Keamanan Aplikasi Web Melalui Penerapan Cross Site Request Forgery(CSRF)

Taufik R. Firdaus
Jurusan Teknik Informatika
STMIK IKMI Cirebon

Absract

Currently the Internet became one of the media that can not be separated, as well as a wide variety of applications supplied her. As the development of technologies, reliance on Web applications also increased. However, web applications have a wide range of threats, one of it is a CSRF (Cross-Site Request Forgery). This study uses CSRF (Cross-Site Request Forgery) Protection. CSRF (Cross-Site Request Forgery) Protection is a treatment method that has a variety of ways, one of which uses a token in the session when the user login. Token generated at login will be used as a user id that the system of web applications to identify where the request originated. The results of this study are expected in order to increase web application defenses against CSRF (Cross-Site Request Forgery), so that web application users will be able to feel safe in using the Internet and its various feature. Reduced level of attacks on web applications. So that visitor traffic on the web application can be increased.

Keyword : CSRF(Cross-Site Request Forgery), Web Application, Request, Token, Security.

Latar Belakang

Aplikasi web merupakan platform perangkat lunak terbaru yang dominan pada saat ini, dikembangkan setiap harinya demi memudahkan kinerja manusia dalam menyelesaikan pekerjaan. Akan tetapi semakin banyak aplikasi web bertambah pula kerentanan keamanan aplikasi web yang dapat merugikan penggunanya. Kerugian yang dapat dialami pengguna seperti halnya berupa kehilangan data pribadi, rahasia dan finansial. Maka dari itu dibutuhkan lah teknik pengamanan yang dapat melindungi data pengguna dari kerentanan keamanan yang ada pada aplikasi web.

Serangan *cross site request* forgery(CSRF) merupakan ancaman aplikasi web yang ditunjukkan untuk mencuri informasi pengguna aplikasi web. Menurut penelitian yang dilakukan oleh Rupali D. Kombade dan Dr. B.B. Meshram dalam penelitiannya yang berjudul "*CSRF Vulnerabilities and Defensive Techniques*" mengatakan bahwa:

> "*CSRF(Cross-Site Request Forgery)* merupakan sebuah serangan yang mana memaksa pengguna untuk mengeksekusi aksi yang tidak di inginkan pada aplikasi web, dimana dia

(korban) saat ini terautentikasi. Serangan ini memanfaatkan keuntungan fungsi protokol *HTTP* untuk mengirim *session cookie* pada saat *request* ke *server* setiap kali pengguna berhasil melakukan autentikasi, yang membantu server dalam memastikan request tersebut berasal dari pengguna otentik."(Kombade & Meshram, 2012, p. 31)

Aplikasi web merupakan aplikasi yang memanfaatkan teknologi *web* dan *web browser* untuk menyelesaikan satu tugas atau lebih pada jaringan khususnya pada sebuah *web browser*. Menurut jurnal yang ditulis oleh Rupali D. Kombade dan Dr. B.B Meshram yang berjudul "*CSRF Vulnerabilities and Defensive Techniques*", dikatakan bahwa:

"Aplikasi web menjadi bagian dari kehidupan manusia. Beberapa diantaranya mengurangi usaha mereka seperti (sistem pemesanan, perbankan online, dll) dan beberapa diantaranya menghibur dan menghubungkan mereka secara social (facebook, myspace, dll). Akan tetapi dengan semua fasilitas ini mereka juga membawa beberapa masalah seperti hal nya serangan aplikasi web."(Kombade & Meshram, 2012)

Penelitian yang ditulis oleh Rupali D. Kombade dan Dr. B.B Meshram yang berjudul "*CSRF Vulnerabilities and Defensive Techniques*" yang dimuat dalam jurnal tahun 2012 serangan ini sedikit sekali diketahui oleh *developers*, beberapa menganggap ini sama seperti dengan XSS dan beberapa diantaranya menganggap ini teknik mitigasi XSS akan bekerja pada serangan ini. Tetapi ini berbeda dari XSS dan teknik mitigasi membutuhkan sesuatu usaha tambahan dibandingkan dengan langkah – langkah pertahanan XSS. Serangan CSRF telah diketahui sebagai "raksasa yang tertidur(sleeping giant)" pada celah keamanan berbasis web, karena banyak situs di *internet* gagal melindungi dari mereka dan mereka telah dihiraukan oleh *web developmer* dan komunitas keamanan.(Kombade & Meshram, 2012)

Pada penelitian lain yang dilakukan oleh Adam Barth, Collin Jackson, dan John Mitcheil pada jurnal tahun 2013 tentang "*Robust Defenses for Cross Site Request Forgery*" pertahanan CSRF yang paling populer adalah dengan menambahkan *token* rahasia pada setiap *request* dan untuk validasi itu *token* yang diterima terikat dengan tepat pada sesi pengguna, mencegah CSRF dengan memaksa penyerang untuk menebak sesi *token*. Ada sejumlah variasi terhadap pendekatan ini, masing – masing penuh dengan jebakan, dan bahkan situs yang mengimplementasikan teknik ini dengan benar seringkali mengabaikan *login requests* mereka karena *login requests* kekurangan sesi untuk mengikat *token*.(Barth, Jackson, & Mitchell, 2008, pp. 75–76)

Dari penelitian diatas dapat disimpulkan bahwa untuk mengatasi serangan *cross site request forgery* dapat dilakukan melalui cara menambahkan *token* rahasia yang mengikat sesi pengguna dengan *token*.

