(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103617069 A
(43)申请公布日 2014.03.05

(21)申请号 201310595022.2

(22)申请日 2011.09.14

(62)分案原申请数据
201110272443.2 2011.09.14

(71)申请人 北京奇虎科技有限公司
地址 100088 北京市西城区新街口外大街28号D座112室(德胜园区)
申请人 奇智软件(北京)有限公司

(72)发明人 邵坚磊 谭合力

(74)专利代理机构 北京润泽恒知识产权代理有限公司 11319
代理人 苏培华

(51)Int.Cl.
G06F 9/455(2006.01)
G06F 21/56(2013.01)

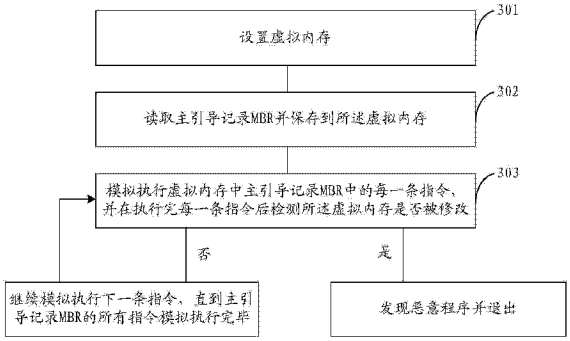
权利要求书2页 说明书8页 附图6页

(54)发明名称

恶意程序检测方法和虚拟机

(57)摘要

本申请提供了一种检测恶意程序的方法、装置及虚拟机，以解决现有技术无法检测出变形的恶意程序的问题。所述方法包括：设置虚拟内存；读取主引导记录MBR并保存到所述虚拟内存；模拟执行虚拟内存中主引导记录MBR中的每一条指令，并在执行完每一条指令后检测所述虚拟内存是否被修改，如果被修改，则发现恶意程序；否则，继续模拟执行下一条指令，直到主引导记录MBR的所有指令模拟执行完毕。本申请可以无视任何特征码变换技术，只要实际运行中发生了这个行为即可被检测出来。



CN 103617069 A

1. 一种恶意程序检测方法，其包括：

在内存中设置一虚拟内存；

开机通电自检后，将主引导记录 MBR 读入所述虚拟内存中；

模拟执行读入所述虚拟内存中的主引导记录 MBR 中的每一条指令；

在执行完每一条指令后检测所述虚拟内存是否被修改；

如果所述虚拟内存被修改，则表明发现恶意程序。

2. 根据权利要求 1 所述的方法，其中，所述检测所述虚拟内存是否被修改进一步包括：

检测所述虚拟内存的大小是否改变；

如果改变，则表明所述虚拟内存被修改；否则表明所述虚拟内存未被修改。

3. 根据权利要求 1 或 2 所述的方法，还包括：

如果所述虚拟内存中的主引导记录 MBR 中的每一条指令都执行完毕且所述虚拟内存并未被修改，则将控制权交给系统中真实的活动分区的引导记录，以便由所述引导记录加载操作系统启动文件。

4. 根据权利要求 3 所述的方法，还包括：对主引导记录 MBR 中的每一条指令进行反汇编并输出显示。

5. 一种恶意程序检测方法，其包括：

设置虚拟 CPU；

在内存中设置一虚拟内存；

开机通电自检后，将主引导记录 MBR 读入所述虚拟内存中；

由所述虚拟 CPU 模拟执行读入所述虚拟内存中的主引导记录 MBR 中的每一条指令；

在执行完每一条指令后检测所述虚拟内存是否被修改；

如果所述虚拟内存被修改，则表明发现恶意程序。

6. 根据权利要求 5 所述的方法，其中，所述检测所述虚拟内存是否被修改进一步包括：

检测所述虚拟内存的大小是否改变；

如果改变，则表明所述虚拟内存被修改；否则表明所述虚拟内存未被修改。

7. 一种恶意程序检测方法，其包括：

在内存中设置一虚拟内存；

设置虚拟硬盘；

开机通电自检后，将主引导记录 MBR 读入所述虚拟内存中；

将读入所述虚拟内存中的主引导记录 MBR 拷贝到所述虚拟硬盘中；

从所述虚拟硬盘中读取主引导记录 MBR 中的每一条指令并对其进行模拟执行；

在执行完每一条指令后检测所述虚拟内存是否被修改；

如果所述虚拟内存被修改，则表明发现恶意程序。

8. 根据权利要求 7 所述的方法，其中，所述检测所述虚拟内存是否被修改进一步包括：

检测所述虚拟内存的大小是否改变；

如果改变，则表明所述虚拟内存被修改；否则表明所述虚拟内存未被修改。

9. 一种虚拟机，用于检测恶意程序，其包括：虚拟 CPU 和虚拟内存，其中，

所述虚拟内存，适于存储主引导记录 MBR；

所述虚拟 CPU，适于模拟执行虚拟内存中主引导记录 MBR 中的每一条指令，并在执行完

每一条指令后检测所述虚拟内存是否被修改，如果被修改，则发现恶意程序。

10. 一种虚拟机，用于检测恶意程序，其包括：虚拟内存、虚拟硬盘和虚拟 CPU，其中，所述虚拟内存，适于存储主引导记录 MBR；

所述虚拟硬盘，适于存储从所述虚拟内存拷贝的主引导记录 MBR；

所述虚拟 CPU，适于从所述虚拟硬盘读取主引导记录 MBR 并模拟执行，并在执行完每一条指令后检测所述虚拟内存是否被修改，如果被修改，则发现恶意程序。

恶意程序检测方法和虚拟机

[0001] 本发明专利申请是申请日为2011年9月14日、申请号为201110272443.2、名称为“一种检测恶意程序的方法、装置及虚拟机”的中国发明专利申请的分案申请。

技术领域

[0002] 本申请涉及信息安全技术领域，特别是涉及一种检测恶意程序的方法、装置及虚拟机。

背景技术

[0003] 恶意程序通常是指未获得授权而非法在计算机系统中运行的程序。例如，计算机病毒就是一种运行在计算机系统中的恶意程序，可对计算机系统的安全性造成威胁。

[0004] 随着计算机和网络技术的发展，出现了形式多样的计算机病毒。其中，Rootkit是一种内核级的木马病毒，它是一种隐藏其他程序或进程的软件，可能是一个或一个以上的软件组合，广义而言，Rootkit也可视为一项技术。

