Digital Transformation Technology (Digitech) | e-ISSN: 2807-9000
Volume 5, **Number** 1, Maret 2025
https://doi.org/10.47709/digitech.v5i1.5922

Identifikasi Serangan SQL Injection Berbantuan Aplikasi Pengujian Keamanan Web DVWA(Damn Vulnerable Web Application)

Rahadian Setiya Wiandani[1] , Muhlis Tahir[2] , Ilham Akbar Dyransyha[3] , Rovikotul Ummah[4]
[1,2,3,4]Universitas Trunojoyo Madura, Indonesia
[1]220631100038@student.trunojoyo.ac.id, [2]muhlistahir@trunojoyo.ac.id,
[3]220631100059@student.trunojoyo.ac.id, [4]220631100037@student.trunojoyo.ac.id

open access
Histori Artikel:
Diajukan: 19 Mei 2025
Disetujui: 17 Juli 2025
Dipublikasi: 24 Juli 2025
Kata Kunci:
DVWA; Keamanan Website; *Web Security Analysis*; *Sql Injection*; OWASP

Abstrak
Seiring dengan meningkatnya penggunaan aplikasi web di berbagai sektor, keamanan menjadi aspek penting yang harus diperhatikan, terutama terhadap ancaman serangan SQL Injection. Serangan ini memanfaatkan kelemahan pada validasi input untuk menyisipkan perintah SQL berbahaya ke dalam sistem basis data. Penelitian ini bertujuan untuk mengidentifikasi dan menganalisis celah keamanan terhadap serangan SQL Injection menggunakan metode *penetration testing* berdasarkan standar OWASP. Media pengujian yang digunakan adalah Damn Vulnerable Web Application (DVWA), yaitu aplikasi web yang secara sengaja dirancang rentan untuk keperluan edukasi dan pengujian keamanan. Pengujian dilakukan pada berbagai level keamanan DVWA Low, Medium, High, dan Impossible dengan teknik injeksi SQL secara manual maupun otomatis menggunakan SQLMap. Hasil pengujian menunjukkan bahwa serangan berhasil dilakukan pada level Low dan Medium akibat lemahnya validasi input, sedangkan pada level High dan Impossible, sistem berhasil mencegah serangan berkat penerapan teknik *prepared statements* dan validasi input numerik. Penelitian ini menunjukkan bahwa DVWA efektif sebagai media simulasi untuk memahami mekanisme serangan SQL Injection serta strategi mitigasinya, sekaligus menekankan pentingnya penerapan standar keamanan OWASP dalam pengembangan aplikasi web yang aman.

PENDAHULUAN

Seiring semakin berkembang nya teknologi system informasi dikalangan masyarakat, berkembang pula system yang dapat memudahkan masyarakat untuk mengakses dan mencari suatu informasi dalam bentuk sebuah website. Teknologi informadi menjadi salah satu peran penting dalam suatu aktivitas perusahaan, organisasi untuk mendukung kinerja dan aktivitas . namun dalam pengelolaan IT keamanan suatu website dalah hal yang sangat penting. Resiko keamanan menjadi salah satu hal yang berada pada urutann terakhir dalam hal – hal yang dianggap penting. Dan apabilla menggangu performa seringkali di kurangi . hal itu berbanding terbalik dengan semakin banyak nya celah yang keamanan dari website tersebut.

Open Web Application Security Project (OWASP) merupakan organisasi non profit berfokus pada peningkatan keamanan perangkat lunak. Tujuan utama dari OWASP adalah meningkatkan keamanan perangkat lunak dengan menyediakan sumber daya, pedoman, dan tools yang dapat digunakan oleh para pengembang, profesional keamanan, dan organisasi untuk mengidentifikasi kerentanan keamanan dalam aplikasi web (Ela Nurelasari, 2024). *OWASP* menjadi *framework* yang digunakan oleh pengembang dan ahli teknologi untuk mengamankan website. *OWASP* memberikan platform bagi pengembang untuk meningkatkan keamanan sistem melalui proyek yang open-source bersama dengan tools dari *OWASP* sebagai pendukung dalam pengujian sistem. *SQL Injection* adalah sebuah teknik hacking yang bertujuan untuk menyusup ke dalam sistem sebuah website untuk mengetahui isi database, ataupun informasi-informasi yang terdapat di situs tersebut. Teknik ini dapat terjadi dikarenakan adanya kode-kode program yang lemah dan adanya proteksi yang kurang aman dan lemah dari pengelola website tersebut atau yang disebut administrator. *SQL Injection* merupakan salah satu teknik hacking yang hanya memerlukan port 80 meskipun sering dilakukan patch pada server. Kode SQL tersebut diinjeksikan pada *query SQL* sebagai input melalui halaman website. Hal ini memungkinkan intruder untuk memasuki halaman web administrator tanpa melakukan scan port yang terbuka, tidak terdeksi oleh firewall dan bahkan tidak menggunakan tool apapun.

