# Deteksi Serangan SQL *Injection* Menggunakan *Hidden Markov Model*

**Pramono[1], Andi Sunyoto[2], Eko Pramono[3]**
Magister Teknik Informatika, Universitas Amikom Yogyakarta
Email: pramono.23@students.amikom.ac.id[1], andi@amikom.ac.id[2], eko.p@amikom.ac.id[3]

**Abstrak**

Serangan aplikasi web terus meningkat jumlahnya dan dalam tingkat keparahan. Information besar tersedia di internet memotivasi penyerang untuk melakukan serangan jenis baru. Di dalam konteks, penelitian intensif tentang keamanan aplikasi web telah dilakukan. Serangan berbahaya yang menargetkan web aplikasi adalah Structured Query Language Injection (SQLI). Serangan ini merupakan ancaman serius bagi web aplikasi. Beberapa pekerjaan penelitian melakukan cara untuk mengurangi serangan ini baik dengan mencegahnya dari awal tahap atau mendeteksinya saat itu terjadi. Dalam tulisan ini, kami sajikan gambaran umum tentang serangan injeksi SQL dan klasifikasi dari solusi deteksi dan pencegahan yang baru diusulkan. Dalam penelitian ini kami menggunakan Hidden Markov Model (HMM) untuk melakukan metode deteksi dan pencegahan dari serangan SQLI untuk mengurangi serangan ini khususnya yang didasarkan pada ontologi dan pembelajaran mesin.

**Kata kunci:** HMM, SQL Injection, Web Security

**Abstract**

*Web application attacks continue to increase in number and in severity. The massive information available on the internet motivates hackers to launch new types of attacks. In that context, intensive research on web application security has been carried out. A malicious attack targeting web applications is Structured Query Language Injection (SQLI). This attack is a serious threat to web applications. Some research work has been*

*done to reduce these attacks either by preventing them from starting or detecting them as they occur. In this paper, we present an overview of SQL injection attacks and a classification of the newly proposed detection and prevention solutions. In this study we use the Hidden Markov Model (HMM) to perform detection and prevention methods from SQLI attacks to reduce these attacks, especially those based on ontology and machine learning.*

**Keywords:** *HMM, SQL Injection, Web Security*

## A. PENDAHULUAN

Karena pesatnya pertumbuhan minat dalam keamanan informasi, sistem deteksi intrusi telah dipelajari secara ekstensif di dunia akademis dan industri dalam beberapa dekade terakhir. National Benchmarking and Innovation Foundation (NIST) mendefinisikan deteksi intrusi sebagai "proses pemantauan peristiwa yang terjadi dalam sistem atau jaringan komputer dan menganalisisnya untuk mencari tanda-tanda gangguan. Ini adalah proses untuk mencoba menghancurkan kerahasiaan, integritas, dan ketersediaan. Atau melewati mekanisme atau jaringan keamanan komputer. Sistem yang memproses atau secara otomatis melakukan deteksi intrusi disebut sistem deteksi intrusi (IDS) (Kizza et al., 2011). Pada 2017, 953.000 serangan dunia maya diblokir setiap hari, Ada peningkatan 611.000 serangan yang diblokir per hari di tahun sebelumnya. Menurut Aplikasi Web Terbuka (OWASP), kerentanan Injeksi masih merupakan kerentanan yang paling umum diaplikasi web (*Owasp.Org*, n.d.-a).

Di antara berbagai ancaman keamanan terhadap aplikasi web, SQLIA telah dominan dalam 15 tahun terakhir (*Owasp.Org*, n.d.-b). Di antara berbagai ancaman keamanan terhadap aplikasi web, SQLIA telah dominan dalam 15 tahun terakhir. Ini adalah teknik yang relatif sederhana dan mudah dipahami banyak sumber daya di Internet. Terjadi masalah dari

