Cross-Site Request Forgeries: Exploitation and Prevention

跨站点请求伪造：开发和预防

William Zeller\* and Edward W. Felten\*†

\*Department of Computer Science

\*Center for Information Technology Policy

†Woodrow Wilson School of Public and International Affairs

Princeton University

fwzeller,felteng@cs.princeton.edu

修订10/15/2008：注意到“纽约时报”已经修复了下面所述的漏洞。 另外澄清了我们的服务器端CSRF保护建议并不能阻止[17]中描述的主动网络攻击。

本文的最新版本可以在这里找到

<http://citp.princeton.edu/csrf/>

摘要  
跨站点请求伪造（CSRF）攻击发生在恶意网站导致用户的Web浏览器对受信任站点执行不必要的操作时。这些攻击被称为基于Web的漏洞的“睡眠巨人”，因为互联网上的许多站点都不能对其进行保护，因为它们在很大程度上被Web开发和安全社区忽视。我们提出了四个严重的CSRF漏洞在四个主要网站上发现，包括我们认为是首次发布的涉及金融机构的攻击。这些漏洞允许攻击者从用户银行帐户中转账，获得用户电子邮件地址，侵犯用户隐私和妥协用户帐户。我们推荐能够完全保护站点免受CSRF攻击的服务器端更改（我们已经实现的）。我们还描述了服务器端解决方案应该具有的功能（缺少这些功能导致CSRF保护不必要地破坏了典型的Web浏览行为）。此外，我们还实施了一个客户端浏览器插件，即使站点没有采取措施保护自身，也可以保护用户免受某些类型的CSRF攻击。我们希望提高CSRF攻击的意识，同时向负责的Web开发人员提供保护用户免受这些攻击的工具

1.简介  
跨站点请求伪造1（CSRF）攻击发生在恶意网站导致用户的Web浏览器对受信任站点执行不必要的操作时。 这些攻击被称为基于Web的漏洞的“睡眠巨人”[23]，因为互联网上的许多站点都不能对其进行保护，因为它们在很大程度上被Web开发和安全社区忽视。 CSRF攻击不会出现在Web Security Threat Classification [12]中，很少在学术或技术文献中进行讨论.2 CSRF攻击简单易于诊断，易于开发，易于修复。 它们存在是因为Web开发人员对于CSRF攻击的原因和严重性没有受过教育。 Web开发人员也可能会受到错误的印象，即针对更好的跨站脚本（XSS）问题的防御也可以防范CSRF攻击。

在第3节中，我们介绍了四个主要站点发现的四个严重的CSRF漏洞。 这些漏洞允许攻击者从用户银行帐户中转账，收获用户电子邮件地址，侵犯用户隐私和妥协用户帐户。

—————————————————

1跨站点请求伪造攻击也称为跨站点参考伪造，XSRF，会话骑乘和混淆副攻击。我们使用术语CSRF，因为它似乎是这种攻击最常用的术语。  
2在ACM数字图书馆搜索“跨站脚本”（与CSRF不同）返回72篇论文，搜索“xsrf OR csrf”只返回四篇论文。 在Safari Books Online上搜索“xss”（收集了4752本技术书籍），该书出现在96本书中，而“csrf OR xsrf”只出现在13本书中。

在第4.1节中，我们建议可以完全保护站点免受CSRF攻击的服务器端更改（我们已经实施）。这些建议具有优于先前提出的解决方案的优点，因为它们不需要服务器状态，并且不会破坏典型的Web浏览行为此外，我们实施了一个客户端浏览器插件，可以保护用户免受某些类型的CSRF攻击（第4.2节）。服务器端保护允许站点完全保护自己免受CSRF攻击，而客户端保护允许用户采取主动措施来保护自身免受许多类型的CSRF攻击，即使站点没有采取措施保护自身。我们希望提高CSRF攻击的意识，同时向负责的Web开发人员提供保护用户免受这些攻击的工具。

2 .CSRF概述

图1，图2和图3显示了CSRF攻击是如何工作的。

下面我们使用一个具体的例子更详细地描述CSRF攻击。

2.1一个例子

让我们考虑一个容易遭受CSRF攻击的网站的假设例子。该网站是一个基于网络的电子邮件网站，允许用户发送和接收电子邮件。该站点使用隐式身份验证（见第2.2节）来验证其用户。一个页面http://example.com/compose.htm包含一个HTML表单，允许用户输入收件人的电子邮件地址，主题和消息以及一个“发送电子邮件”按钮。

