Journal of Technology and System Information Vol: 1, No 4, 2024, Page: 1-12

Strategi Mitigasi SQL *Injection* dengan Implementasi SQL*Map* dan *Web Application Firewall*

Dea Ummul Khabibah*, Yana Nurrohman, Kenzi Dewandaru, Steven Jona Duari Huta Balian, Aep Setiawan

Teknologi Rekayasa Komputer, Sekolah Vokasi, IPB University

Abstrak: Kemajuan teknologi informasi, terutama internet, telah menjadikan internet sebagai media utama pertukaran informasi dan data di era digital. Aplikasi berbasis web menyediakan layanan global dengan akses luas bagi pengguna di seluruh dunia. Namun, kemajuan ini juga dimanfaatkan oleh penyerang untuk tujuan ilegal, seperti serangan SQL *Injection*. Penelitian ini menyoroti penggunaan teknologi *Web Application Firewall* (WAF) sebagai langkah proaktif dalam menguji dan meningkatkan ketahanan aplikasi terhadap serangan SQL *Injection*. Penelitian ini bertujuan mendalami mekanisme serangan SQL *Injection*, menerapkan teknik SQL*Map* untuk mengidentifikasi dan mengekstrak informasi sensitif dari basis data, serta memahami cara kerja SQL*Map* dalam memanfaatkan celah keamanan. Penelitian ini juga mengembangkan strategi mitigasi efektif untuk melindungi aplikasi web dari serangan SQL *Injection*. Dengan fokus pada langkah-langkah keamanan seperti WAF, penelitian ini tidak hanya meningkatkan kesadaran akan keamanan aplikasi web tetapi juga melindungi data sensitif dari ancaman serangan siber yang semakin kompleks. Implementasi SQL*Map* pada *server* Ubuntu menjadi bagian penting penelitian ini, menambah kompleksitas dalam pengujian keamanan aplikasi web dan menunjukkan relevansi teknologi *open-source* dalam konteks keamanan informasi.

Kata Kunci: Mitigasi, *Web Application Firewall*, SQL *Injection*, SQL*Map*, Ubuntu.

DOI: https://doi.org/10.47134/jtsi.v1i4.2656
*Correspondence: Dea Ummul Khabibah
Email: deaummul@apps.ipb.ac.id

Received: 01-08-2024
Accepted: 15-09-2024
Published: 31-10-2024

Abstract: *The advancement of information technology, especially the internet, has made it the primary medium for exchanging information and data in the digital era. Web-based applications provide global services with broad access for users worldwide. However, this progress has also been exploited by attackers for illegal purposes, such as SQL Injection attacks. This research highlights the use of Web Application Firewall (WAF) technology as a proactive step to test and enhance application resilience against SQL Injection attacks. The study aims to delve into the mechanisms of SQL Injection attacks, apply SQLMap techniques to identify and extract sensitive information from databases, and understand how SQLMap exploits security vulnerabilities. It also develops effective mitigation strategies to protect web applications from SQL Injection attacks. With a focus on security measures like WAF, this research not only raises awareness of web application security but also shields sensitive data from increasingly complex cyber threats. The implementation of SQLMap on Ubuntu servers is a crucial part of this research, adding complexity to web application security testing and demonstrating the relevance of open-source technology in information security contexts.*

