有可能会使系统变的很卡，甚至在配置太差的机器中会引起系统假死。
Call SetProcessPriority(12384)
End If

' 清除前次扫描结果
ListViewScanResult.ListItems.Clear

' 停止扫描标志，当要停止扫描时，会通过外部操作，置此变量的值为 True，当检测到此变量变为 True 时，程序会
中止扫描过程
bStopScanFlag = False

' 初始化扫描文件数
lScannedFiles = 0

' 初始化病毒数
lTrojanFiles = 0
LabelScannedFileCount.Caption = 0
LabelTrojanFileCount.Caption = 0

' 清空扫描列表（这个控件不可见，是一个 List 控件，用于存放待扫描的目录）
ListFolders.Clear

' 如果自定义扫描控件中没有节点就退出
If FormSelectFolders.TreeViewScanTarget.Nodes.Count = 0 Then
Exit Sub
End If

' 文件夹
Dim sFolder As String

' 文件夹路径长度
Dim lFolderLen As Long
Dim i As Long
Dim j As Long
For i = 1 To FormSelectFolders.TreeViewScanTarget.Nodes.Count

' 判断是否是选中的节点
If FormSelectFolders.TreeViewScanTarget.Nodes(i).Checked = True Then

' 排除根节点，这里判断字符串是否等于“我的电脑”是因为在初始化阶段将根结点赋值为了此字符串

```

```

If FormSelectFolders.TreeViewScanTarget.Nodes(i).FullPath <> "我的电脑" Then

' 判断最后一个字符是否是 "\", 并以此取文件夹路径
If Right(FormSelectFolders.TreeViewScanTarget.Nodes(i).FullPath, 1) <> "\" Then
sFolder=Right(FormSelectFolders.TreeViewScanTarget.Nodes(i).FullPath,
Len(FormSelectFolders.TreeViewScanTarget.Nodes(i).FullPath) - 5) & "\"
Else
sFolder=Right(FormSelectFolders.TreeViewScanTarget.Nodes(i).FullPath,
Len(FormSelectFolders.TreeViewScanTarget.Nodes(i).FullPath) - 5)
End If

' 检查是否已有文件夹的父目录
For j = 0 To ListFolders.ListCount - 1
lFolderLen = Len(ListFolders.List(j))
If Len(ListFolders.List(j)) < Len(sFolder) Then
If ListFolders.List(j) = Left(sFolder, lFolderLen) Then

' 如果父目录在扫描列表中, 则移除
ListFolders.RemoveItem j
Exit For
End If
End If
Next j

' 给扫描列表添加新的扫描目录
ListFolders.AddItem sFolder
End If
End If
Next i

' 扫描开始时间
dScanStartTime = Time
Do While ListFolders.ListCount <> 0
DoEvents

' 开始扫描
ScanFolder ListFolders.List(0)

' 第一行搜索完成后删除第一行
ListFolders.RemoveItem 0

' 用户停止扫描 (通过判断上面讲过的扫描停止标识)
If bStopScanFlag = True Then
Exit Do
End If
Loop

' 扫描结束
Call ShowAndRecordScanResult
If bStopScanFlag = True Then
With FormScanFinish

```

```

' 由用户点击了“停止扫描”造成的扫描结果
.LabelTitle.Caption = "扫描中止"
.LabelMessage.Caption = "已扫描 " & lScanedFiles & " 个文件, 发现 " & lTrojanFiles & " 个病毒。"
.Show vbModal
End With
Else
With FormScanFinish

' 正常完成扫描时的扫描结果
.LabelTitle.Caption = "扫描完成"
.LabelMessage.Caption = "已扫描 " & lScanedFiles & " 个文件, 发现 " & lTrojanFiles & " 个病毒。"
.Show vbModal
End With
End If

' 扫描按钮与状态可见状态
PictureScanBottoms.Visible = True
PictureScanState.Visible = False

' 设置当前进程优先级为标准
Call SetProcessPriority(32)
LabelNowScaning.Caption = "扫描结束。"

' 自动选中扫描出的病毒
If ListViewScanResult.ListItems.Count > 0 Then
Dim k As Long
For k = 1 To ListViewScanResult.ListItems.Count
ListViewScanResult.ListItems(k).Checked = True
Next k
End If

' 退出菜单可用
FormMenu.m_Exit.Enabled = True

' 使软件使用最少内存
MiniUseMemory
End Sub

```

#### 鼠标在自定义扫描按钮上移动

```

Private Sub ImageCustomerScan_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single,
Y As Single)

' 设置快速扫描按钮可见状态
ImageMemoryScan(0).Visible = True
ImageMemoryScan(1).Visible = False

' 设置全盘扫描按钮可见状态
ImageFullDiskScan(0).Visible = True

```



```
ImageFullDiskScan(1).Visible = False
```

```
' 设置自定义扫描按钮可见状态
```

```
ImageCustomerScan(0).Visible = False
```

```
ImageCustomerScan(1).Visible = True
```

```
End Sub
```

```
' 设置快速扫描按钮可见状态拖放文件到窗体中的自定义扫描按键上时，扫描此文件
```

```
Private Sub ImageCustomerScan_OLEDragDrop(Index As Integer, Data As DataObject, Effect As Long, Button  
As Integer, Shift As Integer, X As Single, Y As Single)
```

```
' 退出菜单不可用
```

```
FormMenu.m_Exit.Enabled = False
```

```
' 病毒列表显示
```

```
ImageScanResult_Click (0)
```

```
' 扫描按钮与状态可见状态
```

```
PictureScanBottoms.Visible = False
```

```
PictureScanState.Visible = True
```

```
Dim sSettingsFile As String
```

```
' 软件设置记录文件
```

```
If Right(App.Path, 1) = "\" Then
```

```
sSettingsFile = App.Path & "Settings.ini"
```

```
Else
```

```
sSettingsFile = App.Path & "\Settings.ini"
```

```
End If
```

```
' 读取扫描时减少 CPU 占用设置
```

```
Dim lLowCPU As Long
```

```
lLowCPU = ReadIni(sSettingsFile, "Scan", "CheckLowCPU")
```

```
If lLowCPU = 1 Then
```

```
' 设置当前进程优先级为低于标准
```

```
Call SetProcessPriority(16384)
```

```
End If
```

```
' 清除前次扫描结果
```

```
ListViewScanResult.ListItems.Clear
```

```
' 停止扫描标志
```

```
bStopScanFlag = False
```

```
' 扫描文件数
```

```
lScannedFiles = 0
```

```
' 病毒数
```

```
lTrojanFiles = 0
```

```

LabelScannedFileCount.Caption = 0
LabelTrojanFileCount.Caption = 0

' 扫描开始时间
dScanStartTime = Time
Dim sFilename
For Each sFilename In Data.Files
DoEvents
If GetAttr(sFilename) And vbDirectory Then

' 添加要扫描的目录
If Right(sFilename, 1) = "\" Then

' 目录
ListFolders.AddItem sFilename
Else

' 目录
ListFolders.AddItem sFilename & "\"
End If
Do While ListFolders.ListCount <> 0
DoEvents

' 开始扫描
OleScanFolder ListFolders.List(0)

' 第一行搜索完成后删除第一行
ListFolders.RemoveItem 0

' 用户停止扫描
If bStopScanFlag = True Then
Exit Do
End If
Loop
Else
DoEvents

' 扫描文件
Call OleScanFile(sFilename)
End If
Next

' 扫描结束
Call ShowAndRecordScanResult
If bStopScanFlag = True Then
With FormScanFinish
.LabelTitle.Caption = "扫描中止"
.LabelMessage.Caption = "已扫描 " & lScannedFiles & " 个文件, 发现 " & lTrojanFiles & " 个病毒。"
.Show vbModal
End With
Else

```

```

With FormScanFinish
.LabelTitle.Caption = "扫描完成"
.LabelMessage.Caption = "已扫描 " & lScanedFiles & " 个文件, 发现 " & lTrojanFiles & " 个病毒。"
.Show vbModal
End With
End If

' 扫描按钮与状态可见状态
PictureScanBottoms.Visible = True
PictureScanState.Visible = False

' 设置当前进程优先级为标准
Call SetProcessPriority(32)
LabelNowScaning.Caption = "扫描结束。"

' 选中病毒
If ListViewScanResult.ListItems.Count > 0 Then
Dim k As Long
For k = 1 To ListViewScanResult.ListItems.Count
ListViewScanResult.ListItems(k).Checked = True
Next k
End If

' 退出菜单可用
FormMenu.m_Exit.Enabled = True

' 使软件使用最少内存
MiniUseMemory
End Sub

```

#### 退出窗体

```

Private Sub ImageExit_Click(Index As Integer)
Me.Hide
End Sub

```

#### 全盘扫描

```

Private Sub ImageFullDiskScan_Click(Index As Integer)

```

```

' 退出菜单不可用
FormMenu.m_Exit.Enabled = False

```

```

' 病毒列表显示
ImageScanResult_Click (0)

```

```

' 扫描按钮与状态可见状态
PictureScanBottoms.Visible = False
PictureScanState.Visible = True
Dim sSettingsFile As String

```

```

' 软件设置记录文件

```

```

If Right(App.Path, 1) = "\" Then
sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\\Settings.ini"
End If

' 读取扫描时减少 CPU 占用设置
Dim lLowCPU As Long
lLowCPU = ReadIni(sSettingsFile, "Scan", "CheckLowCPU")
If lLowCPU = 1 Then

' 设置当前进程优先级为低于标准
Call SetProcessPriority(16384)
End If

' 清除前次扫描结果
ListViewScanResult.ListItems.Clear

' 停止扫描标志
bStopScanFlag = False

' 扫描文件数
lScannedFiles = 0

' 病毒数
lTrojanFiles = 0
LabelScannedFileCount.Caption = 0
LabelTrojanFileCount.Caption = 0

' 清空扫描列表
ListFolders.Clear
Dim sDrive As String
sDrive = String(256, Chr(0))
Dim sDriveID As String

' 取得磁盘串
Call GetLogicalDriveStrings(256, sDrive)

' 返回光盘盘符到数组, 注意这里 step 是 4
Dim i As Long
For i = 1 To 100 Step 4
sDriveID = Mid(sDrive, i, 3)
If sDriveID = Chr(0) & Chr(0) & Chr(0) Then

' 没有盘符, 即时退出循环
Exit For
Else

' 添加盘符
ListFolders.AddItem Left(sDriveID, 2)
End If

```

```

Next i

' 扫描开始时间
dScanStartTime = Time
Do While ListFolders.ListCount <> 0
DoEvents

' 开始扫描
ScanFolder ListFolders.List(0) & "\"

' 第一行搜索完成后删除第一行
ListFolders.RemoveItem 0

' 用户停止扫描
If bStopScanFlag = True Then
Exit Do
End If
Loop

' 扫描结束
Call ShowAndRecordScanResult
If bStopScanFlag = True Then

' 显示扫描结果
With FormScanFinish
.LabelTitle.Caption = "扫描中止"
.LabelMessage.Caption = "已扫描 " & lScanedFiles & " 个文件，发现 " & lTrojanFiles & " 个病毒。"
.Show vbModal
End With
Else
With FormScanFinish
.LabelTitle.Caption = "扫描完成"
.LabelMessage.Caption = "已扫描 " & lScanedFiles & " 个文件，发现 " & lTrojanFiles & " 个病毒。"
.Show vbModal
End With
End If

' 设置当前进程优先级为标准
Call SetProcessPriority(32)

' 扫描按钮与状态可见状态
PictureScanBottoms.Visible = True
PictureScanState.Visible = False
LabelNowScaning.Caption = "扫描结束。"

' 选中病毒
If ListViewScanResult.ListItems.Count > 0 Then
Dim k As Long
For k = 1 To ListViewScanResult.ListItems.Count
ListViewScanResult.ListItems(k).Checked = True
Next k

```

```
End If
```

’ 使退出菜单可用。扫描进行中时，不允许同步进行其它菜单操作，比如可能重复开启扫描，那样会引起软件出错

```
FormMenu.m_Exit.Enabled = True
```

’ 使软件使用最少内存

```
MiniUseMemory
```

```
End Sub
```

#### 鼠标在全盘扫描按钮上移动

```
Private Sub ImageFullDiskScan_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
ImageMemoryScan(0).Visible = True
```

```
ImageMemoryScan(1).Visible = False
```

```
ImageFullDiskScan(0).Visible = False
```

```
ImageFullDiskScan(1).Visible = True
```

```
ImageCustomerScan(0).Visible = True
```

```
ImageCustomerScan(1).Visible = False
```

```
End Sub
```

#### ’ 进行快速扫描

```
Private Sub ImageMemoryScan_Click(Index As Integer)
```

’ 扫描前使退出菜单不可用，也就是扫描时不允许退出程序

```
FormMenu.m_Exit.Enabled = False
```

’ 使病毒列表控件可见，用于实时显示扫描到的病毒

```
ImageScanResult_Click (0)
```

’ 扫描按钮与状态可见状态

```
PictureScanBottoms.Visible = False
```

```
PictureScanState.Visible = True
```

```
Dim sSettingsFile As String
```

’ 软件设置记录文件

```
If Right(App.Path, 1) = "\" Then
```

```
sSettingsFile = App.Path & "Settings.ini"
```

```
Else
```

```
sSettingsFile = App.Path & "\\Settings.ini"
```

```
End If
```

’ 读取扫描时减少 CPU 占用设置

```
Dim lLowCPU As Long
```

```
lLowCPU = ReadIni(sSettingsFile, "Scan", "CheckLowCPU")
```

```
If lLowCPU = 1 Then
```

’ 设置当前进程优先级为低于标准，防止扫描时系统变的很卡

```
Call SetProcessPriority(16384)
```

```

End If

' 清除前次扫描结果
ListViewScanResult.ListItems.Clear

' 停止扫描标志，这是全局变量，在这里先置为 False，当点下停止按钮时，会置为 True，进而停止扫描
bStopScanFlag = False

' 扫描文件数
lScannedFiles = 0

' 病毒数
lTrojanFiles = 0
LabelScannedFileCount.Caption = 0
LabelTrojanFileCount.Caption = 0

' 扫描开始时间
dScanStartTime = Time
Dim lProcess As Long

' 取进系统快照，以便获取进程列表文件进行扫描
lProcess = CreateToolhelp32Snapshot (TH32CS_SNAPPROCESS, 0)

' 如果出错就退出
If lProcess = 0 Then
MsgBox "无法读取内存，请确认有足够的权限", vbInformation
Exit Sub
End If
Dim uProcess As PROCESSENTRY32
uProcess.dwSize = Len(uProcess)

' 枚举进程列表
If Process32First(lProcess, uProcess) Then
Do

' 用户停止扫描
If bStopScanFlag = True Then
Exit Do
End If

' 枚举进程模块
Dim lModule As Long
lModule = CreateToolhelp32Snapshot (TH32CS_SNAPMODULE, uProcess.th32ProcessID)
Dim fAlertForm As New FormAlertVirus
If lModule <> -1 Then
Dim ModEntry As MODULEENTRY32
ModEntry.dwSize = LenB(ModEntry)
ModEntry.szExePath = ""

' 取各模块
If Module32First(lModule, ModEntry) Then

```

```

Do

' 用户停止扫描的标志, 如果为 True, 则停止扫描
If bStopScanFlag = True Then
Exit Do
End If
DoEvents
Dim sFile As String

' 获取文件全路径
sFile = TrimNull(ModEntry.szExePath)

' 获取标识, 判断扫描全部文件或 PE 文件
Dim bScanAllFile As Boolean
bScanAllFile = ReadIni(sSettingsFile, "Scan", "ScanAllFiles")

' 读取发现病毒时自动清除的标志
Dim bClearVirus As Boolean
bClearVirus = ReadIni(sSettingsFile, "Scan", "ClearVirus")

' 读取发现病毒时是否发出提示的标志
Dim bAlertVirus As Boolean
bAlertVirus = ReadIni(sSettingsFile, "Scan", "AlertVirus")

' 读取发现病毒时是否进行忽略操作的标志
Dim bIgnoreVirus As Boolean
bIgnoreVirus = ReadIni(sSettingsFile, "Scan", "IgnoreVirus ")

' 读取发现病毒时是否进行隔离的标志
Dim bQuarantineVirus As Boolean
bQuarantineVirus = ReadIni(sSettingsFile, "Scan", "QuarantineVirus")
If bScanAllFile = True Then

' 扫描文件数
lScannedFiles = lScannedFiles + 1
LabelScannedFileCount.Caption = lScannedFiles

' 计算扫描耗时
LabelTimeUsed.Caption = CDate(Time - dScanStartTime)

' 显示正在扫描的文件, 如果文件长度大于 60, 在界面中显示不完, 则只显示头尾部分, 中间用省略号代替
If Len(sFile) > 60 Then
LabelNowScanning.Caption = Left(sFile, 47) & "... " & Right(sFile, 10)
Else
LabelNowScanning.Caption = sFile
End If

' 调用扫描函数, 开始扫描
Dim sScanResult As String
sScanResult = ScanFile(sFile)

```



```

' 清除病毒返回值
Dim lDeleteFileReturn As Long

' 当前路径
Dim sTempAppPath As String

' 复制文件
Dim sTargetFileName As String

' 复制文件返回值
Dim lCopyFileReturn As Long

' 检查并显示扫描结果
If sScanResult <> "" Then

' 发现病毒数
lTrojanFiles = lTrojanFiles + 1
LabelTrojanFileCount.Caption = lTrojanFiles

' 忽略
If bIgnoreVirus = True Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "未清除"
End With
End If

' 自动清除
If bClearVirus = True Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已清除"
End With
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "清除失败"
End With
End If
End If

```

```

' 隔离
If bQuarantineVirus = True Then
' 确保当前路径包含"\
If Right(App.Path, 1) = "\" Then
sTempAppPath = App.Path
Else
sTempAppPath = App.Path & "\"
End If

' 目标复制文件名
sTargetFileName = sTempAppPath & "quarantine\" & RndChr(8) & ".tmp"

' 复制文件
lCopyFileReturn = CopyFile(sFile, sTargetFileName, 0)

' 复制成功
If lCopyFileReturn <> 0 Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已隔离"
End With

' 隔离记录文件
Call WriteQuarantineFile(sFile, sTargetFileName)
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If

' 复制失败，隔离失败
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If
End If

' 提示

```

```

If bAlertVirus = True Then
SetForegroundWindow FormScan.hWnd
With fAlertForm
.LabelFile = sFile
.LabelVirusName = sScanResult
.Show vbModal
End With
End If
End If
Else

' 判断文件类型
If UCase(Right(sFile, 4)) = UCase(".dll") _
Or UCase(Right(sFile, 4)) = UCase(".exe") _
Or UCase(Right(sFile, 4)) = UCase(".com") _
Or UCase(Right(sFile, 4)) = UCase(".ocx") _
Or UCase(Right(sFile, 4)) = UCase(".scr") _
Then

' 扫描文件数
lScannedFiles = lScannedFiles + 1
LabelScannedFileCount.Caption = lScannedFiles

' 扫描耗时
LabelTimeUsed.Caption = CDate(Time - dScanStartTime)

' 正在扫描的文件
If Len(sFile) > 60 Then
LabelNowScanning.Caption = Left(sFile, 47) & "... " & Right(sFile, 10)
Else
LabelNowScanning.Caption = sFile
End If

' 扫描
sScanResult = ScanFile(sFile)

' 检查并显示扫描结果
If sScanResult <> "" Then

' 发现病毒数
lTrojanFiles = lTrojanFiles + 1
LabelTrojanFileCount.Caption = lTrojanFiles

' 忽略
If bIgnoreVirus = True Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "未清除"
End With

```

```

End If

' 自动清除
If bClearVirus = True Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then

' 向查杀结果控件中添加记录
With ListViewScanResult

' Listview 控件第一列设置为序列
.ListItems.Add , , .ListItems.Count + 1

' Listview 控件第二列显示为文件名
.ListItems(.ListItems.Count).SubItems(1) = sFile

' ListView 控件第三列显示为扫描结果，即病毒名笱
.ListItems(.ListItems.Count).SubItems(2) = sScanResult

' 在此处，ListView 控件第四显示为处理结果：已清除。这是因为上面的分枝中，这里进入的直接是已清除的处理
结果分枝，并且清除成功
.ListItems(.ListItems.Count).SubItems(3) = "已清除"
End With
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "清除失败"
End With
End If
End If

' 隔离
If bQuarantineVirus = True Then

' 确保当前路径包含"\
If Right(App.Path, 1) = "\" Then
sTempAppPath = App.Path
Else
sTempAppPath = App.Path & "\"
End If

' 目标复制文件名
sTargetFileName = sTempAppPath & "quarantine\" & RndChr(8) & ".tmp"

' 复制文件：将原始文件复制到软件目录下 quarantine 下，文件名称是使用 RndChr(8) 函数得到的 8 位随机字符串
为名称
lCopyFileReturn = CopyFile(sFile, sTargetFileName, 0)

```

```

' 复制成功
If lCopyFileReturn <> 0 Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then

' 向扫描结果控件中添加记录
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已隔离"
End With

' 隔离记录文件
Call WriteQuarantineFile(sFile, sTargetFileName)
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If

' 复制失败, 隔离失败
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If
End If

' 判断是否要向用户发出提示信息
If bAlertVirus = True Then
SetForegroundWindow FormScan.hWnd

' 设置提示窗体包含的各种信息
With fAlertForm

' 被隔离的文件名
.LabelFile = sFile

' 病毒名
.LabelVirusName = sScanResult

```

```

' 弹出窗体，显示给用户
.Show vbModal
End With
End If
End If
End If
End If
Loop While Module32Next(lModule, ModEntry)
End If

' 关闭模块枚举句柄
CloseHandle lModule
End If
Loop While Process32Next(lProcess, uProcess)

' 关闭进程枚举句柄
CloseHandle lProcess
End If

' 扫描结束
Call ShowAndRecordScanResult
If bStopScanFlag = True Then
With FormScanFinish
.LabelTitle.Caption = "扫描中止"
.LabelMessage.Caption = "已扫描 " & lScannedFiles & " 个文件，发现 " & lTrojanFiles & " 个病毒。"
.Show vbModal
End With
Else
With FormScanFinish
.LabelTitle.Caption = "扫描完成"
.LabelMessage.Caption = "已扫描 " & lScannedFiles & " 个文件，发现 " & lTrojanFiles & " 个病毒。"
.Show vbModal
End With
End If

' 扫描按钮与状态可见状态
PictureScanBottoms.Visible = True
PictureScanState.Visible = False

' 设置当前进程优先级为标准
Call SetProcessPriority(32)
LabelNowScaning.Caption = "扫描结束。"

' 选中病毒
If ListViewScanResult.ListItems.Count > 0 Then
Dim k As Long
For k = 1 To ListViewScanResult.ListItems.Count
ListViewScanResult.ListItems(k).Checked = True
Next k
End If

```

```
' 退出菜单可用
FormMenu.m_Exit.Enabled = True
```

```
' 使软件使用最少内存
MiniUseMemory
End Sub
```

#### 鼠标在快速扫描按钮上移动

```
Private Sub ImageMemoryScan_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
' 快速扫描按钮状态
ImageMemoryScan(0).Visible = False
ImageMemoryScan(1).Visible = True
```

```
' 全盘扫描按钮状态
ImageFullDiskScan(0).Visible = True
ImageFullDiskScan(1).Visible = False
```

```
' 自定义扫描按钮状态
ImageCustomerScan(0).Visible = True
ImageCustomerScan(1).Visible = False
End Sub
```

#### 鼠标在窗体右上角最小化按钮上移动

```
Private Sub ImageMin_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
' 最小化按钮状态
ImageMin(0).Visible = False
ImageMin(1).Visible = True
ImageMin(2).Visible = False
End Sub
```

#### 鼠标在窗体右上角最小化按钮上按下

```
Private Sub ImageMin_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
' 最小化按钮状态
ImageMin(0).Visible = False
ImageMin(1).Visible = False
ImageMin(2).Visible = True
End Sub
```

#### 鼠标在窗体右上角最小化按钮上抬起

```
Private Sub ImageMin_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
```

Single)

’判断是否在点击最小化按钮

```
If ImageMin(2).Visible = True Then
```

’发送窗体最小化消息

```
SendMessage Me.hWnd, &H112, &HF020&, 0
```

```
End If
```

```
End Sub
```

#### 鼠标在窗体右上角退出按钮上移动

```
Private Sub ImageExit_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

’退出按钮状态

```
ImageExit(0).Visible = False
```

```
ImageExit(1).Visible = True
```

```
ImageExit(2).Visible = False
```

```
End Sub
```

#### 鼠标在窗体右上角退出按钮上按下

```
Private Sub ImageExit_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

’退出按钮状态

```
ImageExit(0).Visible = False
```

```
ImageExit(1).Visible = False
```

```
ImageExit(2).Visible = True
```

```
End Sub
```

#### 鼠标在窗体右上角退出按钮上抬起

```
Private Sub ImageExit_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

’退出点击按钮

```
If ImageExit(2).Visible = True Then
```

```
Me.Hide
```

```
End If
```

```
End Sub
```

#### 扫描目录下所有文件，查找病毒

```
Public Sub ScanFolder(sFolder As String)
```

’搜索用自定义结构

```
Dim uWFD As WIN32_FIND_DATA
```



```

' 文件
Dim sFile As String

' 目录
Dim sDir As String

' 搜索句柄
Dim lSerachHwnd As Long
Dim lSerachNextHwnd As Long

' 特征码扫描结果
Dim sScanResult As String
lSerachHwnd = FindFirstFile(sFolder & "*.*", uWFD)

' 遍历文件及目录
Do
DoEvents

' 停止扫描
If bStopScanFlag = True Then
Exit Do
End If

' 如果是目录
If uWFD.dwFileAttributes And FILE_ATTRIBUTE_DIRECTORY Then

' 判断是否是根目录和上级目录名
If Left(uWFD.cFileName, 1) <> "." And Left(uWFD.cFileName, 2) <> ".." Then

' 取得路径
sDir = sFolder & TrimNull(uWFD.cFileName)

' 添加到待扫描目录
If Right(sDir, 1) = "\" Then
ListFolders.AddItem sDir
Else
ListFolders.AddItem sDir & "\"
End If
End If

' 如果是文件
Else

' 获取文件全路径
sFile = sFolder & TrimNull(uWFD.cFileName)
sFile = Replace(sFile, "\\ ", "\")

' 软件设置记录文件
Dim sSettingsFile As String
If Right(App.Path, 1) = "\" Then

```

```

sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\Settings.ini"
End If

' 扫描全部文件或 PE 文件
Dim bScanAllFile As Boolean
bScanAllFile = ReadIni(sSettingsFile, "Scan", "ScanAllFiles")

' 发现病毒时自动清除
Dim bClearVirus As Boolean
bClearVirus = ReadIni(sSettingsFile, "Scan", "ClearVirus")

' 发现病毒时提示
Dim bAlertVirus As Boolean
bAlertVirus = ReadIni(sSettingsFile, "Scan", "AlertVirus")

' 发现病毒时忽略
Dim bIgnoreVirus As Boolean
bIgnoreVirus = ReadIni(sSettingsFile, "Scan", "IgnoreVirus ")

' 发现病毒时隔离
Dim bQuarantineVirus As Boolean
bQuarantineVirus = ReadIni(sSettingsFile, "Scan", "QuarantineVirus")
Dim fAlertForm As New FormAlertVirus
If bScanAllFile = True Then

' 扫描文件数
lScannedFiles = lScannedFiles + 1
LabelScannedFileCount.Caption = lScannedFiles

' 扫描耗时
LabelTimeUsed.Caption = CDate(Time - dScanStartTime)

' 正在扫描的文件
If Len(sFile) > 60 Then
LabelNowScanning.Caption = Left(sFile, 47) & "... " & Right(sFile, 10)
Else
LabelNowScanning.Caption = sFile
End If

' 调用扫描函数扫描文件
sScanResult = ScanFile(sFile)

' 清除病毒返回值
Dim lDeleteFileReturn As Long

' 当前路径
Dim sTempAppPath As String

' 复制文件

```

```

Dim sTargetFileName As String

' 复制文件返回
Dim lCopyFileReturn As Long

' 检查并显示扫描结果
If sScanResult <> "" Then

' 发现病毒数
lTrojanFiles = lTrojanFiles + 1
LabelTrojanFileCount.Caption = lTrojanFiles

' 忽略
If bIgnoreVirus = True Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "未清除"
End With
End If

' 自动清除
If bClearVirus = True Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已清除"
End With
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "清除失败"
End With
End If
End If

' 隔离
If bQuarantineVirus = True Then

' 确保当前路径包含"\
If Right(App.Path, 1) = "\" Then
sTempAppPath = App.Path
Else

```

```

sTempAppPath = App.Path & "\
End If

' 目标复制文件名
sTargetFileName = sTempAppPath & "quarantine\" & RndChr(8) & ".tmp"

' 复制文件
lCopyFileReturn = CopyFile(sFile, sTargetFileName, 0)

' 复制成功
If lCopyFileReturn <> 0 Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已隔离"
End With

' 隔离记录文件
Call WriteQuarantineFile(sFile, sTargetFileName)
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If

' 复制失败, 隔离失败
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If
End If

' 提示
If bAlertVirus = True Then
SetForegroundWindow FormScan.hWnd
With fAlertForm
.LabelFile = sFile
.LabelVirusName = sScanResult
.Show vbModal

```

```

End With
End If
End If
Else

' 判断文件类型
If UCase(Right(sFile, 4)) = UCase(".dll") _
Or UCase(Right(sFile, 4)) = UCase(".exe") _
Or UCase(Right(sFile, 4)) = UCase(".com") _
Or UCase(Right(sFile, 4)) = UCase(".ocx") _
Or UCase(Right(sFile, 4)) = UCase(".scr") _
Then

' 扫描文件数
lScannedFiles = lScannedFiles + 1
LabelScannedFileCount.Caption = lScannedFiles

' 扫描耗时
LabelTimeUsed.Caption = CDate(Time - dScanStartTime)

' 显示正在扫描的文件
If Len(sFile) > 60 Then
LabelNowScanning.Caption = Left(sFile, 47) & "... " & Right(sFile, 10)
Else
LabelNowScanning.Caption = sFile
End If

' 扫描文件
sScanResult = ScanFile(sFile)

' 检查并显示扫描结果
If sScanResult <> "" Then

' 发现病毒数
lTrojanFiles = lTrojanFiles + 1
LabelTrojanFileCount.Caption = lTrojanFiles

' 忽略
If bIgnoreVirus = True Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "未清除"
End With
End If

' 自动清除
If bClearVirus = True Then

' 清除病毒

```

```

lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已清除"
End With
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "清除失败"
End With
End If
End If

' 隔离
If bQuarantineVirus = True Then

' 确保当前路径包含"\
If Right(App.Path, 1) = "\" Then
sTempAppPath = App.Path
Else
sTempAppPath = App.Path & "\"
End If

' 目标复制文件名
sTargetFileName = sTempAppPath & "quarantine\" & RndChr(8) & ".tmp"

' 复制文件
lCopyFileReturn = CopyFile(sFile, sTargetFileName, 0)

' 复制成功
If lCopyFileReturn <> 0 Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then

' 向扫描结果控件中添加记录
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已隔离"
End With

' 隔离记录文件
Call WriteQuarantineFile(sFile, sTargetFileName)

```

```

Else

' 向扫描结果控件中添加记录
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If

' 复制失败，隔离失败
Else

' 向扫描结果控件中添加记录
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If
End If

' 根据配置设置判断是否发出提示
If bAlertVirus = True Then
SetForegroundWindow FormScan.hWnd

' 设置提示窗体中的信息
With fAlertForm
.LabelFile = sFile
.LabelVirusName = sScanResult

' 弹出提示窗体
.Show vbModal
End With
End If
End If
End If
End If
End If

' 继续遍历文件
lSerachNextHwnd = FindNextFile(lSerachHwnd, uWFD)
Loop While lSerachNextHwnd <> 0

' 关闭搜索句柄
FindClose lSerachHwnd
End Sub

```

#### \*点击选项卡控件: 木马病毒选项卡

```
Private Sub ImageScanResult_Click(Index As Integer)

' 设置安全状态选项卡状态
ImageSecurityState(0).Visible = False
ImageSecurityState(1).Visible = True

' 设置扫描结果选项卡状态
ImageScanResult(0).Visible = True
ImageScanResult(1).Visible = False

' 设置防护记录选项卡状态
ImageShield(0).Visible = False
ImageShield(1).Visible = True

ImageSoftware(0).Visible = False
ImageSoftware(1).Visible = True

' 设置安全状态、扫描结果、防护记录三个选项卡中控件容器的图片控件状态
PictureSecurityState.Visible = False
PictureScanResult.Visible = True
PictureShield.Visible = False
End Sub
```

#### \*点击选项卡控件: 安全状态选项卡

```
Private Sub ImageSecurityState_Click(Index As Integer)

' 设置安全状态选项卡状态
ImageSecurityState(0).Visible = True
ImageSecurityState(1).Visible = False

' 设置扫描结果选项卡状态
ImageScanResult(0).Visible = False
ImageScanResult(1).Visible = True

' 设置防护记录选项卡状态
ImageShield(0).Visible = False
ImageShield(1).Visible = True

ImageSoftware(0).Visible = False
ImageSoftware(1).Visible = True

' 设置安全状态、扫描结果、防护记录三个选项卡中控件容器的图片控件状态
PictureSecurityState.Visible = True
PictureScanResult.Visible = False
PictureShield.Visible = False
End Sub
```



#### ·点击设置按钮

```
Private Sub ImageSetting_Click()
```

```
’ 弹出软件设置窗体
```

```
FormSetting.Show vbModal
```

```
End Sub
```

#### ·点击选项卡控件：防护状态选项卡

```
Private Sub ImageShield_Click(Index As Integer)
```

```
’ 设置安全状态选项卡状态
```

```
ImageSecurityState(0).Visible = False
```

```
ImageSecurityState(1).Visible = True
```

```
’ 设置扫描结果选项卡状态
```

```
ImageScanResult(0).Visible = False
```

```
ImageScanResult(1).Visible = True
```

```
’ 设置防护记录选项卡状态
```

```
ImageShield(0).Visible = True
```

```
ImageShield(1).Visible = False
```

```
ImageSoftware(0).Visible = False
```

```
ImageSoftware(1).Visible = True
```

```
’ 设置安全状态、扫描结果、防护记录三个选项卡中控件容器的图片控件状态
```

```
PictureSecurityState.Visible = False
```

```
PictureScanResult.Visible = False
```

```
PictureShield.Visible = True
```

```
End Sub
```

#### ·换肤

```
Private Sub ImageSkin_Click(Index As Integer)
```

```
’ 弹出换肤窗体
```

```
FormSkin.Show vbModal
```

```
End Sub
```

#### ·显示和记录上次扫描结果

```
Private Function ShowAndRecordScanResult()
```

```
Dim sSettingsFile As String
```

```
’ 软件设置记录文件
```

```
If Right(App.Path, 1) = "\" Then
```

```
sSettingsFile = App.Path & "Settings.ini"
```

```
Else
```

```
sSettingsFile = App.Path & "\\Settings.ini"
```

```

End If

' 扫描用时
Call WriteIni(sSettingsFile, "History", "LastScanUsedTime", LabelTimeUsed.Caption)

' 扫描文件数
Call WriteIni(sSettingsFile, "History", "LastScanFiles", LabelScannedFileCount.Caption)

' 病毒数
Call WriteIni(sSettingsFile, "History", "LastDetectedSpywares", LabelTrojanFileCount.Caption)
DoEvents
Dim sLascScanUsedTime As String

' 上次扫描用时
sLascScanUsedTime = ReadIni(sSettingsFile, "History", "LastScanUsedTime")

' 将各种信息显示在界面中

' 扫描用时
LabelLastScanUsedTime.Caption = sLascScanUsedTime

' 扫描文件数
LabelLastScanFiles.Caption = ReadIni(sSettingsFile, "History", "LastScanFiles") & "个"

' 检测到的病毒数量
LabelLastDetectedSpywares.Caption = ReadIni(sSettingsFile, "History", "LastDetectedSpywares") & "个"

' 设置各个控件中位置
LabelLastScanUsedTimeTitle.Left = 120
LabelLastScanUsedTime.Left = LabelLastScanUsedTimeTitle.Left + LabelLastScanUsedTimeTitle.Width + 20
LabelLastScanFilesTitle.Left = LabelLastScanUsedTime.Left + LabelLastScanUsedTime.Width + 200
LabelLastScanFiles.Left = LabelLastScanFilesTitle.Left + LabelLastScanFilesTitle.Width + 20
LabelLastDetectedSpywaresTitle.Left = LabelLastScanFiles.Left + LabelLastScanFiles.Width + 200
LabelLastDetectedSpywares.Left=LabelLastDetectedSpywaresTitle.Left+LabelLastDetectedSpywaresTitle.Width + 20
End Function

```

#### 鼠标在换肤按钮上移动

```

Private Sub ImageSkin_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

' 控制换肤图片按钮状态，使之看起来像按钮的三态效果
ImageSkin(0).Visible = False
ImageSkin(1).Visible = True
ImageSkin(2).Visible = False
End Sub

```

#### 鼠标在换肤按钮上按下

```
Private Sub ImageSkin_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
’ 控制换肤按钮状态
```

```
ImageSkin(0).Visible = False
```

```
ImageSkin(1).Visible = False
```

```
ImageSkin(2).Visible = True
```

```
End Sub
```

#### 鼠标在换肤按钮上抬起

```
Private Sub ImageSkin_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
’ 控制换肤按钮状态
```

```
If ImageSkin(2).Visible = True Then
```

```
FormSkin.Show vbModal
```

```
End If
```

```
End Sub
```

#### 点击升级按钮

```
Private Sub ImageUpdate_Click()
```

```
FormUpdate.Show vbModal
```

```
End Sub
```

#### \*鼠标在扫描结果中点击，弹出右键菜单

```
Private Sub ListViewScanResult_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
’ 判断是否有扫描结果内容
```

```
If ListViewScanResult.ListItems.Count > 0 Then
```

```
’ 判断是否是右键
```

```
If Button = 2 Then
```

```
’ 弹出菜单
```

```
FormMenu.PopupMenu FormMenu.m_ScanResultMenu
```

```
End If
```

```
End If
```

```
End Sub
```

#### \*鼠标在防护记录中点击，弹出右键菜单

```
Private Sub ListViewShieldLog_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
’ 判断是否有内容
```

```
If ListViewShieldLog.ListItems.Count > 0 Then
```

```
’ 判断是否是右键
```

```
If Button = 2 Then
```

```
’ 弹出菜单
```

```
FormMenu.PopupMenu FormMenu.m_ShieldLog
End If
End If
End Sub
```

