发 明 专 利

申请号 200910043831.6
申请日 2009.07.03
专利权人 中国人民解放军国防科学技术大学
地址 410073 湖南省长沙市芙蓉区正街 47 号
发明人 张卫华 刘晓建 易晓东 谭郁松
吴庆波 戴洪林 孔宪柱
专利代理机构 国防科技大学专利服务中心
代理人 郭敏

权利要求书 4 页 说明书 9 页 附图 4 页

发明名称
基于块对目标代码扫描并替换敏感指令的方法

摘要
本发明公开了一种基于块对目标代码扫描并替换敏感指令的方法，目的是提供一种当敏感指令长度小于 JMP 时，寻找包含该敏感指令且长度不小于 JMP 的最小目标代码块，并用 JMP 替换该最小目标代码块的方法。技术方案是针对首个扫描入口地址进行动态扫描，主要包括对基本块起始边界进行基本块扫描，对待转换敏感指令队列中的敏感指令进行基于替换块的敏感指令本地替换；然后从首个扫描入口地址开始执行 GOS 内核代码，直至产生硬件中断或异常或执行间接控制转移指令的替换代码；将 GOS 中断或异常处理函数的入口地址或间接控制转移指令的目标地址设为首个扫描入口地址，然后循环执行动态扫描。采用本发明能够最大限度地避免使用 INTO/INT3 指令进行替换，有效提高 BT 虚拟化性能。

对比文件
WO 2009075281 A1, 2009.06.18,
陈龙等，二进制翻译中解析多目标分支语句的图匹配方法，《计算机研究与发展》，2008 年第 10 期，789-1798。
审查员 张一良
1. 一种基于块对目标代码扫描并替换敏感指令的方法，其特征在于包括以下步骤：
步骤 1. 创建并初始化全局数据结构：
步骤 1.1 创建并初始化扫描信息位图页目录表；分配一个数据表作为扫描信息位图索引结构中的页目录表，并将页目录表中的页目录项全部初始化为 0；扫描信息位图是指以位图形式实现的，用于保存 GOS 内核虚地址扫描状态及相关信息的数据结构；在扫描信息位图中，每条 GOS 虚地址的扫描信息由与该地址一一对应的扫描信息位元被称为位元进行描述，每个扫描信息位元由二个比特位组成：00 表示该位元对应的虚地址未扫描或非指令首地址；01 表示该位元对应的虚地址已扫描；10 表示该位元对应的虚地址未扫描或非指令首地址且为基本块边界或控制转移指令目标地址；11 表示该位元对应的虚地址已扫描且为基本块边界或控制转移指令目标地址；基本块是指以 GOS 起始执行地址、GOS 中断处理函数首地址或控制转移指令目标地址为起始边界，以无条件控制转移指令或已扫描地址为结束边界的连续目标指令序列；扫描信息位图索引结构由扫描信息位图页目录表、扫描信息位图页表页和扫描信息位图页图组成，GOS 内核虚地址划分为页目录索引、页表索引和页内偏移三个位段，页目录表索引存放页表页的首地址，页表索引存放位图页的首地址，页内偏移存放内核虚地址对应的扫描信息位元在位图页表中的偏移量；
步骤 1.2 初始化扫描入口地址队列及已替换 RB 有效信息平衡二叉树；将待扫描入口地址队列及已替换 RB 有效信息平衡二叉树的头指针设为 NULL；待扫描入口地址队列是指以线性链表形式实现的，保存动态扫描时发现的控制转移指令目标地址及其调用地址的数据库，待扫描入口地址队列节点以目标地址为索引值，按地址大小按序排列；已替换 RB 有效信息平衡二叉树是指以平衡二叉树 AVL 形式实现的，保存被替换 RB 相关信息的数据结构，树中节点以被替换 RB 首地址为索引值，其它信息包括被替换 RB 的长度及其与对应 TB 中每条指令地址的对应关系；RB 是替换块，指满足以下条件的最短连续目标指令序列：位于基本块内部，包括基本块自身，包含指定敏感指令且长度不小于 JMP 指令长度；不含所属基本块范围之内、本替换块范围之外任何控制转移指令的目标地址，本替换块首地址除外；TB 是替换块，指替换块被替换之后，经由替换指令执行、包含虚拟化代码或安全检查代码且与原替换块具有等价执行语义的一组指令序列；
步骤 1.3 令全局变量 M = JMP 替换指令长度；
步骤 2. 从 GOS 内核加载程序输入参数中获取 GOS 内核起始执行地址，记为首个扫描入口地址，然后对首个扫描入口地址进行动态扫描：
步骤 2.1 采用扫描状态查询方法检查首个扫描入口地址的扫描状态；若已扫描，转步骤 3；若未扫描，将首个扫描入口地址记为基本块起始边界，转 2.2；
步骤 2.2 对基本块起始边界进行基本块扫描，方法是：
步骤 2.2.1 初始化待替换敏感指令队列；将待替换敏感指令队列头指针设为 NULL；待替换敏感指令队列是指以线性链表形式实现的，保存基本块扫描时发现的敏感指令地址及其相关信息的数据结构，待替换敏感指令队列节点以敏感指令地址为索引值，按地址大小按序排列；
步骤 2.2.2 将基本块起始地址设为待扫描地址；
步骤 2.2.3 待待扫描地址处指令进行译码，根据译码结果执行以下操作；
步骤 2.2.3.1 若为敏感指令或间接控制转移指令，则将该指令添加到待替换敏感指令队列，
2.2.3.2 若为直接控制转移指令，则首先采用目标地址有效性检查方法检查该目标地
址是否指向已替换 RB 内部；若指向上已替换 RB 内部，则将直接控制转移指令添加到待转换
敏感指令队列，然后转 2.2.4；若不指向已替换 RB 内部，则采用扫描状态查询方法检查目标
地址的扫描状态，若目标地址已扫描则转 2.2.4，否则将目标地址添加到待扫描入口地址队
列，转 2.2.4；

