北京信息科技大学

毕业设计（论文）

题　　目： 安卓中间人攻击的研究与实现

学　　院： 计算机学院

专　　业： 计算机科学与技术

学生姓名： 幸小文 **班级/学号** 计科 1302/2013011172

指导老师/督导老师： 路旭强

起止时间： 2017 年 2 月 20 日 至 2017 年 6 月 2 日

摘　要

随着网络信息技术的快速发展和普及，尤其是近年来无线接入技术和和移动终端技术的飞速发展，无线局域网已经成为人们工作和日常生活的必须。同时由于底层网络协议的不安全，导致主机对接受到的数据盲目信任，以至于让数据的传输不安全。中间人攻击（MITM）由于攻击方式简单，难于察觉等特点，成为危害无线局域网的主要方式。生活在局域网随处可见的环境下，了解一些网络数据传输原理和熟悉普遍的局域网攻击方式是很有必要的。

本文对 ARP 协议的内容、工作原理和协议漏洞有一定详细的介绍，并介绍了 ARP 欺骗攻击的原理和基本的防御方法。开发了一款运行在安卓系统上的应用，该应用实现了局域网 ARP 欺骗攻击和防御，主要功能有断网攻击、数据嗅探、HTTP 数据劫持、和防御 ARP 欺骗攻击。通过该应用，普通用户可以体验 ARP 欺骗攻击的危害，加深对局域网安全的认识；安全研究人员可以学习 ARP 欺骗的原理和检测局域网是否安全。

关键词：中间人攻击；　ARP 欺骗；　局域网安全 ；　安卓；数据劫持；

Abstract

With the rapid development and popularization of network information technology, especially in recent years, the rapid development of wireless access technology and mobile terminal technology, wireless LAN has become a must for people to work and daily life. At the same time due to underlying network protocol insecurity, resulting in the host to blindly trust received data, so that the transmission of data is not safe. Man-in-the-middle attack (MITM) is the main way to harm wireless LANs because of the simple attack mode and difficult to detect. Living in the local area network can be seen everywhere in the environment, to understand some of the principles of network data transmission and familiar with the common LAN attack is necessary.

This article describes the contents of ARP protocol, working principle and protocol vulnerabilities, and introduces the principle of ARP spoofing attack and the basic defense method. Has developed an application running on the Android system, the application implements LAN ARP spoofing attack and defense, the main features are broken network attacks, data sniffing, HTTP data hijacking, and defense ARP spoofing attacks. Through this application, ordinary users can experience the dangers of ARP spoofing attacks, deepen the understanding of LAN security; security researchers can learn the principle of ARP spoofing and detect whether the LAN security.

Keywords: Man-in-the-middle attack; MITM; ARP spoofing; Android; Data hijacking; LAN security;

目　　录

第一章　概述

本章主要阐述了什么是中间人攻击、局域网中 ARP 协议工作原理和 ARP 欺骗的原理，并简要的说明了本课题实现了什么功能。

1.1　什么是中间人攻击和 ARP 欺骗

在计算机安全中，中间人攻击（Man-in-the-middle Attack，MITM）是一种“间接”的入侵攻击，攻击者扮演“中间人”的角色，秘密的转播和修改两台相互信任主机间的通信。

在计算机网络中，ARP 欺骗、ARP 缓存中毒或 ARP 毒化路由是一种被攻击者用来发送具有欺骗性的 ARP 消息到局域网中的技术。

1.2　ARP 协议的工作原理和 ARP 欺骗的原理

1.2.1 ARP 工作原理

ARP（Address Resolution Protocol）地址解析协议是用于将网络层地址（IP 地址 32 位）解析为链路层地址（MAC 地址 48 位）的链路层协议。在局域网中，主机间的是以“帧”为单位进行信息传输的，一台主机要和另一台主机进行通信，必须知道目标主机的 MAC 地址。“地址解析”就是在主机发送“帧”之前把目标主机的 IP 地址转换成 MAC 地址的过程。ARP 协议的基本作用就是通过目标主机的 IP 地址来获取目标主机的 MAC 地址。

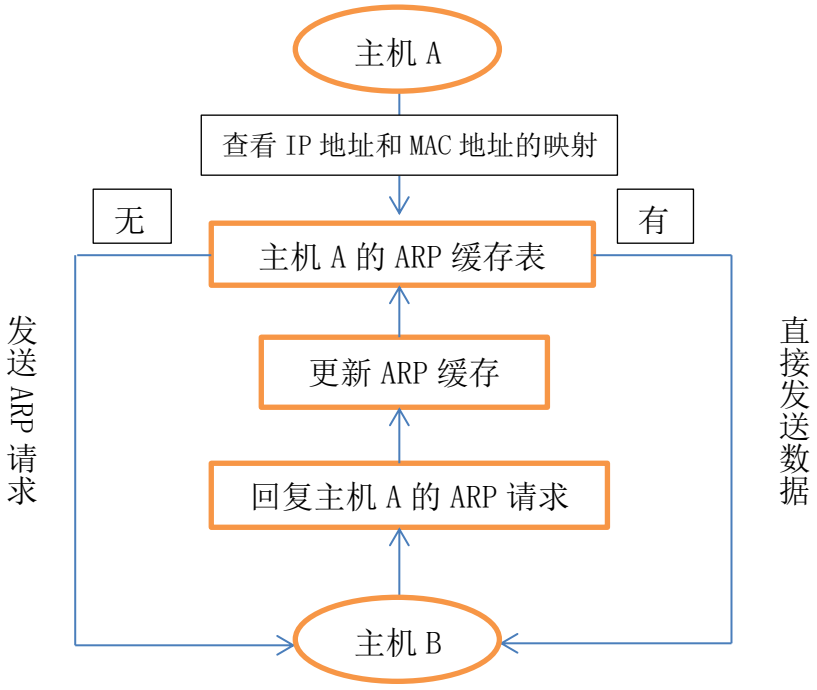

图 1.1 局域网 ARP 工作原理

图 1.1 展示了局域网中 ARP 的工作原理。在局域网中，每一台主机都有一个 ARP 缓存表，这个表保存了该局域网中主机的 IP 地址和 MAC 地址的映射关系。在图一中，主机 A 要向主机 B 发送数据，在主机 A 发送数据之前，要把目标主机 B 的 IP 地址和 MAC 地址封装到数据包里，起初主机 A 只知道

主机 B 的 IP 地址，而不知道 MAC 地址。主机 A 会根据目标主机 B 的 IP 地址去 ARP 缓存表找到和目标 IP 对应的 MAC 地址，如果找到了对应的 IP-MAC 地址，则主机 A 把目标主机 B 的 IP 和 MAC 地址封装后，根据主机 B 的 MAC 地址直接发送数据包。如果没有找到 IP-MAC 映射关系，主机 A 会广播一个包含目标主机 B 的 IP 地址的 ARP 请求的数据帧到局域网的所有主机，目标主机 B 收到该 ARP 请求后会向主机 A 发送一个包含自己 IP-MAC 地址对应的 ARP 应答包。主机 A 收到目标主机 B 的 ARP 应答包后会更新本地的 ARP 缓存表，并根据目标主机 B 的 MAC 地址发送数据包。

1.2.2 ARP 协议的漏洞

ARP 协议是一个高效的数据链路层协议，但也是一个“无状态” 协议，ARP 协议是建立在网络中各个主机互相信任的基础上的，因此其本身就存在设计上的缺陷，主要有下面三个方面：

1、ARP 协议没有认证机制，ARP 请求是以广播形式发送的且即使没有发送 ARP 请求，局域网中的所有主机都可以发送 ARP 应答包，主机不会验证收到的 ARP 应答包是否是目标主机发送的，只要收到的 ARP 应答包是有效的就把收到的 IP-MAC 地址刷新到本地 ARP 缓存表中。

2、ARP 缓存表默认是动态更新的，IP-MAC 地址映射有一个到期时间，ARP 缓存并不维护 IP-MAC 地址映射的状态，也不进行认证，因此，ARP 协议本身不能保证 IP-MAC 地址映射永远都是正确的，它只能保证该映射在得到正确 ARP 应答包至下一个 ARP 应答包之间的一定时间内是有效的。

1.2.3 ARP 欺骗的原理

在 1.2.2 中提出了 ARP 协议的设计缺陷，ARP 协议没有“连接”，它没有认证机制，对所有的 ARP 请求和应答都不做校验，并且 ARP 缓存表是动态更新的，只要收到 ARP 应答包就刷新缓存表，下图 2 展示了 ARP 欺骗的原理。

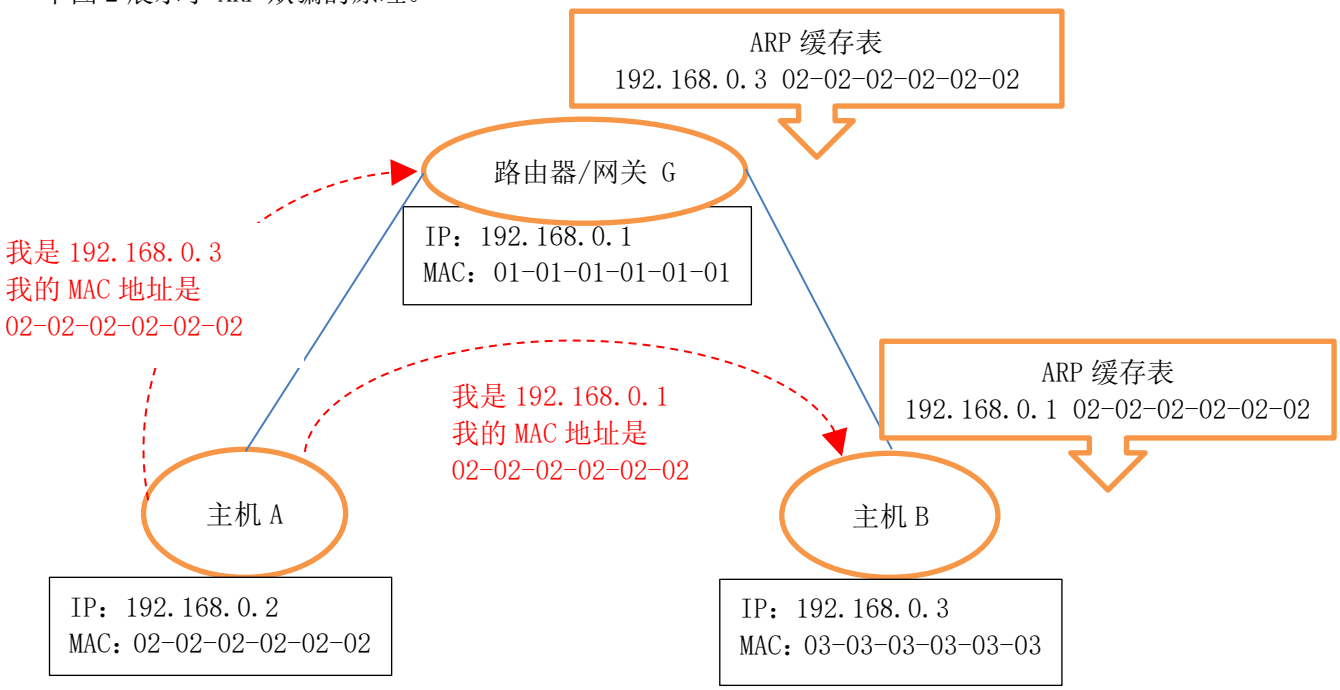

图 1.2 ARP 欺骗原理

上图 1.2 展示了局域网 ARP 欺骗的基本过程，有一台网关，两台主机，它们各自的 IP-MAC 地址映射如下：

网关 G：

IP 地址：192.168.0.1

MAC 地址：01-01-01-01-01-01

主机 A：

IP 地址：192.168.0.2

MAC 地址：02-02-02-02-02-02

主机 B：

IP 地址：192.168.0.3

MAC 地址：03-03-03-03-03-03

主机 A 是攻击者，网关 G 和主机 B 是被攻击者，攻击者 A 分别向网关 G 和主机 B 投 ARP 毒。①攻击者 A 广播 ARP 请求报文来获取 G 的 MAC 地址，再根据 G 的 MAC 地址向网关 G 发送内容为主机 B 的 IP 地址和主机 A 的 MAC 地址的伪造的 ARP 响应报文，网关 G 根据响应报文更新本地 ARP 缓存，网关 G 的缓存表结果为：

