[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04L 12/24 (2006.01)
H04L 29/00 (2006.01)

[12] 发明专利说明书

专利号 ZL 200310106551.8

[45] 授权公告日 2007 年 2 月 14 日

[11] 授权公告号 CN 1300982C

[22] 申请日 2003.12.5
[21] 申请号 200310106551.8
[73] 专利权人 中国科学技术大学
地址 230026 安徽省合肥市金寨路 96 号
[72] 发明人 王煦法 曹先彬 罗文坚 马建辉 张四海
[56] 参考文献
CN1439208A 2003.8.27
US2002/007330A1 2002.6.13
审查员 郭风顺

[74] 专利代理机构 合肥华信专利商标事务所
代理人 余成俊

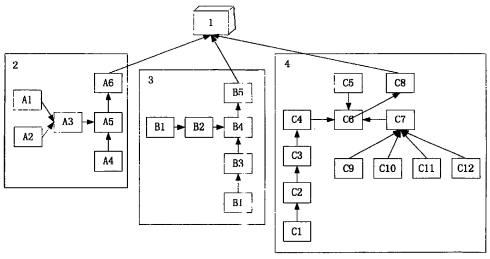
权利要求书 3 页 说明书 13 页 附图 1 页

[54] 发明名称

一种分层协同的网络病毒和恶意代码识别方法

[57] 摘要

本发明分层协同的网络病毒和恶意代码识别方法，特征是借鉴生物免疫强大的自我保护机制，将网络病毒和恶意代码识别技术和生物免疫系统的多层保护机制对应起来，通过统计分析关键词词频判断待检测脚本的危险度，基于注册表操作“自我集”的角度来分析判断注册表写入表项路径的异常行为，以及对应用程序编程接口执行序列进行非我识别，最终将全部异常行为信息通过网络发送到网络控制台，较好地解决了未知网络病毒和恶意代码的异常行为识别问题，对未知网络病毒和恶意代码的具较好的识别能力，实现了对单个系统及整个子网中的网络病毒和恶意代码异常行为的监控和管理。



1、一种分层协同的网络病毒和恶意代码识别方法，包括：

从脚本文件中分离出关键词，通过注入动态链接库 DLL 的方法获得应用程序编程接口 API 执行序列和注册表写入表项路径，将注册表写入表项路径和 API 序列保存在硬盘或内存中；其特征在于：

对脚本的关键词词频统计分析并作出异常判断；

对注册表写入表项路径进行自我识别并作出异常判断；

对 API 序列进行非我识别并作出异常判断；

将异常行为信息发送到网络控制台；

所述脚本文件是指用 Javascript 语言写的脚本文件、用 VBScript 语言写的脚本文件以及嵌入了 Javascript 或 VBScript 代码的脚本文件；

所述注入 DLL 获得 API 执行序列和注册表写入表项路径是指，通过将 DLL 作为远程线程注入到目标程序中，然后采用替换输入地址表 IAT 的方法截取目标程序的 API 执行序列，并从注册表 API 函数的参数获得注册表写入表项路径；

所述对脚本的关键词统计分析并作出异常判断是指从脚本文件中分离出 29 个关键词 copyfile、Createobject、Delete、FolderDelete、RegWrite、Virus、.Write、GetSpecialFolder、keys、opentextfile、readall、.save、startup、execute、.add、buildpath、copyfolder、createfolder、createtextfile、deletefile、fileexists、folderexists、getfile、getfolder、getparentfolder、format、.run、do copy、document.write，并进行如下步骤：

（1）将 29 个关键词分为三组，第一组为创建对象关键词：Createobject；第二组为本身无危险操作关键词：Virus、.Write、GetSpecialFolder、keys、opentextfile、readall、startup、execute、.add、buildpath、fileexists、folderexists、getfile、getfolder、getparentfolder、.run、document.write；第三组为具有可能进行破坏操作的关键词：copyfile、Delete、FolderDelete、RegWrite、.save、copyfolder、createfolder、createtextfile、deletefile、format、do copy；

（2）统计正常脚本中这 29 个关键词出现的词频的期望值 f_i，$1 \le i \le 29$，统计异常脚本中这 29 个关键词出现的词频的期望值 f_i'，$1 \le i \le 29$，计算 29 个关键词在正常和异常脚本中的归一化词频差 $e_i = (f_i - f_i') \Big/ \sum_{i=1}^{29} (f_i - f_i')$，$1 \le i \le 29$；

