BAB 2
LANDASAN TEORI

2.1 SQL Injection

2.1.1 Definisi SQL Injection

SQL Injection (SQLi) merupakan kerentanan keamanan pada aplikasi web yang memungkinkan penyerang menyisipkan perintah SQL berbahaya ke dalam query database.

Menurut CISA (2024) yang dikutip dalam [10], SQL injection adalah "the insertion of user-supplied input directly into a SQL command, allowing threat actors to execute arbitrary queries". Hal ini terjadi akibat kurangnya validasi input, sehingga memungkinkan eksekusi perintah yang tidak sah.

Kerentanan ini masih menjadi ancaman utama karena praktik pengembangan yang tidak aman dan teknik serangan yang terus berkembang.

2.1.2 Cara Kerja SQL Injection

SQL injection terjadi ketika input pengguna tidak divalidasi dengan baik dan langsung digunakan dalam query SQL, sehingga memungkinkan penyerang menyisipkan perintah berbahaya ke dalam query.

Serangan dimulai dengan mengidentifikasi titik input seperti form login atau parameter URL, kemudian menguji dengan karakter khusus untuk melihat respons sistem [11]. Jika terdapat kelemahan, input dapat memodifikasi struktur query yang dijalankan oleh database.

Sebagai contoh, query autentikasi:

```
SELECT * FROM users WHERE username = 'admin' AND password = '123';
```

Dapat dimanipulasi menjadi:

```
SELECT * FROM users WHERE username = '' OR '1'='1' --
```

Kondisi tersebut selalu bernilai benar, sehingga penyerang dapat melewati proses autentikasi tanpa kredensial yang valid.

2.1.3 Penyebab SQL Injection

Penyebab utama SQL injection adalah kurangnya validasi dan sanitasi input, sehingga data dari pengguna langsung digunakan dalam query SQL tanpa pengamanan yang memadai [10, 5]. Praktik *dynamic string building* tanpa parameterisasi memperbesar risiko karena input pengguna digabungkan langsung ke dalam query.

Contoh kode yang rentan:

```
$query = "SELECT * FROM table WHERE field = '" + input + "'";
```

Tidak digunakannya *prepared statements* atau *parameterized queries* membuat database tidak dapat membedakan antara kode dan data [5]. Selain itu, penggunaan akun database dengan hak akses berlebih juga meningkatkan dampak serangan.

Faktor lain meliputi kurangnya kesadaran keamanan pengembang, penanganan error yang menampilkan informasi sensitif, serta konfigurasi sistem dan pengujian keamanan yang tidak memadai [11, 10].

2.1.4 Jenis SQL Injection

SQL Injection memiliki berbagai jenis teknik serangan yang digunakan untuk mengeksploitasi kerentanan pada aplikasi web. Setiap jenis memiliki karakteristik dan metode eksploitasi yang berbeda.

A Tautology-based SQL Injection (Authentication Bypass)

Tautology-based SQL injection merupakan teknik yang memanfaatkan kondisi logika yang selalu bernilai benar (true) untuk melewati autentikasi [5]. Serangan dilakukan dengan menyisipkan kondisi seperti `1=1` ke dalam query.

Contoh payload:

```
' OR 1=1 --
```

Payload tersebut mengubah klausa WHERE menjadi selalu benar, sehingga memungkinkan penyerang melewati proses login tanpa kredensial yang valid [10].

B Union-based SQL Injection)

Union-based SQL injection memungkinkan penyerang untuk mengakses data dari tabel lain dengan menggabungkan hasil query asli dan query berbahaya menggunakan operator UNION [5]. Teknik ini memerlukan pengetahuan tentang struktur query, seperti jumlah kolom dan tipe data.

Penyerang biasanya menentukan jumlah kolom dengan teknik seperti ORDER BY atau mencoba beberapa variasi UNION hingga tidak terjadi error.

