OWASP

TOP 10

2021

中文版

目录

OWASP TOP10

01 项目介绍

2021版变化说明

欢迎参阅最新一期OWASP Top 10！2021年版OWASP Top 10是全新的，使用了新图形设计，且可以单页的形式打印出来，或在OWASP的主页浏览项目内容。

非常感谢所有为2021年版发布贡献时间和数据的人们。如果没有您们，就没有2021年版OWASP Top 10的产出。**谢谢！**

2021年版Top 10有哪些变化?

2021年版Top 10产生了三个新类别，原有四个类别的命名和范围也发生了变化，且进行了一些整合。考虑到应关注根本原因而非症状，我们更改了一些类别的名称（*来源于社区调查结果）。

Top 10 2017	Top 10 2021
A01：2017-注入 Injection	A01：2021-失效的访问控制 Broken Access Control
A02：2017-失效的身份认证 Broken Authentication	A02：2021-加密机制失效 Cryptographic Failures
A03：2017-敏感信息泄漏 Sensitive Data Exposure	A03：2021-注入 Injection
A04：2017-XML外部实体 XML External Entities(XXE)	A04：2021-不安全设计 Insecure Design 新
A05：2017-失效的访问控制 Broken Access Control	A05：2021-安全配置错误 Security Misconfiguration
A06：2017-安全配置错误 Security Misconfiguration	A06：2021-自带缺陷和过时的组件 Vulnerable and Outdated Components
A07：2017-跨站脚本 Cross-Site Scripting(XSS)	A07：2021-身份识别和身份验证错误 Identification and Authentication Failures
A08：2017-不安全的反序列化 Insecure Deserialization	A08：2021-软件和数据完整性故障 Software and Data Integrity Failures 新
A09：2017-使用含有已知漏洞的组件 Using Components with Known Vulnerabilities	A09：2021-安全日志和监控故障 * Security Logging and Monitoring Failures
A10：2017-不足的日志记录和监控 Insufficient Logging & Monitoring	A10：2021-服务端请求伪造 * 新 Server-Side Request Forgery

2021版变化说明

OWASP Open Web Application Security Project

A01：2021-失效的访问控制 Broken Access Control	从第5位上升成为Web应用程序安全风险最严重的类别；提供的数据表明，平均3.81%的测试应用程序具有一个或多个CWE，且此类风险中CWE总发生漏洞应用数超过31.8万次。在应用程序中出现的34个匹配为“失效的访问控制”的CWE次数比任何其他类别都多。
A02：2021-加密机制失效 Cryptographic Failures	排名上升一位。其以前被称为“A3:2017-敏感信息泄漏（Sensitive Data Exposure）”。敏感信息泄漏是常见的症状，而非根本原因。更新后的名称侧重于与密码学相关的风险，即之前已经隐含的根本原因。此类风险通常会导致敏感数据泄露或系统被攻破。
A03：2021-注入 Injection	排名下滑两位。94%的应用程序进行了某种形式的注入风险测试，发生安全事件的最大率为19%，平均率为3.37%，匹配到此类别的33个CWE共发生27.4万次，是出现第二多的风险类别。原“A07:2017-跨站脚本（XSS）”在2021年版中被纳入此风险类别。
A04：2021-不安全设计 Insecure Design	2021年版的一个新类别，其重点关注与设计缺陷相关的风险。如果我们真的想让整个行业“安全左移”，我们需要更多的威胁建模、安全设计模式和原则，以及参考架构。不安全设计是无法通过完美的编码来修复的；因为根据定义，所需的安全控制从来没有被创建出来以抵御特定的安全攻击。
A05：2021-安全配置错误 Security Misconfiguration	排名上升一位。90%的应用程序都进行了某种形式的配置错误测试，平均发生率为4.5%，超过20.8万次的CWE匹配到此风险类别。随着可高度配置的软件越来越多，这一类别的风险也开始上升。原“A04:2017-XML External Entities(XXE) XML外部实体”在2021年版中被纳入此风险类别。
A06：2021-自带缺陷和过时的组件 Vulnerable and Outdated Components	排名上升三位。在社区调查中排名第2。同时，通过数据分析也有足够的数据进入前10名，是我们难以测试和评估风险的已知问题。它是唯一一个没有发生CVE漏洞的风险类别。因此，默认此类别的利用和影响权重值为5.0。原类别命名为“A09:2017-Using Components with Known Vulnerabilities 使用含有已知漏洞的组件”。
A07：2021-身份识别和身份验证错误 Identification and Authentication Failures	排名下滑五位。原标题“A02:2017-Broken Authentication失效的身份认证”。现在包括了更多与识别错误相关的CWE。这个类别仍然是Top 10的组成部分，但随着标准化框架使用的增加，此类风险有减少的趋势。
A08：2021-软件和数据完整性故障 Software and Data Integrity Failures	2021年版的一个新类别，其重点是：在没有验证完整性的情况下做出与软件更新、关键数据和CI/CD管道相关的假设。此类别共有10个匹配的CWE类别，并且拥有最高的平均加权影响值。原“A08:2017-Insecure Deserialization不安全的反序列化”现在是本大类的一部分。
A09：2021-安全日志和监控故障 Security Logging and Monitoring Failures	排名上升一位。来源于社区调查（排名第3）。原名为“A10:2017-Insufficient Logging & Monitoring 不足的日志记录和监控”。此类别现扩大范围，包括了更多类型的、难以测试的故障。此类别在CVE/CVSS 数据中没有得到很好的体现。但是，此类故障会直接影响可见性、事件告警和取证。
A10：2021-服务端请求伪造 Server-Side Request Forgery	2021年版的一个新类别，来源于社区调查（排名第1）。数据显示发生率相对较低，测试覆盖率高于平均水平，并且利用和影响潜力的评级高于平均水平。加入此类别风险是说明：即使目前通过数据没有体现，但是安全社区成员告诉我们，这也是一个很重要的风险。

2021版方法论

2021年版OWASP Top 10的编制比以往更受数据驱动，但又并非盲目地受数据驱动。我们从公开收集的数据中选定了8个类别，又从Top 10社区调查结果中选择了2个高级别的类别，组成了10个类别。我们这样做是为了一个根本原因，通过查看收集到的数据来回顾过去。因为应用安全研究人员寻找新的漏洞和测试它们的新方法需要时间，将这些测试集成到工具和流程中也需要时间。当我们能够可靠地大规模测试弱点时，可能已经过去了很长时间。为了平衡这种观点，我们使用社区调查来向一线应用程序安全和开发专家征求意见，询问他们认为数据可能尚未体现出来的、极其重要的弱点。

我们采用了一些关键变化来增加Top 10的成熟度。

- **类别是如何构建的?**
- **如何使用数据选择类别?**
- **为什么不只是单纯的统计数据?**
- **为什么是事件发生率而不是频率?**
- **您的数据收集和分析过程是怎样的?**
- **数据要素**

类别是如何构建的?

与以往版本的OWASP Top 10相比，本版本的一些类别发生了变化。以下是类别变化的简要描述。

以前的数据收集工作集中在大约30个CWE的规定子集上，并有一个要求额外发现的领域。我们了解到，组织将主要关注规定的30个CWE，且很少添加他们看到的其他CWE。在本次版本更新过程中，我们打破这种方法，仅询问数据，且对CWE没有限制。我们查询了特定年份（从2017年开始）测试的应用程序数量，以及在测试中发现至少一个CWE实例的应用程序数量。这种格式使我们能够跟踪每个CWE在应用程序群体中的流行程度。为了达到我们的目的，我们放弃使用了频率；虽然在其他情况下可能有必要，但它仅隐藏了应用程序群体中的实际流行率。一个应用是否具有4个CWE实例或4000个实例，这都不属于Top 10计算的范围。我们从大约30个CWE增加到近400个CWE，以在数据集中进行分析（我们计划在未来做额外的数据分析作为补充）。而CWE数量的显着增加导致我们需要改变类别的结构。

我们花了几个月的时间对CWE进行分组和分类，本可以再持续几个月，但我们不得不在某个时候停下来。CWE有“根本原因”类型和“症状”类型，其中，“根本原因”类型如“Cryptographic Failure加密机制失效”和“Misconfiguration配置错误”，与“Sensitive Data Exposure敏感信息泄露”和“Denial of Service拒绝服务”等“症状”类型形成对比。我们决定尽可能关注“根本原因”，因为根据“根本原因”来提供识别和补救指导更合乎逻辑。关注“根本原因”而不是“症状”并不是一个新概念。Top 10是“症状”和“根本原因”的混合体。CWE也是“症状”和“根本原因”的混合体；我们只是更加深思熟虑并把它体现出来。本版本中，每个类别平均有19.6个CWE，从“A10: 2021-服务器端请求伪造（SSRF）”的1个CWE到“A04: 2021-不安全设计”中的40个CWE 。这种更新的类别结构也有利于人员培训，因为可以更专注于对语言或框架有意义的 CWE。

如何使用数据选择类别?

在2017年版中，按风险的发生率选择类别以确定可能性，然后根据数十年间有关可利用性、可检测性（也称为可能性）和技术影响的经验，通过团队讨论对它们进行排名。如果可能的话，对于2021年版，我们希望使用“可利用性”和“（技术）影响”的数据。

我们下载了OWASP Dependency Check并提取了CVSS Exploit，以及按相关CWE分组的影响分数。由于所有CVE都有CVSSv2分数，因此需要进行大量研究和努力，但CVSSv2中存在CVSSv3应该解决的缺陷。在某个时间点之后，所有CVE也会被分配一个CVSSv3分数。此外，在CVSSv2和CVSSv3之间更新了评分范围和公式。

在 CVSSv2 中，“可利用性”和“技术影响”都可达到10.0分，但公式会将它们降低到“可利用性”的60%和“技术影响”的40% 。在CVSSv3中，“可利用性”的理论最大值限制为6.0，“技术影响”的理论最大值限制为4.0。考虑到权重，“技术影响”得分上升了，在 CVSSv3中平均下降了近一个半点，而“可利用性”平均下降了近半个点。

从OWASP Dependency Check提取的NVD数据中，有12.5万条CVE记录映射到CWE，有241个唯一的CWE映射到CVE。6.2万条CWE映射有一个CVSSv3分数，大约是数据集中的一半。

对于2021年版OWASP Top 10，我们按以下方式计算平均“可利用性”和“技术影响”分数。我们将所有具有CVSS分数的CVE按CWE分组，并按拥有CVSSv3分数的总数百分比和CVSSv2分数的剩余总数加权利用率和影响得分，以获得总体平均分。我们将这些平均值映射到数据集中的CWE，以用作风险等式另一半的“可利用性”和“技术影响”评分。

为什么不只是单纯的统计数据?

