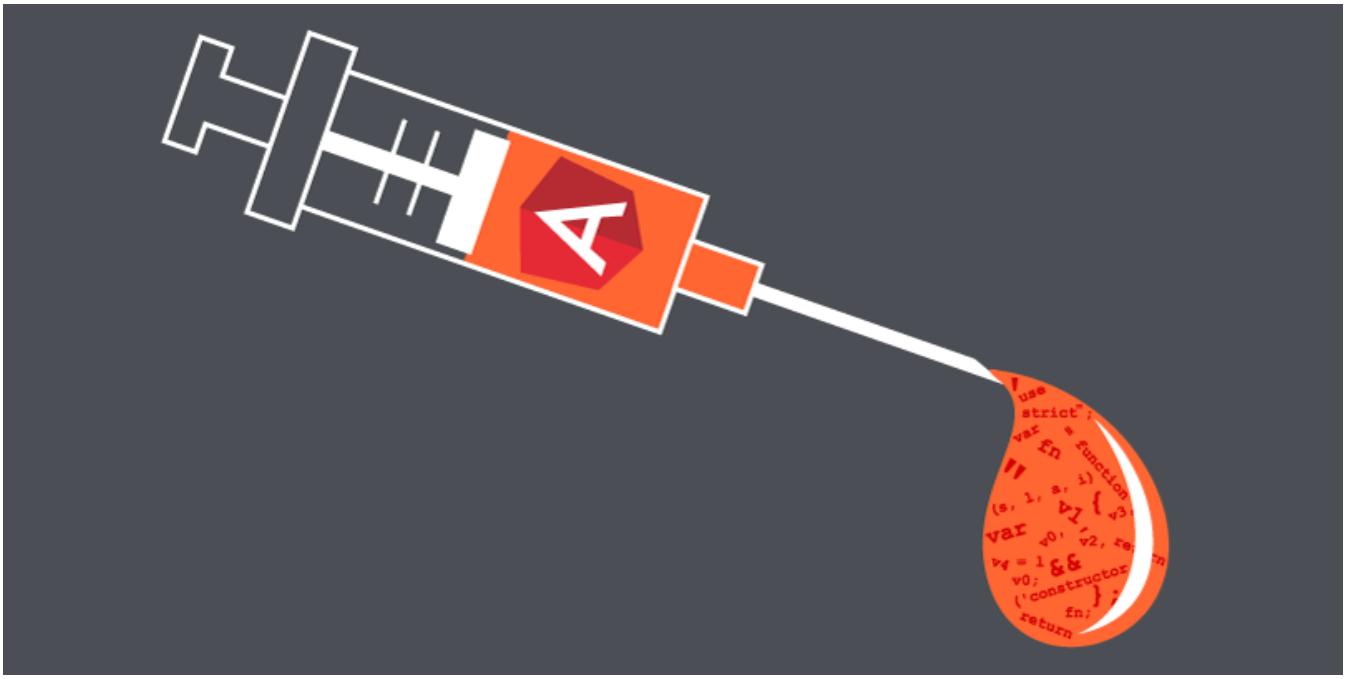
Guidance to Cross-Site Scripting for beginners- I: Reflected XSS

Cyber Army
Jun 30 · 5 min read

Before diving into how XSS could be injected or even before starting with anything, let us first explore a little bit of its definition and types.

> ***What is it?***
>
> ***XSS(Cross-Site Scripting)*** *is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. There are three main types of XSS which are:*

- **Reflected XSS:** Malicious code comes from HTTP Request
- **Stored XSS:** malicious script comes from the website's database.
- **DOM-Based XSS:** vulnerability exists in client-side code rather than the er-side code.



Here, in this article, we will be mainly focusing on the first type of XSS, which is Reflected XSS. Then, in our next article, we will discuss further on the other types. Also, I will be showing you a brief example using **Mutillidae** and ***DVWA.*** *You can use any other labs as per your preference.*

Reflected XSS

Reflected XSS? You can clearly guess by the name itself. **"Reflected",** that means something needs to be reflected. Exactly, XSS Vulnerability arises when the injected script is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request.

