DOI: 10.33197/justinfo.vol1.iss1.2023.1200 ISSN: 3031-3449

# Strategi Pengamanan *Front-end* dalam Pengembangan *Website*

**[1]Steven, [2]Wahyu Rifaldi, [3]Ucu Nugraha**
[1,2]Sistem Informasi, Fakultas Teknik, Universitas Widyatama, Bandung, Indonesia, ucu.nugraha@widyatama.ac.id

**ABSTRACT**

*Front-end design in the context of a website involves all elements that can be seen and accessed by users, from visual displays to user interface interactions. The visual and interactive side is the main focus, but security in the background has a crucial role in protecting data and preventing unauthorized access by irresponsible parties or attempts to hack the website. Front-end security functions protect components on the website from various external attacks and threats. This includes efforts to prevent unauthorized access, protect against hacking, and safeguard against the theft of sensitive data. Even so, there are still many front-end developers who focus more on the visual aspect without understanding the importance of security integration. This lack of understanding can have a negative impact on the security level of a website. Therefore, this research aims to provide information and understanding regarding security practices on the front-end side of a website. With the hope that front-end developers can become more proactive and competent in ensuring the security of the websites they develop so that they can provide a safe and reliable user experience.*

***Keywords:*** *Security, Front-end, Website*

**ABSTRAK**

Front-end dalam konteks website melibatkan semua elemen yang dapat terlihat dan diakses oleh pengguna, mulai dari tampilan visual hingga interaksi antarmuka pengguna. Sisi visual dan interaktif ini menjadi fokus utama, namun keamanan di latar belakang memiliki peran krusial dalam melindungi data dan mencegah akses yang tidak sah oleh pihak yang tidak bertanggung jawab atau berupaya meretas website. Keamanan front-end berfungsi untuk melindungi komponen-komponen dalam website dari berbagai serangan dan ancaman eksternal. Ini mencakup upaya pencegahan akses yang tidak sah, perlindungan terhadap peretasan, dan pengamanan terhadap pencurian data sensitif. Meskipun begitu, masih banyak front-end developer yang lebih fokus pada aspek visual tanpa memahami pentingnya integrasi keamanan. Kurangnya pemahaman ini dapat berdampak negatif pada tingkat keamanan suatu website. Oleh karena itu, penelitian ini bertujuan untuk memberikan informasi dan pemahaman terkait praktik-praktik menjaga keamanan di sisi front-end suatu website. Dengan harapan bahwa front-end developer dapat menjadi lebih proaktif dan kompeten dalam memastikan keamanan website yang mereka kembangkan, sehingga dapat memberikan pengalaman pengguna yang aman dan terpercaya.

**Kata Kunci:** Keamanan, Front-end, Website

## 1. PENDAHULUAN

*Website* semakin umum dimiliki dan digunakan dalam era saat ini, melibatkan hampir semua bidang yang ada[1]. Perusahaan, organisasi, merek, dan pemerintah memiliki kehadiran daring masing-masing, yang digunakan untuk memenuhi kebutuhan individu atau kelompok tersebut. Selain sebagai representasi identitas, *website* juga menjadi alat yang mempermudah berbagai sektor industri sebagai sumber utama pendapatan. Dukungan dari perkembangan teknologi yang sangat pesat mendorong setiap orang untuk memiliki pemahaman dan kepemilikan *website* sendiri, mengingat peningkatan penggunaan teknologi dan *internet* secara global. Suatu website yang berkualitas pastinya dilengkapi dengan komponen-komponen krusial seperti *front-end*, *back-end*[2], *database*, *server*, *framework*, API (*Application Programming Interface*)[3], SSL (*Secure Socket Layer*)[4], dan tentu saja, sistem keamanan. Keamanan dalam konteks *website* menjadi aspek yang sangat vital, berfungsi untuk melindungi *website* dari pengaruh dan serangan pihak luar yang berpotensi meretas situs tersebut. Saat ini, banyak oknum yang memiliki kemampuan untuk meretas *website*, baik itu *website* perusahaan, pribadi, maupun pemerintah. Oleh karena itu, penting bagi para pengembang *website* untuk memahami dan menerapkan praktik keamanan pada setiap *website* yang mereka kelola.

Keamanan dalam suatu *website* menghadirkan kompleksitas yang dapat dikategorikan berdasarkan komponennya, melibatkan *front-end*, *back-end*, *database*, dan *server*[2]. Di antara aspek-aspek

tersebut, keamanan pada front-end memegang peran sentral dan menjadi kunci utama dalam memitigasi risiko yang dapat terjadi[5]. Pentingnya penerapan keamanan *front-end* tidak bisa diabaikan, terutama mengingat berbagai ancaman yang dapat mengintai, seperti SQL *Injection*, *Cross-Site Scripting* (XSS)[6], serangan DDoS[7], dan *Cross-Site Request Forgery* (CSRF)[8]. Pada tahap ini, fokus utama adalah melindungi tampilan visual dan elemen interaktif yang diakses langsung oleh pengguna. Tindakan keamanan pada front-end mencakup langkah-langkah proaktif untuk mencegah eksploitasi potensial yang dapat merugikan keutuhan *website*. Oleh karena itu, pemahaman mendalam tentang metode perlindungan terhadap ancaman tertentu sangat penting. Selain itu, kesadaran akan risiko yang mungkin timbul dari ketidakamanan *front-end* menjadi dasar penting bagi pengembang website. Dengan menerapkan praktik-praktik keamanan yang efektif, pengembang *front-end* dapat memberikan lapisan pertahanan yang solid dan dapat diandalkan, menciptakan lingkungan yang aman bagi pengguna dan integritas data *website*[9].

