Draft NIST Special Publication 800-163

# 모바일 애플리케이션 보안 심사 가이드라인

## Vetting the Security of Mobile Applications


# 권 리

이 문서는 NIST(National Institue of Standards and Technonolgy)에서 발행한 "Vetting the Security of Moblie Applications(Draft NIST Special Publication 800-163 Revision 1)"을 한국어로 번역한 결과물이다. 이 문서와 관련된 모든 권리는 NIST에 있다.

• 저자
  - Michael Ogata, NIST
  - Josh Franklin, NIST
  - Jeffrey Voas, NIST
  - Vincent Sritapan, U.S. Department of Homeland Security
  - Stephen Quirolgico, U.S. Department of Homeland Security

• 번역 : Youngeun Moon, Blackfalcon Security

# 개 요

모바일 애플리케이션은 일상 생활에서 필수적인 부분이 되었다. 공공 및 민간 조직 모두 모바일 애플리케이션에 의존하고 있다. 따라서 모바일 애플리케이션을 취약점 및 결함으로부터 보호하는 것이 더욱 중요해지고 있다. 이 문서에서는 모바일 애플리케이션 심사 프로세스에 대해 간략히 설명한다. 이 프로세스는 모바일 애플리케이션이 조직의 보안 요구사항을 준수하고 있음을 보장하고, 취약점을 식별하기 위해 사용할 수 있다.

# 목 차

## 부록 목차



## 그림 목차



## 표 목차

# 1 서론

모바일 어플리케이션(이하 모바일 앱)은 조직에 변화를 가져오고 있다. 모바일 앱의 기능은 지속적으로 추가되고 있으며, 시간과 장소과 관계없이 업무 핵심 정보에 신속하게 접근할 수 있다. 이를 통해 모바일 앱은 조직 목표 달성을 지원하고 있다. 그러나, 모바일 앱에 취약점이 존재할 경우, 조직 및 사용자는 심각한 보안 위험에 노출될 수 있다. [1)] 이러한 취약점은 정보 탈취, 사용자 기기 제어, 하드웨어 자원 고갈, 앱이나 기기의 예상치 못한 동작 등을 유발한다.

취약점은 의도적이거나 부주의함에서 비롯된 설계의 결함 및 프로그래밍 오류 등 여러 요인에 의해 발생한다. 앱 스토어에 취약점이 존재하는 앱이 만연한 것은 개발자가 비용을 절감하고 출시 시간을 단축하기 위해 보안보다 기능 구현에 치중한 결과이다.

취약점에 의한 위험 수준은 앱에서 접근할 수 있는 데이터를 비롯한 여러 요인에 따라 달라진다. 예를 들어 위치 정보, 개인 건강 측정 정보 또는 개인식별정보(PII)와 같은 데이터에 접근하는 앱은 민감정보에 접근하지 않는 앱보다 더 위험할 수 있다. 또한 데이터 전송을 위해 무선 네트워크(예: Wi-Fi, 데이터 통신, 블루투스)를 사용하는 앱은 무선 네트워크 기술이 원격 공격의 매개체로 사용될 수 있어 위험할 수 있다. 위험성이 낮은 앱도 악용될 경우 심각한 영향을 미칠 수 있다. 예를 들어, 공공 안전 앱에 취약점이 존재하고 악용될 경우, 소중한 생명을 빼앗아 갈 수 있다.

모바일 앱과 관련된 잠재적인 보안 위험을 완화하기 위해 조직은 소프트웨어 보증 프로세스를 채택해야 한다. 소프트웨어 보증 프로세스란, 소프트웨어에 취약점이 존재하지 않고, 올바르게 동작한다는 것을 보장하는 절차이다. 이 문서에서는 모바일 앱의 소프트웨어 보증 프로세스를 정의하며, 이를 앱 심사 프로세스로 지칭한다.

## 1.1 목적

이 문서는 앱 심사 프로세스를 정의한다. (1) 앱 심사 프로세스의 계획 및 구현, (2) 모바일 앱의 보안 요구 사항 개발, (3) 모바일 앱 테스트 도구의 식별, (4) 모바일 앱 배포의 적절성을 결정하기 위한 가이드를 제공한다. 또한, 소프트웨어 보증 전문가가 일반적으로 사용하는 기술을 개략적으로 설명한다. 여기에는 개별 소프트웨어 취약점 테스트 방법 및 모바일 앱 소프트웨어의 설정 오류와 관련된 내용이 포함되어 있다.

1) 취약점은 우연히 유발되거나 의도적으로 악용되어, 시스템이 설계와 다르게 동작하는 것을 유발하는 하나 이상의 약점으로 정의된다. [1]

## 1.2 범위

모바일 앱에 대한 소프트웨어 보증 활동은 모바일 앱 수명주기의 여러 단계에서 수행된다. 모바일 앱 수명주기는 (1) 앱 개발 단계, (2) 앱 획득 단계, (3) 앱 배포 단계로 구분된다. 그림 1은 모바일 앱 수명주기의 3단계를 보여준다.

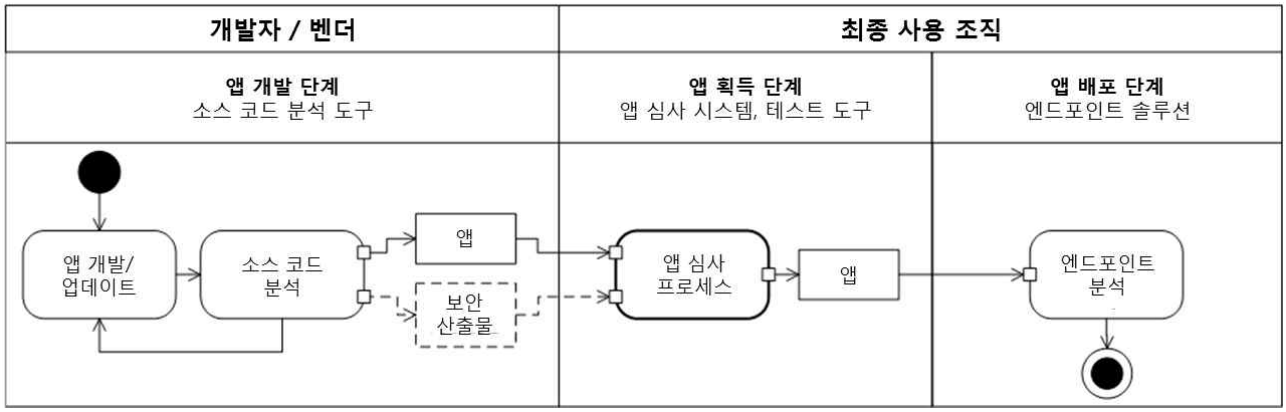


그림 1 - 모바일 앱 수명주기에 따른 소프트웨어 보증 활동

이 문서는 앱 심사 프로세스에서의 소프트웨어 보증 활동에 중점을 두고 있다. 앱 심사 프로세스는 모바일 앱 수명주기에서 앱 획득 단계의 일부로 정의한다. 따라서 앱의 개발 단계(예: 소스 코드 분석 도구) 또는 앱의 배포 단계(예: 엔드포인트 솔루션) 중에 수행되는 소프트웨어 보증 활동에 대해 설명하지 않는다.

또한 이 문서는 EMM(Enterprise Mobility Management), 모바일 앱 관리 또는 모바일 위협 방어 시스템의 사용에 대해 설명하지 않는다. 그러나, 이러한 시스템과의 통합에 대해 간략하게 언급한다. 또한, 사물인터넷(IoT) 앱의 보안 점검에 대해 언급하지 않으며, 모바일 플랫폼 및 운영체제의 보안에 대해 언급하지 않는다. 이러한 주제는 다른 참고자료를 [3]-[5] 참고하기 바란다. 마지막으로, 앱의 백엔드(back-end) 처리에 사용되는 웹 서비스 및 클라우드 인프라의 보안에 대해서도 설명하지 않는다.

## 1.3 대상 독자

이 문서는 모바일 기기에 배포된 모바일 앱의 소프트웨어 보증을 개선하고자 하는 공공 및 민간 조직을 대상으로 한다. 구체적인 대상은 다음과 같다.

- 조직의 모바일 기기에 대한 보안 경계태세를 설정하는 책임자
- 조직 모바일 기기 관리 및 보안 책임자, 조직에서 사용하는 앱을 결정하는 책임자
- 앱 심사 프로세스에서 제공하는 보증의 유형에 관심이 있는 인원

## 1.4 문서 구조

이 문서의 나머지 부분은 다음과 같이 구성되어 있다.

- 2장 - 앱 보안 요구사항
- 3장 - 앱 심사 프로세스
- 4장 - 앱 테스트 접근방법 및 취약점 분류
- 5장 - 앱 심사 시 고려사항
- 6장 - 앱 심사 시스템
- 부록 1 - 모바일 앱 위협
- 부록 2 - Android 앱 취약점 유형
- 부록 3 - iOS 앱 취약점 유형
- 부록 4 - 약어
- 부록 5 - 용어
- 부록 6 - 참고 문헌

## 1.5 문서 규칙

모바일 플랫폼용으로 특별히 작성된 애플리케이션을 앱(apps)이라고 지칭한다.

# 2 앱 보안 요구사항

모바일 앱을 심사하기에 앞서, 조직은 앱이 반드시 충족해야 하는 보안 요구사항을 정의해야 하며, 조직에 의해 승인되어야 한다. 이 문서에서는 두 가지 유형의 앱 보안 요구사항(일반 및 조직)을 정의한다.

## 2.1 일반 요구사항

일반 요구사항은 앱의 보안을 보장하기 위해 있어야 반드시 존재해야 하거나, 반드시 제거되어야 하는 앱의 소프트웨어적 특성 및 행동적 특성을 정의한다. 이러한 요구사항은 모든 모바일 앱에 적용할 수 있기 때문에 "일반"으로 간주한다. 일반 보안 요구사항은 다수의 표준, 모범사례 및 NIAP, OWASP, MITRE 및 NIST에서 지정한 자료를 이용하여 작성할 수 있다. [2)]

### 2.1.1 NIAP (National Information Assurance Partnership)

NIAP 보호 프로파일은 IT 제품을 대상으로 하는 보안 요구사항을 명시하고 있다. 이 보안 요구사항은 특정 제품에 종속적이지 않다. 특히, NIAP 보호 프로파일은 국가 보안 시스템에서 사용할 제품의 인증을 위해서 사용된다. 또한, NIAP 보호 프로파일에 의한 제품 평가가 ISO/IEC 15408 인증[6]으로 간주될 수 있도록 제품이 충족해야 하는 보안 목적, 요구사항 및 보증 활동을 상세하게 정의한다. 애플리케이션 심사(모바일 앱 심사 포함)를 위해, NIAP은 애플리케이션 보호 프로파일을 정의하고 있다. [7]

NIAP 애플리케이션 보호 프로파일에 정의된 요구사항은 크게 두 가지로 구분된다.

1) 기능 요구사항 - 특정 소프트웨어가 반드시 보유해야 하거나 제거해야 하는 동작 또는 속성
2) 보증 요구사항 - 성공적인 점검을 위해 평가자가 반드시 수행해야 하는 조치 또는 평가자가 반드시 준수해야 하는 규정

표 1은 NIAP 기능 요구 사항을 요약한 것이다. [3)]

[2)] 추가적인 위협과 취약점은 부록(A, B, C)에서 확인할 수 있다.
[3)] 표 1은 모든 기능 요구사항을 명시하고 있지 않다.

| 기능 요구사항 |
| --- |
| 플랫폼 리소스에 대한 액세스 |
| 익스플로잇에 대한 대응 능력 |
| 암호키 기능 |
| 암호 연산 |
| 중요한 애플리케이션 데이터의 암호화 |
| HTTPS 프로토콜 |
| 설치 및 업데이트를 위한 무결성 |
| 네트워크 통신 |
| 전송 중 데이터 보호 |
| 랜덤 비트 생성 |
| 보안 관련 기본 설정 |
| 소프트웨어 식별 및 버전 |
| 관리 기능 명세서 |
| 크리덴셜 저장소 |
| 지원되는 구성 메커니즘 |
| 전송 계층 보안 연산 |
| 지원되는 서비스 및 애플리케이션 프로그래밍 인터페이스의 사용 |
| 서드파티 라이브러리의 사용 |
| 개인식별정보의 전송에 대한 사용자 동의 |
| X.509 기능 |

표 1 - NIAP 기능 요구사항

보호 프로파일에 있는 보증 요구사항은 다음과 같이 요약할 수 있다.

- 애플리케이션에 고유한 참조번호를 명시해야 한다.
- 평가자는 평가 대상의 보안 기능(TSF, Target of Evaluation Security Function)이 명시된 대로 동작하는지 확인하기 위해, TSF의 하위 항목을 테스트해야 한다.
- 애플리케이션은 테스트에 적합(난독화 미적용)해야 한다.
- 평가자는 잠재적인 취약점을 식별하기 위해, 평가 대상이 공개된 소스 코드를 사용하는지 확인해야 한다.

### 2.1.2 OWASP 모바일 위험, 통제 및 앱 테스트 안내서

OWASP(Open Web Application Security Project)는 모바일 앱 테스트 및 모바일 앱 보안과 관련된 유용한 자료를 관리하고 있다. MASVS(모바일 앱 보안 검증 표준, Mobile Application Security Verification Standard)는 모바일 앱 보안을 위한 상세 모델이다. MASVS는 조직에서 기본적인 보안 요구사항을 설정하는데 사용할 수 있다. NIAP 보호 프로파일과 마찬가지로 MASVS는 앱의 구조 및 동작을 정의한다. 그러나, NIAP 보호 프로파일과 다르게, MASVS는 세 가지 레벨로 검증의 수준을 정의하고 있다.

- 레벨 1 : 표준 보안
- 레벨 2 : 심층 방어
- 레벨 3 : 역공학 및 위협에 대한 내성

각 단계의 통제 목록은 아래와 같이 구분되며, 각 통제 목록은 세부적인 요구사항으로 구성된다. 요구사항은 원하는 검증 단계에 따라 다르다.

- 아키텍처, 설계 및 위협 모델 요구사항
- 데이터 저장 및 개인정보보호 요구사항
- 암호화 요구사항
- 인증 및 세션 관리 요구사항
- 네트워크 통신 요구사항
- 플랫폼 통합 요구사항
- 코드 품질 및 빌드 설정 요구사항
- 내성 요구사항

OWASP MSTG(모바일 보안 테스트 안내서, Mobile Security Testing Guide)는 모바일 앱의 보안을 테스트하기 위한 매뉴얼이며, MASVS의 요구사항을 검증하기 위한 기술적인 절차에 대해 설명한다. [9]

### 2.1.3 MITRE 앱 평가 기준

2016년 MITRE는 기업에서 심사 프로세스의 일부를 자동화하는 것을 지원할 목적으로 모바일 앱 보안 심사 솔루션의 유효성을 분석했다. 분석을 수행하기 위해 MITRE는 NIAP의 애플리케이션 보호 프로파일을 기반으로 솔루션에 대한 기준을 마련했다. 또한, 범용적인 앱 심사 솔루션의 기능, 앱 심사 솔루션 자체에 대한 위협, 기타 일반적인 모바일 앱 취약점 및 악의적인 행위를 확인하기 위한 추가 기준을 마련했다.

