IMPLEMENTASI KEAMANAN *WEBSITE* DARI SERANGAN *CROSS SITE REQUEST FORGERY* MENGGUNAKAN ALGORITMA *RIJNDAEL* PADA *FRAMEWORK CODEIGNITER*

SKRIPSI

DISUSUN OLEH

Aulil Mushalli
NPM. 1909010056

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA
MEDAN
2023

IMPLEMENTASI KEAMANAN *WEBSITE* DARI SERANGAN *CROSS SITE REQUEST FORGERY* MENGGUNAKAN ALGORITMA *RIJNDAEL* PADA *FRAMEWORK CODEIGNITER*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer (S.Kom) dalam Program Studi Sistem Informasi pada Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Muhammadiyah Sumatera Utara

Aulil Mushalli
NPM. 1909010056

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA
MEDAN
2023

LEMBAR PENGESAHAN

LEMBAR PENGESAHAN

Judul Skripsi : Implementasi Keamanan *Website* Dari Serangan *Cross Site Request Forgery* Menggunakan Algoritma *Rijndael* Pada *Framework Codeigniter*

Nama Mahasiswa : Aulil Mushalli

NPM : 1909010056

Program Studi : Sistem Informasi

Menyutujui
Komisi Pembimbing

(Ferdy Riza ST, M.Kom)
NIDN. 0103068901

Ketua Program Studi

(Martiano S.Pd, S.Kom., M.Kom)
NIDN. 0128029302

Dekan

(Dr. Al - Khowarizmi, S.Kom., M.Kom)
NIDN. 0127099201

PERNYATAAN ORISINALITAS

PERNYATAAN ORISINALITAS

IMPLEMENTASI KEAMANAN *WEBSITE* DARI SERANGAN *CROSS SITE REQUEST FORGERY* MENGGUNAKAN ALGORITMA *RIJNDAEL* PADA *FRAMEWORK CODEIGNITER*

SKRIPSI

Saya menyatakan bahwa karya tulis ini adalah hasil karya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing disebutkan sumbernya.

Medan, 14 Agustus 2023

Yang membuat pernyataan

Aulil Mushalli
NPM. 1909010056

PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademika Universitas Muhammadiyah Sumatera Utara, saya bertanda tangan dibawah ini:

Nama	: Aulil Mushalli
NPM	: 1909010056
Program Studi	: Sistem Informasi
Karya Ilmiah	: Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Muhammadiyah Sumatera Utara Hak Bebas Royalti Non-Eksekutif (*Non-Exclusive Royalty free Right*) atas penelitian skripsi saya yang berjudul:

IMPLEMENTASI KEAMANAN *WEBSITE* DARI SERANGAN *CROSS SITE REQUEST FORGERY* MENGGUNAKAN ALGORITMA *RIJNDAEL* PADA *FRAMEWORK CODEIGNITER*

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-Eksekutif ini, Universitas Muhammadiyah Sumatera Utara berhak menyimpan, mengalih media, memformat, mengelola dalam bentuk database, merawat dan mempublikasikan Skripsi saya ini tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis dan sebagai pemegang dan atau sebagai pemilik hak cipta.

Demikian pernyataan ini dibuat dengan sebenarnya.

Medan, 14 Agustus 2023

Yang membuat pernyataan

Aulil Mushalli
NPM. 1909010056

RIWAYAT HIDUP

1. DATA PRIBADI

Nama	: Aulil Mushalli
NPM	: 1909010056
Tempat dan Tanggal Lahir	: Sei Bilah, 01 Agustus 2001
Jenis Kelamin	: Laki - Laki
Agama	: Islam
Kewarganegaraan	: Indonesia
Alamat	: Jl. H. Hasan Perak Desa Teluk Meku Dusun III
No Telepon	: 081269338690
Email	: aulilmushalli@gmail.com