Keamanan *front-end* memegang peranan utama dalam melindungi komponen-komponen di dalam suatu website dari potensi serangan atau peretasan pihak luar[10]. Penerapan keamanan pada sisi *front-end* tidak hanya menjadikan *website* terhindar dari ancaman, tetapi juga menciptakan lapisan pertahanan yang kokoh. Banyak insiden pencurian data dan peretasan *server* terjadi akibat kekurangan keamanan pada *front-end*, menandakan urgensi dan relevansinya. Oleh karena itu, penerapan keamanan di sisi *front-end* menjadi faktor kunci untuk memastikan bahwa *website* yang dibangun memiliki tingkat keamanan yang optimal[11]. *Front-end* developer memiliki peran sentral dalam memperkuat keamanan pada sisi *front-end*. Beberapa langkah yang dapat diambil melibatkan peningkatan validasi pada inputan formulir, penyaringan inputan yang mengandung karakter khusus, pembatasan jumlah permintaan dari pengguna dalam periode waktu tertentu, implementasi verifikasi middleware, dan penerapan token sesi. Kesadaran akan praktik-praktik ini dapat memitigasi risiko yang mungkin timbul dan memberikan perlindungan yang lebih baik terhadap potensi ancaman. Oleh karena itu, penelitian ini bertujuan untuk memberikan panduan praktis bagi front-end developer dalam menjaga keamanan suatu website dengan fokus pada sisi *front-end*[12].

Tujuan penelitian ini adalah untuk mengidentifikasi, menganalisis, dan merumuskan strategi pengamanan *front-end* yang efektif dalam pengembangan *website*. Penelitian ini bertujuan untuk memberikan wawasan mendalam terkait praktik-praktik terbaik dalam keamanan *front-end*, dengan fokus pada pencegahan potensi ancaman seperti SQL *Injection*, *Cross-Site Scripting* (XSS), serangan DDoS, dan *Cross-Site Request Forgery* (CSRF). Dengan tujuan ini, diharapkan penelitian dapat memberikan panduan praktis dan pemahaman yang lebih baik bagi pengembang *front-end* dalam mengimplementasikan strategi keamanan yang optimal dalam lingkungan pengembangan *website*.

## 2. TINJAUAN PUSTAKA

*Front-end* dalam sebuah website merupakan aspek yang sangat vital karena menjadi antarmuka langsung antara pengguna dan sistem[13]. Tampilan *front-end* mencakup elemen visual dan fungsionalitas yang memungkinkan pengguna berinteraksi dengan *website*, seperti menampilkan informasi, mengedit data, mengunggah *file*, dan berbagai aktivitas lainnya. Pentingnya *front-end* dalam menyajikan pengalaman pengguna yang baik tidak dapat diabaikan, dan itulah sebabnya pengembangan *front-end* sering menggunakan teknologi seperti *Hypertext Markup Language* (HTML), *Cascading Style Sheets* (CSS)[14], dan *Javascript*. Ketiga bahasa atau tools ini menjadi fondasi utama dalam menciptakan tampilan yang menarik dan fungsional dalam suatu website. HTML digunakan untuk menentukan struktur dan konten dari halaman web, CSS bertanggung jawab untuk mengatur tata letak dan gaya visual, sedangkan Javascript memberikan fungsi interaktif yang dinamis. Keberhasilan implementasi *front-end* juga bergantung pada desain yang responsif, memastikan tampilan yang optimal di berbagai perangkat dan ukuran layar. Oleh karena itu, pengembang front-end harus memahami tidak hanya aspek estetika, tetapi juga performa dan responsivitas agar dapat menciptakan pengalaman pengguna yang memuaskan.

*Hypertext Markup Language* (HTML) memainkan peran utama dalam pengembangan website dengan menyediakan struktur dan konten dasar. Meskipun bukan bahasa pemrograman konvensional, HTML

adalah bahasa markah yang dirancang untuk memberikan struktur pada konten web. Dengan menggunakan sintaks-sintaksnya, pengembang dapat membuat tampilan yang dapat diinterpretasi dan ditampilkan oleh browser. HTML memberikan dasar untuk menyusun elemen-elemen halaman web, seperti teks, gambar, dan tautan, yang memungkinkan pengguna untuk berinteraksi dengan konten secara efektif. Selain HTML, *Cascading Style Sheets* (CSS) digunakan untuk mengatur tata letak dan gaya visual dari elemen-elemen yang telah ditentukan dalam HTML. Dengan demikian, HTML bekerja bersama CSS untuk memberikan presentasi yang menarik dan terstruktur[14]. Penggunaan HTML sebagai bahasa markah fundamental dalam pengembangan web menunjukkan betapa pentingnya struktur dasar untuk menciptakan tampilan yang kohesif dan mudah dipahami oleh pengguna.

*Javascript* menjadi bahasa pemrograman yang mendasar dalam upaya membuat website lebih dinamis dan interaktif. Fungsinya tidak terbatas pada manipulasi objek-objek di HTML untuk merespons aksi pengguna, tetapi juga memungkinkan pembangunan fitur-fitur yang responsif dan menarik. Penggunaan *Javascript* tidak hanya terbatas pada pengembangan *front-end*; saat ini, bahasa pemrograman ini telah merambah ke pengembangan back-end, terutama dengan memanfaatkan lingkungan yang disediakan oleh Node.js. Dengan adanya Node.js, pengembang dapat menggunakan Javascript di sisi *server* untuk membangun aplikasi web yang berjalan efisien dan menyeluruh, menunjukkan fleksibilitas dan peran yang semakin berkembang dari bahasa ini dalam dunia pengembangan web.

Regex atau *Regular Expression* adalah sebuah pola yang digunakan untuk memanipulasi string atau teks berdasarkan karakter-karakter yang telah ditentukan dalam suatu class. Pada konteks pemrograman, Regex dapat diaplikasikan untuk berbagai tujuan, termasuk pencarian dan manipulasi teks dalam string. Fungsionalitas Regex meliputi kemampuan untuk mencari kecocokan pola tertentu dalam suatu string, mengganti atau menimpa teks yang sesuai dengan pola tertentu, serta menyaring atau mengekstrak teks dalam string sesuai dengan pola yang telah ditentukan. Kelebihan Regex terletak pada kemampuannya dalam menangani pola-pola kompleks dan memberikan fleksibilitas ekstra dalam memanipulasi data teks dengan cara yang spesifik dan terstruktur.

