BAB II

TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 Keamanan *Website*

Keamanan *website* merupakan serangkaian tindakan yang diambil untuk melindungi Situs *web* dari berbagai ancaman dikenal sebagai keamanan *website*. Menurut (Gultom & Harahap, 2015) *website* merupakan kumpulan halaman *web* yang dihubungkan melalui jalur internet, yang memungkinkan orang diseluruh dunia dengan koneksi tanpa batas.

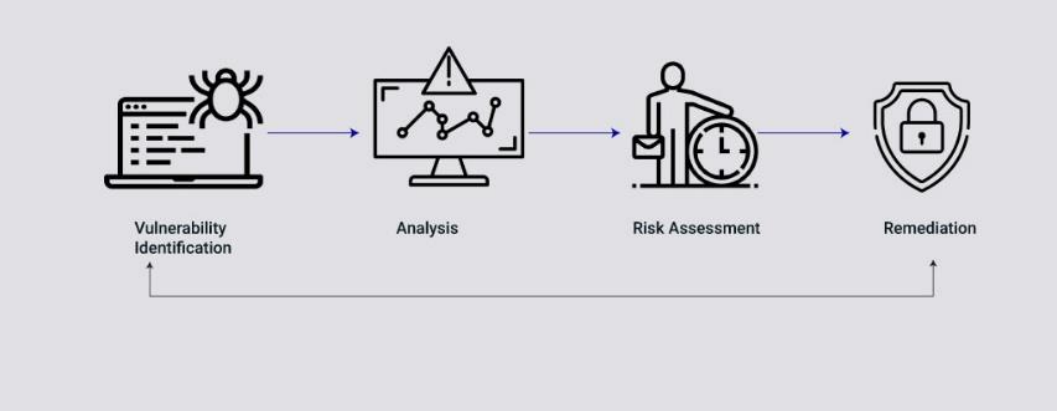
Website adalah layanan informasi yang dapat diakses oleh pengguna yang terhubung dalam suatu jaringan internet, kasus penelitian (Ary et al., 2020) terjadi serangan deface pada *website* Lembaga X pada tahun 2014. Serangan ini mengakibatkan *website* Lembaga X tidak dapat diakses. Berdasarkan kejadian ini, dilakukan evaluasi keamanan *website* Lembaga X dengan menggunakan metode *penetration testing* untuk mengetahui dan mengidentifikasi celah keamanan pada *website.*

2.1.2 Definisi *Website*

Website adalah kumpulan dari halaman-halaman situs yang terangkum dalam sebuah domain atau subdomain, yang tempatnya berada di dalam *Word Wide Web (WWW)* di dalam internet. *Website* terdiri dari beberapa laman yang berisi informasi dalam bentuk data digital baik berupa *text*, gambar, vidio, audio, dan animasi lainnya yang disediakan melalui jalur koneksi internet (Jonathan & Lestari, 2015).

2.1.3 *Vulnerability Assessment*

Menurut (Anjar Priandoyo, 2006), salah satu cara untuk mengukur keamanan system adalah dengan menggunakan *VA*. *VA* merupakan komponen pengendalian *preventif* dalam keseluruhan *spektrum* pengendalian IT. *VA* diharapkan memberi inspirasi untuk mengontrol keamanan informasi dalam Perusahaan. Menurut (Achmad Nur Sholeh, 2019) *VA* merupakan suatu rangkaian proses yang dilakukan untuk meninjau *system* yang memiliki potensi kerentanan.



Gambar 2. 1 Topologi *Vulnerability Assessment*

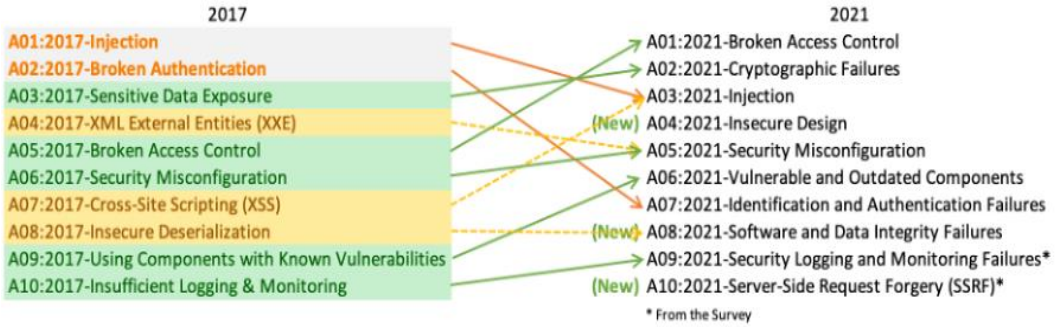
(Achmad Nur Sholeh, 2019)

Dari beberapa penelitian terdahulu, metode *VA* bekerja secara efektif dalam mengevaluasi kerentanan suatu *web.* Pada penelitian (Jurnal & Lana Rahardian, 2022) menunjukkan dalam pengujian keamanan *website* dapat dilakukan dengan metode *VA* menggunakan aplikasi *Acunetix* hingga menemukan hasil yang detail, dan berhasil membuat *website new golf* menjadi bernilai *high.* Pada penelitian (Orisa & Ardita, 2021) menunjukkan pengujian pada *host* target yaitu *POST, OPTIONS, GET,* dan *HEAD* menggunakan metode *VA* dibantu dengan *Networking Mapping (NMAP).* Hasil peneliti bahwa *VA* dengan *NMAP* menunjukkan bahwa

target yang diuji sebagai *HTTP open proxy.* Kemudian target yang diuji tidak terdeteksi *XSS* dan *host* juga tidak terdeteksi *SQL Injection.*

2.1.4 *OWASP TOP 10*

OWASP Top 10 adalah sebuah daftar yang diliris oleh komunitas *OWASP* yang berisikan 10 daftar teratas celah keamanan yang dapat mengancam keamanan suatu *website,* daftar ini terus berkembang dan berubah-ubah mengikuti perkembangan teknologi *website* yang terus berkembang. *OWASP Top 10* pertama kali dirilis tahun 2003 lalu update minor pada tahun 2004, 2007, 2010, dan 2017. *OWASP Top 10* dibuat dengan tujuan untuk meningkatkan kesadaran tentang keaman aplikasi dengan mengidentifikasi beberapa risiko celah keamanan yang sering dihadapi atau ditemui dalam banyak kasus (Alexander Dharmawan, 2022).