[0005] 在现代操作系统中，应用程序不能直接访问硬件，而是通过调用操作系统提供的接口来使用硬件，而操作系统依赖内核空间来管理和调度这些应用。内核空间由四大部分组成，分别是：进程管理(负责分配CPU时间)、文件访问(把设备调配成文件系统，并提供一个一致的接口供上层程序调用)、安全控制(负责强制规定各个进程的具体的权限和单独的内存范围，避免各进程之间发生冲突)和内存管理(负责进程运行时对内存资源的分配、使用、释放和回收)。内核是一种数据结构，Rootkit技术通过修改内核的数据结构来隐藏其他程序的进程、文件、网络通讯和其它相关信息(比如注册表和可能因修改而产生的系统日志等)。

[0006] Bootkit是更高级的Rootkit，Bootkit通过感染MBR(Master Boot Record，磁盘主引导记录)的方式，实现绕过内核检查和启动隐身，即Bootkit是一种基于MBR的Rootkit。可以认为，所有在开机时比Windows内核更早加载、实现内核劫持的技术，都可以称之为Bootkit，例如后来的BIOS Rootkit、VBootkit、SMM Rootkit等。

[0007] 目前常规安全软件对于各种恶意程序(如病毒)的查杀，主要基于传统特征码的检测技术。这是因为通常情况下各种恶意程序在运行过程中都会运行一些特有的指令代码(即特征码)，通过查找到这些的特征码，就可以检测出恶意程序。例如对于Bootkit的检测，由于这种MBR病毒的特殊性，其一般会驻留在高端内存(即内存中的高地址位)，因此通过搜索高端内存中是否有特征码，就可以检测出Bootkit。

[0008] 但是，越来越多的病毒出现了变形，有些通过花指令，有些通过变形代码，甚至现在大多病毒都是事先加密，运行时动态解密后再运行。所以，对于这些变形的病毒而言，在达到同样效果的情况下，指令是随机变化的，按照上述方法检测时找不到特征码，因此可绕过常规安全软件的查杀。

[0009] 综上所述，目前需要解决的问题是：如何检测出变形的恶意程序，尤其是基于MBR的Bootkit或类似于Bootkit的病毒。

发明内容

[0010] 本申请提供了一种检测恶意程序的方法、装置及虚拟机，以解决现有技术无法检测出变形的恶意程序的问题。

[0011] 为了解决上述问题，本申请公开了一种检测恶意程序的方法，包括：

[0012] 设置虚拟内存；

[0013] 读取主引导记录 MBR 并保存到所述虚拟内存；

[0014] 模拟执行虚拟内存中主引导记录 MBR 中的每一条指令，并在执行完每一条指令后检测所述虚拟内存是否被修改，如果被修改，则发现恶意程序；否则，继续模拟执行下一条指令，直到主引导记录 MBR 的所有指令模拟执行完毕。

[0015] 优选的，所述检测所述虚拟内存是否被修改，包括：检测所述虚拟内存的大小是否改变，如果改变，则所述虚拟内存被修改；否则，未被修改。

[0016] 优选的，所述设置虚拟内存之前，还包括：设置虚拟 CPU；则所述模拟执行虚拟内存中主引导记录 MBR 中的每一条指令，包括：由所述虚拟 CPU 模拟执行虚拟内存中主引导记录 MBR 中的每一条指令。

[0017] 其中，所述设置虚拟 CPU 包括：初始化虚拟 CPU；所述设置虚拟内存包括：初始化 BIOS 数据区，所述 BIOS 数据区保存虚拟内存的大小。

[0018] 优选的，所述模拟执行虚拟内存中主引导记录 MBR 中的每一条指令之前，还包括：设置虚拟硬盘；则所述模拟执行虚拟内存中主引导记录 MBR 中的每一条指令包括：将虚拟内存中的主引导记录 MBR 拷贝到所述虚拟硬盘；从所述虚拟硬盘读取主引导记录 MBR，并模拟执行主引导记录 MBR 中的每一条指令。

[0019] 优选的，所述方法还包括：对主引导记录 MBR 中的每一条指令进行反汇编，并输出显示。

[0020] 本申请还提供了一种检测恶意程序的装置，包括：

[0021] 第一设置模块，用于设置虚拟内存；

[0022] 读取及保存模块，用于读取主引导记录 MBR 并保存到所述虚拟内存；

[0023] 模拟执行模块，用于模拟执行虚拟内存中主引导记录 MBR 中的每一条指令；

[0024] 检测模块，用于在所述模拟执行模块执行完每一条指令后检测所述虚拟内存是否被修改，如果被修改，则发现恶意程序；否则，触发所述模拟执行模块继续模拟执行下一条指令，直到主引导记录 MBR 的所有指令模拟执行完毕。

[0025] 优选的，所述检测模块通过检测所述虚拟内存的大小是否改变来判断是否被修改，如果改变，则所述虚拟内存被修改；否则，未被修改。

[0026] 优选的，所述装置还包括：第二设置模块，用于设置虚拟 CPU，所述虚拟 CPU 触发所述模拟执行模块和检测模块的执行。

[0027] 优选的，所述装置还包括：第三设置模块，用于设置虚拟硬盘，并将虚拟内存中的主引导记录 MBR 拷贝到所述虚拟硬盘；则所述模拟执行模块从所述虚拟硬盘读取主引导记录 MBR，并模拟执行主引导记录 MBR 中的每一条指令。

[0028] 优选的，所述装置还包括：反汇编引擎，用于对主引导记录 MBR 中的每一条指令进行反汇编，并输出显示。

[0029] 本申请还提供了一种检测恶意程序的虚拟机，包括：

[0030] 虚拟 CPU 初始化模块，用于初始化虚拟 CPU；

[0031] 虚拟内存初始化模块，用于初始化虚拟内存，并在初始化的过程中读取主引导记录 MBR 然后保存到所述虚拟内存；

[0032] 虚拟内存，用于存储主引导记录 MBR；

[0033] 虚拟 CPU，用于模拟执行虚拟内存中主引导记录 MBR 中的每一条指令，并在执行完每一条指令后检测所述虚拟内存是否被修改，如果被修改，则发现恶意程序；否则，继续模拟执行下一条指令，直到主引导记录 MBR 的所有指令模拟执行完毕。

