前端安全问题及防范措施

讲师：柳明

2019年8月

目录

CONTENTS

前端安全概述

如何理解?

- 前端是指一个Web应用中运行在客户端的部分，大多数情况下，指的就是运行在浏览器中由HTML组建起来的脚本等资源。

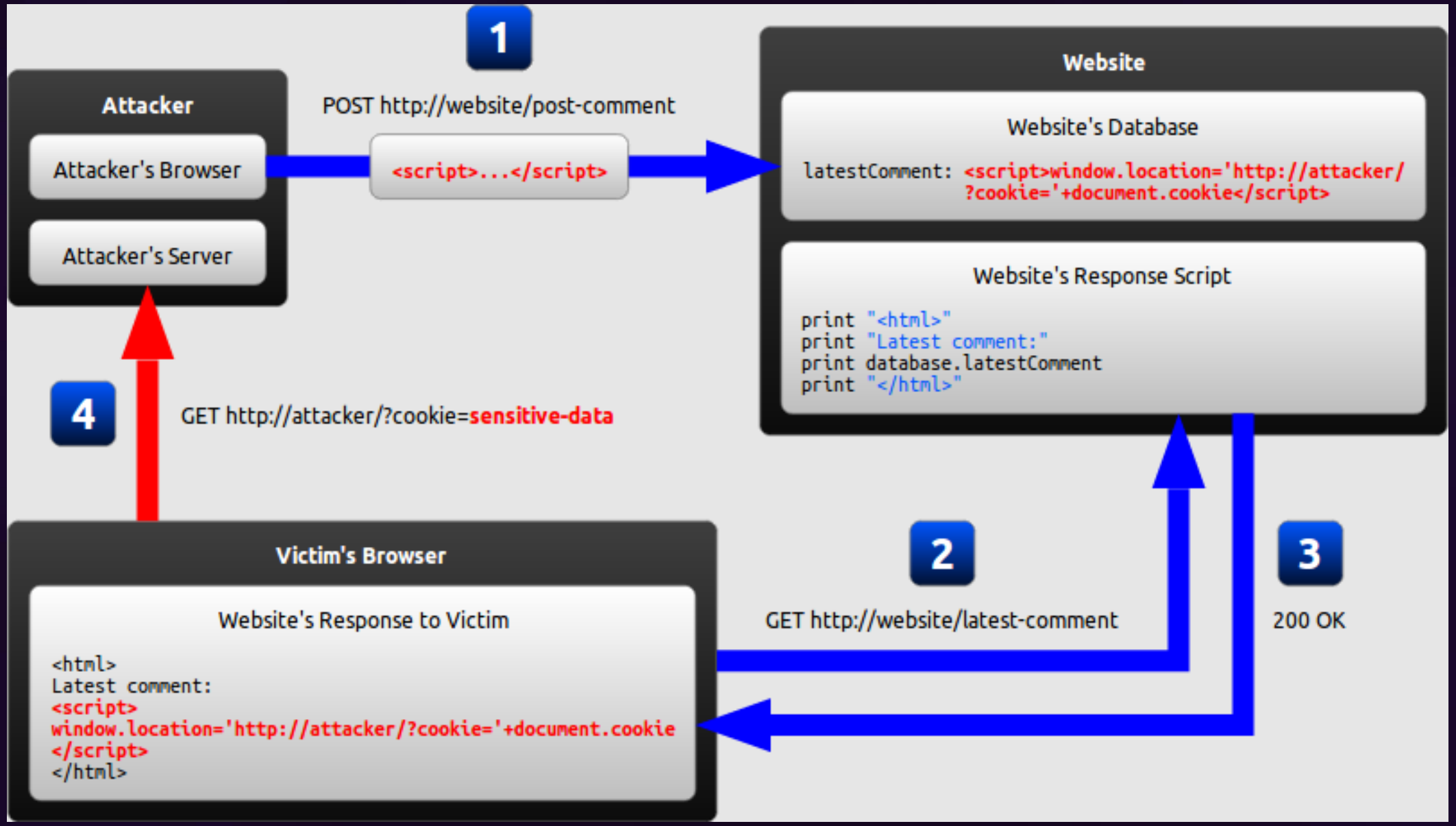
