

NODE. JS组件漏洞检测方法及系统

技术领域

[0001] 本发明涉及一种漏洞检测技术，尤其是指一种NODE. JS组件漏洞检测方法及系统。

背景技术

[0002] 如今开发人员广泛使用开源组件，据估计，每个应用程序的80%-90%都由开源组件组成。有研究显示，软件应用程序中使用的第三方组件中有一半已经过时，可能不安全。而且使用开源组件的所有应用程序中，超过60%包含已知的软件漏洞。那么分析每个开源组件的CVE，将会给项目的组成分析（SCA）提供有效的信息支持。而目前市场上还没有相关成熟的技术和产品。因此为解决这一问题，一般都会采用人工的方式寻找一个个漏洞，再根据漏洞的描述去相关的产品官网找寻相关的信息，然后确定该NODE. JS组件的漏洞。然而，应用人工的方式审查漏洞，工作量大，效率低下。

发明概述

技术问题

[0003] 本发明所要解决的技术问题是：提供一种NODE. JS组件漏洞检测方法及系统，以此快速、高效的检测NODE. JS组件的漏洞。

问题的解决方案

技术解决方案

[0004] 为了解决上述技术问题，本发明采用的技术方案为：一种NODE. JS组件漏洞检测方法，包括如下步骤：

[0005] 从NODE. JS漏洞数据库中搜集第一基础漏洞信息；

[0006] 解析package. json文件，获得NODE. JS组件的关键信息；

[0007] 根据所述NODE. JS组件的关键信息，从所述第一基础漏洞信息提取第一目标漏洞信息。

[0008] 其中，所述根据所述NODE. JS组件的关键信息，从所述第一基础漏洞信息提取

第一目标漏洞信息的步骤中，包括如下步骤：

- [0009] 根据关键信息的相关度，设定关键信息优先级；
- [0010] 获取CVE信息，以搜集CPE信息；
- [0011] 按照所述关键信息优先级，针对所述NODE.JS组件的关键信息与CPE信息进行匹配，生成第一目标漏洞信息。
- [0012] 可选地，所述根据所述NODE.JS组件的关键信息，从所述第一基础漏洞信息提取第一目标漏洞信息的步骤之后，包括如下步骤：
 - [0013] 计算所述NODE.JS组件的sha1编码哈希值；
 - [0014] 应用所述NODE.JS组件的sha1编码哈希值与所述NODE.JS第一目标漏洞信息进行匹配，生成第三目标漏洞信息。
 - [0015] 进一步地，所述针对所述NODE.JS组件的关键信息与CPE信息进行匹配，生成第一目标漏洞信息之后，还包括如下步骤：
 - [0016] 从所述NODE.JS的关键信息中，提取NODE.JS组件名；
 - [0017] 确认所述NODE.JS组件名与所述CPE信息一一对应。
 - [0018] 可选地，所述生成第一目标漏洞信息的步骤之后，还包括如下步骤：
 - [0019] 调用NODE.JS组件的接口，从所述package.json文件中，获取第二目标漏洞信息。
 - [0020] 进一步地，所述NODE.JS组件的关键信息包括NODE.JS组件的名称信息及NODE.JS组件的版本信息，所述获取第二目标漏洞信息的步骤之后，还包括如下步骤：
 - [0021] 应用retirejs整理所述npm漏洞信息，获得第二基础漏洞信息；
 - [0022] 根据所述NODE.JS组件的名称信息及NODE.JS组件的版本信息，与所述第二基础漏洞信息进行匹配，生成第三目标漏洞信息。
 - [0023] 进一步地，所述生成第三目标漏洞信息的步骤之后，还包括如下步骤：
 - [0024] 定时下载更新的retirejs，以分析所述第三目标漏洞信息，生成第四目标漏洞信息。
 - [0025] 进一步地，所述根据所述NODE.JS组件的关键信息，从所述第一基础漏洞信息提取第一目标漏洞信息的步骤中，具体包括：
 - [0026] 根据所述NODE.JS组件的关键信息，获取版本信息，产品名称及厂商信息；

