Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 4, August 2024, pp. 745-752

DOI: https://doi.org/10.52436/1.jutif.2024.5.4.2157
p-ISSN: 2723-3863
e-ISSN: 2723-3871

NETWORK'S ACCESS LOG CLASSIFICATION FOR DETECTING SQL INJECTION ATTACKS WITH THE LSTM ALGORITHM

Fajar Dzulnufrie Hafriadi[*1], Rizka Ardiansyah[2]

[1,2]Informatics Engineering, Engineering, Universitas Tadulako, Indonesia
Email: [1]backupdezet@gmail.com, [2]ardiansyah.rizka@gmail.com

(Article received: May 30, 2024; Revision: June 12, 2024; published: September 03, 2024)

Abstract

SQL Injection attacks are one of the popular web attacks. This attack is a network security problem focused on the application layer which is one of the causes of a large number of user data leaks. Currently available SQL detection techniques mostly rely on manually created features. Generally, the detection results of SQL Injection attacks depend on the accuracy of feature extraction, so they cannot overcome increasingly complex SQL Injection attacks on various systems. Responding to these problems, this research proposes a SQL Injection attack detection method using the long short term memory (LSTM) algorithm. The LSTM algorithm can learn data characteristics effectively and has strong advantages in sorting data so that it can handle massive, high-dimensional data. The research results show that the accuracy of the model approach created is able to recognize objects with a high accuracy value of 98% in identifying SQL Injection attacks.

Keywords: *attack detection, LSTM, machine learning, SQL injection, web attacks*.

KLASIFIKASI *NETWORK'S ACCESS LOG* UNTUK DETEKSI SERANGAN *SQL INJECTION* DENGAN *ALGORITMA LSTM*

Abstrak

Serangan injeksi *Structured Query Language* (*SQL)* merupakan salah satu serangan web yang populer. Serangan tersebut menjadi masalah keamanan jaringan tertuju pada layer aplikasi yang menjadikan salah satu penyebab sejumlah besar kebocoran data pengguna. Teknik deteksi SQL yang tersedia saat ini sebagian besar mengandalkan fitur yang dibuat secara manual. Umumnya hasil deteksi serangan injeksi SQL bergantung pada akurasi dari ekstraksi fitur, sehingga tidak dapat mengatasi serangan SQL Injection yang semakin kompleks pada berbagai sistem. Menanggapi permasalahan tersebut, penelitian ini mengusulkan metode deteksi serangan injeksi SQL menggunakan algoritma *long short term memory* (LSTM). Algoritma LSTM bisa mempelajari karakteristik data secara efektif dan memiliki keunggulan kuat dalam pengurutan data sehingga mampu mengatasi data masif berdimensi tinggi. Hasil penelitian menunjukkan bahwa keakuratan pendekatan model yang dibuat mampu mengenali objek dengan nilai akurasi yang tinggi sebesar 98% dalam identifikasi serangan injeksi SQL.

Kata kunci: *deteksi serangan, LTSM, machine learning, serangan web, SQL injection*.

1. PENDAHULUAN

Abad ke-20 adalah abad di mana internet ditemukan dan terus dikembangkan. Seiring perkembangan internet yang sangat pesat hingga memasuki abad ke-21 sekarang ini, penggunaan internet yang semula digunakan untuk memudahkan bertukar informasi memiliki celah untuk melangsungkan aksi kejahatan digital. Kejahatan yang dilakukan salah satunya dengan memanfaatkan protokol jaringan internet sebagai sarana pertukaran paket data. Melalui protokol jaringan internet, pelaku kejahatan digital (*hacker*) melakukan upaya manipulasi pertukaran paket data. *Hacker* menyerang menggunakan metode tertentu antara lain dengan membajak perangkat, merusak data, mengkopi data hingga mencuri data tanpa diketahui oleh korban [1]-[5].

Pencurian data masih sangat sering terjadi dikarenakan sistem seperti apa pun tidak akan terlepas dari data. Sederhananya sistem adalah kumpulan banyak data yang saling terhubung secara konstruktif dan sistematis. Data dalam suatu sistem akan tersimpan dalam penyimpanan basis data. Keamanan basis data menjadi hal yang sangat penting. Basis data sangat sering dijadikan target serangan *hacker* untuk mendapatkan data yang dibutuhkan untuk mendapat hak akses terhadap sistem. Metode yang sering digunakan adalah Injeksi

SQL. Injeksi SQL adalah salah satu masalah keamanan jaringan yang paling umum. Ini dianggap sebagai salah satu dari sepuluh ancaman keamanan aplikasi web teratas oleh *Open Web Applications Security Project* (OWASP) dan juga dianggap sebagai salah satu dari sepuluh kerentanan teratas oleh OWASP selama 15 tahun terakhir [6]-[10].

