[原创] 关于抓包的碎碎念 优

ChenSem 5 1
专家

2020-7-27 16:25 3265

0x01

抓包是作为apk分析的首要切入点，获取apk的通信协议的必要手段。常见的抓包手段是基于中间人攻击来进行的，还有另外一种抓包手段是基于手机VPN进行的。这里将罗列常见的抓包手段，以及SSL Pinning手段的对抗。

正文

基于中间人攻击的抓包

比较常规的抓包手段，常用的抓包工具有Burpsuite、Charles、Fiddler、Mitmproxy等等。通常在手机端安装工具的证书，如若抓取的手机客户端未做SSL pinning便可直接进行抓包。

手机APP不走代理

通常有些APP为了防止被抓包，会设置APP默认不走代理，在分析的时候你会发现即使你设置了APP代理，也不影响APP的正常通信。通过tcpdump抓取网络报文可以发现app直接和服务器建立了连接而没有和代理建立连接。对应Java代码如下。

```
OkHttpClient client = new OkHttpClient().newBuilder().proxy(Proxy.NO_PROXY).build（）;
```

这种情况下可以通过建立VPN服务，直接将流量导向我们的抓包工具。当然也有现成的工具，如SocksDroid直接将流量导向抓包工具，Burpsuite是不支持socks5代理的，可以使用Charles设置Socks5代理，然后设置Charles的上游代理到Burpsuite。
还有的检测手段就是检测代理状态，如果设置代理了就拒绝通信，这种情况下也可以使用SocksDroid工具直接转发流量，这种情况下检测不到代理，另外的方法就是将检测的代码Bypass掉。
之前遇到过用Flutter写的应用，这种应用默认不走系统代理，所以也使用SocksDroid进行流量转发，然后需要注意的是需要将抓包证书安装在System Level而不是User Level。因为在User level的证书，应用是不信任的。

[安装系统证书] https://blog.ropnop.com/configuring-burp-suite-with-android-nougat/

证书绑定（单向绑定）

证书绑定的安全策略在目前的APP开发中是越来越常见了，其根本策略就是检测客户端发来的证书是否合法，一般在APP中会硬编码合法的服务端证书信息，然后进行对比，笔者见的比较多的是OkHttp3中的一些SSL Pinning策略，而且很多APP也是使用的OKHttp3框架进行流量通信，比较常见的就是实现 HostnameVerifier接口中的verify函数，其verify函数中实现检测逻辑，返回值是boolean类型，常见的bypass策略就是Hook verify函数使其返回true。那么如何找到verify函数呢？下面将列举常见的方法：

1. frida-trace

```
frida-trace
 -U \
 -F \
 --runtime=v8 \
 -j '*!verify/iu'
```

```
 re  frida-trace -U -F -j '*!verify/iu'
Instrumenting...
int.verify: Auto-generated handler at "/Users/chensem/Documents/re/__handlers__/okhttp3.internal.byte.int/verify.js"
do.verify: Auto-generated handler at "/Users/chensem/Documents/re/__handlers__/com.realcloud.http.byte.do/verify.js"
s$1.verify: Auto-generated handler at "/Users/chensem/Documents/re/__handlers__/com.umeng.commonsdk.statistics.idtracking.s_1/verify.js"
Started tracing 3 functions. Press Ctrl+C to stop.
           /* TID 0x463 */
 21234 ms  do.verify("cps.loocha.cn", "<instance: javax.net.ssl.SSLSession, $className: com.android.org.conscrypt.Java8ExtendedSSLSession>"
)
 21235 ms     | int.verify("cps.loocha.cn", "<instance: javax.net.ssl.SSLSession, $className: com.android.org.conscrypt.Java8ExtendedSSLSes
sion>")
 21236 ms     | <= true
 21237 ms  <= true
```

