代码疫苗技术在DevSecOps体系下的实践

Practice of Code Vaccine Technology Under DevSecOps System

董毅/DONG Yi
（北京安普诺信息技术有限公司，中国 北京 100193）
(Beijing Anpro Information Technology Co. ,Ltd. , Beijing 100193, China)

DOI:10.12142/ZTETJ.202206008
网络出版地址：https://kns.cnki.net/kcms/detail/34.1228.TN.20221221.1419.001.html
网络出版日期：2022-12-23
收稿日期：2022-10-17

摘要：安全工具在帮助开发人员构建安全软件方面发挥着至关重要的作用。然而，在不影响DevOps部署速度或交付频率的情况下，引入和充分利用安全工具是具有挑战性的。通过分析当下传统安全工具存在的问题，提出了目前已成功应用于交互式应用安全测试（IAST）工具和运行时应用自保护（RASP）工具的代码疫苗技术。阐述了代码疫苗技术在DevSecOps体系下的实践，并以Log4j2组件的远程代码执行漏洞的防护为例，梳理了代码疫苗技术的防护流程。

关键词：DevSecOps；代码疫苗技术；RASP；IAST；DevOps

Abstract: Security tools play a vital role in helping developers build secure software. However, it is challenging to introduce and fully utilize security tools without compromising the speed or frequency of DevOps deployments. By analyzing the current problems of traditional security tools, the code vaccine technology covering interactive application security testing (IAST) technology and runtime application self-protection (RASP) technology is proposed. The practice of code vaccine technology under the DevSecOps system is expounded. Taking the remote code execution vulnerability protection of Log4j2 component as an example, the protection process of code vaccine technology is summarized.

Keywords: DevSecOps; code vaccine technology; RASP; IAST; DevOps

DevOps的兴起打破了开发与运营之间的壁垒，帮助企业快速地将新产品或服务推向市场。然而，随着交付效率的提升，传统的“外挂式”安全和合规检测工具无法跟上快速迭代的步伐，已成为开发运营效能提升的瓶颈。内生安全的概念在DevOps实践中随即被提出[1]。DevSecOps的核心理念是将安全嵌入DevOps工作流程中，通过在软件生命周期的不同阶段加入相对应的安全工具，赋能软件内生的安全性，有效保障软件系统的正常上线发布和稳定运行[2]。

传统的安全检测工具，如静态应用程序安全测试（SAST）工具[3]、动态应用程序安全测试（DAST）工具[4]，经常会输出一些不准确的结果（误报或漏报），因此需要工程师手动评估输出结果的准确性。这与DevSecOps的自动化理念背道而驰。此外，防火墙、Web应用防火墙（WAF）、下一代WAF、入侵检测系统（IDS）、入侵防御系统（IPS）等都是基于边界防护手段[5]。随着“云大物移智”等技术的到来，“无界防护”需求的产生给企业的安全防护带来了新的考验。基于边界的传统网络安全防护远不能应对当下的网络攻击[6]。

新时代下的软件安全开发运营既要实现“安全左移”，又要实现“敏捷右移”。这一理念催生出代码疫苗技术。代码疫苗技术是指把某项技术像疫苗一样注入应用服务器内部，在内部清晰看到解析后的流量，感知业务运行过程的情境上下文。该技术可被应用到软件成分分析（SCA）、交互式应用安全测试（IAST）、运行时应用自保护（RASP）以及入侵和攻击模拟（BAS）等工具中[7]。这样一来，系统既能诊断应用自身存在的漏洞和缺陷，也能积极防御外部危险，进行自主检测和响应。本文主要分析当前传统应用安全防护面临的问题，阐述当前已成功应用于IAST和RASP工具中的代码疫苗技术，并对代码疫苗技术在DevSecOps体系下的实践展开深入研究。

1 代码疫苗技术相关理论概述

1.1 研究背景及现状

国际著名咨询调查机构Gartner制定了一个较为全面的DevSecOps工具链实践清单，并将DevSecOps生命周期分成计划、创建、验证、预发布、发布、预防、检测、响应、预测和改进10个环节。其中，每个环节都具有相应的安全工

