

# CSRF

---

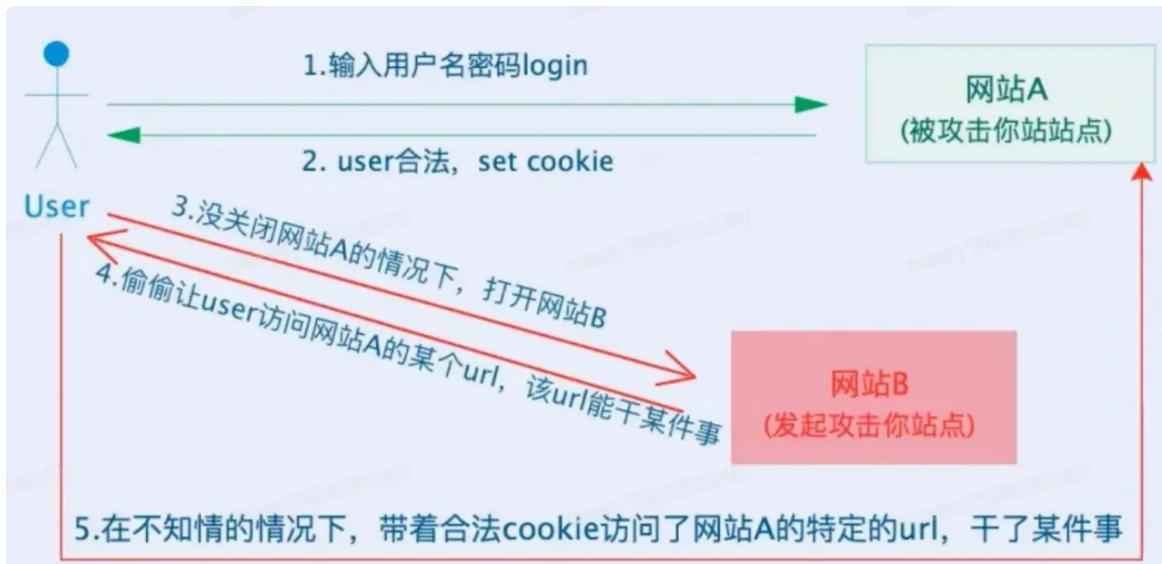
- [1. 原理](#)
- [2. 与XSS区别](#)
- [3. 常见场景](#)
- [4. 常见漏洞点](#)
- [5. 漏洞危害](#)
- [6. CSRF Poc 构造](#)
- [7. 漏洞审计](#)
- [8. 漏洞修复](#)
- [9. Webgoat](#)

## 1. 原理

跨站请求伪造 (Cross-site request forgery) CSRF，是一种使已登录用户在不知情的情况下执行某种动作的攻击，**利用用户已登录身份，伪造请求操作**。

实际上这种方式是攻击者通过一些钓鱼等手段欺骗用户去访问一个自己曾经认证过的网站，然后执行一些操作（如后台管理、发消息、添加关注甚至是转账等行为）。浏览器曾经认证过，因此被访问的网站会认为是真正的用户操作而去运行。**CSRF 漏洞的工作原理是攻击者盗用了用户的身份，以用户的名义发送恶意请求**。因为攻击者看不到伪造请求的响应结果，所以CSRF攻击主要用来执行动作，而非窃取用户数据。

当受害者是一个普通用户时，CSRF可以实现在其不知情的情况下转移用户资金、发送邮件等操作；但是如果受害者是一个具有管理员权限的用户时CSRF则可能威胁到整个Web系统的安全。



CSRF 漏洞的攻击原理

一次完整的 CSRF 攻击需要具备以下两个条件：

- 用户已经登录某站点，并且在浏览器中存储了登录后的 Cookie 信息。
- 在不注销某站点的情况下，去访问攻击者构造的站点。

例：

网站管理员添加用户的 url 链接是“add?name=admin1&password=admin1”，如果在添加用户处有 CSRF 漏洞，那么当攻击者把“add?name=hack&password=hack”这个链接发给网站管理员，而管理员点击了这个链接，那么就会在管理员并不知情的情况下给网站创建了“hack”账户。