*Middleware web* adalah lapisan perangkat lunak yang berfungsi sebagai filter atau penyaring pada jalur komunikasi antara pengguna dan sisi *server* dalam pengembangan web. Fungsinya melibatkan validasi autentikasi, pemeriksaan input data, dan authorization sebelum meneruskan request ke sisi server. Dengan adanya *middleware*, tercipta suatu gerbang keamanan yang mampu memeriksa dan memvalidasi token autentikasi yang dikirimkan bersamaan dengan request pengguna. Hal ini berarti hanya request yang telah terverifikasi dan memenuhi syarat autentikasi yang akan diteruskan ke sisi *server* untuk diproses lebih lanjut. Dengan demikian, *middleware* menjadi bagian kritis dalam menjaga keamanan aplikasi web dengan memastikan bahwa setiap interaksi pengguna dengan *server* melewati proses validasi yang ketat.

*Structured Query Language* atau SQL adalah bahasa pemrograman yang secara luas digunakan untuk memanipulasi dan mengelola data dalam *Relational Database Management System* (RDBMS). Dengan SQL, pengguna dapat membuat, mengubah, dan menghapus data yang terdapat dalam database. SQL menyediakan sintaks atau perintah-perintah yang digunakan untuk melakukan berbagai operasi terhadap data, yang sering disebut sebagai *query*. Dengan *query*, pengguna dapat mengekstrak informasi tertentu dari *database*, mengubah nilai-nilai data, serta menghapus atau menambahkan data baru sesuai dengan kebutuhan aplikasi atau sistem yang sedang dikembangkan. Pentingnya SQL dalam pengembangan aplikasi atau sistem database terletak pada kemampuannya untuk menyediakan antarmuka standar yang efektif untuk berinteraksi dengan basis data relasional. Melalui sintaks yang jelas dan struktur yang terorganisir, SQL memungkinkan pengembang untuk melakukan tugas-tugas kompleks terkait pengelolaan data dengan mudah dan efisien. Dengan demikian, SQL menjadi bagian integral dari proses pengembangan perangkat lunak yang memerlukan manajemen data yang handal dan efektif.

*Cross-site Scripting* (XSS) adalah bentuk serangan keamanan siber yang dilakukan dengan cara menyisipkan atau mengunggah skrip atau kode berbahaya ke dalam halaman web yang dituju.

Serangan ini terjadi ketika aplikasi web tidak memadai melindungi diri dari input pengguna yang tidak sah atau tidak terpercaya. Dalam serangan XSS, penyerang memanfaatkan celah tersebut untuk menyisipkan skrip yang kemudian dieksekusi oleh browser pengguna, yang dapat merugikan integritas dan keamanan website serta membahayakan privasi pengguna yang mengakses halaman tersebut. Serangan XSS dapat dilakukan melalui berbagai cara, termasuk menyisipkan skrip pada formulir input, kotak komentar, atau URL. Ketika pengguna mengakses halaman web yang terinfeksi, skrip berbahaya tersebut akan dijalankan tanpa sepengetahuan mereka. Dampak dari serangan XSS dapat beragam, mulai dari mencuri informasi pribadi pengguna, seperti sesi login, hingga mengarahkan pengguna ke halaman palsu yang mungkin menyebabkan kerugian finansial atau kerugian data lainnya. Oleh karena itu, perlindungan terhadap serangan XSS menjadi sangat penting dalam pengembangan web guna memastikan keamanan dan privasi pengguna terjaga dengan baik.

*Distributed Denial of Service* (DDoS) merupakan serangan siber yang ditujukan untuk membuat sebuah layanan online tidak tersedia dengan cara mengakibatkan gangguan pada *server* yang diakses oleh banyak komputer atau perangkat yang terdistribusi. Dalam serangan ini, pelaku DDoS menggunakan jaringan yang terdiri dari banyak perangkat, seringkali disusun dalam *botnet*, untuk secara bersama-sama menyerang target yang sama. Metodenya melibatkan banjir lalu lintas internet ke server atau infrastruktur jaringan, menyebabkan penurunan kinerja dan bahkan kegagalan layanan yang bersangkutan. Serangan DDoS dapat menyebabkan kerugian signifikan bagi *website* atau layanan online, seperti menurunkan kecepatan akses, menghentikan akses pengguna yang sah, atau bahkan mengakibatkan kerusakan permanen pada sistem. Pelaku DDoS sering kali memanfaatkan sejumlah besar perangkat terdistribusi untuk memperkuat serangan mereka, membuatnya sulit untuk diidentifikasi dan ditanggulangi. Oleh karena itu, perlindungan terhadap serangan DDoS menjadi kritis dalam memastikan ketersediaan dan keandalan layanan online, termasuk menerapkan solusi keamanan dan mitigasi yang efektif.

*Cross-site Request Forgery* (CSRF) merupakan serangan keamanan yang memanfaatkan otentikasi yang sudah terverifikasi dari seorang pengguna dalam sebuah *website*. Dalam skenario serangan CSRF, penyerang membuat situs palsu atau menyisipkan skrip berbahaya pada situs yang sah untuk mengeksploitasi otentikasi pengguna yang sedang aktif. Dengan cara ini, penyerang dapat membuat permintaan HTTP yang tidak sah atas nama pengguna yang sah tanpa sepengetahuan atau persetujuannya. Dampak serangan CSRF dapat bervariasi, mulai dari mengubah pengaturan akun hingga melakukan transaksi finansial yang merugikan pengguna. Serangan CSRF mengandalkan fakta bahwa browser pengguna biasanya menyertakan otentikasi cookie saat membuat permintaan ke situs yang terpercaya. Oleh karena itu, pencegahan CSRF umumnya melibatkan penggunaan token *antiforgery* yang disematkan dalam formulir atau permintaan HTTP. Dengan cara ini, setiap permintaan yang diajukan harus menyertakan token yang valid, yang dapat dihasilkan oleh *server*. Dengan menerapkan langkah-langkah pencegahan ini, pengembang dapat mengurangi risiko dan melindungi pengguna dari potensi penyerangan CSRF.

