LAB11A SQL injection & avoiding

Class: M02
Name: Tran Dang Khoa
Student ID: B2014926

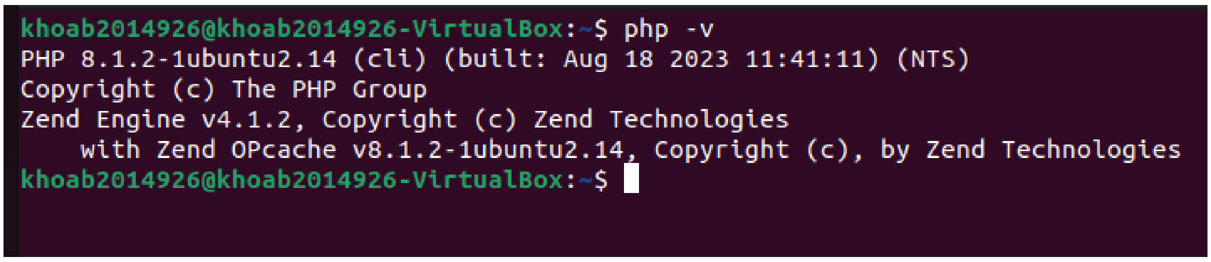
System environment for developing

Resources	Sender(attacker)	Receiver(victim)	Homepage
OS			
IP address			
URL			
Web browser			
CSS language			
Web server			
Web application			
DB server script			
Others			

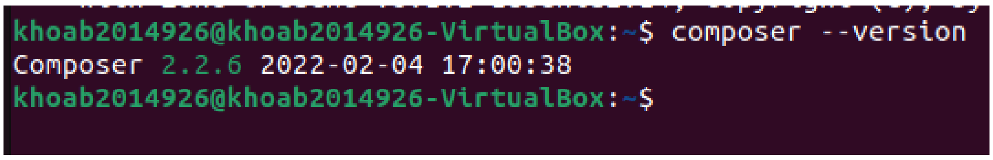
Exercise following scenario on your terminal as far as you do, (if you meet error please describe error message)

① Check the following software installed and enabled on your (pen-test) system:

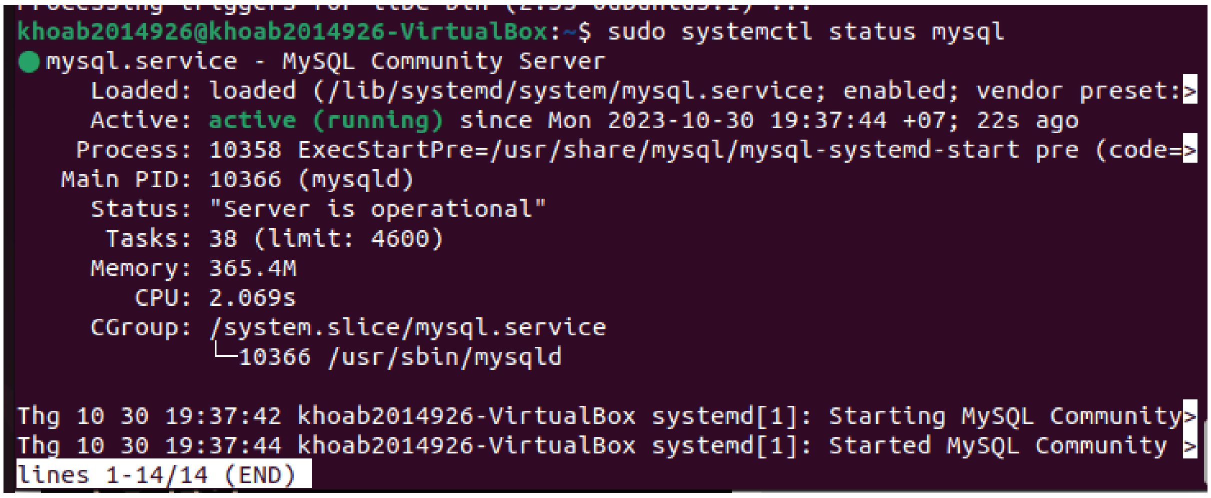
PHP 7, Composer, PHP PDO Extensions for SQLite (and, optionally, for MySQL as well)



