(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 103562923 B
(45)授权公告日 2016.09.07

(21)申请号 201180071281.4
(22)申请日 2011.05.31
(65)同一申请的已公布的文献号
申请公布号 CN 103562923 A
(43)申请公布日 2014.02.05
(85)PCT国际申请进入国家阶段日
2013.11.29
(86)PCT国际申请的申请数据
PCT/US2011/038609 2011.05.31
(87)PCT国际申请的公布数据
WO2012/166120 EN 2012.12.06
(73)专利权人 惠普发展公司,有限责任合伙企业
地址 美国德克萨斯州
(72)发明人 B.V.彻斯 I.拉戈勒 P.E.哈默 R.A.斯皮特勒 S.P.费 P.S.贾格达尔
(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 马丽娜 徐红燕
(51)Int.Cl.
G06F 21/10(2013.01)
G06F 15/16(2006.01)
G06F 11/36(2006.01)
(56)对比文件
US 2007/0156644 A1,2007.07.05,
US 6996845 B1,2006.02.07,
US 2005/0044420 A1,2005.02.24,
CN 1610887 A,2005.04.27,
US 2002/0010855 A1,2002.01.24,
审查员 唐剑

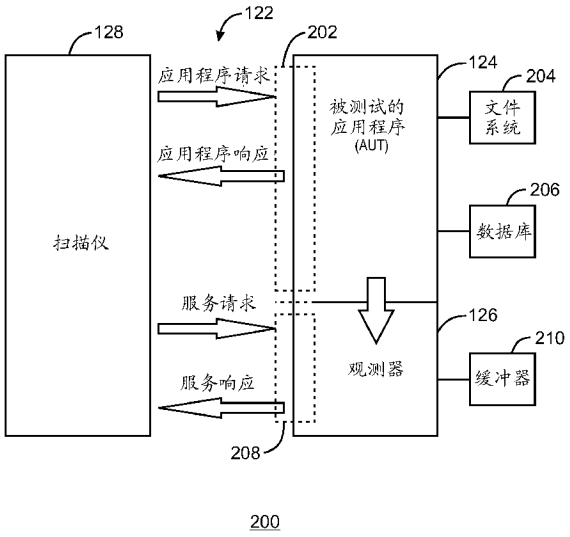
权利要求书2页 说明书11页 附图4页

(54)发明名称

应用程序安全测试

(57)摘要

本公开提供一种系统,其包括主管被测试的应用程序(AUT)的服务器,被配置为监控由AUT执行的指令的观测器,和通过公共的通信信道被通信地耦合到AUT和观测器的计算装置。计算装置可以被配置为发送应用程序请求到AUT,其中应用程序请求被配置为暴露AUT的潜在的漏洞。计算装置可以根据AUT的编程从AUT接收应用程序响应。计算装置可以发送服务请求到观测器,并且从观测器接收服务响应,其包含对应于由于应用程序请求由AUT执行的指令的信息,关于AUT的信息,或者关于主管AUT的服务器的信息。



CN 103562923 B

1.一种应用程序安全测试系统，包括：

主管被测试的应用程序(AUT)的服务器；

观测器，其被配置为监控由所述AUT执行的指令；和

计算装置，其通过公共的通信信道被通信地耦合到所述AUT和所述观测器，所述计算装置包括处理器和用于存储计算机可读指令的存储器装置，所述计算机可读指令被配置为指示所述处理器执行以下操作：

发送应用程序请求到所述AUT，其中所述应用程序请求被配置为暴露所述AUT的潜在的漏洞；

根据AUT的编程从所述AUT接收应用程序响应；

发送服务请求到所述观测器；并且

从所述观测器接收服务响应，其包含对应于由于所述应用程序请求由所述AUT执行的指令的信息，关于所述AUT的信息，或者关于主管所述AUT的服务器的信息。

2.权利要求1的系统，其中所述观测器被配置为生成标识由于所述应用程序请求由所述AUT执行的指令的踪迹，并且在所述服务响应的主体中发送所述踪迹到所述计算装置。

3.权利要求1的系统，其中所述观测器被配置为至少部分地通过将定制的标题添加到所述应用程序响应来与所述计算装置通信。

4.权利要求1的系统，所述存储器装置包括计算机可读指令，其被配置为指示所述处理器从所述观测器接收踪迹信息，所述踪迹信息包括多个漏洞踪迹节点，每个漏洞踪迹节点包含对应于由所述观测器检测的漏洞的代码位置。

5.权利要求4的系统，所述存储器装置包括计算机可读指令，其被配置为基于包含相同代码位置的漏洞踪迹节点来指示所述处理器将所述多个漏洞踪迹节点分组。

6.权利要求1的系统，其中所述观测器被配置为监控所述AUT以识别新的在所述AUT的运行时间期间动态地生成的统一资源定位符(URL)，并且在所述应用程序响应的标题中返回更新字段，所述更新字段被配置为通知所述计算装置所述AUT的攻击表面已经改变。

7.权利要求1的系统，其中所述观测器被配置为：

从所述计算装置接收请求并且分析所述请求的标题；

基于所述标题的分析来识别所述请求为所述应用程序请求还是所述服务请求；

将所述应用程序请求传递到所述AUT；并且

在不将所述服务请求传递到所述AUT的情况下处理所述服务请求。

8.一种应用程序安全测试方法，包括：

通过处理器发送应用程序请求到被测试的应用程序(AUT)，其中所述应用程序请求被配置为暴露所述AUT的潜在的漏洞；

通过所述处理器根据所述AUT的编程从所述AUT接收应用程序响应；

通过所述处理器发送服务请求到监控由所述AUT执行的指令的观测器；并且

通过所述处理器从所述观测器接收服务响应，其包含对应于由于所述应用程序请求由所述AUT执行的指令的信息，关于所述AUT的信息，或者关于主管所述AUT的服务器的信息；

其中所述应用程序请求，应用程序响应，服务请求，和服务响应通过相同的网络信道被传送。

9.权利要求8的方法，包括在所述应用程序响应中接收文件未找到标题，通过所述观测

器将所述文件未找到标题添加到所述应用程序响应以表明由所述AUT产生文件未找到错误。

10.权利要求8的方法，其中所述服务请求是踪迹服务请求，所述方法包括在从所述观测器接收的所述服务响应的主体中接收栈踪迹。

11.权利要求10的方法，包括在所述服务响应的所述主体中接收漏洞踪迹节点，其中所述漏洞踪迹节点标识由所述观测器检测到的漏洞。

12.权利要求8的方法，其中所述服务请求是攻击表面服务请求，所述方法包括在所述服务响应的主体中接收关于所述AUT的所述攻击表面的信息，所述攻击表面包括在运行时间期间由所述AUT生成的静态URL和动态URL。

