

DVWA Installation and Vulnerability Discovery

Step 1 — Installation & Setup

Update & install DVWA

Command-sudo apt update

sudo apt install dvwa

```
[marie@math:~]
$ sudo apt install dvwa
[sudo] password for marie:
The following packages were automatically installed and are no longer required:
  amass-common      libgeos3.14.0    libobjc-14-dev   libwireshark18
  docker-buildx     libgirepository-1.0-1  libplacebo349  libwiretap15
  gir1.2-girepository-2.0 libpngmepp6t64  libportmidi0   libwsutil16
  libarmadillo14    libinstpatch-1.0-2  libravie0.7    libx264-164
  libbluray2        libjs-jquery-ui   libsqlcipher1  python3-bluepy
  libbson-1.0-0t64  libjs-underscore  libtheoradec1 python3-click-plugins
  libdisplay-info2   libmongoc-1.0-0t64  libtheoraenc1 python3-gpg
  libgdal37         libnet1          libudfread0    python3-kismetcapturebtgeiger
Use 'sudo apt autoremove' to remove them.

Installing:
  dvwa

Installing dependencies:
  php8.4-fpm  php8.4-gd

Suggested packages:
  php-pear

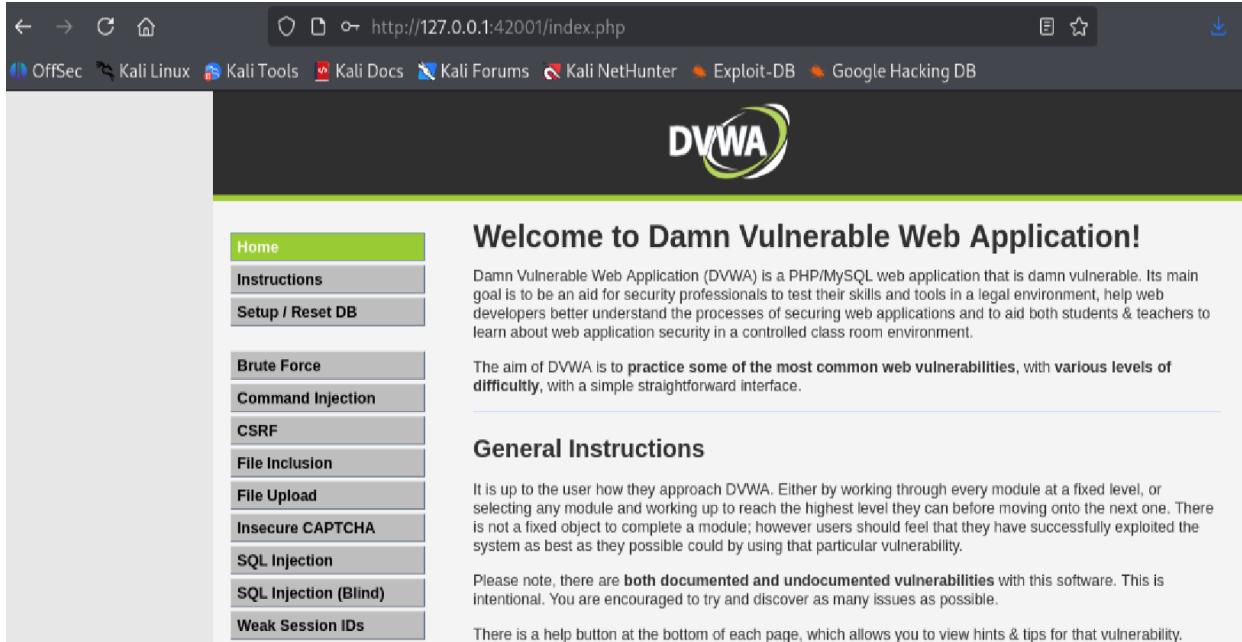
Summary:
  Upgrading: 0, Installing: 3, Removing: 0, Not Upgrading: 77
  Download size: 2,689 kB
  Space needed: 7,975 kB / 2,417 MB available

Continue? [Y/n] y
Get:1 https://http.kali.org/kali kali-rolling/main amd64 php8.4-fpm amd64 8.4.11-1+b1 [1,860 kB]
```

```
Unpacking php8.4-fpm (8.4.11-1+b1) ...
Selecting previously unselected package php8.4-gd.
Preparing to unpack .../php8.4-gd_8.4.11-1+b1_amd64.deb ...
Unpacking php8.4-gd (8.4.11-1+b1) ...
Selecting previously unselected package dvwa.
Preparing to unpack .../dvwa_2.4-0kali1_all.deb ...
Unpacking dvwa (2.4-0kali1) ...
Setting up php8.4-gd (8.4.11-1+b1) ...
Creating config file /etc/php/8.4/mods-available/gd.ini with new version
Setting up php8.4-fpm (8.4.11-1+b1) ...
Creating config file /etc/php/8.4/fpm/php.ini with new version
NOTICE: Not enabling PHP 8.4 FPM by default.
NOTICE: To enable PHP 8.4 FPM in Apache2 do:
NOTICE: a2enmod proxy_fcgi setenvif
NOTICE: a2enconf php8.4-fpm
NOTICE: You are seeing this message because you have apache2 package installed.
update-rc.d: We have no instructions for the php8.4-fpm init script.
update-rc.d: It looks like a network service, we disable it.
Setting up dvwa (2.4-0kali1) ...
dvwa.service is a disabled or a static unit, not starting it.
Processing triggers for man-db (2.13.1-1) ...
Processing triggers for libapache2-mod-php8.4 (8.4.11-1+b1) ...
Processing triggers for kali-menu (2025.4.2) ...
Processing triggers for php8.4-cli (8.4.11-1+b1) ...
Processing triggers for php8.4-fpm (8.4.11-1+b1) ...
NOTICE: Not enabling PHP 8.4 FPM by default.
NOTICE: To enable PHP 8.4 FPM in Apache2 do:
NOTICE: a2enmod proxy_fcgi setenvif
NOTICE: a2enconf php8.4-fpm
NOTICE: You are seeing this message because you have apache2 package installed.
```