## 3. METODE PENELITIAN

Metode penelitian yang diadopsi dalam penelitian ini adalah metode penelitian deskriptif, yang memungkinkan peneliti untuk mengamati dan mendeskripsikan objek penelitian secara rinci. Pendekatan deskriptif digunakan untuk memberikan gambaran menyeluruh tentang keamanan suatu website. Penelitian ini bertujuan untuk mengidentifikasi celah keamanan yang mungkin ada, mengamati ancaman-ancaman yang potensial, dan menawarkan solusi antisipatif untuk meningkatkan keamanan *website*. Pengumpulan data dalam penelitian ini sangat penting untuk memvalidasi temuan dan mencapai tujuan penelitian. Oleh karena itu, teknik pengumpulan data yang diterapkan melibatkan observasi dan dokumentasi. Melalui observasi, peneliti dapat mengamati secara langsung situasi keamanan website dan mencatatnya. Selain itu, pengumpulan data juga melibatkan teknik dokumentasi, di mana peneliti mengumpulkan literatur terkait dari berbagai sumber seperti catatan transkrip, buku, surat, jurnal, dan lainnya. Dengan menggabungkan kedua teknik ini, diharapkan penelitian dapat memberikan pemahaman yang mendalam tentang keamanan website dan solusi yang efektif untuk mengatasi potensi ancaman[15].

## 4. HASIL DAN PEMBAHASAN

*Website* adalah suatu perangkat lunak atau aplikasi yang dapat diakses melalui *browser*, baik itu di perangkat seluler maupun *desktop*, memerlukan koneksi internet untuk mengakses halaman web tersebut. Saat ini, perkembangan *website* melibatkan dua sisi utama, yaitu back-end dan front-end. Bagian *back-end* berfungsi sebagai *server* yang melayani request yang dihasilkan oleh pengguna, mengelola logika, dan mengakses *database*. Sementara itu, *front-end* bertanggung jawab untuk menampilkan data kepada pengguna dan mengirimkan *request* yang berasal dari tindakan pengguna. *Back-end* dan *front-end* saling berinteraksi untuk menciptakan pengalaman pengguna yang menyeluruh. *Back-end* mengelola proses di balik layar, seperti logika bisnis dan penyimpanan data, sementara *front-end* bertanggung jawab untuk membuat tampilan yang interaktif dan menarik bagi pengguna. Kolaborasi antara kedua sisi ini memastikan bahwa *website* berfungsi secara efektif dan menyajikan informasi atau layanan dengan cara yang mudah dipahami dan diakses oleh pengguna.

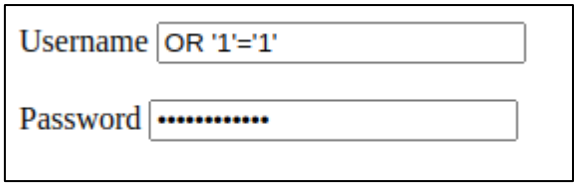
Perkembangan teknologi *website* membawa dampak positif dengan memberikan pengguna kemampuan untuk berinteraksi lebih luas. Pengguna sekarang dapat dengan mudah melakukan operasi *Create, Read, Update,* dan *Delete* (CRUD) pada data, berbagi informasi dengan sesama pengguna, dan menyimpan file pribadi di *platform* web. Hal ini menciptakan lingkungan yang lebih terbuka dan terkoneksi, memungkinkan pengguna untuk lebih aktif terlibat dalam kegiatan *online*. Tidak hanya itu, evolusi teknologi *website* juga mempermudah proses pembayaran dengan metode yang semakin efisien. Dengan hanya satu kali klik, pengguna dapat melakukan pembayaran secara cepat dan aman langsung melalui platform web. Integrasi fitur-fitur seperti ini tidak hanya meningkatkan kenyamanan pengguna tetapi juga membuka peluang baru untuk inovasi dalam pengembangan *website*.

Perkembangan fitur-fitur pada web memberikan keleluasaan bagi pengguna untuk berinteraksi lebih dinamis. Namun, keleluasaan ini juga membawa risiko dengan munculnya ancaman yang dilakukan oleh pihak yang tidak bertanggung jawab. Mereka memanfaatkan celah keamanan untuk mencuri informasi atau melakukan transaksi yang tidak sah, dengan melakukan injeksi dari *front-end* sebuah website ke *server*. Ancaman-ancaman keamanan yang dapat muncul pada *front-end* web meliputi berbagai teknik, seperti SQL *Injection*, *Cross-Site Scripting* (XSS), *Distributed Denial of Service* (DDoS), dan *Cross-Site Request Forgery* (CSRF). Keberagaman ancaman ini menunjukkan kompleksitas tantangan yang dihadapi dalam menjaga keamanan front-end sebuah website. Oleh karena itu, pengembang perlu memahami dan mengimplementasikan langkah-langkah keamanan yang efektif untuk melindungi data dan transaksi pengguna dari potensi risiko yang dapat merugikan integritas dan keamanan sebuah *website*.

### SQL *Injection*

SQL *Injection* merupakan teknik serangan yang dilakukan dengan cara memasukkan *query* secara tidak sah ke dalam formulir atau input yang tersedia di suatu *website*. Dalam serangan ini, *query* yang dimasukkan akan dieksekusi oleh server, mengakibatkan akses tidak sah atau bahkan pencurian dan penghapusan data dari *database*. Dampak dari serangan ini sangat serius, dapat merugikan pengguna yang terkait dengan data yang dicuri, sekaligus merugikan pemilik website yang harus menanggung dampak reputasi dan kepercayaan pengguna yang terganggu.

Penting untuk menyadari bahwa serangan SQL *Injection* dapat menyebabkan kerugian finansial dan merusak reputasi. Pengembang dan pemilik website harus aktif melibatkan langkah-langkah keamanan untuk melindungi sistem mereka dari potensi serangan. Dengan memahami cara SQL *Injection* bekerja dan melibatkan tindakan pencegahan, pengembang dapat meminimalkan risiko dan menjaga integritas data di dalam database mereka. Contoh SQL *injection* pada form HTML dapat memberikan wawasan praktis tentang bagaimana serangan tersebut dapat terjadi dan mengapa langkah-langkah keamanan harus diterapkan dengan sungguh-sungguh.

