TINDAKAN PENCEGAHAN ANCAMAN WEB MELALUI PENDEKATAN TEKNIK *WEB APPLICATION FIREWALL* MENGGUNAKAN MODSECURITY

SKRIPSI

DICKY IKBAL PRATAMA

H071191060

PROGRAM STUDI SISTEM INFORMASI

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN

UNIVERSITAS HASANUDDIN

MAKASSAR

FEBRUARI 2024

TINDAKAN PENCEGAHAN ANCAMAN WEB MELALUI PENDEKATAN TEKNIK *WEB APPLICATION FIREWALL* MENGGUNAKAN MODSECURITY

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi Sistem Informasi Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin Makassar

DICKY IKBAL PRATAMA

H071191060

PROGRAM STUDI SISTEM INFORMASI

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN

UNIVERSITAS HASANUDDIN

MAKASSAR

FEBRUARI 2024

HALAMAN PERNYATAAN KEOTENTIKAN

Yang bertanda tangan di bawah ini:

Nama	:	DICKY IKBAL PRATAMA
NIM	:	H071191060
Program Studi	:	Sistem Informasi
Jenjang	:	S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul

TINDAKAN PENCEGAHAN ANCAMAN WEB MELALUI PENDEKATAN TEKNIK *WEB APPLICATION FIREWALL* MENGGUNAKAN MODSECURITY

Adalah benar hasil karya saya sendiri bukan merupakan pengambilan alihan tulisan orang lain dan belum pernah dipublikasikan dalam bentuk apapun.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini merupakan hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Makassar, 04 Maret 2024

DICKY IKBAL PRATAMA

NIM. H071191060

TINDAKAN PENCEGAHAN ANCAMAN WEB MELALUI PENDEKATAN TEKNIK *WEB APPLICATION FIREWALL* MENGGUNAKAN MODSECURITY

Disusun dan diajukan oleh:

DICKY IKBAL PRATAMA

H071191060

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Sarjana pada Program Studi Sistem Informasi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama	Pembimbing Pertama
Dr. Hendra, S.Si., M.Kom.	Muhammad Sadno, S.Si., M.Si.
NIP. 197601022003121001	NIP. 199008162022043001

Ketua Program Studi

Dr. Khaeruddin, M.Sc
NIP. 196509141991031003

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh:

Nama	:	DICKY IKBAL PRATAMA
NIM	:	H071191060
Program Studi	:	Sistem Informasi
Judul Skripsi	:	TINDAKAN PENCEGAHAN ANCAMAN WEB MELALUI PENDEKATAN TEKNIK *WEB APPLICATION FIREWALL* MENGGUNAKAN MODSECURITY

Telah dipertahankan di depan Tim Penguji dan diterima sebagai bagian persyaratan untuk memperoleh gelar Sarjana pada Program Studi Sistem Informasi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.

DEWAN PENGUJI

			Tanda Tangan
1.	Ketua	: Dr. Hendra, S.Si., M.Kom.	(.....)
2.	Sekretaris	: Muhammad Sadno, S.Si., M.Si.	(.....)
3.	Anggota	: Dr. Andi Muhammad Anwar, S.Si., M.Si.	(.....)
4.	Anggota	: Jeriko Gormantara, S.Si., M.Si.	(.....)

Ditetapkan di : Makassar

Tanggal : 04 Maret 2024

KATA PENGANTAR

Puji syukur penulis panjatkan kepada **Allah *Subhanahu wa ta'ala***, karena atas berkat dan rahmat-Nya penulis dapat menyelesaikan skripsi ini. Shalawat serta salam senantiasa tercurah kepada ***Rasulullah* Muhammad *Shalallahu Alaihi Wassalam,*** yang merupakan teladan dalam menjalankan kehidupan dunia.

Alhamdulillah, skripsi dengan judul TINDAKAN PENCEGAHAN ANCAMAN WEB MELALUI PENDEKATAN TEKNIK *WEB APPLICATION FIREWALL* MENGGUNAKAN MODSECURITY yang disusun sebagai salah satu syarat untuk mencapai gelar Sarjana pada program studi Sistem Informasi fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin ini dapat diselesaikan. Walaupun adanya kendala-kendala yang dihadapi khususnya wabah Covid-19 ketika skripsi ini dikerjakan. Tetapi dalam penulisan skripsi ini, penulis mampu menyelesaikan pada waktu yang tepat berkat bantuan dan dukungan dari berbagai pihak.

Ucapan terima kasih dan apresiasi yang tak terhingga kepada kedua orang tua penulis Bapak **Iqbal** dan Ibu **Nurdiana** yang tak kenal lelah dalam memanjatkan doa serta memberikan nasihat dan motivasi kepada penulis. Tak lupa juga kepada saudara-saudara penulis **Dirga Iqbal Prayuda**, **Didit Iqbal Alfrauzi**, **Dinda Iqbal Pratiwi**, yang selalu menjadi motivasi bagi penulis untuk terus melangkah maju.

Penulis menyadari bahwa skripsi ini dapat terselesaikan dengan adanya bantuan, bimbingan, dukungan dan motivasi dari berbagai pihak. Oleh karena itu, penulis mengucapkan ucapan **Terima Kasih** dengan tulus kepada:

- Rektor Universitas Hasanuddin, Bapak **Prof. Dr. Ir. Jamaluddin Jompa, M.Sc.** beserta jajarannya.
- Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, **Dr. Eng. Amiruddin, S. Si., M. Si.** beserta jajarannya.
- Ketua Departemen Matematika FMIPA, **Dr. Firman, S.Si., M.Si.** dan juga **Dr. Khaeruddin, M.Sc.,** sebagai ketua Program Studi Sistem Informasi Universitas Hasanuddin.

- Bapak **Dr. Hendra, S.Si., M.Kom.** sebagai pembimbing utama yang telah banyak memberikan arahan, ide, motivavsi serta dukungan kepada penulis dalam banyak hal.

- Bapak **Muhammad Sadno, S.Si., M.Si.** sebagi pembimbing pendamping yang senantiasa, memberikan arahan, ide, motivavsi serta dukungan kepada penulis dalam banyak hal.

- Bapak **Dr. Andi Muhammad Anwar, S.Si., M.Si** dan bapak **Jeriko Gormantara, S.Si., M.Si.** sebagai tim penguji atas saran dan masukan pada penelitian yang telah dilakukan oleh penulis.

- Seluruh Bapak dan Ibu dosen FMIPA Universitas Hasanuddin yang telah mendidik dan memberikan ilmunya sehingga penulis mampu menyelesaikan program sarjana. Serta para staf yang telah membantu dalam pengurusan berkas administrasi.

- Bapak **Dr. phil. Christian Folini** sebagai Co-Lead proyek OWASP ModSecurity *Core Rule Set* (CRS) yang telah memberikan masukan, ide serta dukungan kepada penulis dalam banyak hal terumatama dalam implementasi OWASP ModSecurity CRS.

- Saudara-saudara **KEPOMPONG SQUAD (Dani, Wawan, Ardi, Andika, Eca, Pallu, Eldin, Fadli, Alfath, Veri)** yang telah menemani penulis selama perkuliahan, saling memberi motivasi dan bantuan, meluangkan waktu dan berbagi suka-duka serta kebersamaan selama menuntut ilmu.

- Keluarga besar **Sistem Informasi Unhas 2019** yang setia menemani dan membatu penulis selama menjalani pendidikan. Serta kakak-kakak dan adik-adik **Sistem Informasi** yang telah banyak membantu, semoga tetap semangat dalam mengejar impian.

- Ibu Dewi Shinta yang setia menemani dan membatu penulis selama menjalani pendidikan.

- Serta semua pihak yang telah banyak berpartisipasi, baik secara langsung maupun tidak langsung dalam penyusunan skripsi ini yang tidak sempat penulis sebutkan satu per satu.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna dikarenakan terbatasnya pengalaman dan pengetahuan yang dimiliki penulis. Oleh karena itu, penulis mengharapkan segala bentuk saran serta masukan bahkan kritik yang membangun dari berbagai pihak. Semoga tulisan ini memberikan manfaat kepada semua pihak yang membutuhkan dan terutama untuk penulis.

Makassar, 04 Maret 2024

DICKY IKBAL PRATAMA

NIM. H071191060

PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai civitas akademik Universitas Hasanuddin, saya yang bertanda tangan di bawah ini:

Nama	:	DICKY IKBAL PRATAMA
NIM	:	H071191060
Program Studi	:	Sistem Informasi
Departemen	:	Matematika
Fakultas	:	Matematika dan Ilmu Pengetahuan Alam
Jenis Karya	:	Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Hasanuddin **Hak Predikator Royalti Noneksklusif** (*Non-exclusive Royalty-Free Right*) atas tugas akhir saya yang berjudul:

"TINDAKAN PENCEGAHAN ANCAMAN WEB MELALUI PENDEKATAN TEKNIK *WEB APPLICATION FIREWALL* MENGGUNAKAN MODSECURITY"

beserta perangkat yang ada (jika diperlukan). Terkait dengan hal diatas, maka pihak Universitas Hasanuddin berhak menyimpan, mengalih-media/format-kan, mengola dalam bentuk pangkalan data (*database*), merawat dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di Makassar pada 04 Maret 2024

Yang menyatakan

(DICKY IKBAL PRATAMA)

ABSTRAK

TINDAKAN PENCEGAHAN ANCAMAN WEB MELALUI PENDEKATAN TEKNIK *WEB APPLICATION FIREWALL* MENGGUNAKAN MODSECURITY

DICKY IKBAL PRATAMA
H071191060

Penelitian ini menyoroti peran kritis keamanan aplikasi web di era digital, ketika ancaman terhadap kerentanan aplikasi dapat mengancam integritas dan ketersediaan data. Fokus utama penelitian ini adalah pencegahan ancaman web menggunakan *Web Application Firewall* (WAF), dengan penekanan khusus pada implementasi ModSecurity dan aturan dari proyek OWASP ModSecurity *Core Rule Set* (CRS). OWASP ModSecurity CRS memiliki level paranoia (PL) yang mengontrol tingkat keparahan aturan, dengan PL yang lebih tinggi menandakan aturan yang lebih luas dan potensi false positive (FP). Pengujian efektivitas ModSecurity dan aturan OWASP ModSecurity CRS dilakukan menggunakan Damn Vulnerable Web Application (DVWA) dengan level kerentanan *low*, *medium*, dan *high*. Hasil penelitian menunjukkan bahwa pada PL 1 dan 2, ModSecurity berhasil memblokir 65% dari 48 percobaan serangan pada DVWA. Beberapa serangan, seperti Brute Force, tidak dapat diblokir karena aturan yang terkait telah dihapus dalam CRS v4. Peningkatan level paranoia 3 dan 4 mencapai 100% pemblokiran, namun dengan *false positive* pada permintaan yang sah. Penelitian ini memberikan kontribusi signifikan terhadap pemahaman dan implementasi pencegahan ancaman web melalui penerapan WAF, khususnya ModSecurity dengan aturan OWASP ModSecurity CRS.