2. DATA ORANGTUA

Nama Ayah	: Sahniar, S.Pd
Pekerjaan	: PNS
Nama Ibu	: Itawati
Pekerjaan	: Ibu Rumah Tangga
Alamat	: Jl. H. Hasan Perak Desa Teluk Meku Dusun III
No Telepon	: 085270044162

3. DATA PENDIDIKAN FORMAL

Sekolah Dasar	: SDN 050752 P. Brandan	TAMAT: 2013
Sekolah Menegah Pertama	: SMP Negeri 1 Babalan	TAMAT: 2016
Sekolah Menengah Atas	: SMA Negeri 1 Babalan	TAMAT: 2019

KATA PENGANTAR

بسم الله الرحمن الرحيم

Penulis ingin mengekspresikan rasa syukur kepada Allah SWT atas semua berkat-Nya, sembari berdoa agar limpahan rahmat dan kebaikan-Nya selalu menyertai Nabi Muhammad Saw, keluarganya, para sahabatnya, dan seluruh umat hingga akhir waktu. Semoga ini membantu penulis memenuhi salah satu persyaratan untuk meraih gelar Sarjana Komputer dari Universitas Muhammadiyah Sumatera Utara yang berjudul **"IMPLEMENTASI KEAMANAN *WEBSITE* DARI SERANGAN *CROSS SITE REQUEST FORGERY* MENGGUNAKAN ALGORITMA RIJNDAEL PADA *FRAMEWORK CODEIGNITER*"**.

Penulis dengan penuh kesadaran mengakui bahwa skripsi ini belum mencapai standar kesempurnaan yang diharapkan. Karenanya, dengan tulus hati, penulis mengundang berbagai kritik dan saran yang berharga untuk mengembangkan kualitas skripsi ini. Dalam perjalanan menyusun karya ini, penulis telah menghadapi beragam tantangan dan rintangan, namun berkat bimbingan serta nasihat yang berlimpah dari berbagai pihak, penulis berhasil menyelesaikan skripsi ini sesuai dengan batas waktu yang telah ditetapkan.. Serta dalam penyusunan dan penelitian skripsi ini tidak terlepas dari bantuan, bimbingan serta dukungan dari berbagai pihak. Oleh karena itu, penulis dengan senang hati menyampaikan terima kasih kepada:

1. Bapak Prof. Dr. Agussani. M.AP selaku Rektor Universitas Muhammadiyah Sumatera Utara.
2. Bapak Dr. Al-Khowarizmi, S.Kom., M.Kom., selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi (FIKTI) UMSU.

3. Bapak Martiano S.Kom., M.Kom. Selaku Ketua jurusan program S1 Sistem Informasi yang telah memberikan arahan dan bimbingan kepada penulis
4. Ibu Yoshida Sary, S.E., S.Kom., M.Kom. Selaku Sekretaris jurusan program S1 Sistem Informasi yang telah memberikan arahan dan bimbingan kepada penulis.
5. Bapak Ferdy Riza S.T, M.Kom selaku dosen pembimbing skripsi penulis yang telah banyak membantu penulis baik itu dukungan, saran, dan bimbingan dalam penyusunan penelitian ini.
6. Kedua orang tua penulis yang selalu memberikan dukungan dan do'a, Ayahanda Sahniar, S.Pd dan Ibunda Itawati
7. Kepada seluruh staf dan dosen Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Muhammadiyah Sumatera Utara yang telah banyak memberikan ilmu, wawasan, motivasi dan pembelajaran berharga dari awal sampai akhir perkuliahan.
8. Ucapan terima kasih kepada teman baik saya M. Arizki yang telah membantu penulis dalam memecahkan masalah program dalam penelitian ini.
9. Terima kasih kepada teman-teman saya dari grup "Pejuang S.Kom" yang telah memberi semangat dan motivasi serta senda guraunya dalam menyusun penelitian ini.

