JustTrustMe 原理分析

——挽秋

1、JustTrustMe 简介

JustTrustMe 一个用来禁用、绕过 SSL 证书检查的基于 Xposed 模块。
项目地址：https://github.com/Fuzion24/JustTrustMe

2、实现原理分析

JustTrustMe 是将 APK 中所有用于校验 SSL 证书的 API 都进行了 Hook，从而绕过证书检查的，所以弄请原理之前，先得弄清楚 Android 上实现 Https 通信有哪几种方式。

2.1Android 上实现 Https 的几种方式

2.1.1 通过 OkHttp 来实现

OkHttp 是一个第三方库，OkHttp 中进行 SSL 证书校验，有如下两种方式：
1）CertificatePinner(证书锁定)：
通过 CertificatePinner 进行连接的 OkHttp，在连接之前，会调用其 check 方法进行证书校验。
实现代码如下：

```
public  OkHttpClient getUnsafeOkHttpClient() {
    OkHttpClient client = new OkHttpClient.Builder().certificatePinner( new CertificatePinner.Builder()
            .add("publicobject.com", "sha1/DmxUShsZuNiqPQsX2Oi9uv2sCnw=")
            .add("publicobject.com", "sha1/SXxoaOSEzPC6BgGmxAt/EAcsajw=")
            .add("publicobject.com", "sha1/blhOM3W9V/bVQhsWAcLYwPU6n24=")
            .add("publicobject.com", "sha1/T5x9IXmcrQ7YuQxXnxoCmeeQ84c=")
            .build()).build();
    return client;
}
```

2）自定义证书和 HostnameVerify 来实现 Https 校验：
Okhttp 中如果不指定 HostnameVerifier 默认调用的是 OkHostnameVerifier.verify 进行服务器主机名校验；如果设置了 HostnameVerifier，则默认调用的是自定义的 verify 方法。
实现代码如下：

```
        public OkHttpClient getCustomTrustedCertificatesOkHttpClient(Context context){
 OkHttpClient client = null;
     SSLContext sslContext = sslContextForTrustedCertificates(context);
 //对于其他不是自定义证书的网站，可以通过自定义 HostnameVerify 来实现校验策略
client = new OkHttpClient.Builder()
```

```
                    .sslSocketFactory(sslContext.getSocketFactory()).hostnameVerifier(new
MyHostnameVerifier()).build();
      return client;
}
public SSLContext sslContextForTrustedCertificates(Context context){
      InputStream in = null;
      try{
            context.getAssets().open("app_pay.cer"); //自定义的证书放到项目中的 assets 目录中
      }catch (Exception e){
            e.printStackTrace();
      }
      return null;
}
private class MyHostnameVerifier implements HostnameVerifier {

      public boolean verify(String hostname, SSLSession session) {
            return true;//在这里进行主机名校验，此处未实现
      }
}
```

绕过上述 SSL 证书验证，Xposed 需要 Hook 的方法名和类名如下表所示：

类名	方法名
com.squareup.okhttp.CertificatePinner	public void check(String hostname, List<Certificate> peerCertificates) throws SSLPeerUnverifiedException{}
com.squareup.okhttp.CertificatePinner	public void check(String,List)
okhttp3.internal.tls.OkHostnameVerifier	public boolean verify(String, SSLSession)
okhttp3.internal.tls.OkHostnameVerifier	public boolean verify(String, X509Certificate)
okhttp3.OkHttpClient.Builder	public OkHttpClient.Builder hostnameVerifier(HostnameVerifier hostnameVerifier)

JustTrustME 中的代码并没有 Hook (public OkHttpClient.Builder hostnameVerifier)这个方法，应该是漏掉了这个方法。
对其中上述四个方法只需要 Hook 函数后，不抛出异常，并设置函数返回值为 true 即可绕过验证。
对于 OkHttpClient.Builder 中的 hostnameVerifier 方法的 Hook，替换成自定义的 HostnameVerifier(上述代码中的 MyHostnameVerifier 即可)。

