



针对 Android 移动应用的恶意加密流量标注方法研究

何高峰^{1,2}, 司勇瑞¹, 徐丙凤³

(1. 南京邮电大学 物联网学院, 南京 210003; 2. 东南大学 计算机网络和信息集成教育部重点实验室, 南京 211189;
3. 南京林业大学 信息科学技术学院, 南京 210037)

摘 要: 为区分恶意 Android 移动应用在运行过程中产生的恶意流量和正常流量, 提出一种 Android 移动应用恶意流量标注方法。针对加密类型的网络流量, 根据端口号和流载荷内容的字节熵值进行加密检测, 依据服务器证书等内容判断加密流量是否异常, 同时对恶意 Android 移动应用进行反编译, 并利用程序控制流程图分析该加密流量是否涉及敏感操作, 从而标注出恶意加密流量。对 300 个重打包类型的恶意移动应用进行测试, 实验结果与同基准值对比分析表明, 与未采用该方法的标注结果(1 602 条恶意加密流量)相比, 该方法检测出的恶意加密流量有 341 条, 且标注结果中仅有 28 条为误报流量。

关键词: 移动应用; 加密流量; 数据标注; 异常检测; 恶意代码分析

开放科学(资源服务)标志码(OSID):



中文引用格式: 何高峰, 司勇瑞, 徐丙凤. 针对 Android 移动应用的恶意加密流量标注方法研究[J]. 计算机工程, 2020, 46(7): 116-121, 128.

英文引用格式: HE Gaofeng, SI Yongrui, XU Bingfeng. Research on malicious encrypted traffic annotation method for Android mobile application[J]. Computer Engineering, 2020, 46(7): 116-121, 128.

Research on Malicious Encrypted Traffic Annotation Method for Android Mobile Application

HE Gaofeng^{1,2}, SI Yongrui¹, XU Bingfeng³

(1. College of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210003, China;
2. Key Laboratory of Computer Network and Information Integration Ministry of Education, Southeast University, Nanjing 211189, China;
3. College of Information Science and Technology, Nanjing Forestry University, Nanjing 210037, China)

[Abstract] In order to distinguish malicious traffic generated by running malicious Android applications from normal traffic, this paper proposes a method for annotating malicious traffic of mobile Android applications. For encrypted network traffic, encryption detection is performed based on the port number and the value of byte entropy of the stream payload content. Then whether the encrypted traffic is abnormal is determined based on the server certificate and other content. At the same time, the malicious Android mobile applications are decompiled, and the program is used to control the flow chart to analyze whether the encrypted traffic involves sensitive operations, so as to annotate malicious encrypted traffic. Tests are performed on 300 repackaged types of malicious mobile applications. The comparison of the experimental results with the same benchmark value show that the proposed method detects 341 malicious encrypted traffic where only 28 are false alarms. The result is more accurate than that of annotation that does not use the proposed method, which reports 1 602 malicious encrypted traffic.

[Key words] mobile application; encrypted traffic; data annotation; anomaly detection; malicious code analysis

DOI: 10.19678/j.issn.1000-3428.0055613

基金项目: 国家自然科学基金青年基金项目“面向网络加密流量的恶意移动应用检测研究”(61702282); 国家自然科学基金青年基金项目“集成防范性与安全性建模的信息物理融合系统风险分析”(61802192); 江苏省高等学校自然科学研究面上项目“面向移动应用加密流量的恶意攻击检测研究”(17KJB520023); 江苏省高等学校自然科学研究面上项目“集成防范性与安全性信息物理融合系统风险建模及分析”(18KJB520024)。

作者简介: 何高峰(1984—), 男, 讲师、博士, 主研方向为移动应用安全、网络流量分析; 司勇瑞, 硕士研究生; 徐丙凤, 讲师、博士。

收稿日期: 2019-07-31 **修回日期:** 2019-09-06 **E-mail:** hegaofeng@njupt.edu.cn

0 概述

随着移动互联网的迅速发展,Android 智能终端已经成为日常社会活动的重要辅助工具,与此同时,Android 智能终端的广泛使用也吸引了众多攻击者的目光,造成各类恶意移动应用攻击事件层出不穷。如 2019 年趋势科技安全团队在 Google Play 中检测出恶意应用 Flappy Birr Dog,其会窃取用户的短信、联系人信息以及截屏、录音等敏感性文件,对全球 196 个国家都造成了严重的影响^[1]。恶意移动应用现已普遍采用 HTTPS 等加密流量进行网络数据传输从而躲避检测^[2],这给移动网络和移动用户的安全防护带来了巨大的挑战。