具和安全活动，如图1所示。验证阶段应用的IAST技术来自2012年Gartner提出的一种新的应用程序安全测试方案。检测响应阶段应用的RASP技术是2014年被Gartner在应用安全报告里列为应用安全领域的关键技术。IAST和RASP技术已连续数年被Gartner列入“十大安全技术”[8]。

DevSecOps的根本目标是使企业组织内部拥有不断进化的安全能力，让安全变成一种内在属性嵌入企业数字化应用的全生命周期中。这样可以使企业具备持续安全的开发和运营能力，而不是单纯地保障业务和业务系统的安全[2]。在DevSecOps中，安全工具起到了关键作用。安全活动工具化是组织在DevOps基础上实现安全敏捷化的前提条件。只有将具体的安全活动工具化，追求安全的敏捷化，安全的“敏捷的价值交付”才能成为可能。建立“端到端”的、完整的安全工具链是DevSecOps安全活动工具化的内在需求和关键目标。近年来，用于改进软件构建和交付方式的工具越来越成熟，如持续集成（CI）/持续部署（CD）平台、微服务、容器等，但针对安全的工具开发还相对滞后。DevSecOps每个阶段集成了不同的安全工具。每个安全工具只能发现应用程序中存在的所有漏洞的部分子集。检测出的结果在很大程度上是不同的，甚至是混乱的。另外，通过人工手动对检测出来的漏洞信息进行筛查整理的方式，会影响开发效率，致使DevSecOps难以实现自动化。

针对上述问题，本文提出当前已成功应用于IAST工具和RASP工具的代码疫苗技术。其中，相较于SAST白盒安全测试技术和DAST黑盒安全测试技术，IAST技术具有更高的漏洞检出率和更多的适配场景，同时也更加适用于目前流行的DevOps场景[9]。相较于传统的WAF或IDS在流量层的检测，RASP技术更多的是与应用耦合在一起，通过运行时插桩技术，对应用的运行环境进行检测。这样的方式有助于拦截从应用程序到系统的所有调用，确保应用程序的安全，从而实时检测和阻断各式各样的安全攻击。在2021年底爆发的“核弹级”Apache Log4j2安全漏洞事件[10]后，RASP的能力得到了更为广泛的认可。

1.2 代码疫苗技术基本概述

代码疫苗技术是一种能够通过运行时插桩技术进行风险自发现及威胁自免疫的新一代安全技术。该技术可将IAST技术的漏洞检测能力与RASP技术的攻击防护能力进行合并，形成一个统一的IAST和RASP探针，实现IAST技术与RASP技术的集成，发挥各自技术的最大优势，进而形成统一的从漏洞检测到漏洞防护全生命周期的检测与防护解决方案，如图2所示。代码疫苗技术得核心内涵主要包含4个方面：