Keywords: *Mitigation, SQL Injection, SQLMap, Web Application Firewall, Ubuntu.*

Pendahuluan

Kemajuan teknologi informasi yang begitu pesat terutama dalam bidang teknologi internet, telah menjadikan internet sebagai salah satu media pertukaran informasi dan data yang utama (Nursapdahi *et al.* 2022). Kemajuan teknologi informasi, terutama internet, memiliki hubungan erat dengan aplikasi web, yang kini menawarkan layanan global seperti komunikasi, kolaborasi, *e-commerce*, dan informasi. Namun, kemajuan ini juga dimanfaatkan oleh penyerang untuk tujuan ilegal (Yulianingsih 2016). Penyerang sering menggunakan SQL *Injection* untuk mengeksploitasi kelemahan aplikasi web. Teknik ini memanfaatkan celah di basis data, memungkinkan penyisipan kode dalam pernyataan SQL karena kelemahan dalam kode aplikasi tanpa penyaringan memadai (Ardiansyah *et al.* 2016). Serangan ini memungkinkan penyerang untuk mengubah, menghapus, atau mengakses data tanpa izin jika berhasil (Hidayah dan Saptono 2017). Senjata utama dalam serangan ini adalah SQL*Query*, yang memungkinkan penyerang meng-*inject* perintah SQL ke formulir *login* dan mengakses data dalam *database* (Singh 2012). SQL*Map* adalah teknik untuk SQL *Injection* yang memungkinkan penyerang mengakses dan memperoleh informasi dari *database* tanpa autentikasi (Puspa Ira Dewi Candra Wulan *et al.* 2023). SQL*Map* memiliki dukungan penuh untuk berbagai sistem basis data seperti *My*SQL, Oracle, dan banyak lagi. Fiturnya mencakup pencarian *database*, tabel, dan kolom tertentu di seluruh *database*, yang bermanfaat untuk mengidentifikasi informasi penting seperti ID, *username*, dan *password* (Hermawan 2021). Penerapan *Web Application Firewall* (WAF) menjadi langkah mitigasi yang penting untuk melindungi aplikasi web dari serangan yang dapat merusak atau mencuri data (Bangkit Wiguna *et al.* 2020). WAF beroperasi dengan memantau, memfilter, dan memblokir lalu lintas data antara pengguna dan aplikasi web untuk melindungi dari serangan di lapisan aplikasi, dengan fokus pada analisis permintaan HTTP untuk mengidentifikasi ancaman (Haikal Muhammad *et al.* 2024). Kemajuan teknologi internet memicu perkembangan aplikasi web dan ide-ide kreatif dari penyerang yang mengakibatkan ancaman SQL *Injection*, sehingga penggunaan *Web Application Firewall* penting untuk menguji ketahanan aplikasi terhadap serangan tersebut (Humaira *et al.* 2024). Penelitian ini mengintegrasikan pengetahuan tentang SQL *Injection*, pemahaman tentang SQL*Map*, dan implementasi praktis *Web Application Firewall* untuk meningkatkan keamanan aplikasi web dan melindungi data sensitif dari serangan siber.

Keamanan siber (*Cyber Security*) merupakan rangkaian alat, kebijakan, dan teknologi untuk melindungi aset organisasi yang mencakup perangkat, personel, infrastruktur, aplikasi, layanan, sistem telekomunikasi, dan informasi di dunia maya (Khoironi 2020). Keamanan siber mencakup aspek-aspek seperti keamanan jaringan, informasi, aplikasi, dan lainnya. Meskipun "keamanan informasi" dan "keamanan siber" memiliki kesamaan dalam proteksi aset dan perlawanan terhadap ancaman, perbedaan signifikan terletak pada fokus *cyber security* pada pengawasan komputer dan kontrol ketat, sementara keamanan informasi melibatkan isu-isu lebih luas seperti kedaulatan negara, keamanan nasional, dan perlindungan data pribadi (Budi *et al.* 2021). Keamanan siber berperan dalam mendeteksi, memperbaiki, dan mengurangi risiko ancaman serta serangan siber terhadap semua komponen sistem, termasuk perangkat keras, perangkat lunak, data, dan infrastruktur (Ramadhani dan Raf). Ancaman siber adalah keadaan, situasi, atau kapabilitas yang dapat melakukan tindakan merusak dan merugikan, mengancam kerahasiaan, ketersediaan, dan

integritas sistem dan informasi. Meskipun belum terjadi, ancaman ini berpotensi menimbulkan kerugian, dan baik alat yang digunakan maupun ancamannya berbentuk siber, bukan fisik (Putri *et al.* 2022). Serangan siber merupakan ancaman signifikan terhadap keamanan informasi dan infrastruktur teknologi yang meningkat dalam tingkat kecanggihan dan kerugian. Ini meliputi berbagai bentuk seperti *malware*, DDoS, *phishing*, dan SQL *injection* (Uzlah *et al.* 2024).

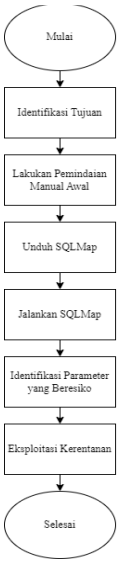
"Ubuntu" berasal dari bahasa kuno Afrika yang mengandung makna "rasa kemanusiaan terhadap sesama." Istilah ini juga dapat diartikan sebagai "saya adalah diri saya karena keberadaan kita semua." Tujuan distribusi Linux Ubuntu adalah untuk mengintegrasikan semangat kemanusiaan ini ke dalam dunia perangkat lunak (Muslim 2017). Ubuntu Server adalah sistem operasi berbasis Linux yang sering digunakan untuk mengelola *server*, seperti *server* web dan *database*. Selain itu, karena memiliki banyak *tools* dan teknik yang dapat digunakan, juga sering digunakan untuk melakukan pengujian keamanan, seperti menemukan kerentanan SQL *Injection* (Riau *et al.* 2010). Ubuntu *Server*, sering digunakan untuk mengelola server web dan *database*, juga populer dalam pengujian keamanan, termasuk menemukan kerentanan SQL *Injection*, sementara *Database Management System* (DBMS) adalah perangkat lunak yang mengelola dan memanggil kueri basis data (Saputra 2012).