## 2. 与XSS区别

从信任的角度来区分

XSS：利用用户对站点的信任

CSRF：利用站点对已经身份认证的信任

## 3. 常见场景

CSRF 攻击可能出现的场景有很多，如更改个人信息、修改密码、添加/修改资料、关注/取关用户、发布主题或信息、与交易相关的操作等。

1. CSRF 基于 post 请求添加账号
2. CSRF 修改密码

## 4. 常见漏洞点

检查是否校验 Referer、敏感操作是否会生成 CSRF token，如果都不存在再查看请求参数中是否存在不可被攻击者猜测的字段，比如验证码等参数。

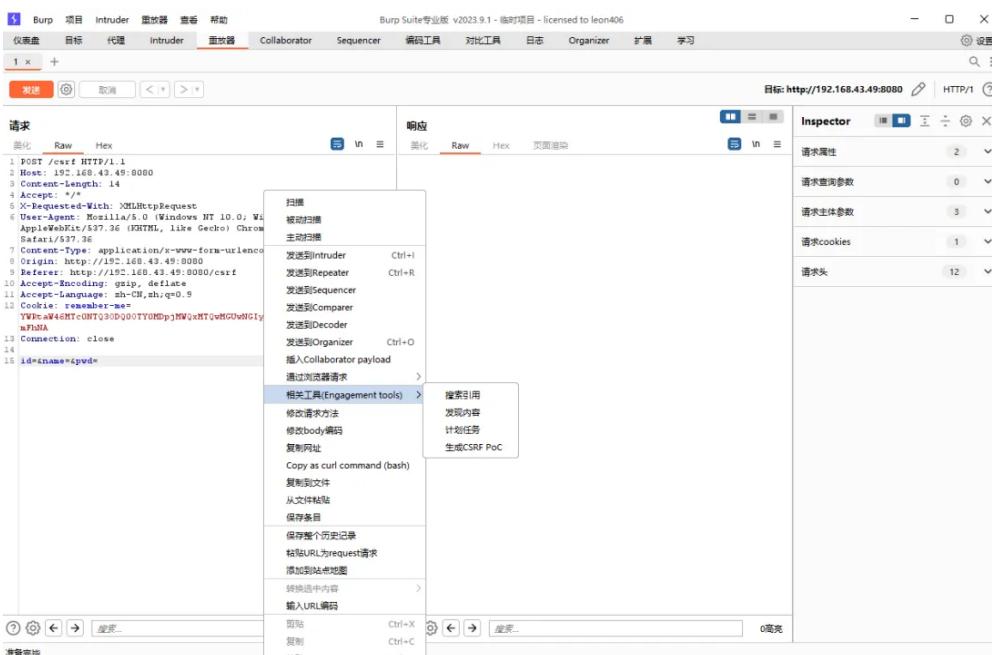
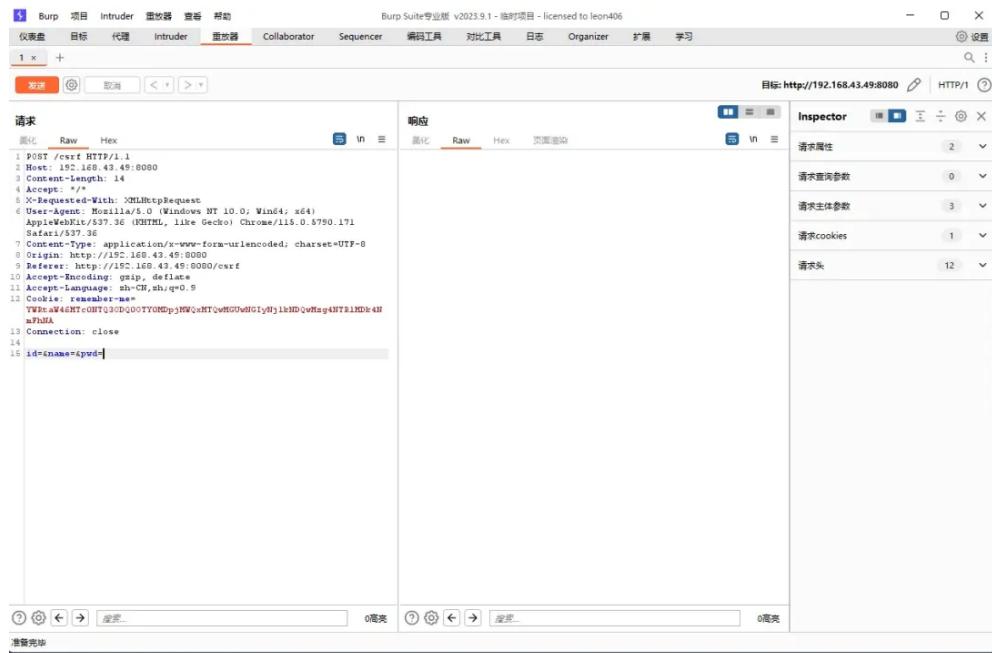
Referer记录这个请求来源于哪个网址，也就是请求的上一个网址是什么。通过Referer就可以去判断这个请求是用户授权的还是伪造的。所以可以对Referer进行限制。

