Using Burp Scanner to Test for DOM-Based XSS

DOM-based XSS (sometimes referred to as DOM-based JavaScript injection) vulnerabilities arise when a client-side script within an application's respon data from a controllable part of the DOM (for example, the URL), and executes this data as JavaScript. An attacker may be able to use the vulnerability t construct a URL which, if visited by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in t context of that user's session with the application. The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes.

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

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



One way to test a web application for potential DOM XSS vulnerabilities is by using Burp Scanner. By applying certain options, Burp Scanner will passively scan for DOM XSS vulnerabilities.

Go to the Scanner "Options" tab and locate the "Static Code Analysis" options.

By default, Burp only performs static analysis for bugs like XSS during active scanning, but you can also enable this for passive scanning.

Target Proxy Spider Scanner Intruder Repeater Sequence
Results Scan queue Live scanning Options

Static Code Analysis

These settings control the types of scanning that will include and so it may be desirable to restrict static analysis to key tar

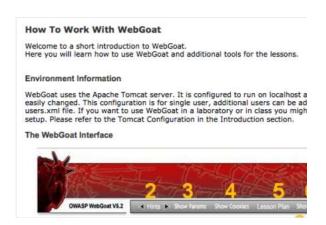
Active scanning only

Active and passive scanning

Don't perform static code analysis

Maximum analysis time per item (seconds): 120

Now, visit the page of the website you wish to test for XSS vulnerabilities.





Burp Scanner will now passively detect any DOM XSS vulnerabilities as you browse.

Go to the Scanner "Results" tab to view any potential vulnerabilities.

The Scanner "Advisory" tab provides further information on the issue.

You can review the JavaScript in the "Issue detail" to discern the plausibility of the vulnerability.

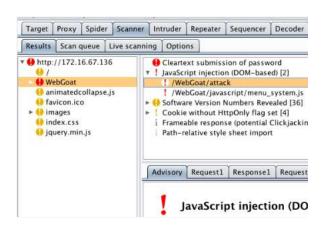
In this example, the unfiltered input from a URL in to an eval() statement is worth further investigation.

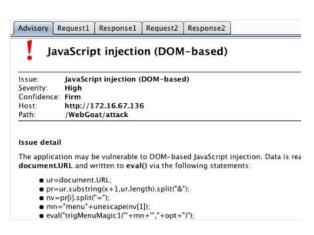
In the "Response 1" tab you can view the vulnerable JavaScript file being imported.

In the "Response 2" tab you can see how a value is read from a "document.url" file and written to "eval()".

The information in these tabs provides details of what will be needed when attempting to produce a valid payload for a proof of concept.

In this example we can discern that the parameter name must be equal to the param variable when the function is called. This satisfies the "if" statement.





```
Advisory Request1 Response1 Request2 Response2

Raw Headers Hex

g=m2[x];ts=im.replace("_open","");ts=ts.replace("_over
if(g.open){j=ts.lastIndexOf(".");
nu=ts.substring(0,j)+"_open"+ts.substring(j,ts.length)
}else{nu=ts;}e.src=nu;break;}}

function trigMMlurl(param,opt){
  var ur,x,i,nv,mn,pr=new Array();
  ur=document.URL;x=ur.indexOf("?");
  if(x>1){pr=ur.substring(x+1,ur.length).split("&");
  for(i=0;i<pr.length;i++){nv=pr[i].split("=");
  if(nv.length>0){if(unescape(nv[0])==param){
  mn="menu"+unescape(nv[1]);
  eval("trigMenuMagicl("+mn+"',"+opt+")");}}}

document.mm10=true;
```

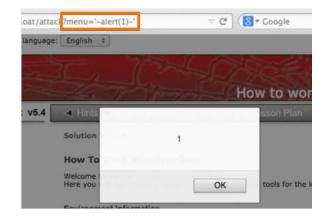
The function called is "trigMMlurl".

We can then discern that the parameter name is "menu".

We can also use single quotation marks to break out of the function.

The payload is added to the URL in the address bar of your browser.

The payload we have used to produce the proof of concept is **?menu='-alert(1)-**.



Related articles:

Getting started with Burp Proxy

Getting started with Burp Scanner

Web vulnerability scanner
Burp Suite Editions
Release Notes

Burp Suite

Vulnerabilities Cross-site scripting (XSS) SQL injection Cross-site request forgery XML external entity injection Directory traversal Server-side request forgery

Customers

Organizations Testers Developers

Company

About

PortSwigger News Careers Contact Legal Privacy Notice

Insights

Web Security Academy Blog Research The Daily Swig





© 2020 PortSwigger Ltd