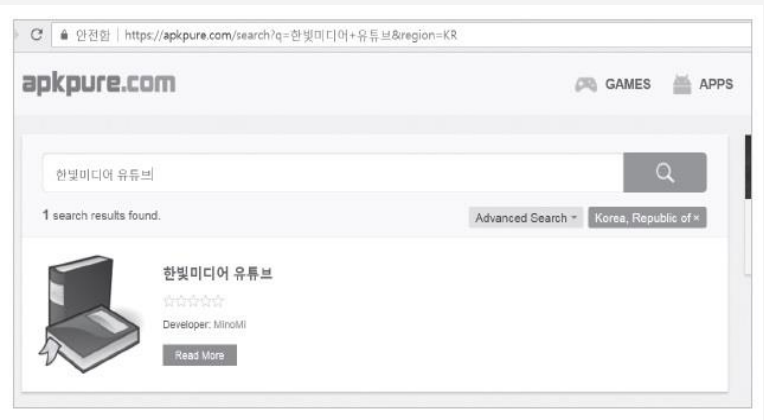
1 모바일 위협 요소에 대한 대응 방안

1 안드로이드 앱 정적 분석하기

1. apk 파일 구하기(첫번째)
   - 다음 사이트에서 검색하여 특정 apk 파일을 내려받음
   - https://apkpure.com

[한빛미디어 유튜브의 apk파일 다운로드 화면]



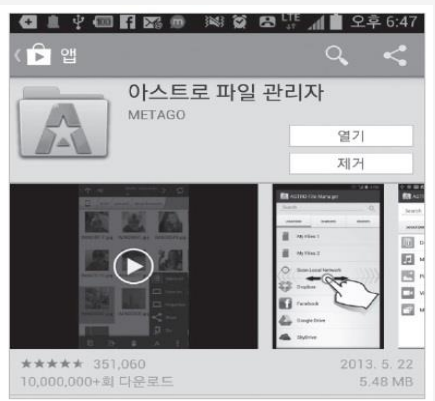
※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

1 안드로이드 앱 정적 분석하기

2 아스트로 파일 관리자 앱으로 apk 파일 추출(두번째)

- 안드로이드 기반 스마트폰에서 아스트로 파일 관리자 앱을 이용하여 스마트폰에 설치된 앱을 apk 확장자로 백업

[아스트로 파일 관리자]



※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

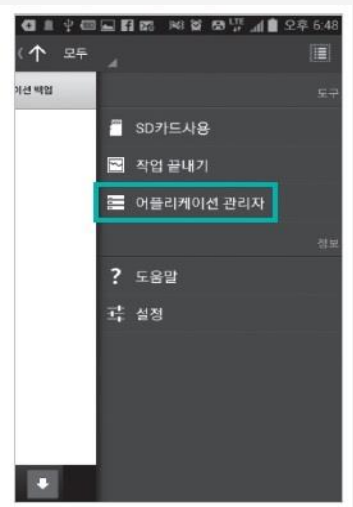
1 안드로이드 앱 정적 분석하기

2 아스트로 파일 관리자 앱으로 apk 파일 추출

- 설정에서 앱의 백업 경로를 지정하고, [어플리케이션 관리자]를 클릭
- 원하는 앱을 선택하고 〈백업〉을 클릭하여 파일 복사 시작

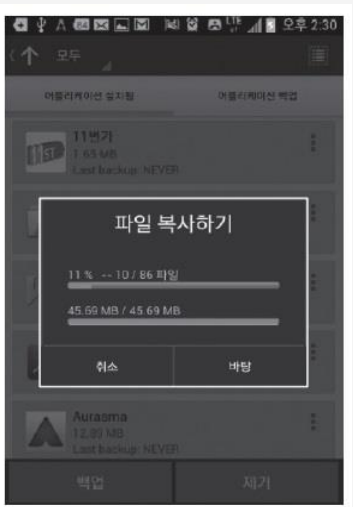
1 안드로이드 앱 정적 분석하기

2 아스트로 파일 관리자 앱으로 apk 파일 추출



[어플리케이션관리자]메뉴

[어플리케이션]백업



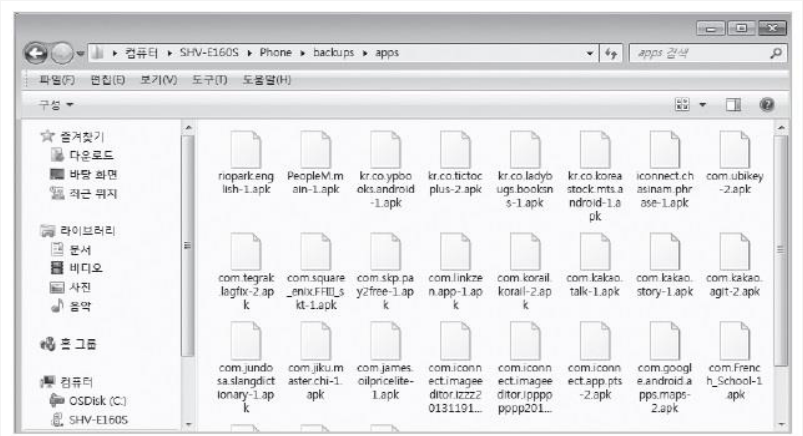
[어플리케이션]백업 실행화면

1 안드로이드 앱 정적 분석하기

2 아스트로 파일 관리자 앱으로 apk 파일 추출

- 백업이 완료되면 스마트폰과 컴퓨터를 연결한 후 지정한 폴더를 열어 apk 파일을 확인(apk 파일)

[추출한 apk 파일]