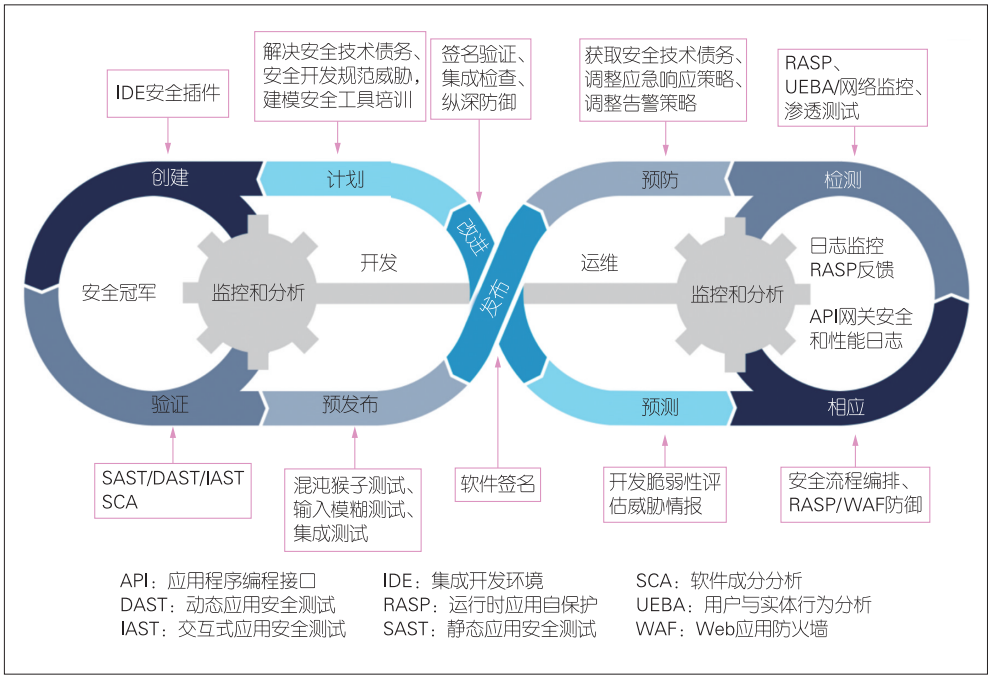
（1）不需要代码安全专家逐行分析源代码；

（2）不需要对原有代码逻辑进行修改调整；

（3）不需要维护复杂流量过滤策略及规则；

（4）实时监测应用程序中由第三方组件引入的风险。

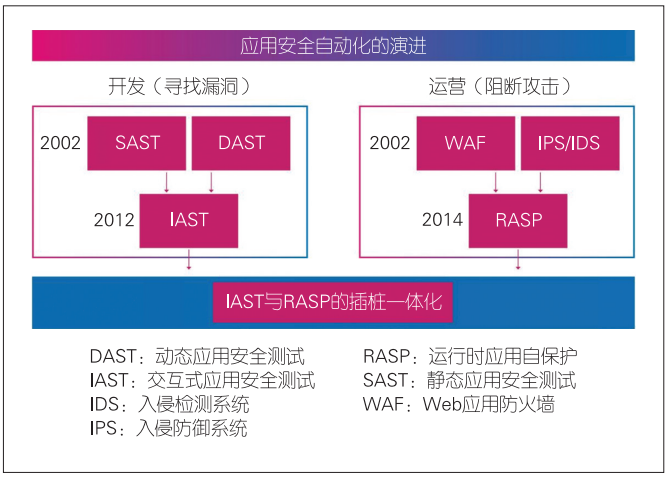
与医学界对疫苗的定义相似，代码疫苗技术并不是传统的外挂式安全技术，其侧重点是在软件应用的内部构筑安全屏障，以便搭建更加有效的内生安全防御体系。



▲图1 Gartner DevSecOps工具链

1.3 代码疫苗技术实现原理

无论是IAST技术还是RASP技术，都依赖以代码疫苗技术为内核的运行时插桩组件。在应用层运行时插桩是通过应用启动后替换函数体或在函数前后插入检测代码来实现的。通过插桩代码，记录应用程序运行时的堆栈数据可获得应用运行在关键点的数据信息。需要注意的是，由于不同语言间存在运行时的环境差异，因此不同语言实现插桩的方式也有所区别。以Java为例，由于Java拥有Instrument的特性，因此在



▲图2 统一的应用安全探针

类加载的过程中，需要首先对所关注的关键类与方法的字节码进行修改，然后才能够达到插入检测逻辑的目的。

运行时插桩技术和流量代理技术是IAST最具代表性的两种技术。其中，流量代理技术通过对镜像流量、日志等数据进行重放和分析来达到检测目的，对研发测试人员等完全透明，无流程侵入，也不依赖应用编程语言，如图3所示。

运行时插桩技术通过插桩帮助研发测试人员快速完成业务安全测试，精准定位漏洞细节并进行修复指导。运行时插桩技术又分为交互式缺陷定位（主动式插桩技术）和动态污点追踪（被动式插桩技术）。其中，动态污点分析技术能够基于运行时插桩跟踪外部可控数据对应用的影响，分析外部数据在应用内部的流转情况，从而确定应用是否存在漏洞。动态IAST拥有无重放数据、无脏数据、可应对加密签名接口、可适配复杂场景等优点，因此目前的适用面比较广泛。