2.Run the Dvwa with the command

```
sudo dvwa-start
```



The screenshot shows a web browser window with the URL `http://127.0.0.1:42001/index.php` in the address bar. The page title is "Welcome to Damn Vulnerable Web Application!". On the left, there is a sidebar menu with the following items:

- Home (highlighted in green)
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection
- SQL Injection (Blind)
- Weak Session IDs

The main content area contains the following text:

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is 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 to aid both students & teachers to learn about web application security in a controlled class room environment.

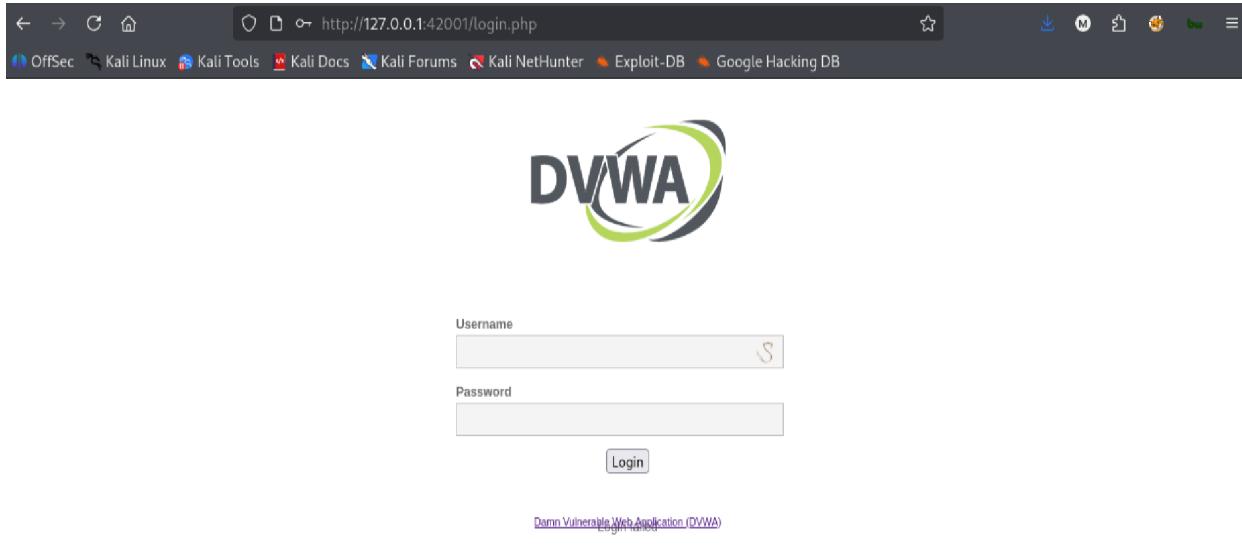
The aim of DVWA is to practice some of the most common web vulnerabilities, with various levels of difficulty, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are both documented and undocumented vulnerabilities with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability.



The screenshot shows a web browser window with the URL `http://127.0.0.1:42001/login.php` in the address bar. The page title is "DVWA". The main content area contains a login form with the following fields:

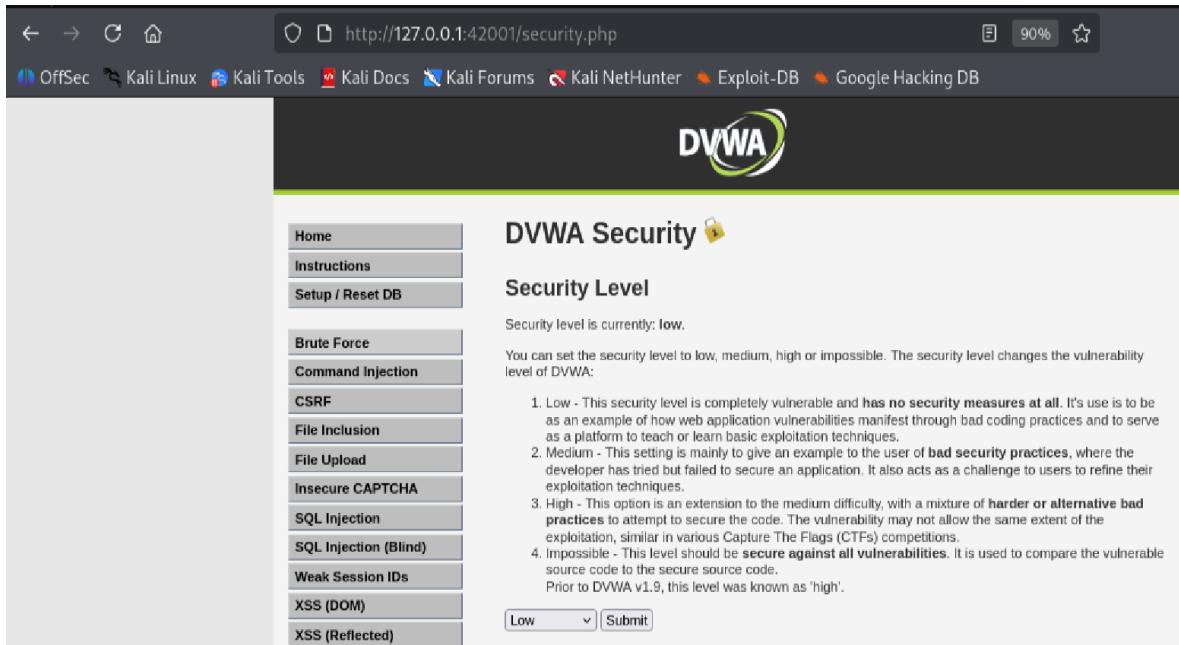
Username:

Password:

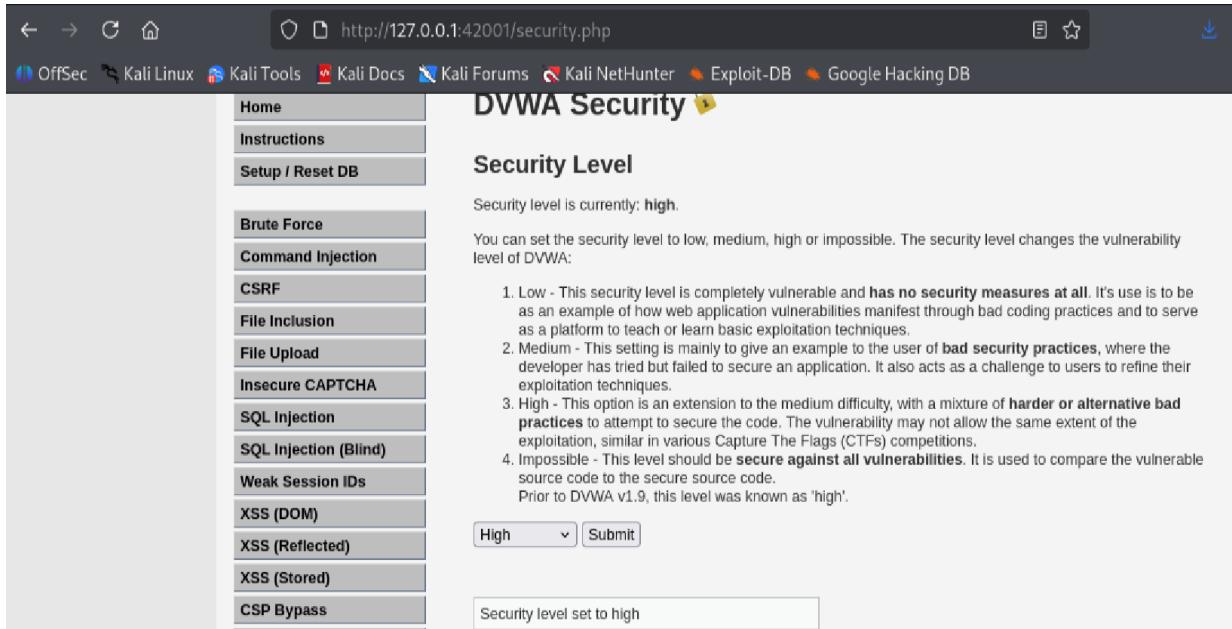
At the bottom of the page, there is a small link: "Damn Vulnerable Web Application (DVWA) Login failed".

Step 2: DVWA Security Levels

In DVWA set Security to Low first , Run tests , then set to medium and high & Rerun the tests



The screenshot shows the DVWA security level set to 'Low'. The interface includes a sidebar with various attack options like Brute Force, Command Injection, and SQL Injection. The main content area displays the DVWA logo and a 'DVWA Security' section with a lock icon. It states: 'Security level is currently: low.' Below this, it explains the security levels: Low (completely vulnerable), Medium (example of bad security practices), High (extension of medium difficulty), and Impossible (secure against all vulnerabilities). A dropdown menu shows 'Low' is selected, and a 'Submit' button is present.



