AN INTERNET 2.0 PAPER

Internet 2.0

RELENTLESS SECURITY

DIGITAL SURVEILLANCE IN CHINA

JANUARY 2022

Authors:

David Robinson
Thomas Perkins

internet2-0.com

Introduction

With the upcoming 2022 Winter Olympics in China the team at Internet 2.0 saw the need to publish case studies demonstrating the sophisticated and broad surveillance culture that exists in China. All Chinese companies are compelled to follow the national security legislation while operating in China. All athletes and visitors to China for the Olympics will be exposed to such laws and surveillance culture. In this paper we show how these laws manifest in terms of surveillance of mobile phones through mobile applications and internet browsers through desktop software. Part 1 is an analysis of the QI-ANXIN VPN. Part 2 is an analysis of the Kingsoft's anti-virus and WPS Office software. We must state that this is not a criticism or endorsement of QI-ANXIN and Kingsoft. Internet 2.0 does not allege inappropriate conduct by QI-ANXIN or Kingsoft, rather we see it as case studies to understand exactly how the Chinese Communist Party imposes its national security legislation and its implementation in the commercial market from a technical standpoint.

Part 1 QI-ANXIN

Executive Summary

Part 1 is a technical analysis of QI-ANXIN Technology Group Inc's (QI-ANXIN) Virtual Private Network (VPN) software product. QI-ANXIN is an official sponsor to the Beijing 2022 Olympic and Paralympic Winter Games, see Figure 1.[1] The analysis is based on QI-ANXIN's publicly available Winter Olympics mobile protection software, as described on their website.[2] The Internet 2.0 intelligence team analyzed the VPN and discovered a significant amount of user data being collected by the software.

In our opinion QI-ANXIN's VPN provides a limited degree of privacy and security to its users, this assessment is based on the device and network data collected by the software. We recommend that visitors and athletes travelling to the 2022 Winter Olympics in China are aware of the risks in taking and using personal devices during the event, particularly that China's national data security laws are not designed with western values of privacy and liberty and do not offer the same level of protections. This is true for all digital communications in China and not just while using VPN software.

[1] https://en.qianxin.com/news/detail/205
[2] https://fanghu.qianxin.com/web/index/index.html

To mitigate the risk of sensitive information and personal data being collected on personal phones during the 2022 Beijing Winter Olympics, we recommend:

Athletes and visitors to the games buy and take a new phone with them to use only while inside China. This will protect their sim information of their devices that they use in their home country.

Creating a new email address and browser account and using these on the 'burner' Phone. This mitigates the risk of cloud accounts such as google, apple or internet browsers connecting all your personal information with this new and isolated 'burner' phone.

Not using this device or account upon leaving China again as these details are likely collected and stored. Using this device outside of China poses the same risk as taking your personal devices and accounts into China.

QI-ANXIN produces a VPN that is capable of being hosted on Windows, Mac, Linux operating systems, and Apple or Android phones. The QI-ANXIN VPN harvests all available network information on Apple and Android phones, including SIM, MAC Addresses, IMEI, IMSI, Operating Systems information and telephone network information. The VPN software also collects previous network interface information on the device.

Equipped with current and historical device information available to the VPN provider, it is possible to easily identify the user with limited privacy. The VPN's software design has facilitated all the required information on the user's device information and historical network information, which could be provided to Chinese authorities, if requested, under the country's national security laws. Hypothetically, if this information was also combined with telecommunications metadata records, it would be possible to correlate both telecommunications metadata and the information provided by the VPN software to gain a complete picture of the users' internet usage history, network history and location history. We note that the network architecture crossover for the VPN between Legendsec, QI-ANXIN and 360.net is an integrated network design, which can be seen in the VPN Log file uploads between Android and IOS mobile applications. This is problematic, as Qihoo 360 (with the same branding as 360.net) was placed on the Entities list for the Department of Commerce and is considered a separate company to QI-ANXIN.[3]

[3] https://2017-2021.commerce.gov/news/press-releases/2020/05/commerce-department-add-two-dozen-chinese-companies-ties-wmd-and.html

Who is QI-ANXIN?

QI-ANXIN is one of largest cyber security companies in the China domestic market. QI-ANXIN markets itself as the cybersecurity provider of choice for 90 percent of the central government, state-owned enterprises, and major banks.[4] QI-ANXIN also advertises branded subsidiaries on its website, key of which is 网神 'Net God' (domain name is legendsec.com), the primary operator of the VPN according to our metadata analysis. We note that QI-ANXIN is one of the key companies that operates and is heavily invested in China's National Cybersecurity Centre according to Dakota Cary writing for CSET at Georgetown. China's National Cybersecurity Centre serves the Chinese Communist Party's strategic interests and supports the People's Liberation Army (PLA) Strategic Support Force's hacking teams.[5] Qi Xiangdong is QI-ANXIN's major shareholder. Qi Xiangdong is also a major shareholder, and president and co-founder, of 360 Enterprise Security Technology (Beijing) Group Co. Ltd.[6]

奇安信
BEIJING 2022™
北 京 2 0 2 2 年 冬 奥 会 官 方 赞 助 商
Official Sponsor of the Olympic Winter Games Beijing 2022

Figure 1: QI-ANXIN logo.

[4] https://en.qianxin.com/news/detail/205
[5] https://cset.georgetown.edu/wp-content/uploads/CSET-Chinas-National-Cybersecurity-Center.pdf
[6] https://www.bloomberg.com/profile/person/6035447

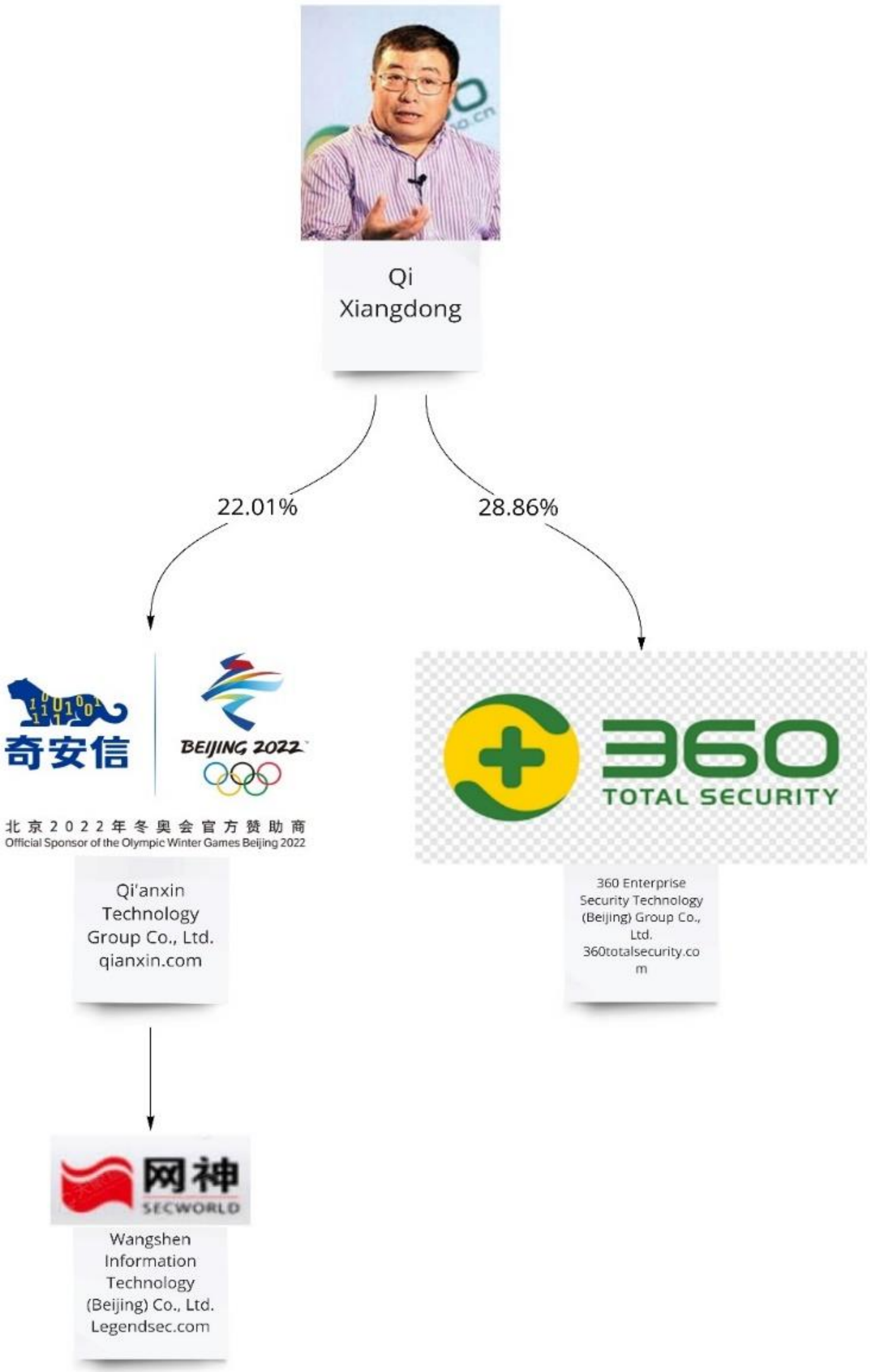

Qi
Xiangdong
22.01%
28.86%
奇安信
BEIJING 2022
北京2022年冬奥会官方赞助商
Official Sponsor of the Olympic Winter Games Beijing 2022
Qi'anxin Technology Group Co., Ltd. qianxin.com
360 TOTAL SECURITY
360 Enterprise Security Technology (Beijing) Group Co., Ltd. 360totalsecurity.com
网神 SECWORLD
Wangshen Information Technology (Beijing) Co., Ltd. Legendsec.com

Figure 2: Qi Xiandong's shareholding in QI-ANXIN and 360 Enterprise Security Technology (Beijing) Group Co., Ltd. As at 3 January 2022.

Multiple open-source reporting suggests that in May 2019 Qi Xiangdong separated his segment of the business from Qihoo 360 and Zhou Hongyi's. 360 Enterprise Security Technology (Beijing) Group Co. Ltd, which is owned by Qi Xiangdong, was now primarily operating under the name of QI-ANXIN.[7] This was hard to verify as we documented over 61 member companies in the QI-ANXIN Group and 13 member companies in the Qihoo 360 Group, not including 360 Enterprise Security Technology (Beijing) Group Co. Ltd. We could not verify if this is the complete operating structure and saw the complex overlapping company structure, website and branding between QI-ANXIN /360 Enterprise Security Technology (Beijing) Group and the Qihoo 360 Group as confusing to any retail investor. This does raise interesting questions for the United States Government concerning their policy towards close or previous associated entities of US Government sanctioned entities.

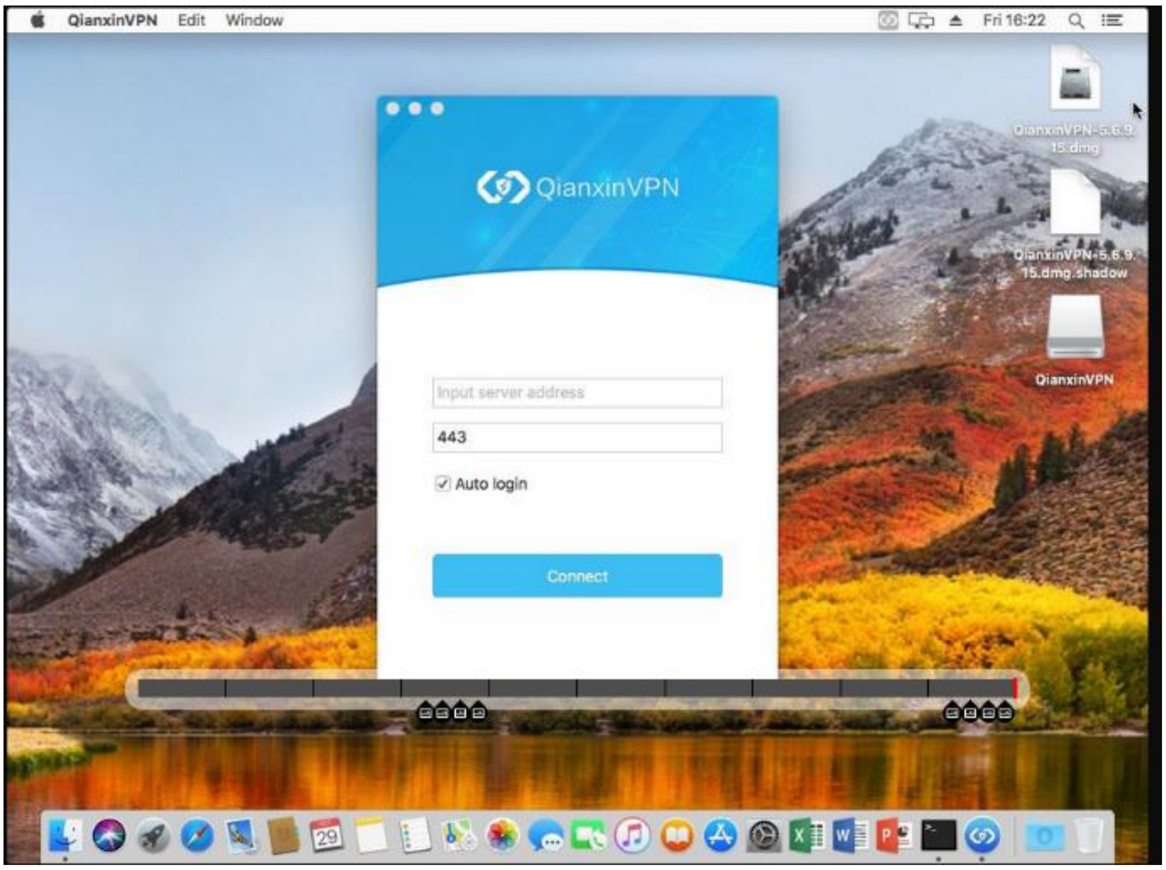
Qihoo 360 says they offer internet and mobile security products and services to over 400 million internet users.[8] We note that Qihoo 360 is recorded as participating in PLA cyber operations through training,[9] and was placed on the Entities list by the US Department of Commerce.[10]

It is important to note that the major shareholder Qi Xiangdong, as seen in Figure 2 above, is the primary shareholder in 360 Enterprise Security Technology (Beijing) Group Co. Ltd and not the higher Qihoo 360 entity. The operating agreement and split between Qihoo 360 and 360 Enterprise Security Technology (Beijing) Group Co. Ltd is not clear. They still share the same branding on their websites between 360.net and 360.cn and when Qihoo 360 is advertised on western sites such as Bloomberg Qi Xiangdong is included in the Qihoo 360 Group.[11] 360.net was registered in China as being connected in website to 360.cn, the filing date was 04 Aug 2020.[12] On balance we assess they are still related entities.