可以看到的确调用了verify函数，上图是我在没有设置代理的情况下返回的是true。

2. frida Java.enumerateMethods

前不久frida提供了枚举Methods的接口，这样就比以前更易于操作了，不然之前得枚举类然后得到方法。使用以下脚本便可

```
const groups = Java.enumerateMethods('*!verify/u')
// console.log(JSON.stringify(groups, null, 2));
var classes = null;
for(var i in groups){
 var classes = groups[i]['classes']

 for(var i in classes){
     Java.use(classes[i]['name'])
     .verify
     .overload('java.lang.String', 'javax.net.ssl.SSLSession')
     .implementation = function(){
         console.log("[+] invoke verify");
         return true
     }
 }
}
```

3. 找到verify函数，直接Hook
   此方法不能实现通用脚本
4. 上述情况是针对okhttp3的情况说明的，如遇到其他框架请采用网上集成的bypass脚本。

https://raw.githubusercontent.com/WooyunDota/DroidSSLUnpinning/master/ObjectionUnpinningPlus/hooks.js

证书绑定（双向绑定）

双向绑定即客户端验证服务端证书，服务端验证客户端证书。要实现抓包需要绕过客户端对服务端的校验，以及服务端对客户端的证书校验，首先绕过客户端对服务端的校验一般采用以上方法即可。绕过服务端的校验一般需要提取嵌入在客户端中的证书，有的开发者将此加密了，有的可以在assets目录中找到，以及raw目录中。对于加密的证书提取，可以Hook Keystore的load方法

```
Java.perform(function () {
    var StringClass = Java.use("java.lang.String");
    var KeyStore = Java.use("java.security.KeyStore");
    KeyStore.load.overload('java.security.KeyStore$LoadStoreParameter').implementation = function (arg0) {
        console.log("KeyStore.load1:", arg0);
        this.load(arg0);
    };
    KeyStore.load.overload('java.io.InputStream', '[C').implementation = function (arg0, arg1) {
        send(arg0)
        console.log("KeyStore.load2:", arg0, arg1 ? StringClass.$new(arg1) : null);
        this.load(arg0, arg1);
    };
});
```

```
private static KeyManager[] g(Context context) {
    try {
        InputStream openRawResource = context.getResources().openRawResource(2131558401);
        if (openRawResource == null) {
            return null;
        }
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(openRawResource));
        StringBuffer stringBuffer = new StringBuffer();
        while (true) {
            String readLine = bufferedReader.readLine();
            if (readLine != null) {
                stringBuffer.append(readLine);
            } else {
                bufferedReader.close();
                ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(MutualSSLWebViewClient.decryptToE
                KeyStore instance = KeyStore.getInstance("PKCS12");
                instance.load(byteArrayInputStream, a.n.toCharArray());
                KeyManagerFactory instance2 = KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
                instance2.init(instance, a.n.toCharArray());
                return instance2.getKeyManagers();
            }
        }
    } catch (Exception e2) {
        e2.printStackTrace();
        return null;
    }
}
```

比如上述策略，证书是经过加密保存在apk中的，取出来先进行解密，然后使用Keystore进行加载，这时候可以通过Hook load方法得到证书以及证书的密钥，然后将证书导入到Burpsuite中，便可抓取报文。

奇淫技巧抓包

0x01 Hook SSL_write SSL_read

```
var ssl_write = Module.getExportByName(null,"SSL_write");
var ssl_read = Module.getExportByName(null,"SSL_read");
Interceptor.attach(ssl_write,
{
    onEnter: function (args)
    {
      send(,Memory.readByteArray(args[1], parseInt(args[2])));
    },
    onLeave: function (retval){
}
Interceptor.attach(ssl_read,
{
    onEnter: function (args)
    {
      send(,Memory.readByteArray(args[1], parseInt(args[2])));
    },
    onLeave: function (retval){
 }
```

https://sec.mrfan.xyz/2019/12/16/%E5%AE%89%E5%8D%93%E6%B5%8B%E8%AF%95%E4%B9%8BHook%20SSL_read%E5%92%8CSSL_write/

https://github.com/fanxs-t/Android-SSL_read-write-Hook/blob/master/frida-hook.py

这种方式抓取的是未经加密的报文，经测试，有的场景下无效。

0x02 SSL key logger

这种技巧相当于PC上运行chrome设置SSLLOGFILE环境变量记录SSL协商的密钥，在android上，默认未设置需要通过Hook进行设置

```
frida -U -f com.realcloud.loochadroid.college -l hook.js --no-pause -o key
```

