**哈尔滨工业大学计算机学院软件工程系**

**软件工程硕士学位论文开题报告**

**研 究 生 吕婷婷**

**学 号 16S137002**

**入 学 时 间 2016年 09月**

**实 习 单 位**

**校 内 导 师 傅忠传教授**

**实习单位导师**

**论 文 题 目 Android应用漏洞扫描系统的设计与实现**

**开题报告日期 2017年12月08日**

目 录

[目 录 I](#_Toc500010111)

[1 论文选题的目的和意义 1](#_Toc500010112)

[1.1课题来源，项目名称 1](#_Toc500010113)

[1.2课题研究背景与意义 1](#_Toc500010114)

[1.3本课题有关的国内外研究情况 2](#_Toc500010115)

[1.4本课题研究的主要内容 3](#_Toc500010116)

[2 研究方案 4](#_Toc500010117)

[2.1需求分析 4](#_Toc500010118)

[2.1.1系统模块需求分析 5](#_Toc500010119)

[2.1.2系统性能需求分析 6](#_Toc500010120)

[2.1.3系统可行性分析 6](#_Toc500010121)

[2.2系统总体设计方案 7](#_Toc500010122)

[2.2.1系统总体架构设计 7](#_Toc500010123)

[2.2.2系统功能结构设计 12](#_Toc500010124)

[2.2.3系统数据结构设计 17](#_Toc500010125)

[2.3方案实施所需的条件 17](#_Toc500010126)

[2.3.1技术条件 17](#_Toc500010127)

[2.3.2试验条件 18](#_Toc500010128)

[2.4存在的主要问题和技术关键 18](#_Toc500010129)

[2.4.1 存在的主要问题 18](#_Toc500010130)

[2.4.2 技术关键 19](#_Toc500010131)

[2.5预期达到的目标 20](#_Toc500010132)

[3 研究计划进度表和经费预算及经费落实 21](#_Toc500010133)

[3.1 研究计划进度表 21](#_Toc500010134)

[3.2 经费预算及经费落 21](#_Toc500010135)

[主要参考文献 22](#_Toc500010136)

[校内外导师意见 24](#_Toc500010137)

# 1 论文选题的目的和意义

## 1.1课题来源，项目名称

Black Hat（黑帽）大会在8月落幕，大会每年都会汇聚世界上最优秀的安全研究人员，展示新的威胁研究和漏洞发现，揭露最大的、致命性的新攻击。今年的威胁研究名单中，包括了一些渠道最常见的软硬件中发现的安全漏洞，包括苹果设备、容器、Office 365、Windows 10、VMware、安卓移动设备等。

关于安卓的议题有两个，分别为安卓固件挑战和安卓防病毒。安卓固件挑战：Krytowire安全研究员公布了某些安卓移动设备上的中间件允许远程监控用户，并在未经用户许可的情况下获取个人信息，包括用户设备信息、短信、通话记录、应用使用信息等；安卓防病毒：安全研究员发现安卓最新的漏洞在于移动病毒，AVPASS（一个可以绕过安卓防病毒程序的工具），利用AV程序的漏洞检测模型，加上APL扰动技术，将恶意软件伪装成良性应用程序。

近日，Wifi加密协议被比利时鲁汶大学的研究员MathyVanhoef发现存在重大安全漏洞：用于保护Wifi网路安全的WPA2安全加密协议被破解。据Global Web Index数据显示，近50%的设备受到影响，导致用户信息被泄漏。总之，随着移动互联网的发展，由安全漏洞尤其是高危漏洞造成的网络安全威胁不断加剧，各类网络安全事件层出不穷，移动设备安全已经上升到国家战略层面。因此，针对安卓app漏洞扫描系统的研究具有重要的意义。

## 1.2课题研究背景与意义

随着移动互联网的发展和智能手机的普及，手机上网给手机网民带来了极大的便利，随之而来的手机安全问题也不断突出。据中国互联网络信息中心发布的第40次《中国互联网络发展状况统计报告》，截至2017年6月，我国手机网民规模到达7.51亿，其中Android设备占80%，较2016年底增加2830万人。Android平台使用数量的日益增加，每个设备存在google、手机开发商、芯片开发商等多种来源的软件和每个软件模块没有统一的安全审计制度和测试流程导致质量良莠不齐，给恶意攻击者提供一个良好的攻击目标。其次，安全部门监管力度不够以及网民用户缺乏安全意识，使得Android手机安全问题日益严重，用户个人信息泄漏事件增加，对用户的隐私以及财产安全造成巨大风险。例如，2012年底开始出现mmap类的漏洞利用，第三方软件可以轻易地修改内核数据甚至代码来提权，Samsung平台上的CVE-2012-6422漏洞[12]。除了mmap类的漏洞外，用户态与驱动数据交换导致的任意地址写漏洞也在2013年开始大量被发现，例如Qualcomm平台上的copy\_from\_user类漏洞CVE-2013-6123[13]。第三方软件可以用伪造的数据与摄像头驱动做交互，进而修改任意地址的内容。

面对上述问题，各大手机厂商、应用发布平台以及Android系统发布商逐步加大对Android应用安全检测力度。但是，Android应用开发人员对安全知识的匮乏以及没有足够安全意识使开发的应用依然存在极大的安全漏洞威胁。即使在一些大型公司，有类似于SonarQube、Coversity等检测代码缺陷工具，但是这些检测工具也仅仅能检测出代码层面的缺陷与威胁，对于漏洞检测依然于事无补。因此，针对Android手机开发出自动化漏洞检测工具很有必要。