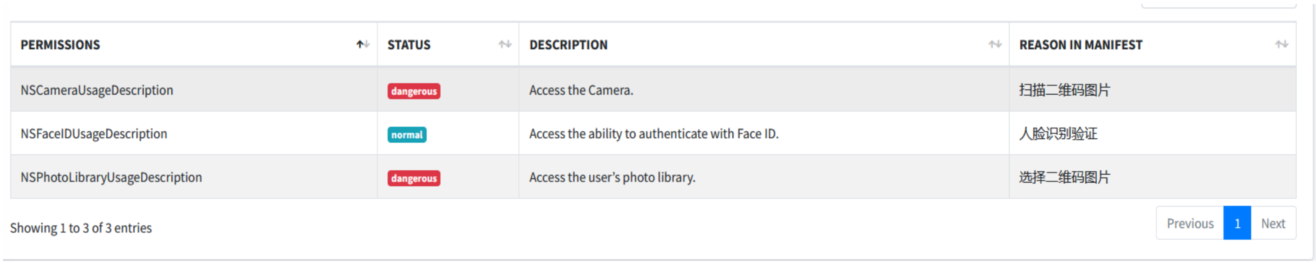
[7]https://baike.baidu.com/item/%E5%A5%87%E5%AE%89%E4%BF%A1%E7%A7%91%E6%8A%80%E9%9B%86%E5%9B%A2%E8%82%A1%E4%BB%BD%E6%9C%89%E9%99%90%E5%85%AC%E5%8F%B8/23546207#reference-[11]-24068412-wrap
[8] http://www.360.cn/about/englishversion.html
[9] https://cset.georgetown.edu/wp-content/uploads/CSET-Chinas-National-Cybersecurity-Center.pdf
[10] https://2017-2021.commerce.gov/news/press-releases/2020/05/commerce-department-add-two-dozen-chinese-companies-ties-wmd-and.html
[11] https://www.bloomberg.com/profile/company/QIHOZ:CH
[12] http://www.beian.gov.cn/portal/registerSystemInfo

QI-ANXIN Apple IOS VPN

The VPN applications for both Android and Apple are similar and the Apple version collects all network information that Android collects. While analysing the source code we noted that the Apple IOS VPN software allows access to the camera and photo library as standard permissions. These third-party application permissions for the camera and photo library are not functional for the use of VPN software.



Input server address
443
☑ Auto login



PERMISSIONS	STATUS	DESCRIPTION	REASON IN MANIFEST
NSCameraUsageDescription	dangerous	Access the Camera.	扫描二维码图片
NSFaceIDUsageDescription	normal	Access the ability to authenticate with Face ID.	人脸识别验证
NSPhotoLibraryUsageDescription	dangerous	Access the user's photo library.	选择二维码图片

Showing 1 to 3 of 3 entries

Previous 1 Next

Figure 3: IOS VPN Camera and Photo permissions.

Both IOS and Android software upload log files back to the VPN Provider. One major difference in this process was the Apple IOS software uploaded the log file to a QI-ANXIN domain and the Android software uploaded the log file to a 360.net domain running the Qihoo 360 branding. This is important to note as they are marketed as separate companies but have overlapping networks. 360 Enterprise Security Technology (Beijing) Group Co., Ltd would be considered a related entity as the major shareholder of both QI-ANXIN and 360 Enterprise Security Technology (Beijing) Group Co., Ltd is Qi Xiangdong. Figure 4 is a diagram separating the difference in log file uploads for the QI-ANXIN VPN across the available platforms.

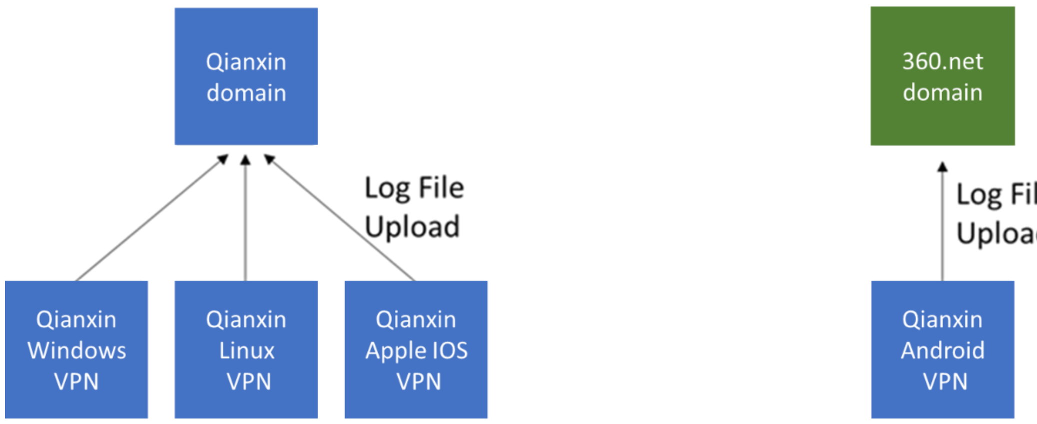
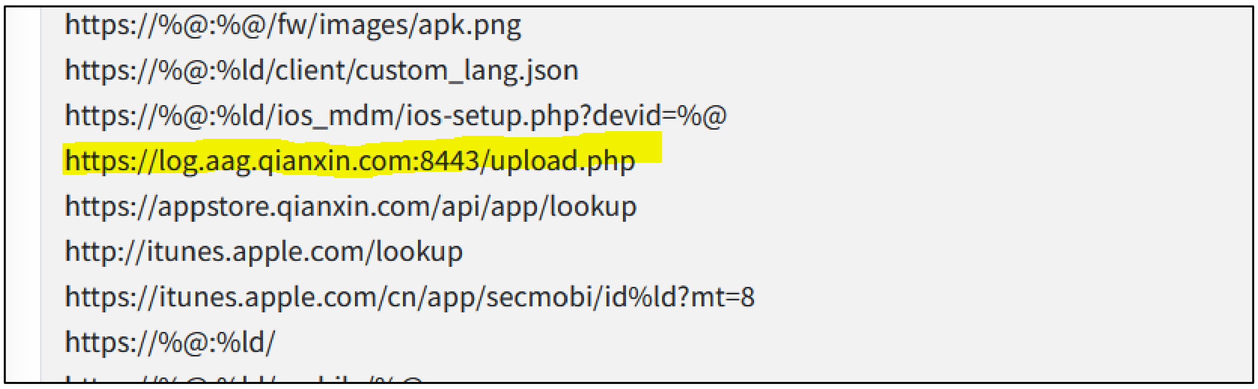


Figure 4: Log File Upload Diagram.

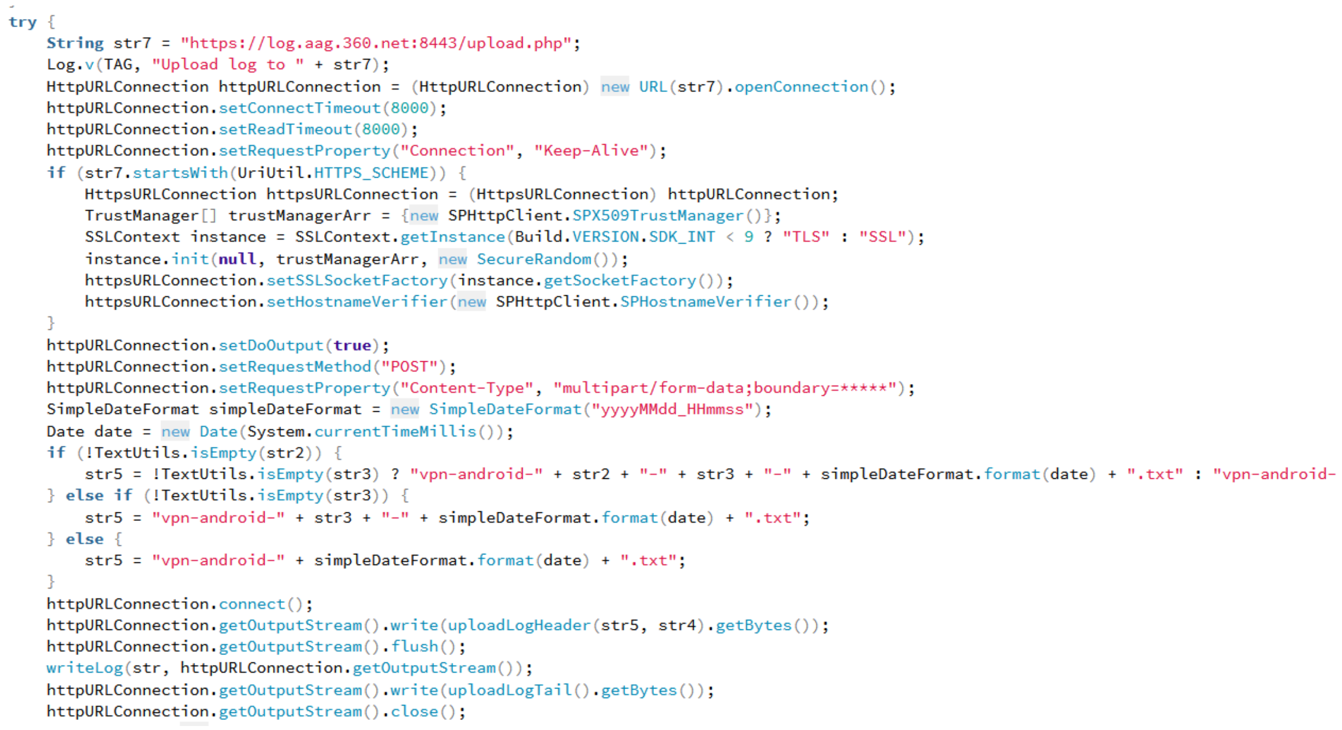
Figure 5 is the source code for the Apple IOS VPN showing the exact line of code where the log file is uploaded. As seen below the VPN log is uploaded to qianxin.com through port 8443.



```
https://%@:%@/fw/images/apk.png
https://%@:%ld/client/custom_lang.json
https://%@:%ld/ios_mdm/ios-setup.php?devid=%@
https://log.aag.qianxin.com:8443/upload.php
https://appstore.qianxin.com/api/app/lookup
http://itunes.apple.com/lookup
https://itunes.apple.com/cn/app/secmobi/id%ld?mt=8
https://%@:%ld/
```

Figure 5: Example of Apple IOS Log file upload.

Figure 6 below is the source code for the Android VPN showing the exact line of code where the log file is uploaded. As can be seen the Android VPN log is uploaded to 360.net through port 8443. 360.net is running a 360 Government and Enterprise Security Group webpage using the same logo branding depicted in Figure 1 and is also the same logo for Qihoo 360. According to Internet Whois registrant records 360.net is owned by Sun CHANG XIN of BeiJing Qi AnXin KeJi Co.,Ltd, address ChaoYangQu JiuXianQiao Lu 6 Hao Yuan 2 Hao Lou Beijing China. POC: its@360.net +86.1052448735.

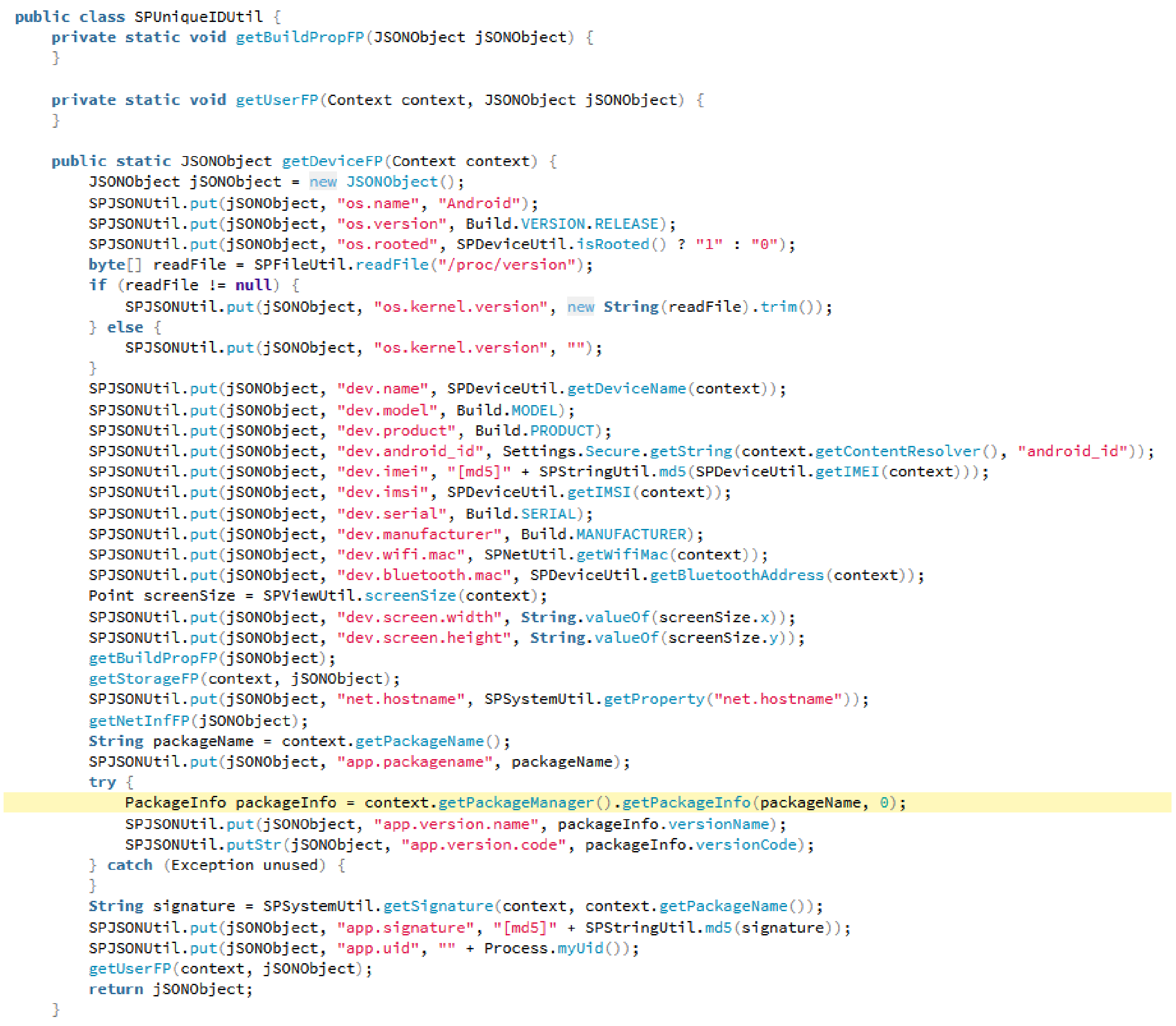


```
try {
    String str7 = "https://log.aag.360.net:8443/upload.php";
    Log.v(TAG, "Upload log to " + str7);
    HttpURLConnection httpURLConnection = (HttpURLConnection) new URL(str7).openConnection();
    httpURLConnection.setConnectTimeout(8000);
    httpURLConnection.setReadTimeout(8000);
    httpURLConnection.setRequestProperty("Connection", "Keep-Alive");
    if (str7.startsWith(UriUtil.HTTPS_SCHEME)) {
        HttpsURLConnection httpsURLConnection = (HttpsURLConnection) httpURLConnection;
        TrustManager[] trustManagerArr = {new SPHttpClient.SPX509TrustManager()};
        SSLContext instance = SSLContext.getInstance(Build.VERSION.SDK_INT < 9 ? "TLS" : "SSL");
        instance.init(null, trustManagerArr, new SecureRandom());
        httpsURLConnection.setSSLSocketFactory(instance.getSocketFactory());
        httpsURLConnection.setHostnameVerifier(new SPHttpClient.SPHostnameVerifier());
    }
    httpURLConnection.setDoOutput(true);
    httpURLConnection.setRequestMethod("POST");
    httpURLConnection.setRequestProperty("Content-Type", "multipart/form-data;boundary=*****");
    SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyyMMdd_HHmmss");
    Date date = new Date(System.currentTimeMillis());
    if (!TextUtils.isEmpty(str2)) {
        str5 = !TextUtils.isEmpty(str3) ? "vpn-android-" + str2 + "-" + str3 + "-" + simpleDateFormat.format(date) + ".txt" : "vpn-android-
    } else if (!TextUtils.isEmpty(str3)) {
        str5 = "vpn-android-" + str3 + "-" + simpleDateFormat.format(date) + ".txt";
    } else {
        str5 = "vpn-android-" + simpleDateFormat.format(date) + ".txt";
    }
    httpURLConnection.connect();
    httpURLConnection.getOutputStream().write(uploadLogHeader(str5, str4).getBytes());
    httpURLConnection.getOutputStream().flush();
    writeLog(str, httpURLConnection.getOutputStream());
    httpURLConnection.getOutputStream().write(uploadLogTail().getBytes());
    httpURLConnection.getOutputStream().close();
```

