Mobile Application Penetration Testing Process

A combination of manual and automated tests is used in all engagements.

The engagement begins with some simple standard test steps, such as self-signed certificate rejection, root/jailbreak detection, SSL pinning checks and automatic decompiling and static code analysis with MobSF and QARK.

Standard checks are followed by SSL pinning circumvention attempts using Xposed Framework and Frida in Android and SSL Killswitch-2 in IOS.

Then a detailed walk-through of the mobile application is performed, with an intercepting proxy attached, which records and analyzes the communication between the client and the backend. In most assignments the tool of choice is Burp Suite Professional, however for applications making heavy use of Web Sockets technology, OWASP's Zed Attack Proxy (ZAP) may be preferred. The walk-through step also provides information about the back-end servers and technologies used, which then aids in choosing the right tools for the engagement.

An additional back-end infrastructure discovery is performed using Maltego. This activity aims to reveal all the sub-domains, servers, IP addresses and services in connection to the test subject. Port scan & service discovery is additionally performed with NMAP, to complement Maltego's results. Where necessary, customer's approval is sought to expand the test scope. This step is complemented with reviewing the decompiled source code of the binaries, to reveal URLs and sensitive information such as API keys, hard-coded credentials, etc.

The next step is automated spidering and discovery of the application's backend URLs and components, with the following tools:

- Burp Suite Professional's Content Discovery module
- OWASP DirBuster
- WFUZZ
- DirB
- WhatWeb

An additional scan is performed with a tool called Arjun, to enumerate hidden API parameters in API backends.

After the discovery, an automated vulnerability scan is performed, using tools such as:

- Nikto
- Nessus

In some cases, specialized vulnerability scan tools may also be deployed, such as CMSeek, WPScan for WordPress, Droopescan for Drupal and JoomScan for Joomla.

Having information on the components and libraries used in the application, the tester also deploys relevant Burp Suite Professional plug-ins or other tools, to enhance the accuracy of the automated tests.

Following the automated vulnerability scans, the tester initiates the detailed application tests. The tester examines the communication (every request-response pair), initially focusing on functions like authentication, session lifecycle, and authorization then proceeds to test input validation (looking for XSS, SQL Injection and similar flaws), also looks for logic errors and possible insecure direct object reference cases. At this step, all the functionality available in the application is examined one by one.

The tests cover, among others:

- Authentication and authorization (unauthorized access to data, unauthorized transactions, privilege escalation, user role controls, authentication bypass, password reset tests, insecure direct object reference tests, CAPTCHA use and bypass, rate-limiting controls, etc.)
- Session management and security (Session fixation, cookie randomness, CSRF, session termination, etc.)
- Input controls (SQL, code, OS command injections, XSS, URL redirection, file uploads, etc.)
- Information leakage (information gathering on software, OS, platforms/frameworks used, supported HTTP methods, the discovery of critical/sensitive pages such as administration interfaces, user enumeration, etc.)
- Web service tests (SOAP and Ajax)

While every request-response pair is examined manually, they are also submitted to automated scans in Burp Suite Professional. Suspected parameters are additionally fuzzed both manually and with the help of tools such as SleuthQL (to optimize the efforts, especially when the application is very large, and the number of suspected SQL-injection cases are very high) and SQLMAP, where necessary.

Finally, additional binary tests are performed where necessary, using tools like Frida and various log analysis tools.

Our standard checklist is strictly followed during the detailed application test, to assure that no critical checks are accidentally skipped. The checklist is created based on OWASP cheat sheets and checklists, OSSTMM (The Open Source Security Testing Methodology Manual), ISSAF (Information Systems Security Assessment Framework), SANS Top 25 Software Errors and NIST SP800-115.

Also, following every assignment, a “lessons learned” session is performed, where new checks are added. A similar activity is performed for every security training received, to further enhance the existing checklist.

Deliverable

The final output of the assignment is a formal penetration test report, containing all identified vulnerabilities, along with risk grading for each issue, potential impacts of the vulnerabilities, recommendations on how to remediate the problem, evidence and supporting documentation, such as screenshots, request-response pairs and necessary steps to replicate the attack.

The report also contains an Executive Summary to guide the senior management about where the weak points are and what impacts they may have on the business, as well as a Risk Classification section that explains the rationale behind the risk assessments.

After the recipient remediates the reported issues and the test team performs a verification study, a redacted version of the report can be produced, for the recipient to share with prospects and clients.