

[原创]自识别类名 自动化Hook JustTrustMe 升级版



珍惜Any



2019-8-24 17:56

10133

抓包 往往是逆向分析的 第一步

很多的 App 在 抓包 的 时候 会断流 因为本地 做了 证书检测 这个时候 我们通常都会 会使用JustTrustMe 干掉 本地的 证书检测

但是 这个时候弊端就出来了 很多 app 是混淆的

在 JustTrustMe 的 源码里

```
try {
    classLoader.loadClass(className: "com.squareup.okhttp.CertificatePinner");
    findAndHookMethod(className: "com.squareup.okhttp.CertificatePinner",
        classLoader,
        methodName: "check",
        String.class,
        List.class,
        new XC_MethodReplacement() {
            @Override
            protected Object replaceHookedMethod(MethodHookParam methodHookParam) throws Throwable {
                return true;
            }
        });
} catch (ClassNotFoundException e) {
    // pass
    Log.d(TAG, msg: "OKHTTP 2.5 not found in " + currentPackageName + "-- not hooking");
}

//https://github.com/square/okhttp/blob/parent-3.0.1/okhttp/src/main/java/okhttp3/CertificatePinner.java#L144
Log.d(TAG, msg: "Hooking okhttp3.CertificatePinner.check(String,List) (3.x) for: " + currentPackageName);

try {
    classLoader.loadClass(className: "okhttp3.CertificatePinner");
    findAndHookMethod(className: "okhttp3.CertificatePinner",
        classLoader,
        methodName: "check",
        String.class,
```

路径都是写死的 也就导致 很多 混淆了的 App 是无效的

出现 抓包断流 无响应等等

这个时候 如果 可以 做到 自识别类名的话 就会方便很多 而不需要 静态分析 就可以直接 定位

检测证书就那几种办法 就拿常用的 okHttp框架来说

```
//OkHttpClient okHttpClient = new OkHttpClient();
OkHttpClient.Builder builder = new OkHttpClient.Builder()
    .connectTimeout(timeout: 20, TimeUnit.SECONDS)
    .certificatePinner(null) //证书锁定
    .hostnameVerifier(null) //域名验证
    .sslSocketFactory(sslSocketFactory: null, trustManager: null) //证书验证
```

系统的classloader里面全部的类名，找到和okHttp有关系的类名，根据类的一些特征，比如，父类对象名字，接口信息， 字段信息，方法参数信息，每一个参数的类型，
我是在application的oncreate后面执行的遍历，这个时候壳的dex已经释放到内存里，原版本 JustTrustMe 是在attach里面，这个方法很多壳没有进行解密，所以也会失效，（外国壳很少，可能没考虑到吧）

比如获取 OkHttpClient 来说

```
8      + import ...
0
1      public class OkHttpClient implements Cloneable, Factory, okhttp3.WebSocket.Factory {
2          static final List<Protocol> DEFAULT_PROTOCOLS;
3          static final List<ConnectionSpec> DEFAULT_CONNECTION_SPECS;
4          final Dispatcher dispatcher;
5          @Nullable
6          final Proxy proxy;
7          final List<Protocol> protocols;
8          final List<ConnectionSpec> connectionSpecs;
9          final List<Interceptor> interceptors;
0          = final List<Interceptor> networkInterceptors;
1          final okhttp3.EventListener.Factory eventListenerFactory;
2          final ProxySelector proxySelector;
3          final CookieJar cookieJar;
4          @Nullable
5          final Cache cache;
```

大家 可以 找一下他的 特征 接口类是三个 6个 集合类型，四个final类型 另外两个是 final并且是 static类型 并且 在 okhttp 包下就可以完美定位到 这个类的 名字

利用 Xposed 对里面的方法进行 批量 Hook

定位方法的办法也有很多 参数类型 返回值 等 比如 下面的方法

```
public OkHttpClient.Builder sslSocketFactory(SSLSSocketFactory sslSocketFactory, X509TrustManager trustManager) {
    if (sslSocketFactory == null) {
        throw new NullPointerException("sslSocketFactory == null");
    } else if (trustManager == null) {
        throw new NullPointerException("trustManager == null");
    } else {
        this.sslSocketFactory = sslSocketFactory;
        this.certificateChainCleaner = CertificateChainCleaner.get(trustManager);
        return this;
    }
}
```

