前面我们介绍的都是爬取 Web 网页的内容。随着移动互联网的发展，越来越多的企业并没有提供 Web 网页端的服务，而是直接开发了 App，更多更全的信息都是通过 App 来展示的。那么针对 App 我们可以爬取吗？当然可以。

我们知道 Web 站点有多种渲染和反爬方式，渲染分为服务端渲染和客户端渲染；反爬也是多种多样，如请求头验证、WebDriver 限制、验证码、字体反爬、封禁 IP、账号验证等等，综合来看 Web 端的反爬虫方案也是多种多样。

但 App 的情况略有不同，一般来说，App 的数据通信大都需要依赖独立的服务器，比如请求某个 HTTP 接口来获取数据或做登录校验等。这种通信其实就类似 Web 中的 Ajax，客户端向服务器发起 HTTP 请求，获取到数据之后再做一些处理，数据的格式大多也是 JSON、XML 等，基本不会有 HTML 代码这样的数据。

所以说，对于 App 来说，其核心就在于找到这些数据请求到底是怎样的，比如某次 HTTP POST 请求的 URL、Headers、Data 等等，知道了这些，我们就能用程序模拟这个请求过程，从而就能完成爬虫了。

那么怎么知道 App 到底在运行过程中发起了什么请求呢？最有效且常见的方式就是抓包了，抓包工具也非常多，比如 Fiddler、Charles、mitmproxy、anyproxy 等等，我们用这些工具抓到 HTTP 请求包，就能看到这个请求的 Method、Headers、Data 等内容了，知道了之后再用程序模拟出来就行了。

但是，这个过程中你可能遇到非常多的问题，毕竟 App 的数据也是非常宝贵的，所以一些 App 也添加了各种反爬措施，比如：

- 这个 App 的请求根本抓不到包，原因可能是 App 本身设置了不走系统代理。
- 对一些 HTTPS 的请求，抓包失败，原因可能是系统或 App 本身设置了 SSL Pining，对 HTTPS 证书进行了校验，代理软件证书校验不通过，拒绝连接。
- 某些包即使抓到了，也发现了其中带了加密参数，比如 sign、token 等等，难以直接用程序模拟。
- 为了破解一些加密参数可能需要对 App 进行逆向，逆向后发现是混淆后的代码，难以分析逻辑。
- 一些 App 为了防止逆向，本身进行了加固，需要对 App 进行脱壳处理才能进行后续操作。
- 一些 App 将核心代码进行编译，形成 so 库，因此可能需要对 so 库进行逆向才能了解其逻辑。
- 一些 App 和其服务器对以上所有的流程进行了风控处理，如果发现有疑似逆向或破解或访问频率等问题，返回一些假数据或拒绝服务，导致爬虫难以进行。

随着移动互联网的发展，App 上承载的数据也越来越多，越来越重要，很多厂商为了保护 App 的数据也采取了非常多的手段。因此 App 的爬取和逆向分析也变得越来越难，本课时我们就来梳理一些 App 爬取方案。

以下内容针对 Android 平台。

抓包

对于多数情况来说，一台 Android 7.0 版本以下的手机，抓一些普通的 App 的请求包还是很容易做到的。

抓包的工具有很多，常见的如 Charles、Fiddler、mitmproxy 等。

抓包的时候在 PC 端运行抓包软件，抓包软件会开启一个 HTTP 代理服务器，然后手机和 PC 连在同一个局域网内，设置好抓包软件代理的 IP 和端口，另外 PC 和手机都安装抓包软件的证书并设置信任。这样在手机上再打开 App 就能看到 App 在运行过程中发起的请求了。

抓包完成之后在抓包软件中定位到具体数据包，查看其详情，了解其请求 Method、URL、Headers、Data，如果这些没有什么加密参数的话，我们用 Python 重写一遍就好了。

当然如果遇到抓不到包或者有加密参数的情形，无法直接重写，那就要用到后面介绍的方法了。

抓不到包

一些 App 在内部实现的时候对代理加了一些校验，如绕过系统代理直接连接或者检测到了使用了代理，直接拒绝连接。

这种情形往往是手机的 HTTP 客户端对系统的网络环境做了一些判断，并修改了一些 HTTP 请求方式，使得数据不走代理，这样抓包软件就没法直接抓包了。

另外对于一些非 HTTP 请求的协议，利用常规的抓包软件也可能抓不到包。这里提供一些解决方案。

强制全局代理

虽然有些数据包不走代理，但其下层还是基于 TCP 协议的，所以可以将 TCP 数据包重定向到代理服务器。比如软件 ProxyDroid 就可以实现这样的操作，这样我们就能抓到数据包了。

