(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102184360 B
(45)授权公告日 2013.06.05

(21)申请号 201110124371.7
(22)申请日 2011.05.13
(73)专利权人 华中科技大学
地址 430074 湖北省武汉市洪山区珞喻路1037号
(72)发明人 刘政林 秦保力 朱庆春 周昭柳 李东方 殷雄 陈天山 董磬 郭超
(74)专利代理机构 华中科技大学专利中心 42201
代理人 曹葆青
(51)Int.Cl.
G06F 21/57(2013.01)
G06F 9/30(2006.01)
审查员 杨庆丽

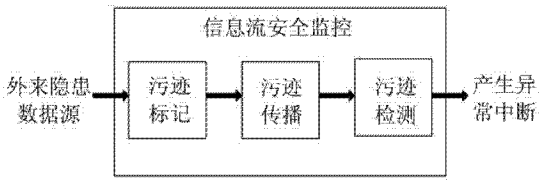
权利要求书1页 说明书7页 附图6页

(54)发明名称

一种适用于嵌入式处理器的信息流安全监控方法

(57)摘要

本发明公开了一种适用于嵌入式处理器的信息流安全监控方法，包括污迹的标记、污迹的传播和污迹的检测。污迹标记是对外来隐患数据进行标记。污迹传播是使污迹信息随数据源一起参与到流水线的运算当中，并在数据操作过程中与数据绑定在一起且与数据保持同步。污迹检测对污迹数据源行为的安全性进行检测，监控污迹数据源在传播过程中的行为，当发现污迹数据被不安全的方式使用时，产生异常中断。由于本发明在污迹信息的传播过程中省略了污迹传播寄存器而以全传播的形式进行传播，在一定程度上减少了系统性能方面的开销，同时由于在污迹信息的检测过程中使用了污迹检测寄存器，可以更有针对性的防御不同类型的攻击，减小了整个信息流安全监控方法的误报率。



CN 102184360 B

1. 一种适用于嵌入式处理器的信息流安全监控方法，其特征在于，该方法包括：

(A) 对攻击目标进行污迹标记；

(B) 对污迹进行传播：

当污迹数据源进入到嵌入式处理器内核时，在处理器流水线上污迹标记位随着系统对该污迹数据源进行读入、读出及运算操作而传播；将污迹标记位在污迹数据源被读入、读出及运算的过程中和污迹数据源绑定在一起，以追踪污迹数据源的整个传播过程；

(C) 对污迹进行检测：

(C1) 在处理器内核设置污迹检测寄存器，在指令通过流水线时，处理器通过污迹检测寄存器完成对污迹的检测；

(C2) 在污迹检测过程中对威胁进行分类，将攻击源在内核中的威胁行为分为不同的级别

步骤(C1)中，污迹检测寄存器控制字的定义和操作如下：

(C11) 污迹检测寄存器中的 pc 定义了对程序指针的检测规则，如果 pc 位为“1”而且程序指针跳转到 tag（0）值为“1”的地址就会产生异常；

(C12) 污迹检测寄存器中的 inst 定义了对指令的检测规则，如果 inst 位为“1”而且系统在取指令阶段所取指令的 tag（0）值为“1”就会产生异常；

(C13) 污迹检测寄存器中的 addrs 定义对敏感地址及敏感地址段的检测规则，如果 addrs 位为“01”而且外来攻击对敏感地址有操作，且污迹数据源对其内容进行改写时就产生异常；如果 addrs 位为“10”而且外来攻击对敏感地址段有操作，且污迹数据源对其内容进行改写时就产生异常；

(C14) 污迹检测寄存器中的 move 定义了 mov 运算操作中的检测规则，如果在 mov 运算中 move(0) 位为“1”而且源操作数的 tag（0）值为“1”就会产生异常；如果在 mov 运算中 move(1) 位为“1”而且目的操作数的 tag（0）值为“1”就会产生异常；如果在 mov 运算中 move(2) 位为“1”而且源地址的 tag(0)值为“1”就会产生异常；如果在 mov 运算中 move(3) 位为“1”而且目的地址的 tag（0）值为“1”就会产生异常；

(C15) 污迹检测寄存器中的 comp 定义了对比运算操作中的检测规则，如果 comp(0)位为“1”而且在对比运算中源操作数的 tag（0）值为“1”就会产生异常；如果 comp（1）位为“1”而且在对比运算中目的操作数的 tag（0）值为“1”就会产生异常；

(C16) 污迹检测寄存器中的 logic 定义了逻辑运算操作中的检测规则，如果 logic(0)位为“1”而且在对比运算中源操作数的 tag（0）值为“1”就会产生异常；如果 logic（1）位为“1”而且在对比运算中目的操作数的 tag（0）值为“1”就会产生异常。

一种适用于嵌入式处理器的信息流安全监控方法

技术领域

[0001] 本发明属于数字集成电路和嵌入式系统安全领域，具体涉及一种适用于嵌入式处理器的信息流安全监控方法，该方法的核心是利用污迹追踪的方法监控嵌入式处理器信息流的安全，是一种高性能，低功耗，误报率较低的有效防御嵌入式系统中恶意软件攻击的方法。由于在设计时就考虑了性能与开销等因素，该方法完全适用于嵌入式系统，也适用于计算机系统，能够为其提供可靠的信息流安全的保障。

背景技术

[0002] 近些年来，嵌入式系统受到了来自恶意软件的严重威胁。2005 年在芬兰赫尔辛基世界田径锦标赛上大规模爆发的手机病毒 Cabir 便是其中的典型代表。截至到 2006 年 4 月，全球仅针对智能手机的病毒就出现了近两百种，并且数量还在迅猛增加。恶意软件已经开始威胁嵌入式设备的正常使用。嵌入式系统之所以容易成为恶意软件攻击的对象是基于以下几点原因。

[0003] 首先嵌入式系统的应用环境越来越开放，随着以 Symbian、Windows CE、Linux 为代表的嵌入式操作系统的推广，以及蓝牙、无线网络的兴起，导致恶意软件可以更方便的传播；其次嵌入式系统缺乏必要的安全防护措施；最后也是最关键的原因是作为嵌入式系统中核心部件的嵌入式处理器本身缺乏必要的安全机制。嵌入式处理器在嵌入式系统中从事着数据交换、处理等重要工作，但是嵌入式处理器不会检查所执行的程序是否安全。从处理器的角度来看，恶意软件与正常程序相同，这就造成了恶意软件在指令级层次是透明的，从而为恶意软件的攻击留下了潜在隐患。

