软件供应链安全综述

何熙巽[1], 张玉清[1,2], 刘奇旭[3]

[1] 中国科学院大学 国家计算机网络入侵防范中心 北京 中国 101408
[2] 西安电子科技大学 网络与信息安全学院 西安 中国 710071
[3] 中国科学院信息工程研究所 北京 中国 100093

摘要 随着信息技术产业的发展和软件开发需求的扩展, 软件开发的难度与复杂度不断上升, 针对软件供应链的重大安全事件时有发生。这些事件展现了软件供应链攻击低成本而高效的特点以及软件供应链管理的复杂性, 使得软件供应链的安全问题受到了广泛的关注, 相关领域的研究工作也进入了起步阶段。本文从软件供应链安全的定义以及发展历程入手, 介绍了软件供应链安全问题的相关背景, 并通过对现有研究成果的调研分析, 将软件供应链安全问题分为管理问题和技术问题两个方面, 从这两个方面入手介绍了软件供应链安全的研究现状, 然后结合研究现状总结了软件供应链安全所面临的现实挑战, 并提出了未来可能的研究方向。

关键词 软件供应链; 网络供应链; 网络与信息系统安全; 软件安全; 供应链风险管理
中图法分类号 TP393.0 DOI 号 10.19363/J.cnki.cn10-1380/tn.2020.01.06

Survey of Software Supply Chain Security

HE Xixun[1], Zhang Yuqing[1,2], Liu Qixu[3]

[1] National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, Beijing 101408, China
[2] School of Cyber Engineering, Xidian University, Xi'an 710071, China
[3]Institude of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

Abstract With the development of information technology industry and the expansion of the demand in software development, the difficulty and complexity of software development are rising continuously, and the major events of software supply chain security occur from time to time. These events show the low-cost as well as efficiency of software supply chain attack and the complexity of software supply chain management, which has led to widespread attention on software security issues, and the research in related field has also entered the initial phase. Starting with the definition and development history of software supply chain security, this paper introduces the background of software supply chain security, divides the software supply chain security problem into two aspects of management and technical problems through the survey and analysis of existing researches, and introduces the current status of software supply chain security from these two aspects. Then, based on the current research status, the current challenges faced by software supply chain security are summarized, and the possible future research direction are pointed out.

Key words software supply chain; cyber supply chain; network and information system security; software security; supply chain risk management

1 引言

软件是信息技术的重要组成部分, 其发展历程贯穿了信息技术产业的始终。在这一过程中, 人类社会对软件的依赖逐渐增加, 正在进入“软件无处不在, 软件定义一切、软件使能一切”的时代[1], 这就使得软件安全的重要性随之增长。人们对软件安全的关注不仅限于计算科学的领域, 更延伸到了人类社会的方方面面。为此, 学术界和产业界在长期的研究过程中提出了软件工程[2]以及程序分析[3]等概念、方法和工具, 以保障软件的正常运行。

在软件技术飞速发展, 软件开发的手段不断进步, 软件安全领域的方法和工具逐渐投入应用的同时, 一种新型的安全威胁的出现使得软件产业面临了严重的安全问题。2014 年 4 月, 一个后来被称为心脏出血(HeartBleed)漏洞的 OpenSSL 加密软件包漏

通讯作者: 刘奇旭, 博士, 副研究员, Email: liuqixu@iie.ac.cn。
本课题得到国家重点研发计划基金资助项目(No.2016YFB0800700), 国家自然科学基金资助项目(No.61572460, No.61272481), 信息安全国家重点实验室的开放课题基金资助项目(No.2017-ZD-01), 国家发改委信息安全专项基金资助项目(No.(2012)1424)资助。
收稿日期: 2019-05-30; 修改日期: 2019-09-23; 定稿日期: 2019-12-13

洞被公开披露。一段缺少边界检查, 可能出现缓冲区过读, 导致使用该软件包的进程的内存信息泄漏的代码在 2012 年被引入 OpenSSL, 使得大量使用该软件包构建的基于 TLS 的软件和网络服务受到影响[4]。心脏出血漏洞事件体现了一个软件模块的漏洞在该软件被引入下游其他软件的开发流程后, 所能造成的惊人的影响和潜在的巨大破坏力。

利用这一特性, 一种新型的网络攻击方式得到了许多攻击者的青睐。2013 年 9 月, “棱镜门”事件曝光了美国国家安全局(NSA)利用买通 RSA 公司的机会, 在其开发的软件工具 BSAFE 的伪随机数生成算法中植入了后门, 影响了大量使用该工具开发的软件[5]。2015 年 9 月, 攻击者在非官方渠道的 APP 开发工具 Xcode 中注入 XcodeGhost 病毒进行传播, 感染了所有通过被污染了的开发工具编译出的 APP, 影响了一批拥有数亿用户的知名 APP[6]。这些事件中所使用的网络攻击方式与传统方式相比具有鲜明的隐蔽性强, 影响范围广, 低成本, 高效率的特点, 使得软件供应链这一概念以及相关的安全问题进入了人们的视野。

在软件供应链安全这一概念出现之前, 攻击行为主要针对目标计算机系统的漏洞。在各类安全报告[7]中, 对安全现状的分析基本等同于分析目标计算机系统的漏洞数量以及漏洞级别。针对流行漏洞的攻击需要能够掌握有效的 0-day 漏洞, 并构造有效的攻击向量(Attack vector) [8], 在攻击难度上较高, 在攻击影响上依赖于所掌握的 0-day 漏洞存在的广泛性。近几年来涌现的大量软件供应链安全事件则具有不同的特点——攻击软件供应链与攻击软件本身相比, 在难度和成本上有显著的降低, 在影响范围上一般有显著的扩大, 并且在发生之后由于攻击被供应链上的多次传递所掩盖, 难以被现有的计算机系统安全防范措施识别和处理。软件供应链安全事件这些特点被广泛认识后, 针对软件供应链的攻击会受到攻击者的青睐, 因此可以预见到软件供应链安全事件数量和影响力在未来的进一步扩张。

通过调研 21 世纪以来, 特别是近年来网络与系统安全领域的论文, 我们发现软件供应链安全的相关研究处于起步阶段, 关于软件供应链安全问题的文献较少, 特别是许多与软件供应链相关的基本概念尚未厘清, 对于“软件供应链”一词的使用不够严谨, 缺乏一定界限。此外, 讨论软件供应链安全问题研究现状的调研报告数量较少, 且现有的调研报告[9, 10]调研范围不够明确, 对问题的理解不够深入等问题。为了便于研究者对软件供应链安全问题进一步开展研究工作, 本文从明确软件供应链安全的概念入手, 总结并归纳了软件供应链安全的研究现状, 并据此提出了该领域所面临的现实挑战, 指出了未来可能的研究方向。

2 背景介绍

2.1 软件供应链安全的定义讨论

软件供应链安全作为一种新的安全问题, 缺乏权威而准确的定义。在目前由产业界以及学术界发布的关于软件供应链安全的研究成果中, 通过经验主义的方式由软件供应链安全事件来定义软件供应链安全的概念是一种常见的方法。在这种方法下, 很难区分一些安全事件是否属于软件供应链安全事件, 其中一个重要的例子是心脏出血漏洞事件。

要明确软件供应链安全的定义, 首先要明确一个问题, 即什么是软件供应链。软件供应链一词, 顾名思义, 是根据软件生命周期[11]中一系列环节与传统供应链的相似性, 由传统供应链的概念推广而来的。传统供应链的概念是一个由各种组织、人士、信息、资源以及行为组成的将商品或者服务从供应者转移到消费者的系统。这一系统从自然资源以及原材料开始, 将其加工制成中间组件乃至送往终端消费者的终极产品[12]。在软件供应链的语境中, 商品与服务是软件, 供应者与消费者分别指软件供应商与软件用户, 自然资源、原材料、中间组件可以对应软件设计与开发的各个阶段中编入软件的代码、模块和服务, 加工过程则对应了编码过程, 工具以及设备。故而软件供应链可以被定义为一个通过一级或多级软件设计、开发阶段编写软件, 并通过软件交付渠道将软件从软件供应商送往软件用户的系统。

在软件供应链的这种定义下, 软件供应链的安全应该被定义为软件供应链上软件设计与开发的各个阶段中来自本身的编码过程、工具、设备或供应链上游的代码、模块和服务的安全, 以及软件交付渠道安全的总和。在这一定义中, 针对软件供应链的攻击相对于传统的针对软件漏洞[13]的攻击相比, 软件的攻击面[14]由软件本身的边界扩大为软件本身以及内部的所有代码、模块和服务的边界, 以及与这些模块相关的供应链上游供应商的编码过程, 开发工具和设备的边界, 显著降低了攻击者攻击的难度。同时软件设计与开发的任何一个阶段的安全问题都会直接影响供应链中所有下游阶段的安全, 显著扩大了攻击的影响。这两个方面可以解释为何针对软件供应链的攻击具有低成本, 高效率的特点。

心脏出血漏洞作为被基于 SSL\TLS 的软件和网络服务所广泛使用的开源软件包的漏洞, 被引入了这些软件和网络服务的开发阶段的上游代码和模块, 并且沿着软件供应链, 造成了供应链下游的广泛影响, 故而该事件应该被认为是一起典型的软件供应链安全事件。

2.2 软件供应链安全发展历程

软件供应链的概念早在 1995 年就被提出, 被用于描述软件开发过程中合作关系[15], 软件供应链安全这个概念的起源则较晚。近年来, 由于一系列具有相同模式的安全事件的发生, 软件供应链安全在产业界被提出, 并且受到了产业界和学术界的关注。2017 年 5 月, 微软公司一个研究团队声称他们的安全软件挫败了一起精心策划的, 通过攻击软件更新渠道, 将插入了恶意代码的第三方软件分发给使用该软件的多家知名机构的高级持续性威胁(Advanced Persistent Threat, APT)攻击[16]。在这篇声明中, 微软公司首次提出了“针对软件供应链的网络攻击”(Software Supply Chain Cyberattack)这一概念, 指向该起软件供应链安全事件。软件供应商视角下软件供应链安全相关概念的提出则要更早——在此之前的 2010 年, R.J. Ellison 和 C. Woody 两人针对当时软件开发过程中直接采购商品化的产品和技术(Commercial Off-The-Shelf, COTS)以及外包(Outsourcing)部分逐渐增加的趋势, 出于软件安全方面的考虑, 针对软件开发提出了软件供应链风险管理(Supply Chain Risk Management)的概念, 并介绍了相关风险的来源、种类、风险分析的方法, 威胁模型, 并讨论了在软件用户为有安全应对能力的组织机构的情况下, 应对这一风险的相关措施[17]。