ProxyDroid：https://github.com/madeye/proxydroid

手机代理

如果不通过 PC 上的抓包软件设置代理，还可以直接在手机上设置抓包软件，这种方式是通过 VPN 的方式将网络包转发给手机本地的代理服务器，代理服务器将数据发送给服务端，获取数据之后再返回即可。

使用了 VPN 的方式，我们就可以截获到对应的数据包了，一些工具包括 HttpCanary、Packet Capture、NetKeeper 等。

- HttpCanary：https://play.google.com/store/apps/details?id=com.guoshi.httpcanary
- Packet Capture：https://play.google.com/store/apps/details?id=app.greyshirts.sslcapture
- NetKeeper：https://play.google.com/store/apps/details?id=com.minhui.networkcapture.pro

以上应用链接来源于 Google Play，也可以在国内应用商店搜索或直接下载 apk 安装。

特殊协议抓包

可以考虑使用 Wireshark、Tcpdump 在更底层的协议上抓包，比如抓取 TCP、UDP 数据包等等。

使用的时候建议直接 PC 上开热点，然后直接抓取 PC 无线网卡的数据包，这样 App 不管有没有做系统代理校验或者使用了非 HTTP 协议，都能抓到数据包了。

SSL Pining

SSL Pining，就是证书绑定，这个只针对 HTTPS 请求。

SSL Pining 发生在下面的一些情况：

- 对于 Android 7.0 以上的手机，系统做了改动，HTTPS 请求只信任系统级别证书，这会导致系统安全性增加，但是由于抓包软件的证书并不是系统级别证书，就不受信任了，那就没法抓包了。
- 一些 App 里面专门写了逻辑对 SSL Pining 做了处理，对 HTTPS 证书做了校验，如果发现是不在信任范围之内的，那就拒绝连接。

对于这些操作，我们通常有两种思路来解决：

- 让系统信任我们的 HTTPS 证书；
- 绕开 HTTPS 证书的校验过程。

对于这两种思路，有以下一些绕过 SSL Pining 的解决方案。

修改 App 的配置

如果是 App 的开发者或者把 apk 逆向出来了，那么可以直接通过修改 AndroidManifest.xml 文件，在 apk 里面添加证书的信任规则即可，详情可以参考 https://crifan.github.io/app_capture_package_tool_charles/website/how_capture_app/complex_https/https_ssl_pinning/，这种思路属于第一种信任证书的解决方案。

将证书设置为系统证书

当然也可以将证书直接设置为系统证书，只需要将抓包软件的证书设置为系统区域即可。但这个前提是手机必须要 ROOT，而且需要计算证书 Hash Code 并对证书进行重命名，具体可以参考 https://crifan.github.io/app_capture_package_tool_charles/website/how_capture_app/complex_https/https_ssl_pinning，这种思路也是第一种信任证书的解决方案。

Xposed + JustTrustMe

Xposed 是一款 Android 端的 Hook 工具，利用它我们可以 Hook App 里面的关键方法的执行逻辑，绕过 HTTPS 的证书校验过程。JustTrustMe 是基于 Xposed 一个插件，它可以将 HTTPS 证书校验的部分进行 Hook，改写其中的证书校验逻辑，这种思路是属于第二种绕过 HTTPS 证书校验的解决方案。

当然基于 Xposed 的类似插件也有很多，如 SSLKiller、sslunpining 等等，可以自行搜索。

不过 Xposed 的安装必须要 ROOT，如果不想 ROOT 的话，可以使用后文介绍的 VirtualXposed。

Frida

Frida 也是一种类似 Xposed 的 Hook 软件，使用它我们也可以实现一些 HTTPS 证书校验逻辑的改写，这种思路也是属于第二种绕过 HTTPS 证书校验的方案。

具体可以参考 https://codeshare.frida.re/@pcipolloni/universal-android-ssl-pinning-bypass-with-frida/。

VirtualXposed

Xposed 的使用需要 ROOT，如果不想 ROOT 的话，可以直接使用一款基于 VirtualApp 开发的 VirtualXposed 工具，它提供了一个虚拟环境，内置了 Xposed。我们只需要将想要的软件安装到 VirtualXposed 里面就能使用 Xposed 的功能了，然后配合 JustTrustMe 插件也能解决 SSL Pining 的问题，这种思路是属于第二种绕过 HTTPS 证书校验的解决方案。