[0034] 优选的，所述虚拟机还包括：

[0035] 虚拟硬盘初始化模块，用于初始化虚拟硬盘，并在初始化的过程中将虚拟内存中的主引导记录 MBR 拷贝到所述虚拟硬盘，所述虚拟 CPU 从虚拟硬盘读取主引导记录 MBR 并模拟执行；

[0036] 虚拟硬盘，用于存储拷贝的主引导记录 MBR。

[0037] 优选的，所述虚拟机还包括：

[0038] 反汇编引擎，用于对主引导记录 MBR 中的每一条指令进行反汇编，并输出显示。

[0039] 与现有技术相比，本申请包括以下优点：

[0040] 首先，本申请在开机后并在加载操作系统文件之前，通过模拟的方式先将读取的主引导记录 MBR 存到所设置的虚拟内存中，然后模拟实现主引导记录 MBR 的加载执行过程，并且每当模拟执行完 MBR 中的一条指令后，检测所述虚拟内存是否被修改，如果被修改，则发现恶意程序；否则，继续模拟执行下一条指令，直到主引导记录 MBR 的所有指令模拟执行完毕。

[0041] 由于实际情况中，基于 MBR 的 Bootkit 或类似于 Bootkit 的病毒等恶意程序，即使进行了变形，也必须要驻留系统的高端内存，所以必然会修改高端内存，因此上述的检测方法通过设置虚拟内存来模拟高端内存，并通过检测虚拟内存是否被修改，就可以发现可疑的恶意程序，从而无视任何特征码变换技术，只要实际运行中发生了这个行为即可被检测出来。所述的检测方法在很大程度上可以检测出过去、现在和未来的基于 MBR 的 Bootkit。

[0042] 其次，本申请还实现了一种虚拟机，所述虚拟机通过实现虚拟 CPU、虚拟内存、反汇编引擎、虚拟硬盘以及其他相关部分，如虚拟 BIOS(Basic Input Output System，基本输入输出系统)、虚拟 I/O 设备等，可以模拟实现主引导记录 MBR 的加载执行过程，并检测出是否存在 Bootkit 等恶意程序。而且，所述虚拟机既可以作为单独的工具，也可以作为动态库被其他程序调用，使用灵活。同时，考虑到性能和效率等实用性方面，整个虚拟机的实现控制在几百 K 字节内，是一种轻量级的虚拟机。

[0043] 当然，实施本申请的任一产品不一定需要同时达到以上所述的所有优点。

附图说明

[0044] 图 1 是现有技术中鬼影 3 中的代码示意图；

[0045] 图 2 是现有技术中鬼影的一个变种代码的示意图；

[0046] 图 3 是本申请实施例所述一种检测恶意程序的方法流程图；

[0047] 图 4 是本申请实施例所述一种检测恶意程序的装置结构图；

[0048] 图 5 是本申请另一实施例所述虚拟机的结构图；

[0049] 图 6 是本申请实施例中正常的 MBR 运行后的显示结果示意图；

[0050] 图 7 是本申请实施例中中了鬼影 1 后的 MBR 运行结果示意图；

[0051] 图 8 是本申请实施例中中了鬼影 3 后的 MBR 运行结果示意图；

[0052] 图 9 是本申请实施例中中了顶级 Bootkit 后的 MBR 运行结果示意图。

具体实施方式

[0053] 为使本申请的上述目的、特征和优点能够更加明显易懂，下面结合附图和具体实施方式对本申请作进一步详细的说明。

[0054] 对于恶意程序的检测，尤其是对基于 MBR 的 Bootkit 或类似于 Bootkit 的病毒等恶意程序的检测，本申请提出一种检测方法，无论这些恶意程序有何种变形，都可以被检测出来。

[0055] 下面首先介绍本申请提出的思路，如下：

[0056] 正常情况下，计算机系统的开机过程是：

[0057] 开机通电自检 --> 主板 BIOS 根据用户指定的启动顺序从软盘、硬盘或光驱进行启动 --> 系统 BIOS 将主引导记录 MBR 读入内存 --> 控制权交给主引导程序 --> 主引导程序检查分区表状态，寻找活动的分区 --> 主引导程序将控制权交给活动分区的引导记录，由引导记录加载操作系统启动文件。

[0058] 由上可知，MBR 是电脑通电开机，主板自检完成后，被第一个读取到的位置，位于硬盘的 0 磁头 0 磁道 1 扇区，它的大小是 512 字节，不属于任何一个操作系统，也不能用操作系统提供的磁盘操作命令来读取。

[0059] DOS 时代泛滥成灾的引导区病毒多寄生于 MBR 中。以鬼影病毒为例，该病毒寄生在 MBR 中，病毒释放的驱动程序，能够破坏大多数安全工具和系统辅助工具。当系统再次重启时，该病毒会早于操作系统内核先行加载。而当病毒成功运行后，在进程中、系统启动加载项里找不到任何异常。即使格式化重装系统，也无法将该病毒清除。

[0060] 鬼影病毒驻留在系统的高端内存中，因此现有技术通过搜索高端内存的特征码来定位其是否是病毒。例如，鬼影 3 中的代码如图 1 所示，通过搜索特征码 0X0413 来检测是否修改了高端内存来驻留内存。而对于变形的鬼影代码，参照图 2 所示的一个鬼影变种代码，这段代码可以达到和图 1 一样的修改高端内存的效果，但却通过指令的变形，找不到特征的代码，从而饶过常规安全软件的查杀。

[0061] 仔细分析图 1 和图 2 所示的鬼影病毒，可以发现，无论其是否变形，只要运行就能够达到修改高端内存的效果。因此，通过检测高端内存就可以检测出各种形式的鬼影病毒。本申请正是利用这一点，通过设置虚拟内存来模拟高端内存，并通过模拟 MBR 的加载执行过程来检测所述虚拟内存，从而在恶意程序真正运行之前就查找出各种基于 MBR 的 Bootkit（如鬼影病毒）或类似于 Bootkit 的病毒等恶意程序。

