[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 21/22 (2006.01)

[12] 发明专利说明书

专利号 ZL 200610057400.1

[45] 授权公告日 2009 年 1 月 21 日

[11] 授权公告号 CN 100454326C

[22] 申请日 2006.3.14
[21] 申请号 200610057400.1
[30] 优先权
[32] 2005. 3.14 [33] JP [31] 2005 - 071823
[73] 专利权人 株式会社 NTT 都科摩
地址 日本东京都
[72] 发明人 铃木敬 金野晃 藤本拓 中山雄大
[56] 参考文献
CN1239787A 1999.12.29
US2005/0050354A1 2005.3.3
CN1556959A 2004.12.22
审查员 张 千

[74] 专利代理机构 北京银龙知识产权代理有限公司
代理人 许 静

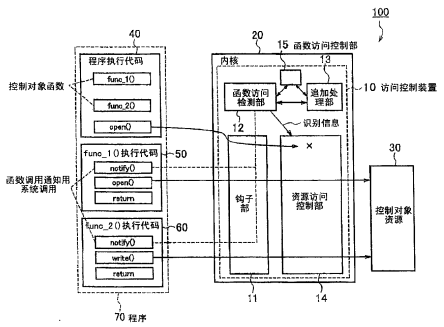
权利要求书 2 页 说明书 12 页 附图 7 页

[54] 发明名称

访问控制装置以及访问控制方法

[57] 摘要

一种访问控制装置，其控制通过程序向控制对象资源的访问，具有：函数访问检测部，其检测来自程序的对控制对象函数的调用，并生成确定从该检测到的控制对象函数所发行的访问请求的识别信息；资源访问控制部，其从所述程序取得对所述控制对象资源的访问请求，并根据所述识别信息判定可否进行所述访问请求。



1. 一种访问控制装置，其控制通过程序向控制对象资源的访问，其特征在于，具有：

函数访问检测部，其检测来自程序的对控制对象函数的调用，并生成识别信息，该识别信息确定从该检测到的控制对象函数所发行的访问请求；

资源访问控制部，其从所述程序取得对所述控制对象资源的访问请求，并根据所述识别信息判定可否进行针对所述控制对象资源的访问请求。

2. 根据权利要求 1 所述的访问控制装置，其特征在于，

还具有函数访问控制部，其取得所述控制对象函数的调用源进程的信息，并根据函数许可用访问控制规则判定可否进行所述控制对象函数的调用，

所述函数访问检测部在允许了调用的情况下，生成所述识别信息。

3. 根据权利要求 1 所述的访问控制装置，其特征在于，

还具有在检测到所述控制对象函数的调用时，执行预先规定的处理的追加处理部。

4. 根据权利要求 2 所述的访问控制装置，其特征在于，

还具有执行预先规定的处理的追加处理部，所述函数访问控制部在所述追加处理部执行了预先规定的处理的情况下，允许所述控制对象函数的调用。

5. 根据权利要求 1 所述的访问控制装置，其特征在于，

所述资源访问控制部除了所述识别信息之外，可以根据资源许可用访问控制规则判定可否进行针对所述控制对象资源的访问请求，

在所述资源许可用访问控制规则中记述了对访问进行控制的规则，该访问是通过规定的所述控制对象函数而进行的、针对所述控制对象资源的访问。

6. 根据权利要求 1 所述的访问控制装置，其特征在于，

所述函数访问检测部通过取得为了通知所述控制对象函数的调用而设置的通知命令，来检测对所述控制对象函数的调用，

在所述通知命令中包含所述识别信息或生成所述识别信息所必需的信息。

7. 根据权利要求 6 所述的访问控制装置，其特征在于，

所述函数访问检测部在确认了正在从所述控制对象函数发行所述通知命令的情况下，生成所述识别信息。

8. 根据权利要求 1 所述的访问控制装置，其特征在于，

所述函数访问检测部在调用了所述控制对象函数的情况下，将所述控制对象函数的函数名堆积在堆栈；在处理返回所述控制对象函数的调用源进程的情况下，从堆栈除去所述控制对象函数的函数名，并作为所述识别信息生成在所述堆栈中所存储的信息，

所述资源访问控制部根据所述控制对象函数的函数名是否正堆积在所述堆栈中来判定可否进行针对所述控制对象资源的访问请求。

9. 根据权利要求 1 所述的访问控制装置，其特征在于，

所述函数访问检测部对所述控制对象函数的执行代码在实际存储器上的存储地址范围进行计算，作为所述识别信息生成所述存储地址范围，

所述资源访问控制部参照进程的程序计数器，根据发行了所述资源访问请求的命令代码的存储地址是否在所述存储地址范围内，来判定可否进行针对所述控制对象资源的访问请求。

10. 一种访问控制方法，其控制通过程序向控制对象资源的访问，其特征在于，具有如下步骤：

检测来自程序的对控制对象函数的调用，并生成识别信息，该识别信息确定来自该检测出的控制对象函数的访问请求；

从所述程序取得对所述控制对象资源的访问请求，根据所述识别信息判定可否进行针对所述控制对象资源的访问请求。

访问控制装置以及访问控制方法

相关技术的交叉引用

本申请基于先前于 2005 年 3 月 14 日递交的日本专利申请 No.P2005－71823 并享受其优先权的好处；其全部内容被收容于本申请中，以资参考。

技术领域

本发明涉及一种控制通过程序向控制对象资源的访问的访问控制装置以及访问控制方法。

背景技术

经由网络等得到程序，并在个人计算机或 PDA、移动电话中进行执行的使用方式正在普及。这样的使用方式由于可以迅速地得到所需要的应用程序，所以具有可以提高用户的便利性的效果。另一方面，在通过网络流通的程序代码中存在以破坏系统或非法操作为目的的具有恶意的代码、引起系统资源的乱用或动作故障等的具有缺陷的代码。随着下载应用程序的使用扩大，确保用户的便利行并确保程序执行的安全性成为紧要的课题，防止由于具有恶意或缺陷的代码（非法代码）导致的伤害的机构的重要性正在提高。

作为防止非法代码伤害的机构，具有被称为沙箱（sandbox）的、限制了向资源的访问的程序代码的执行环境。沙箱具有访问控制功能，该功能按照被称为安全策略的访问控制规则，控制对来自在沙箱上执行的程序代码（被沙箱化的代码）的文件或网络等进行的访问。这里，为了确保沙箱的安全性，需要防止被沙箱化的代码旁路（bypath）访问控制处理来对资源进行直接访问。

作为防止旁路访问控制的有力方法，具有使用了系统调用钩子（hook）的沙箱。系统调用是在程序代码对文件或网络等资源进行访问时对操作系统（OS）进行发行的请求。因为不经由系统调用操作计算机的资源是困难的，所以通过以系统调用为中介（以下称为「钩子」。）附加访问控制处理，由此，旁路变得困难。（例如参照 Using Kernel Hypervisors to Secure Applications,in

