



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

欢迎提出您的宝贵意见

OWASP组织计划在2017年6月30日公众反馈意见收集结束后，于2017年7月或8月发布最终版的2017年版《OWASP Top 10》。

本次《OWASP Top 10》的发布，意味着我们已经持续14年专注于对重要应用安全风险的意识提升。本版继承了2013年版的更新（主要变化是增加了“2013-A9 使用含有已知漏洞的组件”）。自2013年版《OWASP Top 10》发布以来，我们很高兴看到为应对A9这种问题已经出现了一个包括诸多开源与商业软件的生态系统；同时，对于开源组件的应用，已经迅速渗入几乎每一种编程语言。数据还表明，使用含已知漏洞的组件是一个普遍现象，虽然已经不像以前那么广泛了。我们认为正是因为2013版《OWASP Top 10》提出了这一问题和配套解决方案，才促成了这两种改进。

我们也注意到自从CSRF被引入到2007年版的Top 10项目以后，它已经从一个广泛影响的漏洞变得不那么常见了。许多防御框架，包括了大大降低这种漏洞影响范围的自动化CSRF防御措施，极大提高了开发人员对于这种攻击的防范意识。

关于对本2017年版《OWASP Top 10》（Release Candidate版）的建设性意见和反馈，可通过电子邮件发送到OWASP-TopTen@lists.owasp.org。私人评论可以发送到dave.wichers@owasp.org。也欢迎匿名反馈。所有非私人反馈将在最终版发布的同时，被分类发布。您对Top 10所列出十类风险的修订反馈意见，我们建议请包括对Top 10所有项目的完整建议列表，以及修订的理由，所有反馈都应标注具体的相关页面和章节。

在2017年版《OWASP Top 10》最终发布之后，OWASP组织将同步更新相关支持文档，包括《OWASP Wiki》、《OWASP Developer's Guide》、《OWASP Testing Guide》、《OWASP Code Review Guide》和《OWASP Prevention Cheat Sheets》，并将Top 10翻译为不同的语言版本。

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。感谢大家为改进、提升全球的软件应用安全水平所做出的杰出贡献。

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、能源和其他重要基础设施。随着我们的软件变得越来越重要、复杂且相互关联，实现应用程序安全的难度也呈指数级增长。而现代软件开发过程的飞速发展，使得快速、准确地识别风险变得愈发的重要。我们再也不能忽视像《OWASP Top 10》中所列出相对简单的安全问题。

Top 10项目的目标，是通过识别出企业组织所面对最严重的风险来提高人们对应用程序安全的意识。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年相继做了少许的更新。2010年版的修改则以针对风险进行排序，而不仅仅对于流行程度。2013年版和本次发布的2017年版也沿用了该方法。

我们鼓励各位通过使用《OWASP Top 10》帮助您企业组织开始了解应用程序安全。开发人员可以从其他企业组织的错误中学到经验。管理人员则需要开始思考如何在企业内部开发软件应用和API时管理风险。

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规模，您还应该试图避免开展过程模型中规定的每件事。相反，您应该利用您企业组织的现有优势，并衡量对您有用的事物。

我们希望《OWASP Top 10》能对您的应用程序安全有帮助。如果有任何疑问、评论和想法，请不要犹豫，立即通过公开的owasp-topten@lists.owasp.org或者私人的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完整书籍
- 标准的安全控制和安全库
- [全球各地分会](#)
- 尖端技术研究
- [专业的全球会议](#)
- [邮件列表](#)
- [OWASP中国分会](#)

更多信息，请访问：<http://www.owasp.org>
<http://www.owasp.org.cn>

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。我们没有商业压力，使得我们能够提供无偏见、实用、低成本的应用安全信息。尽管OWASP支持合理使用商业安全技术，但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、全球各地分会会长、项目领导和项目成员。我们用资金和基础设施来支持创新的安全研究。

我们期待您的加入！

版权和许可

2003—2017 OWASP基金会©版权所有

2006—2017 OWASP中国©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。



欢迎

欢迎阅读2017年版的《OWASP Top 10》！该版本的主要变化是首次添加了两类风险：（1）“攻击检测与防护不足”；（2）“未受有效保护的API”。针对这两类新的风险，我们通过将两个与访问控制有关的类别（2013-A4和2013-A7）合并成“失效的访问控制”（正如在2004年版《OWASP Top 10》中那样命名），并移除了在2010年版《OWASP Top 10》中添加的2013-A10“未验证的重定向和转发”。

2017年版的《OWASP Top 10》文档主要基于专业应用安全公司（包含8家咨询公司和3家产品厂商）提供的11组大型数据。这些数据包含了从数以百计的组织机构和从超过50,000个真实应用程序和API中收集的漏洞。从这些流行数据挑选出排名前十位的类别，并结合对可利用性、可探测性和影响的协商一致估计。Top 10中的条目从所有这些相关数据中被挑选和排序，并与可利用性、可检测性和影响程度的综合评估结果相结合。

OWASP Top 10的首要目的是教导开发人员、设计人员、架构师、经理和企业组织，让他们认识到最严重Web应用程序安全弱点所产生的后果。Top 10提供了防止这些高风险问题发生的基本方法，并为获得这些方法的提供了指引。

警告

不要停滞于OWASP Top 10：正如在[《OWASP开发者指南》](#)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应用程序和API开发人员的必读资料。而[《OWASP测试指南》](#)和[《OWASP代码审查指南》](#)则指导相关人员如何有效地查找Web应用程序和API中的漏洞。

持续完善：本《OWASP 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，并且攻击方法也在不断改进。要了解更多信息，请查看本文结尾的建议部分，“开发人员、测试人员和企业组织下一步做什么”。

积极思考：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正在维护和鼓励企业组织和应用程序审查者将[《应用程序安全验证标准（ASVS）》](#)作为如何去开展验证工作的指导手册。

合理使用工具：安全漏洞可能很复杂并且藏匿在代码行的深处。在许多案例中，查找并消除这些弱点的成本最有效方法，就是为专家配备好的工具。

其他：在您的企业组织中，需重点关注如何将安全成为组织文化的一部分。更多信息，请参见[《OWASP软件验证成熟度模型（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。