Gambar 2. 2 Update data *OWASP Top 10* dari tahun 2017-2021

Sumber : (www.owasp.org)

Gambar 2.2 menunjukkan data *OWASP TOP 10* terbaru, yang mencakup 10 masalah keamanan *website* yang paling umum. Di *OWASP TOP 10* 2021, beberapa konsolidasi baru, tiga kategori baru, dan empat kategori yang mengalami perubahan nama dan ruang lingkup (www.owasp.org).

2.1.5 *OWASP ZAP 2.15.0*

Zed Attack Proxy (ZAP) adalah aplikasi *pentest* untuk menemukan *vulnerabilitie*s dalam suatu *web application* dengan cara mudah, *ZAP* menyediakan *scanner* automatis sebaik bila kita menggunakan *tools* untuk menemukan *vulnerabilities* secara manual (Gregorius, 2022).

Berdasarkan keterangan dari *web* resmi *ZAP, tools* ini mendeteksi kerentanan pada suatu *website* dengan beberapa metode utama :

1. *Proxy Intercept – ZAP* bertindak sebagai perantara antara browser dan aplikasi *web,* memungkinkan analisis serta modifikasi lalu lintas *HTTP/S* secara *real-time* untuk mengidentifikasi kelemahan keamanan.
2. Pemindaian Pasif – *ZAP* secara otomatis menganalisis permintaan dan respons *HTTP* tanpa mengirimkan permintaan tambahan ke *server.* Ini berguna untuk mendeteksi masalah seperti informasi sensitif yang terekspos, *header* keamanan yang hilang, atau konfigurasi yang tidak aman.
3. Pemindaian Aktif – *ZAP* mengirimkan permintaan khusus ke aplikasi *web* untuk menguji kerentanan yang lebih kompleks, seperti *SQL Injection, Cross-Site Scripting (XSS),* dan *Remote Code Execution (RCE).*
4. *Spidering (Crawling) – ZAP* menjelajahi aplikasi *web* untuk menemukan semua halaman, *endpoint,* dan parameter yang berpotensi rentan terhadap serangan.
5. *Fuzzing – ZAP* mengirimkan berbagai jenis data acak atau berbahaya ke dalam formulir dan parameter web untuk menguji bagaimana aplikasi menangani input yang tidak diharapkan.

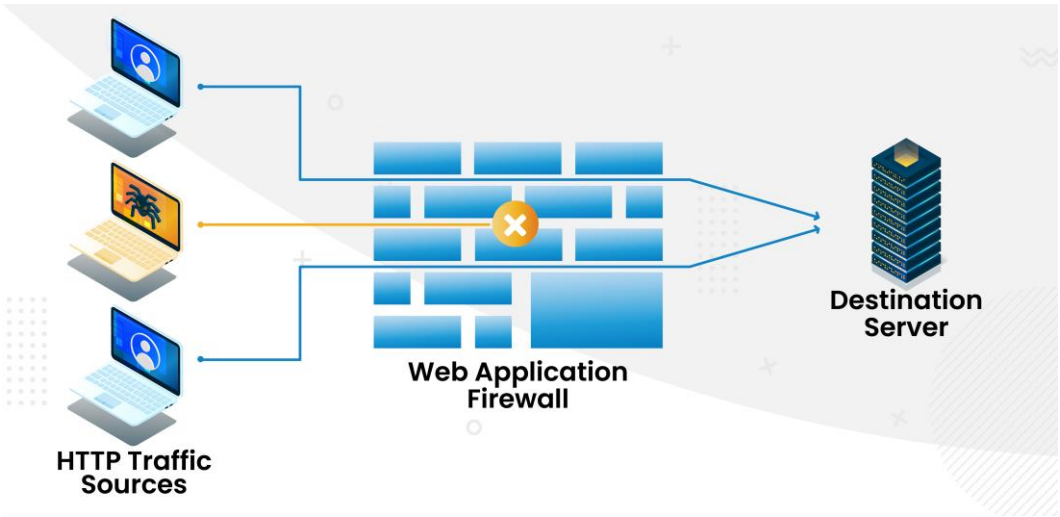
6. *Scripting & Automasi – ZAP* memungkinkan penggunaan *skrip custom* untuk mendeteksi kerentanan spesifik yang mungkin tidak terdeteksi oleh pemindaian standar.
7. Integrasi dengan *CI/CD – ZAP* dapat digunakan dalam *pipeline* pengembangan perangkat lunak *(DevSecOps)* untuk melakukan pengujian keamanan otomatis sejak tahap awal pengembangan aplikasi.

Summary of Alerts adalah ringkasan dari ancaman yang didapatkan melalui scanning menggunakan *OWASP ZAP*, berikut penjelasan tingkat risiko berdasarkan hasil grafik tersebut:

1. *High* (Tinggi) merupakan kerentanan yang sangat serius dan jika dieskploitasi dapat menyebabkan dampak sangat buruk, seperti kebocoran data sensitif atau pengambilalihan sistem.
2. *Medium* (Sedang) menunjukkan kerentanan yang berpotensi menyebabkan kerugian yang signifikan, seperti gangguan layanan atau kerusakan sebagai data.
3. *Low* (Rendah) menunjukkan kerentanan yang memiliki dampak yang kecil, namun tetap perlu diperbaiki untuk meningkatkan keamanan secara keseluruhan.
4. *Informational* merupakan informasi tambahan yang tidak langsung menunjukkan adanya kerentanan, namun perlu diperhatikan untuk meningkatkan keamanan, ini bisa berupa konfigurasi yang tidak optimal atau pratik pengkodean yang kurang baik.