Let us now start with discovering basic Reflected XSS

Low-Security Level

I am going to show you how it is discovered using DVWA. Firstly, I will show you with low-security level, then I will help you explore further.

Step 1: Open DVWA. Change the security level into Low.

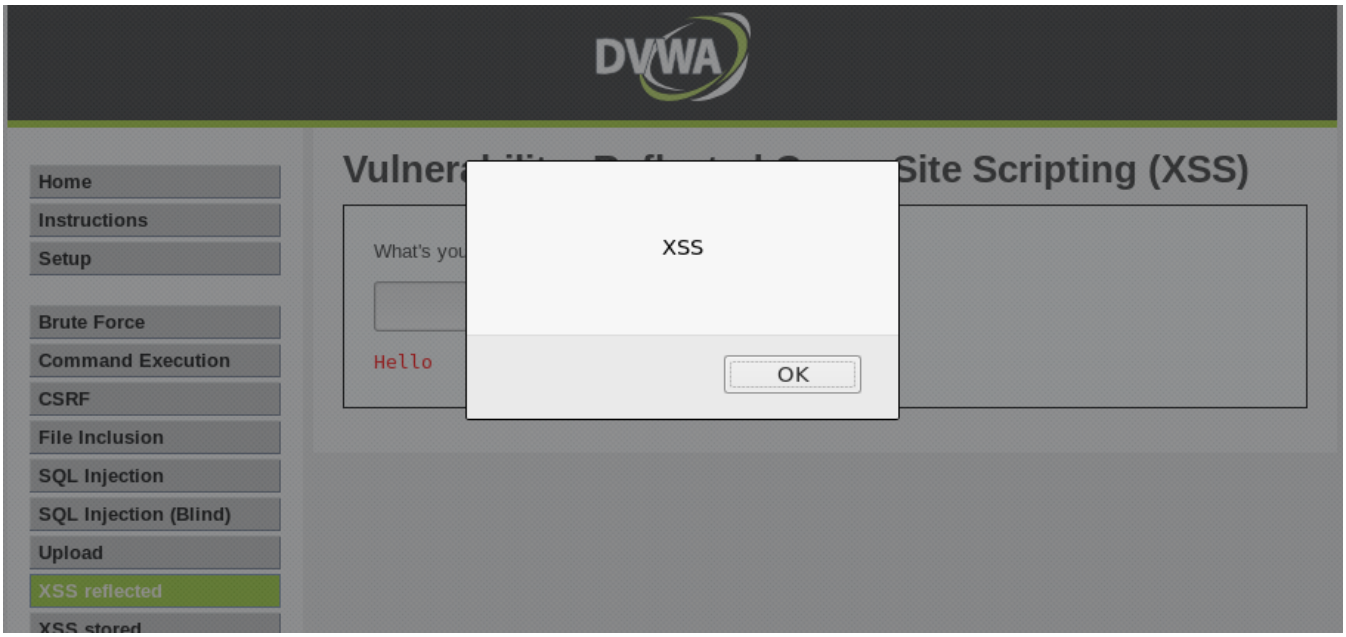
Step 2: Go to Reflected XSS

Step 3: You can see the text box. When you enter your name, it will display something like, "Hello XXX".

Step 4: Now, let's try to inject some XSS code and see if it's executed. I will be using a very simple script. In the text box, you can enter the script

