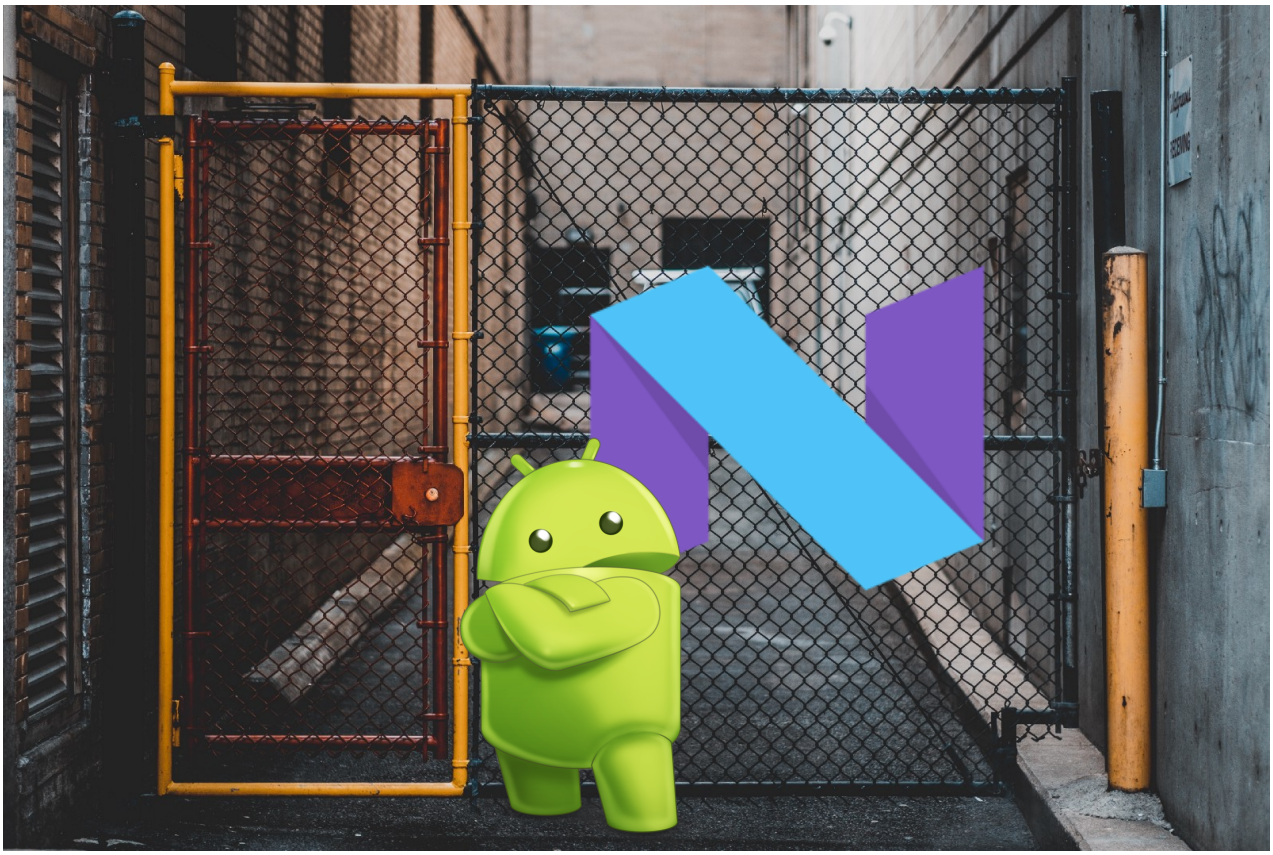
#10

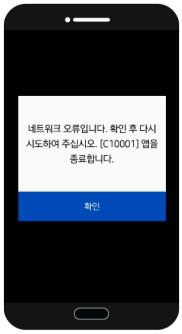
Make user certificates available on Android 7.0



2016년 8월 안드로이드 7.0 누가(Nougat) 버전이 배포되었다. 새로운 메모리 보호 기능이 추가되었고, APK 서명 스킴(Scheme) v2가 추가되었다. 네트워크 트래픽 보안을 위해 사용자가 직접 설치한 인증서는 더 이상 신뢰하지 않는다. API 레벨이 24 이상인 애플리케이션의 경우 해당된다.

Android 7.0 Nougat 부터 사용자 인증서를 통한 HTTPS 통신이 더이상 허용되지 않는다.

앱 보안을 위한 개선사항으로 볼 수 있겠지만, 모바일 앱 취약점 점검을 위해 Burp Suite 같은 프록시 도구를 사용하는 상황에선 문제가 된다. HTTPS 통신을 분석하기 위해 Burp Suite의 인증서(사용자 인증서)를 휴대폰에 설치하여 사용해야 하는데, 사용할 수 없기 때문이다.