2.1.6 *Web Application Firewall*

Menurut *Web Application Security Consortium (WASC), WAF* merupakan perangkat perantara yang berfungsi antara *web client* dan *web server* dan melakukan analisis pesan pada *layer* 7 *Open Sistem Innerconnection (OSI)* ketika terjadi pelanggara kebijakan keamanan yang telah ditentukan (Alfiani et al., 2021). Menurut (Yanti & et al., 2015) *WAF* dapat diterapkan pada aplikasi yang sudah berjalan karena dapat melakukan konfigurasi tambahan pada *web server* tanpa perlu mengubah *script* pembangun aplikasi. Seperti *firewall* biasa melakukan filter data masuk dan keluar dan memiliki kemampuan untuk menghentinkan *traffic* yang dianggap berbahaya menurut aturan yang ditetapkan.



Gambar 2. 3 Topologi *Web Application Firewall* (Yanti & et al., 2015)

WAF bekerja untuk melindungi aplikasi *web* dengan menyaring, mengawasi, dan memblokir lalu lintas *berbahaya* sebelum mencapai server. *WAF* mendeteksi ancaman *seperti SQL Injection, XSS,* dan *CSRF* dengan menganalis permintaan *HTTP/HTTPS* menggunakan aturan keamanan *(ruleset).* Jika permintaan mencurigakan ditemukan, *WAF* dapat langsung memblokir,

menampilkan *CAPTCHA*, membatasi akses, atau mencatat aktivitas untuk investigasi lebih lanjut (Riska & Alamsyah, 2021).

Pemeriksaan permintaan masuk *(request inspection)* dan pemeriksaan respons keluar *(response inspection)* adalah dua komponen utama proses *WAF*. Pada tahap awal, *WAF* mengidentifikasi pola serangan dengan melihat *header*, parameter *URL*, *payload,* dan *cookies. WAF* akan bertindak sesuai aturan yang diterapkan jika ditemukan tanda-tanda eksploitasi, seperti skrip berbahaya atau perintah *SQL* yang mencurigakan. Pembelajaran mesin dan analisis perilaku juga digunakan oleh beberapa *WAF* kontemporer untuk menemukan ancaman baru yang belum terdaftar dalam database keamanan (Riska & Alamsyah, 2021).

Didasarkan pada bagaimana penggunaannya, *WAF* dibagi menjadi tiga kategori: *Network*-based *WAF* (perangkat keras jaringan), *Host-based WAF* (*software* di server aplikasi), dan *Cloud-based WAF* (layanan berbasis *cloud* seperti *AWS WAF* atau *Cloudflare*). Setiap jenis memiliki kelebihan dan kekurangan tergantung pada infrastruktur dan keamanan yang dibutuhkan. Aplikasi *web* dapat dilindungi dari serangan *siber* dengan menerapkan *WAF* yang dikonfigurasi dengan baik tanpa perlu mengubah kode aplikasi, meningkatkan keamanan dan keandalan layanan online (Widiyono & Oktiawati, 2024).

2.1.7 *ModSecurity*

ModSecurity dikenal sebagai *ModSec*, merupakan aplikasi *WAF* yang dirancang khusus untuk mendukung *Server Web Open Source Apache* dan *Nginx*. *Server ModSecurity* yang diinstal melakukan perlindungan ditingkat aplikasi *web*. *ModSecurity* menyediakan aturan konfigurasi yang disebut *SecRules* untuk

memantau lalu lintas *HTTP*/S secara *real-time* dan untuk mencatat serta memfilter komunikasi *HTTP*/S berdasarkan aturan yang telah ditentukan sebelumnya. *ModSecurity* menghentikan atau memblokir lalu lintas yang dianggap berbahaya atau merugikan situs *web* (Haikal Muhammad et al., 2023).

Core Rule Set (CRS) merupakan kumpulan aturan deteksi serangan yang digunakan oleh *ModSecurity* atau *WAF,* tujuan *CRS* adalah untuk melindungi aplikasi *web* dari berbagai jenis serangan sambil meminimalkan alarm palsu. Serangkaian aturan ini melindungi terhadap serangan yang sering terjadi, seperti *SQL* Injecion dan *XSS*, karena *CRS* dilisensikan di bawah lisensi *Apache* 2.0, ia dirancang untuk melindungi aplikasi *web* dari serangan umum yang dapat mempengaruhi keamanan dan kerentanan aplikasi. Oleh karena itu, *CRS* dapat digunakan dengan bebas dan disesuaikan sesuai kebutuhan (Haikal Muhammad et al., 2023).

2.1.8 *Web Host Manager*

Web Host Manager (WHM) adalah alat control administratif yang menyediakan kemampuan manajemen untuk *server* khusus atau *VPS, WHM* memungkinkan kita untuk membuat, menghapus, atau menangguhkan akun, memantau *server*, mengatur beberapa paket *host*ing, mentransfer file, menyesuaikan merek *reseller*, dan mengelola sertifikat *SSL*. *WHM* juga memungkinkan untuk mengelola semua aktivitas di *server* dan melompat antara *cPanel* dengan mudah (Muhammad Nugi Abdiansyah, 2018).

