基于证据链生成的 Android 勒索软件检测方法

王持恒[1)] 陈晶[1),2)] 陈祥云[1)] 杜瑞颖[1),3)]

[1)](武汉大学计算机学院软件工程国家重点实验室 武汉 430072)

[2)](保密通信重点实验室 成都 610041)

[3)](地球空间信息技术协同创新中心 武汉 430079)

摘 要 近年来，基于 Android 平台的勒索软件呈现爆发式增长趋势，恶意行为也正在不断的进化。Android 勒索软件专门以用户智能设备和隐私文件为攻击目标，给受害者带来了严重的精神和财产损失。本文提出了一种轻量化的勒索软件检测方法，能够在应用安装到手机之前发现潜在的勒索风险。通过广泛收集 2721 个勒索软件样本，并对这些样本进行深入分析，本文在勒索软件锁屏、加密、权限、威胁文本、支付方式和网络通信等方面提取特征，利用模块化规则归纳学习算法实时检测风险。另外，基于自然语言生成技术提出了一种证据链生成方法，将待检测应用的外围信息和匹配的分类规则以普通用户能够理解的方式展现，帮助非专业用户做出合理的决策。最终实验表明，所实现系统能够达到 95%的检测准确率，90%的普通用户表示能够正常理解证据链描述，性能分析结果证明系统能够满足智能手机实时检测的需求。

关键词 Android；勒索软件；模块化规则归纳学习；自然语言生成；证据链；静态分析

中图法分类号 TP309

An Android Ransomware Detection Scheme Based on Evidence Chain Generation

Wang Chi-Heng[1)] Chen Jing[1),2)] Chen Xiang-Yun[1)] Du Rui-Ying[1),3)]

[1)](State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072)

[2)](Science and Technology on Communication Security Laboratory, Chengdu 610041)

[3)](Collaborative Innovation Center of Geospatial Technology, Wuhan 430079)

Abstract In recent years, we witness a drastic increase of ransomware, especially on the popular platforms such as Android. Generally, ransomware limits legitimate users from accessing their own mobile devices by locking the screen or encrypting the storage data. Then, it blackmails victims for a sum of money in return for their devices or files. Different from other type of malware, by removing which the damage can be controlled, deleting ransomware from the infected devices cannot help victims get their files back. In light of ransomware's rapid growth, it is imperative to develop effective solutions in real-time mode that can detect and stop the malicious behaviors of ransomware in early time. However, the research community is still constrained by the lack of a comprehensive dataset, and there exists no insightful understanding of mobile ransomware in the wild, which makes researchers to develop an effective mitigation solution very hard. Although several standalone systems have been designed to detect PC ransomware, the detection approach running on mobile devices should be lightweight to provide good experience to users, which makes directly applying the existing PC solutions to

本课题得到国家自然科学基金项目(No. 61572380, No. 61772383, No. 61702379)和国家重点基础研究发展规划（No. 2014CB340600）资助. 王持恒，男，1990年生，博士研究生，主要研究领域为移动安全. E-mail: chwang@whu.edu.cn. 陈晶（**通讯作者**），男，1981年生，教授，博士生导师，主要研究领域为云计算、密码学、移动安全.E-mail: chenjing@whu.edu.cn. 陈祥云，男，1992年生，硕士研究生，主要研究领域为移动安全、恶意软件检测.E-mail: chenxiangyun@whu.edu.cn. 杜瑞颖，女，1964年生，教授，博士生导师，主要研究领域为云计算、密码学、移动安全.E-mail: duraying@whu.edu.cn.

mobile devices infeasible. Moreover, none of these detection systems can help users understand how, where, when, and why a threat operates. In this paper, we propose a novel light-weight ransomware detection scheme, called RansomGuard, which decompiles an APK file to extract sensitive features and utilizes machine learning method to detect ransomware-like app before the user installs it on the device. Specifically, we have managed to collect 2721 ransomware samples from 15 different families, which seems to be the first large collection of Android ransomware and cover the majority of existing families. Also, we divide the collected samples into three classes, i.e., controlling devices, kidnapping data, and intimidating users. Considering the real-time requirements, we focus on malicious features that can be statically extracted from an APK file without resorting to heavy and time-consuming program analysis. Through analyzing the features of the collected samples, such as lock screen, encrypt file, permission, threaten text, payment and network communication, RansomGuard utilizes inductive learning with modular classification rules to identify ransomware sample. Based on a large number of ransomware samples, RansomGuard can provide specific classification rules for three ransomware classes. The proposed method has an advantage that it is succinct and suitable for the environment of mobile devices. Moreover, to help general users understand the detection result, this paper designs an evidence chain generation method by utilizing the Nature Language Generation (NLG) technique. When RansomGuard detects a high risk application, it will trigger an alarm and generate human readable descriptions to help users make informed decision. The generated evidence can describe all the security related information we have collected, and give the reason why RansomGuard alerts it as a ransomware-like app. The experimental results show that RansomGuard can effectively detect Android ransomware with high accuracy (98%), and most users (90%) can understand the description of evidence chain. In addition, RansomGuard also shows high performance of time and memory, which is suitable for the environment of mobile devices.

Key words Android; Ransomware; Inductive Learning with Modular Classification Rules; Nature Language Generation; Evidence Chain; Static Analysis

1 引言

顾名思义，勒索软件是一种通过技术手段控制用户设备或者数据，并以此胁迫用户支付解锁费用的恶意软件。勒索软件常用的方法包括锁定屏幕和加密文件等。另外，除了勒索用户钱财，勒索软件还会破坏用户数据和手机系统。当它们达到攻击目的之后，会向受害者发出通知，要求他们支付赎金。最初这种攻击主要针对 Windows 平台，但是随着近几年 Android 平台的快速发展，其开放的应用市场和相对宽松的审查机制导致勒索软件泛滥，安全威胁也与日俱增。

2013 年 9 月，SecureWorks 发现 CryptoLocker 勒索软件家族在 100 天之内感染了 20 万到 25 万台设备[1]。纽约时报称仅在 2014 年 8 月，ScarePackage 勒索软件就感染了超过 90 万台 Android 设备，而且危害性仍在不断扩大[2]。2017 年 5 月，WannaCry 勒索软件在三天内迅速感染了 150 国家的超过 350 00 个主机。受害者需要支付 300 美元的赎金才能重新恢复被加密的文件。根据 McAfee[1]实验室 2017 年发布的报告显示，勒索软件的数量已经超过 900 万，而且未来仍然会是主要的设备安全威胁之一。

针对勒索软件的攻击，研究人员提出了专门的检测方案。例如，UNVEIL[2]提出了一种基于动态行为分析的检测方法，能够批量检测针对 Windows 平台的勒索软件。ShieldFS[3]通过监控文件系统的操作来判断进程是否违反了特定的行为模式，以发现文件加密风险。但是它运行在系统的内核层，如果没有设备制造商的协助很难部署在 Android 平台上。CryptoDrop[4]提出了一种专门检测加密勒索软件的早期警告系统，能够在文件被加密之前发现潜在风险。另外，HelDroid[5]针对 Android 平台上的勒索软件也提出了一种基于静态分析的检测方法。以上方法虽然都有一定的检测效果，但是大多是针对

1 CryptoLocker Ransomware. https://www.secureworks.com/research/cryptolocker-ransomware.

2 Android phones hit by ransomware. http://bits.blogs.nytimes.com/2014/08/22/android-phones-hit-by-ransomware/.

Windows 平台，而且都是离线检测，难以直接部署在 Android 平台。另外，这些检测系统在发现勒索风险之后，不能给出很好的解释，用户往往无法理解检测结果，影响了方案的可用性。

为了降低勒索软件对 Android 用户的危害，安全检测系统面临两个重要挑战。首先，勒索软件的检测应该是实时在线的，从而能够在造成危害之前就发现勒索风险。一旦设备被锁定或者文件被加密，用户很可能需要通过恢复出厂设置才能解决，而这一定会丢失重要的数据。其次，勒索软件的检测结果应该是用户友好的。在提供检测结果时，专业化的分析或者分类规则将会使普通用户难以理解。因此，检测系统应该将整个潜在勒索软件的风险以自然语言的形式告知用户，以帮助其决策。

本文提出了一种轻量化的勒索软件实时检测方法 RansomGuard，能够在造成不可挽回的损失之前发现风险并报警。首先，本文详细分析了 Android 勒索软件的家族和传播规模，并根据具体的攻击行为对当前的家族进行分类。然后，通过轻量化的静态分析方法，利用模块化规则归纳学习算法对应用的勒索风险进行检测。最后，提出了一种基于证据链的行为表示方法，能够在分类规则的基础上用自然语言的形式描述出被检测应用的完整生命周期，以辅助用户决策。