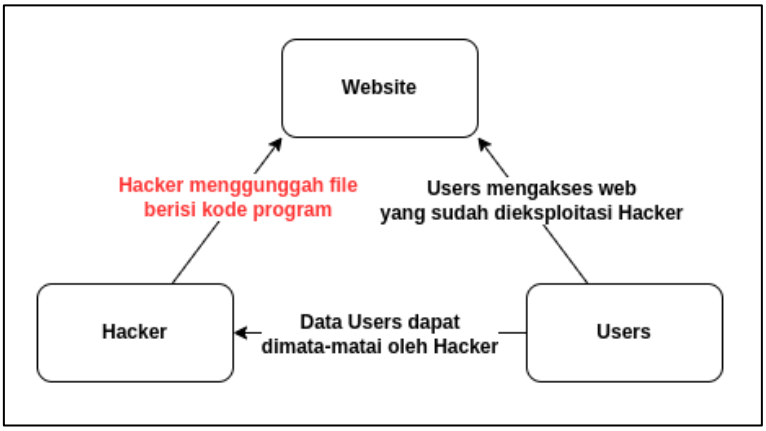



**Gambar 1. Contoh SQL *Injection***

### *Cross-Site Scripting* (XSS)

*Cross-site Scripting* (XSS) adalah bentuk serangan yang dilakukan dengan menyisipkan file berisi kode program ke dalam suatu *website*, memungkinkan pelaku untuk mengeksploitasi *website* tersebut. Umumnya, *file* yang diunggah oleh pelaku memiliki tipe *file* seperti .js atau .php, yang berisi skrip yang dapat dijalankan oleh *browser* pengguna. Serangan XSS dapat memungkinkan pelaku untuk mencuri informasi pengguna, menyebabkan perubahan pada tampilan halaman web, atau bahkan melakukan tindakan yang merugikan bagi pengguna yang mengakses web yang terkena dampak.

Penting untuk diingat bahwa serangan XSS dapat berdampak serius pada keamanan dan privasi pengguna. Oleh karena itu, pengembang web harus mengimplementasikan langkah-langkah keamanan, seperti validasi input, penyaringan karakter khusus, dan pembatasan eksekusi skrip, untuk melindungi website dari potensi serangan XSS. Memahami cara kerja serangan ini dan menerapkan praktik-praktik keamanan yang tepat adalah kunci untuk menjaga integritas *website* dan melindungi data serta informasi pengguna dari ancaman yang mungkin muncul.

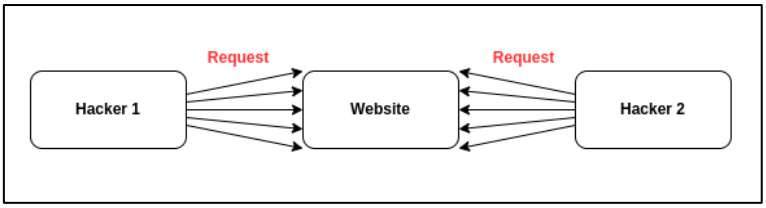



**Gambar 2. Ilustrasi Serangan XSS**

### Serangan *Distributed Denial of Service* (DDoS)

*Distributed Denial of Service* (DDoS) adalah jenis serangan yang dilakukan oleh pelaku dengan meningkatkan jumlah *traffic request* pada suatu website korban. Serangan ini bertujuan membuat *website* tersebut mengalami penurunan kecepatan atau bahkan kegagalan dalam merespons permintaan pengguna yang sah. DDoS bekerja dengan cara membanjiri server atau infrastruktur jaringan dengan sejumlah besar permintaan, sehingga sumber daya yang tersedia habis digunakan untuk merespons permintaan palsu. Akibatnya, website yang menjadi target serangan DDoS dapat mengalami downtime, yang dapat menyebabkan kerugian finansial dan merugikan reputasi web tersebut.

Pentingnya untuk melindungi *website* dari serangan DDoS dengan menerapkan solusi keamanan yang tepat. Penggunaan teknologi dan perangkat lunak yang dapat mendeteksi dan merespons serangan DDoS secara otomatis dapat membantu meminimalkan dampak serangan tersebut. Selain itu, perusahaan atau organisasi yang memiliki website yang rentan terhadap serangan DDoS sebaiknya memiliki strategi pemulihan yang cepat untuk mengurangi dampak dari *downtime* yang mungkin terjadi. Dengan memahami risiko DDoS dan mengambil langkah-langkah proaktif untuk melawan serangan ini, pengelola website dapat memastikan bahwa layanan mereka tetap berjalan lancar dan aman dari gangguan yang tidak diinginkan.

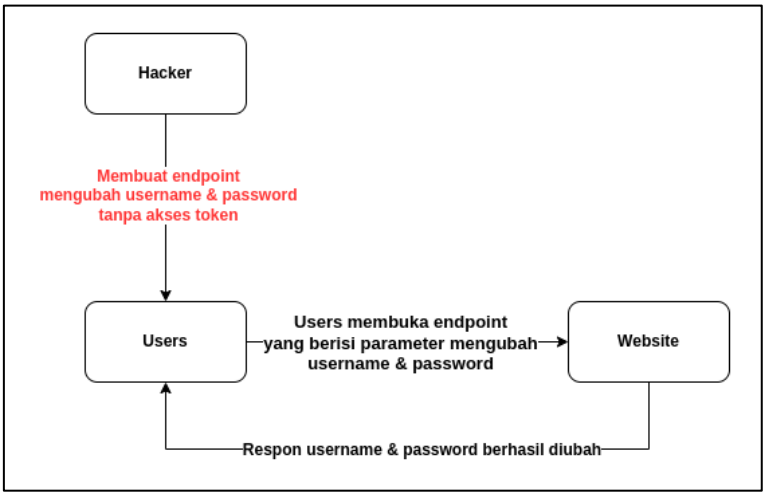



**Gambar 3. Ilustrasi Serangan DDoS**

### *Cross-site Request Forgery* (CSRF)

*Cross-site Request Forgery* (CSRF) merupakan serangan yang dilakukan dengan memanfaatkan kelemahan keamanan pada request antara *front-end* dan *back-end* tanpa adanya validasi token. Dalam serangan ini, pelaku dapat mengeksploitasi celah tersebut untuk menggunakan identitas seseorang tanpa izin dan melakukan transaksi yang tidak sah. Sebagai contoh, ketika seorang pengguna sedang masuk ke akun banknya dan secara bersamaan membuka halaman web yang telah dimanipulasi oleh pelaku, maka transaksi yang dilakukan di halaman tersebut dapat menggunakan identitas pengguna yang sudah login.

