https防抓包机制

抓包实战

Charles　　www.charlesproxy.com

- PC端安装Charles根证书

help->SSLProxying-> Install Charles Root Ceriticate

- 安装Charles根证书到手机

help->SSLProxying-> Install Charles Root Ceriticate on a Mobile Device or Remote Browser

注意：　安装证书过程需要手机wifi设置电脑IP地址代理,否则不会下载证书

- 在手机浏览器中访问手机 chls.pro/ssl
- PC端设置代理https端口

Proxy -> SSL Proxying Settings

抓包实战

Fiddler　　https://www.telerik.com/download/fiddler

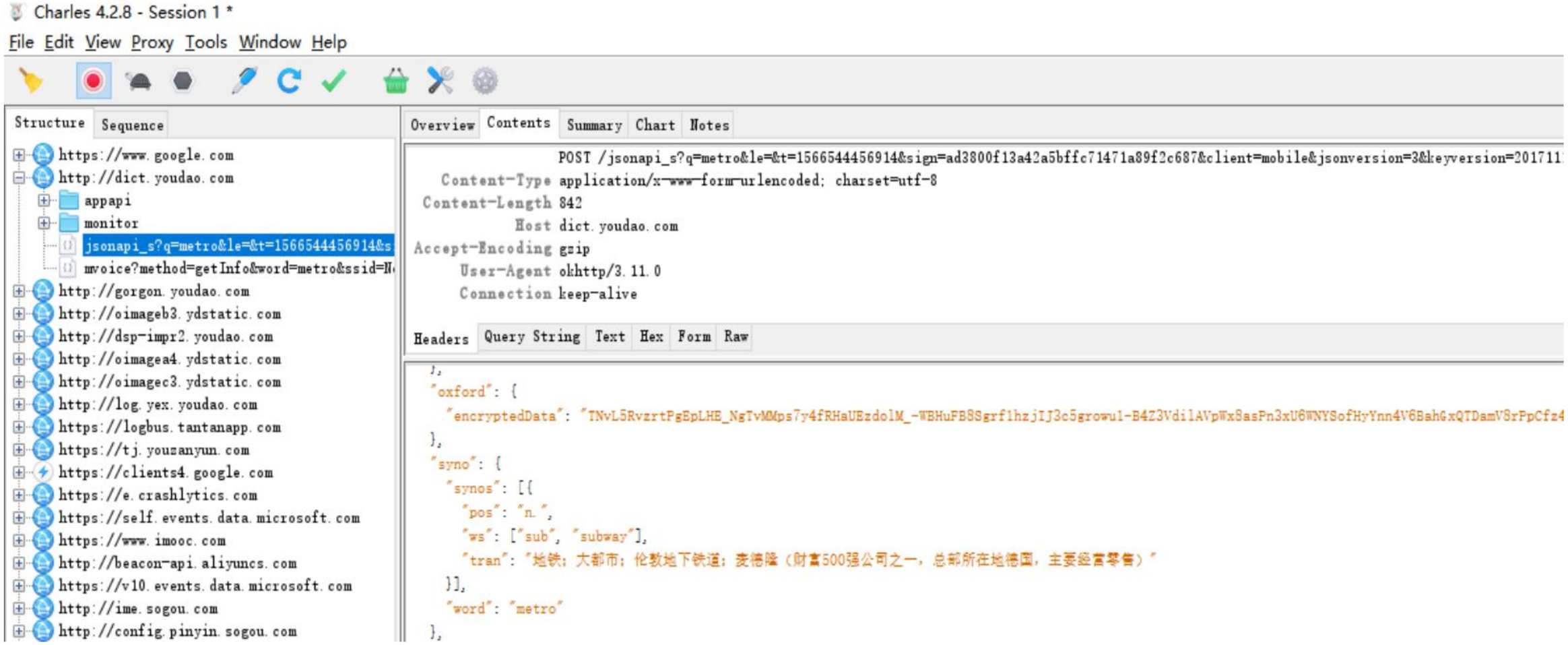
- 确保Android设备和安装Fiddler的电脑连接到同一个Wifi AP上
- 配置Fiddler抓取并解密HTTPS包

Tools->Fiddler Option->HTTPS选项卡勾选“Capture HTTPS CONNECTs”和“Decrypt HTTPS traffic”。由于通过Wifi远程连过来，所以在下面的选项框中选择“...from remote clients only”。切换到“Connections”选项卡修改监听端口，勾选上“Allow remote computers to connect”。