2 相关工作

近年来，基于 Android 平台的恶意软件检测方法引起了很多安全研究人员的关注。目前的解决方案可以分为两类：离线检测[6-9]和实时检测[10,11]。离线检测旨在对恶意软件进行深入的分析，不需要考虑时间和性能的限制，一般在服务器端完成。例如，CHEX[6]利用静态信息流分析方法来识别 Android 应用的敏感数据传输。DroidAPIMiner[7]在 API（Application Programming Interface, 应用程序编程接口）级别上提取出恶意软件特征并提出轻量化分类器来阻止恶意软件的安装。离线检测方式的优点在于准确率高，缺点在于不能在设备上直接发现风险并阻止。实时检测直接部署在终端上，能够快速发现恶意软件的高风险行为，以避免其造成不可挽回的危害。例如，AppAudit[10]利用动态分析方法来模拟程序的执行，并对程序的状态进行检查。Daniel[11]在智能手机上提出了一种静态检测方法来发现恶意行为。实时检测的优点在于快速发现风险，而缺点在于需要消耗终端性能。

在检测勒索软件方面，实时检测比离线检测更加适合，这样能够在勒索软件造成不可挽回的损失之前及时告知用户。但是，现有的实时方案（例如 AppAudit[10]和 Daniel[11]）在检测勒索软件时效果并不好。这主要是因为现有方案大多通过对比恶意软件与正常软件在行为上的差异来建立模型，而这种差异在勒索软件上表现的并不明显。无论是弹窗还是加密行为，这些在正常软件中都非常普遍。本文通过对勒索软件行为特征的深入分析，设计一种专门化的检测方案，不仅能够实时发现风险，而且可以满足智能终端轻量化的需求。

目前，有一些研究人员针对勒索软件提出了专门的解决方法[2-5]。例如，Kharraz[2]等实现了一种动态分析系统来检测 Windows 勒索软件。虽然能够批量检测，但是它是离线检测，不能直接部署在 Android 平台。Scaife[4]等通过监控用户数据的变化（如熵、后缀名等）来判断文件是否被加密，一旦变化超过一定阈值就会报警。这种检测方法只能检测加密类勒索软件，而且会损坏一定数量的文件。HelDroid[5]通过对 Android 应用的静态分析提出了三种勒索软件特征：锁屏、加密和威胁文本。但是该方案为离线检测，而且特征较少容易被攻击者绕过。

除了方案本身不适用于 Android 平台的实时检测之外，以上针对勒索软件的检测方法都有一个没有解决的问题，即检测结果的可视化。为了帮助用户进行决策，检测系统应该以自然语言的形式对结果进行合理的解释。目前，存在一些方案基于自然语言处理来生成应用相关的安全描述[12-16]。但是，这些方案集中于应用的权限、代码分析和用户评论方面，没有对恶意软件的整个生命周期给出全面的证据链[17][18]。因此，检测结果证据链的生成也是一个亟待解决的关键问题。

3 勒索软件家族及分类

3.1 勒索软件家族

早在 1989 年，第一个勒索软件病毒——AIDS 木马就已经出现，这一木马激发了随后近乎 30 年的勒索软件攻击。最初的勒索软件主要是针对 Windows 平台，直到 2013 年第一款针对 Android 平台的勒索软件（即 Fakedefender）才横空出世。Fakedefender 谎称设备存在严重安全风险，要求用户付费安装指定的“杀毒软件”。然而，这个所谓的“杀

毒软件”其实没有执行任何杀毒操作，只是去掉了安全风险提示而已。近几年，随着 Android 平台的蓬勃发展，针对它的勒索攻击也如雨后春笋般迅速扩张，出现了越来越多的勒索软件家族，例如 Koler、Simplocker、Svpeng 和 PronDroid 等。

在安天实验室[1]的帮助下，本文从论坛、安全报告、博客和开源数据集等渠道收集到了 2721 个 Android 勒索软件样本。通过将这些样本上传到 VirusTotal[2]进行检测，本文对每个样本所属的勒索软件家族进行了统计。从 2014 年到 2017 年，全球范围内接近 15 个基于 Android 的勒索软件家族被研究人员发现，主要遍布于欧洲、俄罗斯、美国和中国等国家。表 1 给出了勒索软件家族的统计信息，并列出了首次发现的时间。

表 1 勒索软件家族统计

序号	家族名称	样本数	发现时间
1	Fakedefender	1	2013-6
2	Jisut	613	2014-6
3	Simplocker	427	2014-6
4	Artemis	13	2014-7
5	Gepew	2	2014-9
6	Koler	434	2014-9
7	Svpeng	57	2014-10
8	ScarePackage	17	2014-10
9	Cokri	18	2015-1
10	Locker	39	2015-4
11	SLocker	745	2015-5
12	Lockerpin	23	2015-9
13	Wipelocker	26	2015-10
14	Fobus	2	2015-10
15	PronDroid	124	2015-10
16	其他	180	N/A

3.2 勒索软件分类

通过对勒索软件样本的分析和总结，本文按照勒索软件的攻击方式将其分为三类：控制设备类、绑架数据类和恐吓用户类。

3.2.1 控制设备类

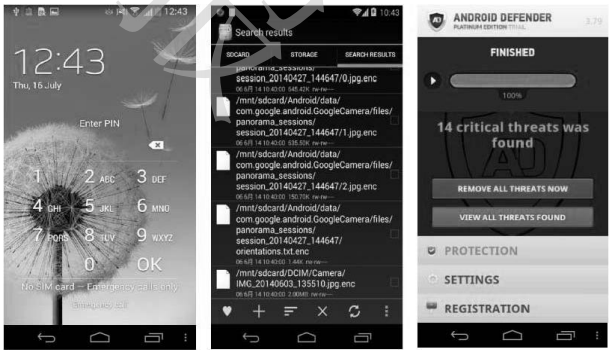
勒索软件对用户进行敲诈勒索的一个最主要方式就是控制受害者的设备。无论在 Windows 平台还是 Android 平台，勒索软件制造者经常恶意指控受害者访问非法内容，从而导致了设备异常。该类勒索软件控制设备的方式包括锁定屏幕、修改 PIN（Personal Identification Number）码、持续弹窗和屏蔽按键等。一旦设备被攻击者控制，用户将无法访问任何数据和任何功能，使得智能设备“变砖”。本文将这类勒索软件称为“锁定勒索”，如图 1 (a)所示。

3.2.2 绑架数据类

勒索软件实施敲诈勒索的第二个重要方法是绑架受害者的数据，包括数据加密、隐私窃取和文件贩卖等。近几年来，智能手机变得越来越私有化，很多重要的数据都保存在这些设备上。因此，攻击者可以通过加密或者窃取用户数据的方式对用户进行敲诈。通常，他们会选择性地加密对用户有价值的文件，例如文档、密钥、短信等。相比普通系统文件，一旦这些有价值文件被加密或窃取，用户支付赎金以降低损失的可能性会更大。本文将这类勒索软件称为“加密勒索”，如图 1(b)所示。

3.2.3 恐吓用户类

除了上述两类，还有一类特殊的勒索软件。它们本身并没有控制设备或者绑架数据，只是利用用户的恐惧心理达到攻击目的。一般情况下，它们会扮演当地政府机构控告用户违反了严重的法律规定或者发现了严重的安全威胁，并给出详细的处罚说明。例如，控告用户访问了色情网站，违反了互联网管理条例等。这些指控往往看起来非常真实，没有经验的用户会感到紧张并试图支付罚款。本文将这类勒索软件称为“恐吓勒索”，如图 1(c)所示。



(a) 控制设备类 (b) 绑架数据类 (c) 恐吓用户类

图 1 勒索软件分类

4 RansomGuard 系统设计

4.1 系统框架

本文提出一种基于 Android 平台的轻量级勒索

1 安天实验室. http://www.antiy.cn/.

