REPOSITOR, Vol. 2, No. 9, September 2020, Pp. 1273-1280
ISSN : 2714-7975
E-ISSN : 2716-1382

Evaluasi Potensi Celah Keamanan SQL Injection Menggunakan Nearest Neighbor pada Security-Software Development Life Cycle

Dia Putera Idiana Kusuma[1], Nur Hikmatul Maulida[2], Ma'rifat[3], Dedy Hariyadi[*4]
[1,2,3]Universitas Amikom Yogyakarta
[4]Universitas Jenderal Achmad Yani Yogyakarta
dia.ku@students.amikom.ac.id[1], nur.maulida@students.amikom.ac.id[2]
marifat.1999@students.amikom.ac.id[3], dedy@unjaya.ac.id*[4]

Abstrak

Berdasarkan survei oleh Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) tentang penggunaan internet di Indonesia, selalu menggunakan peningkatan dari tahun ke tahun. Meningkatnya penggunaan internet di Indonesia Berdasarkan laporan dari Tim Tanggap Insiden Keamanan Indonesia tentang Infrastruktur Internet / Pusat Koordinasi (ID-SIRTII / CC) serangan siber di Indonesia terjadi pada sektor pemerintah. Teknik yang digunakan pada serangan siber menurut catatan beberapa lembaga seperti ID-SIRTII / CC, Badan Siber dan Sandi Negara (BSSN), dan perusahaan analis keamanan informasi adalah SQL Injection (Structured Query Language Injection). Teknik SQL Injection menggunakan permintaan pada sistem basis data. Bahkan Open Security Project Project (OWASP) menempatkan SQL Injection sebagai teknik yang paling populer. Untuk menganalisis potensi celah kemanan SQL Injection memerlukan analisis yang lebih komprehensif berdasarkan kasus-kasus sebelumnya bersama praktisi bidang keamanan informasi. Pada penelitian ini diusulkan untuk menganalisis serangan SQL Injection menggunakan Algoritma Nearest Neighbor berdasarkan kasus injection yang terjadi selama ini yang telah dilakukan oleh profesional penguji keamanan informasi kemudian pada tahap pengolahannya didapatkan suatu nilai kedekatan untuk menghitung kemungkinan terjadinya serangan SQL Injection. Analisis menggunakan Algoritma Nearest Neighbor dapat menentukan potensi celah keamanan dari SQL Injection. Hasil penelitian ini adalah memberikan nilai evaluasi potensi serangan SQL Injection dengan memperhitungkan potensi celah keamanan berdasarkan hasil pengolahan menggunakan algoritma Nearest Neighbor sehingga dapat mengurangi terjadinya serangan SQL Injection saat proses produksi.

Kata Kunci: *SQL Injection, Keamanan Informasi, Analisis Keamanan, Algoritma Nearest Neighbor*

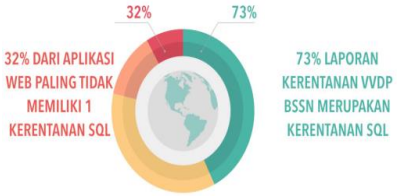
Abstract

Based on a survey by the Indonesian Internet Service Providers Association (APJII) on internet usage in Indonesia, the use has always been increasing from year to year. Increased use of the internet in Indonesia Based on reports from the Indonesian Security Incident Response Team on the Internet Infrastructure / Coordination Center (ID-SIRTII / CC) cyber attacks in Indonesia occurred in the government sector. The technique used in cyberattacks, according to records from several agencies such as ID-SIRTII / CC, Badan Siber dan Sandi Negara (BSSN), and information security analyst companies is SQL Injection (Structured Query Language Injection). The SQL Injection technique uses a query on the database system. Even the Open Security Project Project (OWASP) ranks SQL Injection as the most popular technique. To analyze potential SQL injection vulnerabilities requires a more comprehensive analysis based on previous cases with information security practitioners. In this study, it is proposed to analyze the SQL Injection attack using the Nearest Neighbor Algorithm based on the injection cases that have occurred so far that have been carried out by information security testing professionals then at the processing stage a proximity value is obtained to calculate the possibility of a SQL Injection attack. Analysis using the Nearest Neighbor Algorithm can determine potential security holes from SQL Injection. The results of this study are to provide an evaluation value of the potential for SQL Injection attacks by taking into account the potential for security holes based on the processing results using the Nearest Neighbor algorithm so that it can reduce the occurrence of SQL Injection attacks during the production process.