我们要感谢很多企业组织，感谢它们为支持2017年版的更新提供了漏洞数据，包括下述贡献了大组数据的企业组织：

- Aspect Security
- Branding Brand
- EdgeScan
- Mindred Security
- Softtek
- Veracode
- AsTech Consulting
- Contrast Security
- iBLISS
- Paladion Networks
- Vantage Point

针对为Top 10贡献的所有数据和做出贡献的完整人员名单，我们史无前例的对[公众发布](#)。

另外，我们要感谢为本次Top 10更新做出显著的、有建设性评论的专家：

- Neil Smithline—感谢他此前为Top 10建立Wiki。

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

OWASP Top 10中文翻译项目组

项目组组长：王颀、包悦忠

翻译人员：顾凌志、王厚奎、王文君、吴楠、夏玉明、杨天识、袁明坤、张镇（排名不分先后，按姓氏拼音排列）

审查人员：Rip、夏天泽

说明：本版本已对2013年中文版《OWASP Top 10》中翻译不准确的内容进行修正。因翻译水平有限，敬请指正。

2013版至2017版改变了哪些内容？

应用程序和API的威胁情况不断变化。这一演变的关键因素是新技术(包括云、容器和API)的快速采用、软件开发过程(如敏捷软件开发和DevOps)的加速和自动化、第三方库和框架的爆炸式增长，以及攻击者在攻击方面取得的进展。这些因素常常导致应用程序和API更加难以分析，而且也会显著改变威胁环境。为与不断变化的威胁保持同步，我们周期性地更新《OWASP Top 10》。在本次2017年版本中，我们做出了以下变更：

1) 我们将2013-A4 “不安全地直接对象引用”和2013-A7 “功能级访问控制缺失”合并到2017-A4 “失效的访问控制”中。

• 在2007年，我们将“失效的访问控制”拆分成这两类，以使数据类风险和功能类风险分别得到有关访问控制问题的更多关注。我们现在觉得不再有此必要，因此，将它们重新合并。

2) 我们增加了2017-A7 “攻击检测与防护不足”。

+ 多年来，我们一直考虑增加对自动化攻击防御不足的问题。根据调查数据，我们可以看到绝大多数应用程序和API在应对攻击时（无论是手动攻击还是自动攻击）都缺乏检测、防护和响应的基本能力。应用程序和API的所有者也需要能够快速部署补丁，以对应用程序和API所受的攻击提供防护。

3) 我们增加了2017-A10 “未受有效保护的API”。

+ 现代应用程序和API常涉及富客户端应用程序，如：浏览器和移动APP中的JavaScript，其与某类API(SOAP/XML、REST/JSON、RPC、GWT等)连接。这些API通常未受保护并且含有大量漏洞。我们将其包括以引起各组织对该重大新兴风险的关注。

4) 我们去掉了2013-A10 “未定向的重验证和转发”。

- 在2010年，我们增加了该类别以唤醒对此问题的意识。然而数据显示，该问题并不像预期中的那样普遍。因此，在前两次的Top 10版本发布中都其列入，这次，我们决定将其删除。

注意：Top 10 是围绕主要风险领域而组织形成的，它们并非严密的、不重叠的或严格的分类。有的风险是围绕攻击者组织形成的，也有围绕漏洞、防御和资产的。企业组织应考虑制定相关举措以消除这些问题。

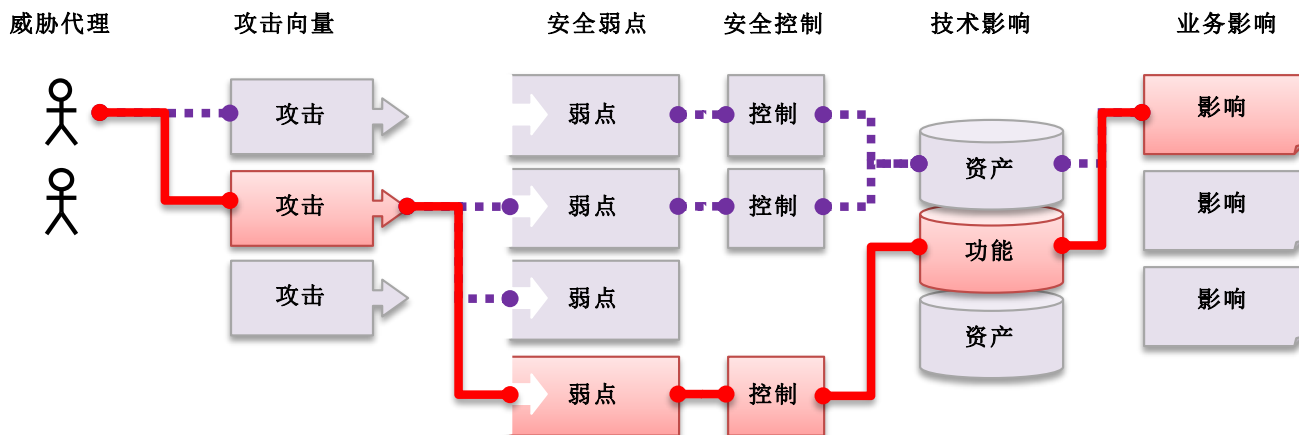
OWASP Top 10 – 2013 （旧版）	OWASP Top 10 – 2017 （新版）
A1 - 注入	A1 - 注入
A2 - 失效的身份认证和会话管理	A2 - 失效的身份认证和会话管理
A3 - 跨站脚本 (XSS)	A3 - 跨站脚本 (XSS)
A4 - 不安全的直接对象引用 - 与A7合并	A4 - 失效的访问控制（最初归类在2003/2004版）
A5 - 安全配置错误	A5 - 安全配置错误
A6 - 敏感信息泄漏	A6 - 敏感信息泄漏
A7 - 功能级访问控制缺失 - 与A4合并	A7 - 攻击检测与防护不足(新增)
A8 - 跨站请求伪造 (CSRF)	A8 - 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 - 未受有效保护的API(新增)

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。

我有什么风险?