2 VirusTotal. http://www.virustotal.com.

软件检测方法，称为 RansomGuard。所提方案不仅能够在勒索软件安装到设备之前发现潜在风险，而且能够将完整的检测过程以自然语言的形式告知用户，以帮助用户做出决策。RansomGuard 的系统框架如图 2 所示，主要包括以下三个功能模块：

（1）特征提取。在用户安装应用之前，利用轻量级的静态分析方法对下载的 APK（Android Package）文件进行风险检测。本模块通过对大量样本的分析，共提取出六组勒索软件的典型特征，包括锁屏、加密、权限、威胁文本、支付方式和网络通信等，这些特征均可以通过对 APK 文件的线性分析得到。

（2）模块化规则归纳学习。本模块利用模块化规则归纳学习算法分别训练得到三类勒索软件的分类规则。通过快速匹配提取到的特征向量，能够有效发现风险并降低终端性能的开销。一旦发现高风险应用，本模块可以终止相应程序的安装，并告知用户。

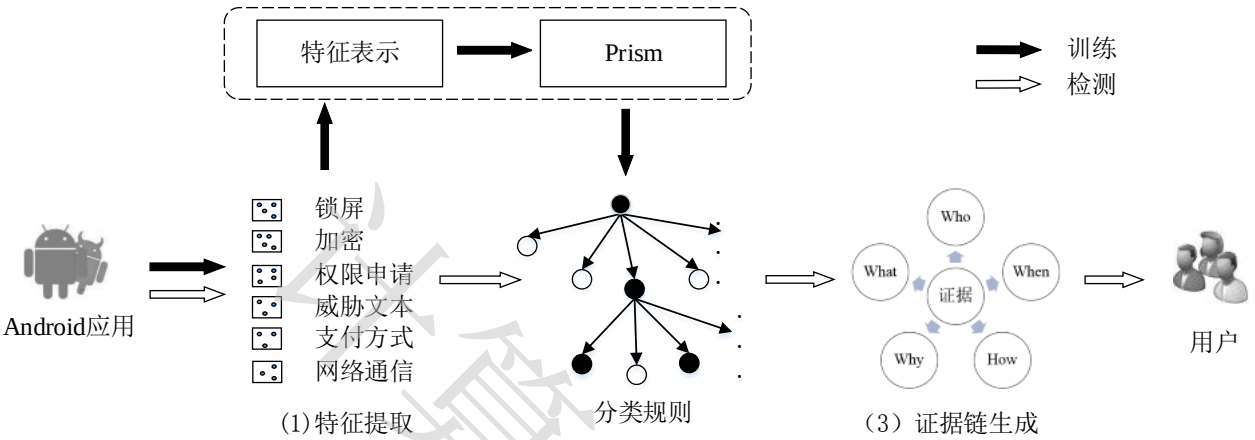


图 2 系统框架图

（3）证据链生成。一旦检测系统发现潜在的勒索风险，本模块将通过证据链的形式告知用户。基于自然语言生成算法，收集到的证据会以用户容易理解的方式进行描述，涵盖外围信息、特征提取和风险评估等安全相关内容。

接下来分别对这三个功能模块进行详细介绍。

4.2 特征提取

在智能手机上提取和分析勒索软件特征的首要前提是轻量化。如果整个过程耗费了大量的时间和资源，超过了用户能够忍受的范围，那么就很可能会被用户跳过并终止后续的检测。因此，本文针对 Android 勒索软件的典型特点提出了六种轻量化的特征集合。这些特征的提取不需要依赖繁琐的程序行为分析方法，而只需要对 Androidmanifest.xml 文件和反编译得到的 dex 文件（Android 平台上 Dalvik 虚拟机的可执行文件）进行线性分析就能够发现勒索风险，从而提高效率。具体特征如下：

（1）锁屏

锁定设备屏幕是勒索软件的典型特征。通过对 Android 开发者文档和大量勒索软件样本的手动分析，本文统计出了五种常用的锁屏方法，如下：

1）全屏悬浮。勒索软件调用 addView()弹出一个悬浮窗，并且设置 WindowManager.LayoutParams 的 flags 属性为“FLAG_FULLSCREEN”。为了使得悬浮窗全屏置顶且无法被关闭，勒索软件需要同时申请 SYSTEM_ALERT_WINDOWN 权限。

2）持续弹窗。勒索软件调用 setCancelable()，并设置其参数为 false，能够阻碍用户的关闭操作。弹出对话框后，用户无法直接关闭该窗体。即使利用实体键返回，仍然可以不断弹出。

3）界面劫持。勒索软件持续监控当前设备的顶层 Activity，判断是否为自身程序。如果不是，便会重新启动。为了覆盖当前的顶层 Activity，需要设置 addFlags 值为“FLAG_ACTIVITY_NEW_TASK”，达到页面置顶的效果。另外，勒索软件利用 Activity 劫持手段的同时，也会结束掉原来的后台进程。

4）按键屏蔽。与 Windows 设备相比，Android 设备包含了一些操作按键，并具有虚拟和实体两种形式。为了阻止受害者关闭或者返回主界面，勒索软件通过改写 onKeyDown()和 onKeyUp()能够实现按键的屏蔽，包括返回键、音量键、菜单键等虚拟按键。这样就造成设备不响应用户的按键动作，达到锁屏的目的。

5）密码修改。通过欺骗受害者激活 Android 系统的设备管理器功能，勒索软件能够在不需要用户确认的情况下给手机设置一个解锁 PIN 码。新设置的 PIN 码以特定的方式生成，具体逻辑由攻击者设计，从而导致用户无法解锁手机。

（2）加密

在对用户文件进行加密时，攻击者不仅可以采用系统提供的 API，也可以自行定制加密函数。因此，根据加密 API 是否存在来判断应用是否为加密类勒索软件的方式容易被攻击者绕过。通过对勒索软件样本的分析，本文发现无论攻击者采用标准或者定制的加密系统，文件访问方式都存在一定的规律。这些文件访问形式在正常应用中很少出现，能够作为识别勒索软件的特征。具体来说，勒索软件加密文件的操作过程分为以下三类：

1）打开-读取-写入-关闭。通过 open 方法打开源文件，利用 read 方法读取内容，调用 encrypt 方法加密数据，将加密后的数据利用 write 方法写入原始文件，然后 close 方法关闭。

2）移动-打开-读取-写入-移回。该方法与（1）非常类似，只不过在进行文件操作前先将源文件移动到其他目录下，加密之后再移回。这种方法产生的加密文件名称与源文件可能不同。

3）读取-创建-加密-删除。将源文件的内容进行读取加密后，重新创建新的文件进行写入。为了防止用户自行恢复，同时删除源文件。

（3）权限

与其他恶意软件的行为相比，勒索软件的攻击方式与正常软件非常类似，例如弹出对话框、加密文件等。另外，普通的恶意软件一定会试图隐藏自身以防止用户察觉，而勒索软件在达到控制效果之后，一定会通知用户以索要赎金。因此，勒索软件在权限申请方面有自己的独特性，可以通过对权限的分析发现潜在的勒索行为。

通过对大量勒索软件和普通恶意软件的对比分析，本文总结了各自申请数量排名前 20 的权限列表。其中，INTERNET、ACCESS_NETWORK_STATE 和 READ_PHONE_STATE 在所有恶意软件样本中都普遍存在，包括勒索软件样本。特别地，勒索软件更倾向于申请设备相关的权限，例如 DISABLEKEYGUARD，SYSTEM_ALERT_WINDOW 和 WALK_LOCK。另外，由于界面劫持的需要，申请 GET_TASKS 权限的勒索软件样本数量是普通恶意软件的三倍之多，KILL_BACKGROUND_PROCESSES 更是达到了十倍。

（4）威胁文本

为了达到更好的勒索效果，攻击者在控制用户设备或者数据之后，会同时弹出一些威胁信息。除了莫须有的指控之外，勒索软件还会给出详细的赎金支付方式。攻击者通常采用的威胁方法包括：检测到色情内容、侵犯版权、违反伦理道德等。而且，这些威胁信息一般会以政府公告的形式给出，使得它们看起来更加可信。例如，PronDroid 声称受害者访问了儿童色情网站，并散播了大量木马病毒。攻击者以政府的名义锁定设备，要求用户及时支付罚款，否则将立案调查。本文通过对大量威胁文本的分析，共提取出 156 个关键词，包括 lock、pron、FBI 和 fix 等。利用 OpenNLP 开源自然语言处理工具，对待检测样本的文本进行语言检测、特征词分析和关键词比对，能够提取出威胁文本。

（5）支付方式

勒索软件的最终目的是非法获得利益，它们会要求受害者向指定账户汇款或者向指定的号码发送扣费短信。随着电子货币的发展，越来越多的攻击者倾向于使用电子货币来索要赎金，例如 Bitcoin、MoneyPak、Paysafecard 和 cashU。这些电子货币一般通过匿名或者伪匿名的方式进行交易，不容易暴露攻击者的行踪。在 2013 年，勒索软件 CryptoLocker 首次使用比特币作为支付方式。可以说，电子货币的发展极大促进了勒索软件的攻击成功率。本文在对威胁文本进行分析时，将这些电子货币的出现也作为特征之一。