Damn Vulnerable Web Application (DVWA) adalah sebuah aplikasi web berbasis *PHP* dan *MySQL* yang sangat rentan terhadap sebuah serangan keamanan. Tujuan utama dari dibuatnya aplikasi ini adalah sebagai sarana bagi praktisi keamanan untuk menguji kemampuan yang bersifat legal. DVWA juga ditujukan untuk membantu

pengembang web dalam memahami lebih dalam tentang proses pengamanan web serta dapat digunakan oleh kalangan akademik untuk mempelajari sistem keamanan berbasis aplikasi wcb. Damn Vulnerable Web Application (DVWA) juga merupakan alat yang dibangun menggunakan PHP/MySQL Alat ini merupakan bantuan bagi para profesional keamanan dan pengembang Web. Hal ini dapat membantu pengembang web untuk memahami proses pengamanan aplikasi web. Alat DVWA digunakan untuk menganalisis kerentanan melalui SQL Injection. (Ashish Kumar, 2016).

STUDI LITERATUR

Untuk mendukung penelitian ini, dilakukan penelusuran terhadap beberapa studi terdahulu yang relevan terkait serangan SQL Injection, metode deteksi dan pencegahannya, serta praktik *penetration testing* pada aplikasi web. Studi-studi tersebut memberikan gambaran mengenai perkembangan teknologi keamanan web dan pendekatan yang digunakan dalam mengidentifikasi serta menangani kerentanan terhadap SQL Injection. Tabel berikut menyajikan ringkasan dari lima penelitian terdahulu yang memiliki keterkaitan erat dengan topik dan metode yang digunakan dalam penelitian ini:

Penelitian mengenai keamanan aplikasi web terhadap serangan SQL Injection telah banyak dilakukan dengan pendekatan dan metode yang beragam. Salah satu penelitian terbaru adalah oleh Ren dkk. (2025) dalam studi berjudul *Enhancing SQL Injection Detection and Prevention Using Generative Models*. Penelitian ini bertujuan untuk mengembangkan model deteksi SQL Injection yang lebih adaptif terhadap serangan baru dengan memanfaatkan model generatif seperti Variational Autoencoder (VAE) dan CWGAN-GP. Melalui pendekatan ini, peneliti berhasil menghasilkan data sintetik berupa payload SQL yang memperkaya pelatihan model. Hasilnya, sistem deteksi menunjukkan kemampuan dalam mengenali pola serangan yang belum pernah ada sebelumnya serta mengurangi tingkat false positive dan false negative secara signifikan.

Penelitian lain oleh Li dkk. (2025) yang berjudul *Are Your LLM-based Text-to-SQL Models Secure?* mengkaji kerentanan model Large Language Model (LLM) dalam menerjemahkan teks alami menjadi query SQL. Studi ini menggunakan teknik poisoning model dengan menyisipkan trigger tersembunyi (ToxicSQL) yang dievaluasi terhadap model LLM. Hasilnya menunjukkan bahwa model dapat disusupi untuk menghasilkan query SQL berbahaya saat diberi input tertentu, menandakan perlunya kewaspadaan terhadap penerapan LLM dalam sistem-sistem yang bersifat kritikal.

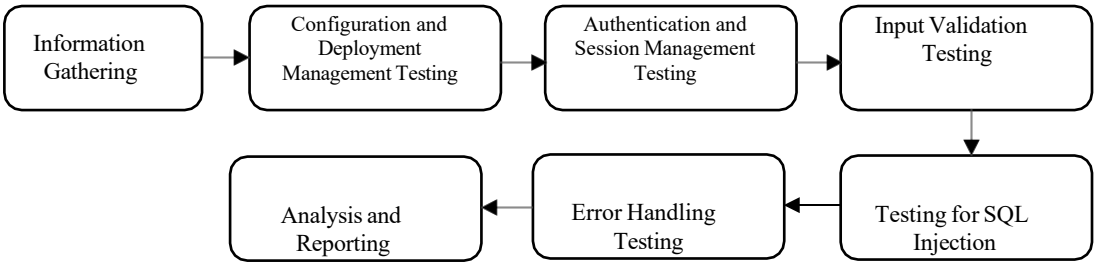
Penelitian Yang dan Wu (2024) dalam penelitiannya berjudul *AdvSQLi: Generating Adversarial SQL Injections against Real-world WAF-as-a-service* berfokus pada pengembangan payload SQL Injection yang mampu melewati sistem Web Application Firewall (WAF), baik komersial maupun open-source. Dengan menggunakan framework generatif, peneliti menciptakan adversarial payloads yang berhasil mengelabui deteksi WAF, dengan tingkat keberhasilan 100% pada WAF publik dan 79% pada WAF komersial. Hal ini menunjukkan bahwa WAF bukanlah solusi absolut dan memerlukan pembaruan berkelanjutan terhadap pola serangan.

