跨站请求伪造

“谁打开了饼干罐？”

部分基于Collin Jackson等人的/ OWASP幻灯片

OWASP十佳 （2013年）

A-1	注射	不可信数据作为命令或查询的一部分发送到解释器。
A-2	身份验证和会话管理的其他实施缺陷	攻击密码，密钥或会话令牌或利用以假定其他用户的身份。
A-3	跨站点脚本	应用程序获取不受信任的数据并将其发送到Web浏览器，而没有进行正确的验证或转义
…	各种实作问题	…暴露文件，目录或数据库密钥而没有访问控制检查，…配置错误，…缺少功能级别的访问控制
A-8	跨网站要求伪造	登录的受害者的浏览器发送伪造的HTTP请求，包括受害者的会话cookie和其他身份验证信息

https://www.owasp.org/index.php/Top_10_2013-Top_10

更多OWASP

- «跨站请求伪造（CSRF）是一种攻击 **强制最终用户** 在其中存在有害行为的Web应用程序上执行有害行为 **目前已认证** …在社会工程学的帮助下（例如通过电子邮件或聊天发送链接），攻击者可能会欺骗Web应用程序的用户执行攻击者的操作

  选择…»

2007年：Gmail被黑了……

- 登录Gmail后，访问恶意网站的用户会生成一个请求，该请求被理解为源自受害者用户

- 这用于注入电子邮件过滤器，将受害者用户的电子邮件转发给攻击者
- 允许攻击者控制davidairey.com（因为域注册商使用基于电子邮件的身份验证……）

浏览器执行模型

- 每个浏览器窗口/框架
  - 上传网页内容
  - 呈现Web内容，静态（HTML，子帧）或动态（脚本）以显示页面
    - 包括图像之类的外部资源
  - 响应事件（见下文）
- 大事记
  - 渲染：OnLoad
  - 时间：setTimeout（），clearTimeout（）
  - 对用户操作做出反应：OnClick，OnMouseover

维护客户状态

- Web交互本质上是无状态的
  - 来回发送HTTP请求
- 如何知道哪个浏览器连接？
- 维持状态的方法：
  - Cookies：浏览器状态
  - 会话：服务器状态
  - URL重写：浏览器状态
  - 甚至更多的选择：cf. http://en.wikipedia.org/wiki/HTTP_cookie

状态管理：Cookies

- “脚本可以存储在客户端计算机上的少量信息”
- 可以在HTTP标头中设置
  - 产地和有效期
  - 例：

HTTP / 1.0 200 OK
内容类型：text / html
Set-Cookie：名称=值
Set-Cookie：name2 = value2; Expires = 2021年6月9日星期三10:18:14 GMT（页面内容）

- 操作方式：
  - 当浏览器连接到URL时，它首先检查相关的cookie
  - 如果找到该URL的cookie，它将通过HTTP请求将cookie信息发送到服务器。
  - 一个网页可以包含来自多个网站的内容，因此在浏览过程中可以发送多个cookie

- 寿命长：用户标识（首选项，身份验证，跟踪...）
  - Cookie =用户ID，可能受到保护（完整性，机密性）
- 临时：会话标识
  - Cookie =随机数
- “安全”属性指示cookie仅应通过HTTPS发送（保密以防止中间人攻击）

HTTP Cookies安全历史记录

- 1994年：Netscape – Cookie源自并且仍然主要基于4页的草稿
- 1997：RFC 2109 –隐私问题，意图
- 2000年：RFC 2965 –使用方面的进一步建议
- 2002年：HttpOnly（XSS）
- 2011年：RFC6265-
- 2017年进行中：RFC 6265bis（草稿）-SameSite

状态管理：会议

- 通常由网络框架处理

- 有助于区分其他同时进行的会话
- 数据存储：
  - 会话存储正在进行的交易（工作流，购物车，登录）中的数据
  - 信息也可以从会话中删除
- 操作方式：
  - 开始会议
  - 会话ID是在浏览器中设置的（开头是Cookie，稍后是URL重写）在Web服务器上存储和管理的数
  - 据（成本高，无法扩展）
  - 结束会话（处理数据）

- 优点/缺点：由服务器管理数据并由服务器管理

状态管理：URL重写

- 网址已修改为：
  - 存储参数（RESTful方法）

    例如，http：// host：port / shopping.html; sessionid = value
  - 强制使用代理：目标成为参数

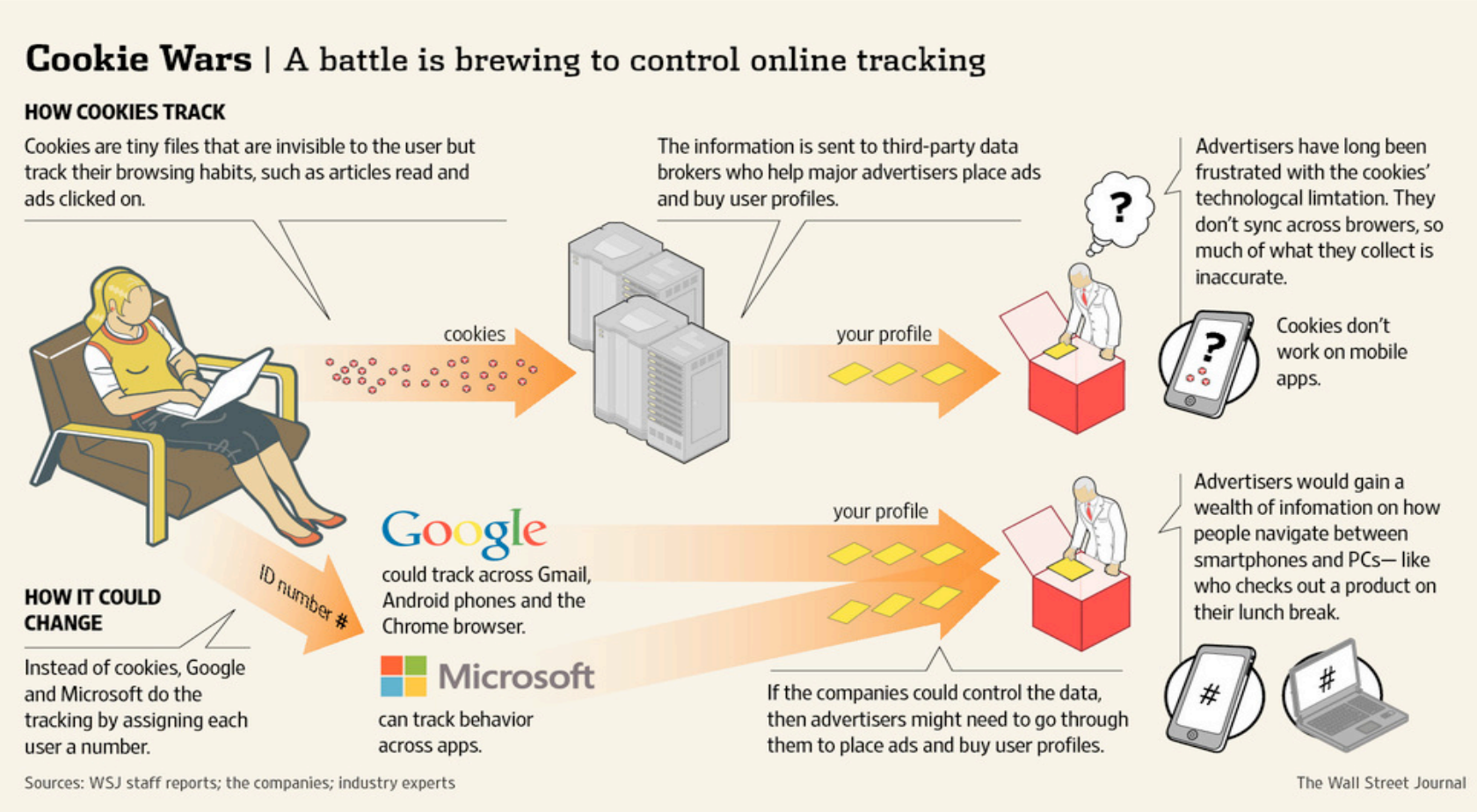
- 操作（例如：Google）
  - 研究结果指向：https://www.google.fr/url？q = http://fr.wikipedia.org/Cookie_(informatique） & sa = ü & ei = ü