（3）统计在当前待检测脚本中关键词出现的词频 m_i，$1 \le i \le 29$，计算待检测脚本的

危险度 Risk，

$$Risk = G\sum_{i=1}^{29} P(i)F(i)$$

其中 $P(i)$、$F(i)$ 和 G 分别为：

$$P(i) = \begin{cases} 0 & \text{第}i\text{个关键字在脚本中未出现} \\ 1 & \text{第}i\text{个关键字在脚本中出现过} \end{cases} \quad 1 \le i \le 29$$

$$F(i) = \begin{cases} 0 & m_i < 1 \\ e_i & m_i = 1 \\ e_i(1 + 2^{-1} + 2^{-2} + ... + 2^{-m_i}) = 2e_i(1 - 2^{1-m_i}) & m_i > 1 \end{cases} \quad 1 \le i \le 29$$

$$G = \begin{cases} 1 & \text{脚本同时包含第一组和第三组关键词} \\ 0 & \text{否则} \end{cases}$$

（4）将危险度阈值 TH 定义为：

$$TH = \sum_{i=0}^{29} P(i) \Big/ 29$$

当危险度 Risk 超过阈值 TH 时，发送预警信息至网络控制台；

所述对注册表写入表项路径进行自我识别并作出异常判断采取如下步骤：

（1）收集正常状态下目标程序的正常注册表写入表项路径，并存入数据库中，每个正常注册表写入表项路径称为“自我”，其集合称为“自我集”；

（2）读取当前注册表写入表项路径，与数据库中原有的“自我”操作相比较，如果不在“自我集”中，则发送异常行为信息至网络控制台；

所述对 API 序列进行非我识别并作出异常判断采取如下步骤：

（1）API 选取操作：

（a）截取正常状态下目标程序的 API 序列，并以滑动步长为 W_0 的方式将之截成长度为 L_0 的串集 S_0；

（b）截取带毒运行状态下目标程序的 API 序列，并以滑动步长为 W_0 的方式将之截成长度为 L_0 的串集 R_0；

（c）比较串集 S_0 和 R_0 中不同的序列，抽取出构成这些序列的 API 函数，将这些 API 函数作为待监视的 API 函数集；

（2）根据选定的 API 函数，截取正常状态下目标程序的 API 序列，并以滑动步长为 W 将之截成长度为 L 的串，生成自我集 S；

（3）获取目标程序的当前 API 执行序列，并以滑动步长为 W 将之截成长度为 L 的串，每次读取 N 个 API 序列进行如下检测过程：

（a）产生初始检测器集 D_0：根据选定的 API 函数随机产生预检测器，过滤自我，进而获得初始检测器集；这里的匹配策略是部分匹配策略，即两个序列匹配当且仅当这两个字符串在连续 r 个位置一致；

（b）比较当前 AP 执行序列和检测器集中的任一检测器：如果发现匹配则标记该序列并将总匹配数目加 1，当实时获取的待检测 API 序列总匹配数目达到阈值 G_n 时，向网络控制台发异常行为信息；

（c）如果进化代数 t 超过阈值 G_e 或全部 API 序列已被标记，继续读取下一批 API 序列并进行检测；否则，对于不匹配的 API 序列，则依据亲合度变异、基因库进化、随机产生的三个子集 D_A、D_G、D_R 和记忆集 D_M 共同组成下一代检测器集 $D_t = D_A + D_G + D_R + D_M$，且 D_A、D_G、D_R 子集满足 $\frac{D_A}{1} \approx \frac{D_G}{2} \approx \frac{D_M}{1}$；

通过亲合度变异产生检测器子集 D_A，亲合度变异是指当 API 序列与检测器集中的任一检测器的匹配程度超过亲合度阈值 G_f 时，通过变异产生 N_c 个子代个体，$N_c \geq 1$；

通过基因库进化产生检测器子集 D_G，基因库进化是指提高组成有效检测器的 API 的选择概率，即 $P_{api} = P_{api} + \Delta P$；并在实际生成检测器时，依据 API 选择概率通过赌轮法生成预检测器，最后过滤自我生成检测器子集 D_G；

通过随机产生检测器子集 D_R；

将已有的能够匹配异常序列的检测器组成记忆集 D_M；

所述网络控制台是指用来接收对脚本、注册表写入表项路径以及 API 序列进行分析处理所获得的异常信息的网络程序。

一种分层协同的网络病毒和恶意代码识别方法

技术领域:

本发明属于计算机网络安全技术领域，特别是涉及网络病毒和恶意代码的识别技术。

背景技术:

根据在美国出版的电气和电子工程师协会《潜力》杂志（IEEE POTENTIALS，2001 年 10 月第四期第 16-19 页）介绍，现有的计算机反病毒识别技术大致可以分为以下几种：（1）基于特征码的扫描，主要针对已知病毒。（2）虚拟机技术，其基本思想是将可疑程序置于虚拟机环境下执行用于判断是否为病毒，但目前仍然面临虚拟机的效果以及如何保证虚拟机的自身安全性等诸多问题。（3）启发式方法，其基本思想是试图通过泛化的特征码来检测家族病毒以及检测未知病毒。该方法常常依赖于特征码技术和虚拟机技术，目前对未知病毒的识别效果也有待改进。（4）行为分析法，即利用监视病毒的特有行为来检测病毒的方法。这种方法要求首先归纳出病毒的一般行为模式，然后设计一个有限状态机对应该行为模式，状态迁移对应于程序的行为，接受状态为检测到病毒。这种方法的问题是对层出不穷的新病毒，很难归纳出一个一般的行为模式。（5）校验和法。这种方法在机器的初始状态生成一个校验信息并保存，然后在校验信息发生异常变化时（校验失败）报警，这种方法的主要问题是实现起来开销太大，同时也面临新应用程序的安装和版本升级等问题。总的来说，在现有的计算机反病毒技术中，特征码扫描技术主要用于识别已知病毒，其余各种针对未知病毒而提出的识别技术都还有各自的缺点和局限性。

