ISCAS-SKLCS-16-07 **December, 2016**

中国科学院软件研究所
计算机科学国家重点实验室
技术报告

软件与网络安全研究综述

by

刘剑，苏璞睿，杨珉，和亮，张源，朱雪阳，林惠民

State key Laboratory of Computer Science
Institute of Software
Chinese Academy of Sciences
Beijing 100190. China

软件与网络安全研究综述*

刘剑[1],苏璞睿[2],杨珉[3],和亮[2],张源[3],朱雪阳[4],林惠民[4]

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

[2](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)

[3](复旦大学软件学院,上海, 201203)

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

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、网络窃密等问题频频发生，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。本文从恶意软件、软件漏洞和软件安全机制三个方面综述国内外研究现状，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

Software and Cyber Security - A Survey

LIU Jian[1], SU Purui[2], YANG Min[3], HE Liang[2], ZHANG Yuan[3], ZHU Xue-Yang[4], LIN Huimin[4]

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

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

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

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

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. In this paper, we review the state of the art of malware, software vulnerabilities and software security mechanism, and then analyze 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、网络窃密等问题频频发生，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 6,6186 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

* 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.

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]。

随着互联网正进入移动互联阶段，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；出现集中式的软件分发渠道，可以将软件快速传播到大量的用户，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，涵盖平台、软件和数据诸多维度，对国家网络空间安全边界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，这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。本报告将在接下来的三节里综述这三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、跟踪软件等等。随着互联网的广泛普及和应用，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，恶意软件的发展与硬件平台的发展息息相关，大致可分为三个阶段。第一阶段，单机传播。在上世纪 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。第二阶段，网络传播。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]。这一变化，带来的结果就是恶意软件的传播能力大大增强。第三阶段，协同攻击。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 月在我国发生的 XcodeGhost 恶意软件，具有窃取用户隐私信息和远程控制的安全风险。

以下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方面的最新研究进展。

1.1 恶意软件攻击

传统软件平台上恶意软件。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[8][9]。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。

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、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]。以 Android 系统为例，该系统上的恶意软件呈现出一些特有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][18]、中间人攻击[19][20]等为主的攻击模式。在传播方式方面，Android 生态系统中出现了以重打包（Piggy-packing）[21]、第三方库（Third-party library）[22][23]、恶意编译工具、刷机等新的传播模式。上述 Android 恶意代码的发展主要呈现三方面的特点：(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和移动平台体系架构及编程语言等相关。例如，Android 系统采用四层架构的体系；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。

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[24]。例如：针对无线传感网存在虚假路由信息、Sinkhole 攻击、Wormholes 攻击等[25]；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[26]；而在物联网应用层，通常会收集和处理用户大量隐私数据，例如，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。

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 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[27]。这类攻击消耗 P2P 网络的资源，削弱网络的冗余性。在云计算平台中，出现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）[28][29][30][31]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。在 SDN/NFV 方面，也出现针对控制器、开放式编程接口等核心组件或薄弱的环节的攻击方式[32][33]。同时，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，有效降低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）[34]。此外，与传统 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[35]。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。

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，恶意软件已被广泛地作为一种牟利工具，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，以对其它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[36]。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[37][38]。

1.2 恶意软件对抗

为了达到获取信息或破坏的目的，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。

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，例如，对基于导入表函数特征进行分类的算法造成误导。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。

移动智能平台上恶意软件对抗。在移动智能平台上，一方面由于 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[39]。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 Dash-O、Proguard 和 Dexguard 等自动混淆和加固的工具[40][41]。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，例如：DroidChameleon[42]对多种混淆方法进行分析，并在商业分析平台上进行实验检测，发现很多现有的基于特征匹配技术 Android 恶意软件检测工具不能对抗特定的混淆、加固方法。

恶意软件利用新网络技术对抗。恶意软件利用新的网络机制进行对抗和增强。例如，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

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三种。

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[43]。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 CWSandbox[44]、TTAnalyze[45]和 Ether [46]等动静态融合的沙箱检测机制。同时出现了集成多个杀毒软件和扫描引擎的网站，提供在线可疑软件扫描服务，如 VirusTotal 和 VirSCAN 等。

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层面，一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；另一个是研究如何高效精确地对样本进行聚类与分类，进而检测[47][48][49][50]。综合来看，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[51][52][53][54]；使用的机器学习算法有：贝叶斯、SVM、决策树和深度学习等[55][56][57][58]。

3) 研究高度定向的分析手段，以提高分析的精确度和效率。例如：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[59][60][61]，提出了 REANIMATOR[62]、HASTEN[63]、TaintDroid[64]等分析方法或工具。

2 软件漏洞

软件漏洞是信息系统安全的主要威胁。软件已变得越来越复杂，虽然各大软件厂商在不断改进和完善软件开发质量管理，软件漏洞问题仍无法彻底消除。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，给软件生态带来了严重危害。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。

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[65]。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。因此，为了更好的应对现实需求，目前自动化、智能化的软件漏洞发现与利用逐渐成为软件安全行业研究攻关的热点问题。