#### ‘右键扫描定时器控件’

```
Private Sub TimerRightClickScan_Timer()
Dim sRightClickScanFile As String

' 右键扫描记录文件
If Right(App.Path, 1) = "\" Then
sRightClickScanFile = App.Path & "RCS.ini"
Else
sRightClickScanFile = App.Path & "\RCS.ini"
End If
WaitForWrite:

' 同时读写的标识
Dim lWriteReadFlag As Long
lWriteReadFlag = ReadIni(sRightClickScanFile, "Normal", "WRFlag")
If lWriteReadFlag = 0 Then

' 防止同时读写的标识
Call WriteIni(sRightClickScanFile, "Normal", "WRFlag", 1)

' 记录数
Dim lPreCount As Long
lPreCount = ReadIni(sRightClickScanFile, "Normal", "RC")
Dim i As Long
Dim sRightScanFile As String
For i = 1 To lPreCount

' 读取要扫描的文件
sRightScanFile = ReadIni(sRightClickScanFile, "Normal", i)
If GetAttr(sRightScanFile) And vbDirectory Then

' 扫描目录
RightClickScanCall sRightScanFile
Else
DoEvents

' 扫描文件
RightClickScanFile sRightScanFile
End If
Next i

' 重置记录数
Call WriteIni(sRightClickScanFile, "Normal", "RC", 0)
```

```
' 防止同时读写的标识
Call WriteIni(sRightClickScanFile, "Normal", "WRFlag", 0)
Else
DoEvents
GoTo WaitForWrite
End If
End Sub
```

‘定时器控件，定时更新版本信息，主要用于下载完成后同步界面中的版本号

```
Private Sub TimerUpdateRefresh_Timer()
On Error GoTo ErrorExit

' 读取软件版本
Dim sVersionFile As String
If Right(App.Path, 1) = "\" Then
sVersionFile = App.Path & "Version.ini"
Else
sVersionFile = App.Path & "\Version.ini"
End If
```

```
' Version.ini 文件内容如下：
[Version]
WholeVersion=000300
SoftwareVersion=1.0.6
SignatureVersion=20130313001
```

各字段的含意如下：

名称	含意
WholeVersion	整体版本号，这个版本号主要用来判断是否有升级的内容
SoftwareVersion	软件版本号，这个版本号是用于在软件界面中显示版本
SignatureVersion	病毒库版本号，这个版本号是用于在软件界面中显示病毒库版本

```
Dim sSoftwareVersion As String
sSoftwareVersion = ReadIni(sVersionFile, "Version", "SoftwareVersion")
```

```
' 读取特征码版本
Dim lSignatureVersion
lSignatureVersion = ReadIni(sVersionFile, "Version", "SignatureVersion")
FormScan.LabelSoftwareVersion.Caption = sSoftwareVersion
FormScan.LabelSignatureVersion.Caption = lSignatureVersion
TimerUpdateRefresh.Enabled = False
ErrorExit:
End Sub
```

扫描拖放文件

```
Private Function OleScanFile(ByVal sFile As String) As String

' 软件设置记录文件
```

```

Dim sSettingsFile As String
If Right(App.Path, 1) = "\" Then
sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\Settings.ini"
End If

' 发现病毒时自动清除
Dim bClearVirus As Boolean
bClearVirus = ReadIni(sSettingsFile, "Scan", "ClearVirus")

' 发现病毒时提示
Dim bAlertVirus As Boolean
bAlertVirus = ReadIni(sSettingsFile, "Scan", "AlertVirus")

' 发现病毒时忽略
Dim bIgnoreVirus As Boolean
bIgnoreVirus = ReadIni(sSettingsFile, "Scan", "IgnoreVirus ")

' 发现病毒时隔离
Dim bQuarantineVirus As Boolean
bQuarantineVirus = ReadIni(sSettingsFile, "Scan", "QuarantineVirus")
Dim fAlertForm As New FormAlertVirus

' 扫描文件数
lScannedFiles = lScannedFiles + 1
LabelScannedFileCount.Caption = lScannedFiles

' 扫描耗时
LabelTimeUsed.Caption = CDate(Time - dScanStartTime)

' 正在扫描的文件
If Len(sFile) > 60 Then
LabelNowScaning.Caption = Left(sFile, 47) & "... " & Right(sFile, 10)
Else
LabelNowScaning.Caption = sFile
End If

' 扫描结果
Dim sScanResult As String

' 扫描
sScanResult = ScanFile(sFile)

' 清除病毒返回值
Dim lDeleteFileReturn As Long

' 当前路径
Dim sTempAppPath As String

' 复制文件

```

```

Dim sTargetFileName As String

' 复制文件返回
Dim lCopyFileReturn As Long

' 检查并显示扫描结果
If sScanResult <> "" Then

' 发现病毒数
lTrojanFiles = lTrojanFiles + 1
LabelTrojanFileCount.Caption = lTrojanFiles

' 忽略
If bIgnoreVirus = True Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "未清除"
End With
End If

' 自动清除
If bClearVirus = True Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已清除"
End With
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "清除失败"
End With
End If
End If

' 隔离
If bQuarantineVirus = True Then

' 确保当前路径包含"\
If Right(App.Path, 1) = "\" Then
sTempAppPath = App.Path
Else

```

```

sTempAppPath = App.Path & "\
End If

' 目标复制文件名
sTargetFileName = sTempAppPath & "quarantine\" & RndChr(8) & ".tmp"

' 复制文件
lCopyFileReturn = CopyFile(sFile, sTargetFileName, 0)

' 复制成功
If lCopyFileReturn <> 0 Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已隔离"
End With

' 隔离记录文件
Call WriteQuarantineFile(sFile, sTargetFileName)
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If

' 复制失败, 隔离失败
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If
End If

' 提示
If bAlertVirus = True Then
SetForegroundWindow FormScan.hWnd
With fAlertForm
.LabelFile = sFile
.LabelVirusName = sScanResult
.Show vbModal

```



```
End With
End If
End If
```

```
' 返回扫描结果
OleScanFile = sScanResult
End Function
```

#### 右键扫描文件

```
Public Function RightClickScanFile(ByVal sFile As String)
```

```
' 使退出菜单不可用
FormMenu.m_Exit.Enabled = False
```

```
' 病毒列表显示
ImageScanResult_Click (0)
```

```
' 扫描按钮与状态可见状态
PictureScanBottoms.Visible = False
PictureScanState.Visible = True
Dim sSettingsFile As String
```

```
' 软件设置记录文件
If Right(App.Path, 1) = "\" Then
sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\Settings.ini"
End If
```

```
' 读取扫描时减少 CPU 占用设置
Dim lLowCPU As Long
lLowCPU = ReadIni(sSettingsFile, "Scan", "CheckLowCPU")
If lLowCPU = 1 Then
```

```
' 设置当前进程优先级为低于标准
Call SetProcessPriority(16384)
End If
```

```
' 清除前次扫描结果
ListViewScanResult.ListItems.Clear
```

```
' 停止扫描标志
bStopScanFlag = False
```

```
' 扫描文件数
lScannedFiles = 0
```

```
' 病毒数
lTrojanFiles = 0
```

```

LabelScanedFileCount.Caption = 0
LabelTrojanFileCount.Caption = 0

' 扫描开始时间
dScanStartTime = Time

' 右键扫描结果
With FormRightClickScanResult
.Top = (Screen.Height - .Height) / 2
.Left = (Screen.Width - .Width) / 2
End With

' 扫描文件
Dim sScanResult As String
sScanResult = OleScanFile(sFile)
With FormRightClickScanResult

' id 号
.ListViewScanResult.ListItems.Add , , .ListViewScanResult.ListItems.Count + 1

' 文件
.ListViewScanResult.ListItems(.ListViewScanResult.ListItems.Count).SubItems(1) = sFile

' 扫描结果
If Len(sScanResult) < 2 Then
.ListViewScanResult.ListItems(.ListViewScanResult.ListItems.Count).SubItems(2) = "安全"
Else
.ListViewScanResult.ListItems(.ListViewScanResult.ListItems.Count).SubItems(2) = sScanResult
End If
End With
DoEvents

' 扫描结束
Call ShowAndRecordScanResult

' 扫描按钮与状态可见状态
PictureScanBottoms.Visible = True
PictureScanState.Visible = False

' 设置当前进程优先级为标准
Call SetProcessPriority(32)
LabelNowScaning.Caption = "扫描结束。"

' 选中病毒
If ListViewScanResult.ListItems.Count > 0 Then
Dim k As Long
For k = 1 To ListViewScanResult.ListItems.Count
ListViewScanResult.ListItems(k).Checked = True
Next k
End If

```

```
' 退出菜单可用
FormMenu.m_Exit.Enabled = True
```

```
' 使软件使用最少内存
MiniUseMemory
```

```
' 显示右键扫描结果
With FormRightClickScanResult
.Show vbModal
End With
End Function
```

## 右键扫描目录

```
Public Sub RightClickScanFolder(sFolder As String)

' 搜索用自定义结构
Dim uWFD As WIN32_FIND_DATA

' 文件
Dim sFile As String

' 目录
Dim sDir As String

' 搜索句柄
Dim lSerachHwnd As Long
Dim lSerachNextHwnd As Long

' 特征码扫描结果
Dim sScanResult As String
lSerachHwnd = FindFirstFile(sFolder & ".*", uWFD)

' 遍历文件及目录
Do
DoEvents

' 停止扫描
If bStopScanFlag = True Then
Exit Do
End If

' 如果是目录
If uWFD.dwFileAttributes And FILE_ATTRIBUTE_DIRECTORY Then

' 判断是否是根目录和上级目录名
If Left(uWFD.cFileName, 1) <> "." And Left(uWFD.cFileName, 2) <> ".." Then

' 取得路径
sDir = sFolder & TrimNull(uWFD.cFileName)
```

```

' 添加到待扫描目录
If Right(sDir, 1) = "\" Then
ListFolders.AddItem sDir
Else
ListFolders.AddItem sDir & "\"
End If
End If

' 如果是文件
Else

' 获取文件全路径
sFile = sFolder & TrimNull(uWFD.cFileName)
sFile = Replace(sFile, "\\ ", "\ ")

' 软件设置记录文件
Dim sSettingsFile As String
If Right(App.Path, 1) = "\" Then
sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\Settings.ini"
End If

' 扫描全部文件或 PE 文件
Dim bScanAllFile As Boolean
bScanAllFile = ReadIni(sSettingsFile, "Scan", "ScanAllFiles")

' 发现病毒时自动清除
Dim bClearVirus As Boolean
bClearVirus = ReadIni(sSettingsFile, "Scan", "ClearVirus")

' 发现病毒时提示
Dim bAlertVirus As Boolean
bAlertVirus = ReadIni(sSettingsFile, "Scan", "AlertVirus")

' 发现病毒时忽略
Dim bIgnoreVirus As Boolean
bIgnoreVirus = ReadIni(sSettingsFile, "Scan", "IgnoreVirus ")

' 发现病毒时隔离
Dim bQuarantineVirus As Boolean
bQuarantineVirus = ReadIni(sSettingsFile, "Scan", "QuarantineVirus")
Dim fAlertForm As New FormAlertVirus
If bScanAllFile = True Then

' 扫描文件数
lScannedFiles = lScannedFiles + 1
LabelScannedFileCount.Caption = lScannedFiles

' 扫描耗时

```

```

LabelTimeUsed.Caption = CDate(Time - dScanStartTime)

' 正在扫描的文件
If Len(sFile) > 60 Then
LabelNowScanning.Caption = Left(sFile, 47) & "... " & Right(sFile, 10)
Else
LabelNowScanning.Caption = sFile
End If

' 扫描
sScanResult = ScanFile(sFile)

' 清除病毒返回值
Dim lDeleteFileReturn As Long

' 当前路径
Dim sTempAppPath As String

' 复制文件
Dim sTargetFileName As String

' 复制文件返回
Dim lCopyFileReturn As Long
With FormRightClickScanResult

' id 号
.ListViewScanResult.ListItems.Add , , .ListViewScanResult.ListItems.Count + 1

' 文件
.ListViewScanResult.ListItems(.ListViewScanResult.ListItems.Count).SubItems(1) = sFile

' 扫描结果
If Len(sScanResult) < 2 Then
.ListViewScanResult.ListItems(.ListViewScanResult.ListItems.Count).SubItems(2) = "安全"
Else
.ListViewScanResult.ListItems(.ListViewScanResult.ListItems.Count).SubItems(2) = sScanResult
End If
End With
DoEvents

' 检查并显示扫描结果
If sScanResult <> "" Then

' 发现病毒数
lTrojanFiles = lTrojanFiles + 1
LabelTrojanFileCount.Caption = lTrojanFiles

' 忽略
If bIgnoreVirus = True Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1

```

```

.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "未清除"
End With
End If

' 自动清除
If bClearVirus = True Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已清除"
End With
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "清除失败"
End With
End If
End If

' 隔离
If bQuarantineVirus = True Then

' 确保当前路径包含"\
If Right(App.Path, 1) = "\" Then
sTempAppPath = App.Path
Else
sTempAppPath = App.Path & "\"
End If

' 目标复制文件名
sTargetFileName = sTempAppPath & "quarantine\" & RndChr(8) & ".tmp"

' 复制文件
lCopyFileReturn = CopyFile(sFile, sTargetFileName, 0)

' 复制成功
If lCopyFileReturn <> 0 Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then
With ListViewScanResult

```

```

.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已隔离"
End With

```

' 隔离记录文件

```
Call WriteQuarantineFile(sFile, sTargetFileName)
```

```
Else
```

```
With ListViewScanResult
```

```
.ListItems.Add , , .ListItems.Count + 1
```

```
.ListItems(.ListItems.Count).SubItems(1) = sFile
```

```
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
```

```
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
```

```
End With
```

```
End If
```

' 复制失败, 隔离失败

```
Else
```

```
With ListViewScanResult
```

```
.ListItems.Add , , .ListItems.Count + 1
```

```
.ListItems(.ListItems.Count).SubItems(1) = sFile
```

```
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
```

```
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
```

```
End With
```

```
End If
```

```
End If
```

' 提示

```
If bAlertVirus = True Then
```

```
SetForegroundWindow FormScan.hWnd
```

```
With fAlertForm
```

```
.LabelFile = sFile
```

```
.LabelVirusName = sScanResult
```

```
.Show vbModal
```

```
End With
```

```
End If
```

```
End If
```

```
Else
```

' 判断文件类型

```
If UCase(Right(sFile, 4)) = UCase(".dll") _
```

```
Or UCase(Right(sFile, 4)) = UCase(".exe") _
```

```
Or UCase(Right(sFile, 4)) = UCase(".com") _
```

```
Or UCase(Right(sFile, 4)) = UCase(".ocx") _
```

```
Or UCase(Right(sFile, 4)) = UCase(".scr") _
```

```
Then
```

' 扫描文件数

```
lScannedFiles = lScannedFiles + 1
```

```
LabelScannedFileCount.Caption = lScannedFiles
```

```

' 扫描耗时
LabelTimeUsed.Caption = CDate(Time - dScanStartTime)

' 正在扫描的文件
If Len(sFile) > 60 Then
LabelNowScanning.Caption = Left(sFile, 47) & "... " & Right(sFile, 10)
Else
LabelNowScanning.Caption = sFile
End If

' 扫描
sScanResult = ScanFile(sFile)
With FormRightClickScanResult

' id 号
.ListViewScanResult.ListItems.Add , , .ListViewScanResult.ListItems.Count + 1

' 文件
.ListViewScanResult.ListItems(.ListViewScanResult.ListItems.Count).SubItems(1) = sFile

' 扫描结果
If Len(sScanResult) < 2 Then
.ListViewScanResult.ListItems(.ListViewScanResult.ListItems.Count).SubItems(2) = "安全"
Else
.ListViewScanResult.ListItems(.ListViewScanResult.ListItems.Count).SubItems(2) = sScanResult
End If
End With
DoEvents

' 检查并显示扫描结果
If sScanResult <> "" Then

' 发现病毒数
lTrojanFiles = lTrojanFiles + 1
LabelTrojanFileCount.Caption = lTrojanFiles

' 忽略
If bIgnoreVirus = True Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "未清除"
End With
End If

' 自动清除
If bClearVirus = True Then

' 清除病毒

```



```

lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then

' 向扫描结果控件中添加记录
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已清除"
End With
Else

' 向扫描结果控件中添加记录
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "清除失败"
End With
End If
End If

' 隔离
If bQuarantineVirus = True Then

' 确保当前路径包含"\
If Right(App.Path, 1) = "\" Then
sTempAppPath = App.Path
Else
sTempAppPath = App.Path & "\"
End If

' 目标复制文件名
sTargetFileName = sTempAppPath & "quarantine\" & RndChr(8) & ".tmp"

' 复制文件
lCopyFileReturn = CopyFile(sFile, sTargetFileName, 0)

' 复制成功
If lCopyFileReturn <> 0 Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then

' 向扫描结果控件中添加记录
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已隔离"

```

```

End With

' 隔离记录文件
Call WriteQuarantineFile(sFile, sTargetFileName)
Else

' 向扫描结果控件中添加记录
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If

' 复制失败, 隔离失败
Else

' 向扫描结果控件中添加记录
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If
End If

' 判断是否发出提示的标志
If bAlertVirus = True Then

SetForegroundWindow FormScan.hWnd

' 在事下角弹出窗体, 提示用户
With fAlertForm

' 文件名
.LabelFile = sFile

' 病毒名称
.LabelVirusName = sScanResult
.Show vbModal
End With
End If
End If
End If
End If
End If

' 继续遍历文件
lSerachNextHwnd = FindNextFile(lSerachHwnd, uWFD)

```

```
Loop While lSerachNextHwnd <> 0
```

```
' 关闭搜索句柄
```

```
FindClose lSerachHwnd
```

```
End Sub
```

### 右键扫描全部类型的文件

```
Private Function RightClickScanCall(ByVal sFolder As String)
```

```
' 退出菜单不可用
```

```
FormMenu.m_Exit.Enabled = False
```

```
' 病毒列表显示
```

```
ImageScanResult_Click (0)
```

```
' 扫描按钮与状态可见状态
```

```
PictureScanBottoms.Visible = False
```

```
PictureScanState.Visible = True
```

```
Dim sSettingsFile As String
```

```
' 软件设置记录文件
```

```
If Right(App.Path, 1) = "\" Then
```

```
sSettingsFile = App.Path & "Settings.ini"
```

```
Else
```

```
sSettingsFile = App.Path & "\Settings.ini"
```

```
End If
```

```
' 读取扫描时减少 CPU 占用设置
```

```
Dim lLowCPU As Long
```

```
lLowCPU = ReadIni(sSettingsFile, "Scan", "CheckLowCPU")
```

```
If lLowCPU = 1 Then
```

```
' 设置当前进程优先级为低于标准
```

```
Call SetProcessPriority(16384)
```

```
End If
```

```
' 清除前次扫描结果
```

```
ListViewScanResult.ListItems.Clear
```

```
' 停止扫描标志
```

```
bStopScanFlag = False
```

```
' 扫描文件数
```

```
lScanedFiles = 0
```

```
' 病毒数
```

```
lTrojanFiles = 0
```

```
LabelScanedFileCount.Caption = 0
```

```
LabelTrojanFileCount.Caption = 0
```

```

' 扫描开始时间
dScanStartTime = Time

' 右键扫描结果
With FormRightClickScanResult
.Top = (Screen.Height - .Height) / 2
.Left = (Screen.Width - .Width) / 2
End With
If Right(sFolder, 1) = "\" Then

' 右键扫描目录
ListFolders.AddItem sFolder
Else

' 右键扫描目录
ListFolders.AddItem sFolder & "\"
End If
Do While ListFolders.ListCount <> 0
DoEvents

' 开始扫描
RightClickScanFolder ListFolders.List(0)

' 第一行搜索完成后删除第一行
ListFolders.RemoveItem 0

' 用户停止扫描
If bStopScanFlag = True Then
Exit Do
End If
Loop

' 扫描结束
Call ShowAndRecordScanResult
' If bStopScanFlag = True Then
' With FormScanFinish
'.LabelTitle.Caption = "扫描中止"
'.LabelMessage.Caption = "已扫描 " & lScannedFiles & " 个文件，发现 " & lTrojanFiles & " 个病毒。"
'.Show vbModal
' End With
' Else
' With FormScanFinish
'.LabelTitle.Caption = "扫描完成"
'.LabelMessage.Caption = "已扫描 " & lScannedFiles & " 个文件，发现 " & lTrojanFiles & " 个病毒。"
'.Show vbModal
' End With
' End If

' 设置当前进程优先级为标准
Call SetProcessPriority(32)

```

```

' 扫描按钮与状态可见状态
PictureScanBottoms.Visible = True
PictureScanState.Visible = False
LabelNowScaning.Caption = "扫描结束。"

' 选中病毒
If ListViewScanResult.ListItems.Count > 0 Then
Dim k As Long
For k = 1 To ListViewScanResult.ListItems.Count
ListViewScanResult.ListItems(k).Checked = True
Next k
End If

' 退出菜单可用
FormMenu.m_Exit.Enabled = True

' 使软件使用最少内存
MiniUseMemory
With FormRightClickScanResult
.Show vbModal
End With
End Function

```

使软件支持文件拖放扫描，将文件拖动到界面中时，扫描被拖动的文件或目录

```

Public Sub OleScanFolder(sFolder As String)

' 搜索用自定义结构
Dim uWFD As WIN32_FIND_DATA

' 文件
Dim sFile As String

' 目录
Dim sDir As String

' 搜索句柄
Dim lSerachHwnd As Long
Dim lSerachNextHwnd As Long

' 特征码扫描结果
Dim sScanResult As String
lSerachHwnd = FindFirstFile(sFolder & ".*", uWFD)

' 遍历文件及目录
Do
DoEvents

' 停止扫描

```

```

If bStopScanFlag = True Then
Exit Do
End If

' 如果是目录
If uWFD.dwFileAttributes And FILE_ATTRIBUTE_DIRECTORY Then

' 判断是否是根目录和上级目录名
If Left(uWFD.cFileName, 1) <> "." And Left(uWFD.cFileName, 2) <> ".." Then

' 取得路径
sDir = sFolder & TrimNull(uWFD.cFileName)

' 添加到待扫描目录
If Right(sDir, 1) = "\" Then
ListFolders.AddItem sDir
Else
ListFolders.AddItem sDir & "\"
End If
End If

' 如果是文件
Else

' 获取文件全路径
sFile = sFolder & TrimNull(uWFD.cFileName)
sFile = Replace(sFile, "\\ ", "\ ")

' 软件设置记录文件
Dim sSettingsFile As String
If Right(App.Path, 1) = "\" Then
sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\Settings.ini"
End If

' 扫描全部文件或 PE 文件
Dim bScanAllFile As Boolean
bScanAllFile = ReadIni(sSettingsFile, "Scan", "ScanAllFiles")

' 发现病毒时自动清除
Dim bClearVirus As Boolean
bClearVirus = ReadIni(sSettingsFile, "Scan", "ClearVirus")

' 发现病毒时提示
Dim bAlertVirus As Boolean
bAlertVirus = ReadIni(sSettingsFile, "Scan", "AlertVirus")

' 发现病毒时忽略
Dim bIgnoreVirus As Boolean
bIgnoreVirus = ReadIni(sSettingsFile, "Scan", "IgnoreVirus ")

```

```

'发现病毒时隔离
Dim bQuarantineVirus As Boolean
bQuarantineVirus = ReadIni(sSettingsFile, "Scan", "QuarantineVirus")
Dim fAlertForm As New FormAlertVirus
If bScanAllFile = True Then

'扫描文件数
lScannedFiles = lScannedFiles + 1
LabelScanedFileCount.Caption = lScannedFiles

'扫描耗时
LabelTimeUsed.Caption = CDate(Time - dScanStartTime)

'正在扫描的文件
If Len(sFile) > 60 Then

'如果文件路径加文件名长度超过 60, 则只显示前后部分, 中间用 "..." 代替, 这样显示更加美观
LabelNowScaning.Caption = Left(sFile, 47) & "..." & Right(sFile, 10)
Else

'如果文件路径加文件名长度小于 60, 则显示全路径
LabelNowScaning.Caption = sFile
End If

'扫描文件
sScanResult = ScanFile(sFile)

'清除病毒返回值
Dim lDeleteFileReturn As Long

'当前路径
Dim sTempAppPath As String

'复制文件
Dim sTargetFileName As String

'复制文件返回
Dim lCopyFileReturn As Long

'检查并显示扫描结果
If sScanResult <> "" Then

'发现病毒数
lTrojanFiles = lTrojanFiles + 1

'更新界面中显示的发现病毒数量
LabelTrojanFileCount.Caption = lTrojanFiles

'忽略
If bIgnoreVirus = True Then

```

```

' 向扫描结果中添加记录
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1、

' 文件名
.ListItems(.ListItems.Count).SubItems(1) = sFile

' 病毒名
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "未清除"
End With
End If

' 自动清除
If bClearVirus = True Then

' 删除病毒文件，清除病毒
lDeleteFileReturn = DeleteFile(sFile)

' 判断删除病毒文件是否成功
If lDeleteFileReturn <> 0 Then

' 向扫描结果控件中添加记录
With ListViewScanResult

' 序号
.ListItems.Add , , .ListItems.Count + 1

' 文件名
.ListItems(.ListItems.Count).SubItems(1) = sFile

' 病毒名
.ListItems(.ListItems.Count).SubItems(2) = sScanResult

' 处理结果
.ListItems(.ListItems.Count).SubItems(3) = "已清除"
End With
Else

' 向扫描结果控件中添加记录
With ListViewScanResult

' 序号
.ListItems.Add , , .ListItems.Count + 1

' 文件名
.ListItems(.ListItems.Count).SubItems(1) = sFile

' 病毒名
.ListItems(.ListItems.Count).SubItems(2) = sScanResult

```



```

' 处理结果
.ListItems(.ListItems.Count).SubItems(3) = "清除失败"
End With
End If
End If

' 隔离
If bQuarantineVirus = True Then

' 确保当前路径包含"\
If Right(App.Path, 1) = "\" Then
sTempAppPath = App.Path
Else
sTempAppPath = App.Path & "\"
End If

' 目标复制文件名
sTargetFileName = sTempAppPath & "quarantine\" & RndChr(8) & ".tmp"

' 复制文件
lCopyFileReturn = CopyFile(sFile, sTargetFileName, 0)

' 复制成功
If lCopyFileReturn <> 0 Then

' 复制之后再删除原文件，这样就完成了隔离
lDeleteFileReturn = DeleteFile(sFile)

' 判断删除文件是否成功
If lDeleteFileReturn <> 0 Then

' 向扫描结果控件中添加记录
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已隔离"
End With

' 隔离记录文件
Call WriteQuarantineFile(sFile, sTargetFileName)
Else

' 向扫描结果控件中添加记录
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With

```

```

End If

' 复制失败，隔离失败
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If
End If

' 弹出对话框，发用户发出提示
If bAlertVirus = True Then
SetForegroundWindow FormScan.hWnd
With fAlertForm
.LabelFile = sFile
.LabelVirusName = sScanResult
.Show vbModal
End With
End If
End If
Else

' 判断文件类型
If UCase(Right(sFile, 4)) = UCase(".dll") _
Or UCase(Right(sFile, 4)) = UCase(".exe") _
Or UCase(Right(sFile, 4)) = UCase(".com") _
Or UCase(Right(sFile, 4)) = UCase(".ocx") _
Or UCase(Right(sFile, 4)) = UCase(".scr") _
Then

' 扫描文件数
lScannedFiles = lScannedFiles + 1
LabelScannedFileCount.Caption = lScannedFiles

' 扫描耗时
LabelTimeUsed.Caption = CDate(Time - dScanStartTime)

' 正在扫描的文件
If Len(sFile) > 60 Then
LabelNowScanning.Caption = Left(sFile, 47) & "... " & Right(sFile, 10)
Else
LabelNowScanning.Caption = sFile
End If

' 扫描
sScanResult = ScanFile(sFile)

' 检查并显示扫描结果

```

```

If sScanResult <> "" Then

' 发现病毒数
lTrojanFiles = lTrojanFiles + 1
LabelTrojanFileCount.Caption = lTrojanFiles

' 忽略
If bIgnoreVirus = True Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "未清除"
End With
End If

' 自动清除
If bClearVirus = True Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已清除"
End With
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "清除失败"
End With
End If
End If

' 隔离
If bQuarantineVirus = True Then

' 确保当前路径包含"\
If Right(App.Path, 1) = "\" Then
sTempAppPath = App.Path
Else
sTempAppPath = App.Path & "\"
End If

' 目标复制文件名
sTargetFileName = sTempAppPath & "quarantine\" & RndChr(8) & ".tmp"

```

```

' 复制文件
lCopyFileReturn = CopyFile(sFile, sTargetFileName, 0)

' 复制成功
If lCopyFileReturn <> 0 Then

' 清除病毒
lDeleteFileReturn = DeleteFile(sFile)
If lDeleteFileReturn <> 0 Then
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "已隔离"
End With

' 隔离记录文件
Call WriteQuarantineFile(sFile, sTargetFileName)
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If

' 复制失败, 隔离失败
Else
With ListViewScanResult
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = sFile
.ListItems(.ListItems.Count).SubItems(2) = sScanResult
.ListItems(.ListItems.Count).SubItems(3) = "隔离失败"
End With
End If
End If

' 提示
If bAlertVirus = True Then
SetForegroundWindow FormScan.hWnd
With fAlertForm
.LabelFile = sFile
.LabelVirusName = sScanResult
.Show vbModal
End With
End If
End If
End If
End If
End If

```

```

' 继续遍历文件
lSerachNextHwnd = FindNextFile(lSerachHwnd, uWFD)
Loop While lSerachNextHwnd <> 0

' 关闭搜索句柄
FindClose lSerachHwnd
End Sub

```

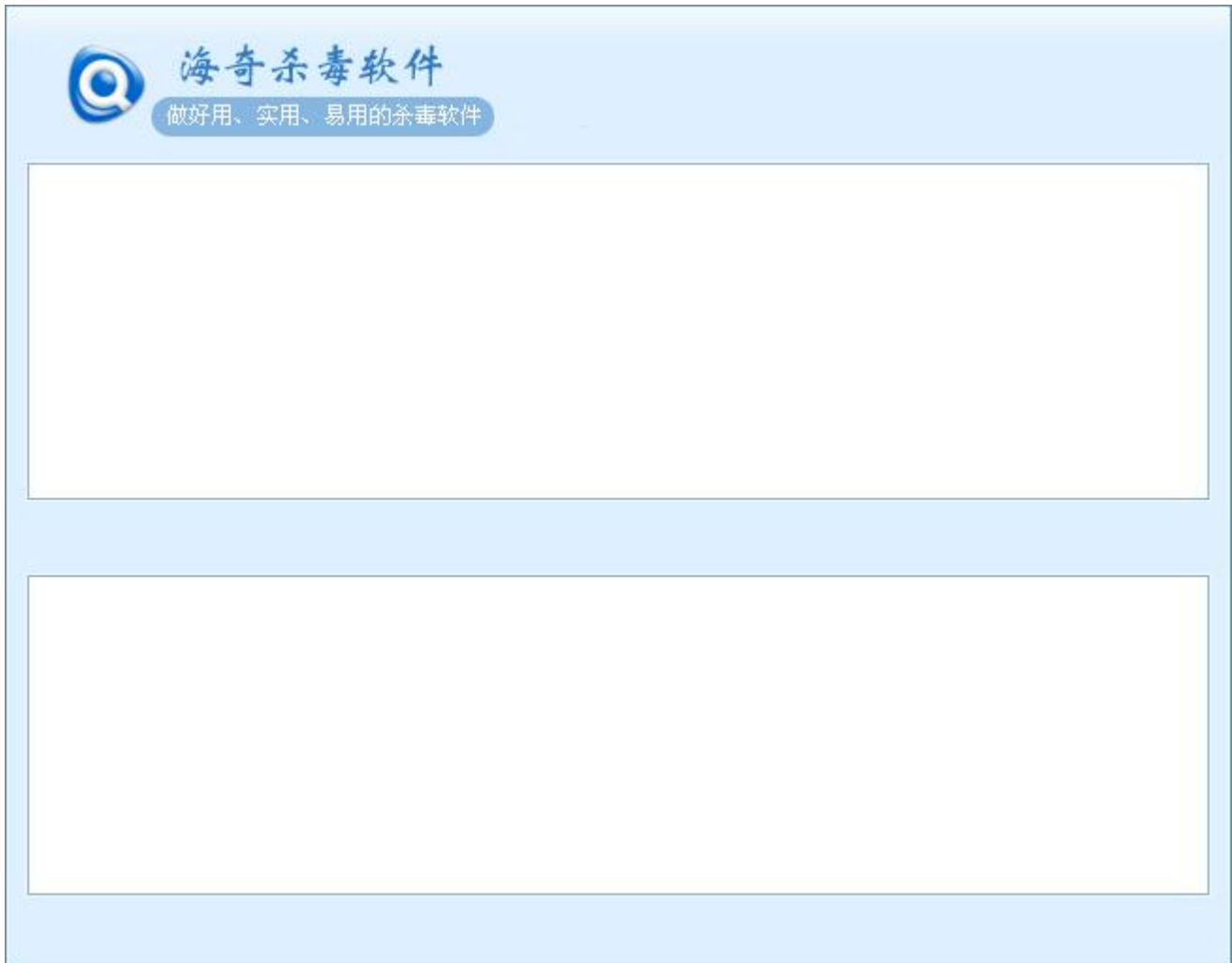
本窗体的换肤函数，根据每个窗体包含的内容的不同，每个窗体的换肤函数都有差异

```

Public Function ReSkinMe()
With Me

```

' 加载窗体背景图片，图片是：



```

.Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Scan.bmp")

```

' 加载图片模拟的三态度退出按键图片

```

.ImageExit(0).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit0.bmp")
.ImageExit(1).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit1.bmp")

```


```
.ImageExit(2).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit2.bmp")
```


```
.ImageMin(0).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Min0.bmp")
```


```
.ImageMin(1).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Min1.bmp")
```

```
.ImageMin(2).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Min2.bmp")
```

’ 以上加载的是界面右上角用图片模拟的最小化按钮，之所以使用三张图片，是因为要实现按钮的三态效果，即：普通状态下、鼠标移动时、鼠标按下时三种不同状态时的效果。这里使用的三张图片分别是：

’ 正常状态下：

’ 鼠标移动时：

’ 鼠标按下时：

```
.ImageSkin(0).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Skin0.bmp")
```

```
.ImageSkin(1).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Skin1.bmp")
```

```
.ImageSkin(2).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Skin2.bmp")
```

```
.ImageMemoryScan(0) = LoadPicture(App.Path & "\Skin\" & sSkin & "\QuickScan0.bmp")
```

```
.ImageMemoryScan(1) = LoadPicture(App.Path & "\Skin\" & sSkin & "\QuickScan1.bmp")
```

’ 上面是加载全盘扫描按钮的背景图片，这个按钮是用两张图片模拟的两态按钮，一用图片是正常状态下的图片，一张是鼠标在此按钮上移动的图片，加载到程序中时，两个图片是重叠在一起的，所以看起来会是个大按钮的样子。这两张图分别是：



```
.ImageFullDiskScan(0) = LoadPicture(App.Path & "\Skin\" & sSkin & "\FullScan0.bmp")
```

```
.ImageFullDiskScan(1) = LoadPicture(App.Path & "\Skin\" & sSkin & "\FullScan1.bmp")
```

’ 上面是加载全盘扫描按钮的背景图片，这个按钮是用两张图片模拟的两态按钮，一用图片是正常状态下的图片，一张是鼠标在此按钮上移动的图片，加载到程序中时，两个图片是重叠在一起的，所以看起来会是个大按钮的样子。这两张图分别是：



```
.ImageCustomerScan(0)=LoadPicture(App.Path & "\Skin\" & sSkin & "\CustomerScan0.bmp")
```

```
.ImageCustomerScan(1)=LoadPicture(App.Path & "\Skin\" & sSkin & "\CustomerScan1.bmp")
```

’ 上面是加载自定义扫描按钮的背景图片，这个按钮是用两张图片模拟的两态按钮，一用图片是正常状态下的图片，一张是鼠标在此按钮上移动的图片，加载到程序中时，两个图片是重叠在一起的，所以看起来会是个大按钮的样子。这两张图分别是：



```
.ImageSecurityState(0)=LoadPicture(App.Path & "\Skin\" & sSkin & "\SecurityState0.bmp")
.ImageSecurityState(1)=LoadPicture(App.Path & "\Skin\" & sSkin & "\SecurityState1.bmp")
```

’ 以上是加载用图片模拟的两态选项卡控件中的安全状态图片，两张图片分别是：



```
.ImageScanResult(0) = LoadPicture(App.Path & "\Skin\" & sSkin & "\ScanResult0.bmp")
.ImageScanResult(1) = LoadPicture(App.Path & "\Skin\" & sSkin & "\ScanResult1.bmp")
```

’ 以上是加载用图片模拟的两态选项卡控件中的安全状态图片，两张图片分别是：



```
.ImageShield(0) = LoadPicture(App.Path & "\Skin\" & sSkin & "\Shield0.bmp")
.ImageShield(1) = LoadPicture(App.Path & "\Skin\" & sSkin & "\Shield1.bmp")
```

’ 以上是加载用图片模拟的两态选项卡控件中的安全状态图片，两张图片分别是：



```
.ImageHelp = LoadPicture(App.Path & "\Skin\" & sSkin & "\Help.bmp")
.ImageUpdate = LoadPicture(App.Path & "\Skin\" & sSkin & "\Update.bmp")
.ImageSetting = LoadPicture(App.Path & "\Skin\" & sSkin & "\Setting.bmp")
.ImageHomePage = LoadPicture(App.Path & "\Skin\" & sSkin & "\HomePage.bmp")
```

’ 以上是加载使用帮助、升级、设置、官网三个图片，图片如下：



```
End With
End Function
```

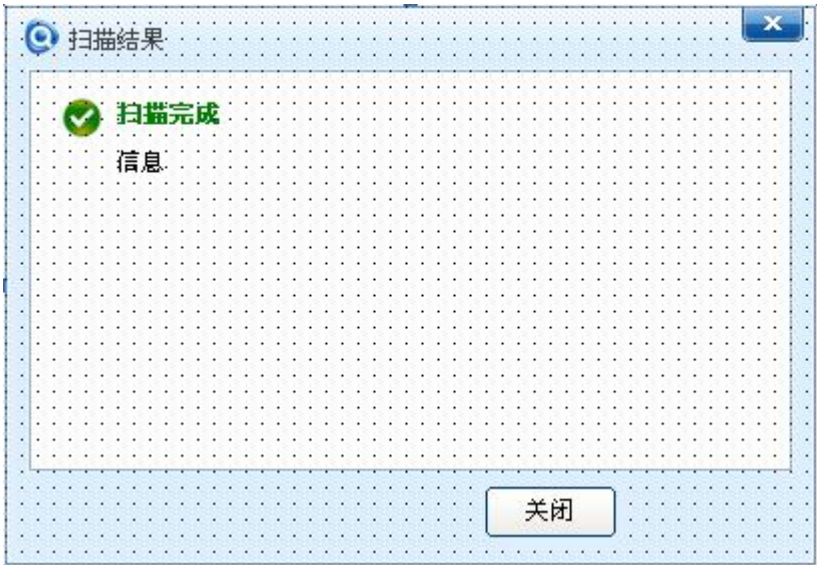
### 5.3.8. 病毒扫描完成窗体

窗体：

```
FormScanFinish
```

功能：  
扫描完成后提示用户的窗体。

界面设计：



窗体中包含的控件及功能说明：

控件名称	类别	功能
LabelMessage	文本框控件	用于显示扫描结果信息。
CommandClose	按钮控件	关闭按钮。
ImageExit	图片控件	用图片控件模拟的退出按钮。