网关 G 的 ARP 缓存表	
主机 B 的 IP 地址 192.168.0.3	主机 A 的 MAC 地址 02-02-02-02-02-02

这样网关 G 本来要发往主机 B 的报文就会发往主机 A。②攻击者 A 广播 ARP 请求报文来获取主机 B 的 MAC 地址，再根据 B 的 MAC 地址向主机 B 发送内容为网关 G 的 IP 地址和主机 A 的 MAC 地址的伪造的 ARP 响应报文，主机 B 根据响应报文更新本地 ARP 缓存，结果为：

主机 B 的 ARP 缓存表	
网关 G 的 IP 地址 192.168.0.3	主机 A 的 MAC 地址 02-02-02-02-02-02

经过上面两次 ARP 投毒，网关原来要发送给主机 B 的报文会发送到主机 A 上，主机 B 原来要发送给网关的报文会发送给主机 A，这样主机 A 就可以秘密的获取主机 B 和网关 G 之间的通信数据。

1.3 本课题实现的功能

1.3.1 课题实现的功能

本系统是一款使用 JAVA 语言开发的运行在 Android 系统上的应用程序，主要包含以下功能。

1、局域网扫描：该功能主要是对局域网中可能存在的主机进行扫描，使用的技术是本机向每一个存在该子网下的 IP 地址发送 UDP 报文，由局域网内是根据 MAC 地址来通信的，该局域网下的所有主机都会向本机发送 ARP 响应报文，本机可以缓存所有主机的 IP-MAC 地址映射。本机可以读取本地 ARP 缓存来获取在线的主机。

2、断网攻击：在获取了局域网的所有主机 IP-MAC 地址映射后，本机可以选取某个主机做为被攻击的目标主机，本机可以向目标主机发送伪造的 ARP 应答报文，该报文内容包含网关的 IP 地址和本机的 MAC 地址。这样目标主机发往网关的数据就会发往本机，但因为本机没有开启转发功能，使目标主机不能向外网发送数据，使其断网。

3、数据嗅探：在断网功能的基础上，攻击者主机开启 IP 转发，被攻击者就可以通过攻击者与外网传输数据，使用 tcpdump 命令可以嗅探到被攻击者主机与其它主机的通信数据，保存为.pcap 的文件格式

4、HTTP 数据劫持：在数据嗅探功能的基础上，对嗅探到的数据进行整理和分析，获取到被攻击主机与外界的 HTTP 通信数据，主要获取 cookie 数据。

5、HTTP 数据的保存和查看：本功能主要定义 HTTP 数据结构，并把截取到的 HTTP 数据进行结构化，并以 json 结构保存到本地。对保存的 json 数据文件解析并通过应用显示。

6、ARP 防御：用户开启该功能后，可以防止局域网的其它主机冒充网关。

1.4 论文的组织结构

文章的第一章讲述了课题研究中用到的一些原理；第二章描述了开发应用需要的环境和工具以及它们的安装和使用方法；第三章给出了应用的系统设计，包括系统功能图、包图和界面的设计；第四章对 Android 应用开发的项目结构和 Android 系统的一些组件有了详细的介绍；第五章是系统功能的实现部分，对每个功能的实现细节都有了详细的步骤，包括一些算法流程图等；第六章是系统测试部分，包括测试环境的介绍和测试用例的编写；最后就是结束语和参考文献。

第二章　开发环境和工具

2.1 基本开发环境和工具

本课题是开发一款运行在 Android 系统上的应用，搭建开发环境过程中主要用到了以下环境。

2.1.1 搭建 JDK 环境

JDK(Java Development Kit) 是 JAVA 语言的软件开发工具包(SDK)，它是 JAVA 开发过程的核心，它包括了 JAVA 的运行时环境、JAVA 工具和 JAVA 基础类库。

JDK 的安装：我们可以在 ORACLE 官网下载最新的 JDK 安装包，因为我是在 windows 系统上搭建开发环境，所以我下载的是 windows 版的安装包，双击安装包按照提示把 JDK 装到系统中。

JDK 环境的配置：右键计算机 -> 属性 -> 高级系统设置 -> 环境变量 -> 新建

变量名：JAVA_HOME
变量值：JDK 的安装目录

添加后选择 PATH 变量，追加 ;%JAVA_HOME%\bin

新建 CLASSPATH 变量

变量名：CLASSPATH
变量值：.;%JAVA_HOME%\lib\tools.jar;%JAVA_HOME%\lib\dt.jar

打开命令行工具（CMD），运行 java 和 javac 命令，如果出现命令帮助，则 JDK 环境配置成功，否则重新检查前面的步骤是否出错。

2.1.2 安装 Eclipse

Eclipse 是一款开源的 JAVA 集成开发环境（IDE），使用它可以很简单的编译和打包基于 JAVA 语言的应用程序。

Eclipse 的安装:eclipse 可以在 Eclipse 官网下载最新的 Eclipse 安装包,我下载的是 windows 解压版的安装包，解压版的好处是可以解压在某个目录就可以使用，卸载时只要把文件夹删除即可。

2.1.3 安装 Eclipse ADT 插件

Eclipse ADT 是 GOOGLE 为了方便开发者在 Eclipse 上开发 Android 应用程序而开发的插件。

Eclipse ADT 的安装：

打开 Eclipse，选择 help -> Install New Software -> Add

Name : ADT
Location : http://dl-ssl.google.com/android/eclipse

点击 OK 加载完成后选上所有，跟着引导完成安装。

2.1.4 安装 Android SDK

Android SDK 是专门用于开发 Android 应用程序的软件开发工具包。

Android SDK 的安装：在 2.1.3 中我们安装了 Eclipse ADT，在安装完成重启后，Eclipse 会弹出窗口提示我们没有安装 Android SDK，我们可以选择安装 Android SDK，Eclipse ADT 插件会自动帮我们把 Android SDK 下载到指定文件夹下。在安装完 Android SDK 后，我们可以通过点击 Window -> Android SDK Manager 来安装和管理 Android SDK Tools 和 Android API 版本，在这里我选的 Android API 版本是 API 14。

2.1.5 安装 VirtualBox 虚拟机

VirtualBox 是一款可以性能优异可以虚拟 Android 系统的开源虚拟机软件。

VirtualBox 的安装：VirtualBox 可以在 VirtualBox 官网下载最新的版本，这里我下的是 windows 版本，下载安装包后按照提示完成安装即可。

2.1.6 安装 Android 系统

Android x86 是为了方便 Android 系统在 x86 架构的 CPU 上运行而设计的。

Android x86 可以在 Android x86 官网下载，这里我下载的是 android-x86-4.0-RC2-eeepc 版本。

在 VirtualBox 里安装 Android x86 系统：

打开 VirtualBox，点击新建，主要参数如下：

虚拟电脑名称：Android-4.0
系统类型：Linux
Linux 版本：Linux 2.6 / 3.x (32bit)
虚拟内存大小：512MB
虚拟硬盘：8GB 动态分配

在虚拟机建好后，加载 android-x86-4.0-RC2-eeepc.iso 镜像，按照提示完成 Android x86 系统的安装。

2.1.7 android 系统的网络配置

在虚拟机中安装好 Android x86 系统后，要对虚拟机进行网络设置。为了方便移动，实体机笔记本主要接入的是无线网络，虚拟机要联网，需要采用共享的方式。在 windows 系统上添加设备。选择网络适配器中的 Microsoft loopback Adapter 设备，然后设置当前无线连接共享给该网络适配器。对虚拟机进行网络设置，主要参数如下：

连接方式：桥接网卡
界面名称：Microsoft Loopback Adapter
控制芯片：PCnet-FAST III (Am79C973)
混杂模式：拒绝

完成设置后，启动 Android 虚拟机，在终端输入 `dhcpcd eth0` 命令给网卡 eth0 动态分配 IP 地址。配置完 IP 地址后还要配置 dns，查看实体机的网络属性的 IPv4 DNS 服务器 IP 地址，

在 android 虚拟机的终端输入

```
setprop net.dns1 DNS 服务器的 IP 地址
```

这样就配好了 Android 虚拟机的网络环境。

2.2 其它工具

由于 JAVA 不能对系统的底层进行操作,但其提供了 Runtime.getRuntime.exec(String command)方法来获取系统的运行时环境并执行终端命令。由于 Android 系统底层是基于 Linux 内核的，所以其运行时环境就是 Linux 的终端，其可以执行 shell 命令，我们可以通过执行编译好的可执行程序来实现对底层的操作。

2.2.1 Arpspoof

Arpspoof 是一个开源的 ARP 欺骗工具，是 Linux 环境下的可执行程序，要想在 Android 系统中运行这个命令,要先使用 NDK 进行交叉编译,通过 adb 工具把编译好的静态可执行文件复制到 Android 系统的/system/xbin 目录下，具体方法如下

```
adb push arpspoof /system/xbin
adb shell chmod 777 /system/xbin/arpspoof
```

命令用法为：

```
arpspoof -i interface -t target host
-i 参数：指定使用哪个网卡接口
-t 参数：要欺骗的主机
host:要冒充的主机
```

2.2.2 Tcpdump

Tcpdump 是一个开源的抓包工具，要想在 Android 系统中运行这个命令，要先使用 NDK 进行交叉编译，通过 adb 工具把编译好的静态可执行文件复制到 Android 系统的/system/xbin 目录下，具体方法如下

```
adb push tcpdump /system/xbin
adb shell chmod 777 /system/xbin/tcpdump
```

命令的基本用法为：

```
tcpdump -w fileName -s snaplen host target
-w 参数：嗅探到的数据保存路径
-s 参数：数据包的截取长度
Host 参数：要截取数据的目标主机
```

第三章　系统设计

3.1 系统功能图

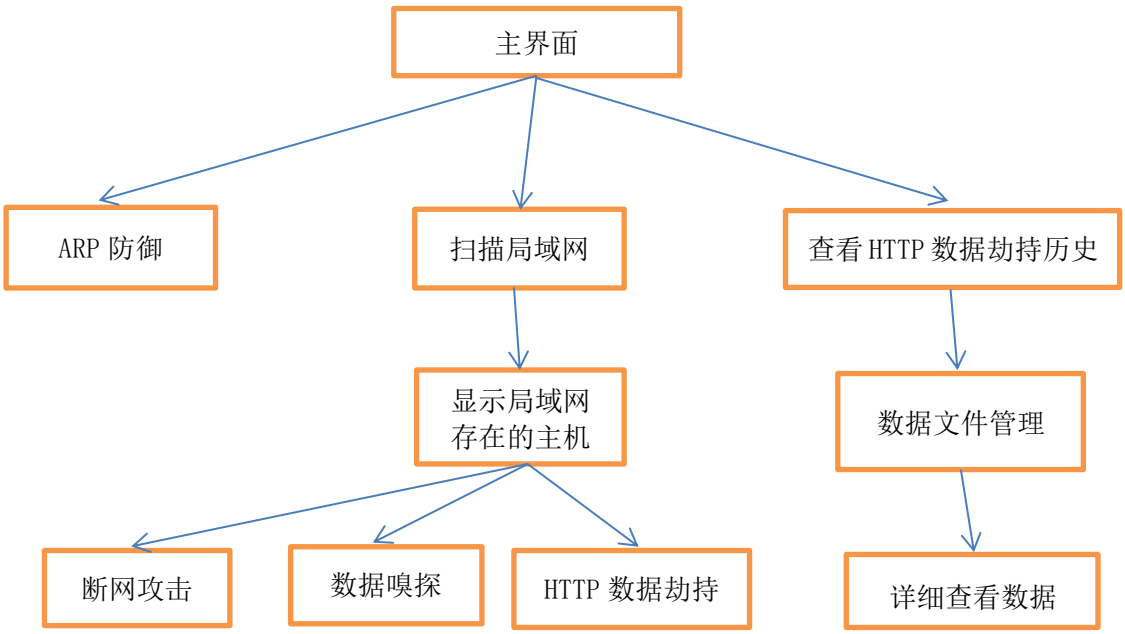

图 3.1 系统功能图

上图 3.1 展示了系统的总体功能和功能之间的联系，主界面是应用程序的入口功能，它是应用程序启动后显示的第一个界面，在这个过程中初始化了一系列的常量和获取了 Android 系统的一些参数，该功能在第四章有详细的介绍。通过主界面可以启动其它三个功能，分别是 ARP 防御、扫描局域网和查看 HTTP 数据劫持历史。ARP 防御功能是一个开关功能，用户可以选择关闭和开启，开启后可以防御其它主机冒充网关。扫描局域网功能和显示局域网中存在的主机是后面断网攻击、数据嗅探和 HTTP 数据劫持功能的基础，它主要是通过一直向局域网发送 UDP 报文来获取主机的响应，再通过读取本地 ARP 缓存来确定局域网中活跃的主机并获取它们的一些信息。断网攻击功能主要是对目标主机进行 ARP 投毒，使其找不到网关从而断网。数据嗅探功能是通过 tcpdump 可执行程序来截取目标主机的通信数据并保存到本地。HTTP 数据劫持功能是对数据嗅探功能的过滤，截取的主要是 HTTP 通信数据。查看 HTTP 数据劫持历史功能是对 HTTP 数据劫持功能保存的数据进行管理和查看，它包括了对数据文件的管理和对某个数据文件的详细查看。