软件漏洞相关方法的研究按照漏洞产生和发展的过程可分为三个阶段，即软件漏洞挖掘、软件漏洞分析和软件漏洞利用生成。下面围绕这三个方面分别介绍目前国内外研究的相关进展。

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。自上世纪七十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。

基于补丁对比的漏洞挖掘。基于补丁对比的漏洞挖掘是指比较分析原始程序和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。由于补丁发布与部署之间存在较长时间窗口，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 D.Brumley 团队提出的 APEG（Automatic Pathc-Based Exploit Generation）方案[66]，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。

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实施，能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[67]、SAGE[68]以及商用版本 SpringField 等）等四个主要方向，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。

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。该方法的优点在于具有较强针对性，针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。这方面工作中，D.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[69]，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。F.Yamaguchi 等人提出的基于敏感数据检测缺失的漏洞挖掘方法[70]，对用户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。

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原因及影响要素等内容，并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化的软件漏洞分析方法。

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。其中污点传播分析方法是将程序外部输入数据“绑定”污点标签方式以追踪污点数据的处理，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[71]、实现比特级细粒度污点传播传播的 DECAF 系统[72]、具备较高分析效率的 libdft 系统[73]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，D.Song（DTA++)[74]和 Clause（Dytan）[75]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；在污点分析方法应用于软件漏洞研究方面，W.Cui（RETracer）[76]和 Xing（CREDAL）[77]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[78]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。

软件漏洞判定。软件脆弱性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脆弱性类型（或称漏洞类型）进行匹配识别。当前软件脆弱性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，D.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[79]；W.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[64]。

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。目前脆弱点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，漏洞形成信息部分或永久性缺失。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脆弱性定位方法。R.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[80]；微软公司的 W.Cui 等人研制的 RETracer 系统[76]，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 J.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[77]。

2.3 软件漏洞利用

软件漏洞利用，指的是利用软件内部缺陷，以实现在该软件的正常运行过程中无法达到的目的。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趋势下，人工分析处理方式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。

面向控制流的利用方法。面向控制流的自动利用方法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。目前该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[81]，综合利用在线式符号执行和离线式符号执行，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。M.H.Wang 等人提出的多样性利用样本自动生成方法 PolyAEG[82]，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。

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。该方面的代表性工作是 Liang 团队的 FlowStitch[83]，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 DOP[84]，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，研究者通过设计多种安全机制，提高整个软件体系应对安全攻击的能力。主要技术思路包括三个方面：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，提高用户对软件的安全管理能力。下面分别介绍这三方面工作的研究现状。

3.1 软件行为管控机制

基于系统的软件行为管控机制。现代移动操作系统普遍地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[64]和 AppFence 系统[85]。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[86]、LP-Guardian 系统[87]和 MockDroid 系统[88]。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。例如，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[89]、AdSplit 系统[90]和 AFrame 系统[91]将广告插件置于单独的系统服务中运行，隔离广告插件对宿主引用的影响。

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[92]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[93]利用 Intel 的 EPT（Extended Page Tables）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[94]，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。为保证软件的可靠性，应该尽量减少内核的规模。Zhou 等人提出一种称为 Wimpy Kernel 的内核架构[95]，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[96]基于 Xen Hypervisor 的 Service Domain 功能将内核划分为不同区域。

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，TCB）来提供基于硬件的安全管控机制。Azab 等人[97]基于 ARM TrustZone 的安全模式（Secure World）设计可信语言运行时（Trusted Language Runtime，TLR）框架来保护系统内核的机制。Wenhao 等人[98]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，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[99]提出了一种基于 ARM 内存域（Memory Domain）特性的软件错误隔离（Software Fault Isolation，SFI）机制，可以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，而这为内存泄漏攻击提供了有利条件。中科院信工所的研究人员[100]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。

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，研究者也提出一些基于软件重写的行为管控方案。Aurasium 系统[101]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[102]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[103]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。

基于规则的软件行为管控机制。为了提供灵活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[104]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。Sven Bugiel 等人设计的 FlaskDroid 系统[105]，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，Stephan Heuser 等人提出的 ASM 框架[106]，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，Michael Backes 等人提出了一个可扩展的 Android 安全框架 ASF[107]，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[108]，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。为克服此问题，Ruowen Wang 等人提出 EASEAndroid 系统[109]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

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，但是无法防御未知漏洞，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[110]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）[111]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）[112]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有一定缺陷，攻击者能够利用 ROP（Retrun-oriented Programmming）[113]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）[114]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[115]，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。

Web 软件的安全机制。不同于传统操作系统，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无权访问的资源。Maliheh Monshizadeh 等人提出的 MACE 工具[116]，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。Giancarlo Pellegrino 等人[117]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。Michael Weissbacher 等人[118]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[118]，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。Adam 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做区分引起的。基于此观点，他们提出并实现了 deDacota 系统[119]，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。