（6）网络通信

为了隐藏自己的行踪，勒索软件在网络通信时会利用一些技术手段进行处理。例如，有些勒索软件会利用 Tor（The Onion Router）网络进行远控端通信，甚至要求受害者同样采用 Tor 浏览器支付赎金。而且，有些勒索软件的主控端选择部署在云平台上，例如 Google Cloud Messaging 和 Baidu Cloud Push 等。另外，更高级的勒索软件会利用域名自动生成算法（DGA）[19]来生成主控端的域名，以逃避静态分析工具的检测。本文通过对代码中字符串常量的分析，将 Tor、黑名单 IP 地址和 DGA 自动生成的特殊域名作为关键词并提取出相应特征。

4.3 模块化规则归纳学习算法

模块化规则归纳学习算法旨在从大量的经验数据中归纳抽取出一般的判定规则和模式，是从特殊情况推导出一般规则的学习方法[20]。模块化规则

归纳学习的目标是形成能解释已知事实和预见新事实的一般性结论，能够以“*IF…THEN*”规则的形式表示分类规则。与普通的恶意软件相比，勒索软件的目标更加明确，其表现出的特征也相对固定。因此，可以利用归纳学习算法分别训练得到三种勒索软件类别的分类规则并部署在终端上，以满足智能终端对轻量化的要求。

目前，模块化规则归纳学习算法主要包括 RIPPER、FOIL 和 Prism 等。RIPPER 算法是一种通用的分类规则生成算法，在包含大量噪声数据时具有很好的性能。RIPPER 的主要缺点是当规则本身变得很复杂（即特征性较多）时，算法本身的效果大大下降。FOIL 采用一般到特殊和特殊到一般的双向搜索策略，在训练数据存在噪声的情况下，算法分类的准确率会受到影响。实验研究表明（见 6.2 节），Prism 系列算法能够在有噪声的训练数据集中得到不错的检测准确率，而且分类规则的表示非常容易理解和操作。因此，本系统选择 Prism 算法作为最终的分类器。

模块化规则归纳学习算法是对决策树算法的改进，其分类规则的表示形式比决策树形式更加直观，例如：

$$IF\ f_1 = 1 \cap f_2 = 2\ THEN\ Class = y_1$$

$$IF\ f_3 = 2 \cap f_4 = 1\ THEN\ Class = y_2$$

模块化规则归纳学习算法要求每条分类规则都要保证尽量覆盖所有正例，并且尽量少地覆盖反例。这样的特性非常有利于勒索软件的检测。如果待检测应用的特征满足某个勒索软件类别的分类规则，那么就是该规则的正例，否则就是其反例。由于勒索软件危害的严重性和难以恢复性，检测系统应该在保证漏报率非常低的前提下，尽量降低误报率。

4.3.1 特征表示

为了表示勒索软件的特征和对应的类别，本文将 3.2 节中定义的三种勒索类别称为值域空间 ，将 4.2 节中提取的六种特征集称为属性空间 ，将收集到的训练样例集合称为样本空间 。

对于样本空间 中的每一个样本 ，拥有多个属性空间 中的特征属性 ，对应的值域为 。因此，训练集中的全部样例可表示为 *m* 个元组

$$D = (< x_1, y_1 >, \cdots < x_m, y_n >)$$

式中， $x_i(i = 1,2,$ 属于样本空间， $y_k(k = 1$ 属于类别值域空间。分类规则以属性-值配对的形式出现，用 a_i表示， $a_i(i = 1,2,$为属性空间 中的一个具体特征， $v \in$ 表示当前特征是否存在。

给定上述的特征表示和训练集，模块化规则归纳学习算法得到的分类规则形式为

$$IF\ f_1 \cap f_2 \cap \cdots \cap f_l\ THEN\ Class = y_k$$

其中，分类规则的前半部分是勒索软件特征的逻辑合取， 用于判断待检测应用的特征取值是否满足条件。分类规则的后半部分给出检测结果，即 = 控制设备类， =绑架数据类， =敲诈勒索类。

4.3.2 基本算法描述

本系统利用 Prism 算法来训练 Android 勒索软件的分类规则。作为一种典型的模块化规则归纳学习算法，Prism 算法由 Cendrowska[21]于 1987 年首次提出。经过近 30 年的改进，逐渐形成了 Prism 算法系列。具体的归纳步骤如过程 1 所示。

过程 1. Prism 算法基本步骤.

Step1 对每一个类别 ，计算条件概率 $p(Class = y_j|a_i$，其中， a_i表示属性-值配对；

Step2 选择步骤 1 中的概率最大值作为当前分类规则的一个子项，将所有满足该条件的样本重新构建为训练子集 ；

Step3 在训练子集 上重复执行步骤 1 和 2，直到其中仅包含类别为 的训练样本，对应的分类规则即为步骤 2 确定的最大属性-值配对；

Step4 从 中删除步骤 3 对应规则覆盖的所有样本，得到训

练子集 ；

Step5 重复执行步骤 1~4，直到当前类别 的所有样本都从训练集 中删除。

算法执行过程中，条件概率 $p(Class = y_j|a_i$ 表示所有符合 a_i条件的训练样例中，勒索软件类别为 的样例所占的比率。每一个分类规则的归纳都从完全的数据集 开始训练，这样能够保证不同类别之间是相互独立的，不存在先后关系。另外，当出现多个 $p(Class = y_j|a_i$ 最大时，优先选择覆盖样例最多的作为分类规则。最终得到的分类规则形式如图 3 所示。

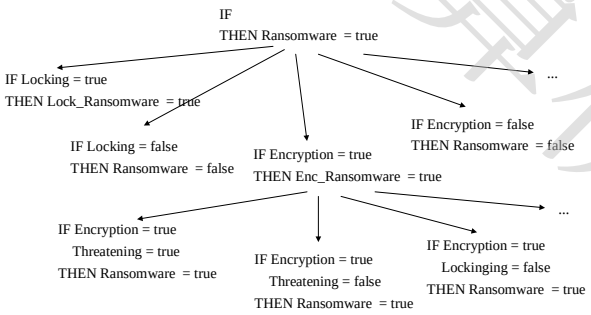


图 3 分类规则示例

4.4 证据链生成

目前，大部分的恶意软件检测方案都存在一个很大问题，即无法向用户解释检测结果，从而导致普通用户难以决定是否应该停止当前应用的执行。本文利用自然语言生成技术来解决这个问题，能够根据采集到的外围信息和分析到的内部信息自动生成普通用户可读的恶意行为描述，从而帮助用户做出决策。具体来说，本方案主要提供以下五个证据：Who、When、Why、How 和 What，简称 4W1H。4W1H 的具体含义如表 2 所示：

表 2 证据链组成 4W1H 的含义

类型	含义
Who	待检测应用的名称
When	何时安装、分析和报警？
Why	为什么该应用存在勒索风险？
How	该应用的具体勒索行为包括哪些？
What	该应用访问了哪些文件或资源？

本文将用户比作法官，拥有自身设备的绝对控制权。检测系统作为诉讼方，当发现异常行为时能够发出警告。勒索软件作为被告方，存在高风险行为。当检测系统发现某个应用存在风险时，需要提供一套完整的证据链，涵盖从下载，到安装，再到静态分析的完整生命周期。证据链提交给用户之后，由用户作为法官决定如何量刑：直接卸载或者放行。这其中也涉及外围信息的收集工作，包括黑名单/白名单的设计，可以减少用户的频繁操作。另外，如果将没有经验的用户直接作为法官看待，可能存在误判风险，所以系统进一步给出其他用户的选择作为参考。例如：80%的用户选择卸载。这样，用户就无形中增加了陪审团，由陪审团的共同建议来辅助决策。在下一节，本文将会提供生成证据链更多的设计细节。

5 基于自然语言生成的证据链表示

本节介绍一种新的基于自然语言生成技术的证据链表示方法。该方法将分类规则作为输入，将证明存在勒索风险的自然语言句子解释作为输出，有利于不具有安全知识背景的普通用户理解。

5.1 自然语言生成概述

自然语言生成（Natural Language Generation, NLG）的目标是根据一些机器“可读”的关键信息，经过一个规划过程，自动产生出人类“可理解”的自然语言句子。NLG 技术已经相对比较成熟，本文采用自然语言生成的经典管道框架[22]，包括三部分：内容规划、句子规划和表层实现，如图 4 所示。