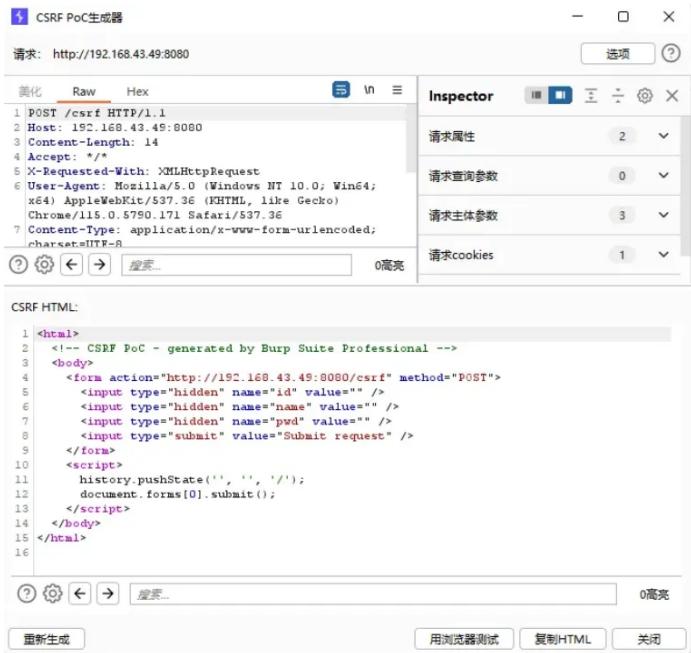
CSRF攻击无法伪造正确的Token值，从而阻断恶意请求。

## 5. 漏洞危害

越权操作	攻击者可借助受害者的登录态执行操作，如修改密码、提交表单、转账等。
账户被篡改	修改邮箱、手机号、密码等敏感信息，导致账号被劫持。
数据泄露或破坏	若操作涉及数据访问或提交，攻击者可能借此窃取或污染数据。
伪造交易或操作请求	可用于实施钓鱼或诈骗，例如伪造支付、点赞、关注、发帖等行为。
提升攻击隐蔽性	攻击过程无需用户交互，用户完全不知情。与XSS相比更加隐蔽。
扩大攻击面	搭配社交工程，攻击者可将钓鱼链接发送至社交媒体或邮件，扩大传播范围。

## 6. CSRF Poc 构造





## 7. 漏洞审计

- 未对数据来源进行校验：后台修改密码的代码存在 CSRF 攻击风险。攻击者可在其服务器上创建钓鱼页面，其中包含构造的发送修改用户密码请求的代码，当用户在已登录网站的情况下点击攻击者发送的钓鱼链接时，密码将被修改。

