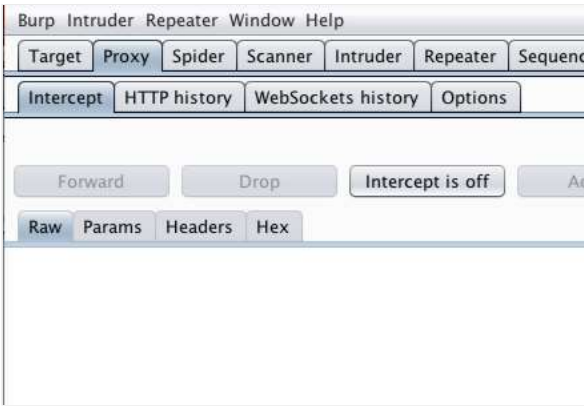
Using Burp to Manually Test for Stored XSS

Stored cross-site scripting vulnerabilities arise when data originating from any tainted source is copied into the application's responses in an unsafe way. attacker can use the vulnerability to inject malicious JavaScript code into the application, which will execute within the browser of any user who views the relevant application content. The attacker-supplied code perform a wifde variety of actions, such as stealing the victim's session token or login credential performing arbitrary actions on the victims behalf, and logging their keystrokes.

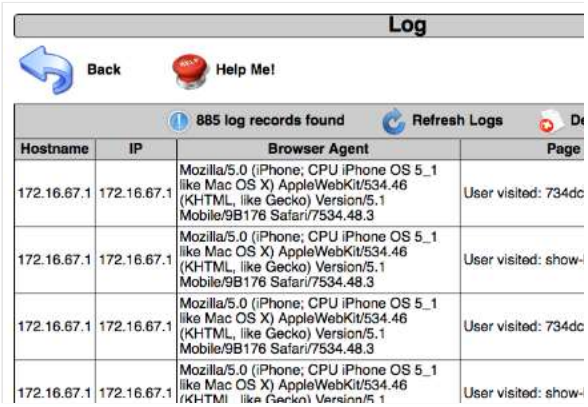
In this tutorial we will demonstrate how to generate a proof-of-concept stored XSS exploit. The example uses a version of "Mutillidae" taken from OWASI Broken Web Application Project. Find out how to download, install and use this project.

First, ensure that Burp is correctly configured with your browser.

With intercept turned off in the Proxy "Intercept" tab, visit the web application you are testing in your browser.



Visit the page of the website you wish to test for XSS vulnerabilities.



Return to Burp.

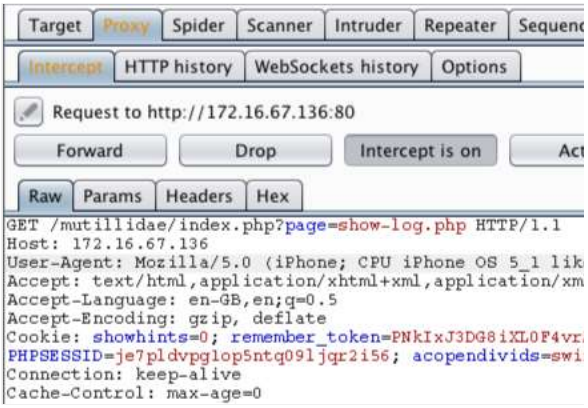
In the Proxy "Intercept" tab, ensure "Intercept is on".

Submit a request by refreshing the web application in your browser. The request will be captured by Burp. You can view the HTTP request in the Proxy "Intercept" tab.

You can also locate the relevant request in various Burp tabs without having to use the intercept function, e.g. requests are logged and detailed in the "HTTP history" tab within the "Proxy" tab.

Right click anywhere on the request to bring up the context menu.