[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 \(STRIDE and DREAD\)](#)

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，这就导致了攻击者攻击者破坏密码、密钥、会话令牌或攻击其他的漏洞去冒充其他用户的身份（暂时或永久的）。

A3 – 跨站脚本 (XSS)

当应用程序收到含有不可信的数据，在没有进行适当的验证和转义的情况下，就将它发送给一个网页浏览器，或者使用可以创建JavaScript脚本的浏览器API利用用户提供的数据更新现有网页，这就会产生跨站脚本攻击。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重定向到恶意网站。

A4 – 失效的访问控制

对于通过认证的用户所能够执行的操作，缺乏有效的限制。攻击者就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账户，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。此外，所有的软件应该保持及时更新。

A6 – 敏感信息泄露

许多web应用程序和API没有正确保护敏感数据，如财务、医疗保健和PII。攻击者可能会窃取或篡改此类弱保护的数据，进行信用卡欺骗、身份窃取或其他犯罪行为。敏感数据应该具有额外的保护，例如在存放或在传输过程中的加密，以及与浏览器交换时进行特殊的预防措施。

A7 – 攻击检测与防护不足

大多数应用和API缺乏检测、预防和响应手动或自动化攻击的能力。攻击保护措施不限于基本输入验证，还应具备自动检测、记录和响应，甚至阻止攻击的能力。应用所有者还应能够快速部署安全补丁以防御攻击。

A8 – 跨站请求伪造 (CSRF)

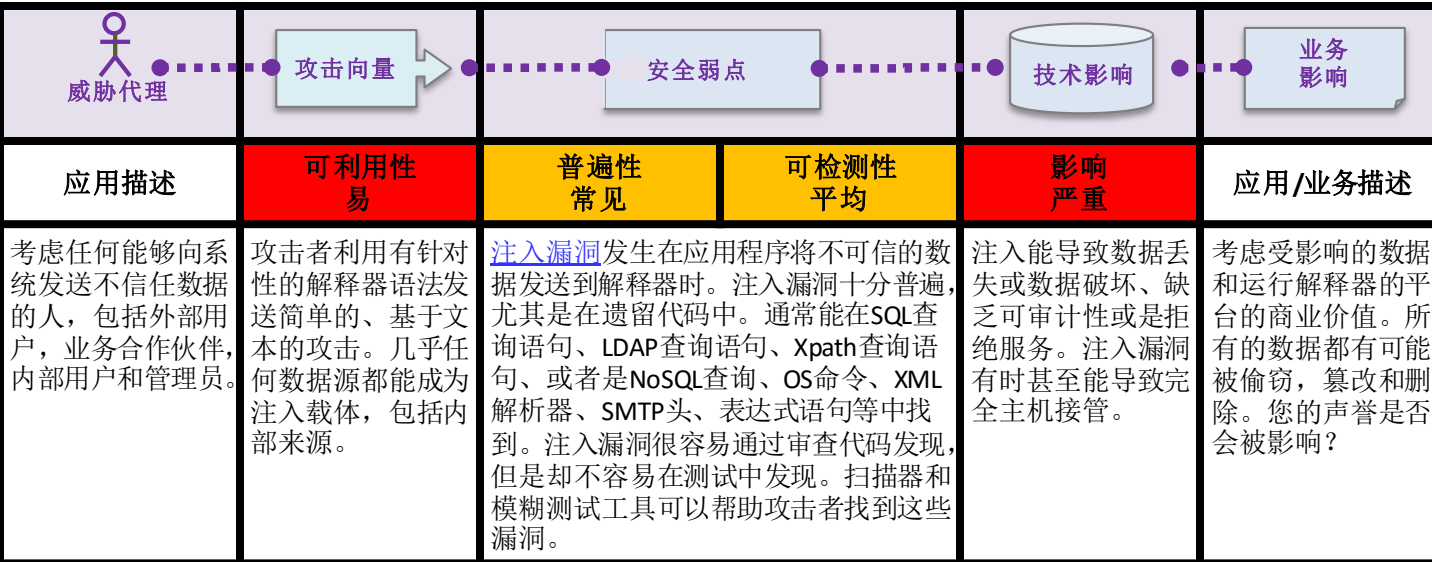
一个跨站请求伪造攻击迫使登录用户的浏览器将伪造的HTTP请求，包括受害者的会话cookie和所有其他自动填充的身份认证信息，发送到一个存在漏洞的web应用程序。这种攻击允许攻击迫使受害者的浏览器生成让存在漏洞的应用程序认为是受害者的合法请求的请求。

A9 – 使用含有已知漏洞的组件

组件，比如：库文件、框架和其他软件模块，具有与应用程序相同的权限。如果一个带有漏洞的组件被利用，这种攻击可以促成严重的数据丢失或服务器接管。应用程序和API使用带有已知漏洞的组件可能会破坏应用程序的防御系统，并使一系列可能的攻击和影响成为可能。

A10 – 未受有效保护的API

现代应用程序通常涉及丰富的客户端应用程序和API，如：浏览器和移动APP中的JavaScript，其与某类API (SOAP/XML、REST/JSON、RPC、GWT等) 连接。这些API通常是不受保护的，并且包含许多漏洞。



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。如果可能的话，在许多情况下，建议避免解释器，或禁用它（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。