[0062] 下面通过实施例对本申请所述方法的实现流程进行详细说明。

[0063] 参照图 3，是本申请实施例所述一种检测恶意程序的方法流程图。

[0064] 参照上面的计算机系统开机过程，在系统开机后并且在加载操作系统文件之前，执行以下步骤：

[0065] 步骤 301，设置虚拟内存；

[0066] 即分配一块内存区域作为虚拟内存使用，所述虚拟内存是对实际的系统高端内存的模拟。

[0067] 步骤 302，读取主引导记录 MBR 并保存到所述虚拟内存；

[0068] 真实的开机过程中，开机通电自检后，系统 BIOS 将主引导记录 MBR 读入真实的高端内存；而本实施例中，是将 MBR 读入所述虚拟内存中。

[0069] 步骤 303，模拟执行虚拟内存中主引导记录 MBR 中的每一条指令，并在执行完每一条指令后检测所述虚拟内存是否被修改；

[0070] 如果被修改，则发现恶意程序，退出检测过程，并可以进行提示；否则，如果未被修改，则继续模拟执行下一条指令，直到主引导记录 MBR 的所有指令模拟执行完毕，退出检测过程。若 MBR 的所有指令都模拟执行完毕也没有发现恶意程序，则将控制权交给系统中真实的活动分区的引导记录，由引导记录加载操作系统启动文件。

[0071] 具体的，可以通过检测所述虚拟内存的大小是否改变来判断是否被修改，如果改变，则所述虚拟内存被修改；否则，未被修改。目前的实际应用中，由于 Bootkit 可修改内存使内存的大小变小，因此在模拟执行每一条指令之后，可通过判断虚拟内存的大小是否变小来进行检测。当然，本实施例也不排除其他判断内存是否被修改的方法。

[0072] 此外，本实施例进一步可以通过虚拟 CPU 来完成上述步骤 303，。具体的，在步骤 301 设置虚拟内存之前，先设置虚拟 CPU，然后在步骤 303 由所述虚拟 CPU 模拟指令的执行，并对虚拟内存进行检测。

[0073] 基于上述图 3 所示实施例的内容：

[0074] 进一步可选的，设置完虚拟内存后，还可设置虚拟硬盘，并将虚拟内存中的 MBR 拷贝到所述虚拟硬盘中，当模拟执行 MBR 中的指令时，直接从所述虚拟硬盘中进行读取。

[0075] 进一步可选的，为了整个模拟过程的完整性，还可以设置出虚拟 BIOS、虚拟 I/O 设备等其他相关部分，用于所述检测过程。

[0076] 进一步可选的，为了便于相关人员查看整个模拟过程的执行，还可以对主引导记录 MBR 中的每一条指令进行反汇编，并输出显示。

[0077] 综上所述，由于实际情况中，基于 MBR 的 Bootkit 或类似于 Bootkit 的病毒等恶意程序，即使进行了变形，也必须要驻留系统的高端内存，所以必然会修改高端内存，因此上述的检测方法通过设置虚拟内存来模拟高端内存，并通过检测虚拟内存是否被修改，就可以发现可疑的恶意程序，从而无视任何特征码变换技术，只要实际运行中发生了这个行为即可被检测出来。所述的检测方法在很大程度上可以检测出过去、现在和未来的基于 MBR 的 Bootkit。

[0078] 需要说明的是，对于前述的方法实施例，为了简单描述，故将其都表述为一系列的动作组合，但是本领域技术人员应该知悉，本申请并不受所描述的动作顺序的限制，因为依据本申请，某些步骤可以采用其他顺序或者同时进行。其次，本领域技术人员也应该知悉，说明书中所描述的实施例均属于优选实施例，所涉及的动作并不一定是本申请所必须的。

[0079] 基于上述检测方法的实施例，本申请还提供了相应的装置实施例。

[0080] 参照图 4，是本申请实施例所述一种检测恶意程序的装置结构图。

[0081] 所述检测装置可以包括：

[0082] 第一设置模块 41，用于设置虚拟内存；

[0083] 读取及保存模块 42，用于读取主引导记录 MBR 并保存到所述虚拟内存；

[0084] 模拟执行模块 43，用于模拟执行虚拟内存中主引导记录 MBR 中的每一条指令；

[0085] 检测模块 44，用于在所述模拟执行模块 43 执行完每一条指令后检测所述虚拟内存是否被修改，如果被修改，则发现恶意程序；否则，触发所述模拟执行模块 43 继续模拟执行下一条指令，直到主引导记录 MBR 的所有指令模拟执行完毕。

[0086] 优选的，所述检测模块 44 可通过检测所述虚拟内存的大小是否改变来判断是否被修改，如果改变，则所述虚拟内存被修改；否则，未被修改。

[0087] 进一步可选的，所述检测装置还可以包括：

[0088] 第二设置模块 45，用于设置虚拟 CPU，所述虚拟 CPU 可触发所述模拟执行模块 43 和检测模块 44 的执行。

[0089] 进一步可选的，所述检测装置还可以包括：

[0090] 第三设置模块 46，用于设置虚拟硬盘，并将虚拟内存中的主引导记录 MBR 拷贝到所述虚拟硬盘；

[0091] 此时，所述模拟执行模块 43 从所述虚拟硬盘读取主引导记录 MBR，并模拟执行主引导记录 MBR 中的每一条指令。

[0092] 进一步可选的，所述检测装置还可以包括：

[0093] 反汇编引擎 47，用于对主引导记录 MBR 中的每一条指令进行反汇编，并输出显示。

[0094] 所述检测装置既可以作为单独的工具，也可以作为动态库被其他程序调用，使用灵活。

[0095] 对于上述检测装置实施例而言，由于其与方法实施例基本相似，所以描述的比较简单，相关之处参见上述方法实施例的部分说明即可。

[0096] 基于上述内容，为了使本领域技术人员更加了解本申请的实现，本申请还提供了另一更具体的实施例，内容如下。

[0097] 实现一种虚拟机，所述虚拟机通过实现虚拟 CPU、虚拟内存、反汇编引擎、虚拟硬盘以及其他相关部分，如虚拟 BIOS、虚拟 I/O 设备等，可以模拟实现主引导记录 MBR 的加载执行过程，并检测出是否存在 Bootkit 等恶意程序。

