大家好，在前面两个小节中，我们对网络协议相关的知识点做了简单的介绍。本小节是网络协议篇的最后一节，主要介绍和Web开发相关的安全漏洞，包括XSS跨站脚本攻击，CSRF跨站请求伪造，SSRF服务端请求伪造以及SQL注入漏洞等。 3

XSS，CSRF以及SSRF，SQL注入都是Web开发中最为常见的攻击手段，日常开发中，我们必须采取有效的防御措施。所以，让我们来一起看看常见Web安全漏洞的相关技术原理吧~

（1）XSS跨站脚本攻击：

答：XSS （Cross-Site Scripting）跨站脚本攻击是一种常见的安全漏洞，恶意攻击者在用户提交的数据中加入一些代码，将代码嵌入到了Web页面中，从而可以盗取用户资料，控制用户行为或者破坏页面结构和样式等。为了和 CSS 区分，这里把攻击的第一个字母改成了 X，于是叫做 XSS。

最简单的就是当我们提交一个查询后弹出一个alert页面，却无论如何都关不掉，这就是发生了XSS跨站脚本攻击。

XSS产生原因：

XSS产生的原因是过于信任客户端的数据，没有做好过滤或者转义等工作。如果客户端上传的数据中插入一些符号以及javascript代码，那么这些数据将会成为应用代码中的一部分了，这样就造成了XSS攻击。

XSS分类：

- **存储型：**攻击者将恶意代码存储到了数据库中，在响应浏览器请求的时候返回恶意代码，并且执行。这种攻击常见于带有用户保存数据的网站功能；
- **反射型：**将恶意代码放在URL中，将参数提交到服务器。服务器解析后响应，在响应结果中存在XSS代码，最终通过浏览器解析执行；
- **DOM型：**取出和执行恶意代码由浏览器端完成，属于前端 JavaScript的安全漏洞。

XSS防御：

- 对重要的 cookie设置 httpOnly, 防止客户端通过document.cookie读取 cookie；
- 对输入内容的特定字符进行编码，前端后端都可以对传入的内容进行过滤，去掉带javascript等字段的输入

（2）CSRF跨站请求伪造：

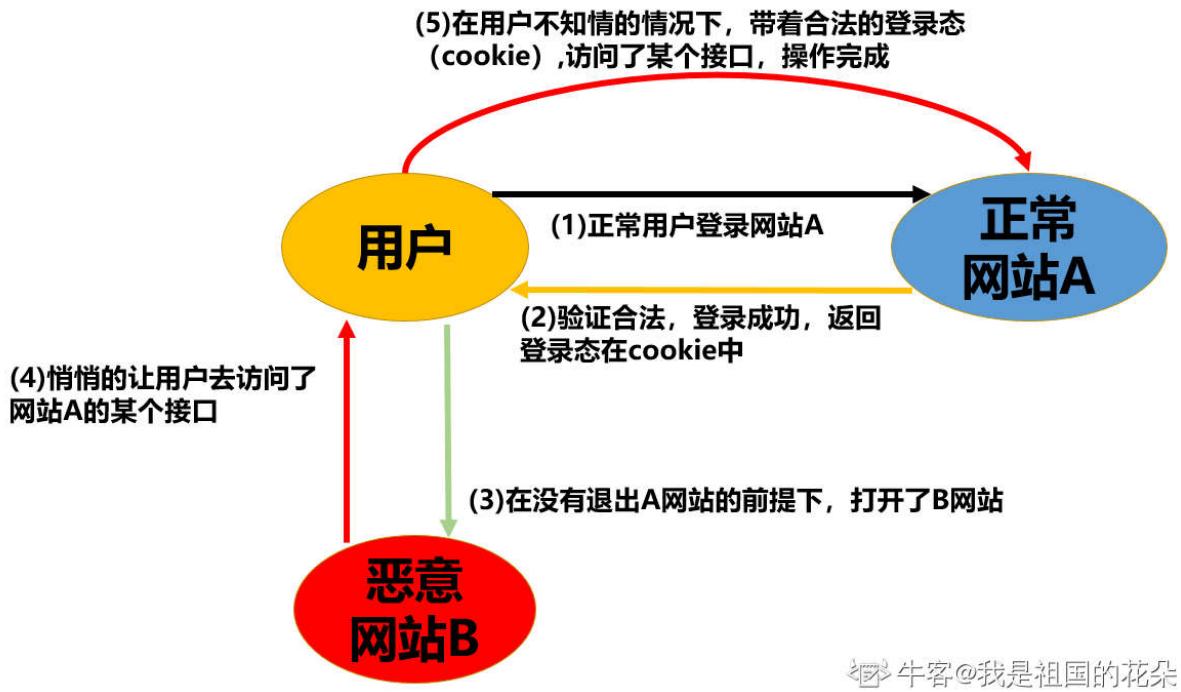
答：CSRF（Cross-site request forgery）跨站请求伪造，也是一种常见的安全漏洞。XSS相当于是控制了站点内的信任用户，而CSRF则通过**伪装**成受信任用户的请求来利用受信任的网站。