Penelitian tentang celah keamanan *cross site request forgery* yang telah dilakukan oleh Rupali D. Kombade dan Dr. B.B. Meshram tentang "CSRF Vulnerabilities and Defensive Techniques" masalah yang diankat pada jurnal ini bahwa aplikasi perangkat lunak CSRF *guard* dan CSRF *detector* merupakan perangkat yang sangat kuat akan tetapi masih belum dapat memberikan proteksi penuh, mereka hanya dapat mengurangi serangan CSRF. Oleh karena itu pertahanan menyeluruh terhadap serangan CSRF tidak tersedia.(Kombade & Meshram, 2012) Pada penelitian yang dilakukan oleh Ramisetty Ramaro yang dimuat pada tesisnya pada tahun 2009 yang berjudul "*Heuristics for Preventing Cross Site Request Forgery*" dengan mengangkat masalah bahwa mekanisme validasi token rahasia merupakan salah satu mekanisme yang efisien untuk mencegah serangan CSRF. SERIDE (SEssion RIding DEfender) merupakan *open source php library* yang menggunakan mekanisme validasi token rahasia untuk mencegah serangan CSRF dengan sukses pada sisi *server* di setiap php

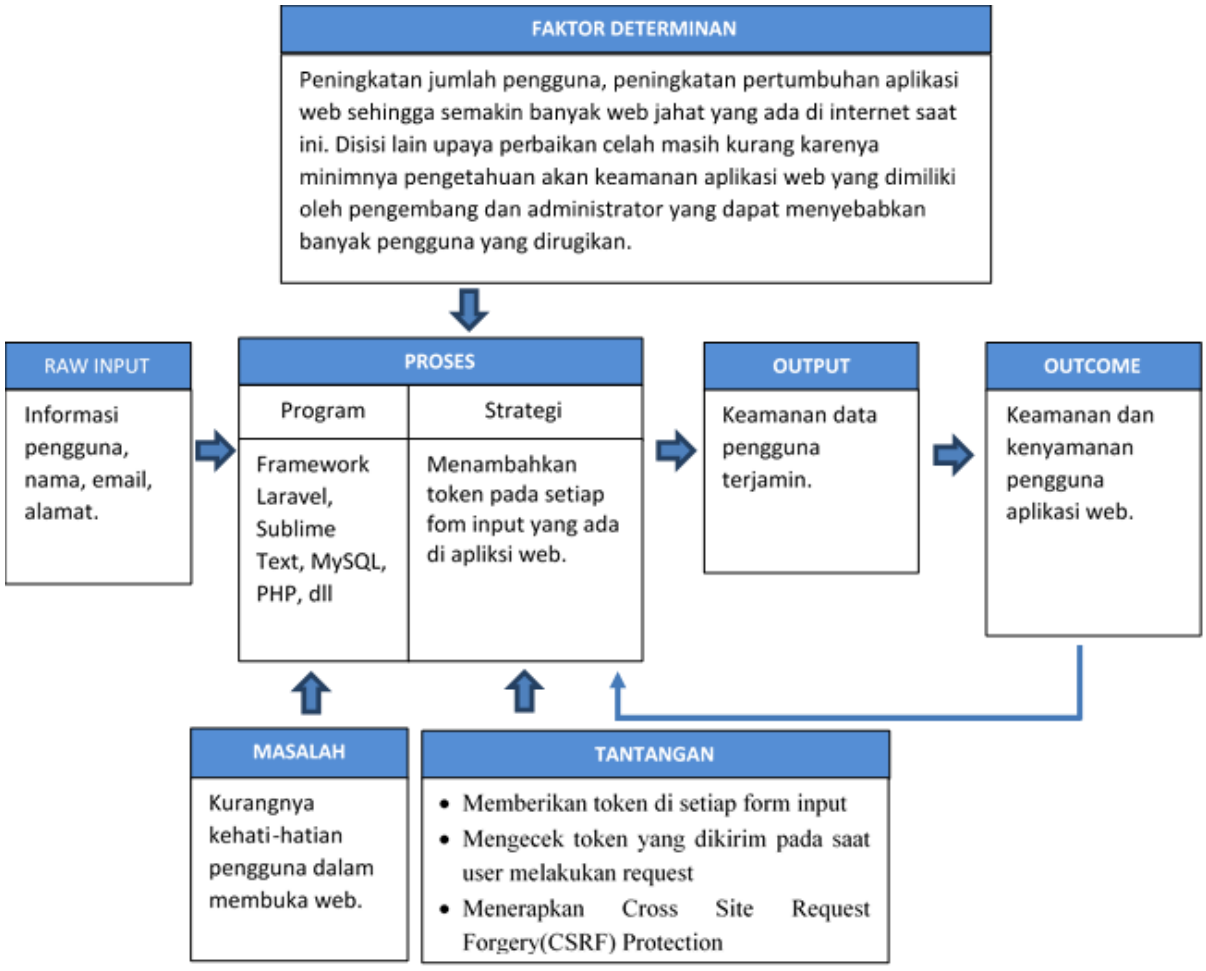
aplikasi web. Dalam proyek ini, kita mengembangkan sebuah alat *open source* SA-SERIDE yang mana merupakan alat semi otomatis dan pengembangan kode program SERIDE yang telah ada.(Ramarao, 2009, p. 82) Dari penelitian yang telah dilakukan tersebut di atas disimpulkan bahwa penggunaan *token* dalam mengamankan aplikasi web merupakan pilihan yang tepat untuk mengatasi serangan *cross site request forgery* dimana *token* akan di enkripsi terlebih dahulu agar penyerang tidak dapat mengira apa isi token tersebut.