[0004] 如果嵌入式处理器在软件运行的过程中能及时发现具有安全威胁的程序并予以中止，就可以大大提高嵌入式系统在运行时的安全性，为嵌入式设备在不安全的环境下的应用提供可靠保障。与桌面计算机所不同的是嵌入式系统是一个相对封闭的系统环境，与前代产品的兼容性问题小，并且嵌入式处理器结构简单，在体系结构上具有较大的改进余地。因而本发明从指令级层次入手研究嵌入式处理器的细粒度安全运行机制，探索在恶意软件威胁下提高嵌入式系统安全性的方法。国外从 2000 年开始，就展开了针对处理器安全性的研究。如图 1 所示，我们列举了近年来，在处理器架构安全性研究方面的发展历程。

[0005] 2000 年，针对处理器的安全性问题，美国 Stanford 大学率先提出了 XOM(eXecute-Only-Memory) 架构。XOM 主要思想是在存储器中保存只能执行的指令，不允许指令发生修改，并且通过对指令进行加密来保证指令的安全性。XOM 可以抵御恶意篡改以及窃听，但是由于其对进出外部存储器的数据以及程序都要进行验证，因而工作效率很低，同时这种方法不能防御来自应用程序本身的攻击。

[0006] 2003 年麻省理工 (MIT) 人工智能实验室针对物理攻击提出的一种安全处理器架构 AEGIS。AEGIS 利用物理真随机数发生器 (TRNG) 产生一个唯一的身份标示，并作为签名添加到加密算法中，所有保存在外部存储器的数据都经过加密运算。AEGIS 对物理攻击能做出很好的防范，但是 AEGIS 同样无法抵御来自内部的破坏，而且 AEGIS 的加密方式使其本身

的性能受到了较大的影响。

[0007] 从 2004 年开始，利用硬件提高处理器安全性的研究开始逐步升温，出现了以 Mios 为代表的控制流监控和以 RIFLE 为代表的信息跟踪两种主要方式。到了 2006 年出现了以 Heapmon 为代表的通过存储器访存追踪来进行安全性判断的方式。从追踪数据的角度上看，这种方法与信息追踪方式相似，都是依靠监控运行时的数据轨迹来完成工作的。这两种方法同时也都存在着运行效能不高的问题。

[0008] 目前，控制流监控、信息追踪以及存储器追踪这三种安全处理方法已成为研究硬件架构安全性的主流方法。从 2007 年到 2008 年的研究中，不同方法之间的交叉研究成为新的趋势，例如 FlexTaint。

[0009] 以上的安全架构多是以桌面计算机中的通用处理器为假想应用环境。其中 XOM 与 AEGIS 不能应对来自应用软件本身的安全威胁，而 RTM 没有考虑嵌入式系统对来自系统外部程序的安全监控要求。因而这些模型不能解决嵌入式系统中存在的恶意软件威胁的问题。以嵌入式系统应用环境为参考点解决嵌入式处理器所面临的运行安全性问题将是我们的研究重点。

[0010] 国内在硬件安全方面的研究主要集中于安全协议以及 AES、RSA 等安全加密算法的实现上。但是在关于嵌入式处理器安全架构方面的研究，特别是在嵌入式处理器的硬件层次上，探索抗恶意软件攻击的研究基本上处于空白。目前工业界所广泛采用的安全产品缺乏自主知识产权，而且较为陈旧，不够安全。另一方面，一些先进的安全产品属于受限范围并且价格昂贵，在我国的使用受到了限制。这种情况严重威胁了我国在相关领域的公共安全。由于安全嵌入式处理器涉及敏感，国外公司不可能向我国公开其完整的设计方案以及其中可能存在的安全隐患，因而国外嵌入式处理器是否真正安全值得怀疑，嵌入式处理器在特殊行业的使用也受到了相应的限制。这些都不利于嵌入式设备在我国的进一步推广发展。

发明内容

[0011] 本发明的目的在于提供一种适用于嵌入式处理器的信息流安全监控方法，该方法与现有技术相比具有更小的硬件开销和误报率，能达到更优的性能，能够真正应用于对设计要求较高的嵌入式处理器中。

[0012] 本发明提供的一种适用于嵌入式处理器的信息流安全监控方法，其特征在于，该方法包括：

[0013] (A) 对攻击目标进行污迹标记；

[0014] (B) 对污迹进行传播：

[0015] 当污迹数据源进入到嵌入式处理器内核时，在处理器流水线上污迹标记位随着系统对该污迹数据源进行各种各样的操作而传播；将污迹标记位在数据源被读入、读出及运算的过程中和数据源绑定在一起，以追踪污迹数据源的整个传播过程；

[0016] (C) 对污迹进行检测：

[0017] (C1) 在处理器内核设置污迹检测寄存器，在指令通过流水线时，处理器通过污迹检测寄存器完成对污迹的检测；

[0018] (C2) 在污迹检测过程中对威胁进行分类，将攻击源在内核中的威胁行为分为不同

的级别。

[0019] 鉴于嵌入式处理器安全性的重要性以及国内外的发展现状，为了解决嵌入式系统中存在的运行安全性差的问题，本发明从嵌入式处理器的体系结构入手，研究嵌入式处理器在恶意软件威胁条件下的安全运行机制及实现方法，采用针对嵌入式处理器的信息流安全监控方法。本发明针对嵌入式处理器的安全运行机制，采用污迹追踪的形式，对从外来隐患程序进行实时追踪和监控，从而保证原有程序的安全性。该污迹追踪方法在对污迹进行传播时采用了全传播的方法，这样在一定程度上减小了系统性能上的开销。同时该方法添加了污迹检测寄存器，用户可以通过软件进行配置这些寄存器的控制字，而且每个寄存器对应一类攻击，这样可以同时防御多个类型的攻击，因此可以大大的减小该设计在防恶意攻击时的误报率。在充分考虑嵌入式处理器的安全性、性能以及成本开销的前提下，提高嵌入式处理器的安全性，为嵌入式系统的安全应用提供可靠保障。