※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

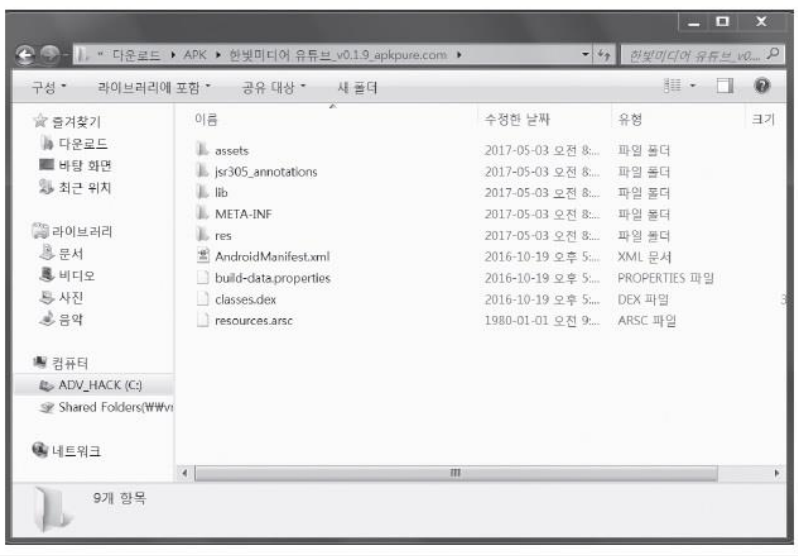
1 안드로이드 앱 정적 분석하기

3 압축 풀기로 dex 파일 추출

- 7-zip을 설치한 뒤 apk 파일을 선택하고
  마우스 오른쪽 버튼을 눌러 7-zip으로 압축 풀기
  (apk → 7-zip → dex)
- 다운로드 : http://www.7-zip.org/download.html

1 안드로이드 앱 정적 분석하기

3 압축 풀기로 dex 파일 추출(apk → 7-zip → dex)



[7-zip으로 apk 파일의 압축을 해제한 화면]

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

1 안드로이드 앱 정적 분석하기

3 압축 풀기로 dex 파일 추출(apk → 7-zip → dex)

[apk 파일의 압축을 풀었을때 생성되는 폴더와 파일]

경로	설명
apkName/assets/	• 앱에 필요한 각종 자원이 담겨 있는 폴더로, 데이터베이스 파일도 이 폴더에 있다.
apkName/lib/	• 앱에서 사용하는 라이브러리 파일이 담겨 있는 폴더이다.
apkName/META-INF/	• 앱의 메타 정보가 담겨 있는 폴더이다.
apkName/res/	• 앱에서 사용하는 리소스 파일이 담겨 있는 폴더이다.

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

1 안드로이드 앱 정적 분석하기

3 압축 풀기로 dex 파일 추출(apk → 7-zip → dex)

[apk 파일의 압축을 풀었을때 생성되는 폴더와 파일](달빅)

경로	설명
apkName/AndroidManifest.xml	• 앱의 이름, 버전, 구성 요소, 권한 등을 담고 있는 파일이다. • 단순 7-zip으로 apk 파일을 풀었을 때는 읽을 수 없는 형태이다.
apkName/build-data.properties	• 앱의 빌드 정보를 담고 있는 파일이다.
apkName/classes.dex	• 달빅 실행 파일이다.
apkName/resources.arsc	• 리소스 정보가 담겨 있는 파일이다. • 단순 7-zip으로 apk 파일을 풀었을 때는 읽을 수 없는 형태이다.

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

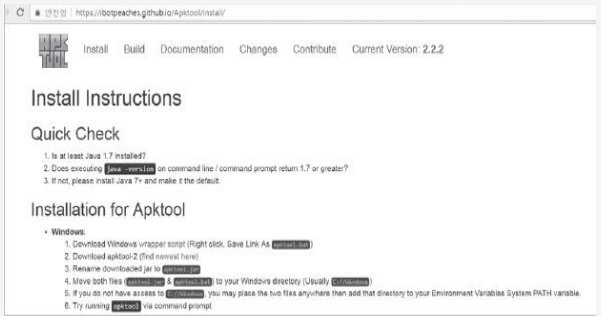
1 안드로이드 앱 정적 분석하기

4 apktool로 스마일리 파일 추출(apk → apktool → smali)

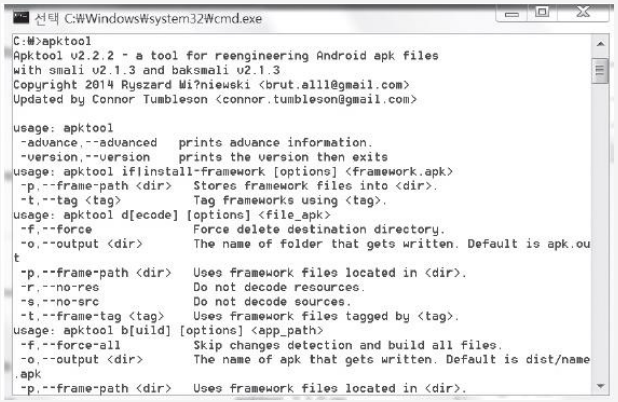
- apk 파일로 패키징되어 있는 앱을 언패키징하여 내부 파일을 보기 위해 apktool을 사용
- 다운로드
  : https://ibotpeaches.github.io/Apktool/

1 안드로이드 앱 정적 분석하기

4 apktool로 스마일리 파일 추출(apk → apktool → smali)



[apktool의 다운로드 및 설치 설명 화면]



[apktool이 성공적으로 실행된 화면]

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

1 안드로이드 앱 정적 분석하기

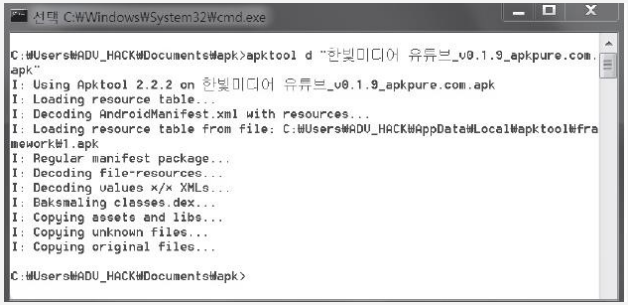
4 apktool로 스마일리 파일 추출(apk → apktool → smali)