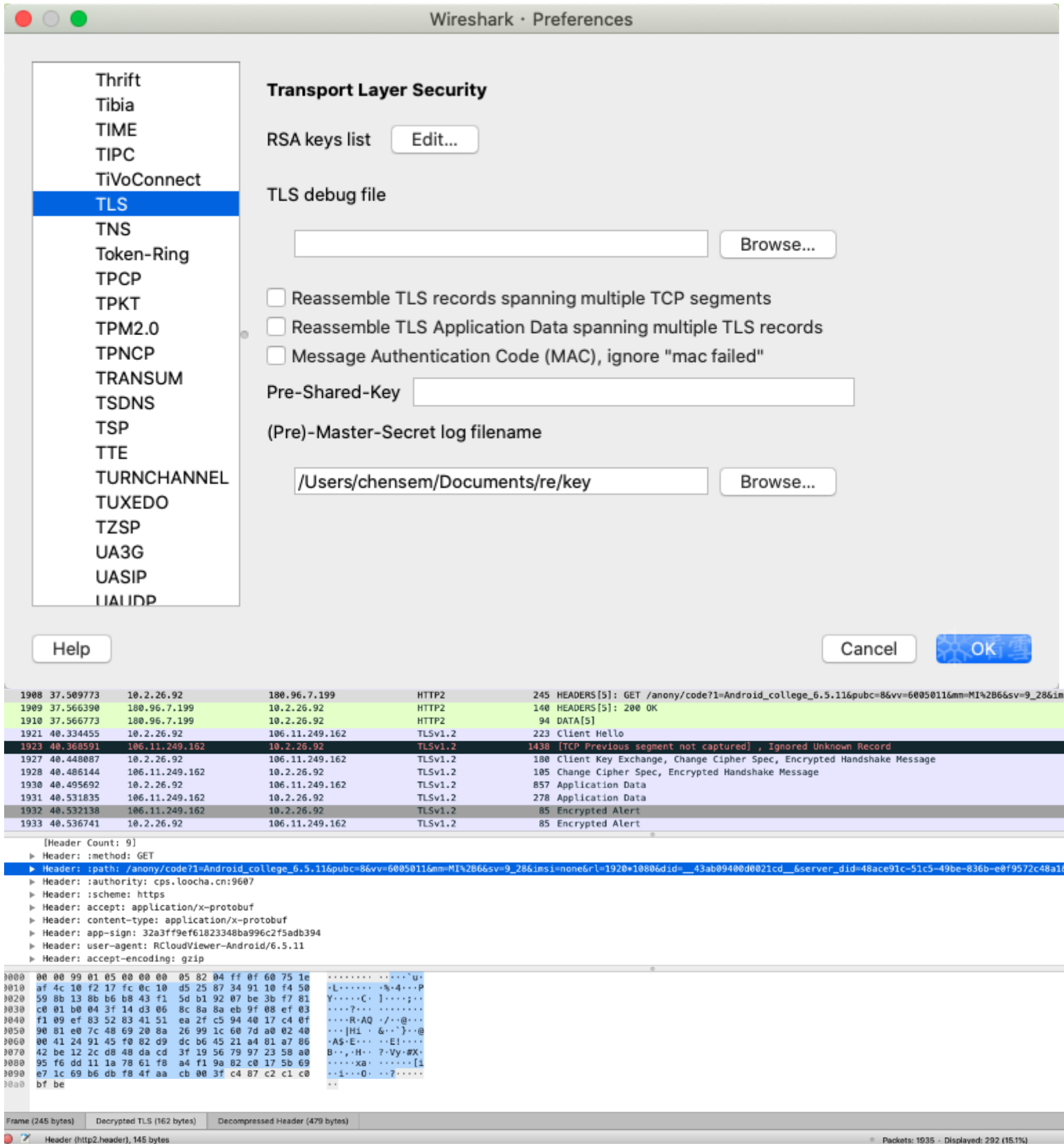
抓出来是这样的格式

```
CLIENT_RANDOM c9bc999bd139d547d49843a335d08cedf111b1254e1ae54cfb2c81c991620b22 0d63099a15c786ec238af7223c2b8d4075395b4eedabab9a963c4c95f615cdab098d633da51ff4bd813d45003fcfaf90
CLIENT_RANDOM a7c0ffd240be882ba81a45a18b26de770c033712c9ae0335f3ddac2f887b4237 931d4cfb0253e1d65f354b36353de6318471a90fac77ea9e9f64c8a4baa0400a4b57b4348cd6eb2299d5e5e6af65247e
CLIENT_RANDOM 5c693602f4ea17c112c517084edc480515d4d7edf09ee65ce966005b031ffec0 dd09e924819e5bf7f67c655e17f4fd6bd7e291636929a559f30de3260ddf54dfc772e1b7c2db11280bfd90b1f953c27e
CLIENT_RANDOM c0bdf0c9122b121e54899e5698a172c79c007c3b6cdecb11330e716fc42447a6 0d63099a15c786ec238af7223c2b8d4075395b4eedabab9a963c4c95f615cdab098d633da51ff4bd813d45003fcfaf90
CLIENT_RANDOM ae9b763a3d81a431bbcdb336bf27cab75a8879f91c97d890618a8e12e78874ce 05077ee0e3552268b2e696db32fe396fa5822497ca4562ad5acaaa70145192b127498fafdc9bbd01ac42987129a0d98b
CLIENT_RANDOM bfc1e5ebd10b288a1922f49114d79dcabd02142b01bd1910839e74da7c0becf0 3063f196bf4f16f40db1947e564e6479f9d34c8d75140f00b97db923ff81a1b455db9049575cb6b67fedb4d1178f7ddc
CLIENT_RANDOM 18d82f9e1c2faf64023eeef7eb70bca73af01487e0da8bc8b29822f969969767 289d0e0344b90b1b49fd725838bad588d27dd9432644e700be27b236e0303f41651b5d7f7c829b2c5ae63fd8a111bad1
CLIENT_RANDOM 4993657f6c5658106c0fd64c7f0596b9799ad9c89b5c43d48dc9562056d95642 81589d8c2e5cceb6d591989278f6b97dde1e9dde2e4a6d837ffe859318bba65fcb9a9ea61245cca47ebbfe5cd874dd18
CLIENT_RANDOM ca2b2baaa07102a1519c5c6c24981a308a701fe26a4c87beb588e5ab9a102c7e 10f046e879b3179315212d2cdce5e18a346b2995a6af7d23fb91408ae8fb97d7fc2d63f7ff9e4df3666f1cdefd52c502
CLIENT_RANDOM aab2f10dc3a7ac1d4eda5666fe223d2c0cfb8e5d2b100a9e45fa3f9695cdb46f 1b1c6b9080e90fd8a540cc9afd369b2ccc6ecb649760c0e2dcc72e8b40c62491ec0e3f2001d204f46d639f6cf9c98aaa
CLIENT_RANDOM 536e07a3f893521fcb6f8c7afa4002e34bf207b9a2f3098b244b201b3c670650 0d63099a15c786ec238af7223c2b8d4075395b4eedabab9a963c4c95f615cdab098d633da51ff4bd813d45003fcfaf90