代码：

```
Option Explicit
'api 声明
Private Declare Function ReleaseCapture Lib "user32" () As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long, ByVal lParam As Any) As Long
```

点击关闭按钮

```
Private Sub CommandClose_Click()
Unload Me
End Sub
```

鼠标在窗体中按下

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

'判断是否按下鼠标左键
If Button = vbLeftButton Then
```



```
' 为当前的应用程序释放鼠标捕获  
ReleaseCapture
```

```
' 移动窗体到鼠标位置  
SendMessage Me.hWnd, &HA1, 2, 0  
End If  
End Sub
```

#### 窗体启动函数

```
Private Sub Form_Load()
```

```
' 加载皮肤  
ReSkinMe  
Dim j As Long  
For j = 0 To 2
```

```
' 初始化关闭按钮位置  
With ImageExit(j)  
.Left = 5520  
.Top = 0  
End With  
Next
```

```
' 初始化关闭按钮  
ImageExit(0).Visible = True  
ImageExit(1).Visible = False  
ImageExit(2).Visible = False  
End Sub
```

#### 鼠标在窗体中移动

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
' 关闭按钮  
ImageExit(0).Visible = True  
ImageExit(1).Visible = False  
ImageExit(2).Visible = False  
End Sub
```

#### 退出窗体

```
Private Sub ImageExit_Click(Index As Integer)  
Unload Me  
End Sub
```

#### 鼠标在退出函数上移动

```
Private Sub ImageExit_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    ' 设置退出按钮状态
```

```
    ImageExit(0).Visible = False
```

```
    ImageExit(1).Visible = True
```

```
    ImageExit(2).Visible = False
```

```
End Sub
```

#### 鼠标在退出按钮上按下

```
Private Sub ImageExit_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    ' 设置退出按钮状态
```

```
    ImageExit(0).Visible = False
```

```
    ImageExit(1).Visible = False
```

```
    ImageExit(2).Visible = True
```

```
End Sub
```

#### 鼠标在退出按钮上抬起

```
Private Sub ImageExit_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    ' 设置退出点击按钮状态
```

```
    Unload Me
```

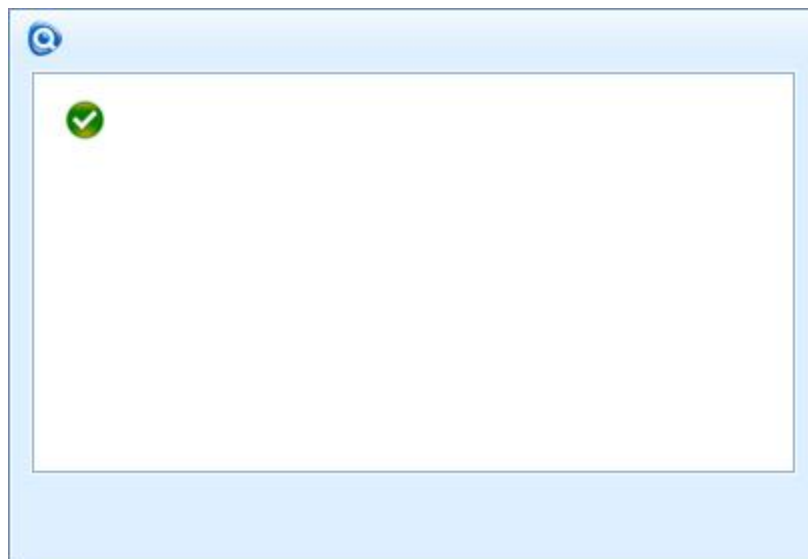
```
End Sub
```

#### 本窗体的换肤函数

```
Public Function ReSkinMe()
```

```
With Me
```

```
    ' 加载窗体背景图片，图片是：
```



```
.Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\ScanResult.bmp")

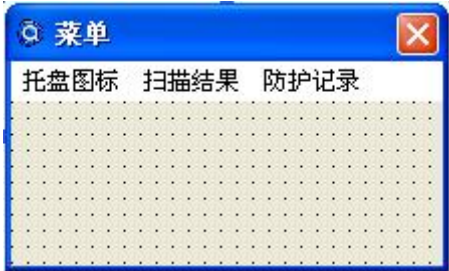
' 加载图片模拟的三态退出按钮图片
.ImageExit(0).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit0.bmp")
.ImageExit(1).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit1.bmp")
.ImageExit(2).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit2.bmp")
End With
End Function
```

5.3.9. 右键菜单设计窗体

窗体：  
FormMenu

功能：  
提供软件中各种使用的菜单，共三个菜单：托盘图标右键菜单、主界面中扫描完成后病毒列表中的右键菜单、主界面防护记录中的右键菜单。

界面：



托盘图标右键菜单说明：

菜单名称	标题	功能
m_Setting	设置	托盘区图标的右键菜单，用于调出“设置”界面。
m_Quarantine	隔离文件管理	托盘区图标的右键菜单，用于调出“隔离文件管理”界面。
m_BlackAndWhiteList	黑白名单管理	托盘区图标的右键菜单，用于调出“黑白名单管理”界面。
m_Update	升级	托盘区图标的右键菜单，用于调出“升级”界面。
m_Exit	退出	托盘区图标的右键菜单，用于退出程序。

扫描结果右键菜单说明：

菜单名称	标题	功能
m_ClearAll	清除所有	扫描结果网格控件的右键菜单，用于清除所有扫描出的病毒。
m_ClearSelected	清除选中	扫描结果网格控件的右键菜单，用于清除选中病毒。
m_QuarantineAll	隔离所有	扫描结果网格控件的右键菜单，用于隔离所有病毒。
m_QuarantineSelected	隔离选中	扫描结果网格控件的右键菜单，用于隔离选中的病毒。

m_location	文件定位	扫描结果网格控件的右键菜单，用于定位到病毒位置。
------------	------	--------------------------

防护记录右键菜单说明：

菜单名称	标题	功能
m_ClearAllShieldLog	清空所有	防护记录网格控件中的右键菜单，用于清空所有记录。
m_ClearSelectedShieldLog	清空选中记录	防护记录网格控件中的右键菜单，用于清空选中的记录。
m_ShieldLogLocation	文件定位	防护记录网格控件中的右键菜单，用于定位到文件位置。

代码：

```
Option Explicit
' api 声明
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hWnd As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
Private Declare Function DeleteFile Lib "kernel32" Alias "DeleteFileA" (ByVal lpFileName As String) As Long
Private Declare Function CopyFile Lib "kernel32" Alias "CopyFileA" (ByVal lpExistingFileName As String, ByVal lpNewFileName As String, ByVal bFailIfExists As Long) As Long
```

黑白名单管理

```
Private Sub m_BlackAndWhiteList_Click()
FormBlackAndWhiteList.Show vbModal
End Sub
```

清除所有已经检测到的病毒

```
Private Sub m_ClearAll_Click()
Dim i As Long

' 遍历主窗体扫描结果窗体中的数据
For i = 1 To FormScan.ListViewScanResult.ListItems.Count
Dim lDeleteFileReturn As Long
lDeleteFileReturn = DeleteFile(FormScan.ListViewScanResult.ListItems(i).SubItems(1))
If lDeleteFileReturn <> 0 Then
FormScan.ListViewScanResult.ListItems(i).SubItems(3) = "已清除"
Else
FormScan.ListViewScanResult.ListItems(i).SubItems(3) = "清除失败"
End If
Next i
End Sub
```

清空防护记录

```
Private Sub m_ClearAllShieldLog_Click()
FormScan.ListViewShieldLog.ListItems.Clear
End Sub
```

#### 清除选中的病毒

```
Private Sub m_ClearSelected_Click()
Dim i As Long
For i = 1 To FormScan.ListViewScanResult.ListItems.Count

' 判断是否被选中
If FormScan.ListViewScanResult.ListItems(i).Checked = True Then
Dim lDeleteFileReturn As Long
lDeleteFileReturn = DeleteFile(FormScan.ListViewScanResult.ListItems(i).SubItems(1))
If lDeleteFileReturn <> 0 Then
FormScan.ListViewScanResult.ListItems(i).SubItems(3) = "已清除"
Else
FormScan.ListViewScanResult.ListItems(i).SubItems(3) = "清除失败"
End If
End If
Next i
End Sub
```

#### 删除选中的防护记录

```
Private Sub m_ClearSelectedShieldLog_Click()
FormScan.ListViewShieldLog.ListItems.Remove FormScan.ListViewShieldLog.SelectedItem.Index
End Sub
```

#### 退出软件

```
Private Sub m_Exit_Click()

' 调用
TrueEnd
End Sub
```

#### 连接到官方网站

```
Private Sub m_Homepage_Click()
Call ShellExecute(Me.hWnd, "open", "http://www.haiqi.cn/", 0, 0, 1)
End Sub
```

#### 定位文件

```
Private Sub m_location_Click()
Dim sFilePath As String
sFilePath
FormScan.ListViewScanResult.ListItems(FormScan.ListViewScanResult.SelectedItem.Index).SubItems(1)
If Dir(sFilePath) <> "" Then
```

```
Shell "explorer.exe /select, " & sFilePath, vbNormalFocus
End If
End Sub
```

#### 调出隔离文件管理窗体

```
Private Sub m_Quarantine_Click()
FormQuarantine.Show vbModal
End Sub
```

#### 隔离所有主窗体扫描结果中的病毒文件

```
Private Sub m_QuarantineAll_Click()
Dim i As Long
For i = 1 To FormScan.ListViewScanResult.ListItems.Count
DoEvents

' 确保当前路径包含"\
Dim sTempAppPath As String
If Right(App.Path, 1) = "\" Then
sTempAppPath = App.Path
Else
sTempAppPath = App.Path & "\"
End If

' 目标复制文件名
Dim sTargetFileName As String
sTargetFileName = sTempAppPath & "quarantine\" & RndChr(8) & ".tmp"

' 复制文件
Dim lCopyFileReturn As Long
lCopyFileReturn = CopyFile(FormScan.ListViewScanResult.ListItems(i).SubItems(1), sTargetFileName, 0)

' 复制成功
If lCopyFileReturn <> 0 Then

' 清除病毒
Dim lDeleteFileReturn As Long
lDeleteFileReturn = DeleteFile(FormScan.ListViewScanResult.ListItems(i).SubItems(1))
If lDeleteFileReturn <> 0 Then
FormScan.ListViewScanResult.ListItems(i).SubItems(3) = "已隔离"

' 隔离记录文件
Call WriteQuarantineFile(FormScan.ListViewScanResult.ListItems(i).SubItems(1), sTargetFileName)

Else
FormScan.ListViewScanResult.ListItems(i).SubItems(3) = "隔离失败"
End If

' 复制失败，隔离失败
```

```

Else
FormScan.ListViewScanResult.ListItems(i).SubItems(3) = "隔离失败"
End If
Next i
End Sub

```

### 隔离选中的文件

```

Private Sub m_QuarantineSelected_Click()
Dim i As Long
For i = 1 To FormScan.ListViewScanResult.ListItems.Count
DoEvents

' 判断是否被选中
If FormScan.ListViewScanResult.ListItems(i).Checked = True Then

' 确保当前路径包含"\
Dim sTempAppPath As String
If Right(App.Path, 1) = "\" Then
sTempAppPath = App.Path
Else
sTempAppPath = App.Path & "\"
End If

' 目标复制文件名
Dim sTargetFileName As String
sTargetFileName = sTempAppPath & "quarantine\" & RndChr(8) & ".tmp"

' 复制文件
Dim lCopyFileReturn As Long
lCopyFileReturn = CopyFile(FormScan.ListViewScanResult.ListItems(i).SubItems(1), sTargetFileName, 0)

' 复制成功
If lCopyFileReturn <> 0 Then

' 清除病毒
Dim lDeleteFileReturn As Long
lDeleteFileReturn = DeleteFile(FormScan.ListViewScanResult.ListItems(i).SubItems(1))
If lDeleteFileReturn <> 0 Then
FormScan.ListViewScanResult.ListItems(i).SubItems(3) = "已隔离"

' 隔离记录文件
Call WriteQuarantineFile(FormScan.ListViewScanResult.ListItems(i).SubItems(1), sTargetFileName)
Else
FormScan.ListViewScanResult.ListItems(i).SubItems(3) = "隔离失败"
End If

' 复制失败，隔离失败
Else
FormScan.ListViewScanResult.ListItems(i).SubItems(3) = "隔离失败"

```

```
End If
End If
Next i
End Sub
```

#### 调出软件设置界面

```
Private Sub m_Setting_Click()
FormSetting.Show vbModal
End Sub
```

#### 定位防护记录中的文件

```
Private Sub m_ShieldLogLocation_Click()
Dim sFilePath As String
sFilePath
FormScan.ListViewShieldLog.ListItems(FormScan.ListViewShieldLog.SelectedItem.Index).SubItems(2)
If Dir(sFilePath) <> "" Then
Shell "explorer.exe /select, " & sFilePath, vbNormalFocus
End If
End Sub
```

#### 调出升级界面

```
Private Sub m_Update_Click()
FormUpdate.Show vbModal
End Sub
```

### 5.3.10. 病毒自定义扫描窗体

窗体:

FormSelectFolders

功能:

进行自定义扫描时，让用户选择扫描目标的窗体，比如可以选择任意驱动器、目录。

界面设计:





窗体中包含的控件及功能说明：

控件名称	类别	功能
TreeViewScanTarget	树型控件	
CommandOK	按钮控件	
ImageExit	图片控件	
TimerSelectTreeViewCheck	定时器控件	

代码：

```
Option Explicit
' api 声明
Private Declare Function GetLogicalDriveStrings Lib "kernel32" Alias "GetLogicalDriveStringsA" (ByVal nBufferLength As Long, ByVal lpBuffer As String) As Long
Private Declare Function GetDriveType Lib "kernel32" Alias "GetDriveTypeA" (ByVal nDrive As String) As Long
Private Declare Function FindFirstFile Lib "kernel32" Alias "FindFirstFileA" (ByVal lpFileName As String, lpFindFileData As WIN32_FIND_DATA) As Long
Private Declare Function FindNextFile Lib "kernel32" Alias "FindNextFileA" (ByVal hFindFile As Long, lpFindFileData As WIN32_FIND_DATA) As Long
Private Declare Function FindClose Lib "kernel32" (ByVal hFindFile As Long) As Long
Private Declare Function ReleaseCapture Lib "user32" () As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long, ByVal lParam As Any) As Long
```

’ 常量定义

```
Private Const MAX_PATH As Long = 260
Private Const DRIVE_REMOVABLE As Long = 2
Private Const DRIVE_FIXED As Long = 3
Private Const DRIVE_REMOTE As Long = 4
Private Const DRIVE_CDROM As Long = 5
Private Const DRIVE_RAMDISK As Long = 6
```

```
Private Const TH32CS_SNAPPROCESS = &H2&
Private Const TH32CS_SNAPMODULE = &H8
Private Const TH32CS_SNAPHEAPLIST = &H1
Private Const TH32CS_SNAPTHREAD = &H4
Private Const TH32CS_SNAPALL = (TH32CS_SNAPHEAPLIST Or TH32CS_SNAPPROCESS Or TH32CS_SNAPTHREAD Or
TH32CS_SNAPMODULE)
```

```
Private Const FILE_ATTRIBUTE_DIRECTORY = &H10
```

’ 自定义类型

```
Private Type FILETIME
dwLowDateTime As Long
dwHighDateTime As Long
End Type
```

```
Private Type WIN32_FIND_DATA
dwFileAttributes As Long
ftCreationTime As FILETIME
ftLastAccessTime As FILETIME
ftLastWriteTime As FILETIME
nFileSizeHigh As Long
nFileSizeLow As Long
dwReserved0 As Long
dwReserved1 As Long
cFileName As String * MAX_PATH
cAlternate As String * 14
End Type
```

**单击确定按钮**

```
Private Sub CommandOK_Click()
Me.Hide
End Sub
```

**鼠标在窗体中移动**

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

’ 关闭按钮

```
ImageExit(0).Visible = True
ImageExit(1).Visible = False
ImageExit(2).Visible = False
```

```
End Sub
```

#### 窗体卸载函数

```
Private Sub Form_Unload(Cancel As Integer)
If bEnableUnloadForm = False Then
Cancel = 1
Me.Hide
End If
End Sub
```

#### 点击退出按钮

```
Private Sub ImageExit_Click(Index As Integer)
Me.Hide
End Sub
```

#### 用定时器控件，控制复选状态，选中所有节点

```
Private Sub TimerSelectTreeviewCheck_Timer()
If Me.Visible = False Then Exit Sub
Dim i As Long

' 选中所有节点
For i = 1 To TreeViewScanTarget.Nodes.Count
DoEvents
TreeViewScanTarget.Nodes(i).Checked = True
Next
TimerSelectTreeviewCheck.Enabled = False
End Sub
```

#### 窗体启动函数

```
Private Sub Form_Load()

TreeViewScanTarget.Nodes.Add , , "Lord", "我的电脑", 1
Dim sDrive As String
sDrive = String(256, Chr(0))
Dim sDriveID As String

' 取得磁盘串
Call GetLogicalDriveStrings(256, sDrive)
Dim i As Long

' 返回光盘盘符到数组, 注意这里 step 是 4
For i = 1 To 100 Step 4
sDriveID = Mid(sDrive, i, 3)
If sDriveID = Chr(0) & Chr(0) & Chr(0) Then

' 没有盘符, 即时退出循环
```

```

Exit For
Else

' 移动设置(u 盘)
If GetDriveType(sDriveID) = 2 Then
TreeViewScanTarget.Nodes.Add "Lord", tvwChild, sDriveID, Left(sDriveID, 2), 5
End If

' 软盘
If GetDriveType(sDriveID) = 4 Then
TreeViewScanTarget.Nodes.Add "Lord", tvwChild, sDriveID, Left(sDriveID, 2), 2
End If

' 硬盘
If GetDriveType(sDriveID) = 3 Then
TreeViewScanTarget.Nodes.Add "Lord", tvwChild, sDriveID, Left(sDriveID, 2), 3
End If

' 光盘
If GetDriveType(sDriveID) = 5 Then
TreeViewScanTarget.Nodes.Add "Lord", tvwChild, sDriveID, Left(sDriveID, 2), 4
End If

' RAM 盘
If GetDriveType(sDriveID) = 6 Then
TreeViewScanTarget.Nodes.Add "Lord", tvwChild, sDriveID, Left(sDriveID, 2), 6
End If
End If
Next i
Dim j As Long
For j = 0 To 2

' 初始化关闭按钮位置
With ImageExit(j)
.Left = 4800
.Top = 0
End With
Next

' 初始化图片模拟的关闭按钮
ImageExit(0).Visible = True
ImageExit(1).Visible = False
ImageExit(2).Visible = False

' 设置根节为选中状态
TreeViewScanTarget.Nodes(1).Checked = True

' 设置根节为展开状态
TreeViewScanTarget.Nodes(1).Expanded = True

' 设置子节选中状态

```

```
CheckChildNodes TreeViewScanTarget.Nodes(1)
CheckParentNodes TreeViewScanTarget.Nodes(1)
DoEvents
```

```
’ 设置皮肤界面
ReSkinMe
TimerSelectTreeviewCheck.Enabled = True
End Sub
```

#### **Treeview 单击事件**

```
Private Sub TreeViewScanTarget_Click()

’ 处理节点选中状态
CheckChildNodes TreeViewScanTarget.SelectedItem
CheckParentNodes TreeViewScanTarget.SelectedItem
End Sub
```

#### **Treeview 节点单击事件**

```
Private Sub TreeViewScanTarget_NodeClick(ByVal Node As MSComctlLib.Node)

’ 转移 Treeview 中节点焦点
CheckChildNodes TreeViewScanTarget.SelectedItem
CheckParentNodes TreeViewScanTarget.SelectedItem
End Sub
```

#### **Treeview 节点展开事件**

```
’ 获取节点所对应的子目录，增加子节点
Private Sub TreeViewScanTarget_Expand(ByVal Node As MSComctlLib.Node)

’ 如果 Treeview 无内容则退出
If TreeViewScanTarget.Nodes.Count = 0 Then Exit Sub
Dim i As Long

’ 如果选中的节点无子节点
If Node.Children = 0 Then
Call GetSubFolders(TreeViewScanTarget, Node)

’ 如果有子节点
Else
Dim uNode As Node

’ 节点的第一子节点
Set uNode = TreeViewScanTarget.Nodes(Node.Index).Child

’ 遍历所有子节点
For i = 1 To Node.Children
DoEvents
```

```

' 如果子节点没有子节点，则取子节点的下级目录，增加子节点
If uNode.Children = 0 Then
Call GetSubFolders(TreeViewScanTarget, uNode)
End If
Set uNode = uNode.Next
Next
End If
End Sub

```

#### 处理子节点选中状态

```

Private Function CheckChildNodes(ByVal Node As MSComctlLib.Node)
Dim i As Long
Dim uChildNode As Node

' 如果没有子节点，则退出
If Node.Children = 0 Then
Exit Function
End If
If Node.Checked = True Then

' 选中
Set uChildNode = Node.Child
For i = 1 To Node.Children
uChildNode.Checked = True
If uChildNode.Children <> 0 Then
CheckChildNodes uChildNode
End If
Set uChildNode = uChildNode.Next
Next
Else

' 不选中
Set uChildNode = Node.Child
For i = 1 To Node.Children
uChildNode.Checked = False
If uChildNode.Children <> 0 Then
CheckChildNodes uChildNode
End If
Set uChildNode = uChildNode.Next
Next
End If
End Function

```

#### 处理父节点选中状态

```

Private Function CheckParentNodes(ByVal Node As MSComctlLib.Node)
Dim i As Long
Dim uParentNode As Node

```

```

' 同层 node 个数
Dim lSameLevelNodes As Long
Dim uSameLevelNode As Node

' 如果没有子节点，则退出
If Node.Root = Node Then
Exit Function
End If
If Node.Checked = True Then

' 选中
Set uParentNode = Node.Parent
uParentNode.Checked = True
Do While uParentNode <> Node.Root
Set uParentNode = uParentNode.Parent
uParentNode.Checked = True
Loop
Else

' 检查是否同级节点都是不选中状态
If Node.Parent.Children > 0 Then
lSameLevelNodes = Node.Parent.Children
Set uSameLevelNode = Node.Parent.Child
For i = 1 To lSameLevelNodes
If uSameLevelNode.Checked = True Then
Exit Function
End If
Set uSameLevelNode = uSameLevelNode.Next
Next
End If

' 不选中
Set uParentNode = Node.Parent
uParentNode.Checked = False
Do While uParentNode <> Node.Root
Set uParentNode = uParentNode.Parent
uParentNode.Checked = False
Loop
End If
End Function

```

#### **TreeView 节点选中/反选事件**

```
Private Sub TreeViewScanTarget_NodeCheck(ByVal Node As MSComctlLib.Node)
```

```

' 转移 Treeview 中节点焦点
TreeViewScanTarget.SelectedItem = Node

```

```

' 处理节点选中状态

```

```
CheckChildNodes Node
CheckParentNodes Node
End Sub
```

#### 取得目录或驱动器下子目录

```
' 参数:
'uTreeView, 放置目录的 Treeview 控件
'uParentNode, Treeview 选中节点, 包含文件路径, 本函数即取该目录下的子目录
Public Function GetSubFolders(uTreeView As Control, ByVal uParentNode As Node)
Dim sPath As String

' 检查最后字母是否是 “\”
' Treeview 以 “\” 分隔每个节点, 所以: “我的电脑” + “\” 长度为 5
If Right(uParentNode.FullPath, 1) <> “\” Then
sPath = Right(uParentNode.FullPath, Len(uParentNode.FullPath) - 5) & “\”
Else
sPath = Right(uParentNode.FullPath, Len(uParentNode.FullPath) - 5)
End If
Dim lRet As Long
Dim uWFD As WIN32_FIND_DATA

' 查找目录下文件
lRet = FindFirstFile(sPath & “*.*” & Chr(0), uWFD)
If lRet <> -1 Then

' 不为. 或.. 且是目录
If (TrimNull(uWFD.cFileName) <> “.”) And (TrimNull(uWFD.cFileName) <> “..”) And (uWFD.dwFileAttributes
And vbDirectory) Then

' 增加子节点, 添加目录
Call uTreeView.Nodes.Add(uParentNode.Key, tvwChild, sPath & TrimNull(uWFD.cFileName),
TrimNull(uWFD.cFileName), 7)
End If
While FindNextFile(lRet, uWFD)

' 不为. 或.. 且是目录
If (TrimNull(uWFD.cFileName) <> “.”) And (TrimNull(uWFD.cFileName) <> “..”) And (uWFD.dwFileAttributes
And vbDirectory) Then

' 增加子节点, 添加目录
Call uTreeView.Nodes.Add(uParentNode.Key, tvwChild, sPath & TrimNull(uWFD.cFileName),
TrimNull(uWFD.cFileName), 7)
End If
Wend
End If
Call FindClose(lRet)
End Function
```

#### 鼠标在窗体中按下



```

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

' 按下鼠标左键
If Button = vbLeftButton Then

' 为当前的应用程序释放鼠标捕获
ReleaseCapture

' 移动窗体
SendMessage Me.hWnd, &HA1, 2, 0
End If
End Sub

```

#### 鼠标在退出按钮上移动

```

Private Sub ImageExit_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

' 退出按钮状态
ImageExit(0).Visible = False
ImageExit(1).Visible = True
ImageExit(2).Visible = False
End Sub

```

#### 鼠标在退出按钮上按下

```

Private Sub ImageExit_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

' 退出按钮状态
ImageExit(0).Visible = False
ImageExit(1).Visible = False
ImageExit(2).Visible = True

' 是否进行扫描的标识
bDoCustmerScan = False

End Sub

```

#### 鼠标在退出按钮上抬起

```

Private Sub ImageExit_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

' 退出点击按钮
Me.Hide
End Sub

```

#### 本窗体的界面换肤函数

```
Public Function ReSkinMe()
```

```
With Me
```

```
' 窗体的背景图片
```

```
.Picture = LoadPicture(App.Path & "\\Skin\\" & sSkin & "\\SelectFolders.bmp")
```

```
' 上面加载的图片如下:
```



```
' 退出按钮的三态图片
```

```
' 正常状态下的图片
```

```
.ImageExit(0).Picture = LoadPicture(App.Path & "\\Skin\\" & sSkin & "\\Exit0.bmp")
```

```
' 鼠标移动时的图片
```

```
.ImageExit(1).Picture = LoadPicture(App.Path & "\\Skin\\" & sSkin & "\\Exit1.bmp")
```

```
' 鼠标移动时的图片
```

```
.ImageExit(2).Picture = LoadPicture(App.Path & "\\Skin\\" & sSkin & "\\Exit2.bmp")
```

```
End With
```

```
End Function
```

### 5.3.11. 软件功能设置窗体

窗体:

FormSetting

功能:

对程序中各种功能进行配置和设置的窗体。

界面设计：



窗体中包含的控件及功能说明：

控件名称	类别	功能
ImageGeneral	图片控件	用图片控件模拟的一般设置选项卡控件。
ImageScan	图片控件	用图片控件模拟的扫描设置选项卡控件。
ImageShield	图片控件	用图片控件模拟的防护设置选项卡控件。
ImageUpdate	图片控件	用图片控件模拟的升级设置选项卡控件。
CommandOK	按钮控件	确定按钮。
CommandCancel	按钮控件	取消按钮。
ImageExit	图片控件	用图片模拟的退出按钮。
CheckLowCPU	复选框控件	扫描时是否降低进程 CPU 优先级。
OptionScanAllFiles	单选框控件	进行扫描时扫描全部文件。
OptionScanPeFiles	单选框控件	进行扫描时只扫描 PE 文件。
OptionClear	单选框控件	扫描到病毒时自动清除。
OptionAlert	单选框控件	扫描到病毒时提示用户。
OptionIgnore	单选框控件	扫描到病毒时自动忽略。
OptionQuarantin	单选框控件	扫描到病毒时自动隔离。

e		
CheckAutoUpdate	复选框控件	是否启用自动更新。
ComboAutoUpdateInterval	下拉框控件	自动升级频率。
OptionNormalLevel	单选框控件	防护级别设置为普通级别。
OptionHighLevel	单选框控件	防护级别设置为高级别。
CheckEnableAlert	复选框控件	启用消息提示窗口。
CheckAutoCloseAlert	复选框控件	自动关闭消息提示窗口。
ComboAutoCloseAlertInterval	下拉框控件	自动关闭消息窗口时间。
CheckCloudSecurity	复选框控件	是否启用云安全。
CheckSafeGuard	复选框控件	是否启用自我保护。
CheckAutoRun	复选框控件	是否启用开机自启动。
CheckAutoTray	复选框控件	软件启动后是否自动最小化。
CheckEnableRightClickMenu	复选框控件	是否启用文件或文件夹右键菜单关联。

软件功能设置配置文件说明：

本软件中使用 Settings.ini 来记录各种配置信息。事例内容如下：

```
[Normal]
SafeGuard=1
AutoRun=1
AutoTray=1
RightClickMenu=0
Skin=天空蓝
[Scan]
CheckLowCPU=1
ScanAllFiles=True
ScanPeFiles=False
ClearVirus=False
AlertVirus=True
IgnoreVirus=False
QuarantineVirus=False
[Shield]
NormalLevel=True
HighLevel=False
CloudSecurity=0
EnableAlertMessage=0
AutoCloseAlertMessage=0
AutoCloseAlertMessageInterval=3
[Update]
AutoUpdate=1
AutoUpdateInterval=1
[History]
LastScanUsedTime=0:00:00
LastScanFiles=0
```

LastDetectedSpywares=0

各项内容说明：

项目	功能说明
SafeGuard	是否启用软件自我保护，1 为启用，0 为不启用。
AutoRun	是否让软件在系统启动后自动运行，1 为启用，0 为不启用。
AutoTray	软件启动后是否自动缩小到系统托盘区，1 为启用，0 为不启用。
RightClickMenu	是否给文件夹和文件添加右键病毒扫描功能，1 为启用，0 为不启用。
Skin=Blue	软件皮肤，本例中皮肤为当前目录下的 Blue 目录中的皮肤文件。
CheckLowCPU	扫描时是否使用低 CPU 使用率，1 为启用，0 为不启用。
ScanAllFiles	扫描病毒时是否扫描所有文件，True 为是，False 为否。
ScanPeFiles	扫描病毒时是否只扫描 PE 文件，True 为是，False 为否。
ClearVirus	扫描到病毒后是否自动清除，True 为是，False 为否。
AlertVirus	扫描到病毒后是否向用户发出提示，True 为是，False 为否。
IgnoreVirus	扫描到病毒后是否忽略不处理，True 为是，False 为否。
QuarantineVirus	扫描到病毒后是否自动隔离，True 为是，False 为否。
NormalLevel	软件防护级别是否记用普通级别，True 为是，False 为否。启用普通级别时，当拦截到文件启动后，先检测是否是病毒木马，如果是则直接阻止；如果没有检测到，则自动放行。
HighLevel	软件防护级别是否记用高级别，True 为是，False 为否。启用高级别时，当拦截到文件启动后，先检测是否是病毒木马，如果是则直接阻止；如果没有检测到，则向用户发出提示，向用户征求是否允许运行。
CloudSecurty	是否启用云安全，1 为启用，0 为不启用。如果启用了云安全，则在拦截文件启动后，在本地检测过病毒木马后，并且没有检测到的情况下，还会连接到云安全服务器进行检测。方法即上文中所讲述过的。
EnableAlertMessage	拦截到病毒木马启动行为时，是否发用户发出提示，即在右下角弹出提示消息框。1 为启用，0 为不启用。
AutoCloseAlertMessage	弹出病毒木马拦截提示框后，是否在一定时间后自动关闭。1 为启用，0 为不启用。
AutoCloseAlertMessageInterval	自动关闭病毒木马拦截提示框的时间，时间为秒。例：如果设置值为 3，则窗体在 3 秒后自动关闭。
AutoUpdate	是否启用自动更新软件和病毒库。1 为启用，0 为不启用。
AutoUpdateInterval	自动更新软件和病毒库的频率，时间为小时。例：如果设置为 1，则软件在运行每 1 小时后都会自动检测是否有新的软件版本或病毒库需要更新，如果有，则自动下载更新。
LastScanUsedTime	记录软件上次进行病毒扫描的时间。
LastScanFiles	记录软件上次病毒扫描时扫描了多少文件。
LastDetectedSpywares=0	记录软件上次病毒扫描时检测到了多少病毒木马。

代码：

```
Option Explicit
'api 声明
Private Declare Function ReleaseCapture Lib "user32" () As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long, ByVal wMsg As Long, ByVal lParam As Any) As Long
```

点击取消按钮

```
Private Sub CommandCancel_Click()  
Unload Me  
End Sub
```

#### 确定按钮，保存设置信息

```
Private Sub CommandOK_Click()  
Dim sSettingsFile As String  
  
' 软件设置记录文件  
If Right(App.Path, 1) = "\" Then  
sSettingsFile = App.Path & "Settings.ini"  
Else  
sSettingsFile = App.Path & "\Settings.ini"  
End If  
  
' 自我保护设置  
Call WriteIni(sSettingsFile, "Normal", "SafeGuard", CheckSafeGuard.value)  
  
' 自启动设置  
Call WriteIni(sSettingsFile, "Normal", "AutoRun", CheckAutoRun.value)  
If CheckAutoRun.value = 1 Then  
SetAutoRun  
Else  
StopAutoRun  
End If  
  
' 自动放入托盘图标  
Call WriteIni(sSettingsFile, "Normal", "AutoTray", CheckAutoTray.value)  
  
' 扫描时减少 CPU 占用  
Call WriteIni(sSettingsFile, "Scan", "CheckLowCPU", CheckLowCPU.value)  
  
' 扫描所有文件  
Call WriteIni(sSettingsFile, "Scan", "ScanAllFiles", OptionScanAllFiles.value)  
  
' 扫描可执行文件  
Call WriteIni(sSettingsFile, "Scan", "ScanPeFiles", OptionScanPeFiles.value)  
  
' 自动清除  
Call WriteIni(sSettingsFile, "Scan", "ClearVirus", OptionClear.value)  
  
' 询问  
Call WriteIni(sSettingsFile, "Scan", "AlertVirus", OptionAlert.value)  
  
' 忽略  
Call WriteIni(sSettingsFile, "Scan", "IgnoreVirus", OptionIgnore.value)  
  
' 隔离
```

```

Call WriteIni(sSettingsFile, "Scan", "QuarantineVirus", OptionQuarantine.value)

' 防护级别 - 正常
Call WriteIni(sSettingsFile, "Shield", "NormalLevel", OptionNormalLevel.value)

' 防护级别 - 高
Call WriteIni(sSettingsFile, "Shield", "HighLevel", OptionHighLevel.value)

' 云安全
Call WriteIni(sSettingsFile, "Shield", "CloudSecurity", CheckCloudSecurity.value)

' 防护提醒
Call WriteIni(sSettingsFile, "Shield", "EnableAlertMessage", CheckEnableAlert.value)

' 自动关闭防护提示消息
Call WriteIni(sSettingsFile, "Shield", "AutoCloseAlertMessage", CheckAutoCloseAlert.value)

' 自动关闭防护提示消息频率
Call WriteIni(sSettingsFile, "Shield", "AutoCloseAlertMessageInterval",
ComboAutoCloseAlertInterval.Text)

' 自动更新
Call WriteIni(sSettingsFile, "Update", "AutoUpdate", CheckAutoUpdate.value)

' 自动更新频率
Call WriteIni(sSettingsFile, "Update", "AutoUpdateInterval", ComboAutoUpdateInterval.Text)

' 关联右键菜单
Call WriteIni(sSettingsFile, "Normal", "RightClickMenu", CheckEnableRightClickMenu.value)
If CheckEnableRightClickMenu.value = 1 Then

' 添加关联
AddFileRightClickMenu
AddFolderRightClickMenu
Else

' 删除关联
DeleteRightClickMenu
End If
DoEvents

' 更改扫描界面设置状态
Dim lAutoUpdate As Long
lAutoUpdate = ReadIni(sSettingsFile, "Update", "AutoUpdate")

' 自动更新设置
If lAutoUpdate = 1 Then
With FormScan
.LabelAutoUpdateState.Caption = "自动升级：已启用"
.ImageAutoUpdate(0).Visible = False
.ImageAutoUpdate(1).Visible = True

```

```

End With
Else
With FormScan
.LabelAutoUpdateState.Caption = "自动升级：已禁用"
.ImageAutoUpdate(0).Visible = True
.ImageAutoUpdate(1).Visible = False
End With
End If

' 云安全启用状态
Dim lCloudSecurty As Long
lCloudSecurty = ReadIni(sSettingsFile, "Shield", "CloudSecurty")
If lCloudSecurty = 1 Then
With FormScan
.LabelCloud.Caption = "云安全防护：已启用"
.ImageCloud(0).Visible = False
.ImageCloud(1).Visible = True
End With
Else
With FormScan
.LabelCloud.Caption = "云安全防护：未启用"
.ImageCloud(0).Visible = True
.ImageCloud(1).Visible = False
End With
End If

' 防护级别
Dim lShieldLevel As Boolean
lShieldLevel = ReadIni(sSettingsFile, "Shield", "NormalLevel")
If lShieldLevel = True Then
With FormScan
.LabelShieldLevel.Caption = "系统防护级别：正常"
.ImageShieldLevel(0).Visible = False
.ImageShieldLevel(1).Visible = True
End With
Else
With FormScan
.LabelShieldLevel.Caption = "系统防护级别：严格"
.ImageShieldLevel(0).Visible = True
.ImageShieldLevel(1).Visible = False
End With
End If

' 开机自启动
Dim lAutorun As Long
lAutorun = ReadIni(sSettingsFile, "Normal", "AutoRun")
If lAutorun = 1 Then
With FormScan
.LabelAutorun.Caption = "开机自启动：已启用"
.ImageAutoRun(0).Visible = False
.ImageAutoRun(1).Visible = True

