PENGAMANAN APLIKASI WEB DARI SERANGAN SQL INJECTION DAN CROSS SITE SCRIPTING MENGGUNAKAN WEB APPLICATION FIREWALL

Husein Haikal Muhammad, Asep Id Hadiana, Herdi Ashaury
Informatika, Universitas Jenderal Achmad Yani Cimahi
Jl. Terusan Jend. Sudirman, Cibeber, Kec. Cimahi Sel., Kota Cimahi, Jawa Barat 40531
husein.haikal@student.unjani.ac.id

ABSTRAK

Aplikasi web merupakan salah satu media yang paling dominan digunakan untuk pengiriman data melalui internet saat ini. banyaknya aplikasi web yang tersedia di internet saat ini membuatnya sering dijadikan sasaran untuk berbagai jenis serangan seperti SQL Injection Dan XSS. Maka dari itu diperlukannya penerapkan sistem keamanan yang dapat mencegah serangan SQL Injection dan XSS yang masuk ke aplikasi web. Web Application Firewall (WAF) adalah perangkat atau sistem yang digunakan untuk melindungi aplikasi web dari serangan yang dapat merusak atau mencuri data. dalam penelitian ini bertujuan untuk mengimplementasikan Web Application Firewall guna mencegah serangan SQL injection dan Cross Site Scripting (XSS) pada aplikasi web. Metode Web Application Firewall digunakan untuk memblokir serangan SQL injection berdasarkan aturan yang telah ditentukan sebelumnya. Penelitian ini menggunakan Web Application Firewall ModSecurity sebagai mekanisme untuk mencegah serangan SQL Injection dan Cross Site Scripting (XSS) pada web server. penelitian yang diusulkan bertujuan untuk menganalisis efektivitas penggunaan WAF dalam melindungi aplikasi web dari serangan SQL injection dan XSS. Dari hasil beberapa pengujian yang dilakukan, Web Application Firewall Modsecurity berhasil mendeteksi dan memblokir serangan SQL Injection dan Cross Site Scripting (XSS) yang masuk ke Aplikasi web.

Kata kunci : *Web Application Firewall(WAF), Vulnerability, Security, ModSecurity, SQL Injection, Cross Site Scripting (XSS)*

1. PENDAHULUAN

Aplikasi web merupakan salah satu media yang paling dominan digunakan untuk pengiriman data melalui internet saat ini. Aplikasi web merupakan aplikasi yang tersimpan pada web server dan dieksekusi oleh web server[1]. Fungsi web server adalah untuk menyediakan konten web kepada pengguna melalui internet. Ini dilakukan dengan menerima permintaan dari web browser melalui protokol HTTP atau HTTPS, dan mengirimkan kembali konten web yang diminta, seperti halaman HTML, gambar, atau file multimedia [2]. Seiring dengan berkembangnya internet aplikasi web pun turut ikut berkembang dan menjadi populer, banyaknya aplikasi web yang tersedia di internet saat ini membuatnya sering dijadikan sasaran untuk berbagai jenis serangan oleh orang-orang yang memiliki niat buruk[3]. Aplikasi web pastinya memiliki data – data pribadi setiap pengguna jika data tersebut bisa diakses oleh pengguna lain yang tidak memiliki hak akses maka dapat berakibat fatal.

Ada banyak jenis serangan web yang sering terjadi saat ini, SQL Injection dan Cross Site Scripting (XSS) merupakan kedua jenis serangan yang sering digunakan oleh penyerang (*attacker*)[4]. SQL Injection bekerja dengan cara mengirimkan perintah SQL melalui URL atau inputan di dalam halaman aplikasi web yang nantinya dieksekusi oleh web server dan akan menampilkan data – data didalam databases[5]. Lalu ada jenis serangan Cross Site Scripting (XSS),jenis serangan ini dilakukan dengan cara memasukan kode script kedalam form yang tersedia di halaman aplikasi web yang nantinya akan dieksekusi oleh web server dan akan menampilkan cookies, session token, dll[6]. Di tahun 2020 kedua jenis serangan ini masih menjadi masalah yang sering terjadi, serangan Cross Site Scripting (XSS) tercatat sebanyak 3,068 kasus selama tahun 2020, lalu SQL Injection dengan kasus sebanyak 1,711 [7].

Ada beberapa cara untuk mencegah serangan pada aplikasi web. Seperti yang telah dilakukan oleh peneliti sebelumnya[8], melakukan penelitian untuk mendeteksi serangan SQL Injection dan Cross Site Scripting (XSS) dengan menggunakan metode Intrution Detection System (IDS)[9]. Metode ini berkerja dengan cara memantau lalu lintas jaringan untuk aktivitas yang mencurigakan dan mengeluarkan peringatan saat aktivitas tersebut ditemukan, hasil dari penerapan metode ini peneliti berhasil mendeteksi serangan SQL Injection dan XSS, namun kekurangan dari metode ini hanya bisa mendeteksi serangan saja tetapi tidak bisa memblokir serangan tersebut secara otomatis.

