SISTEM PENGAMANAN WEB SERVER DENGAN WEB APPLICATION FIREWALL (WAF)

I Made Suartana [1], Henni Endah Wahanani [2], Aditya Noor Sandy [3]
Jurusan Teknik Informatika, Fakultas Teknologi Industri, UPN "Veteran" Jawa Timur, Surabaya[1, 2, 3]
made.suartana84@gmail.com[1]

Abstrak. *Seiring dengan perkembangan jaringan internet dan semakin banyaknya pengguna aplikasi pada platform web, ancaman terhadap keamanan data dan informasi juga semakin meningkat. Berdasarkan laporan yang dibuat oleh IBM research development, serangan pada aplikasi web yang paling populer pada retang waktu tahun 2009 sampai 2013 adalah SQL Injection dan Cross Site Scripting (XSS) [5]. Untuk melindungi pengguna aplikasi web dan menjamin keamanan data dan transaksi pada jaringan internet maka perlu dikembangkan suatu mekanisme keamanan untuk mencegah serangan yang mungkin muncul. Penelitian ini membahas penggunaan Wireless Apllication Firewall (WAF) dengan menggunakan ModSecurity sebagai mekanisme untuk mencegah serangan SQL Injection dan XSS. Penggunaan WAF dilengkapi dengan mekanisme pelaporan dan audit terhadap serangan yang terjadi menggunakan Jwall Auditconsole. Berdasarkan hasil ujicoba terhadap skenario yang digunakan penerapan rule dari ModSecurity dapat mencegah serangan SQL Injection dan XSS dan Auditconsole dapat digunakan sebagai mekanisme pelaporan serangan, dan mengetahui tingkat bahaya dari serangan tersebut.*

Kata kunci*: Keamanan Web, Web Application Firewall (WAF), Modsecurity.*

Penggunaan jaringan *Internet Protocol* (IP) semakin meningkat dari tahun ke tahun. Alasan utama pesatnya perkembangan internet adalah kemampuan internet menyediakan layanan yang berguna dan disukai oleh milyaran pengguna[2]. Menurut data dari *internet-society* penggunaan internet mencapai lebih dari dua milyar orang di seluruh dunia. Lebih dari 42.3% orang dari seluruh populasi di dunia menggunakan internet. Prosentasenya pertumbuhan penggunaan internet 741% antara tahun 2000 sampai 2010[2].

Seiring dengan pesatnya penggunaan internet, penggunaan aplikasi berbasis web juga meningkat. Dengan banyaknya pengguna memanfaatkan aplikasi berbasis web termasuk untuk layanan yang penting dan berharga, menjadikan aplikasi web target serangan yang populer. Serangan pada aplikasi web juga diakibatkan oleh adanya kerentanan yang terdapat pada teknologi web [6]. Jenis ancaman terhadap aplikasi web sifatnya dinamis, kemungkinan muncul jenis serangan baru seiring perkembangan dari teknologi. Dari beberapa jenis serangan yang pernah dilaporkan atau terjadi akibat dari kelemahan sistem, serangan dengan *Sql Injection* dan XSS (*cross side scripting*) yang paling banyak dilaporkan sesuai laporan yang dibuat oleh IBM *research development*, pada retang waktu tahun 2009 sampai 2013 [5].

Untuk mengatasi permasalahan keamanan pada penggunaan Aplikasi Web dan meminimalisir kerugian yang ditimbulkan terutama berkaitan dengan serangan SQL Injection dan XSS, maka diperlukan suatu mekanisme untuk mengatasi serangan tersebut. Beberapa mekanisme keamanan sudah digunakan, seperti penggunaan framework untuk mengatur flow data pada aplikasi web [6].

Pada penelitian ini, membahas penggunaan *Wireless Application Firewall* (WAF) dengan menggunakan Modsecurity sebagai solusi untuk pengamanan *web server*. WAF adalah modul *open source* untuk *web server* Apache yang dikembangkan oleh SpiderLabs, Trustwave. WAF merupakan produk *open source*, dengan lisensi gratis yang memungkinkan dikembangkan oleh banyak pengguna [4]. ModSecurity dapat melindungi aplikasi web dari berbagai serangan dan memungkinkan pemantauan lalu lintas HTTP tanpa banyak melakukan perubahan pada infrastruktur yang ada [4].