Proceedings of the Annual Computer Security Application Conference,1997。)。

但是，系统调用是OS的低等级接口，OS难以由已挂钩的系统调用的信息来取得程序的意思信息。因此，存在以下的课题：在通过系统调用钩子的访问控制中，对程序的意思信息进行了考虑的控制/处理是困难的。在无法得到程序的意思信息的情况下，有如下问题：需要对低等级的接口设定访问控制规则，需要与系统有关的较深知识和较费功夫。此外，还存在以下的问题：通过系统调用等级来描述被称为「允许备份程序安装或取得履历为条件」的允许附加条件等基于程序意思信息的柔性访问规则是困难的。

因无法取得意思信息而引起的课题可以通过使用了库（library）函数调用的钩子的沙箱来解决。具有以下的实现方法：把包含控制对象的函数的库置换为包含向沙箱的钩子处理的库（例如参照DITools:Application Level Support for Dynamic Extension and Flexible Composition,in Proceedings of 2000 USENIX Annual Technical Conference,2000）。由此，可以对具有高抽象度的接口追加访问控制处理，可以高效率地设定基于意思信息的多样的访问规则。

但是，在仅依存于库函数调用钩子的沙箱中，无法防止称为直接发行系统调用的旁路攻击，存在安全性方面的重大问题。

发明内容

因此，本发明是鉴于上述课题而发明的，其目的在于提供一种实现基于程序意思信息的高度的资源访问控制、和防止旁路的强安全性两者的访问控制装置以及访问控制方法。

一种访问控制装置，其控制通过程序向控制对象资源的访问，具有函数访问检测部，其检测来自程序的对控制对象函数的调用，并生成识别信息，该识别信息确定从该检测到的控制对象函数所发行的访问请求；资源访问控制部，其从所述程序取得对所述控制对象资源的访问请求，并根据所述识别信息判定可否进行所述访问请求。

一种访问控制方法，其控制通过程序向控制对象资源的访问，具有如下步骤：检测来自程序的对控制对象函数的调用，并生成识别信息，该识别信息确定来自该检测出的控制对象函数的访问请求；以及从所述程序取得对所述控制对象资源的访问请求，根据所述识别信息判定可否进行所述访问的请

求。

附图说明

图 1 是使用了本实施方式的访问控制装置的系统的说明图。

图 2 是本实施方式的访问控制装置的结构方框图。

图 3 是本实施方式的资源访问控制规则的一个例子。

图 4 是表示本实施方式的访问控制方法的流程图。

图 5 是安装了本实施方式的程序之后的状态的说明图。

图 6 是用于说明对函数调用通知用系统调用的发行源进行验证的说明图。

图 7 是用于对本实施方式的访问控制装置的堆栈（stack）进行说明的图。

具体实施方式

附图描述了本发明的各种实施方式。在附图中对相同或相近的部分及要素赋予相同或相近的数字，并且对于说明书中相同或相近的部分及要素予以省略或者进行简单说明。

（访问控制装置）

图 1 表示对本发明实施方式的应用程序进行发布以及执行的系统形态。在本系统中，个人计算机或 PDA、移动电话等计算机 100 执行 Web 浏览器或邮件客户端、游戏等各种各样的程序。这里，在计算机 100 执行的程序中包含在计算机 100 售出时已经安装的程序、从 CD－ROM、DVD－ROM 等记录媒体 400 复制的程序、通过网络 200 或基站 500 从应用程序下载服务器 300 下载的程序等，可靠度和品质不同。

如图 2 所示，本实施方式的计算机 100 通过具备访问控制装置 10，保护用户数据、系统文件、网络等控制对象资源 30 远离非法的程序。具体地说，访问控制装置 10 控制通过各个系统调用（例如 open()、write()）向控制对象资源 30 的访问。

访问控制装置 10 具备：钩子部 11，其取得程序 70 对内核（kernel）20 发行的系统调用，并传送给预先设定的函数访问检测部 12 以及资源访问控制部 14；函数访问检测部 12，其检测来自程序 70 的对控制对象函数的调用，并生成识别信息，该识别信息确定是从所检测到的控制对象函数发行的访问

请求；函数访问控制部 15，取得控制对象函数的调用源进程的信息，根据函数许可用访问控制规则判定可否调用控制对象函数；资源访问控制部 14，从程序 70 取得对控制对象资源 30 的访问请求，根据识别信息判断可否请求访问；追加处理部 13，在函数访问检测部 12 允许控制对象函数的调用的情况下，执行预先规定的处理。

在图 2 中，在程序执行代码 40 中包含多个函数调用，将 func_1()、func_2()作为控制对象函数。此外，在 func_1()执行代码 50、func_2()执行代码 60 的开始，存在作为用于对访问控制装置 10（内核 20）通知函数调用的通知命令（称为「函数调用通知用系统调用」）的 notify()，当处理移动到控制对象函数（func_1()、func_2()）时，发行 notify()。

对系统调用 notify()的自变量设定控制对象函数名等、在内核 20 确定从控制对象函数发行的系统调用时所需要的识别信息或用于生成识别信息的关联信息。例如，可以将函数执行代码在存储器上的存储地址或用于计算存储地址的虚拟地址、在堆栈堆积的信息等作为识别信息来使用。对于识别信息在后面进行详细的叙述。

然后，对访问控制装置 10 的各个处理部进行详细的说明。

钩子部 11 取得函数调用通知用系统调用 notify()，并传送给函数访问检测部 12。此外，钩子部 11 把请求向控制对应资源 30 进行访问的系统调用传送给资源访问控制部 14。

函数访问检测部 12 检测对来自程序 70 的控制对象函数的调用，生成确定是从该检测到的控制对象函数发行的系统调用的识别信息。

例如，函数访问检测部 12 由系统调用 notify()的自变量取得控制对象函数名和控制对象函数调用源进程（例如程序执行代码 40）的信息，根据函数许可用访问控制规则判断可否调用控制对象函数。这里，在「函数许可用访问控制规则」中记述了是否为允许调用的控制对象函数。

此外，函数访问检测部 12 在判断可否调用之前验证函数调用通知用系统调用 notify()是否没有被非法调用，仅在确认了从该控制对象函数发行系统调用 notify()的情况下可以继续进行处理。关于具体的确认方法将在后面进行详细的叙述。

函数访问控制部 15 取得控制对象函数的函数名和控制对象函数的调用源进程的信息，根据函数许可用访问控制规则判定可否调用控制对象函数。然后，函数访问检测部 12 在允许调用的情况下生成识别信息。