移动软件的安全机制。得益于移动互联的发展，移动平台在过去十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了开发效率的同时，引入了组件通信这一新型攻击面。Michael Dietz 提出的 QUIRE 系统[120]用于在程序进行 IPC（Inter-Process Call）调用时，维护调用路径上的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，研究者也采用动静态分析进行检测，代表性工作包括工具 DroidChecker[121]、CHEX[122]和 IntentFuzzer[123]。上述工作着眼于漏洞的防御和检测，AppSealer 系统[124]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[125]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的分析软件中存在隐私泄露问题。针对组件之间的数据流依赖，工具 Amandroid[126]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[127]和 StubDroid[128]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、误报率高的特点，研究者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[129]，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[130]和 SUPOR[131]基于自然语言处理的方法去自动化地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。研究者发现，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[132]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。

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[133]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 MRT (Minimum Run Time) 技术[134]，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它虚拟 CPU 抢占。由于侧信道攻击种类繁多，原理各异，Nomad 系统[135]提出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。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[136]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。

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。Jinjin Liang 等人[137]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 Jianjun 等人[138]发现一种可导致 CDN 资源耗尽的循环转发攻击。根据他们的实验结果，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[139]；2012 年，Nikos 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[140]。同时，只有正确的使用 TLS 协议才能保障通信安全。2012 年，Martin Georgiev 等人发现很多非浏览器应用未能正确使用 TLS 进行连接[141]；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[142]。TLS 的保护策略很大程度上依赖于对 X.509 证书的验证。2014 年，Chad 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[143]。Boyuan He 等人于 2015 年 Oakland 大会上展示的 SSLINT 工具基于静态分析技术[144]，通过对数据依赖图的挖掘，可有效检测软件对 TLS API 的误用。

3.3 面向终端用户的软件安全机制

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[145]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[146]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[147]采取了另一种思路研究软件描述和实际行为是否匹配。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是否能够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。

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[148]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[149]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[150]用于识别隐私搜集行为过程中的用户感知度。DESCRIBEME 工具[151]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。

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：一种是安装时权限授予，一种是使用时权限授予。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。FranziskaRoesner 等人[152]提出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，FranziskaRoesner 等人在 USENIX Security 大会展示的 LayerCake 系统[153]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，Talia Ringer 等人[154]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。

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大量敏感数据，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（CGC，Cyber Grand Challenge）”，旨在实验性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离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，是当前恶意软件中的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。

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！

References:

[1] CNCERT 互联网安全威胁报告-2016 年 6 月. 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. 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. http://www.cert.org.cn/publish/main/12/2016/20161201134333495740421/20161201134333495740421_.html

[4] Skoudis E, Zeltser L. Malware: Fighting malicious code[M]. Prentice Hall Professional, 2004.

[5] Sikorski M, Honig A. Practical malware analysis: the hands-on guide to dissecting malicious software[M]. no starch press, 2012..

[6] Weaver N, Paxson V, Staniford S, et al. A taxonomy of computer worms[C]//Proceedings of the 2003 ACM workshop on Rapid malcode. ACM, 2003: 11-18.

[7] Dittrich D, Dietrich S. P2P as botnet command and control: a deeper insight[C]//Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on. IEEE, 2008: 41-48.

[8] Xu Z, Zhang J, Gu G, et al. GOLDENEYE: Efficiently and Effectively Unveiling Malware’s Targeted Environment[C]//International Workshop on Recent Advances in Intrusion Detection. Springer International Publishing, 2014: 22-45.

[9] Song C, Royal P, Lee W. Impeding Automated Malware Analysis with Environment-sensitive Malware[C]//HotSec. 2012.

[10] Castillo C A. Android malware past, present, and future[J]. White Paper of McAfee Mobile Security Working Group, 2011, 1: 16.

[11] Zhou Y, Jiang X. Dissecting android malware: Characterization and evolution[C]//2012 IEEE Symposium on Security and Privacy. IEEE, 2012: 95-109.

[12] 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, http://www.csc.ncsu.edu/faculty/jiang/DroidKungFu3/

[14] Fang Z, Han W, Li Y. Permission based android security: Issues and countermeasures[J]. computers & security, 2014, 43: 205-218..

[15] Bugiel S, Davi L, Dmitrienko A, et al. Towards Taming Privilege-Escalation Attacks on Android[C]//NDSS. 2012, 17: 19.

[16] Marforio C, Francillon A, Capkun S, et al. Application collusion attack on the permission-based security model and its implications for modern smartphone systems[M]. Zürich, Switzerland: Department of Computer Science, ETH Zurich, 2011.

[17] Schlegel R, Zhang K, Zhou X, et al. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones[C]//NDSS. 2011, 11: 17-33.

[18] Chin E, Felt A P, Greenwood K, et al. Analyzing inter-application communication in Android[C]//Proceedings of the 9th international conference on Mobile systems, applications, and services. ACM, 2011: 239-252.

[19] Egele M, Brumley D, Fratantonio Y, et al. An empirical study of cryptographic misuse in android applications[C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013: 73-84.

[20] Sounthiraraj D, Sahs J, Greenwood G, et al. Smv-hunter: Large scale, automated detection of ssl/tls man-in-the-middle vulnerabilities in android apps[C]//In Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS'14). 2014.

