

## (12) 发明专利申请

(10) 申请公布号 CN 102939587 A

(43) 申请公布日 2013. 02. 20

(21) 申请号 201080065909. 5

(51) Int. Cl.

(22) 申请日 2010. 03. 31

G06F 9/445 (2006. 01)

(85) PCT申请进入国家阶段日

G06F 21/14 (2013. 01)

2012. 09. 28

G06F 21/51 (2013. 01)

### (86) PCT申请的申请数据

PCT/CA2010/000450 2010. 03. 31

### (87) PCT申请的公布数据

W02011/120122 EN 2011. 10. 06

(71) 申请人 埃德图加拿大公司

地址 加拿大安大略省

(72) 发明人 格兰特·斯图尔特·古德斯

克利福德·立厄姆

(74) 专利代理机构 北京英赛嘉华知识产权代理

有限责任公司 11204

代理人 余朦 杨莘

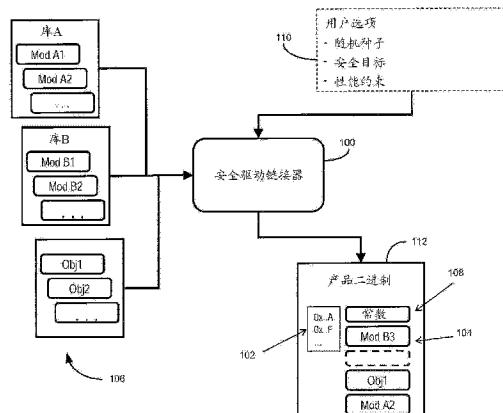
权利要求书 3 页 说明书 8 页 附图 6 页

### (54) 发明名称

用以保护应用程序的链接和加载的方法

### (57) 摘要

本发明描述了链接器或加载器、以及相关方法，依靠该链接器或加载器、以及相关方法，对目标代码模块应用安全变换可被推迟到链接或加载时，例如，通过存储重定位、从模块的多样实例的选择、以及常数的晚绑定。这提供了超越传统源到源安全变换的若干好处。这些推迟的安全变换可以非常轻质的方式应用并且为所产生的可执行程序的多样性创造许多机会，增强了安全性，同时使对执行性能和正确性的影响最小化，并且降低了调试的复杂性。



1. 在构建时或加载时将安全变换应用于目标代码的方法,包括 :

指定随机种子作为伪随机数生成器的输入 ;

通过以下步骤生成可执行模块的不同实例 :

通过应用所述随机种子作为所述伪随机数生成器的输入从多个已编译的软件模块中每一个的不同实例组中对所述多个已编译的软件模块中每一个的实例作出随机选择并确定所述已编译的软件模块的所选实例的随机布局来对所述多个已编译的软件模块应用多样化操作 ;

根据所述随机布局组合所述已编译的软件模块的所选实例以生成所述可执行模块的不同实例 ;以及

解析所述可执行模块的不同实例内的符号。

2. 如权利要求 1 所述的方法,其中作出随机选择的步骤包括从包含多个实例化模块的多样化提供的库中作出随机选择。

3. 如权利要求 2 所述的方法,其中所述多样化提供的库中的每个实例化模块均被实例化有不同的安全特性,并且作出随机选择的步骤还包括 :

嵌入所述多个已编译的软件模块的资产级别以及所述多个已编译的软件模块内的调用位置的剖析加权 ;

基于所述资产级别和所述剖析加权限制所述随机选择。

4. 如权利要求 1 所述的方法,其中确定随机布局的步骤包括将所述已编译的软件模块的所选实例拆分成分离的代码部分,以及使所述代码部分在存储映像内的定位随机化。

5. 如权利要求 4 所述的方法,其中使所述代码部分的定位随机化的步骤包括使来自所述已编译的软件模块的所选实例中的不同实例的代码部分混合。

6. 如权利要求 4 所述的方法,其中使所述代码部分的定位随机化的步骤包括使来自所述已编译的软件模块的所选实例中的不同实例的代码部分与数据部分混合。

7. 如权利要求 1 所述的方法,其中应用多样化操作的步骤还包括通过以下步骤实现的安全变换常数的晚绑定 :

应用所述随机种子作为所述伪随机数生成器的输入以根据与应用于所述已编译的软件模块的所选实例的安全变换有关的元数据随机地确定所述安全变换常数。

8. 如权利要求 7 所述的方法,其中所述安全变换是程序变换。

9. 如权利要求 8 所述的方法,其中所述程序变换是数据流变换或控制流变换。

10. 生成可执行程序映像的多个多样实例的方法,包括 :

对于所述可执行程序映像的每个所需实例 :

指定随机种子作为伪随机数生成器的输入 ;

通过以下步骤生成所述可执行程序映像的不同实例 :

通过应用所述随机种子作为所述伪随机数生成器的输入从多个已编译的软件模块中每一个的不同实例组中对所述多个已编译的软件模块中每一个的实例作出随机选择并确定所述已编译的软件模块的所选实例的随机布局来对所述多个已编译的软件模块应用多样化操作 ;

根据所述随机布局组合所述已编译的软件模块的所选实例以生成所述可执行程序映像的不同实例 ;以及

解析所述可执行程序映像的不同实例内的符号。

11. 如权利要求 10 所述的方法,其中作出随机选择的步骤包括从包含多个实例化模块的多样化提供的库中作出随机选择。

12. 如权利要求 11 所述的方法,其中所述多样化提供的库中的每个实例化模块均被实例化有不同的安全特性,并且作出随机选择的步骤还包括:

嵌入所述多个已编译的软件模块的资产级别以及所述多个已编译的软件模块内的调用位置的剖析加权;

基于所述资产级别和所述剖析加权限制所述随机选择。

13. 如权利要求 10 所述的方法,其中确定随机布局的步骤包括将所述已编译的软件模块的所选实例拆分成分离的代码部分,并使所述代码部分在存储映像内的定位随机化。

14. 如权利要求 13 所述的方法,其中使所述代码部分的定位随机化的步骤包括使来自所述已编译的软件模块的所选实例中的不同实例的代码部分混合。

15. 如权利要求 13 所述的方法,其中使所述代码部分的定位随机化的步骤包括使来自所述已编译的软件模块的所选实例中的不同实例的代码部分与数据部分混合。

16. 如权利要求 10 所述的方法,其中应用多样化操作的步骤还包括通过以下步骤实现的安全变换常数的晚绑定:

应用所述随机种子作为所述伪随机数生成器的输入以根据与应用于所述已编译的软件模块的所选实例的安全变换有关的元数据随机地确定所述安全变换常数。

17. 如权利要求 16 所述的方法,其中所述安全变换是程序变换。

18. 如权利要求 17 所述的方法,其中所述程序变换是数据流变换或控制流变换。

19. 生成可执行程序映像的多个多样实例方法,包括:

对于所述可执行程序映像或动态库的每个所需实例:

指定随机种子作为伪随机数生成器的输入以生成随机选择;

通过对多个已编译的软件模块应用由所述随机选择确定的多样化操作并且通过解析与所述多个已编译的软件模块关联的符号来对所述多个已编译的软件模块进行变换,

生成具有由所解析的符号确定的不同配置的可执行程序映像的实例。

20. 如权利要求 19 所述的方法,其中对所述多个已编译的软件模块进行变换的步骤包括解析所述符号以提供所述多个软件模块的随机化的存储映像布局。

21. 如权利要求 19 所述的方法,其中对所述多个已编译的软件模块进行变换的步骤包括解析所述符号以提供所述多个软件模块的不同实例的随机化选择。

22. 用于提供在构建时封装的可执行模块的多个多样不同实例的链接器,包括:

伪随机数生成器,被配置为使处理器在运行时为每个可执行模块的多个多样不同实例生成不同随机选择;以及

运行时模块,用于使所述处理器通过如下步骤对多个已编译的软件模块进行变换:

通过对多个已编译的软件模块应用由所述随机选择确定的多样化操作来对所述多个已编译的软件模块进行变换;

解析与所述多个已编译的软件模块关联的符号;以及,

每当调用所述程序时生成具有由所解析的符号确定的不同配置的可执行程序映像的实例。

23. 用于在运行时提供可执行程序映像的多样不同实例的加载器,包括 :

伪随机数生成器,被配置为使处理器为所述可执行程序映像的每个多样不同实例生成不同随机选择;以及

链接模块,用于使所述处理器通过如下步骤对多个已编译的软件模块进行变换:

通过对多个已编译的软件模块应用由所述随机选择确定的多样化操作来对所述多个已编译的软件模块进行变换;

解析与所述多个已编译的软件模块关联的符号;以及,

生成具有由所解析的符号确定的不同配置的可执行程序映像的实例。

24. 用于生成可执行程序映像的多个多样实例的加载器,包括 :

伪随机数生成器,被配置为使处理器为所述可执行程序映像的每个多样实例生成不同随机选择;以及

运行时模块,用于使所述处理器通过如下步骤生成所述可执行程序映像的不同实例:

通过应用所述随机选择从所述多个已编译的软件模块中每一个的不同实例组中对所述多个已编译的软件模块中每一个的实例作出选择并确定所述已编译的软件模块的所选实例的随机布局来对所述多个已编译的软件模块应用多样化操作;

根据所述随机布局组合所述已编译的软件模块的所选实例以生成所述可执行程序映像的不同实例;以及

解析所述可执行程序映像的不同实例内的符号。

25. 如权利要求 24 所述的加载器,其中所述随机布局包括在运行时加载期间的随机地址分配。

26. 如权利要求 24 所述的加载器,其中所述多个已编译的软件模块中每一个的实例的选择还基于运行时条件。

27. 如权利要求 26 所述的加载器,其中所述运行时条件包括检测未授权入侵。

## 用以保护应用程序的链接和加载的方法

### 技术领域

[0001] 本公开涉及保护软件不被篡改的方法和系统。具体地，本公开涉及在链接和加载时保护应用软件的方法和系统。

### 背景技术

[0002] 传统上，用于给定软件模块的安全变换直接应用于源代码。例如，程序变换已被证明是抗对软件的逆向工程和篡改攻击的有效途径。此外，这些安全变换的多样性对不同攻击、共谋、和其它对比威胁来说是进一步的屏障。例如，2003年7月15日发布的第6,594,761号美国专利和2005年1月11日发布的第6,842,862号美国专利描述了可用于保护软件的数据流变换技术；并且2004年8月17日发布的第6,779,114号美国专利描述了可用于保护软件的控制流变换技术。

[0003] 一旦已经应用了安全变换，源文件首先由实现软件模块的源到源安全变换的预编译器处理，基于用户确定的安全决策生成包含变换的变换源文件。变换源文件然后由本地编译器处理，生成目标代码。所产生的本地目标代码（可能被打包为静态库）直到涉及终端用户为止都是不可变的，并且必须通过链接器处理以产生可执行的二进制或动态库。库由链接器集成至程序。链接器的输出（特别地，存储器中的软件模块的布局）被自动生成，并且是输入软件模块的确定性函数，且针对运行时性能被优化。如果想要修改存储布局，则大部分链接器都需要使用复杂的元数据，这通常超出了所有用户甚至是最高级用户的范围。

[0004] 如果需要重访软件模块的安全决策，或哪怕需要产生模块的多样实例，只能重复预编译、编译、以及链接 / 加载的步骤。不论是从所涉及的构建时还是从质量保证观点来看，都是很昂贵的。重建软件模块要求必须重新验证所产生的可执行 / 动态库的性能（速度、大小等）和正确性。

[0005] 因此，希望提供改进的应用安全变换的方法和系统，该改进的方法和系统诸如当需要改变安全决策、或需要创建多样实例时不需要重建软件模块。

### 发明内容

[0006] 本公开提供了能够以多种方式修改二进制输出代码的内容的软件链接器或软件加载器，所有方式均通过多样性在所产生的可执行程序 / 动态库中提供了安全性。公开了三种具体实施方式。第一种涉及存储映像重定位。链接器的结果可以是随机化的地址分配和可重定位的代码的各段或部分的布局，这与由传统链接器作出的确定性的决策相反。对于基于可预见的部分定位的攻击来说，这一努力可能是重大的屏障。第二种实施方式涉及选择性构造。单个代码模块可以多个多样形式可用并在大容量库中单独提供。链接器将为最终的可执行程序随机选择代码实例。这一成就将阻挠不同攻击。第三种实施方式涉及常数的晚绑定。程序变换诸如数据和控制流安全变换，具有以所选常数为特征的操作参数。这些参数被类似于重定位地址进行处理，并且在链接时被选择，从而在执行时导致变换的多样化。这一成就还可用于抵抗不同攻击并且还能够通过编排的更新使可再生能力的成就

成为可能。

[0007] 链接器或加载器接受选项以控制其动作范围、以及允许创建可预见地多样的输出映像的随机种子参数。因此,如果链接器通过精确相同输入软件模块以及相同的随机种子被调用两次,则输出映像将是相同的。为了生成不同的输入映像,必需选择不同的随机种子。

[0008] 在第一个方面,提供了一种在构建时或加载时将安全变换应用于目标代码的方法。该方法包括指定随机种子作为伪随机数生成器的输入。随后通过以下步骤生成可执行模块、或可执行程序映像的不同实例:通过应用随机种子作为伪随机数生成器的输入从多个已编译的软件模块中每一个的不同实例的组中对多个已编译的软件模块中每一个的实例作出随机选择并确定已编译的软件模块的所选实例的随机布局来对多个已编译的软件模块应用多样化操作。随后根据随机布局将已编译的软件模块的所选实例组合以生成可执行模块的不同实例。随后解析可执行模块的不同实例内的符号。

[0009] 在另一个方面中,提供了一种生成可执行程序映像的多个多样实例方法。对于可执行程序映像的每个所需实例,指定随机种子作为伪随机数生成器的输入。随后通过以下步骤生成可执行程序映像的不同实例:通过应用随机种子作为伪随机数生成器的输入从多个已编译的软件模块中每一个的不同实例的组中对多个已编译的软件模块中每一个的实例作出随机选择并确定已编译的软件模块的所选实例的随机布局来对多个已编译的软件模块应用多样化操作。随后根据随机布局将已编译的软件模块的所选实例组合以生成可执行程序映像的不同实例。随后解析可执行程序映像的不同实例内的符号。

[0010] 随机选择可涉及从包含多个实例化模块的多样化提供的库中作出随机选择。多样化提供的库中的每个实例化模块均可被实例化有不同的安全特性,并且作出随机选择的步骤还包括:嵌入多个已编译的软件模块的资产级别以及多个已编译的软件模块内的调用位置的剖析加权;以及基于资产级别和所述剖析加权限制随机选择。