内容规划主要用于确定文档结构和所要传达出的信息，包括结构构造和内容确定。通过输入分类规则的前件、后件项集，从知识库中调用预定义的解释模式，生成单项句子解释。句子规划主要包括选词、生成表达式和聚合，以单项句子解释作为输入，调用知识库中的合并规则，生成概括的自然语言句子。表层实现主要用于将句子的抽象表示转换成普通用户能够理解的真正文本，包括内容实现和结构实现。本文采用基于解释模板的表层实现，以概括句子解释、多项句子解释作为输入，生成分类规则的自然语言表示。

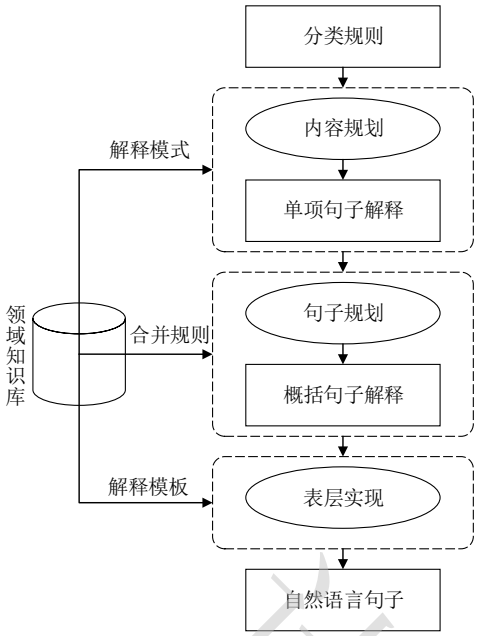


图 4 自然语言生成经典管道框架

5.2 分类规则表示算法

从图 4 中可以看出，领域知识库是自然语言生成方法的重要组成部分，在分类规则的自然语言表示过程中起着关键作用。领域知识库主要由三部分构成，包括解释模式、合并规则和解释模板。

5.2.1 解释模式

在中文的语言结构中，大多数句子都是由主语、谓语和宾语三部分构成，因此，本文采用形如资源描述框架（Resource Description Framework, RDF）语义关系三元组<subject, predicate, object>的形式，设计了通用的分类规则解释模式：

定义 1 解释模式 $pattern(s,$。其中，s 表示分类规则的主语，p 表示分类规则的谓语，o 表示分类规则的宾语。

以控制设备类的勒索软件样本集上学习到的分类规则“*IF* 持续弹窗 *THEN* 锁定勒索”为例，解释模式为 $pattern(s,$，s 为“控制用户设备类的勒索软件”，p 为“包含”，o 为变量“持续弹窗”。将该分类规则代入对应的解释模式后，得到单项自然语言句子解释“控制用户设备类的勒索软件包含持续弹窗特征”。

5.2.2 合并规则

每条分类规则往往不只含有一个特征，因此，为了保证生成的句子自然顺畅，需对它们进行适当地合并。针对勒索软件检测时可能存在的特征集合，本文定义了三种通用的合并规则：

规则 1 解释模式主语 不同，谓语 和宾语 相同，则合并规则为：

$$pattern(s_1,p,o)+pattern(s_2,p,o)=pattern(s_1\ s_2,p,o)$$

例如：两个分类规则解释分别为 *pattern*(控制设备，含有，持续弹窗)和 *pattern*(绑定数据，含有，持续弹窗)，经过规则 1 的合并后，表示的自然语言句子为“控制设备和绑定数据类的勒索软件都含有持续弹窗特征”。

规则 2 解释模式谓语 不同，主语 和宾语 相同，则合并规则为：

$$pattern(s,p_1,o)+pattern(s,p_2,o)=pattern(s,p_1\ p_2,o)$$

例如：两个分类规则解释分别为 *pattern*(绑架数据，删除，文件)和 *pattern*(绑定数据，重写，文件)，经过规则 2 的合并后，表示的自然语言句子为“绑定数据类的勒索软件删除和重写文件”。

规则 3 解释模式宾语 不同，主语 和谓语 相同，则合并规则为：

$$pattern(s,p,o_1)+pattern(s,p,o_2)=pattern(s,p,o_1\ o_2)$$

例如：两个分类规则解释分别为 *pattern*(控制设备，含有，持续弹窗)和 *pattern*(锁屏勒索，含有，修改 PIN 码)，经过规则 3 的合并后，表示的自然语言句子为“控制设备类的勒索软件同时含有持续弹窗和修改 PIN 码特征”。

5.2.3 解释模板

为了保证对 4.3 节学习得到的分类规则解释的完整性，需引入支持度和置信度两个重要参数到自然语言句子解释中，分别用 和 表示。对于分类规则 $IF\ f_1 \cap f_2 \cap \cdots \cap f_l\ THEN\ Class$，具有支持度 表示类别 包含 中任何一个特征的百分比，具有置信度 表示类别 同时包含 $f_1 \cap f_2 \cap \cdots$所有特

征的百分比。支持度和置信度是自然语言解释中不可或缺的组成部分，能够以概率的形式强调存在勒索可能的风险高低。本文设计了两种通用的解释模板：

模板 1 *template1(patterns,)*与支持度有关，为分类规则的支持度。

例如：存在规则 *pattern*(控制设备，含有，持续弹窗)和支持度 =60%后，表示的自然语言句子为“60%的控制设备类勒索软件含有持续弹窗特征”。

模板 2 *template2(pattern1, pattern2,)*与置信度有关， 为分类规则的置信度。

例如：存在两个规则 *pattern1*(控制设备，含有，持续弹窗)、*pattern2*(控制设备，含有，修改 PIN 码)和置信度 =40%，代入模板 2 后表示的自然语言句子为“如果控制设备类的勒索软件含有持续弹窗特征，则也会含有修改 PIN 的特征，其可能性为 40%”。

事实上，对于普通用户来说，概率问题也是不易理解的。因此，本文通过设置阈值把支持度和置信度相对应的概率也转换成了自然语言词汇，具体设置如表 3 所示：

表 3 概率转换规则

支持度、置信度概率值	转化后的词语
、 < 20%	很小部分
20% < 、 < 50%	将近一半
50% < 、 < 80%	超过一半
80% < 、 100%	绝大部分

5.3 证据链生成

5.3.1 算法

证据链生成算法的基本思想是将分类规则中的每一个特征与对应类别的概念属性、概念关系进行匹配，根据解释模板和分类规则，将匹配关系映射到具体的解释模式上生成自然语言句子解释。具体步骤如过程 2 所示。

过程 2. 证据链生成过程.

Step1 定义勒索软件检测的解释模式 *pattern(s,*，并确定解释模板和具体的合并规则；

Step2 调用 **Step1** 中的解释模板 *pattern(s,*将分类规则中的每一个特征项和对应的勒索软件类别代入 *pattern* 中，依次生成多个单项句子解释 *pattern_i*；

Step3 进一步调用 **Step1** 中定义的合并规则，对 **Step2** 中输出的多个单项句子解释 *pattern_i* 进行合并，得到多项句子解释 *pattern*；

Step4 最后将支持度 、置信度 和 **Step3** 中生成的多项句子解释 *pattern* 代入 **Step1** 中确定的模板 *template1* 和 *template2* 中，生成证据链的自然语言表示。

5.3.2 算法举例

为了生成勒索软件的完整证据链，本系统共包括四类解释模式：立案、侦查、排除合理嫌疑和其他量刑情节，如图 5 所示。

（1）立案，即应用的传播途径和安装方式。本部分主要回答 Who、When 和 How 的问题，包括应用何时安装？从哪里下载？下载后是否自动安装？安装时申请的默认权限等。该模式可以通过对系统的广播事件进行监控得到。

（2）侦查，即应用的静态分析和动态监控。本部分主要回答 Why、How、When 和 What 的问题，包括应用的代码分析、运行过程监控等，以说明应用的哪些敏感操作导致报警的产生。该模式通过代码的静态分析得到。

（3）排除合理嫌疑，即应用的动机分析。本部分主要回答 Who 和 Why 的问题，包括白名单/黑名单的维护、用户结果统计等。该模式通过检测系统对应的云端服务器提供。

（4）其他量刑情节，即应用的额外信息。本部分主要回答 Who 的问题，包括应用的下载量、用户评论、开发者信誉等，能够从侧面反映该应用非法的概率。该模式通过对官方应用市场和非官方应用市场的抓取获得。