追加处理部 13 在函数访问控制部 15 允许控制对象函数的调用的情况下，执行预先规定的处理。这里，在上述的「函数许可用访问控制规则」中也可以记述以特定处理为条件地允许函数调用的条件附加许可规则。在允许附加条件情况下，函数访问检测部 12 将处理移动至追加处理部 13，执行作为条件记述的处理。

作为在条件中设定的处理，例如具有：伴随函数调用的向资源访问履历的记录（log）、取得控制对象资源 30 的备份等。

此外，追加处理部 13 执行预先已规定的处理，函数访问控制部 15 在所述追加处理部执行了预先已规定的处理的情况下，可以允许调用控制对象函数。

此外，在允许函数调用的情况下，函数访问检测部 12 使用函数调用通知用系统调用自变量的信息和可以从内核 20 取得的信息的一方或双方，生成对来自该函数的系统调用发行进行检测时所必需的识别信息。

资源访问控制部 14 从程序 70 取得对控制对象资源 30 的访问请求，根据函数访问检测部 12 所生成的识别信息以及资源许可用访问控制规则，判定系统调用的访问可否进行。这里，如图 3 所示，在「资源访问控制规则」中记述了控制对象资源 30 的信息、访问请求源的程序 70 的信息、函数访问检测部 12 生成的识别信息、通过许可动作组记述的规则。作为进行控制的规则，可以列举出仅规定的控制对象函数可以访问规定的控制对象资源 30、仅规定的控制对象函数拒绝规定的控制对象资源 30。例如，在资源访问控制规则中记述了以下的规则：无条件地允许来自计算机 100 售出时所安装的程序的系统调用，但对于下载的程序仅允许由识别信息确定的系统调用。此时，资源访问控制部 14 对于下载的程序，仅通过函数中被允许调用的函数，允许对控制对象资源 30 进行访问。

（访问控制方法）

然后，使用图 4 对本实施方式的访问控制方法进行说明。

首先，作为前提如图 5 所示，设为：通过编译程序源（programme source）来生成程序目标文件（programme object file），在程序执行文件上配置应用程序执行代码。此外，通过与静态链接库（static link library）进行链接，作为本实施方式控制对象函数的 func_1()、func_2()的执行代码被配置在程序执行文件上。此外，在程序执行文件上配置表示了各个执行代码的配置信息的再配置信息、作为用于动态地链接公共库的信息的动态（dynamic）信息等。通过安装这些程序，在实际存储器地址上配置应用程序执行代码、func_1()执行代码、func_2()执行代码。此外，通过与公共库进行链接，也可以配置作为库目标的 func_3()执行代码。

在图 4 的步骤 S101 中，钩子部 11 取得为了通知控制对象函数的调用而设置的通知命令（例如函数调用通知用系统调用 notify()）。然后，钩子部 11 将通知命令传送给函数访问检测部 12。

然后，在步骤 S102，函数访问检测部 12 从程序检测对于控制对象函数的调用。

然后，在步骤 S103，函数访问检测部 12 验证函数调用通知用系统调用 notify()是否没有被非法调用。例如，通过参照程序计数器等方法取得系统调用 notify()发行源的存储器地址，通过验证是否与由系统调用自变量确定的控制对象函数的存储地址范围一致，可以进行确认。如图 5 所示，控制对象函数的存储地址例如在程序执行代码的头（head）区域记述向函数目标的偏移（offset），通过将偏移与程序的偏移目的地地址相加，可以计算 func_1()偏移地址、func_2()偏移地址。

此外，在控制对象函数（func_3()）包含在公共库中的情况下，按照图 6 所示的顺序，可以验证 notify 系统调用的发行源。设在库中包含 notify 系统调用的发行函数，从控制对象函数调用库内的 notify 系统调用发行函数。通过函数访问检测部 12 可以对表示配置了公共库的地址范围的地址解决表进行管理，判定 notify 系统调用的发行源地址是否位于上述地址范围内。

在函数调用通知用系统调用 notify()是恰当的情况下，进入到步骤 S104，在不恰当的情况下进入到步骤 S111，进行显示函数调用通知用系统调用 notify()不恰当的错误处理，并结束处理。

然后，在步骤S104，函数访问检测部12从系统调用notify()的自变量取得控制对象函数名和控制对象函数调用源进程（例如程序执行代码40）的信息，根据函数许可用访问控制规则判断可否进行控制对象函数的调用。在允许调用的情况下，进入步骤S105，在不允许的情况下进行步骤S11，进行显示没有控制对象函数调用权限等消息的错误处理，并结束处理。

然后，在步骤S105，函数访问检测部12判断在函数许可用访问控制规则中是否记述了以特定处理为条件允许函数调用的条件附加许可规则。在允许附加条件的情况下，进入步骤S106；在不允许附加条件的情况下进入步骤S107。

然后，在步骤S106，追加处理部13执行作为条件所记述的处理。

然后，在步骤S107，函数访问检测部12生成确定是从该检测到的控制对象函数所发行的系统调用的识别信息。

例如，函数访问检测部12在调用了控制对象函数的情况下，将控制对象函数的函数名堆积在堆栈；在处理返回控制对象函数的调用源进程的情况下，从堆栈除去控制对象函数的函数名，并作为识别信息生成在堆栈所存储的信息。具体地说，如图7所示，在对在自变量中包含了控制对象函数的函数名的、函数调用通知用系统调用（notify1()）进行接收时，将函数名（func_1()、func_2()）堆积在堆栈信息，在接收函数返回通知用系统调用（notify2()）时，从堆栈信息去除函数名（func_1()、func_2()），通过以上顺序可以生成识别信息。

而且，函数访问检测部12仅在取得通知命令时的程序正在执行时、代码的存储器地址包含在函数目标文件的存储范围内的情况下，可以把从通知命令所取得的控制对象函数名堆积在堆栈。

除此之外，函数访问检测部12计算控制对象函数的执行代码的实际存储器上的存储地址的范围，可以作为识别信息生成存储地址范围。具体地说，如图4所示，当在函数调用通知用系统调用的自变量中设定了函数执行代码的地址偏移和大小的情况下，取得已安装了程序执行代码的实际存储器地址，通过将偏移与大小进行相加来计算函数执行代码的实际存储器上的存储地址范围。然后，作为识别信息生成计算出的存储地址范围。此外，在函数调用

通知用系统调用的自变量中也可以直接包含存储器地址范围。

而且，在函数调用通知用系统调用的自变量中也可以包含控制对象函数的执行代码的虚拟存储器地址范围。此时，函数访问检测部 12 根据虚拟存储器地址范围求出配置了执行代码的实际存储器地址范围，作为识别信息生成实际存储器地址范围。作为求出实际存储器地址范围的方法，例如虚拟存储器地址范围，根据函数调用通知用系统调用的执行代码设为是控制对象函数的最后的执行代码的地址偏移，函数访问检测部 12 通过将上述地址偏移与接收函数调用通知用系统调用时的正在执行的代码的存储器地址相加，由此可以求出实际存储器地址范围。