2.2.3.3 若为普通指令，则转 2.2.4；

2.2.4 根据译码结果记录被扫描指令的扫描状态，方法为：首先采用扫描信息位元索
引方法得到被扫描指令首地址对应位元的实际位置，将该位元的低位设为 1；然后采用扫
描信息位元索引方法依次得到被扫描指令所有剩余字节对应位元的实际位置，将这些位元
设为 00；最后若被扫描指令为直接控制转移指令，则采用扫描信息位元索引方法得到该控
制转移指令目标地址对应的位元的实际位置，并将该位元的高位设为 1；

2.2.5 将被扫描指令地址加上该指令长度，得到下一条指令的首地址；

2.2.6 若被扫描指令为无条件控制转移指令或下一条指令首地址已扫描，则将下一条
指令首地址设为基本块的结束边界，并将其扫描信息位元的高位设为 1，表示该地址为基本
块边界地址或控制转移指令目标地址，然后结束本次基本块扫描并转 2.3；否则转 2.2.7；

2.2.7 将下一条指令首地址设为待扫描地址，转 2.2.3，继续进行基本块扫描；

2.3 对待转换敏感指令队列中的敏感指令进行基于替换块的敏感指令本地替换；

2.3.1 若待转换敏感指令队列为空，则停止替换，转步骤 3；若待转换敏感指令队列不
为空，从待转换敏感指令队列中取出最小的敏感指令地址，在基本块内寻找包含该敏感指
令的替换块，寻找方法如下：

2.3.1.1 将该敏感指令自身设为当前 RB；

2.3.1.2 若当前 RB 长度不小于 M 字节，则将当前 RB 设为待转换 RB，转 2.3.3；若当前
RB 长度小于 M 字节，尝试向低地址空间扩展相邻指令到当前 RB，扩展方法如下：

A1. 若当前 RB 首地址为基本块起始边界，转 2.3.1.3；否则向低地址空间扩展一条指令
到当前 RB，转 B1；

B1. 若该低位为 1 的位元的高位为 1，执行 C1，否则转 2.3.1.2；

C1. 检查待扫描入口地址队列，若发现以低位为 1 位元对应的虚地址为索引值的节点，
则采用遍历扩展方法检查该节点的调度器队列，将所有位于基本块范围之内的调度器地址
与当前 RB 之间的所有指令扩展到当前 RB，转 2.3.1.2；若未发现以低位为 1 位元对应的虚
地址为索引值的节点，则转 2.3.1.2，调度器队列是指以线性链表形式实现的、保存具有相
同目标地址的控制转移指令地址的数据结构，调度器地址队列节点以控制转移指令地址为
索引值，包含在待扫描入口地址队列的每个节点之中；

2.3.1.3 尝试向高地址空间扩展相邻指令到当前 RB，扩展方法如下：

A2. 若当前 RB 结束边界为基本块的结束边界，转 2.3.2，否则向高地址空间扩展一条基
本块指令到当前 RB；

B2. 若被扩展指令首地址对应的位元为 10，转 2.3.2；否则检查被扩展指令首地址对应
位元的高位，若高位为 1，转 C2，若高位为 0，转 2.3.1.2；

C2. 检查待扫描入口地址队列，若发现以被扩展指令首地址为索引值的节点，则采用遍
历扩展方法检查该节点的调用者队列，将所有位于基本块范围之内的调用者地址与当前 RB 之间的所有指令扩展到当前 RB，转 2.3.1.2，若未发现以被扩展指令首地址为索引值的节点，直接转 2.3.1.2；

2.3.2 若当前 RB 长度小于 M 字节，则将敏感指令设为待转换 RB；否则将当前 RB 设为待转换 RB；

2.3.3 采用二进制翻译中的常规方法生成与待转换 RB 具有等价执行语义的转换代码块，记为 TB；

2.3.4 若待转换 RB 的长度不小于 M 字节，则将该 RB 前 M 字节替换为 JMP 指令，目标地址为 TB 首地址，剩余字节用 NOP 指令填充，得到已替换 RB；若待转换 RB 的长度小于 M 字节，则将该 RB 的首字节替换为 INT3 指令，剩余字节用 NOP 指令填充，得到已替换 RB；

2.3.5 在已替换 RB 有效信息平衡二叉树中创建已替换 RB 的节点，该节点包括已替换 RB 的首地址、长度及其与 TB 指令之间的地址对应关系这些有效信息；

2.3.6 遍历待扫描敏感指令队列，删除所有与已替换 RB 中的敏感指令地址为索引值的节点，转 2.3.1；

2.4 遍历待扫描入口地址队列，删除所有与已扫描入口地址为索引值的节点；

2.5 若待扫描入口地址队列为空，则结束本次动态扫描，转步骤 3；否则从待扫描入口地址队列中取出最大未扫描入口地址，记为基本块起始边界，转 2.2 执行下一次基本块扫描；

步骤 3 从首个扫描入口地址开始执行 Gos 内核代码，直至产生硬件中断或异常或执行间接控制转移指令的转换代码；

步骤 4 将 Gos 中断或异常处理函数的入口地址或间接控制转移指令的目标地址设为首个扫描入口地址，转 2.1。

2. 如权利要求 1 所述的基于块对目标代码扫描并替换敏感指令的方法，其特征在于所述扫描模式查询方法是：首先将被查询地址划分为页目录索引、页表索引和页内偏移三个位段；然后检查页目录索引指向的页目录项。若页目录项为 0，则返回 -1 表示该地址未扫描且该地址对应位置不存在。若页目录项不为 0，则继续检查页表索引指向的页表项，若页表项为 0，则返回 -1 表示该地址未扫描且该地址对应位置不存在。若页表项不为 0，则根据页表项中的存储信息位图页首地址和页内偏移计算得到被查询地址对应位置的实际位置并返回页首的实际值；最后根据页首的实际值确定被扫描指令的首地址的扫描状态。位首的实际值高低位为 1 表示已扫描，为 0 表示未扫描；

3. 如权利要求 1 所述的基于块对目标代码扫描并替换敏感指令的方法，其特征在于所述目标地址有效性检查方法为：采用平衡二叉树遍历方法遍历已替换 RB 有效信息平衡二叉树，找到索引值小于指定目标地址且与指定目标地址最接近的节点。若指定目标地址小于或等于已替换 RB 的首地址加上节点中记录的已替换 RB 长度，则表明指定目标地址指向已替换 RB 内部，否则不指定已替换 RB 内部。