## 1.3本课题有关的国内外研究情况

Android常见的漏洞主要有：越权绕过、钓鱼欺诈、拒绝服务、权限提升、权限泄漏。越权绕过：没有对调用Activity的组件进行权限验证。钓鱼欺诈：启动一个Activity时，加入标志位FLAG\_ACTIVITY\_NEW\_TASK，能够使被启动的Activity立马呈献给用户，用于钓鱼欺诈。拒绝服务：本地组件启动时没有对Intent.getXXExtra()获取或者处理数据进行异常捕获，从而导致攻击者可以通过向受害者应用发送空数据、异常或者畸形数据来使应用crash的目的。权限提升：当一个具有高权限的Sevice是被导出的时，如果没有调用这个Service进行权限限制和调用者身份验证时，那么恶意的应用将具有调用高权限的Service的能力来执行高权限行为等。权限泄漏：主要存在于某些具有高权限操作的组件被调用，而系统没有进行严格的验证和权限控制而导致其他应用可以利用该组件而产生越权操作的权利。

目前国内外Android漏洞扫描的方法主要是以下三种方式：静态分析、动态分析和人工分析。静态分析：利用apktool、dex2jar、jd-gui、smali2dex等静态分析工具对应用进行反编译，并对反编译后的class文件、xml文件静态分析扫描，通过关键字搜索静态方法将具有安全隐患的代码进行摘录并存入检测平台后台，为后续的安全检测报告提供数据。动态分析：对应用软件安装、运行过程的行为检测和分析。检测的方法包括沙箱模型和虚拟方式。虚拟机方式通过建立与Android手机终端软件运行环境几乎一致的虚拟执行环境，手机应用软件独立在其中运行，从外界观察应用程序的执行过程和动态，进而记录应用程序可能表现出的恶意行为。人工分析：专业安全人员对待检测应用，对其进行安装、运行和试用，通过在试用过程中，逐步掌握应用的特点，并通过专业经验，来确定检测重点。当前免费的安卓漏扫平台只能风险监测，扫出来的漏洞大部分是不可利用、很难利用甚至误报。真正可以扫出高危漏洞的工具不开源，需要支付高额的费用。总之当前Android漏洞扫描工具普遍存在以下缺点：

（1）app先安装，再检测漏洞，类似于Drozer、Xpose Framework。安装过程包括内存计算、用户授权等，耗费时间。

1. 分析源码。Apk经过一系列反编译得到Java源码后分析检测。

（3）付费扫描。免费漏扫工具只提供扫描概述，详细描述和漏洞利用需支付昂贵的费用。

（4）云检测系统。云检测系统是收入导向型，检测需付费；检测一个app需大约5-10分钟，耗费时间与金钱。

（5）单次扫描单个app，不支持单次扫描多个app。

（6）app编译过程需安装第三方类库，存在兼容性问题。

## 1.4本课题研究的主要内容

本课题主要研究内容是Android应用漏洞扫描系统的设计与实现。系统分为六个模块：漏洞引擎、漏洞向量库、漏洞管理、重新打包检测、批量app分析检测、POC利用。漏洞引擎包含：静态DVM引擎、关键字搜索引擎、过滤引擎，漏洞向量库包含7种漏洞的扫描与分析，漏洞管理包含：漏洞的扩展、更新，重新打包检测用于检测应用是否被恶意修改。以上六个模块中漏洞向量库、重新打包检测、三大引擎是设计的重点与难点。批量检测是系统的一大亮点与难点，系统检测一个应用的时间将是制约批量检测速度与效率的因素。针对系统的跨平台使用和简单操作，系统最终以可执行文件的形式呈现。

# 2 研究方案

## 2.1需求分析

首先使用本系统的人物角色只有需要扫描Android应用漏洞的人，不存在不同权限管理的问题。其次，系统不需要根据不同角色划分不同的操作界面和功能权限，所以，系统最终以可执行文件的形式呈现。

漏洞扫描分析的前提条件是apk经过反编译，反编译apk后的包结构组成如图2-1所示：



图2-1.apk组成

Lib文件是apk依赖的第三方类库；res包存放的是可编译的资源文件，例如图片、字符串、界面布局等；Assets目录下是不会被编译处理的资源文件,例如可配置文件。Classes.dex是java代码翻译后的Dalvik（ART）可执行字节码。AndroidManifest.xml：该文件是每个应用都必须定义和包含的，它描述了应用的名字、版本、权限、引用的库文件等信息。META-INF文件包含apk签名，修改文件后重新打包签名是很重要的；esources.arsc：编译后的二进制资源文件的索引（apk文件的资源表索引）。其中Classes.dex、AndroidManifest.xml和META-INF是系统漏洞扫描分析的重点。

### 2.1.1系统模块需求分析

首先，安卓app漏洞扫描系统分为六个模块：漏洞引擎、漏洞向量库、漏洞管理、重新打包检测、批量app分析检测、POC利用。其中漏洞引擎、漏洞向量库、批量app分析检测是核心模块，POC利用是可扩展模块。

（1）漏洞引擎模块包括三大引擎：静态DVM引擎、字符串搜索引擎、过滤引擎。静态DVM引擎为漏洞扫描扫描分析提供运行环境，类似于运行在安卓中的Dalvik VM（ART），是一个指令集表。相比较真机，指令集删减一些无用的指令。字符串搜索引擎用于定位关键代码位置，python工具包fuzzy是采用正则表达式模糊匹配工具，速度慢，过滤引擎是过滤掉不需要扫描的代码库，例如com.parse、com.facebook等。

（2）漏洞向量库实现了目前发现的常见漏洞类型，根据漏洞类型将其分为用户app漏洞和系统漏洞两大类。用户app漏洞目前发现的主要类型有：World Readable与World Writable漏洞，例如Microsoft Office漏洞和android NDK漏洞；ContentProvider漏洞和 Directory Traversal漏洞，app直接和SQLite 数据库连接就会爆发ContentProvider漏洞；WebView文件访问和Exported components漏洞；SSL漏洞；Implicit Broadcast漏洞；Dynamically Registered Unprotected BroadcastReceiver漏洞；Allow Debuggable漏洞。漏洞向量库的设计与实现是本系统的核心，每一种向量库的实现都有5个步骤：