3.2 系统界面设计

3.2.1 主界面

图 3.2 主界面

3.2.2 显示局域网中活动的主机

图 3.3 显示主机

3.2.3 选择攻击方式

图 3.4 攻击方式选择

3.2.4 攻击功能界面

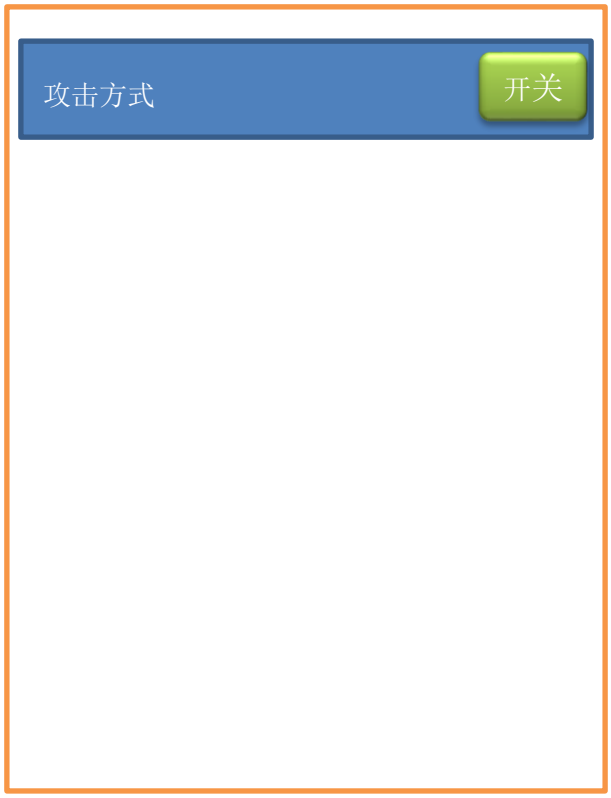

图 3.5 攻击界面

3.2.5 查看 HTTP 劫持历史

图 3.6 查看文件

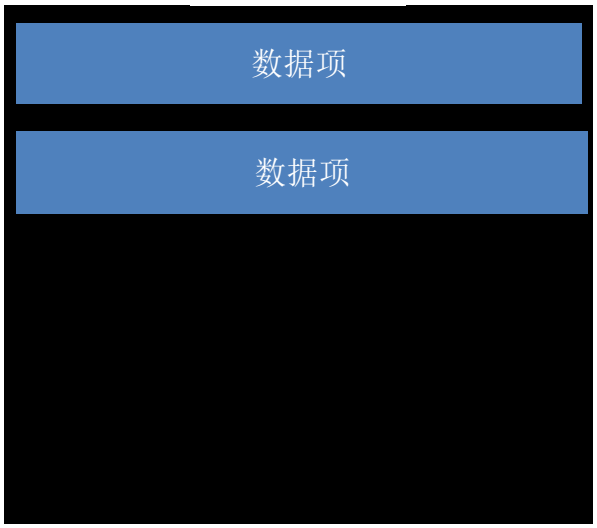

图 3.7 数据项

图 3.8 数据内容

3.3 包图

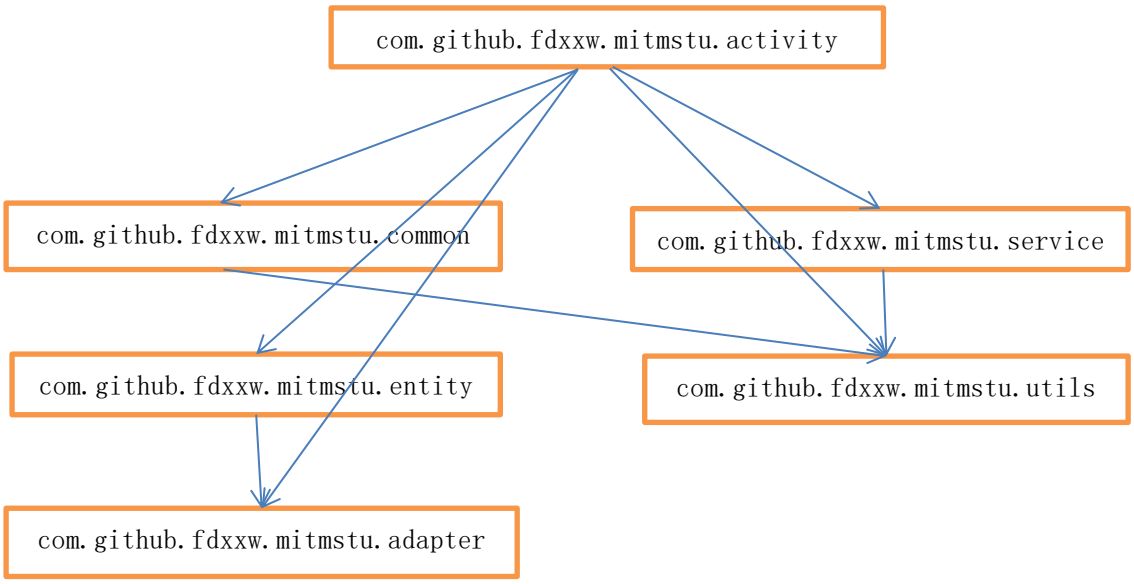

图 3.9 系统包结构

图 3.9 展示了系统的包组织结构，activity 下主要放的是系统的图形活动界面的类，common 下放的是通用的类，service 下放的是服务类，entity 下放的是 javaBean 类，utils 下放的是通用工具类，adapter 下放的是 ListView 需要的适配器类。

第四章　组件和配置文件介绍

4.1 Android 系统组件介绍

4.1.1 Activity 组件

Activity 是应用程序组件，提供一个与用户交互的屏幕，是一个与用户交互的组件，用户可以编写一个继承 Activity 的类来完成某个界面的设计，通过 setContentView(View)来设置屏幕的具体布局。我们可以为每一个 Activity 创建一个 layout.xml 的布局文件，我们可以在这个布局文件中添加我们想要的标签来设计主界面的交互元素和其样式。

4.1.2 Service 组件

Service 也是 Android 系统中的组件，它不提供可视化的交互界面，是一个生命周期长运行在后台的服务组件，它可以无限的运行下去，除非调用 stopService()方法来停止。用户可以编写继承于 Service 的类来实现自定义的后台服务，别的组件可以通过 startService()和 stopService()方法来启动和停止该服务。

4.1.3 Intent 组件

Intent 虽然不是 Android 系统的大组件，但是它是连接 Activity 组件和 Service 组件的桥梁，是用来协调组件之间的交互和通讯，在本课题中 Intent 组件主要被用来从一个 Activity 屏幕跳到另一个屏幕，在一个 Activity 中启动一个 Service 以及在一个 Service 中启动另一个 Service。

4.1.4 Application 组件

Application 组件也是一个系统组件，它是 Android 应用程序的入口点，相当于 Main 函数，每一个 Android 应用程序在启动时都会创建一个 Application 对象，并且这个对象中的数据可以被应用中的所有对象使用，可以被用来保存公共的数据。本课题通过编写一个继承于 Application 的类 AppContext 来保存一些基础公共数据。

4.2 项目配置

4.2.1 项目的目录结构

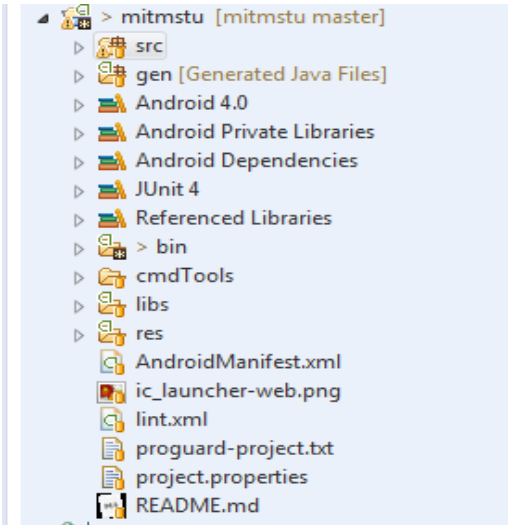

图 4.1 项目目录

根目录：主要是 Android 项目的配置文件 AndroidManifest.xml，包含一些类和权限的注册。

src 目录：该目录下主要用来放置我们写的 JAVA 代码，系统的功能都是在这里实现的。

gen 目录：存放 R.java 文件，是自动生成的，给项目中的每一个资源都生成一个独立的资源 ID。

bin 目录：该目录下主要是存放应用编译后的文件，包括打包后的*.apk 可安装的 Android 应用程序。

libs 目录：该目录放的是应用需要用到的第三方类库。

res 目录：该目录存放的是项目的资源文件，包括图片、字符串、动画、音频和一些 XML 文件。这些资源都会在 R.java 文件中生成唯一资源 ID，我们可以通过 R.java 来访问这些资源。

4.2.2 res 目录介绍

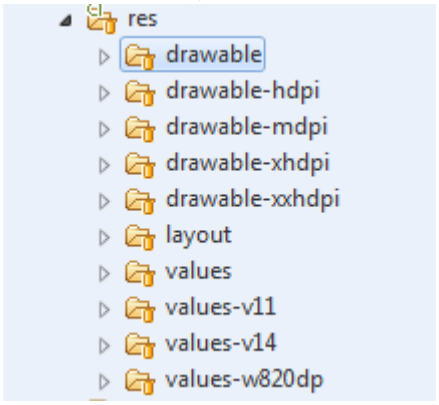

图 4.2 res 目录

drawable 目录：该目录存放的主要是各种图片文件以及 drawable 类型的 xml 文件。

Layout 目录：存放布局文件，每个 Activity 组件都可以设置布局，体现在用户可以看到的屏幕界面，用户可以与屏幕进行交互，比如用户点击某个按钮来触发某些计算和显示。

values 目录：该目录下存放了比较多的资源，包括尺寸、字符串、样式、颜色和数组等资源。这些资源

4.2.3 AndroidManifest.xml

AndroidManifest.xml 是 Android 项目的配置文件，在项目的根目录，项目中自定义的 Activity 和 Service 都必须在 AndroidManifest.xml 中用<activity>标签和<service>标签注册，项目中需要用到的权限比如网络权限、存储权限等，都必须在 AndroidManifest.xml 中用<uses-permission>标签来注册，在 4.1.4 中的 Application 组件也同样需要在 AndroidManifest.xml 中的<application>标签注册，这样应用程序才能正常的运行而不会报错。

第五章　系统实现

5.1 运行 shell 命令的实现

JAVA 提供了一个获取系统运行环境且可以运行终端命令的方法，它就是 Runtime.getRuntime().exec(command)，它返回一个进程，我们可以通过这个进程来获取输入流和输出流，具体代码片段如下：

```
Prccess process = Runtime.getRuntime().exec(command);
DataOutputStream os = new DataOutputStream(process.getOutputStream());
process.waitFor();
BufferedReader successResult = new BufferedReader(new
InputStreamReader(process.getInputStream()));
BufferedReader errorResult = new BufferedReader(new
InputStreamReader(process.getErrorStream()));
```