应用程序安全测试

背景技术

[0001] 软件安全测试被用来识别应用程序(例如Web应用程序)中的漏洞。用于基于Web的软件的传统黑盒子安全测试通过使用安全测试应用程序来工作,经常被称作扫描仪,其伪装成攻击者。在黑盒子方法中,扫描仪通过作出HTTP请求并估计HTTP响应来探查被测试的应用程序(AUT)以便找到所有URL(在此处AUT接受输入)。AUT在此处接受输入的URL可以涉及AUT的攻击表面。于是扫描仪基于攻击表面和可能的漏洞类别生成攻击。扫描仪通过估计程序的HTTP响应来应用攻击以诊断漏洞的存在或不存在。在黑盒子方法中,扫描仪对AUT的内部工作不具有任何洞察力。

[0002] 黑盒子漏洞测试在概念上是易懂的,但是在实践中其提出一些挑战。例如,探查AUT可能不揭示所有攻击表面,因此扫描仪可能不对AUT易受伤害的所有地方发起攻击。另外,一些漏洞不能通过在HTTP响应中返回的信息被准确地识别。如果扫描仪未发现漏洞,扫描仪就不能提供关于漏洞在AUT的代码内何处的信息。此外,扫描仪可以报告都与在AUT中相同潜在的问题相关的几个漏洞,引起程序员设法修补漏洞以完成大量的重复工作。

附图说明

[0003] 在下面详细的描述中并且参考附图来描述某些实施例,其中:

[0004] 图1是根据实施例的可以被用来进行灰盒子安全测试的系统的框图;

[0005] 图2是示出根据实施例的用于进行灰盒子安全测试的测试系统配置的框图;

[0006] 图3是根据实施例的执行灰盒子安全测试的方法的过程流程图;和

[0007] 图4是示出根据实施例的存储被配置为进行灰盒子安全测试的代码的非暂时性计算机可读介质的框图。

具体实施方式

[0008] 在本文描述的实施例提供用于执行Web应用程序的灰盒子安全测试的技术。在灰盒子安全测试中,在本文中被称作观测器的软件程序被用来观察由AUT执行的内部操作。观测器使得扫描仪能够确定AUT的操作和其响应攻击如何表现。观测器也使得扫描仪能够确定AUT响应于正常应用程序请求的行为,扫描仪可以使用其来确定发送何种类型的攻击。扫描仪继续向AUT传递攻击,并且扫描仪从观测器接收AUT的内部工作的知识。这样,扫描仪能找到更多漏洞并且产生更好的漏洞报告,从而提供基于web的应用程序的更加全面和详细的软件安全测试。

[0009] 根据实施例,在观测器和扫描仪之间提供通信信道。扫描仪使用这个通信信道来在其扫描期间深入了解AUT。可以通过使用已经由AUT使用的通信信道来实施在扫描仪和观测器之间的通信信道。这样,进行测试的人不必执行另外的配置或者设置工作,并且通信信道不干扰AUT或者在其上运行AUT的计算机系统的正常操作。本发明的另外的益处可以参照下面提供的描述而被更好地理解。

[0010] 图1是根据实施例的可被用来进行灰盒子安全测试的系统的框图。系统通常由参

考数字100来指代。本领域普通技术人员将领会到在图1中示出的功能块和装置可以包括包含电路的硬件元件，包含在非暂时性机器可读介质上存储的计算机代码的软件元件，或者硬件和软件元件两者的结合。此外，配置不局限于在图1中示出的配置，因为在本发明的实施例中可以使用任何数目的功能块和装置。基于对具体电子装置的涉及考虑，本领域普通技术人员将容易地能够定义具体的功能块。

[0011] 如在图1中所示，系统100可以包括计算装置102，其通常将包括通过总线106连接到显示器108的处理器104，键盘110，和一个或多个输入装置112，例如鼠标，触摸屏，或键盘。在实施例中，计算装置102是通用计算装置，例如台式计算机，膝上型计算机，服务器等。计算装置102也可以具有一种或多种类型的非暂时性计算机可读介质，例如可以在各种操作程序(包括在本发明的实施例中使用的操作程序)的执行期间使用的存储器114。存储器114可以包括只读存储器(ROM)，随机存取存储器(RAM)等。装置102也可以包括其它非暂时性计算机可读介质，例如用于操作程序和数据(包括在本发明的实施例中使用的操作程序和数据)的长期存储的存储系统116。

[0012] 在实施例中，计算装置102包括网络接口控制器(NIC)118，用于将装置102连接到服务器120。计算装置102可以通过网络122，例如因特网，局域网(LAN)，广域网(WAN)，或者其它网络配置被通信地耦合到服务器120。服务器120可以具有非暂时性计算机可读介质，例如存储装置，用于存储数据，缓冲通信，和存储服务器120的操作程序。可以使用请求响应协议例如超文本传输协议(HTTP)来进行在装置102和服务器120之间的通信。

[0013] 服务器120可以是主管AUT 124的应用程序服务器。服务器120也包括在执行期间监控AUT 124的观测器126。计算装置102可以包括对AUT 124执行安全测试的扫描仪128。例如，扫描仪128可以通过网络122发送HTTP请求到AUT 124，其中HTTP请求被配置为尝试暴露AUT 124的漏洞。HTTP请求可以包括HTTPS请求，其将超文本传输协议与SSL(安全套接层)和TLS(传输层安全)协议结合以提供加密通信并且确保网络Web服务器的识别。在由AUT 124的HTTP请求的处理期间，观测器126监控由AUT 124执行的内部过程。例如，观测器126可以识别由AUT 124执行的代码的行，被存取的文件，被执行的数据库询问等。观测器126和AUT 124两者可以被配置为通过相同的HTTP信道与扫描仪128通信。如进一步参照图2描述的，从扫描仪128发送到服务器120的一些请求可以根据其编程把AUT 124作为目标以从AUT 124引起响应。从扫描仪128发送到服务器120的其它请求可以把观测器126作为目标以得到关于具体请求对由AUT 124执行的操作所具有的效果的附加的信息，或者与AUT 124，观测器126，或者主管AUT 124的服务器120相关的其它信息。响应于应用程序请求和服务请求由扫描仪128接收的数据可以被扫描仪128使用以生成漏洞报告。可以通过由扫描仪128提供的用户界面来将漏洞报告显示给用户。

[0014] 图2是示出根据实施例的用于进行灰盒子安全测试的测试系统配置的框图。系统200可以包括扫描仪128，AUT 124，和观测器126。AUT 124可以用任何合适的基于Web的计算机语言，例如JAVA，或.NET等等来编码。AUT 124可以在合适的软件框架，例如Struts，Struts2，ASP.NET MVC，Oracle WebLogic，和Spring MVC等等内来操作。软件框架包括提供通用功能的一组共用代码模块，其可以由用户代码选择性地重写(overridden)或专门化以提供特定的功能。AUT 124可以被配置为执行Java虚拟机(JVM)，公共语言运行时(CLR)，用于处理来自扫描仪128的请求的其它运行时环境中的一个或多个实例。由软件框架或者运