이 기준을 사용하여 MITRE는 모바일 앱 심사 솔루션 평가에 사용할 앱(취약점이 존재하거나 악의적인 행위를 수행하는 앱)을 개발했다. MITRE는 이러한 앱을 사용하여 모바일 앱 심사 솔루션의 기능을 테스트했다.

MITERE는 당시 분석했던 솔루션에 사용한 방법론, 평가 기준, 테스트 앱 및 전체 분석 결과를 기술한 보고서를 발표했다. [10] 보고서 및 테스트 앱은 MITRE의 GitHub 사이트에서 확인할 수 있다.

### 2.1.4 NIST SP-800-53

NIST의 SP-800-53은 연방 정보 시스템에 적용하기 위해 고안된 보안 및 개인정보보호 통제 항목이다. [5] 또한, 악의적인 사이버 공격, 자연 재해, 구조적 문제 및 인적 오류와 같은 다양한 위협으로부터 IT 시스템, 개인 및 기타 조직의 자산을 보호하기 위한 통제 절차를 정의하고 있다. 정보 보안 및 개인정보 위험을 관리하기 위해 조직에 적합하게 통제 항목을 수정할 수 있기 때문에 조직의 필수 정책, 표준, 비즈니스 요구에 따라 다양한 보안 및 개인정보보호 요구사항을 충족시킬 수 있다. 영향도(상-중-하)에 따라 세 가지 수준의 보안 통제 기준이 제공된다. 또한, 전문화된 통제를 개발하는 방법에 대해 언급한다. 이를 이용하면 특수한 유형의 임무/비즈니스 기능 및 기술에 적합하게 통제 항목을 조정할 수 있다. NIST SP-800-53 보안 통제는 기능적 관점과 보증의 관점에서 보안 및 개인정보보호를 다룬다. 보안 기능과 보안 보증을 모두 다루는 경우, IT 제품과 IT 제품을 이용하여 구축한 정보 시스템이 정상적이며 보안 공학의 원칙을 이용하여 충분히 신뢰할 수 있음을 보증한다.

## 2.2 조직 요구사항

조직에 특화된 보안 요구사항(이하 조직 요구사항)은 조직의 보안 상태를 보증하기 위해 조직에서 반드시 준수해야 하는 정책, 규정 및 지침을 정의한다. 예를 들어, 소셜 미디어 앱과 특정 업체에서 개발한 앱을 조직의 모바일 기기에 설치하지 못하게 하는 것이 해당된다.

조직 요구사항을 작성할 경우, 취약하지는 않지만 모바일 앱의 보안 상태에 영향을 줄 수 있는 요인을 파악하면 유용할 수 있다. 이러한 요인은 표 2에 제시된 기준을 고려하여 도출할 수 있다.

| 분류 | 설명 | |
|---|---|---|
| 정책 | 보안, 개인정보보호 및 이용 목적 제한 방침 (AUP, Acceptable Use Policy), 소셜 미디어 가이드라인, 조직에 적용되는 규정 | |
| 출처 | 개발자, 개발자 조직, 개발자 평판, 고객 평가 등 식별정보 | |
| 데이터 민감도 | 앱에서 수집, 저장 또는 전송되는 데이터의 민감도 | |
| 앱 중요도 | 앱이 조직의 비즈니스에 미치는 중요도 | |
| 대상 사용자 | 조직의 앱 사용자 그룹 | |
| 대상 하드웨어 | 앱을 배포하는 대상 하드웨어 플랫폼, 운영체제 및 구성 | |
| 대상 환경 | 앱의 운영 환경 (예: 일반 사용자 vs 민감한 군사 환경) | |
| 전자서명 | 앱 바이너리, 라이브러리 또는 패키지에 적용된 디지털 서명 | |
| 앱 설명서 | 사용자 안내서 | 사용자 안내서는 앱의 기능 및 동작이 명시되어 있어 테스트에 도움이 된다. |
| | 테스트 계 획 | 테스트 계획 검토를 통해 누락되었거나 부적절하게 테스트된 영역을 식별함으로써 앱 심사의 방향을 설정할 수 있다. 개발자는 자체 테스트에 대한 자신들의 수고를 입증하기 위해, 특정한 상황에 대한 테스트 방향을 조언해 줄 수 있다. |
| | 테스트 결 과 | 코드 리뷰 결과 및 다른 테스트 결과는 어떤 보안 표준을 준수하고 있는지 보여 준다. 예를 들어, 앱 위협 모델을 작성한 경우, 준수한 표준을 명시한다. 이 표준을 통해 앱 설계 및 코딩 중에 식별되고 처리했어야 하는 약점이 식별될 것이다. |
| | 서비스 수준 협약 | 타사에서 조직의 앱을 개발한 경우, 서비스 수준 협약(SLA, Service Level Agreement)가 공급 업체와 계약의 일부로 포함되었을 수 있다. 서비스 수준 협약은 조직의 보안 정책을 만족해야 한다. |

표 2 - 조직 보안 기준

경우에 따라 앱 설명서에서 일부 정보를 수집할 수 있다. 하지만, 기술적으로 명확하지 않거나 앱을 구매할 사용자 조직에서 사용하는 특수한 용어를 사용할 수 있다. 앱 설명서는 앱에 따라 다른 방식으로 구성되기 때문에 평가를 위한 정보를 수집하는데 많은 시간이 소요될 수 있다. 따라서, 앱의 목적을 파악하고 개발자가 보안 취약점을 해결하기 위한 노력을 평가하기 위해서는 표준화된 설문지가 적절하다. 이러한 설문 조사는 소프트웨어 개발 프로세스에 대한 최종 사용자 관점의 질문에 개발자가 대답하게 함으로써 소프트웨어 품질 문제와 보안 취약점을 파악하는 것을 목표로 한다. 예를 들어, DHS(국토 안보부, Department of Homeland Security)의 사용자 정의 소프트웨어 설문지는 다음과 같은 질문이 포함되어 있다. "당신의 소프트웨어는 신뢰할 수 없는 리소스에서 입력되는 데이터를 검증합니까?" "당신의 소프트웨어는 어떤 위협을 가정하여 설계하였습니까?" DHS 설문지에 포함되어 있지 않지만, "당신의 앱은 네트워크 API(Application Programming Interface)에 접근합니까?" 같은 질문은 유용하다. 이러한 설문 조사는 소스 코드를 이용할 수 있고, 개발자가 질문에 대답할 수 있는 경우에만 효과적이다.

앱을 설계하거나 코딩 시 발생할 수 있는 결함은 NVD(National Vulnerability Database) [4]와 같이 공개적으로 접근할 수 있는 취약점 데이터베이스에서 확인할 수 있다. 앱에 대한 전체 심사 절차를 수행하기에 앞서, 분석가는 취약점 데이터베이스를 검사하여, 해당 버전의 앱에 알려진 결함이 있는지 확인해야 한다. 한 가지 이상의 심각한 결함이 이미 존재하는 경우, 이 결과만으로 조직에서 앱을 사용하지 않아야 하는 충분한 근거가 될 수 있으므로 나머지 절차는 생략할 수 있다. 그러나, 이러한 심각한 결함은 알려지지 않는 경우가 대부분이기 때문에 전체 점검 절차가 필요할 것이다. 앱을 설계하거나 코딩 시 발생할 수 있는 결함 이외에도 다양한 형태의 취약점이 존재한다. 이러한 취약점을 식별하기 위해 앱에 대한 요구사항을 정의하는 것이 선행되어야 하며, 요구사항과 차이가 존재한다면 취약점으로 표시할 수 있다.

정의된 조직 요구사항이 없을 수도 있다. 이 경우, 분석가는 테스트 도구로부터 획득한 보고서와 위험 평가만으로 앱의 보안 상태를 평가해야 한다.

조직 요구사항에 대한 충족 여부는 소프트웨어 취약점의 존재 여부에 기반하지 않는다. 따라서, 테스트 도구로서 결정할 수 없는 경우가 대부분이다. 조직 요구사항에 대한 충족 여부는 분석가의 손으로 결정해야 한다.

[4] 취약점 데이터베이스는 일반적으로 CVE(Common Vulnerabilities and Exposures)[12]로 취약점을 표시한다.

## 2.3 위험 허용

위험 허용은 조직이 수용할 수 있는 위험 수준이나 불확실성의 정도이다. 조직의 위험 허용 수준은 허용 수준까지 위험을 감수할 수 있는 데이터 및 시스템의 수량이다. 정의된 위험 허용 수준은 기밀성, 무결성 또는 가용성 저하로부터 조직이 어느 정도 보호되어야 하는지 나타낸다.

위험 허용은 다음의 사항을 고려해야 한다.

- 준수해야 할 보안규정, 권고안 및 모범사례
- 개인정보 위험
- 보안 위협
- 데이터 및 자산의 가치
- 산업 및 경쟁 업체의 압박
- 관리상의 기호 ( 嗜好 )

위험 허용은 일반적으로 상-중-하로 수준을 구분한다. 표 3은 각 수준에 대해 설명한다.

| 기 준 | 상 | 중 | 하 |
|---|---|---|---|
| 핵심 영역이거나 수직 시장[5]에 형성되어 있는가? (예: 금융, 정부, 의료) | 아니요 | 약간 | 예 |
| 준수해야 할 보안 관련 법 및 규제가 있는가? | 없음 | 약간 | 다수, 엄격 |
| 민감한 데이터를 보유하고 있는가? | 없음 | 약간 | 있음 |
| 고객이 강력한 보안 통제를 기대하고 있는가? | 없음 | 약간 | 있음 |
| 보안보다 혁신 및 수익에 우선순위를 두는가? | 예 | 약간 | 아님 |
| 조직이 여러 위치에 분산되어 있는가? | 아니요 | 약간 | 다수 |

표 3 - 위험 허용 수준

[5] 수직 시장(vertical market)은 비슷한 방법을 사용하여 비슷한 제품이나 서비스를 개발하는 특정 산업이나 기업군을 말한다.

### 2.3.1 보고서 분석

위험 분석 및 보고서와 관련한 한 가지 문제는 서로 다른 위험 분석나 보고서를 비교하고 정규화하고 해석하기 어렵다는 것이다. 이는 서로 다른 테스트 도구에서 사용되는 보안 관련된 용어 정의, 의미, 명명, 측정 방법 등이 매우 다양하기 때문이다. 예를 들어, 한 테스트 도구는 앱 사용에 대한 예상 위험을 상-중-하로 분류한다. 하지만, 다른 테스트 도구는 통과-경고-실패로 분류한다. 위험 평가 및 취약점 보고와 관련한 몇 가지 표준이 [6] 존재하지만, 이러한 표준이 테스트 도구에 적용될 확률은 낮다. 조직은 위험 평가 및 취약점 보고 표준을 적용한 테스트 도구를 사용하는 것이 바람직하다. 이것이 불가능하다면, 조직은 분석가가 테스트 도구에 의해 생성된 위험 평가 및 보고서를 해석할 수 있도록 분석가를 충분히 교육해야 한다.

### 2.3.2 컴플라이언스 vs 인증

컴플라이언스와 인증은 모바일 앱 심사에서 보안 요구사항을 성공적으로 구현했음을 입증하기 위해 주로 사용된다. 특별한 보안 요구사항(예 : 모바일 앱 심사에 대한 NIAP 보호 프로파일)을 만족하도록 개발된 모바일 앱의 경우, 개발자는 컴플라이언스나 인증을 선택할 수 있다. 컴플라이언스와 인증의 선택은 조직의 요구에 따라 달라진다.

모바일 앱 보안 컴플라이언스는 모바일 앱이 보안 요구사항을 충족하는지 자체적으로 또는 비공식적인 제 3자를 통해서 입증하는 것을 의미한다. 예를 들어, 기업은 자체적으로 마련한 모바일 앱 심사 절차를 사용하여, 모바일 앱의 보안 및 개인정보보호를 검증할 수 있다. 조직의 내부 절차에 따라 모바일 앱의 사용을 승인한다.

반면, 인증은 권한을 보유한 검증자에 의해 검증이 통과하였음을 의미한다. 예를 들어, NIAP 인증을 위해서는 공식적인 NIAP 검증 절차(https://www.niap-ccevs.org/Ref/Evals.cfm)를 따라야 한다. 이 경우 앱 개발사는 제품 평가를 위해 승인된 CC 테스트 연구소(Common Criterial Testing Lab)를 선택할 수 있다. 검증 절차가 완료되면, 공식적으로 인증이 승인되며, 승인된 제품 목록에 등록된다.

[6] 위험평가 표준은 CVSS(Common Vulnerability Scoring System)를 4.2.3에서 설명하며, 취약점 보고 표준은 2.1에서 설명했다.

# 3 앱 심사 프로세스

앱 심사 프로세스는 모바일 앱이 조직의 앱 보안 요구사항을 준수하는지 확인하기 위해 조직에서 수행하는 일련의 활동이다. [7] 앱이 조직의 앱 보안 요구사항을 준수하는 것으로 확인되면, 조직에서 사용하는 기기에 앱을 배포할 수 있다. 그림 2는 앱 점검 절차에 대한 개요를 보여준다.

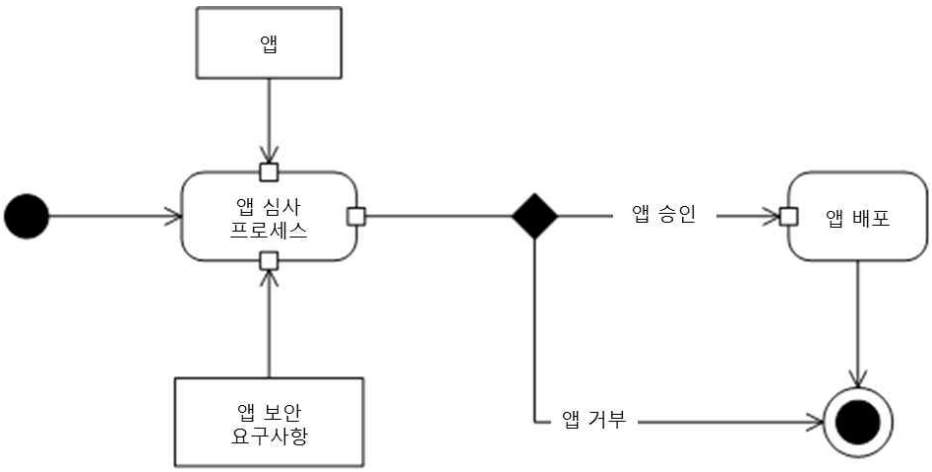


그림 2 - 앱 점검 절차 개요

앱 심사 프로세스는 조직마다 다를 수 있다. 하지만, 반복적으로 수행 가능해야 하고, 효율적이어야 하며, 일관성을 유지해야 한다. 또한, 오탐과 같은 오류는 최소화해야 한다. 일반적으로 앱 심사 프로세스는 수동으로 수행되거나, 앱 심사 시스템에 의해 자동으로 수행된다. 앱에 취약점이나 악성 행위가 존재하는지 확인하기 위해 앱 심사 시스템의 일부로 여러 가지 테스트 도구를 사용할 수 있다.

