모바일 앱 구분과 보안 취약점 분석 보고서

1. 모바일 앱의 구분

모바일 애플리케이션은 구현 방식과 배포 형태에 따라 크게 **네이티브 앱**, **모바일 웹 앱**, **하이브리드 앱**으로 나눌 수 있습니다.

구분	설명	장점	단점
네이티브 앱	iOS, Android 등 특정 플랫폼 SDK와 언어(Swift, Kotlin 등)로 개발	성능 우수, 기기 기능(카메라, GPS 등) 활용 용이	플랫폼별 별도 개발 필요, 유지보수 비용 증가
모바일 웹 앱	브라우저에서 동작하는 웹 기반 앱 (HTML, CSS, JS)	크로스플랫폼 지원, 개발 속도 빠름	네트워크 의존도 높음, 기기 기능 접근 제한
하이브리드 앱	네이티브와 웹 앱을 결합, 웹뷰(WebView) 기반	단일 코드로 다중 플랫폼 지원, 유지보수 용이	네이티브보다 성능 저하 가능, 보안 이슈 발생 가능

2. 각 앱 유형별 보안 취약점

(1) 네이티브 앱 취약점

- **취약점 예시**
  - 디컴파일/리버스 엔지니어링을 통한 코드 유출
  - 인증/인가 로직 우회
  - 로컬 저장소의 민감 데이터 노출 (SQLite, SharedPreferences)
  - 취약한 API 호출 (HTTPS 미사용, 인증 토큰 하드코딩)

대응 방안

- 코드 난독화 및 바이너리 보호
- 안전한 인증/인가 구현 (OAuth2, JWT)
- 민감 데이터 암호화 저장
- 안전한 네트워크 프로토콜(HTTPS/TLS) 사용

(2) 모바일 웹 앱 취약점

- **취약점 예시**
  - XSS(교차 사이트 스크립트)
  - CSRF(사이트 간 요청 위조)
  - SQL Injection
  - 세션 하이재킹(Session Hijacking)

대응 방안

- 입력값 검증 및 출력 시 이스케이프 처리
- CSRF 토큰 적용
- 보안 HTTP 헤더 적용(Content-Security-Policy 등)
- 세션 타임아웃 및 안전한 쿠키 설정(HttpOnly, Secure)

(3) 하이브리드 앱 취약점

- **취약점 예시**
  - WebView를 통한 자바스크립트 인터페이스 악용
  - 인텐트(Intent) 스니핑/변조
  - 파일 접근 권한 남용
  - 웹 취약점과 네이티브 취약점의 결합 공격

대응 방안

- WebView에서 addJavascriptInterface사용 최소화
- 인텐트 필터 제한 및 암호화
- 필요 최소한의 권한만 부여
- 웹 보안 가이드라인과 네이티브 보안 가이드라인 동시 준수

3. 개인 의견

네이티브 앱은 **바이너리 보호**와 **데이터 암호화**가 핵심입니다. 리버스 엔지니어링 방어 없이는 경쟁사나 공격자가 손쉽게 로직과 키를 확보할 수 있습니다.
모바일 웹 앱은 서버·클라이언트 전 구간에서 **입력 검증**이 중요합니다. 특히 프런트엔드 보안만으로는 불충분하므로 **백엔드에서도 동일한 검증 로직**을 적용해야 합니다. 세션 보안이 취약하면 공격자가 기기 없이도 계정에 접근 가능하므로, **토큰 재발급 주기와 만료 시간 관리**가 필요합니다.
하이브리드 앱은 **웹 취약점과 네이티브 취약점이 동시에 존재**할 수 있다는 점이 가장 큰 위험 요소입니다.
세 가지 앱 유형 모두 **공격 표면이 다르지만 취약점은 상호 보완적으로 악용**될 수 있습니다.
개발 단계에서부터 **보안 코딩, 취약점 점검, 모의해킹**을 정기적으로 수행하는 것이 필요하며, 배포 후에도 **지속적인 보안 업데이트**가 필수적입니다.

4. 참고 자료

https://owasp.org/www-project-mobile-top-10/
https://developer.android.com/topic/security
https://developer.apple.com/security/
https://csrc.nist.gov/publications