Indonesian Journal of Computer Science
ISSN 2302-4364 (*print*) dan 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam, No. 1, Padang, Indonesia, Telp. (0751) 7056199, 7058325
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Infiltrasi Union: *SQL injection* untuk Ekstraksi Kredensial Admin

Rangga Wahyu Setiawan[1], Wahid Miftahul Ashari[2]
ranggawahyu.s14@students.amikom.ac.id, wahidashari@amikom.ac.id
Universitas Amikom Yogyakarta

Informasi Artikel

Diterima : 20 Okt 2023
Direview : 31 Okt 2023
Disetujui : 30 Nov 2023

Kata Kunci

Eksploitasi Web, Injeksi SQL, Kueri Union, Basis Data, Kredensial Admin

Abstrak

Web eksploitasi adalah suatu tindakan untuk memanfaatkan celah keamanan pada sebuah situs web untuk mendapatkan akses yang tidak sah ke sistem tersebut, SQL injection adalah jenis kerentanan keamanan yang terjadi dalam aplikasi web berbasis database di mana penyerang menyuntikkan kode berbahaya ke dalam aplikasi untuk mendapatkan akses tidak sah ke informasi sensitif. Makalah ini bertujuan untuk memberikan tinjauan yang komprehensif dan sistematis tentang metode yang ada untuk mendeteksi serangan injeksi SQL. Dalam penelitian ini, penulis mengambil aplikasi web sebagai objek dan mencoba mendemonstrasikan serangan aplikasi web umum yaitu SQL injection. Penggunaan query union memiliki peran penting dalam melakukan SQL injection pada kasus ini, karena jika dilihat dari database tersebut tidak memiliki field name username dan password. Inti serangan ini mencoba untuk menggabungkan hasil dari query orisinal dengan hasil dari query yang dicuri dari tabel "users".

Keywords

Web Exploitation, SQL Injection, Union Query, Database, Admin Credentials

Abstract

Web exploitation is an act to take advantage of security holes on a website to gain unauthorized access to the system, SQL injection is a type of security vulnerability that occurs in database driven web applications where an attacker injects malicious code into the application to gain unauthorized access to sensitive information. This paper aims to provide a comprehensive and systematic review of existing methods for detecting SQL injection attacks. In this research, the author takes a web application as an object and tries to demonstrate a common web application attack which is SQL injection. The use of query union has an important role in performing SQL injection in this case because when viewed from the database it does not have username and password fields. The core of this attack tries to combine the results of the original query with the results of the stolen query from the "users" table.

Attribution-ShareAlike 4.0 International License

A. Pendahuluan

Web eksploitasi adalah suatu tindakan untuk memanfaatkan celah keamanan pada sebuah situs web untuk mendapatkan akses yang tidak sah ke sistem tersebut[1]. Celah keamanan ini bisa berupa kesalahan kode, konfigurasi yang tidak aman, atau kerentanan lainnya[2]. Tujuan dari web eksploitasi bisa bermacam-macam, mulai dari mencuri data, merusak sistem, atau bahkan mengambil alih kontrol atas situs web tersebut[3].

Salah satu celah keamanan yang sering ditemukan pada aplikasi web adalah celah keamanan pada lapisan *database*. Celah ini dapat terjadi karena adanya *input* yang tidak difilter dengan benar. *Input* ini dapat berupa data yang dimasukkan oleh pengguna melalui formulir, *URL*, atau parameter API. Penyerang dapat memanfaatkan celah ini untuk menyisipkan kode berbahaya ke dalam *query SQL*. Kode berbahaya ini kemudian akan dieksekusi oleh *database*, yang dapat menyebabkan berbagai dampak negatif, seperti pengambilan data, manipulasi data, gangguan operasi server, dan kerusakan reputasi organisasi. Pada serangan *SQL injection*, penyerang akan memasukkan kode SQL berbahaya ke dalam *input* yang dikirim ke aplikasi[4]. Kode ini kemudian akan dieksekusi oleh *database*, sehingga penyerang dapat mengakses data atau melakukan tindakan lain yang tidak seharusnya dapat dilakukan. Secara intuitif, *SQL injection* terjadi ketika penyerang mengubah efek yang dimaksudkan dari *query SQL* dengan memasukkan kata kunci atau operator SQL baru ke dalam *query*.