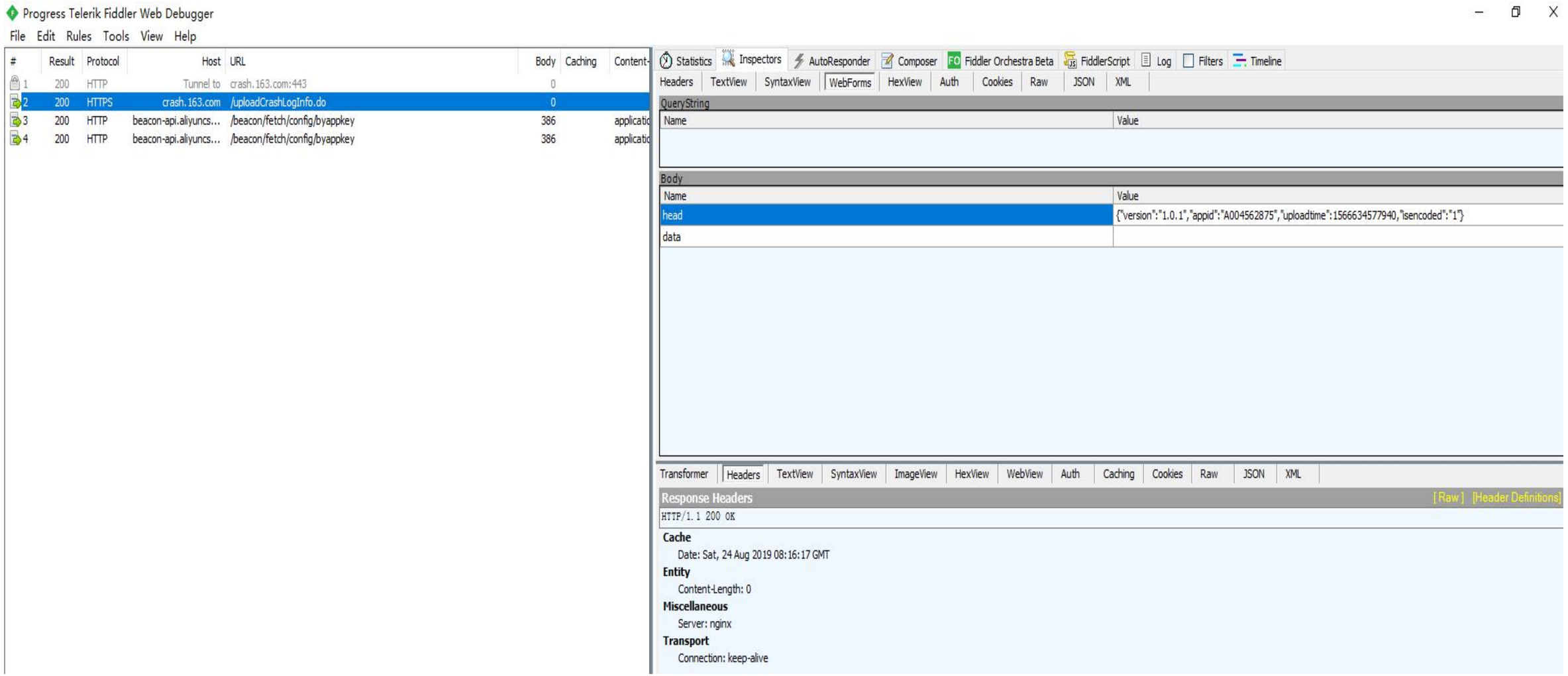
- 设置Android设备，添加上代理服务器
- 导证书到Android设备

打开设备自带的浏览器，在地址栏中输入代理服务器的IP和端口导入FiddlerRoot certificate

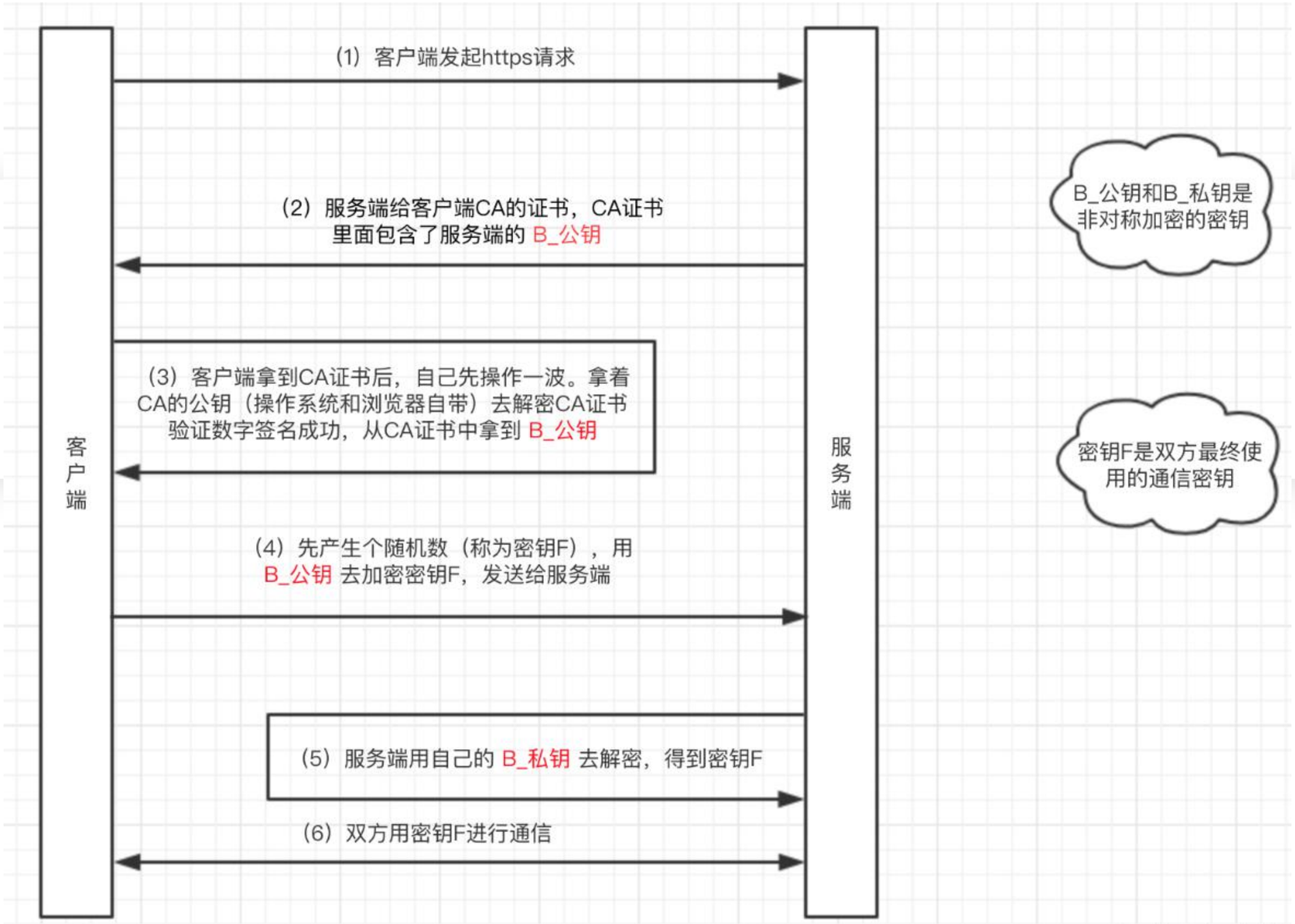
http抓包


Charles 4.2.8 - Session 1 *
File Edit View Proxy Tools Window Help
Structure Sequence
https://www.google.com
http://dict.youdao.com
appapi
monitor
jsonapi_s?q=metro&le=&t=1566544456914&s
mvoice?method=getInfo&word=metro&ssid=M
http://gorgon.youdao.com
http://oimageb3.ydstatic.com
http://dsp-impr2.youdao.com
http://oimagea4.ydstatic.com
http://oimagec3.ydstatic.com
http://log.yex.youdao.com
https://logbus.tantanapp.com
https://tj.youzanyun.com
https://clients4.google.com
https://e.crashlytics.com
https://self.events.data.microsoft.com
https://www.imooc.com
http://beacon-api.aliyuncs.com
https://v10.events.data.microsoft.com
http://ime.sogou.com
http://config.pinyin.sogou.com
Overview Contents Summary Chart Notes
POST /jsonapi_s?q=metro&le=&t=1566544456914&sign=ad3800f13a42a5bffc71471a89f2c687&client=mobile&jsonversion=3&keyversion=201711
Content-Type application/x-www-form-urlencoded; charset=utf-8
Content-Length 842
Host dict.youdao.com
Accept-Encoding gzip
User-Agent okhttp/3.11.0
Connection keep-alive
Headers Query String Text Hex Form Raw
},
"oxford": {
"encryptedData": "TNvL5RvzrtPgEpLHE_NgTvMMps7y4fRHaUEzdolM_-WBHuFB8Sgrf1hzjIJ3c5growu1-B4Z3VdilAVpWx8asPn3xU6WNYSofHyYnn4V6BahGxQTDamV8rPpCfz4
},
"syno": {
"synos": [{
"pos": "n.",
"ws": ["sub", "subway"],
"tran": "地铁；大都市；伦敦地下铁道；麦德隆（财富500强公司之一，总部所在地德国，主要经营零售）"
}],
"word": "metro"
},