그림 1에서 보듯이 조직은 모바일 앱 수명주기의 앱 획득 단계에서 앱 심사 프로세스를 수행한다. 즉, 앱은 준비되었으나, 아직 조직에 배포되지 않은 상황이다. 이런 상황에서 앱 심사 프로세스를 수행하는 근본적인 이유는 개발자에 의한 소프트웨어 보증 절차로는 앱이 조직의 보안 요구사항을 준수하고 있음을 보장할 수 없기 때문이다. 또한, 개발자에 의한 앱 테스트는 앱 심사 프로세스에 포함되지 않는다. 따라서, 조직은 소스 코드 분석 등 앞서 수행한 보증 활동의 결과를 신뢰해야 한다. 조직은 공식 앱 스토어에서 앱을 이용할 수 있다는 이유만으로 앱이 완전히 심사되었거나, 보안 요구사항을 준수한다고 가정해서는 안된다.

[7] 앱 심사 프로세스는 안정성, 성능 및 접근성 등의 문제를 평가하기 위해 사용할 수도 있다. 하지만, 주로 보안 관련 문제를 평가하기 위해 사용한다.

모바일 기기에 앱을 배포하기 이전에 앱 심사 프로세스를 수행하면 엄격하고 포괄적인 분석을 수행할 수 있는 이점이 있다. 또한, 테스트가 배포 전에 수행되기 때문에, 발견된 위협을 수정하기 위한 시기 때문에 심사 프로세스의 진행이 방해받지 않는다. 이 문서는 조직이 앱 획득 단계에서 모바일 앱을 심사하는데 초점을 맞추고 있지만, NIST는 배포 단계(예 : 모바일 기기의 엔드포인트 솔루션)에서도 보안 분석을 수행할 것을 권고한다.

앱 심사 프로세스는 앱 접수, 앱 테스트, 앱 승인 및 거부, 결과 제출의 네 가지 하위 프로세스로 구성된다. 그림 3은 이 하위 프로세스를 보여준다.

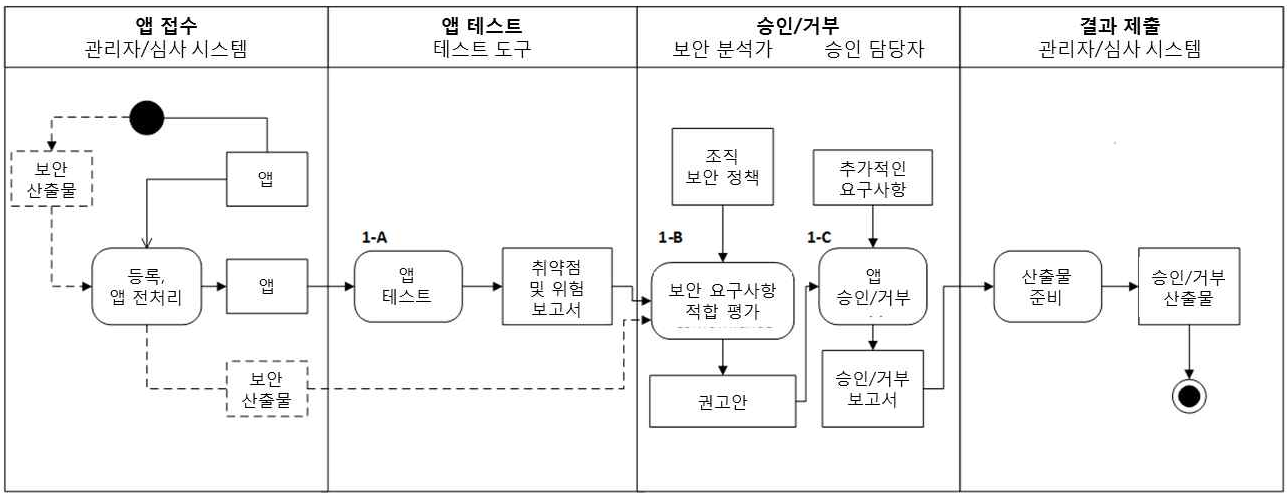


그림 3 – 앱 심사 프로세스의 하위 프로세스

## 3.1 앱 접수

앱 접수 절차는 분석을 위해 앱을 받으면 시작된다. 이 절차는 일반적으로 조직의 관리자에 의해 수동으로 수행되거나, 앱 심사 시스템에서 자동으로 수행된다. 앱 접수 절차에서 앱은 필수적으로 제출해야 하며, 추가적인 심사 산출물은 선택적으로 제출한다.

앱을 받은 후 개발자 정보, 제출 시간 및 데이터 및 앱 심사 프로세스에 필요한 기타 관련 정보 등을 기록하여 앱을 등록한다. 등록 후, 앱에 대해 전처리(preprocessing)를 수행할 수도 있다. 전처리에는 일반적으로 디코딩이나 디컴파일이 포함되며, 필요한 메타데이터(예 : 앱 이름, 버전)를 추출하고 앱이 적절하게 디코딩 또는 디컴파일되는지 확인한다. 테스트 도구로 분석을 수행하기에 앞서 이러한 작업이 필요할 수 있다.

앱 이외에도 개발자는 이전에 작성된 보안 분석 보고서를 비롯한 소프트웨어 보증 산출물을 선택적으로 제공할 수 있다. 이러한 산출물을 인정하는 조직은 개발자가 산출물에 명시한 앱 품질에 대한 서술을 신뢰해야 한다.

## 3.2 앱 테스트

앱 테스트 절차는 앱이 등록되고 전처리 후 한 가지 이상의 점검 도구로 수행된다. 테스트 도구는 소프트웨어 취약점 여부를 점검하는 소프트웨어 도구나 서비스이다. 이러한 테스트는 각각 다른 분석 방법(예 : 정적 분석)으로 진행될 수 있고, 수동이나 자동으로 수행될 수 있다. [8] 테스트 도구에 의해 수행되는 점검은 앱에서 공통적으로 발생하는 소프트웨어 취약점을 식별하고, NIAP에서 명시하는 것과 같은 일반적인 앱 보안 요구사항이 충족되는지 확인한다.

테스트 도구는 앱을 테스트한 후 발견된 소프트웨어 취약점 또는 잠재적으로 유해한 행위를 식별한 보고서를 생성한다. 일반적으로 보고서에는 탐지된 취약점이 악용될 가능성과 발견된 취약점이 앱이나 기기 또는 네트워크에 미칠 수 있는 영향을 평가하는 점수가 포함된다. 테스트 도구는 NIAP 등 표준이 적용된 보고서를 생성할 수 있다. 일부 테스트 도구의 경우, 일반적인 앱 보안 요구사항의 위반을 탐지할 수 있지만, 조직의 정책이나 규정 등을 위반하는 사항은 탐지할 수 없음에 유의해야 한다.

그림 4는 일반적인 테스트 도구의 작업 흐름을 보여준다. 테스트 도구를 실행하면, 앱은 일반적으로 테스트 도구 공급 업체의 서버에 저장된다. 앱의 코드를 분석하는 정적 도구의 경우, 바이너리 실행 파일을 디코딩, 디컴파일, 복호화하여 분석이 가능한 중간 형태로 만든다. [9] 앱의 런타임 동작을 분석하는 동적 도구의 경우, 앱의 동작을 분석할 수 있는 장치 또는 에뮬레이터에 앱을 설치하고 실행한다. 이 도구는 앱을 분석한 후 취약점 보고서 및 위험 평가를 생성하고 이 보고서를 앱 심사 시스템에 제출한다.

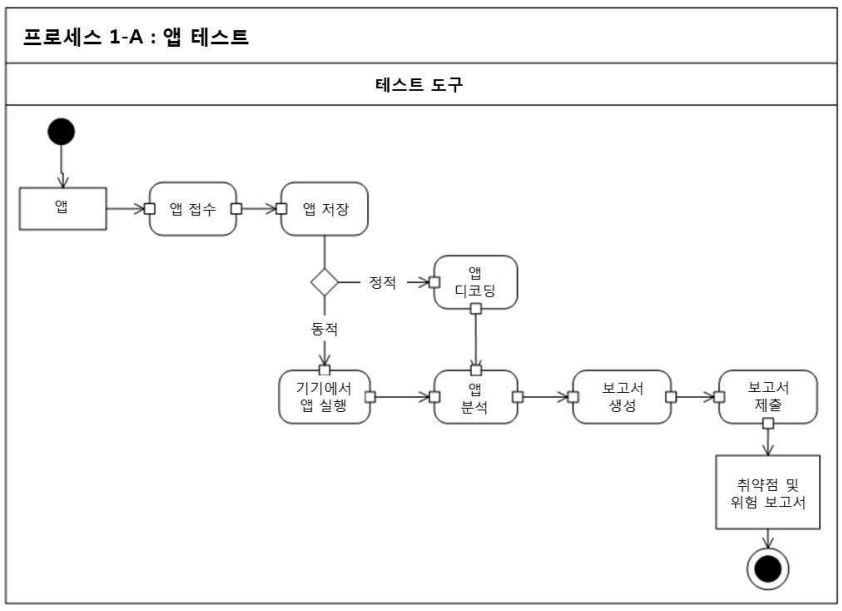


그림 4 - 테스트 도구의 작업 흐름

[8] 앱 심사 도구에서 사용하는 기술 및 방법에 대해 4장에서 설명한다.
[9] 일반적으로 디컴파일로 얻어지는 코드는 소스 코드가 아닌, 분석할 수 있는 중간 코드이다.

## 3.3 앱 승인 및 거부

앱 승인 및 거부 절차는 취약점 및 위험 보고서가 점검 도구에 의해 생성되고, 한 명 이상의 보안 분석가가 검토할 수 있을 때 진행된다. 보안 분석가는 한 가지 이상의 점검 도구에서 생성된 취약점 보고서 및 위험 평가를 검토하여, 일반 보안 요구사항을 모두 충족하는지 확인한다. 분석가는 앱이 보안 정책이나 규정을 위반하는지 결정하기 위해 조직 보안 요구사항에 대해 평가한다. 분석가는 일반 및 조직 보안 요구사항을 모두 평가한 후, 조직의 모바일 기기에 앱의 배포를 승인하거나 거부할 것을 권고하는 보고서를 작성한다.

분석가의 권고 보고서는 앱에 대한 배포 승인 권한을 보유한 임직원(이하, 승인 담당자)이 사용한다. 승인 담당자는 분석가가 제공한 권고안을 이용하여 앱에 대한 승인 및 거부를 결정한다. 또한, 승인 담당자는 보안과 관련이 없는 다른 기준(예 : 비용, 필요성 등)도 고려한다. 이러한 문서에는 앱의 보안 상태를 비롯하여, 보안과 관련이 없는 요구사항도 명시된다. 조직의 공식적인 승인 및 거부는 최종 승인/거부 보고서에 명시된다. 그림 5는 앱 승인/거부 프로세스를 보여준다.

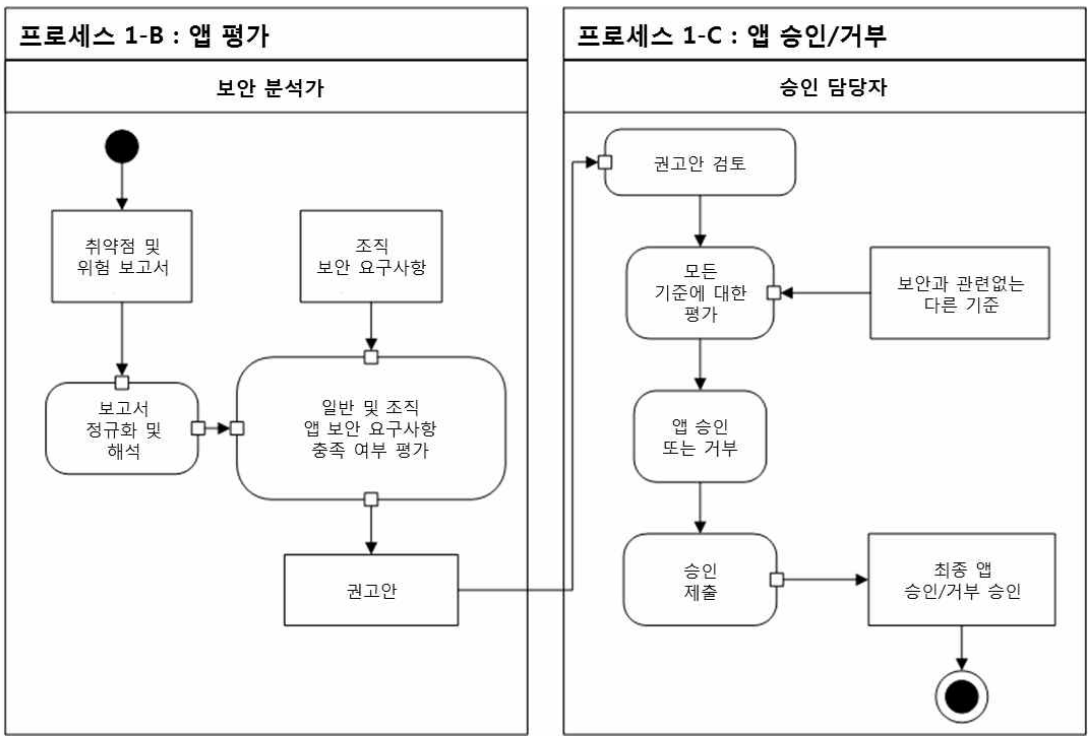


그림 5 - 앱 승인/거부 절차

## 3.4 결과 제출

결과 제출 프로세스는 승인 담당자가 앱의 최종 승인/거부 보고서를 마무리하고, 각종 산출물의 제출이 준비된 이후 시작된다. 산출물에는 최종 승인/거부 보고서 및 점검 도구 보고서가 포함된다. 또한, 앱 심사 프로세스를 완료한 앱이라는 것을 표시하기 위해 디지털 서명한 앱을 산출물에 포함할 수도 있다. 디지털 서명을 사용하면 원본을 인증하고 무결성이 보호되기 때문에, 접수 단계에서 제출된 앱이 변경되지 않았음을 증명할 수 있다.