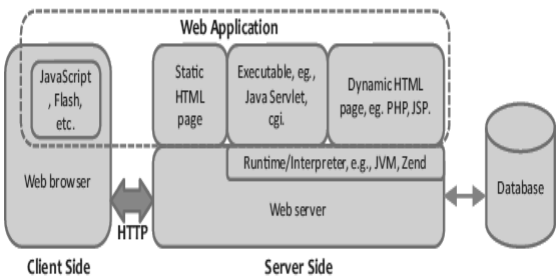
Manfaat dari Sistem yang akan di implementasikan adalah untuk menjaga validitas pada *web server* serta menjamin ketersediaan layanan bagi penggunanya dan melindungi *web server* dari serangan Sql Injection dan XSS serta memudahkan *administrato*r untuk mengatur dan memantau keamanan *web server* secara langsung tanpa

harus mengawasi langsung dari *computer host* atau *server*.

I. METODOLOGI

Aplikasi Web

Aplikasi Web adalah aplikasi *client-server* yang dijalankan di atas platform Web. Aplikasi *client-server* pada web merupakan bagian tidak terpisahkan yang memungkinkan informasi dan layanan yang bersifat dinamis tersampaikan. Aplikasi web terdiri dari beberapa komponen yang menjadi satu kesatuan Seperti ditunjukkan dalam Gambar 1



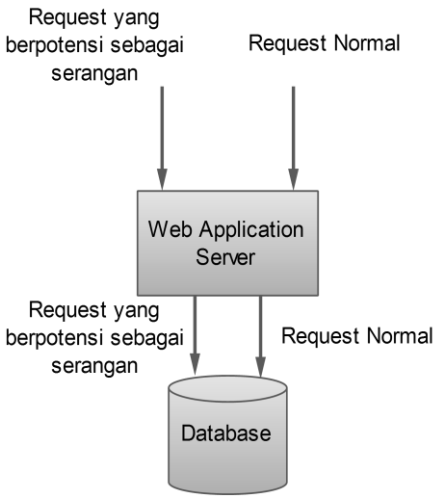
Gambar 1. Aplikasi Web

Aplikasi Web terdiri dari beberapa komponen yang menjadi satu kesatuan fungsi. Hal ini yang menyebabkan Aplikasi Web memiliki kerentanan yang mungkin menimbulkan masalah keamanan. Secara umum, ada tiga jenis kerentanan keamanan dalam aplikasi web pada tingkat yang berbeda: pertama kerentanan pada validasi *input* (tingkat *request* ke sistem), kerentanan manajemen sesi komunikasi (tingkat *session*), dan kerentanan pada tingkat aplikasi [6]. Pada penelitian ini membahas contoh serangan yang mungkin muncul akibat dari kerentanan pada tingkat *request* akibat validasi *input*, serangan yang memanfaatkan kerentanan tersebut adalah SQL Injection dan XSS.

