网安加社区

2024应用程序安全领域现状报告

网安加社区高级分析师:严雪伦

目录

1 什么是应用程序安全?

应用程序安全是指旨在保护应用程序免受可能危及其安全性的威胁的措施、方法和实践。这些威胁可能是恶意软件、注入攻击、软件漏洞或拒绝服务（DoS）等任何威胁。

应用程序安全的目标是识别、纠正和预防安全漏洞。这包括应用程序生命周期的所有阶段，从初始设计、开发、部署、维护到最终退役。

整体应用程序安全不仅涉及保护应用程序本身，还涉及确保应用程序运行的安全环境。它涉及保护应用程序与之交互的网络和数据库，确保传输中和静态数据的安全，以及维护应用程序代码的完整性。

2 应用程序安全的重要性与日俱增

随着越来越多的企业将业务转移到网上，企业对安全应用程序的需求变得至关重要：

保护敏感数据：随着数据泄露事件的日益普遍，确保应用程序的安全比以往任何时候都更加重要。违规行为可能导致重大经济损失、声誉受损和客户信任丧失。

法规遵从性：许多行业都受到严格的数据保护法规的约束，未能保护组织的应用程序可能会导致巨额罚款和法律诉讼。

确保功能和正常运行时间：网络攻击可能会破坏组织的应用程序，导致停机和生产力损失。通过保护应用程序，可以确保它们继续顺利运行。

2.1 2024 年应用程序安全攻击事件

2.1.1 ESXi 勒索软件攻击

- 时间：2024 年 2 月
- 攻击者：“ESXiArgs”组织
- 目标：运行 VMware ESXi 虚拟机监控程序的客户
- 影响：全球超过 3800 台服务器被感染，主要针对美国、加拿大、法国和德国等国家的组织
- 攻击方式：利用了一个两年前的漏洞（CVE-2021-21974），实现代码的远程执行，主要影响了旧版本的 VMware ESXi 中的 OpenSLP 服务

2.1.2 GoAnywhere 攻击

- 时间：2024 年 2 月
- 攻击者：未知
- 目标：Fortra 的 GoAnywhere 文件传输平台
- 影响：导致 300 万会员的部分隐私信息被泄露，窃取了包括宝洁公司、多伦多市政府、皇冠度假村和数据安全公司 Rubrik 等大型组织的数据
- 攻击方式：利用 GoAnywhere 平台的一个零日漏洞，在易受攻击的系统上远程执行代码

2.1.3 3CX 软件供应链攻击

- 时间：2024 年 3 月
- 攻击者：未知
- 目标：3CX 的 VoIP 电话系统应用程序
- 影响：超过 600,000 家组织的客户群受到影响，主要客户包括美国运通、麦当劳、可口可乐、NHS、丰田、宝马和本田等
- 攻击方式：利用 3CX 电话系统应用程序的漏洞进行攻击

2.1.4 Phorpiex 僵尸网络与 LockBit3 勒索软件团伙的攻击

- 时间：2024 年 5 月及之后
- 攻击者：Phorpiex 僵尸网络和 LockBit3 勒索软件团伙
- 影响：Phorpiex 僵尸网络被用于策划恶意垃圾邮件攻击，LockBit3 勒索软件团伙在沉寂后卷土重来，其攻击数量占已发布勒索软件攻击的三分之一
- 攻击方式：利用 Phorpiex 僵尸网络发送数百万封带有 LockBit Black 勒索病毒的网络钓鱼电子邮件

3 影响应用安全领域发展趋势的核心要素

3.1 核心影响要素一：推动安全策略的法规变化

随着网络威胁的持续发酵以及对隐私和数据保护的日益关注，这些管理数据和隐私保护的法规也在不断发展。并且它们在制定应用程序安全策略和实践方面发挥着重要作用。其中一个潜在的变化可能是更加强调用户同意和对个人数据的控制。这一变化将要求应用程序为用户提供更高的透明度，并控制其数据的使用和共享方式。另一个潜在的监管变化可能涉及对数据泄露的更严厉的处罚。这一变化将使企业比以往任何时候都更需要优先考虑应用程序安全。不合规不仅会导致巨额罚款，还会损害公司的声誉。

在此大趋势下，2024 年国内新发布了有关数据管理和隐私保护的法律法规，如下表所示：

表 1.2024 年新发布有关数据管理和隐私保护的法律法规

序号	法律法规名称	发布时间	发布单位 或标准编号	主要内容
1	《关于加强数据资产管理的指导意见》	2024 年 1 月 1 日	财政部	推动数据资产的合规高效流通使用，加强数据资产的全过程管理，并更好地发挥数据资产的价值。
2	《关于优化中央企业资产评估管理有关事项的通知》	2024 年 1 月 30 日	国家监管 与国资改革	首次提出中央企业应当对资产评估项目实施分类管理，要求中央企业确定重大资产评估项目划分标准，并对重大资产评估项目管理提出一系列有针对性的管控要求。
3	《关于加强行政事业单位数据资产管理的通知》	2024 年 2 月 5 日	财政部	要求各部门及其所属单位按照国家有关规定及通知要求，切实加强行政事业单位数据资产管理，充分实现数据要素价值。
4	《数字经济 2024 年工作要点》	2024 年 4 月 30 日	国家发展改革委办公厅、国家数据局	围绕高质量构建数字化发展基础、数字赋能引领经济社会高质量发展等方面部署重点任务。
5	《数据安全技术 数据分类分级规则》	2024 年 4 月 11 日	GB/T 43697-2024	规定于 2024 年 10 月 1 日开始实施，规定了数据分类分级的通用规则，为数据分类分级管理工作的落地执行提供重要指导。
6	《物联网—边缘计算—第 2 部分：数据管理要求》	2024 年 5 月 28 日	GB/T 41780.2-2024	规定于 2024 年 9 月 1 日开始实施，规定了对物联网—边缘计算中的数据管理提出了具体要求，旨在规范数据管理过程，确保数据安全。

3.2 核心影响要素二：软件供应链攻击的持续增长

供应链攻击是应用程序安全领域日益关注的问题。在 Sonatype 的一份报告中显示 [1]，过去三年全球软件供应链攻击的平均年增长率高达 742%。2024 年供应链攻击增加的原因之一是企业对第三方

供应商的依赖增加。随着企业继续将任务外包给外部各方，供应链攻击的风险也在增加，因此企业需要彻底审查其供应商资质并实施强有力的安全措施。

另一个攻击增加的原因是攻击方法的不断变化。在 2024 年，软件供应链攻击方式继续呈现多样化和复杂化的趋势。以下是一些近年主流的软件供应链攻击方式：

（1）入侵上游服务器并注入恶意代码

攻击者通过入侵上游服务器或代码仓库，在其中注入恶意代码。这种攻击方式能够迅速将恶意代码分发到大量用户，从而放大攻击的影响范围。

（2）入侵中间环节发送恶意更新

攻击者入侵软件供应链中间环节的软件升级功能或 CI/CD 工具，通过修改升级流程来实施攻击，而不直接修改源代码库。

（3）依赖性混淆攻击

攻击者利用开源生态系统中的设计弱点，在公共仓库中注册与私有依赖项同名的依赖项，并通过提高版本号来使其被软件构建拉取。

（4）开源组件恶意投毒

攻击者在开源组件中植入恶意代码，并通过组件管理器分发到大量项目中。这种攻击方式利用了开发人员对开源组件的信任。

（5）利用软件构建工具或更新的基础结构破坏

攻击者针对软件构建工具或更新机制进行破坏，以在构建或更新过程中植入恶意代码。这种攻击方式依赖于对软件构建和更新过程的深入理解，因此具有高度的隐蔽性和针对性。

（6）利用信任与依赖关系进行攻击

攻击者通过伪造身份或社交工程等方式，欺骗供应链成员，利用供应链中的信任与依赖关系进行攻击。这种攻击方式利用了人类心理和社会工程学原理，因此难以通过技术手段完全防范。