2.1.2 通过 Apache 的 HttpClient 来实现

HttpClient 中进行 SSL 证书校验，也分为两种方式：

1）通过在 APK 中内置的证书初始化一个 KeyStore，然后用这个 KeyStore 去引导生成的 TrustManager 来提供验证。

实现代码如下：

```
public   HttpClient requestHTTPSPage(String mUrl, Context context) {
      HttpClient mHttpClient = null;
```

```
    InputStream ins = null;
    String result = "";
    try {
        ins = context.getAssets().open("app_pay.cer"); //下载的证书放到项目中的 assets 目录中
        CertificateFactory cerFactory = CertificateFactory
                .getInstance("X.509");
        Certificate cer = cerFactory.generateCertificate(ins);
        KeyStore keyStore = KeyStore.getInstance("PKCS12", "BC");
        keyStore.load(null, null);
        keyStore.setCertificateEntry("trust", cer);
        SSLSocketFactory socketFactory = new SSLSocketFactory(keyStore);
        Scheme sch = new Scheme("https", socketFactory, 443);
        mHttpClient = new DefaultHttpClient();
        mHttpClient.getConnectionManager().getSchemeRegistry()
                .register(sch);
    } catch (Exception e) {
    } finally {
        try {
            if (ins != null)
                ins.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    return mHttpClient;
}
```

2）自定义 SSLSocketFactory 实现其中的 TrustManager 校验策略。

实现代码如下：

```
public   HttpClient getHttpClient() {
    HttpClient httpClient = null;
        //  初始化工作
        try {
            KeyStore trustStore = KeyStore.getInstance(KeyStore
                    .getDefaultType());
            trustStore.load(null, null);
            SSLSocketFactory sf = new SSLSocketFactoryEx(trustStore);
            //sf.setHostnameVerifier(SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER);    //允许
所有主机的验证
            sf.setHostnameVerifier(SSLSocketFactory.STRICT_HOSTNAME_VERIFIER);
            HttpParams params = new BasicHttpParams();
            HttpProtocolParams.setVersion(params, HttpVersion.HTTP_1_1);
            HttpProtocolParams.setContentCharset(params,
                    "utf-8");
            HttpProtocolParams.setUseExpectContinue(params, true);
```

```
                // 设置连接管理器的超时
                ConnManagerParams.setTimeout(params, 10000);
                // 设置连接超时
                HttpConnectionParams.setConnectionTimeout(params, 10000);
                // 设置 socket 超时
                HttpConnectionParams.setSoTimeout(params, 10000);
                // 设置 http https 支持
                SchemeRegistry schReg = new SchemeRegistry();
                schReg.register(new Scheme("http", PlainSocketFactory
                        .getSocketFactory(), 80));
                schReg.register(new Scheme("https", sf, 443));
                ClientConnectionManager conManager = new ThreadSafeClientConnManager(
                        params, schReg);
                httpClient = new DefaultHttpClient(conManager, params);
            } catch (Exception e) {
                e.printStackTrace();
                return new DefaultHttpClient();
            }
        return httpClient;
}
SSLSocketFactoryEx.java
class SSLSocketFactoryEx extends SSLSocketFactory {
    SSLContext sslContext = SSLContext.getInstance("TLS");
    public SSLSocketFactoryEx(KeyStore truststore)
            throws NoSuchAlgorithmException, KeyManagementException,
            KeyStoreException, UnrecoverableKeyException {
        super(truststore);
        TrustManager tm = new X509TrustManager() {
            @Override
            public java.security.cert.X509Certificate[] getAcceptedIssuers() {
                return null;
            }
            @Override
            public void checkClientTrusted(
                    java.security.cert.X509Certificate[] chain, String authType)
                    throws java.security.cert.CertificateException {
            }
            @Override
            public void checkServerTrusted(
                    java.security.cert.X509Certificate[] chain, String authType)
                    throws java.security.cert.CertificateException {
            }
        };
        sslContext.init(null, new TrustManager[] { tm }, null);
    }
    @Override
    public Socket createSocket(Socket socket, String host, int port,
                                        boolean autoClose) throws IOException, UnknownHostException {
        return sslContext.getSocketFactory().createSocket(socket, host, port,
                autoClose);
    }
    @Override
    public Socket createSocket() throws IOException {
        return sslContext.getSocketFactory().createSocket();
    }
}
```

绕过上述 SSL 证书验证，Xposed 需要 Hook 的方法名和类名如下表所示：

类名	方法名
external/apache-http/src/org/apache/http/impl/client/DefaultHttpClient.java	public DefaultHttpClient()
external/apache-http/src/org/apache/http/impl/client/DefaultHttpClient.java	public DefaultHttpClient(HttpParams params)
external/apache-http/src/org/apache/http/impl/client/DefaultHttpClient.java	public DefaultHttpClient(ClientConnectionManager conman, HttpParams params)
external/apache-http/src/org/apache/http/conn/ssl/SSLSocketFactory.java	public SSLSocketFactory(String, KeyStore, String, KeyStore)
external/apache-http/src/org/apache/http/conn/ssl/SSLSocketFactory.java	public SSLSocketFactory(String, KeyStore, String, KeyStore)