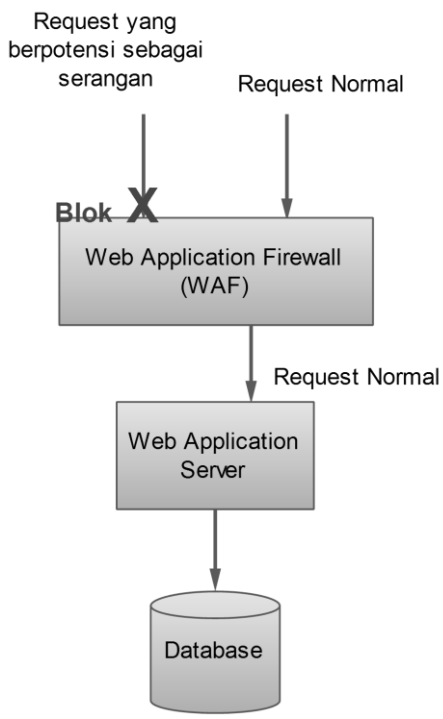
Web Application Firewall

Konsep di balik *Web Application Firewall* (WAF) sangat mirip dengan bagaimana *firewall* tradisional bekerja. *Firewall* bekerja berdasarkan suatu set aturan yang dikonfigurasi pada *firewall* atau yang disebut dengan *rule*. Aturan ini yang selektif mengizinkan atau menolak lalu lintas jaringan. Aturan pada WAF secara khusus dirancang untuk menyaring lalu lintas jaringan dengan menggunakan protokol HTTP. Aturan ini juga mampu mendeteksi serangan umum, seperti *probe* (upaya mendapatkan informasi awal sebelum melakukan serangan) dari serangan SQL Injection dan upaya XSS. *Firewall* bisa berupa perangkat lunak yang di-instal pada *host*, atau sebagai perangkat keras khusus. WAF adalah salah satu mekanisme pertahanan awal pada sistem.

Cara kerja WAF dilihat pada gambar 2 dan gambar 3, gambar 2 kondisi dimana WAF belum diaktifkan pada sistem. *Request* yang berpontensi melakukan serangan langsung bisa mengakses *web server* dan *database server*.



Gambar 2. Skema Web Server Tanpa WAF



Gambar 3. Skema Web Server dengan WAF

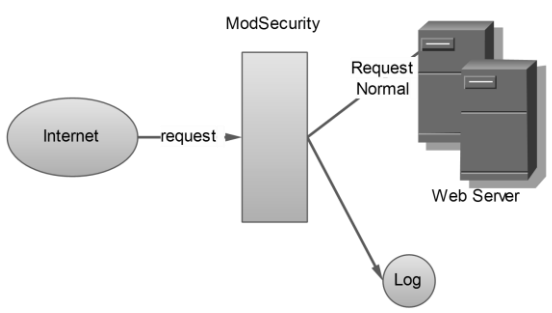
Pada gambar 3 lalu lintas yang sah diperbolehkan melalui WAF, namun lalu lintas pada jaringan yang berpotensi sebagai serangan dihentikan sebelum dapat mencapai *Web Application Server* dan *database server*.

Aplikasi Web *firewall* adalah perlindungan dari akses yang tidak sah di internet. Aplikasi *web firewall* melakukan penyaringan *request* HTTP dan mencegah akses yang tidak sah.

ModSecurity adalah WAF yang bersifat *open source* yang merupakan modul tambahan pada Apache. Beberapa fitur mod_security adalah pemeriksaan log, akses ke setiap bagian dari *request* yang ditujukan ke server (termasuk isi *request* atau *body*) dan memberikan respon terhadap hasil pemeriksaan, memiliki *rule* yang berdasarkan aturan *regular expression* yang fleksibel, pemeriksaan berkas yang diunggah, validasi *real-time* dan juga perlindungan *buffer-overflow*.

Fungsi modul dalam ModSecurity dibagi menjadi empat bidang utama:

1. *Parsing*: parser akan melakukan ekstraksi bit setiap *request* dan atau *respond* dan menyimpannya untuk digunakan dalam *rule*.
2. *Buffering*: bertindak sebagai *buffer* baik bagi *request* dan *respond*, *body* dari *request* akan di buffer sehingga modul biasanya melihat isi *request* secara lengkap (sebelum dilewatkan ke aplikasi untuk pengolahan), dan tanggapan lengkap (sebelum dikirim ke *client*). Buffering sangat penting, karena merupakan satu-satunya cara untuk memberikan blocking handal.
3. *Logging*: Berfungsi untuk merekam lalu lintas HTTP secara lengkap pada jaringan, menyimpan dalam bentuk log semua *request* atau *respond* *header* dan *body* pesan.
4. *Rule engine*: Bekerja dengan data dari komponen lainnya, untuk menilai transaksi dan mengambil tindakan, yang diperlukan.



Gambar 4. Alur ModSecurity

mod_security dapat digunakan dalam dua mode: *Embedded mode* (Modus tertanam): dengan menambahkan mod_security sebagai modul ke Server Apache. Namun, dalam mode ini, tidak dapat memeriksa isi *header* Server. *Network Gateway* (mode Gateway): Dalam mode ini semua lalu lintas web melewati *proxy*, mod_security diinstal sebagai *reverse proxy* gambar 4.