- [0027] 分别应用所述版本信息，产品名称及厂商信息，与所述CPE信息进行匹配，获得匹配信息；
- [0028] 根据所述匹配信息，提取对应的CVE信息；
- [0029] 其中，所述CVE信息包括CVE编号。
- [0030] 具体的，所述第三目标漏洞信息包括漏洞信息、版本号信息、危险等级信息及CVE信息中的一种或多种。
- [0031] 本申请还提供了一种NODE.JS组件漏洞检测系统，包括如下模块：
- [0032] 搜集模块，用于从NODE.JS漏洞数据库中搜集第一基础漏洞信息；
- [0033] 解析模块，用于解析package.json文件，获得NODE.JS组件的关键信息；
- [0034] 生成模块，用于根据所述NODE.JS组件的关键信息，从所述第一基础漏洞信息提取第一目标漏洞信息。

发明的有益效果

有益效果

- [0035] 本发明的有益效果在于：应用了本发明提供的NODE.JS组件漏洞检测方法，可以从NODE.JS漏洞数据库中搜集第一基础漏洞信息，快速获得可能存在的NODE.JS组件的漏洞信息。package.json文件是NODE.JS组件中的一个文件，在解析package.json文件，即可获得待检测NODE.JS组件的关键信息，有助于数据的调用与整理。由此，只需要检测少量关键信息，即可从待检测NODE.JS组件获得大量的漏洞信息。并以此生成第一目标漏洞信息。

对附图的简要说明

附图说明

- [0036] 下面结合附图详述本发明的具体结构
- [0037] 图1为本发明第一实施例中的NODE.JS组件漏洞检测方法；
- [0038] 图2为本发明第二实施例中从第一基础漏洞信息提取第一目标漏洞信息的步骤流程图；
- [0039] 图3为CVE的结构示意图；
- [0040] 图4为CPE的结构示意图；
- [0041] 图5为本发明第三实施例中被审核的表格；

- [0042] 图6为本发明第三实施例中审核表格的审核结果；
- [0043] 图7为本发明第四实施例中获取第三目标漏洞信息的流程图；
- [0044] 图8为本发明第五实施例中获取第三目标漏洞信息的流程图；
- [0045] 图9为本发明第五实施例中第二基础漏洞信息的示意图；
- [0046] 图10为本发明第六实施例中生成第四目标漏洞信息的结果图；
- [0047] 图11为本发明的NODE.JS组件漏洞检测系统第一实施例的结构图。

发明实施例

本发明的实施方式

- [0048] 为详细说明本发明的技术内容、构造特征、所实现目的及效果，以下结合实施方式并配合附图详予说明。
- [0049] 请参阅图1，图1为本发明第一实施例中的NODE.JS组件漏洞检测方法；
- [0050] 一种NODE.JS组件漏洞检测方法，包括如下步骤：
- [0051] 步骤S100、从NODE.JS漏洞数据库中搜集第一基础漏洞信息；
- [0052] 步骤S200、解析package.json文件，获得NODE.JS组件的关键信息；
- [0053] 步骤S300、根据NODE.JS组件的关键信息，从第一基础漏洞信息提取第一目标漏洞信息。
- [0054] 应用了本发明提供的NODE.JS组件漏洞检测方法，可以实现的功能如下：从NODE.JS漏洞数据库中搜集第一基础漏洞信息，快速获得可能存在的NODE.JS组件的漏洞信息。package.json文件是NODE.JS组件中的一个文件，在解析package.json文件，即可获得待检测NODE.JS组件的关键信息，有助于数据的调用与整理。由此，只需要检测少量关键信息，即可从待检测NODE.JS组件获得大量的漏洞信息。并以此生成第一目标漏洞信息。综上所述，可以较为准确地获取NODE.JS组件的漏洞信息，保障漏洞审核的效率及效果。
- [0055] 具体的实施例中，步骤S200中，解析package.json文件的步骤中，包括：
- [0056] 步骤S201、使用nodejs包管理工具npm，执行npm install组件名，生成node_modules文件架及package-lock.json文件或npm-shrinkwrap.json文件。
- [0057] 步骤S202、根据package-lock.json文件或npm-shrinkwrap.json文件，获取所

