Web 前端安全

提纲

1. 跨站脚本（XSS）
2. 新开窗口漏洞
3. URL跳转漏洞
4. 跨站请求伪造（CSRF）
5. 劫持
6. DDOS攻击

XSS介绍

XSS：　（Cross Site Scripting），中文名为跨站脚本，是一种代码注入攻击。

原因：开发者对用户提交的内容没有进行合理的检查。

后果：恶意代码被执行，对网站进行了破坏或窃取了某些敏感信息。

XSS案例

DOM型XSS示例：

```
<input type="text" onchange="inputChangeHandle(this)">
<script>function inputChangeHandle(e){ document.write(e.value); }</script>
```

在input 中输入：<script>alert('xss')</script>

XSS案例

在php渲染前端的时代

HTML节点内容

```
<div>
    ${content}
</div>
```

转义< -> < 和 > -> >

如果不转译，则会直接当成html渲染

HTML属性

```
<img clsss='${class}' />
<img class="${class}" />
<img src=/404.png onerror=alert(1)  />
```

转义” -> &quto; ' -> ' 和 空格 ->

如果不转译，则class=‘’imgclass’’，破坏了dom结构

十进制html实体编码

```
<a
href=&#0000106&#0000097&#0000118&
#0000097&#0000115&#0000099&#0000
114&#0000105&#0000112&#0000116&#
0000058&#0000097&#0000108&#00001
01&#0000114&#0000116&#0000040&#0
000039&#0000088&#0000083&#000008
3&#0000039&#0000041>Click</a>
```

转义 & -> &

如果不转译，则会直接渲染成

```
<a href="javascript:alert('XSS')">Click</a>
```

React Vue 等前端渲染框架会自动转译

XSS案例

php渲染前端，React，Vue 中需要注意的特定属性

a的href属性

<a href="javascript:alert(1)" />

不区分大小写

<a href="JaVaScRiPt:alert(1)" />

TAB键分隔以及被编码的TAB

（新版浏览器已能自动预防tab）

<a href=ja vascript:alert(1) />

<a href=jav	ascript:alert('XSS')>

iframe的src属性

<iframe src="data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg==">

```
<iframe src="data:text/html;base64,PHNjc…wdD5hbGVydCgxKTwvc2NyaXB0Pg==">
  #document
    <html>
      <head>
        <script>alert(1)</script>
      </head>
      <body></body>
    </html>
</iframe>
```

object的data属性

<object data="data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg==">

```
<object data="data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg==">
  #document
    <html>
      <head>
        <script>alert(1)</script>
      </head>
      <body></body>
    </html>
</object>
```

XSS案例

JSON输出

特定Unicode转译

```
'中' === '\u4e2d' // true
```

但是，JavaScript 中有些字符不能在字符串里面直接使用，只能使用转义形式。例如：

- U+005C：反斜杠（reverse solidus)
- U+000D：回车（carriage return）
- U+2028：行分隔符（line separator）
- U+2029：段分隔符（paragraph separator）
- U+000A：换行符（line feed）

举例来说，字符串里面不能直接包含反斜杠，一定要转义写成\\或者\u005c

参考：阮一峰 ECMAScript 6 (ES6) 标准入门教程 第三版

XSS案例

JSON实践

```
// chrome 下
JSON.parse('{"name":"\u0008"}') // 报错
JSON.parse('{"name":"\u0009"}') // 报错
JSON.parse('{"name":"\u000A"}') // 报错
JSON.parse('{"name":"\u000B"}') // 报错
JSON.parse('{"name":"\u000C"}') // 报错
JSON.parse('{"name":"\u000D"}') // 报错
JSON.parse('{"name":"\u0022"}') // 报错
JSON.parse('{"name":"\u0027"}') // 没出错
JSON.parse('{"name":"\u005C"}') // 报错
JSON.parse('{"name":"\u00A0"}') // 没出错
JSON.parse('{"name":"\u2028"}') // 报错
JSON.parse('{"name":"\u2029"}') // 报错
JSON.parse('{"name":"\uFEFF"}') // 没出错
```