# 4 앱 테스트 및 취약점 분류

앱 테스트 프로세스에서 취약점 및 악의적인 행위가 존재하는지 확인하기 테스트하기 위해 테스트 도구가 사용된다. 테스트 도구는 NIAP과 같은 표준을 기반으로 하여 일반 앱 보안 요구사항의 충족 여부를 판단하는 데 사용될 수 있다. 이 장에서는 테스트 도구 및 서비스를 사용하는 전략과 접근법에 대해 설명한다. 또한 취약점의 분류 및 지표에 대해 설명한다.

## 4.1 테스트 접근법

테스트 도구는 정확성 점검, 소스 코드 또는 바이너리 코드 분석, 정적 또는 동적 분석, 수동 또는 자동 앱 테스트과 같은 여러 분석 기법을 사용한다.

### 4.1.1 정확성 테스트

소트트웨어 정확성 테스트는 앱을 테스트하는 접근법 중 하나이다. [14] 소프트웨어 정확성 테스트는 오류를 탐지하기 위해 프로그램을 실행하는 절차이다. 소프트웨어 정확성 테스트의 원래 목적은 품질 보증, 기능의 검증, 신뢰도 측정에 있다. 그러나, 소프트웨어 정확성 테스트를 통해 소프트웨어의 품질, 기능, 신뢰도에 부정적인 영향을 미칠 수 있는 잠재적인 보안 취약점을 식별할 수도 있다. 예를 들어, 예기치 않은 동작을 일으키는 소프트웨어는 보안 결함이 자주 발견된다. 소프트웨어 정확성 테스트의 가장 큰 장점은 테스트할 소프트웨어의 설계 명세서를 기반으로 한다는 것이다. 이러한 설계 명세서는 테스트 수행 시, 소프트웨어의 동작을 지정하는 요구사항이 된다. 이는 분석가가 요구사항을 직접 도출해야 하는 보안 평가 접근법과 구별된다. 보안 평가 접근법에서 요구사항은 다양한 소프트웨어에 공통적으로 적용할 수 있는 보안 요구사항을 기반으로 한다. 따라서, 점검 중인 소프트웨어에 대한 고유한 취약점을 점검하지 못할 수 있다. 그러나, 보안과 품질을 상호 밀접하게 연관되어 있기 때문에, 가급적 소프트웨어 정확성 테스트을 수행할 것을 권고한다.

### 4.1.2 소스 코드 및 바이너리 코드 테스트

앱 테스트에서 소스 코드를 활용할 수 있는지 여부는 가장 중요한 요소이다. 일반적으로 앱 스토어에서 다운로드한 앱은 소스 코드를 확인할 수 없다. 오픈 소스 앱과 같이 소스 코드를 활용할 수 있는 경우, 소스 코드를 분석하기 위해 다양한 도구를 사용할 수 있다. 소스 코드 리뷰의 목적은 소스 코드에서 취약점을 식별하고, 테스트 도구로 테스트한 결과를 검증하는 것이다. 자동화된 도구를 사용하더라도 분석은 상당한 업무 강도를 요구한다. 자동화된 정적 분석 도구를 사용하는 경우 코드 리뷰의 일관성을 유지할 수 있고, 많은 양의 코드에 대한 리뷰도 가능하다. 따라서, 분석가는 자동 또는 수동으로 분석하는 것에 관계없이 자동화된 정적 분석 도구를 일반적으로 사용해야 하며, CWE(Common Weakness Enumeration) ID 또는 기타 통용되는 명명법으로 발견한 취약점을 표현해야 한다. 코드 리뷰를 수행하려면 소프트웨어 개발과 앱 보안 영역에 대한 지식이 필요하다. 조직은 분석가가 소스 코드 리뷰에 필요한 기술과 전문 지식을 보유하고 있는지 확인해야 한다. 자체적으로 앱을 개발하려는 조직은 소프트웨어 개발 생명주기의 전 과정에서 시큐어코딩 및 소프트웨어 품질 보증 절차에 대해 설명하고 있는 지침을 참고해야 한다. [15][16]

앱의 소스 코드를 활용할 수 없는 경우, 바이너리 코드를 분석하는 것으로 대신할 수 있다. 앱에서 바이너리 코드라는 용어는 바이트 코드와 기계 코드 모두를 의미한다. 예를 들어, 안드로이드 앱은 JVM(Java Virtual Machine)에서 실행되는 바이트 코드로 컴파일된다. 하지만, 자체 제작한 라이브러리가 함께 제공되는 경우도 존재한다. 이러한 라이브러리는 모바일 기기 CPU에서 직접 실행되는 기계 코드 형식으로 제공된다. 안드로이드 바이너리 앱에 포함된 바이트 코드는 안드로이드 기기가 없어도 에뮬레이터 또는 가상 환경을 이용하여 분석할 수 있다.

### 4.1.3 정적 및 동적 테스트

분석 도구는 주로 정적 또는 동적 분석 도구로 구분된다. [10] 정적 분석은 소스 코드와 바이너리 코드를 검사하고, 실행 중 발생할 수 있는 모든 동작을 추측한다. 분석 결과는 입력이나 실행 환경에 관계없이 프로그램의 동작을 정확하게 설명할 정도의 수준이다. 동적 분석은 프로그램을 입력값을 주면서 실행하고, 실행 과정에서 행위를 분석한다. 테스트해야 하는 입력값의 수가 많은 경우, 처리 시간이 길어진다. 그러나, 조합 점검(combinatorial testing)과 같은 방법을 사용할 경우, 입력값의 수를 줄여 분석 결과의 도출까지 소요되는 시간을 단축할 수 있다. [18] 그러나, 동적 분석은 코드 전체를 100% 분석하지 않는다. [19] 조직은 정적 분석 도구와 동적 분석 도구의 기술적 절충점을 고려하여, 조직에서 설정한 소프트웨어 보증 목표에 따라 사용량을 조절해야 한다.

소스 코드를 활용할 수 없는 경우, 정적 분석을 위해서는 바이너리 코드에 대한 리버스 엔지니어링(이하 리버싱)이 선행되어야 한다. 바이트 코드 [11] 를 분석하는 것은 기계 코드를 분석하는 것보다 상대적으로 쉽다. 많은 오픈 소스 및 학술 도구 [20]-[23] 뿐만 아니라, 많은 상용 정적 분석 도구에서도 바이트 코드에 대한 분석을 지원한다. 기계 코드의 경우, 제어 흐름이나 변수의 데이터를 추적하기 어렵다. 메모리에 저장되는 변수는 이름으로 식별되지 않으며, 다양한 방법으로 접근할 수 있기 때문이다. 기계 코드를 리버싱하는 가장 일반적인 방법은 디스어셈블러나 디컴파일러를 사용하여 원본 소스 코드로 복원해 보는 것이다. 사람이 코드를 테스트하기 위한 목적으로 리버싱하는 경우, 디스어셈블러나 디컴파일러는 유용하다. 디스어셈블러나 디컴파일러의 결과물이 적절한 기술을 보유한 사람이라면 이해가 가능한 형식이기 때문이다. 가장 우수한 디스어셈블러 조차도 오류가 존재한다. [21] 이러한 오류의 일부는 정적 분석으로 보완할 수 있다. 정적 분석을 위해 리버싱하는 경우, 사람이 읽을 수 있는 형태로 변환하는 중간 단계를 생략하고, 정적 분석 도구가 해석할 수 있는 형태로 바로 변환하는 것이 바람직하다. 기계 코드 분석을 목적으로 하는 정적 분석 도구는 이런 과정을 자동화할 수 있다.

[10] 모바일 기기에는 행위 테스트를 수행하고, 스스로 분류하는 분석 도구가 있다. 행위 테스트(행위 분석이라고도 함)는 연락처 목록에 액세스하는 플래시 라이트 앱과 같이 악의적인 행동이나 위험한 행동을 탐지하는 정적/동적 테스트의 한 형태이다. [17] 이 문서에서는 정적 또는 동적 테스트의 하위에 행위 테스트가 포함되어 있는 것으로 가정한다.

[11] ASM 프레임워크는 바이트 코드 분석을 위해 일반적으로 사용하는 프레임워크이다. [20]

정적 분석과 대조적으로, 동적 분석은 실행되는 코드의 동작을 보는 것이 가장 중요하다. 코드의 동작을 보기 위해서는 기본적으로 두 가지 방법이 존재한다. 첫째, 실행중인 코드에 원격 디버거를 연결하는 것이다. 둘째, 디버깅 기능이 내장된 에뮬레이터에서 앱을 실행하는 것이다. 배포될 모바일 기기에서 코드를 실행할 경우, 테스트 도구에서 기기의 정확한 속성을 선택하여 앱의 동작을 더 정확하게 보여줄 수 있다. 반면, 에뮬레이터는 제어가 용이하다. 특히, 에뮬레이터가 오픈 소스인 경우, 분석가가 필요한 정보를 수집할 수 있도록 수정할 수 있다. 에뮬레이터는 다른 장치를 시뮬레이션할 수 있지만, 모든 장치를 시뮬레이션하는 것은 아니다. 따라서, 시뮬레이션이 정확하지 않을 수 있다. 악성코드가 에뮬레이터의 사용을 탐지하고, 탐지를 회피하기 위해 동작을 변경하는 사례가 증가하고 있다. 따라서, 악성코드를 탐지하지 못하는 상황을 방지하기 위해 모바일 기기와 에뮬레이터를 함께 사용하여 테스트할 것을 권고한다.

개별 기능의 목적은 알지 못하지만, 앱의 행위를 관찰함으로써 유용한 정보를 수집할 수도 있다. 예를 들어, 테스트 도구는 앱이 외부 자원과 상호 작용하는 방식을 관찰하고, 운영체제에서 요청한 서비스 및 앱이 실행되는 권한을 기록한다. 앱에서 사용하는 기기의 기능 다수가 테스트 도구에 의해 추론(예 : 카메라 앱은 카메라에 접근)될 것이다. 그러나, 앱은 설명된 기능의 범위를 초과하여 추가적인 기기의 기능에 접근(예 : 네트워크에 접근하려는 카메라 앱)할 수 있다. 특히, 특정 입력에 대한 앱의 동작을 관찰함으로써 분석가는 점검 중인 기능이 특정 입력에 대해 적절하게 동작하는지 테스트할 수 있다. 예를 들어, 캘린더 앱이 여러 기기 간 동기화를 위해 캘린더 데이터를 네트워크에 전송할 수 있는 권한을 가지는 것은 적절하다. 그러나, 앱이 사용자 요청을 처리하기 위해 필요한 것 이상의 데이터를 전송하는 경우, 점검 도구는 전송되는 데이터와 용도를 조사할 수 있다.

## 4.2 취약점 분류 및 측정 기준

모바일 앱의 취약점을 설명할 때, 공통적인 용어를 사용하는 것이 유리하다. 이 장에서는 취약점의 식별, 설명, 위험도 측정에 일반적으로 사용되는 분류 기준과 측정 기준에 대해 설명한다.

### 4.2.1 CWE (Common Weakness Enumeration)

CVE는 MITRE에서 관리하는 소프트웨어 취약점 분류 체계이다. [24] CWE는 소프트웨어 취약점을 분류하는 공통적인 용어로 사용된다. 동일한 소프트웨어 오류에 대해 프로그래밍 언어별로 각각의 버전을 생성할 수 있다. CWE는 프로그래밍 언어는 다르지만, 동일한 오류에 대해 공통적으로 참조할 수 있는 전문 용어를 정의한다. 그리고, 각각의 오류에 대해 오류를 완화할 수 있는 전략을 제공한다.

### 4.2.2 CVE (Common Vulnerability Enumeration)

CVE는 MITRE에서 제공하는 소프트웨어 취약점에 대한 명명 체계이다. [44] 취약점이 식별되면, CNA(CVE Numbering Authority)에 보고되며, 고유의 식별자가 부여된다. 각각의 CVE는 점수 산정과 설명을 위해 NVD(National Vulnerability Database)로 보고된다. NVD는 미국 정부의 표준 기반 취약점 관리 데이터 저장소이다. NVD는 특정 컴퓨터 시스템에 대한 취약점을 설명하는 데이터를 수집하고 분석하며 저장한다. 또한, NVD는 보안 점검 목록, 보안 관련 소프트웨어 결함, 부적절한 설정, 제품 이름, 영향 지표에 대한 데이터베이스를 제공한다. NVD는 이러한 목적을 위해 CWE와 CVE를 광범위하게 사용한다.

### 4.2.3 CVSS (Common Vulnerability Scoring System)

CVSS는 FIRST(the Forum of Incident Response and Security Teams)에서 소유하고 있는 취약점 점수 산정 체계이다. [25] CVSS 모델은 반복적이고 정확한 측정을 보장하면서, 사용자가 점수 산정에 사용된 취약점의 근본적인 특성을 확인할 수 있게 해준다. CVSS는 정확하고, 일관성 있는 취약점 영향 점수 산정이 요구되는 산업, 조직 및 정부기관에서 사용할 수 있다. 취약점 점수 산정에 사용되는 알고리즘은 모두 공개되어 있으며, 분석가가 제공한 세 가지 지표(기본, 일시, 환경)에 의해 주로 산정된다. CVSS의 일반적인 용도는 취약점의 영향도와 보완 우선순위를 계산하는 것이다. NVD는 CVSS를 이용하여, 취약점에 대한 점수를 제공한다.

# 5 앱 심사 시 고려사항

이 장에서는 조직에서 앱 심사 프로세스를 수립할 때 고려해야 하는 추가적인 기준에 대해 설명한다.

## 5.1 관리되는 앱과 관리되지 않는 앱

조직의 업무용 앱이나 조직의 모바일 기기(조직의 업무를 위해 사용되는 임직원의 개인 모바일 기기를 포함)에 설치된 타사의 앱을 배포 주기 전체(애플리케이션의 초기 배포 및 구성부터 기기에서 제거하기까지의 과정)에 대해 관리할 수 있다. 조직의 업무 서비스 및 데이터에 접근하는 앱과 같이 관리되는 앱은 MAM(Mobile Application Management) 시스템을 사용하여 관리할 수 있다. 관리되지 않는 앱은 MAM 또는 그와 유사한 시스템에서 관리하지 않을 것이다.

기기를 제외하고 앱만 관리하는 경우, MAM 시스템에 기기를 등록할 필요가 없으며, 기기에 조직의 프로파일을 설치하지 않아도 되는 장점이 있다. MAM 솔루션은 조직의 업무용 앱 목록을 애플 앱스토어나 구글 플레이스토어와 통합함으로써 사용자가 쉽게 조직의 업무용 앱을 설치할 수 있게 한다. MAM 관리자는 앱을 배포하거나, OTA(Over-The-Air) 업데이트를 제공할 수 있다. 또한, 기기에 영향을 주지 않으면서 앱의 기능을 제한할 수 있어, BYOD(Bring Your Own Device) 사용자의 거부감을 줄여줄 것이다. 일부 MDM(Mobile Device Management) 시스템은 MAM의 기능을 포함하고 있다. 이에 따라 한 기기에 설치된 각각의 앱에 대해 세밀한 제어가 가능하다.