行时环境的共用代码模块提供的编程指令可以被称作集装箱代码。对AUT 124特定的定制的编程指令可以被称作用户代码。

[0015] AUT 124包括网络接口202，用于通过网络122实现在扫描仪128和AUT 124之间的通信。当AUT 124被使得可用于一般用途时，网络接口202暴露AUT 124的攻击表面并且是将最终被用来提供对AUT 124存取的相同的接口。可以通过从扫描仪128发出到AUT 124的HTTP请求和从AUT 124发出到扫描仪128的HTTP响应来进行在扫描仪128和AUT 124之间的通过网络接口202的通信。把AUT 124作为目标的请求可以被称作应用程序请求，并且从AUT 124接收的响应可以被称作应用程序响应。由扫描仪128生成的应用程序请求可以被配置为暴露AUT 124的潜在的漏洞。

[0016] AUT 124可以被耦合到文件系统204，数据库206，和由AUT 124使用的其它资源。数据库206可以包括多种用户信息，例如诸如用来允许存取AUT 124的各种资源的用户名和密码的表格。文件系统204可以包括由AUT 124使用的数据和程序，以及可由用户请求的数据，例如HTTP页面，软件程序，媒体文件等。

[0017] 观测器126在AUT 124的执行环境内操作并且可以使用由AUT 124执行的内部操作。例如，观测器可以通过在各种程序点处注入另外的代码，例如JAVA类来修改AUT 124的字节码。注入的代码充当观测AUT 124的监控器。注入的监控器代码可以被定位在AUT 124中的战略程序点处，例如在执行特定操作诸如读取URL参数或者写入到文件系统204的应用程序编程接口(API)调用处。无论何时执行AUT 124中的这种程序点，监控器都调入由观测器126提供的服务以记录由AUT 124执行的操作。观测器126可以被耦合到缓冲器210，用于存储已经被收集的关于AUT 124的内部操作的信息。缓冲器210可以被用来存储已经被收集但还未被报告到扫描仪128的数据。缓冲器210可以被存储在非易失性存储介质例如硬盘，固态驱动器等中。

[0018] 观测器126也可以包括另外的网络接口208，用于通过网络122实现在观测器126和扫描仪128之间的通信。如上面指出的，两个网络接口202和208可以使用相同的通信信道，例如相同的HTTP信道。在扫描仪128和观测器126之间的通信可以通过定制的请求和响应标题的使用来实现。定制的标题可以通过扫描仪128被添加到应用程序请求，并且定制的标题可以通过观测器126被添加到应用程序响应。这样，在扫描仪128和观测器126之间的通信中的至少一些可以在与AUT 124的正常通信上被捎带(piggy-backed)。使用单一信道的通信除去了打开专用的，辅助信道的任何问题，并且加入HTTP标题通常不干扰AUT 124的正常操作。

[0019] 扫描仪128可以将一个或多个定制的标题加入到每个应用程序请求，其中定制的标题包括观测器126可以用来诊断与进行中的攻击相关的漏洞的信息。在定制的标题内的信息可以包括扫描仪128的版本或者扫描仪128在攻击中使用的有效负载。可以由观测器126使用有效负载信息来确定攻击是否成功。

[0020] 扫描仪128也可以使用定制的请求标题来生成把观测器126作为目标的请求以得到关于由AUT 124执行的内部过程的另外的信息，或者关于AUT，服务器120(图1)，或观测器126的信息。把观测器126作为目标的请求可以被称作服务请求，并且从观测器126接收的响应可以被称作服务响应。由观测器126发出的服务响应可以包括在服务响应的主体中的补充信息，如下面进一步描述的。

[0021] 在实施例中，观测器126被配置为接收从扫描仪128发送到AUT 124的应用程序请求和服务请求。于是观测器126可以分析标题信息以确定请求是应用程序请求还是服务请求。在接收到应用程序请求时，观测器126可以分析标题信息以获得由观测器126使用的关于特定应用程序请求的数据。然后应用程序请求可以由观测器126传递到AUT 124用于根据AUT的编程由AUT 124来处理。当AUT 124生成应用程序响应时，观测器126可以将一个或多个定制的标题加入到应用程序响应以将另外的信息发回到扫描仪128。加入到应用程序请求和应用程序响应的定制的标题可以被称作每一请求标题并且在题为“每一请求标题”的部分中被进一步描述。

[0022] 在接收到服务请求时，观测器126可以在未将服务请求传递到AUT 124的情况下处理请求。服务请求可以包括一个或多个定制的标题，其包括被配置为请求观测器126的具体服务，例如被请求的服务的名称的信息。观测器126可以以在HTTP响应的主体中被请求的信息来响应，在本文被称作“服务响应”。在实施例中，在服务响应的主体中由观测器126提供的信息可以使用Java脚本对象表示法(JSON)来被格式化并且可以是自识别JSON对象。如果观测器126没有发送的信息，响应主体可以是空的。服务请求在题为“服务请求”的部分中被进一步描述。

[0023] 每一请求标题

[0024] 每一请求标题可以是定制的HTTP标题，其可以包括定制的字段名，后面是被观测器126和扫描仪128理解的一个或多个字段值。定制的HTTP标题被AUT 124忽略。将领会到在本文描述的字段名仅被用作字段名的示例，其可以在特定实施方式中使用并且不旨在限制权利要求的范围。

[0025] 每一请求标题可以包括用来协调在扫描仪128和观测器126之间的相互作用的版本标题。观测器126可以将版本标题加入到每一应用程序响应。扫描仪128可以使用版本标题来验证观测器126被安装。作为示例，版本标题可以被如下格式化：

[0026] X-WIPP-Version: <language> / <version> / <vm_id>

[0027] 在示例版本标题中，词头“X-WIPP”将标题标识为由观测器126使用的定制标题。字段名“X-WIPP-VERSION”是将定制的标题唯一地标识为版本标题的字符串。<language>字段值可以是由AUT 124使用来处理应用程序请求的运行时环境的名称，例如Java或.NET等等。在一些情况下，AUT 124可以执行两个或更多个过程，其可以被不同的运行时实例来处理。例如，AUT 124可以使用负载平衡器或者用于处理应用程序请求的其它工作分配布置。扫描仪128可以使用<vm_id>字段值来标识在AUT 124内处理应用程序请求的过程。例如，<vm_id>字段值可以是唯一地标识处理应用程序请求的运行时实例(例如特定JVM实例(在JAVA的情况下)，CLR实例(在.NET的情况下)，或其它类型的运行时实例)的名称。<version>字段值可以是标识观测器126的版本的数字或者字符串。如果观测器的接口208在观测器软件的版本之间变化，扫描仪128可以使用观测器126的被标识的版本来适当地协调与观测器126的相互作用。

[0028] 每一请求标题也可以包括由扫描仪128使用的文件未找到(FNF)标题来识别文件未找到条件。在HTTP中，如果客户请求不存在或者不能被找到的资源，Web应用程序可以生成被称作HTTP代码404的标准错误代码。简单的Web应用程序经常通过在HTTP响应中返回HTTP代码404来表示文件未找到条件。更加复杂的Web应用程序可以“吞下”404代码。换句话

说，在HTTP响应中不是仅返回HTTP代码404错误，而是代码404可以触发Web应用程序将客户重定向到错误页面，着陆页，或者Web应用程序的任何其它部分。在传统的黑盒子测试中，将扫描仪128重定向到Web应用程序的不同部分而不是仅报告错误可能引起扫描仪128不正确地报告错误肯定(false positive)。FNF标题可被用来避免这种结果。因为观测器126在应用程序内操作，观测器126可以检测文件未找到错误并且通过将FNF标题加入到应用程序响应来报告文件未找到错误。例如，如果应用程序请求引发来自AUT 124的文件未找到响应，观测器126可以将下面的标题加入到应用程序响应：

