# Mengatasi Keterbatasan KMP dalam Deteksi Serangan Menggunakan Algoritma Aho-Corasick

## Pendahuluan

Serangan terhadap aplikasi web, seperti injeksi Structured Query Language (SQL) dan Cross-Site Scripting (XSS), terus menjadi ancaman utama bagi integritas dan keamanan data. Ketika aplikasi web memproses input pengguna yang belum divalidasi secara memadai, peretas dapat mengeksploitasi kerentanan tersebut untuk mendapatkan akses tidak sah, mengubah, atau merusak data. Oleh karena itu, pengembangan teknik deteksi yang lebih efektif menjadi prioritas utama dalam bidang keamanan siber.

Salah satu pendekatan yang telah banyak digunakan dalam mendeteksi pola berbahaya adalah Algoritma Knuth-Morris-Pratt (KMP). Meskipun KMP efisien dalam mencocokkan string dengan kompleksitas waktu O(n + m), kinerjanya menurun ketika dihadapkan dengan banyak pola, seperti dalam kasus deteksi berbagai bentuk serangan SQL dan XSS. Keterbatasan ini muncul karena KMP dirancang untuk mencocokkan satu pola dengan teks, yang membuatnya kurang optimal untuk menangani banyak pola secara bersamaan.

Sebagai solusi, penelitian baru ini mengusulkan penggunaan Algoritma Aho-Corasick, yang dirancang untuk pencocokan string dengan banyak pola secara simultan. Algoritma Aho-Corasick menggunakan automata berorientasi pohon (trie) untuk membangun struktur data yang dapat memproses banyak pola dengan efisien. Dengan kompleksitas waktu O(n + k), di mana *n* adalah panjang teks dan *k* adalah jumlah pola yang cocok, algoritma ini memungkinkan deteksi lebih cepat dalam situasi di mana banyak pola perlu dipantau, seperti dalam pencegahan injeksi SQL dan XSS.

## Methodology

### Research Framework

### Penelitian ini berfokus pada implementasi dan evaluasi performa algoritma Knuth-Morris-Pratt (KMP) dan Aho-Corasick (AC) dalam bahasa pemrograman C++. Penelitian ini bertujuan untuk menganalisis serta membandingkan kompleksitas waktu (*time complexity*) dari kedua algoritma tersebut ketika digunakan dalam proses pencocokan string. Untuk mencapai tujuan ini, implementasi dilakukan pada dataset yang telah ditentukan guna memberikan hasil yang terukur dan relevan.

### Algorithm Implementation

**Knuth-Morris-Pratt (KMP):** Algoritma KMP diimplementasikan dengan dua tahap utama yang bertujuan untuk meningkatkan efisiensi pencocokan string. Tahap pertama adalah *Preprocessing Pattern*, yang membentuk tabel *partial match* untuk memetakan pola sehingga meminimalkan pergeseran pencocokan yang tidak perlu. Tahap kedua adalah *Pattern Matching*, yang melakukan pencocokan string secara efektif dengan memanfaatkan tabel yang telah dibuat sebelumnya.

**Aho-Corasick (AC):** Implementasi algoritma AC mencakup tiga langkah utama. Langkah pertama adalah *Trie Construction*, di mana struktur trie dibangun berdasarkan pola yang diberikan untuk memungkinkan pencarian simultan. Langkah kedua adalah *Failure Link Creation*, di mana tautan gagal (*failure links*) ditambahkan ke trie untuk memfasilitasi navigasi saat terjadi ketidakcocokan. Langkah terakhir adalah *Pattern Matching*, yang menggunakan automata deterministik untuk mencocokkan string secara efisien dengan kompleksitas waktu linier.

### Experimental Setup

**Dataset Uji**: Dataset yang digunakan dalam penelitian ini berupa string sederhana dengan karakter acak, yang dirancang untuk merepresentasikan pola (*patterns*) dan teks (*text*).

**Pengukuran Waktu Eksekusi**: Waktu eksekusi algoritma diukur menggunakan fungsi *clock()* yang tersedia di pustaka <time.h> dalam bahasa pemrograman C++. Fungsi ini digunakan untuk menghitung durasi proses algoritma dengan resolusi tinggi, sehingga memberikan hasil yang akurat untuk membandingkan performa KMP dan Aho-Corasick.

### Testing Scenarios

...................................