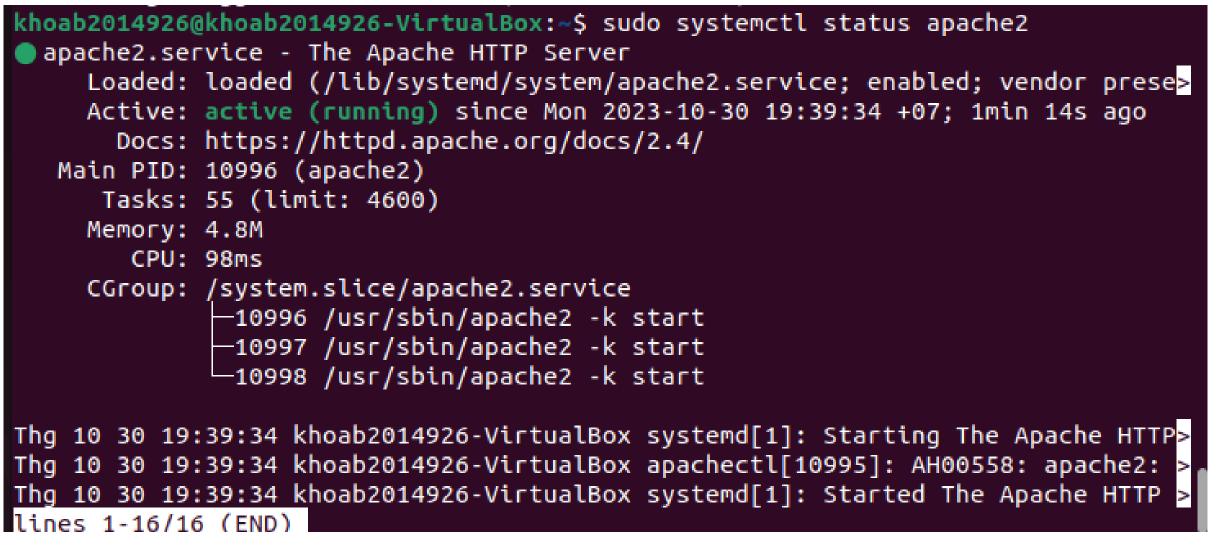
```
khoab2014926@khoab2014926-VirtualBox:~$ php -v
PHP 8.1.2-1ubuntu2.14 (cli) (built: Aug 18 2023 11:41:11) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.1.2, Copyright (c) Zend Technologies
    with Zend OPcache v8.1.2-1ubuntu2.14, Copyright (c), by Zend Technologies
khoab2014926@khoab2014926-VirtualBox:~$
```



```
khoab2014926@khoab2014926-VirtualBox:~$ composer --version
Composer 2.2.6 2022-02-04 17:00:38
khoab2014926@khoab2014926-VirtualBox:~$
```



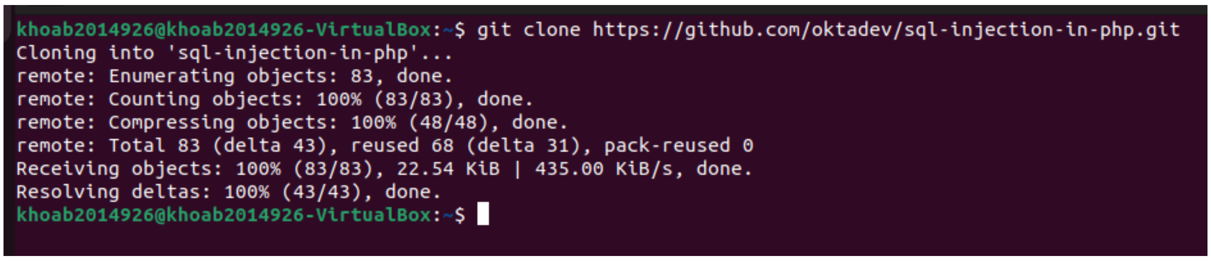
```
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
khoab2014926@khoab2014926-VirtualBox:~$ sudo systemctl status mysql
● mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset:>
     Active: active (running) since Mon 2023-10-30 19:37:44 +07; 22s ago
    Process: 10358 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=>
   Main PID: 10366 (mysqld)
     Status: "Server is operational"
      Tasks: 38 (limit: 4600)
     Memory: 365.4M
        CPU: 2.069s
     CGroup: /system.slice/mysql.service
             └─10366 /usr/sbin/mysqld

Thg 10 30 19:37:42 khoab2014926-VirtualBox systemd[1]: Starting MySQL Community>
Thg 10 30 19:37:44 khoab2014926-VirtualBox systemd[1]: Started MySQL Community >
lines 1-14/14 (END)
```