[0011] 确定随机布局的步骤可包括将已编译的软件模块的所选实例拆分成分离的代码部分,并使代码部分在存储映像内的定位随机化。使代码部分的定位随机化的步骤可包括使来自自己编译的软件模块的所选实例中的不同实例的代码部分混合、或者使来自自己编译的软件模块的所选实例中的不同实例的代码部分与数据部分混合。

[0012] 应用多样化操作的步骤还可包括安全变换常数的晚绑定,安全变换常数的晚绑定通过如下步骤实现:应用随机种子作为伪随机数生成器的输入以根据与应用于已编译的软件模块的所选实例的安全变换有关的元数据随机地确定安全变换常数。安全变换可以选自程序变换,尤其是数据流变换或控制流变换。

[0013] 在另一个方面中,提供了一种生成可执行程序映像的多个多样实例的方法。对于可执行程序映像或动态库的每个所需实例,指定随机种子作为伪随机数生成器的输入以生成随机选择。通过对多个已编译的软件模块应用由随机选择确定的多样化操作并且通过解析与多个已编译的软件模块关联的符号对多个已编译的软件模块进行变换。随后生成具有由所解析的符号确定的不同配置的可执行程序映像的实例。对多个已编译的软件模块进行变换的步骤可包括解析符号以提供多个软件模块的随机化的存储映像布局,或解析符号以提供多个软件模块的不同实例的随机化选择。

[0014] 在又一个方面,提供了一种用于提供在构建时封装的可执行模块的多个多样不同

实例的链接器或加载器。该链接器或加载器包括：伪随机数生成器，被配置为使处理器在运行时为每个可执行模块的每个多样不同实例生成不同随机选择；以及运行时模块，用于使处理器变换多个已编译的软件模块。变换通过如下步骤实现：通过对多个已编译的软件模块应用由随机选择确定的多样化操作来对多个已编译的软件模块进行变换；解析与多个已编译的软件模块关联的符号；以及每当调用程序时生成具有由所解析的符号确定的不同配置的可执行程序映像的实例。

[0015] 在另一个方面，提供了一种用于生成可执行程序映像的多个多样实例的加载器。加载器包括：伪随机数生成器，被配置为使处理器为可执行程序映像的每个所需实例生成不同随机选择；以及运行时模块，用于使所述处理器生成所述可执行程序映像的不同实例。不同实例通过如下步骤生成：通过应用随机选择从多个已编译的软件模块中每一个的不同实例组中对多个已编译的软件模块中每一个的实例作出选择并确定已编译的软件模块的所选实例的随机布局来对多个已编译的软件模块应用多样化操作；根据随机布局组合已编译的软件模块的所选实例以生成可执行程序映像的不同实例；以及解析可执行程序映像的不同实例内的符号。随机布局可包括例如在运行时加载期间的随机地址分配。多个已编译软件模块中每一个的实例的选择还可基于运行时条件，诸如检测未授权入侵。

[0016] 对本领域普通技术人员来说，在结合附图阅读了本发明的具体实施方式的下面的描述之后，本发明的其它方面和特征将变得显而易见。

## 附图说明

- [0017] 下面将通过仅示例参照附图描述本公开的实施方式。
- [0018] 图 1 示出具有多样性保护的产品二进制的构建时链接；
- [0019] 图 2 是为安全 / 性能目标优化混合匹配选择的流程图；
- [0020] 图 3 示出变换常数的晚绑定；
- [0021] 图 4 示出在运行时加载期间选择不同模块；
- [0022] 图 5 示出在运行时加载期间的随机地址分配；
- [0023] 图 6 示出具有可替换的、不同模块的动态库的构造。

## 具体实施方式

[0024] 本公开描述了一种方法，依靠该方法，软件模块的安全变换可被推迟到链接或加载时，从而与传统的源到源安全变换相比提供了若干优点。这些推迟的安全变换可以非常轻质的方式应用并且为所产生的可执行程序的多样性创造了许多机会，增强了安全性，同时使对执行性能和正确性的影响最小化并降低了调试的复杂性。

[0025] 软件链接器和链接器技术已经存在了许多年，并且提供将一些预先编译的软件模块组合成单个可执行二进制或动态库的能力。链接器的主要工作是将编译的软件模块组合成单个存储映像(memory image)，从而决定各模块在存储器中应驻留的位置，并“安置”或重定位任何模块间的地址引用(诸如子程序调用)，使得它们在给定所选的存储布局的情况下是正确的(见例如 J. Levine, Linkers and Loaders(链接器和加载器), Morgan Kaufmann Publishers, 2000)。多年以来，用于链接器技术的主要创新已经被限制为改善性能、提供针对不同处理器和操作系统的能力、以及对高级编程语言的支持。

[0026] 传统的软件链接器主要将本地目标代码看作是不可变的,从而将修改限制为将输入软件模块(以目标代码的形式)组合成单个可执行二进制或动态库所必需的绝对最小。链接器或链接编辑器是获得由编译器生成的一个或多个目标并将它们组合成单个可执行程序的程序。计算机程序通常包括若干部分或模块;所有这些部分 / 模块不需要包含在单个目标文件内,并且在这种情况下依靠符号彼此引用。通常,目标文件可包含三类符号:定义符号,允许目标文件被其它模块调用;未定义符号,调用定义有这些符号的其它模块;以及局部符号,在目标文件内内部地使用以有助于重定位。当程序包括多个目标文件时,链接器将这些文件组合成统一的可执行程序,从而在该程序执行时解析这些符号。

[0027] 链接器可从一些目标或子程序(术语叫库)获取目标。某些链接器在输出中不包括整个库;它们仅包括它们的从其它目标文件或库引用的符号。链接器还负责将目标布置在程序的地址空间中。这可涉及将特定基地址假定为另一个基的重定位代码。由于编译器几乎不知道目标将驻留的位置,故其常常假定固定的基位置(例如,0)。重定位机器代码可例如涉及将绝对跳转、加载和储存重新作为目标。编译器的可执行输出在其刚好在执行前最终被加载至存储器内时可能需要另一个重定位途径。

[0028] 动态链接涉及在运行时将库的子程序加载至应用程序,而不是在编译时链接库的子程序;这些子程序依然作为盘上的单独文件存在。仅最小量的工作由链接器在编译时完成;其仅记录该程序需要的库程序和库中的程序的索引名称或数量。链接的主要工作在应用程序被加载时(加载时)或在执行时(运行时)完成。被称为加载器的必要的链接功能实际上是底层操作系统的一部分。在合适的时间,加载器找到盘上的相关库并将来自这些库的相关数据添加至处理的存储空间。

[0029] 因为链接器通常是构建过程的最后一步,故存在超越符号解析、布局确定、以及地址绑定的传统工作的机会。链接步骤可用作组合抗一组类攻击的多个多样实施的手段,并提供再访传统链接器技术的基础。在链接时引入多样性具有轻质的好处。不需要重新运行昂贵的编译步骤,使得正确地生成大量多样实例变得可行。每个实例具有可预测的性能。由于代码基本是固定的(已经编译),这类多样性不可能在执行性能中引入大变化。这还将导致减少的测试工作,由于一个多样实例的代码代表所有其他多样实例的行为:一个多样实例的全面测试可被假定为代表所有其他多样实例的行为。

