关于Android的https通讯安全

2016-06-24 09:26:53 0个评论 来源：xiangzhihong8的专栏 收藏 我要投稿

起因

前段时间，同事拿着一个代码安全扫描出来的 bug 过来咨询，我一看原来是个 https通信时数字证书校验的漏洞，一想就明白了大概；其实这种问题早两年就有大规模的暴露，各大厂商App也纷纷中招，想不到过了这么久天猫客户端里还留有这种坑；然后仔细研究了漏洞所在的代码片段，原来所属的是新浪微博分享 sdk 内部的，因为这个 sdk是源码引用的，一直没有更新，年久失修，所以也就被扫描出来了。因此给出的解决方案是：

先获取最新的 sdk，看其内部是否已解决，已解决的话升级 sdk 版本即可；

第1步行不通，那就自己写校验逻辑，猫客全局通信基本已经使用 https 通信，参考着再写一遍校验逻辑也不是问题；

后来查了一下网上信息，早在2014年10月份，乌云
平台里就已经暴露过天猫这个漏洞，想必当时一定是忙于双十一忽略了这个问题。

虽然这个问题通过升级 sdk
解决了，但是这个问题纯粹是由于开发者本身疏忽造成的；特别是对于初级开发人员来说，可能为了解决异常，屏蔽了校验逻辑；所以我还是抽空再 review
了一下这个漏洞，整理相关信息。

漏洞描述

对于数字证书相关概念、Android 里 https 通信代码就不再复述了，直接讲问题。缺少相应的安全校验很容易导致中间人攻击，而漏洞的形式主要有以下3种：

自定义X509TrustManager

在使用HttpsURLConnection发起 HTTPS 请求的时候，提供了一个自定义的X509TrustManager，未实现安全校验逻辑，下面片段就是当时新浪微博 sdk 内部的代码片段。如果不提供自定义的X509TrustManager，代码运行起来可能会报异常（原因下文解释），初学者就很容易在不明真相的情况下提供了一个自定义的X509TrustManager，却忘记正确地实现相应的方法。本文重点介绍这种场景的处理方式。

```
TrustManager tm = new X509TrustManager() {
    public void checkClientTrusted(X509Certificate[] chain, String authType)
            throws CertificateException {
              //do nothing，接受任意客户端证书
    }
```

```
    public void checkServerTrusted(X509Certificate[] chain, String authType)
            throws CertificateException {
              //do nothing，接受任意服务端证书
    }

    public X509Certificate[] getAcceptedIssuers() {
        return null;
    }
};

sslContext.init(null, new TrustManager[] { tm }, null);
```

自定义了HostnameVerifier

在握手期间，如果 URL 的主机名和服务器的标识主机名不匹配，则验证机制可以回调此接口的实现程序来确定是否应该允许此连接。如果回调内实现不恰当，默认接受所有域名，则有安全风险。代码示例。

```
HostnameVerifier hnv = new HostnameVerifier() {
  @Override
  public boolean verify(String hostname, SSLSession session) {
    // Always return true，接受任意域名服务器
    return true;
  }
};
HttpsURLConnection.setDefaultHostnameVerifier(hnv);
```

信任所有主机名

```
SSLSocketFactory sf = new MySSLSocketFactory(trustStore);
sf.setHostnameVerifier(SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER);
```

修复方案

分而治之，针对不同的漏洞点分别描述，这里就讲的修复方案主要是针对非浏览器App，非浏览器App 的服务端通信对象比较固定，一般都是自家服务器，可以做很多特定场景的定制化校验。如果是浏览器 App，校验策略就有更通用一些。

自定义X509TrustManager。前面说到，当发起 HTTPS 请求时，可能抛起一个异常，以下面这段代码为例（来自官方文档）：

```
try {
    URL url = new URL("https://certs.cac.washington.edu/CAtest/");
    URLConnection urlConnection = url.openConnection();
    InputStream in = urlConnection.getInputStream();
    copyInputStreamToOutputStream(in, System.out);
} catch (MalformedURLException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

```
private void copyInputStreamToOutputStream(InputStream in, PrintStream out) throws IOEx
    byte[] buffer = new byte[1024];
    int c = 0;
    while ((c = in.read(buffer)) != -1) {
        out.write(buffer, 0, c);
    }
}
```

它会抛出一个SSLHandshakeException的异常。

