第 3 章

安全测试原则

在测试 IoT 系统的漏洞时，应该从何处着手呢？如果攻击面非常小，例如，仅仅是控制单个监控摄像头的一个门户网站，那么设计相应的安全测试就很简单。然而，如果测试团队不遵循既定的方法，即使简单的测试也可能会错过应用程序中很多的关键点。

本章列出了进行渗透测试(penetration testing)时需要严格遵循的步骤。为此将把 IoT 的攻击面划分为不同的概念层，如图 3-1 所示。

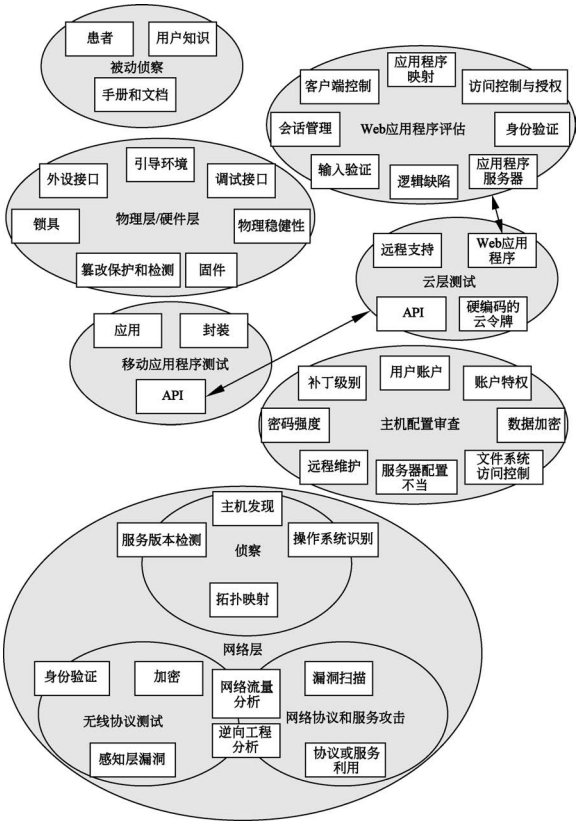

图 3-1　安全评估测试中的概念层

在对 IoT 系统进行测试时，需要遵循一个类似的稳健评估方法，因为 IoT 系统通常由许多交互的组件构成。以连接到家庭监测设备的起搏器为例，监测设备通过 4G 连接将患者的数据发送到云门户，以便临床医生检查心率是否异常。临床医生还可以使用依赖于近场通信（Near-Field Communication，NFC）棒和专有无线协议的编程器对起搏器进行配置。该系统的组成部分很多，每个部分都存在潜在的实质性的攻击面，盲目的、无组织的安全评估很可能无法取得成功。为了更好地实施评估，本章将一步步地演练被动侦察，然后讨论对物理、网络、Web 应用程序、主机、移动应用程序和云层进行测试的方法。

3.1 被动侦察

被动侦察（passive reconnaissance）通常也称为开源情报（Open Source Intelligence，OSINT），是指通过不需要直接与系统进行通信的方式来收集相关数据的过程。这是实施任何评估的第一步，从这一过程可以了解面临的基本情况。例如，可以通过下载并查阅设备的手册和芯片组的数据表、浏览在线论坛和社交媒体、采访实际用户和技术人员等方式获取相应的信息。还可以根据证书透明度（certificate transparency）要求发布的 TLS 证书收集内部的主机名，证书透明度是一项标准，要求证书颁发机构（certificate authority）在公共日志记录中发布它们颁发的证书。

1. 手册和文档

系统手册可以提供有关设备内部工作情况的大量信息，通常可以从设备供应商的官方网站上找到。如果找不到，可以通过 Google 的高级搜索功能搜寻包含设备名称的 PDF 文档，例如，搜索设备时在查询条件中可以添加“inurl:pdf”。

从手册中找出的重要信息可能会多到令人吃惊。经验表明，可以从中发掘的信息包括产品中仍然有效的默认用户名和密码、系统及其组件的详细规格、网络和体系结构图以及在其故障排除部分中隐含的有助于识别薄弱环节的信息等。

如果已经确认了硬件中包含哪些芯片组，那就有必要去查看相关的数据表（电子元件手册），因为其中可能会列出用于调试的芯片组引脚（如将在第 7 章中讨论的 JTAG 调试接口）等信息。

对于使用无线电通信的设备，另一个有用的资源是 FCC ID 网站的在线数据库。FCC ID 是分配给已在美国联邦通信委员会（United States Federal Communications Commission，USFCC）注册的设备的唯一标识符。所有在美国销售的无线发射设备都必须有 FCC ID。通过搜索特定设备的 FCC ID，可以找到其无线工作频率（如强度）、设备的内部照片、用户手册等详细信息。FCC ID 通常刻在电子元件或设备的外壳上，如图 3-2 所示的 CatWAN USB stick 的 RFM95C 芯片，第 13 章将会用到该设备。

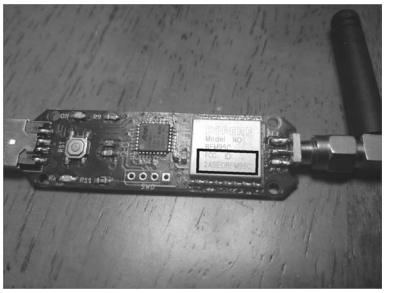

图 3-2　CatWAN USB stick 的 RFM95C 芯片上的 FCC ID

2. 专利

专利可以提供有关设备内部工作的信息。通过搜索引擎搜索供应商名称，例如，输入关键字 medtronic bluetooth，可以找到一条 2004 年发布的植入式医疗设备（Implantable Medical Devices，IMD）之间通信协议的专利。

这些专利中几乎都提供了流程图，当需要评估设备与其他系统之间的通信信道时，这些流程图会很有帮助。在图 3-3 中，同一 IMD 的简单流程图就隐含了关键的攻击途径。

在图 3-3 中既有指向也有流出 IMD 列的箭头，显示可以通过移动电话在设备和远程系统之间进行双向通信。远程系统的“患者行为和建议”操作可以启动与设备的连接。沿着箭