[0029] X-WIPP-FNF:404

[0030] 这样，可以向扫描仪128报告文件未找到错误，而不管由AUT 124提供的HTTP响应。

[0031] 每一请求标题也可以包括请求ID标题。请求ID标题可以被如下格式化：

[0032] X-WIPP-RequestID:<request_id>

[0033] 如下面进一步被描述的，在题为“服务请求”部分下面，响应于应用程序请求，但在应用程序响应中不被报告，扫描仪128可以请求由观测器126收集的另外的信息。另外的信息可以被包括在本文被称作“踪迹”的数据结构中。在踪迹中包括的信息描述AUT 124的操作，其由特定的应用程序请求触发。为了支持踪迹请求服务，观测器126可以将请求ID标题加入到每个应用程序响应以使得扫描仪128能够将被请求的踪迹与对应于被请求的踪迹的特定应用程序响应相关联。在实施例中，<request_id>字段值被扫描仪128分配并且被包括在应用程序请求中。然后观测器126可以在请求ID标题中使用相同的request_id值，其被加入到对应的应用程序响应。在实施例中，扫描仪128不将请求ID标题加入到应用程序请求，在该情况下，观测器126可以为request_id生成唯一的值并且包括在请求ID标题中被加入到应用程序响应的request_id。在任一情况下，可以由扫描仪128使用相同的request_id值以从观测器126请求对应的踪迹。

[0034] 每一请求标题也可以包括由观测器126使用的更新标题以通知扫描仪128关于AUT 124的各种类型的变化。更新标题可以被如下格式化：

[0035] X-WIPP-Update:<service_name_list>

[0036] <service_name_list>值可以是逗号分开的服务名称列表，其每个指的是由观测器126提供的服务，由于在AUT 124中的变化，其可以提供新的信息。当关于AUT 124的新的信息变得可用时，观测器126可以通过将更新标题加入到应用程序响应来通知扫描仪128。这样，更新标题通知扫描仪128关于AUT 124的新的信息可用并且请求什么服务来得到信息。例如，如果观测器126检测到在AUT 124的安全测试期间已经生成的另外的URL，观测器126可以将更新标题发送到扫描仪128，其中<service_name_list>等于“攻击表面”。在接收到更新标题时，扫描仪128可以将服务请求发送到感测器126，请求被识别的一种或多种服务。观测器126可以在每个应用程序响应中继续发送更新标题直到扫描仪128为命名的一种或多种服务发出服务请求为止。

[0037] 服务请求—踪迹

[0038] 响应于应用程序请求，观测器126可以通过确定例如已经由AUT 124执行的代码的特定行，已经由AUT 124存取的文件，由AUT 124执行的数据库询问，或者其它信息来确定应用程序请求的效果。由观测器126收集的数据可被存储到在本文被称作“踪迹”的数据结构。

在实施例中，每个踪迹可以被存储到缓冲器210。每个踪迹可以包括应用程序请求的请求ID和与踪迹对应的应用程序响应。扫描仪128可以通过从观测器126重新得到对应的踪迹来了解由特定的应用程序请求触发的AUT 124的内部操作。为了重新得到踪迹，扫描仪128可以向观测器126发出服务请求，其包括标题字段名/值对，所述标题字段名/值对被配置为表示对应于特定应用程序请求或者响应的踪迹的请求。例如，用于请求踪迹的字段名/值对可以被如下格式化：

[0039] Trace=<request_id>

[0040] 值<request_id>是由扫描仪128或者观测器126分配的值，其与和被请求的踪迹相关联的应用程序请求和/或应用程序响应对应，如上面关于题为“每一请求标题”的部分描述的。在接收到踪迹服务请求时，观测器126可以绕过AUT 124并且生成包括被请求的踪迹的服务响应。在实施例中，被请求的踪迹可以通过观测器126从缓冲器210重新得到并且被加入到服务响应的主体，然后可以将其发送到扫描仪128。服务响应标题包括被请求的踪迹的request_id值，并且服务响应的主体可以被格式化为JSON对象。

[0041] 观测器126可以在缓冲器210中保持多个踪迹以便扫描仪128可以为已经作出的任何应用程序请求请求踪迹。缓冲器210可以具有适合于特定实施方式的任何尺寸。在实施例中，如果缓冲器210变为满的，存储到缓冲器210的踪迹可以从缓冲器210中以先入先出的方式被去除。如果扫描仪128请求未知的请求ID，观测器126可以返回错误。如果其是无效的，从未被使用，或者已经是由观测器126保持的踪迹的缓冲器210的过期的，请求ID可以是未知的。

[0042] 在作出对应的应用程序请求并且从AUT 124接收到响应之后，扫描仪128可以被配置为发送单独的踪迹服务请求。request_id使得观测器126能够接收失序踪迹请求，同时仍然能够使接收到的踪迹与适当的应用程序请求和响应相关联。部分地因为扫描仪128可能具有发出应用程序请求到AUT 124的多个执行的线程，踪迹请求可以是接收到的失序应得物(due)。扫描仪128也可以被配置为中断超时的应用程序请求，在该情况下，扫描仪128可以从观测器126重新得到不完整的踪迹。为了在完整的和不完整的踪迹之间区别，观测器126可以被配置为将特殊节点加入到每个表明对应于这种踪迹请求的应用程序请求被成功地完成的完成的踪迹。例如，观测器126可以在每个完成的踪迹的末端处加入类型为“request_complete”的特殊节点。“request_complete”节点的缺少可以向扫描仪128表明对应的应用程序请求失败。

[0043] 观测器126可以监控由AUT 124执行的在应用程序请求的情境之外发生的过程，例如通过由观测器126注入的另外的监控代码启动的过程。为了避免招致性能开销的不可接受的水平，观测器126可以被配置为将与应用程序请求无关的监控过程的性能开销最小化。例如，性能开销可通过注入监控代码以选择性地监控特定API调用和AUT的用户代码的相关部分来被最小化。

[0044] 返回到扫描仪128的踪迹可以包括一个或多个不同类型的踪迹节点。每个踪迹节点传递对应于由AUT 124执行的内部过程的一些比特的信息。在实施例中，每个踪迹节点包括类型属性，其可以是唯一地标识踪迹节点的类型的任何合适的字符串。一些踪迹节点类型可以基于由AUT 124执行的动作的类型。

[0045] 在实施例中，观测器126可以记录关于由AUT 124使用的调用栈和集装箱代码的信

息。调用栈是存储关于计算机程序的动态子程序的信息的数据结构。例如，当其完成执行时，调用栈可以记录动态子程序应将控制返回到其的一行代码。调用栈也可以被用来传递参数到子程序，并且为子程序本地的变量分配存储器，以及其它功能。调用栈可以包括代表当前执行的由AUT 124或AUT的集装箱代码调用的子程序的顶部栈框架。顶部栈框架可以包括文件名称和标识代码的特定行的行编号。