[21] Zhou W, Zhou Y, Grace M, et al. Fast, scalable detection of piggybacked mobile applications[C]//Proceedings of the third ACM conference on Data and application security and privacy. ACM, 2013: 185-196.

[22] K. Chen, X. Wang, Y. Chen, P. Wang, Y. Lee, X. Wang, B. Ma, A. Wang, Y. Zhang, and W. Zhou, "Following devils footprints: Crossplatform analysis of potentially harmful libraries on Android and iOS," in Proceedings of the 37th IEEE Symposium on Security and Privacy,ser. S&P '16, 2016.

[23] Backes M, Bugiel S, Derr E. Reliable Third-Party Library Detection in Android and its Security Applications[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016: 356-367.

[24] Jing Q, Vasilakos A V, Wan J, et al. Security of the internet of things: Perspectives and challenges[J]. Wireless Networks, 2014, 20(8): 2481-2501.

[25] Cao Z, Hu J, Chen Z, et al. Feedback: Towards Dynamic Behavior and Secure Routing forWireless Sensor Networks[C]//Proceedings of the 20th International Conference on Advanced Information Networking and Applic ations-Volume 02. IEEE Computer Society, 2006: 160-164.

[26] Jhaveri R H, Patel S J, Jinwala D C. DoS attacks in mobile ad hoc networks: A survey[C]//2012 Second International Conference on Advanced Computing & Communication Technologies. IEEE, 2012: 535-541.

[27] Douceur J R. The sybil attack[C]//International Workshop on Peer-to-Peer Systems. Springer Berlin Heidelberg, 2002: 251-260.

[28] Hlavacs H, Treutner T, Gelas J P, et al. Energy consumption side-channel attack at virtual machines in a cloud[C]//Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on. IEEE, 2011: 605-612.

[29] Wu Z, Xu Z, Wang H. Whispers in the hyper-space: high-speed covert channel attacks in the cloud[C]//Presented as part of the 21st USENIX Security Symposium (USENIX Security 12). 2012: 159-173.

[30] Liu F, Yarom Y, Ge Q, et al. Last-level cache side-channel attacks are practical[C]//IEEE Symposium on Security and Privacy. 2015: 605-622.

[31] Rocha F, Correia M. Lucy in the sky without diamonds: Stealing confidential data in the cloud[C]//2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W). IEEE, 2011: 129-134..

[32] Hong S, Xu L, Wang H, et al. Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures[C]//NDSS. 2015.

[33] Dhawan M, Poddar R, Mahajan K, et al. SPHINX: Detecting Security Attacks in Software-Defined Networks[C]//NDSS. 2015

[34] Slopek A, Vlajic N. Economic denial of sustainability (EDoS) attack in the cloud using web-bugs[C]//Proc. of the 17th Int'l Symp. on Research in Attacks, Intrusions, and Defenses (RAID 2014). Switzerland: Springer Int'l Publishing. 2014: 469-471.

[35] Zhao S, Lee P P C, Lui J, et al. Cloud-based push-styled mobile botnets: a case study of exploiting the cloud to device messaging service[C]//Proceedings of the 28th Annual Computer Security Applications Conference. ACM, 2012: 119-128.

[36] Tankard C. Advanced persistent threats and how to monitor and deter them[J]. Network security, 2011, 2011(8): 16-19.

[37] Li F, Lai A, Ddl D. Evidence of Advanced Persistent Threat: A case study of malware for political espionage[C]//Malicious and Unwanted Software (MALWARE), 2011 6th International Conference on. IEEE, 2011: 102-109.

[38] Juels A, Yen T F. Sherlock Holmes and the case of the advanced persistent threat[C]//Presented as part of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats. 2012.

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

[40] Kovacheva A. Efficient code obfuscation for Android[C]//International Conference on Advances in Information Technology. Springer International Publishing, 2013: 104-119.

[41] Faruki P, Bharmal A, Laxmi V, et al. Evaluation of android anti-malware techniques against dalvik bytecode obfuscation[C]//2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, 2014: 414-421.

[42] Rastogi V, Chen Y，Jiang X. Droidchameleon: Evaluating Android anti-malware against transformation attacks. In: Proc. ofthe 8th ACMSIGSAC Symp．on Information，Computer and Communications Security(ASIACCS 2013), 2013, 329-334.

[43] Egele M, Scholte T, Kirda E, et al. A survey on automated dynamic malware-analysis techniques and tools[J]. ACM Computing Surveys (CSUR), 2012, 44(2): 6.

[44] Willems C, Holz T, Freiling F. Toward automated dynamic malware analysis using cwsandbox[J]. IEEE Security and Privacy, 2007, 5(2): 32-39.

[45] Bayer U, Moser A, Kruegel C, et al. Dynamic analysis of malicious code[J]. Journal in Computer Virology, 2006, 2(1): 67-77.

[46] Dinaburg A, Royal P, Sharif M, et al. Ether: malware analysis via hardware virtualization extensions[C]//Proceedings of the 15th ACM conference on Computer and communications security. ACM, 2008: 51-62.