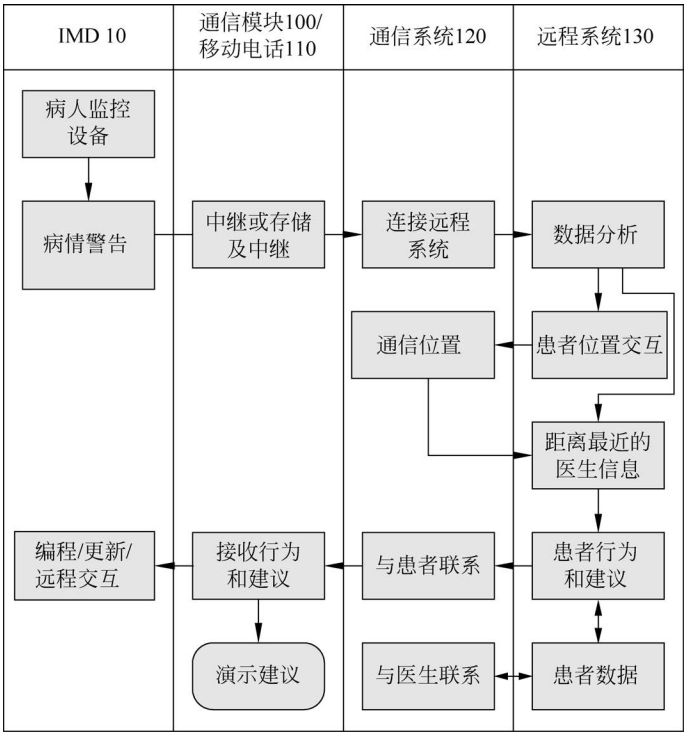

图 3-3　专利的流程图

头链的指向，该操作也可以通过更新设备的编程更改可能伤害患者的设置。因此，远程系统通过不安全的移动应用程序或实际的远程系统（通常在云端实现），就可能造成远程入侵的风险。

3. 用户知识

通过社交媒体、在线论坛和聊天室等可能会发现的公共信息真是令人惊讶。甚至可以将 Amazon 和 eBay 的用户评论作为知识来源。可以关注用户对于某些设备功能的吐槽，因为某设备存在较多的故障，往往说明存在潜在的漏洞。例如，某个用户在抱怨当触发了一系列条件后，设备会崩溃。这就是一个很好的调查线索，说明设备在特定的输入条件下，会导致某种逻辑故障或内存损坏，这就是个漏洞。另外，许多用户会发布详细的产品评论并附上明细清单和图片等。此外，留意 LinkedIn 和 Twitter 上的个人资料或帖子，从事 IoT 系统工作的工程师和 IT 人员可能会发布很多生动有趣的技术信息方面的爆料。例如，若某人发帖说，他在特定的 CPU 体系结构方面具有很强的专业背景，那么相关制造商的许多设备很可能都是基于该体系结构的。若另一名员工对特定的框架疯狂吐槽（或赞扬，尽管这种情况不太常见），那么可能会有相关公司在使用该框架开发软件。

一般来说，每个 IoT 行业都有自己的专家组，通过咨询这些专家可以获取有用的信息。例如，如果正在对发电厂进行评估，向操作员或技术人员询问他们的工作流程，对于确定潜在的攻击途径可能会很有价值。在医学界，护士通常是 IoT 系统的系统管理员和主要操作员，因此，他们对设备的来龙去脉了解得最清楚，可能的情况下应该多咨询他们。

3.2 物理层或硬件层

IoT 设备中最重要的攻击途径之一是硬件。如果攻击者可以掌控系统的硬件组件，他们通常可以将自己的权限提升，因为系统总是无条件地信任所有具有物理访问权限的人。换句话说，如果一个专业的攻击者对系统具有物理访问权限，则这个系统处于极大的危险中。假设攻击者是动机最强的威胁实施者，如国家资助的、拥有几乎无限的时间和资源的威胁行为者，他们将会拥有可供其使用的设备的物理备份。即使是一些特殊用途的系统（如大型超声机等），攻击者也可以从网上市场、未能安全地处置设备的公司，甚至通过盗窃，来获得硬件。即使系统更迭了很多代，漏洞可能依旧没变，所以他们甚至不需要知道设备的确切版本。

对硬件层的评估应包括外设接口、引导环境、物理锁、篡改保护、固件、调试端口和物理健壮性等。

3.2.1 外设接口

外设接口（peripheral interface）是物理通信端口，允许连接外部设备，如键盘、硬盘和网卡等。对外设接口的测试首先检查是否启用了任何活动的 USB 端口或 PC 卡插槽，以及它

们是不是可引导的。通过在设备上启动自己的操作系统，安装未加密的文件系统，提取可破解的散列(hash)或密码以及在文件系统上安装自己的软件，将技术安全控制覆盖掉，就可获得对各种 x86 系统的管理访问权限。即使没有可引导的 USB 端口，也可以提取硬盘并进行读写，尽管这种技术不太容易做到。注意，篡改硬件以提取磁盘可能会对组件造成损坏。

USB 端口可能成为攻击途径还有另一个原因：有些设备(主要是基于 Windows 的)具有 **kiosk 模式(kiosk mode)**，该模式限制了用户界面。比如用于取款的 ATM 机，尽管在后端可能运行于 Windows XP 嵌入式操作系统上，但用户能看到的只是具有一组特定选项的受限的图形界面。设象一下，如果可以将 USB 键盘连接到设备已暴露的端口，再使用一些特定的组合键(如 Ctrl＋Alt＋Delete 或 Windows 键)，或许就能够退出 kiosk 模式并直接访问系统的其余部分了。