- 前端安全即是用户与浏览器及其中网页进行交互时所产生的安全问题。

常见问题及防范措施

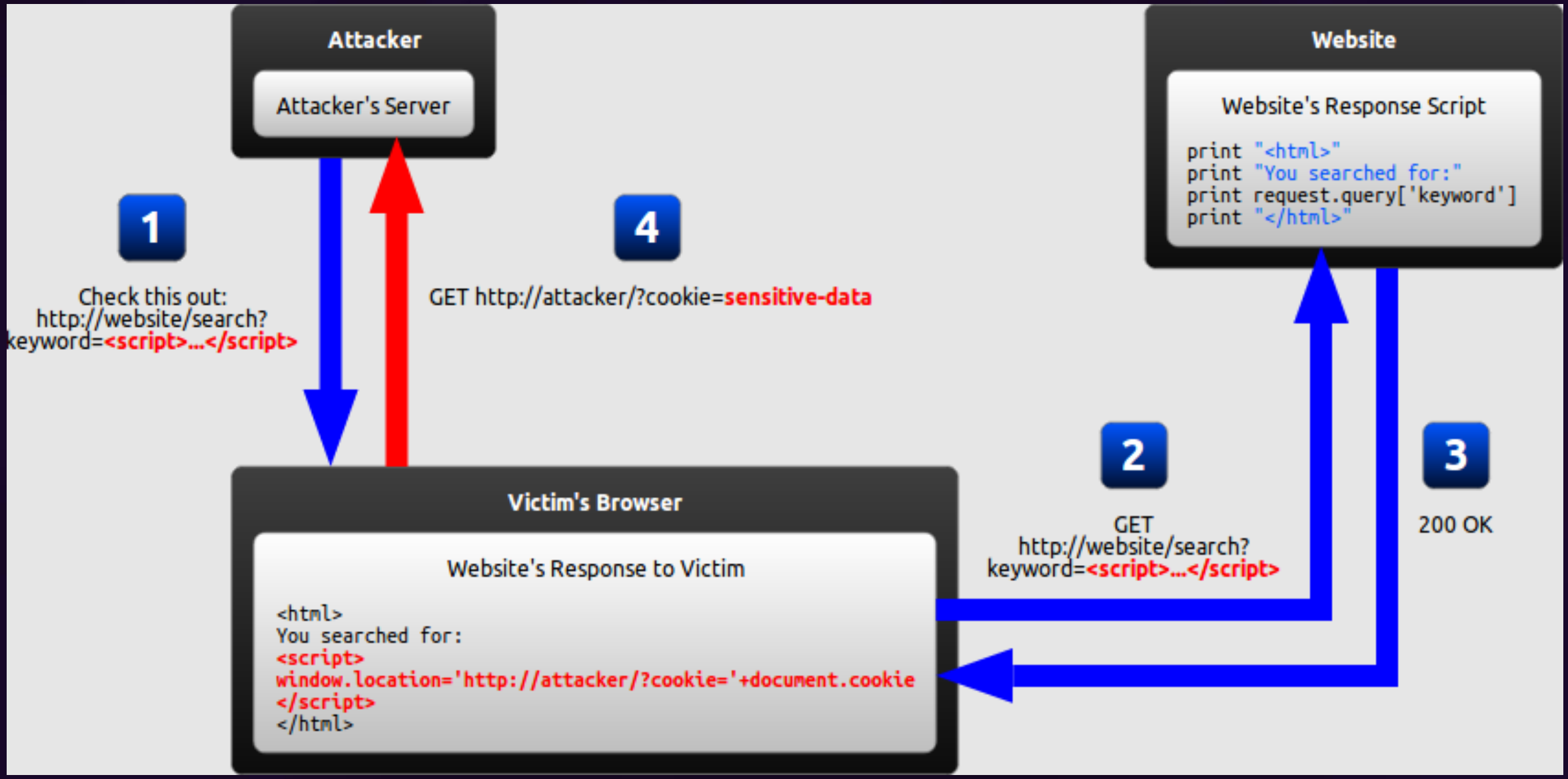
跨站脚本攻击

- 跨站脚本攻击又名XSS（Cross-Site Scripting）。XSS这类安全问题发生的本质原因在于：浏览器错误的将攻击者提供的用户输入数据当做JavaScript脚本给执行了