以上代码片段就是执行终端命令的过程，我们可以获取执行命令后的输入流来知道命令的执行情况，是正确执行还是执行错误。

5.2 主界面的实现

5.2.1 程序入口 Application 实现

包：com.github.fdxxw.mitmstu.common
类名：AppContext
父类：Application

应用程序启动时需要初始化的参数

本机 IP 地址
本机 MAC 地址
本机使用的网卡名称
网关地址
网关 MAC 地址
子网掩码
系统的扩展存储路径

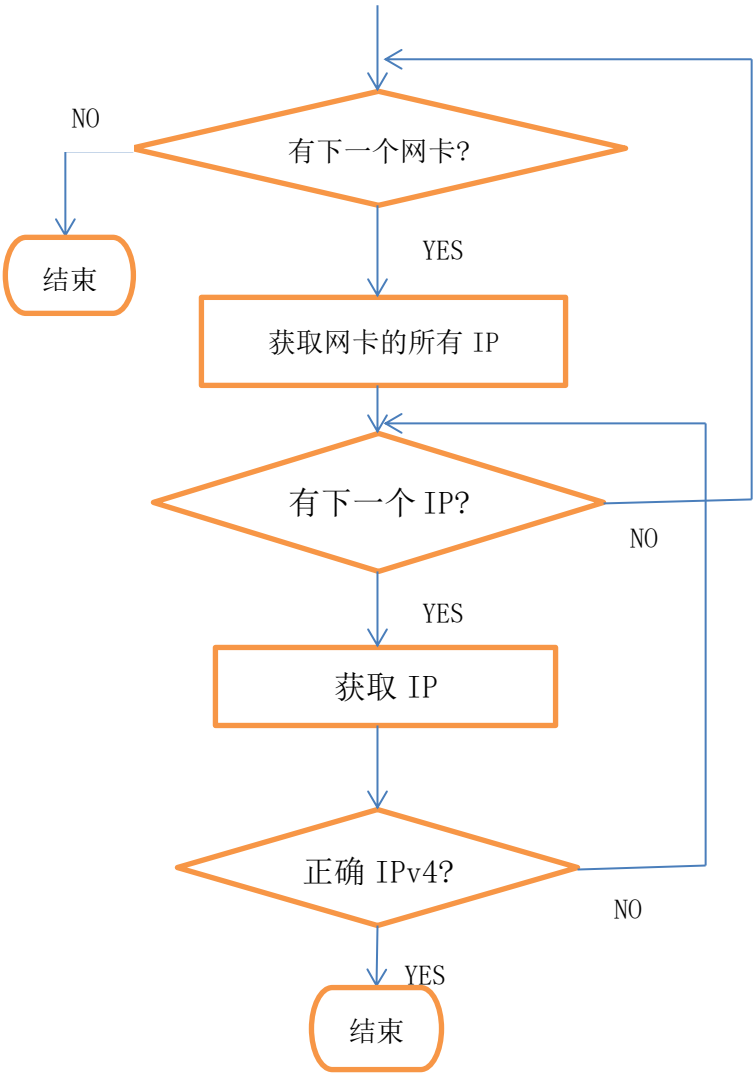

图 5.1 本机 IP 地址获取

上图 5.1 展示了获取本机 IP 地址的算法过程，先遍历所有的网卡，再遍历每个网卡的 IP 地址，直到找到有效的 IPv4 地址。

本机 MAC 地址的获取：在获取有效的 IP 地址后可以返回对应的网卡，通过调用网卡 NetworkInterface 的 getHardwareAddress()方法可以获取该网卡的 MAC 地址。

网卡名称：通过调用网卡 NetworkInterface 的 getDisplayName()方法可以获取该网卡的名称，例如 eth0。

网关地址的获取：在 Android 系统中，有一些初始化的配置文件，文件里面里面配置了开机设置的系统属性值，这些属性我们可以通过在终端执行 getprop [key] 和 setprop [key] [value]来获取和设置属性值。网关地址在 key 为 dhcp.网卡名称.gateway 中，我们可以通过执行终端命令 getprop dhcp.网卡名称.gateway 来获取网关的 IP 地址。

网关 MAC 地址的获取：在获取到网关的 IP 地址后，我们可以执行 arp –a 来获取 ARP 缓存表并通过正则表达式来获取网关 IP 地址对应的网关 MAC 地址，代码片段如下：

```
String gatewayMac = null;
CommandResult result = ShellUtils.execCommand("arp -a " + getGateway(), false, true, true);
String[] msgs = result.successMsg.split("\\s");
for(String msg : msgs) {
     if(msg.trim().matches("([A-Fa-f0-9]{2}:){5}[A-Fa-f0-9]{2}")) {
          gatewayMac = msg;
          break;
     }
}
```

子网掩码的获取：子网掩码位数可以通过执行**ip a** 终端命令来获取网络信息，并通过本机**IP**地址来获取子网掩码位数，代码片段如下：

```
int maskBit = 0;
CommandResult result = ShellUtils.execCommand("ip a", false, true, true);
String successMsg = result.successMsg;
int start = successMsg.lastIndexOf(ip) + ip.length() + 1;
maskBit = Integer.parseInt(successMsg.substring(start, start + 2));
return maskBit;
```

在获取到子网掩码位数后，可以计算出子网的**IP**地址，代码片段如下：

```
int[] tempMask = {0,0,0,0}; //保存子网掩码地址
int times = maskBit/8;    //多少个8位
int i = 0;
for(;i<times;i++) {
    tempMask[i] = 255;
}
for(int j=1;j<=8;j++) {
    if(j <= maskBit-times*8) {
```

```
            tempMask[i] += (int) Math.pow(2, 8-j); //2的多少次方
        } else {
            break;
         }
    }
```

系统扩展存储路径的获取：可以通过Android系统提供的方法Environment.getExternalStorageDirectory().toString() 来获取，Environment是一个提供访问环境变量的类，通过它可以获取一些系统参数。

5.2.2 主界面MainActivity的实现

主界面是在应用程序启动后用户看到的第一个界面，也是其它功能的入口，在本课题里主界面主要有三个触发事件，分别是启用ARP防御的开关、启动HTTP数据文件管理界面和启动扫描局域网功能并显示局域网界面，用到的UI组件有SwitchButton、Button和TextView，布局文件名为activity_main.xml。

ARP防御功能的实现：

给SwitchButton添加开关改变的监听器，当用户点击时改变开关状态，我们通过开关的状态来实现具体的功能。

本课题实现ARP防御的原理是通过静态绑定网关的IP地址和MAC地址，在应用程序启动时，我们获取了局域网网关的IP地址和MAC地址，我们可以使用arp －s 网关IP 网关MAC 来静态绑定，这样网关IP地址对应的MAC地址就不会动态改变，从而其它主机发送的ARP应答报对本机无效，这样就防止了其它主机冒充网关。当开关开启时，执行arp －s 网关IP 网关MAC 命令来启用防护；当开关断开时，执行arp －d 网关IP地址 命令来取消静态绑定而关闭防护。流程图如下：

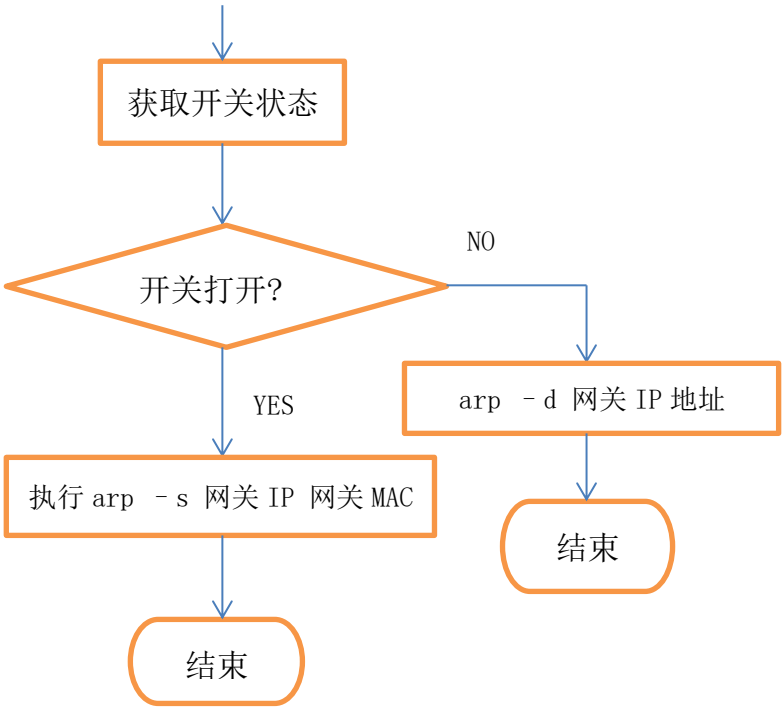

图 5.2 ARP 防御

启动HTTP数据文件管理界面和启动扫描局域网功能并显示局域网界面的实现：

一个界面要启动另一个界面，需要使用Activity组件的startActivity(Intent intent)方法，Intent组件作为参数用来完成两个界面间的通信，使用方法如下：

```
startActivity(new Intent(Context packageContext, Class<?> cls));
```

其中Context类型参数是当前Activity组件对象，cls参数是被启动的Activity组件的类属性。在这里我们给UI组件Button和TextView添加单击的监听事件，当用户点击某个UI组件时就会触发相应的事件，从而实现开启某些功能。Button组件开启的就是启动扫描局域网功能并显示局域网界面，通过startActivity(new Intent(MainActivity.this, HostsActivity.class));来启动。TextView组件开启的是HTTP数据文件管理界面，通过startActivity(new Intent(MainActivity.this, FileActivity.class));来启动。

5.3 显示局域网主机HostsActivity的实现

5.3.1 扫描局域网功能的实现

要获取局域网中活跃的主机，我们可以通过遍历子网下的所有IP地址，并向每一个IP地址发送数据报文，这样该子网下的所有IP-MAC地址映射都会保存在本地ARP缓存表中，我们可以定时读取ARP缓存表来获取活跃的主机信息。要实现这样的功能我们需要实现两个技术，第一个是向每个IP地址发送数据报文和读取动态读取ARP缓存表。

本应用发送数据报文采用的是发送UDP报文，UDP协议全称是用户数据报协议，是一种无连接不保证可靠的协议，工作在传输层上，由于UDP协议是无连接的，所以它具有资源消耗小，处理速度快的优点。JAVA中要发送UDP报文可以通过创建java.net.DatagramSocket类的对象来实现，其使用代码片段如下：

```
DatagramSocket socket = new DatagramSocket();
DatagramPacket packet = new DatagramPacket(message.getBytes(), message.length(), ip,
137);
socket.setSoTimeout(200);
socket.send(packet);
socket.close();
```

其中创建DatagramPacket类的对象来构造发送的数据包，第一个参数是数据的byte数组，第二个参数是数据的长度，第三个参数是接收数据包的目标主机IP地址，第四个参数是发送给目标主机的哪个端口。数据包构造好后，通过DatagramSocket类型的对象的send(packet)方法来发送报文，在发送之前可以通过setSoTimeout(200)来设置如果目标主机200毫秒都没有收到数据包就强制断开连接。

