软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(1):42−68 [doi: 10.13328/j.cnki.jos.005320]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

软件与网络安全研究综述*

刘 剑[1,2], 苏璞睿[3], 杨 珉[4], 和 亮[3], 张 源[4], 朱雪阳[5], 林惠民[5]

[1](中国科学院 信息工程研究所 网络测评技术重点实验室,北京 100195)

[2](北京市网络安全技术重点实验室(中国科学院 信息工程研究所),北京 100195)

[3](中国科学院 软件研究所 可信计算与信息保障实验室,北京 100190)

[4](复旦大学 软件学院,上海 201203)

[5](计算机科学国家重点实验室(中国科学院 软件研究所),北京 100190)

通讯作者: 朱雪阳, E-mail: zxy@ios.ac.cn

摘 要: 互联网已经渗入人类社会的各个方面,极大地推动了社会进步.与此同时,各种形式的网络犯罪、网络窃密等问题频繁发生,给社会和国家安全带来了极大的危害.网络安全已经成为公众和政府高度关注的重大问题.由于互联网的大量功能和网络上的各种应用都是由软件实现的,软件在网络安全的研究与实践中扮演着至关重要的角色.事实上,几乎所有的网络攻击都是利用系统软件或应用软件中存在的安全缺陷实施的.研究新形势下的软件安全问题日益迫切.从恶意软件、软件漏洞和软件安全机制这 3 个方面综述了国内外研究现状,进而分析软件生态系统面临的全新安全挑战与发展趋势.

关键词: 软件;网络安全;恶意软件;软件漏洞;软件安全机制

中图法分类号: TP311

中文引用格式: 刘剑,苏璞睿,杨珉,和亮,张源,朱雪阳,林惠民.软件与网络安全研究综述.软件学报,2018,29(1):42−68. http://www.jos.org.cn/1000-9825/5320.htm

英文引用格式: Liu J, Su PR, Yang M, He L, Zhang Y, Zhu XY, Lin HM. Software and cyber security—A survey. Ruan Jian Xue Bao/Journal of Software, 2018,29(1):42−68 (in Chinese). http://www.jos.org.cn/1000-9825/5320.htm

Software and Cyber Security—A Survey

LIU Jian[1,2], SU Pu-Rui[3], YANG Min[4], HE Liang[3], ZHANG Yuan[4], ZHU Xue-Yang[5], LIN Hui-Min[5]

[1](Key Laboratory of Network Assessment Technology, Institute of Information Engineering, The Chinese Academy of Sciences, Beijing 100195, China)

[2](Beijing Key Laboratory of Network Security Technology (Institute of Information Engineering, The Chinese Academy of Sciences), Beijing 100195, China)

[3](Trusted Computing and Information Assurance Laboratory, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[4](School of Software, Fudan University, Shanghai 201203, China)

[5](State Key Laboratory of Computer Science (Institute of Software, The Chinese Academy of Sciences), Beijing 100190, China)

* 基金项目: 国家自然科学基金(61572483, 61572481, 61602123, 61572478, U1636204, 61602457); 上海市青年科技英才扬帆计划(16YF1400800)

Foundation item: National Natural Science Foundation of China (61572483, 61572481, 61602123, 61572478, U1636204, 61602457); Shanghai Sailing Program (16YF1400800)

收稿时间: 2016-12-22; 修改时间: 2017-02-08; 采用时间: 2017-06-19; jos 在线出版时间: 2017-07-12

CNKI 网络优先出版: 2017-07-12 16:17:02, http://kns.cnki.net/kcms/detail/11.2560.TP.20170712.1617.018.html

本文作者贡献说明:刘剑、苏璞睿和杨珉负责主体内容的组织及撰写,为共同第一作者;和亮与张源参与资料收集与撰写;朱雪阳负责合稿及部分内容的撰写;林惠民发起并主持本文撰写

Abstract: The Internet has penetrated into all aspects of human society and has greatly promoted social progress. At the same time, various forms of cybercrimes and network theft occur frequently, bringing great harm to our society and national security. Cyber security has become a major concern to the public and the government. As a large number of Internet functionalities and applications are implemented by software, software plays a crucial role in cyber security research and practice. In fact, almost all cyberattacks were carried out by exploiting vulnerabilities in system software or application software. It is increasingly urgent to investigate the problems of software security in the new age. This paper reviews the state of the art of malware, software vulnerabilities and software security mechanism, and analyzes the new challenges and trends that the software ecosystem is currently facing.

Key words: software, cyber security, malware, software vulnerabilities, software security mechanism

互联网已经渗透到人类社会的各个方面,极大地推动了社会进步.但与此同时,各种形式的网络犯罪、网络窃密等问题频繁发生,给社会和国家安全带来了极大的危害.网络安全已经成为公众和政府高度关注的重大问题.由于互联网的大量功能和网络上的各种应用都是由软件实现的,软件在网络安全的研究与实践中扮演着至关重要的角色.

随着互联网的发展,由软件引起的安全问题波及面越来越广,厉害关系越来越大,影响层次也越来越深.根据国际权威漏洞发布组织 CVE 的统计,1999 年发现的软件漏洞数量不到 1 600 个;而 2014 年,新发现的软件漏洞数量已接近 10 000 个.2016 年 6 月,国家互联网应急中心(CNCERT)发布的互联网安全威胁报告显示,境内感染网络病毒的终端数近 297 万个,被植入后门的网站数量 8 815 个,境外 3 637 个 IP 地址通过植入后门对境内 66 186 个网站实施远程控制[1].2016 年 4 月,CNCERT 监测到一个名为“Ramnit”的网页恶意代码,该恶意代码被挂载在境内近 600 个党政机关、企事业单位网站上,一旦用户访问网站就有可能受到挂马攻击,这对网站访问用户的 PC 主机构成安全威胁,对国家安全及政府公信力造成重大影响[2].

随着互联网进入移动互联阶段,人类的社会活动与网络空间的融合更加深入.透过泛在网络和各型感知设备,人、机、物彼此交织,密不可分,而软件作为人类智慧的二进制投射,操控硬件平台,提供着各型服务,在网络空间中的枢纽特性被进一步强化.2016 年 10 月 21 日,恶意软件 Mirai 控制的僵尸网络对美国域名服务器管理服务供应商 Dyn 发起 DDoS 攻击(分布式拒绝服务攻击),由于 Dyn 为 GitHub、Twitter、PayPal 等平台提供服务,攻击导致这些网站无法访问.本次 Mirai 的攻击目标与以往控制服务器或个人 PC 机的方式有所不同,它主要通过控制大量的物联网(IoT)设备,如路由器、数字录像机、网络摄像头等,形成僵尸网络来进行大规模协同攻击,造成特别严重的危害[3].研究新形势下的软件安全问题日益迫切.

随着平台的发展,软件形态产生了重要变化.软件的目标由最初的提供基本功能进化为提供更好的服务和体验;围绕多种异构互联的硬件平台,形成了差异化的软件群体;软件核心服务往往通过云平台完成部署,功能和服务在云和端之间体现出很强的交互性和个性化;软件普遍基于开放式的第三方开源组件快速构造,开发周期和版本迭代周期明显缩短,导致软件开发门槛显著降低;出现集中式软件分发渠道,可以将软件快速传播给大量的用户,使得软件的受众范围明显增大;软件系统之间的协同配合显著增强,敏感数据的网间流动大幅增加.

在软件形态的这种演变趋势下,软件安全逐渐演变为整个软件生态系统的体系安全,涵盖了平台、软件和数据等诸多维度,对国家网络空间安全和公众权益的重要性与日俱增.目前,软件生态系统面临的威胁呈现出如下特点.

1) 软件生产环节引入多种新型攻击面.

软件生产过程中,代码复用程度越来越高,各种开发包、核心库的应用越来越广泛.一些攻击者利用开源代码植入后门,这些代码的复用度越高,其中的后门影响范围越广,为攻击者创造新的攻击途径提供了机会;软件之间协同越来越普遍,软件中的一些服务共享、数据共享越来越多,单一软件产品的安全已经不仅仅影响其功能或服务的安全,还会直接影响到其他一系列软件功能或服务的安全,从单一软件产品服务而言,其可能被攻击的渠道增加;软件开发过程、分发过程在逐步发生变化,依赖的工具手段越来越多,各种支撑环境、开发工具和分发渠道是否安全可信,也直接关系到出产软件产品的安全,造成软件功能或服务可能被攻击的环节越来越多.

2) 漏洞的形成机理、利用方式、危害程度出现新的特点.

软件之间的依赖性导致漏洞的危害由“软件内”蔓延到“软件间”,为攻击者提供了更灵活的渠道,对防御者提出了更复杂的分析与防御难题;更加灵活、复杂的软件功能也为攻击者提供了更多的攻击方式,以致新的漏洞模式不断涌现,逻辑型漏洞越来越普遍;而软件漏洞的多样化以及软件之间的依赖关系增强也带来了漏洞之间的交叉协同,攻击者组合应用多个漏洞进行攻击越来越普遍.

3) 恶意软件的生产方式、攻击方式、对抗能力以及分析方法等方面出现新的特点.

在恶意软件生产方式方面,随着开源软件的发展,恶意软件开始通过重打包、共享库、恶意编译工具链等机制进行生产和传播;新的网络技术也为恶意软件的传播和增强提供了新的途径.在恶意攻击方面,传统平台上的恶意攻击趋向于高度定向化,智能移动平台、物联网设备上恶意软件大量爆发,针对新网络技术的恶意攻击正在形成.此外,恶意软件应用多种对抗技术反分析、反检测,出现以协同攻击和深度攻击为特点的 APT (advanced persistent threat)等高级攻击模式.在恶意软件分析方面,趋向静态和动态分析相融合,以便全面获取样本的行为,同时趋向利用数据分析技术来处理大规模恶意样本.

为了应对这些威胁,学术界近些年在恶意软件、软件漏洞和软件安全机制方面开展了大量的研究工作.本文将在接下来的 3 节里综述这 3 方面的国内外研究现状.第 4 节理清软件生态系统面临的全新安全挑战与发展趋势.

1 恶意软件

恶意软件(malware)是对非用户期望运行的、怀有恶意目的或完成恶意功能的软件的统称[4,5].恶意软件种类繁多,如木马、病毒、蠕虫、DDoS、劫持、僵尸、后门/陷门、恶意编译、间谍软件、广告软件、勒索软件、跟踪软件等.早期恶意软件的概念主要局限于计算机病毒等,但近年来,随着网络平台的发展和网络攻击的多样性,恶意软件的概念已经超越了传统的狭义概念.特别是随着勒索软件、后门/陷门、恶意编译、APT 等新型恶意软件的出现,恶意软件更多地凸显出对期望目标的控制性、专有性、定制性、规模性和破坏性.在强制执行目标对象的非法功能、限制或变更系统正常执行程序功能等方面,它们在某种人为控制专有目标程度上远比计算机病毒体现出更强的攻击性和更大的危害性.随着互联网的普及和广泛应用,网络环境下多样化的传播途径和复杂的应用环境给恶意软件的攻击和传播带来巨大便利,对网络系统及网络上主机的安全构成巨大威胁.总的来说,恶意软件的发展与软硬件平台的发展息息相关,大致可分为单机传播、网络传播和协同攻击这 3 个阶段.在 20 世纪 80 年代,计算机应用以单机为主,计算机之间的数据交换主要借助于磁盘,因此,当时的恶意软件主要是磁盘病毒、文件宏病毒,如 Brain、黑色星期五、Wazzu 等.随着网络技术的发展和应用,计算机之间实现了直接互联,邮件等应用也越来越普遍,随之而来的是 Melissa、Loveetter、Nimda 和 Code Red 等邮件病毒和蠕虫[6]等通过网络传播的恶意软件.这一变化带来的结果就是恶意软件的传播能力大大增强.传统的病毒、木马实施破坏仍以单一的节点单独实施,而僵尸网络的出现则实现了被感染节点之间的协同,可以实施分布式拒绝服务攻击(DDoS 攻击)、多链跳转攻击等大规模的协同攻击.特别是结构化 P2P 僵尸网络[7],它可以高效地管理大规模的节点,进一步提升了僵尸网络的控制能力.

近年来,随着软件向互联化、生态化方向发展,恶意软件在生产、攻击、对抗和传播等方面都有了深刻的变化.一方面,传统平台和体系上的恶意软件对抗正在加剧;另一方面,随着新的软硬件环境、网络技术的发展,新平台中的恶意软件正在兴起;新旧平台上的恶意软件相互协作,出现以协同攻击和深度攻击为特点的 APT 攻击模式.此外,随着开源软件等开发模式的兴起,软件开发无论在平台和环境(例如操作系统、网络),还是在开发工具链本身(例如编译器、第三方库),都大量基于开源或遗留代码,在提高了软件生产率的同时,也出现了通过恶意编译器、第三方库等软件“供应链”进行传播的恶意攻击代码.例如:2015 年 9 月在我国发现的 XcodeGhost 恶意软件,具有窃取用户隐私信息和远程控制的恶意行为.下面首先综述当前恶意软件在不同平台上的攻击和对抗模式,然后介绍恶意软件分析、清除方面的最新研究进展.

1.1 恶意软件攻击

1.1.1 传统软件平台上的恶意软件

随着传统平台上软硬件的多样性发展,恶意软件的开发更有针对性,朝着高度定向的方向发展.这类恶意代码的开发采用环境敏感、休眠行为和条件触发等机制,攻击行为在具备特定的触发条件时才会表现出来.如Stuxnet、Duqu、Gauss 和 Flame 等,都是针对特定平台或者操作系统版本的恶意软件.Xu 等人[8]通过分析近期赛门铁克等安全厂商信息,认为恶意软件采用 Query-then-infect 模式,朝定制化发展成为趋势.恶意软件数量本来已呈指数级增长,定制化的发展趋势给恶意软件分析带来了更大的挑战.为此,他们提出了当前恶意软件分析面临的两个新问题:(1) 给定一个恶意软件样本,如何确定该样本是否是环境敏感恶意代码?(2) 如果是环境敏感的恶意代码,如何确定其目标环境,并触发其恶意行为?基于这样的研究背景,该工作还提出了主动(proactively)、动态(dynamically)以及自适应(adaptively)的恶意软件分析工具 GOLDENEYE.较之相关工作,该工具在时间、效率、性能上均能获得更好的平衡.佐治亚理工大学的团队[9]通过分析现有的恶意代码分析机制,提出了恶意软件定制和强化技术 HIE(host identity-based encryption)和 ISL(instruction set localization).其中,HIE 的思想是恶意软件在投放前根据目标主机系统特征(host ID)进行加密,在执行前进行环境识别.如果目标环境和定制环境不一致,说明其被捕获分析,恶意软件将执行不同的系统行为,从而构造针对特定目标系统的恶意代码.ISL 类似于代码虚拟化技术,主要是通过设计特定的二进制代码和中间代码变换机制来规避对恶意软件的逆向分析.