攻击案例场景

场景#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") + "'";
```

场景#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's Java Encoder](#)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。




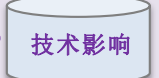

参考资料

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](#)

外部资源

- [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	 攻击向量	 安全弱点		 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 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吗？

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2017-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2017-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。详见本文档A3。

攻击案例场景

场景#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;sessionid=2P0OC2JDPXM00Q$NDLPSKHJCJUN2JV?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。

场景#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。

场景#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哈希和加盐，所有用户的密码都被攻击者获得。

参考资料

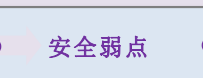
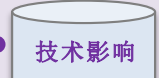
OWASP

完整的要求和预防资料，见[ASVS requirements areas for Authentication \(V2\) 和 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](#)

外部资源

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

 威胁代理	 攻击向量	 安全弱点		 技术影响	 业务影响
应用描述	可利用性 容易	普遍性 广泛	可检测性 容易	影响 中等	应用/业务描述
考虑系统的授权用户类型。用户是否受限于访问某些功能和数据？未经过认证的用户是否允许访问任何功能或者数据？	作为授权的系统用户，攻击者只需要修改指向一个系统资源的直接引用参数值，让其指向另一个无权访问的资源。系统是否会允许这样的访问？	当生成web页面时，数据、应用程序和APIs经常使用对象的实名或关键字。功能、URLs和函数名称经常容易被猜解。而应用程序和APIs并不总是验证用户对目标资源的访问授权，这就导致了访问控制失效。测试者能轻易操作参数值以检测该漏洞。代码分析能很快显示应用程序是否进行了正确的权限验证。		这种漏洞能破坏通过该参数引用的所有功能和数据。除非引用是不可预知的或者强制执行访问控制，否则数据和功能可以被窃取，或滥用。	考虑暴露的数据和功能的商业价值。还应该考虑漏洞被暴露后对商业的不利影响。

我存在漏洞吗？

检测一个应用程序是否存在失效的访问控制漏洞的最好办法，就是验证所有数据和函数引用是否有适当的防御。要达到这一目的，可以考虑：

- 对于**数据**引用，应用程序是否通过使用引用映射或访问控制检查确保用户被授权，以确保用户对该数据授权？
- 对于非公开**功能**请求，应用程序是否确保用户身份验证，并具有使用该函数所需的角色或权限？

对应用程序进行代码审查能验证这些控制措施是否正确实施并且存在于需要的任何地方。手工测试同样是找出失效的访问控制的有效方法。然而，自动化工具通常无法检测到这些漏洞，因为它们无法识别哪些需要保护、哪些是安全或不安全的。

攻击案例场景

场景#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中将“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
```

场景#2：攻击者能轻易强制浏览器访问目标URLs。管理员权限也需要访问管理页面。

```
http://example.com/app/getappInfo
```

```
http://example.com/app/admin_getappInfo
```

如果一个未经认证的的用户能访问这两个页面，说明存在漏洞。

如果非管理员能够访问admin页面，也说明存在漏洞。

我如何防止？

要预防失效的访问控制，需要选择一个适当的方法来保护每一个功能和每种数据类型（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。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。
- 自动化验证。**利用自动化验证正确的授权部署。这是通常做法。

参考资料



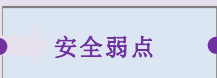
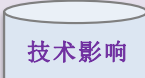
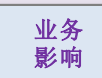
OWASP

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

更多的访问控制需求，请见 [ASVS requirements area for Access Control \(V4\)](#)。

外部资源

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

 威胁代理	 攻击向量	 安全弱点		 技术影响	 业务影响
应用描述	可利用性 容易	普遍性 常见	可检测性 容易	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。

我易受攻击吗？

你的应用程序是否在应用程序栈的任何部分缺少适当的安全加固？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、APIs、和其它所有的组件和库文件（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. 你的错误处理机制是否防止堆栈跟踪和其他含有大量的错误信息被泄露？
5. 对你的应用服务器和应用框架是否进行了安全配置（比如：Struts、Spring、ASP.NET）和库文件、数据库等，没有进行安全配置？

缺少一个体系的、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组件和库文件（详见新的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. 在所有环境中能够正确安全配置和设置的自动化过程。

攻击案例场景

场景#1：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。

场景#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。

场景#3：应用服务器配置允许堆栈跟踪信息返回给用户，这样就暴露了潜在的漏洞。如已知的有漏洞的框架版本。

场景#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

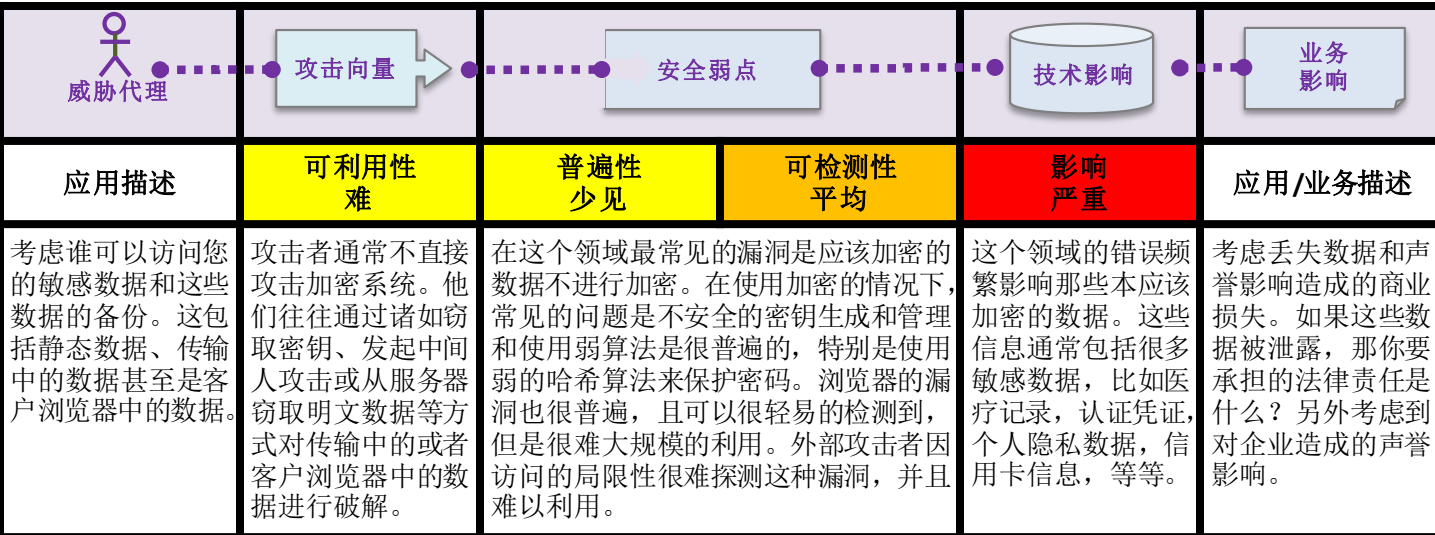
OWASP

- [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 area for Security Configuration \(V11 and V19\)](#)。