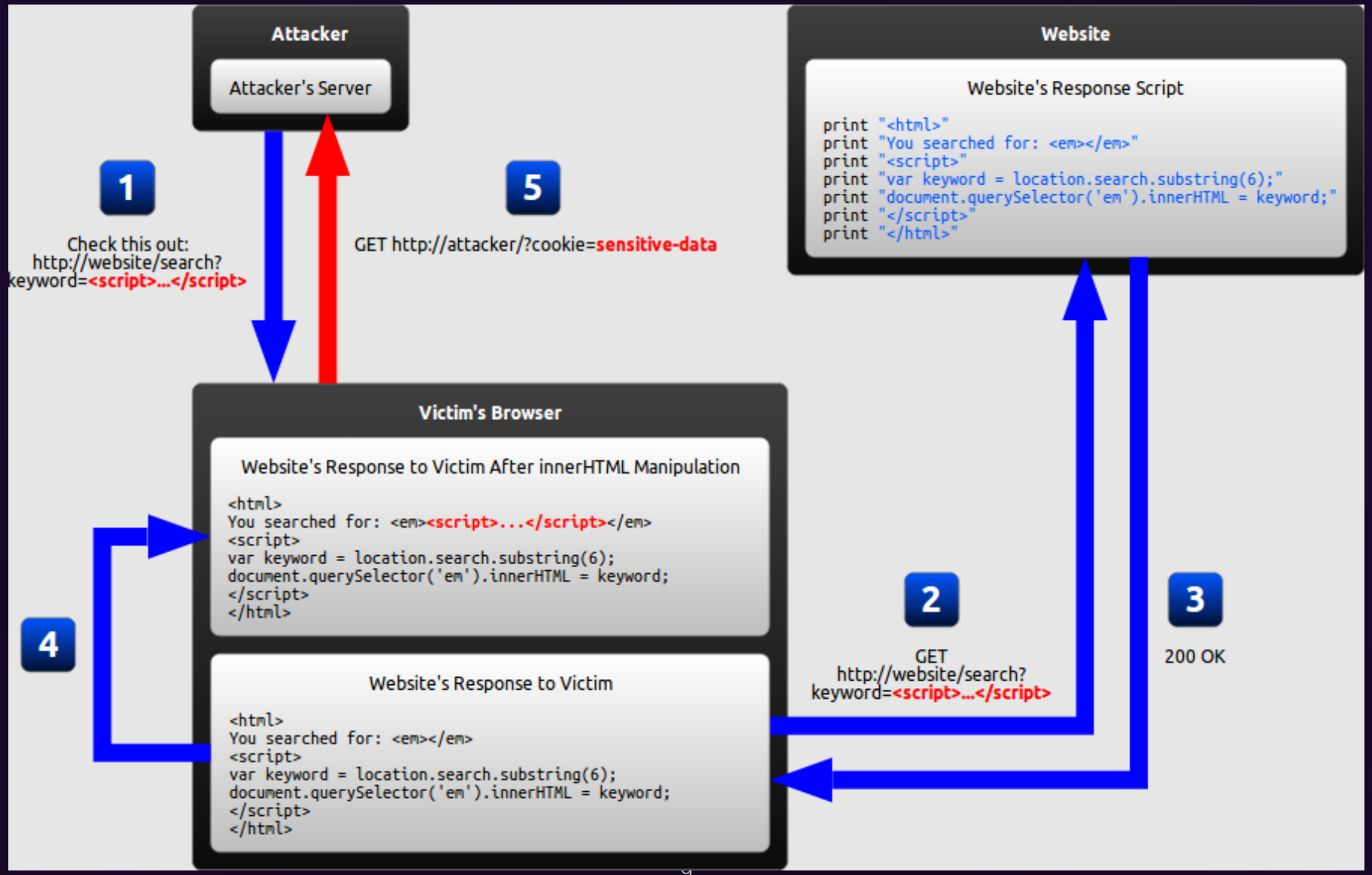
跨站脚本攻击-Persistent XSS



跨站脚本攻击-Reflected XSS



跨站脚本攻击-DOM-based XSS



跨站脚本攻击-如何防御

- 表单提交时按照预期格式进行数据验证
- 服务端接收数据时验证，渲染时转码
- 尽量减少动态改变DOM结构的操作
- 头部设置HttpOnly Cookie
- 使用WAF(Web Application Firewall)

跨站脚本攻击-开发组件

基于现代框架，可以引入xss组件提供全套XSS相关前端API。

在Node.js上使用

安装：

```
$ npm install xss --save