Setelah proses perubahan atas *SQL query* yang akan digunakan untuk memastikan bahwa *query* tersebut dapat menghasilkan hasil yang diinginkan. Proses validasi ini penting untuk dilakukan agar data yang dihasilkan dari *query* tersebut akurat[5]. Proses validasi *SQL query* dapat dilakukan secara manual atau otomatis. Proses validasi *SQL query* secara manual dilakukan dengan memeriksa sintaks dan logika dari *query* tersebut. Sintaksis adalah aturan penulisan *query SQL* yang benar. Logika adalah cara kerja dari *query SQL* tersebut. Proses validasi *SQL query* secara otomatis dapat dilakukan dengan menggunakan alat bantu seperti SQL validator atau SQL debugger. Alat bantu ini dapat membantu memeriksa sintaks dan logika dari *query SQL* secara otomatis.

Oleh karena itu, dengan adanya serangan *SQL injection* ini dapat membuat kepedulian terhadap dalam menutup celah yang ada[6]. Solusi dari hal tersebut dapat dilakukan dengan menggunakan *SQL injection Sanitizers* yang digunakan dalam *Directory of Useful Decoy* (DUD) untuk mendeteksi intervensi dalam aplikasi berbasis web. Dan selanjutnya adalah dengan menyediakan *firewall* untuk server SQL[7]. Dalam mengatasi hal tersebut juga dapat dilakukan dengan memahami cara kerja *SQL injection* yang memanfaatkan kata kunci seperti 'WHERE', 'FROM', 'SELECT', digunakan[8]. Jika tidak menerima kata kunci ini di dalam kolom *input*, masalah ini dapat diselesaikan. Meskipun, beberapa pengguna mencoba untuk melewatkan permintaan yang sah pada *database*[9]. Jadi berhati-hatilah dalam menangani *input* dari pengguna. Jika suatu metode menjalankan perintah SELECT yang hanya menyentuh tiga tabel di *database mainframe*, metode tersebut tidak perlu memiliki izin untuk menyisipkan catatan, membuat catatan pengguna, dan menjalankan xp_cmdshell. Terdapat banyak algoritma untuk menghindari serangan *SQL injection*, salah satunya adalah Knuth-Morris-Pratt string match algorithm[10].

B. Tinjauan Pustaka

Seiring dengan semakin pentingnya peran aplikasi web dalam kehidupan sehari-hari, keamanan web semakin mendapat perhatian publik. Injeksi SQL adalah jenis kerentanan web yang paling umum dalam beberapa tahun terakhir, karena penyerang dapat memperoleh informasi privasi pengguna, atau mengontrol server melalui injeksi SQL. Metode paling abadi dan efektif untuk mendeteksi kerentanan injeksi SQL adalah pengujian penetrasi[11].

SQL injection adalah jenis kerentanan keamanan yang terjadi dalam aplikasi web berbasis *database* di mana penyerang menyuntikkan kode berbahaya ke dalam aplikasi untuk mendapatkan akses tidak sah ke informasi sensitif. Makalah ini bertujuan untuk memberikan tinjauan yang komprehensif dan sistematis tentang metode yang ada untuk mendeteksi serangan injeksi SQL. Tinjauan ini mencakup berbagai teknik, termasuk validasi *input*, kueri parameter, dan sistem deteksi intrusi, serta kelebihan dan kekurangan dari masing-masing metode. Makalah ini menyimpulkan bahwa serangan injeksi SQL terus menjadi ancaman keamanan yang signifikan bagi aplikasi web, dan penting bagi organisasi untuk menerapkan metode pencegahan dan deteksi yang efektif untuk mengamankan aplikasi web mereka terhadap serangan injeksi SQL[1].

Pernyataan SQL yang disuntikkan biasanya terdiri dari dua bagian: pernyataan SELECT yang sah dan pernyataan SELECT yang tidak valid. Pernyataan SELECT yang tidak valid berisi kode berbahaya yang ingin dijalankan oleh penyerang. Kata kunci UNION digunakan untuk menggabungkan hasil dari dua pernyataan SELECT menjadi satu hasil.