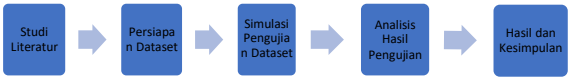
Pendeteksian Injeksi SQL masih banyak dilakukan secara manual dengan mempelajari *log* paket jaringan pada *router* atau *access point*. Setelah itu dilakukan pencocokan data dengan *log* pada *server* sistem dan basis data. Hal ini cukup memakan waktu dan kurang efektif mengingat *log router* dan *log server* selalu bertambah karena terus diakses oleh pengguna. Sejauh ini, beberapa solusi yang tersedia hanya dapat mendeteksi beberapa serangan Injeksi SQL dan tidak dapat menangani masalah yang muncul dari berbagai metode serangan. Sangat penting untuk membuat dan mempelajari skema deteksi berbasis *deep learning* yang dapat secara otomatis mengekstrak fitur [11]-[15].

Beberapa tahun terakhir, berbagai teknik deteksi telah diusulkan para peneliti. Metode-metode ini terbagi menjadi tiga kategori: tradisional, berbasis machine-learning, dan berbasis *deep neural network*. Pendekatan berbasis tradisional memiliki skala yang baik dalam pencocokan *string*, tetapi saat *hacker* menggunakan alat yang semakin canggih untuk serangan injeksi otomatis, pendekatan berbasis *machine-learning* menyelesaikan masalah pendekatan tradisional tetapi mengalami *overfitting* dan membutuhkan pemfilteran *manual* untuk mengekstrasi fitur [16]-[22].

Metode deteksi secara otomatis Injeksi SQL yang efektif adalah salah satu fokus penelitian saat ini. Tujuan penelitian ini adalah untuk membangun klasifikasi model untuk mendeteksi Injeksi SQL menggunakan LSTM. Model LSTM dilatih menggunakan data aktual melalui analisis sejumlah besar data Injeksi SQL. Dengan menggunakan *K-Fold Cross Validation* untuk menganalisis dan memahami seberapa efektif algoritma LSTM dalam deteksi data Injeksi SQL yang dapat membantu menentukan apakah model memiliki Tingkat akurasi yang baik, serta membantu mencegah *overfitting* atau *underfitting* [23]-[28].

2. METODE PENELITIAN

Metodologi penelitian ini menjelaskan bagaimana alur penelitian dilakukan sehingga dapat diketahui rincian tentang tahapan-tahapan yang dibuat secara sistematis, logis sehingga dapat dijadikan pedoman yang jelas dan mudah dalam menyelesaikan permasalahan dan kesulitan yang dihadapi dalam penelitian. Tahapan penelitian secara berurut ditunjukkan pada Gambar 1.



Gambar 1. Bagan Alur Metode Penelitian.

2.1. Studi Literatur

Studi literatur merupakan tahap awal yang dilakukan untuk menemukan permasalahan dalam penelitian disertai solusi untuk menangani permasalahan tersebut. Tinjuan yang digunakan merujuk pada karya ilmiah dan praktis tentang penelitian pada lingkup *cyber security* dan machine learning.

2.2. Persiapan Dataset

Persiapan *dataset* merupakan tahapan yang dilakukan untuk menentukan sampai mengumpulkan Kumpulan *dataset* untuk kebutuhan penelitian. *Dataset* yang digunakan merupakan data sekunder yang didapat dari Kaggle. Hal ini karena pertimbangan pada jenis data yang kritikal apabila data yang digunakan merupakan data primer karena dapat memungkinkan adanya pihak yang dirugikan. Tabel 1 merupakan uraian isi dari dataset yang digunakan, antara lain terdapat 148326 baris *sql injection query* dan pada setiap baris tersebut memiliki dua jenis label yang disimbolkan menggunakan angka 0 dan 1.

Tabel 1. Uraian Isi Database

Keterangan	Jumlah
SQL Injection Query	148438
Label	2

2.3. Simulasi Pengujian Dataset

Simulasi pengujian dataset dengan melakukan simulasi serangan injeksi SQL menggunakan *dataset* yang telah disiapkan sebelumnya. Simulasi tersebut berupa pengujian beberapa *query* untuk melakukan injeksi pada *database* berdasarkan kumpulan dataset dan target serangan. Simulasi pengujian dilakukan dengan melakukan serangan injeksi SQL menggunakan automatic mode dengan sqlmap dan manual mode menggunakan injeksi langsung ke *database*. Simulasi serangan yang dilakukan menggunakan dua metode tersebut direkam melalui *Snort Log* untuk memastikan bahwa simulasi serangan yang dilakukan berjalan dengan baik seperti pada Gambar 2 dan Gambar 3.

Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 4, August 2024, pp. 745-752
DOI: https://doi.org/10.52436/1.jutif.2024.5.4.2157
p-ISSN: 2723-3863
e-ISSN: 2723-3871

datetime	ip.src	port.src	ip.dst	port.dst	proto	method	url	query	class	score
02/03-11:22:40.466551	180.247.6.238	15377	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit%27%29%3BSELECT%20DBMS_PIPE.RECEIVE_MES	1	Non-Malicious	0.5168312788009644
02/03-11:22:40.466551	180.247.6.238	15377	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit%27%29%3BSELECT%20DBMS_PIPE.RECEIVE_MES	1	Non-Malicious	0.5168312788009644
02/03-11:22:33.648421	180.247.6.238	2271	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=1%20WAITFOR%20DELAY%20%270%3A 0%3A5%27--%20PQax&Subm	1 WAITFOR DELAY '0:	Malicious	0.9894206523895264
02/03-11:22:33.648421	180.247.6.238	2271	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=1%20WAITFOR%20DELAY%20%270%3A 0%3A5%27--%20PQax&Subm	1 WAITFOR DELAY '0:	Malicious	0.9894206523895264
02/03-11:22:32.231572	180.247.6.238	21159	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=1%29%3BWAITFOR%20DELAY%20%270 %3A0%3A5%27--&Submit=Su	1);WAITFOR DELAY '0	Malicious	0.9894206523895264
02/03-11:22:32.231572	180.247.6.238	21159	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=1%29%3BWAITFOR%20DELAY%20%270 %3A0%3A5%27--&Submit=Su	1);WAITFOR DELAY '0	Malicious	0.9894206523895264
02/03-11:22:30.608366	180.247.6.238	15013	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=1%27%29%20AND%20EXTRACTVALUE% 286425%2CONCAT%280x5c	1') AND EXTRACTVALUE%	Malicious	0.9981378316879272
02/03-11:22:30.608366	180.247.6.238	15013	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=1%27%29%20AND%20EXTRACTVALUE% 286425%2CONCAT%280x5c	1') AND EXTRACTVALUE%	Malicious	0.9981378316879272

Gambar 2. *Live Testing* pada *Snort Log* menggunakan Sqlmap

datetime	ip.src	port.src	ip.dst	port.dst	proto	method	url	query	class	score
01/31-17:59:09.031173	180.247.44.30	25619	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=%27+OR+%271%27%3D%271&Submit= Submit	'+OR+'1'='1	Malicious	0.9953846335411072
01/31-17:59:06.915296	180.247.44.30	25619	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=2&Submit=Submit	2	Non-Malicious	0.9349966959953308
01/31-17:54:05.088646	180.247.44.30	17218	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=%27+OR+%271%27%3D%271&Submit= Submit	'+OR+'1'='1	Malicious	0.9953846335411072
01/31-17:58:06.190618	180.247.44.30	31459	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit	1	Non-Malicious	0.5168312788009644
01/31-17:53:47.388033	180.247.44.30	19889	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=1%27&Submit=Submit	1'	Malicious	0.8621855974197388
01/31-17:59:09.485187	180.247.44.30	14280	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=%27+OR+%271%27%3D%271&Submit= Submit	'+OR+'1'='1	Malicious	0.9953846335411072
01/31-22:18:37.544456	180.247.5.97	2715	178.128.210.109	80	TCP	GET	/DVWA/vulnerabilities/sqli/?id=%22+OR+1+%3D+1+--+-&Submit=Su bmit	"+OR+1+=+1+--+-	Malicious	0.9718218445777893

Gambar 3. *Live Testing* pada *Snort Log* menggunakan injeksi manual

2.4. Analisis Hasil Pengujian

Analisis hasil pengujian dilakukan setelah data baru dihasilkan dari proses simulasi pengujian *dataset*. Analisis ini dilakukan untuk mendapatkan nilai akurasi dari pengukuran model deteksi serangan sql injection menggunakan algoritma *Long Short Term Memory*. Penggunaan algoritma *machine learning* pada umumnya memiliki beberapa tahapan sebelum implementasi algoritma berlangsung. Tahapan ini dinamakan *preprocessing* yang terdiri dari beberapa sub proses sebelum berlanjut pada proses *machine learning*. Hal ini diperlukan agar dapat menghasilkan model *machine learning* yang optimal, meminimalisir *overfitting*, dan hal lainnya yang dapat mempengaruhi akurasi model *machine learning* tersebut.

Tahapan *preprocessing* yang dilakukan pada penelitian ini diantaranya:

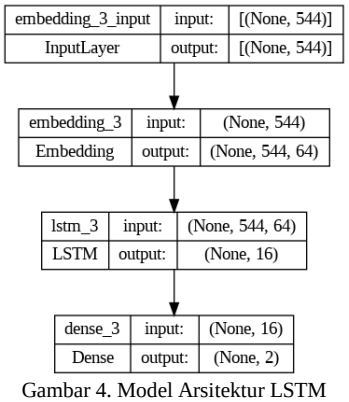
1) *Labeling*: Proses *labeling* dilakukan karena dataset yang digunakan hanya terdiri dari *query* SQL, sehingga kemungkinan *noise* yang terjadi pada data sangat kecil maka proses data *cleaning* tidak dilakukan. Terkikisnya data sehingga menjadi lingkup yang lebih kecil menjadi kekhawatiran apabila dilakukan proses data *cleaning* secara berlanjut.
2) *Transformation*: Proses transformasi data dilakukan untuk mengeliminasi variasi data dan kecenderungan data yang tidak relevan dengan hasil model yang diharapkan. Data yang telah ditransformasi akan mempermudah proses perhitungan algoritma dan dapat memberikan hasil yang optimal.