```

使用：

```
var xss = require('xss');
console.log(xss('<a href="#" onclick="alert(/xss/)">click
 me</a>'));
```

在浏览器上使用

引入文件：

```
https://raw.github.com/leizongmin/js-xss/master/dist/xss.j
s
```

使用：

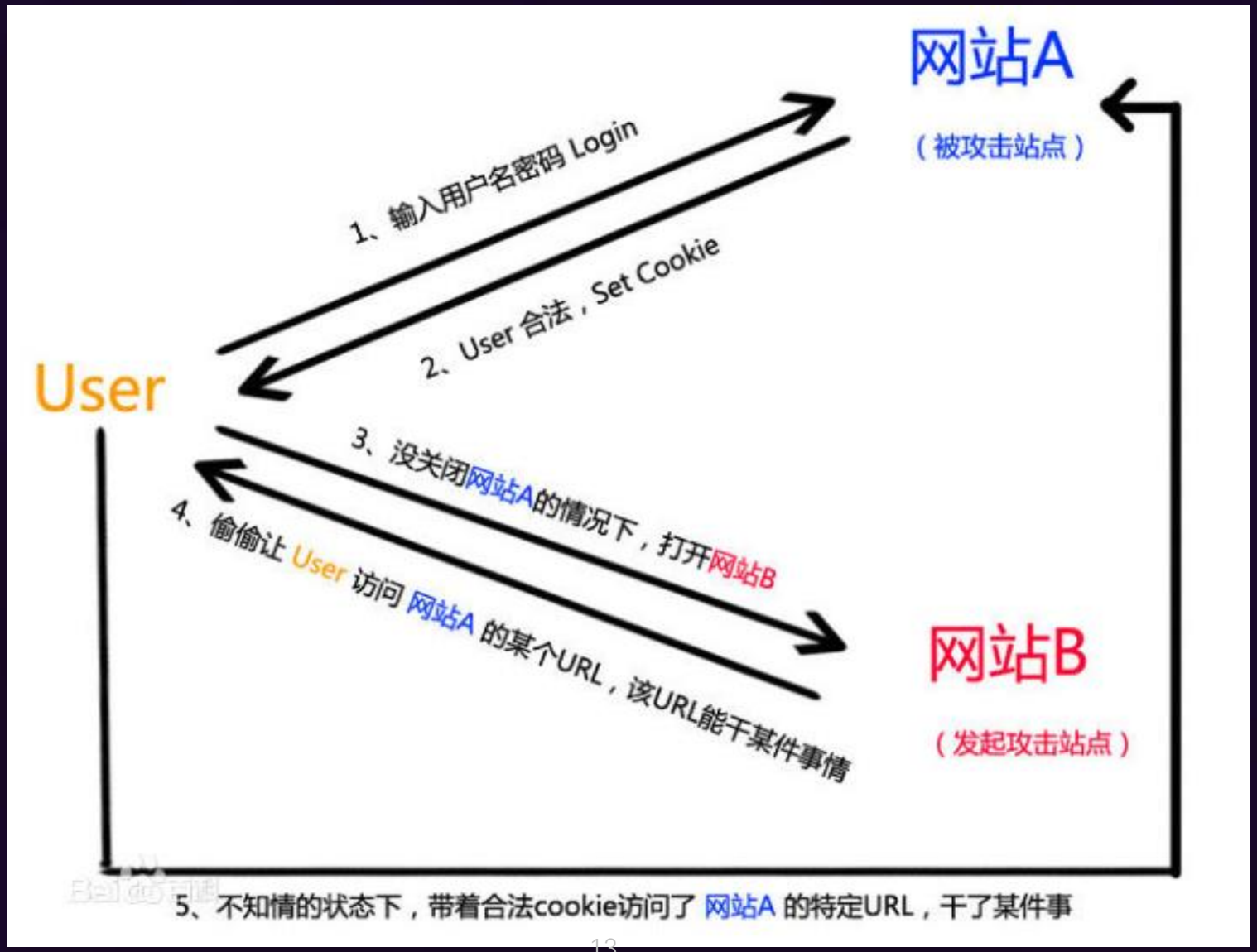
```
console.log(filterXSS('<a href="#" onclick="alert(/xss/)">
click me</a>'));
```