Menurut (Muhammad Nugi Abdiansyah, 2018) *cPanel* adalah panel control berbasis *web* yang sering digunakan oleh pemilik situs *web* untuk mengelola

berbagai aspek layanan *hosting* tanpa memerlukan pengetahuan teknis yang mendalam tentang administrasi server. *cPanel* salah satu panel kontrol *hosting* paling populer, memiliki antarmuka yang mudah digunakan dan banyak fitur untuk mengelola situs *web*, seperti:

1. Manajemen File: pengguna dapat mengunggah, mengedit, menghapus, serta mengatur izin file dan folder melalui File Manager atau FTP.
2. Manajemen Domain: *cPanel* memungkinkan pengguna untuk menambahkan domain utama, subdomain, dan addon domain, serta mengatur *redirect* dan konfigurasi DNS.
3. Pengelolaan Email : pengguna dapat membuat dan mengelola akun email berbasis domain, mengatur *forwarders, autoresponders*, *filter spam*, serta mengakses email melalui *webmail* atau *klien* email *eksternal.*
4. Manajemen Basis Data : *cPanel* mendukung pembuatan dan pengelolaan *database MySQL* dan *PostgreSQL*, termasuk konfigurasi pengguna dan *akses database.*
5. Keamanan Situs *Web* : terdapat berbagai fitur keamanan seperti pengelolaan *SSL/TLS*, proteksi direktori dengan kata sandi, *IP blocking*, dan pemantauan aktivitas login.
6. Pemantauan Statistik dan Performa : *cPanel* membantu pemilik situs *web* mengoptimalkan kinerja *website* mereka dengan menyediakan laporan tentang penggunaan sumber daya, trafik pengunjung, dan *log error*.

7. Instalasi Aplikasi : pengguna dapat menginstal berbagai aplikasi *web* seperti *WordPress, Joomla*, dan *Drupal* dengan hanya beberapa klik dengan fitur *Softaculous* atau *Fantastico.*

2.1.9 *Penetration Testing* (Uji Penetrasi)

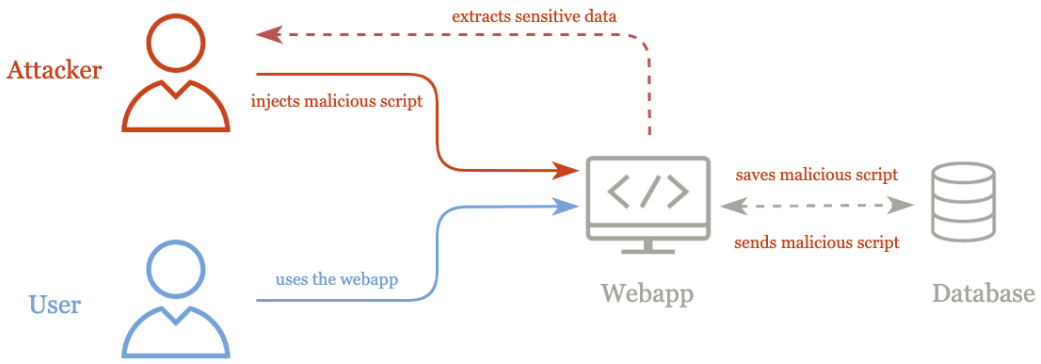
Penetration testing adalah alat penilaian jaminan bernilai penetrasi menguntungkan operasi dan bisnis. *Penetration testing* berguna dalam hal operasional karena membantu dalam pembentukan *strategi* keamanan informasi dengan menemukan kerentanan dengan cepat dan tepat. Informasi rinci tentang ancaman diberikan oleh *penetration testing*, yang digunakan jika tercakup dalam aliran dan proses keamanan organisasi (Hasibuan & Elhanafi, 2022).

Berikut 4 metode penetrasi utama yang dipakai pada pengujian *VA* di penelitian ini adalah sebagai berikut :

A. *Cross-Site Scripting* (*XSS*)

XSS adalah jenis serangan injeksi kode *Javascript* yang memungkinkan penyerangan untuk menjalankan skrip berbahaya di *browser web* korban. Serangan ini dapat mengakibatkan berbagai efek samping seperti kompromi data, pencurian *cookie*, kata sandi, nomor kartu kreadit, dan lain-lain (Gupta & Gupta, 2017).

XSS Reflected adalah salah satu jenis serangan *XSS* di mana skrip berbahaya disisipkan ke dalam permintaan *HTTP* dan kemudian "dipantulkan" kembali oleh *server* ke dalam respons yang dikirimkan ke pengguna, dalam serangan ini skrip jahat tidak disimpan di *server*, melainkan dikirimkan secara langsung melalui *URL* atau parameter permintaan (Gupta & Gupta, 2017).



Gambar 2. 4 Cara Kerja Serangan *XSS* (Putra et al., 2021)

Gambar 2.4 menggambarkan secara visual bagaimana serangan *XSS*, berikut penjelasan gambar:

1. Penyerangan (*Hacking*)

Bertujuan menyusup kode berbahaya *(malicious code)* ke dalam sebuah *website* dengan cara menyuntikkan kode jahat ke dalam data yang diterima oleh *website*, seperti kolom komentar, *formular login*, atau *field* pencarian.

2. Input data

Kode jahat yang disuntikkan oleh penyerang berupa potongan kode *javascript*, contoh kode jahat yang sederhana *: <script>alert('Anda telah diretas!');</script>* , kode ini akan dieksekusi oleh *browser* korban saat halaman *web* dimuat.

3. *Website*

Kerentanan : *website* yang tidak memiliki perlindugan yang cukup terhadap serangan *XSS* akan menyimpan dan menampilkan input pengguna secara langsung tanpa melakukan validasi atau sanitasi terlebih dahulu.

Akibatnya : kode jahat yang disuntikkan akan tersimpan dalam *database website* dan ditampilkan kembali kepada pengguna lain yang mengunjungi halaman tersebut.