在动态污点分析技术中，污点传播过程包括污点输入、污点传播、污点汇集3个阶段，如图4所示。

• 污点输入阶段。所有外部数据都被默认为不可信数据，因此需要在外部数据进入应用的时候，对其添加污点标记。

• 污点传播阶段。此阶段的主要目标为跟踪污点数据的传播过程。由于外部数据在进入应用程序时已被污点标记，因此当被标记数据进行运算或字符串拼接等操作时，所产生的新数据也将会携带污点标记。

• 污点汇集阶段。此阶段需要对可能触发漏洞的函数进行关注，即确定携带污点标记的数据是否会汇聚到敏感函数之上。若该过程发生，则意味着在应用程序中的这些函数执行流程中可能存在着漏洞。在污点传播阶段，如果携带污点标记的数据遇到清洁函数，并被成功执行过滤操作或其他安全操作，则该数据所携带的污点标记将被消除，以此来确认这条链路的安全性。

RASP的核心是通过插桩技术将防护逻辑与防护功能注入应用程序，深入应用运行时的环境内部，通过分析与了解上下文数据流及事件流，依据自有的安全规则列表，检测和防护无法预见的安全威胁与攻击事件（如0day攻击）。这种运作模式使得RASP能够解决一些难题，包括WAF所存在的检测规则与功能无法对应、服务端防御方式无法知晓、变形及未知威胁防御乏力、微服务场景难适配等。

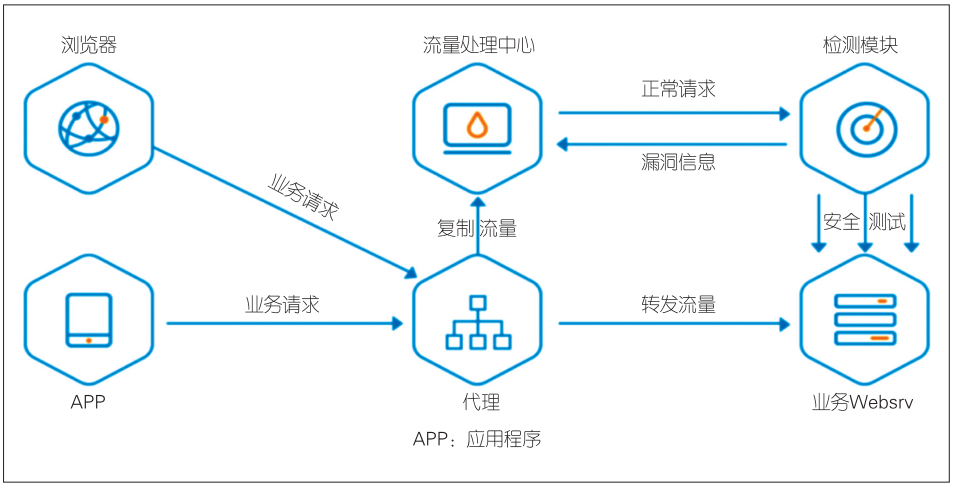
RASP主要获取以下4种运行时的上下文数据：

（1）超文本传输协议（HTTP）请求与响应数据，以及各种远程调用协议（RPC），例如Dubbo的请求与响应数据、gRPC等各式RPC框架。

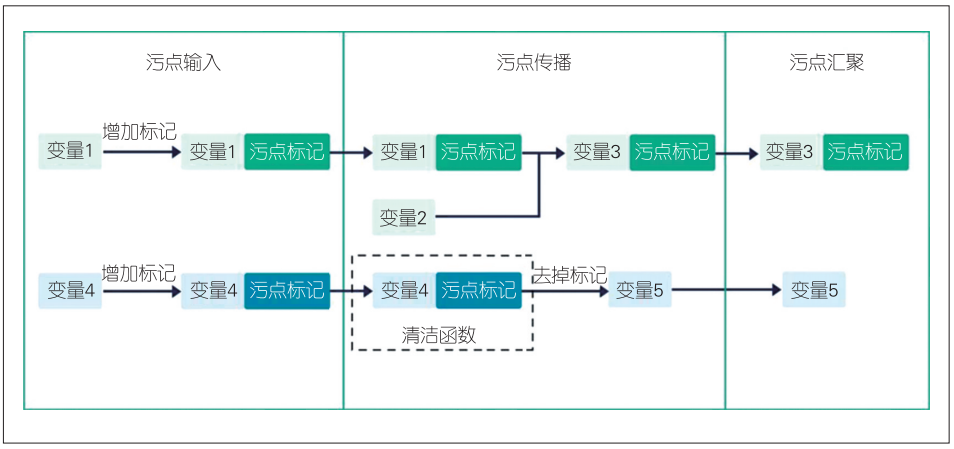
（2）所关注函数的执行数据，包括动态运行时函数所接收到的完整参数、调用函数的对象实例以及函数执行的返回值。获取运行时过程中的函数整体执行状态，就能够判断运行函数在执行过程中是否存在所关注的数据。