조직은 모바일 앱 관리를 위한 솔루션, 요구사항 [12] 및 정책을 설계할 때, 관리되는 앱과 관리되지 않는 앱 사이의 적절한 균형을 고려해야 한다. 관리되는 앱만 조직의 네트워크 및 서비스에 접근할 수 있게 하는 경우 보안은 보장될 것이다. 하지만, 간접적인 관리비용과 추가적인 비용이 필요할 것이다.

## 5.2 앱 심사의 한계

다른 소프트웨어 보증 절차와 마찬가지로, 철저한 심사 절차를 거치더라도 잠재적인 취약점이나 악의적인 행위를 전부 찾을 수는 없다. 앱 보안 평가는 일반적으로 조직의 보안 태세 개선에 도움을 주겠지만, 개선의 정도를 쉽고 빠르게 확인할 수 없다는 것을 인식해야 한다. 또한, 보안의 관점에서 앱 점검 절차를 통해 제공받을 수 있는 것과 제공받을 수 없는 것에 대해 인지하고 있어야 한다.

[12] 이러한 보안 요구사항 사례로 미 국방부의 "모바일 앱 보안 요구사항"이 있다. [22]

조직은 보안 평가 절차를 수행하는 임직원의 역할에 대한 중요성을 교육해야 하고, 앱 심사가 자동화된 점검에만 의존하지지 않도록 해야 한다. 보안 분석은 본질적으로 인간 중심의 프로세스이다. [15][27] 자동화 도구는 소프트웨어 보안의 바탕이 되는 상호 관계나 상호 의존성을 분석할 수 없다. 소프트웨어의 동작을 완전히 분석하는 것은 불가능하기 때문이다. 이는 컴퓨터 공학에서 오랫동안 해결하지 못한 문제[28] 중의 하나이며, 현재의 기술로는 이론적인 가능성도 제시하지 못하고 있다. 자동화된 방식으로 복잡하고 다면적인 소프트웨어 아키텍처를 완벽하게 분석할 수는 없다.

또 다른 문제점으로, 현재 사용하는 소프트웨어 분석 도구는 소프트웨어가 특정 상황에서 안전한 방식으로 동작하기 위해 어떻게 해야하는지 본질적으로 이해하지 않는다. 예를 들어, 가상 사설망(VPN)을 사용하는 경우, 클라우드로 전송하기 위한 데이터를 암호화하지 않더라도 보안 문제가 되지 않을 수 있다. 앱에 대한 보안 요구사항을 정확하게 예상하고, 완전하게 이해하더라도, 기계가 이해할 수 있는 형태로 모호하지 않게 번역할 수 있는 기술은 현재 존재하지 않는다.

이러한 이유로 분석가(사람)는 보안 분석에서 매우 중요하며, 더 나아가 결과의 품질은 사람의 노력과 전문 지식의 수준에 의해 결정된다. 분석가는 소프트웨어 보안 평가를 위한 표준 절차 및 모범 사례에 대해 잘 알고 있어야 한다. [15][29]-[31] 강력한 앱 심사 절차는 다양한 평가 도구 및 프로세스, 사람과 상호 작용이 함께 작용하는 복합 검사 접근법(toolbox approach)을 사용해야 성공할 수 있다. 각 도구는 고유한 한계를 가지고 있다. 따라서, 하나의 도구에만 의존하는 것은 사람과 상호 작용이 있는 경우라도 상당히 위험하다.

## 5.3 로컬 및 원격 도구와 서비스

모바일 앱 분석을 위한 전용 도구와 서비스가 다수 존재한다. [32][33] 도구 및 서비스 제공 업체에서 사용하는 모델에 따라 앱 분석은 서로 다른 물리적 위치에서 수행될 수 있다. 예를 들어, 앱을 사용하는 조직의 네트워크 내부에 분석 도구가 설치되어 실행될 수 있다. 다른 공급 업체는 외부에서 점검 서비스를 진행할 수 있다. 원격 도구(offsite tool)는 도구 및 서비스 제공 업체에 직접 설치되어 있거나, 클라우드 인프라에 구성되어 있을 수 있다. 따라서, 조직에서는 검증 도구 및 서비스를 사용하기 전에 이러한 제공 형태에 대해 이해하고 있어야 한다. 특히, 앱에 민감한 정보나 기밀 정보가 포함될 가능성이 있는 경우에는 이를 유의해야 한다.

## 5.4 자동화된 승인 및 거부

앱의 승인 및 거부와 관련하여 조직 공유의 정책, 규제 등이 존재하지 않는 경우, 분석가가 앱을 승인하거나 거부하도록 권고안을 도출하는 활동을 자동화할 수 있다. 앱 심사 시스템을 사용하여 다양한 도구의 위험 평가를 기반으로 앱을 자동으로 승인하거나 거부하도록 규칙을 지정할 수 있다. 예를 들어, 모든 점검 도구가 낮은 위험이 있는 것으로 간주하는 경우, 앱을 자동으로 승인하도록 설정할 수 있다. 이와 유사하게, 앱 심사 시스템은 조직의 고유한 요구사항을 자동으로 수행하도록 구성할 수 있다. 예를 들어, 앱의 전처리 과정에서 추출된 메타 데이터를 이용하여 특정 회사의 앱을 자동으로 거부하도록 앱 심사 시스템을 설정할 수 있다.

## 5.5 상호 관계

앱 심사 중 발견한 내용을 공유하면, 앱의 검증 작업의 중복과 비용을 크게 줄일 수 있다. 소프트웨어 보증 커뮤니티에서의 정보 공유는 필수적이며, 점검 도구가 전 세계 보안 전문가의 집단적인 노력으로 이익을 얻는 데 도움이 될 수 있다. NVD(National Vulnerability Database) [34] 는 SCAP(Security Content Automation Protocol) [35] 을 사용하여 표현되는 미국 정부의 표준 기반 취약점 관리 데이터 저장소이다. 이 데이터를 이용하여 취약점 관리, 보안 측정 및 컴플라이언스를 자동화할 수 있다. NVD는 보안 점검표, 보안 관련 소프트웨어의 결함, 잘못된 구성, 제품명과 영향 평가 지표가 포함되어 있다. SCAP는 보안 소프트웨어 제품이 소프트웨어 결함과 보안 구성 정보를 전달하는 형식 및 명명법을 표준화한 일련의 명세서로 자동화된 취약점 검사, 기술 통제 컴플라이언스 활동(technical control compliance activity) 및 보안 측정을 지원하는 다목적 프로토콜이다. SCAP를 개발한 목적에는 시스템 보안 관리 표준화, 보안 제품의 상호 운용성 촉진 및 표준화된 보안 표현 사용 촉진 등이 포함된다. CWE [24] 및 CAPEC(Common Attack Pattern Enumeration and Classification) [36] 은 바이너리 또는 운영 시스템에 대한 침투 테스트를 수행할 때, 유용하게 사용할 수 있는 약점 및 공격 방법을 제공한다. 소프트웨어 보증 커뮤니티는 소프트웨어 취약점을 분류하고 설명하는 작업을 지속하고 있다. 이와 더불어 앱에 존재할 수 있는 다양한 약점에 우선 순위를 지정하는 작업도 지속하고 있다. 지정된 우선 순위를 통해 조직은 앱의 용도 및 목적을 고려하였을 때, 앱을 가장 위험하게 하는 약점 [40] 이 효과와 적용 범위에 서로 다른 도구 및 기술을 적용한 심사 활동을 통해 다루어져야 한다는 것을 알 수 있다.

## 5.6 예산 및 인원 배치

앱 보증 활동과 관련한 비용은 프로젝트 예산에 포함되어야 하며, 뒤늦게 고려되어서는 안된다. 앱 보증 활동과 관련한 비용은 상당히 높을 수 있으며, 점검 도구의 라이선스 비용과 분석가, 승인자, 관리자의 급여를 포함할 수 있다. 외부 위탁을 통해 앱을 개발하는 조직은 앱 평가 비용을 앱 개발 절차의 일부로 포함하도록 계약서에 명시해야 한다. 조직 내부에서 앱을 개발하는 경우 개발 작업의 최종 단계에서만 앱 심사 계획하면, 비용 증가와 프로젝트 일정 지연을 초래할 수 있음에 유의해야 한다. 앱을 구현한 개발자가 직접 취약점을 보완할 수 있도록 개발 절차 중간에 잠재적인 취약점을 식별하는 것을 강력하게 권고한다. 개발 과정에서 오류를 확인하고 수정하는 것은 제품이 출시된 후 오류를 수정하는 것보다 저렴하다. [37]

최적화된 앱 심사 절차를 구현하려면 적절한 전문성을 보유한 인력을 고용하는 것이 중요하다. 예를 들어, 조직에서는 소프트웨어 보안 및 정보 보증 분야에 경력을 보유한 분석가와 모바일 보안에 경력을 보유한 관리자를 고용해야 한다.

# 6 앱 심사 시스템

앱 심사 절차는 수동으로 수행할 수 있으나, 일반적으로 앱 심사 시스템(예 : NIST AppVet 시스템 [13] )을 사용하여 반자동이나 완전 자동화된 방식으로 앱 심사 절차를 수행하는 것이 유리하다. 앱 심사 시스템은 앱 심사 절차를 관리하고 자동화하는 시스템으로 웹 기반 서비스로 구현될 수 있다. 앱 심사 시스템은 일반적으로 점검 도구 및 서비스, 앱 스토어, EMM (Enterprise Mobility Management) 및 사용자로 구성된 앱 심사 생태계의 일부이다.

앱 심사 시스템은 보안 분석가 또는 전사 시스템 관리자가 앱의 사용자의 모바일 기기에 배포하기 전에 앱의 보안 문제를 식별하는데 사용된다. 앱 심사 시스템이 앱을 분석한 후, 보안 분석가는 심사 결과와 전사의 보안 환경을 고려하여 보안 권고안을 작성한다. 승인 담당자는 사용자의 역할, 앱의 필요 목적, 보안 분석가의 보안 권고안을 고려하여 앱의 사용을 승인한다. 그림 6은 앱 심사 시스템의 아키텍처 예시이다.

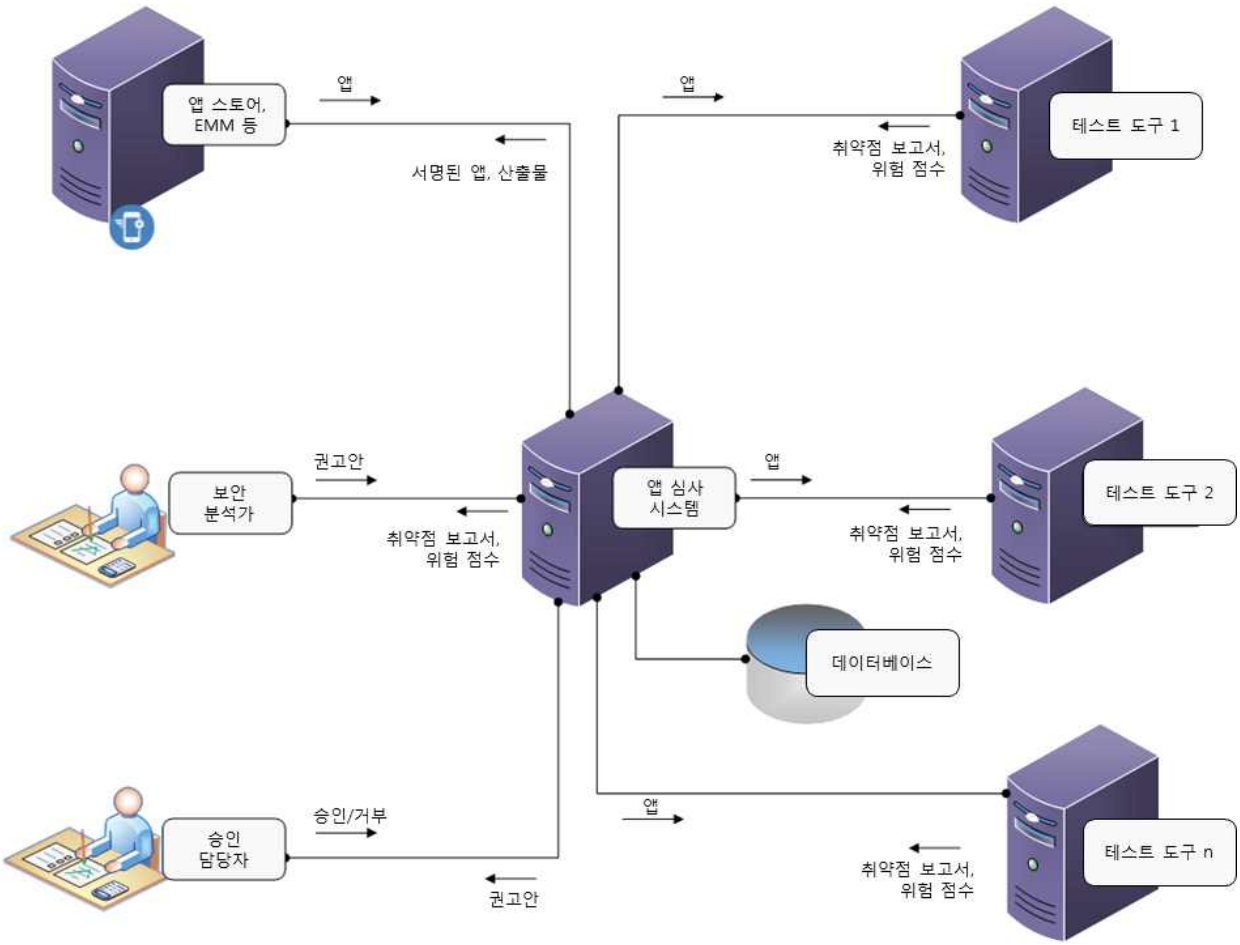


그림 6 – 앱 심사 시스템 아키텍처 예시

가운데 앱 심사 시스템이 있다. 앱 심사 시스템은 앱 심사 생태계의 중심 허브 역할을 수행한다. 앱 심사 시스템은 다른 시스템, 보안 분석가, 승인 담당자 사이의 요청과 응답을 조정한다. 데이터베이스는 앱 심사 시스템의 중요한 구성요소이며, 의사 결정 저장소로서 기능을 수행한다. 데이터베이스에는 차후에 참고할 수 있도록 테스트 보고서와 보안 분석가와 승인 담당자가 입력한 내용을 저장해야 한다.