此外，也可以将上述堆栈信息和地址范围进行组合来作为识别信息。例如，可以使在堆栈堆积的函数信息和函数执行代码的地址范围相互对应来生成识别信息。

然后，在步骤 S108，钩子部 11 取得系统调用（例如 open()、write()），并参照系统调用自变量检测对控制对象资源 30 的访问请求。然后，钩子部 11 把向控制对象资源 30 的访问请求传送给资源访问控制部 14。

然后，在步骤 S109，资源访问控制部 14 根据函数访问检测部 12 所生成的识别信息，判定可否进行系统调用的访问。在允许的情况下，进入步骤 S110，允许向控制对象资源 30 的访问。另一方面，在不允许的情况下，进入步骤 S112，进行显示没有资源访问权限的错误处理，拒绝向控制对象资源 30 的访问。在图 2 中，在 func_1()执行代码 50、func_2()执行代码 60 中包含的系统调用 open()、write()由资源访问控制部 14 许可了向控制对象资源 30 的访问，但在程序执行代码 40 中包含的系统调用 open()由资源访问控制部 14 拒绝了访问。

此外，作为可否访问的判定方法，例如在作为识别信息使用在堆栈中存储的信息的情况下，资源访问控制部 14 根据在堆栈中是否堆积了控制对象函数的函数名，判定可否进行系统调用的访问（在已堆积时允许系统调用）。

此外，在作为识别信息正在使用存储地址范围的情况下，资源访问控制部 14 参照进程的程序计数器，根据发行了系统调用的命令代码的存储地址是否在存储地址的范围内，判定可否进行系统调用的访问。

此外，在作为识别信息正在使用把上述堆栈信息和地址范围组合起来的信息的情况下，资源访问控制部 14 验证控制对象函数正堆积在堆栈和发行了系统调用的命令代码的存储地址正收容在函数执行代码的存储地址范围内，在满足两个条件的情况下，进行允许系统调用的判定处理。

此外，在上述的说明中，使用单一的流程图对钩子部 11、函数访问检测部 12 以及资源访问控制部 14 的动作进行了说明，但函数访问检测部 12 和资源访问控制部 14 可以作为单独的模块进行动作。

（作用以及效果）

当使用本实施方式的访问控制装置 10 以及访问控制方法时，生成确定是从检测到的控制对象函数发行的系统调用的识别信息，通过根据该识别信息判定可否进行向资源的访问，可以根据识别信息将控制对象函数调用信息反映在对低等级的资源访问请求的访问控制中，可以同时确保函数调用单位的访问控制的高抽象度和低等级资源访问请求单位的访问控制的强安全性两者。由此，可以高效率地记述基于程序的意思信息的多样的访问规则，可以安全地执行多功能的程序。

此外，函数访问控制部 15 取得控制对象函数的函数名和控制对象函数的调用源进程的信息，根据函数许可用访问控制规则判定可否进行控制对象函数的调用。然后，函数访问控制部 12 在允许了调用的情况下，可以生成识别信息。因此，可以拒绝非法的控制对象函数的调用。

此外，当使用本实施方式的访问控制装置 10 以及访问控制方法时，在允许了控制对象函数的调用的情况下，可以执行预先规定的处理。或者，在执行了预先规定的处理的情况下，可以允许进行控制对象函数的调用。因此，可以对控制对象函数调用进行挂钩，插入预先所规定的处理。由此，例如可安装以备份的取得或履历的取得等处理的执行为条件的高度的访问控制规则，可以应对应用程序或系统的多样的必要条件。

此外，资源访问控制部 14 除了识别信息之外，可以根据资源许可用访问控制规则判定可否进行系统调用的访问。因此，关于向特定资源的访问，可以附加使用特定函数的义务。由此，可以仅通过信赖的函数允许向特性资源的访问，可以提高程序执行的安全性。

此外，函数访问检测部 12 通过取得为了通知控制对象函数的调用而设置的通知命令（例如 notify ()），可以检测对控制对象函数的调用。因此，可以通过对该通知命令进行挂钩检测控制对象函数的调用，来取得通知命令的自变量中包含的识别信息或用于生成识别信息的关联信息，可以通过简易的处理实现函数调用检测。由此，可以削减本访问控制装置以及方法的处理成本和安装成本。

此外，函数访问检测部 12 在确认了上述通知命令是从控制对象函数发行的情况下，可以生成识别信息。因此，例如在程序为了非法访问资源而假冒发行通知控制对象函数调用的命令时，可以将其检测到。由此可以进一步提高程序执行的安全性。

此外，函数访问检测部 12 生成已在堆栈存储的信息来作为识别信息，资源访问控制部 14 可以根据是否在堆栈堆积了控制对象函数的函数名来判定可否进行通过系统调用的资源访问。因此，即使在控制对象函数进一步调用了函数的情况下，因为函数的调用履历包含在堆栈信息中，所以可以识别通过控制对象函数以及其下级函数的资源访问。

此外，上述通知命令包含控制对象函数名，函数访问检测部 12 可以把根据通知命令得到的控制对象函数名堆积在堆栈。这样，通过对该通知命令进行挂钩可以检测控制对象函数调用并取得函数名，可以通过简单的处理实现函数调用检测。由此，可以削减本访问控制装置以及方法的处理成本和安装成本。

此外，函数访问检测部 12 仅在函数目标文件的存储范围中包含了取得通知命令时的程序的执行中代码的存储器地址时，可以把从通知命名取得的控制对象函数名堆积在堆栈。由此，通知命令可以验证是否正在从在程序头中记述的目标存储范围进行发行。例如，在程序为了非法访问资源而假冒发行通知控制对象函数调用的命令时，可以将其检测到。由此，可以进一步提高程序执行的安全性。

此外，函数检测部 12 对控制对象函数的执行代码在实际存储器上的存储地址范围进行计算，作为识别信息生成存储地址范围，资源访问检测部 14 参照进程的程序计数器，可以根据发行了系统调用的命令代码的存储地址是否

在存储地址范围内来判定可否进行系统调用的访问。因此，根据配置了函数的执行代码的存储器地址范围，可以验证向控制对象资源的访问请求是否是从控制对象函数发行的。

此外，上述通知命令包含控制对象函数的执行代码的存储器地址，函数访问检测部 12 可以生成根据通知命令得到的存储器地址范围来作为识别信息。如此，通过对该通知命令进行挂钩可以检测控制对象函数调用并取得存储器地址范围，可以通过简易的处理实现控制对象函数调用的检测。由此，可以削减本访问控制装置以及方法的处理成本和安装成本。