软件开发过程中的 COTS 因素以及外包因素逐渐增加的趋势同时也受到了各国政府的高度关注——被商品化的产品或服务以及软件开发外包的对象, 在经济全球化的今天可能来自国外的产品供应商, 并被用于构建一个国家的关键基础设施和关键资源(Critical Infrastructure and Key Resources, CIKR)。信息通信技术(Information and Communication Technology, ICT)供应链的概念在 2009 年由 S. Boyson 和 H. Rossman 提出[18], 并被认为“是其他所有供应链的基础, 是供应链的供应链”[19]。鉴于国家 CIKR 对于 ICT 供应链的依赖性, 包括美国、俄罗斯、欧盟以及中国在内的各国及地区将 ICT 供应链安全上升到了战略地位, 并制定了用以保障 ICT 供应链安全的法律和技术规范[20]。由于软件提供商作为重要角色参与了 ICT 供应链, 软件供应链成为了 ICT 供应链的重要组成部分, 受到相关法律和技术规范的制约。

攻击软件供应链这一想法的产生可能远早于软件供应链的概念本身。早在 1984 年, K. Thompson 在其演讲中提出了一种通过攻击一种非常重要的软件开发工具——编译器, 污染所有通过此编译器编译并发布的软件的方法, 被称为 KTH 攻击[21]。Thompson 的方法可以在难以被发现的情况下修改编译器, 在其中设置后门。这与 XcodeGhost 事件相比仅仅缺少一个能够使得大量软件开发者信任的, 能够分发被设置了后面的编译器的非官方渠道。而在针对软件供应链的网络攻击出现之前, 针对传统供应链的网络攻击已经成为了一种重要而有效的攻击手段——通过攻击一个组织供应链上的薄弱环节来攻击这一组织[22]。这一理念被广泛地应用于 APT 攻击。在著名的震网事件[23]中, APT 组织利用震网病毒破坏设施的计算机系统, 进而破坏国家重要的基础设施。早在 2000 年, M. Warren 和 W. Hutchinson 就提出了利用网络攻击破坏传统供应链的可能性, 并列举了当时这类网络攻击可能的主要方式[24]。

在软件开发的生命周期中, 通过安全管理等手段尽可能降低包括软件供应链安全等安全问题造成的影响乃至损失这一思想的出现也早于软件供应链的概念本身。在 2004 年, 微软公司提出了著名的安全开发生命周期(Security Development Lifecycle, SDL)流程, 这一流程将软件开发流程划分多个阶段, 并在各个阶段引入各种安全措施, 保障软件开发以及软件终端用户的安全, 并建立了漏洞发现和处理框架机制。S. Linpner 的文档[25]详细介绍了 SDL 流程, 并将这一流程中的各个阶段之间的联系描述为一种螺旋式的关系。其中提出的多种安全测试工具以及软件发布运营管理规范, 至今仍然是处理软件供应链安全问题的一种重要手段。

2.3 典型的软件供应链安全事件分析

目前已经发生了多起软件供应链安全事件, 其中也包含了数起攻击者利用软件供应链上安全防范的缺失发动网络攻击的事件。通过整理和分析现有的软件供应链安全事件, 一个显而易见的规律是这些安全事件除了具有发生在软件供应链上的共同点外, 在表现形式上呈现多样化的特点[26], 并且与已知的网络攻击行为与软件安全事件高度重合。这其中有数起软件供应链安全事件在表现形式, 在软件供应链上的爆发点以及最终影响上具有典型性。其中 XcodeGhost 事件最为典型, 并且影响极大, 在这一节中会对其进行着重分析。

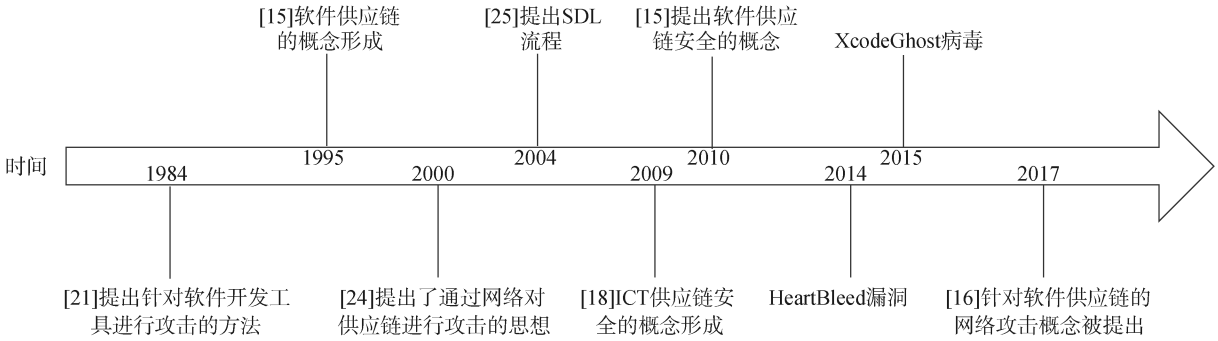


图 1　软件供应链安全的发展历程

Figure 1　Timelines of Development of Software Supply Chain Security

1) XcodeGhost 开发工具污染事件

Xcode 是一个用以开发 OS X 和 IOS 应用程序的集成开发环境(Integrated Development Environment, IDE), 由苹果公司官方发布。XcodeGhost 事件爆发于 2015 年 9 月, 是一起有攻击者的网络攻击事件。攻击者利用当时通过官方渠道获取 Xcode 官方版本困难的情况, 通过注入病毒污染了在非官方渠道发布的 Xcode[27]。其具体方法为修改 Xcode 软件的用于加载动态库的配置文件, 在其中添加了非官方的, 具有恶意功能的 Framwork 软件开发工具包(Software Development Kit, SDK), 同时利用 Xcode 开发环境中使用的 Object-C 语言的扩展类功能这一特性重写系统应用启动时调用的名为 makeKeyAndVisible 的函数, 使得恶意代码能够随着应用的启动而自行启动。多个知名 APP 的开发组织在通过非官方渠道获取了被注入病毒的 Xcode, 并使用被污染的工具编译出了被注入病毒的 APP。受感染的 APP 会将运行过程中获取的含有用户隐私数据的信息发送到病毒作者架构的域名为“init.icloud-analysis.com”的服务器中, 并可能被用于各种潜在的非法用途。这些木马化的 APP 受到开发者的签名认证, 得以在官方渠道发布, 对企业业务逻辑的正常运行以及它们在 OS X 和 IOS 等平台下用户隐私的安全造成了巨大的威胁, 并可能为攻击者带来了可观的经济利益。

2) CCleaner 恶意代码植入事件

CCleaner 是一款具有庞大用户群体的系统优化和隐私保护工具, 由 Piriform 公司开发和发布。2017 年 9 月, Piriform 官方声称该公司发布的被公开下载的 CCleaner 和 CCleaner Cloud 软件中被植入了后门。由于 CCleaner 软件具有隐私保护的功能, 这一事件可能影响了数百万终端用户的数据安全。此次事件的源头是攻击者入侵了该公司的开发环境, 篡改了 CRT 函数库(C Runtime Lib)[28], 在其中植入了恶意代码, 影响了所有在编译过程中使用了被污染的库函数的程序, 因而可以被认为是表现为恶意代码植入的软件供应链安全事件。类似的, NetSarang 公司开发的 Xshell 软件在 2017 年 7 月被以相同手段植入了恶意代码[29]。

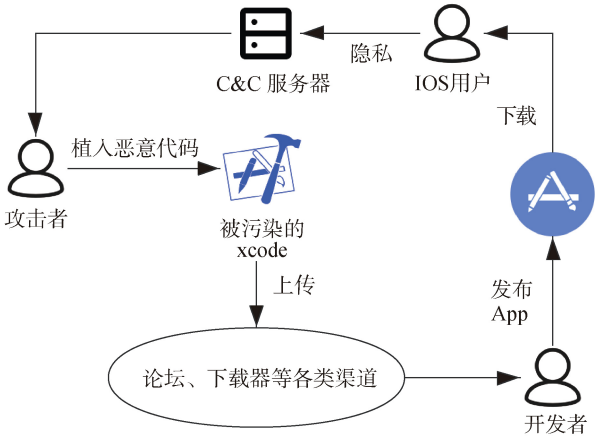


图 2　XcodeGhost 事件流程

Figure 2　Procedure of XcodeGhost Event

3) WireX Android 僵尸网络事件

WireX Botnet 是一种典型的僵尸网络, 通过控制大量的安卓设备发动 DDoS 攻击。WireX 僵尸网络中的僵尸程序病毒通过与普通 Android 程序捆绑进行伪装, 成功避过了 Google Play 应用商店对于恶意软件的检测。被捆绑的普通 Android 程序通过 Google Play 的官方渠道被用户下载安装后将 Android 主机感染为僵尸主机[30]。WireX Botnet 的网络传播过程利用了软件从软件开发商到交付用户使用的环节, 使其成为一次典型的软件供应链安全事件。

4) NotPetya 勒索病毒事件

勒索病毒(Ransomware)是一种通过加密用户数据向用户勒索赎金的病毒[31]。NotPetya 勒索病毒事件于 2017 年 6 月被披露, 来自欧洲多个国家的多个机构的数万台机器受到了感染。在这次事件中, 病毒感染用户计算机系统的行为与传统的勒索病毒行为

一致, 但是具有独特的网络感染手段。攻击者通过劫持乌克兰 MeDoc 公司开发的会计软件 me-doc 的升级渠道, 使得该软件的用户在软件更新的过程中自动下载并感染了 NotPetya 病毒[32]。NotPetya 事件中劫持软件更新渠道的做法使其成为一次典型的软件供应链安全事件, 同时攻击者利用大型组织供应链中的薄弱环节进行攻击的做法体现了供应链攻击的思想。

3 软件供应链安全研究现状

软件供应链安全是网络与信息安全领域中一类新型的安全问题, 针对软件供应链安全的研究在“提出问题、分析问题、解决问题”的流程中处于提出问题或分析问题的起步阶段, 在一定程度上缺少直接的有针对性且有价值的研究成果。尽管如此, 我们在调研了目前安全领域中与软件供应链安全相关的论文, 报告以及技术资料等内容之后, 发现部分研究者试图通过划分软件供应链的主要环节, 构建软件供应链结构模型和威胁模型来解释软件供应链安全问题的同时, 也有一些参考传统供应链的管理方式, 通过网络供应链风险管理(Cyber Supply Chain Risk Management, CSCRM)手段使得软件供应链管理规范化的尝试, 并且由于人们对软件供应链安全问题研究的深入, 一些研究者指出传统软件安全问题的技术手段有助于解决软件供应链安全问题。我们将目前软件供应链安全的研究分为上述三大分类, 并通过这一分类方法介绍软件供应链安全的研究现状。图 3 反应了自 2009 年以来 IEEE、ACM、Springer 和 Elsevier 等数据库中找到的直接与软件供应链安全问题相关的各类文献的分布情况, 除了这些内容与软件供应链紧密相关的文献, 一些文献的部分内容对软件供应链安全有重要的借鉴价值, 本文的后面部分内容同样会对这些文献进行介绍。