[0030] 本公开提供软件链接器或加载器、以及相关的链接和加载方法,其可以多种方式生成二进制输出代码的内容,所有方式都通过所产生的可执行 / 动态库中的多样性提供了安全性。描述了可应用于目标代码的三种特殊的多样化操作。这些多样化操作可根据设计需要以任何合适的方式单独地或组合地应用。扩展的链接器工具接受选项以控制其动作的范围,并且接受种子作为伪随机数生成器(PRNG)的输入,其允许创建可预见的多样输出映像。因此,如果链接器通过精确相同的输入软件模块以及完全相同的种子调用两次,则输出映像将是相同的。为了生成多样的输出映像,选择不同的种子。

[0031] 图 1 示出安全驱动链接器 100,安全驱动链接器 100 被启用以执行三种多样化操作(下面详细讨论):存储重定位 102、基于多样化实例化的库和目标模块 106 的选择性构造 104、以及变换常数 108 的晚绑定,所有这些都是根据输入 110(诸如 PRNG 的种子、以及任选的安全目标和性能约束)确定的。多样化操作的结果是抗篡改的产品二进制、或输出映像的安全增强实例 112、或通过改变输入 110 确定的输出映像的多个不同实例 112。根据上下

文,输出映像 112 被描述为可执行程序映像、存储映像、可执行模块、或动态库。

[0032] 存储映像重定位

[0033] 传统链接器将基于输入软件模块在命令行上的排序以及输入软件模块的相关性(一个模块引用另一个模块)在输出存储映像中可预见地定位其输入软件模块。此外,传统链接器通常将确保相似类型的存储段集合在一起,例如,将所有常量字符串数据放在一个存储段中。这么做是为了方便(简化重定位算法)、易于调试、并且由于缓存位置而导致所产生的可执行映像的运行时性能最大化。

[0034] 参照图 1,首先公开的多样化操作涉及存储映像重定位 102。链接器的结果可以是随机化的基址分配,0x..A,0x..F 等以及可重定位代码的各段或部分的随机化布局,这与由传统链接器作出的确定性的决策相反。这一成就可以是基于可预见的部分定位的重要屏障。基于请求和输入 110(诸如用于 PRNG 的种子),链接器 100 可确保各输出映像 112 具有全新的存储映像布局(即,可重定位代码的各段或部分的位置由随机种子参数随机确定),并且还可在整个存储映像上掺和或混合不同类型(即数据和代码)的存储段。其通过向链接器 100 提供自由混合代码和数据的能力,扩展了链接器的仅可以确定性方式为软件模块中的单独功能的代码重定位的传统重定位功能。混合数据和代码用作对二进制映像反汇编器 / 分析器诸如 IDA Pro™ 的正确操作的妨碍。

[0035] 理想地,可执行输出映像 112 的正确功能行为对于存储布局的所有多样实例来说都应该是相同的。原始执行性能将受到缓存位置问题的影响,虽然通过存储布局算法的合适选择,性能变化可被保持为最小。因此,当使用存储映像重定位时,在由扩展的链接器创建的所有多样实例上,性能和正确性都将得到维持。

[0036] 选择性构造

[0037] 对于传统的链接器,各外部可见的实体(诸如函数或全局变量)必须在输入软件模块组中一次性准确地提供。否则,该实体就被认为是“多重定义”,并且将出现错误。在本链接器 100 中,第二多样化操作涉及“混合与匹配”链接,或基于各库 106 内的多样化实例化的目标模块的选择性构造 104。

[0038] 库 106 可用多个实施模块创建。例如,可创建程序模块的多个版本、或实例(Mod A1、Mod A2、Mod B1、Mod B2)。给定程序模块的各实例执行相同的功能行为。多个多样实例(Mod A1、Mod A2、Mod B1、Mod B2...;Obj1、Obj2、...)可在大容量库中单独提供。由于所有实例均提供相同的功能,故本链接器 100 可为最终的可执行程序随机选择特定的实例。这一成就可阻挠试图通过对比获得信息的不同攻击。链接器 100 可支持一种模式,在该模式中将再次基于种子所确定的随机选择选择给定外部可见实体(通常为函数)的一组多个多样实例之一。如果许多函数在许多多样实例中可用(各实例具有它们自己的安全变换组),则可能的、多样输出映像的数量是这些函数和实例的数量的乘积。这给出了极大的范围以创建大量的多样实例,除此之外,由于各实例可涉及单独函数的相当不同的实施,故将对一个实例进行逆向工程而获得的知识应用于任何其它实例的能力被最小化。

[0039] 多个实例不仅可具有多样的实施以阻挠不同的分析;还可被创建为具有不同的性能和 / 或安全特性(见输入 110)。例如,对于执行密码哈希的函数,链接器 100 可选择在非时间关键区域或处理中使用高度保护的、但较慢的实施。在时间关键区域或处理中,链接器可选择在安全方面与高性能实施之间进行折衷。考虑到安全和性能目的的正确输入,如通

过设计考虑确定、结合实施的正确特性和算法分析信息，链接器 100 有机会通过混合和匹配功能性实施来平衡约束。

[0040] 图 2 描绘了用于平衡安全与性能的优化方法，该优化方法按如下执行。在步骤 200 中，用户提供与应用中的高价值资产有关的信息。这一步骤包括增加关键资产相对于较不关键资产的级别(rating)（即，资产级别）。这些关键资产的示例可包括：系统中的密钥、节点锁定计算、以及权利管理逻辑。在步骤 202 中，基于剖析将加权分配给调用位置。动态或静态源级剖析是为函数和基础块级的应用程序而产生的。动态剖析可能是优选的，但在没有动态剖析的情况下，静态剖析可由源级分析器(例如，编译器或预编译器)通过传播对整个调用图上的循环频率的估计来产生。对于在应用程序中发现的每个函数调用位置，剖析加权可被关联。在步骤 204 中，通过合适的目标和库模块嵌入资产级别和剖析加权(profile weighting)。例如，所产生的包含在资产级别和剖析加权中的信息通过编译和库创建阶段被携带，使得它们可作为与库中的模块关联的额外信息被嵌入。在步骤 206 中，链接器 100 使用嵌入的信息来选择模块，使用基于约束的解决算法来为安全 / 性能目标进行优化。存在许多已知约束解决(constraint-solving)途径。

#### [0041] 变换常数的晚绑定

[0042] 传统软件链接器基于链接器所获得的最终的存储映像布局决策标识必须安置(或重定位)的输入本地目标代码的位置。这些通常是对在其他软件模块中发现的外部可见符号(诸如全局变量或函数)的引用。该重定位通常仅仅是实际上通过链接过程修改的本地目标代码的一部分。对代码和数据的安全变换常常涉及对特征常数的使用。例如，数据变换涉及对数学映射函数的使用，于 2003 年 7 月 15 日发布的第 6,594,761 号美国专利以及于 2005 年 1 月 11 日发布的第 6,842,862 号美国专利中描述了将数据操作和位置变换至可替换的数学空间的数学映射函数，这两个专利的全部内容通过引用并入本文。为此，映射工具(即“代码转换器”)为每次映射使用一组函数族和多个特征常数。类似地，通过使用多个函数来控制流，控制流变换(如于 2004 年 8 月 17 日发布的第 6,779,114 号美国专利中所述，该专利的全部内容通过引用并入本文)将给定程序的控制流映射为新的控制流形式。这些函数还携带有一组选定的常数。

