当打开一个web页面，呈现出来的数据有：

服务端存储：数据库，内存，文件系统

客户端存储：本地cookie，Flash cookies

文本数据：html，JavaScript，css

多媒体数据：Flash，MP3

Web安全问题：

1. **点击劫持：**

点击劫持是一种视觉上的欺骗手段。有两种攻击方式：

(1) 攻击者使用一个透明的iframe，覆盖在一个网页上，然后诱使用户在该页面上进 行操作，此时用户将在不知情的情况下点击透明的iframe。

(2) 攻击者使用一张图片覆盖在网页，遮挡网页原有位置的含义。利用图片的style 或者能够控制的css，将图片覆盖在网页上，形成点击劫持。

防御：

要想防御点击劫持，你得知道什么样的情况下可以产生点击劫持：

1) 恶意网页能够以一定的方式存在于正常网页中，比如iframe;

2) 客户端浏览器cookie,攻击者可以利用用户身份进行恶意攻击;

3) 网页之上有欺骗鼠标操作交互，诱骗用户点击，拖放。

客户端进行防御：

目前所有的浏览器都提供了防御点击劫持的安全机制，所以客户端可以通过及 时更新浏览器，修复安全漏洞，不随意访问点击不安全的网页进行防御。

服务器端防御：

1) 可以在html responses中加入一个X-Frame-Options字段,浏览器会根据 X-Frame-Options字段中参数来判断页面是否可以被iframe标签包含，通 过拒绝或者限制iframe载入，黑客就无法通过iframe隐藏特定的网页。

有三个可选的值：

deny：浏览器会拒绝当前页面加载任何frame页面;

sameoption：frame页面的地址只能为同源域名下的页面;

allow-from origin：允许frame加载的页面地址。

一般是后台进行设置防御，通过配置apache, nginx, 或者IIS来进行防御。

apache：

Header always append X-Frame-Options SAMEORIGIN

nginx：

add\_header X-Frame-Options SAMEORIGIN

IIS:

<httpProtocol>

<customHeaders>

<add name="X-Frame-Options" value="SAMEORIGIN" />

</customHeaders>

</httpProtocol>

2) 使用验证码验证用户。

1. **跨站脚本攻击XSS：**

发生在浏览器客户端，（黑客通过html注入篡改页面，插入恶意脚本，从而在用户浏 览页面时控制用户浏览器的一种攻击。）

1. **SQL注入攻击：**

基于数据库的一种攻击。(把用户输入的数据当做代码来执行)，一般后台在开发的时候进行防御。

要想产生这种攻击：

(1) 用户能够控制输入;

(2) 原本程序要执行的代码拼接了用户输入的数据.

**4. CSRF：跨站点请求伪造漏洞:**

CSRF攻击--可以在受害者毫不知情的情况下以受害者名义伪造请求发送给受攻击站 点，从而在并未授权的情况下执行在权限保护之下的操作。

黑客借助受害者的cookie骗取服务器的信任，但是黑客拿不到cookie，也看不到 cookie，由于浏览器同源策略的限制，黑客也无法解析。因此黑客无法从返回的结果中 得到任何东西，他所能做的就是服务器发送请求，以执行请求中所描述的命令，在服务 端直接改变数据的值，而非窃取服务器中的数据。

攻击者伪造的请求之所以能够被搜狐浏览器通过，是因为用户的浏览器成功发送了 cookie的缘故。

(1) session Cookie：生存周期为浏览器关闭前，浏览器关闭后session Cookie就失效了。

(2) 本地Cookie：生存周期为过期时间,过期时间过了才失效。

CSRF攻击对象：可以直接产生数据改变的服务。

防御：

(1) 验证HTTP Referer字段(记录该http请求的来源地址)。

在http头中有一个referer属性，用来记录请求的来源地址。

比如：当用户登录某银行网站进行转账，输入用户名，密码验证成功后，点击 转账按钮，进入到转账的页面，此时referer值就是转账页面所在的URL，而 如果黑客要对某银行网站进行csrf攻击，他只能在自己的网站发送请求到银 行，这时http的referer指向黑客自己的网站，因此要防御CSRF,银行网站 要对发送的转账请求，验证其referer值，如果是本网站的域名则证明是正常 的请求，否则拒绝该请求。