<form action="http://example.com/send\_email.htm" method="GET">

Recipient’s Email address: <input type="text" name="to">

Subject: <input type="text" name="subject">

Message: <textarea name="msg"></textarea>

<input type="submit" value="Send Email">

</form>

当example.com的用户点击“发送电子邮件”时，他输入的数据将作为GET请求发送到http://example.com/send\_email.htm。由于GET请求只需将表单数据附加到URL，用户将被发送到以下URL（假设他输入“bob@example.com”作为收件人，“hello”作为主题，“该提案的状态如何？”作为消息）：

[http://example.com/send\_email.htm?to=bob%40example.com&subject=hello&msg=What%](http://example.com/send_email.htm?to=bob%40example.com&subject=hello&msg=What%25)27s+the+status+of+that+proposal%3F3

发送email.htm的页面将收到它收到的数据，并从用户发送电子邮件给收件人。注意，发送email.htm只需要获取数据并对该数据执行操作。它不在乎发出请求的位置，只有请求已经出现。这意味着，如果用户手动将上述URL输入到浏览器中，则example.com仍然会发送电子邮件。例如，如果用户在浏览器中输入以下三个URL，则发送email.htm将发送三封电子邮件（每个邮件给Bob，Alice和Carol）：

http://example.com/send\_email.htm?to=bob%40example.com&subject=hi+Bob&msg=test

http://example.com/send\_email.htm?to=alice%40example.com&subject=hi+Alice&msg=test

<http://example.com/send_email.htm?to=carol%40example.com&subject=hi+Carol&msg=test>

在这里可能发生CSRF攻击，因为send email.htm会接收到任何数据，并发送电子邮件。它不会验证数据源自compose.htm上的表单。因此，如果攻击者可以使用用户send\_email.htm的请求，则该页面将导致example.com代表包含攻击者选择的任何数据的用户发送电子邮件，并且攻击者将成功执行CSRF攻击

—————————————————

3 URL数据编码，将@转换成％40等等

为了利用此漏洞，攻击者需要强制用户的浏览器发送send\_email.htm的请求，以执行一些恶意的操作。（我们假设用户访问攻击者控制的站点，目标站点不会防御CSRF攻击）。具体来说，攻击者需要从他的站点到example.com建立一个跨站点请求。不幸的是，HTML提供了许多方法来提出这样的请求。例如，<img>标签将导致浏览器加载任何设置为src属性的URI，即使该URI不是图像（因为浏览器只能在加载URI后才能告诉URI是一个图像）。攻击者可以使用以下代码创建一个页面：

<img src="http://example.com/send\_email.htm?

to=mallory%40example.com&subject=Hi&msg=My+email+address+has+been+stolen">

当用户访问该页面时，将发送请求send\_email.htm，然后将从用户向Mallory发送电子邮件。这个例子与我们在“纽约时报”网站上发现的实际漏洞几乎相同，我们在3.1节中介绍。

攻击者可能会导致用户浏览器对其他站点执行不必要的操作时，CSRF攻击成功。要使此操作成功，用户必须能够执行此操作。 CSRF攻击通常与用户一样强大，意味着用户可以执行的任何操作也可以由攻击者使用CSRF攻击执行。因此，网站给用户的力量越大，可能的CSRF攻击就越严重。

几乎每个使用隐式认证的站点都可以成功实施CSRF攻击（见第2.2节），并且没有明确地保护自己免受CSRF攻击。

同源策略（见附录B）旨在防止攻击者访问第三方站点上的数据。此策略不会阻止发送请求，它只能防止从第三方服务器返回的数据读取的攻击。由于CSRF攻击是发送请求的结果，所以同源策略不会防止CSRF攻击。

