# 4 Command injection

This is the start of the second exercise session.
Log in to this security challenge at overthewire with the credentials:
Username: natas9
Password: W0mMhUcRRnG8dcghE4qvk3JA9lGt8nDl
This webpage has a field for looking up words in a predefined dictionary. This field is not secure.
In the exercise you will use your programming knowledge to attempt a command injection.

Clicking the "View sourcecode", will reveal the code behind the website. Use this knowledge along with the reading material for today (link below) to get the password for the user: natas10

The passwords are stored on the web server folder with the following path:/etc/natas_webpass/(Hint: Commands like grep, ls, cat, curl might be useful)

# 5 SQL injection

Log in to this security challenge at overthewire with the credentials:
Username: natas14
Password: Lg96M10TdfaPyVBkJdjymbllQ5L6qdl1
This webpage is a login page. In the exercise you will use your programming knowledge to attempt a SQL-Injection.Clicking the "View sourcecode", will reveal the code behind the website. Use this knowledgealong with the reading material for today (link below) to get the password for the user:natas15(Hint: Unsanitized user inputs are dangerous)

## Exercise 4:

### Step 1) Understanding the source code.

The PHP code effectively has the following logic:

- A variable named $key: this variable is initialized to a blank string.
- An if statement that looks for a variable in the request named “needle”, and applies the value to the “key” variable.
- And an if statement that performs an action if the key is not an empty string. (the `passthru` function is called).

In essence, it will call the "passthru" function on, using grep to find the keyword in a dictionary text file. We should familiarize ourselves with the function. Here is the PHP function documentation. It works much like the `os.system('..command')` from the previous exercise.

### Step 2) Injecting the correct command to steal the password.

> The executed command is: `grep -i $key dictionary.txt` and the `$key` will be replaced by our input.
>
> We can terminate the command and add our own, by inserting `; ls`. This will show us the files in the folder.
>
> We can move around the server by providing `../` paths. The path for the password is provided in the assignment: it is `:/etc/natas_webpass/`.
>
> We can move out to the root by using multiple upwards paths chained together like so: `../../`. We can now find the password files with the following command:
>
> ```
> ; ls ../../../../etc/natas_webpass
> ```

**SOLUTION:**

Finally, we can read the password for user natas10 by utilizing the `cat` and our relative path command:

```
;cat ../../../../etc/natas_webpass/natas10
```

## Exercise 5:

> Following the same approach as before, we see that the code executes an SQL select-statement based on our inputs. It then later checks if the select returned any results - and if it did, we are given the password for the user `natas15`.
>
> We first break down the SQL statement. It looks something like this:
>
> `SELECT * from users where username="..input" and password="..input"`

**SOLUTION:**

The inputs are not sanitized, and we can use them to alter the intent SQL statement. One way to go about it would be to make the query select data from all rows in the table, by creating a tautalogy as the filter condition:

`"or 1=1 #`

> Note that the `#` is a MySQL comment. The executed SQL statement effectively becomes:
>
> `SELECT * from users where username="" 1=1 #..`