**渗透测试**

1. XSS跨站脚本攻击（反射型）
2. 攻击思路

在tp\_shop网页中嵌入客户端脚本（JavaScript编写的恶意代码），当用户使用tp\_shop时，恶意代码将会在用户的浏览器上执行。

1. 攻击过程

tp\_shop原URL地址为<http://localhost:9090/>，将恶意的脚本附加到URL地址的参数，即[http://localhost:9090/<script>document.cookie</script](http://localhost:9090/%3cscript%3edocument.cookie%3c/script)>；

编写如下代码：<?php  
     $username = $\_GET[‘username’];  
     echo $username;  
 ?>

将上述文件放在Apache根目录下，使用火狐浏览器打开，再次访问tp\_shop时，若提交xss.php?username=13800138006,则程序输出用户名13800138006

1. 攻击分析

① 将带有XSS代码的数据发送给用户浏览器，用户浏览器解析带有XSS代码的数据后造成漏洞，之后用户再访问的则是带有XSS代码的tp\_shop网址  
② 在开发浏览器时，应尽量防范反射型XSS漏洞：  
（PHP中直接输出html的，可以采用htmlspecialchars函数、htmlentities函数、HTMLPurifier.auto.php插件、removexss函数进行过滤；PHP输出到JS代码中，或者开发Json API的，则需要前端在JS中进行过滤。）

1. SQL注入漏洞
2. 攻击思路

通过把SQL命令插入到tp\_shop网页的登录表单中，最终达到欺骗服务器执行恶意的SQL命令。

1. 攻击过程

① 访问tp\_shop网站，寻找注入点；

② 构造SQL注入语句和程序中的SQL语句结合生成新的SQL语句：

var sql = “SELECT \* FROM users WHERE username = “’” + formusr + “’AND password = ‘” + formpwd + “’”;

接收用户输入查询语句：SELECT \*FROM users WHERE user=’13800138006’ AND password=’123456’

通过字符产注入，使用如下用户名和密码：formusr=’or 1=1--

formpassword=anything

之后查询语句就变为：SELECT \*FROM users WHERE username=’ ’ or 1=1 –AND password=’anything’

1. 攻击分析

有SQL注入漏洞的网站会使用户信息面临安全考验，可采用输入验证、加密处理、错误消息处理、使用专业的漏洞扫描工具等方法来防御SQL注入漏洞。

1. 文件上传漏洞
2. 攻击思路

IIS在解析文件名的时候会将分号后面的内容丢弃，攻击者会利用此问题，在上传的时候给文件名后 加入分号内容来避免黑名单过滤。

1. 攻击过程
2. 新建文本文件，写入：<?php

phpinfo();

?>

1. 保存该文件为bug.php.rar
2. 在火狐浏览器地址栏输入<http://localhost/bug.php.rar>，正常应该提示文件下载框，但此时显示phpinfo()的内容。
3. 攻击分析

Apache在解析文件时，当碰到不认识的扩展名时，将会从后向前解析直到碰到认识的扩展名为止，如果都不认识，则会暴露其源代码；可以利用白名单与黑名单过滤、MIME验证、目录验证等方式预防文件上传漏洞。

1. 点击劫持——图片覆盖攻击
2. 攻击思路

通过调整图片的style使得图片能够覆盖在所指定的任意位置，例如将一张图片覆盖在tp\_shop网页图书馆超链接上，当用户点击图书馆后，就会被链接到攻击者所指向的目的网站。

1. 攻击过程

通过调整图片的style使其能够覆盖在tp\_shop网页的图书馆超链接上：

<a href=” /index.php/Home/Goods/goodsList/id/83”>

<img src=”危险网站” right:320px;top:90px;/>

</a>

1. 攻击分析

由于<img>标签在很多系统中对用户开放，因此tp\_shop的站点存在被点击劫持攻击的可能；可以通过检查<img>标签的style属性是否可能导致浮出的方式防御点击劫持。

1. 跨站点请求伪造
2. 攻击思路

攻击者通过盗用身份，并用盗用来的身份发送恶意请求。

1. 攻击过程

① 用户浏览并登录tp\_shop网站，验证通过后产生tp\_shop的cookie；

② 用户在未退出tp\_shop登录的情况下访问一危险网站；

③ 该危险网站要求访问第三方站点tp\_shop，发出一个（request）；

④ 根据该危险网站的请求，浏览器带着上述cookie访问tp\_shop；

⑤ tp\_shop不知道该请求是谁发出的，由于浏览器会自动带上用户的cookie，所以tp\_shop会根据用户的权限处理该请求，这样，该危险网站就达到了模拟用户操作的目的。

1. 攻击分析

可以通过在客户端页面增加伪随机数，在表单中增加Hash值，以认证是否是用户发送的请求，然后在服务器端进行Hash值验证或验证码等的方式防御CSRF。