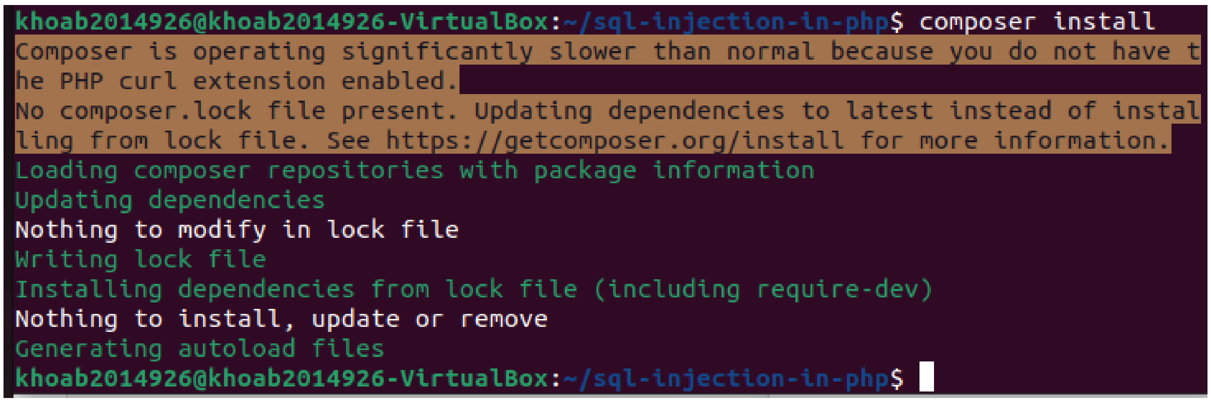
```
khoab2014926@khoab2014926-VirtualBox:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor prese>
     Active: active (running) since Mon 2023-10-30 19:39:34 +07; 1min 14s ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 10996 (apache2)
      Tasks: 55 (limit: 4600)
     Memory: 4.8M
        CPU: 98ms
     CGroup: /system.slice/apache2.service
             ├─10996 /usr/sbin/apache2 -k start
             ├─10997 /usr/sbin/apache2 -k start
             └─10998 /usr/sbin/apache2 -k start

Thg 10 30 19:39:34 khoab2014926-VirtualBox systemd[1]: Starting The Apache HTTP>
Thg 10 30 19:39:34 khoab2014926-VirtualBox apachectl[10995]: AH00558: apache2: >
Thg 10 30 19:39:34 khoab2014926-VirtualBox systemd[1]: Started The Apache HTTP >
lines 1-16/16 (END)
```

② Set up and start the exploitable PHP application

③ Download the source code from GitHub.