3

CSRF举例：

用户登录受信任网站A，并在本地生成登录态Cookie。（如果用户没有登录网站A，那么网站B在获取A网站的信息并且去请求网站A的接口时，会提示登录）**在不登出A的情况下**，访问恶意网站B，那么网站B得到了网站A的所有信息，然后B网站去请求A网站的接口，伪装成A网站的正常请求为所欲为。

下边以示意图来说明**CSRF整个流程**：



注意：

CSRF中恶意网站仅仅是伪装成了正常用户，但是其并不可以直接获取到正常用户的登录态cookie等信息。如果不做防御，被攻击网站服务器是无法区分是否是冒用用户，因为当前请求确实带着登录凭证等信息。

CSRF防御：

- **Referer 头验证**：在 HTTP 头中有一个字段叫 Referer，它记录了该 HTTP 请求的来源地址。不靠谱，Referer可以被改变；
- **Token验证**：服务器发送给客户端一个Token，客户端提交的表单中（或者URL上）带着这个Token。如果这个Token 不合法，那么服务器拒绝这个请求。
- **双重Cookie验证**：利用恶意网站无法获取cookie信息，仅可冒用的特点，我们将cookie中的参数取出来，加入到请求参数中，服务端进行校验，如果参数中没有附加额外的cookie中的参数，那么就拒绝请求。

解析：

XSS和CSRF均属于安全漏洞，和我们的开发工作息息相关。对于一些对安全性有一定要求的方向和岗位，了解常见的XSS和CSRF攻击无疑是面试的一大加分项。

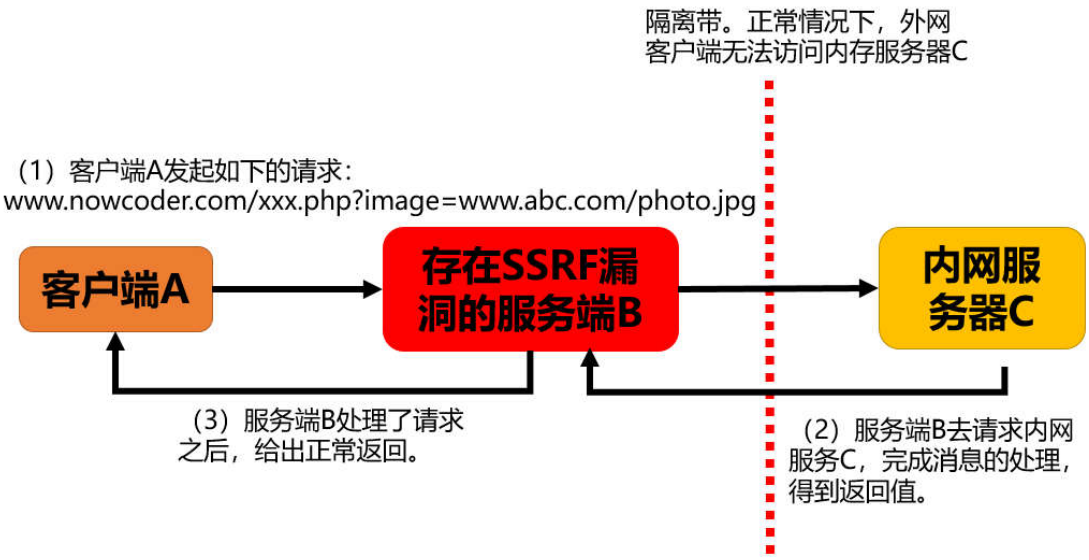
3

那么接下来，我们看看**CSRF和XSS的区别有哪些呢**？

- CSRF攻击需要用户先登录网站A，恶意网站B获取到A网站用户的 cookie；
- XSS攻击则不需要登录。
- CSRF攻击本质是利用网站A本身的漏洞，去请求网站A的相关接口；
- XSS攻击向网站 A 注入恶意代码，然后通过执行恶意代码，篡改了网站A的内容。

（3）SSRF服务端请求伪造：

答：SSRF是一种由攻击者构造请求，利用服务端发起的一种安全漏洞。一般情况下，SSRF攻击的目标是**外网无法访问的内部系统。**我们先来看下SSRF攻击的示意图：



SSRF服务端请求伪造示意图：

SSRF漏洞举例：

- 正常的网络请求流程：客户端A发起请求 -> 服务端B接收请求 -> 服务端B处理请求 -> 服务端B返回响应
- 存在SSRF漏洞下的网络请求流程：

比如现在客户端A发起的请求是这样的 www.nowcoder.com/xxx.php?image=www.abc.com/photo.jpg。 服务端B收到该请求后，会接着取访问www.abc.com/photo.jpg获取资源文件。如果服务端B对客户端发起的请求没有进行过滤等操作，那么?image=可能会被恶意篡改。最后的结果就是，**借助于公网上的服务器来访问了内网系统。**