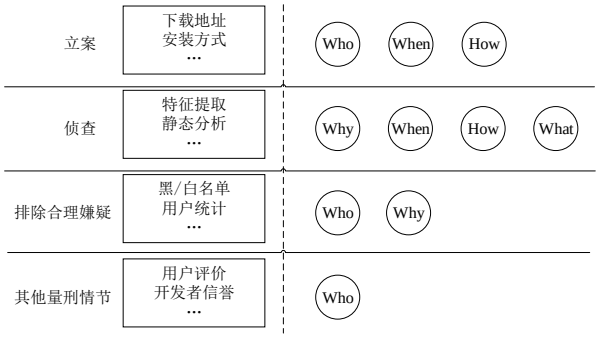


图 5 四类解释模式与证据链的对应关系

下面以分类规则“*IF* 弹窗 不可关闭 屏蔽按键 *THEN* 控制设备, [= 80%, = 90%]”为例，运用上述方法将其解释成自然语言句子。定义的解释模式为 $pattern(s,$，s 为控制用户设备类的勒索软件，p 为包含，o 为变量，$o \in \{a_1 = 弹窗，a_2 = 不可关闭，a_3 = 屏蔽按键\}$。根据合并规则，*pattern*(控制用户设备类的勒索软件，包含，a_1、a_2)的自然语言句子解释为“80%的控制用户设备类的勒索软件包含弹窗、不可关闭或屏蔽按键三个特征之一；如果一个应用同时包含弹窗、不可关闭和屏蔽按键三个特征，那么它为控制设备类勒索软件的概率为 90%”。

进一步结合上述四类解释模式，对待检测应用的完整自然语言解释如下：该应用从非官方应用市场下载，于 2017 年 4 月 10 日上午自动安装到手机上。通过对 AndroidManifest.xml 文件的分析，该应用安装时共申请了 14 个权限，其中 2 个权限能够锁定屏幕，3 个权限能够加密用户文件。通过静态代码分析发现，该应用共调用了 13 个高风险 API，其中 1 个函数能够弹出不可关闭的对话框，1 个函数能够搜索用户文件，2 个函数能够利用 AES 加密算法。通过对 Assets、Resource 等资源文件的分析，该应用存在“色情”、“FBI”和“赎金”等威胁文本。另外，该应用尝试使用 Tor 网络进行通信。通过云端数据分析，该应用有很大风险为勒索软件，91%的用户拒绝该应用的安装，建议立刻停止安装并删除源文件。是否继续？

显然，通过对检测结果的可视化表示，对用户来说是简单易懂的。即使是对于不具有安全知识的普通用户，也能帮助他们了解待安装应用中的高风险特征，为进一步的决策提供了指导数据。

6 实验结果

本节从检测准确率、证据链生成可行性和运行性能三个方面对系统进行评估。其中，检测准确率用于验证 RansomGuard 能否正确地发现勒索软件，证据链生成可行性用于验证基于自然语言生成的证据链是否能够被普通用户理解，运行性能用于验证 RansomGuard 的效率和开销。

6.1 实验准备

6.1.1 系统实现

本文在 Androguard[1]的基础上，结合 Java 语言和 Python 脚本设计并实现了轻量级的原型系统 RansomGuard，如图 6 所示。原型系统包括静态分析模块、归纳学习模块、证据链生成模块等。其中归纳学习模块在服务器端进行训练，服务器配置 Inter(R) Xeon(R) E5-2650 CPU 和 128GB 内存。训练得到分类规则之后直接部署在移动设备上进行检测，默认智能设备为小米 5s，运行 Android6.0 系统，CPU 1.5GHZ，2GB RAM 和 16GB ROM。

（a）静态分析界面 （b）证据展示界面

图 6 原型系统截图

在证据链生成模块中，特征本体的构建质量直接决定着生成的自然语言句子的准确性。因此，本文利用国内外研究最火的 Protégé[23]工具进行特征本体的构建，它能够快速定义所有的特征及其属性，并方便各种解释模式的制定。

1 Androguard. https://code.google.com/p/androguard/.

6.1.2 数据集

为了防止恶意软件被非法利用，很多安全厂商都选择不公开已有的样本，这就导致研究人员能够获得的测试样本非常有限。为此，本文从安天实验室、论坛、安全报告和博客等渠道收集到了 2435 个 Android 恶意软件样本，并从开源数据库 Heldroid[5]中进一步获得了 675 个勒索软件。通过比较 MD5 值，本文删除了重复样本，最终获得 2721 个 Android 勒索软件。样本集涵盖了从 2013 年到 2016 年所有已知的勒索软件家族，是目前研究人员能够获得的最大数据集[1]。另外，本文利用 PlayDrone[24] 工具从 Google 官方应用市场爬取到 5396 个样本，作为正常软件数据集。

本文将收集到的勒索软件样本集上传到 VirusTotal 进行检测，并统计了 10 个病毒扫描工具的检测结果，包括：AntiVir、AVG、BitDefender、ClamAV、ESET、F-Secure、Kaspersky、McAfee、Panda 和 Sophos。根据返回结果，本文将所有样本划分为具体的家族，最终共得到了 15 个不同家族，例如 Koler、Simplocker 和 ScarePackage 等。

6.2 检测准确率

为了验证所设计检测系统的准确率，本文分别利用决策树算法和模块化规则归纳学习算法对模型进行训练，包括 C4.5、Prism、RIPPER 和 FOIL。对每一个算法，本实验利用十折交叉验证（10-fold cross-validation）方法来测试算法性能，包括准确率（Accuracy）、弃真率（False Positives）和纳假率（False Negatives）三个指标。通过将数据集等分为 10 份，依次选择 9 份作为训练样本，剩余 1 份作为测试样本。这样，每次训练都会得到评估结果，连续进行多次之后求其均值作为最终算法的评估。

图7给出了四个算法的ROC（Receiver Operating Characteristic）曲线。ROC 曲线是根据一系列不同的二分类方式，以真阳性率（True Positive Rate, TPR）为纵坐标，假阳性率（False Positive Rate, FPR）为横坐标绘制的曲线。TPR 越高，FPR 越低，则算法性能越好。另外，也可通过计算各个算法的 ROC 曲线下的面积（Area Under the ROC curve, AUC）进行比较，AUC 越大，则算法的性能越佳。从图中可以看出，对于所有的归纳学习算法，RansomGuard 都能得到不错的准确率（AUC 值均高于 85%）。其中，Prism 算法的性能最好（AUC=95.3%），而且三种模块化规则归纳学习算法的性能都要明显优于决策树算法。这是因为决策树算法属性结构的知识表示形式会造成逻辑上的冗余，造成“重复子树问题”。由此可见，本系统采用模块化规则归纳学习算法检测勒索软件有非常好的准确率。

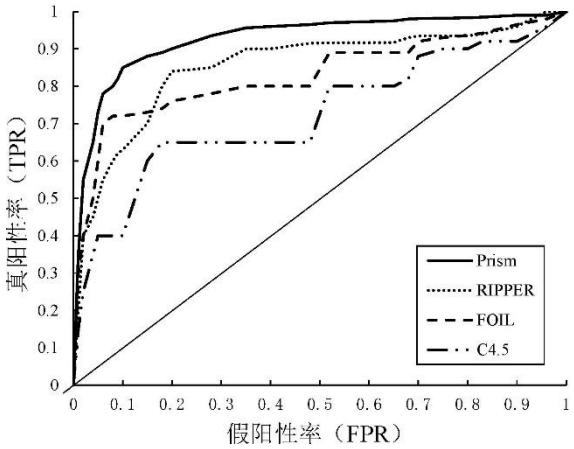


图 7 四个归纳算法的 ROC 曲线对比

本文对系统假正类（False Positive, FP）和假负类（False Negative, FN）的情况进行评估。FP 表示正常软件被错误识别为勒索软件，FN 表示勒索软件被漏报为正常软件。通过分析发现，FP 出现的原因主要分为两类：一是正常软件弹出对话框后调用 setCancelable(false)方法屏蔽了关闭按钮，但是在界面上提供了关闭功能，从而造成检测系统的误报；二是软件开发者在调试时加入了锁定设备的功能，但是在实际运行版本上将这些功能屏蔽。由于静态分析的局限性，这两类正常软件均可能被识别为勒索软件。FN 出现的原因主要为高级的恶意软件攻击者采用代码混淆、加密加壳的方式阻碍程序的静态分析，导致有些勒索软件特征无法正常提取。最终实验表明，本文系统在检测勒索软件时有较低的 FP 和 FN（FP=0.6%和 FN=0.1%），具有较高的实用性。