向子网发送UDP报文的算法流程图：

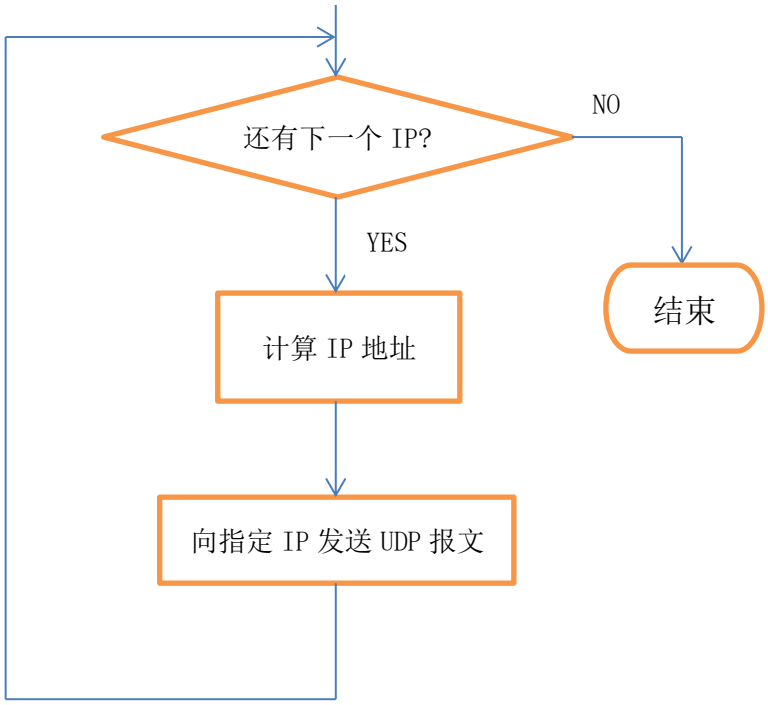

图 5.3 发送 UDP 报文

在Android系统中ARP缓存表的内容会保存在/proc/net/arp文件里，我们可以读取该文件来获取IP-MAC地址映射记录，该文件的内容格式如下：

IP address	HW type	Flags	HW address	Mask	Device

IP address 是主机的IPv4地址，HW type 是地址的硬件类型，Flags是arp 结构体定义的值，HW address 是IPv4地址对应的MAC地址，Device是网卡接口的名称。在读取文件时读取完一行就用这行正则表达式匹配字符串来获取每一项的内容，正则表达式如下：
"^([\\d]{1,3}\\.{3}[\\d]{1,3})\\s+([0-9-a-fx]+)\\s+([0-9-a-fx]+)\\s+([a-f0-9]{2}:{5}[a-f0-9]{2})\\s+([^\\s]+)\\s+(.+)$" 匹配之后需要判断Device项与本机使用的网卡接口名称相等且HW address不等于 00:00:00:00:00:00（无效地址，没有连接到局域网）才是有效的记录，如果IP地址项存在于先前读取的记录里面则不添加到结果中，算法流程图如5.4所示：

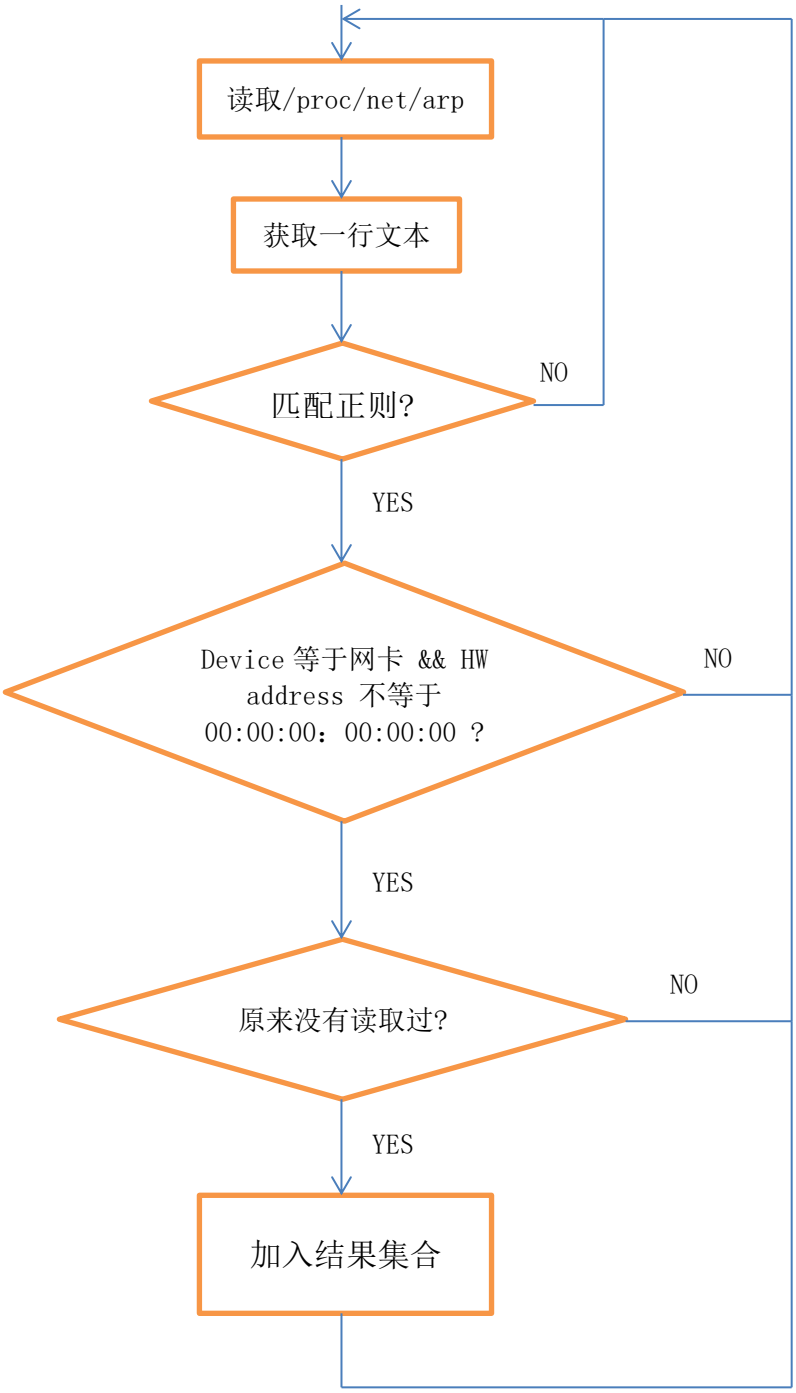

图 5.4 获取局域网活动主机

5.3.2 HostsActivity的设计

HostsActivity是显示当前局域网中活动主机的界面，使用的UI组件是ListView组件，ListView是一个列表显示的组件，可以显示多个具有共同性质的对象，在HostsActivity中对象是主机Host，其类的定义如下：

类名：LanHost
属性：IP 地址、MAC 地址
方法：属性的 setter 和 getter 方法

要把主机对象集合用ListView组件显示，需要建一个继承自BaseAdapter的类，我们定义该类如下：

类名：HostsAdapter
父类：BaseAdapter
属性：主机集合 List<LanHost>、上下文 Context
方法：重写父类 BaseAdapter 方法

在HostsAdapter类中getView方法是最主要的，其形式如下：

```
public View getView(int position, View convertView, ViewGroup parent) {}
```

position是该对象在集合中的索引，convertView是ListView组件中的每一个对象对应的View组件，我们需要为ListView组件中的对象集合中的每个对象设置统一的样式，convertView使用该样式来显示对象属性，ListView包含很多这样相同的convertView。往ListView中添加项可以通过如下代码

```
ListView hostListView = (ListView)findViewById(R.id.host_listview);
List<LanHost> mHosts = new ArrayList<LanHost>();
HostAdapter hostAdapter = new HostsAdapter(mHosts, this);
hostListView.setAdapter(hostAdapter);
```

添加集合后，我们有时还要动态的添加和删除某个项，这时我们需要先修改对象集合，然后调用BaseAdapter的notifyDataSetChanged()方法来动态刷新ListView，添加一个显示主机项的代码片段如下：

```
mHosts.add(new LanHost());
hostAdapter.notifyDataSetChanged();
```

在动态显示局域网中活动的主机信息后，还要对每个主机信息项添加单击事件的监听，ListView组件提供了setOnItemClickListener(new OnItemClickListener(){})方法来实现监听每一个项的单击事件。事件触发后，需要把点击项对应的LanHost局域网主机对象赋给公用AppContext的目标主机引用，并通过startActivity(Intent intent)方法来启动攻击方式选择界面，代码如下：

```
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    LanHost target = (LanHost)parent.getItemAtPosition(position);
    AppContext.setTarget(target);
    startActivity(new Intent(HostsActivity.this, MitmActivity.class));
}
```

5.4 MitmActivity 攻击方式选择界面的实现

本界面主要是显示三种攻击方式给用户，有断网攻击、数据嗅探和HTTP数据劫持，用到的UI组件只有TextView组件，添加了单击事件监听，用户点击某个功能就会启动相应的攻击方式界面，这三个功能启动的对应界面如下：

断网攻击：BrokenNetworkActivity
数据嗅探：SniffActivity
HTTP 数据劫持：HijackActivity

在启动某个攻击界面时，应当先判断原来该攻击有没有启动，如果原来启动过，应该先关闭在重新打开，这样可以保证一段时间内只有一个目标主机被攻击，从而不会发生线程过多和次序乱掉。

5.5 断网攻击的实现

5.5.1 ArpService的实现

Service是Android的服务组件，用来后台执行某些任务，在本应用中ArpService是一个运行在后台持续向目标主机发送伪ARP应答报文的服务。由于JAVA提供了系统运行环境的获取方法，本应用采用了通过JAVA来执行linux版本的可执行程序arpspoof来欺骗目标主机，具体算法流程图如下：

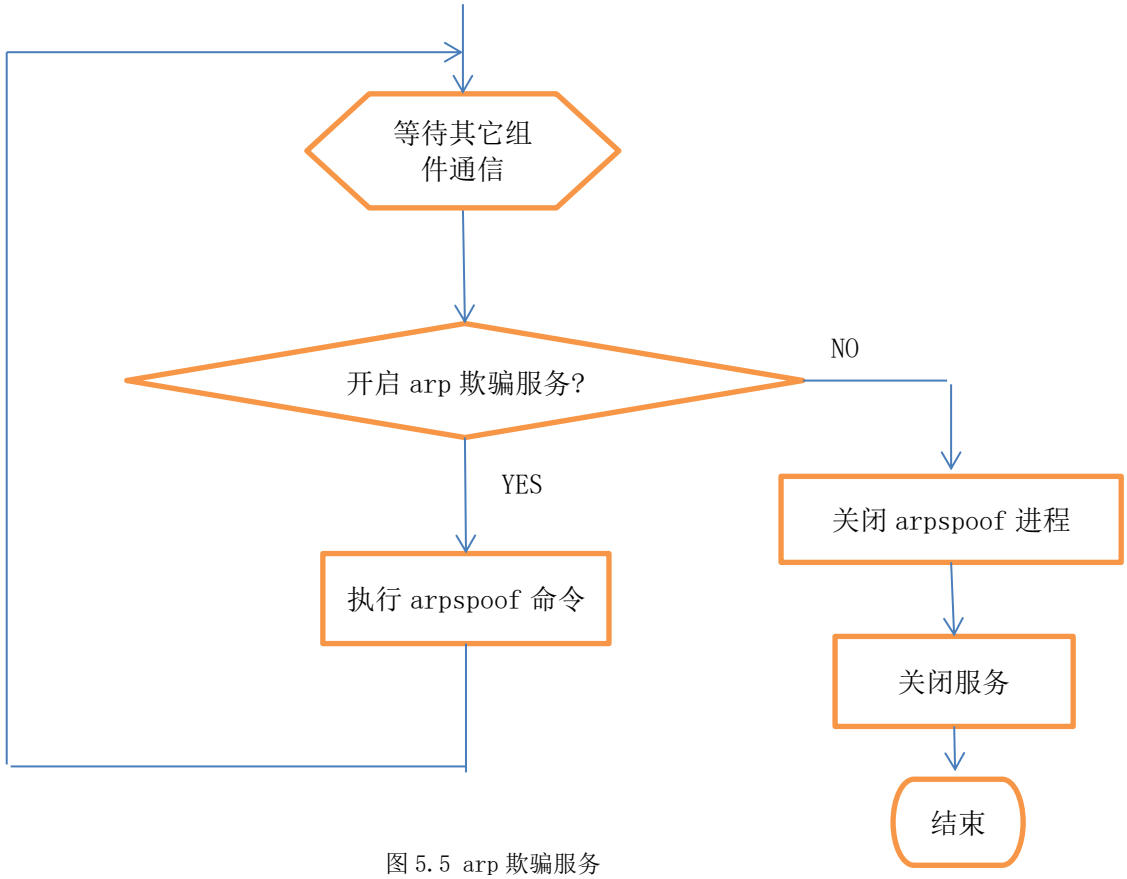

图 5.5 arp 欺骗服务

上图5.5是arp欺骗服务开启的基本流程，开始时等待其它组件对该服务发送启动或停止信号，当为启动信号时开启欺骗服务并等待关闭信号，当接收到关闭停止信号时关闭arpspoof在终端的进程并关闭服务。断网攻击用到了arpspoof -i［网卡接口名称］-t［目标主机IP地址］［网关IPv4地址］命令，关闭arpspoof进程用到了killall arpspoof 命令。在执行完攻击命令后不应该把终端进程关掉，这样可以一直向目标主机发送伪造的ARP报文。执行完关闭攻击程序的命令后，应该在执行exit命令来把当前终端进程关闭，这样可以释放占用的资源。