下面三个字符unicode 可转可不转（某些资料建议转）

```
console.log(JSON.parse('{"name":"<", "unicode:"\u003C"}'))
console.log(JSON.parse('{"name":">"}, "unicode:"\u003E"'))
console.log(JSON.parse('{"name":"/", "unicode:"\u002F"}'))
```

Javascript中的特殊unicode一共有13个，建议都进行转义处理，如下：

Unicode 字符值	转义序列	含义	类别
\u0008	\b	Backspace	
\u0009	\t	Tab	空白
\u000A	\n	换行符（换行）	行结束符
\u000B	\v	垂直制表符	空白
\u000C	\f	换页	空白
\u000D	\r	回车	行结束符
\u0022	\"	双引号 (")	
\u0027	\'	单引号 (')	
\u005C	\\	反斜杠 (\)	
\u00A0		不间断空格	空白
\u2028		行分隔符	行结束符
\u2029		段落分隔符	行结束符
\uFEFF		字节顺序标记	空白

XSS分类

类型	存储区	插入点	说明
存储型	后端数据库	HTML	网站服务端将恶意代码从数据库取出，拼接在 HTML 中返回给浏览器。（常见于服务端渲染，或者php渲染）
反射型	URL参数	HTML	网站服务端将恶意代码从 URL 中取出，拼接在 HTML 中返回给浏览器。（常见于搜索跳转时url带参数）
DOM型	后端数据库/前端存储/URL参数	前端JavaScript	取出和执行恶意代码由浏览器端完成，属于前端 JavaScript 自身的安全漏洞。（如ajax获取db中的数据，或者js获取url中的参数，然后innerhtml到dom中）

XSS预防

预防存储型和反射型

预防这两种漏洞，有两种常见做法：

- 对 HTML 做充分转义（在服务端选择则需要在服务器转译，在前端渲染，则需要在前端转译）。
- 改成纯前端渲染，把代码和数据分隔开。

XSS预防

预防 DOM 型 XSS 攻击

- 谨慎使用改变dom的API .innerHTML、.outerHTML、document.write()。
- Vue/React 技术栈，谨慎使用 v-html/dangerouslySetInnerHTML 功能
- 警惕DOM 中的内联事件监听器，如 location、onclick、onerror、onload、onmouseover 等，<a> 标签的 href 属性，JavaScript 的 eval()、setTimeout()、setInterval() 等。

XSS预防

其他XSS预防措施

- Content Security Policy
- 输入内容长度控制：增加 XSS 攻击的难度。
- HTTP-only Cookie：禁止 JavaScript 读取某些敏感 Cookie，攻击者完成 XSS 注入后也无法窃取此 Cookie。
- 验证码，防止脚本冒充用户提交危险操作。

XSS总结

1. 谨慎调用dom api，尽可能调用安全的dom api
2. 对html内容进行转译： < > ‘ “ & 空格 /
3. 特定属性如src, href需要校验是否有javascript: data: 等关键词（这些关键词可能带有空格，如 j a vaScripT: ）
4. json场景需要转义 < > / 和不可见字符
5. React，Vue 可以预防大多数 XSS，但是有很多也需要我们注意，比如：v-html / dangerouslySetInnerHTML 和内联的 onclick、 href、 src 等等属性。

新开窗口漏洞

打开新页面存在安全问题

- window.open()
- a标签用target=”_blank”

示例：以下两种方式打开b页面，在b页面都可以通过window.opener获取a的引用

- window.open(‘b.html’)
- 到b页面

	window.opener是否有值	跨域	不跨域
1	window.open(‘https://baidu.com');	有	有
2	location.href=”https://baidu.com";	无	无
3	< a href=”https://baidu.com" target=”_balank”>百度 	有	有
4	< a href=”https://baidu.com" >百度 	无	无