表 1 软件供应安全研究问题

Table 1 Software Supply Chain Security Research Issue

分类	研究问题
结构模型	① 软件供应链的主要环节 ② 软件供应链的威胁定位 ③ 软件供应商角色的系统模型
技术手段	① 软件漏洞挖掘和分析手段 ② 恶意软件及模块的识别和清除手段 ③ 网络劫持的检测和防御手段 ④ 用户端软件安全机制
管理手段	① 传统供应链风险管理在软件供应链中的应用 ② 软件供应链的风险管理手段 ③ 软件供应链安全标准的制定和实施

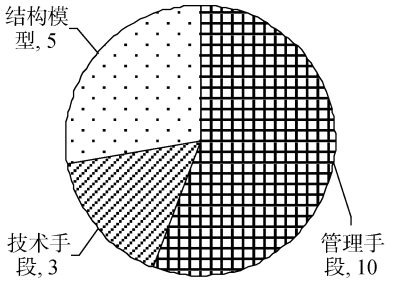


图 3 软件供应链安全问题文献情况

Figure 3 Papers in Software Supply Chain Security

3.1 软件供应链结构模型

3.1.1 软件供应链的主要环节模型与威胁定位

通过第二章中对软件供应链安全的发展历程的介绍可知, 软件供应链安全概念的产生源于大量的软件供应链安全事件。360 威胁情报中心的研究人员以这些事件为切入点, 通过对具体案例的分析总结, 从软件生命周期的视角观察软件供应链, 提出了将软件供应链简单抽象成开发环节、交付环节、使用环节这三个环节的划分方案[33]。在这一划分方案中, 软件在开发环节中按照软件工程的思想, 经过需求分析、设计实现、测试等流程, 形成最终用户可用的形态, 在交付环节中通过软件交付渠道从软件提供商被分发到终端用户, 并在最后的使用环节中经历用户使用产品的整个生命周期, 其中包括可能的升级和定期维护流程。

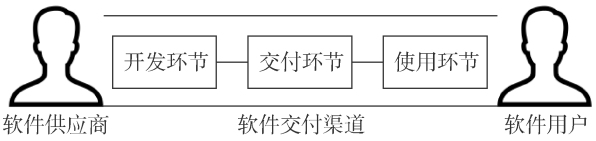


图 4 软件供应链的主要环节模型

Figure 4 Key Factor Model of Software Supply Chain

软件供应链主要环节的抽象划分方案得到了其他研究者的广泛认同, 其他关于软件供应链主要环节的划分方案均与该方案有很高程度的共性。如在马金鑫对于软件供应链攻击的研究中, 将交付环节和使用环节合称为运营环节, 将软件的开发环节称为生产环节, 认为软件供应链攻击主要体现在生产和运营两个环节[34]。根据这一划分方案, 研究者可以标记典型软件供应链安全事件在供应链上的位置, 这是一个容易理解的对软件供应链安全事件进行分类的方式。邹维, 霍玮等在这一方案的基础上划分软件供应链攻击的类型, 将之总结为针对三大环节的四种攻击途径[35]。这四种途径分别为针对软件开发过程, 即开发环节的开发工具攻击和源代码攻击,

针对软件分发系统, 即交付环节的下载网站攻击以及针对软件更新网站, 即使用环节的补丁网站攻击。周振飞则在这一方式的基础上, 根据软件供应链三个主要环节的划分方法对已知典型软件供应链安全事件进行分类, 并且针对三个环节分别总结攻击面。然后通过 STRIDE 威胁模型建模方法[36], 将六种常见的信息系统安全威胁类型和软件供应链的三个主要环节组合, 对软件供应链进行威胁建模, 总结出了十类软件供应链威胁以及软件供应链威胁主要存在的两个方面——影响软件本身的功能以及影响软件运行环境中其他资源[37]。

3.1.2 软件供应商角色系统模型与社会技术框架

根据传统供应链的定义以及我们对软件供应链定义的探讨, 供应链上的角色可以分为软件供应商, 用户以及软件交付渠道三类。在这三类角色中, 软件供应商在软件的生命周期中具有举足轻重的地位, 对软件供应链安全负有重要责任——软件供应商在开发环节中构建软件, 在交付环节为软件的分发提供或选择渠道, 在使用环节中为用户提供可能的维护和升级服务。由于 COTS 和外包部分在软件生命周期中比重逐渐增加的趋势, 软件供应商的结构从原先的单一实体逐渐变为一个复杂的结构, 成为一个系统的系统(System of Systems, SoS), 为软件工程带来了全新的挑战[38]。为了应对这一挑战, 一些研究者通过研究软件供应商的业务行为, 建立供应商角色的结构和开发业务等活动的相关模型。

知名供应链风险管理咨询服务公司 Interos Solutions 发布的研究报告中将 ICT 供应商分为三个级别, 即主要供应商、向主要供应商提供产品和服务支持的层级供应商以及通过商业、金融或其他相关关系与层级供应商关联的其他实体[39], 软件供应商作为 ICT 供应商的一种类型, 也具有这种多级别之间相互关联的特性。C. Woody 和 R. Ellison 将软件供应商的集合形容为一个所有为软件内容做出贡献或有机会修改软件内容的利益相关网络[17]。C. Alberts 和 A. Dorofee 等引用了这种定义方式, 将软件供应商角色的系统模型构建为一个以总承包人(Prime Contractor)为根节点的树状结构[40]。每个节点通过复用(Reuse)、外包、内部开发(Develop in House)、购买(Acquire)等方式与其子节点相连接, 子节点的角色可能由 COTS、开发者、外包商以及遗留软件(Legacy Software)[41]等实体担任, 如图 5 所示。

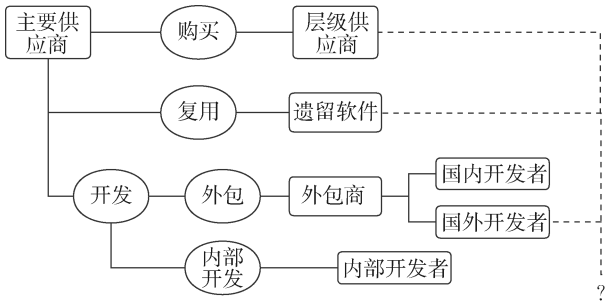


图 5 软件供应商系统的结构模型

Figure 5 Structure Model of Software Supplier

与此同时, 一些研究者也尝试通过社会技术框架(Socio-technical Framework)方法从系统分析的角度对软件供应商们进行本体论的分析(Ontological Analysis), 建立目标系统的模型。S. Kowalski 提出了框架所包含的两个基本模型, 即动态的社会技术系统(Socio technical system) 模型和静态的基于共识的安全(Security by consensus)模型[42]。在此基础上, 他和 Sabbagh 构建了单个软件供应商实体的模型, 并将这些实体进行连接, 构成软件供应商系统模型[43]。他们在这一模型上寻找导致软件供应链安全问题中存在的社会关系问题和技术问题。

由于软件供应商间的跨国合作对国家信息系统以及其他关键资源安全的影响, 主要软件供应商视角下供应链的管理作为软件供应链安全研究的组成部分, 最早得到研究者的重视。上述文献对于软件供应商的结构和业务流程进行了解析, 展开了供应商系统的内部结构, 作为寻找其中薄弱环节的依据。

3.2 防范软件供应链安全事件的技术手段

使用软件供应链的主要环节模型, 分析并研究典型的软件供应链安全事件, 可以定位这些事件发生的环节, 锁定事件中涉及的关键技术, 将软件供应链安全事件归约为已知的软件与网络安全事件类型, 并且使用该领域的技术手段检测和防御针对软件供应链的攻击, 降低软件供应链安全事件的影响。研究发现典型的软件供应链安全事件可以归约为以下三类: 软件产品内部开发过程中产生的以及从上游继承的软件漏洞、软件开发环节的污染使得软件产品变为恶意软件、软件交付渠道以及维护和升级渠道遭到网络劫持。通过将这些领域的研究与典型的软件供应链安全事件中体现的关键技术相结合, 可以讨论这些领域的技术手段在软件供应链中的应用。最后, 由于软件供应链中代码和服务由上游流向用户端的特性, 部署在终端用户的软件安全机制在软件供应链安全事件的应对中显得尤为重要。

3.2.1 软件漏洞挖掘和分析手段的应用

软件漏洞是软件以及信息系统长期以来一直面临的重要安全问题, 会导致软件产生错误的运行结

果, 或者表现出预期之外的行为。近年来, 随着软件需求和功能的不断扩展, 软件的复杂度不断提高, 其存在源于软件的高复杂度的软件漏洞的产生无法避免, 并且有着越来越复杂和多样化的趋势。这些软件漏洞可能被攻击者利用, 对软件以及相关联的计算机系统造成严重的安全威胁[44]。在这一背景下, 软件供应链的存在则为软件漏洞的引入提供了新的渠道——软件漏洞现在不仅产生于内部开发流程中, 还可能来自上游供应商提供的 COTS、复用开源软件[45]或翻译遗留软件[46], 显著影响了用户对软件功能的信任, 对于软件生命周期中软件供应商和其他研究机构、社会团体对抗软件漏洞的能力提出了更高的要求。

软件漏洞挖掘是一种对软件的可执行文件以及源代码进行动态或静态分析, 检测软件可能存在的漏洞的方法, 是研究者对抗软件漏洞的第一步。软件漏洞挖掘的技术思路主要有三类, 分别为基于源代码的漏洞挖掘, 基于代码特征的漏洞挖掘和基于模糊测试(Fuzz testing)的漏洞挖掘。基于源代码的漏洞挖掘需要在拥有软件源代码的前提下进行, 因而适用于软件供应商自身在软件开发环节测试软件功能的场景, 如 V. Livshits 等使用静态分析的方法在 Java 源代码中进行脆弱性(vulnerability)检测的策略[47]。基于模糊测试的挖掘是一种动态的漏洞挖掘方法, 使用黑盒测试的手段, 不需要掌握目标软件的源代码。P. GodeFroid 等的研究成果基于白盒测试, 实现了模糊测试的漏洞挖掘方法[48]。基于代码特征的漏洞挖掘方法根据已发现的漏洞提取漏洞特征, 然后检测目标中是否含有该特征。由于机器学习方法在软件漏洞挖掘中的应用, 该方法与机器学习方法的结合可能是成为软件漏洞挖掘未来主要的发展方向, G. Grieco 和 G. Grinblat 等使用机器学习方法根据软件的内存错误信息训练测试模型, 检测软件漏洞的研究[49]是该方向的一个尝试。

在通过软件漏洞挖掘方法确定软件存在漏洞之后, 需要通过软件漏洞分析找到软件漏洞形成的原因, 类型, 位置, 确定修补软件漏洞的方案。一些研究者在软件漏洞分析各个环节的研究成果提供了软件漏洞分析的有效工具: G. Portokalidis 等提出的 Agros 工具[50]是软件漏洞匹配识别的一个典型方法。J. Xu 等人提出的 CREDAL 方法[51]可以定位软件漏洞。而 SemFix[52]和 GenProg[53]等工具则采用自动化补丁技术, 可以自动生成漏洞修补程序。