5.5.2 BrokenNetworkActivity断网攻击界面的实现

BrokenNetworkActivity界面主要用到了两个UI组件，分别是TextView和SwitchButton。TextView组件用来显示被攻击目标主机的IP地址和MAC地址，这样可以直观的显示被攻击主机的信息。SwitchButton组件需要添加开关改变的监听事件，当开关打开时启动arp欺骗服务，断开时关闭arp欺骗服务，流程图如图5.6所示。

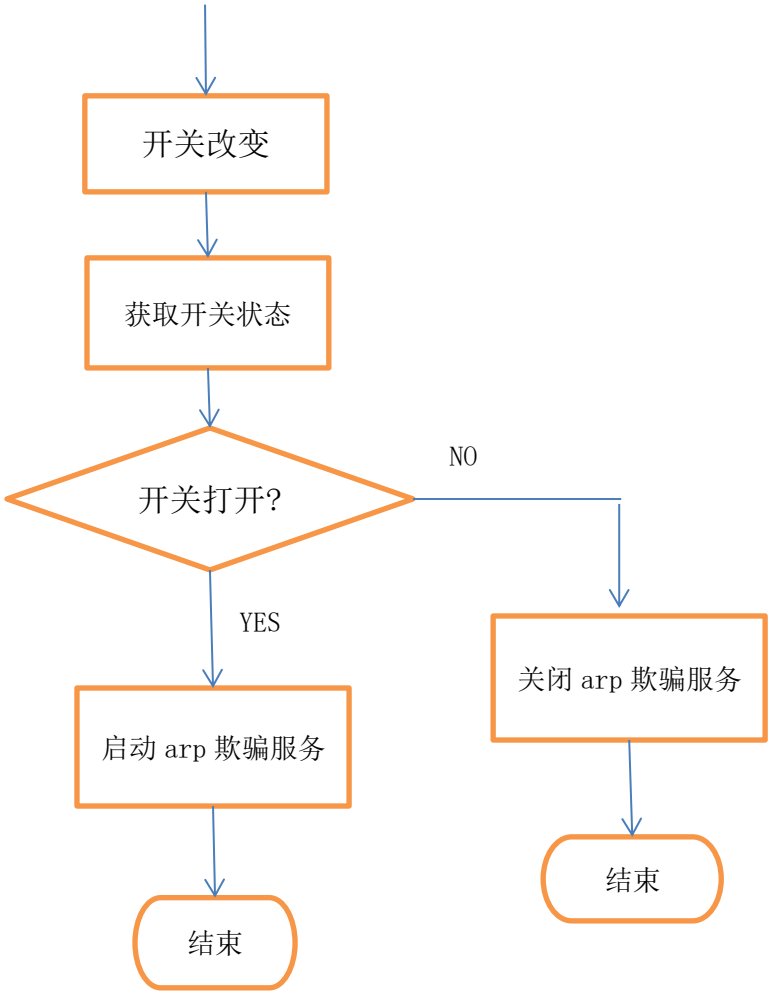

图 5.6 arp 欺骗服务的开启

上图5.6描述了监听开关的过程，当开关变化时会触发我们预先设定好的事件，先获取开关的状态，如果是开的则启动服务，否则关闭服务。

5.6 数据嗅探的实现

5.6.1 SniffService数据嗅探服务的实现

数据嗅探服务是运行在后台并秘密的截获目标主机与外界的通信数据，它继承于ArpService并实现了一些自己的功能，主要添加有以下功能：

1、开启IP转发：arp欺骗服务只是告诉了被攻击的目标主机我是网关，但是没有开启转发功能。目标主机把数据发送给本机，本机没有开启转发，不能把被攻击主机的数据发往外网，目标主机就不能上网，这和数据嗅探功能不符，所以要开启IP转发功能。Android系统底层是linux系统，所以其开启IP转发和在linux下一样。IP转发默认是不开启的，其配置文件是在/proc/sys/net/ipv4/ip_forward 和 /proc/sys/net/ipv6/conf/all/forwarding 分别代表的是IPv4和IPv6。文件中的内容可以为0和1，为0时表示不开启IP转发，不能转发收到的数据包，为1时表示开启了IP转发，本机就像一个路由器一样可以转发数据包。在开启IP转发后，还要设置防火墙

来真正的开启，iptables是linux下非常强大的防火墙命令。开启IP转发的命令格式如下：

```
"iptables -t nat -F",
"iptables -F",
"iptables -X",
"iptables -P FORWARD ACCEPT"
```

其中-t 参数表示对哪个表操作， -F 参数表示清空规则链上的规则，“iptables -t nat -F”表示清空 nat 表的所有规则，-X 参数表示删除用户添加的规则链。-P 参数表示设置默认策略的，有 DROP 和 ACCEPT 两种，分别是默认关和默认开。FORWARD 是网络数据包经服务器路由策略，相当于数据包转发，“iptables -P FORWARD ACCEPT”表示网络数据包转发策略默认打开，这样就完成了转发的开启。

2、抓包：通过 tcpdump 命令来抓取目标主机与外界的网络通信数据包，如果没有对目标主机进行 ARP 欺骗则我们不能抓取目标主机的所有通信数据，只有对目标主机欺骗告诉它本机是网关并欺骗网关告诉它本机是目标主机，这样目标主机和外界的通信数据都会流过本机从而被秘密地截取到。数据嗅探功能就是把目标主机与外界的通信数据截取并保存为 pcap 格式的文件，用到的终端命令如下：

```
tcpdump -w filename host target
```

其中 finename 是数据文件要保存的路径和文件名，target 是目标主机的 IP 地址。数据嗅探服务的流程图如下图所示：

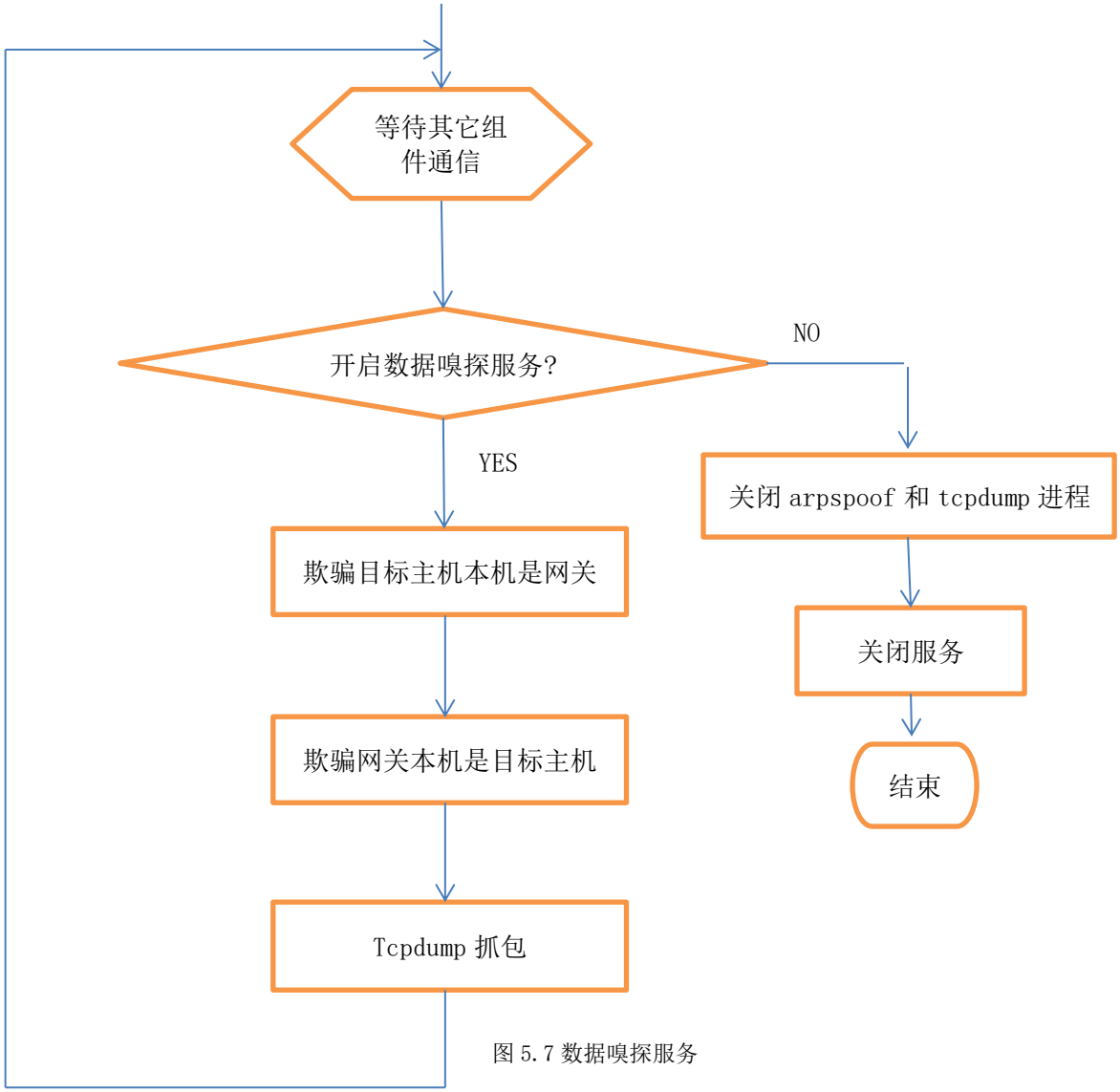

图 5.7 数据嗅探服务

上图 5.7 显示了数据嗅探服务内部的运行基本流程，先等待其它组件的通信，当获取到启动信号时启动数据嗅探服务，关闭信号时关闭服务。服务启动时先分别向目标主机和网关发送伪造的 ARP 应答报文来告诉目标主机我是网关和告诉网关我是目标主机，然后执行 tcpdump 的抓包命令来秘密的截取数据。关闭服务前要先把 arpspoof 和 tcpdump 命令的进程关闭，这样是为了保证程序的可靠性，不能让程序表面上看来服务已经关闭，实际上还有终端命令在系统中运行，既能释放系统资源又能提高效率和可靠性。

5.6.2 SniffActivity 数据嗅探界面的实现

SniffActivity 界面同样用到了两个 UI 组件，分别是 TextView 和 SwitchButton。TextView 组件用来显示被攻击目标主机的 IP 地址和 MAC 地址，这样可以直观的显示被攻击主机的信息。SwitchButton 组件需要添加开关改变的监听事件，当开关打开时启动数据嗅探服务，断开时关闭数据嗅探服务。

5.7 HTTP 数据劫持的实现

5.7.1 ProxyService 数据劫持服务的实现

HTTP 数据劫持服务是在数据嗅探服务上对数据进行过滤处理并保存 HTTP 数据的后台服务。它把数据嗅探服务的 tcpdump 命令改变了，改成只截取目标主机与外界的 HTTP 通信数据，命令形式如下：

```
tcpdump -A -s 0 -n 'host target and tcp and (tcp[20:2]=0x4745 or tcp[20:2]=0x4854)'
```

