第 3 章

CHAPTER 3

iOS 基础知识

作为移动设备市场的重要组成部分，苹果公司开发的 iPhone 设备也是攻防双方对抗的战场之一。本章将介绍有关 iOS 应用的相关知识，包括 iOS 应用包结构以及 iOS 应用的启动流程。

3.1 iOS 包结构分析


视频讲解


视频讲解

每一个应用程序都有一个载体，载体内部保存着程序的字节码以及程序运行过程中的各种配置文件，所有的文件会被打包在一起，形成一个压缩包，以便于程序的复制分发以及网络传输。

不同平台的程序包文件扩展名以及结构都不一样，例如，Windows 平台的程序包通常是以. exe 为扩展名的可执行文件；Android 系统的程序包则是以. apk 为扩展名的压缩包文件。对于 iOS 系统，iOS 程序包文件是以. ipa 为扩展名的压缩包。

IPA 包文件与 APK 包文件类似，以 ZIP 压缩包格式作为基础。如果将一个 iOS 应用的 IPA 包文件的扩展名修改为. zip，则可以使用解压缩工具打开。IPA 包解压效果如图 3.1 所示。

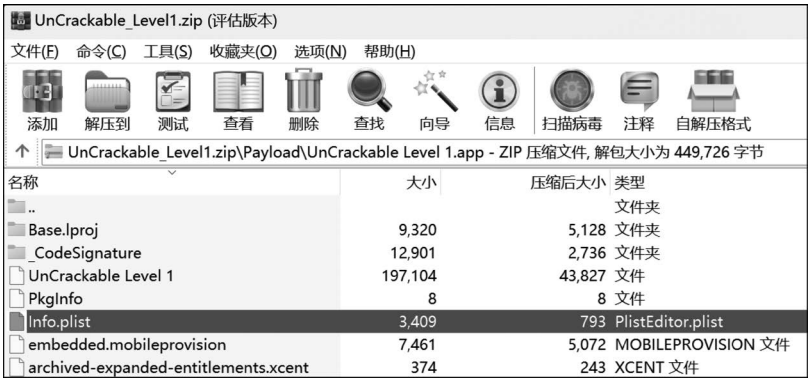

图 3.1 IPA 包解压效果

3.1.1 _CodeSignature 文件夹

_CodeSignature 文件夹内有一个 CodeResources 文件，该文件内包含一个字典，字典的内容是 IPA 包内文件的哈希表。字典的键是文件名，字典的值是 Base64 格式的哈希值。

CodeResources 文件的作用与 Android APK 包内的 META-INF 文件夹类似，用于判断一个应用程序包是否完好无损。如果应用包内部的任意一个文件被篡改，那么文件的哈希值就会发生变化，iOS 系统以 CodeResources 文件作为参照，对包内文件进行校验的时候，就能发现应用包是否经过篡改。

3.1.2 lproj 文件夹

为了给不同地区、不同母语的用户提供服务，iOS 提供了一套应用国际化的机制，具体来说，就是将针对不同国家、不同语言环境的资源文件，分别保存在不同的文件夹下。这些文件夹被统一称为 lproj 文件夹，也可以称为本地化文件夹。

iOS 项目在构建的时候，项目路径下会生成一个默认的本地化文件夹 Base. lproj，如果需要针对某个语言环境进行适配，则需要额外添加 lproj 文件夹，例如，简体中文的资源文件保存在 zh-Hans. lproj 文件夹内，美式英语的资源文件保存在 en. lproj 文件夹内。

3.1.3 xcent 文件

archived-expanded-entitlements. xcent 被称为授权文件，此文件决定 iOS 应用在何种情况下被允许使用何种系统资源。可以将 archived-expanded-entitlements. xcent 看作 iOS 应用沙盒的配置列表。

3.1.4 mobileprovision 文件

iOS 包内以 mobileprovision 为后缀的文件是 iOS 私钥证书与描述文件，开发者需要在苹果开发者官网申请开发证书与发布证书，才能通过证书生成该描述文件。

3.1.5 info. plist 文件

info. plist 文件是 iOS 应用的功能配置文件。如果 iOS 应用需要使用一些功能，则需要在 info. plist 文件中进行配置，比如，iOS 应用后台运行、应用支持读取的文件类型等。