Penelitian lainnya oleh Fernandes dan Costa (2025) mengusulkan *A Hybrid Approach for Detecting SQL Injection Using Machine Learning Techniques*. Penelitian ini menggabungkan beberapa metode machine learning, seperti Naive Bayes, LSTM, dan Random Forest, untuk meningkatkan akurasi dalam mendeteksi serangan SQL Injection. Pengujian dilakukan pada data nyata serta aplikasi simulasi seperti DVWA. Hasilnya menunjukkan bahwa model hybrid memberikan hasil deteksi yang lebih akurat dibandingkan dengan metode tunggal, menjadikannya pendekatan yang efektif dalam mendeteksi berbagai bentuk serangan SQLi.

Terakhir, studi oleh Rahman dan Putra (2024) yang berjudul *Uji Penetrasi Injeksi SQL terhadap Celah Keamanan Database Website menggunakan SQLMap* dilakukan dengan melakukan pengujian terhadap beberapa situs institusi, termasuk sekolah dan lembaga pemerintah di Indonesia. Penelitian ini menggunakan metode penetration testing secara manual dan otomatis dengan SQLMap. Hasil pengujian menunjukkan bahwa banyak situs masih rentan karena belum mengimplementasikan validasi input secara memadai. Penelitian ini memberikan kontribusi penting sebagai bukti empiris mengenai lemahnya pengamanan database di tingkat lokal dan urgensi penerapan standar keamanan web.

METODE

Dalam penelitian ini, digunakan metode pengujian keamanan aplikasi web berdasarkan OWASP (Open Web Application Security Project) Testing Guide v4. *OWASP menyediakan pendekatan terstruktur dalam pengujian aplikasi web untuk mengidentifikasi dan mengevaluasi potensi kerentanan seperti SQL Injection.* (OWASP, 2017). Metode ini dipilih karena memberikan pendekatan sistematis dan terstruktur dalam mengidentifikasi serta menganalisis kerentanan keamanan pada aplikasi web, khususnya pada jenis serangan *SQL Injection*. OWASP merupakan standar terbuka yang diakui secara internasional dalam pengembangan dan pengujian keamanan perangkat lunak. Tahapan dalam metode OWASP Testing Guide mencakup proses dari pengumpulan informasi hingga pelaporan hasil pengujian secara menyeluruh. Tahapan sebagai berikut :



Gambar 1. Metode OWASP (Open Web Application Security Project) Testing Guide v4

.