软件供应链的主要供应商和其他安全人员在挖掘, 分析和修补自身内部开发的漏洞的同时, 也需要通过相应的技术手段, 减少来自上游软件和模块的漏洞对最终软件产品的影响, 一些研究者和开发人员针对这种应用场景进行研究, 提供了一些有效的方法, 并开发了相关工具。Y. Ma 的研究提供了一种通过测量依赖关系, 代码重用关系及知识流网络(knowledge flow network), 在软件供应链中量化和交流开源软件安全风险的方法[54]。腾讯安全玄武实验室开发的阿图因工具[55]则采用了一种基于知识图谱对存在已知漏洞的共享软件模块进行反向溯源的软件漏洞挖掘方法, 可以沿着软件模块的复用关系网络快速找到因为复用存在已知漏洞的模块而引入漏洞的软件, 特别适用于软件供应链的应用场景。

3.2.2 恶意软件及模块的识别和清除

恶意软件(malware)是一类被设计为用于破坏计算机系统的软件的总称, 典型的恶意软件有蠕虫、木马、病毒, 后门等[56]。随着软件复杂度的提高、恶意软件的攻击模式与功能展现出多样化的发展趋势, 出现了如僵尸网络病毒, 勒索病毒等新型的恶意软件类型。软件供应链的存在则为恶意软件的开发和传播提供了便利条件。在 Xcode 事件中, 攻击者设计了作为恶意软件的 Xcode 开发工具并通过非官方渠道提供下载, 使之进入并污染了软件供应链, 导致所有经过该工具开发的软件成为新的恶意软件, 并得到软件供应商的签名担保, 披上了合法软件的外衣, 在官方渠道上被交付给终端消费者。这开创了恶意软件增殖的新模式, 并受到攻击者的青睐, 发展出通过恶意软件模块污染第三方库等攻击方式。与此同时, 由于软件供应链攻击的这种攻击模式需要恶意软件或模块作为载体, 以恶意软件为成果, 针对恶意软件及模块的识别和清除方法会成为检测和防御软件供应链攻击的重要手段。

由于恶意软件具有的破坏计算机系统的恶意性, 攻击者会采取隐蔽手段对抗安全人员的检测, 使得恶意软件的源代码难以获取, 并可能采取代码混淆(Obfuscation)等手段增加逆向分析的难度。C. Linn 的研究成果[57]提供了一种通过混淆技术对抗软件静态反汇编分析的方法。为了应对恶意软件的隐蔽性, 研究者设计了多种分析采用了混淆技术的恶意代码的方案, 如 W. Xu 等设计的 JStill 工具[58], 可以识别经过混淆的 JavaScript 代码。彭小详等人的研究针对加壳技术, 提出了对恶意程序进行自动脱壳的方法[59]。与软件漏洞挖掘方法类似的, 研究者对恶意软件的特征提取方法进行了研究[60]。在此基础上, 产生了多种基于统计分析[61]以及机器学习[62]方法对恶意软件代码进行静态分析的应用成果。同时, 采用动态分

析识别恶意软件的沙箱办法也得到了发展, 诞生了CWsandbox[63]等模拟真实系统环境运行目标软件进行恶意软件动态自动分析的工具。

在软件供应链攻击这一特定场景中, 攻击者对现有软件与模块进行修改, 注入恶意代码并重新打包是用作攻击并污染软件供应链的恶意软件和模块的一个主要来源。对现有软件进行修改的流程类似 Android 平台应用安装包(APK)的重打包(Repackage)过程。Z. Wu 等提供了一种识别这类应用的方法[64]。在这种情况下, 软件对于反汇编分析等静态分析方式的抵抗能力不仅能够被攻击者利用, 隐蔽恶意软件, 也可以被合法软件的供应商利用, 防止正常软件通过“逆向分析-注入代码-重新打包”的流程被攻击者修改为恶意软件。与此同时, 由于修改后的软件与模块势必与未注入恶意代码的软件与模块的代码产生一定区别, BinDiff[65]等在软件逆向分析的基础上, 基于图比较算法分析相似二进制文件间差异的工具可以用于检测目标软件与官方渠道版本的差异, 作为识别此类恶意软件的重要手段。

在识别恶意软件的基础上, 需要对恶意软件或模块进行清除, 常用的手段包括隔离、删除等。在软件供应链上, 恶意软件会影响供应链的开发环节和交付环节。这就要求软件供应商通过技术手段清除恶意软件或模块对开发环境以及自身交付渠道的污染。独立于供应商的软件交付渠道, 如 Google Play 等集中式的软件交付渠道平台, 同样需要承担识别并清除通过该渠道交付的恶意软件的责任。

3.2.3 网络劫持的检测和安全防范

在典型的软件供应链安全事件中多次出现针对软件交付环节和使用环节的一种攻击模式——攻击者通过中间人攻击(Man-in-the-middle Attack, MITM)或其他攻击方式, 劫持或篡改用户与软件交付或维护升级渠道之间的数据, 使得用户从交付渠道获得了非预期的恶意软件, 或者在软件维护和升级的过程中被植入了恶意代码。这种以网络为切入点对软件供应链进行攻击的模式被研究者称为网络劫持。网络劫持作为一种常见的网络攻击手段, 有着包括 Web 前端会话在内的多种劫持对象, 以及一些成熟的劫持方法[66]。由于目前软件供应链的交付阶段和使用阶段对于网络环境的高度依赖, 这被认为是污染软件供应链的关键技术[37]。在用户从各种渠道获取软件下载或者更新的时候, 需要与渠道服务器建立数据链路以传输数据, 此时攻击者通过攻击这条数据链路上的任意一点, 就可以篡改数据链路上交换的数据。这一点可能是任意一个网络节点, 甚至可能是服务器本身——在NotPeya事件中, 软件供应商的更新 FTP 服务器被攻击者控制, 使得大量该供应商的 me-doc 软件用户的计算机系统被注入病毒。安全的网络连接是软件安全的保证, 针对网络劫持的检测的防范措施对软件供应链安全至关重要。

由于服务器端一般具有一定程度的安全措施, 中间人攻击是网络劫持的主要实现方式。在数据链路上缺乏有效的身份验证方式的情况下, 攻击者通过控制链路上的一个网络节点, 就可以对通信双方实施中间人攻击, 典型的攻击方式如 DNS 劫持[67]以及对无线通信链路[68]的劫持。U. Meyer 等实现的在 3G 基站和移动设备用户之间进行中间人攻击的方式体现了中间人攻击的有效性和数据链路的脆弱性[69]。使用 SSL\TLS 协议等加密的通信方式[70]或者有效的身份验证方式[71]是防御中间人攻击的重要方式, 但是研究者的结果证明即使采用了这类安全措施, 链路仍然有被攻击者劫持的风险。一些研究者试图通过对现有网络协议的扩展和增强抵御网络劫持攻击: R. Oppliger 等的工作中提出了一种基于 SSL/TLS 会话感知(session-aware)的安全增强方案, 可以抵御中间人攻击[72]。M. Alicherry 等人则提出了一种名为双重检查(DoubleCheck)的多渠道获取服务器证书以抵御中间人攻击的解决方案[73]。

与此同时, 另一些研究者的研究成果试图通过“入侵检测-响应”的流程防御网络劫持攻击。R. Gill 等人研究了在 802.11 无线网络协议通信过程中通过中间人攻击进行会话劫持的流程, 并据此提出了一种被动检测的方案[74]。T. R. Schmoyer 等的研究成果在检测无线网络中间人攻击等入侵行为的基础上, 增加了根据攻击方式自动采取响应措施, 阻断攻击的机制[75]。在针对网络入侵的响应中, 网络攻击的追踪溯源是一种重要手段, 在软件供应链攻击的场景下, 可以锁定攻击者, 以采取进一步措施。姜建国等人对常见的网络攻击追踪溯源技术根据虚假 IP、匿名网络、跳板主机等场景进行分类, 分别总结了相关的研究成果[76]。

上述文献提出了一些常见的网络劫持手段, 并提出了多种防御这类网络攻击的方法。值得注意的是, 根据对现有软件供应链典型安全事件的分析, 针对软件供应链的攻击获得成功后, 攻击者的目的除了破坏受影响用户的计算机系统外, 通过植入僵尸网络病毒构建僵尸网络进行 DDoS 攻击, 或者通过植入后门记录并上传用户的隐私数据等信息等利用方式非常常见。这意味着软件供应链攻击的成功可能伴随网络流量的异常变化, 使得安全人员可以

通过异常流量检测[77]等方式发现可能的针对软件供应链的成功攻击的迹象, 及时采取应急响应措施, 降低攻击对于网络环境与计算机系统的影响。

3.2.4 用户端的软件安全机制

用户在软件供应链中处于最下游, 由于软件供应链中上游产生的安全事件会随着供应链传递到其下游所有的节点, 用户终端在需要应对几乎所有的软件供应链安全事件的同时, 也是几乎所有的软件供应链攻击的目标。而当安全人员发现了新的软件供应链安全事件后, 针对安全事件的应急处理措施, 例如相应系统或软件漏洞的补丁以及恶意软件的检测与清除方式, 也需要部署到用户终端后才能生效。软件供应链安全事件的这一特性使得部署在用户终端的软件安全机制可以显著提高整个软件供应链系统承受安全事件的能力。

用户端的软件安全机制可以来自用户的计算机系统, 也可以来自软件本身。现代计算机系统的操作系统会采用各种机制来管控软件的行为, 例如 Android 操作系统基于 Linux 操作系统 UGO (User-Group-Other)访问控制策略的沙箱机制[78]——每个运行在 Android 操作系统上的 APP 都运行在自己被操作系统分配的沙箱内, 不能访问被分配给其他应用程序的系统资源。这种机制在一定程度上能够阻止恶意软件的恶意行为, 但是可以被恶意软件通过权限升级等方式绕过[79]。SELinux(Security Enhanced Linux)等安全增强型操作系统能够有效保护用户的计算机系统[80], 但可用性较差。故而研究者做了一些将 SELinux 的安全机制用于 Android 运行环境上的尝试, 获得了一定的成果。例如 A. Shabtai 等将 SELinux 用于保护 Android 设备的研究[81]。由于软件供应链在开发环节受到攻击时生产的软件产品一般受到合法厂商认证, 有合法软件的外衣, 基于硬件的可信计算(Trusted Computing)安全机制无法应对这类软件, 但是可以阻止在交付渠道被篡改的恶意软件运行, 也可以帮助软件供应商保护自身开发环境和渠道的安全[82], 阻止软件供应链安全事件的发生或是其向用户终端的流动。