1.1.2 智能移动平台上的恶意软件

近年来,移动智能系统和移动互联网迅速发展,越来越多的软件运行于移动终端,软件互联网化变得十分流行,移动系统呈现“碎片化”发展的趋势.智能终端的安全风险增大,针对 Android 和 iOS 的恶意软件层出不穷.除熟知的木马、病毒、蠕虫、DDoS、劫持、僵尸、后门之外,移动互联平台也出现了间谍软件、广告软件、恐吓软件、勒索软件和跟踪软件(trackware)等恶意软件[10,11].Zhou 等人[11]系统化地总结了 Android 系统中恶意软件的形态、分类和演化,将恶意软件分为重打包(repackaging)、更新攻击(update attack)、诱惑下载(drive-by download)、提权攻击(privilege escalation)、远程控制(remote control)等 9 种,并进一步针对恶意软件攻击的行为(权限)进行细化,例如针对短信、电话、网络、个人通信录等.分析指出:在所考察的 1 260 个恶意软件样本中,有 1 083 个属于重打包型恶意软件,占比为 86%.Android 上的恶意软件也呈现出一些特有的攻击模式和传播渠道.例如,在恶意攻击方面出现“越狱(root exploit)”[12,13]、困惑代理(confused deputy)攻击[14,15]、合谋攻击(collusion attack)[16,17]、中间人攻击[18,19]等为主的攻击模式.“越狱”是指用户可以获取 Android 操作系统的超级用户权限,某些用户通过 Root 越过手机制造商的限制进行设备定制,卸载预装在手机中的某些应用程序,或运行一些需要超级用户权限的应用程序.“越狱”软件主要通过利用 Android 系统层的代码漏洞来实现攻击,例如,Android Setuid()机制的漏洞等.恶意软件则通过 Root 提升权限,绕过 Android 系统的安全机制.最著名的三大 Android root exploit 恶意软件是 Zimpeflich[12]、Exploid 和 RATC(RageAgainstTheCage).许多其他恶意软件都是在它们的基础上研发的,如 DroidDream、Zhash、DroidKungFu 和 Basebridge[20].Android 应用和传统的 Windows、Linux 系统应用有很多不同的特性,例如四大组件、权限机制(permission)等.Chin 等人[21]发现:Android 应用之间的消息传递机制(message passing)成为恶意软件的主要攻击点,消息常常被监听、修改、替换、注入.他们还系统化地总结了针对 Intent 的主要攻击模式:广播信息窃取(broadcast theft)、Activity 劫持(activity hijacking)、Service 劫持(ServiceHijacking)、恶意广播消息注入(malicious broadcast injection)、恶意 Activity 启动(malicious activity launch)等.提权(permission re-delegation)攻击[22]是针对 Android 权限机制的恶意行为,是指一个拥有较低(较少)权限的应用程序访问拥有较高(较多)权限的组件而不受任何限制的行为.具体的提权攻击实现方法各不相同,有的利用 Intent 盗窃/劫持,有的需要多个应用程序之间的相互配合,例如困惑代理、合谋攻击等.在传播方式方面, Android 生态系统中出现了以 Piggy-packing[23]、第三方库(third-party library)[24,25]、恶意编译工具、刷机等新的传播模式.Piggy-packing 指的是恶意软件开发者通过在正常发布的软件中注入恶意 payload 的方式来传播恶意软件的行为,被注入了恶意代码的应用称为 Piggy-packing 应用.Zhou 等人[23]分析了来自多个应用市

场的 84 767 个 Android 应用,发现在谷歌官方应用市场中,1.0%的应用是 Piggy-packing 应用,在第三方应用市场中,有 0.97%~2.7%不等的应用是 Piggy-packing 应用.通过手工分析,该团队发现,在注入的恶意代码中主要包含两类代码:一类是通过注入恶意广告库来窃取正常广告收入,如 admob、wooboo、youmi 等 15 个广告库常被攻击;另一类是注入恶意代码攻击用户手机,发现了 Geinimi、ADRD、Pjapps 等 5 种重要的攻击代码.在 Android 第三方库研究方面,中国科学院信息工程研究所的研究团队[26]研制了基于特征的、变异敏感的挖掘工具 LibD,对包括 Google Play 在内的 45 个国内外主要的 Android 应用市场的 1 427 395 个 Android 应用进行分析,发现了 11 458 个第三方库.在其工作中,还发现很多第三方库都是多包库(multi-package libraries),共 5 141 个,占比 8.4%.所谓多包库就是一个第三方库由多个已有的包组成.该工作还发现大量第三方库采用混淆(obfuscation)技术,主要是名字混淆(name obfuscation),共 5 000 个;很多库还具有多个不同的版本(或变种),这些变种中,有一些是开发人员迭代开发产生的正常版本,但有很多只修改了部分 opcode,往往注入了恶意代码.此外,中国科学院信息工程研究所的研究团队对 Android 应用市场和 iOS 应用市场第三方库中恶意代码进行了研究[24].通过对 130 万个 Android 应用和 1.4 万个 iOS 应用进行分析,发现了 117 个潜在的 Android 恶意库 PHLib(potentially-harmful libraries),共 1 008 个变种;发现 23 个潜在的 iOS 恶意库,共 706 个变种.该工作还发现,很多恶意库不仅有 Android 版本,还有 iOS 版本.例如,iOS 的恶意库 mobiSage 有多个 Android 版本,其中 6 个是最新发现的,这些库往往从用户手机上窃取音视频或个人隐私数据.上述 Android 恶意代码的发展主要呈现 3 个方面的特点.

(1) 由于移动互联网融合了传统的互联网和电信网,恶意软件除了管理软件信息外,还能获取短信、IEMI (international mobile equipment identity)等重要的用户隐私资源.因此,出现了如伪造电话或短信、恶意扣费等针对用户隐私信息的攻击行为;

(2) 很多恶意移动应用的开发模式、发布模式和软件生态密切相关.例如:互联网服务的应用常常提供第三方库和通用接口,移动应用的开发以重打包等方式开发,恶意广告、跟踪软件随之产生;

(3) 移动平台恶意软件的开发与移动平台体系架构及编程语言等相关.例如,Android 系统采用 4 层架构体系;在编程语言方面,主要包括底层的 C/C++代码和上层的 Java 代码;在安全控制方面,包括 Permission、UID/GID、SELinux 等机制.恶意软件往往利用编程语言的特性或者系统安全机制的漏洞进行攻击,例如通过反射代码进行恶意提权,利用静态 Permission 机制的缺陷收集个人隐私信息等.

1.1.3 物联网平台上的恶意软件

随着网络空间的不断延伸,越来越多的物理设备接入互联网平台,例如传感器、网络摄像头、激光扫描器、网络打印机、全球定位设备等,形成了上规模的物联网.目前,普遍将物联网的体系架构分为 3 个层次:感知层、传输层和应用层.感知层解决对物理世界的数据获取的问题,以达到对数据全面感知的目的;传输层主要通过移动通信网、互联网等网络对数据进行传输;应用层利用高性能计算设备、智能计算技术,解决对海量数据的智能处理问题,以达到信息为人所用的目的.物联网设备上通常运行着嵌入式操作系统和应用软件,并通过特定的网络协议进行互联.物联网系统中,上述 3 个层次都存在各种恶意攻击模式[27],如:针对无线传感网,存在虚假路由信息、Sinkhole 攻击、Wormholes 攻击等[28];针对传输层安全,存在拒绝服务攻击、中间人攻击、异步攻击等[29];而在物联网应用层,通常会收集和处理用户大量隐私数据,如个人信息、通讯薄、出行线路、消费习惯等,因此也存在针对网络服务设备的攻击.工业控制终端也成为恶意软件攻击的平台,Chen 等人[30]分析了 Stuxnet 等工控恶意软件,指出工控恶意软件和通用平台的恶意软件相比所具有的独特特征,例如:工控恶意软件很多是条件触发的,感染途径独特,往往利用了 PC 和特定嵌入式设备的连接通道,采用混合编程语言,涉及传统平台的编程和 PLC 编程等.

1.1.4 新型网络平台上的恶意软件

传统网络以 C/S(client/server)结构提供服务,然而,随着 P2P、云计算和网络虚拟化等新网络技术的出现和普及,针对网络的恶意软件有了新的演化和发展,出现了 P2P 蠕虫、云计算攻击代码,甚至针对 SDN/NFV 网络的攻击行为.一些恶意软件和恶意攻击与 P2P、云计算或 SDN/NFV 独特的软硬件架构密切相关.在 P2P 网络中,出现 Sybil 攻击和路由欺骗攻击等[31].这类攻击消耗了 P2P 网络的资源,削弱了网络的冗余性.在云计算平台中,

出现了针对虚拟化架构的恶意攻击软件、交叉虚拟机侧信道攻击(cross VM side channels attack)和定向共享内存攻击(targeted shared memory)[32−35]等利用虚拟化体系结构的特点攻击云计算租户,甚至底层的虚拟监控机.Wu 等人[33]研究了云计算环境下 x86 体系结构计算机面临的侧信道攻击,发现传统的侧信道不可行的主要原因包括地址访问的不确定性(addressing uncertainty)、调度机制的不确定性(scheduling uncertainty)以及 Cache 缓存的物理局限(cache physical limitations).在此基础上,该团队设计了一种基于时间的数据传播模式,并利用内存总线作为高带宽的侧信道传输中介,从而提出了一种针对云计算系统的、鲁棒的高带宽侧信道攻击方法.该方法在 Amazon EC2 cloud 的实际环境中能获得 100bps 传输速率.Yarom 等人[34]针对云计算和多核架构提出了一种利用 LLC(last-level cache)进行侧信道攻击的模式.传统的利用 Cache 进行侧信道攻击主要利用 L1 cache,但在很多虚拟监控机(VMM)的实现中,一个 VM 在一个独立核上运行,L1 cache 是隔离的,限制了攻击的成功率.该团队发现:尽管很多攻击模式在 LLC 上不可行,但是 FLUSH+RELOAD 攻击是可行的.为此,他们设计了针对 LLC 的攻击算法,获得了 1.2 Mb/s 时间侧信道传输率,并成功获得了 GnuPG 的秘钥.在 SDN/NFV 方面,也出现了针对控制器、开放式编程接口等核心组件或薄弱环节的攻击方式[36,37].德州 A&M 大学的团队[36]研究了 SDN 网络中的 Openflow 控制器,提出一种新的 SDN 攻击模式.该攻击模式类似于传统的 ARP 欺骗攻击(ARP spoofing attack),其思想是:通过伪造路由信息,引导 Openflow 控制器服务或上层应用错误的访问,从而导致劫持、拒绝服务或者中间人攻击等恶意行为.在该项工作中,作者分析了 Openflow 控制器中拓扑管理服务(topology management services)的漏洞,提出了主机定位劫持攻击(host location hijacking attack)、链接伪造攻击(link fabrication attack),并在 Floodlight、OpenDaylight、Beacon 以及 POX 等主流的 SDN 控制中实现了实际的攻击场景.云环境下,针对网络应用层的恶意攻击既继承了传统网络的攻击模式,又融合了新的网络结构特点,出现了一些演化形式.以云计算模式为例,弹性的计算资源与灵活的访问方式为僵尸网络提供了良好的运行环境,攻击者既可以使用云服务器作为主控机,也可以使用窃取到的高性能虚拟机作为僵尸机,有效地降低了被检测或追溯的可能性.针对云计算的按需计费模式,出现了欺骗性资源耗尽攻击(fraudulent resource consumption attack)[38].此外,与传统的 Web 应用环境不同,云计算环境的虚拟化特征加剧了恶意代码注入攻击的安全威胁[39].中国科学院信息工程研究所和西安交通大学的团队[39]协同设计了一种利用云计算系统中推送服务的僵尸网络,该网络利用谷歌的云到设备消息服务(cloud to device messaging service,简称 C2DM)控制多个“肉鸡”,具有隐蔽、资源消耗低、稳定等特点.并通过 C2DM 实现了 SMS-Spam-and-Click 等多种攻击模式.云端的服务迁移、虚拟机共存等操作使得恶意代码的检测工作异常困难.

1.1.5 APT 等高级攻击模式

随着软件承载越来越多的经济价值和国家利益,恶意软件已被广泛用作牟利工具,形成规模化、专业化的黑色产业链.一方面,各类黑客组织或一部分民间的技术爱好者开发恶意软件,通过窃取网络虚拟资产、个人隐私信息等方式牟利;另一方面,一些国家、组织机构也在发展这方面的能力,以对其他国家和组织实施破坏或从中获取情报.恶意软件成为实施主动攻击、情报搜集的重要手段,恶意软件的发展,上升到国家间“网络战”对抗层面.恶意软件开发者综合利用隐私信息、漏洞挖掘等知识,发展出 APT 高级攻击形态,成为当前网络安全的重要威胁[40].从 2009 年谷歌公司报告遭受代号为“极光”(aurora)的攻击开始,“震网(stuxnet)”“火焰(flame)”“夜龙(night dragon)”“雷金(regan)”“方程式(equation)”等 APT 攻击不断被发现[41,42].RAS 实验室的 Juels 等人[42]认为,现有工作常常将 APT 攻击分为社会工程、渗透、平行移动、窃取数据 4 个步骤,但是这样的分析方法往往限制了对 APT 的认识,要应对新型 APT 攻击,需要排除这种固化认识,用更广的视野来进行研究分析.

1.2 恶意软件对抗

为了达到获取信息或破坏信息的目的,恶意软件常常利用对抗技术来逃避反病毒软件和分析人员的检测和分析,以延长存活时间,获取更大的利益.恶意软件的对抗技术从实现原理上主要分为两类:反静态分析和反动态分析,具体包括加壳、代码混淆、信息隐藏、代码虚拟化、调试工具检测、沙箱和监视工具检测等.

1.2.1 传统软件平台上恶意软件对抗

传统平台上的恶意软件大量使用加壳、混淆等对抗手段.目前,恶意软件采用的反分析技术进一步深化,针

对性更强.在反静态分析技术方面,加壳、花指令(junk instruction)变换等技术出现新的变化.以加壳技术为例,一方面,目前加壳软件种类繁多,常用的加壳软件就有 UPX、PECompact、ASPack、Petite 和 WinUpack[5]等;另一方面,加壳程序会有针对性地误导分析软件.Linn 等人[43]的工作发现:主流的逆向或调试工具,如 IDA Pro、Ollydbg、Windbg 等,在反汇编阶段大都采用线性扫描(linear sweep)算法和递归遍历(recursive traversal)算法.他们针对两种算法提出了对抗技术.针对线性扫描算法设计了分支翻转(branch flipping)花指令,针对递归遍历算法设计分支函数(branch function)、调用转换(call conversion)以及跳表伪造(jump table spoofing)等多种控制流混淆技术.实验结果表明:这些技术在对抗逆向工具方面效果明显,加入这些混淆技术后,最坏情况下导致 65%的指令和 85%的函数被错误反汇编.Moser 等人[44]分析了传统恶意代码对抗静态分析能力的演变过程,他们认为:早期的恶意软件查杀工具依靠特征匹配(pattern matching)来分析恶意软件,已很难应对新型恶意软件.随着模型检测等基于语义特征(semantic signature)的静态分析方法的出现,恶意软件的分析能力得到提升.但是恶意软件静态分析技术本身存在很多局限性,例如:通过设计透明常量(opaque constant)等变换技术,基于语义特征的静态分析方法不能正确跟踪变量值的计算过程.因此,仅仅依靠静态分析不足以应对恶意软件对抗技术,需要动、静态结合的解决方案.动态反分析技术也日益丰富,触发条件更加复杂,隐蔽性和不确定性进一步增强,可通过人机操作检测、系统配置检测等手段来触发恶意行为.例如:当检测到左键点击后,恶意软件 UpClicker 才会连接远程的 C&C 服务器;当检测到 3 次鼠标点击后,恶意软件 BaneChant 才会激活功能.恶意软件 Trojan Nap 通过睡眠(19 分钟)来躲避检测,代码里还使用了未公开的 API 以对其他可疑操作提供掩护;ActionScript 恶意代码中,下载恶意 Flash 时需要输入系统安装的 Flash player 版本号,如果版本号不满足要求,则无法下载恶意 Flash[45].Kolbitsch 等人[46]介绍了一种恶意代码对抗动态沙箱分析的手段,无论该样本在什么环境下,执行恶意动作前都要先执行 Stalling Code.Stalling Code 会调一些无关紧要的系统 API,由于沙箱一般会对系统 API 进行监视,因此,如果在沙箱环境中运行就会产生较大的开销,从而规避分析.如果是真实环境,则其开销变得微不足道,并不影响程序正常执行.该恶意代码在 Anubis 环境中执行需要数天,但在真实环境中仅需数秒.加密技术(encrypted malware)也是恶意软件主要的对抗机制之一,恶意软件主要通过加密关键代码(payload 等)或数据(流量)的方式进行保护,例如,将加密数据混入正常的 HTTP 或 DNS 流量中加以隐藏.近年来,恶意勒索软件也利用加密函数劫持用户数据,勒索钱财[47].

1.2.2 移动智能平台上恶意软件对抗

在移动智能平台上,由于 Android 应用市场通常采用静态分析的方法对上载到市场的应用程序进行安全分析,Android 恶意程序的开发者通常采用反分析技术来对抗检测.另一方面,由于 Dex 字节码的逆向和反分析相对而言比 C/C++二进制代码容易,应用开发者出于对知识产权的保护,也通常采用对抗技术保护自身利益[48].因此,针对 Android 应用的反分析技术近年来也迅速发展.Android 应用发布者通常利用混淆、代码加密、密钥置换、动态加载、代码反射和本地代码执行等机制对软件产品进行加固,出现了 Dash-O、Proguard 和 Dexguard 等自动混淆和加固的工具[49,50].学术界多从对抗技术和分析技术能力对比的角度出发,对移动智能系统中对抗技术进行研究.中国科学院信息工程研究所的团队[26]在对 45 个 Android 应用市场中 1 427 395 个应用进行分析的过程中发现:当前, Android 应用主要采用名字混淆技术进行保护,名字混淆主要是将代码中包、类或者方法的名字变换为无意义的字符串.例如,Android SDK 集成的 Proguard 就提供名字混淆.但是一些商业工具,例如 Dexguard、Dash-O 以及 Dexprotector 等除了提供名字混淆外,还提供控制流混淆、代码加密等手段.Li 等人[51]对 500 多个典型的 Android 应用进行分析发现,很多 Android 应用开发中用到了反射调用(reflective calls).反射调用几乎可以被看成恶意应用和非恶意应用的主要区别之一,很多恶意应用利用反射隐藏对敏感 API 的调用以及对敏感数据的存取.他们在此基础上开发了反射分析工具 DroidRA,并利用这一工具挖掘多个恶意 Android 应用中隐藏的反射调用,例如:com.boyaa.bildf 利用反射调用旁过 GET_TASKS 权限检测,调用敏感 API Activity Manager.getRunningTasks(int).Rastogi 等人[52]设计了一个针对 Android 应用的混淆技术框架 DroidChameleon,并将混淆分为简单混淆、通过静态分析可能发现的混淆以及只有通过动态分析才能发现的混淆.其中,简单混淆不需要进行代码级的改变或 Android Manifest 中存储的元数据的改变,例如重打包、反汇编与再汇编;通过静

态分析可能发现的混淆包括重命名包、重命名标识符、数据加密、代码重排序、无用码插入、组合变形技术等;只有通过动态分析才能发现的混淆技术包括代码反射与字节码加密等.Rastogi 等人在实验中发现:现有的很多商业恶意软件分析工具仅使用包名、类名和方法名作为检测特征,所以,简单的混淆技术就能规避这些反恶意软件检测,更不用提复杂的混淆.该工作提出,应采用语义敏感的方法来提升检测工具的能力.