4. 如权利要求 1 所述的基于块对目标代码扫描并替换敏感指令的方法，其特征在于所述存储信息位元索引方法为：首先将被索引地址划分为页目录索引、页表索引和页内偏移三个位段；然后检查页目录索引指向的页目录项是否为 0。是则立即分配一个数据页作为扫描信息位图页表页并将该页表页的首地址写入页目录页中对应的页目录项中；接
查页表项指示的页表项是否为 0，是则立即分配一个数据页作为扫描信息位图页并将该页图的首地址写入页表项中对应的页表项中，最后根据扫描信息位图页首地址和页内偏移得到被索引地址对应位元的实际位置。

5. 如权利要求 1 所述的基于块对目标代码扫描并替换敏感指令的方法，其特征在于所述向低地址空间扩展一条基本块指令到当前 RB 的方法是：采用扫描信息位元索引方法从当前 RB 首地址开始向低地址空间扫描扫描信息位图，依次得到每条连续地址对应位元的实际位置并检查每个位元的实际值，找到第一个或最后一个为 1 的位元，将该位或最后一个为 1 的位元对应的虚地址即被扩展指令的首地址设为当前 RB 起始边界。

6. 如权利要求 1 所述的基于块对目标代码扫描并替换敏感指令的方法，其特征在于所述遍历扩展方法为：采用目标地址有效性检查方法依次检查每个调用者地址是否位于基本块范围之内且位于当前 RB 范围之外，是则将调用者地址设为当前 RB 的边界，即若调用者地址小于当前 RB 首地址，则将该地址设为当前 RB 首地址，若调用者地址大于当前 RB 的结束边界，则将该地址设为当前 RB 的结束边界，否则检查下一个调用者地址。

7. 如权利要求 1 所述的基于块对目标代码扫描并替换敏感指令的方法，其特征在于所述向高地址空间扩展一条基本块指令到当前 RB 的方法是：采用扫描信息位元索引方法从当前 RB 结束边界开始向高地址空间扫描扫描信息位图，依次得到每条连续地址对应位元的实际位置并检查每个位元的实际值，找到第二个或最后一个为 1 的位元或第一个被设置为 10 的位元，将该位元对应的虚地址设为当前 RB 结束边界，而找到的第一个或最后一个为 1 的位元对应的虚地址即被扩展指令首地址。
基于块对目标代码扫描并替换敏感指令的方法

技术领域
[0001] 本发明涉及面向X86体系结构操作系统虚拟化的二进制翻译（Binary Translation，BT）方法，尤其是对目标代码进行扫描并对扫描中发现的敏感指令进行替换的方法。

背景技术
[0002] 近年来，随着硬件技术的迅猛发展尤其是多核乃至众核处理器的出现，提高计算机系统高速增长计算能力的使用率日益成为迫切的市场需求。操作系统虚拟化技术通过在同一台物理机器上同时运行多个操作系统并对外提供服务，能够有效提高服务器的利用率，因而受到各研究机构和商业机构的高度重视，并在服务器整台、计算机安全、虚拟数据中心等领域得到广泛应用。

[0003] 操作系统虚拟化依赖于一种称为虚拟监控器（Virtual Machine Monitor，VMM）的系统软件，被虚拟化的操作系统则称为客户操作系统（Guest OS，GOS）。目前，VMM软件主要基于三种方法实现操作系统虚拟化，即硬件辅助虚拟化、半虚拟化和BT虚拟化。三种方法的基本原理及特点简述如下：

[0004] 1）硬件辅助虚拟化要求处理器提供硬件支持，如Intel的VT技术和AMD的SVM技术。VMM通过捕获在特殊处理器状态下运行的客户操作系统执行敏感指令时触发的硬件异常来实现操作系统虚拟化，如XenSource公司的虚拟机监视器产品Xen和开源软件KVM等。该方法的优点是实现简单，缺点是需要在处理器普通状态与虚拟化状态之间频繁切换，并且切换的系统开销远大于普通系统调用。

[0005] 2）半虚拟化要求客户操作系统降特权级（如Ring1）运行，并对其源代码相应部分进行修改以配合VMM实现虚拟化。该方法的局限性在于必须获得GOS源代码，并且需要对任何希望虚拟化的操作系统进行修改，移植和维护工作量巨大。Xen是采用半虚拟化技术的代表（同一VMM可对不同GOS采用不同的虚拟化方法）。

[0006] 3）BT虚拟化则通过对GOS内核二进制代码进行扫描，发现其中的敏感指令，并将其替换为相应的虚拟化代码来实现GOS虚拟化。采用BT虚拟化方法的VMM有VMWare公司的VMWare Workstation和Sun公司的VirtualBox。

[0007] 与前两种方法相比，BT虚拟化对软件等外部条件的依赖最小——既不需要硬件支持，也不需要获得客户操作系统源代码，其关键就在于通过扫描发现内核二进制代码中的敏感指令，并将其替换为包含相应虚拟化功能的代码且保持GOS原有执行语义不变的一段指令序列。所谓“敏感指令”泛指所有一旦在真实处理器上执行即可能改变处理器的寄存器状态，从而导致VMM或其它客户操作系统无法正常运行的指令或基于安全隔离目的需要VMM介入的指令。

[0008] 敏感指令替换方法有指令缓存和本地替换两种，指令缓存方法不修改内存中的GOS代码，而是在指令缓冲区内复制、替换即将执行的目标代码块，并在该缓冲区内执行转换后的目标代码块，如VMWare和开源软件Qemu。本地替换方法则直接在敏感指令所在内
存位置将其替换为一条转移指令，通常为 JMP 指令，使目标代码改变流程去执行一段与敏感指令原有语义等价的简单化功能代码或安全检查代码，以达到虚拟化或安全隔离目的。与指令缓存方法相比，本地替换方法具有简单直观，内存占用量小等优点，但是采用该方法时，将不可避免地遇到部分敏感指令长度小于替换指令即短敏感指令的情形。例如：在 X86 体系结构中，为了能够直接跳转到任意虚拟地址空间，通常使用 5 字节或更长的 JMP 跳转指令作为替换指令，而某些敏感指令，如 POPF 指令，只有 1 字节长度，无法直接替换成 JMP 指令。