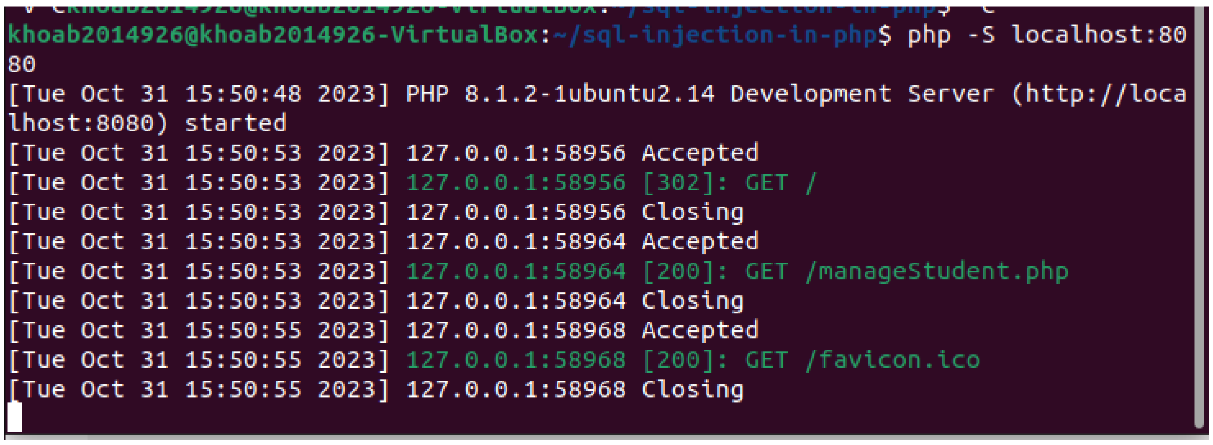


```
khoab2014926@khoab2014926-VirtualBox:~$ git clone https://github.com/oktadev/sql-injection-in-php.git
Cloning into 'sql-injection-in-php'...
remote: Enumerating objects: 83, done.
remote: Counting objects: 100% (83/83), done.
remote: Compressing objects: 100% (48/48), done.
remote: Total 83 (delta 43), reused 68 (delta 31), pack-reused 0
Receiving objects: 100% (83/83), 22.54 KiB | 435.00 KiB/s, done.
Resolving deltas: 100% (43/43), done.
khoab2014926@khoab2014926-VirtualBox:~$ 
```



```
khoab2014926@khoab2014926-VirtualBox:~/sql-injection-in-php$ composer install
Composer is operating significantly slower than normal because you do not have t
he PHP curl extension enabled.
No composer.lock file present. Updating dependencies to latest instead of instal
ling from lock file. See https://getcomposer.org/install for more information.
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
khoab2014926@khoab2014926-VirtualBox:~/sql-injection-in-php$ 
```

④ Execute the PHP built-in server in the port 8080 (you can choose another port if you wish):

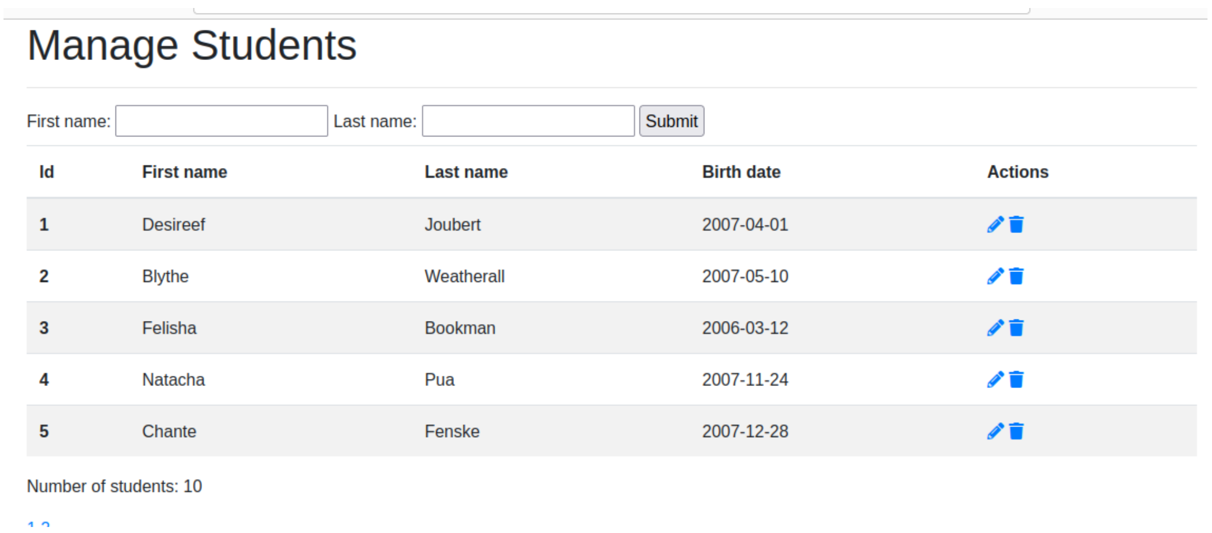


```
khoab2014926@khoab2014926-VirtualBox:~/sql-injection-in-php$ php -S localhost:80
80
[Tue Oct 31 15:50:48 2023] PHP 8.1.2-1ubuntu2.14 Development Server (http://loca
lhost:8080) started
[Tue Oct 31 15:50:53 2023] 127.0.0.1:58956 Accepted
[Tue Oct 31 15:50:53 2023] 127.0.0.1:58956 [302]: GET /
[Tue Oct 31 15:50:53 2023] 127.0.0.1:58956 Closing
[Tue Oct 31 15:50:53 2023] 127.0.0.1:58964 Accepted
[Tue Oct 31 15:50:53 2023] 127.0.0.1:58964 [200]: GET /manageStudent.php
[Tue Oct 31 15:50:53 2023] 127.0.0.1:58964 Closing
[Tue Oct 31 15:50:55 2023] 127.0.0.1:58968 Accepted
[Tue Oct 31 15:50:55 2023] 127.0.0.1:58968 [200]: GET /favicon.ico
[Tue Oct 31 15:50:55 2023] 127.0.0.1:58968 Closing
```