Kata Kunci : *Web Application Firewall* (WAF), ModSecurity, OWASP ModSecurity CRS, DVWA, Level Paranoia (FP).

ABSTRACT

WEB THREAT PREVENTION MEASURES THROUGH WEB APPLICATION FIREWALL TECHNIQUE USING MODSECURITY

DICKY IKBAL PRAMATA
H071191060

This research highlights the critical role of web application security in the digital age, when threats to application vulnerabilities can threaten data integrity and availability. The main focus of this research is web threat prevention using a Web Application Firewall (WAF), with particular emphasis on the ModSecurity implementation and rules from the OWASP ModSecurity Core Rule Set (CRS) project. The OWASP ModSecurity CRS has a paranoia level (PL) that controls the severity of the rules, with higher PLs signaling more extensive rules and potential false positives (FPs). Testing the effectiveness of ModSecurity and OWASP ModSecurity CRS rules was conducted using Damn Vulnerable Web Application (DVWA) with low, medium, and high vulnerability levels. The results showed that in PL 1 and 2, ModSecurity successfully blocked 65% of 48 attack attempts on DVWA. Some attacks, such as Brute Force, could not be blocked because the corresponding rules have been removed in CRS v4. Increased paranoia levels 3 and 4 achieved 100% blocking, but with false positives on legitimate requests. This research makes a significant contribution to the understanding and implementation of web threat prevention through the application of WAF, specifically ModSecurity with OWASP ModSecurity CRS rules.

Keywords : *Web Application Firewall (WAF), ModSecurity, OWASP ModSecurity CRS, DVWA, Paranoia Level(FP).*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi dan internet telah memberikan dampak yang signifikan terhadap cara kita berkomunikasi, bertransaksi, dan mengakses informasi. Evolusi teknologi internet dan web, dikombinasikan dengan konektivitas internet yang meningkat pesat, jumlah pengguna internet terus meningkat secara signifikan dari tahun ke tahun. Menurut data terbaru, lebih dari 4,8 miliar orang di seluruh dunia telah menggunakan internet pada tahun 2021, telah menyebabkan munculnya lanskap bisnis baru pada aplikasi web. Kemajuan dalam teknologi mobile dan perangkat seluler telah memungkinkan akses internet dari hampir mana saja dan kapan saja. Pengguna dapat mengakses aplikasi web melalui *smartphone*, tablet, dan perangkat *wearable*, memberikan fleksibilitas dalam berkomunikasi.

Seiring dengan pertumbuhan penggunaan aplikasi web, kekhawatiran tentang ancaman keamanan aplikasi web juga meningkat. Berdasarkan analisis (SiteLock) melaporkan bahwa situs web saat ini mengalami rata-rata 94% serangan setiap hari, dan dikunjungi oleh bot sekitar 2.608 kali seminggu. Bot adalah aplikasi perangkat lunak yang menjalankan tugas otomatis di internet dan digunakan oleh pelaku ancaman dunia maya atau Threat Actors untuk memindai situs web dari kerentanan dan menjalankan pola serangan dasar, dengan begitu banyak bot yang ditugaskan untuk mengungkap kerentanan sistem aplikasi web. Aplikasi web menimbulkan sejumlah masalah keamanan yang berasal dari pengkodean yang tidak aman, konfigurasi database yang tidak tepat dan lainnya, yang menimbulkan kelemahan atau kerentanan yang serius yang memungkinkan pelaku ancaman melakukan serangan untuk mendapatkan akses langsung pada database untuk meretas data sensitif yang berisi informasi berharga (misalnya data pribadi dan detail keuangan) (SiteLock, 2022). Terdapat beberapa serangan berbasis web saat ini seperti *SQL Injection* yang bentuk seranganya menyuntikkan perintah SQL yang berbahaya, terdapat juga XSS (*Cross-Site Scripting*) menyuntikkan kode berbahaya menggunakan input pengguna dari aplikasi web yang rentan untuk mengelabui

pengguna dan mengarahkan mereka ke situs *phising*, dan ada banyak serangan umum lainnya seperti *Directory Traversal*, *local file inclusion*, DOS (*Denial-of-Service*), dan lainnya.

Ada berbagai metode dan pendekatan yang dapat digunakan untuk menganalisis dan mengimplementasikan pencegahan ancaman web. Salah satunya yang digunakan dalam penelitian ini yaitu dengan menggunakan pendekatan teknik *Web Application Firewall* (WAF). WAF adalah teknologi keamanan yang dirancang untuk melindungi aplikasi web dari berbagai serangan berbasis web lainnya. WAF beroperasi dengan memantau dan menganalisis lalu lintas HTTP antara aplikasi web dan klien. WAF memeriksa semua permintaan dan tanggapan yang masuk untuk mengidentifikasi dan memblokir lalu lintas berbahaya. WAF juga memiliki dua jenis yaitu ada yang komersil dan *open-source*, untuk yang komersil seperti AppTrana WAF, Cloudflare WAF, AWS WAF dan lainnya. Sedangkan untuk yang *open-source* seperti ModSecurity, NAXSI, WebKnight, Shadow Daemon dan lainnya. Dalam penelitian ini menggunakan WAF yang *open-source* dari ModSecurity, pemilihan ModSecurity sebagai WAF open-source karna memiliki kumpulan aturan yang kuat yang terus diperbarui oleh komunitas dalam hal pencegahan yaitu OWASP ModSecurity CRS dan ModSecurity juga hemat biaya dibandingkan dengan WAF komersil dalam hal penggunaan.

Sederhananya ModSecurity merupakan *gatekeepers* atau penjaga gerbang untuk aplikasi web, dengan melakukan pencegahan terkait serangan berbasis web, terdapat banyak serangan berbasis web, maka dari itu dalam penelitian ini melakukan pengujian efektivitas ModSecurity pada beberapa serangan berbasis web dan seperangkat alat serangan untuk pengujian. Adapun serangannya terdapat dalam DVWA (Damn Vulnerable Web Application) yang merupakan aplikasi web yang sengaja dibuat rentan dan digunakan untuk pelatihan dalam hal pengujian keamanan, skill dan alat. Terdapat beberapa serangan berbasis web dalam DVWA seperti *SQL Injection*, XSS, LFI, *Brute Force* dan lainnya tergantung versi apa yang digunakan. Efektivitas ModSecurity sebagai WAF *open-source* dilihat dari seberapa baik ModSecurity dalam hal pencegahan serangan berbasis web. ModSecurity memerlukan seperangkat aturan untuk memaksimalkan pencegahan terhadap

serangan berbasis web maka dari itu pada penelitian ini, menggunakan OWASP ModSecurity CRS yang merupakan sekumpulan aturan untuk *Web Application Firewall*. OWASP Core Rule Set (CRS) adalah seperangkat aturan keamanan yang digunakan bersama dengan ModSecurity, untuk melindungi aplikasi web dari berbagai serangan siber. OWASP CRS menjadi panduan untuk para profesional keamanan yang berusaha memitigasi risiko keamanan pada aplikasi web. OWASP ModSecurity CRS, ketika diintegrasikan dengan ModSecurity, menyediakan solusi keamanan web yang efektif dan dapat disesuaikan. Dengan fokus pada pencegahan serangan dan perlindungan terhadap kerentanan umum, OWASP ModSecurity CRS menjadi alat yang sangat berharga dalam upaya melindungi aplikasi web dari ancaman siber yang terus berkembang (Folini dan Ristic, 2017).

Pengujian ModSecurity sebagai WAF *open-source* telah banyak dilakukan, yang membedakan penelitian ini dari yang lain, yaitu terkait dengan bentuk serangan dan pengaturan tingkat *paranoia* yang berbeda pada OWASP ModSecurity CRS, contohnya pada peneilitian yang berjudul "*Evaluating the Modsecurity Web Application Firewall Against SQL Injection Attacks*" pada penelitian tersebut pengujian ModSecurity dilakukan dengan serangan *SQL Injection* dan menggunakan alat serangan Sqlmap (Mukhtar dan Azer, 2020). Terdapat juga penelitian yang pengujiannya berfokus pada *paranoia level* yang ada pada OWASP ModSecurity CRS yang berjudul "*Impact of Paranoia Levels on the Effectiveness of the ModSecurity Web Application Firewall*" (Singh dkk., 2018).

1.2 Rumusan Masalah

Peningkatan keamanan aplikasi web menjadi suatu kebutuhan mendesak di era digital saat ini, ancaman terhadap kerentanan aplikasi web dapat menimbulkan dampak serius terhadap integritas dan ketersediaan data. Oleh karena itu, rumusan masalah penelitian ini yaitu:

Bagaimana efektivitas ModSecurity *Web Application Firewall* (WAF) dengan aturan OWASP ModSecurity CRS dalam pencegahan ancaman web?

1.3 Batasan Masalah

Untuk menghindari terjadinya pembahasan diluar dari pokok permasalahan yang ada, maka penulis menetapkan batasan masalah yang akan di bahas untuk penelitian ini yaitu:

Efektivitas ModSecurity *Web Application Firewall* (WAF) dengan aturan ModSecurity CRS diuji dengan serangan yang ada pada kerentanan *Damn Vulnerable Web Application* (DVWA) pada level (*Low,Medium,High*) dan juga menggunakan alat serangan Burp Suite, Nikto dan Sqlmap.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah, maka tujuan yang ingin dicapai dari penelitian ini adalah :

Mengetahui efektivitas dari implementasi ModSecurity *Web Application Firewall* (WAF) dengan aturan OWASP ModSecurity CRS dalam pencegahan ancaman web.

1.5 Manfaat Penelitian

Berdasarkan rumusan masalah dan tujuan penelitian maka dapat diharapkan penelitian ini dapat memberi manfaat yaitu :

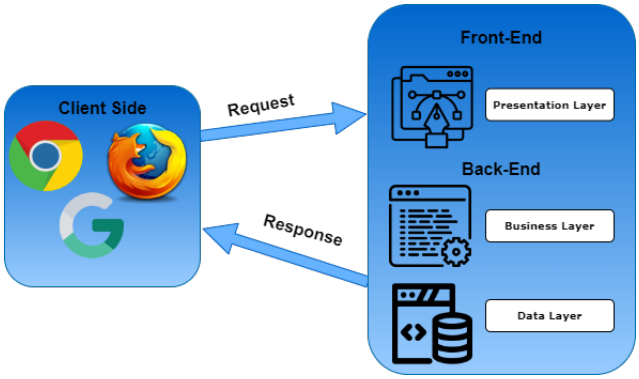
1. Meningkatkan keamanan aplikasi web: Implementasi ModSecurity *Web Aplication Firewall* (WAF) dengan aturan OWASP ModSecurity CRS pada aplikasi web dapat membantu mencegah berbagai jenis serangan berbasis web.

2. Memberikan pemahaman tentang serangan berbasis web: Melalui tahapan serangan, dapat memberikan pemahaman tentang ancaman keamanan web.