外部资源

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



我易受攻击吗？

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. \(V9\)](#) 和 [SSL \(V10\)](#)。

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。

攻击案例场景

场景#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。

场景#2: 一个网站上所有需要身份验证的网页都没有使用TLS。攻击者只需监控网络数据流（比如一个开放的无线网络），并窃取一个已验证的受害者的session 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。

场景#3: 密码数据库使用未加盐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。所有这些未加盐哈希的密码通过彩虹表暴力破解方式破解。

参考资料



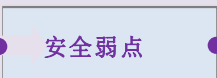
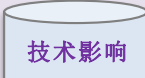
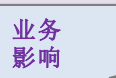
OWASP

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](#)

外部资源

- [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](#)

 威胁代理	 攻击向量	 安全弱点		 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 中等	应用 /业务描述
任何具有网络访问权限的人都可以向你的应用程序发送一个请求。你的应用程序能检测到手动攻击和自动化攻击并做出响应吗？	已知用户或匿名用户发动攻击。应用程序或API能检测到吗？如何响应？能否阻止对已知漏洞发动的攻击吗？	应用程序和API无时无刻都在遭受着攻击。它们检测到非法输入后，只是丢弃它，从而使得攻击者可以反复的实施攻击。其实这往往表明一个恶意用户或者是被盗用的用户正在实施探测或利用漏洞。检测并阻止手动或自动化的攻击是提高安全性最有效的手段之一。对于刚被你发现的高危漏洞，你能多快的打补丁呢？		许多成功的攻击都起始于探测漏洞。允许这种持续的探测会大大的增加成功利用漏洞的可能性。不能快速的打补丁也有利于攻击者。	对于业务应该考虑到应对攻击防护不足的影响。成功的攻击可能不会被阻止，很长时间未被发现并远远超出预期。

我是否存在应对攻击防护不足漏洞？

检测、响应并阻止攻击极大的增加了应用程序被攻破的难度，可惜的是几乎没有应用程序或API这么做。代码或组件中存在的高危漏洞一直被批露，然后应用供应商往往要花数周或几个月的时间来制定出新的防护措施。

试着手动攻击或者运行扫描器，就很容易判断一个应用程序是否具有攻击检测和响应功能。应用程序或API应该识别出攻击，阻止任何可能的攻击，并提供出攻击者的具体信息以及攻击的类型。当一个高危漏洞被检测出，如果你不能快速发布虚拟或实际补丁，那你就很容易遭受攻击。

一定要弄清楚攻击防护覆盖的攻击类型有哪些，仅仅只能防护跨站和SQL注入吗？你可以使用一些技术如WAF, RASP和OWASP AppSensor以及漏洞的虚拟补丁来检测和阻止攻击。

攻击案例场景

场景#1: 攻击者利用自动化工具如OWASP ZAP或SQLMap来检测和利用漏洞。

攻击检测应该识别出应用程序正在遭受着异常请求和大流量攻击，对于自动化扫描也很容易的和正常流量区分开。

场景#2: 熟练的攻击者小心的探测着可能存在的漏洞，最终发现哪些隐藏的漏洞。

然而这类攻击可能包含着正常用户从不会发送的一些请求，如禁止在界面上输入的内容，很难被检测到。跟踪这类攻击需些时间创建一些用以阐明恶意意图的用例。

场景#3: 攻击者已经开始利用你应用系统中的一个漏洞，而你当前的攻击检测未能阻止。

你能多快的部署一个真实或虚拟补丁来阻止对此漏洞的持续攻击？

我如何防止？

充分的攻击防护应该包括以下三点内容：

- 1. 检测攻击。**合法用户不可能做的事情发生了吗（如合法用户不可能创造出的输入）？正常用户是否用永远不会这样使用应用系统（如输入速度过快，不合规则的输入，非正常的使用模式，重复的请求）？
- 2. 响应攻击。**日志和通知对于及时的响应异常重要。考虑对于某个IP或者一个IP网段是否实施自动阻止。考虑对哪些异常的用户账号进行禁用或者监控。
- 3. 快速打补丁。**如果你的开发团队不能在一天内对高危漏洞发布补丁，可以尝试部署一个虚拟补丁来分析HTTP流量，数据流，代码执行并且防止漏洞被利用。


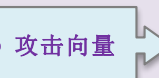
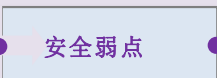
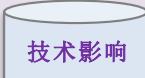
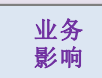
参考资料

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](#)

外部资源

- [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		 安全弱点	 技术影响	 业务影响
应用描述	可利用性 平均	普遍性 不常见	可检测性 易	影响 中等	应用 /业务描述
考虑可能将内容载入用户的浏览器并迫使他们向你的网站提交请求的任何人，包括用户所访问的任何网站或者HTML 源	攻击者创建伪造HTTP请求并通过图片标签、iframe、跨站脚本或许多其它技术诱使受害用户提交这些请求。 <u>如果该受害用户已经经过身份认证，那么攻击就能成功。</u>	<u>CSRF</u> 是因为某些web应用程序允许攻击者预测一个特定操作的所有细节。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，并且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是他们要提交请求，如重新认证。

重点关注那些调用能够改变状态功能的链接和表单，因为他们是跨站请求伪造攻击的最重要的目标。多步交易并不具备内在的防攻击能力，并且攻击者也可以利用SSRF来诱导应用程序和API来产生任意的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？