4. Korban

Ketika korban mengakses halaman *web* yang telah terkontiminasi oleh kode jahat, *browser* korban akan mengeksekusi kode tersebut. Dampat yang akan muncul yaitu munculnya *pop-up* atau pesan peringatan yang tidak diinginkan, pencuran informasi sensitive seperti *cookie*, token sesi, atau kredensial *login*, pengalihan ke situs *web* palsu (*phising*), dan penyebaran *malware.*

B. *SQL injection*

SQL injection merupakan jenis serangan yang menggunakan kode *SQL* berbahaya untuk memanipulasi basis data *backed* dan mengakses data yang seharusnya tidak dapat diakses. Memanfaatkan kerentanan *SQL injection*, penyerangan dapat melewati mekanisme otentikasi dan otorisasi aplikasi *web* dan mengambil isi seluruh basis data (Krishnan et al., 2021). Menurut (Krishnan et al., 2021) jenis-jenis *SQL injection* ada 3, diantaranya adalah :

1. *Union Based SQL injection*

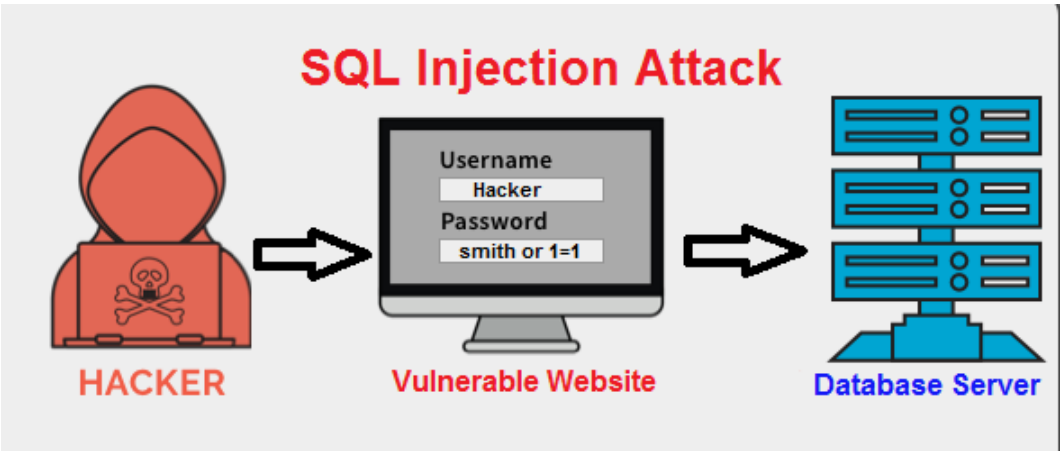
Penyerang menggunakan operator UNION untuk menggabungkan *query* berbahaya dengan *query* asli. Hasil dari *query* berbahaya akan digabungkan dengan hasil *query* asli, memungkinkan penyerang mendapatkan nilai kolom dari tabel lain.

2. *Error Based SQL injection*

Serangan ini dilakukan dengan mengirimkan data yang tidak valid ke dalam query, yang menyebabkan basis data menghasilkan kesalahan. Penyerang kemudian dapat mencari kesalahan yang dibuat oleh basis data dan menggunakan informasi ini untuk maju dalam basis data menggunakan *query SQL*.

3. *Blind SQL injection*

Serangan ini menghasilkan pertanyaan benar atau salah ke basis data dan menemukan solusi berdasarkan respons aplikasi. Serangan ini biasanya diterapkan pada aplikasi *web* yang menunjukkan pesan kesalahan umum tetapi tidak mengurangi kode yang rentan terhadap *SQL injection*.



Gambar 2. 5 Cara Kerja *SQL injection* (Aziz, 2022)

Gambar 2.5 menggambarkan cara kerja *SQL injection*, berikut penjelasan pada gambar:

1. Hacker

Sosok yang melakukan serangan, mereka memanfaatkan celah keamanan pada aplikasi *web* untuk menyusupkan kode berbahaya.

2. *Vulnerable Website*

Memiliki kelemahan dalam *system* keamanannya. Dalam kasus ini, *system* tidak memvalidasi input pengguna dengan benar, sehingga *Hacker* bisa menyuntikkan kode *SQL*.

3. *Database Server*

   Tempat penyimpanan data yang terhubung dengan *website*. Data-data penting seperti informasi pengguna, transaksi, dan lainnya disimpan disini.

4. *Username* dan *Password*

   Bagian yang biasanya diminta saat *login.* Namun, *Hacker* tidak memasukkan *password* yang benar.

5. "*smith or 1=1*"

   Kode *SQL* yang disuntikkan oleh *Hacker*. Kode ini akan membuat *database* mengembalikan semua data, karena kondisi "1=1" selalu benar.

a. Cara kerja serangan *SQL injection* :