1.2.3 恶意软件利用新网络技术对抗

前面提到的结构化 P2P 僵尸网络具有良好的可靠性和隐蔽性,成为当前 DDoS 的主流模式.同时,恶意软件也利用 P2P 网络的内部共享机制进行扩散.

1.3 恶意软件分析、清除和防御

随着恶意软件的大量爆发、恶意对抗技术的发展,如何发现潜在的恶意软件并对恶意软件进行清除和防御,成为软件安全人员研究的重点内容.

1.3.1 恶意软件分析

恶意软件分析是检测、清除和防御恶意软件的基础和前提.现有的恶意软件分析方法主要依赖于传统的软件分析方法,利用反编译和调试等工具进行分析;分析方式主要分为人工分析、静态反编译分析和动态跟踪调试这 3 种.近年来,恶意软件的检测和分析方法出现了一些新的趋势.

1) 融合静态分析和动态分析技术,研究更加完善的沙箱环境,真实、透明地捕获样本的动态行为[53].

恶意软件的静态和动态分析两种方法各有优缺点,近年来出现了 CWSandbox[54]、TTAnalyze[55]和 Ether[56]等动、静态融合的沙箱检测机制,同时也出现了集成多个杀毒软件和扫描引擎的网站,以提供在线可疑软件扫描服务,如 VirusTotal 和 VirSCAN 等.CWSandbox[54]是由德国曼海姆大学开发的一款自动行为分析工具,通过在真实环境或者虚拟 Windows 环境中执行要分析的样本来对其进行分析.该工具从 API 级别进行监控,主要关注文件、注册表、网络、系统交互这 4 种类型的行为.其主要监控原理是通过 API Hook,在加载样本进内存时对样本进行重写来实现的.Ether[56]是佐治亚理工大学开发的一种基于硬件虚拟化的“透明”恶意代码分析框架,其动机是弥补原有的分析系统易于被检测的缺点.它基于 Xen 中的虚拟监控机实现,因此具有较好的透明性.Ether 支持监视指令、内存写入以及 Windows XP 环境中的系统调用.另外,Ether 还提供将监控局限在特定进程的机制.加密恶意软件的分析主要采用动、静态结合等方法对加密代码进行解密或者绕过加密函数的调用,因此,如何识别加密函数、加密配置等,成为近年来恶意软件分析关注的主要问题之一[57].

2) 研究高效、快速的数据分析方法,以满足实际应用的需求.

恶意代码呈现出爆发式增长的趋势,日均出现上百万的样本量.但是当前恶意代码分析的形式主要依赖人工提取特征码.相关资料显示,平均一名熟练的分析人员一天只能分析 12.8 个样本,供需矛盾严重.基于数据分析的恶意软件分析的工作主要集中在两个方面:一方面是考虑如何抽取或演化出针对恶意软件家族或者特定平台的精确特征,主要是动、静态特征信息.当前使用的静态信息特征通常为字节序列(定长或者变长);使用的动态行为特征包括系统调用序列、系统状态变化和资源操作序列等[58−61].例如:Park 等人[61]提出的从同一家族样本的行为信息中提取共有行为来作为该家族样本行为特征的方法,通过分析家族中的每一个样本,得到对应的包含内核对象及其属性的图,再组合这些图,可以得到一个涵盖了该家族样本所有行为的超图.超图中含有一个唯一的所有样本共有的子图,称为热点路径.结合超图和热点路径,该方法可以较为准确地检测出该家族样本的变种.陈志锋等人[62]针对内核恶意软件的检测,通过分析内核运行过程中内核数据对象的访问过程,构建了内核数据对象访问模型,基于该模型讨论了构建数据特征的过程,采用动态监控和静态分析相结合的方法提出内核恶意软件检测算法.另一方面是研究如何高效、精确地对样本进行聚类与分类,进而加以检测[63−66].这一类型的工作比较多,使用的机器学习算法有贝叶斯、SVM、决策树和深度学习等[67−70].例如:Rieck 等人早期采用动态分析工具(CWSandbox)获取样本的行为信息,这些行为信息包括样本在执行过程中的文件行为、进程行为、注册表行为、网络行为等.采用了 SVM 方法进行分类,作者收集了 10 072 个样本,并分为训练集、验证集和测试集.实验结果表明,有 88%的样本被正确地分配到了对应的家族里.2011 年,Rieck 等人[68]使用无监督学习算法对样本的动态行为信息进行了聚类分析,并在提取样本特征时提出了一种类似 CPU 指令集概念的 MIST(malware

instruction set)来描述样本的特征空间,并对机器学习算法的扩展性进行了优化.此外,针对代理恶意软件等重要软件的特征分析也有一些研究工作,崔杰等人[71]通过逆向技术对 UltraSurf、SecurityKISS、GreenVPN 等穿透软件进行分析,得到软件的内部结构、工作流程、加解密技术、服务器节点地址的获取方式、网络通信方式等关键信息,在此基础上,构建软件特征和流量特征,提出自动分析-阻断工具.除了典型的通过逆向分析来获得特征外,也有研究工作采用 SVM 对恶意代理软件 Freegate 网络流量进行自动特征提取.近年来,随着人工智能、深度学习技术的发展,一些监督、半监督的智能技术也应用到恶意软件的分析中,例如智能化的变异恶意软件特征归纳等.Greengard 等人[22]对包括 CMU、佐治亚理工在内的一些学术界和工业界的工作进行了分析,其中,Deep Instinct 发现:很多恶意软件仅对 2%的代码进行变异就能绕过查杀软件,形成 0-day 攻击或 APT 攻击.他们采用 ANN (artificial neural network)等技术提出了自动化的恶意软件特征提取,以替代人工操作.

3) 研究高度定向的分析手段,以提高分析的精确度和效率.

恶意代码动态分析中存在一个比较严重的缺点是:在一次分析中只能覆盖一条执行路径.如果样本在该次执行过程中没有表现出恶意行为,则无法判断该样本是否包含潜在的恶意行为.为了弥补动态分析的缺点,很多学者利用软件分析、测试的方法提高分析的覆盖率或者触发隐藏的行为分支[72−74],提出了 REANIMATOR[75]、HASTEN[46]、TaintDroid[76]等分析方法或工具.例如,Kolbitsch 等人[46]针对 Stalling 代码设计了分析工具 HASTEN,其主要思想是:在动态执行的过程中,通过统计成功/失败的系统调用、相同/不同系统调用等信息来计算分析“进展”;如果在分析的过程中没有达到期望的“进展”,则构造动态控制流图,查找活循环(live loop),并确定 Stalling 循环;最后,设计状态条件跳过 Stalling 循环,触发恶意代码后续行为.TaintDroid 系统[76]将动态污点跟踪技术移植到了 Android 系统上,对敏感数据进行污点标记,当这些数据通过程序变量、文件和进程间通信等途径扩散时进行跟踪审查,实现对多种敏感数据泄露源点的追踪.

4) 基于源代码的恶意软件分析方法研究.

恶意软件往往具有隐蔽性、对抗性,主要以二进制代码的方式进行发布,因此,恶意软件的分析主要针对二进制代码来进行.随着开源软件的兴起,基于源代码的恶意软件分析也受到研究人员的关注.当前,基于源代码的恶意软件分析技术主要包括动静态软件分析和软件挖掘技术.例如,Christodorescu 等人[77]提出了一种通过对比恶意软件与正常(可能具有源代码)软件行为来挖掘恶意行为的方法,如果一种行为在恶意软件中出现但在正常软件中没有出现,则认为该行为具有一定的恶意性.

1.3.2 恶意软件的清除和防御

在恶意软件清除方面,当前主要基于隔离(quarantine)、删除、“检测-阻断”等机制.恶意软件防御没有一种 One-Fit-All 的策略,针对不同的系统和攻击方式,安全人员须采用不同的软件安全机制来进行防御,详见本文第 3 节的论述.例如,张慧琳等人[78]针对网页木马的机理、检测和防御提出系统的分析工作,指出对网页木马首先要分析其类型(网站服务器木马、代理端木马、基于客户端网页木马),再有针对性地设计清除和防范策略.各种策略所采用的机理、防御的代价也各不相同.其中提到一种利用代理进行浏览器不安全功能隔离的网页木马防范框架,尽管隔离效果良好,但却带来了较大的时间开销,不适于实际应用.另外,恶意软件的攻击往往由多个步骤构成,形成攻击路径,恶意软件的防御应该针对不同的需求、环境选择合适的策略.围绕恶意软件的攻防博弈在持续进行:攻击者不断采用一些新的手段来提高恶意软件攻击成功率以及增强其隐蔽性;防御方的安全研究人员随着对各种恶意软件研究的深入,也不断提出相应的检测和防范方法.

2 软件漏洞

软件漏洞是信息系统安全的主要威胁.特别是近年来,一些基础软件和系统中的漏洞出现得越来越频繁,给软件生态造成了严重危害.软件漏洞的存在源于软件的高复杂度.虽然各大软件厂商在不断改进和完善软件开发质量管理,开发测试人员也付出了大量工作,软件漏洞问题仍无法彻底消除.当前的软件系统,无论是代码规模、功能组成,还是涉及的技术均越来越复杂,其带来的直接结果就是从软件的需求分析、概要设计、详细设计到具体的编码实现,均无法做到全面的安全性论证,不可避免地会在结构、功能和代码等不同层面存在可能

被恶意攻击者利用的漏洞.

漏洞形成的诱因多种多样,根据诱因形成的不同,可将常见的软件漏洞分为整数溢出漏洞、缓冲区溢出漏洞、逻辑错误漏洞等.整数溢出漏洞通常是由于无符号类型数与有符号类型数混用导致,也可能是程序设计开发人员未考虑到数据运算的边界问题所致,属于原始软件本身设计的安全结构问题.例如求解 fibonacci 数列第 100 项,已经无法用普通的整型数表示,如果开发者未意识到这一点就很容易犯这类错误.缓冲区溢出漏洞的形成源自于分配空间过小与分配使用限制不严格,常出现于 scanf、strcpy、sprintf 等不安全的字符串复制函数,属于滥用不安全代码模块,也有大量的缓冲区溢出是由整数溢出导致,由于整数溢出突破了边界检测,因而导致缓冲区溢出.逻辑错误漏洞涉及面较广,跨站脚本、SQL 注入、多线程条件竞争漏洞等都可以归为由逻辑错误导致,由代码执行过程中逻辑规则出错造成.

随着各种新技术的引入,软件漏洞的形态越来越复杂和多样化,从最初的栈溢出和堆溢出等溢出型漏洞,到跨站脚本、SQL 注入等网页漏洞以及最近 HeartBleed 等敏感数据泄漏漏洞等[79].工业控制系统也成为漏洞的栖息地,这类系统的多样化给漏洞的分析修复增加了难度.与传统的用户系统相比,工业控制系统的运行不可随意中断,漏洞修复周期更长,对热补丁有更迫切的需求.软件漏洞自身复杂性和多样化不仅使得我们迄今仍无法对“软件漏洞”给出准确的定义,更是对传统的依靠人工调试进行软件漏洞发现和利用的方式提出了巨大的挑战.以 PLC 为代表的漏洞态势感知预测在当前大数据背景下不断发展,所谓态势[80]是指由各种网络设备运行状况、网络行为以及用户行为等因素构成的网络当前状态和变化趋势,是一个宏观和整体的概念.基于漏洞库通过远程渗透测试的方式是评估特定漏洞在网络空间环境下存在的普遍程度和危害程度的重要方法,也是漏洞攻击采用的重要手段之一.这方面目前已有大量的相关工作,Metasploit 是典型的代表,集成了大规模 CVE 的利用框架,并及时更新最新漏洞库.本节侧重于从微观原理的角度对未知漏洞的相关研究.按照漏洞产生和发展的过程可分为 3 个阶段,即软件漏洞挖掘、软件漏洞分析和软件漏洞利用生成.下面围绕这 3 个方面分别介绍目前国内外研究的相关进展情况.

2.1 软件漏洞挖掘

软件漏洞挖掘是指针对软件的源代码或可执行代码进行分析,检测其是否存在有可能被利用的缺陷.自 20 世纪 70 年代美国南加州大学发起 PA(protection analysis project)研究计划以来,人们提出了各种各样软件漏洞挖掘的方法.目前,除了具有丰富领域经验知识的专家、黑客仍然依靠手工代码分析以及软件逆向调试的方式挖掘漏洞以外,出现了基于源代码、补丁对比、模糊测试以及代码特征挖掘等技术思路的漏洞挖掘方法,这些方法正在软件安全研究工作中发挥着重要作用.

2.1.1 基于源代码的漏洞挖掘

基于源代码的漏洞挖掘技术是软件漏洞挖掘的重要手段,其前提是拥有软件源代码,适用于开源软件或者软件开发者自身的挖掘测试.基于源代码的漏洞挖掘分析通常与编译器相关技术相结合,通过词法、语法、语义等分析抽象程序表示,进一步结合数据流分析、控制流分析、静态符号执行等技术识别代码缺陷位置,再进一步验证漏洞的存在与否.基于信息流的整数漏洞插装和验证[81]是在源代码条件下,基于 GCC 编译器进行修改实现的系统.此外,还可根据一定的安全规则,例如检测代码中是否使用不安全的 strcpy、sprintf 等函数.基于源代码的漏洞挖掘更适用于开源的 Linux 环境,也包括 Android 平台,不适用于闭源环境下的软件系统,如 Windows 系统及其下的 Office、IE、Adobe 等大型商业软件.

2.1.2 基于补丁对比的漏洞挖掘

基于补丁对比的漏洞挖掘,是指通过比较分析原始程序与漏洞修复后的更新程序的差异,依据已知漏洞对应的软件缺陷位置寻找新漏洞的一种漏洞挖掘方法.补丁发布与部署之间存在较长的时间窗口,因此,通过补丁漏洞挖掘迅速提取漏洞特征,对于软件运行安全检测和防护具有重要意义.目前,软件自身的复杂性导致简单地进行指令对比难以快速逆向恢复漏洞细节并理解漏洞机理.与漏洞不相关的补丁修改造成相当程度的干扰,补丁漏洞挖掘仍然需要软件基础理论的支撑.为此,研究人员提出了多种基于图比较算法发现程序差异的基础分析工具,如 IDACompare、EBDS、BinDiff 等.补丁漏洞挖掘的代表性工作是 Brumley 团队提出的 APEG(automatic

patch-based exploit generation)方案[82],该方案依据原程序与补丁程序的对比分析,找到能触发原程序异常的过滤条件,并依据该条件构造出使原始程序缓冲区溢出的输入样本,从而实现漏洞挖掘.虽然补丁对比具有一定的效果,但并不适用于新软件、新模块以及未知类型漏洞的挖掘.

2.1.3 基于模糊测试的漏洞挖掘

模糊测试是一种通过提供非预期输入,并监视目标软件在处理该输入后是否出现异常的动态测试方法.模糊测试实现了测试用例生成、变异和导入以及测试目标状态监控的自动化,可有效提高软件测试效率.其中,测试用例生成方法和变异策略决定漏洞挖掘效率,是模糊测试研究中的难点.经过多年研究,模糊测试已逐渐分化为盲目随机生成用例(测试用例随机生成,如 zzuf)、基于目标静态结构信息生成用例(测试用例基于文件格式、网络协议格式规范生成,如 Peach、Sulley 等)、基于目标动态反馈生成用例的测试(测试用例基于代码覆盖率反馈进行变异,如 honggfuzz 和 AFL)以及基于符号执行和求解生成用例(通过符号执行和求解高效遍历程序执行路径,如 KLEE[83]、SAGE[84]以及商用版本 SpringField 等)这 4 个主要方向,分别满足不同层次模糊测试的要求,并已在软件漏洞挖掘中发挥重要作用.Peach、Sulley、AFL 是较为成熟的模糊测试工具,在实际环境中具有不错的效果;KLEE、SAGE 这类符号求解工具仅适用于小型程序,或者代码片段的求解,对于复杂的符号表达式求解存在一定困难.模糊测试在软件测试和漏洞挖掘中具有非常重要的地位,但由于软件的高复杂度,其仍然无法遍历所有可能情况,结果的好坏取决于种子文件和变异规则.在模糊测试的框架下如何优化变异规则、如何优化测试路径,是当前普遍的研究思路.

