



# OWASP

The Open Web Application Security Project

## OWASP Top 10 - 2017 rc1

The Ten Most Critical Web Application Security Risks

### Release Candidate

Comments requested per instructions within

# release



Creative Commons (CC) Attribution Share-Alike  
Free version at <https://www.owasp.org>

# 重要提示

## 欢迎反馈

OWASP计划在2017年6月30日公共评议期结束后，于2017年7月或者8月公布OWASP Top 10 - 2017年度的最终公开发布版。

该版本的OWASP Top 10标志着该项目第十四年提高应用安全风险重要性的意识。该版本遵循2013年更新，2013版的Top 10主要变化是添加了2013-A9使用含有已知漏洞的组件。我们很高兴看到，自2013版Top 10以来，随着自由和商业工具的整体生态系统出现，帮助解决这个问题，因为开源组件的使用几乎在每种编程语言中都在继续迅速扩大。数据还表明，使用已知的易受攻击的组件仍然很普遍，但并不像以前那么普遍。我们认为，针对这个问题的关注意识在2013版Top 10出现后已经导致了这两个变化。

我们也注意到，自从CSRF被引入2007版的top 10项目以来，它已经从广泛的脆弱性下降到不常见的脆弱性。许多框架包括自动的CSRF防御，这大大促进了普遍性的下降，以及开发人员对于防范这种攻击的意识提高。

关于这个OWASP top 10 – 2017候选发布版的建设性意见可以通过电子邮件发送到[OWASP-TopTen@lists.owasp.org](mailto:OWASP-TopTen@lists.owasp.org)。私人评论可以发送到[dave.wichers@owasp.org](mailto:dave.wichers@owasp.org)。欢迎匿名反馈。所有非私人反馈将在最终公开发布的同时进行编目和发布。我们建议您对top 10列出的项目进行更改的评论应包括top 10个项目的完整列表，以及更改的理由，所有反馈都应显示具体的相关页面和部分。

在OWASP top 10 - 2017年最终出版之后，OWASP社区的协作工作将持续更新支持文档，包括OWASP Wiki, OWASP Develop's Guide, OWASP Test Guide, OWASP Code Review Guide和OWASP Prevention Cheat Sheets，以及将top 10翻译成许多不同的语言。

您的反馈对于OWASP Top 10项目和所有其他OWASP项目的持续成功至关重要。感谢大家竭尽全力提高全世界的软件的安全性。

Jeff Williams, OWASP Top 10项目创始者和共同作者  
Dave Wichers, OWASP Top 10共同作者和项目负责人



# 关于 OWASP

## 前言

不安全的软件已经在破坏着我们的金融、医疗、国防、能源和其他重要网络架构。随着我们的数字化架构变得越来越复杂并相互关联，实现应用程序安全的难度也呈指数级增加。现代软件开发过程的快速发展使风险更加难以快速准确地发现。我们再也不能忽视像OWASP Top 10中所列出的相对简单的安全问题

Top 10项目的目标是通过找出企业组织所面临的最严重的风险来提高人们对应用程序安全的关注度。Top 10项目被众多标准、书籍、工具和相关组织引用，包括 MITRE、PCI DSS、DISA、FTC等等。OWASP Top 10最初于2003年发布，并于2004年和2007年相继做了少许的修改更新。2010版做了修改以对风险进行排序，而不仅仅对于流行程度。这种模式在2013版和最新的2017版得到了延续。

我们鼓励各位通过使用此Top 10帮助您企业组织了解应用程序安全。开发人员可以从其他企业组织的错误中学习经验。而执行人员需要开始思考如何管理软件应用程序在企业内部产生的风险。

从长远来看，我们鼓励您创建一个与您的文化和技术都兼容的应用安全计划。这些计划可以是任意形式和大小的，您还应该试图避免做过程模型中规定的每件事。相反，利用您组织的现有优势并衡量什么对您有用。

我们希望OWASP Top 10能有助于您的应用程序安全。如果有任何疑问、评论以及想法，请不要犹豫，立即通过公开的[owasp-topten@lists.owasp.org](mailto:owasp-topten@lists.owasp.org)或者私人的[dave.wichers@owasp.org](mailto:dave.wichers@owasp.org)，与我们联系。

## 关于 OWASP

开源web应用安全项目（OWASP）是一个开放的社区，致力于帮助各企业组织开发、购买和维护可信任的应用程序。在OWASP，您可以找到以下免费和开源的信息：

- 应用安全工具和标准
- 关于应用安全测试、安全代码开发和安全代码审查方面的全面书籍
- 标准的安全控制和安全库
- 全球各地分会
- 尖端研究
- 专业的全球会议
- 邮件列表

更多信息，请访问：<http://www.owasp.org>

所有的OWASP工具、文档、论坛和全球各地分会都是免费的，并对所有致力于改进应用程序安全的人士开放。我们主张将应用程序安全问题看作是人、过程和技术的问题，因为提供应用程序安全最有效的方法是在这些方面提升。

OWASP是一个新型组织。没有商业压力使得我们能够提供无偏见、实用、低成本的应用安全方面的信息。尽管OWASP支持合理使用商业的安全技术，但是OWASP不隶属于任何技术公司。和许多开源软件项目一样，OWASP以一种协作、开放的方式制作了许多不同种类的材料。

OWASP基金会是确保项目长期成功的非营利性组织。几乎每一个与OWASP相关的人都是一名志愿者，这包括了OWASP董事会、全球委员会、全球各地分会会长、项目领导和项目成员。我们用捐款和基础设备来支持创新的安全研究。

我们期待您的加入

## 版权和许可



2003—2013 OWASP基金会©版权所有

本文档的发布基于Creative Commons Attribution ShareAlike 3.0 license。任何重复使用或发行，都必须向他人澄清该文档的许可条款。

# 简介

## 欢迎

欢迎阅读2017年版的OWASP Top 10! 这个主要的更新首次增加了两个新的漏洞类别: (1) 攻击检测与防范不足 (2) 未受保护的API。我们通过将两个访问控制类别 (2013-A4和2013-A7) 合并回到失效的访问控制 (这是2014年版top 10的分类名) 中, 为这两个新类别腾出空间, 并将2013-A10 “未经验证的重定向和转发去掉”。

2017年版的OWASP Top 10文档基于来自专业的应用安全公司的11个大型数据库, 其中包括: 8家咨询公司, 3家产品提供商。数据涵盖了来自上百家组织上千个应用, 超过50,000个真实环境中的漏洞和APIs。Top 10根据所有这些相关数据挑选和排序, 并与可利用性、可检测性和影响程度的一致评估相结合。

OWASP Top 10的主要目的是教育开发人员, 设计师, 架构师, 经理和组织, 了解最重要的Web应用程序安全漏洞的后果。Top 10提供了防范这些高风险问题领域的基本技术, 并为您提供了从这里走到哪里的指导

## 警告

**不要仅关注OWASP Top 10:** 正如在《OWASP开发指南》和《OWASP Cheat Sheet》中所讨论的, 能影响整个web应用程序安全的漏洞成百上千。这些指南是当今web应用程序开发人员的必读资料。而《OWASP测试指南》和《OWASP代码审查指南》则指导人们如何有效地查找web应用程序中的漏洞。这两本指南在发布OWASP Top 10的前版本时就已经进行了明显更新。

**不断修改:** 此Top 10将不断更新。即使您不改变应用程序的任何一行代码, 您的应用程序可能已经存在从来没有被人发现过的漏洞。要了解更多信息, 请查看Top 10结尾的建议部分, “开发人员、测试人员和企业组织下一步做什么”。

**正面思考:** 当您已经做好准备停止查找漏洞并集中精力建立强大的应用程序安全控制时, OWASP已经制作了《[应用程序安全验证标准 \(ASVS\)](#)》指导企业组织和应用程序审查者如何去进行验证。

**明智使用工具:** 安全漏洞可能很复杂并且藏匿在代码行的深处。查找并消除这些漏洞的最根本有效的方法就是利用专家的经验以及好的工具。

**其他:** 在您的组织中, 重点关注让安全成为组织文化的一部分。更多信息, 请参见《[开放软件保证成熟度模型 \(SAMM\)](#)》和《[Rugged Handbook](#)》

## 鸣谢

感谢[Aspect Security](#)自2003年OWASP Top 10项目成立以来, 对该项目的创始、领导及更新, 同时我们也感谢主要作者: Jeff Williams和Dave Wichers。



我们也要感谢以下组织贡献了它们的漏洞数据用于支持该项目2017版的更新, 包括这些提供了大量数据集的组织。

- [Aspect Security](#), [AsTech Consulting](#)
- [Aspect Security](#)
- [Branding Brand](#)
- [EdgeScan](#)
- [Minded Security](#)
- [Softtek](#)
- [Veracode](#)
- [AsTech Consulting](#)
- [Contrast Security](#)
- [iBLISS](#)
- [Paladion Networks](#)
- [Vantage Point](#)

第一次我们将向top 10项目提供的所有数据公开, 并且公开了完整的贡献者名单。

我们还要感谢为Top 10本版本做出显著内容贡献和花时间审阅的专家们:

- Neil Smithline – 提供了Top 10的 Wiki版, 并提供了宝贵反馈意见。

最后, 我们感谢所有的翻译人员将Top 10翻译成不同的语言, 帮助让OWASP Top 10对全世界的人们都可以更容易获得。



## 从2013版到2017版有什么改变？

应用程序安全的威胁情况不断改变。这种演变的关键因素是迅速采用新技术（包括云计算，容器和APIs），软件开发过程（如敏捷开发和DevOps）的加速和自动化，第三方库和框架的爆炸式增长以及攻击者的进步。这些因素往往使应用程序和APIs更难分析，并可以显著改变威胁形态。为跟上发展，我们周期性的更新OWASP Top 10。在本次2017年版本中，我们做了以下改变：

- 1) 我们合并了2013-A4 “不安全的直接对象引用”和2013-A7 “功能级访问控制功能缺失”到2017-A4 “无效的访问控制”。
  - 2007年，我们将失效的访问控制分为两类，以便更多地关注访问控制问题（数据和功能）的每一方面。我们不再觉得这是必要的，所以我们将它们合并在一起。
- 2) 我们增加了2017-A7：攻击检测与防范不足
  - + 多年来，我们考虑增加对自动攻击的防御力不足。基于数据调用，我们看到大多数应用程序和API缺乏检测与防范和响应手动或者自动化攻击的基本功能。应用程序和APIs所有者还需要能够快速部署补丁以保护和防止攻击。
- 3) 我们增加了2017-A10: 未受保护的API
  - + 现代应用程序和API通常涉及丰富的客户端应用程序，例如浏览器中的JavaScript和移动端应用程序，连接到某种API（SOAP / XML，REST / JSON，RPC，GWT等）。这些API通常是不受保护的，并且包含许多漏洞。我们将其包括在这里，以帮助组织专注于这一主要的新兴风险。
- 4) 我们去掉了2013-A10:未验证的重定向和转发
  - 2010年，我们增加了这一类别，以提高对这个问题的认识。然而，数据显示，这个问题并不像预期那么普遍。所以在进入top 10的最后两个版本之后，这一次并没有削减。

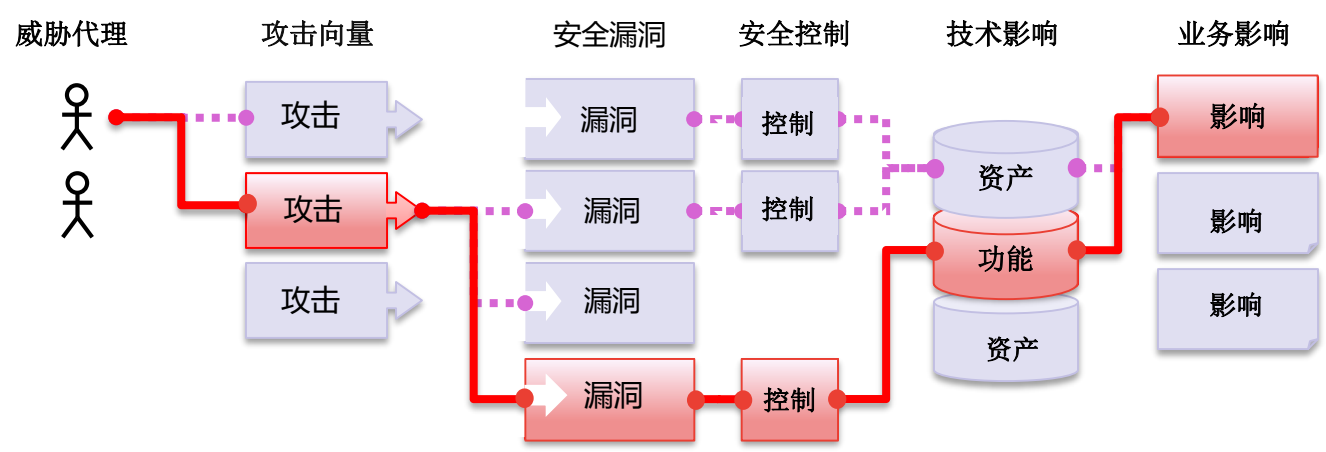
注意：T10围绕主要风险区域进行整理，而不是密封，不重叠或严格的分类。其中一些是围绕着攻击者整理的，一些是脆弱性，一些是防御，一些是资产。组织应考虑制定措施，以消除这些问题。

OWASP Top 10 – 2013 (旧版)	OWASP Top 10 – 2017 (新版)
A1 – 注入	A1 – 注入
A2 – 失效的身份认证和会话管理	A2 – 失效的身份认证和会话管理
A3 – 跨站脚本 (XSS)	A3 – 跨站脚本 (XSS)
A4 – 不安全的直接对象引用	A4 – 失效的访问控制 (最初归类在2003/2004)
A5 – 安全配置错误	
A6 – 敏感信息泄露	A6 – 敏感信息泄露
A7 – 功能级访问控制缺失	A7 – 攻击检测与防范不足 (NEW)
A8 – 跨站请求伪造 (CSRF)	A8 – 跨站请求伪造 (CSRF)
A9 – 使用含有已知漏洞的组件	A9 – 使用含有已知漏洞的组件
A10 – 未验证的重定向和转发	A10 – 未受保护的APIs (NEW)

# 风险 应用程序安全风险

## 什么是应用程序安全风险？

攻击者可以通过应用程序中许多不同的路径方法去危害您的业务或者企业组织。每种路径方法都代表了一种风险，这些风险可能会，也有可能不会严重到值得您去关注



有时，这些路径方法很容易被发现并利用，但有的则非常困难。同样，所造成危害的范围也从无损坏到有可能完全损害您的整个业务。为了确定您的企业的风险，可以结合其产生的技术影响和对企业的业务影响，去评估威胁代理、攻击向量和安全漏洞的可能性。总之，这些因素决定了全部的风险。

## 我的风险是什么？

[OWASP Top 10](#)的重点在于为广大企业组织确定一组最严重的风险。对于 其中的每一项风险，我们将使用基于[OWASP风险等级排序方法](#)的简单评级方案，提供关于可能性和技术影响方面的普遍信息。

威胁代理	攻击向量	漏洞普遍性	漏洞可检测性	技术影响	业务影响
应用描述	易	广泛	易	严重	应用/业务描述
	平均	常见	平均	中等	
	难	少见	难	小	

只有您了解您自己的系统环境和企业的具体情况。对于任何已知的应用程序，可能某种威胁代理无法实施相应的攻击，或者技术影响并没有什么差别。因此，您必须亲自评估每一种风险，特别是需要针对您企业内部的威胁代理、安全控制、业务影响等方面。我们将“威胁代理”作为“应用描述”，“业务影响”作为“应用/业务描述”，以说明这些依赖于您企业中应用的详细信息。

Top 10中风险的名称，有的来自于攻击的类型，有的来自于漏洞，而有的来自于所造成的影响。我们选择了最能准确反应出风险名称，并在可能的情况下，同时使用最为常用的专业名词来得到最高的关注度。

## 参考资料

### OWASP资料

- [OWASP Risk Rating Methodology](#)
- [Article on Threat/Risk Modeling](#)

### 其他资料

- [FAIR Information Risk Framework](#)
- [Microsoft Threat Modeling Tool](#)

**A1 – 注入**

注入攻击漏洞，例如SQL，OS 以及 LDAP注入。这些攻击发生在当不可信的数据作为命令或者查询语句的一部分，被发送给解释器的时候。攻击者发送的恶意数据可以欺骗解释器，以执行计划外的命令或者在未被恰当授权时访问数据。

**A2 – 失效的身份认证和会话管理**

与身份认证和会话管理相关的应用程序功能往往得不到正确的实现，这就导致了攻击者破坏密码、密匙、会话令牌或攻击其他的漏洞去冒充其他用户的身份（暂时的或者永久的）

**A3 – 跨站脚本 (XSS)**

每当应用程序在新网页中包含不受信任的数据而无需正确的验证或转义时，或者使用可以创建JavaScript的浏览器API并使用用户提供的更新现有网页就会发生XSS缺陷。XSS允许攻击者在受害者的浏览器上执行脚本，从而劫持用户会话、危害网站、或者将用户转向至恶意网站。

**A4 – 失效的访问控制**

仅允许通过身份验证的用户的限制没有得到适当的强制执行。攻击者可以利用这些缺陷来访问未经授权的功能和/或数据，例如访问其他用户的帐户，查看敏感文件，修改其他用户的数据，更改访问权限等。

**A5 – 安全配置错误**

好的安全需要对应用程序、框架、应用程序服务器、web服务器、数据库服务器和平台定义和执行安全配置。由于许多设置的默认值并不是安全的，因此，必须定义、实施和维护这些设置。这包含了对所有的软件保持及时地更新，包括所有应用程序的库文件。

**A6 – 敏感信息泄露**

许多Web应用程序没有正确保护敏感数据，如信用卡，税务ID和身份验证凭据。攻击者可能会窃取或篡改这些弱保护的数据以进行信用卡诈骗、身份窃取，或其他犯罪。敏感数据值需额外的保护，比如在存放或在传输过程中的加密，以及在与浏览器交换时进行特殊的预防措施。

**A7 – 攻击检测与防范不足**

大多数应用程序和API缺乏针对手动和自动攻击的检测，预防和响应的基本功能。攻击保护远远超出了基本输入验证，并且涉及自动检测，记录，响应甚至阻止攻击。应用程序所有者还需要有快速部署补丁以防止攻击的能力。