▼ 没有手动的对数据的来源进行校验。 Java

```

1  @RequestMapping("/doUpdate")
2  public String doUpdate(HttpServletRequest request, Users user, Model model){
3      HttpSession session = request.getSession();
4      user.setUsername((String)session.getAttribute("user"));
5      System.out.println(session.getAttribute("user"));
6      usersService.update(user); // 更新用户信息
7      model.addAttribute("success", "修改成功");
8      return "sqli/update";
9  }

```

- 对 Referer 过滤不严导致的 CSRF 漏洞

```

public class RefererInterceptor extends HandlerInterceptorAdapter {
    private Boolean check = true;
    @Override
    public boolean preHandle(HttpServletRequest req,
    HttpServletResponse resp, Object handler) throws Exception {
        if (!check) {
            return true;
        }
        String referer=request.getHeader("Referer");
        if((referer!=null) &&(referer.trim().startsWith("www.testdomain.com"))){
            chain.doFilter(request, response);
        }else{
            request.getRequestDispatcher("index.jsp").forward(request, response);
        }
    }
}

```

www.testdomain.com.hacker.com

对 Referer 过滤不严导致 CSRF 漏洞的核心代码。逻辑判断的关键点在于第二步，仅判断请求 Referer 字段是否以 www.testdomain.com 开头，若构建一个二级域名为 www.testdomain.com.hacker.com 的地址，则可能成功绕过该判断，从而进行 CSRF 攻击。

该段源码对于 CSRF 漏洞的防御流程如下：

- 从用户的请求头中取得 Referer 值，判断其是否为空。
- 若为空，则跳转至首页；若不为空，则进行下一步判断。
- 判断 Referer 是否以 www.testdomain.com 开头，若不是，则跳转至首页；若是，则执行该操作请求。

### 3. token 可重用导致 CSRF 漏洞

```

String sToken = generateToken();
String pToken = req.getParameter("csrf-token");
if(sToken != null && pToken != null
&& sToken.equals(pToken)){
    chain.doFilter(request, response);
} else{
    request.getRequestDispatcher(index.jsp").forward(request, response);
}
}

```

某个源程序中判断 token 是否可用的核心代码。源码在用户成功登录后，生成了唯一的令牌，直至该用户注销前，该 token 都是有效的，如果这个 token 被盗用或者泄露，那么就可能导致 CSRF 漏洞的发生。

## 8. 漏洞修复

**STP ( Synchronizer Token Pattern, 令牌同步模式)**：当用户发送请求时，服务器端应用将 token 嵌入 HTML 表格中，并发送给客户端。客户端提交HTML 表格，会将令牌发送到服务端，令牌的验证是由服务端实行的。令牌可以通过任何方式生成，只要确保其随机性和唯一性。这样就能够确保攻击者发送请求的时候，由于没有该令牌而无法通过验证。

**检查 Referer 字段。** HTTP 头中有一个 Referer 字段，这个字段用以标明请求来源于哪个地址。在处理敏感数据请求时，一般情况下，Referer 字段应该与请求地址位于同一域名下。而如果是 CSRF 攻击传递来的请求，Referer 字段会是包含恶意攻击载荷的地址（如站点 B），通过这种判断能够识别出 CSRF 攻击。这种防御手段的关键点在于如何建立合适的白名单校验机制。

**添加校验 token。** CSRF 的本质是攻击者通过欺骗用户去访问自己设置的地址，所以如果在所有用户进行敏感操作时，要求用户浏览器提供未保存在 Cookie 中且攻击者无法伪造的数据作为校验，那么攻击者就无法再进行 CSRF 攻击。这种方式通常是在请求时增加一个加密的字符串 token，当客户端提交请求时，这个字符串 token 也被一并提交上去以供校验。当用户进行正常的访问时，客户端的浏览器能够正确得到并传回这个字符串 token。而通过 CSRF 攻击的方式，攻击者无法事先获取到该 token 值，服务端就会因为校验 token 的值为空或者错误，拒绝这个可疑请求，从而达到防范 CSRF 攻击的目的。