附图说明

[0020] 图 1 为处理器架构安全性研究的发展示意图；

[0021] 图 2 为污迹追踪形式的示意图；

[0022] 图 3 为信息安全监控的基本过程图；

[0023] 图 4 为信息安全监控方法设计流程图；

[0024] 图 5 污迹全传播的操作示意图；

[0025] 图 6 为污迹检测寄存器示意图；

[0026] 图 7 为实现信息安全监控方法的系统结构图；

[0027] 图 8 为 stack smashing 的实验结果图；

[0028] 图 9 为 format string 的实验结果图；

[0029] 图 10 为处理器加入信息安全监控前后 FPGA 资源开销对比示意图。

具体实施方式

[0030] 下面结合附图和实例对本发明作进一步详细的说明。

[0031] 本发明提供了一种适用于嵌入式处理器的信息流监控方法，包括对来自恶意软件威胁的标记，传播和检测。它主要通过污迹追踪的形式对来自处理器外部的程序进行跟踪鉴别，对程序进行实时监控，及时判别其行为的安全性，从而保证嵌入式系统正常的运行。

[0032] 当前嵌入式恶意攻击多利用程序中的函数返回地址，指针等来插入恶意攻击程序片段，对外部输入的敏感信息是攻击者的主要目标。在嵌入式系统在运行过程中，会有新的程序下载到系统中运行。外部程序可能会含有危险的病毒、木马等恶意软件，因而这些程序存在潜在的威胁和隐患，即低安全性程序。在运行这些程序时，追踪这些污迹在处理器中运行的轨迹，并判别这些程序运行时是否安全。如图 2 所示，C 程序是带有污迹的程序，因而追踪程序 C 的运行轨迹。程序 A、B 和 D 是本地的安全程序。步骤 1、2 是安全程序 A、B 的正常运行状态，步骤 3、4 是污迹程序 C 的是污迹程序的运行状态，在步骤 5，当污迹程序 C 与安全程序 A 的运行轨迹重叠时，将会侦测到程序 A 与 C 发生数据交换时，并干预程序运行，阻断步骤 6，从而达到保护目的。

[0033] 本发明在不改变现有体系结构中的主要数据通路的前提下进行。整体的设计流程

如图 4 所示。本发明在应用程序代码编译工作完成后，增加后处理环节。当程序下载到嵌入式系统后，对程序进行实时监控。

[0034] 为了给信息安全监控方法提供一个标记污迹和追踪污迹信息的工作环境，本设计在嵌入式处理器内核上扩充了污迹的标记位，而且每一个字上带有四个污迹的标记位。在硬件上添加四个标记位到寄存器，而且在 cache、memory 上扩展数据总线从 32 位到 36 位。在总线及存储器上所扩展的标记位添加在数据的高位上，以下是各个标记位的定义：

[0035]

标记位	定义
tag（0）	污迹的标记位：为“1”时相应的数据为不可信数据；为“0”时相应的数据为可信数据
tag（2～1）	威胁级别分类位：为“00”时相应数据为低威胁数据；为“01”时为中威胁数据；为“11”时为高威胁数据
tag（3）	敏感信息标志位：为“1”时为敏感信息；为“0”时为非敏感信息

[0036] 信息流安全监控方法主要分为三个步骤：

[0037] (A) 对攻击目标进行污迹标记：

[0038] 在程序运行过程中，当外部数据通过嵌入式处理器的外设或外部端口进入嵌入式处理器内核时，将所有输入的外部数据源标记为带有污迹的数据源。即将任何与 CPU 连接的输入端口输入的数据标记为不可信数据，即污迹数据源。污迹标记的具体的做法就是：在外部数据源进入处理器内核之前，对该外部数据源进行标记，即将该数据的污迹标记位 tag(0) 置为“1”.

[0039] (B) 对污迹进行传播：

[0040] 当污迹数据源进入到嵌入式处理器内核时，在处理器流水线上污迹标记位会随着系统对该污迹数据源进行各种各样的操作而传播。为了保持污迹标记位和数据源的完全同步，在嵌入式内核中使污迹标记位和数据源使用相同的时钟源和同步信号，将污迹标记位在数据源被读入、读出及运算的过程中和数据源绑定在一起，以追踪污迹数据源的整个传播过程，从而开辟了类似于数据流的污迹传播通道。

[0041] 本发明在处理器内核中对污迹进行传播时打开所有的传播通道，将传播过程设计成全传播形式。外来数据源进入到处理器内核后，对污迹进行全传播。如图 5 所示，污迹全传播的具体操作就是：处理器在算术逻辑单元 (ALU) 内的进行算术运算时，对所有运算操作中的污迹标记位 tag(0) 进行逻辑或运算操作；只要数据源进行运算时有一个源操作数的污迹标记位 tag(0) 为“1”，目的操作数的污迹标记位 tag(0) 就为“1”。污迹数据源在处理器内核传播时虽然可以通过设置寄存器的方法选择不同的传播通道以减小系统开销，但是污迹数据源在处理器内核的传播形式是多变的，而且在恶意程序进行非常规的形式进行攻击时，系统会存在一定的误报率。

[0042] 污迹数据源从处理器外设或端口进入到处理器内核，经一系列运算后，污迹信息被暂存到的专用于存储污迹信息的 ram 中。当污迹数据源在被使用到时，该污迹信息从 ram 中取出，随着数据的运算参与下一步的传播。

[0043] (C) 对污迹进行检测：

[0044] 外来攻击一般就是借助外来数据在操作过程中产生一些恶意的行为破坏正常程序的运行。因此本发明对污迹的检测关注的是恶意程序的行为，当带有污迹信息的数据源被不安全使用或有恶意的行为时，则产生异常报警或异常中断。

[0045] (C1) 图 6 为污迹检测寄存器。在处理器内核设置污迹检测寄存器，在指令通过流水线时，处理器通过污迹检测寄存器完成对污迹的检测。在处理器内核里系统通过污迹检测寄存器控制字的配置值打开和关闭相应的污迹检测功能。污迹检测寄存器定义了污迹检测的操作，如果检测开关被打开 (相应的使能位置 1) 同时污迹标记位 tag(0) 的值为 1 则系统将产生异常。污迹检测寄存器控制字的定义和操作如下 :

[0046] (C11) 污迹检测寄存器中的 pc 定义了对程序指针的检测规则，如果 pc 位为“1”而且程序指针跳转到 tag(0) 值为“1”的地址就会产生异常。

[0047] (C12) 污迹检测寄存器中的 inst 定义了对指令的检测规则，如果 inst 位为“1”而且系统在取指令阶段所取指令的 tag(0) 值为“1”就会产生异常。

[0048] (C13) 污迹检测寄存器中的 addrs 定义对敏感地址及敏感地址段的检测规则，如果 addrs 位为“01”而且外来攻击对敏感地址有操作，且污迹源对其内容进行改写时就产生异常 ;如果 addrs 位为“10”而且外来攻击对敏感地址段有操作，且污迹源对其内容进行改写时就产生异常。