1. Pengumpulan Informasi (Information Gathering)
   Pada tahap awal ini, peneliti mengumpulkan informasi sebanyak mungkin mengenai target aplikasi. Informasi ini meliputi struktur direktori, parameter input, teknologi yang digunakan (seperti PHP dan MySQL), serta halaman yang berpotensi menjadi target serangan. Tahap ini penting untuk mengetahui bagaimana aplikasi bekerja dan bagian mana yang dapat dimanfaatkan oleh penyerang. Tahapan ini dilakukan untuk mengidentifikasi struktur dan teknologi dari sistem yang diuji, sebagai dasar untuk melakukan pengujian lanjutan terhadap kerentanannya (Nugroho & Laksana, 2021).
2. Pengujian Konfigurasi dan Penerapan Sistem (Configuration and Deployment Management Testing) Pada tahap ini, peneliti mengevaluasi konfigurasi sistem, termasuk pengaturan server, database, dan aplikasi web. Konfigurasi sistem yang tidak aman dapat membuka peluang serangan, sehingga perlu diuji apakah pengaturan default telah diubah atau tidak (Pratama & Sari, 2020). Kesalahan konfigurasi seperti direktori yang terbuka, file konfigurasi yang dapat diakses, atau pengaturan default yang tidak diubah dapat menjadi celah yang dimanfaatkan oleh penyerang.
3. Pengujian Autentikasi dan Manajemen Sesi (Authentication and Session Management Testing)
   Tahap ini dilakukan pengujian terhadap proses login dan pengelolaan sesi pengguna. Peneliti mencoba serangan login bypass dengan SQL Injection serta memeriksa apakah session ID dapat ditebak, atau apakah sesi tetap aktif setelah logout. Hal ini penting karena kontrol autentikasi merupakan gerbang utama akses ke sistem. Uji autentikasi bertujuan untuk mengevaluasi apakah sistem dapat dibobol melalui celah login dan bagaimana sesi pengguna dikelola setelah proses masuk (Syahputra & Rahman, 2022).
4. Pengujian Validasi Input (Input Validation Testing)
   Aplikasi web harus mampu memvalidasi setiap input dari pengguna. Pada tahap ini, peneliti memberikan input tidak wajar atau berbahaya (seperti karakter SQL) untuk melihat apakah sistem melakukan penyaringan. Salah satu penyebab utama SQL Injection adalah tidak adanya validasi input yang baik terhadap data yang masuk ke sistem (Fadillah & Wahyuni, 2020). Jika input tidak tervalidasi dengan baik, maka sistem rentan terhadap berbagai serangan, termasuk SQL Injection.
5. Pengujian SQL Injection (Testing for SQL Injection)
   Ini merupakan fokus utama dari penelitian. Peneliti menguji apakah parameter input pada DVWA rentan terhadap injeksi SQL. Uji coba dilakukan baik secara manual menggunakan payload seperti ' OR 1=1 -- maupun otomatis menggunakan tools seperti SQLMap. SQL Injection terjadi karena sistem tidak mampu memfilter perintah SQL yang disisipkan melalui input pengguna, yang kemudian langsung dieksekusi oleh database (Rohman & Arifin, 2021). Pengujian dilakukan pada berbagai level keamanan DVWA (low, Medium, high, dan implossible).
6. Pengujian Penanganan Error (Error Handling Testing)
   Tahapan ini bertujuan untuk melihat bagaimana aplikasi merespon jika terjadi kesalahan, terutama saat payload SQL dikirimkan. Aplikasi yang baik harus menyembunyikan pesan kesalahan teknis dari pengguna. Namun, jika aplikasi menampilkan pesan kesalahan SQL yang jelas, maka informasi tersebut dapat dimanfaatkan oleh penyerang. Pesan kesalahan yang terlalu detail dapat menjadi petunjuk bagi penyerang untuk memahami struktur query dan tabel yang digunakan (Prasetyo & Nugraha, 2021).
7. Analisis dan Pelaporan (Analysis and Reporting)
   Pada tahap akhir, peneliti melakukan analisis terhadap semua hasil pengujian, mencatat parameter rentan, payload yang berhasil, serta level keamanan yang berhasil ditembus. Dari temuan tersebut, peneliti memberikan rekomendasi seperti penggunaan prepared statement, sanitasi input, dan validasi sisi server.

HASIL

Berdasarkan table 1 dibawah, Pengujian dilakukan dengan menyisipkan payload SQL sederhana untuk mengevaluasi efektivitas mekanisme pertahanan di setiap level keamanan, mulai dari yang paling lemah (Low) hingga yang paling ketat (Impossible). Tabel berikut merangkum hasil temuan tersebut:

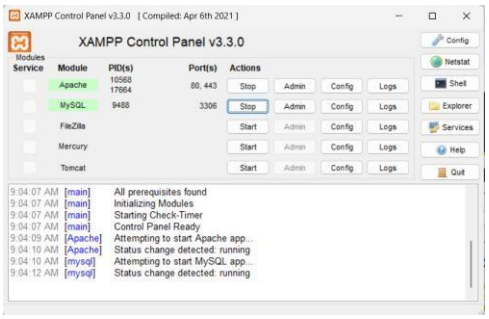
Tabel 1. Hasil Pengujian *SQL Injection* dari berbagai tingkat keamanan