由于网络病毒和恶意代码是在近年来才开始流行并带来严重危害的网络安全事件，中国专利申请号 96114050 提出的一种可防止计算机病毒感染的方法只能防范部分早期的计算机病毒，目前这种防毒卡已经彻底淡出市场；中国专利申请号 96109573 提出的防火墙系统是对进、出内部网络的连接或信息进行安全检查，基本不具有识别网络病毒和恶意代码的能力。因此，这些技术不适用于网络病毒和恶意代码的识别。

发明内容:

针对现有网络病毒和恶意代码识别技术的不足，本发明提出一种分层协同的网络病毒和恶意代码识别方法，以解决未知网络病毒和恶意代码的异常行为识别问题，实现对单个系统及整个子网中的未知网络病毒和恶意代码异常行为的监控。

本发明分层协同的网络病毒和恶意代码识别方法，包括：从脚本文件中分离出关键词，通过注入动态链接库（Dynamic Linked Library：简称 DLL）的方法获得应用程序编程接口（Application Programming Interface：简称 API）执行序列和注册表写入表项路径，将注册表写入表项路径和 API 序列保存在硬盘或内存中；其特征在于：

对脚本的关键词词频统计分析并作出异常判断；

对注册表写入表项路径进行自我识别并作出异常判断；

对 API 序列进行非我识别并作出异常判断；

将异常行为信息发送到网络控制台；

所述脚本文件是指用 Javascript 语言写的脚本文件、用 VBScript 语言写的脚本文件以及嵌入了 Javascript 或 VBScript 代码的脚本文件；

所述注入 DLL 获得 API 执行序列和注册表写入表项路径是指，通过将 DLL 作为远程线程注入到目标程序（即待监控程序）中，然后采用替换输入地址表（Import Address Table：IAT）的方法截取目标程序的 API 执行序列，并从注册表 API 函数的参数获得注册表写入表项路径；

所述对脚本的关键词统计分析并作出异常判断是指从脚本文件中分离出 29 个关键词 copyfile、Createobject、Delete、FolderDelete、RegWrite、Virus、.Write、GetSpecialFolder、keys、opentextfile、readall、.save、startup、execute、.add、buildpath、copyfolder、createfolder、createtextfile、deletefile、fileexists、folderexists、getfile、getfolder、getparentfolder、format、.run、do copy、document.write，并进行如下步骤：

（1）将 29 个关键词分为三组，第一组为创建对象关键词：Createobject；第二组为本身无危险操作关键词：Virus、.Write、GetSpecialFolder、keys、opentextfile、readall、startup、execute、.add、buildpath、fileexists、folderexists、getfile、getfolder、getparentfolder、.run、document.write；第三组为具有可能进行破坏操作的关键词：copyfile、Delete、FolderDelete、RegWrite、.save、copyfolder、createfolder、createtextfile、deletefile、format、do copy；

（2）统计正常脚本中这 29 个关键词出现的词频的期望值 f_i，$1 \le i \le 29$，统计异常脚本中这 29 个关键词出现的词频的期望值 f_i'，$1 \le i \le 29$，计算 29 个关键词在正常和异常脚本中

的归一化词频差 $e_i = (f_i - f_i') \Big/ \sum_{i=1}^{29}(f_i - f_i')$，$1 \le i \le 29$；

（3）统计在当前待检测脚本中关键词出现的词频 m_i，$1 \le i \le 29$，计算待检测脚本的危险度 Risk，

$$Risk = G\sum_{i=1}^{29} P(i)F(i)$$

其中 $P(i)$、$F(i)$ 和 G 分别为：

$$P(i) = \begin{cases} 0 & （第i个关键字在脚本中未出现） \\ 1 & （第i个关键字在脚本中出现过） \end{cases} \quad (1 \le i \le 29)$$

$$F(i) = \begin{cases} 0 & m_i < 1 \\ e_i & m_i = 1 \\ e_i(1 + 2^{-1} + 2^{-2} + ... + 2^{-m_i}) = 2e_i(1 - 2^{1-m_i}) & m_i > 1 \end{cases} \quad (1 \le i \le 29)$$