参数1 和参数2 都是 JDK里面的方法是不可以被混淆的 只需要反射即可拿到


```
//方法 1 证书检测 2个 参数类型
if (OkHttpBuilder != null) {
    Class SSLSocketFactoryClass = getClass(path: "javax.net.ssl.SSLSocketFactory");
    Class X509TrustManagerClass = getClass(path: "javax.net.ssl.X509TrustManager");
    //先拿到 参数类型的 类
    Method SslSocketFactoryMethod = getSslSocketFactoryMethod(SSLSocketFactoryClass, X509TrustManagerClass);
    if (SslSocketFactoryMethod != null) {
        CLogUtils.e(msg: "拿到 SslSocketFactoryMethod " + SslSocketFactoryMethod.getName());
        //需要先拿到方法名字
        XposedHelpers.findAndHookMethod(OkHttpBuilder, SslSocketFactoryMethod.getName(),
            SSLSocketFactoryClass,
            X509TrustManagerClass,
            new XC_MethodHook() {
                protected void beforeHookedMethod(MethodHookParam param) throws Throwable {
                    super.beforeHookedMethod(param);
                    CLogUtils.e(msg: "Hook到 sslSocketFactory 2个参数类型 ");
                    param.args[0] = new MySSLSocketFactory();
                    param.args[1] = new MyX509TrustManager();
                    CLogUtils.e(msg: " sslSocketFactory 2个参数类型 替换成功 ");
                }
            });
    } else {
        CLogUtils.e(msg: "没有拿到 SslSocketFactoryMethod ");
    }
}
```

对 参数1 和 参数2 进行比较

```
private Method getSslSocketFactoryMethod(Class sslSocketFactoryClass, Class x509TrustManagerClass) {
    Method[] declaredMethods = OkHttpBuilder.getDeclaredMethods();
    for (int i = 0; i < declaredMethods.length; i++) {
        declaredMethods[i].setAccessible(true);
        Class<?>[] parameterTypes = declaredMethods[i].getParameterTypes();
        if (parameterTypes.length == 2) {
            if (parameterTypes[0].getName().equals(sslSocketFactoryClass.getName()) &&
                parameterTypes[1].getName().equals(x509TrustManagerClass.getName())) {
                return declaredMethods[i];
            }
        }
    }
    CLogUtils.e(msg: "没有找到 SslSocketFactoryMethod ");
    return null;
}
```

将参数 替换成 自己的 自定义的 无检测即可

因为 需要 初始化大量的类

所以 我做了选择 需要先打开 App 选择对应的 App名字 在开启抓包软件 开启 App即可 即可

下载地址 在附件

后期 我会给大家介绍， 自动化爬虫机， 利用 Hook抓包技术 直接把抓到的数据进行 转发 到自己服务器 以及，通用算法服务器 搭建等，再也不用担心 遇到 LLVM的 So 了

如果需要 App xy 等级 安全分析测试的 可以 私聊我
会用 一些特殊办法对 App的 net安全 进行评测 提 出 解决办法 和复现步骤 和建议等。

----- 更新

发现 模块如果混淆了 有的 App会出现找不到类的情况 以更新

[培训]科锐逆向工程师培训班38期--远程教学预课班将于 2020年5月28日 正式开班！

最后于 2019-8-25 00:12 被珍惜Any编辑，原因：App混淆 问题

上传的附件:

[JustMePlush--8.25.0.10.apk](#) (1.20MB, 1528次下载)

66

☆ 收藏

6

👍 赞


¥

打赏

🔗

分享

最新回复 (28)



- 

lengyang ☆

2019-8-24 18:24

2 楼

👍 0



😄支持珍惜大佬
- 
- 

sogohere 🌙 ☆

2019-8-24 18:49

3 楼

👍 0



谢谢了，前排支持
- 
- 

wx_金抽抽 ☆

2019-8-24 19:02

4 楼

👍 0



为珍惜酱打call
- 
- 

流星暴雨 🌙 🌙 ☆ ☆ ☆

2019-8-24 20:00

5 楼

👍 0

前排支持
- 
- 

小白abc 🌙 🌙 ☆

2019-8-24 20:52

6 楼

👍 0

珍惜珍惜大佬

★ ★