3.3 核心影响要素三：量子计算对加密的影响

量子计算多年来一直备受瞩目。通过巧妙应用量子力学原理，这些计算机能以远超传统计算机的速度处理信息，其强大的计算能力能在极短时间内破解那些传统计算机需要数千年才能解开的加密代码。然而，尽管这项技术展现出巨大的潜力，但它同样对应用程序安全构成了一定挑战，这种对加密

技术的潜在威胁要求各个组织必须提前做好准备：

（1）破解现有加密算法

量子计算机利用量子力学原理，通过量子比特（qubit）进行存储和计算信息，可以实现高效的并行计算。这种计算能力使得量子计算机能够破解目前广泛使用的加密算法，如 RSA 和椭圆曲线加密。Shor 算法是量子计算中一种重要的算法，它可以利用量子计算机的并行计算能力在多项式时间内破解 RSA 和椭圆曲线加密算法。这将对现代密码学造成重大影响，因为 RSA 和椭圆曲线加密是目前互联网安全体系中的基石。

（2）严重影响非对称加密算法的安全性

非对称加密算法使用一对密钥进行加密和解密，其中一个密钥是公开的，另一个密钥是私密的。RSA 和椭圆曲线加密是目前最常用的非对称加密算法，但这两种算法都面临着量子计算机的威胁。由于 Shor 算法可以在多项式时间内破解非对称加密算法，因此整个非对称加密体系的安全性都将受到威胁。这将对密钥交换、数字签名等安全应用造成严重影响。然而，量子计算机对对称加密算法的影响相对较小，因为对称密钥的长度通常较短，量子计算机需要耗费大量时间和资源来破解它们。因此，对称加密算法在短期内仍然可以保持一定的安全性。

尽管面临这样的威胁，但量子计算也为应用程序安全领域带来了新的机遇。目前，一种名为量子加密或量子密钥分发的新型加密方法正在研发中。这些方法基于量子力学的原理，旨在创建理论上无法破解的密钥。这一进步或许能为量子计算带来的加密挑战提供有效的解决方案。

4 2024 年应用安全领域现状分析

4.1 DevSecOps 成熟度提升

加速开发、持续交付、管道弹性、可扩展性以及端到端的透明度，是 DevOps 实践中的核心支柱。要实现这些标准，开发、安全和运维团队必须紧密协作，共同努力。

DevSecOps，作为 DevOps 方法论的自然演进，核心在于深化和普及多团队间的安全文化，确保在 DevOps 环境下，安全挑战能以统一且前瞻性的方式被及时应对。通过将安全实践自然融入软件开发生命周期（SDLC）和持续集成（CI）流程，DevSecOps 致力于将安全性从孤立的阶段提升为开发

生命周期中不可或缺、贯穿始终的要素。这样，安全性不再仅仅是一个附加的考量，而是深植于开发、测试和部署的每一个环节，成为其固有的、不可或缺的特性。

4.1.1 为什么要自动化?

DevOps 的核心原则在于在整个软件开发生命周期（SDLC）中自动化手动流程，以提高效率和质量。对于任何组织而言，自动化都是通过持续集成和持续部署来加速软件开发和交付的关键所在。成功的 DevSecOps（即融入安全性的 DevOps）依赖于自动化与集成的协同作用，以及明确的标准和策略作为指导。这样的实践不仅能让安全团队确信安全利益得到了有效保障，还能让 DevOps 团队保持高效运作，有信心避免流水线中断。

相较于手动测试，自动安全测试能迅速且一致地执行，使得开发人员能够在开发早期阶段即发现并解决问题，从而不会对交付进度或工作效率产生负面影响。这种预防性措施极大地提高了软件质量和安全性，同时降低了潜在风险。

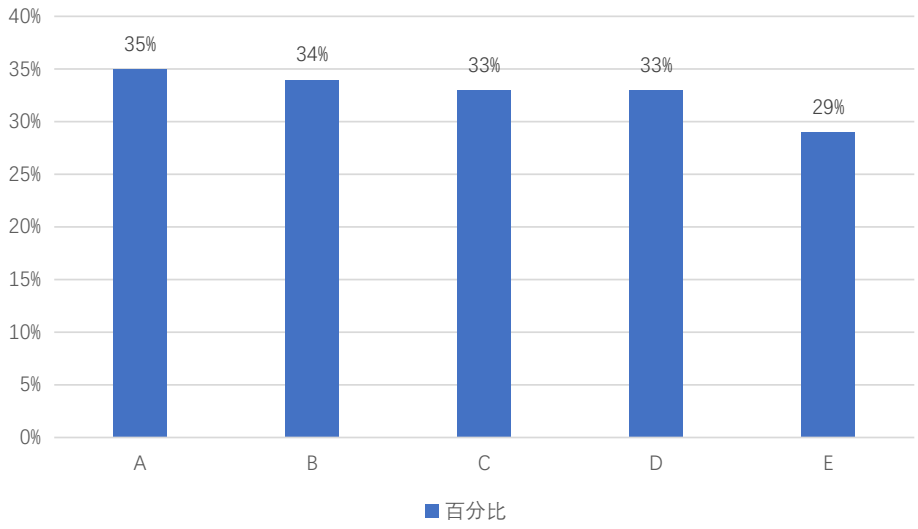
- **一致性：**自动测试可确保对每一次的构建和部署一致地进行安全检查。手动测试可能会导致测试过程和覆盖范围不一致。
- **可扩展性：**随着软件复杂性的增长，手工测试将变得不切实际。自动测试容易扩展，以便跨越不同组件进行大量测试。
- **持续集成和持续部署（CI/CD）：**自动测试在 CI/CD 管道中至关重要，因为这些管道中会发生快速而频繁的代码变更。自动测试可以快速验证变更，防止错误代码进入生产环境。
- **持续改进：**自动测试提供数据和洞察，可以帮助开发和安全团队随时间的推移改进安全实践，允许他们系统地分析和处理漏洞模式。
- **记录：**自动测试可以记录整个测试过程，从而更容易跟踪和审计安全措施与合规要求。
- **减少人为错误：**由于疲劳或疏忽，手动测试容易出错。自动测试遵循预定义的脚本，能够降低人为错误的风险。
- **节省时间和成本：**在开发过程的后期或生产过程中识别和修复安全问题既耗时又昂贵。自动测试可将这些费用降至最低。
- **改进开发者体验：**自动的应用安全测试允许开发者采取主动的、全面的、有助于学习和提高安全知识和技能的方式来解决安全问题，从而增强开发者体验，最终提高软件安全性并提升整个开发过程的效率。