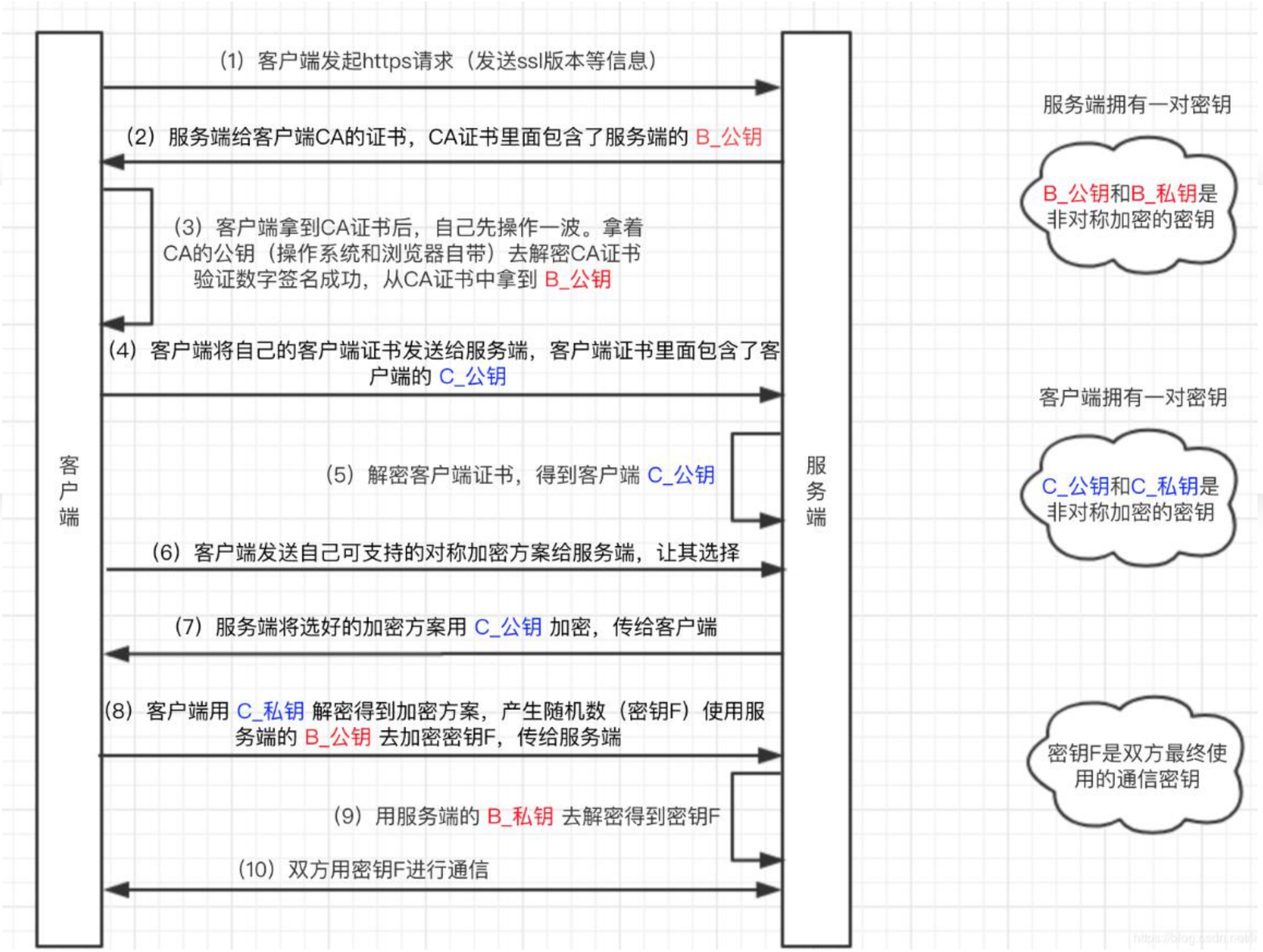
https抓包


Progress Telerik Fiddler Web Debugger
File Edit Rules Tools View Help
Result Protocol Host URL Body Caching Content-
1 200 HTTP Tunnel to crash.163.com:443 0
2 200 HTTPS crash.163.com /uploadCrashLogInfo.do 0
3 200 HTTP beacon-api.aliyuncs... /beacon/fetch/config/byappkey 386 applicatio
4 200 HTTP beacon-api.aliyuncs... /beacon/fetch/config/byappkey 386 applicatio
Statistics Inspectors AutoResponder Composer Fiddler Orchestra Beta FiddlerScript Log Filters Timeline
Headers TextView SyntaxView WebForms HexView Auth Cookies Raw JSON XML
QueryString
Name Value
Body
Name Value
head {"version":"1.0.1","appid":"A004562875","uploadtime":1566634577940,"isencoded":"1"}
data
Transformer Headers TextView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML
Response Headers [Raw] [Header Definitions]
HTTP/1.1 200 OK
Cache
Date: Sat, 24 Aug 2019 08:16:17 GMT
Entity
Content-Length: 0
Miscellaneous
Server: nginx
Transport
Connection: keep-alive