Metode

Dalam metodologi penelitian ini, analisis data dilakukan menggunakan metode kualitatif yang memanfaatkan berbagai sumber dari platform *online*. Sumber-sumber tersebut mencakup jurnal ilmiah, video tutorial di *YouTube*, dan lain-lain. Data dikumpulkan dari berbagai sumber yang relevan dan dianalisis secara kualitatif untuk memahami cara penggunaan SQL*Map* dalam mengidentifikasi dan mengatasi serangan SQL *Injection*.

Selain itu, dalam teknik pengumpulan data penelitian ini juga mempertimbangkan studi kasus praktis dengan memilih aplikasi web yang diketahui memiliki celah keamanan SQL *Injection* dan menggunakan SQL*Map* untuk mengidentifikasi dan mengekstrak informasi sensitif. Data yang dikumpulkan melalui studi kasus ini mencakup informasi tentang cara SQL*Map* memanfaatkan celah keamanan, jenis informasi sensitif yang dapat diakses, serta strategi mitigasi yang efektif untuk melindungi aplikasi web dari serangan SQL *Injection*.

Serta eksperimen langsung yang dilakukan dengan menggunakan *Virtual Machine* Ubuntu dan Oracle VM VirtualBox untuk menguji SQL*Map* pada berbagai aplikasi web yang diketahui memiliki celah keamanan SQL *Injection*. Data yang dikumpulkan melalui eksperimen ini mencakup informasi tentang cara SQL*Map* memanfaatkan celah keamanan, tingkat keberhasilan serangan SQL *Injection*, serta efektivitas strategi mitigasi dengan menggunakan *Web Application Firewall*. Hasil analisis ini diharapkan dapat memberikan wawasan mendalam mengenai strategi mitigasi terbaik dan langkah-langkah pencegahan yang dapat diimplementasikan untuk meningkatkan keamanan aplikasi web. Berdasarkan analisis data dan teknik pengumpulan data, prosedur kerja dari penelitian ini terdapat pada Gambar 1.



Gambar 1. *Flowchart*

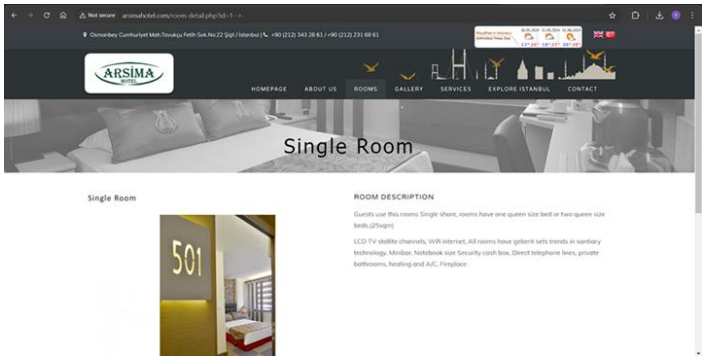
Berdasarkan hasil eksperimen yang akan dilakukan, penelitian ini membutuhkan beberapa alat (*hardware*) dan bahan (*software*) dalam pelaksanaannya. Alat atau perangkat keras yang digunakan berupa laptop dengan spesifikasi seperti yang tercantum dalam Tabel 1. Untuk bahan atau perangkat lunak yang digunakan yaitu Oracle VM VirtualBox, *Virtual Machine* Ubuntu, Internet, dan *Browser*.

Tabel 1. Spesifikasi Laptop

Spesifikasi	*Client*	*Server*
CPU	Ryzen 5 5500u 6 Core CPU	1 Core CPU
RAM	8 GB	1 GB
Storage	512 GB	10 GB
OS	Windows 11 64 Bit	Ubuntu Server 18.04
Device	Laptop	*Virtual Machine*

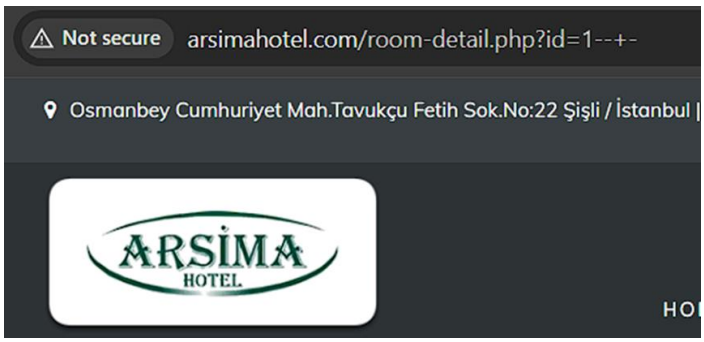
Hasil Eksperimen Serangan SQL *Injection* menggunakan Teknik SQL*Map*

Berdasarkan metode penelitian yang telah dilaksanakan berikut merupakan hasil eksperimen yang berhasil dilakukan pengujian untuk menerapkan SQL*Map* dalam serangan SQL *Injection*.