新开窗口漏洞预防

防止新窗口的window.opener有值的方法

1. location.href=”https://baidu.com“; // 可直接使用
2. < a href=”https://baidu.com“ >百度 // 可直接使用
3. < a href=”https://baidu.com“ target=”_balank”>百度 // 不可直接使用

   a 链接解决办法：在a链接中加入rel=”noopener noreferrer”,如下：

   进入一个有漏洞的页面

4. 不要直接使用 window.open()方法，需要进行如下包装。

```
function openUrl(url) {
        var newTab = window.open();
        newTab.opener = null;
        newTab.location = url;
}
```

URL跳转漏洞介绍

1. 腾讯云的统一登录模块。

   https://cloud.tencent.com/login?s_url=https%3A%2F%2Fconsole.cloud.tencent.com%2Ftbaas%2Fbcos%2Fdeploy。

   经过decodeURIComponent解析得到：

   'https://cloud.tencent.com/login?s_url=https://console.cloud.tencent.com/tbaas/bcos/deploy‘

2. 微信内H5登录跳转，要传appid，redirect_uri 等参数。其中redirect_uri 就是微信登录成功后跳转的参数。

URL跳转漏洞预防

对要跳转的域名进行白名单校验。

通过正则的方式来验证白名单和参数值需要注意：

防止被利用@符号伪造合法域名。

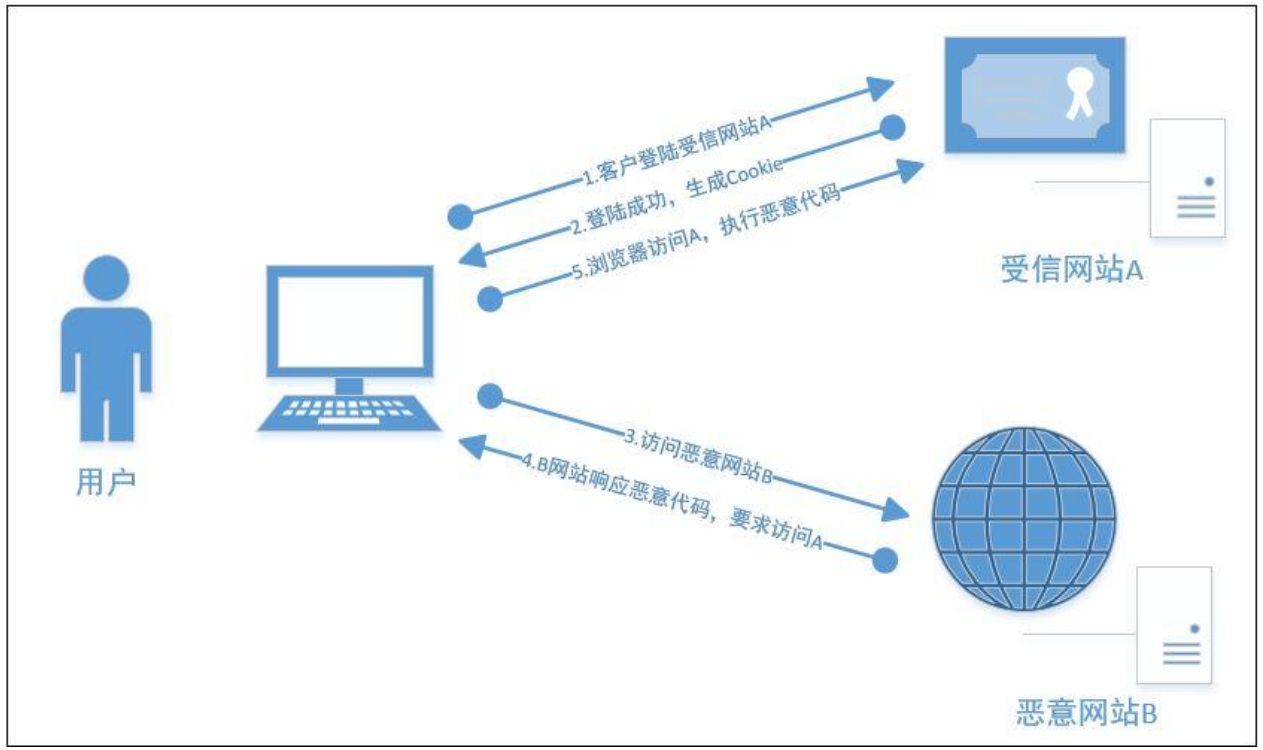
https://taobao.com@baidu.com/ ——> 浏览器 ——> baidu.com