接下来详细介绍 info. plist 文件的内部结构。info. plist 是一个键值对文件。根据功能，iOS 系统提供的键大致可被分为下面 4 类。

第一类是 Core Foundation Keys，此类键的名称通常以 CF 作为前缀，被用来描述一些常用的行为，表 3-1 给出了 Core Foundation Keys 的名称以及功能描述。

表 3-1 Core Foundation Keys 的名称以及功能描述

属　性	名　称	类　型	描　述
CFBundleDevelopmentRegion	Localization native development region	String	本地化相关数据，如果用户没有响应的语言资源，则默认使用这个键的值
CFBundleExecutable	Executable file	String	程序安装包的名称
CFBundleIdentifier	Bundle indentifier	String	唯一标识字符串
CFBundleInfoDictionaryVersion	InfoDictionary version	String	info. plist 格式的版本信息
CFBundleName	Bundle name	String	程序安装后在界面上显示的名称

第二类是 Lanch Services Keys，该类型的键名称通常以 LS 作为前缀，被用来提供应用加载所依赖的配置，描述应用启动的方式。表 3-2 给出了 Lanch Services Keys 的名称以及功能描述。

表 3-2 Lanch Services Keys 的名称以及功能描述

属 性	功 能 描 述
LSBackgroundOnly	该属性取值设置为 1，则启动的服务只会在后台运行
LSRequiresCarbon	该属性取值设置为 1，则启动的服务只会在 Carbon 环境下运行
LSRequiresClassic	该属性取值设置为 1，则启动的服务只会在 Classic 环境下运行
LSUIElement	该属性取值设置为 1，则启动的服务会把应用程序作为一个用户界面组件来运行

第三类是 Cocoa keys，该类型的键名称通常以 NS 作为前缀，iOS 应用的 Cocoa 框架或者 Cocoa Touch 框架会依赖该类的键值标识更高级的配置项目，比如：

- NSPhotoLibraryUsageDescription 属性——声明 App 需要得到用户的同意才能访问相册。
- NSCameraUsageDescription 属性——声明 App 需要得到用户的同意才能使用相机。

第四类是 App Extension Keys，该类型的键被用来扩展默认的 plist，以便描述更多的信息，比如，定义 iOS 应用启动后的默认旋转方向、标识应用是否支持文件共享。

3.2 iOS 应用启动过程分析


视频讲解


视频讲解


视频讲解

iOS 应用的启动过程主要分为 3 个阶段：第一个阶段是 iOS 应用的 main()函数执行之前，第二个阶段是 main()函数执行到屏幕渲染相关方法执行完毕的阶段，第三个阶段是屏幕渲染完成后的收尾阶段。

在第一个阶段中，iOS 系统内核会创建一个进程，然后加载 iOS 应用包内的 Mach-O 可执行文件。Mach-O 可执行文件是应用内所有 Objective-C 字节码文件的集合。接下来 iOS 系统使用动态链接器加载 iOS 应用的动态链接库，然后进行 rebase 指针的调整与 bind 符号的绑定。

接下来，iOS 系统会执行 Objective-C 的 runtime 初始化操作，包括相关类的注册、category 的注册和 selector 唯一性检查。在第一个阶段的最后，iOS 系统会进行初始化操作，包括创建 C++静态全局变量，调用 Object-C 类和分类的＋load 函数、被-attribute-((constructor))属性修饰的函数，这些函数需要在 main()函数开始前被调用。

进入第二个阶段，main()函数执行完毕，进入手机屏幕渲染相关的工作。第二个阶段的流程如图 3.2 所示。

main()函数执行完毕之后，调用 UIApplicationMain()函数，创建一个 UIApplication 对象和属于 UIApplication 的 delegate 对象。接下来加载 info.plist，同时 delegate 对象开始监听系统事件，程序启动完毕后，调用 Application:didFinishLaunchingWithOptions()函数，创建 UIWindow 与 rootViewController 对象。

最后一个阶段将执行完 Application:didFinishLaunchingWithOptions()剩余的代码，具体是指从设置 UIWindow 的 rootViewController，到 didFinishLaunchingWithOptions()