此外，上述通知命令包含控制对象函数的执行代码的虚拟存储器地址范围，函数访问检测部 12 从虚拟的存储器地址范围求出配置了执行代码的实际存储器地址范围，并可以作为识别信息生成实际存储器地址范围。这样，可以从包含在通知命令中的虚拟存储器地址范围求出存储函数的执行代码的实际存储器地址范围。由此，即使在生成程序执行代码时无法导出实际存储器地址范围的情况下，也可以使用本访问控制装置以及方法。

此外，将虚拟存储器地址范围，根据函数调用通知用系统调用的执行代码设为是控制对象函数的最后的执行代码的地址偏移，函数访问检测部 12 通过将上述地址偏移与接收函数调用通知用系统调用时的正在执行的代码的存储器地址相加，由此可以求出实际存储器地址范围。如此，可以从包含在通知命令中的虚拟存储器地址范围求出存储函数的执行代码的实际存储器地址范围。由此，即使在生成程序执行代码时无法导出实际存储器地址范围的情况下，也可以使用本访问控制装置以及方法。

（其他实施方式）

通过第一以及第二实施方式对本发明进行了记述。但应该理解为部分构成本发明的实施方式以及附图并没有限定本发明。通过本发明的公开，本领域的技术人员可以清楚本发明不同其他的实施例以及操作技术。