기업의 모바일 기기에서 앱을 사용하기 위한 방법은 다양하다. 심사를 통과한 앱만 등록된 앱 스토어를 구축할 수 있다. 또는 EMM 시스템으로 모바일 기기에 어떤 앱이 어떤 소스에서 설치될 수 있는지를 규제하는 정책을 적용할 수 있다. EMM 시스템은 요청된 앱에 대한 정보를 앱 심사 시스템으로 전송하고 앱 심사 프로세스가 시작된다.

앱 심사 시스템은 다양한 테스트 도구에 의해 작성된 결과 보고서를 종합하여 요약하고, 대시보드를 통해 보안 분석가에게 제공한다. 다양한 테스트 결과를 검토한 후 보안 분석가는 권고안을 제출하며, 제출된 권고안은 앱 심사 시스템에 저장된다. 승인 담당자는 보안 분석가의 권고안과 앱이 필요한 목적을 함께 고려하여 앱의 사용을 승인하거나 거부할 수 있다. 앱 설치가 승인되면 앱 심사 시스템은 전자 서명된 앱을 포함한 산출물(artifact)을 앱 스토어 또는 EMM 시스템에 다시 제공하여 앱을 설치할 수 있게 한다.

그림 6은 로컬에서 제공되는 앱 심사 시스템을 나타내고 있지만, 다수 앱 심사 시스템이 클라우드 환경에서 제공될 수 있다. 클라우드 환경의 경우 그림 6에 표시된 구성 요소는 개인 또는 공용 클라우드 서비스 공급자가 제공하며, 다수의 기능은 가상화된다. 보안 분석가와 승인 담당자는 앱 심사 시스템이 어떻게 구현되는지 알아야 할 필요가 없다. 구축 형태에 관계없이 보안 분석가와 승인 담당자는 적절한 서비스와 뷰를 제공하는 대시보드를 통해 시스템과 상호작용할 것이다. 또한, 새로운 검사 도구가 사용 가능한 경우, 앱 심사 시스템을 모듈식으로 확장할 수 있다.

앱 심사 시스템은 분산되어 운영중인 타사의 테스트 도구를 통합하기 위해 API, 네트워크 프로토콜 및 스키마를 사용한다. 앱 심사 시스템에는 관리자, 분석가 및 승인 담당자와 같은 사용자가 보고서 및 위험 평가를 확인할 수 있는 UI 대시보드가 존재할 수 있다. 그림 6은 앱 심사 시스템이 API와 UI를 통해 앱 심사 생태계의 모든 구성 요소와 사용자간의 통합을 지원하는 방법에 관한 예시를 보여준다.

# 부록 A – 모바일 앱에 대한 위협

모든 소프트웨어와 마찬가지로 모바일 앱에서도 사용자, 모바일 기기, 데이터, 서비스를 공격에 노출시키는 취약점이 존재하는 경우가 자주 발생한다. 이러한 취약점은 설계나 구현의 오류이거나 악의적인 의도에 의해 발생한다. 취약점을 유발할 수 있는 모바일 소프트웨어 오류는 다수의 공통적인 분류가 존재한다. 여기에는 암호화 사용 및 구현 오류, 보안 서비스 사용 및 구현 오류, 모바일 기기의 소프트웨어 구성 요소 간 위험한 상호작용, 모바일 기기와 다른 시스템 간 위험한 상호작용 등이 포함된다. 암호화나 보안 서비스 사용 시 발생하는 일반적인 오류에는 사용자 또는 시스템의 미흡한 인증, 암호화 기본요소의 부적절한 구현, 취약한 암호화 알고리즘 및 파라미터 사용, 모바일 기기와 서비스 간 트래픽 비암호화 등이 있다. 모바일 기기의 소프트웨어 구성 요소 간 위험한 상용작용에는 보안이 중요한 작업에 출처를 신뢰할 수 없는 데이터의 사용, 취약한 서드파티 라이브러리 사용, 앱 외부로 민감한 데이터를 유출하는 코드 등이 포함된다.

모바일 앱을 심사한 후 사용자의 모바일 기기에 배포하는 경우, 전사 시스템 관리자는 취약점을 유발하거나 전사 보안 및 개인정보보호 정책을 위반하는 소프트웨어나 설정의 결함을 탐지할 수 있다. 모바일 앱 심사 시스템은 일반적으로 자동화된 테스트와 분석 도구를 포함하며, 외부 심사 서비스와 연계할 수 있다. 이 부록에서는 모바일 기기에 영향을 미치는 다양한 종류의 악성코드에 대해 설명한다. 모바일 앱 심사 시스템은 이러한 악성코드의 증거를 찾도록 설계되었다.

## A.1 랜섬웨어

랜섬웨어는 데이터를 암호화하고 복호화 키를 인질로 금전을 요구하는 악성코드이다. [38] 모바일 환경에서, 사용자의 데이터를 암호할 뿐만 아니라 잠금화면의 PIN 번호를 변경하여 기기를 잠그는 신종 랜섬웨어가 발견되었다. [39] 이러한 랜섬웨어는 침해된 웹 사이트를 통해 위조된 소트프웨어 업데이트로 확산되었다.

## A.2 스파이웨어

스파이웨어는 개인이나 조직이 인지하지 못한 상태에서 개인과 조직의 정보를 수집하고, 공격자의 시스템으로 수집한 정보를 전송하도록 설계된 악성코드이다. [40] 스파이웨어는 주로 웹에서 사용자의 인터넷 사용 기록을 추적하는 용도로 사용되지만, SMS 메시지, 사진, 통화 내역, 인증 정보나 금융 정보와 같은 중요한 데이터를 수집하기 위한 용도로 사용할 수 있다. 대부분의 스파이웨어는 사용자나 조직이 인지하지 못한 상태에서 설치되지만, 사기적인 방법으로 사용자를 속여 설치하기도 한다. 스파이웨어는 일반적으로 합법적이며, 부모가 자녀를 감시하거나 속임수를 쓰는 배우자를 붙잡을 수 있는 도구로 판매되는 경우도 많다. 또한, 국가적 차원에서 모바일 사용자의 정보를 수집하기 위해 스파이웨어를 사용했다. [41]

## A.3 애드웨어

애드웨어는 광고의 일환으로 내부에 탑재되거나 호출되는 악성코드이며, 전 세계적으로 모바일 기기에 대한 가장 공통적인 위협 중 하나이다. 현재 모바일 생태계에서 모바일 광고는 중요한다. 무료 모바일 앱을 제공하는 소프트웨어 개발자의 수입원이 되기 때문이다. 광고는 타사 웹사이트를 통해 제공될 수 있고, 사용자의 허가가 없거나 인지하지 못한 상태에서 개인 정보를 수집하는 악성코드가 포함되기도 한다. 최근 보고서에 의하면, 일부 저가형 모바일 기기에 애드웨어가 제조업체에 의해 사전 설치되어 운송되었다. [42] 이 모바일 기기의 사용자는 파업 광고와 기타 성가신 문제를 겪는다. 또한 애드웨어가 펌웨어 수준에서 설치되기 때문에 제거하기 매우 어렵다.

## A.4 루터

루터는 사용자가 모바일 기기를 루팅할 수 있는 도구이다. 루팅은 사용자가 기기의 운영체제에서 루트 권한을 획득할 수 있도록 하는 프로세스이다. 루팅은 통신 사업자 및 기기 제조업체가 설정한 제한을 우회하기 위해 수행한다. 루팅은 시스템 애플리케이션과 설정을 변경하거나 대체하고, 관리자 권한이 필요한 특별한 앱을 실행하거나 통신 사업자가 금지한 작업을 수행할 수 있게 한다. 안드로이드와 같은 일부 모바일 플랫폼에서 루팅은 기기의 운영체제를 완전히 삭제하거나 대체(예 : 새로운 버전 설치)할 수 있다.

- 소프트 루팅은 일반적으로 루트 익스플로잇이라고 부르는 보안 취약점을 이용하여 서드파티 애플리케이션을 통해 수행된다.
- 하드 루팅은 바이너리 실행 파일의 플래싱이 필요하고, 수퍼 유저 권한이 부여된다.

## A.5 트로이 목마

트로이 목마는 이상이 없고 익숙한 소프트웨어를 가장하여 사용자를 속여 실행시키는 악성코드이다. 전통적인 컴퓨터 플랫폼의 경우, 공격자는 일반적으로 악성코드를 doc, jpg와 같이 잘 알려진 확장자를 사용하여 숨긴다. 사용자가 트로이 목마 파일을 열면, 악성코드가 실행되기 시작한다. 모바일 환경의 새로운 트렌드인 모바일 뱅킹 토로이 목마는 피해자가 자신의 은행에서 전송된 것처럼 보이는 피싱 메시지에 응답한 후 설치되는 악성코드이다. 모바일 뱅킹 트로이 목마는 금융 정보, 인증 정보 및 신용카드 정보를 수집한다.

## A.6 인포스틸러

인포스틸러는 감염된 시스템에서 인증 정보를 포함한 정보를 수집하여 공격자의 시스템으로 전송하는 트로이 목마이다. 탈취하는 가장 일반적인 형태의 정보는 사용자 인증 정보와 금융 정보이다. 인포스틸러는 일반적으로 전통적인 컴퓨터 플랫폼에 영향을 미치지만, 최근 모바일 플랫폼에 영향을 미치기 시작했다. 최근 보고서에서 안드로이드 기기의 구글 크롬 업데이트로 가장하고 백신을 중지시키는 것으로 보고되었다. 인포스틸러는 사용자의 뱅킹 정보, 통화 내역, SMS 메시지와 웹 방문 기록을 수집하여 원격 서버로 전송한다.

## A.7 적대적인 다운로더(Hostile Downloader)

적대적인 다운로드의 주요 목적은 통상 인터넷에서 컨텐츠를 다운로드하는 것이다. 다운로드한 컨텐츠에는 악성 앱, 다운로더나 시스템에 설치된 다른 소프트웨어에 대한 설정이나 명령, 추가적인 소프트웨어 컴포넌트를 포함하여 공격을 용이하게 한다. 예를 들어, 2017년, 공격자는 스팸 메일에 뱅킹 트로이 목마를 실행하는 악성 파워포인트 파일을 첨부하였다. [46] 파워포인트 파일을 열고, 하이퍼링크에 마우스를 단지 올려놓기만 해도 트로이 목마를 다운로드하는 악의적인 스크립트가 실행된다.

## A.8 모바일 결제 사기

다수 모바일 서비스 제공자는 제품과 서비스에 대해 사용자의 모바일 서비스 계정에 과금하는 것을 허용하며, 사용자나 계정의 소유자가 매월 대금을 지불한다. 실제로 모바일 계정은 신용카드와 같은 역할을 수행함으로써 사용자에게 편의성을 제공하지만, 역설적으로 사기에 취약해진다. 은행을 이용하지 않는 저소득층과 같이 기존 신용 계정이 없는 사용자는 이동 통신사 결제를 통해 온라인 컨텐츠 또는 서비스를 구매한다.

사용자에 대한 이동 통신사의 사기, 이동 통신사에 대한 사용자의 사기, 사용자와 이동 통신사에 대한 제 3자의 사기가 발생했다. 미 연방 거래 위원회(FTC, Federal Trade Commission)는 승인되지 않은 서비스에 대한 수백만 달러의 요금을 고객에게 과다 청구한 협의로 AT&T, Verizon, Sprint에 대한 소송을 제기했다. [47][48] FTC는 모바일 고객에게 과다 청구를 예방하는 방법을 조언하였다. 이와 동시에, 이동 통신사는 고객에 의한 사기를 경험하고 있다. 이는 신용카드 사용자가 은행을 상대로 한 것과 유사하다. 사용자가 구매를 하고, 구매를 취소한 후 환불을 요청하는 것이 가장 일반적이다. 마지막으로, 제 3자에 의한 신원 도용이다. 모바일 기기 사용자의 식별 정보를 사용하여 모바일 계정을 탈취하고, 스마트 폰과 같은 신상품을 구매하면서 모바일 계정을 통해 청구한 후 현금으로 장비를 재판매한다. [50] 이동 통신 사업자는 새로운 기기의 활성화나 서비스 변경을 허용하기 전에 가입자에 대한 인증을 강화하기 위해 노력하고 있다.

## A.9 SMS 사기

한때 메일을 통해 저질러진 스캠(scam) 현재 SMS 메시지를 통해 저질러지고 있다. 사기성 거래, 스미싱(smishing), 허위 기부금 요청, 추첨 경품에 대한 수수료 및 데이트 사이트에서 유래한 속임수는 모두 SMS 스캠이다. [51] 사용자는 낯선 사람이나 모르는 번호에서 수신한 요청하지 않은 메시지, 특히 돈이나 개인정보, 민감한 정보에 대한 요청에 주의해야 한다.

## A.10 전화 사기

전화 사기는 여러 가지 악의적인 불법 행위를 의미한다. 예를 들어, 일부 이동 통신 서비스 사용자는 국내 지역번호에서 발신된 것처럼 보이지만 실제로는 국제 유료 통화 서비스인 전화를 받을 수 있다. 이러한 통화는 벨소리가 한 번 울린 후 끊어진다. 피해자가 전화를 걸고 끊지 않으면, 국제 통화료와 상당한 금액의 분당 요금이 추가적으로 청구된다. 이러한 요금은 보통 피해자의 휴대전화 청구서에 프리미엄 서비스로 명시된다.

## A.11 크래밍

크래밍은 피해자가 주문하지 않은 수수료가 비공개된 서비스에 대해 피해자의 휴대전화 청구서에 전화료 또는 서비스 수수료와 같은 요금을 발생시키는 사기성 행위를 말한다. 이러한 요금은 정직하지 못한 서드파티에 의해 청구되는 경우가 많다. 통신 사업자는 서드파티가 서비스 요금을 사용자의 휴대전화 청구서에 과금하는 것을 허용한다. 서드파티에 의한 다른 형태의 전화 사기에는 "PBX 전화 걸기"가 포함된다. "PBX 전화 걸기"는 기업에 전화를 걸어, "9-0" 또는 다른 외부 유로 전화 번호로 전환하도록 요청한다. 다양한 사기 행위에 대한 자세한 내용은 FTC 및 CFCA(Communication Fraud Control Association)에서 이용할 수 있다.

## A.12 전화 요금 사기

전화 요금 사기는 모바일 기기 사용자가 프리미엄 서비스를 사용하여 전화를 걸 때 발생한다. 해커가 전화를 건 사람에게 과금을 하는 서비스에서 전화 번호를 임대하고, 일정한 비율의 수익을 제공받는 것이 일반적인 공격 방법이다. 전화 요금 사기의 수익성을 높이기 위해 해커는 독립된 비즈니스 VoIP 네트워크에 침투하여 해커의 프리미엄 서비스 번호로 전화가 전달되게 한다. 회사에는 해커가 임대한 웹 기반 서비스를 호출한 것에 대한 요금이 청구되고, 해커는 일정 비율의 이익을 얻는다. 이러한 유형의 공격을 차단하기 위해 조직은 네트워크 보안을 강력하게 구축해야 한다.