```



```

End With
Else
With FormScan
.LabelAutorun.Caption = "开机自启动：未启用"
.ImageAutoRun(0).Visible = True
.ImageAutoRun(1).Visible = False
End With
End If

' 自动最小化
Dim lAutomin As Long
lAutomin = ReadIni(sSettingsFile, "Normal", "AutoTray")
If lAutomin = 1 Then
With FormScan
.LabelAutoMin.Caption = "开机最小化：已启用"
.ImageAutoMin(0).Visible = False
.ImageAutoMin(1).Visible = True
End With
Else
With FormScan
.LabelAutoMin.Caption = "开机最小化：未启用"
.ImageAutoMin(0).Visible = True
.ImageAutoMin(1).Visible = False
End With
End If

' 自我保护
Dim lSafeGuard As Long
lSafeGuard = ReadIni(sSettingsFile, "Normal", "SafeGuard")
If lSafeGuard = 1 Then
With FormScan
.LabelShieldSelf.Caption = "自我保护：已启用"
.ImageShieldSelf(0).Visible = False
.ImageShieldSelf(1).Visible = True
End With
Else
With FormScan
.LabelShieldSelf.Caption = "自我保护：未启用"
.ImageShieldSelf(0).Visible = True
.ImageShieldSelf(1).Visible = False
End With
End If

' 右键菜单
Dim lRightClickMenu As Long
lRightClickMenu = ReadIni(sSettingsFile, "Normal", "RightClickMenu")
If lRightClickMenu = 1 Then
With FormScan
.LabelRightClickMenu.Caption = "关联右键菜单：已启用"
.ImageRightClickMenu(0).Visible = False
.ImageRightClickMenu(1).Visible = True

```

```

End With
Else
With FormScan
.LabelRightClickMenu.Caption = "关联右键菜单：未启用"
.ImageRightClickMenu(0).Visible = True
.ImageRightClickMenu(1).Visible = False
End With
End If

'发现病毒时自动清除
Dim bClearVirus As Boolean
bClearVirus = ReadIni(sSettingsFile, "Scan", "ClearVirus")

'发现病毒时提示
Dim bAlertVirus As Boolean
bAlertVirus = ReadIni(sSettingsFile, "Scan", "AlertVirus")

'发现病毒时忽略
Dim bIgnoreVirus As Boolean
bIgnoreVirus = ReadIni(sSettingsFile, "Scan", "IgnoreVirus ")

'发现病毒时隔离
Dim bQuarantineVirus As Boolean
bQuarantineVirus = ReadIni(sSettingsFile, "Scan", "QuarantineVirus")
If bClearVirus = True Then
FormScan.LabelActionWhenDetectVirus.Caption = "检测到病毒处理方式：自动清除"
End If
If bAlertVirus = True Then
FormScan.LabelActionWhenDetectVirus.Caption = "检测到病毒处理方式：询问"
End If
If bIgnoreVirus = True Then
FormScan.LabelActionWhenDetectVirus.Caption = "检测到病毒处理方式：忽略"
End If
If bQuarantineVirus = True Then
FormScan.LabelActionWhenDetectVirus.Caption = "检测到病毒处理方式：隔离"
End If
Unload Me
MsgBox "设置已保存。", vbInformation
End Sub

```

#### 窗体启动函数

```

Private Sub Form_Load()

' 设置皮肤
ReSkinMe

' 初始化控件位置
PictureGeneral.Left = 480
PictureGeneral.Top = 1200

```

```

PictureScan.Left = PictureGeneral.Left
PictureScan.Top = PictureGeneral.Top
PictureShield.Left = PictureGeneral.Left
PictureShield.Top = PictureGeneral.Top
PictureUpdate.Left = PictureGeneral.Left
PictureUpdate.Top = PictureGeneral.Top
PictureGeneral.Visible = True
PictureScan.Visible = False
PictureShield.Visible = False
PictureUpdate.Visible = False
ImageGeneral(0).Visible = True
ImageGeneral(1).Visible = False
ImageScan(0).Visible = False
ImageScan(1).Visible = True
ImageShield(0).Visible = False
ImageShield(1).Visible = True
ImageUpdate(0).Visible = False
ImageUpdate(1).Visible = True
Dim i As Long
For i = 0 To 2

```

’ 初始化关闭按钮位置

```

With ImageExit(i)
.Left = 8760
.Top = 0
End With
Next

```

’ 关闭按钮

```

ImageExit(0).Visible = True
ImageExit(1).Visible = False
ImageExit(2).Visible = False

```

’ 提示窗体自动关闭频率

```

With ComboAutoCloseAlertInterval
.AddItem "3"
.AddItem "5"
.AddItem "10"
.AddItem "15"
End With

```

’ 自动升级频率

```

With ComboAutoUpdateInterval
.AddItem "1"
.AddItem "3"
.AddItem "5"
End With
Dim sSettingsFile As String

```

’ 软件设置记录文件

```

If Right(App.Path, 1) = "\" Then

```

```

sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\\Settings.ini"
End If

' 切换到扫描设置
ImageScan_Click (0)

' 读取自我保护设置
CheckSafeGuard.value = ReadIni(sSettingsFile, "Normal", "SafeGuard")

' 读取右键关联
CheckEnableRightClickMenu.value = ReadIni(sSettingsFile, "Normal", "RightClickMenu")

' 读取自启动设置
CheckAutoRun.value = ReadIni(sSettingsFile, "Normal", "AutoRun")

' 读取自动放入托盘图标设置
CheckAutoTray.value = ReadIni(sSettingsFile, "Normal", "AutoTray")

' 读取扫描时减少 CPU 占用设置
CheckLowCPU.value = ReadIni(sSettingsFile, "Scan", "CheckLowCPU")

' 读取扫描所有文件设置
OptionScanAllFiles.value = ReadIni(sSettingsFile, "Scan", "ScanAllFiles")

' 读取扫描可执行文件设置
OptionScanPeFiles.value = ReadIni(sSettingsFile, "Scan", "ScanPeFiles")

' 读取自动清除设置
OptionClear.value = ReadIni(sSettingsFile, "Scan", "ClearVirus")

' 读取询问设置
OptionAlert.value = ReadIni(sSettingsFile, "Scan", "AlertVirus")

' 读取忽略设置
OptionIgnore.value = ReadIni(sSettingsFile, "Scan", "IgnoreVirus")

' 读取隔离设置
OptionQuarantine.value = ReadIni(sSettingsFile, "Scan", "QuarantineVirus")

' 读取防护级别 - 正常 设置
OptionNormalLevel.value = ReadIni(sSettingsFile, "Shield", "NormalLevel")

' 读取防护级别 - 高 设置
OptionHighLevel.value = ReadIni(sSettingsFile, "Shield", "HighLevel")

' 读取云安全设置
CheckCloudSecurity.value = ReadIni(sSettingsFile, "Shield", "CloudSecurity")

' 读取防护提醒设置

```

```

CheckEnableAlert.value = ReadIni(sSettingsFile, "Shield", "EnableAlertMessage")

' 自动关闭防护提示消息
CheckAutoCloseAlert.value = ReadIni(sSettingsFile, "Shield", "AutoCloseAlertMessage")

' 自动关闭防护提示消息频率
ComboAutoCloseAlertInterval.Text = ReadIni(sSettingsFile, "Shield", "AutoCloseAlertMessageInterval")

' 自动更新是否开启的标志
CheckAutoUpdate.value = ReadIni(sSettingsFile, "Update", "AutoUpdate")

' 自动升级频率
ComboAutoUpdateInterval.Text = ReadIni(sSettingsFile, "Update", "AutoUpdateInterval")
End Sub

```

#### 鼠标在界面窗体中按下

```

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

' 按下鼠标左键
If Button = vbLeftButton Then

' 为当前的应用程序释放鼠标捕获
ReleaseCapture

' 移动窗体
SendMessage Me.hWnd, &HAI, 2, 0
End If
End Sub

```

#### 鼠标在界面窗体中移动

```

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

' 设置关闭按钮
ImageExit(0).Visible = True
ImageExit(1).Visible = False
ImageExit(2).Visible = False
End Sub

```

#### 窗体卸载

```

Private Sub Form_Unload(Cancel As Integer)
If bEnableUnloadForm = False Then
Cancel = 1
Me.Hide
End If
End Sub

```

#### 退出窗体

```
Private Sub ImageExit_Click(Index As Integer)
Me.Hide
End Sub
```

#### 鼠标在退出按钮上移动

```
Private Sub ImageExit_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

' 退出按钮状态
ImageExit(0).Visible = False
ImageExit(1).Visible = True
ImageExit(2).Visible = False
End Sub
```

#### 鼠标在退出按钮上按下

```
Private Sub ImageExit_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
' 退出按钮状态
ImageExit(0).Visible = False
ImageExit(1).Visible = False
ImageExit(2).Visible = True
End Sub
```

#### 鼠标在退出按钮上抬起

```
Private Sub ImageExit_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

' 退出点击按钮
If ImageExit(2).Visible = True Then
Me.Hide
End If
End Sub
```

#### 点击选项卡：一般设置

```
Private Sub ImageGeneral_Click(Index As Integer)

PictureGeneral.Visible = True
PictureScan.Visible = False
PictureShield.Visible = False
PictureUpdate.Visible = False
ImageGeneral(0).Visible = True
ImageGeneral(1).Visible = False
ImageScan(0).Visible = False
ImageScan(1).Visible = True
ImageShield(0).Visible = False
ImageShield(1).Visible = True
```

```
ImageUpdate(0).Visible = False
ImageUpdate(1).Visible = True
End Sub
```

#### \*点击选项卡: 扫描设置

```
Private Sub ImageScan_Click(Index As Integer)
PictureGeneral.Visible = False
PictureScan.Visible = True
PictureShield.Visible = False
PictureUpdate.Visible = False
ImageGeneral(0).Visible = False
ImageGeneral(1).Visible = True
ImageScan(0).Visible = True
ImageScan(1).Visible = False
ImageShield(0).Visible = False
ImageShield(1).Visible = True
ImageUpdate(0).Visible = False
ImageUpdate(1).Visible = True
End Sub
```

#### \*点击选项卡: 防护设置

```
Private Sub ImageShield_Click(Index As Integer)
PictureGeneral.Visible = False
PictureScan.Visible = False
PictureShield.Visible = True
PictureUpdate.Visible = False
ImageGeneral(0).Visible = False
ImageGeneral(1).Visible = True
ImageScan(0).Visible = False
ImageScan(1).Visible = True
ImageShield(0).Visible = True
ImageShield(1).Visible = False
ImageUpdate(0).Visible = False
ImageUpdate(1).Visible = True
End Sub
```

#### \*点击选项卡: 升级设置

```
Private Sub ImageUpdate_Click(Index As Integer)

PictureGeneral.Visible = False
PictureScan.Visible = False
PictureShield.Visible = False
PictureUpdate.Visible = True

ImageGeneral(0).Visible = False
ImageGeneral(1).Visible = True
ImageScan(0).Visible = False
```

```

ImageScan(1).Visible = True
ImageShield(0).Visible = False
ImageShield(1).Visible = True
ImageUpdate(0).Visible = True
ImageUpdate(1).Visible = False
End Sub

```

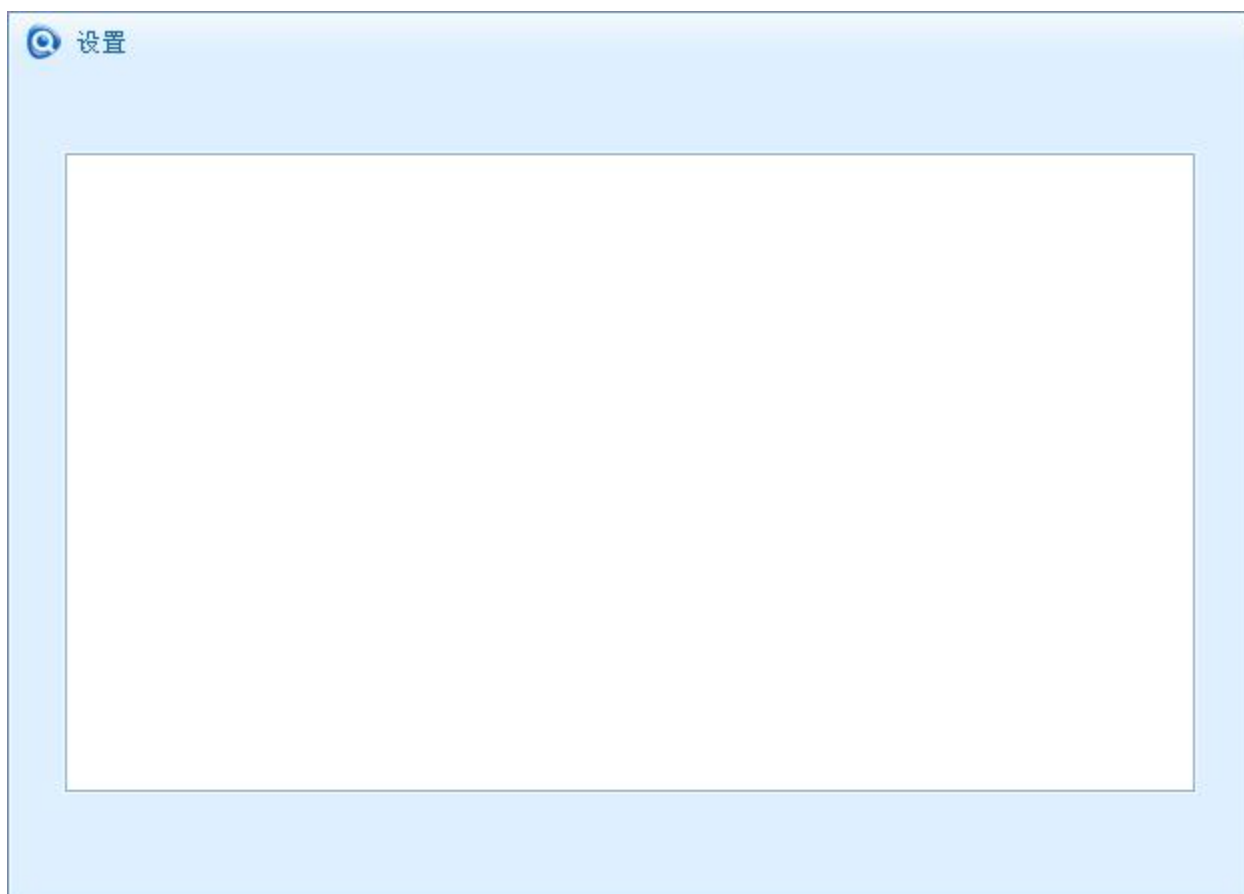
‘换肤

```

Public Function ReSkinMe()
With Me
.Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Setting2.bmp")

```

’ 上面是加载窗体的背景界面图片，图片如下：



’ 加载图片模拟的三态退出按钮图片

```

.ImageExit(0).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit0.bmp")
.ImageExit(1).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit1.bmp")
.ImageExit(2).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit2.bmp")

.ImageGeneral(0).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\GeneralSetting0.bmp")
.ImageGeneral(1).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\GeneralSetting1.bmp")

```

’ 上面是用图片模拟的选项卡控件中的一般设置选项卡，是双态效果，加载的两个图片分别是：





```
. ImageScan(0).Picture = LoadPicture(App.Path & "\\Skin\\" & sSkin & "\\ScanSetting0.bmp")
. ImageScan(1).Picture = LoadPicture(App.Path & "\\Skin\\" & sSkin & "\\ScanSetting1.bmp")
’ 上面是用图片模拟的选项卡控件中的扫描设置选项卡，是双态效果，加载的两个图片分别是：
```



```
. ImageShield(0).Picture = LoadPicture(App.Path & "\\Skin\\" & sSkin & "\\ShieldSetting0.bmp")
. ImageShield(1).Picture = LoadPicture(App.Path & "\\Skin\\" & sSkin & "\\ShieldSetting1.bmp")
’ 上面是用图片模拟的选项卡控件中的防护设置选项卡，是双态效果，加载的两个图片分别是：
```



```
. ImageUpdate(0).Picture = LoadPicture(App.Path & "\\Skin\\" & sSkin & "\\UpdateSetting0.bmp")
. ImageUpdate(1).Picture = LoadPicture(App.Path & "\\Skin\\" & sSkin & "\\UpdateSetting1.bmp")
’ 上面是用图片模拟的选项卡控件中的升级设置选项卡，是双态效果，加载的两个图片分别是：
```



```
End With
End Function
```

5.3.12. 软件换肤窗体

窗体：  
FormSkin

功能：  
实现换肤功能。



窗体中包含的控件及功能说明：

控件名称	类别	功能
ComboSkin	下拉框控件	放置皮肤列表的下拉框控件。

CommandOK	按钮控件	确定按钮。
ImageExit	图片控件	用图片模拟的关闭按钮。

```
代码:
Option Explicit
' api 声明
Private Declare Function ReleaseCapture Lib "user32" () As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long, ByVal lParam As Any) As Long
Private Declare Function FindFirstFile Lib "kernel32" Alias "FindFirstFileA" (ByVal lpFileName As String, lpFindFileData As WIN32_FIND_DATA) As Long
Private Declare Function FindNextFile Lib "kernel32" Alias "FindNextFileA" (ByVal hFindFile As Long, lpFindFileData As WIN32_FIND_DATA) As Long
Private Declare Function FindClose Lib "kernel32" (ByVal hFindFile As Long) As Long

' 自定义类型
Private Type FILETIME
    dwLowDateTime As Long
    dwHighDateTime As Long
End Type

' 查找文件使用的自定义类型
Private Type WIN32_FIND_DATA
    dwFileAttributes As Long
    ftCreationTime As FILETIME
    ftLastAccessTime As FILETIME
    ftLastWriteTime As FILETIME
    nFileSizeHigh As Long
    nFileSizeLow As Long
    dwReserved0 As Long
    dwReserved1 As Long
    cFileName As String * 1024
    cAlternate As String * 256
End Type

Private Const FILE_ATTRIBUTE_DIRECTORY = &H10
```

点击确定按钮

```
Private Sub CommandOK_Click()

' 改变皮肤
sSkin = ComboSkin.Text

' 为其它所有界面换肤
ReSkinAll
DoEvents
Dim sSettingsFile As String
```

```

' 软件设置记录文件
If Right(App.Path, 1) = "\" Then
sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\Settings.ini"
End If

' 将换肤后使用的皮肤信息写入配置文件
Call WriteIni(sSettingsFile, "Normal", "Skin", sSkin)
Unload Me
End Sub

```

#### 鼠标在窗体中按下

```

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

' 按下鼠标左键
If Button = vbLeftButton Then

' 为当前的应用程序释放鼠标捕获
ReleaseCapture

' 移动窗体
SendMessage Me.hWnd, &HA1, 2, 0
End If
End Sub

```

#### 窗体启动函数

```

Private Sub Form_Load()

' 设置皮肤
ReSkinMe
Dim j As Long
For j = 0 To 2

' 初始化关闭按钮位置
With ImageExit(j)
.Left = 4560
.Top = 0
End With
Next

' 初始化图片模拟的关闭按钮
ImageExit(0).Visible = True
ImageExit(1).Visible = False
ImageExit(2).Visible = False

' 初始化可换肤的皮肤列表
ComboSkin.Clear

```

```

Dim sSkinPath As String
sSkinPath = App.Path
If Right(sSkinPath, 1) <> "\" Then
sSkinPath = sSkinPath & "\"
End If
sSkinPath = sSkinPath & "Skin\"
InitSkins sSkinPath, "*.*)"

' 当前使用的皮肤
ComboSkin.Text = sSkin
End Sub

```

#### 鼠标在窗体中移动

```

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

' 设置关闭按钮
ImageExit(0).Visible = True
ImageExit(1).Visible = False
ImageExit(2).Visible = False
End Sub

```

#### 退出窗体

```

Private Sub ImageExit_Click(Index As Integer)
Unload Me
End Sub

```

#### 鼠标在退出按钮上移动

```

Private Sub ImageExit_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

' 退出按钮状态
ImageExit(0).Visible = False
ImageExit(1).Visible = True
ImageExit(2).Visible = False
End Sub

```

#### 鼠标在图片模拟的退出按钮上按下

```

Private Sub ImageExit_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

' 退出按钮状态
ImageExit(0).Visible = False
ImageExit(1).Visible = False
ImageExit(2).Visible = True
End Sub

```

#### 鼠标在图片模拟的退出按钮上抬起

```
Private Sub ImageExit_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
' 退出窗体
```

```
Unload Me
```

```
End Sub
```

### 初始化换肤功能

```
Public Sub InitSkins(DirPath As String, FileSpec As String)
```

```
' API 用自定义结构。
```

```
Dim FindData As WIN32_FIND_DATA
```

```
' 要搜索的目录
```

```
DirPath = Trim(DirPath)
```

```
' 构成完整目录形式
```

```
If Right(DirPath, 1) <> "\" Then
```

```
DirPath = DirPath & "\"
```

```
End If
```

```
' FindFirstfile 返回的句柄
```

```
Dim FindHandle As Long
```

```
' 在目标目录中取得第一个文件名
```

```
FindHandle = FindFirstFile(DirPath & FileSpec, FindData)
```

```
' 如果没有失败(说明有文件)
```

```
If FindHandle <> 0 Then
```

```
If FindData.dwFileAttributes And FILE_ATTRIBUTE_DIRECTORY Then
```

```
' 如果是一个目录
```

```
If Left(FindData.cFileName, 1) <> "." And Left(FindData.cFileName, 2) <> ".." Then
```

```
' 添加皮肤信息到界面下拉控件列中
```

```
ComboSkin.AddItem Trim(FindData.cFileName)
```

```
End If
```

```
End If
```

```
End If
```

```
' 现在开始找其它文件
```

```
If FindHandle <> 0 Then
```

```
Dim sFullName As String
```

```
' FindNextFile 返回的句柄
```

```
Dim FindNextHandle As Long
```

```
Do
```

```
DoEvents
```

```

' 找下一个文件
FindNextHandle = FindNextFile(FindHandle, FindData)
If FindNextHandle <> 0 Then
If FindData.dwFileAttributes And FILE_ATTRIBUTE_DIRECTORY Then

' 是目录的话,就加到目录列表
If Left(FindData.cFileName, 1) <> "." And Left(FindData.cFileName, 2) <> ".." Then
ComboSkin.AddItem Trim(FindData.cFileName)
End If
End If
Else
Exit Do
End If
Loop
End If

' 关闭文件查找句柄
Call FindClose(FindHandle)
End Sub

```

#### 本窗体的换肤函数

```

Public Function ReSkinMe()
With Me
.Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Skin.bmp")

```

' 以上是加载窗体的背景图片, 图片为:



```

' 加载图片模拟的三态退出按钮图片
.ImageExit(0).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit0.bmp")
.ImageExit(1).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit1.bmp")
.ImageExit(2).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit2.bmp")
End With
End Function

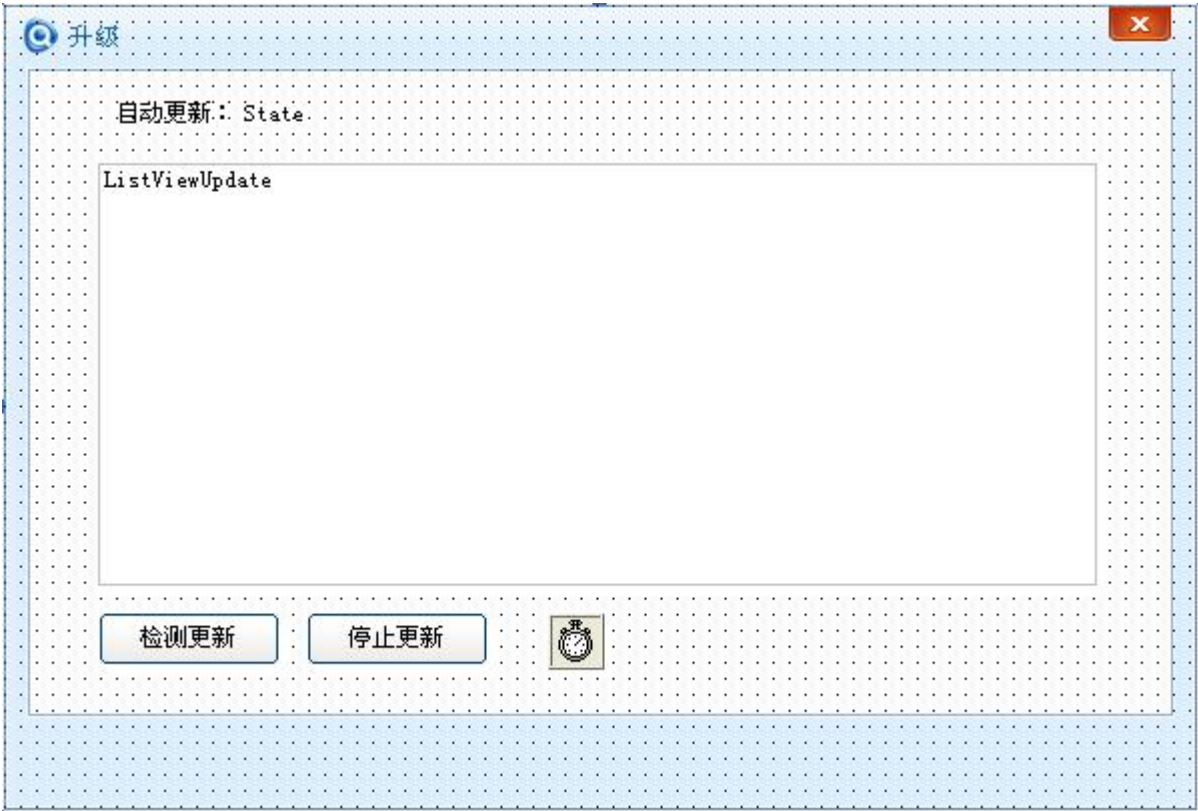
```

#### 5.3.13. 软件及病毒库升级窗体

窗体：  
FormUpdate

功能：  
实现升级功能。

界面设计：



窗体中包含的控件及功能说明：

控件名称	类别	功能
LabelAutoUpdate State	文本框控件	用于显示是否开启了自动升级。
ListViewUpdate	网格控件	用于放置更级文件列表的网格控件。
CommandUpdate	按钮控件	更新按钮。
TimerAutoUpdate	定时器控件	用于实现自动升级的定时器控件。
ImageExit	图片控件	用图片控件模拟的关闭按钮。

代码：

```
Option Explicit
' api 声明

' 函数功能：初始化一个应用程序，以使用 WinINet 函数。
Private Declare Function InternetOpen Lib "WinInet.dll" Alias "InternetOpenA" (ByVal sAgent As String,
ByVal lAccessType As Long, ByVal sProxyName As String, ByVal sProxyBypass As String, ByVal lFlags As Long)
```

As Long

’ 函数功能：通过一个完整的 FTP，Gopher 或 HTTP 网址打开一个资源。

```
Private Declare Function InternetOpenUrl Lib "WinInet.dll" Alias "InternetOpenUrlA" (ByVal hInternetSession As Long, ByVal sUrl As String, ByVal sHeaders As String, ByVal lHeadersLength As Long, ByVal lFlags As Long, ByVal lContext As Long) As Long
```

’ 函数功能：读取网络文件内容

```
Private Declare Function InternetReadFile Lib "WinInet.dll" (ByVal hFile As Long, ByVal sBuffer As String, ByVal lNumBytesToRead As Long, lNumberOfBytesRead As Long) As Integer
```

’ 函数功能：关闭刚才连接句柄

```
Private Declare Function InternetCloseHandle Lib "WinInet.dll" (ByVal hInet As Long) As Integer  
Private Declare Function ReleaseCapture Lib "user32" () As Long  
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long, ByVal wMsg As Long, ByVal lParam As Long, lParam As Any) As Long
```

’ 函数功能：移动正在执行文件的文件保护，使正在执行的文件可以被替换、移动或删除

```
Private Declare Function MoveFileEx Lib "kernel32" Alias "MoveFileExA" (ByVal lpExistingFileName As String, ByVal lpNewFileName As String, ByVal dwFlags As Long) As Long  
Private Declare Function WinExec Lib "kernel32" (ByVal lpCmdLine As String, ByVal nCmdShow As Long) As Long
```

’ 常量定义

```
Private Const INTERNET_FLAG_NO_CACHE_WRITE = &H4000000
```

’ 是否可以下载下一个文件

```
Dim bEnableDownloadNext As Boolean
```

’ 是否中止下载

```
Dim bStopDownload As Boolean
```

’ 当前下载的文件大小

```
Dim lCurrentFileSize
```

’ 自动升级计数

```
Dim lCurrentSecond As Long
```

’ 获取指定网址返回内容

```
Public Function DetectUpdateFile(sUrl As String) As String  
Dim lInternetOpenUrl As Long  
Dim lInternetOpen As Long  
Dim sTemp As String * 1024  
Dim lInternetReadFile As Long  
Dim lSize As Long  
Dim sContent As String  
sContent = vbNullString
```

’ 初始化，以使用 WinINet 函数

```
lInternetOpen = InternetOpen("WangLiwen", 1, vbNullString, vbNullString, 0)
```



```

If lInternetOpen Then

' 打开网址
lInternetOpenUrl = InternetOpenUrl(lInternetOpen, sUrl, vbNullString, 0, INTERNET_FLAG_NO_CACHE_WRITE,
0)
If lInternetOpenUrl Then
Do

' 读取网页内容
lInternetReadFile = InternetReadFile(lInternetOpenUrl, sTemp, 1024, lSize)
sContent = sContent & Mid(sTemp, 1, lSize)
Loop While (lSize <> 0)
Else

' 显示连接错误
If Me.Visible = True Then
MsgBox "错误代码：001。连接升级服务器失败，请检查网络连接状况。"
Else

' 如果窗体可见，则弹出对话框，如果窗体被隐藏则将提示信息显示在窗体中
LabelAutoUpdateMsg.Caption = "错误代码：001。连接升级服务器失败，请检查网络连接状况。"
End If
DetectUpdateFile = ""
Exit Function
End If
lInternetReadFile = InternetCloseHandle(lInternetOpenUrl)
Else

' 显示连接错误
If Me.Visible = True Then
MsgBox "错误代码：002。连接升级服务器失败，请检查网络连接状况。"
Else

' 如果窗体可见，则弹出对话框，如果窗体被隐藏则将提示信息显示在窗体中
LabelAutoUpdateMsg.Caption = "错误代码：002。连接升级服务器失败，请检查网络连接状况。"
End If
DetectUpdateFile = ""
Exit Function
End If

' 升级查询返回以“Update”开始为标识
If UCase(Left(sContent, 6)) = UCase("Update") Then
DetectUpdateFile = Right(sContent, Len(sContent) - 6)
Else
DetectUpdateFile = ""
End If
End Function

```

**检测升级更新内容**

```

Private Sub CommandUpdate_Click()

' 是否中止下载标志
bStopDownload = False

' 从指定网址获取更新文件内容
Dim sUpdateFiles As String
sUpdateFiles = DetectUpdateFile("http://www.haiqi.cn/update/update.txt")
If sUpdateFiles = "" Then

' 如果窗体可见，则弹出对话框，如果窗体被隐藏则将提示信息显示在窗体中
If Me.Visible = True Then
MsgBox "升级服务器错误，无法获取升级文件。"
Else
LabelAutoUpdateMsg.Caption = "升级服务器错误，无法获取升级文件。"
End If
Exit Sub
End If

' 取得新整体版本号
Dim lNewWholeVersion As Long
lNewWholeVersion = Left(sUpdateFiles, 6)
sUpdateFiles = Right(sUpdateFiles, Len(sUpdateFiles) - 6)

' 软件版本记录文件
Dim sVersionFile As String
If Right(App.Path, 1) = "\" Then
sVersionFile = App.Path & "Version.ini"
Else
sVersionFile = App.Path & "\Version.ini"
End If

' 读取旧整体版本
Dim lOldWholeVersion As Long
lOldWholeVersion = ReadIni(sVersionFile, "Version", "WholeVersion")

' 判断是否有升级内容
If lNewWholeVersion <= lOldWholeVersion Then

' 如果窗体可见，则弹出对话框，如果窗体被隐藏则将提示信息显示在窗体中
If Me.Visible = True Then
MsgBox "已是最新版本，无需升级。"
Else
LabelAutoUpdateMsg.Caption = "已是最新版本，无需升级。"
End If
Exit Sub
End If

' 清空旧升级内容
ListViewUpdate.ListItems.Clear
CommandUpdate.Enabled = False

```

```

CommandStop.Enabled = True

' 如果有持更新内容
Do While Not sUpdateFiles = ""
DoEvents

' 从返回内容中取得持更新文件名
With ListViewUpdate
.ListItems.Add , , .ListItems.Count + 1
.ListItems(.ListItems.Count).SubItems(1) = Left(sUpdateFiles, InStr(1, sUpdateFiles, "$") - 1)
End With
sUpdateFiles = Right(sUpdateFiles, Len(sUpdateFiles) - InStr(1, sUpdateFiles, "$"))
Loop

' 升级更新每个文件
Dim i As Long
For i = 1 To ListViewUpdate.ListItems.Count
DoEvents

' 是否中止下载
If bStopDownload = True Then
Exit For
End If

' 如果是要下载 exe 文件, 先用 movefileex 去掉它的执行保护
If InStr(1, LCase(ListViewUpdate.ListItems(i).SubItems(1)), LCase(".exe")) Then
Call MoveFileEx(ListViewUpdate.ListItems(i).SubItems(1), RndChr(6), 1)
End If

' 是否可以更新下一个文件, 下载依次进行, 正在下载文件时不能进行此操作
If bEnableDownloadNext = True Then
With UserControlUpdate1

' 文件下载地址
.URL = "http://www.haiqi.cn/update/" & ListViewUpdate.ListItems(i).SubItems(1)

' 本地保存地址
.SaveFile = App.Path & "\" & ListViewUpdate.ListItems(i).SubItems(1)
If .Init = True Then

' 当前下载的文件大小, 单位: byte
lCurrentFileSize = .GetHeader("Content-Length")

' 设置正在下载标志
bEnableDownloadNext = False

' 开始下载文件
.StartUpdate
End If
End With
End If

```

```
Next  
End Sub
```

#### ‘停止下载’

```
Private Sub CommandStop_Click()  
bStopDownload = True  
UserControlUpdate1.CancelDownload  
CommandUpdate.Enabled = True  
CommandStop.Enabled = False  
End Sub
```

#### 鼠标在窗体上移动

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)  
  
' 关闭按钮  
ImageExit(0).Visible = True  
ImageExit(1).Visible = False  
ImageExit(2).Visible = False  
End Sub
```

#### 窗体卸载

```
Private Sub Form_Unload(Cancel As Integer)  
If bEnableUnloadForm = False Then  
Cancel = 1  
Me.Hide  
End If  
End Sub
```

#### 退出窗体

```
Private Sub ImageExit_Click(Index As Integer)  
Me.Hide  
End Sub
```

#### 鼠标在图片模拟的退出按钮上移动

```
Private Sub ImageExit_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y  
As Single)  
  
' 退出按钮状态  
ImageExit(0).Visible = False  
ImageExit(1).Visible = True  
ImageExit(2).Visible = False  
End Sub
```

#### 鼠标在图片模拟的退出按钮上按下

```
Private Sub ImageExit_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    ' 退出按钮状态
```

```
    ImageExit(0).Visible = False
```

```
    ImageExit(1).Visible = False
```

```
    ImageExit(2).Visible = True
```

```
End Sub
```

#### 鼠标在图片模拟的退出按钮上抬起

```
Private Sub ImageExit_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    ' 退出点击按钮
```

```
    If ImageExit(2).Visible = True Then
```

```
        Me.Hide
```

```
    End If
```

```
End Sub
```

#### 窗体启动函数

```
Private Sub Form_Load()
```

```
    ' 加载本窗体皮肤
```

```
    ReSkinMe
```

```
    Dim i As Long
```

```
    For i = 0 To 2
```

```
        ' 初始化关闭按钮位置
```

```
        With ImageExit(i)
```

```
            .Left = 8280
```

```
            .Top = 0
```

```
        End With
```

```
    Next
```

```
    ' 自动升级计数
```

```
    lCurrentSecond = 0
```

```
    ' 关闭按钮
```

```
    ImageExit(0).Visible = True
```

```
    ImageExit(1).Visible = False
```

```
    ImageExit(2).Visible = False
```

```
    ' 设置正在下载标志
```

```
    bEnableDownloadNext = True
```

```
    ' 软件设置记录文件
```

```
    Dim sSettingsFile As String
```

```
    If Right(App.Path, 1) = "\" Then
```

```

sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\Settings.ini"
End If

' 读取升级设置
Dim lAutoUpdate As Long
lAutoUpdate = ReadIni(sSettingsFile, "Update", "AutoUpdate")

' 自动更新设置
If lAutoUpdate = 1 Then
LabelAutoUpdateState.Caption = "已启用"
TimerAutoUpdate.Enabled = True
Else
LabelAutoUpdateState.Caption = "未启用"
TimerAutoUpdate.Enabled = False
End If
CommandUpdate.Enabled = True
CommandStop.Enabled = False
End Sub

```

#### 鼠标在窗体中按下

```

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

' 按下鼠标左键
If Button = vbLeftButton Then

' 为当前的应用程序释放鼠标捕获
ReleaseCapture

' 移动窗体
SendMessage Me.hWnd, &HA1, 2, 0
End If
End Sub

```

#### 自动升级

```

Private Sub TimerAutoUpdate_Timer()
Dim sSettingsFile As String

' 软件设置记录文件是软件目录下的 Settings.ini 文件
If Right(App.Path, 1) = "\" Then
sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\Settings.ini"
End If

获取自动升级频率
Dim lAutoUpdateIntervel As Long

```

```
lAutoUpdateInterval = ReadIni(sSettingsFile, "Update", "AutoUpdateInterval")
```

```
' 将升级频率变换为秒
```

```
Dim lMaxSecond As Long
```

```
lMaxSecond = lAutoUpdateInterval * 3600
```

```
' 自动升级计数
```

```
lCurrentSecond = lCurrentSecond + 1
```

```
' 判断是否到达自动升级时间
```

```
If lCurrentSecond >= lMaxSecond Then
```

```
' 升级计数清零
```

```
lCurrentSecond = 0
```

```
' 升级下载文件
```

```
CommandUpdate_Click
```

```
End If
```

```
End Sub
```

```
' 升级控件事件：一个下载完成
```

```
Private Sub UserControlUpdate1_DownloadFinish()
```

```
bEnableDownloadNext = True
```

```
Dim i As Long
```

```
For i = 1 To ListViewUpdate.ListItems.Count
```

```
If ListViewUpdate.ListItems(i).SubItems(2) = "100%" Then
```

```
' 判断是否是 exe 文件
```

```
If InStr(1, LCase(ListViewUpdate.ListItems(i).SubItems(1)), LCase(".exe")) Then
```