Keywords: *SQL Injection, Information Security, Security Analysis, Nearest Neighbor Algorithm*

1. Pendahuluan

Peningkatan pengguna internet di Indonesia meningkat dari tahun ke tahun, berdasarkan survei Asosiasi Penyelenggara Jasa Internet Indonesia pada tahun 2018 peningkatan signifikan dari tahun 2017 menuju 2018 terdapat peningkatan 27.900.000 pengguna [1]. Peningkatan pengguna internet juga mempengaruhi tingkat kejahatan di ruang siber. Kejahatan di ruang siber berdasarkan laporan kerentanan Badan Siber dan Sandi Negara (BSSN) menggunakan teknik SQL Injection. Hal ini sesuai dengan tingkat kerentanan aplikasi berbasis web yang dilaporkan melalui program *Voluntary Vulnerability Disclosure Program* (VVDP) yang diselenggarakan oleh BSSN. Pada laporan tersebut *SQL Injection* menunjukan presentase kerentanan sebesar 73% pada situs web di Indonesia, tampak seperti pada Gambar 1 [2].

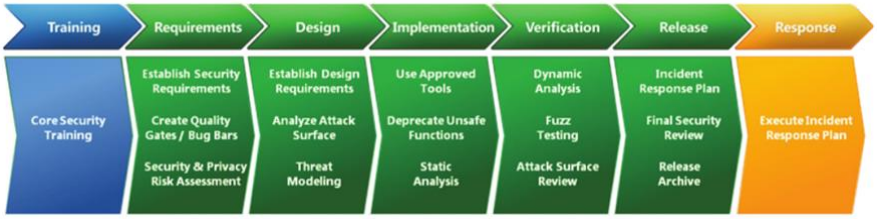


Gambar 1. Persentase Kerentanan Aplikasi Berbasis Web di Indonesia

Teknik *SQL Injection* dapat mengambil alih server, memanipulasi data maupun mengubah tampilan sistus web. Teknik selanjutnya untuk mengubah tampilan situs web disebut web defacement [3]. Teknik *SQL Injection* merupakan teknik injeksi kode yang mengeksploitasi kerentanan keamanan yang terjadi di lapisan basis data suatu aplikasi berbasis web. Penelitian sebelumnya untuk melakukan deteksi terhadap serangan siber teknik *SQL Injection* menggunakan analisis statik. Analisis statik yang digunakan untuk mendeteksi teknik *SQL Injection* dengan membenamkan ke dalam kode halaman login menggunakn mekanisme pencocokan ASCII [4]. Teknik lain untuk mendeteksi potensi serangan *SQL Injection* dengan cara melakukan validasi *Uniform Resource Locator* (URL) menggunakan Algoritman S*QL-Injection Free* [4].

Sebelum melakukan deteksi terhadap serangan siber yang menggunakan teknik *SQL Injection* perlu dilakukan definisi potensi serangan yang kemudian disebut sebagai atribut. Nilai-nilai atribut tersebut dapat diolah menjadi sebuah nilai kedekatan. Berdasarkan nilai kedekatan dari sebuah atribut maka deteksi serangan *SQL Injection* dapat diukur menggunakan algoritma Nearest Neigbor. Maka pada tulisan ini diusulkan metode evaluasi potensi celah keamanan aplikasi berbasis web yang menggunakan algoritma Nearest Neighbor. Evaluasi ini perlu diterapkan pada *Security-Software Development Life Cycle* yang terdiri dari tujuh tahapan, diantaranya: *Training, Requirements, Design, Implementation, Verification, Release,* dan *Response* untuk mengurangi potensi celah keamanan. Adapun tujuh tahapan *Security-Software Development Life Cycle* rekomendasi dari Microsoft tampak seperti Gambar 2 [5].