Mekanisme Sql Injection

Pada dasarnya SQL Injection merupakan cara mengeksploitasi celah keamanan yang muncul pada level atau "layer" *database* dan aplikasinya. Celah keamanan tersebut ditunjukan berupa *respond* dari *request* data yang salah ke *server* yang sengaja dikirim oleh penyerang.

Contoh-contoh celah kerentanan yang kerap menjadi korban SQL Injection adalah:

1. Karakter-karakter kendali, kontrol, atau filter tidak didefinisikan dengan baik dan benar (*Incorrectly Filtered Escape Characters*),
2. Tipe pemilihan dan penanganan variabel maupun parameter program yang keliru (*Incorrect Type Handling*),
3. Celah keamanan berada dalam *database server* (*Vulnerabilities Inside the Database Server*)
4. Dilakukan mekanisme penyamaran SQL Injection (*Blind SQL Injection*); dan lain sebagainya.

Salah satu teknik SQL Injection Blind Sql Injection merupakan teknik serangan yang dilakukan dengan memasukan sintaks atau perintah Sql pada sebuah web yang memiliki kerentanan terhadap serangan, untuk mengetahui isi dari *database* dari web tersebut. Berikut contoh skrip program serangan SQL injection.

```
localhost/index.php?id=-9union
select
1,group_concat(table_name),3,4,5
command from
information_schema.tables where
table_schema=database()--
```

Gambar 5. Contoh Skrip SQL Injection

Hasil dari skrip ketika dijalankan pada aplikasi web yang memiliki kerentanan terhadap serangan SQL Injection dapat dilihat pada gambar 5.

Gambar 6. Hasil serangan Sql Injection

Mekanisme XSS

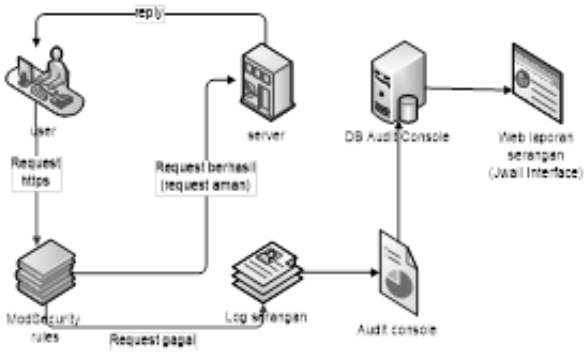
Cross Site Scripting (XSS) adalah suatu serangan dengan menggunakan mekanisme "*injection*" pada aplikasi web dengan memanfaatkan metode HTTP GET atau HTTP POST. XSS biasa digunakan oleh pihak-pihak yang berniat tidak baik dalam upaya mengacaukan konten website dengan memasukkan naskah program (biasanya java script) sebagai bagian dari teks *input* melalui mekanisme inputan yang tersedia [7].

Ada tiga jenis serangan XSS berdasarkan bagaimana script serangan diinjeksi yaitu *Non-persistent*, *Persistent* dan *DOM-base* [8]. Non-persistent atau *Reflected* XSS dijalankan ketika korban mengklik link web dibuat, yang akan mengirimkan kembali data XSS ke pengirim melalui aplikasi web. *Persisten*t XSS terjadi ketika *script* serangan yang dikirim ke *database* aplikasi *back-end*, misalnya, posting forum, komentar, dan lain-lain, dan disimpan untuk jangka waktu tertentu. Script berbahaya dipicu oleh korban ketika pengguna mengunjungi sebuah halaman web yang berisi script serangan tersebut. DOM berbasis XSS terjadi ketika script berbahaya yang dimasukkan ke dalam sisi *client* dengan kode JavaScript untuk eksekusi, bahkan tanpa mengirim ke sisi *server*.