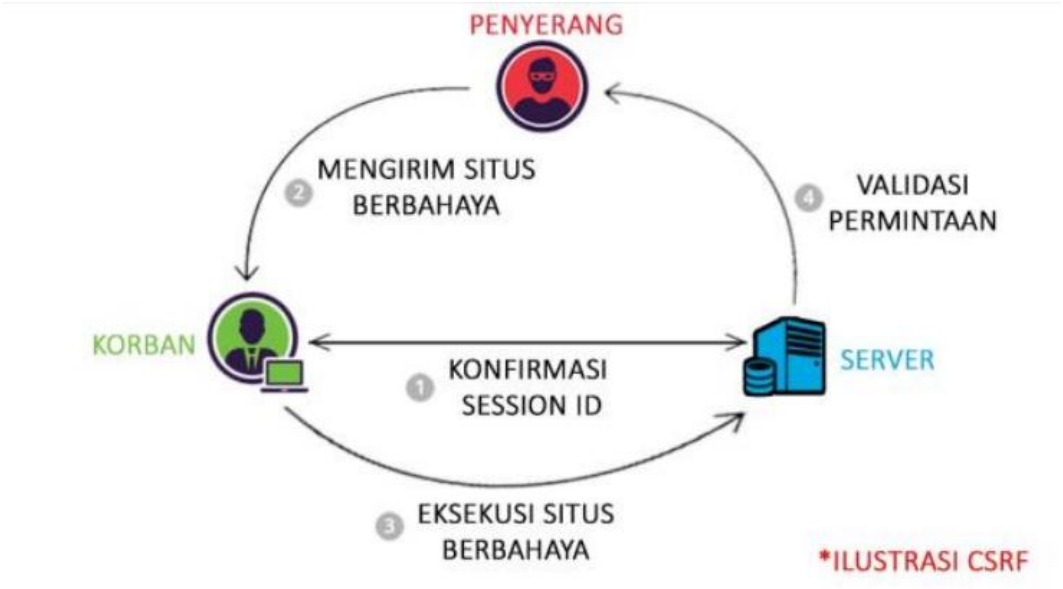
Seorang hacker dapat mengeksploitasi kerentanan pada sebuah aplikasi web dengan mengirimkan input berisi kombinasi username dan password yang telah dimodifikasi menggunakan kode SQL berbahaya. Masalah ini muncul karena aplikasi tidak melakukan validasi input dengan baik, sehingga perintah SQL berbahaya tersebut langsung diteruskan ke database tanpa pemeriksaan. Akibatnya, database akan mengeksekusi perintah yang dimanipulasi tersebut. Misalnya, jika hacker menyisipkan kondisi seperti "1=1" dalam perintah SQL, maka kondisi ini akan selalu bernilai benar. Hal ini menyebabkan database mengembalikan seluruh data pengguna tanpa batasan, sehingga hacker bisa memperoleh akses ke informasi sensitif yang seharusnya tidak dapat mereka lihat.

Cara mencegah SQL Injection dengan memvalidasi input, gunakan *parameterisasi query,* btasi hak akses, dan perbarui aplikasi dan *database.*

C. *Cross-Site Request Forgery (CSRF)*

CSRF merupakan jenis serangan *web* yang memaksa pengguna untuk mengirimkan permintaan *HTTP* yang tidak diinginkan dan dikendalikan oleh penyerang ke aplikasi *web* yang rentan dimana pengguna tersebut seang *terautentikasi.* Permintaan ini tampak sah karena dikirim melalui *browser* pengguna yang sudah *terautentikasi* (Makalalag et al., 2017). Proses serangan *CSRF* sebagai berikut :

1. Pengguna login ke aplikasi web yang sah namun memiliki kerentanan, misalnya jejaring sosial favoritnya. Sebagai contoh, seorang pengguna seperti Alice melakukan login, dan proses otentikasi dilakukan melalui cookie sesi yang secara otomatis akan dilampirkan oleh browser pada setiap permintaan selanjutnya ke aplikasi web tersebut.
2. Setelah login, Alice membuka tab baru di browser dan mengunjungi situs web lain yang tidak terkait, seperti situs berita. Situs tersebut memuat halaman web yang tanpa sepengetahuan Alice menyertakan iklan berbahaya.
3. Iklan berbahaya ini kemudian mengirimkan permintaan lintas situs (cross-site request) ke jejaring sosial tempat Alice telah login, menggunakan HTML atau JavaScript. Misalnya, permintaan tersebut bisa berupa aksi untuk “menyukai” partai politik tertentu. Karena permintaan ini dikirim dari browser yang masih menyimpan cookie sesi milik Alice, maka permintaan tersebut diproses oleh jejaring sosial seolah-olah berasal dari Alice sendiri.



Gambar 2. 6 Gambaran Umum *CSRF* (Deepika, 2024)

Gambar 2.6 menggambarkan sebuah serangan siber yang disebut *Cross-Site Request Forgery (CSRF)*. Berikut penjelasan langkah-langkah serangan *CSRF* :

1. *Konfirmasi session ID*

   Korban telah berhasil ke *login* ke situs *web* yang aman dan mendapatkan *session ID*. *Session ID* ini berfungsi sebagai tanda pengenal sesi pengguna saat berinteraksi dengan dengan situs *web*.

2. Mengirimkan situs berbahaya

   Penyerang membuat sebuah situs *web* palsu atau link berbahaya yang dirancang untuk mengesploitasi kerentanan *CSRF* pada situs *web* target.

3. Eksekusi situs berbahaya

   Korban secara tidak sengaja mengklik link berbahaya yang dibuat oleh penyerangan. *Browser* korban kemudian mengirimkan permintaan ke situs *web* target dengan menyertakan *session ID* yang valid.

4. Validasi permintaan

Situs *web* target menerima permintaan dari *browser* korban dan melakukan validasi terhadap *session ID.* Karena *session ID* yang dikirimkan masih valid, Situs *web* target mempercayai bahwa permintaan tersebut berasal dari pengguna yang sah.

5. *Server* mengeksekusi permintaan

Situs *web* target kemudian mengeksekusi perintah yang terdapat pada link berbahaya tersebut seolah-seolah perintah tersebut berasal dari pengguna yang sah.

2.2 Penelitian Terkait

2.2.1 *State of The Art*

Beberapa penelitian terdahulu yang dilakukan dalam metode *VA* dan *WAF*, berikut penelitian terkait disajikan pada Tabel 2.1