에스원 모바일 방문자 앱의 사용자 인증서 통신 에러

지난 3월 에스원에서 만든 모바일 방문객 앱의 경우에도 이와 같은 문제로 최신 안드로이드 휴대폰(버전 7.0 이상)에서는 통신 내용을 살펴볼 수 없었다. 휴대폰을 루팅(rooting)한 경우 여러가지 방법으로 이 문제를 해결할 수 있겠지만, 루팅이 어려운 상태에서 이 문제를 어떻게 해결할 수 있는지 알아보자.

Network Config

안드로이드 API 24 이상 버전부터 기본 네트워크 설정이 변경되었다.

앱이 동작할 때 사용자(user) 인증서는 더 이상 신뢰하는 대상에 포함되어 있지 않다. 사용자 인증서를 신뢰할 수 있도록 만들어야 한다. 기본적인 아이디어는 API 23 버전까지 사용하던 네트워크 설정을 가져오는 것이다. 안드로이드 개발자 사이트를 참고하여 앱을 변조해보자.

Decompile

apktool을 이용해 대상 APK파일을 디컴파일한다. AndroidManifest.xml과 Smali파일들을 얻을 수 있다.

APKTOOL.BAT D <APK FILE> -O <DIRECTORY OF DECOMPILED FILES>

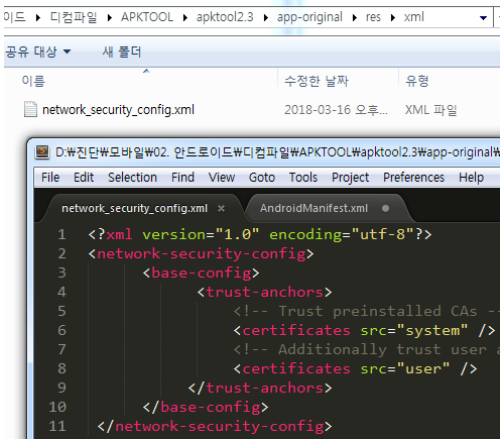
Modify AndroidManifest.xml

앱이 네트워크 설정 정보를 참조할 수 있도록 networkSecurityConfig 속성 설정

```
<uses-permission android:name="android.permission.INTERNET"/>
<application android:allowBackup="true" android:debuggable="true" android:icon="
@mipmap/ic_launcher" android:label="@string/app_name" android:roundIcon="
@mipmap/ic_launcher_round" android:supportsRtl="true" android:theme="@style/AppTheme"
android:networkSecurityConfig="@xml/network_security_config">
    <activity android:name="com.example.jbari.test2.MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
```

Add res/xml/network_security_config.xml

AndroidManifest.xml에 **@xml/network_security_config**로 값을 지정했다. xml 폴더의 network_security_config.xml 파일을 참조하라는 의미다. 따라서 이 네트워크 설정 파일을 앱이 참조할 수 있도록 xml파일을 오른쪽 그림과 같이 생성해준다.



AndroidManifest 파일에 직접 설정 파일명을 적어주지 않고 annotation으로 설정하는 이유는 앱이 빌드되는 과정에서, R.smali 파일이 annotation을 통해 필요한 리소스를 매핑하기 때문이다.

Put R$xml into R.smali

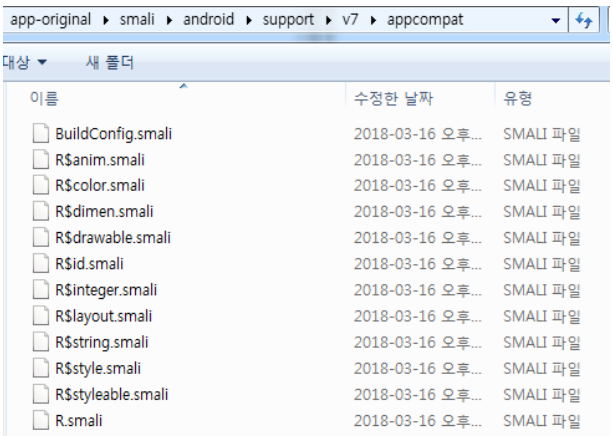
앱이 참조하는 리소스 목록은 다음과 같이 R.smali 파일에 annotations로 정의되어 있다. 여기에 앞서 생성한 network_security_config.xml 파일과 이 파일이 들어있는 xml 폴더를 포함시키기 위해 annotations 에 R$xml을 추가해준다.

```
# annotations
.annotation system Ldalvik/annotation/MemberClasses;
    value = {
        Landroid/support/v7/appcompat/R$styleable;,
        Landroid/support/v7/appcompat/R$style;,
        Landroid/support/v7/appcompat/R$string;,
        Landroid/support/v7/appcompat/R$layout;,
        Landroid/support/v7/appcompat/R$integer;,
        Landroid/support/v7/appcompat/R$id;,
        Landroid/support/v7/appcompat/R$drawable;,
        Landroid/support/v7/appcompat/R$dimen;,
        Landroid/support/v7/appcompat/R$color;,
        Landroid/support/v7/appcompat/R$bool;,
        Landroid/support/v7/appcompat/R$attr;,
        Landroid/support/v7/appcompat/R$anim;
        Landroid/support/v7/appcompat/R$xml;
    }
.end annotation
```