引用的组件。

- [0058] 步骤S203、将其他开源组件下载到node_modules文件夹。
- [0059] 应用上述方法，不仅可以获得NODE.JS组件的本身代码的漏洞，还可以获得被应用的代码漏洞。可以理解的是，其引用方式，可以通过继承、封装或其他方式。
- [0060] 具体的，请参阅图2，图2为本发明第二实施例中从第一基础漏洞信息提取第一目标漏洞信息的步骤流程图一种NODE.JS组件漏洞检测方法。步骤S300、根据NODE.JS组件的关键信息，从第一基础漏洞信息提取第一目标漏洞信息的步骤中，包括如下步骤：
- [0061] 步骤S310、根据关键信息的相关度，设定关键信息优先级。
- [0062] 关键信息的相关度具有多种确定方式，可以通过程序员的经验进行设定，也可以根据具体的软件来进行制定。在一具体实施例中，应用版本信息，产品名称及厂商信息依次用vender, product, version来表示。三者可以同属于高优先级，且三者之间具有一定优先级差异。
- [0063] 而在另一实施例中，可能并不直接用上述方法来表述，而是应用name字段来表述。此时的name字段可能是vender，也有可能是product，且优先级都是高，此时可以直接为其定义为高优先级的。还有一些字段，比如Description, author, maintainers, homepage, bugs等都有可能是vender，且优先级都为低。
- [0064] 步骤S320、获取CVE信息，以搜集CPE信息。
- [0065] CVE的英文全称是“Common Vulnerabilities & Exposures”，即通用漏洞披露。CVE为广泛认同的信息安全漏洞或者已经暴露出来的弱点给出一个公共的名称。使用一个共同的名字，可以帮助用户在各自独立的各种漏洞数据库中和漏洞评估工具中共享数据。CVE的结构如图3所示，而CVE信息可以包括多个CPE配置信息。
- [0066] 需要了解的是，其结构如图4所示，需要了解的是CPE的格式如下：
- [0067] cpe:2.3:part:vender:product:version:update:edition:language:sw_edition:target_sw:target_hw:other
- [0068] 其中，part表示目标类型，part可以是a、h、o中的任意一种；vender表示厂

商；product表示产品名称；version表示版本号；update表示更新包；edition表示版本；language表示语言项。

- [0069] 本实施例中，part是a，表示某一软件的漏洞信息，尤其是指Node.js组件。
- [0070] 步骤S330、按照关键信息优先级，针对NODE.JS组件的关键信息与CPE信息进行匹配，生成第一目标漏洞信息。
- [0071] 在解析出应用版本信息，产品名称及厂商信息的一系列值，以及其对应的优先级；与cpe信息中的vender，product和version进行匹配。按优先级从高到低，从高的vender、product和version进行匹配；
- [0072] 如果同一个优先级有多个的，比如高的优先级，vender，product和version对应的值有多个的情况，比如vender有cn、seczone，product有seczone、sca、sdlc，version有1.0、2.0，那每种情况都进行混合匹配。
- [0073] 在一实施例中，vender为cn，product为seczone，version为1.0，查找对应的cpe。待上述的vender，product和version中的一个匹配成功后，继续试用该高优先级的其他组合找匹配的cpe。最终根据找到的cpe找出所有匹配的cve信息。
- [0074] 当高优先级的匹配信息匹配失败之后，再用低优先级的信息进行匹配。
- [0075] 进一步地，步骤S300、根据NODE.JS组件的关键信息，从第一基础漏洞信息提取第一目标漏洞信息的步骤中，还包括：
- [0076] 步骤S301、根据所述NODE.JS组件的关键信息，获取版本信息，产品名称及厂商信息；
- [0077] 步骤S302、分别应用版本信息，产品名称及厂商信息，与CPE信息进行匹配，获得匹配信息。
- [0078] 由于NODE.JS组件的不同的版本可能具有不同的代码与框架，甚至可能只有文件名称是没有改变的，因此需要NODE.JS组件版本信息，才能更好地检测漏洞。而不同的厂商，也可能采用同样的名称来对不同的软件命名。
- [0079] 步骤S303、根据匹配信息，提取对应的CVE信息。
- [0080] 其中，CVE信息包括CVE编号。
- [0081] CVE编号是识别公开的漏洞的编号，是针对具体漏洞问题的编号。
- [0082] 进一步地，步骤S330中，针对NODE.JS组件的关键信息与CPE信息进行匹配，生