[0049] (C14) 污迹检测寄存器中的 move 定义了 mov 运算操作中的检测规则，如果在 mov 运算中 move(0) 位为“1”而且源操作数的 tag(0) 值为“1”就会产生异常 ;如果在 mov 运算中 move(1) 位为“1”而且目的操作数的 tag(0) 值为“1”就会产生异常 ;如果在 mov 运算中 move(2) 位为“1”而且源地址的 tag(0) 值为“1”就会产生异常 ;如果在 mov 运算中 move(3) 位为“1”而且目的地址的 tag(0) 值为“1”就会产生异常。

[0050] (C15) 污迹检测寄存器中的 comp 定义了对比运算操作中的检测规则，如果 comp(0) 位为“1”而且在对比运算中源操作数的 tag(0) 值为“1”就会产生异常 ;如果 comp(1) 位为“1”而且在对比运算中目的操作数的 tag(0) 值为“1”就会产生异常。

[0051] (C16) 污迹检测寄存器中的 logic 定义了逻辑运算操作中的检测规则，如果 logic(0) 位为“1”而且在对比运算中源操作数的 tag(0) 值为“1”就会产生异常 ;如果 logic(1) 位为“1”而且在对比运算中目的操作数的 tag(0) 值为“1”就会产生异常。

[0052] 每一个污迹检测寄存器的配置对应一类攻击的防御，针对不同种类的攻击添加相应的污迹检测寄存器。而且有其所对应的污迹检测规则可进行软件配置。为了实现实时防御不同的攻击，该寄存器的配置在应用程序运行之前全部完成，而且只需要一次完成所有污迹检测寄存器的配置，避免了当防御其他类型的恶意攻击时重新配置寄存器的麻烦。

[0053] (C2) 信息安全监控方法在污迹检测过程中对威胁进行分类，将攻击源在内核中的威胁行为分为不同的级别。对威胁进行分类可以清楚地分析外来攻击对系统造成的威胁程度和影响，以便于系统检测到威胁后的能够做出准确的异常情况判断并进行相应的异常处理。

[0054] 对于 tag(3)、tag(2) 和 tag(1)，不参与传播，其值只有在特殊情况下才能修改。tag(3) 为敏感信息标记位，被标记为敏感信息的数据不能被送出设备之外，否则产生异常。tag(2) 和 tag(1) 标记了威胁的级别，这两个标记位编码为高威胁、中威胁和低威胁三种威胁级别。威胁分类的具体操作如下 :恶意程序在重点保护区或者是易被攻击区进行恶意的

修改，将其标记为高级别威胁，当进行污迹检测时如果产生异常，系统输出严重异常报警信号；恶意程序在敏感区进行恶意的修改，将其标记为中级别威胁，当进行污迹检测时如果产生异常，系统输出次严重异常报警信号；恶意程序在非敏感区域进行恶意的修改，将其标记为低级别威胁，当进行污迹检测时如果产生异常，系统输出常见异常报警信号。其中每个威胁级别对应的是源数据在CPU内进行传播时其威胁行为的一种分类。因为数据在进CPU后的传播过程中情况是多变的、而且是不能预知的，这样必然会增加在识别威胁时工作量和性能方面的开销，所以从外部输入的数据源在污迹识别中将其统统归为不可信数据。不可信数据在整个传播中都可能有恶意行为，就是这种恶意的行为会造成系统紊乱、失控甚至系统崩溃，所以对外来攻击的威胁分析是建立在行为级的层次上的。

[0055] 图7展示了在处理器流水线中实现信息安全监控方法的系统结构。除了对所有的寄存器及Caches等存储体扩展4-bits标记位，另外还需要扩展AMBA总线以兼容带标记位的存储体，并且为污迹检测添加新的异常产生机制。Leon3处理器用的是七级流线，污迹的标记、传播和检测分别添加在AMBA总线、算术逻辑单元ALU和流水线的异常阶段。当外来数据源从输入端口经过AMBA总线时污迹标记模块对其进行标记，即将数据的tag(0)置为“1”。在七级流水线中，各个阶段具体操作如下：

[0056] (1)取指阶段检查PC指针的tag(0)和来自icache的指令的tag(0)；

[0057] (2)译码阶段对每条指令进行分解；

[0058] (3)在寄存阶段系统读来自寄存器组的源操作数的标记，同时读出污迹检测寄存器的内容；

[0059] (4)执行阶段污迹传播逻辑模块对操作数的tag(0)进行传播；

[0060] (5)存储阶段污迹传播逻辑模块对操作数的tag(0)进行传播；

[0061] (6)异常阶段根据寄存阶段读出污迹检测寄存器的值控制污迹检测逻辑模块执行相应的标记检测规则，当检测到相应的tag(0)值为“1”就产生安全异常；

[0062] (7)写回阶段对寄存器组进行状态更新。

[0063] 本发明是基于嵌入式处理器架构的，通过修改处理器内核以达到保护系统安全的目的。我们选用LEON3处理器作为仿真平台验证信息安全监控的性能。为了实现对信息安全监控，我们对LEON3处理器RTL代码进行了修改，以满足信息流监控的特点。整个系统经过综合和布局布线映射到型号为xilinx virtex5 xc5vfx70t的FPGA开发板上且通过了验证。

[0064] 图8、9显示了本发明防御两种攻击的实验结果。针对信息安全监控的性能及误报率的分析，我们做了stack smashing和format string两个实验。对于stack smashing和format string两个攻击所对应的两个污迹检测寄存器，设置了相应的检测规则。我们的信息安全监控具有很好的安全特性，可以正确检测出stack smashing和format string攻击。在已搭建的攻击环境中进行试验，实验结果显示这两个攻击均属于常见威胁级别的攻击。这两个攻击模型最终都是通过buffer溢出，进而修改返回地址进行攻击的。污迹检测逻辑在外来数据源进入CPU后对其进行检测，对于stack smashing和format string攻击，当检测到带有污迹的数据覆盖了返回地址时，产生异常并根据威胁类型标志位判断为低威胁攻击。从实验结果分析由于污迹传播是全传播形式，并且应用了污迹分类，这样就大大降低了污迹追踪的误报率。由于该信息安全监控方法省略了污迹传播寄存器，在某种程