2.1.4 基于代码特征的漏洞挖掘

基于代码特征的漏洞挖掘,是指依据程序中漏洞相关的代码特征提取漏洞模型,并基于该模型对目标软件进行静态代码审计或动态运行检测以发现漏洞的方法.该方法的优点在于具有较强的针对性,并对大规模复杂程序仍然有较好的扩展适应能力.它的缺点是存在不可忽视的误报率,仍然较多地依赖于人工经验的筛选过滤,实用性还有待提高.需要解决的问题包括程序漏洞代码特征和模型的描述以及基于漏洞模型的程序漏洞搜索挖掘等,在这方面的工作中,Engler 等人提出的基于程序代码行为异常进行漏洞挖掘的方法[85]将软件错误看作程序运行过程中的偏离特定规范的偏差行为,结合数据挖掘、统计分析等方法发现偏差.Yamaguchi 等人提出的基于敏感数据检测缺失的漏洞挖掘方法[86]对用户的相关数据进行污点跟踪分析,将对敏感数据内容缺少必要检测的程序行为视为潜在漏洞.Grieco 和 Grinblat 等人提出一种将动、静态特征与机器学习进行结合的漏洞挖掘方法[87],收集了上千个测试程序的内存错误信息,开发了 VDISCOVER 工具,使用机器学习的方法预测测试用例中的漏洞.传统的特征分析已无法应对日益增加的代码复杂度,基于代码特征的漏洞挖掘与数据挖掘、机器学习的结合也是未来发展的一个趋势.

2.2 软件漏洞分析

软件漏洞分析主要是指对软件漏洞形成机理的分析,即:在确定软件漏洞存在的情况下,进一步深入剖析软件漏洞形成的原因及影响要素等内容,最终判定软件漏洞的类型以及定位软件漏洞的发生位置及成因等关键要素.在软件漏洞分析研究的早期,主要是依赖具有丰富漏洞分析经验的专家进行手工分析和定位.随着程序分析技术的不断发展,尤其是污点传播、符号执行等基础方法的发展和完善,研究人员结合相关技术提出了一系列自动或半自动的软件漏洞分析方法.

2.2.1 软件漏洞分析基础方法

漏洞分析所依赖的基础方法主要包括污点传播分析和符号执行等.污点传播分析方法将程序外部输入数据“绑定”污点标签,用来追踪污点数据.相关工作包括最早基于硬件虚拟化技术实现的动态污点分析系统 TEMU[88]、实现比特级细粒度污点传播的 DECAF 系统[89]、具备较高分析效率的 libdft 系统[90]等.污点传播分析方法应用的难点在于制定合适的污点传播策略以应对复杂的程序处理模式带来的污点“过标记”和“漏标记”的问题.为此,需要依据不同的问题采取不同的传播策略.例如,Song(DTA++)[91]和 Clause(Dytan)[92]等人的工作分别对指针污点、控制流依赖带来的隐式污点等传播策略问题对分析效果的影响进行了评估和总结,并提出了各自改进方案;在将污点分析方法应用于软件漏洞研究方面,Cui(RETracer)[93]和 Xing(CREDAL)[94]等人的工作

利用后向污点回溯方法逆向恢复程序脆弱点的位置.符号执行通过收集符号变量的路径表达式并加以求解,以获得程序输入与程序异常之间的关系.符号执行除应用于漏洞挖掘之外,在漏洞脆弱性分析和检测中也有广泛应用,如Stelios等人提出的DIODE方法[95]通过动态计算程序输入与堆操作函数参数的关系,实现了堆溢出漏洞条件的精确构造.

2.2.2 软件漏洞判定

软件漏洞判定是指发现程序异常或故障等缺陷后,依据漏洞成因对软件漏洞类型进行匹配识别.当前,软件漏洞判定的主要思路是通过动态污点传播分析技术,分析外部污点数据与程序中关键函数和敏感数据的关系,再依据污点对关键函数和数据的影响识别漏洞类型.这方面的代表性工作有:Song团队提出的TaintCheck系统实现了缓冲区溢出、格式化字符串、多次释放等软件漏洞类型的检测识别和特征提取[96];Enck等人提出的TaintDroid系统实现了移动应用中个人敏感信息泄露漏洞的检测识别[76].软件漏洞的判定仅适用于已知漏洞类型,且具备固定特征模型的判定,对于新型的程序异常和故障,尤其是不属于漏洞的异常难以有效而精确地进行判别筛除,还需要大量的人工分析.

2.2.3 软件漏洞定位

在确定程序中存在漏洞后,需要进一步定位程序中的脆弱点,明确漏洞机理,以便对软件漏洞进行及时修复.目前,漏洞定位的主要困难在于软件异常或故障发生位置滞后于软件错误发生位置,漏洞形成信息部分可能永久性缺失.目前,主要研究思路是依据软件故障和异常点程序状态进行逆向推导回溯分析.研究人员已提出一系列程序漏洞定位方法:Wu等人在静态函数调用图的基础上提出了一种程序崩溃运行记录恢复方法,能够快速定位程序中的漏洞位置[97];微软公司的Cui等人研制的RETracer系统[93]通过分析崩溃点被破坏的指针进行解引用的过程,准确定位程序错误成因;宾夕法尼亚大学的Xu等人提出的CREDAL方法借助控制流图以及数据依赖错配的识别来定位软件脆弱性[94],该方法建立在具备源代码的条件下,依赖栈空间的完整性,不适用于栈空间被破坏的情况.虽然目前有不少关于软件漏洞定位的相关工作,但其仍有各自的局限性,在高度复杂的软件环境下,通常需要融合多种方法和技术进行漏洞定位分析.

2.2.4 软件漏洞修复

软件漏洞修复是软件漏洞分析的目的之一,通过分析漏洞类型、漏洞位置、形成原因等因素,形成漏洞的修复方案.目前普遍采用的漏洞修复模式是厂商或者第三方安全机构发布软件补丁,用户通过自动更新、手动更新的方式将补丁安装到原始系统中.这种模式依赖于用户的习惯和安全意识,存在周期长、时效性差等缺陷.为弥补这些缺陷,Brumley等人提出了一种根据已知版本补丁自动生成另一软件版本补丁的方案[98];Sha等人基于漏洞描述和模糊测试技术提出了一种漏洞补丁自动生成方法PVDF[99];Zhang和Yin提出了在Android平台为阻止组件劫持攻击的特定漏洞补丁自动生成方法[100],虽然该方法只是专门针对特定的一类问题,但也可预示出软件漏洞自动修复是未来的研究趋势.

2.3 软件漏洞利用

软件漏洞利用,指的是利用软件的内部缺陷,以实现通过该软件的正常运行无法达到的目的.软件漏洞只有被攻击者利用才会对安全造成直接的危害.根据利用目标的类型,软件漏洞利用可以分为执行代码、绕过认证、权限提升和信息窃取等多种类型.软件漏洞利用的构造方式也存在较大差异.传统的软件漏洞利用主要以手工方式构造,研究人员不仅需要具备较为全面的系统底层知识,同时还需要对漏洞机理进行深入、细致的分析.在软件功能越来越复杂、漏洞越来越多样化的发展趋势下,人工分析已难以应对上述挑战.随着程序分析技术的不断发展,研究人员开始尝试利用污点分析、符号执行技术来进行高效的软件漏洞利用自动构造.

2.3.1 面向控制流的利用方法

面向控制流的自动利用方法的核心思想是:借助程序验证、动态污点传播、混合符号执行等方法和技术分析程序对输入数据的处理错误,构造出一个通过输入数据改变程序控制流的利用路径,从而实现任意代码的执行.该方法要解决的主要问题是面向控制流的漏洞可利用性判断和漏洞利用生成等.目前,这方面的相关工作已经可以实现在一定约束条件下的漏洞自动利用生成.2016年,美国漏洞自动攻防竞赛(CGC)冠军团队提出的

Mayhem 方案[101]综合利用在线式符号执行和离线式符号执行技术,并基于索引的内存模型构建了一套较为实用的针对二进制程序的漏洞挖掘与利用自动生成系统.但针对 CGC 的解决方案是在简化的系统环境下的方案,该环境缺少现实环境下的安全防护机制,未考虑随机化因素带来的影响.Wang 等人提出的多样性利用样本自动生成方法 PolyAEG[102],其核心思想是通过动态污点分析,找出程序所有控制流劫持点,再通过构建不同控制流转移模式来完成漏洞利用样本的多样性构造,实现了一套完整的针对控制流劫持类漏洞的漏洞利用自动生成系统.

2.3.2 面向数据流的利用方法

面向数据流的利用方法是指在不直接影响和操控程序控制流的前提下,通过分析程序对输入数据的处理,构造出一个或多个利用输入数据改变程序数据流的利用路径,进而完成权限提升、认证机制绕过等功能.在数据执行保护、地址随机化以及控制流完整性防护手段大范围部署的情况下,面向数据流的利用方法具有更强的适用性和灵活性.这方面的代表性工作是 Liang 团队的 FlowStitch[103],该方法利用已知内存错误直接或间接地篡改程序原有数据流中的关键变量,通过比对错误执行记录和正常执行记录,筛选并确定内存错误影响范围中的敏感数据来完成信息泄露、权限提升等利用的构造.Liang 团队继而提出的 DOP[104]即基于数据流的攻击代码编程模型,同时给出针对实际应用程序的基于数据流的攻击代码块和指令调度分配代码块的搜索、提取和编程方法.初步显示:DOP 是图灵完备的,能够实现任意代码的执行,并能绕过 DEP 和 ASLR 等系统防护措施.

3 软件安全机制

为了主动防御恶意软件的攻击、预防软件漏洞被利用,研究人员通过设计多种安全机制来提高整个软件体系应对安全攻击的能力.主要技术思路包括 3 个方面:通过设计软件行为管控机制,规范不受信软件的行为来规避恶意软件的攻击;通过提高软件的自身安全性,来提升软件应对攻击的能力;设计终端用户可感可控可知的安全机制,来提高用户对软件的安全管理能力.下面分别介绍这 3 方面工作的研究现状.

3.1 软件行为管控机制

3.1.1 基于系统的软件行为管控机制

现代移动操作系统普遍使用沙箱对程序进行隔离,并提供基于权限的安全管控机制,以管控应用对系统和用户资源的访问和使用.一些研究工作基于污点跟踪技术对软件访问系统资源的行为进行跟踪和管控,如 TaintDroid 系统[76]和 AppFence 系统[105].也有一些研究工作通过提供伪造敏感数据来控制不受信的软件,使之无法访问真实的敏感数据,如 TISSA 系统[106]、LP-Guardian 系统[107]和 MockDroid 系统[108].还有一些研究工作采用隔离的方式管理不同受信级别的软件,如:TrustDroid 系统将软件划分到个人域或公司域并且隔离域之间的数据;AdDroid[109]、AdSplit 系统[110]以及 AFrame 系统[111]将广告插件置于单独的系统服务中运行,以隔离广告插件对宿主引用的影响.

3.1.2 基于虚拟化的软件行为管控机制

运用虚拟化技术将软件隔离在沙箱之中,是一种常见的安全保护方式.AirBag 系统[112]利用虚拟化技术将恶意应用的行为限制在沙箱之内.SeCage 系统[113]利用 Intel 的 EPT(extended page table)机制实现了不同区域的内存隔离,并且利用硬件虚拟化技术为不同的程序区域提供不同的内存视图.Kurmus 等人提出的 Split Kernel 技术[114],通过在内核中同时保存加固与未加固的函数,实现性能与安全性的同步提升.为了保证软件的可靠性,应该尽量缩小内核的规模.Zhou 等人提出一种称为 Wimpy Kernel 的内核架构[115],通过将 Wimpy Kernel 置于操作系统内核之下的 Hypervisor 层,并将 I/O 操作委托给不受信任的操作系统,将内核的规模减到了最小.为了隔离内核中不受信的驱动程序,VirtuOS 系统[116]基于 Xen Hypervisor 的 Service Domain 功能将内核划分为不同区域.

3.1.3 基于硬件的软件行为管控机制

一些硬件级的安全特性,如 ARM TrustZone,可以被用作可信计算基(trust computing base,简称 TCB)来提供基于硬件的安全管控机制.Azab 等人[117]基于 ARM TrustZone 的安全模式(secure world)设计可信语言运行时

(trusted language runtime,简称 TLR)框架来保护系统内核的机制.Li 等人[118]提出了一种基于 ARM TrustZone 技术的可信移动广告框架,提供可验证的广告显示与可验证的广告点击.软件错误隔离(software fault isolation,简称 SFI)机制通常被用于隔离应用中不受信任的代码,以实现对宿主应用的保护.Zhou 等人[119]提出了一种基于 ARM 内存域(memory domain)特性的软件错误隔离机制,可将不受信代码的访存操作严格限制在沙箱之内.当前的加密系统在使用密钥时均需将密钥加载到内存中进行处理,这为内存泄漏攻击提供了有利条件.中国科学院信息工程研究所的研究人员[120]基于 Intel TSX(transactional synchronization extensions)扩展设计了一种阻止内存泄漏攻击的 Mimosa 系统,该系统基于硬件事务内存,可保证恶意应用读取密钥时,事务会自动终止并回滚,并且事务操作的所有敏感信息均被存放于 CPU 缓存之中,从而确保该系统不会受到与内存芯片相关的硬件攻击.

3.1.4 基于软件重写的行为管控机制

基于系统的工作往往需要修改系统代码,比较难以部署到真实系统中,研究人员也提出一些基于软件重写的行为管控方案.Aurasium 系统[121]通过自动化重打包 APP 并加入用户层面的沙箱以及策略实施代码,来监控程序对 API 的使用行为.Uranine 系统[122]也基于 APP 重写实现了 Android 平台上隐私泄露的实时检测系统.为了避免修改应用软件,Boxify 系统[123]基于动态加载的方式将目标程序加载到受控的进程中运行,并且采用函数钩子(HOOK)拦截目标程序对关键系统 API 的调用,从而管控程序的行为.

3.1.5 基于规则的软件行为管控机制

为了提供灵活的管控软件行为的能力,学术界研究规则驱动的软件行为管控机制.SEAndroid 系统[124]将强制访问控制(MAC)扩展至 Android 系统,以更好地防御各种 root 行为以及应用漏洞.Bugiel 等人设计的 FlaskDroid 系统[125]将强制访问控制扩展到 Android 系统的中间层.为了提供一种通用的行为管控机制,Heuser 等人提出的 ASM 框架[126]通过提供一个可编程的接口去扩展软件行为管控机制.与 ASM 类似,Backes 等人提出了一个可扩展的 Android 安全框架 ASF[127],支持安全扩展人员以模块方式去开发和配置 ASF 的安全模型.权限管理是 Android 系统安全机制的重要组成部分,然而现有工作无法通用地管控权限.为此,复旦大学的研究团队提出了上下文敏感的权限管理机制[128],并且提出一种表述权限管理的规则语言,以支持对权限使用行为进行灵活的控制.基于规则的软件行为管控的核心在于规则,但由于实际情况的复杂性,规则的制定很容易出错.为了克服这一问题,Wang 等人提出了 EASEAndroid 系统[129],通过对日志进行自动化审计,结合半监督自动学习的方法,为细化规则提供建议.

3.2 软件自身安全机制

3.2.1 二进制软件的安全机制

针对二进制软件漏洞的防护机制主要包括漏洞补丁修复、安全机制缓解以及基于漏洞攻击检测的防御.软件漏洞补丁修复是在存在缺陷的软件系统上发布修复问题缺陷的小程序包,或者替换存在问题的程序包.补丁方案能够有效防御已知漏洞,但却无法防御未知漏洞,同时,新的补丁也有可能引入新的漏洞.另外,补丁更新不及时也会带来安全隐患.大部分补丁更新需要重启软件,系统补丁甚至需要重启操作系统,而骨干网络等核心控制系统不可轻易重启,难免存在更新不及时的问题.安全机制缓解技术可以在一定程度上弥补这些缺陷,主要从保护系统的完整性和机密性来保护系统的安全性.研究表明:漏洞利用往往需要通过程序缺陷来破坏内存数据、劫持控制流、跳转到攻击代码并执行这几个步骤.漏洞缓解技术 Stack Guard[130]是通过 Canary 破坏检测方法,阻止栈溢出来保护栈数据的完整性;DEP(data execution prevention)[131]方法通过限定数据段不可执行来阻止攻击代码的运行;ASLR(address space layout randomization)[132]方法通过地址随机化技术保护关键数据位置不被轻易预测,增加控制流劫持的跳转位置预测难度.然而,这些安全机制在实现过程中也存在一定缺陷,攻击者能够利用 ROP(retrun-oriented programmming)[133]绕过 DEP 保护,通过使用未开启 ASLR 的模块作为跳板进行攻击.CFI(control-flow integrity)[134]技术通过分析程序的控制流图来获取间接转移指令目标的白名单,并在运行过程中核对间接转移指令的目标是否在白名单中.控制流劫持攻击往往会违背原有的控制流图,CFI 使得这种攻击行为难以实现,从而保障软件系统的安全.但是这种方法的开销过大,难以得到实际部署;粗粒度 CFI 能够满

足性能上的需求,其完整度却不够,存在被绕过的可能.另外,研究人员已经开始尝试将 CFI 集成到硬件中以便加速.基于漏洞攻击检测的防御是根据漏洞攻击的特点进行防御[135],这些特点包括漏洞签名特征、攻击代码特征、攻击行为特征.其中,漏洞签名特征只能针对已知漏洞,攻击代码特征容易因代码变型和混淆而失效,攻击行为特征通过检测敏感行为特征提取攻击模式,是较为准确、有效的防御方案,但需要动态触发攻击行为,常因漏洞环境不一致、恶意行为长期潜伏等因素未能及时发现.