Penulis berharap agar penelitian ini menjadi masukan bagi kita semua dan berguna bagi penulis sendiri agar dapat melihat sejauh mana kemempuan yang dimiliki penulis selama mengikuti perkuliahan program Strata-I/Sarjana jurusan Sistem Informasi di Universitas Muhammadiyah Sumatera Utara. Dan juga bisa

menjadi sebagai referensi bagi adik-adik kelas yang kelak akan menyusun skripsi nantinya.

Medan,14 Agustus 2023

Penulis,

AULIL MUSHALLI

IMPLEMENTASI KEAMANAN *WEBSITE* DARI SERANGAN *CROSS SITE REQUEST FORGERY* MENGGUNAKAN ALGORITMA *RIJNDAEL* PADA *FRAMEWORK CODEIGNITER*

ABSTRAK

Teknologi informasi dan komunikasi saat ini memberikan kemudahan bagi kehidupan manusia. Salah satu perkembangan yang cukup signifikan adalah aplikasi yang berbasis *web*. Namun, karena jumlah aplikasi berbasis *web* meningkat, kerentanan aplikasi berbasis *web* dapat membahayakan pengguna. Terdapat beberapa cara yang dapat digunakan untuk melakukan pengujian terhadap keamanan *Website*. *Cross Site Request Forgery* atau *CSRF* adalah salah satu serangan yang dilakukan pada *website* berdasarkan celah *input* pada *website*. Salah satu yang dapat dilakukan untuk mengatasi serangan *CSRF* yaitu dengan menggunakan *token CSRF* tersembunyi. Salah satu teknik pengamanan sistem aplikasi berbasis *Website* adalah dengan menggunakan teknik enkripsi data. Satu algoritma yang dipilih oleh *National Institute of Standards and Technology* (NIST) sebagai pemenang dalam perlombaan untuk menjadi kandidat *Advanced Encryption Standard* (AES) adalah algoritma *Rijndael*. *Token* anti *CSRF* akan di enkripsi menggunakan *AES* kemudian *token* akan di sembunyikan melalui *hidden input* pada *form* dan *token* akan di sembunyikan pada *cookie browser*. Sistem *website* yang dibangun menggunakan *Codeigniter*. Pengujian *website* yang sudah terlindungi dari serangan *CSRF* menggunakan *penetration testing*, dimana serangan akan dilakukan sebanyak sepuluh kali dengan metode pengujian *black box*. Hasil pengujian menggunakan metode *blackbox* adalah *website* asli mampu menahan serangan *CSRF* dengan menggunakan *token* anti *CSRF* yang telah di enkripsi menggunakan *AES*.

Kata Kunci: *Website, CSRF, Aes, Codeigniter,* Keamanan *Website, Penetration Testing*

IMPLEMENTATION OF WEBSITE SECURITY FROM CROSS SITE REQUEST FORGERY ATTACKS USING THE RIJNDAEL ALGORITHM ON THE CODEIGNITER FRAMEWORK

ABSTRACT

Information and communication technology is currently providing convenience for human life. One of the significant developments is a web-based application. However, as the number of web-based applications increases, web-based application vulnerabilities can harm users. Several ways can be used to test website security. Cross-Site Request Forgery, or CSRF, is one of the attacks carried out on websites based on input gaps on the website. One thing that can be done to overcome CSRF attacks is to use hidden CSRF tokens. One of the security techniques for Web-based application systems is to use data encryption techniques. One algorithm that was chosen by the National Institute of Standards and Technology (NIST) as the winner in the race to become a candidate for Advanced Encryption Standard (AES) is the Rijndael algorithm. The anti-CSRF token will be encrypted using AES, then the token will be hidden via input on the form, and the token will be hidden in the browser cookie. Website system built using CodeIgniter. Website testing that has been protected from CSRF attacks uses penetration testing, where the attack will be carried out ten times using the black box testing method. The testing results using the black box method show that the original website can withstand CSRF attacks using anti-CSRF tokens encrypted using AES.