好处：简单易行，开发者在最后给所有的安全敏感的请求统一增加一个拦截器

检查referer的值就可以。

缺点：referer的值是由浏览器提供的，每个浏览器对于referer的具体实现 可能有差异，并不能保证浏览器本身没有安全漏洞。

(2) 在请求字段中添加token并验证。

黑客之所以能够攻击成功是因为用户的一些登录信息是放在cookie中的，黑 客可以通过获取cookie伪造请求来通过网站的安全验证，要防止CSRF，前提 是在请求中添加黑客所不能伪造的信息。可以在请求中以参数的形式添加 token，并且在服务器端设置拦截器来验证这个token，发现请求中没有token 或者token不正确，则认为是CSRF攻击，拒绝该请求。

}

如何选择用哪一种方法来防御CSRF：

(1) 现有系统，短时间内获得一定程度的CSRF攻击，可以采用referer。

(2) 需要高安全性：token

(3) 如果开发一个全新的系统，则抵御csrf的选择要大得多。可以对于重要的服务使 用XMLHttpRequest来访问，这样增加token要容易得多。

1. **文件上传漏洞：**

文件上传漏洞是指用户上传了一个可执行的脚本文件，并通过此脚本文件获得了执行服务器命令的能力。【上传web脚本能够被web服务器解析】

文件上传后导致的常见安全问题有：

(1) 上传文件是web脚本语言，服务器的web容器解释并执行了用户上传的脚本，导致代码执行;

(2) 上传文件是flash的策略文件，黑客用于控制flash在该域下的行为;

(3) 上传文件是病毒，木马文件，黑客用以诱骗用户或者管理员下载执行;

(4) 上传文件是钓鱼图片或者包含了脚本的图片，在某些版本的浏览器中会被作为脚本执行，被用于钓鱼和欺诈。

防御：

(1) 文件上传的目录设置为不可执行，(web容器无法解析该目录下的文件);

(2) 判断文件类型;

(3) 使用随机数改写文件名和文件路径;

(4) 单独设置文件服务器的域名

1. **html5新增标签可能造成的安全问题：**

html5安全：

(1) video audio标签远程加载一段音频或视频，可能造成xss攻击。

(2) iframe的sandbox(在h5中专门为iframe定义的新属性)，使用后该iframe 标签加载的内容将被视为一个独立的源，其中的脚本将被禁止执行，表单被禁 止提交，插件被禁止加载指向其他浏览对象的的链接也会被禁止。

参数有：

allow-same-origin

allow-top-navigation

allow-forms

allow-scripts

(3) 在h5中为a标签和area标签定义了新的属性：noreferrer,需要自己手动添加

<a href="www.baidu.com" rel="noreferrer"></a>

(4) canvas可以实现破解验证码，通过自动破解验证码可以在浏览器环境中实现在线破解，大大降低了攻击的门槛。

**7. 其他安全问题：**

(1) 浏览器实现的同源策略限制了脚本的跨域请求。(如果在服务器端设置：Access-Control-Allow-Origin: \*)

(2) postMessage--跨窗口传递：【postMessage允许每一个window对象往其他的窗口发送文本消息，从而实现跨窗口的消息传递。这个功能不受同源策略的限制。】

postMessage：页面和其打开的的新窗口的数据传递；多窗口之间消息传递；页面 与嵌套的iframe消息传递；跨域数据传递

接受两个参数：postMessage(data, origin);

data: 要传递的数据(部分浏览器只能处理字符串参数，因此传递参数的时候需要用JSON.stringify()对对象参数序列化)

origin: 字符串参数，指明目标窗口的源[协议+主机+端口号+url],url会被忽略所以可以不写，这样子设置只会将message传递到指定窗口，如果参数为"\*"，可以传递给任何窗口，如果设置为"/",则是传递给当前窗口同源的窗口。

发送数据：

window.onload = function() {

window.frames[0].postMessage('getcolor', 'http://lsoib.com');

}

接收消息：

window.addEventListener('message', function(e) {

if(e.source!=window.parent) {

return;

}

var color = container.style.backgroundColor;

window.parent.postMessage(color, '\*');

}, false);

(3) Web Storage(强大的本地存储功能) 受同源策略的限制，每个域所拥有的信息 只会保存在自己的域下。

分为：sessionStorage和LocalStorage

webStorage就像一个非关系型数据库，由key-value对组成。 window.sessionStorage.setItem(key,value) window.sessionStorage.getItem(key)