3.2.2　引导环境

对于使用传统 BIOS 的系统(通常是 x86 和 x64 平台)，首先要检查 BIOS 和引导加载程序是否受密码保护，并确定首选的引导顺序。如果系统首先从可移动设备引导，那么，无须对 BIOS 设置进行任何更改就可以引导自己的操作系统了。此外，检查系统是否启用并优先处理预启动执行环境(Pre-boot Execution Environment，PXE)，该机制允许客户端使用 DHCP 和 TFTP 的组合通过网络来引导系统，这就为攻击者设置恶意网络引导服务器留下了空间。即使安全地配置了启动顺序并且所有设置都受密码保护，仍可以将 BIOS 重置为其默认的、干净的且未受保护的设置(例如，通过临时卸下 BIOS 电池)。如果系统具有统一可扩展固件接口(Unified Extensible Firmware Interface，UEFI)的安全启动(secure boot)，也需要评估其实现情况。UEFI 安全启动(UEFI secure boot)是一种安全标准，用于验证引导软件是否被 rootkit 之类的工具篡改。它通过检查 UEFI 固件驱动程序和操作系统的签名来实现。

还可以使用可信执行环境(Trusted Execution Environment，TEE)技术，例如 ARM 平台中的 TrustZone 或高通公司的安全引导功能，这些技术可以验证安全引导镜像。

3.2.3　锁具

评估还应检查设备是否受到某种锁具的保护，如果是，开锁的难易程度如何？此外，检查是否有万能钥匙可以打开所有的锁具，或者每台设备是否都配备了单独的钥匙。在实施的评估中，可能会发现同一制造商的所有设备都使用了同一把钥匙的情况，这使得锁具毫无用处，因为任何人都可以轻松地复制该钥匙。例如，曾发现一把钥匙可以打开产品系列的所有机柜(cabinet)，从而可以物理访问药物输液泵的系统配置。

要对锁具进行评估，除了需了解目标锁具的类型外，还需要一套开锁的工具。例如，弹簧锁(tumbler lock)的打开方式与电子锁的打开方式就不同，如果关闭电源，电子锁可能就

无法打开或关闭了。

3.2.4 篡改保护和检测

检查设备是不是防篡改(tamper-resistant)且保留了篡改痕迹(tamper-evident)。例如,使设备保留篡改痕迹的一种方法是使用带有穿孔带(perforated tape)的标签,该标签在打开后会永久留下痕迹。其他防篡改保护包括溢出物、防拆夹、用环氧树脂密封的特殊外壳或在拆卸设备时可擦除敏感内容的物理保险丝等。篡改检测机制在检测到有人试图破坏设备的完整性时,会发送警报或在设备中创建日志文件。在企业内部对 IoT 系统进行渗透测试时,对篡改保护和检测的检查尤为重要。许多威胁来自企业内部,由员工、承包商甚至是前员工造成,因此,防篡改保护有助于识别所有故意对设备的更改。攻击者在拆卸防篡改设备时就会遇到困难。

3.2.5 固件

第 9 章会详细介绍固件的安全性,这里就不再赘述。切记,未经许可访问固件可能会产生法律后果。如果计划发表涉及访问固件或对其中找到的可执行文件进行逆向工程的安全研究,这一点就很值得重视。有关法律环境方面的信息,可参阅 1.4.1 节。

3.2.6 调试接口

制造商用于简化开发、制造和调试的调试、服务或测试点接口(debug,service,or test point interface)通常可以在嵌入式设备中找到,利用它们马上可以获得 root 访问权限,所以对这些接口的检查也很重要。如果不首先通过与调试端口的连接进入系统的 root shell,就无法完全理解测试过的许多设备,因为没有其他方法来访问和检查实时系统。想做到这一点,首先要求熟悉这些调试接口使用的通信协议的内部工作原理。最常见的调试接口类型包括 UART、JTAG、SPI 和 I^2C 等。第 7 章和第 8 章会详细讨论这些接口。

3.2.7 物理稳健性

对硬件物理特性造成的任何限制进行测试可确定系统的物理稳健性。例如,对系统是否存在电池耗尽攻击(battery drain attack)进行评估,当攻击者使设备过载并导致设备在短时间内耗尽电池电量,从而有效地导致拒绝服务时,就会发生此类情况。设想当一个患者的生命完全依赖可植入的起搏器时,这是多么的危险。这类测试的另一种类型是噪声攻击(glitching attack),即在某些敏感操作期间故意引入硬件故障,以破坏系统的安全性。在作者最令人惊讶的一次成功案例中,对一个嵌入式系统的印刷电路板(Printed Circuit Board,PCB)发动了噪声攻击,成功地使嵌入式系统的启动过程进入 root shell。此外,还可以尝试诸如差分功率分析(differential power analysis)之类的侧信道攻击,该方法可以通过测量加

密操作的功耗来获取机密信息。

检查设备的物理特性也有助于对其他安全功能的稳健性做出有根据的猜测。例如，电池寿命较长的微型设备，其网络通信可能采用的就是弱加密形式，原因是更强的加密形式所需的处理能力会更快地耗尽电池电量，并且由于设备的尺寸限制，电池的容量是有限的。

3.3　网络层

网络层包括通过标准网络通信路径直接/间接进行通信的所有组件，通常是最主要的攻击途径。因此，将它分解为更小的部分：侦察（reconnaissance）、网络协议（network protocol）、服务攻击（service attacks）以及无线协议测试（wireless protocol testing）。

尽管本章涵盖的很多其他测试都涉及网络，但如有必要，后续会对这些测试进行专门介绍。例如，因为 Web 应用程序评估的复杂性和涉及的测试活动的数量，后续会有专门的章节进行讨论。

3.3.1　侦察

前面已经讨论了对 IoT 设备实施被动侦察时需要采取的一般步骤。本节将概述针对网络的主动和被动侦察，这是任何网络攻击的第一步。被动侦察是指在网络上侦听有用的数据，而主动侦察（active reconnaissance）是指需要与目标对象进行交互，对设备直接进行查询的侦察。

若是对单个 IoT 设备进行测试，过程相对来说较为简单，因为只需要扫描一个 IP 地址。但对于一个大型的生态系统，例如智能家居或含有很多医疗设备的医疗保健场景，网络侦察就会变得很复杂，主要包括下面具体讨论的主机发现、服务版本检测、操作系统识别和拓扑映射等。