Keywords: Website, CSRF, Aes, Codeigniter, Website Security, Penetration Testing

DAFTAR ISI

DAFTAR TABEL

HALAMAN

DAFTAR GAMBAR

HALAMAN

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Teknologi informasi dan komunikasi saat ini memberikan kemudahan bagi kehidupan manusia. Salah satu perkembangan yang cukup signifikan adalah aplikasi yang berbasis *web* (Budiman et al., 2021). Namun, karena jumlah aplikasi berbasis *web* meningkat, kerentanan aplikasi berbasis *web* dapat membahayakan pengguna. Setiap kerugian yang mungkin anda alami, termasuk kehilangan data pribadi, sensitif, atau keuangan. Oleh karena itu, diperlukan suatu teknologi keamanan yang dapat melindungi data pengguna dari kerentanan yang ada pada aplikasi *web*. Resiko ancaman menjadi masalah nyata ketika berhasil dieskploitasi dalam sebuah serangan. Resiko ancaman akan benar-benar menjadi masalah jika berhasil diekspolitasi dalam sebuah serangan.

Terdapat beberapa cara yang dapat digunakan untuk melakukan pengujian terhadap keamanan *Website*. *Cross Site Request Forgery* atau *CSRF* adalah salah satu serangan yang dilakukan pada *website* berdasarkan celah *input* pada *website*. Kerentanan keamanan tersebut terjadi karena adanya celah pada *form* pada *website*, sehingga dari sini penyerang dapat melakukan *request* ke *form* asli dengan *script* yang telah disiapkan. Salah satu yang dapat dilakukan untuk mengatasi serangan *CSRF* yaitu dengan menggunakan *token CSRF*. Pada saat *action* POST pada *form* dijalankan, *token CSRF* akan dimasukkan. Saat *request* dibuat, maka *backend* akan memvalidasi apakah *CSRF* yang dikirim valid atau tidak (Ashari et al., 2022).

Menurut (Semastin et al., 2018) Melalui *pseudo random token* yang tersembunyi dianggap sebagai solusi paling efisien yang ada untuk mencegah serangan *CSRF* di aplikasi berbasis *web*. Di akhir *server*, *token* ini divalidasi terhadap *token* dalam sesi. Jika kedua *token* cocok, maka dapat dipastikan bahwa permintaan tersebut berasal dari asal yang sama.

Salah satu strategi keamanan yang diterapkan pada aplikasi berbasis *Website* adalah dengan menerapkan teknik enkripsi dan dekripsi data. Salah satu algoritma yang dipilih oleh National Institute of Standards and Technology (NIST) sebagai pemenang dalam kompetisi untuk menjadi kandidat *Advanced Encryption Standard* (*AES*) adalah algoritma *Rijndael*. Algoritma *Rijndael* ini dikembangkan oleh Vincent Rijmen dan Joan Daemen dari Belgia dan mendapatkan pengakuan sebagai standar *AES* pada bulan Oktober 2000 oleh NIST. *Rijndael* adalah algoritma kriptografi simetris berbasis blok yang mendukung pilihan panjang kunci 128 *bit*, 192 *bit*, dan 256 *bit*, serta ukuran blok yang dapat diatur secara independen. Proses enkripsi dalam algoritma *Rijndael* terdiri dari empat tahap transformasi *byte*, yaitu *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey*. Saat memulai proses enkripsi, *array state* yang telah dihasilkan akan melalui tahap transformasi *AddRoundKey*. Selanjutnya, *array state* akan mengalami berulang-ulang tahap transformasi *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey* sebanyak *Nr* kali. Proses ini, dalam konteks algoritma Rijndael, disebut sebagai "*round function*." Namun, tahap *round* terakhir memiliki sedikit perbedaan dibandingkan dengan tahap-tahap sebelumnya, karena pada tahap *round* terakhir, *array state* tidak akan melalui transformasi *MixColumns* (Sianipar et al., 2019).