3.2.2 Web 软件的安全机制

不同于传统的操作系统,Web 应用没有内置的访问控制模块,访问控制由应用开发者自己实现.Web 应用中访问控制机制的漏洞,使得攻击者能够越权访问其无权访问的资源.Monshizadeh 等人提出的 MACE 工具[136]基于多条路径授权不一致的行为特征,自动地检测 Web 应用中的越权漏洞.Pellegrino 等人[137]提出了一种新型的黑盒测试的方法,通过从用户和 Web 应用交互的网络记录中识别 Web 软件的行为模式,再加上常见的攻击场景,生成测试样例.这一方法成功地在真实的商用 Web 应用中发现多个未知漏洞.除了针对 Web 服务器端防护外,Web 客户端的防护也是一个重要的研究领域.Weissbacher 等人[138]指出:浏览器中部分函数(如 postMessage)并不会受到 SOP(同源策略,是当下浏览器中一个重要的安全策略)的限制,导致 Web 程序可能会因这些渠道被攻击.基于 Web 程序自动重写提出的 ZigZag 系统[138]不需要对浏览器和应用程序做修改,就能防止攻击者通过上述通道进行跨域攻击.Doupé等人认为,Web 软件中的很多问题是由于对代码和数据的不加区分引起的.基于此观点,他们提出并实现了 deDacota 系统[139],通过对现有的 Web 软件进行自动化的重写,实现代码和数据的分离.

3.2.3 移动软件的安全机制

得益于移动互联的发展,移动平台在过去 10 年中得到了飞速的发展.与此同时,移动平台在编程模型、系统结构方面的特点,使得移动软件的软件安全机制面临独特的挑战.

首先,移动软件基于组件的开发模式在极大地提高开发效率的同时,引入了组件通信这一新型攻击面.Dietz 提出的 QUIRE 系统[140]用于在程序进行 IPC(inter-process call)调用时维护调用路径上所有应用程序的元信息,从而支持有效防御攻击者的跨组件攻击.针对应用软件中存在的组件泄露问题,也有研究人员采用动、静态分析进行检测,代表性工作有工具 DroidChecker[141]、CHEX[142]、IntentFuzzer[143].上述工作着眼于漏洞的防御和检测,AppSealer 系统[144]则基于静态重写的方式引入漏洞利用检测逻辑,从漏洞自动化修复的角度尝试解决组件漏洞问题;

其次,移动平台包含大量的敏感数据,隐私泄露是移动平台一种常见的攻击方式.静态分析是检测信息泄露的一个非常重要的手段,研究者致力于提高它的精确度和效率.工具 FlowDroid[145]针对 Android 软件设计了一种上下文敏感、流敏感、字段敏感、对象敏感的污点分析技术,并且通过对 Android 中组件的生命周期进行建模,可以更加精确和全面地分析软件中存在的隐私泄露问题.针对组件之间的数据流依赖,工具 AmAndroid[146]结合前人工作,增加了对组件调用过程中的静态分析支持,能够对应用进行跨组件的控制流和数据流分析.针对移动应用开发过程中普遍使用的异步调用特性,EdgeMiner[147]和 StubDroid[148]通过对 Android Framework 进行静态分析,对 Android 的平台回调、跨组件调用等行为进行精确的描述,有效地提高了静态分析的覆盖面和精度.由于静态分析有效率较低、误报率较高的特点,研究人员也尝试使用动态分析的方法研究隐私泄露问题.复旦大学研究团队基于污点分析技术提出 VetDroid 系统[149],通过找出应用中与权限相关的行为,来对隐私泄露行为进行更深入的分析.移动软件普遍存在通过用户输入的方式搜集用户隐私信息的行为,区别于传统的通过规整 API 获得的隐私数据,用户输入的隐私数据不存在固定的数据格式和统一的获取通道,为保护这类隐私带来了极大的困难.针对这一问题,UIPicker[150]和 SUPOR[151]技术基于自然语言处理的方法自动识别包含隐私信息的 UI 组件;

最后,由于移动设备中通常会安装多个应用软件,这些应用软件共享 I/O 设备、CPU、内存和网络接口等.这些共享通道为攻击者进行侧信道攻击制造了条件.研究人员发现:通过侧信道攻击,攻击者可以获取用户的软键盘输入、当前访问的网页、用户的身份、感兴趣的股票、疾病、位置和行进路线等信息.AppGuardian 系统[152]基于侧信道攻击需要经常定期地在后台搜集数据这一特征,提出一种轻量级的防范技术.

3.2.4 云端软件的安全机制

在云计算中,不同的虚拟机可能运行在同一个物理机器上,多个虚拟机共享同一个物理CPU和内存.这为攻击者进行侧信道攻击带来了便利.攻击者基于CPU的共享L1缓存、Last-Level-Cache、内存的Deduplication机制和内存的Row-Hammer效应,使用侧信道攻击并成功读取其他虚拟机的敏感数据.针对此类攻击,学术界也研究了多种防御措施.Düppel系统[153]通过在缓存中自动产生额外的噪音来对可能的攻击产生干扰,从而以较低的代价有效地防御侧信道攻击.注意到侧信道攻击往往需要在受攻击者填充缓存之后快速抢占CPU,Varadarajan等人提出了MRT(minimum run time)技术[154],通过改进虚拟机的调度算法,以确保每个虚拟CPU在一定时间内不会被其他虚拟CPU抢占.由于侧信道攻击种类繁多、原理各异,Nomad系统[155]提出了基于虚拟机实时迁移的防御方法,通过对侧信道攻击进行建模分析,尝试从根本上消除此类攻击.除了增强云平台的安全性外,研究者们还尝试改进云平台上运行的程序,以增强其抗攻击能力.在各类侧信道攻击中,加密算法中使用的密钥是最常见的被攻击目标.HERMES系统[156]利用分布式RSA(distributed RSA)和门限RSA(threshold RSA)算法将原本存储在一个虚拟机当中的密钥分散到不同的虚拟机中,并采用门限加密的方法来保护云上的虚拟机在加密过程中使用的密钥.

3.2.5 网络节点和网络协议的安全机制

安全的网络连接对于软件生态的安全性至关重要,网络连接的安全性取决于网络节点上运行软件的安全性以及网络协议实现的安全性.为了提供更快捷的网络服务,CDN(content delivery network)技术被广泛应用.Liang等人[157]针对CDN环境下HTTPS客户端收到的证书与实际的服务器证书不匹配的问题,提出了一种基于证书认证委托的HTTPS部署方式.Chen等人[158]发现一种可导致CDN资源耗尽的循环转发攻击,根据所做实验的结果,他们建议所有CDN提供商应该在请求中添加一个统一的循环检测头来防御这种攻击.在互联网软件体系结构中,端到端通信安全是基本的保障,因此,TLS协议被设计出来,以提供端到端的通信安全.尽管TLS的核心密码学算法被证明是安全的,但是一些额外的特性可能会导致TLS的安全性降低.2009年,Ray等人利用TLS重协商成功发起中间人攻击[159];2012年,Mavrogiannopoulos等人展示了利用不同加密套接字间的交互来进行的跨协议攻击[160].同时,只有正确地使用TLS协议才能保障通信安全.2012年,Georgiev等人发现很多非浏览器应用未能正确使用TLS进行连接[161];同年,SaschaFahl等人研制的MalloDroid工具基于静态分析发现大量的Android应用软件错误地使用了TLS协议[162].TLS的保护策略很大程度上依赖于对X.509证书的验证.2014年,Brubaker等人通过自动生成错误的测试证书来检测软件是否正确,验证了TLS返回的证书[163].He等人于2015年Oakland大会上展示的SSLINT工具基于静态分析技术[164],通过对数据依赖图的挖掘,可以有效地检测软件对TLS API的误用.

3.3 面向终端用户的软件安全机制

3.3.1 基于软件描述的软件敏感行为理解

软件中的敏感行为可能是软件的正常功能,也可能是潜在的恶意行为.软件描述由开发者在软件分发环节提供给用户,以便用户理解软件的功能.在软件的功能描述中,也暗含着软件对敏感资源使用的描述.研究人员通过对软件描述进行自动化分析,识别与敏感行为相关的描述,可以帮助用户理解软件中敏感行为是否出于功能需求.WHYPER系统[165]利用自然语言处理技术,根据软件描述来判断哪些敏感权限是软件在描述中说明的.AutoCog系统[166]在WHYPER的基础上引入敏感行为描述的语义模型,大大提高了分析的准确度和召回率.区别于WHYPER和AutoCog,CHABADA系统[167]采取了另一种思路来研究软件描述与实际行为是否匹配.CHABADA的设计基于软件描述相近的软件功能和行为也应该相近的想法.若某个软件行为与同类软件不同,则可能含有一些恶意的敏感行为.使用自然语言处理技术对软件描述进行分析,虽然能够对软件描述能否帮助用户理解软件中的敏感行为进行检测,但也存在一些局限性:一方面,自然语言处理技术本身还不能完全正确理解文本信息;另一方面,恶意的开发者可能会在软件描述中添加虚假的内容,将一些恶意的敏感行为阐述为软件的正常功能需求.

3.3.2 软件敏感行为的意图识别和表达

软件中的敏感行为在执行时应该能够让用户充分理解这些敏感行为的意图,否则,软件就可能滥用敏感资源.Asdroid 系统[168]通过分析敏感行为发生时的用户界面信息,检测软件中是否存在不符合用户意图的敏感行为.例如:界面中按钮文字为“改变界面背景”,但是软件却在执行发送短信的行为,那么这一行为就可以判定为是不符合用户意图的.Asdroid 的主要局限在于对界面的文本分析可能不够准确,因为部分软件界面可能使用图片或者混淆性的语句.AppContext 工具[169]通过对软件进行分析,识别只有在特定上下文才能产生的软件敏感行为,例如只在晚上执行的敏感行为.AppContext 可以有效检测出软件中是否存在用户无法感知的敏感行为.隐私搜集行为是软件中非常主流的一类敏感行为,复旦大学的研究团队发现很多软件都存在隐私数据搜集行为,它们是否有恶意难以识别.为了解决这一问题,该团队提出用户感知度作为判定隐私搜集行为是否有恶意的判定标准,并研制了 AppIntent 系统[170]用于识别隐私搜集行为过程中的用户感知度.DESCRIBEME 工具[171]则通过分析软件中的敏感行为,自动生成相应的文字表述来告知用户软件中存在的风险.该工具可以帮助终端用户更好地理解软件的敏感行为,从而评估软件的风险.

3.3.3 基于用户交互的访问控制研究

访问控制是软件安全机制研究的一个重要方面.在互联网环境下,系统内部包含的敏感资源丰富且复杂,这就要求用户能够正确地授予敏感资源的访问权限.目前,主流的权限授予方式分为两类:一类是安装时权限授予,另一类是使用时权限授予.但是这两种方式都不满足最小特权原则(principle-of-least-privilege),因为一旦被授予权限,即使之后应用实际功能并不需要该权限,应用还是可以一直使用它.Roesner 等人[172]提出了用户驱动的访问控制(user-driven access control)机制,在该机制中,为了使用敏感资源,开发者需要在界面中引入操作系统定义的 ACG(access control gadget)组件.用户在使用软件的过程中,可根据软件的意图,通过 ACG 将相关资源的访问权限授予该软件.通过这种方式,开发者会在 ACG 出现的地方尽可能地让用户理解访问敏感资源的意图,从而帮助用户进行决策;操作系统保证程序只有通过 ACG 才能获得敏感资源.ACG 的实施需要满足两个要求:一是真实可靠地将 ACG 显示给用户;二是确保用户针对 ACG 的操作不可被伪造.针对这个问题,Roesner 等人在 USENIX Security 大会展示的 LayerCake 系统[173]基于多进程隔离来提供 ACG 的实施能力.由于 LayerCake 系统很难部署到现有操作系统中,Ringer 等人[174]提出了基于安全库(secure library)支持用户驱动的访问控制的系统 AUDACIOUS.基于 ACG 的用户驱动访问控制保证了最小特权原则,同时也具有较好的可用性.但是该方案也有局限性.首先,它并不能防止社会工程攻击(social engineering attack),比如应用通过某种方式欺骗或诱惑用户去点击 ACG 让系统授权给它;其次,并非所有的权限都能找到合适的 ACG,因此,该方案无法用于所有资源的保护.

软件安全机制的目标主要是保护可信软件的安全运行,同时防范不可信软件的攻击.针对这两大问题,学术界从操作系统、虚拟化、安全硬件、编译和程序执行技术等多个层次,融合自然语言处理、密码学等相关领域的积累展开了大量的研究.然而,现有软件安全机制的研究受制于软件产业的飞速发展,大多在被动地解决问题.随着网络技术的进一步发展,越来越多的设备将具备计算能力并接入网络,软件安全的重要性将会更加突出,迫切需要学术界运用超前思维开展工作,充分吸收现有的学术成果,为新型软件及软件平台的发展夯实安全根基.此外,现有安全机制的研究大多需要开发者或者用户进行一定的参与才能发挥作用,这也影响了新型安全机制的应用和普及,具备智能化能力的安全机制将会具有更大、更广阔的发展空间.

4 面临的挑战与发展趋势

综合上述对软件安全领域国内外研究现状的分析,我们认为,当前软件生态系统安全主要面临如下挑战.

(1) 随着系统平台的互联异构化和互联网移动化、软件分发机制的集中化和软件开发流程的组件化,软件的设计、开发、组装和分发变得更为便利和灵活.这也导致终端系统中众多软件之间的相互依赖性增强,软件运行的内外部环境变得更为复杂,任何一个环节的疏忽都可能导致整个软件体系遭受攻击.如何评估和保障这种复杂软件系统的安全,成为面临的重大挑战;

(2) 软件功能复杂和软件间普遍需要协同工作的特点,导致业务逻辑的安全架构面临很大困难.尤其是软件版本升级频繁时,如何保障软件的安全质量控制以及如何保障安全补丁的快速可达等,是软件生态系统不可回避的难题;

(3) 软件系统生成和管理着大量敏感的数据,已经成为极高价值的攻击目标.在目前不规范的监管体系中,如何防止敏感数据被滥用、如何更好地规范软件行为以及如何帮助用户更加安全地管理设备和软件,是对互联网行业的一个重大挑战;

(4) 利益驱动的攻击行为日益普遍,攻击者的手段呈现多样化、工具化和分工协作的特征,使得原本就处于被动态势的安全防护方难以应对.如何在知识和数据的支撑下发展主动、智能的安全技术体系,是摆在科研人员面前的一项艰巨任务.

作为功能与服务提供的主体,软件在网络空间的地位和价值越来越突出,软件安全问题也必将成为网络空间安全的核心问题.当前,各种“修修补补”的策略显然难以满足未来的安全防御需求.如何构建更系统、更智能化的防御体系,是未来软件安全领域的发展趋势.2016 年 8 月 4 日,由美国国防部高级研究计划局(DARPA)发起的“网络安全挑战赛(cyber grand challenge,简称 CGC)”旨在实验性地探索无人干预条件下的智能攻防体系,实现完全无人干预的软件漏洞识别、攻击利用和主动防御.该比赛中的各种技术方案与实际应用仍有很大距离,但代表了未来的发展趋势.就当前技术现状而言,恶意软件的防御、软件漏洞的检测与评估、软件安全防护仍是未来软件安全领域发展的重点方向.

随着软件系统朝生态化、互联化和服务化方向发展,如何提高恶意软件的分析、检测和防范,将是网络空间面临的重要挑战.如何实时地识别和分析深度潜藏的系统后门,对保障网络安全具有重要的意义.近年来,尽管我国科研人员在对抗恶意软件攻击方面不乏亮点工作,但仍有很多难题亟待解决.