$$G = \begin{cases} 1 & 脚本同时包含第一组和第三组关键词 \\ 0 & 否则 \end{cases}$$

（4）将危险度阈值 TH 定义为：

$$TH = \sum_{i=0}^{29} P(i) \Big/ 29$$

当危险度 Risk 超过阈值 TH 时，发送预警信息至网络控制台；

所述对注册表写入表项路径进行自我识别并作出异常判断采取如下步骤：

（1）收集正常状态下目标程序（待监控程序）的正常注册表写入表项路径，并存入数据库中，每个正常注册表写入表项路径称为“自我”，其集合称为“自我集”；

（2）读取当前注册表写入表项路径，与数据库中原有的“自我”操作相比较，如果不在“自我集”中，则发送异常行为信息至网络控制台；

所述对 API 序列进行非我识别并作出异常判断采取如下步骤：

（1）API 选取操作：

（a）截取正常状态下目标程序的 API 序列，并以滑动步长为 W_0 的方式将之截成长度为 L_0 的串集 S_0；

（b）截取带毒运行状态下目标程序的 API 序列，并以滑动步长为 W_0 的方式将之截成长度为 L_0 的串集 R_0；

（c）比较串集 S_0 和 R_0 中不同的序列，抽取出构成这些序列的 API 函数，将这些 API 函数作为待监视的 API 函数集；

（2）根据选定的 API 函数，截取正常状态下目标程序的 API 序列，并以滑动步长为 W 将之截成长度为 L 的串，生成自我集 S；

（3）获取目标程序的当前 API 执行序列，并以滑动步长为 W 将之截成长度为 L 的串，每次读取 N 个 API 序列进行如下检测过程：

（a）产生初始检测器集 D_0：根据选定的 API 函数随机产生预检测器，过滤自我（即把与自我匹配的 API 序列删除），进而获得初始检测器集；这里的匹配策略是部分匹配策略，即两个序列匹配当且仅当这两个字符串在连续 r 个位置一致；

（b）比较当前 AP 执行序列和检测器集中的任一检测器：如果发现匹配则标记该序列并将总匹配数目加 1，当实时获取的待检测 API 序列总匹配数目达到阈值 G_n 时，向网络控制台发异常行为信息；

（c）如果进化代数 t 超过阈值 G_e 或全部 API 序列已被标记，继续读取下一批 API 序列并进行检测；否则，对于不匹配的 API 序列，则依据亲合度变异、基因库进化、随机产生的三个子集 D_A、D_G、D_R 和记忆集 D_M 共同组成下一代检测器集 $D_t = D_A + D_G + D_R + D_M$，且 D_A、D_G、D_R 子集满足 $\frac{D_A}{1} \approx \frac{D_G}{2} \approx \frac{D_M}{1}$；

通过亲合度变异产生检测器子集 D_A，亲合度变异是指当 API 序列与检测器集中的任一检测器的匹配程度超过亲合度阈值 G_f 时，通过变异产生 $N_c(N_c \geq 1)$ 个子代个体；

通过基因库进化产生检测器子集 D_G，基因库进化是指提高组成有效检测器的 API 的选择概率，即 $P_{api} = P_{api} + \Delta P$；并在实际生成检测器时，依据 API 选择概率通过赌轮法生成预检测器，最后过滤自我生成检测器子集 D_G；

通过随机产生检测器子集 D_R；

将已有的能够匹配异常序列的检测器组成记忆集 D_M；

所述网络控制台是指用来接收对脚本、注册表写入表项路径以及 API 序列进行分析处理所获得的异常信息的网络程序。

与现有技术相比较，本发明的优点在于：

1、本发明通过统计正常脚本和异常脚本中选定的 29 个关键词词频来获得归一化词频，并以此为基础给出危险度和危险度阈值计算方法来判断待检测脚本的危险度，解决了恶意脚本的识别问题。

2、本发明基于注册表操作“自我集”的角度来判断分析注册表写入表项路径的异常行为，适用于各种目标程序。

3、本发明将包括基因库进化、随机产生、亲合度变异和记忆集在内的四个学习与记忆模块和 API 执行序列的异常检测结合起来，使得对 API 序列进行非我识别的异常检测效果较好，且适用于各种目标程序。

4、本发明借鉴生物免疫强大的自我保护机制，首次将对脚本进行关键词统计分析、对注册表写入表项路径进行自我识别、对 API 执行序列进行非我识别这三个方面统一起来对目标程序的异常行为进行监视，使得对未知网络病毒和恶意代码的识别效果更好。

5、采用本发明可以自动详实地记录程序的注册表写入表项路径和 API 执行序列，为进一步分析网络病毒和恶意代码提供了第一手资料。

综上所述，本发明借鉴生物免疫强大的自我保护机制，将网络病毒和恶意代码识别技术和生物免疫系统的多层保护机制对应起来，分别从对脚本进行关键词统计分析、对注册表写入表项路径进行自我识别、对 API 执行序列进行非我识别这三个方面来较好地解决了未知网络病毒和恶意代码的异常行为识别问题，进而解决了现有技术对病毒变种和未知病毒难于识别的问题，不仅实现了对单个系统中网络病毒和恶意代码异常行为的监控，而且使得管理员能够通过网络控制台对整个子网的安全情况实时监控和管理。