Sebagai contoh:

```
SELECT product_name, price FROM products WHERE id=1;
```

Dapat dimanipulasi menjadi:

```
SELECT product_name, price FROM products WHERE id=1'
UNION SELECT username, password FROM users --';
```

Query tersebut akan menampilkan data dari tabel products sekaligus data sensitif seperti username dan password dari tabel users. Teknik ini banyak digunakan oleh tools otomatis seperti sqlmap untuk mengekstrak data dari database.

C Error-based SQL Injection

Error-based SQL injection memanfaatkan pesan error dari database untuk memperoleh informasi sensitif seperti nama database, tabel, atau kolom [10]. Serangan dilakukan dengan memicu error agar sistem mengungkapkan informasi internal.

Contoh payload untuk memicu error:

```
' AND (SELECT 1/0) --
```

Payload tersebut menyebabkan error (pembagian dengan nol) sehingga database dapat menampilkan informasi terkait struktur sistem.

Selain itu, penyerang juga dapat mengekstrak informasi melalui fungsi tertentu, misalnya:

```
' AND EXTRACTVALUE(1, CONCAT(0x7e, database())) --
```

Payload ini menghasilkan error yang berisi nama database yang sedang digunakan, sehingga informasi penting dapat diperoleh tanpa akses langsung ke data.

D Blind SQL Injection

Blind SQL Injection terjadi ketika aplikasi tidak menampilkan error atau output langsung, sehingga penyerang harus mengekstrak data secara tidak langsung.

1. Boolean-based Blind SQL Injection Boolean-based blind SQL injection digunakan ketika aplikasi tidak menampilkan hasil query maupun pesan error secara langsung [11]. Dalam kondisi ini, penyerang mengandalkan perbedaan respons aplikasi terhadap kondisi true dan false.

   Sebagai contoh:

   Payload:

   ```
   ' OR 1=1 --
   ```

   menghasilkan respons berhasil (true)

   Payload:

   ```
   ' OR 1=0 --
   ```

   menghasilkan respons gagal (false)

   Dengan membandingkan respons tersebut, penyerang dapat mengekstrak informasi secara bertahap, misalnya dengan menguji karakter tertentu dari data yang ditargetkan. Meskipun proses ini relatif lambat, teknik ini efektif ketika metode lain tidak dapat digunakan.

2. Time-based Blind SQL Injection. Time-based blind SQL injection memanfaatkan perbedaan waktu respons server untuk menentukan apakah suatu kondisi bernilai true atau false [10]. Teknik ini digunakan ketika aplikasi tidak memberikan perbedaan respons secara visual.

   Penyerang menggunakan fungsi delay seperti SLEEP() pada MySQL atau WAITFOR pada SQL Server.

   Sebagai contoh:

   ```
   ' OR IF(ASCII(SUBSTRING(DATABASE(),1,1))=109, SLEEP(5), 0) --
   ```

Jika respons server tertunda selama 5 detik, maka kondisi bernilai true, yang menunjukkan bahwa karakter pertama nama database sesuai dengan nilai yang diuji. Jika tidak ada delay, maka kondisi bernilai false. Dengan metode ini, penyerang dapat mengekstrak informasi database secara bertahap.

2.2 Penetration Testing

2.2.1 Definisi Penetration Testing

Penetration testing merupakan proses sistematis untuk mengevaluasi keamanan sistem dengan mensimulasikan serangan secara terkontrol guna mengidentifikasi kerentanan [12, 13].

Pengujian ini dilakukan secara legal dan etis oleh *ethical hacker* dengan izin resmi dari pemilik sistem, sehingga berbeda dengan serangan siber yang bersifat ilegal [14].

2.2.2 Metodologi Penetration Testing

Metodologi penetration testing terdiri dari lima fase utama yang dilakukan secara berurutan untuk memastikan proses pengujian yang sistematis dan efektif [12, 14].

1. **Reconnaissance (Information Gathering)**
   Tahap awal untuk mengumpulkan informasi tentang target, baik secara aktif maupun pasif, seperti alamat IP, domain, struktur sistem, dan informasi publik lainnya yang mendukung proses pengujian.

2. **Scanning (Pemindaian)**
   Tahap identifikasi teknis terhadap sistem target, seperti port terbuka, layanan yang berjalan, serta potensi kerentanan menggunakan tools seperti vulnerability scanner atau network mapper.