优先考虑的就是利用已有的CSRF防护方案。许多框架，如[Spring](#)、[Play](#)、[Django](#)以及[AngularJS](#)，都内嵌了CSRF防护，一些web开发语言如[.Net](#)也提供了类似的防护。[OWASP CSRGuard](#)对Java应用提供了CSRF防护，[OWASP CSRFProtector](#)对PHP应用和Apache server也提供CSRF了防护。否则需要在每个HTTP请求中添加一个不可预测的令牌，这种令牌至少应该对每一个用户会话来说是唯一的。

- 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。
- 将令牌放在URL中或作为一个URL参数，但是这种方法的风险在于令牌会暴露给攻击者。
- 考虑对所有cookie使用“SameSite=strict”标签，该标签逐渐被各类浏览器所支持。

攻击案例场景

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中以图片请求或iframe中嵌入这种攻击。

```

```

如果受害用户通过了example.com认证，则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。

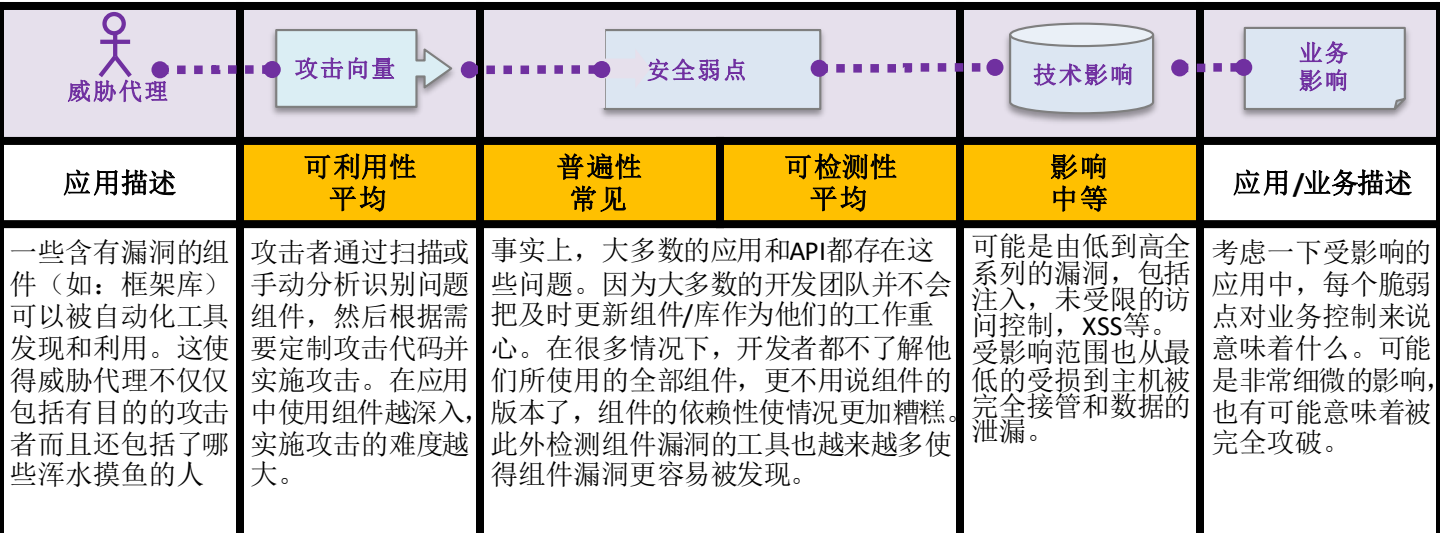
参考资料

OWASP

- [OWASP CSRF Article](#)
- [OWASP CSRF Prevention Cheat Sheet](#)
- [OWASP CSRGuard- 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](#)

外部资源

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



我存在使用已知漏洞组件的漏洞？

我们面临的挑战是根据新的漏洞报告持续监控使用的组件（客户端和服务端），但很可能由于漏洞报告没有使用标准化式而使得监控困难，并难以搜索你想要的细节（如产品中哪些具体的组件具有此漏洞）。更糟糕的是，很多漏洞从来不上报给漏洞中心如[CVE](#)和[NVD](#)。

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。这个过程可以手动完成或者使用自动化工具。如果发现了组件中的漏洞，请仔细检查应用是否真的会收到影响。检查代码中是否使用了具有漏洞的组件，这个漏洞是否对你的业务造成了影响。现实情况是很可能由于漏洞报告中描述得很模糊导致检测难以进行。

攻击案例场景

很多时候组件都是以最高权限运行的，这使得组件里的缺陷可能导致各式各样的问题。这些缺陷可能是一些偶然的（如编码错误）也可能是蓄意的（如组件里的后门）。下面是一些发布具有可以被利用漏洞的组件：

- [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web service框架，不要与Apache应用服务器混淆。）
- [Struts2远程漏洞执行](#) — 在Content-Type报头中发送一个攻击会将报头中的内容作为OGNL表达式进行执行，这样就可以导致在服务器端执行任何代码。

使用上述任意一个组件的应用程序都易受到攻击，因为两个组件都会被用户直接访问。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。

我如何防止？

很多第三方组件项目对旧版本并不发布补丁，唯一的解决方案就是升级到下一个版本，而这可能需要更改代码。软件项目应该遵循下面的流程：

1. 利用工具如[versions](#)、[DependencyCheck](#)、[retire.js](#)等来持续的记录客户端和服务端以及它们的依赖库的版本信息
2. 对使用的组件持续监控如[NVD](#)等漏洞中心，可以使用自动化工具来完成此功能
3. 在升级组件之前请分析一下它们是否在程序运行的时候被调用到了，很多组件其实从来没被加载或调用过
4. 决定到底是升级组件（如果必要可能需要重写应用代码来与之匹配），还是部署一个[虚拟补丁](#)分析HTTP流量、数据流、代码执行来防止漏洞被利用