Gambar 2. *Website* target

Salah satu cara untuk mencari *website* yang rentan terhadap serangan SQL *Injection* adalah dengan mencari yang masih menggunakan protokol HTTP daripada HTTPS. *Website* HTTP cenderung lebih rentan karena data tidak dienkripsi, sehingga dapat disadap atau dimanipulasi. Contohnya, *http://arsimahotel.com/room-detail.php?id=1* seperti pada Gambar 2 yang menunjukkan bahwa situs tersebut belum menggunakan HTTPS. Hal ini memudahkan identifikasi target yang belum memiliki perlindungan memadai terhadap serangan SQL *Injection*.

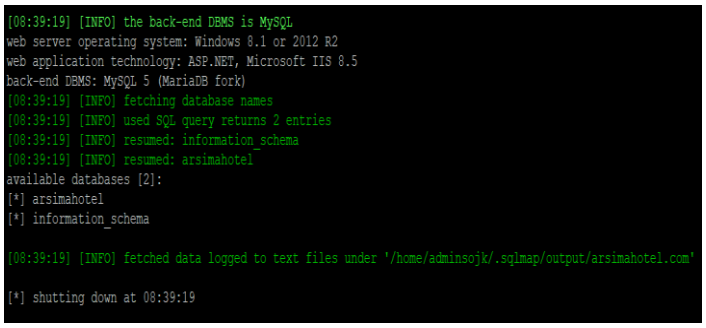


Gambar 3. *Balancer* URL

Setelah menemukan *website* yang berpotensi rentan terhadap serangan SQL *Injection* karena masih menggunakan protokol HTTP, langkah berikutnya adalah melakukan pengujian sederhana. Caranya adalah dengan menambahkan karakter *balancer* seperti " **'--+-** " pada akhir URL seperti yang terlihat pada Gambar 3. Jika tidak terdapat *error* setelah penambahan tersebut, maka dapat disimpulkan bahwa *website* tersebut rentan terhadap serangan SQL *Injection*.

Gambar 4. *Syntax* informasi *database*

Setelah menemukan bahwa *website* rentan terhadap serangan SQL *Injection*, langkah berikutnya adalah menggunakan SQL*Map* untuk menguji dan mengeksploitasi kerentanan tersebut. Gunakan parameter **--dbs** seperti Gambar 4 untuk menampilkan daftar *database* pada *server* target dan informasi tambahan tentang *back-end* seperti sistem operasi, teknologi aplikasi web, dan versi *database*, memberikan wawasan yang penting untuk mitigasi dan perbaikan kerentanan.

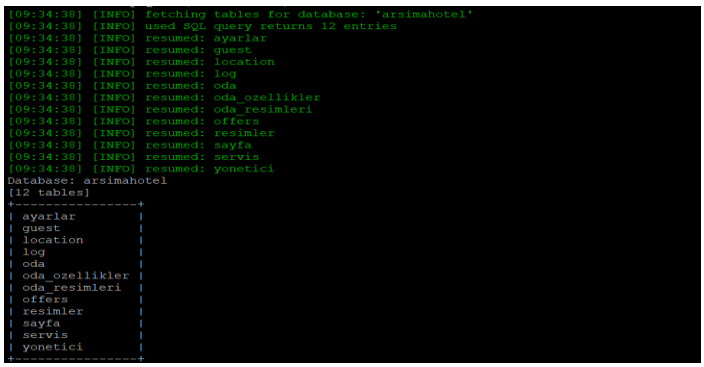


Gambar 5. *Output* informasi *database*

Setelah menjalankan perintah SQL*Map* dengan parameter "**--dbs**", *output* dari perintah SQL*Map* yang terlihat pada Gambar 5 menunjukkan bahwa *server* web menjalankan OS tertentu, menggunakan teknologi aplikasi web tertentu seperti PHP atau ASP.NET, dan menjalankan versi *database* tertentu seperti *My*SQL atau PostgreSQL.

Gambar 6. *Syntax* daftar tabel

Setelah mengetahui *database* pada *server* target, langkah selanjutnya adalah menggunakan SQL*Map* dengan parameter "**-D nama_database --tables**" seperti pada Gambar 6 untuk melihat daftar tabel dalam *database* tersebut. Dengan perintah ini, SQL*Map* mencoba mengeksploitasi kerentanan SQL *Injection* untuk mengakses struktur tabel dari *database* tersebut.