Ada juga penelitian dilakukan[10], Studi ini melakukan evaluasi terhadap efektivitas penggunaan Web Application Firewall (WAF) dalam mencegah serangan SQL injection. Peneliti melakukan serangkaian uji coba dan analisis untuk menentukan sejauh mana WAF dapat melindungi website dari serangan SQL injection. Hasil penelitian ini mendukung penggunaan WAF sebagai solusi yang efektif dalam mencegah serangan SQL injection. [11], WAF hanya difokuskan untuk melindungi jaringan trafik aplikasi web (HTTP/S), Cara kerja WAF adalah

dengan menerima trafik web yang masuk ke aplikasi web lalu WAF menganalisis trafik tersebut dengan menggunakan aturan yang telah ditentukan, selanjutnya WAF mengevaluasi trafik tersebut untuk menentukan apakah permintaan tersebut valid atau tidak [12].

Dari dua penelitian terdahulu yang sudah dijelaskan ada beberapa alasan yang menjadi tujuan dari penelitian ini yaitu. Pada penelitian sebelumnya yang menggunakan IDS penelitian melakukan pengujian serangan SQL Injection dan Cross Site Scripting namun dikarenan sistem kemanan IDS hanya bisa mendeteksi serangan dan memblokir serangan secara manual maka diperlukan sistem keamanan yang dapat mendeteksi serangan dan juga memblokir serangan tersebut secara otomatis. Maka dari itu penelitian ini menggunakan WAF Modsecurity sebagai sistem keamanan pada aplikasi web dikarenakan sistem keamanan ini berfingsi sesuai dengan kebutuhan penelitian ini yaitu sistem keamanan yang dapat mendeteksi serangan dan juga memblokir serangan tersebut secara otomatis. Walaupun sudah ada penelitian terdahulu yang menggunakan WAF penelitian hanya berfokus pada pencegahan serangan SQL Injection saja. Maka dari itu dalam penelitian ini menambahkan satu jenis serangan lagi yaitu Cross Site Scripting (XSS), alasanya kedua jenis serangan ini masih menjadi masalah yang sering terjadi [7].

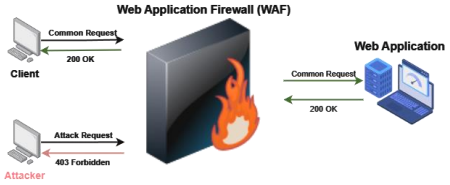
Penelitian ini berfokus pada dua jenis serangan injection dikarenakan SQL Injection dan Cross Site Scripting adalah dua jenis serangan yang sangat umum dan memiliki potensi risiko yang signifikan bagi aplikasi web. SQL Injection terjadi ketika penyerang memasukkan kode SQL berbahaya ke dalam input yang tidak terlindungi di aplikasi web, sedangkan XSS melibatkan penyisipan kode skrip berbahaya ke dalam halaman web yang kemudian dieksekusi oleh pengguna lain. Kedua serangan ini telah banyak menyebabkan pelanggaran keamanan data dan privasi. penggunaan Web Application Firewall (WAF) untuk melindungi terhadap serangan ini adalah pendekatan yang umum digunakan dalam pengamanan aplikasi web. Fokus pada dua jenis serangan ini mungkin ditujukan untuk menilai efektivitas WAF dalam mengatasi serangan yang umum terjadi.

Berdasarkan uraian diatas tujuan dari penelitian ini adalah melakukan penerapan dan menganalisis efektivitas dari Web Application Firewall (WAF) dalam mencegah serangan SQL Injection dan Cross Site Scripting dan Aplikasi web yang digunakan adalah DVWA. Dari hasil beberapa pengujian yang dilakukan, Web Application Firewall Modsecurity berhasil mendeteksi dan memblokir serangan SQL Injection dan Cross Site Scripting (XSS) yang masuk ke Aplikasi web.

2. TINJAUAN PUSTAKA

2.1. Web Application Firewall (WAF)

Web Application Firewall (WAF) adalah metode keamanan dalam aplikasi web yang digunakan untuk melindungi aplikasi web dari serangan yang dapat merusak atau mencuri data[3]. WAF berfungsi dengan memantau, memfilter, dan memblokir data antara klien dan aplikasi web. WAF dapat berbasis jaringan, berbasis host, atau berbasis cloud, dan fokus pada serangan berbasis web di lapisan aplikasi. WAF menganalisis permintaan HTTP, terutama metode GET dan POST, untuk menentukan bagian yang sah atau mencurigakan. WAF memberikan perlindungan terhadap eksploitasi, malware, dan ancaman yang melebihi firewall tradisional. [13].



Gambar 1. Skema cara kerja WAF

2.2. Core Rule Set (CRS)

Core Rule Set (CRS) adalah kumpulan aturan deteksi serangan yang digunakan bersama dengan ModSecurity atau Web Application Firewall (WAF)l. CRS bertujuan untuk melindungi aplikasi web dari berbagai jenis serangan, termasuk, sambil meminimalkan alarm palsu. Kumpulan aturan ini memberikan pertahanan terhadap kategori serangan umum seperti SQL Injection dan Cross Site Scripting [14].

CRS dapat digunakan dengan bebas dan disesuaikan sesuai kebutuhan dengan atribusi yang tepat, karena CRS dilisensikan di bawah Apache License 2.0. Aturan yang terdapat dalam CRS dirancang untuk melindungi aplikasi web dari serangan umum yang dapat mempengaruhi keamanan dan kerentanan aplikasi [15].