1. 主机发现

主机发现（host discovery）是指使用各种技术进行探测以便确定哪些网络中哪些主机处于活动状态。这类探测技术包括发送互联网控制消息协议（Internet Control Message Protocol，ICMP）、回显请求（echo-request）数据包、对公用端口进行 TCP/UDP 扫描、侦听网络上的广播流量，或者如果主机位于同一 L2 网段上，则执行 ARP 请求扫描。其中，L2 是指计算机网络 OSI 模型的第 2 层，即数据链路层，负责跨物理层在同一网段上的节点之间传送数据。以太网就是一种常见的数据链路协议。对于一个复杂的 IoT 系统，例如管理跨网段监控摄像头的服务器，不依赖某一种特定的技术就很重要。相反，需要利用多样化的技术才能有更大的机会绕过防火墙或严密的虚拟局域网（Virtual Local Area Network，VLAN）配置。

在对 IoT 系统进行渗透测试时，如果不知道被测试系统的 IP 地址，这一步可能是最有

用的。

2. 服务版本检测

在识别实时主机之后，还需要确定其上运行的所有侦听服务。可以从 TCP 和 UDP 端口扫描开始。然后，使用服务指纹工具（如 Amap 或 Nmap 的-sV 选项）同时进行横幅抓取（banner grabbing，指连接到网络服务并读取其作为响应发送回来的初始信息）和探测。某些服务，尤其是医疗设备上的服务，即使进行简单的探测，也特别容易发生故障。例如，仅仅因为使用 Nmap 的版本检测功能进行扫描，IoT 系统就发生了崩溃并重启。这种扫描发送特制的数据包，以从某些类型的服务中引发响应，但是在连接到这些服务时并不会发送任何信息。显然，这些相同的数据包可能会使某些敏感设备不稳定，原因是这些设备在其网络服务上缺乏强大的输入清理（input sanitization），从而导致内存损坏并发生崩溃。

3. 操作系统识别

需要对每个测试主机上运行的到底是什么操作系统进行确认，以便后续开发相应的漏洞滥用程序。至少，需要确定其体系结构（例如，x86、x64 或 ARM）。理想情况下，对于 Windows 操作系统来说，需要确定其服务包级别（service pack level），对于 Linux 或基于 Unix 的系统，需要确定内核版本（kernel version）。

通过分析主机对特制的 TCP、UDP 和 ICMP 数据包的响应，可以对操作系统进行识别，这一过程称为**指纹识别（fingerprinting）**。由于不同操作系统中 TCP/IP 网络堆栈的实现存在细微差异，因此响应就会有所不同。例如，针对开放端口进行的 FIN 探测，某些老版本 Windows 系统的响应是 FIN/ACK 数据包，有些系统的响应则是 RST 数据包，而其他系统可能根本不响应。通过对这些响应的统计分析，可以对每种操作系统的版本概况有所了解，然后就可以对其操作系统及版本进行识别了。更多信息，访问 Nmap 相关文档的“TCIP/IP Fingerprinting Methods Supported by Nmap”页面。

服务扫描还可以帮助执行操作系统的指纹识别，因为许多服务在其横幅公告中公开了系统的信息。Nmap 是开展这两项工作的绝佳工具。但请注意，对于某些敏感的 IoT 设备，操作系统的指纹识别可能是侵入性的，并可能导致系统崩溃。

4. 拓扑映射

拓扑映射（Topology mapping）对网络中不同系统之间的连接进行建模。当必须要测试设备及系统的整个生态系统时，此步骤适用，其中一些设备和系统可能通过路由器和防火墙进行连接，不一定位于同一 L3 网段上（L3 是指计算机网络 OSI 模型的第 3 层。即网络层，主要负责数据包的转发和路由。当数据通过路由器传输时，第 3 层开始发挥作用）。创建待测试资产的网络映射对于威胁建模非常有用：有助于了解滥用不同主机中的一系列的漏洞进行攻击，是如何导致关键资产受损的。图 3-4 给出了一个高级拓扑图。

这张抽象的网络拓扑图描述了一名佩戴了 IMD 的患者如何与家庭监控设备进行通信，继而家庭设备依赖本地 Wi-Fi 连接将诊断数据发送到云端，医生可以定期对其进行监控以

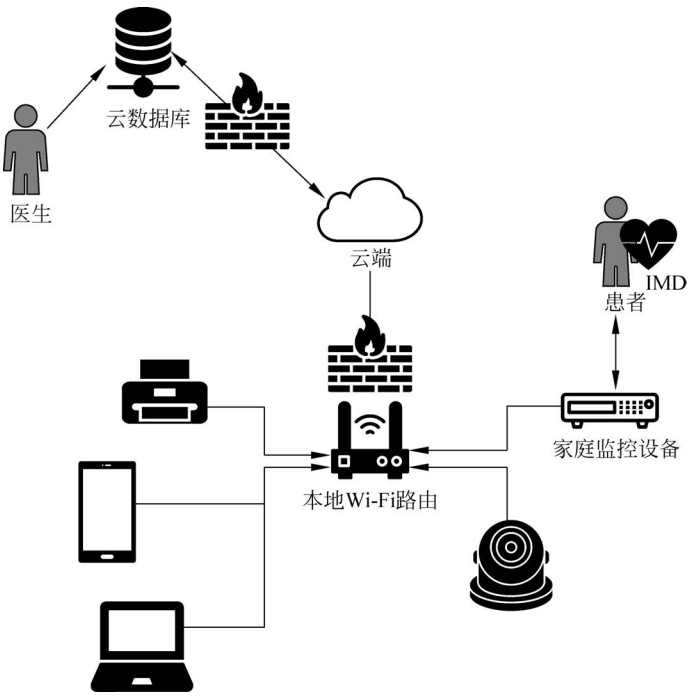

图 3-4　一个包含 IMD 患者家庭监控设备的家庭网络拓扑图

检测任何异常状况。

3.3.2　网络协议和服务攻击

网络协议和服务攻击包括以下几个阶段：漏洞扫描、网络流量分析、协议逆向工程以及协议或服务利用。尽管可以独立于其他阶段进行漏洞扫描，但其他各个阶段是相互依赖的。

1. 漏洞扫描

首先检查数据库，例如检查国家漏洞数据库(National Vulnerability Database，NVD)或VulnDB，查找已公开的网络服务中的任何已知漏洞。有时，系统已经太过时了，以至于自动漏洞扫描工具将生成连篇累牍的报告。甚至可以在没有身份验证的情况下远程滥用某些漏洞。为了进行尽职调查(due diligence)，至少运行一种扫描工具就可以马上收获唾手可得的成果(low-hanging fruit)。如果发现了严重的漏洞(如远程执行代码)，则或许能够在设备上获取 shell，这将有助于完成评估的其余部分。对漏洞进行扫描需要确保始终在受控环境中进行，并在发生不可预见的停机时对其进行密切监视。