4.1.2 全球 DevSecOps 企业应用现状

新思科技的 CyRC（网络安全研究中心）于 2023 年初联合国际市场咨询公司 Censuswide，对负责安全事务的 1,000 名 IT 专业人士开展了一项调查。受访者包括开发人员、应用安全专业人员、DevOps 工程师、CISO 以及在技术、网络安全和应用、软件开发领域担任各种职务的专家。受访者来自美国、英国、法国、芬兰、德国、中国、新加坡和日本。

调研结果 [2] 中显示，大多数受访者对自己使用的 AST 工具普遍感到不满：

表 2. 受访者对 AST 工具意见收录表



- A：工具不能根据漏洞的暴露程度、可利用性和严重程度来确定修复顺序
- B：因速度太慢而无法适应快速的发布周期以及持续部署
- C：性价比低
- D：不准确以及不可靠
- E：无法合并以及关联数据来解决问题

在这 1,000 名被访问人员中有 35% 的人员认为这些安全工具不能根据漏洞的暴露程度、可利用性和严重程度来确定修复顺序，也无法合并以及关联数据来帮助解决问题（29%）。有 34% 的人员认为这些安全工具因速度太慢而无法适应快速的发布周期以及持续部署。有 33% 的人觉得这些安全工具性价比低并且还有 33% 的人认为这些安全工具的结果并不准确。

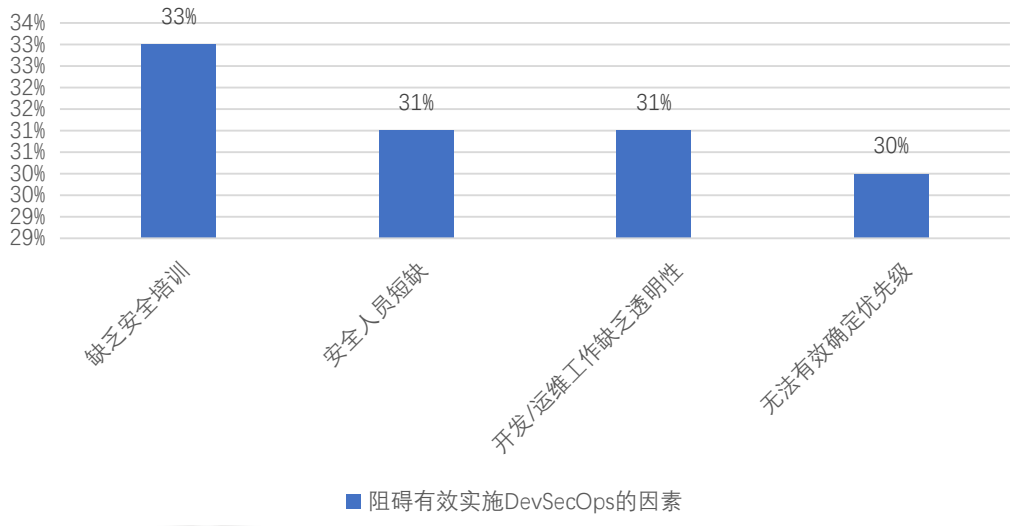
4.1.3 大多数 DevOps 团队都在某种程度上采用了 DevSecOps

共有 91% 的受访者表示，他们已将开展 DevSecOps 活动相关的安全措施纳入到了软件开发管道中。可以肯定地说，采用 DevSecOps 方法论现已成为软件开发的一部分。29% 的受访者表示他们拥

有跨职能部门的 DevSecOps 团队—由开发、安全和运维部门成员构成的协作团队，是安全计划取得成功的重要因素。

然而想要有效实施 DevSecOps 存在许多障碍。超过 33% 的受访者指出，缺乏安全培训是主要障碍。紧随其后的是安全人员短缺（31%）。开发以及运维工作缺乏透明性（31%）以及无法有效确定优先级（30%）也是主要原因。

表 3. 阻碍有效实施 DevSecOps 的因素



超过三分之一的受访者表示，将自动安全测试集成到构建以及部署工作流中是安全计划取得成功的关键。其他的主要成功因素还包括通过基础架构即代码来执行安全以及合规策略，在开发和运维团队中培养安全支持者 (security champions)，以及加强开发、运维和安全团队之间的沟通等。

4.1.4 在 SDLC 后期再处理重大漏洞和技术债务的成本将会非常高

最合理以及最节省成本的方式是在开发生命周期的最早期就介入安全，尽可能不要留下技术债务问题。超过 80% 的受访者表示，2022-2023 年间，已部署软件中的重大漏洞以及安全问题以某种形式影响了他们的工作进度。28% 的受访者表示，他们的组织需要长达三周的时间来修补已部署应用程序中的重大安全风险以及漏洞；另有 20% 的受访者表示，这可能需要一个月的时间。考虑到现在的漏洞利用速度比以往任何时候都要快，这些数字令人担忧。最新研究表明，**超过一半的漏洞在披露后的一周内即被利用。**

超过 70% 的受访者表示，通过自动扫描代码来查找漏洞或编码缺陷是一种有用的安全措施，34% 的受访者认为自动 AST“非常有用”。对代码进行安全漏洞和其他缺陷的自动化扫描，在“工具以及

流程的有用性”类别中排名第一，紧随其后的是“在 SDLC 的需求挖掘阶段明确安全需求”以及“通过 BSIMM 和 SAMM 等模型对软件安全计划进行正式评估”。

4.1.5 AST 工具现阶段使用场景的弊端

几乎所有受访者都认为 AST 工具与其业务需求不符。在 1,000 名受访者中，大多数人都认为 AST 工具存在各种各样的问题是他们面临的主要挑战，包括这些工具无法根据业务需求对修复措施进行优先级排序 (35%)，也无法合并以及关联数据来帮助解决问题 (29%)。