（3）函数执行过程中的调用栈。获取完整的函数调用栈不仅有助于进行漏洞分析与攻击分析，还有助于分析攻击者的行为。常用的一些反序列化的攻击手段都可以通过函数调用栈进行分析。

（4）应用配置信息。获取应用的各类安全配置、代码内属性配置等信息，可以完整得知该应用是否执行安全策略。



▲图3 流量代理模式



▲图4 污点数据流

通过对上述4种运行时的上下文数据进行分析和运用，RASP可实现应用运行时的自我防护。根据采用的算法或者检测逻辑，基于运行时上下文的防护逻辑包括以下4类：

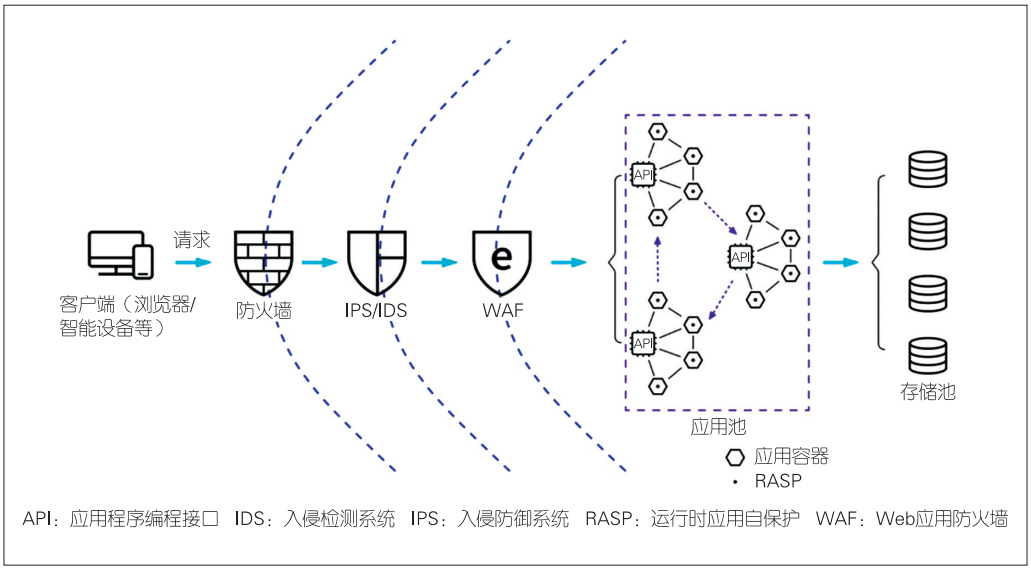
第1类是规则方式，即对获取的参数或者HTTP请求进行规则匹配。

第2类是基于词法的分析。由于RASP所获取的数据更加全面，RASP能够针对完整的输入信息（如SQL等）进行词法分析，以判断关键函数执行点上的数据是否存在异常。

第3类是行为及运行堆栈检测，该检测主要用来检测敏感函数的执行。例如，WebShell在植入系统后会通过变形等方式绕过检测，但在执行系统命令或文件操作的过程中，其必定会调用底层运行时的应用程序编程接口（API）。此时借助行为及运行堆栈分析能够获取执行调用的函数或函数调用栈信息。