度上减少了系统的开销。

[0065] 图 10 显示了处理器加入信息安全监控前后在 FPGA 综合时在资源的利用率上的对比。具体如下 :d1 :Slice 寄存器资源对比 ;d2 :Slice 的 LUT 资源数目对比 ;d3 :用作逻辑的资源数目对比 ;d4 :用作存储器的资源数目对比 ;d5 :带有一个无用 Flip Flop 的资源数目对比 ;d6 :带有一个无用 LUT 的资源数目对比 ;d7 :全部用作 LUT-FF 对的资源数目对比 ;d8 :可接合的 IO 块资源对比 ;d9 :RAM/FIFO 块资源对比 ;d10 :BUFG 和 BUFG 的控制资源对比 ;d11 :DCM ADVs 资源对比 ;d12 :DSP48Es 资源对比。

[0066] 为了估计信息安全监控的性能,我们做了相关的实验来观察和研究当处理器运行恶意程序时,该信息安全监控对整个嵌入式系统的造成的性能开销和影响。对于性能方面的开销,我们主要估计系统在 FPGA 实现时的面积上的开销。其中加入信息安全监控后对 LUT 的利用率的增加只有 1% ;对 RAM 和 FIFO 的利用率的增加有 5% ;对其他的器件的利用率基本无增加。从资源利用率的对比中可以看出嵌入式处理器在加入该信息安全监控后性能平均开销只有 1% ～ 3%,因此在面积上的开销很小。综合的报表显示信息安全监控对系统延时的影响很小,与无信息安全监控的系统相比延时的增加平均只有 0.05ns 左右 ;同时该信息安全监控在系统在运行程序时与先前的系统相比时间的消耗几乎一样,因此该机制在时间上开销几乎为零,对该系统在速度上的影响很小。从速度和面积上来看,它对系统性能上的开销很小,对系统的影响也很小。

[0067] 本发明方法是针对外来数据在处理器内部的使用,而且信息安全监控方法基于恶意攻击的行为,与处理器运行环境和恶意攻击采用的语言无关,故可以广泛使用在提供不同服务的电信设备上。该监控方法是针对软件攻击的硬件的信息流追踪方法,相对于以前的动态信息流追踪本文提出了灵活、全面且有新意的信息流追踪方法。该信息安全监控方法可以防御多种攻击和多个并发攻击,用户可以通过软件配置防御各种新的类型的攻击,同时具有威胁分类功能。我们的设计是基于嵌入式处理器架构,通过在 RTL 级修改和添加功能模块实现的。由于本设计扩位带来了面积上的开销,必将对性能上有所影响,所以需要进一步进行性能方面进行改进。

[0068] 本发明不仅局限于上述具体实施方式,本领域一般技术人员根据本发明公开的内容,可以采用其它多种具体实施方式实施本发明,因此,凡是采用本发明的设计结构和思路,做一些简单的变化或更改的设计,都落入本发明保护的范围。

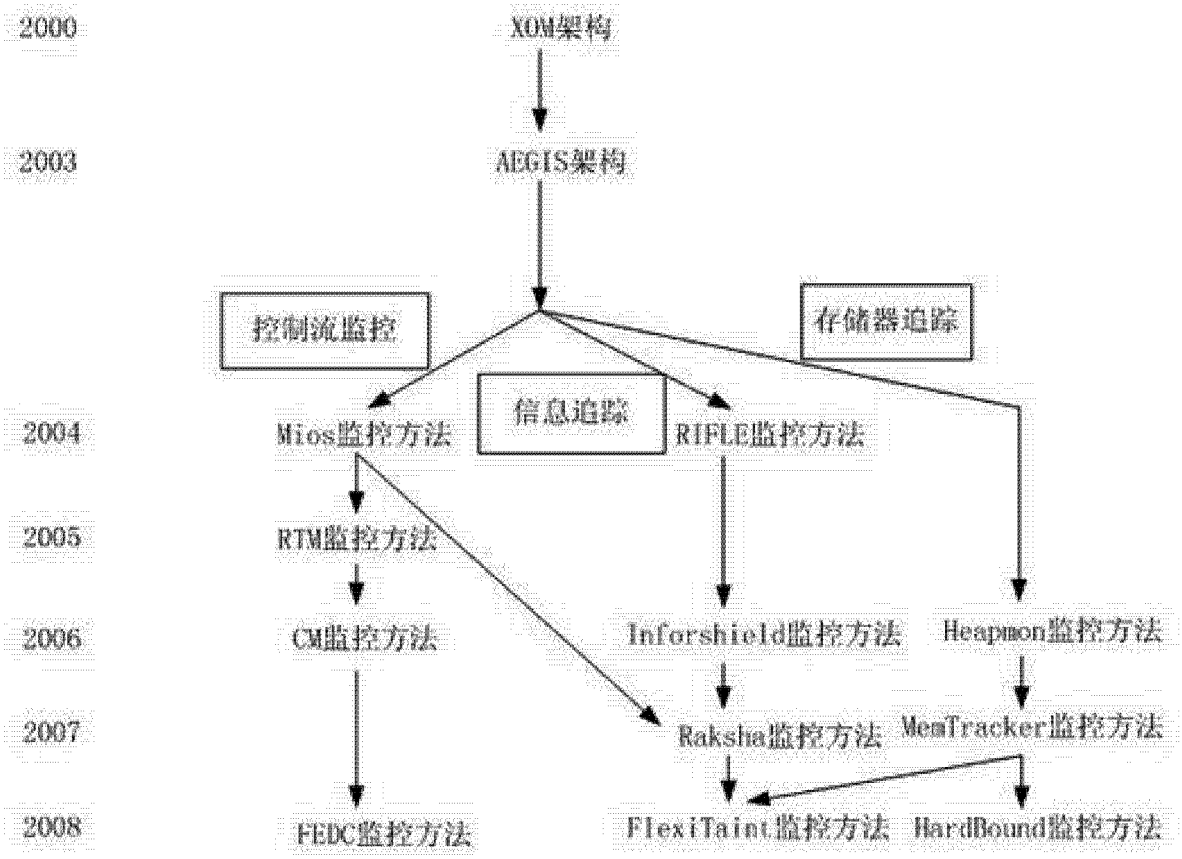

2000
XOM架构
2003
AEGIS架构
控制流监控
存储器追踪
信息追踪
2004
Mios监控方法
RIFLE监控方法
2005
RTM监控方法
2006
CM监控方法
Inforshield监控方法
Heapmon监控方法
2007
Raksha监控方法
MemTracker监控方法
2008
FEDC监控方法
FlexiTaint监控方法
HardBound监控方法