Permasalahan

Berdasarkan latar belakang masalah diatas, maka rumusan masalah dalam penelitian ini, adalah :

a. Kurangnya kesadaran pengembang aplikasi web terhadap *Cross Site Request Forgery*(CSRF)
b. Semakin banyaknya situs yang tidak dipercaya sehingga kemungkinan pencurian data bertambah
c. Semakin banyaknya aplikasi web yang dibuat setiap tahunnya

Selanjutnya terkait dengan masalah tersebut di atas penulis merumuskannya ke dalam bagan di bawah ini:



Gambar 1 : Rumusan Masalah

Landasan Teori

1. Web Server

Apa yang memulai *server* sederhana yang memproses HTTP *request* biasa berubah menjadi *server* besar *multithread* yang mampu melayani ribuan *request*. Semakin permintaan berkembang begitu juga halnya pada jumlah webserver.

Web server mulai menyediakan semakin banyak fitur. Semakin berkembangnya permintaan, keinginan orang untuk melakukan transaksi menggunakan media ini juga bertambah. Web server menjadi bahan utama pada fungsi ini yang dapat membantu mengendalikan beberapa keadaan.(Wells, 2007, p. 17)

2. Aplikasi Web

Berdasarkan definisi, itu sesuatu yang lebih dari sekedar “situs web”. Itu adalah aplikasi klien*/server* yang menggunakan *web browser* sebagai program klien nya, dan melakukan pelayanan interaktif dengan menghubungkannya bersama server melalui internet (atau intranet). Sebuah website hanya mengirimkan kontent simple yang berasal dari file statik. Aplikasi web menyediakan konten dinamis yang disesuaikan berdasarkan parameter, tingkah laku pengguna, dan pertimbangan keamanan.(Shklar & Rosen, 2003, p. 5)

3. Kriptografi

Menurut Fresly Nandar Pabokory, Indah Fitri Astuti, Awang Harsa Kridalaksana dalam jurnalnya pada tahun 2015 dengan judul Implementasi Kriptografi Pengamanan Data Pada Pesan Teks , Isi File Dokumen , Dan File Dokumen Menggunakan Algoritma *Advanced Encryption* kriptografi merupakan seni dan ilmu untuk memproteksi pengiriman data dengan mengubahnya menjadi kode tertentu dan hanya ditujukan untuk orang yang hanya memiliki sebuah kunci untuk mengubah kode itu kembali yang berfungsi dalam menjaga kerahasiaan data atau pesan. Dalam kriptografi, data atau pesan yang dikirimkan melalui jaringan akan disamarkan sedemikian rupa sehingga tidak dapat diketahui isi dan arti dari pesan tersebut. Dalam bidang kriptografi terdapat dua konsep yang sangat penting atau utama yaitu:

A. Enkripsi

Enkripsi merupakan proses dimana informasi atau data yang hendak dikirimkan diubah menjadi bentuk yang hampir tidak dikenali sebagai informasi awalnya dengan menggunakan algoritma tertentu.

B. Dekripsi

Dekripsi adalah kebalikan dari enkripsi yaitu mengubah kembali bentuk tersamar tersebut menjadi informasi awal.

Sebuah pesan atau data yang masih asli dan belum mengalami penyandian dikenal dengan istilah *plaintext*. Kemudian setelah disamarkan dengan suatu cara penyandian, maka *plaintext* ini disebut sebagai *chipertext*. Proses penyamaran dari plaintext ke *ciphertext* disebut enkripsi (*encryption*), dan proses pengembalian dari *ciphertext* menjadi *plaintext* kembali disebut dekripsi (*decryption*).

4. Keamanan Aplikasi Web

Seperti halnya kelas teknologi terbaru, aplikasi web telah membawa jarak kerentanan keamanan terbaru. Kumpulan yang paling sering ditemui cacat telah berkembang seiring nya waktu. Serangan terbaru yang telah telah tercipta tidak di pertimbangkan pada saat aplikasi

dibuat. Berbagai masalah terlah menjadi umum sebagaimana telah bertambahnya kesadaran mereka. Teknologi baru yang telah dikembangkan mengenalkan kemungkinan terbaru untuk di eksploitasi. Beberapa kategori kelemahan sebagian besar telah mengilang sebagai hasil dari perubahan yang dibuat pada perangkat lunak *web browser*.

Serangan yang paling serius pada aplikasi web adalah mereka yang membongkar data sensitif atau mendapatkan hak akses yang memperbolehkan mengakses ke backend sistem dimana tempat aplikasi berjalan. Kompromi profil tingkat tinggi semacam ini sering terjadi saat ini. Untuk banyak organisasi, bagaimana pun juga, apapun serangan yang menyebabkan berhentinya sistem adalah kejadian yang kritis. Aplikasi setingkat serangan *denial of service*(dos) dapat digunakan untuk memperoleh hasil yang sama dengan habis nya sumber daya pada serangan terhadap infrastruktur. Akan tetapi mereka juga sering digunakan dengan teknik dan tujuan yang lebih halus. Mereka mungkin digunakan untuk mengganggu pengguna atau layanan tertentu untuk mendapatkan akhir dari kompetisi dengan pesaing pada dunia pertukaran finansial, permainan, judi *online,* dan pemesanan tiket.