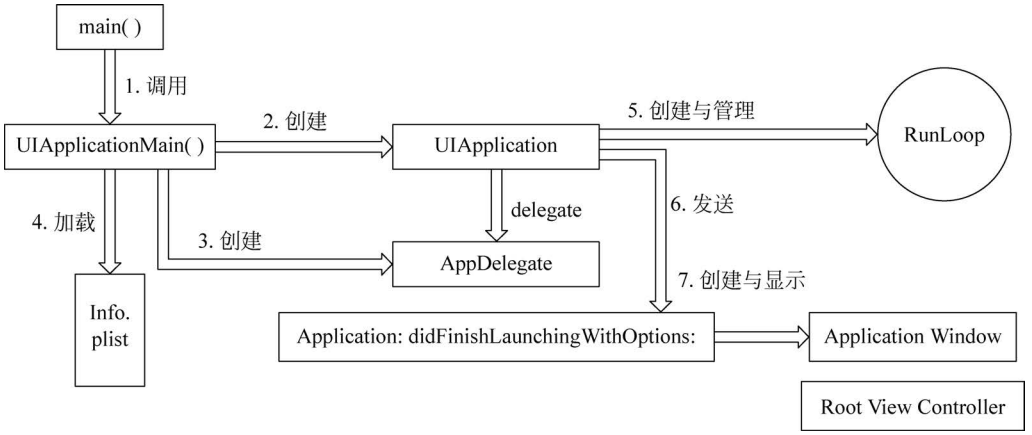

图 3.2　第二个阶段的流程

函数运行结束。

当渲染工作完成后，手机屏幕上就会显示出应用的界面。

3.3　本章小结

本章介绍了 iOS 应用包的结构以及应用启动的流程，由于 iOS 系统的封闭性，大部分逆向人员很难接触到 iOS 系统的底层，因此大部分的攻防活动还是集中在应用的层面上。希望读者通过本章的学习，能够更好地了解 iOS 应用。

理 论 篇

本篇是移动安全攻防进阶的理论基础，共 3 章。

其中，第 4 章主要介绍 ATT&CK 框架的技战术，将攻防过程中的技术点映射到矩阵中；第 5 章重点学习 ATT&CK 框架中移动安全攻防框架；第 6 章重点学习 LLVM 编译框架的编译、用法以及 Pass 程序的编写，作为后续移动应用 Native 层攻防的铺垫。

本篇针对移动安全攻防进阶阶段所涉及的核心理论进行详细讲解，从攻防技战术角度引入 ATT&CK 框架作为全局的理论指导，在原生层的攻防领域着重学习 LLVM 编译框架及二进制逆向，为后续 Native 层逆向、应用加固等攻防技术打下坚实的理论基础。

第 4 章

CHAPTER 4

ATT&CK 框架

在逆向攻防技术的发展过程中，涌现出了许多技战术，一些网络安全社区或者非营利的网络安全组织将这些技战术整合成资料库。这些资料库对逆向攻防的初学者是不可多得的参考资料。本章介绍的 ATT&CK 框架就是一个攻防技术资料库。

4.1 ATT&CK 框架背景介绍

ATT&CK 框架的推出者 MITRE 是美国政府资助的一家研究机构，MITRE 公司自 1958 年从麻省理工学院分离出来后，参与了大量商业与机密项目，在美国国家标准技术研究所的资助下从事了大量的网络安全实践。

2013 年，MITRE 公司正式推出 ATT&CK 框架，该框架根据真实的案例来描述网络安全对抗行为，并进行分类。通过对网络攻击者行为的总结归纳，形成一个结构化的列表，通过矩阵的方式展示出来。

MITRE 公司提出 ATT&CK 框架的目的是创建网络攻击中使用的对抗战术和技术的详尽列表，该列表会对收录其中的每一种技术的使用方式进行详细介绍，同时借助具体的场景实例，向企业的安全人员说明攻击者如何通过某个恶意软件或方案执行网络攻击，以及安全人员应该如何减轻或者检测网络攻击所造成的影响。