针对这一情形，目前已知的基于本地替换方法的 VMM 软件均采用可触发硬件异常的软中断指令（INTO 或 INT3 指令）进行替换，如 Sun 公司的 VirtualBox 采用 1 字节长度的 INT3 指令 (0xCC) 替换所有长度小于 JMP 指令的敏感指令。但是，采用 INTO/INT3 指令替换的主要缺点在于：每次执行该指令时都会触发硬件异常，尤其是当被替换指令被频繁调用时，硬件状态的反复切换将导致大量的系统开销。因此，当敏感指令长度小于 JMP 指令长度时，如何寻找包含该敏感指令且长度不小于 JMP 指令的最小目标代码块，并将该最小目标代码块替换为 JMP 指令，以减少使用 INTO3 指令替换的频率，同时保持目标代码原有执行语义不变，是本领域技术人员关注的重要问题。

发明内容

本发明要解决的技术问题为：提供一种基于块对目标代码扫描并替换敏感指令的方法，当敏感指令长度小于 JMP 跳转指令时，寻找包含该敏感指令且长度不小于 JMP 指令的最小目标代码块，并用 JMP 指令替换该最小目标代码块，同时保持最小目标代码块原有执行语义不变。包含指定敏感指令长度不小于 JMP 指令的最小目标代码块称为该敏感指令的替换代码块，简称替换块。

为了准确描述本发明，首先给出如下说明及定义：

在 X86 体系结构中，所有控制转移指令包括：JCC、JMP、CALL、RET 和 IRET 指令。这些控制转移指令可划分为两类：有条件转移指令和无条件转移指令。除全部 JCC 指令以外，JMP/CALL/RET/IRET 指令都是无条件转移指令。所有控制转移指令又可划分为目标地址明确控制转移指令和目标地址不明确控制转移指令。目标地址明确控制转移指令包括：全部 JCC 指令、直接 CALL 指令和直接 JMP 指令，简称为直接控制转移指令；目标地址不明确控制转移指令则包括：间接 CALL 指令、间接 JMP 指令、LCALL 指令、LJMP 指令、RET 指令和 IRET 指令，简称为间接控制转移指令。正是由于控制转移指令固有的、改变程序执行流程的特性，才使得目标代码具有代码块的概念，即从一个代码块转移到另外一个代码块继续执行。

定义 1：基本块 (Basic Block, BB) 是指以 GOS 起始执行地址、GOS 中断处理函数首地址或控制转移指令目标地址为起始边界，以无条件控制转移指令或已扫描地址为结束边界的连续目标指令序列。

定义 2：替换块 (Replaced Block, RB) 是指满足以下条件的最短连续目标指令序列：
1) 位于基本块内部（包括基本块自身）；
2) 包含指定敏感指令且长度不小于 JMP 指令长度；
3) 不含所属基本块范围之内、本替换块范围之外任何控制转移指令的目标地址
（本替换块首地址除外）。

[0018] 定义3：转换块（Translated Block，TB）是指替换块被替换之后，经由替换指令（通常为JMP指令或INT3指令）执行、包含虚拟化代码或安全检查代码且与原替换块具有等价执行语义的一组指令序列。

[0019] 定义4：待替换敏感指令队列是指以线性链表形式实现的、保存基本块扫描时发现的敏感指令地址及其相关信息的数据结构。待替换敏感指令队列节点以敏感指令地址为索引值，按地址大小按序排列。

[0020] 定义5：待扫描入口地址队列是指以线性链表形式实现的、保存动态扫描时发现的控制转移指令目标地址及其调用地址的数据结构。待扫描入口地址队列节点以目标地址为索引值，按地址大小按序排列。

[0021] 定义6：调用者地址队列是指以线性链表形式实现的、保存具有不同目标地址的控制转移指令地址的数据结构。调用者地址队列节点以控制转移指令地址为索引值，包含在待扫描入口地址队列的每个节点之中。

[0022] 定义6：已替换RB有效信息平衡二叉树是指以平衡二叉树AVL（balanced binary tree，以发明者G.M. Adelson-Velskii和E.M. Landis名字命名）形式实现的、保存被替换RB相关信息的数据结构。树中节点以被替换RB首地址为索引值，其它信息包括被替换RB的长度及其与对应TB中每条指令地址的对应关系等。

[0023] 定义7：扫描信息位图是指以位图（一段连续虚拟内存单元）形式实现的，用于保存GOS内核虚拟地址扫描状态及相关信息的数据结构。在扫描信息位图中，每条GOS虚地址的扫描信息由与该地址一一对应的扫描信息位元（简称位元）描述，每个扫描信息位元由二个比特位组成：00表示该位元对应的虚地址未扫描或非指令首地址；01表示该位元对应的虚地址未扫描；10表示该位元对应的虚地址未扫描或非指令首地址且为基本块边界或控制转移指令目标地址；11表示该位元对应的虚地址已扫描且为基本块边界地址或控制转移指令目标地址。从以上描述看出：当位元的高位为1时，表示对应的虚地址为基本块边界或控制转移指令的目标地址；当位元的低位为1时，表示对应的虚地址已扫描。由于每个虚地址对应二个比特位，因此每个4KB GOS内核代码页需要1KB大小的连续虚拟内存作位扫描信息位图。反之，每个4KB数据页可作为4个连续GOS内核代码页的扫描信息位图，将这种用途的数据页称为扫描信息位图。

[0024] 扫描信息位图索引结构由扫描信息位图页目录页（以下简称位图页目录），扫描信息位图页表页（以下简称位图页表）和扫描信息位图页（以下简称位图页）组成。相应的，GOS内核虚地址划分为页目录区、页表索引和页内偏移三个位段：最高位X位为页目录索引，对应2X个页目录项，存放页表页的首地址；中间Y位为页表索引，对应2Y个页表项，存放位图页的首地址；最低Z位为页内偏移，存放内核虚地址对应的扫描信息位元在位图页中的偏移量（以二个比特为单位计算）。

[0025] 在上述定义的基础之上，本发明具体技术方案如下：

[0026] 步骤1：创建并初始化全局数据结构；

[0027] 1.1创建并初始化扫描信息位图页目录页：分配一个数据页作为扫描信息位图索引结构中的页目录页，并将页目录页中的页目录项全部初始化为0（表示全部GOS内核虚拟地址空间尚未扫描）；
1.2 初始化待扫描入口地址队列及已替换 RB 有效信息平衡二叉树;将待扫描入口地址队列及已替换 RB 有效信息平衡二叉树的头指针设为 NULL。

