(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 118886008 A
(43)申请公布日 2024.11.01

(21)申请号 202410915441.8

(22)申请日 2024.07.09

(71)申请人 上海弘连网络科技有限公司
地址 201100 上海市闵行区中春路7001号7幢3楼

(72)发明人 向瑜强 陈晖 刘海飞 王凯明 张雨 王世和 张虎 林港 黄锐峰 翟泽雨 郭淑香 黄浩楠

(74)专利代理机构 上海光华专利事务所(普通合伙) 31219
专利代理师 徐秋平

(51)Int.Cl.
G06F 21/56(2013.01)

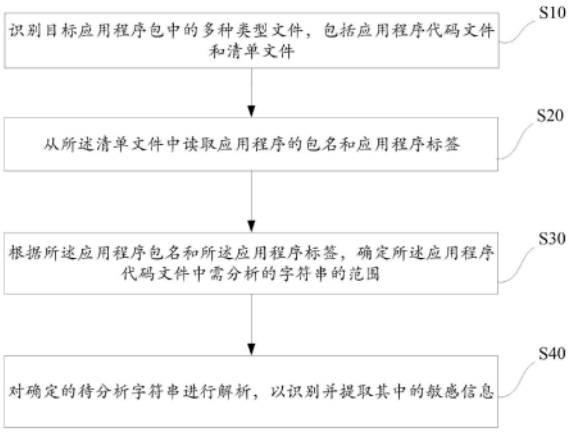
权利要求书2页 说明书12页 附图5页

(54)发明名称

一种提取应用程序包中敏感信息的方法、装置、电子设备及介质

(57)摘要

本申请提供一种提取应用程序包中敏感信息的方法、装置、电子设备及介质,所述方法包括:识别目标应用程序包中的多种类型文件,包括应用程序代码文件和清单文件;从所述清单文件中读取应用程序的包名和应用程序标签;根据所述应用程序包名和所述应用程序标签,确定所述应用程序代码文件中需分析的字符串的范围;对确定的待分析字符串进行解析,以识别并提取其中的敏感信息。能够高效地从应用程序包中提取关键的敏感信息,极大地提高了分析的效率和响应速度。通过综合分析应用程序包中的多种文件类型和特征,能够提供准确的敏感信息提取结果,减少误判和漏判的情况。



CN 118886008 A

1.一种提取应用程序包中敏感信息的方法,其特征在于,包括:

识别目标应用程序包中的多种类型文件,包括应用程序代码文件和清单文件;

从所述清单文件中读取应用程序的包名和应用程序标签;

根据所述应用程序包名和所述应用程序标签,确定所述应用程序代码文件中需分析的字符串的范围;

对确定的待分析字符串进行解析,以识别并提取其中的敏感信息。

2.根据权利要求1所述的提取应用程序包中敏感信息的方法,其特征在于,所述敏感信息包括API密钥、IP地址、URL链接、邮箱地址、用户凭证、加密数据、系统命令、哈希值中的一种或多种组合。

3.根据权利要求1所述的提取应用程序包中敏感信息的方法,其特征在于,所述对确定的待分析字符串进行解析,以识别并提取其中的敏感信息,包括:

读取所述待分析字符串的类名称;

基于所述类名称判断所述待分析字符串是否为可疑字符串;

若是,则解析所述应用程序代码文件中定义所述可疑字符串的具体指令,以确定所述可疑字符串的用途;

根据所述可疑字符串的用途,判断可疑字符串是否为敏感字符串;

若是,则提取所述敏感字符串中的敏感信息。

4.根据权利要求3所述的提取应用程序包中敏感信息的方法,其特征在于,所述根据所述可疑字符串的用途,判断可疑字符串是否为敏感字符串,包括:

将所述可疑字符串与预定义的敏感信息类型进行格式校验;

基于校验结果确定所述可疑字符串是否为敏感字符串。

5.根据权利要求4所述的提取应用程序包中敏感信息的方法,其特征在于,所述基于校验结果确定所述可疑字符串是否为敏感字符串,包括:

若所述格式校验通过,则判定所述可疑字符串为敏感字符串;

记录所述敏感字符串的内容以及对应的敏感信息类型。

6.根据权利要求4所述的提取应用程序包中敏感信息的方法,其特征在于,所述基于校验结果确定所述可疑字符串是否为敏感字符串,包括:

若所述格式校验未通过,则判断所述可疑字符串是否被加密;

若是,则将所述可疑字符串进行解密,并再次进行格式校验;

若解密后的格式校验通过,则判定所述可疑字符串为敏感字符串,并记录所述敏感字符串的内容、使用的加密算法以及敏感信息类型。

7.根据权利要求3所述的提取应用程序包中敏感信息的方法,其特征在于,所述提取所述敏感字符串中的敏感信息,包括:

确定所述敏感字符串被赋值的变量名称;

基于所述变量名称提取所述敏感字符串中的敏感信息。

8.根据权利要求7所述的提取应用程序包中敏感信息的方法,其特征在于,所述基于所述变量名称提取所述敏感字符串中的敏感信息,包括:

对所述变量名称进行分词处理;

基于分词处理后的变量名称提取所述敏感字符串中的敏感信息。

9.一种提取应用程序包中敏感信息的装置,其特征在于,包括:

识别模块,用于识别目标应用程序包中的多种类型文件,包括应用程序代码文件和清单文件;

读取模块,用于从所述清单文件中读取应用程序的包名和应用程序标签;

范围确定模块,用于根据所述应用程序包名和所述应用程序标签,确定所述应用程序代码文件中需分析的字符串的范围;

提取模块,用于对确定的待分析字符串进行解析,以识别并提取其中的敏感信息。

10.一种电子设备,其特征在于,所述电子设备包括:

处理器和存储器;

其中,所述存储器用于存储计算机程序;

所述处理器用于执行所述存储器存储的计算机程序,以使所述电子设备执行权利要求1至8任一项所述的提取应用程序包中敏感信息方法。

11.一种计算机可读存储介质,其上存储有计算机程序,其特征在于,该程序被电子设备执行时实现权利要求1至8任一项所述的提取应用程序包中敏感信息方法。

一种提取应用程序包中敏感信息的方法、装置、电子设备及介质

技术领域

[0001] 本公开信息网络安全的技术领域,尤其涉及一种提取应用程序包中敏感信息的方法、装置、电子设备及介质。

背景技术

[0002] 在当今社会,移动设备的广泛普及和应用程序的蓬勃发展极大地便利了人们的日常生活,然而也带来了新型犯罪活动的滋生。这些犯罪活动不再局限于传统的线下模式,而是利用应用程序中的漏洞和用户信息的泄露进行了一系列高技术含量的非法行为,包括但不限于网络诈骗、信息盗窃、网络钓鱼等。由于这类犯罪的特殊性,它们往往涉及复杂的技术手段和加密方法,使得案件的侦破难度大幅提升。

[0003] 目前,针对此类犯罪的侦查工作面临重大挑战。一方面,犯罪活动具有很强的隐蔽性,并且犯罪分子通常具备一定的技术背景,他们利用匿名网络、加密通讯等手段来隐藏身份和位置,使得追踪和取证变得异常困难。另一方面,犯罪程序往往设计有自毁机制或在短时间后变得不可访问,导致关键证据的迅速丢失。对于基层民警来说,他们通常缺乏必要的专业技术训练和经验积累,面对复杂多变的网络犯罪形式,难以在短时间内有效地识别、分析和处理相关敏感信息。大量案件因此积压并集中到少数专业人员手中,加重了他们的工作负担,降低了整体办案效率。

发明内容

[0004] 鉴于以上所述现有技术的缺点,本公开的目的在于提供一种提取应用程序包中敏感信息的方法、装置、电子设备及介质,旨在提升应用程序包中敏感信息的识别速度和准确率,帮助分析人员节省时间和精力。

[0005] 本公开第一方面提供一种提取应用程序包中敏感信息的方法,包括:识别目标应用程序包中的多种类型文件,包括应用程序代码文件和清单文件;从所述清单文件中读取应用程序的包名和应用程序标签;根据所述应用程序包名和所述应用程序标签,确定所述应用程序代码文件中需分析的字符串的范围;对确定的待分析字符串进行解析,以识别并提取其中的敏感信息。

[0006] 在第一方面的实施例中,所述敏感信息包括API密钥、IP地址、URL链接、邮箱地址、用户凭证、加密数据、系统命令、哈希值中的一种或多种组合。

[0007] 在第一方面的实施例中,所述对确定的待分析字符串进行解析,以识别并提取其中的敏感信息,包括:读取所述待分析字符串的类名称;基于所述类名称判断所述待分析字符串是否为可疑字符串;若是,则解析所述应用程序代码文件中定义所述可疑字符串的具体指令,以确定所述可疑字符串的用途;根据所述可疑字符串的用途,判断可疑字符串是否为敏感字符串;若是,则提取所述敏感字符串中的敏感信息。