例如，在图 2 中采用了将访问控制装置 10 安装在操作系统的内核 20 的内部的结构，但也可以将访问控制装置 10 安装在其他的位置。

此外，在上述的实施方式中，对判定可否进行来自系统调用的访问请求进行了说明，但本发明并不限于系统调用，即使是来自其他程序的访问请求

也可以适用。

在领会了本发明的内容并且不偏离本发明范围的情况下，技术人员可能提出不同的修改。

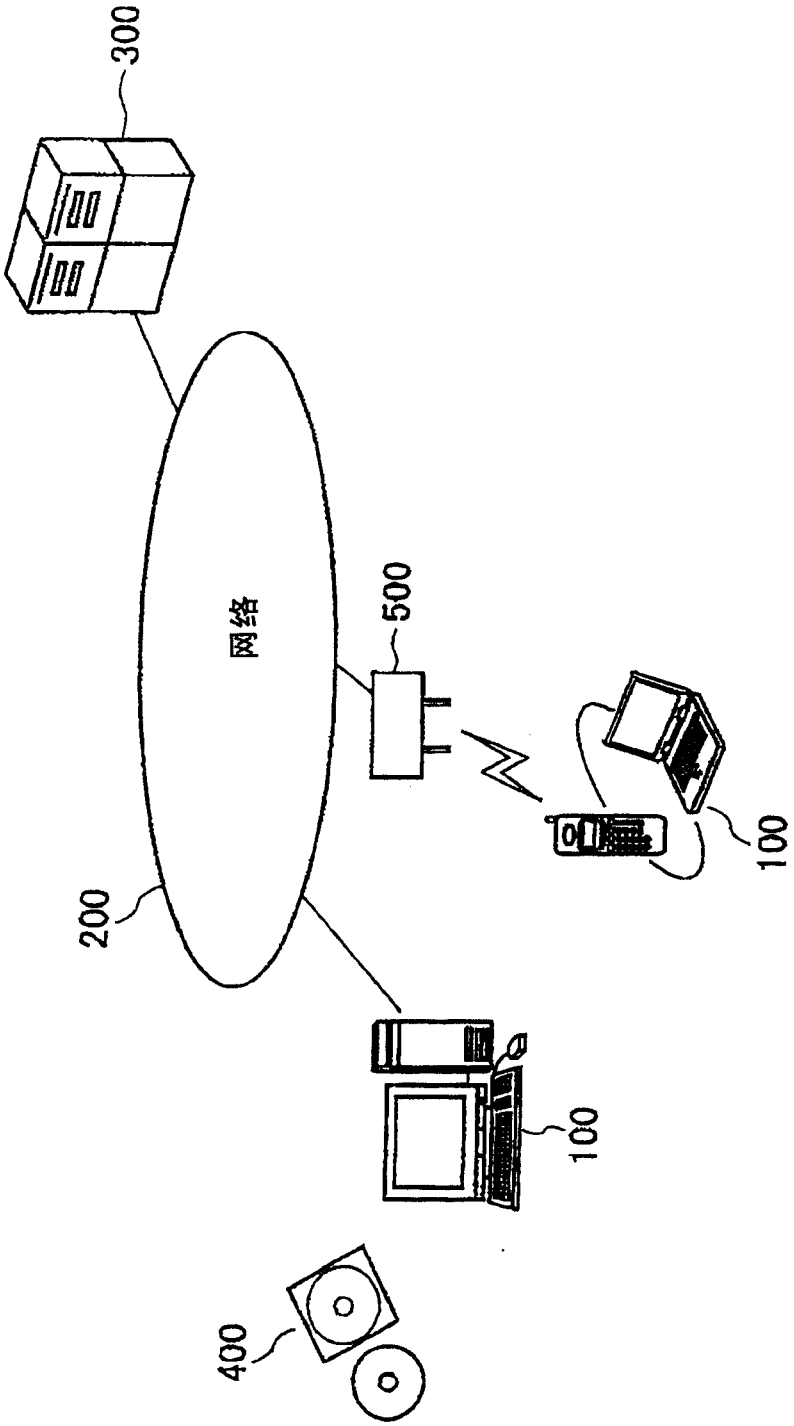


图 1

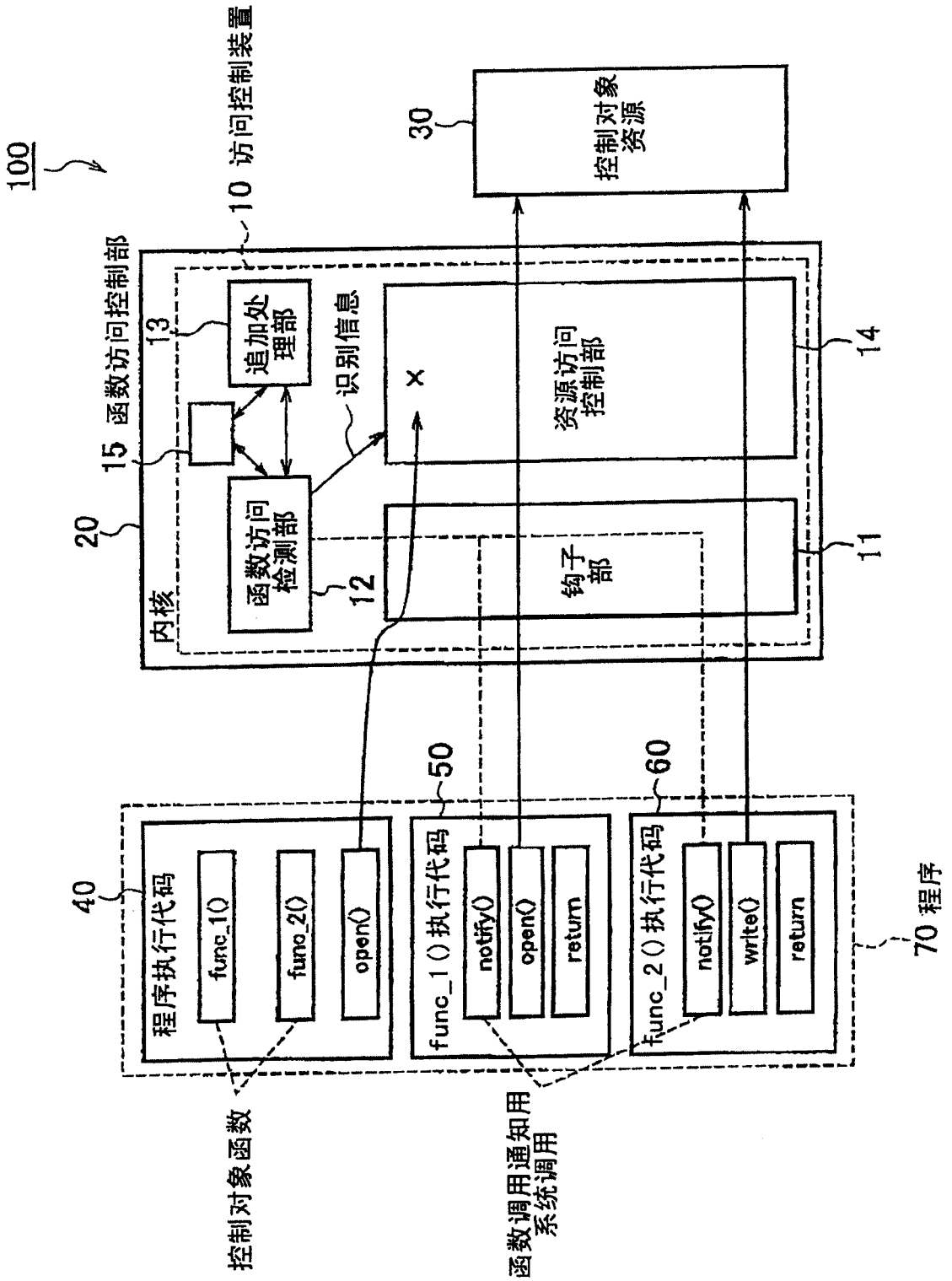

100
10 访问控制装置
30
控制对象资源
15 函数访问控制部
13
追加处理部
识别信息
资源访问控制部
14
20
函数访问检测部
12
钩子部
11
内核
40
50
60
程序执行代码
func_1()
func_2()
open()
func_1()执行代码
notify()
open()
return
func_2()执行代码
notify()
write()
return
70 程序
控制对象函数
函数调用通知用系统调用

图 2

资源	访问请求源	识别信息	许可动作
type_a	domain_1 (pre-install)		ioctl read write create • • •
type_b	domain_2 (download)	无	
		有	ioctl read write create • • •

图 3

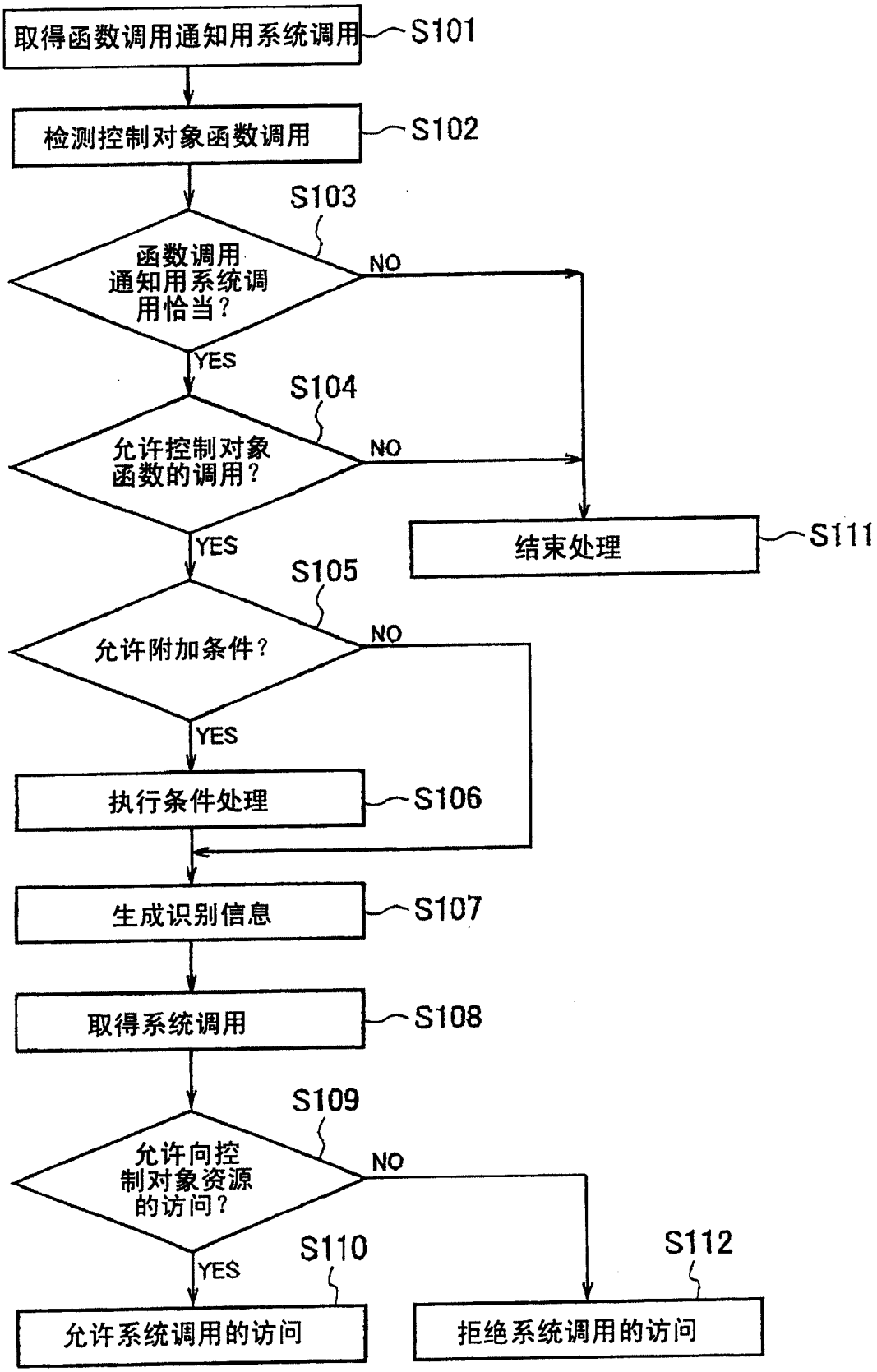

取得函数调用通知用系统调用
S101
检测控制对象函数调用
S102
S103
函数调用通知用系统调用恰当？
NO
YES
S104
允许控制对象函数的调用？
NO
YES
结束处理
S111
S105
允许附加条件？
NO
YES
执行条件处理
S106
生成识别信息
S107
取得系统调用
S108
S109
允许向控制对象资源的访问？
NO
YES
S110
允许系统调用的访问
S112
拒绝系统调用的访问

