**XSS STORED**

"Cross-Site Scripting (XSS)" attacks are a type of injection problem, in which malicious scripts are injected into the otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application using input from a user in the output, without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the JavaScript. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by your browser and used with that site. These scripts can even rewrite the content of the HTML page.

The XSS is stored in the database. The XSS is permanent, until the database is reset or the payload is manually deleted.

**Description:**Stored cross-site scripting (also known as second-order or persistent XSS) arises when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.

**Objective:**

Redirect everyone to a web page of your choosing.

**Impact:**XSS can have huge implications for a web application and its users. User accounts can be hijacked, credentials could be stolen, sensitive data could be exfiltrated, and lastly, access to your client computers can be obtained.

**Prevention:**

* + 1. Server-Side Defense

The best way to prevent Persistent XSS is to make sure that you properly sanitize all user input before you store it on the web server. As a second line of defense, sanitize all static content presented to users. Malicious scripts can be encoded and injected in various ways, so source code sanitization parsers should take it into consideration.

* + 1. Client-Side Defense

As a user, you cannot take any actions to prevent a Persistent XSS attack. You could only disable JavaScript in your browser but then most websites would not function properly. The best that you can do is use a secure and up to date browser with XSS filters turned on.

**LOW**

**Steps to reproduce:**

1. Configure your browser

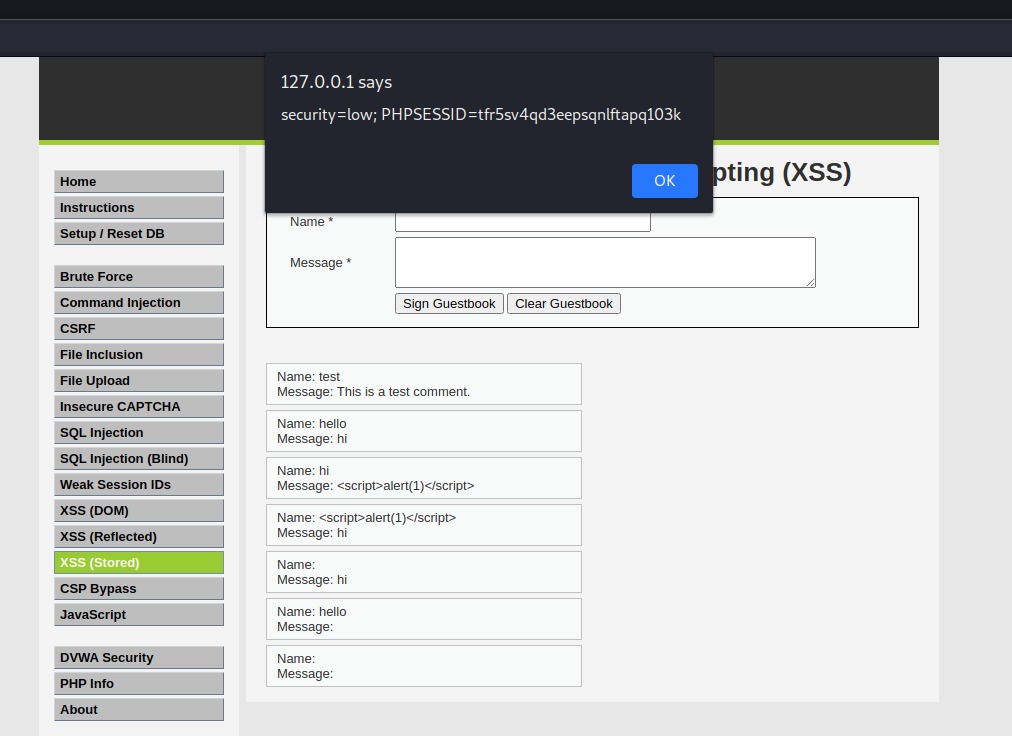
2. Go to the dvwa page and set level of XSS (Stored) to the low level.

3. In this both name and message have stored XSS so in name we have to increase max length of input box.

4. In name and message box write “<script>alert(1)</script>” and click on the sign guestbook.

5. Get a pop up of alert 1.

6. For user cookies write “<script>alert(document.cookie)</script>” in both input.



**MEDIUM**

**Steps to reproduce:**

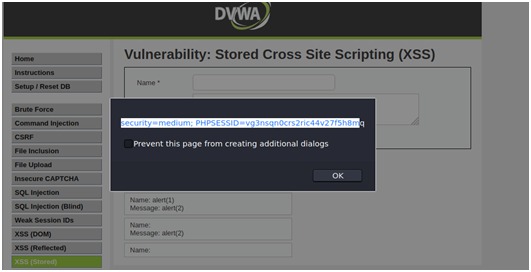
1. Configure your browser

2. Go to the dvwa page and set level of XSS (Stored) to the medium level.

3. “<Script>alert(1)</script>” it worked in name input but had to increase the maxlength in inspect element of $name.

4. Now click on the sign guestbook, get a pop up of alert 1.

5. For user cookies “<Script>alert(document.cookie)</script>” in name field.



**HIGH**

**Steps to reproduce:**

1. Configure your browser
2. Go to the dvwa page and set level of XSS (Stored) to the high level.
3. “<imgsrc=”” onerror=”alert(1)”>”it worked in name input but had to increase the maxlength in inspect element of $name.
4. Now click on the sign guestbook, get a pop up of alert 1.
5. For user cookies “<imgsrc=”” onerror=”alert(document.cookie)”>” in name field.