ATT&CK 框架采用军事战争中的 TTP(Tactics, Techniques & Procedures)方法论，在攻防双方之间形成了一套标准与通用的交流语言，除了实战价值以外，还具备学术价值。在网络红蓝对抗演习领域，红队(进攻方)可以从 ATT&CK 框架中学习各种攻击战术，充实自己的武器库。蓝队(防守方)可以通过 ATT&CK 框架判断红队的攻击链条。网络安全研究员可以利用 ATT&CK 框架对真实黑客团队的 APT 攻击进行复盘。

随着计算机技术与网络安全技术的不断发展，ATT&CK 框架也会随之扩充自身的内容，有针对企业的企业矩阵(Enterprise Matrix)，企业矩阵又根据不同的网络环境进一步细分为 Windows 平台、macOS 平台、Linux 平台、云端平台、网络平台以及容器等几类。由于近几年移动安全技术的发展，ATT&CK 框架又扩展了移动平台的安全矩阵，包括 Android 与 iOS 系统。

在 2020 年，ATT&CK 框架进行了一次比较大的更新，将 PRE-ATT&CK 与 ATT&CK for Enterprise 进行了合并，PRE-ATT&CK 包含了攻击者在尝试利用特定网络与系统漏洞时所使用的战术与技术。在合并之后，原本的 ATT&CK for Enterprise 的框架左侧新增了两个战术：Reconnaissance(侦察)与 Resource Development(资源开发)。

4.2 ATT&CK 框架的使用

在 MITRE 的官网可以找到 ATT&CK 框架的详细内容，网址是：https://attack.mitre.org/matrices/enterprise，ATT&CK 框架图如图 4.1 所示。

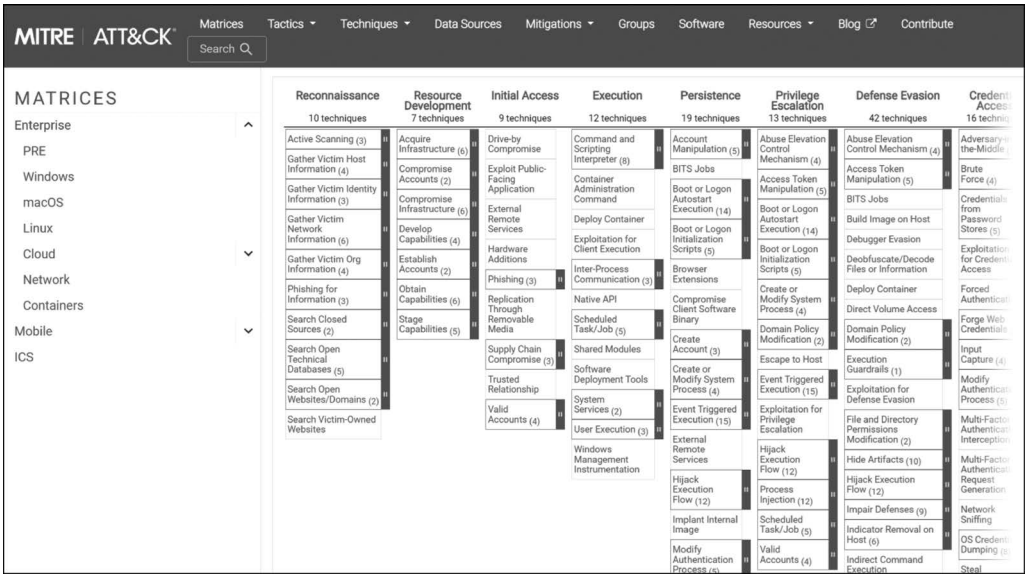

图 4.1 ATT&CK 框架图

ATT&CK 框架采用矩阵的方式对目前已知的战术与技术进行排列，矩阵的顶端为攻击战术，每一个攻击战术代表矩阵中的一列，列的内容是实现战术所使用的技术。一个攻击序列从左侧的战术开始，向右侧移动，其中，一种战术可能使用到多种技术，但是一个攻击序列不一定会使用矩阵所展示的所有战术，攻击者会倾向于使用最少数量的战术来实现目标，以降低暴露的概率。