图 4

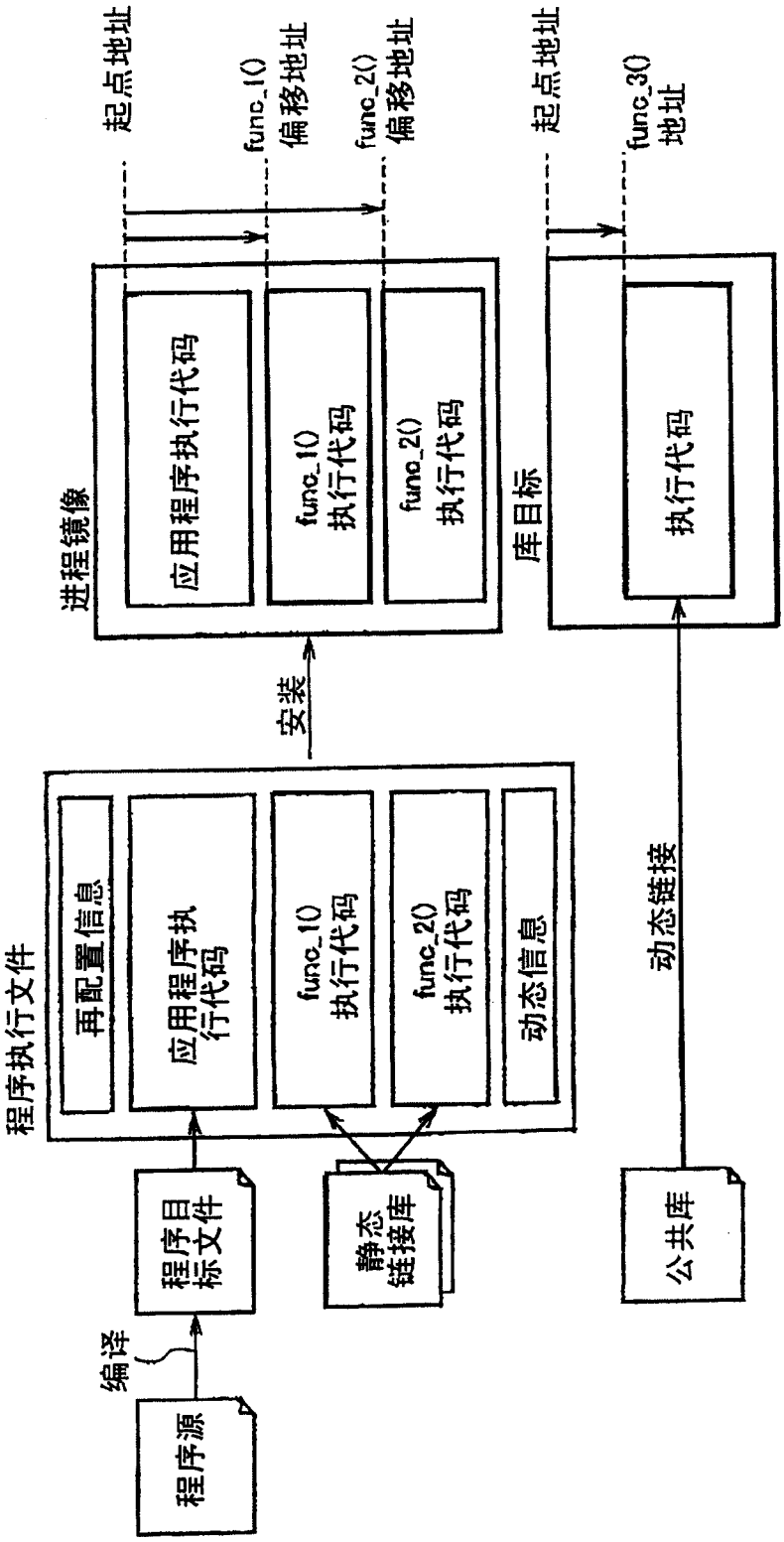

程序源
编译
程序目标文件
静态链接库
公共库
程序执行文件
再配置信息
应用程序执行代码
func_1()执行代码
func_2()执行代码
动态信息
安装
动态链接
进程镜像
应用程序执行代码
func_1()执行代码
func_2()执行代码
起点地址
func_1()偏移地址
func_2()偏移地址
库目标
执行代码
起点地址
func_3()地址