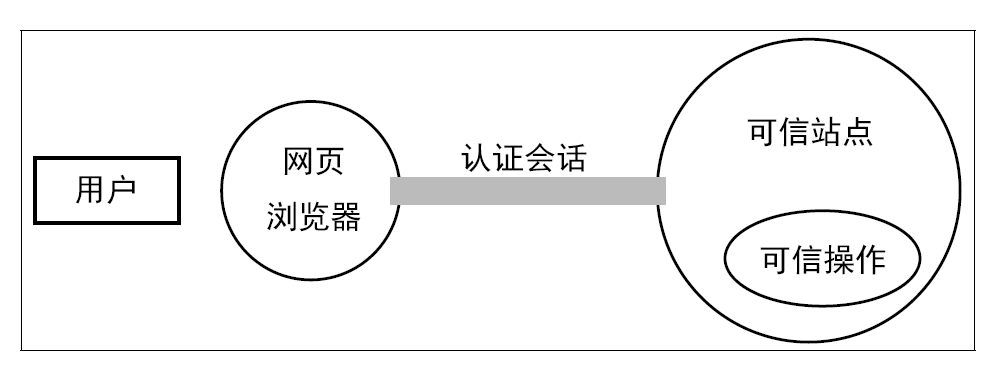


图1：这里，Web浏览器已建立与受信任站点的认证会话。 只有当Web浏览器通过授权会话发出请求时，才能执行“可信操作”。

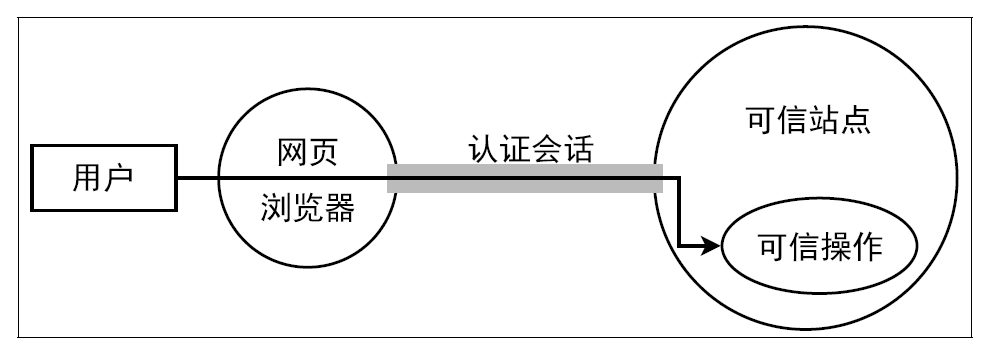


图2：有效请求。 网络浏览器尝试执行受信任的操作。 受信任的站点确认Web浏览器已通过身份验证，并允许执行该操作。

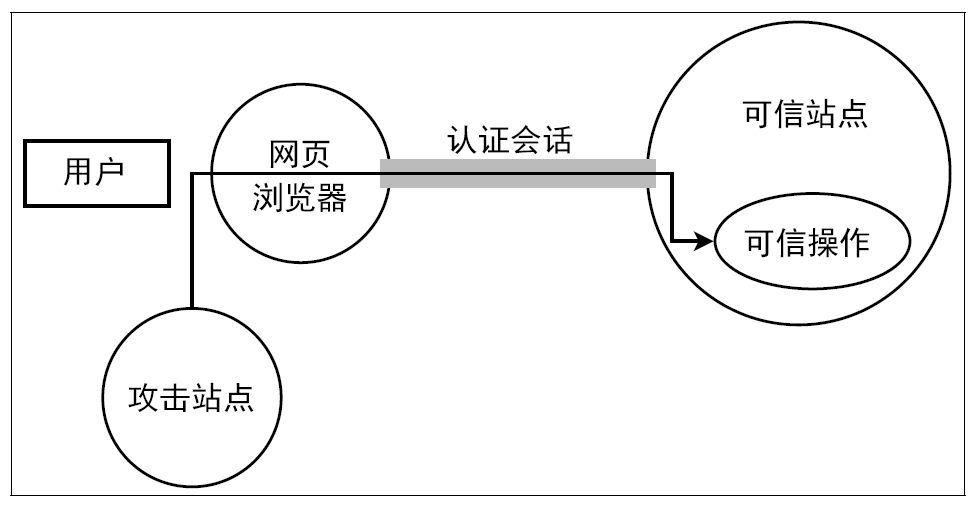


图3：CSRF攻击。攻击网站导致浏览器向受信任的站点发送请求。 受信任的站点从Web浏览器看到一个有效的经过身份验证的请求，并执行“可信任操作”。 CSRF攻击是可能的，因为网站认证Web浏览器，而不是用户。