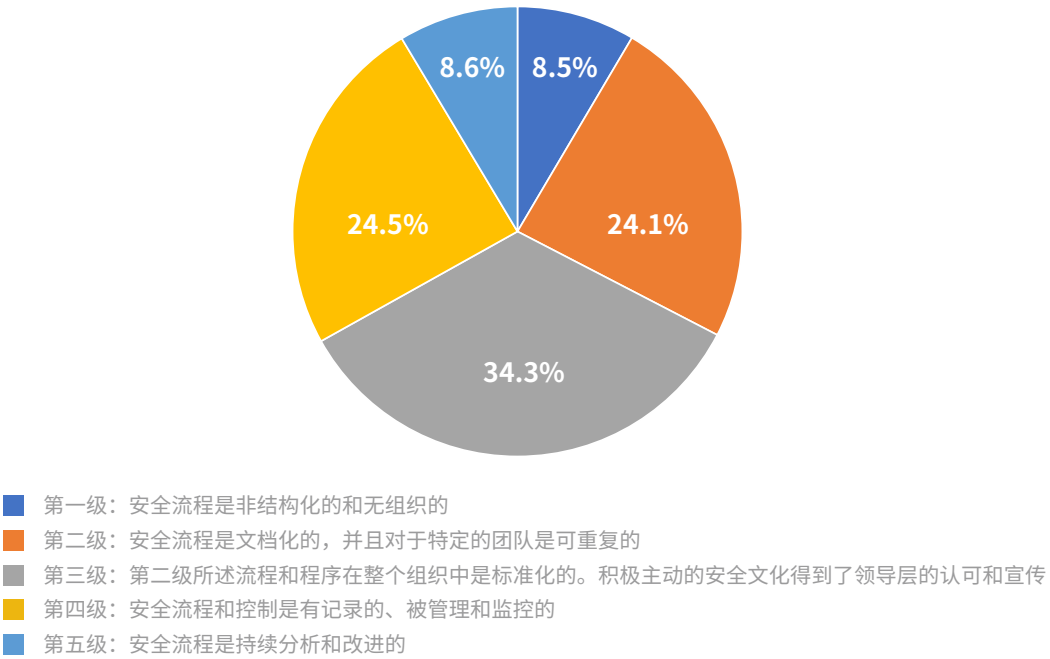
4.1.6 安全与 AI 技术结合大势所趋的同时也存在多维度风险

52% 的安全专业人员已经开始在 DevSecOps 活动中积极使用 AI，但超过四分之三的人担心 AI 的使用问题。调查结果表明，安全团队正在积极使用 AI、机器学习、自然语言处理和神经网络。然而，随着生成式 AI 工具（如 AI 驱动的编码建议）的使用日益增多，引发了围绕 AI 所生成代码的一系列知识产权、版权和许可问题，某些情况下甚至引起了诉讼。

4.1.7 组织 DevSecOps 部署以及成熟度现状

在 1,000 名受访者中，超过三分之一的受访者认为其安全计划已经达到了成熟度的第三级，即整个组织的安全流程都是文档化的、可重复的和标准化的。另有 25% 的受访者认为其安全计划已经达到了第四级，即安全流程也被记录，同时还被监控和评估。

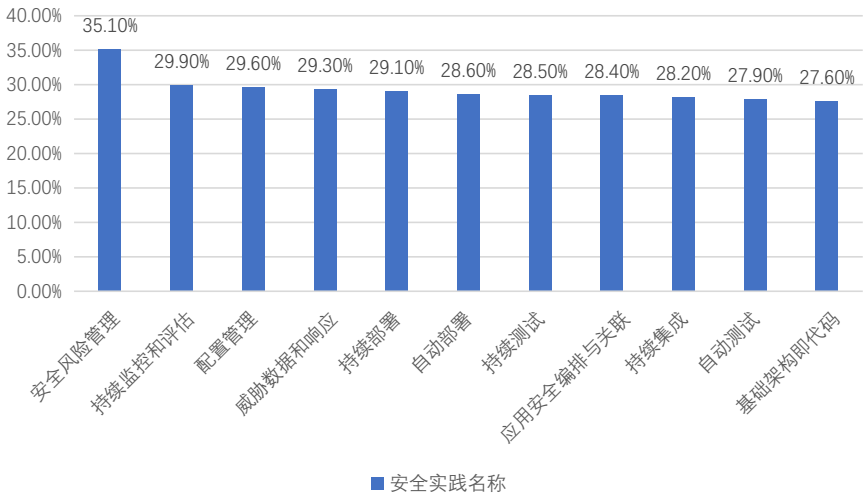
表 4. 组织当前的软件安全项目以及计划的成熟度所处等级占比表



4.1.8 组织广泛采用安全实践

在被调研的组织中，它们分别真实采用了以下这些安全实践：

表 5. 组织安全实践采用

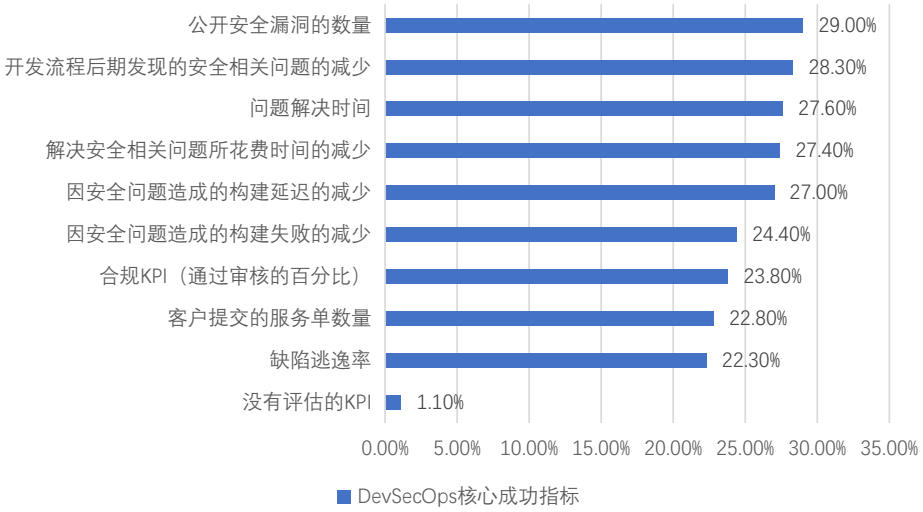


其中被 358 名受访者（约 35.1%）所提及的最佳实践“安全风险管理”，其涉及到在开发过程中的每个阶段整合安全考虑因素，以识别、评估和减轻与软件应用相关的潜在安全风险。在 SDLC 的框架下，整体安全风险管理内容见附录 A。

4.1.9 评估 DevSecOps 实践成功的主要 KPI

如表 6 所示，受访组织认为公开安全漏洞的数量、开发流程后期发现的安全相关问题的减少、问题解决时间为核心主要衡量 DevSecOps 实践是否成功的核心 KPI。

表 6. 评估 DevSecOps 实践成功的主要 KPI



4.1.10 AST 工具对于开发安全以及 DevSecOps 的成功实践是否真实有效?

AST 工具是用来检查开发安全的核心工具类别。那么 AST 工具对于 DevSecOps 的成功是否真实有效？使用方的真实使用感受如何？本项调研的结果给出了真实用户的使用感受以及实际情况总结，详情见下述图表合集：