The screenshot shows the DVWA security level set to 'High'. The interface is identical to the previous one, with the same sidebar and main content area. The main content area now shows: 'Security level is currently: high.' Below this, it reiterates the explanation of the four security levels. A dropdown menu shows 'High' is selected, and a 'Submit' button is present. A message at the bottom of the main content area says: 'Security level set to high'.

Step 3: Vulnerability Exercises

Sql Injection

The screenshot shows a web application titled "Vulnerability: SQL Injection". On the left is a sidebar menu with various exploit categories. The "SQL Injection" item is highlighted in green. The main content area displays several examples of successful SQL注入 (SQL injection) queries:

- ID: 1' or '1'='1
First name: admin
Surname: admin
- ID: 1' or '1'='1
First name: Gordon
Surname: Brown
- ID: 1' or '1'='1
First name: Hack
Surname: Me
- ID: 1' or '1'='1
First name: Pablo
Surname: Picasso
- ID: 1' or '1'='1
First name: Bob
Surname: Smith

Below the examples is a "More Information" section with two links:

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netparker.com/blog/web-security/sql-injection-cheat-sheet/>

The URL in the browser bar is <http://127.0.0.1:42001/vulnerabilities/sqli/#>.

This screenshot shows the same "Vulnerability: SQL Injection" interface. In this version, only one successful injection attempt is displayed:

- ID: 1
First name: admin
Surname: admin

The sidebar menu is identical to the first screenshot. The URL in the browser bar is <http://127.0.0.1:42001/vulnerabilities/sqli/#>.

The screenshot shows a browser window with the URL <http://127.0.0.1:42001/vulnerabilities/sql/>. The title bar says "SQL Injection Session Input :: Damn Vulnerable Web Application". The main content area displays a list of user profiles with their ID, first name, and last name. A sidebar on the left lists various attack types, with "SQL Injection" highlighted. A modal dialog box in the center shows the exploit payload being entered into a text input field.

Vulnerability: SQL Injection

Click [here](#) to change your ID.

ID	First name	Last name
1 OR 1=1...	admin	admin
1 OR 1=1...	Gordon	Brown
1 OR 1=1...	Hack	No
1 OR 1=1...	Pablo	Picasso
1 OR 1=1...	Bob	Smith

More Information

- https://www.w3schools.com/sql/sql_injection.asp
- https://www.owasp.org/www-community/attacks/SQL_Injection
- https://www.owasp.org/www-community/vulnerabilities/SQL_Injection
- <https://www.owasp.org/www-community/attacks/XSS>

www.kali.org/docs/

Cross Site Scripting-

When the following payload is entered

<script>alert("You are hacked!")</script>

The screenshot shows a browser window with the URL [http://127.0.0.1:42001/vulnerabilities/xss_r/?name=<script>alert\("You+are+hacked!"\)</script>](http://127.0.0.1:42001/vulnerabilities/xss_r/?name=<script>alert('You+are+hacked!')</script>). The title bar says "XSS (Reflected) :: Damn Vulnerable Web Application". The main content area displays a "Hello" message followed by the injected script. A sidebar on the left lists various attack types, with "XSS (Reflected)" highlighted.

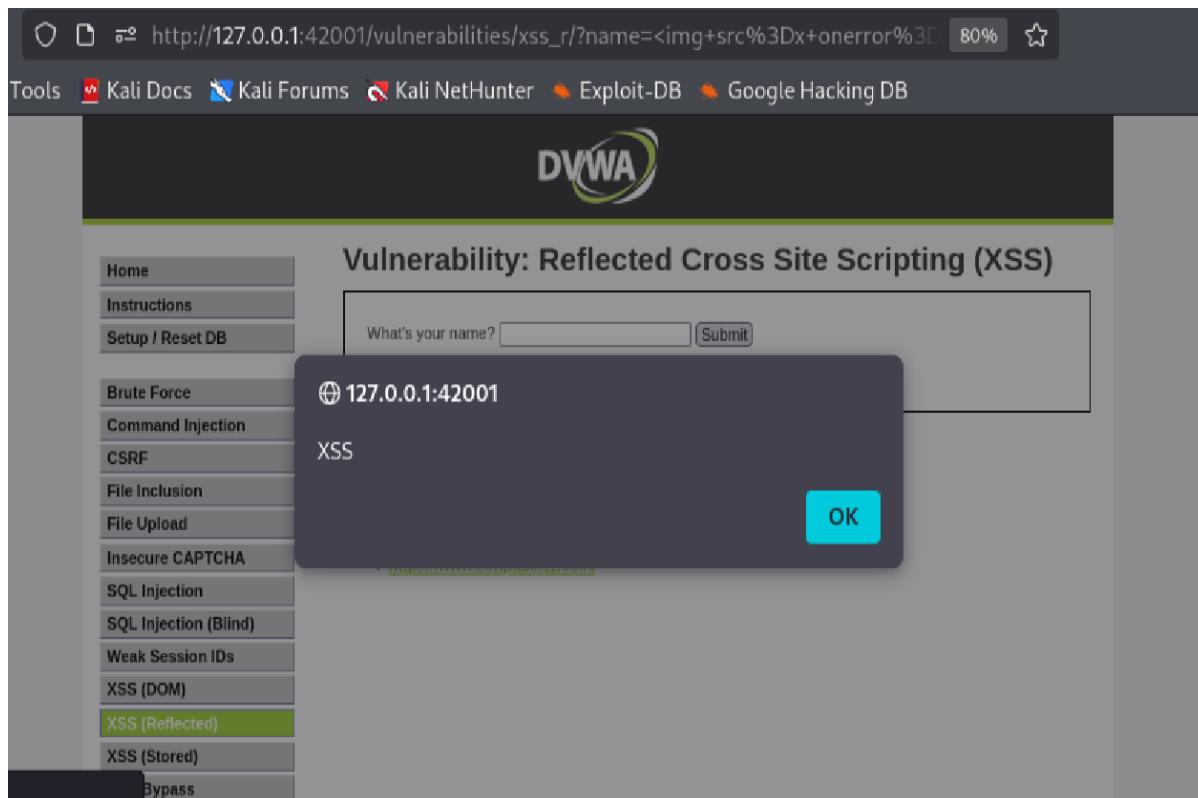
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

