**安全标准征求意见稿**

**征求意见-安全内容大纲**

**1 概述**

**1.1 背景简介**

**1.2 技术框架**

**1.3 使用对象**

**1.4 适用范围**

**1.5 用词约定**

**2 常见Web安全漏洞**

**3 Web设计安全规范**

**3.1 Web部署要求**

**3.2 身份验证**

**3.2.1 口令**

**3.2.2 认证**

**3.2.3 验证码**

**3.3 会话管理**

**3.4 权限管理**

**3.5 敏感数据保护**

**3.5.1 敏感数据定义**

**3.5.2 敏感数据存储**

**3.5.3 敏感数据传输**

**3.6 安全审计**

**3.7 Web Service**

**3.8 RESTful Web Service**

**4 Web编程安全规范**

**4.1 输入校验**

**4.2 输出编码**

**4.3 上传下载**

**4.4 异常处理**

**4.5 代码注释**

**4.6 归档要求**

**4.7 其他**

**4.8 PHP**

**5 Web安全配置规范**

**6 验证码**

**征求意见-参考内容**

1. 安全知识

常见web安全漏洞

Web应用的安全漏洞有很多，无法穷举。针对众多的Web漏洞，OWASP的专家们结合各自在各领域的应用安全工作经验及智慧，提出了十大Web应用程序安全漏洞，帮助人们关注最严重的漏洞。（OWASP即开放Web应用安全项目，是一个旨在帮助人们理解和提高Web应用及服务安全性的项目组织。）

十大Web应用程序安全漏洞列表

|  |  |  |
| --- | --- | --- |
| **序号** | **漏洞名称** | **漏洞描述** |
| 1 | 注入 | 注入攻击漏洞，例如SQL、OS命令以及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。 |
| 2 | 跨站脚本 | 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。 |
| 3 | 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。 |
| 4 | 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。 |
| 5 | 跨站请求伪造 | 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。 |
| 6 | 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。这包括了对所有的软件保护及时地更新，包括所有应用程序的库文件。 |
| 7 | 失败的URL访问权限限制 | 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装这些URL去访问隐藏的网页。 |
| 8 | 未经验证的重定向和前转 | Web应用程序经常将用户重定向和前转到其他网页和网站，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，攻击者可以将受害用户重定向到钓鱼网站或恶意网站，或者使用前转去访问未经授权的网页。 |
| 9 | 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）、认证凭据等。攻击者可能利用这种弱保护数据实行身份盗窃、信用卡欺骗或或其他犯罪。 |
| 10 | 传输层保护不足 | 应用程序时常没有身份认证、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。 |