表 7. 组织 SAST 使用反馈

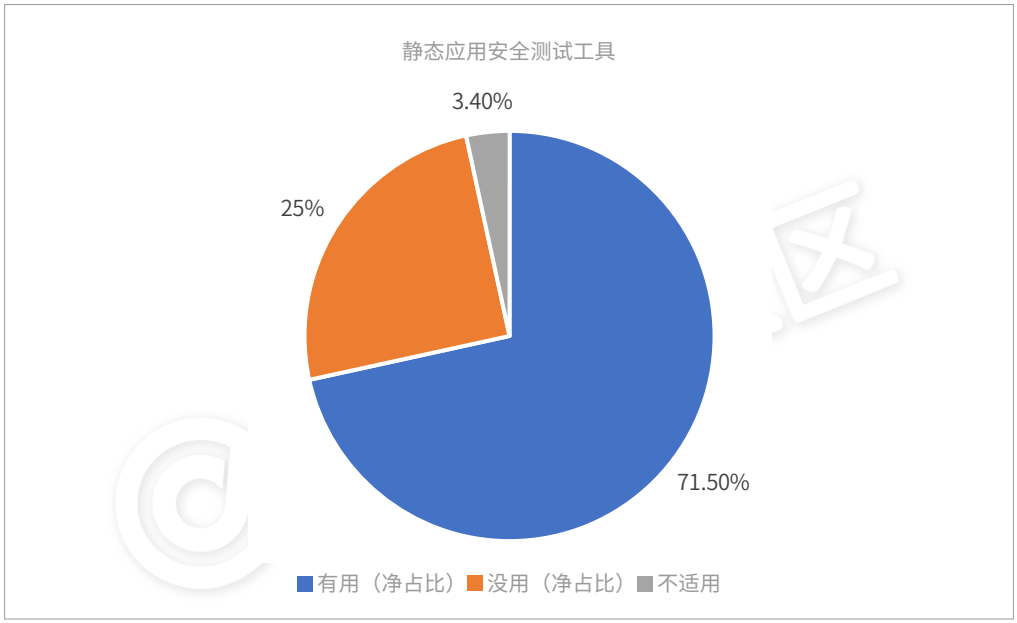


表 8. 组织 DAST 使用反馈

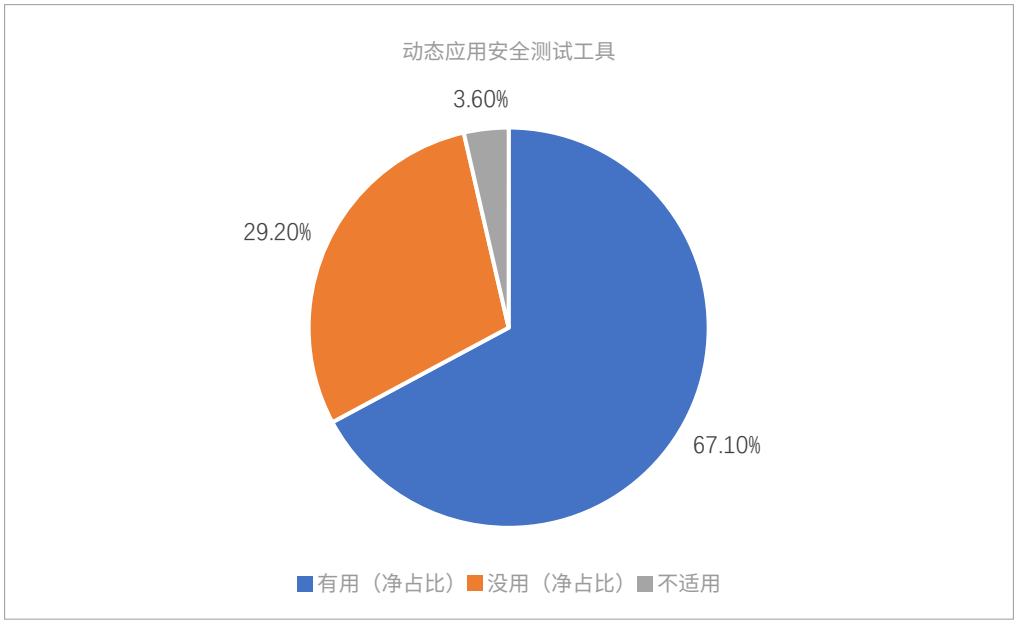


表 9. 组织 SCA 使用反馈

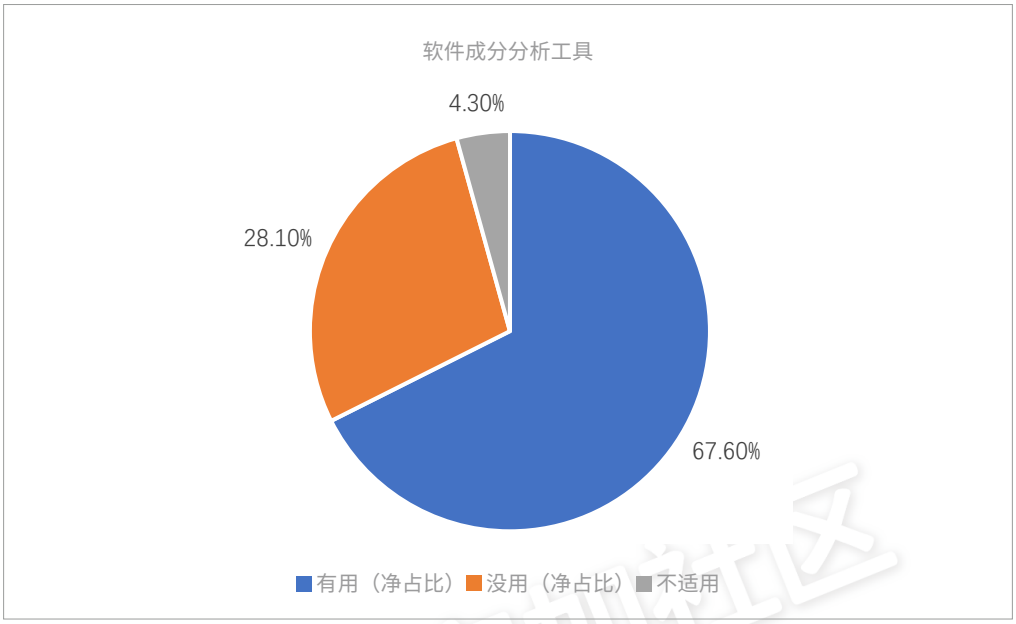
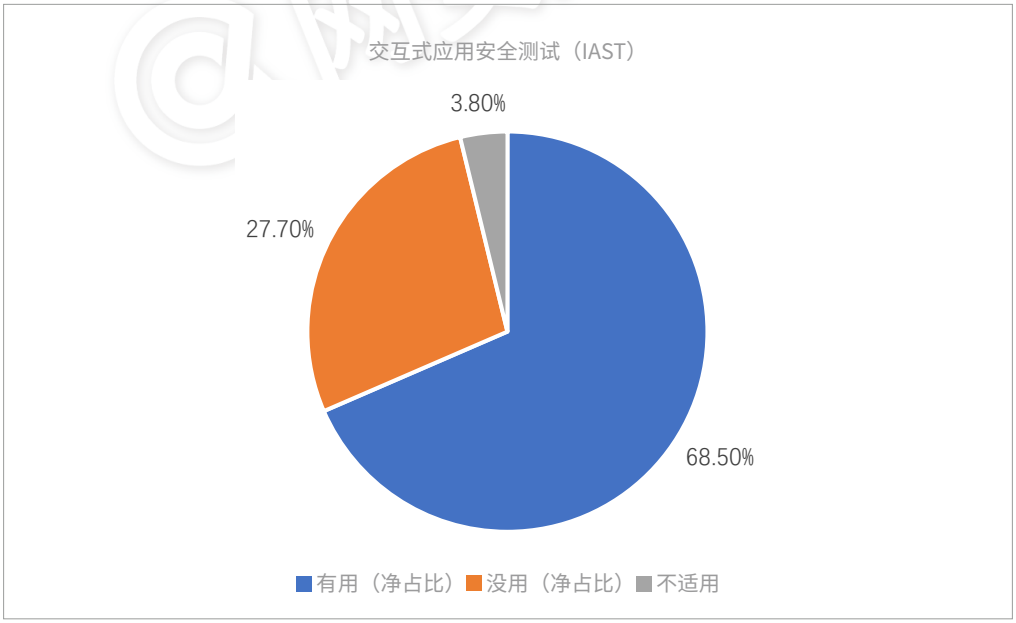