[47] Christodorescu M, Jha S, Seshia S A, et al. Semantics-aware malware detection[C]//2005 IEEE Symposium on Security and Privacy (S&P'05). IEEE, 2005: 32-46.

[48] Kirda E, Kruegel C, Banks G, et al. Behavior-based Spyware Detection[C]//Usenix Security. 2006, 6.

[49] Fuchs A P, Chaudhuri A, Foster J S. Scandroid: Automated security certification of android[J]. 2009.

[50] Chan P P F, Hui L C K, Yiu S M. Droidchecker: analyzing android applications for capability leak[C]//Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. ACM, 2012: 125-136.

[51] Park Y, Reeves D S, Stamp M. Deriving common malware behavior through graph clustering[J]. Computers & Security, 2013, 39: 419-430.

[52] Liang Z, Yin H, Song D. HookFinder: Identifying and understanding malware hooking behaviors[J]. Department of Electrical and Computing Engineering, 2008: 41.

[53] Wu D J, Mao C H, Wei T E, et al. Droidmat: Android malware detection through manifest and api calls tracing[C]//Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on. IEEE, 2012: 62-69.

[54] Burguera I, Zurutuza U, Nadjm-Tehrani S. Crowdroid: behavior-based malware detection system for android[C]//Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. ACM, 2011: 15-26.

[55] Schultz M G, Eskin E, Zadok F, et al. Data mining methods for detection of new malicious executables[C]//Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on. IEEE, 2001: 38-49.

[56] Rieck K, Trinius P, Willems C, et al. Automatic analysis of malware behavior using machine learning[J]. Journal of Computer Security, 2011, 19(4): 639-668.

[57] Amos B, Turner H, White J. Applying machine learning classifiers to dynamic android malware detection at scale[C]//2013 9th international wireless communications and mobile computing conference (IWCMC). IEEE, 2013: 1666-1671.

[58] Sadeghi A, Bagheri H, Garcia J. A Taxonomy and Qualitative Comparison of Program Analysis Techniques for Security Assessment of Android Software[J]. IEEE Transactions on Software Engineering, 2016.

[59] Wilhelm J, Chiueh T. A forced sampled execution approach to kernel rootkit identification[C]//International Workshop on Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2007: 219-235.

[60] Moser A, Kruegel C, Kirda E. Exploring multiple execution paths for malware analysis[C]//2007 IEEE Symposium on Security and Privacy (SP'07). IEEE, 2007: 231-245.

[61] Cadar C, Ganesh V, Pawlowski P M, et al. EXE: automatically generating inputs of death[J]. ACM Transactions on Information and System Security (TISSEC), 2008, 12(2): 10.

[62] Comparetti P M, Salvaneschi G, Kirda E, et al. Identifying dormant functionality in malware programs[C]//2010 IEEE Symposium on Security and Privacy. IEEE, 2010: 61-76..

[63] Kolbitsch C, Kirda E, Kruegel C. The power of procrastination: detection and mitigation of execution-stalling malicious code[C]//Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011: 285-296.

[64] Enck W, Gilbert P, Han S, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones[J]. ACM Transactions on Computer Systems (TOCS), 2014, 32(2): 5.

[65] Common Vulnerabilities and Exposures. https://cve.mitre.org.

[66] D. Brumley, P. Poosankam, D. Song, et al. Automatic patch-based exploit generation is possible: Techniques and implications[C] //2008 IEEE Symposium on Security and Privacy (sp 2008). IEEE, 2008: 143-157.

[67] C. Cadar, D. Dunbar and D. Engler. KLEE: Unassisted and Automatic Generation of High Coverage Tests for Complex Systems Programs. In OSDI 2008.

[68] P. Godefroid, M. Y. Levin, D. Molnar. Automated Whitebox Fuzz Testing. In NDSS 2008.

[69] D. Engler, D. Y. Chen, S. Hallem, A. Chou, and B. Chelf. Bugsas deviant behavior: A general approach to inferring errors in systems code. In Proc. of ACM Symposium on Operating Systems Principles (SOSP), pages 57–72, 2001.

[70] F. Yamaguchi, C. Wressnegger, H. Gascon, and K. Rieck. Chucky: Exposing missing checks in source code for vulnerability discovery. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pages 499-510, 2013.

[71] H. Yin andD. Song. Temu: Binary code analysis via whole-system layered annotative execution[J]. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-3, 2010.

[72] A. Henderson, A. Prakash, etc. Make It Work, Make It Right, Make It Fast: Building a Platform-Neutral Whole-System Dynamic Binary Analysis Platform. In ISSTA 2014.

[73] V. P. Kemerlis, G. Portokalidis, etc. libdft: Practical Dynmaic Data Flow Tracking for Commodity Systems. In VEE 2012.

[74] M. G. Kang, S. Mccamant, P. Poosankam, et al. DTA++: Dynamic Taint Analysis with Targeted Control-Flow Propagation.[C]// Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, Usa, February -, February. 2011.