为了从加密网络流量中识别出恶意移动应用,研究人员提出多种基于机器学习的检测方法,如文献[3-5]等。此类检测方法的共同流程是同时运行恶意移动应用和正常移动应用,采集其产生的网络流量作为数据样本,从收集的网络流量样本中提取检测特征,利用支持向量机等机器学习方法对检测特征进行训练,从而得出检测模型。再利用检测模型对未知流量进行分类检测,判定未知流量是否由恶意移动应用产生。在上述流程中,准确的网络流量采集是检测方法的基础,这是因为机器学习分类方法的有效性依赖于数据标注的正确性^[6]。

然而,现有工作对恶意移动应用的流量标注研究较少。实际上,已有相关研究均将恶意移动应用产生的所有网络流量直接标注为恶意流量,并对其进行特征提取和建模分类,但该方法会产生较多的错误标注,如对公开的恶意移动应用网络流量数据集 CICAAGM^[7]进行分析时,150 个恶意移动应用总共产生 477 条 TLS (Transport Layer Security) 流量,但其中多数 TLS 流量都具有 Google 的合法证书。进一步利用 TLS 证书的别名选项排除 Google 和 Facebook 中的正常流量,剩余的恶意流量数目仅有 196 条。在此案例中,若将所有加密 TLS 流量均标注为恶意流量,则会产生大量的错误样本标注数据,从而影响最终检测结果。究其原因,多数 Android 恶意移动应用通过重打包方式形成^[8],黑客攻击者通常选取对热门移动应用进行反编译并加入恶意代码,再对修改后的代码和资源文件进行重新打包发布,诱骗用户下载运行。因此,移动用户在运行该恶意移动应用时,原有代码也将被执行以完成主体功能,从而产生正常交互网络流量。这类流量与恶意代码无关,应当标注为正常流量,进而使得恶意流量样本数据集更加精确。

为区分恶意移动应用产生的加密网络流量为恶意流量还是正常流量,本文提出一种针对 Android 移动应用的恶意加密流量标注方法。通过 TLS 握手协议等特征筛选出可疑加密流量,再对恶意移动应用进行反编译,利用代码分析对可疑加密流量进行确

认,从而标注出恶意加密流量,以减少错误标注。

1 相关工作

现有研究工作提出多种恶意移动应用检测方法,如文献[9]提出一种三步骤检测方法。首先,识别 HTTP POST/GET 请求报文;然后,在识别出的报文中查找用户身份敏感内容,如国际移动用户识别码 IMSI 和移动设备国际识别码 IMEI 等;最后,若检测出敏感内容,使用 WHOIS 工具查询远端服务器的注册信息,并判断远端服务器是否为攻击者服务器。若远端服务器为攻击者服务器,则判定当前流量为恶意移动应用产生的恶意流量,反之则为恶意移动应用产生的正常流量,该方法需要检测报文内容,因而只适用于明文流量,无法应用于加密流量的检测判断。

针对加密网络流量的检测判断,文献[10]以平均报文长度、流平均长度、报文时间间隔等网络流量统计值为检测特征,利用机器学习分类方法对 Android 恶意移动应用进行检测。在测试的 48 个移动应用样本中(包括正常移动应用和恶意移动应用),总共成功检测出 45 个样本,准确率为 93.75%。文献[11]以发送和接收的报数字节数、网络状态(WiFi、蜂窝等)、应用状态(前台、后台等)为移动应用的行为特征,利用机器学习方法建立恶意移动应用检测模型,实验结果表明检测率超过 80%,误报率小于 10%。

文献[12]提出一种基于网络行为对比的重打包应用检测方法。通过选择类似应用将待检测应用的网络行为与类似应用的网络行为进行比对,若行为差别超过一定阈值,则判断为重打包应用。该检测方法以网络行为作为考虑对象,不涉及流量内容,因而适用于加密流量的检测,但如何选择合适的类似应用是一项挑战。为此,文献[13]提出一种基于移动边缘计算的恶意重打包应用检测方法,在移动边缘计算节点处对不同移动设备产生的同一应用流量进行聚类分析,从而能够准确发现重打包应用。文献[14]提出基于攻击图的移动应用进行安全评估。文献[15]构建了移动应用安全防护生态链。文献[16]使用域名对恶意移动应用进行检测。但上述研究工作均未涉及恶意加密流量的标注问题。