表 10. 组织 IAST 使用反馈



从上述组表的结果反馈显示，SAST 是目前最受欢迎的 AST 工具，其次是 IAST 和 SCA，最后是 DAST。SAST 和 DAST 采用不同的测试策略，并分别适用于软件开发生命周期（SDLC）的不同阶段。SAST 在 SDLC 的早期阶段，即应用部署之前，扮演着至关重要的角色，它能够有效地识别和消除专有代码中的潜在漏洞。相对而言，DAST 则更侧重于在部署后，通过模拟实际运行环境来发现应用程序在运行时可能出现的问题，如身份验证漏洞和网络配置缺陷。

IAST（交互式应用程序安全测试）结合了 SAST 和 DAST 的某些关键功能，其目标在于检测那些可能无法被其他类型测试轻易识别出的重大安全缺陷。IAST 的出现，为开发者提供了一种更为全面、高效的安全测试手段，以确保应用程序的安全性。而 SCA 在识别和管理开源软件中的安全和许可风险方面起着至关重要的作用，这是现代软件开发不可或缺的一环。特别是在当前环境下，任何给定应用中超过四分之三的代码可能是开源的，因此确保这些代码的安全性和合规性变得尤为重要。

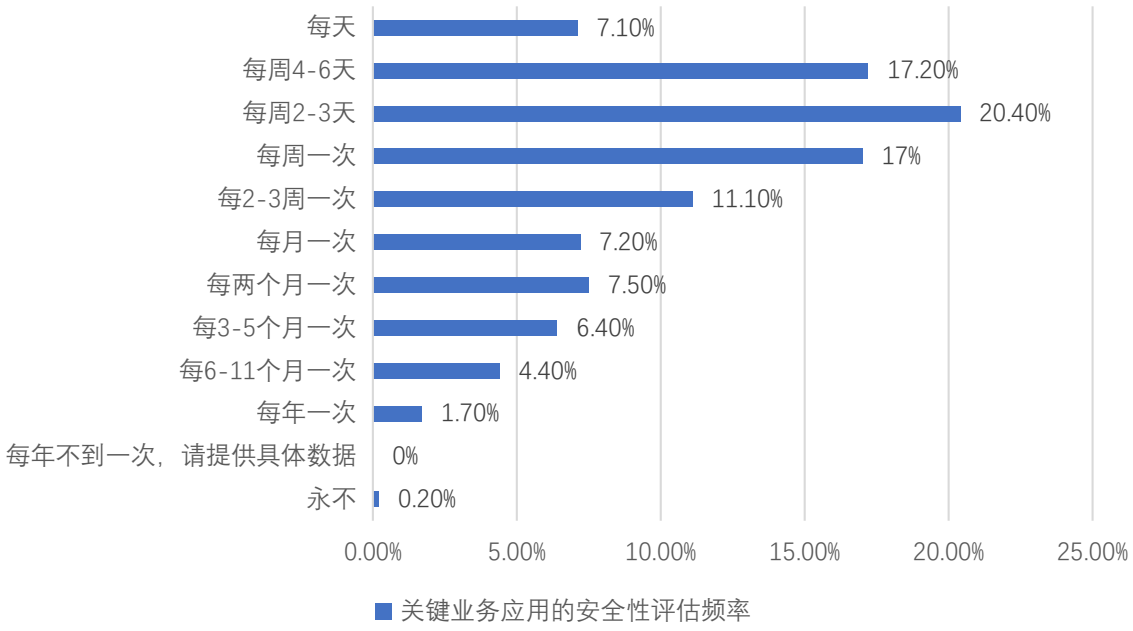
此外，由于许多组织依赖从独立软件供应商处购买的打包软件，以及物联网（IoT）设备和嵌入式固件，他们可能需要在其应用程序安全测试（AST）工具箱中纳入某种形式的 SCA 二进制分析。这种分析能够帮助组织识别并管理潜在的开源风险，无论是在源代码层面还是二进制层面，确保软件栈的完整性和安全性。

总的来说，所有 AST 工具都有近 7 成的组织认为该工具是真实有用并且可以为公司有效解决漏洞风险问题的选择，是 DevSecOps 实践成功落地不可或缺的环节。

4.1.11 应用安全测试的频率和漏洞风险处理周期

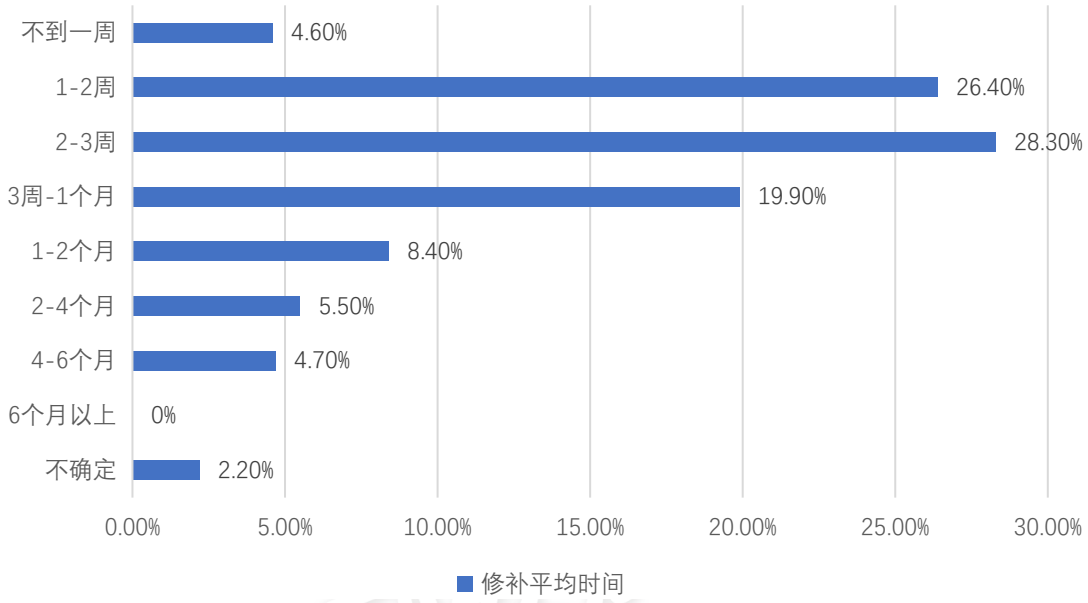
应用安全测试的频率取决于多个因素，包括应用程序的业务关键性、行业和威胁情况等。对于非常重要的应用程序，应定期进行评估（见表 11）。