```
DoEvents
```

```
Dim sRunFile As String
```

```
sRunFile = ListViewUpdate.ListItems(i).SubItems(1)
```

```
' 如果是 exe 文件，则在下载后自动执行
```

```
Call WinExec(sRunFile, 1)
```

```
DoEvents
```

```
End If
```

```
' 如果正下载完成的文件数量与要下载的文件数量相等，说明全部文件都已经下载完成
```

```
If ListViewUpdate.ListItems.Count = i Then
```

```
CommandUpdate.Enabled = True
```

```
CommandStop.Enabled = False
```

```
' 如果窗体可见，弹出消息框提示
```

```
If Me.Visible = True Then
```

```
MsgBox "升级完成。"
```

```
Else
```

```
' 如果窗体不可见（多数是主程序最小化在系统托盘区），则将下载完成的消息显示在窗体中
```

```

LabelAutoUpdateMsg.Caption = "升级完成。"

' 提醒窗体
Dim uAlertForm As New FormAlertMessage

' 弹出窗体提示用户
With uAlertForm

' 设置窗体位置
.Top = Screen.Height - .Height - 300
.Left = Screen.Width - .Width

' 升级完成提示
Dim sUpdated As String
sUpdated = "自动升级完成, 已升级文件: " & vbCrLf
Dim j As Long
For j = 1 To ListViewUpdate.ListItems.Count
sUpdated = sUpdated & ListViewUpdate.ListItems(j).SubItems(1) & " "
Next j

' 显示内容
.LabelInfo.Caption = sUpdated

' 显示窗体
.Show vbModal
End With
End If

' 刷新软件版本、病毒库版本、Web 控件
FormScan.TimerUpdateRefresh.Enabled = True
End If
End If
Next
End Sub

```

#### 升级控件事件：实时下载进展

```

Private Sub UserControlUpdate1_DownloadProcess(lProgress As Long)
bEnableDownloadNext = True

' 遍历所有待升级文件
Dim i As Long
For i = 1 To ListViewUpdate.ListItems.Count

' 如果下载进度达到 100%表示下载已经完成, 就不再更新下载进度
If ListViewUpdate.ListItems(i).SubItems(2) <> "100%" Then

' 显示正在下载的文件当前下载进度
ListViewUpdate.ListItems(i).SubItems(2) = ((lProgress * 100) \ lCurrentFileSize) & "%"
Exit For

```



```
End If
Next
End Sub
```

#### 升级控件事件：错误消息

```
Private Sub UserControlUpdate1_ErrorMessage(sDescription As String)

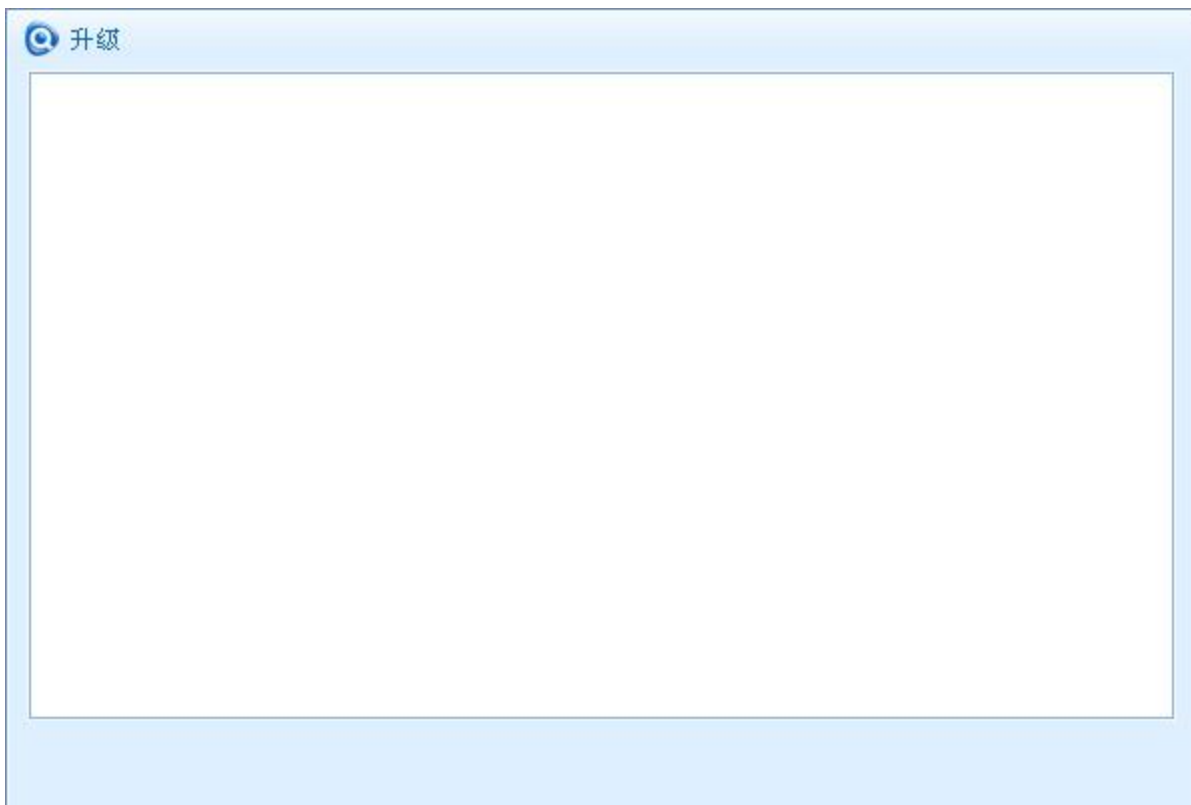
' 如果窗体可见，则弹出对话框
If Me.Visible = True Then
MsgBox sDescription
Else

' 如果窗体不可见则显示在窗体的文字控件中
LabelAutoUpdateMsg.Caption = sDescription
End If
End Sub
```

#### 本窗体的换肤函数

```
Public Function ReSkinMe()
With Me
.Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Update2.bmp")

' 以上是加载本窗体的背景界面图片，图片为：
```



```

' 加载图片模拟的三态退出按钮图片
.ImageExit(0).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit0.bmp")
.ImageExit(1).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit1.bmp")
.ImageExit(2).Picture = LoadPicture(App.Path & "\Skin\" & sSkin & "\Exit2.bmp")
End With
End Function

```

## 5.4. 模块文件说明及代码剖析

### 5.4.1. 主动防御模块

模块:

ModuleActiveDefense

功能:

主动防护函数实现模块。

代码:

Option Explicit

' API 声明

' 以下这两个 API 函数是我们用 VC 写成的，用于实现主动防御，相关的代码会在后文 DLL 模块代码中展示。

```

Private Declare Function ActiveDefenseON Lib "ActiveDefense.dll" (ByVal TargetHwnd As Long) As Boolean
Private Declare Function ActiveDefenseOFF Lib "ActiveDefense.dll" () As Boolean

```

' 函数说明：获取当前进程一个唯一的标识符

```

Private Declare Function GetCurrentProcessId Lib "kernel32" () As Long
Private Declare Function InternetOpen Lib "WinInet.dll" Alias "InternetOpenA" (ByVal sAgent As String,
ByVal lAccessType As Long, ByVal sProxyName As String, ByVal sProxyBypass As String, ByVal lFlags As Long)
As Long
Private Declare Function InternetOpenUrl Lib "WinInet.dll" Alias "InternetOpenUrlA" (ByVal
hInternetSession As Long, ByVal sUrl As String, ByVal sHeaders As String, ByVal lHeadersLength As Long,
ByVal lFlags As Long, ByVal lContext As Long) As Long
Private Declare Function InternetReadFile Lib "WinInet.dll" (ByVal hFile As Long, ByVal sBuffer As String,
ByVal lNumBytesToRead As Long, lNumberOfBytesRead As Long) As Integer
Private Declare Function InternetCloseHandle Lib "WinInet.dll" (ByVal hInet As Long) As Integer
Private Declare Function SetForegroundWindow Lib "user32" (ByVal hWnd As Long) As Long

```

```

Private Const INTERNET_FLAG_NO_CACHE_WRITE = &H4000000

```

#### \* 开启或关闭主动防御

' 参数：Boolean 型，传入 True 时启动主动防御，传入 False 时关闭主动防御

' 返回值：True 表示成功，False 表示失败

```

Public Function ActiveDefense(ByVal bOperation As Boolean) As Boolean

```

```

Dim bRet As Boolean
If bOperation = True Then

' 开启主动防御
bRet = ActiveDefenseON()

' 函数返回结果
' 函数所在 Dll 是用 VC 编写
' VC 中 return true 返回的 true 值，跟 VB 中 true 值不同。故以下代码不能使用 bret=true 进行比较。
' VC 中 True=1
' VB 中 true=0ffffff
If bRet Then
' 主动防御开启成功
ActiveDefense = True
Else

' 主动防御开启失败
ActiveDefense = False
End If
Else

' 关闭主动防御
ActiveDefenseOFF
End If
End Function

```

#### 向服务器上报病毒拦截信息

```

' 参数：URL 地址，格式：url+病毒名
' 返回值：空
Public Function ResponseVirusInfo(sUrl As String) As String
Dim lInternetOpenUrl As Long
Dim lInternetOpen As Long
Dim sTemp As String * 1024
Dim lInternetReadFile As Long
Dim lSize As Long
Dim sContent As String
sContent = vbNullString

' 初始化，以使用 WinINet 函数
lInternetOpen = InternetOpen("WangLiwen", 1, vbNullString, vbNullString, 0)
If lInternetOpen Then

' 打开网址
lInternetOpenUrl = InternetOpenUrl(lInternetOpen, sUrl, vbNullString, 0, INTERNET_FLAG_NO_CACHE_WRITE, 0)
If lInternetOpenUrl Then
Do
lInternetReadFile = InternetReadFile(lInternetOpenUrl, sTemp, 1024, lSize)

```

```

sContent = sContent & Mid(sTemp, 1, lSize)
Loop While (lSize <> 0)
Else
Exit Function
End If
lInternetReadFile = InternetCloseHandle(lInternetOpenUrl)
Else
Exit Function
End If
End Function

```

#### 5.4.2. 软件开机自启动实现模块

模块:

ModuleAutorun

功能:

程序自启动实现模块。

代码:

```
Option Explicit
```

' api 声明

' 函数说明: 在指定的项下创建一个新项。如指定的项已经存在, 那么函数会打开现有的项

```
Private Declare Function RegCreateKey Lib "advapi32.dll" Alias "RegCreateKeyA" (ByVal hKey As Long, ByVal lpSubKey As String, ByRef phkResult As Long) As Long
```

' 创建或改变一个键值, 有时 lpData 应由缺省的 ByRef 型改为 ByVal 型

```
Private Declare Function RegSetValueExString Lib "advapi32.dll" Alias "RegSetValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal Reserved As Long, ByVal dwType As Long, ByVal lpData As Any, ByVal cbData As Long) As Long
```

' 函数功能: 释放指定注册键的句柄

```
Private Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hKey As Long) As Long
```

' 函数功能: 删除指定项下方的一个值

```
Private Declare Function RegDeleteValue Lib "advapi32.dll" Alias "RegDeleteValueA" (ByVal hKey As Long, ByVal lpValueName As String) As Long
```

' 常量定义

```
Private Const REG_SZ = 1
```

```
Private Const HKEY_LOCAL_MACHINE = &H80000002
```

#### 设置开机启动

```

Public Function SetAutoRun()
On Error Resume Next
Dim sAddAutoRunReg As String

```

```

' 要写入注册表的启动项名称
sAddAutoRunReg = "HQ"
Dim nKeyHandle As Long

' 打开注册表启动项
' 注册表中 HKEY_LOCAL_MACHINE 主键的 Software\Microsoft\Windows\CurrentVersion\Run 分枝下项目对应的文件，会在 Windows 系统启动后会被执行，想让程序随系统自动启动，可以在这里进行操作
Call RegCreateKey(HKEY_LOCAL_MACHINE, "Software\Microsoft\Windows\CurrentVersion\Run", nKeyHandle)

' 向启动项中添加一个新的启动记录
If Right(App.Path, 1) = "\" Then
Call RegSetValueExString(nKeyHandle, sAddAutoRunReg, 0, REG_SZ, App.Path & App.EXENAME & ".exe", 255)
Else
Call RegSetValueExString(nKeyHandle, sAddAutoRunReg, 0, REG_SZ, App.Path & "\" & App.EXENAME & ".exe", 255)
End If

' 关闭上面打开的注册表项
Call RegCloseKey(nKeyHandle)
End Function

```

#### 取消开机启动

```

Public Function StopAutoRun()
On Error Resume Next
Dim ret As Long, hKey As Long

' 打开注册表项
Call RegCreateKey(HKEY_LOCAL_MACHINE, "Software\Microsoft\Windows\CurrentVersion\Run", hKey)

' 删除指定名称的自启动项
Call RegDeleteValue(hKey, "HQ")

' 关闭上面打开的注册表项
Call RegCloseKey(hKey)
End Function

```

### 5.4.3. 云安全防护实现模块

模块：

ModuleCloudShield

功能：

云安全防护实现模块。

代码：

```

Option Explicit
Private Declare Function ReleaseCapture Lib "user32" () As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam

```

```

As Long, ByVal wParam As Long, lParam As Any) As Long
Private Declare Function InternetOpen Lib "WinInet.dll" Alias "InternetOpenA" (ByVal sAgent As String,
ByVal lAccessType As Long, ByVal sProxyName As String, ByVal sProxyBypass As String, ByVal lFlags As Long)
As Long
Private Declare Function InternetOpenUrl Lib "WinInet.dll" Alias "InternetOpenUrlA" (ByVal
hInternetSession As Long, ByVal sUrl As String, ByVal sHeaders As String, ByVal lHeadersLength As Long,
ByVal lFlags As Long, ByVal lContext As Long) As Long
Private Declare Function InternetReadFile Lib "WinInet.dll" (ByVal hFile As Long, ByVal sBuffer As String,
ByVal lNumBytesToRead As Long, lNumberOfBytesRead As Long) As Integer
Private Declare Function InternetCloseHandle Lib "WinInet.dll" (ByVal hInet As Long) As Integer
Private Declare Function SetForegroundWindow Lib "user32" (ByVal hWnd As Long) As Long

Private Const INTERNET_FLAG_NO_CACHE_WRITE = &H4000000

```

· 获取指定网址返回值内容，用于进行云安全查询

```

' 参数：URL 地址
' 返回值：返回网页内容，可为空
Public Function CloudMatch(sUrl As String) As String
Dim lInternetOpenUrl As Long
Dim lInternetOpen As Long
Dim sTemp As String * 1024
Dim lInternetReadFile As Long
Dim lSize As Long
Dim sContent As String
sContent = vbNullString

' 初始化，以使用 WinINet 函数
lInternetOpen = InternetOpen("WangLiwen", 1, vbNullString, vbNullString, 0)
If lInternetOpen Then

' 打开网址
lInternetOpenUrl = InternetOpenUrl(lInternetOpen, sUrl, vbNullString, 0, INTERNET_FLAG_NO_CACHE_WRITE,
0)
If lInternetOpenUrl Then
Do

' 读取网页内容
lInternetReadFile = InternetReadFile(lInternetOpenUrl, sTemp, 1024, lSize)
sContent = sContent & Mid(sTemp, 1, lSize)
Loop While (lSize <> 0)
Else
CloudMatch = ""
Exit Function
End If
lInternetReadFile = InternetCloseHandle(lInternetOpenUrl)
Else
CloudMatch = ""
Exit Function

```

```
End If
```

```
' 云安全防御系统查询返回: Cloud$(云安全标识)+1/0(病毒 or 可信文件标识)+病毒名称/可信软件名称
If UCase(Left(sContent, 6)) = UCase("Cloud$") Then
CloudMatch = Right(sContent, Len(sContent) - 6)
Else
CloudMatch = ""
End If
End Function
```

#### 5.4.4. 可执行文件图标获取模块

模块:

```
ModuleGetFileIcon
```

功能:

获取文件图标。

代码:

```
Option Explicit
```

```
' API 声明
```

```
' 函数功能:创建一个新的图片对象初始化根据到 PICTDESC 结构。
```

```
Private Declare Function OleCreatePictureIndirect Lib "oleaut32.dll" (pPicDesc As TypeIcon, riid As CLSID,
ByVal fown As Long, lpUnk As Object) As Long
```

```
' 函数功能: 获取文件信息
```

```
Private Declare Function SHGetFileInfo Lib "shell32.dll" Alias "SHGetFileInfoA" (ByVal pszPath As String,
ByVal dwFileAttributes As Long, psfi As SHFILEINFO, ByVal cbFileInfo As Long, ByVal uFlags As Long) As
Long
```

```
' 常量定义
```

```
Private Const SHGFI_ICON = &H100
Private Const SHGFI_LARGEICON = &H0
Private Const SHGFI_hSmallIcon = &H1
```

```
' 自定义类型
```

```
Private Type TypeIcon
cbSize As Long
picType As PictureTypeConstants
hIcon As Long
End Type
```

```
Private Type CLSID
id(16) As Byte
End Type
```

```
Private Type SHFILEINFO
hIcon As Long
```

```

iIcon As Long
dwAttributes As Long
szDisplayName As String * 255
szTypeName As String * 80
End Type

```

#### ICON 转化为 Picture

```

Public Function IconToPicture(hIcon As Long) As IPictureDisp
Dim cls_id As CLSID
Dim hRes As Long
Dim new_icon As TypeIcon
Dim lpUnk As IUnknown
With new_icon
.cbSize = Len(new_icon)
.picType = vbPicTypeIcon
.hIcon = hIcon
End With
With cls_id
.id(8) = &HCO
.id(15) = &H46
End With
hRes = OleCreatePictureIndirect(new_icon, cls_id, 1, lpUnk)
If hRes = 0 Then
Set IconToPicture = lpUnk
End If
End Function

```

#### 获得文件 ICON

```

Public Function GetFileIcon(sFilename, ByVal bSmallIcon As Boolean) As IPictureDisp
Dim hIcon As Long
Dim item_num As Long
Dim icon_pic As IPictureDisp
Dim sh_info As SHFILEINFO
If bSmallIcon = True Then
SHGetFileInfo sFilename, 0, sh_info, Len(sh_info), SHGFI_ICON + SHGFI_bSmallIcon
Else
SHGetFileInfo sFilename, 0, sh_info, Len(sh_info), SHGFI_ICON + SHGFI_LARGEICON
End If
hIcon = sh_info.hIcon
Set icon_pic = IconToPicture(hIcon)
Set GetFileIcon = icon_pic
End Function

```

#### 5.4.5. 获取文件或文件节的哈希值模块

模块:



ModuleHashFileStream

功能:

获取文件或文件节的哈希值。

代码:

Option Explicit

' API 声明

```
Private Declare Function CryptAcquireContext Lib "advapi32.dll" Alias "CryptAcquireContextA" (ByRef  
phProv As Long, ByVal pszContainer As String, ByVal pszProvider As String, ByVal dwProvType As Long, ByVal  
dwFlags As Long) As Long
```

```
Private Declare Function CryptReleaseContext Lib "advapi32.dll" (ByVal hProv As Long, ByVal dwFlags As  
Long) As Long
```

```
Private Declare Function CryptCreateHash Lib "advapi32.dll" (ByVal hProv As Long, ByVal AlgId As Long,  
ByVal hKey As Long, ByVal dwFlags As Long, ByRef phHash As Long) As Long
```

```
Private Declare Function CryptDestroyHash Lib "advapi32.dll" (ByVal lHash As Long) As Long
```

```
Private Declare Function CryptHashData Lib "advapi32.dll" (ByVal lHash As Long, pbData As Any, ByVal  
dwDataLen As Long, ByVal dwFlags As Long) As Long
```

```
Private Declare Function CryptGetHashParam Lib "advapi32.dll" (ByVal lHash As Long, ByVal dwParam As Long,  
pbData As Any, pdwDataLen As Long, ByVal dwFlags As Long) As Long
```

```
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (Destination As Any, Source As Any,  
ByVal Length As Long)
```

' 常量定义

```
Private Const PROV_RSA_FULL = 1
```

```
Private Const CRYPT_NEWKEYSET = &H8
```

```
Private Const ALG_CLASS_HASH = 32768
```

```
Private Const ALG_TYPE_ANY = 0
```

```
Private Const ALG_SID_MD5 = 3
```

```
Private Const ALGORITHM = ALG_CLASS_HASH Or ALG_TYPE_ANY Or ALG_SID_MD5
```

```
Private Const HP_HASHVAL = 2
```

```
Private Const HP_HASHSIZE = 4
```

**获得文件流 hash, 文件中部分内容的 hash, 这里是取 section**

' 参数: ByteStream 流(Byte 数组)

' 成功返回流的 hash, 失败返回空

```
Public Function HashFileStream(ByteStream() As Byte) As String
```

' 初始化函数返回值

```
HashFileStream = ""
```

```
Dim lCtx As Long
```

```
Dim lHash As Long
```

```
Dim lFile As Long
```

```
Dim lRes As Long
```

```
Dim lLen As Long
```

```
Dim lIdx As Long
```

```
Dim bHash() As Byte
```

```
Dim bBlock() As Byte
```

```
Dim bStream() As Byte
```

```
ReDim bStream(1 To UBound(ByteStream)) As Byte
```

```
CopyMemory bStream(1), ByteStream(1), UBound(ByteStream)
```

```

' API 功能: 得到 CS, 即密码容器
' 参数说明:
' 参数 1 lCtx : 返回 CSP 句柄
' 参数 2 : 密码容器名, 为空连接缺省的 CSP
' 参数 3 : 为空时使用默认 CSP 名 (微软 RSA Base Provider)
' 参数 4 PROV_RSA_FULLCSP : 类型
lRes = CryptAcquireContext(lCtx, vbNullString, vbNullString, PROV_RSA_FULL, 0)
If lRes <> 0 Then

' API 功能: 创 hash 对象
' 参数说明:
' 参数 1: CSP 句柄
' 参数 2: 选择 hash 算法, 比如 CALG_MD5 等
' 参数 3: HMAC 和 MAC 算法时有用
' 参数 4: 保留, 传入 0 即可
' 参数 5: 返回 hash 句柄
lRes = CryptCreateHash(lCtx, ALGORITHM, 0, 0, lHash)
If lRes <> 0 Then

' 32K 数字后加&的意思: 系统默认数字是 Integer 型, 加了&就强制把这个数转换成 Long 型
Const BLOCK_SIZE As Long = 32 * 1024&
ReDim bBlock(1 To BLOCK_SIZE) As Byte
Dim lCount As Long
Dim lBlocks As Long
Dim lLastBlock As Long
' 块个数
lBlocks = UBound(ByteStream) \ BLOCK_SIZE

' 最后一个块
lLastBlock = UBound(ByteStream) - lBlocks * BLOCK_SIZE
For lCount = 0 To lBlocks - 1
CopyMemory bBlock(1), bStream(1 + lCount * BLOCK_SIZE), BLOCK_SIZE

' API 功能: hash 数据
' 参数说明:
' 参数 1: hash 对象
' 参数 2: 被 hash 的数据
' 参数 3: 数据的长度
' 参数 4: 微软的 CSP 这个值会被忽略
lRes = CryptHashData(lHash, bBlock(1), BLOCK_SIZE, 0)
Next
If lLastBlock > 0 And lRes <> 0 Then
ReDim bBlock(1 To lLastBlock) As Byte
CopyMemory bBlock(1), bStream(1 + lCount * BLOCK_SIZE), lLastBlock
lRes = CryptHashData(lHash, bBlock(1), lLastBlock, 0)
End If
If lRes <> 0 Then

' API 功能: 取 hash 数据大小
' 参数说明:

```

```

' 参数 1: hash 对像
' 参数 2: 取 hash 数据大小 (HP_HASHSIZE)
' 参数 3: 返回 hash 数据长度
lRes = CryptGetHashParam(lHash, HP_HASHSIZE, lLen, 4, 0)
If lRes <> 0 Then
ReDim bHash(0 To lLen - 1)

' API 功能: 取得 hash 数据
' 参数说明:
' 参数 1: hash 对像
' 参数 2: 取 hash 数据(值) (HP_HASHVAL)
' 参数 3: hash 数组
lRes = CryptGetHashParam(lHash, HP_HASHVAL, bHash(0), lLen, 0)
If lRes <> 0 Then
For lIdx = 0 To UBound(bHash)

' 构造 hash 值, 2 位一组, 不足补 0
HashFileStream = HashFileStream & Right("0" & Hex(bHash(lIdx)), 2)
Next
End If
End If
End If
CryptDestroyHash lHash
End If
End If
CryptReleaseContext lCtx, 0
End Function

```

### 获得文件 hash 值

```

' 参数: 文件名
' 成功返回文件的 hash, 失败返回空
' 代码与 HashFileStream 相似, 注释可参见 HashFileStream
Function HashFile(ByVal sFilename As String) As String
HashFile = ""
Dim lCtx As Long
Dim lHash As Long
Dim lFile As Long
Dim lRes As Long
Dim lLen As Long
Dim lIdx As Long
Dim bHash() As Byte
If Len(Dir(sFilename)) = 0 Then
Exit Function
End If
lRes = CryptAcquireContext(lCtx, vbNullString, vbNullString, PROV_RSA_FULL, 0)
If lRes <> 0 Then
lRes = CryptCreateHash(lCtx, ALGORITHM, 0, 0, lHash)
If lRes <> 0 Then
lFile = FreeFile

```

```

Open sFilename For Binary As #lFile
Const BLOCK_SIZE As Long = 32 * 1024&
ReDim bBlock(1 To BLOCK_SIZE) As Byte
Dim lCount As Long
Dim lBlocks As Long
Dim lLastBlock As Long
lBlocks = LOF(lFile) \ BLOCK_SIZE
lLastBlock = LOF(lFile) - lBlocks * BLOCK_SIZE
For lCount = 1 To lBlocks
Get lFile, , bBlock
lRes = CryptHashData(lHash, bBlock(1), BLOCK_SIZE, 0)
Next
If lLastBlock > 0 And lRes <> 0 Then
ReDim bBlock(1 To lLastBlock) As Byte
Get lFile, , bBlock
lRes = CryptHashData(lHash, bBlock(1), lLastBlock, 0)
End If
Close lFile
If lRes <> 0 Then
lRes = CryptGetHashParam(lHash, HP_HASHSIZE, lLen, 4, 0)
If lRes <> 0 Then
ReDim bHash(0 To lLen - 1)
lRes = CryptGetHashParam(lHash, HP_HASHVAL, bHash(0), lLen, 0)
If lRes <> 0 Then
For lIdx = 0 To UBound(bHash)
HashFile = HashFile & Right("0" & Hex(bHash(lIdx)), 2)
Next
End If
End If
End If
CryptDestroyHash lHash
End If
End If
CryptReleaseContext lCtx, 0
End Function

```

#### 5.4.6. Ini 文件读写操作模块

模块:

ModuleIniFileOperation

功能:

读写 Ini 文件。

代码:

Option Explicit

' API 声明

' 函数功能: 初始化文件中指定的条目取得字符串。

```
Private Declare Function GetPrivateProfileString Lib "kernel32" Alias "GetPrivateProfileStringA" (ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpDefault As String, ByVal lpReturnedString As String, ByVal nSize As Long, ByVal lpFileName As String) As Long
```

’ 函数功能：在初始化文件指定小节内设置一个字串

```
Private Declare Function WritePrivateProfileString Lib "kernel32" Alias "WritePrivateProfileStringA" (ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpString As Any, ByVal lpFileName As String) As Long
```

### **Ini 文件写操作**

’ 参数：

’ sFile - Ini 文件名

’ sSection - 段

’ sKeyName - 键名

’ sKeyValue - 写入键值

’ 返回值：

’ 成功返回 True，失败返回 False

```
Public Function WriteIni (ByVal sFile As String, ByVal sSection As String, ByVal sKeyName As String, ByVal sKeyValue As String) As Boolean
```

’ API 执行返回值

```
Dim lRet As Long
```

’ 写操作

```
lRet = WritePrivateProfileString(sSection, sKeyName, sKeyValue, sFile)
```

```
If lRet = 0 Then
```

’ 写文件失败，返回

```
WriteIni = False
```

```
Exit Function
```

```
Else
```

’ 成功，返回

```
WriteIni = True
```

```
Exit Function
```

```
End If
```

```
End Function
```

### **Ini 文件读操作**

’ 参数：

’ sFile - Ini 文件名

’ sSection - 段

’ sKeyName - 键值

’ 返回值：

’ 读取到的字符串

```
Public Function ReadIni (ByVal sFile As String, ByVal sSection As String, ByVal sKeyName As String) As String
```

```

' API 返回值
Dim lRet As Long
Dim sTemp As String * 255

' 读操作
lRet = GetPrivateProfileString(sSection, sKeyName, "", sTemp, 255, sFile)
If lRet = 0 Then

' 返回空
ReadIni = ""
Exit Function
Else

' 去掉不可见字符
ReadIni = TrimNull(sTemp)
End If
End Function

```

#### 5.4.7. 特征码加载和验证模块

模块:

ModuleLoadSignatures

功能:

加载和验证特征码。

代码:

```
Option Explicit
```

```
' 自定义病毒库格式
```

```
' 特征码(32 位 hash 值) + 病毒名(32 位, ) + 回车换行(chr(13)chr(10))
```

```
Private Type Signature
```

```
' Section hash (节的哈希值)
```

```
sHash As String * 32
```

```
' 病毒名称
```

```
sName As String * 32
```

```
' 回车换行
```

```
sVbCrLf As String * 2
```

```
End Type
```

```
' 定义 16 个存放病毒特征码的数组
```

```
Public uSignature0() As Signature
```

```
Public uSignature1() As Signature
```

```
Public uSignature2() As Signature
```

```
Public uSignature3() As Signature
```

```
Public uSignature4() As Signature
```

```

Public uSignature5() As Signature
Public uSignature6() As Signature
Public uSignature7() As Signature
Public uSignature8() As Signature
Public uSignature9() As Signature
Public uSignatureA() As Signature
Public uSignatureB() As Signature
Public uSignatureC() As Signature
Public uSignatureD() As Signature
Public uSignatureE() As Signature
Public uSignatureF() As Signature
Public uSignatureG() As Signature

```

#### · 检查病毒库文件完整性

’ 返回值：病毒库完整返回 True，否则返回 False

```
Public Function CheckSignatureFile() As Boolean
```

’ 软件当前路径

```
Dim sAppPath As String
```

```
sAppPath = App.Path
```

’ 检查路径完整性

```
If Right(sAppPath, 1) <> "\" Then
```

```
sAppPath = sAppPath & "\"
```

```
End If
```

```
Dim sSignatureFile As String
```

’ 检查 0~9 病毒库文件是否存在

```
Dim i As Long
```

```
For i = 0 To 9
```

’ 病毒库文件

```
sSignatureFile = sAppPath & i & ".sig"
```

```
If Dir(sSignatureFile) = "" Then
```

```
CheckSignatureFile = False
```

```
Exit Function
```

```
End If
```

```
Next
```

’ 检查 A~Z 病毒库文件是否存在

```
Dim j As Long
```

’ 65~71 是字母 A~F 的 Ascii 码值，使用 VB 内置的 Chr 函数可以将数字转化为字母，方便使用 For 循环操作

```
For j = 65 To 71
```

’ 病毒库文件

```
sSignatureFile = sAppPath & Chr(j) & ".sig"
```

```
If Dir(sSignatureFile) = "" Then
```

```
CheckSignatureFile = False
```

```
Exit Function
End If
Next
CheckSignatureFile = True
End Function
```

#### · 从病毒库文件加载特征码

```
Public Function LoadSignatures() As Long

' 软件当前路径
Dim sAppPath As String
sAppPath = App.Path

' 检查路径完整性
If Right(sAppPath, 1) <> "\" Then
sAppPath = sAppPath & "\"
End If

' 病毒库文件
Dim sSignatureFile As String

' 文件句柄
Dim lFile As Long

' 文件长度
Dim lFileLen As Long

' 特征码个数
Dim lSignatureCount As Long
lSignatureCount = 0

' 加载 0~9 病毒库文件
Dim i As Long
For i = 0 To 9
DoEvents

' 病毒库文件
sSignatureFile = sAppPath & i & ".sig"

' 从病毒库文件加载特征码
Select Case i
Case "0"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignature0(1 To lFileLen / 66) As Signature
```



```

Get lFile, , uSignature0

' 计算特征码数量
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
Case "1"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignature1(1 To lFileLen / 66) As Signature
Get lFile, , uSignature1
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
Case "2"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignature2(1 To lFileLen / 66) As Signature
Get lFile, , uSignature2
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
Case "3"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignature3(1 To lFileLen / 66) As Signature
Get lFile, , uSignature3
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
Case "4"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignature4(1 To lFileLen / 66) As Signature
Get lFile, , uSignature4

```

```

lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
Case "5"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignature5(1 To lFileLen / 66) As Signature
Get lFile, , uSignature5
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
Case "6"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignature6(1 To lFileLen / 66) As Signature
Get lFile, , uSignature6
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
Case "7"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignature7(1 To lFileLen / 66) As Signature
Get lFile, , uSignature7
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
Case "8"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignature8(1 To lFileLen / 66) As Signature
Get lFile, , uSignature8
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile

```

```

Case "9"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignature9(1 To lFileLen / 66) As Signature
Get lFile, , uSignature9
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
End Select
Next

' 加载 A~F 病毒库文件
Dim j As Long
For j = 65 To 70
DoEvents

' 病毒库文件
sSignatureFile = sAppPath & Chr(j) & ".sig"

' 从病毒库文件加载特征码
Select Case UCase(Chr(j))
Case "A"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignatureA(1 To lFileLen / 66) As Signature
Get lFile, , uSignatureA
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
Case "B"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignatureB(1 To lFileLen / 66) As Signature
Get lFile, , uSignatureB
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
Case "C"
lFile = FreeFile

```

```

Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignatureC(1 To lFileLen / 66) As Signature
Get lFile, , uSignatureC
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
Case "D"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignatureD(1 To lFileLen / 66) As Signature
Get lFile, , uSignatureD
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
Case "E"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignatureE(1 To lFileLen / 66) As Signature
Get lFile, , uSignatureE
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
Case "F"
lFile = FreeFile
Open sSignatureFile For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignatureF(1 To lFileLen / 66) As Signature
Get lFile, , uSignatureF
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile
End Select
Next

' "G"
lFile = FreeFile

```

```

Open sAppPath & "G.sig" For Binary As #lFile
lFileLen = LOF(lFile)

' 判断文件是否为空
If lFileLen <> 0 Then
ReDim uSignatureG(1 To lFileLen / 66) As Signature
Get lFile, , uSignatureG
lSignatureCount = lSignatureCount + (lFileLen / 66)
End If
Close lFile

' 返回特征码总加载数
LoadSignatures = lSignatureCount
End Function

```

#### 5.4.8. 特征码匹配检测模块

模块:

ModuleMatchSignature

功能:

特征码匹配检测。

代码:

#### 特征码匹配

' 成功返回病毒名称，失败返回空

```
Public Function MatchSignature(ByVal sHash As String) As String
```

' 初始化返回值

```
MatchSignature = ""
```

' 检查传入的参数 sHash 是否为空

```
If sHash = "" Then
```

```
Exit Function
```

```
End If
```

' 取 sHash 第一位为标志，确定使用特征码类

```
Dim sFlagWord As String
```

```
sFlagWord = UCase(Left(sHash, 1))
```

```
Dim i As Long
```

```
Select Case sFlagWord
```

```
Case "0"
```

```
For i = 1 To UBound(uSignature0)
```

```
If uSignature0(i).sHash = sHash Then
```

' 匹配成功，返回病毒名称

```
MatchSignature = uSignature0(i).sName
```

```
Exit Function
```

```
End If
```

```
Next i
```

```

Case "1"
For i = 1 To UBound(uSignature1)
If uSignature1(i).sHash = sHash Then

' 匹配成功，返回病毒名称
MatchSignature = uSignature1(i).sName
Exit Function
End If
Next i
Case "2"
For i = 1 To UBound(uSignature2)
If uSignature2(i).sHash = sHash Then

' 匹配成功，返回病毒名称
MatchSignature = uSignature2(i).sName
Exit Function
End If
Next i
Case "3"
For i = 1 To UBound(uSignature3)
If uSignature3(i).sHash = sHash Then

' 匹配成功，返回病毒名称
MatchSignature = uSignature3(i).sName
Exit Function
End If
Next i
Case "4"
For i = 1 To UBound(uSignature4)
If uSignature4(i).sHash = sHash Then

' 匹配成功，返回病毒名称
MatchSignature = uSignature4(i).sName
Exit Function
End If
Next i
Case "5"
For i = 1 To UBound(uSignature5)
If uSignature5(i).sHash = sHash Then

' 匹配成功，返回病毒名称
MatchSignature = uSignature5(i).sName
Exit Function
End If
Next i
Case "6"
For i = 1 To UBound(uSignature6)
If uSignature6(i).sHash = sHash Then

' 匹配成功，返回病毒名称
MatchSignature = uSignature6(i).sName

```

```

Exit Function
End If
Next i
Case "7"
For i = 1 To UBound(uSignature7)
If uSignature7(i).sHash = sHash Then

' 匹配成功，返回病毒名称
MatchSignature = uSignature7(i).sName
Exit Function
End If
Next i
Case "8"
For i = 1 To UBound(uSignature8)
If uSignature8(i).sHash = sHash Then

' 匹配成功，返回病毒名称
MatchSignature = uSignature8(i).sName
Exit Function
End If
Next i
Case "9"
For i = 1 To UBound(uSignature9)
If uSignature9(i).sHash = sHash Then

' 匹配成功，返回病毒名称
MatchSignature = uSignature9(i).sName
Exit Function
End If
Next i
Case "A"
For i = 1 To UBound(uSignatureA)
If uSignatureA(i).sHash = sHash Then

' 匹配成功，返回病毒名称
MatchSignature = uSignatureA(i).sName
Exit Function
End If
Next i
Case "B"
For i = 1 To UBound(uSignatureB)
If uSignatureB(i).sHash = sHash Then

' 匹配成功，返回病毒名称
MatchSignature = uSignatureB(i).sName
Exit Function
End If
Next i
Case "C"
For i = 1 To UBound(uSignatureC)
If uSignatureC(i).sHash = sHash Then

```