Kemudian masuk ke tahapan *modeling* penggunaan model LSTM dimulai dengan mengubah karakter ke dalam matriks numerik secara berurutan. Banyak metode yang dapat digunakan untuk melakukan konversi bentuk karakter menjadi numerik. *Tokenizing* merupakan metode yang dipilih pada penelitian ini untuk mengkonversi nilai karakter ke dalam bentuk numerik. Hal ini karena *tokenizing* dapat mengkonversi teks ke dalam numerik tanpa mengurangi konteksnya dan mempermudah model untuk mengidentifikasi pola dari kumpulan data.

Selanjutnya dalam melatih model LSTM pada penelitian ini menggunakan *Optimization of Loss Function* yaitu *Cross-Entropy (Cross-Entropy Loss)* sebagai fungsi untuk menggambarkan kesalahan antara model yang diprediksi dan nilai dan nilai sebenarnya yang diketahui, dicatat sebagai berikut:

$$L(y,p) = -\frac{1}{N}\sum_{i=1}^{N} \quad [y_i \cdot \log(p_i) + (1-y_i) \cdot \log(1-p_i)] \quad (1)$$

Dimana N adalah jumlah sampel latih dan $y = [y_1, y_2 \cdots y_n,]T; yi$ mewakili nilai keluaran yang diharapkan dari sampel ke-I, yaitu label sebenarnya dari sampel tersebut. p_i menunjukkan probabilitas bahwa sampel ke-I diprediksi sebagai kasus positif.



Gambar 4. Model Arsitektur LSTM

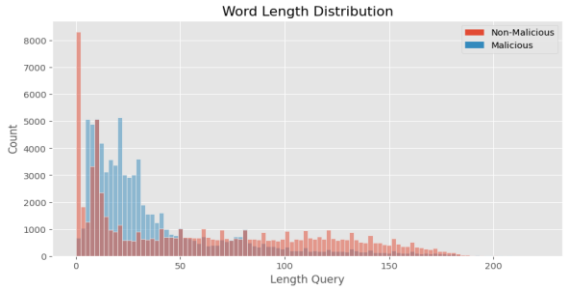
3. HASIL DAN PEMBAHASAN

Dataset yang didapat dari Kaggle berupa kumpulan *query* SQL sebanyak 148438 dengan distribusi seperti pada Tabel 2.

Tabel 2. Distribusi Dataset

Label	Deskripsi	Jumlah	Rasio
1	SQL Injection Statement	77,760	52.42%
0	Non-SQL Injection Statement	70,678	47.58%

Trigger utama yang menjadi indikasi suatu serangan injeksi SQL adanya perintah SELECT pada url suatu *website*, hal ini mengindikasikan adanya perintah untuk memilih suatu data pada *database*. Berdasarkan proses analisis yang telah dilakukan dan membagi respon *query* berdasarkan *malicious* dan *non-malicious query* didapati keragaman pada dataset tersebut, seperti yang ditampilkan pada Gambar 5.



Gambar 5. Perbandingan kata pada *query* SQL

Pemicu indikasi serangan injeksi SQL tidak hanya pada kata SELECT, namun ada beberapa kata lainnya termasuk karakter yang menjadi *trigger* sebagai serangan injeksi SQL. Tabel 3 dan Tabel 4 merupakan kata dan karakter yang umum digunakan untuk melakukan serangan injeksi SQL beserta pengoprasiannya.

Tabel 3. Kata Umum Perintah SQL

Kategori	Kata
Perintah Dasar	select, order by, group by, union, count
Perintah Modifikasi	Insert, drop table, update, update
Simbol Koneksi	or, and, into, from, where
Operasi pada Sistem	net, master, xp_cmdshell, exec
Operasi pada File	dumpfile, load_file, outfile

Tabel 4. Karakter Umum Perintah SQL

Instruksi	Karakter
Operasi Dasar	*, /, +, -
Kondisi dan Keputusan bersyarat	<>, <=, >=, !=, =
Komentar	#, */, --, /*
Lainnya	(), @, “”, ’’, \

Tabel 5 dan Tabel 6 merupakan beberapa sampel pada data yang telah dikonversi menjadi numerik.