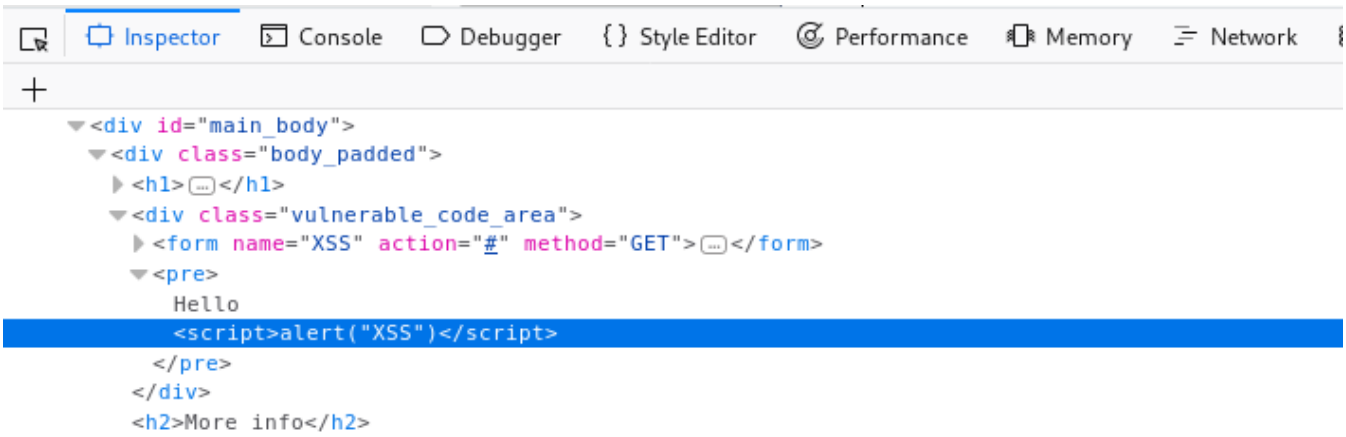
```
<script>alert('XSS')</Script>
```

As you click on the submit button, this must display you an alert box with the message, "XSS".



This means the site is vulnerable to XSS. And whoever uses the URL, the code will be executed on their machine.

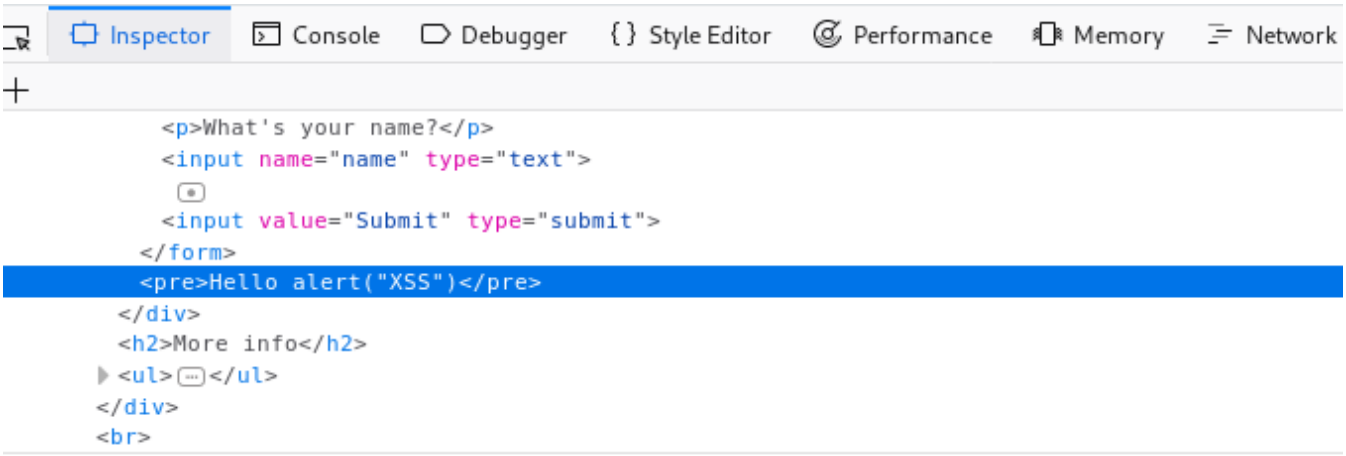
Now, let me show what happens. Inspect the text and here you can see the script has been injected the same way we had given before, which means the site is vulnerable and could be exploited easily.



Medium Security Level

Step 1: Go to Security and change the level to Medium

Step 2: Follow the same process we did before in low security. You will not see the pop up this time.