BAB II
TINJAUAN PUSTAKA

2.1 Aplikasi Web

Sejak tahun 1994 ketika internet menjadi tersedia untuk umum dan khususnya pada tahun 1995 ketika *World Wide Web* (WWW) menampilkan wajah yang dapat digunakan internet. WWW yang juga dikenal sebagai web, adalah kumpulan situs web atau halaman web yang disimpan di *server* web dan terhubung ke komputer lokal melalui internet. Hanya dalam satu dekade, web telah berevolusi dari gudang halaman yang digunakan terutama untuk mengakses informasi statis, sebagian besar ilmiah, ke platform yang kuat untuk pengembangan dan penerapan aplikasi. Aplikasi web merupakan program perangkat lunak yang berjalan di server web dan diakses melalui browser web seperti Google Chrome, Mozilla Firefox, Microsoft Edge dan lainya. Ini memungkinkan pengguna untuk berinteraksi dengan halaman web dinamis, menginput data, mengubah data, dan mengambil informasi dari database (Jazayeri, 2007). Aplikasi web dapat dikembangkan menggunakan berbagai teknologi dan bahasa pemrograman, untuk sisi *backend* termasuk PHP, Python, Ruby, Java, JavaScript dan lainnya. Untuk sisi *frontend* seperti HTML, CSS dan JavaScript. Ada juga kerangka kerja (*framework*) yang memfasilitasi pengembangan aplikasi web dengan menyediakan struktur, alat, dan komponen yang diperlukan. Secara umum, aplikasi web terdiri dari dua komponen utama: sisi klien (*client-side*) dan sisi *server* (*server-side*).



Gambar 2.1. *Standar Arsitektur Web Aplikasi*

Pada Gambar 2.1 sisi klien (*Client Side*) terdiri dari semua elemen yang berjalan di *browser* web pengguna, pada *client* melakukan *request*, untuk *Presentation Layer* atau (*frontend*) memungkinkan interaksi antara antarmuka dan *browser* web. Pada *Business Layer* atau *backend* yaitu logika bisnis atau *service* berjalan, permintaan *client* diproses di sini, diikuti dengan pelaksanaan instruksi logika bisnis yang ditentukan dalam permintaan tersebut. *Data Layer* terhubung ke logika bisnis dan, selain pengambilan data, juga mengelola penyimpanan data. Data diambil dari *server database* dan kemudian diteruskan ke *client* sebagai *response* sesuai dengan *request* dari *client* (Sturm dkk., 2017). Fungsi utama aplikasi web adalah mengambil data yang diminta pengguna dari *database*. Saat pengguna mengklik atau memasukkan URL di *browser*, aplikasi web segera menampilkan konten situs web yang diminta di *browser* (EC-Council, 2021).

2.2 Kerentanan, Ancaman dan Serangan Aplikasi Web dari OWASP Top 10

Tren terbaru dalam pelanggaran keamanan sistem menggambarkan bahwa tidak ada sistem atau jaringan yang kebal terhadap serangan. Semua organisasi yang menggunakan aplikasi khususnya aplikasi web perlu menerapkan keamanan yang kuat untuk memantau lingkungan aplikasi mereka untuk mengidentifikasi kerentanan dan mengatasinya sebelum dieksploitasi. Penting untuk memahami perbedaan antara kerentanan, ancaman, serangan keamanan yang merupakan insiden yang berdampak negatif pada infrastruktur organisasi.

2.2.1 Kerentanan

Dalam sebuah sistem aplikasi, umumnya ada dua penyebab utama sistem menjadi rentan: (1) konfigurasi perangkat lunak atau perangkat keras yang salah dan (2) praktik pemrograman yang buruk. Para penyerang mengeksploitasi kerentanan ini untuk melakukan berbagai jenis serangan terhadap sumber daya organisasi. Kerentanan mengacu pada kelemahan dalam desain atau implementasi sistem yang dapat dieksploitasi untuk membahayakan keamanan sistem. Kerentanan sering kali merupakan celah keamanan yang memungkinkan penyerang memasuki sistem dengan melewati otentikasi pengguna (Aslan dkk., 2023). Kerentanan dalam sistem atau jaringan diklasifikasikan kedalam kategori berikut:

- *Misconfiguration* adalah kerentanan yang paling umum dan terjadi secara sengaja atau tidak sengaja yang mempengaruhi server web, database dan jaringan. Penyerang dapat dengan mudah mendeteksi kerentanan *misconfiguration* ini menggunakan alat pemindaian dan kemudian mengeksploitasi sistem backend.

- *Buffer Overflows* adalah kerentanan perangkat lunak yang terjadi karena kesalahan pengkodean yang memungkinkan penyerang mendapatkan akses ke sistem target.

- *Unpatched Servers* adalah kerentanan yang diketahui dan belum diatasi melalui penerapan pembaruan atau perbaikan. Terdapat beberapa kasus organisasi yang mengoperasikan server tanpa tambalan dan konfigurasi yang keliru, yang berakibat pada terancamnya keamanan

- *Appliaction Flaws* merupakan kerentanan yang terjadi akibat tidak menerapkan validasi dan otoriasi pengguna.

2.2.2 Ancaman

Ancaman merupakan suatu tindakan yang memanfaatkan kerentanan keamanan dalam suatu sistem yang tidak diinginkan yang akhirnya bisa merusak, mengganggu operasional dan aktivitas fungsional sebuah organisasi. Ancaman dilakukan oleh penyerang untuk mendapatkan akses ke suatu sistem, ancaman hadir karna adanya kerentanan dalam suatu sistem. Ancaman dapat berupa segala jenis entitas atau tindakan yang dilakukan terhadap aset fisik atau tidak berwujud yang dapat mengganggu keamanan. Berikut ini merupakan klasifikasi ancaman seperti *Natural Threat*, *Unintentional Threats*, dan *Intentional Threats* (EC-Council, 2021).

- *Natural Threats* atau Faktor alam seperti kebakaran, banjir, listrik mati, petir, dan gempa bumi merupakan ancaman potensial terhadap aset organisasi. Sebagai contoh, hal ini dapat menyebabkan kerusakan fisik yang parah pada sistem komputer.

- *Unintentional Threats* adalah ancaman yang ada karena potensi kesalahan yang tidak disengaja terjadi di dalam organisasi. Contohnya termasuk keamanan

yang berasal dari pelanggaran orang dalam, kelalaian, kesalahan operator, administrator yang tidak terampil, karyawan yang malas, dan kecelakaan.

- *Intentional Threats* atau ancaman yang disengaja memiliki dua tipe
    - *Internal Threats* merupakan ancaman yang dilakukan oleh orang dalam, pada suatu organisasi seperti karyawan yang tidak puas atau karyawan yang lalai dan merugikan organisasi baik secara sengaja maupun tidak. Sebagian besar ancaman ini dilakukan oleh pengguna jaringan yang memiliki hak istimewa.
    - *External Threats* merupakan ancaman eksternal yang dilakukan dengan mengeksploitasi kerentanan yang sudah ada dalam jaringan, tanpa bantuan dari karyawan orang dalam. Oleh karena itu, potensi untuk melakukan ancaman eksternal tergantung pada tingkat keparahan sistem yang teridentifikasi akan kerentanan.

2.2.3 Serangan Aplikasi Web dari OWASP Top 10

Serangan merupakan bentuk konkret dari ancaman, serangan adalah tindakan atau metode aktual yang digunakan oleh aktor ancaman untuk mengeksploitasi kerentanan dan membahayakan keamanan sistem atau organisasi. OWASP Top 10 berbicara tentang sepuluh peringkat serangan keamanan informasi paling berbahaya untuk aplikasi web, yang disusun oleh komunitas pakar industri (Kumar dkk., 2021). *The Open Web Application Security Project* (OWASP) adalah komunitas terbuka yang didedikasikan untuk memungkinkan organisasi untuk mengembangkan, membeli, dan memelihara aplikasi dan API yang bisa dipercaya. OWASP Top 10 adalah daftar 10 risiko keamanan paling penting yang memengaruhi aplikasi web, yang dimulai pada tahun 2003 untuk membantu organisasi dan pengembang dengan titik awal untuk pengembangan yang aman. Pada Tabel 2.1 menunjukkan 10 risiko keamanan aplikasi web tahun 2017 & 2021 yang dikeluarkan OWASP Top 10 (Van Der Stock dkk., 2017). Daftar ini bertujuan untuk memberi pemahaman tentang risiko keamanan 10 teratas oleh OWASP (Bach-Nutman, 2020). OWASP Top 10 - 2021 telah mengalami beberapa perubahan besar dari OWASP Top 10 - 2017. Ada tiga kategori baru, empat kategori dengan perubahan penamaan

agar lebih akurat mencerminkan jenis kerentanan yang dihadapi dalam aplikasi saat ini, dan beberapa konsolidasi dalam 10 Teratas untuk tahun 2021. Adapun kategori baru, seperti *Server Side Request Forgery* (SSRF), *Insecure Design* dan *Software and Data Integrity Failures*, atau rekombinasi dari 10 kategori teratas sebelumnya seperti *Injection* yang sekarang juga mencakup *Cross-Site Scripting* (XSS).

Tabel 2.1. *OWASP Top 10 2017 to 2021*

OWASP Top 10 - 2017	**OWASP Top 10 - 2021**
A1:2017-*Injection*	A1:2021-*Broken Access Control*
A2:2017-*Broken Authentication*	A2:2021-*Cryptographic Failures*
A3:2017-*Sensitive Data Exposure*	A3:2021-*Injection*
A4:2017-*XML External Entities* (XXE)	A4:2021-*Insecure Design*
A5:2017-*Broken Access Control*	A5:2021-*Security Misconfigurations*
A6:2017-*Security Misconfiguration*	A6:2021-*Vulnerable & Outdated Components*
A7:2017-*Cross-Site Scripting* (XSS)	A7:2021-*Indentifications & Authentications Failures*
A8:2017-*Insecure Deserialization*	A8:2021-*Software & Data Integrity Failures*
A9:2017-*Using Components with Known Vulnerabilities*	A9:2021-*Security Logging & Monitoring Failures*
A10:2017-*Insufficient Logging & Monitoring*	A10:2021-*Server-Side Request Forgery*

Risiko dalam daftar diidentifikasi berdasarkan peringkat mereka pada daftar dan tahun daftar. Jadi, misalnya, risiko keamanan teratas dalam daftar terbaru adalah *Broken Access Control*. Ini diberi pengidentifikasi A1:2021. ”A” untuk AppSec, diikuti dengan peringkatnya dalam daftar, simbol ”:”, dan tahun. Transisi risiko keamanan OWASP Top 10 - 2021 (EC-Council, 2021):

- A1:2021-*Broken Accesss Control* berada pada posisi pertama dari posisi kelima di edisi sebelumnya, *Broken Accesss Control* adalah suatu metode mengidentifikasi kelemahan yang terkait dengan *access control*, melewati otentikasi, manipulasi metadata melalui token seperti JSON Web Token (JWT) dan kemudian membahayakan sistem. *access control* yang lemah, umumnya karena kurangnya deteksi otomatis dan fungsional yang efektif oleh pengembang aplikasi.

