

SDL

◎ 编码

- 编码规范
 - 5大漏洞
 - Upload

SDL

◎ 测试

- XSS
- CSRF
- SQL注入
- URL跳转
- 权限控制
- 上传

SDL

◎ 代码安全review

- XSS
- CSRF
- SQL注入
- URL跳转
- 权限控制
- 上传

目录

- ◎ XSS
- ◎ CSRF
- ◎ 注入攻击
- ◎ 文件系统、系统命令
- ◎ 上传、下载
- ◎ 权限
- ◎ Cookie与会话
- ◎ 随机与加密

跨站脚本(XSS)

- ◎ XSS又叫CSS (Cross Site Script) ， 跨站脚本攻击。它指的是恶意攻击者往Web页面里插入恶意html代码，当用户浏览该页之时，嵌入其中Web里面的html代码会被执行，从而达到恶意用户的特殊目的。
 - 盗取Cookie
 - 钓鱼
 - 操纵受害者的浏览器
 - 蠕虫攻击

反射型跨站(Reflected XSS)

- 服务端获取HTTP请求中的参数，未经过滤直接输出到客户端。如果这些参数是脚本，它将在客户端执行。(钓鱼常见)

```
/* error.php */  
<?php  
echo "<p>Error: " . $_GET['msg']. "</p>";  
?>
```

- 请求<http://a.com/error.php?msg=file%20not%20found>

<p>Error: file not found</p>

- 请求
[http://a.com/error.php?msg=<script>alert\(123\)</script>](http://a.com/error.php?msg=<script>alert(123)</script>)

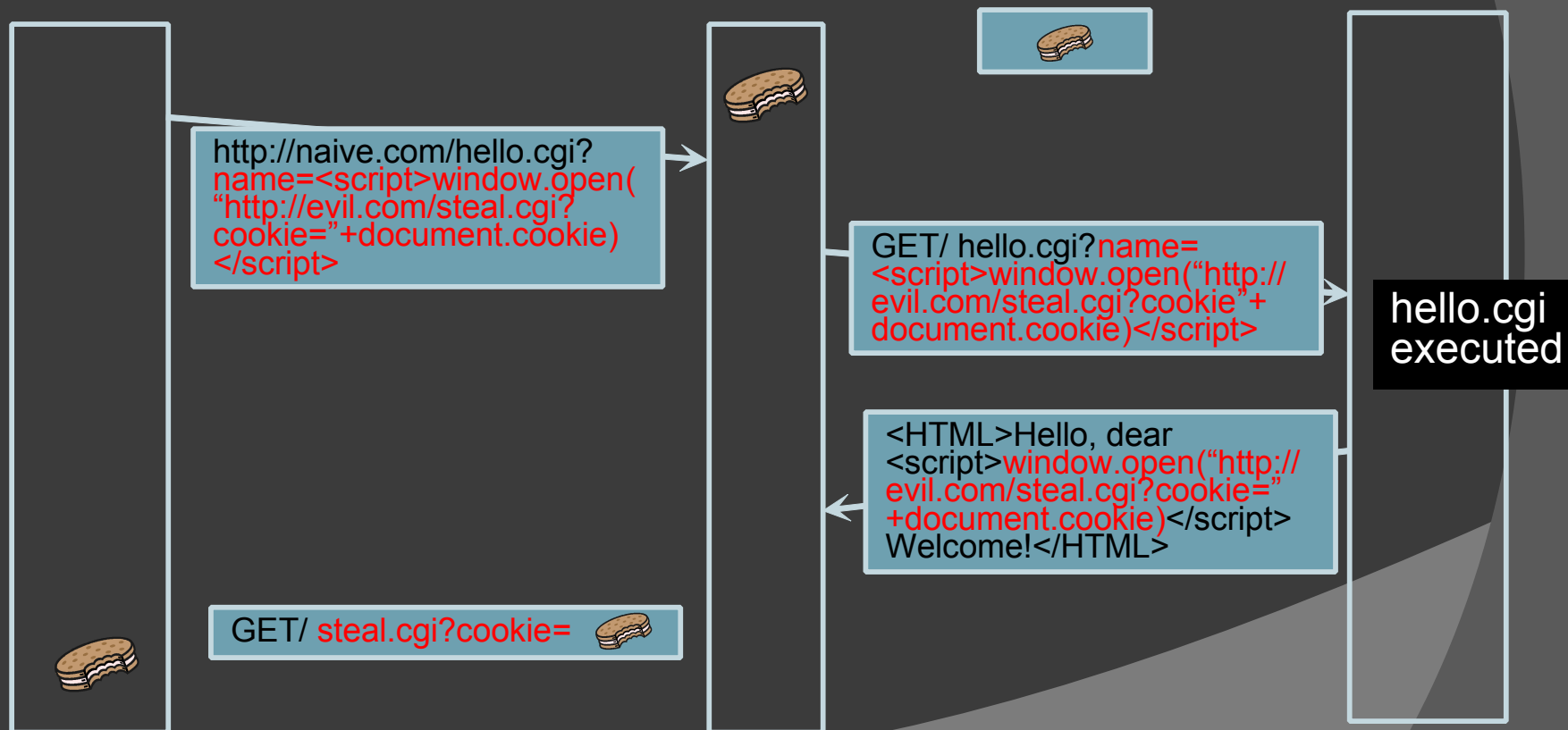
<p>Error: <script>alert(123)</script></p>