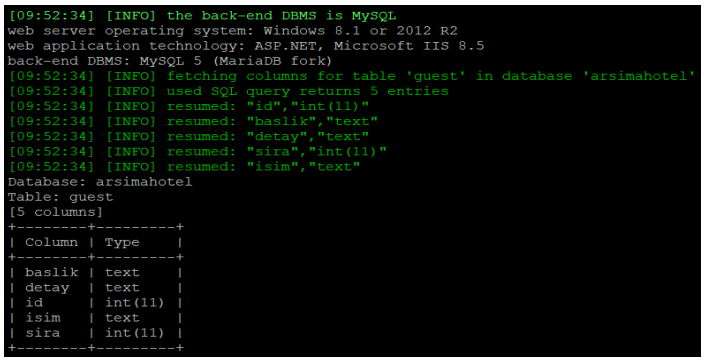


Gambar 7. *Output* daftar tabel

Output dari *query* SQL pada Gambar 7 menunjukkan terdapat 12 tabel dalam database arsima hotel, termasuk 'ayarlar', '*guest*', '*location*', 'log', 'oda', 'oda_ozellikler', 'oda_resimleri', '*offers*', 'resimler', 'sayfa', 'servis', dan 'yonetici'. Informasi ini memberikan gambaran struktur *database* arsima hotel untuk mengelola data terkait pengaturan, tamu, lokasi, log, kamar, dan manajemen.

Gambar 8. *Syntax* informasi tabel *guest*

Perintah "**sqlmap -u http://arsimahotel.com/room-detail.php?id=1 -D arsima hotel -T guest --columns**" pada Gambar 8 digunakan untuk mengidentifikasi kolom-kolom dalam tabel *guest* pada *database* arsimahotel. Hasilnya menunjukkan bahwa SQL*Map* berhasil melanjutkan proses pada *back-end* DBMS *My*SQL dan menguji koneksi ke URL target, serta melanjutkan ke titik injeksi yang disimpan dari sesi sebelumnya.

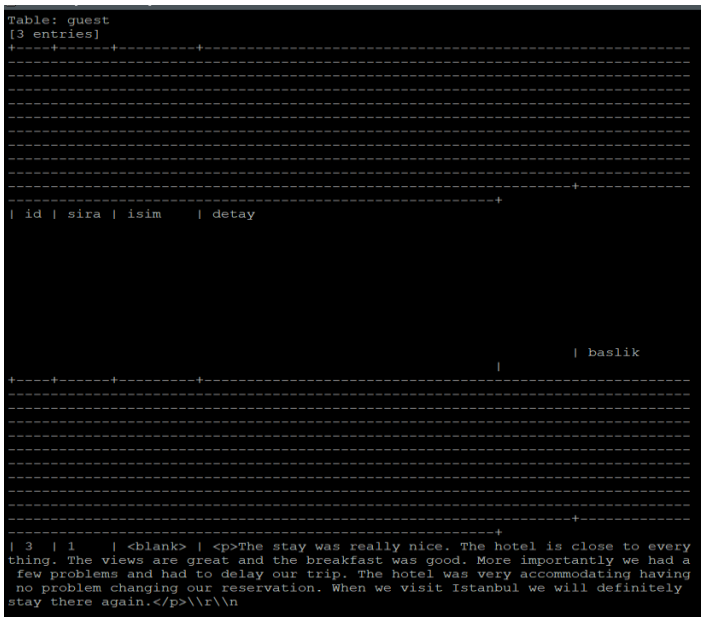


Gambar 9. *Output* informasi tabel *guest*

Output dari SQL*Map* pada Gambar 10 menunjukkan struktur tabel '*guest*' dalam *database* arsima hotel menggunakan *My*SQL versi 5 (MariaDB fork) dan dijalankan pada Windows 8.1 atau 2012 R2 dengan teknologi aplikasi web ASP.NET dan Microsoft IIS 8.5.

Gambar 10. *Syntax* isi tabel *guest*

Perintah SQL*Map* "**sqlmap -u http://arsimahotel.com/room-detail.php?id=1 -D arsimahotel -T guest --dump**" mengarahkan SQL*Map* untuk mengekstrak dan menampilkan semua data dalam tabel *guest*, memberikan detail lengkap tentang isi tabel tersebut. Ini berguna untuk analisis lebih mendalam terhadap data tamu dalam *database* arsima hotel.



Gambar 11. *Output* isi tabel *guest*

Output dari perintah SQL*Map* "**sqlmap -u http://arsimahotel.com/room-detail.php?id=1 -D arsimahotel -T guest --dump**" pada Gambar 11 berhasil mengekstrak dan menampilkan detail lengkap mengenai entri data dalam tabel *guest* di *database* arsima hotel. Ini mencakup informasi seperti ID unik tamu, judul, deskripsi, nomor urut, dan nama tamu. Data ini berguna untuk analisis lebih mendalam terkait manajemen dan keamanan data di arsima hotel.