附图说明：

图 1 是本发明进行分层协同的网络病毒和恶意代码识别的工作流程图。

具体实施方式：

下面结合附图和实例对本发明方法作进一步具体的描述。

实施例 1：

1、利用几台通用微型个人计算机，通过交换机连成一个网络环境

本实施例中具体采用的是三台奔腾 IV 微机，和一台 Dell 笔记本，以及一台企业服务器，外加一个长城 24 端口 10M/100M 自适应以太网交换机 GES-1125 交换机，通过交换机将几台微机三台奔腾 IV 微机、一台 Dell 笔记本和一台企业服务器连成一个网络。

图 1 给出了本实施例进行分层协同的网络病毒和恶意代码识别的工作流程。箭头方向指示了工作流向顺序，箭头尾部是下一步的输入，箭头端是下一步进行的操作。其中一台奔腾序列微机用于运行网络控制台 1，其余的两台奔腾 IV 微机、一台 Dell 笔记本和一台企业服务器都用于执行对脚本进行关键词词频统计分析 2、对注册表写入表项路径进行自我识别 3 和对 API 执行序列进行非我识别 4，并将这三个方面的分析结果都发送到网络控制台 1。

2、对脚本的关键词统计分析并作出恶意代码异常判断

如图 1 中的对脚本进行关键词词频统计分析 2，具体采取如下操作步骤：

（1）收集大量的正常脚本文件和恶意脚本文件，建议正常脚本文件和恶意脚本文件均不少于 50 个，从脚本文件中分离出 29 个关键词 copyfile、Createobject、Delete、FolderDelete、RegWrite、Virus、.Write、GetSpecialFolder、keys、opentextfile、readall、.save、startup、execute、.add、buildpath、copyfolder、createfolder、createtextfile、deletefile、fileexists、folderexists、getfile、getfolder、getparentfolder、format、.run、do copy、document.write：

（2）将 29 个关键词分为三组，第一组为创建对象关键词：Createobject，第二组为本身无危险操作关键词：Virus、.Write、GetSpecialFolder、keys、opentextfile、readall、startup、execute、.add、buildpath、fileexists、folderexists、getfile、getfolder、getparentfolder、.run、document.write，第三组为具有可能进行破坏操作的关键词：copyfile、Delete、FolderDelete、RegWrite、.save、copyfolder、createfolder、createtextfile、deletefile、format、do copy；

（3）如图 1 中的正常脚本关键词词频统计 A1：统计正常脚本中这 29 个关键词出现的词频的期望值 $f_i(1 \le i \le 29)$；

（4）如图 1 中的异常脚本关键词词频统计 A2：统计恶意脚本中这 29 个关键词出现的词频的期望值 $f_i'(1 \le i \le 29)$；

（5）如图 1 中的计算归一化词频 A3：计算 29 个关键词在正常和异常脚本中的归一化词

频差 $e_i = (f_i - f_i') \Big/ \sum_{i=1}^{29}(f_i - f_i')$ $(1 \le i \le 29)$；

（6）如图 1 中的分析待检测脚本 A4：从硬盘中读取指定的脚本文件或从浏览器（如 IExplore.exe）的临时文件目录中读取浏览器正在访问的脚本文件，统计在该脚本中这 29 个关键词出现的词频 m_i；

（7）如图 1 中的危险度计算 A5：计算待检测脚本的危险度 Risk，

$$Risk = G\sum_{i=0}^{29} P(i)F(i)$$

其中 $P(i)$、$F(i)$ 和 G 分别为：

$$P(i) = \begin{cases} 0 & \text{（第}i\text{个关键字在脚本中未出现）} \\ 1 & \text{（第i个关键字在脚本中已出现）} \end{cases} \qquad (1 \le i \le 29)$$

$$F(i) = \begin{cases} 0 & m_i < 1 \\ e_i & m_i = 1 \\ e_i(1 + 2^{-1} + 2^{-2} + \ldots + 2^{-m_i}) = 2e_i(1 - 2^{1-m_i}) & m_i > 1 \end{cases}$$

$$G = \begin{cases} 1 & \text{脚本同时包含第一组和第三组关键词} \\ 0 & \text{否则} \end{cases}$$

（8）计算危险度阈值，危险度阈值 TH 的计算方法为：

$$TH = \sum_{i=0}^{29} P(i) \Big/ 29$$

（9）如图 1 中的发送预警信息 A6：当危险度 Risk 超过阈值 TH 时，通过网络将预警信息发送至网络控制台 1（根据 Windows 操作系统的 Socket 编制相应的发送接收程序）。