    -9wU-27O8Gm0AWc2IGAAQ & usg = AFQjCNEltv3EUaJHvFL_fM-

    _7lmX9VzCLQ & sig2 = wdr5pg0cOye893nHZJO-hw & bvm = bv.66330100，d.bGQ
  - 代替：http://fr.wikipedia.org/wiki/Cookie_%28informatique%29
  - 在页面上不可见（链接不以纯文本显示），仅在链接栏中

- 优点：不能被客户抑制

大数据大战...



< Cookie的生命周期也是一个严重的问题。隐私权！

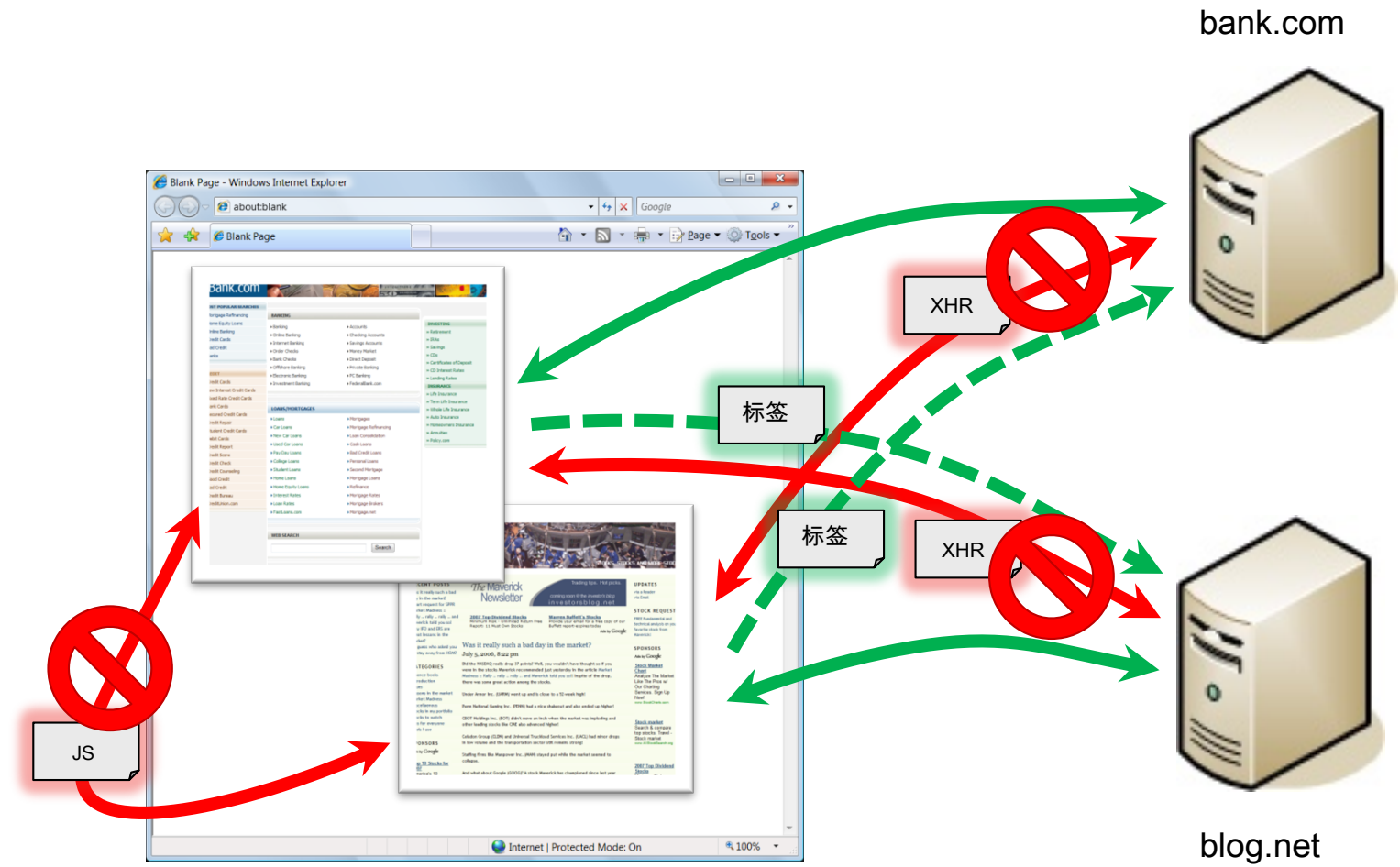
< 过期/最大年龄属性

< 应用程序应使无关的cookie无效，而不要依赖bowser来删除它们

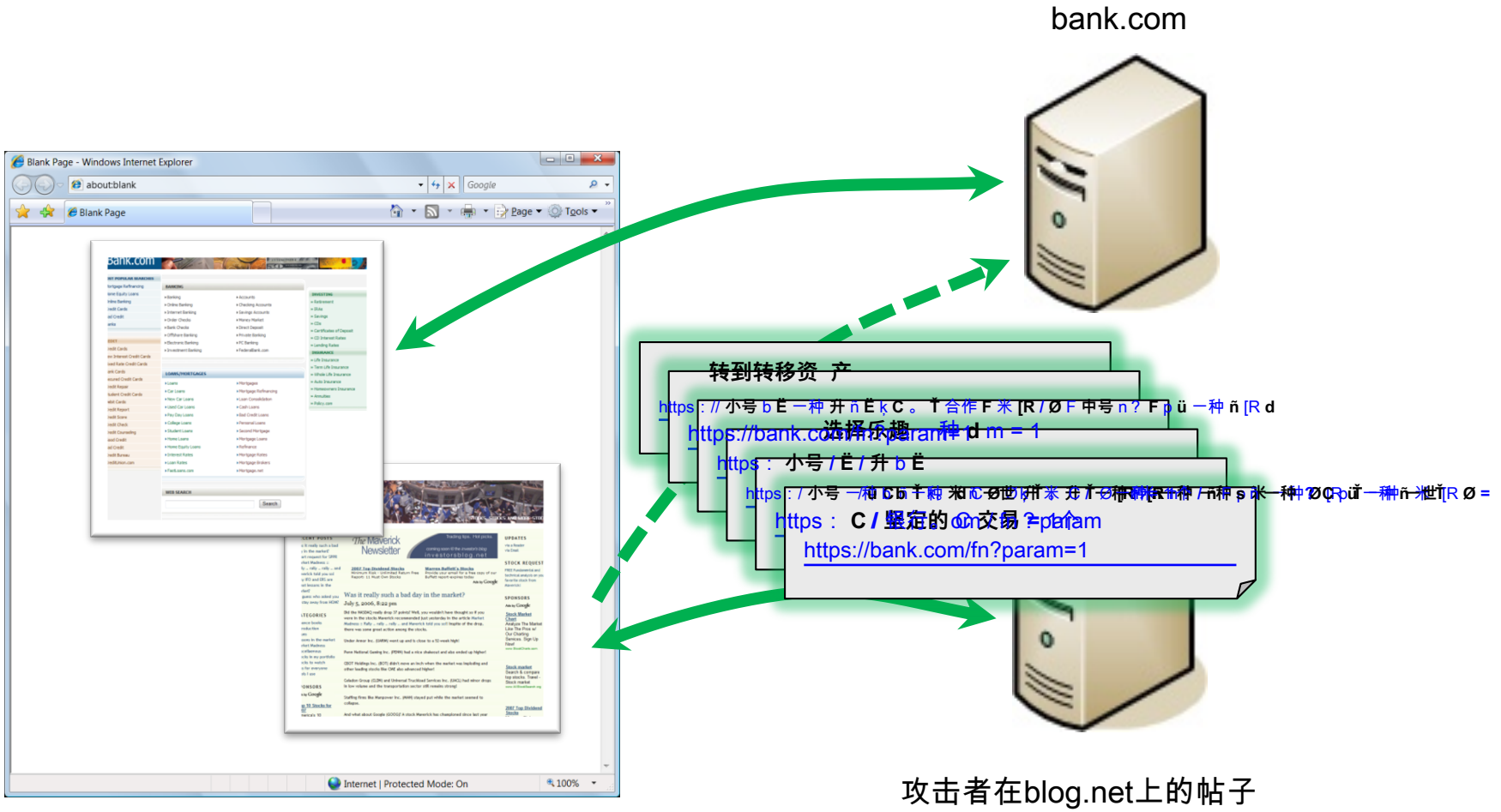
浏览器“相同来源”策略（SOP）

- 浏览器中的每个框架都与一个域相关联
  - 一个域由下载框架内容的服务器，协议和端口确定
  - 如果框架明确包含外部代码，则该代码将在框架域内执行，即使它来自其他主机
- 脚本只能访问与相同来源关联的资源（尤其是Cookie）
  - 防止恶意脚本篡改浏览器中的其他页面
  - 防止脚本监听其他窗口的输入（密码）
- 安全问题：主要是浏览器错误
  - 特别是在1990年代末-2000年代初

浏览器“相同来源”策略


bank.com
Blank Page - Windows Internet Explorer
about:blank
Blank Page
XHR
标签
标签
XHR
JS
blog.net
文件，cookie

跨站请求伪造


bank.com
Blank Page - Windows Internet Explorer
转到转移资 产
https://bank.com/fn?param=1
攻击者在blog.net上的帖子