Dari penjelasan tersebut, tampaknya menggunakan token sebagai metode keamanan pada aplikasi berbasis web adalah langkah yang cerdas untuk mengatasi resiko serangan *Cross-Site Request Forgery* (*CSRF*). *Token* akan mengalami proses enkripsi sebelum digunakan, sehingga membuatnya sangat sulit bagi penyerang untuk mencoba menebak konten yang ada di dalamnya.. Untuk itu penulis tertarik untuk melakukan penelitian dengan judul: **"IMPLEMENTASI KEAMANAN *WEBSITE* DARI SERANGAN *CROSS SITE REQUEST FORGERY* MENGGUNAKAN ALGORITMA RIJNDAEL PADA *FRAMEWORK CODEIGNITER*".**

1.2. Identifikasi Masalah

Berdasarkan paparan diatas maka dapat diidentifikasi beberapa masalah sebagai berikut:

1. Bagaimana terjadinya perubahan isi situs *web* tanpa sepengetahuan admin *Website*?
2. Bagaimana Adanya *request* ke *server* yang dieksekusi atas wewenang admin *Website* tanpa dikehendakinya?
3. Bagaimana *CSRF* mampu menipu situs *web* dengan cara membuat *request* yang seolah-olah *request* tersebut berasal dari *user* yang dipercaya?
4. Bagaimana rendahnya tingkat keamanan *form Website* tanpa adanya *token* yang akan melindungi dari serangan *CSRF*?
5. Bagaimana implementasi algoritma *Rijndael* untuk keamanan *website* dari serangan *CSRF*?

1.3. Batasan Masalah

Untuk menghindari pembahasan yang tidak terarah dan tidak teratur, penulis membatasi masalah sebagai berikut:

1. Teknik kriptografi yang akan digunakan untuk enkripsi *token* menggunakan Algoritma Rijndael atau AES 128 *bit* (*Advanced Encryption Standar*).
2. Penelitian ini berupa implementasi keamanan hak akses dari serangan *CSRF*.
3. Penelitian ini membahas tentang penambahan *token* yang telah dienkripsi pada setiap *method action POST* pada *form*, tidak sampai kedalam *database*.
4. Pada penelitian ini *token randomize* yang akan ditambahkan pada form tidak akan bisa dibuat secara manual, *token* akan otomatis di *generate* berdasarkan program algoritma *Rijndael*.

1.4. Rumusan Masalah

Berdasarkan latar belakang masalah yang telah disebutkan, dapat disimpulkan beberapa rumusan masalah sebagai berikut:

1. Apakah ada pengaruh berubahnya isi situs *web* tanpa sepengetahuan pemiliknya?
2. Apakah benar serangan ini mampu memanipulasi situs *web* yang seolah-olah *request* tersebut berasal dari *user* yang terautentikasi?
3. Apakah dengan menambahkan *token* enkripsi pada setiap *request* bisa terhindar dari serangan *CSRF*?

4. Apakah menambahkan *token randomize* disetiap *request* bisa meningkatkan kemanan *Website*?
5. Apakah dengan mengimplementasikan algoritma Rijndael bisa meningkatkan keamanan *website* dari serangan *CSRF*?

1.5. Tujuan Penelitian

1. Untuk mengetahui pengaruh keamanan *Website* dengan *token randomize* di setiap *request*.
2. Bagaimana pengaruh penambahan *token* dari serangan *Cross site request forgery*.
3. Bagaimana pengaruh serangan *Cross site request forgery* untuk memanipulasi situs *web* yang seolah – olah *request* tersebut berasal dari *Website* yang asli.
4. Untuk mengetahui pengaruh penambahan *token randomize* disetiap *request* bisa meningkatkan keamanan *Website*.
5. Bagaimana pengaruh berubahnya isi situs *web* akibat serangan *Cross site request forgery*.