Figure 6: Example of Android Log file upload.

QI-ANXIN Android VPN

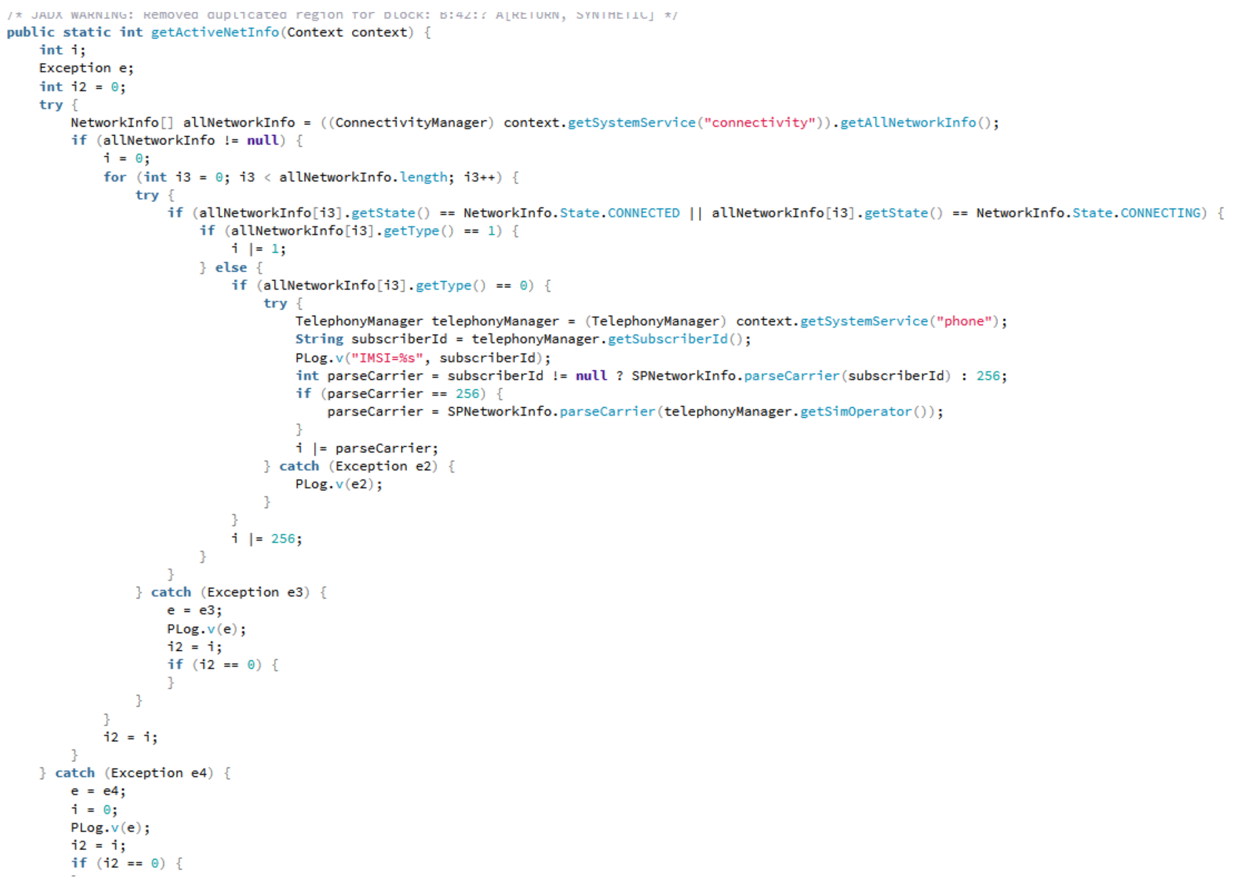
The following data are only specific to the Android version of the VPN software provided by QI-ANXIN. As seen below in Figure 7, this VPN is grabbing a significant amount of information off this phone, including screen height and width, as well as Bluetooth MAC addresses.



```
public class SPUniqueIDUtil {
    private static void getBuildPropFP(JSONObject jSONObject) {
    }

    private static void getUserFP(Context context, JSONObject jSONObject) {
    }

    public static JSONObject getDeviceFP(Context context) {
        JSONObject jSONObject = new JSONObject();
        SPJSONUtil.put(jSONObject, "os.name", "Android");
        SPJSONUtil.put(jSONObject, "os.version", Build.VERSION.RELEASE);
        SPJSONUtil.put(jSONObject, "os.rooted", SPDeviceUtil.isRooted() ? "1" : "0");
        byte[] readFile = SPFileUtil.readFile("/proc/version");
        if (readFile != null) {
            SPJSONUtil.put(jSONObject, "os.kernel.version", new String(readFile).trim());
        } else {
            SPJSONUtil.put(jSONObject, "os.kernel.version", "");
        }
        SPJSONUtil.put(jSONObject, "dev.name", SPDeviceUtil.getDeviceName(context));
        SPJSONUtil.put(jSONObject, "dev.model", Build.MODEL);
        SPJSONUtil.put(jSONObject, "dev.product", Build.PRODUCT);
        SPJSONUtil.put(jSONObject, "dev.android_id", Settings.Secure.getString(context.getContentResolver(), "android_id"));
        SPJSONUtil.put(jSONObject, "dev.imei", "[md5]" + SPStringUtil.md5(SPDeviceUtil.getIMEI(context)));
        SPJSONUtil.put(jSONObject, "dev.imsi", SPDeviceUtil.getIMSI(context));
        SPJSONUtil.put(jSONObject, "dev.serial", Build.SERIAL);
        SPJSONUtil.put(jSONObject, "dev.manufacturer", Build.MANUFACTURER);
        SPJSONUtil.put(jSONObject, "dev.wifi.mac", SPNetUtil.getWifiMac(context));
        SPJSONUtil.put(jSONObject, "dev.bluetooth.mac", SPDeviceUtil.getBluetoothAddress(context));
        Point screenSize = SPViewUtil.screenSize(context);
        SPJSONUtil.put(jSONObject, "dev.screen.width", String.valueOf(screenSize.x));
        SPJSONUtil.put(jSONObject, "dev.screen.height", String.valueOf(screenSize.y));
        getBuildPropFP(jSONObject);
        getStorageFP(context, jSONObject);
        SPJSONUtil.put(jSONObject, "net.hostname", SPSystemUtil.getProperty("net.hostname"));
        getNetInfFP(jSONObject);
        String packageName = context.getPackageName();
        SPJSONUtil.put(jSONObject, "app.packagename", packageName);
        try {
            PackageInfo packageInfo = context.getPackageManager().getPackageInfo(packageName, 0);
            SPJSONUtil.put(jSONObject, "app.version.name", packageInfo.versionName);
            SPJSONUtil.putStr(jSONObject, "app.version.code", packageInfo.versionCode);
        } catch (Exception unused) {
        }
        String signature = SPSystemUtil.getSignature(context, context.getPackageName());
        SPJSONUtil.put(jSONObject, "app.signature", "[md5]" + SPStringUtil.md5(signature));
        SPJSONUtil.put(jSONObject, "app.uid", "" + Process.myUid());
        getUserFP(context, jSONObject);
        return jSONObject;
    }
```

Figure 7: Android VPN device information gathering.

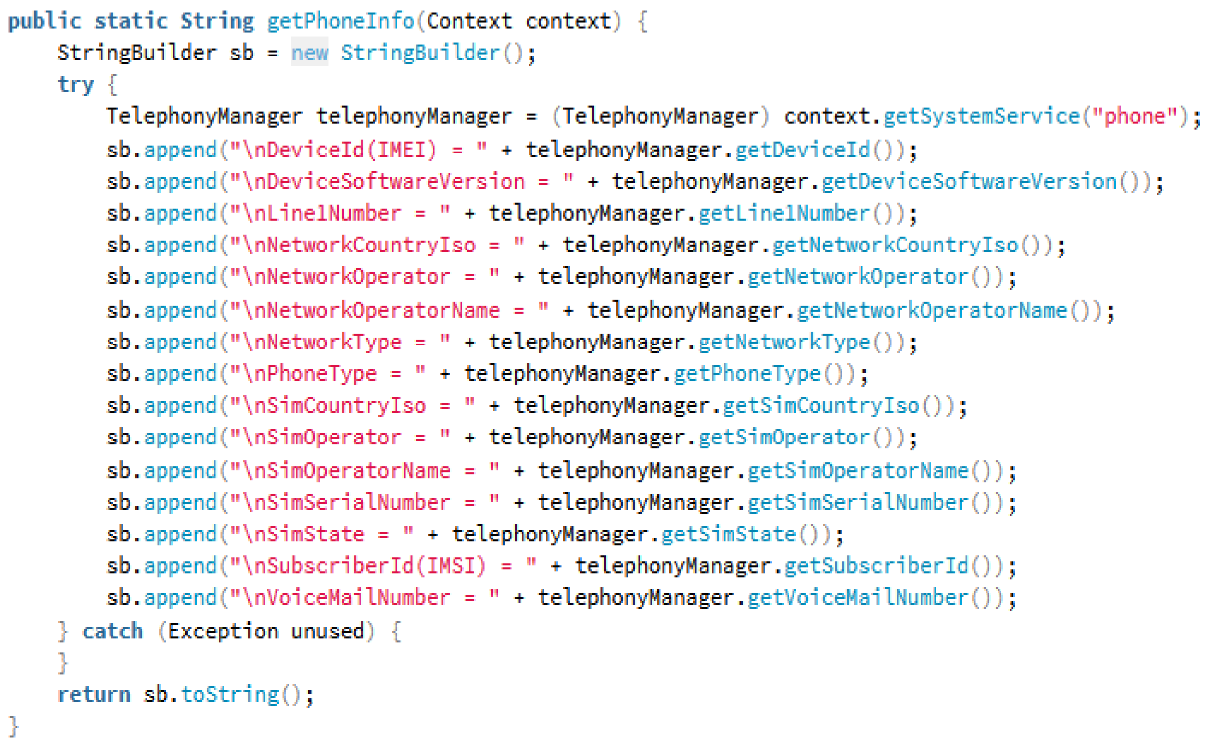
Figure 8 below is an example of the source code where the software records all network information on all previously connected network interfaces. For example, if you have three network interfaces on your phone that your phone has connected to, the above for-loop will run three times to gather each interface IMSI, and carrier information. This is not a functional process as previous network information is not standard or needed for a VPN to work.



```
/* JADX WARNING: Removed duplicated region for block: B:42:? A[RETURN, SYNTHETIC] */
public static int getActiveNetInfo(Context context) {
    int i;
    Exception e;
    int i2 = 0;
    try {
        NetworkInfo[] allNetworkInfo = ((ConnectivityManager) context.getSystemService("connectivity")).getAllNetworkInfo();
        if (allNetworkInfo != null) {
            i = 0;
            for (int i3 = 0; i3 < allNetworkInfo.length; i3++) {
                try {
                    if (allNetworkInfo[i3].getState() == NetworkInfo.State.CONNECTED || allNetworkInfo[i3].getState() == NetworkInfo.State.CONNECTING) {
                        if (allNetworkInfo[i3].getType() == 1) {
                            i |= 1;
                        } else {
                            if (allNetworkInfo[i3].getType() == 0) {
                                try {
                                    TelephonyManager telephonyManager = (TelephonyManager) context.getSystemService("phone");
                                    String subscriberId = telephonyManager.getSubscriberId();
                                    PLog.v("IMSI=%s", subscriberId);
                                    int parseCarrier = subscriberId != null ? SPNetworkInfo.parseCarrier(subscriberId) : 256;
                                    if (parseCarrier == 256) {
                                        parseCarrier = SPNetworkInfo.parseCarrier(telephonyManager.getSimOperator());
                                    }
                                    i |= parseCarrier;
                                } catch (Exception e2) {
                                    PLog.v(e2);
                                }
                            }
                            i |= 256;
                        }
                    }
                } catch (Exception e3) {
                    e = e3;
                    PLog.v(e);
                    i2 = i;
                    if (i2 == 0) {
                    }
                }
            }
            i2 = i;
        }
    } catch (Exception e4) {
        e = e4;
        i = 0;
        PLog.v(e);
        i2 = i;
        if (i2 == 0) {
```

Figure 8: Android VPN information gathering previous network connections.