- apktool의 다운로드 및 설치(윈도우를 예로 살펴봄)
  - wrapper script 링크를 마우스 오른쪽 버튼으로 눌러 나타나는 메뉴에서 [다른 이름으로 대상(링크) 저장]을 선택, 해당 파일을 특정 폴더apktool.bat로 저장
  - find newest here 링크를 클릭하면 나타나는 사이트에서 최신 버전의 apktool을 다운로드하여 apktool.bat 파일이 있는 폴더로 복사

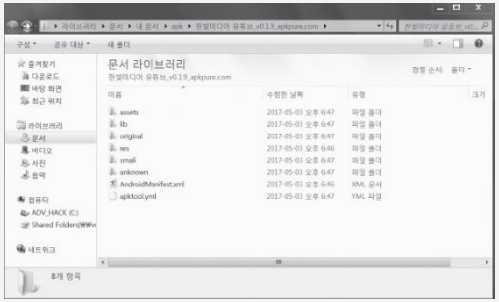
1 안드로이드 앱 정적 분석하기

4 apktool로 스마일리 파일 추출(apk → apktool → smali)

- apktool의 다운로드 및 설치(윈도우를 예로 살펴봄)
  - apktool d [압축을 풀려는 apk 파일]



[apktool을 이용하여 apk 파일 압축풀기]



[apktool로 압축이 풀린 폴더]

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

1 안드로이드 앱 정적 분석하기

4 apktool로 스마일리 파일 추출(apk → apktool → smali)

[apktool로 압축을 풀었을 때 생성되는 주요 폴더와 파일]

경로	용도
apkName/res/	• drawable, layout, raw, value 등의 하위 디렉터리를 포함한다. • 앱에 필요한 각종 리소스가 담겨 있다.
apkName/smali/	• smali 언어로 된 중간 소스코드를 포함한다.
apkName/AndroidManifest.xml	• 앱의 설정과 관련된 정보를 포함한다.
apkName/apktool.yml	• apktool에 의해 생성되는 파일이다.

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

1 안드로이드 앱 정적 분석하기

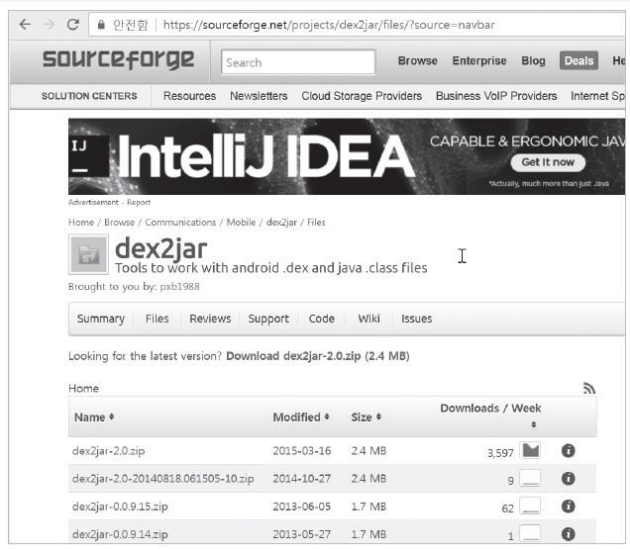
5 dex 파일에서 jar 파일 추출(dex → dex2jar → jar)

- dex 파일에서 jar 파일을 만들려면 dex2jar 프로그램이 필요
- 다운로드 : https://sourceforge.net/projects/dex2jar/files/

1 안드로이드 앱 정적 분석하기

5 dex 파일에서 jar 파일 추출(dex → dex2jar → jar)

[dex2jar 다운로드 화면]



※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

1 안드로이드 앱 정적 분석하기

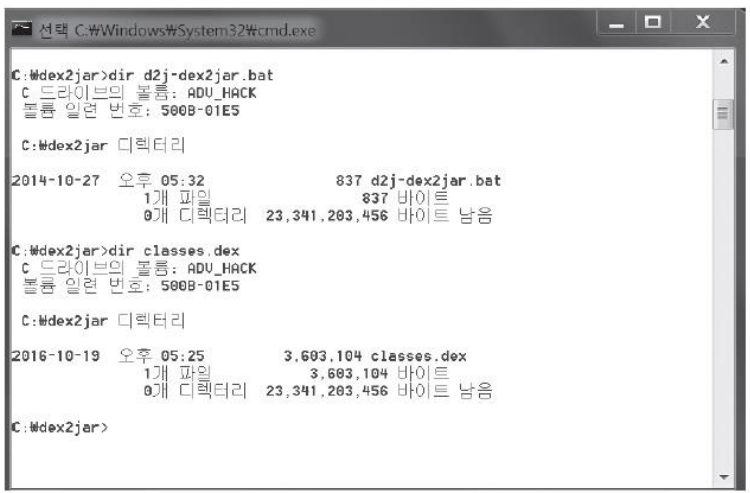
5 dex 파일에서 jar 파일 추출(dex → dex2jar → jar)

- dex2jar-2.0.zip 파일을 내려받아 zip 파일의 압축을 풀고 해당 폴더를 임의의 폴더에 복사
- '압축 풀기로 dex 파일 추출'에서 얻은 classes.dex 파일을 해당 폴더에 복사

1 안드로이드 앱 정적 분석하기

5 dex 파일에서 jar 파일 추출(dex → dex2jar → jar)

[classes.dex 파일과 d2j-dex2jar.bat 파일 확인]



※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

1 안드로이드 앱 정적 분석하기

5 dex 파일에서 jar 파일 추출(dex → dex2jar → jar)