数据结果主要受限于我们以自动化方式测试获得的内容。与经验丰富的应用安全专业人士交谈，他们会告诉我们他们发现尚未出现在数据中的的弱点和他们看到的趋势。人们需要时间为某些缺陷类型开发测试方法，然后需要更多时间让这些测试自动化并针对大量应用程序运行。我们发现的一切都是回顾过去，并且可能会遗漏去年的趋势，因为这些趋势在数据中并不存在。

因此，为了完整性，我们只从数据中挑选了10个类别中的8个，其他2个类别来自Top 10 社区调查。它允许一线从业者投票选出他们认为可能没有在数据中（并且可能永远不会在数据中表达）的最高风险。

为什么是事件发生率而不是频率?

有三个主要数据来源。我们将它们识别为“人工辅助的工具（HaT）”、“工具辅助的人工(TaH)”和原始的“工具”。

工具和HaT是高频的缺陷查找生成器。工具将寻找特定的漏洞并不知疲倦地尝试找到该漏洞的每个实例，并将为某些缺陷类型生成高发现计数。看看跨站脚本，它通常是两种风格之一：它要么是更小的、孤立的错误，要么是系统性问题。当它是一个系统性问题时，单个应用程序的发现计数可能会达到数千个。这种高频率淹没了报告或数据中发现的大多数其他漏洞。

另一方面，TaH会发现更广泛的漏洞类型，但由于时间限制，发现频率要低得多。当人们测试应用程序并看到诸如跨站脚本之类的东西时，他们通常会发现三四个实例并停止。他们可以确定一个系统性的发现，并用建议将其写下来，以在应用程序范围内进行修复。没有必要（或花时间）找到每个实例。

假设我们采用这两个不同的数据集并尝试按频率合并它们。在这种情况下，工具和HaT数据将淹没更准确（但广泛）的TaH数据，这也是为什么跨站脚本之类的东西在许多列表中排名如此之高的原因，而影响通常是低到中等。这是因为发现的数量庞大。（跨站脚本也相当容易测试，因此还有更多测试）。

在2017年版中，我们引入了使用发生率来重新审视数据，并将工具和HaT数据与TaH数据干净地合并。事件发生率是指应用程序总数中至少有一个缺陷类型实例的百分比。我们不在乎它是一次性的还是系统性的。这与我们的目的无关；我们只需要知道有多少应用程序至少有一个实例，这有助于更清晰地了解测试结果，而不会淹没在高频结果中的数据。这对应于与风险相关的观点，因为攻击者只需要一个实例即可通过该类别成功攻击应用程序。

数据收集和分析过程是怎样的?

在2017年的Open Security Summit上正式确定了OWASP Top 10的数据收集流程。OWASP Top 10的领导者们和社区花了两天时间制定了透明的、正式化的数据收集流程。2021版是我们第二次使用这种方法。我们通过项目、OWASP社交媒体渠道发布数据征集的消息。在OWASP项目的网页上，我们列出了我们正在寻找的数据元素和结构以及如何提交它们。在GitHub项目中，我们提供用作模板的示例文件。我们根据需要与不同的组织开展合作，帮助确定结构并匹配到CWE。

我们从不同组织获得数据，包括商业测试工具厂商、漏洞赏金供应商以及贡献内部测试数据的组织。获得数据后，我们将其加载在一起，并对CWE匹配到风险类别的内容进行基本分析。一些CWE之间存在重叠，而另一些则非常密切相关（例如加密漏洞）。我们公开透明地记录和发布与提交的原始数据相关的任何决定，以说明如何规范化数据。

我们研究了Top 10中事件发生率最高的8个类别，并纳入2021 版Top 10。我们还查看了Top 10社区调查结果，看看哪些类别可能已经存在于数据中。数据中尚未出现的前两名投票被选为Top 10中的其他两个名额。在10个类别被全部选中后，我们应用广义的可利用性和影响因素，按照基于风险的顺序对2021年版的Top 10进行排名。

数据要素

每个Top 10类别中列出的数据要素，其含义如下：

- 匹配的CWE种类：被Top 10团队匹配到CWE的类别数量。
- 事件发生率：该组织当年测试的应用程序总数中受该CWE影响的应用程序百分比。
- （测试）覆盖率：所有组织针对特定CWE进行应用程序测试的百分比。
- 加权漏洞可利用性：将CVSSv2和CVSSv3分数分配给CVE所形成的“可利用性”子分数，映射到CWE，规范化，并设置在10分的范围内。
- 加权影响值：将CVSSv2和CVSSv3分数分配给CVE所形成的“影响”子分数，映射到CWE，规范化，并设置在10分的范围内。
- 总发生漏洞应用数：发现可以将CWE匹配为该类别的应用程序总数。
- 总CVE数：NVD数据库中映射到CWE的CVE总数，并映射到一个类别。

鸣谢

感谢项目数据贡献者

以下组织（连同一些匿名数据贡献者）为项目贡献了超50万个应用程序的数据，使其成为最大、最全面的应用程序安全数据集。若没有您们，该工作是不可能完成的。

- AppSec Labs
- Cobalt.io
- Contrast Security
- GitLab
- HackerOne
- HCL Technologies
- Micro Focus
- PenTest-Tools
- Probely
- Sqreen
- Veracode
- WhiteHat (NTT)

感谢项目赞助商

2021 OWASP Top 10 团队衷心感谢 Secure Code Warrior 和Just Eat的财务支持。

项目发布说明

OWASP Top 10 2021于2021年9月24日发布。

主要作者

- Andrew van der Stock (twitter: @vanderaj)
- Brian Glas (twitter: @infosecdad)
- Neil Smithline (twitter: @appsecneil)
- Torsten Gigler (twitter: @torsten_tweet)

其他参与者

- Orange Tsai @orange_8361， “A10-2021: Server Side Request Forgery”作者
- Jim Manico @manicode和 Jakub Maćkowski @kubamackowski - OWASP CheatSheets 协调

您可以如何帮助我们?

在这个阶段，我们需要：

- 数据科学家 - 请同行审查我们的分析
- 网页设计师 - 我们需要制作一个适合移动设备的版本
- 翻译人员 - 请检查英文文本以确保其可翻译
- ASVS、测试指南、代码审查指南项目的领导人 - 请使用我们的数据并帮助我们将我们的文档和标准链接在一起

如何反馈意见或是建议?

请通过以下渠道提交任何更正或问题：

- https://github.com/OWASP/Top10/issues

中文版参与者

曹晓俊、郭振新、呼和、黄圣强、刘国强、王颉、吴楠、许飞、肖文棣、张坤、Rip（排名不分先后，按姓氏拼音排列）

由于中文版参与者水平有限，存在的错误敬请指正。邮箱：project@owasp.org.cn

中文版发布说明

- 截止2021年12月17日，2021年版OWASP Top 10仅在指定网站上发布内容，未提供文档版本。本中文版文档由上述参与者汇编形成。
- 英文版未对“paved road”提供定义。中文版参与者结合自身理解，将其翻译为“AppSec规划”。

OWASP TOP10

02 推荐使用方式

如何将 OWASP Top 10 作为一个标准使用

OWASP Top 10主要是一个提高意识的文档项目。但是，自2003年首次发布以来，很多组织都将其用作事实上的行业AppSec标准。 如果您想使用OWASP Top 10作为编码或测试标准，首先必须明确OWASP Top 10仅是标准的起点，是最低要求。

使用OWASP Top 10作为标准的难点之一是我们记录了应用安全风险，而这些风险不一定都是易于测试的。例如，“A04:2021-Insecure Design 不安全设计”超出了大多数测试形式的范围。另一个例子是就地、使用中的测试，有效的日志记录和监控只能通过访谈和请求有效事件响应的样本来完成。静态代码分析工具可以查找日志记录的缺失，但可能无法确定业务逻辑或访问控制是否记录了严重的安全漏洞。渗透测试人员可能只能确定他们在测试环境中调用了事件响应，而很少以与生产相同的方式对其进行监控。

以下是我们对在何时适合使用OWASP Top 10的建议：

使用场景	OWASP Top 10 2021	OWASP ASVS
意识	是	
培训	入门级	综合
设计和架构	偶尔	是
编码标准	最低限度	是
安全代码审查	最低限度	是
同行评审清单	最低限度	是
单元测试	偶尔	是
集成测试	偶尔	是
渗透测试	最低限度	是
工具支持	最低限度	是
安全供应链	偶尔	是

我们鼓励所有想要采用应用程序安全标准的人们使用OWASP Application Security Verification Standard (ASVS)（即，OWASP应用程序安全验证标准），因为它被设计为可验证和测试的，并且可用于安全开发生命周期的所有部分。

OWASP ASVS是工具供应商唯一可接受的选择。参考“A04:2021-Insecure Design 不安全设计”可以看出，OWASP Top 10项目中某几类风险的性质决定了工具无法全面检测、测试或防范OWASP Top 10的全部风险。鉴于以上原因，任何对OWASP Top 10进行全面防范或测试的说法，都是不真实的。

如何使用 OWASP Top 10 启动一个应用安全项目

在以前，OWASP Top 10从未被设计作为应用安全项目（AppSec）的基准。但是，对于许多刚刚启动应用安全工作的组织来说，必然要有一个最初的参考基准。2021 版OWASP Top 10 可以作为检查清单等的基线，但必须要注意，它本身并不能覆盖所有范围。

阶段1. 确定您的应用安全计划的差距和目标

- 许多应用安全（AppSec）计划不切实际，不会爬就想跑，这注定要失败。我们强烈鼓励CISO和AppSec领导层使用 OWASP软件保障成熟度模型（OWASP Software Assurance Maturity Model (SAMM)）来确定1-3年内的弱点和需要改进的领域。第一步是评估您现在所处的位置，确定您在治理、设计、实施、验证和运营方面需要立即解决的的差距、与可以后续再解决的差距，并优先实施或改进OWASP SAMM的15项安全实践。OWASP SAMM可以帮助您构建和衡量软件保障工作的改进。

阶段2. 为规划安全开发生命周期铺平道路

- 从传统意义上来说，这是所谓的“独角兽”区域，而铺平道路（即，建立AppSec规划）的办法是用最简单的方式，跟上开发团队壮大的速度来扩张AppSec资源。
- AppSec规划的概念是“最简单的方法也是最安全的方法”，并且应该涉及开发团队和安全团队之间深度合作的文化，最好是他们是同一个团队。铺平道路旨在通过拥有一个企业范围内的制品库，通过不断改进、测量、检测来替换不安全的制品，并使用工具帮助查看通过采用铺平道路可以在哪些方面进行改进。这允许现有的开发工具报告不安全构建的情况，并帮助开发团队自我纠正，远离不安全的制品。
- AppSec规划可能看起来可以有很多，但它应该随着时间的推移逐步构造。还有其他形式的AppSec计划，特别是Microsoft Agile Secure Development Lifecycle。并非每种AppSec计划都适合所有业务。