成第一目标漏洞信息之后，还包括如下步骤：

- [0083] 步骤S331、从NODE.JS的关键信息中，提取NODE.JS组件名；
- [0084] 步骤S332、确认NODE.JS组件名与CPE信息一一对应。
- [0085] 需要了解的是，CPE信息与CVE信息并不是一一对应的关系，一个CVE信息可以含有多个CPE信息，而一个CPE信息可以存在于多个CVE信息之中。基于此，需要去除第一目标漏洞信息中的重复信息，以保证JS脚本文件名与CPE信息一一对应的。本实施例中，将第一目标漏洞信息形成一个表格，再通过审核来去除重复信息，该表格如图5所示的表格，该过程可以是人工审核，也可以通过某些程序进行审核。其审核结果如图6所示。
- [0086] 进一步地，请参阅图7，图7为本发明第四实施例中获取第三目标漏洞信息的流程图。在步骤S300、根据NODE.JS组件的关键信息，从第一基础漏洞信息提取第一目标漏洞信息的步骤之后，包括如下步骤：
- [0087] 步骤S410、计算NODE.JS组件的sha1编码哈希值；
- [0088] 其中，sha1编码哈希值是JS脚本文件通过哈希算法算出的。应用哈希算法，可以将任意长度的二进制转化为固定长度的哈希值，应用哈希值，可以便捷快速地找出对应的文件。
- [0089] 步骤S420、应用NODE.JS组件的sha1编码哈希值与第一目标漏洞信息进行匹配，生成第三目标漏洞信息。
- [0090] 本实施例中，直接调用sha1编码哈希值来与第一目标漏洞信息中对应的信息进行匹配，由此，只需要对NODE.JS组件进行一次解析，即可较为快捷、准确地获得漏洞。既节省扫描所花费的时间，也避免了可能存在的部分数据分析不准确的情况。
- [0091] 在另一实施例中，在步骤S330中，生成第一目标漏洞信息的步骤之后，包括如下步骤：
- [0092] 步骤S340、调用NODE.JS组件的接口，从package.json文件中，获取第二目标漏洞信息。
- [0093] 本实施例中，调用NODE.JS官方所提供的接口，去寻找其他的漏洞，其步骤与上述步骤相似，在此不做赘述。而本实施例中，可以获得上述从CVE漏洞信息，

也可以获得一些非CVE漏洞信息，将这些信息组合到一起，就构成了第二目标漏洞信息。相对于第一目标漏洞信息，第二目标漏洞信息具有更全面的漏洞数据，可以保障NODE.JS组件的安全性。

[0094] 进一步地，NODE.JS组件的关键信息包括NODE.JS组件的名称信息及NODE.JS组件的版本信息。请参阅图8，图8为本发明第五实施例中获取第三目标漏洞信息的流程图。

[0095] 步骤S350、应用retire.js整理npm漏洞信息，获得第二基础漏洞信息。

[0096] 本实施例中，仍然运用与上述步骤相近的方式检测漏洞。然而，也具有不同之处，本实施例中，第二基础漏洞信息同时运用组件名及多个vulnerabilities记录其漏洞信息；在vulnerabilities中，应用atOrAbove及below表示其版本号信息，用severity表示严重等级，identifiers来表示漏洞的具体内容。

