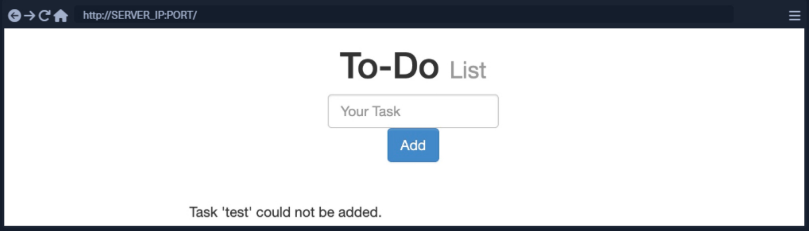
Reflected XSS

There are two types of `Non-Persistent XSS` vulnerabilities: `Reflected XSS`, which gets processed by the back-end server, and `DOM-based XSS`, which is completely processed on the client-side and never reaches the back-end server. Unlike Persistent XSS, `Non-Persistent XSS` vulnerabilities are temporary and are not persistent through page refreshes. Hence, our attacks only affect the targeted user and will not affect other users who visit the page.

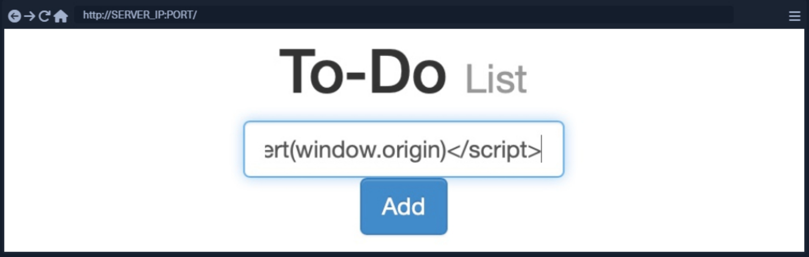
`Reflected XSS` vulnerabilities occur when our input reaches the back-end server and gets returned to us without being filtered or sanitized. There are many cases in which our entire input might get returned to us, like error messages or confirmation messages. In these cases, we may attempt using XSS payloads to see whether they execute. However, as these are usually temporary messages, once we move from the page, they would not execute again, and hence they are `Non-Persistent`.

We can start the server below to practice on a web page vulnerable to a Reflected XSS vulnerability. It is a similar `To-Do List` app to the one we practiced with in the previous section. We can try adding any `test` string to see how it's handled:

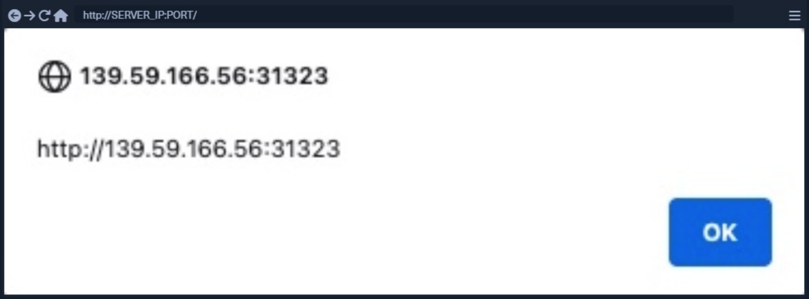


Task 'test' could not be added.

As we can see, we get `Task 'test' could not be added.`, which includes our input `test` as part of the error message. If our input was not filtered or sanitized, the page might be vulnerable to XSS. We can try the same XSS payload we used in the previous section and click `Add`:



Once we click `Add`, we get the alert pop-up:



In this case, we see that the error message now says `Task '' could not be added.`. Since our payload is wrapped with a `<script>` tag, it does not get rendered by the browser, so we get empty single quotes `''` instead. We can once again view the page source to confirm that the error message includes our XSS payload:

Code: html

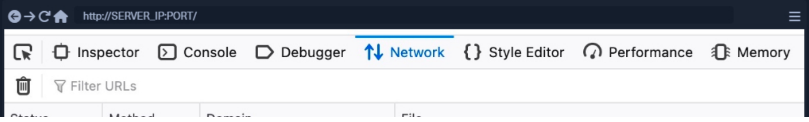
```html
<div></div><ul class="list-unstyled" id="todo"><div style="padding-left:25px">Task '<script>alert(window.origin)</scri
```

As we can see, the single quotes indeed contain our XSS payload `'<script>alert(window.origin)</script>'`.

If we visit the `Reflected` page again, the error message no longer appears, and our XSS payload is not executed, which means that this XSS vulnerability is indeed `Non-Persistent`.

`But if the XSS vulnerability is Non-Persistent, how would we target victims with it?`

This depends on which HTTP request is used to send our input to the server. We can check this through the Firefox `Developer Tools` by clicking [`CTRL+Shift+I`] and selecting the `Network` tab. Then, we can put our `test` payload again and click `Add` to send it:

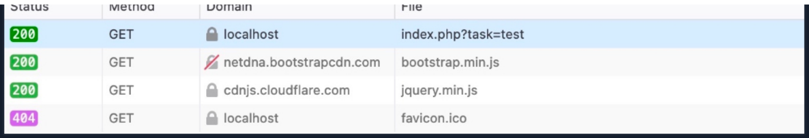


My Workstation

OFFLINE

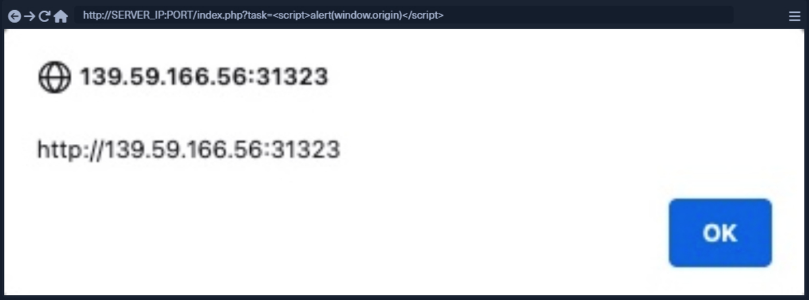
Start Instance

∞ / 1 spawns left



Status	Method	Domain	File
200	GET	localhost	index.php?task=test
200	GET	netdna.bootstrapcdn.com	bootstrap.min.js
200	GET	cdnjs.cloudflare.com	jquery.min.js
404	GET	localhost	favicon.ico

As we can see, the first row shows that our request was a `GET` request. `GET` request sends their parameters and data as part of the URL. So, `to target a user, we can send them a URL containing our payload`. To get the URL, we can copy the URL from the URL bar in Firefox after sending our XSS payload, or we can right-click on the `GET` request in the `Network` tab and select `Copy>Copy URL`. Once the victim visits this URL, the XSS payload would execute:



```
http://SERVER_IP:PORT/index.php?task=<script>alert(window.origin)</script>
```

Connect to Pwnbox
Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location
UK 137ms

Terminate Pwnbox to switch location

Start Instance
∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Cheat Sheet

Answer the question(s) below to complete this Section and earn cubes!

Target(s): Click here to spawn the target system!

+ 2 To get the flag, use the same payload we used above, but change its JavaScript code to show the cookie instead of showing the url.

HTB{r3fl3c73d_b4ck_2_m3}

Submit Hint

Previous Next Mark Complete & Next