Pentingnya menerapkan validasi token untuk melindungi website dari serangan CSRF sangatlah krusial. Dengan menggunakan token, setiap *request* yang masuk dapat diverifikasi keabsahannya, sehingga aksi yang dilakukan hanya dapat berasal dari sumber yang sah dan terotorisasi. Selain itu, penggunaan teknologi keamanan seperti *SameSite Cookies* juga dapat membantu mengurangi risiko serangan CSRF dengan membatasi pengiriman cookies hanya pada *request* yang berasal dari domain yang sama. Dengan memahami cara kerja dan risiko CSRF, pengembang web dapat mengimplementasikan tindakan pencegahan yang efektif untuk menjaga keamanan front-end dan melindungi pengguna dari potensi eksploitasi yang merugikan.




**Gambar 4. Ilustrasi Serangan CSRF**

Penyerangan terhadap keamanan *website*, jika tidak ditangani dengan baik, dapat menimbulkan kerugian signifikan bagi pemilik dan pengguna. Kerugian tersebut dapat berupa kerugian materil, seperti biaya ganti rugi finansial, atau kerugian non-materil, seperti hilangnya kepercayaan pengguna terhadap *website* yang dapat berdampak pada reputasi buruk dan ulasan negatif. Pentingnya menjaga keamanan *(Cyber Security)* pada sebuah *website* menjadi semakin krusial mengingat maraknya ancaman dan serangan siber saat ini.

Untuk mencegah peretasan dan memberikan keamanan yang optimal, ada beberapa langkah dan cara yang dapat diambil. Pertama, memastikan bahwa sistem dan perangkat lunak yang digunakan selalu diperbarui dengan *patch* keamanan terbaru untuk menutup celah potensial. Kedua, menerapkan enkripsi data untuk melindungi informasi sensitif dan mencegah akses tidak sah. Selain itu, monitoring aktivitas *website* secara terus-menerus dengan sistem deteksi intrusi juga dapat membantu mendeteksi dan merespons cepat terhadap ancaman yang muncul. Dengan kombinasi langkah-langkah

tersebut, pemilik *website* dapat meningkatkan keamanan *front-end* dan melindungi pengguna serta aset digitalnya dari potensi risiko keamanan.

### Memfilter Jenis *File* yang Dapat Diunggah

Pentingnya melakukan *filtering* pada *file* yang diunggah menjadi salah satu strategi efektif untuk meningkatkan keamanan suatu *website* yang sedang dalam pengembangan. Dengan menetapkan batasan terhadap jenis *file* yang dapat diunggah melalui formulir, *website* dapat memberikan lapisan perlindungan tambahan terhadap potensi ancaman keamanan. Pembatasan ini menjadi kritis karena tanpa adanya kontrol terhadap tipe *file* yang diterima, ada potensi besar bagi peretas untuk mengunggah *file* yang berisi malware, virus, atau bahkan kode program yang dapat mengeksploitasi kerentanannya dalam struktur web tersebut.

```
<labe>Pilih Gambar</labe>
<input type="file" accept="image/png, image/jpeg">
```

**Gambar 5. Sintaks *Filter File***

Proses *filtering* pada *file* yang diunggah tidak hanya berdampak pada aspek keamanan, tetapi juga membantu dalam memitigasi risiko dari segi fungsionalitas dan kinerja *website*. Dengan cara ini, pengembang dapat memastikan bahwa hanya file yang diijinkan dan aman yang dapat diakses, mengurangi kemungkinan celah keamanan dan menjaga integritas sistem secara keseluruhan. Dengan demikian, implementasi *filter* pada *file* yang diunggah menjadi langkah penting dalam membangun pertahanan keamanan yang solid pada *front-end* suatu *website*.

### Memfilter Input Pengguna Dari *Special Character*

Batasan input melalui pemberian regex pada karakter khusus memiliki peran signifikan dalam meningkatkan keamanan suatu website. Penggunaan regex untuk membatasi karakter khusus dalam input memastikan bahwa data yang dimasukkan bersih dari simbol atau karakter khusus yang dapat digunakan untuk memanfaatkan celah keamanan. Sebagai contoh, dalam pengisian username, pembatasan karakter khusus seperti #, !, ?, - dapat diimplementasikan menggunakan regex. Dengan menggunakan pendekatan ini, ketika pengguna mencoba memasukkan karakter khusus tersebut, regex akan secara otomatis menggantinya dengan teks kosong atau karakter yang telah ditentukan sebelumnya.

Selain memberikan lapisan keamanan, pembatasan input dengan regex juga berperan dalam meningkatkan integritas data dan mencegah potensi kerentanan keamanan. Dengan menghilangkan karakter khusus yang dapat disalahgunakan, *website* dapat mengurangi risiko serangan seperti SQL *injection* atau *cross-site scripting*. Oleh karena itu, penerapan regex sebagai alat pengaman *front-end* pada *website* menjadi strategi yang penting untuk menjaga keamanan dan keberlanjutan fungsionalitas *platform* tersebut.

```
<script>
    let username = '#!wahyu?~'
    username = username.replace(/[^\w\s]/gi, '')

    console.log(username)
</script>
```

**Gambar 6. Regex *Filter Special Character***

Berikut adalah contoh hasil dari username yang mengalami pembatasan terhadap karakter khusus, sebagaimana terlihat pada contoh kode dalam gambar 6 di atas.

```
wahyu                                   index.html:21
>
```

