首先可以让服务器配置支持TLS1.0、1.1、1.2，这样客户端不需要做任何处理，就可以兼容Android5.0以下的手机了。

但后台那边也是出于安全，才会只放行安全性更高的TLS1.1和TLS1.2协议。所以我们只能在客户端添加新协议的支持。

因此我们需要给低版本的协议添加Cipher suites,我们需要自定义SSLSocketFactory,自定义的SSLSocketFactory如下:

import java.io.IOException;

import java.net.InetAddress;

import java.net.Socket;

import java.net.UnknownHostException;

import java.security.GeneralSecurityException;

import java.util.Arrays;

import java.util.HashSet;

import java.util.LinkedList;

import java.util.List;

import javax.net.ssl.SSLContext;

import javax.net.ssl.SSLSocket;

import javax.net.ssl.SSLSocketFactory;

import javax.net.ssl.X509TrustManager;

public class SSLSocketFactoryCompat extends SSLSocketFactory {

private SSLSocketFactory defaultFactory;

// Android 5.0+ (API level21) provides reasonable default settings

// but it still allows SSLv3

// https://developer.android.com/about/versions/android-5.0-changes.html#ssl

static String protocols[] = null, cipherSuites[] = null;

static {

try {

SSLSocket socket = (SSLSocket)SSLSocketFactory.getDefault().createSocket();

if (socket != null) {

/\* set reasonable protocol versions \*/

// - enable all supported protocols (enables TLSv1.1 and TLSv1.2 on Android <5.0)

// - remove all SSL versions (especially SSLv3) because they're insecure now

List<String> protocols = new LinkedList<>();

for (String protocol : socket.getSupportedProtocols())

if (!protocol.toUpperCase().contains("SSL"))

protocols.add(protocol);

SSLSocketFactoryCompat.protocols = protocols.toArray(new String[protocols.size()]);

/\* set up reasonable cipher suites \*/

if (Build.VERSION.SDK\_INT < Build.VERSION\_CODES.LOLLIPOP) {

// choose known secure cipher suites

List<String> allowedCiphers = Arrays.asList(

// TLS 1.2

"TLS\_RSA\_WITH\_AES\_256\_GCM\_SHA384",

"TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256",

"TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256",

"TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256",

"TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384",

"TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256",

"TLS\_ECHDE\_RSA\_WITH\_AES\_128\_GCM\_SHA256",

// maximum interoperability

"TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA",

"TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA",

// additionally

"TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA",

"TLS\_ECDHE\_ECDSA\_WITH\_3DES\_EDE\_CBC\_SHA",

"TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA",

"TLS\_ECDHE\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA",

"TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA");

List<String> availableCiphers = Arrays.asList(socket.getSupportedCipherSuites());

// take all allowed ciphers that are available and put them into preferredCiphers

HashSet<String> preferredCiphers = new HashSet<>(allowedCiphers);

preferredCiphers.retainAll(availableCiphers);

/\* For maximum security, preferredCiphers should \*replace\* enabled ciphers (thus disabling

\* ciphers which are enabled by default, but have become unsecure), but I guess for

\* the security level of DAVdroid and maximum compatibility, disabling of insecure

\* ciphers should be a server-side task \*/

// add preferred ciphers to enabled ciphers

HashSet<String> enabledCiphers = preferredCiphers;

enabledCiphers.addAll(new HashSet<>(Arrays.asList(socket.getEnabledCipherSuites())));

SSLSocketFactoryCompat.cipherSuites = enabledCiphers.toArray(new String[enabledCiphers.size()]);

}

}

} catch (IOException e) {

throw new RuntimeException(e);

}

}

public SSLSocketFactoryCompat(X509TrustManager tm) {

try {

SSLContext sslContext = SSLContext.getInstance("TLS");

sslContext.init(null, (tm != null) ? new X509TrustManager[] { tm } : null, null);

defaultFactory = sslContext.getSocketFactory();

} catch (GeneralSecurityException e) {

throw new AssertionError(); // The system has no TLS. Just give up.

}

}

private void upgradeTLS(SSLSocket ssl) {

// Android 5.0+ (API level21) provides reasonable default settings

// but it still allows SSLv3

// https://developer.android.com/about/versions/android-5.0-changes.html#ssl

if (protocols != null) {

ssl.setEnabledProtocols(protocols);

}

if (Build.VERSION.SDK\_INT < Build.VERSION\_CODES.LOLLIPOP && cipherSuites != null) {

ssl.setEnabledCipherSuites(cipherSuites);

}

}

@Override

public String[] getDefaultCipherSuites() {

return cipherSuites;

}

@Override

public String[] getSupportedCipherSuites() {

return cipherSuites;

}

@Override

public Socket createSocket(Socket s, String host, int port, boolean autoClose) throws IOException {

Socket ssl = defaultFactory.createSocket(s, host, port, autoClose);

if (ssl instanceof SSLSocket)

upgradeTLS((SSLSocket)ssl);

return ssl;

}

@Override

public Socket createSocket(String host, int port) throws IOException, UnknownHostException {

Socket ssl = defaultFactory.createSocket(host, port);

if (ssl instanceof SSLSocket)

upgradeTLS((SSLSocket)ssl);

return ssl;

}

@Override

public Socket createSocket(String host, int port, InetAddress localHost, int localPort) throws IOException, UnknownHostException {

Socket ssl = defaultFactory.createSocket(host, port, localHost, localPort);

if (ssl instanceof SSLSocket)

upgradeTLS((SSLSocket)ssl);

return ssl;

}

@Override

public Socket createSocket(InetAddress host, int port) throws IOException {

Socket ssl = defaultFactory.createSocket(host, port);

if (ssl instanceof SSLSocket)

upgradeTLS((SSLSocket)ssl);

return ssl;

}

@Override

public Socket createSocket(InetAddress address, int port, InetAddress localAddress, int localPort) throws IOException {

Socket ssl = defaultFactory.createSocket(address, port, localAddress, localPort);

if (ssl instanceof SSLSocket)

upgradeTLS((SSLSocket)ssl);

return ssl;

}

}

然后只需要给我们的请求设置这个SSLSocketFactory就可以了,我们以okhttp为例：

//定义一个信任所有证书的TrustManager

final X509TrustManager trustAllCert = new X509TrustManager() {

@Override

public void checkClientTrusted(java.security.cert.X509Certificate[] chain, String authType) throws CertificateException {

}

@Override

public void checkServerTrusted(java.security.cert.X509Certificate[] chain, String authType) throws CertificateException {

}

@Override

public java.security.cert.X509Certificate[] getAcceptedIssuers() {

return new java.security.cert.X509Certificate[]{};

}

};

//设置OkHttpClient

OkHttpClient client = new OkHttpClient.Builder().

sslSocketFactory(new SSLSocketFactoryCompat(trustAllCert)， trustAllCert).build();

设置之后,用低版本手机测试https,就可以测试成功了。

————————————————

版权声明：本文为CSDN博主「江户川米兰」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/webber888/java/article/details/80993471