[0008] 在第一方面的实施例中,所述根据所述可疑字符串的用途,判断可疑字符串是否

为敏感字符串,包括:将所述可疑字符串与预定义的敏感信息类型进行格式校验;基于校验结果确定所述可疑字符串是否为敏感字符串。

[0009] 在第一方面的实施例中,所述基于校验结果确定所述可疑字符串是否为敏感字符串,包括:若所述格式校验通过,则判定所述可疑字符串为敏感字符串;记录所述敏感字符串的内容以及对应的敏感信息类型。

[0010] 在第一方面的实施例中,所述基于校验结果确定所述可疑字符串是否为敏感字符串,包括:若所述格式校验未通过,则判断所述可疑字符串是否被加密;若是,则将所述可疑字符串进行解密,并再次进行格式校验;若解密后的格式校验通过,则判定所述可疑字符串为敏感字符串,并记录所述敏感字符串的内容、使用的加密算法以及敏感信息类型。

[0011] 在第一方面的实施例中,所述提取所述敏感字符串中的敏感信息,包括:确定所述敏感字符串被赋值的变量名称;基于所述变量名称提取所述敏感字符串中的敏感信息。

[0012] 在第一方面的实施例中,所述基于所述变量名称提取所述敏感字符串中的敏感信息,包括:对所述变量名称进行分词处理;基于分词处理后的变量名称提取所述敏感字符串中的敏感信息。

[0013] 本公开第二方面公开一种提取应用程序包中敏感信息的装置,包括:识别模块,用于识别目标应用程序包中的多种类型文件,包括应用程序代码文件和清单文件;读取模块,用于从所述清单文件中读取应用程序的包名和应用程序标签;范围确定模块,用于根据所述应用程序包名和所述应用程序标签,确定所述应用程序代码文件中需分析的字符串的范围;提取模块,用于对确定的待分析字符串进行解析,以识别并提取其中的敏感信息。

[0014] 本公开第三方面公开一种电子设备,所述电子设备包括:处理器和存储器;其中,所述存储器用于存储计算机程序;所述处理器用于执行所述存储器存储的计算机程序,以使所述电子设备执行如第一方面任一项所述的提取应用程序包中敏感信息方法。

[0015] 本公开第四方面公开一种计算机可读存储介质,其上存储有计算机程序,该程序被电子设备执行时实现第一方面任一项所述的提取应用程序包中敏感信息方法。

[0016] 如上所述,本公开提供的一种提取应用程序包中敏感信息的方法、装置、电子设备及介质,至少包括以下技术效果:

[0017] (1)能够高效地从应用程序包中提取关键的敏感信息,如API密钥、IP地址、URL链接等,极大地提高了分析的效率和响应速度。

[0018] (2)通过综合分析应用程序包中的多种文件类型和特征,能够提供准确的敏感信息提取结果,减少误判和漏判的情况。

[0019] (3)简化了敏感信息提取的流程,降低了工作人员在技术和经验方面的要求,使得非专业人士也能够迅速上手并有效执行敏感信息提取任务。

[0020] (4)能够快速识别出带有敏感信息的应用程序,帮助分析人员迅速识别并打击利用移动应用程序进行的犯罪活动,提高了打击犯罪的效率和准确性。

附图说明

[0021] 图1展示本公开一实施例中提取应用程序包中敏感信息的方法的流程示意图。

[0022] 图2展示本公开一实施例中解析字符串并提取敏感信息的流程示意图。

[0023] 图3展示本公开一实施例中判定敏感字符串的流程示意图。

[0024] 图4展示本公开另一实施例中判定敏感字符串的流程示意图。

[0025] 图5展示本公开另一实施例中提取应用程序包中敏感信息的方法的流程示意图。

[0026] 图6展示本公开一具体示例的原理图。

[0027] 图7展示本公开一实施例中提取应用程序包中敏感信息的装置模块示意图。

[0028] 图8展示本公开一实施例中电子设备的电路结构示意图。

具体实施方式

[0029] 以下通过特定的具体实例说明本发明的实施方式,本领域技术人员可由本说明书所揭露的内容轻易地了解本发明的其他优点与功效。本发明还可以通过另外不同的具体实施方式加以实施或应用,本说明书中的各项细节也可以基于不同观点与应用,在没有背离本发明的精神下进行各种修饰或改变。需说明的是,在不冲突的情况下,以下实施例及实施例中的特征可以相互组合。

[0030] 需要说明的是,以下实施例中所提供的图示仅以示意方式说明本发明的基本构想,遂图式中仅显示与本发明中有关的组件而非按照实际实施时的组件数目、形状及尺寸绘制,其实际实施时各组件的型态、数量及比例可为一种随意的改变,且其组件布局型态也可能更为复杂。

[0031] 在移动互联网蓬勃发展的今天,移动应用程序已成为我们日常生活的重要组成部分,但同时也带来了新的安全挑战。犯罪分子越来越多地利用这些应用程序进行非法活动,如网络诈骗、数据盗窃等,严重威胁用户隐私和财产安全。在现有的技术体系中,安全防御措施主要集中在利用智能终端中的数据进行事后分析,当可疑行为发生后,安全人员通过对智能终端中的数据进行收集和分析,以识别和追踪犯罪嫌疑人。然而,这种反应性策略存在显著缺陷:它只能在犯罪行为发生后采取行动,导致反应延迟和犯罪后果已然产生。此外,犯罪证据的收集和固定变得更加困难,且需消耗大量资源,效率较低。更重要的是,现有方法可能侵犯用户隐私,并难以应对使用高科技手段的犯罪分子。

[0032] 本公开旨在将犯罪扼杀在萌芽阶段,即通过实时监测和分析移动应用程序中的敏感信息,主动识别并打击用于犯罪的应用程序。此方法能减少犯罪案件的发生,提升社会整体安全。与现有技术相比,本公开的预防性措施可实现早期发现和快速干预,有效弥补了事后处理的不足。

[0033] 下面将结合本发明实施例中的附图,对本发明实施例中的技术方案进行详细描述。

[0034] 如图1所示,展示本公开第一方面一实施例中提取应用程序包中敏感信息方法的流程示意图,包括步骤S10-S40,其中,

[0035] 步骤S10:识别目标应用程序包中的多种类型文件,包括应用程序代码文件和清单文件。

[0036] 具体的,在提取应用程序包(APK文件)中的敏感信息之前,首先需要对APK文件进行解压缩,以便访问其中的文件。解压缩过程可以通过Java的ZipInputStream类来完成,该类能够读取APK文件并将其内容解压到一个临时目录中。在解压后的目录中,可以遍历所有文件并检查它们的扩展名或文件头来确定它们是否为特定类型的文件。例如,可以通过检查文件扩展名(如.dex、.html、.js、.json等)或文件头(如特定的魔数)来识别不同类型的

文件。

[0037] 对于应用程序代码文件,即包含可执行代码的文件,如DEX文件(Android应用程序的字节码),需要进行代码还原操作。代码还原是将编译后的代码转换回源代码的过程。这通常涉及反编译工具的使用,如dex2jar和jd-gui,可以将DEX文件转换为JAR文件,然后使用反编译器将其转换为Java源代码。通过还原源代码,可以更清楚地了解应用程序的逻辑和功能,从而更容易地识别和提取敏感信息。

[0038] 对于包含脚本的文件,如HTML、JavaScript和JSON文件,可以直接查看其内容以确定其结构和功能。这些文件可能包含应用程序的用户界面元素、交互逻辑以及数据交换格式等信息。通过分析这些文件,可以了解应用程序的交互方式和数据处理流程,从而识别和提取与用户数据相关的敏感信息。

[0039] 除了代码和脚本文件外,APK文件还可能包含其他类型的资源文件,如图像、音频、视频等。这些资源文件通常位于res目录下,可以通过分析清单文件(AndroidManifest.xml)来确定它们的位置和用途。清单文件是APK文件中最重要的文件之一,它包含了应用程序的基本信息和权限声明。通过解析清单文件,可以获取应用程序的名称、版本号、所需的权限等信息。此外,还可以从中提取出应用程序的主要组件(如活动、服务、广播接收器等)及其配置信息。这些信息有助于理解应用程序的整体结构和功能,从而更好地识别和提取敏感信息。