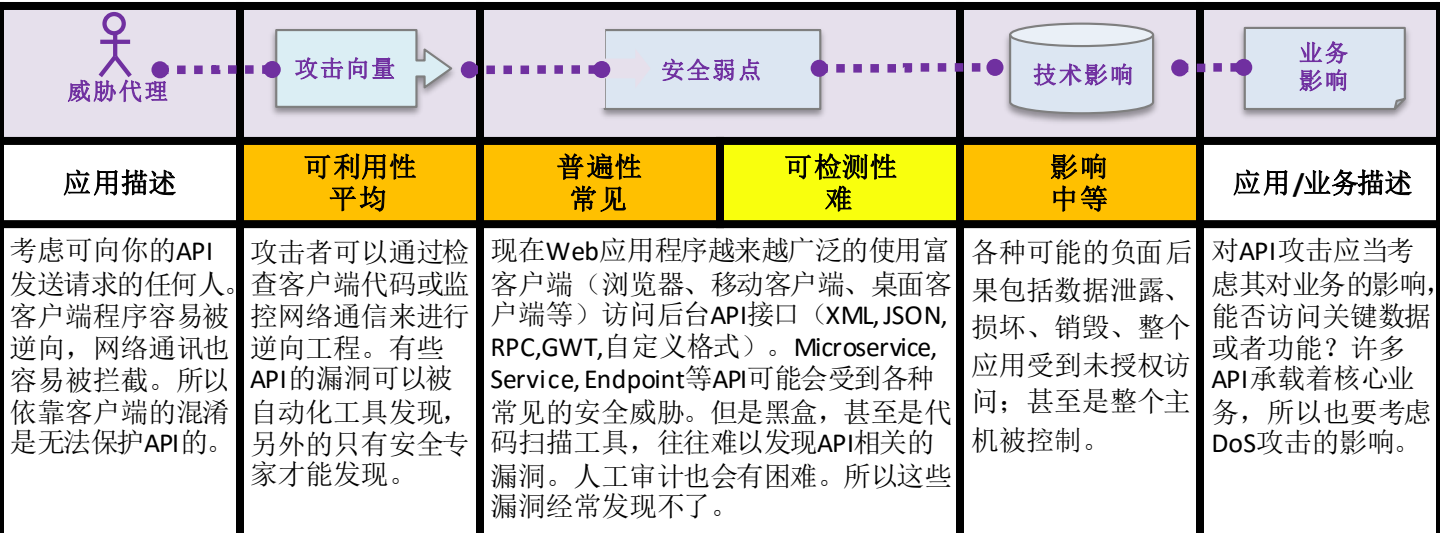
参考资料

OWASP

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

外部资源

- [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](#)



我易受攻击吗？

测试API的漏洞和测试一般应用层面的漏洞是相似的。各种注入、认证、访问控制、加密、配置等普通应用程序存在的问题在API中一样会存在。

然而，因为API一般是设计给程序而不是人使用，一般不提供UI，而且使用比较复杂的协议和数据结果。这使得安全测试会比较困难。如果API使用了比较常见的格式，比如Swagger（OpenAPI）、REST、JSON、XML，那么安全测试会容易一些。而GWT这样的框架，以及有些RPC实现使用了非标准的数据格式。有些应用和API使用了自定义的协议和数据格式，比如WebSockets。API技术特性的广泛性和复杂性让自动化安全测试难以得到有效结果，这往往会导致安全的错觉。

总之，需要仔细考虑一种测试策略去测试所有安全防护措施，才能了解你的API是否安全。

攻击案例场景

场景 #1: 想像一下有一个银行移动APP，通过访问XML API来访问账户信息，进行交易操作。攻击者逆向分析了移动APP，发现登录请求中，银行账号和用户名、密码一起发送给了服务器端的API。攻击者篡改发送了一个带着有效用户名、密码的登录请求，但是银行账号确实是别人的，通过这个请求，成功登录，然后完全控制别人的银行账号

场景 #2: 有一个互联网创业公司提供了一个公开的API，用以自动发送短消息。这个API接受JSON格式的请求，其中有transactionid参数。API把transactionid作为字符串来解析，然后拼接到SQL查询里，没有做任何参数化或者转义。所以这里的API和其他应用一样会受到SQL注入攻击

上面的两个例子中，供应商可能没有提供Web界面来访问这些服务，从而造成安全测试更加困难

我如何阻止对API的攻击？

保护API的关键是全面理解威胁模型，了解所有可用的防护措施：

1. 确保Client和API之间通过安全信道进行通讯。
2. 确保API有强安全级别的认证模式，并且所有的凭据、密钥、令牌都得到保护
3. 确保不管使用哪种数据格式，解析器都应当做好安全加固，防止攻击
4. 为API访问实现权限控制，防止不当访问，包括未授权的功能访问和数据引用
5. 防护所有各种形式的注入攻击，因为这些攻击对API和攻击普通应用是一样有效。

确保安全分析和测试涵盖所有的API，确保安全工具有效发现和分析它们。

参考资料

OWASP

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

外部资源

- [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安全风险，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。若您需要管理一个大型的应用程序系统群，那任务将十分艰巨。

为帮助企业 and 开发人员以节省成本的方式降低应用程序的安全风险，OWASP创建了相当多的免费和开源的资源。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为帮助企业组织创建安全的web应用程序提供的一些资源。在下一页中，我们将展示其它可以帮助企业用于检查web应用程序和接口安全性的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）》](#)作为您的应用设置安全需求的指导。ASVS在最近几年更新得非常好，最新版是2016年中的3.0.1版。如果您的应用程序是外包的，建议使用[《OWASP安全软件合同附件》](#)。

应用程序安全架构

- 与其费力去提升应用程序和接口的安全，不如在应用程序开发的初始阶段就进行安全设计，这样更能节约成本。作为好的开始，OWASP推荐[《OWASP开发者指南》](#)和[《OWASP Prevention Cheat Sheets》](#)，用于指导如何在应用程序开发的初始阶段进行安全设计。2013版TOP 10 发布以来，Cheat Sheet获得了极大的更新和扩展。

标准的安全控制

- 建立强大并可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和接口的安全开发过程。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提供了Java语言的参考实现代码。许多通用框架都已经有了授权、验证、CSRF等安全控制。

安全开发生命周期

- 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面对特定风险的软件安全战略。2017年Open SAMM发布了一个重大的更新。