Berdasarkan hasil observasi di beberapa institusi masih belum menerapkan pengembangan perangkat lunak menggunakan metode *Security-Software Development Life Cycle* seperti pada Gambar 2 [6]. Pada penelitian sebelumnya pengujian keamanan informasi pada perangkat lunak masih berfokus pada pasca proses produksi, yaitu melaksanakan *penetration testing* secara rutin dan berkala [7]. Proses penetration testing secara berkala memang baik, tetapi evaluasi potensi celah keamanan pada proses perbaikan dan pembuatan aplikasi perlu dilakukan. Maka pada penelitian ini diusulkan melakukan evaluasi potensi serangan SQL Injection menggunakan algoritma Nearest Neighbor sehingga dapat mengurangi terjadinya serangan SQL Injection pada saat proses produksi.



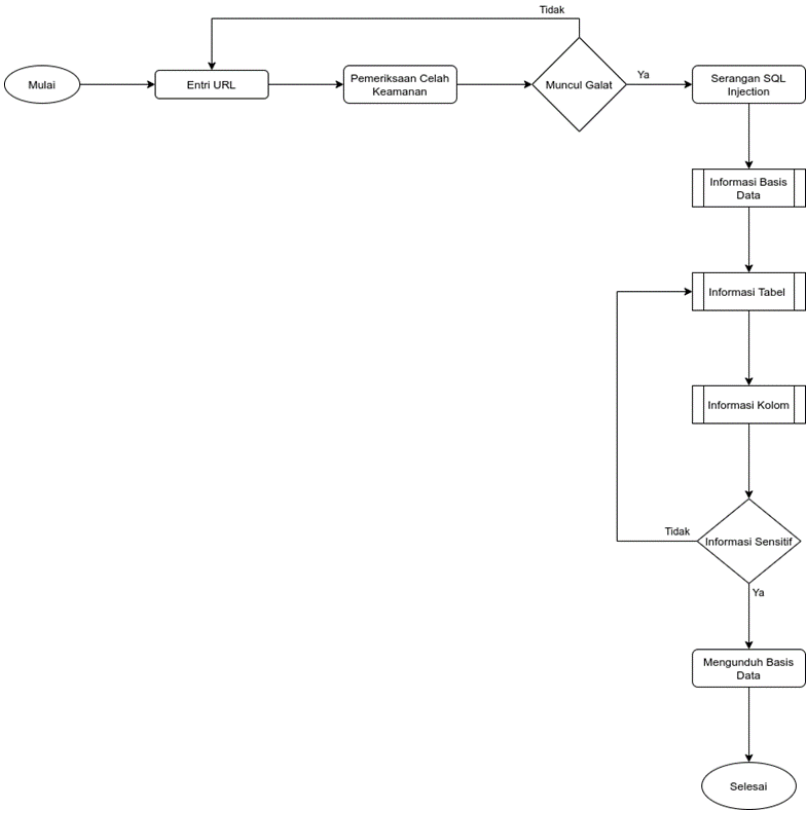
Gambar 2. Microsoft Secuirty-Software Development Life Cycle

2. Metode Penelitian

SQL Injection merupakan teknik injeksi yang memanfaatkan celah keamanan pada validasi masukan aplikasi berbasis web. OWASP menempatkan celah keamanan injeksi termasuk diantaranya *SQL Injection* dalam urutan pertama pada *The Ten Most Critical Web Application Security Risks* [8]. Dampak dari celah keamanan *SQL Injection* informasi terkait dengan sistem basis data dapat dibaca dengan mudah bahkan melakukan perubahan isi pada basis data atau mematikan layanan basis data [9]. Adapun teknik melakukan *SQL Injection* terbagi menjadi enam, diantaranya [10] [11]:

1. *Boolean-based blind* merupakan teknik injeksi yang membandinkan respon karakter untuk mendapatkan nilai boolean true atau false. Hasil bernilai true akan dilanjutkan proses selanjutnya.
2. *Time-based blind* merupakan teknik mendapatkan pesan galat dengan membandingkan waktu HTTP Response dengan waktu aktual.
3. *Error-based* merupakan teknik yang sering digunakan dalam pengujian aplikasi berbasis web dengan memberikan query select yang dapat mempengaruhi sistem basis data.
4. *Union-based* merupakan teknik yang menambahkan query union all select sehingga mempengaruhi sistem basis data sehingga muncul galat.
5. *Stacked queries* merupakan suatu teknik yang menambahkan tanda semicolon (;) pada HTTP Request yang disertai dengan perintah SQL lainnya dengan studi kasus aplikasi web yang dibangun menggunakan stacked queries.
6. *Inline queries* merupakan teknik yang menambahkan query INLINE sehingga mempengaruhi sistem basis data untuk mendapatkan galat.

Pada umumnya serangan *SQL Injection* menyerang pada aplikasi berbasis web. Teknik dasar serangan memanfaatkan entri atau masukan dari *Uniform Resource Locator* (URL) atau alamat situs web. Pada URL tersebut ditambahkan karakter apostrof atau minus untuk mengetahui galat dari aplikasi berbasis web. Jika muncul pesan galat maka dilakukan serangan *SQL Injection* diawali mencari informasi basis data, tabel, dan kolom. Setelah informasi sensitif ditemukan maka teknik selanjutnya mengunduh data-data yang tersimpan pada basis data. Adapun alur serangan *SQL Injection* secara umum dapat dilihat pada Gambar 3.



Gambar 3. Alur Serangan SQL Injection Secara Umum

Algoritma Nearest Neighbor dapat mengklasifikasikan berdasarkan kemiripan suatu data uji dengan data uji lainnya berdasarkan nilai kedekatan suatu atribut [12]. Berdasarkan Gambar 3 pemeriksaan celah keamanan dapat dijadikan sebuah atribut yang dapat diukur nilai kedekatannya.

Tabel ***1*** merupakan definisi dari atribut deteksi serangan *SQL Injection* yang dapat diukur bobotnya:

1. Deteksi Parameter merupakan teknik deteksi celah keamanan pada URL berdasarkan suatu paramter basis data yang digunakan pada bahasa pemrograman. Sebagai contoh parameter "view_items.php?id=". Parameter tersebut juga dapat dicari menggunakan teknik *Google Dork* [13] [14].
2. *SQL Error Line* merupakan galat yang ditampilkan pada halaman situs web jika diberikan sebuah karakter (seperti tanda apostrof atau minus) pada URL [15].
3. Deteksi *Filtering* merupakan teknik untuk memeriksa implementasi dari proses filtering terhadap penambahan karakter apostrof atau minus pada URL. Jika pada URL ditambahkan karakter apostrof atau minus maka memunculkan galat dengan kode 404 [16].

Tabel 1. Atribut Nilai Deteksi SQL Injection

Atribut	Bobot
Deteksi Parameter	0,75
SQL Error Line	0,60
Deteksi *Filtering*	0,50

Penelitian ini fokus pada proses mengevaluasi celah keamanan *SQL Injection* suatu sistem atau aplikasi berbasis web. Maka pada penelitian ini dibagi menjadi tiga tahapan utama, yaitu:

1. Pengumpulan Data merupakan tahapan observasi terhadap target dan merumuskan data uji dengan melakukan definisi dan pembobotan dari deteksi serangan SQL Injection.
2. Pemrosesan Data merupakan tahapan menentukan nilai atribut dan nilai kedekatan atribut berdasarkan definisi dan bobot dari deteksi serangan *SQL Injection*.
3. Presentasi dan Pembuktian merupakan tahapan mempresentasikan hasil perhitungan berdasarkan algoritma Nearest Neighbor sebagai bahan pertimbangan dalam bentuk nilai potensi kerentanan terhadap serangan *SQL Injection*.