[0043] 在这些情况的每一个中，常数被随机(且彼此之间一致地)选择。该组常数成为程序实例的区别因素并且是程序正确工作所必需的。这些常数的精确值是由映射工具(即，代码转换器)基于变换函数的特性选择的。此外，在整个应用中可能存在许多相互关联的安全变换常数，这些常数已经以相互一致的方式选择，使得改变一个常数可能需要以某些一致方式改变许多其它常数。

[0044] 作为可应用于目标代码的又一多样化操作，还可执行用于程序变换尤其是数据和控制流变换的常数的晚绑定。在这种情况下，该组常数由第二组常数(随机选择)替代，第二组常数必须一致地一同工作，然而却代表程序实例的第二区别因素。这些常数可被相似地处理以重定位地址，并且在链接时被选择，从而在执行时导致变换的多样化。这种修改进一步地防止不同攻击，并且还能够通过编排的更新使可再生能力的成就成为可能。

[0045] 链接器可通过与传统链接器的重定位能力类比来修改程序变换(诸如数据和控制流变换)的特征常数组。这些特征常数组由 PRNG(伪随机数生成器)生成，PRNG 由种子初始化。各组中的常数的可能的值非常大，导致所产生的可执行或动态库中的极大范围的多

样性。参照图 3,示出了用于在链接时选择常数的方法。链接器 300 与提供多个软件模块(模块 A1、模块 A2、...)的库 A 通信。各软件模块包括常数部分 302 和与安全变换的特性有关的元数据 304。链接器 300 首先选择模块(步骤 310),随后读取与所选模块关联的元数据 304(步骤 312)。基于该元数据,链接器 300 选择新的常数(步骤 314),并在产品二进制 322 中用新的一组常数 320 替换常数部分(步骤 316)。

[0046] 由于各组常数不影响应用程序的正确行为(假设该组常数满足元数据中给出的数学约束),故针对该不同实例的性能 / 正确性的验证是简单的。一个一致常数组被期望与第二个一致常数组表现相同。这种形式的链接时多样性因此代表用户的最小开销与抵抗不同攻击的最大安全之间的良好折衷。其还理想地适于在现场应用中提供再生能力,其中应用二进制通过新的特征常数组周期性地更新,这意味着其它攻击者没有机会用固定的二进制映像进行分析。

#### [0047] 运行时加载

[0048] 加载器还可被看作是在运行时被调用的链接器。描述构建时、或加载时的上述所有技术、链接特征也可考虑在运行时被调用。在运行时执行这些特征的优点是插入其他安全保护技术、以及对现场中的条件作出反应的能力。加载器在图 4 和 5 中示出。图 4 示出从模块的多样实例进行选择以加载至产品二进制,而图 5 示出在运行时加载期间的随机地址分配。

[0049] 在图 4 的实施方式中,软件模块的多个多样实例(Mod A1、Mod A2、Mod A3、Mod A4、Mod B1、Mod B2、Mod B3、Mod B4...)可在动态库诸如动态库 A 和动态库 B 中单独提供。这些实例可以单独的实施因素或标准为特征。例如,某些示例可被特征化为具有高安全级别,诸如模块 Mod A1 和 Mod A2 的“强大保护”,或它们可以它们的执行速度为特征,诸如“快速”模块 Mod A3、Mod A4。

[0050] 为了提供运行时的多样性,加载器 400 需要运行时熵源 402 和伪随机数生成器(PRNG),PRNG 控制多样模块选择器 404 作出伪随机决策,并确保运行时加载器的行为、或运行时加载器的输出可执行程序 406 在每次被调用时都是不同的。运行时加载器 400 可基于运行时条件从动态库中找到的各种实施进行选择。例如,如果入侵(breach)管理模块发现攻击正在发生(例如,进程正在被仿真和调试),则入侵管理模块可向加载器发送信号以加载具有当前最高安全级别和保护能力的实施(即,从模块 Mod A1 和 Mod A2、或 Mod B1 和 Mod B2 的随机化选择)。基于其它条件,入侵管理模块可作出其它决策。例如,如果应用程序被重复调用,并且从未到达其关键执行点,则加载器可选择将随机实施加载至比平常更细粒度。这将阻碍不同攻击并使依靠可重复性和可预见性的对手气馁。

[0051] 在图 5 所示的实施方式中,软件模块的多个多样实例(Mod A1、Mod A2、Mod A3、Mod A4、Mod B1、Mod B2、Mod B3、Mod B4...;以及 Mod C1、Mod C2、Mod C3、Mod C4)可在动态库诸如动态库 A、动态库 B 和动态库 C 中再次单独提供,并且运行时加载器 500 再次包括运行时熵源 502 和伪随机数生成器(PRNG),PRNG 控制多样模块选择器 504 作出伪随机决策。在该实施方式中,多样选择器模块 504 可被实施以如上面参照存储映像重定位和选择性构造描述一样工作。可选择模块的特定多样实例,并且每个所选模块当其在运行时被加载时可被分配有随机地址。可执行程序 506 可以其大部分功能都由动态库模块构成的方式组成。因为模块在执行的变化时间处被加载,故除了从一组多样的选项选择的模块之外,模

块还在存储器中的随机地址处被加载。这为运行调试或仿真会话的攻击者带来重大阻碍。当程序正在执行时,不仅存在不可预见的多样模块正被加载,而且该模块的位置也是不可预见且不可重复的。

[0052] 动态库的构造

[0053] 动态库(例如 Windows 的 .dll、Linux 上的 .so、Mac 上的 .dylib)通常由本地链接器创建并且能够与可直接执行的映像近距离对比。通常,动态实体中的符号中的绝大部分都在创建该库之前已经被解析。在该库被创建之前已经准备好执行。另一方面,动态库在其运行之前不需要被加载到执行的程序内。此时,加载器在库被执行之前有机会重定位某些符号。动态库中的所有未解析的符号刚好在执行之前被解析。因为该库由链接器创建,故该库不包含不明确定义的符号。除了少量的可重定位的符号之外,该库基本准备好执行。

[0054] 然而,多样性还可被引入产品动态库的构造,如图 6 所示。动态库创建机制提供多个可替换模块(即,函数和数据):AltMod A1、AltMod A2、AltMod B1、AltMod B2、……。各可替换模块给出与各自模块(Mod A1、Mod A2、Mod B1、Mod B2…;Obj1、Obj2、… )的实例的任选等同。库的创建需要链接器 600,链接器 600 将可替换模块封装在产品动态库 602 中。其还携带指向可替换模块的表。随后,在运行时,适当实施的加载器可通过查询该表在可替换的、不同模块之中进行随机选择,如上面详细说明。

[0055] 在先前的描述中,出于解释的目的,陈述了大量细节以提供对实施方式的完整理解。然而,本领域技术人员应理解,这些具体的细节是不需要的。在其他实例中,为了不使理解晦涩,以框图形式示出已知的电气结构和电路。例如,未提供文中所述的实施方式是否被实施为软件程序、硬件电路、固件、或其组合的具体细节。