1.3 令全局变量 M = JMP 替换指令长度。

步骤 2. 从 GOS 内核加载程序（如 Grub）输入参数中获取 GOS 内核起始执行地址，记为首扫描入口地址，然后对首个扫描入口地址进行动态扫描。

2.1. 采用扫描状态查询方法检查首个扫描入口地址的扫描状态;若已扫描,转步骤 3；若未扫描,将首个扫描入口地址记为基本块起始边界,转 2.2。扫描状态查询方法为:首先将被查询地址划分为页目录索引、页表索引和页内偏移三个位段;然后检查页目录索引指向的页目录项,若页目录项为 0,则返回 -1 表示该地址未扫描且该地址对应位元不存在。若页目录项不为 0,则继续检查页表索引指向的页表项,若页表项为 0,则返回 -1 表示该地址未扫描且该地址对应位元不存在。若页表项不为 0,则根据页表项中的扫描信息位图页首地址和页内偏移计算得到被查询地址对应位元的实际位置并返回位元的实际值。最后根据位元的实际值确定被扫描指令的首地址的扫描状态;位元的实际值低位为 1 表示已扫描,为 0 表示未扫描。

2.2. 对基本块起始边界进行基本块扫描,方法是:

2.2.1 初始化待转换敏感指令队列;将待转换敏感指令队列头指针设为 NULL；

2.2.2 将基本块起始地址设为待扫描地址；

2.2.3 对待扫描地址处指令进行译码,根据译码结果执行以下操作；

2.2.3.1 若为敏感指令或间接控制转移指令,则将该指令添加到待转换敏感指令队列,转 2.2.4；

2.2.3.2 若为直接控制转移指令,则首先采用目标地址有效性检查方法检查该目标地址是否指向已替换 RB 内部;若指向已替换 RB 内部,则将直接控制转移指令添加到待转换敏感指令队列以便替换时对其目标地址进行修正,然后转 2.2.4; 若不指向已替换 RB 内部,则采用 2.1 中扫描状态查询方法检查目标地址的扫描状态,若目标地址已扫描则转 2.2.4, 否则将目标地址添加到待扫描入口地址队列,转 2.2.4。目标地址有效性检查方法为: 采用通用的平衡二叉树遍历方法遍历已替换 RB 有效信息平衡二叉树, 找到索引值小于指定目标地址且与指定目标地址最接近的节点, 若指定目标地址小于或等于该节点索引值 (即已替换 RB 的首地址) 加上节点中记录的已替换 RB 长度, 则表明指定目标地址指向已替换 RB 内部, 否则不指向已替换 RB 内部。

2.2.3.3 若为普通指令,则转 2.2.4；

2.2.4 根据译码结果记录被扫描指令的扫描状态, 方法为: 首先采用扫描信息位元索引方法得到被扫描指令首地址对应位元的实际位置, 将该位元的低位设为 1; 然后采用扫描信息位元索引方法依次得到被扫描指令所有剩余字节对应位元的实际位置, 将这些位元设为 00; 最后若被扫描指令为直接控制转移指令, 则采用扫描信息位元索引方法得到该控制转移指令目标地址对应的位元的实际位置, 并将该位元的高位设为 1。其中, 扫描信息位元索引方法为: 首先将被索引地址划分为页目录索引、页表索引和页内偏移三个位段; 然后检查页目录索引指向的页目录项是否为 0, 是则立即分配一个数据页作为扫描信息位图页表页并将该页表页的首地址写入页目录页中对应的页目录项中; 接着检查表索引指

地址写入页表项中的对应页表项中；最后根据扫描信息位图页首地址和页内偏移得到被索引地址对应位元的实际位置。

【0040】2.2.5 将被扫描指令地址加上该指令长度，得到下一条指令的首地址。

【0041】2.2.6 若被扫描指令为无条件控制转移指令或采用 2.1 中扫描状态查询方法发现下一条指令首地址已扫描，则将下一条指令首地址设为基本块的结束边界，并将其扫描信息位元的高位设为 1，表示该地址为基本块边界地址或控制转移指令目标地址，然后结束本次基本块扫描并转 2.3；否则转 2.2.7。

【0042】2.2.7 将下一条指令首地址设为待扫描地址，转 2.2.3，继续进行基本块扫描。

【0043】2.3 对待转换敏感指令队列中的敏感指令进行基于替换块的敏感指令本地替换；

【0044】2.3.1 若待转换敏感指令队列为空，则停止替换，转步骤 3；若待转换敏感指令队列不为空，从待转换敏感指令队列中取出最小的敏感指令位置，在基本块内（从基本块起始边界到基本块结束边界之间的连续虚地址空间）寻找包含该敏感指令的替换块，寻找方法如下：

【0045】2.3.1.1 将该敏感指令自身设为当前 RB。

【0046】2.3.1.2 若当前 RB 长度不小于 M 字节，则将当前 RB 设为待替换 RB，转 2.3.3；若当前 RB 长度小于 M 字节，尝试向低地址空间扩展相邻指令到当前 RB，扩展方法如下：

【0047】A1. 若当前 RB 首地址为基本块起始边界，则转 2.3.1.3；否则向低地址空间扩展一条指令到当前 RB，方法是：采用 2.2.4 中的扫描信息位元索引方法从当前 RB 首地址开始向低地址空间搜索扫描信息位图，依次得到每条连续地址对应位元的实际位置和检查每个位元的实际值，找到第一个低为 1 的位元，将该低为 1 的位元对应的虚地址（即被扩展指令的首地址）设为当前 RB 起始边界，转 B1。