[0098] 参照图 5，是本申请另一实施例所述虚拟机的结构图。

[0099] 具体的，所述虚拟机可以包括：

[0100] 虚拟 CPU 初始化模块 51，用于初始化虚拟 CPU54；

[0101] 虚拟内存初始化模块 52，用于初始化虚拟内存 53，并在初始化的过程中读取主引导记录 MBR 然后保存到所述虚拟内存 53；

[0102] 虚拟内存 53，用于存储主引导记录 MBR；

[0103] 虚拟 CPU54，用于模拟执行虚拟内存 53 中主引导记录 MBR 中的每一条指令，并在执行完每一条指令后检测所述虚拟内存是否被修改，如果被修改，则发现恶意程序；否则，继续模拟执行下一条指令，直到主引导记录 MBR 的所有指令模拟执行完毕。

[0104] 进一步可选的，所述虚拟机还可以包括：

[0105] 虚拟硬盘初始化模块 55，用于初始化虚拟硬盘 56，并在初始化的过程中将虚拟内存 53 中的主引导记录 MBR 拷贝到所述虚拟硬盘 56，所述虚拟 CPU54 从虚拟硬盘 56 读取主

引导记录 MBR 并模拟执行 ;

[0106] 虚拟硬盘 56,用于存储拷贝的主引导记录 MBR。

[0107] 进一步可选的,所述虚拟机还可以包括 :

[0108] 反汇编引擎 57,用于对主引导记录 MBR 中的每一条指令进行反汇编,并输出显示。

[0109] 由于实际应用中 Bootkit 病毒多运行于计算机系统的实模式下,因此下面将以实模式下的虚拟机为例进行详细说明。当然,所述虚拟机可应用于保护模式或其他计算机模式下,本实施例不受此限定。

[0110] 在实模式下,上述虚拟 CPU 可模拟实现所有实模式下的指令,主要为 8086 指令,还可以包括 386 以后的指令。此外,随着木马等 Bootkit 技术的发展,如果木马调用了特殊的指令,所述虚拟 CPU 还可以进行相应特殊指令的模拟。例如,魅影病毒为了防止被调试采用了 586 以后才支持的指令 RDTSC,则虚拟 CPU 还会模拟所述 RDTSC 指令。

[0111] 在开机通电自检后,系统 BIOS 读取系统内置的 MBR,然后传给所述虚拟机,同时虚拟机进行初始化。虚拟机的执行步骤如下 :

[0112] 步骤 1,初始化虚拟 CPU ;

[0113] 首先进行虚拟 CPU 初始化,虚拟 CPU 为单核的 80x86,支持的寄存器同真实的机器。

[0114] 此外,支持指令的初始化,虚拟机内部有个指令支持列表,该列表可根据实际需要,不断增加需要支持的指令,初始化的时候会填充已经支持的指令列表。然后,将虚拟机中的 CPU 的指令指针指向虚拟内存中的 BIOS 指令开始执行处(即 MBR 指令)0xf000:0xfff0。

[0115] 步骤 2,初始化虚拟内存 ;

[0116] 可通过初始化 BIOS 数据区,所述 BIOS 数据区保存常规的虚拟内存的大小,在实模式下可分配 640KB 左右的内存空间作为虚拟内存使用。然后,将系统 BIOS 读取的 MBR 存入所述虚拟内存中。

[0117] 步骤 3,初始化虚拟硬盘以及虚拟机的其他相关部分 ;

[0118] 一般在 DOS 下只使用 1MB 的磁盘,所以通过分配 1MB 左右的内存空间作为虚拟硬盘来模拟实际的硬盘。相应的,还可通过访问所述虚拟硬盘来模拟访问实际的硬盘。然后,将虚拟内存中的 MBR 拷贝到所述虚拟磁盘的开始处。

[0119] 此外,还会初始化虚拟机的其他相关部分,如虚拟 BIOS、虚拟 I/O 设备等。

[0120] 步骤 4,运行虚拟 CPU ;

[0121] 与实际的执行过程类似,开始执行虚拟 CPU 后,从 BIOS 指令开始执行处(即 MBR 指令)开始执行。每执行 MBR 的一条指令,虚拟 CPU 会进行指令译码,根据指令译码结果修改内部的寄存器和相关内存,并执行相应的流程。如果指令中包含病毒,则虚拟 CPU 不仅会修改内部的寄存器,还会修改相关的虚拟内存 ;如果不包含,则不会修改相关的虚拟内存。

[0122] 步骤 5,虚拟 CPU 检测虚拟内存。

[0123] 每执行一条指令后,虚拟 CPU 会检测前面设置的 BIOS 数据区保存的虚拟内存大小,如果发现被改变了,就认为发现了可疑 MBR 病毒,然后退出虚拟机,并进行提示。如果没有发现则继续执行,如果发现执行到 CS=0,IP=0X7C00 的时候就检测是否执行到了操作系统引导区,比如是否是 NTFS 或 FAT 的文件系统的引导区,如是的,则认为执行 MBR 结束,没有发现可疑的 MBR,然后退出。

[0124] 需要说明的是,上述步骤的先后顺序可根据实际需要进行调整,本申请并不受所

描述的动作顺序的限制，因为依据本申请，某些步骤可以采用其他顺序或者同时进行。

[0125] 上述虚拟机既可以作为单独的工具，也可以作为动态库被其他程序调用，使用灵活。同时，考虑到性能和效率等实用性方面，整个虚拟机的实现控制在几百 K 字节内，是一种轻量级的虚拟机。