Hook 的 DefaultHttpClient 三个构造方法，对中都调用(ClientConnectionManager, HttpParams)这个函数，其中重点需要 Hook 的是 ClientConnectionManager 这个参数，将其替换成如下函数内容，让其信任所有证书：

```
public ClientConnectionManager getSCCM() {
    KeyStore trustStore;
    try {
        trustStore = KeyStore.getInstance(KeyStore.getDefaultType());
        trustStore.load(null, null);
        SSLSocketFactory sf = new TrustAllSSLSocketFactory(trustStore);
        sf.setHostnameVerifier(SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER);
        SchemeRegistry registry = new SchemeRegistry();
        registry.register(new Scheme("http", PlainSocketFactory.getSocketFactory(), 80));
        registry.register(new Scheme("https", sf, 443));
        ClientConnectionManager ccm = new SingleClientConnManager(null, registry);
        return ccm;
    } catch (Exception e) {
        return null;
    }
}
```

Hook 的 SSLSocketFactory 重点是替换其中 TrustManager，将其策略可以加载信任任意证书，替换后“TrustManager”代码如下：

```
class ImSureItsLegitTrustManager implements X509TrustManager {
        @Override
        public void checkClientTrusted(X509Certificate[] chain, String authType) throws CertificateException
{ }
        @Override
        public void checkServerTrusted(X509Certificate[] chain, String authType) throws CertificateException
{ }
        @Override
        public X509Certificate[] getAcceptedIssuers() {
            return new X509Certificate[0];
        }
}
```

2.1.3 通过 HttpsURLConnection 来实现

HttpsURLConnection 中进行 SSL 证书校验，也分为两种方式：

1）自定义的 HostnameVerifier 和 X509TrustManager 实现。

实现代码如下：

```
public HttpsURLConnection getHttpsURLConnectionByTrustManagerAndHostName(String
urlStr){
    HttpsURLConnection conn = null;
    try{
        SSLContext sc = SSLContext.getInstance("TLS");
        sc.init(null, new TrustManager[]{new MyTrustManager()}, new SecureRandom());
        HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
        HttpsURLConnection.setDefaultHostnameVerifier(new MyHostnameVerifier());
        conn = (HttpsURLConnection)new URL(urlStr).openConnection();
        conn.setDoOutput(true);
        conn.setDoInput(true);
    }catch(Exception e){
        Log.e(this.getClass().getName(), e.getMessage());
    }
    return conn;
}
class MyHostnameVerifier implements HostnameVerifier {
    public boolean verify(String hostname, SSLSession session) {
        // TODO Auto-generated method stub
        return true;
    }
}
class MyTrustManager implements X509TrustManager {
    @Override
    public void checkClientTrusted(X509Certificate[] x509Certificates, String s)
throws CertificateException {
    }
    @Override
    public void checkServerTrusted(X509Certificate[] x509Certificates, String s)
throws CertificateException {
    }
    @Override
    public X509Certificate[] getAcceptedIssuers() {
        return new X509Certificate[0];
    }
}
```