3. **Gaining Access (Eksploitasi)**
   Tahap eksploitasi kerentanan untuk memperoleh akses ke sistem, misalnya melalui teknik SQL Injection menggunakan tools seperti SQLMap.

4. **Maintaining Access**
   Tahap mempertahankan akses ke sistem untuk mensimulasikan serangan lanjutan dan menilai dampak keberlanjutan dari kerentanan.

5. **Covering Tracks**
   Tahap akhir berupa penghapusan jejak aktivitas pengujian serta dokumentasi hasil dan rekomendasi perbaikan keamanan.

2.2.3 Jenis Penetration Testing

1. **White Box Testing**
   Pendekatan dimana tester memiliki akses penuh terhadap sistem, termasuk kode sumber dan dokumentasi, sehingga pengujian dapat dilakukan secara lebih cepat dan terarah [13, 14].

2. **Black Box Testing**
   Pendekatan tanpa informasi awal tentang sistem, yang mensimulasikan serangan dari pihak eksternal dan memberikan gambaran kondisi keamanan yang realistis [13, 14].

3. **Gray Box Testing**
   Pendekatan dengan akses terbatas terhadap sistem, menggabungkan efisiensi white box dan realisme black box dalam proses pengujian [14].

4. **Berdasarkan Target dan Lokasi**
   Penetration testing juga dibedakan menjadi *external testing*, *internal testing*, dan *web application testing* yang berfokus pada kerentanan aplikasi web seperti SQL Injection [13, 14].

2.3 Vulnerable Machine

Aplikasi web yang sengaja dirancang rentan merupakan aplikasi yang dengan sengaja dibuat memiliki celah keamanan untuk keperluan edukasi, pelatihan, dan pengujian keamanan aplikasi web [15]. Aplikasi jenis ini menyediakan lingkungan terisolasi yang memungkinkan eksploitasi dilakukan secara mendalam tanpa menimbulkan risiko hukum maupun kerusakan pada sistem produksi, sehingga banyak digunakan sebagai media pengujian dalam penelitian keamanan aplikasi web [15, 16].

Sebagai contoh, *Damn Vulnerable Web Application* (DVWA) merupakan salah satu aplikasi rentan yang umum digunakan dalam pengujian kerentanan SQL Injection, karena menyediakan berbagai skenario kerentanan dengan tingkat

keamanan yang dapat dikonfigurasi sehingga memungkinkan evaluasi teknik keamanan secara menyeluruh [15, 16].

2.4 SQL MAP

2.4.1 Definisi SQL MAP

SQLMap adalah sebuah tool penetration testing open-source yang dirancang khusus untuk mendeteksi dan mengeksploitasi kerentanan SQL injection pada aplikasi web. T. Saputra et al [17] menjelaskan bahwa SQLMap merupakan alat yang efektif untuk menguji kerentanan terhadap serangan SQL Injection dengan kemampuan otomatis dan mendalam dalam eksploitasi. A. Riyanti et al [18] menyebutkan SQLMap sebagai salah satu alat yang cukup efektif untuk menembus keamanan situs web yang berasal dari Kali Linux, memungkinkan penyerang untuk dengan mudah mengambil alih data autentikasi penting seperti username dan password.

2.4.2 Fungsi SQLMAP

SQLMap memiliki berbagai fungsi dan kemampuan komprehensif dalam pengujian keamanan aplikasi web. T. Saputra et al [17] menjelaskan bahwa SQLMap dapat menangani berbagai teknik SQL Injection, termasuk boolean-based blind dan error-based injection, dengan kemampuan tidak hanya mendeteksi kerentanan tetapi juga mengeksploitasinya untuk penilaian yang lebih menyeluruh terhadap ancaman potensial. M. Begum et al [19] menambahkan bahwa SQLMap dapat diintegrasikan dengan model machine learning Random Forest untuk meningkatkan akurasi deteksi kerentanan dan memberikan solusi yang scalable. Gilang Ryan Fernandes et al [20] menekankan bahwa SQLMap dapat menemukan dan mengambil semua database yang tersimpan pada server website serta dapat berjalan secara otomatis.