⑤ Visit the vulnerable app from your browser by navigating to http://localhost:8080.

Reference

https://developer.okta.com/blog/2020/06/15/sql-injection-in-php


Manage Students

First name: Last name: Submit

Id	First name	Last name	Birth date	Actions
1	Desireef	Joubert	2007-04-01	
2	Blythe	Weatherall	2007-05-10	
3	Felisha	Bookman	2006-03-12	
4	Natacha	Pua	2007-11-24	
5	Chante	Fenske	2007-12-28	

Number of students: 10

LAB11B SQL injection & avoiding

Class Name Students ID

Class Name Students ID

System environment for developing

Resources	Sender(attacker)	Receiver(victim)	Homepage
OS			
IP address			
URL			
Web browser			
CSS language			
Web server			
Web application			
DB server script			
Others			

Exercise following scenario as far as you do

① Explain how to search PHP code that contains an SQL Injection vulnerability?

- To search for PHP code contains SQL Injection Vulnerability, we can find the code that accepts user input (in thí case, from a GET parameter) and includes it directly in the SQL statement. This allows an attacker to inject SQL in the query, therefore tricking th e application into sending a malformed query to the database.

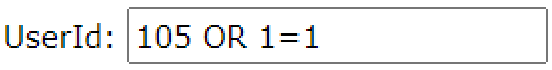
https://www.acunetix.com/blog/articles/prevent-sql-injection-vulnerabilities-in-php-applications/

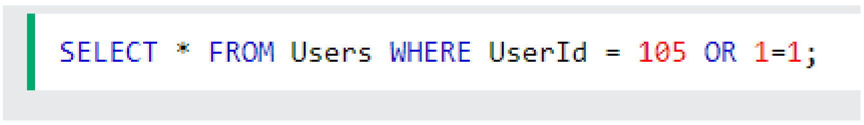
② Survey and explain web application database query model for SQL injection test

- Injection based on 1=1 is always True: if the coder don't prevent wrong input, attacker can enter some input like:

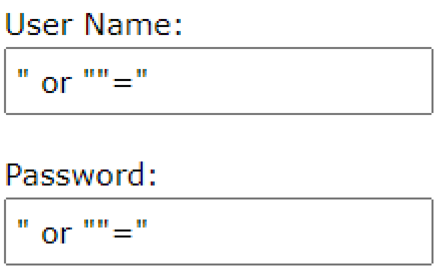
https://www.w3schools.com/sql/sql_injection.asp .