The VPN absorbs even more information into its system by grabbing as much information on the hosting device as possible, including, but not limited to, IMEI, system software version, phone number, network information (provider, country, etc.), SIM information, seen in red in Figure 9.

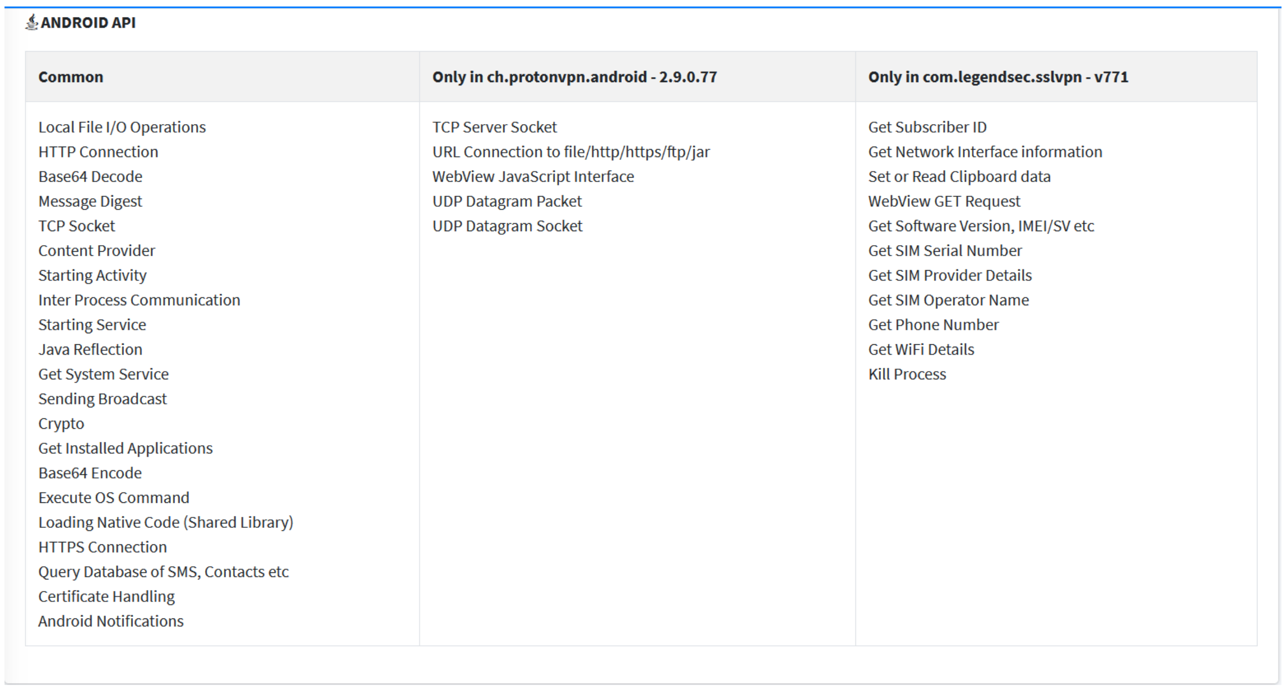


```
public static String getPhoneInfo(Context context) {
    StringBuilder sb = new StringBuilder();
    try {
        TelephonyManager telephonyManager = (TelephonyManager) context.getSystemService("phone");
        sb.append("\nDeviceId(IMEI) = " + telephonyManager.getDeviceId());
        sb.append("\nDeviceSoftwareVersion = " + telephonyManager.getDeviceSoftwareVersion());
        sb.append("\nLine1Number = " + telephonyManager.getLine1Number());
        sb.append("\nNetworkCountryIso = " + telephonyManager.getNetworkCountryIso());
        sb.append("\nNetworkOperator = " + telephonyManager.getNetworkOperator());
        sb.append("\nNetworkOperatorName = " + telephonyManager.getNetworkOperatorName());
        sb.append("\nNetworkType = " + telephonyManager.getNetworkType());
        sb.append("\nPhoneType = " + telephonyManager.getPhoneType());
        sb.append("\nSimCountryIso = " + telephonyManager.getSimCountryIso());
        sb.append("\nSimOperator = " + telephonyManager.getSimOperator());
        sb.append("\nSimOperatorName = " + telephonyManager.getSimOperatorName());
        sb.append("\nSimSerialNumber = " + telephonyManager.getSimSerialNumber());
        sb.append("\nSimState = " + telephonyManager.getSimState());
        sb.append("\nSubscriberId(IMSI) = " + telephonyManager.getSubscriberId());
        sb.append("\nVoiceMailNumber = " + telephonyManager.getVoiceMailNumber());
    } catch (Exception unused) {
    }
    return sb.toString();
}
```

Figure 9: Android VPN information gathering network connections.

It is important to compare two VPNs from different distributors to show that this level of data collection from QI-ANXIN VPN is not normal for a VPN. We used ProtonVPN as the secondary VPN for comparison due to its reputation in the market as a privacy first VPN based in Switzerland, which is underpinned by Switzerland's robust privacy and data laws.[13] Figures 10 and 11 (on the next page) compare the software's commands depicting the difference in data collected between the two VPNs. QI-ANXIN collects a lot more data that is personal to the user than Proton VPN does.

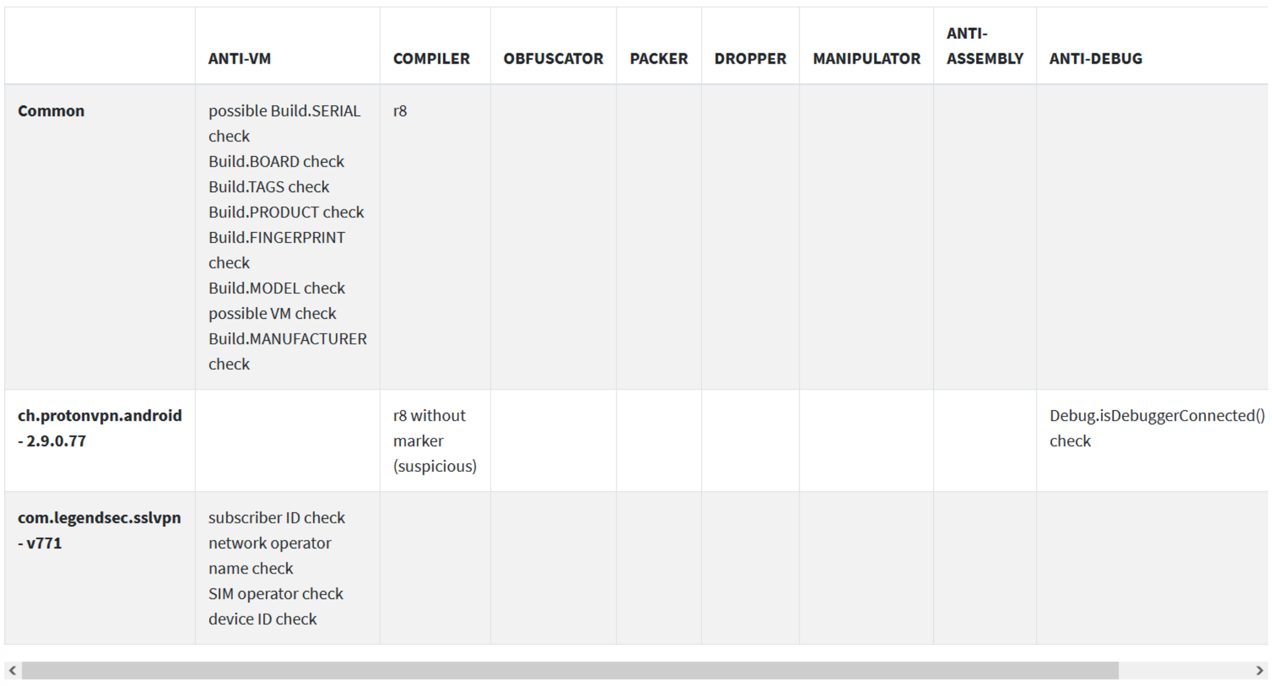
[13] https://www.dataguidance.com/notes/switzerland-data-protection-overview



ANDROID API

Common	Only in ch.protonvpn.android - 2.9.0.77	Only in com.legendsec.sslvpn - v771
Local File I/O Operations HTTP Connection Base64 Decode Message Digest TCP Socket Content Provider Starting Activity Inter Process Communication Starting Service Java Reflection Get System Service Sending Broadcast Crypto Get Installed Applications Base64 Encode Execute OS Command Loading Native Code (Shared Library) HTTPS Connection Query Database of SMS, Contacts etc Certificate Handling Android Notifications	TCP Server Socket URL Connection to file/http/https/ftp/jar WebView JavaScript Interface UDP Datagram Packet UDP Datagram Socket	Get Subscriber ID Get Network Interface information Set or Read Clipboard data WebView GET Request Get Software Version, IMEI/SV etc Get SIM Serial Number Get SIM Provider Details Get SIM Operator Name Get Phone Number Get WiFi Details Kill Process

Figure 10: Comparison of Application Programming Interface calls between ProtonVPN and QI-ANXIN VPN



	ANTI-VM	COMPILER	OBFUSCATOR	PACKER	DROPPER	MANIPULATOR	ANTI-ASSEMBLY	ANTI-DEBUG
Common	possible Build.SERIAL check Build.BOARD check Build.TAGS check Build.PRODUCT check Build.FINGERPRINT check Build.MODEL check possible VM check Build.MANUFACTURER check	r8						
ch.protonvpn.android - 2.9.0.77		r8 without marker (suspicious)						Debug.isDebuggerConnected() check
com.legendsec.sslvpn - v771	subscriber ID check network operator name check SIM operator check device ID check							

Figure 11: Comparison of Anti-Virtual Machine and Anti-Debug between ProtonVPN and QI-ANXIN VPN.

Part 2 KINGSOFT

Executive Summary

Part 2 is a technical analysis of Beijing Kingsoft Office Software Co., Ltd's (Kingsoft) anti-virus software product. Kingsoft through the "WPS Office" suite of software is an Official Supplier to the Beijing 2022 Olympic and Paralympic Winter Games.[14] This analysis is based on Kingsoft's publicly available anti-virus software product. On 5 January 2021 the Whitehouse released Executive Order 13873 effectively banning the use of or transaction with WPS Office.[15] As seen in Figure 12 "WPS Office" is a product of the company Beijing Kingsoft Software Co. Ltd which runs the Office software. The "WPS Office" suite is a pillar of the software company Kingsoft.

The Internet 2.0 intelligence team found the anti-virus installer file flagged as potentially containing malicious behaviours or properties. The installer also runs a file that potentially accesses data from internet browsers that are also running on the user's desktop computer. This file potentially copies all browser cookies as well as personal information and credentials. In our opinion the use of this anti-virus carries risk on use as the software provider could attain access to the internet usage history of the user. In the Android version of the Office suite the team found data collection and upload functions from iciba.com to Kingsoft.com which included GPS location, MAC address, installed applications, the phone number and sim information of the device, operating system information and other features including screenshots and clipboard access.

Who is Kingsoft?

Kingsoft also known as Beijing Kingsoft Software Co Ltd or Zuhai Jinshan Software Co Ltd is a Chinese Technology company that has four main business lines according to their website.[16] These are Cheetah Mobile, Xishanju (entertainment and online games), Kingsoft Cloud and WPS Office Software. See figure 12 for a company structure overview. 邹涛 (Zou Tao) is the executive director and CEO for the conglomerate. He is also the listed legal representative or company director on all company branches. In total we found 52 entities that Zou Tao is either the legal representative or a director for. Zou Tao is the primary connecting figure that would make all these companies to be probably considered as related entities.

[14] https://www.beijing2022.cn/a/20200630/011345.htm
[15] https://trumpwhitehouse.archives.gov/presidential-actions/executive-order-addressing-threat-posed-applications-software-developed-controlled-chinese-companies/
[16] https://www.baike.com/wikiid/4264469297351337403?from=wiki_content&prd=innerlink&view_id=5olg7tk4bm4000

邹涛 Zou Tao legal Representative for:

- Chengdu Xishanju Interactive Entertainment Technology Co., Ltd. 成都西山居互动娱乐科技有限公司
- Beijing Kingsoft Digital Entertainment Technology Co., Ltd. 北京金山数字娱乐科技有限公司
- Beijing Kingsoft Office Software Co., Ltd 北京金山办公软件股份有限公司

and company director for Cheetah which is run by 傅盛 Fu Sheng

- Beijing Cheetah Mobile Technology Co., Ltd. 北京猎豹移动科技有限公司
- Beijing Kingsoft Security Software Co., Ltd 北京金山安全软件有限公司

