**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 校内自定义实习单位**

**校 内 导 师 傅忠传副教授**

**实习单位导师 傅忠传副教授**

**论 文 题 目 安卓app漏洞扫描系统 的设计与实现**

**开题报告日期 2017年10月26日**

**目录**

[1 论文选题的目的和意义 3](#_Toc496455317)

[1.1课题来源，项目名称 3](#_Toc496455318)

[1.2课题研究背景与意义 3](#_Toc496455319)

[1.3与本课题有关的国内外研究状况 4](#_Toc496455320)

[1.4本课题研究的主要内容 5](#_Toc496455321)

[2 研究方案 8](#_Toc496455322)

[2.1技术方案 8](#_Toc496455323)

[2.1.1技术路线 8](#_Toc496455324)

[2.1.2技术措施 9](#_Toc496455325)

[2.2方案实施所需的条件 11](#_Toc496455326)

[2.2.1技术条件 11](#_Toc496455327)

[2.2.2试验条件 11](#_Toc496455328)

[2.3存在的主要问题和技术关键 11](#_Toc496455329)

[2.3.1 存在的主要问题 11](#_Toc496455330)

[2.3.2技术关键 12](#_Toc496455331)

[2.3.3技术方案 13](#_Toc496455332)

[2.4预期达到的目标 18](#_Toc496455333)

[3 研究计划进度表和经费预算及经费落实 19](#_Toc496455334)

[3.1 研究计划进度表 19](#_Toc496455335)

[3.2 经费预算及经费落 19](#_Toc496455336)

[主要参考文献 20](#_Toc496455337)

[校内外导师意见 21](#_Toc496455338)

# 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总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安全漏洞扫描系统的研究具有重要的意义和重要性。

## 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，但与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，我们手机网民规模到达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导致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，有类似与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静态代码缺陷或者冗余，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与本课题有关的国内外研究状况

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到处，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。过程步骤包括内存计算、用户授权等，耗费时间。

（2）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，耗费财力物力。

（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。

针对以上问题，本课题将提出完全不同于市场漏洞扫描工具，支持批量大规模扫描、不需要安装检测漏洞、不需要分析源码、免费的自动化漏洞扫描工具。

## 1.4本课题研究的主要内容

本课题主要研究内容是安卓app漏洞扫描系统，整体扫描流程如图1所示：



图1.漏洞扫描过程

第一步，使用google开源工具AndroidGuard反编译apk，第二步，扫描。以实现的漏洞向量库为依据，辅助扫描三大引擎（详细说明见下文）扫描。最后一步，根据扫描结果回溯到漏洞代码被调用的代码位置，重新扫描相关的代码。

系统的主要功能模块设计如图2。

图2.系统模块功能图

首先，系统分为个大模块。核心是三大引擎的设计以及漏洞向量库的实现。三大引擎分别是静态DVM引擎、字符串搜索引擎、过滤引擎。漏洞向量库实现了目前发现的常见的静态漏洞类型，根据漏洞类型将其分为用户app漏洞和系统漏洞两大类，以后随着漏洞类型的发现，更新漏洞库将会是设计上的一个技术难点。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。

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。

系统可扩展模块有漏洞管理、POC漏洞利用和MongDB大数据分析检测。这三部分是系统的亮点与难点。MongDB大数据分析主要功能是可以批量检测从应用商城中下载的app，并且将安全评估结果存入数据库，其中安全评估结果包含的内容有：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。漏洞管理和POC漏洞利用是系统的难点与扩展点，漏洞向量库应该是不断扩充的，扩充算法还在调研中，基本思路是连接CVE漏洞库，实现自动化扩充下载；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 研究方案

## 2.1技术方案

### 2.1.1技术路线

依据软件设计开发周期与流程，将项目分为以下五个阶段：

1. 需求分析阶段

通过调研国内外安卓漏扫工具的功能需求发现当前工具普遍存在的问题有耗费时间、扫描成本高、不能批量扫描。本系统克服以上功能不足的缺点，最大的亮点是可以模拟安卓运行环境，直接扫描，节约安装过程耗费的时间；系统还具有以下功能：