白名单注意事项：

1. 正则校验URL协议（不要太依赖框架）
2. 过滤掉\n\r （有些浏览器会被换行符干扰）
3. 校验path里是否有@
4. 极端场景需要协议+域名+path一起校验

CSRF介绍

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



参考文章：https://tech.meituan.com/2018/10/11/fe-security-csrf.html （作者：刘烨）

CSRF案例

GET类型

GET类型的CSRF利用非常简单，只需要一个HTTP请求

- <img src="https://bank.com/fn?param=1">
- <iframe src="https://bank.com/fn?param=1">
- <script src="https://bank.com/fn?param=1">
- <a href=" https://bank.com/fn?param=1 " taget="_blank"> 重磅消息！！ <a/> //不常见

POST类型

这种类型的CSRF利用起来通常使用的是一个自动提交的表单

```
<body onload="document.forms[0].submit()">
<form method="POST" action="https://bank.com/fn">
        <input type="hidden" name="sp" value="8109"/>
</form>
```

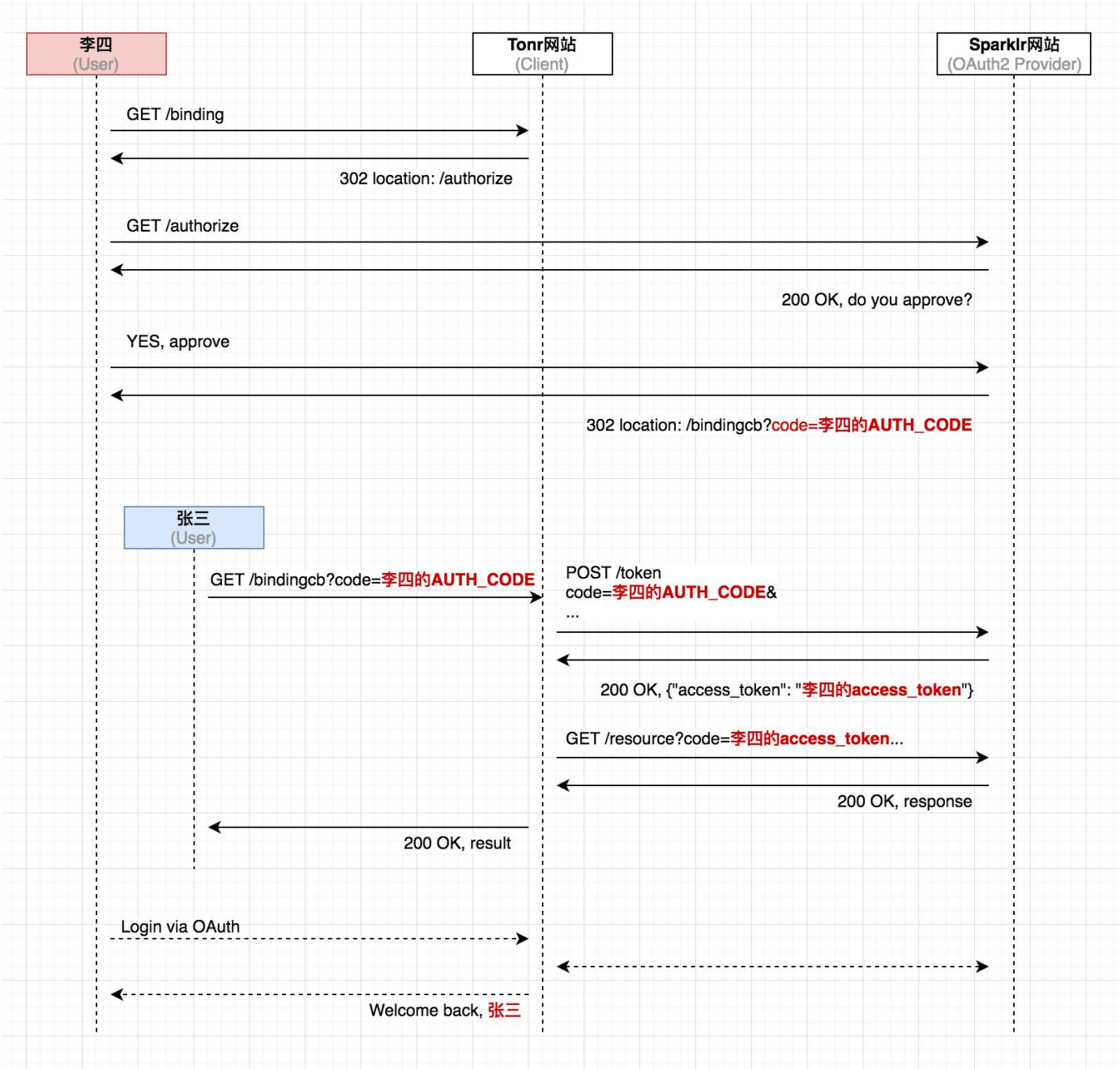
CSR案例

背景：平台登录账号与微信号绑定

李四：黑客

张三：小白用户

目的：李四用自己的微信登录张三的平台账号



CSRF防御

- 阻止不明外域的访问
  - 同源检测
  - Samesite Cookie
- 提交时要求附加本域才能获取的信息
  - CSRF Token
  - 双重Cookie验证
  - 验证码、密码
  - X-Content-Type-Options: nosniff

劫持

DNS劫持

案例：输入京东被强制跳转到淘宝这就属于dns劫持。

劫持方式：本机DNS劫持、路由DNS劫持、攻击DNS服务器。

解决办法：DNS劫持由于涉嫌违法,已经被监管起来,现在很少会有DNS劫持,而http劫持依然非常盛行.