阶段3. 与您的开发团队一起执行AppSec规划

- AppSec规划应该是在相关的开发团队和运营团队的同意和直接参与下建造的。AppSec规划应与业务战略保持一致，并有助于更快地交付更安全的应用程序。建设AppSec规划应该是涵盖整个企业或应用程序生态系统的整体行为，而不是像过去那样靠事后“贴创可贴”的形式对每个应用的安全保障修修补补。

阶段4. 将所有即将到来和现有的应用程序转移到AppSec规划上

- 在AppSec规划上添加检测工具，并向开发团队提供信息，以帮助他们直接利用AppSec规划中的元素来提高其应用程序的安全性。一旦AppSec规划的某个方面被采用，组织应实施持续集成检查，检查现有代码和使用禁止替代品的签入，并警告或拒绝构建或签入。这可以防止不安全的选项随着时间的推移蔓延到代码中，防止技术欠债、形成带有安全缺陷的应用。此类警告应链接到安全替代方案，以便开发团队立即得到正确答案。他们可以快速重构和采用铺好的道路组件。

如何使用 OWASP Top 10 启动一个应用安全项目

阶段5. 测试AppSec规划是否减轻了OWASP Top 10 中发现的问题

- AppSec规划的内容应该能够解决OWASP Top 10中的典型问题，例如，如何自动检测或修复易受攻击的组件，或者使用静态代码分析IDE插件来检测注入，甚至更好地开始使用已知安全的库来防止注入。提供给团队的这些安全的替代品越多越好。AppSec团队的一项重要任务是确保这些软件产品组件的安全性得到持续评估和改进。一旦它们得到改进，应通过特定的沟通渠道向软件产品客户表明软件产品应该尽快升级。最好是能自动升级，但如果不是，则至少在软件产品的仪表板或类似界面上明显显示。

阶段6. 将您的计划构建为成熟的AppSec计划

- 您不能止步于OWASP Top 10。它仅涵盖10个风险类别。我们强烈鼓励组织采用OWASP ASVS，并根据开发的应用程序的风险级别逐步添加AppSec规划的组成要素和对应级别1、级别2 和级别3的测试。

持续改进

所有出色的AppSec计划都超越了最低限度。如果我们要掌握应用软件的安全缺陷，每个人都必须继续前进。

- **概念完整性。**成熟的AppSec计划必须包含一些安全架构的概念，无论是正式的云还是企业安全架构或威胁建模。
- **自动化和规模化。**成熟的AppSec计划尝试尽可能多地自动化其可交付成果，使用脚本来模拟复杂的渗透测试步骤、开发团队直接可用的静态代码分析工具、协助开发团队构建AppSec单元和集成测试等。
- **文化。**成熟的AppSec计划试图通过成为开发团队的一部分而不是旁观者来构建不安全设计并消除现有代码的技术债务。将开发团队视为“我们”和“他们”的AppSec团队注定要失败。
- **持续改进。**成熟的AppSec计划希望不断改进。如果某件事不起作用，请停止这样做。如果某些东西笨拙或不可扩展，请努力改进它。如果开发团队没有使用某些东西并且没有影响或影响有限，请做一些不同的事情。仅仅因为我们自1970年代以来就进行了诸如桌面检查之类的测试，但这并不意味着这是一个好主意。测量、评估，然后构建或改进。

03 2021 OWASP Top 10 清单

A01：2021-失效的访问控制
Broken Access Control

风险因素

匹配CWE种类	最大事件发生率	平均事件发生率	平均加权漏洞可利用性	平均加权影响值	最大测试覆盖率	平均测试覆盖率	总发生漏洞应用数	总CVE数
34	55.97%	3.81%	6.92	5.93	94.55%	47.72%	318487	19013

风险概述

从第五位上升到第一位，94%的应用程序都接受了某种形式的针对“失效的访问控制”的测试，该事件的平均发生率为 3.81%，该漏洞在提供的数据集中出现漏洞的应用数量最多，总发生漏洞应用数量超过31.8万多次。 此类值得注意的常见CWE包括： CWE-200: Exposure of Sensitive Information to an Unauthorized Actor（将敏感信息泄漏给未经授权的参与者）、CWE-201: Exposure of Sensitive Information Through Sent Data（通过发送的数据泄漏敏感信息）、CWE-352: Cross-Site Request Forgery （跨站请求伪造）。

风险说明

访问控制强制实施策略，使用户无法在其预期权限之外进行操作。失败的访问控制通常会导致未经授权的信息泄露、修改或销毁所有数据、或在用户权限之外执行业务功能。常见的访问控制脆弱点包括：

- 违反最小特权原则或默认拒绝原则，即访问权限应该只授予特定能力、角色或用户，但实际上任何人都可以访问。
- 通过修改 URL（参数篡改或强制浏览）、内部应用程序状态或 HTML 页面，或使用修改 API 请求的攻击工具来绕过访问控制检查。
- 通过提供唯一标识符（不安全的直接对象引用）允许查看或编辑其他人的帐户
- API没有对POST、PUT 和DELETE强制执行访问控制。
- 特权提升。 在未登录的情况下假扮用户或以用户身份登录时充当管理员。
- 元数据操作，例如通过重放或篡改 JSON Web 令牌 (JWT) 来访问控制令牌，或操纵cookie 或隐藏字段以提升权限，或滥用 JWT 失效。
- CORS 配置错误以致允许未授权或不可信的API访问。
- 以未通过身份验证的用户身份强制浏览的通过身份验证时才能看到的页面或作为标准用户身份访问特权页面。

A01：2021-失效的访问控制
Broken Access Control

预防措施

访问控制只在受信服务器端代码或无服务器API中有效，这样攻击者才无法修改访问控制检查或元数据。

- 除公有资源外，默认为“拒绝访问”。
- 使用一次性的访问控制机制，并在整个应用程序中不断重用它们，包括最小化跨源资源共享（CORS）的使用。
- 建立访问控制模型以强制执行所有权记录，而不是简单接受用户创建、读取、更新或删除的任何记录。
- 特别的业务应用访问限制需求应由领域模型强制执行。
- 禁用Web服务器目录列表，并确保文件元数据（例如：.git）和备份文件不存在于Web的根目录中。
- 在日志中记录失败的访问控制，并在适当时向管理员告警（例如：重复故障）。
- 对API和控制器的访问进行速率限制，以最大限度地降低自动化攻击工具带来的危害。
- 当用户注销后，服务器上的状态会话标识符应失效。无状态的JWT令牌应该是短暂的，以便让攻击者的攻击机会窗口最小化。 对于时间较长的JWT，强烈建议遵循OAuth标准来撤销访问。

开发人员和QA人员应进行访问控制功能的单元测试和集成测试。

攻击范例

范例 #1 应用程序在访问帐户信息的SQL调用中使用了未经验证的数据：

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

攻击者只需修改浏览器的“acct”参数即可发送他们想要的任何帐号。如果没有正确验证，攻击者可以访问任何用户帐户。

https://example.com/app/accountInfo?acct=notmyacct

范例 #2 攻击者仅强制浏览目标URL。 访问管理页面一定需要管理员权限。

https://example.com/app/getappInfo

https://example.com/app/admin_getappInfo

如果一个未经身份验证的用户可以访问任何页面，那么这是一个缺陷。

如果一个非管理员权限的用户可以访问管理页面，那么这同样也是一个缺陷。

A02：2021-加密机制失效
Cryptographic Failures

OWASP Open Web Application Security Project

风险因素

匹配CWE种类	最大事件发生率	平均事件发生率	平均加权漏洞可利用性	平均加权影响值	最大测试覆盖率	平均测试覆盖率	总发生漏洞应用数	总CVE数
29	46.44%	4.49%	7.29	6.81	79.33%	34.85%	233788	3075

风险概述

上升一位到第二名，以前称为“敏感数据泄漏”。“敏感数据泄漏”更像是一种常见的表象问题而不是根本原因，这项风险重点是与加密机制相关的故障（或缺乏加密机制）。 这往往会导致敏感数据泄漏。 值得注意的常见CWE包括：CWE-259: Use of Hard-coded Password （使用硬编码密码）、CWE-327: Broken or Risky Crypto Algorithm（损坏或有风险的加密算法）、CWE-331 Insufficient Entropy （熵不足）。

风险说明

首先要确认：对传输中数据和存储数据都有哪些保护需求。 例如，密码、信用卡号、医疗记录、个人信息和商业秘密需要额外保护，尤其是在这些数据属于隐私保护法（如：欧盟GDPR）或法规条例（如：金融数据保护标准PCI DSS)适用范围的情况下。 对于这些数据，要确定：

- 在传输数据过程中是否使用明文传输？这和传输协议有关，如：HTTP、SMTP、经过TLS升级（如STARTTLS ）的FTP。外部网络流量是有害的。需要验证所有的内部通信，如，负载平衡、Web服务器或后端系统之间的流量。
- 无论是在默认情况下还是在旧的代码中，是否还在使用任何旧的或脆弱的加密算法或传输协议？
- 是否使用默认加密密钥、生成或重复使用脆弱的加密密钥，或者是否缺少适当的密钥管理或密钥回转？加密密钥是否已经提交到源代码存储库？
- 是否未执行强制加密，例如，是否缺少安全相关的HTTP（浏览器）指令或标头？
- 接收到的服务器证书和信任链是否经过正确验证？
- 初始化向量是否忽略，重用或生成的密码操作模式是否不够安全？ 是否正在使用不安全的操作模式，例如欧洲央行正在使用的操作模式？ 当认证加密更合适时是否使用加密？
- 在缺乏密码基密钥派生函数的情况下，是否将密码用作加密密钥？
- 随机性是否用于并非旨在满足加密要求的加密目的？ 即使选择了正确的函数，它是否需要由开发人员播种，如果不需要，开发人员是否用缺乏足够熵/不可预测性的种子覆盖了内置的强大播种功能？
- 是否使用过时的散列函数，例如 MD5 或 SHA1，或者在散列函数需要加密时使用非加密散列函数？
- 是否在使用已弃用的加密填充方法，例如 PCKS number 1 v1.5？
- 加密的错误消息或侧信道信息是否可以利用，例如使用填充预言机攻击的形式？