图 1

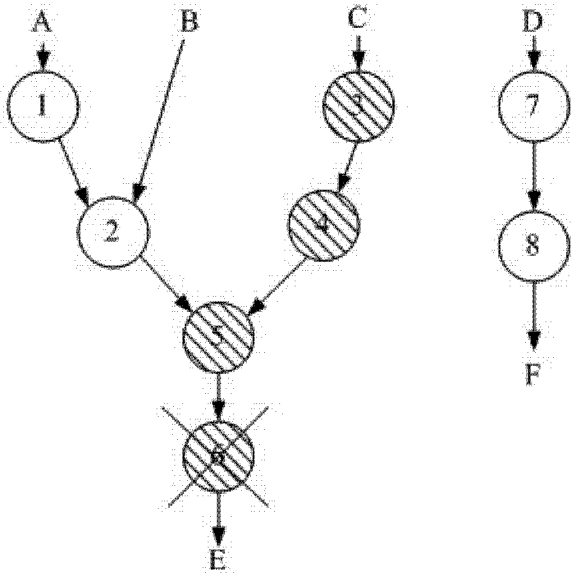

A
B
C
D
1
2
3
4
5
6
7
8
E
F

图 2

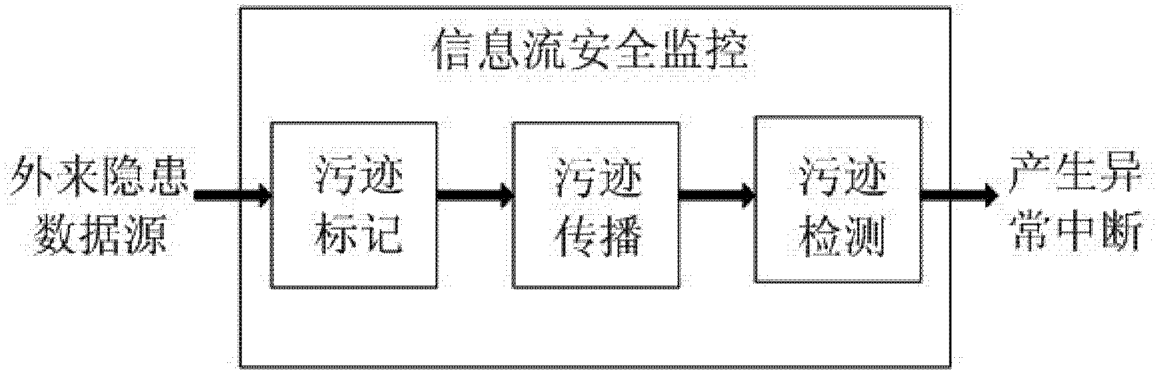


图 3

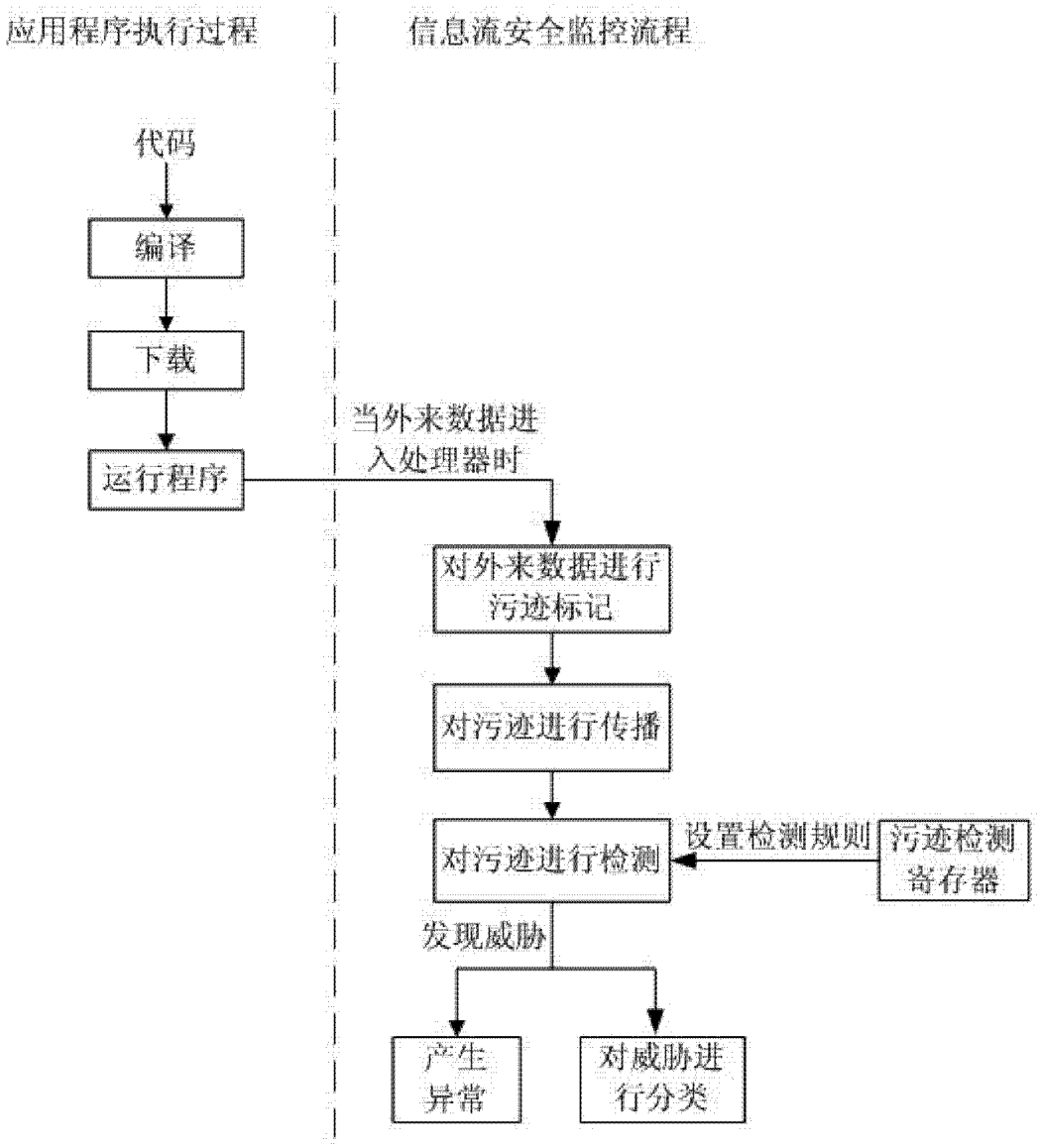

应用程序执行过程
信息流安全监控流程
代码
编译
下载
运行程序
当外来数据进入处理器时
对外来数据进行污迹标记
对污迹进行传播
对污迹进行检测
设置检测规则
污迹检测寄存器
发现威胁
产生异常
对威胁进行分类