图 5

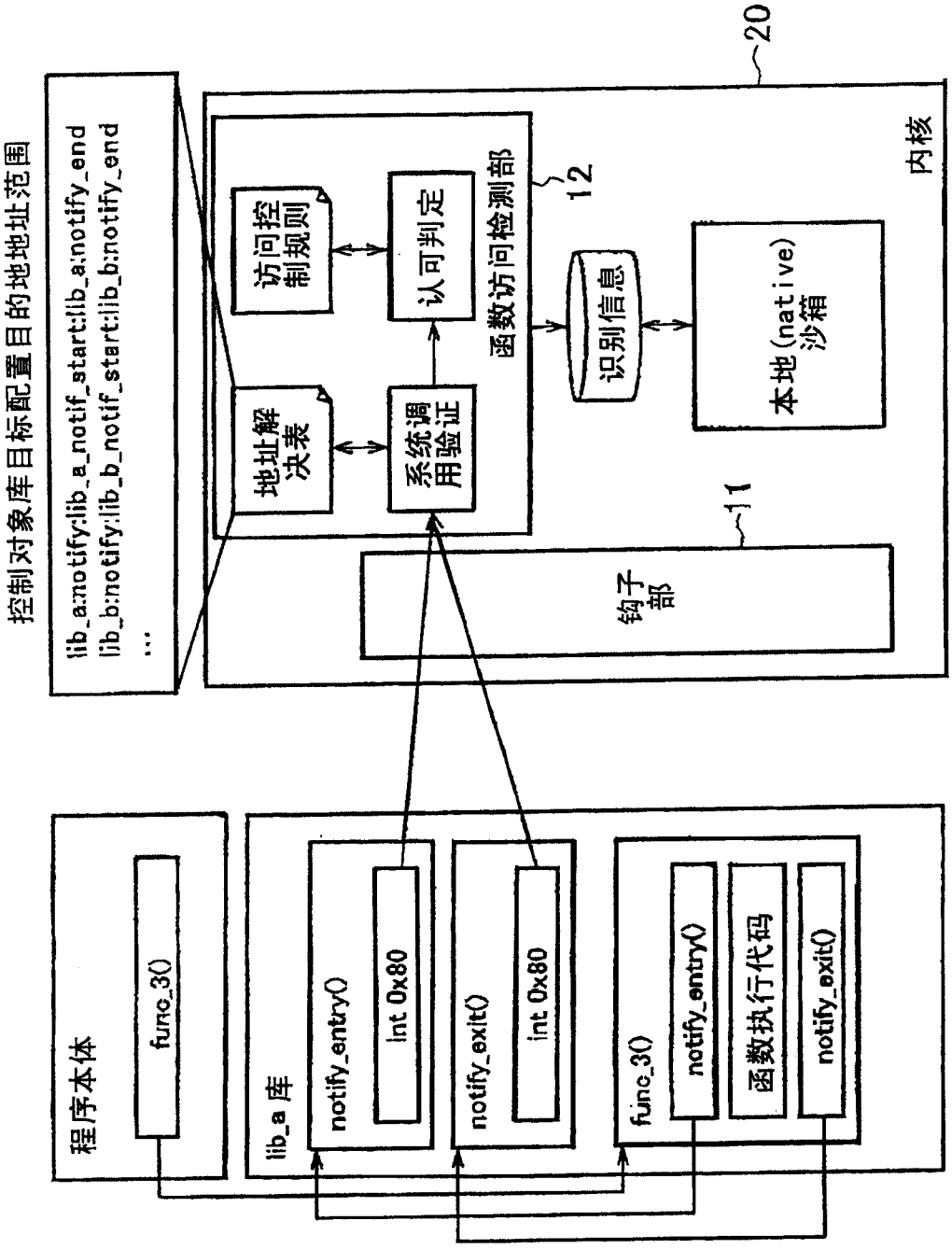

控制对象库目标配置目的地地址范围
lib_a:notify:lib_a_notif_start:lib_a:notify_end
lib_b:notify:lib_b_notif_start:lib_b:notify_end
...
20
内核
函数访问检测部
12
访问控制规则
认可判定
地址解决表
系统调用验证
识别信息
本地(native)沙箱
钩子部
11
程序本体
func_3()
lib_a 库
notify_entry()
int 0x80
notify_exit()
int 0x80
func_3()
notify_entry()
函数执行代码
notify_exit()

图 6

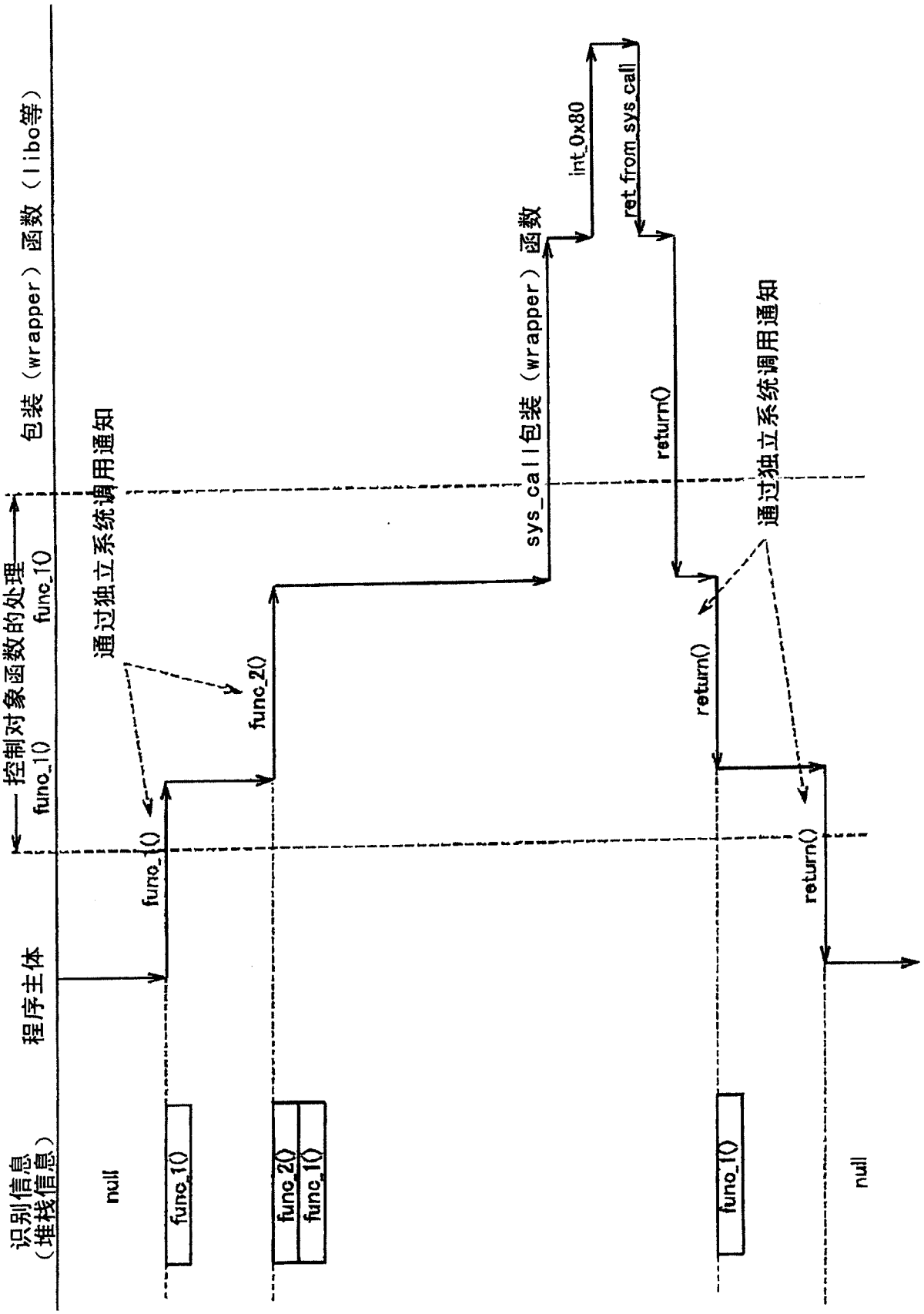

识别信息
（堆栈信息）
程序主体
控制对象函数的处理
func_10
func_1()
func_2()
func_1()
包装（wrapper）函数（libc等）
通过独立系统调用通知
sys_call包装（wrapper）函数
int_0x80
ret from sys_call
return()
return()
return()
通过独立系统调用通知
null
func_1()
func_2()
func_1()
func_1()
null

图 7