[75] J. Clause, W. Li, and A. Orso. Dytan: a generic dynamic taint analysis framework[C]// Acm/sigsoft International Symposium on Software Testing and Analysis, ISSTA 2007, London, Uk, July. 2007:196—206.

[76] W. Cui, M. Peinado, S. K. Cha, Y. Fratantonio, and V. P. Kemerlis. Retracer: Triaging crashes by reverse execution from partial memory dumps. In Proceedings of the 38th International Conference on Software Engineering, 2016.

[77] J. Xu, D. Mu, P. Chen, X. Xing, P. Wang and P. Liu. CREDAL: Towards Locating a Memory Corruption Vulnerability with Your Core Dump. In CCS 2016.

[78] S. S. Douskos, E. Lahtinen, etc. Targeted Automatic Interger Overflow Discovery Using Goal-Directed Conditional Branch Enforcement. In ASPLOS 2015.

[79] J. Newsome andD. Song. Dynamic taint analysis: Automatic detection, analysis, and signature generation of exploit attacks on commodity software[C]//In In Proceedings of the 12th Network and Distributed Systems Security Symposium. 2005.

[80] R. Wu, H. Zhang, S.-C. Cheung, and S. Kim. Crashlocator: Locating crashing faults based on crash stacks. In Proceedings of the 2014 International Symposium on Software Testing and Analysis, 2014.

[81] S. K. Cha, T. Avgerinos, A. Rebert, et al. Unleashing mayhem on binary code[C]//2012 IEEE Symposium on Security and Privacy. IEEE, 2012: 380-394.

[82] M. H. Wang, P. Su, Q. Li, et al. Automatic polymorphic exploit generation for software vulnerabilities[C]//International Conference on Security and Privacy in Communication Systems. Springer International Publishing, 2013: 216-233.

[83] H. Hu, Z. L. Chua, S. Adrian, et al. Automatic generation of data-oriented exploits[C]//24th USENIX Security Symposium (USENIX Security 15). 2015: 177-192.

[84] H. Hu, S. Shinde, et al. Data-Oriented Programming: On the Expresivenes of Non-Control Data Attacks. In S&P 2016.

[85] Hornyack, Peter and Han, Seungyeop and Jung, Jaeyeon and Schechter, Stuart and Wetherall, David. These Are not the Droids You'Re Looking for: Retrofitting Android to Protect Data from Imperious Applications[C]. //Proceedings of the 18th ACM Conference on Computer and Communications Security. ACM, 2011.

[86] Zhou Y, Zhang X, Jiang X, et al. Taming information-stealing smartphone applications (on android)[C]//International conference on Trust and trustworthy computing. Springer Berlin Heidelberg, 2011: 93-107.

[87] Fawaz K, Shin K G. Location privacy protection for smartphone users[C]//Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014: 239-250.

[88] Beresford A R, Rice A, Skehin N, et al. Mockdroid: trading privacy for application functionality on smartphones[C]//Proceedings of the 12th workshop on mobile computing systems and applications. ACM, 2011: 49-54.

[89] Pearce P, Felt A P, Nunez G, et al. Addroid: Privilege separation for applications and advertisers in android[C]//Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security. ACM, 2012: 71-72.

[90] Shekhar S, Dietz M, Wallach D S. Adsplit: Separating smartphone advertising from applications[C]//Presented as part of the 21st USENIX Security Symposium (USENIX Security 12). 2012: 553-567.

[91] Zhang X, Ahlawat A, Du W. AFrame: isolating advertisements from mobile applications in Android[C]//Proceedings of the 29th Annual Computer Security Applications Conference. ACM, 2013: 9-18.

[92] Wu C, Zhou Y, Patel K, et al. AirBag: Boosting Smartphone Resistance to Malware Infection[C]//NDSS. 2014.

[93] Liu Y, Zhou T, Chen K, et al. Thwarting memory disclosure with efficient hypervisor-enforced intra-domain isolation[C]//Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 2015: 1607-1619.

[94] Kurmus A, Zippel R. A tale of two kernels: Towards ending kernel hardening wars with split kernel[C]//Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014: 1366-1377.

[95] Zhou Z, Yu M, Gligor V D. Dancing with giants: wimpy kernels for on-demand isolated I/O[C]//2014 IEEE Symposium on Security and Privacy. IEEE, 2014: 308-323.

[96] Nikolaev R, Back G. VirtuOS: an operating system with kernel virtualization[C]//Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles. ACM, 2013: 116-132.

[97] Azab A M, Ning P, Shah J, et al. Hypervision across worlds: Real-time kernel protection from the arm trustzone secure world[C]//Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014: 90-102.

[98] Li W, Li H, Chen H, et al. Adattester: Secure online mobile advertisement attestation using trustzone[C]//Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services. ACM, 2015: 75-88.

[99] Zhou Y, Wang X, Chen Y, et al. Armlock: Hardware-based fault isolation for arm[C]//Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014: 558-569.

