医疗器械 PC 端软件的网络安全测试技术分析

文◆广安门医院济南医院　冯金玲

引言

在医疗设备智能化趋势下，医疗器械 PC 端软件不仅能够处理敏感医疗数据，还能够直接控制设备运行。当前医疗软件面临的网络安全威胁日益复杂，包括黑客攻击、数据泄露、通信协议漏洞等。建立专业的网络安全测试体系，既是技术发展的必然要求，也是保障医疗安全的重要防线。本文聚焦该领域网络安全测试技术，分析渗透测试、漏洞扫描、通信安全验证以及代码审计四大核心技术。通过解析外部攻击、内部威胁以及软件缺陷等风险，提出多技术协同的测试策略，构建覆盖软件全生命周期的安全防护体系。研究证实，专业化测试可显著提升医疗软件安全防御能力，为行业标准化测试流程提供实践指导。

1 医疗器械 PC 端软件网络安全测试概述

1.1 网络安全测试的重要性

医疗器械 PC 端软件作为医疗业务流程的核心载体，直接介入患者诊疗数据的采集、处理与存储环节，其网络安全状态与患者生命健康、医疗体系数据资产安全形成强绑定关系。当软件遭受网络攻击或出现安全漏洞时，则会引发诊疗指令误判、设备控制异常等风险。例如，远程恶意代码注入会导致影像诊断系统数据篡改，进而影响医生临床决策，甚至诱发误诊误治等严重医疗事故。同时，医疗数据具有高度敏感性，包含患者身份信息、诊疗记录等隐私内容，若因网络安全防护失效导致数据泄露，不仅违反《中华人民共和国数据安全法》《中华人民共和国个人信息保护法》等法律法规，还会对患者隐私权造成侵害，引发公众对医疗行业的信任危机。从行业发展层面看，构建完善的网络安全测试体系，能够提前识别软件在网络通信、数据交互、权限管理等环节的安全隐患，通过修复漏洞、优化架构等技术手段，保障软件在复杂网络环境下的稳定运行，为智慧医疗系统的规模化应用筑牢安全底座 [1]。

1.2 常见网络安全风险

医疗器械 PC 端软件面临的网络安全风险呈现多元化特征，其中外部攻击威胁首当其冲。黑客常利用软件暴露的网络端口、未修复的系统补丁等薄弱环节，实施渗透攻击与恶意代码植入。例如，钓鱼邮件携带木马程序，一旦用户点击执行邮件，即可获取软件后台管理权限，进而对患者电子病历数据库进行窃取或篡改；勒索软件则通过加密关键业务数据，向医疗机构索要高额赎金，导致系统瘫痪与诊疗服务中断。而内部威胁表现在医疗人员操作失误或权限滥用可能引发数据安全事件，如误将包含患者隐私的数据包发送至非安全网络区域或因账号密码管理不善导致未授权用户访问系统。同时，部分软件仍采用安全性较低的传输

【作者简介】冯金玲（1980—），女，山东济南人，本科，工程师，研究方向：机械电气。

协议，如未加密的 HTTP 协议传输患者影像数据，容易被中间人攻击截取并篡改；即使使用 HTTPS 协议，若证书配置不当或加密算法存在漏洞，仍会导致数据传输过程中的机密性与完整性丧失 [2]。此外，软件设计缺陷更是长期潜伏的安全隐患，输入验证机制缺失会引发缓冲区溢出攻击，越权访问漏洞会导致低权限用户获取敏感操作权限，而会话管理漏洞则会被利用进行会话劫持，冒充合法用户执行恶意操作。

2 医疗器械 PC 端软件网络安全核心测试技术分析

2.1 渗透测试技术

2.1.1 黑盒测试的边界攻击实践

黑盒测试聚焦软件外部暴露面，通过模拟真实攻击场景验证边界安全性，测试人员利用 Nmap 等工具枚举开放端口与服务版本，针对 HTTP 接口实施 OWASP Top10 漏洞验证，如构造畸形 URL 触发路径遍历或通过 Burp Suite 篡改请求参数尝试越权访问。典型案例中，某影像管理软件因未限制 API 调用频率，被发现可通过自动化脚本暴力破解医生账号，存在暴露患者影像数据强制访问的风险。