第4类是应用运行配置检测，即对代码中的动态安全配置及其他配置检测。例如，当增加某些安全配置后，部分漏洞就无法被再利用了。这其中就包括预编译这类防范SQL注入的方式，以及XML外部实体注入（XXE）的关闭外部实体访问方式等。这样便能够完整地了解目前应用所存在的安全防护情况。

在整体防护体系中，RASP会与每个应用耦合，但其与WAF、IDS/IPS、防火墙等并不冲突。RASP适用于现代开发或应用架构，与应用、微服务是相伴相生的。因此，每个安全解决方案都是纵深防御体系中的一个环节。图5为RASP防护体系位置。



▲图5 RASP防护体系位置

2 代码疫苗技术在DevSecOps体系下的应用

代码疫苗技术可以应用到多个不同的场景中。相关应用场景主要分为以下4类：

（1）应对DevOps的检测防护一体化。探针是比较轻量级的，可以随流水线一同上线发布。在开发和测试环节，IAST可被用于漏洞检测。在上线后，RASP则会被开启进行漏洞防护，以实现全流程的检测防护一体化，提高DevOps的效率。

（2）红蓝对抗。在这一场景下，RASP充当高级漏洞攻击防护工具的角色。目前红蓝对抗会更多地应用0day、1day或一些未公开的漏洞利用程序（EXP）进行攻防。这对于传统的流量手段而言是难以防护的，而RASP有能力应对一些高级攻击。

（3）应对突发漏洞。RASP能够提供针对基于行为与调用栈位置威胁的检测。这可以在一定程度上缓解0day或1day攻击，为漏洞修复争取时间；也可以利用RASP提供的热补丁功能，通过一些简单配置，先进行第一波漏洞攻击的防护。

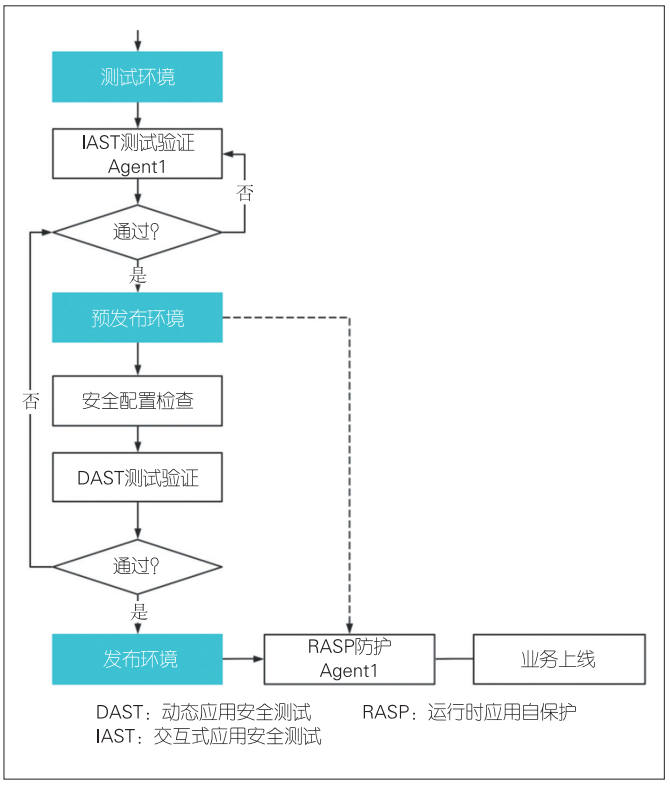
（4）应用上线的自免疫。在容器化的环境之中，将探针

和应用进行打包可使应用在上线之后能够自带攻击防护效果。

应对DevOps的检测防护一体化场景是代码疫苗技术应用最为重要的场景。代码疫苗技术实现安全工具链自动化，与DevOps的核心需求（高效和自动）高度一致。在开发流程中同步实现安全，确保应用安全上线，实现DevSecOps，不仅能为企业业务提供更加及时、可靠的服务支持，还能使企业在数字化转型中持续占据身位优势与竞争优势。代码疫苗技术与DevSecOps流水线集成流程如图6所示。系统在测试阶段采用IAST工具来检测识别漏洞，在预发布和发布阶段利用同一Agent获取的函数堆栈信息，并采用RASP技术对识别出来的漏洞进行热补丁修复，从而实现对应用程序的实时防护。