[0040] 在一些实施方式中,所述敏感信息包括API密钥、IP地址、URL链接、邮箱地址、用户凭证、加密数据、系统命令、哈希值中的一种或多种组合。

[0041] 具体的,在应用程序包中,敏感信息是指那些可能被恶意攻击者利用以获取不当利益或破坏系统安全的信息。以下是一些常见的敏感信息类型及其详细描述:

[0042] API密钥:API密钥是用于访问特定服务的凭证,通常由服务提供商提供。它们可以用于访问云服务、第三方库、数据库等。泄露API密钥可能导致未经授权的数据访问、滥用服务资源或执行恶意操作。在应用程序包中,API密钥可能以明文或加密的形式存储,需要仔细分析以确定其安全性和潜在风险。

[0043] IP地址:IP地址是分配给网络设备的标识符,用于在网络上进行通信。如果应用程序包中包含IP地址,可能会暴露服务器的位置和网络拓扑结构,从而增加被攻击的风险。IP地址的泄露可能导致网络攻击、数据泄露或其他安全问题。

[0044] URL链接:URL链接指向网络上的资源,如网页、文件或API端点。如果应用程序包中包含URL链接,攻击者可以利用这些链接来访问敏感数据或执行恶意操作。URL链接的泄露可能导致数据泄露、恶意软件传播或其他安全问题。

[0045] 邮箱地址:邮箱地址是用户的个人联系信息之一,通常用于接收通知、验证身份等。泄露邮箱地址可能导致垃圾邮件、钓鱼攻击或其他形式的欺诈行为。邮箱地址的泄露可能导致隐私泄露、账户劫持或其他安全问题。

[0046] 用户凭证:用户凭证包括用户名、密码、令牌、会话ID等,用于验证用户身份并授权访问特定资源。泄露用户凭证可能导致未经授权的访问、账户劫持或其他安全问题。在应用程序包中,用户凭证可能以明文或加密的形式存储,需要仔细分析以确定其安全性和潜在风险。

[0047] 加密数据:加密数据是指使用加密算法保护的数据。如果应用程序包中包含加密

数据的密钥或解密逻辑,攻击者可能能够解密数据并访问其中的信息。加密数据的泄露可能导致数据泄露、信息泄露或其他安全问题。

[0048] 系统命令:系统命令是操作系统提供的一组功能,用于执行特定的任务。如果应用程序包中包含系统命令,攻击者可能能够利用这些命令执行恶意操作,如删除文件、修改系统设置等。系统命令的泄露可能导致系统破坏、数据丢失或其他安全问题。

[0049] 哈希值:哈希值(Hash Value)是一种通过特定算法将任意长度的数据转换为固定长度唯一标识符的过程。这个过程是不可逆的,即从哈希值无法直接还原原始数据。然而,如果应用程序包中包含哈希值,并且这些哈希值是基于敏感信息(如密码、密钥等)生成的,攻击者可能会试图通过各种手段破解这些哈希值,以还原原始数据。哈希值在应用程序中通常用于存储用户密码、API密钥、敏感配置信息等,以防止直接暴露这些敏感数据。然而,如果哈希值被攻击者获取,他们可能会使用哈希碰撞攻击、彩虹表攻击或暴力破解等方法来尝试破解哈希值,从而获取原始敏感信息。

[0050] 步骤S20:从所述清单文件中读取应用程序的包名和应用程序标签。

[0051] 具体的,清单文件(AndroidManifest.xml)是Android应用程序的核心配置文件,它包含了应用程序的元数据、权限声明、组件定义以及应用程序的行为规则。在解析清单文件时,可以利用Java的XML解析器来读取XML文件中的节点和属性,并将其转换为可操作的数据结构。

[0052] 为了解析XML文件,可以使用Java提供的XML解析器,如SAX(SimpleAPI for XML)、DOM(Document Object Model)或StAX(StreamingAPI for XML)。这些解析器能够读取XML文件中的节点和属性,并将其转换为Java对象或数据结构,以便进行进一步的操作和分析。

[0053] 在解析清单文件时,需要查找特定的节点和属性来提取应用程序的包名和应用程序标签。应用程序包名是应用程序的唯一标识符,通常由开发者在创建应用程序时指定。例如,一个典型的Android应用程序包名可能是"com.example.myapp"。应用程序包名通常位于节点的package属性中。例如,在节点中,包名定义为"package=“com.example.myapp”"。应用程序标签通常用于描述应用程序的功能或用途,位于节点的android:label属性中。例如,一个电子邮件应用可能有一个标签如"Email",而一个社交媒体应用可能有一个标签如"Social Networking"。

[0054] 使用XML解析器解析清单文件,并定位到节点和节点,从节点的package属性中提取应用程序的包名,从节点的android:label属性中提取应用程序的标签。

[0055] 根据提取的应用程序包名和应用程序标签,可以确定需分析的字符串的范围。例如,如果应用程序的标签表明它是一个社交应用,可以更关注与社交功能相关的代码部分。

[0056] 结合其他分析技术,如代码还原、脚本解析等,以全面分析应用程序中的敏感信息。例如,可以使用反编译工具将DEX文件转换为JAR文件,然后使用反编译器将其转换为Java源代码,以便更清晰地了解应用程序的逻辑和功能。

[0057] 步骤S30:根据所述应用程序包名和所述应用程序标签,确定所述应用程序代码文件中需分析的字符串的范围。

[0058] 具体的,在提取应用程序包中的敏感信息时,使用文件系统搜索功能是一个关键步骤。这一步骤的目的是找到与给定应用程序包名相关的所有文件,这些文件可能包括源代码文件、资源文件等。可以使用命令行工具(如find命令)或者编程语言中的文件操作函

数来实现这一步骤。

[0059] 使用文件系统搜索功能,可以查找与给定应用程序包名相关的所有文件。这些文件可能包含应用程序的源代码、配置文件、资源文件等。文件系统搜索功能可以通过命令行工具(如find命令)或者编程语言中的文件操作函数来实现。

[0060] 在找到相关文件后,可以根据应用程序标签进一步筛选出需要分析的文件。例如,如果应用程序标签是"Email",则可以选择包含"email"关键字的文件进行进一步分析。筛选过程可以通过文本搜索算法或正则表达式来实现,以提高搜索的准确性和效率。

[0061] 在筛选出的文件中,找到特定的字符串范围,这些字符串范围可能包括变量名、方法名、注释等。根据应用程序标签来确定需要关注的具体字符串类型。例如,如果应用程序标签是"Email",则可能需要关注与电子邮件相关的字符串,如邮件地址、主题等。

[0062] 对找到的字符串范围进行深入分析,以识别和提取其中的敏感信息。例如,可以检查变量名中是否包含敏感关键词,如"password"、"apiKey"等。还可以检查方法名和注释中是否包含敏感信息,如"getEmail"、"sendEmail"等。

[0063] 上述实施方式中,通过读取应用程序清单文件,快速识别出与敏感信息可能相关的类和资源。这一步骤利用了应用程序的结构信息,避免了对整个应用程序包的逐一检查,大大提高了初始筛选的速度。

[0064] 步骤S40:对确定的待分析字符串进行解析,以识别并提取其中的敏感信息。

[0065] 具体的,确定字符串范围后,系统会对这些范围内的字符串进行深入分析。系统会解析应用程序代码文件中的字符串,包括它们的赋值和使用情况。通过分析字符串的赋值和使用情况,可以确定它们是否包含敏感信息,如API密钥、用户凭证等。例如,如果一个字符串被赋值给一个名为"apiKey"的变量,并且该变量在应用程序的登录过程中被使用,那么这个字符串很可能是API密钥。

[0066] 通过深入分析字符串的赋值和使用情况,系统可以更准确地识别和提取敏感信息。例如,如果一个字符串被用于网络请求的URL或作为文件名,这些使用情况可以进一步确认其敏感性。通过对字符串的赋值和使用情况进行深入分析,系统可以提高识别和提取敏感信息的准确性。例如,通过分析字符串的赋值和使用情况,系统可以更准确地识别和提取API密钥、用户凭证等敏感信息。

[0067] 在一些实施方式中,如图2所示,所述步骤S40包括步骤S41-S45,其中,

[0068] 步骤S41:读取所述待分析字符串的类名称。

[0069] 具体的,在所述应用程序代码文件中,每个字符串通常与一个特定的类相关联。因此,首先需要读取待分析字符串所属的类名称。这可以通过解析代码文件的结构来实现,例如使用编程语言中的反射机制或解析器库来获取字符串所在的类名。