1.6. Manfaat Penelitian

Adapun beberapa manfaat dari penelitian yang penulis buat sebagai berikut:

1. Manfaat teoritis; hasil peneilitian ini diharapkan dapat menambah pengetahuan dan memberikan sumbangan berupa ilmu yang berkaitan dengan keamanan *Website* khususnya dari serangan *Cross site request forgery*. Selain itu, penelitian ini diharapkan dapat bermanfaat sebagai sumber referensi untuk penelitian selanjutnya. Dan bagi peneliti lain, dapat dijadikan bahan perbandingan.

2. Manfaat praktis; penelitian ini bermanfaat bagi pembaca. Manfaat bagi pembaca dapat memberikan gambaran tentang bahayanya jika sebuah *Website* tidak ada keamanan dari serangan *Cross site request forgery*.

BAB II

LANDASAN TEORI

2.1. Landasan Teori

2.1.1. Website

Situs *web* merupakan koleksi dari dokumen-dokumen dan berbagai jenis sumber daya yang terkait satu sama lain, dihubungkan melalui *hyperlink* dan *URL*. Situs *website* dapat diakses melalui *browser web* di komputer atau perangkat seluler yang terhubung ke *internet*. *Website* biasanya di*hosting* di *server web* dan dapat menyertakan teks, gambar, video, dan konten multimedia lainnya. Situs *web* dapat diciptakan dan diurus oleh individu, kelompok, atau entitas bisnis, dan memiliki kemampuan untuk memenuhi beragam tujuan., seperti memberikan informasi, mempromosikan produk atau layanan, memungkinkan transaksi online, atau memfasilitasi komunikasi dan kolaborasi. *Web* bisa statis atau dinamis, artinya kontennya dapat diperbarui dan diubah seiring waktu. *Website* dapat dibangun menggunakan berbagai bahasa dan teknologi, seperti *HTML*, *CSS*, dan *JavaScript* untuk *front-end* dan bahasa seperti *PHP*, *Python*, *Ruby*, dan *.NET* untuk *back-end*. Beberapa *website* juga dibangun menggunakan *Conten Managemen System* (CMS) seperti *WordPress*, *Joomla*, atau *Drupal* yang memudahkan pengguna non-teknis untuk membuat, mengedit, dan mengelola konten *website*.

Menurut (Al et al., 2021) Situs *web* adalah rangkaian halaman yang berfungsi sebagai wadah untuk menyampaikan berbagai informasi, seperti teks, gambar, animasi, audio, atau gabungan dari semuanya. Halaman-halaman ini dapat

berupa konten yang bersifat statis atau dinamis, dan semuanya terkait erat satu sama lain serta terkoneksi melalui jaringan halaman.

Kemudian menurut (Abdullah, 2018) Situs *web* bisa dijelaskan sebagai sekumpulan halaman yang berisi berbagai informasi digital seperti teks, gambar, animasi, suara, video, atau campuran dari semua itu. Informasi ini tersedia melalui *internet* dan dapat diakses oleh siapa saja di seluruh dunia. Pembuatan halaman-halaman situs web menggunakan bahasa standar, yaitu *HTML*. *Script HTML* ini kemudian diterjemahkan oleh peramban *web* sehingga dapat ditampilkan sebagai informasi yang dapat dibaca oleh siapapun. Situs *web* memungkinkan penyebaran informasi *global* secara mudah dan luas.

Website dapat digunakan untuk berbagai tujuan, seperti menyediakan informasi produk atau jasa, memberikan layanan komunikasi, atau sebagai *platform* untuk berinteraksi dengan pengguna lain. *Website* juga dapat digunakan sebagai alat untuk meningkatkan *brand* atau reputasi perusahaan, atau sebagai sarana untuk menjual produk atau jasa. *Website* dapat dibuat dan dioperasikan oleh individu, perusahaan, atau organisasi, dan dapat diakses oleh siapa saja yang memiliki koneksi internet.

Secara keseluruhan, situs *web* dapat diklasifikasikan ke dalam tiga kategori, yakni situs web dengan karakter statis, dinamis, serta situs web yang memiliki unsur interaktivitas. (Rohi Abdulloh, 2018).