Tabel 5. Distribusi Dataset

Label	Deskripsi	Jumlah	Rasio
1	SQL Injection Statement	77,760	52.42%
0	Non-SQL Injection Statement	70,678	47.58%

Tabel 6. Tokenisasi

Index	Query	Label	Tokenizer
0	" or pg_sleep (__time__) --	1	15,1,131,100
1	create user name identified by pass123 temporary tablespace temp default tablespace users;	1	1,1,1,1,40,1,1,1, 1,1,1,97
2	and 1 = utl_inaddr.get_host_ address ((distinct (table_name) from (distinct (table_name) , rownum as limit from sys.all_tables) where limit = 5)) and 'i' = 'i	1	3,14,1,1,125,1,1 ,1,1,1,16,1,1,1,1 ,13,1,16,1,21,1, 23,1,33,3,1,1
3	* from users where id = '1' or @ @1 = 1 union 1,version ( ) -- 1'	1	16,97,23,1,1,15, 14,14,81,14,1,1 07
4	* from users where id = 1 or 1#" (union 1,version ( ) -- 1	1	16,97,23,1,14,1 5,14,81,14,1,14
5	name from syscolumns where id = (id from sysobjects where name = tablename') --	1	1,16,1,23,1,1,16 ,1,23,1,1
6	* from users where id = 1 +$+ or 1 = 1 -- 1	1	16,97,23,1,14,1 5,14,14,14
7	1; (load_file (char (47,101,116,99,47,11 2,97,115,115,119,10 0))) ,1,1,1;	1	14,1,1,22,1,1,1, 1,1,1,1,1,1,1,1,1 4,14,14
8	* from users where id = '1' or ‖/1 = 1 union 1,version ( ) -- 1'	1	16,97,23,1,1,15, 14,14,81,14,1,1 07
9	* from users where id = '1' or \.<\ union 1,@@version -- 1'	1	16,97,23,1,1,15, 81,14,1,107

Hasil dari model dataset terbagi pada empat karakteristik, diantaranya: TP, TN, FP, FN. Karakteristik tersebut digunakan sebagai komponen evaluasi pada model dan akan didapatkan hasil pengukuran yang terdiri dari beberapa aspek diantaranya:

Akurasi, merupakan sebagian jumlah sampel yang benar dari keseluruhan total sampel, hal ini dapat diukur menggunakan rumus:

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

Presisi, merupakan banyaknya sampel yang diprediksi positif dan memang memiliki nilai positif, untuk mengukurnya dapat menggunakan rumus berikut:

$$Presisi = \frac{TP}{TP+FP} \quad (3)$$

Recall, merupakan proporsi sampel positif dalam sampel yang diprediksi secara akurat, hal ini dapat diukur menggunakan rumus:

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

F1 Score, merupakan gabungan dari skor presisi dan perolehan model untuk mengukur akurasi model sebagai bentuk evaluasi, cara untuk mendapatkan score tersebut menggunakan rumus berikut:

$$F1 = \frac{2\ x\ Precission\ x\ Recall}{Precission+Recall} \quad (5)$$

Hasil eksperimen didasarkan pada metode *k-fold cross validation*. Metode ini melibatkan pemisahan kumpulan data menjadi beberapa sub kumpulan yang tidak tumpang tindih, menggunakan nilai k di antaranya sebagai data pelatihan dan satu sisanya sebagai data pengujian. Kesalahan tes diperkirakan dengan menghitung rata-rata kesalahan tes di k percobaan. Penelitian ini menggunakan 3x validasi menggunakan *k-fold cross validation*.

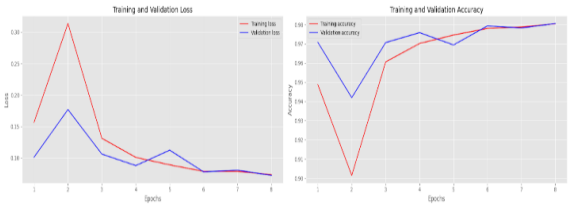
Tabel 7. Distribusi Dataset

Label	Deskripsi	Jumlah	Rasio
1	SQL Injection Statement	77,760	52.42%
0	Non-SQL Injection Statement	70,678	47.58%

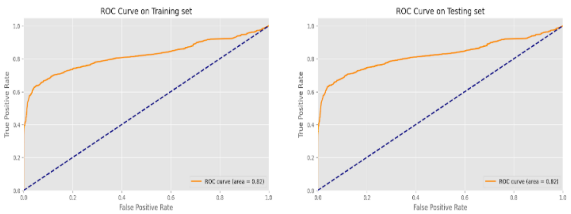
Tabel 8. Distribusi Dataset

Label	Deskripsi	Jumlah	Rasio
1	SQL Injection Statement	77,760	52.42%
0	Non-SQL Injection Statement	70,678	47.58%

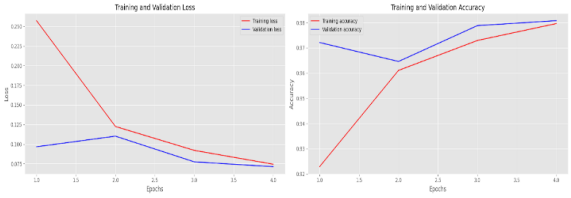
Tabel 7 merupakan *confusion matrix* yang didapat dari *k-fold cross validation*. Tabel tersebut menunjukkan rasio pengujian yang relatif stabil dengan selisih nilai yang masih normal. Berdasarkan hasil dari *confusion matrix* yang didapat, Tabel 8 menunjukkan report akurasi model LSTM yang telah dibuat. Hasil pengujian model menggunakan k-fold *cross validation* menunjukkan bahwa model LSTM yang dibuat memiliki nilai stabilitas dan akurasi yang tinggi. Hal ini dibuktikan dengan 3x pengujian validasi dengan score yang didapat menunjukkan nilai yang stabil yaitu pada angka 0.98 atau 98%.