[0046] 每个踪迹可以包括一个或多个踪迹节点，其中每个踪迹节点描述关于由AUT 124生成的特定调用栈的细节。踪迹节点可以包括位置属性，其在AUT 124集装箱代码中标识在观测器126外部的顶部栈框架的文件名称和行编号。如果这种栈框架可以被识别，踪迹节点也可以包括在AUT 124用户代码中给出用于顶部栈框架的文件名称和行编号的“user_context”属性。情境属性使得扫描仪128能够为漏洞生成包括根本原因分析的漏洞报告，并且使得扫描仪128能够将与在代码中相同位置相关联的漏洞分组在一起。

[0047] 在实施例中，观测器126被配置为检测漏洞。例如，扫描仪128可以以应用程序请求的形式发送攻击到AUT 124，所述应用程序请求被配置为在文件系统204上生成任意的文件。应用程序请求可以包括定制的标题信息，其通知观测器126关于攻击的性质。如果AUT 124是易受伤害的，观测器126将遇到文件创建API调用，从而通知观测器126文件上载攻击是成功的并且已经检测到漏洞。

[0048] 如果观测器126检测到漏洞，观测器126可以生成在本文被称作“漏洞踪迹节点”的踪迹节点。漏洞踪迹节点可以包括一个或多个栈踪迹，其向扫描仪128提供代码位置信息，例如下沉程序点和当可用时的一个或多个潜在的源程序点。源程序点是此处恶意输入被AUT 124使用的代码位置，并且下沉程序点是此处恶意输入修改AUT 124的行为的代码位置。例如，在跨站脚本漏洞的情况下，源程序点是从URL参数中读取用户供应值的地方，并且下沉程序点是保留的参数值被写入到HTML页面的地方。栈踪迹可以被削减使得它们不包括来自观测器126的栈框架。漏洞踪迹节点可以包括与AUT 124代码和集装箱代码两者相关的栈框架，并且每个栈框架可以包括栈框架与用户代码还是与集装箱代码相关的指示。由于由观测器126提供的代码位置信息，扫描仪128可以创建在AUT 124中准确地确定(pinpoint)漏洞的位置的漏洞报告，并且可以将在AUT 124中相同位置处发生的漏洞分组，从而在由扫描仪128生成的漏洞报告中减少重复。代码位置信息也给用户提供对漏洞的性质的深入了解并且因此减少用来修复问题的补救努力的量。

[0049] 漏洞踪迹节点也可以包括漏洞种类，例如“跨站脚本”或者“SQL注入”，和对应于每个漏洞种类的标准漏洞标识符，例如常见缺陷列举(CWE)标识符。漏洞踪迹节点也可以包括关于漏洞的检测的相关细节。例如，如果漏洞是SQL注入漏洞，漏洞踪迹节点可以包括在由观测器126进行的漏洞检测中涉及的SQL询问。这样，观测器126能够检测并且向扫描仪128报告SQL注入漏洞，即使漏洞在返回到扫描仪128的应用程序响应中不显现自身。

[0050] 如果AUT 124执行针对数据库的询问，例如SQL询问，观测器126也可以生成在本文被称作“数据库踪迹节点”的踪迹节点。数据库踪迹节点可以包括数据库询问的文本和在数据库询问中用于由AUT 124使用的绑定参数的值。其它类型的踪迹节点可以包括用于源代码文件的由AUT 124例如JAVA小服务程序，JAVA服务器页面(JSP)等调用的开始节点和结束节点。开始节点和结束节点指的是在代表通过AUT 124的执行流程的控制流结构中的节点。开始节点和结束节点可以包括源代码文件的文件名称和传递到源代码文件的参数。观测器

126也可以生成其它类型的踪迹节点以代表例如由AUT 124执行的文件系统204的读取和写入，由AUT 124执行的Web服务调用，和由AUT 124执行的网络服务操作等等。

[0051] 如上面所述，扫描仪128可以使用踪迹信息来把重复的漏洞分组在一起。重复漏洞的分组可以通过由扫描仪128实施的去重复过程来执行。在去重中，扫描仪128可以将散列算法应用到漏洞踪迹节点的各部分，例如user_context属性和漏洞种类，以便为漏洞创建标识符。从AUT 124的角度，具有相同标识符的两个漏洞可以被认为是复制品。漏洞标识符可以被用来通知用户修补这些漏洞中的一个将可能补救具有相同漏洞标识符的其它漏洞。扫描仪的用户界面可以被配置为将复制的漏洞提供到在一组中的用户。

[0052] 在实施例中，扫描仪128被配置为基于踪迹信息优化其发送到AUT 124的攻击。例如，如果踪迹显示特定的应用程序请求未访问文件系统204，扫描仪可以被配置为省略与针对与文件系统204相关的漏洞的该应用程序请求相关的类似的攻击。类似地，如果踪迹显示特定的应用程序请求未调用数据库询问，扫描仪可以被配置为省略与针对数据库中的漏洞的该应用程序请求相关的类似的攻击，例如SQL注入。

[0053] 在实施例中，来自数据库踪迹节点的数据库询问信息可以被用来识别更持久的跨站脚本漏洞。例如，由于应用程序请求，把数据库206作为目标的应用程序请求可以与由AUT 124访问的数据库表和列相关联。扫描仪128可以使用这种信息来发送试图在数据库206中存储数据的攻击和将数据写入到在数据库206中的特定位置的应用程序请求。扫描仪128可以通过发送从在数据库206中的相同的位置读取的应用程序请求来确定攻击的效果。

[0054] 服务请求—服务器信息

[0055] 观测器126可以被配置为提供在本文被称作“服务器信息服务”的服务，其用来向扫描仪128告知服务器120。为了重新得到服务器信息，扫描仪128可以向观测器126发出服务器信息服务请求，其包括被配置为表明服务器信息的请求的标题字段名，例如“服务器”。在接收到服务器信息服务请求时，观测器126可以绕过AUT 124并且将被请求的服务器信息返回到扫描仪128。

[0056] 被请求的服务器信息可以被返回在由观测器126生成的服务响应的主体中并且被格式化为例如JSON对象。在服务响应中包括的服务器信息的示例可以包括主操作系统的名称和版本，应用程序服务器的名称和版本，在没有任何停机时间情况下应用程序服务器已经运行的时间量，当前被处理的线程的数目，和当前使用的存储器的量，以及其它信息。扫描仪128可以使用服务器信息来生成适于主管AUT 124的服务器120的攻击。例如，如果主管AUT 124的服务器120正运行Linux，扫描仪128可以被配置为避免将基于Microsoft® Windows的攻击发送到AUT 124。

[0057] 服务请求—应用程序信息

[0058] 观测器126可以被配置为提供在本文被称作“应用程序信息服务”的服务，其被用来向扫描仪128告知AUT 124。为了重新得到应用程序信息，扫描仪128可以向观测器126发出应用程序信息服务请求，其包括被配置为表明应用程序信息的请求的字段名，例如“应用程序”。在接收到应用程序信息服务请求时，观测器126可以绕过AUT 124并且将被请求的应用程序信息返回到扫描仪128。

