MeDUSA 应用全套测试操作指南（详细版）

CT24 — 应用静态分析（MobSF）

【测试目的】

识别应用源码中的安全弱点，包括不安全 API、错误配置、敏感信息泄漏等。

【测试工具】

- Mobile Security Framework (MobSF)
- Visual Studio Code（VSCode）

【测试步骤】

1. 启动 MobSF（推荐 Docker）：

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

2. 浏览器访问 http://localhost:8000。

3. 上传 APK 或源码，生成分析报告。

4. 查看以下关键内容：

- Manifest 权限、导出组件
- 网络安全配置（Network Security Config）
- WebView 使用
- 硬编码字符串

5. 打开 VS Code 对相关文件逐条验证。

【通过标准（Pass）】

无中高危问题。

【失败标准（Fail）】

发现危险 API、调试模式开启、敏感数据泄漏等。

CT28 — 配置文件硬编码 Secret 检查

【测试目的】

检查是否存在硬编码密码、Token、密钥等敏感信息。

【测试工具】

- MobSF
- VS Code

【测试步骤】

1. 在 MobSF 报告中查看 Hardcoded Secrets。
2. 在 VS Code 全局搜索：

   password、secret、apikey、bearer、token。
3. 确认是否为真实敏感内容。

【Pass】

无明文硬编码敏感信息。

【Fail】

发现真实 Token/密码硬编码。

CT37 — 外网可访问性检查

【测试目的】

确认 App 服务是否意外暴露在公网。

【测试工具】

- nmap

【测试步骤】

1. 获取应用后端公网 IP 或域名。

2. 运行端口扫描：

```
nmap -Pn -sV <public-ip>
```

3. 检查是否暴露非预期端口。

【Pass】

仅有 443 等必要端口。

【Fail】

出现额外端口或管理接口暴露。

CT38 — TLS 协议与 Cipher 测试

【测试目的】

确保仅启用 TLS1.2/1.3 且无弱算法。

【测试工具】

- testssl.sh

【测试步骤】

1. 下载 testssl.sh

2. 运行：

./testssl.sh <domain>

重点检查：

- TLS1.0/1.1 禁用

- 禁用 SHA1/3DES/CBC/RC4 等弱算法

- Cipher 是否提供前向保密（PFS）

【Pass】

仅 TLS1.2/1.3 + 强 Cipher。

【Fail】

存在弱 Cipher 或支持旧协议。

CT39 — 抗压/抗 DoS 能力测试

【测试目的】

检验应用在高流量下能否正常运作。

【测试工具】

- Baton

【测试步骤】

1. 对应用域名施压：

   baton -u https://your-domain

2. 压测期间：

   - 同 IP 再次访问

   - 另一个 IP 同时访问

3. 检查是否出现超时或服务不可用。

【Pass】

服务稳定，响应正常。

【Fail】

延迟大幅上升、服务崩溃或拒绝访问。

CT40 — 网络栈漏洞扫描（Nessus）

【测试目的】

检查服务器操作系统与网络协议栈是否存在已知漏洞。

【测试工具】

- Nessus

【测试步骤】

1. 在 Nessus 新建“Basic Network Scan”。

2. 目标填写应用域名或 IP。

3. 扫描后查看：

  - OpenSSL 漏洞

  - 信息泄漏

  - 弱服务配置

【Pass】

无高危/中危漏洞。

【Fail】

发现已知 CVE 或敏感版本信息。

CT43 — HTTP Header 安全配置检查（ZAP）

【测试目的】

确认 HTTP 请求与响应 Header 是否符合 OWASP 最佳实践。

【测试工具】

- OWASP ZAP

【测试步骤】

1. 使用 ZAP 对域名执行 Active Scan。

2. 在报告中查看 Header 相关项。

3. 核查是否缺少：

  - Content-Security-Policy

  - X-Frame-Options

  - X-Content-Type-Options

  - HSTS

【Pass】

所有重要安全 Header 已配置。

【Fail】

Header 缺失或配置不安全。

CT46 — 账户锁定机制测试

【测试目的】

验证系统是否存在防暴力破解保护。

【工具】

- 手动

【测试步骤】