**Gambar 7. Hasil *Filter Special Character***

### Membatasi Proses *Request Server*

Konsep *client-server* saat ini menjadi populer dalam pengembangan *website* untuk menciptakan pengalaman pengguna yang lebih dinamis dan interaktif. Meskipun demikian, tingginya *traffic request* yang dihasilkan oleh web klien atau *front-end* dapat berpotensi menyebabkan penurunan kecepatan atau bahkan *down*-nya *server*. Untuk mengatasi hal ini, salah satu strategi yang dapat diterapkan adalah dengan menonaktifkan tombol *send request* selama proses masih berlangsung. Dengan demikian, dapat mencegah terjadinya penumpukan antrian di *server* yang berpotensi mengakibatkan *downtime* pada *website*. Keamanan dan kinerja website sangat penting dalam memberikan pengalaman yang baik kepada pengguna. Strategi pengelolaan *traffic request*, seperti menonaktifkan tombol *send request*, merupakan langkah preventif yang dapat mengoptimalkan kinerja *website* dan mencegah terjadinya penurunan kecepatan atau *downtime* yang dapat merugikan pengguna dan pemilik *website*.

```
<button type="submit" id="btnSubmit" onclick="submit()">Submit</button>
```

**Gambar 8. Tombol *Submit Form***

Pada tahap implementasi, memahami cara menonaktifkan tombol pada HTML menjadi kunci untuk meningkatkan pengalaman pengguna dan mencegah terjadinya antrian yang berlebihan pada *server*. Dengan menggunakan fungsi bawaan dari *JavaScript*, kita dapat mengakses elemen tombol dalam HTML dan mengatur propertinya untuk mengubah statusnya menjadi tidak aktif. Proses ini melibatkan penggunaan *event listener* untuk mendeteksi interaksi pengguna, seperti mengklik tombol, dan kemudian menjalankan fungsi yang menonaktifkan tombol.

Langkah-langkah tersebut menciptakan respons yang lebih cepat dan efisien terhadap aksi pengguna, serta meminimalkan risiko *server overload*. Oleh karena itu, pemahaman dan implementasi kode *JavaScript* untuk menonaktifkan tombol pada HTML menjadi elemen kunci dalam strategi manajemen *traffic request* yang efektif, membantu menciptakan pengalaman pengguna yang optimal.

```
<script>
    const submit = () => {
        document.getElementById('btnSubmit').disabled = true

        // MISAL: REQUEST API KE SERVER
        setTimeout(() => {
            document.getElementById('btnSubmit').disabled = false
        }, 3000)
    }
</script>
```

**Gambar 9. Aktif dan Nonaktif Tombol *Submit***

### Membuat Verifikasi *Middleware*

Verifikasi *middleware* pada sebuah *website* menjadi suatu aspek yang sangat penting untuk menerapkan keamanan yang optimal. Tanpa adanya *middleware*, sistem keamanan *website* dapat menjadi rentan, memungkinkan peretas untuk dengan mudah masuk dan melakukan pencurian data. Seiring dengan perkembangan teknologi, banyak *framework front-end* modern yang menyediakan *library routing* yang mendukung konsep *middleware* untuk sisi *front-end*. Contoh konkret penerapan middleware dapat diilustrasikan pada gambar 10, di mana *website* menerapkan suatu mekanisme yang memeriksa status pendaftaran dan *login* pengguna serta kevalidan token akses. Jika seorang pengguna belum melakukan pendaftaran, belum *login*, atau token aksesnya tidak *valid*, maka akses ke halaman home akan ditolak, dan pengguna akan dialihkan ke halaman *login*. Sebaliknya, jika pengguna telah melakukan pendaftaran, *login*, dan memiliki token akses yang *valid*, maka mereka akan diberikan akses untuk masuk ke halaman home. Dengan penerapan *middleware* yang tepat, *website* dapat meningkatkan lapisan keamanan, mencegah akses yang tidak sah, dan memberikan pengalaman pengguna yang lebih terkendali dan aman.


**Gambar 10. *Middleware* Pada Web *Front-end***

## Menerapkan *Session Token*

Penerapan *session token* di dalam suatu sistem memberikan tingkat keamanan yang signifikan dengan membatasi hanya *request* yang dilengkapi dengan token *valid* yang dapat diproses oleh *server*. Hal ini berfungsi sebagai langkah preventif terhadap upaya peretasan, di mana request tanpa token yang valid yang dilakukan oleh pihak yang tidak berhak akan diabaikan oleh *server*. Proses pemberian token ini biasanya terjadi saat pengguna melakukan login. Setelah login berhasil, *server* akan menghasilkan dan memberikan token khusus yang harus disertakan dalam setiap request selanjutnya.

Token yang diberikan tersebut kemudian dapat disimpan oleh *front-end* di beberapa tempat, seperti *local storage*, *cookies*, atau *session storage*. Dengan menyimpan token ini, *front-end* dapat memastikan bahwa setiap request yang dikirimkan oleh pengguna telah diberikan token yang valid, sehingga memperkuat lapisan keamanan dan mencegah potensi risiko terkait dengan otentikasi dan otorisasi dalam suatu *website*.


**Gambar 11. *Session Token* di *Localstorage***

Langkah selanjutnya dalam implementasi keamanan menggunakan token adalah dengan mengirimkan token bersama setiap *request* yang dilakukan oleh pengguna. Tindakan ini memungkinkan *server* untuk melakukan validasi secara efektif, memastikan bahwa setiap *request* berasal dari pengguna yang telah terverifikasi baik dari sisi akun maupun sesi akses pada suatu *website*. Dengan memasukkan token ke dalam setiap *request*, *server* dapat membaca dan memeriksa keabsahan token tersebut, memverifikasi identitas pengguna, dan memberikan izin akses sesuai dengan hak yang dimilikinya.

Pengiriman token dalam setiap *request* menjadi suatu langkah yang esensial untuk menjaga integritas dan keamanan suatu sesi pengguna. Sebagai tambahan, teknik ini dapat meminimalkan potensi risiko terkait akses tidak sah atau aktivitas yang mencurigakan di dalam sistem. Oleh karena itu, penggunaan token sebagai mekanisme otentikasi dan otorisasi memberikan kontribusi penting dalam menciptakan lapisan keamanan yang efektif dalam suatu lingkungan web.