【0048】B1. 若被扩展指令首地址对应位元的高位为 1，表示该位元对应的地址为控制转移指令的目标地址，执行 C1；否则转 2.3.1.2。

【0049】C1. 查找待扫描入口地址队列，若发现以被扩展指令首地址为索引值的节点，则采用遍历扩展方法检查该节点的调用者队列，将所有位于基本块范围之内的调用者地址与当前 RB 之间的所有指令扩展到当前 RB，转 2.3.1.2，若未发现以被扩展指令首地址为索引值的节点，则转 2.3.1.2。遍历扩展方法是：采用 2.2.3 中的目标地址有效性检查方法依次检查每个调用者地址是否位于基本块范围之内，且位于当前 RB 范围之外，是将调用者地址设为当前 RB 的边界；若该调用者地址位于当前 RB 范围之内，则将该地址设为当前 RB 首地址，若该地址大于当前 RB 的结束边界，将调用者地址的下一条指令首地址（即调用者地址加上该地址处的控制转移指令长度）设为当前 RB 的结束边界。否则检查下一个调用者地址。

【0050】2.3.1.3 尝试向高地址空间扩展相邻指令到当前 RB，扩展方法如下：

【0051】A2. 若当前 RB 结束边界为基本块的结束边界，转 2.3.2，否则向高地址空间扩展一条指令到当前 RB，方法是：采用 2.2.4 中的扫描信息位元索引方法从当前 RB 结束边界开始向高地址空间搜索扫描信息位图，依次得到每条连续地址对应位元的实际位置和检查每个位元的实际值，找到第二个低为 1 的位元（对应当前 RB 之后第一位已被扫描指令的下一条指令首地址）或第一个被设置为 10 的位元（对应基本块的结束边界），将该位元对应的虚地址设为当前 RB 结束边界，而找到的第一个低为 1 的位元对应的虚地址即被扩展指令首
地址。

【0052】B2. 若被扩展指令首地址对应的位元为 10, 转 2.3.2; 否则检查被扩展指令首地址对应位元的高位, 若高位为 1, 转 2.3.2, 若高位为 0, 转 2.3.1.2。

【0053】C2. 检查待扫描入口地址队列, 若发现被扩展指令首地址为索引值的节点, 则采用 C1 中的遍历扩展方法检查该节点的调用者队列, 将所有位于基本块块范围之内的调用者地址与当前 RB 之间的所有指令扩展到当前 RB, 转 2.3.1.2。若未发现被扩展指令首地址为索引值的节点, 直接转 2.3.1.2。

【0054】2.3.2 若当前 RB 长度小于 M 字节（表明基本块块长度小于 JMP 替换指令）, 则将敏感指令设为待转换 RB; 否则将当前 RB 设为待转换 RB。

【0055】2.3.3 采用二进制翻译中的常规方法生成与待转换 RB 具有等价执行语义的转换代码块, 记为 TB。

【0056】2.3.4 若待转换 RB 的长度不小于 M 字节, 则将该 RB 前 M 字节替换为 JMP 指令, 目标地址为 2.2.3 中生成的 TB 首地址, 剩余字节用 NOP 指令填充, 得到已替换 RB; 若待转换 RB 的长度小于 M 字节, 则将该 RB 的首字节替换为 INT3 指令, 剩余字节用 NOP 指令填充, 得到已替换 RB。

【0057】2.3.5 在已替换 RB 有效信息平衡二叉树中创建已替换 RB 的节点, 该节点包括已替换 RB 的首地址、长度及其与 TB 指令之间的地址对应关系这些有效信息。对于被替换为 INT3 指令的 RB（实际为敏感指令）, 将在该 INT3 指令执行时由 VMM INT3 异常处理函数根据已替换 RB 有效信息平衡二叉树中记录的信息找到并执行其转换代码; 对于被替换为 JMP 指令的 RB, 则通过 JMP 指令直接执行已替换 RB 的转换代码。

【0058】2.3.6 遍历待扫描敏感指令队列, 删除所有以已替换 RB 中的敏感指令地址为索引值的节点, 转 2.3.1。

【0059】2.4 遍历待扫描入口地址队列, 删除所有以已扫描入口地址为索引值的节点。

【0060】2.5 若待扫描入口地址队列为空, 则结束本次动态扫描, 转步骤 3; 否则从待扫描入口地址队列中取出最大未扫描入口地址, 记为基本块起始边界, 转 2.2 执行下一次基本块扫描。

【0061】步骤 3. 从首个扫描入口地址开始执行 GOS 内核代码, 直至产生硬件中断或异常, 或执行间接控制转移指令的转换代码。

【0062】步骤 4. 将 GOS 中断或异常处理函数的入口地址或间接控制转移指令的目标地址设为首个扫描入口地址, 转 2.1。

【0063】根据以上过程可知：在对 GOS 执行完第一次动态扫描之后, GOS 便处于原有代码与转换代码反复、交替运行状态, 并且随着 GOS 的运行, 在动态加载新的 GOS 内核模块的前提下, 未经扫描的 GOS 内核代码将会越来越少, 从而动态扫描所引入的系统开销也会越来越小。

【0064】采用本发明可以达到以下技术效果：

【0065】1. 采用基于替换块的敏感指令本地替换方法, 以替换块为单位进行替换, 能够最大限度地避免使用 INT0/INT3 指令进行替换（只在基本块长度小于 JMP 替换指令时使用 INT3 指令替换）, 从而有效提高 BT 虚拟化性能。

【0066】2. 可对任意定义的 X86 敏感指令进行替换, 可扩展性好, 适用性广。
3. 以 GOS 内核地址为索引动态创建扫描信息位图，不仅能够减少内存开销，而且能够有效防止转位攻击和基于缓冲区溢出的地址变化攻击。