menggunakan data input yang tidak tepercaya hingga membuat SQL dinamis Queri tanpa verifikasi yang tepat. Penggunaan umum dapat di temukan kerentanan, penyerang dapat mengekstrak, mengubah, atau menghapus file konten dalam database back-end. Pernyataan SQL juga memungkinkan penyerang untuk mendapatkan hak administrator atas data base juga, yang berarti penyerang dapat menambah, mengedit, atau menghapus data dengan tidak ada yang menghentikan mereka untuk melakukannya. Kotak log in atau in put adalah tempatnya tempat pernyataan SQL diketik, yang kemudian mengirim kode berbahaya ke dijalankan di server yang menghosting database (Tajpour et al., 2012). Beberapa jenis serangan injeksi SQL dapat digunakan. Bergantung pada tujuan penyerang, injeksi SQL akan dikirim ke server sekaligus atau sekaligus. Setelah penyerang memasuki database, mereka dapat memaksakan Ancaman dari berbagai sudut. Penyerang memiliki akses ke informasi sensitif, sehingga mereka dapat menghancurkan dan mengubah informasi tersebut (Loughran et al., 2018).

Memang, sebagian besar aplikasi web saat ini menggunakan backend Database menyimpan data yang dikumpulkan dari pengguna dan / atau Ambil informasi yang dipilih oleh pengguna. Interaksi untuk pengguna, ini biasanya melalui formulir dan queri. Peretas mencoba memanfaatkan fitur ini dengan injeksi Kode berbahaya yang dimasukkan oleh pengguna ini, yang akan digunakan nanti untu queri SQL. Verifikasi yang tidak tepat masukan pengguna dapat menyebabkan serangan SQLI yang berhasil, dan Oleh karena itu, dapat menimbulkan konsekuensi bencana, seperti Hapus database atau kumpulkan data sensitif, dan Data rahasia klien aplikasi web (Kar et al., 2016)

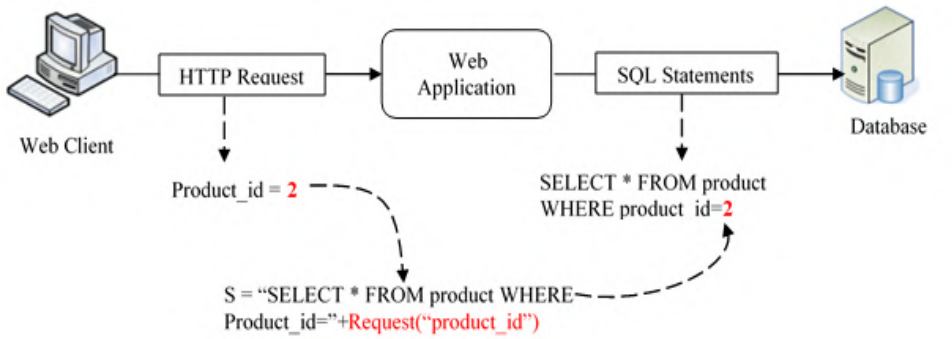
Selain penelitian ekstensif tentang SQLIA, banyak metode memiliki beberapa keterbatasan dan tidak dapat mempromosikan aplikasinya. Pada paper

ini, Kami mengusulkan metode deteksi baru menggunakan Hidden Markov Model (HMM) dan analisis log khusus untuk mendeteksi SQLIA, SQLIA urutan kedua dan serangan dengan risiko serupa pelaku. Kemudian kami juga mengusulkan teori unit perilaku di log, emproses untuk melindungi niat pengguna dan mengurangi kesalahan klasifikasi.

## B. METODE

Metode yang dilakukan agar tujuan paper ini tercapai adalah dengan melakukan literasi dimulai dengan Tahap pertama mempelajari tentang bagaimana Serangan SQL Injection Bekerja. Saat ini, sebagian besar aplikasi web menggunakan desain berlapis-lapis, biasanya dengan tiga level: presentasi, pemrosesan, dan level data. Menunjukkan bahwa level adalah antarmuka Web HTTP, level aplikasi mengimplementasikan fungsi perangkat lunak, dan level data jaga agar data tetap terstruktur dan tanggapi permintaan tingkat aplikasi (Tajpour et al., 2012).