CSRF如何工作？

< 劫持浏览器固有的功能和HTTP规范的某些方面

< SOP控件和cookie

< 特权升级攻击类型

< “困惑的代理人”：浏览器认为标签/表单/ XHR来自与目的地相同的来源

< 攻击者执行盲目攻击（看不到服务器响应）

< 除非与XSS结合使用……

< 标签

<img src =“ https://bank.com/fn?param=1 ”>

<iframe src =” https://bank.com/fn?param=1 ”>

<script src =“ https://bank.com/fn?param=1 ”>

< 自动投寄表格

<body onload =“ document.forms [0] .submit （ ） ”>

<form method =“ POST” action =“ https://bank.com/fn ”>

<input type =“ hidden” name =“ sp” value =“ 8109” /> </ form>

< GET请求是最危险的，但是任何请求都是易受攻击的（也是POST）

< XmlHttpRequest（AJAX）

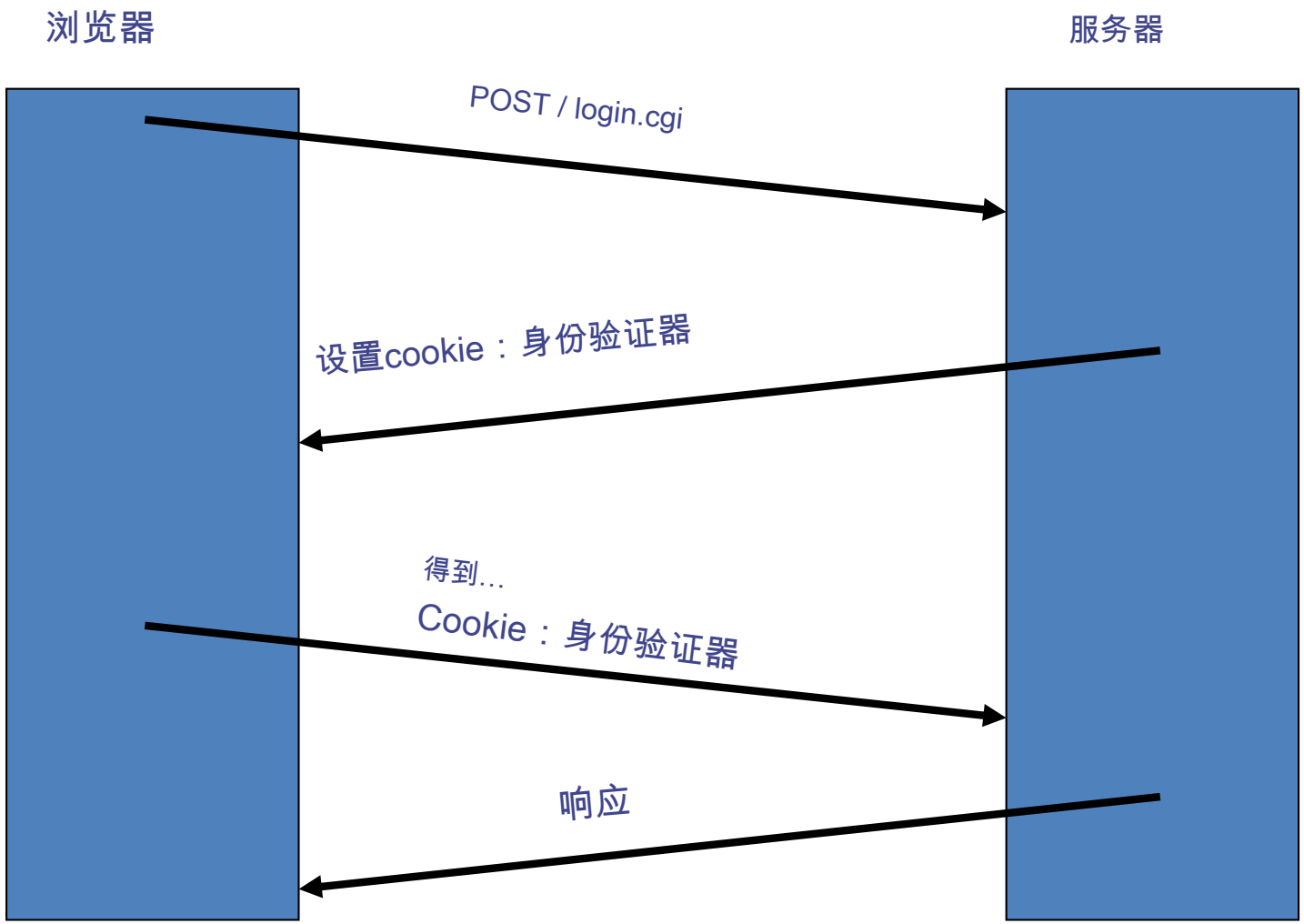
4 通常受同一原产地政策的约束

4 但是管理不善的CORS（跨源资源共享）可能会放松这些限制……

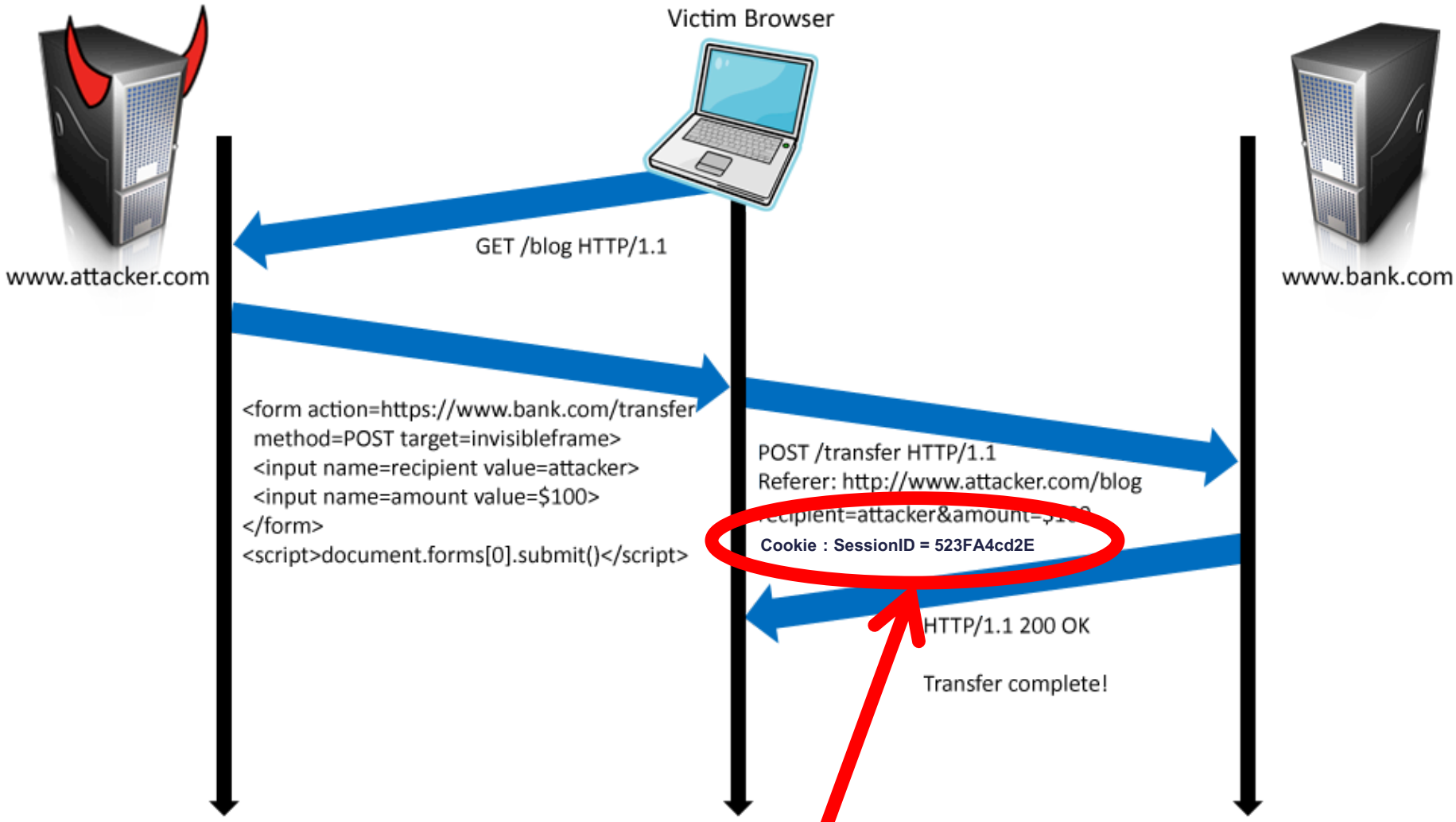
4 也可能被代理人欺骗