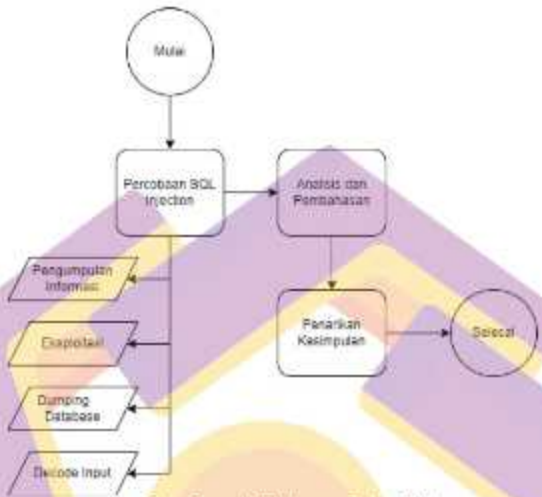
Pada penelitian lainnya yang berjudul "*SQL Injection Detection Using Machine Learning*" juga menjelaskan bahwa serangan dengan *query* UNION memungkinkan untuk mendapatkan nilai kolom dari table lain[12], namun pada penelitian sebelumnya hanya berfokus pada mendeteksi serangan *SQL Injection* saja, sedangkan dalam penelitian ini penulis mencoba mengimplementasi serangan *SQL Injection* dengan memanfaatkan *queri* UNION.

Jika aplikasi rentan terhadap serangan injeksi SQl. UNION, maka aplikasi tersebut akan mengembalikan informasi dari hasil kueri asli dan juga informasi dari tabel lain. Hal ini dapat memungkinkan penyerang untuk mengakses data sensitif yang seharusnya tidak dapat mereka akses[13].

Salah satunya adalah kontrol akses yang merupakan suatu komponen yang sangat krusial dan penting dalam keamanan sistem, biasa digunakan untuk melindungi suatu data penting atau sebuah tindakan yang sensitif, salah satunya adalah penggunaan pada *web application*. Untuk mengatur hak akses pada *web application* dengan tujuan membatasi akses penggunanya, penyimpanan hak akses pada web banyak menggunakan *database*. Namun, terdapat sebuah resiko apabila menggunakan penyimpanan pada *database*, yaitu serangan *SQL injection*[14].

C. Metode Penelitian

Proses *SQL injection* dilakukan secara manual, dengan tahapan-tahapan sebagai berikut:



Gambar 1. Tahapan Penelitian

1. Percobaan *SQL injection*

Dalam penelitian ini, penulis mengambil aplikasi web sebagai objek dan mencoba mendemonstrasikan serangan aplikasi web umum yaitu *SQL injection*.

a. Pengumpulan Informasi

Setelah mengakses *URL Target*[15], terdapat tampilan seperti pada gambar 2 yang mana *injection* pointnya terdapat pada parameter *category*.

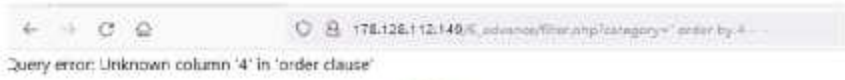
Gambar 2. *URL Target*

1. Eksploitasi

Pada bagian ini penulis menemukan bahwa total *column* yang di-*return* oleh *query* original adalah 3. Hal ini didasari pengujian dengan menggunakan *payload*[16] *' order by 3-- -* yang masih menampilkan informasi seperti gambar 3.

Gambar 3. *Input Payload*

Akan tetapi saat penulis mencoba meng*input*kan *payload ' order by 4-- -* maka server akan menampilkan *error* seperti yang ada pada gambar 4, yang mana hal ini menandakan jika jumlah *column* yang tersedia tidak lebih dari 3 *column*.