其中-A 参数表示以 ASCII 输出到控制台；-s 表示单个数据包截取多长，为 0 表示全部截取；-n 参数表示禁用域名解析，我们只要 IP 地址；host 参数表示要抓取的目标主机；tcp 表示只抓取 tcp 协议的数据，tcp[20:2]=0x4745 表示 HTTP GET 请求方法，tcp[20:2]=0x4854 表示 HTTP POST 请求方法；这样我们就可以获取目标主机的 HTTP 中的 GET 和 POST 请求的数据包，并获取控制台的输出流来对数据进行解析。在对数据包进行解析时，我们应该先定义好数据包的数据结构，HTPP 数据包的数据结构定义如下：

表 5.1 HTTP 数据结构

类名	字段	类型	描述
HttpPacket	time	String	数据包发送时间、源 IP 以及目的 IP 信息
	path	String	HTTP 请求路径
	paths	Set<String>	该域名下的 HTTP 请求路径集合
	method	String	HTTP 请求方法
	host	String	HTTP 请求域名
	connection	String	HTTP 通信长时间的处理
	user_agent	String	客户端浏览器信息
	accept_encoding	String	浏览器支持的编码类型
	accept_language	String	浏览器支持的语言
	accept_charset	String	浏览器支持的字符集
	cookie	String	浏览器的 cookie

上表是 HTTP 数据的数据结构，在定义好数据结构后，我们按行读取 tcpdump 命令的输出流并通过正则表达式来知道是哪个字段的信息，每个数据包通过正则匹配 time 信息来分割开，下面是一些字段的正则匹配表达式：

表 5.2 HTTP 字段对应的正则表达式

字段	正则表达式
time	"^(\\d{2}:){2}\\d{2}\\.(.*?)"
path	"(.*?)(GET|POST)(.*?)HTTP/1.1(.*?)"
host	"^(Host:).*?"
connection	"^(Connection:).*?"
user_agent	"^(User-Agent:).*?"
accept_encoding	"^(Accept-Encoding:).*?"
accept_language	"^(Accept-Language:).*?"
accept_charset	"^(Accept-Charset:).*?"
cookie	"^(Cookie:).*?"

上表 5.2 展示了通过正则表达式来区分并获取 HTTP 数据的一些字段信息，其中最重要的信息就是 cookie 信息，cookie 里保存了用户浏览网页时保存的重要信息。在对数据的保存上，我采用了 JSON 文件的保存方式，JSON 是一种轻量级的数据交换格式并且不受开发语言的限制，它的层次分明，易于理解和观看，能够存储所有的数据类型，采用 Key-Value 的存储方式方便了存储和查找。在对 JSON 处理库的选择上，我选择了阿里的 fastjson 处理库，fastjson 在对 JSON 的解析处理上速度快且能够支持多种 JAVA 类型，比如 List 和 Map。下图展示了 HTTP 数据劫持服务内部的基本流程：

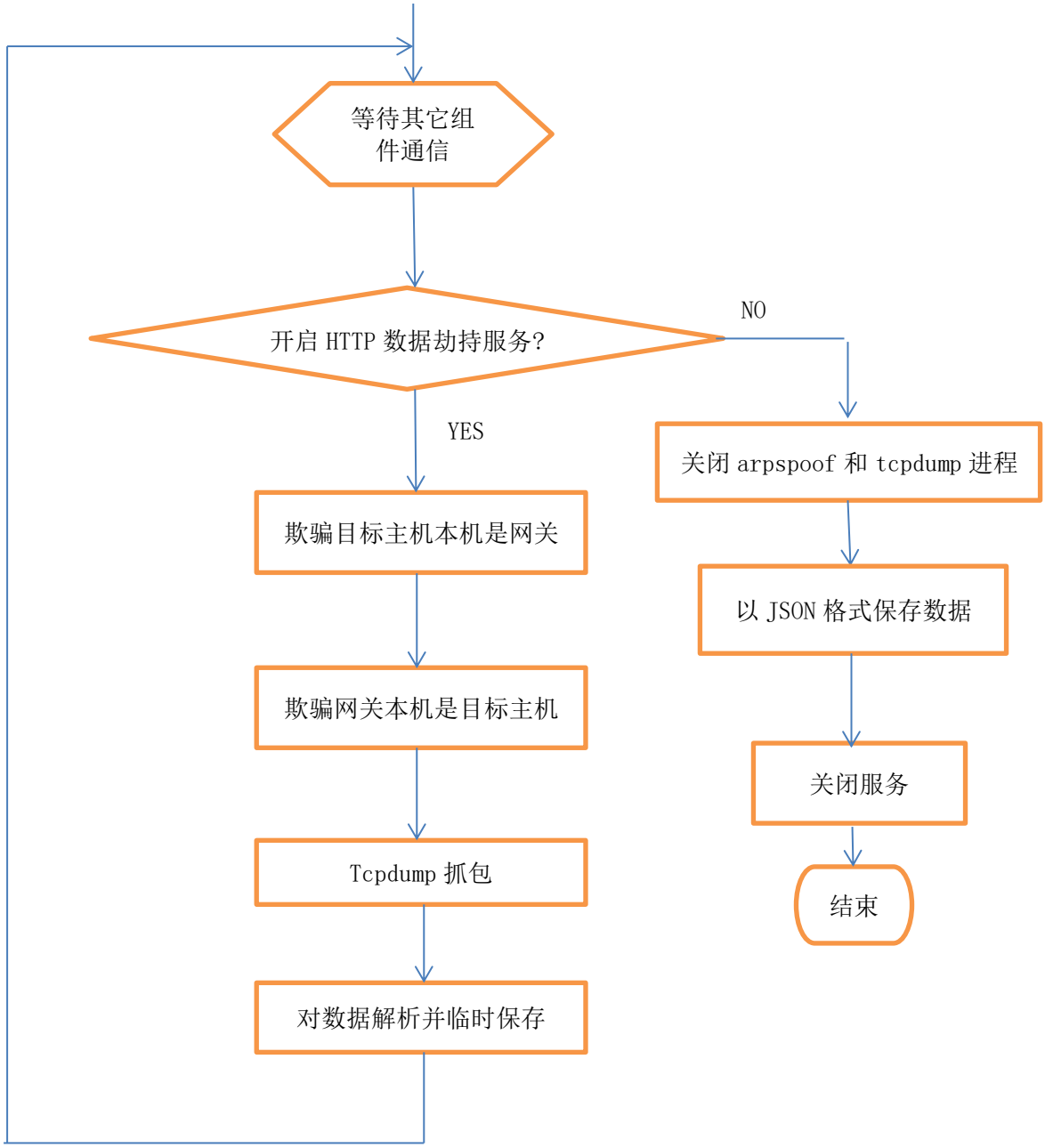

图 5.8 HTTP 数据劫持服务

5.7.2 HijackActivity HTTP 数据劫持界面的设计

HijackActivity 界面同样用到了两个 UI 组件，分别是 TextView 和 SwitchButton。TextView

组件除了被用来显示被攻击目标主机的 IP 地址和 MAC 地址外，还实时的显示了 tcpdump 命令的控制台输出结果，可以实时的了解捕获了什么信息。SwitchButton 组件需要添加开关改变的监听事件，当开关打开时启动 HTTP 数据劫持服务，断开时关闭 HTTP 数据劫持服务。

5.8 HTTP 数据劫持历史管理的实现

5.8.1 FileActivity 数据文件管理界面的实现

FileActivity 是对保存到本地的 JSON 格式的文件进行管理的界面，主要功能就是删除文件和打开查看具体文件内容的界面。用到了 ListView 组件，与 HostsActivity 类似同样需要定义项的公用样式、实体类和 adapter 适配器，具体如下：

表 5.3 数据文件定义

类名	字段	数据类型	描述
DataFile	filename	String	JSON 文件名
	saveDate	Date	文件保存时间

表 5.4 文件适配器定义

类名	父类	字段	数据类型	描述
FileAdapter	BaseAdapter	mFileList	List<DataFile>	数据文件对象集合
		mContext	Context	上下文

在完成上面两个类的定义后就是读取默认目录下的 JSON 格式文件并通过 ListView 展示出来，具体算法流程图如下图 5.9 所示。

开始先获取存储目录下的所有文件和文件夹，然后遍历文件对象集合，只要当前文件对象不是文件夹且文件后缀为.json 则把当前文件对象加入的结果集合中，直到遍历完文件对象集合。获取到文件结果集合后，把文件集合添加到 FileAdapter 中并获取 FileAdapter 对象，在把 FileAdapter 放到ListView组件对象中就可以在界面上显示所有后缀为json的文件信息了。显示文件后给ListView添加对所有项的单击事件来启动查看某个文件具体内容的界面。还添加了长按事件来弹出确认框询问用户是否删除该文件，用户点击是则删除文件，点击否则不删除，算法流程图如图 5.10 所示。