身份验证：使用Cookie的会话

< 浏览器行为：自动附加服务器先前设置的cookie

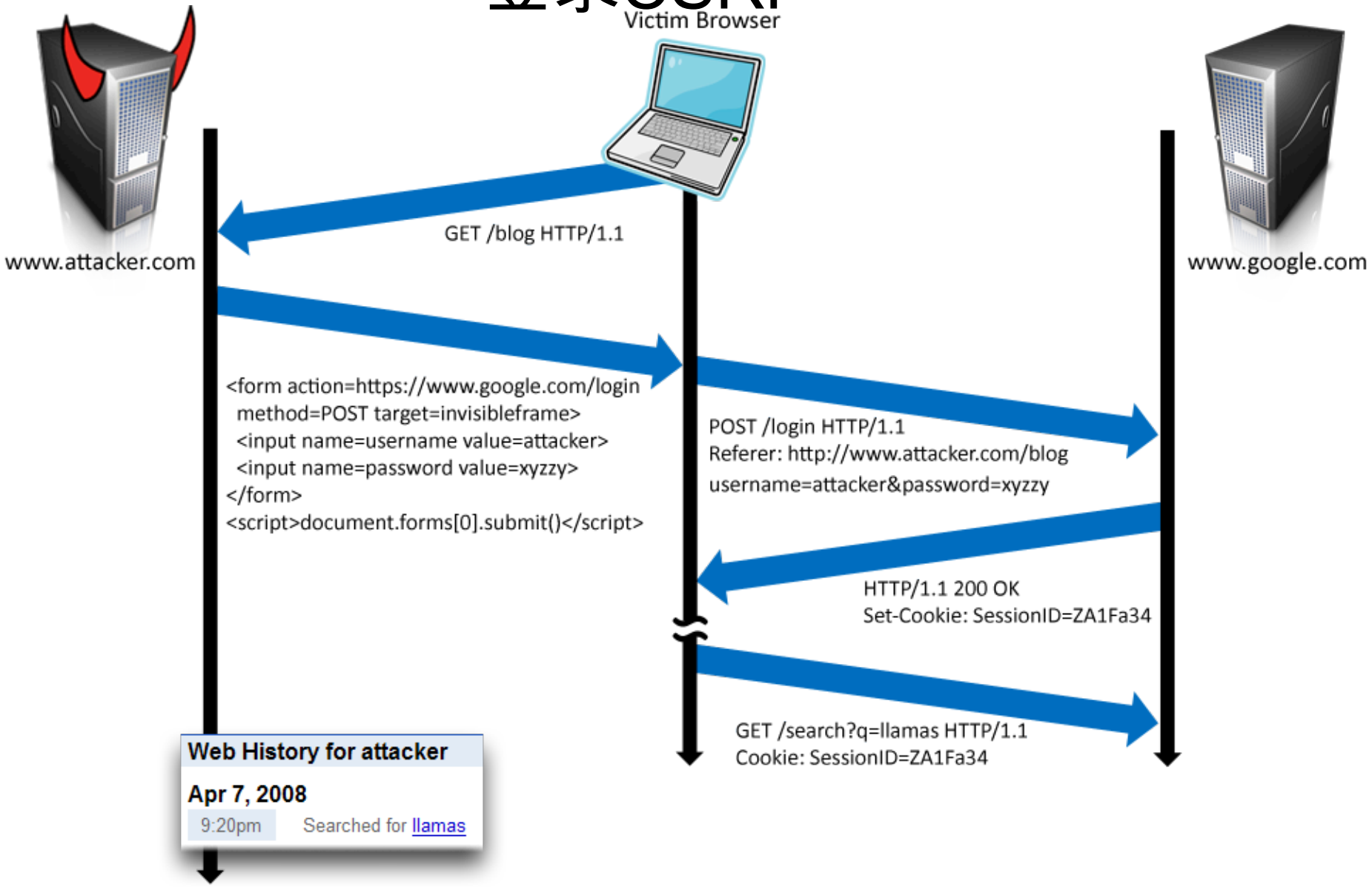


CSRF：带Cookie的表单发布



问：您要登录Gmail多长时间？Facebook的？…。

登录CSRF


Victim Browser
www.attacker.com
www.google.com
GET /blog HTTP/1.1
<form action=https://www.google.com/login
method=POST target=invisibleframe>
<input name=username value=attacker>
<input name=password value=xyzzy>
</form>
<script>document.forms[0].submit()</script>
POST /login HTTP/1.1
Referer: http://www.attacker.com/blog
username=attacker&password=xyzzy
HTTP/1.1 200 OK
Set-Cookie: SessionID=ZA1Fa34
GET /search?q=llamas HTTP/1.1
Cookie: SessionID=ZA1Fa34
Web History for attacker
Apr 7, 2008
9:20pm
Searched for llamas