- A2:2021-*Cryptographic failures* naik ke posisi peringkat 2 yang sebelumnya dikenal sebagai *Sensitive Data Exposure*. Risiko ini terjadi ketika metode kriptografi tidak digunakan dengan tepat untuk melindungi data. Banyak aplikasi web tidak melindungi data sensitifnya dengan benar, dari pengguna yang tidak berwenang. Aplikasi web menggunakan algoritma kriptografi untuk mengenkripsi data mereka dan informasi sensitif lainnya yang mereka butuhkan untuk ditransfer dari server ke klien atau sebaliknya. *Sensitive Data Exposure* terjadi karena kelemahan seperti penyimpanan kriptografi yang tidak aman dan kebocoran informasi. Meskipun data dienkripsi, beberapa metode enkripsi kriptografi mempunyai kelemahan memungkinkan penyerang untuk mengeksploitasi dan mencuri data. Pengembang dapat menghindari serangan tersebut dengan kode aman yang benar dienkripsi menggunakan algoritma kriptografi yang aman untuk mengenkripsi data sensitif.

- A3:2021-*Injection* turun ke posisi ke-3 yang sebelumnya peringkat pertama di edisi sebelumnya. *Injection* adalah kerentanan situs web yang paling umum di Internet dan menduduki peringkat nomor satu di antara (OWASP Top 10). Kerentanan *Injection* memungkinkan penyerang menginput data yang tidak dipercaya dan dieksekusi sebagai bagian dari perintah atau kueri. Penyerang mengeksploitasi aplikasi web menyuntikkan kode, perintah, atau skrip berbahaya di gerbang masukan web yang rentan sedemikian rupa sehingga aplikasi menafsirkan dan menjalankan input berbahaya yang baru disediakan, yang pada gilirannya memungkinkan mereka untuk mengekstrak informasi sensitif, mengakibatkan hilangnya atau rusaknya data, kurangnya akuntabilitas, atau penolakan akses. Ada banyak tipe *Injection*, beberapa di antaranya dibahas di bawah ini :

    - ***SQL Injection*** Dalam beberapa tahun terakhir injeksi SQL telah muncul sebagai salah satu jenis serangan yang paling berbahaya untuk sistem berbasis web. Eksploitasi injeksi SQL pertama kali didokumentasikan pada tahun 1998 oleh peneliti keamanan siber dan peretas Jeff Forristal, dan digunakan untuk memanfaatkan kerentanan masukan yang tidak

divalidasi untuk meneruskan perintah SQL melalui aplikasi web untuk dieksekusi oleh database backend.

- ***Command Injection*** juga dikenal sebagai injeksi *shell* adalah kerentanan keamanan web, yaitu penyerang mengidentifikasi kerentanan validasi masukan dalam aplikasi web dan mengeksploitasi kerentanan dengan memasukkan perintah jahat ke dalam aplikasi untuk dieksekusi. Dalam serangan ini, perintah sistem operasi yang disediakan penyerang biasanya dijalankan dengan hak istimewa dari aplikasi yang rentan.
- ***LDAP Injection*** *The Lightweight Directory Application Protocol* (LDAP) adalah protokol yang digunakan untuk mendistribusikan daftar informasi yang diatur ke dalam pohon informasi direktori yang disimpan dalam *database* LDAP. *LDAP Injection* merupakan serangan yang digunakan untuk mengeksploitasi aplikasi berbasis web yang membuat pernyataan LDAP berdasarkan input pengguna. Ketika sebuah aplikasi gagal membersihkan input pengguna, penyerang mengubah pernyataan LDAP dengan bantuan *proxy* lokal. Teknik eksploitasi lanjutan yang sama yang tersedia di *SQL Injection* dapat diterapkan dengan cara yang sama di *LDAP Injection*.

■ A4:2021-*Insecure Design* adalah kategori baru dalam OWASP Top 10 tahun 2021, ini adalah kategori luas yang terkait dengan desain kritis dan kelemahan arsitektur dalam aplikasi web yang dapat dieksploitasi oleh peretas. Kerentanan ini berkaitan dengan bagaimana pengembang merancang program, merancang solusi, dan menggunakan praktik keamanan seperti pemodelan ancaman yang berarti bahwa *Insecure Design* dapat muncul dengan sendirinya dalam berbagai cara. Perlu diperhatikan bahwa *Insecure Design* berbeda dari *Insecure Implementation*, dan implementasi yang hampir sempurna tidak dapat mencegah cacat yang timbul dari desain yang tidak aman.

■ A5:2021-*Security Misconfiguration* naik ke posisi 5 dari posisi 6 di edisi sebelumnya, *Security Misconfiguration* masalah paling umum dalam keamanan web, yang sebagian disebabkan ke konfigurasi manual atau

tidak dikonfigurasi sama sekali , konfigurasi *header* HTTP yang salah, pesan kesalahan yang berisi informasi sensitif, dan tidak menambal atau meningkatkan sistem kerangka kerja secara tepat waktu atau tidak sama sekali. Apabila pengembang tidak mengonfigurasi server dengan benar, dapat mengakibatkan berbagai masalah yang dapat mempengaruhi keamanan aplikasi web. Masalah yang mengarah ke contoh seperti itu termasuk *Unvalidated Inputs*, *Parameter/Form Tampering*, *Improper Error Handling* dan *Insufficient Transport Layer Protection*.

- A6:2021-*Vulnerable and Outdated Components* kategori ini naik ke posisi ke 6, yang sebelumnya diberi judul *Components with Known Vulnerabilities* di edisi sebelumnya. Risiko ini adalah komponen sistem tidak lagi didukung atau telah ditentukan kerentananya. Contoh paling umum dari hal ini dalam suatu organisasi adalah versi perangkat lunak yang lama atau usang. Versi perangkat lunak yang lebih lama sering kali mengetahui kerentanan keamanan, membuatnya sangat rentan terhadap aktor ancaman, dan penting untuk menambal sistem ini dengan cepat untuk menghindari kemungkinan kerentanan keamanan. Penyerang dapat mengidentifikasi komponen atau ketergantungan yang lemah dengan memindai atau dengan melakukan manual analisis. Penyerang mencari kerentanan apa pun di situs eksploitasi seperti *Exploit Database* (https://www.exploit-db.com), *Security Focus* (https://www.securityfocus.com), dan *Zero Day Intiative* (https://www.zerodayinitiative.com).

  Ketika komponen yang rentan teridentifikasi, maka penyerang mengkustomisasi *exploit* sesuai kebutuhan dan mengeksekusi serangan. Eksploitasi yang berhasil memungkinkan penyerang menyebabkan kehilangan data yang serius atau mengambil alih kendali server.

- A7:2021-*Identification and Authentication Failures* berada pada posisi ke-7 dari posisi ke-2, yang sebelumnya dikenal *Broken Authentications*. Risiko ini terjadi saat aplikasi gagal menerapkan fungsi yang terkait dengan identitas pengguna, autentikasi, dan manajemen sesi dengan benar. Penerapan

autentikasi gagal karena fungsi kredensial yang lemah seperti ”ubah kata sandi”, ”lupa kata sandi”, ”pembaruan akun”, dan seterusnya. Maka dari itu pengembang harus sangat berhati-hati untuk menerapkan autentikasi pengguna dengan menggunakan metode otentikasi yang kuat melalui perangkat lunak khusus dan berbasis perangkat keras token kriptografi atau biometrik. Fungsi aplikasi yang sering terkait dengan otentikasi dan manajemen sesi diimplementasikan secara tidak benar, sehingga memungkinkan penyerang dapat mengeksploitasi kerentanan dalam otentikasi atau fungsi manajemen sesi seperti akun terbuka, *session IDs*, *logout*, *password manajement*, *timeouts*, dan lainya untuk mengambil identitas pengguna, pencurian data, atau kompromi seluruh sistem.

- ***Session ID in URLs*** aplikasi web membuat Id sesi untuk masing-masing pengguna, saat pengguna *login* (http://xxxxxxxx.com). Penyerang menggunakan *sniffer* untuk mengendus *cookie* yang berisi Id sesi atau mengelabui pengguna untuk mendapatkan Id sesi dan lalu gunakan kembali Id sesi tersebut untuk tujuan jahat.
- ***Password Exploitation*** Penyerang dapat mengidentifikasi kata sandi yang disimpan dalam database karena algoritma hashing yang lemah.
- ***Timeout Exploitation*** Jika waktu tunggu sesi aplikasi disetel ke durasi yang lebih lama, yang berarti sesi akan berlaku untuk jangka waktu yang lebih lama. Ketika pengguna menutup *browser* tanpa keluar dari situs yang diakses melalui komputer umum, penyerang dapat menggunakan browser yang sama nanti untuk melakukan serangan, seperti halnya Id sesi tetap berlaku; dengan demikian, penyerang dapat mengeksploitasi hak pengguna.

  Misal Seorang pengguna masuk ke (www.xxxxxx.com) menggunakan kredensialnya. Setelah melakukan tugas tertentu, dia menutup browser web tanpa keluar dari halaman. Batas waktu sesi aplikasi web diatur ke dua jam. Selama ditentukan interval sesi, jika penyerang memiliki akses fisik ke sistem pengguna, maka dia dapat

melakukannya dengan membuka *browser*, periksa riwayatnya, dan klik tautan www.xxxxxx.com, yang secara otomatis mengarahkannya ke akun pengguna tanpa perlu masuk kredensial pengguna.

- A8:2021-*Software and Data Integrity Failures* adalah kategori baru untuk tahun 2021, sebagai salah satu kerentanan paling umum yang diketahui dalam aplikasi web modern, yang berfokus pada pembaruan perangkat lunak dan *pipeline CI/CD* tanpa memverifikasi integritas. Kompleksitas arsitektur dalam siklus pembaruan modern sering kali memaksa pengembang untuk menggunakan *plugin*, modul, dan pustaka dari repositori publik dan sumber tidak tepercaya. Karena kerumitan seperti itu, kegagalan integritas perangkat lunak dan data (dikategorikan sebagai cacat desain) yang terjadi ketika data penting dan pembaruan perangkat lunak ditambahkan ke alur pengiriman tanpa memverifikasi integritasnya. Dengan tidak adanya validasi yang memadai, kegagalan integritas perangkat lunak dan data membuat aplikasi rentan terhadap pengungkapan informasi yang tidak sah, kompromi sistem, atau penyisipan kode berbahaya.