2）使用内置的证书初始化一个 KeyStore,实现 TrustManager

实现代码如下：

```
public   HttpsURLConnection getHttpsURLConnectionByKeyStore(Context context, String url, String
method) {
    URL u;
    HttpsURLConnection connection = null;
    try {
        SSLContext sslContext = getSSLContext(context);
        if (sslContext != null) {
            u = new URL(url);
            connection = (HttpsURLConnection) u.openConnection();
            connection.setRequestMethod(method);//"POST" "GET"
            connection.setDoOutput(true);
            connection.setDoInput(true);
            connection.setUseCaches(false);
            connection.setRequestProperty("Content-Type", "binary/octet-stream");
            connection.setSSLSocketFactory(sslContext.getSocketFactory());
            connection.setConnectTimeout(30000);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return connection;
}
private   SSLContext getSSLContext(Context context) {
    try {
        // 服务器端需要验证的客户端证书
        KeyStore keyStore = KeyStore.getInstance(KEY_STORE_TYPE_P12);
        // 客户端信任的服务器端证书
```

```
            KeyStore trustStore = KeyStore.getInstance(KEY_STORE_TYPE_BKS);

            InputStream ksIn = context.getResources().getAssets().open(KEY_STORE_CLIENT_PATH);
            InputStream tsIn = context.getResources().getAssets().open(KEY_STORE_TRUST_PATH);
            try {
                keyStore.load(ksIn, KEY_STORE_PASSWORD.toCharArray());
                trustStore.load(tsIn, KEY_STORE_TRUST_PASSWORD.toCharArray());
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                try {
                    ksIn.close();
                } catch (Exception ignore) {
                }
                try {
                    tsIn.close();
                } catch (Exception ignore) {
                }
            }
            SSLContext sslContext = SSLContext.getInstance("TLS");
            TrustManagerFactory trustManagerFactory = 
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
            trustManagerFactory.init(trustStore);
            KeyManagerFactory keyManagerFactory = KeyManagerFactory.getInstance("X509");
            keyManagerFactory.init(keyStore, KEY_STORE_PASSWORD.toCharArray());
            sslContext.init(keyManagerFactory.getKeyManagers(), 
trustManagerFactory.getTrustManagers(), null);
            return sslContext;
        } catch (Exception e) {
            Log.e("tag", e.getMessage(), e);
        }
        return null;
}
```

绕过上述 SSL 证书验证，Xposed 需要 Hook 的方法名和类名如下表所示：

类名	方法名
libcore/luni/src/main/java/javax/net/ssl/TrustManagerFactory.java	public final TrustManager[] getTrustManager()
libcore/luni/src/main/java/javax/net/ssl/HttpsURLConnection.java	public void setDefaultHostnameVerifier(HostnameVerifier)
libcore/luni/src/main/java/javax/net/ssl/HttpsURLConnection.java	public void setSSLSocketFactory(SSLSocketFactory)
libcore/luni/src/main/java/javax/net/ssl/HttpsURLConnection.java	public void setHostnameVerifier(HostNameVerifier)

getTrustManager 的 Hook，跟 2.1.1 中替换方式一样，换成自定义的 TrustManager 让其信任所有证书，此处不再列出代码。
其他三个函数都是“set”代码，只需要函数替换，不做任何操作即可。

2.1.4 WebView 加载 Https 页面时的证书校验

Android 中通过 WebView 加载 Https 页面时，如果出现证书校验错误，则会停止加载页面，因为只需要 Hook 掉 webview 的证书校验失败的处理方法：onReceivedSslError，让其继续加载即可。

类名	方法名
frameworks/base/core/java/android/webkit/WebViewClient.java	public void onReceivedSslError(Webview, SslErrorHandler, SslError)
frameworks/base/core/java/android/webkit/WebViewClient.java	public void onReceivedError(WebView, int, String, String)

对于其中上述两个方法，只需要 webview 继续加载网页即可：handler.proceed()。

2.1.5 JustTrustMe 中其他 Hook 函数

JustTrustMe 中其他的 Hook 函数如下：

类名	方法名
external/apache-http/src/org/apache/http/conn/ssl/SSLSocketFactory.java	public static SSLSocketFactory getSocketFactory()/已废弃
external/apache-http/src/org/apache/http/conn/ssl/SSLSocketFactory.java	public boolean isSecure(Socket)/已废弃
ch.boye.httpclientandroidlib.conn.ssl.AbstractVerifier	verify(String, String[], String[], boolean)

前两个函数已经基本没有在使用，而第三个是使用的第三方的库——httpclientandroidlib 进行进行 https 连接的，该 jar14 年以后也没在更新了，几乎没人在使用，所以此处对着三个函数不继续进行分析，有兴趣的可以继续进行升入分析。

2.1.6 参考文章

https://docs.oracle.com/javase/7/docs/api/javax/net/ssl/SSLContext.html
http://www.apihome.cn/api/android/SSLSocketFactory.html
http://blog.sina.com.cn/s/blog_616e189f01018rpk.html
https://github.com/square/okhttp/wiki/HTTPS
https://square.github.io/okhttp/2.x/okhttp/com/squareup/okhttp/CertificatePinner.html
http://hc.apache.org/httpcomponents-client-ga/httpclient/apidocs/org/apache/http/conn/ssl/SSLSocketFactory.html
http://pingguohe.net/2016/02/26/Android-App-secure-ssl.html
http://frodoking.github.io/2015/03/12/android-okhttp/