3、如图 1 中的对注册表写入表项路径进行自我识别 3，对注册表写入表项路径进行自我识别并作出异常判断可采取如下实施步骤：

（1）如图 1 中的截取注册表写入表项路径 B1：注入截取注册表 API 函数的 DLL 至目标程序中，如 IExplore.exe 和 Outlook.exe，获得注册表 API 函数执行情况和参数，并从注册表 API 函数的参数获得注册表写入表项路径。注入 DLL 的方法可以用远程线程注入方法，远程线程函数可参见 MSDN 中的 CreateRemoteThread，在注入 DLL 中采用替换 IAT（Import Address Table：输入地址表）的方法可截取目标程序的 API 执行序列，注意要对

GetProcAddress 和 LoadLibraryA、LoadLibraryExA、LoadLibraryW、LoadLibraryExW 做特殊处理，具体可参见微软公司出版、杰弗里•里克特（Jeffrey Ritcher）著的《窗口操作系统核心编程》（Programming Applications for Windows）；

（2）如图 1 中的收集自我 B2：在正常状态下运行目标程序，如用 IExplore.exe 访问不含恶意代码的网页或用 Outlook.exe 收取不含网络病毒和恶意代码的信件等，收集正常状态下目标程序（在此是 IExplore.exe 或 Outlook.exe）的正常注册表写入表项路径，并存入数据库中，每个正常注册表写入表项路径称为“自我”，其集合称为“自我集”；

（3）如图 1 中的收集当前待检测的注册表写入表项路径 B3：在目标程序运行过程中，通过注入的 DLL 实时获取目标程序的注册表写入表项路径，如 IExplore.exe 或 Outlook.exe 的注册表写操作，并将注册表写入表项路径保存在共享内存中；与此同时，注册表检测模块从共享内存中读取当前注册表写入表项路径，与数据库中原有的“自我”操作相比较，如图 1 中的自我识别 B4；如果不在“自我集”中，则发送异常行为信息至网络控制台，如图 1 中的发送异常行为信息 B5。

4、如图 1 中的对 API 执行序列进行非我识别 4，对 API 序列进行非我识别并作出异常判断可采取如下实施步骤。

需要说明的是：如果不考虑速度的话，可以不运行第（1）步和第（2）步，直接使用全体 API 函数；或者不运行第（1）步，直接在全体 API 函数中进行选取。

（1）首先对全部 API 函数进行重新编号，并确定目标程序使用的 API 函数总集，如图 1 中的使用的 API 集合 C1：

（a）由于全部 API 函数过多，约 3000 个，可以将 API 函数分为 20 组，每组约 150 个，并针对各组 API 函数生成相应的注入 DLL；

（b）将这些 DLL 分别注入目标程序，如 IExplore.exe 或 Outlook.exe，在正常和带毒情况下运行目标程序，并从记录的文件中获得目标程序使用的 API 函数列表；

（2）API 选取操作，如图 1 中的 API 选取 C2：

（a）截取正常状态下目标程序的 API 序列，并以滑动步长为 W_0 的方式将之截成长度为 L_0 的串集 S_0，其中 W_0 的取值可以为 1 至 L_0 间的任意整数，建议取 $\lfloor L_0/2 \rfloor$；L_0 的取值可以为大于 8 的整数，建议取 8、16、32 或 64；

（b）截取带毒运行状态下目标程序的 API 序列，并以滑动步长为 W_0 的方式将之截成长

度为L_0的串集R_0；

（c）比较串集S_0和R_0中不同的序列，抽取出构成这些序列的 API 函数，将这些 API 函数作为待监视的API函数集；

（3）如图1中的API重新编号C3：对选定的API函数进行重新编号，以便于表示API序列；

（4）如图1中的收集自我C4：根据选定的API函数，截取正常状态下目标程序的API序列，并以滑动步长W将之截成长度为L的串，生成自我集S，其中W_0的取值可以为1至L_0间的任意整数，建议取$\lfloor L_0/2 \rfloor$；L_0的取值可以为大于8的整数，建议取8、16、32或64；

（5）获取目标程序的当前API执行序列，每次读取N个API序列进行如下检测过程，如IExplore.exe或Outlook.exe，建议N取值为128，如图1中的获取目标程序的当前API执行序列C5：

（a）如图1中的启动检测并判断结束条件是否满足C7，产生初始检测器集D_0：根据选定的API函数随机产生预检测器，过滤自我（即把与自我匹配的API序列删除），进而获得初始检测器集；这里的匹配策略是部分匹配策略，即两个序列匹配当且仅当这两个字符串在连续r个位置一致；

（b）如图1中的匹配C6，比较当前API执行序列和检测器集中的任一检测器：如果发现匹配则标记该序列并将总匹配数目加1，当实时获取的待检测API序列总匹配数目达到阈值G_n时，向网络控制台发异常行为信息，如图1中的发送异常行为信息C8；