- classes.dex 파일을 풀면 같은 폴더에 classes-dex2jar.jar 파일이 생성

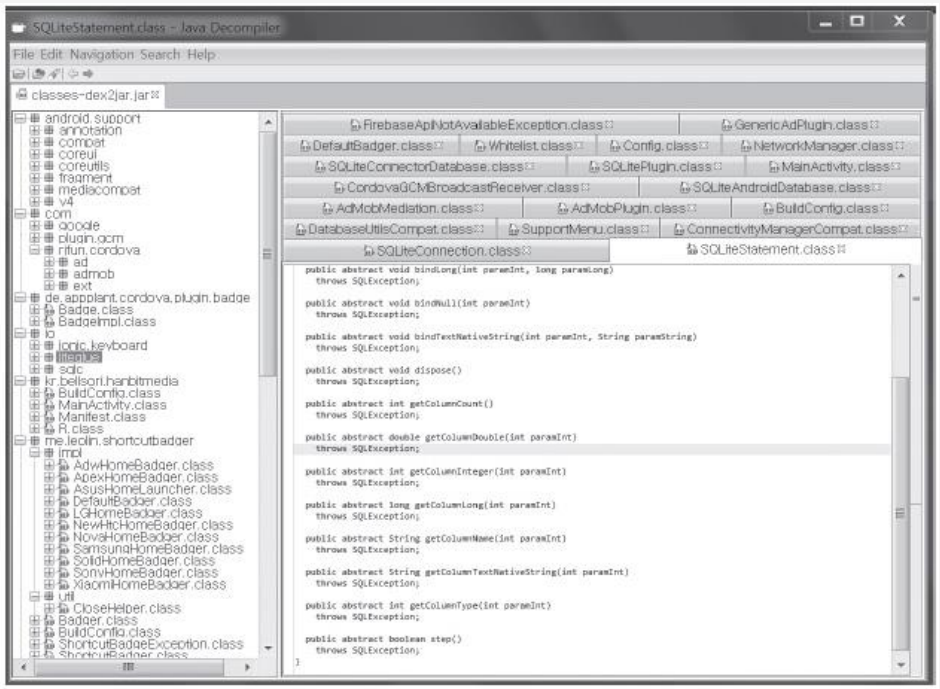
```
C:₩dex2jar>d2j-dex2jar.bat classes.dex
dex2jar classes.dex -> .₩classes-dex2jar.jar
```

- jd-gui와 같은 프로그램을 이용하면 내용을 살펴볼 수 있음(jar → jd-gui)
- 다운로드 : http://jd.benow.ca/

1 안드로이드 앱 정적 분석하기

5 dex 파일에서 jar 파일 추출

- 압축을 풀고 jdgui.exe 파일을 실행하여 classes-dex2jar.jar 파일을 불러오면 디컴파일 된 class 파일을 살펴볼 수 있음
- (jd-gui → jar)



[jd-gui 프로그램으로 jar파일을 열람한 화면]

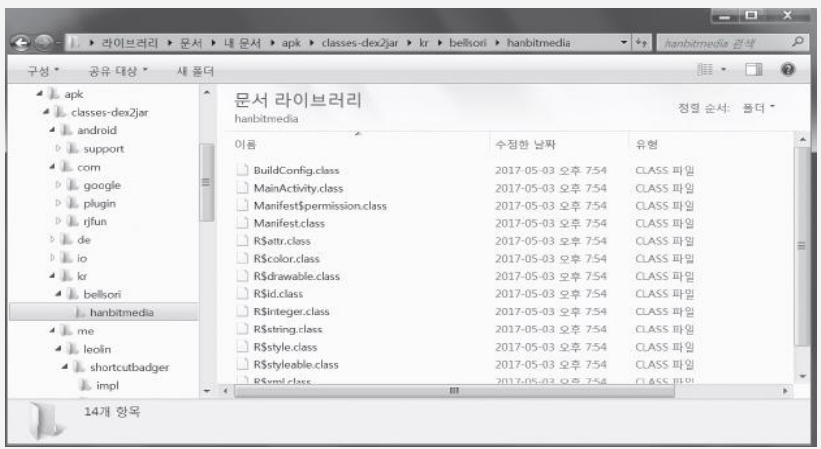
※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

1 안드로이드 앱 정적 분석하기

6 jar 파일 압축 풀기로 class 파일 만들기 (jar → 7-zip → class)

- 7-zip으로 jar 파일을 풀면 class 파일이 생성

[jar 파일의 압축을 풀어 살펴본 class]



※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

1 안드로이드 앱 정적 분석하기

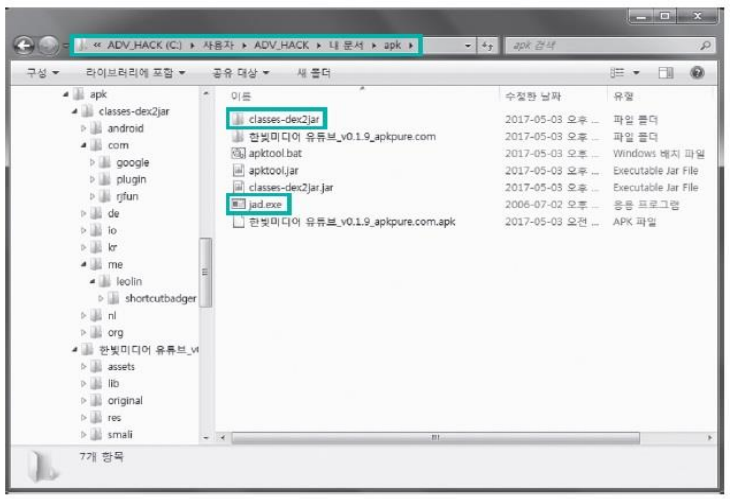
7 class 파일을 디컴파일하여 java 파일 만들기
(class → jad → java)