2. 网络流量分析

在安全评估过程的早期，让类似于 Wireshark 或 tcpdump 这样的流量捕获工具运行一段时间，以了解正在使用的通信协议。如果 IoT 系统涉及不同的交互组件，例如带有服务器的监控摄像头或带有 EHR 系统的药物输液泵，应该能够捕获它们之间传输的任何网络

流量。已知的攻击，如 ARP 缓存投毒（ARP cache poisoning），通常会在同一 L3 网段上起作用。

理想情况下，可以直接在设备上运行这些流量捕获工具，以捕获本地主机上潜在的进程间通信（Inter-Process Communication，IPC）流量。在嵌入式设备上运行这些网络工具可能会遇到更大的困难，因为此类嵌入式设备中通常不会安装相关的工具，没有一个简单的流程来设置它们。但是，可以将 tcpdump 等工具交叉编译并安装在非常受限的设备上（例如起搏器家庭监控系统等），第 6 章会对此进行演示。

捕获了网络流量的代表性样本后，就可以开始对其进行分析了，确定是否存在不安全的通信信道，如明文协议；确认已知的易受攻击的协议，如通用即插即用（Universal Plug and Play，UPnP）网络协议集；进一步检查逆向工程的专有协议。

3. 逆向工程协议

应该对发现的任何适当的通信协议进行逆向工程。创建新协议总是一把双刃剑；有些系统确实需要自己的协议栈来实现其性能、功能甚至安全性。但是，设计和实现一个健壮的协议通常是一项非常复杂的任务。目前，许多 IoT 系统都利用 TCP 或 UDP，并在其上进行构建，通常使用 XML、JSON 或其他结构化语言的某种变体。在复杂的情况下，会碰到专有的无线协议，这些协议几乎没有可用的公开信息，例如在植入式起搏器中遇到的那些协议。在这种情况下，从不同的角度去检查协议可能更容易。例如，尝试调试与驱动程序层进行通信的系统服务，该驱动程序层是负责传输无线电信号的。这样，就没必要去分析专有无线协议了。相反，也许可以通过了解其上的图层来弄清楚它是如何工作的。

例如，在评估起搏器时就使用了该技术。为此，利用附加到与驱动程序层通信的进程的工具，如 strace。通过分析日志和 pcap 文件，识别了底层通信信道，而无须在专有无线信道上进行无线电信号分析或其他费时的方法（如傅里叶变换）。

4. 协议或服务利用

作为网络攻击的最后一步，实际上，应该通过编写一个滥用该协议或侦听服务的概念验证（proof-of-concept）程序，以便利用该协议或侦听服务。至关重要的是，必须确定可利用性所需的确切条件。该漏洞滥用是否 100％可重现？是否要求系统首先处于某种状态？防火墙的规则是否阻止入口或出口通信？成功利用系统后，系统是否可用？以上问题都必须确保给出了可靠的答复。

3.3.3 无线协议测试

由于 IoT 生态系统中短程、中程和远程无线电通信协议的普遍性，本章中我们将用整整一小节专门来介绍无线协议测试。该部分可能与其他文献所描述的**感知层（perception layer）**一致，其中包括**无线射频识别（Radio-Frequency Identification，RFID）**、**全球定位系统（Global Positioning System，GPS）**和 NFC 等传感技术。

分析这些技术的过程与本章之前讨论的网络层的“网络流量分析”和“逆向工程协议”有重叠。对无线协议的分析和攻击通常都需要专用的设备，包括某些可注入(injection-capable)的 Wi-Fi 芯片组(如 Atheros)；蓝牙加密狗(如 Ubertooth)；和软件无线电(Software Defined Radio，SDR)工具(如 HackRF 或 LimeSDR)。

在此阶段，测试与正在使用的特定无线协议相关的某些攻击。例如，如果任何 IoT 组件使用了 Wi-Fi，则测试关联攻击、是否使用有线等效保密(Wired Equivalent Privacy，WEP)以及基于弱证书的不安全的 Wi-Fi 保护访问(Wi-Fi Protected Access，WPA/WPA2)。WPA3 可能很快也会属于这一类了。第 10～13 章会一步步地介绍针对这些协议的最重要的攻击。对于自定义的协议，需要测试是否缺乏身份验证(包括相互身份验证)以及是否缺乏加密和完整性检查。即使在一些关键的基础架构设备中，以上这些情形都经常出现。

3.4　Web 应用程序评估

Web 应用程序，包括 IoT 系统中使用的应用程序，提供了最简单的一种网络入口，因为它们通常可以从外部访问，并且充斥着大量漏洞。评估 Web 应用程序是一个庞大的主题，并且已有巨量的资源来提供指导。因此，本节重点介绍适用于 IoT 设备中 Web 应用程序的技术。事实上，这些技术与已有的几乎所有 Web 应用程序中的技术没有明显区别，但是，在嵌入式设备中发现的那些应用程序通常缺乏安全的软件开发生命周期，从而导致很多明显的漏洞。Web 应用程序测试的资源包括 *The Web Application Hacker's Handbook* 以及 OWASP 所有的项目，例如其 Top 10 列表、应用安全评估标准(Application Security Verification Standard，ASVS)项目和 OWASP 测试指南(OWASP testing guide)。

3.4.1　应用程序映射

要映射一个 Web 应用程序，首先浏览网站的可见的、隐藏的以及默认的内容，识别数据输入点和隐藏字段，并枚举所有参数。一次抓取一个页面的数据挖掘软件——**自动爬虫工具(spidering tools)**有助于加快该过程，但也可以始终手动进行浏览。映射过程中，还可以利用拦截代理进行**被动爬取(passive spidering**，在手动进行浏览时用来监视 Web 内容)以及**主动爬取(active spidering**，使用先前发现的 URL 和嵌入 JavaScript 中的 AJAX 请求作为起点主动抓取网站)。