Gambar 4. *Query error*

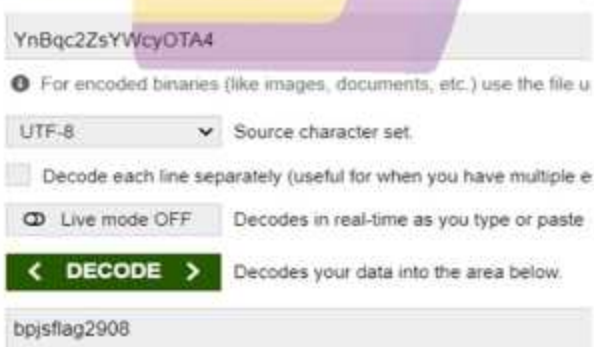
2. *Dumping Database*

Setelah melakukan eksploitasi, selanjutnya penulis melanjutkan serangan *SQL injection*[17] ini dengan menggunakan *payload ' union select 1,(select group_concat(username,password) from users),3-- -* untuk mengambil informasi *username* dan *password* dari *database*, dan hasilnya seperti pada gambar 5.

Gambar 5. Hasil *Dumping Database*

3. *Decode Input*

Setelah mendapatkan *username* dan *password*, terdapat tantangan baru yaitu bahwa *password* masih berupa hasil enkripsi dari basc64. Maka dari itu perlu melakukan *Decode*[18] agar *password* aslinya dapat di temukan, dengan menggunakan *tools online* yang tersedia *password* asli sudah ditemukan.



Gambar 6. *Decode* Base64

2. Analisis dan Pembahasan

Penggunaan *query union*[19] memiliki peran penting dalam melakukan *SQL injection* pada kasus ini, karena jika dilihat dari *database* tersebut tidak memiliki *field name username* dan *password*.

' union select 1, adalah bagian pertama dari upaya penyerangan untuk mencoba mengakhiri pernyataan SQL yang sedang dieksekusi dan memasukkan pernyataan UNION. Perintah UNION digunakan untuk menggabungkan hasil dari dua kueri yang berbeda.

(select group_concat(username,password) from users) adalah *query* yang dimaksudkan untuk mencuri data dari tabel "*users*". Penyerang mencoba untuk mengambil kolom "*username*" dan "*password*" dari tabel "*users*" dan menggabungkannya.

3-- - dan bagian ini berfungsi untuk menutup pernyataan SQL yang dimasukkan sebelumnya dan mengakhiri pernyataan SQL yang orisinal. "--" digunakan untuk membuat komentar dalam SQL, sehingga pernyataan SQL setelah itu diabaikan.

3. Penarikan Kesimpulan

Inti serangan ini mencoba untuk menggabungkan hasil dari *query* orisinal dengan hasil dari *query* yang dicuri dari tabel "*users*". Hal ini dapat memberikan penyerang akses ke informasi yang tidak seharusnya mereka miliki, seperti nama pengguna dan kata sandi dari basis data tersebut. Ini adalah contoh yang sangat sederhana dari *SQL injection*, dan dalam praktiknya, serangan semacam ini dapat jauh lebih kompleks.

D. Hasil dan Pembahasan

Dalam penelitian ini, *SQL injection* adalah salah satu kasus pencurian data paling serius dalam *database* terkait aplikasi web. *SQL injection* sering kali terjadi karena kerentanan aplikasi web dan kurangnya kesadaran tentang keamanan *database* sehingga pada penelitian ini mendapatkan kredensial admin[20].

Selanjutnya untuk mendapatkan *hash* yang menjadi *Target* serangan, penulis langsung mencoba untuk masuk sebagai admin dengan *username* dan *password* yang sudah di dapatkan saat melakukan *SQL injection*.

Gambar 7. Login Sebagai Admin

Setelah berhasil masuk sebagai admin, penulis mendapatkan sebuah *hash* yang menjadi alasan serangan *SQL injection* ini dilakukan, karena dengan nilai *hash* tersebut berguna untuk menguraikan dan mendapatkan akses ke data sensitif lainya[21].

Gambar 8. Mendapatkan *Hash*

Ada banyak cara bagi peretas untuk melakukan *SQL injection*. Oleh karena itu, untuk mencegah hal tersebut terjadi, sebagai pengembang aplikasi web, *Blockchain* harus menjadi prioritas penting dalam keamanan aplikasi web untuk memastikan seluruh data dalam *database* tetap aman dan terlindungi[14]. Keamanan aplikasi web perlu diuji untuk memeriksa apakah aplikasi tersebut rentan terhadap serangan *SQL injection*. Hal ini untuk memastikan tidak ada yang dapat membahayakan keamanan *database* aplikasi web.