2.3. Mod Security

ModSecurity[16] bisa disebut juga ModSec merupakan Salah satu aplikasi Web Application Firewall (WAF) khusus untuk mendukung web server Apache dan Nginx yang bersifat Open Source. Server yang diinstal ModSecurity akan melakukan pertahanan di tingkat aplikasi web. ModSecurity menyediakan rule konfigurasi yang disebut SecRules untuk memantau traffic HTTP/S secara real-time, pencatatan (logging) dan melakukan filter pada komunikasi HTTP/S berdasarkan aturan yang sudah ditentukan. ModSecurity menghentikan atau block pada lalu lintas yang dianggap berbahaya atau membahayakan dari pada website. [17]

2.4. SQL Injection

SQL injection adalah teknik penyalahgunaan celah keamanan dengan menjalankan perintah SQL

Command ke dalam string kueri pengiriman formulir Web atau memasukkan nama domain atau permintaan halaman. Serangan SQL Injection terjadi ketika pernyataan SQL dinamis dibangun menggunakan konten input untuk mengakses database. SQL Injection juga terjadi jika kode menggunakan stored procedure, yang diteruskan sebagai string yang berisi input pengguna tanpa filter.[12]

2.5. Cross Site Scripting (XSS)

XSS adalah eksploitasi keamanan yang dimana penyerang melakukan Code injection ke dalam Halaman HTML yang ditampilkan kepada pengguna. Jika code yang dimasukan berhasil di eksekusi, tindakan atau perilaku sistem atau situs web dapat sepenuhnya diubah dan memungkinkan penyerang dapat mengakses data user yang seharusnya tidak dapat diakses [18].

2.6. Penetration Testing

Penetration testing adalah metode sistematis dan disiplin ilmu yang digunakan untuk menguji keamanan sistem komputer, jaringan, aplikasi, atau infrastruktur IT lainnya. Tujuan utama dari penetration testing adalah untuk mengidentifikasi kerentanan dan kelemahan dalam sistem yang dapat dieksploitasi oleh penyerang potensial. Metode ini melibatkan upaya simulasi serangan nyata terhadap sistem untuk mengukur sejauh mana sistem tersebut dapat tahan terhadap serangan dan apakah ada celah yang dapat dimanfaatkan oleh pihak yang tidak sah. Penetration testing melibatkan serangkaian langkah dan teknik yang kompleks, mulai dari pemetaan dan pemahaman tentang target yang akan diuji, identifikasi kerentanan potensial, hingga eksploitasi secara terkontrol untuk membuktikan bahwa suatu kerentanan dapat dieksploitasi. Hasil dari pengujian ini memberikan wawasan yang berharga kepada pemilik sistem atau organisasi terkait tingkat keamanan sistem mereka, serta rekomendasi tentang tindakan perbaikan yang perlu diambil untuk meminimalkan risiko keamanan.[19]

3. METODE PENELITIAN

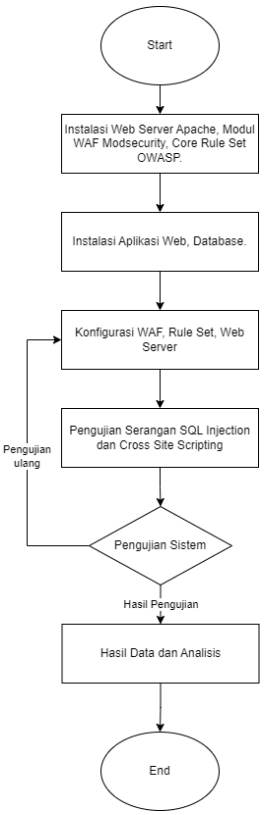
Dalam melakukan penelitian ini memiliki beberapa tahapan yang dilakukan, seperti dapat dilihat pada gambar 2.

3.1. Tahap Perancangan

Pada tahap ini akan dilakukan Perancangan Web server, WAF, Rule set, Aplikasi web, Database yang digunakan, lalu melakukan konfigurasi agar penelitian berjalan sesuai dengan yang dibutuhkan untuk melakukan tahap eksperimen.

3.2. Tahap Eksperimen

Pada tahap ini akan melakukan eksploitasi pada aplikasi web untuk menguji efektifitas dalam penerapan Web Application Firewall, setelah melakukan pengujian akan menghasilkan hasil eksperimen yang nantinya akan dilanjutkan ke tahap Analisis.



Gambar 2. Metode Penelitian

3.3. Tahap Analisis

Pada tahap ini melakukan Analaisis terhadap hasil eksperimen yang sudah dilakukan pada tahap eksperimen.

3.4. Tahap Akhir

Ada tahap akhir ini berupa laporan penelitian, dengan langkah terakhir menarik kesimpulan yang diperoleh dari hasil analisis penelitian.