Here, you see the script tag has been filtered by some medium and the rest of the text has only been displayed. That is why the code was not executed. However, there are many ways you can bypass this.

Step 3: Among the different methods, we will try one simple method. We can just alter the letters of the script tag and will make the code:

```
<scriPt>alert('XSS')</sCrIpt>
```

Now, you will again see the pop-up. This is one of the ways we can bypass the filter. You can also use various other tags to bypass the filter. You can refer the ***OWASP cheatsheet*** *to explore deeper.*

More Advanced XSS

In order to show you some advanced XSS, I will be using ***Mutillidae.***

> ***Tip:*** *Don't always copy and paste the payloads. Try to explore and alter it!*

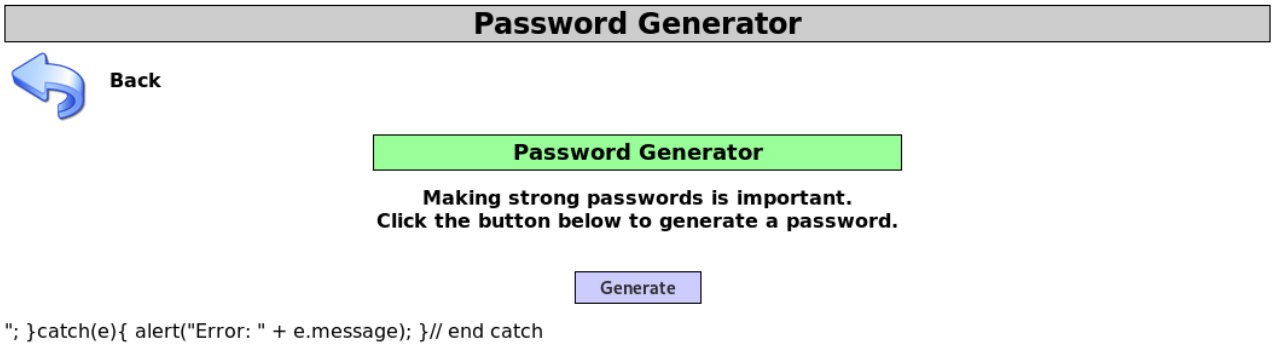
Step 1: Open ***Mutillidae*** and go to OWASP TOP 10 — Injection — Javascript Injection — Password Generator

This page generates a random password for users.

Let's try the same script we had tried before by injecting it into the URL. Now your URL will look something like:

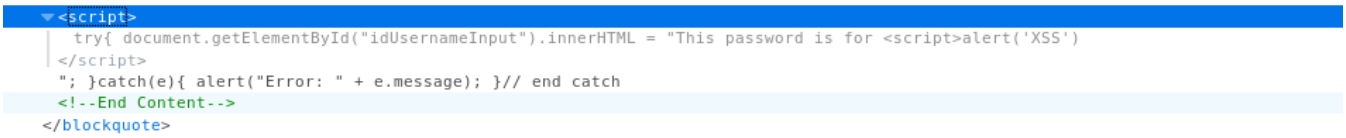
```
username=<script>alert('XSS')</script>
```

Now, as you click enter you will not see the alert pop up, instead you will see an error message.



This means something is going wrong. Let's dive into it by inspecting the page.

Here, you will see this part of the code was broken.



It is trying to automatically keep the script tag inside the user input. So that means, we don't need to add out script tags while injecting. Also, now we need to end the quotation mark before alert and end the line. Then after your injected code, you will need to comment rest of the hardcoded code. Now, your injection code will look like:

```
";alert('XSS'); //
```