2.4.3 Cara Kerja SQLMAP

SQLMap bekerja melalui proses otomatis yang sistematis dalam mendeteksi kerentanan SQL injection. A. Riyanti et al [18] menjelaskan bahwa SQLMap memaksa server web mengeluarkan informasi dari database, termasuk tabel, kolom, dan isi data melalui skrip query SQL khusus yang ditulis dalam bahasa

pemrograman Python. T. Saputra et al [17] menggambarkan bahwa SQLMap dapat mengakses konten tabel dari database menggunakan command seperti "python sqlmap.py -u -D database_name -T products -dump" untuk menampilkan isi tabel produk. Gilang Ryan Fernandes et al [20] menjelaskan bahwa SQLMap dapat dijalankan pada platform mobile melalui Termux, menunjukkan fleksibilitas penggunaan tool ini di berbagai environment.

2.4.4 Fitur SQLMap

SQLMap menyediakan berbagai fitur untuk pengujian keamanan database. Fitur utama meliputi command "–dbs" untuk mengidentifikasi database, "–tables" untuk menampilkan daftar tabel, "–columns" untuk melihat struktur kolom, dan "–dump" untuk mengekstrak data dari tabel tertentu [17]. Tool ini dapat menampilkan daftar tabel dan kolom pada database target melalui teknik eksploitasi [18]. SQLMap juga dapat diotomatisasi dengan input numerik sederhana untuk meningkatkan usability [19]. Selain itu, SQLMap dapat dijalankan pada berbagai platform, termasuk platform mobile melalui Termux Gilang Ryan Fernandes et al., 2024.

2.4.5 Kelebihan dan Keterbatasan SQLMap

Kelebihan SQLMap telah terbukti melalui berbagai penelitian. T. Saputra et al [17] menyatakan bahwa SQLMap banyak digunakan karena dapat memberikan solusi yang robust dalam melindungi aplikasi web. A. Riyanti et al[18]menunjukkan bahwa SQLMap dapat dilakukan dengan relatif mudah dan efektif, bahkan oleh orang awam tanpa pemahaman hacking karena tool ini berjalan secara otomatis. M. Begum et al [19] menekankan bahwa otomatisasi SQLMap dapat membuat penetration testing lebih cepat dan dapat diakses bahkan oleh pengguna non-teknis. Hazem M. Harb et al [21] mengidentifikasi SQLMap sebagai salah satu tool deteksi dan pencegahan SQL injection yang penting dengan kelebihan dan kelemahan yang perlu dipertimbangkan.

Namun, SQLMap juga memiliki keterbatasan. A. Riyanti et al [18] mencatat keterbatasan dalam cakupan jenis serangan yang diuji dan lingkungan pengujian yang mungkin tidak sepenuhnya merefleksikan kondisi dunia nyata. Hazem M. Harb et al [21] menyoroti bahwa meskipun SQLMap efektif, masih terdapat kelemahan yang perlu diatasi dalam implementasinya.

2.5 Common Vulnerability Scoring System (CVSS v3.1)

2.5.1 Definisi CVSS

Common Vulnerability Scoring System (CVSS) merupakan suatu kerangka kerja terbuka (open framework) yang digunakan untuk menggambarkan karakteristik serta mengukur tingkat keparahan (severity) dari kerentanan pada perangkat lunak, perangkat keras, maupun firmware. CVSS menghasilkan skor numerik yang merepresentasikan tingkat risiko suatu kerentanan dibandingkan dengan kerentanan lainnya [8].

CVSS dirancang untuk menyediakan metode penilaian yang standar, transparan, dan independen terhadap vendor maupun platform, sehingga dapat digunakan secara luas oleh organisasi, peneliti, maupun praktisi keamanan informasi dalam melakukan analisis kerentanan [8].

Output utama dari CVSS berupa skor numerik (0.0 – 10.0) yang menunjukkan tingkat keparahan dan Vector string, yaitu representasi tekstual dari metrik yang digunakan dalam perhitungan skor [8].