- A9:2021-*Security Logging and Monitoring Failures* yang sebelumnya *Insufficient Logging and Monitoring* yang berada pada posisi 9 dari yang sebelumnya ke-10. Aplikasi web memelihara *log* untuk melacak pola penggunaan, seperti kredensial pengguna dan kredensial admin. Pencatatan dan pemantauan yang tidak memadai mengacu pada skenario perangkat lunak pendeteksi tidak merekam peristiwa jahat atau mengabaikan detail penting tentang peristiwa tersebut. Penyerang biasanya menyuntikkan, menghapus, atau mengutak-atik *log* aplikasi web yang terlibat dalam aktivitas jahat atau menyembunyikan identitas mereka. Karena pencatatan dan pemantauan yang tidak memadai, deteksi upaya jahat penyerang menjadi lebih sulit dan penyerang bisa melakukan serangan jahat, seperti *password brute-force*, untuk mencuri kata sandi rahasia. Pengembang harus menerapkan kontrol berikut, bergantung pada risiko aplikasi seperti

- Pastikan semua login, kontrol akses, dan kegagalan validasi input sisi *server* dapat dicatat dengan konteks pengguna yang memadai untuk mengidentifikasi akun yang mencurigakan dan ditahan untuk waktu yang cukup agar analisis forensik tertunda.
- Memastikan *log* dibuat dalam format yang dapat digunakan dengan mudah oleh solusi manajemen *log*.
- Tim *DevSecOps* harus bekerja sama melakukan pemantauan dan peringatan yang efektif sehingga aktivitas mencurigakan dapat terdeteksi dan ditanggapi dengan cepat.
- Memastikan data *log* dikodekan dengan benar untuk mencegah injeksi atau serangan pada sistem *logging* atau pemantauan.
- Atau menggunakan kerangka kerja perlindungan aplikasi komersial dan sumber terbuka seperti OWASP ModSecurity CRS, dan perangkat lunak korelasi *log* sumber terbuka, seperti tumpukan Elasticsearch, Logstash, Kibana (ELK), yang menampilkan dasbor dan peringatan khusus.

■ A10:2021-*Server-Side Request Forgery* (SSRF) merupakan kategori baru untuk tahun 2021. Penyerang mengeksploitasi kerentanan keamanan web yang memungkinkan penyerang mendorong aplikasi sisi server untuk membuat permintaan ke lokasi yang tidak diinginkan. Oleh karena itu, penyerang memanfaatkan kerentanan SSRF di *server* web yang terhubung ke internet untuk mendapatkan akses ke *backend server* yang dilindungi oleh *firewall*. *Server* backend percaya bahwa permintaan dibuat oleh *server* web karena mereka berada di jaringan yang sama dan merespon dengan data yang disimpan di dalamnya.

2.3 Aktor Ancaman

Dari banyaknya ancaman dan serangan khususnya berbasis web, tidak lain karna aktor ancaman atau peretas. Motif aktor ancaman berasal dari gagasan bahwa sistem target menyimpan atau memproses sesuatu yang berharga, dan ini mengarah pada ancaman serangan terhadap sistem. Aktor ancaman mencoba

berbagai alat dan teknik serangan untuk mengeksploitasi kerentanan pada suatu sistem. Aktor ancaman atau peretas adalah orang yang membobol sistem atau jaringan tanpa otorisasi untuk menghancurkan, mencuri data sensitif, atau melakukan serangan berbahaya. Seorang peretas adalah individu yang cerdas dengan keterampilan komputer yang sangat baik, bersama dengan kemampuan untuk membuat dan menjelajahi perangkat lunak dan perangkat keras komputer. Biasanya, seorang peretas adalah insinyur atau pemrogram yang terampil dengan pengetahuan yang cukup untuk menemukan kerentanan dalam sistem target. Mereka umumnya memiliki keahlian subjek dan senang mempelajari detail berbagai bahasa pemrograman, perangkat lunak, dan sistem komputer. Pada Tabel 2.2 peretas biasanya termasuk dalam salah satu kategori berikut (EC-Council, 2021):

Tabel 2.2. Kelas Peretas atau Aktor Ancaman

Aktor Ancaman	Deskripsi Dari Aktor Ancaman
Black Hats	*Black Hats* adalah individu yang menggunakan keterampilan komputasi luar biasa mereka untuk tujuan ilegal atau berbahaya.
White Hats	*White Hats* atau penguji penetrasi adalah individu yang menggunakan keterampilan meretas mereka untuk tujuan *defensive*.
Gray Hats	*Gray Hats* adalah individu yang bekerja secara *offensive* dan *defensive* di berbagai waktu.
Script kiddies	*Script kiddies* adalah peretas tidak terampil yang mengkompromikan sistem dengan menjalankan skrip, alat, dan perangkat lunak yang dikembangkan oleh peretas sungguhan. Mereka biasanya berfokus pada kuantitas, bukan kualitas, dari serangan yang mereka mulai.
Cyber Terrorists	*Cyber Terrorists* adalah individu dengan berbagai keterampilan yang dimotivasi oleh keyakinan agama atau politik untuk menciptakan ketakutan akan gangguan jaringan komputer berskala besar.

Aktor Ancaman	Deskripsi Dari Aktor Ancaman
State-Sponsored Hackers	*State-Sponsored Hackers* atau Peretas yang disponsori negara adalah individu terampil yang memiliki keahlian dalam peretasan dan dipekerjakan oleh pemerintah untuk mendapatkan informasi rahasia dan merusak sistem informasi pemerintah atau militer lainnya.
Hacktivist	Hacktivisme melibatkan peretasan sistem digital pemerintahan atau perusahaan oleh peretas (*hacker*) sebagai tindakan protes untuk meningkatkan kesadaran akan agenda sosial atau politik mereka, dan untuk meningkatkan reputasi mereka sendiri.
Hacker Team	*Hacker Team* adalah peretas terampil yang memiliki sumber daya dan pendanaan sendiri. Mereka bekerja sama dalam sinergi untuk meneliti teknologi canggih.
Industrial Spies	*Industrial Spies* atau mata-mata industri adalah individu yang melakukan *spionase* perusahaan dengan memata-matai organisasi pesaing secara ilegal. Pelaku menggunakan *Advanced Persistent Threats* (APT) untuk menembus jaringan dan dapat tidak terdeteksi selama bertahun-tahun.
Insiders	*Insiders* adalah setiap karyawan (orang tepercaya) yang memiliki akses ke aset penting suatu organisasi. Ancaman orang dalam melibatkan penggunaan akses istimewa untuk melanggar aturan atau dengan sengaja menyebabkan kerusakan.
Criminal Syndicates	*Criminal Syndicates* adalah kelompok individu atau komunitas yang terlibat dalam kegiatan kriminal yang terorganisir, terencana, dan berkepanjangan. Tujuan utama para pelaku ancaman ini adalah menggelapkan uang secara ilegal.
Organized Hackers	*Organized Hackers* atau peretas terorganisir adalah sekelompok peretas yang bekerja sama dalam kegiatan kriminal. Peretas ini adalah penjahat kelas kakap yang menggunakan perangkat sewaan atau *botnet* dan layanan *crimeware*.

2.4 Alat Serangan Aplikasi Web

Alat serangan aplikasi web adalah program perangkat lunak yang dapat digunakan untuk mengeksploitasi kerentanan dalam aplikasi web. Beberapa alat dapat digunakan untuk mencuri data sensitif, seperti nomor kartu kredit atau kata sandi, atau untuk mendapatkan akses tidak sah ke sistem yang mendasari aplikasi web. Ada banyak jenis alat serangan aplikasi web yang tersedia, baik *open source* maupun komersial. Beberapa alat *open source* paling populer yaitu:

2.4.1 *Burp Suite*

Burp Suite merupakan platform terintegrasi untuk melakukan pengujian keamanan aplikasi web. *Burp Suite* adalah alat pengujian penetrasi yang digunakan oleh auditor keamanan, peretas, dan penguji penetrasi. *Burp Suite* ditempatkan di antara server dan klien web sehingga semua permintaan dan respons antara server dan klien web dialihkan melalui dan ditangkap oleh *Burp Suite*. Dengan jenis kerangka kerja ini, *Burp* memiliki kemampuan untuk mencatat, memodifikasi, dan menampilkan semua (HTTP) atau *Hypertext Transfer Protocol Secure* (HTTPS). Alat ini ditulis dalam Java dan dikembangkan oleh *PortSwigger Web Security* (Kim, 2020). *Burp Suite* memiliki berbagai alat yang bekerja sama untuk mendukung seluruh proses pengujian, dari awal pemetaan dan analisis permukaan serangan aplikasi untuk menemukan dan mengeksploitasi kerentanan keamanan.

2.4.2 OWASP *Zed Attack Proxy* (ZAP)

OWASP *Zed Attack Proxy* (ZAP) merupakan alat pengujian penetrasi yang hampir sama dengan *Burp Suite* alat ini digunakan untuk menemukan kerentanan dalam aplikasi web. Alat ini menawarkan pemindai otomatis serta seperangkat alat yang memungkinkan menemukan kerentanan keamanan secara manual. OWASP ZAP adalah *open source scanner* untuk mendeteksi kerentanan dalam aplikasi web dan mudah digunakan. Ini adalah salah satu proyek unggulan OWASP pengujian kerentanan aplikasi web (Jobin dkk., 2021).

2.4.3 Nikto

Nikto adalah pengujian kerentanan *server* web yang *open source* yang ditulis dalam bahasa Perl. alat ini akan menguji situs web untuk ribuan kemungkinan

masalah keamanan. Termasuk *file* berbahaya, layanan yang salah konfigurasi, skrip yang rentan, dan masalah lainnya. Nikto juga memeriksa beberapa *file* indeks di *server* web, terlihat untuk berbagai opsi di *server* HTTP. Selain itu, diperoleh informasi tentang *server* web dan perangkat lunak yang diinstal. Nikto menguji *server* web secara ketat dengan kemungkinan durasi minimum (Karangle dkk., 2019). Alat pemindaian Nikto juga digunakan menghasilkan permintaan berbahaya, termasuk pemeriksaan keberadaan *file* yang diketahui rentan, XSS, dan jenis serangan lainnya. Nikto sangat cocok untuk pengujian sistem keamanan seperti *Intrution Detection System* (IDS) ataupun WAF. Alat ini juga melaporkan permintaan mana yang diteruskan ke aplikasi, mengungkapkan potensi kerentanan dalam aplikasi.

2.4.4 *Sqlmap*

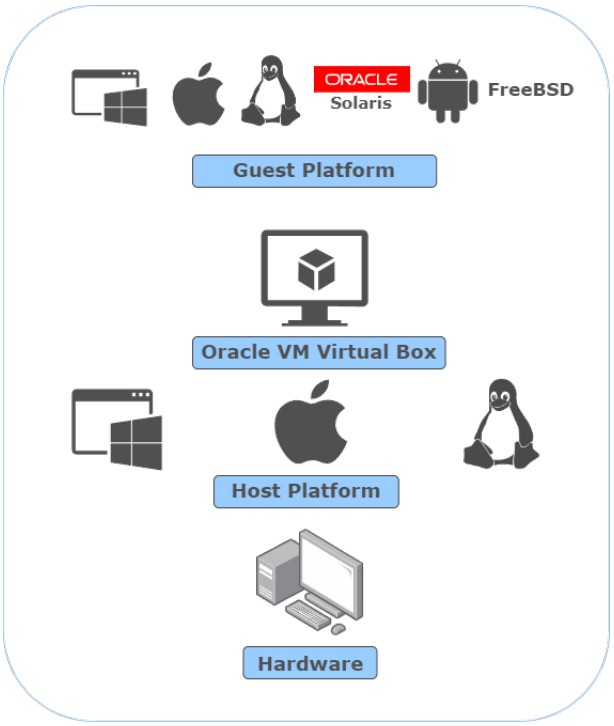
Sqlmap menjadi alat pengujian penetrasi *open source*, sqlmap mengotomatiskan proses mendeteksi dan mengeksploitasi kelemahan injeksi SQL dan mengambil alih *server* basis data. *Sqlmap* datang dengan mesin pendeteksi yang kuat, banyak fitur khusus untuk penetrasi tingkat lanjut penguji, dan berbagai sakelar untuk sidik jari basis data, pengambilan data dari *database*, mengakses sistem file yang mendasarinya, dan menjalankan perintah pada OS melalui koneksi *out-of-band*. Penyerang dapat menggunakan *sqlmap* untuk melakukan injeksi SQL pada situs web target melalui berbagai teknik *SQL Injection* (EC-Council, 2021).