**A8 – 跨站请求伪造 (CSRF)**



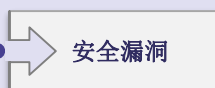

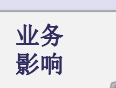
一个跨站请求伪造攻击迫使登录用户的浏览器将伪造的HTTP请求，包括该用户的会话cookie 和其他认证信息，发送到一个存在漏洞的web应用程序。这就允许了攻击者迫使用户浏览器向存在漏洞的应用程序发送请求，而这些请求会被应用程序认为是用户的合法请求。

**A9 – 使用含有已知漏洞的组件**

组件，比如：库文件、框架和其它软件模块，几乎总是以全部的权限运行。如果一个带有漏洞的组件被利用，这种攻击可以造成更为严重的数据丢失或服务器接管。应用程序使用带有已知漏洞的组件会破坏应用程序防御系统，并使一系列可能的攻击和影响成为可能。

**A10 – 未受保护的 APIs**

现代应用程序和API通常涉及丰富的客户端应用程序，例如浏览器中的JavaScript和移动端应用程序，连接到某种API（SOAP / XML，REST / JSON，RPC，GWT等）。这些API通常是不受保护的，并且包含许多漏洞

 威胁代理	 攻击向量			 安全漏洞	 技术影响	 业务影响
应用描述	可利用性 易	普遍性 常见	可检测性 平均	影响 严重	应用/业务描述	
考虑任何能够向系统发送不信任数据的人，包括外部用户，内部用户和管理员。	攻击者利用有针对性的解释器语法发送简单的、基于文本的攻击。几乎任何数据源都能成为注入载体，包括内部来源。	<u>注入漏洞</u> 发生在应用程序将不可信的数据发送到解释器时。注入漏洞十分普遍，尤其是在遗留代码中。通常能在SQL查询语句、LDAP查询语句、Xpath查询语句、OS命令、XML解析器、SMTP头、程序参数等中找到。注入漏洞很容易通过审查代码发现，但是却不容易在测试中发现。扫描器和模糊测试工具可以帮助攻击者找到这些漏洞。		注入能导致数据丢失或数据破坏、缺乏可审计性或是拒绝服务。注入漏洞有时甚至能导致完全主机接管。	考虑受影响的数据和运行解释器的平台的商业价值。所有的数据都有可能被偷窃，篡改和删除。您的声誉是否会被影响？	

## 我是否存在注入漏洞？

检测应用程序是否存在注入漏洞的最好的办法就是确认所有解释器的使用都明确地将不可信数据从命令语句或查询语句中区分出来。在许多情况下，建议避免解释器或禁用它（例如XXE）。对于SQL调用，这就意味着在所有准备语句（prepared statements）和存储过程（stored procedures）中使用绑定变量（bind variables），并避免使用动态查询语句。

检查应用程序是否安全使用解释器的最快最有效的方法是代码审查。代码分析工具能帮助安全分析者找到使用解释器的代码并追踪应用的数据流。渗透测试者通过创建攻击的方法来确认这些漏洞。

可以执行应用程序的自动动态扫描器能够提供一些信息，帮助确认一些可利用的注入漏洞是否存在。然而，扫描器并非总能达到解释器，所以不容易检测到一个攻击是否成功。不恰当的错误处理使得注入漏洞更容易被发现。

## 攻击案例

**案例 #1:** 应用程序在下面存在漏洞的SQL语句的构造中使用不可信数据：

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

**案例 #2:** 同样的，框架应用的盲目信任，仍然可能导致查询语句的漏洞。（例如：Hibernate查询语言（HQL））：

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

在这两个案例中，攻击者在浏览器中将“id”参数的值修改成 ' or '1'='1。如：

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

这样查询语句的意义就变成了从accounts表中返回所有的记录。更危险的攻击可能导致数据被篡改甚至是存储过程被调用。

## 我如何防止注入漏洞？

防止注入漏洞需要将不可信数据从命令及查询中区分开。

1. 最佳选择是使用安全的API，完全避免使用解释器或提供参数化界面的API。但要注意有些参数化的API，比如存储过程（stored procedures），如果使用不当，仍然可以引入注入漏洞。
2. 如果没法使用一个参数化的API，那么你应该使用解释器具体的escape语法来避免特殊字符。OWASP的ESAPI就有一些escape例程。
3. 使用正面的或“白名单”的具有恰当的规范化的输入验证方法同样会有助于防止注入攻击。但由于很多应用在输入中需要特殊字符，这一方法不是完整的防护方法。OWASP的ESAPI中包含一个白名单输入验证例程的扩展库。

## 参考资料






### OWASP

- [OWASP SQL Injection Prevention Cheat Sheet](#)
- [OWASP Query Parameterization Cheat Sheet](#)
- [OWASP Command Injection Article](#)
- [OWASP XXE Prevention Cheat Sheet](#)
- [OWASP Testing Guide: Chapter on SQL Injection Testing](#)

### 其他资料

- [CWE Entry 77 on Command Injection](#)
- [CWE Entry 89 on SQL Injection](#)
- [CWE Entry 564 on Hibernate Injection](#)
- [CWE Entry 611 on Improper Restriction of XXE](#)
- [CWE Entry 917 on Expression Language Injection](#)



 威胁代理	 共击向量		 安全漏洞	 技术影响	 业务影响
应用描述	可利用性 平均	普遍性 广泛	可检测性 平均	影响 严重	应用/业务描述
任何匿名的外部攻击者和拥有账号的用户都可能试图盗取其他用户账号。同样也会有内部人员为了掩饰他们的行为而这么做	攻击者使用认证或会话管理功能中的泄露或漏洞（比如暴露的帐户、密码、或会话ID）来假冒用户	开发者通常会建立自定义的认证和会话管理方案。但要正确实现这些方案却很难，结果这些自定义的方案往往在如下方面存在漏洞：退出、密码管理、超时、记住我、秘密问题、帐户更新等等。因为每一个实现都不同，要找出这些漏洞有时会很困难。		这些漏洞可能导致部分甚至全部帐户遭受攻击。一旦成功，攻击者能执行受害用户的任何操作。因此特权帐户是常见的攻击对象。	需要考虑受影响的数据及应用程序功能的商业价值。  还应该考虑漏洞公开后对业务的不利影响。

## 我存在会话劫持漏洞么？

如何能够保护用户凭证和会话ID等会话管理资产呢？以下情况可能产生漏洞：

1. 用户身份验证凭证没有使用哈希或加密保护。详见A6
2. 认证凭证可猜测，或者能够通过薄弱的帐户管理功能（例如账户创建、密码修改、密码恢复、弱会话ID）重写。
3. 会话ID暴露在URL里（例如，URL重写）。
4. 会话ID容易受到会话固定（[session fixation](#)）的攻击。
5. 会话ID没有超时限制，或者用户会话或身份验证令牌特别是单点登录令牌在用户注销时没有失效。
6. 成功注册后，会话ID没有轮转。
7. 密码、会话ID和其他认证凭据使用未加密连接传输。详见A6。

更多详情请见ASVS要求部分V2和V3。

## 我如何防止？

对企业最主要的建议是让开发人员使用如下资源：

1. 一套单一的强大的认证和会话管理控制系统。这套控制系统应：
  - a) 满足OWASP的[应用程序安全验证标准](#)（ASVS）中V2（认证）和V3（会话管理）中制定的所有认证和会话管理的要求。
  - b) 具有简单的开发界面[ESAPI认证器和用户API](#)是可以仿照、使用或扩展的好范例。
2. 企业同样也要做出巨大努力来避免跨站漏洞，因为这一漏洞可以用来盗窃用户会话ID。详见A3。

## 攻击案例

**案例 #1:** 机票预订应用程序支持URL重写，把会话ID放在URL里：

<http://example.com/sale/saleitems.jsessionid=2P0OC2JSNDLPKHCJUN2JV?dest=Hawaii>

该网站一个经过认证的用户希望让他朋友知道这个机票打折信息。他将上面链接通过邮件发给他朋友们，并不知道自己已经泄漏了自己的会话ID。当他的朋友们使用上面的链接时，他们将会使用他的会话和信用卡。

**案例#2:** 应用程序超时设置不当。用户使用公共计算机访问网站。离开时，该用户没有点击退出，而是直接关闭浏览器。攻击者在一个小时后能使用相同浏览器通过身份认证。

**案例#3:** 内部或外部攻击者进入系统的密码数据库。存储在数据库中的用户密码没有被加密，所有用户的密码都被攻击者获得。

## 参考资料

### OWASP



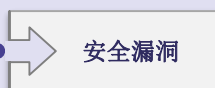

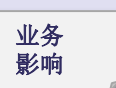
完整的参考资料，见 [ASVS requirements areas for Authentication \(V2\) and Session Management \(V3\)](#).