传统的网络安全研究中已注意到恶意网络流量的标注问题。文献[17]提出一种利用网络攻防比赛来标注网络攻击数据集的思路,并说明了相关系统的组成部分和功能。文献[18]针对恶意 DNS 查询流量,提出一种半人工方式的标注方法。文献[19]利用黑名单、安全报告和沙盒分析对恶意网络流量进行标注,进而提高机器学习分类方法的检测效果。现有工作主要解决传统网络攻击流量的标注问题,对本文研究具有重要的启发和借鉴意义。

2 Android 移动应用恶意加密流量标注

2.1 方法概述

本文提出 Android 移动应用恶意加密流量标注方法的整体思路,具体如图 1 所示。首先,根据端口号和报文熵值判断流量是否为加密流量,若为加密流量,则依据 TLS 握手协议解析 Client Hello 报文和服务器端返回的证书等内容,若解析

失败或握手协议内容异常,则标记该加密流量为异常流量。然后,对移动应用 APK 文件进行反编译,在代码中检索对应域名或 IP 地址。最后,以对应域名或 IP 地址所在函数为中间节点,构建程序控制流程图,并在图中寻找是否含有敏感函数调用,若含有敏感函数调用则标注该流量为恶意加密流量。

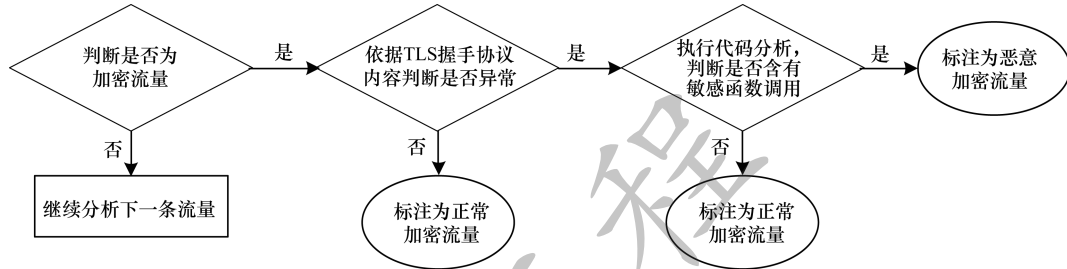


图 1 本文标注方法的整体思路

Fig. 1 Overall idea of the proposed annotation method

在上述基本思路中,有 2 点需要加以解释。一是网络流量与移动应用的对应关系,在采集移动应用网络流量数据时会单独运行某移动应用程序,并利用 VPN 程序排除其他干扰流量^[13],因而可以明确网络流量所对应的移动应用。二是恶意移动应用多数调用 Android 系统函数来完成攻击目的^[20],如调用读取短信、日历或联系人的系统函数来窃取用户数据,或调用发送短信、拨打电话相关系统函数进行恶意扣费等,因此本文利用是否含有敏感函数调用来判断是否存在恶意行为。

2.2 异常加密流量的检测

从图 1 可以看出,标注加密网络流量时需判断该流量是否为加密流量,以及是否为异常流量。Android 移动应用多数利用 HTTPS 协议进行数据的加密传输^[21],判断服务器端口是否为 443,若是,则判定该流量为加密流量。值得注意的是,若 443 端口流量不属于加密流量,后续的代码分析将会对其进行排除。若远程端口不为 443,则计算报文内容载荷的熵值^[22-23],具体过程如下:

针对网络流 f ,合并所有报文内容(不考虑报文头部信息),形成长度为 N 的字节串,即网络流 f 内容大小为 N 字节。计算每一字节对应的数值(0 ~ 255),并计算数值 n_i 出现的频率。如第一个字节对应的数值为 3,且在 N 个字节中,数值 3 一共出现 10 次,则数值 3 的频率计算为 $10/N$,记频率 p_i 为 n_i/N ,则网络流 f 的内容载荷熵值 $H(f)$ 的计算方法如式(1)所示:

$$H(f) = - \sum_{i=1}^m p_i \lg p_i \quad (1)$$

其中, m 为不同数值的个数。

理想情形下的所有数值都均匀分布,此时熵值最大为 8。计算 $H(f)$ 与 8 的差值 d ,若 d 小于设定的

阈值 τ ,则判定该流量为加密流量, d 的计算方法如式(2)所示:

$$d = 8 - H(f) \quad (2)$$

根据文献[23]可知,当 $N \geq 1\,024$ 时,加密的网络流 f 的内容大小大于 1 KB 时, $H(f)$ 值无限趋近于 8,在实际中易于满足该条件。因而本文方法对 τ 值的设定没有严格要求,实验中 τ 值设定为 0.8。