[0097] 本实施例中，有angular、hubot-scripts、connect、libnotify等组件名，针对每个组件名可以设置一个或多个vulnerabilities。在一个vulnerabilities中，应用atOrAbove表示版本号大于等于某个版本号，below是版本号小于等于某个版本号，以此划分区间。在此区间内，用severity表示该区间内的漏洞严重等级；identifiers来表示漏洞的具体内容，如果是cve漏洞，会有一个cve编号，如果不是CVE漏洞，概括性描述该漏洞的具体状态，具体如图9所示，图9为本发明第五实施例中第二基础漏洞信息的示意图。

[0098] 步骤S360、根据NODE.JS组件的名称信息及NODE.JS组件的版本信息，与第二基础漏洞信息进行匹配，生成第三目标漏洞信息。

[0099] 本实施例中，只需采用NODE.JS组件的名称信息及NODE.JS组件的版本信息，可以与上述的第二基础漏洞信息按照映射规则进行匹配，即可获得准确而全面的漏洞信息。

[0100] 可选地，步骤S360、生成第三目标漏洞信息的步骤之后，还包括如下步骤：

[0101] 步骤S370、定时下载更新的retirejs和/或package.json，以分析第三目标漏洞信息，生成第四目标漏洞信息。其中，第三目标漏洞信息包括漏洞信息、版本号信息、危险等级信息及CVE信息中的一种或多种。

[0102] 在一具体的实施例中，包括如下步骤：

- [0103] 步骤S371、制作定时下载程序。
- [0104] 步骤S372、应用定时程序，定时更新retires和/或package.json。
- [0105] 本步骤中，可以单独更新retires与package.json中的一种，也可以同时对二者进行更新。
- [0106] 步骤S373、针对第三目标漏洞信息修改或新增漏洞数据。
- [0107] 步骤S374、生成更新后的目标漏洞信息，即第四目标漏洞信息。
- [0108] 由此，可以保证漏洞数据与时俱进，降低本技术方案落后于时代的可能性。具体的如图10所示，CVE编号为CVE-2020-001的漏洞即为更新后的数据。
- [0109] 请参阅图11，本申请还提供了一种NODE.JS组件漏洞检测系统，包括如下模块：
- [0110] 搜集模块100，用于从NODE.JS漏洞数据库中搜集第一基础漏洞信息；
- [0111] 解析模块200，用于解析package.json文件，获得NODE.JS组件的关键信息；
- [0112] 检测模块300，用于根据NODE.JS组件的关键信息，从第一基础漏洞信息提取第一目标漏洞信息。
- [0113] 上述的模块用于承载上述方法，任意模块如果以软件功能模块的形式实现并作为独立的产品销售或使用时，可以存储在一个计算机可读取存储介质中。基于这样的理解，本发明的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的全部或部分可以以软件产品的形式体现出来。可以理解的是，本方法及系统应用于一种计算机可读存储介质，该计算机可读存储介质可以是存储器。该计算机可读存储介质上存储有计算机程序。进一步地，该计算机可读存储介质还可以是U盘、移动硬盘、只读存储器(ROM, Read-Only Memory)、随机存取存储器(RAM, Random Access Memory)、磁碟或者光盘等各种可以存储程序代码的介质。
- [0114] 需要说明的是，对于前述的各方法实施例，为了简便描述，故将其都表述为一系列的动作组合，但是本领域技术人员应该知悉，本发明并不受所描述的动作顺序的限制，因为依据本发明，某些步骤可以采用其它顺序或者同时进行。其次，本领域技术人员也应该知悉，说明书中所描述的实施例均属于优选实施例，所涉及的动作和模块并不一定是本发明所必须的。

[0115] 以上所述仅为本发明的实施例，并非因此限制本发明的专利范围，凡是利用本发明说明书及附图内容所作的等效结构或等效流程变换，或直接或间接运用在其他相关的技术领域，均同理包括在本发明的专利保护范围内。