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](#)，[OWASP NodeJS Goat](#)，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。[OWASP项目网页](#)列明了所有的OWASP项目，并根据发布的版本情况进行编排（发布质量、Beta版和Alpha版）。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](#)。

建立持续性的应用安全测试

安全编码很重要。但验证你想达到的安全性是否真实存在、是否正确、是否像我们想的那样也很关键。应用程序安全测试的目标是提供这些证据。这项工作困难而复杂，敏捷和DevOps当前快速发展的过程给传统的方法和工具带来的极大的挑战。因此，我们强烈建议你思考如何专注于整个应用程序组合中重要的地方，并且低成本高收益。

当前安全风险变化很快，每年进行一次的扫描或渗透测试的日子已经过去了。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安全测试。通过安全自动化来加强现有的开发管道并不会减缓开发速度。无论你选择哪种方法，都需考虑一下每年随着应用系统的规模倍增的定期测试、修复、复测并重新部署应用程序的成本。

理解威胁模型

- 在你测试之前，请了解业务中需要耗时的重要部分。优先级来源于威胁模型，所以如果你还没有威胁模型，那么需要在测试开始前建立一个。考虑使用[《OWASP ASVS》](#)和[《OWASP测试指南》](#)作为指导标准，而不依赖工具厂商的结果来判断哪些是重要业务。

理解你的SDLC

- 你的应用安全测试方法必须与你们软件开发流程（SDLC）中的人员、工具和流程高度匹配。试图强制推动额外的步骤、门限和审查可能会导致摩擦、绕行和一定范围内的争议。寻找自然而然的机会去收集安全信息，然后将融合进你的流程。

测试策略

- 选择最简单、快速、准确的方法去验证每项需求。[OWASP Benchmark Projects](#)可以有助于评估各安全工具对OWASP TOP10风险的检测能力，从而为你的特定需求挑选出最合适的工具。注意考虑用于工具误报的人力成本和漏报的严重危害。

实现全面性和准确性

- 你不需要一切都要立刻测试。先关注那些重要的方面，然后随着时间扩展你的全面性。这意味着逐步扩展安全防御库和自动验证库，以及扩展应用系统和API本身的覆盖。目标是所有的应用程序和API基本安全性都能获得持续性的验证。

体现报告的价值

- 不管你测试得怎么专业，若无效与别人沟通都等于白做。展示你对程序运行的理解，从而建立互信关系。不必用晦涩难懂专业用语，清楚描述漏洞的滥用风险，然后在某场景下真实展现攻击。要对漏洞发现与利用难度及引发的后果做真实的评估。最后提交结果时请使用开发团队正在使用的文档工具格式，而不是简单的PDF。

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和APIs的安全。由于在生产环境中的应用程序和APIs的代码行数惊人，许多企业组织都在努力处理数量巨大的漏洞。OWASP建议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。为了实现应用程序的安全性，需要企业组织中的不同部门之间有效地协同工作，这包括安全和审计、软件开发、和商业与执行管理。它需要安全的可视化，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。它还需要我们付诸实践和投入，以最有效的方式降低风险，改善企业的安全。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行业](#)，从而定义重点改善的领域和执行计划。
- 得到管理层的批准，并建立针对整个IT组织的[应用程序安全意识宣传活动](#)。

基于风险的组合方法

- 从固有风险的角度来识别并[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排序您的所有应用程序和APIs。
- 建立保证准则，合理定义所需的覆盖范围和级别。
- 建立一个[通用的风险等级模型](#)，该模型中的可能性和影响要素应该与组织风险承受能力一致的。

建立雄厚的基础

- 建立一套集中关注的[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。

提高管理层对安全的可见度

- 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，应用程序覆盖率，按类型和实例数量衡量缺陷密度等等。
- 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，而不是弱点

虽然2007年以及更早版本的OWASP Top 10专注于识别最流行的“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偏差。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，和业务影响等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。




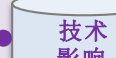
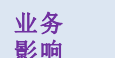
Top 10 的风险评级方法是基于《OWASP风险评级方法》。对于Top10中每一项，我们通过查看每个常见漏洞一般情况下的可能性因素和影响因素，评估了每个漏洞对于典型的Web应用程序造成的典型风险，然后根据漏洞给应用程序带来的风险程度的不同来对Top 10进行分级。

《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，Top10应该讨论普遍性，而不是在真实的应用程序和APIs中讨论具体的漏洞的风险。因此，我们无法像系统所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Top10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体可能性。这个等级同样没有将对您的业务的实际影响考虑进去。您的企业组织需要参照自身的企业文化，行业，以及监管环境，确定企业组织可以承受的应用安全和APIs的风险有多大。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”的风险评估方法。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”的普遍性值为0。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	 安全弱点		 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Top10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	可检测性		
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 (XSS)	应用描述	平均	非常广泛	平均	中等	应用描述
A4-失效的访问控制	应用描述	易	广泛	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-敏感信息泄漏	应用描述	难	少见	平均	严重	应用描述
A7-攻击检测和预防不足	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-使用已知含有漏洞的组件	应用描述	平均	常见	平均	中等	应用描述
A10-未受保护的API	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：

- [Clickjacking](#) ([CAPEC-103](#))
- [拒绝服务](#) ([CWE-400](#)) ([2004年版《OWASP Top 10》中的A9部分](#))
- [反序列化不可信数据](#) ([CWE-502](#)) 针对防御，请参见：[OWASP反序列化备忘录](#)
- [表达式语言注入](#) ([CWE-917](#))
- [信息泄漏](#) ([CWE-209](#)) 和 [不恰当的错误处理](#) ([CWE-388](#)) ([2007年版《OWASP Top 10》的A6部分](#))
- [链接第三方内容](#) ([CWE-829](#))
- [恶意文件执行](#) ([CWE-434](#)) ([2007年版《OWASP Top 10》的A3部分](#))
- [质量分配](#) ([CWE-915](#))
- [服务器端请求伪造 \(SSRF\)](#) ([CWE-918](#))
- [未验证的重定向和转发](#) ([CWE-601](#)) ([2013年版《OWASP Top 10》的A10部分](#))
- [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 title'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.