[0126] 下面以鬼影病毒为例说明本申请的上述内容。

[0127] 参照图 6，是本申请实施例中正常的 MBR 运行后的显示结果示意图；

[0128] 参照图 7，是本申请实施例中中了鬼影 1 后的 MBR 运行结果示意图；

[0129] 参照图 8，是本申请实施例中中了鬼影 3 后的 MBR 运行结果示意图；

[0130] 参照图 9，是本申请实施例中中了顶级 Bootkit 后的 MBR 运行结果示意图。

[0131] 其中，

[0132] MbrVmConsole 为主程序；

[0133] MbrVM.ini 为配置文件，用来指定 VM 内存大小和指定虚拟硬盘的文件；

[0134] Mbr.img 为指定的虚拟硬盘文件；

[0135] BIOS 下为虚拟机用到的虚拟 BIOS 文件；

[0136] 使用时候，将指定的 MBR DUMP 文件 mbr.bin 拷贝到虚拟机目录下，运行 MbrVmconsole 即可运行虚拟机进行检测。

[0137] mbrGood.bin 为正常 MBR；

[0138] mbrguiying1.bin 为鬼影 1MBR；

[0139] mbrguiying3.bin 为鬼影 3MBR；

[0140] mbrTdl.bin 为 TDL4 MBR；

[0141] 运行时，将相应的文件改名为 mbr.bin 即可进行相应的检测。

[0142] 综上所述，随着现在杀毒软件技术的日益成熟，木马等病毒生存的空间越来越狭小，传统的木马技术，已经很难生存和发展了。但 Bootkit 技术的出现，给病毒一个很大的生存发展空间，使其可以做到无文件、无进程、无注册表修改等任何杀软能检测到的东西，只需要在 MBR 里写入加载代码，就可以加载起一个完整的病毒执行体系。而且即使格式化重装，也照样能复活。所以基于本申请所实现的方法和装置，在以后检测该方面的木马等病毒中起到决定性的作用，而这正是目前机会所有杀毒软件的盲点。

[0143] 本说明书中的各个实施例均采用递进的方式描述，每个实施例重点说明的都是与其他实施例的不同之处，各个实施例之间相同相似的部分互相参见即可。

[0144] 还需要说明的是，在本文中，诸如第一和第二等之类的关系术语仅仅用来将一个实体或者操作与另一个实体或操作区分开来，而不一定要求或者暗示这些实体或操作之间存在任何这种实际的关系或者顺序。

[0145] 以上对本申请所提供的一种检测恶意程序的方法、装置及虚拟机，进行了详细介绍，本文中应用了具体个例对本申请的原理及实施方式进行了阐述，以上实施例的说明只是用于帮助理解本申请的方法及其核心思想；同时，对于本领域的一般技术人员，依据本申请的思想，在具体实施方式及应用范围上均会有改变之处，综上所述，本说明书内容不应理解为对本申请的限制。

```
seg000:0025 66 60                       pushad
seg000:0027 1E                          push    ds
seg000:0028 2E 8B 1E 13 04              mov     bx, cs:word_413
seg000:002D 81 EB 0D 00                 sub     bx, 0Dh
seg000:0031 80 E3 FC                    and     bl, 0FCh
seg000:0034 2E 89 1E 13 04              mov     cs:word_413, bx
seg000:0039 C1 E3 06                    shl     bx, 6
seg000:003C 8E C3                       mov     es, bx
seg000:003E 31 DB                       xor     bx, bx
seg000:0040 B8 01 02                    mov     ax, 201h
seg000:0043 B9 01 00                    mov     cx, 1
seg000:0046 BA 80 00                    mov     dx, 80h ; '€'
seg000:0049 CD 13                       int     13h             ; DISK - READ SECTORS INTO MEMORY
seg000:0049                                                     ; AL = number of sectors to read, CH = track, CL = sector
seg000:0049                                                     ; DH = head, DL = drive, ES:BX -> buffer to fill
seg000:0049                                                     ; Return: CF set on error, AH = status, AL = number of sectors
```

图 1

```
seg000:0025                     mov     si, 533h
seg000:0028                     xor     si, 120h
seg000:002C                     lodsw
seg000:002D                     sub     si, 2
seg000:0030                     shl     ax, 6
seg000:0033                     and     ax, 0FFFh
seg000:0036                     shr     ax, 6
seg000:0039                     sub     [si], ax
```