尽管软件的升级和维护可能被攻击者利用, 引发软件供应链使用环节的安全事件, 但是这一机制可以允许软件供应商和安全人员及时修补软件漏洞, 缓解安全事件的影响, 是非常重要的软件安全机制。除此之外, 在软件安全事件发生时, 软件本身的安全机制可以缓解漏洞或者软件部分模块的恶意行为对软件正常模块以及用户计算机系统造成的影响。例如使用 C 语言开发的软件在编译过程中可以启用数据执行保护(Data excution prevention), 栈保护, 地址空间随机化(Position Independent Execution)以及只读数据重定位保护(RELocation Read-Only)等漏洞缓解技术[83]。这些安全机制在阻止栈溢出等常见类型的软件漏洞被攻击者利用的同时, 对未知的软件漏洞, 只要其符合相应的模式, 也有一定的缓解能力。软件供应商在制成软件产品的时候需要根据软件的类型和可能的运行环境启用软件的安全机制, 并在软件生命周期中提供可靠的维护和升级服务。为了使得软件本身的安全机制生效, 用户需要通过及时更新等方式配合安全事件处理措施在用户端的部署, 也可以通过安装反病毒软件等安全防护软件降低自身计算机系统受到来自软件供应链或其他渠道安全事件影响的风险。

3.3 软件供应链的风险管理

软件供应链的产生源于信息技术的飞速发展, 以及随之而来的软件功能复杂化, 架构模块化和开发产业化的趋势, 是软件产业以及整个信息技术产业由新兴走向成熟的必经之路——供应链的产生可以进一步挖掘降低成本, 满足客户需要, 提高产品竞争力的潜力[84]。许多传统行业都存在供应链, 例如制造业与食品药品行业, 并发生过或者正在发生严重的供应链安全问题, 例如 2005 年的“苏丹红”食品添加剂事件。为了解决传统供应链的安全问题, 研究者提出了供应链风险管理(Supply Chain Risk Management, SCRM)的概念, 并且在社会各界的共同努力下, 完善了供应链管理的标准与规范, 通过国家主管部门予以推行。与传统供应链类似, 软件供应链安全问题的解决同样需要规范化的流程管理和标准。一些研究者通过研究软件供应链与传统供应链的共性与差异性, 提出了软件供应链风险管理的基本规范与安全评定机制, 在此过程中, 多部与软件供应链安全管理相关的安全标准得以出台。

3.3.1 传统供应链风险管理机制的推广

软件供应链与信息技术产业, 特别是互联网产业息息相关, 与传统产业的供应链相比, 供应链中的多数渠道依托现有的互联网服务, 具有明显的互联网特性。这一特性在实践中表现为软件供应链中各个环节的流程具有一定的随意性——曾经有研究者指出特立独行的软件开发者(maverick developers)对软件和物联网(Internet of Things, IoT) 供应链安全造成的负面影响, 并且认为软件行业缺乏与航空工业类似的库存清单追踪管理体系[85]。在 Xcode 事件中, 被攻击者篡改的开发工具被正规软件供应商的开发者通过非官方渠道下载获得, 这在传统供应链

行业是难以想象的。但是祝国邦等人的研究通过将软件供应链的业务流程与传统供应链进行比对, 如表 2, 发现其中关键因素的基本属性差异不大[86]。这表明传统供应链的业务流程以及管理机制对于软件供应链的正规化发展而言有非常重要的参考价值。

传统供应链管理的研究经历过从供应链管理(Supply Chain Management, SCM)到 SCRM 的演变过程。SCM 主要关注供应链的效率问题, 包括供应商之间的协调机制、交付渠道的设计、以及外包与采购策略。这些因素同样出现在软件供应链中, 例如软件供应商的多级别结构模型, 软件交付渠道, COTS 与外包等。而 SCRM 认为脆弱性(Vulnerability)对供应链的绩效有巨大负面作用, 因而将风险管理的理论和技术拓展到 SCM 中[87], 最大程度降低风险带来的损失, 并将 SCRM 分为风险识别、风险评估、风险决策、风险过程管理四个领域。类似的划分方法同样在对信息安全风险的相关研究中出现。尽管传统供应链安全领域研究者总结的风险因素与软件供应链安全所面临的主要风险不一致[88]。但是其中领域的划分与各领域的相关研究和相关概念的模型和框架, 如 E. Pater 等提出的供应链暴露(Supply Chain Exposure)概念以及评分模型[89]等, 对软件供应链的风险管理机制的研究有参考意义。利用传统供应链质量保证管理的工具, G. Reddy N.的研究提供了一种软件开发流程的改进方法, 并提供了一个新的项目管理模型作为流程改进的解决方案[90]。

表 2 软件供应链与传统供应链关键业务对比

Table 2 Comparison of the Key Business of Software Supply Chain and Traditional Supply Chain

传统供应链	软件供应链
购买原材料	准备编程语言和知识库
准备生产、工作环境	准备软件开发环境
准备专业人员	组织开发人员
组装中间产品、最终产品	通过编码实现软件模块和最终软件产品
销售网络运输	网络平台下载、光盘刻录销售
消费者消费、维保、升级	软件安装, 软件漏洞修复、升级

3.3.2 软件供应链的风险管理研究

在近年来典型的软件供应链安全事件以及针对软件供应链的网络攻击这一概念出现之前, 软件供应链作为 ICT 供应链或者网络供应链(Cyber supply chain)的一部分被学术界和产业界所认识, 其主要环节模型当时尚未形成。ICT 供应链安全及其风险管理同样是新兴的概念, 这从对其称呼的混乱性可以看出——ICT 供应链与网络供应链等一系列概念尽管看上去不同, 其实指向同一实体, 彼此间可以相互转换[91]。此时研究者对于软件供应链的关注主要源于国家 CIKR 对于软件供应链的依赖, 故而这一阶段对于软件供应链风险管理的研究主要针对供应链中的大型实体, 如主要软件供应商以及作为软件用户的机构、集团或组织。这些大型实体本身具有复杂的结构体系, 在提供了管理能力的同时也增大了管理的需求。

R. Ellison 等的研究成果为主要软件供应商, 特别是在大型软件开发项目中处于总承包人地位的软件供应商对其上游供应链的风险进行评估提供了基础。他们罗列了在软件开发周期中从立项到被废弃各个阶段总承包人管理供应链的主要活动以及软件在开发、交付和使用过程中可能遇到的威胁, 然后将项目供应链风险的评估分为供应商能力、软件安全性、软件控制逻辑以及软件运营管理四个方面[15]。在此基础上, J. Alberts 等的研究根据 R. Charette 关于风险定义的三大条件的讨论[92], 确定了软件供应链中能够触发风险的实体, 据此提出了一种基于驱动因素(drivers), 即能够强烈影响最终结果的因素的, 评估软件供应链风险的方法, 并绘制了该方法的主要流程模型, 如图 6 所示, 同时给出了软件供应链中的 24 种驱动因素[40]。这一研究为处于总承包人地位的软件供应商管理供应链中的风险提供了一种有效的方法。

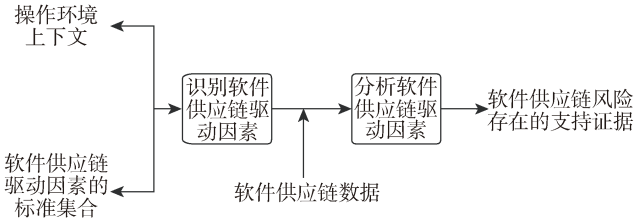


图 6 软件供应链风险评估流程

Figure 6 Assess Procedure of Software Supply Chain Risk

上述研究表明软件供应链的风险管理需要管理者对于组成软件的来自供应链的各类实体的信息的充分了解, 这一管理者可能是担任总承包人角色的软件供应商, 在软件用户为大型实体的情况下也可能为软件用户, 或者说软件的收购者本身。无论管理者的身份如何, 软件供应链的管理者在规划和设计供应链结构, 选择上游供应商, 以及在过程中对供应链进行管理时, 需要掌握多种关键信息以评估来自供应链的风险。这种掌握关键信息的需求很多时

候无法被满足, 并且可能产生管理者与供应商的矛盾——COTS 的供应商在披露产品关键信息的同时有泄露产品关键技术的风险。R. Dan 的研究关注了软件供应链风险管理过程中的这一矛盾, 并且提出了一种使得 COTS 供应商能够在不泄露关键技术信息的情况下披露产品安全相关信息的“代表性保证(representational assurance)”方法[93]。并且介绍了 CHESS[94], Treemap[95]等多种适用于供应链管理者根据 COTS 供应商披露的信息评估系统可靠性这一情景的方法和工具。

3.3.3 软件供应链安全标准的制定与实施

软件供应链正在成为网络与系统安全的一个正在兴起的重要分支, 软件供应链上管理措施的缺乏和潜在的攻击面对整个信息产业的安全造成了巨大的威胁, 需要有能够被信息产业内部相关企业和组织所共同遵守的业务安全守则与标准。一些研究者对于这类安全守则应该如何制定进行了研究。S. Boyson 通过企业风险管理, 供应链管理以及网络安全三个领域的研究成果和相关标准进行了总结性的研究, 并根据自己建立的信息技术产业商业生态系统(Business ecosystem)模型, 与其他行业, 如药品行业的传统供应链运行的实际情况进行对比, 分析了软件供应链乃至网络供应链所面对的威胁, 设计了一个评估企业网络供应链风险管理能力成熟度的模型[96]。这个模型将网络供应链管理分为公司治理(Governance)、系统完整性(Systems Integration)和运营(Operations)三个层次, 每个层次包含多项细则, 并通过企业在这些细则上的表现对一个企业网络供应链管理的水平的成熟度进行评价。这一研究成果可以指导企业对自身的供应链管理水平进行评估, 并据此提高自身的网络供应链管理水平。

这些关于安全守则和标准制定的研究很多本身是与制定标准的组织或机构, 例如美国国家标准与技术研究院(National Institute of Standards and Technology, NIST)和国际标准化组织(International Organization for Standardization, ISO)等合作进行的, 故而其研究成果也被用于制定与软件供应链安全相关的多个安全标准, 如 NIST 颁布的《ICT 供应链风险管理标准》(NIST SP 800-161)的草案。在这一过程中, 来自信息安全, 系统工程, 软件工程, 供应链和物流等多个不同专业的研究人员进行了跨学科的交流, 并参与了标准的制定。N. Bartol 对这些标准的制定过程进行了总结, 并认为这类跨学科的交流为软件供应链乃至网络供应链安全的研究做出了巨大的贡献——研究人员发现许多在网络供应链安全中缺失的机制已经在另一个学科中得到了充分的研究[91]而许多大型企业、组织和机构, 例如微软公司, 在没有意识到软件供应链安全这一概念的情况下, 已经在根据这些其他领域的研究来保障自身的供应链, 并且发展出一些具有实用价值的实践流程和工具。Bartol 罗列了现有的软件供应链安全领域和其他信息技术安全领域的安全标准以及在实践过程中衍生出的重要流程、工具和技术, 并且肯定了这些成果对于软件供应链乃至网络供应链安全的贡献。