1. 收集漏洞信息，分析漏洞原理，定位到漏洞代码所在位置。
2. 分析不同版本apk，确定最新受到影响的apk版本。
3. 分析不同加密程度的apk。
4. 漏洞复现与利用，确定受到影响的Android平台以及Android SDK版本。

测试修复漏洞向量。

（3）重新打包检测是系统的一个重要模块，检测该模块用于解决当apk被反编译后，黑客对classes.dex文件做了修改以及添加危害代码给用户带来隐私财产损失。系统针对重新打包检测提出四种检测方法：SSL指纹验证、结合基于编译指纹的方法进行检测、base64编码字符串、签名保护。SSL指纹验证是用于校验WebView验证过程是否存在缺陷，验证方式有SHA256记录值、如果不一致，表明数据通讯被修改过，base64编码字符串并不是最好的方案。签名保护是目前大部分app采用的方式，所以，系统重点在于签名检测。

（4）MongDB大数据分析检测是系统支持批量分析app。目前市场中大部分漏洞扫描工具只支持单次扫描单个app，可以从应用商店下载批量的app，一次扫描出结果，并将扫描结果存入mongoDB数据库中。安全评估结果内包括：漏洞向量名称、源码路径、安全级别（安全级别分为四个级别：Critical、Warning、Notice、Info）、向量类别、漏洞详细解释、解决办法、参考文献。

（5）系统可扩展模块有漏洞管理、POC漏洞利用，这两部分是系统的难点。漏洞管理和POC漏洞利用是系统的难点与扩展点，随着CVE漏洞的增长，漏洞向量库应该及时的扩充，扩充过程是需要技术支撑；POC漏洞利用属于扩展内容，因为目前漏洞的利用是比较花费时间精力的，对安全研究员的专业素养要求很高，其次很多漏洞的利用方式没有公开的说明。

### 2.1.2系统性能需求分析

系统扫描一个app所需的时间是衡量系统高效性的一个重要指标，大概一个app所需要的时间在20s左右，批量扫描速度不会是所有app扫描速度的叠加，会在整个基础上优化，所以，批量的概念需要后期实现的时候实验得出。

### 2.1.3系统可行性分析

可行性研究的目的就是要用最小的代价在尽可能短的时间内确定问题是否能够解决。当然不能靠主观猜想而是要靠客观分析。必须分析几种主要的可能解法的利弊，从而判断原定的系统目标和规模是否现实，系统完成后所能带来的效益是否大到值得去投资开发这个系统的程度。因此，可行性研究实质上是要进行一次压缩简化的系统分析和设计的过程，也就是在较高层次以较抽象的方式进行的系统分析和设计的过程。可行性研究主要从以下三个方面来进行。

（1）技术可行性

使用现有的技术能不能实现这个系统。由于本系统主要围绕着漏洞向量库的设计、三大引擎的实现等，这部分内容涉及漏洞扫描静态分析流程，重点难点是漏洞向量库的设计与实现。需要大量的实验支撑，所以实施起来时间耗费占整个项目一半时间。POC漏洞利用，这部分是可扩展部分，目前，国内外并没有漏洞利用的可行性分析报告，所以属于可扩展部分。在开发工具方面：安卓app漏洞检测系统会依赖大量的类库，其中数据分析的图构建与分析、apk分析用到的xml分析、crc32字符校验分析等，所以采用Python为系统实现语言。

（2）经济可行性

系统可以采用Pycharm进行开发，结合国内外发表的论文开发设计，在开发过程中不涉及到财力的投入。但系统一旦投入到使用后，将在很大程度上方便协助安全开发者挖掘漏洞，提高工作效率。

（3）操作可行性

使用系统的用户主要是安全开发者以及需要扫描应用漏洞的人，并且系统是完全以可执行文件的方式呈现，仅仅需要单条命令行即可拿到扫描结果。

## 2.2系统总体设计方案

本系统分析与设计阶段采用Python作为开发语言，应用apktool、dex2jar、Pycharm作为辅助工具，目前完成了系统架构设计、功能结构设计和主要数据结构设计。

### 2.2.1系统总体架构设计

本课题主要研究内容是安卓app漏洞扫描系统的设计与实现，漏洞扫描流程如图2-2所示：



图2-2.漏洞扫描过程

（1）第一步，反编译Apk。反编译后文件目录如下表2-1：

表2-1.apk组成

|  |  |
| --- | --- |
| **文件或目录** | **作用** |
| META-INF/ | 从java jar文件引入的描述包信息的目录 |
| res/ | 存放资源文件的目录 |
| libs/ | 存放的ndk编出来的so库 |
| AndroidManifest.xml | 程序全局配置文件 |
| Classes.dex | 最终生成的dalvik字节码 |
| Resource.ars | 编译后的二进制 |

不同的apk反编译程度不一致，一般分为以下四种：

1. 无混淆无加密五加壳。直接利用反编译程序得到的源码，代码逻辑清晰，可分析性强。
2. 混淆。通常利用Proguard做的防护。因为对jar不可逆混淆，因此不能还原出原来的代码。但是代码结构一致、代码逻辑清晰。
3. 加密。对于这种加密方式，在程序运行中会解密，或者从Smali代码分析。
4. 加壳。即使代码加壳，但是运行的时候一定是Dalvik可是别的Odex代码，可利用Xpose分析。