2.5 Oracle VM *Virtual Box*

VirtualBox adalah aplikasi *open-source*, lintas platform untuk teknologi virtualisasi yang memungkinkan pengguna menjalankan *Operating Sytems* secara bersamaan dan dapat dijalankan pada *Host Platform* seperti Microsoft Windows, Linux, atau MacOS (https://www.virtualbox.org/). Teknologi mesin virtual pertama kali diperkenalkan oleh IBM pada tahun 1960-an dan 1970-an dan didefinisikan sebagai implementasi emulasi perangkat lunak dari perangkat keras.

VirtualBox awalnya dikembangkan oleh perusahaan Jerman (Innotek GmbH) pada 17 Januari 2007 sebagai paket perangkat lunak *open-source*, dan *Sun*

Microsystems mengakuisisi InnoTek GmBH pada Februari 2008, Pada 27 Januari 2010, Oracle *Corporation* membeli *Sun Microsystems* dan mengambil alih pengembangan *VirtualBox*. Secara resmi berganti nama menjadi Oracle VM *VirtualBox*, yang kini dikelola dan ditingkatkan oleh Oracle *Corporation* dan merupakan versi baru dari teknologi platform virtualisasi xVM Oracle. Di dalam *Machince Virtual*, Program utama yang mengelola *Virtual Machine* disebut *Virtual Machine Manager* (VMM). Sistem semacam ini disebut virtual sistem mesin, mesin yang disimulasikan disebut *Virtual Machine* (VM), dan perangkat lunak simulator disebut *Virtual Machine Monitor* (VMM) atau *Hypervisor*. Mekanisme riwayat proses *Machince Virtual* ini juga dapat digunakan untuk mengatur dan memelihara *database* pemantauan tambahan yang merekam berbagai data dan pengukuran tujuan khusus di luar status sroses (Goldberg, 1974). Dalam *VirtualBox*, Seperti yang ditunjukkan pada Gambar 2.2, pengguna dapat menjalankan *Operating Systems* (OS) seperti DOS, Windows, Linux, Solaris, IBM OS2, OpenBSD, dan bahkan Android.



Gambar 2.2. *Oracle VM VirtualBox Architecture*

2.5.1 Linux

Setelah AT&T *(American Telephone & Telegraph Company)* keluar dari proyek *Multics*. Sistem operasi Unix dirancang menggunakan bahasa C dan diimplementasikan oleh Ken Thompson dan Dennis Ritchie (juga adalah *developer* bahasa C), para peneliti dari AT&T *Bell Laboratories* perusahan riset industri dan pengembangan ilmiah yang dimiliki oleh Nokia, pada tahun 1969 dan pertama kali dirilis pada tahun 1970. Ketersediaan dan portabilitas Unix menyebabkannya diadopsi secara luas, disalin, dan dimodifikasi oleh institusi akademis dan organisasi bisnis, Salah satunya dikembangkan oleh Universitas Berkeley (Freddy, 2013).

Pada tanggal 27 September 1983, Richard Stallman memulai proyek GNU dengan tujuan untuk menciptakan sistem operasi yang mirip dengan Unix. Kebetulan, nama GNU adalah akronim rekursif yang sebenarnya adalah singkatan dari *'GNU is Not Unix'*. Proyek ini bertujuan untuk membuat alternatif yang kompatibel dengan sistem Unix berpemilik. Pekerjaan dimulai pada tahun 1984. Kemudian, pada tahun 1985, Stallman memulai *Free Software Foundation* dan menulis GNU *General Public License* (GNU GPL) pada tahun 1989. Pada awal tahun 1990-an, banyak program yang dibutuhkan dalam sistem operasi (seperti perpustakaan, kompiler, editor teks, *shell UNIX*, dan sistem *windowing*) telah selesai, meskipun tingkat rendah seperti driver perangkat, daemon, dan kernel terhenti dan tidak lengkap (Elcot, 2014).

Di sisi lain, MINIX sistem operasi mirip Unix, dibuat oleh seorang profesor Belanda kelahiran AS yang bernama Andrew S. Tanenbaum, tujuanya ingin mengajari murid-muridnya cara kerja sistem operasi. Pada tahun 1991, Linus Torvalds seorang mahasiswa Ilmu Komputer di Universitas Helsinki dan seorang hacker otodidak (Bretthauer, 2001). Dia menulis program khusus untuk perangkat keras yang dia gunakan dan tidak bergantung pada sistem operasi karena dia ingin menggunakan fungsi-fungsi PC barunya dengan prosesor 80386. Pengembangan dilakukan di MINIX dengan menggunakan kompiler GNU C. Kompiler GNU C masih menjadi pilihan utama untuk *meng-compile* Linux saat ini. Namun demikian, kode tersebut dapat dibuat dengan kompiler lain, seperti *Intel C Compiler*, seperti yang ditulis Torvalds dalam bukunya *Just for Fun* dia akhirnya menulis sebuah kernel

sistem operasi. Pada tanggal 25 Agustus 1991, (pada usia 21 tahun) mengumumkan sistem ini dalam sebuah posting *Usenet* ke *newsgroup* ”comp.os.minix.”.

Freax adalah nama yang ingin dipilih Linus Torvalds untuk penemuannya. Nama ini berasal dari singkatan dari kata *”free”, ”freak”,* dan ”x”, untuk menyinggung Unix. Ia menyimpan *file-file* tersebut dengan nama *”Freax”* selama kurang lebih lima tahun selama awal pengerjaannya pada sistem ini. Pada awalnya, Torvalds menolak untuk menggunakan nama ”Linux” karena terlalu egois. Pada bulan September 1991, file-file tersebut diunggah ke *server FTP FUNET* (ftp.funet.fi) untuk membantu proses pengembangan. Ari Lemmke, rekan kerja Torvalds di Universitas Teknologi Helsinki (HUT) yang pada saat itu menjadi salah satu administrator sukarelawan untuk *server* FTP, menganggap nama *”Freax”* sebagai nama yang tidak tepat. Akibatnya, ia menamai server tersebut dengan nama ”Linux” tanpa berkonsultasi dengan Torvalds. Namun, setelah menyetujui ”Linux”, Torvalds menyertakan panduan audio untuk menunjukkan cara mengucapkan kata ”Linux” dengan benar (Linus dan Diamond, 2001). Ada banyak distribusi *(distro)* Linux yang tersedia, masing-masing dengan karakteristik, tujuan, dan target audiensnya sendiri, seperti distribusi Linux yang populer seperti Ubuntu, *RedHat*, Debian, CentOS, Fedora, Arch Linux, *openSUSE*, Manjaro dan ada banyak distribusi Linux lain yang tersedia untuk memenuhi berbagai kebutuhan dan preferensi.

2.5.2 Kali Linux

Proyek Kali Linux dimulai pada tahun 2012, ketika *Offensive Security* memutuskan untuk mengganti proyek *Backtrack Linux* mereka, yang dikelola secara manual, dengan sesuatu yang dapat menjadi turunan *distro* Debian yang lengkap dengan semua infrastruktur yang diperlukan dan teknik pengemasan yang lebih baik. Keputusan dibuat untuk membangun Kali Linux diatas distribusi Debian karena berbagai pilihan perangkat lunak yang kualitasnya dan stabilitasnya tersedia. Pada Maret 2013 rilis pertama (versi 1.0) muncul dan didasarkan pada debian 7 *”Wheezy”*, distribusi stabil pada saat itu. Selama dua tahun setelah versi 1.0, Kali memunculkan banyak pembaruan tambahan, memperluas jangkauan aplikasi yang tersedia dan meningkatkan dukungan perangkat keras, berkat rilis kernel yang lebih baru.

Pada tahun 2015, ketika Debian 8 *"Jessie"* keluar, tindakan untuk *me-rebase* Kali Linux diatasnya. Sementara Kali Linux 1.x menghindari GNOME *Shell* sebagai gantinya mengandalkan GNOME *Fallback*. Dalam versi ini memutuskan untuk menyempurnakan dengan menambah beberapa ekstensi GNOME *Shell* untuk mendapatkan fitur yang hilang, terutama untuk menu aplikasi. Output dari pekerjaan ini menjadi Kali Linux 2.0, diterbitkan pada Agustus 2015. Distribusi Kali Linux didasarkan pada Debian *Testing*, karenanya sebagian besar paket yang tersedia di Kali Linux berasal langsung dari repositori Debian (Hertzog dkk., 2017). Kali dirancang untuk pengujian penetrasi dan dilengkapi dengan *tools hacking* yang signifikan, Kali Linux menawarkan (*Installer, NetInstaller* dan *Live*) untuk diunduh (https://www.kali.org/). Kali Linux tersedia untuk arsitektur ukuran 64*bit* dan 32*bit* dan tersedia sebagai *Virtual Machine* untuk dijalankan sebagai *guest platform* di bawah *VirtualBox* atau *VMware* (Santoso, 2022).

2.6 Apache *Server*

Sebelum Apache, perangkat lunak *server* yang paling terkenal di web adalah "*daemon*" HTTP domain publik (httpd), yang diciptakan oleh Rob McCool di Pusat Nasional untuk aplikasi *Supercomputing*, Universitas Illinois di Urbana-Champaign. Namun, setelah McCool meninggalkan NCSA pada tahun 1994 untuk bekerja di Netscape, pengembangan NCSA httpd berhenti, dan banyak pengelola situs web mulai mengembangkan ekstensi dan memperbaiki bug mereka sendiri. Pada Februari 1995, sekelompok *Webmasters* ini bertemu melalui internet untuk mengoordinasikan perubahan mereka dan menghasilkan distribusi umum. Daftar surat, ruang informasi bersama (FTP dan *HTTP-served directories*) dan kredensial untuk pengembang inti yang dibuat pada mesin di San Francisco Bay Area California, dengan *bandwidth* dan ruang *disk* yang disumbangkan oleh *HotWired* dan Organik *Online*. Kurang dari setahun setelah rilis pertamanya, server Apache melampaui httpd NCSA sebagai server terkemuka di Internet.