不只是Web服务器：对家用路由器的攻击

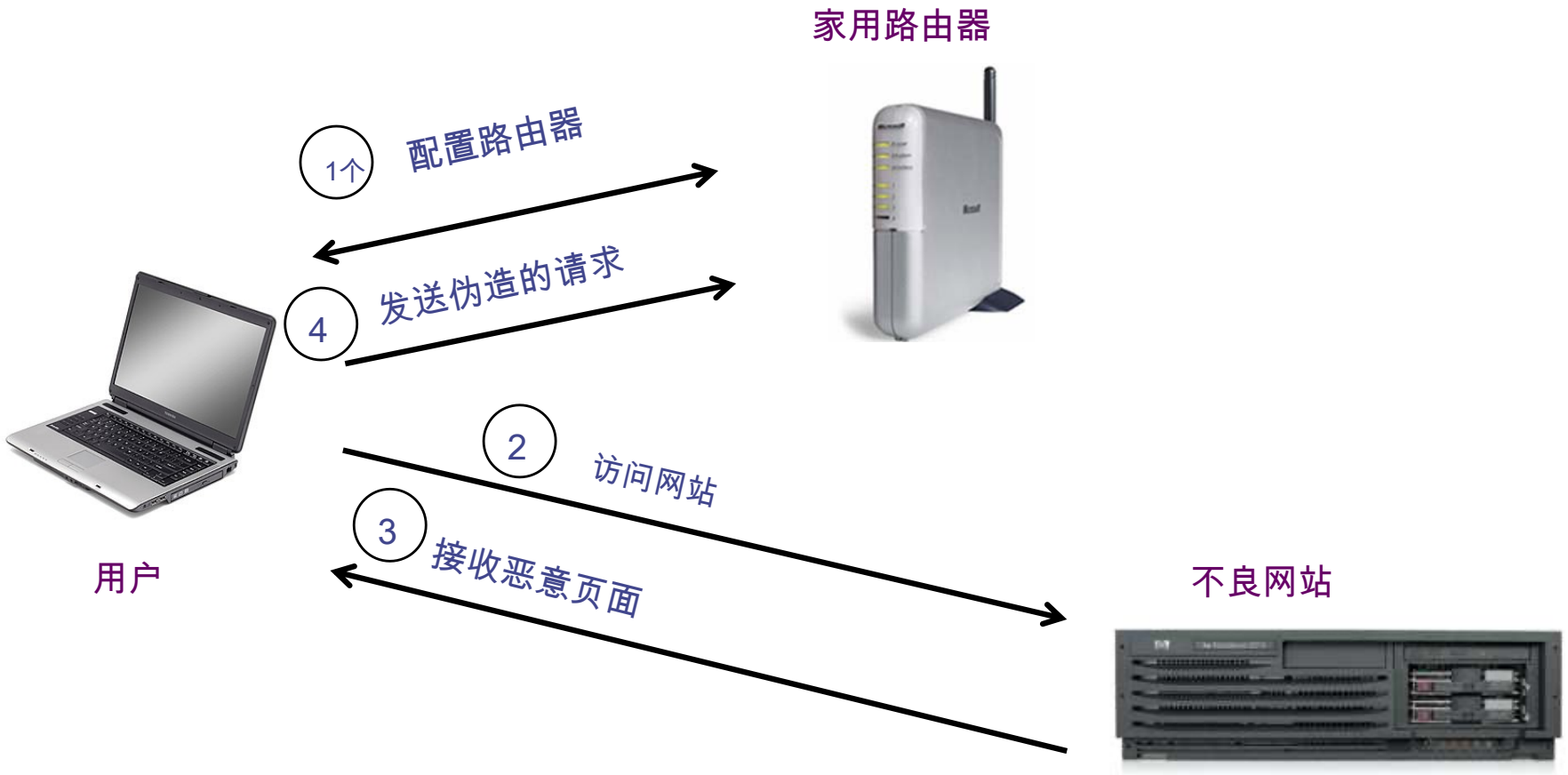
[Stamm，Ramzan，Jakobsson 2006]

- 事实：
  - 50%的家庭用户使用带有默认密码或没有密码的宽带路由器

- 偷渡式Pharm攻击：
  - 场景：用户访问恶意网站
  - 攻击者脚本扫描家庭网络中的宽带路由器：
    - SOP允许“仅发送”消息
    - 使用onError和可能的地址（例如192.168.0.1）检测成功：

      <IMG SRC = 192.168.0.1 onError = do ( ) >
  - 攻击者脚本可以登录路由器并更改DNS服务器
    - 控制用户导航
  - 攻击者可以将恶意软件分发到路由器
  - 攻击者可以阻止病毒定义更新
  - 攻击者可以宣传易受攻击的主机