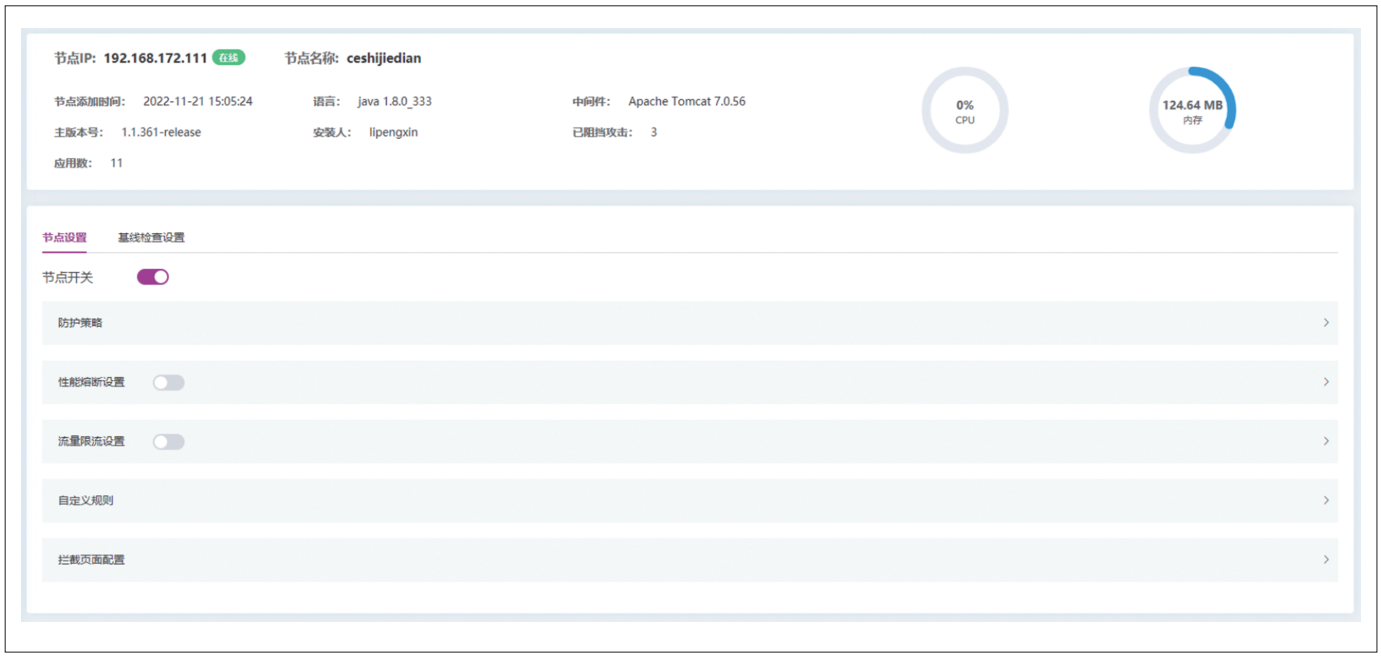
DevOps大多在容器虚拟化环境中进行软件的开发交付。IAST的安全融合可充分利用DevOps的自动化特性，在构建测试环境的同时引入代码疫苗技术插桩，即IAST和RASP的同一探针。这里我们使用修改容器配置文件（如DockerFile）的方式进行插桩。由于代码疫苗技术依赖软件的运行环境，在简单修改测试环境的环境变量或启动参数后，我们就可以应用该插桩对软件的运行时内存和数据流进行全面监控。当流水线运行到测试阶段时，软件需要进行全面的自动化功能测试，或者根据测试用例进行人工测试。对此，IAST可依托污点追踪完成软件每个功能点的安全漏洞风险和敏感数据泄露风险验证；RASP可依据同一Agent获取代码执行流程，对应用进行防护。