图 4

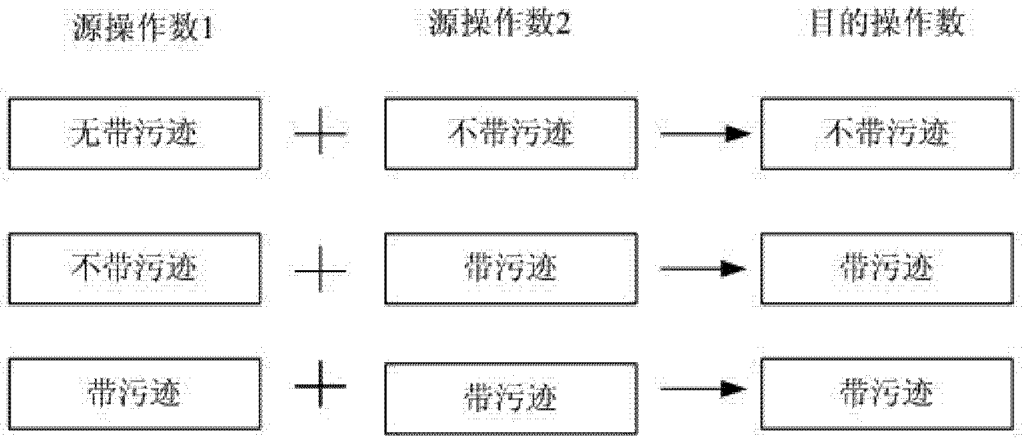

源操作数1
源操作数2
目的操作数
无带污迹
不带污迹
不带污迹
不带污迹
带污迹
带污迹
带污迹
带污迹
带污迹

图 5

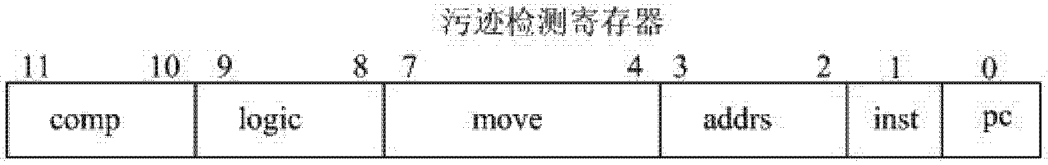

污迹检测寄存器
11
10
9
8
7
4
3
2
1
0
comp
logic
move
addrs
inst
pc

图 6

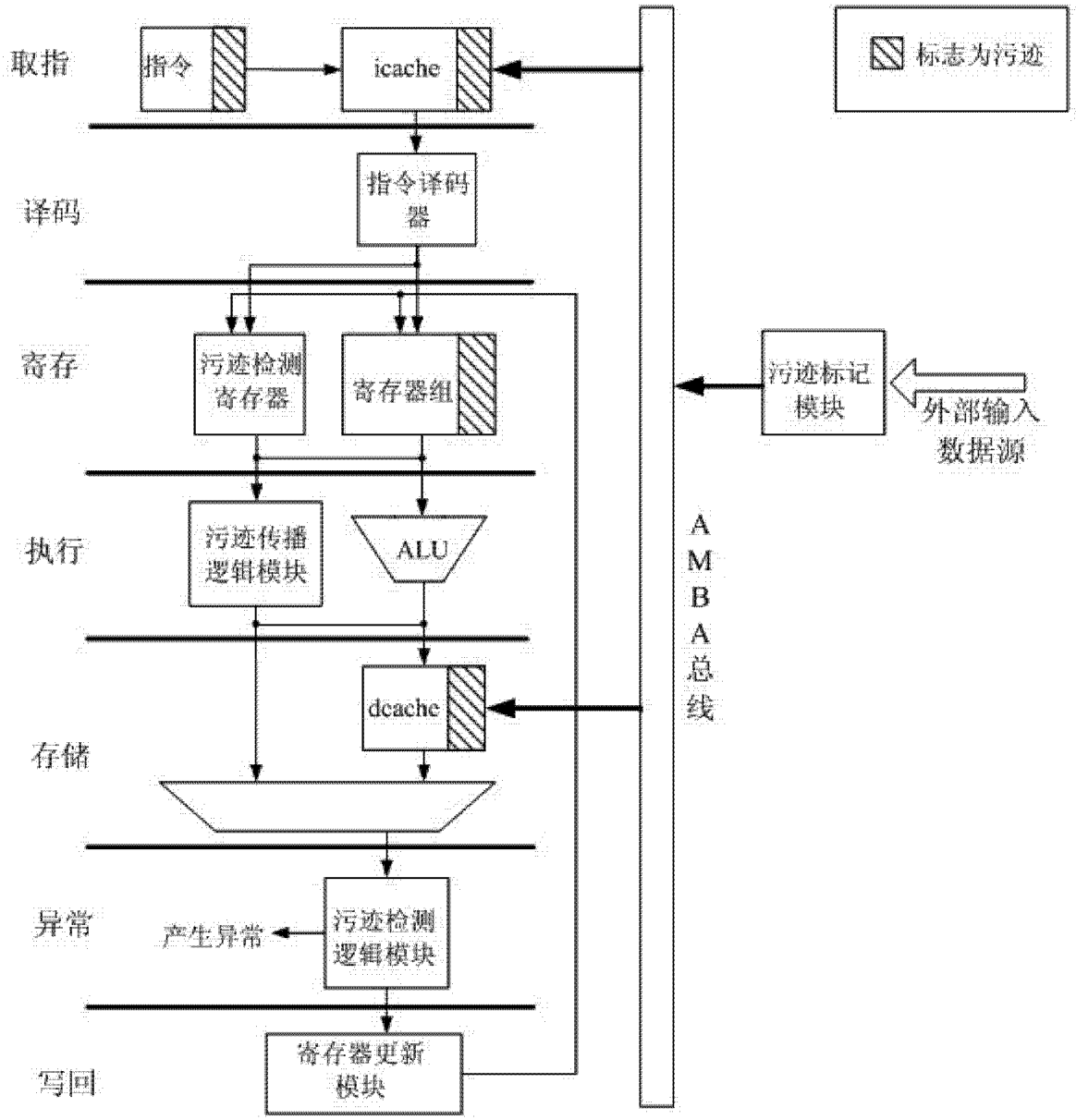

取指
指令
icache
标志为污迹
译码
指令译码器
寄存
污迹检测寄存器
寄存器组
污迹标记模块
外部输入数据源
执行
污迹传播逻辑模块
ALU
AMBA总线
dcache
存储
异常
产生异常
污迹检测逻辑模块
写回
寄存器更新模块