Tabel 2. 1 Penelitian Terkait *(State of the art)*

1.	Nama, tahun peneliti	(Fadila Burhani & Priyawati, 2024)
	Judul	Analisis Pengujian Keamanan *Website* Pengelolaan Internet Desa Kragan Menggunakan Metode *Penetration Testing Execution Standard (PTES)*
	Metode/solusi	Metode *Vulnerability Assessment* dan Metode *PTES*
	Hasil Penelitian	Penelitian ini menemukan bahwa pengujian menggunakan metode *Penetration testing Execution Standard (PTES)* berhasil memberikan hasil 14 celah keamanan dimana diantaranya menjadi bahan ekspolitasi. Ketiga celah keamanan tersebut yaitu *SQL injection*, *Missing Anti-clickjacking Header,* dan Absence of anti-*CSRF* to-kens. Menggunakan metode *PTES* mendapatkan 14 celah keamanan dengan level resiko tinggi, *medium*, dan rendah.
2.	Nama, tahun peneliti	(Wahyudin, 2024)
	Judul	Metode *Vulnerability Assessment* Dalam Pengujian Kinerja Sistem Keamanan *Website Points of Sales*
	Metode/solusi	Metode *VA*
	Hasil Penelitian	Penelitian ini melakukan pengujian kerentanan w*eb* menggunakan metode *VA* untuk mengidentifikasi celah

		kelemahan pada aplikasi berbasis w*eb*, dengan jenis kerentanan seperti *SQL injection*, kerentanan *authentication* dan *acces control.* Hasil dari penelitian ini menunjukkan adanya 10 kerentanan yang terdeteksi, ditemukan 7 kerentanan denga level/Tingkat resiko *medium*.
3.	Nama, tahun peneliti	(Orisa & Ardita, 2021)
	Judul	*Vulnerability Assessment* Untuk Meningkatkan Kualitas Keamanan *Web*
	Metode/solusi	Metode *VA*
	Hasil Penelitian	Peneliti melakukan pengujian pada *Host* target yaitu *POST, OPTIONS, GET,* dan *HEAD* menggunakan metode *VA* dibantu dengan *Network Mapping* (*NMAP*). Hasil peneliti menunjukkan bahwa *VA* dengan *NMAP* juga menunjukkan bahwa target tersebut terdeteksi sebagai *HTTP open proxy*. Kemudian target yang diuji tidak terdeteksi *Cross-Site Scripting* dan *Host* juga tidak terdeteksi *SQL Injection.*
4.	Nama, tahun peneliti	(Mulyanto & Haryanti, 2021)
	Judul	Analisis Keamanan *Website* SMAN 1 Sumbawa Menggunakan Metode *Vulnerability Assessment*
	Metode/solusi	Metode *VA*
	Hasil Penelitian	Penelitian ini melakukan pengujian dengan mengidentifikasi 4 tingkat celah kerentanannya yaitu *high, medium, low,* dan *informational* pada *Website* SMAN 1 Sumbawa, Adapun Tingkat kerentanan *high* yang didapatkan adalah *SQL injection*, dengan kerentanan *SQL injection* memudahkan penyerang mengakses seluruh *database*. Hasil pengujian pada penelitian ini menunjukkan bahwa pada *Website* SMAN 1 Sumbawa memiliki banyak celah kerentanan *VA* bahwa *Website* SMAN 1 Sumbawa masih dalam keadaan tidak aman.
5.	Nama, tahun peneliti	(Rohim & Setiyani, 2023)
	Judul	Analisis Celah Keamanan *E-Learning* Perguruan Tinggi Menggunakan *Vulnerability Assessment*
	Metode/solusi	Metode *VA*
	Hasil Penelitian	Penelitian ini menganalisis dan melakukan pengujian keamanan, kerentanan pada *Website* E-Learning menggunakan metode *VA* dan *Vulnerability scanning* dengan *OWASP tools*, dan berhasil menunjukkan ada 15 kerentanan dengan 3 kategori yaitu 2 kategori *medium*, 8 kategori *low*, dan 5 kategori *informational.*
6.	Nama, tahun peneliti	(Jurnal & Lana Rahardian, 2022a)
	Judul	Analisis Keamanan *Web* New Kuta Golf Menggunakan Metode *Vulnerability Assessment* dan Perhitungan *Security Metriks*
	Metode/solusi	Metode *VA*

	Hasil Penelitian	Hasil penelitian menunjukkan dalam pengujian keamanan w*ebsite* dapat dilakukan dengan metode *VA* menggunakan aplikasi acunetix hingga menemukan hasil yang detail, dan berhasil membuat *Website* new kuta golf menjadi bernilai *high*.
7.	Nama, tahun peneliti	(Mamuriyah et al., 2024)
	Judul	Rancangan Sistem Keamanan Jaringan dari Serangan *DDoS* Menggunakan Metode Pengujian Penetrasi
	Metode/solusi	*WAF*
	Hasil Penelitian	Peneliti ini mengusulkan menggunakan metode penetrasi dengan melibatkan target w*ebsite*, pelaksanaan serangan *DDoS* dengan alat uji, analisis log, dan evaluasi serangan dengan focus pada penggunaan *WAF cloudflare, kali linux,* dan *goldeneye*. Hasil dari metode yang diusulkan menunjukkan bahwa rancangan sistem keamanan ini berhasil mengidentifikasi, mengatasi, dan mengurangi kerentanan terhadap serangan *DDoS*. Pengujian penetrasi membuktikan bahwa *Cloudflare* dapat efektif mendeteksi dan mencegah serangan *DDoS*, sementara kali linux dan *goldeneye* digunakan sebagai alat uji yang andal.
8.	Nama, tahun peneliti	(Ardiansyah et al., 2023)
	Judul	Implementasi Keamanan *Website* Dengan Metode *Firewall Aplikasi Web (WAF)*
	Metode/solusi	*WAF*
	Hasil Penelitian	Hasil dari penelitian yang sudah dilakukan, serangan *SQL injection* dan *DDoS* yang telah diujicobakan pada *website* berhasil diblokir sehingga membuat w*ebsite* menjadi aman dari serangan tersebut.
9.	Nama, tahun peneliti	(Riska & Alamsyah, 2021)
	Judul	Penerapan Sistem Keamanan *Web* Menggunakan Metode *Web Application Firewall*
	Metode/solusi	*WAF*
	Hasil Penelitian	Hasil dari penelitian ini menunjukkan bahwa *firewall* dengan *menggunakan module dan rule ModSecurity* berbasis *WAF* pada system keamanan w*eb* dapat memblokir *SQL injection, Cross-Site Scripting (XSS),* dan *Command Execution* dengan menampilkan pesan *error* kepada *user* yang melakukan perintah tersebut.
10.	Nama, tahun peneliti	(Wiguna et al., 2020)
	Judul	Implementasi *Web Application Firewall* Dalam Mencegah Serangan *SQL injection* Pada *Website*
	Metode/solusi	*WAF*
	Hasil Penelitian	Hasil dari ujicoba serangan peneliti yaitu implementasi *WAF* mampu meminimalisir serangan *SQL injection* dikarenakan dalam *tools ModSecurity* ini sistem akan berfungsi membatasi penyerang untuk mengeksekusi *SQL injection* dalam sebuah