- class 파일을 원래 소스코드인 java 파일로 바꾸기 위해 jad를 사용
- 다운로드 : http://www.javadecompilers.com/jad
- 내려받은 jad.exe 파일을 class 파일이 있는 상위 디렉터리로 복사

1 안드로이드 앱 정적 분석하기

7 class 파일을 디컴파일하여 java 파일 만들기 (class → jad → java)

[class 폴더가 있는 경로로 jad.exe 파일복사]



※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

1 안드로이드 앱 정적 분석하기

7 class 파일을 디컴파일하여 java 파일 만들기 (class → jad → java)

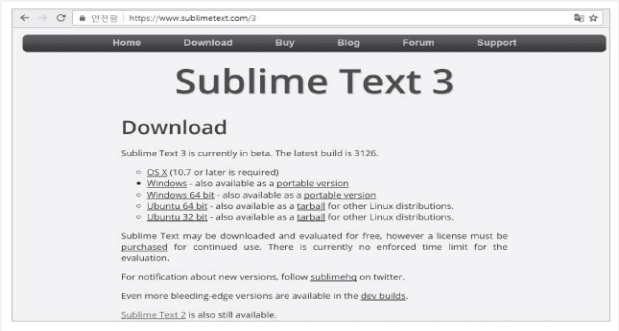
- 윈도우 커맨드 창에 다음과 같이 입력하면 class 파일이 java 파일로 디컴파일

```
jad -r -sjava -d classes-dex2jar **/*.class
```

1 안드로이드 앱 정적 분석하기

7 class 파일을 디컴파일하여 java 파일 만들기

- Sublime Text 프로그램으로 보면 보다 정교하게 디컴파일된 java 코드 확인 (sublime → java)
- 다운로드 : https://www.sublimetext.com/3



[Sublime Text 다운로드 페이지]

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

2 '안전하지 않은 데이터 저장'에 대한 대응 방안

- iOS 환경에서의 대응 방안
  - 모바일 기기의 파일 시스템에 중요한 개인 식별 정보를 저장하지 않음(개인 식별 정보)
  - 불가피하게 사용자 인증 정보를 제공해야 한다면 반드시 표준 웹 또는 API 로그인 스키마(HTTPS 등)를 사용(API : 응용 프로그램에서 사용할 수 있도록, 운영 체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스를 뜻함. 주로 파일 제어, 창 제어, 화상 처리, 문자 제어 등을 위한 인터페이스를 제공함)(사용자 인증 정보, https)

2 '안전하지 않은 데이터 저장'에 대한 대응 방안

- iOS 환경에서의 대응 방안
  - 정보를 저장하는 곳이나 캐시는 CommonCrypto와 같은 표준 iOS 암호화 라이브러리 사용을 고려
  - 데이터가 적다면 애플에서 제공하는 keychain API 사용을 권장
  - 데이터베이스에 대해서는 SQLite 데이터를 보호하기 위해 SQLcipher 사용을 권장

2 '안전하지 않은 데이터 저장'에 대한 대응 방안

- iOS 환경에서의 대응 방안
  - 민감한 데이터를 plist 파일에 저장할 때는 가급적 NSUserDefaults를 사용하지 않음
  - NSManagedObjects를 이용한 모든 데이터/엔티티는 보호되지 않는 데이터베이스 파일에 저장된다는 것을 인지해야 함

2 '안전하지 않은 데이터 저장'에 대한 대응 방안

- 안드로이드 환경에서의 대응 방안
  - 로컬 저장소에 대한 엔터프라이즈 안드로이드 장치 관리 API는 'setStorageEncryption'을 통해 로컬 파일 저장소를 강제로 암호화하는 데 사용할 수 있음
  - SD 카드 저장소에 대한 보안은 'javax.crypto' 라이브러리를 통해 부분적으로 해결할 수 있음 (가장 쉬운 것은 마스터 패스워드와 AES128로 평문 데이터를 암호화하는 것) (AES)

2 ‘안전하지 않은 데이터 저장’에 대한 대응 방안

- 안드로이드 환경에서의 대응 방안
  - 응용 프로그램 간의 정보 공유를 위해 반드시 필요한 경우가 아니면 모든 공유 환경 설정에 MODE_WORLD_READABLE이 되지 않도록 함

3 '충분하지 않은 전송 계층 보호'에 대한 대응 방안

- 일반적인 고려 사항
  - 네트워크 계층은 안전하지 않기 때문에 잠재적으로 도청이 가능하다는 사실을 항상 염두에 두어야 함(스니핑)
  - 민감한 정보, 세션 토큰, 그 외 중요한 데이터를 백 엔드 API 또는 웹 서비스와 주고받을 때는 모든 전송 채널에 대해 SSL/TLS를 사용 함
  - 산업 표준 암호 알고리즘을 사용하고, 키의 길이는 짧지 않게 적절히 설정 함(전사 공격)

3 '충분하지 않은 전송 계층 보호'에 대한 대응 방안

- 일반적인 고려 사항
  - 신뢰할 수 있는 CA(인증 기관) 제공자가 서명한 인증서를 사용 함
  - 자체 서명한 인증서는 절대 받아들이지 않음
  - SSL chain 검증을 무시하거나 비활성화시키지 않음
  - 유효하지 않은 인증서가 탐지되면 경고를 보냄

3 '충분하지 않은 전송 계층 보호'에 대한 대응 방안

- iOS 환경에서의 대응 방안
  - CFNetwork를 사용할 때 신뢰할 수 있는 클라이언트 인증서를 지정하기 위해 안전한 전송 API를 사용하는지 확인 함
  - 대부분의 경우 NSStreamSocketSecurityLevelSSLv3 또는 NSStreamSocketSecurityLevelTLSv1은 매우 높은 암호화 표준 길이를 사용함