识别出加密流量后,统一执行 TLS 协议解析,解析 Client Hello 报文和服务器端返回的证书内容。若 Client Hello、证书内容或服务器自身异常,则判定该加密流量为异常流量。具体的加密流量异常判断条件如下:

- 1) TLS 协议解析失败。
- 2) 或者 Client Hello 报文中的密码套件支持 3DES、RC4 等不安全加密算法。
- 3) 或者解析服务器的证书内容失败。
- 4) 或者服务器证书为自签名。
- 5) 或者服务器位于云平台中,如 Google App Engine、百度云盘等。
- 6) 或者服务器不在 Alexa top 200 中。

上述 6 个判断条件为“或者”关系,只需满足其一,则可判定为异常加密流量。异常加密流量判定的背后逻辑是:正常 Android 移动应用多数基于 HTTPS(HTTP + TLS) 协议进行数据的加密传输,且客户端优先选择椭圆曲线加密算法等高安全算法,以保证通信内容的安全。而恶意移动应用利用加密流量仅仅是为了躲避检测,且多数自签名其证书以降低攻击成本和隐藏自身信息(购买和维护 X.509 证书需要一定费用,且泄露攻击者自身信息)^[24-25]。由于黑客趋向于借助公共云平台隐藏身份^[26],且黑客自身部署的服务器排名较低,因而上述 6 个条件能够有效检测出异常加密流量。

2.3 恶意加密流量的确认

检测出异常加密流量后,执行代码分析确认该流量是否为恶意流量,具体步骤如下:

输入 流量域名或 IP 地址 A 以及对应的移动应用 APK 文件 F

输出 流量标注

//初始化

步骤 1 判定 APK 文件 F 是否存在全局程序控制流程图 G ,若不存在,则反编译 APK 文件 F ,构建全局程序控制流程图 G 并保存。

步骤 2 在反编译后的源代码中寻找 IP 地址 A ,确定 IP 地址 A 所在的函数 $\text{Fun}(A)$ 以及产生该流量的参数变量 $\text{Par}_1, \text{Par}_2, \dots, \text{Par}_p$ 。

步骤 3 通过函数调用判断与 IP 地址 A 的网络连接是否为加密连接,若不为加密连接,则返回。

//前向搜索(确定参数是否敏感)

步骤 4 在函数 $\text{Fun}(A)$ 中检查参数 $\text{Par}_1, \text{Par}_2, \dots, \text{Par}_p$ 的赋值情形,若在赋值过程中调用了敏感函数,返回“恶意”标注。

步骤 5 在全局程序控制流程图 G 中,以函数 $\text{Fun}(A)$ 为起点,深度优先向前遍历所有父节点。

步骤 6 在父节点中,检查对参数 $\text{Par}_1, \text{Par}_2, \dots, \text{Par}_p$ 的隐形赋值情况,若在赋值过程中调用了敏感函数,返回“恶意”标注。

步骤 7 若遍历结束,返回“正常”标注。

//后向搜索(判定后续是否有敏感函数调用)

步骤 8 在全局程序控制流程图 G 中,以函数 $\text{Fun}(A)$ 为起点,向后遍历所有子节点。

步骤 9 若子节点中存在敏感函数调用,返回“恶意”标注。

步骤 10 若遍历结束,返回“正常”标注。

在上述步骤中,初始化阶段的核心功能是反编译 APK 文件 F ,然后针对得到的源代码,构建全局程序控制流程图 G 。APK 文件名称、反编译后的文件与全局程序控制流程图将一一对应保存,便于后续分析使用。程序控制流程图反映程序调用关系,示例如图 2 所示。Android 应用的程序控制流程图可由 Soot 框架自动生成,本文不再进行赘述。