请参考 OWASP ASVS中的Crypto加密 (V7)、Data Protection数据保护(V9) 和 SSL/TLS (V10)。

A02：2021-加密机制失效
Cryptographic Failures

OWASP Open Web Application Security Project

预防措施

至少执行以下操作，并查阅参考资料：

- 对应用程序处理、存储或传输的数据分类，并根据隐私法、监管要求或业务需求确定哪些数据是敏感的。
- 对于没有必要存储的敏感数据，应当尽快清除，或者通过PCI DSS标记化或拦截。 未存储的数据不能窃取。
- 确保加密存储的所有敏感数据。
- 确保使用了最新的、强大的标准算法、协议和密钥；并且密钥管理到位。
- 确保加密传输过程中的数据，如使用安全协议（例如具有前向保密（FS）密码的 TLS、服务器的密码优先级和安全参数）。确保强制执行数据加密，如使用HTTP 严格安全传输协议 (HSTS) 等指令。
- 禁用缓存对包含敏感数据的响应。
- 根据数据分类应用实施所需的安全控制。
- 不要使用FTP 和SMTP 等传统协议来传输敏感数据。
- 使用具有工作因子（延迟因子）的强自适应和加盐散列函数存储密码，例如 Argon2、scrypt、bcrypt 或 PBKDF2。
- 必须选择适合操作模式的初始化向量。对于大多数模式，可以使用CSPRNG（密码安全伪随机数生成器）。对于需要随机数的模式，则初始化向量 (IV) 不需要使用CSPRNG。 在所有情况下，对于一个固定密钥，永远不应该使用两次IV。
- 始终使用经过验证的加密，而不仅仅是加密。
- 密钥应以加密方式随机生成并作为字节数组存储在内存中。 如果使用密码，则必须通过适当的密码基密钥派生函数将其转换为密钥。
- 确保在适当的地方使用加密随机性，并且没有以可预测的方式或低熵进行播种。 大多数现代 API 不需要开发人员为 CSPRNG 设置种子以获得安全性。
- 避免使用的已废弃的加密函数和填充方案，例如 MD5、SHA1、PKCS number 1 v1.5。
- 单独验证每个安全配置项的有效性。

攻击范例

范例 #1

一个应用程序使用自动化的数据加密系统加密存储中数据库中的信用卡号。 但是，这些数据在检索时会自动解密，这就使得SQL 注入漏洞以明文形式获得信用卡号。

范例 #2

一个网站对所有网页没有使用或对强制使用TLS，或者使用弱加密。 攻击者经过监测网络流量（例如，在不安全的无线网络中），将网络连接从 HTTPS 降级到 HTTP，就可以拦截请求并窃取用户会话cookie。 之后，攻击者重放这个 cookie 并劫持用户的（经过身份验证的）会话，访问或修改用户的私人数据。 除此之外，攻击者还可以更改所有传输过程中的数据，例如，汇款的接收人。

范例 #3

密码数据库使用未加盐或弱哈希算法来存储每个人的密码。一个文件上传漏洞允许攻击者获取密码数据库。 所有未加盐哈希的密码都可以通过预先计算的哈希值彩虹表破解。 由简单或快速散列函数生成的加盐哈希也可能通过GPU 破解。

A03：2021-注入 Injection

风险因素

匹配CWE种类	最大事件发生率	平均事件发生率	平均加权漏洞可利用性	平均加权影响值	最大测试覆盖率	平均测试覆盖率	总发生漏洞应用数	总CVE数
33	19.09%	3.37%	7.25	7.15	94.04%	47.90%	274228	32078

风险概述

注入降至第三。94%的统计应用针对某种形式的注入进行了测试，最大发生率为19%，平均发生率为3%，共计发生了27.4万次。值得注意的常见弱点枚举（CWE）包括CWE-79： Cross-site Scripting（跨站点脚本）、CWE-89：SQL Injection（SQL注入）和CWE-73：External Control of File Name or Path（文件名或路径的外部控制）。

风险说明

在以下情况下，应用程序易受攻击：

- 应用程序不会验证、过滤或清洗用户提供的数据。
- 动态查询或无上下文感知转义的非参数化调用直接在解释器中使用。
- 恶意数据在对象关系映射（ORM）搜索参数中用于提取额外的敏感记录。
- 恶意数据被直接使用或连接。SQL或命令包含动态查询、命令或存储过程中的结构和恶意数据。

一些更常见的注入包括：SQL、NoSQL、OS命令、对象关系映射（ORM）、LDAP和表达式语言（EL）或对象图导航库（OGNL）注入。这一概念在所有解析器中都是相同的。源代码审查是检测应用程序是否易受注入攻击的最佳方法。强烈鼓励针对所有参数、标题、URL、cookies、JSON、SOAP和XML数据输入的自动测试。组织可以在CI/CD管道中引入静态（SAST）、动态（DAST）和交互式（IAST）应用程序安全测试工具，以在产品部署之前识别引入的注入缺陷。

A03：2021-注入
Injection

预防措施

防止注入需要将数据与命令和查询分开：

- 推荐的选择是使用安全的API，这样可以避免完全使用解释器、提供参数化接口或迁移到对象关系映射工具（ORM）。注意：即使参数化，如果PL/SQL或T-SQL将查询和数据连接起来，或者使用EXECUTE IMMEDIATE或exec（）执行恶意数据，则存储过程仍然可以引入SQL注入。
- 使用肯定（positive）或“白名单”服务器端输入验证。这并不是一种完美的防御，因为许多应用程序需要特殊字符，例如移动应用程序中的文本区域或API。
- 对于任何残余的动态查询，请使用该解释器的特定转义语法转义特殊字符。**注意：**SQL结构（如表名、列名等）无法转义，因此用户提供的结构名是危险的。这是报表编写软件中的常见问题。
- 在查询中使用LIMIT和其他SQL控件，以防止在SQL注入的情况下大量披露记录。

攻击范例

范例 #1

应用程序在构造以下易受攻击的SQL调用时使用不受信任的数据：

```
String query = "SELECT \* FROM accounts WHERE custID='" + request.getParameter("id") + "'";
```

范例 #2

类似地，应用程序对框架的盲目信任可能会导致易受攻击的查询（例如，Hibernate查询语言（HQL））

```
Query HQLQuery = session.createQuery("FROM accounts WHERE custID='" + request.getParameter("id") + "'");
```

在这两种情况下，攻击者都会修改浏览器中的“id”参数值，以发送：'或'1'='1。例如：

```
http://example.com/app/accountView?id=' or '1'='1
```

这将更改两个查询的含义，以返回accounts表中的所有记录。更危险的攻击可能会修改或删除数据，甚至调用存储过程。

A04：2021-不安全设计
Insecure Design

风险因素

匹配CWE种类	最大事件发生率	平均事件发生率	平均加权漏洞可利用性	平均加权影响值	最大测试覆盖率	平均测试覆盖率	总发生漏洞应用数	总CVE数
40	24.19%	3.00%	6.46	6.78	77.25%	42.51%	262407	2691

风险概述

2021年版的一个新类别，侧重于与设计和体系结构缺陷相关的风险，呼吁更多地使用威胁建模、安全设计模式和参考体系结构。作为一个应用安全技术社区，我们需要超越软件开发过程编码空间中的“左移”，转向对设计安全原则至关重要的预编码活动。值得注意的常见弱点枚举（CWE）包括CWE-209：Generation of Error Message Containing Sensitive Information（生成包含敏感信息的错误消息）、CWE-256：Unprotected Storage of Credentials（凭证的未保护存储）、CWE-501：Trust Boundary Violation（信任边界冲突）和CWE-522：Insufficiently Protected Credentials（凭证保护不足）。

风险说明

不安全设计是一个广泛的类别，代表不同的弱点，表示为“缺少或无效的控制设计”。不安全设计不是所有其他前10个风险类别的来源。不安全设计和不安全的实现之间存在差异。我们区分设计缺陷和实现缺陷是有原因的，它们有不同的根本原因和补救措施。安全设计仍然可能存在实现缺陷，从而导致可能被利用的漏洞。一个不安全设计不能通过一个完美的实现来修复，因为根据定义，所需的安全控制从未被创建来抵御特定的攻击。导致不安全设计的因素之一是开发的软件或系统中缺乏固有的业务风险分析，因此无法确定需要何种级别的安全设计。

- **需求和资源管理**

收集应用程序的业务需求并与业务部门协商，包括所有数据资产和预期业务逻辑的机密性、完整性、可用性和真实性方面的保护需求。考虑应用程序的公开程度，以及是否需要租户隔离（除访问控制外）。编制技术要求，包括功能性和非功能性安全要求。计划和协商所有设计、建造、测试和运营的预算，包括安全活动。

- **安全设计**

安全设计是一种文化和方法，它不断评估威胁，并确保代码经过稳健的设计和测试，以防止已知的攻击方法。威胁建模应整合到细化会议（或类似活动）中；查看数据流和访问控制或其他安全控制中的更改。在用户故事开发中，确定正确的流程和故障状态，确保责任方和受影响方充分理解并同意这些状态。分析预期和故障流的假设和条件，确保其仍然准确和可取。确定如何验证正确行为所需的假设和实施条件。确保结果记录在用户故事中。从错误中吸取教训，并提供积极的激励以促进改进。安全设计既不是附加组件，也不是可以添加到软件中的工具。

- **安全开发生命周期**

安全的软件需要安全开发生命周期、某种形式的安全设计模式、AppSec规划方法、安全的组件库、工具和威胁建模。在整个项目和软件维护过程中，在软件项目开始时联系您的安全专家。考虑利用OWASP软件保证成熟度模型（SAMM）来帮助构建您的安全软件开发工作。

A04：2021-不安全设计
Insecure Design

预防措施

- 与应用安全专业人员建立并使用安全的开发生命周期，以帮助评估和设计与安全和隐私相关的控制。
- 建立并使用安全设计模式的库，或使用AppSec规划中现有的要素。
- 对关键身份验证、访问控制、业务逻辑和密钥流使用威胁建模。
- 将安全语言和安全控制集成到用户故事中。
- 在应用程序的每一层（从前端到后端）集成合理性检查。
- 编写单元和集成测试，以验证所有关键流是否能够抵抗威胁模型。为应用程序的每一层编译正确用例和误用案例。
- 根据暴露和保护需要，对系统层和网络层进行分层。
- 通过设计在所有层中严格隔离租户。
- 限制用户或服务的资源消耗。

攻击范例

范例 #1

身份凭证恢复工作流可能包括NIST 800-63b 、 OWASP ASVS和OWASP Top 10中已禁止的“问题和答案”功能。“问题和答案”不能作为可信的证据来证明身份，因为不止一个人知道答案，这就是为什么它们被禁止的原因。应删除此类代码，并用更安全的设计替换。

范例 #2

一家连锁电影院允许团体预订折扣，支持最多15名观众而不要押金。攻击者可以对这种流量进行威胁建模，并测试他们是否可以在几个请求中同时预订600个座位和所有电影院，从而造成巨大的收入损失。