[0070] 为了确定字符串所属的类名称,系统需要解析代码文件的结构。代码文件通常由编程语言的编译器或解释器生成,其中包含类、方法、变量等元素。系统可以使用编程语言提供的反射机制或解析器库来解析代码文件的结构。

[0071] 反射机制是编程语言提供的一种机制,允许程序在运行时检查和修改其本身的结构。解析器库是一组函数或类,用于解析特定类型的文件或数据格式。通过使用反射机制或解析器库,系统可以获取字符串所在的类名。

[0072] 在解析代码文件的结构后,系统可以读取待分析字符串的类名称。例如,如果待分

析字符串出现在名为"MyClass"的类的某个方法中,系统可以读取该字符串所属的类名"MyClass"。

[0073] 步骤S42:基于所述类名称判断所述待分析字符串是否为可疑字符串。

[0074] 具体的,可以使用一系列预定义的模式或规则来检查类名称,以确定它是否包含潜在的敏感信息。这些模式或规则可以包括特定的关键字、正则表达式或其他匹配逻辑。例如,可以创建一个包含敏感词汇的列表,如"password"、"secret"等。然后,将待分析的类名称与这个列表中的每个关键词进行比较。如果类名称中包含任何敏感词汇,就可以将其标记为可疑字符串。

[0075] 另一种方法是使用正则表达式来定义更复杂的模式。例如,可以编写一个正则表达式来查找包含数字和字母组合的字符串,这可能暗示着密码或密钥的存在。通过应用这些模式,可以更准确地识别出可疑字符串。

[0076] 此外,还可以考虑其他因素来确定类名称是否可疑。例如,可以检查类名称的长度、字符分布或其他特征,以进一步确认其是否具有潜在的敏感信息。

[0077] 一旦确定了类名称是可疑的,可以执行步骤S43。

[0078] 步骤S43:解析所述应用程序代码文件中定义所述可疑字符串的具体指令,以确定所述可疑字符串的用途。

[0079] 具体的,可以使用编程语言的解析器库来分析代码文件并提取相关信息。通过分析这些指令,可以了解可疑字符串在应用程序中的用途和上下文。

[0080] 首先,需要选择一个适合目标编程语言的解析器库。这些库通常提供对源代码进行语法分析和词法分析的功能,使能够准确地识别和处理代码中的不同元素。例如,对于Java语言,可以使用ANTLR或JavaParser等库;对于Python语言,可以使用Python自带的ast模块等。

[0081] 一旦选择了适当的解析器库,可以将其应用于代码文件,以便对其进行解析。解析过程将生成一个抽象语法树(AST),其中包含了代码文件中的所有元素及其关系。通过遍历AST,可以访问每个节点并提取有关可疑字符串的信息。

[0082] 在遍历AST时,需要关注与可疑字符串相关的指令。这可能包括变量声明、赋值语句、函数调用等。可以检查这些指令是否直接引用了可疑字符串,或者是否在上下文中使用了可疑字符串。此外,还需要考虑其他因素,如可疑字符串是否被用作参数传递给函数、是否与其他敏感信息相关联等。

[0083] 通过分析这些指令,可以了解可疑字符串在应用程序中的用途和上下文。例如,如果可疑字符串出现在一个密码验证函数中,可以推断它可能是用于存储或验证用户密码的变量。如果可疑字符串被用作API密钥的一部分,可以推断它可能是用于访问外部服务的凭证。

[0084] 需要注意的是,解析代码文件可能会涉及到复杂的语法结构和嵌套结构,因此可能需要一定的编程知识和经验来正确解析和理解代码。此外,不同的编程语言可能具有不同的语法规则和特性,因此在实际应用中需要根据具体的编程语言选择合适的解析器库。

[0085] 步骤S44:根据所述可疑字符串的用途,判断可疑字符串是否为敏感字符串。若是,则执行步骤S45。

[0086] 具体的,根据可疑字符串的用途来判断它是否为敏感字符串。具体来说,可以比较

可疑字符串的用途与已知的敏感信息模式或规则。

[0087] 首先,需要定义一组敏感信息的模式或规则。这些模式可以包括特定的关键字、正则表达式或其他匹配逻辑,用于识别可能包含敏感信息的字符串。例如,可以创建一个列表,其中包含诸如"password"、"secret"、"api_key"等敏感词汇。然后,将可疑字符串的用途与这些模式进行比较。

[0088] 如果可疑字符串的用途与任何敏感信息模式相匹配,那么可以将其标记为敏感字符串。这可以通过设置一个布尔变量或使用其他标识符来实现。一旦确定可疑字符串是敏感的,可以执行步骤S45,即采取适当的措施来处理该字符串。

[0089] 需要注意的是,敏感信息的定义可能会因应用程序和上下文而有所不同。因此,在实际实施过程中,可能需要进一步优化和调整这些模式和方法,以确保最佳的检测效果和安全性。

[0090] 示例性的,如图3所示,所述步骤S44包括步骤S441-S442,其中,

[0091] 步骤S441:将所述可疑字符串与预定义的敏感信息类型进行格式校验。

[0092] 具体的,需要定义一组敏感信息类型的格式规则。这些规则可以包括电子邮件地址、电话号码、信用卡号码等常见敏感信息的格式模式。例如,对于电子邮件地址,可以使用正则表达式来匹配常见的电子邮件格式;对于电话号码,我们可以使用正则表达式来匹配国际或本地电话号码的格式。

[0093] 然后,将可疑字符串与这些格式规则进行比较。如果可疑字符串符合任何一个敏感信息类型的格式,那么可以将其标记为敏感字符串。这可以通过设置一个布尔变量或使用其他标识符来实现。

[0094] 步骤S442:基于校验结果确定所述可疑字符串是否为敏感字符串。

[0095] 具体的,如果可疑字符串通过了格式校验,即符合预定义的敏感信息类型的格式要求,那么可以将其标记为敏感字符串。否则,可以将其视为非敏感字符串。

[0096] 例如,如果在步骤S441中使用了正则表达式来匹配电子邮件地址的格式,并且可疑字符串通过了这个校验,那么可以将该字符串标记为敏感字符串。同样地,如果使用了其他文本匹配技术来检查电话号码或信用卡号码等格式,并且可疑字符串符合这些格式要求,也可以将它们标记为敏感字符串。

[0097] 需要注意的是,即使可疑字符串通过了格式校验,仍然需要进一步的分析来确定其是否真正包含敏感信息。例如,虽然一个字符串可能符合电子邮件地址的格式,但它可能只是一个普通的公共邮箱地址,而不是一个实际的用户邮箱地址。因此,在进行格式校验之后,还需要进一步分析可疑字符串的内容和上下文,以确定其是否真正包含敏感信息。

[0098] 进一步的,如图4所示,所述步骤S442包括步骤a)-d),其中,

[0099] 步骤a):判断所述格式校验是否通过,若是,则进入步骤b);若否,则进入步骤c)。

[0100] 具体的,在进行格式校验时,是基于字符串的结构特征与预定模式的匹配结果,其中预定模式包括但不限于正则表达式、长度要求、字符组合规则等。

[0101] 步骤b):判定所述可疑字符串为敏感字符串,并记录所述敏感字符串的内容以及对应的敏感信息类型。

[0102] 具体的,如果可疑字符串符合预定义的敏感信息类型(如信用卡号、社会保障号等)的格式要求,则该字符串被明确判定为敏感字符串。一旦确定了字符串为敏感字符串,

应记录其内容以及它所对应的敏感信息类型。这一步骤可以通过将敏感字符串及其类型存储在安全的数据管理系统中来实现。记录的信息应包括敏感字符串的值、检测到的敏感信息类型(如“信用卡号”、“社会保障号”等)、检测时间、相关代码文件的名称和位置等。这一信息将被用于后续的安全审计、报告或采取补救措施,确保所有敏感信息的使用、存储和传输都符合相关的隐私法规和安全标准。

[0103] 上述实施方式中,根据预先定义的规则和模式,如特定关键字或格式,对潜在的敏感信息进行针对性分析。这种方法减少了无关数据的干扰,提高了分析的针对性和效率。

[0104] 步骤c):判断所述可疑字符串是否被加密。

[0105] 具体的,对于未通过格式校验的可疑字符串,需要进一步判断其是否被加密。这可以通过检查字符串是否符合已知的加密格式(如特定字符分布、加密标识符等)或通过分析代码上下文(如是否存在加密/解密函数调用)来确定。若是,则执行步骤d)。