Pada saat yang sama, perusahaan besar yang mengembangkan sistem manajemen data base berbasis SQL sangat bergantung pada perangkat keras untuk memastikan kinerja yang diperlukan (- et al., 2010). Injeksi adalah serangan di mana penyerang menambahkan kode "Bahasa Kueri Terstruktur" ke kotak masukan formulir web dapatkan akses atau ubah data. Kerentanan *SQL Injection* bisa saja penyerang untuk mengalirkan perintah langsung ke web Basis data yang membentuk dasar aplikasi dan menghancurkan fungsionalitas atau kerahasiaan.




**Gambar 1. Bentuk Serangan SQL Injection Bekerja**

SQLI adalah teknik peretasan penyerang menambah atau menyembunyikan pernyataan SQL melalui bidang input aplikasi jaringan parameter yang digunakan untuk mengakses sumber daya. Kurangnya validasi input dalam aplikasi web dapat menyebabkan kesuksesan hacker. Dalam contoh berikut, kami akan mengasumsikan bahwa aplikasi web menerima permintaan HTTP dari klien sebagai input dan menghasilkan SQL pernyataan tersebut berfungsi sebagai output dari server database back-end. Misalnya, administrator akan diautentikasi setelah mengetik berikut ini: ID karyawan = 112, sandi = admin. Gambar 1 menggambarkan a masuk oleh pengguna jahat yang mengeksploitasi kerentanan injeksi SQL(Charania & Vyas, 2016). hal ini pada dasarnya dibagi menjadi tiga tahap:

1) Penyerang mengirimkan permintaan HTTP berbahaya ke aplikasi web
2) Membuat pernyataan SQL
3) Kirim pernyataan SQL ke database back-end

Tahap kedua mempelajari tentang Hidden Markov Models dan mempelajari bagaimana penggunan Hidden Markov Models kemudia Meneliti contoh masalah log web server menggunakan Hidden Markov

Models. Hidden Markov Model (HMM) adalah data alat statistik yang kuat dengan pemodelan urutan. Sistem yang dimodelkan adalah dianggap memiliki proses Markov yang tidak teramati (tersembunyi) dan menghasilkan serangkaian keadaan yang dapat diamati. Matematika HMM dikembangkan oleh Baum et al (Leonard et al., 1970). HMM banyak digunakan dalam pengenalan pola Pengenalan ucapan dan tulisan tangan (Moerland, 1996; Huang & Jack, 1989; Huang & Jack, 1989). Ariu dkk. berhasil mengintegrasikan HMM untuk intrusion detection Analisis urutan byte payload HTTP (Ariu et al., 2011).

Ciri khas HMM adalah serangkaian observasi yang dapat diamati pengamat yang sesuai dengan proses Markov dasar dan satu set status tersembunyi, yaitu:

$S = \{s_1, s_2, \ldots, s_N\}$ : set N state (Hidden) (1)

$O = \{O_1, O_2, \ldots, O_M\}$ : set M pengamatan yang berbeda (2)

Model matematika HMM diwakili oleh :

$$\lambda = (A, B, \pi) \quad (3)$$

dimana

$$A = \{a_{ij}\} : N \; x \; N \quad (4)$$

matriks probabilitas transisi saat proses Markov beralih dari status $S_i$ ke $S_j$, yaitu : $a_{ij} = P(S_{n+1} = s_j | S_n = s_i)$ (5)

$$B = \{b_{jk}\}: NxM \quad (6)$$

matriks probabilitas emisi mencirikan kemungkinan observasi tertentu Pada $O_n \epsilon O_1 \ldots O_k$ jika model dalam keadaan $s_j$ yaitu :