范例 #3

零售连锁店的电子商务网站没有针对黄牛党运行的机器人的保护，黄牛党购买高端显卡然后在拍卖网站上销售。这给显卡制造商和零售连锁店老板创造了糟糕的形象，并与那些无法以任何价格获得这些卡的爱好者产生了持久的伤害。适当的反机器人设计和域逻辑规则，例如在可用的几秒钟内进行的购买，可能会被识别为不可信的购买并拒绝此类交易。

A05：2021-安全配置错误
Security Misconfiguration

风险因素

匹配CWE种类	最大事件发生率	平均事件发生率	平均加权漏洞可利用性	平均加权影响值	最大测试覆盖率	平均测试覆盖率	总发生漏洞应用数	总CVE数
20	19.84%	4.51%	8.12	6.56	89.58%	44.84%	208387	789

风险概述

从上一版本的第六名提升，90%的调查应用都进行了某种形式的配置错误测试，平均发生率为4%，并且在此风险类别的CWE出现了超过20.8万次。随着当今软件正转向高度可配置，看到这一类别上升也就不足为奇了。 需要注意的是 CWE包括： CWE-16 Configuration（配置）和 CWE-611 Improper Restriction of XML External Entity Reference（XML 外部实体引用的不当限制）。

风险说明

您的应用程序可能受到攻击，如果应用程序是：

- 应用程序栈的任何部分缺少适当的安全加固，或者云服务的权限配置错误。
- 应用程序启用或安装了不必要的功能（例如：不必要的端口、服务、网页、帐户或权限）。
- 默认帐户和密码仍然可用且没有更改。
- 错误处理机制向用户纰漏堆栈信息或其他大量错误信息。
- 对于升级的系统，最新的安全特性被禁用或未安全配置。
- 应用程序服务器、应用程序框架（如：Struts、Spring、ASP.NET）、库文件、数据库等没有进行安全配置。
- 服务器不发送安全标头或指令，或未被设定安全参数。
- 您的应用软件已过期或易受攻击（参见“A6:2021-脆弱和过时的组件”）。

缺少一个体系的、可重复的应用程序安全配置过程，系统将处于高风险中。

A05：2021-安全配置错误
Security Misconfiguration

预防措施

应实施安全的安装过程，包括：

- 一个可以快速且易于部署在另一个锁定环境的可重复的加固过程。开发、质量保证和生产环境都应该进行相同配置，并且在每个环境中使用不同的密码。这个过程应该是自动化的，以尽量减少安装一个新安全环境的耗费。
- 搭建最小化平台，该平台不包含任何不必要的功能、组件、文档和示例。移除或不安装不适用的功能和框架。
- 检查和修复安全配置项来适应最新的安全说明、更新和补丁，并将其作为更新管理过程的一部分（参见“A6:2021-脆弱和过时的组件”）。在检查过程中，应特别注意云存储权限（如：S3桶权限）。
- 一个能在组件和用户间提供有效的分离和安全性的分段应用程序架构，包括：分段、容器化和云安全组（ACL）。
- 向客户端发送安全指令，例如：安全标头。
- 一个能够验证所有环境中进行了正确的安全配置和设置的自动化过程。

攻击范例

范例 #1

应用程序服务器附带了未从产品服务器中删除的应用程序样例。这些样例应用程序具有已知的安全漏洞，攻击者利用这些漏洞来攻击服务器。假设其中一个应用程序是管理员控制台，并且没有更改默认账户，攻击者就可以通过默认密码登录，从而接管服务器。

范例 #2

目录列表在服务器端未被禁用。攻击者发现他们很容易就能列出目录列表。攻击者找到并下载所有已编译的Java类，他们通过反编译来查看代码。然后，攻击者在应用程序中找到一个严重的访问控制漏洞。

范例 #3

应用程序服务器配置允许将详细的错误信息（如：堆栈信息）返回给用户，这可能会暴露敏感信息或潜在的漏洞，如：已知含有漏洞的组件的版本信息。

范例 #4

云服务提供商向其他CSP用户提供默认的网络共享权限。这允许攻击者访问存储在云端的敏感数据。

A06：2021-自带缺陷和过时的组件
Vulnerable and Outdated Components

风险因素

匹配CWE种类	最大事件发生率	平均事件发生率	平均加权漏洞可利用性	平均加权影响值	最大测试覆盖率	平均测试覆盖率	总发生漏洞应用数	总CVE数
3	27.96%	8.77%	5.00	5.00	51.78%	22.47%	30457	0

风险概述

它在Top 10社区调查中排名第二，也有足够的数据让它进入前十。不安全的组件是我们努力测试和评估风险的已知问题，并且它是在CWE中唯一没有任何 CVE 对应的类别，因此，使用默认的利用/影响加权值为5.0。值得注意的是CWE包括：CWE-1104 Use of Unmaintained Third-Party Components（使用未维护第三方组件），以及2013年和2017年Top 10中关于此问题的两个CWE。

风险说明

如果满足下面的某个条件，那么您的应用就易受此类攻击：

- 如果您不知道所有使用的组件版本信息（包括：服务端和客户端）。这包括了直接使用的组件或间接依赖的组件。
- 如果软件易受攻击，不再支持或者过时。这包括：系统、Web服务器、应用程序服务器、数据库管理系统（DBMS）、应用程序、API和所有的组件、运行环境和库。
- 如果您没有定期做漏洞扫描和订阅使用组件的安全公告。
- 如果您不基于风险及时修复或升级底层平台、框架和依赖库。很可能发生这种情况：根据变更控制，每月或每季度进行升级，这使得组织在这段时间内会受到已修复但未修补的漏洞的威胁。
- 如果软件工程师没有对更新的、升级的或打过补丁的组件进行兼容性测试。
- 如果您没有对组件进行安全配置（参见“A05:2021–安全配置错误”）。

A06：2021-自带缺陷和过时的组件
Vulnerable and Outdated Components

预防措施

应制定一个补丁管理流程：

- 移除不使用的依赖、不需要的功能、组件、文件和文档。
- 利用如versions、OWASP Dependency Check、retire.js等工具来持续的记录客户端和服务器端以及它们的依赖库的版本信息。持续监控如CVE和NVD等是否发布已使用组件的漏洞信息，可以使用软件分析工具来自动完成此功能。订阅关于使用组件安全漏洞的警告邮件。
- 仅从官方渠道安全的获取组件，并使用签名机制来降低组件被篡改或加入恶意漏洞的风险（参见“A08:2021-软件和数据完整性故障”）。
- 监控那些不再维护或者不发布安全补丁的库和组件。如果不能打补丁，可以考虑部署虚拟补丁来监控、检测或保护。

每个组织都应该制定相应的计划，对整个软件生命周期进行监控、评审、升级或更改配置。

攻击范例

范例 #1

通常组件都是以与应用相同的权限运行的，这使得组件里的缺陷可能导致各式各样的问题。这些缺陷可能是偶然的（如：编码错误），也可能是蓄意的（如：组件里的后门）。下面是一些已被利用的漏洞：

CVE-2017-5638，一个Struts2远程执行漏洞。可在服务端远程执行代码，并已造成巨大的影响。

虽然物联网（IoT）设备一般难以通过打补丁来修复。但对它打补丁非常重要（例如：医疗设备）。

有些自动化工具能帮助攻击者发现未打补丁的或配置不正确的系统。例如：Shodan IOT搜索引擎能帮助您发现从2014年4月至今仍存在心脏出血漏洞的设备。

A07：2021-身份识别和身份验证错误
Identification and Authentication Failures

风险因素

匹配CWE种类	最大事件发生率	平均事件发生率	平均加权漏洞可利用性	平均加权影响值	最大测试覆盖率	平均测试覆盖率	总发生漏洞应用数	总CVE数
22	14.84%	2.55%	7.40	6.50	79.51%	45.72%	132195	3897

风险概述

之前被称之为“无效的身份认证”，此类别从第二名下滑，现在包含了与身份识别失效相关的CWE，如知名的“CWE-297: Improper Validation of Certificate with Host Mismatch（与不匹配的服务端进行不适当的凭证确认）”，“CWE-287: Improper Authentication（不适当的认证）”，“CWE-384: Session Fixation（会话固定攻击）”。

风险说明

- 允许像是攻击者已经拥有有效用户名称和密码列表的撞库自动化攻击。
- 允许暴力或其他自动化攻击。
- 允许预设、脆弱、常见的密码，像是"Password1"或"admin/admin"。
- 使用脆弱或无效的认证资讯回复或忘记密码的流程，如不安全的"知识相关问答"。
- 使用明码、被加密的或使用较脆弱杂凑法的密码(参考A3：2017-敏感性资料泄漏)。(TODO) https://github.com/OWASP/Top10/issues/553
- 不具有或是无效的多因素认证。
- 于URL中泄漏会话(session) ID(如URL重写)。
- 成功登入后没有轮换会话(session) ID。
- 没有正确的注销会话(session) ID。用户的会话(session)或认证tokens(主要是单一登入(SSO)token) 没有在登出时或一段时间没活动时被适当的注销。

A07：2021-身份识别和身份验证错误
Identification and Authentication Failures

预防措施

- 在可能的情况下，实施多因素认证来防止自动化撞库攻击、暴力破解、以及遭窃认证资讯被重复利用的攻击。
- 不要交付或部署任何预设的认证凭证，特别是管理者。
- 实施弱密码的检查，如测试新设定或变更的密码是否存在于前10000个最差密码清单。
- 将密码长度、复杂度和轮换政策与NIST 800-63b文件“第5.1.1节-被记忆的秘密或其他现代基于证据的密码政策”的内容保持一致。
- 对所有结果使用相同的讯息回应，确保注册、认证凭据回复以及API路径能够抵御帐号枚举攻击。
- 限制或增加登入失败尝试的延迟。记录所有失败并于侦测到撞库、暴力破解或其他攻击时发出告警。
- 使用服器端、安全的内建会话管理器，在登入后产生有高熵值的新随机会话ID。会话ID不应出现在URL中，必须被安全的储存，并且在登出后、闲置、超时后被注销。

攻击范例

情境#1:

使用已知列表密码的撞库攻击是一种常见的攻击方式，假设应用没有实施自动化威胁或撞库攻击的保护，在这种情况下，应用会被利用为密码预报的工具来判断认证凭据是否有效。

情境#2:

大多数身份验证攻击都是由于密码作为单一因素持续使用而发生的。一旦考虑，最佳实践、密码轮换和复杂性要求鼓励用户使用和重用弱密码。建议组织按照NIST 800-63停止这些做法并使用多因素认证。

情境#3:

应用会话超时设置错误。一个用户使用公用电脑来访问应用时，用户没有选择“登出”而是简单的关闭浏览器标签页就离开。一小时后，一个攻击者使用同一个浏览器，而那个应用仍处于前面用户认证通过的状态。

A08：2021-软件和数据完整性故障
Software and Data Integrity Failures

风险因素

匹配CWE种类	最大事件发生率	平均事件发生率	平均加权漏洞可利用性	平均加权影响值	最大测试覆盖率	平均测试覆盖率	总发生漏洞应用数	总CVE数
10	16.67%	2.05%	6.94	7.94	75.04%	45.35%	47972	1152

风险概述

2021年版本的一个新类别，聚焦于在未验证完整性的情况下做出与软件更新、关键数据和CI/CD管道相关的假设。这是来自漏洞披露/通用漏洞评估系统（CVE/CVSS）数据的最高权重影响之一。 值得注意的是，CWE包括：CWE-829：Inclusion of Functionality from Untrusted Control Sphere（包含来自不受信任控制领域的功能），CWE-494：Download of Code Without Integrity Check（不进行完整性检查的代码下载）和CWE-502：Deserialization of Untrusted Data（不可信数据的反序列化）。

风险说明

软件和数据完整性故障与无法防止违反完整性的代码和基础设施有关。这方面的一个例子是，应用程序依赖于不受信任的源、存储库和内容分发网络（CDN）的插件、库或模块。不安全的CI/CD管道可能会带来未经授权的访问、恶意代码或系统安全风险。最后，许多应用程序现在包括自动更新功能。其中，更新包在没有进行充足完整性验证的情况下被下载，并应用于以前受信任的应用程序。攻击者可能会上传自己的更新包，以便在所有安装上分发和运行。另一个例子是，对象或数据被编码或序列化为攻击者可以看到和修改的结构，很容易受到不安全的反序列化的影响。

A08：2021-软件和数据完整性故障
Software and Data Integrity Failures

预防措施

- 使用数字签名或类似机制来验证软件或数据来自预期来源，且未被修改。
- 确保库和依赖项目，如：npm 或 Maven，正在使用受信任的存储库。如果您的风险较高，请考虑托管一个经过审核的、内部已知合格的存储库。
- 确保使用软件供应链安全工具（如：OWASP Dependency Check 或 OWASP CycloneDX）来验证组件不包含已知漏洞。
- 确保对代码和配置更改进行审核，以最大限度地减少恶意代码或配置引入软件管道的可能性。
- 确保您的CI/CD管道具有适当的隔离、配置和访问控制，以确保代码在构建和部署过程中的完整性。
- 确保通过特定形式的完整性检查或数字签名来检测序列化数据是否存在篡改或重播，所有未签名或未加密的序列化数据不会发送到不受信任的客户端。

攻击范例

范例 #1无需签名既可更新

许多家庭路由器、机顶盒、设备固件和其他设备通过没有签名的固件进行更新。未签名固件是越来越多的攻击者的目标，预计情况只会变得更糟。这是一个主要问题，因为很多时候除了在未来的版本中修复和等待以前的版本过期之外，没有任何其他补救措施。

范例#2 SolarWinds恶意更新

众所周知一种新的攻击机制，即，最近值得注意的SolarWinds Orion攻击。开发该软件的公司具有安全的构建和更新完整性流程。尽管如此，这些都能够被颠覆。几个月来，该公司向18,000多个组织分发了一个高度针对性的恶意更新，其中大约100个组织受到影响。这是历史上同类性质中影响最深远和最严重的违规行为之一。

范例 #3 不安全的反序列化

一个React 应用程序调用一组 Spring Boot 微服务。作为函数程序员，他们试图确保他们的代码是不可变的。他们提出的解决方案是将用户状态序列化，并在每个请求中来回传递。攻击者注意到“rO0”Java 对象签名（在 base64 中），并使用 Java Serial Killer工具在应用服务器上获取远程代码执行权。

A09：2021-安全日志和监控故障
Security Logging and Monitoring Failures

风险因素

匹配CWE种类	最大事件发生率	平均事件发生率	平均加权漏洞可利用性	平均加权影响值	最大测试覆盖率	平均测试覆盖率	总发生漏洞应用数	总CVE数
4	19.23%	6.51%	6.87	4.99	53.67%	39.97%	53615	242

风险概述

安全日志和监控故障来自于Top 10的社区调查（排名第3位），比2017年OWASP Top 10社区调查时的第10位略有上升。日志记录和监控是一项具有挑战性的测试，通常涉及访谈或询问渗透测试期间是否检测到攻击。这个类别的CVE/CVSS数据不多，但检测和应对违规行为是至关重要的。同时，它对问责制、可见性、事件告警和取证仍有很大的影响。这个类别除了CWE-778 Insufficient Logging（日志记录不足）外，还包括CWE-117 Improper Output Neutralization for Logs（日志输出不当），CWE-223 Omission of Security-relevant Information（安全事件信息漏报），以及CWE-532 Insertion of Sensitive Information into Log File（在日志文件中包含敏感信息）。

风险说明

2021年版OWASP Top 10中，该类别是为了帮助检测、升级和应对活跃的违规行为。如果不进行日志记录和监测，就无法发现违规行为。任何时候都会发生日志记录、检测、监视和主动响应不足的情况:

- 需要审计的事件，例如：登录、失败的登录和高价值交易，但未记录。
- 警告和错误未生成日志或日志记录不充分或日志消息不清晰。
- 应用程序和 API的日志未进行安全可疑活动的监控。
- 日志只存储在本地。
- 适当的警报阈值和响应升级过程不到位或无效。
- 渗透测试和动态应用安全测试（DAST）工具（例如：OWASP ZAP）的扫描没有触发警报。
- 应用无法实时或接近实时地检测、升级或或对主动攻击发出警报。
- 如果让用户或攻击者看到日志和警报事件，您就容易受到信息泄露的攻击（查看“A01:2021-失效的访问控制”）。

A09：2021-安全日志和监控故障
Security Logging and Monitoring Failures

预防措施

开发人员应根据应用的风险，实施以下部分或全部控制：

- 确保所有的登录、访问控制和服务器端输入验证失败都可以被记录在足够的用户上下文中，以识别可疑或恶意的帐户，并保留足够的时间以允许延迟的取证分析。
- 确保日志是日志管理解决方案以方便使用的格式生成的。
- 确保日志数据被正确编码加密，以防止对日志或监控系统的注入或攻击。
- 确保高价值交易有完整性控制的审计跟踪，以防止篡改或删除，例如：只附加数据库表或类似的内容。
- DevSecOps团队应该建立有效的监控和警报，以便发现可疑的活动并迅速做出反应。
- 建立或采用事故应对和恢复计划，例如：美国国家标准技术研究所(NIST)800-61r2或更高版本。
- 有一些商业和开源的应用程序保护框架，如：OWASP ModSecurity核心规则集，以及开源的日志相关软件，如：Elasticsearch、Logstash、Kibana（ELK），具有自定义仪表盘和告警功能。

攻击范例

范例 #1

一家儿童健康计划供应商的网站运营者由于缺乏安全监控和记录而无法发现漏洞。一个外部团体告知儿童健康计划供应商，某恶意攻击者已经获取并修改了超过350万儿童的数千份敏感健康记录。事后审查发现，网站开发人员没有修复重大漏洞。由于没有对系统进行日志记录或监控，因此数据泄露可能自2013年以来一直在进行，时间超过7年。

范例 #2

印度一家大型航空公司发生数据泄露事件，涉及数百万乘客十多年的个人信息，包括护照和信用卡信息。数据泄露发生在一家第三方云托管服务提供商，该提供商在一段时间后通知了航空公司此次泄露事件。

范例 #3

欧洲一家大型航空公司遭遇一起 可报告的GDPR违规事件。据报道，攻击者利用支付应用程序的安全漏洞，获取了超过40万客户的支付记录，从而导致了此次攻击。该航空公司因此被隐私监管机构处以2000万英镑的罚款。

A10：2021-服务端请求伪造
Server-Side Request Forgery

风险因素

匹配CWE种类	最大事件发生率	平均事件发生率	平均加权漏洞可利用性	平均加权影响值	最大测试覆盖率	平均测试覆盖率	总发生漏洞应用数	总CVE数
1	2.72%	2.72%	8.28	6.72	67.72%	67.72%	9503	385

风险概述

这个类别是从Top 10社区调查（排名第1位）中新添加的。数据显示，该类别安全事件发生率相对较低，测试覆盖率高于平均水平，平均漏洞利用脚本数和影响潜力等级高于平均水平。因为新类别可能是单一或小型集群的CWE，为了引起关注和认识，希望这个类别能受到关注并且在未来的版本可以推出一个更大的类别。

风险说明

一旦Web应用在获取远程资源时没有验证用户提供的URL，就会出现SSRF缺陷。它允许攻击者强制应用程序发送一个精心构建的请求到一个意外目的地，即使是在有防火墙、VPN或其他类型的网络访问控制列表（ACL）保护的情况下。

随着现代Web应用为终端用户提供便利的功能，获取URL成为一种常见的场景。因此，SSRF安全攻击事件也在不断增加。此外，由于云服务和架构的复杂性，SSRF的严重性也越来越高。

A10：2021-服务端请求伪造
Server-Side Request Forgery

预防措施

开发者可以通过实现以下部分或全部防御手段，纵深防御来阻止 SSRF:

网络层防御建议：

- 在隔离的网络中设置多个远程资源访问功能的网段，以减少SSRF的影响。
- 执行“默认拒绝”防火墙策略或网络访问控制规则，以阻止除必要的内部网通信外的所有通信。

提示:

- ✓ 建立基于应用的防火墙规则的所有权和生命周期。
- ✓ 在防火墙上记录所有接受和阻止的网络流(参见“A09:2021-安全日志和监控故障”)。

应用层防御建议：

- 检查和验证所有客户端提供的输入数据。
- 使用白名单允许列表允许列表执行URL统一资源标志符、端口和目标。
- 不要给客户端发送原始的回复。
- 禁用 HTTP 重定向。
- 注意URL的一致性，以避免DNS重新绑定和“检查时间，使用时间”(TOCTOU)竞争条件等攻击。
- 不要通过使用黑名单拒绝列表或正则表达式来缓解SSRF。攻击者拥有有效载荷列表、工具和绕过拒绝列表的技能。

需额外考虑的措施:

- 不要在前端系统上部署其他与安全相关的服务（如：OpenID）。控制这些系统上的本地流量（如：localhost）。
- 对于专用和可管理的前端用户，可以在独立系统上使用网络加密（如：vpn）来满足非常高的安全保护需求。