[0059] 被请求的应用程序信息可以被返回在由观测器126生成的服务响应的主体中并且被格式化为例如JSON对象。如果没有可用的应用程序信息被发送，服务响应的主体可以是

空的。由服务响应返回的应用程序信息的示例可以包括与AUT 124相互作用的所有数据库的名称和版本，由AUT 124使用的文件库，Web服务子系统，和与AUT 124相互作用的其它子系统和软件框架，以及其它信息。扫描仪128可以使用应用程序信息来更高效地生成攻击。例如，正被AUT 124使用的关于数据库的信息可以使得扫描仪128能够生成适于被识别的数据库的攻击并且避免为未使用的数据库生成攻击。另外，扫描仪128可以避免将把Microsoft® SQL服务器作为目标的攻击发送到仅使用Oracle数据库的AUT 124。

[0060] 服务请求—攻击表面

[0061] 观测器126可以被配置为提供在本文被称作“攻击表面服务”的服务，其用来识别可能不被简单的网络爬虫检测到的攻击表面的组成部分。为了重新得到攻击表面信息，扫描仪128可以发出包括被配置为表明攻击表面信息的请求的标题字段名(例如“攻击表面”)的攻击表面服务请求。在接收到应用程序信息服务请求时，观测器126可以绕过AUT 124并且将被请求的攻击表面信息返回到扫描仪128。攻击表面信息可以被返回在由观测器126生成的服务响应的主体中并且被格式化为例如JSON对象。如果没有可用的攻击表面信息被发送，服务响应的主体可以是空的。

[0062] AUT 124的攻击表面包括可进入扫描仪128的资源，例如诸如网页链接。扫描仪128或者观测器126可以被配置为分析AUT 124或者在AUT 124上“爬行”以发现这种网页链接。可进入扫描仪128的一些资源可以不与网页链接相关联并且在这种意义上是隐藏资源，其是不会通过在AUT 124上爬行被发现的。隐藏资源可以作为在文件系统204中的文件存在并且可以被可进入文件系统204的观测器126发现。另外，响应于接收到的应用程序请求，在运行时间，换句话说在AUT 124的执行期间，一些资源可以由AUT 124动态地生成。动态资源可以在运行时间期间基于配置文件和映射文件(例如Web.xml文件)生成，所述映射文件是基于预定义规则将被请求的URL映射到在文件系统204上的资源的文件。动态地生成的资源可以通过检查映射和配置文件并且通过观测AUT 124的执行来被观测器126发现以识别在运行时间动态绑定的统一资源定位符(URL)。

[0063] 可进入扫描仪128的每个资源可以被称作攻击表面组成部分。被观测器126发现的每个攻击表面组成部分可以在攻击表面服务响应的主体中被格式化为URL并且被报告给扫描仪128。每个攻击表面组成部分也可以在攻击表面服务响应的主体内被加标记以将攻击表面组成部分标识成静态的或者动态的。通过探查文件系统204，包括位于文件系统204的根目录及以下中的文件来发现的资源可以被标记为静态的。通过检查映射和配置文件发现的资源可以标记为动态的。在不将Web应用文档(WAR)文件扩展为运行应用程序的一部分的集装箱(例如WebLogic)的情况下，观测器126可以让集装箱代码来处理从WAR文件提取资源的任务并且然后使用被提取资源的列表来定义攻击表面。

[0064] 错误处理

[0065] 在一些情况下，扫描仪128可以发出不能由观测器126履行的服务请求。例如，扫描仪128可以发出不被观测器126认出的服务请求或者具有不被观测器126认出的标题字段值的服务请求，例如具有未知的请求ID的踪迹请求。如果观测器126遇到不能被履行的服务请求，观测器126可以在服务响应或者应用程序响应的标题中将错误返回到扫描仪128。例如，观测器126可以发出被如下格式化的服务响应：

[0066] X-WIPP-Error: <error_text_string>

[0067] 标题字段值<error_text_string>可以是给出遇到的错误的简要描述的任何合适的文本串。描述错误的文本串可以由扫描仪128存储到可由用户看见的错误日志。此外，观测器126可以保持错误日志，其中每个错误日志入口包括问题的更详细的描述。由观测器126保持的错误日志可以记录在观测器126的操作期间遇到的任何错误。

[0068] 可以响应于服务请求或者应用程序请求标题的包括旨在把观测器126作为目标的信息的部分来由观测器126生成错误。如果响应于服务请求生成错误，可以在服务响应中返回错误消息，并且服务响应的主体可以是空的。如果响应于在应用程序请求中包括的信息来生成错误，可以在应用程序响应中返回错误消息，并且应用程序响应的主体可以包含AUT 124根据其编程提供的任何数据。

[0069] 图3是概述根据实施例的执行灰盒子安全测试的方法的过程流程图。方法300可以被与AUT 124和126通信的扫描仪128来执行，如参考图2描述的。方法300可以在方框302处开始，其中应用程序请求可以被发送到AUT 124。应用程序请求可以被配置为暴露AUT 124的潜在的漏洞。如上面讨论的，应用程序请求可以包括被用来将信息传送到观测器126的定制的标题。例如，扫描仪128可以将请求ID值加入到唯一地标识对观测器126的应用程序请求的应用程序请求的标题。

[0070] 在方框304处，扫描仪128可以根据AUT的编程从AUT 124接收到应用程序响应。如果扫描仪128被配置为将请求ID加入到每个应用程序请求，观测器126可以将相同的请求ID加入到应用程序响应的标题。否则，扫描仪128可以生成唯一的请求ID并且将请求ID加入到应用程序响应的标题。如上面描述的，观测器126也可以将另外的信息加入到应用程序响应的标题，例如文件未找到标题，观测器版本标题，更新标题等。

[0071] 在方框306处，扫描仪128可以将服务请求发送到观测器126。在实施例中，服务请求可以通过将另外的标题信息加入到应用程序请求来被包括在方框302的应用程序请求中。在实施例中，服务请求可以是单独的请求，换句话说，没有与应用程序请求结合。服务请求由观测器126来处理并且不被传递到AUT 124。服务请求可以被配置为请求信息，例如攻击表面信息，服务器或应用程序信息，和与AUT 124的内部过程相关的踪迹信息等等。扫描仪128可以将请求ID加入到服务请求的标题以得到对应于特定应用程序响应的踪迹信息。

[0072] 在方框308处，扫描仪128可以从观测器126接收服务响应。服务响应可以包括关于由于应用程序请求由AUT 124执行的过程的信息。例如，服务响应可以包括标识由于应用程序请求由AUT 124执行的过程的栈踪迹信息。服务响应可以包括漏洞踪迹节点，其包含对应于由观测器126检测的漏洞的代码位置。服务响应也可以包括对应于由于应用程序请求由AUT 124执行的数据库询问的信息。在实施例中，服务响应可以包括关于AUT 124的信息，例如AUT 124的编程语言，AUT 124的名称和版本，和包括静态和动态URL的AUT 124的攻击表面以及其它信息。服务响应也可以包括关于观测器126的信息，例如操作系统名称和版本，应用程序服务器名称和版本，线程的数目，存储器使用率，和在没有任何停机时间的情况下服务器120已经运行的量以及其它信息。