-


UserId: 105 OR 1=1

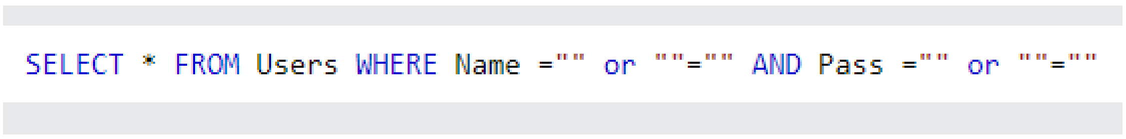

```
SELECT * FROM Users WHERE UserId = 105 OR 1=1;
```

-SQL injection Ba on ""="" is Always True: attacker can enter some input like


User Name:
" or ""="

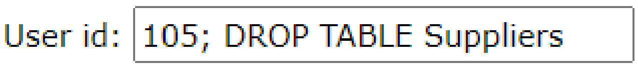
Password:
" or ""="

- The sql statement will become


```
SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""
```

+ A batch of SQL statements is a group of two or more SQL statements, separated by semicolons

+ The attack can delete our database by entering some input like:


User id: 105; DROP TABLE Suppliers

+ The SQL will be:

```
SELECT * FROM Users WHERE UserId = 105; DROP TABLE Suppliers;
```

③ Survey and explain SQL Injection Prevention in PHP

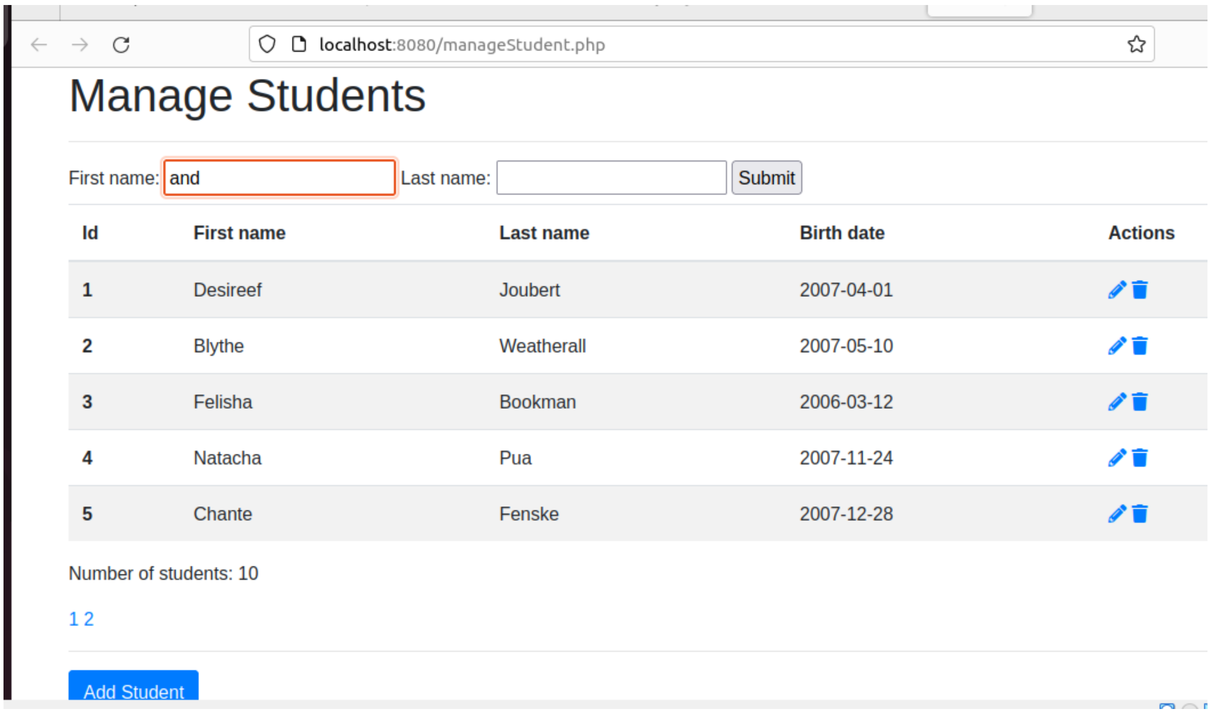
https://www.acunetix.com/blog/articles/prevent-sql-injection-vulnerabilities-in-php-applications/