不同app采用的加密程度不同，不能从Java源码角度分析漏洞，系统在字节码分析漏洞。反编译过程是分析apk的前提，系统采用三种反编译方式：dad、ded、dex2jar，按照指定的参数采用不同的方式编译。Apk本质是一个压缩包，在未指定反编译参数的情况下，采用zip解压方式。反编译后重点解析Classes.dex和AndroidManifest.xml。AndroidManifest.xml包含应用组件、数据权限、版本信息、debug等重要信息。AndroidManifest.xml文件采用sax分析方式，将信息结构化为类存储在内存中，在漏洞扫描分析中调用。Classes.dex是java源码编译后生成的java字节码文件，是漏洞扫描分析的重点文件。Classes.dex是Java源码经过Jvm编译后成class文件，然后经过Dvm编译后的dex文件，数据文件格式有严格的定义，如图2-3所示

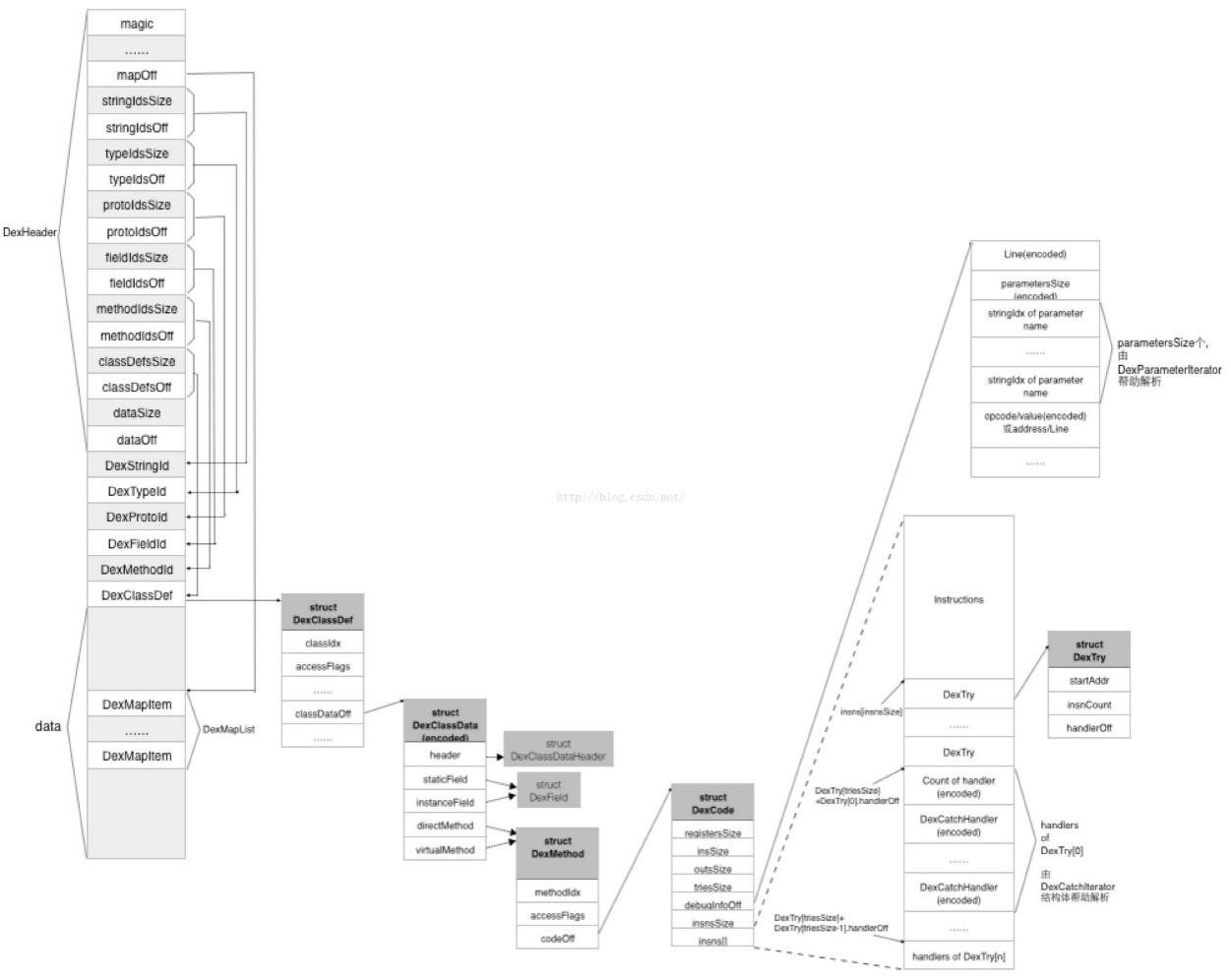


图2-3.dex文件格式

dex文件是一张二进制表，没有明确的分隔符，按照文件规定的格式排版。总共分为三块：文件头、索引区、数据区。文件头包含android sdk版本信息、魔数、校验值、SHA-1哈希值以及文件大小等信息。索引区定义了整个dex中的字符串、类型、方法声明以及方法的信息，结构体的开始位置和个数来自于dex文件头中的记录。数据区是索引区的最终数据偏移以及文件头中的描述的map\_off偏移所指的数据，以及即将解析的class\_def\_item结构，这个结构由dex文件头中的classDefsSize和classDefsOff所指向, 描述Dex文件中所有类定义信息, 每一个DexClassDef中包含一个DexClassData的结构(classDataOff),每一个DexClassData中包含了一个Class的数据, Class数据中包含了所有的方法, 方法中包含了该方法中的所有指令。基本数据结构含义如表2-2所示：

表2-2.dex文件数据结构

|  |  |
| --- | --- |
| **类型** | **含义** |
| U1 | 等同于uint8\_t，表示1字节的无符号数 |
| U2 | 等同于uint16\_t，表示2字节的无符号数 |
| U4 | 等同于uint32\_t，表示4字节的无符号数 |
| U8 | 等同于uint64\_t，表示8字节的无符号数 |
| sleb128 | 有符号LEB128，可变长度1~5字节 |
| uleb128 | 无符号LEB128，可变长度1~5字节 |
| uleb128p1 | 无符号LEB128加1，可变长度1~5字节 |