[100] Guan L, Lin J, Luo B, et al. Protecting private keys against memory disclosure attacks using hardware transactional memory[C]//2015 IEEE Symposium on Security and Privacy. IEEE, 2015: 3-19.

[101] Xu R, Saïdi H, Anderson R. Aurasium: Practical policy enforcement for android applications[C]//Presented as part of the 21st USENIX Security Symposium (USENIX Security 12). 2012: 539-552.

[102] Rastogi V, Qu Z, McClurg J, et al. Uranine: Real-time Privacy Leakage Monitoring without System Modification for Android[C]//International Conference on Security and Privacy in Communication Systems. Springer International Publishing, 2015: 256-276.

[103] Backes M, Bugiel S, Hammer C, et al. Boxify: Full-fledged app sandboxing for stock Android[C]//24th USENIX Security Symposium (USENIX Security 15). 2015: 691-706.

[104] Smalley S, Craig R. Security Enhanced (SE) Android: Bringing Flexible MAC to Android[C]//NDSS. 2013, 310: 20-38.

[105] Bugiel S, Heuser S, Sadeghi A R. Flexible and fine-grained mandatory access control on Android for diverse security and privacy policies[C]//Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13). 2013: 131-146.

[106] Heuser S, Nadkarni A, Enck W, et al. Asm: A programmable interface for extending android security[C]//23rd USENIX Security Symposium (USENIX Security 14). 2014: 1005-1019.

[107] Backes M, Bugiel S, Gerling S, et al. Android Security Framework: Extensible multi-layered access control on Android[C]//Proceedings of the 30th annual computer security applications conference. ACM, 2014: 46-55.

[108] Zhang Y, Yang M, Gu G, et al. Rethinking Permission Enforcement Mechanism on Mobile Systems[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(10): 2227-2240.

[109] Wang R, Enck W, Reeves D, et al. EASEAndroid: automatic policy analysis and refinement for security enhanced android via large-scale semi-supervised learning[C]//24th USENIX Security Symposium (USENIX Security 15). 2015: 351-366.

[110] Cowan C, Pu C, Maier D, et al. StackGuard: automatic adaptive detection and prevention of buffer-overflow attacks[C]// Conference on Usenix Security Symposium. USENIX Association, 1998:63--78.

[111] Trev N. Data Execution Prevention[J]. Lect Publishing, 2011.

[112] Miller F P, Vandome A F, Mcbrewster J. Address space layout randomization[M]. Alphascript Publishing, 2010.

[113] Prandini M, Ramilli M. Return-Oriented Programming[J]. IEEE Security & Privacy Magazine, 2012, 10(6):84-87.

[114] M. Abadi, M. Budiu, U. Erlingsson and J. Ligatti. Control-flow integrity. In Proceedings of the 12th ACM conference on Computer and communications security, (Alexandria, VA, USA, 2005), ACM, 340-353.

[115] Meining Nie, Purui Su, Qi Li, Zhi Wang, Lingyun Ying, Jinlong Hu, Dengguo Feng. Xede: Practical Exploit Early Detection. The 18th International Symposium on Research in Attacks, Intrusions and Defenses. RAID 2015, Springer, 2015: 198-221.

[116] Monshizadeh M, Naldurg P, Venkatakrishnan V N. Mace: Detecting privilege escalation vulnerabilities in web applications[C]//Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014: 690-701.

[117] Pellegrino G, Balzarotti D. Toward Black-Box Detection of Logic Flaws in Web Applications[C]//NDSS. 2014.

[118] Weissbacher M, Robertson W, Kirda E, et al. Zigzag: Automatically hardening web applications against client-side validation vulnerabilities[C]//24th USENIX Security Symposium (USENIX Security 15). 2015: 737-752.

[119] Doupé A, Cui W, Jakubowski M H, et al. deDacota: toward preventing server-side XSS via automatic code and data separation[C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013: 1205-1216.

[120] Dietz M, Shekhar S, Pisetsky Y, et al. QUIRE: Lightweight Provenance for Smart Phone Operating Systems[C]//USENIX Security Symposium. 2011, 31.

[121] Chan P P F, Hui L C K, Yiu S M. Droidchecker: analyzing android applications for capability leak[C]//Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. ACM, 2012: 125-136.

[122] Lu L, Li Z, Wu Z, et al. Chex: statically vetting android apps for component hijacking vulnerabilities[C]//Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012: 229-240.

[123] Yang K, Zhuge J, Wang Y, et al. IntentFuzzer: detecting capability leaks of android applications[C]//Proceedings of the 9th ACM symposium on Information, computer and communications security. ACM, 2014: 531-536.

[124] Zhang M, Yin H. AppSealer: Automatic Generation of Vulnerability-Specific Patches for Preventing Component Hijacking Attacks in Android Applications[C]//NDSS. 2014.

[125] Arzt S, Rasthofer S, Fritz C, et al. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps[J]. ACM SIGPLAN Notices, 2014, 49(6): 259-269.

[126] Wei F, Roy S, Ou X. Amandroid: A precise and general inter-component data flow analysis framework for security vetting of android apps[C]//Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014: 1329-1341.