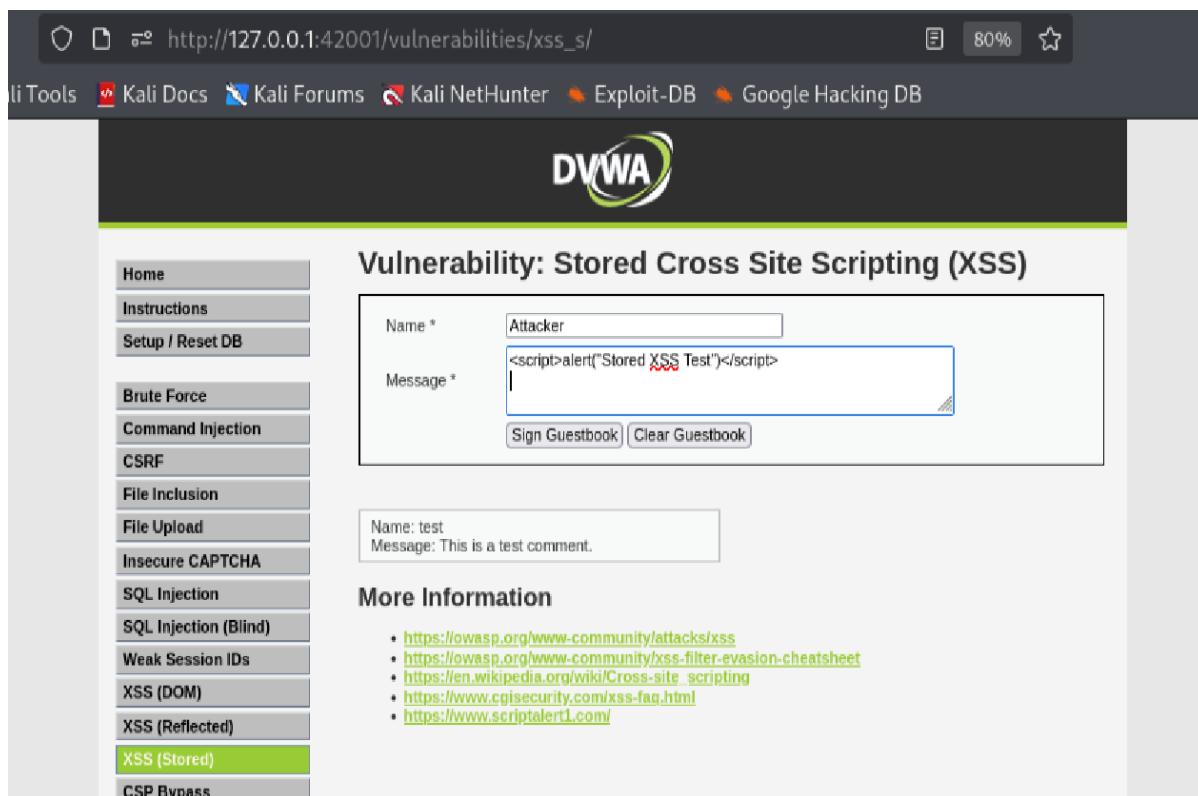
Hello >

More Information

- <https://owasp.org/www-community/attacks/xss/>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <https://www.cgisecurity.com/xss-faq.html>
- <https://www.scriptalert1.com/>



Screenshot of DVWA showing a reflected XSS attack. The URL is http://127.0.0.1:42001/vulnerabilities/xss_r/?name=<img+src%3Dx+onerror%3D. A modal dialog box displays the injected script: <script>alert("127.0.0.1:42001")</script>. The DVWA navigation menu on the left includes XSS (Reflected) under the XSS category.