（2）第二步，依据漏洞向量库扫描出潜在的漏洞。Android常见的漏洞类型：World Readable与World Writable漏洞，是由于外部存储卡的文件没有权限管理，所有应用都可读可写，可以全局读写敏感信息或者root后读取明文信息；WebView文件访问和Exported components漏洞，是由于恶意网页调用APP代码或者通过Java Reflection调用Runtime执行恶意代码；App很容易被反编译成可读文件，然后加入恶意代码。组件暴露后，使得组件被调用时未做验证或者调用其他组件时未做验证。当前Android被爆出的漏洞主要有：应用反编译（apk包被反编译成可读文件，修改后重新打包新的apk）、数据的存储与传输（外部存储SD卡上的文件没有权限，任何应用都可以读写）、密码泄露（密码明文存储、传输）、组件暴露（组件被调用时为做验证、调用其他组件时没有验证）、WebView（恶意app可以注入JavaScript代码进去WebView中的网页，网页未作验证）、LogCat泄露用户敏感信息、恶意的广告包。针对以上常见问题，总结分析以后本系统的漏洞向量库主要有7种：World Readable与World Writable漏洞、 ContentProvider漏洞和 Directory Traversal漏洞、WebView文件访问和Exported components漏洞、SSL漏洞、Implicit Broadcast漏洞、Dynamically Registered Unprotected BroadcastReceiver漏洞、Allow Debuggable漏洞。

（3）第三步，完成分析当前漏洞以后，需要进一步确认漏洞代码范围。回溯到漏洞代码位置，对当前漏洞代码上下文，重新扫描分析一遍。确认以后进行下一个漏洞扫描分析。

本课题针对目前存在的安卓应用检测系统存在的缺点，设计实现apk静态扫描、支持大规模漏洞扫描的系统。系统的主要功能模块设计如图2-4所示：

图2-4.系统模块功能图

首先，安卓app漏洞扫描系统分为六个模块：漏洞引擎、漏洞向量库、漏洞管理、重新打包检测、批量app分析检测、POC利用。其中漏洞引擎、漏洞向量库、批量app分析检测是核心模块，POC利用是可扩展模块。

### 2.2.2系统功能结构设计

漏洞引擎模块、向量库模块、重新打包检测模块将是整个系统最核心的实现技术。由于Python简洁高效的开发效率，漏扫系统编程语言选Python作为开发语言。以下对系统关键模块的实现方案进行描述。

（1）DVM漏洞扫描引擎

静态DVM引擎做大的特点是要做到部分代码扫描，以Writable漏洞为例，DVM引擎的工作原理如图2-5所示。



图2-5.静态DVM引擎部分代码分析过程

SharedPreferences是Android平台访问和修改配置信息等轻量级存储数据的接口，数据采用xml键值对的方式保存，一共有四种创建文件模式：Context.MODE\_PRIVATE、Context.MODE\_APPEND、Context.MODE\_WORLD\_READABLE、ontext.MODE\_WORLD\_WRITEABLE。其中Wordl\_Readable和world\_writeable模式存在安全漏洞。如上图所示：Java代码context.getSharedPreferences(“sensitive\_file”,1)是存在World\_Readable漏洞，所以这段代码是搜索关键字。代码对应的Smli字节码如图所示，对应到指令集，静态DVM引擎中是代码对应的指令，然后应用字符串搜索引擎找到漏洞代码位置。

（2）DVM字符串搜索引擎

字符串搜索引擎，引擎的输入参数为：关键字。经过分析后输出参数：完整的字符串、匹配位置或者匹配路径。在字节码中，字符串的指令为const-string/const-string-jumbo，在安卓中，字符串被表示string\_id\_item[offset]。算法查找步骤为：在ids（string\_id\_item[]to data）映射所有的字符串， 根据用户输入查找字符串并获得ids，用opcode指令查找代码，并且与第二步的字符串比较。如图2-6所示：

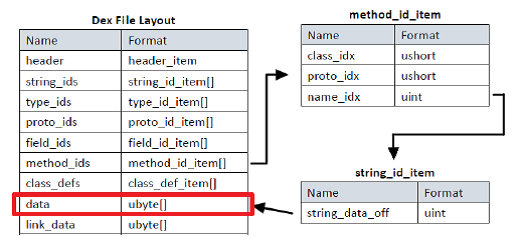


图2-6.字符串查找流程

（3）World Readable 漏洞和 World Writable漏洞

World\_Readable World\_Writeable漏洞（SharedPreferences任意读写漏洞），是由于文件没用使用正确的创建模式，导致文件可被其他应用访问，并执行读写操作，可能会造成敏感信息的泄漏。检测流程为：Android API 为getSharedPreferences(String name, int mode)，对应到Smali中的字节码文件，判断对寄存器的赋值。mode模式定义如图2-7所示：

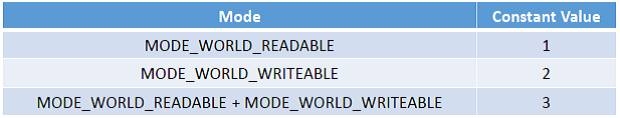


图2-7.模式分类

（4）ContentProvider 漏洞& Directory Traversal漏洞

该漏洞源于对外暴露Content Provider组件的应用，没有对Content Provider组件的访问进行权限控制和对访问的目标文件的Content Query Uri进行有效判断，攻击者利用该应用暴露的Content Provider的openFile()接口进行文件目录遍历以达到访问任意可读文件的目的；触发漏洞的原因是：对外暴露的Content Provider组件实现了openFile()接口； 没有对所访问的目标文件Uri进行有效判断，如没有过滤限制如“../”可实现任意可读文件的访问Content Query Uri；漏洞代码位置如图2-8、图2-9所示：