```

' 匹配成功, 返回病毒名称
MatchSignature = uSignatureC(i).sName
Exit Function
End If
Next i
Case "D"
For i = 1 To UBound(uSignatureD)
If uSignatureD(i).sHash = sHash Then

' 匹配成功, 返回病毒名称
MatchSignature = uSignatureD(i).sName
Exit Function
End If
Next i
Case "E"
For i = 1 To UBound(uSignatureE)
If uSignatureE(i).sHash = sHash Then

' 匹配成功, 返回病毒名称
MatchSignature = uSignatureE(i).sName
Exit Function
End If
Next i
Case "F"
For i = 1 To UBound(uSignatureF)
If uSignatureF(i).sHash = sHash Then

' 匹配成功, 返回病毒名称
MatchSignature = uSignatureF(i).sName
Exit Function
End If
Next i
End Select
For i = 1 To UBound(uSignatureG)
If uSignatureG(i).sHash = sHash Then

' 匹配成功, 返回病毒名称
MatchSignature = uSignatureG(i).sName
Exit Function
End If
Next i
End Function

```

#### 5.4.9. PE 文件操作和文件扫描模块

模块:

ModulePeOperation

功能:



PE 文件操作和文件扫描。

代码:

Option Explicit

' 函数功能: 将一块内存的数据从一个位置复制到另一个位置。

```
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (Destination As Any, Source As Any, ByVal Length As Long)
```

' 函数功能: 这是一个多功能的函数, 可打开或创建以下对象, 并返回可访问的句柄: 控制台, 通信资源, 目录(只读打开), 磁盘驱动器, 文件, 邮箱, 管道。

' 在本程序中用来打开文件

```
Private Declare Function CreateFile Lib "kernel32" Alias "CreateFileA" (ByVal lpFileName As String, ByVal dwDesiredAccess As Long, ByVal dwShareMode As Long, lpSecurityAttributes As Any, ByVal dwCreationDisposition As Long, ByVal dwFlagsAndAttributes As Long, ByVal hTemplateFile As Long) As Long
```

' 函数功能: 从文件指针指向的位置开始将数据读出到一个文件中, 且支持同步和异步操作

```
Private Declare Function ReadFile Lib "kernel32" (ByVal hFile As Long, lpBuffer As Any, ByVal nNumberOfBytesToRead As Long, lpNumberOfBytesRead As Long, lpOverlapped As Any) As Long
```

' 函数功能: 将数据写入一个文件。该函数比 `fwrite` 函数要灵活的多。也可将这个函数应用于对通信设备、管道、套接字以及邮箱的处理

```
Private Declare Function WriteFile Lib "kernel32" (ByVal hFile As Long, lpBuffer As Any, ByVal nNumberOfBytesToWrite As Long, lpNumberOfBytesWritten As Long, lpOverlapped As Any) As Long
```

' 函数功能: 在一个文件中设置当前的读写位置

```
Private Declare Function SetFilePointer Lib "kernel32" (ByVal hFile As Long, ByVal lDistanceToMove As Long, lpDistanceToMoveHigh As Long, ByVal dwMoveMethod As Long) As Long
```

' 在本程序中, `CloseHandle` 用来关闭打开的文件

```
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long
```

' 函数功能: 获取文件大小

```
Private Declare Function GetFileSizeEx Lib "kernel32.dll" (ByVal hFile As Long, ByRef lpFileSize As LARGE_INTEGER) As Long
```

' 自定义类型: 大的整形。有些数值特别大, VB 程序中变量容纳不了如此大的变量, 使用这种方式, 将变量分为高位部分和低位部分进行存储

```
Private Type LARGE_INTEGER
```

```
lowpart As Long
```

```
highpart As Long
```

```
End Type
```

' PE 文件的 DOS 头

```
Private Type IMAGE_DOS_HEADER
```

```
Magic As Integer
```

```
cblp As Integer
```

```
cp As Integer
```

```
crlc As Integer
```

```
cparhdr As Integer
```

```
minalloc As Integer
```

```

maxalloc As Integer
ss As Integer
sp As Integer
csum As Integer
ip As Integer
cs As Integer
lfarlc As Integer
ovno As Integer
res(3) As Integer
oemid As Integer
oeminfo As Integer
res2(9) As Integer
lfanew As Long
End Type

```

’ PE 文件的文件头

```

Private Type IMAGE_FILE_HEADER
Machine As Integer
NumberOfSections As Integer
TimeDateStamp As Long
PointerToSymbolTable As Long
NumberOfSymbols As Long
SizeOfOptionalHeader As Integer
Characteristics As Integer
End Type

```

’ PE 文件的目录数据

```

Private Type IMAGE_DATA_DIRECTORY
DataRVA As Long
DataSize As Long
End Type

```

’ PE 文件的可选文件头

```

Private Type IMAGE_OPTIONAL_HEADER
Magic As Integer
MajorLinkVer As Byte
MinorLinkVer As Byte
CodeSize As Long
InitDataSize As Long
unInitDataSize As Long
EntryPoint As Long
CodeBase As Long
DataBase As Long
ImageBase As Long
SectionAlignment As Long
FileAlignment As Long
MajorOSVer As Integer
MinorOSVer As Integer
MajorImageVer As Integer
MinorImageVer As Integer
MajorSSVer As Integer

```

```

MinorSSVer As Integer
Win32Ver As Long
ImageSize As Long
HeaderSize As Long
Checksum As Long
Subsystem As Integer
DLLChars As Integer
StackRes As Long
StackCommit As Long
HeapReserve As Long
HeapCommit As Long
LoaderFlags As Long
RVAsAndSizes As Long
DataEntries(15) As IMAGE_DATA_DIRECTORY
End Type

```

' PE 文件的节头

```
Private Type IMAGE_SECTION_HEADER
```

' 节名

```
SectionName(7) As Byte
```

```
VirtualSize As Long
```

' 节的 RVA (虚拟相对地址)

```
VirtualAddress As Long
```

' 节大小

```
SizeOfRawData As Long
```

' 节的文件偏移量

```
PointerToRawData As Long
```

```
PointerToRelocations As Long
```

```
PLineNums As Long
```

```
RelocCount As Integer
```

```
LineCount As Integer
```

' 节的属性, 如可读可写

```
Characteristics As Long
```

```
End Type
```

```
Private Const GENERIC_READ = &H80000000
```

```
Private Const FILE_SHARE_READ = &H1
```

```
Private Const OPEN_EXISTING = 3
```

**扫描文件 Section, 以 hash(节大小+节 hash)进行特征码匹配检测病毒**

' 特别说明: 扫描文件入口地址所在节

' 参数: hFile 文件句柄, 传入前检查合法性

' 返回值: 检测到返回病毒名, 否则返回空

```

Private Function ScanSections(hFile As Long) As String

' 初始化返回
ScanSections = ""

' DOS 头
Dim uDos As IMAGE_DOS_HEADER

' 文件头
Dim uFile As IMAGE_FILE_HEADER

' 可选头
Dim uOptional As IMAGE_OPTIONAL_HEADER

' 节头
Dim uSections() As IMAGE_SECTION_HEADER
Dim lBytesRead As Long
Dim i As Long
Dim bTempMemory() As Byte

' 把指针移动到文件头
SetFilePointer hFile, 0, 0, 0

' 读取 DOS 头
ReadFile hFile, uDos, Len(uDos), lBytesRead, ByVal 0&

' 根据 DOS 头的 lfanew, 指向 PE 头+4 的地方(即文件头开始地址)
SetFilePointer hFile, ByVal uDos.lfanew + 4, 0, 0

' 读取文件头
ReadFile hFile, uFile, Len(uFile), lBytesRead, ByVal 0&

' 读取可选头
ReadFile hFile, uOptional, Len(uOptional), lBytesRead, ByVal 0&

' 判断节数是否正确
If uFile.NumberOfSections < 1 Or uFile.NumberOfSections > 99 Then
Exit Function
End If

' 重写义节数(uFile.NumberOfSections 是节个数)
ReDim uSections(uFile.NumberOfSections - 1) As IMAGE_SECTION_HEADER

' 读取节头
ReadFile hFile, uSections(0), Len(uSections(0)) * uFile.NumberOfSections, lBytesRead, ByVal 0&

' 扫描每个节
For i = 0 To UBound(uSections)

' 防止 CPU 占用过高及程序无响应
DoEvents

```

```

' SizeOfRawData 有为 0 的时候, 如果为 0 就要出错
If uSections(i).SizeOfRawData > 0 Then

' 最大段不超过 10MB
If (uSections(i).SizeOfRawData / 1024 / 1024) <= 10 Then

ReDim bTempMemory(1 To uSections(i).SizeOfRawData) As Byte

' 设置指针到节的起始处
SetFilePointer hFile, ByVal uSections(i).PointerToRawData, 0, 0

' 读取整节数据, 读取后 TemoMemory 里存放的是要 hash 的数据, 完整的 section
ReadFile hFile, bTempMemory(1), uSections(i).SizeOfRawData, lBytesRead, ByVal 0&
Dim sTargetStr As String

' 构造特征码: 节大小+节的 hash 值
sTargetStr = uSections(i).SizeOfRawData & ":" & LCase(HashFileStream(bTempMemory))
Dim bTempStrByte() As Byte
bTempStrByte = sTargetStr

' 用哈希算法加密特征码(hash)
sTargetStr = HashFileStream(bTempStrByte)

' 特征码匹配检测
Dim sMatchResult As String
sMatchResult = MatchSignature(sTargetStr)

' 如果不为空表示检测到是病毒, 返回
If sMatchResult <> "" Then
ScanSections = Trim(sMatchResult)
Exit Function
End If
End If
End If
Next i
End Function

' 扫描 PE 文件
Public Function ScanFile(sFile As String) As String

' 初始化返回值
ScanFile = ""

' 判断文件大小, 如果文件太大, 则不可能是病毒, 直接放行
Dim lFileSize As Long
Dim uSize As LARGE_INTEGER

' 打开文件
lFileSize = CreateFile(sFile, GENERIC_READ, FILE_SHARE_READ, ByVal 0&, OPEN_EXISTING, ByVal 0&, ByVal 0&)

```

```

' 获取文件大小
GetFileSizeEx lFileSize, uSize

' 关闭文件
CloseHandle lFileSize
If uSize.lowpart / 1024 / 1024 > 15 Then
Exit Function
End If

' 打开文件
Dim lFileHwnd As Long
lFileHwnd = CreateFile(sFile, ByVal &H80000000, 0, ByVal 0&, 3, 0, ByVal 0)
If lFileHwnd Then

' 打开文件, 检查是否是 PE 文件
Dim bBuffer(12) As Byte
Dim lBytesRead As Long
Dim uDosHeader As IMAGE_DOS_HEADER

' 读取文件的 DOS 头
ReadFile lFileHwnd, uDosHeader, ByVal Len(uDosHeader), lBytesRead, ByVal 0&
CopyMemory bBuffer(0), uDosHeader.Magic, 2

' 第一个检查点
If (Chr(bBuffer(0)) & Chr(bBuffer(1)) = "MZ") Then

' 把指针设置到 PE 头结构偏移处
SetFilePointer lFileHwnd, uDosHeader.lfanew, 0, 0

' 读取 4byte
ReadFile lFileHwnd, bBuffer(0), 4, lBytesRead, ByVal 0&

' 检查是否是 PE(回车)(回车)
If (Chr(bBuffer(0)) = "P") And (Chr(bBuffer(1)) = "E") And (bBuffer(2) = 0) And (bBuffer(3) = 0) Then

' 以 Section 方式扫描文件, 成功返回病毒名, 失败返回空
Dim sScanSectionResult As String
sScanSectionResult = ScanSections(lFileHwnd)

' 如果返回不为空, 表示检测到是病毒, 则关闭文件, 返回
If sScanSectionResult <> "" Then
ScanFile = sScanSectionResult
CloseHandle lFileHwnd
Exit Function
End If
End If
End If
End If
CloseHandle lFileHwnd
End Function

```

#### 5.4.10. 进程优先级设置模块

模块:

ModuleProcessPriority

功能:

设置进程优化级。

代码:

' 函数功能: 打开进程

```
Private Declare Function OpenProcess Lib "kernel32" (ByVal dwDesiredAccess As Long, ByVal bInheritHandle As Long, ByVal dwProcessID As Long) As Long
```

' 函数功能: 设置进程的优先级

```
Private Declare Function SetPriorityClass Lib "kernel32" (ByVal hProcess As Long, ByVal dwPriorityClass As Long) As Long
```

```
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long
```

```
Private Declare Function GetCurrentProcessId Lib "kernel32" () As Long
```

' 打开进程常量

```
Private Const PROCESS_QUERY_INFORMATION As Long = &H400
```

```
Private Const PROCESS_SET_INFORMATION As Long = &H200
```

'设置进程优先级

' 正常优先级=32, 低于正常优先级=16384

```
Public Function SetProcessPriority(ByVal Priority As Long) As Boolean
```

```
Dim lProcessID As Long
```

' 取得当前进程 ID

```
lProcessID = GetCurrentProcessId()
```

' 打开进程

```
Dim lOpenProcessReturn As Long
```

```
lOpenProcessReturn = OpenProcess(PROCESS_QUERY_INFORMATION Or PROCESS_SET_INFORMATION, 0, lProcessID)
```

```
If lOpenProcessReturn Then
```

' 设置优先级

```
Call SetPriorityClass(lOpenProcessReturn, Priority)
```

' 设置函数返回值

```
SetProcessPriority = True
```

```
Else
```

' 设置函数返回值

```
SetProcessPriority = False
```

```
End If
```

' 关闭句柄

```
Call CloseHandle(lOpenProcessReturn)
End Function
```

#### 5.4.11. 公共函数模块

模块:

```
ModulePublicFunctions
```

功能:

公共函数, 包含了在程序中多次使用的全局函数。

代码:

```
Option Explicit
Private Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hWnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long

' 函数功能: 设置操作系统实际划分给进程使用的内存容量
Private Declare Function SetProcessWorkingSetSize Lib "kernel32" (ByVal hProcess As Long, ByVal dwMinimumWorkingSetSize As Long, ByVal dwMaximumWorkingSetSize As Long) As Long
Private Declare Function GetCurrentProcess Lib "kernel32" () As Long
```

**扩展功能的Trim, 可去掉字符串中的chr(0), 即不可见字符, 回车\换行**

' 参数: 字符串

' 返回值: 去掉 chr(0) 后的字符串

```
Public Function TrimNull(sStr As String) As String
If InStr(1, sStr, Chr(0)) Then
TrimNull = Trim(Left(sStr, InStr(1, sStr, Chr(0)) - 1))
Else
TrimNull = Trim(sStr)
End If
End Function
```

**完整退出程序**

' 恢复子类化、停止 Hook, 完整退出程序

```
Public Function TrueEnd()
```

' 停止自我保护, 此函数是由我们的 DLL 导出, 导出函数在后文将会介绍

```
Call SafeGuard(False)
```

' 停止主动防御, 此函数是由我们的 DLL 导出, 导出函数在后文将会介绍

```
Call ActiveDefense(False)
```

' 恢复窗体消息处理

```
SetWindowLong FormScan.hWnd, -4, lSubClassOldAddress
```

' 移除托盘图标

```
RemoveTrayIcon
```



```

' 改变窗体卸载标志
bEnableUnloadForm = True

' 卸载程序所包含的所有窗体
Unload FormSetting
Unload FormUpdate
Unload FormSelectFolders
Unload FormMenu
Unload FormScan
Unload FormQuarantine
Unload FormRightClickScanResult
Unload FormSkin
Unload FormActiveDefense
Unload FormAlertMessage
Unload FormBlackAndWhiteList
Unload FormScanFinish
Unload FormAlertVirus
Unload FormAbout
Unload FormSoftware
End Function

```

#### \*使软件使用最少内存

```

Public Function MiniUseMemory()

' 此函数将软件不活动使用的内存，放到系统磁盘缓存中
SetProcessWorkingSetSize GetCurrentProcess(), -1&, -1&
End Function

```

#### \*产生随机数

```

Public Function RndChr(Numbers As Long) As String
Dim i As Long
Dim sAscii As String
For i = 1 To Numbers

' 对随机数生成器做初始化的动作。
Randomize

' 生成 65-90 之间的随机数值。
sAscii = Int((90 - 65) * Rnd + 65)

' 将数字转换为字母并拼合在一起
RndChr = RndChr + Chr(sAscii)
Next i

' 将整体字符串转化为大写状态
RndChr = UCase(RndChr)
End Function

```

## 写隔离记录文件

```
' 参数:
'sS:文件原路径
'sQ:文件隔离路径
Public Function WriteQuarantineFile(ByVal ss As String, ByVal sQ As String)

' 隔离记录文件
Dim sQuarantineFile As String
If Right(App.Path, 1) = "\" Then
sQuarantineFile = App.Path & "Quarantine.ini"
Else
sQuarantineFile = App.Path & "\"Quarantine.ini"
End If

' 先打开隔离记录文件并获取目前最大记录值（也就是当前隔离着多少文件）
Dim lMaxID As Long
lMaxID = ReadIni(sQuarantineFile, "Count", "MaxID")

' 更新隔离记录最大值（即：将原值加 1 后重新保存回文件中）
Call WriteIni(sQuarantineFile, "Count", "MaxID", lMaxID + 1)

' 向 SourceFile 字段的最大记录值项写入隔离前文件位置
Call WriteIni(sQuarantineFile, "SourceFile", Format(lMaxID + 1, "00000"), ss)

' 向 QuarantineFile 字段的最大记录值项写入隔离文件位置
Call WriteIni(sQuarantineFile, "QuarantineFile", Format(lMaxID + 1, "00000"), sQ)
End Function
```

## 总换肤函数，调用各个窗体中的换肤函数

```
Public Function ReSkinAll()

' 加载主动防御拦截窗体的皮肤
FormActiveDefense.ReSkinMe
DoEvents

' 加载右下角弹出的消息提示窗体的皮肤
FormAlertMessage.ReSkinMe
DoEvents

' 加载检测到病毒时弹出警告窗体的皮肤
FormAlertVirus.ReSkinMe
DoEvents

' 加载黑白名单管理窗体的皮肤
FormBlackAndWhiteList.ReSkinMe
DoEvents
```

```

' 加载隔离文件管理窗体的皮肤
FormQuarantine.ReSkinMe
DoEvents

' 加载右键扫描结果窗体的皮肤
FormRightClickScanResult.ReSkinMe
DoEvents

' 加载扫描窗体（即主界面）的窗体
FormScan.ReSkinMe
DoEvents

' 加载扫描结果窗体的皮肤
FormScanFinish.ReSkinMe
DoEvents

' 加载自定义扫描时选择目录窗体的皮肤
FormSelectFolders.ReSkinMe
DoEvents

' 加载设置窗体的皮肤
FormSetting.ReSkinMe
DoEvents

' 加载换肤窗体的皮肤
FormSkin.ReSkinMe
DoEvents

' 加载升级窗体的皮肤
FormUpdate.ReSkinMe
DoEvents

' 加载关于窗体的皮肤
FormAbout.ReSkinMe
DoEvents
End Function

```

#### 5.4.12. 公共变量模块

模块:

```
ModulePublicVariables
```

功能:

公共变量，这里定义的变量可以在所有窗体和模块中被访问。

代码:

```
Option Explicit
```

```
' 主动防御接收消息窗体子类化旧地址
```

```
Public lSubClassOldAddress As Long

' 是否可以卸载窗体
Public bEnableUnloadForm As Boolean

' 是否进行自定义扫描标识
Public bDoCustmerScan As Boolean

' 向 dll 传送的放行或阻止程序标识
Public lUserAction As Long

' 皮肤路径
Public sSkin As String
```

#### 5.4.13. 右键菜单关联与解除模块

模块:

ModuleRightClickMenu

功能:

右键菜单关联与解除。

代码:

```
Option Explicit
```

```
Private Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hKey As Long) As Long
```

' 建立键

```
Private Declare Function RegCreateKey Lib "advapi32.dll" Alias "RegCreateKeyA" (ByVal hKey As Long, ByVal lpSubKey As String, phkResult As Long) As Long
```

' 写入启动值

```
Private Declare Function RegSetValueEx Lib "advapi32" Alias "RegSetValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal Reserved As Long, ByVal dwType As Long, ByVal lpData As String, ByVal cbData As Long) As Long
```

' 删除项目

```
Private Declare Function RegDeleteKey Lib "advapi32.dll" Alias "RegDeleteKeyA" (ByVal hKey As Long, ByVal lpSubKey As String) As Long
```

' 打开注册表 subkey 的 hkey

```
Private Declare Function RegOpenKey Lib "advapi32.dll" Alias "RegOpenKeyA" (ByVal hKey As Long, ByVal lpSubKey As String, phkResult As Long) As Long
```

```
Private Const HKEY_CLASSES_ROOT = &H80000000
```

```
Private Const REG_SZ = 1
```

#### 向系统增加文件和文件夹的右键菜单

```
Public Function AddFileRightClickMenu() As Boolean
```

```

Dim hKey As Long, ret As Long

' 建立注册表项
RegCreateKey HKEY_CLASSES_ROOT, "*\shell\海奇杀毒\command", hKey

' 设置右键菜单项目
ret = RegSetValueEx(hKey, "", 0, REG_SZ, ByVal App.Path & "\" & App.EXEName & ".exe %1", ByVal
LenB(StrConv(App.Path & "\" & App.EXEName & ".exe %1", vbFromUnicode)) + 1)
If ret = 0 Then

' 添加成功
AddFileRightClickMenu = True
Else

' 添加失败
AddFileRightClickMenu = False
End If
RegCloseKey hKey
End Function

```

#### 删除右键菜单

```

Public Function DeleteRightClickMenu() As Boolean

' 这里必须分步执行，如同删除文件夹一样不能删除非空的文件夹

' 删除文件关联的右键菜单
DeleteKey HKEY_CLASSES_ROOT, "*\shell\海奇杀毒", "command"
DoEvents
DeleteKey HKEY_CLASSES_ROOT, "*\shell", "海奇杀毒"
DoEvents

' 删除文件夹关联的右键菜单
DeleteKey HKEY_CLASSES_ROOT, "Directory\shell\海奇杀毒", "command"
DoEvents
DeleteKey HKEY_CLASSES_ROOT, "Directory\shell", "海奇杀毒"
DoEvents
DeleteRightClickMenu = True
End Function

```

#### 删除注册表键函数

```

Function DeleteKey(RootKey As Long, ParentKeyName As String, SubKeyName As String)
Dim hKey As Long
RegOpenKey RootKey, ParentKeyName, hKey
RegDeleteKey hKey, SubKeyName
RegCloseKey hKey
End Function

```

这函数是用来单独添加到目录的

```

Public Function AddFolderRightClickMenu() As Boolean

' 把应用程序加入右键菜单
Dim hKey As Long, ret As Long

' 建立注册表项
RegCreateKey HKEY_CLASSES_ROOT, "Directory\shell\海奇杀毒\command", hKey

' 设置右键菜单项目
ret = RegSetValueEx(hKey, "", 0, REG_SZ, ByVal App.Path & "\" & App.EXENAME & ".exe %1", ByVal
LenB(StrConv(App.Path & "\" & App.EXENAME & ".exe %1", vbFromUnicode)) + 1)
If ret = 0 Then
AddFolderRightClickMenu = True
Else
AddFolderRightClickMenu = False
End If

' 关闭注册表项
RegCloseKey hKey
End Function

```

#### 5.4.14. 自我保护功能实现模块

模块:

ModuleSafeGuard

功能:

自我保护功能实现模块。

代码:

```
Option Explicit
```

```
' API 声明
```

```
' 以下两个 API 函数是用 VC 写成，用于实现软件自我保护，相关的代码会在后文 DLL 模块中展示。
```

```
Private Declare Function SafeGuardON Lib "SafeGuard.dll" (ByVal TargetHwnd As Long) As Boolean
```

```
Private Declare Function SafeGuardOFF Lib "SafeGuard.dll" () As Boolean
```

```
' 获取进程唯一标识 ID 值
```

```
Private Declare Function GetCurrentProcessId Lib "kernel32" () As Long
```

**开启或关闭自我保护，防止进程被非法结束**

```
' 参数: Boolean 型，传入 True 时启动自我保护，传入 False 时关闭自我保护
```

```
' 返回值: True 表示成功，False 表示失败
```

```
Public Function SafeGuard(ByVal bOperation As Boolean) As Boolean
```

```
Dim lHwnd As Long
```

```
' 取得当前进程 Pid 值
```

```

lHwnd = GetCurrentProcessId
Dim bRet As Boolean

' 判断操作行为，执行保护或停止保护
If bOperation = True Then
bRet = SafeGuardON(lHwnd)

' 返回结果
' 注：VC 中 return true 返回的 true 值，跟 VB 中 true 值不同。故以下代码不能使用 bret=true 进行比较。
' VC True=1
' VB true=0ffffff
If bRet Then
SafeGuard = True
Else
SafeGuard = False
End If
Else
SafeGuardOFF
SafeGuard = True
End If
End Function

```

#### 5.4.15. 窗体消息处理模块

模块：

```
ModuleSubClass
```

功能：

处理窗体消息，这其中会有大量操作，比如接收程序启动消息、弹出右键菜单、重建托盘图标等。

代码：

```

Option Explicit
' API 声明
Private Declare Function SetForegroundWindow Lib "user32" (ByVal hWnd As Long) As Long

' 函数功能：将消息信息传送给指定的窗口过程
Private Declare Function CallWindowProc Lib "user32" Alias "CallWindowProcA" (ByVal lpPrevWndFunc As Long,
ByVal hWnd As Long, ByVal Msg As Long, ByVal lParam As Long, ByVal lParam As Long) As Long
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (Destination As Any, Source As Any,
ByVal Length As Long)
Private Declare Function WinExec Lib "kernel32" (ByVal lpCmdLine As String, ByVal nCmdShow As Long) As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long,
ByVal lParam As Long, ByVal lParam As Any) As Long
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
Private Declare Function GetFileSizeEx Lib "kernel32.dll" (ByVal hFile As Long, ByRef lpFileSize As
LARGE_INTEGER) As Long
Private Declare Function CreateFile Lib "kernel32" Alias "CreateFileA" (ByVal lpFileName As String, ByVal
dwDesiredAccess As Long, ByVal dwShareMode As Long, lpSecurityAttributes As Any, ByVal dwCreationDisposition As Long,
ByVal dwFlagsAndAttributes As Long, ByVal hTemplateFile As Long) As Long

```

```
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long
```

```
' 自定义类型
```

```
Private Type COPYDATASTRUCT
```

```
dwData As Long
```

```
cbData As Long
```

```
lpData As Long
```

```
End Type
```

```
Private Type LARGE_INTEGER
```

```
lowpart As Long
```

```
highpart As Long
```

```
End Type
```

```
Private Const GENERIC_READ = &H80000000
```

```
Private Const FILE_SHARE_READ = &H1
```

```
Private Const OPEN_EXISTING = 3
```

```
Private Const WM_SYSCOMMAND = &H112
```

```
Private Const SC_RESTORE = &HF120&
```

```
' Explorer 出错时，重建托盘图标
```

```
Public WM_TASKBARCREATED As Long
```

## 主动防御消息处理函数

```
' 接收 Dll 拦截到的进程启动消息，并进行处理
```

```
' DLL 函数根据此操作结果进行放行或禁止运行操作
```

```
' 参数：句柄、自定义的消息类型，被创建进程父进程 PID，进程路径等
```

```
Public Function SubClassMessage(ByVal lHwnd As Long, ByVal lMessage As Long, ByVal lParentPID As Long,  
ByVal lParam As Long) As Long
```

```
On Error Resume Next
```

```
Dim sTemp As String
```

```
Dim uCDS As COPYDATASTRUCT
```

```
' 双击弹出窗口
```

```
If lParam = &H203 Then
```

```
' 设置弹出窗体的位置在屏幕中央
```

```
With FormScan
```

```
.Top = (Screen.Height - .Height) / 2
```

```
.Left = (Screen.Width - .Width) / 2
```

```
.Show
```

```
Call SendMessage(.hWnd, WM_SYSCOMMAND, SC_RESTORE, 0)
```

```
End With
```

```
Exit Function
```

```
End If
```

```
' 右键弹出菜单
```

```
If lParam = &H204 Then
```



```

' 将菜单窗体置于屏幕最前。必须这样操作一下菜单窗体，才能正常的弹出右键菜单
SetForegroundWindow FormMenu.hWnd
FormMenu.PopupMenu FormMenu.m_TrayMenu
Exit Function
End If

' 判是否是 DLL 传来的消息
If lMessage = &H4A Then

' 使软件使用最少内存
MiniUseMemory

' 取得传来的数据长度, Dll 中传来 512byte
CopyMemory uCDS, ByVal lParam, Len(uCDS)
sTemp = Space(uCDS.cbData)

' 取得传来的数据
CopyMemory ByVal sTemp, ByVal uCDS.lpData, uCDS.cbData

' 获取文件名
Dim sFile As String
sFile = Left(sTemp, InStr(1, sTemp, "$") - 1)
sTemp = Right(sTemp, Len(sTemp) - InStr(1, sTemp, "$"))

' 获取文件的命令行参数
Dim sCommand As String
sCommand = TrimNull(Left(sTemp, InStr(1, sTemp, "$") - 1))

' 判断文件名是否为空
If TrimNull(sFile) = "" Then

' 设置自动放行标识，并返回给 Dll，Dll 在接收到此标识后，会自动放行正在拦截的程序，在后文介绍的 VC 写的
DLL 代码中，会看到对数字 915 的判断操作
SubClassMessage = 915
Exit Function
End If

' 判断是否是程序自己
Dim sTrimFile As String
sTrimFile = TrimNull(sFile)
If InStr(1, LCase(sTrimFile), LCase(App.Path & "\" & App.EXENAME)) Or InStr(1, LCase(App.Path & "\" &
App.EXENAME), LCase(sTrimFile)) Then

' 如果是程序自己同样自动放行
SubClassMessage = 915
Exit Function
End If

' 判断文件大小，如果文件太大，则不可能是病毒，直接放行
Dim lFileSize As Long

```

```

Dim uSize As LARGE_INTEGER
' 打开文件
lFileSize = CreateFile(sFile, GENERIC_READ, FILE_SHARE_READ, ByVal 0&, OPEN_EXISTING, ByVal 0&, ByVal 0&)

' 获取文件大小
GetFileSizeEx lFileSize, uSize

' 关闭文件
CloseHandle lFileSize
If uSize.lowpart / 1024 / 1024 > 15 Then

' 文件超过 15MB 则自动放行
SubClassMessage = 915

' 向主界面窗体的防护记录中添加信息记录
With FormScan
.ListViewShieldLog.ListItems.Add , , .ListViewShieldLog.ListItems.Count + 1
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(1) = Date & " " & Time
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(2) = sFile
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(3) = "放行"
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(4) = "安全文件"

' 获取取被放行文件的图标
.ImageListListviewShieldLog.ListImages.Add , "Ico" & .ListViewShieldLog.ListItems.Count,
GetFileIcon(sFile, True)

' 设置 listview 行图标为此放行文件的图标
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SmallIcon
= .ImageListListviewShieldLog.ListImages.Item("Ico" & .ListViewShieldLog.ListItems.Count).Key
End With

' 退出函数
Exit Function
End If

' 主动防御规则记录文件
Dim sDefenseIniFile As String
If Right(App.Path, 1) = "\" Then
sDefenseIniFile = App.Path & "ActiveDefense.ini"
Else
sDefenseIniFile = App.Path & "\ActiveDefense.ini"
End If

' 软件设置记录文件
Dim sSettingsFile As String
If Right(App.Path, 1) = "\" Then
sSettingsFile = App.Path & "Settings.ini"
Else
sSettingsFile = App.Path & "\Settings.ini"
End If

```

```

' 读取防护提醒开启状态
Dim lEnableAlert As Long
lEnableAlert = ReadIni(sSettingsFile, "Shield", "EnableAlertMessage")

' 防护提醒窗体
Dim uAlertForm As New FormAlertMessage

' 获取主动防御规则数
Dim lMaxID As Long
lMaxID = ReadIni(sDefenseIniFile, "Count", "MaxID")

' 是否有主动防御规则记录
If lMaxID > 0 Then

' 规则对应文件
Dim sDefenseRuleFile As String

' 规则对应文件特征码
Dim sDefenseRuleFileSignature As String

' 规则标志, 1 为放行, 0 为禁止
Dim sDefenseRuleFileFlag As String
Dim i As Long
For i = 1 To lMaxID

' 文件
sDefenseRuleFile = ReadIni(sDefenseIniFile, "File", Format(i, "00000"))

' 特征码
sDefenseRuleFileSignature = ReadIni(sDefenseIniFile, "Signature", Format(i, "00000"))

' 标志
sDefenseRuleFileFlag = ReadIni(sDefenseIniFile, "DefenseFlag", Format(i, "00000"))

' 比对正在启动的进程是否与之批配
If (sDefenseRuleFileSignature = HashFile(sFile)) And (Len(sDefenseRuleFileSignature) = 32) And
(Len(HashFile(sFile)) = 32) Then
If CLng(sDefenseRuleFileFlag) = 1 Then

' 如果匹配则自动放行
SubClassMessage = 915

' 向主窗体防护记录中添加放行记录信息
With FormScan
.ListViewShieldLog.ListItems.Add , , .ListViewShieldLog.ListItems.Count + 1
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(1) = Date & " " & Time
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(2) = sFile
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(3) = "放行"
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(4) = "白名单文件"

```

```

' 获取被放行文件图标
.ImageListListViewShieldLog.ListImages.Add , "Ico" & .ListViewShieldLog.ListItems.Count,
GetFileIcon(sFile, True)

' 设置 listview 行图标
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SmallIcon
= .ImageListListViewShieldLog.ListImages.Item("Ico" & .ListViewShieldLog.ListItems.Count).Key
End With
If lEnableAlert = 1 Then
SetForegroundWindow FormScan.hWnd

' 在屏幕右下角弹出窗体提示用户
With uAlertForm

' 设置窗体位置
.Top = Screen.Height - .Height - 300
.Left = Screen.Width - .Width

' 显示内容
.LabelInfo.Caption = "白名单文件, " & sFile & ", 已放行。"

' 显示窗体
.Show 'vbModal
End With
End If

' 退出函数
Exit Function
ElseIf CLng(sDefenseRuleFileFlag) = 0 Then

' 如果标识为 0, 则自动阻止
SubClassMessage = 0

' 向主窗体界面中的防护记录中添加记录
With FormScan
.ListViewShieldLog.ListItems.Add , , .ListViewShieldLog.ListItems.Count + 1
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(1) = Date & " " & Time
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(2) = sFile
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(3) = "阻止"
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(4) = "黑名单文件"

' 取文件图标
.ImageListListViewShieldLog.ListImages.Add , "Ico" & .ListViewShieldLog.ListItems.Count,
GetFileIcon(sFile, True)

' 设置 listview 行图标
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SmallIcon
= .ImageListListViewShieldLog.ListImages.Item("Ico" & .ListViewShieldLog.ListItems.Count).Key
End With
SetForegroundWindow FormScan.hWnd

```

```

' 弹出窗体提示用户
With uAlertForm

' 设置窗体位置
.Top = Screen.Height - .Height - 300
.Left = Screen.Width - .Width

' 显示内容
.LabelInfo.Caption = "黑名单文件，" & sFile & "，已阻止。"

' 显示窗体
.Show
End With

' 退出函数
Exit Function
End If
End If
Next
End If

' 扫描文件，判断是否是病毒
Dim sScanResult As String
sScanResult = ScanFile(sFile)
If sScanResult <> "" Then

' 如果扫描结果不为空则说明检测到了病毒，这时向主窗体界面的防护记录中添加记录
With FormScan
.ListViewShieldLog.ListItems.Add , , .ListViewShieldLog.ListItems.Count + 1
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(1) = Date & " " & Time
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(2) = sFile
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(3) = "阻止"
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(4) = sScanResult

' 取文件图标
.ImageListListViewShieldLog.ListImages.Add , "Ico" & .ListViewShieldLog.ListItems.Count,
GetFileIcon(sFile, True)

' 设置 listview 行图标
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SmallIcon
= .ImageListListViewShieldLog.ListImages.Item("Ico" & .ListViewShieldLog.ListItems.Count).Key
End With
SetForegroundWindow FormScan.hWnd

' 弹出窗体提示用户
With uAlertForm

' 设置窗体位置
.Top = Screen.Height - .Height - 300
.Left = Screen.Width - .Width

```

```

' 显示内容
.LabelInfo.Caption = "发现病毒，" & sFile & "(" & sScanResult & ")" & "，已拦截。"

' 显示窗体
.Show
End With

' 上报给病毒监控服务器
Call ResponseVirusInfo("http://www.haiqi.cn/monitor/?virus=" & sScanResult)

' 退出函数
Exit Function
End If

' 读取云安全防护系统开启状态
Dim lCloudSecurty As Long
lCloudSecurty = ReadIni(sSettingsFile, "Shield", "CloudSecurty")
If lCloudSecurty = 1 Then

' 云安全查询
Dim sCloudAskRet As String

' 查询方式：URL + 文件 Hash 值
sCloudAskRet = CloudMatch("http://www.haiqi.cn/cloudsafe/ask.asp?Hash=" & HashFile(sFile))

' 返回 sCloudAskRet 格式为：1/0(放行或阻止标识) + $ + 病毒名或可信文件名
DoEvents

' 如果返回值不为空说明是病毒或可信文件
If sCloudAskRet <> "" Then
Dim lCloudFlag As Long
Dim sCloudName As String

' 放行或阻止标识：1 为放行，0 为阻止
lCloudFlag = Left(sCloudAskRet, InStr(1, sCloudAskRet, "$") - 1)

' 病毒或可信文件名称
sCloudName = Right(sCloudAskRet, Len(sCloudAskRet) - InStr(1, sCloudAskRet, "$"))

' 弹出窗体提示用户
Dim uCloudAlertForm As New FormAlertMessage

' 主动防御规则文件
Dim sLocalDefenseRuleFile As String
If Right(App.Path, 1) = "\" Then
sLocalDefenseRuleFile = App.Path & "ActiveDefense.ini"
Else
sLocalDefenseRuleFile = App.Path & "\ActiveDefense.ini"
End If

' 获取主动防御规则数

```

```

Dim lLocalMaxID As Long
lLocalMaxID = ReadIni(sLocalDefenseRuleFile, "Count", "MaxID")

' 阻止
If lCloudFlag = 0 Then

' 自动阻止
SubClassMessage = 0

' 添加记录
With FormScan
.ListViewShieldLog.ListItems.Add , , .ListViewShieldLog.ListItems.Count + 1
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(1) = Date & " " & Time
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(2) = sFile
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(3) = "阻止"
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(4) = sCloudName

' 取文件图标
.ImageListListviewShieldLog.ListImages.Add , "Ico" & .ListViewShieldLog.ListItems.Count,
GetFileIcon(sFile, True)