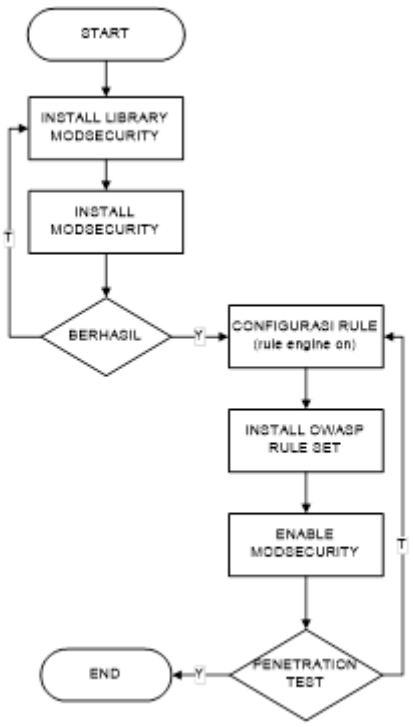
II. HASIL DAN PEMBAHASAN

Dalam penelitian ini akan dibuat sistem pengamanan dan pelaporan *Intrusi* dari serangan *Web Server* menggunakan Modsecurity. *Intrusi* yang di deteksi dan dilaporkan adalah serangan Sql Injection dan *Cross Site Scripting*. Dalam penelitian ini akan di jelaskan metode pengamanan aplikasi web menggunakan Modsecurity dan pelaporan serangan menggunakan Jwall Auditconsole terhadap serangan Sql Injection dan XSS. serta dijelakan tentang skenario penyerangan dan pengujian sistem dengan parameter-parameter yang sudah ditentukan.

Mekanisme pengamanan server dengan menggunakan WAF dapat dilihat pada ilustrasi pada gambar 7, semua *request* HTTP yang ditujukan ke *server* akan melewati *firewall*, request yang diteruskan ke server hanya *request* yang dianggap tidak membahayakan *server* dan *request* yang berpotensi sebagai serangan tidak diteruskan ke *server* dan dicatat pada log serangan dan kemudian dianalisa dengan Auditconsole hasil analisa dilaporkan dengan aplikasi Jwall.



Gambar 7. Skema Implemetasi



Gambar 8. Implementasi ModSecurity

Implementasi ModSecurity untuk SQL Injection dan XSS ditujukan pada gambar 8. Modsecurity akan memeriksa dan melakukan *filter request* yang datang. Pemeriksaan ini termasuk *ip address user*, *user agent*, jenis file yang di *request*, serta inputan yang diinputkan pada *Web Server*. *Request* tersebut kembali diperiksa apakah objek yang diminta dianggap berbahaya atau tidak oleh *firewall* atau *security* yang ada. Jika *request* yang di minta dianggap membahayakan maka akan ditolak dan semua informasi mengenai *user* akan disimpan pada log file yang telah di sediakan dan log file akan diproses dengan Audit Console untuk menata laporan log serangan,

setelah itu laporan serangan akan di dimasukan ke dalam database serangan untuk ditampilkan oleh web interface serangan. Tetapi jika *request* tersebut dianggap tidak membahayakan maka akan diijinkan untuk mengaksess halaman yang di minta dalam format *reply* http.

Ujicoba dilakukan dengan menggunakan dua skenario skenario pertama dilakukan serangan SQL Injection dan XSS pada Aplikasi Web sebelum ModSecurity diaktifkan, sedangkan pada skenario kedua dilakukan serangan SQL Injection dan XSS pada Aplikasi Web setelah ModSecurity diaktifkan.

Tabel 1. Hasil Uji Coba Dengan Serangan SQL Injection

No	Script Serangan	Skenario 1	Skenario 2	Alert
1.	localhost/index.php?id=-9'	Berhasil	Gagal	Sql Injection
2.	localhost/index.php?id=-9 order by 1,2,3,4,5,6--	Berhasil	Gagal	Sql Injection
3.	localhost/index.php?id=-9 union select 1,database(),user(),4,5--	Berhasil	Gagal	Sql Injection
4.	localhost/index.php?id=-9 union select 1,concat(version(),0x3a,database(),0x3a,user()),3,4,5--	Berhasil	Gagal	Sql Injection
5.	localhost/index.php?id=-9 union select 1,group_concat(column_name),3,4,5 from information_schema.columns where table_name='table_user' limit 0,1--	Berhasil	Gagal	Sql Injection
5.	localhost/index.php?id=-9 union select 1, concat(user_id,0x3a,password_id),3,4,5 from table_user--	Berhasil	Gagal	Sql Injection

Table 1. merupakan hasil dari uji coba yang dilakukan menggunakan teknik serangan SQL Injection. Dari table 1 dapat dilihat bahwa sistem pengamanan dan pelaporan intrusi yang terpasang pada web server dapat mendeteksi, menangkal, dan melaporkan serangan Sql Injection.