地址： https://jsxss.com/zh/index.html

跨站请求伪造

- 跨站请求伪造又名CSRF(Cross Site Request Forgery)，是一种常见的Web攻击，通过伪装成受信任用户的请求来利用受信任的网站

跨站请求伪造


网站A
（被攻击站点）
1、输入用户名密码 Login
2、User 合法，Set Cookie
User
3、没关闭网站A的情况下，打开网站B
4、偷偷让 User 访问 网站A 的某个URL，该URL能干某件事情
网站B
（发起攻击站点）
5、不知情的状态下，带着合法cookie访问了 网站A 的特定URL，干了某件事

跨站请求伪造-如何防范

- 验证HTTP Referer字段
- 在请求地址中加入token以验证
- 在Http投中自定义属性并验证
- 关键业务增加验证码

文件上传漏洞

- 文件上传漏洞是指用户上传了一个可执行的脚本文件，并通过此脚本文件获得了执行服务器端命令的能力。常见场景是web服务器允许用户上传图片或者普通文本文件保存，而用户绕过上传机制上传恶意代码并执行从而控制服务器。

文件上传漏洞-如何防御

- X-Content-Type-Options: nosniff

- 通过设置该响应标头，对script和styleSheet在执行时是通过MIME类型来过滤掉不安全的文件

iframe安全问题

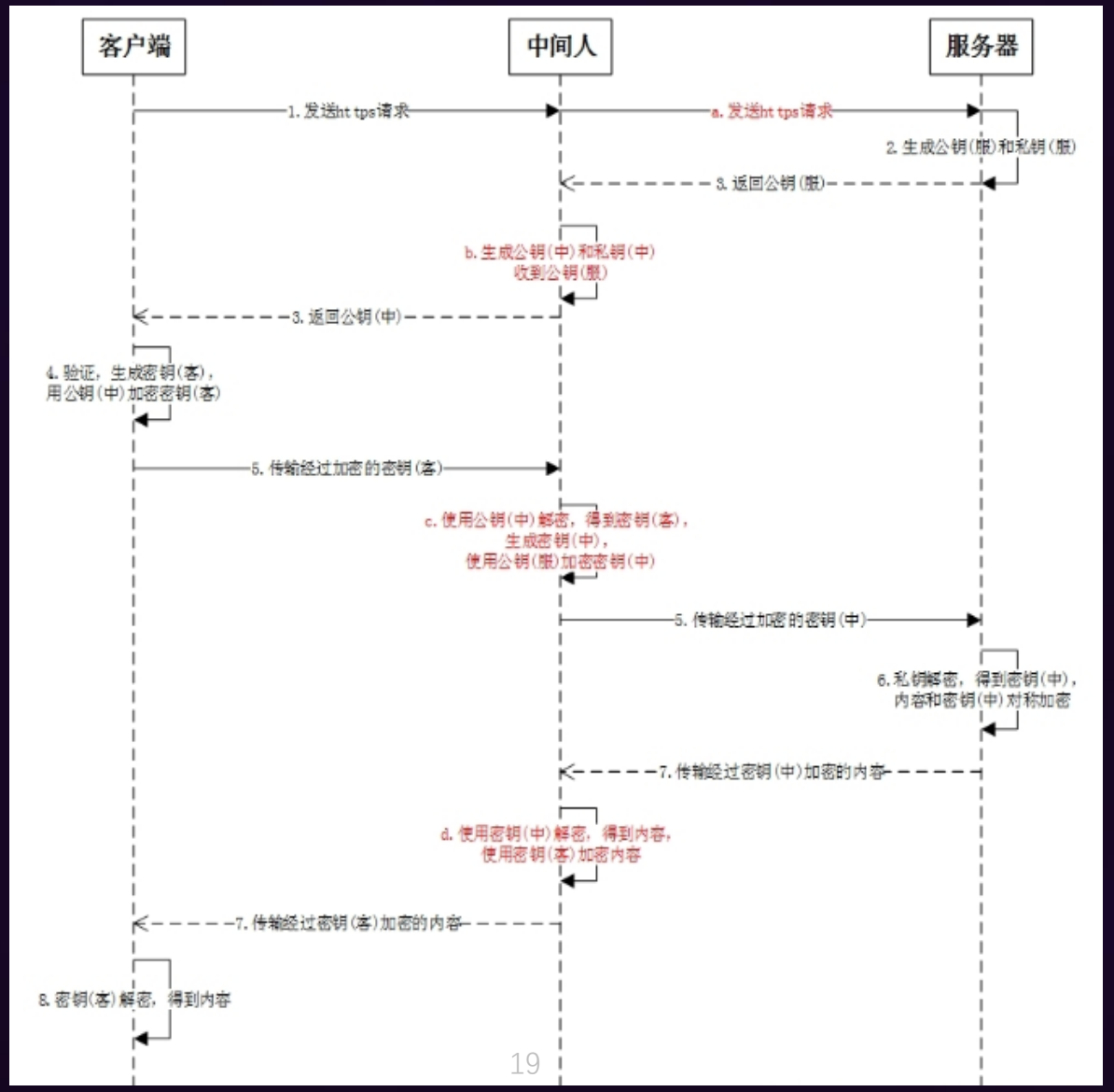
- 甲方安全问题，点击劫持
- 乙方安全问题，第三方网站被攻破或被抢注

iframe安全问题-如何防御

