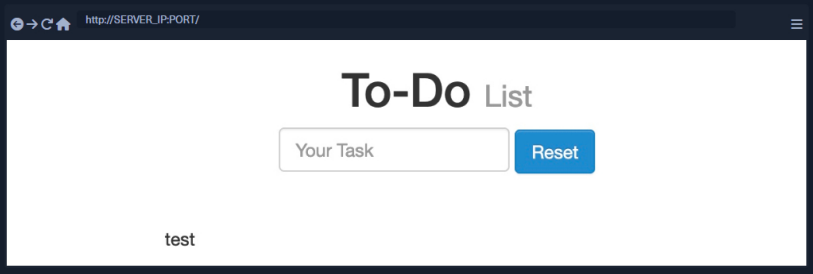
Stored XSS

Before we learn how to discover XSS vulnerabilities and utilize them for various attacks, we must first understand the different types of XSS vulnerabilities and their differences to know which to use in each kind of attack.

The first and most critical type of XSS vulnerability is `Stored XSS` or `Persistent XSS`. If our injected XSS payload gets stored in the back-end database and retrieved upon visiting the page, this means that our XSS attack is persistent and may affect any user that visits the page.

This makes this type of XSS the most critical, as it affects a much wider audience since any user who visits the page would be a victim of this attack. Furthermore, Stored XSS may not be easily removable, and the payload may need removing from the back-end database.

We can start the server below to view and practice a Stored XSS example. As we can see, the web page is a simple `To-Do List` app that we can add items to. We can try typing `test` and hitting enter/return to add a new item and see how the page handles it:



As we can see, our input was displayed on the page. If no sanitization or filtering was applied to our input, the page might be vulnerable to XSS.

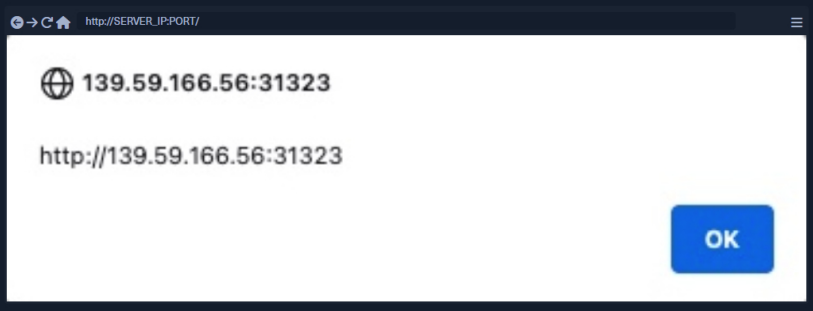
XSS Testing Payloads

We can test whether the page is vulnerable to XSS with the following basic XSS payload:

Code: html

```
<script>alert(window.origin)</script>
```

We use this payload as it is a very easy-to-spot method to know when our XSS payload has been successfully executed. Suppose the page allows any input and does not perform any sanitization on it. In that case, the alert should pop up with the URL of the page it is being executed on, directly after we input our payload or when we refresh the page:



As we can see, we did indeed get the alert, which means that the page is vulnerable to XSS, since our payload executed successfully. We can confirm this further by looking at the page source by clicking `[CTRL+U]` or right-clicking and selecting `View Page Source`, and we should see our payload in the page source:

Code: html

```
<div></div><ul class="list-unstyled" id="todo"><ul><script>alert(window.origin)</script>
</ul></ul>
```

Tip: Many modern web applications utilize cross-domain IFrames to handle user input, so that even if the web form is vulnerable to XSS, it would not be a vulnerability on the main web application. This is why we are showing the value of `window.origin` in the alert box, instead of a static value like `1`. In this case, the alert box would reveal the URL it is being executed on, and will confirm which form is the vulnerable one, in case an IFrame was being used.

As some modern browsers may block the `alert()` JavaScript function in specific locations, it may be handy to know a few other basic XSS payloads to verify the existence of XSS. One such XSS payload is `<plaintext>`, which will stop rendering the HTML code that comes after it and display it as plaintext. Another easy-to-spot payload is `<script>print()</script>` that will pop up the browser print dialog, which is unlikely to be blocked by any browsers. Try using these payloads to see how each works. You may use the reset button to remove any current payloads.

To see whether the payload is persistent and stored on the back-end, we can refresh the page and see whether we get the alert again. If we do, we would see that we keep getting the alert even throughout page refreshes, confirming that this is indeed a `Stored/Persistent XSS` vulnerability. This is not unique to us, as any user who visits the page will trigger the XSS payload and get the same alert.