（第一种是电脑host被修改了，后两种则运营商存在责任，我们无法防御，发现只能直接工信部电话12300投诉）

劫持

HTTP劫持

1. iframe嵌套展示原来正常网页
2. 在原html中插入js，再通过js脚本安插广告
3. 直接返回一个带广告img的HTML

HTTP劫持解决方案：

1. 设置HTTP头 X-Frame-Options
2. 检查是否被iframe嵌套：if (window.self != window.top) { var url = location.href; top.location = url; }
3. 对于在返回html内容中插入广告，我们可以借鉴注入的方式，进入页面就检测的img图片路径是否在白名单内
4. 如果页面有使用到iframe，src不确定，则可以通过sanbox属性提高iframe安全性。
5. 最有效的办法就是全站HTTPS,将HTTP加密,这使得运营商无法获取明文,就无法劫持你的响应内容.，防止中间人攻击

DDOS攻击

分布式拒绝服务攻击(Distributed Denial of Service，简称DDoS)是指处于不同位置的多个设备同时向一个或数个目标发动攻击

攻击方式

1. SYN Flood攻击
2. ACK Flood攻击
3. UDP Flood攻击
4. ICMP FLOOD攻击
5. Connection Flood攻击
6. HTTP Get攻击
7. UDP DNS Query Flood攻击

Hybrid安全

目前 Hybrid 开发形式有三种：

1.原生开发一部分+H5开发一部分，通过 WebView 桥接；

2. 纯 H5 开发，需要本地功能时通过第三方打包工具实现 如 HBuilder+ ；

3. 用 H5 或 JS 开发，但最终编译成 Native 应用，如 ReactNative、APICloud、Cocos2d-js、Unity3D-js 等。还有种形式是 Webview+Runtime ，一般用在H5游戏加速，像腾讯 X5 浏览器、UC 等都内置了 Runtime。

WebView中的安全漏洞主要来自三个方面:

1. WebView任意代码执行，WebView可创造执行原生native api的对象，需要充分了解Hybrid 框架的WebView api
2. 密码明文存储， WebView默认开启密码保存功能 webView.setSavePassword(true)
3. 域控制不严格