Poster: Understanding Malicious Cross-library Data Harvesting on Android

PUBLISHED PAPER

Title: Understanding Malicious Cross-library Data Harvesting on Android
Authors: Jice Wang, Yue Xiao, Xueqiang Wang, Yuhong Nan, Luyi Xing, JinWei Dong, Nicolas Serrano, Haoran Lu, XiaoFeng Wang, Yuqing Zhang
Email: {xiaoyue, haorlu, luyixing, xw7, xw48, nicserra}@indiana.edu, {wangjc, zhangyq, dongjw}@nipc.org.cn, nanyuhong@fudan.edu.cn
Date: August 11-14, 2021
Venue: Proceedings of the 2021 USENIX Security Symposium(USENIX'21)
DOI: https://www.usenix.org/system/files/sec21fall-wang-jice.pdf

ABSTRACT

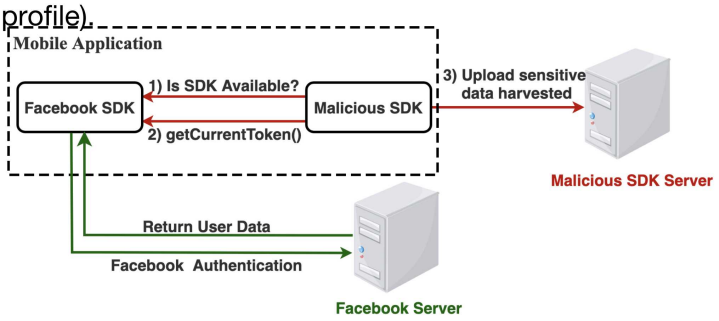
Recent years have witnessed the rise of security risks of libraries integrated in mobile apps, which are reported to steal private user data from the host apps and the app backend servers. Their security implications, however, have never been fully understood. In our research, we brought to light a new attack vector long been ignored yet with serious privacy impacts – malicious libraries strategically target other vendors'SDKs integrated in the same host app to harvest private userdata (e.g., Facebook's user profile). Using a methodology that incorporates semantic analysis on an SDK's Terms of Services (ToS, which describes restricted data access and sharing policies) and code analysis on cross-library interactions, we were able to investigate 1.3 million Google Play apps and the ToSes from 40 highly-popular SDKs, leading to the discovery of 42 distinct libraries stealthily harvesting data from 16 popular SDKs, which affect more than 19K apps with a total of 9 billion downloads. Our study further sheds light on the underground ecosystem behind such library-based data harvesting (e.g., monetary incentives for SDK integration), their unique strategies (e.g., hiding data in crash reports and using C2 server to schedule data exfiltration) and significant impacts.

Poster: Understanding Malicious Cross-library Data Harvesting on Android

Jice Wang*, Yue Xiao*, Xueqiang Wang, Yuhong Nan, Luyi Xing,
JinWei Dong, Nicolas Serrano, Haoran Lu, XiaoFeng Wang, Yuqing Zhang
(*co-first authors))

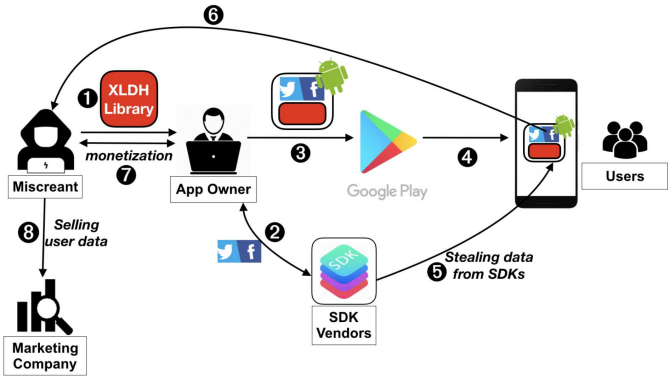
Abstract

In our research, we brought to light a new attack–malicious libraries strategically target other vendors' SDKs integrated in the same host app to harvest private user data (e.g., Facebook's user profile).