表 11. 关键业务应用的安全性评估频率



参与本次调查的大多数受访机构都表示，他们平均每周对业务关键型应用进行两到三天的漏洞扫描。

表 12. 修补或处理已部署的或正在使用的应用程序中的重大安全风险以及漏洞的平均时间



调查结果显示有 28% 的组织需要花费长达三周的时间来修补重大漏洞（表 12），这个数据可能会令人担忧，但这要结合其他因素来考虑。不是所有漏洞都需要进行修复，组织应当优先修复那些超危高危风险的漏洞，以及会对组织自身业务造成直接影响或者带来风险的漏洞。

5 安全即代码的发展

当前，随着企业对于提升敏捷性和支持混合工作模式的迫切需求不断增长，云基础架构的采用率也随之飙升。这一趋势推动了开发团队更加倾向于采用“安全性即代码”的理念，将安全性无缝集成到软件和产品之中，确保整个开发流程的安全性。安全即代码是一种将安全策略编纂和自动化的方法，与基础结构即代码非常相似。这种方法允许一致、可重复和可扩展的安全实践。其核心是将安全配置编码作为开发和部署流程输入的代码，可以让组织分析其安全配置，轻松更改和重新部署，并持续监控其安全配置的状态，以评估其是否匹配策略。如此更有利于分析安全配置，并且持续可验证。

5.1 实施安全即代码 SaC 的基础要素

实施 SaC 的基础要素包括以下几个方面：

（1）安全模型的应用

在软件开发阶段实施主动控制的安全模型，如 OWASP，为开发者提供了一套明确的安全实践指南。这确保了开发过程始终遵循行业最佳实践，从而提高了软件的安全性。

（2）依赖关系检查

在应用程序的开发和构建过程中，复杂的代码依赖关系需要被清晰管理。依赖关系图作为一种工具，可以帮助开发者深入了解代码库的各个部分及其组件间的协同工作方式，从而更有效地识别和修复潜在的安全隐患。

（3）安全工具集成

将安全工具融入软件开发流程是实现安全编码（SaC）的核心。在整合过程中需要关注工具的易用性、团队协作的流畅性、自动化的实现以及速度和准确性。这些确保了开发人员能够高效、准确地利用安全工具进行扫描和验证。

（4）自动化与持续集成

通过自动化和持续集成，开发团队能够快速、频繁地构建、测试和部署应用程序。这不仅提高了开发效率，而且有助于在开发早期发现并修复安全问题，降低漏洞风险。同时，自动化确保了安全措施与开发流程的紧密结合。

（5）培训与安全意识

为了增强开发团队对 SaC 的理解和应用能力，组织应提供专门的培训以提升他们的安全意识。这些培训涵盖了常见的安全漏洞、攻击方式以及如何在编写代码时考虑安全性。

（6）监控与日志分析

组织通过实施全面的监控和日志分析策略，可以实时发现潜在的安全问题。通过分析应用程序的日志数据，组织能够检测到异常行为或潜在的安全威胁，并快速响应以减轻潜在损失。

（7）建立反馈机制

为了确保 SaC 的持续改进效果，组织需建立一个有效的反馈循环机制。通过定期评估和审查现有的安全措施，组织能够发现并改进不足之处。同时，组织须与业务利益相关者保持密切沟通，了解他们的需求和期望，以便不断优化组织的安全策略。

总的来说，为了有效落地 SaC 组织需要从上述 7 个维度着手，以帮助组织在整个开发过程中融入

安全性，降低软件面临的风险。组织对安全即代码的需求将在近几年出现显著增长。随着越来越多的组织采用 DevOps 和云原生技术，对自动化和可扩展的安全解决方案的需求只会增加。安全即代码使组织能够自动执行其安全策略，从而降低人为错误的风险并提高安全实践的一致性。

6 对零信任架构的广泛采用

零信任是一种安全模型，默认情况下假定不信任，即使对于内部网络也是如此。相反，信任必须赢得并不断得到验证。

6.1 对传统安全防护机制缺陷进行弥补

现如今组织的 IT 架构面临越来越多的安全挑战：

- **数据中心内部东西向流量安全防护薄弱：**在应用云化、构建方式微服务化的大背景下，服务间需要更加频繁地进行通信和交互，在访问流量增大的情况下，必不可免地会遇到东西向流动的恶意流量。然而传统安全防护是以南北向为主，因此薄弱环节可能会被攻击者所利用。
- **安全策略仍待细化：**企业所管理的资源增多，粒度细化，安全防护策略也应随着资源的细粒度的变化而变化。
- **跨云的连接、数据传输使得资源暴露面增大：**用户现在可以在任意时间在任意位置使用任意设备快速获取资源。但是云间的连接点非常薄弱，攻击者可以通过攻击进入云中，并在云间实现威胁渗透。
- **防火墙与 VPN 无法保证用户操作合法性：**办公形式多样化，用户接入方式复杂多样，传统的网关安全防护设备无法判断使用账号、密码的用户身份是否合法、操作行为是否符合其身份。

此外随着数字经济走深向实，企业不断开展产品服务创新实现业务转型的大场景之下，组织可能会面临新的特性威胁：

① 业务面临更多欺诈威胁；

② 物联网终端安全防护能力差异大，终端设备易被入侵。

在信通院的一份关于零信任现状的报告中提及到 [4]，零信任之所可以脱颖而出弥补传统安全机制

的短板的主要原因包括以下几项：

① 零信任面向资源管理而非网络；

② 零信任屏蔽安全策略预分配带来的威胁；

③ 零信任资源对外隐身；

④ 以身份为核心执行动态安全防护。

零信任可以基于最小化权限原则对供应商的访问过程进行动态授权，限制上游供应商权限，防止攻击者通过对供应商的成功攻击进而对企业网络环境进行入侵。零信任与“安全左移”理念相符合，助力研发运营关键环节的风险监测与安全准入。其默认第三方组件、容器镜像、代码仓库等实体不可信任。

2024 年零信任架构已经逐渐得到广泛采用。随着网络威胁变得越来越复杂，传统的基于边界的安全模型被证明是不够的。零信任架构提供了更强大的安全解决方案，因为它们需要持续验证信任，而不管用户的位置或网络如何。

6.2 加强对 API 安全性的关注

API（应用程序编程接口）是现代应用程序的关键组件。但是，它们也存在重大的安全风险，因为它们可以为网络犯罪分子提供访问敏感数据的网关。目前常见的四种主要类型是 RESTFUL、SOAP、GraphQL 和 gPRC。

6.2.1 API 安全问题有多严重

企业安全团队面临的 API 安全风险日益加剧，而随着使用 API 的客户互动和业务流程越来越多，这种挑战只会越来越大。简言之，API 使用量迅猛增长，许多安全团队正在努力完善他们的 API 安全策略。因此，API 安全迅速成为 IT 和安全主管的首要任务和一大关注领域。