图 2

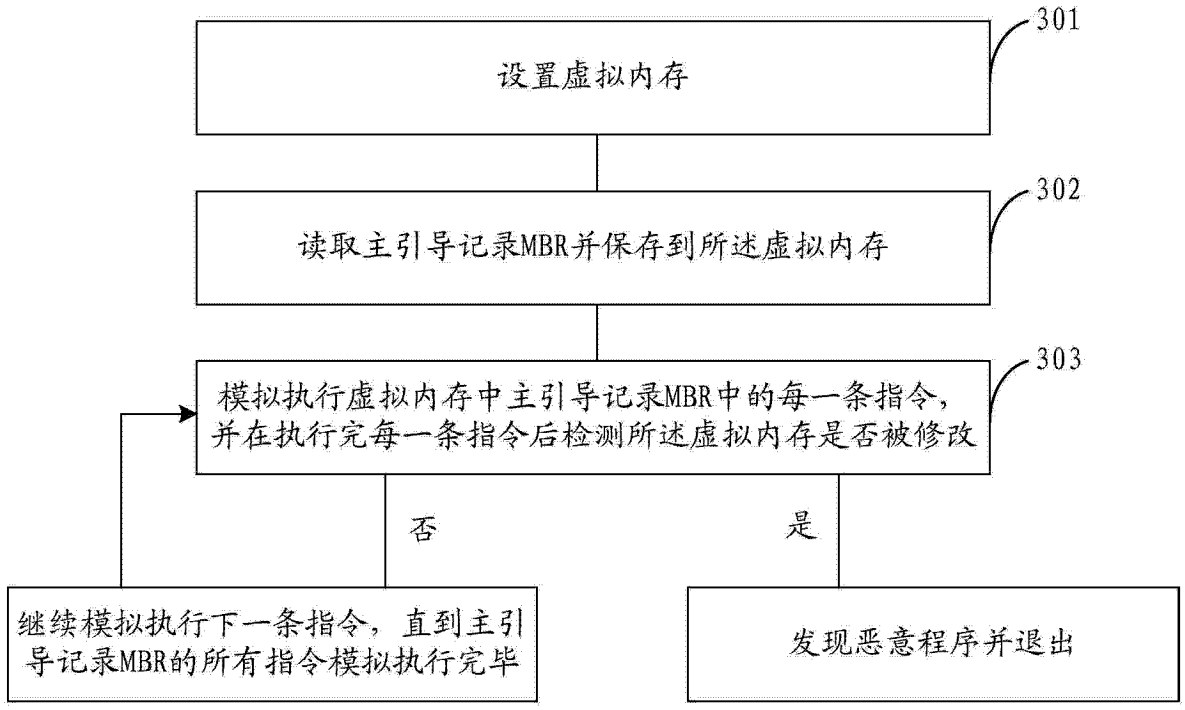


图 3

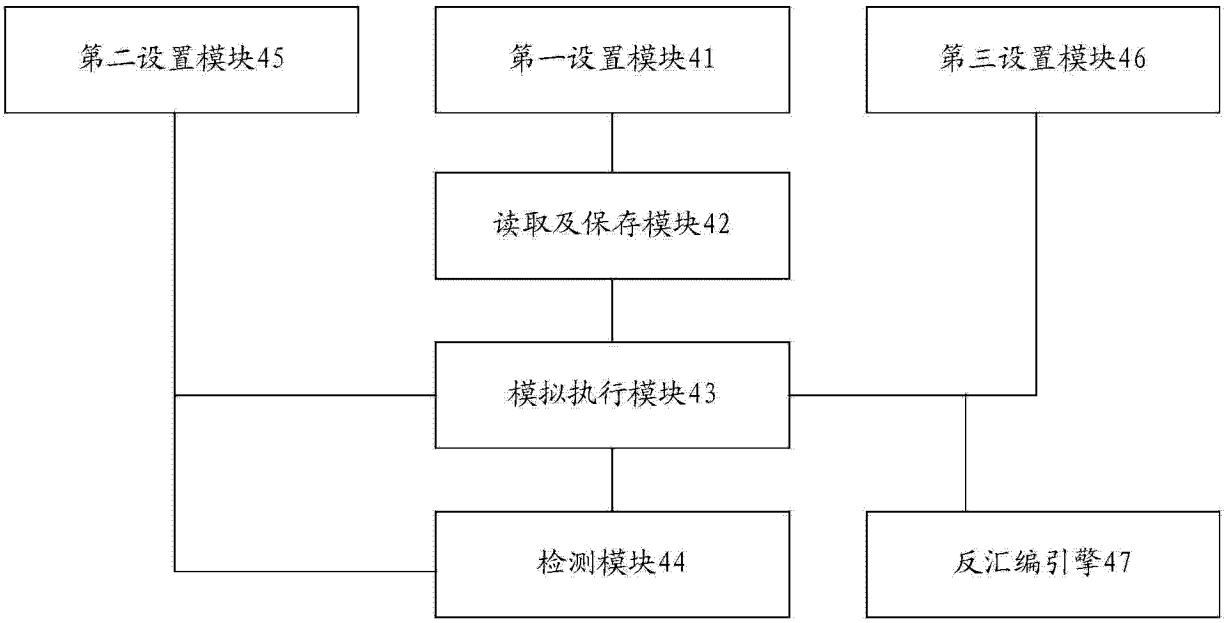

第二设置模块45
第一设置模块41
第三设置模块46
读取及保存模块42
模拟执行模块43
检测模块44
反汇编引擎47

图 4

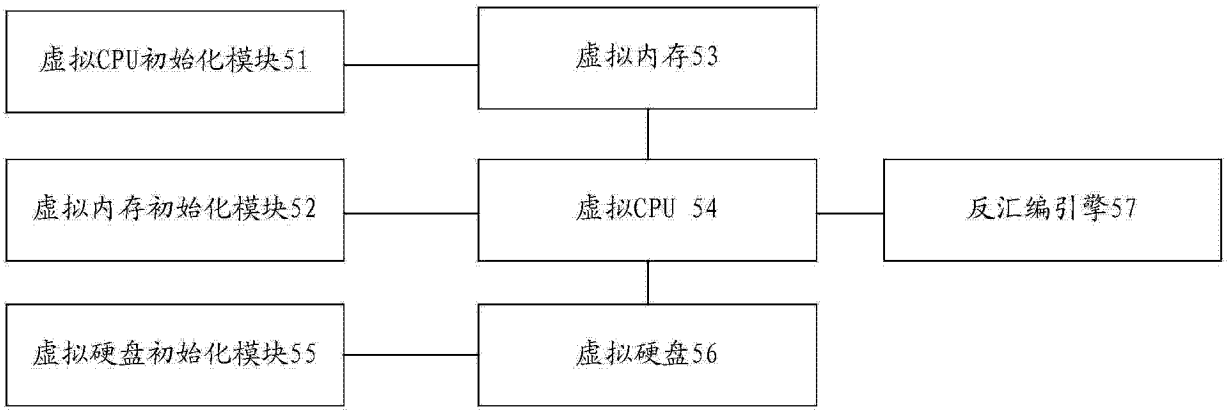

虚拟CPU初始化模块51
虚拟内存53
虚拟内存初始化模块52
虚拟CPU 54
反汇编引擎57
虚拟硬盘初始化模块55
虚拟硬盘56

图 5


F:\MbrVmConsole\MbrVmConsole.exe
0:6da mov al,0xff
0:6dc out 0x64,al
0:6de call 0x78
0:756 sub cx,cx
0:758 in al,0x64
0:75a jmp 0x2
0:75c and al,0x2
0:75e loopn 0xfffffffa
0:760 and al,0x2
0:762 ret
0:6e1 sti
0:6e2 mov ax,0xbb00
0:6e5 int 0x1a
f000:90 out 0x73,al
f000:92 iret
0:6e7 and eax,eax
0:6e7 and ax,ax
0:6ea jnz 0x3d
0:727 pop dx
0:728 xor dh,dh
0:72a jmpf 0x0:0x7c00
boot filesystem: NTFS
execute over mbr code can exit!
high mem not modified!
MBR V1
高端内存正常，正常MBR!
确定