通过使用常见的文件或目录名称及扩展名，可以发现通常无法通过超链接访问的**隐藏内容(hidden content)**或 Web 应用程序端点。注意，这个过程可能会非常繁杂，因为所有的请求都会生成大量的网络流量。例如，DirBuster 网络爬网工具的通用目录和文件名的中型列表就有 220 560 个条目。这意味着，如果要使用它，将会向目标发送至少 220 560 个

HTTP 请求，以期发现隐藏的 URL。特别是在受控环境中进行评估时，这是不能忽视的问题。在 IoT 设备中经常会发现一些非常有趣、未经身份验证的 Web 应用端点。例如，作者曾经在一款流行的监控摄像头模型上发现了一个隐藏的 URL，它允许在完全未经身份验证的情况下拍摄照片，这实际上就是允许攻击者远程监控摄像头中的所有内容。

识别 Web 应用程序接收用户数据的入口点也很重要。Web 应用程序中的大多数漏洞都是因为应用程序从未经身份验证的远程操作者接收了不可信的输入信息。后续可以使用这些入口点进行模糊测试（一种自动提供无效随机数据作为输入的方法）和注入测试。

3.4.2 客户端控制

客户端控制（client-side control）是由浏览器、胖客户端或移动应用处理的任何内容。客户端控件可能包括隐藏字段、Cookies 和 Java 小程序，也可以是 JavaScript、AJAX、ASP.NET、ViewState、ActiveX、Flash 或 Silverlight 对象。例如，嵌入式设备上的许多 Web 应用程序会在客户端执行用户身份验证，但攻击者总是可以绕过这些身份验证，因为用户可以控制在客户端发生的一切。这些设备了使用 JavaScript 或.jar、.swf 和.xap 文件，攻击者可以反编译和修改这些文件以实现自己的企图。

3.4.3 身份验证

应用程序身份验证机制中存在大量的漏洞。众所周知，大量的 IoT 系统中存在较弱的预配置证书，并且用户通常不会更改这些证书。通过参考手册或其他在线资源，或者只是通过猜测就可以发现这些证书。在测试 IoT 系统时，经常可以看到的用户名/密码包括流行的 admin/admin，到 a/a（即用户名：a，密码：a），甚至根本不需要身份验证。若想破解非默认密码，可以对所有身份验证端点实施字典攻击。**字典攻击（dictionary attack）**使用自动化的工具，通过测试字典中最常见的单词或已泄露的常见密码列表来猜测密码。作者编制的几乎每份安全评估报告中都将“缺乏暴力保护”（lack of brute-force protection）列为一项发现，因为 IoT 嵌入式设备的硬件资源通常都很有限，可能无法像 SaaS 应用程序那样保持应有的状态。

常见的身份验证测试包括：

（1）对不安全的证书传送进行测试（通常包括默认的 HTTP 访问，未重定向到 HTTPS）；

（2）对任何“forgot password”和“remember me”功能进行检查；

（3）实施猜测并列出有效的用户名的**用户名枚举（username enumeration）**；

（4）并且查找身份验证失败但由于某些例外情况，应用程序提供了开放访问的**失效开放（fail-open）**条件。

3.4.4　会话管理

Web 应用程序会话(**Web application session**)是与单个用户关联的 HTTP 事务序列。会话管理或跟踪这些 HTTP 事务的过程可能会变得复杂,因此需要检查这些进程是否存在缺陷。检查是否使用了可预测的令牌、不安全的令牌传输以及日志中令牌是否泄露。可能还会发现会话到期时间不足、**会话修复**(**session-fixation**)漏洞和**跨站点请求伪造**(**Cross-Site Request Forgery**,**CSRF**)攻击等,利用这些攻击,可以操纵经过身份验证的用户去执行不必要的操作。

3.4.5　访问控制与授权

访问控制与授权测试重点检查站点是否强制实施了适当的访问控制。**用户级隔离**(**user-level segregation**)或者不同的用户被赋予不同的数据/功能的访问权限,是 IoT 设备中常见的做法,也称为**基于角色的访问控制**(**Role-Based Access Control**,**RBAC**)。对于复杂的医疗设备来说尤其如此,例如,在一个 EHR 系统中,临床医生账户将比护士账户拥有更多的访问权限,后者可能仅具备读取的访问权限。同样,在摄像头系统中至少要有一个管理员账户,其权限包括对系统配置的修改,同时还有一个权限较低的仅供查看的账户,该账户允许设备操作员查看监控录像。但是,这些系统需要有适当的访问控制才能正常工作。前面已经讨论过一些系统,只要知道正确的 URL 或 HTTP 请求,就可以由非特权账户申请特权操作,也称为**强制浏览**(**forced browsing**)。如果系统拥有多个账户,必须对所有账户的权限边界进行测试。例如,访客账户是否可以访问只有管理员才能使用的 Web 应用程序功能?访客账户是否可以访问由其他授权框架管理的管理员的 API?

3.4.6　输入验证

输入验证可以确保应用程序对所有数据输入点的用户输入都正确地进行了验证和清理。此操作至关重要,因为最普遍的 Web 应用程序漏洞类型就是注入(injection),即用户向应用程序提交自己的代码作为其输入(参阅 OWASP 的 Top 10 漏洞列表)。测试应用程序的输入验证可能是一个非常漫长的过程,因为它要对所有类型的注入攻击进行测试,包括 SQL 注入、跨站点脚本(Cross-Site Scripting,CSS)、操作系统命令注入以及 XML 外部实体(XML External Entity,XEE)注入等。

3.4.7　逻辑缺陷

逻辑缺陷测试可以检查是否存在由于逻辑缺陷导致的漏洞。当 Web 应用程序具有多阶段进程时,一个操作必须紧跟另一个操作,在这些进程中,此任务尤其重要。如果无序地执行这些操作会导致 Web 应用程序进入不可预计且不希望的状态,即 Web 应用程序存在