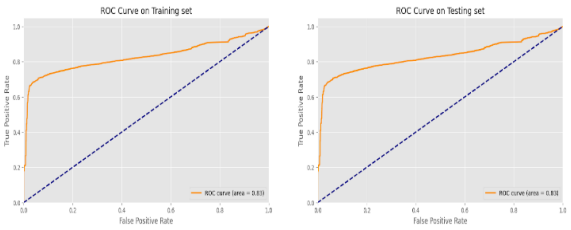
Gambar 6. K-Fold 0



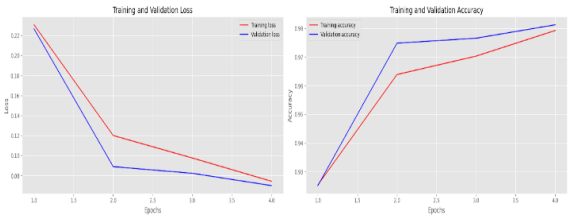
Gambar 7. ROC K-Fold 0



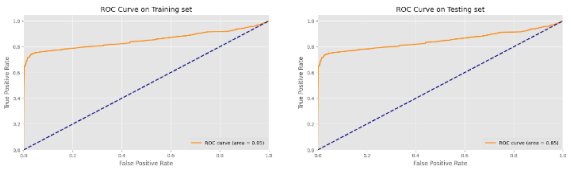
Gambar 8. K-Fold 1



Gambar 9. ROC K-Fold 1



Gambar 10. ROC K-Fold 1



Gambar 11. ROC K-Fold 1.

4. DISKUSI

Penelitian ini mengusulkan kerangka kerja untuk mendeteksi serangan injeksi SQL menggunakan algoritma *Long Short-Term Memory* (LSTM) dalam machine learning. LSTM dipilih karena kemampuannya yang unggul dalam menangkap urutan data dan ketergantungan jangka panjang, yang sangat relevan dalam mendeteksi pola kompleks seperti serangan injeksi SQL.

Pada kerangka kerja yang diusulkan, *query* dalam URL diubah menjadi format numerik. Transformasi ini penting untuk memfasilitasi proses pelatihan model LSTM, yang memerlukan input numerik untuk mengidentifikasi pola dan hubungan dalam data. Setelah transformasi, data numerik ini digunakan sebagai input untuk melatih model deteksi.

Keunggulan utama LSTM terletak pada kemampuannya menangkap urutan data dan ketergantungan temporal, memungkinkan model untuk memahami hubungan antara karakter dalam *string* SQL. Hal ini memungkinkan LSTM untuk membedakan dengan lebih akurat antara pernyataan SQL yang sah dan yang berbahaya, karena model dapat mempelajari pola dan struktur *internal string*.

Hasil penelitian menunjukkan bahwa model deteksi berbasis LSTM memiliki akurasi yang tinggi dalam mendeteksi serangan injeksi SQL, dengan akurasi model mencapai 98%. Angka ini menunjukkan potensi signifikan dari algoritma LSTM dalam mengenali pola-pola yang terkait dengan serangan *cyber* pada URL. Jika dibandingkan dengan penelitian serupa dari Deep Neural Network-Based SQL Injection Detection Method [29] pada

penelitian serupa mempunyai empat model yaitu LSTM dengan akurasi 62%, KNN dengan akurasi 82%, DT dengan akurasi 92% dan SQLNN dengan akurasi 96% tetapi akurasi model pada penilian ini lebih unggul dengan skor 98% dibanding dengan penelitian serupa.

Selain mendeteksi injeksi SQL, algoritma LSTM menunjukkan potensi yang baik dalam mengidentifikasi berbagai jenis ancaman siber lainnya. Karena LSTM tidak memerlukan ekstraksi fitur yang berbeda untuk berbagai ancaman, algoritma ini sangat terukur dan mampu mengenali karakteristik berbagai serangan dengan data pelatihan yang memadai. Beberapa serangan siber pada website yang berinteraksi pada URL, seperti *cross-site scripting* (XSS) dan *code injection*, memiliki karakteristik serupa yang dapat dikenali dengan baik oleh model LSTM.