Screenshot of DVWA showing a stored XSS attack. The URL is http://127.0.0.1:42001/vulnerabilities/xss_s/. The message input field contains <script>alert("Stored XSS Test")</script>. The DVWA navigation menu on the left includes XSS (Stored) under the XSS category.

Script is saved in the database

The screenshot shows the DVWA application interface. The URL in the browser is `http://127.0.0.1:42001/vulnerabilities/xss_s/`. The page title is "Vulnerability: Stored Cross Site Scripting (XSS)". On the left, a sidebar menu lists various security modules: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored) (which is highlighted in green), and CSP Bypass. The main content area displays two examples of stored XSS attacks. The first example shows a guestbook entry with "Name: test" and "Message: This is a test comment.". The second example shows an attacker's message with "Name: Attacker" and "Message: alert('Stored XSS Test!')". Below this, a "More Information" section provides links to external resources about XSS attacks.

- <https://owasp.org/www-community/attacks/xss>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <https://www.cisecurity.com/xss-faq.html>
- <https://www.scriptalert1.com/>

Command Injection

The screenshot shows the DVWA application interface. The URL in the browser is `http://127.0.0.1:42001/vulnerabilities/exec/#`. The page title is "Vulnerability: Command Injection". On the left, a sidebar menu lists various security modules, with "Command Injection" highlighted in green. The main content area shows a "Ping a device" form where the IP address field contains "127.0.0.1; whoami". When the "Submit" button is clicked, the page outputs the results of the ping command, including the host information and the user "dwva". Below this, a "More Information" section provides links to external resources about Command Injection attacks.

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/intl/>
- https://owasp.org/www-community/attacks/Command_Injection

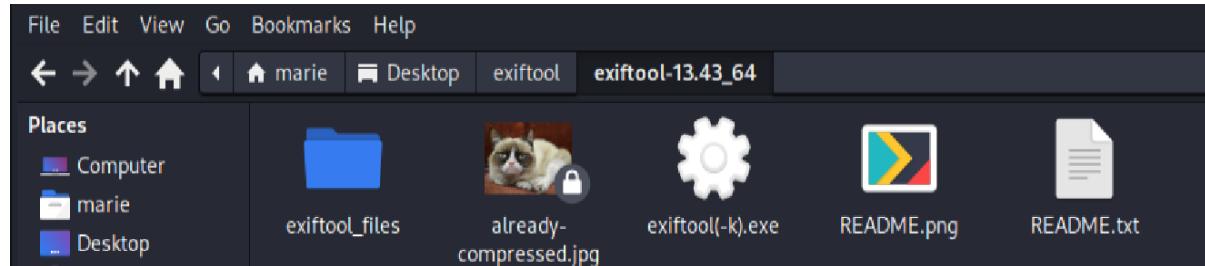
127.0.0.1 && cat /etc/passwd

The screenshot shows the DVWA Command Injection page. On the left, a sidebar lists various vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, **Command Injection**, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), and CSP Bypass. The main content area is titled "Ping a device" and contains a form with "Enter an IP address:" and a "Submit" button. Below the form, red text displays the output of a ping command to 127.0.0.1, followed by a root shell dump of the /etc/passwd file.