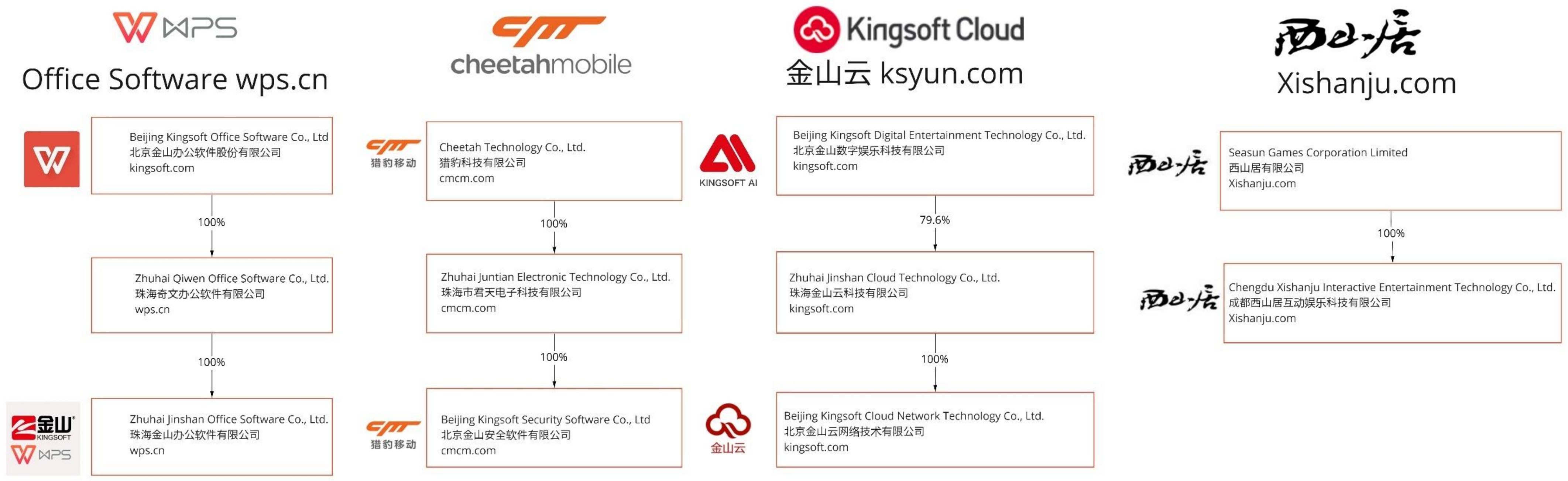
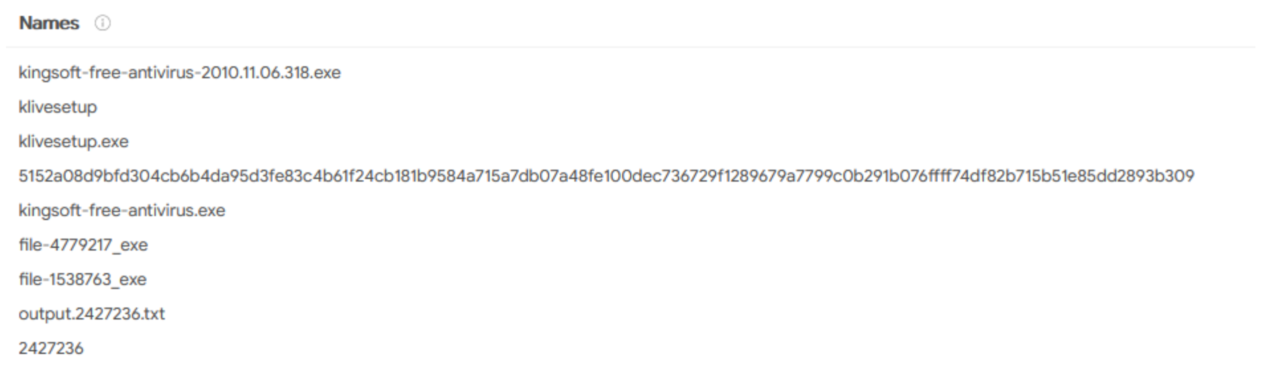

Office Software wps.cn
Beijing Kingsoft Office Software Co., Ltd 北京金山办公软件股份有限公司 kingsoft.com
100%
Zhuhai Qiwen Office Software Co., Ltd. 珠海奇文办公软件有限公司 wps.cn
100%
Zhuhai Jinshan Office Software Co., Ltd. 珠海金山办公软件有限公司 wps.cn
cheetahmobile
猎豹移动
Cheetah Technology Co., Ltd. 猎豹科技有限公司 cmcm.com
100%
Zhuhai Juntian Electronic Technology Co., Ltd. 珠海市君天电子科技有限公司 cmcm.com
100%
Beijing Kingsoft Security Software Co., Ltd 北京金山安全软件有限公司 cmcm.com
Kingsoft Cloud
金山云 ksyun.com
KINGSOFT AI
Beijing Kingsoft Digital Entertainment Technology Co., Ltd. 北京金山数字娱乐科技有限公司 kingsoft.com
79.6%
Zhuhai Jinshan Cloud Technology Co., Ltd. 珠海金山云科技有限公司 kingsoft.com
100%
金山云
Beijing Kingsoft Cloud Network Technology Co., Ltd. 北京金山云网络技术有限公司 kingsoft.com
西山居
Xishanju.com
Seasun Games Corporation Limited 西山居有限公司 Xishanju.com
100%
Chengdu Xishanju Interactive Entertainment Technology Co., Ltd. 成都西山居互动娱乐科技有限公司 Xishanju.com

Figure 12: Kingsoft company structure.

Kingsoft Anti-virus

For thoroughness the antivirus software was copyrighted 1998-2010 Kingsoft Corporation and signed by the Zhuhai Kingsoft Office Software Co. Ltd. For Kingsoft antivirus creator the names Kingsoft and Jinshan are interchangeable as seen on Figure 12. The software has multiple http connections to kingsoftsecurity.com and info.duba.net both domains associated with Beijing Kingsoft Security Software Co., Ltd.[17] When the software installer is downloaded the file has multiple different names as seen on VirusTotal. The antivirus installer is named "kingsoft-free-antivirus-*.*.*.*.exe". These names are seen in Figure 13 below.[18] On analysis there are a total of 2,809 files that come compiled inside of the installer.

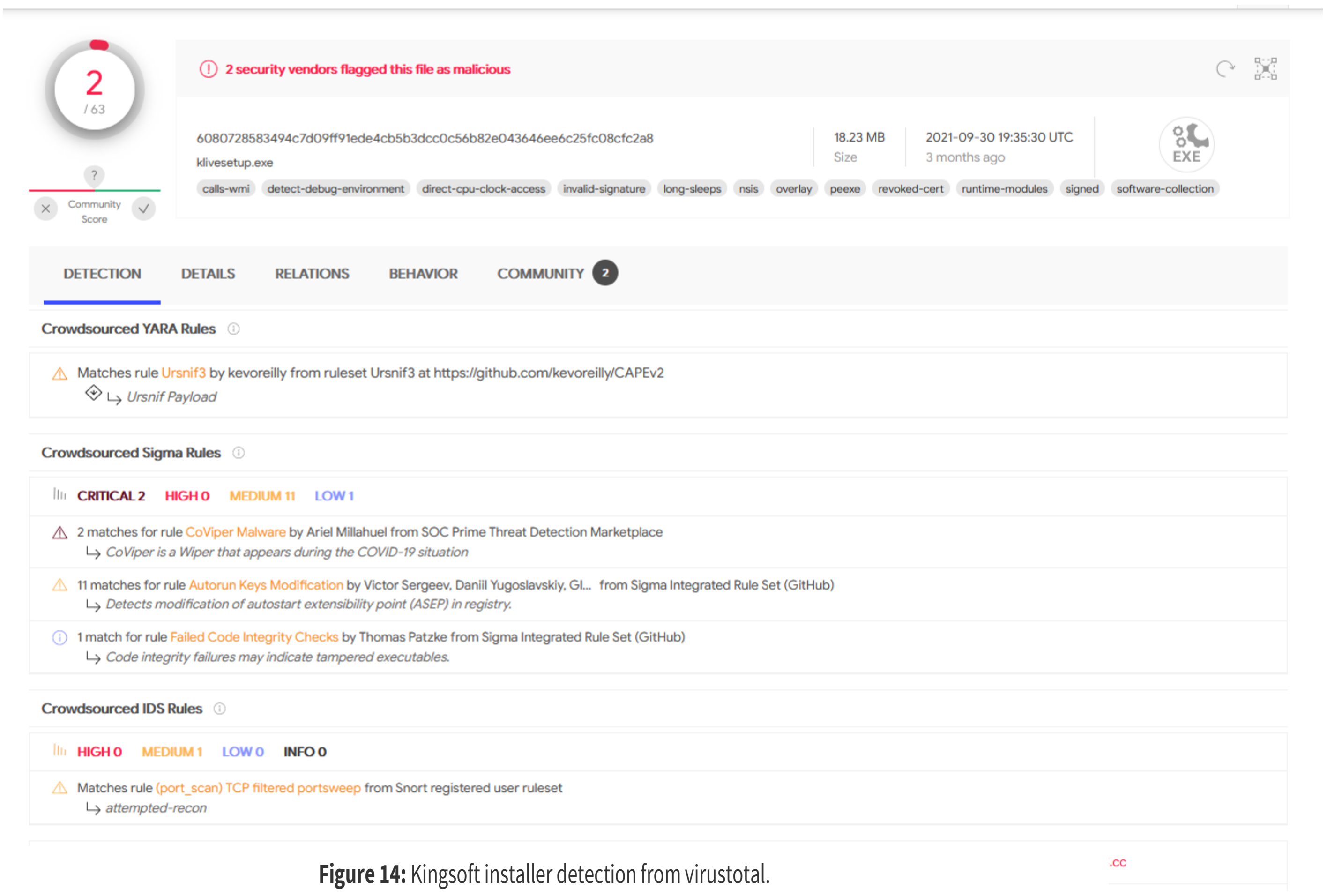

Names

kingsoft-free-antivirus-2010.11.06.318.exe
klivesetup
klivesetup.exe
5152a08d9bfd304cb6b4da95d3fe83c4b61f24cb181b9584a715a7db07a48fe100dec736729f1289679a7799c0b291b076ffff74df82b715b51e85dd2893b309
kingsoft-free-antivirus.exe
file-4779217_exe
file-1538763_exe
output.2427236.txt
2427236

Figure 13: Kingsoft installer file names

Importantly, according to VirusTotal, the installer is detected by multiple Sigma rules as being CoViper malware, autorun keys modifications, code integrity check failure, port sweeping, as well as being detected by a Yara rule as having a Ursnif3 payload embedded inside of one of the packed files. It is also detected by two antivirus software as a "Trojan" and "Malware".[19] This is seen in Figure 14. VirusTotal is a community driven security platform that allows security professionals and companies to input flags for malicious files, IP addresses and websites. As it is a community driven platform, we cannot assess the veracity of these malicious flags for the installer by the members. However, as named and referenced researchers of the VirusTotal community have placed flags and their names to these citations, we would flag that multiple professionals have flagged the file as having malicious behaviours or properties and that this is definitely an indicator of probable risk on use.

[17] https://www.virustotal.com/gui/file/6080728583494c7d09ff91ede4cb5b3dcc0c56b82e043646ee6c25fc08cfc2a8/behavior https://www.virustotal.com/gui/file/6080728583494c7d09ff91ede4cb5b3dcc0c56b82e043646ee6c25fc08cfc2a8/behavior
[18] https://www.virustotal.com/gui/file/6080728583494c7d09ff91ede4cb5b3dcc0c56b82e043646ee6c25fc08cfc2a8/details
[19] https://www.virustotal.com/gui/file/6080728583494c7d09ff91ede4cb5b3dcc0c56b82e043646ee6c25fc08cfc2a8



2
/ 63

? Community Score

2 security vendors flagged this file as malicious

6080728583494c7d09ff91ede4cb5b3dcc0c56b82e043646ee6c25fc08cfc2a8
klivesetup.exe

18.23 MB
Size

2021-09-30 19:35:30 UTC
3 months ago

EXE

calls-wmi detect-debug-environment direct-cpu-clock-access invalid-signature long-sleeps nsis overlay peexe revoked-cert runtime-modules signed software-collection

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 2

Crowdsourced YARA Rules

Matches rule Ursnif3 by kevoreilly from ruleset Ursnif3 at https://github.com/kevoreilly/CAPEv2
↳ *Ursnif Payload*

Crowdsourced Sigma Rules

CRITICAL 2 HIGH 0 MEDIUM 11 LOW 1

2 matches for rule CoViper Malware by Ariel Millahuel from SOC Prime Threat Detection Marketplace
↳ *CoViper is a Wiper that appears during the COVID-19 situation*

11 matches for rule Autorun Keys Modification by Victor Sergeev, Daniil Yugoslavskiy, Gl... from Sigma Integrated Rule Set (GitHub)
↳ *Detects modification of autostart extensibility point (ASEP) in registry.*

1 match for rule Failed Code Integrity Checks by Thomas Patzke from Sigma Integrated Rule Set (GitHub)
↳ *Code integrity failures may indicate tampered executables.*

Crowdsourced IDS Rules

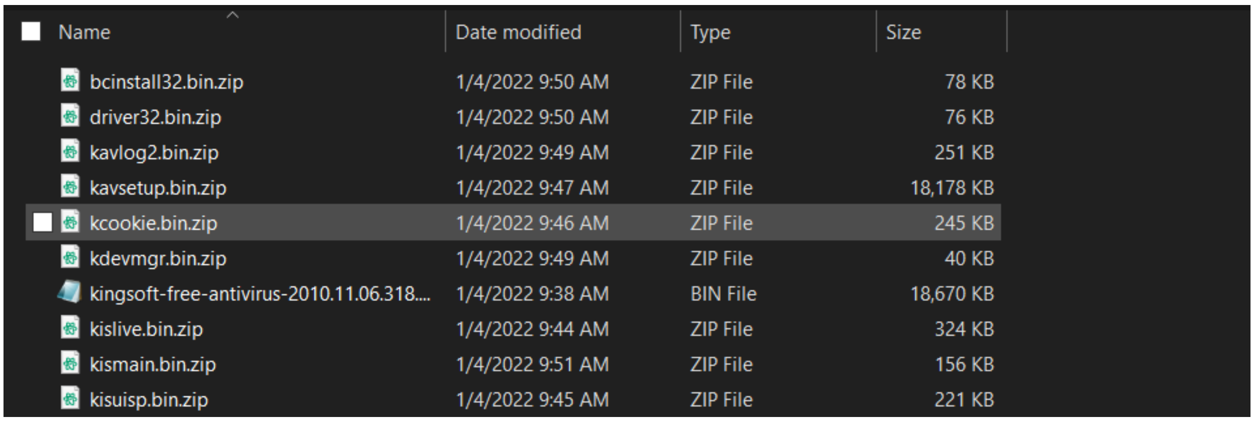
HIGH 0 MEDIUM 1 LOW 0 INFO 0

Matches rule (port_scan) TCP filtered portsweep from Snort registered user ruleset
↳ *attempted-recon*

.cc

Figure 14: Kingsoft installer detection from virustotal.