' 设置 listview 行图标
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SmallIcon
= .ImageListListviewShieldLog.ListImages.Item("Ico" & .ListViewShieldLog.ListItems.Count).Key
End With
With uCloudAlertForm

' 设置窗体位置
.Top = Screen.Height - .Height
.Left = Screen.Width - .Width

' 显示内容
.LabelInfo.Caption = "云安全防御系统发现病毒, " & sFile & "(" & sCloudName & ")" & ", 已拦截。"

' 显示窗体
.Show ' vbModal
End With

' 上报给病毒监控服务器
Call ResponseVirusInfo("http://www.haiqi.cn/monitor/?virus=" & HashFile(sFile))

' 向 ini 文件中写入主动防御文件内容
' 更新最大 ID 值
Call WriteIni(sLocalDefenseRuleFile, "Count", "MaxID", lLocalMaxID + 1)

' 写文件路径
Call WriteIni(sLocalDefenseRuleFile, "File", Format(lLocalMaxID + 1, "00000"), sFile)

' 写文件特征码(文件 hash 获得的 md5 值)
Call WriteIni(sLocalDefenseRuleFile, "Signature", Format(lLocalMaxID + 1, "00000"), HashFile(sFile))

```

```

' 文件描述
Call WriteIni(sLocalDefenseRuleFile, "Description", Format(lLocalMaxID + 1, "00000"), sCloudName)

' 防御标志(放行还是阻止)
Call WriteIni(sLocalDefenseRuleFile, "DefenseFlag", Format(lLocalMaxID + 1, "00000"), 0)

Else

' 反 DLL 返回放行标识
SubClassMessage = 915

' 向主窗体的防护记录中添加记录
With FormScan

' ListView 控件第一列设置为序号
.ListViewShieldLog.ListItems.Add , , .ListViewShieldLog.ListItems.Count + 1

' ListView 控件第二列显示记录时间
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(1) = Date & " " & Time

' ListView 控件第三列显示放行的文件名
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(2) = sFile
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(3) = "放行"
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(4) = sCloudName

' 取文件图标
.ImageListListviewShieldLog.ListImages.Add , "Ico" & .ListViewShieldLog.ListItems.Count,
GetFileIcon(sFile, True)

' 设置 listview 行图标
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SmallIcon
= .ImageListListviewShieldLog.ListImages.Item("Ico" & .ListViewShieldLog.ListItems.Count).Key
End With
If lEnableAlert = 1 Then
With uCloudAlertForm

' 设置窗体位置
.Top = Screen.Height - .Height
.Left = Screen.Width - .Width

' 显示内容
.LabelInfo.Caption = "云安全防御系统发现可信文件, " & sFile & "(" & sCloudName & ")" & ", 已放行。"

' 显示窗体
.Show ' vbModal
End With
End If

' 向更新最大 ID 值, 也就是有多少条记录
Call WriteIni(sLocalDefenseRuleFile, "Count", "MaxID", lMaxID + 1)

```



```

' 写入文件路径
Call WriteIni(sLocalDefenseRuleFile, "File", Format(lMaxID + 1, "00000"), sFile)

' 写入文件特征码(文件哈希操作获得的 Md5 值)
Call WriteIni(sLocalDefenseRuleFile, "Signature", Format(lMaxID + 1, "00000"), HashFile(sFile))

' 文件描述
Call WriteIni(sLocalDefenseRuleFile, "Description", Format(lMaxID + 1, "00000"), sCloudName)

' 防御标志(放行还是阻止)
Call WriteIni(sLocalDefenseRuleFile, "DefenseFlag", Format(lMaxID + 1, "00000"), 1)
End If
Exit Function
End If
End If

' 读取防护等级
Dim bHighLevel As Boolean
bHighLevel = ReadIni(sSettingsFile, "Shield", "HighLevel")

' 如果安全等级高, 拦截程序运行, 否则直接放行
If bHighLevel = True Then
SetForegroundWindow FormScan.hWnd

' 新建窗体显示启动进程等相关信息
lUserAction = 0
Dim uActiveDefenseForm As New FormActiveDefense
With uActiveDefenseForm

' 设置窗体位置
.Top = Screen.Height - .Height - 300
.Left = Screen.Width - .Width

' 设置窗体中要显示的内容
.LabelFile.Caption = sFile
.LabelSignature.Caption = HashFile(sFile)
.LabelCommand.Caption = sCommand
.ImageFileIcon = GetFileIcon(sFile, True)
.ImageFileIconShow = GetFileIcon(sFile, False)

' 显示窗体
.Show vbModal
End With
Do While lUserAction = 0
DoEvents
Sleep 10
Loop
If lUserAction = 1 Then
SubClassMessage = 915
Else
SubClassMessage = 0

```

```

End If
Else

' 自动放行标识, 会传递给 Dll
SubClassMessage = 915

' 向主窗体防护记录中添加记录
With FormScan
.ListViewShieldLog.ListItems.Add , , .ListViewShieldLog.ListItems.Count + 1
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(1) = Date & " " & Time
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(2) = sFile
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(3) = "放行"
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SubItems(4) = "防护级别: 正常, 未实施拦截"

' 获取文件图标
.ImageListListviewShieldLog.ListImages.Add , "Ico" & .ListViewShieldLog.ListItems.Count,
GetFileIcon(sFile, True)

' 设置 listview 行图标为刚刚获取的文件图标
.ListViewShieldLog.ListItems(.ListViewShieldLog.ListItems.Count).SmallIcon
= .ImageListListviewShieldLog.ListImages.Item("Ico" & .ListViewShieldLog.ListItems.Count).Key
End With
End If

' 退出函数
Exit Function
End If

' 判断是否是重建托盘图标消息, 如果是则重建托盘图标。此消息发生在这种情况下: 当系统因为意外错误导致任务
栏崩溃时, 托盘区的所有图标都会消息, 然后系统会重建任务栏并向所有程序发送任务栏创建消息:
TASKBARCREATED, 在这时我们接管了窗体的消息处理函数, 也收到此消息时, 这时就可以重新在刚刚重建的任务栏
托盘区增加自身图标, 否则系统重建的托盘区中不会存在我们软件自己的图标
If lMessage = WM_TASKBARCREATED Then
AddTrayIcon
End If

' 非 DLL 传来的数据, 执行正常的消息处理函数
SubClassMessage = CallWindowProc(lSubClassOldAddress, lHwnd, lMessage, lParentPID, lParam)
End Function

```

#### 5.4.16. 添加和移除托盘图标模块

模块:

ModuleTrayIcon

功能:

添加和移除托盘图标。

代码:

```

Option Explicit
'api 声明
Private Declare Function Shell_NotifyIcon Lib "shell32.dll" Alias "Shell_NotifyIconA" (ByVal dwMessage As Long, lpData As NOTIFYICONDATA) As Long
Private Declare Function LoadImage Lib "user32" Alias "LoadImageA" (ByVal hInst As Long, ByVal lpsz As String, ByVal un1 As Long, ByVal n1 As Long, ByVal n2 As Long, ByVal un2 As Long) As Long

' 自定义类型
Private Type NOTIFYICONDATA
    cbSize As Long
    hWnd As Long
    Uid As Long
    uFlags As Long
    UCallbackMessage As Long
    hIcon As Long
    SzTip As String * 64
End Type

' 常量定义
Private Const NIF_MESSAGE = &H1
Private Const NIF_TIP = &H4
Private Const WM_USER = &H400
Private Const TRAY_CALLBACK = (WM_USER + 1001&)
Private Const NIF_ICON = &H2

' 公共变量
Dim uTrayIcon As NOTIFYICONDATA

```

#### \* 添加托盘图标

```

Public Function AddTrayIcon()
    Dim lIcon As Long

    ' 使用 LoadImage 加载图标，可以使托盘区使用到真彩图标，远比使用程序自身图标漂亮许多
    lIcon = LoadImage(0&, App.Path & "\hq.ico", 1, 16, 16, &H10)

    ' 配置托盘信息
    With uTrayIcon
        .Uid = 0
        .hWnd = FormScan.hWnd
        .cbSize = Len(uTrayIcon)
        .hIcon = lIcon
        .uFlags = NIF_ICON
        .UCallbackMessage = TRAY_CALLBACK
        .uFlags = .uFlags Or NIF_MESSAGE Or NIF_TIP
        .cbSize = Len(uTrayIcon)
        .SzTip = "海奇杀毒软件" & Chr(0)
    End With

    ' 添加图标到系统托盘区

```

```
Shell_NotifyIcon 0, uTrayIcon
End Function
```

## ' 移除托盘图标

```
Public Function RemoveTrayIcon()
Shell_NotifyIcon 2, uTrayIcon
End Function
```

## 5.5. 控件文件说明及代码剖析

### 5.5.1. 软件及病毒库升级控件

控件:

UserControlUpdate

功能:

升级功能控件。

代码:

' API 声明

Option Explicit

```
Private Declare Function InternetOpen Lib "wininet" Alias "InternetOpenA" (ByVal sAgent As String, ByVal lAccessType As Long, ByVal sProxyName As String, ByVal sProxyBypass As String, ByVal lFlags As Long) As Long
```

```
Private Declare Function InternetCloseHandle Lib "wininet" (ByRef hInet As Long) As Long
```

```
Private Declare Function InternetReadFile Lib "wininet" (ByVal lFile As Long, pBuffer As Byte, ByVal lNumBytesToRead As Long, lNumberOfBytesRead As Long) As Integer
```

```
Private Declare Function InternetOpenUrl Lib "wininet" Alias "InternetOpenUrlA" (ByVal hInternetSession As Long, ByVal lpszUrl As String, ByVal lpszHeaders As String, ByVal dwHeadersLength As Long, ByVal dwFlags As Long, ByVal dwContext As Long) As Long
```

```
Private Declare Function HttpQueryInfo Lib "WinInet.dll" Alias "HttpQueryInfoA" (ByVal hHttpRequest As Long, ByVal lInfoLevel As Long, pBuffer As Any, ByRef lsBufferLength As Long, ByRef lIndex As Long) As Integer
```

```
Private Declare Function InternetSetFilePointer Lib "WinInet.dll" (ByVal lFile As Long, ByVal lDistanceToMove As Long, ByVal pReserved As Long, ByVal dwMoveMethod As Long, ByVal dwContext As Long) As Long
```

```
Private Declare Function DeleteFile Lib "kernel32" Alias "DeleteFileA" (ByVal lpFileName As String) As Long
```

```
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
```

' 常量定义

```
Private Const INTERNET_OPEN_TYPE_PRECONFIG = 0
```

```
Private Const scUserAgent = "Wing"
```

```
Private Const INTERNET_OPEN_TYPE_DIRECT = 1
```

```
Private Const INTERNET_OPEN_TYPE_PROXY = 3
```

```
Private Const INTERNET_FLAG_RELOAD = &H80000000
```

```

' 控件内公供变量
Private sUrl As String
Private sSaveFile As String

Private bConnect As Boolean
Private lOpen As Long
Private lFile As Long
Private sBuffer As String
Private lBuffer As Long
Private bRetQueryInfo As Boolean

' 控件公供事件，可在外部调用

' 下载进度
Public Event DownloadProcess(lProgress As Long)

' 下载完成事件
Public Event DownLoadFinish()

' 错误信息
Public Event ErrorMessage(sDescription As String)

```

## 初始化

```

Public Function Init() As Boolean
bConnect = True

```

' 该函数是第一个由应用程序调用的 WinINet 函数。它告诉 Internet DLL 初始化内部数据结构并准备接收应用程序之后的其他调用。

' 当应用程序结束使用 Internet 函数时，应调用 InternetCloseHandle 函数来释放与之相关的资源

```

lOpen = InternetOpen(scUserAgent, INTERNET_OPEN_TYPE_PRECONFIG, vbNullString, vbNullString, 0)

```

' 设置 bConnect 状态可以中止连接

```

If bConnect = False Then
CancelDownload

```

' 用户中止，函数返回失败

```

Init = False
Exit Function
End If
If lOpen = 0 Then
CancelDownload

```

' 创建网络连接失败

```

bConnect = False

```

' 函数返回失败

```

Init = False

```

```

' 引发错误事件
RaiseEvent ErrorMessage("创建网络连接失败。")
Exit Function
Else

' 通过一个完整的 FTP, Gopher 或 HTTP 网址打开一个资源。
lFile = InternetOpenUrl(lOpen, sUrl, vbNullString, 0, INTERNET_FLAG_RELOAD, 0)
If bConnect = False Then
CancelDownload

' 用户中止, 函数返回失败
Init = False
Exit Function
End If
If lFile = 0 Then
CancelDownload

' 无法连接到服务器。
bConnect = False

' 函数返回失败
Init = False

' 引发错误事件
RaiseEvent ErrorMessage("无法连接到升级服务器。")
Exit Function
Else
sBuffer = Space(1024)
lBuffer = 1024

' 从服务器获取 HTTP 请求头信息
bRetQueryInfo = HttpQueryInfo(lFile, 21, sBuffer, lBuffer, 0)

' 获取成功
If bRetQueryInfo Then
sBuffer = Mid(sBuffer, 1, lBuffer)

' 设置函数返回成功
Init = True
Exit Function
Else
sBuffer = ""

' 函数返回失败
Init = False
Exit Function
End If
End If
End If
End Function

```

## 开始升级

```
Public Function StartUpdate() As Boolean
Dim bBuffer(1 To 1024) As Byte
Dim lRet As Long

' 已下载大小
Dim lDownloadSize As Long
Dim i As Long

' 用户中止
If bConnect = False Then
CancelDownload
StartUpdate = False
Exit Function
End If

' 打开文件序号，如：#1
Dim lFileNumber As Long
lFileNumber = FreeFile()

' 下载保存文件方式：下载保存为 file.exe.bak 下载完后 删除 file.exe 重命名 file.exe.bak 为 file.exe
Dim sTempDownloadFile As String
sTempDownloadFile = Trim(sSaveFile) & ".bak"

' 如果文件已经存在，先删除
If Dir(sTempDownloadFile) <> "" Then
Call DeleteFile(sTempDownloadFile)
DoEvents
End If

' 打开文件，保存下载数据
Open sTempDownloadFile For Binary Access Write As #lFileNumber
Do

' 读取被下载文件内容
InternetReadFile lFile, bBuffer(1), 1024, lRet
DoEvents

' 完整块为 1024Byte
If lRet = 1024 Then
If bConnect = False Then
StartUpdate = False
Close #lFileNumber
GoTo UserCancel
End If

' 写入文件
Put #l, , bBuffer
```

```

Else

' 剩余部分块
For i = 1 To lRet
Put #l, , bBuffer(i)
DoEvents
Next i
End If

' 已经下载的字节数
lDownloadSize = lDownloadSize + lRet

' 引发下载进度事件
RaiseEvent DownloadProcess(lDownloadSize)
Loop Until lRet < 1024
Close #lFileNumber

' 检查已下载字节数与要下载文件大小是否一至，一至说明下载完成了。
If lDownloadSize = CLng(GetHeader("Content-Length")) Then

' 下载完成，删除原文件，重命名下载文件为原文件名
If Dir(Trim(sSaveFile)) <> "" Then
If Dir(Trim(sSaveFile)) <> "" Then
Call DeleteFile(sSaveFile)
End If
DoEvents
End If
Sleep 50
DoEvents
If Dir(sSaveFile) = "" Then

' 重命名下载的文件
Name sTempDownloadFile As sSaveFile
End If
DoEvents
End If

' 下载完成, 引发下载完成事件
RaiseEvent DownloadFinish

' 用户操作，正常中止
UserCancel:
CancelDownload
Exit Function

' 出错退出
ErrorExit:

' 引发错误事件
RaiseEvent ErrorMessage("升级过程发生意外错误。")
CancelDownload

```



```
Close #lFileNumber
End Function
```

#### **用户中止升级过程**

```
Public Function CancelDownload()

' 通过设置中止标志，当程序中其它地方检测到此标志时，会停止下载行为
bConnect = False
InternetCloseHandle lOpen
InternetCloseHandle lFile
End Function
```

#### **获取要下载文件的信息**

```
Public Function GetHeader(Optional hdrName As String) As String
Dim sTemp As Long
Dim sTemp2 As String

' 检测中止下载标志，如果标志为 False，则停止下载行为
If bConnect = False Then
GetHeader = "0"
CancelDownload
Exit Function
End If
If sBuffer <> "" Then
Select Case UCase(hdrName)

' 文件大小
Case "CONTENT-LENGTH"
sTemp = InStr(sBuffer, "Content-Length")
sTemp2 = Mid(sBuffer, sTemp + 16, Len(sBuffer))
sTemp = InStr(sTemp2, Chr(0))
GetHeader = CStr(Mid(sTemp2, 1, sTemp - 1))
Case "CONTENT-TYPE"
sTemp = InStr(sBuffer, "Content-Type")
sTemp2 = Mid(sBuffer, sTemp + 14, Len(sBuffer))
sTemp = InStr(sTemp2, Chr(0))
GetHeader = CStr(Mid(sTemp2, 1, sTemp - 1))
Case "DATE"
sTemp = InStr(sBuffer, "Date")
sTemp2 = Mid(sBuffer, sTemp + 6, Len(sBuffer))
sTemp = InStr(sTemp2, Chr(0))
GetHeader = CStr(Mid(sTemp2, 1, sTemp - 1))

' 最后修改时间
Case "LAST-MODIFIED"
sTemp = InStr(sBuffer, "Last-Modified")
sTemp2 = Mid(sBuffer, sTemp + 15, Len(sBuffer))
sTemp = InStr(sTemp2, Chr(0))
```

```

GetHeader = CStr(Mid(sTemp2, 1, sTemp - 1))
Case "SERVER"
sTemp = InStr(sBuffer, "Server")
sTemp2 = Mid(sBuffer, sTemp + 8, Len(sBuffer))
sTemp = InStr(sTemp2, Chr(0))
GetHeader = CStr(Mid(sTemp2, 1, sTemp - 1))
Case vbNullString
GetHeader = sBuffer
Case Else
GetHeader = "0"
End Select
Else
GetHeader = "0"
End If
End Function

```

#### 控件设置属性,保存文件名

```

Public Property Let SaveFile(ByVal sInFileName As String)
sSaveFile = sInFileName
End Property

```

#### 控件设置属性, URL 地址

```

Public Property Let URL(ByVal sInUrl As String)
sUrl = sInUrl
End Property

```

#### 控件窗体大小变化

```

Private Sub UserControl_Resize()

```

’ 设置控件大小为控件中图标的大小, ImageLogo 是 Image 控件, 放置 32x32 的图片, 做为控件的图标, 当控件被调用时, 拖放到窗体中的控件显示为此图标。

```

With UserControl
.Width = ImageLogo.Width
.Height = ImageLogo.Height
End With
End Sub

```

## 5.6. DLL 模块说明及代码剖析

### 5.6.1. 主动防护 DLL 模块功能说明及代码剖析

主动防御Dll模块名称为: ActiveDefense.dll, 使用API Hook技术, 拦截函数, 实现软件启动拦截。

完整DLL代码如下：

```
#include <windows.h>
#include "stdio.h"
#include "stdlib.h"

HANDLE hProcess=0;
UCHAR OldCode[5]={0}, NewCode[5]={0};
ULONG FunAddr=0;
HWND hExe=0;
HINSTANCE hMod=0;
HHOOK hHook=0;

//设置Api HOOK状态
BOOL HookStatus(BOOL Status)
{
    __try
    {
        BOOL ret=FALSE;

        //Status变量为真时，开启API HOOK
        if (Status)
        {
            //将CreateProcess函数入口处5字节内容写为Jump到CreateProcessWCallBack地址
            ret = WriteProcessMemory(hProcess, (void *)FunAddr, NewCode, 5, 0);
            if (ret) return TRUE;
        }
        else
        {
            //恢复CreateProcess函数入口处5字节代码
            ret = WriteProcessMemory(hProcess, (void *)FunAddr, OldCode, 5, 0);
            if (ret) return TRUE;
        }
        return FALSE;
    }
    __except(1)
    {
        return FALSE;
    }
}

//被HOOK函数CreateProcess的替换函数CreateProcessWCallBack，与CreateProcess有相同的参数
BOOL WINAPI CreateProcessWCallBack(LPCWSTR lpApplicationName,
                                   LPWSTR lpCommandLine,
                                   LPSECURITY_ATTRIBUTES lpProcessAttributes,
                                   LPSECURITY_ATTRIBUTES lpThreadAttributes,
                                   BOOL bInheritHandles,
                                   DWORD dwCreationFlags,
                                   LPVOID lpEnvironment,
                                   LPCWSTR lpCurrentDirectory,
                                   LPSTARTUPINFO lpStartupInfo,
                                   LPPROCESS_INFORMATION lpProcessInformation)
{
    __try
    {
```

```

BOOL b=FALSE;

//获取要启动的进程名
char AppName[256]={0};
WideCharToMultiByte(CP_ACP, 0,(const unsigned short *)lpApplicationName, -1, AppName, 256,NULL,
NULL);

//获取要启动的进程的命令行参数
char CommandLine[256]={0};
WideCharToMultiByte(CP_ACP, 0,(const unsigned short *)lpCommandLine, -1, CommandLine,
256,NULL, NULL);

char* CopyData="";

//连接进程名和命令行参数字符串，以便一起发送给上层VB程序
strcpy(CopyData,AppName);

//以符号“$”连接进程名和进程的命令行参数
strcat(CopyData,"$");
strcat(CopyData,CommandLine);
strcat(CopyData,"$");

COPYDATASTRUCT cds={0};
cds.lpData = CopyData;
cds.cbData = 1024;

//取得dll所在进程pid
DWORD dwProcessId;
dwProcessId=GetCurrentProcessId();

//发送数据给上层VB程序，发送的目标是在DLL被加载时，用FindWindow函数获取上层VB程序窗口
标题对应的程序窗体句柄
DWORD ret;
ret=SendMessage(hExe, WM_COPYDATA, dwProcessId, (LPARAM)&cds);

//在Win32中，WM_COPYDATA消息主要目的是允许在进程间传递只读数据。SDK文档推荐用户使用
SendMessage()函数，接收方在数据复制完成前不返回，这样发送方就不可能删除和修改数据。

//使用SendMessage发送消息给上层VB程序后，会一直等待返回值，这是由于SendMessage()是阻塞的，
只有接收方响应了消息，SendMessage()才能返回，否则一直阻塞。也正是因为如此，才能被我们使用：等待
VB程序返回数据。在本程序中我们约定，如果返回915时说明正启动的文件被放行了，否则就是要阻止。

if(ret==915)
{
    //恢复API HOOK
    HookStatus(FALSE);

    //调用真正的CreateProcessW函数执行正在启动的程序
    b = CreateProcessW(lpApplicationName,
        lpCommandLine,
        lpProcessAttributes,
        lpThreadAttributes,
        bInheritHandles,
        dwCreationFlags,
        lpEnvironment,
        lpCurrentDirectory,

```

```
lpStartupInfo,  
lpProcessInformation);
```

//上面刚刚恢复API HOOK，在执行完正要启动的程序后，再马上恢复HOOK，以便使我们的API HOOK功能持续生效

```
HookStatus(TRUE);  
return b;
```

```
}  
else  
{
```

//如果返回值不是915，则要阻止程序启动，只需简单的返回FALSE即可。这样程序便不会启动：启动失败

```
return FALSE;
```

```
}
```

```
return FALSE;
```

```
}
```

```
__except(1)
```

```
{
```

```
return FALSE;
```

```
}
```

```
}
```

//对CreateProcessW函数进行API HOOK

```
BOOL HookCreateProcess(){
```

```
__try  
{
```

```
ULONG JumpAddr=0;
```

//获取CreateProcessW函数地址

```
FunAddr = (ULONG)GetProcAddress(LoadLibrary("Kernel32.dll"), "CreateProcessW");
```

//保存CreateProcessW函数入口的五字节数据

```
memcpy(OldCode, (void *)FunAddr, 5);
```

//构建新的入口点代码

//0xe9是JMP指令

//JumpAddr2是CreateProcessWCallBack的JMP地址

```
NewCode[0] = 0xe9;
```

```
JumpAddr = (ULONG)CreateProcessWCallBack - FunAddr - 5;
```

```
memcpy(&NewCode[1], &JumpAddr, 4);
```

/开启API HOOK，在原理上是把CreateProcessW入口处五字节改为跳转到CreateProcessWCallBack，也就意味着当系统的CreateProcessW函数被调用时，会先跳转到我们的CreateProcessWCallBack 函数

```
HookStatus(TRUE);  
return TRUE;
```

```
}
```

```
__except(1)
```

```
{
```

```
return FALSE;
```

```
}
```

```
}
```

//DLL入口函数

```

BOOL APIENTRY DllMain( HANDLE hModule, DWORD    ul_reason_for_call, LPVOID lpReserved)
{
    hMod = (HINSTANCE)hModule;
    if (ul_reason_for_call==DLL_PROCESS_ATTACH)
    {
        //dll加载时执行这里
        __try
        {

            hProcess = OpenProcess(PROCESS_ALL_ACCESS, 0, GetCurrentProcessId());

            //取上层VB程序标题，以便发送消息
            hExe = FindWindow(NULL, "海奇杀毒软件");

            //挂钩createprocess函数
            HookCreateProcess();

        }
        __except(1)
        {
        }
    }

    //DLL卸载时执行这里
    if (ul_reason_for_call==DLL_PROCESS_DETACH)
    {
        __try
        {
            //程序终止时退出挂钩
            HookStatus(FALSE);

        }
        __except(1)
        {
        }
    }

    return TRUE;
}

LRESULT CALLBACK HookProc(int nCode, WPARAM wParam, LPARAM lParam){
    return(CallNextHookEx(hHook,nCode,wParam,lParam));
}

//此函数提供给上层VB程序调用，用于关闭主动防御
BOOL WINAPI ActiveDefenseOFF()
{
    return(UnhookWindowsHookEx(hHook));
}

//此函数提供给上层VB程序调用，用于开启主动防御
BOOL WINAPI ActiveDefenseON(){

    //向各进程注入dll，开启API HOOK
    hHook = SetWindowsHookEx(WH_GETMESSAGE,(HOOKPROC)HookProc, hMod, 0);
    if (hHook)
    {
        return TRUE;
    }
}

```

```

    }
    else
    {
        return FALSE;
    }
}

```

### 5.6.2. 自我保护 DLL 模块功能说明及代码剖析

自我保护DLL模块名称为：SafeGuard.dll，使用API Hook技术，拦截函数，实现软件启动拦截。

完整DLL代码如下：

```

#include <windows.h>
#include "stdio.h"
#include "stdlib.h"

//设置共享数据段，值里面的变量可以从上层VB程序中传递
#pragma data_seg(".Shared2")
    static DWORD g_PID2Protect=0;
#pragma data_seg()
#pragma comment(linker, "/section:.Shared2,rws")

HANDLE hProcess=0;

//要Hook的API的入口处的5字节
UCHAR OldCode2[5]={0}

//用于替换被Hook的API的入口处的5字节
UCHAR NewCode2[5]={0};

ULONG FunAddr2=0;
HINSTANCE hMod=0;
HHOOK hHook=0;

//控制Hook状态是否开启
BOOL HookStatus2(BOOL Status){
    BOOL ret2=FALSE;
    if (Status) {

        //向被HOOK函数入口写入修改过的代码
        ret2 = WriteProcessMemory(hProcess, (void *)FunAddr2, NewCode2, 5, 0);
        if (ret2) return TRUE;}
    else {

        //向被Hook函数的入口写入原始数据，也就是恢复Hook
        ret2 = WriteProcessMemory(hProcess, (void *)FunAddr2, OldCode2, 5, 0);
        if (ret2) return TRUE;}
    return FALSE;
}

```

```

//被Hook函数的替换函数，在本DLL中Hook的是OpenProcess，因此这里是它的替换函数
HANDLE WINAPI OpenProcessCallBack(DWORD dwDesiredAccess,BOOL bInheritHandle,DWORD dwProcessId)
{
    HANDLE hProc=NULL;

    //g_PID2Protect变量中保存的是要保护的进程ID，这里判断要打开的进程ID是否是此值，并且是否含有
    //PROCESS_TERMINATE标识。如果条件成立，则将返回值hProc设置为NULL，也就是使OpenProcess函数调用函数失败，
    //这样调用此函数企图打开进程并进行结束的操作将失败
    if(dwProcessId==g_PID2Protect && (dwDesiredAccess & PROCESS_TERMINATE!=0))
    {
        hProc=NULL;
    }else{

        //还原OpenProcess函数的原始入口数据
        HookStatus2(FALSE);

        //调用原始OpenProcess函数
        hProc= OpenProcess(dwDesiredAccess,bInheritHandle,dwProcessId);

        //调用完成后，立刻再进行HOOK
        HookStatus2(TRUE);
    }
    return hProc;
}

//对OpenProcess函数进行Hook
BOOL HookOpenProcess(){

    ULONG JmpAddr2=0;

    //获取OpenProcess函数的地址
    FunAddr2 = (ULONG)GetProcAddress(LoadLibrary("Kernel32.dll"), "OpenProcess");

    //保存OpenProcess函数入口的五字节数据
    memcpy(OldCode2, (void *)FunAddr2, 5);

    //构建新的入口点代码
    //0xe9是JMP指令
    //JmpAddr2是OpenProcessCallBack的JMP地址
    NewCode2[0] = 0xe9;
    JmpAddr2 = (ULONG)OpenProcessCallBack - FunAddr2 - 5;

    memcpy(&NewCode2[1], &JmpAddr2, 4);

    //开启API HOOK，在原理上是
    //将OpenProcess入口处五字节改为跳转到OpenProcessCallBack，也就意味着当
    //系统的OpenProcess函数被调用时，会先跳转到我们的OpenProcessCallBack函数，在我们的OpenProcessCallBack函数
    //中，我们可以随心所欲的做任何事情，这也就是API HOOK的本质
    HookStatus2(TRUE);
    return TRUE;
}

//DLL的入口函数
BOOL APIENTRY DllMain( HANDLE hModule,
                      DWORD  ul_reason_for_call,
                      LPVOID lpReserved)

```



```

{
    hMod = (HINSTANCE)hModule;

    //当DLL被加载时执行此处
    if (ul_reason_for_call==DLL_PROCESS_ATTACH)
    {
        hProcess = OpenProcess(PROCESS_ALL_ACCESS, 0, GetCurrentProcessId());

        //开启API HOOK
        HookOpenProcess();
    }

    //当DLL被卸载时执行此处
    if (ul_reason_for_call==DLL_PROCESS_DETACH)
    {
        //停止API HOOK
        HookStatus2(FALSE);
    }

    return TRUE;
}

LRESULT CALLBACK HookProc(int nCode,WPARAM wParam,LPARAM lParam){
    return(CallNextHookEx(hHook,nCode,wParam,lParam));
}

//此函数提供给上层VB程序调用，停止API HOOK
BOOL WINAPI SafeGuardOFF(){
    return(UnhookWindowsHookEx(hHook));
}

//此函数提供给上层VB程序调用，开启API HOOK，参数是进程PID
BOOL WINAPI SafeGuardON(DWORD uPID){

    g_PID2Protect=uPID;
    hHook = SetWindowsHookEx(WH_GETMESSAGE,(HOOKPROC)HookProc, hMod, 0);
    if (hHook)
    {
        return TRUE;
    }else{
        return FALSE;
    }
}

```

## 6. 感染型病毒的杀毒方案与编程实现详解

前面部分介绍的是历古以来的主流特征码匹配式杀毒方法，也是有史以来直到目前为止、乃至可预见的未来都会被广泛使用的基础杀毒方法。然而病毒并不会拘泥于一种方式不变，静静的等着杀毒软件来收割，它也在不断发展，每一段时期，都会有相较流行的病毒手段出现，从 DOS\Win32 时代流行的 PE 感染型病毒；到 2000 年流行的脚本病毒爱虫、冲击波病毒；到近年来花样纷呈的 U 盘病毒、ROOTKIT 病毒等，单就种类而言就令人不胜胜数，而杀毒软件在应些这些不同种类病毒的方式上，单纯存用特征码匹配的检测和防护，有时会略显不足，有时甚至会

无法下手，比如：感染型病毒是无法使用我们上面介绍的方法进行检测和清除的。那么如果我们遇到感染型病毒该如何下手？本章接下来将向您展示。

首先，我们要“遇到”一个感染型病毒，以便能够分析它、了解它的工作原理、了解它的各种属性，进而才能有对应的查杀思路。然而，这是非常困难的，因为现实中无乎得不到一份真正的病毒源码、原始样本。那么我们只好自己动手丰衣足食：先来写一个感染型的“病毒”，之所以病毒这个词打上双引号，因为并不是要真正写一个病毒，那是国家法律法规所不允许的，而是要写一个模仿病毒功能的测试，并且确保此程序无任何病毒行为、不会造成任何危害。

## 6.1. “感染型病毒”源代码及功能说明：

此程序将运行时，可以手动操作“感染”一个正常的 EXE 文件，为文件新增一个节(Section)，在节中新增加一块代码，新增加代码被运行时系统会发出“哔”的一声，并修改文件的入口地址，指向新增的代码，使程序启动时首先执行我们新增的代码，然后跳转回原始入口地址执行正常的功能。被“感染”后的程序在运行时，

```
//包含必要的头文件
```

```
#include <windows.h>
```

```
#include <winnt.h>
```

```
#include <stdio.h>
```

```
#include <assert.h>
```

```
//常量定义
```

```
#define DEBUG 1
```

```
#define EXTRA_CODE_LENGTH 18
```

```
//增加的节的大小
```

```
#define SECTION_SIZE 0x1000
```

```
//增加的节的名字，设置为：.test
```

```
#define SECTION_NAME ".test"
```

```
//文件名最大长度(包括路径)
```

```
#define FILE_NAME_LENGTH 30
```

```
//数量对齐函数
```

```
int Align(int size, int ALIGN_BASE)
```

```
{
```

```
    int ret;
```

```
    int result;
```

```
    assert( 0 != ALIGN_BASE );
```

```
    result = size % ALIGN_BASE;
```

```
    //余数不为零，也就是没有整除
```

```
    if (0 != result)
```

```
    {
```

```
        ret = ((size / ALIGN_BASE) + 1) * ALIGN_BASE;
```

```

    }
    else
    {
        ret = size;
    }

    return ret;
}

```

//工具使用方法

```
void usage()
```

```

{
    printf("使用方法: \n");
    printf("virTest.exe FileName\n");
    printf("例如: \n");
    printf("virTest.exe test.exe\n");
}

```

//入口函数

```
int main(int argc, char *argv[])
```

```

{

    //DOS 头
    IMAGE_DOS_HEADER DosHeader;

    //NT 头
    IMAGE_NT_HEADERS NtHeader;

    //节头
    IMAGE_SECTION_HEADER SectionHeader;

    //要新增加的节的节头
    IMAGE_SECTION_HEADER newSectionHeader;

    //节的数量
    int numOfSections;

    FILE *pNewFile;
    int FILE_ALIGN_MENT;
    int SECTION_ALIGN_MENT;

    char srcFileName[FILE_NAME_LENGTH];
    char newFileName[FILE_NAME_LENGTH];

    int i;
    int extraLengthAfterAlign;

    //新入口点
    unsigned int newEP;
}

```

```

//原始入口点
unsigned int oldEP;

//汇编指令: jmp
BYTE jmp;

char *pExtra_data;
int extra_data_real_length;

//判断命令行参数是否为空
if (NULL == argv[1])
{
    puts("参数错误\n");
    usage();
    exit(0);
}

//原始文件名
strcpy(srcFileName, argv[1]);

//目标文件名
strcpy(newFileName, srcFileName);

//给目标文件名再加一个后缀, 比如原来是: Test.exe, 再增加一次.exe 后缀, 成为: Test.exe.exe, 以此区分与
//原文件
strcat(newFileName, ".exe");

//复制一份原始文件
if (!CopyFile(srcFileName, newFileName, FALSE))
{
    //提示错误
    puts("复制文件失败");
    exit(0);
}

//打开新文件
//打开方式为"rb+"
pNewFile = fopen(newFileName, "rb+");
if (NULL == pNewFile)
{
    puts("打开文件失败");
    exit(0);
}

//定位到文件开始位置
fseek(pNewFile, 0, SEEK_SET);

//读取 IMAGE_DOS_HEADER
fread(&DosHeader, sizeof(IMAGE_DOS_HEADER), 1, pNewFile);

//判断是否是 PE 文件
if (DosHeader.e_magic != IMAGE_DOS_SIGNATURE)
{
    puts("Not a valid PE file");
    exit(0);
}

```

```

    }

    //再定位到 PE 文件头，然后读取 IMAGE_NT_HEADERS
    fseek(pNewFile, DosHeader.e_lfanew, SEEK_SET);
    fread(&NtHeader, sizeof(IMAGE_NT_HEADERS), 1, pNewFile);
    if (NtHeader.Signature != IMAGE_NT_SIGNATURE)
    {
        puts("文件错误，非 PE 文件");
        exit(0);
    }

    //经过两次验证，到此处可以确认这是 PE 文件，接下来可以放心的进行感染操作了

    //此 PE 文件节的数量
    numOfSections = NtHeader.FileHeader.NumberOfSections;

    //文件对齐量
    FILE_ALIGN_MENT = NtHeader.OptionalHeader.FileAlignment;

    //节对齐量
    SECTION_ALIGN_MENT = NtHeader.OptionalHeader.SectionAlignment;

#ifdef DEBUG

    //打印出调试信息

    printf("FILE_ALIGN_MENT-> %x\n", FILE_ALIGN_MENT);
    printf("SECTION_ALIGN_MENT-> %x\n", SECTION_ALIGN_MENT);
#endif

    //保存原来的入口地址
    oldEP = NtHeader.OptionalHeader.AddressOfEntryPoint;

    //定位到最后一个 SectionHeader
    for (i = 0; i < numOfSections; i++)
    {
        fread(&SectionHeader, sizeof(IMAGE_SECTION_HEADER), 1, pNewFile);

#ifdef DEBUG

        //打印出调试信息

        printf("节的名字: %s\n", SectionHeader.Name);
#endif

    }

    //增加一个新节前的准备工作
    extraLengthAfterAlign = Align(EXTRA_CODE_LENGTH, FILE_ALIGN_MENT);

    //节的总数加一
    NtHeader.FileHeader.NumberOfSections++;

```

```

//先清零
memset(&newSectionHeader, 0, sizeof(IMAGE_SECTION_HEADER));

//修改清零后部分数据：节名
strncpy(newSectionHeader.Name, SECTION_NAME, strlen(SECTION_NAME));

//重新设置 VirtualAddress
newSectionHeader.VirtualAddress = Align(SectionHeader.VirtualAddress +
SectionHeader.Misc.VirtualSize,SECTION_ALIGN_MENT);