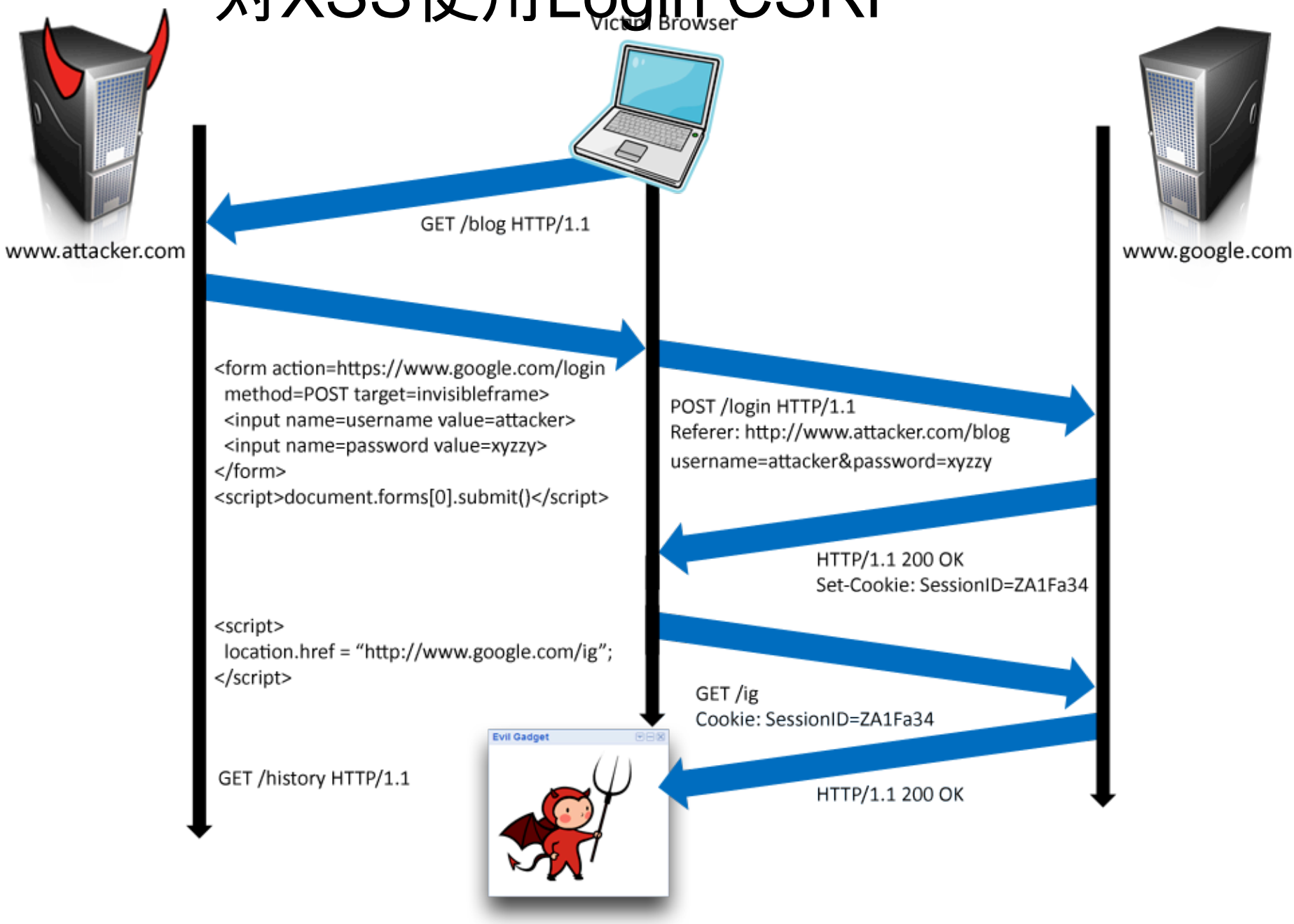
无Cookie示例：家用路由器



CSRF的广阔视野

- 滥用跨站点数据导出功能
  - 从用户的浏览器到诚实的服务器
  - 破坏用户会话的完整性
- 为什么要发起CSRF攻击？
  - 网络连接
  - 读取浏览器状态
  - 编写浏览器状态
- 不只是“会话骑行”

对XSS使用Login CSRF



攻击者的观点

- 攻击者可以：
  - 控制表单/ XHR有效载荷
  - 控制内容类型（«enctype»属性）
  - 控制方法（GET或POST）
- 攻击者不能：
  - 控制其他标题
  - 控制Cookie

CSRF基本防御

- 推荐人验证

```
引荐来源：http：//www.facebook.com/home.php
```

- 永久身份验证（登录/会话数据）：
  - 客户端会话信息的存储（无效）
    - 但是容易受到XSS攻击……
    - ……以及用户对服务器状态的操纵！
  - 服务器端会话ID +秘密令牌验证

```
<输入类型=隐藏值= 23a3af01b>
```

- 自定义HTTP标头：AJAX / XHR的更简单方法

```
X-Requested-By：XMLHttpRequest