2.5.2 Struktur CVSS v3.1

CVSS terdiri dari tiga kelompok metrik utama yang digunakan untuk mengevaluasi kerentanan dari berbagai perspektif, yaitu [8]:

1. **Base Metrics**
   Metrik dasar yang merepresentasikan karakteristik intrinsik kerentanan yang bersifat tetap dan tidak bergantung pada waktu maupun lingkungan. Digunakan untuk menghitung Base Score sebagai acuan utama tingkat keparahan.

2. **Temporal Metrics**
   Metrik yang mempertimbangkan faktor yang berubah seiring waktu, seperti ketersediaan exploit dan patch, sehingga menyesuaikan Base Score dengan kondisi aktual.

3. **Environmental Metrics**
   Metrik yang menyesuaikan skor berdasarkan kondisi spesifik organisasi, seperti nilai aset dan dampak terhadap bisnis, sehingga menghasilkan penilaian risiko yang lebih kontekstual.

2.5.3 Base Metrics

Base Metrics merupakan komponen utama dalam CVSS v3.1 yang merepresentasikan karakteristik dasar kerentanan yang bersifat tetap dan tidak dipengaruhi oleh waktu maupun lingkungan. Metrik ini digunakan untuk menghitung Base Score sebagai acuan tingkat keparahan kerentanan [8].

Base Metrics terdiri dari dua komponen, yaitu *exploitability* dan *impact*. Exploitability mengukur kemudahan eksploitasi berdasarkan faktor seperti attack vector, attack complexity, privileges required, dan user interaction, sedangkan impact mengukur dampak terhadap confidentiality, integrity, dan availability. Kombinasi keduanya menghasilkan nilai Base Score yang merepresentasikan tingkat risiko kerentanan [8].

A Exploitability Metrics

Exploitability Metrics merupakan kelompok metrik dalam Base Metrics yang digunakan untuk mengukur tingkat kemudahan suatu kerentanan untuk dieksploitasi oleh penyerang. Penilaian pada metrik ini didasarkan pada beberapa parameter yang berkaitan dengan proses eksploitasi, yaitu sebagai berikut:

1. Attack Vector menggambarkan sejauh mana lokasi atau jalur akses yang dibutuhkan penyerang untuk mengeksploitasi kerentanan. Semakin luas jangkauan akses (misalnya melalui internet), maka semakin tinggi nilai keparahannya karena jumlah potensi penyerang lebih besar [8].

   (a) Network (N)

   Kerentanan dapat dieksploitasi melalui jaringan, termasuk internet publik. Eksploitasi tidak memerlukan akses fisik atau lokal, sehingga memungkinkan penyerang dari lokasi mana pun untuk melakukan serangan. Contohnya adalah serangan SQL Injection pada aplikasi web yang dapat diakses secara online.

   (b) Adjacent (A)

   Eksploitasi hanya dapat dilakukan dalam jaringan yang berdekatan secara logis, seperti jaringan lokal (LAN), Wi-Fi, atau VPN tertentu. Penyerang harus berada dalam segmen jaringan yang sama atau memiliki akses ke jaringan tersebut.

(c) Local (L)

Penyerang harus memiliki akses lokal ke sistem target, baik secara langsung maupun melalui koneksi remote seperti SSH. Dalam beberapa kasus, eksploitasi juga dapat bergantung pada interaksi pengguna, seperti membuka file berbahaya.

(d) Physical (P)

Eksploitasi membutuhkan akses fisik langsung ke perangkat, misalnya melalui USB atau manipulasi perangkat keras. Jenis ini memiliki tingkat risiko lebih rendah karena keterbatasan akses.

2. Attack Complexity (AC)

Attack Complexity menunjukkan tingkat kesulitan eksploitasi berdasarkan kondisi eksternal yang berada di luar kendali penyerang. Parameter ini tidak mempertimbangkan interaksi pengguna (dibahas pada UI) [8].