[0106] 步骤d):将所述可疑字符串进行解密,并返回步骤a)进行格式校验。

[0107] 具体的,如果确定可疑字符串被加密,需要将其解密以获取原始数据。这通常涉及使用应用程序中使用的相同解密算法,该算法可以是对称加密算法、非对称加密算法或任何其他适当的加密方法。解密过程应在安全的环境中进行,以确保敏感数据不会被泄露。解密后,应再次进行格式校验以确定其是否为敏感字符串。

[0108] 若解密后的格式校验通过,则返回步骤b)。

[0109] 具体的,如果解密后的字符串通过了格式校验,它应被判定为敏感字符串。同时,应记录其内容、所使用的加密算法以及对应的敏感信息类型。

[0110] 上述实施方式中,对于加密的敏感信息,本申请提供了解密和再次校验的步骤,确保了即使是经过加密处理的敏感信息也能被准确提取和识别。

[0111] 步骤S45:提取所述敏感字符串中的敏感信息。

[0112] 具体的,在确定了字符串为敏感字符串后,下一步是提取其中的敏感信息。这可能涉及从字符串中解析出具体的敏感数据,如信用卡号、密码等。提取的过程应确保遵守适当的数据保护和隐私标准。

[0113] 在一些实施方式中,所述步骤S45包括:确定所述敏感字符串被赋值的变量名称;基于所述变量名称提取所述敏感字符串中的敏感信息。

[0114] 具体的,在代码审查过程中,首先需要确定敏感字符串被赋值给了哪个变量。这通常涉及对代码的分析,以识别出哪些变量接受了敏感字符串的值。这些变量的名称可能提供了关于敏感数据用途的直接线索,有助于更准确地识别和分类敏感信息。利用已确定的变量名称,可以更精确地从代码中提取与该变量相关联的敏感信息。例如,如果一个变量名为userPassword,显然它存储的是用户密码信息,那么任何赋给这个变量的字符串都应被视为敏感信息。

[0115] 为了进一步精细化敏感信息的提取过程,可以对变量名称执行分词处理。这意味着将变量名称分解成更小的语义单元,这有助于更好地理解变量的用途。例如,变量名userCredentials可以分词为user和Credentials,从而更清楚地指示该变量用于存储用户的凭证信息。

[0116] 使用分词处理后的变量名称,可以进一步提高从代码中提取敏感信息的准确度。通过分析分词后的语义单元,可以更精确地判断变量的用途,并据此提取相关的敏感信息。

这有助于减少误报和漏报,确保敏感信息管理系统的效率和准确性。

[0117] 基于上述全部实施例的内容,如图5所示,展示本公开一实施例中提取应用程序包中敏感信息方法完整的流程示意图。

[0118] 为了对本公开介绍的提取应用程序包中敏感信息的方法更好的阐述,下面列举一个具体示例。如图6所示,展示本示例的原理图。

[0119] 在本示例中,假设一个基层民警接到了一起涉嫌利用移动应用程序进行犯罪活动的案件。犯罪分子制作了一个名为“EasyCredentials”的应用程序,该应用程序能盗取用户的密码、银行卡号、API密钥等信息。为了快速锁定犯罪分子,民警使用本技术方案对“EasyCredentials”应用程序进行敏感信息提取。

[0120] 民警首先使用本技术方案识别和分析了“EasyCredentials”应用程序包中的多种类型文件,包括应用程序代码文件和清单文件。从清单文件中,民警提取了应用程序的包名和应用程序标签,即“EasyCredentials”。根据应用程序包名和应用程序标签,民警确定了需分析的字符串的范围。

[0121] 对确定的待分析字符串进行解析,以识别并提取其中的敏感信息。例如,民警发现了一个字符串“password123”,这可能是用户的密码。

[0122] 民警读取了待分析字符串的类名称,并根据类名称判断该字符串是否为可疑字符串。例如,如果字符串出现在与网络通信相关的类中,则可能是可疑字符串。若是,则解析应用程序代码文件中定义该字符串的具体指令,以确定其用途。例如,民警发现该字符串被赋值给了一个名为“userPassword”的变量。

[0123] 根据可疑字符串的用途,民警判断该字符串是否为敏感字符串,并提取其中的敏感信息。例如,如果该字符串被用于登录操作,则很可能是敏感信息。

[0124] 民警将可疑字符串与预定义的敏感信息类型进行格式校验,以确定其是否为敏感字符串。例如,民警检查了字符串是否符合常见的密码格式。

[0125] 若格式校验未通过,则判断该字符串是否被加密。例如,民警发现该字符串被加密了。若是,则将字符串进行解密,并再次进行格式校验。例如,民警解密了字符串,发现其内容是“password123”。

[0126] 若解密后的格式校验通过,则判定该字符串为敏感字符串,并记录其内容、使用的加密算法以及敏感信息类型。例如,民警记录了该字符串是用户的密码,使用了AES加密算法。

[0127] 民警确定敏感字符串被赋值的变量名称,并基于变量名称提取敏感字符串中的敏感信息。例如,民警发现“userPassword”变量存储了用户的密码。

[0128] 对变量名称进行分词处理,以提高敏感词识别的准确性。例如,民警将“userPassword”分词为“user”和“Password”。基于分词处理后的变量名称提取敏感字符串中的敏感信息。例如,民警确认“Password”是用户的密码。

[0129] 通过以上步骤,民警使用本技术方案成功地提取了“EasyCredentials”应用程序中的敏感信息,包括用户密码、银行卡号、API密钥等。

[0130] 需特别说明的是,本公开上述实施例的流程图表示的流程或方法表示可以被理解为,表示包括一个或更一组被配置成实现特定逻辑功能或过程的步骤的可执行指令的代码的模块、片段或部分。并且本公开的优选实施方式的范围包括另外的实现,其中可以不按所

示出或讨论的顺序,包括根据所涉及的功能按基本同时的方式或按相反的顺序,来执行功能。

[0131] 如图7所示,展示本公开提取应用程序包中敏感信息的装置70,需要说明的是所述提取应用程序包中敏感信息的装置的原理、技术实现可以参考之前实施例中的提取应用程序包中敏感信息的方法实施例(例如图1),因此本实施例中不作重复赘述。

[0132] 具体的,所述提取应用程序包中敏感信息的装置70,包括:识别模块71、读取模块72、范围确定模块73、提取模块74,其中,

[0133] 所述识别模块71,用于识别目标应用程序包中的多种类型文件,包括应用程序代码文件和清单文件。

[0134] 所述读取模块72,用于从所述清单文件中读取应用程序的包名和应用程序标签。

[0135] 所述范围确定模块73,用于根据所述应用程序包名和所述应用程序标签,确定所述应用程序代码文件中需分析的字符串的范围。

[0136] 所述提取模块74,用于对确定的待分析字符串进行解析,以识别并提取其中的敏感信息。

[0137] 需特别说明的是,图7实施例中的各个功能模块,可以全部或部分地通过软件、硬件、固件或者其任意组合来实现。当使用软件实现时,可以全部或部分地以程序指令产品的形式实现。程序指令产品包括一个或一组程序指令。在计算机上加载和执行程序指令指令时,全部或部分地产生按照本公开的流程或功能。计算机可以是通用计算机、专用计算机、计算机网络、或者其他可编程装置。程序指令可以存储在计算机可读存储介质中,或者从一个计算机可读存储介质向另一个计算机可读存储介质传输。

[0138] 并且,图7实施例所揭露的装置,可通过其它的模块划分方式实现。以上所表示的装置实施例仅仅是示意性的,例如所述模块的划分,仅仅为一种逻辑功能划分,实际实现时可以有另外的划分方式,例如一组模块或模块可以结合或者可以动态到另一个系统,或一些特征可以忽略,或不执行。另一点,所显示或讨论的相互之间的耦合或直接耦合或通信连接于可以是通过一些接口,装置或模块的间接耦合或通信连接于,可以是电性或其它的形式。

[0139] 另外,图7实施例中的各功能模块及子模块可以动态在一个处理部件中,也可以是各个模块单独物理存在,也可以两个或两个以上模块动态在一个部件中。上述动态的部件既可以采用硬件的形式实现,也可以采用软件功能模块的形式实现。上述动态的部件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,也可以存储在一个计算机可读存储介质中。该存储介质可以是只读存储器,磁盘或光盘等。

[0140] 如图8所示,展示本公开一实施例中电子设备的结构示意图。

[0141] 所述电子设备可以通过运行计算机程序指令,以执行如图1任一项中的方法。示例性地,所述电子设备可以是服务器组/服务器、台式机、笔记本电脑等,也可以是与本地终端远程通信的云端的服务器/服务器组、分布式计算节点系统等。