图 6


F:\MbrVmConsole\MbrVmConsole.exe
f000:1b9 mov bp,ax
f000:1bb mov ax,0xaa55
f000:1be jmpf 0x0:0x7c00
0:7c00 jmp 0x12
0:7c12 cli
0:7c13 xor bx,bx
0:7c15 mov ss,bx
0:7c17 mov ss:[0x7bfe],sp
0:7c17 mov [0x7bfe],sp
0:7c1c mov sp,0x7bfe
0:7c1f push ds
0:7c20 pusha
0:7c20 pusha
0:7c22 cld
0:7c23 mov ds,bx
0:7c25 mov si,0x533
0:7c28 xor si,0x120
0:7c2c lodsd
0:7c2d sub si,0x2
0:7c30 shl ax,0x6
0:7c33 and ax,0xfff
0:7c36 shr ax,0x6
0:7c39 sub [si],ax
high mem modified!
MBR VM
高端内存被修改，可疑MBR!
确定

图 7

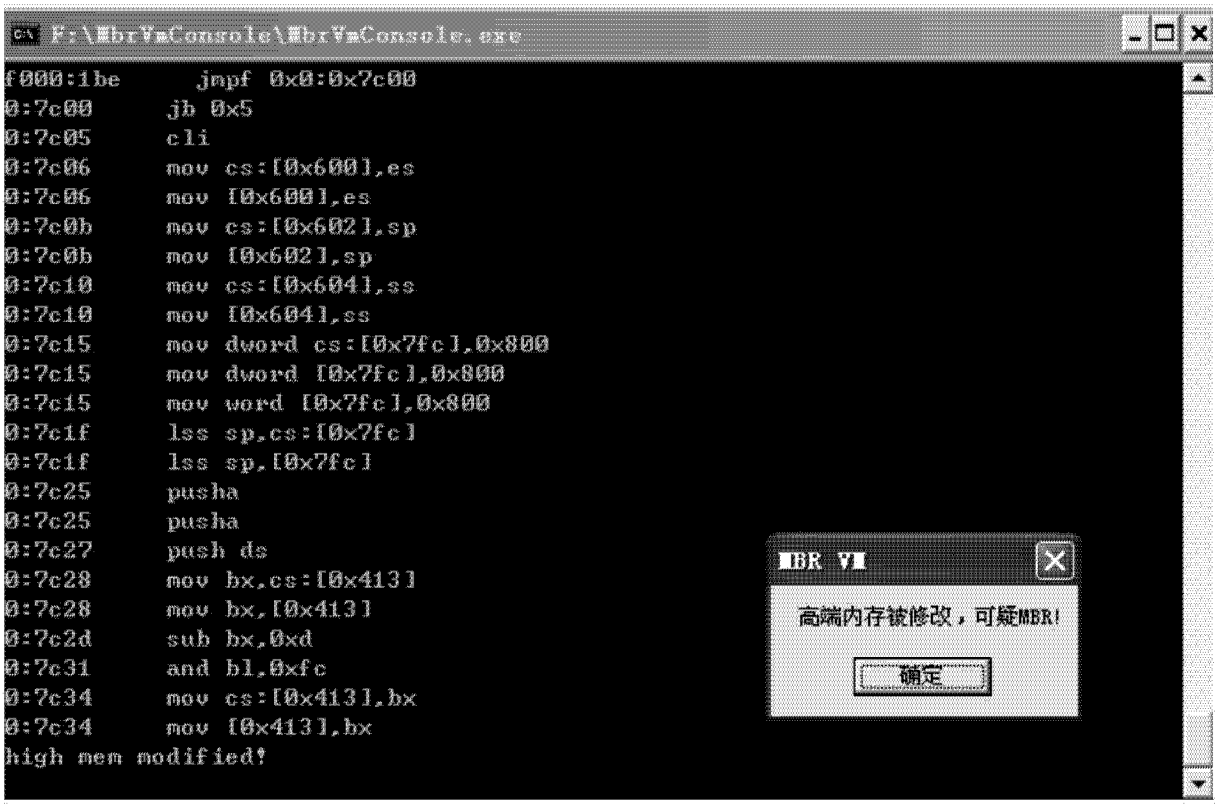

F:\MbrVmConsole\MbrVmConsole.exe
f000:1be jmpf 0x0:0x7c00
0:7c00 jb 0x5
0:7c05 cli
0:7c06 mov cs:[0x600],es
0:7c06 mov [0x600],es
0:7c0b mov cs:[0x602],sp
0:7c0b mov [0x602],sp
0:7c10 mov cs:[0x604],ss
0:7c10 mov [0x604],ss
0:7c15 mov dword cs:[0x7fc],0x800
0:7c15 mov dword [0x7fc],0x800
0:7c15 mov word [0x7fc],0x800
0:7c1f lss sp,cs:[0x7fc]
0:7c1f lss sp,[0x7fc]
0:7c25 pusha
0:7c25 pusha
0:7c27 push ds
0:7c28 mov bx,cs:[0x413]
0:7c28 mov bx,[0x413]
0:7c2d sub bx,0xd
0:7c31 and bl,0xfc
0:7c34 mov cs:[0x413],bx
0:7c34 mov [0x413],bx
high mem modified!
MBR V1
高端内存被修改，可疑MBR!
确定

图 8


F:\MbrVmConsole\MbrVmConsole.exe
0:624 ror [bp+0x0],cl
0:627 inc bp
0:628 loop 0xfffffffc
0:624 ror [bp+0x0],cl
0:627 inc bp
0:628 loop 0xfffffffc
0:624 ror [bp+0x0],cl
0:627 inc bp
0:628 loop 0xfffffffc
0:624 ror [bp+0x0],cl
0:627 inc bp
0:628 loop 0xfffffffc
0:624 ror [bp+0x0],cl
0:627 inc bp
0:628 loop 0xfffffffc
0:624 ror [bp+0x0],cl
0:627 inc bp
0:628 loop 0xfffffffc
0:62a mov [0x7b2],dl
0:62e sub word [0x413],0x10
high mem modified!
MBR V1
高端内存被修改，可疑MBR!
确定

图 9