Introduction

1. app integrated with the library andthe Facebook SDK
2. passing the SDK vendor's review
3. app store vetting
4. is available for downloading
5. the XLDH library will stealthily access the Facebook token
6. send them out to its back-end platform
7. the app owner receives commissions from the adversaries
8. the brokerage platform share it with a marketing company



A Real-world example –Mobiburn

1. the classcom.mobiburn.e.h in *Mobiburn* library invoke function *com.facebook.AccessToken.getToken()* in the Facebook SDK, as shown below (a).
2. Then XLA looks up the meta-DB to determine the return value of the function, which is the user's Facebook session token, and tracks down the data flow using taint tracking.
3. Finally, the token is used to fetch a user's Facebook profile data (ID, name, gender, email, locale, link, etc.) in function com.mobiburn.e.h.getFbProfile(), and send out. (see Figure b)

```
public class h {
    public static String getAccessToken() {
        Class[] param = new Class[0];
        Class clz = Class.forName("com.facebook.AccessToken");
        Method meth1 = clz.getDeclaredMethod("getCurrentAccessToken", param)
        Object curToken = meth1.invoke(clz, null);
        Method meth2 = clz.getDeclaredMethod("getToken", param)
        return meth2.invoke(curToken, null);
    }
    public JSONObject getFbProfile(String token) {
        String uri = Uri.parse("https://graph.facebook.com/v2.10/me")
        .appendParam("accesstoken", token)
        .appendParam("fields", "id,first_name,gender,last_name,link,locale,name
            ,timezone,updated_time,verified,email");
        HttpsURLConnection httpsURLConnection = new URL(uri).openConnection();
        return new JSONObject(httpsURLConnection.getInputStream().readLine());
    }
```

(a) Reading app users' Facebook access token and profile

```
public class f {
    public void a() {
        JSONObject userData = new JSONObject();
        userData.put("accessToken", getAccessToken());
        userData.put("accountJson", getFbProfile());
        HttpsURLConnection httpsURLConnection
        = new URL(this.serverUri).openConnection();
        DataOutputStream dataOutputStream
        = httpsURLConnection.getOutputStream();
        dataOutputStream.write(userData);
    }
}
```

(b)Sending the Facebook token and profile to mobiburn server

Findings

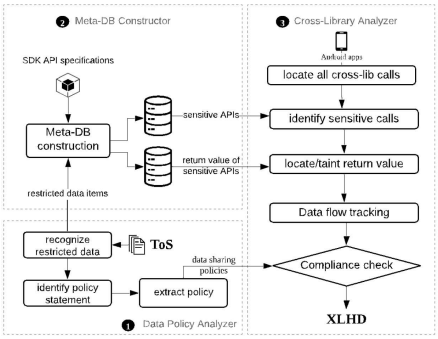
- 42 distinct XLDH libraries
- 16 victim SDKs
- 19K apps are involved
- over 35% of apps are in the categories of game and entertainment.
- more than 9 billion downloads

Table 8: Top-10 XLDH libraries (integrated in the most apps)

XLDH library	# of apps/downloads	Harvested data
com.yandex.metrica	8,014/2B+	Google Advertising ID, Android ID
com.inmobi	4,283/4B+	Google Activity Recognition
com.appsgeyser	4,202/15M+	Google Advertising ID, Android ID, IMEI, Mac Address
com.oneaudience	1,738/100M+	Facebook ID/name/gender/email/link, Twitter user data
cn.sharesdk	815/191M+	Bytedance ID/name
com.umeng.socialize	495/175M+	Facebook/Twitter/Dropbox/Kakao/Yixin/Wechat/QQ/Sina/Alipay/Laiwang/Vk/Line/Linkedin's AccessToken and user data (ID/name/link/photo)
com.revmob	340 /36M+	Facebook AccessToken
ru.mail	299/100M+	Google Advertising ID, Mac Address, Android ID, IMEI
com.ad4screen	245/183M+	Facebook appid, AccessToken
com.devtodev	231/318M+	Facebook user gender, birtbday

Methodology

- Data Policy Analyzer(DP
- Meta-DB Constructor
- Cross library Analyzer(XLA)



Awards

$ $ $

Facebook awarded us $30,000 USD through their white hat/bug bounty program, which they told us is one of their largest awards ever;

Google awarded us$5,000 USD and solicited from us the list of affected apps.

Twitter awarded us $576 USD for finding this risk to twitter user privacy.