Melalui evolusi ini, kompromi pada aplikasi web terkemuka masih tetap pada berada pada berita. Disini tidak terasa bahwa sudut akan berubah dan permasalahan keamanan ini berkurang. Oleh beberapa ukuran, keamanan aplikasi web saat ini menjadi sebagian besar medan perang antara penyerang dan mereka yang bersama sumberdaya komputer dan data untuk di pertahankan, dan ini sepertinya akan terus berlangsung pada masa depan yang telah terduga.(Stuttard & Pinto, 2011, p. 6)

5. Cross Site Request Forgery(CSRF)

Pada serangan *cross site request forgery,* penyerang mengganggu integritas sesi pengguna dengan *website* melalui memasukkan *network request* lewat *browser* pengguna. Peraturan keamanan *browser* memperbolehkan *website* untuk mengirimkan HTTP *requests* kepada setiap alamat jaringan. Peraturan ini memperbolehkan penyerang untuk mengontrol content yang ditampilkan oleh *browser* untuk menggunakan *resource* yang tidak dinyatakan dibawah kekuasaannya(penyerang):

a. *Network Connectivity*(Konektivitas Jaringan)

   Sebagai contoh, jika seorang pengguna berada dibalik *firewall*, penyerang dapat mempengaruhi browser pengguna untuk mengirimkan *network requests* kepada mesin dibalik *firewall* yang mungkin tidak secara langsung berada dalam batas jangkauan mesin penyerang. Walau pun pengguna tidak berada dibalik *firewall*, *request* membawa

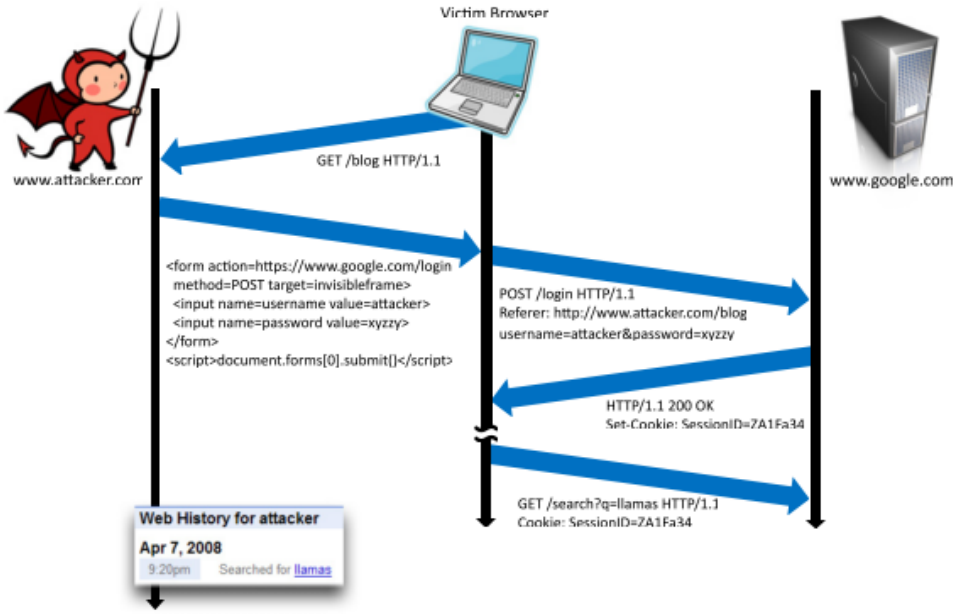
alamat IP pengguna dan kemungkinan membingungkan layanan yang menggunakan autentikasi alamat IP.

b. *Read Browser State*(Membaca Kondisi *Browser*)

Request dikirim melalui tumpukan jaringan *browser* biasanya menyertakan keadaan *browser*, seperti halnya *cookie*, sertifikat klien, atau *headers* autentikasi dasar. Situs yang mengandalkan kondisi autentikasi seperti ini kemungkinan akan di pusingkan dengan *requests* semacam ini.

c. *Write Browser State*(Menulis Kondisi *Browser*)

Ketika penyerang menyebabkan permasalahan *network request* pada *browser*, *browser* juga menguraikan dan bertindak pada respon. Seperti contoh, jika respon mengandung sebuah *Set-Cookie header*, sebuah *browser* akan memodifikasi *cookie* yang disimpannya. Modifikasi ini dapat mengawali serangan yang mulus(lancar), seperti gambar berikut:



Gambar 2 : Contoh serangan CSRF dengan memodifikasi *cookie*

Ancaman yang berada di dalam lingkup atau ruang. Kami mempertimbangkan tiga model ancaman yang berbeda, bervariasi dengan kemampuan penyerang:

a. Forum Poster

Banyak *website*, seperti forum, membiarkan pengguna untuk memberikan konten yang terbatas. Sebagai contoh, situs seringkali memperbolehkan pengguna untuk

meyerahkan konten pasif, seperti gambar atau *hyperlinks*. Jika penyerang memilih URL gambar jahat, kemungkinan *network request* mengarah kepada serangan CSRF.

b. Web Attacker

Sebuah web penyerang adalah pelaku utama berbahaya yang memiliki nama *domain*, sertifikat SSL yang resmi, dan beroperasi pada *web server*. Semua kemampuan ini dapat diperoleh hanya dengan $10USD. Jika pengguna mengunjungi `attacker.com`, seorang penyerang dapat menempelkan sebuah serangan CSRF dengan menginstruksikan *browser* pengguna untuk melakukan *cross-site requests* menggunakan kedua metode GET dan POST.

c. Network Attacker

Sebuah jaringan penyerang adalah pelaku utama berbahaya yang mengontrol koneksi jaringan pengguna. Seperti contoh, sebuah "*evil twin*" *wireless router* atau server DNS yang terambil alih dapat di manfaatkan oleh penyerang untuk mengendalikan koneksi jaringan pengguna. Serangan ini membutuhkan lebih banyak sumber daya dari pada serangan *web*, tetapi kita mempertimbangkan ancaman ini dalam lingkup situs HTTPS karena HTTPS dirancang untuk melindungi dari serangan jaringan yang aktif.