- [OWASP Authentication Cheat Sheet](#)
- [OWASP Forgot Password Cheat Sheet](#)
- [OWASP Password Storage Cheat Sheet](#)
- [OWASP Session Management Cheat Sheet](#)
- [OWASP Testing Guide: Chapter on Authentication](#)

### 其他

- [CWE Entry 287 on Improper Authentication](#)
- [CWE Entry 384 on Session Fixation](#)



 威胁代理	 攻击向量		 安全漏洞	 技术影响	 业务影响
应用描述	可利用性 易	普遍性 广泛	可检测性 易	影响 中等	应用/业务描述
考虑系统的授权用户的类型。用户是否受限于某些功能和数据？未经身份验证的用户是否允许任意访问任何功能或数据？	经过了验证的攻击者只需将参数值更改为未授权的其他资源。是否可以访问此功能或数据？	对于数据，应用程序和API在生成网页时经常使用对象的实际名称或键。对于函数，URL和函数名通常很容易猜出。应用程序和API并不总是验证用户是否被授权给目标资源，这就会导致访问控制缺陷。测试人员可以轻松操作参数来检测这些缺陷。代码分析可以快速显示授权是否正确。		这种缺陷可能会损害所有可访问的功能或数据。除非引用是不可预知的，或者访问控制被强制执行，数据和功能可能会被盗或被滥用。	考虑暴露的数据和功能的商业价值。  还应该考虑漏洞公开后对业务的不利影响。

## 我存在么？

查找应用程序是否容易受到访问控制漏洞的最佳方法是验证所有数据和函数引用是否具有适当的防御。要确定你是否容易受到攻击，请考虑：

- 对于数据引用，应用程序是否通过使用映射表或访问控制检查确保用户获得授权，以确保用户对该数据进行授权
- 对于非公共功能请求，应用程序是否确保用户进行了身份验证，并具有使用该功能所需的角色或权限？

应用程序的代码审查可以验证这些控件是否正确实施，并且在任何地方都需要进行审计。手动测试对于识别访问控制缺陷也是有效的。自动化工具通常不会找到这样的缺陷，因为他们无法识别需要什么保护或什么是安全的或不安全的。

## 我如何防止？

防止访问控制缺陷。需要选择一个适当的方法 来保护每一个用户可访问的对象（如对象号码、文件名）。

- 检查访问。**任何来自不可信源的直接对象引用都必须通过访问控制检测，确保该用户对请求的对象有访问权限。
- 使用基于用户或者会话的间接对象引用。**这样能防止攻击者直接攻击未授权资源。例如，一个下拉列表包含6个授权给当前用户的资源，它可以使用数字1-6来指示哪个是用户选择的值，而不是使用资源的数据库关键字来表示。在服务器端，应用程序需要将每个用户的间接引用映射到实际的数据库关键字。OWASP的ESAPI包含了两种序列和随机访问引用映射，开发人员可以用来消除直接对象引用。
- 自动验证。**利用自动化来验证正确的授权部署。这要成为习惯。

## 攻击案例

**案例 #1:**应用程序在访问帐户信息的SQL调用中使用未验证数据：

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

攻击者能轻易在浏览器将“acct”参数修改成他所想要的任何账户号码。如果应用程序没有进行恰当的验证，攻击者就能访问任何用户的账户，而不仅仅是该目标用户的账户。

<http://example.com/app/accountInfo?acct=notmyacct>

**案例 #2:**攻击者只是简单的强制浏览目标URL。还需要管理员权限才能访问的管理页面。

<http://example.com/app/getapplnfo>  
[http://example.com/app/admin\\_getapplnfo](http://example.com/app/admin_getapplnfo)

如果未经身份验证的用户可以访问任何一个页面，这是一个缺陷。如果非管理员可以访问管理页面，这也是一个缺陷。

## 参考资料



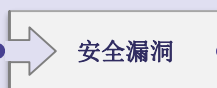

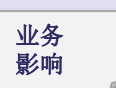
### OWASP

- [OWASP Top 10-2007 on Insecure Direct Object References](#)
- [OWASP Top 10-2007 on Function Level Access Control](#)
- [ESAPI Access Reference Map API](#)
- [ESAPI Access Control API](#) (See `isAuthorizedForData()`, `isAuthorizedForFile()`, `isAuthorizedForFunction()`)

For additional access control requirements, see the [ASVS requirements area for Access Control \(V4\)](#).

### 其他资料

- [CWE Entry 285 on Improper Access Control \(Authorization\)](#)
- [CWE Entry 639 on Insecure Direct Object References](#)
- [CWE Entry 22 on Path Traversal](#) (一个直接对象引用攻击的例子)

 威胁代理	 攻击向量			 安全漏洞	 技术影响	 业务影响
应用描述	可利用性 易	普遍性 常见	可检测性 易	影响中等	应用/业务描述	
考虑外部的匿名攻击者和拥有自己帐户的内部用户都可能会试图破坏系统的。另外考虑想要掩饰他们的攻击行为的内部攻击者。	攻击者访问默认帐户、未使用的网页、未安装补丁的漏洞、未被保护的文件和目录等，以获得对系统未授权的访问或了解。	安全配置错误可以发生在一个应用程序堆栈的任何层面，包括平台、Web服务器、应用服务器、数据库、框架和自定义代码。开发人员和系统管理员需共同努力，以确保整个堆栈的正确配置。自动扫描器可用于检测未安装的补丁、错误的配置、默认帐户的使用、不必要的服务等。		这些漏洞使攻击者能经常访问一些未授权的系统数据或功能。有时，这些漏洞导致系统的完全攻破。		系统可能在你未知的情况下被完全攻破。你的数据可能会随着时间推移被全部盗走或者篡改。恢复的花费可能会很昂贵。

## 我易受到攻击吗？

您的应用程序是否对整个程序堆栈实施了恰当的安全加固措施？这些措施包括：

1. 是否有软件没有被及时更新？这包括操作系统、Web/应用服务器、数据库管理系统、应用程序和其它所有的代码库文件（详见2017-A9）。
2. 是否使用或安装了不必要的功能（例如，端口、服务、网页、帐户、权限）？
3. 默认帐户的密码是否仍然可用或没有更改？
4. 你的错误处理设置是否防止堆栈跟踪和其他含有大量信息的错误消息被泄露？
5. 你的开发框架（比如：Struts、Spring、ASP.NET）和库文件中的安全设置是否理解正确并配置恰当？

缺少一个协定的、可重复的应用程序安全配置的过程，系统将处于高风险中。

## 我如何防止？

主要的建议建立在以下几个方面：

1. 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配置相同（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。
2. 一个能及时了解并部署每个已部署环境的所有最新软件更新和补丁的过程。这需要包括通常被忽略的所有代码的库文件（详见2017-A9）。
3. 一个能在组件之间提供有效的分离和安全性的强大应用程序架构。
4. 一个自动化过程来验证在所有环境中配置和设置是否正确。

## 攻击案例

**案例 #1:** Th应用程序服务器管理员控制台自动安装后没有被删除。而默认帐户也没有被改变。攻击者在你的服务器上发现了标准的管理员页面，通过默认密码登录，从而接管了你的服务器。

**案例 #2:** 目录列表在你的服务器上未被禁用。攻击者发现只需列出目录，她就可以找到你服务器上的任意文件。攻击者找到并下载所有已编译的Java类，她通过反编译获得了所有你的自定义代码。然后，她在你的应用程序中找到一个访问控制的严重漏洞。

**案例 #3:** 应用服务器配置允许堆栈跟踪返回给用户，这样就暴露了潜在的漏洞，例如已知易受攻击的框架版本。

**案例 #4:** 应用服务器自带的示例应用程序没有从您的生产服务器中删除。该示例应用有已知安全漏洞，攻击者可以利用这些漏洞破坏您的服务器。

## 参考资料

### OWASP






- [OWASP Development Guide: Chapter on Configuration](#)
- [OWASP Code Review Guide: Chapter on Error Handling](#)
- [OWASP Testing Guide: Configuration Management](#)
- [OWASP Testing Guide: Testing for Error Codes](#)
- [OWASP Top 10 2004 - Insecure Configuration Management](#)

为了更详尽的了解该领域的需求信息，请参见 [ASVS requirements areas for Security Configuration \(V11 and V19\)](#)。

### 其他资料

- [NIST Guide to General Server Hardening](#)
- [CWE Entry 2 on Environmental Security Flaws](#)
- [CIS Security Configuration Guides/Benchmarks](#)



 威胁代理	 攻击向量		 安全漏洞	 技术影响	 业务影响
应用描述	可利用性 难	普遍性 少见	可检测性 平均	影响 严重	应用/业务描述
考虑谁可以访问您的敏感数据和这些数据的备份。这包括静态数据、传输中的数据甚至是客户浏览器中的数据。	攻击者通常不直接攻击加密系统。他们往往通过诸如窃取密钥、发起中间人攻击或从服务器窃取明文数据等方式对传输中的或者客户浏览器中的数据进行破解。	在这个领域最常见的漏洞是应该加密的数据不进行加密。在使用加密的情况下，常见的问题是不安全的密钥生成和管理和使用弱算法是很普遍的，特别是使用弱的哈希算法来保护密码。浏览器的漏洞也很普遍，且可以很轻易的检测到，但是很难大规模的利用。外部攻击者因访问的局限性很难探测这种漏洞，并且难以利用。		这个领域的错误频繁影响那些本应该加密的数据。这些信息通常包括很多敏感数据，比如医疗记录，认证凭证，个人隐私数据，信用卡信息，等等。	考虑丢失数据和声誉影响造成的商业损失。如果这些数据被泄露，那你要承担的法律責任是什么？另外考虑到对企业造成的声誉影响。

## 我易受攻击么？

首先你需要确认的是哪些数据是敏感数据而需要被加密。例如：密码、信用卡、医疗记录、个人信息应该被加密。对于这些数据，要确保：

1. 当这些数据被长期存储的时候，无论存储在哪里，它们是否都被加密，特别是对这些数据的备份？
2. 无论内部数据还是外部数据，传输时是否是明文传输？在互联网中传输明文数据是非常危险的。
3. 是否还在使用任何旧的或脆弱的加密算法？
4. 加密密钥的生成是否是脆弱的，或者缺少恰当的密钥管理或缺少密钥回转？
5. 当浏览器接收或发送敏感数据时，是否有浏览器安全指令或头文件丢失？

还有更多...关于在这一领域应该避免的更多问题请参见 [ASVS areas Crypto \(V7\)](#), [Data Prot \(V9\)](#), and [SSL/TLS \(V10\)](#).