在恶意攻击防范方面,如何针对一些典型恶意攻击,如拒绝服务攻击等,提出有效的识别和防范方法,是主动防御需要考虑的重要问题;针对层出不穷的新型软件系统和网络技术,如何分析潜在的攻击面和攻击方法以提高网络空间主体的安全,需要重点加以研究.在恶意软件对抗方面,除了软件分析方法,如何综合利用数据分析、深度学习等智能技术来提高恶意软件识别率和分析能力,将成为未来研究的趋势之一.在恶意软件分析方面,如何提高分析的效率和精确度,始终是一个重要问题.如何利用软件分析的方法和技术支持对新平台、新技术中恶意软件的分析以提高分析的深度,也是一个重要的研究课题;特别是提出高效的方法来关联网络流量和软件漏洞等多维度的信息以提高对 APT 等高级攻击的检测和分析的能力,是当前恶意软件中最具有挑战性的问题之一.

随着软件生态趋势的形成,多平台下的软件交互和协同服务、软件结构的复杂化以及软件功能的智能化等趋势,都对于软件质量和安全保障提出了更高的要求和挑战.从软件安全防御的角度来说,如何及时地发现漏洞、分析漏洞,并从攻击者利用的角度评估漏洞进而修补漏洞,是最积极的防御思路.

软件漏洞的研究虽然已经出现了一批具有较高参考价值的研究成果,但当前无论是漏洞的发掘、分析、可利用性评估以及修补,都仍存在众多难点有待突破.在软件漏洞挖掘方面,如何进一步提高软件漏洞挖掘的效率、并行化程度、准确性以及如何减少人为干预,都将是未来发展的重要方向.在软件漏洞分析方面,如何借助大数据、机器学习以及深度学习等技术来提高软件漏洞判定和定位的效率和可靠性,是今后的主要趋势.在软件漏洞利用方面,随着软件网络智能攻防概念的提出,如何进一步实现软件漏洞可利用性智能评估和提高利用自动生成能力,将是最大的挑战和难点之一.

互联网技术的不断创新与发展,倒逼软件安全技术向前推进.在移动互联时代,我国渐进式的互联网技术革命出现了明显的分水岭:商业模式逐渐出现超越欧美的创新能力,显著带动了基础架构的发展,尤为典型的是移动电子商务、移动社交、移动支付、移动金融等领域,呈现出从“模仿”到“赶超”,进而“引领”的态势.相关行业的软件系统,面临着前所未有的各类新型安全问题,但在全球范围内,并没有很多可以借鉴的成功经验,加上安全产品的领域特殊性,需要依靠行业和国内科研院所针对现实问题开展深入研究与探索.

致谢 本文在由林惠民院士主持的中国科学院学部学科发展战略研究“计算机软件”项目的执行过程中逐步形成.特此向该项目的支持表示感谢.中国人民大学的梁斌和中国科学院信息工程研究所的赵双、马新建对本文也提出了很多建设性意见,在此一并感谢.

References:

[1] CNCERT Internet Security Threat Report——2016,6 (in Chinese). http://www.cac.gov.cn/2016-08/01/c_1119418586.htm

[2] National Computer Network Emergency Response Technical Team/Coordination Center of China. An investigation of security attacks relating to Ramnittrojan in China (in Chinese). http://www.cert.org.cn/publish/main/10/2016/20160422145241769412671/20160422145241769412671_.html

[3] National Computer Network Emergency Response Technical Team/Coordination Center of China. Many IoT devices in China hijacked by Mirai botnet (in Chinese). http://www.cert.org.cn/publish/main/12/2016/20161201134333495740421/20161201134333495740421_.html

[4] Skoudis E, Zeltser L. Malware: Fighting Malicious Code. Upper Saddle River: Prentice Hall Professional, 2004.

[5] Sikorski M, Honig A. Practical Malware Analysis: The Hands-on Guide to Dissecting Malicious Software. San Francisco: No Starch Press, 2012.

[6] Weaver N, Paxson V, Staniford S, Cunningham R. A taxonomy of computer worms. In: Proc. of the 2003 ACM Workshop on Rapid Malcode. ACM Press, 2003. 11–18. https://dl.acm.org/citation.cfm?id=948190

[7] Dittrich D, Dietrich S. P2P as botnet command and control: A deeper insight. In: Proc. of the 3rd Int’l Conf. on Malicious and Unwanted Software. IEEE, 2008. 41–48. [doi: 10.1109/MALWARE.2008.4690856]

[8] Xu Z, Zhang J, Gu G, Lin Z. GOLDENEYE: Efficiently and effectively unveiling malware’s targeted environment. In: Proc. of the Int’l Workshop on Recent Advances in Intrusion Detection. Springer Int’l Publishing, 2014. 22–45. [doi: 10.1007/978-3-319-11379-1_2]

[9] Song C, Royal P, Lee W. Impeding automated malware analysis with environment-sensitive malware. In: Proc. of the HotSec. 2012. https://dl.acm.org/citation.cfm?id=2372391

[10] Castillo CA. Android malware past, present, and future. White Paper, McAfee Mobile Security Working Group, 2011. 1–16.

[11] Zhou Y, Jiang X. Dissecting Android malware: Characterization and evolution. In: Proc. of the 2012 IEEE Symp. on Security and Privacy. IEEE, 2012. 95–109. [doi: 10.1109/SP.2012.16]

[12] Krahmer S. Zimperlich sources. 2011. http://c-skills.blogspot.com/2011/02/zimperlich-sources.html

[13] Jiang X. Security alert: New DroidKungFu variant—AGAIN!—Found in alternative Android markets. 2011. http://www.csc.ncsu.edu/faculty/jiang/DroidKungFu3/

[14] Fang Z, Han W, Li Y. Permission based Android security: Issues and countermeasures. Computers & Security, 2014,43:205–218. [doi: 10.1016/j.cose.2014.02.007]

[15] Bugiel S, Davi L, Dmitrienko A, Shastry B. Towards taming privilege-escalation attacks on Android. In: Proc. of the 19th Annual Network and Distributed System Security Symp. (NDSS 2012). 2012. 17–19. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.468.8514

[16] Marforio C, Francillon A, Capkun S. Application collusion attack on the permission-based security model and its implications for modern smartphone systems. Technical Report, Zürich: Department of Computer Science, 2011. [doi: 10.3929/ethz-a-006720730]

[17] Schlegel R, Zhang K, Zhou X, Intwala M, Kapadia A, Wang XF. Soundcomber: A stealthy and context-aware sound Trojan for smartphones. In: Proc. of the 18th Annual Network and Distributed System Security Symp. (NDSS 2011). 2011. 17–33. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.363.1699&rep=rep1&type=pdf

[18] Egele M, Brumley D, Fratantonio Y, Kruegel G. An empirical study of cryptographic misuse in Android applications. In: Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security. ACM Press, 2013. 73–84. [doi: 10.1145/2508859.2516693]

[19] Sounthiraraj D, Sahs J, Greenwood G, Lin Z, Khan L. SMV-Hunter: Large scale, automated detection of SSL/TLS man-in-the-middle vulnerabilities in Android apps. In: Proc. of the 21st Annual Network and Distributed System Security Symp. (NDSS 2014). 2014. [doi: 10.14722/ndss.2014.23205]

[20] Qing SH. Research progress on Android security. Ruan Jian Xue Bao/Journal of Software, 2016,27(1):45–71 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4914.htm [doi: 10.13328/j.cnki.jos.004914]

[21] Chin E, Felt AP, Greenwood K, Wagner D. Analyzing inter-application communication in Android. In: Proc. of the 9th Int'l Conf. on Mobile Systems, Applications, and Services. ACM Press, 2011. 239−252. https://dl.acm.org/citation.cfm?id=2000018

[22] Greengard S. Cybersecurity gets smart. Communications of the ACM, 2016,59(5):29−31. [doi: 10.1145/2898969]

[23] Zhou W, Zhou Y, Grace M, Jiang X, Zou X. Fast, scalable detection of piggybacked mobile applications. In: Proc. of the 3rd ACM Conf. on Data and Application Security and Privacy. ACM Press, 2013. 185−196. [doi: 10.1145/2435349.2435377]

[24] Chen K, Wang XQ, Chen Y, Wang P, Lee Y, Wang XF, Ma B, Wang AH, Zhang YJ, Zou W. Following devils footprints: Crossplatform analysis of potentially harmful libraries on Android and iOS. In: Proc. of the 37th IEEE Symp. on Security and Privacy, Ser. (S&P 2016). 2016. [doi: 10.1109/SP.2016.29]

[25] Backes M, Bugiel S, Derr E. Reliable third-party library detection in Android and its security applications. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. ACM Press, 2016. 356−367. [doi: 10.1145/2976749.2978333]

[26] Li MH, Wang W, Wang P, Wu DH, Liu J, Xue R, Huo W. LibD: Scalable and precise third-party library detection in Android markets. In: Proc. of the 39th Int'l Conf. on Software Engineering. ACM Press, 2017. [doi: 10.1109/ICSE.2017.38]

[27] Jing Q, Vasilakos AV, Wan J, Lu J, Qiu D. Security of the Internet of things: Perspectives and challenges. Wireless Networks, 2014, 20(8):2481−2501. [doi: 10.1007/s11276-014-0761-7]

[28] Cao Z, Hu J, Chen Z, Xu M, Zhou X. Feedback: Towards dynamic behavior and secure routing for wireless sensor networks. In: Proc. of the 20th Int'l Conf. on Advanced Information Networking and Applications—Vol.2. IEEE Computer Society, 2006. 160−164. [doi: 10.1109/AINA.2006.179]

[29] Jhaveri RH, Patel SJ, Jinwala DC. DoS attacks in mobile ad hoc networks: A survey. In: Proc. of the 2nd Int'l Conf. on Advanced Computing & Communication Technologies. IEEE, 2012. 535−541. [doi: 10.1109/ACCT.2012.48]

[30] Chen TM, Abu-Nimeh S. Lessons from Stuxnet. Computer, 2011,44(4):91−93. [doi: 10.1109/MC.2011.115]

[31] Douceur JR. The Sybil attack. In: Proc. of the Int'l Workshop on Peer-to-Peer Systems. Berlin, Heidelberg: Springer-Verlag, 2002. 251−260. [doi: 10.1007/3-540-45748-8_24]

[32] Hlavacs H, Treutner T, Gelas JP, Lefevre L, Orgerie AC. Energy consumption side-channel attack at virtual machines in a cloud. In: Proc. of the 9th Int'l Conf. on Dependable, Autonomic and Secure Computing (DASC). IEEE, 2011. 605−612. [doi: 10.1109/DASC.2011.110]

[33] Wu Z, Xu Z, Wang H. Whispers in the hyper-space: High-Speed covert channel attacks in the cloud. In: Proc. of the 21st USENIX Security Symp. 2012. 159−173. https://dl.acm.org/citation.cfm?id=2362802

[34] Liu F, Yarom Y, Ge Q, Heiser G, Lee RB. Last-Level cache side-channel attacks are practical. In: Proc. of the IEEE Symp. on Security and Privacy. 2015. 605−622. [doi: 10.1109/SP.2015.43]

[35] Rocha F, Correia M. Lucy in the sky without diamonds: Stealing confidential data in the cloud. In: Proc. of the 2011 IEEE/IFIP the 41st Int'l Conf. on Dependable Systems and Networks Workshops (DSN-W). IEEE, 2011. 129−134. [doi: 10.1109/DSNW.2011.5958798]

[36] Hong S, Xu L, Wang H, Gu G. Poisoning network visibility in software-defined networks: New attacks and countermeasures. In: Proc. of the 22th Annual Network and Distributed System Security Symp. (NDSS 2015). 2015. [doi: 10.14722/ndss.2015.23283]

[37] Dhawan M, Poddar R, Mahajan K, Mann V. SPHINX: Detecting security attacks in software-defined networks. In: Proc. of the 22th Annual Network and Distributed System Security Symp. (NDSS 2015). 2015. [doi: 10.14722/ndss.2015.23064]

[38] Slopek A, Vlajic N. Economic denial of sustainability (EDoS) attack in the cloud using Web-bugs. In: Proc. of the 17th Int'l Symp. on Research in Attacks, Intrusions, and Defenses (RAID 2014). Switzerland: Springer Int'l Publishing. 2014. 469−471. https://link.springer.com/book/10.1007/978-3-319-11379-1#page=482

[39] Zhao S, Lee PPC, Lui J, Guan X, Ma X, Tao J. Cloud-Based push-styled mobile botnets: A case study of exploiting the cloud to device messaging service. In: Proc. of the 28th Annual Computer Security Applications Conf. ACM Press, 2012. 119−128. [doi: 10.1145/2420950.2420968]

[40] Tankard C. Advanced persistent threats and how to monitor and deter them. Network Security, 2011,2011(8):16−19. [doi: 10.1016/S1353-4858(11)70086-1]

[41] Li F, Lai A, Ddl D. Evidence of advanced persistent threat: A case study of malware for political espionage. In: Proc. of the 6th Int'l Conf. on Malicious and Unwanted Software (MALWARE). IEEE, 2011. 102−109. [doi: 10.1109/MALWARE.2011.6112333]

[42] Juels A, Yen TF. Sherlock Holmes and the case of the advanced persistent threat. In: Proc. of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats. 2012. https://dl.acm.org/citation.cfm?id=2228343

[43] Linn C, Debray S. Obfuscation of executable code to improve resistance to static disassembly. In: Proc. of the 10th ACM Conf. on Computer and Communications Security. ACM Press, 2003. 290−299. [doi: 10.1145/948109.948149]

[44] Moser A, Kruegel C, Kirda E. Limits of static analysis for malware detection. In: Proc. of the 23rd Annual Computer Security Applications Conf. IEEE, 2007. 421−430. [doi: 10.1109/ACSAC.2007.21]

[45] Ma XJ. Sandbox based intelligent malware analysis technology [Ph.D. Thesis]. Beijing: University of Chinese Academy of Sciences, 2017 (in Chinese with English abstract).

[46] Kolbitsch C, Kirda E, Kruegel C. The power of procrastination: Detection and mitigation of execution-stalling malicious code. In: Proc. of the 18th ACM Conf. on Computer and Communications Security. ACM Press, 2011. 285−296. [doi: 10.1145/2046707.2046740]

[47] Kharraz A, Robertson W, Balzarotti D, Bilge L, Kirda E. Cutting the Gordian knot: A look under the hood of ransomware attacks. In: Proc. of the Int'l Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer Int'l Publishing, 2015. [doi: 10.1007/978-3-319-20550-2_1]

[48] Google LLC. Google Play. All your entertainment, anywhere you go. 2012. http://googleblog.blogspot.co.uk/2012/03/introducing-google-play-all-your.html

[49] Kovacheva A. Efficient code obfuscation for Android. In: Proc. of the Int'l Conf. on Advances in Information Technology. Springer Int'l Publishing, 2013. 104−119. [doi: 10.1007/978-3-319-03783-7_10]

[50] Faruki P, Bharmal A, Laxmi V, Gaur MS, Conti M, Rajarajan M. Evaluation of Android anti-malware techniques against Dalvik bytecode obfuscation. In: Proc. of the IEEE 13th Int'l Conf. on Trust, Security and Privacy in Computing and Communications. IEEE, 2014. 414−421. [doi: 10.1109/TrustCom.2014.54]

[51] Li L, Bissyandé TF, Octeau D, Klein J. Droidra: Taming reflection to support whole-program analysis of Android apps. In: Proc. of the 25th Int'l Symp. on Software Testing and Analysis. ACM Press, 2016. 318−329. [doi: 10.1145/2931037.2931044]

[52] Rastogi V, Chen Y, Jiang X. Droidchameleon: Evaluating Android anti-malware against transformation attacks. In: Proc. of the 8th ACMSIGSAC Symp. on Information,Computer and Communications Security (ASIACCS 2013). 2013. 329−334. http://dl.acm.org/citation.cfm?id=2484355

[53] Egele M, Scholte T, Kirda E, Kruegel C. A survey on automated dynamic malware-analysis techniques and tools. ACM Computing Surveys (CSUR), 2012,44(2):6. [doi: 10.1145/2089125.2089126]

[54] Willems C, Holz T, Freiling F. Toward automated dynamic malware analysis using cwsandbox. IEEE Security and Privacy, 2007, 5(2):32−39. [doi: 10.1109/MSP.2007.45]

[55] Bayer U, Moser A, Kruegel C, Kirda E. Dynamic analysis of malicious code. Journal in Computer Virology, 2006,2(1):67−77. [doi: 10.1007/s11416-006-0012-2]

[56] Dinaburg A, Royal P, Sharif M, Lee W. Ether: Malware analysis via hardware virtualization extensions. In: Proc. of the 15th ACM Conf. on Computer and Communications Security. ACM Press, 2008. 51−62. [doi: 10.1145/1455770.1455779]