Table 2 merupakan hasil dari uji coba dengan menggunakan teknik serangan XSS. Dari table 2 bisa dilihat bahwa sistem pengamanan dan pelaporan intrusi yang terpasang pada web server dapat mendeteksi, menangkal, dan melaporkan serangan XSS.

Tabel 2. Hasil uji coba serangan XSS

No	Script Serangan	Skenario 1	Skenario 2	Alert
1	<script>alert('XSS')</script>	Berhasil	Gagal	XSS
2	<script>window.open("http://localhost/log/index.php")</script>	Berhasil	Gagal	XSS
3	'><marquee> XSS </marquee>	Berhasil	Gagal	XSS
4	<iframe<?php echo chr(11)?> onload=alert('XSS')></iframe>	Berhasil	Gagal	XSS
5	<script src="http://www.evilsite.org/cookiegrabber.php"></script>	Berhasil	Gagal	XSS

Gambar 6.Hasil Audit data serangan

Pada hasil analisa log dari modsecurity gambar 6 akan didapatkan hasil dari data informasi log file modsecurity berupa event ID , Date, Sensor, Site, Received, Severity, Tags, Source, Destination, Message.

1. Event ID : menampilkan ID log serangan yang menampung
2. *Date* : menampilkan informasi waktu serangan.
3. Sensor : menampilkan informasi. sensor yang digunakan
4. *Received* : menampilkan informasi waktu dari Jwall Auditconsole
5. *Severity* : adalah tingkat bahaya dari serangan tersebut,
6. *Source* : menampilkan informasi *attacker* berupa IP, *port*, *country*
7. *Destination* : menampilkan direktori atau *database* tujuan dari *attacker*
8. *Message* : menampilkan pesan jenis serangan dari *rule* Modsecurity

III. SIMPULAN

Wireless Application Framework dengan menggunakan ModSecurity dapat digunakan untuk melindungi pengguna aplikasi web dari serangan terhadap keamanan. WAF terutama melindungi dari kerentanan aplikasi web. Kerentanan pada aplikasi web seperti XSS (*Cross Site Scripting*), dan SQL Injection. Kerentanan pada aplikasi web ini yang sering dimanfaatkan oleh pihak yang tidak bertanggung jawab untuk menyerang situs resmi dan membajak kode resmi. Jadi WAF digunakan untuk mencegah kerentanan pada aplikasi web.

WAF melakukan mekanisme pemindaian situs dan kode-kode yang dapat membajak, atau kode berbahaya dan melakukan penolakan terhadap kode-kode tersebut. Mekanisme ini dapat mencegah serangan ke *server* dalam bentuk request HTTP sesuai dengan *rule*. WAF bekerja berdasarkan *rule*, jadi WAF dapat mencegah hanya serangan-serangan yang berbahaya sesuai dengan aturan yang ditetapkan.

IV. DAFTAR PUSTAKA

[1] Gonzalo, C. Miguel A. dan Garc´ıa-Mart´., 2006, *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds*, Second Edition, John Wiley & Sons, Inc., New York.

[2] Internet usage statistics The Internet Big Picture 2014, *World Internet Users and Population Stats*, Miniwatts Marketing Group. http://www.internetworldstats.com/stats.html

[3] Kim Issac Museong, 2011, Using Web Application Firewall to detect and block common web application attacks, The SANS Institut

[4] Owasp, (2011) . Web application firewall. Retrieved from https://www.owasp.org/index.php/Web_Application_Firewall

[5] Web Application Vulnerabilities by Attack Technique, IBM X-Force Research and Development, http://securityintelligence.com/cyber-attacks-research-reveals-top-tactics-xforce/#.VLjRTck2ctk

[6] Xiaowei Li., Yuan Xue., 2013, A Survey on Server-side Approaches to Securing Web Applications, ACM Transactions on Computing Surveys.

[7] Indrajit, R. E, Aneka Ragam Serangan di Dunia Maya, Artikel IDISIRTII.