https://developer.okta.com/blog/2020/06/15/sql-injection-in-php

To prevent SQL Injection

- Santizing your inputs. Don't ever trust incomming data. Before even processing the database query, validate user input – When dealing with SQL queries that contain user input, use prepared statements also known as parameterized queries.
- Do not display SQL errors to the user. If you need to show the user an error, use a generic error massage that does not give away sensitive information.
- Don't rely on client-side input sanitation. An attacker could launch SQL Injection attacks emulating the calls from a browser, using unsanitized data.

④ Set up and start the exploitable PHP, my SQL, test data



localhost:8080/manageStudent.php

Manage Students

First name: and Last name: Submit

Id	First name	Last name	Birth date	Actions
1	Desireef	Joubert	2007-04-01	
2	Blythe	Weatherall	2007-05-10	
3	Felisha	Bookman	2006-03-12	
4	Natacha	Pua	2007-11-24	
5	Chante	Fenske	2007-12-28	

Number of students: 10

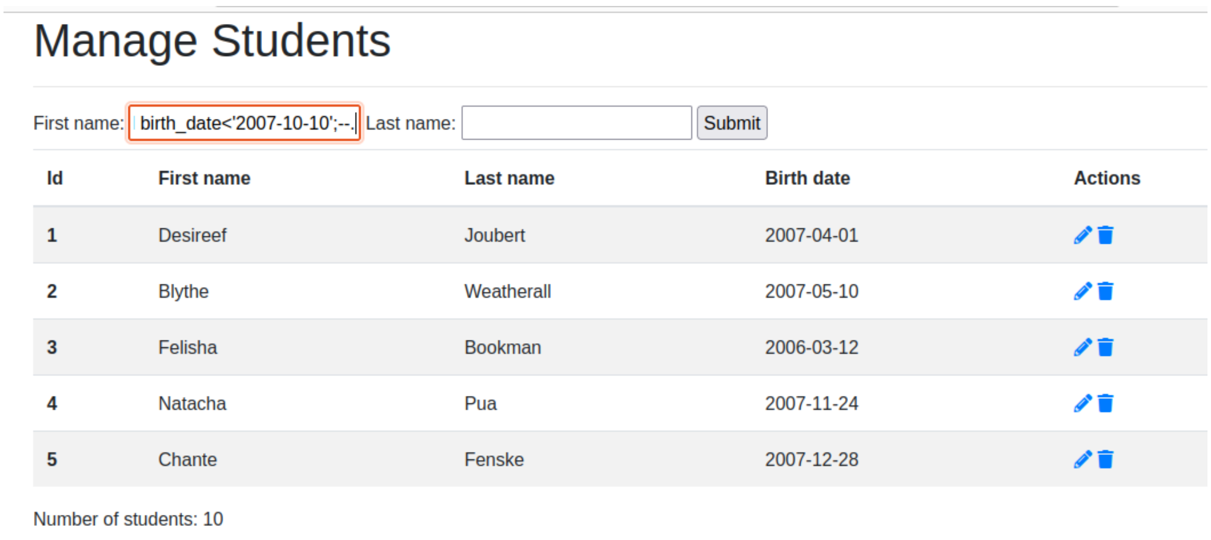
1 2

Add Student

⑤ Select one test run step among 1,2,3 steps for <u>SQL injection & avoiding test (if you meet error please describe error situation)</u>

In the first test try searching for students including the following first name; ‘and birth_date<’2007-10-10’;--.