4. HASIL DAN PEMBAHASAN

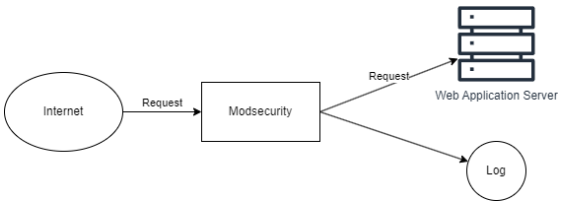
4.1. Perancangan Web Application Firewall (WAF) Modsecurity

ModSecurity adalah Web Application Firewall (WAF) yang berfungsi sebagai modul tambahan pada Apache. Ada beberapa fitur dari ModSecurity diantaranya pemeriksaan log, akses ke setiap bagian dari permintaan yang ditujukan ke server (termasuk isi permintaan atau body), memberikan respon terhadap hasil pemeriksaan, dan memiliki aturan yang didasarkan pada regular expression yang fleksibel. Selain itu, ModSecurity juga melakukan pemeriksaan pada berkas yang diunggah, validasi real-time, dan menyediakan perlindungan terhadap buffer-overflow. Modsecurity memiliki fungsi modul yang dibagi menjadi empat bidang yaitu:

1. Parsing: Modul ini bertanggung jawab untuk mengurai dan menganalisis request HTTP yang masuk ke web server. Proses ini mencakup

pemecahan request menjadi bagian-bagian seperti URL, header, parameter, dan body.

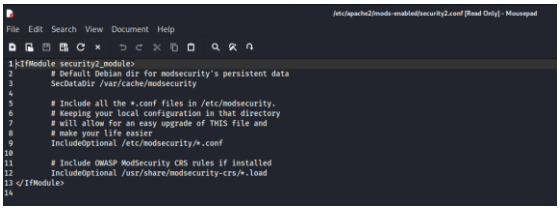
2. Buffering: Buffering berfungsi untuk membuffer request dan response, di mana request body akan disimpan dalam buffer sehingga modul biasanya dapat melihat seluruh isi request dengan lengkap sebelum diteruskan ke aplikasi untuk pengolahan, begitu juga dengan response lengkap sebelum dikirim ke client. Buffering menjadi sangat penting, karena merupakan satu-satunya cara untuk memberikan blokir yang handal (reliable blocking).
3. Logging: Logging Berfungsi untuk mencatat seluruh traffic HTTP pada jaringan secara lengkap, dengan menyimpan dalam bentuk log seluruh header request dan response, serta isi pesan (body).
4. Rule Matching: ModSecurity menggunakan aturan keamanan (rule) untuk mendeteksi serangan atau perilaku berbahaya pada permintaan yang masuk. Modul ini berfungsi untuk mencocokkan konten permintaan dengan aturan-aturan yang telah ditentukan sebelumnya untuk mengidentifikasi potensi serangan.
5. Modsecurity dapat digunakan dalam dua mode: yaitu mode Embedded, dengan menambahkan modsecurity sebagai modul ke Server Apache. Namun, dalam mode ini, modsecurity tidak dapat memeriksa isi header Server. Modus Gateway (Network Gateway mode), di mana semua lalu lintas web melewati proxy, dan modsecurity diinstal sebagai reverse proxy, seperti yang ditunjukkan pada gambar 3.



Gambar 3. Alur Modsecurity

4.2. Konfigurasi WAF pada web server Apache.

Pada konfigurasi ini berfungsi untuk meyambungkan WAF Mosecurity pada Apache2, terdapat tiga sintaks yang berfungsi untuk menyambungkan fungsi-fungsi yang ada di WAF Modsecurity, pada sintaks pertama "SecDataDir /var/cache/modsecurity" berfungsi untuk menyimpan semua history log request http yang terjadi di aplikasi web, lalu sintaks kedua "IncludeOptional /etc/modsecurity/*conf" yaitu berfungsi untuk konfigurasi utama pada WAF Modsecurity tersebut terutamanya pada pengaktifan sistem keamanan, lalu pada sintaks ketiga "IncludeOptional /usr/share/modsecurity-crs/*.load" berfungsi untuk menyambungkan Core Rule Set (CRS) pada apache2 agar WAF Modsecurity dapat menangkal serangan dengan menggunakan text filter yang tersedia di CRS. Seperti yang di tampilkan pada Gambar 4.



Gambar 4. Konfigurasi apache untuk menyambungkan mod security

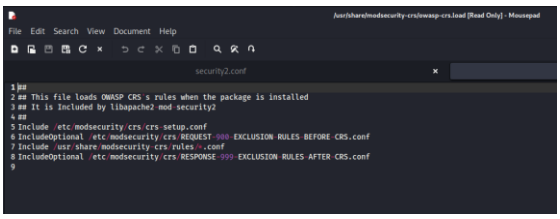
4.3. Konfigurasi Modsecurity

Pada konfigurasi ini berfungsi untuk konfigurasi utama pada Modsecurity, di file konfigurasi ini terdapat fungsi untuk mengaktifkan dan menonaktifkan sistem keamanan. Seperti yang di tampilkan pada Gambar 5.

Gambar 5. Konfigurasi Modsecurity

4.4. Konfigurasi pada Modsecurity CRS

Pada konfigurasi ini fungsinya utamanya untuk memuat semua konfigurasi text filter yang ada seperti pada Gambar 6.



Gambar 6. Konfigurasi Modsecurity-crs

4.5. Tools yang digunakan

Alat yang digunakan pada penelitian ini yaitu berupa web server, aplikasi web, sistem keamanan, dan alat penyerangan aplikasi web. Pengujian sistem ini akan menggunakan spesifikasi perangkat keras dan sqlmap dan xsstrike sebagai tools tambahan untuk pengujian SQL Injection dan Cross Site Scripting.