**Gambar 12. *Session Token* Dikirim Saat Membuat *Request* ke *Server***

## 5. KESIMPULAN

Keamanan *front-end* dalam pengembangan *website* menjadi elemen krusial yang sering terabaikan oleh sebagian besar *front-end developer*. Terlalu fokus pada aspek visual dan interaktivitas, mereka cenderung mengesampingkan langkah-langkah keamanan yang seharusnya diterapkan. Hal ini membuka celah bagi potensi pencurian data atau peretasan yang dapat membahayakan integritas suatu *website*. Banyak *developer* kurang mendapatkan informasi dan edukasi terkait keamanan *front-end*, yang menyebabkan implementasi keamanan tersebut kurang optimal pada *website* yang

dikembangkan. Penelitian ini memberikan sorotan pada keamanan *front-end* sebagai elemen utama dalam mengamankan alur data masuk dan keluar dari *server*. Dengan membahas beberapa aspek keamanan yang dapat diterapkan, penelitian ini berkontribusi dalam meningkatkan pemahaman dan kesadaran para developer mengenai pentingnya menjaga keamanan *front-end*. Tujuan utama adalah memastikan bahwa praktik keamanan ini menjadi standar dalam pengembangan *website*, mengurangi potensi risiko peretasan, dan menciptakan lingkungan digital yang lebih aman di dunia internet, terutama di Indonesia. Dengan adanya peningkatan pemahaman ini, diharapkan bahwa keamanan front-end akan menjadi praktik umum yang diadopsi oleh banyak developer. Hal ini akan membawa dampak positif dalam mengurangi angka peretasan website dan aplikasi di Indonesia, menciptakan ekosistem digital yang lebih aman dan handal. Seiring dengan perkembangan teknologi dan kesadaran keamanan, diharapkan praktik keamanan *front-end* akan menjadi suatu keharusan yang tidak terpisahkan dari proses pengembangan *website*.

## 6. DAFTAR PUSTAKA


[1] P. G. Smith, *Professional Website Performance: Optimizing the Front-End and Back-End*. Wiley, 2012. [Online]. Available: https://books.google.co.id/books?id=MHLJlUfXV4QC

[2] P. D. Dutonde, S. S. Mamidwar, M. S. Korvate, S. Bafna, and P. D. D. Shirbhate, "Website Developmemt Technologies : A Review," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 1, p. 359, 2022.

[3] M. Meng, S. Steinhardt, and A. Schubert, "Application Programming Interface Documentation: What Do Software Developers Want?," *J. Tech. Writ. Commun.*, vol. 48, no. 3, pp. 295–330, Jul. 2017, doi: 10.1177/0047281617721853.

[4] R. Dastres and M. Soori, "Secure Socket Layer (SSL) in the Network and Web Security," *Interna- tional J. Comput. Inf. Eng.*, vol. 14, no. 10, pp. 330–333, 2020, [Online]. Available: https://hal.science/hal-03024764

[5] P. M. Purba, A. C. Amandha, R. H. Purnama, and A. Ikhwan, "ANALISIS KEAMANAN WEBSITE PRODI SISTEM INFORMASI UINSU MENGGUNAKAN METODE APPLICATION SCANNING," *JINTEKS (Jurnal Inform. Teknol. dan Sains)*, vol. 4, no. 4, pp. 325–329, 2022.

[6] L. K. Shar and H. B. K. Tan, "Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns," *Inf. Softw. Technol.*, vol. 55, no. 10, pp. 1767–1780, 2013, doi: https://doi.org/10.1016/j.infsof.2013.04.002.

[7] A. Bhardwaj, V. Mangat, R. Vig, S. Halder, and M. Conti, "Distributed denial of service attacks in cloud: State-of-the-art of scientific and commercial solutions," *Comput. Sci. Rev.*, vol. 39, p. 100332, 2021, doi: https://doi.org/10.1016/j.cosrev.2020.100332.

[8] G. S. Pandi (Jain), S. Shah, and K. H. Wandra, "Exploration of Vulnerabilities, Threats and Forensic Issues and its impact on the Distributed Environment of Cloud and its mitigation," *Procedia Comput. Sci.*, vol. 167, pp. 163–173, 2020, doi: https://doi.org/10.1016/j.procs.2020.03.194.

[9] A. P. Nugraha and E. Gunadhi, "PENERAPAN KRIPTOGRAFI BASE64 UNTUK KEAMANAN URL (UNIFORM RESOURCE LOCATOR) WEBSITE DARI SERANGAN SQL INJECTION," *J. Algoritm.*, vol. 13, no. 2, pp. 491–498, 2016.

[10] M. D. Al Vriano, "PENGUJIAN KEAMANAN WEB JUICE SHOP DENGAN METODE PENTESTING BERBASIS OWASP TOP 10," *Kohesi J. Multidisiplin Saintek*, vol. 01, no. 06, pp. 81–90, 2023.

[11] Terttiaavini, I. M. A. Gunawan, Kraugusteeliana, E. Winarno, and R. S. Y. Zebua, "Perancangan dan Implementasi Frontend Web untuk Sistem Pengaduan Masyarakat," *J. Inf. dan Teknol.*, vol. 5, no. 1, pp. 1–2, 2023, doi: 10.37034/jidt.v5i1.290.

[12] Q. Oktiriani, A. K. Nugroho, and E. Maryanto, "FRONTEND DEVELOPMENT IN THE FINAL STUDY MANAGEMENT SYSTEM (SIPEDA) AT THE ENGINEERING FACULTY OF JENDERAL SOEDIRMAN UNIVERSITY," *J. Tek. Inform.*, vol. 3, no. 2, pp. 321–329, 2022.

[13] A. C. Pratama, E. S. Pramukantoro, and A. Basuki, "Penerapan Metode Token-Based sebagai Mekanisme Autentikasi pada IoT Middleware," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 3, no. 10, pp. 9357–9366, 2019.

[14] H. W. Lie and B. Bos, *Cascading Style Sheets: Designing for the Web*. Pearson Education, 2005. [Online]. Available: https://books.google.co.id/books?id=McUjSXNvLI0C

[15] Sugiyono, *Metode penelitian pendidikan pendekatan kuantitatif, kualitatif dan R&D*. Alfabeta, 2018.