[127] Cao Y, Fratantonio Y, Bianchi A, et al. EdgeMiner: Automatically Detecting Implicit Control Flow Transitions through the Android Framework[C]//NDSS. 2015.

[128] Arzt S, Bodden E. StubDroid: automatic inference of precise data-flow summaries for the android framework[C]// International Conference on Software Engineering. ACM, 2016.

[129] Zhang Y, Yang M, Xu B, et al. Vetting undesirable behaviors in android apps with permission use analysis[C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013: 611-622.

[130] Nan Y, Yang M, Yang Z, et al. Uipicker: User-input privacy identification in mobile applications[C]//24th USENIX Security Symposium (USENIX Security 15). 2015: 993-1008.

[131] Huang J, Li Z, Xiao X, et al. Supor: Precise and scalable sensitive user input detection for android apps[C]//24th USENIX Security Symposium (USENIX Security 15). 2015: 977-992.

[132] Zhang N, Yuan K, Naveed M, et al. Leave me alone: App-level protection against runtime information gathering on Android[C]//2015 IEEE Symposium on Security and Privacy. IEEE, 2015: 915-930.

[133] Zhang Y, Reiter M K. Düppel: retrofitting commodity operating systems to mitigate cache side channels in the cloud[C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013: 827-838.

[134] Varadarajan V, Ristenpart T, Swift M. Scheduler-based defenses against cross-vm side-channels[C]//23rd USENIX Security Symposium (USENIX Security 14). 2014: 687-702.

[135] Moon S J, Sekar V, Reiter M K. Nomad: Mitigating arbitrary cloud side channels via provider-assisted migration[C]//Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 2015: 1595-1606.

[136] Pattuk E, Kantarcioglu M, Lin Z, et al. Preventing cryptographic key leakage in cloud virtual machines[C]//23rd USENIX Security Symposium (USENIX Security 14). 2014: 703-718.

[137] Liang J, Jiang J, Duan H, et al. When HTTPS meets CDN: A case of authentication in delegated service[C]//2014 IEEE Symposium on Security and Privacy. IEEE, 2014: 67-82.

[138] Chen J, Jiang J, Zheng X, et al. Forwarding-Loop Attacks in Content Delivery Networks[J]. 2010-12-10]. http://netsec. ccert. edu. cn/duanhx/files/2010/12/cdn_loop-final-camera-ready. pdf, 2016.

[139] Ray M, Dispensa S. Renegotiating TLS[J]. 2009.

[140] Mavrogiannopoulos N, Vercauteren F, Velichkov V, et al. A cross-protocol attack on the TLS protocol[C]//Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012: 62-72.

[141] Georgiev M, Iyengar S, Jana S, et al. The most dangerous code in the world: validating SSL certificates in non-browser software[C]//Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012: 38-49.

[142] Fahl S, Harbach M, Muders T, et al. Why Eve and Mallory love Android: An analysis of Android SSL (in) security[C]//Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012: 50-61.

[143] Brubaker C, Jana S, Ray B, et al. Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations[C]//2014 IEEE Symposium on Security and Privacy. IEEE, 2014: 114-129.

[144] He B, Rastogi V, Cao Y, et al. Vetting SSL usage in applications with SSLint[C]//2015 IEEE Symposium on Security and Privacy. IEEE, 2015: 519-534.

[145] Pandita R, Xiao X, Yang W, et al. Whyper: Towards automating risk assessment of mobile applications[C]//Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13). 2013: 527-542.

[146] Qu Z, Rastogi V, Zhang X, et al. Autocog: Measuring the description-to-permission fidelity in android applications[C]//Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014: 1354-1365.

[147] Gorla A, Tavecchia I, Gross F, et al. Checking app behavior against app descriptions[C]//Proceedings of the 36th International Conference on Software Engineering. ACM, 2014: 1025-1035.

[148] Huang J, Zhang X, Tan L, et al. AsDroid: detecting stealthy behaviors in Android applications by user interface and program behavior contradiction[C]//Proceedings of the 36th International Conference on Software Engineering. ACM, 2014: 1036-1046., ICSE 2014.

[149] Yang W, Xiao X, Andow B, et al. Appcontext: Differentiating malicious and benign mobile app behaviors using context[C]//2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. IEEE, 2015, 1: 303-313.

[150] Yang Z, Yang M, Zhang Y, et al. Appintent: Analyzing sensitive data transmission in android for privacy leakage detection[C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013: 1043-1054.

[151] Zhang M, Duan Y, Feng Q, et al. Towards automatic generation of security-centric descriptions for Android apps[C]//Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 2015: 518-529.

[152] Roesner F, Kohno T, Moshchuk A, et al. User-driven access control: Rethinking permission granting in modern operating systems[C]//2012 IEEE Symposium on Security and Privacy. IEEE, 2012: 224-238.

[153] Roesner F, Kohno T. Securing embedded user interfaces: Android and beyond[C]//Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13). 2013: 97-112.

[154] Ringer T, Grossman D, Roesner F. AUDACIOUS: User-Driven Access Control with Unmodified Operating Systems[J].