（c）如图1中的启动检测并判断结束条件是否满足C7，如果进化代数t超过阈值G_e或全部API序列已被标记，继续对下一批API序列进行检测；

（d）对于不匹配的API序列，则依据亲合度变异、基因库进化、随机产生的三个子集D_A、D_G、D_R和记忆集D_M共同组成下一代检测器集$D_t = D_A + D_G + D_R + D_M$，且$D_A$、$D_G$、$D_R$子集满足$\frac{D_A}{1} \approx \frac{D_G}{2} \approx \frac{D_M}{1}$；

（e）如图1中的亲合度变异C9，检测器子集D_A由亲合度变异产生，亲合度变异是指当API序列与检测器集中的任一检测器的匹配程度超过亲合度阈值G_f时，通过变异产生

$N_c(N_c \geq 1)$个子代个体；

一种建议采用的具体变异方法可以为：如果当前 API 执行序列与任一检测器匹配位数超过亲和度变异阈值，随机生成一个[1，L]的数 a，对此检测器第 a 位发生变异，得到一个子代检测器；如此循环 4 次，对每个需要变异的检测器生成 4 个子代检测器。

（f）如图 1 中的基因库进化 C10：检测器子集 D_G 由基因库进化产生，基因库进化是指提高组成有效检测器的 API 的选择概率，使得在通过赌轮法生成预检测器时，该 API 具有较高的选择概率，即 $P_{api} = P_{api} + \Delta P$。需要指出的是，所有 API 的选择概率在开始时是一致的，具有相同的被选择概率 P_{api}；而且为避免局部最优，每一次基因库进化的步长是很小的，即 API 选择概率的递增量 ΔP 很小，且对于所有的 API，这里 ΔP 是相同的；

基因库进化中 API 选择概率提升部分的代码可以简写为：

for(有效检测器的每一个基因 Gene)

Begin

该基因 Gene 的选择概率 P[Gene]=P[Gene]+ΔP。

End

其中 ΔP 是一个较小的常常数。如果对于任意 Gene，初始 P[Gene]为 100，ΔP 可设为 0.1 或 0.01。

（g）如图 1 中的随机产生 C11，检测器子集 D_R 由随机产生，随机产生检测器是指在每一代检测器集中保持一定比例的检测器来自于随机产生的方式，这是为了维持检测器的多样性；

（h）如图 1 中的记忆集 C12：记忆集 D_M 由能够匹配异常序列的检测器组成，它既可以在开始实时检测前通过离线生成，也可以在实际监测过程中将能检测到异常序列的检测器加入到记忆集中；

5、网络控制台 1 是具有网络数据报接收功能的程序，可以用可视化编程工具编写，如 VC++或 Delphi 编写，具有可视化界面并能够接收网络数据报和读写数据库；数据库可以使用 Microsoft SQL Server 数据库。管理员可以通过网络控制台获取对脚本、注册表写入表项路径以及 API 序列进行分析处理获得的异常行为信息。

6、按照上述方法，包括对脚本进行关键词词频统计分析 2、对注册表写入表项路径进行自我识别 3 和对 API 执行序列进行非我识别 4，下面列出了针对 75 种 Email 病毒、Email 蠕虫病毒和恶意代码的检测结果，结果表明本发明对网络病毒和恶意代码具有很好的效果。

序号	名称	种类	是否报病毒
1	Bloodhound.vbs.worm	Email,worm	是
2	Bloodhound.vbs.worm 变种	Email,worm	是
3	vbs.mesut	email	是
4	Jesus	Email,worm	是
5	Vbs.jadra	email	是
6	Vbs.infi	email	是
7	Vbs.hatred.b	email	是
8	Vbs.godog	email	是
9	Vbs.hard	Email,worm	是
10	Vbs.gascript	Email,Trojan	是
11	I-Worm.CIAN	email	是
12	Vbs.vbswg.qen	Email,worm	是
13	I-Worm.doublet	Email,worm	是
14	White house	Email,worm	是
15	I-Worm.chu	email	是
16	Loveletter	Email.worm	是
17	freelink	Email,worm	是
18	Mbop.d	Email,worm	是
19	Kounikewa	Email,worm	是
20	json888	恶意代码	是
21	gator[1]	恶意代码	是
22	overkill2	恶意代码	是
23	redlof	恶意代码	是

24	script.unrealer	恶意代码	是
25	vbs.both	恶意代码	是
26	VBS.kremp	恶意代码	是
27	script.exploit	恶意代码	否
28	script.happytime	恶意代码	是
29	vbs.godog	恶意代码	是
30	I-worm.doublet	恶意代码	是
31	I-worm.chu	恶意代码	是
32	vbs.baby	恶意代码	是
33	vbs.gascript	恶意代码	是
34	vbs.jesus	恶意代码	是
35	vbs.mbop.d	恶意代码	是
36	vbs.fasan	恶意代码	是
37	vbs.hard.vbs	恶意代码	是
38	vbs.infi	恶意代码	是
39	vbs.jadra	恶意代码	是
40	LOVE-LETTER-FOR-YOU	恶意代码	是
41	vbs.mesut	恶意代码	是
42	JS.Exception.Exploit1	恶意代码	是
43	JS.Exception.Exploit2	恶意代码	是
44	自编 Writefile	恶意代码	是
45	Writefile 变种	恶意代码	是
46	IRC.salim	恶意代码	是
47	Vbs.vbswg.qen	恶意代码	是
48	Bloodhound.vbs.3	恶意代码	是
49	Bloodhound.vbs.3 变种 1	恶意代码	是
50	Bloodhound.vbs.3 变种 2	恶意代码	是
51	Bloodhound.vbs.3 变种 3	恶意代码	是