## 我如何防止？

有关使用不安全的加密算法、SSL使用和数据保护的风险超出了Top 10的范围。尽管如此，对一些需要加密的敏感数据，应该起码做到以下几点：

1. 预测一些威胁（比如内部攻击和外部用户），加密这些数据的存储以确保免受这些威胁。
2. 对于没必要存放的、重要的敏感数据，应当尽快清除。
3. 确保使用了合适的强大的标准算法和强大的密钥，并且密钥管理到位。可参考[FIPS 140 validated cryptographic modules](#).
4. 确保使用密码专用算法存储密码，如：[bcrypt](#), [PBKDF2](#), 或者 [scrypt](#).
5. 禁用自动完成防止敏感数据收集，禁用包含敏感数据的缓存页面。

## 攻击案例

**Scenario #1** 一个应用程序加密存储在数据库的信用卡信息，以防止信用卡信息暴露给最终用户。但是，数据库设置为对信用卡表列的查询进行自动解密，这就使得SQL注入漏洞能够获得所有信用卡信息的明文。备选方案包括不存储信用卡号码，使用标记化或使用公钥加密。

**Scenario #2:** 一个网站上所有需要身份验证的网页都没有使用SSL。攻击者只需监控网络数据流（比如一个开放的无线网络或其社区的有线网络），并窃取一个已验证的受害者的会话cookie。然后，攻击者利用这个cookie执行重放攻击并接管用户的会话从而访问用户的隐私数据。

**Scenario #3:** 密码数据库使用unsalted的哈希算法去存储每个人的密码。一个文件上传漏洞使黑客能够获取密码文件。所有这些unsalted哈希的密码通过彩虹表暴力破解方式破解。

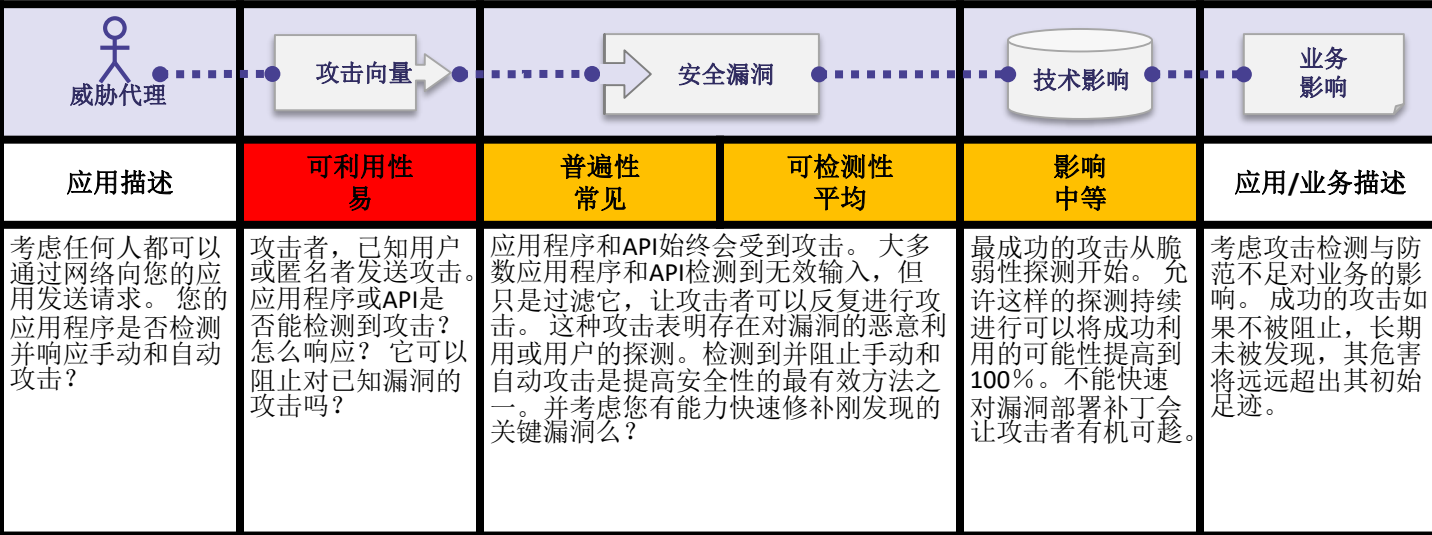
## 参考资料

**OWASP** - 为了更详尽的了解该领域的相关需求和因避免的相关问题，请参见[ASVS req'ts on Cryptography \(V7\)](#), [Data Protection \(V9\)](#) 和 [Communications Security \(V10\)](#)

- [OWASP Cryptographic Storage Cheat Sheet](#)
- [OWASP Password Storage Cheat Sheet](#)
- [OWASP Transport Layer Protection Cheat Sheet](#)
- [OWASP Testing Guide: Chapter on SSL/TLS Testing](#)

### 其他资料

- [CWE Entry 310 on Cryptographic Issues](#)
- [CWE Entry 312 on Cleartext Storage of Sensitive Information](#)
- [CWE Entry 319 on Cleartext Transmission of Sensitive Information](#)
- [CWE Entry 326 on Weak Encryption](#)



## 我容易受攻击吗？

检测，响应和阻止攻击使应用程序的漏洞更难以利用，但几乎没有任何应用程序或API具有此类保护。通用代码和组件中的危险漏洞可在任何时候被发现，但组织通常需要几周甚至几个月才能推出新的防御方案。

如果攻击检测和响应不到位应该是非常明显的。只需尝试手动攻击或针对应用程序运行扫描器。应用程序或API应该识别攻击，阻止任何可行的攻击，并识别收集攻击者的细节和攻击的特征。如果在发现关键漏洞时无法快速推出虚拟和/或实际补丁，则会受到攻击。

一定要了解针对攻击的防范措施所能覆盖的攻击类型，不应该只是XSS和SQL注入。你可以通过如 [WAFs](#), [RASP](#), 和 [OWASP AppSensor](#) 等技术来检测并阻止攻击。

## 我如何防止？

合格的攻击保护有三个主要目标：

- 1. 检测攻击。** 有没有发生合法用户不可能产生（例如，正常用户使用客户端无法生成的输入）的情况？应用程序是否以普通用户永远不会做的方式运行（例如，请求频率太高，非典型输入，异常使用模式，重复请求）？
- 2. 对攻击的响应。** 日志和通知对及时响应至关重要。考虑是否自动阻止请求，并确定阻止的IP地址或IP段。考虑禁用或监控不良行为的用户帐户。
- 3. 快速修复。** 如果您的开发或运维团队无法在一天内推出关键修补程序，请部署一个可以分析HTTP流量，数据流和/或代码执行的虚拟补丁，并防止漏洞被利用。

## 攻击案例

**Scenario #1:** 攻击者使用自动化工具（如OWASP ZAP或SQLMap）来检测漏洞并可能利用它们。

攻击检测应该以识别应用程序的异常请求和巨大流量为目标。自动扫描应易于与正常流量区分开来。

**Scenario #2:** 熟练的攻击者使用仔细挖掘潜在的漏洞，最终确认了一个难以发现的缺陷。

虽然难以检测，但这种攻击仍然包含正常用户永远不会发送的请求，例如UI不允许的输入。跟踪这个攻击者可能需要建立一个时间表现恶意的案例。

**Scenario #3:** 攻击者开始利用应用程序中的一个漏洞，但您当前的攻击保护无法阻止。

您可以快速部署一个真正的或虚拟的补丁来阻止对此漏洞的持续利用？






## 参考资料

### OWASP

- [OWASP Article on Intrusion Detection](#)
- [OWASP AppSensor](#)
- [OWASP Automated Threats Project](#)
- [OWASP Credential Stuffing Cheat Sheet](#)
- [OWASP Virtual Patching Cheat Sheet](#)
- [OWASP Mod Security Core Ruleset](#)

### 其他资料

- [WASC Article on Insufficient Anti-automation](#)
- [CWE Entry 778 - Insufficient Logging](#)
- [CWE Entry 799 - Improper Control of Interaction Frequency](#)