Tingkat Keamanan	Status Serangan SQL Injection	Teknik Pengamanan	Analisis dan Keterangan Detail
Low	**Berhasil**	Tidak ada validasi atau sanitasi input	Sistem sangat rentan. Input dari pengguna langsung dimasukkan ke dalam perintah SQL tanpa proses pemfilteran. Payload sederhana seperti `' OR '1'='1` dapat menampilkan seluruh data pengguna. Data sensitif seperti username, email, hingga *password hash* dapat diekstraksi. Ini mencerminkan kondisi rawan pada banyak aplikasi web yang belum menerapkan standar keamanan dasar.
Medium	**Berhasil**	Menggunakan `mysqli_real_escape_string()`	Ada sedikit peningkatan dari sisi sanitasi input, namun sistem masih menyusun query secara dinamis (string concatenation) dan **belum menggunakan prepared statements**. Teknik ini tidak cukup kuat menahan injeksi kompleks atau *encoded payloads*. Hasil pengujian menunjukkan bahwa meskipun payload sederhana dibatasi, SQLMap tetap bisa menembus dengan modifikasi tertentu.
High	**Sebagian berhasil** *(pada modul tertentu)*	Validasi input, pembatasan pesan error, penggunaan `$_SESSION`	Tingkat keamanan mulai menunjukkan ketahanan. Beberapa fungsi telah membatasi input hanya berupa angka dan mengurangi keluaran error. Namun, modul tertentu seperti `vulnerabilities/sqli/session- input.php` masih bisa dieksploitasi karena masih menerima input dinamis dan tidak semua endpoint memakai *prepared statements*. Ini menandakan bahwa keamanan belum diimplementasikan secara menyeluruh di seluruh modul.
Impossible	**Gagal total (tidak berhasil dieksploitasi)**	Menggunakan prepared statements, `is_numeric()`, token CSRF	Sistem menunjukkan ketahanan penuh terhadap berbagai jenis payload, baik manual maupun otomatis. Semua input divalidasi secara ketat dan dibatasi hanya untuk nilai numerik. Query SQL dibentuk melalui prepared statements yang mengikat parameter dengan aman, sehingga tidak bisa disisipi kode SQL. Ditambah dengan proteksi CSRF, membuat level ini sangat sulit ditembus bahkan dengan SQLMap.

PEMBAHASAN

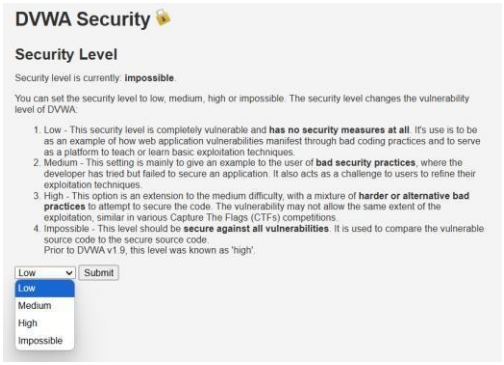
Hasil pengujian SQL Injection pada platform DVWA menunjukkan bahwa tingkat keamanan aplikasi web sangat dipengaruhi oleh mekanisme validasi input dan metode penyusunan query yang digunakan. Pada level *Low* dan *Medium*, input pengguna masih dapat dimanipulasi untuk mengeksekusi perintah SQL berbahaya karena tidak adanya penggunaan *prepared statements* dan sanitasi yang belum menyeluruh. Sebaliknya, pada level *High* dan *Impossible*, terutama dengan penerapan *prepared statements* dan validasi numerik, serangan berhasil digagalkan. Hal ini menegaskan pentingnya implementasi standar keamanan aplikasi web sebagaimana direkomendasikan oleh OWASP, khususnya dalam menangani serangan injeksi.

Sebelum melakukan uji penetrasi, penting untuk membuat lingkungan uji yang menyerupai kondisi produksi, namun tetap aman dan terkendali (Williams, 2021). Lingkungan uji ini disiapkan dengan menginstal **XAMPP** sebagai server lokal.



Gambar 2. XAMPP Control Panel

Penggunaan Platform DVWA sebagai target pengujian kemudian Peneliti memastikan bahwa semua layanan seperti Apache dan MySQL berjalan dengan baik. DVWA dikonfigurasi pada empat level keamanan: *Low, Medium, High*, dan *Impossible*, untuk membandingkan efektivitas mitigasi pada tiap level terhadap serangan SQL Injection.



Gambar 3. Platform Pengujian DVWA

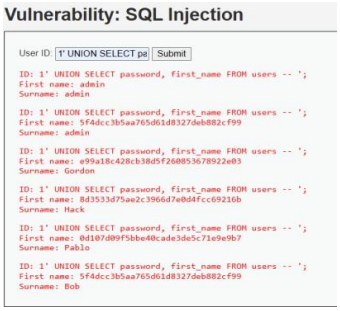
8. Tahap Pengujian (Testing Phase)

Pengujian dilakukan secara manual dan otomatis. Dalam pengujian manual, peneliti menyisipkan payload SQL sederhana seperti ***1' UNION SELECT password, first_name FROM users -- ';*** ke dalam form login untuk melihat apakah sistem dapat dibypass. Teknik seperti Union-based injection, error-based injection, dan commenting (--) juga diuji pada berbagai modul input DVWA. Tujuannya adalah mengobservasi bagaimana aplikasi merespons input yang tidak valid.

```
switch ($_DVWA['SQLI_DB']) {
    case MYSQL:
        // Check database
        $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
        $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' .
```

Gambar 4. *Low SQL Injection Source*

Pada code keamanan Low security tidak ada validasi/sanitasi atau penggunaan prepared statement, membuatnya sangat rentan terhadap serangan SQL Injection.