5. KESIMPULAN

Penelitian ini berhasil menunjukkan bahwa algoritma LSTM memiliki kemampuan yang kuat dalam mendeteksi serangan injeksi SQL melalui analisis *query* URL. Dengan akurasi model mencapai 98%, LSTM membuktikan dirinya sebagai alat yang efektif dalam mengenali pola serangan siber.

Keunggulan utama LSTM adalah kemampuannya dalam menangkap urutan data dan ketergantungan jangka panjang, yang memungkinkan model untuk memahami hubungan kompleks antara karakter dalam *string* SQL. Hal ini membuat LSTM dapat membedakan antara pernyataan SQL yang sah dan berbahaya dengan akurasi yang tinggi.

Selain itu, potensi penggunaan LSTM untuk deteksi ancaman *cyber* lainnya juga sangat menjanjikan. Dengan kemampuan mengenali pola tanpa memerlukan ekstraksi fitur yang berbeda, LSTM dapat diaplikasikan untuk mendeteksi berbagai serangan web lainnya seperti *cross-site scripting* (XSS) dan *code injection*, memberikan solusi yang lebih terukur dan adaptif dalam keamanan siber.

Secara keseluruhan, penelitian ini menegaskan bahwa algoritma LSTM adalah pilihan yang kuat dan fleksibel untuk deteksi ancaman *cyber*, terutama dalam konteks serangan yang berhubungan dengan URL. Dengan pengembangan lebih lanjut dan pelatihan menggunakan dataset yang lebih luas, potensi aplikasi LSTM dalam keamanan siber dapat terus ditingkatkan, memberikan perlindungan yang lebih baik terhadap berbagai jenis serangan.

DAFTAR PUSTAKA

[1] G. C. Amaizu, C. I. Nwakanma, S. Bhardwaj, J. M. Lee, and D. S. Kim, "Composite and efficient DDoS attack detection framework for B5G networks," *Computer Networks*, vol. 188, Apr. 2021, doi: 10.1016/j.comnet.2021.107871.

[2] S. A. Reddy and B. Rudra, "Evaluation of Recurrent Neural Networks for Detecting Injections in API Requests," in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference, CCWC 2021*, Institute of Electrical and Electronics Engineers Inc., Jan. 2021, pp. 936–941. doi: 10.1109/CCWC51732.2021.9376034.

[3] W. Yang, M. N. Johnstone, S. Wang, N. M. Karie, N. M. bin Sahri, and J. J. Kang, "Network Forensics in the Era of Artificial Intelligence," in *Studies in Computational Intelligence*, vol. 1025, Springer Science and Business Media Deutschland GmbH, 2022, pp. 171–190. doi: 10.1007/978-3-030-96630-0_8.

[4] L. F. Sikos, "Packet analysis for network forensics: A comprehensive survey," *Forensic Science International: Digital Investigation*, vol. 32. Elsevier Ltd, Mar. 01, 2020. doi: 10.1016/j.fsidi.2019.200892.

[5] F. Yasin, Abdul Fadlil, and Rusydi Umar, "Identifikasi Bukti Forensik Jaringan Virtual Router Menggunakan Metode NIST," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 5, no. 1, pp. 91–98, Feb. 2021, doi: 10.29207/resti.v5i1.2784.

[6] J. Zhou, X. Luo, Q. Shen, and Z. Xu, Eds., *Information and Communications Security*, vol. 11999. in Lecture Notes in Computer Science, vol. 11999. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-41579-2.

[7] R. K. Jothi, S. Balaji B, N. Pandey, P. Beriwal, and A. Amarajan, "An Efficient SQL Injection Detection System Using Deep Learning," in *Proceedings of 2nd IEEE International Conference on Computational Intelligence and Knowledge Economy, ICCIKE 2021*, Institute of Electrical and Electronics Engineers Inc., Mar. 2021, pp. 442–445. doi: 10.1109/ICCIKE51210.2021.9410674.

[8] B. Aruna and B. Usharani, "SQLID Framework in Order ToPerceive SQL Injection Attack on Web Application," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing Ltd, 2020. doi: 10.1088/1757-899X/981/2/022013.

[9] A. Rai, M. M. I. Miraz, D. Das, H. Kaur, and Swati, "SQL Injection: Classification and Prevention," in *Proceedings of 2021 2nd International Conference on Intelligent Engineering and Management, ICIEM 2021*, Institute of Electrical and Electronics Engineers Inc., Apr. 2021, pp. 367–372. doi: 10.1109/ICIEM51511.2021.9445347.

[10] I. S. Crespo-Martínez, A. Campazas-Vega,

Á. M. Guerrero-Higueras, V. Riego-DelCastillo, C. Álvarez-Aparicio, and C. Fernández-Llamas, "SQL injection attack detection in network flow data," *Comput Secur*, vol. 127, Apr. 2023, doi: 10.1016/j.cose.2023.103093.