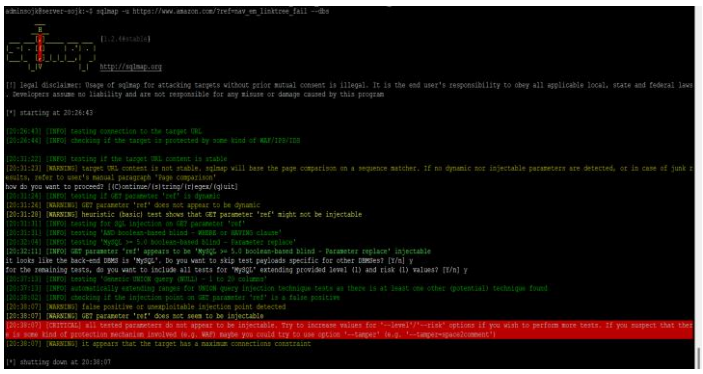
Pembahasan Serangan SQL *Injection* menggunakan Teknik SQL*Map*

Hasil eksperimen menunjukkan bagaimana SQL*Map* dapat digunakan untuk melakukan serangan SQL *Injection* pada sebuah *website* yang rentan. Dalam kasus ini, *website* target yang belum menggunakan protokol HTTPS, seperti *http://arsimahotel.com/room-detail.php?id=1*, diidentifikasi sebagai target potensial. Tahap pertama adalah menguji kerentanan dengan menambahkan karakter *balancer* pada URL, yang jika tidak memunculkan *error*, mengindikasikan kerentanan terhadap SQL Injection. Setelah mengonfirmasi kerentanan, SQL*Map* digunakan untuk mengeksploitasi celah tersebut. Dengan perintah "**sqlmap -u http://arsimahotel.com/room-detail.php?id=1 --dbs**". Langkah berikutnya adalah mengakses salah satu *database* arsimahotel, untuk melihat struktur tabelnya dengan perintah "**sqlmap -u http://arsimahotel.com/room-detail.php?id=1 -D arsimahotel --tables**". Output dari perintah ini menunjukkan adanya 12 tabel, termasuk tabel *guest*. Untuk menggali lebih dalam, kolom-kolom dalam tabel *guest* dieksplorasi menggunakan perintah "**sqlmap -u http://arsimahotel.com/room-detail.php?id=1 -D arsimahotel -T guest --columns**". Hasilnya menampilkan lima kolom dalam tabel *guest*, memberikan gambaran tentang jenis data yang disimpan. Langkah terakhir adalah mengekstrak isi dari tabel *guest* dengan perintah "**sqlmap -u http://arsimahotel.com/room-detail.php?id=1 -D arsimahotel -T guest --dump**". *Output* ini mengungkapkan semua entri data dalam tabel tersebut, termasuk ID unik, judul, detail deskriptif, urutan, dan nama tamu.

Implementasi SQL*Map* untuk melakukan penyerangan SQL Injection kali ini bertujuan untuk mengetahui mekanisme dari penyerangan SQL *Injection* menggunakan tools SQL*Map* pada Ubuntu Server. SQL *Injection* merupakan penyerangan dengan tujuan untuk mengeksploitasi *database* sebuah *website* atau aplikasi. Dengan menggunakan *tools* SQL*Map*, pengguna dengan mudah bisa mengetahui informasi database yang digunakan pada suatu *website*. Informasi ini berisi tentang *back-end* DBMS (*DataBase Management System*) yang digunakan. Tidak hanya mengetahui informasi *database*, SQL*Map* juga memungkinkan pengguna untuk mengetahui isi dari *database* seperti tabel, kolom dan tipe data yang ada pada tabel, dan isi dari tabel tersebut. Penggunaan *tools* SQL*Map* pada Ubuntu *Server* sangat efektif dalam mengeksploitasi keamanan sebuah *website*. SQL*Map* juga memiliki antarmuka *command line* yang intuitif dan mudah digunakan. Hal ini memungkinan pengguna pengguna melakukan eksploitasi dengan mudah tanpa perlu pelatihan mendalam. Otomatisasi proses pada SQL*Map* juga membuat pengguna lebih mudah mengeksploitasi *website* karena tidak membutuhkan intervensi manual yang dapat memakan waktu dan rentan terhadap kesalahan.

Hasil dan Pembahasan Mitigasi Serangan SQL *Injection* dengan Teknik SQL*Map* menggunakan *Web Application Firewall*