特殊改写

对于第二种绕过 HTTPS 证书校验的解决方案，其实本质上是对一些关键的校验方法进行了 Hook 和改写，去除了一些校验逻辑。但是对于一些代码混淆后的 App 来说，其校验 HTTPS 证书的方法名直接变了，那么 JustTrustMe 这样的插件就无法 Hook 这些方法，因此也就无效了。

所以这种 App 可以直接去逆向，找到其中的一些校验逻辑，然后修改写 JustTrustMe 的源码就可以成功 Hook 住了，也就可以重新生效了。

逆向秘钥

还有一种硬解的方法，可以直接逆向 App，反编译得到证书秘钥，使用秘钥来解决证书限制。

逆向

以上解决了一些抓包的问题，但是还有一个问题，就是抓的数据包里面带有加密参数怎么办？比如一个 HTTP 请求，其参数还带有 token、sign 等参数，即使我们抓到包了，那也没法直接模拟啊？

所以我们可能需要对 App 进行一些逆向分析，找出这些加密过程究竟是怎样的。这时候我们就需要用到一些逆向工具了。

JEB

JEB 是一款适用于 Android 应用程序和本机机器代码的反汇编器和反编译器软件。利用它我们可以直接将安卓的 apk 反编译得到 Smali 代码、jar 文件，获取到 Java 代码。有了 Java 代码，我们就能分析其中的加密逻辑了。

JEB：https://www.pnfsoftware.com/

JADX

与 JEB 类似，JADX 也是一款安卓反编译软件，可以将 apk 反编译得到 jar 文件，得到 Java 代码，从而进一步分析逻辑。
JADX：https://github.com/skylot/jadx

dex2jar、jd-gui

这两者通常会配合使用来进行反编译，同样也可以实现 apk 文件的反编译，但其用起来个人感觉不如 JEB、JADX 方便。

脱壳

一些 apk 可能进行了加固处理，所以在反编译之前需要进行脱壳处理。一般来说可以先借助于一些查壳工具查壳，如果有壳的话可以借助于 Dumpdex、FRIDA-DEXDump 等工具进行脱壳。

- FRIDA-DEXDump：https://github.com/hluwa/FRIDA-DEXDump
- Dumpdex：https://github.com/WrBug/dumpDex

反汇编

一些 apk 里面的加密可能直接写入 so 格式的动态链接库里面，要想破解其中的逻辑，就需要用到反汇编的一些知识了，这里可以借助于 IDA 这个软件来进行分析。
IDA：https://www.hex-rays.com/

以上的一些逆向操作需要较深的功底和安全知识，在很多情况下，如果逆向成功了，一些加密算法还是能够被找出来的，找出来了加密逻辑之后，我们用程序模拟就方便了。

模拟

逆向对于多数有保护 App 是有一定作用的，但有的时候 App 还增加了风控检测，一旦 App 检测到运行环境或访问频率等信息出现异常，那么 App 或服务器就可能产生防护，直接停止执行或者服务器返回假数据等都是有可能的。

对于这种情形，有时候我们就需要回归本源，真实模拟一些 App 的手工操作了。

adb

最常规的 adb 命令可以实现一些手机自动化操作，但功能有限。

触动精灵、按键精灵

有很多商家提供了手机 App 的一些自动化脚本和驱动，如触动精灵、按键精灵等，利用它们的一些服务我们可以自动化地完成一些 App 的操作。

触动精灵：https://www.touchsprite.com/

Appium

类似 Selenium，Appium 是手机上的一款移动端的自动化测试工具，也能做到可见即可爬的操作。

Appium：http://appium.io/

AirTest

同样是一款移动端的自动化测试工具，是网易公司开发的，相比 Appium 来说使用更方便。

AirTest：http://airtest.netease.com/

Appium/AirTest + mitmdump

mitmdump 其实是一款抓包软件，与 mitmproxy 是一套工具。这款软件配合自动化的一些操作就可以用 Python 实现实时抓包处理了。

mitmdump：https://mitmproxy.readthedocs.io/

到此，App 的一些爬虫思路和常用的工具就介绍完了，在后面的课时我们会使用其中一些工具来进行实战演练。

参考来源

- https://zhuanlan.zhihu.com/webspider
- https://www.zhihu.com/question/60618756/answer/492263766
- https://www.jianshu.com/p/a818a0d0aa9f
- https://mp.weixin.qq.com/s/O6iWb2VL4SH9UNLwk2FCMw
- https://zhuanlan.zhihu.com/p/60392573
- https://crifan.github.io/app_capture_package_tool_charles/website/
- https://github.com/WooyunDota/DroidDrops/blob/master/2018/SSL.Pinning.Practice.md