图 7

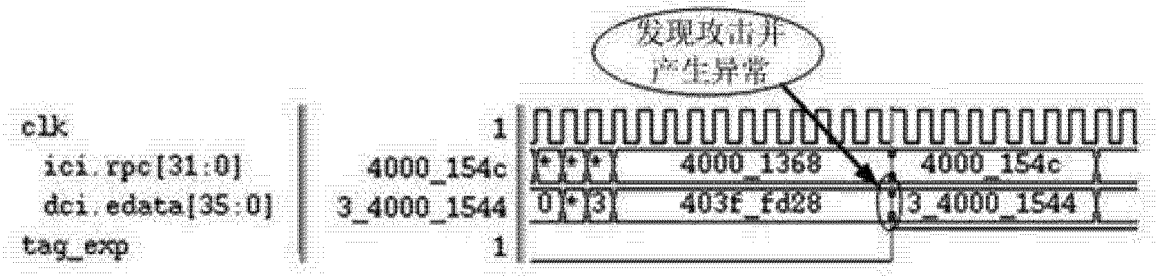

发现攻击并产生异常
clk
ici.rpc[31:0]
dci.edata[35:0]
tag_exp
1
4000_154c
3_4000_1544
1
4000_1368
4000_154c
0
3
403f_fd28
3_4000_1544

图 8

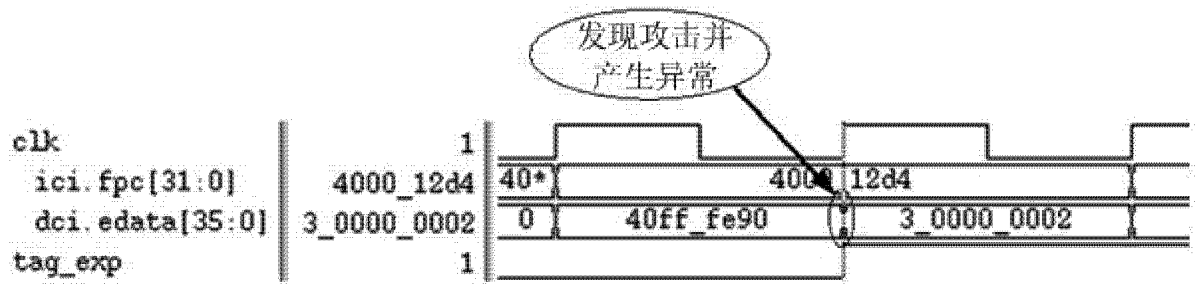

发现攻击并
产生异常
clk
ici.fpc[31:0]
dci.edata[35:0]
tag_exp
1
4000_12d4
3_0000_0002
1
40*
0
40ff_fe90
12d4
3_0000_0002

图 9

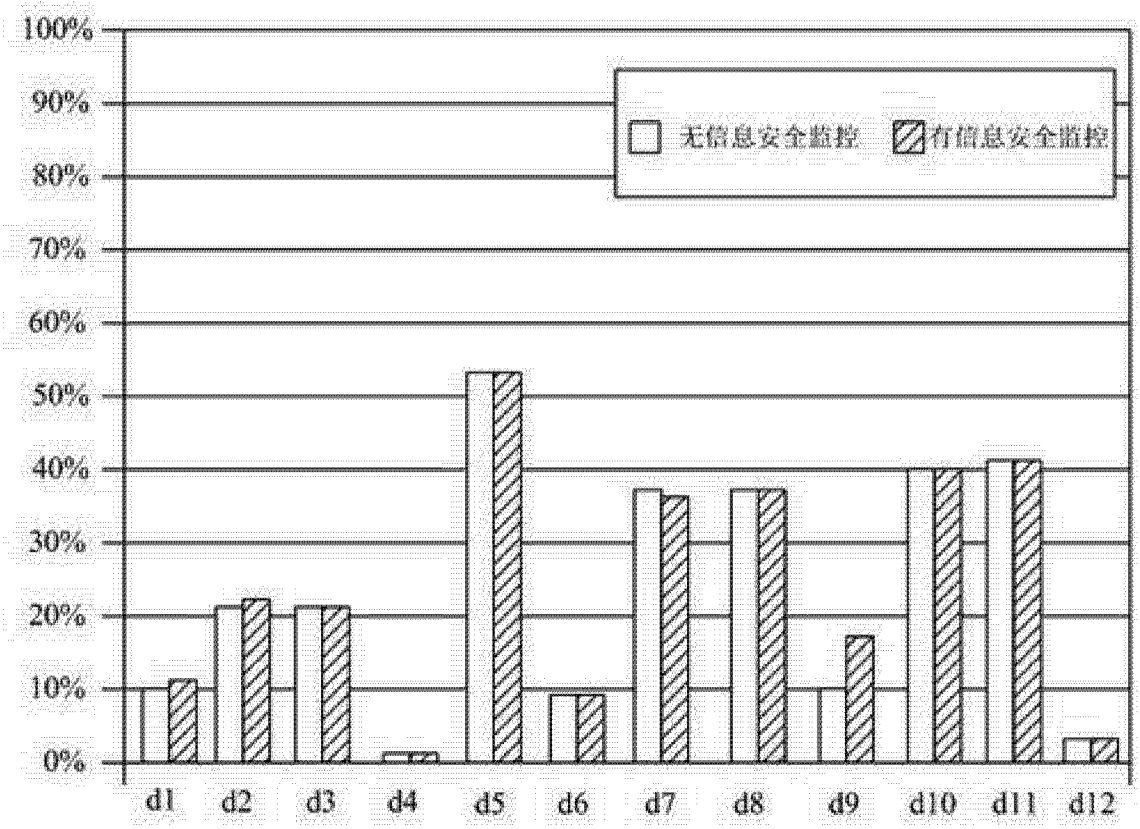

100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
无信息安全监控
有信息安全监控
d1
d2
d3
d4
d5
d6
d7
d8
d9
d10
d11
d12

图 10