2.2认证和CSRF

CSRF攻击经常利用目标网站的认证机制。问题的根源在于，Web身份验证通常保证来自某个用户浏览器的请求的站点; 但不能确保用户实际请求或授权请求。

例如，假设Alice访问目标站点T.T给Alice的浏览器一个包含伪随机会话标识符sid的cookie，以跟踪她的会话。Alice被要求登录到该站点，并在输入有效的用户名和密码后，该站点记录了Alice登录到会话sid的事实。 当Alice向T发送请求时，她的浏览器会自动发送包含sid的会话cookie。 T然后使用其记录将会话识别为来自Alice。

现在假设Alice访问恶意网站M.由M提供的内容包含Javascript代码或导致Alice的浏览器向T发送HTTP请求的图像标签。由于请求将转到T，Alice的浏览器“有帮助”附加会话cookie sid 要求。 在查看请求时，T推测来自Cookie的请求来自Alice，所以T对Alice的帐户执行请求的操作。 这是一次成功的CSRF攻击。

大多数其他Web认证机制也遇到同样的问题。例如，HTTP BasicAuth机制[22]将让爱丽丝告诉她的浏览器她的T的网站的用户名和密码，然后浏览器将“有帮助”地将用户名和密码附加到将来发送给T的请求。或者，T可能会使用客户端边SSL证书，但同样的问题会导致浏览器会“有帮助”地使用证书来向T的站点执行请求。类似地，如果T通过她的IP地址认证Alice，则CSRF攻击是可能的。

一般来说，每当认证发生时，由于哪个站点发送了一个请求以及哪个浏览器来自哪里，那么就有一个CSRF攻击的危险。 原则上，可以通过要求用户对发送到某个站点的每个请求采取明确的，不可扩展的操作（例如重新输入用户名和密码）来消除这种危险，但实际上这会导致主要的可用性问题。 最标准和广泛使用的认证机制不能防止CSRF攻击，因此在其他地方必须寻求实际的解决方案。

2.3 CSRF攻击载体

要使攻击成功，用户必须登录到目标站点，并且必须访问攻击者的站点或攻击者部分控制的站点。

如果服务器包含CSRF漏洞，并且接受GET请求（如上例所示），则可以在不使用JavaScript的情况下进行CSRF攻击（例如，可以使用简单的<img>标记）。 如果服务器只接受POST请求，则需要JavaScript才能将POST请求从攻击者的站点发送到目标站点。

2.4 CSRF与XSS

最近非常注意跨站脚本（XSS）[20]漏洞。 当攻击者注入恶意代码（通常为

JavaScript）转换为网站，目的是定位该网站的其他用户。 例如，一个网站可能允许用户发表评论。 这些注释由用户提交，存储在数据库中，并显示给该网站的所有未来用户。 如果攻击者能够在注释中输入恶意JavaScript，则JavaScript将嵌入到包含注释的任何页面上。 当用户访问该站点时，攻击者的JavaScript将以目标站点的所有权限执行。 嵌入到目标站点中的恶意JavaScript将能够发送和接收网站上任何页面的请求，并访问该站点设置的Cookie。从XSS攻击的保护措施需要网站仔细筛选任何用户输入，以确保不会注入恶意代码。

CSRF和XSS攻击的不同之处在于XSS攻击需要JavaScript，而CSRF攻击则不需要。 XSS攻击要求网站接受恶意代码，而使用CSRF攻击恶意代码位于第三方站点。 过滤用户输入将防止恶意代码在特定站点上运行，但不会阻止恶意代码在第三方站点上运行。 由于恶意代码可以在第三方站点上运行，所以防止XSS攻击不能保护站点免受CSRF攻击。 如果一个站点容易遭受XSS攻击，那么它容易遭受CSRF攻击。 如果一个站点完全受到XSS攻击的保护，那么很可能仍然容易受到CSRF攻击。

3 .CSRF漏洞

在本节中，我们描述了我们发现的四个漏洞。这些攻击是通过测量大约十个流行网站的列表来发现的。 我们分析的许多网站都有CSRF漏洞或漏洞的历史（例如，网页搜索将显示一个已经被修复的CSRF漏洞的报告）。 事实上，这么多网站容易遭受CSRF攻击，直到第三方披露这些问题表明，许多网站管理员没有受到关于CSRF漏洞的风险和存在的教育。

我们相信ING Direct，Metafilter和YouTube以及“纽约时报”已经纠正了我们下面描述的漏洞。 所有四个站点似乎已经使用类似于我们在4.1节中提出的方法来解决问题。