E. Simpulan

Serangan *SQL injection* dapat dilakukan dengan berbagai cara, dengan menggunakan metode untuk menyuntikkan kode berbahaya ke dalam sebuah parameter *URL* aplikasi web, termasuk dengan memberikan karakter khusus atau memanfaatkan kerentanan dalam kode aplikasi yang dapat menyebabkan pencurian data, kerusakan sistem.

Kerentanan *SQL injection* yang berbasis UNION secara tidak langsung mengarah pada eksekusi kode jarak jauh, dengan peringkat resiko tinggi dan kesulitan untuk dieksploitasi rendah. Penilaian kerentanan dihitung menggunakan ketentuan yang sudah disediakan oleh NIST (*National Institute of Standards and Technology*), dengan perhitungan sebagai berikut.

Skor dasar merupakan fungsi dari persamaan subskor dampak dan eksploitasi. dimana skor dasar didefinisikan sebagai,

If (Impact sub score <= 0) 0 else,
Scope Unchanged $Roundup(Minimum[(Impact + Exploitability), 10])$
Scope Changed $Roundup(Minimum[1.08 \times (Impact + Exploitability), 10])$

Dan *Impact sub score* (ISC) didefinisikan sebagai,

Scope Unchanged $6.42 \times ISC_{Base}$
Scope Changed $7.52 \times [ISC_{Base} - 0.029] - 3.25 \times [ISC_{Base} - 0.02]15$

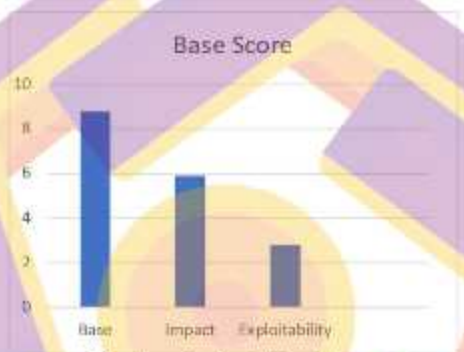
Dimana,

$$ISC_{Base} = 1 - [(1 - Impact_{Conf}) \times (1 - Impact_{Integ}) \times (1 - Impact_{Avail})]$$

Dan sub skor Eksploitabilitasnya adalah,

$$8.22 \times AttackVector \times AttackComplexity \times PrivilegeRequired \times UserInteraction$$

Hingga di dapatkanlah hasil dalam bentuk *chart* sebagai berikut.



Gambar 9. *Chart Base Score*

Dimana "*Round up*" didefinisikan sebagai angka terkecil, yang ditetapkan ke satu tempat desimal, yang sama dengan atau lebih tinggi dari *input*nya. Misalnya, *Round up* (4.02) adalah 4.1; dan Pembulatan (4,00) adalah 4,0.

Untuk mencegah serangan *SQL injection*, pengembang aplikasi web harus menerapkan langkah-langkah keamanan yang tepat. Langkah-langkah ini termasuk melakukan pengujian keamanan secara berkala, menggunakan *input* filtering, dan menerapkan kontrol akses yang ketat. Harus ada angka atau nilai yuang disampaikan.

F. Ucapan Terima Kasih

Terimakasih banyak kepada Program Studi Teknik Komputer Fakultas Ilmu Komputer Universitas Amikom Yogyakarta atas bimbingan, bantuan, dan dukungannya dalam penelitian ini.

G. Referensi

[1] Laksono, A. T., & Santoso, J. D. (2021). Analysis of Website Security of SMKN 1 Pangandaran Against *SQL injection* Attack Using OWASP Method. *The IJICS*

(International Journal of Informatics and Computer Science), *5*(2), 209. https://doi.org/10.30865/ijics.v5i2.3208

[2] Onyekachi, A. A., Agbakwuru, A. O., & Njoku, D. O. (2021). *SQL injection* Attack on Web Base Application: Vulnerability Assessments and Detection Technique Application of ICT Services in fight against COVID 19 View project An Enhanced *Query* Process Algorithm for Distributed *Database* system View project *SQL injection* Attack on Web Base Application: Vulnerability Assessments and Detection Technique. *International Research Journal of Engineering and Technology*. https://www.researchgate.net/publication/353257660