 威胁代理	 攻击向量	 安全漏洞		 技术影响	 业务影响
应用描述	可利用性 平均	普遍性 常见	可检测性 易	影响 中等	应用/业务描述
考虑可能将内容载入你用户的浏览器并迫使他们向你的网站提交请求的任何人。你的用户所访问的任何网站或者HTML源（feed）都可以这样做。	攻击者创建伪造HTTP请求并通过图片标签、跨站脚本或许多其他技术诱使受害用户提交这些请求。如果 <b>该受害用户已经经过身份认证</b> ，那么攻击就能成功。	CSRF 是利用某些web应用程序允许攻击者预测一个特定操作的所有细节这一特点。由于浏览器自动发送会话cookie等认证凭证，攻击者能创建恶意web页面产生伪造请求。这些伪造请求很难与合法请求区分开。跨站请求伪造漏洞可以很容易通过渗透测试或代码分析检测到。		攻击者能欺骗受害用户完成该受害者所允许的任意状态改变的操作，比如：更新帐号细节，完成购物，修改数据等操作	考虑受影响的数据和应用功能的商业价值。试想如果并不知道这些操作是否是用户的真正意愿会产生什么后果。同时考虑带来的声誉影响。

## 我存在CSRF漏洞？

检测应用程序是否存在该漏洞的方法是查看是否每个链接和表单都提供了不可预测的CSRF令牌。没有这样的令牌，攻击者就能够伪造恶意请求。另一种防御的方法是要求用户证明他们要提交请求，比如通过重新认证的方式。

重点关注那些调用能够改变状态功能的链接和表格，因为他们是跨站请求伪造攻击的最重要的目标。由于多步交易并不具备内在的防攻击能力，因此我们需要检测这些交易。还要注意，服务器端请求伪造（SSRF）也可以通过欺骗应用和API来生成任意HTTP请求。

请注意：会话cookie、源IP地址和其他浏览器自动发送的信息不能作为防攻击令牌，因为这些信息已经包含在伪造的请求中。OWASP的[CSRF测试工具](#)有助于生成测试案例，可用于展示跨站请求伪造漏洞的危害。

## 我如何防止CSRF？

首选方案是使用现有的CSRF防御措施。许多框架现在包括内置的CSRF防御，如[Spring](#)、[Play](#)、[Django](#)和[AngularJS](#)。一些Web开发语言，如[.NET](#)也是如此。OWASP的[CSRF Guard](#)可以自动将CSRF防御添加到Java应用程序。OWASP的[CSRF Protector](#)对于PHP或Apache有着相同的过滤作用。

否则，阻止CSRF通常需要在每个HTTP请求中包含不可预测的令牌。并且这种令牌在每个用户会话中都是唯一的。

1. 最好的方法是将独有的令牌包含在一个隐藏字段中。这将使得该令牌通过HTTP请求体发送，避免其包含在URL中从而被暴露出来。
2. 该独有令牌同样可以包含在URL中或作为一个URL参数。但是这种方法的巨大风险在于：URL会暴露给攻击者，这样秘密令牌也会被泄露。
3. 考虑在所有Cookie中[使用](#)“SameSite = strict”标志，这在浏览器中越来越受到[支持](#)。

## 攻击案例

应用程序允许用户提交不包含任何保密字段的状态改变请求，如：

```
http://example.com/app/transferFunds?amount=1500
&destinationAccount=4673243243
```

因此，攻击者构建一个请求，用于将受害用户账户中的现金转移到自己账户。然后攻击者在其控制的多个网站的图片请求或iframe中嵌入这种攻击。

```

```

如果受害用户通过example.com认证后访问任何一个攻击者的网站，伪造的请求将自动包含用户的会话信息，授权执行攻击者的请求。

## 参考资料

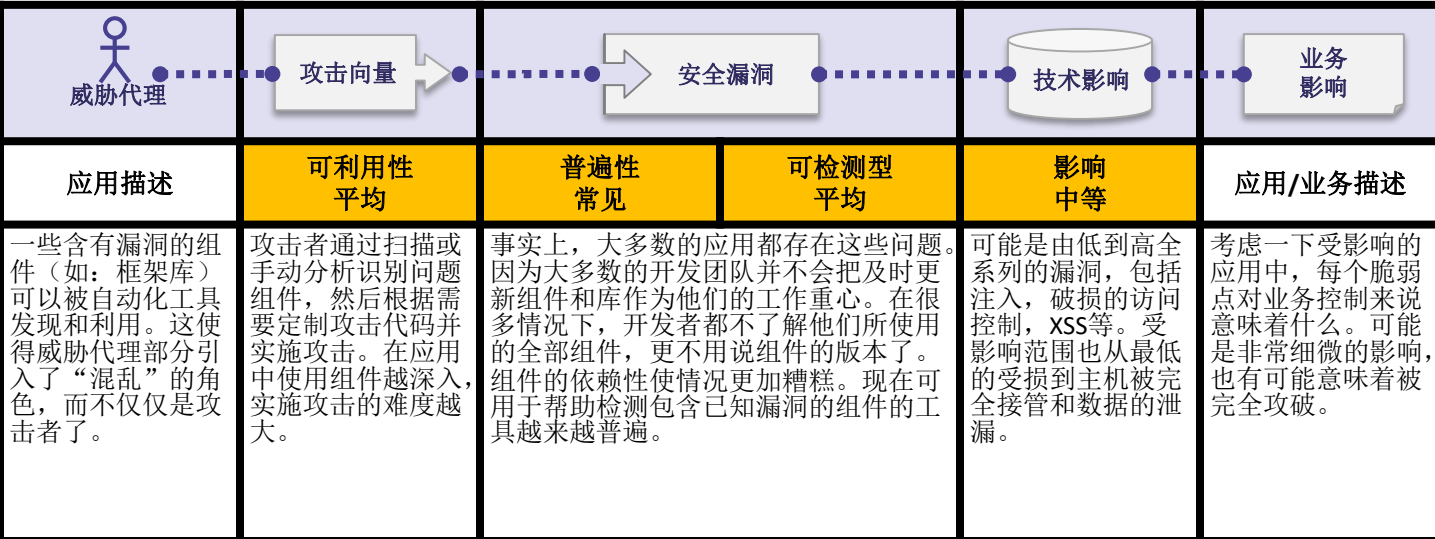
### OWASP

- [OWASP CSRF Article](#)
- [OWASP CSRF Prevention Cheat Sheet](#)
- [OWASP CSRFGuard - Java CSRF Defense Tool](#)
- [OWASP CSRFProtector - PHP and Apache CSRF Defense Tool](#)
- [ESAPI HTTPUtilities Class with AntiCSRF Tokens](#)
- [OWASP Testing Guide: Chapter on CSRF Testing](#)
- [OWASP CSRFTester - CSRF Testing Tool](#)

### 其他资料

- [CWE Entry 352 on CSRF](#)
- [Wikipedia article on CSRF](#)





## 我存在含有已知漏洞组件的漏洞？

理论上，应该是很容易确定您当前是否在使用含有漏洞的组件或者库。不幸的是，商业或开源软件的漏洞报告并不能以标准的、可查找的方式指定受影响组件的确切版本信息。更有甚者，并不是所有的库都使用易于理解的版本编号系统。最糟糕的是，不是所有的漏洞都报告给一个方便查询的漏洞中心，尽管，像CVE或NVD这样的网站正变得更易于搜索。

判断您是否易于受到这类攻击，要求您不但要不停地搜索这些数据库，还要关注大量的邮件列表和可能包含漏洞发布的公告信息。如果您使用的组件之一存在漏洞，您应该仔细评估该漏洞是否给您的业务也带来了缺陷。此评估可以通过检查您的代码使用该组件的部分，以及该缺陷可能导致的您关心的结果来完成。

## 我如何防止？

大多数组件项目并不为其老版本提供漏洞补丁。相反，它们仅仅在下个版本中修正此问题。所以升级到新版本是很重要的。软件项目应该有如下流程：

1. 持续的清点整理客户端和服务端所使用的组件和与组件存在依赖关系的组件的版本等信息
2. 连续监控如NVD等披露的组件中的漏洞是否出现在您的应用中。使用软件结合分析工具自动化进行这个过程。
3. 分析库文件以确保在进行更改之前在运行时实际调用了它，因为绝大多数组件都不会被加载或调用。
4. 决定是升级组件（如果需要，重写应用程序以匹配）还是部署一个分析HTTP流量，数据流或代码执行的虚拟补丁，并防止漏洞被利用。

## 攻击案例

组件几乎总是以应用程序的全部特权运行，因此任何组件的缺陷都可能导致严重的影响。这种缺陷可能是偶然的（例如，编码错误）或有意的（例如组件中的后门）。一些攻击使用含有已知漏洞的组件的例子是：

- [Apache CXF认证绕过](#)—未能提供身份令牌的情况下，攻击者可以以最高权限调用任意的web服务。（Apache CXF 是一个服务框架，不要与Apache应用服务器混淆。）
- [Spring远程代码执行](#)—滥用Spring中语言表达式的实现允许攻击者执行任意代码，有效的接管服务器。

每个使用上述两个任意一个库的应用程序，都是易于受到攻击的。因为两个组件都会被应用用户直接访问。其他的漏洞库，在应用程序中使用的越深入，可能越难被利用。

## 参考资料

### OWASP

- [OWASP Dependency Check \(for Java and .NET libraries\)](#)
- [OWASP Virtual Patching Best Practices](#)

### 其他资料

- [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](#)



