北京信息科技大学

（楷体1号居中）

**毕业设计（论文）**

（楷体初号加粗居中）

**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：**计科1302 **班级/学号** 2013011172

**指导老师/督导老师：** 路旭强

**起止时间：**2007年 月 日 **至** 2007年 月 日

（以上宋体四号居中）

**摘 要**

（宋体、三号、加粗、居中）

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。

（宋体、五号）

**关键词：**（3~8个词）中间人攻击； ARP欺骗； 局域网安全 ； 安卓；

数据劫持；

注：**关键词：**（宋体、五号、加粗） ；□□□□□□（宋体、五号）

**Abstract**

(Times New Roman、16磅、加粗、居中)

（Times New Roman 、12磅）

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:** ( 3~8 words) **Man-in-the-middle attack; MITM; ARP spoofing; Android; Data hijacking; LAN security;**

注：**Keywords:** 12磅加粗 ； □□□□□□ 12磅

**目 录**

（宋体、三号、加粗、居中）

[**摘要** （摘要） …………………………………………………………………………I](#摘要)

[（Abstract）……………………………………………………………………… II](#Abstract)

**第一章** 概述 …………………………………………………………………………

1.1 □□□□□□□□□□□□□□□□ ……………………………………

1.2 □□□□□□□□□□□□□□□□ ……………………………………

**第二章** □□□□□□□□□□□□□□□□ ……………………………………

2.1 □□□□□□□□□□□□□□□□ ……………………………………

2.2 □□□□□□□□□□□□□□□□ ……………………………………

∶

∶

**第X章** □□□□□□□□□□□□□□□□ …………………………………

**结束语** …………………………………

**参考文献** …………………………………

注：1 摘要、章、结束语、参考文献：（宋体、小四、加粗、居左）；

2 节：（宋体、五号）；

3 中文摘要、英文摘要、目录的页码依次编排用“Ⅰ、Ⅱ、Ⅲ …”罗马数字（宋体、小五号、居中、底部）。

**第一章 概述**（宋体、三号、加粗、居中）

本章主要（宋体、五号）

**1.1 什么是中间人攻击和ARP欺骗**（宋体、小四、加粗）

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。

（宋体、五号）

**1.2**  **ARP协议的工作原理和ARP欺骗的原理**（宋体、小四、加粗）

**1.2.1 ARP工作原理**（宋体、五号、加粗）

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。

主机A的ARP缓存表

更新ARP缓存

回复主机A的ARP请求

发送ARP请求

直接发送数据

查看IP地址和MAC地址的映射

无

有

**图1 局域网ARP工作原理**

图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**（宋体、五号、加粗）

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。

IP：192.168.0.2

MAC：02-02-02-02-02-02

IP：192.168.0.3

MAC：03-03-03-03-03-03

图2 ARP欺骗原理

我是192.168.0.3

我的MAC地址是

02-02-02-02-02-02

我是192.168.0.1

我的MAC地址是

02-02-02-02-02-02

ARP缓存表

192.168.0.3 02-02-02-02-02-02

ARP缓存表

192.168.0.1 02-02-02-02-02-02

IP：192.168.0.1

MAC：01-01-01-01-01-01

上图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。

（宋体、五号）

**1.3 本课题的主要原理和功能**

**1.3.1 课题的主要原理**

**1.3.2 课题的实现的功能**

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**

（行距：最小值，18磅）

注：1论文正文的页码依次编排用“1、2、3 …”数字（宋体、小五号、居中、底部）

1. **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](http://www.oracle.com/technetwork/java/javase/downloads/index.html)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

;%JAVA\_HOME%\bin

添加后选择PATH变量，追加

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](https://www.eclipse.org/downloads/)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](https://www.virtualbox.org/)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](http://www.android-x86.org/download)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 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，

dhcpcd eth0

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

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

命令的基本用法为：

1. **系统设计**

**3.1系统功能图**

主界面

扫描局域网

ARP防御

查看HTTP数据劫持历史

显示局域网存在的主机

断网攻击

数据嗅探

数据文件管理

详细查看数据

HTTP数据劫持

图3 系统功能图

上图3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**

启用ARP防御

查看HTTP数据劫持历史

扫描局域网

开关

**3.2.2 显示局域网中活动的主机**

IP：

MAC：

IP：

MAC：

局域网中的主机

**3.2.3 选择攻击方式**

断网攻击

数据嗅探

HTTP数据劫持

**3.2.4 攻击功能界面**

攻击方式

开关

**3.2.5 查看HTTP劫持历史**

文件名和日期

文件名和日期

数据项

数据项

数据的具体内容

**3.3 包图**

com.github.fdxxw.mitmstu.activity

com.github.fdxxw.mitmstu.entity

com.github.fdxxw.mitmstu.common

com.github.fdxxw.mitmstu.adapter

com.github.fdxxw.mitmstu.service

com.github.fdxxw.mitmstu.utils

图4 系统包结构

图4 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。

1. **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

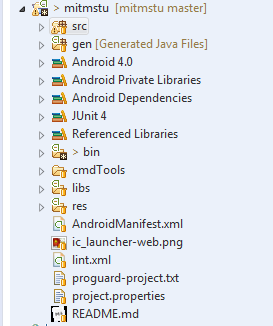
Intent虽然不是Android系统的大组件，但是它是连接Activity组件和Service组件的桥梁，是用来协调组件之间的交互和通讯，在本课题中Intent组件主要被用来从一个Activity屏幕跳到另一个屏幕，在一个Activity中启动一个Service以及在一个Service中启动另一个Service。

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**



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。

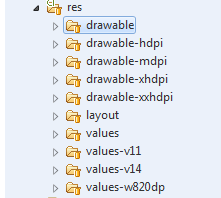
bin目录：该目录下主要是存放应用编译后的文件，包括打包后的\*.apk可安装的Android应用程序。

lib目录：该目录

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。

**4.2.1.1 res目录介绍**



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

**4.2.2 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。

1. **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**

入口类的基本参数 本机IP地址获取的算法流程图：

YES

YES

YES

有下一个网卡?个

有下一个IP?

获取网卡的所有IP

获取IP

正确IPv4?

结束

NO

NO

结束

NO

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

上图展示了获取本机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：

获取开关状态

开关打开?

YES

NO

执行arp –s 网关IP 网关MAC

结束

arp –d 网关IP地址

结束

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：

计算IP地址

还有下一个IP?

向指定IP发送UDP报文

结束

YES

NO

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，算法流程图如下所示：

读取/proc/net/arp

获取一行文本

匹配正则?

Device等于网卡 && HW address 不等于 00:00:00：00:00:00 ?

原来没有读取过?

加入结果集合

NO

NO

NO

YES

YES

YES

**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**