[3] Ahmad, K., & Karim, M. (n.d.). A Method to Prevent *SQL injection* Attack using an Improved Parameterized Stored Procedure. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 12, Issue 6). www.ijacsa.thesai.org

[4] Abdullayev, V., & Chauhan, Dr. A. S. (2023). *SQL injection* Attack: Quick View. *Mesopotamian Journal of Cyber Security*, 30–34. https://doi.org/10.58496/mjcs/2023/006

[5] Oakland University, IEEE Region 4, & Institute of Electrical and Electronics Engineers. (n.d.). *2018 IEEE International Conference on Electro/Information Technology (EIT) : 3-5 May 2018.*

[6] Amin Mohd Yunus, M., Zaimulariff Brohan, M., Mohd Nawi, N., Salwana Mat Surin, E., Azwani Md Najib, N., & Wei Liang, C. (n.d.). *Review of SQL injection : Problems and Prevention*.

[7] Akbar, M., & Arif Fadhly Ridha, M. (n.d.). *SQL injection and Cross Site Scripting Prevention Using OWASP Web application Firewall*.

[8] Ma, L., Zhao, D., Gao, Y., & Zhao, C. (2019). Research on *SQL injection* Attack and Prevention Technology Based on Web. *Proceedings - 2nd International Conference on Computer Network, Electronic and Automation, ICCNEA 2019*, 176–179. https://doi.org/10.1109/ICCNEA.2019.00042

[9] Viddin, I. M. S., Prihandoko, A. C., & Firmansyah, D. M. (2021). An authentication alternative using histogram shifting steganography method. *Jurnal Teknologi Dan Sistem Komputer*, *9*(2), 106–112. https://doi.org/10.14710/jtsiskom.2021.13931

[10] Abikoye, O. C., Abubakar, A., Dokoro, A. H., Akande, O. N., & Kayode, A. A. (2020). A novel technique to prevent *SQL injection* and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm. *Eurasip Journal on Information Security*, *2020*(1). https://doi.org/10.1186/s13635-020-00113-y

[11] *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*. (2018). IEEE.

[12] Krishnan, S. S. A., Sabu, A. N., Priya, ;, Sajan, P., & Sreedeep, ; A L. (2021). *SQL Injection Detection Using Machine Learning* (Vol. 11, Issue 3).

[13] Su, Z. C., Hlaing, S., & Khaing, M. (n.d.). *A Detection and Prevention Technique on SQL injection Attacks*.

[14] Fauzan, M. R., Sukarno, P., & Wardana, A. A. (n.d.). *ANALISIS DAN IMPLEMENTASI KONTROL AKSES PADA WEB BERBASIS BLOCKCHAIN*.

[15] Alghawazi, M., Alghazzawi, D., & Alarifi, S. (2022). Detection of *SQL injection* Attack Using Machine Learning Techniques: A Systematic Literature Review. *Journal of Cybersecurity and Privacy*, *2*(4), 764–777. https://doi.org/10.3390/jcp2040039

[16] Rawat, S., Bhatia, T., & Chopra, E. (n.d.). *Web application* Vulnerability Exploitation using Penetration Testing scripts. In *International Journal of Scientific Research & Engineering Trends* (Vol. 6, Issue 1). www.google.com

[17] Alanda, A., Satria, D., Isthofa Ardhana, M., Dahlan, A. A., & Mooduto, A. (n.d.). *INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION journal homepage : www.joiv.org/index.php/joiv INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION Web application Penetration Testing Using SQL injection Attack*. www.joiv.org/index.php/joiv

[18] Santoso, A. B., & Dewi, M. U. (n.d.). *Algoritma Base64 Untuk Encode Decode Sistem Keamanan Dokumen Dan Link URI. Website*.

[19] Garg, S. (2021). *Incorporating Disjunction and Union in Hidden Query Extraction A PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF Faculty of Engineering*.