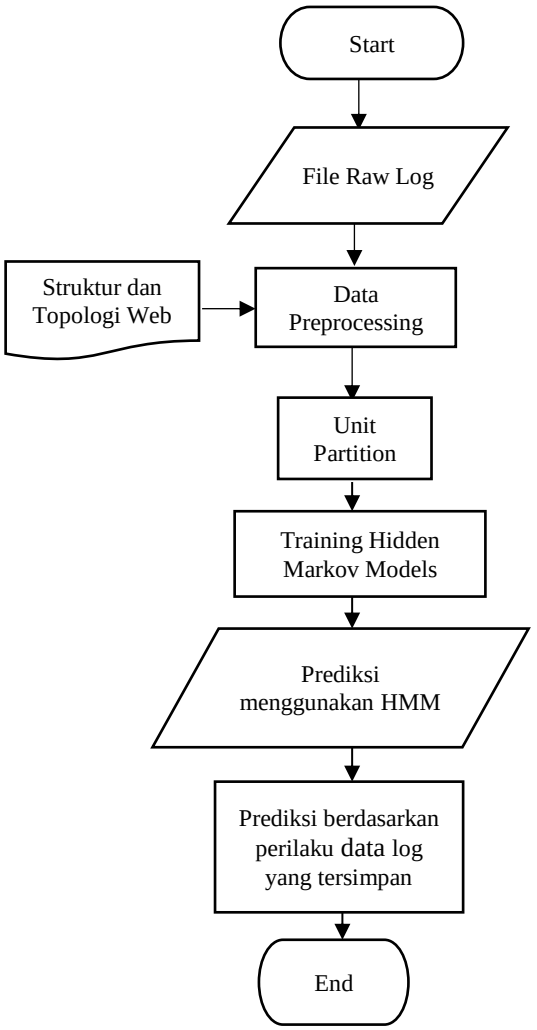
$$b_{jk} = P(O_n = O_k | S_n = s_j) \quad (7)$$

$\pi =$ Probabilitas sebelumnya dari $s_i$ menjadi state bagian pertama masuk urutan, yaitu:

$$\pi = P(S_1 = s_i) \quad (8)$$

Matriksnya *A, B,* dan $\pi$ acak baris, yaitu setiap baris apakah distribusi probabilitas. HMM memungkinkan transisi dari keadaan emitor ke situasi lain disebut traversal HMM. Ada tiga masalah utama dapat diselesaikan menggunakan HMM:

1) Evaluasi: Menurut model $\lambda = (A, B, \pi)$, urutan observasi $O = \{O_1, O_2, \ldots, O_N\}$, sebutkan $P(O|\lambda)$ urutan observasi yang mungkin sesuai model yang diberikan.
2) Decoding: model yang diberikan $A = (A, B, \pi)$, Urutan observasi $O = \{O_1, O_2, \ldots, O_N\}$ mencari yang terbaik urutan situasi yang sesuai dengan Markov dasar, proses menemukan bagian tersembunyi dari model.
3) Training: Menurut dimensi *N* dan *M*, dan amati $O = \{O_1, O_2, \ldots, O_N\}$, temukan modelnya $\lambda = (A, B, \pi)$ Maksimalkan probabilitas dari *O*. Ini dianggap melatih model agar paling sesuai dengan data yang diamati. gunakan Forward, Viterbi dan Algoritma Baum-Welch. Periksa Serangan SQL injeksi, pertama kita latih HMM (pertanyaan 3) untuk digunakan sejumlah besar kueri asli dan kueri yang dimasukkan telah distandarisasi sebagai pesanan token. Saat runtime, normalkan kueri yang masuk dan periksa urutan token yang dihasilkan dengan evaluasi untuk parameter HMM terlatih (pertanyaan 1).




**Gambar 2. Flow Chart HMM**

## C. HASIL DAN PEMBAHASAN

Setup eksperimental terdiri dari server CentOS 5.3 Gunakan Apache 2.2.3, PHP 5.3.28 dan MySQL 5.5.29, dan Beberapa PC klien terhubung di jaringan area lokal. Lima situs web PHP / MySQL Aplikasi; yaitu, toko buku, portal berita, forum, portal pekerjaan, Dan rahasia dikembangkan dan dipasang di server dengan fungsi berikut Simulasikan

database back-end dari solusi hosting bersama. Kode tersebut sengaja ditulis, tidak ada validasi input Oleh karena itu, semua halaman rentan terhadap serangan SQLIA.