逻辑缺陷。通常,逻辑缺陷的查找是一个手动过程,需要了解应用程序的应用场景及其所针对行业的情况。

3.4.8 应用程序服务器

应用程序服务器测试用于检查托管应用程序的服务器是否安全。如果将安全的 Web 应用程序托管在一个不安全的应用程序服务器上,则违背了保护实际应用程序的目的。要测试服务器的安全性,主要的测试包括以下几点。

(1) 使用漏洞扫描程序检查应用程序服务器的缺陷以及公开的漏洞。

(2) 检查是否存在反序列化攻击(deserialization attack)并测试所有 Web 应用程序防火墙的稳健性。

(3) 要测试服务器是否配置不当,如目录列表、默认内容以及有风险的 HTTP 方法等。

(4) 评估 SSL/TLS 的稳健性,检查弱密码、自签名证书以及其他常见漏洞。

3.5 主机配置审查

在获得本地访问权限后从内部对系统进行**主机配置审查**(**host configuration review**)。例如,可以从 IoT 系统的 Windows 服务器组件上的本地用户账户执行此评估。进入内部后,可以对各种技术进行评估,包括用户账户、远程支持连接、文件系统访问控制、已公开的网络服务、不安全的服务器配置等。

3.5.1 用户账户

用户账户测试用于评估系统中配置的用户账户的安全程度,包括测试是否存在默认用户账户及检查账户策略的稳健性。用户账户策略如下。

(1) **密码历史记录**(**password history**):是否以及何时可以重复使用旧密码。

(2) **密码过期**(**password expiration**):系统强制用户更改密码的频率。

(3) **锁定机制**(**lockout mechanisms**):用户账户被锁定之前,允许的错误次数。

如果该 IoT 设备属于某企业网络,还应考虑该公司的安全策略,以确保账户一致。例如,若该公司的安全策略要求用户每 6 个月更改一次密码,请检查所有账户是否都符合该策略。理想情况下,如果系统允许将账户与公司的 Active Directory 或 LDAP 服务进行集成,那么该公司应该能够通过服务器集中实施这些策略。

这一测试步骤听起来可能很平常,但它却是最重要的步骤之一。攻击者经常滥用配置较弱的用户账户,这些账户未以集中方式进行管理,结果常常被忽视。在作者开展的评估中,经常发现本地用户账户的非过期密码与用户名相同。

3.5.2　密码强度

密码强度测试用于评估用户账户密码的安全性。密码的强度很重要，因为攻击者可以使用自动化工具来猜测密码强度较弱的证书。检查是否强制实施了密码复杂性的要求，对于 Windows 操作系统是通过成组策略或本地策略来实现的；而基于 Linux 的系统则通过可插入身份验证模块（Pluggable Authentication Modules，PAM）实现。但有一条注意事项：身份验证要求不能影响业务工作流。设想以下的场景：某个外科手术系统强制实施 16 个字符的密码复杂性要求，并在 3 次错误尝试后就将用户锁定。若外科医生或护士遇到紧急情况，并且没有其他方式向系统进行身份验证时，就会导致灾难性的后果。因为在紧急情况下，患者的生命岌岌可危，哪怕几秒钟都是很重要的，这就必须确保安全性要求不会产生负面的后果。

3.5.3　账户特权

账户特权测试用于检查账户和服务是否配置了**最小权限原则**（**principle of least privilege**），即用户只能访问所需要的资源，不能越雷池半步。测试中经常会碰到没有进行精细权限分离的配置欠佳的软件。例如，当主进程不再需要提升的权限时，通常没有放弃该特权；或者，系统允许不同的进程都在同一个账户下运行。这些进程通常只需要访问一组有限的资源，结果它们被过度特权化；一旦遭到入侵，将为攻击者提供对系统的完全控制权。我们还经常发现使用 SYSTEM 或 root 权限运行的简单日志记录服务。“服务的权限过高”（services with excessive privilege）几乎出现在作者编制的每份安全评估报告中。

在 Windows 系统中，可以使用**托管服务账户**（**managed service account**）解决账户特权引发的问题，隔离重要应用程序的域账户，并自动管理其证书。对于 Linux 系统，使用 **capability**、**seccomp**（将系统调用列入白名单）、**SELinux** 以及 **AppArmor** 等安全机制，可以帮助限制进程的权限并强化操作系统。此外，Kerberos、OpenLDAP 以及 FreeIPA 等解决方案都可以帮助进行账户管理。

3.5.4　补丁级别

补丁级别测试用于检查操作系统、应用程序和所有第三方库是否都是最新的，并且会检查更新过程。补丁很重要，也很复杂，而且在很大程度上被误解了。测试过时的软件可能看起来像是一项常规任务（通常可以使用漏洞扫描工具自动执行），但很难找到一个完全更新的生态系统。要检测具有已知漏洞的开源组件，可以利用**软件成分分析**（**software composition analysis**）工具自动检查第三方代码是否缺少补丁。要检测操作系统中缺少的补丁，可以依靠经过身份验证的漏洞扫描，甚至可以手动进行检查。同时还需要检查供应商是否仍然支持 IoT 设备的 Windows 或 Linux 内核版本，避免供应商不再支持的情况。

对系统组件打补丁是信息安全行业，尤其是 IoT 领域的祸根之一。一个原因是，嵌入式设备本质上更难修补，因为它们通常依赖一成不变的复杂固件。另一个原因是，由于**停机**（**downtime**）成本（客户无法访问系统的时间）以及所涉及的工作量，定期修补某些系统（如 ATM 机）的成本可能高得令人望而却步。对于医疗设备等更特殊的系统，供应商必须在发布任何新补丁之前首先执行严格的测试。例如，没有人希望由于血液分析仪更新引起的浮点误差而意外地给出了肝炎阳性的结果。又如，对植入式起搏器，如果需要更新，也是很难想象的，难道将所有患者召集到医生办公室“打补丁”吗？