# A10

# 未受保护的 APIs

应用描述	可利用性 平均	普遍性 常见	可检测性 难	影响 中等	应用/业务描述
考虑有能力向API发送请求的人。客户端软件很容易逆向，通信容易被拦截，所以简单的混淆无法用于API的防御。	攻击者可以通过逆向工程来检查客户端代码或简单地监控通信。一些API漏洞可以自动发现，其他的只有专家才能发现。	现在丰富的客户端（浏览器，移动端，桌面）越来越多地使用Web应用程序和API使用连接到后端API（XML，JSON，RPC，GWT，自定义）。API（微服务，服务，终端）可能会受到全面的攻击，不幸的是，动态的或者静态的工具在API检测分析上不能很好的工作，而且手工分析也很难分析，所以这些漏洞经常被放过。		全面的负面结果是可能的，包括数据窃取和破坏；未经授权访问整个应用程序；并完成控制主机。	考虑API攻击对业务的影响。API访问关键数据还是功能？许多API上承载着核心业务，所以也考虑到拒绝服务攻击的影响。

## 我容易受到攻击么？

测试您的API漏洞应该类似于测试其他应用程序的漏洞。所有不同类型的注入，认证，访问控制，加密，配置和其他问题都可以在传统应用程序中出现的也存在于API中。

然而，由于API被程序（而不是人类）使用，所以他们经常缺少UI，并且还使用复杂的协议和复杂的数据结构。这些因素可以使安全测试变得困难。使用广泛使用的格式可以帮助，例如Swagger（OpenAPI），REST，JSON和XML。一些框架，如GWT和一些RPC实现使用自定义格式。一些应用程序和API创建自己的协议和数据格式，如WebSockets。API的广泛性和复杂性使得难以进行有效的自动化安全测试，这可能导致虚假的安全感。

最终，知道您的API是否安全意味着需要仔细选择一种攻击策略来测试所有重要的防御。

## 我如何防止？

保护API的关键在于确保您充分了解威胁模型以及防御方式：

1. 确保您已经保护客户端和您的API之间的通信。
2. 确保您的API具有强大的身份验证方案，并且所有凭据、密钥和令牌已被保护。
3. 确保您的请求使用的任何数据格式，解析器都被配置并强化到可以防止此类攻击。
4. 实现访问控制方案，保护API不被不正确地调用，包括未经授权的功能和数据引用。
5. 防止所有形式的注入，即便它们适用于普通应用，但是这些攻击对API同样可行。

确保您的安全分析和测试涵盖所有API，您的工具可以有效地发现和分析它们。

## 攻击案例

**案例 #1:** 想象一下，一个移动端银行应用程序连接到银行的XML API，用于获取帐户信息和执行交易。攻击者逆向应用程序，并发现用户帐号作为认证请求的一部分与用户名和密码一起传递到服务器。攻击者发送合法凭据，但是发送另一个用户的帐号，可以获得对其他用户帐号的完全访问权限。

**Scenario #2:** 想象一下由网络启动公共API，用于自动发送短信。API接受包含“transactionid”字段的JSON消息。API将此“transactionid”值解析为字符串，并将其连接到SQL查询中，而不需要转义或参数化它。您可以看到API与任何其他类型的应用程序一样容易受到SQL注入。

在任何一种情况下，供应商可能不提供使用这些服务的Web UI，从而使安全测试更加困难。

## 参考资料

### OWASP

- [OWASP REST Security Cheat Sheet](#)
- [OWASP Web Service Security Cheat Sheet](#)

### 其他资料

- [Increasing Importance of APIs in Web Development](#)
- [Tracking the Growth of the API Economy](#)
- [The API Centric Future](#)
- [The Growth of the API](#)
- [What Do You Mean My Security Tools Don't Work on APIs?!!](#)
- [State of API Security](#)

## 建立并使用可重复使用的安全流程和标准安全控制

无论您是刚接触web应用程序安全还是已经非常熟悉各种风险，创建一个安全的web应用程序或修复一个已存在的 应用程序的任务都可能很困难。如果您需要管理一个大型的应用程序组合，那任务将是十分艰巨的。

为了帮助企业组织和开发人员以最低成本降低应用程序的安全风险，OWASP制作了许多免费和开源的资源。您可以使用这些资源来解决您企业组织的应用程序安全问题。以下内容是OWASP提供的为帮助企业组织创建安全的web应用程序 的一些资源。在下一页中，我们将展示其他可以帮助企业组织验证web应用程序安全性的OWASP资源。

### 应用程序安全需求

为了创建一个安全的web应用程序，您必须定义安全对该应用程序的意义。OWASP建议您使用[《OWASP应用程序安全验证标准（ASVS）》](#)，作为指导，帮助您设置您的应用程序的安全需求。如果您的应用程序是外包的，您需要考虑使用[《OWASP安全软件合同附件》](#)。

### 应用程序安全架构

与其改造应用程序的安全，不如在应用程序开发的初始阶段进行安全设计，更能节约成本。OWASP推荐[《OWASP开发者指南》](#)和[《OWASP防护最佳实践》](#)，这是很好的起点，用于指导如何在应用程序开发的初始阶段进行安全设计。自2013年十大发行以来，Cheat Sheets已经更新和扩大。

### 标准的安全控制

建立强大并有用的安全控制极度困难。给开发人员提供一套标准的安全控制会极大简化应用程序的安全开发过程。OWASP推荐[OWASP企业安全API（ESAPI）项目](#)作为安全API的模型，用于创建安全的web应用程序。ESAPI提供多种语言的参考实现，包括[Java](#)，许多流行框架都附带了用于授权，验证，CSRF等的标准安全控制。

### 安全的开发周期

为了改进企业遵循的应用程序开发流程，OWASP推荐使用[《OWASP软件保证成熟模型（SAMM）》](#)。该模型能帮助企业组织制定并实施根据企业面临的特定风险而定制的软件安全战略。Open SAMM的重大更新于2017年发布。

### 应用程序安全教育

[OWASP教育项目](#)为培训开发人员的web应用程序安全知识提供了培训材料，并编制了大量OWASP教育演示材料。如果需要实际操作了解漏洞，可以使用[OWASP WebGoat](#)，[WebGoat.NET](#)，或者[OWASP Broken Web Application项目](#)。如果想了解最新资讯，请参加[OWASP AppSec大会](#)，OWASP会议培训，或者[本地的OWASP分部会议](#)。

还有许多其他的OWASP资源可供使用。请访问[OWASP项目页面](#)，其中列出了OWASP项目库存中的所有前沿项目，实验室和孵化器项目。大多数OWASP资源都可以在我们的[wiki](#)上查看到，同时可以订购各种[OWASP纸质文档](#)或[电子书](#)。

## 建立持续的应用安全测试

安全地构建代码很重要。但是，验证您要构建的安全性是否真实存在，正确实施并在其应用的任何地方都实施变得至关重要。应用程序安全测试的目标就是提供这些证据。工作艰巨复杂，敏捷开发和DevOps等现代化快速过程对传统的方法和工具施加了极大的压力。因此，我们强烈建议您考虑如何专注于整个应用程序组合中的主要内容，并以经济高效的方式进行。

现代风险迅速发展，所以每年对应用程序进行一次扫描或渗透测试日子已经过去了。现代软件开发需要在整个软件开发生命周期中进行持续的应用程序安全测试。寻求通过自动化安全测试来增强现有开发流水线，而且开发速度不会降低。无论您选择哪种方法，都可以考虑定期测试，分类，修复，重新测试和重新部署单个应用程序的成本乘以应用程序组合的大小。

### 理解威胁模型

在您开始测试之前，请您花时间了解重要的内容。优先级来自威胁模型，所以如果你没有这个模型，你需要在测试之前创建一个。考虑使用OWASP ASVS和OWASP测试指南作为指导标准，不要依靠工具供应商来决定您的业务的重要性。

### 理解你的安全生命周期

您的应用程序安全测试方法必须与您在软件开发生命周期（SDLC）中使用的人员，流程和工具高度兼容。尝试强制执行额外的步骤和复查可能会导致摩擦，可以寻找自然的机会来收集安全信息并将其反馈到您的过程中。

### 测试策略

选择最简单，最快，最准确的技术来验证每个要求。[OWASP Benchmark Project](#)有助于衡量安全工具检测许多OWASP Top 10风险的能力，可能有助于为您的特定需求选择最佳工具。一定要考虑处理误报所需的人力以及漏报的严重危害。

### 实现全面性和准确性

你不必测试所有的东西。专注于什么是重要的，并随着时间的推移优化扩展您的验证程序。这意味着扩展一系列针对安全防御和风险以及应用程序和API的自动验证程序。目标是让所有应用程序和API的基本安全性得到持续保证。

### 让结果有价值

无论您在测试中有多好，除非您有效沟通，否则不会有什么作用。通过展示您理解应用程序的工作原理与相关人员建立信任。清楚地描述一个漏洞如何被滥用，并通过例子将漏洞情况展示出来。之后对漏洞的发现和利用难度，以及结果的危害程度，做一个现实的评估。最后，确保您提供测试结果在开发团队中已经使用，而不是PDF文件。