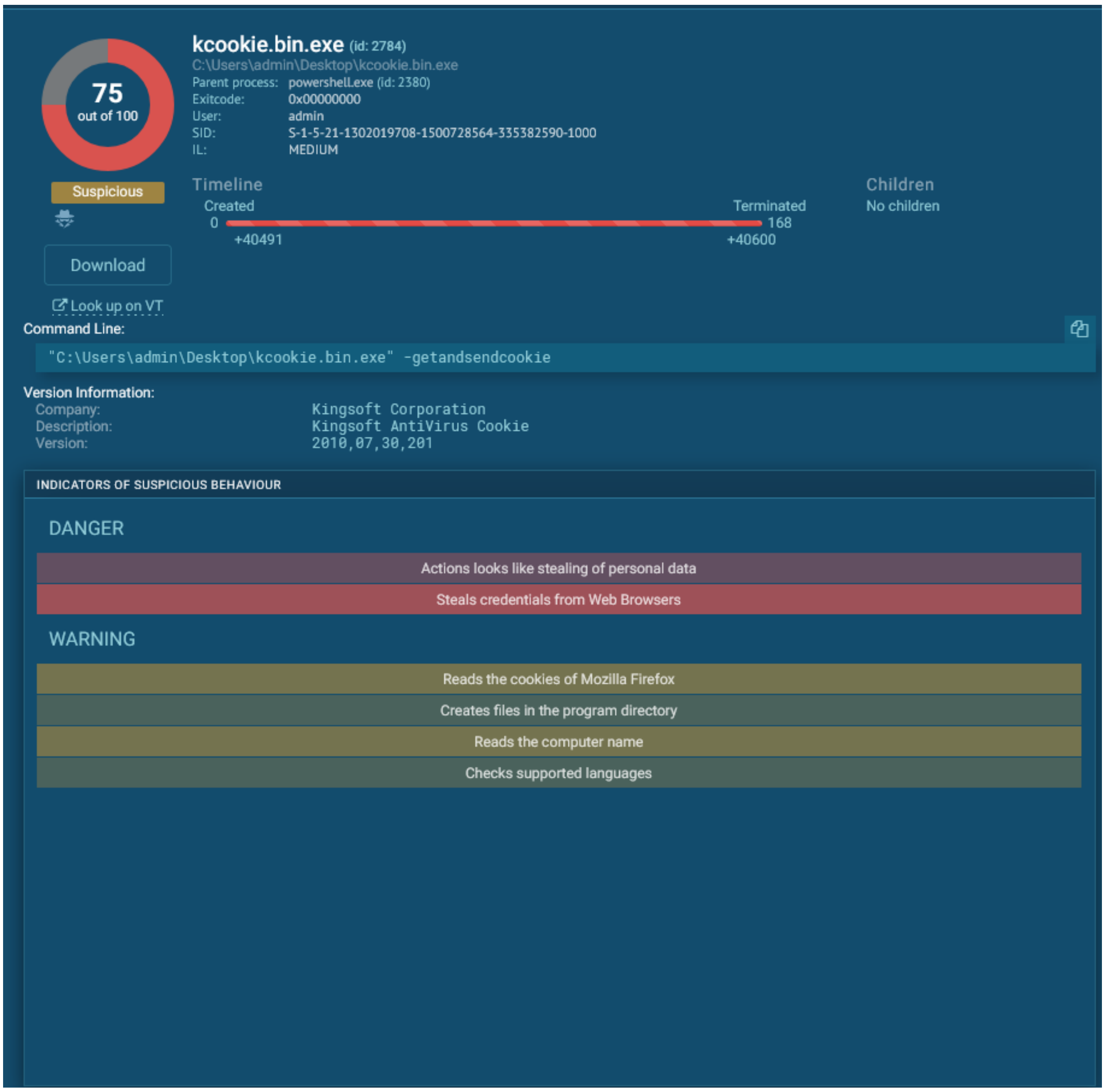
As the installer runs, we found a file that we saw risk with. This was kcookie.bin.exe. This file was 245 KB in size and comes with the installer. See Figure 15.



Name	Date modified	Type	Size
bcinstall32.bin.zip	1/4/2022 9:50 AM	ZIP File	78 KB
driver32.bin.zip	1/4/2022 9:50 AM	ZIP File	76 KB
kavlog2.bin.zip	1/4/2022 9:49 AM	ZIP File	251 KB
kavsetup.bin.zip	1/4/2022 9:47 AM	ZIP File	18,178 KB
kcookie.bin.zip	1/4/2022 9:46 AM	ZIP File	245 KB
kdevmgr.bin.zip	1/4/2022 9:49 AM	ZIP File	40 KB
kingsoft-free-antivirus-2010.11.06.318....	1/4/2022 9:38 AM	BIN File	18,670 KB
kislive.bin.zip	1/4/2022 9:44 AM	ZIP File	324 KB
kismain.bin.zip	1/4/2022 9:51 AM	ZIP File	156 KB
kisuisp.bin.zip	1/4/2022 9:45 AM	ZIP File	221 KB

Figure 15: Kingsoft installer files

As we ran the executable file through an online sandbox (any.app.run) it flagged as a 75 per cent risk, as it appears to access personal information and takes credentials and cookies from the internet browser that the user is running. There is also a command line argument in the source code that states “getandsendcookies”. See Figure 16 for a detailed view of the file from the sandbox.



kcookie.bin.exe (id: 2784)
C:\Users\admin\Desktop\kcookie.bin.exe

Parent process:	powershell.exe (id: 2380)
Exitcode:	0x00000000
User:	admin
SID:	S-1-5-21-1302019708-1500728564-335382590-1000
IL:	MEDIUM

Timeline
Created 0 +40491
Terminated 168 +40600

Children
No children

Command Line:

```
"C:\Users\admin\Desktop\kcookie.bin.exe" -getandsendcookie
```

Version Information:

Company:	Kingsoft Corporation
Description:	Kingsoft AntiVirus Cookie
Version:	2010,07,30,201

INDICATORS OF SUSPICIOUS BEHAVIOUR

DANGER

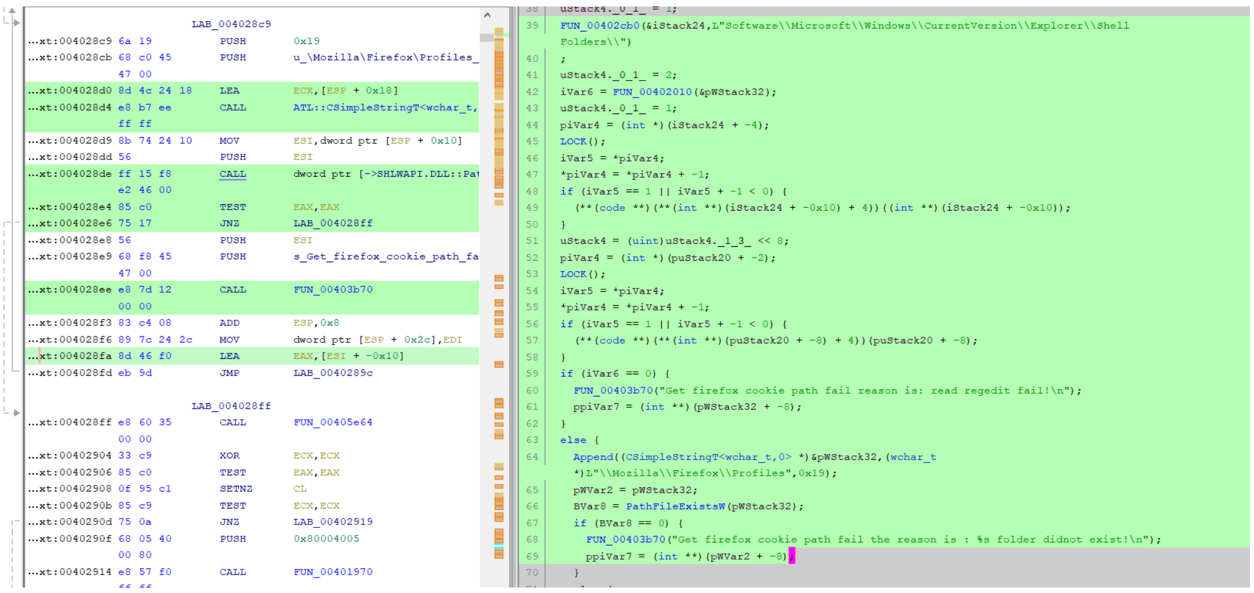
- Actions looks like stealing of personal data
- Steals credentials from Web Browsers

WARNING

- Reads the cookies of Mozilla Firefox
- Creates files in the program directory
- Reads the computer name
- Checks supported languages

Figure 16: kcookie.exe taken from the Kingsoft AV installer detected on the sandbox.

As the installation continued it appeared that “kcookie.exe” is responsible for accessing the internet browsers information, as you can see in the following image “kcookie.exe” takes cookies out of the Firefox browsers SQLite file. See Figure 17 for this command.



```
                        LAB_004028c9
...xt:004028c9 6a 19            PUSH       0x19
...xt:004028cb 68 c0 45         PUSH       u_\Mozilla\Firefox\Profiles_
               47 00
...xt:004028d0 8d 4c 24 18      LEA        ECX,[ESP + 0x18]
...xt:004028d4 e8 b7 ee         CALL       ATL::CSimpleStringT<wchar_t,
               ff ff
...xt:004028d9 8b 74 24 10      MOV        ESI,dword ptr [ESP + 0x10]
...xt:004028dd 56               PUSH       ESI
...xt:004028de ff 15 f8         CALL       dword ptr [->SHLWAPI.DLL::Pat
               e2 46 00
...xt:004028e4 85 c0            TEST       EAX,EAX
...xt:004028e6 75 17            JNZ        LAB_004028ff
...xt:004028e8 56               PUSH       ESI
...xt:004028e9 68 f8 45         PUSH       s_Get_firefox_cookie_path_fa
               47 00
...xt:004028ee e8 7d 12         CALL       FUN_00403b70
               00 00
...xt:004028f3 83 c4 08         ADD        ESP,0x8
...xt:004028f6 89 7c 24 2c      MOV        dword ptr [ESP + 0x2c],EDI
...xt:004028fa 8d 46 f0         LEA        EAX,[ESI + -0x10]
...xt:004028fd eb 9d            JMP        LAB_0040289c

                        LAB_004028ff
...xt:004028ff e8 60 35         CALL       FUN_00405e64
               00 00
...xt:00402904 33 c9            XOR        ECX,ECX
...xt:00402906 85 c0            TEST       EAX,EAX
...xt:00402908 0f 95 c1         SETNZ      CL
...xt:0040290b 85 c9            TEST       ECX,ECX
...xt:0040290d 75 0a            JNZ        LAB_00402919
...xt:0040290f 68 05 40         PUSH       0x80004005
               00 80
...xt:00402914 e8 57 f0         CALL       FUN_00401970
```

```
uStack4._0_1_ = 1;
FUN_00402cb0(&iStack24,L"Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Shell
Folders\\")
;
uStack4._0_1_ = 2;
iVar6 = FUN_00402010(&pWStack32);
uStack4._0_1_ = 1;
piVar4 = (int *)(iStack24 + -4);
LOCK();
iVar5 = *piVar4;
*piVar4 = *piVar4 + -1;
if (iVar5 == 1 || iVar5 + -1 < 0) {
  (**(code **)(**(int **)(iStack24 + -0x10) + 4))((int **)(iStack24 + -0x10));
}
uStack4 = (uint)uStack4._1_3_ << 8;
piVar4 = (int *)(puStack20 + -2);
LOCK();
iVar5 = *piVar4;
*piVar4 = *piVar4 + -1;
if (iVar5 == 1 || iVar5 + -1 < 0) {
  (**(code **)(**(int **)(puStack20 + -8) + 4))(puStack20 + -8);
}
if (iVar6 == 0) {
  FUN_00403b70("Get firefox cookie path fail reason is: read regedit fail!\n");
  ppiVar7 = (int **)(pWStack32 + -8);
}
else {
  Append((CSimpleStringT<wchar_t,0> *)&pWStack32,(wchar_t
  *)L"\\Mozilla\\Firefox\\Profiles",0x19);
  pWVar2 = pWStack32;
  BVar8 = PathFileExistsW(pWStack32);
  if (BVar8 == 0) {
    FUN_00403b70("Get firefox cookie path fail the reason is : %s folder didnot exist!\n");
    ppiVar7 = (int **)(pWVar2 + -8);
  }
```

Figure 17: kcookie.exe source code.

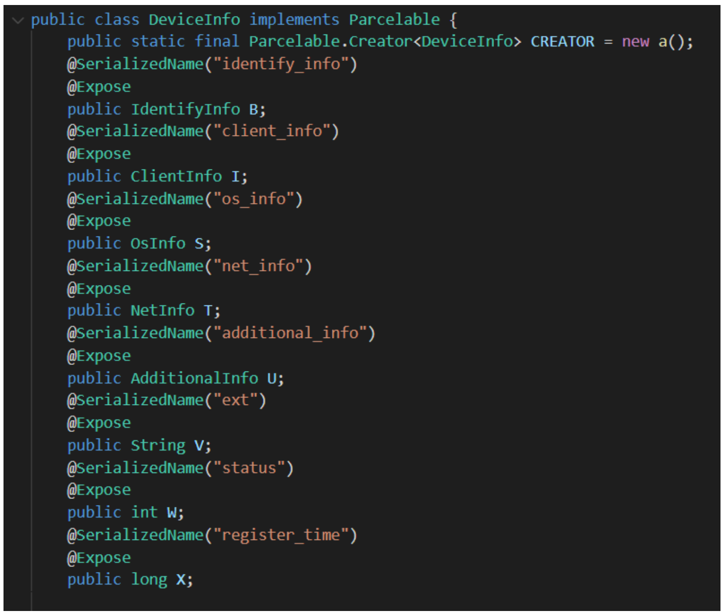
WPS Office

The WPS Office software was copyrighted 2021 Kingsoft Corporation which is run by Beijing Kingsoft Office Software Co., Ltd as seen in figure 12. The WPS Office software comes in four major packages which are very similar and is compatible with the Microsoft Office Software. The software is marketed as being free to download. The packages are titles Writer, Presentation, Spreadsheet and PDF files. The packages are usable across all major platforms including Windows, Mac, Android, IOS and Linux. They also have a cloud platform and templates store.[20] For the purpose of this analysis we will only refer to information pertaining to the android version of the software package.

In our analysis of the android version of the software we found several items that in our opinion can be used for data gathering purposes. The software gathers nearly all information about the current system it is on such as: GPS location, MAC address, installed applications, the phone number of the device, operating system information and taking screenshots (it is worth mentioning that these screenshots may be used to provided previews for files). It also takes information on device details and network and sim information. As seen in figures 18 to 22 below. We did note that a lot of the features including access to clipboard probably enhance the compatibility and user experience of the software. We also noted that the software has multiple http connections to upload its user information back to iciba.com which links back to kingsoft.com and that this information probably comes under Chinese Government jurisdiction. This upload function can be seen in figures 23 and 24.