源	目标	类型
<dummyMainClass: void dum...	<dummyMainClass: com.zuse.co...	有向的
<dummyMainClass: void dum...	<dummyMainClass: com.mobile.c...	有向的
<dummyMainClass: void dum...	<dummyMainClass: com.mobile.c...	有向的
<dummyMainClass: void dum...	<dummyMainClass: com.prime31...	有向的
<dummyMainClass: void dum...	<dummyMainClass: com.zuse.co...	有向的
<dummyMainClass: void dum...	<dummyMainClass: com.mobile.c...	有向的
<dummyMainClass: void dum...	<dummyMainClass: com.unity3d...	有向的
<dummyMainClass: void dum...	<dummyMainClass: com.mobile.c...	有向的
<dummyMainClass: void dum...	<dummyMainClass: com.unity3d...	有向的
<dummyMainClass: void dum...	<dummyMainClass: com.prime31...	有向的
<dummyMainClass: void dum...	<dummyMainClass: com.unity3d...	有向的
<dummyMainClass: void dum...	<dummyMainClass: com.mobile.c...	有向的
<dummyMainClass: void dum...	<dummyMainClass: com.mobile.c...	有向的

图 2 程序调用关系示例

Fig. 2 Example of program call relationship

在初始化阶段的步骤 2 中,利用域名或 IP 地址 A 定位至特定函数。但在实际中,域名可以动态生成(通过参数赋值或多个参数串联而成,如 `new URL(str)`),无法直接进行匹配查询。因此,在生成全局程序控制流程图时,对于网络访问类函数,需要对访问地址参数进行数据流分析,进而得到域名的完整表示,具体流程同前向搜索类似。

执行完初始化后,进行前向搜索,确定函数参数是否含有敏感数据。参数变量 $\text{Par}_1, \text{Par}_2, \dots, \text{Par}_p$ 表示 Android 系统提供的网络相关函数,如 `setURI` 函数的参数。在当前节点所在的函数 $\text{Fun}(A)$ 中寻找对 $\text{Par}_1, \text{Par}_2, \dots, \text{Par}_p$ 的赋值语句,若在赋值语句的右边存在变量 Var ,则继续向上遍历其父节点,寻找对变量 Var 的赋值语句(即对参数 $\text{Par}_1, \text{Par}_2, \dots, \text{Par}_p$ 的隐形赋值)。以此类推,形成递归分析,从而最终确定参数 $\text{Par}_1, \text{Par}_2, \dots, \text{Par}_p$ 的具体值,如图 3 所示。若在赋值路径中存在敏感函数调用,则返回“恶意”标注。

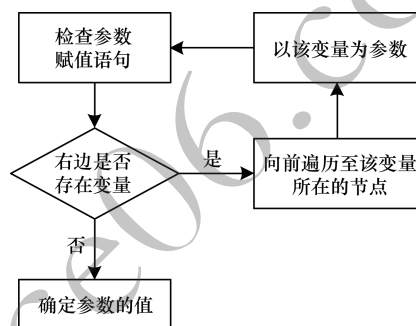


图 3 参数值回溯流程

Fig. 3 The backtracking process of parameter value

在后向搜索中,无需考虑参数问题,只需判断子节点是否为敏感函数。即在网络通信结束后,根据通信内容,恶意移动应用会执行相关攻击代码。此行为在远程控制、广告展示类恶意应用中较为常见。恶意流量的确认算法中的敏感函数具体有 `startService()`、`loadClass()`、`sendTextMessage()`、`getContentResolver()`、`query()`、`mkdir()`、`delete()`、`ListFiles()` 等。完整的函数列表及其在恶意应用的常见应用解释见文献[27]的第 4 节内容,但不包括其中的 `URLConnection` 和 `Sockets` 等网络相关函数。

3 实验结果与分析

为了测试本文方法的准确性,实验重点对重打包类型恶意应用进行分析。这是由于可以通过比对原始应用和重打包应用的代码,确定恶意代码位置,从而明确网络流量的真实类别。即由原始代码产生的网络流量为正常流量,而由额外添加的代码所产生的网络流量为恶意流量,进而为本文方法的测试