Keluar dari ruang lingkup ancaman. Ada sejumlah model ancaman yang terkait yang tidak dipertimbangkan. CSRF *defenses* dan *protection* merupakan pertahanan yang saling mengisi dari ancaman berikut ini:

a. Cross Site Scripting(XSS)
b. Malware
c. DNS Rebinding
d. Certificate Errors
e. Phishing
f. User Tracking(Barth et al., 2008, p. 76–77)

6. Laravel Framework

Menurut buku yang ditulis oleh Jack Vo yang berjudul *Learning Laravel: The Easiest Way* Laravel adalah *full stack framework*, yang artinya anda dapat mengembangkan aplikasi web secara komplit dari awal menggunakan *warper database* menakjubkan yang disebut *Eloquent ORM* dan mesin *template* dengan nama *Blade*. Banyak permasalahan dalam proses pembuatan aplikasi web yang sudah diselesaikan oleh Laravel.

"Syntax dari Laravel sangatlah bersih dan mudah di ikuti. Metode dan fungsi di definisikan dengan baik. Terkadang anda dapat mengiranya tanpa melihat ke dokumentasi. Anda juga

dapat membuat peraturan sendiri, gaya anda untuk menulis kode. Laravel juga memberikan bayak kebebasan lainnya."(Vo, 2014, p. 6)

7. Metode Analisis GAP

Menurut buku yang ditulis oleh K. Douglas Hoffman dan Johtn E.G. Bateson yang berjudul *Services Marketing Concept, Strategies, & Cases* Gap analysis adalah suatu alat yang digunakan untuk mengetahui mengenai kondisi aktual yang sedang berjalan di perusahaan tersebut, untuk kemudian diperbandingkan dengan sumber daya perusahaan tersebut. Hal tersebut dilakukan agar untuk mengetahui apakah suatu perusahaan sudah bergerak di proses bisnisnya secara optimal untuk memaksimalisasikan kinerja perusahaan tersebut. Gap analysis dapat dilihat melalui beberapa perspektif, yaitu :

1. Organisasi (Sumber daya manusia),
2. Arah bisnis perusahaan
3. Proses bisnis perusahaan

Teknologi informasi.(Hoffman & Bateson, 2010, p. 334)

Analisa gap dapat juga diartikan sebagai perbandingan kinerja aktual dengan kinerja potensial atau yang diharapkan. Sebagai metoda, analisa gap digunakan sebagai alat evaluasi bisnis yang menitikberatkan pada kesenjangan kinerja perusahaan saat ini dengan kinerja yang sudah ditargetkan sebelumnya(Saluky, 2016).

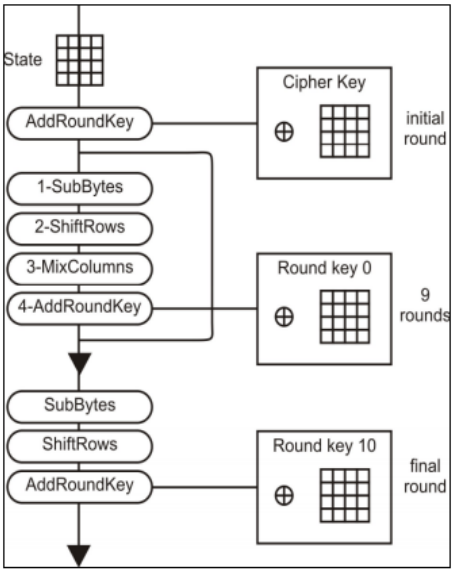
Pembahasan

Keamanan aplikasi web melalui penerapan teknik *cross site request forgery*(csrf) *protection*, dilakukan dengan tujuan sebagai berikut:

1. **Mengetahui penggunaan metode enkripsi keamanan aplikasi web melalui teknik *cross site request forgery(*csrf*) protection***

   Penentuan metode enkripsi yang dipakai *csrf-token* pada aplikasi web adalah AES-256-CBC. AES (Advanced Encryption Standard) menggunakan algoritma Rijndael yang telah memenangkan sayembara terbuka yang dilakukan oleh NIST (National Institute of Standard and Technology). Sayembara ini dilakukan untuk menemukan algoritma baru untuk menggantikan algoritma DES (DataEncryption Standard) yang dirasa sudah tidak aman lagi. Algoritma Rijndael menggunakan substitusi dan permutasi, dan

sejumlah putaran (cipher berulang), setiap putaran menggunakan kunci internal yang berbeda (kunci setiap putaran disebut round key). Rijndael beroperasi dalam orientasi byte. Garis besar algoritma Rijndael yang beroperasi pada blok 256 bit dengan panjang kunci 256 bit.(Silva, Dessyanto, & Heriyanto, 2013, p. 34)