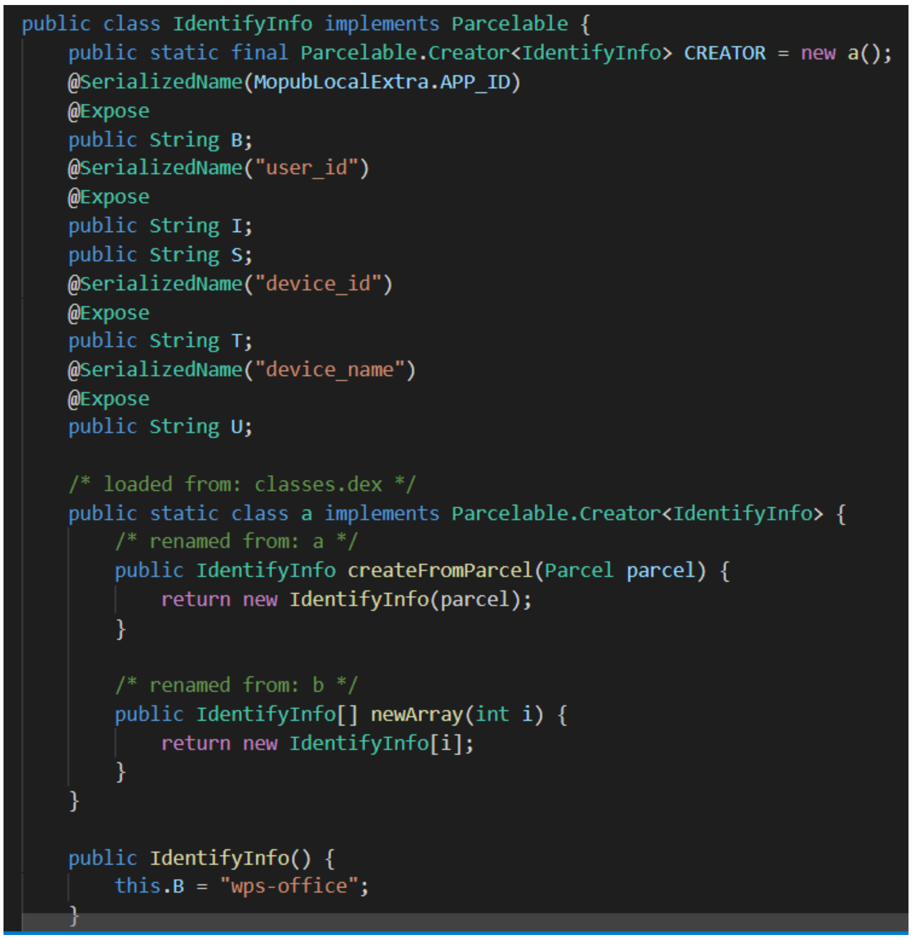
From a risk perspective we would state that gathering network connection information like sim data and location data in unnecessary collection for the function of office software. Complete access to clipboard, camera access and device details would be needed for functional use but the broad scope of the data collection shown is an example of the surveillance culture which imposed by the Chinese Government national security legislation. From a risk perspective users should be aware that this data is able to be processed and sent back via network connections to Kingsoft.com which would place the data under Chinese Government jurisdiction.

[20] https://www.wps.com/



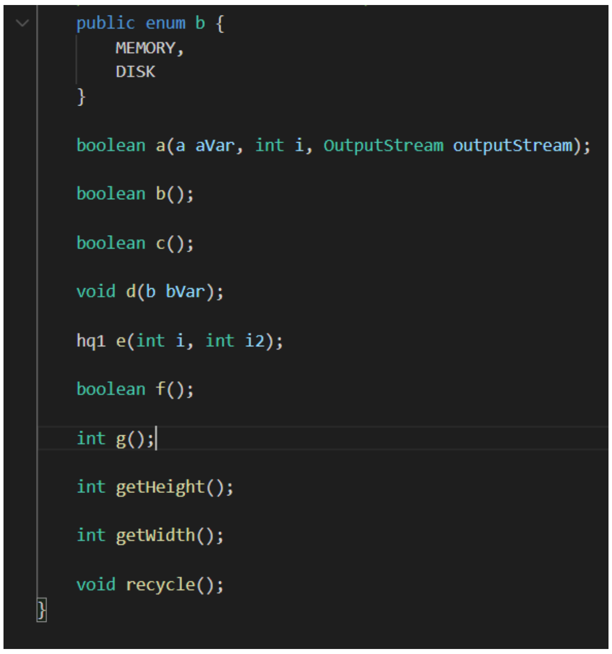
```java
public class DeviceInfo implements Parcelable {
    public static final Parcelable.Creator<DeviceInfo> CREATOR = new a();
    @SerializedName("identify_info")
    @Expose
    public IdentifyInfo B;
    @SerializedName("client_info")
    @Expose
    public ClientInfo I;
    @SerializedName("os_info")
    @Expose
    public OsInfo S;
    @SerializedName("net_info")
    @Expose
    public NetInfo T;
    @SerializedName("additional_info")
    @Expose
    public AdditionalInfo U;
    @SerializedName("ext")
    @Expose
    public String V;
    @SerializedName("status")
    @Expose
    public int W;
    @SerializedName("register_time")
    @Expose
    public long X;
```

Figure 18: WPS Office Android software gathering information in preparation of sending out to other networks



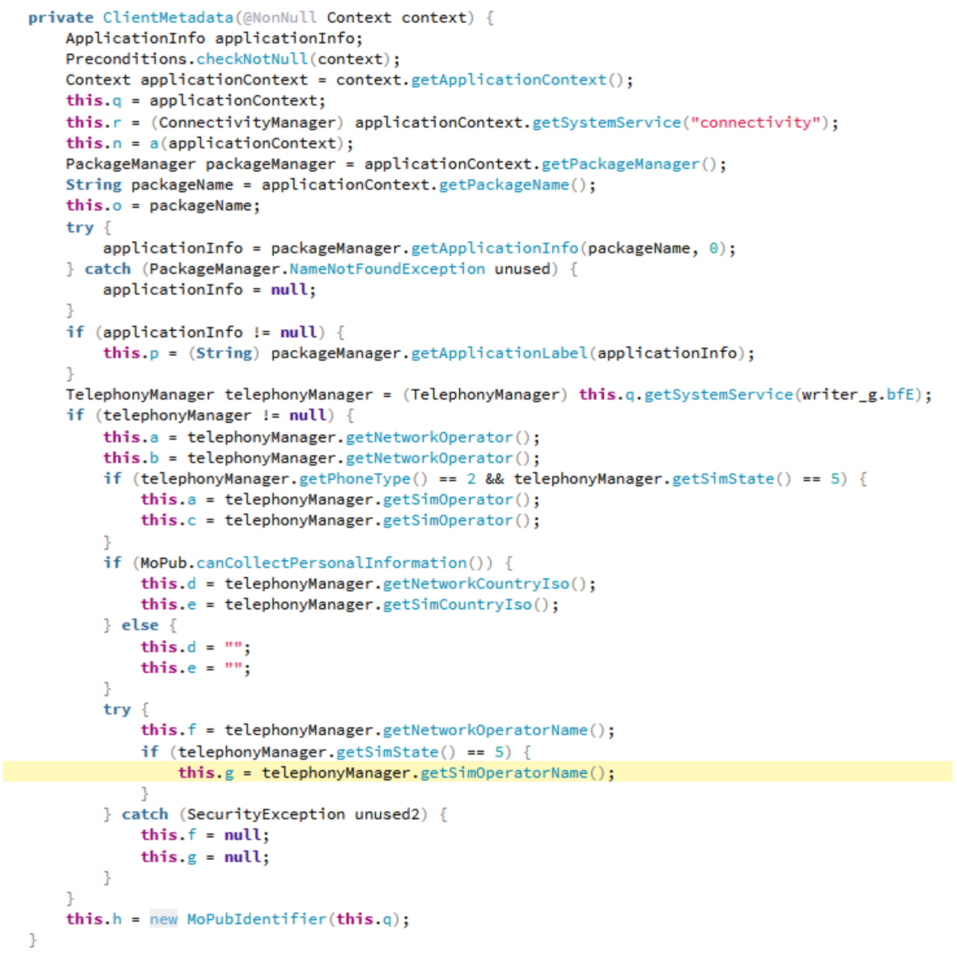
```java
public class IdentifyInfo implements Parcelable {
    public static final Parcelable.Creator<IdentifyInfo> CREATOR = new a();
    @SerializedName(MopubLocalExtra.APP_ID)
    @Expose
    public String B;
    @SerializedName("user_id")
    @Expose
    public String I;
    public String S;
    @SerializedName("device_id")
    @Expose
    public String T;
    @SerializedName("device_name")
    @Expose
    public String U;

    /* loaded from: classes.dex */
    public static class a implements Parcelable.Creator<IdentifyInfo> {
        /* renamed from: a */
        public IdentifyInfo createFromParcel(Parcel parcel) {
            return new IdentifyInfo(parcel);
        }

        /* renamed from: b */
        public IdentifyInfo[] newArray(int i) {
            return new IdentifyInfo[i];
        }
    }

    public IdentifyInfo() {
        this.B = "wps-office";
    }
```

Figure 19: WPS Office Android software obtaining user ID, Device ID and Device name.



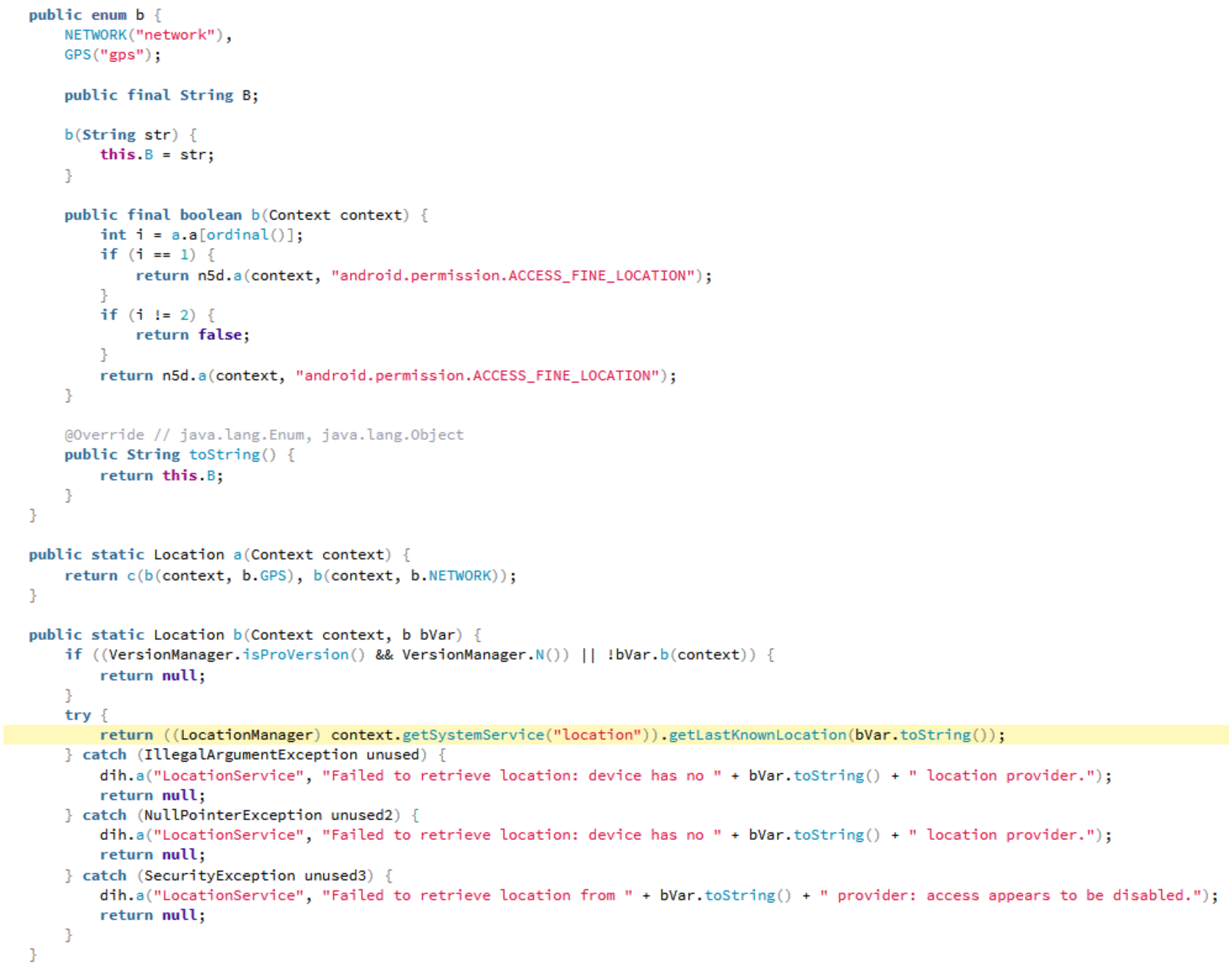
```
    public enum b {
        MEMORY,
        DISK
    }

    boolean a(a aVar, int i, OutputStream outputStream);

    boolean b();

    boolean c();

    void d(b bVar);

    hq1 e(int i, int i2);

    boolean f();

    int g();

    int getHeight();

    int getWidth();

    void recycle();
}
```

Figure 20: WPS Office Android software gathering device height and width dimensions.