为了进一步对比所提检测方案与现有工具的区别，本文利用 VirusTotal 提供的 56 个扫描工具对勒索软件样本集进行检测，结果如图 8 所示。从图中可以看出，只有 30%的样本能够被 36 个扫描工具报警，大部分现有的扫描工具对新的勒索软件样本都更新缓慢。即使顶级的杀毒软件厂商，例如 NOD32（95%）、Avast（95%）和 Symantec（81%），检测结果也没有本文所提方案的准确率高。另外，所有的扫描工具都没有对 56 个勒索软件样本报警，而本文方法能够成功发现这些样本的勒索风险。这

1 作者计划在论文发表后将数据集发布，届时研究人员可访问 http://csp.whu.edu.cn/RansomGuard/download.html 获取样本集。

主要是因为这些样本出现的时间相对较新，现存的反病毒公司还没有提取出对应的签名并更新病毒库。从该实验可以看出，本文的实时检测方法具有发现 0-day 恶意软件的能力。

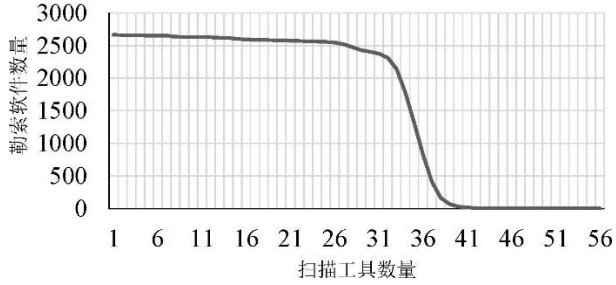


图 8 VirusTotal 检测结果

6.3 证据链生成可行性

为了验证自然语言生成证据链的可读性，本实验在作者单位招募了 50 名志愿者进行测试。这些志愿者包括学生、教师、物业和食堂工作人员，年龄段分布在 20-50 岁之间。而且，有意排除了信息安全相关专业人员，以观察生成结果对普通用户的可读性。测试的目标主要是验证生成的描述能否被普通用户所理解，即检测结果能否帮助用户做出正确决策。

出于效率和人力成本的考虑，本部分随机选择了 50 个应用进行测试，其中 40 个是勒索软件，5 个非勒索软件但是具有较高风险，剩余 5 个为正常软件。首先，本文人工分析了这些样本并提取出各自的行为特征。然后，要求每个志愿者对这些应用进行安装，并利用系统自动进行分析，得到检测结果。当系统执行之后，志愿者需要填写一份调查问卷，告知他们阻止和放行了哪些应用。另外，志愿者还需要对证据链描述可读性的进行评分（1-5），1 表示完全不可读，5 表示非常容易理解。

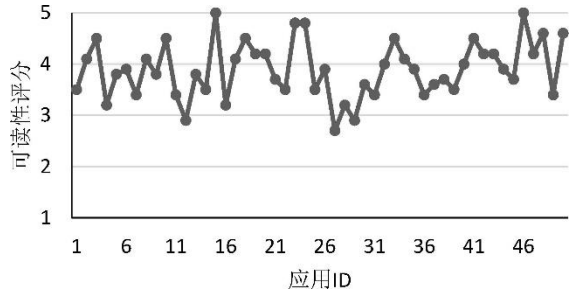


图 9 应用可读性评分

最终，本文共收集到 2115 份志愿者的回复。剔除掉一些无效回复之后，剩余 1913 份结果。图 9 给出了 50 个应用的可读性评分结果。从中可以看出，可读性平均得分为 3.867，而且超过 90%的用户给出的评分超过 3 分，说明自然语言生成的证据链能够被大部分普通用户所理解。另外，在检测出风险并弹出证据链结果之后，大部分用户能够在 30 秒之内做出决定。对于勒索软件应用，全部用户都选择了拒绝安装。对于高风险应用，有 85%的用户选择拒绝安装。这表明系统的检测结果能够帮助用户做出正确的决策。

6.4 运行性能

对于实时检测系统来说，时间开销和内存消耗是两个重要评估指标。本实验利用目前市场占有率较高的 5 个智能手机品牌进行测试，包括 Galaxy S7（S7）、小米 5（M5）、HTC One M8（M8）、华为 Mate9（M9）和 Nexus 4（N4）。另外，为了比较应用文件大小对分析时间的影响，本文随机选择了几组不同 APK 大小的应用（<5M、5M-10M、10M-15M 和 15M-20M）进行检测。

表 4 给出了检测系统各个模块运行时间的均值和方差。其中，均值表示各模块的平均执行时间，方差表示各个测试样例执行时间的平均偏差。在静态分析模块，系统平均花费 5 秒钟分析应用并弹出警告。相对整个应用的下载和安装周期来说，时间开销是可以被用户接受的。另外，与实时检测工具 DREBIN[11]相比，本文方案的针对性更强，检测速度也更快（DREBIN 至少需要 10 秒钟才能发现风险）。通过更细粒度的实验发现，该模块最主要的时间开销为 dexdump 工具的执行。该工具需要对整个应用的 dex 文件进行反编译，得到相应的 smail 中间代码。该工具性能的改进是本文下一步的工作。

表 4 系统运行时间（单位：秒）

步骤	特征提取	归纳学习	证据链生成	合计
均值	3.19	0.58	1.25	5.02
方差	1.02	0.27	0.32	1.61

为了评估系统在不同性能设备上的时间开销，本实验进一步在上述五个智能手机上分别运行 RansomGuard 来检测 50 个勒索软件样本。实验结果如图 10 所示。从图中可以看出，即使在性能最差的设备上（即 Nexus 4），本系统仍然能够在 8 秒钟之内发现勒索软件并报警。特别地，Galaxy S7 的时间开销最少，最快能够在 3.4 秒之内分析完毕。

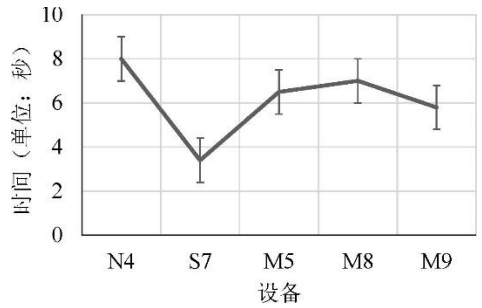


图 10 设备性能不同对时间的影响

图 11 给出了分析时间随着应用大小变化的趋势。从中可以看出，文件越大，分析时间就相对越长。这主要是因为静态分析时间依赖于源代码的复杂度，从而间接依赖于应用 dex 文件的大小。更进一步发现，分析时间并没有随着文件大小成线性增长。原因是文件较大的应用并不一定意味着程序逻辑更复杂，可能只是增加了一些资源文件而已。

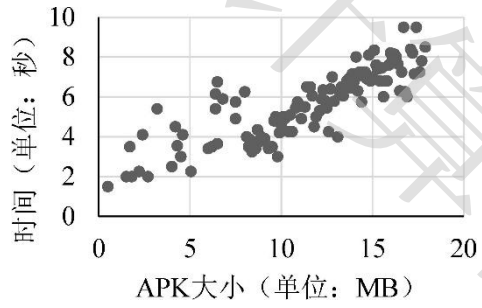


图 11 应用大小对分析时间的影响

一般情况下，由于数据结构的处理复杂，依赖于程序分析的静态检测工具需要消耗相对较多的内存。也正是这个原因导致很多现有方案不能直接部署在终端上运行。本系统在特征提取时，没有采用复杂的程序污点追踪和逻辑分析技术，从而保证了整个检测过程的轻量化。通过实验发现，本系统在检测时仅仅需要 19MB 左右的内存，不足整个系统的 5%，如图 12 所示。这对于性能越来越强的智能终端来说是可以接受的，不会影响到正常使用。

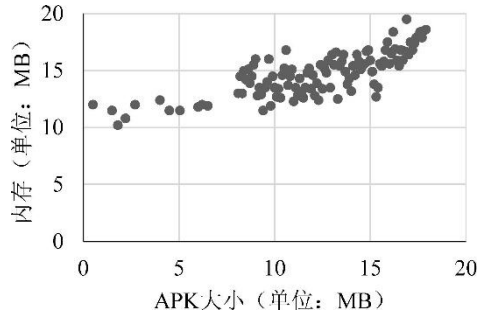


图 12 应用大小对内存消耗的影响

6.5 方法不足与改进方向

通过改善技术，攻击者总能找到方法绕过检测，本系统也不例外。由于 RansomGuard 建立在静态分析之上，缺少动态审计的支持，因此恶意应用通过反射或者代码加密技术能够使得一部分特征无法提取。另外，攻击者也可能通过 JNI 接口调用本地 C/C++代码完成勒索功能。为了弥补动态分析缺失带来的问题，RansomGuard 在提取勒索软件相关特征的同时，也提取了代码混淆和动态加载相关的 API，例如 DexClassLoader.loadClass()和 Cipher.getInstance()。这些特征使得系统能够发现可能存在风险的代码，从而与那些没有被混淆的特征一起作为识别勒索软件的证据。对于 JNI 接口调用的问题，本系统下一步考虑将本地 so 库文件也纳入静态分析的范围之内，这也是未来的一个改进方向。