(a) Low (L)

Eksploitasi dapat dilakukan secara langsung tanpa kondisi khusus. Penyerang dapat mengulangi serangan dengan tingkat keberhasilan yang tinggi tanpa memerlukan persiapan tambahan.

(b) High (H)

Eksploitasi memerlukan kondisi tertentu, seperti:

i. Informasi spesifik mengenai target (konfigurasi sistem, token, atau parameter tertentu)
ii. Lingkungan yang harus disiapkan terlebih dahulu
iii. Teknik khusus seperti race condition atau man-in-the-middle

Hal ini membuat eksploitasi tidak selalu berhasil dan membutuhkan usaha tambahan.

3. Privileges Required (PR)

Privileges Required menunjukkan tingkat hak akses yang harus dimiliki penyerang sebelum eksploitasi dilakukan [8].

(a) None (N)

Penyerang tidak memerlukan autentikasi atau akses awal ke sistem. Kerentanan dapat langsung dieksploitasi oleh pihak eksternal tanpa login.

(b) Low (L)

Penyerang membutuhkan hak akses sebagai pengguna biasa. Hak akses ini biasanya terbatas dan tidak mencakup kontrol penuh terhadap sistem.

(c) High (H)

Penyerang membutuhkan hak akses tingkat tinggi, seperti administrator, yang memberikan kontrol luas terhadap sistem, termasuk konfigurasi dan data penting.

4. User Interaction (UI) User Interaction menunjukkan apakah eksploitasi membutuhkan keterlibatan pengguna selain penyerang [8].

(a) None (N)

Eksploitasi dapat dilakukan sepenuhnya oleh penyerang tanpa bantuan atau tindakan dari pengguna lain.

(b) Required (R)

Eksploitasi memerlukan tindakan pengguna, seperti:

i. Mengklik tautan berbahaya
ii. Membuka file atau dokumen
iii. Menjalankan aplikasi tertentu

Hal ini biasanya ditemukan pada serangan berbasis social engineering.

5. Scope

Scope menunjukkan apakah dampak dari eksploitasi tetap berada dalam satu sistem keamanan atau meluas ke sistem lain [8].

(a) Unchanged (U)

Dampak eksploitasi hanya terbatas pada komponen atau sistem yang sama. Tidak terjadi pelanggaran batas keamanan.

(b) Changed (C) Eksploitasi menyebabkan dampak pada sistem lain di luar kontrol awal, seperti:

i. Aplikasi yang berhasil mengakses database sistem lain
ii. Eskalasi dari aplikasi ke sistem operasi

Hal ini menunjukkan adanya pelanggaran boundary keamanan dan meningkatkan tingkat keparahan.

B Impact Metrics

Impact Metrics mengukur dampak yang ditimbulkan terhadap sistem setelah eksploitasi berhasil dilakukan. Dampak ini dinilai berdasarkan tiga aspek utama [8]:

1. Confidentiality (C)

   Confidentiality mengukur sejauh mana kerahasiaan informasi terganggu akibat eksploitasi.

   (a) High (H)

   Terjadi kebocoran total terhadap data sensitif. Penyerang dapat mengakses seluruh informasi penting, seperti kredensial pengguna atau data rahasia.

   (b) Low (L)

   Terjadi kebocoran sebagian data. Informasi yang diperoleh terbatas dan tidak sepenuhnya berdampak kritis.

   (c) None (N)

   Tidak ada informasi yang berhasil diakses oleh penyerang.

2. Integrity (I)

   Integrity mengukur sejauh mana data dapat dimodifikasi oleh penyerang.

   (a) High (H)

   Penyerang dapat mengubah seluruh data atau konfigurasi sistem secara bebas, yang dapat menyebabkan kerusakan serius.

   (b) Low (L)

   Penyerang berpotensi atau dapat melakukan perubahan terbatas dan tidak berdampak besar terhadap sistem secara keseluruhan.

   (c) None (N)

   Tidak ada perubahan terhadap data atau sistem.

3. Availability (A)

   Availability mengukur dampak terhadap ketersediaan layanan atau sistem.

(a) High (H)

Sistem atau layanan tidak dapat diakses sama sekali (down total), baik sementara maupun permanen.