除以上 3 种主流方式外，还有很多其他方式，比如验证码机制、自定义 http 请求头方式、Origin 字段等，但是这些方法都存在各自的问题，如友好度差、存在机制绕过的可能等，因而只是作为辅助防御方式使用。

## 9. Webgoat

1. 进入到 Webgoat CSRF 页面
2. 是一个发表评论的功能，还是先发表正常评论，然后/抓包找到处理接口：“<http://127.0.0.1:8080/WebGoat/csrf/review>”：
3. 在 idea 里搜索关键词“review”，找到接口代码：

在文件中查找 在 5 个文件中有 8 个匹配项

文件掩码(A): \*.java

Q /review

在项目(P) 模块(M) 目录(D) 作用域(S)

uploadTrickHtml("csrf4.html", trickHTML4.replace("WEBGOATURL", webGoatCSRFIntegrationTest.java 72  
 path = "/csrf/review", ForgedReviews.java 54  
 @PostMapping("/csrf/review") ForgedReviews.java 70  
 The page below simulates a comment/review page. The difference here is that you CSRF\_Reviews.adoc 3  
 <link rel="stylesheet" type="text/css" th:href="@{/lesson\_css/reviews.css}"/> CSRF.html 60

CSRFIntegrationTest.java src/main/java/org/owasp/webgoat/integration

```

69     : "CSRF");
70     webWolfFileServerLocation();
71     fileName: "csrf3.html", trickHTML3.replace( target: "WEBGOATURL", webGoatUrlConfig.url());
72     fileName: "csrf4.html", trickHTML4.replace( target: "WEBGOATURL", webGoatUrlConfig.url());
73     fileName: "csrf7.html", trickHTML7.replace( target: "WEBGOATURL", webGoatUrlConfig.url());
74
75     html",
76     e( target: "WEBGOATURL", webGoatUrlConfig.url( path: "login")).replace( target: "USE
77
78
79
    
```

在新标签打开(B) 打开查找窗口

4. 在 org.owasp.webgoat.csrf.ForgedReviews 类的 createNewReview 的方法:

```

70     @PostMapping(path = "/csrf/review")
71     @ResponseBody
72     public AttackResult createNewReview(
73         String reviewText,
74         Integer stars,
75         String validateReq,
76         HttpServletRequest request,
77         @CurrentUsername String username) {
78     final String host = (request.getHeader("host") == null) ? "NULL" : request.getHeader("host");
79     final String referer =
80         (request.getHeader("referer") == null) ? "NULL" : request.getHeader("referer");
81     final String[] refererArr = referer.split(regex: "/");
82
83     Review review = new Review();
84     review.setText(reviewText);
85     review.setDateTime(LocalDateTime.now().format(fmt));
86     review.setUser(username);
87     review.setStars(stars);
    
```

5. 整段代码都是正常的，接收参数，然后存入数据库。但是缺了一个步骤，就是对 CSRF 的防御。没有防御 CSRF 的 token 校验或者 refer 判断，就直接执行了该功能。

```

70     @PostMapping(value="/csrf/review")
71     @ResponseBody
72     public AttackResult createNewReview(
73         String reviewText,
74         Integer stars,
75         String validateReq,
76         HttpServletRequest request,
77         @CurrentUser String username) {
78     final String host = (request.getHeader("host") == null) ? "NULL" : request.getHeader("host");
79     final String referer =
80         (request.getHeader("referer") == null) ? "NULL" : request.getHeader("referer");
81     final String[] refererArr = referer.split("/");
82
83     Review review = new Review();
84     review.setText(reviewText);
85     review.setDateTime(LocalDateTime.now().format(fmt));
86     review.setUser(username);
87     review.setStars(stars);
88     var reviews = userReviews.getOrDefault(username, new ArrayList<>());
89     reviews.add(review);
90     userReviews.put(username, reviews);
91     // short-circuit
92     if (validateReq == null || !validateReq.equals(weakAntiCSRF)) {
93         return failed(assignment: this).feedback(resourceBundleKey: "csrf-you-forgot-something").build();
94     }
95     // we have the spoofed files
96     if (referer != "NULL" && refererArr[2].equals(host)) {
97         return failed(assignment: this).feedback(resourceBundleKey: "csrf-same-host").build();
98     } else {
99         return success(assignment: this)
100             .feedback(resourceBundleKey: "csrf-review.success")
101             .build(); // feedback("xss-stored-comment-failure")
102     }
103 }
104 }
```