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.084 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.028 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.035 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.062 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3078ms  
rtt min/avg/max/mdev = 0.028/0.052/0.084/0.022 ms  
root:x:0:0:root:/root:/usr/bin/zsh  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin  
sync:x:4:65534:sync:/bin:/sync  
games:x:5:60:games:/usr/games:/usr/sbin/nologin  
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin  
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin  
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin  
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin  
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin  
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin  
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
```

File Upload / Unrestricted File Upload

Trying to upload a “Readme.txt” file with the extension changed to “.png” as shown below:



Setting the security level to “Low”

The screenshot shows a web browser window with three tabs open:

- HOW TO DO Command injection
- DVWA Security :: Damn V1.9
- dvwa file upload check - Google Hacking DB

The main content area displays the DVWA logo and the title "DVWA Security". On the left, there is a sidebar menu with the following items:

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection
- SQL Injection (Blind)
- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypass

The "File Upload" item is highlighted. The main content area has a heading "Security Level" and a sub-section "Security level is currently: low." It explains that the security level can be set to low, medium, high, or impossible. A list of four options is provided:

1. Low - This security level is completely vulnerable and **has no security measures at all**. Its use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

A dropdown menu is set to "Low" and a "Submit" button is present.

The file is accepted and uploaded as shown below though it is not originally a .png file:

The screenshot shows a web browser window with the URL `http://127.0.0.1:42001/vulnerabilities/upload/#`. The page title is "Vulnerability: File Upload". On the left, there is a sidebar menu with various security vulnerability categories. The "File Upload" option is highlighted with a green background. The main content area has a form for uploading files. It includes a "Browse..." button, which shows "No file selected.", and an "Upload" button. Below the form, a message in red text reads ".../.../hackable/uploads/README.png successfully uploaded!". At the bottom of the page, there is a section titled "More Information" with two links:

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- <https://www.acunetix.com/websitedevelopment/upload-forms-threat/>

The same observation is made for “Medium” level too. Setting the security level above “Medium” i.e, to “Impossible” or “High” to check if the vulnerability will be detected or not for the falsely created “Readme.png” file:

http://127.0.0.1:42001/security.php

ools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

DVWA

DVWA Security 🔒

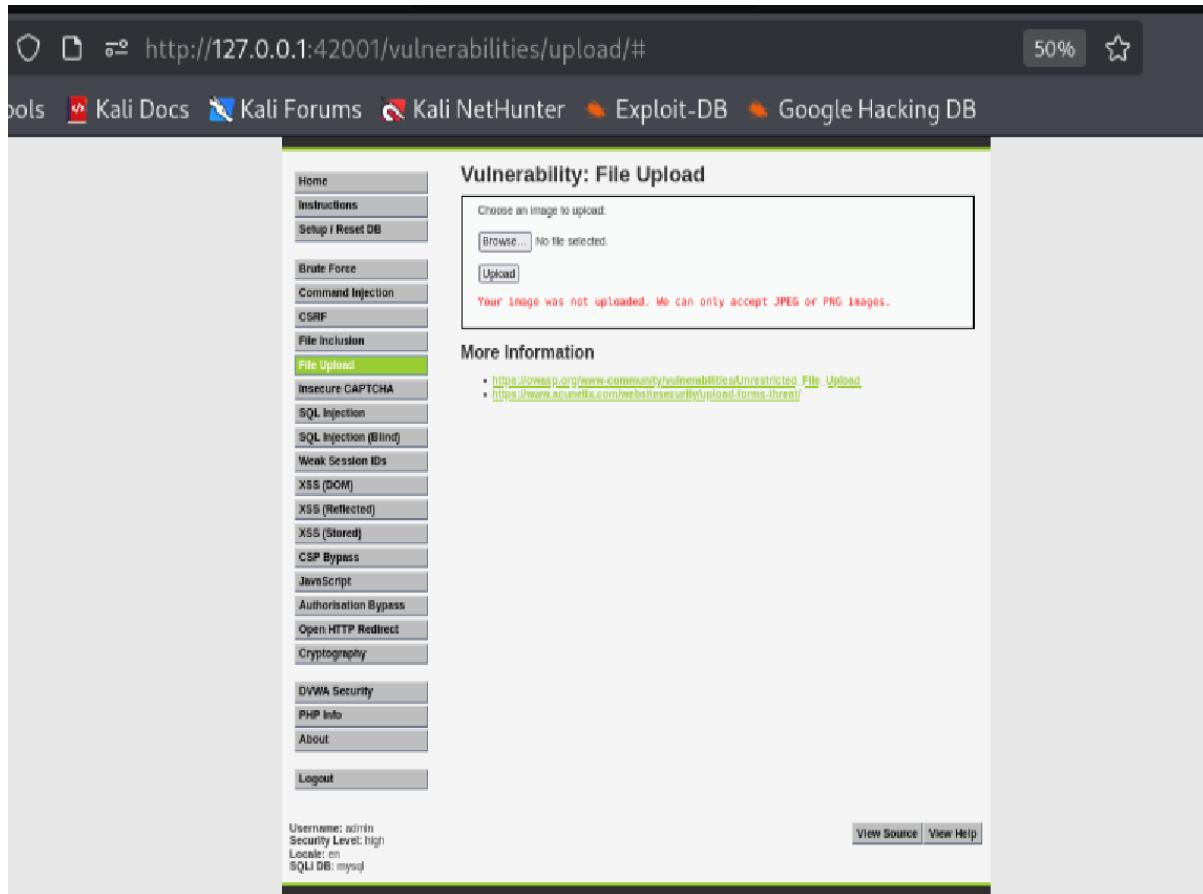
Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. Its use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

Medium



A screenshot of a web browser showing the DVWA (Damn Vulnerable Web Application) File Upload page. The URL is `http://127.0.0.1:42001/vulnerabilities/upload/#`. The page title is "Vulnerability: File Upload". On the left, there's a sidebar with various menu items: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, **File Upload**, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, Authorization Bypass, Open HTTP Redirect, Cryptography, DVWA Security, PHP Info, About, and Logout. The "File Upload" item is highlighted. The main content area has a form for uploading an image. It includes a "Browse..." button, a message "No file selected.", and a "Upload" button. Below the form, a red error message says "Your image was not uploaded. We can only accept JPEG or PNG images." At the bottom of the page, it shows the user information: Username: admin, Security Level: high, Locale: en, and MySQL DB: mysql. There are "View Source" and "View Help" links at the bottom right.

Security Misconfigurations / Insecure headers