6.2.2 API 安全与应用程序安全有何不同

虽然 API 安全和传统应用程序安全是两个相互关联的领域，但 API 安全是一个独特的挑战，主要有两个原因，即规模和复杂性。

（1）更大的防护规模

API 使用量快速增长有三个因素：

① 微服务的使用日益广泛。

② 在直接用户渠道中，有 React、Angular 和 Vue 等现代前端应用程序框架使用 API 并且正在取代有时不使用 API 的传统 Web 应用程序。

③ 为了适应全新的渠道（例如合作伙伴、物联网和业务自动化），也会添加 API。

（2）灵活性导致复杂性

与 Web 应用程序不同，API 可以在许多不同的场景中以编程方式使用，这使得区分合法使用与攻击和滥用十分困难。

6.2.3 保护 API 的最佳实践

想增强 API 安全性的企业从以下最佳实践开始：

- 将 API 安全标准和实践与企业的软件开发生命周期进行整合。
- 将 API 文档和自动化安全测试纳入持续集成以及持续交付（CI/CD）管道。
- 确保对 API 应用适当且有效的身份验证和授权控制。
- 实施速率限制措施，帮助防止 API 滥用或崩溃。
- 使用专用网关和以及或内容交付网络（CDN）增强速率限制和其他应用程序级措施，以降低分布式拒绝服务（DDoS）攻击的风险。
- 让 API 安全测试成为更大范围的应用程序测试流程中不可或缺的一部分。
- 执行持续的 API 发现。
- 实施系统化的方法来识别和修复常见 API 漏洞，包括 OWASP 十大 API 安全风险。
- 使用基于签名的威胁检测和预防，作为针对已知 API 攻击的基准级防护。
- 利用人工智能 AI 和行为分析来增强基于签名的检测，使 API 威胁检测的扩展性、准确性和业务相关性更强，并且能够抵御新型威胁。
- 确保 API 安全监控和分析持续数周并覆盖多个 API 会话。
- 作为 API 安全监控和告警的补充，为威胁搜寻人员、开发人员、DevOps 和支持人员提供对 API 清单和活动数据的按需访问权限。

6.2.4 常见的 API 滥用方法

API 会受到多种方式的攻击和滥用，以下是一些常见的例子：

（1） 漏洞利用：

底层基础架构中的技术漏洞可能会导致服务器受损。此类漏洞的例子很多，从 Apache Struts 漏洞（CVE-2017-9791、CVE-2018-11776 等）到 Log4j 漏洞（CVE-2021-44228 等）都包括在内。

（2）业务逻辑滥用：

这些可怕的场景时常让首席信息安全官 (CISO) 彻夜难眠，因为传统的安全控制措施对此毫无用处。逻辑滥用是指攻击者利用应用程序设计或实施的缺陷来引发意外行为和未经批准的行为。

（3）未经授权的数据访问：

API 滥用的另一种常见形式是攻击者利用失效的授权机制来访问其无权访问的数据。这些漏洞有很多名称，例如失效的对象级授权 (BOLA)、不安全的直接对象引用 (IDOR)，以及失效的功能级授权 (BFLA)。

（4）帐户接管：

在凭据被盗乃至跨站点脚本攻击之后，帐户可能会被接管。一旦发生这种情况，即使是编写得最好、安全性最高的 API 也可能被滥用。毕竟，如果不执行行为分析，任何经过身份验证的活动都被视为合法使用。

（5）数据抓取：

如果企业通过公共 API 提供数据集，攻击者就可能积极查询这些资源，以便批量捕获大体量、有价值的数据集。

（6） 业务拒绝服务 (DoS)：

API 攻击者如果请求后端执行繁重任务，可能引发应用程序层的“服务侵蚀”或完全拒绝服务（GraphQL 中十分常见的一个漏洞，但任何资源密集型 API 端点实施都可能发生这种情况）。

随着越来越多的组织依赖 API 用于其应用程序，确保其安全性将成为重中之重。这将涉及实施强大的身份验证和授权机制，加密传输中的数据，以及定期测试和监控 API 的漏洞。

7 结语

随着信息技术的飞速发展，应用安全风险正呈现出前所未有的剧增态势，这使得应用安全的重要性愈发凸显。在当今日益复杂的网络环境中，保护应用程序免受各种威胁和攻击已成为企业和组织必

须面对的重大挑战。

为了有效应对这些挑战，DevSecOps（开发安全运维一体化）的成熟度正在不断提升。DevSecOps 通过将安全性融入软件开发和运维的每一个环节，实现了安全、开发和运维团队之间的紧密协作，从而显著提高了应用程序的安全性。随着 DevSecOps 的普及和实践，我们有望看到更多安全、高效且可靠的软件产品问世。

与此同时，“安全即代码”的理念也得到了快速发展。这种理念将安全性视为与功能性和性能同等重要的代码属性，要求开发者在编写代码时充分考虑安全性因素。通过引入自动化安全测试、静态代码分析等工具和方法，我们能够更早地发现并修复潜在的安全漏洞，从而降低应用程序的安全风险。

零信任架构的广泛应用为应用安全提供了全新的解决思路。零信任架构强调“不信任，验证一切”的原则，要求对所有访问请求进行严格的身份验证和授权管理。这种架构模式能够有效防止内部和外部威胁的入侵，保护应用程序和数据的安全。此外，随着 API（应用程序接口）在业务系统中的广泛应用，API 安全性也受到了越来越多的关注。API 作为连接不同服务和应用程序的桥梁，其安全性直接关系到整个业务系统的稳定性和可靠性。因此，加强 API 安全性管理和监控已成为当前应用安全领域的重要任务之一。

总的来说，面对日益严峻的应用安全风险挑战，企业需要持续关注安全现状以及变化趋势，才能构建更加安全、可靠的软件应用生态系统，为企业和组织的数字化转型提供坚实的支撑。

8 参考文献

[1]8th State of the software supply chain, Sonatype

[2]Global State of DevSecOps 2023, Synopsys（2023-10）

[3]DevSecOps 安全即代码基础指南（2024-01-18）

[4] 零信任发展研究报告，中国信通院（2023-08）

9 附录 A

整体安全风险管理办法

- **需求分析：**在 SDLC 中尽早识别安全需求和限制，并定义安全目标。
- **设计：**将安全原则纳入到系统架构和设计中，以确保应用程序的设计包含针对常见漏洞的适当防护措施。
- **开发：**实施安全编码实践，并遵守解决安全问题的编码标准。使用集成的安全测试工具，如静态应用安全测试 (SAST) 和软件组成分析 (SCA)，在编写代码和引入开源或第三方代码时捕获漏洞。
- **测试：**执行各种类型的安全测试来识别应用程序中的漏洞，如 SAST、动态应用安全测试 (DAST)、SCA 和渗透测试。
- **部署：**安全地配置应用程序的运行环境。实施访问控制、网络安全以及适当的身份验证和授权机制。
- **监控与评估：**持续监控生产环境中的应用程序，以发现安全事件和异常情况。实施日志记录和监控解决方案，以检测和响应潜在的违规行为。30% 的受访者表示，这是其组织采用的主要安全实践。
- **响应和补救：**制定事件响应计划，以快速有效地处理安全事件。修复在测试阶段检测到的问题。
- **透明度和安全性：**建立明确的规范、标准和策略，并报告安全风险和风险容忍度。
- **培训：**为开发团队提供安全编码实践、常见漏洞和最佳安全实践方面的培训，以使开发人员能够主动解决安全问题。遗憾的是，34% 的受访者认为，“开发人员、工程师的安全培训不足或无效”是导致其组织无法有效实施 DevSecOps 的主要障碍之一。
- **持续改进：**定期审查和改进 SDLC 中的安全流程和实践。

网安加社区

官网：www.cwasp.com