# 부록 B – 안드로이드 앱 취약점 유형

이 부록은 안드로이드 모바일 기기에서 실행되는 앱에 대한 취약점을 설명한다. 이 부록에는 자바(Java)로 작성되어 안드로이드 기반 모바일 기기에서 실행되는 앱의 취약점에 대해 설명하며, 모바일 플랫폼 하드웨어와 통신 네트워크에 대한 취약점에 대해서는 설명하지 않는다. 아래 명시한 취약점 중 일부는 전반적인 모바일 기기 환경에서 공통적으로 적용되는 취약점이지만, 이 부록에서는 안드로이드 관련 취약점에 초점을 맞추고 있다.

이 부록에서는 취약점을 A 레벨, B 레벨, C 레벨의 세 가지 계층 구조로 구분한다. A 레벨은 취약점 계층이라고 하며, A 레벨로 분류된 취약점에 대해 광범위하게 설명하고 있다. B 레벨은 하위 계층이라고 하며, A 레벨로 분류된 취약점을 더 작은 일반적인 취약점 그룹으로 분류한다. C 레벨은 식별된 개별 취약점을 열거한다. 이렇게 계층 구조로 설계한 목적은 취약점을 빠르게 검색할 수 있도록 하는 것이다.

표 4에서는 안드로이드 앱 취약점에 대한 일반적인 분류를 보여주며, 표 5에서는 안드로이드 앱 취약점에 대한 계층 구조(A레벨 – C레벨)를 보여준다.

| 유형 | 설명 | 부정적인 결과 |
|---|---|---|
| 부정확한 권한 | 권한은 카메라, GPS와 같은 통제된 기능에 대한 접근할 수 있게 한다. 권한은 프로그램에서 요청한다.<br>권한은 사용자의 동의 없이 앱에 암시적으로 부여될 수 있다. | 과도한 권한이 허용된 앱은 의도한 기능 기능의 범위를 벗어나 의도하지 않은 기능을 수행할 수 있다. 더불어, 권한은 다른 앱에 의한 하이잭킹에 취약하다. 너무 적은 권한이 부여된 경우, 앱은 요구된 기능을 수행할 수 없다. |
| 통신 노출 | 내부 통신 프로토콜은 앱이 기기 내부에서 자신 또는 다른 앱과 메시지를 전달하기 위한 수단이다. 외부 통신은 정보를 기기 외부로 전송한다. | 내부 통신이 노출되면 다른 앱에 의해 의도하지 않은 정보가 수집될 수 있고, 새로운 정보가 삽입될 수 있다. 외부 통신(데이터 통신, 와이파이, 블루투스, NFC 등)이 노출되면 정보가 공개되거나 중간자 공격에 노출된다. |
| 잠재적으로 위험한 기능 | 세심한 관리가 필요한 기능으로 중요한 시스템 자원이나 사용자의 개인정보에 접근하는 기능을 말한다.<br>API를 통해 호출되거나 앱 내부에 하드 코딩될 수 있다. | 앱 기능 범위를 초과하는 의도되지 않은 기능을 수행할 수 있다. |
| 앱 콜루전 (collusion) | 2개 이상의 앱이 기능 향상을 위해 선언된 범위를 초과한 정보를 상호 교환하는 것을 말한다. | 게임 앱이 사용자의 연락처에 접근하는 것과 같이 의도되지 않은 데이터를 획득할 수 있다. |
| 난독화 | 사용자에게 기능이나 제어 흐름을 숨기거나 불명확하게 하는 것을 말한다.<br>이 부록에서는 난독화를 외부 라이브러리 호출, 리플렉션, 원시 코드 사용으로 분류한다. | 1. 외부 라이브러리에는 불필요하거나 악의적인 기능을 포함될 수 있다.<br>2. 리플렉션은 앱의 제어 흐름을 불명확하게 하고 앱의 권한을 와해시킬 수 있다.<br>3. 원시 코드(안드로이드에서 자바가 아닌 다른 언어로 작성된 코드)는 불필요하거나 악의적인 기능을 수행할 수 있다. |
| 과도한 전원 소비 | 배터리를 소모하는 과도한 기능이나 의도하지 않은 앱을 말한다. | 배터리의 수명이 짧아지면 업무 핵심 기능을 수행하는데 영향을 줄 수 있다. |
| 기존 소프트웨어 취약점 | 인증, 접근 통제, 버퍼 처리, 제어 흐름 관리, 암호화와 무작위성(randomness), 에러 처리, 파일 처리, 정보 노출, 초기화와 종료, 인젝션, 악의적인 로직, 숫자 처리, 포인터와 레퍼런스 처리와 같은 자바 코드와 관련된 기존의 모든 취약점을 말한다. | 일반적으로 예상하지 못한 결과, 자원 고갈, 서비스 거부(DoS) 등의 원인이 될 수 있다. |

표 4 – 안드로이드 취약점, 레벨 A

| A 레벨 | B 레벨 | C 레벨 |
|---|---|---|
| 권한 | 과다 부여 | 코드에서 과다 부여 |
| | | API에서 과다 부여 |
| | 과소 부여 | 코드에서 과소 부여 |
| | | API에서 과소 부여 |
| | 개발자가 생성한 권한 | 코드에서 개발자가 생성 |
| | | API에서 개발자가 생성 |
| | 불명확한 권한 | API를 통해 부여 |
| | | 다른 권한을 통해 부여 |
| | | Grandfathering[13]을 통해 부여 |
| 통신 노출 | 외부 통신 | 블루투스 |
| | | GPS |
| | | 데이터 통신 |
| | | NFC |
| | 내부 통신 | 보호되지 않은 인텐트 |
| | | 보호되지 않은 액티비티 |
| | | 보호되지 않은 서비스 |
| | | 보호되지 않은 컨텐트 프로바이더 |
| | | 보호되지 않은 브로드캐스트 리시버 |
| | | 플래그 디버깅 |
| 잠재적으로 위험한 기능 | 직접 접근 | 메모리 접근 |
| | | 인터넷 접근 |
| | 잠재적으로 위험한 API | 결재 관련 API |
| | | 개인 정보 관련 API |
| | | 기기 관리 API |
| | 권한 상승 | 파일 권한 변경 |
| | | 수퍼 유저나 루트 접근 |
| 앱 콜루전 | 컨텐트 프로바이더 및 인텐트 | 보호되지 않은 컨텐트 프로바이더 |
| | | 보호된 컨텐트 프로바이더 권한 |
| | | 보류(pending)된 인텐트 |
| | 브로드캐스트 리시버 | 중요한 메시지에 대한 브로드캐스트 리시버 |
| | 데이터 생성, 변경 및 삭제 | 파일 생성, 변경 및 삭제 |
| | | 데이터베이스 생성, 변경 및 삭제 |
| | 다수 서비스 | 서비스 상태에 대한 과도한 확인 |
| 난독화 | 라이브러리 호출 | 잠재적으로 위험한 라이브러리 사용 |
| | | 잠재적으로 악의적인 미사용 라이브러리 |
| | 원시 코드 탐지 | |
| | 리플렉션 | |
| | 패키징 | |
| 과도한전원 소비 | CPU 사용량 | |
| | I/O | |

표 5 – 레벨 별 안드로이드 취약점

[13] 새로운 규제 등에 의해 어떤 행위가 금지되었을 때, 규제가 제정되기 이전에 그 행위를 수행하던 주체에 대해서는 예외를 인정해 주는 것을 말한다.

# 부록 C – iOS 앱 취약점 유형

이 부록은 애플 iOS 운영체제에서 실행되는 앱에 대한 취약점을 설명한다. 모바일 플랫폼 하드웨어와 통신 네트워크에 대한 취약점에 대해서는 설명하지 않는다. 아래 명시한 취약점 중 일부는 전반적인 모바일 기기 환경에서 공통적으로 적용되는 취약점이지만, 이 부록에서는 iOS 관련 취약점에 초점을 맞추고 있다.

이 부록에서는 취약점을 A 레벨, B 레벨, C 레벨의 세 가지 계층 구조로 구분한다. A 레벨은 취약점 계층이라고 하며, A 레벨로 분류된 취약점에 대해 광범위하게 설명하고 있다. B 레벨은 하위 계층이라고 하며, A 레벨로 분류된 취약점을 더 작은 일반적인 취약점 그룹으로 분류한다. C 레벨은 식별된 개별 취약점을 열거한다. 이렇게 계층 구조로 설계한 목적은 취약점을 빠르게 검색할 수 있도록 하는 것이다.

표 6에서는 iOS 앱 취약점에 대한 일반적인 분류를 보여주며, 표 7에서는 iOS 앱 취약점에 대한 계층 구조(A레벨 - C레벨)를 보여준다.

| 유형 | 설명 | 부정적인 결과 |
|---|---|---|
| 프라이버시 | 안드로이드의 권한과 유사하다. iOS의 프라이버시 설정을 통해 앱은 중요한 정보에 접근할 수 있다. 중요한 정보는 연락처, 캘린더, 미리 알림, 작업, 사진, 블루투스 접근 등을 포함한다. | iOS는 공유할 정보를 생성하고 보호하는 기능이 존재하지 않는다. 정보를 공유하기 위해서는 iOS 앱 프레임워크를 거쳐야만 하며, 이런 방식이 변경될 것 같지 않다. 안드로이드와 다르게, 추후 앱 별로 개별 권한을 수정할 수 있다. |
| 통신 노출 | 내부 통신 프로토콜은 앱이 정보를 처리하고 다른 앱과 통신하기 위한 수단이다. 외부 통신은 정보를 기기 외부로 전송한다. | 내부 통신이 노출되면 다른 앱에 의해 의도하지 않은 정보가 수집될 수 있고, 새로운 정보가 삽입될 수 있다. 외부 통신(데이터 통신, 와이파이, 블루투스 등)이 노출되면 정보가 공개되거나 중간자 공격에 노출된다. |
| 잠재적으로 위험한 기능 | 세심한 관리가 필요한 기능으로 중요한 시스템 자원이나 사용자의 개인정보에 접근하는 기능을 말한다.<br>API를 통해 호출되거나 앱 내부에 하드 코딩될 수 있다. | 앱 기능 범위를 초과하는 의도되지 않은 기능을 수행할 수 있다. |
| 앱 콜루전 (collusion) | 2개 이상의 앱이 기능 향상을 위해 선언된 범위를 초과한 정보를 상호 교환하는 것을 말한다. | 게임 앱이 사용자의 연락처에 접근하는 것과 같이 의도되지 않은 데이터를 획득할 수 있다. |
| 난독화 | 사용자에게 기능이나 제어 흐름을 숨기거나 불명확하게 하는 것을 말한다.<br>이 부록에서는 난독화를 외부 라이브러리 호출, 리플렉션, 패킹으로 분류한다. | 1. 외부 라이브러리에는 불필요하거나 악의적인 기능을 포함될 수 있다.<br>2. 리플렉션은 앱의 제어 흐름을 불명확하게 하고 앱의 권한을 와해시킬 수 있다.<br>3. 패킹은 리버스 엔지니어링을 어렵게 하며, 악성코드를 은닉하는데 사용될 수 있다. |
| 과도한 전원 소비 | 배터리를 소모하는 과도한 기능이나 의도하지 않은 앱을 말한다. | 배터리의 수명이 짧아지면 업무 핵심 기능을 수행하는데 영향을 줄 수 있다. |
| 기존 소프트웨어 취약점 | 인증, 접근 통제, 버퍼 처리, 제어 흐름 관리, 암호화와 무작위성(randomness), 에러 처리, 파일 처리, 정보 노출, 초기화와 종료, 인젝션, 악의적인 로직, 숫자 처리, 포인터와 레퍼런스 처리와 같은 Objective-C와 관련된 기존의 모든 취약점을 말한다. | 일반적으로 예상하지 못한 결과, 자원 고갈, 서비스 거부(DoS) 등의 원인이 될 수 있다. |

표 6 – iOS 취약점, 레벨 A

| A 레벨 | B 레벨 | C 레벨 |
|---|---|---|
| 프라이버시 | 민감한 정보 | 연락처 |
| | | 캘린더 정보 |
| | | 작업 |
| | | 미리 알림 |
| | | 사진 |
| | | 블루투스 접근 |
| 통신 노출 | 외부 통신 | 전화 |
| | | 블루투스 |
| | | GPS |
| | | SMS/MMS |
| | | 데이터 통신 |
| | 내부 통신 | 프로토콜 핸들러 오용 |
| 잠재적으로 위험한 기능 | 메모리 직접 매핑 | 메모리 접근 |
| | | 파일 시스템 접근 |
| | 잠재적으로 위험한 API | 결재 관련 API |
| | | 기기 관리 API |
| | | 개인 정보 관련 API |
| 앱 콜루전 | 데이터 변경 | 공유 파일 리소스 변경 |
| | | 공유 데이터베이스 리소스 변경 |
| | | 공유 컨텐트 프로바이더 변경 |
| | 데이터 생성 및 삭제 | 공유 파일 리소스 생성 및 삭제 |
| 난독화 | 다수 서비스 | 과도한 서비스 상태 확인 |
| | 원시 코드 | 잠재적으로 악의적인 미사용 라이브러리 |
| | | 잠재적으로 위험한 라이브러리 사용 |
| | | 리플렉션 식별 |
| | | 클래스 인트로스펙션 |
| | 라이브러리 호출 | 생성자 인트로스펙션 |
| | | 필드 인트로스펙션 |
| | | 메소드 인트로스펙션 |
| | 패킹 | |
| 과도한 전원 소비 | CPU 사용량 | |
| | I/O | |

표 7 – 레벨 별 iOS 취약점

# 부록 D – 약어

이 문서에서 사용된 약어는 다음과 같다.

| | |
|---|---|
| API | Application Programming Interface |
| BYOD | Bring Your Own Device |
| CAPEC | Common Attack Pattern Enumeration and Classification |
| CERT | Computer Emergency Response Team |
| CPU | Central Processing Unit |
| CVE | Common Vulnerabilities and Exposures |
| CWE | Common Weakness Enumeration |
| DHS | Department of Homeland Security |
| DoD | Department of Defense |
| EMM | Enterprise Mobility Management |
| GPS | Global Positioning System |
| IEEE | Institute of Electrical and Electronics Engineers |
| I/O | Input/Output |
| IoT | Internet of Things |
| ISO | International Organization for Standardization |
| ITL | Information Technology Laboratory |
| JVM | Java Virtual Machine |
| NFC | Near Field Communication |
| NIST | National Institute of Standards and Technology |
| NVD | National Vulnerability Database |
| OMB | Office of Management and Budget |
| PII | Personally Identifiable Information |
| PIN | Personal Identification Number |
| PIV | Personal Identity Verification |
| SAMATE | Software Assurance Metrics and Tool Evaluation |
| SCAP | Security Content Automation Protocol |
| SLA | Service Level Agreement |
| SP | Special Publication |
| UI | User Interface |
| VPN | Virtual Private Network |
| Wi-Fi | Wireless Fidelity |