R.smali는 R$로 시작하는 smali 파일들의 부모 파일이다. R$xml은 R$xml.smali 파일을 참조하겠다는 선언으로 이해하면 된다.

Create R$xml.smali

이제 R.smali 파일은 R$xml.smali을 참조할 수 있다.하지만 R$xml.smali 파일은 아직 존재하지 않는다. 따라서 직접 만들어줘야 하는데, 파일을 쉽게 만들기 위해 다른 임의의 파일(R$*.xml)을 아래 경로에 복사해 만들어준다.*(아래 예시에서는 **R$anim.xml**을 복사)* 그리고 복사한 파일을 R$xml.smali로 정의하기 위해 아래 붉은색 상자처럼 내용을 바꾸어주면 된다.




```
.class public final Landroid/support/v7/appcompat/R$anim;
.super Ljava/lang/Object;
.source "R.java"                                    -> R$xml


# annotations
.annotation system Ldalvik/annotation/EnclosingClass;
    value = Landroid/support/v7/appcompat/R;
.end annotation

.annotation system Ldalvik/annotation/InnerClass;
    accessFlags = 0x19
    name = "anim"  -> name="xml"
.end annotation


# static fields
.field public static final abc_fade_in:I = 0x7f010000

.field public static final abc_fade_out:I = 0x7f010001

.field public static final abc_grow_fade_in_from_bottom:I = 0x7f010002

.field public static final abc_popup_enter:I = 0x7f010003
```

Modify R$xml.smali

#static fields에 network_security_config의 주소를 할당해준다. network_security_config라는 리소스가 0x7f0d0000 위치에 할당된다는 의미다. 이 주소는 다른 smali 파일을 참고해 사용하지 않는 주소로 정해준다.

```
# static fields
.field public static final network_security_config:I = 0x7f0d0000
```

Rebuild APK and Install

apktool을 이용해서 재빌드하고 sign 과정을 거치면 완료된다.(설명 생략) 새롭게 빌드한 앱(APK)을 설치하고 Burp suite 같은 프록시 도구를 이용해 네트워크 트래픽을 확인해보면 기존과 같이 잘 분석이 되는 것을 알 수 있다.

Obfuscation, 난독화

모바일 앱의 변조나 분석을 막기 위해 난독화 도구를 사용하는 앱이 늘고 있다. 난독화 솔루션이 적용된 모바일 앱의 경우 apktool을 이용한 디컴파일(decompile) 시 분석이나 변조가 불가능하게 보일 수 있다. 따라서 앞에 소개한 내용을 그대로 적용하기 어려울 수 있다. 현재 삼성에서 가장 많이 사용하고 있는 방식으로 난독화된 앱(e.g. 에스원 모바일 방문객 앱)을 분석해보면 R.smali 파일을 찾을 수 없다. 이 경우 다음과 같이 진행하면 된다. 앞의 내용과 다른 부분만 설명한다.

Put R$xml into App.smali

난독화 솔루션이 적용된 앱은 R.smali 파일 대신 App.smali 파일이 존재한다. 이름만 바뀌었을 뿐 내용은 같기 때문에 앞에서 했던 방법과 동일하게 R$xml을 annotations에 추가한다. (xml 행을 추가하면서 앞의 항목인 MsgHandler 뒤에 쉼표(,)를 추가하는 것을 잊으면 안된다.)

```
# annotations
.annotation system Ldalvik/annotation/MemberClasses;
    value = {
        Lweb/apache/sax/app$MsgHandler;,
        Lweb/apache/sax/app$xml;
    }
.end annotation
```

Modify R$xml.smali

#static fields에 network_security_config의 주소를 할당 해준다. 다른 점은 별도의 주소를 할당해주지 않아도 된다는 점이다.

```
# static fields
.field public static final network_security_config:I
```

이 내용에서 강조하고자 하는 점은 별도의 루팅(rooting) 없이 안드로이드 API 24 버전의 인증서 관련 보안 기능을 우회할 수 있다는 점이다. 안드로이드 운영체제의 루팅이 어려워지고 있는 시점에 모바일 앱 보안점검을 위해 필요한 기술이라 할 수 있다.