提供基准值。

从公开数据集 AndroZoo^[28] 中下载了 300 个合法移动应用和 300 个对应的重打包移动应用。针对下载的 600 个移动应用,首先反编译其代码,经文件比对定位出恶意代码位置。代码反编译由 Jadx 工具完成,代码文件的比较通过 WinMerge 来实现。然后,将重打包应用运行于真实的智能手机中,捕获其产生的网络流量。若运行出错(如移动应用版本较早,无法运行于最新手机),则根据移动应用的 API 版本,配置对应虚拟机进行安装运行。移动应用的安装运行均自动化进行,且安装过程使用 adb install 命令实现。安装完成后,使用开源的 DroidBot 自动点击产生点击事件,运行移动应用不同组件。

如图 4 所示,移动应用流量的采集由网关完成。网关运行 Ubuntu 16.04 操作系统,并配置 WiFi 热点功能,智能手机及虚拟机均通过无线网络连接至该网关。在网关中,部署 tshark 软件进行网络流量报文的抓取。流量域名的提取同样由 tshark 完成,具体命令为 tshark-e dns. qry. name。本文重点解决加密流量的标注问题,在流量采集中仅保存加密流量。首先判断服务器端口是否为 443,若不为 443,则计算流载荷的字节熵值。若熵值大于 7.2,即式(2)中的 τ 值为 0.8,则标记为加密流量。对于采集的加密流量统一按照 TLS 协议进行解析,若解析成功,则标记为 TLS 流量,否则标记为其他类型的加密流量。结果发现,实验中采集的 TLS 流量数量为 1 523 条,其他类型的加密流量数量为 116 条。

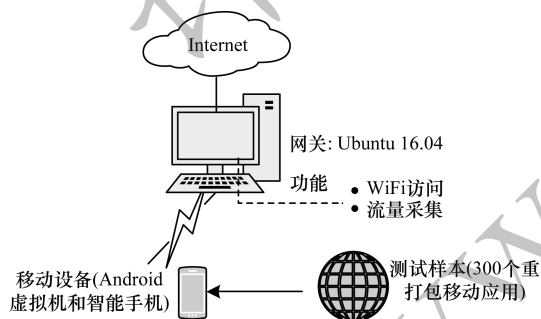


图 4 采集网络流量环境设置

Fig. 4 Environment settings for collecting network traffic

针对采集到的其他类型的加密流量,进一步分析其对应的移动应用源代码,确定其是否为真正加密类型的网络流量。在 116 条其他类型的加密流量中,总共有 79 条流量为加密流量,其产生的方式为使用 AES 或自定义的加密算法对数据内容进行加密,通过 Java Socket 编程将数据发送至远程服务器。而剩余的 37 条网络流量为压缩流量,数据内容经 gzip 等压缩后发送至服务器。后续的分析仅针对 1 602 (1 523 + 79) 条加密网络流量进行。

针对筛选后的加密网络流量,利用 2.2 节中的

异常判断条件按照先后顺序匹配检测,若满足其中一项,则判断异常并停止后续匹配,检测结果如表 1 所示。

表 1 异常检测结果
Table 1 Results of anomaly detection

检测结果	数量
正常流量	1 180
TLS 协议解析失败异常	79
Client Hello 报文异常	15
服务器证书自签名异常	217
服务器位于云平台异常	111

对表 1 中的 422 (79 + 15 + 217 + 111) 条异常加密流量执行恶意流量确认算法步骤进行确认。使用 Soot 生成全局程序控制流程图,然后对图执行前向搜索和后向搜索。全局程序控制流程图规模庞大,难以直接展示,因而将数据保存为 XML 格式。利用 Java 解析 XML 文件进行节点的遍历操作,在前向遍历时,需要结合源代码确定变量赋值语句,此部分由人工方式完成。

代码分析结果表明,总共有 341 条加密流量为恶意流量,剩余的 81 条加密流量为正常流量。进一步同基准值进行对比,结果如表 2 所示。

表 2 基准值对比结果
Table 2 Comparison results of benchmark values

检测结果	数量	基准值对比
异常检测为正常流量	1 180	全部为正常流量
代码确认为正常流量	81	全部为正常流量
代码确认为恶意流量	341	313 条为真实恶意流量, 28 条为误报流量

从表 2 可知,在异常检测阶段检测出的正常流量均为正常加密流量,从而减少了恶意流量确认部分的分析负载。在恶意流量确认阶段,针对 422 条异常加密流量,共产生了 28 条误报流量,误报率为 6.6%。与未采用本文方法标注的 1 602 条恶意加密流量相比,采用本文方法标注的恶意加密流量仅有 341 条,总共减少了 1 261 条错误标注,占总数量的 78.7%。由此可知,本文方法能有效减少错误标注,从而构建出更加精确的恶意移动应用网络流量数据集。