- 增加X-Frame-Options头配置
  - DENY
  - SAMEORIGIN
  - ALLOW-FROM

- 使用iframe的sandbox安全属性限制第三方网站行为
  - allow-forms
  - allow-popups
  - allow-scripts
  - allow-same-origin

SSL中间人攻击


客户端
中间人
服务器
1. 发送ht tps请求
a. 发送ht tps请求
2. 生成公钥(服)和私钥(服)
3. 返回公钥(服)
b. 生成公钥(中)和私钥(中)
收到公钥(服)
3. 返回公钥(中)
4. 验证，生成密钥(客)，
用公钥(中)加密密钥(客)
5. 传输经过加密的密钥(客)
c. 使用公钥(中)解密，得到密钥(客)，
生成密钥(中)，
使用公钥(服)加密密钥(中)
5. 传输经过加密的密钥(中)
6. 私钥解密，得到密钥(中)，
内容和密钥(中)对称加密
7. 传输经过密钥(中)加密的内容
d. 使用密钥(中)解密，得到内容，
使用密钥(客)加密内容
7. 传输经过密钥(客)加密的内容
8. 密钥(客)解密，得到内容

SSL中间人攻击-如何防御

- Strict-Transport-Security: max-age=<expire-time>
- Strict-Transport-Security: max-age=<expire-time>; includeSubDomains
- Strict-Transport-Security: max-age=<expire-time>; preload