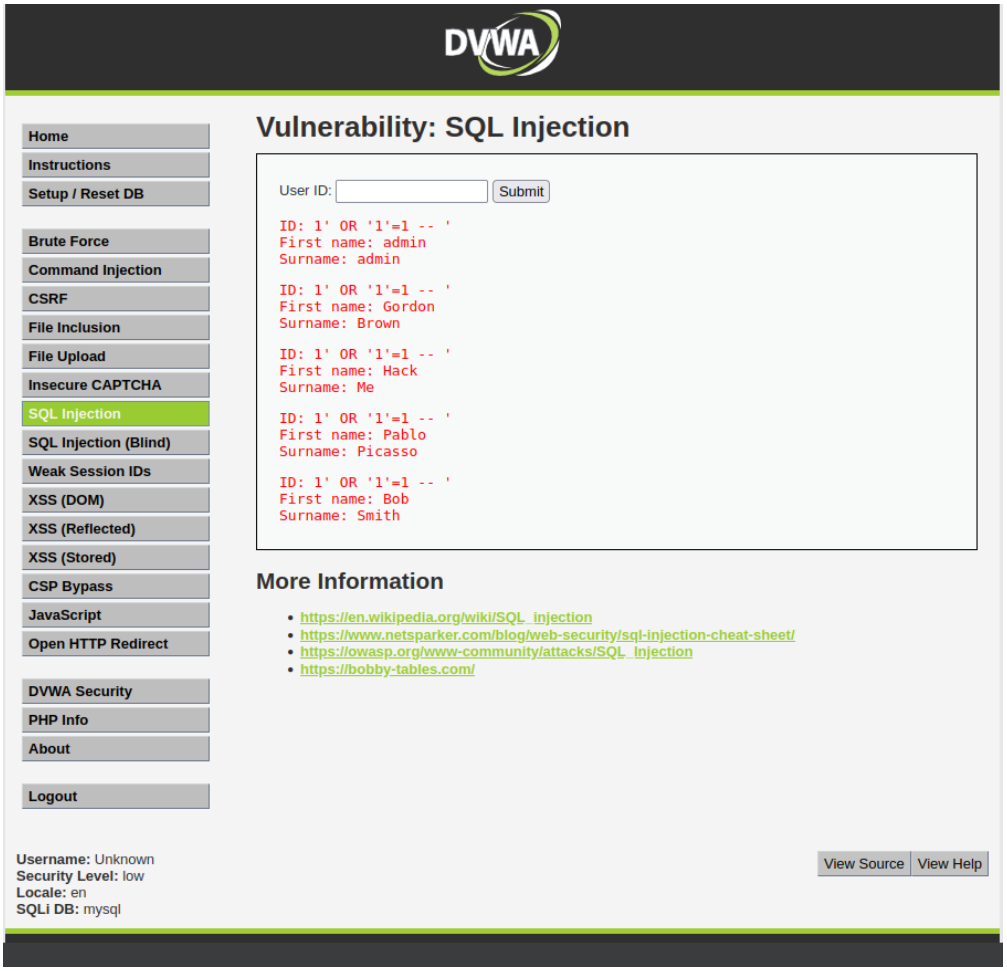
Pada Juni 1997, survei *Netcraft* melaporkan lebih dari setengah juta situs Web menggunakan Apache dan turunannya (Fielding dan Kaiser, 1997). Apache *Server*, juga dikenal sebagai Apache HTTP *Server*, adalah perangkat lunak server web sumber terbuka dan gratis yang dikembangkan dan dikelola oleh Apache *Software*

Foundation. Apache *Server* dirancang untuk menghosting dan menyajikan konten web, memungkinkan pengguna untuk mengakses situs web, aplikasi web, dan sumber daya online lainnya. Ini beroperasi pada protokol HTTP (*Hypertext Transfer Protocol*) dan HTTPS (HTTP *Secure*), memungkinkan komunikasi antara *browser* web (klien) dan *server*.

2.7 *Damn Vulnerable Web Application* (DVWA)

Damn Vulnerable Web Application (DVWA) dikembangkan oleh RandomStorm yang merupakan aplikasi web *open-source* yang diatur ke dalam modul yang terkait dengan kerentanan tertentu (Giannini, 2015). DVWA adalah aplikasi web yang dibangun dengan menggunakan MySQL sebagai *backend database* dan PHP. Tujuan utamanya adalah untuk menjadi bantuan bagi para profesional keamanan untuk menguji keterampilan dan alat mereka. Tujuan DVWA adalah untuk membantu pengembang keamanan aplikasi web di kelas yang terkontrol, untuk mempraktikkan beberapa kerentanan web paling umum, dengan berbagai tingkat kesulitan dengan antarmuka yang sederhana. Pada DVWA memiliki kerentanan yang nantinya akan dieksploitasi baik secara manual maupun dengan alat serangan, yang nantinya menjadi alat pengujian efektivitas ModSecurity (Priyanka dan Smruthi, 2020).

Dalam petunjuk DVWA terdapat fitur bantuan yang memungkinkan melihat petunjuk dan tip mengenai kerentanan tersebut. Ada juga tautan tambahan untuk melihat latar belakang lebih lanjut, yang terkait dengan masalah keamanan tersebut. Pada DVWA terdapat peringatan yang diberikan oleh pihak *developer* mengenai implementasi DVWA ini seperti (”Jangan mengunggahnya ke folder HTML publik penyedia *hosting* atau *server* apa pun yang menghadap Internet”), karena *server* tersebut sangat rentan tersusupi melalui pemasangan DVWA. Pihak *developer* menyarankan penggunaan seperti *VirtualBox* atau *VMware* untuk *Machine Vitrual*, yang diatur ke mode jaringan NAT (DVWA, 2023). Pada Gambar 2.3, DVWA memiliki banyak kerentanan aplikasi web yang memengaruhinya dan mempunyai tingkat keamanan yang berbeda. Setiap tingkat keamanan memberikan tantangan kepada pengguna dan juga menunjukkan bagaimana setiap kerentanan dapat dilawan diukur dengan pengkodean aman (Ryan Dewhurst, 2009).



Gambar 2.3. Tampilan Antarmuka Beberapa Serangan **(DVWA)**

Sumber: (DVWA, 2023)

Berikut adalah penjelasan dari masing-masing tingkat keamanan:

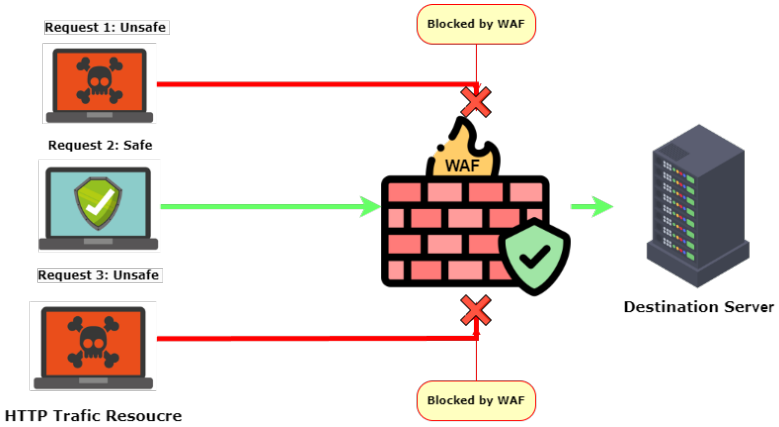
- *Low* - Tingkat keamanan ini benar-benar rentan dan tidak memiliki tindakan pengamanan sama sekali . Kegunaannya adalah sebagai contoh bagaimana kerentanan aplikasi web bermanifestasi melalui praktik pengkodean yang buruk dan berfungsi sebagai *platform* untuk mengajar atau mempelajari teknik eksploitasi dasar.
- *Medium* - Pengaturan ini dirancang terutama untuk memberikan contoh praktik keamanan yang buruk kepada pengguna. Pengembang telah mencoba, tetapi gagal, untuk mengamankan aplikasi. Hal ini juga berfungsi sebagai tantangan bagi pengguna untuk menyempurnakan teknik eksploitasi mereka.

- *High* - Opsi ini merupakan perluasan ke tingkat kesulitan sedang, dengan campuran praktik buruk yang lebih keras atau alternatif untuk mencoba mengamankan kode. Kerentanan mungkin tidak memungkinkan tingkat eksploitasi yang sama, serupa di berbagai kompetisi *Capture The Flags* (CTFs).

- *Imposibble* - Tingkat ini harus aman terhadap semua kerentanan . Ini digunakan untuk membandingkan kode sumber yang rentan dengan kode sumber yang aman. Sebelum DVWA v1.9, level ini dikenal sebagai 'tinggi'.

2.8 *Web Application Firewall* (WAF)

Pada dasarnya *Web Application Firewall* merupakan *gatekeeper* untuk sebuah *website*. WAF dapat melindungi aplikasi web dari *cyber attack* seperti *cross-site-scripting* (XSS), *cross-site forgery*, SQL *Injection*, DDoS, dan lainnya (Fauzi dkk., 2020). WAF dipasang di mesin yang sama dengan *server* web, (Endraca, 2013). Mesin WAF terdiri dari dua modul: (a) *Configuration Module* (CM) dan (b) *Packet Analysis Module* (PAM), ketika paket langsung diterima dari internet, *file* aturan memfilternya dari CM dan meneruskan lalu lintas ke PAM. PAM menganalisis paket dan mengekstrak fitur dari paket. Menggunakan data yang telah dilatih sebelumnya, ia menguji dan mengidentifikasi sifat/karakter dari paket itu. Oleh karena itu, hanya paket yang dianalisis dan diizinkan yang diteruskan melalui PAM ke *server* aplikasi web, WAF beroperasi menggunakan seperangkat aturan tertentu yang disebut kebijakan (Dawadi dkk., 2023). WAF merupakan kemanan pertama bagi aplikasi web tetapi WAF tidak menyelesaikan semuah masalah keamanan, masih diperlukan penerapan metode seperti *secure code*, *patch server*, memperhatikan keamanan untuk mempraktikkan pengkodean dan keamanan lainnya, tapi ketika ada ancaman serangan maka sistem akan lebih baik dengan WAF, bukan berarti itu menyelesaikan semuah masalah keamanan pada sistem aplikasi web. WAF biasanya memberi waktu lebih sulit bagi penyerang untuk mengeksploitasi situs aplikasi web dan memungkinkan seorang penyerang akan menyerah dan pindah pada situs lain yang lebih mudah untuk di serang.



Gambar 2.4. Skema *Web Application Firewall* **(WAF)**

Pada Gambar 2.4 dalam setiap model penerapan WAF selalu ditempatkan di depan aplikasi web, memeriksa semua *request* yang dikirimkan klien. Dengan demikian, kebijakan ini menentukan WAF mencari perilaku lalu lintas dan memutuskan tindakan apa yang perlu diambil dengan kerentanan (Pantoulas, 2022). WAF akan terus memindai aplikasi web dan menerima permintaan *GET* dan *POST* untuk mengidentifikasi dan memfilter permintaan dengan aktivitas berbahaya (Alaoui dan Nfaoui, 2022). Selain itu, WAF yang cerdas bahkan dapat meminta untuk mengidentifikasi apakah pesertanya adalah manusia atau *bot*. Ketika kerentanan ditemukan dalam aplikasi WAF segera memblokir permintaan penyerang, misalnya *bot* dan alamat IP yang diserang. WAF adalah garis pertahanan pertama melawan serangan kompleks yang mengancam integritas organisasi. WAF menyediakan solusi yang paling efektif dan efisien seperti (Clincy dan Shahriar, 2018):

- Perlindungan input, menyediakan filter aplikasi komprehensif yang hanya menerima input pengguna yang valid.
- Validasi HTTP mendeteksi kerentanan HTTP dan mencegah serangan dengan menyiapkan aturan validasi.
- Kebijakan yang disesuaikan dengan aplikasi yang digunakan secara luas diatur sesuai dengan persyaratan dan kebutuhan khusus. Dengan demikian, ini melindungi aplikasi dari kerentanan dan juga memberikan wawasan waktu nyata tentang lalu lintas.