[0056] 本公开的实施方式可被称为储存于机器可读媒介(又称为具有计算机可读程序代码的计算机可读媒介、处理器可读媒介、或计算机可用媒介)中的计算机程序产品。机器可读媒介可以是任何合适的有形非短暂媒介,有形非短暂媒介包括磁、光或电存储媒介,磁、光或电存储媒介包括磁盘、只读光盘存储器(CD-ROM)、存储器设备(易失或非易失的)、或类似存储机构。机器可读媒介可包含各种指令集、代码序列、配置信息或其它数据,其被执行时使处理器执行根据本公开的实施方式的方法中的步骤。本领域技术人员将理解,实施所述实施所需的其它命令和操作也可储存在机器可读媒介上。储存在机器可读媒介上的指令可由处理器或其它合适的处理设备执行,并且可与电路系统交互以完成所述工作。

[0057] 本发明的上述实施方式仅作为示例。在不背离本发明的范围的情况下,本领域技术人员可对具体实施方式进行改变、修改和变更,本发明的范围仅由所附权利要求限定。

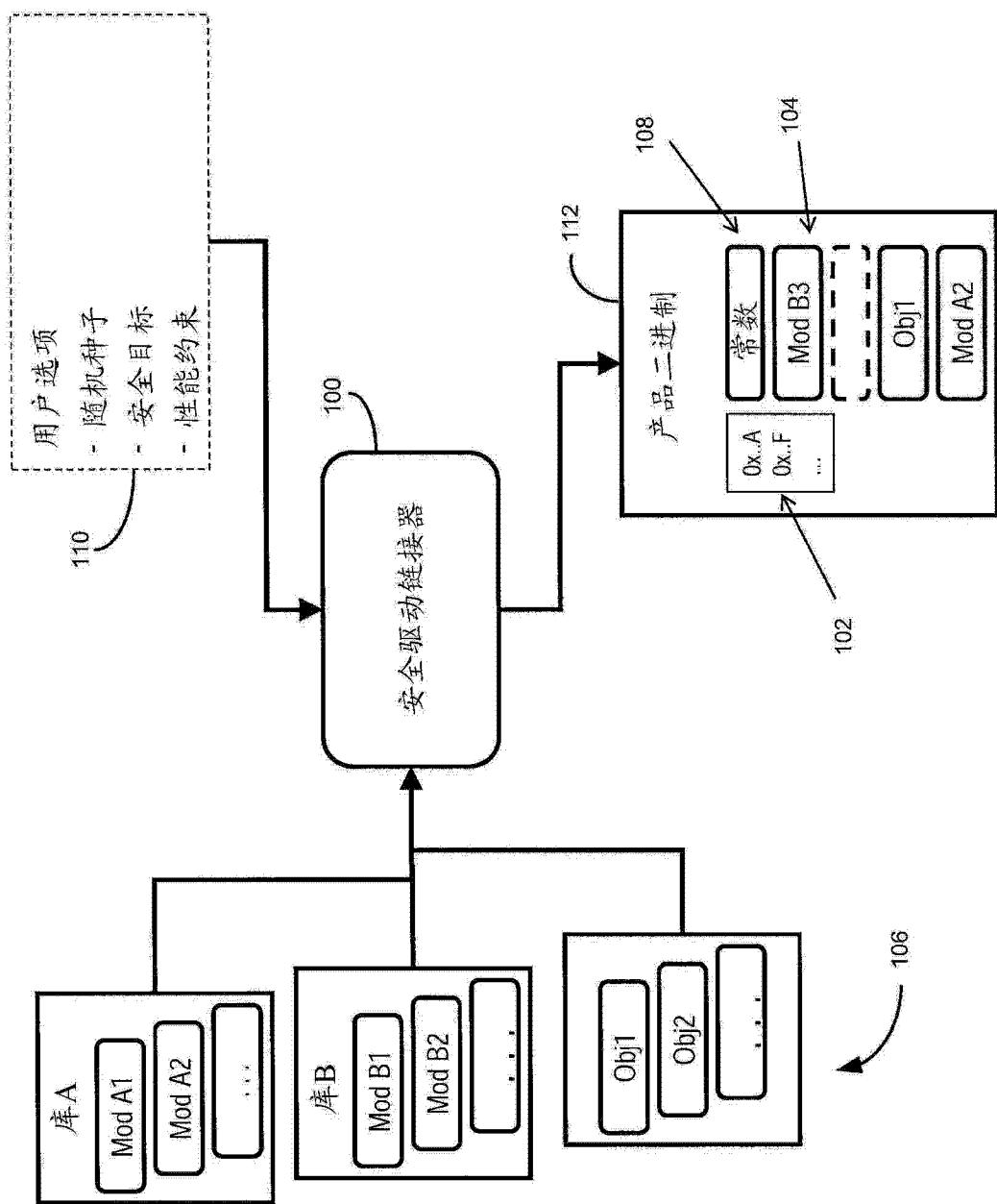


图 1