CLIENT_RANDOM e7f776f7fb8bcaafa00d3784d78f065276ec2418e1bcabc318032498ae3ba9a9 0d63099a15c786ec238af7223c2b8d4075395b4eedabab9a963c4c95f615cdab098d633da51ff4bd813d45003fcfaf90
CLIENT_RANDOM 9980e9f4b37556e6874ba862e5fbf3f5f15d3da3eb83743896e608e55a4e0e3f 0d63099a15c786ec238af7223c2b8d4075395b4eedabab9a963c4c95f615cdab098d633da51ff4bd813d45003fcfaf90
CLIENT_RANDOM 85e29fcf53cb366160fb0f11eb5a1e96d5916d83d118a2fac01c0ca1b1b1d2c6 08c639ce096e0f36b98acfafd09a221aaf1acbcced6129b54b7de6ce79dc8986ed80fd4a70111fc19aed94aee6df26ce
CLIENT_RANDOM 9f47ffdb3cbb88c65494faec27f543e850622c1bb57e0b837e6eceb6bb15bb78 8df512f2e9e3977457d6584d1acb20e8d463da29f8ee7edbd83792ab718cd632e1ff7b7c9649938960c85ef51de3301e
CLIENT_RANDOM 23805c87c87928a705a151a878f6828bf6d171d037c466dca932ae5062b177d9 931d4cfb0253e1d65f354b36353de6318471a90fac77ea9e9f64c8a4baa0400a4b57b4348cd6eb2299d5e5e6af65247e
CLIENT_RANDOM 0fbcb031f91ed91e1b455044f404a1399e85f350821253fe9b4173f9f4ff35ec 30315c0f61964f8c69cd1cf65d8ec1329a6173c91507bd92aa2bad86a1f68593e096523744 0a657e23b0e6545f9fc852
```

```
function startTLSKeyLogger(SSL_CTX_new, SSL_CTX_set_keylog_callback) {
    function keyLogger(ssl, line) {
        console.log(new NativePointer(line).readCString());
    }
    const keyLogCallback = new NativeCallback(keyLogger, 'void', ['pointer', 'pointer']);

    Interceptor.attach(SSL_CTX_new, {
        onLeave: function(retval) {
            const ssl = new NativePointer(retval);
            const SSL_CTX_set_keylog_callbackFn = new NativeFunction(SSL_CTX_set_keylog_callback, 'void', ['pointer', 'pointer']);
            SSL_CTX_set_keylog_callbackFn(ssl, keyLogCallback);
        }
    });
}
startTLSKeyLogger(
    Module.findExportByName('libssl.so', 'SSL_CTX_new'),
    Module.findExportByName('libssl.so', 'SSL_CTX_set_keylog_callback')
)
```