1. web设计规范
2. web部署要求

**规则3.1.1：**如果 Web 应用对 Internet 开放，Web服务器应当置于DMZ区，在Web服务器与Internet之间，Web服务器与内网之间应当有防火墙隔离，并设置合理的策略。

**规则3.1.2：**如果 Web 应用对 Internet 开放，Web服务器应该部署在其专用的服务器上，应避免将数据库服务器或其他核心应用与Web服务器部署在同一台主机上。

说明：Web服务器比较容易被攻击，如果数据库或核心应用与Web服务器部署在同一台主机，一旦Web服务器被攻陷，那么数据库和核心应用也就被攻击者掌控了。

**规则3.1.3：**Web站点的根目录必须安装在非系统卷中。

说明：Web站点根目录安装在非系统卷，如单独创建一个目录/home/Web作为Web站点根目录，能够防止攻击者使用目录遍历攻击访问系统工具和可执行文件。

**建议3.1.1：**Web服务器与应用服务器需物理分离（即安装在不同的主机上），以提高应用的安全性。

**建议3.1.2：**如果Web应用系统存在不同的访问等级（如个人帐号使用、客户服务、管理），那么应该通过不同的Web服务器来处理来自不同访问等级的请求，而且Web应用应该鉴别请求是否来自正确的Web服务器。

说明：这样便于通过防火墙的访问控制策略和Web应用来控制不同访问等级的访问，比如通过防火墙策略控制，只允许内网访问管理Portal。

**建议3.1.3：**对于“客户服务”和“管理”类的访问，除了普通的认证，还应该增加额外的访问限制。

说明：额外的访问限制，可以限制请求来自企业内网，可以建立VPN，或采用双向认证的SSL；或采用更简单的办法，通过IP地址白名单对客户端的IP地址进行过滤判断。

1. 身份验证

**口令**

待定

**认证**

**规则3.2.2.1：**对用户的最终认证处理过程必须放到应用服务器进行。

说明：不允许仅仅通过脚本或其他形式在客户端进行验证，必须在应用服务器进行最终认证处理（如果采用集中认证，那么对用户的最终认证就是放在集中认证服务器进行）。

**规则3.2.2.2：**网页上的登录/认证表单必须加入验证码。

说明：使用验证码的目的是为了阻止攻击者使用自动登录工具连续尝试登录，从而降低被暴力破解的可能。如果觉得验证码影响用户体验，那么可以在前3次登录尝试中不使用验证码，3次登录失败后必须使用验证码。验证码在设计上必须要考虑到一些安全因素，以免能被轻易地破解。具体实现细节请查看3.2.3验证码。如图：



验证码

实施指导：

建议使用电信软件与核心网网络安全工程部提供的[验证码CBB](http://wss-rnd.huawei.com/sites/ipd-tmg-anquan/CBB1/验证码CBB.htm)。

备注：对于嵌入式系统，如果实现验证码比较困难，可以通过多次认证失败锁定客户端IP的方式来防止暴力破解。

**规则3.2.2.3：**用户名、密码和验证码必须在同一个请求中提交给服务器，必须先判断验证码是否正确，只有当验证码检验通过后才进行用户名和密码的检验，否则直接提示验证码错误。

说明：如果验证码和用户名、密码分开提交，攻击者就可以绕过验证码校验（如：先手工提交正确的验证码，再通过程序暴力破解），验证码就形同虚设，攻击者依然可以暴力破解用户名及口令。

**规则3.2.2.4：**所有登录页面的认证处理模块必须统一。

说明：可以存在多个登录页面，但是不允许存在多个可用于处理登录认证请求的模块，防止不一致的认证方式。

**规则3.2.2.5：**所有针对其他第三方开放接口的认证处理模块必须统一。

**规则3.2.2.6：**认证处理模块必须对提交的参数进行合法性检查。

说明：具体输入校验部分请查看

**规则3.2.2.7：**认证失败后，不能提示给用户详细以及明确的错误原因，只能给出一般性的提示。

说明：可以提示：“用户名或者口令错误，登录失败”；不能提示：“用户名不存在”、“口令必须是 6 位”等等。

**规则3.2.2.8：**最终用户portal和管理portal分离。

说明：最终用户portal和管理portal分离，防止相互影响，防止来自用户面的攻击影响管理面。

实施指导：

将最终用户portal和管理portal分别部署在不同的物理服务器；如果为了解决成本合设（部署在同一台物理服务器上），那么，必须做到端口分离（通过不同的端口提供Web服务），一般的Web容器（如tomcat）支持为不同的Web应用创建不同的端口。