[0142] 所述电子设备80包括总线81、处理器82、存储器83。处理器82、存储器83之间可以通过总线81通信。所述存储器83中可以存储有程序指令。所述处理器72通过运行存储器83中的程序指令来实现之前实施例中方法步骤,比如图1任一项的方法。

[0143] 总线81可以是外设部件互连标准(Peripheral一氧化碳mponent Inter一氧化碳

nnect,PCI)总线或扩展工业标准结构(Extended Industry StandardArchitecture,EISA)总线等。总线可以分为地址总线、数据总线、控制总线等。为便于表示,虽然图中仅用一条粗线表示,但并不表示仅有一根总线或一种类型的总线。

[0144] 在一些实施例中,处理器82可以为中央处理器(Central Processing Unit,CPU)、微处理单元(MCU)、片上系统(System On Chip)、或现场可编程逻辑阵列(FPGA)等实现。存储器83可以包括易失性存储器(Volatile Memory)以用于运行程序时的数据暂存使用,例如随机存取存储器(RandomAccess Memory,RAM)。

[0145] 存储器83还可以包括非易失性存储器(non-volatile memory)以用于数据存储,例如只读存储器(Read-Only Memory,ROM),快闪存储器,硬盘驱动器(Hard DiskDrive,HDD)或固态盘(Solid-State Disk,SSD)。

[0146] 在一些实施例中,所述电子设备80还可以包括通信器84。所述通信器84用于与外部通信。在具体实例中,所述通信器84可以包括一个或一组有线和/或无线通信电路模块。举例来说,所述通信器84可以包括例如有线网卡、USB模块、串行接口模块等中的一种或多种。无线通信模块所遵循的无线通信协议包括:例如近距离无线通信(Nearfield一氧化碳mmunication,NFC)技术、红外(Infared,IR)技术、全球移动通讯系统(Global System for Mobile一氧化碳mmunications,GSM)、通用分组无线服务(General Packet Radio Service,GPRS)、码分多址引入(一氧化碳de Division MultipleAccess,CDMA)、宽带码分多址(Wideband一氧化碳de division multiple access,WCDMA)、时分码分多址(Time-Division一氧化碳de Division MultipleAccess,TD-SCDMA)、长期演进(Long Term Evolution,LTE)、蓝牙(BlueTooth,BT)、全球导航卫星系统(Global Navigation Satellite System,GNSS)等中的一种或多种。

[0147] 本公开实施例中还可以提供一种计算机可读存储介质,其特征在于,存储有程序指令,所述程序指令被运行执行例如图1实施例中的提取应用程序包中敏感信息的方法。

[0148] 即上述实施例中的方法步骤被实现为可存储在记录介质(诸如CD ROM、RAM、软盘、硬盘或磁光盘)中的软件或计算机代码,或者被实现通过网络下载的原始存储在远程记录介质或非暂时机器可读介质中并将被存储在本地记录介质中的计算机代码,从而在此表示的方法可被存储在使用通用计算机、专用处理器或者可编程或专用硬件(诸如ASIC或FPGA)的记录介质上的这样的软件处理。

[0149] 综上所述,本公开的提取应用程序包中敏感信息的多语言文本的识别方法、装置、电子设备及介质,通过读取应用程序清单文件,快速识别出与敏感信息可能相关的类和资源,利用了应用程序的结构信息,避免了对整个应用程序包的逐一检查,大大提高了初始筛选的速度。另外,根据预先定义的规则和模式,如特定关键字或格式,对潜在的敏感信息进行针对性分析,减少了无关数据的干扰,提高了分析的针对性和效率。并且对于加密的敏感信息,提供了解密和再次校验的步骤,确保了即使是经过加密处理的敏感信息也能被准确提取和识别。

[0150] 上述实施例仅例示性说明本公开的原理及其功效,而非用于限制本公开。任何熟悉此技术的人士皆可在不违背本公开的精神及范畴下,对上述实施例进行修饰或改变。因此,举凡所属技术领域中具有通常知识者在未脱离本公开所揭示的精神与技术思想下所完成的一切等效修饰或改变,仍应由本公开的权利要求所涵盖。

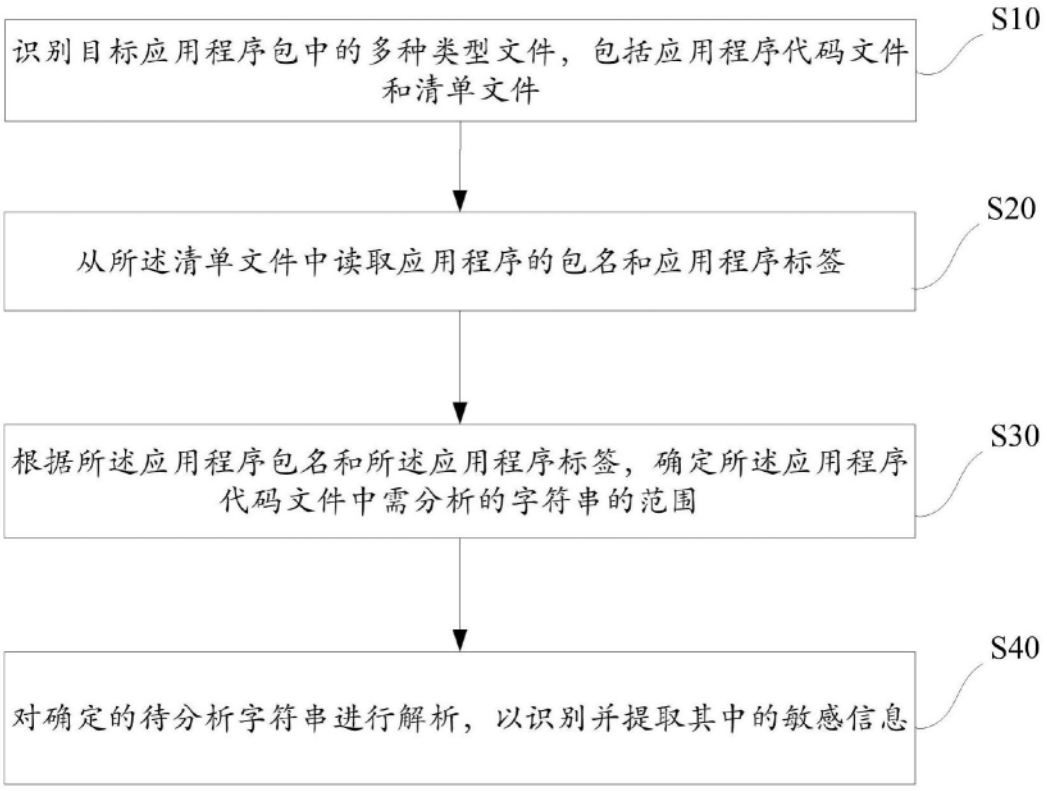

识别目标应用程序包中的多种类型文件，包括应用程序代码文件和清单文件
S10
从所述清单文件中读取应用程序的包名和应用程序标签
S20
根据所述应用程序包名和所述应用程序标签，确定所述应用程序代码文件中需分析的字符串的范围
S30
对确定的待分析字符串进行解析，以识别并提取其中的敏感信息
S40