## 现在就启动您的应用程序安全计划

应用程序安全已经不再是一个选择了。在日益增长的攻击和监管的压力下，企业组织必须建立一个有效的能力去确保应用程序的安全。由于已经在生产环境中的应用程序和代码行数量惊人，许多企业组织都得努力处理数量巨大的漏洞。OWASP推荐这些企业组织建立一个应用程序安全计划，深入了解并改善它们的应用程序组合的安全性。为了获得应用程序的安全性，需要不同企业组织中的多个部门之间协同工作，这包括了安全和审计、软件开发、和商业与执行管理。它要求提供安全的可见度，让所有不同角色的人都可以看到并理解企业组织的应用程序的安全态势。它还要求将重点集中在能以最低成本有效减少风险的实际活动和成果上，以提高整个企业组织的安全性。一个有效的应用程序安全计划中的一些关键活动包括：

### 开始阶段

- 建立一个[应用程序安全计划](#)并被采纳
- 进行[能力差距分析以比较您的组织和您的同行](#)，从而定义关键有待改善的领域和一个执行计划
- 得到管理层的批准，并建立一个针对企业的整个IT组织的应用程序安全宣传活动。

### 基于风险的 组合方法

- 从固有风险的角度来确定并[对您的应用程序组合进行优先排序](#)。
- 建立一个应用程序的风险特征分析模型来衡量和优先考虑您的应用程序组合。建立保证准则，合理定义需要的覆盖范围和严格水平。
- 建立一个[通用的风险等级模型](#)，该模型应该包含一组一致的可能性和影响因素，来反应您的企业组织的风险承受能力。

### 建立 强大的基础

- 建立一组集中关注的[策略和标准](#)，用于提供所有开发团队所遵循的一个应用程序安全底线。
- 定义一组通用的[可重复使用的安全控制](#)，用于补充这些政策和标准，并提供使用它们的设计和开发指南。
- 建立一个[应用程序安全培训课程](#)，此课程应该要求所有的开发人员参加，并且针对不同的开发责任和话题进行修改。

### 将安全 整合入 现有流程

- 定义并集成[安全实施](#)和[核查](#)活动到现有的开发与操作流程之中。这些活动包括了[威胁建模](#)，安全设计和[审查](#)，安全编码和[代码审查](#)，[渗透测试](#)，修复等等。
- [为开发和项目团队提供主题专家和支持服务](#)，以保证他们的工作顺利进行。

### 提高安全在 管理层的可 见度

- 通过度量进行管理。根据对获取的度量和分析数据决定改进和投资的方向。这些度量包括：遵循安全实践/活动，引入的漏洞，修复的漏洞，应用程序覆盖的范围等等。
- 对实现和核查活动进行数据分析，寻找根本原因和漏洞模式，以推动整个企业的战略和系统改进。



## 这里讲述的是风险，而不是漏洞

虽然2007年和之前更老版本的OWASP Top 10专注于查找最常见的“漏洞”，但是OWASP TOP 10仍然一直围绕着风险而组织。这使得一些试图寻找一个严格的漏洞分类结构的人产生了一些理解上的混乱。2010年的OWASP Top 10项目首次明确了10大风险，十分明确地描述了威胁代理、攻击向量、漏洞、技术风险，和业务风险这些因素如何结合在一起产生风险，这个版本的OWASP TOP 10仍然采用相同的方法论。






Top 10 的风险评级方法是基于《OWASP风险评级方法》。对于Top 10中每一项，我们通过查看每个常见漏洞一般情况下的可能性因素和影响因素，评估了每个漏洞对于典型的Web应用程序造成的典型风险，然后根据漏洞给应用程序带来的风险程度的不同来对Top 10进行分级。随着事件的变化，这些因素将随着每个新的十大版本的更新。

《OWASP风险评级方法》定义了许多用于计算漏洞风险等级的因素。但是，Top 10应该讨论普遍性，而不是在真实的应用程序中讨论具体的漏洞的风险。因此，我们无法像系统拥有者那样精确计算应用程序中的风险高低。我们也不知道您的应用程序和数据有多重要、您的威胁代理是什么、或是您的系统是如何架构和如何操作的。

对于每一个漏洞，我们的方法包含三种可能性因素（普遍性、可检测性和可利用性）和一个影响因素（技术影响）。漏洞的普遍性我们通常无需计算。许多不同的组织一直在提供普遍性的数据给我们（请参考第3页致谢中的内容）。我们取了这些数据的平均数得到了根据普遍性排序的10种最可能存在的漏洞。然后将这些数据和其他两个可能性因素结合（可检测性和可利用性），用于计算每个漏洞的可能性等级。然后用每个漏洞的可能性等级乘以我们估计的每个漏洞的平均技术影响，从而得到了Top 10列表中每一项的总的风险等级。

值得注意的是这个方法既没有考虑威胁代理的可能性，也没有考虑任何与您的特定应用程序相关的技术细节。这些因素都可以极大影响攻击者发现和利用某个漏洞的整个可能性。这个等级同样没有将对您的业务的实际影响考虑进去。您的企业组织需要自己确定企业组织可以承受的应用安全风险有多大。OWASP Top 10的目的并不是替您做这一风险分析。

下面举例说明A3: 跨站脚本的风险计算方法。注意到XSS的风险非常普遍，以致于它被唯一赋予了“非常广泛”的普遍性值。其他所有风险值的范围从广泛到少见（值从1到3）。

 威胁代理	 攻击向量		 安全漏洞		 技术影响	 业务影响
应用描述	可利用性 平均	普遍性 非常广泛	可检测性 易	影响 中等	应用/业务描述	
	2	0	1	2		
		1	*	2		
			2			

## Top 10风险因素总结

下面的表格总结了2013年版Top 10应用程序安全风险因素，以及我们赋予每个风险因素的风险值。这些因素基于 OWASP 团队拥有的统计数据和经验而决定。为了了解某个特定的应用程序或者企业组织的风险，您必须考虑您自己的威胁代理和业务影响。如果没有相应位置上的威胁代理去执行必要的攻击，或者产生的业务影响微不足道，那么就是再臭名昭著的软件漏洞也不会导致一个严重的安全风险。

风险	威胁代理	攻击向量			技术影响	业务影响
		可利用性	普遍性	利用难度		
A1-注入	应用描述	易	常见	平均	严重	应用描述
A2-失效的身份认证和会话管理	应用描述	平均	常见	平均	严重	应用描述
A3-跨站脚本	应用描述	平均	非常流行	平均	中等	应用描述
A4-失效的访问控制	应用描述	易	流行	易	中等	应用描述
A5-安全配置错误	应用描述	易	常见	易	中等	应用描述
A6-敏感信息泄露	应用描述	困难	少见	平均	严重	应用描述
A7-攻击检测与防范不足	应用描述	易	常见	平均	中等	应用描述
A8-跨站请求伪造 (CSRF)	应用描述	平均	少见	易	中等	应用描述
A9-使用含有已知漏洞的组件	应用描述	平均	常见	平均	中等	应用描述
A10-未受保护的 APIs	应用描述	平均	常见	难	中等	应用描述

## 额外需要考虑的风险

虽然Top 10的内容覆盖广泛，但是在您的企业组织中还有其他的风险需要您考虑并且评估。有的风险出现在了OWASP Top 10的以前版本中，而有的则没有，这包括在不停被发现的新的攻击技术。其他您需要考虑的重要应用程序安全风险包括以下方面：

- [点击劫持 \(CAPEC-103\)](#)
- [拒绝服务 \(CWE-400\)](#) (Was 2004 Top 10 – [Entry 2004-A9](#))
- [反序列化 \(CWE-502\)](#) For defenses, see: [OWASP Deserialization Cheat Sheet](#)
- [表达式语言注入 \(CWE-917\)](#)
- [信息泄露 \(CWE-209\)](#) 和 不恰当的的错误处理 ([CWE-388](#)) (Was part of 2007 Top 10 – [Entry 2007-A6](#))
- [链接第三方内容 \(CWE-829\)](#)
- [恶意文件执行 \(CWE-434\)](#) (Was 2007 Top 10 – [Entry 2007-A3](#))
- [质量分配 \(CWE-915\)](#)
- [服务器端请求伪造 \(SSRF\) \(CWE-918\)](#)
- [未验证的转发和重定向 \(CWE-601\)](#) (Was 2013 Top 10 – [Entry 2013-A10](#))
- [用户隐私 \(CWE-359\)](#)

THE BELOW ICONS REPRESENT WHAT OTHER VERSIONS ARE AVAILABLE IN PRINT FOR THIS TITLE BOOK.

**ALPHA:** "Alpha Quality" book content is a working draft. Content is very rough and in development until the next level of publication.

**BETA:** "Beta Quality" book content is the next highest level. Content is still in development until the next publishing.

**RELEASE:** "Release Quality" book content is the highest level of quality in a book's lifecycle, and is a final product.



**ALPHA**  
PUBLISHED



**BETA**  
PUBLISHED



**RELEASE**  
PUBLISHED

YOU ARE FREE:



to share - to copy, distribute and transmit the work



to Remix - to adapt the work

UNDER THE FOLLOWING CONDITIONS:



**Attribution.** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).



**Share Alike.** - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.



**OWASP**

The Open Web Application Security Project

The Open Web Application Security Project (OWASP) is a worldwide free and open community focused on improving the security of application software. Our mission is to make application security "visible," so that people and organizations can make informed decisions about application security risks. Everyone is free to participate in OWASP and all of our materials are available under a free and open software license. The OWASP Foundation is a 501c3 not-for-profit charitable organization that ensures the ongoing availability and support for our work.