1. *Website* Statis

Situs *web* tipe statis merujuk pada jenis situs yang memiliki konten yang tidak mengalami perubahan secara rutin, sehingga kontennya tetap konstan dari waktu ke

waktu. Jenis situs ini biasanya dimanfaatkan untuk menampilkan profil pribadi atau organisasi, seperti profil perusahaan atau entitas lainnya.

2. *Website* Dinamis

Situs *web* dengan karakter dinamis merujuk pada jenis situs yang secara berkelanjutan diperbaharui oleh pengelola atau pemiliknya. Biasanya, jenis situs ini dimiliki oleh perusahaan atau individu yang bergerak dalam bisnis yang terkait dengan internet. Contohnya mencakup blog atau situs web berita.

3. *Website* Interaktif

Website interaktif, pada prinsipnya, masuk ke dalam jenis situs web yang memiliki karakter dinamis, di mana kontennya terus diperbarui secara berkala. Namun, perbedaannya adalah bahwa informasi di situs web ini tidak hanya dapat diubah oleh pemilik situs, tetapi juga dapat disunting oleh pengguna situs itu sendiri. Sebagai contoh, platform jejaring sosial seperti *Facebook* dan *Twitter*, atau situs web pasar *online* seperti *Bukalapak*, *Tokopedia*, dan sejenisnya, memungkinkan interaksi dan kontribusi dari pengguna.

Sedangkan menurut (Ronaldo & Pasha, 2021) terdapat 2 jenis *website*, yaitu:

1. *Website* Statis

Website statis adalah *Website* yang isinya tidak mudah diubah oleh pengguna. Perubahan konten hanya dapat dilakukan dengan mengedit kode pada halaman *website* atau melalui *database*.

2. *Website* Dinamis

Web dinamis adalah jenis *website* yang dapat beradaptasi dan secara otomatis menyesuaikan perubahan konten tanpa perlu merubah struktur kode *website*.

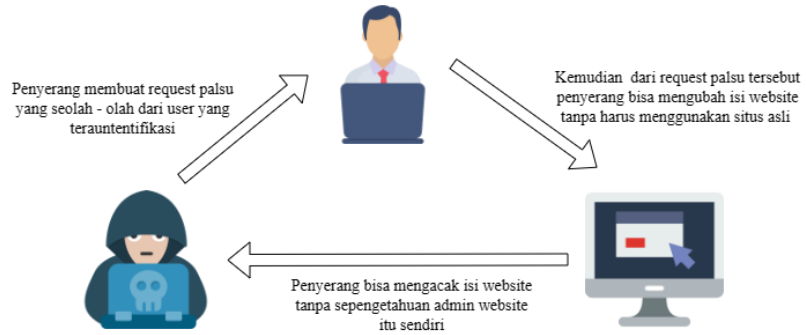
2.1.2. Cross Site *Request* Forgery (*CSRF*)

Cross Site Request Forgery (*CSRF*) adalah jenis serangan keamanan *web* dimana *attacker* menyalahgunakan kepercayaan yang diberikan oleh *browser* kepada *website* yang sah untuk mengirim permintaan yang tidak diinginkan pada *website* lain. Ini dilakukan dengan menyuntikkan *link* atau *script* ke dalam *website* yang sah yang diakses oleh korban, yang kemudian mengirim permintaan yang tidak diinginkan dari korban ke *website* yang tidak sah. Tujuan dari serangan ini biasanya adalah untuk melakukan tindakan yang merugikan pada *website* yang diserang. Menurut (Kour MTech Student, 2020) Serangan *CSRF* adalah ketika penyerang mampu membuat permintaan atas nama pengguna. Penyerang mengambil keuntungan dari fakta bahwa pengguna sudah diautentikasi ke aplikasi *web* atau situs *web* tertentu.