[57] Xu D, Ming J, Wu D. Cryptographic function detection in obfuscated binaries via bit-precise symbolic loop mapping. In: Proc. of the 38th IEEE Symp. on Security and Privacy. 2017. 22−24. [doi: 10.1109/SP.2017.56]

[58] Park Y, Reeves DS, Stamp M. Deriving common malware behavior through graph clustering. Computers & Security, 2013,39: 419−430. [doi: 10.1016/j.cose.2013.09.006]

[59] Liang Z, Yin H, Song D. HookFinder: Identifying and understanding malware hooking behaviors. In: Proc. of the Network & Distributed System Security Symp. 2008. 41. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.8123&rep=rep1&type=pdf

[60] Wu DJ, Mao CH, Wei TE, Lee HM, Wu KP. Droidmat: Android malware detection through manifest and API calls tracing. In: Proc. of the 7th Asia Joint Conf. on Information Security (Asia JCIS). IEEE, 2012. 62−69. [doi: 10.1109/AsiaJCIS.2012.18]

[61] Burguera I, Zurutuza U, Nadjm-Tehrani S. Crowdroid: Behavior-Based malware detection system for Android. In: Proc. of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. ACM Press, 2011. 15−26. [doi: 10.1145/2046614.2046619]

[62] Chen ZF, Li QB, Zhang P, Ding WB. Data characteristics-based kernel malware detection. Ruan Jian Xue Bao/Journal of Software, 2016,27(12):3172−3191 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4927.htm [doi: 10.13328/j.cnki.jos.004927]

[63] Christodorescu M, Jha S, Seshia SA, Song D, Bryant R. Semantics-Aware malware detection. In: Proc. of the 2005 IEEE Symp. on Security and Privacy (S&P 2005). IEEE, 2005. 32−46. [doi: 10.1109/SP.2005.20]

[64] Kirda E, Kruegel C, Banks G, Vigna G, Kemmerer RA. Behavior-Based spyware detection. In: Proc. of the Usenix Security. 2006. 6. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.148.8514

[65] Fuchs AP, Chaudhuri A, Foster JS. ScAndroid: Automated security certification of Android. 2009. https://www.researchgate.net/publication/228847936_SCanDroid_Automated_security_certification_of_Android_applications

[66] Chan PPF, Hui LCK, Yiu SM. Droidchecker: Analyzing Android applications for capability leak. In: Proc. of the 5th ACM Conf. on Security and Privacy in Wireless and Mobile Networks. ACM Press, 2012. 125−136. [doi: 10.1145/2185448.2185466]

[67] Schultz MG, Eskin E, Zadok F, Zadok E, Stolfo SJ. Data mining methods for detection of new malicious executables. In: Proc. of the 2001 IEEE Symp. on Security and Privacy. IEEE, 2001. 38−49. [doi: 10.1109/SECPRI.2001.924286]

[68] Rieck K, Trinius P, Willems C, Holz T. Automatic analysis of malware behavior using machine learning. Journal of Computer Security, 2011,19(4):639−668. [doi: 10.3233/JCS-2010-0410]

[69] Amos B, Turner H, White J. Applying machine learning classifiers to dynamic Android malware detection at scale. In: Proc. of the 9th Int'l Wireless Communications and Mobile Computing Conf. (IWCMC). IEEE, 2013. 1666−1671. [doi: 10.1109/IWCMC.2013.6583806]

[70] Sadeghi A, Bagheri H, Garcia J. A taxonomy and qualitative comparison of program analysis techniques for security assessment of android software. IEEE Trans. on Software Engineering, 2016. [doi: 10.1109/TSE.2016.2615307]

[71] Cui J. The analysis and research of proxy and VPN communication software [MS. Thesis]. Beijing: Beijing University of Posts and Telecommunications, 2012 (in Chinese with English abstract).

[72] Wilhelm J, Chiueh T. A forced sampled execution approach to kernel rootkit identification. In: Proc. of the Workshop on Recent Advances in Intrusion Detection. Berlin, Heidelberg: Springer-Verlag, 2007. 219−235. [doi: 10.1007/978-3-540-74320-0_12]

[73] Moser A, Kruegel C, Kirda E. Exploring multiple execution paths for malware analysis. In: Proc. of the 2007 IEEE Symp. on Security and Privacy (SP 2007). IEEE, 2007. 231−245. [doi: 10.1109/SP.2007.17]

[74] Cadar C, Ganesh V, Pawlowski PM, Dill DL, Engler DR. EXE: Automatically generating inputs of death. ACM Trans. on Information and System Security, 2008,12(2):10. [doi: 10.1145/1180405.1180445]

[75] Comparetti PM, Salvaneschi G, Kirda E, Kolbitsch C, Kruegelx C, Zanero S. Identifying dormant functionality in malware programs. In: Proc. of the 2010 IEEE Symp. on Security and Privacy. IEEE, 2010. 61−76. [doi: 10.1109/SP.2010.12]

[76] Enck W, Gilbert P, Han S, Tendulkar V, Chun BG, Cox LP, Jung J, Mcdaniel P, Sheth AN. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. ACM Trans. on Computer Systems, 2014,32(2):5. [doi: 10.1145/2494522]

[77] Christodorescu M, Jha S, Kruegel C. Mining specifications of malicious behavior. In: Proc. of the 1st India Software Engineering Conf. ACM Press, 2008. 5−14. [doi: 10.1145/1342211.1342215]

[78] Zhang HL, Zou W, Han XH. Drive-by-Download mechanisms and defenses. Ruan Jian Xue Bao/Journal of Software, 2013,24(4):843−858 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4376.htm [doi: 10.3724/SP.J.1001.2013.04376]

[79] Common vulnerabilities and exposures. https://cve.mitre.org

[80] Bass T, Gruber D. A glimpse into the future of id. Login: Special Issue Intrusion Detection, The USENIX Association Magazine, 1999, 40−45.

[81] Sun H, Li HP, Zeng QK. Statically detect and run-time check integer-based vulnerabilities with information flow. Ruan Jian Xue Bao/Journal of Software, 2013,24(12):2767−2781 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4385.htm [doi: 10.3724/SP.J.1001.2013.04385]

[82] Brumley D, Poosankam P, Song D, Zheng J. Automatic patch-based exploit generation is possible: Techniques and implications. In: Proc. of the 2008 IEEE Symp. on Security and Privacy (SP 2008). IEEE, 2008. 143−157. [doi: 10.1109/SP.2008.17]

[83] Cadar C, Dunbar D, Engler D. KLEE: Unassisted and automatic generation of high coverage tests for complex systems programs. In: Proc. of the OSDI 2008. 2008. http://zoo.cs.yale.edu/classes/cs422/2010/bib/engler08klee.pdf

[84] Godefroid P, Levin MY, Molnar D. Automated whitebox fuzz testing. In: Proc. of the 15th Annual Network and Distributed System Security Symp. (NDSS 2008). 2008. http://www.microsoft.com/en-us/research/wp-content/uploads/2007/05/tr-2007-58.pdf

[85] Engler D, Chen DY, Hallem S, Chou A, Chelf B. Bugsas deviant behavior: A general approach to inferring errors in systems code. In: Proc. of the ACM Symp. on Operating Systems Principles (SOSP). 2001. 57−72. http://web.stanford.edu/~engler/deviant-sosp-01.pdf

[86] Yamaguchi F, Wressnegger C, Gascon H, Rieck K. Chucky: Exposing missing checks in source code for vulnerability discovery. In: Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security. 2013. 499−510. [doi: 10.1145/2508859.2516665]

[87] Grieco G, Grinblat GL, Uzal L, Rawat S, Feist J, Mounier L. Toward large-scale vulnerability discovery using machine learning. In: Proc. of the Codaspy. 2016. 85−96. [doi: 10.1145/2857705.2857720]

[88] Yin H, Song D. Temu: Binary code analysis via whole-system layered annotative execution. Technical Report, UCB/EECS-2010-3, Berkeley: EECS Department, University of California, 2010.

[89] Henderson A, Prakash A, Yan LK, Hu X. Make it work, make it right, make it fast: Building a platform-neutral whole-system dynamic binary analysis platform. In: Proc. of the ISSTA 2014. 2014. http://www.cs.ucr.edu/~heng/pubs/issta14.pdf

[90] Kemerlis VP, Portokalidis G, Jee K, Keromytis A. libdft: Practical dynmaic data flow tracking for commodity systems. In: Proc. of the VEE 2012. 2012. [doi: 10.1145/2151024.2151042]

[91] Kang MG, Mccamant S, Poosankam P, Song D. DTA++: Dynamic taint analysis with targeted control-flow propagation. In: Proc. of the Network and Distributed System Security Symp. (NDSS 2011). 2011. https://people.eecs.berkeley.edu/~dawnsong/papers/2011%20dta++-ndss11.pdf

[92] Clause J, Li W, Orso A. Dytan: A generic dynamic taint analysis framework. In: Proc. of the Int'l Symp. on Software Testing and Analysis (ISSTA 2007). London, 2007. 196−206. https://www.cc.gatech.edu/fac/Alex.Orso/papers/clause.li.orso.ISSTA07.pdf

[93] Cui W, Peinado M, Cha SK, Fratantonio Y, Kemerlis VP. Retracer: Triaging crashes by reverse execution from partial memory dumps. In: Proc. of the 38th Int'l Conf. on Software Engineering. 2016. [doi: 10.1145/2884781.2884844]

[94] Xu J, Mu D, Chen P, Xing X, Wang P, Liu P. CREDAL: Towards locating a memory corruption vulnerability with your core dump. In: Proc. of the CCS 2016. 2016. [doi: 10.1145/2976749.2978340]

[95] Sidiroglou-Douskos S, Lahtinen E, Rittenhouse N, Piselli P, Long F, Kim D, Rinard M. Targeted automatic interger overflow discovery using goal-directed conditional branch enforcement. In: Proc. of the ASPLOS 2015. 2015. [doi: 10.1145/2775054.2694389]

[96] Newsome J, Song D. Dynamic taint analysis: Automatic detection, analysis, and signature generation of exploit attacks on commodity software. In: Proc. of the 12th Network and Distributed Systems Security Symp. 2005. http://users.ece.cmu.edu/~dawnsong/papers/taintcheck.pdf

[97] Wu R, Zhang H, Cheung SC, Kim S. Crashlocator: Locating crashing faults based on crash stacks. In: Proc. of the 2014 Int'l Symp. on Software Testing and Analysis. 2014. [doi: 10.1145/2610384.2610386]

[98] Brumley D, Poosankam P, Song D, Zheng J. Automatic patch-based exploit generation is possible: Techniques and implications. In: Proc. of the IEEE Symp. on Security and Privacy. IEEE, 2008. 143−157. [doi: 10.1109/SP.2008.17]

[99] Sha L, Fu J, Chen J, Peng G. PVDF: An automatic patch-based vulnerability description and fuzzing method. In: Proc. of the Communications Security Conf. IET, 2014. 1−8. [doi: 10.1049/cp.2014.0733]

[100] Zhang M, Yin H. Automatic generation of vulnerability-specific patches for preventing component hijacking attacks. In: Proc. of the Android application security. Springer Int'l Publishing, 2016. 45−61. [doi: 10.1007/978-3-319-47812-8_4]

[101] Cha SK, Avgerinos T, Rebert A, Brumley D. Unleashing mayhem on binary code. In: Proc. of the IEEE Symp. on Security and Privacy. IEEE, 2012. 380−394. [doi: 10.1109/SP.2012.31]

[102] Wang MH, Su P, Li Q, Ying L, Yang Y, Feng D. Automatic polymorphic exploit generation for software vulnerabilities. In: Proc. of the Int'l Conf. on Security and Privacy in Communication Systems. Springer Int'l Publishing, 2013. 216−233. [doi: 10.1007/978-3-319-04283-1_14]

[103] Hu H, Chua ZL, Adrian S, Saxena P, Liang Z. Automatic generation of data-oriented exploits. In: Proc. of the 24th USENIX Security Symp. (USENIX Security 2015). 2015. 177−192. https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-hu.pdf

[104] Hu H, Shinde S, Adrian S, Chua ZL, Saxena P, Liang Z. Data-Oriented programming: On the expresivenes of non-control data attacks. In: Proc. of the S&P 2016. 2016. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7546545

[105] Hornyack P, Han S, Jung J, Schechter S, Wetherall D. These are not the droids you're looking for: Retrofitting Android to protect data from imperious applications. In: Proc. of the 18th ACM Conf. on Computer and Communications Security. ACM Press, 2011. [doi: 10.1145/2046707.2046780]

[106] Zhou Y, Zhang X, Jiang X, Freeh VW. Taming information-stealing smartphone applications (on Android). In: Proc. of the Int'l Conf. on Trust and Trustworthy Computing. Berlin, Heidelberg: Springer-Verlag, 2011. 93−107. [doi: 10.1007/978-3-642-21599-5_7]

[107] Fawaz K, Shin KG. Location privacy protection for smartphone users. In: Yung M, Li N, eds. Proc. of the 2014 ACM SIGSAC Conf. on Computer and Communications Security. Scottsdale: ACM Press, 2014. 239−250.

[108] Beresford AR, Rice A, Skehin N, Sohan R. Mockdroid: Trading privacy for application functionality on smartphones. In: Proc. of the 12th Workshop on Mobile Computing Systems and Applications. ACM Press, 2011. 49−54. [doi: 10.1145/2184489.2184500]

[109] Pearce P, Felt AP, Nunez G, Wagner D. Addroid: Privilege separation for applications and advertisers in Android. In: Proc. of the 7th ACM Symp. on Information, Computer and Communications Security. ACM Press, 2012. 71−72. [doi: 10.1145/2414456.2414498]

[110] Shekhar S, Dietz M, Wallach DS. ADsplit: Separating smartphone advertising from applications. In: Kohno T, ed. Proc. of the 21st USENIX Security Symp. Bellevue: USENIX Association, 2012. 553−567.

[111] Zhang X, Ahlawat A, Du W. AFrame: Isolating advertisements from mobile applications in Android. In: Proc. of the 29th Annual Computer Security Applications Conf. ACM Press, 2013. 9−18. [doi: 10.1145/2523649.2523652]

[112] Wu C, Zhou Y, Patel K, Liang Z, Jiang X. AirBag: Boosting smartphone resistance to malware infection. In: Bauer L, ed. Proc. of the 21st Annual Network and Distributed System Security Symp. (NDSS 2014). San Diego: Internet Society, 2014.

[113] Liu Y, Zhou T, Chen K, Chen H, Xia Y. Thwarting memory disclosure with efficient hypervisor-enforced intra-domain isolation. In: Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security. ACM Press, 2015. 1607−1619. [doi: 10.1145/2810103.2813690]

[114] Kurmus A, Zippel R. A tale of two kernels: Towards ending kernel hardening wars with split kernel. In: Proc. of the 2014 ACM SIGSAC Conf. on Computer and Communications Security. ACM Press, 2014. 1366−1377. [doi: 10.1145/2660267.2660331]

[115] Zhou Z, Yu M, Gligor VD. Dancing with giants: Wimpy kernels for on-demand isolated I/O. In: Proc. of the 2014 IEEE Symp. on Security and Privacy. IEEE, 2014. 308−323. [doi: 10.1109/SP.2014.27]

[116] Nikolaev R, Back G. VirtuOS: An operating system with kernel virtualization. In: Proc. of the 24th ACM Symp. on Operating Systems Principles. ACM Press, 2013. 116−132. [doi: 10.1145/2517349.2522719]

[117] Azab AM, Ning P, Shah J, Chen Q. Hypervision across worlds: Real-Time kernel protection from the arm trustzone secure world. In: Proc. of the 2014 ACM SIGSAC Conf. on Computer and Communications Security. ACM Press, 2014. 90−102. [doi: 10.1145/2660267.2660350]

[118] Li W, Li H, Chen H, Xia Y. Adattester: Secure online mobile advertisement attestation using trustzone. In: Proc. of the 13th Annual Int'l Conf. on Mobile Systems, Applications, and Services. ACM Press, 2015. 75−88. [doi: 10.1145/2742647.2742676]

[119] Zhou Y, Wang X, Chen Y, Wang Z. Armlock: Hardware-Based fault isolation for arm. In: Proc. of the 2014 ACM SIGSAC Conf. on Computer and Communications Security. ACM Press, 2014. 558−569. [doi: 10.1145/2660267.2660344]

[120] Guan L, Lin J, Luo B, Jing J, Wang J. Protecting private keys against memory disclosure attacks using hardware transactional memory. In: Proc. of the 2015 IEEE Symp. on Security and Privacy. IEEE, 2015. 3−19. [doi: 10.1109/SP.2015.8]

[121] Xu R, Saïdi H, Anderson R. Aurasium: Practical policy enforcement for Android applications. In: Kohno T, ed. Proc. of the 21st USENIX Security Symp. Bellevue: USENIX Association, 2012. 539−552.