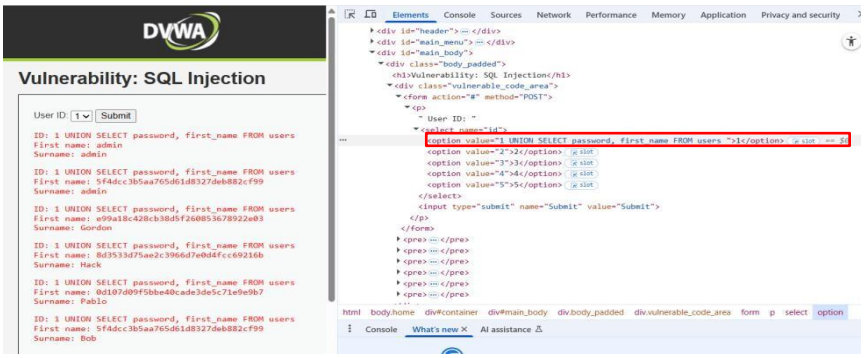
Gambar 5. Uji keamanan *SQL Injection Low security*

Kode selanjutanya merupakan kode keamanan yang sedikit lebih aman dibanding versi sebelumnya karena sudah ada penggunaan mysqli_real_escape_string() untuk men-sanitasi input yang hanya sama dengan kode yang low security hanya saja pada penerapan dan kode yang digunakan tidak memiliki simbol atau karakter didalamnya seperti berikut ***1 UNION SELECT password, first_name FROM users***. Namun, ini masih belum sepenuhnya aman, dan masih tergolong sebagai tingkat keamanan *Medium* dalam konteks DVWA *(Damn Vulnerable Web Application)*.

```
$id = mysqli_real_escape_string($GLOBALS["___mysqli_ston"], $id);

switch ($_DVWA['SQLI_DB']) {
    case MYSQL:
        $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
        $result = mysqli_query($GLOBALS["___mysqli_ston"], $query) or die( '<pre>'
```

Gambar 6. *Medium SQL Injection Source*

Pada medium security ini Masih menyusun query SQL secara dinamis dan Kode Ini masih memungkinkan bypass jika input tidak berupa angka atau jika sanitasi tidak sempurna, serta belum menggunakan prepared statement yang merupakan standar terbaik saat ini.



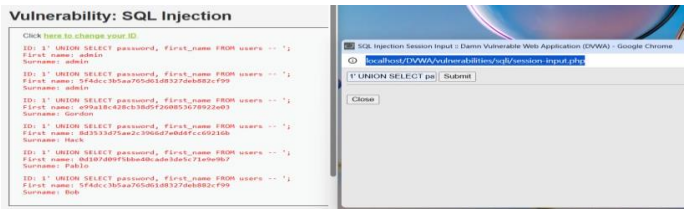
Gambar 7. Uji keamanan *SQL Injection Medium security*

Pada level **High Security** di DVWA, aplikasi sudah menerapkan perlindungan yang kuat terhadap serangan, seperti dengan mengambil input dari sumber yang tidak dapat dimanipulasi langsung oleh pengguna (misalnya $_SESSION), membatasi output error agar tidak menampilkan detail sistem, serta menyusun query SQL yang lebih aman. Meskipun belum selalu menggunakan prepared statements, input telah difilter dengan baik sehingga eksploitasi seperti SQL Injection menjadi sangat sulit dilakukan.

```
if( isset( $_SESSION [ 'id' ] ) ) {
    // Get input
    $id = $_SESSION[ 'id' ];

    switch ($_DVWA['SQLI_DB']) {
        case MYSQL:
            // Check database
            $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
            $result = mysqli_query($GLOBALS["___mysqli_ston"], $query ) or die( '<pre>Something went wrong.</pre>' );
```