52	Bloodhound.vbs.3 变种 4	恶意代码	是
53	Bloodhound.vbs.3 变种 5	恶意代码	是
54	Bloodhound.vbs.3 变种 6	恶意代码	是
55	Bloodhound.vbs.3 变种 7	恶意代码	是
56	Bloodhound.vbs.3 变种 8	恶意代码	是
57	Bloodhound.vbs.3 变种 9	恶意代码	是
58	Vbs.bound	恶意代码	是
59	Vbs.charl	恶意代码	是
60	VBS.Phram.D(vbs.cheese)	恶意代码	是
61	Vbs.entice	恶意代码	是
62	Vbs.ave.a	恶意代码	是
63	Vbs.exposed	恶意代码	是
64	Vbs.annod(vbs.jadra)	恶意代码	是
65	Vbs.nomekop	恶意代码	是
66	Html.reality(vbs.reality)	恶意代码	是
67	Bloodhound.vbs.3	恶意代码	是
68	Bloodhound.vbs.3 变种 1	恶意代码	是
69	Bloodhound.vbs.3 变种 2	恶意代码	是
70	Bloodhound.vbs.3 变种 3	恶意代码	是
71	Bloodhound.vbs.3 变种 4	恶意代码	是
72	Bloodhound.vbs.3 变种 5	恶意代码	是
73	Bloodhound.vbs.3 变种 6	恶意代码	是
74	Bloodhound.vbs.3 变种 7	恶意代码	是
75	Bloodhound.vbs.3 变种 8	恶意代码	是

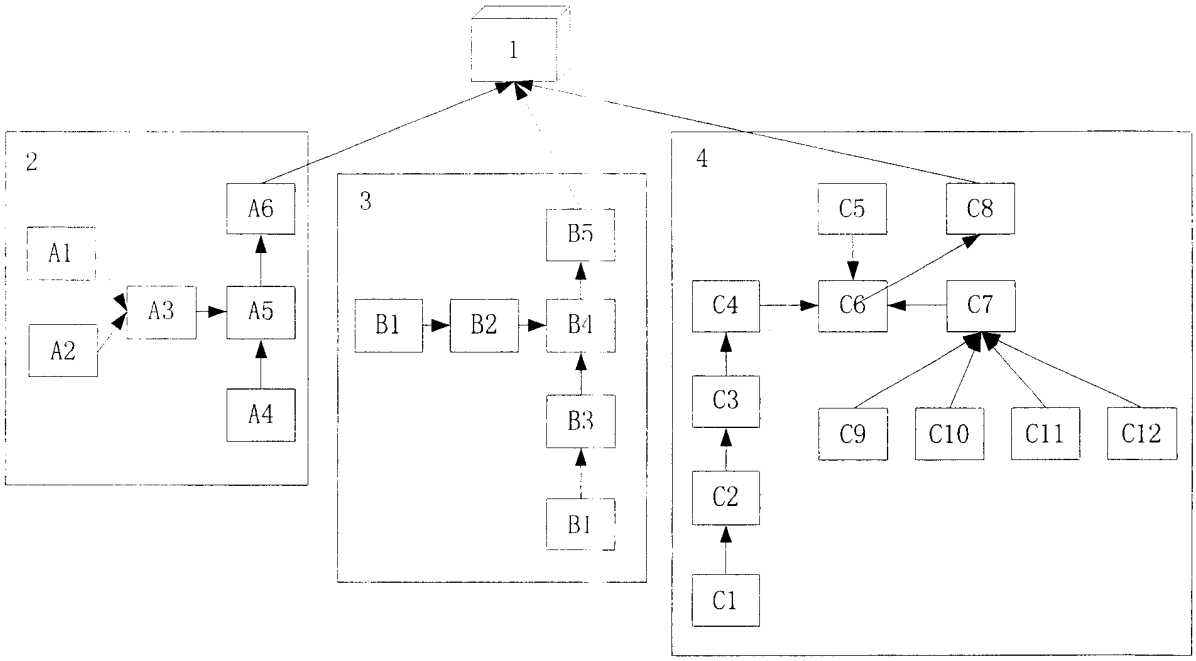

1
2
A1
A2
A3
A4
A5
A6
3
B1
B2
B3
B4
B5
4
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12

图 1