[0073] 将领会到方法300只是用来解释本技术的实施例的示例过程流程，并且实际的过程流程可以根据特定的实施方式变化。例如，扫描仪128可以不为每个应用程序请求发出服务请求。另外，扫描仪128可以在发送与应用程序响应中的特定的一个应用程序响应相关的服务请求之前发送和接收多个应用程序请求和响应。

[0074] 在方框310处，扫描仪128可以基于从AUT 124和观测器126接收到的信息来生成漏洞报告。漏洞报告可以基于在方框308处接收到的包括在踪迹节点中的代码位置信息将检测到的漏洞分组。漏洞报告可以通过由扫描仪128提供的用户界面被呈现给用户。漏洞报告也可以被存储到存储器，被打印等。

[0075] 图4是示出根据实施例的存储被配置为进行灰盒子安全测试的代码的非暂时性计算机可读介质的框图。由参考数字400指代该非暂时性机器可读介质。非暂时性机器可读介质400可以包括RAM，硬盘驱动器，硬盘驱动器阵列，光学驱动器，光学驱动器阵列，非易失性存储器，通用串行总线(USB)驱动器，数字通用盘(DVD)，致密盘(CD)等。非暂时性机器可读介质900可以通过通信路径904由处理器902访问。

[0076] 如在图4中所示，本文讨论的各种部件可以被存储在非暂时性，机器可读介质400上。在非暂时性机器可读介质400上的区域406可以包括被配置为将应用程序请求发送到AUT 124的应用程序接口，其中应用程序请求被配置为暴露AUT 124的潜在的漏洞。应用程序接口也可以从AUT 124接收应用程序响应，其中应用程序响应根据AUT的编程由AUT 124生成。区域408可以包括被配置为将服务请求发送到观测器126的观测器接口。观测器接口也接收服务响应，其包含例如对应于由于应用程序请求由AUT 124执行的过程的信息，关于AUT 124的信息，或者关于主管AUT 124的服务器120的信息。区域410可以包括漏洞报告生成器，其被配置为分析从AUT 124和观测器126接收到的数据并且基于该分析生成漏洞报告。

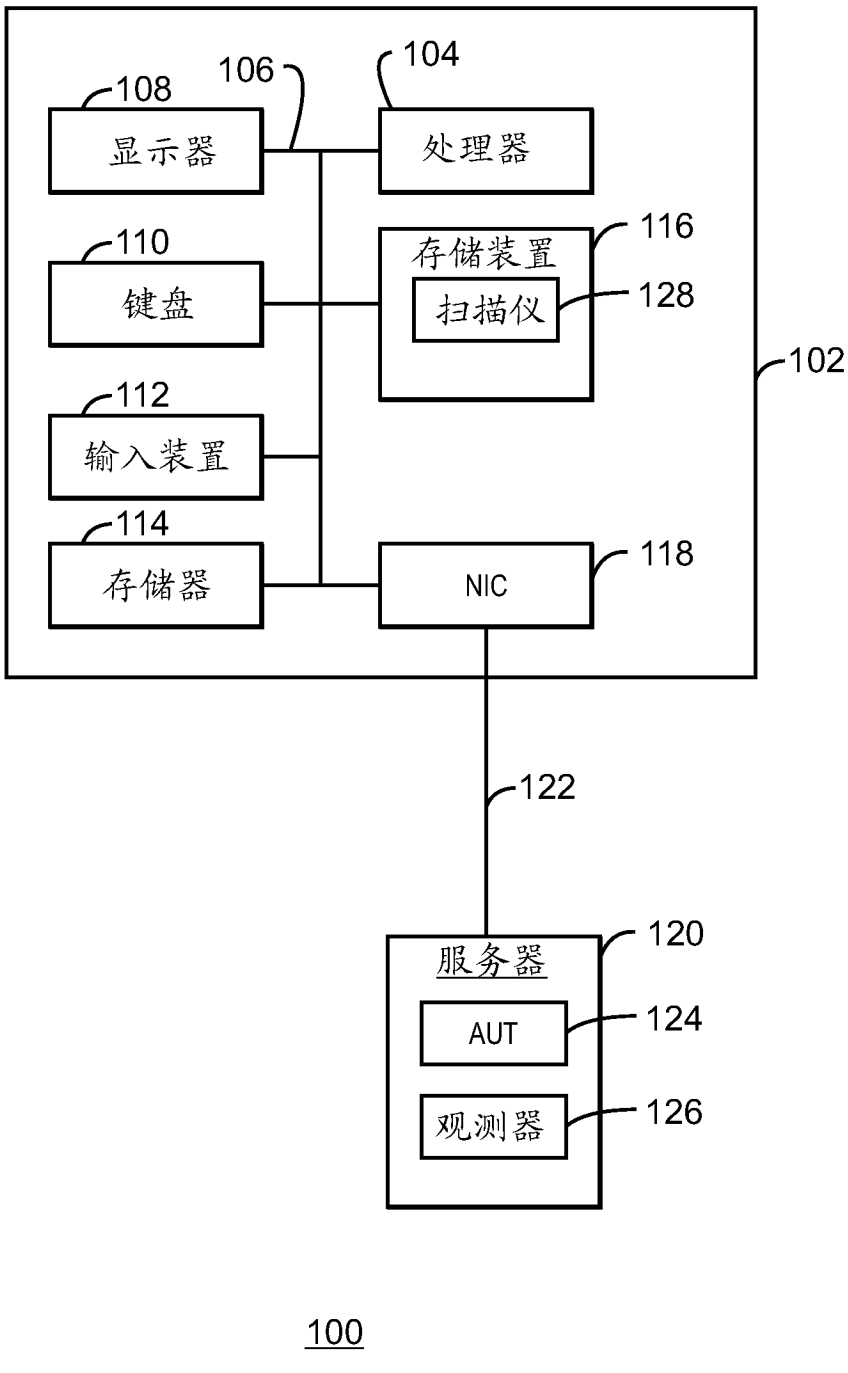

108
106
104
显示器
处理器
110
存储装置
116
键盘
扫描仪
128
102
112
输入装置
114
存储器
NIC
118
122
120
服务器
AUT
124
观测器
126
100