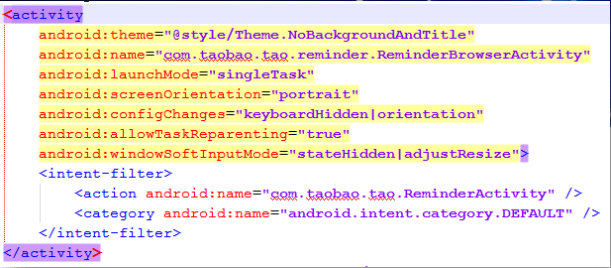


图2-8.文件遍历漏洞源码位置



图2-9.文件遍历漏洞代码实现位置

（5）WebView 文件访问和Exported components漏洞

WebView中，主要漏洞有三类：任意代码执行、密码明文传输漏洞、域控制不严格漏洞，任意代码执行漏洞原因为：WebView 中 add[Java](https://www.2cto.com/kf/ware/Java/)scriptInterface（） 接口 WebView 内置导出的 searchBoxJavaBridge\_对象 WebView 内置导出的 accessibility 和 accessibilityTraversalObject 对象；密码铭文传输漏洞原因：WebView默认开启密码保存功能，开启后，在用户输入密码时，会弹出提示框，询问用户是否保存密码，密码会被明文保存到/data/data/com.package.name/databases/webview.db中，存在密码被盗取的危险。

（6）SSL 漏洞

OpenSSL的某个模块存在一个BUG，当攻击者构造一个特殊的数据包，满足用户心跳包中无法提供足够多的数据会导致memcpy把SSLv3记录之后的数据直接输出，该漏洞导致攻击者可以远程读取存在漏洞版本的openssl服务器内存中长达64K的数据。 也就是说，当攻击者得到这64K数据后，就有可能从数据中得到当前用户的用户名，密码，Cookies等敏感信息，更要命的是，这是远程获取的，也就是攻击者只要在自己电脑上提交恶意数据包，就能从服务器上取得这些数据。不要认为区区64K问题不大，问题是攻击者可以反复提交，也就是能源源不断的得到“新的”64K，理所当然的包含了新的用户信息，因为网站的用户也是源源不断的。

（7）Implicit Broadcast漏洞

该漏洞形成原因是：当发送未经许可或者隐式发送broadcast ，并且broadcast携带有敏感信息。隐式与显示发送广播示意图如图2-10，图2-11。

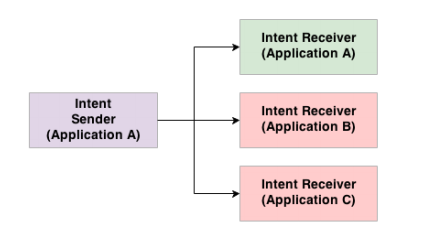


图2-10.隐式广播

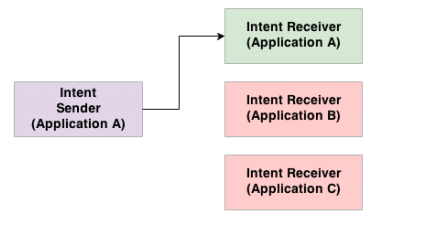


图2-11.显示广播

（8）动态注册不受保护的BroadcastReceiver漏洞

漏洞原因：开发者有时候忘记保护动态注册不受保护的BroadcastReceiver，而且，BroadcastReceiver携带敏感信息。传播流程如图2-12所示。

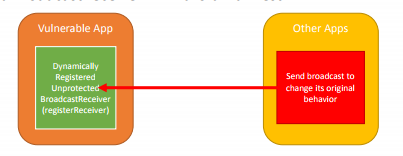


图2-12. BroadcastReceiver敏感信息泄漏过程

（9）Allow debug漏洞。

AndroidManifest.xml ，它是每个 Android 程序中必须的文件。它位于整个项目的根目录，描述了 package 中暴露的组件（ activities, services等），他们各自的实现类，各种能被处理的数据和启动位置。 除了能声明程序中的 Activities, ContentProviders, Services, 和 Intent Receivers,还能指定 permissions 和 instrumentation （安全控制和测试）。而在 AndroidManifest.xml 文件中， debuggable 属性值被设置为 true 时（默认为 false ），该程序可被任意调试 ，这就产生了任意调试漏洞。

（10）重新打包检测

应用市场上的app有可能是经过恶意改造后的山寨app，用户安装后会造成一定程度的隐私财产损失。所以检测重新打包是本系统又一个重要的功能。一般的检测方案有四种检测方式：SSL证书认证、基于编译质问的方法进行验证、签名保护、base64编码字符串。具体的实现方案，会在系统设计与实现阶段详细研究。

### 2.2.3系统数据结构设计

系统在批量应用漏洞扫描分析过程中会用到数据库，由于分析结果数据结构松散，类似于json格式，以及数据类型复杂，系统采用MongoDB作为数据库。总共有一张表，用于存储所有app扫描结果，存储数据如下表：

表2-3.mongdb字段含义

|  |  |
| --- | --- |
| **字段名称** | **解释说明** |
| Platform | 应用平台（Android、IOS） |
| Package Name | 包名 |
| Package Version Name | 包名版本 |
| Package Version Code | 包名版本代码 |
| Min SDK | 最小SDK版本 |
| Target SDK | 目标sdk版本 |
| MD5 | MD5值 |
| SHA1 | Sha1值 |
| SHA256 | SHA256值 |
| SHA512 | SHA512值 |
| Signature | 签名值 |
| Result | 分析结果，分析结果分为Critical、Warning、Notice、Info，表示结果的严重性 |
| Time | 分析一共花费的时间 |