		w*ebsite*, sehingga hasil pengujian menunjukkan penyerang tidak mendapat informasi apapun yang terkandung di dalam w*ebsite* karena di dalam sistem *security* ini mengurangi celah-celah yang terbuka dieksploitasi lebih dalam lagi.

2.3 Matriks Penelitian

Perbedaan dalam metode, kebaruan penelitian, dan objek penelitian terhadap penelitian sebelumnya pada tabel 2.2

Tabel 2.2 Matriks Penelitian

No.	Peneliti	Ruang Lingkup										
				Serangan					Implementasi			
		VA	*WAF*	*XSS*	*SQL injection*	*CSRF*	*DoS*	*DDoS*	*Website*	Jaringan	Sistem Operasi	Layanan *Web*
1.	(Fadila Burhani & Priyawati, 2024)	✓	-	-	✓	✓	-		✓	-	-	-
2.	(Wahyudin, 2024)	✓	-	-	✓	✓	-		✓	-	-	-
3.	(Orisa & Ardita, 2021)	✓	-	✓	✓		-		✓	-	-	-
4.	(Mulyanto & Haryanti, 2021)	✓	-	-	✓	✓	-		✓	-	-	-
5.	(Rohim & Setiyani, n.d.2020)	✓	-	✓	-	✓	-		✓	-	-	-
6.	(Jurnal & Lana Rahardian, 2022)	✓	-	-	✓	-	-		✓	-	-	-
7.	(Mamuriyah et al., 2024)	-	✓	-	-	-	✓			✓	-	-
8.	(Ardiansyah et al., 2023)	-	✓	-	✓	-	✓		✓	-	-	-
9.	(Orisa & Ardita, 2021)	-	✓	✓	✓	-	-		✓	-	-	-
10.	(Wiguna et al., 2020)	-	✓	-	✓	-	-		✓	-	-	-
11.	(Narhudin et al., 2024)	-	✓	✓	✓	-	-		-	-	-	-
12.	(Ardiansyah et al., 2023)	✓	✓	-	✓	-	-	✓	✓	-	-	-
13.	Usulan Penelitian	✓	✓	✓	✓	✓	-		✓	-	-	-

Tabel 2.2 merupakan matriks penelitian yang menunjukkan perbandingan berbagai pendekatan *VA* dan *WAF.* Penelitian ini berfokus pada pengujian keamanan terhadap *website* pengelolaan internet milik Desa Kragan dengan menggunakan metode *Penetration Testing Execution Standard (PTES)*. Pengujian ini menggunakan 4 *tools* yaitu *woish, zenmap, Owasp ZAP*, dan *SQLMap*. Dilakukan secara menyeluruh melalui lima tahapan, yaitu: *information gathering, threat modelling, vulnerability scanning, exploitation, dan reporting*. Tahap pemindaian, digunakan *tools OWASP ZAP* untuk mengidentifikasi celah keamanan. Hasilnya ditemukan 14 kerentanan, termasuk beberapa kerentanan kritis seperti *Cloud Metadata Exposure, SQL Injection, Absence of Anti-CSRF Tokens,* dan *Missing Anti-clickjacking Header*. Beberapa di antaranya berhasil dieksploitasi, sementara *SQL Injection* gagal dieksploitasi karena sistem telah dilindungi oleh WAF dan SSL. Penelitian ini tidak ada penilaian kuantitatif skor resiko dan tidak ada *before-after* perbaikan setelah rekomendasi perbaikan diterapkan (Fadila Burhani & Priyawati, 2024). Penelitian ini membahas penerapan *WAF* sebagai sistem keamanan pada *website* wantilandesa.id metode eksperimen, yaitu menguji sistem keamanan sebelum dan sesudah *WAF* diterapkan. Pengujian mencakup serangan menggunakan *SQL Injection, DDoS*, serta pemindaian kerentanan dengan OWASP ZAP. Hasil pengujian menunjukkan bahwa sebelum *WAF* diterapkan, *OWASP* menemukan 20 jenis celah keamanan, dan *website* berhasil diserang menggunakan *DDoS.* Setelah *WAF* diaktifkan, jumlah celah berkurang

menjadi 10, dan serangan *DDoS* tidak lagi berhasil. Penggunaan *WAF* juga dikombinasikan dengan pemantauan menggunakan *Wireshark,* serta pengamanan tambahan seperti *SSL.* Penelitian ini belum adanya eksploitasi langsung terhadap celah keamanan yang ditemukan oleh *OWASP,* sehingga tidak bisa dipastikan seberapa besar dampaknya jika celah tersebut dimanfaatkan penyerang. Penilaian risiko juga belum menggunakan standar kuantitatif yang seharusnya bisa menunjukkan tingkat keparahan tiap kerentanan secara lebih objektif. Selain itu, penelitian hanya fokus pada satu metode pengamanan yaitu WAF, tanpa membandingkan efektivitasnya dengan metode keamanan lainnya. Hal ini membuat hasil penelitian kurang variatif dan terbatas dalam memberikan gambaran perlindungan keamanan secara menyeluruh (Ardiansyah et al., 2023).