Click "Send to Repeater"



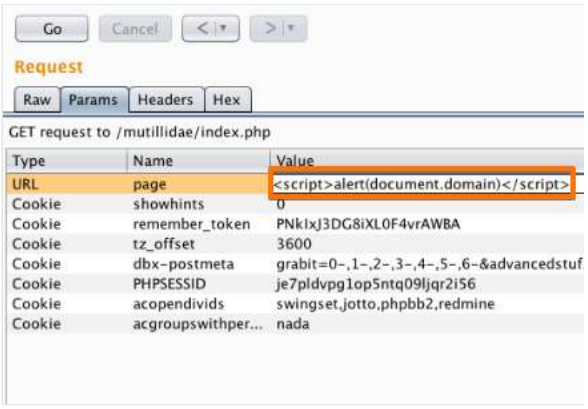
Go to the "Repeater" tab.

Here we can input various XSS payloads in to the input field of a web application.

We can test various inputs by editing the "Value" of the appropriate parameter in the "Raw" or "Params" tabs.

In this example we have used a payload in an attempt to provide a proof of concept pop up in our browser.

Click "Go".

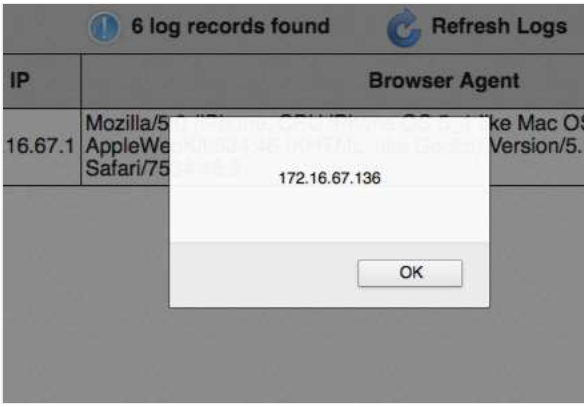


In the Proxy "Intercept" tab, ensure "Intercept is on".

Then return to your browser and refresh the page, thereby sending a second request to the application. This time simulating the user/victim.

If the payload fires, then the attack has been successful.

In this example we have been able to produce a proof of concept pop-up. The application is vulnerable to stored XSS.

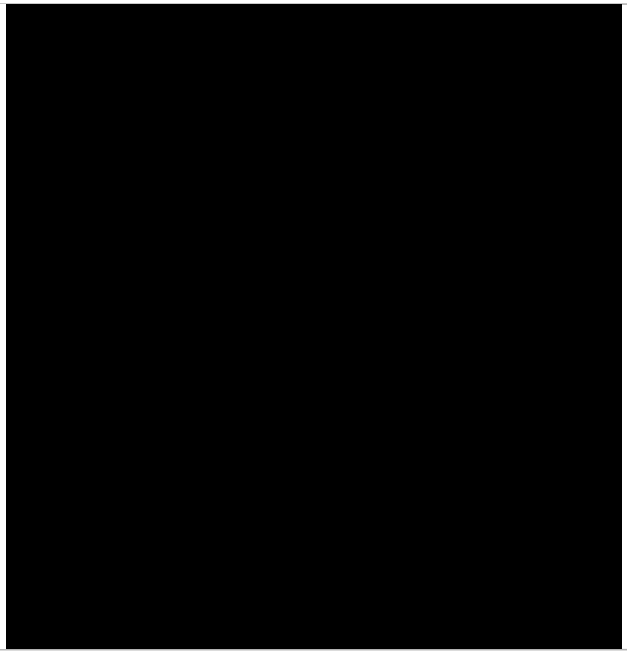


Why We Alert document.domain

In this example, we have used a variation on the standard proof-of-concept attack string for detecting an XSS vulnerability. For example:

```
"><script>alert(document.domain)</script>
```

One reason for using a payload of this nature is that a pop up alert gives us a strong visual proof of concept, which is easy to identify and explain in a rep
Using document.domain within the alert also demonstrates unequivocally which domain you are able to execute arbitrary JavaScript on.



0:00 / 2:23

Related articles:

Getting started with Burp Proxy

Using Burp Repeater