- max-age=<expire-time>：设置在浏览器收到这个请求后的<expire-time>秒的时间内凡是访问这个域名下的请求都使用HTTPS请求。
- includeSubDomains：可选，如果这个可选的参数被指定，那么说明此规则也适用于该网站的所有子域名。
- preload：可选，简单来说就是谷歌维护了一个域名列表，凡是注册在列表内的域名都会具有从 http 跳转到 https 的浏览器内部行为。虽然列表由谷歌在维护，但是所有其他浏览器也会尝试使用这个。

内容安全策略

内容安全策略

- 内容安全策略CSP(Content Security Policy) 是一个额外的安全层，用于检测并削弱某些特定类型的攻击，并形成日志通知管理者

- Content-Security-Policy（新版本）
- X-Content-Security-Policy（老版本）

- CSP配置由一到多个策略组成，策略之间用;隔开，每个策略由一个策略指令和源列表组成，用空格分开。源列表是一个字符串，指定了一个或多个互联网主机、协议或端口

内容安全策略-源列表

- http://*.foo.com

  匹配所有使用 http: 协议加载 foo.com 任何子域名的尝试。
- mail.foo.com:443

  匹配所有访问 mail.foo.com 的 443 端口 的尝试。
- https://store.foo.com

  匹配所有使用 https: 协议访问 store.foo.com 的尝试。

- 如果端口号没有被指定，浏览器会使用指定协议的默认端口号。如果协议没有被指定，浏览器会使用访问该文档时的协议。

内容安全策略-常用关键字

有一些关键字可以用来描述某类特别的内容源。它们是：

- 'none' 代表空集；即不匹配任何 URL。两侧单引号是必须的。
- 'self' 代表和文档同源，包括相同的 URL 协议和端口号。两侧单引号是必须的。
- 'unsafe-inline' 允许使用内联资源，如内联的 <script> 元素、javascript: URL、内联的事件处理函数和内联的 <style> 元素。两侧单引号是必须的。
- 'unsafe-eval' 允许使用 eval() 等通过字符串创建代码的方法。两侧单引号是必须的。

内容安全策略-策略指令

- default-src

  default-src 指令定义了那些没有被更精确指令指定的（默认）安全策略。该指令包含了以下指令：

- child-src　指定嵌套的浏览上下文的源（iframe）
- connect-src　　指定各种请求的源(跳转、Ajax、websocket)
- font-src　指定字体文件的源
- img-src　指定被<img>标签加载的源
- media-src　指定被<video><audio>标签加载的源
- object-src　指定被<object><embed><applet>标签加载的源
- script-src　指定脚本文件加载的源
- style-src　指定样式文件加载的源

内容安全策略-样例

- 示例 1,一个网站管理者想要所有内容均来自站点的同一个源 (不包括其子域名)

  Content-Security-Policy: default-src 'self'

- 示例 2,一个网站管理者允许内容来自信任的域名及其子域名 (域名不必须与CSP设置所在的域名相同)

  Content-Security-Policy: default-src 'self' *.trusted.com

- 示例 3,一个网站管理者允许网页应用的用户在他们自己的内容中包含来自任何源的图片,但是限制音频或视频需从信任的资源提供者(获得)，所有脚本必须从特定主机服务器获取可信的代码.

  ```
  Content-Security-Policy:
      default-src 'self';
      img-src *;
      media-src media1.com media2.com;
      script-src userscripts.example.com
  ```

内容安全策略-样例

- 示例 4，一个线上银行网站的管理者想要确保网站的所有内容都要通过SSL方式获取，以避免攻击者窃听用户发出的请求。

```
Content-Security-Policy:
    default-src https://onlinebanking.jumbobank.com
```

该服务器仅允许通过HTTPS方式并仅从onlinebanking.jumbobank.com域名来访问文档。

- 示例 5，一个在线邮箱的管理者想要允许在邮件里包含HTML，同样图片允许从任何地方加载，但不允许JavaScript或者其他潜在的危险内容(从任意位置加载)。

```
Content-Security-Policy:
    default-src 'self' *.mailsite.com;
    img-src *
```

注意这个示例并未指定script-src。在此CSP示例中，站点通过 default-src 指令的对其进行配置，这也同样意味着脚本文件仅允许从原始服务器获取。

THANKS

前端赋能 共创卓越