3 '충분하지 않은 전송 계층 보호'에 대한 대응 방안

- iOS 환경에서의 대응 방안
  - 개발 후 NSURL 콜(또는 NSURL의 래핑)은 자체 서명된 것이나 NSURL 클래스 메소드인 setAllowsAnyHTTPSCertification과 같은 유효하지 않은 인증서를 받아들여서는 안됨

3 '충분하지 않은 전송 계층 보호'에 대한 대응 방안

- 안드로이드 환경에서의 대응 방안
  - 애플리케이션에서 org.apache.http.conn.ssl.AllowAllHostnameVerifier 또는
  - SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER와 같은 모든 인증서를 받아들이지 않도록 개발 사이클이 끝나면 해당 부분을 모두 제거해야 함

2 모바일 통제와 설계 원칙

1 모바일 장치에 있는 민감한 데이터를 식별하고 보호

- 설계 단계에서 데이터의 중요도와 통제 적용을 고려하여 데이터 저장 방식을 분류 함(예를 들어 패스워드, 개인 데이터, 위치 정보, 에러 로그 등)

- 데이터의 처리와 저장, 사용은 이 분류 기준에 따라 수행되어야 함, 또한 API 함수에 대한 보안 검증은 민감한 데이터에 적용되어야 함(시큐어 코딩)

1 모바일 장치에 있는 민감한 데이터를 식별하고 보호

- 민감하고 중요한 데이터는
  클라이언트 측 장치가 아닌 서버에 저장함(서버)

  이는 안전한 네트워크 연결이 보장되고,
  서버 측에 적용된 보호 메커니즘이 높은 수준이라는
  것을 전제로 함

- 장치에 데이터를 저장할 때
  OS에서 제공하는 파일 암호화 API를 사용 함

- 애플리케이션을 개발할 때 데이터를 최초 생성할
  때부터 저장, 이용, 삭제할 때까지 유출되지 않도록
  안전한 방법을 고려 함(개발)

2 장치에 있는 패스워드 식별 정보를 안전하게 처리

- 패스워드 대신 장치에 안전하게 저장할 수 있는 긴 문자열의 인증 토큰 사용을 고려 함(인증 토큰)
  해당 토큰이 전송될 때 SSL/TLS를 이용하여 안전하게 교환되도록 함

- 패스워드를 장치에 저장해야 할 경우 모바일 OS에서 제공하는 암호화와 키 저장 메커니즘을 이용 함(암호화)

- 모든 패스워드의 복잡도를 테스트 함(다양, 길이)

2 장치에 있는 패스워드 식별 정보를 안전하게 처리

- 패스워드와 키가 캐시 또는 로그에 보이지 않도록 함(캐시/로그)

- 패스워드 또는 비밀 정보가 애플리케이션 바이너리에 포함되지 않도록 함(바이너리)

3 전송 시 민감한 데이터가 보호되는지 확인

- 전송되는 네트워크가 안전하지 않다고 가정 함
- 애플리케이션에서 네트워크를 통해 민감한 데이터를 전송할 때 SSL/TLS와 같은 end-to-end 시큐어 채널을 사용 함(4계층 보안)
- AES와 같은 검증된 암호화 알고리즘을 사용하고 적절한 키 길이를 적용 함(대칭키)
- 신뢰된 CA 제공자에 의해 서명된 인증서를 사용함, 자체 서명된 인증서는 허용하지 않음(인증 기관)

3 전송 시 민감한 데이터가 보호되는지 확인

- 민감한 데이터에 대한 Man-in-the-middle 공격 (SSL proxy, SSL strip)의 위험을 줄이기 위해 원격 엔드-포인트(서버) 식별을 검증한 후 안전한 채널을 연결 함(MITM)

- 민감한 데이터를 전달할 때는 SMS, MMS 또는 공지 문자 등을 사용하지 않음

4 SSL/TLS

- 전송 계층 보안 (영어 : Transport Layer Security, TLS, 과거 명칭 : 보안 소켓 레이어/Secure Sockets Layer, SSL)는 암호 규약 임

  그리고 '트랜스포트 레이어 보안'이라는 이름은 '보안 소켓 레이어'가 표준화 되면서 바뀐 이름 임 (4계층 보안)

- 이 규약은 인터넷 같이 TCP/IP 네트워크를 사용하는 통신에 적용되며, 통신 과정에서 전송계층 종단간 보안과 데이터 무결성을 확보해 줌(무결성)

4 SSL/TLS

- 이 규약은 웹 브라우징, 전자 메일, 인스턴트 메신저, voice-over-IP (VoIP) 같은 응용 부분에 적용되고 있음

- 국제 인터넷 표준화 기구(IETF)에 의해 현재 구식(Deprecate)으로 간주되어 있음

  최종 갱신은 RFC 5246이고, 최종 갱신 버전은 넷스케이프에서 만든 SSL 표준을 바탕으로 했음

5 AES

- 고급 암호화 표준(Advanced Encryption Standard, AES)은 2001년 미국 표준 기술 연구소(NIST)에 의해 제정된 암호화 방식
- AES는 두 명의 벨기에 암호학자인 존 대먼과 빈센트 라이먼에 의해 개발된 Rijndael(레인달, [rɛinda ː l]) 에 기반하며 AES 공모전에서 선정 됨
- AES는 미국 정부가 채택한 이후 전 세계적으로 널리 사용되고 있음, 1977년 공표된 DES를 대체한 AES는, 암호화와 복호화 과정에서 동일한 키를 사용하는 대칭키 알고리즘임(대칭키 vs. 공개키)

5 AES

- 미국 표준 기술 연구소(NIST)는 2001년 11월 26일 AES를 미국 연방 정보 처리 표준(FIPS-197)으로 공포함, NIST는 5년의 표준화 과정을 거쳤으며 이 과정에서 15개의 알고리즘이 경쟁, Rijndael 암호가 가장 적합한 알고리즘으로 선정되었음