**规则3.2.2.9：**禁止在系统中预留任何的后门帐号或特殊的访问机制。

**规则3.2.2.10：**对于重要的管理事务或重要的交易事务要进行重新认证，以防范会话劫持和跨站请求伪造给用户带来损失。

说明：重要的管理事务，比如重新启动业务模块；重要的交易事务，比如转账、余额转移、充值等。重新认证，比如让用户重新输入口令。

**规则3.2.2.11：**用户名和密码认证通过后，必须更换会话标识，以防止会话固定（session fixation）漏洞。

实施指导：

|  |
| --- |
| 场景一：对于从HTTP转到HTTPS再转到HTTP（也就是仅在认证过程采用HTTPS，认证成功后又转到HTTP）的，在用户名和密码认证通过后增加以下行代码：  request.getSession().invalidate();  HttpSession newSession=request.getSession(true);  Cookie cookie = new Cookie("JSESSIONID",newSession.getId());  cookie.setMaxAge(-1);  cookie.setSecure(false);  cookie.setPath(request.getContextPath());  response.addCookie(cookie);  场景二：对于全程采用HTTPS协议，或者全程采用HTTP协议的，在用户名和密码认证通过后增加以下行代码：  request.getSession().invalidate();  request.getSession(true); |

**建议3.2.2.1：**管理页面建议实施强身份认证。

说明：如双因素认证、SSL双向证书认证、生物认证等；还可以通过应用程序限制只允许某些特定的IP地址访问管理页面，并且这些特定的IP地址可配置。

**建议3.2.2.2：**同一客户端在多次连续尝试登录失败后，服务端需要进行用户帐号或者是客户端所在机器的 IP 地址的锁定策略，且该锁定策略必须设置解锁时长，超时后自动解锁。

说明：

登录失败应该提示用户：如果重试多少次不成功系统将会锁定。在锁定期间不允许该用户帐号（或者客户端所在机器的 IP 地址）登录。

允许连续失败的次数（指从最后一次成功以来失败次数的累计值）可配置，取值范围为：0-99 次，0表示不执行锁定策略，建议默认：5 次。

锁定时长的取值范围为：0-999 分钟，建议默认：30 分钟，当取值为 0 时，表示无限期锁定，只能通过管理员手动解锁（需要提供管理员对服务器锁定其它用户帐号/IP进行解锁的功能界面）。建议优先使用帐号锁定策略。

注意：应用程序的超级用户帐号不能被锁定，只能锁定操作的客户端所在的 IP，这是为了防止系统不可用。特别说明：锁客户端IP策略存在缺陷，当用户使用proxy上网时，那么锁定客户端IP会导致使用该proxy上网的所有用户在IP锁定期间都不能使用该Web应用；锁定用户帐户的策略也存在缺陷，当攻击者不断尝试某帐户的口令，就给该帐户带来拒绝服务攻击，使该帐户不可用。

**验证码**

**规则3.2.3.1：**验证码必须是单一图片，且只能采用 JPEG、PNG或GIF格式。

说明：验证码不能使用文本格式，不允许多图片组合（如用四个图片拼成的验证码）。

**规则3.2.3.2：**验证码内容不能与客户端提交的任何信息相关联。

说明：在使用验证码生成模块时不允许接收来自客户端的任何参数，例如：禁止通过getcode.jsp?code=1234的URL请求，将1234作为验证码随机数。

**规则3.2.3.3：**验证码模块生成的随机数不能在客户端的静态页面中的网页源代码里出现。

说明：在客户端网页上点击鼠标右键、选择“查看源文件”时，必须看不到验证码模块生成的随机数。

**规则3.2.3.4：**验证码字符串要求是随机生成，生成的随机数必须是安全的。

说明：对于java语言可以使用类 java.security.SecureRandom来生成安全的随机数。

**规则3.2.3.5：**验证码要求有背景干扰，背景干扰元素的颜色、位置、数量要求随机变化。

**规则3.2.3.6：**验证码在一次使用后要求立即失效，新的请求需要重新生成验证码。

说明：进行验证码校验后，立即将会话中的验证码信息清空，而不是等到生成新的验证码时再去覆盖旧的验证码，防止验证码多次有效；注意：当客户端提交的验证码为空，验证不通过。

