# 4.4.2支持TLS1.2

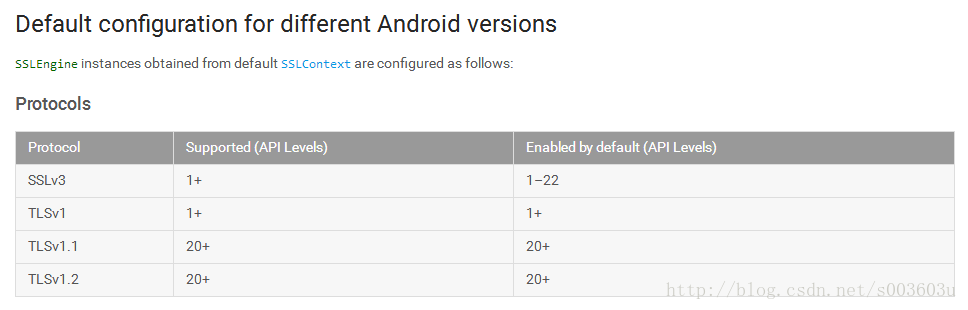
### 原因

Android4.x系统对TLS的支持存在版本差异，具体细节请看以下分析

### 分析

首先我们查看一下Google关于[SSLEngine](https://developer.android.com/reference/javax/net/ssl/SSLEngine.html" \t "/home/thtfit/文档\\x/_blank)的官方文档说明

这里截取不同**[Android](http://lib.csdn.net/base/android" \o "Android知识库" \t "/home/thtfit/文档\\x/_blank)**版本针对于TLS协议的默认配置图如下：



从上图可以得出如下结论：

* TLSv1.0从API 1+就被默认打开
* TLSv1.1和TLSv1.2只有在API 20+ 才会被默认打开
* 也就是说低于API 20+的版本是默认关闭对TLSv1.1和TLSv1.2的支持，若要支持则必须自己打开

### 解决方案

* 项目网络请求库使用的是基于okhttp的okhttputils，在APP中可以这样初始化OkHttpClient，这里通过在AppParams中配置isBypassAuthen，来判断是否绕过认证，也就是无条件信任所有HTTPS网站
* 这里只是 单向认证 客户端对服务端证书的单向认证

在MainActivity中，我们这样初始化TLSOkhttpConn网络连接实例：

setContentView(R.layout.*activity\_main*);

//1.获取单例

conn = TLSOKHttpConn.*getSingleton*();

//2.是否绕过认证

conn.setBypassAuthen(**false**);

//3.设置证书以及密码

conn.setCertificates(**null**, **null**, **null**);//setCertificates(certificates, bksFile, password);

//4.初始化工具类

conn.initHttpsClient();

//5.设置回调监听

conn.setHttpGetCallBackListener(**this**);

其中，

**public** **void** initHttpsClient() {

OkHttpClient.Builder builder = **new** OkHttpClient.Builder()

.connectTimeout(30000L, TimeUnit.*MILLISECONDS*)

.readTimeout(30000L, TimeUnit.*MILLISECONDS*)

.addInterceptor(**new** LoggerInterceptor("OkHttpClient"))

.hostnameVerifier(**new** HostnameVerifier() {

@Override

**public** **boolean** verify(String hostname, SSLSession session) {

**return** **true**;

}

});

**if**(isBypassAuthen) {

SSLContext sslContext = **null**;

**try** {

sslContext = SSLContext.*getInstance*("TLSv1.2");//SSLv3

**try** {

sslContext.init(**null**, **null**, **null**);

} **catch** (KeyManagementException e) {

e.printStackTrace();

}

} **catch** (NoSuchAlgorithmException e) { e.printStackTrace(); }

SSLSocketFactory socketFactory = **new** Tls12SocketFactory(sslContext.getSocketFactory());

builder.sslSocketFactory(socketFactory,**new** HttpUtils.UnSafeTrustManager());

} **else**{

HttpUtils.SSLParams sslParams = HttpUtils.*getSslSocketFactory*(certificates, bksFile, password); //lu

builder.sslSocketFactory(sslParams.sSLSocketFactory, sslParams.trustManager);

}

OkHttpClient okHttpClient = builder.build();

OkHttpUtils.*initClient*(okHttpClient);

}

自行实现SSLSocketFactory ，实现对TLSv1.1、TLSv1.2的支持

**package** com.example.tlsconn;

**import** java.io.IOException;

**import** java.net.InetAddress;

**import** java.net.Socket;

**import** java.net.UnknownHostException;

**import** javax.net.ssl.SSLSocket;

**import** javax.net.ssl.SSLSocketFactory;

**public** **class** Tls12SocketFactory **extends** SSLSocketFactory{

**private** **static** **final** String[] *TLS\_SUPPORT\_VERSION* = {"TLSv1.1", "TLSv1.2"};

**final** SSLSocketFactory delegate;

**public** Tls12SocketFactory(SSLSocketFactory base) {

**this**.delegate = base;

}

@Override

**public** String[] getDefaultCipherSuites() {

**return** delegate.getDefaultCipherSuites();

}

@Override

**public** String[] getSupportedCipherSuites() {

**return** delegate.getSupportedCipherSuites();

}

@Override

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

**return** patch(delegate.createSocket(s, host, port, autoClose));

}

@Override

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

**return** patch(delegate.createSocket(host, port));

}

@Override

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

**return** patch(delegate.createSocket(host, port, localHost, localPort));

}

@Override

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

**return** patch(delegate.createSocket(host, port));

}

@Override

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

**return** patch(delegate.createSocket(address, port, localAddress, localPort));

}

**private** Socket patch(Socket s) {

**if** (s **instanceof** SSLSocket) {

((SSLSocket) s).setEnabledProtocols(*TLS\_SUPPORT\_VERSION*);

}

**return** s;

}

//Error:java.net.SocketException: Unconnected is not implemented

// @Override

// public Socket createSocket() throws IOException {

// return patch(delegate.createSocket());

// }

}

具体的网络访问操作，参考okhttp以及okhttputils的使用。