（1）扫描安全漏洞。系统依据实现的漏洞向量库扫描app存在的漏洞，扫描流程见上文图1说明。

（2）扩充漏洞向量库库。漏洞库会随着漏洞类型的发现，不断扩充。

（3）检测应用安全保护机制（防止黑客重新打包后入侵）。apk被不法入侵者修改后重新打包发布，对用户危害极大，所以，重新打包检测尤为重要。

需求分析阶段调研漏洞扫描流程、主要的扫描工具流程，储备相关知识。

1. 概要设计阶段

概要设计阶段从漏洞扫描系统的整体出发，详细划分每一个功能需求模块、模块内部漏洞扫描算法的选择、模块层级结构之间的设计，接口的设计，数据结构设计，数据存储方式设计。从大体方向上把握整个系统的结构框架设计、结构之间耦合度的设计以及模块内部算法的设计。

1. 详细设计阶段

详细设计主要是为编码阶段提供指导，所以，在概要设计基础上，完成功能模块的详细设计。功能模块分之间的关系耦合实现，严格遵循软件设计7大原则，选择合适的设计模式，调研选择最优的算法，画出程序详细设计类图、序列图和包图。

1. 设计实现阶段。

根据详细设计图，完成编码。编码过程中，选择合适的编码语言，实现每一个模块功能，调试模块之间的依赖关系，使得整个系统完整运行。

1. 测试阶段。

编码完成后，测试每一个漏洞扫描模块，与当前工具比较，重新优化。

### 2.1.2技术措施

调研发现，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。

当前漏洞扫系统都是反编译apk，然后对反编译后的文件进行静态扫描。本课题沿用这种扫描方式，整个扫描流程图见图1.

扫描流程大体分为三个步骤。第一步，反编译apk。反编译apk后的包结构组成如图4所示：



图4.apk组成

上图只有四部分是需要我们扫描的，classes.dex、AndroidManifest.xml、META-INF以及lib文件。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；Lib文件是apk依赖的第三方类库；classes.dex是java代码翻译后的Dalvik（ART）可执行字节码。其余的资源文件包是不需要我们扫描的，所以，系统会开发一个Filtering Engine，用于过滤不需要扫描的包，类似于开源类库。

第二步，根据漏洞向量库扫描漏洞。这部分是本系统的核心，关键是漏洞向量库的实现。调研目前存在的漏洞模式，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。所以，向量库主要是上述7种漏洞扫描实现。其次，系统主要特点有：不需要安装即可扫描、字符串的快速查找、过滤不需要扫描的包。以上三点特点分别设计三种不同的引擎实现。

1. 静态DVM引擎是类似于运行在安卓中的Dalvik VM（ART），是一个简单的指令集表。静态DVM只对某一块关键字节码或者指令码分析，不需要分析Java源码（从apk反编译出字节码，字节码然后编译成Java源码，从字节码到Java源码不是每一个app都可以实现的，假如存在代码混淆，会严重干扰扫描结果）。
2. 字符串搜索引擎。在自然语言处理领域，正则表达式用于查找字符串，但是正则表达式是3型文法，也称为上下文无关文法，而程序代码是上下文相关文法，用正则表达式处理速度非常慢。其次，正则表达式不能返回漏洞代码在代码中出现的位置，所以，本系统要设计一个新的用于扫描定位漏洞代码位置的引擎。
3. 过滤引擎。在反编译后的项目中会依赖一些开源代码，类似于com.parse、com.facebook、com.tapjoy等开源类库，扫描过程中应该过滤掉，避免不必要的扫描，避免浪费时间。

第三步，根据第二步定分析结果，依次回到方法最初始调用代码段，分析相关代码，重新确认漏洞类型。漏扫系统的特点是，不会分析每一行代码，只扫描漏洞代码初始调用相关代码段，如图5所示：

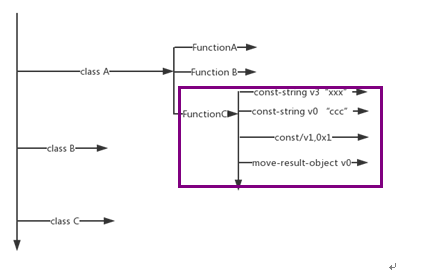


图5.部分代码扫描示意图

## 2.2方案实施所需的条件

### 2.2.1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。

### 2.2.2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。

## 2.3存在的主要问题和技术关键

### 2.3.1 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，而漏扫系统则支撑。批量扫描是指从应用市场一次下砸属多个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，扫描结果是否可行。
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，将漏洞库与CVE漏洞库关联起来，可以做到自动更新漏洞库。
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。
7. 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，所以，在严格把控开发时间的基础上，有一些功能是不一定可以完成，主要看完成进度而定。