使用方法，在程序spwan前使用tcpdump dump流量。然后wireshark设置TLS Logfile的位置



参考链接：https://codeshare.frida.re/@k0nserv/tls-keylogger/

0x03 CE扫描内存

这种方式是实在没有办法采用的方法，通过扫描关键字如HTTP、接口信息搜出http报文。

总结

获取HTTPS报文的方式多种多样，下面是此篇文章参考的文章

https://hackmag.com/security/ssl-sniffing/ 这是一篇讲通用SSL Pinning Bypass的方法

http://www.moserware.com/2009/06/first-few-milliseconds-of-https.html SSL建立流程

最后于 2020-7-27 16:30 被ChenSem编辑，原因:

最新回复 (10)

最新回复 (10)

darbra 4 2020-7-27 17:09

2 楼 0

感谢大佬分享 大佬优秀

极客

bluegatar 2 2020-7-27 18:17

3 楼 0

博大精深

极客

wx_Kevin Chou 0 2020-7-28 07:43

4 楼 0

🥰🥰🥰果斷先讚再看~ 寫的很詳細....

临时

愤怒的平头哥 2 2020-7-28 09:39

5 楼 0

感谢大佬分享

极客

Editor 8 2020-7-28 13:29

6 楼 0

感谢分享啊！

版主

0x指纹 10 3 2020-7-28 15:46

7 楼 0

感谢分享！

大牛

Loopher 3 2020-7-29 14:09

8 楼 0

感谢分享!

极客

Lateautumn4 0 6天前

9 楼 0

收藏收藏

临时

木鲤鱼 2 5天前

10 楼 0

有啥 方式 可以对抗hook 的呢?

极客

ChenSem 5 1 5天前

11 楼 0

> 木鲤鱼 有啥 方式 可以对抗hook 的呢?

对抗Hook，那就对抗Hook之类的工具呗。判断是否存在Hook工具运行啥的。

专家

游客

登录 | 注册 方可回帖

返回