Tabel 1. Tools yang digunakan

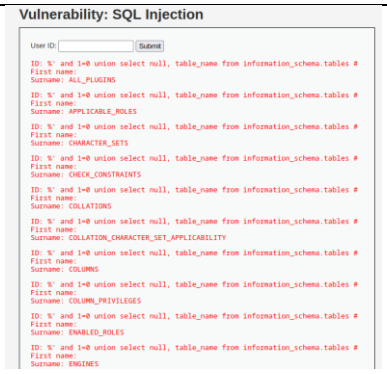
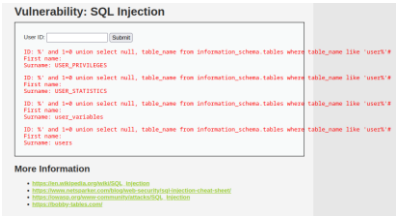
Tools	Deskripsi
Kali Linux	Distribusi Linux berbasis Debian yang dirancang khusus untuk pengujian keamanan dan penetrasi jaringan.
Apache2	Web server
DVWA	aplikasi web open souce yang dikembangkan untuk tujuan pembelajaran dan pelatihan dalam pengujian penetrasi serta pengembangan keamanan web.
Modsecurity	Sebuah sistem keamanan firewall open source yang digunakan untuk melindungi aplikasi web

Tools	Deskripsi
	dari berbagai serangan dan ancaman keamanan.
CRS	seperangkat aturan keamanan ModSecurity WAF untuk meningkatkan keamanan aplikasi web
SQLMAP	Alat yang digunakan untuk melakukan serangan SQL injection
XSSTRIKE	Alat yang digunakan untuk melakukan serangan Cross Site Scripting (XSS)

4.6. Pengujian Serangan SQL Injection Tanpa Keamanan WAF

Dalam melakukan serangan SQL injection ini, penulis melakukan serangan secara manual dan juga menggunakan tools yang bernama SQLMap. Pada pengujian pertama akan melakukan serangan SQL injection terhadap aplikasi web yang belum menggunakan Web Application Firewall (WAF). Hasil dari serangan SQL Injection berhasil membobol database pada Aplikasi web dengan menggunakan command pada Tabel 2.

Tabel 2. SQL Injection query yang digunakan untuk menyerang aplikasi web tanpa WAF

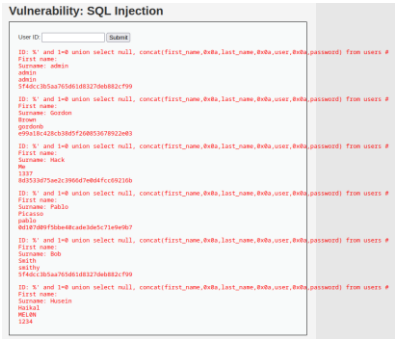
Manual SQL Injection Query
1' and 1 = 1# Dalam query di atas, bagian '1' and 1 = 1#' disisipkan ke dalam klausa WHERE sebagai nilai dari kolom username. Bagian 1 = 1 adalah pernyataan logika yang selalu benar, sehingga kondisi ini akan selalu dievaluasi sebagai benar, tidak peduli apa pun nilai asli dari kolom username. Tanda pagar (#) digunakan untuk memberi tahu database untuk mengabaikan sisa query setelahnya.
 Gambar 7. Hasil serangan SQL Injection manual 1
%' and 1=0 union select null, table_name from information_schema.tables # Dalam query di atas, bagian %' and 1 = 0 UNION SELECT NULL, table_name FROM information_schema.tables # disisipkan ke dalam klausa WHERE sebagai nilai dari kolom username. Bagian 1 = 0 adalah pernyataan logika yang selalu salah, sehingga kondisi ini akan selalu dievaluasi sebagai salah. Namun, dengan menggunakan pernyataan UNION, penyerang mencoba untuk menggabungkan hasil dari query asli dengan hasil dari query tambahan yang mengambil nama-nama tabel dari information_schema.tables.
 Gambar 8. Hasil seramgam SQL Injection manual 2
%' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'# Dalam query ini, bagian %' and 1 = 0 UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_name LIKE 'user%' # disisipkan ke dalam klausa WHERE sebagai nilai dari kolom username. Seperti sebelumnya, bagian 1 = 0 adalah pernyataan logika yang selalu salah, dan dengan menggunakan pernyataan UNION, penyerang mencoba untuk menggabungkan hasil dari query asli dengan hasil dari query tambahan. Pada kali ini, query tambahan mencari nama-nama tabel dari information_schema.tables yang dimulai dengan "user" menggunakan klausa WHERE table_name LIKE 'user%'. Hasil dari query tambahan ini akan ditampilkan sebagai bagian dari hasil query asli, memberikan penyerang informasi tentang tabel-tabel yang sesuai dengan pola yang diinginkan.
 Gambar 9. Hasil serangan SQL Injection manual 3
%' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' # Pada query ini tambahan menggunakan pernyataan CONCAT untuk menggabungkan nama tabel dan nama kolom dalam format yang lebih mudah dibaca. Ekspresi 0x0a mewakili karakter newline dalam bentuk heksadesimal, sehingga memungkinkan hasil yang lebih terstruktur. Query tambahan ini mencari kolom-kolom dalam tabel 'users' menggunakan klausa WHERE table_name = 'users'. Hasil dari query tambahan ini akan ditampilkan sebagai bagian dari hasil query asli, memberikan penyerang informasi tentang nama-nama kolom dalam tabel 'users'.