[11] S. S. Nagasundari and P. B. Honnavali, "SQL Injection Attack Detection using ResNet." [Online]. Available: http://www.dockguard.co.uk/page.php?id=18'

[12] H. Zhang, J. Zhao, B. Zhao, X. Yan, H. Yuan, and F. Li, "SQL injection detection based on deep belief network," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Oct. 2019. doi: 10.1145/3331453.3361280.

[13] C. Arumugam *et al.*, "Prediction of SQL Injection Attacks in Web Applications," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2019, pp. 496–505. doi: 10.1007/978-3-030-24305-0_37.

[14] P. Roy, R. Kumar, and P. Rani, "SQL Injection Attack Detection by Machine Learning Classifier," in *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, 2022, pp. 394–400. doi: 10.1109/ICAAIC53929.2022.9792964.

[15] J. Hu, W. Zhao, and Y. Cui, "A Survey on SQL Injection Attacks, Detection and Prevention," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Feb. 2020, pp. 483–488. doi: 10.1145/3383972.3384028.

[16] P. Tang, W. Qiu, Z. Huang, H. Lian, and G. Liu, "Detection of SQL injection based on artificial neural network ☆," vol. 190, p. 105528, 2020, doi: 10.1016/j.knosys.

[17] K. Zhang, "A machine learning based approach to identify SQL injection vulnerabilities," in *Proceedings - 2019 34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019*, Institute of Electrical and Electronics Engineers Inc., Nov. 2019, pp. 1286–1288. doi: 10.1109/ASE.2019.00164.

[18] Q. Li, W. Li, J. Wang, and M. Cheng, "A SQL Injection Detection Method Based on Adaptive Deep Forest," *IEEE Access*, vol. 7, pp. 145385–145394, 2019, doi: 10.1109/ACCESS.2019.2944951.

[19] P. Tang, W. Qiu, Z. Huang, H. Lian, and G. Liu, "Detection of SQL injection based on artificial neural network ☆," vol. 190, p. 105528, 2020, doi: 10.1016/j.knosys.

[20] D. Chen, Q. Yan, C. Wu, and J. Zhao, "SQL Injection Attack Detection and Prevention Techniques Using Deep Learning," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Feb. 2021. doi: 10.1088/1742-6596/1757/1/012055.

[21] Dwi Kurnia Wibowo, Ahmad Luthfi, Yudi Prayudi, Erika Ramadhani, and Muhamad Maulana, "Faux Insider Hazard Investigation on Non-Public Cloud Computing by Using ADAM's Technique," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 6, no. 6, pp. 1028–1036, Dec. 2022, doi: 10.29207/resti.v6i6.4714.

[22] R. Patria Avrianto, J. Dio Firizqi, R. Dwi Kurniawan, R. Eko Indrajit, and E. Dazki, "SELECTION OF PAYMENT METHODS IN ONLINE MARKETS USING ANALYTICAL HIERARCHICAL PROCESS," *Jurnal Teknik Informatika (JUTIF)*, vol. 3, no. 3, pp. 697–705, 2022, doi: 10.20884/1.jutif.2022.3.3.232.

[23] X. Song, C. Chen, B. Cui, and J. Fu, "Malicious javascript detection based on bidirectional LSTM model," *Applied Sciences (Switzerland)*, vol. 10, no. 10, May 2020, doi: 10.3390/app10103440.

[24] Z. shi Gao, Y. Su, Y. Ding, Y. dong Liu, X. an Wang, and J. wei Shen, "Key Technologies of Anomaly Detection Using PCA-LSTM," in *Advances in Intelligent Systems and Computing*, Springer Verlag, 2020, pp. 246–254. doi: 10.1007/978-3-030-22263-5_24.

[25] T. Y. Kim and S. B. Cho, "Optimizing CNN-LSTM neural networks with PSO for anomalous query access control," *Neurocomputing*, vol. 456, pp. 666–677, Oct. 2021, doi: 10.1016/j.neucom.2020.07.154.

[26] S. Hao, J. Long, and Y. Yang, "BL-IDS: Detecting Web Attacks Using Bi-LSTM Model Based on Deep Learning," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, Springer Verlag, 2019, pp. 551–563. doi: 10.1007/978-3-030-21373-2_45.

[27] R. W. Kadhim and M. T. Gaata, "A hybrid of CNN and LSTM methods for securing web application against cross-site scripting attack," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no. 2, pp. 1022–1029, Feb. 2020, doi: 10.11591/ijeecs.v21.i2.pp1022-1029.

[28] N. Gupta, V. Jindal, and P. Bedi, "LIO-IDS: Handling class imbalance using LSTM and improved one-vs-one technique in intrusion

detection system," *Computer Networks*, vol. 192, Jun. 2021, doi: 10.1016/j.comnet.2021.108076.

[29] W. Zhang *et al.*, "Deep Neural Network-Based SQL Injection Detection Method," *Security and Communication Networks*, vol. 2022, 2022, doi: 10.1155/2022/4836289..