攻击范例

范例 #1内部服务器端口扫描

如果网络架构未进行网络隔离，攻击者可以访问并绘制出内部网络地图，并根据连接结果或SSRF有效载荷连接的运行时间和拒绝时间判断内部服务器端口是打开还是关闭。

范例 #2敏感数据泄露

攻击者可以访问本地文件或内部服务，以获得敏感信息，例如：file:///etc/passwd

http://localhost:28017/。

范例 #3访问云服务的元数据存储

大多数云服务提供商都有元数据存储，比如：通过http://169.254.169.254/访问。攻击者可以通过读取元数据来获取敏感信息。

范例 #4危害内部服务

攻击者可以滥用内部服务进行进一步的攻击，比如：远程代码执行（RCE）或者拒绝服务攻击（DoS）。

OWASP TOP10

04 相关资料文献

参考文献 A01-A05

A01：2021-失效的访问控制 Broken Access Control

OWASP Proactive Controls: Enforce Access Controls
OWASP Application Security Verification Standard: V4 Access Control
OWASP Testing Guide: Authorization Testing
OWASP Cheat Sheet: Access Control
OWASP Cheat Sheet: Authorization
PortSwigger: Exploiting CORS misconfiguration
OAuth: Revoking Access

A02：2021-加密机制失效 Cryptographic Failures

OWASP Proactive Controls: Protect Data Everywhere
OWASP Application Security Verification Standard (V7, 9, 10)
OWASP Cheat Sheet: Transport Layer Protection
OWASP Cheat Sheet: User Privacy Protection
OWASP Cheat Sheet: Password and Cryptographic Storage
OWASP Cheat Sheet: HSTS
OWASP Testing Guide: Testing for weak cryptography

A03：2021-注入 Injection

OWASP Proactive Controls: Secure Database Access
OWASP ASVS: V5 Input Validation and Encoding
OWASP Testing Guide: SQL Injection, Command Injection, and ORM Injection
OWASP Cheat Sheet: Injection Prevention
OWASP Cheat Sheet: SQL Injection Prevention
OWASP Cheat Sheet: Injection Prevention in Java
OWASP Cheat Sheet: Query Parameterization
OWASP Automated Threats to Web Applications – OAT-014
PortSwigger: Server-side template injection

A04：2021-不安全设计 Insecure Design

OWASP Cheat Sheet: Secure Design Principles
OWASP SAMM: Design:Security Architecture
OWASP SAMM: Design:Threat Assessment
NIST – Guidelines on Minimum Standards for Developer Verification of Software
The Threat Modeling Manifesto
Awesome Threat Modeling

A05：2021-安全配置错误 Security Misconfiguration

OWASP Testing Guide: Configuration Management
OWASP Testing Guide: Testing for Error Codes
Application Security Verification Standard V14 Configuration
NIST Guide to General Server Hardening
CIS Security Configuration Guides/Benchmarks
Amazon S3 Bucket Discovery and Enumeration

参考文献 A06-A10

A06：2021-自带缺陷和过时的组件 Vulnerable and Outdated Components

OWASP Application Security Verification Standard: V1 Architecture, design and threat modelling
OWASP Dependency Check (for Java and .NET libraries)
OWASP Testing Guide - Map Application Architecture (OTG-INFO-010)
OWASP Virtual Patching Best Practices
The Unfortunate Reality of Insecure Libraries
MITRE Common Vulnerabilities and Exposures (CVE) search
National Vulnerability Database (NVD)
Retire.js for detecting known vulnerable JavaScript libraries
Node Libraries Security Advisories
Ruby Libraries Security Advisory Database and Tools
https://safecode.org/publication/SAFECode_Software_Integrity_Controls0610.pdf

A07：2021-身份识别和身份验证错误 Identification and Authentication Failures

OWASP Proactive Controls: Implement Digital Identity
OWASP Application Security Verification Standard: V2 authentication
OWASP Application Security Verification Standard: V3 Session Management
OWASP Testing Guide: Identity, Authentication
OWASP Cheat Sheet: Authentication
OWASP Cheat Sheet: Credential Stuffing
OWASP Cheat Sheet: Forgot Password
OWASP Cheat Sheet: Session Management
OWASP Automated Threats Handbook
NIST 800-63b: 5.1.1 Memorized Secrets

A08：2021-软件和数据完整性故障 Software and Data Integrity Failures

[OWASP Cheat Sheet: Software Supply Chain Security]（即将发布）
[OWASP Cheat Sheet: Secure build and deployment]（即将发布）
OWASP Cheat Sheet: Infrastructure as Code
OWASP Cheat Sheet: Deserialization
SAFECode Software Integrity Controls
A 'Worst Nightmare' Cyberattack: The Untold Story Of The SolarWinds Hack
CodeCov Bash Uploader Compromise
Securing DevOps by Julien Vehent

A09：2021-安全日志和监控故障 Security Logging and Monitoring Failures

OWASP Proactive Controls: Implement Logging and Monitoring
OWASP Application Security Verification Standard: V8 Logging and Monitoring
OWASP Testing Guide: Testing for Detailed Error Code
OWASP Cheat Sheet: Application Logging Vocabulary
OWASP Cheat Sheet: Logging
Data Integrity: Recovering from Ransomware and Other Destructive Events
Data Integrity: Identifying and Protecting Assets Against Ransomware and Other Destructive Events
Data Integrity: Detecting and Responding to Ransomware and Other Destructive Events

A10：2021-服务端请求伪造 Server-Side Request Forgery

OWASP - Server-Side Request Forgery Prevention Cheat Sheet
PortSwigger - Server-side request forgery (SSRF)
Acunetix - What is Server-Side Request Forgery (SSRF)?
SSRF bible
A New Era of SSRF - Exploiting URL Parser in Trending Programming Languages!

对应的CWE A01

OWASP Open Web Application Security Project

A01：2021-失效的访问控制 Broken Access Control

1	CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
2	CWE-23 Relative Path Traversal
3	CWE-35 Path Traversal: '.../...//'
4	CWE-59 Improper Link Resolution Before File Access ('Link Following')
5	CWE-200 Exposure of Sensitive Information to an Unauthorized Actor
6	CWE-201 Exposure of Sensitive Information Through Sent Data
7	CWE-219 Storage of File with Sensitive Data Under Web Root
8	CWE-264 Permissions, Privileges, and Access Controls (should no longer be used)
9	CWE-275 Permission Issues
10	CWE-276 Incorrect Default Permissions
11	CWE-284 Improper Access Control
12	CWE-285 Improper Authorization
13	CWE-352 Cross-Site Request Forgery (CSRF)
14	CWE-359 Exposure of Private Personal Information to an Unauthorized Actor
15	CWE-377 Insecure Temporary File
16	CWE-402 Transmission of Private Resources into a New Sphere ('Resource Leak')
17	CWE-425 Direct Request ('Forced Browsing')
18	CWE-441 Unintended Proxy or Intermediary ('Confused Deputy')
19	CWE-497 Exposure of Sensitive System Information to an Unauthorized Control Sphere
20	CWE-538 Insertion of Sensitive Information into Externally-Accessible File or Directory
21	CWE-540 Inclusion of Sensitive Information in Source Code
22	CWE-548 Exposure of Information Through Directory Listing
23	CWE-552 Files or Directories Accessible to External Parties
24	CWE-566 Authorization Bypass Through User-Controlled SQL Primary Key
25	CWE-601 URL Redirection to Untrusted Site ('Open Redirect')
26	CWE-639 Authorization Bypass Through User-Controlled Key
27	CWE-651 Exposure of WSDL File Containing Sensitive Information
28	CWE-668 Exposure of Resource to Wrong Sphere
29	CWE-706 Use of Incorrectly-Resolved Name or Reference
30	CWE-862 Missing Authorization
31	CWE-863 Incorrect Authorization
32	CWE-913 Improper Control of Dynamically-Managed Code Resources
33	CWE-922 Insecure Storage of Sensitive Information
34	CWE-1275 Sensitive Cookie with Improper SameSite Attribute

对应的CWE A02

OWASP Open Web Application Security Project

A02：2021-加密机制失效 Cryptographic Failures

1	CWE-261 Weak Encoding for Password
2	CWE-296 Improper Following of a Certificate's Chain of Trust
3	CWE-310 Cryptographic Issues
4	CWE-319 Cleartext Transmission of Sensitive Information
5	CWE-321 Use of Hard-coded Cryptographic Key
6	CWE-322 Key Exchange without Entity Authentication
7	CWE-323 Reusing a Nonce, Key Pair in Encryption
8	CWE-324 Use of a Key Past its Expiration Date
9	CWE-325 Missing Required Cryptographic Step
10	CWE-326 Inadequate Encryption Strength
11	CWE-327 Use of a Broken or Risky Cryptographic Algorithm
12	CWE-328 Reversible One-Way Hash
13	CWE-329 Not Using a Random IV with CBC Mode
14	CWE-330 Use of Insufficiently Random Values
15	CWE-331 Insufficient Entropy
16	CWE-335 Incorrect Usage of Seeds in Pseudo-Random Number Generator(PRNG)
17	CWE-336 Same Seed in Pseudo-Random Number Generator (PRNG)
18	CWE-337 Predictable Seed in Pseudo-Random Number Generator (PRNG)
19	CWE-338 Use of Cryptographically Weak Pseudo-Random Number Generator(PRNG)
20	CWE-340 Generation of Predictable Numbers or Identifiers
21	CWE-347 Improper Verification of Cryptographic Signature
22	CWE-523 Unprotected Transport of Credentials
23	CWE-720 OWASP Top Ten 2007 Category A9 - Insecure Communications
24	CWE-757 Selection of Less-Secure Algorithm During Negotiation('Algorithm Downgrade')
25	CWE-759 Use of a One-Way Hash without a Salt
26	CWE-760 Use of a One-Way Hash with a Predictable Salt
27	CWE-780 Use of RSA Algorithm without OAEP
28	CWE-818 Insufficient Transport Layer Protection
29	CWE-916 Use of Password Hash With Insufficient Computational Effort