本节将为读者按照攻击序列的发展方向，简要介绍 ATT&CK 矩阵中的各战术阶段。首先，在正式对目标展开攻击之前，攻击者需要充分收集目标的信息，涉及这个阶段的战术是矩阵左侧的 Reconnaissance(侦察)。侦察战术包括主动或者被动收集可用于目标定位的信息的技术，比如主动扫描目标的 IP 块、主机和应用程序的漏洞，通过鱼叉式网络钓鱼的方式收集目标组织成员的私人信息。攻击者可以利用这些信息规划入侵目标的范围以及优先级。侦察阶段技术如图 4.2 所示。

在侦察阶段完成信息的收集后，攻击者需要建立用来支持后续活动的资源，此类资源通常包括基础设施、账户以及功能等。基础设施具体来说可以是服务器、域名解析服务器以及僵尸网络等，其中服务器可以是物理服务器，也可以是云服务商提供的虚拟机或者容器，使用虚拟服务器的优势是防守方难以将攻击行为与攻击方进行物理绑定，云服务架构可以快速配置修改与关闭基础设施。攻击者也可能会租用受感染的系统网络，通常被称为僵尸网络。有了僵尸网络，攻击者可以执行大规模的网络钓鱼或者 DDoS 攻击。资源开发战术涉及的技术如图 4.3 所示。

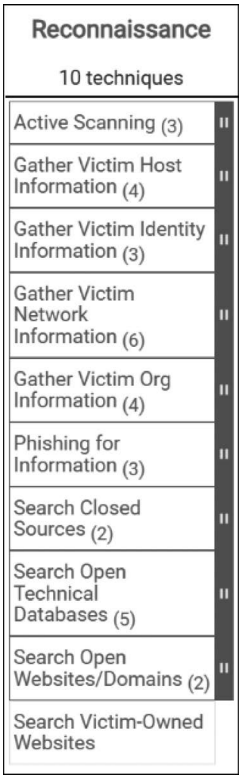

图 4.2 侦察阶段技术

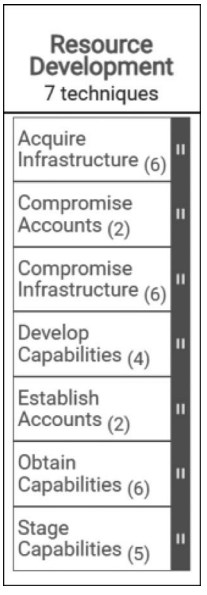

图 4.3 资源开发战术涉及的技术

账户相关的资源包括社交账号(Facebook、Twitter)、电子邮件账户等,其中社交账号和社交工程学关系比较密切,攻击者可能会更倾向于入侵已有的社交账号。与从零开始创建的社交角色相比,攻击目标对现有的社交角色更容易产生一定程度上的信任,尤其是当这类社交角色是他们感兴趣的,或者是存在一定关系时。而电子邮件账户常用于网络钓鱼,一个实际的例子是 2020 年下半年,一个名为 TA453 的黑客组织针对美国和以色列部分高级医研人员开展了网络钓鱼活动,代号“坏血行动”。在该行动中,TA453 的一名成员控制一个谷歌邮箱账号 zajfman. daniel[@]gmail. com,伪装成著名的以色列物理学家,发送与以色列核武器相关的社会工程学诱饵邮件,邮件里指向一个伪造的微软 OneDrive 服务的登录网页,其中包含一个 PDF 文档,如果邮件的接收者对该文档产生兴趣,输入自己 OneDrive 的用户名以及密码后,那么伪造的页面会跳转到真正的 OneDrive 页面,其中也确实包含 PDF 文档,从受害者的角度来看这个过程没有问题,然而当受害者在伪造页面中输入密码信息之后,自己的登录数据就泄露了。伪造的 OneDrive 页面如图 4.4 所示。

侦察战术与资源开发阶段结束后,将正式进入攻击的实施阶段。首先攻击者需要进行的是在目标企业环境中站稳脚跟,对应 ATT&CK 矩阵的初始访问(Initial Access)战术。攻击者使用初始访问内的各种技术进入目标的网络环境,比如使用远程管理和 VNC 等服务从外部访问目标内部的资源,以及将计算机配件、网络硬件等设备引入目标的系统网络中,这些设备可以作为访问权限的载体。更加常见的是利用上一个资源开发战术中收集到的用户登录数据登录受害者的账号,利用这些账号获得持续访问和访问控制的权限。初始访问战术涉及的技术如图 4.5 所示。