2.1.2 白盒测试的代码级漏洞挖掘

白盒测试基于代码可读性优势，深入函数逻辑检测潜在缺陷，通过分析设备控制模块的 C 语言代码，可发现未对用户输入做长度校验的 strcpy 函数调用，存在缓冲区溢出风险。针对 Python 编写的后台服务，可定位未经验证的反序列化操作，防止恶意构造对象引发远程代码执行。结合 Coverity 等静态分析工具，能精准识别空指针解引用、竞争条件等底层代码问题，尤其关注多线程数据交互场景的同步机制漏洞。

2.1.3 灰盒测试的混合信息利用策略

灰盒测试结合部分系统设计文档与运行时状态，重点突破权限边界。例如，在测试医疗数据导出功能时，已知业务逻辑需调用加密中间件，可通过 Wireshark 抓取传输数据，验证 AES 加密密钥是否动态生成且符合国密标准。针对登录模块，利用已知的账号权限分级规则，构造跨角色权限调用请求，检测服务端是否缺失强制访问控制（MAC）机制，如普通护士账号能否绕过逻辑直接下载全院患者数据。

2.2 漏洞扫描技术

2.2.1 静态扫描的代码缺陷检测

静态扫描工具通过语法分析与规则匹配，在代码编译前识别结构性漏洞。SonarQube 可检测 SQL 语句硬编码问题，提示使用参数化查询防止注入攻击。针对医疗设备通信模块，需特别检查 DICOM 协议解析代码是否存在未处理的异常字段，避免恶意构造的畸形数据包导致程序崩溃。此外，对配置文件的扫描不可或缺，如 nginx.conf 是否禁用了 TRACEMethod，防止 HTTP 方法滥用引发的安全风险 [3]。

2.2.2 动态扫描的运行时漏洞验证

动态扫描在软件运行环境中模拟攻击行为，捕捉内存泄漏、逻辑错误等动态缺陷，可使用 OWASPZAP 对设备控制接口进行模糊测试，向串口通信函数发送随机字节流，观测是否出现缓冲区溢出导致的程序异常终止问题。针对 HTTPS 服务，通过模拟 TLS 握手失败场景，验证是否存在降级攻击漏洞，如客户端能否被强制使用 SSLv3 协议，进而利用 POODLE 漏洞解密数据。

2.2.3 立体化扫描策略的实施路径

立体化扫描需构建“开发阶段—集成测试—上线前”的全周期扫描体系，在开发阶段采用 IDE 插件实时检测代码提交中的高危函数（如 C++ 的 std::getline 未指定长度限制）。集成测试阶段部署 Nessus 扫描器，重点检测第三方组件漏洞，如某版本 OpenSSL 的心脏出血漏洞。上线前结合人工验证，对扫描结果中的“误报”进行二次确认，避免因过度依赖工具导致关键漏洞漏检，如医疗专用协议解析模块的定制化漏洞需人工比对协议规范。

2.3 通信安全测试技术

2.3.1 数据加密传输的协议安全性验证

针对医疗数据传输，需验证 TLS 协议配置的合规性，可使用 SSLLabs 工具检测服务器支持的加密套件，禁用 RC4 等弱算法，确保证书链完整且未被吊销。在 DICOM 图像传输测试中，通过构造无效的 TLS 证书强制连接，观察软件是否具备证书校验熔断机制，防止中间人攻击篡改影像数据。同时，需测试加密性能，避免因 AES-GCM 算法调用异常导致的传输延迟影响设备控制实时性。

2.3.2 通信协议的合规性与健壮性测试

协议测试需对照 HL7FHIR、DICOM 等医疗标准，以 HL7 消息为例，故意构造缺少必选字段的异常消息，验证软件是否拒绝处理并记录日志。针对设备控制使用的 Modbus TCP 协议，模拟连续发送未认证的写寄存器指令，