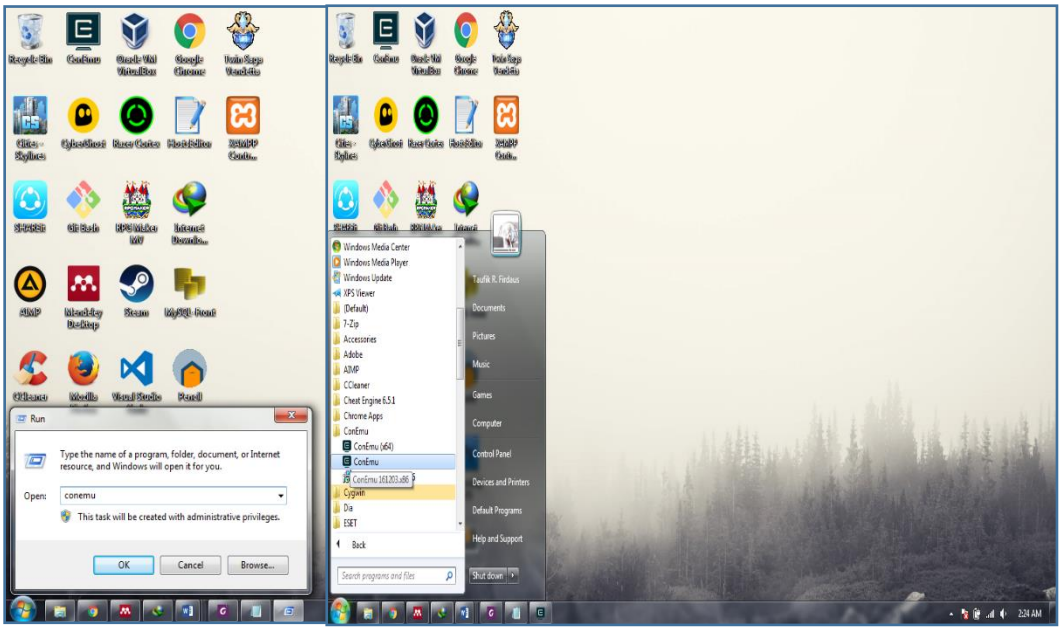


Gambar 3 Diagram Proses Enkripsi AES

2. **Merancang kunci enkripsi *token* keamanan aplikasi web melalui teknik *cross site request forgery(*csrf*) protection***

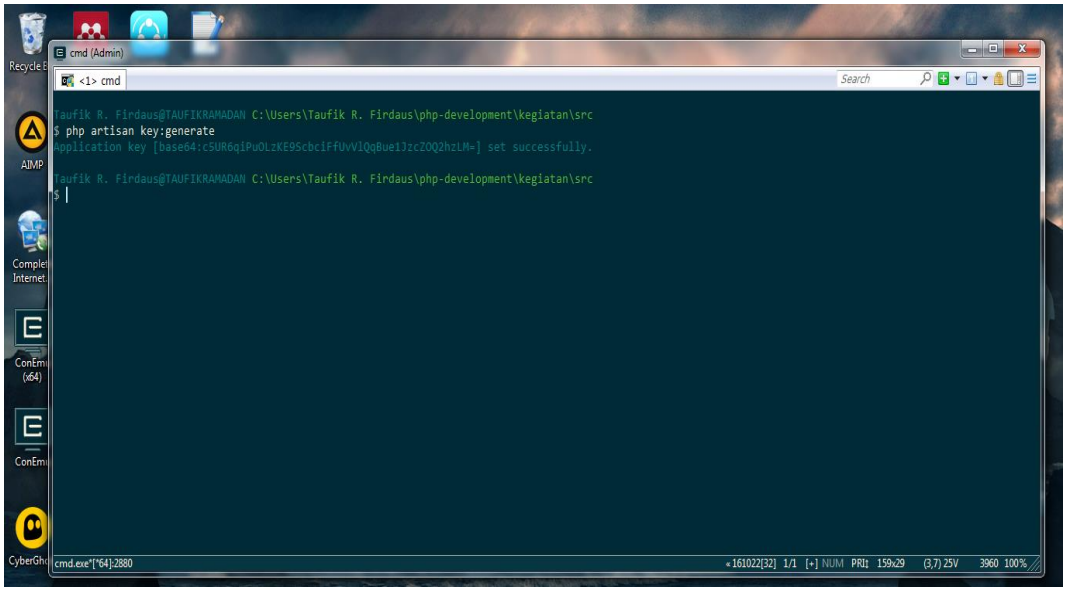
Dalam merancang kunci enkripsi yang pertama kali diperhatikan adalah panjang kunci enkripsi dan kedua adalah *alpha-numeric* acak agar kunci tidak dapat ditebak oleh penyerang. Panjang kunci akan digunakan pada tahap pengulangan proses enkripsi, semakin panjang kunci maka semakin aman. Pada Laravel PHP Framework disediakan sebuah fungsi yang dapat memudahkan pembuatan kunci enkripsi dengan cara sebagai berikut:

1. Buka console atau cmd lewat start menu atau fitur run



Gambar 5 Petunjuk Membuka Console Atau CMD

2. Masuk ke folder aplikasi web site yang telah dibuat dengan laravel framework dengan menggunakan perintah `cd` untuk melakukan pindah folder
3. Setelah berhasil masuk ke folder aplikasi web selanjutnya adalah dengan mengetikkan atau menuliskan perintah bawaan laravel framework untuk membuat kunci enkripsi yaitu `php artisan key:generate`



Gambar 6 Membuat Kunci Enkripsi

4. Jika kunci enkripsi telah dibuat maka kunci yang dihasil kan akan tersimpan didalam file .env

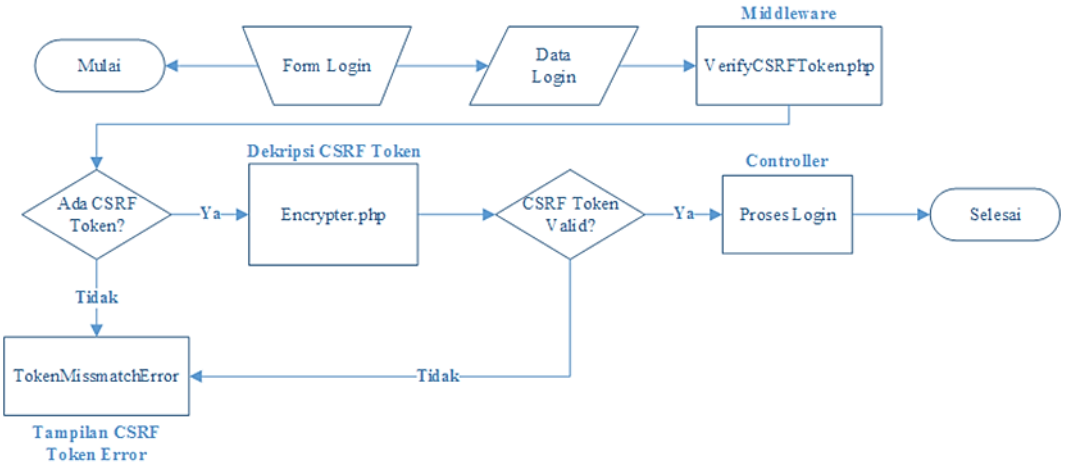
3. Mengimplementasi keamanan aplikasi web melalui teknik *cross site request forgery*(csrf) *protection*