图 1

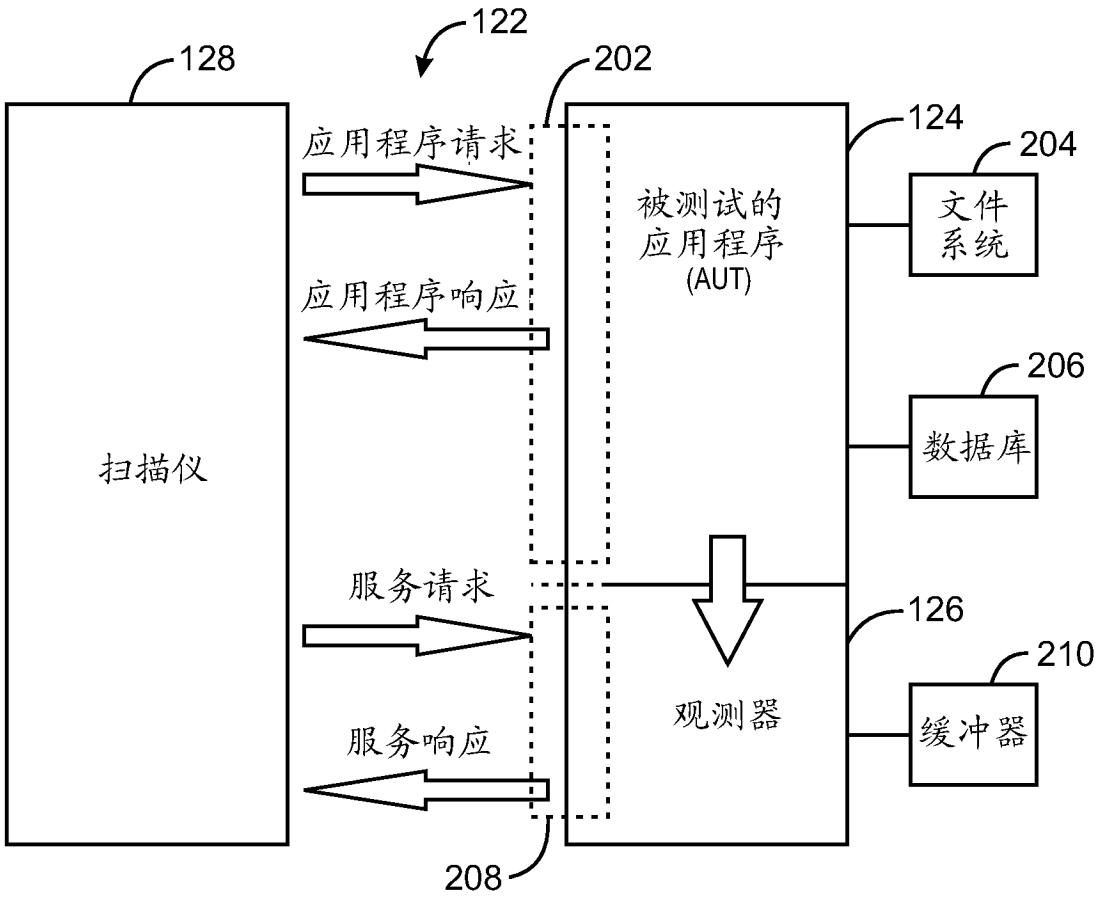

122
128
202
124
204
应用程序请求
被测试的
应用程序
(AUT)
文件
系统
应用程序响应
206
数据库
扫描仪
服务请求
126
210
观测器
缓冲器
服务响应
208
200

图 2

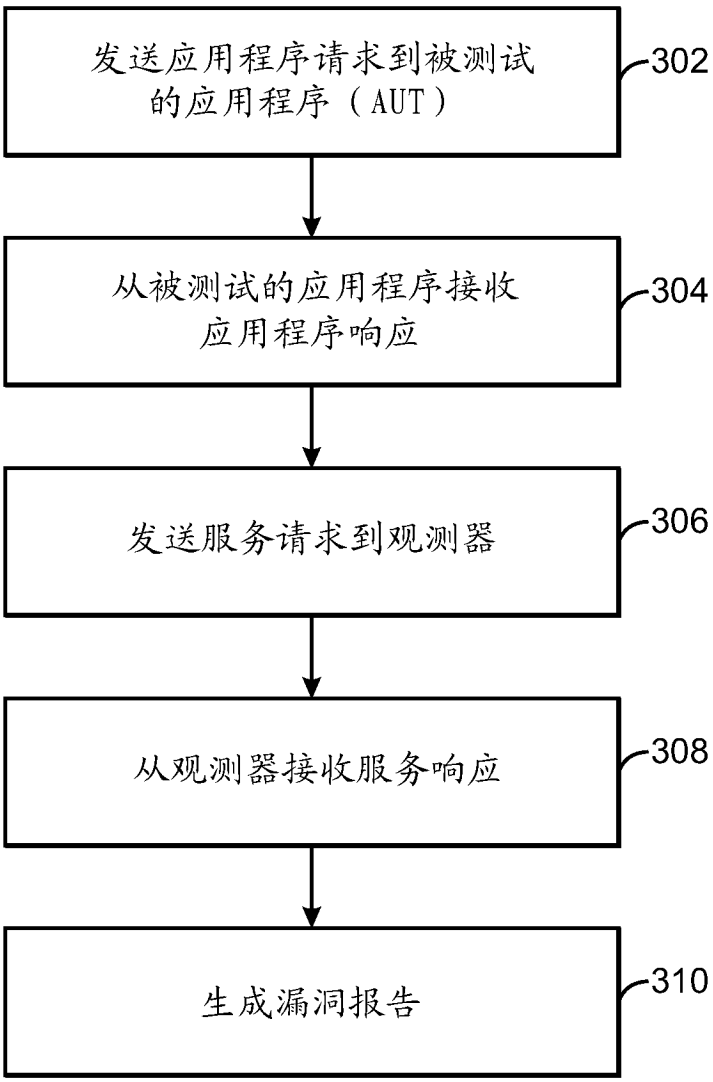

发送应用程序请求到被测试的应用程序（AUT）
302
从被测试的应用程序接收应用程序响应
304
发送服务请求到观测器
306
从观测器接收服务响应
308
生成漏洞报告
310
300

图 3

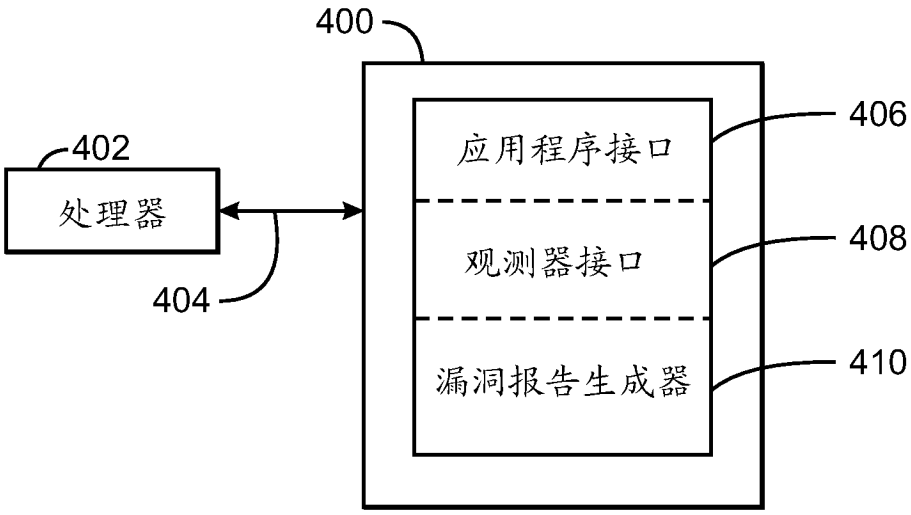


图 4