检测是否触发访问控制机制，防止恶意设备伪造指令篡改治疗参数。此外，需验证协议版本兼容性，如旧版软件能否正确解析新版 DICOM 文件的加密元数据。

2.3.3 身份认证与访问控制的旁路攻击测试

在双因素认证（2FA）测试中，尝试绕过短信验证码机制，如通过拦截通信接口伪造验证码接收响应或利用会话固定攻击劫持已认证的会话。对于基于角色的访问控制（RBAC）系统，需验证权限边界是否清晰。例如，低权限的护士账号是否可通过修改 HTTP 请求中的用户 ID 参数，访问其他患者的诊断报告。此外，针对医疗软件的本地认证（如 Windows 登录集成），需检测是否存在缓存凭证泄露风险，如通过内存 dump 工具提取明文密码。在生物识别认证场景中，需测试指纹识别模块的抗伪造能力，使用硅胶指纹膜模拟非法身份验证，验证系统是否具备活体检测机制。

2.4 安全代码审计技术

2.4.1 自动化工具的规则化漏洞扫描

在 Java 开发的医疗信息系统中，Checkmarx 可检测不安全的反序列化漏洞（如使用 Apache Commons Collections 库的默认配置），此类漏洞会导致远程代码执行。针对 Python 代码，Bandit 工具可识别加密函数的不当使用，如使用 MD5 进行密码哈希存储。此外，针对医疗软件特有的数据脱敏模块，需定制化规则检测敏感信息（如身份证号、医保卡号）的脱敏策略，确保掩码处理符合 GB/T35273 等个人信息保护标准，避免因正则表达式错误导致部分敏感字段未被过滤。自动化工具的输出需结合人工复核。例如，误报的“未使用 HTTPS”警告可能因测试环境配置差异产生，需排除环境因素干扰 [4]。

2.4.2 人工审计的逻辑漏洞挖掘

人工审计侧重业务逻辑缺陷，如预约系统的时间戳校验是否存在跨日漏洞，导致患者预约时间被错误覆盖；在设备校准功能中，检查是否缺少二次确认机制，防止未授权用户通过伪造 HTTP POST 请求触发设备参数重置。针对权限管理，需逐行审查 ACL（访问控制列表）配置，确认“只读账号”是否被错误赋予数据修改权限，尤其关注复杂业务流程中的权限传递逻辑，如实习生账号通过多级跳转获取管理员操作界面的漏洞。

2.4.3 代码缺陷的修复与防护体系构建

对于缓冲区溢出漏洞，需采用安全编码实践（如使用 strncpy 替代 strcpy），并启用编译器防护技术（如 GCC 的 –fstack–protector）。针对 SQL 注入风险，推广参数化查询（如 PreparedStatement）而非拼接 SQL 语句。修复完成后需进行回归测试，验证补丁是否引入新漏洞。例如，某 PACS 系统修复文件上传漏洞后，应测试是否导致合法 DICOM 文件无法正常导入。同时，构建持续集成 / 持续部署（CI/CD）管道中的安全门禁，将代码扫描工具集成至 Jenkins 流水线，强制要求漏洞修复率达到 100% 方可发布。此外，针对医疗行业的长期维护需求，需建立代码版本控制系统（如 Git）的安全分支策略，确保历史版本漏洞在补丁回退时可追溯。例如，为遗留系统创建独立的安全维护分支，定期同步通用漏洞库（CVE）的修复方案。

结语

医疗器械 PC 端软件的网络安全测试，是技术规范与医疗伦理的交汇点。通过渗透测试模拟攻击场景、漏洞扫描实现风险可视化、通信协议验证保障数据传输安全以及代码审计筑牢安全防线，可构建多层次防御体系。实践表明，将测试技术融入软件研发全流程，是应对网络威胁的有效途径。医疗行业需持续完善测试标准，推动技术创新与临床需求融合，实现安全与效率的平衡发展。

引用

[1] 吴王震,李敏.医疗器械PC端软件网络安全测试方法研究[J].电子质量,2025(2):19-22.

[2] 林香,侯建勋,古锐鹏,等.浅谈医疗器械软件网络安全[J].网络安全技术与应用,2024(8):109-110.

[3] 郝鹏飞,李庆雨,柴蕊,等.医疗器械软件信息安全现状分析[J].中国医疗设备,2023,38(7):120-123.

[4] 肖斌,陆晓琳.医疗器械的网络安全质量控制途径分析[J].中国设备工程,2023(10):247-249.