Dengan mengaktifkan opsi MySQL "general_log", kami jelajahi setiap bagian dari aplikasi web dan pulihkan Kueri asli direkam oleh MySQL. Lalu kami mulai serangan injeksi SQL menggunakan berbagai alat otomatis, seperti WebCruiser Pro, NetSparker, Wapiti, Skipfish, BSQL Hacker, SQLi Dumper, SQL Invader, darkMySQL, IronWasp, sqlmap, sqlsus, The Mole, jSQL Injector, SQL Sentinel dan alat lain seperti ini Capture card, bbqsql, nikto, w3af, vega, dll., Digabungkan dengan Kali Linux (Platform Pengujian Penetrasi Lanjutan). Semua alat telah diterapkan beberapa kali dalam situasi yang berbeda konfigurasi dan ekstrak kueri yang dimasukkan darinya buat kueri file log. Setelah menormalkan dan menghapus duplikat, Kami melakukan 4.593 Injektion dan 592 urutan asli. 3654 Pesanan asli dibuat secara sintetis sintaks dengan klausa SQL WHERE.

Kumpulan dataset akhir terdiri dari 4.593 injeksi dan 4246 native pesanan token. Setiap kelompok dibagi menjadi tujuh kelompok. Di setiap cluster, kami menggunakan 90% urutan untuk pelatihan dan 10% digunakan untuk pengujian. Pelatihan dan pengujian dilakukan terhadap 50 orang periode dan hasil dirata-ratakan. Hasilnya ditampilkan sebagai matriks konfusi pada Tabel I. Nilai dalam FN dan FP termasuk sampel ditandai sebagai mencurigakan. Menurut nilai-nilai ini, hitung dan tampilkan kinerja prediktif system pada Tabel II. Hasilnya menunjukkan bahwa sistem memberikan hasil yang sangat baik akurasi tinggi, tingkat alarm palsu rendah.

**Tabel 1. Confusion Matric**

| | | Prediksi | | Total |
|---|---|---|---|---|
| | | Asli | Injeksi | |
| Actual | Asli | 423 (TN) | 2 (FP | 425 (N) |
| | Injeksi | 7 (FN) | 452 (TP | 459 (P) |
| Total | | | | 884 |

**Tabel 2. Kinerja Prediksi**

| | **Rumus** | **Hasil** |
|---|---|---|
| Presisi | TP / (TP + FP) | 99,56% |
| Penarikan | TP / (TP + FN) | 98,47% |
| Fallout (FPR) | FP / (FP + TN) | 0,47% |
| Khusus | TN / (FP + TN) | 99,53% |
| Ketepatan | (TP + TN) / (P + N) | 99,53% |
| Skor F1 | 2 TP / (2TP + FP + FN) | 99,01% |

Sistem akan meningkatkan *overhead* pemrosesan saat sedang berjalan pada tiga bagian: (a) Ekstrak bagian terakhir kuery, (b) menormalkan dan menghasilkan urutan indeks, dan (c) evaluasi dan pengambilan keputusan HMM. Dimana (a) bisa diabaikan oleh karena itu, jumlah waktu diabaikan. Waktu pemrosesan rata-rata pengukuran untuk (b) dan (c) adalah dengan menerapkan langkah-langkah ke 1000 pertanyaan yang dipilih secara acak ditunjukkan pada Tabel 3

**Tabel 3. Waktu Pemrosesan perQuery**

| **Komponen** | **Asli** | **Ijeksi** |
|---|---|---|
| Ekstraksi Bagian Akhir | ≃ 0 ms | ≃ 0 $ms$ |
| Normalisasi dan urutan indeks | 1.1731 | 3.1385 |
| Evaluasi dan keputusan | 4.7226 | 10.6950 |
| Total | **5.8958** | **13.8335** |

Perlu waktu lebih lama untuk memasukkan kueri karena alasan berikut bagian terakhir (misalnya, serangan buta dan berbasis UNION) adalah biasanya lebih panjang dari kueri aslinya. rata-rata waktu pemrosesan setiap kueri di bawah 10 md. Jika standar dinamis halaman web mengajukan total 10 pertanyaan melalui implementasi waktu tunda

untuk setiap pemuatan halaman adalah 100 md. Saat halaman dimuat waktu di Internet sekitar beberapa detik tidak akan memengaruhi pengalaman pengguna atau *end user*.