[122] Rastogi V, Qu Z, McClurg J, Cao Y, Chen Y. Uranine: Real-Time privacy leakage monitoring without system modification for Android. In: Proc. of the Int'l Conf. on Security and Privacy in Communication Systems. Springer Int'l Publishing, 2015. 256−276. [doi: 10.1007/978-3-319-28865-9_14]

[123] Backes M, Bugiel S, Hammer C, Schranz O, Styp-Rekowsky P. Boxify: Full-Fledged app sandboxing for stock Android. In: Jung J, ed. Proc. of the 24th USENIX Security Symp. Washington: USENIX Association, 2015. 691−706.

[124] Smalley S, Craig R. Security enhanced (SE) Android: Bringing flexible MAC to Android. In: Ning P, ed. Proc. of the 20th Annual Network and Distributed System Security Symp. (NDSS 2013). San Diego: Internet Society, 2013. 20−38.

[125] Bugiel S, Heuser S, Sadeghi AR. Flexible and fine-grained mandatory access control on Android for diverse security and privacy policies. In: King S, ed. Proc. of the 22nd USENIX Security Symp. Washington: USENIX Association, 2013. 131−146.

[126] Heuser S, Nadkarni A, Enck W, Sadeghi AR. ASM: A programmable interface for extending Android security. In: Fu K, ed. Proc. of the 23rd USENIX Security Symp. San Diego: USENIX Association, 2014. 1005−1019.

[127] Backes M, Bugiel S, Gerling S, Styp-Rekowsky P. Android security framework: Extensible multi-layered access control on Android. In: Proc. of the 30th Annual Computer Security Applications Conf. ACM Press, 2014. 46−55. [doi: 10.1145/2664243.2664265]

[128] Zhang Y, Yang M, Gu G, Chen H. Rethinking permission enforcement mechanism on mobile systems. IEEE Trans. on Information Forensics and Security, 2016,11(10):2227−2240. [doi: 10.1109/TIFS.2016.2581304]

[129] Wang R, Enck W, Reeves D, Zhang X. EASEAndroid: Automatic policy analysis and refinement for security enhanced Android via large-scale semi-supervised learning. In: Jung J, ed. Proc. of the 24th USENIX Security Symp. Washington: USENIX Association, 2015. 351−366.

[130] Cowan C, Pu C, Maier D, Hinton H, Walpole J. StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. In: Rubin A, ed. Proc. of the Conf. on Usenix Security Symp. San Antonio: USENIX Association, 1998. 63−78.

[131] Trev N. Data Execution Prevention. Lect Publishing, 2011. http://nosmut.com/Executable_space_protection.html

[132] Miller FP, Vandome AF, Mcbrewster J. Address Space Layout Randomization. Alphascript Publishing, 2010. http://nosmut.com/Address_space_layout_randomization.html

[133] Prandini M, Ramilli M. Return-Oriented programming. IEEE Security & Privacy Magazine, 2012,10(6):84–87. [doi: 10.1109/MSP.2012.152]

[134] Abadi M, Budiu M, Erlingsson U, Ligatti J. Control-Flow integrity. In: Proc. of the 12th ACM Conf. on Computer and Communications Security. Alexandria: ACM Press, 2005. 340–353. [doi: 10.1145/1102120.1102165]

[135] Nie M, Su P, Li Q, Wang Z, Ying L, Hu J, Feng D. XEDE: Practical exploit early detection. In: Proc. of the 18th Int'l Symp. on Research in Attacks, Intrusions and Defenses (RAID 2015). Springer-Verlag, 2015. 198–221. [doi: 10.1007/978-3-319-26362-5_10]

[136] Monshizadeh M, Naldurg P, Venkatakrishnan VN. MACE: Detecting privilege escalation vulnerabilities in Web applications. In: Proc. of the 2014 ACM SIGSAC Conf. on Computer and Communications Security. ACM Press, 2014. 690–701. [doi: 10.1145/2660267.2660337]

[137] Pellegrino G, Balzarotti D. Toward black-box detection of logic flaws in Web applications. In: Bauer L, ed. Proc. of the 21st Annual Network and Distributed System Security Symp. (NDSS 2014). San Diego: Internet Society, 2014.

[138] Weissbacher M, Robertson W, Kirda E, Kruegel C, Vigna G. Zigzag: Automatically hardening Web applications against client-side validation vulnerabilities. In: Jung J, ed. Proc. of the 24th USENIX Security Symp. San Antonio: USENIX Association, 2015. 737–752.

[139] Doupé A, Cui W, Jakubowski MH, Peinado M, Kruegel C, Vigna G. deDacota: Toward preventing server-side XSS via automatic code and data separation. In: Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security. ACM Press, 2013. 1205–1216. [doi: 10.1145/2508859.2516708]

[140] Dietz M, Shekhar S, Pisetsky Y, Shu A, Wallach DS. QUIRE: Lightweight provenance for smart phone operating systems. In: Wagner D, ed. Proc. of the USENIX Security Symp. San Francisco: USENIX Association, 2011. 31.

[141] Chan PPF, Hui LCK, Yiu SM. Droidchecker: Analyzing Android applications for capability leak. In: Proc. of the fifth ACM Conf. on Security and Privacy in Wireless and Mobile Networks. ACM Press, 2012. 125–136. [doi: 10.1145/2185448.2185466]

[142] Lu L, Li Z, Wu Z, Lee W, Jiang G. Chex: Statically vetting Android apps for component hijacking vulnerabilities. In: Proc. of the 2012 ACM Conf. on Computer and Communications Security. ACM Press, 2012. 229–240. [doi: 10.1145/2382196.2382223]

[143] Yang K, Zhuge J, Wang Y, Zhou L, Duan H. IntentFuzzer: Detecting capability leaks of Android applications. In: Proc. of the 9th ACM Symp. on Information, Computer and Communications Security. ACM Press, 2014. 531–536. [doi: 10.1145/2590296.2590316]

[144] Zhang M, Yin H. AppSealer: Automatic generation of vulnerability-specific patches for preventing component hijacking attacks in Android applications. In: Proc. of the 21st Annual Network and Distributed System Security Symp. (NDSS 2014). 2014. [doi: 10.14722/ndss.2014.23255]

[145] Arzt S, Rasthofer S, Fritz C, Bodden E, Bartel A, Klein J, Traon YL, Octeau D, McDaniel P. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. ACM SIGPLAN Notices, 2014,49(6):259–269. [doi: 10.1145/2666356.2594299]

[146] Wei F, Roy S, Ou X. AMAndroid: A precise and general inter-component data flow analysis framework for security vetting of Android apps. In: Proc. of the 2014 ACM SIGSAC Conf. on Computer and Communications Security. ACM Press, 2014. 1329–1341. [doi: 10.1145/2660267.2660357]

[147] Cao Y, Fratantonio Y, Bianchi A, Egelez M, Kruegely C, Vignay G, Chen Y. EdgeMiner: Automatically detecting implicit control flow transitions through the Android framework. In: Proc. of the 22nd Annual Network and Distributed System Security Symp. (NDSS 2015). 2015. [doi: 10.14722/ndss.2015.23140]

[148] Arzt S, Bodden E. StubDroid: Automatic inference of precise data-flow summaries for the Android framework. In: Proc. of the Int'l Conf. on Software Engineering. ACM Press, 2016. [doi: 10.1145/2884781.2884816]

[149] Zhang Y, Yang M, Xu B, Yang Z, Gu G, Ning P, Wang XS, Zhang B. Vetting undesirable behaviors in Android apps with permission use analysis. In: Gligor V, Yung M, eds. Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security. Berlin: ACM Press, 2013. 611–622.

[150] Nan Y, Yang M, Yang Z, Zhou S, Gu G, Wang XF. Uipicker: User-Input privacy identification in mobile applications. In: Jung J, ed. Proc. of the 24th USENIX Security Symp. Washington: USENIX Association, 2015. 993–1008.

[151] Huang J, Li Z, Xiao X, Wu Z, Lu K, Zhang X, Jiang G. Supor: Precise and scalable sensitive user input detection for Android apps. In: Jung J, ed. Proc. of the 24th USENIX Security Symp. Washington: USENIX Association, 2015. 977−992.

[152] Zhang N, Yuan K, Naveed M, Zhou X, Wang XF. Leave me alone: App-Level protection against runtime information gathering on Android. In: Proc. of the 2015 IEEE Symp. on Security and Privacy. IEEE, 2015. 915−930. [doi: 10.1109/SP.2015.61]

[153] Zhang Y, Reiter MK. Düppel: Retrofitting commodity operating systems to mitigate cache side channels in the cloud. In: Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security. ACM Press, 2013. 827−838. [doi: 10.1145/2508859.2516741]

[154] Varadarajan V, Ristenpart T, Swift M. Scheduler-Based defenses against cross-VM side-channels. In: Fu K, ed. Proc. of the 23rd USENIX Security Symp. San Diego: USENIX Association, 2014. 687−702.

[155] Moon SJ, Sekar V, Reiter MK. Nomad: Mitigating arbitrary cloud side channels via provider-assisted migration. In: Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security. ACM Press, 2015. 1595−1606. [doi: 10.1145/2810103.2813706]

[156] Pattuk E, Kantarcioglu M, Lin Z, Ulusoy H. Preventing cryptographic key leakage in cloud virtual machines. In: Fu K, ed. Proc. of the 23rd USENIX Security Symp. San Diego: USENIX Association, 2014. 703−718.

[157] Liang J, Jiang J, Duan H, Li K, Wan T, Wu J. When HTTPS meets CDN: A case of authentication in delegated service. In: Proc. of the 2014 IEEE Symp. on Security and Privacy. IEEE, 2014. 67−82. [doi: 10.1109/SP.2014.12]

[158] Chen J, Jiang J, Zheng X, Duan H, Liang J, Li K, Wan T, Paxson V. Forwarding-Loop attacks in content delivery networks. In: Proc. of the 23th Annual Network and Distributed System Security Symp. (NDSS 2016). 2016. [doi: 10.14722/ndss.2016.23442]

[159] Ray M, Dispensa S. Renegotiating TLS. 2009. https://kryptera.se/Renegotiating%20TLS.pdf

[160] Mavrogiannopoulos N, Vercauteren F, Velichkov V, Preneel B. A cross-protocol attack on the TLS protocol. In: Proc. of the 2012 ACM Conf. on Computer and Communications Security. ACM Press, 2012. 62−72. [doi: 10.1145/2382196.2382206]

[161] Georgiev M, Iyengar S, Jana S, Anubhai R, Boneh D, Shmatikov V. The most dangerous code in the world: Validating SSL certificates in non-browser software. In: Proc. of the 2012 ACM Conf. on Computer and Communications Security. ACM Press, 2012. 38−49. [doi: 10.1145/2382196.2382204]

[162] Fahl S, Harbach M, Muders T, Smith M, Baumgärtner L, Freisleben B. Why eve and mallory love Android: An analysis of Android SSL (in) security. In: Proc. of the 2012 ACM Conf. on Computer and Communications Security. ACM Press, 2012. 50−61. [doi: 10.1145/2382196.2382205]

[163] Brubaker C, Jana S, Ray B, Khurshid S, Shmatikov V. Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations. In: Proc. of the 2014 IEEE Symp. on Security and Privacy. IEEE, 2014. 114−129. [doi: 10.1109/SP.2014.15]

[164] He B, Rastogi V, Cao Y, Chen Y, Venkatakrishnan VN, Yang R, Zhang Z. Vetting SSL usage in applications with SSLint. In: Proc. of the 2015 IEEE Symp. on Security and Privacy. IEEE, 2015. 519−534. [doi: 10.1109/SP.2015.38]

[165] Pandita R, Xiao X, Yang W, Enck W, Xie T. Whyper: Towards automating risk assessment of mobile applications. In: King S, ed. Proc. of the 22nd USENIX Security Symp. Washington: USENIX Association, 2013. 527−542.

[166] Qu Z, Rastogi V, Zhang X, ChenY, ZhuT, Chen Z. Autocog: Measuring the description-to-permission fidelity in Android applications. In: Proc. of the 2014 ACM SIGSAC Conf. on Computer and Communications Security. ACM Press, 2014. 1354−1365. [doi: 10.1145/2660267.2660287]

[167] Gorla A, Tavecchia I, Gross F, Zeller A. Checking app behavior against app descriptions. In: Proc. of the 36th Int'l Conf. on Software Engineering. ACM Press, 2014. 1025−1035. [doi: 10.1145/2568225.2568276]

[168] Huang J, Zhang X, Tan L, Wang P, Liang B. AsDroid: Detecting stealthy behaviors in Android applications by user interface and program behavior contradiction. In: Proc. of the 36th Int'l Conf. on Software Engineering. ACM Press, 2014. 1036−1046. [doi: 10.1145/2568225.2568301]

[169] Yang W, Xiao X, Andow B, Li S, Xie T, Enck W. Appcontext: Differentiating malicious and benign mobile app behaviors using context. In: Proc. of the 37th IEEE Int'l Conf. on Software Engineering. IEEE, 2015. 303−313. [doi: 10.1109/ICSE.2015.50]

[170] Yang Z, Yang M, Zhang Y, Gu G, Ning P, Wang XS. Appintent: Analyzing sensitive data transmission in Android for privacy leakage detection. In: Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security. ACM Press, 2013. 1043−1054. [doi: 10.1145/2508859.2516676]

[171] Zhang M, Duan Y, Feng Q, Yin H. Towards automatic generation of security-centric descriptions for Android apps. In: Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security. ACM Press, 2015. 518−529. [doi: 10.1145/2810103.2813669]

[172] Roesner F, Kohno T, Moshchuk A, Parno B, Wang HJ, Cowan C. User-Driven access control: Rethinking permission granting in modern operating systems. In: Proc. of the 2012 IEEE Symp. on Security and Privacy. IEEE, 2012. 224–238. [doi: 10.1109/SP.2012.24]

[173] Roesner F, Kohno T. Securing embedded user interfaces: Android and beyond. In: Kruegel C, Myers A, Halevi S, eds. Proc. of the 22nd USENIX Security Symp. Vienna: ACM Press, 2013. 97–112.

[174] Ringer T, Grossman D, Roesner F. AUDACIOUS: User-Driven access control with unmodified operating systems. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security (CCS 2016). 2016. 204–216. [doi: 10.1145/2976749.2978344]

附中文参考文献:

[1] CNCERT 互联网安全威胁报告——2016,6. http://www.cac.gov.cn/2016-08/01/c_1119418586.htm

[2] 国家互联网应急中心.关于部分境内网站存在 Ramnit 恶意代码攻击的有关情况通报.2016. http://www.cert.org.cn/publish/main/10/2016/20160422145241769412671/20160422145241769412671_.html

[3] 国家互联网应急中心.植入恶意程序被控制联网智能设备安全隐患多.2016. http://www.cert.org.cn/publish/main/12/2016/20161201134333495740421/20161201134333495740421_.html

[20] 卿斯汉.Android 安全研究进展.软件学报,2016,27(1):45−71. http://www.jos.org.cn/1000-9825/4914.htm [doi: 10.13328/j.cnki.jos.004914]

[45] 马新建.基于沙箱的恶意代码智能分析技术研究[博士学位论文].北京:中国科学院大学,2017.

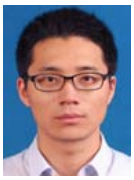
[62] 陈志锋,李清宝,张平,丁文博.基于数据特征的内核恶意软件检测.软件学报,2016,27(12):3172−3191. http://www.jos.org.cn/1000-9825/4927.htm [doi: 10.13328/j.cnki.jos.004927]

[71] 崔杰.代理类和 VPN 类通信工具的分析与研究[硕士学位论文].北京:北京邮电大学,2012.

[78] 张慧琳,邹维,韩心慧.网页木马机理与防御技术.软件学报,2013,24(4):843−858. http://www.jos.org.cn/1000-9825/4376.htm [doi: 10.3724/SP.J.1001.2013.04376]

[81] 孙浩,李会朋,曾庆凯.基于信息流的整数漏洞插装和验证.软件学报,2013,24(12):2767−2781. http://www.jos.org.cn/1000-9825/4385.htm [doi: 10.3724/SP.J.1001.2013.04385]

刘剑(1976－),男,云南石屏人,博士,副研究员,CCF 高级会员,主要研究领域为软件安全,Web 安全,移动安全,网络攻防.



张源(1987－),男,博士,讲师,主要研究领域为系统软件,系统安全.

苏璞睿(1976－),男,博士,研究员,博士生导师,主要研究领域为网络与系统安全.

朱雪阳(1971－),女,博士,副研究员,CCF 专业会员,主要研究领域为嵌入式系统设计,性能分析与优化,形式化方法.

杨珉(1979－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为网络空间安全,移动系统安全.

林惠民(1947－),男,博士,研究员,博士生导师,CCF 会士,主要研究领域为并发理论,进程代数,模型检测,形式化方法.

和亮(1985－),男,博士,副研究员,主要研究领域为网络与系统安全,软件漏洞分析.