图1

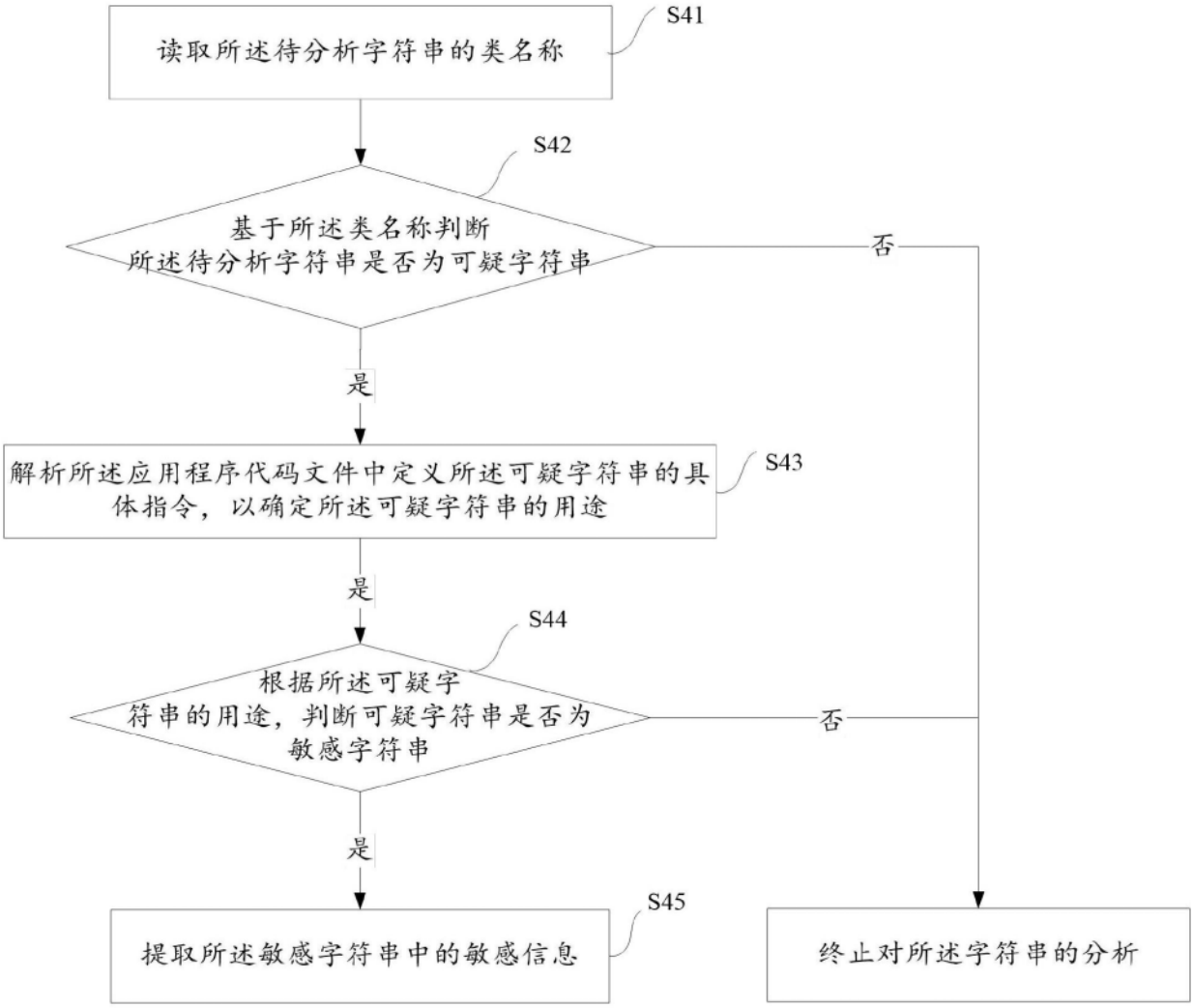

S41
读取所述待分析字符串的类名称
S42
基于所述类名称判断
所述待分析字符串是否为可疑字符串
否
是
解析所述应用程序代码文件中定义所述可疑字符串的具体指令，以确定所述可疑字符串的用途
S43
是
S44
根据所述可疑字
符串的用途，判断可疑字符串是否为
敏感字符串
否
是
S45
提取所述敏感字符串中的敏感信息
终止对所述字符串的分析

图2

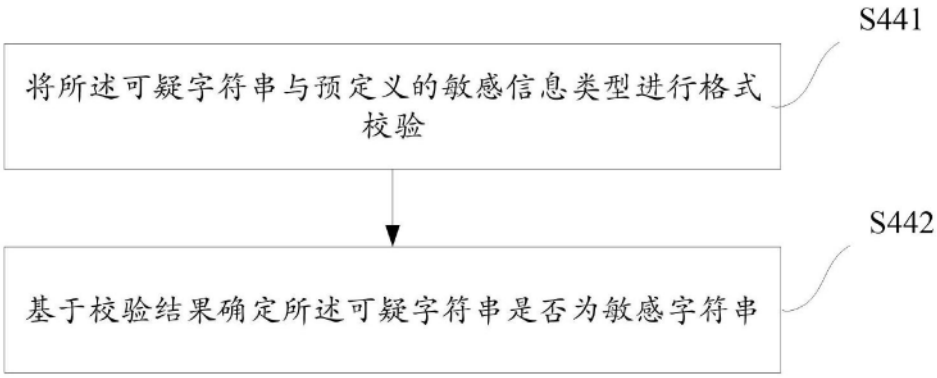

S441
将所述可疑字符串与预定义的敏感信息类型进行格式校验
S442
基于校验结果确定所述可疑字符串是否为敏感字符串

图3

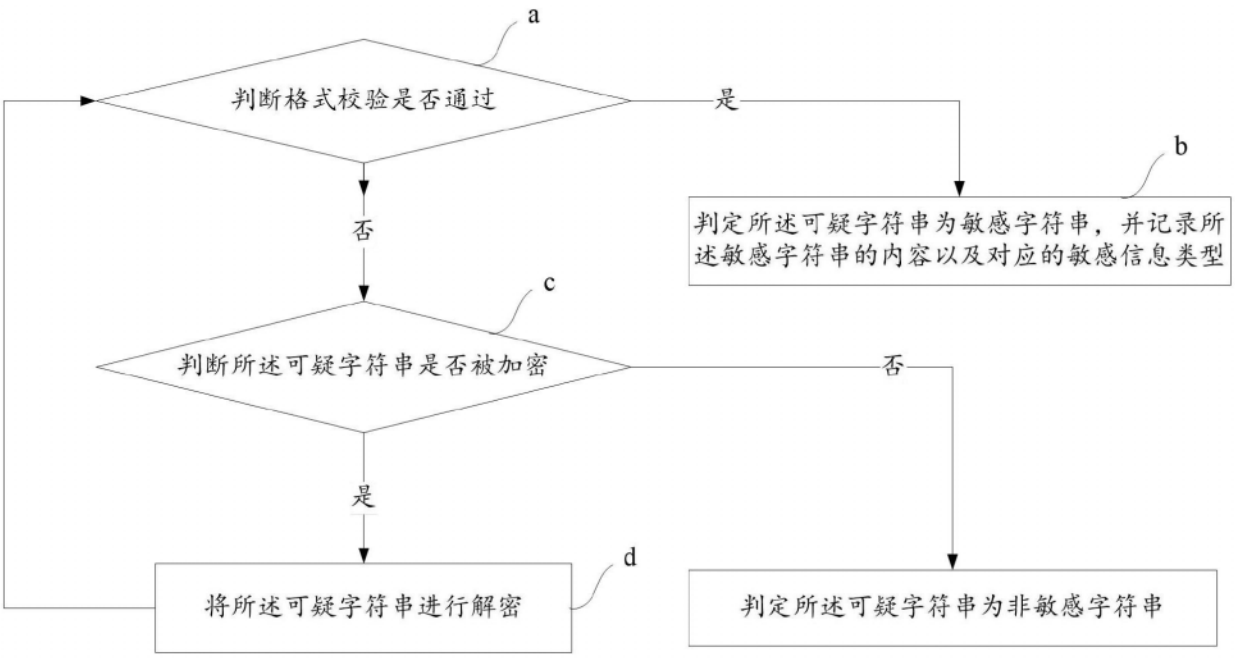

a
判断格式校验是否通过
是
b
判定所述可疑字符串为敏感字符串，并记录所述敏感字符串的内容以及对应的敏感信息类型
否
c
判断所述可疑字符串是否被加密
否
是
d
将所述可疑字符串进行解密
判定所述可疑字符串为非敏感字符串