对应的CWE A03

OWASP Open Web Application Security Project

A03：2021-注入 Injection

1	CWE-20 Improper Input Validation
2	CWE-74 Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')
3	CWE-75 Failure to Sanitize Special Elements into a Different Plane (Special Element Injection)
4	CWE-77 Improper Neutralization of Special Elements used in a Command ('Command Injection')
5	CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
6	CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
7	CWE-80 Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
8	CWE-83 Improper Neutralization of Script in Attributes in a Web Page
9	CWE-87 Improper Neutralization of Alternate XSS Syntax
10	CWE-88 Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')
11	CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
12	CWE-90 Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')
13	CWE-91 XML Injection (aka Blind XPath Injection)
14	CWE-93 Improper Neutralization of CRLF Sequences ('CRLF Injection')
15	CWE-94 Improper Control of Generation of Code ('Code Injection')
16	CWE-95 Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')
17	CWE-96 Improper Neutralization of Directives in Statically Saved Code ('Static Code Injection')
18	CWE-97 Improper Neutralization of Server-Side Includes (SSI) Within a Web Page
19	CWE-98 Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')
20	CWE-99 Improper Control of Resource Identifiers ('Resource Injection')
21	CWE-100 Deprecated: Was catch-all for input validation issues
22	CWE-113 Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')
23	CWE-116 Improper Encoding or Escaping of Output
24	CWE-138 Improper Neutralization of Special Elements
25	CWE-184 Incomplete List of Disallowed Inputs
26	CWE-470 Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')
27	CWE-471 Modification of Assumed-Immutable Data (MAID)
28	CWE-564 SQL Injection: Hibernate
29	CWE-610 Externally Controlled Reference to a Resource in Another Sphere
30	CWE-643 Improper Neutralization of Data within XPath Expressions ('XPath Injection')
31	CWE-644 Improper Neutralization of HTTP Headers for Scripting Syntax
32	CWE-652 Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')
33	CWE-917 Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')

对应的CWE A04

A04：2021-不安全设计 Insecure Design

1	CWE-73 External Control of File Name or Path
2	CWE-183 Permissive List of Allowed Inputs
3	CWE-209 Generation of Error Message Containing Sensitive Information
4	CWE-213 Exposure of Sensitive Information Due to Incompatible Policies
5	CWE-235 Improper Handling of Extra Parameters
6	CWE-256 Unprotected Storage of Credentials
7	CWE-257 Storing Passwords in a Recoverable Format
8	CWE-266 Incorrect Privilege Assignment
9	CWE-269 Improper Privilege Management
10	CWE-280 Improper Handling of Insufficient Permissions or Privileges
11	CWE-311 Missing Encryption of Sensitive Data
12	CWE-312 Cleartext Storage of Sensitive Information
13	CWE-313 Cleartext Storage in a File or on Disk
14	CWE-316 Cleartext Storage of Sensitive Information in Memory
15	CWE-419 Unprotected Primary Channel
16	CWE-430 Deployment of Wrong Handler
17	CWE-434 Unrestricted Upload of File with Dangerous Type
18	CWE-444 Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling')
19	CWE-451 User Interface (UI) Misrepresentation of Critical Information
20	CWE-472 External Control of Assumed-Immutable Web Parameter
21	CWE-501 Trust Boundary Violation
22	CWE-522 Insufficiently Protected Credentials
23	CWE-525 Use of Web Browser Cache Containing Sensitive Information
24	CWE-539 Use of Persistent Cookies Containing Sensitive Information
25	CWE-579 J2EE Bad Practices: Non-serializable Object Stored in Session
26	CWE-598 Use of GET Request Method With Sensitive Query Strings
27	CWE-602 Client-Side Enforcement of Server-Side Security
28	CWE-642 External Control of Critical State Data
29	CWE-646 Reliance on File Name or Extension of Externally-Supplied File
30	CWE-650 Trusting HTTP Permission Methods on the Server Side
31	CWE-653 Insufficient Compartmentalization
32	CWE-656 Reliance on Security Through Obscurity
33	CWE-657 Violation of Secure Design Principles
34	CWE-799 Improper Control of Interaction Frequency
35	CWE-807 Reliance on Untrusted Inputs in a Security Decision
36	CWE-840 Business Logic Errors
37	CWE-841 Improper Enforcement of Behavioral Workflow
38	CWE-927 Use of Implicit Intent for Sensitive Communication
39	CWE-1021 Improper Restriction of Rendered UI Layers or Frames
40	CWE-1173 Improper Use of Validation Framework

对应的CWE A05-A06

A05：2021-安全配置错误 Security Misconfiguration

1	CWE-2 7PK - Environment
2	CWE-11 ASP.NET Misconfiguration: Creating Debug Binary
3	CWE-13 ASP.NET Misconfiguration: Password in Configuration File
4	CWE-15 External Control of System or Configuration Setting
5	CWE-16 Configuration
6	CWE-260 Password in Configuration File
7	CWE-315 Cleartext Storage of Sensitive Information in a Cookie
8	CWE-520 .NET Misconfiguration: Use of Impersonation
9	CWE-526 Exposure of Sensitive Information Through Environmental Variables
10	CWE-537 Java Runtime Error Message Containing Sensitive Information
11	CWE-541 Inclusion of Sensitive Information in an Include File
12	CWE-547 Use of Hard-coded, Security-relevant Constants
13	CWE-611 Improper Restriction of XML External Entity Reference
14	CWE-614 Sensitive Cookie in HTTPS Session Without 'Secure' Attribute
15	CWE-756 Missing Custom Error Page
16	CWE-776 Improper Restriction of Recursive Entity References in DTDs ('XML Entity Expansion')
17	CWE-942 Overly Permissive Cross-domain Whitelist
18	CWE-1004 Sensitive Cookie Without 'HttpOnly' Flag
19	CWE-1032 OWASP Top Ten 2017 Category A6 - Security Misconfiguration
20	CWE-1174 ASP.NET Misconfiguration: Improper Model Validation

A06：2021-自带缺陷和过时的组件 Vulnerable and Outdated Components

1	CWE-937 OWASP Top 10 2013: Using Components with Known Vulnerabilities
2	CWE-1035 2017 Top 10 A9: Using Components with Known Vulnerabilities
3	CWE-1104 Use of Unmaintained Third Party Components

对应的CWE A07

A07：2021-身份识别和身份验证错误 Identification and Authentication Failures

1	CWE-255 Credentials Management Errors
2	CWE-259 Use of Hard-coded Password
3	CWE-287 Improper Authentication
4	CWE-288 Authentication Bypass Using an Alternate Path or Channel
5	CWE-290 Authentication Bypass by Spoofing
6	CWE-294 Authentication Bypass by Capture-replay
7	CWE-295 Improper Certificate Validation
8	CWE-297 Improper Validation of Certificate with Host Mismatch
9	CWE-300 Channel Accessible by Non-Endpoint
10	CWE-302 Authentication Bypass by Assumed-Immutable Data
11	CWE-304 Missing Critical Step in Authentication
12	CWE-306 Missing Authentication for Critical Function
13	CWE-307 Improper Restriction of Excessive Authentication Attempts
14	CWE-346 Origin Validation Error
15	CWE-384 Session Fixation
16	CWE-521 Weak Password Requirements
17	CWE-613 Insufficient Session Expiration
18	CWE-620 Unverified Password Change
19	CWE-640 Weak Password Recovery Mechanism for Forgotten Password
20	CWE-798 Use of Hard-coded Credentials
21	CWE-940 Improper Verification of Source of a Communication Channel
22	CWE-1216 Lockout Mechanism Errors

对应的CWE A08 - A10

A08：2021-软件和数据完整性故障 Software and Data Integrity Failures

1	CWE-345 Insufficient Verification of Data Authenticity
2	CWE-353 Missing Support for Integrity Check
3	CWE-426 Untrusted Search Path
4	CWE-494 Download of Code Without Integrity Check
5	CWE-502 Deserialization of Untrusted Data
6	CWE-565 Reliance on Cookies without Validation and Integrity Checking
7	CWE-784 Reliance on Cookies without Validation and Integrity Checking in a Security Decision
8	CWE-829 Inclusion of Functionality from Untrusted Control Sphere
9	CWE-830 Inclusion of Web Functionality from an Untrusted Source
10	CWE-915 Improperly Controlled Modification of Dynamically-Determined Object Attributes

A09：2021-安全日志和监控故障 Security Logging and Monitoring Failures

1	CWE-117 Improper Output Neutralization for Logs
2	CWE-223 Omission of Security-relevant Information
3	CWE-532 Insertion of Sensitive Information into Log File
4	CWE-778 Insufficient Logging

A10：2021-服务端请求伪造 Server-Side Request Forgery

1	CWE-918 Server-Side Request Forgery (SSRF)

数据要素汇总表A01- A10

OWASP Open Web Application Security Project

No.	匹配CWE种类	最大事件发生率	平均事件发生率	平均加权漏洞可利用性	平均加权影响值	最大测试覆盖率	平均测试覆盖率	总发生漏洞应用数	总CVE数
A01	34	55.97%	3.81%	6.92	5.93	94.55%	47.72%	318487	19013
A02	29	46.44%	4.49%	7.29	6.81	79.33%	34.85%	233788	3075
A03	33	19.09%	3.37%	7.25	7.15	94.04%	47.90%	274228	32078
A04	40	24.19%	3.00%	6.46	6.78	77.25%	42.51%	262407	2691
A05	20	19.84%	4.51%	8.12	6.56	89.58%	44.84%	208387	789
A06	3	27.96%	8.77%	5	5	51.78%	22.47%	30457	0
A07	22	14.84%	2.55%	7.4	6.5	79.51%	45.72%	132195	3897
A08	10	16.67%	2.05%	6.94	7.94	75.04%	45.35%	47972	1152
A09	4	19.23%	6.51%	6.87	4.99	53.67%	39.97%	53615	242
A10	1	2.72%	2.72%	8.28	6.72	67.72%	67.72%	9503	385

OWASP TOP10

05 关于OWASP

关于OWASP

Open Web Application Security Project (OWASP) 是一个开放社区，致力于使组织能够开发、购买和维护可信任的应用程序和 API。

在 OWASP，您会发现免费且开放的：

- 应用程序安全工具和标准
- 前沿研究
- 标准安全控制和库
- 关于应用程序安全测试、安全代码开发和安全代码审查的完整书籍
- 演示文稿和视频
- 许多常见主题的备忘单
- 分会会议
- 活动、培训和会议
- Google讨论组

了解更多信息：https：//www.owasp.org。

所有 OWASP 工具、文档、视频、演示文稿和章节都是免费的，对任何有兴趣提高应用程序安全性的人开放。

我们主张将应用程序安全视为人员、流程和技术问题，因为最有效的应用程序安全方法需要在这些领域进行改进。

OWASP 是一种新型的组织。我们不受商业压力的影响，使我们能够提供有关应用程序安全性的公正、实用且具有成本效益的信息。

OWASP 不隶属于任何技术公司，但我们支持商业安全技术的知情使用。OWASP 以协作、透明和开放的方式制作多种类型的材料。

OWASP 基金会是确保项目长期成功的非营利实体。几乎所有与 OWASP 相关的人都是志愿者，包括 OWASP 董事会、分会负责人、项目负责人和项目成员。我们通过赠款和基础设施支持创新的安全研究。

欢迎加入我们！

OWASP TOP10

谢谢