1. 对某个账号连续输入错误密码（≥5 次）。

2. 检查是否触发锁定提示。

3. 在锁定期间尝试正确密码，应该仍被禁止登录。

【Pass】

错误登录次数达到阈值后账号被锁定。

【Fail】

无限制尝试仍可继续输入密码。

CT47 — 表单超长输入 / Overflow 测试

【测试目的】

检查 App 对超长输入的处理是否安全。

【工具】

- Burp Suite

【测试步骤】

1. 在前端输入 10k 字符以上的内容。

2. 使用 Burp 拦截并修改请求继续增大 Payload。

3. 检查是否出现：

- 500 错误

- Stack trace

- 信息泄漏

【Pass】

返回通用错误信息，无异常暴露。

【Fail】

服务崩溃或泄漏内部错误。

CT49 — 登出后 Token 是否失效

【测试目的】

确保 JWT/Session Token 在登出后立即失效。

【工具】

- Burp Suite

【测试步骤】

1. 登录 App，抓取 Token。

2. 执行 Logout。

3. 使用旧 Token 重新访问 API。

4. 若仍可访问 → 漏洞。

【Pass】

Token 失效，访问被拒绝。

【Fail】

旧 Token 仍可正常使用。

CT51 — MFA 功能检查

【测试目的】

确认是否支持双因素认证（2FA）。

【工具】

- 手动

【测试步骤】

1. 登录账号 → 进入安全设置。

2. 查看是否有“开启 MFA/2FA”选项。

3. 尝试启用。

【Pass】

支持且可以启用 MFA。

【Fail】

系统不支持 MFA。

CT52 — RBAC（基于角色的访问控制）测试

【测试目的】

确认不同角色是否严格遵守权限边界。

【工具】

- ZAP Spider

【测试步骤】

1. 用不同角色（管理员、普通用户）登录。

2. 使用 ZAP Spider 枚举所有可见 API。

3. 使用低权限用户访问高权限端点。

4. 检查是否返回 403。

【Pass】

低权限无法访问敏感端点。

【Fail】

越权访问成功。

CT53 — 密码策略检查

【测试目的】

确认密码策略是否正确执行。

【工具】

- 手动

【测试步骤】

1. 尝试设置弱密码（如 12345）。

2. 尝试使用和旧密码相同的密码。

3. 尝试少于规定长度的密码。

【Pass】

系统正确拒绝弱密码。

【Fail】

弱密码可以被设置。

CT93 — HTTP → HTTPS 强制跳转

【测试目的】

确认系统是否强制使用 HTTPS。

【工具】

- curl

【测试步骤】

运行：

curl -I http://your-domain/api

检查是否返回 301/302 跳转到 HTTPS。

【Pass】

自动跳转至 HTTPS。

【Fail】

HTTP 请求直接成功。

CT100 — API 注入漏洞测试（XSS/SQLi）

【测试目的】

验证 API 是否存在注入漏洞。

【工具】

- ZAP

- Postman

- Python 注入脚本

【测试步骤】

1. 准备常见注入 Payload。

2. 使用 Postman 向所有 API 提交 Payload。

3. 检查是否出现：

- Stack trace

- Payload 回显

- 系统异常

【Pass】

系统拒绝并返回通用错误。

【Fail】

响应中出现回显或异常泄漏。

CT101 — Token 自动过期测试

【测试目的】

验证 Token 是否按预期时间过期。

【工具】

- Postman

【测试步骤】

1. 登录获取 Token。

2. 调用一次 API（成功）。

3. 等待 Token 过期时间。

4. 重新调用同一 API。

【Pass】

过期后访问被拒绝。

【Fail】

Token 过期仍可使用。

CT102 — API 超长输入测试

【测试目的】

确认 API 是否安全处理大输入。

【工具】

- Postman

【测试步骤】

1. 使用正常 Token 调用 API。

2. 将参数替换为超长（10k+ 字符）。

3. 查看响应是否安全。

【Pass】

返回通用错误，无内部信息泄漏。

【Fail】

返回异常或泄漏内部错误堆栈。

CT127 — 恶意代码扫描（ClamAV）

【测试目的】

检测源码中是否包含已知恶意签名。

【工具】

- ClamAV

【测试步骤】

1. 更新签名库：

   freshclam

2. 扫描：

   clamscan -r <project_path>

【Pass】

未发现恶意签名。

【Fail】

检测到恶意代码。