Gambar 10. Hasil serangan SQL Injection manual 4

%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #

Dalam query ini, bagian %' and 1 = 0 UNION SELECT NULL, CONCAT(first_name, 0x0a, last_name, 0x0a, user, 0x0a, password) FROM users # disisipkan ke dalam klausa WHERE sebagai nilai dari kolom username. Seperti sebelumnya, bagian 1 = 0 adalah pernyataan logika yang selalu salah. Query tambahan dalam union ini menggunakan pernyataan CONCAT untuk menggabungkan nama depan, nama belakang, username, dan password dari setiap baris dalam tabel 'users', dipisahkan oleh karakter newline dalam format heksadesimal (0x0a). Hal ini dimaksudkan untuk membuat hasil lebih terstruktur dan mudah dibaca.

Hasil dari query tambahan ini akan ditampilkan sebagai bagian dari hasil query asli, memberikan penyerang informasi sensitif tentang setiap pengguna dalam tabel 'users', termasuk nama depan, nama belakang, username, dan bahkan password.



Gambar 11. Hasil serangan SQL Injection manual 5

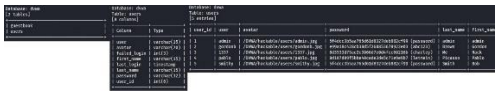
SQLMap Command

sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=123&Submit=submit" --cookie="9c9fjecekjgeefhvew; security:low" --tables

-u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=123&Submit=submit" adalah opsi yang digunakan untuk menyediakan URL target yang akan diuji oleh SQLMap. Dalam kasus ini, URL tersebut mengarah ke suatu aplikasi web yang berpotensi rentan terhadap SQL Injection. Parameter URL seperti id=123 adalah contoh parameter yang digunakan dalam permintaan GET.

--cookie="" adalah opsi yang digunakan untuk menyediakan informasi cookie yang diperlukan oleh aplikasi web. Cookies ini diperlukan untuk menjaga sesi dan autentikasi pada aplikasi.

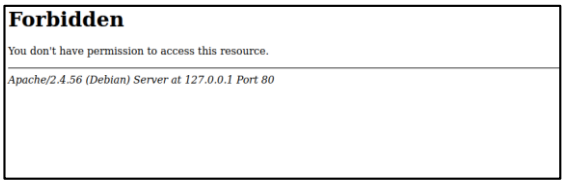
--tables adalah opsi yang meminta SQLMap untuk mencari dan menampilkan daftar tabel dalam basis data yang mungkin rentan terhadap serangan SQL Injection. Dengan menggunakan opsi ini, SQLMap akan mengambil informasi tentang tabel-tabel yang ada dalam basis data yang terhubung dengan aplikasi tersebut.



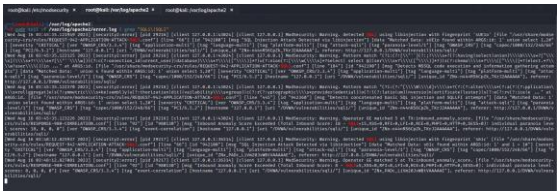
Gambar 12. Hasil serangan SQL Injection menggunakan SQLMap

4.7. Pengujian Serangan SQL Injection Dengan Keamanan WAF

Dalam melakukan serangan SQL injection ini, penulis melakukan serangan secara manual dan juga menggunakan tools yang bernama SQLMap. Pada pengujian kedua akan melakukan serangan SQL injection terhadap aplikasi web yang sudah menggunakan Web Application Firewall (WAF). Untuk pengujian SQL Injection secara manual query yang digunakan sama dengan pengujian pertama. Dan hasil dari serangan SQL injection gagal membobol database pada aplikasi web dan Web server berhasil mendeteksi serangan SQL Injection Seperti pada Gambar 13 dan Gambar 14.



Gambar 13. Hasil serangan SQL Injection pada website yang sudah di pasang WAF



Gambar 14. WAF mendeteksi dan memblokir Serangan SQL Injection

4.8. Pengujian Serangan Cross Site Scripting tanpa keamanan WAF

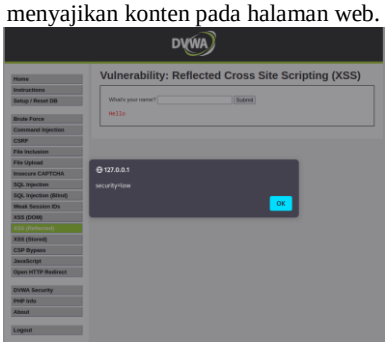
Dalam melakukan serangan Cross Site Scriptng (XSS) ini, penulis menggunakan tools yang bernama XSStrike. Pada pengujian pertama akan melakukan serangan XSS dengan menggunakan XSStrike terhadap aplikasi web yang belum menggunakan Web Application Firewall (WAF). Hasil dari serangan XSStrike berhasil menampilkan script yang dimasukan seperti yang dapat dilihat pada Tabel dan Gambar dibawah.