5 AES

- 이 표준은 2002년 5월 26일부터 효력을 발휘하기 시작, AES는 ISO/IEC 18033-3 표준에 포함되어 있으며 여러 암호화 패키지에서 사용되고 있음, AES는 또한 미 국가 안보국에 의해 1급 비밀(Top Secret)에 사용할 수 있도록 승인된 알고리즘 중 최초로 공개되어 있는 알고리즘임

5 AES

- Rijndael은 알고리즘의 개발자인 빈슨트 레이믄(Vincent Rijmen)과 요안 대믄(Joan Daemen)의 이름을 따서 지은 것으로 AES 표준은 여러 Rijndael 알고리즘 중 블록 크기가 128비트인 알고리즘을 말함(블록 암호)

6 CA

- 암호학에서 인증 기관(Certificate Authority, CA)은 다른 곳에서 사용하기 위한 디지털 인증서를 발급하는 하나의 단위 임, 인증 기관은 많은 공개 키 기반구조(PKI, public key infrastructure)에 설명되어 있음(공인인증서)

6 CA

- 이러한 서비스로 요금을 부과하는 상업 목적의 인증 기관들이 많이 있으며, 공익 단체나 정부들도 저만의 인증 기관을 가지고 있으며 무료 인증 기관들도 있음
  기업이나 단체별로 운영하는 인증 기관도 있는데, 이 경우 사설 인증기관으로 분류하기도 함(공인인증 vs. 사설인증)

7 중간자 공격 방법

- Dsniff : SSH와 SSL 중간자공격을 위한 툴
- Cain and Abel
  : 윈도 GUI기반 도구로 sniffing and ARP poisoning과 함께 MITM 공격에 쓰임
- Ettercap : LAN기반 중간자공격을 위한 툴
- Karma
  : 802.11 Evil Twin 공격을 사용하여 중간자 공격을 하는 툴(이블 트윈)

7 중간자 공격 방법

- Air-Jack : 802.11기반으로 중간자공격을 하는 툴
- SSL Strip : SSL기반 중간자공격툴
- SSL Sniff
  : SSl 기반 중간자공격툴, 원래는 인터넷
  익스플로러의 결함을 찾기 위해 만들어짐

7 중간자 공격 방법

- Intercepter-NG
  : ARP poisoning 기능으로
  윈도 네트워크 비밀번호를 알아내는 툴,
  SSL Strip과 SSL기반 중간자공격을 포함 함

- Mallory
  : TCP와 UDP MiTMing proxy,
  MiTM SSL, SSH, 다른 프로토콜로 확장사용가능 함

- Wsniff
  : 802.11 HTTP/HTTPS 기반 중간자공격을 위한 툴

8 사용자 인증, 권한, 세션 관리를 올바르게 구현

- 애플리케이션에 적절한 길이의 사용자 인증값을 요청
- 초기 인증 이후 세션 관리는 안전한 프로토콜을 통해 처리
- 매우 복잡하고 예측하기 어려운 세션 식별자를 사용
- 인증할 때 IP 주소와 같은 추가 보안 요소를 사용

9 백 엔드 API(서비스)와 플랫폼(서버)을 안전하게 유지

- 모바일 장치와 웹 서버 백 엔드 또는 다른 외부 인터페이스 간에 민감한 데이터가 의도치 않은 상태로 전송되는 경우가 있는지 코드를 면밀히 확인
- 모바일 앱에 대한 모든 백 엔드 서비스 (REST/Web 서비스)는 주기적으로 취약점을 검토
- OS, 웹 서버, 다른 애플리케이션 컴포넌트 등에 적용된 최신 보안 패치가 백 엔드 플랫폼(서버)에 적용되어 있는지 확인
- 사용자/IP별 Rate 최솟값을 적용하여 DDoS 공격의 위험을 줄임(syn 패킷의 ratio)

10 서드 파티 서비스와 애플리케이션의 데이터 통합을 안전하게 수행

- 모바일 애플리케이션에서 사용되는 서드 파티 코드/라이브러리에 대한 보안과 인증이 안전하게 되어 있는지 확인
- 보안 패치를 위해 모바일 애플리케이션에서 사용되고 있는 서드 파티 프레임워크/API를 조사
- 신뢰할 수 없는 서드 파티 앱으로부터 전달받는 모든 데이터를 유심히 살펴 봄

11 사용자 데이터의 수집과 사용 동의를 구할 때는 신중히 대처

- 개인 정보를 보호할 수 있는 정책을 수립하고, 개인 정보 및 데이터를 사용할 경우 사용자의 동의를 받는 채널을 만듦
- 앱을 만들 때 사용자 개인 정보를 수집하는지 확인
- 의도치 않은 노출을 확인하기 위해 커뮤니케이션 메커니즘을 살펴 봄
- 개인 식별 정보 전송에 대한 동의 기록을 보관

12 금융 지불 자원에 불법으로 접근하지 못하게 통제

- 금융 지불 자원에 접근한 로그를 모두 기록
- 금융 지불 자원에 익명으로 접근이 가능한지,
  재인증을 실행하는지 검사
- 금융 지불 자원 주소에 대해
  기본적으로 화이트리스트 모델을 사용(White list)
- 금융 지불 자원에 대한 모든 API 호출을 인증
- wallet API 콜백에서 계정/금액/청구/항목 정보를
  평문으로 전송하지 않음(암호화)

13 모바일 애플리케이션을 안전하게 배포

- 애플리케이션은 보안 패치를 적용시키도록 설계하고, 앱 스토어에서 승인되기 위한 모든 요건을 충족해야 함 (보안 패치/요건 충족)
- 앱 스토어는 안전하지 않은 코드를 포함한 앱을 지속적으로 모니터링하고, 사고가 발생하면 즉시 공지하고 원격에서 바로 삭제함, 공식적인 앱 스토어를 통해 배포되는 앱은 취약점이 발생할 경우를 대비해 안전장치를 제공함(원격 삭제)
- 앱에서 보안 문제가 발견되었을 때 보고할 수 있는 피드백 채널을 제공