## D. PENUTUP

### Simpulan dan Saran

Pada Paper ini mengusulkan metode baru untuk mendeteksi serangan injeksi SQL real-time menggunakan normalisasi kueri yang ditingkatkan gabungkan dengan dua HMM. Cara ini bertujuan untuk berhasil pada lapisan basis data firewall, ini dapat melindungi beberapa aplikasi web yang dihosting di server bersama. Komputasi computer dengan membatasi probe kebagian klausa WHERE dari runtime kueri. Kumpulan data sudah siap diantara lima aplikasi web rentan yang banyak digunakan alat otomatis untuk menangkap berbagai jenis suntikan vektor. Hasil percobaan mengkonfirmasi metode ini deteksi secara efektif semua jenis SQLIA dengan rasio positif palsu yang rendah menilai. Sistem memiliki dampak minimal pada kinerja tidak terlihat di Internet. Ke depan, kami akan bekerja keras kurangi tingkat alarm palsu dan tingkatkan kinerja dengan menjalankan evaluasi HMM pada utas paralel.

## DAFTAR PUSTAKA

-, D. C., -, E. L., & -, M. G. (2010). Hbase - non SQL Database, Performances Evaluation. *International Journal of Advancements in Computing Technology*, 2(5): 42–52. https://doi.org/10.4156/ijact.vol2.issue5.4

Ariu, D., Tronci, R., & Giacinto, G. (2011). HMMPayl: An intrusion detection system based on Hidden Markov Models. *Computers and Security*, *30*(4), 221–241. https://doi.org/10.1016/j.cose.2010.12.004

Charania, S., & Vyas, V. (2016). SQL Injection Attack :Detection and Prevention. *International Research Journal of Engineering and Technology*, 2395–56.

Hu, J., Brown, M. K., & Turin, W. (1996). HMM based on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10): 1039–1045. https://doi.org/10.1109/34.541414

Huang, X. D., & Jack, M. A. (1989). Semi-continuous hidden markov models for speech recognition. *Computer Speech and Language*, 3(3): 239. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.333.8839&rep=rep1&type=pdf

Kar, D., Panigrahi, S., & Sundararajan, S. (2016). SQLiDDS: SQL injection detection using document similarity measure. *Journal of Computer Security*, 24(4): 507–539. https://doi.org/10.3233/JCS-160554

Kizza, J., Migga Kizza, F., Kizza, J., & Migga Kizza, F. (2011). Intrusion Detection and Prevention Systems. *Securing the Information Infrastructure*, 239–258. https://doi.org/10.4018/978-1-59904-379-1.ch012

Leonard, E. B., Ted, P., George, S., & Norman, W. (1970). A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, *41*(1), 164–171.

Loughran, D. T., Salih, M. K., & Subburaj, V. H. (2018). All About SQL Injection Attacks. *... for Information System Security ...*, *1*, 1–24. https://cisse.info/journal/index.php/cisse/article/view/87

Moerland, P. D. (1996). MicroNeuro'96. Lausanne, Switzerland. *Neurocomputing*, 12 (4): 371–373. https://doi.org/10.1016/0925-2312(96)00016-1

*owasp.org*. (n.d.-a). Retrieved January 10, 2021, from

https://owasp.org/www-project-top-ten/2017/A1_2017-Injection

*owasp.org*. (n.d.-b). Retrieved January 15, 2021, from https://owasp.org/www-project-top-ten/2017/A1_2017-Injection

Tajpour, A., Ibrahim, S., & Sharifi, M. (2012). Web Application Security by SQL Injection DetectionTools. *International Journal of Computer Science Issues*, 9(2): 332–339.