因此，本发明既可满足采用 BT 技术实现操作系统虚拟化的软件支持需求，也能够应用于要求对目标代码进行安全检查或隔离的应用场景。

附图说明

图 1 是本发明总体流程图；
图 2 是本发明第 2.2 步中基本块扫描流程图；
图 3 是本发明第 2.3 步中敏感指令本地替换流程图；
图 4 是本发明定义 7 中扫描信息位图索引结构及索引方法示意图。
具体实施方法
图 1 为本发明整体流程图
1. 创建并初始化全局数据结构。
2. 从 GOS 内核加载程序输入参数中获取 GOS 内核起始执行地址，记为首个扫描入口地址，然后对首个扫描入口地址进行动态扫描。
2.1. 采用扫描状态查询方法检查首个扫描入口地址的扫描状态；若已扫描，转 3；若未扫描，将首个扫描入口地址记为基本块起始边界，转 2.2。
2.2. 对基本块起始边界进行基本块扫描。
2.3. 对待替换敏感指令队列中的敏感指令进行基于替换块的敏感指令本地替换。
2.4. 遍历待扫描入口地址队列，删除所有以已扫描入口地址为索引值的节点。
2.5. 若待扫描入口地址队列为空，则结束本次动态扫描，转 3；否则从待扫描入口地址队列中取出最大未扫描入口地址，记为基本块起始边界并转 2.2。
3. 从首个扫描入口地址开始执行 GOS 内核代码，直至产生硬件中断或异常或执行间接控制转移指令的转移代码。
4. 将 GOS 中断或异常处理函数的入口地址或间接控制转移指令的目标地址设为首个扫描入口地址，转 2.1。
图 2 是本发明 2.2 步中基本块扫描流程图
2.2.1. 初始化待替换敏感指令队列；将待替换敏感指令队列头指针设为 NULL。
2.2.2. 将基本块起始地址设为待扫描地址。
2.2.3. 对待扫描地址处指令进行译码，根据译码结果执行以下操作；
2.2.3.1. 若为敏感指令或间接控制转移指令，则将该指令添加到待替换敏感指令队列，转 2.2.4；
2.2.3.2. 若为直接控制转移指令，则采用目标地址有效性检查方法检查该目标地址是否指向已替换 RB 内部；若指向已替换 RB 内部，则将直接控制转移指令添加到待替换敏感指令队列，然后转 2.2.4；若不指向已替换 RB 内部，则检查目标地址的扫描状态，若目标地址未扫描则转 2.2.4，否则将目标地址添加到待扫描入口地址队列，转 2.2.4。
2.2.3.3. 若为普通指令，转 2.2.4。
2.2.4. 根据译码结果记录被扫描指令的扫描状态；首先采用扫描信息位元索引方法得到被扫描指令首地址对应位元的实际位置，将该位元的低位设为 1；然后采用扫描信
息位元索引方法依次得到被扫描指令所有剩余字节对应位元的实际位置，将这些位元设为00，最后若被扫描指令为直接控制转移指令，则采用扫描信息位元索引方法得到该控制转移指令目标地址对应的位元的实际位置，并将该位元的高位设为1。

0091.2.2.5 将被扫描指令地址加上该指令长度，得到下一条指令的首地址。
0093.2.2.6 若被扫描指令为无条件控制转移指令或下一条指令首地址已扫描，则将下一条指令的首地址设为基本块的结束边界，并将其扫描信息位元的高位设为1，然后转2.3；否则转2.2.7。

0094.2.2.7 将下一条指令的首地址设为待扫描地址，转2.2.3。
0095.图3是本发明2.3步中基于替换块的敏感指令本地替换流程图。
0096.2.3.1 若待替换敏感指令队列为空，则停止替换，转步骤3；否则将替换敏感指令队列不为空，从待替换敏感指令队列中取出最小的敏感指令地址，在基本块内（即从基本块起始边界到基本块结束边界之间的连续虚拟地址空间）寻找包含该敏感指令的替换块，寻找方法如下：

0097.2.3.1.1 将该敏感指令自身设为当前RB。
0098.2.3.1.2 若当前RB长度不小于M字节，则将当前RB设为待替换RB，转2.3.3；否则当前RB长度小于M字节，尝试向低地址空间扩展相邻指令到当前RB，扩展方法如下：

0099. A1. 若当前RB首地址为基本块起始边界，转2.3.1.3；否则向低地址空间扩展一条指令到当前RB，方法是：采用2.2.4中的扫描信息位元索引方法，从当前RB首地址开始向低地址空间搜索扫描信息位元图，依次得到每条连续地址对应位元的实际位置，并检查每个位元的值，找到一个低截为1的位元，将该低截为1的位元对应的虚地址（即被扩展指令的首地址）设为当前RB起始边界，转B1。

0100. B1. 若被扩展指令首地址对应位元的高截为1，表示该位元对应的地址为控制转移指令的目标地址，转C1；否则转2.3.1.2。

0101. C1. 检查待扫描入口地址队列，若发现以被扩展指令首地址为索引值的节点，则采用遍历扩展方法检查该节点的调用者队列，将所有位于基本块范围之内的调用者地址与当前RB之间所有指令扩展到当前RB，转2.3.1.2。若未发现以被扩展指令首地址为索引值的节点，则转2.3.1.2。

0102. 2.3.1.3 尝试向高地址空间扩展相邻指令到当前RB，扩展方法如下：

0103. A2. 若当前RB结束边界为基本块的结束边界，转2.3.2；否则向高地址空间扩展一条指令到当前RB，方法是：采用2.2.4中的扫描信息位元索引方法，从当前RB结束边界开始向高地址空间搜索扫描信息位元图，依次得到每条连续地址对应位元的实际位置，并检查每个位元的值，找到第二个低截为1的位元（对应当前RB之后第一条已扫描指令的下一条指令首地址）或第一个被设置为10的位元（对应基本块的结束边界），将该位元对应的虚地址设为当前RB结束边界，而找到的第一个低截为1的位元对应的虚地址即被扩展指令首地址。

0104. B2. 若被扩展指令首地址对应的位元为10，转2.3.2；否则检查被扩展指令首地址对应位元的高截，若高截为1，转C2，若高截为0，转2.3.1.2。

0105. C2. 检查待扫描入口地址队列，若发现以被扩展指令首地址为索引值的节点，则采用遍历扩展方法检查该节点的调用者队列，将所有位于基本块范围之内的调用者地址与当前
前RB之间的所有指令扩展到当前RB，转2.3.1.2。若未发现以被扩展指令首地址为索引值的节点，直接转2.3.1.2。

【0106】 2.3.2 若当前RB长度小于M字节（表明基本块长度小于JMP替换指令），则将敏感指令设为待转换RB；否则将当前RB设为待转换RB。

【0107】 2.3.3 采用二进制翻译中的常规方法生成与待转换RB具有等价执行语义的转换代码块，记为TB。

【0108】 2.3.4 若待转换RB的长度不小于M字节，则将该RB前M字节替换为JMP指令。目标地址为2.2.3中生成的TB首地址，剩余字节用NOP指令填充，得到已替换RB；若待转换RB的长度小于M字节，则将该RB的首字节替换为INT3指令，剩余字节用NOP指令填充，得到已替换RB。