# 부록 E – 용어

이 문서에서 사용된 용어의 정의는 다음과 같다.

| | |
|---|---|
| Administrator<br>(관리자) | 조직의 모바일 기기에 대한 배포, 유지 및 보안 책임과 함께 배포된 장치와 설치된 앱에 대한 보안 요구사항 준수 여부에 대한 보증 책임이 있는 조직의 구성원이다. |
| App Security Requirement<br>(앱 보안 요구사항) | 앱의 보안을 보증하기 위한 요구사항으로 일반 요구사항과 조직 요구사항으로 구분한다. 일반 요구사항은 앱에 존재해야 하거나 제거되어야 하는 소프트웨어 및 행동 특성을 정의한다. 조직 요구사항은 조직의 보안 상태를 보증하기 위해 준수해야 하는 정책, 규제 및 지침을 정의한다. |
| Analyst<br>(분석가) | 앱이 충족해야 하는 조직의 보안 요구사항을 검증하기 위해 조직 특수한 기준, 다수 테스트 도구로부터 생성된 각종 보고서 및 위험 평가를 분석하는 조직의 구성원이다. |
| App Vetting Process<br>(앱 심사 프로세스) | 모바일 앱이 조직의 보안 요구사항을 충족하는지 판단하기 위한 조직에 의해 수행되는 일련의 활동이다. |
| App Vetting System<br>(앱 심사 시스템) | 앱 심사 프로세스를 관리하고 자동화하기 위한 시스템이다. |
| Authorizing Official<br>(승인 담당자) | 조직에서 사용할 앱을 승인하거나 거부하는 것을 결정하는 조직의 구성원이다. |
| Dynamic Analysis<br>(동적 분석) | 입력값을 주면서 앱을 실행하고, 앱의 런타임 동작을 분석하여 취약점을 탐지하는 것이다. |
| Enterprise Mobility Manager<br>(전사 모바일 관리자) | 모바일 기기, 무선 네트워크 및 모바일 컴퓨팅 서비스를 관리하는데 중점을 두는 사람, 프로세스, 기술의 집합이다. |
| Functionality Testing<br>(기능 테스트) | 앱의 UI 콘텐츠와 기능이 설계대로 수행되고 표시되는 확인하는 것이다. |

| | |
|---|---|
| Mobile Device Management<br>(모바일 기기 관리) | 스마트 폰, 태블릿, 노트북 및 데스트탑과 같은 모바일 장치를 관리하는 것이다. MDM은 일반적으로 특정 모바일 기기에 대한 관리 기능을 보유하고 있는 서드파티 제품으로 구현된다. |
| National Security System<br>(국가 보안 시스템) | 정부 기관에서 운영하거나, 정부에서 외주로 운영하는 모든 정보 시스템(전기 통신 시스템 포함)이다. 정보 활동, 국가 보안과 관련한 암호 활동, 군사 분야의 지휘 통제, 무기 및 무기 체계의 필수적인 부품의 기능, 동작 또는 사용을 포함한다.<br>국방이나 외교 정책의 이익을 위해 기밀로 지정된 정보(행정 명령 또는 의회법에 의해 특별 승인)는 수립된 절차에 항상 보호된다. [52] |
| Personally Identifiable Information<br>(개인 식별 정보) | 악의적인 행위자가 개인의 신원을 식별하거나 추적하기 위해 사용할 수 있는 개인에 대한 정보 및 개인과 연결되거나 연결할 수 있는 정보이다. [45] |
| Risk Assessment<br>(위험 평가) | 앱을 사용할 때 테스트 도구가 예상하는 보안 위험의 수준을 나타내는 값이다. 위험 평가는 일반적으로 탐지된 약점이 악용될 가능성과 발견된 취약점이 앱/관련 장치/네트워크에 미칠 수 있는 영향을 기반으로 한다. 위험 평가는 일반적으로 저-중-고 위험과 같은 카테고리로 표현한다. |
| Static Analysis<br>(정적 분석) | 앱의 소스 코드와 바이너리를 검사하고 런타임에서 발생할 수 있는 모든 가능한 행위를 판단하여 소프트웨어의 취약점을 찾는 것이다. |
| Software Assurance<br>(소프트웨어 보증) | 소프트웨어에 고의적으로 설계되었거나 수명주기 동안 우연히 삽입된 취약점으로부터 소프트웨어가 안전하고, 의도한 방식으로 동작한다는 확신의 수준이다. |
| Software Correctness Testing<br>(소프트웨어 정확도 테스트) | 프로그램의 오류를 찾기 위해 프로그램을 실행하는 프로세스이다. 이 테스트의 목적은 품질 보증 개선하고, 기능을 검증하고 확인하며, 신뢰성 추정하는 것이다. |
| Software Vulnerability<br>(소프트웨어 취약점) | 소프트웨어에서 발견되는 보안 결함, 약점으로 공격자에 의해 악용될 수 있다. |
| Test Tool<br>(테스트 도구) | 특정 소프트웨어에 취약점이 존재하는지 판단하는 도구나 서비스이다. |

# 부록 F – 참고 문헌


[1] P. E. Black, L. Badger, B. Guttman, and E. Fong, "Dramatically reducing software vulnerabilities: Report to the White House Office of Science and Technology Policy," National Institute of Standards and Technology, Gaithersburg, MD, NIST IR 8151, Nov. 2016.

[2] R. Kissel, "Glossary of key information security terms," National Institute of Standards and Technology, NIST IR 7298r2, May 2013.

[3] M. Souppaya and K. Scarfone, "Guidelines for Managing the Security of Mobile Devices in the Enterprise," National Institute of Standards and Technology, NIST SP 800-124r1, Jun. 2013.

[4] "Protection Profile for Mobile Device Fundamentals," p. 183.

[5] Joint Task Force Transformation Initiative, "Security and Privacy Controls for Federal Information Systems and Organizations," National Institute of Standards and Technology, NIST SP 800-53r4, Apr. 2013.

[6] ISO/IEC, "Information technology -- Security techniques -- Evaluation criteria for IT security," ISO/IEC 15408-1:2009.

[7] "Requirements for Vetting Mobile Apps from the Protection Profile for Application Software." [Online]. Available: https://www.niap-ccevs.org/MMO/PP/394.R/pp_app_v1.2_table-reqs.htm. [Accessed: 22-Jun-2018].

[8] OWASP, "Mobile Application Security Verification Standard," v0.9.4.

[9] "OWASP Mobile Security Testing Guide - OWASP." [Online]. Available: https://www.owasp.org/index.php/OWASP_Mobile_Security_Testing_Guide. [Accessed: 22-Jun-2018].

[10] M. Peck and C. Northern, "Analyzing the Effectiveness of App Vetting Tools in the Enterprise," p. 46, Aug. 2016.

[11] "Build Security In | US-CERT." [Online]. Available: https://www.us-cert.gov/bsi#ques. [Accessed: 22-Jun-2018].

[12] "CVE - Common Vulnerabilities and Exposures (CVE)." [Online]. Available: https://cve.mitre.org/. [Accessed: 22-Jun-2018].

[13] I. T. L. Computer Security Division, "AppVet | CSRC." [Online]. Available: https://csrc.nist.gov/projects/appvet/. [Accessed: 22-Jun-2018].

[14] M. Pezzè and M. Young, Software Testing and Analysis: Process, Principles and Techniques. Hoboken, New Jersey: John Wiley & Sons, Inc., 2008.

[15] G. McGraw, Software security: building security in. Upper Saddle River, NJ: Addison-Wesley, 2006.

[16] G. G. Schulmeyer, Handbook of Software Quality Assurance, Fourth Edition. Norwood, Massachusetts: Artech House, Inc., 2008.

[17] B. B. Agarwal, S. P. Tayal, and M. Gupta, Software engineering & testing: an introduction. Sudbury, Mass: Jones and Bartlett, 2010.

[18] J. R. Maximoff, M. D. Trela, D. R. Kuhn, and R. Kacker, "A method for analyzing system state-space coverage within a t-wise testing framework," 2010, pp. 598–603.
[19] G. J. Myers, T. Badgett, T. M. Thomas, and C. Sandler, The art of software testing, 2nd ed. Hoboken, N.J: John Wiley & Sons, 2004.
[20] H. CHEN, T. ZOU, and D. WANG, "Data-flow Based Vulnerability Analysis and Java Byte code," 7th WSEAS Int. Conf. Appl. Comput. Sci. Venice Italy Novemb. 21-23 2007, p. 7.
[21] "FindBugsTM 1122 - Find Bugs in Java Programs." [Online]. Available: http://findbugs.sourceforge.net/. [Accessed: 22-Jun-2018].
[22] R. Shah, "Vulnerability Assessment of Java Bytecode," Auburn University, 2005.
[23] WAIM (Conference) et al., The Ninth International Conference on Web-Age Information Management: WAIM 2008. Piscataway, N.J.: IEEE, 2008.
[24] "CWE - Common Weakness Enumeration." [Online]. Available: http://cwe.mitre.org/. [Accessed: 22-Jun-2018].
[25] FIRST.org, Inc, "CVSS v3.0 Specification," p. 21.
[26] "Mobile Application Security Requirments." Department of Defense, 06-Oct-2017.
[27] M. Dowd, J. McDonald, and J. Schuh, The art of software security assessment: identifying and preventing software vulnerabilities. Indianapolis, Ind: Addison-Wesley, 2007.
[28] H. G. Rice, "Classes of recursively enumerable sets and their decision problems," Trans. Am. Math. Soc., vol. 74, no. 2, pp. 358–358, Feb. 1953.
[29] J. H. Allen, Ed., Software security engineering: a guide for project managers. Upper Saddle River, NJ: Addison-Wesley, 2008.
[30] Archiveddocs, "The STRIDE Threat Model." [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v%3dcs.20). [Accessed: 22-Jun-2018].
[31] "Trike: Trike." [Online]. Available: http://www.octotrike.org/home.shtml. [Accessed: 22-Jun-2018].
[32] "Tool Survey - SAMATE." [Online]. Available: https://samate.nist.gov/index.php/Tool_Survey.html. [Accessed: 22-Jun-2018].
[33] M. A. Ogata, "An overview of mobile application vetting services for public safety," National Institute of Standards and Technology, Gaithersburg, MD, NIST IR 8136, Jan. 2017.
[34] "NVD - Home." [Online]. Available: https://nvd.nist.gov/. [Accessed: 22-Jun-2018].
[35] I. T. L. Computer Security Division, "Security Content Automation Protocol | CSRC." [Online]. Available: https://csrc.nist.gov/projects/security-content-automation-protocol/. [Accessed: 22-Jun-2018].
[36] "CAPEC - Common Attack Pattern Enumeration and Classification (CAPEC)." [Online]. Available: https://capec.mitre.org/. [Accessed: 22-Jun-2018].
[37] J. M. Stecklein, B. Dick, B. Haskins, R. Lovell, and G. Moroney, "Error Cost Escalation Through the Project Life Cycle," presented at the 14th Annual International Symposium, Toulouse; France, 2004.

[38] M. Bartock, J. Cichonski, M. Souppaya, M. Smith, G. Witte, and K. Scarfone, "Guide for cybersecurity event recovery," National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-184, Dec. 2016.

[39] S. Khandelwal, "New Ransomware Not Just Encrypts Your Android But Also Changes PIN Lock," The Hacker News. [Online]. Available: https://thehackernews.com/2017/10/android-ransomware-pin.html. [Accessed: 22-Jun-2018].

[40] W. A. Jansen, T. Winograd, and K. Scarfone, "Guidelines on Active Content and Mobile Code," p. 62.

[41] "When 'Grandma-Proof' Android Spyware Is Good Enough For International Espionage." [Online]. Available: https://www.forbes.com/sites/thomasbrewster/2018/05/15/apple-iphone-spouseware-used-in-pakistan-government-attacks/#74f2e2515668. [Accessed: 22-Jun-2018].

[42] S. Dent, "Report finds Android malware pre-installed on hundreds of phones," Engadget, 24-May-2018. [Online]. Available: https://www.engadget.com/2018/05/24/report-finds-android-malware-pre-installed-on-hundreds-of-phones/. [Accessed: 22-Jun-2018].

[43] H. Zhang, D. She, and Z. Qian, "Android Root and its Providers: A Double-Edged Sword," 2015, pp. 1093–1104.

[44] J. Mello Jr, "Marcher Malware Poses Triple Threat to Android Users | Malware | TechNewsWorld," Tech News Workd, 07-Nov-2017. [Online]. Available: https://www.technewsworld.com/story/84936.html. [Accessed: 22-Jun-2018].

[45] D. Palmer, "Irremovable bank data-stealing Android malware poses as Google Chrome update," ZDNet. [Online]. Available: https://www.zdnet.com/article/irremovable-bank-detail-stealing-android-malware-poses-as-google-chrome-update/. [Accessed: 22-Jun-2018].

[46] M. Moon, "Malware downloader infects your PC without a mouse click," Engadget. [Online]. Available: https://www.engadget.com/2017/06/11/malware-downloader-infects-your-pc-without-a-mouse-click/. [Accessed: 22-Jun-2018].

[47] L. Whitney, "AT&T to pay $105 million to settle charges over mobile billing," CNET, 08-Oct-2014. [Online]. Available: https://www.cnet.com/news/at-t-to-pay-105-million-to-settle-fraudulent-mobile-bill-charges/. [Accessed: 22-Jun-2018].

[48] J. Brodkin, "Verizon and Sprint pay $158 million in fines for fraudulent phone charges," Ars Technica, 12-May-2015. [Online]. Available: https://arstechnica.com/tech-policy/2015/05/verizon-and-sprint-pay-158-million-in-fines-for-fraudulent-phone-charges/. [Accessed: 22-Jun-2018].

[49] "Mystery Phone Charges," Consumer Information, 16-Dec-2013. [Online]. Available: https://www.consumer.ftc.gov/articles/0183-mystery-phone-charges. [Accessed: 22-Jun-2018].

[50] H. Weisbaum, "Fraud Alert: ID Thieves Hijack Mobile Phone Accounts," NBC News. [Online]. Available: https://www.nbcnews.com/tech/tech-news/fraud-alert-id-thieves-hijack-mobile-phone-accounts-n599761. [Accessed: 22-Jun-2018].

[51] L. Spector, "5 common SMS text scams, and how to avoid them | PCWorld," PC World, 01-Mar-2016. [Online]. Available: https://www.pcworld.com/article/3034696/mobile/5-common-sms-text-scams-and-how-to-avoid-them.html. [Accessed: 22-Jun-2018].
[52] Federal Information Security Modernization Act of 2014. 2014.