反射型跨站(Reflected XSS)

attacker

victim's
browser

naive.com



存储型跨站(Stored XSS)

- ① 用户输入的数据存放在服务端(一般放数据库里), 其他用户访问某个页面时, 这些数据未经过滤直接输出。这些数据可能是恶意脚本, 对其他用户造成危害。(挂马常见)

存储型跨站(Stored XSS)

- 在允许上传文件的应用中，攻击者上传一个包含恶意代码的html或txt文件，用户浏览这些文件时执行恶意代码。
- 一般的应用中上传图片最普遍，如果图片中包含恶意代码，低版的IE直接请求这个图片时，将忽略Content-Type执行图片中的代码。

HTTP/1.1 200 OK

Date: Sat, 5 May 2007 11:52:25 GMT

Server: Apache

Content-Length: 39

Content-Type: image/jpeg

`<script>alert(document.cookie)</script>`

DOM跨站(DOM-Based XSS)

- 攻击者构造一个包含恶意Javascript的URL，然后引诱用户请求这个URL。
- 服务器收到请求后没有返回恶意的Javascript。
- 浏览器在处理URL中的数据时，执行恶意代码。

```
<!-- test.html -->
<body>
<script>
document.write(document.location.hash);
</script>
</body>
```

请求

`http://a.com/test.html#<script>alert(1)</script>`

跨站请求伪造(CSRF)

- ◎ 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。
- ◎ 恶意请求会带上浏览器的Cookie。
- ◎ 受攻击的Web应用信任浏览器的Cookie。

```

```

Flash

◎ 客户端

- XSS
- CSRF

◎ 服务端

- `crossdomain.xml`

SQL注入

- ◎ 用户输入的数据未经验证就用来构造SQL查询语句。
 - 查询数据库中的敏感内容
 - 绕过认证
 - 添加、删除、修改数据
 - 拒绝服务
 - ?id=(BENCHMARK(100000000, MD5(RAND())));

SQL注入

- ◎ `$sql = "SELECT name FROM users WHERE id = " . $_GET['id'] . "";`
当id的值为: `1' or 1=1 --`
查询语句: `SELECT name FROM users WHERE id = '1' or 1=1 --'`
- ◎ `$sql = "SELECT * FROM admins WHERE name = " . $_GET['name'] . "" and pass=" . $_GET['pass'] . "";`
当name的值为: `' or 1=1 --`
查询语句: `SELECT * FROM admins WHERE name = " or 1=1 --' and pass=""`