为了弥补静态分析方法的不足，作者通过结合实时监控技术，对比分析勒索软件在 UI 控件表现上与正常软件的差异，能够达到识别用户意图的目的。例如，勒索软件在加密时不会显示文件选择列表、提示文本以及操作按钮等控件，而正常加密类软件为了提示用户，往往需要明确显示多个组件。这样，攻击者要想绕过用户意图的分析，必然需要精心设计相关界面，从而极大提高攻击成本并降低成功率。

7 结束语

本文提出了一种新颖的 Android 勒索软件检测方法，称为 RansomGuard。通过对 2721 个 Android 勒索软件样本的收集和深入分析，设计了轻量化的特征提取方法，并基于模块化规则归纳学习算法实现实时勒索软件检测系统。同时，为了使得检测结果能够被普通用户理解，本文进一步提出基于自然语言生成的证据链表示方法，对静态分析阶段学习到的分类规则和待检测应用的外围信息进行解释。实验表明，原型系统能够以较高的准确率发现勒索软件，证据链表示结果也能够被用户接受并辅助决策。另外，性能分析结果也证明 RansomGuard 能够在不影响系统性能的前提下，直接部署在 Android 平台上进行实时检测。

参 考 文 献

[1] McAfee Labs. McAfee labs threats report: June 2017. McAfee. Sa

nta Clara,USA, 2017

[2] Kharraz A, Arshad S, Mulliner C, et al. Unveil: A large-scale, automated approach to detecting ransomware// Proceedings of the 25st USENIX conference on Security symposium. Austin, USA, 2016:757-772

[3] Continella A, Guagnelli A, Zingaro G, et al. ShieldFS: A self-healing, ransomware-aware filesystem// Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC). Los Angeles,USA , 2016:336-347

[4] Nolen S, Henry C, Patrick T, et al. Cryptolock (and drop it): Stopping ransomware attacks on users data// Proceedings of the 36th International Conference on Distributed Computing Systems (ICDCS). Nara, Japan, 2016:303-312

[5] Nicolo A, Stefano Z, and Federico M. Heldroid: Dissecting and detecting mobile ransom// Proceedings of the International Symposium on Research in Attacks, Intrusion and Detection (RAID). Kyoto, Japan, 2015:382-404

[6] Lu Long, Li Zhi-Chun, Wu Zheng-Yu, et al. CHEX: Statically vet ting Android apps for component hijacking vulnerabilities//Proceedings of the 19th ACM Conference on Computer and Communications Sec urity (CCS). Raleigh,USA, 2012:229-240

[7] Aafer Y, Du Wen-Liang, Yin Heng. DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android//Proceedings of the Security and Privacy in Communication Network. Sydney, Australia, 2013:86-103

[8] Chen Kai, Wang Peng, Lee Yeonjoon, Wang Xiao-Feng, et al. Finding unknown malice in 10 seconds: Mass vetting for new threats at the Google-McPlay scale// Proceedings of the 24st USENIX conference on Security symposium. Washington, USA, 2015:659-674

[9] Yan L K, Yin Heng. DroidScope: Seamlessly Reconstructing the O S and Dalvik Semantic Views for Dynamic Android Malware Analysis // Proceedings of the 21st USENIX conference on Security symposiu m. Bellevue,USA, 2012:569-584

[10] Xia Ming-Yuan, Lu Gong, Lyu Yuan-Hao, et al. Effective real-time Android application auditing// Proceedings of the 36rd IEEE Symposium on Security and Privacy (S&P). San Jose, USA, 2015:899-914

[11] Daniel A, Michael S, Malte H, et al. Drebin: Efficient and explainable detection of Android malware in your pocket// Proceedings of the 21th Network and Distributed System Security Symposium (NDSS). San Diego, USA, 2014:1-15

[12] Sufatrio, Chua Tong-Wei, Tan Darell J.J., and Thing Vrizlynn L.L. Accurate specification for robust detection of malicious behavior in mobile environments// Proceedings of the 20st European Symposium on Research in Computer Security (ESORICS). Vienna, Austria, 2015:355-375

[13] Kong De-Guang, Cen Lei, and Jin Hong-Xia. AUTOREB: Automatically understanding the review-to-behavior fidelity in Android applications// Proceedings of the 22th ACM Conference on Computer and Communications Security (CCS). Denver,USA, 2015:530-541

[14] Huang Jian-Jun, Li Zhi-Chun, Xiao Xu-Sheng, et al. SUPOR: Precise and scalable sensitive user input detection for Android apps// Proceedings of the 24st USENIX conference on Security symposium. Washington,USA, 2015:977-992

[15] Zhang Mu, Duan Yue, Feng Qian, and Yin Heng. Towards automatic generation of security-centric descriptions for Android apps// Proceedings of the 22th ACM conference on Computer and Communications Security (CCS). Denver,USA, 2015:518-529

[16] Pandita R, Xiao Xu-Sheng, Yang Wei, et al. WHYPER: towards automating risk assessment of mobile applications// Proceedings of the 22st USENIX conference on Security symposium. Washington, USA, 2013:527-542

[17] Zhao Zhi-Yan, Shi Wen-Chang. Trustworthiness Analysis of Digit al Evidences Based on Chain of Evidences. Computer Science, 2016, 43(7):131-135 (in Chinese)
(赵志岩, 石文昌. 基于证据链的电子证据可信性分析. 计算机科学, 20 16, 43(7):131-135)

[18] Jia Zhi-Cheng, Bai Jian-Jun. Computer forensics and the process chain of evidence. Automation & Instrumentation, 2014, 8:124-126
(贾志城, 白建军. 计算机取证及其证据链形成. 自动化与仪器仪表, 20 14, 8:124-126) (in Chinese)

[19] Plohmann D, Yakdan K, Klatt M, et al. A Comprehensive Measu rement Study of Domain Generating Malware//Proceedings of the 25st USENIX conference on Security symposium. Austin,USA, 2016:263-2 78

[20] Bramer M. Principles of data mining. 2nd ed. Berlin,Germany: Sp ringer, 2013

[21] Cendrowska J. PRISM: An algorithm for inducing modular rules. International Journal of Man-Machine Studies, 1987, 27(4):349-370.

[22] Reiter E, Dale R. Building natural language generation systems. New York, USA: Cambridge University Press, 2000

[23] Noy F, Fergerson N, Musen M A. The knowledge model of protege// Proceedings of the 12th International Conference on Knowledge Acquisition, Modeling and Management. Juan-les-Pins, France, 2000:17-32

[24]Viennot N, Garcia E, Nieh J. A measurement study of google play// Proceedings of the International Conference on Measurement and Modeling of Computer Systems. Austin,USA, 2014:221-233

Wang Chi-Heng, born in 1990, ph. D. candidate. His research interests focus on Mobile Security.

Chen Jing, born in 1981, ph. D., professor, Ph. D. supervisor. His research interests include Cloud Computing, Cryptography, and Mobile Security.

Chen Xiang-Yun, born in 1992, M.S. His research interests

focus on Mobile Security.

Du Rui-Ying, born in 1964, ph. D., professor, Ph. D. supervisor. Her research interests include Cryptography and Mobile Security.

Background

In recent years, we witnessed a drastic increase of ransomware, especially on the popular platforms such as Android. Generally, ransomware extorts the victims for a sum of money by taking control of their devices or files. In light of their rapid growth, there is a pressing need to develop effective solutions. However, the research community is still constrained by the lack of a comprehensive dataset, and there exists no insightful understanding of mobile ransomware in the wild.

Thus, this paper first systematically characterize the Android ransomware samples from several aspects, including timeline, classification, and features. Furthermore, it makes light-weight and real-time detection of ransomware on mobile device directly. Besides, to help users understand the detection result, this paper studies the basic concept of evidence chain and then introduce the generation method. To the best of our knowledge, our paper is the first work to focus on 2721 Android ransomware samples and present a systematic study of their timeline, classification, and features. Furthermore, we take the first step to detect Android ransomware in real-time mode, and perform a novel evidence chain generation approach.

This work is supported by the National Natural Science Foundation of China (No. 61572380, No. 61772383, No. 61702379) and the National Program on Key Basic Research Project (No. 2014CB340600).