```
javax.net.ssl.SSLHandshakeException: java.security.cert.CertPathValidatorException: Tru
    at com.android.org.conscrypt.OpenSSLSocketImpl.startHandshake(OpenSSLSocketImpl.jav
    at com.android.okhttp.Connection.upgradeToTls(Connection.java:201)
    at com.android.okhttp.Connection.connect(Connection.java:155)
    at com.android.okhttp.internal.http.HttpEngine.connect(HttpEngine.java:276)
    at com.android.okhttp.internal.http.HttpEngine.sendRequest(HttpEngine.java:211)
    at com.android.okhttp.internal.http.HttpURLConnectionImpl.execute(HttpURLConnection
    at com.android.okhttp.internal.http.HttpURLConnectionImpl.getResponse(HttpURLConnec
    at com.android.okhttp.internal.http.HttpURLConnectionImpl.getInputStream(HttpURLCon
    at com.android.okhttp.internal.http.DelegatingHttpsURLConnection.getInputStream(Del
    at com.android.okhttp.internal.http.HttpsURLConnectionImpl.getInputStream(HttpsURLC
    at me.longerian.abcandroid.datetimepicker.TestDateTimePickerActivity$1.run(TestDate
Caused by: java.security.cert.CertificateException: java.security.cert.CertPathValidato
    at com.android.org.conscrypt.TrustManagerImpl.checkTrusted(TrustManagerImpl.java:31
    at com.android.org.conscrypt.TrustManagerImpl.checkServerTrusted(TrustManagerImpl.j
    at com.android.org.conscrypt.Platform.checkServerTrusted(Platform.java:114)
    at com.android.org.conscrypt.OpenSSLSocketImpl.verifyCertificateChain(OpenSSLSocket
    at com.android.org.conscrypt.NativeCrypto.SSL_do_handshake(Native Method)
    at com.android.org.conscrypt.OpenSSLSocketImpl.startHandshake(OpenSSLSocketImpl.jav
 ... 10 more
Caused by: java.security.cert.CertPathValidatorException: Trust anchor for certificatio
 ... 16 more
```

Android 手机有一套共享证书的机制，如果目标 URL 服务器下发的证书不在已信任的证书列表里，或者该证书是自签名的，不是由权威机构颁发，那么会出异常。对于我们这种非浏览器 app 来说，如果提示用户去下载安装证书，可能会显得比较诡异。幸好还可以通过自定义的验证机制让证书通过验证。验证的思路有两种：

方案1

不论是权威机构颁发的证书还是自签名的，打包一份到 app 内部，比如存放在 asset 里。通过这份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，具体代码如下：

```
try {
  CertificateFactory cf = CertificateFactory.getInstance("X.509");
  // uwca.crt 打包在 asset 中，该证书可以从https://itconnect.uw.edu/security/securing-compu
  InputStream caInput = new BufferedInputStream(getAssets().open("uwca.crt"));
  Certificate ca;
  try {
      ca = cf.generateCertificate(caInput);
```

```
        Log.i("Longer", "ca=" + ((X509Certificate) ca).getSubjectDN());
        Log.i("Longer", "key=" + ((X509Certificate) ca).getPublicKey();
    } finally {
        caInput.close();
    }

    // Create a KeyStore containing our trusted CAs
    String keyStoreType = KeyStore.getDefaultType();
    KeyStore keyStore = KeyStore.getInstance(keyStoreType);
    keyStore.load(null, null);
    keyStore.setCertificateEntry("ca", ca);

    // Create a TrustManager that trusts the CAs in our KeyStore
    String tmfAlgorithm = TrustManagerFactory.getDefaultAlgorithm();
    TrustManagerFactory tmf = TrustManagerFactory.getInstance(tmfAlgorithm);
    tmf.init(keyStore);

    // Create an SSLContext that uses our TrustManager
    SSLContext context = SSLContext.getInstance("TLSv1","AndroidOpenSSL");
    context.init(null, tmf.getTrustManagers(), null);

    URL url = new URL("https://certs.cac.washington.edu/CAtest/");
    HttpsURLConnection urlConnection =
            (HttpsURLConnection)url.openConnection();
    urlConnection.setSSLSocketFactory(context.getSocketFactory());
    InputStream in = urlConnection.getInputStream();
    copyInputStreamToOutputStream(in, System.out);
  } catch (CertificateException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  } catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
  } catch (KeyStoreException e) {
    e.printStackTrace();
  } catch (KeyManagementException e) {
    e.printStackTrace();
  } catch (NoSuchProviderException e) {
    e.printStackTrace();
  }
```

这样就可以得到正确的输出内容：

UW Services CA test page

QUESTION: Did you arrive here without any security alerts or warnings?

YES - This test page uses a certificate issued by the UW Services Certificate Authority. If you reached this page without any alerts or

```
warnings from your browser, you have successfully installed the UW
Services CA Certificate into your browser.

NO - If your browser warned you about the validity of this test page's
security certificate, or the certificate authority is unrecognized, you
may not have successfully installed the UW Services CA Certificate.