4 软件供应链安全的现实挑战

根据上述对软件供应链安全问题背景的介绍, 结合对软件供应链安全研究现状的调研, 可以总结出现阶段软件供应链安全所面临的六大现实挑战, 同时在表 3 中总结了这六大挑战与对应的机遇。

表 3 软件供应链安全研究所面临的挑战和机遇

Table 3 Challenges and Opportunities of Software Supply Chain Security Research

挑战	机遇
软件复杂度增加	新的软件质量控制手段
软件的开源化趋势	针对开源软件的风险评估方法
渠道受到网络攻击	供应链各环节对抗网络劫持的手段
软件使用生命周期的安全	构建高效、安全的软件补丁部署体系
面对利益驱动的攻击	对攻击的防御、检测和响应手段
软件供应链面对各类风险	建立有效的软件供应链风险评估应对机制

1) 软件不可逆的复杂化趋势

软件的复杂度正随着对软件功能需求的不断提高而不断上升, 由于信息技术产业的不断发展以及其它产业对软件依赖的加深, 这一趋势是不可逆的[21]。这一过程不仅会增加新软件的开发难度, 也会提高现有软件维护升级的难度, 并且由于软件扩展过程中新增的组件需要与原组件进行交互, 软件复杂度的上升与软件功能增加之间的关系可能是非线性的。这一现状导致了软件开发的组件化, 复用化以及软件供应链的产生, 并且为软件供应链的安全带来极大的挑战——软件设计缺陷与开发中不可避免产生的漏洞在随着软件功能的增加而非线性地增长的同时, 将沿着软件供应链的树状结构由上游向下游以非线性的速度扩散。这为软件的业务逻辑与安全质量控制等工作的水平提出了全新的要求。

2) 软件的开源化趋势

在软件趋于复杂化的同时, 软件开发人员通过各种方式降低软件开发的复杂度。开源软件的存在为软件开发人员提供了一种基于开源软件的开发模

式, 将具有与待开发软件相同或相似功能的开源软件引入软件的开发过程。软件的开源化不仅出现于应用软件, 还体现在系统软件、设备固件、编译与测试软件等支撑应用软件开发的软件中。软件的开源化将开源引入了软件供应链, 在提高软件供应链效率的同时, 也将开源软件的安全问题引入了软件供应链——开源软件在开发过程中可能存在缺陷和漏洞, 也有恶意的开发者故意开发具有后门的开源软件。不仅如此, 开源软件提高了软件供应链的暴露程度, 可能作为攻击者攻击软件供应链的切入点。因而软件的开源化趋势使得软件供应链的安全面临极大的挑战。

3) 不安全的软件交付机制

软件供应链将作为商品的软件从作为供应链生产者的软件供应商转移到作为消费者的软件用户的计算机系统, 这一过程被称为软件交付, 并且需要通过软件交付渠道进行。软件交付渠道可以是传统的以光盘等存储设备为载体的物流体系, 但由于互联网的发展以及通过其交付软件的优越条件, 以互联网为交付渠道被供应链中的生产者与消费者所广泛接受, 并且出现了集中式的软件交付渠道, 可以将软件快速地传播给大量用户[44]—软件交付机制在降低成本, 提高效率的同时, 使得软件交付渠道的安全受到网络安全的制约, 并且在软件供应商和用户之间引入了不受供应商控制的第三方角色。如何在网络安全时刻受到威胁的情况下尽可能保证可信任交付渠道的安全和软件生态的纯净, 并且使得供应链中的各方, 特别是用户拥有识别恶意渠道的能力, 是软件供应链安全中亟待解决的一个问题。

4) 软件补丁部署的保障措施

软件的生命周期并非结束于交付于用户之后, 而是直到软件被废弃为止。软件生命周期的这一特点为软件供应链引入了使用环节。由于软件开发过程中不可避免的缺陷与漏洞, 在软件的使用环节中软件供应商有义务监控相关的安全事件并及时提供安全补丁进行修补。与此同时, 在软件生命周期中用户需求的变化会促使软件供应商升级软件功能。这些安全或新功能的补丁作为软件供应链上流动的产品与服务的一部分, 对软件的安全与业务逻辑非常重要, 需要通过有效的渠道快速地部署到用户端, 对于软件供应商的业务能力有很高的要求, 并将与交付渠道中所面临的情况类似的安全威胁引入了软件供应链。如何构建高效的软件补丁部署体系, 并设计保障该体系正常运行的安全措施是软件供应链安全研究的一大挑战。

5) 可能面对利益驱动的攻击

在当今社会中软件承担了许多重要的功能, 使得对软件的攻击具有潜在的利益, 吸引了攻击者通过网络攻击等各种方式对软件进行攻击。软件供应链的存在展开了软件的内部结构, 将软件原本被隐藏的缺陷暴露在攻击者面前, 降低了攻击者的攻击难度, 同时为攻击者的攻击行为提供了天然的扩散渠道, 具有非常高的攻击价值。这使得软件供应链成为利益驱动的攻击行为的重要目标。为了应对利益驱动的攻击行为, 在引入有效的被动安全防护方案以提高软件供应链系统面对攻击的生存能力, 也需要根据软件供应链安全的实际情况, 设计并开发检测此类攻击行为, 追踪攻击者以及对攻击行为做出响应的技术手段和工具。

6) 需要更有效的供应链管理方法

供应链这一系统在许多传统行业中都有体现, 并且已经产生或正在产生许多安全问题, 造成了各方面的损失, 为此诞生了多种有效的管理方法, 以提高供应链的效率, 并应对来自供应链的风险。软件供应链的组织结构与传统供应链有很高的相似性, 使得通过管理手段缓解软件供应链严峻的安全问题成为可能, 并且可以借鉴其他行业现有的方法。许多大型软件供应商和企业级用户通过自身的实践, 总结出一些管理软件供应链的重要流程、工具和技术手段, 但是软件供应链上频发的由于缺乏管理引起的安全事件使得安全人员和企业的管理者有理由认为这些管理方式并没有得到彻底地执行, 或者管理方式本身存在问题。如何探索更有效且可行的软件供应链管理方法, 设计相关流程和工具, 制定评价指标, 并以技术标准的形式推行, 需要研究人员、产业界和相关组织机构的共同努力。

5 未来的研究方向

根据第四章中我们提出的软件供应链安全所面临的六大现实挑战, 可以指出四个未来的研究方向:

1) 软件安全质量的检测与控制

对于软件供应商而言, 软件供应链在降低自身软件开发工作量的同时, 会向最终软件产品中引入来自上游的软件缺陷和漏洞, 并扩大了单个软件缺陷和漏洞可能带来的影响。且近年来存在的软件开源化趋势, 导致软件供应链的开源化, 使得软件的质量难以控制, 并增加了供应链的暴露程度。软件供应链与其开源化的趋势使得对软件的安全质量的控制更加重要, 对软件漏洞挖掘和恶意软件识别等传

统技术手段提出了更高要求, 需要进一步地研究。

2) 软件供应链各类渠道的安全防范

在软件供应链这一系统中, 存在着各类软件代码、组件、补丁、成品由软件供应链的上游向下游转移的渠道。由于软件行业具有的互联网特性, 许多关键渠道的运行依赖互联网, 使得这些渠道的安全受到网络攻击的威胁, 在被攻击后可以成为网络攻击向供应链下游扩散的渠道, 同时渠道本身也有怀有恶意的可能性。这其中负责软件交付的集中式软件分发渠道直接影响大规模的用户, 显得尤为重要。如何针对软件供应链上的各类渠道设计安全防护措施, 并使得软件供应商和用户能够识别具有恶意的渠道, 需要得到研究人员的重视。

3) 供应链攻击的防护、检测和响应

软件供应链关系着软件产业的命脉, 其中流动着大量的经济利益与敏感信息。且软件供应链作为一个复杂而庞大的系统, 不可避免地有部分节点是系统的弱点, 并被暴露在攻击者面前。近年来软件供应链安全事件的频发一定程度上体现了攻击者对于攻击软件供应链的青睐, 以及供应链攻击相较于传统攻击的特异性。为了应对来自供应链的攻击, 如何针对这一攻击类型设计系统防护措施, 如何检测供应链攻击并做出响应需要更深入的研究。

4) 软件供应链风险的管理流程

软件供应链是一个复杂的系统, 具有多个环节, 站在位于供应链最下游的用户角度, 需要承受来自软件供应链的大量风险。作为大型组织、机构的用户尽管可能对供应链的上游鞭长莫及, 但是有一定的应对风险的能力, 也发展出了应对供应链风险的管理流程。但是作为个体的用户很难有应对风险的能力, 因而需要在供应链上游的软件供应商和各类渠道处发展出一套管理软件供应链风险的手段。目前这方面的研究尚不充分, 需要研究人员设计一套在保障供应链安全的基础上高效且可行的管理流程, 并通过国家和行业层面予以推行[35]。

6 总结

近年来软件供应链安全事件频发, 引发了学术界和产业界对于软件供应链安全问题的关注。本文通过典型的软件供应链安全事件对软件供应链的概念进行了讨论, 并通过调研本世纪以来网络与系统安全领域的研究成果, 总结了软件供应链安全的发展历程。然后从软件供应链的结构, 解决软件供应链安全问题的技术手段和管理手段三个方面介绍了软件供应链安全的研究现状, 并据此提出了软件供应链安全所面临的现实挑战。最后, 结合软件供应链安全的研究现状和现实挑战, 对未来软件供应链安全领域的研究进行了展望, 指出了软件安全质量的检测与控制, 软件供应链各类渠道的安全防范, 供应链攻击的防护、检测和响应, 软件供应链风险的管理流程这四点未来可能的研究方向。

致 谢 本课题得到国家重点研发计划基金资助项目(No.2016YFB0800700), 国家自然科学基金资助项目(No.61572460, No.61272481), 信息安全国家重点实验室的开放课题基金资助项目(No.2017-ZD-01)以及国家发改委信息安全专项基金资助项目(No.(2012)1424)资助。

参考文献

[1] Xiooxing, B. Xie, P. Yu, T. Zhang, L. Bu, and X. Li, “Software Development Methods: Review and Outlook,” *Journal of Software*, vol.30, no.1, pp.3-21, 2019.
(马晓星, 刘譞哲, 谢冰, 余萍, 张天, 卜磊, 李宣东, “软件开法方法发展回顾与展望”, *软件学报*, 2019, 30(1):3-21。)

[2] A. Abran, J. Moore, P. Bourque, et al. Guide to the software engineering body of knowledge[M]. IEEE Press, 2001.

[3] Nielson F, Nielson H R, Hankin C. Principles of Program Analysis[M]. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999.

[4] Z. Durumeric, J. Kasten, D. Adrian, The Matter of Heartbleed[C]. *Internet Measurement Conference(IMC’14)*, 2014:475-488.