【0109】 2.3.5 在已替换RB有效信息平衡二叉树中创建已替换RB的节点。该节点包括已替换RB的首地址、长度及其与TB指令之间的地址对应关系这些有效信息。对于被替换成INT3指令的RB（实际为敏感指令），将在该INT3指令执行时由VMM INT3异常处理函数根据已替换RB有效信息平衡二叉树中记录的信息找到并执行其转换代码；对于被替换成JMP指令的RB，则通过JMP指令直接执行已替换RB的转换代码。

【0110】 2.3.6 遍历待扫描敏感指令队列，删除所有以已替换RB中的敏感指令地址为索引值的节点，转2.3.1。

【0111】 图4是本发明定义7中扫描信息位图及其索引方法示意图

【0112】 以X86-32位体系结构为例；32位内核虚地址共划分为页目录索引、页表索引和页内偏移三个位段。其中：最高8位为页目录索引，对应256个页表；中间10位为页表索引，对应1024个页表；最低14位则为内核虚地址对应位元在扫描信息位图页中的偏移量（以二个比特位为单位计算）。指定GOS内核虚地址va对应的扫描信息位元索引与获取方法如下：

【0113】 1) 取va的高8位（即页目录索引），左移2位后加上页目录页首地址，得到va对应的页目录项的首地址，从页目录项中取出页表页的首地址；

【0114】 2) 取va的中间10位（即页表索引），左移2位后加上页表页的首地址，得到va对应的页表页的首地址，从页表项中取出页图页的首地址；

【0115】 3) 取va的低14位，右移2位后加上页图页的首地址，得到va对应的扫描信息位元所在的内存单元地址（位于扫描信息位图页中），记为pScanByte；

【0116】 4) 将va的低14位与0x3进行位与运算，得到va对应的扫描信息位元在pScanByte中的偏移量，记为offset；

【0117】 5) 按下列公式计算得到va对应的扫描信息位元的实际值，记为ScanBit：

\[
\text{ScanBit} = (\text{pScanByte} \& (0x3<<((\text{offset]<<1))>>>\text{offset}<<1))
\]

【0118】 公式中的“<<”为左移运算符，“>>”为右移运算符，表示将16进制无符号整数左移或右移指定的位数；“&”为位与运算符。
1. 创建并初始化全局数据结构

2. 从 GOS 内核加载程序输入参数中获取 GOS 内核起始执行地址，记为首个扫描入口地址，然后对首个扫描入口地址进行操作

2.1 采用扫描状态查询方法检查首个扫描入口地址的扫描状态

- 未扫描
  - 设首个扫描入口地址为基本块起始边界
  - 2.2 对基本块起始边界进行基本块扫描
  - 2.3 对待转换敏感指令队列中的敏感指令进行基于替换块的敏感指令本地替换
  - 2.4 遍历待扫描入口地址队列，删除所有以已扫描入口地址为索引值的节点
  - 是
    - 2.5 待扫描入口地址队列为空？
    - 否
      - 从待扫描入口地址队列中取出最大未扫描入口地址，记为基本块起始边界

3. 从首个扫描入口地址开始执行 GOS 内核代码，直至产生硬件中断或异常或执行间接控制转移指令的转换代码

4. 将 GOS 中断或异常处理函数的入口地址或间接控制转移指令的目标地址设为首个扫描入口地址

图 1
2.2.1 初始化待转换敏感指令队列

2.2.2 将基本块起始地址设为待扫描地址

2.2.3 对待扫描地址处指令进行译码，根据译码结果执行以下操作

2.2.3.1 敏感指令或间接控制转移

2.2.3.2 直接控制转移指令

采用目标地址有效性检查方法检查目标地址是否指向已替换 RB 内部

不指向

指向

检查目标地址的扫描状态

已扫描

未扫描

将目标地址添加到待扫描入口地址队列

将该指令添加到待转换敏感指令队列

2.2.4 根据译码结果记录被扫描指令的扫描状态

2.2.5 将被扫描指令地址加上该指令长度，得到下一条指令的首地址

2.2.6 被扫描指令为无条件控制转移指令或下一条指令已扫描？

是

将下一条指令的首地址设为基本块的结束边界，并将其扫描信息位元的高位设为 1

否

2.2.7 将下一条指令的首地址设为待扫描地址

图 2
2.3.1 待转换敏感指令队列为空？

是 → (来自 2.3) → (返回 3)

否 →

2.3.1.1 从待转换敏感指令队列取出最小的敏感指令地址

该敏感指令本身设为当前 RB

2.3.1.2 当前 RB 长度不小于 M 字节？

否 →

A1. 当前 RB 首地址为基本块起始边界？

是 → 向低地址空间扩展一条指令到当前 RB

否 → 向高地址空间扩展一条指令到当前 RB

否 →

B1. 被扩展指令首地址对应位元的高位为 1？

是 →

C1、C2. 检查待扫描入口地址队列，发现以低地址为 1 位元对应的虚地址为索引值的节点

否 →

采用遍历扩展方法检查该节点的调用者队列，将所有位于基本块范围之内的调用者地址与当前 RB 之间的所有指令扩展到当前 RB

否 →

是 →

2.3.2 当前 RB 结束边界为基本块结束边界？

否 →

A2. 当前 RB 结束边界为基本块结束边界？

是 →

向高地址空间扩展一条指令到当前 RB

否 → 向低地址空间扩展一条指令到当前 RB

否 →

B2. 被扩展指令首地址对应的位元为 10？

是 →

被扩展指令首地址对应位元的高位为 1

否 →

2.3.3 采用二进制翻译中的常规方法生成和转换 RB 具有等价执行语义的转换代码块，记为 TB

是 →

2.3.4 待转换 RB 长度小于 M 字节？

否 →

将待转换 RB 首字节替换为 INT3 指令

否 →

将待转换 RB 首地址转换为 JMP 指令，目标地址为 TB 首地址

2.3.5 在已替换 RB 有效信息平衡二叉树中创建已替换 RB 的节点

2.3.6 遍历待扫描敏感指令队列，删除所有以已替换 RB 中的敏感指令地址为索引值的节点

图 3

17
图 4