CSci 4271W (011 and 012 Sections) Lab Instructions

Lab 13 **April 28th, 2024**

Ground Rules. You may choose to complete this lab in a group of up to three students. Before you leave the lab, **make sure you have submitted to Gradescope, you included all group members on the submission, and the autograder found all required files!**

1 Cross-Site Scripting (XSS)

As we discussed in lecture, a Cross-Site Scripting (XSS) attack happens when an attacker is able to inject JavaScript into the code of a web page. For today's lab, we'll be using a tool designed by Google's web security team, the XSS game at xss-game.appspot.com. This game has six levels; we'll hope to complete the first three for the lab (if you're having fun, though, you can certainly keep playing on. Do note: the cake is a lie.)

Let's record your progress in the game in a Markdown (basically plaintext) file named `xssgame.md`. Put your name and the date at the top of the file and you can make a section for each level by starting a new line with just the text `# Level 1`, `# Level 2`, or `# Level 3`. Here we go:

2 Level 1

Level 1 is an example of a "reflected XSS" attack, like the first example we saw in class, in which a user's input to a form can be "reflected" as a URL parameter to inject a script. Try experimenting with the input to the form, and see what happens when you embed HTML in the input, like `<br><i>query</i>`. Throughout this lab, we'll consider it a success if we can get the in-game browser to execute the simple JavaScript code `alert('xss');`. Record your inputs and observations until you find an input that causes the script to run. Explain why your input works in `xssgame.md` and move to the next level.

(Note that you can toggle showing the source code for the page and on the server side with the "Target code (toggle)" link underneath the browser frame, and that the game will give you some hints if you're getting stuck.)

3 Level 2

Level 2 is an example of a "stored XSS" attack. Here users can enter content to be stored and displayed to other hypothetical users of a simulated message board. Try entering some status messages, again include various HTML formatting, and see what happens. Now try the input you used in the previous level. You'll see that it doesn't work; this is because messages are displayed by a method that doesn't execute scripts directly. (Again, record your attempts and observations.)

In order to pass this level, we'll need a slightly more sophisticated way to sneak in a script, which is an "event." Events are called by the browser in various html elements when certain conditions are met. If you gave up and read the hints, you'll see that one such event is `onerror`, which is called whenever the browser can't render the element specified by a tag; like trying to include an image file that doesn't exist. (e.g. `http://localhost/nope.jpg`). When you find the right input, record and explain it in `xssgame.md`, and move on to level 3.

4 Level 3

This example is a more complicated reflected attack. You'll manipulate the URL directly; see what happens to the URL bar in the "in-game browser" when you click on the various tabs. If you toggle the code for the page, you'll see in the `window.onload` function, whatever comes after the `#` is passed as an argument to the `chooseTab` function, and this function writes some html to the "tabContent" frame.

See if you can work out for yourself what to put after the hash sign in the URL to pop an `alert`. You might find it helpful to use "Inspect" in the (Chrome) context (i.e. right-click) menu to see the source code for the in-game browser window to see what effect your inputs are having on the source. (Another thing you might find useful: HTML comments are opened by `<!--` and closed by `-->`.) Record your attempts and observations, and explain why your attack works, in `xssgame.md`.

5 Levels 4-6

You aren't required to solve these for the lab, but if you've got time and are having fun, you are certainly welcome to do so!

5.1 All done!

Once you've solved Level 3, you're done with Lab 13! Save the contents of your `xssgame.md` file so you can submit it to the Lab 13 assignment on Gradescope. Make sure you include all of the members of your group!

Once you've submitted your file, the autograder will test to make sure the proper file was submitted, and notify you if anything went wrong, within a few minutes. (We'll manually check your submissions after the deadline.)

Congratulations, you've finished Lab 13!