3. Hasil Penelitian dan Pembahasan

3.1 Pengumpulan Data

Pada tahap awal penelitian melakukan observasi terhadap target situs web yang akan dilakukan pengujian celah keamanan *SQL Injection*. Situs web yang merupakan target pengujian akan dilakukan perhitungan nilai kedekatan atribut sesuai dengan atribut deteksi serangan SQL Injection pada

Tabel ***1***. Tabel kedekatan nilai atribut terbagi menjadi tiga tabel yaitu Deteksi Paramater, *SQL Error Line*, dan Deteksi *Filtering*.

Kedekatan antara nilai-nilai dalam atribut deteksi parameter pada sebuah situs web perlu didefinisikan untuk dilakukan perhitungan prediksi terhadap potensi celah keamanan S*QL Injection*. Nilai ini berdasarkan atribut celah keamanan *SQL Injection* pada kasus sebelumnya dengan kasus yang terbaru. Tabel 2 menunjukan kedekatan nilai atribut deteksi parameter dari deteksi parameter situs web pada kasus sebelumnya dan kasus terbaru. Begitupula serupa pada Tabel 3 dan Tabel 4 yang menunjukan kedekatan nilai masing-masing atribut.

Tabel 2. Kedekatan Nilai Atribut Deteksi Parameter

Nilai 1	Nilai 2	Kedekatan
T	T	1
T	F	0,50
F	T	0,50
F	F	1

Tabel 3. Kedekatan Nilai Atribut SQL Error Line

Nilai 1	Nilai 2	Kedekatan
T	T	1
T	F	0,50
F	T	0,50
F	F	1

Tabel 4. Kedekatan Nilai Atribut Deteksi Filtering

Nilai 1	Nilai 2	Kedekatan
T	T	0,50
T	F	0,25
F	T	0,25
F	F	0,50

3.2 Pemrosesan Data

Celah keamanan berupa *SQL Injection* dalam setiap kasus serangan siber dapat dipelajari dari kasus-kasus serangan sebelumnya. Sehingga pola-pola celah keamanan berupa *SQL Injection* dapat dihitung menggunakan kedekatan antara kasus serangan sebelumnya dengan serangan saat ini. Adapun persamaan untuk menghitung kedekatan menggunakan algoritma Nearest Neighbor seperti pada Persamaan 1.

$$Kedekatan(T,S) = \frac{\sum_{i=1}^{n} f\,(T_i, S_i) * W_i}{W_i} \quad (1)$$

Persamaan 1 menunjukan rumus untuk menghitung jarak antara kasus kedua serangan siber berupa celah *SQL Injection*. Dalam persamaan tersebut berupa kasus atau potensi baru celah *SQL Injection* yang baru *(T)*, kasus atau potensi yang terjadi pada serangan siber sebelumnya *(S)*, atribut sejumlah *n* setiap kasus serangan siber, atribut serangan *SQL Injection* dengan nilai 1 s.d. n *(I).* Fungsi kedekatan serangan siber *(f)* dalam kasus baru *(T)* dan kasus sebelumnya *(S)* dengan bobot nilai *(W)* yang diberikan pada atribut pertama.

Berdasarkan Persamaan 1 dan atribut deteksi *SQL Injection* maka dapat dirumuskan jarak atau potensi celah keamanan *SQL Injection* sesuai dengan kasus sebelumnya dengan kasus baru. Persamaan 2 merupakan potensi celah keamanan *SQL Injection* pada suatu situs web. Dalam Persamaan 2, A menunjukan nilai kedekatan atribut deteksi parameter, B merupakan bobot dari deteksi paramater, C menunjukan nilai kedekatan atribut *SQL Error Line*, D merupakan bobot dari *SQL Error Line*, E menunjukan nilai kedekatan atribut deteksi *filtering*, dan F merupakan bobot dari deteksi filtering.