Return to the Install Page
```

如果你用上述同样的代码访问 https://www.taobao.com/ 或者 https://www.baidu.com/ ，则会抛出那个SSLHandshakeException异常，也就是说对于特定证书生成的TrustManager，只能验证与特定服务器建立安全链接，这样就提高了安全性。如之前提到的，对于非浏览器 app 来说，这是可以接受的。

方案2

同方案1，打包一份到证书到 app 内部，但不通过KeyStore去引导生成的TrustManager，而是干脆直接自定义一个TrustManager，自己实现校验逻辑；校验逻辑主要包括：

?服务器证书是否过期
?证书签名是否合法

```
try {
  CertificateFactory cf = CertificateFactory.getInstance("X.509");
  // uwca.crt 打包在 asset 中，该证书可以从https://itconnect.uw.edu/security/securing-compu
  InputStream caInput = new BufferedInputStream(getAssets().open("uwca.crt"));
  final Certificate ca;
  try {
      ca = cf.generateCertificate(caInput);
      Log.i("Longer", "ca=" + ((X509Certificate) ca).getSubjectDN());
      Log.i("Longer", "key=" + ((X509Certificate) ca).getPublicKey());
  } finally {
      caInput.close();
  }
  // Create an SSLContext that uses our TrustManager
  SSLContext context = SSLContext.getInstance("TLSv1","AndroidOpenSSL");
  context.init(null, new TrustManager[]{
          new X509TrustManager() {
              @Override
              public void checkClientTrusted(X509Certificate[] chain,
                      String authType)
                      throws CertificateException {

              }

              @Override
              public void checkServerTrusted(X509Certificate[] chain,
                      String authType)
                      throws CertificateException {
                  for (X509Certificate cert : chain) {

                      // Make sure that it hasn't expired.
                      cert.checkValidity();
```

```
                    // Verify the certificate's public key chain.
                    try {
                        cert.verify(((X509Certificate) ca).getPublicKey());
                    } catch (NoSuchAlgorithmException e) {
                        e.printStackTrace();
                    } catch (InvalidKeyException e) {
                        e.printStackTrace();
                    } catch (NoSuchProviderException e) {
                        e.printStackTrace();
                    } catch (SignatureException e) {
                        e.printStackTrace();
                    }
                }
            }

            @Override
            public X509Certificate[] getAcceptedIssuers() {
                return new X509Certificate[0];
            }
        }
  }, null);

  URL url = new URL("https://certs.cac.washington.edu/CAtest/");
  HttpsURLConnection urlConnection =
          (HttpsURLConnection)url.openConnection();
  urlConnection.setSSLSocketFactory(context.getSocketFactory());
  InputStream in = urlConnection.getInputStream();
  copyInputStreamToOutputStream(in, System.out);
} catch (CertificateException e) {
  e.printStackTrace();
} catch (IOException e) {
  e.printStackTrace();
} catch (NoSuchAlgorithmException e) {
  e.printStackTrace();
} catch (KeyManagementException e) {
  e.printStackTrace();
} catch (NoSuchProviderException e) {
  e.printStackTrace();
}
```

同样上述代码只能访问 certs.cac.washington.edu 相关域名地址，如果访问 https://www.taobao.com/ 或者 https://www.baidu.com/ ，则会在cert.verify(((X509Certificate) ca).getPublicKey());处抛异常，导致连接失败。

?自定义HostnameVerifier，简单的话就是根据域名进行字符串匹配校验；业务复杂的话，还可以结合配置中心、白名单、黑名单、正则匹配等多级别动态校验；总体来说逻辑还是比较简单的，反正只要正确地实现那个方法。

```
HostnameVerifier hnv = new HostnameVerifier() {
  @Override
```

```
  public boolean verify(String hostname, SSLSession session) {
    //示例
    if("yourhostname".equals(hostname)){
      return true;
    } else {
      HostnameVerifier hv =
            HttpsURLConnection.getDefaultHostnameVerifier();
      return hv.verify(hostname, session);
    }
  }
};

    ?主机名验证策略改成严格模式

SSLSocketFactory sf = new MySSLSocketFactory(trustStore);
sf.setHostnameVerifier(SSLSocketFactory.STRICT_HOSTNAME_VERIFIER);
```


点击复制链接 与好友分享! 回本站首页

相关TAG标签 通讯

上一篇： Android屏幕适配

下一篇： Android-启动模式task-lunchmodle-intent flag 总结

相关文章

android UI进阶之弹窗的使用（2）--实

获取手机通讯录信息方法总结

Android 与 HttpClient 通讯出现乱

Android 个人通讯录【安卓进化十四】

使用Joson的格式字符串在Socket中通讯

android native service编写及两个服

android中跨进程通讯的4种方式

Android中Socket通讯类

Android入门：增删改查通讯录

Android开发：还原通讯录、历史通话记

热门专题推荐 python div+css css教程 html5 html教程 jquery Android SDK php mysql oracle

图文推荐