```
(marie@math)-[~/home/marie]
$ curl -I http://127.0.0.1:42001/
HTTP/1.1 302 Found
Server: nginx/1.28.0
Date: Sat, 20 Dec 2025 22:36:16 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Set-Cookie: security-impossible; path=/; HttpOnly
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=5b9f6546e1e7993e399fe361b3fb4dc; expires=Sun, 21 Dec 2025 22:36:16 GMT; Max-Age=86400; path=/; domain=127.0.0.1; HttpOnly; SameSite=Strict
Location: login.php
```

Step 4: Light automation

Run Nikto for quick server Misconfigurations

```
(marie@math)-[~/home/marie]
└─$ nikto -h http://127.0.0.1:42001/
- Nikto v2.5.0

+ Target IP:      127.0.0.1
+ Target Hostname: 127.0.0.1
+ Target Port:    42001
+ Start Time:    2025-12-20 17:39:24 (GMT-5)

+ Server: nginx/1.28.0
+ /: The Anti-Clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Root page / redirects to: login.php
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /Login.php: Admin login page/section found.
+ 8074 requests: 0 error(s) and 3 item(s) reported on remote host
+ End Time:      2025-12-20 17:39:34 (GMT-5) (10 seconds)

+ 1 host(s) tested
```

Run whatweb for Fingerprinting

```
(marie@math)-[~/home/marie]
└─$ whatweb http://127.0.0.1:42001/
http://127.0.0.1:42001/ [302 Found] Cookies[PHPSESSID,security], Country[RESERVED][ZZ], HTTPServer[nginx/1.28.0], HttpOnly[PHPSESSID,security], IP[127.0.0.1]
, RedirectLocation[Login.php, nginx[1.28.0]
http://127.0.0.1:42001/Login.php [200 OK] Country[RESERVED][ZZ], DVWA, HTML5, HTTPServer[nginx/1.28.0], IP[127.0.0.1], PHP, PasswordField[password], Title[Login :: Damn Vulnerable Web Application (DVWA)], nginx[1.28.0]
```

Run Dirb or any other Content Discovery tool

```
(marie@math)-[~/home/marie]
└─$ dirb http://127.0.0.1:42001/

DIRB v2.22
By The Dark Raver

_____
START_TIME: Sat Dec 20 17:42:45 2025
URL_BASE: http://127.0.0.1:42001/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

_____
GENERATED WORDS: 4612

_____
→ Scanning URL: http://127.0.0.1:42001/ →
→ DIRECTORY: http://127.0.0.1:42001/config/
→ DIRECTORY: http://127.0.0.1:42001/database/
→ DIRECTORY: http://127.0.0.1:42001/docs/
→ DIRECTORY: http://127.0.0.1:42001/external/
+ http://127.0.0.1:42001/favicon.ico (CODE:200|SIZE:1406)
+ http://127.0.0.1:42001/index.php (CODE:302|SIZE:0)
+ http://127.0.0.1:42001/php.ini (CODE:200|SIZE:154)
+ http://127.0.0.1:42001/phpinfo.php (CODE:302|SIZE:0)
+ http://127.0.0.1:42001/robots.txt (CODE:200|SIZE:25)

_____
→ Entering directory: http://127.0.0.1:42001/config/ →
→ Entering directory: http://127.0.0.1:42001/database/ →
```

```
— Entering directory: http://127.0.0.1:42001/docs/ —
+ http://127.0.0.1:42001/docs/copyright (CODE:200|SIZE:1085)
⇒ DIRECTORY: http://127.0.0.1:42001/docs/graphics/
— Entering directory: http://127.0.0.1:42001/external/ —
— Entering directory: http://127.0.0.1:42001/docs/graphics/ —


---


END_TIME: Sat Dec 20 17:42:50 2025
DOWNLOADED: 27672 - FOUND: 6
```

Run nuclei

```
(marie@math)-[/home/marie]
PS> nuclei -u http://127.0.0.1:42001/
```



```
v3.4.10
projectdiscovery.io

[WRN] Found 1 templates with syntax error (use -validate flag for further examination)
[INF] Current nuclei version: v3.4.10 (outdated)
[INF] Current nuclei-templates version: v10.3.5 (latest)
[INF] New templates added in latest release: 57
[INF] Templates loaded for current scan: 8910
[INF] Executing 8908 signed templates from projectdiscovery/nuclei-templates
[WRN] Loading 2 unsigned templates for scan. Use with caution.
[INF] Targets loaded for current scan: 1
[INF] Templates clustered: 1862 (Reduced 1749 Requests)
[dvwa-default-login] [http] [critical] http://127.0.0.1:42001/index.php [password="password",username="admin"]
[cookies-without-secure] [javascript] [info] 127.0.0.1:42001 ["security","PHPSESSID"]
[waf-detect:nginxgeneric] [http] [info] http://127.0.0.1:42001/
[INF] Scan completed in 1m. 3 matches found.
```