SSRF产生原因：

SSRF 形成的原因大都是由于服务端提供了从其他服务器应用获取数据的功能，且没有对目标地址做过滤与限制。比如指定URL地址获取网页文本内容，加载指定地址的图片和文档等。

3

常见SSRF漏洞出现场景：

- 分享场景，通过URL地址分享网页内容。
- 转码服务，在线翻译场景。
- 地址加载或下载图片。
- 图片、文章收藏功能。
- 未公开的api实现以及其他调用URL的功能等。

SSRF漏洞危害：

因为外网借助了服务端来实现了对内网服务器的访问，所以很多操作都可以进行，包括如下的危害：

- 对服务器所在的内网进行端口扫描，获取一些服务的banner信息等。
- 攻击运行在内网或者本地的应用程序。
- 对内网WEB应用进行指纹识别，通过访问默认文件实现。
- 下载内网的一些资源文件等。

SSRF的防御措施：

- 对**错误信息进行统一处理**，避免用户可以根据错误信息来判断远端服务器的端口状态。
- 对请求的**端口进行限制**，限定为HTTP常用的端口，比如，80，443和8080等。
- 设定**IP黑名单**。避免应用被用来获取内网数据，攻击内网。
- **禁用不需要的协议**。仅仅允许HTTP和HTTPS请求。
- 对返回信息进行**有效过滤**等。

（4）SQL注入：

答：SQL注入是指通过把SQL命令插入到Web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器，执行恶意的SQL命令。

SQL注入就是服务端将客户端传入的恶意SQL语句直接进行了执行，这样会导致问题出现。比如说用户在登录的时候，使用了or 1=1来完成身份验证和授权。

SQL注入是一种流行的攻击攻击方法，但是通过一些手段是可以防御该攻击的，**常见的防御手段如下：**

- 使用预编译语句，比如MyBatis中的SQL语句使用#号代替$符号。

- 使用安全的存储过程来防止SQL注入。
- 对客户端的输入进行数据类型的检查等。

解析：

3

做为一名优秀的服务端开发人员，我们应该牢记一条原则，“**永远不要相信客户端**”。对客户端的每次请求，我们都要做好充分的过滤，验证与授权，这样才可以尽可能的避免常见的Web安全漏洞，抵御来自外部世界的攻击。

在网络协议篇的最后，我给大家留一个课后习题吧。

课后习题：（重点掌握）

从浏览器中输入一个网址，比如：https://www.nowcoder.com/ 之后都发生了啥？请尽可能详细的阐述。

讨论：

这是一个面试官经常问的问题，也是一个相当经典的问题。其中涉及到了域名解析，三次握手，HTTP请求，路由转发，服务器处理请求返回文件，浏览器渲染显示页面等诸多知识。

大家可以将回答评论在下方，我会在下一小节中给出大家参考答案~

总结：

在包括本小节的网络协议篇章中，我们较为详细的学习交流了面试中常见的网络协议知识点，由于网络协议知识面广阔，入门学习门槛较高，所以我们就挑高频面试题目进行了交流。大家可以**重点掌握TCP的三次握手，四次挥手状态转换，TCP与UDP的区别等**。在一些业务逻辑涉及到安全的Java岗位面试中，熟悉常见的Web安全漏洞也会是一个很好的面试加分项，希望大家可以掌握。下一小节开始，我们将进入MySQL数据库的学习环节。

限于作者水平，文章中难免会有不妥之处。大家在学习过程中遇到我没有表达清楚或者表述有误的地方，欢迎随时在文章下边指出，我会及时关注，随时改正。另外，大家有任何话题都可以在下边留言，我们一起交流探讨。

讨论 评论

刘畅201904211606816 1#

打卡

发表于 2020-01-08 21:59:22 赞(0) 回复(0)

求亿份offer

打卡 2#

发表于 2020-04-10 11:42:26 赞(0) 回复(0)

Shawn_Liu 3#

3

1.

```
'''
DNS解析：将域名解析成对应的服务器IP地址。
TCP连接：拿到域名对应的IP地址之后，浏览器会以一个随机端口（1024<端口

发送HTTP请求：建立了TCP连接之后，发起一个http请求。一个典型的 http r
客户端向服务器发起http请求的时候，会有一些请求信息，请求信息包含三个

请求方法URI协议/版本
请求头(Request Header)
请求正文：

服务器处理请求并返回HTTP报文：后端从在固定的端口接收到TCP报文开始，它
HTTP报文；就是返回一个HTPP响应。 HTTP响应与HTTP请求相似，HTTP响应也

状态行
响应头(Response Header)
空行
响应正文

浏览器解析渲染页面：WebKit渲染的过程：构建dom树 -> 构建render树 ->

连接结束：现在的页面为了优化请求的耗时，默认都会开启持久连接（keep-a

'''
```

发表于 2020-05-09 10:30:01 赞(0) 回复(0)