(b) Low (L)

Terjadi penurunan performa atau gangguan sebagian, namun sistem masih dapat digunakan.

(c) None (N)

Tidak ada gangguan terhadap ketersediaan sistem.

2.5.4 Qualitative Severity Rating Scale

Setelah nilai Base Score diperoleh, nilai tersebut kemudian dikategorikan ke dalam tingkat keparahan kualitatif (Qualitative Severity Rating Scale). Kategori ini digunakan untuk mempermudah interpretasi tingkat keparahan kerentanan. Adapun pembagian kategori berdasarkan standar CVSS v3.1 adalah sebagai berikut [8]:

Tabel 2.1. Qualitative Severity Rating Scale

Skor CVSS	Tingkat Keparahan
0.0	None
0.1 – 3.9	Low
4.0 – 6.9	Medium
7.0 – 8.9	High
9.0 – 10.0	Critical

Beberapa penelitian telah menerapkan CVSS v3.1 sebagai standar penilaian keparahan kerentanan pada aplikasi web. Ferzha Putra Utama [22] mengidentifikasi kerentanan SQL Injection pada halaman login sistem informasi akademik dengan skor CVSS 7.4 (High), di mana kerentanan tersebut memungkinkan penyerang memperoleh seluruh informasi yang tersimpan dalam database, termasuk data kredensial pengguna. Penelitian ini juga secara eksplisit menetapkan nilai Confidentiality yang tinggi karena dampak eksploitasi yang dihasilkan berupa akses penuh terhadap seluruh data sensitif dalam database.

Dhira Wahyu Febrian [23] menemukan kerentanan SQL Injection pada website perguruan tinggi dengan skor CVSS 9.0 (Critical), di mana penyerang dapat membaca, memodifikasi, atau menghapus data pada basis data melalui

injeksi perintah SQL yang tidak tervalidasi. Selain itu, ditemukan pula kerentanan Insecure Direct Object Reference (IDOR) dengan skor CVSS 7.5 (High) yang memungkinkan penyerang mengakses atau memodifikasi data milik pengguna lain tanpa otorisasi yang memadai.

Diana Rohmaniah [24] menemukan kerentanan Directory Listing yang setelah dihitung menggunakan CVSS menghasilkan kategori High, dengan dampak berupa kemampuan penyerang mengakses data sensitif berupa kredensial pengguna yang tersimpan dalam direktori website yang tidak terlindungi.

2.5.5 Vector String

Vector String merupakan representasi tekstual dari parameter-parameter yang digunakan dalam perhitungan CVSS. Vector String memudahkan dalam mendeskripsikan karakteristik kerentanan secara ringkas dan terstandarisasi.

Format umum Vector String pada CVSS v3.1 adalah sebagai berikut [8]:

$$CVSS:3.1/AV:X/AC:X/PR:X/UI:X/S:X/C:X/I:X/A:X \quad (2.1)$$

Keterangan:

- AV (Attack Vector)
- AC (Attack Complexity)
- PR (Privileges Required)
- UI (User Interaction)
- S (Scope)
- C (Confidentiality)
- I (Integrity)
- A (Availability)

Sebagai contoh, vector string untuk kerentanan SQL Injection dapat dituliskan sebagai berikut [8]:

$$CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L \quad (2.2)$$

2.5.6 Base Metrics Equations

Perhitungan nilai Base Score pada CVSS v3.1 didasarkan pada dua komponen utama, yaitu Exploitability dan Impact [8].