6. 知道了代码层面的缺陷后，接下里就是利用了，回到漏洞页面，右键“查看元素”查看提交评论的表单：

The screenshot shows a browser window with a sidebar containing various tabs like '控制台' (Console), '调试器' (Debugger), and '样式编辑器' (Style Editor). The main content area displays a web page with a header 'I REQUEST YOUR ASSISTANCE'. Below it is a form with two input fields containing 'test' and a 'Submit review' button. A list of reviews follows, each with a small profile picture, name, stars, date, and a short message. The bottom of the page has a footer with a 'postReview' button.

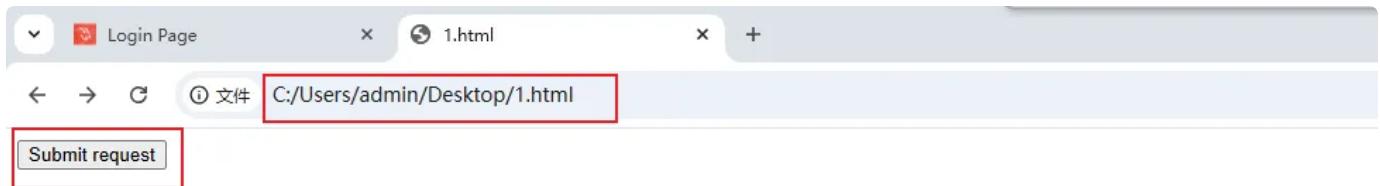
## 7. 根据表单，构造利用代码：

```

1 <html>
2   <body>
3     <script>history.pushState(' ', ' ', '/')</script>
4   <form action="http://127.0.0.1:8080/WebGoat/csrf/review" method="POST"
5     <input type="hidden" name="reviewText" value="hacked by hack3r" />
6     <input type="hidden" name="stars" value="666" />
7     <input type="hidden" name="validateReq" value="2aa14227b9a13d0bede0388
8       a7fba9aa9" />
9     <input type="submit" value="Submit request" />
10   </body>
11 </html>

```

## 8. 先退出账户，打开该恶意 html，点击按钮，发现跳转到登录页面。



9. 再登录 Webgoat 账户，在点开恶意页面，并点击，发现执行成功了。

Introduction	>
General	>
(A1) Broken Access Control	>
(A2) Cryptographic Failures	>
(A3) Injection	>
(A5) Security Misconfiguration	>
Cross-Site Request Forgery	
XXE	
(A6) Vuln & Outdated Components	>
(A7) Identity & Auth Failure	>
(A8) Software & Data Integrity	>
(A9) Security Logging Failures	>
(A10) Server-side Request Forgery	>
Client side	>
Challenges	>

Show hints Reset lesson

← 1 2 3 4 5 6 7 8 9 +

## Post a review on someone else's behalf

The page below simulates a comment/review page. The difference here is that you have to initiate the submission elsewhere as you might see in a classic example is account/wire transfers in someone's bank account.

But we're keeping it simple here. In this case, you just need to trigger a review submission on behalf of the currently logged in user.



John Doe is selling this poster, read reviews below.

24 days ago



Add a Review

Submit review



test1234 / 1 stars 2025-04-08, 17:43:04

1



test1234 / 666 stars 2025-04-08, 17:44:33  
hacked by hack3r