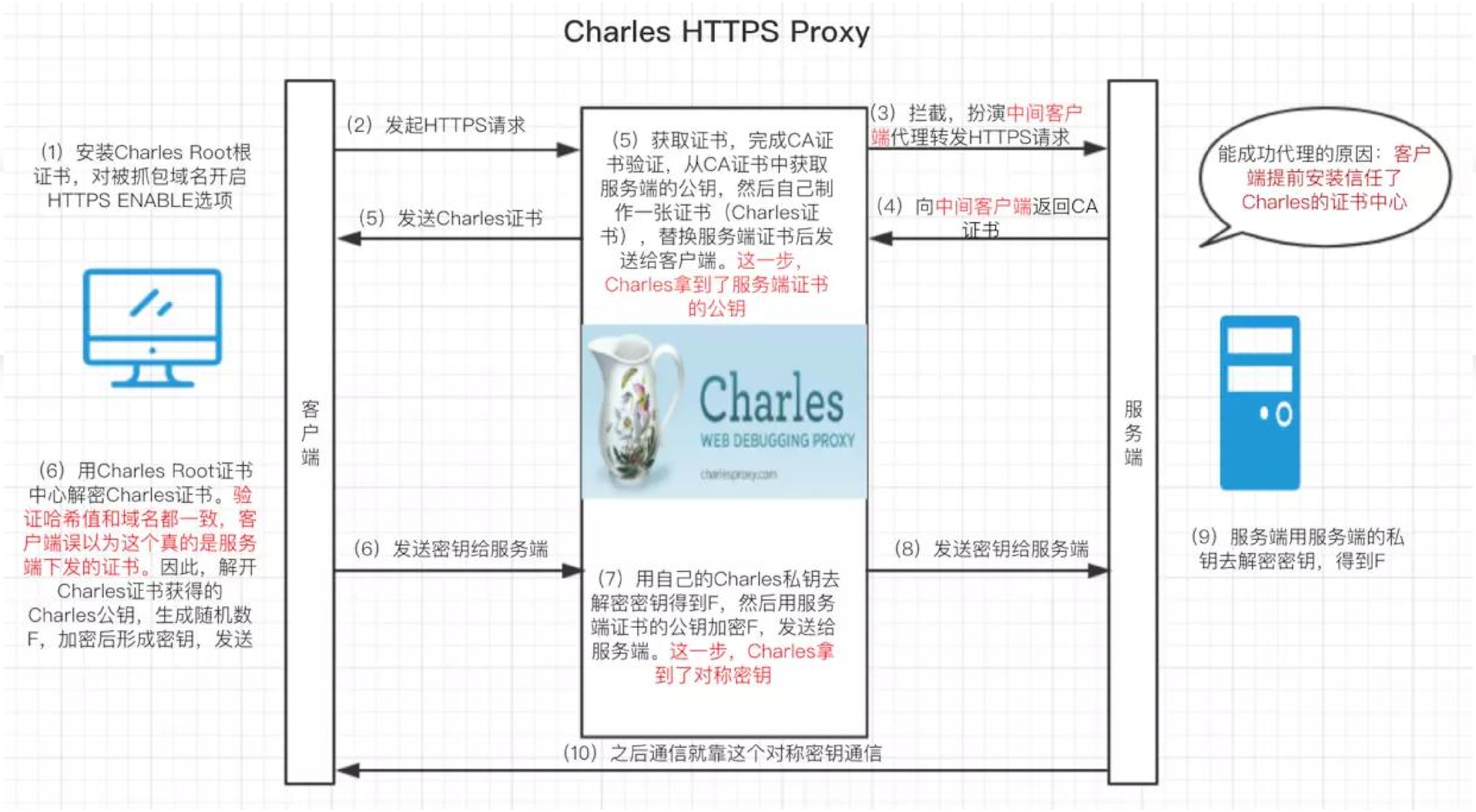
HTTPS单向认证


客户端
服务端
(1) 客户端发起https请求
(2) 服务端给客户端CA的证书，CA证书里面包含了服务端的 B_公钥
(3) 客户端拿到CA证书后，自己先操作一波。拿着CA的公钥（操作系统和浏览器自带）去解密CA证书验证数字签名成功，从CA证书中拿到 B_公钥
(4) 先产生个随机数（称为密钥F），用 B_公钥 去加密密钥F，发送给服务端
(5) 服务端用自己的 B_私钥 去解密，得到密钥F
(6) 双方用密钥F进行通信
B_公钥和B_私钥是非对称加密的密钥
密钥F是双方最终使用的通信密钥

HTTPS双向认证


客户端
服务端
(1) 客户端发起https请求（发送ssl版本等信息）
(2) 服务端给客户端CA的证书，CA证书里面包含了服务端的 B_公钥
(3) 客户端拿到CA证书后，自己先操作一波。拿着CA的公钥（操作系统和浏览器自带）去解密CA证书验证数字签名成功，从CA证书中拿到 B_公钥
(4) 客户端将自己的客户端证书发送给服务端，客户端证书里面包含了客户端的 C_公钥
(5) 解密客户端证书，得到客户端 C_公钥
(6) 客户端发送自己可支持的对称加密方案给服务端，让其选择
(7) 服务端将选好的加密方案用 C_公钥 加密，传给客户端
(8) 客户端用 C_私钥 解密得到加密方案，产生随机数（密钥F）使用服务端的 B_公钥 去加密密钥F，传给服务端