14 에러에 대한 실시간 코드 해석기(인터프리터)를 면밀히 확인

- 실시간 코드 해석기를 최소화하고
  필요할 때는 최소한의 권한으로 실행 함(인터프리터)
- 실시간 코드 해석기의 퍼지 테스트를 함(퍼징)
- 샌드박스로 실시간 코드 해석기를 보호(샌드박스)

14 에러에 대한 실시간 코드 해석기(인터프리터)를 면밀히 확인

- 안전한 앱 코딩 방법
  - 정상적인 방법뿐만 아니라
    비정상적인 방법으로도 앱을 테스트
  - 모든 입력값을 검증
  - 프로그래밍 코드를 최소화
  - 안전한 언어를 사용
    (예를 들어 버퍼 오버플로우가 안 되는 함수 사용 등)
    (버퍼 오버플로우)
  - 보안 결함을 찾기 위해 정적 분석과 동적 분석을
    수행하고 퍼지 테스트를 함 (정적 vs. 동적)

14 에러에 대한 실시간 코드 해석기(인터프리터)를 면밀히 확인

- 안전한 앱 코딩 방법
  - 버퍼, 정수 오버플로우를 피하기 위해 안전한 문자열 함수를 사용
  - 애플리케이션이 요구하는 최소한의 권한으로 앱을 구동 (최소 권한)
  - API에서 기본적으로 부여하는 권한을 인지하고 필요하지 않다면 제거
  - 루트나 시스템 관리자 권한으로 코드나 앱을 실행하지 않음 (관리자 권한)

14 에러에 대한 실시간 코드 해석기(인터프리터)를 면밀히 확인

- 안전한 앱 코딩 방법
  - 권한을 가진 사용자뿐만 아니라 표준 권한으로도 테스트를 수행
  - OS에서 제공하는 통신 메커니즘을 사용
  - 앱을 배포하기 전에 테스트한 모든 코드를 제거 (테스트 코드 제거)
  - 로깅이 적절하게 기록되는지 확인함, 단, 민감한 정보까지 불필요하게 로그에 보관하지 않도록 함 (로그 주의)

15 인터프리터

- 인터프리터(Interpreter, 문화어 : 해석기)는 프로그래밍 언어의 소스 코드를 바로 실행하는 컴퓨터 프로그램 또는 환경을 말함 (컴파일러, 하이브리드)

15 인터프리터

- 원시 코드를 기계어로 번역하는 컴파일러와 대비
  인터프리터는 다음의 과정 가운데 적어도 한 가지
  기능을 가진 프로그램 임(컴파일러 vs. 인터프리터)
  - 소스 코드를 직접 실행
  - 소스 코드를 효율적인 다른 중간 코드로 변환하고,
    변환한 것을 바로 실행
  - 인터프리터 시스템의 일부인 컴파일러가 만든,
    미리 컴파일 된 저장 코드의 실행을 호출

15 인터프리터

- 인터프리터는 고급 언어로 작성된 원시코드 명령어들을 한번에 한 줄씩 읽어들여서 실행하는 프로그램
- 고급언어로 작성된 프로그램들을 실행하는 데에는 두 가지 방법이 있음, 가장 일반적인 방법은 프로그램을 컴파일 하는 것이고, 다른 하나는 프로그램을 인터프리터에 통과시키는 방법
- 고급 명령어들을 중간 형태로 번역한 다음 그것을 실행 함, 이와는 대조적으로, 컴파일러는 고급 명령어들을 직접 기계어로 번역 함

15 인터프리터

- 컴파일 된 프로그램들은 일반적으로 인터프리터를 이용해 실행시키는 것보다 더 빠르게 실행 됨(컴파일러)

- 그러나 인터프리터의 장점은 기계어 명령어들이 만들어지는 컴파일 단계를 거칠 필요가 없다는데 있음, 컴파일 과정은 만약 원시 프로그램의 크기가 크다면 상당한 시간이 걸릴 수 있음

15 인터프리터

- 이와는 달리 인터프리터는 고급 프로그램을 즉시 실행시킬 수 있음, 이런 이유 때문에 인터프리터는 종종 프로그램의 개발단계에서 사용되는데, 그것은 프로그래머가 한번에 적은 양의 내용을 추가하고 그것을 빠르게 테스트 해보길 원하기 때문 임

- 이 외에도 인터프리터를 이용하면 프로그래밍을 대화식으로 할 수 있기 때문에, 학생들의 교육용으로 사용되는 경우도 많음(파이썬)

15 인터프리터

- 인터프리터와 컴파일러는 둘 다 대부분의 고급언어에 적용이 가능하지만 BASIC 이나 LISP(인공지능)과 같은 일부 언어들은 개발 당시에는 특별히 인터프리터에 의해서만 실행되도록 설계되었음

  그 외에도 포스트스크립트과 같은 페이지 기술 언어 들도 인터프리터를 사용 함

  모든 포스트스크립트 프린터는 포스트스크립트 명령문을 실행할 수 있도록 인터프리터가 내장되어 있음

16 샌드박스

- 샌드박스(Sandbox)는 보호가 필요한 어린아이들을 위해 모래통에서만 놀도록 하는 데서 유래한 소프트웨어 보안 개발기법임

- 예를 들면, 악성 바이러스나 악성코드의 경우, 이들의 공격행위를 테스트할 때, 실제 운영체계나 또는 파일 또는 이러한 시스템에 추가적인 악영향을 주거나 이를 감염시킬 수 없도록 하는 차단된 환경이 필요하게 됨(차단 환경)

16 샌드박스

- 이처럼 테스트를 위해 외부로의 연결점을 차단하거나 외부로부터의 접근 및 영향을 필터링 또는 차단할 수 있는 통제된 환경 내에서 프로그램을 동작시키는 것을 가리킴(통제 환경, 허니팟)