A Exploitability Score

Exploitability Score dihitung menggunakan persamaan berikut:

$$Exploitability = 8.22 \times AV \times AC \times PR \times UI \tag{2.3}$$

B Impact Score

Impact Score dihitung berdasarkan nilai Confidentiality (C), Integrity (I), dan Availability (A):

$$ISC_{Base} = 1 - (1 - C) \times (1 - I) \times (1 - A) \tag{2.4}$$

Jika Scope tidak berubah (Unchanged), maka:

$$Impact = 6.42 \times ISC_{Base} \tag{2.5}$$

Jika Scope berubah (Changed), maka:

$$Impact = 7.52 \times (ISC_{Base} - 0.029) - 3.25 \times (ISC_{Base} - 0.02)^{15} \tag{2.6}$$

C Base Score

Nilai akhir Base Score dihitung sebagai berikut:
Jika Impact ≤ 0:

$$BaseScore = 0 \tag{2.7}$$

Jika Scope Unchanged:

$$BaseScore = \text{RoundUp}(\min(Impact + Exploitability, 10)) \tag{2.8}$$

Jika Scope Changed:

$$BaseScore = \text{RoundUp}(\min(1.08 \times (Impact + Exploitability), 10)) \quad (2.9)$$

D Metrics Value

Setiap parameter pada Base Metrics memiliki nilai numerik yang digunakan dalam perhitungan Base Score. Nilai-nilai tersebut berdasarkan spesifikasi resmi CVSS v3.1 [8] adalah sebagai berikut:

Tabel 2.2. Nilai Numerik Base Metrics CVSS v3.1

Metric	**Metric Value**	**Numerical Value**
Attack Vector (AV)	Network	0.85
	Adjacent	0.62
	Local	0.55
	Physical	0.20
Attack Complexity (AC)	Low	0.77
	High	0.44
Privileges Required (PR)	None	0.85
	Low	0.62 (0.68 jika Scope Changed)
	High	0.27 (0.50 jika Scope Changed)
User Interaction (UI)	None	0.85
	Required	0.62
Confidentiality / Integrity / Availability (C/I/A)	High	0.56
	Low	0.22
	None	0.00

2.6 Systematic Literature Review

Tabel 2.3. Systematic Literature Review SQL Injection

Penulis	Judul	Tujuan	Hasil	Metode	Gap	Kelebihan	Kekurangan
Neupane (2025) [5]	Detecting and Mitigating SQL Injection Vulnerabilities in Web Applications	Mengidentifikasi dan memitigasi kerentanan SQL Injection pada aplikasi berbasis PHP-MySQL	Ditemukan kerentanan SQLi pada parameter GET. SQLMap berhasil melakukan ekstraksi data dan bypass autentikasi. Setelah dilakukan patching, kerentanan tidak lagi terdeteksi	Pendekatan tiga fase: preparation, execution, dan patching dengan kombinasi pengujian otomatis dan manual	Tidak menggunakan standar penilaian risiko seperti CVSS	Metodologi terstruktur serta mencakup proses mitigasi dan verifikasi ulang	Tidak terdapat pengukuran tingkat keparahan dan analisis teknis yang mendalam
Southeast Con (2018)[9]	Vulnerability Analysis of CMS to SQL Injection Using SQLMAP	Menganalisis potensi kerentanan SQL Injection pada CMS (WordPress, Drupal, Joomla)	Tidak ditemukan eksploitasi SQLi karena adanya perlindungan WAF/IPS, meskipun terdapat potensi kerentanan pada beberapa parameter	Pengujian menggunakan Kali Linux dengan tools SQLMap dan Nikto pada lingkungan LAMP	Tidak ditemukan eksploitasi SQLi yang berhasil sehingga analisis risiko tidak dapat dilakukan	Menggunakan berbagai tools dan membandingkan beberapa platform CMS	Tidak dilakukan pengujian bypass WAF sehingga tidak dapat diketahui dampak nyata SQLi apabila proteksi berhasil ditembus
Kholilul Adrian et al. (2025) [4]	Integrating CVSS, OWASP, and APPI for SQLi Risk Analysis	Menganalisis risiko SQL Injection dari aspek teknis, bisnis, dan hukum	SQL Injection berhasil dieksploitasi dengan skor CVSS 9.1 (Critical) serta mengakibatkan kebocoran data sensitif	Pendekatan mixed-method dengan integrasi SQLMap, CVSS, OWASP, dan APPI	Analisis CVSS tidak dibahas secara rinci pada setiap parameter	Pendekatan komprehensif dengan hasil kuantitatif yang jelas	Metode kompleks dan kurang fokus pada analisis teknis secara mendalam