4 结束语

本文提出一种针对 Android 移动应用的恶意加密流量标注方法,判断网络流量是否为加密流量,对于非加密流量,使用传统的 DPI 技术对其标注,而对于加密类型的流量,依据 TLS 协议对其解析并利用 TLS 握手消息等内容判断该流量是否异常。若该流量异常,则对相应的恶意 Android 移动应用进行反编译,构建程序控制流程图,并分析该加密流量是否涉

及敏感操作。若该加密流量的参数为敏感数据或执行完网络通信后调用执行敏感函数,则标注该加密流量为恶意加密流量。针对300个重打包类型恶意应用进行测试,分析结果表明,未采用本文方法总共采集1602条加密流量,采用本文方法只有341条流量被标注为恶意加密流量,且仅有28条为误报流量。因此本文方法可准确地标注恶意 Android 移动应用网络流量。

本文仅针对重打包类型恶意 Android 应用进行分析,后续工作将对更多类型的恶意移动应用进行测试,并研究更加合适的异常加密流量判断标准,从而减少异常加密流量的数量,提升代码分析性能,同时识别出APT攻击类型加密网络流量。基于形成的准确数据集,将开展针对单条恶意加密流量的在线检测研究,以更好地保护移动应用和移动网络安全。

参考文献

- [1] XU E, GUO G. Spyware disguises as Android applications on Google play [EB/OL]. [2019-06-03]. <https://blog.trendmicro.com/trendlabs-security-intelligence/spyware-disguises-as-android-applications-on-google-play>.
- [2] HE Gaofeng, XU Bingfeng, ZHANG Lu, et al. Mobile app identification for encrypted network flows by traffic correlation [J]. International Journal of Distributed Sensor Networks, 2018, 14(12): 1-17.
- [3] HE Gaofeng, XU Bingfeng, ZHU Haiting. AppFA: a novel approach to detect malicious Android applications on the network [J]. Security and Communication Networks, 2018, 2018: 1-15.
- [4] SHABTAI A, TENENBOIM-CHEKINA L, MIMRAN D, et al. Mobile malware detection through analysis of deviations in application network behavior [J]. Computers & Security, 2014, 43: 1-18.
- [5] GARG S, PEDDOJU S K, SARJE A K. Network-based detection of Android malicious apps [J]. International Journal of Information Security, 2017, 16(4): 385-400.
- [6] ZHOU L N, PAN S M, WANG J W, et al. Machine learning on big data: opportunities and challenges [J]. Neurocomputing, 2017, 237: 350-361.
- [7] LASHKARI A H, KADIR A F A, GONZALEZ H F, et al. Towards a network-based framework for Android malware detection and characterization [C]//Proceedings of the 15th Annual Conference on Privacy, Security and Trust. Washington D. C., USA: IEEE Press, 2017: 233-239.
- [8] LI L, LI D Y, BISSYANDE T F, et al. Understanding Android app piggybacking: a systematic study of malicious code grafting [J]. IEEE Transactions on Information Forensics and Security, 2017, 12(6): 1269-1284.
- [9] CHEN P S, LIN S C, SUN C H. Simple and effective method for detecting abnormal internet behaviors of mobile devices [J]. Information Sciences, 2015, 321: 193-204.
- [10] ARORA A, GARG S, PEDDOJU S K. Malware detection using network traffic analysis in Android based mobile devices [C]//Proceedings of 2014 8th International Conference on Next Generation Mobile Apps, Services and Technologies. Oxford, UK: IEEE Press, 2014: 1-9.
- [11] SHABTAI A, TENENBOIM-CHEKINA L, MIMRAN D, et al. Mobile malware detection through analysis of deviations in application network behavior [J]. Computers & Security, 2014, 43: 1-18.
- [12] HE Gaofeng, XU Bingfeng, ZHU Haiting. AppFA: a novel approach to detect malicious android applications on the network [J]. Security and Communication Networks, 2018, 2018: 1-15.
- [13] HE Gaofeng, ZHANG Lu, XU Bingfeng, et al. Detecting repackaged Android malware based on mobile edge computing [C]//Proceedings of 2018 6th International Conference on Advanced Cloud and Big Data. Washington D. C., USA: IEEE Computer Society, 2018: 360-365.
- [14] CHEN Lu, LIU Xing, CHEN Mu, et al. Scalable security evaluation model of mobile application based on graph [J]. Computer Engineering, 2018, 44(5): 78-82. (in Chinese)
陈璐, 刘行, 陈牧, 等. 基于图的可扩展移动应用安全评估模型 [J]. 计算机工程, 2018, 44(5): 78-82.
- [15] YANG Xinyu, XU Guoai. Construction method on mobile application security ecological chain [J]. Journal of Software, 2017, 28(11): 3058-3071. (in Chinese)
杨昕雨, 徐国爱. 移动应用安全生态链构建方法 [J]. 软件学报, 2017, 28(11): 3058-3071.
- [16] CAI Rongyan, WANG He, YAO Qigui, et al. Research on malicious mobile application detection based on domain name association [J]. Computer Engineering, 2020, 46(5): 174-180. (in Chinese)
蔡荣彦, 王鹤, 姚启桂, 等. 基于域名关联的恶意移动应用检测研究 [J]. 计算机工程, 2020, 46(5): 174-180.
- [17] SANGSTER B, O'CONNOR T J, COOK T, et al. Toward instrumenting network warfare competitions to generate labeled datasets [C]//Proceedings of the 2nd Conference on Cyber Security Experimentation and Test. San Diego, USA: USENIX Association, 2009: 1-6.
- [18] STEVANOVIC M, PEDERSEN J M, D'ALCONZO A, et al. On the ground truth problem of malicious DNS traffic analysis [J]. Computers & Security, 2015, 55: 142-158.
- [19] FRANC V, SOFKA M, BARTOS K. Learning detector of malicious network traffic from weak labels [C]//Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases. Berlin, Germany: Springer, 2015: 85-99.
- [20] ABRAHAM A, ANDRIATSIMANDEFITRA R, BRUNELAT A, et al. GroddDroid: a gorilla for triggering malicious behaviors [C]//Proceedings of the 10th International Conference on Malicious and Unwanted Software (MALWARE). Washington D. C., USA: IEEE Press, 2015: 15-20.