如图7和图8所示，代码疫苗技术成功应用到IAST工具和RASP工具中，在不影响企业现有开发流程的同时，实现了真正透明无感知的深度漏洞检测与防护。代码疫苗技术使业务和安全充分融合同时又相互解耦，使数字化业务出厂即带有默认安全能力，实现跨维检测、响应与防护，安全能力可编程、可扩展，与业务各自独立演进，让业务按照自身需求敏捷运转并进行迭代。



▲图6 代码疫苗技术与DevSecOps流水线集成流程图

▲图7 代码疫苗技术应用到交互式应用安全测试工具中



▲图8 代码疫苗技术应用到运行时应用自保护工具中

3 结束语

安全工具和软件交付工具之间的技术创新差距是DevOps安全性落后的主要原因之一。IAST与RASP技术的兴起可逐渐满足DevOps的快速交付需求，有助于应对DevOps面临的安全挑战。然而，在DevOps流程中仅仅使用合适的安全工具还远远不够。将安全成功地集成到DevOps中将对安全工具的集成提出更高要求。代码疫苗技术的出现实现了IAST技术与RASP技术的集成。该技术可以将探针布控在软件生命周期的各个环节，在应用程序运行时识别并应对可能的安全风险事件，为应用搭建更加有效的内生积极防御体系。

代码疫苗技术拥有广阔的应用空间。除IAST和RASP外，代码疫苗技术还可以用来实现运行时SCA、API模糊测试、应用性能管理（APM）等工具。在未来，这一技术的应用范围会变得更加广泛，软件生命周期各阶段中不同的安全工具都将能更好地集成到一个通用的安全解决方案中，以实现真正的DevSecOps。

致谢

本文的撰写得到北京安普诺信息技术有限公司子芽、宁戈、陈超的帮助，在此表示感谢！

参考文献

[1] GoUpSec. DevSecOps自动化的九大优点 [EB/OL]. [2022-10-15]. https://www.secrss.com/articles/43793.2022-06-21
[2] 敏捷小智. DevSecOps软件研发安全实践——设计篇 [EB/OL]. [2022-10-15]. https://bbs.huaweicloud.com/blogs/349101
[3] GSA. DevSecOps Guide [EB/OL]. [2022-10-15]. https://tech. gsa. gov/guides/dev_sec_ops_guide/
[4] LIETZ S. What is DevSecOps? [EB/OL]. [2022-10-15]. https://www.devsecops.org/blog/2015/2/15/what-is-devsecops
[5] LIETZ S. Principles of DevSecOps [EB/OL]. [2022-10-15]. https://www.devsecops.org/blog/2015/2/21/Principles of DevSecOps
[6] 陈杨. 从被动防御到主动出击，网络安全的智能进化 [EB/OL]. [2022-10-15]. https://baijiahao.baidu.com/s?id=1680323118631685292.2020-10-12
[7] 宁戈. 内生安全免疫，代码疫苗关键技术剖析 [EB/OL]. [2022-10-15]. https://baijiahao.baidu.com/s?id=1733885397350294816.2022-05-26
[8] 廖翰晖. 浅谈在DevSecOps流程中融合RASP安全防护 [EB/OL]. [2022-10-15]. https://view.inews.qq.com/a/20220216A06XDX00.2022-02-16
[9] DOD Enterprise DevSecOps Strategy Guide [EB/OL]. [2022-10-15]. https://dodcio. defense. gov/Portals/0/Documents/Library/DoD% 20Enterprise% 20DevSecOps% 20Strategy% 20Guide_DoD-CIO_20211019. pdf. 2021-10-19.
[10] 阿里云云原生. Apache Log4j2, RASP防御优势及原理 [EB/OL]. [2022-10-15]. https://developer.aliyun.com/article/836012.2021-12-17

作 者 简 介

董毅，北京安普诺信息技术有限公司COO；全面负责公司运营中心的管理工作，拥有超过10年的网络安全产品设计、研发及运营管理全栈经验，对网络安全行业有深刻的洞察和理解；主导撰写《DevSecOps行业洞察报告》《软件供应链安全治理与运营白皮书》《软件供应链安全白皮书》等，拥有多项发明专利。