Untuk mencegah penyerangan SQL *Injection* pada sebuah web atau aplikasi, *developer* sebuah *website* bisa menerapkan layanan *Web Application Firewall* (WAF). *Web Application Firewall* adalah aplikasi berbasis web yang dapat memfilter, memantau, dan memblokir lalu lintas melalui *Hypertext Transfer Protocol* (HTTP). Karena kemampuan untuk memantau lalu lintas HTTP, WAF dapat mencegah serangan seperti injeksi SQL, *cross site scripting* (XSS), penyertaan *file*, dan kesalahan penginstalan (Rizal *et al.* 2020). Firewall dapat berupa perangkat lunak yang diinstal pada host atau perangkat keras khusus. Aturan WAF secara khusus dirancang untuk menyaring lalu lintas jaringan dengan menggunakan protokol HTTP. Aturan ini juga mampu mendeteksi serangan umum, seperti *probe* (upaya untuk mendapatkan informasi awal sebelum serangan) dan serangan SQL *Injection* (Riska dan Alamsyah 2021). Beberapa *website* yang telah menggunakan layanan WAF adalah Amazon, PayPal, dan Alibaba. Untuk membuktikannya, dilakukan percobaan untuk menyerang *website https://amazon.com* dengan SQL *Injection* menggunakan SQL*Map* seperti yang ditunjukkan pada Gambar 12 Hasil dari perintah SQL*Map* ditunjukan sebagai berikut:



Gambar 12. SQL *Injection* untuk *website* amazon

Percobaan untuk melakukan SQL *Injection* pada *website https://amazon.com* telah gagal. Dalam output tersebut, SQLMap mengindikasikan bahwa URL target *https://amazon.com* dilindungi oleh *Web Application Firewall* (WAF) yang mencegah serangan SQL *Injection*. WAF mendeteksi dan memblokir upaya yang dilakukan oleh SQL*Map*, menunjukkan efektivitas WAF dalam melindungi aplikasi web dari serangan semacam itu

Simpulan

Implementasi SQL*Map* pada *server* Ubuntu untuk melakukan penyerangan SQL *Injection* menunjukkan keefektifan alat ini dalam mengeksploitasi kerentanan keamanan pada aplikasi web. SQL*Map* memudahkan pengguna dalam menemukan kerentanan, mendapatkan akses ke data *database*, dan mengambil data secara otomatis tanpa memerlukan intervensi manual yang rumit. Alat ini sangat kuat karena dapat mendeteksi jenis DBMS dan menggunakan berbagai teknik injeksi. Untuk melindungi aplikasi web dari serangan seperti ini, penggunaan *Web Application Firewall* (WAF) sangat penting. WAF dapat memfilter, memantau, dan memblokir lalu lintas HTTPS yang mencurigakan, sehingga mencegah berbagai jenis serangan, termasuk SQL *Injection* dan XSS. Studi dan percobaan pada situs-situs besar seperti Amazon menunjukkan bahwa penerapan WAF dapat secara efektif mencegah serangan SQL *Injection*, memastikan keamanan data, dan menjaga integritas sistem.

Penerapan langkah-langkah keamanan seperti WAF menjadi sangat penting untuk melindungi aplikasi web dari ancaman yang terus berkembang. *Developer* dan administrator sistem harus selalu mengutamakan keamanan dengan menggunakan alat seperti WAF untuk melindungi aplikasi web mereka dari serangan berbahaya. Dengan demikian, penggunaan SQL*Map* sebagai alat untuk mengevaluasi keamanan aplikasi web harus disertai dengan penerapan WAF untuk menjaga keamanan secara keseluruhan. Seiring dengan peningkatan kompleksitas serangan *cyber*, terutama dalam hal SQL *Injection*, penting bagi organisasi untuk memperhatikan tidak hanya deteksi tetapi juga perlindungan aktif terhadap serangan tersebut. WAF berperan penting dalam melindungi aplikasi web dengan mendeteksi dan mencegah serangan yang mencurigakan. Selain itu, WAF juga dapat memberikan lapisan keamanan tambahan dengan fitur seperti analisis pola lalu lintas, pemantauan aktivitas yang mencurigakan, dan pemblokiran serangan secara *real-time*.

Penting juga untuk dicatat bahwa penggunaan SQL*Map* sebaiknya dilakukan dengan etika yang benar. Alat ini seharusnya digunakan oleh ahli keamanan atau peneliti yang bertanggung jawab untuk menguji keamanan aplikasi web yang sah dan dengan izin yang diperlukan. Menggunakan SQL*Map* untuk melakukan penyerangan ilegal atau tanpa izin adalah pelanggaran serius terhadap etika *hacking* dan hukum yang berlaku. Dalam era di mana serangan *cyber* semakin kompleks dan merugikan, mengintegrasikan alat-alat seperti SQL*Map* dan WAF menjadi keharusan bagi organisasi yang ingin menjaga keamanan dan integritas data mereka. Dengan strategi yang tepat dan implementasi yang benar, aplikasi web dapat lebih terlindungi dari ancaman *cyber* yang terus berkembang.

Daftar Pustaka

Ardiansyah SS, Raharjo S, Triyono J. 2016. Analisis Keamanan Serangan SQL Injection berdasarkan Metode Koneksi Database Jurnal SCRIPT Vol . 4 No . 1 Desember 2016 ISSN : 2338-6313. *J. Scr.* 4(1):79–87.