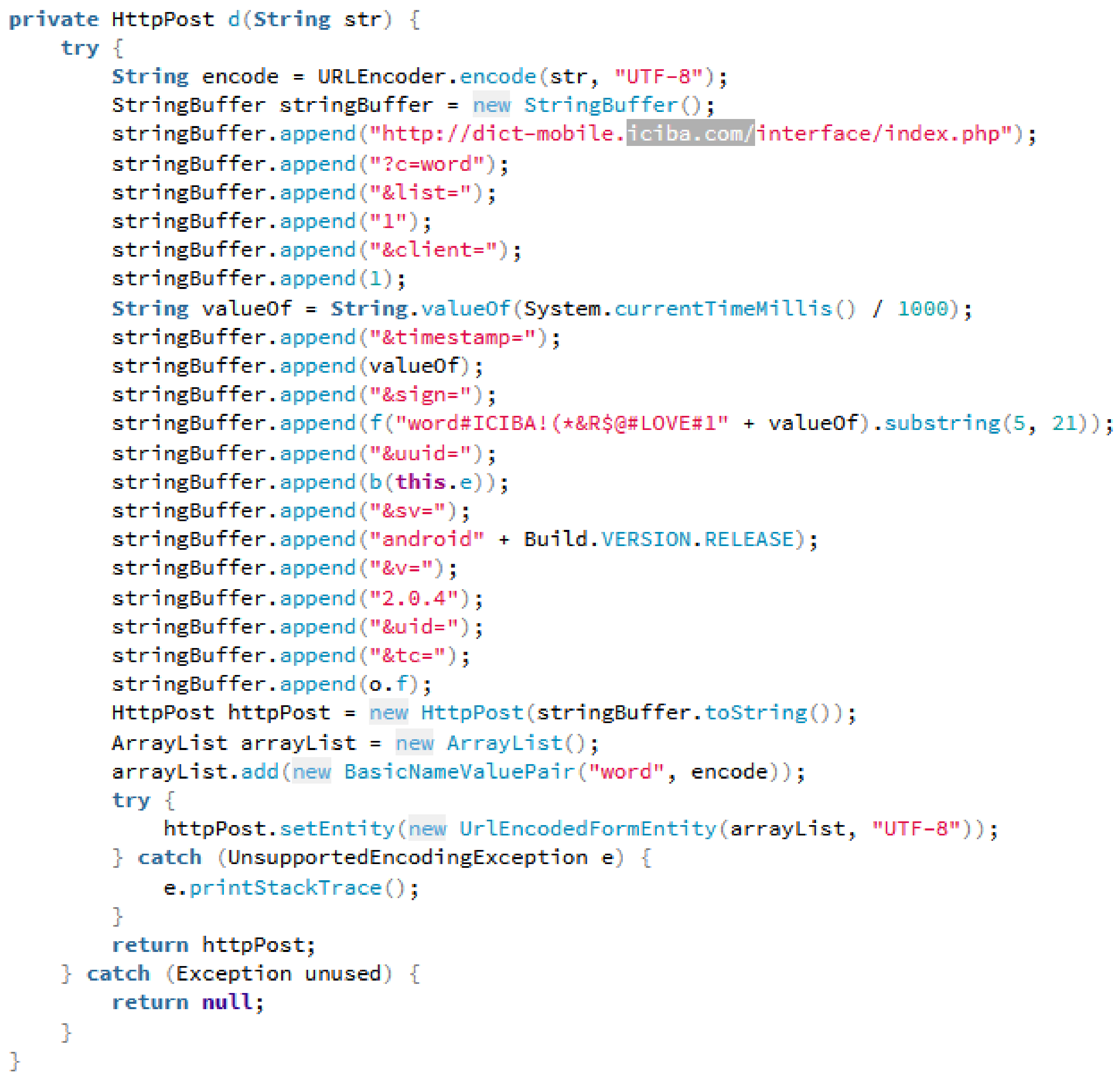
```
private ClientMetadata(@NonNull Context context) {
    ApplicationInfo applicationInfo;
    Preconditions.checkNotNull(context);
    Context applicationContext = context.getApplicationContext();
    this.q = applicationContext;
    this.r = (ConnectivityManager) applicationContext.getSystemService("connectivity");
    this.n = a(applicationContext);
    PackageManager packageManager = applicationContext.getPackageManager();
    String packageName = applicationContext.getPackageName();
    this.o = packageName;
    try {
        applicationInfo = packageManager.getApplicationInfo(packageName, 0);
    } catch (PackageManager.NameNotFoundException unused) {
        applicationInfo = null;
    }
    if (applicationInfo != null) {
        this.p = (String) packageManager.getApplicationLabel(applicationInfo);
    }
    TelephonyManager telephonyManager = (TelephonyManager) this.q.getSystemService(writer_g.bfE);
    if (telephonyManager != null) {
        this.a = telephonyManager.getNetworkOperator();
        this.b = telephonyManager.getNetworkOperator();
        if (telephonyManager.getPhoneType() == 2 && telephonyManager.getSimState() == 5) {
            this.a = telephonyManager.getSimOperator();
            this.c = telephonyManager.getSimOperator();
        }
        if (MoPub.canCollectPersonalInformation()) {
            this.d = telephonyManager.getNetworkCountryIso();
            this.e = telephonyManager.getSimCountryIso();
        } else {
            this.d = "";
            this.e = "";
        }
        try {
            this.f = telephonyManager.getNetworkOperatorName();
            if (telephonyManager.getSimState() == 5) {
                this.g = telephonyManager.getSimOperatorName();
            }
        } catch (SecurityException unused2) {
            this.f = null;
            this.g = null;
        }
    }
    this.h = new MoPubIdentifier(this.q);
}
```

Figure 21: WPS Office Android software gathering device sim information



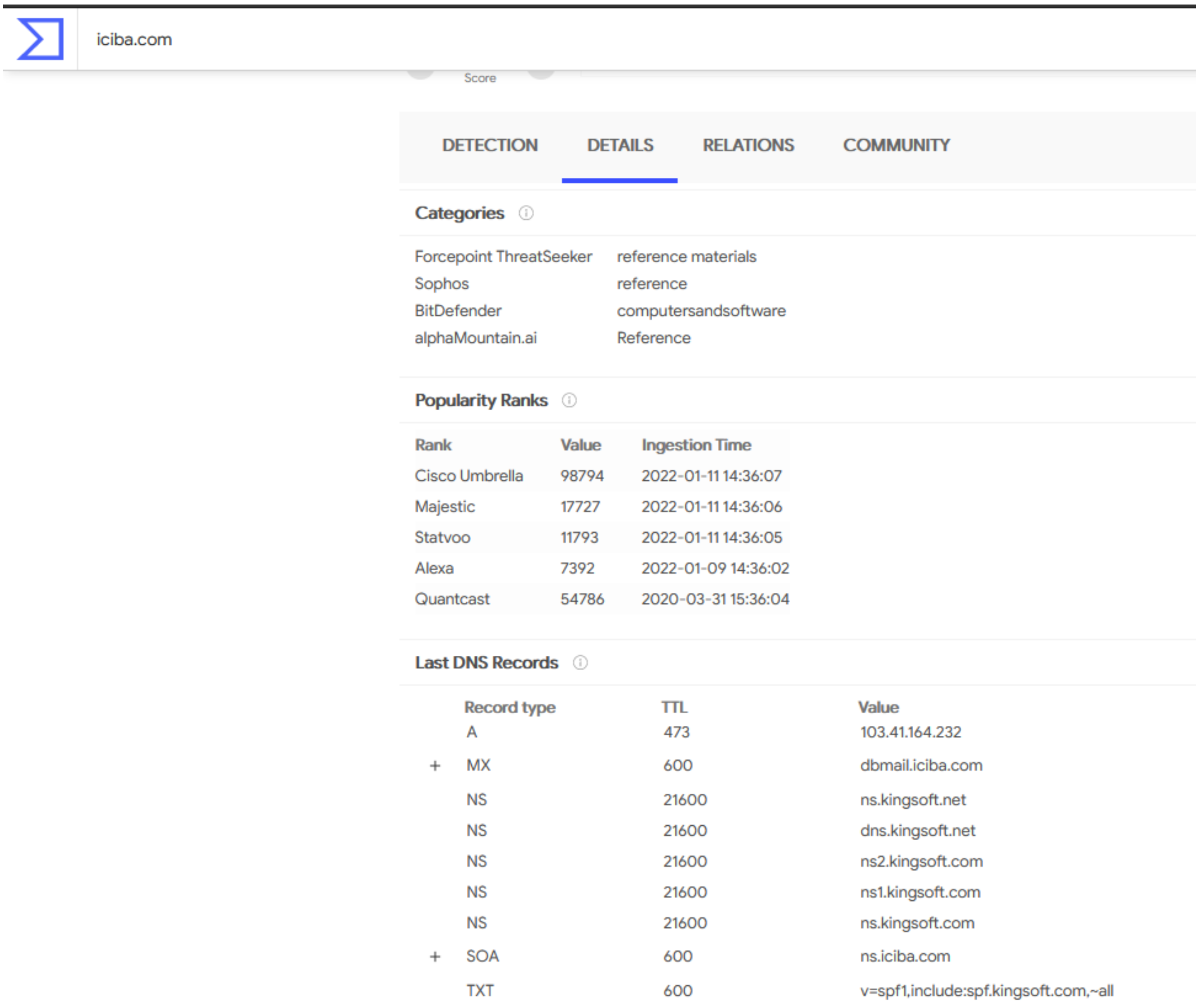
```java
public enum b {
    NETWORK("network"),
    GPS("gps");

    public final String B;

    b(String str) {
        this.B = str;
    }

    public final boolean b(Context context) {
        int i = a.a[ordinal()];
        if (i == 1) {
            return n5d.a(context, "android.permission.ACCESS_FINE_LOCATION");
        }
        if (i != 2) {
            return false;
        }
        return n5d.a(context, "android.permission.ACCESS_FINE_LOCATION");
    }

    @Override // java.lang.Enum, java.lang.Object
    public String toString() {
        return this.B;
    }
}

public static Location a(Context context) {
    return c(b(context, b.GPS), b(context, b.NETWORK));
}

public static Location b(Context context, b bVar) {
    if ((VersionManager.isProVersion() && VersionManager.N()) || !bVar.b(context)) {
        return null;
    }
    try {
        return ((LocationManager) context.getSystemService("location")).getLastKnownLocation(bVar.toString());
    } catch (IllegalArgumentException unused) {
        dih.a("LocationService", "Failed to retrieve location: device has no " + bVar.toString() + " location provider.");
        return null;
    } catch (NullPointerException unused2) {
        dih.a("LocationService", "Failed to retrieve location: device has no " + bVar.toString() + " location provider.");
        return null;
    } catch (SecurityException unused3) {
        dih.a("LocationService", "Failed to retrieve location from " + bVar.toString() + " provider: access appears to be disabled.");
        return null;
    }
}
```

Figure 22: WPS Office Android software gathering location data



```
private HttpPost d(String str) {
    try {
        String encode = URLEncoder.encode(str, "UTF-8");
        StringBuffer stringBuffer = new StringBuffer();
        stringBuffer.append("http://dict-mobile.iciba.com/interface/index.php");
        stringBuffer.append("?c=word");
        stringBuffer.append("&list=");
        stringBuffer.append("1");
        stringBuffer.append("&client=");
        stringBuffer.append(1);
        String valueOf = String.valueOf(System.currentTimeMillis() / 1000);
        stringBuffer.append("&timestamp=");
        stringBuffer.append(valueOf);
        stringBuffer.append("&sign=");
        stringBuffer.append(f("word#ICIBA!(*&R$@#LOVE#1" + valueOf).substring(5, 21));
        stringBuffer.append("&uuid=");
        stringBuffer.append(b(this.e));
        stringBuffer.append("&sv=");
        stringBuffer.append("android" + Build.VERSION.RELEASE);
        stringBuffer.append("&v=");
        stringBuffer.append("2.0.4");
        stringBuffer.append("&uid=");
        stringBuffer.append("&tc=");
        stringBuffer.append(o.f);
        HttpPost httpPost = new HttpPost(stringBuffer.toString());
        ArrayList arrayList = new ArrayList();
        arrayList.add(new BasicNameValuePair("word", encode));
        try {
            httpPost.setEntity(new UrlEncodedFormEntity(arrayList, "UTF-8"));
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return httpPost;
    } catch (Exception unused) {
        return null;
    }
}
```

Figure 23: WPS Office Android software uploading information to iciba.com



iciba.com

Score

DETECTION DETAILS RELATIONS COMMUNITY

Categories

Forcepoint ThreatSeeker	reference materials
Sophos	reference
BitDefender	computersandsoftware
alphaMountain.ai	Reference

Popularity Ranks

Rank	Value	Ingestion Time
Cisco Umbrella	98794	2022-01-11 14:36:07
Majestic	17727	2022-01-11 14:36:06
Statvoo	11793	2022-01-11 14:36:05
Alexa	7392	2022-01-09 14:36:02
Quantcast	54786	2020-03-31 15:36:04

Last DNS Records

	Record type	TTL	Value
	A	473	103.41.164.232
+	MX	600	dbmail.iciba.com
	NS	21600	ns.kingsoft.net
	NS	21600	dns.kingsoft.net
	NS	21600	ns2.kingsoft.com
	NS	21600	ns1.kingsoft.com
	NS	21600	ns.kingsoft.com
+	SOA	600	ns.iciba.com
	TXT	600	v=spf1,include:spf.kingsoft.com,~all

Figure 24: WPS Office Android software uploading information to iciba.com which links back to kingsoft.com data from virustotal.com

Key Terms used in report

Anti-Virtual Machine (anti-VM) and Anti-Debug	Anti-virtual machine (Anti-VM) and Anti-debugging techniques thwart attempts at malware analysis. With these techniques, the malware attempts to detect whether it is being run inside a virtual machine. If a virtual machine is detected, it can act differently or simply not run.[21]
Application Programming Interface (API)	API is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.[22]
IMEI	The International Mobile Equipment Identity (IMEI) is a number, usually unique to identify mobile phones, as well as some satellite phones.[23]
IMSI	An International Mobile Subscriber Identity (IMSI) is a 15-digit number for every user in a Global System for Mobile communication (GSM). The IMSI is used by Mobile Network Operators (MNOs) and is an important part of the Subscriber Identity Module (SIM) profile.[24]
MAC Addresses	Every device connected on a network has a Media Access Control (MAC) address, that uniquely identifies it. The MAC address is a 12-digit hexadecimal number that is most often displayed with a colon or hyphen separating every two digits (an octet), making it easier to read.[25]
Malware	Malware (malicious software) is a file or code, typically delivered over a network, that infects, explores, steals or conducts virtually any behavior an attacker wants.
SIM	A SIM card, or subscriber identity module, is a small card in your cellphone that connects you to the network. A SIM card contains a phone number, and lets you make phone calls, send text messages, and more.
Trojan	A type of malware that downloads onto a computer disguised as a legitimate program.
Virtual Private Network (VPN)	A virtual private network, or VPN, is an encrypted connection over the Internet from a device to a network. The encrypted connection helps ensure that sensitive data is safely transmitted.[26]

[21] https://www.oreilly.com/library/view/practical-malware-analysis/9781593272906/ch18.html
[22] https://www.mulesoft.com/resources/api/what-is-an-api
[23] https://en.wikipedia.org/wiki/International_Mobile_Equipment_Identity#cite_note-3gppspec-1
[24] https://www.simoniot.com/what-is-an-imsi/
[25] https://slts.osu.edu/articles/whats-a-mac-address-and-how-do-i-find-it/
[26] https://www.cisco.com/c/en_au/products/security/vpn-endpoint-security-clients/what-is-vpn.html

Internet 2.0

RELENTLESS SECURITY

AUSTRALIA
L1, 18 National Circuit, Barton
ACT, 2600

ABN: 17 632 726 946

UNITED STATES
Suite 100 211 N Union St
Alexandria, 22314

EIN: 86-1567068

E: contact@internet2-0.com
AUS: 1300 583 007
INTL: +611300583007