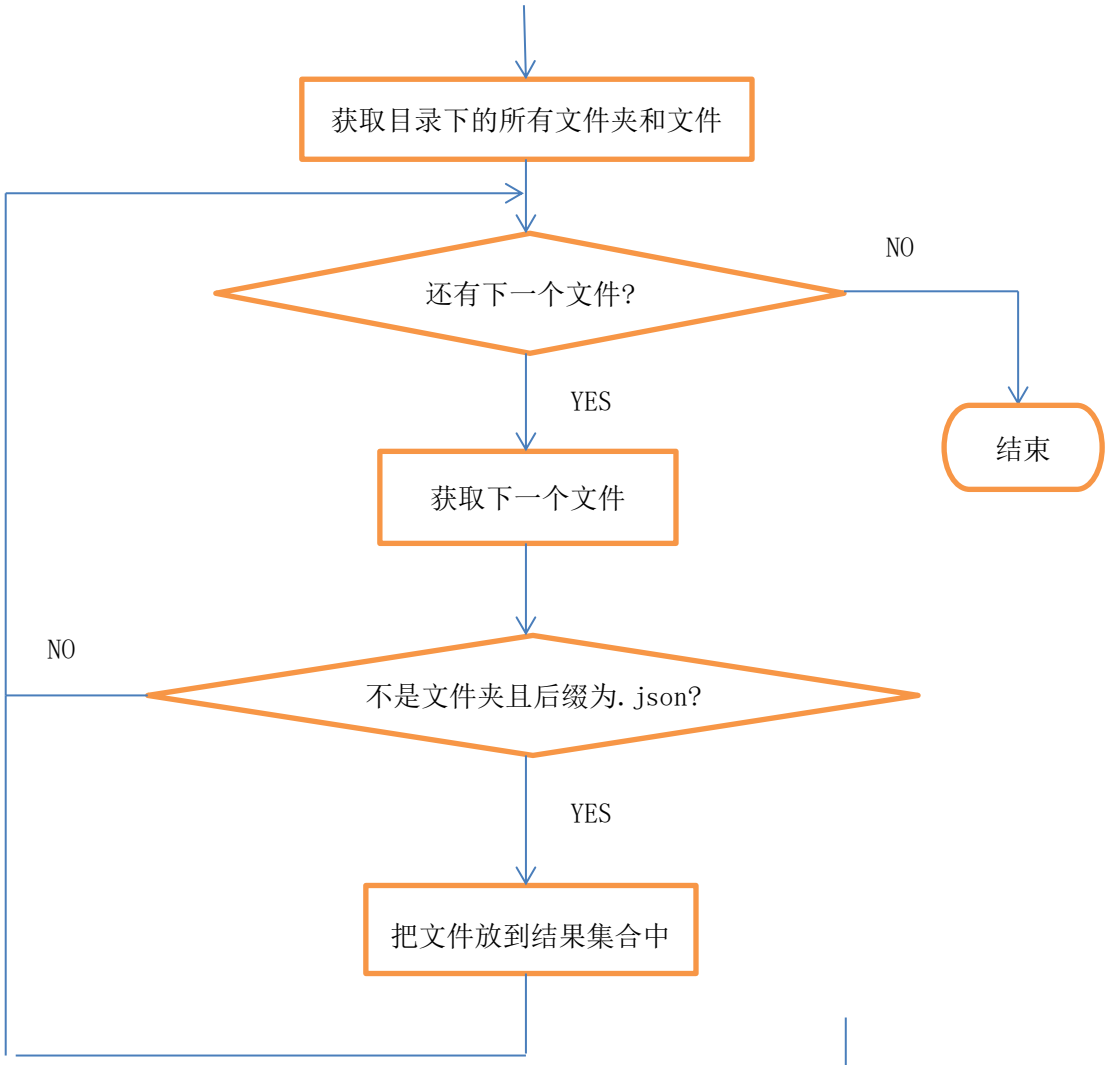

图 5.9 获取 JSON 文件

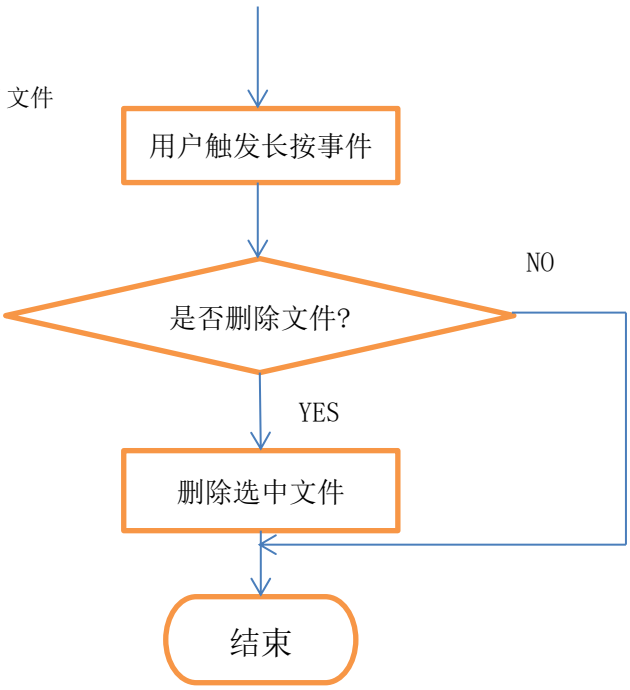

图 5.10 文件删除

5.8.2 HijackHistory 历史记录查看界面的实现

本界面由文件管理界面 FileActivity 启动，用户在文件管理界面点击某项来启动 HijackHistory 界面并通过 Intent 组件把相应的文件信息传递给 HijackHistory，在 HijackHistory 初始化时通过对文件的读取并用 fastjson 来对读取的内容进行解析成 List<HttpPacket>对象，再把 HttpPacket 对象集合放到 ListView 组件中就可以以 Host 来分割文件中的数据并显示。流程图如下所示：

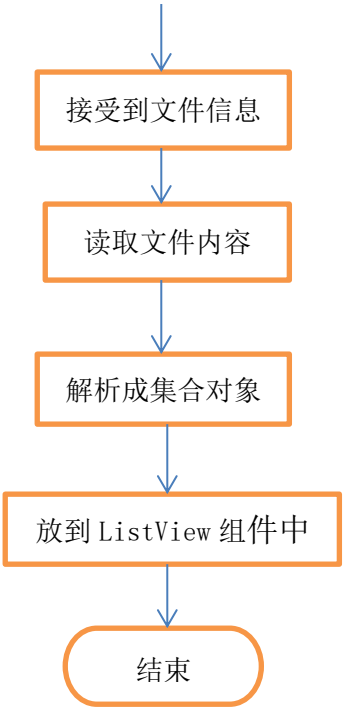

图 5.11 文件内容读取

5.8.3 LookHistory 查看具体信息界面的实现

本界面由 HijackHistory 中用户单击某项来启动，只用到了 TextView 这一个组件，主要就是把截取到目标主机对某个域名访问的 HTTP 请求信息显示出来。

第六章　系统测试

6.1 系统测试环境

在测试本 Android 应用时，采用的是在 VirtualBox 虚拟机中安装 Android x86 系统的方式来搭建测试环境，安装的系统必须是 root 过且添加了 arpspoof 和 tcpdump 命令，我们约定攻击主机为 attacker，被攻击目标主机为 target。

6.2 系统测试流程

在对系统的所有功能测试过程中，需要按照一定的顺序来执行，因为有的功能是在其他功能完成后才能看到效果的，测试的流程图如下所示：

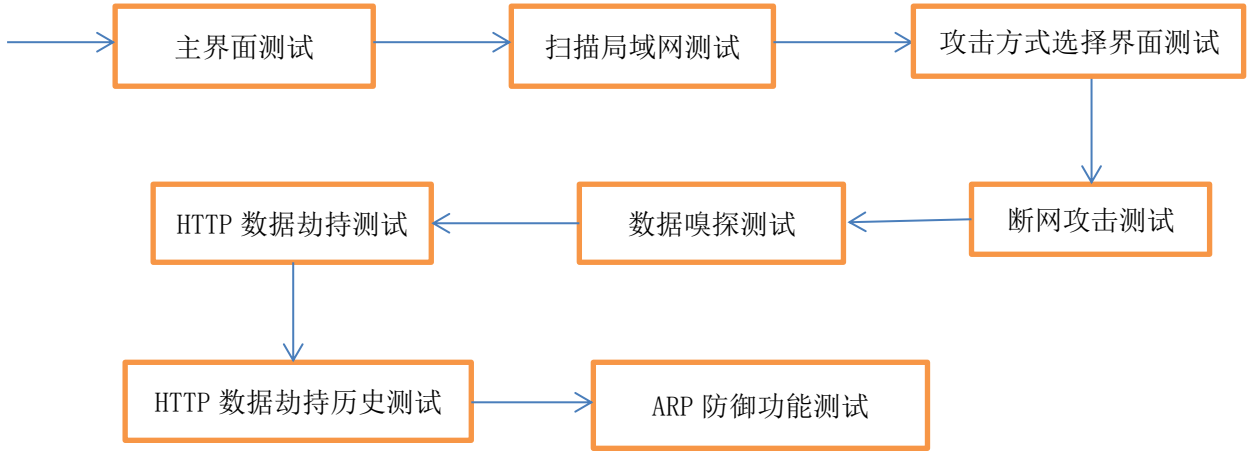

图 6.1 测试流程

6.3 测试用例

用例 ID	用例名称	测试目的	测试环境	前提条件	测试步骤	预期结果
MITMSTU-1	主界面测试	测试应用启动是否正常，主界面是否正常显示	VirtualBox 下 Android-x86 虚拟机	系统已连接网络	1、启动系统 2、配置好网络环境 3、点击应用图标，启动应用	应用不会闪退，界面正常显示
MITMSTU-2	扫描局域网测试	测试扫描局域网功能和显示局域网活动主机信息功能是否正常	Android x86 虚拟机	主界面能够正常显示，系统已 root	1、启动应用 2、单击“扫描局域网”按钮	单击“扫描局域网按钮”后跳到显示主机界面并显示一些主机信息
MITMSTU-3	攻击方式选择界面测试	测试攻击方式选择界面是否正常显示，是否显示三个攻击方式	Android x86 虚拟机	扫描局域网功能正常	1、单击显示主机信息界面中的一个项	能够跳到攻击方式选择界面并正常显示三个攻击方式

MITMSTU-4	断网攻击测试	测试断网功能是否能用	Android x86 虚拟机	攻击方式选择界面正常	1、在attacker主机单击攻击方式选择界面中的“断网攻击” 2、在 target 主机终端执行 ping 百度 3、在attacker主机上开启断网攻击功能 4、在 target 主机终端再 ping 百度	能够跳到断网攻击界面，target 主机第一次能够 ping 通百度，第二次不能 ping 通百度
MITMSTU-5	数据嗅探测试	测试数据嗅探功能是否能用	Android x86 虚拟机	攻击方式选择界面正常	1、在attacker主机单击攻击方式选择界面的“数据嗅探” 2、在attacker主机上开启数据嗅探功能 3、使用target主机上网 4、在attacker主机上关闭数据嗅探功能 5、在attacker主机上查看默认目录下是否有.pcap 文件	能够跳到数据嗅探界面，在开启“数据嗅探”功能后 target 主机仍然可以上网，关闭“数据嗅探”功能后，在默认目录下可以看到新的.pcap 文件
MITMSTU-6	HTTP 数据劫持测试	测试HTTP数据劫持功能是否可用	Android x86 虚拟机	攻击方式选择界面正常	1、在attacker主机单击攻击方式选择界面的“数据嗅探” 2、在attacker主机上开启HTTP数据劫持功能	能够跳到 HTTP 数据劫持界面，在开启 HTTP 数据劫持功能后 target 主机仍然可以上网，关闭 HTTP

					3、使用target主机上网 4、在attacker主机上关闭HTTP数据劫持功能 5、在attacker主机上查看默认目录下是否有.json文件	数据劫持功能后，在默认目录下可以看到新的.json文件
MITMSTU-7	HTTP数据劫持历史管理测试	测试HTTP数据劫持历史管理功能是否正常可用	Android x86虚拟机	主界面正常启动	1、在attacker主机主界面上单击HTTP数据劫持历史项 2、在文件管理界面长按文件项，并做删除和取消两个操作 3、在文件管理界面单击文件项 4、在显示文件内容界面单击域数据包项	能够显示默认目录下的.json文件信息。删除操作可以删除文件，取消操作无任何变化。单击文件项可以正常列出该文件内容并以host分割。单击数据包项可以显示具体信息。
MITMSTU-8	ARP防御功能测试	测试ARP防御功能是否可用	Android x86	主界面正常启动	1、在target主机上启用ARP防御功能 2、在attacker主机上启用断网攻击功能 3、在target主机上网	Target主机仍然可以上网。

6.4 测试结果

本次测试是功能测试，主要对系统的业务逻辑及功能进行测试。在测试过程中严格按照测试用例来进行，按时完成了所有的测试，对系统进行了完整的测试。在测试执行中有好多功能都没能够正常运行，有界面显示不友好、事件不能触发和运行抛出异常等问题，通过对每个功能都进行多次测试和修改，最后所有的功能基本都可以正常运行。

结束语

对课题《安卓中间人攻击的研究与实现》的研究，是我第一次独立的完成一个软件项目，在软件的开发和测试当中学到了很多原来没有接触到知识，对大学四年学到的知识有了升华和凝练。在实现过程中碰到了很多从来没有接触到的问题，有对新知识的不熟悉、搭建测试环境的难题和对逻辑处理等方面的问题。在解决这些问题的过程中，让我学会了用什么方式才能高效快速的解决问题，比如先要明确问题是什么和什么原因导致的，然后可以从问题的源头来解决或者换个思路来实现某个功能。独立完成一个完整的项目对我的综合能力也有了大幅度的提高，同时学会了合理的分配时间以及约束自己的习惯，这几个月来是我进步最快的时间。

最后，感谢路老师对我毕设的悉心指导，同时也感谢大学四年来所有老师对我的栽培和关怀。

参考文献

[1] Mayank, Aggarwal, 小小杉. 基于智能手机设备的中间人攻击技术[J]. 黑客防线, 2010(3):11-15.

[2] 杨萍, 李杰. 基于 ARP 欺骗的中间人攻击的分析与研究[J]. 计算机时代, 2007(5):26-28.

[3] 卿斯汉. Android 安全研究进展[J]. 软件学报, 2016, 27(1):45-71.

[4] 曹刚. 解析中间人攻击原理[J]. 计算机与网络, 2014(22):48-48.

[5] 刘桂泽，耿琛. 无线网络的中间人攻击研究[J]. 《信息安全研究》, 2016, 2(4):361-366.

[6] 郭浩，郭涛. 一种基于 ARP 欺骗的中间人攻击方法及防范[J]. 《信息安全与通信保密》, 2005(10):66-68.

[7] 刘衍斌，王岳斌，陈岗. 基于 ARP 欺骗的中间人攻击的检测与防范[J]. 《微计算机信息》, 2012(8):136-138.

[8] 储柱学，黄莺. 局域网下基于 ARP 欺骗的中间人攻击与防御[J]. 《佳木斯教育学院学报》, 2012(8):428-428.

[9] Yvo Desmedt. Man-in-the-Middle Attack[J]. Encyclopedia of Cryptography & Security, 2011:759-759.

[10] Sounthiraraj D, Sahs J, Greenwood G, et al. SMV-HUNTER: Large Scale,Automated Detection of SSL/TLS Man-in-the-Middle Vulnerabilities in Android Apps[C]// Network and Distributed System Security Symposium. 2014.