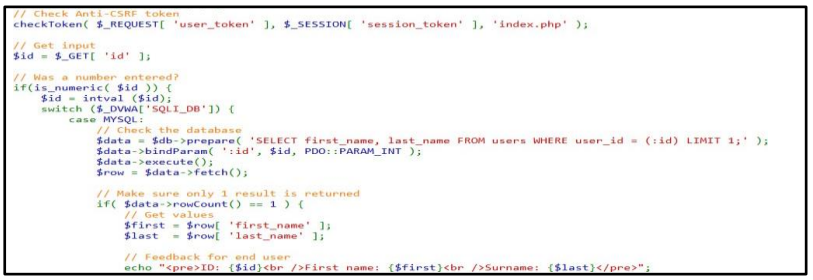
Gambar 8. High *SQL Injection Source*

Kode tersebut membuat pengguna tidak bisa memanipulasi nilai id melalui URL atau form, sehingga eksploitasi SQL Injection jadi sangat sulit. Lalu pada LIMIT 1 bisa meminimalkan dampak dari kesalahan logika dalam query. Serta pada kode *or die('<pre>Something went wrong.</pre>')* berfungsi mencegah informasi struktur database bocor ke attacker (error disclosure).



Gambar 9. Uji keamanan *SQL Injection High security*

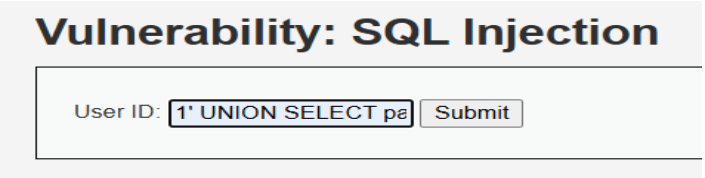
Meskipun kode berada pada level High Security, pada halaman session-input.php masih dapat diretas karena tetap menerima input langsung dari pengguna tanpa validasi atau penggunaan prepared statements, sehingga memungkinkan serangan SQL Injection melalui query seperti *1' UNION SELECT password, first_name FROM users -- ';*, Hal ini menunjukkan bahwa pengamanan pada level High belum diterapkan secara sempurna pada modul DVWA, dan celah ini bisa dimanfaatkan untuk membaca data sensitif dari database, seperti password hash dan nama pengguna.



```
// Check Anti-CSRF token
checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

// Get input
$id = $_GET[ 'id' ];

// Was a number entered?
if(is_numeric( $id )) {
    $id = intval ($id);
    switch ($_DVWA['SQLI_DB']) {
        case MYSQL:
            // Check the database
            $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );
            $data->bindParam( ':id', $id, PDO::PARAM_INT );
            $data->execute();
            $row = $data->fetch();

            // Make sure only 1 result is returned
            if( $data->rowCount() == 1 ) {
                // Get values
                $first = $row[ 'first_name' ];
                $last  = $row[ 'last_name' ];

                // Feedback for end user
                echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
```

Gambar 10. *Impossible SQL Injection Source*

Kode tersebut merupakan implementasi dari tingkat keamanan *Impossible* pada DVWA karena telah menerapkan praktik keamanan terbaik, seperti penggunaan prepared statements untuk mencegah SQL Injection baik pada MySQL (dengan PDO) maupun SQLite, validasi input numerik menggunakan is_numeric() dan intval(), serta perlindungan terhadap serangan CSRF melalui verifikasi token (checkToken). Dengan kombinasi ini, input pengguna tidak dieksekusi secara langsung dalam query SQL dan setiap permintaan diverifikasi, sehingga serangan injeksi dan eksploitasi umum lainnya tidak dapat dilakukan.



Gambar 11. Uji keamanan *SQL Injection Impossible security*

Pada uji coba tersebut saat kode untuk merentas menggunakan SQL Injection, serangan akan gagal karena input telah divalidasi agar hanya berupa angka, dan query menggunakan prepared statements yang memisahkan data dari perintah SQL sehingga karakter berbahaya tidak dieksekusi. Selain itu, adanya perlindungan CSRF melalui token memastikan hanya permintaan sah dari pengguna yang diterima. Akibatnya, upaya injeksi tidak mempengaruhi query dan data sensitif tetap aman.

KESIMPULAN

Berdasarkan studi literatur dan hasil pengujian menggunakan Damn Vulnerable Web Application (DVWA), dapat disimpulkan bahwa SQL Injection merupakan ancaman serius terhadap keamanan aplikasi web, terutama pada sistem yang tidak menerapkan validasi dan sanitasi input secara memadai. Melalui simulasi serangan di DVWA, berbagai celah keamanan berhasil diidentifikasi, menunjukkan bagaimana penyerang dapat mengeksploitasi kelemahan tersebut untuk mengakses, mengubah, atau bahkan menghapus data dalam basis data. DVWA terbukti menjadi media yang efektif dalam pembelajaran dan pengujian keamanan web karena menyediakan lingkungan yang aman, terstruktur, dan terkontrol untuk memahami secara langsung mekanisme serangan serta strategi mitigasinya. Penggunaan DVWA memungkinkan peneliti maupun praktisi keamanan untuk mengembangkan pemahaman teknis yang mendalam serta merancang solusi preventif yang lebih kuat, seperti penggunaan prepared statements, validasi input yang ketat, dan pembaruan sistem secara berkala. Oleh karena itu, pemanfaatan DVWA dalam studi dan pelatihan keamanan siber menjadi sangat penting untuk meningkatkan kesadaran serta kemampuan dalam membangun aplikasi web yang lebih aman dari ancaman serangan SQL Injection.