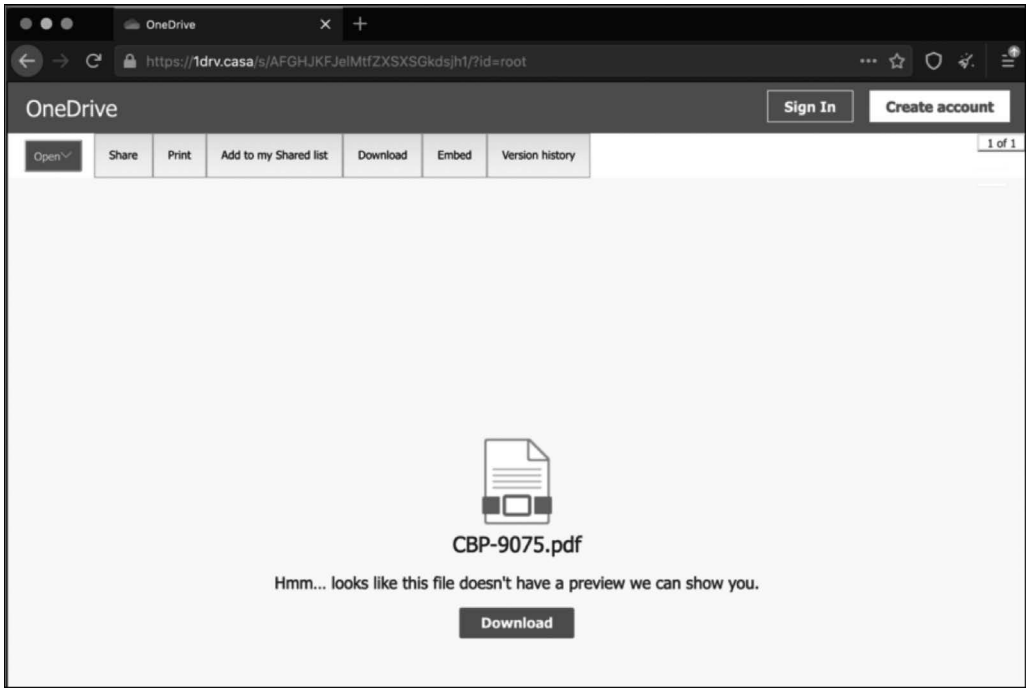

图 4.4 伪造的 OneDrive 页面

攻击者在目标环境中站稳脚跟后，接下来要进行的就是执行战术。恶意软件要发挥自己的作用，必须要运行，因此执行战术中就包括了在本地或者远程系统中运行恶意代码的技术。其中命令行工具是攻击者比较常用的媒介，攻击者将恶意代码编写成脚本，借助命令行工具执行。防守方很难通过删除命令行工具的方式切断该进攻路径，一方面，命令行工具是操作系统的重要部分，命令行工具不能被轻易删除；另一方面，系统管理员在日常工作中也依赖终端工具。除了命令行工具之外，一些脚本解释器也可以被用来执行恶意代码，比如 Python、JavaScript 语言等。

ATT&CK 矩阵的执行战术还包含了许多其他可被用来执行恶意代码的技术，如针对 Docker 容器服务的管理命令，利用不安全编码的客户端执行恶意代码，或浏览包含恶意代码的图片、文档文件等。执行战术涉及的技术如图 4.6 所示。

攻击者为了减少攻击的工作量，即使防守方重启设备之后，依然可以保持对设备的连接，这就是持久化（Persistence）战术需要完成的工作。比如攻击者获取了登录目标内部系统的账户，就会采取账户操纵的技术，延长账户的凭证有效期，包括将目标的登录凭证添加到云账户，以保持对环境中的目标账户与实例的持久访问，或者为控制账户添加其他角色与权限，从而实现对目标账户的持久访问。

如果攻击者成功地向目标设备中插入了恶意程序，那么为了

图 4.5 初始访问战术涉及的技术