## 2.3方案实施所需的条件

### 2.3.1技术条件

Python是主要的编程语言，对Python使用语法、主要的数据结构、编程将要使用到的类似于校验和验证算法、图论使用等工具类熟练掌握。

非关系型数据库MongoDB，基本的使用规则，与关系型数据库的差异，安装、使用步骤熟练掌握，灵活使用并且存储扫描分析结果数据。

Android四大组件：Activity、Service、BroadcastReceiver、ContentProvider，它们的安全性非常重要，四大组件的安全漏洞主要集中在是否可以被外部调用，外部调用是否存在风险，决定因素是在AndroidManifest.xml里面定义的四大组件的标签export的布尔值。安卓四大组件的安全漏洞问题，不仅仅是上述描述，还有以他的一些安全漏洞，所以，在开发过程中不断深入Android相关知识。

安卓漏洞扫描基础：目前安卓app存在主要漏洞类型有Log敏感信息泄露、web https校验错误忽略漏洞、sql注入漏洞、Provide组件暴露漏洞、Fragment注入漏洞等，需要熟悉这些常见漏洞的原理，分析当前漏洞扫描工具实现扫面的流程以及实现原理，综合本系统实现特征，选出最优的实现方案。

### 2.3.2试验条件

根据系统设计与实现，要对结果进行测试验证，所以需要的实验条件有：

1. 不同应用发布平台，类似于华为市场、小米市场等；
2. 不同版本的app，app漏洞会随着版本升级增加或者减少，所以需要不同版本的app；
3. 不同类型的app，不同类型app漏洞种类数量都不一样，所以需要有不同类型的app。上述条件一般都可以在网上找到所需的资料。

开发硬件设备：一台Windows10 配置合适的电脑、一台安卓测试机。

开发环境：Python2.7、Idea、Java开发环境。

## 2.4存在的主要问题和技术关键

### 2.4.1 存在的主要问题

在需求分析阶段，调研需求功能，在整体功能模块划分阶段，发现系统存在一些技术难点、开发时间把控难点，问题具体分为以下几点：

（1）系统模块划分为六个模块：漏洞引擎模块、漏洞向量库块、MongoDB大数据分析模块、漏洞管理模块、重新打包检测模块、漏洞利用模块，每个模块都存在技术难点。漏洞引擎模块分为三个核心模块：静态DVM引擎、字符串过查找引擎、过滤引擎。

（2）静态DVM引擎不是运行在Android OS环境中，是一个模拟环境，所以和真实环境差距是指令集表是不一致的，需要剔除一些无用的指令、其次，在模拟环境中存在的漏洞是否在真实环境中也存在，这个问题需要做实验去验证。字符串查找引擎不仅区别于正则表达式查找方式，而且查找实在字节码中匹配查找并返回目标位置。把Java源码中的字符串转换为字节码的字符将是一大难点，需要去学字节码class结构组成，做一个转换映射。

（3）漏洞向量库是最核心的部分。向量库的实现将直接影响到扫描系统功能是否强大、完善。所以，这部分最大的挑战是将目前存在的漏洞类型实现，如何实现是需要大量的资料支撑，大量的实验验证。

（4）MongoDB大数据分析模块是漏扫系统最大的亮点。目前市场中的扫描工具不支持批量扫描，而本系统是支持。批量扫描是指从应用市场一次下载多个app，然后可以一次扫描完成，将扫描结果存在数据库，进行随后的分析。问题在于，如何整合核心扫描功能，如何做到批量扫描，以及批量是一次最多可以扫多少，扫描结果是否是可信的。

（5）漏洞管理模块是最难的一个模块，目前还没有很合适的解决方案，本系统初步的方案是，及时关注CVE漏洞的发布，在原有代码基础上扩展最新漏洞向量。

（6）漏洞利用模块，这个模块作为可扩展模块，本系统的主要功能是漏洞扫描，漏洞扫描结果的利用在时间充裕的情况下，分析完成。开发时间把控问题，整个系统开发技术要求很高，很多技术当前没有开放的解决方案。

### 2.4.2 技术关键

本系统的的核心功能是漏洞引擎模块、向量库模块、重新打包检测模块。漏洞引擎模块由三个关键引擎组成；静态DVM引擎、字符串搜索引擎、过滤引擎。静态DVM引擎类似于运行在Android OS上的DVM或者ART，扫描部分反编译后的字节码。字符串搜素引擎用于搜索关键字符串，系统不采用正则表达式匹配过程，利用字节码指令在字节码中直接匹配搜索。过滤引擎用于过滤开源包，例如com.parse、com.facebook等第三方依赖库。

漏洞向量库是最核心的模块，扫描过程依赖的核心部分。根据目前已经发现的漏洞，系统计划实现7种：World Readable 漏洞和 World Writable漏洞、ContentProvider 漏洞& Directory Traversal漏洞、WebView 文件访问和Exported components漏洞、SSL 漏洞、Implicit Broadcast漏洞、动态注册不受保护的BroadcastReceiver漏洞、Allow Debug漏洞。

重新打包检测模块，检测apk是否被黑客修改增添恶意代码，系统计划采用四种检测方：式SSL证书认证、基于编译质问的方法进行验证、签名保护、base64编码字符串。

## 

## 2.5预期达到的目标

本课题将设计开发出安卓app漏洞扫描系统，预计该系统功能功能有：（1）扫描安全漏洞；（2）静态检测代码是否安全 ；（3）权限检测；（4）扩充漏洞信息库；（5）检测应用安全保护机制（防止黑客重新打包后入侵）。预计该系统可以快速高效的检测出当前app潜在的漏洞。