- Pencegahan kebocoran data memberikan peringatan dan mencegah segala jenis lalu lintas yang tidak biasa atau kebocoran data dengan mengidentifikasi, memfilter, dan melindungi data pribadi.

- Pemblokiran serangan otomatis menyediakan alat otomatisasi untuk memblokir serangan dengan menolak lalu lintas berbahaya memasuki jaringan.

2.9 ModSecurity

ModSecurity adalah salah satu *Web Appliaction Firewall* (WAF) *open-source* yang awalnya merupakan modul server web apache, sama seperti *firewall* tradisional, memfilter data yang masuk dan keluar serta mampu menghentikan lalu lintas yang dianggap berbahaya menurut seperangkat aturan yang telah ditetapkan. Imajinasikan ModSecurity sebagai penjaga perbatasan, setiap permintaan diperiksa untuk memastikan tidak ada muatan yang tidak sah, yang masuk ke server web. Ketika serangan ditemukan, detailnya dapat ditulis ke *file log*, atau *email* dikirim ke administrator situs untuk mengingatkan upaya intrusi (Mischel, 2009). ModSecurity melindungi aplikasi web terhadap serangan web, seperti *SQL Injection* (SQLi), *Local File Inclusion* (LFI), dan *cross-site scripting* (XSS). Tiga vektor serangan bersama-sama menyumbang 95% dari serangan *Layer* 7 yang diketahui pada Q1 2017, menurut laporan Akamai *State of Internet – Security*.

Laporan tersebut juga menyatakan bahwa tujuan penyerangan berbeda-beda, namun mayoritas memang demikian bertujuan untuk mencuri data. ModSecurity digunakan oleh lebih dari satu juta situs web saat ini, menjadikannya milik dunia WAF yang paling banyak digunakan. ModSecurity merupakan *community-driven* yang telah menyebabkan adopsi yang luas oleh organisasi mulai dari bisnis kecil kepada pemerintah dan perusahaan. ModSecurity menggunakan bahasa aturan berdasarkan standar *Perl Compatible Regular Expression* (PCRE), yang membuatnya jauh lebih mudah diakses daripada banyak WAF lainnya.

2.9.1 Sejarah Singkat ModSecurity

Seperti proyek *open-scource* lainnya, ModSecurity dimulai sebagai hobi. Pengembangan perangkat lunak telah menjadi perhatian utama Ivan Ristic pada

tahun 2002, Dia membayangkan membuat solusi perangkat lunak yang akan duduk di depan aplikasi web, menganalisis data saat mengalir masuk dan keluar. Solusi seperti ini bisa keduanya memberikan visibilitas dan memblokir serangan potensial (Faisal Memon dan Pleshakov, 2017). Versi pertama dirilis pada November 2002. Awalnya Apache 1.3.x, tidak memiliki API pencegatan atau pemfilteran, Apache 2.x memperbaiki banyak hal dengan menyediakan API yang memungkinkan intersepsi konten. Pada tahun 2004, Ivan Ristic menghabiskan waktunya untuk Mengembangkan ModSecurity. Pada 2006, ketika ModSecurity berhadapan langsung dengan *firewall* applikasi web lainnya, dalam evaluasi yang dibuat oleh Forrester Research. Belakangan tahun 2006 Modsecurity diakusisi oleh perusahaan Breach Security dan memiliki banyak tim. ModSecurity 2.0, dirilis pada akhir tahun 2006 pada saat penulisan Modsecurity *Community Console*. Ivan Ristic berhenti atas ModSecurity pada Januari 2009, ketika keluar dari Breach Security, dan struktur tim pun juga berubah dan menghasilkan peningkatan dengan CSR v2. Pada tahun 2010, perusahaan Trustwave mengakuisisi Breach Security dan merevitalisasi ModSecurity (Ristic, 2010).

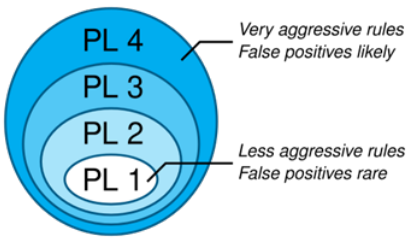
2.9.2 *Open Web Application Security Project* (OWASP) ModSecurity *Core Rule Set* (CRS)

ModSecurity adalah *Web Application Firewall* (WAF) *open source* yang memberikan perlindungan terhadap berbagai jenis serangan berbasis web. Meskipun ModSecurity menyediakan kerangka kerja yang kuat untuk membuat kebijakan keamanan yang disesuaikan, ModSecurity membutuhkan seperangkat aturan untuk menegakkannya. *Open Web Application Security Project* (OWASP) *Core Rule Set* (CRS) adalah seperangkat aturan yang banyak digunakan untuk ModSecurity atau *firewall* aplikasi web yang kompatibel. Seperangkat aturan tersebut dapat dilihat di repositori Github (https://github.com/coreruleset/coreruleset/), yang memberikan perlindungan terhadap berbagai serangan berbasis web termasuk OWASP Top 10 dengan minimal *false alert*.

OWASP ModSecuirty CRS merupakan seperangkat aturan yang telah ditentukan sebelumnya yang dapat digunakan dengan ModSecurity untuk melindungi aplikasi web dari serangan berbasis web yang umum. ModSecurity

sendiri tidak memiliki aturan untuk mendeteksi dan mencegah serangan. Oleh karena itu, diperlukan penggunaan kumpulan aturan seperti OWASP CRS untuk memberikan perlindungan terhadap serangan yang dikenal dan tidak dikenal. OWASP ModSecurity CRS terus diperbarui dengan tanda serangan terbaru, yang menjadikannya solusi efektif untuk melindungi aplikasi web dari ancaman baru dan yang muncul. Singkatnya, ModSecurity membutuhkan penggunaan OWASP ModSecurity CRS untuk memberikan lapisan keamanan yang efektif terhadap berbagai jenis serangan berbasis web. Tanpa aturan yang ditetapkan seperti OWASP ModSecurity CRS, ModSecurity tidak akan dapat mendeteksi dan mencegah serangan secara efektif (Folini dan Ristic, 2017).

Didalam OWASP ModSecurity CRS memiliki keunikan yang dinamai dengan paranoia level (PL) atau tingkat paranoia. Paranoia level adalah konsep yang penting ketika bekerja dengan (CRS), paranoia level (PL) memungkinkan untuk menentukan seberapa agresif (CRS), ada 4 (PL) dalam OWASP ModSecurity (CRS) dan untuk *default*, berada pada (PL) 1. Paranoia level 1 (PL 1) menyediakan seperangkat aturan yang hampir tidak pernah memicu *false positive* (idealnya tidak pernah, tetapi bisa saja terjadi, tergantung pada pengaturan lokal). Paranoia level 2 (PL 2) menyediakan aturan tambahan yang mendeteksi lebih banyak serangan (aturan ini beroperasi sebagai tambahan dari aturan PL 1), tetapi ada kemungkinan bahwa aturan tambahan juga akan memicu *false positive* baru atas permintaan HTTP yang sah. Hal ini berlanjut pada PL 3, terdapat lebih banyak aturan ditambahkan yaitu untuk serangan tertentu dan menyebabkan lebih banyak lagi *false positive*. Kemudian pada PL 4, aturannya sangat agresif sehingga mendeteksi hampir semua serangan yang mungkin terjadi, tetapi juga menandai banyak *traffic* yang sah sebagai berbahaya.

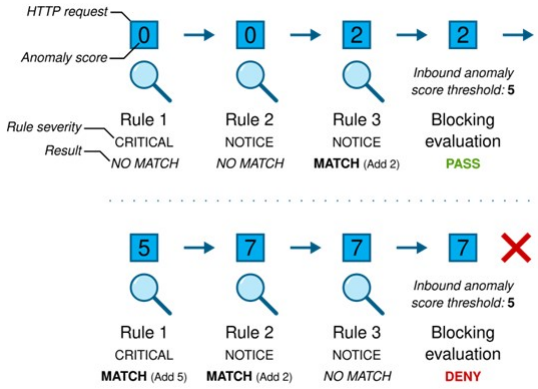


Gambar 2.5. Paranoia Level
Sumber: (OWASP ModSecurity CRS)

Pada Gambar 2.5 merupakan tingkatan PL, tingkat paranoia yang lebih tinggi membuat penyerang lebih sulit untuk tidak terdeteksi. Namun hal ini harus dibayar dengan lebih banyak hasil *false positive*. Itulah sisi negatif dari menjalankan seperangkat aturan yang mendeteksi hampir semua hal: bisnis/layanan/aplikasi web juga terganggu. Ketika *false positive* terjadi, perlu disetel ulang, dalam istilah ModSecurity: pengecualian aturan perlu ditulis. Pengecualian aturan adalah aturan yang menonaktifkan aturan lain.

2.9.3 Penilaian Anomali

Penilaian anomali atau *anomaly score*, juga dikenal sebagai "deteksi kolaboratif", adalah mekanisme penilaian yang digunakan dalam aturan OWASP ModSecurity CRS. CRS memberikan skor numerik pada transaksi HTTP (permintaan dan tanggapan), yang menunjukkan betapa 'anomali' transaksi tersebut. Skor anomali kemudian dapat digunakan untuk membuat keputusan pemblokiran. Kebijakan pemblokiran CRS *default* misalnya, adalah memblokir transaksi apa pun yang memenuhi atau melampaui ambang batas *anomali score* yang ditentukan.



Gambar 2.6. Penilaian anomali pada permintaan

Sumber: (OWASP ModSecurity CRS)

Pada Gambar 2.6 aturan OWASP ModSecurity CRS dirancang untuk mendeteksi jenis serangan tertentu dan perilaku jahat dijalankan. Jika permintaan yang dikirimkan cocok dengan aturan yang ada pada OWASP ModSecurity CRS maka permintaan diberikan *anomaly score*, sampai aturan terakhir jika permintaan

memiliki *anomaly score* lebih atau sama dari ambang batas anomali maka permintaan tersebut diblokir. Selain itu, setiap aturan yang cocok biasanya akan mencatat catatan kecocokan untuk referensi nanti, termasuk ID aturan yang cocok, data yang menyebabkan kecocokan, dan URI yang diminta.

Selanjutnya, setelah semua aturan yang memeriksa data respons telah dijalankan, evaluasi pemblokiran putaran kedua akan dilakukan. Jika skor anomali keluar lebih besar atau sama dengan ambang batas skor anomali keluar, maka respons tidak dikembalikan ke pengguna.