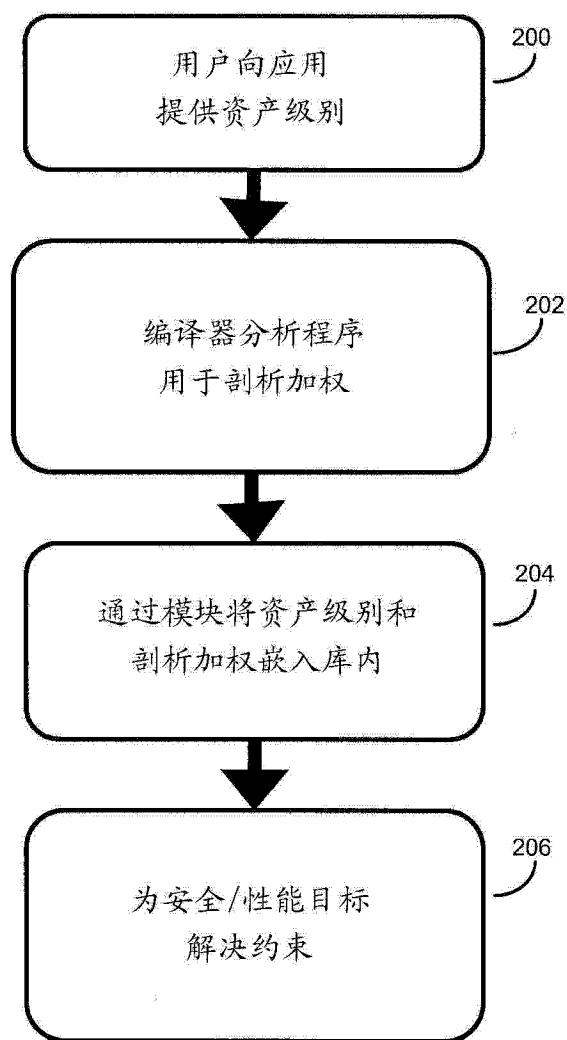


图 2

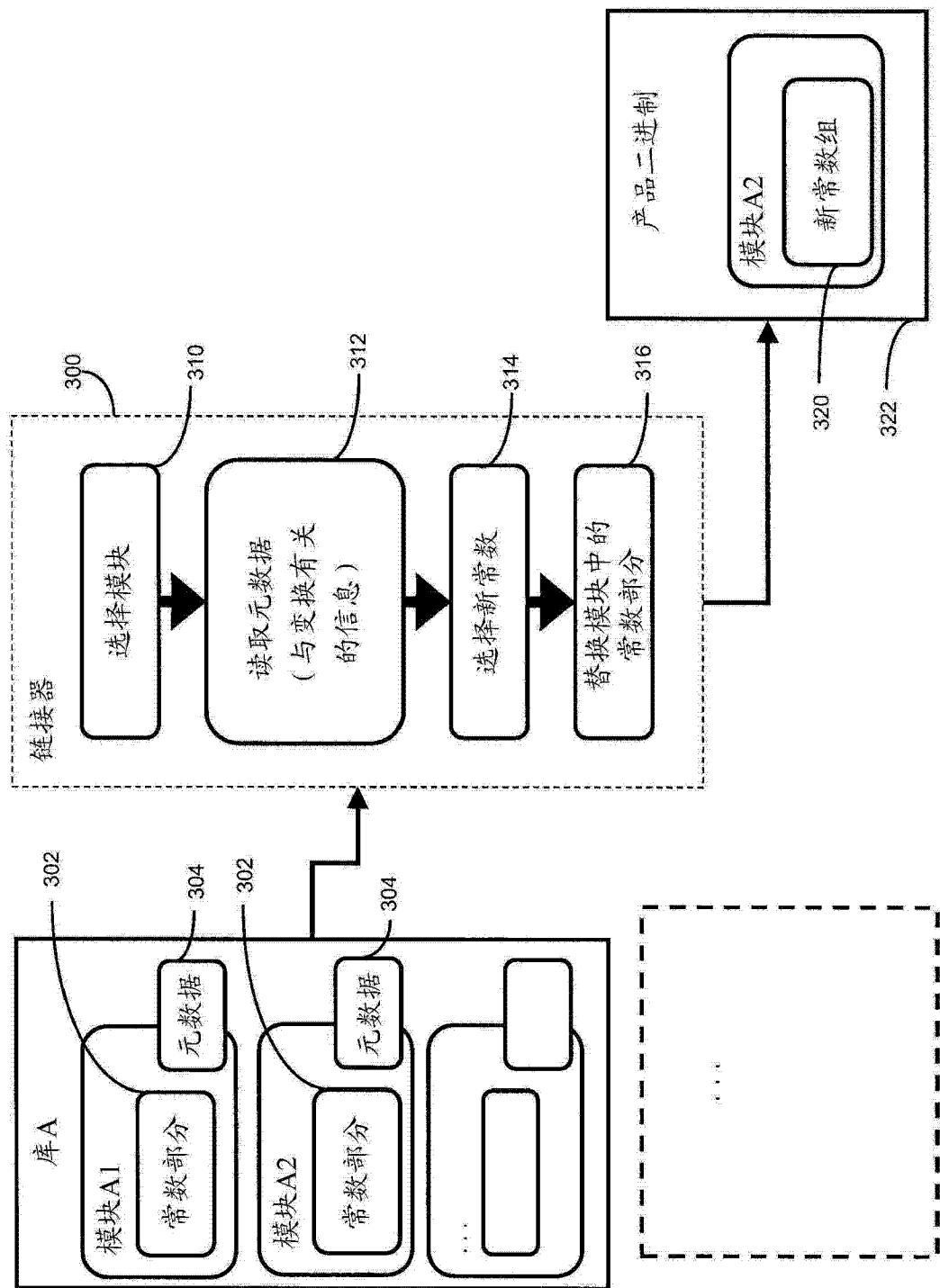


图 3

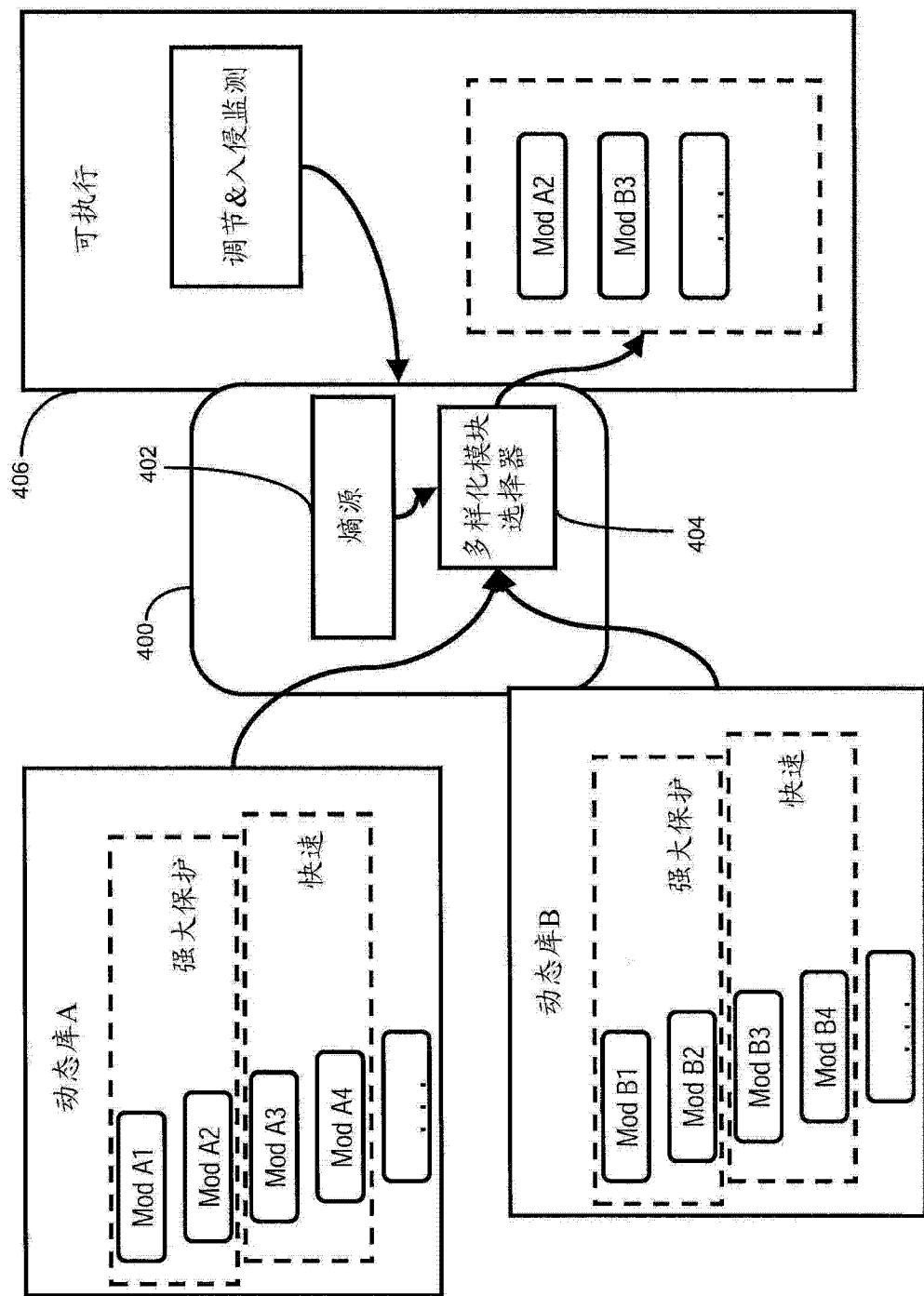


图 4

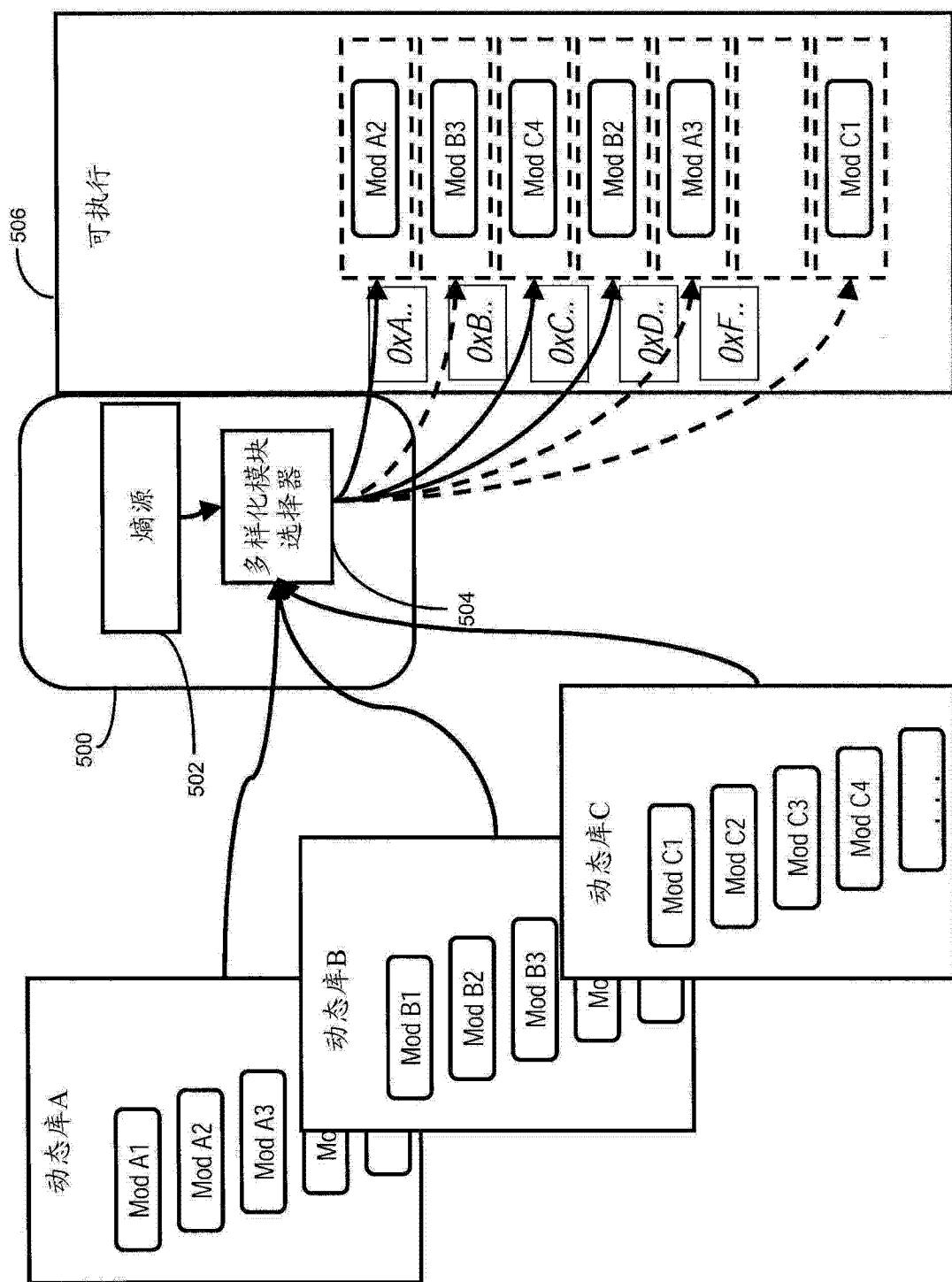


图 5

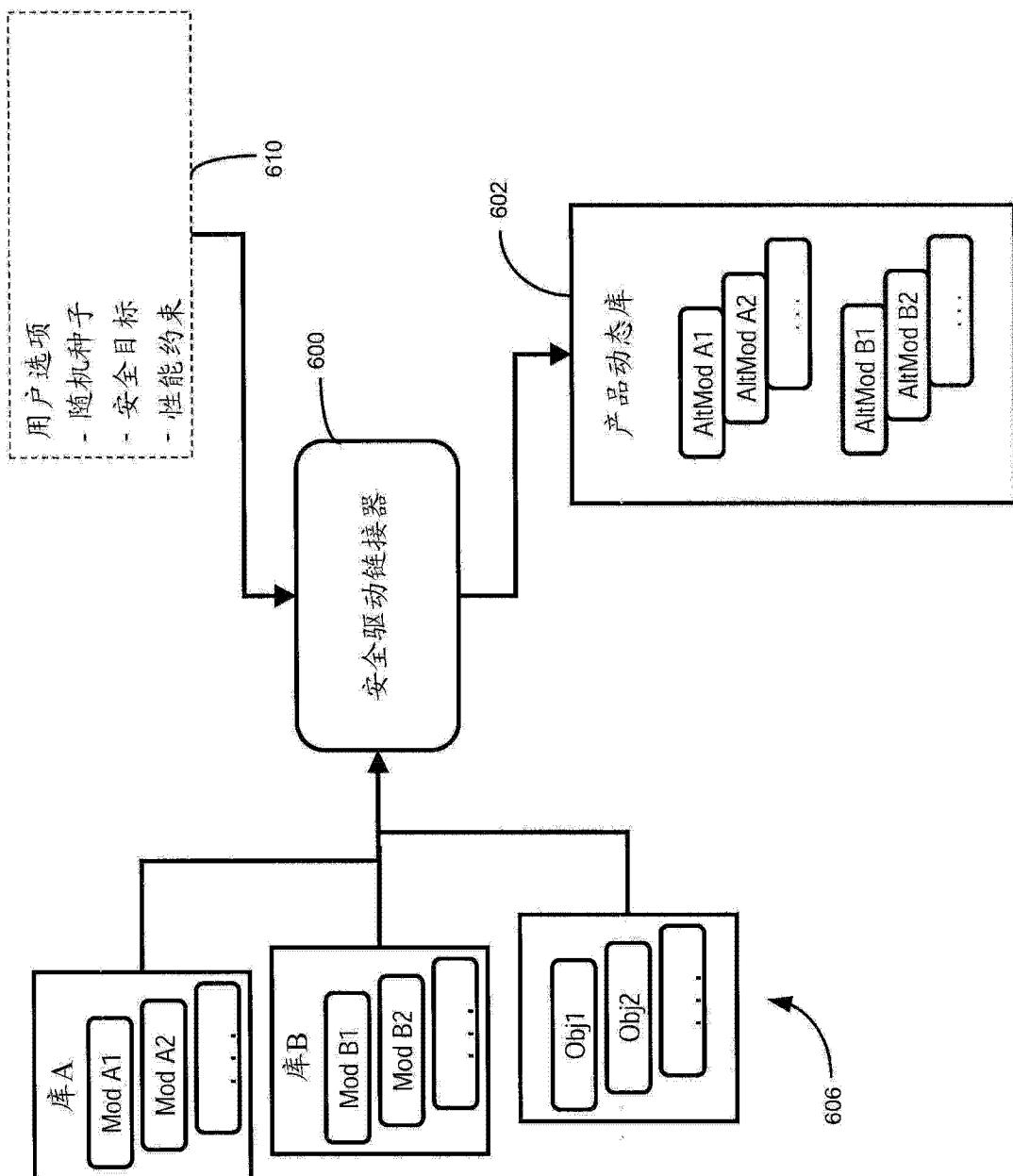


图 6