# 3 研究计划进度表和经费预算及经费落实

## 3.1 研究计划进度表

表3-1.计划表

|  |  |  |
| --- | --- | --- |
| **时间** | **项目进度** | **开发阶段** |
| 2017.07-2017.10 | 调研安全漏洞工具现状、完成开题报告 | 需求分析 |
| 2017.10-2017.11 | 漏洞扫描系统不同安全扫描模块设计、接口设计、MongoDB数据库设计、模块接口层次设计。 | 概要设计 |
| 2017.11-2017.12 | 漏洞扫描系统不同模块算法设计，接口详细设计、数据库数据存储格式详细设计、模块之间耦合、层级关系设计。 | 详细设计 |
| 2018.01-2018.03 | 根据详细设计编码实现模块，实现模块之间层次关系的调用，实现关键算法。调试整个系统顺利运行。 | 编码开发 |
| 2018.04 | 实验测试系统，找出问题，回归优化算法 | 测试 |
| 2018.05-06 | 论文撰写 | 结尾 |

## 3.2 经费预算及经费落

目前没有经费预算。

# 主要参考文献

1. 吴国信. Android通信漏洞扫描方法研究与实现[D]. 大连理工大学, 2014.
2. 董国伟, 王眉林, 邵帅,等. 一种基于特征匹配的Android应用漏洞分析框信息安全漏洞分析与风险评估大会. 2015.
3. 谢健. Android漏洞分类及可扩展扫描引擎的设计与实现[D]. 西安电子科技大学, 2014.
4. 曹琛, 高能, 向继,等. 安卓系统服务中输入验证型漏洞的初步分析[J]. 信息安全学报, 2016, 1(1):1-11.
5. 刘昊晨. 基于Lua引擎的Android漏洞检测工具研究[D]. 西安电子科技大学, 2015.
6. 王永科. 安卓应用登录认证机制安全性分析与改进[D]. 中国科学院大学, 2015.
7. 李千目, 高双双, 侯君,等. 基于Android终端的漏洞扫描系统: CN, CN 103227992 A[P]. 2013.
8. 肖卫, 张源, 杨珉. 安卓应用软件中Intent数据验证漏洞的检测方法[J]. 小型微型计算机系统, 2017, 38(4):813-819.
9. M. Kozlovszky. Cloud security monitoring and vulnerability management. In Critical Infrastructure Protection Research, pages 123--139. Springer, 2016.

[10]M. Elliott, H. Yu, X. Yuan, and J. Zhan. Savmds: A software application vulnerability management dashboard system. In Proceedings of the World Congress on Engineering, volume 1, 2014.

[11]M. Barrere, R. Badonnel, and O. Festor. Vulnerability assessment in autonomic networks and services: a survey. Communications Surveys & Tutorials, IEEE, 16(2):988--1004, 2014.

[12][Jason Bau , Elie Bursztein , Divij Gupta , John Mitchell, State of the Art: Automated Black-Box Web Application Vulnerability Testing, Proceedings of the 2010 IEEE Symposium on Security and Privacy, p.332-345, May 16-19, 2010](https://dl.acm.org/citation.cfm?id=1849982&CFID=821422136&CFTOKEN=59947077) [doi>[10.1109/SP.2010.27](http://dx.doi.org/10.1109/SP.2010.27)]

[13][Prithvi Bisht , Timothy Hinrichs , Nazari Skrupsky , V. N. Venkatakrishnan, WAPTEC: whitebox analysis of web applications for parameter tampering exploit construction, Proceedings of the 18th ACM conference on Computer and communications security, October 17-21, 2011, Chicago, Illinois, USA](https://dl.acm.org/citation.cfm?id=2046774&CFID=821422136&CFTOKEN=59947077) [doi>[10.1145/2046707.2046774](http://doi.acm.org/10.1145/2046707.2046774)]

[14]M. K. Gupta, M. Govil and G. Singh, "Static Analysis Approaches to Detect SQL Injection and Cross Site Scripting Vulnerabilities in Web Applications: A Survey," in IEEE International Conference on Recent Advances and Innovations in Engineering, Jaipur, India, 2014.

[15]B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, and J. K. Zinzindohoue. A messy state of the union: Taming the composite state machines of TLS. In 36th IEEE Symposium on Security and Privacy, May 2015.

[16] J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow. Elliptic curve cryptography in practice. In 18th International Conference on Financial Cryptography and Data Security, Mar. 2014.

[17] Y. Cao, Y. Fratantonio, A. Bianchi, M. Egele, C. Kruegel, G. Vigna, and Y. Chen. EdgeMiner: Automatically Detecting Implicit Control Flow Transitions through the Android Framework. In Symposium on Network and Distributed System Security (NDSS), 2015.

[18] [William Enck , Peter Gilbert , Byung-Gon Chun , Landon P. Cox , Jaeyeon Jung , Patrick McDaniel , Anmol N. Sheth, TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones, Proceedings of the 9th USENIX conference on Operating systems design and implementation, p.393-407, October 04-06, 2010, Vancouver, BC, Canada](https://dl.acm.org/citation.cfm?id=1924971&CFID=1014160285&CFTOKEN=46947418)

[19] K. Z. Chen, N. M. Johnson, V. D'Silva, S. Dai, K. MacNamara, T. R. Magrino, E. X. Wu, M. Rinard, and D. X. Song. Contextual policy enforcement in android applications with permission event graphs. In NDSS, 2013.

[20]H. Huang, S. Zhu, P. Liu, and D. Wu. A framework for evaluating mobile app repackaging detection algorithms. In Trust and Trustworthy Computing. Springer, 2013.

校内外导师意见

|  |
| --- |
| **副导师**（校外导师、实习单位导师）意见：  导师（签字）：  年 月 日 |
| **导师**（校内导师）意见：  导师（签字）：  年 月 日 |