-



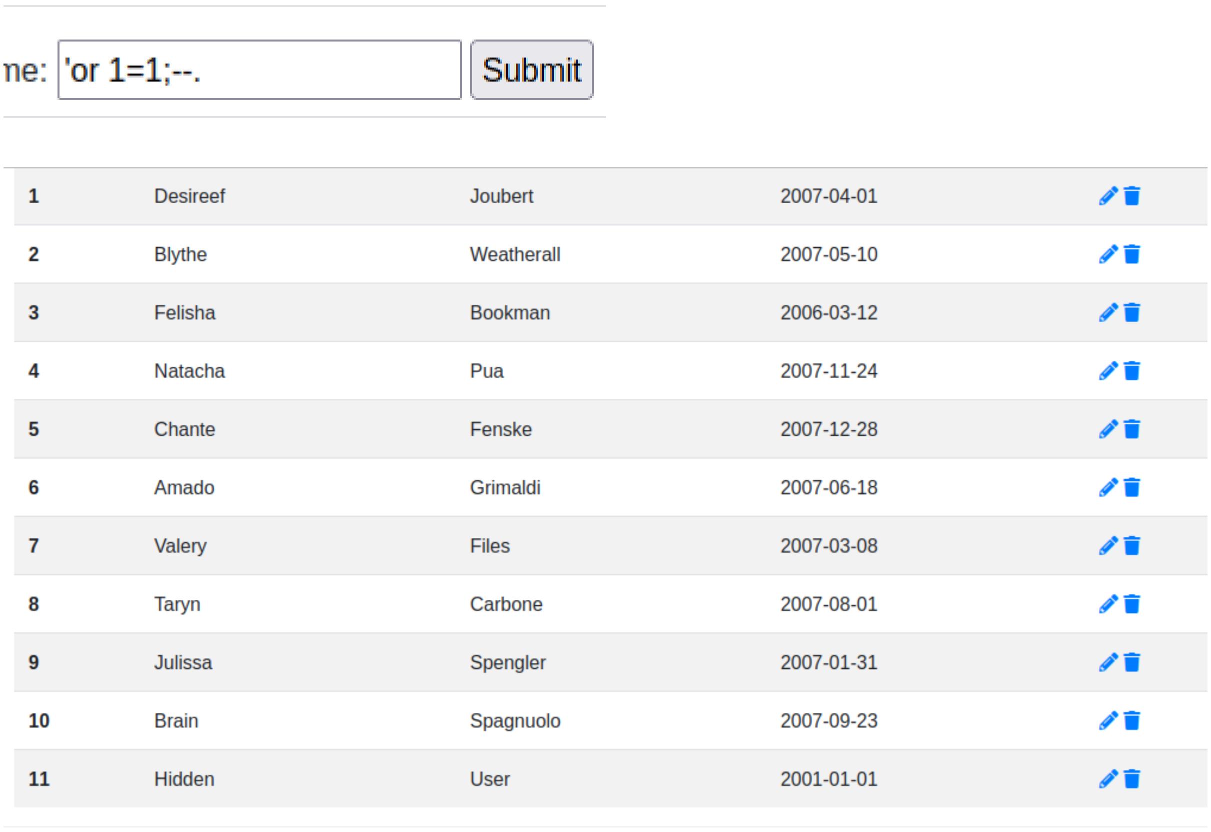
Manage Students

First name: birth_date<'2007-10-10';--. Last name: Submit

Id	First name	Last name	Birth date	Actions
1	Desireef	Joubert	2007-04-01	
2	Blythe	Weatherall	2007-05-10	
3	Felisha	Bookman	2006-03-12	
4	Natacha	Pua	2007-11-24	
5	Chante	Fenske	2007-12-28	

Number of students: 10

- In the next test step: we search for students with the following last name: ‘or 1=1;--



ne: 'or 1=1;--. Submit

1	Desireef	Joubert	2007-04-01	✎🗑
2	Blythe	Weatherall	2007-05-10	✎🗑
3	Felisha	Bookman	2006-03-12	✎🗑
4	Natacha	Pua	2007-11-24	✎🗑
5	Chante	Fenske	2007-12-28	✎🗑
6	Amado	Grimaldi	2007-06-18	✎🗑
7	Valery	Files	2007-03-08	✎🗑
8	Taryn	Carbone	2007-08-01	✎🗑
9	Julissa	Spengler	2007-01-31	✎🗑
10	Brain	Spagnuolo	2007-09-23	✎🗑
11	Hidden	User	2001-01-01	✎🗑