REFERENSI

Williams, J. (2021). *Application Security Testing Best Practices*. CyberSec Insights Press.

OWASP. (2017). *OWASP Testing Guide v4*. Retrieved from https://owasp.org/www-project-testing/

Fadillah, R., & Wahyuni, S. (2020). Pengujian Keamanan Website terhadap Serangan SQL Injection menggunakan Metode OWASP. Jurnal Teknologi Informasi dan Komputer, 6(2), 101–108.

Nugroho, A., & Laksana, A. (2021). Analisis Keamanan Website Menggunakan Metode OWASP. Jurnal Teknologi dan

Sistem Informasi, 7(2), 123–131.
Prasetyo, I. P., & Nugraha, R. F. (2021). Evaluasi Kerentanan Sistem Informasi Akademik menggunakan OWASP Testing Guide. Jurnal Ilmu Komputer dan Informasi, 5(1), 45–53.
Pratama, R. A., & Sari, A. P. (2020). Pengujian Keamanan Website dengan Pendekatan OWASP Top 10. Jurnal Teknik Komputer AMIK BSI, 6(1), 45–52.
Rohman, A., & Arifin, Z. (2021). Identifikasi SQL Injection Menggunakan SQLMap pada Aplikasi Web. Jurnal Teknik Informatika dan Sistem Informasi, 8(2), 155–163.
Syahputra, M., & Rahman, F. (2022). Uji Penetrasi Website Menggunakan Standar OWASP. Jurnal Sistem Informasi dan Keamanan Siber, 3(1), 67–74.
Elu, A. M. (2013). Rancang Bangun Aplikasi Pendeteksian Vulnerability Structured Query Language (Sql) Injection Untuk Keamanan Website . *Jurnal Teknologi Informasi*, 111-124.
Yehezkiel Natanael, R. F. (2024). Analisis Keamanan Informasi Bagi Pengguna Website Menggunakan Kalilinux Melalui Teknik Sql Injection . *Tekinfo Vol. 25, No. 1 , April 2024*, 123-132.
Ashish Kumar, D. S. (2016). Analisis Berbagai Tingkat Penetrasi Dengan Teknik Injeksi Sql Melalui Dvwa. *Jurnal Teknologi Komputasi Dan Komunikasi Mutakhir* , 28-32.
Ela Nurelasari, D. G. (2024). Analisis Keamanan Sistem Website Menggunakan Metode Open Web Application Security Project (Owasp) Pada Simantep.Id. *Jati (Jurnal Mahasiswa Teknik Informatika)*, 3049-3054.
Yang, C., & Wu, D. (2024). *AdvSQLi: Generating adversarial SQL injections against real-world WAF-as-a-service*. arXiv preprint arXiv:2401.02615. https://arxiv.org/abs/2401.02615
Ren, X., Liu, Q., He, Y., Xu, X., & Liu, M. (2025). *Enhancing SQL injection detection and prevention using generative models*. International Journal of Advanced Computer Science and Applications (IJACSA), 16(2), 135–145. https://ojs.trigunadharma.ac.id/index.php/jis/article/view/6119 *(Contoh format jika dari jurnal lokal atau internasional)*
Li, J., Zhao, F., & Chen, Z. (2025). *Are your LLM-based Text-to-SQL models secure? Exploring SQL injection via backdoor attacks*. arXiv preprint arXiv:2503.05445. https://arxiv.org/abs/2503.05445
Rahman, A., & Putra, D. Y. (2024). *Uji penetrasi injeksi SQL terhadap celah keamanan database website menggunakan SQLMap*. Publikasi Jurnal Ilmu Sistem Elektronika, 5(1), 45–52. https://journal.pubmedia.id/index.php/pjise/article/view/2623
Fernandes, M., & Costa, R. (2025). *A hybrid approach for detecting SQL injection using machine learning techniques*. Proceedings of the 11th International Conference on Information Systems Security and Privacy (ICISSP 2025), 120–130. https://www.scitepress.org/PublishedPapers/2025/130781/