Langkah – langkah mengimplementasikan keamanan aplikasi web melalui teknik *cross site request forgery*(CSRF) *protection* adalah sebagai berikut:

a. Membuat diagram alir(*FlowChart*) proses pengamanan melalui penerapan teknik *cross site request forgery*(CSRF) *protection*

Pada tahapan ini, akan dibuat sebuah rancangan *flowchat* yang menggambarkan jalannya alur keamanan aplikasi web melalui penerapan teknik *cross site request forgery*(csrf) *protection*. *FlowChart* akan menggambarkan:

1. Proses Saat Login



Gambar 7 Proses Pengamanan *CSRF Protection* Saat Login

Mengetahui penggunaan metode enkripsi keamanan aplikasi web melalui teknik *cross site request forgery(*csrf*) protection*

Dalam hal ini hasil yang didapat dari mengetahui penggunaan metode enkripsi adalah AES-256-CBC yang merupakan pengembangan terbaru metode enkripsi menggunakan algoritma Rijndael dimana enkripsi ini telah memenangkan lomba yang diselenggarakan oleh NIST (National Institute of Standard and Technology). Dimana lomba tersebut bertujuan mencari metode enkripsi terbaru yang dapat menggantikan metode enkripsi DES yang sudah tidak aman lagi. Keunggulan metode enkripsi AES-256-CBC adalah kecepatan dan keamanan dalam melakukan enkripsi teks atau pesan sehingga isi(arti) pesan tersebut tidak dapat dilihat oleh orang yang tidak memiliki kunci enkripsi tersebut.

Merancang kunci enkripsi *token* keamanan aplikasi web melalui teknik *cross site request forgery(*csrf*) protection*

Dalam pembuatan kunci enkripsi yang dapat dihasilkan melalui penggunaan perintah lewat *command line*(cmd) dapat dilihat pada file .env yang akan digunakan setiap kali melakukan dekripsi password saat login dan disaat melakukan verifikasi *csrf-token*.

Mengimplementasi keamanan aplikasi web melalui teknik *cross site request forgery(*csrf*) protection*

Hasil dari implementasi keamana aplikasi web melalui teknik *cross site request forgery*(CSRF) *protection* yaitu:

a. Form yang aman dari serangan *cross site request forgery*(CSRF)

```
14              <form action="{{ route('admin.user.store') }}" class="form-horizontal" method="POST" accept-charset="UTF
    8">
15                  <input type="hidden" name="_token" value="{{ csrf_token() }}">
16                  <div class="box-body">
17                      <div class="form-group @if($errors->has('nama')) has-error @endif">
18                          <label for="nama" class="control-label col-sm-2">Nama lengkap</label>
19                          <div class="col-sm-9">
20                              <input type="text" class="form-control" name="nama", placeholder="Nama Lengkap">
21                              @if($errors->has('nama'))
22                                  <span class="help-text">{{ $errors->first('nama') }}</span>
23                              @endif
24                          </div>
25                      </div>
26                      <div class="form-group @if($errors->has('email')) has-error @endif">
27                          <label for="email" class="control-label col-sm-2">E Mail</label>
28                          <div class="col-sm-9">
29                              <input type="email" class="form-control" name="email", placeholder="E Mail">
30                              @if($errors->has('email'))
31                                  <span class="help-text">{{ $errors->first('email') }}</span>
32                              @endif
33                          </div>
34                      </div>
35                      <div class="form-group @if($errors->has('password')) has-error @endif">
36                          <label for="password" class="control-label col-sm-2">Password</label>
37                          <div class="col-sm-9">
38                              <input type="password" class="form-control" name="password" placeholder="Password">
39                              @if($errors->has('password'))
40                                  <span class="help-text">{{ $errors->first('password') }}</span>
41                              @endif
42                          </div>
43                      </div>
44                      <div class="form-group @if($errors->has('re_password')) has-error @endif">
45                          <label for="re_password" class="control-label col-sm-2">Tulis ulang password</label>
46                          <div class="col-sm-9">
47                              <input type="password" class="form-control" name="re_password" placeholder="Tulis ulang
    password">
48                              @if($errors->has('re_password'))
49                                  <span class="help-text">{{ $errors->first('re_password') }}</span>
50                              @endif
51                          </div>
52                      </div>
53                  </div>
54                  <div class="box-footer">
55                      <a href="{{ route('admin.user.index') }}" class="btn btn-default">Kembali</a>
56                      <button class="btn btn-primary pull-right" type="submit">Registrasi Pengguna</button>
57                  </div>
58              {!! Form::close() !!}
```

Gambar 8 Kode Program Form Login

b. CSRF-Token dan password pengguna yang tidak mudah ditebak

Keamanan aplikasi web dari serangan *cross site request forgery*(CSRF) tidak mudah ditebak karena *csrf-token* beserta *password* telah dienkripsi serta kunci dari enkripsi tersebut disimpan dalam file .env yang terletak didalam root folder aplikasi web.