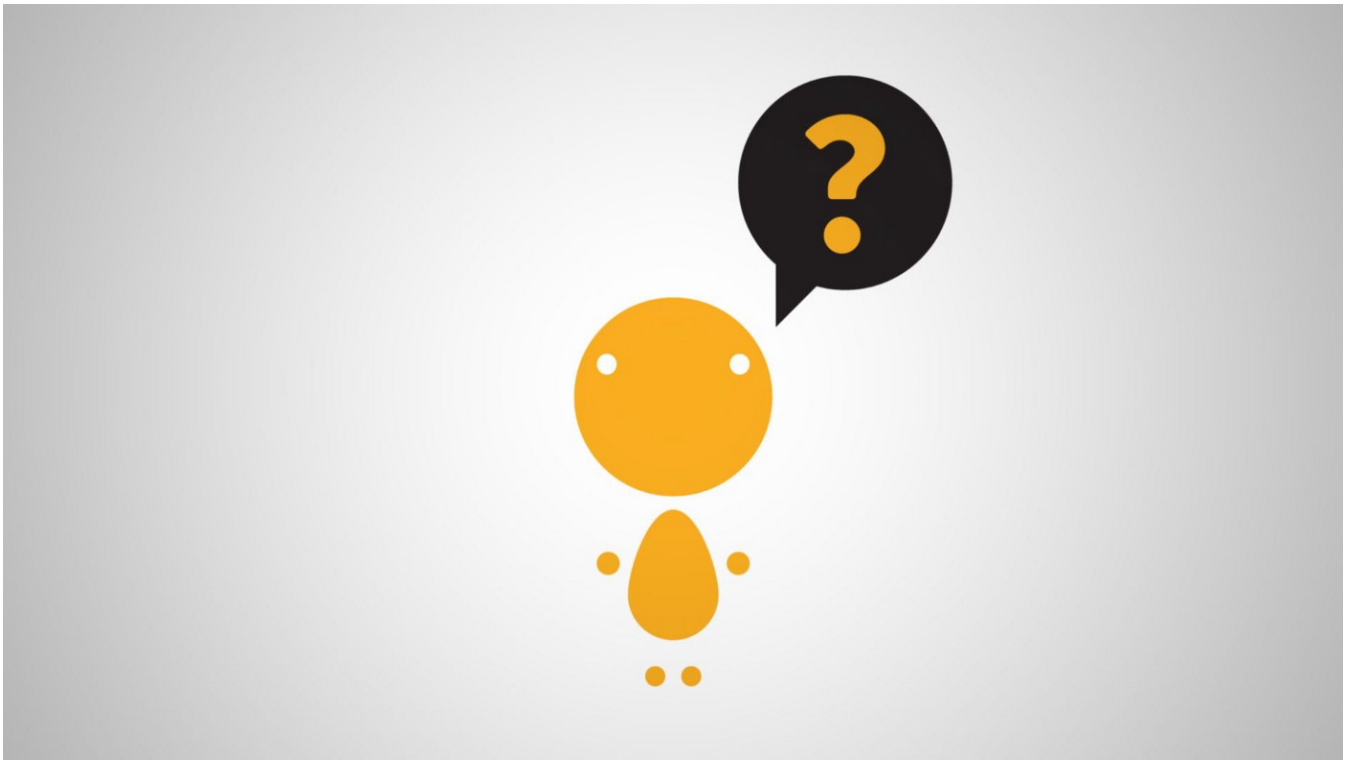
Now, paste it in your URL:

```
page=password-generator.php&username="; alert('XSS'); //
```

Now you will be able to see the alert pop-up!

We managed to run the javascript code in the target website.

How To Protect?



Now, as we talked about Reflected XSS for a while and how it could be executed. Let us now have a glance at how you can protect your site from Reflected XSS.

- As a developer, make sure that any dynamic content coming from the HTTP request cannot be used to inject JavaScript on a page.
- As a user, never click on suspicious links that may contain malicious code.
- Web application firewalls (WAFs) also play an important role in mitigating reflected XSS attacks.
- As a developer, make sure to take the data an application has received and ensuring it's secure before rendering it for the end-user.
- Validating Input.

. . .

Conclusion: *Before diving into Cross-site scripting and exploring its vulnerabilities you need to set up a base first and understand how different types of Cross-site scripting operate. This article was a small demo on how Reflected XSS operates and how it could be prevented. Next, we will further dive into discovering Stored XSS.*

Cybersecurity Cross Site Scripting Pentesting Information Security