### 2.3.2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，类似于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。

### 2.3.3技术方案

针对上述存在的问题，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6所示。



图6静态DVM引擎部分代码分析过程

正如上图所示，漏洞扫描只分析初始调用字节码以及相关字节码。

2、DVM字符串搜索引擎

字符串搜索引擎，其输入参数为：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，字符串被表示为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7所示：



图7.字符串查找流程

漏洞向量库根据现在发现的静态检测漏洞类型，主要实现7种漏洞扫描，以下对7种漏洞的原理与实现进行描述。

3、World Readable 漏洞和 World Writable漏洞

检测方法：首先，找到存在危险调用模式的代码例如openOrCreateDatabase、getDir、getSharedPreferences、openFileOutput。然后用静态DVM引擎扫描模式，模式分类如图8所示。



图8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9所示：

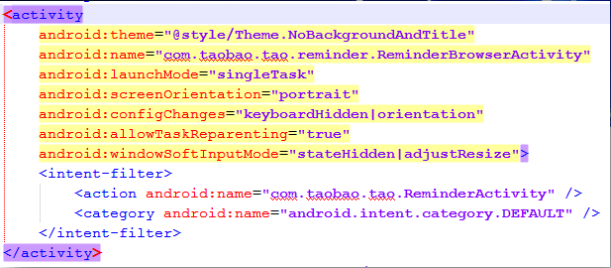


图9.文件遍历漏洞源码位置

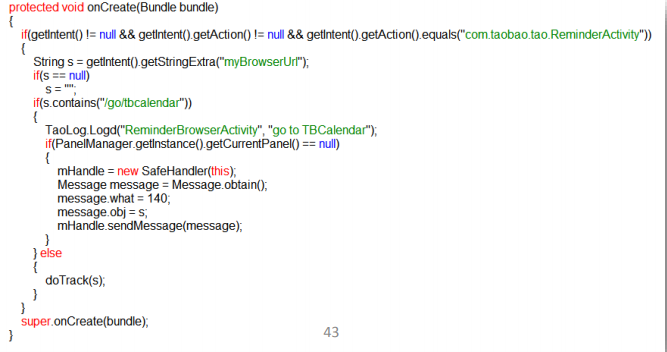


图10.文件遍历漏洞代码实现位置

5、WebView 文件访问和Exported components漏洞

漏洞形成原理：WebView允许开发者将web浏览器潜入到app中，如果Activity有出口组件，就会造成漏洞。在web setting.xml文件将“setAllowFileAccess”设置为true或者缺省，这样会使得其他的app通过url“/data/data/[package name]”访问到数据库或者文件，最终造成敏感信息的泄漏。

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11，图12。

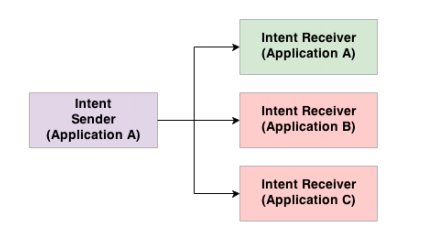


图11.隐式广播

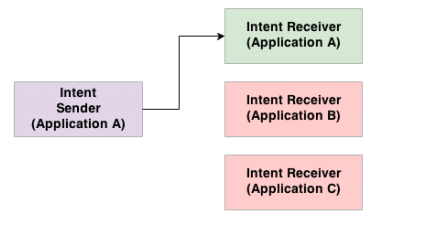


图12.显示广播

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13所示。

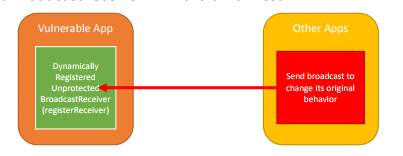


图13. BroadcastReceiver敏感信息泄漏过程

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。

## 2.4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

|  |  |  |
| --- | --- | --- |
| 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

软件开发，不需要硬件支持，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. <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-6422>
15. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-6123

# 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 |