Cheat Sheet
Go to Questions

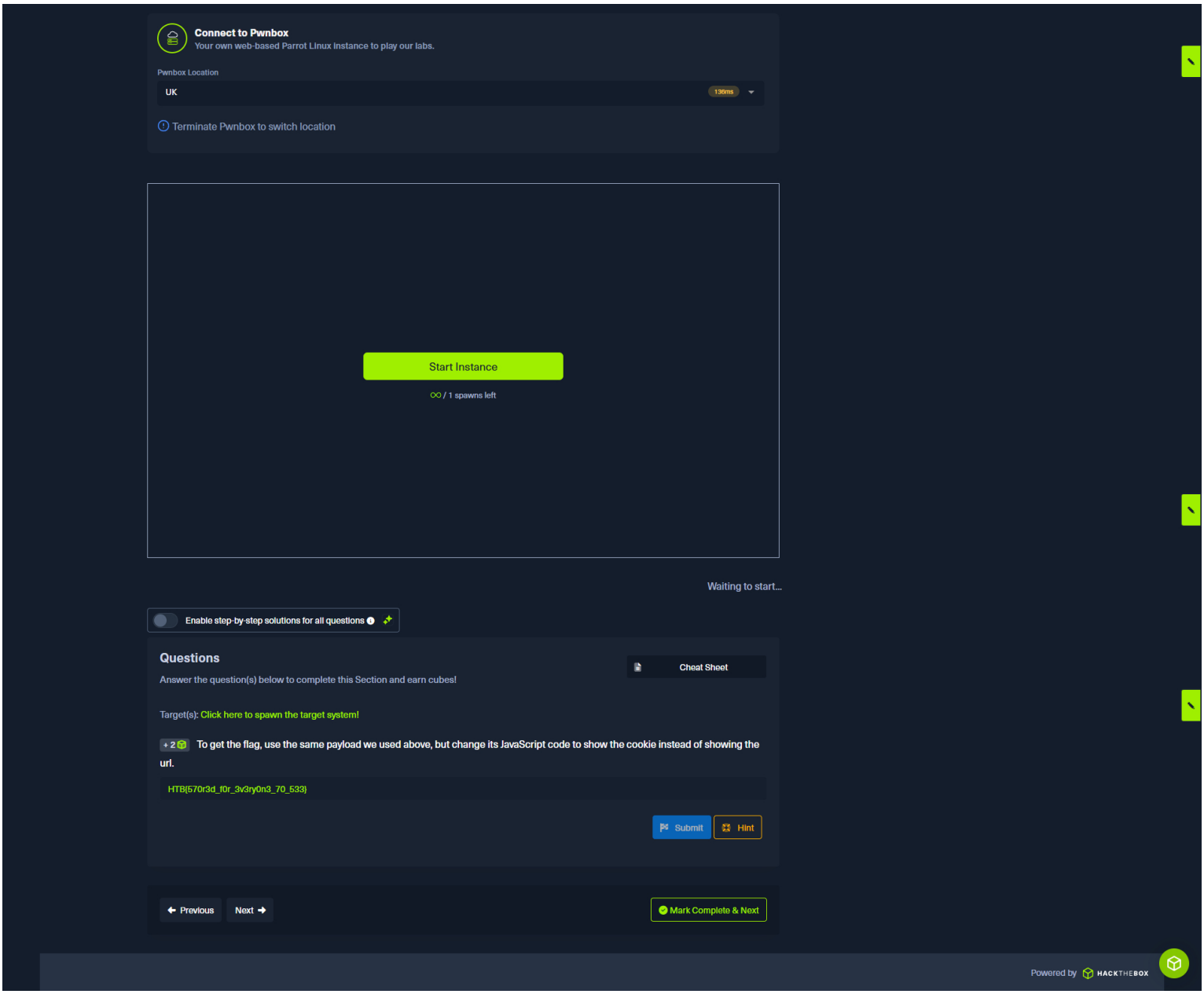
Table of Contents

XSS Basics
Intro to XSS
Stored XSS
Reflected XSS
DOM XSS
XSS Discovery

XSS Attacks
Defacing
Phishing
Session Hijacking

XSS Prevention
XSS Prevention

Skills Assessment
Skills Assessment


Connect to Pwnbox
Your own web-based Parrot Linux instance to play our labs.
Pwnbox Location
UK
136ms
Terminate Pwnbox to switch location
Start Instance
∞ / 1 spawns left
Waiting to start...
Enable step-by-step solutions for all questions
Questions
Answer the question(s) below to complete this Section and earn cubes!
Cheat Sheet
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{570r3d_f0r_3v3ry0n3_70_533}
Submit
Hint
Previous
Next
Mark Complete & Next
Powered by HACKTHEBOX