(9) 用服务端的 B_私钥 去解密得到密钥F
(10) 双方用密钥F进行通信
服务端拥有一对密钥
B_公钥和B_私钥是非对称加密的密钥
客户端拥有一对密钥
C_公钥和C_私钥是非对称加密的密钥
密钥F是双方最终使用的通信密钥

抓包原理


Charles HTTPS Proxy
(1) 安装Charles Root根证书，对被抓包域名开启HTTPS ENABLE选项
(2) 发起HTTPS请求
(3) 拦截，扮演中间客户端代理转发HTTPS请求
(4) 向中间客户端返回CA证书
(5) 发送Charles证书
(5) 获取证书，完成CA证书验证，从CA证书中获取服务端的公钥，然后自己制作一张证书（Charles证书），替换服务端证书后发送给客户端。这一步，Charles拿到了服务端证书的公钥
能成功代理的原因：客户端提前安装信任了Charles的证书中心
客户端
服务端
Charles
WEB DEBUGGING PROXY
(6) 用Charles Root证书中心解密Charles证书。验证哈希值和域名都一致，客户端误以为这个真的是服务端下发的证书。因此，解开Charles证书获得的Charles公钥，生成随机数F，加密后形成密钥，发送
(6) 发送密钥给服务端
(7) 用自己的Charles私钥去解密密钥得到F，然后用服务端证书的公钥加密F，发送给服务端。这一步，Charles拿到了对称密钥
(8) 发送密钥给服务端
(9) 服务端用服务端的私钥去解密密钥，得到F
(10) 之后通信就靠这个对称密钥通信