Mengevaluasi keamanan aplikasi web melalui teknik *cross site request forgery(*csrf*) protection*

Populasi penelitian ini adalah Data Center di IAIN Syekhnurjati Cirebon, dengan sample data yang berupa kode program aplikasi web dan dianalisi menggunakan metode *gap* data yang dihasilkan berupa table, grafik dan gambar hasil ujicoba sebagai berikut:

Table 1 Hasil Ujicoba Keamanan Aplikasi Web Dari Serangan CSRF

No	Nama Form	Hasil Ujicoba Serangan CSRF	
		Sebelum Penerapan	Sesudah Penerapan
1	Form Login	Tidak Aman	Aman
2	Form Pendaftaran Pengguna	Tidak Aman	Aman
3	Form Jabatan	Tidak Aman	Aman
4	Form Lokasi	Tidak Aman	Aman
5	Form Kegiatan	Tidak Aman	Aman

Kesimpulan

Dari hasil di atas dapat diambil kesimpulan bahwa Keamanan Aplikasi Web Melalui Penerapan Teknik *Cross Site Request Forgery*(CSRF) *Protection* diantaranya:

1. Mengetahui penggunaan metode enkripsi keamanan aplikasi web melalui teknik cross site request forgery(csrf) protection

   Penggunaan AES-256-CBC sebagai metode enkripsi dapat dikatakan sebagai pilihan yang tepat dikarenakan telah melalui perbandingan dengan metode enkripsi lain.

Algoritma	MiB/Second	Cycles Per Byte	Microseconds to Setup Key
SHA-256	111	15.8	No Key
SHA-512	99	17.7	No Key
RIPEMD-320	110	15.9	No Key
AES-CBC (256-bit key)	80	21.7	0.619
AES-CBC (128-bit key)	109	16.0	0.569

Table 5.1 Tabel Perbandingan Algoritma Enkripsi

Berdasarkan table 5.1 dapat dilihat bahwa AES –CBC-256 lebih unggul dari metode enkripsi yang lain. Metode enkripsi AES-CBC-256 memiliki kecepatan enkripsi 80 MiB/Second, pengulangan enkripsi 21.7 putaran per byte, dan 0,619 microseconds untuk memasang kunci enkripsi. Dengan ini dapat dikatakan bahwa pemelihan AES-CBC-256 sebagai metode enktipsi dengan tepat.

2. Merancang kunci enkripsi token keamanan aplikasi web melalui teknik cross site request forgery(csrf) protection

   Pada penrancangan kunci enkripsi diperlukan kunci yang tidak mudah ditebak, tidak sering dipakai, dan tidak pendek sehingga, kunci tidak mudah ditebak atau lolos ke tangan orang yang memiliki niat jahat pada aplikasi web.

3. Mengimplementasi keamanan aplikasi web melalui teknik cross site request forgery(csrf) protection

   a. Dalam mengimplementasikannya *token* harus selalu diberikan pada setiap halaman dan di dalam *form*

   b. Saat proses verifikasi, *token* yang disimpan dengan *token* yang ada pada *form* harus sama

   c. Desain ulang *table* pengguna sangat dibutuhkan agar *token* dapat disimpan dan digunakan pada saat yang diperlukan

   d. Alur diagram keamanan aplikasi web dari serangan *cross site request forgery*(csrf) perlu dibuat, untuk memudahkan dalam mengimplementasikan teknik keamanan

4. Mengevaluasi keamanan aplikasi web melalui teknik cross site request forgery(csrf) protection

   Untuk evaluasi keamanan aplikasi web melalui teknik cross site request forgery(csrf) protection masih dibutuhkan tenaga ahli(orang) yang mengerti tentang celah keamanan aplikasi web dan cara mengamankan aplikasi web

Daftar Pustaka

Barth, A., Jackson, C., & Mitchell, J. C. (2008). Robust defenses for cross-site request forgery. *Proceedings of the 15th ACM Conference on Computer and Communications Security CCS 08*, 75. http://doi.org/10.1145/1455770.1455782

Hoffman, K. D., & Bateson, J. E. G. (2010). *Services Marketing Concept, Strategies, & Cases*.

Kombade, R. D., & Meshram, B. B. (2012). CSRF Vulnerabilities and Defensive Techniques. *International Journal of Computer Network and Information Security(IJCNIS)*, *4*(1), 31. http://doi.org/10.5815/ijcnis.2012.01.04

Ramarao, R. (2009). *HEURISTICS FOR PREVENTING CROSS SITE REQUEST FORGERY*.

Saluky. (2016). Pengembangan Blueprint Sistem Informasi Akademik Terintegrasi (Studi Kasus : IAIN Syekh Nurjati Cirebon). *ITEj (Information Technology Engineering Journals)*, *1*(Software Engineering). Retrieved from https://www.syekhnurjati.ac.id/jurnal/index.php/itej/article/view/1251

Shklar, L., & Rosen, R. (2003). *Web Application Architecture: Principles, Protocols and Practices*. *Software Engineering*.

Silva, L. D. P., Dessyanto, B. P., & Heriyanto. (2013). Aplikasi Enkripsi Dan Dekripsi File Dengan Menggunakan Aes (Advanced Encryption Standard) Algoritma Rijndael Pada Sistem Operasi Android. *Aplikasi Enkripsi*, *10*(1), 33–42.

Stuttard, D., & Pinto, M. (2011). *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws* (Vol. 7).

Vo, J. (n.d.). Learning Laravel : The Easiest Way.

Wells, C. (2007). *Securing Ajax Applications*. (Tatiana Apandi, Ed.)*O'Reilly Media, Inc.* (1st ed., Vol. 1). O'Reilly Media, Inc. http://doi.org/10.1017/CBO9781107415324.004