$$Potensi = \frac{(A * B) + (C * D) + (E * F)}{B + D + F} * 100\% \quad (2)$$

3.3 Presentasi dan Pembuktian

Pada bagian ini dilakukan pengujian dua situs web yang memiliki karakter pengembangan, yaitu: toko daring dengan alamat http://***.com/book_detail.php?bookid=7 (toko pertama) dan http://***.hk/en/e-shop.php?id=20 (toko kedua). Toko pertama tidak menerapkan penapis kueri SQL pada URL sedangkan toko kedua telah menerapkan penapis kueri SQL pada URL. Penelitian ini menggunakan situs web yang beroperasi secara langsung. Demi keamanan maka URL lengkapnya tidak dituliskan secara keseluruhan. Tahapan selanjutnya menentukan potensi celah keamanan SQL Injection dari nilai kedekatan dari kedua toko tersebut dengan kasus serangan SQL Injection yang terjadi selama ini. Tabel 5 sebagai dasar untuk menghitung nilai kedekatan dengan mencari jarak kedekatan dengan kasus sebelumnya.

Tabel 5. Nilai Kedekatan antar Kasus pada Kedua Toko Daring

Nilai	Toko Pertama	Toko Kedua
Deteksi Parameter	1,00	1,00
Bobot Deteksi Parameter	0,75	0,75
Sql Error Line	1,00	0,50
Bobot Sql Error Line	0,60	0,60
Deteksi Filtering	0,25	0,50
Bobot Deteksi Filtering	0,50	0,50

$$PotensiI = \frac{(1{,}00 * 0{,}75) + (1{,}00 * 0{,}60) + (0{,}25 * 0{,}50)}{0{,}75 + 0{,}60 + 0{,}50} * 100\%$$
$$PotensiI = \frac{1{,}475}{1{,}85} * 100\%$$
$$PotensiI = 79{,}73\% \quad (3)$$

$$PotensiII = \frac{(0{,}50 * 0{,}75) + (0{,}50 * 0{,}60) + (0{,}50 * 0{,}50)}{0{,}75 + 0{,}60 + 0{,}50} * 100\%$$
$$PotensiII = \frac{0{,}93}{1{,}85} * 100\%$$
$$PotensiII = 50\% \quad (4)$$

Potensi celah keamanan *SQL Injection* dari Toko Pertama (Potensi I) dapat dilihat pada perhitungan dari Persamaan 3, yaitu 79,73%. Artinya situs web dari Toko Pertama memiliki potensi celah keamanan *SQL Injection* sebesar 79,73%. Pembuktian menggunakan perangkat lunak SQLMap menunjukan bahwa situs web Toko Pertama memiliki celah *SQL Injection*, seperti tampak pada Gambar 4. Hasil perhitungan dari Persamaan 4 dari Toko Kedua (Potensi II) sebesar 50%. Artinya Toko Kedua telah menerapkan penapis pada penambahan karakter apostrof atai minus pada URL untuk melindungi dari serangan *SQL Injection*. Gambar 5 menunjukan hasil pengujian menggunakan perangkat lunak SQLMap pada situs web Toko Kedua.

```
[21:57:01] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.4.23, Apache 2.4.3
back-end DBMS: MySQL >= 5.5
[21:57:03] [INFO] fetching database names
available databases [3]:
[*] A[redacted]la
[*] information_schema
[*] test
```

Gambar 4. SQLMap Toko Pertama

```
[21:52:50] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[21:52:52] [WARNING] GET parameter 'id' does not seem to be injectable
[21:52:52] [CRITICAL] all tested parameters do not appear to be injectable.
Try to increase values for '--level'/'--risk' options if you wish to perform
 more tests. If you suspect that there is some kind of protection mechanism
involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--ta
mper=space2comment') and/or switch '--random-agent'
[21:52:52] [WARNING] HTTP error codes detected during run:
412 (Precondition Failed) - 75 times
```

Gambar 5. SQLMap Toko Kedua

4. Kesimpulan

Nilai potensi dari algoritma Nearest Neighbor pada tahapan pengujian di atas dapat digunakan untuk mendeteksi potensi celah keamanan *SQL Injection* berdasarkan URL pada suatu aplikasi berbasis web. Hal ini berdasarkan perbandingan kasus serangan SQL Injection

selama ini yang telah dilakukan oleh profesional bidang keamanan informasi. Tingkat akurasi algoritma Nearest Neighbor dalam mendeteksi potensi celah keamanan *SQL Injection* masih perlu ditingkatkan pada sisi atribut deteksi serangan *SQL Injection* dan pembobotan atribut. Deteksi potensi celah keamanan *SQL Injection* dapat diimplementasikan pada sistem yang lebih besar dan komplek seperti ekosistem pengembangan aplikasi berdasarkan *Security-Software Development Life Cycle*, pembuatan Bot Deteksi, *Vulnerability Assessment* (VA), *Penetration Testing* dan sebagainya.