//重新设置 VirtualSize
newSectionHeader.Misc.VirtualSize = Align(extraLengthAfterAlign, SECTION_ALIGN_MENT);

//重新设置 PointerToRawData
newSectionHeader.PointerToRawData = Align(SectionHeader.PointerToRawData +
SectionHeader.SizeOfRawData,FILE_ALIGN_MENT);

//重新设置 SizeOfRawData
newSectionHeader.SizeOfRawData = Align(SECTION_SIZE, FILE_ALIGN_MENT);

//修改新节的属性为：可读、可写、可执行
newSectionHeader.Characteristics = 0xE0000020;

#if DEBUG

//打印出调试信息

printf("原始 SizeOfCode: %x\n", NtHeader.OptionalHeader.SizeOfCode);
printf("原始 SizeOfImage: %x\n", NtHeader.OptionalHeader.SizeOfImage);
#endif

//重新设置 NtHeader

//重新设置 SizeOfCode
NtHeader.OptionalHeader.SizeOfCode = Align(NtHeader.OptionalHeader.SizeOfCode + SECTION_SIZE,
FILE_ALIGN_MENT);

//重新设置 SizeOfImage
NtHeader.OptionalHeader.SizeOfImage = NtHeader.OptionalHeader.SizeOfImage + Align(SECTION_SIZE,
SECTION_ALIGN_MENT);

#if DEBUG

//打印出调试信息

printf("新的 SizeOfCode: %x\n", NtHeader.OptionalHeader.SizeOfCode);
printf("新的 SizeOfImage: %x\n", NtHeader.OptionalHeader.SizeOfImage);
#endif

//Set zero the Bound Import Directory header
NtHeader.OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT].VirtualAddress = 0;
NtHeader.OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT].Size = 0;

//跳转到文件开始位置

```

```

fseek(pNewFile, 0, SEEK_END);

//新入口地址设置为新节的虚拟地址
newEP = newSectionHeader.VirtualAddress;

#if DEBUG

//打印出调试信息

printf("原入口地址-> %x\n", oldEP);
printf("新入口地址-> %x\n", ftell(pNewFile));
#endif

//设置新的入口地址
NtHeader.OptionalHeader.AddressOfEntryPoint = newEP;

//定位到节表尾部
fseek(pNewFile, DosHeader.e_lfanew + sizeof(IMAGE_NT_HEADERS) + numSections *
sizeof(IMAGE_SECTION_HEADER), SEEK_SET);

#if DEBUG

//打印出调试信息

printf("重新节头前当前指针位置-> %x\n", ftell(pNewFile));
printf("插入新节名称: %s\n", newSectionHeader.Name);

#endif

//写入重新设置后的节头
fwrite(&newSectionHeader, sizeof(IMAGE_SECTION_HEADER), 1, pNewFile);

#if DEBUG

//打印出调试信息

printf("重新节头后当前指针位置-> %x\n", ftell(pNewFile));
#endif

fseek(pNewFile, DosHeader.e_lfanew, SEEK_SET);

#if DEBUG

//打印出调试信息

printf("重写 NtHeader 前当前指针位置-> %x\n", ftell(pNewFile));
printf("(IMAGE_NT_HEADERS)大小: %x\n", sizeof(IMAGE_NT_HEADERS));
#endif

//写入重新设置后的 PE 文件头(NT 头)
fwrite(&NtHeader, sizeof(IMAGE_NT_HEADERS), 1, pNewFile);

//定位到文件尾部
fseek(pNewFile, 0, SEEK_END);

```

```

#if DEBUG

    //打印出调试信息

    printf("文件结尾指针-> %x\n", ftell(pNewFile));
#endif

    //写入新节，这里先写入 0
    for (i=0; i<Align(SECTION_SIZE, FILE_ALIGN_MENT); i++)
    {
        fputc(0, pNewFile);
    }

    //定位到新节的开头
    fseek(pNewFile, newSectionHeader.PointerToRawData, SEEK_SET);

#if DEBUG

    //打印出调试信息

    printf("重写额外数据前指针-> %x\n", ftell(pNewFile));
#endif

    goto GetExtraData;

extra_data_start:

    _asm pushad

    //获取 kernel32.dll 的基址
    _asm    mov eax, fs:0x30          ;PEB 的地址
    _asm    mov eax, [eax + 0x0c]
    _asm    mov esi, [eax + 0x1c]
    _asm    lodsd
    _asm    mov eax, [eax + 0x08] ;eax 就是 kernel32.dll 的基址

    //同时保存 kernel32.dll 的基址到 edi
    _asm    mov edi, eax

    //通过搜索 kernel32.dll 的导出表查找 GetProcAddress 函数的地址
    _asm    mov ebp, eax
    _asm    mov eax, [ebp + 3ch]
    _asm    mov edx, [ebp + eax + 78h]
    _asm    add edx, ebp
    _asm    mov ecx, [edx + 18h]
    _asm    mov ebx, [edx + 20h]
    _asm    add ebx, ebp

search:
    _asm    dec ecx
    _asm    mov esi, [ebx + ecx * 4]

    _asm    add esi, ebp
    _asm    mov eax, 0x50746547

```



```

//比较"PteG"
_asm    cmp [esi], eax
_asm    jne search
_asm    mov eax, 0x41636f72
_asm    cmp [esi + 4], eax
_asm    jne search
_asm    mov ebx, [edx + 24h]
_asm    add ebx, ebp
_asm    mov cx, [ebx + ecx * 2]
_asm    mov ebx, [edx + 1ch]
_asm    add ebx, ebp
_asm    mov eax, [ebx + ecx * 4]

//eax 保存的就是 GetProcAddress 的地址
_asm    add eax, ebp

//为局部变量分配空间
_asm    push ebp
_asm    sub esp, 50h
_asm    mov ebp, esp

//查找 beep 的地址

//把 GetProcAddress 的地址保存到 ebp + 40 中
_asm    mov [ebp + 40h], eax

//开始查找 Beep 的地址, 先构造"Beep\0"

//即'\0'
_asm    push 0x0

//准备压入"Beep"字符串, 也就是要让这段“感染”代码执行 Beep 函数
//字符串和这里使用的 16 进制串的关系可以通过字符与 Ascii 代码对照表获取,
//比如:B 的 Ascii 码是: 66, 转化为 16 进制是: 42, e 的 Ascii 码是: 101, 转化为 16 进制是: 65,p 的 Ascii
码值是 112, 转化为 16 进制是: 70
_asm    push DWORD PTR 0x70656542
_asm    push esp          //压入"Beep\0"的地址

//edi:kernel32 的基址
_asm    push edi

//返回值(即 Beep 函数的地址)保存在 eax 中
_asm    call [ebp + 40h]

//保存 Beep 的地址到 ebp + 44h
_asm    mov [ebp + 44h], eax

//压入 Beep 函数的两个参数, 一个表示声音频率, 一个表示发音时长
_asm    push 0x1b8
_asm    push 0x100

//调用 Beep 函数
_asm    call [ebp + 44h]

```

```

_asm mov esp, ebp
_asm add esp, 0x50
_asm popad

```

extra\_data\_end:

GetExtraData:

```

_asm pushad;
_asm lea eax, extra_data_start;
_asm mov pExtra_data, eax;
_asm lea edx, extra_data_end;
_asm sub edx, eax;
_asm mov extra_data_real_length, edx;
_asm popad;

```

//写入附加数据(用于测试“病毒”感染行为的数据)

```

for (i = 0; i < extra_data_real_length; i++)
{
    fputc(pExtra_data[i], pNewFile);
}

```

oldEP = oldEP - (newEP + extra\_data\_real\_length) - 5;

#if DEBUG

printf(原入口地址-> %x\n", oldEP);

#endif

//“病毒”代码结速，再跳回到原始入口地址继续执行

```

jmp = 0xE9;
fwrite(&jmp, sizeof(jmp), 1, pNewFile);
fwrite(&oldEP, sizeof(oldEP), 1, pNewFile);

```

fclose(pNewFile);

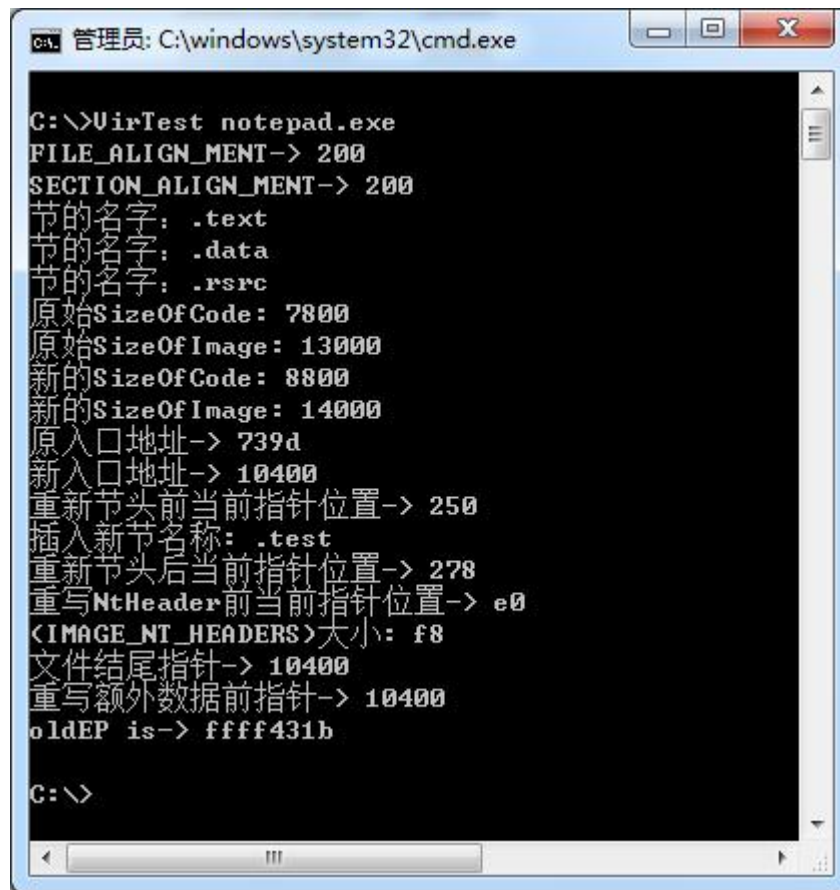
return 0;

}

## 6.2. “感染型病毒”运行测试及“感染文件”演示

从上面的代码中，已经可以看到“病毒”代码，也可以理解了它的感染理论。实际上，它是否能达到我们预期的效果，以能达到分析的价值呢，且看试验效果：

我们以系统的记事本文件 notepad.exe 做为试验对象。运行“病毒”并感染文件：



```
C:\>UirTest notepad.exe
FILE_ALIGN_MENT-> 200
SECTION_ALIGN_MENT-> 200
节的名字: .text
节的名字: .data
节的名字: .rsrc
原始SizeOfCode: 7800
原始SizeOfImage: 13000
新的SizeOfCode: 8800
新的SizeOfImage: 14000
原入口地址-> 739d
新入口地址-> 10400
重新节头前当前指针位置-> 250
插入新节名称: .test
重新节头后当前指针位置-> 278
重写NtHeader前当前指针位置-> e0
<IMAGE_NT_HEADERS>大小: f8
文件结尾指针-> 10400
重写额外数据前指针-> 10400
oldEP is-> ffff431b

C:\>
```

可以看到“感染”已经顺利完成，并且打印出了重要的调试信息。比如：插入的节名、原始入口地址、新入口地址。在运行感染后的文件时，也如我们期望的一般，在打开时，系统发出声音并开启时间比原始文件有一定延长。

有了上面图中这些信息，就可以做此“病毒”的清除工作了。也许有人还觉的疑惑，倒底要做什么呢？我们来理一理头绪：上面病毒感染了系统文件，在感染的过程中它插入了一段新的代码、修改了程序的入口点、执行完它插入的代码后，跳回到原入口点。那么清除它的思路也就清晰了：还原程序入口点，清除插入的代码。简单的说来，就是这么简单。

## 6.3. 编程清除“感染型病毒”

本节将介绍如果编程清除上面介绍的“感染型病毒”，还原被感染的文件。在本书前面的章节中，对 PE 文件的格式做过介绍，并且在编码中也做过演示。使用 PE 格式，可以简单明了的定位到程序的入口点、节等位置并进行操作，但在这里，我们将使用另一种稍有晦涩却也更有技巧的编程方式，实现清除此“感染型病毒”。

代码如下：

```
Public Function killInfection(file As String) As Integer
```

```
    On Error GoTo ErrExit
    Dim i As Long
    Dim MZ(2) As Byte, PeOffset As Long, Number As Long, L(4) As Byte, Low As Byte, High As Byte
    Dim t As Byte, StrHead As String
    Dim SizeOfImage As Long, Point As Long, SizeOfRaw As Long, Entry As Long
    Point = 0: SizeOfRaw = 0: Entry = 0: StrHead = "": SizeOfImage = 0
```

'以二进制方式打开文件  
Open file For Binary As #1

'读取文件第 1、2 字判断是否是可执行文件  
Get #1, 1, MZ(0)  
Get #1, 2, MZ(1)

If MZ(0) <> &H4D And MZ(1) <> &H5A Then  
    killinfection = NOT\_PE  
End If

'得到 PE 头偏移, 因为 PE 文件头格式都是固定的, 得到了这个 PE 头偏移, 就可以以此为基础, 获取其它文件头字段数据

Get #1, 61, PeOffset

'因为文件指针是从 1 开始所以要加上 1  
PeOffset = PeOffset + 1

Get #1, PeOffset, Low  
Get #1, PeOffset + 1, High

'PE 头开始处是 PE 标志, 上面获取的就是得到 PE 标志(Coff header)  
If Low <> &H50 And High <> &H45 Then  
    killinfection = NOT\_W32  
End If

'取得节数(Section 数), 保存节数的共 8 字节, 为了精准的获取数据, 采用分别获取高低八数据, 然后再拼合在一起的办法

'低位  
Get #1, PeOffset + 6, L(0)  
'高位  
Get #1, PeOffset + 7, L(1)  
Number = L(1) \* 256& + L(0)

'取得 SizeOfImage(Optional Header 中), 在 PE 头偏移处加 50H 的地方  
Get #1, PeOffset + &H50, L(0)  
Get #1, PeOffset + &H51, L(1)  
Get #1, PeOffset + &H52, L(2)  
Get #1, PeOffset + &H53, L(3)  
SizeOfImage = L(3) \* 256& \* 256& \* 256 + L(2) \* 256& \* 256& + L(1) \* 256& + L(0)

'减掉病毒增加的 H1000 字节  
SizeOfImage = SizeOfImage - &H1000

'这里开始是获取最后一个节(Section 名), 用 Get 方法读取文件内容时, 会将指针随着移动, 读取到最后一个节, 也就是移动到了最后一个节

For i = 0 To 7  
    Get #1, PeOffset + &H53 + &HA5 + &H28 \* (Number - 1) + i, t

    'get 读到的数据是数字, 用 Chr 将之转化为字母  
    StrHead = StrHead & Chr(t)  
Next i

'判断是否是病毒新增加的节: .test, 如果是, 说明此文件被感染了, 可以进行修改, 当然这里只是演示产

操作，判断依据不够充分，在实际的应用中，为了防止误操作可以再增加一些其它的判断条件，比如节的大小或其它的特征

```
If Left(StrHead, 5) = ".test" = 0 Then

    '重写 sizeofimage
    t = CByte(SizeOfImage Mod 256)

    Put #1, PeOffset + &H50, t

    SizeOfImage = Int(SizeOfImage / 256)
    t = CByte(SizeOfImage Mod 256)

    Put #1, PeOffset + &H51, t

    SizeOfImage = Int(SizeOfImage / 256)
    t = CByte(SizeOfImage Mod 256)
    Put #1, PeOffset + &H52, t
    SizeOfImage = Int(SizeOfImage / 256)
    t = CByte(SizeOfImage Mod 256)

    Put #1, PeOffset + &H53, t

    '重写程序入口点

    Entry = &H739D
    t = CByte(Entry Mod 256)
    Put #1, PeOffset + 40, t

    Entry = Int(Entry / 256)
    t = CByte(Entry Mod 256)

    Put #1, PeOffset + 41, t
    Entry = Int(Entry / 256)

    t = CByte(Entry Mod 256)

    Put #1, PeOffset + 42, t
    Entry = Int(Entry / 256)

    t = CByte(Entry Mod (256&))
    Put #1, PeOffset + 43, t

    '写节数，比原来小 1
    t = CByte((Number Mod 256&) - 1)
    Put #1, PeOffset + 6, t
    t = CByte(Number / 256&)
    Put #1, PeOffset + 7, t

    killinfection = CLEANED
Else
    killinfection = CLEAN_FILE
End If

Close #1
Exit Function
```

ErrExit:

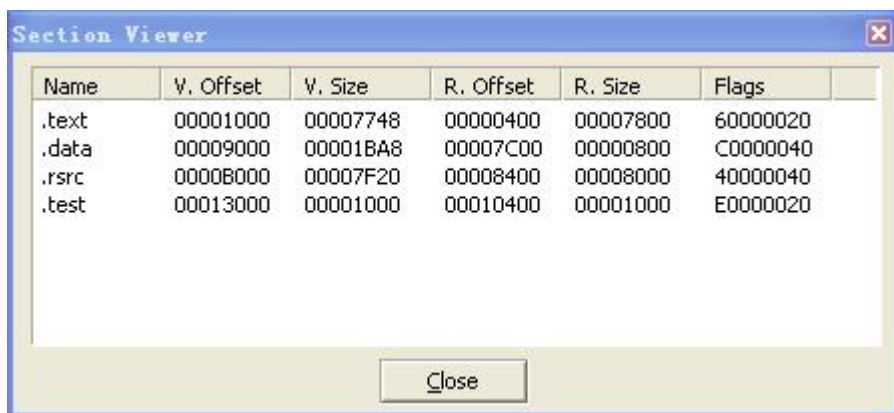
```
killKinfection= CLEAN_FILE
Close #1
```

End Function

这段代码中，使用了大量的数字，比如：Get #1, PeOffset + &H50, L(0)，在代码中已经做了具体的说明：在 PE 头偏移处加 50H 的地方存放着 SizeOfImage，这是有 PE 头结构据可依能计算得出且固定不会变的。

但另外一些，比如：减掉病毒增加的 H1000 字节、Entry = &H739D，这两个是非常重要的数据，这两个值看不出它的出处。虽然从“病毒”源码中可以得知 H1000 字节对应的根据是：#define SECTION\_SIZE 0x1000；但除了作者谁也不知道源码是这样写的。而 Entry = &H739D，是指入口点，同样的，此入口点值是由“病毒”运行时打印出的，真正的病毒，是不会有这样诚实的对白的。那么，我们实战中我们该如何获取这些信息呢？

就 H1000 字节的问题而言，1000H 字节，指的是“病毒”附加在宿主文件上内容容量大小，对于本“病毒”而言，它向被感染文件增加了一个节，新增的代码都放罢在此节中，那么新增加的节的大小，就是所增加内容容量的大小。这个节的大小值，可以通过很多 PE 辅助工具查看。例如下图：



Name	V. Offset	V. Size	R. Offset	R. Size	Flags
.text	00001000	00007748	00000400	00007800	60000020
.data	00009000	00001BA8	00007C00	00000800	C0000040
.rsrc	0000B000	00007F20	00008400	00008000	40000040
.test	00013000	00001000	00010400	00001000	E0000020

可以看到，.test 节的大小为 00001000（十六进制）

同理，再使用其它的逆向工具，可以很轻松的看到文件的反汇编代码，比如被感染后，Notepad.exe 的入口处代码如下：

```
01013000 > 60          pushad
01013001    64:A1 30000000  mov     eax, dword ptr fs:[30]
01013007    8B40 0C         mov     eax, dword ptr [eax+C]
0101300A    8B70 1C         mov     esi, dword ptr [eax+1C]
0101300D    AD             lods     dword ptr [esi]
0101300E    8B40 08         mov     eax, dword ptr [eax+8]
01013011    8BF8           mov     edi, eax
01013013    8BE8           mov     ebp, eax
01013015    8B45 3C         mov     eax, dword ptr [ebp+3C]
01013018    8B5405 78       mov     edx, dword ptr [ebp+eax+78]
0101301C    03D5           add     edx, ebp
0101301E    8B4A 18         mov     ecx, dword ptr [edx+18]
01013021    8B5A 20         mov     ebx, dword ptr [edx+20]
01013024    03DD           add     ebx, ebp
01013026    49             dec     ecx
01013027    8B348B         mov     esi, dword ptr [ebx+ecx*4]
0101302A    03F5           add     esi, ebp
0101302C    B8 47657450    mov     eax, 50746547
```

```

01013031    3906          cmp     dword ptr [esi], eax
01013033    ^ 75 F1       jnz     short 01013026
01013035    B8 726F6341   mov     eax, 41636F72
0101303A    3946 04        cmp     dword ptr [esi+4], eax
0101303D    ^ 75 E7       jnz     short 01013026
0101303F    8B5A 24        mov     ebx, dword ptr [edx+24]
01013042    03DD          add     ebx, ebp
01013044    66:8B0C4B      mov     cx, word ptr [ebx+ecx*2]
01013048    8B5A 1C        mov     ebx, dword ptr [edx+1C]
0101304B    03DD          add     ebx, ebp
0101304D    8B048B        mov     eax, dword ptr [ebx+ecx*4]
01013050    03C5          add     eax, ebp
01013052    55            push    ebp
01013053    83EC 50       sub     esp, 50
01013056    8BEC          mov     ebp, esp
01013058    8945 40       mov     dword ptr [ebp+40], eax
0101305B    6A 00         push    0
0101305D    68 42656570   push    70656542
01013062    54            push    esp
01013063    57            push    edi
01013064    FF55 40       call    dword ptr [ebp+40]
01013067    8945 44       mov     dword ptr [ebp+44], eax
0101306A    68 B8010000   push    1B8
0101306F    68 00010000   push    100
01013074    FF55 44       call    dword ptr [ebp+44]
01013077    8BE5          mov     esp, ebp
01013079    83C4 50       add     esp, 50
0101307C    61            popad
0101307D    - E9 1B43FFFF jmp     0100739D

```

01013000~0101307D 行的代码，都是我们“感染”到正常文件中的入口点代码：

```

0101306A    68 B8010000   push    1B8
0101306F    68 00010000   push    100
01013074    FF55 44       call    dword ptr [ebp+44]

```

这是我们调用的 Beep 函数：

最后一句 jmp 0100739D 是关键，它跳转到了真正的程序入口点，也就是 0100739D。为什么说 0100739D 处就是入口点呢，而不是一个中途的跳转？这是因为：在 0100739D 的地方，出现了正常程序入口点所拥有的特征：

```

0100739D: 6A70          PUSH    00000070H
0100739F: 6898180001    PUSH    01001898H
010073A4: E8BF010000    CALL    01007568H
010073A9: 33DB          XOR     EBX, EBX
010073AB: 53            PUSH    EBX
010073AC: 8B3DCC100001  MOV     EDI, [010010CCH]
010073B2: FFD7          CALL    EDI
010073B4: 6681384D5A    CMP     WORD PTR [EAX], 5A4DH
010073B9: 751F          JNZ     10073DAH
010073BB: 8B483C        MOV     ECX, [EAX+3CH]

```

010073BE: 03C8	ADD ECX, EAX
010073C0: 813950450000	CMP [ECX], 00004550H
010073C6: 7512	JNZ 10073DAH
010073C8: 0FB74118	MOVZX EAX, WORD PTR [ECX+18H]
010073CC: 3D0B010000	CMP EAX, 0000010BH
010073D1: 741F	JZ 10073F2H
010073D3: 3D0B020000	CMP EAX, 0000020BH
010073D8: 7405	JZ 10073DFH
010073DA: 895DE4	MOV [EBP-1CH], EBX
010073DD: EB27	JMP 1007406H
010073DF: 83B98400000000E	CMP [ECX+00000084H], 0000000EH
010073E6: 76F2	JBE 10073DAH
010073E8: 33C0	XOR EAX, EAX
010073EA: 3999F8000000	CMP [ECX+000000F8H], EBX
010073F0: EB0E	JMP 1007400H
010073F2: 8379740E	CMP [ECX+74H], 0000000EH
010073F6: 76E2	JBE 10073DAH
010073F8: 33C0	XOR EAX, EAX
010073FA: 3999E8000000	CMP [ECX+000000E8H], EBX
01007400: 0F95C0	SETNZ AL
01007403: 8945E4	MOV [EBP-1CH], EAX
01007406: 895DFC	MOV [EBP-04H], EBX
01007409: 6A02	PUSH 00000002H
0100740B: FF1538130001	CALL [01001338H] ; __set_app_type
01007411: 59	POP ECX
...	

这只是一个简单的例子，用来进行理论说明，实际的感染型病毒行为可能远比这复杂，但也都是在此基础上的扩充，比如：加密、反调试等等，具体的如何应对还需要各位读者触类旁通、对技术进行灵活应用。

## 7. 启发式杀毒技术原理与编程实现

启发式杀毒是相对于特征码检测杀毒而言的。特征码查杀相对而言是传统的病毒查杀方法，启发式杀毒是较为新兴的技术理论。

启发式杀毒技术又可称为病毒行为检测技术，根据病毒的行为实现对病毒的检测。

### 7.1. 启发式杀毒技术原理

根据不同的病毒行为，启发式查毒也有不同的理论切入点，比如：

（1）、感染型病毒可能会修改文件的入口点，要修改文件，就要使用相应的 API 组合才能完成，通过判断 PE 文件的导入函数组合以及 API 调用顺序，可以判断是否是病毒行为；

（2）、被感染修改后的入口点地址会出现一定的规律的调用方式（比如经典的在内存中搜索 GetProcess 字符



串)，通过判断这种病毒高危行为也可以做为识别病毒的一行方式：

(3)、非感染形病毒入侵系统后，常会向系统添加开机自启动项、添加系统服务、修改文件关联（常用于实现自我复活）等等，比如正常状态下默认.txt文件关联程序是与系统记事本程序关联，如果检测到此关联变为了其它的第三方应用程序，则此异常行为可能是病毒的行为，与此相关的文件则可能是病毒文件。

诸如此类的方式，都可以归为启发式杀毒技术的理论，由于此种理论的基本方式是通过各种组合进行判断操作，因而会有高于特征码检测较多的误报率，应对的办法通常又会引入黑白名单加以辅助。

理论说了这么多，接下来，再从编程的角度进行一些实例的编程方法演示。

## 7.2. 启发式杀毒编程实现

### 启发式扫描注册表 Explorer 入口点

'扫描注册表中 Explorer 入口点键值，判断是否被修改。

'注册表中 HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon 下的 Shell 项是一个注册表自启动项，填在此项中的文件，会在系统启动时被自动执行。默认情况下此项的值为：Explorer.exe。但如果被病毒利用，在其后面跟入病毒文件的路径，则系统在启动时会将病毒也启动。

'本函数，扫描并判断此值。

```
Public Function ScanExplorerShell()
```

```
Dim lKeyHandle As Long
```

```
Dim sTempKeyValue As String * 256
```

```
Dim sKeyValue As String
```

'打开注册表键值

```
Call RegCreateKey(HKEY_LOCAL_MACHINE, "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon",  
lKeyHandle)
```

'获取指定项目的值

```
Call RegQueryValueEx(lKeyHandle, "Shell", 0, REG_SZ, ByVal sTempKeyValue, 256)
```

```
Call RegCloseKey(lKeyHandle)
```

'获取项目的值

```
sKeyValue = Left(sTempKeyValue, InStr(1, sTempKeyValue, Chr(0)) - 1)
```

'去除正常的 Explorer.exe 长度，获取到的就是可能附加的数据

```
Dim sShelltrj As String
```

```
sShelltrj = Trim(Right(sKeyValue, Len(sKeyValue) - Len("Explorer.exe")))
```

'判断附加数据的长度是否大于 4，因为大于 4 才可能是文件

```
If Len(sShelltrj) > 4 Then
```

'这里可以使用白名单（可以带特征码），排除一些合法的文件，防止误报

'没有被排除的就识别为病毒，这部分代码省略

```
End If
```

```
End Function
```

通过扫描文件关联的合法性，判断病毒木马是否进入了系统

'参数 RegPath 是注册表路径，可以是：

'例 1：Applications\notepad.exe\shell\open\command

'例 2：exefile\shell\open\command

'例 3：txtfile\shell\open\command

'对应的 skey 参数可以是：

'例 1：Notepad

'例 2：ExeFile

'例 3：TxtFile

Public Function RealCheck(ByVal RegPath As String, sKey As String)

Dim lTxtKeyHandle As Long

Dim sTempTxtData As String \* 255

Dim sTxtData As String

'获取文件关联的注册表键值，比如位于：HKEY\_CLASSES\_ROOT\exefile\shell\open\command 中的 ExeFile 键值

Call RegCreateKey(HKEY\_CLASSES\_ROOT, RegPath, lTxtKeyHandle)

Call RegQueryValueEx(lTxtKeyHandle, "", 0, REG\_SZ, ByVal sTempTxtData, 255)

RegCloseKey lTxtKeyHandle

sTxtData = Left(sTempTxtData, InStr(1, sTempTxtData, Chr(0)) - 1)

'判断是否含有". "号，也就是是否含有程序被附加，因为程序都是型如：a.exe 的样子，都有“.”号

If InStr(1, sTxtData, ". ") Then

'获取程序名称

Dim sUnionTroy As String

If InStr(1, sTxtData, " ") Then

sUnionTroy = Left(sTxtData, InStr(1, sTxtData, " ") - 1)

End If

'这里使用白名单，排除一些合法的文件，防止误报

'没有被排除的就识别为病毒，即 sUnionTroy 变量中保存的文件名

End If

End Function

扫描注册表 AppInit\_DLLs，这是一个 DLL 自启动项

Public Function ScanAppInit\_DLLs()

Dim lRetVal As Long

Dim lKeyHandle As Long

'获取注册表中：HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows 下的

AppInit\_DLLs，判断是否有内容，默认情况下这里为空。如果这个值中被写入了 dll 文件，则此 dll 会被系统启动后自动加载。

lRetVal = RegOpenKey(HKEY\_LOCAL\_MACHINE, "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows", lKeyHandle)

```
Dim sTempKeyValue As String * 255
Dim sKeyValue As String
Call RegQueryValueEx(lKeyHandle, "AppInit_DLLs", 0, REG_SZ, ByVal sTempKeyValue, 255)
Call RegCloseKey(lKeyHandle)
```

```
sKeyValue = Left(sTempKeyValue, InStr(1, sTempKeyValue, Chr(0)) - 1)
```

```
'判断内容长度是否大于 4，大于 4 说明被写入了内容
```

```
If Len(sKeyValue) > 4 Then
```

```
'如果上面一般，这里同样使用白名单过滤一些合法的文件
```

```
End If
```

```
End Function
```

### **检测 IE BHO(Browser Helper Objects)**

BHO 会劫持 IE 行为，掌握一切通过 IE 浏览网页的操作，相当具有危险性。但同时 BHO 也是一项正常的插件，常被各种正常软件所使用，所以检测 BHO 时，需要先事收集识别大量的白名单文件。

```
Public Function ScanBHO()
```

```
Dim lHwnd As Long
```

```
Dim lRet As Long
```

```
'打开注册表子键：HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\explorer\Browser Helper Objects
```

```
lRet = RegOpenKey(HKEY_LOCAL_MACHINE, "Software\Microsoft\Windows\CurrentVersion\explorer\Browser Helper Objects", lHwnd)
```

```
Dim lIndex As Long
```

```
lIndex = 0
```

```
Dim sBuffer As String * 255
```

```
Dim sSubKey As String
```

```
'遍历所有的 BHO 项目
```

```
Do While lRet = 0
```

```
lRet = RegEnumKey(lHwnd, lIndex, sBuffer, 255)
```

```
If lRet = 0 Then
```

```
lIndex = lIndex + 1
```

```
'从 sBuffer 中取出有用的字符
```

```
sSubKey = Left(sBuffer, InStr(1, sBuffer, Chr(0)) - 1)
```

```
Dim lHwnd2 As Long
```

```
lHwnd2 = 0
```

```
Dim sbuffer2 As String * 255
```

```
'上面取到了 BHO 的 sid，下面开始取它对应的文件，是从 HKEY_CLASSES_ROOT\CLSID 子键下获取
```

```
Call RegOpenKey(HKEY_CLASSES_ROOT, "CLSID" & sSubKey & "\InprocServer32", lHwnd2)
```

```
Call RegQueryValueEx(lHwnd2, "", 0, REG_SZ, ByVal sbuffer2, 255)
```

```
RegCloseKey lHwnd2
```

```
Dim sBho As String
sBho = Trim(Left(sbuffer2, InStr(1, sbuffer2, Chr(0)) - 1))
```

'上面获取到了 BHO 的文件，保存在变量 sBho 中，接下来再使用白名单过滤存活文件，遗留下的就极有可能是病毒

```
End If
```

```
Loop
```

```
Call RegCloseKey(lHwnd)
End Function
```

### **检测 IE ShellExecuteHooks**

这同样是一种 IE 插件，在 IE 执行时被自动调用

```
Public Function ScanShellExecuteHooks()
```

```
Dim lRetVal As Long
```

'打开注册表子键：HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\explorer\ShellExecuteHooks

```
lRetVal = RegOpenKey(HKEY_LOCAL_MACHINE,
"Software\Microsoft\Windows\CurrentVersion\explorer\ShellExecuteHooks", hKey)
```

```
Dim iIndex As Long
```

'iIndex 是要枚举的序号

```
iIndex = 0
```

'注册表键名称

```
Dim sValueName As String * 255
```

'注册表键内容

```
Dim bValueData As Byte
```

'如果打开注册表主键成功才执行下面的代码

```
Do While lRetVal = 0
```

'开始枚举

```
lRetVal = RegEnumValue(hKey, iIndex, ByVal sValueName, 255, 0&, REG_SZ, ByVal bValueData, 255)
```

'枚举成功返回 0

```
If lRetVal = 0 Then
```

```
iIndex = iIndex + 1
```

'去掉字符串尾部不可见字符

```
Dim sSubKey As String
```

```
sSubKey = Left(sValueName, InStr(1, sValueName, Chr(0)) - 1)
```

```
Dim lHwnd2 As Long
```

```
lHwnd2 = 0
```

```
Dim sbuffer2 As String * 255
```

```
Call RegOpenKey(HKEY_CLASSES_ROOT, "CLSID\" & sSubKey & "\InprocServer32", lHwnd2)
```

```
Call RegQueryValueEx(IHwnd2, "", 0, REG_SZ, ByVal sbuffer2, 255)
```

```
RegCloseKey IHwnd2
```

```
Dim sExeShellHook As String
```

```
sExeShellHook = Left(sbuffer2, InStr(1, sbuffer2, Chr(0)) - 1)
```

'使用白名单过滤合活文件，遗留的就很可能是病毒

```
Loop
```

```
Call RegCloseKey(hKey)
```

```
End Function
```

### 检测注册表 **Userinit** 项

与上面讲过的 Explorer 入口点键值相似，也是一个自启动项目

```
Public Function ScanUserInit()
```

```
Dim IKeyHandle As Long
```

```
Dim ITempKeyValue As String * 255
```

```
Dim IKeyValue As String
```

'检测注册表键值：HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon 下的 Userinit

```
Call RegCreateKey(HKEY_LOCAL_MACHINE, "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon", IKeyHandle)
```

```
Call RegQueryValueEx(IKeyHandle, "Userinit", 0, REG_SZ, ByVal ITempKeyValue, 255)
```

```
Call RegCloseKey(IKeyHandle)
```

```
IKeyValue = Left(ITempKeyValue, InStr(1, ITempKeyValue, Chr(0)) - 1)
```

```
Dim sInitTrj As String
```

```
sInitTrj = Trim(Right(IKeyValue, Len(IKeyValue) - InStr(1, IKeyValue, ",")))
```

```
If InStr(1, sInitTrj, ".") Then
```

'使用白名单过滤合活文件，遗留的就可能是病毒

```
End If
```

```
End Function
```

### 检测注册表 **WindowsNtLoad** 项

与上面讲过的 Explorer 入口点键值相似，也是一个自启动项目

```
Public Function ScanWindowsNtLoad()
```

```
Dim IKeyHandle As Long
```

```
Dim ITempKeyValue As String * 255
```

```
Dim IKeyValue As String
```

'检测注册表键值：HKEY\_CURRENT\_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows 下的 Load 项

```
Call RegCreateKey(HKEY_CURRENT_USER, "Software\Microsoft\Windows NT\CurrentVersion\Windows", IKeyHandle)
```

```
Call RegQueryValueEx(IKeyHandle, "load", 0, REG_SZ, ByVal ITempKeyValue, 255)
```

```
Call RegCloseKey(IKeyHandle)
```

```
'这个 IKeyValue 可能是 notepad.exe,calc.exe 这种形式，比较复杂  
IKeyValue = Left(ITempKeyValue, InStr(1, ITempKeyValue, Chr(0)) - 1)
```

```
If IKeyValue <> "" Then
```

```
Dim sLoadTrj As String  
sLoadTrj = IKeyValue
```

```
'使用白名单过滤合法文件  
End Function
```

## 8. 后记

本书所讲述的技术理论是本人从业十年的经验总结之谈。从起初对杀毒技术感兴趣开始、从对杀毒技术一无所知开始、从在国内外网上到处找各种杀毒编程技术开始，一步步走开，经历了一个痛苦而漫长的过程。

网上根本找不到一点系统性的杀毒编程技术资料或理论，一切都需要自己摸索、猜想、试验、测试、验证想法、验证正确性，进而一点点的总结经验。

本人并不是身在大型杀毒软件公司，是以一个自由职业者的身份、以兴趣为驱动对杀毒技术进行的自我研究，因此，可能本书所讲的理论知识也许并不够全面，甚至可能有错误。希望各种读者在发现疏漏之处能够不吝批评赐教。

本人在多年的安全编程生涯中，曾开发出《反木马病毒专家》、《HK AntiSpyware》、《Spyware Nurse》、《海奇杀毒软件》等多款反木马反病毒产品，并参与过中国电信的《宽带卫士》开发。自认为所掌握的相关编程知识可以引领各位有兴趣于此领域的读者入门杀毒软件开发，这是本书形成的基础。本人在经历了多年的个人安全软件开发后，由于工作目标方向变化的原因，将不再继续此方面开发，而此时在市面和网络中仍然没有相关的书籍资料出现，杀毒软件开发技术领域书籍仍是一个空白，于是萌发了一个念头：将我的经验写做成书、公布于众、成为此领域第一本书、将我的“手艺”传承开来，使不失传，也算为全人类某一项福利，这便是本书的直接成因。