Tabel 3. Script XSS yang digunakan

XSS Script
<script>alert(document.cookie)</script>

Dalam Script diatas:
“<script>alert(document.cookie)</script>”

<script>: Ini adalah tag HTML yang digunakan untuk menyisipkan skrip JavaScript ke dalam halaman web. alert(document.cookie): Ini adalah skrip JavaScript yang akan memicu sebuah jendela peringatan yang menampilkan nilai dari cookie untuk domain yang sedang diakses. Cookie adalah informasi yang disimpan oleh browser dan dapat berisi data seperti sesi pengguna atau informasi otentikasi. Serangan XSS seperti ini dapat terjadi jika aplikasi web tidak memvalidasi atau menyaring masukan pengguna dengan benar sebelum menyajikan konten pada halaman web.



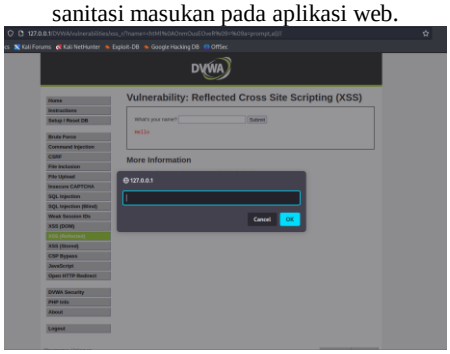
Gambar 15. Pengujian XSS 1

<a%09ONmOUSeOvEr%0a=%0a(prompt)``//v3dm0s

Pada Script Diatas:
"<a%09ONmOUSeOvEr%0a=%0a(prompt)```//v3dm0s
"

<a%09ONmOUSeOvEr%0a=%0a: Ini adalah atribut HTML yang dimanipulasi untuk menyembunyikan skrip XSS. %09 dan %0a adalah kode URL untuk karakter tab dan karakter baris baru yang mengaburkan atribut agar lebih sulit terbaca oleh mata manusia.

(prompt)```: Ini adalah skrip JavaScript yang mencoba mengeksekusi fungsi prompt(). Fungsi prompt()` adalah fungsi bawaan JavaScript yang digunakan untuk menampilkan kotak dialog berisi pesan dan input dari pengguna. Dalam konteks serangan XSS, penyerang mencoba memanfaatkan fungsi ini untuk memicu interaksi dengan pengguna dan mungkin mencuri informasi sensitif dari mereka. Serangan XSS semacam ini mencoba memanfaatkan celah dalam mekanisme sanitasi masukan pada aplikasi web.



Gambar 16. Pengujian XSS 2

<IMG """><SCRIPT>alert("XSS")</SCRIPT>"\>

Pada Script Diatas :
“<IMG """><SCRIPT>alert("XSS")</SCRIPT>"\>”

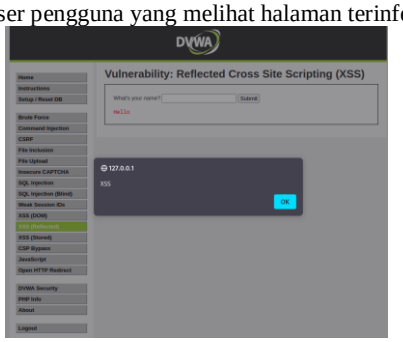
<IMG ": Ini adalah elemen HTML <img> yang dimulai, tetapi diikuti oleh tanda kutip ganda yang akan terpotong dan menyebabkan permasalahan sintaks.

">: Ini adalah kutipan ganda tambahan yang menutup atribut src yang seharusnya ada dalam elemen <img>, tetapi ini digunakan untuk mengakhiri atribut dan menambahkan ruang untuk menyisipkan skrip berbahaya.

<SCRIPT>alert("XSS")</SCRIPT>: Ini adalah skrip JavaScript yang dimasukkan di dalam elemen <img>. Skrip ini akan memicu jendela peringatan yang berisi pesan "XSS" saat halaman dimuat.

"\>: Ini adalah karakter akhir yang menutup tag elemen <img>.

Serangan XSS semacam ini mencoba memanfaatkan kelemahan dalam sanitasi masukan pada aplikasi web. Penyerang menyisipkan skrip berbahaya di dalam elemen HTML yang seharusnya hanya berisi gambar, dan mencoba memicu eksekusi skrip JavaScript pada browser pengguna yang melihat halaman terinfeksi.

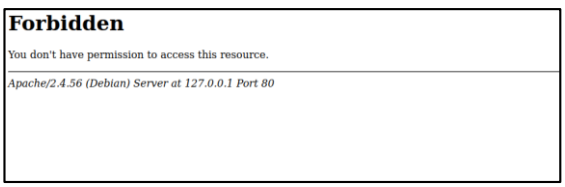


Gambar 17. Pengujian XSS 3

4.9. Pengujian Serangan Cross Site Scripting Dengan Keamanan WAF

Pada pengujian Kedua akan melakukan serangan Cross Site Scripting terhadap aplikasi web yang sudah menggunakan Web Application Firewall (WAF). Dan hasil dari serangan XSS Web server berhasil mendeteksi serangan dan memblokir payload yang dimasukan seperti pada Gambar 18 dan 19.

