CS-C3130 Information Security (2021)
Exercise 3

Web Insecurity

In this exercise, you will continue to hack the Potplants web site and also target an Internet-of-Things device. Everyone should be able complete at least part A, while part B requires some familiarity with server-side JavaScript.

As before, the exercise launcher is running at `https://infosec1.vikaa.fi/`. Note that you will launch Potplants in a new VM for each week's exercise. The web sites for the different exercise rounds look the same but have different port numbers and also slightly different vulnerabilities.

Part A. Bypassing client-side checks (10 p)

Client-side code is JavaScript that runs on the browser. It is common to implement basic correctness checks on user input in the client-side code. However, any checks that are important for security or reliability of the online service must be repeated on the server side. Otherwise, things *will* go wrong.

In the first part, your goal is to bypass client-side integrity checks on user input.

1. The Potplants page has a form for adding new plants. The available plant colors are limited to a small list in a drop-down menu. As you may have noticed, other users on the site have given their plants more imaginative colors. Your goal is to add a new plant with the color "*TiK black*".

2. When signing up as a new user, there is a 20 character limit on the username. Annoyingly, this prevents you from signing up with a long email address as the username. Your goal is to register a username that is longer than the limit.

There are many professional tools, such as the OWASP Zed and the Burp Suite, for penetration testing web sites. They allow you to modify data sent between the web client and server. However, this exercise is easiest to do in the web-browser developer mode (try `Ctrl-Shift-I`, `Cmd-Opt-I`, or `F12`), which allows you to modify data in the web page's DOM model. You can, for example, edit form contents or structure before submitting the form to the server.

Part B. Server-side poisoning (10 p)

The developer of the Potplants web site has used the notorious `eval()` function, which executes code given to it as a string argument. Functions that take code in string format and execute it are generally dangerous. These functions include `eval()` in interpreted languages such as JavaScript and Python as well as `system()`, which executes a shell command in some languages. By using them, the developer basically gives up on all security.

You may have noticed the Potplants feature for sorting the plants list. The server — unwisely — makes clever use of the `eval()` function to implement the sorting. The server-side JavaScript looks something like this:

```
// Field name from drop-down selection
var field = req.body.sortField;
// Sort the plants
plants.sort(function(a, b) {
    var av, bv;
    with (a) { av = eval(field); };
    with (b) { bv = eval(field); };
    return compare(av, bv);
});
```

This vulnerability allows you to inject code into the argument of `eval()`, which pretty much enables you to take over the server. You first need to find a way to receive output from the injected code. Then, your goal is to find a directory and file on the server that contains secrets and copy the contents to the exercise launcher.

The Potplants website is written with *Node.js* and the Express framework. The language is JavaScript, but since the code runs on the server and not in the web browser, the available APIs are quite different. For example, `alert()` will not work because it belongs to client-side JavaScript; you will have to use a server-side API for outputting the stolen data.

Please attack only your own VM. Perform the attack from the *student* account because we have artificially blocked the code injection from other accounts.

The OWASP NodeGoat Tutorial has some helpful examples. You can also install the OWASP WebGoat or NodeGoat for more exercises.