[5] Bernstein D J, Lange T, Niederhagen R. Dual EC: A Standardized back Door[J]. *Journal of Neurosurgery Spine*, 2016, 6(3): 256-281.

[6] Sophia, “China internet website security report of the year 2015,” *China Information Security*, no.2, pp.50-52(in Chinese), 2016.
(Sophia, “2015 年度中国互联网站安全报告 安全漏洞频发 网络攻击行为加剧”, *信息安全与通信保密*, 2016(2):50-52。)

[7] (Sophia. 2015 年度中国互联网站安全报告安全 漏洞频发 网络攻击行为加剧[J]. 信息安全与通信保密, 2016, 14(2): 50-52.)

[8] L. Bilge, T. Dumitras, Before we Knew it:an Empirical Study of Zero-Day Attacks in the Real World[C]. *Acm Conference on Computer & Communications Security(CCS’12)*, 2012:833-844.

[9] B. Cui, Software Supply Chain Security Faces the Challenge of Open Source Software[J]. *China Information Security*, 2018, 107(11):71-72.
(崔宝江, 软件供应链安全面临软件开源化的挑战[J], *中国信息安全*, 2018, 107(11):71-72.)

[10] T. Hang, Software supply chain security risk management is a long way to go[J]. *China Information Security*, 2018,107(11):63-65.
(杭特, 软件供应链安全风险管控,任重而道远[J]. *中国信息安全*, 2018, 107(11):63-65.)

[11] V. Rajlich, K. Bennett, A Staged Model for the Software Life Cycle[J]. *Computer*, 2000, 33(7):66-71.

[12] I. Kozlenkova, G. Hult, D. Lund, et al. The Role of Marketing Channels in Supply Chain Management[J]. *Journal of Retailing*, 2015,91(4): 586-609.

[13] I. Krsul, Software vulnerability analysis[M]. Purdue University West Lafayette, 1998.

[14] Younis A A, Malaiya Y K, Ray I. Using Attack Surface Entry Points and Reachability Analysis to Assess the Risk of Software Vulnerability Exploitability[C]. *2014 IEEE 15th International Symposium on High-Assurance Systems Engineering*, January 9-11, 2014. Miami Beach, FL, USA. Piscataway, NJ: IEEE, 2014: 1-8.

[15] J. Holdsworth, Software Process Design[M]. MacGraw-Hill, 1995.

[16] "Windows Defender Atp thwarts Operation Wilysupply Software-supply Chain Cyberattack," Microsoft Defender Research team, https://www.microsoft.com/security/blog/2017/05/04/windows-defender-atp-thwarts-operation-wilysupply-software-supply-chain-cyberattack/.

[17] Ellison R J, Woody C. Supply-Chain Risk Management: Incorporating Security into Software Development[C]. *2010 43rd Hawaii International Conference on System Sciences*, January 5-8, 2010. Honolulu, Hawaii, USA. Piscataway, NJ: IEEE, 2010: 1-10.

[18] S. Boyson, H. Rossman, Developing a cyber-supply chain assurance reference model. Technical report, Maryland: Supply Chain Management Center, Robert H. Smith School of Business University of Maryland, 2009.

[19] B. Hamilton, Managing risk in global ICT supply chains: Best practices and standards for acquiring ICT. Technical report, McLean, Virginia: Booz Allen Hamilton, 2012.

[20] G. Ni, X. Chen, Y. Shang, et al. Research on Foreign ICT Supply Chain Security Management with Suggestions[J]. *Engineering Sciences*, 201618(6):104-109.
(倪光南, 陈晓桦, 尚燕敏, 等. 国外 ICT 供应链安全管理研究及建议[J]. 中国工程科学, 2016, 18(6): 104-109.)

[21] Thompson K. Reflections on Trusting Trust[J]. *Communications of the Acm*, 1984,27(8):761-763.

[22] L. Urciuoli, S. Mohanty, J. Hintsa, The resilience of energy supply chains: a multiple case study approach on oil and gas supply chains to Europe[J]. Supply Chain Management, 2014,19(1):46-63.

[23] Confirmed US-Israel Created Stuxnet Lost Control of It, Artechnica, https://arstechnica.com/tech-policy/2012/06/confirmed-us-israel-created-stuxnet-lost-control-of-it/.

[24] Warren M, Hutchinson W. Cyber Attacks Against Supply Chain Management Systems: A Short Note[J]. *International Journal of Physical Distribution & Logistics Management*, 2000, 30(7/8): 710-716.

[25] Lipner S. The Trustworthy Computing Security Development Lifecycle[C]. *20th Annual Computer Security Applications Conference*, Tucson, AZ, USA. Piscataway, NJ: IEEE, 2004.41.: 2-13.

[26] Zhang S K, Ma S, Gao Q, et al. Risks and causes analysis of software supply chain security[J]. *China Information Security*, 2018(11): 48-50.
(张世琨, 马森, 高庆, 等. 软件供应链安全的风险和成因分析[J]. *中国信息安全*, 2018(11): 48-50.)

[27] Gui X L, Liu J, Chi M C, et al. Analysis of Malware Application Based on Massive Network Traffic[J]. *China Communications*, 2016, 13(8): 209-221.

[28] Protecting the Software Supply Chain: Deep Insights into the CCleaner Backdoor. Crowdstrike, https://www.crowdstrike.com/blog/protecting-software-supply-chain-deep-insights-ccleaner-backdoor/.

[29] ShadowPad in corporate networks. Securelist, https://securelist.com/shadowpad-in-corporate-networks/81432/.

[30] Tech Firms Unite to Neutralize WireX Android Botnet. Securityweek, https://www.securityweek.com/tech-firms-unite-neutralize-wirex-android-botnet/.

[31] Young A, Yung M. Cryptovirology: Extortion-based Security Threats and Countermeasures[C]. *Proceedings 1996 IEEE Symposium on Security and Privacy*, Oakland, CA, USA. IEEE Comput. Soc. Press, 1996.502676.: 129-129.

[32] This Ukrainian Company Is Likely Behind the Ransomware Wave. Fortune, http://fortune.com/2017/06/27/petya-ransomware-ukraine-medoc/.

[33] Supply Chain Attatcks of Software. 360, https://ti.360.net/blog/ articles/supply-chain-attacks-of-software/.

[34] Ma J X. Protect the risk of software supply chain security and maintain the security of network space [J]. *China Information Security*, 2018(11): 55-57.
(马金鑫. 防范软件供应链安全风险 维护网络空间安全[J]. *中国信息安全*, 2018(11): 55-57.)

[35] W. Zou, W. Huo, Q. Liu, Ensuring software supply chain security is a system engineering[J]. *China Information Security*, 2018,107(11):60-62.
(邹维, 霍玮, 刘奇旭. 确保软件供应链安全是一项系统工程[J]. *中国信息安全*, 2018, 107(11):60-62.)

[36] Yang Z M, Zhang Z G. The Study on Resolutions of STRIDE Threat Model[C]. *2007 First IEEE International Symposium on Information Technologies and Applications in Education*, November 23-25, 2007. Kunming, China. Piscataway, NJ: IEEE, 2007: 271-293.

[37] Zhou Z F. *Research on Software Supply Chain Contamination Mechanism and Defense Technology*[D]. Beijing: Beijing University of Posts and Telecom, 2018.
(周振飞. 软件供应链污染机理与防御研究[D]. 北京: 北京邮电大学, 2018.)

[38] Jamshidi M. System of Systems Engineering - New Challenges for the 21st Century[J]. *IEEE Aerospace and Electronic Systems Magazine*, 2008, 23(5): 4-19.

[39] Fang S S. Us to develop information and communication technology[J]. *Information Security and Communications Privacy*, 2018, 16(6): 85-89.
(方师师. 美拟制定信息通信技术“供应链风险管理国家战略”以应对来自中国的供应链漏洞[J]. *信息安全与通信保密*, 2018, 16(6): 85-89.)

[40] Alberts C J, Dorofee A J, Creel R, et al. A Systemic Approach for Assessing Software Supply-Chain Risk[C]. *2011 44th Hawaii International Conference on System Sciences*, January 4-7, 2011. Kauai, HI. Piscataway, NJ: IEEE, 2011: 1-8.

[41] Sneed H M. Integrating Legacy Software into a Service Oriented Architecture[C]. *Conference on Software Maintenance and Reengineering (CSMR'06)*, March 22-24, 2006. Bari, Italy. Piscataway, NJ: IEEE, 2006: 3-14.

[42] S. Kowalski. IT Insecurity: A Multi-discipline Inquiry[D]. Stockholm: Univ. Stockholm and Royal Inst. Technology, 1994.

[43] Sabbagh B A, Kowalski S. A Socio-technical Framework for Threat Modeling a Software Supply Chain[J]. *IEEE Security & Privacy*, 2015, 13(4): 30-39.

[44] [44] J. Liu, P. Su, et al Software and Cyber Security-A Survey[J]. *Journal of Software*, 2018,29(1):42-68.
(刘剑, 苏璞睿, 杨珉, 等. 软件与网络安全研究综述[J]. *软件学报*, 2018, 29(1):42-68.)

[45] Lawton G. Open Source Security: Opportunity or Oxymoron?[J]. *Computer*, 2002, 35(3): 18-21.

[46] Ohlsson M C, von Mayrhauser A, McGuire B, et al. Code Decay Analysis of Legacy Software through Successive Releases[C]. *1999 IEEE Aerospace Conference. Proceedings (Cat. No.99TH8403)*, March 13, 1999. Snowmass at Aspen, CO, USA. Piscataway, NJ: IEEE, 1999: 69-81.

[47] V. Livshits, M. Lam. Finding Security Vulnerabilities in Java Applications with Static Analysis[C]. *Conference on Usenix Security Symposium(SSYM'05)*, 2005: 18-18.

[48] Godefroid P, Kiezun A, Levin M Y. Grammar-based Whitebox Fuzzing[C]. *Proceedings of the 2008 ACM SIGPLAN conference on Programming language design and implementation - PLDI '08*, June 7-13, 2008. Tucson, AZ, USA. New York, USA: ACM Press, 2008: 206-215.

[49] Grieco G, Grinblat G L, Uzal L, et al. Toward Large-Scale Vulnerability Discovery Using Machine Learning[C]. *Proceedings of the Sixth ACM on Conference on Data and Application Security and Privacy - CODASPY '16*, March 9-11, 2016. New Orleans, Louisiana, USA. New York, USA: ACM Press, 2016: 85-96.

[50] G. Portokalidis, A. Slowinska, H. Bos. Argos:an emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation[C]. *Acm Sigops/Eurosys European Conference on Computer Systems*(EuroSys' 06), 2006: 15-27.

[51] J. Xu, D. Mu, P. Chen, et al. CREDAL: Towards Locating a Memory Corruption Vulnerability with Your Core Dump[C]. *Acm Sigsac Conference on Computer & Communications Security(CCS' 16)*, 2016: 529-540.