Gambar 18. Web Server berhasil mendeteksi dan memblokir serangan XSS



Gambar 19. Hasil serangan XSS pada Website yang dipasang WAF

5. KESIMPULAN DAN SARAN

Penelitian ini bertujuan untuk menguji efektivitas penggunaan Web Application Firewall (WAF) dalam melindungi aplikasi web dari serangan SQL Injection (SQLI) dan Cross-Site Scripting (XSS). Dalam penelitian ini, WAF telah diimplementasikan dan diuji secara menyeluruh untuk mengevaluasi kemampuannya dalam mendeteksi dan mencegah serangan-serangan tersebut. Hasil pengujian menunjukkan penggunaan WAF dapat meningkatkan keamanan aplikasi web. Penggunaan WAF telah berhasil dalam mendeteksi serangan SQL Injection yang ditujukan pada aplikasi web. Dengan menganalisis pola serangan yang mencurigakan, WAF dapat mencegah eksekusi perintah SQL berbahaya, mengurangi risiko kerentanannya, dan melindungi integritas database aplikasi. WAF juga telah berhasil mencegah serangan Cross-Site Scripting yang dapat mengancam kerahasiaan dan integritas pengguna aplikasi web. WAF mengidentifikasi dan menolak permintaan berbahaya yang mengandung skrip berbahaya dengan menggunakan aturan (rules) yang sudah di set, sehingga mengurangi potensi risiko XSS.

Penulis sudah menyajikan sebuah pendekatan yang dilakukan untuk pengamanan aplikasi web menggunakan Web Application Firewall (WAF). Peneliti berharap penelitian ini akan berkontribusi pada analisis yang lebih mendalam mengenai keamanan Web Application Firewall (WAF), penulis menyarankan untuk melakukan penetration testing dengan menggunakan payload yang lebih kompleks yang memungkinkan untuk melewati sistem keamanan WAF.

DAFTAR PUSTAKA

[1] M. Alenezi, M. Nadeem, A. Agrawal, R. Kumar, and R. A. Khan, "Fuzzy multi criteria decision analysis method for assessing security design tactics for web applications," *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 5, pp. 181–196, Oct. 2020, doi: 10.22266/ijies2020.1031.17.

[2] N Schonning, "HTTP request methods - HTTP | MDN," 2020. https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods (accessed Aug. 24, 2022).

[3] Bangkit Wiguna, Wahyu Adi Prabowo, and Ridho Ananda, "Implementasi Web Application Firewall dalam Mencegah Serangan SQL Injection pada Website," 2020, doi: 10.31849/digitalzone.v11i2.4867ICCS.

[4] V Dehalwar, A Kalam, M L Kolhe, and A Zayegh, *OWASP Top 10 - 2021 The Ten Most Critical Web Application Security Risks*. 2021.

[5] "SQL Injection | OWASP Foundation." https://owasp.org/www-community/attacks/SQL_Injection (accessed Jan. 15, 2023).

[6] "Cross Site Scripting (XSS) | OWASP Foundation." https://owasp.org/www-community/attacks/xss/ (accessed Jan. 15, 2023).

[7] Imperva, "The State of Vulnerabilities in 2020," 2020.

[8] Piyush A. Sonewar and Nalini A. Mhetre, "A Novel Approach for Detection of SQL Injection and Cross Site Scripting Attacks," 2015.

[9] H. J. Liao, C. H. Richard Lin, Y. C. Lin, and K. Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1. pp. 16–24, Jan. 2013. doi: 10.1016/j.jnca.2012.09.004.

[10] B. I. Mukhtar and M. A. Azer, "Evaluating the Modsecurity Web Application Firewall against SQL Injection Attacks," in *Proceedings of ICCES 2020 - 2020 15th International Conference on Computer Engineering and Systems*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020. doi: 10.1109/ICCES51560.2020.9334626.

[11] "Web Application Firewall を理解するための手引き A Handbook to Understand Web Application Firewall Web Application Firewall (WAF) Guide 2 nd Edition," 2011, Accessed: Dec. 07, 2022. [Online]. Available: http://www.ipa.go.jp/security/vuln/waf.html

[12] L. Ma, D. Zhao, Y. Gao, and C. Zhao, "Research on SQL Injection Attack and Prevention Technology Based on Web," in *Proceedings - 2nd International Conference on Computer Network, Electronic and Automation, ICCNEA 2019*, Institute of Electrical and Electronics Engineers Inc., Sep. 2019, pp. 176–179. doi: 10.1109/ICCNEA.2019.00042.

[13] V. Clincy and H. Shahriar, "Web Application Firewall: Network Security Models and Configuration," in *Proceedings - International Computer Software and Applications Conference*, IEEE Computer Society, Jun. 2018, pp. 835–836. doi: 10.1109/COMPSAC.2018.00144.

[14] "Core Rule Set Documentation :: Core Rule Set Documentation." https://coreruleset.org/docs/ (accessed Jan. 14, 2023).

[15] "OWASP ModSecurity Core Rule Set | OWASP Foundation." https://owasp.org/www-project-modsecurity-core-rule-set/ (accessed Dec. 06, 2022).

[16] martinhsv, "www.modsecurity.org." https://github.com/SpiderLabs/ModSecurity (accessed Aug. 11, 2023).

[17] IEEE Electrical Insulation Society Staff, *2013 Eleventh International Symposium on Autonomous Decentralized Systems*. IEEE, 2013.

[18] Shrivastava A, Choudhary S, and Kumar A, *XSS Vulnerability Assessment and Prevention in Web Application*. 2016.

[19] M. Denis, C. Zena, and T. Hayajneh, "Penetration Testing: Concepts, Attack Methods, and Defense Strategies." [Online]. Available: www.kali.org