(上接第 121 页)

- [21] WEI X T, WOLF M. A survey on HTTPS implementation by Android apps: issues and countermeasures [J]. Applied Computing and Informatics, 2017, 13(2): 101-117.
- [22] HE Gaofeng, ZHANG Tao, MA Yuanyuan, et al. A novel method to detect encrypted data exfiltration [C]// Proceedings of 2014 International Conference on Advanced Cloud and Big Data. Washington D. C., USA: IEEE Press, 2014: 240-246.
- [23] OLIVAIN J, GOUBAULT-LARRECQ J. Detecting subverted cryptographic protocols by entropy checking [R/OL]. (2006-06-13) [2019-05-08]. http://www.lsv.fr/publis/RAPPORTS_LSV/PDF/rr-lsv-2006-13.pdf.
- [24] ANDERSON B, PAUL S, MCGREW D. Deciphering malware's use of TLS (without decryption) [J]. Journal of Computer Virology and Hacking Techniques, 2018, 14(3): 195-211.
- [25] LIN Jingqiang, JING Jiwu, ZHANG Qionglu, et al. Recent advances in PKI technologies [J]. Journal of Cryptologic Research, 2015, 2(6): 487-496. (in Chinese)
- 林璟锵, 荆继武, 张琼露, 等. PKI 技术的近年研究综述 [J]. 密码学报, 2015, 2(6): 487-496.
- [26] CHUNG H, PARK J, LEE S, et al. Digital forensic investigation of cloud storage services [J]. Digital Investigation, 2012, 9(2): 81-95.
- [27] AAFER Y, DU W L, YIN H. DroidAPIMiner: mining API-level features for robust malware detection in android [C]// Proceedings of International Conference on Security and Privacy in Communication Systems. Berlin, Germany: Springer, 2013: 86-103.
- [28] ALLIX K, BISSYANDE T F, KLEIN J, et al. AndroZoo: collecting millions of Android apps for the research community [C]// Proceedings of the 13th International Conference on Mining Software Repositories. Washington D. C., USA: IEEE Press, 2016: 468-471.

编辑 刘继娟