Bangkit Wiguna, Adi Prabowo W, Ananda R. 2020. Implementasi Web Application Firewall Dalam Mencegah Serangan SQL Injection Pada Website. *Digit. Zo. J. Teknol. Inf. dan Komun.* 11(2):245–256.doi:10.31849/digitalzone.v11i2.4867.

Budi E, Wira D, Ardian I. 2021. Strategi Penguatan Cyber Security Guna Mewujudkan Keamanan Nasional di Era Society 5.0. *Pros. Semin. Nas. Sains Teknol. dan Inov. Indones.* 3(November):223–234.doi:10.54706/senastindo.v3.2021.141.

Haikal Muhammad H, Id Hadiana A, Ashaury H. 2024. Pengamanan Aplikasi Web Dari Serangan Sql Injection Dan Cross Site Scripting Menggunakan Web Application Firewall. *JATI (Jurnal Mhs. Tek. Inform.* 7(5):3265–3273.doi:10.36040/jati.v7i5.7320.

Hermawan R. 2021. Teknik Uji Penetrasi Web Server Menggunakan SQL Injection dengan SQLmap di Kalilinux. *STRING (Satuan Tulisan Ris. dan Inov. Teknol.* 6(2):210.doi:10.30998/string.v6i2.11477.

Hidayah T, Saptono H. 2017. Jurnal Informatika Terpadu Penerapan High Availability Web Server Menggunakan Nginx dan Modsecurity. *J. Inform. Terpadu*. 3(2):95–102.

Humaira NH, Hadiana IA, Ashaury H. 2024. Analisis Ketahanan Web Application Firewall Terhadap Serangan SQL Injection. *J. Ilm. Wahana Pendidik.* 10(5):403–412.

Khoironi SC. 2020. Pengaruh Analisis Kebutuhan Pelatihan Budaya Keamanan Siber Sebagai Upaya Pengembangan Kompetensi bagi Aparatur Sipil Negara di Era Digital. *J. Stud. Komun. dan Media*. 24(1):37.doi:10.31445/jskm.2020.2945.

Muslim MA. 2017. Pengembangan Distro Ubuntu untuk Aplikasi Game Centre. *J. Sains dan Seni ITS*. 6(1):51–66.

Nursapdahi, Senja Fitrani A, Alfan Rosid M, Aji S. 2022. Studi Analisa Serangan Sql Injection. *Semin. Nas. Inov. Teknol.*:185–190.

Puspa Ira Dewi Candra Wulan, At Tafani Filah, Rivort pormes, Rohmatulloh Muhamad Ikhsanuddin. 2023. Audit Kerentanan Menggunakan Sqlmap Dan Reserve Shell Pada Website Staff Bhakti Semesta. *J. Data Sci. Theory Appl.* 2(1):33–44.doi:10.32639/jasta.v2i1.309.

Putri AWOK, Aditya ARM, Musthofa DL, Widodo P. 2022. Serangan Hacking Tools sebagai Ancaman Siber dalam Sistem Pertahanan Negara (Studi Kasus: Predator). *Glob. Polit. Stud. J.* 6(1):35–46.doi:10.34010/gpsjournal.v6i1.6698.

Ramadhani MR, Raf A. Analisis Kesadaran Cybersecurity Pada Pengguna Media Sosial Di Indonesia.

Riau UINS, Suryadi D, Sains F, Teknologi DAN, Informatika JT, Riau UINS. 2010. Tugas akhir judul kostumisasi ubuntu 9.10 untuk kegiatan pembelajaran bahasa pemrograman berbasis open source.

Riska R, Alamsyah H. 2021. Penerapan Sistem Keamanan Web Menggunakan Metode Web Aplication Firewall. *J. Amplif. J. Ilm. Bid. Tek. Elektro Dan Komput.* 11(1):37–42.doi:10.33369/jamplifier.v11i1.16683.

Rizal R, Ruuhwan R, Nugraha KA. 2020. Implementasi Keamanan Jaringan Menggunakan Metode Port Blocking dan Port Knocking Pada Mikrotik RB-941. *J. ICT Inf. Commun. Technol.* 19(1):1–8.doi:10.36054/jict-ikmi.v19i1.119.

Saputra A. 2012. Manajemen Basis Data Mysql Pada Situs FTP Lapan Bandung. *J. Ber. Dirgant.* 13(4):155–162.

Singh N. 2012. Sql i – a w. 2(6):42–46.

Uzlah LI, Saputra RA, Isnawaty. 2024. Deteksi Serangan Siber pada Jaringan Komputer menggunakan Metode Random Forest. 8(3):2787–2793.

Yulianingsih Y. 2016. Menangkal Serangan SQL Injection Dengan Parameterized Query. *J. Edukasi dan Penelit. Inform.* 2(1):46–49.doi:10.26418/jp.v2i1.15507.