[52] Nguyen H D T, Qi D W, Roychoudhury A, et al. SemFix: Program Repair Via Semantic Analysis[C]. *2013 35th International Conference on Software Engineering (ICSE)*, May 18-26, 2013. San Francisco, CA, USA. Piscataway, NJ: IEEE, 2013: 772-781.

[53] Le Goues C, Nguyen T, Forrest S, et al. GenProg: A Generic Method for Automatic Software Repair[J]. *IEEE Transactions on Software Engineering*, 2012, 38(1): 54-72.

[54] Y. Ma, Constructing Supply Chains in Open Source Software[C]. *International Conference on Predictive Models and Data Analytics in Software Engineering(ICSE'18)*, 2018: 458-459.

[55] Tecent Security Xuanwu Lab, https://xlab.tencent.com/.

[56] Langweg H, Snekkenes E. A Classification of Malicious Software Attacks[C]. *IEEE International Conference on Performance, Computing, and Communications, 2004*, Phoenix, AZ. Piscataway, NJ: IEEE, 2004: 827-832.

[57] Linn C, Debray S. Obfuscation of Executable Code to Improve Resistance to Static Disassembly[C]. *Proceedings of the 10th ACM conference on Computer and communication security - CCS '03*, October 27-30, 2003. Washington D.C., USA. New York, USA: ACM Press, 2003: 290-299.

[58] W. Xu, F. Zhang, S. Zhu, JStill: Mostly static detection of obfuscated malicious javascript code[C] *Acm Conference on Data & Application Security & Privacy(CODASPY'13)*, 2013: 117-128.

[59] Peng X X, Hu Z J, Gong T, et al. Research of Malicious Code in Automatic Unpacking[J]. *Netinfo Security*, 2014(5): 41-45.
(彭小详, 户振江, 龚涛, 等. 恶意代码自动脱壳技术研究[J]. *信息网络安全*, 2014(5): 41-45.)

[60] Chen H Q. Feature Selection in Malware Detection[J]. *Journal of University of Electronic Science and Technology of China*, 2009, 38(1): 53-56.
(陈洪泉. 恶意软件检测中的特征选择问题[J]. *电子科技大学学报*, 2009, 38(1): 53-56.)

[61] Sathyanarayan V S, Kohli P, Bruhadeshwar B. Signature Generation and Detection of Malware Families[M]. Information Security and Privacy. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018: 336-349.

[62] Shabtai A, Moskovitch R, Elovici Y, et al. Detection of Malicious Code by Applying Machine Learning Classifiers on Static Features: A State-of-the-art Survey[J]. *Information Security Technical Report*, 2009, 14(1): 16-29.

[63] Willems C, Holz T, Freiling F. Toward Automated Dynamic Malware Analysis Using CWSandbox[J]. *IEEE Security and Privacy Magazine*, 2007, 5(2): 32-39.

[64] Zhou W, Zhou Y J, Jiang X X, et al. Detecting Repackaged Smartphone Applications in Third-party Android Marketplaces[C]. *Proceedings of the second ACM conference on Data and Application Security and Privacy - CODASKY '12*, February 7-9, 2012. San Antonio, Texas, USA. New York, USA: ACM Press, 2012: 317-326.

[65] BinDiff, https://zynamics.com/bindiff/.

[66] Zhao G F, Chen Y, Wang X H. Research on the Web Front-end Hijacking and Defense Against HTTPS[J]. *Netinfo Security*, 2016(3): 15-20.
(赵国锋, 陈勇, 王新恒. 针对HTTPS的Web前端劫持及防御研究[J]. 信息网络安全, 2016(3): 15-20.)

[67] Janbeglou M, Zamani M, Ibrahim S. Redirecting Network Traffic Toward a Fake DNS Server on a LAN[C]. *2010 3rd International Conference on Computer Science and Information Technology*, July 9-11, 2010. Chengdu, China. Piscataway, NJ: IEEE, 2010: 221-230.

[68] Haataja K, Toivanen P. Two Practical Man-in-the-middle Attacks on Bluetooth Secure Simple Pairing and Countermeasures[J]. *IEEE Transactions on Wireless Communications*, 2010, 9(1): 384-392.

[69] Meyer U, Wetzel S. A Man-in-the-middle Attack on UMTS[C]. *Proceedings of the 2004 ACM workshop on Wireless security - WiSe '04*, October 1, 2004. Philadelphia, PA, USA. New York, USA: ACM Press, 2004: 90-97.

[70] Callegati F, Cerroni W, Ramilli M. Man-in-the-Middle Attack to the HTTPS Protocol[J]. *IEEE Security & Privacy Magazine*, 2009, 7(1): 78-81.

[71] Asokan N. Man-in-the-Middle in Tunnelled Authentication Protocols[M]. Security Protocols. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005: 42-48.

[72] Oppliger R, Hauser R, Basin D. SSL/TLS Session-aware User Authentication-Or how to Effectively Thwart the Man-in-the-middle[J]. *Computer Communications*, 2006, 29(12): 2238-2246.

[73] Alicherry M, Keromytis A D. DoubleCheck: Multi-path Verification Against Man-in-the-middle Attacks[C]. *2009 IEEE Symposium on Computers and Communications*, July 5-8, 2009. Sousse, Tunisia. Piscataway, NJ: IEEE, 2009: 557-563.

[74] R. Gill, J. Smith J, A. Clark. Experiences in passively detecting session hijacking attacks in IEEE 802.11 networks[C]. *Australasian Workshops on Grid Computing & E-research(ACSW Frontiers'06)*, 2006: 221-230.

[75] Schmoyer T R, Lim Y X, Owen H L. Wireless Intrusion Detection and Response: A Classic Study Using Main-in-the-middle Attack[C]. *2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No.04TH8733)*, Atlanta, GA, USA. Piscataway, NJ: IEEE, 2004: 883-888.

[76] Jiang J G, Wang J Z, Kong B, et al. On the Survey of Network Attack Source Traceback[J]. *Journal of Cyber Security*, 2018, 3(1): 111-131.
(姜建国, 王继志, 孔斌, 等. 网络攻击源追踪技术研究综述[J]. *信息安全学报*, 2018, 3(1): 111-131.)

[77] Wagner C, François J, State R, et al. Machine Learning Approach for IP-Flow Record Anomaly Detection[M]. NETWORKING 2011. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011: 28-39.

[78] Enck W, Ongtang M, McDaniel P. Understanding Android Security[J]. *IEEE Security & Privacy Magazine*, 2009, 7(1): 50-57.

[79] Davi L, Dmitrienko A, Sadeghi A R, et al. Privilege Escalation Attacks on Android[M]. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011: 346-360.

[80] P. Amthor, W. Kuhnhauser, A. Polck. Analyzing Integrity Protection in the SELinux Example Policy[C]. *International Conference on Network and System Security(NSS'11)*, 2011: 208-215.

[81] Shabtai A, Fledel Y, Elovici Y. Securing Android-Powered Mobile Devices Using SELinux[J]. *IEEE Security & Privacy Magazine*, 2010, 8(3): 36-44.

[82] Iliev A, Smith S W. Protecting Client Privacy with Trusted Computing at the Server[J]. *IEEE Security and Privacy Magazine*, 2005, 3(2): 20-28.

[83] Wei Q, Wei T, Wang J J. Evolution of Exploitation and Exploit Mitigation[J]. *Journal of Tsinghua University(Science and Technology)*, 2011, 51(10): 1274-1280.
(魏强, 韦韬, 王嘉捷. 软件漏洞利用缓解及其对抗技术演化[J]. *清华大学学报(自然科学版)*, 2011, 51(10): 1274-1280.)

[84] Chen G Q. Supply Chain management[J]. *China Soft Science*, 1999(10): 101-104.
(陈国权. 供应链管理[J]. *中国软科学*, 1999(10): 101-104.)

[85] Luszcz J. How Maverick Developers can Create Risk in the Software and IoT Supply Chain[J]. *Network Security*, 2017, 2017(8): 5-7.

[86] Zhu G B, Chen J. The status and countermeasure suggestion of software supply chain security[J]. *China Information Security*, 2018(11): 44-47.
(祝国邦, 陈洁. 软件供应链安全现状与对策建议[J]. 中国信息安全, 2018(11): 44-47.)

[87] Zhou Y J, Qiu W H, Wang Z R. A Review on Supply Chain Risk Management[J]. *Systems Engineering*, 2006, 24(3): 1-7.
(周艳菊, 邱莞华, 王宗润. 供应链风险管理研究进展的综述与分析[J]. *系统工程*, 2006, 24(3): 1-7.)

[88] Hallikas J, Karvonen I, Pulkkinen U, et al. Risk Management Processes in Supplier Networks[J]. *International Journal of Production Economics*, 2004, 90(1): 47-58.

[89] Prater E, Biehl M, Smith M A. International Supply Chain Agility - Tradeoffs between Flexibility and Uncertainty[J]. *International Journal of Operations & Production Management*, 2001, 21(5/6): 823-839.

[90] Gautham R N. Designing Software Project Management Models Based on Supply Chain Quality Assurance Practices[C]. *2009 WRI World Congress on Computer Science and Information Engineering*, March 31-April 2, 2009. Los Angeles, California USA. Piscataway, NJ: IEEE, 2009: 659-663.

[91] Bartol N. Cyber Supply Chain Security Practices DNA – Filling in the Puzzle Using a Diverse Set of Disciplines[J]. *Technovation*, 2014, 34(7): 354-361.

[92] R. Charette. Applications Strategies for Risk Analysis[D] McGraw-Hill, 1990.

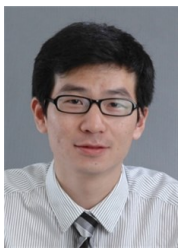
[93] Reddy D. Criticality Analysis and the Supply Chain: Leveraging Representational Assurance[J]. *Technovation*, 2014, 34(7): 362-379.

[94] Chess Project, http://www.chess-project.org/.

[95] Treemap, https://www.treemap.com/.

[96] Boyson S. Cyber Supply Chain Risk Management: Revolutionizing the Strategic Control of Critical IT Systems[J]. *Technovation*, 2014, 34(7): 342-353.

何熙巽 于2018年在清华大学计算机科学与技术专业获得学士学位, 现在中国科学院大学信息安全专业攻读硕士学位。主要研究方向为物联网安全。Email: hexx@nipc.org.cn



刘奇旭 于2011年在中国科学院研究生院信息安全专业获得博士学位。现任中国科学院信息工程研究所副研究员、中国科学院大学网络空间安全学院副教授。主要研究方向为网络攻防技术、网络安全评测。Email: liuqixu@iie.ac.cn

张玉清 于 2000 年在西安电子科技大学获得博士学位。现任中国科学院大学教授, 博士生导师。主要研究方向为网路与信息系统安全。Email: zhangyq@nipc.org.cn