Referensi

[1] Asosiasi Penyelenggara Jasa Internet Indonesia, "Penetrasi dan Perilaku Pengguna Internet Indonesia 2018," Jakarta, 2019.
[2] S. Samonas dan D. Coss, "The CIA Strikes Back: Redefining Confidentiality, Integrity and Availability in Security," *J. Inf. Syst. Secur.*, vol. 10, no. 3, 2014.
[3] Fazlurrahman dan D. Hariyadi, "Analisis Serangan Web Defacement pada Situs Web Pemerintah Menggunakan ELK Stack," *J. Inform. Sunan Kalijaga*, vol. 4, no. 1, hal. 1–8, 2019.
[4] I. Balasundaram dan E. Ramaraj, "An efficient technique for detection and prevention of SQL injection attack using ASCII based string matching," *Procedia Eng.*, vol. 30, no. 2011, hal. 183–190, 2012, doi: 10.1016/j.proeng.2012.01.850.
[5] Microsoft Corporation, "Microsoft Security Development Lifecycle," 2013.
[6] P. Salini dan S. Kanmani, "Survey and analysis on security requirements engineering," *Comput. Electr. Eng.*, vol. 38, no. 6, hal. 1785–1797, 2012, doi: 10.1016/j.compeleceng.2012.08.008.
[7] D. Ciptaningrum, "Audit Keamanan Sistem Informasi pada Pemerintah Kota Yogyakarta Menggunakan COBIT 5," Universitas Gadjah Mada, 2015.
[8] OWASP, "The Ten Most Critical Web Application Security Risks," 2017.
[9] P. D. Ibnugraha, L. E. Nugroho, Widyawan, dan P. I. Santosa, "Risk Analysis of Database Privilege Implementation in SQL Injection Case," *J. Teknol.*, vol. 78: 5–7, hal. 113–116, 2016.
[10] N. Patel dan N. Shekokar, "Implementation of Pattern Matching Algorithm to Defend SQLIA," *Procedia Comput. Sci.*, vol. 45, no. C, hal. 453–459, 2015, doi: 10.1016/j.procs.2015.03.078.
[11] K. N. Durai dan K. Basker, "A Novel Method for SQL Injection Detection using Assocation Rule Mining (Sqlid-Arm) and Binary Transformation," *Asian J. Res. Soc. Sci. Humanit.*, vol. 6, no. cs1, hal. 443, 2016, doi: 10.5958/2249-7315.2016.00976.X.
[12] E. Prasetyo, *Data Mining: Mengolah Data menjadi Informasi Menggunakan Matlab*. Yogyakarta: Penerbit ANDI, 2014.
[13] B. Soewito, F. E. Gunawan, Hirzi, dan Frumentius, "Prevention Structured Query Language Injection Using Regular Expression and Escape String," *Procedia Comput. Sci.*, vol. 135, hal. 678–687, 2018, doi: 10.1016/j.procs.2018.08.218.
[14] J. Clarke, *SQL Injection Attacks and Defense*. Syngress, 2009.
[15] A. B. M. Ali, A. Y. I. Shakhatreh, M. S. Abdullah, dan J. Alostad, "SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks," *Procedia Comput. Sci.*, vol. 3, hal. 453–458, 2011, doi: 10.1016/j.procs.2010.12.076.
[16] D. Kar, S. Panigrahi, dan S. Sundararajan, "SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM," *Comput. Secur.*, vol. 60, hal. 206–225, 2016, doi: 10.1016/j.cose.2016.04.005

ISSN: 2714-7975; E-ISSN: 2716-1382

.