在作者开展的评估中，经常可以看到核心的组件是最新的，但第三方软件却没有更新过。Windows 系统中常见的此类软件包括 Java、Adobe 甚至是 Wireshark。在 Linux 设备中，过时版本的 OpenSSL 也很常见。有时，系统中安装的一些软件完全没有存在的理由，最好的办法是将其删除而不是对它进行更新或修补。例如，在与超声波机器连接的服务器上就没有安装 Adobe Flash 的必要。

3.5.5 远程维护

远程维护测试用于检查设备的远程维护和支持连接的安全性。通常，一个组织不会将设备运送至供应商处进行修补，而是会致电设备供应商，让其技术人员远程连接到系统。攻击者有时会利用这些功能，将其用作获取管理员权限的后门。大多数的此类远程连接方法都是不安全的。以美国 Target 百货公司入侵事件为例，攻击者通过第三方暖通空调公司（HVAC）渗透到了该商场的主网络。

供应商一般会对设备进行远程修补，因为通常没有好的方法可以及时修补网络中的 IoT 设备。因为涉及一些敏感且复杂的设备，公司员工不能悄无声息地就开始在这些设备上安装补丁，且修补过程中设备总是难免会发生宕机。如果设备在紧急需要使用时发生了故障（设想若是医院的 CT 扫描仪或发电厂的关键温度传感器），会出现什么样的情况呢？

不仅要评估远程支持软件（理想情况下通过对其二进制文件进行逆向工程）及其通信信道，还要评估已建立的远程维护流程，这一点很重要，要检查所有设施是否都是 24/7 连接，供应商进行连接时是否采取了双因素身份验证，以及是否有日志记录。

3.5.6 文件系统访问控制

文件系统访问控制测试用于检查关键的文件和目录是否遵循了最小权限原则。经常可以看到，低权限的用户可以读/写关键的目录和文件（如服务可执行文件），从而导致了权限提升攻击（privilege escalation attack）。非管理员用户真的需要对 C:\Program Files 具有写入权限吗？是否所有的用户都需要访问/root 的权限？作者曾经评估过一个嵌入式设备，其中包含 5 个以上的不同启动脚本，这些脚本都可由非 root 用户编写。这样产生的后果是一个具有本地访问权限的攻击者基本可以用 root 身份运行自己的程序，并获得对系统的完全控制。

3.5.7　数据加密

数据加密测试用于检查敏感数据是否已加密。首先识别最敏感的数据，例如**受保护的健康信息（Protected Health Information，PHI）**或**个人身份信息（Personally Identifiable Information，PII）**。PHI 包括有关健康状况、提供或支付医疗保健的任何记录，而 PII 是指能对特定个人进行识别的任何数据。通过检查系统配置中的密码算法，确保此数据处于静态加密状态。如果有人窃取了设备的磁盘，他们能读取这些数据吗？是否进行了全磁盘加密、数据库加密或任何类型的静态加密，其加密安全性如何？

3.5.8　服务器配置不当

配置不当的服务可能是不安全的服务。例如，仍然有系统在默认情况下，启用了访客用户对 FTP 服务器的访问权限，从而允许攻击者匿名连接并对特定文件夹进行读/写。作者曾经发现一个 Oracle 企业管理器（Oracle Enterprise Manager，OEM），作为 SYSTEM 运行，可以使用默认证书远程访问，这样，攻击者可以通过滥用存储的 Java 程序执行操作系统命令。该漏洞使攻击者能够通过网络完全破坏系统。

3.6　移动应用程序和云测试

移动应用程序和云测试用于评估与 IoT 系统关联的任何移动应用程序的安全性。开发人员一般希望为所有内容创建 Android 和 iOS 应用程序。有关移动应用程序安全测试的更多内容参见第 14 章。此外，还可以查阅 OWASP Mobile Top 10 列表、移动安全测试指南（mobile security testing guide）以及移动应用程序安全验证标准（mobile application security verification standard）等。

在最近的一项评估中，作者发现一个应用程序将 PHI 发送到了云端，而操作该设备的医生/护士均毫不知情。尽管这不是一个技术漏洞，但仍然是利益攸关方应该知道的违反保密规定的行为。

此外，还需要评估与 IoT 系统相关的任何云组件的安全状况，检查云与 IoT 组件之间的交互。特别要注意后端 API 及其在云平台的实现，包括但不限于 AWS、Azure 和 Google Cloud Platform 等云平台。评估中通常会发现**不安全的直接对象引用（Insecure Direct Object References，IDOR）**漏洞，这种漏洞可以让知道 URL 的任何人都能访问敏感数据。例如，有时 AWS 会允许攻击者使用与 bucket 包含的数据对象关联的 URL 访问 S3 buckets。

云测试中涉及的许多任务将与移动和 Web 应用程序评估重叠。在前一种情况下，原因是使用这些 API 的客户端通常是 Android 或 iOS 应用程序。而后一种情况，原因是许多云组件基本上都是 Web 服务。云测试还要对云端的远程维护和支持连接进行检查，如 3.5 节

所述。

作者碰到过很多与云相关的漏洞：硬编码的云令牌、嵌入在移动应用和固件二进制文件中的 API 密钥、缺少 TLS 证书锁定(TLS-certificate pinning)，以及由于配置不当而向公众泄露的 Intranet 服务(例如未经身份验证的 Redis 缓存服务器或元数据服务)。注意，任何云测试都必须获得云服务所有者的许可才能进行。

结语

本书的多位作者都曾在重要的网络安全部门服务，也都了解到尽职调查是信息安全最重要的方面之一。遵循一套安全测试原则对于避免一些明显的错漏非常重要。不要因为这些方法看起来太过简单或显而易见，就错过了唾手可得的成果。

本章概述了一套用于实施 IoT 系统安全评估的测试原则，一步步地演练了被动侦察，然后描述并细分为物理层、网络层、Web 应用程序、主机、移动应用程序和云端进行介绍。

注意，本章涉及的这些概念层不是绝对的；两层或多层之间通常有很多的重叠。例如，电池耗尽攻击(battery exhaustion attack)可能是物理层评估的一部分，因为电池属于硬件；但也可能是网络层的一部分，因为攻击者可以通过组件的无线网络协议进行攻击。要评估的组件列表也不是详尽无遗的，这就是为什么本书附录推荐了其他扩展资源的原因。