```

推荐人验证防御

- HTTP Referer标头
  - 引荐来源：http：//www.gmail.com/ 好
  - 引荐来源：http://www.bad.com/evil.html KO
  - 推荐人： ???
- 宽松引荐来源验证
  - 如果没有推荐人，则不会阻止请求
- 严格的引荐来源验证
  - 安全，但有时没有推荐人…

推荐人隐私问题

- 推荐人还可能泄漏对隐私敏感的信息！

  http://intranet.corp.apple.com/

  projects / iphone / competitors.html

- 可能会根据用户的偏好在浏览器中删除

- 网站通常无法阻止这些用户

那么……宽松引荐检查？

- 其他常见的阻止来源：
  - 组织的网络剥离（代理）
  - 通过本地计算机进行网络剥离
  - 被浏览器剥离以进行HTTPS-> HTTP转换
  - 笨拙的用户代理
- 不安全：攻击者可能剥夺引荐来源，例如：

ftp://www.attacker.com/index.html

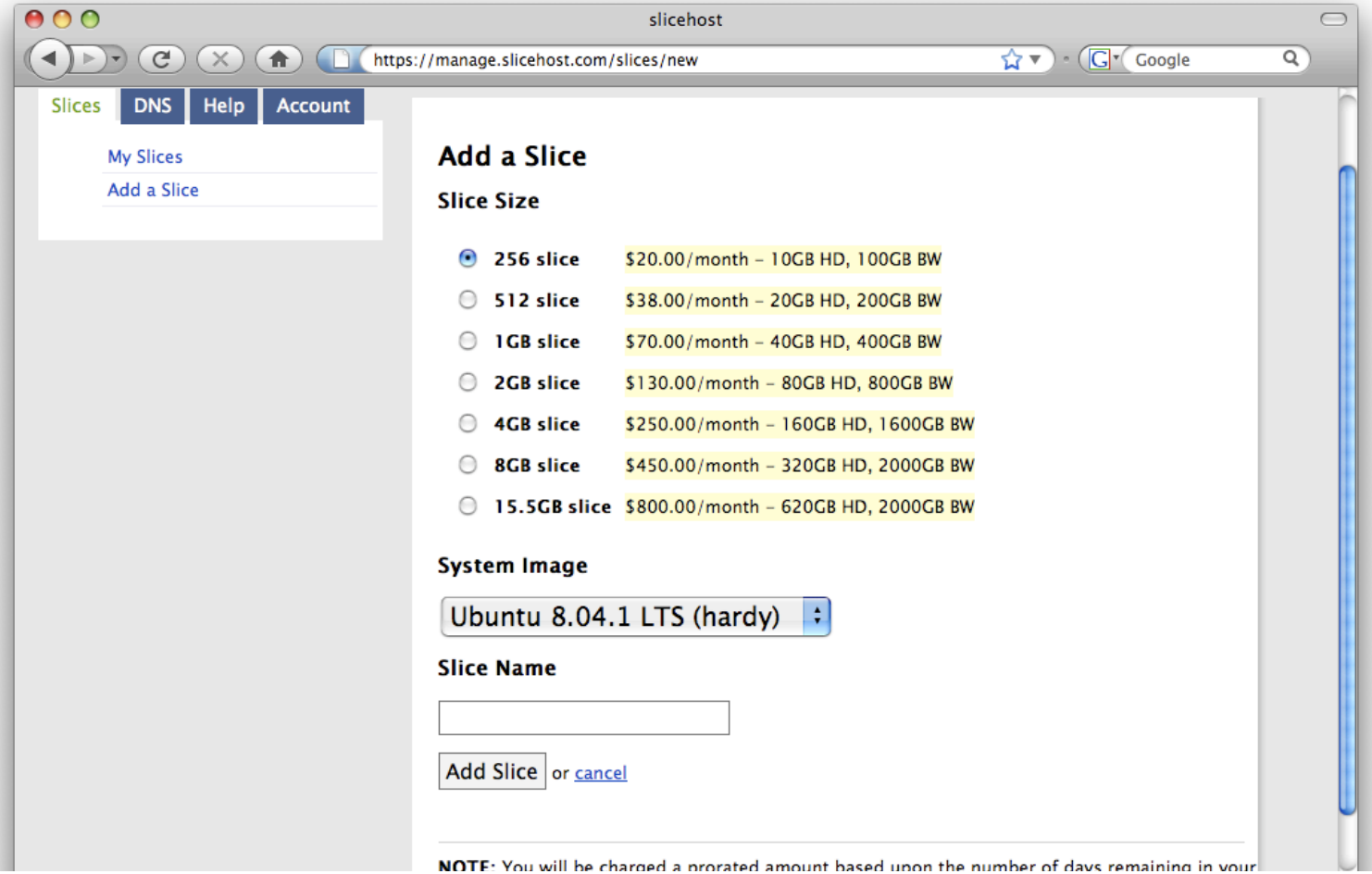
javascript：“ <script> / * CSRF * / </ script>” data：text /