XML注入

- 和SQL注入原理一样，XML是存储数据的地方，如果在查询或修改时，如果没有做转义，直接输入或输出数据，都将导致XML注入漏洞。攻击者可以修改XML数据格式，增加新的XML节点，对数据处理流程产生影响。

```
$xml = "<USER role=guest><name>" . $_GET['name'] .  
"</name><email>" . $_GET['email'] . "</email></USER>";
```

原本应该产生一条这样的记录：

```
<?xml version="1.0" encoding="UTF-8"?>  
<USER role=guest>  
    <name>user1</name>  
    <email>user1@a.com</email>  
</USER>
```

XML注入

如果用户输入email的内容是这样：

```
user1@a.com</email></USER><USER  
role=admin><name>test</name><email>user2@a.com
```

最终结果将是这样：

```
<?xml version="1.0" encoding="UTF-8"?>  
<USER role=guest>  
    <name>user1</name>  
    <email>user1@a.com</email>  
</USER>  
<USER role=admin>  
    <name>test</name>  
    <email>user2@a.com</email>  
</USER>
```

URL跳转

- ◎ Web应用程序接收到用户提交的URL参数后，没有对参数做“可信任URL”的验证，就向用户浏览器返回跳转到该URL的指令。
 - 钓鱼攻击
- ◎ <http://m.yahoo.cn/log.php?c=web&u=http://www.163.com>
- ◎ http://test.aliyun.com/user/index.php?pre_url=http://www.baidu.com

文件系统跨越

- ◎ 文件系统中../代表上级目录
- ◎ 通过一个或多个../跨越目录限制

```
$fp = fopen("image/{$_GET['filename']}", 'r');
```

```
Getfile?filename=../../../../etc/passwd
```

- %00截断路径名

```
echo file_get_contents($_GET['filename'].'.txt');
```

```
Getfile?filename=../../../../etc/passwd%00
```

系统命令

- 用户提交的参数用于执行系统命令的参数。
- 使用“|”或“;”执行多条命令
- passthru(), popen(), shell_exec(), system()
system("ls -l \${_GET['dir']}");
?dir=path; cat /etc/passwd

文件上传

- ◎ Web应用程序在处理用户上传的文件时，没有判断文件的扩展名是否在允许的范围内，或者没检测文件内容的合法性，就把文件保存在服务器上，甚至上传脚本木马到web服务器上，直接控制web服务器。
 - 未限制扩展名
 - 未检查文件内容
 - 病毒文件

任意文件下载

- ◎ 下载附件等功能
 - Apache虚拟目录指向
 - Java/PHP读取文件
- ◎ 下载数据库配置文件等
- ◎ 目录浏览

权限控制

- ◎ 有没有权限
 - 有些系统不需要权限控制
- ◎ 有没有设置权限
 - 有了强大的权限系统，但是没有使用
- ◎ 有没有偷工减料权限
 - URL级别
 - 菜单级别

访问控制

◎ 水平权限

- Web应用程序接收到用户请求，修改某条数据时，没有判断数据的所属人，或判断数据所属人时，从用户提交的request参数（用户可控数据）中，获取了数据所属人id，导致恶意攻击者可以通过变换数据ID，或变换所属人id，修改不属于自己的数据。

◎ 垂直权限

- 由于web应用程序没有做权限控制，或仅仅在菜单上做了权限控制，导致的恶意用户只要猜测其他管理页面的URL，就可以访问或控制其他角色拥有的数据或页面，达到权限提升目的。

Cookie

- ⦿ Cookie httponly flag
- ⦿ Cookie secure flag

Session Expires

- ◎ 会话过期
 - 浏览器过期
 - 服务器30分钟无动作过期
 - 服务器24小时强制过期
- ◎ 保持会话

随机与加密

- ◎ 随机
 - 不可预测
- ◎ 区分编码与加密
- ◎ 加密强度



Q&A