图4

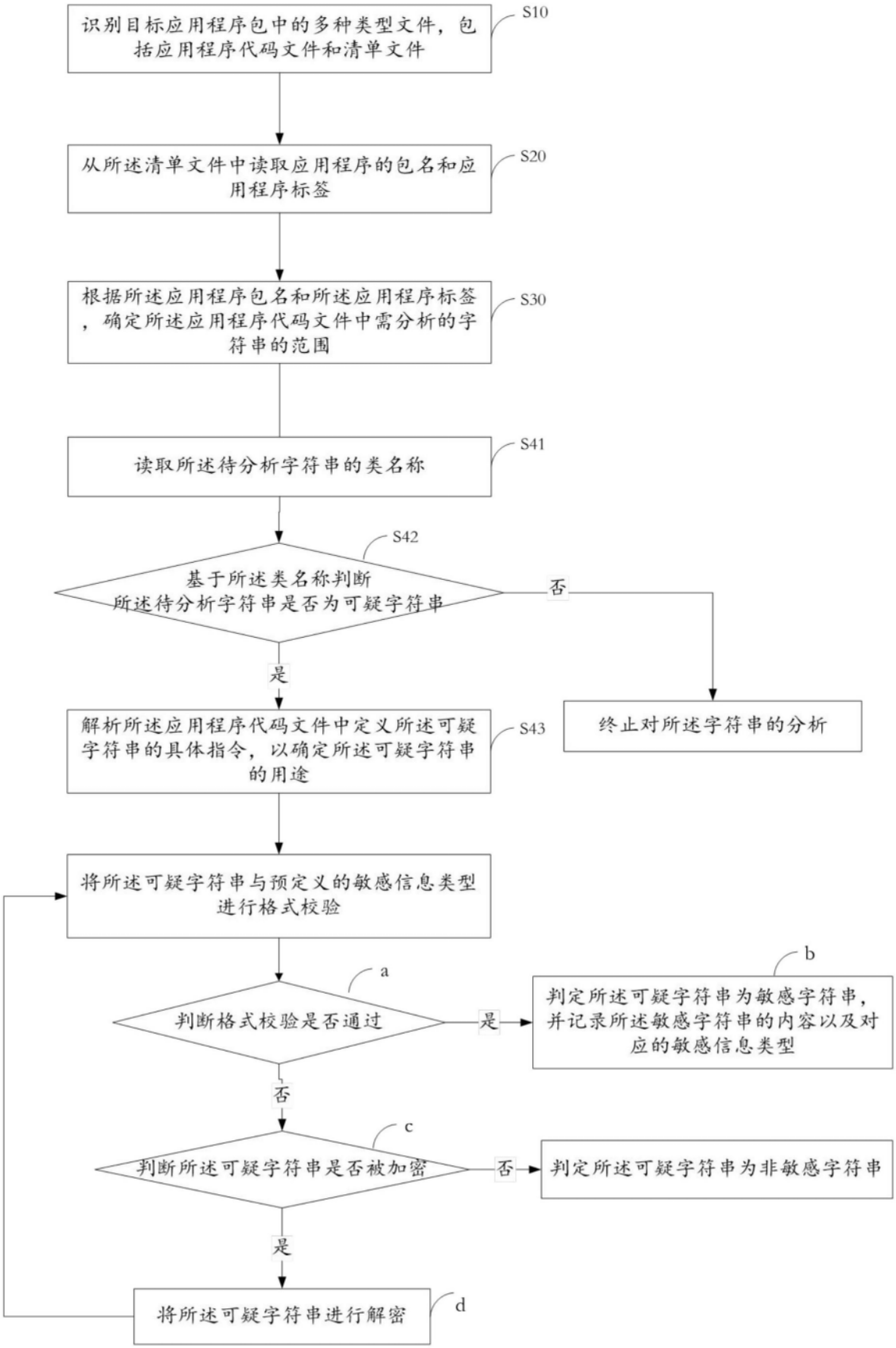

识别目标应用程序包中的多种类型文件，包括应用程序代码文件和清单文件
S10
从所述清单文件中读取应用程序的包名和应用程序标签
S20
根据所述应用程序包名和所述应用程序标签，确定所述应用程序代码文件中需分析的字符串的范围
S30
读取所述待分析字符串的类名称
S41
S42
基于所述类名称判断所述待分析字符串是否为可疑字符串
否
是
终止对所述字符串的分析
解析所述应用程序代码文件中定义所述可疑字符串的具体指令，以确定所述可疑字符串的用途
S43
将所述可疑字符串与预定义的敏感信息类型进行格式校验
a
判断格式校验是否通过
是
b
判定所述可疑字符串为敏感字符串，并记录所述敏感字符串的内容以及对应的敏感信息类型
否
c
判断所述可疑字符串是否被加密
否
判定所述可疑字符串为非敏感字符串
是
将所述可疑字符串进行解密
d

图5

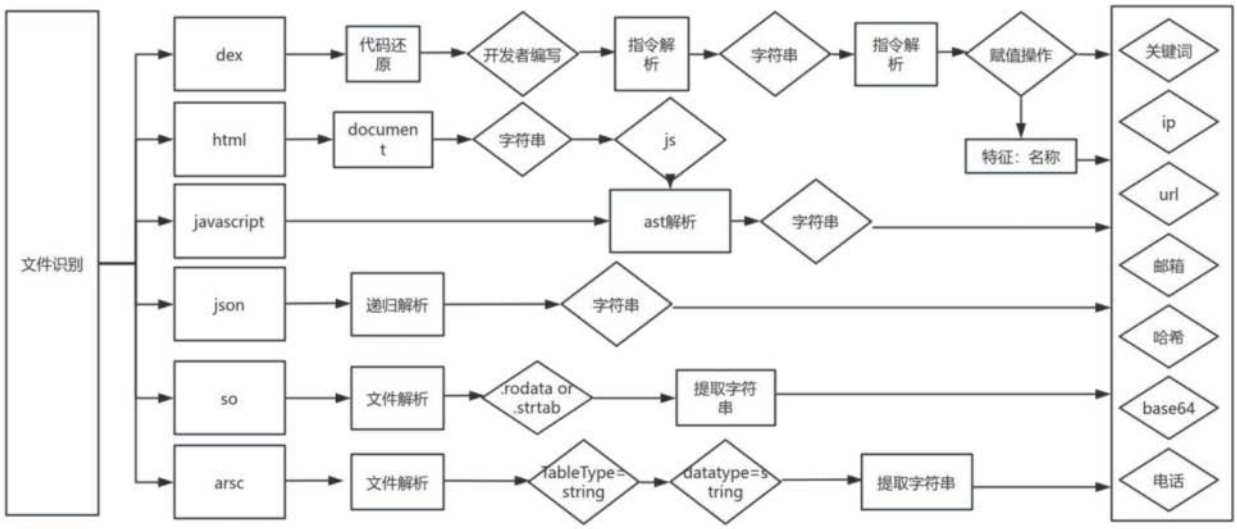

文件识别
dex
代码还原
开发者编写
指令解析
字符串
指令解析
赋值操作
关键词
html
document
字符串
js
特征：名称
ip
javascript
ast解析
字符串
url
json
递归解析
字符串
邮箱
so
文件解析
.rodata or .strtab
提取字符串
哈希
arsc
文件解析
TableType=string
datatype=string
提取字符串
base64
电话

图6

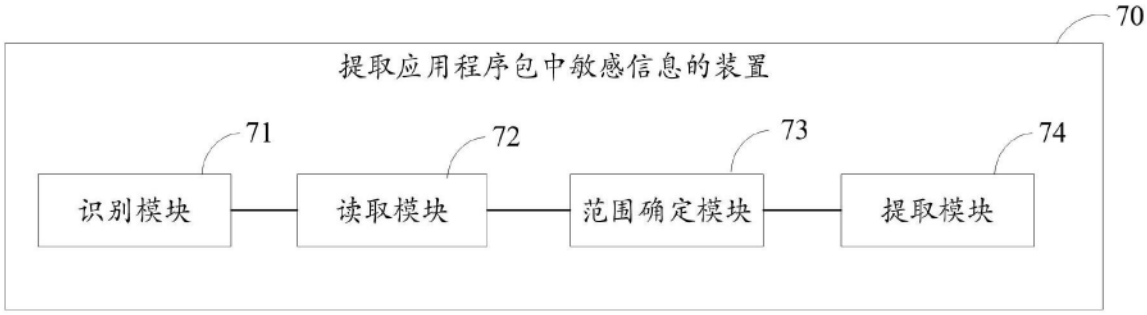

70
提取应用程序包中敏感信息的装置
71
识别模块
72
读取模块
73
范围确定模块
74
提取模块

图7

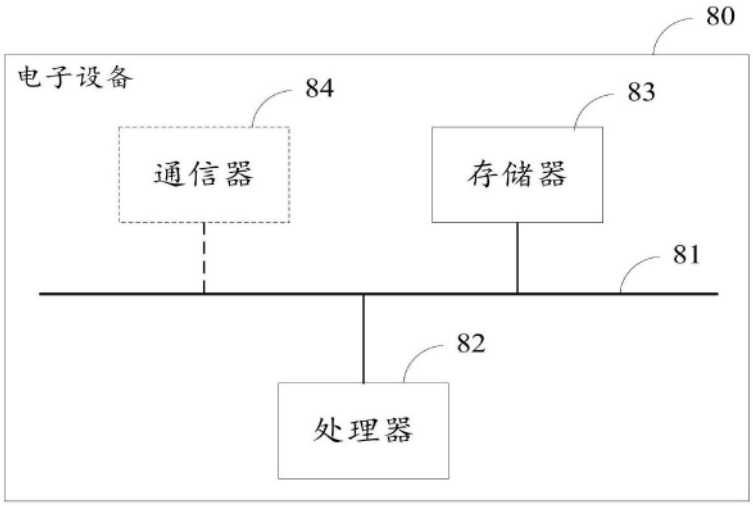

80
电子设备
84
通信器
83
存储器
81
82
处理器

图8