html，<script> / * CSRF * / </ script>

秘密令牌验证

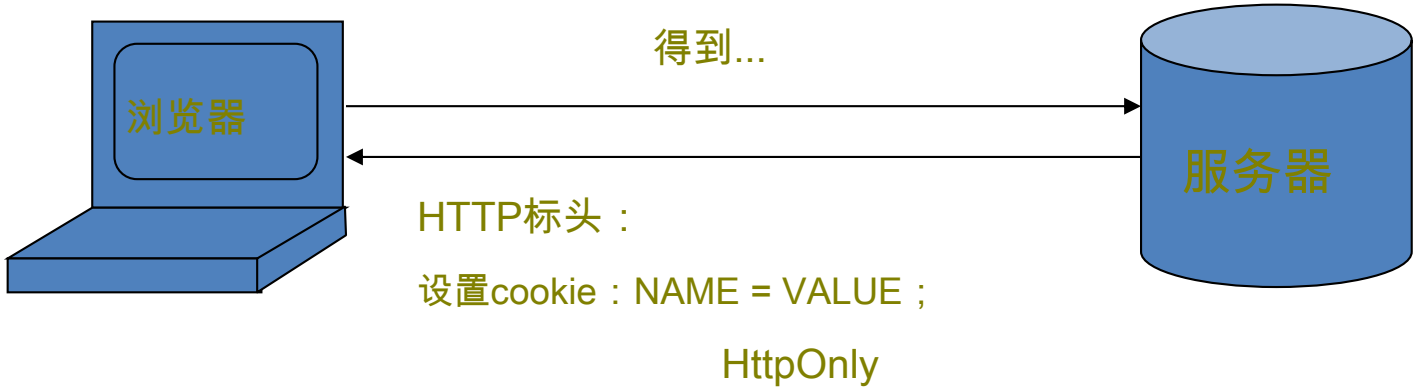
- 请求中包含难以猜测的秘密
  - 不可猜测性取代不可伪造性
- 变化
  - 会话标识符
  - 与会话无关的令牌
  - 会话相关令牌
  - 会话标识符的HMAC / MD5 / SHA-1用于完整性保护

秘密令牌验证




```
g:0"><input name="authenticity_token" type="hidden" value="0114d5b35744b522af8643921bd5a3d899e7fbd2" /></d
="/images/logo.jpg" width='110'></div>
```

XSS：HttpOnly Cookies

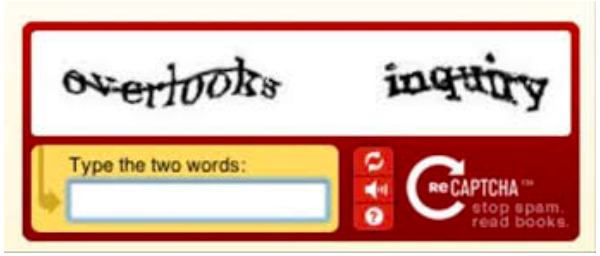


- 通过HTTP发送的Cookie，但脚本无法访问
  - 无法通过读取 document.cookie
    - 还阻止对XMLHttpRequest标头的访问
  - 帮助防止通过XSS盗窃Cookie

…但不能阻止大多数其他风险：典型的攻击是溢出cookie存储库（用攻击者值替换cookie）！这取决于浏览器的实现……

其他缓解策略

- 令牌：如果在服务器端维护CSRF令牌是有问题的，则需要双重提交：令牌要在标头（请求参数）中发送，而主体中要包含Cookie
  - 严格的要求（特别是HTTPS可以防止攻击者注入cookie，加密的cookie）
- 其他反CSRF HTML元素
  - 源头
  - SameSite cookie（自2017年以来起草RFC 6265bis）
  - 检查一下 https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
- 使用具有内置反CSRF机制的库和框架
  - 例如Angular：“ X-XSRF-TOKEN”
- 关键操作之前基于用户交互的CSRF防御
  - 验证码：确保有人干预（没有自动欺骗）
  - 一次性令牌
  - 重新认证（登录名/密码）



深度防御：起源头

来源：http：//www.evil.com

- 具有较少隐私问题的替代推荐人
- 仅在POST上发送，仅发送必要的数据
- 防御基于重定向的攻击
- 隐私
  - 仅标识发起请求的主体（而不是路径或查询）
  - 仅针对POST请求发送；以下超链接没有显示任何内容
- 易用性
  - 使用简单的防火墙规则授权子域和会员站点

```
SecRule REQUEST_HEADERS：Host！^ www \ .example \ .com（ ：\ d + ）？$ deny，status：403 SecRule
REQUEST_METHOD` POST $ chain，deny，status：403
SecRule REQUEST_HEADERS：来源！^（https？：// www \ .example \ .com（ ：\ d + ）？ ）？
```

  - 无需管理秘密令牌状态
  - 与现有防御措施一起使用以支持旧版浏览器（例如Referer）
- 标准化
  - 由W3C XHR2和JSONRequest支持

深度防御：SameSite Cookie

draft-ietf-httpbis-rfc6265bis-latest （2019年10月8日）

- 设置：

```
Set-Cookie: CookieName=CookieValue; SameSite=Lax;
Set-Cookie: CookieName=CookieValue; SameSite=Strict;
```

  - 严格：Cookie不会包含在第三方发送的请求中（可能会对浏览体验产生负面影响）
  - 松懈：Cookie将与第三方网站发起的GET请求一起发送，但仅适用于顶级导航请求（必须在浏览器中更改URL）
  - 浏览器正在逐步集成此功能

还有一件事...

- Cookie范围：
  - 基于路径属性+主机/域
  - 将Cookie的使用限制为网站上的某些应用程序
  - 这与基于主机/域+端口的SOP分开
  - 可能进一步限制Cookie滥用

外卖留言

- Cookie保护可能很棘手，特定于浏览器，并且仍在研究和标准化
- “安全” cookie的原型？

  Set-Cookie：__ Host-SessionID = 43a2;
  路径= / myapplication;安全; HttpOnly; SameSite =严格
- …也就是说，直到下一个RFC 6265bis版本…
- …加上代币…
- …以及通过HTTPS！
- 注意可能危害cookie完整性（写攻击）的XSS和MITM
- …。和隐私！