1. 会话管理

**规则3.3.1：**使用会话cookie维持会话。

说明：目前主流的Web容器通过以下几种方式维持会话：隐藏域、URL重写、持久性cookie、会话cookie，但通过隐藏域、URL重写或持久性cookie方式维持的会话容易被窃取，所以要求使用会话cookie维持会话。如果条件限制必须通过持久性cookie维持会话的话，那么cookie信息中的重要数据部分如身份信息、计费信息等都必须进行加密。（cookie有两种：会话cookie和持久性cookie；会话cookie，也就是非持久性cookie，不设置过期时间，其生命期为浏览器会话期间，只要关闭浏览器窗口，cookie就消失了；会话cookie一般不存储在硬盘上而是保存在内存里。持久性cookie，设置了过期时间，被浏览器保存到硬盘上，关闭后再次打开浏览器，持久性cookie仍然有效直到超过设定的过期时间。）

**规则3.3.2：**会话过程中不允许修改的信息，必须作为会话状态的一部分在服务器端存储和维护。

说明：会话过程中不允许修改的信息，例如，当用户通过认证后，其用户标识在整个会话过程中不能被篡改。禁止通过隐藏域或URL重写等不安全的方式存储和维护。对JSP语言，就是应该通过session对象进行存储和维护。

**规则3.3.3：**当Web应用跟踪到非法会话，则必须记录日志、清除会话并返回到认证界面。

说明： 非法会话的概念就是通过一系列的服务端合法性检测（包括访问未授权资源，缺少必要参数等情况），最终发现的不是正常请求产生的会话。

**规则3.3.4：**禁止使用客户端提交的未经审核的信息来给会话信息赋值。

说明：防止会话信息被篡改，如恶意用户通过URL篡改手机号码等。

**规则3.3.5：**当用户退出时，必须清除该用户的会话信息。

说明：防止遗留在内存中的会话信息被窃取，减少内存占用。

实施指导：对于JSP或java语言使用如下语句：request.getSession().invalidate();

**规则3.3.6：**必须设置会话超时机制，在超时过后必须要清除该会话信息。

说明：建议默认会话超时时间为10分钟（备注：对于嵌入式系统中的Web，建议默认超时时间为5分钟，以减少系统资源占用）。如果没有特殊需求，禁止使用自动发起请求的机制来阻止session超时。

**规则3.3.7：**在服务器端对业务流程进行必要的流程安全控制，保证流程衔接正确，防止关键鉴别步骤被绕过、重复、乱序。

说明：客户端流程控制很容易被旁路（绕过），因此流程控制必须在服务器端实现。

实施指导：可以通过在session对象中创建一个表示流程当前状态的标识位，用0、1、2、3、…、N分别表示不同的处理步骤，标识位的初始值为0，当接收到步骤N的处理请求时，判断该标识位是否为N-1，如果不为N-1，则表示步骤被绕过（或重复或乱序），拒绝受理，否则受理，受理完成后更改标识位为N。

**规则3.3.8：**所有登录后才能访问的页面都必须有明显的“注销（或退出）”的按钮或菜单，如果该按钮或菜单被点击，则必须使对应的会话立即失效。

说明：这样做是为了让用户能够方便地、安全地注销或退出，减小会话劫持的风险。

**规则3.3.9：**如果产品（如嵌入式系统）无法使用通用的Web容器，只能自己实现Web服务，那么必须自己实现会话管理，并满足以下要求：

* 采用会话cookie维持会话。
* 生成会话标识（session ID）要保证足够的随机、离散，以便不能被猜测、枚举，要求session ID要至少要32字节，要支持字母和数字字符集。
* 服务端必须对客户端提交的session ID的有效性进行校验。

说明：在嵌入式系统中部署Web应用，由于软硬件资源所限，往往无法使用通用的Web容器及容器的会话管理功能，只能自己实现。另外，为了节省内存，嵌入式webserver进程往往是动态启动，为了使session更快的超时，建议增加心跳机制，对客户端浏览器是否关闭进行探测，5s一个心跳，30s没有心跳则session超时，关闭该session。