[20] Alenezi, M., Nadeem, M., & Asif, R. (2020). *SQL injection* attacks countermeasures assessments. *Indonesian Journal of Electrical Engineering and Computer Science*, *21*(2), 1121–1131. https://doi.org/10.11591/ijeecs.v21.i2.pp1121-1131

[21] Legler. (2016). *Mai Mulle A Minha O U T On M A Raton Mini Us0100134426 2 (12) United States Patent Access A First Value Identifier And A First Value From A Dictionary 810 Determine , Using At Least A Hash Map And The First Value , A First Index In A Bucket List For Inserting The First Value Identifier 820 Insert The First Value Identifier At The First Index Without Inserting The First Value 830*.

LAMPIRAN

1. Letter of Acceptance (LoA)

Indonesian Journal of Computer Science
ISSN 2302-4364
Khatib Sulaiman Dalam 1, Padang, Indonesia, Phone +62-751-7056199
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Letter of Acceptance

Based on the editorial decision of Indonesian Journal of Computer Science (IJCS), we would like to inform you that your article entitled:

" **Infiltrasi Union: SQL Injection untuk Ekstraksi Kredensial Admin**"

Authors:
Rangga Wahyu Setiawan, Wahid Miftahul Ashari

has been accepted for publication in Indonesian Journal of Computer Science (IJCS).

Padang, 30 Oct 2023
Editor in Chief IJCS

Tri A. Sundara

2. Lembar Review dari IJCS (Publisher Jurnal)

a. Sebelum

Terdapat revisi dari Reviever B "Sebaiknya dilengkapi dengan penelitian sebelumnya yang terkait jika ada."

Pernyataan SQL yang disuntikkan biasanya terdiri dari dua bagian: pernyataan SELECT yang sah dan pernyataan SELECT yang tidak valid. Pernyataan SELECT yang tidak valid berisi kode berbahaya yang ingin dijalankan oleh penyerang. Kata kunci UNION digunakan untuk menggabungkan hasil dari dua pernyataan SELECT menjadi satu hasil. Jika aplikasi rentan terhadap serangan injeksi SQL UNION, maka aplikasi tersebut akan mengembalikan informasi dari hasil kueri asli dan juga informasi dari tabel lain. Hal ini dapat memungkinkan penyerang untuk mengakses data sensitif yang seharusnya tidak dapat mereka akses[12].

Salah satunya adalah kontrol akses yang merupakan suatu komponen yang sangat krusial dan penting dalam keamanan sistem, biasa digunakan untuk melindungi suatu data penting atau sebuah tindakan yang sensitif, salah satunya adalah penggunaan pada *web application*. Untuk mengatur hak akses pada *web application* dengan tujuan membatasi akses penggunanya, penyimpanan hak akses pada web banyak menggunakan *database*. Namun, terdapat sebuah resiko apabila menggunakan penyimpanan pada *database*, yaitu serangan *SQL injection*[13].

b. Sesudah

Penambahan penelitian sebelumnya berada pada halaman Tinjauan Pustaka.

Pernyataan SQL yang disuntikkan biasanya terdiri dari dua bagian: pernyataan SELECT yang sah dan pernyataan SELECT yang tidak valid. Pernyataan SELECT yang tidak valid berisi kode berbahaya yang ingin dijalankan oleh penyerang. Kata kunci UNION digunakan untuk menggabungkan hasil dari dua pernyataan SELECT menjadi satu hasil.

Pada penelitian lainnya yang berjudul "*SQL Injection Detection Using Machine Learning*" juga menjelaskan bahwa serangan dengan *query* UNION memungkinkan untuk mendapatkan nilai kolom dari table lain[12], namun pada penelitian sebelumnya hanya berfokus pada mendeteksi serangan *SQL Injection* saja, sedangkan dalam penelitian ini penulis mencoba mengimplementasi serangan *SQL Injection* dengan memanfaatkan *queri* UNION.

Jika aplikasi rentan terhadap serangan injeksi SQL UNION, maka aplikasi tersebut akan mengembalikan informasi dari hasil kueri asli dan juga informasi dari tabel lain. Hal ini dapat memungkinkan penyerang untuk mengakses data sensitif yang seharusnya tidak dapat mereka akses[13].

Salah satunya adalah kontrol akses yang merupakan suatu komponen yang sangat krusial dan penting dalam keamanan sistem, biasa digunakan untuk melindungi suatu data penting atau sebuah tindakan yang sensitif, salah satunya

3. Bukti Publish