Serangan *CSRF* dapat dilakukan dalam langkah yang berbeda tergantung pada jenis serangan *CSRF*. *CSRF* terutama dapat diklasifikasikan tergantung pada cara penyerang membuat korban mengirim permintaan *HTTP* palsu dan juga tergantung pada keadaan korban *CSRF* dengan aplikasi *web* tepercaya (Lalia & Moustafa, 2019).

Dalam situasi di mana serangan *CSRF* berhasil, pelaku menciptakan situasi di mana korban tidak sengaja melakukan tindakan yang tidak dikehendaki, seperti mengganti alamat *email* pada akun atau menginputkan data yang tidak sah. Tingkat dampaknya tergantung pada keparahan tindakan yang dilakukan oleh penyerang, dimana dalam beberapa kasus, penyerang dapat memperoleh akses sepenuhnya ke dalam akun pengguna. Terutama jika korban memiliki hak istimewa khusus dalam

aplikasi, penyerang bisa saja mengambil alih seluruh kontrol atas data dan fitur yang ada dalam aplikasi tersebut. (Patil et al., 2019).



Sumber: (Rankothge & Randeniya, 2020)

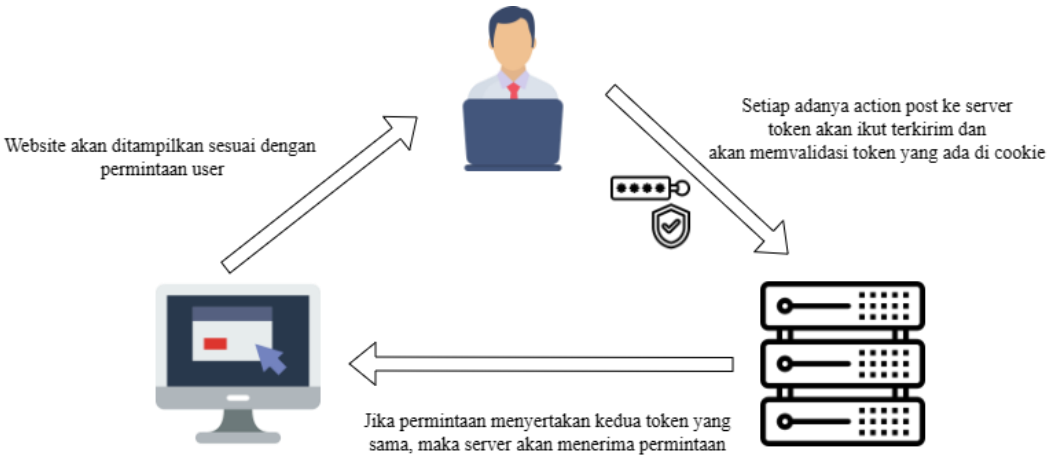
Gambar 2.1 Cara Kerja *CSRF*

Serangan *CSRF* (*Cross Site Request Forgery*) adalah jenis serangan keamanan yang mengeksploitasi kepercayaan aplikasi *web* terhadap pengguna yang sudah terauthentikasi. Berikut adalah cara kerja serangan *CSRF*:

1. Penyerang menargetkan pengguna yang seolah sudah login ke aplikasi *web* tertentu.
2. Penyerang membuat halaman *web* atau link yang mengandung *script* untuk melakukan *request* ke aplikasi *web* yang dituju.
3. Pengguna yang terkena serangan tidak sadar melakukan *request* yang dimaksudkan oleh penyerang, seperti membuat atau memodifikasi data.
4. *Request* tersebut akan diterima oleh aplikasi *web* seolah-olah dikirim oleh pengguna yang terauthentikasi, meskipun sebenarnya *request* tersebut dibuat oleh penyerang.

Serangan *CSRF* sangat berbahaya karena dapat mengeksploitasi aplikasi *web* tanpa mengetahui sandi atau informasi lain dari pengguna. Oleh karena itu, pencegahan yang efektif dalam mengatasi serangan *CSRF* adalah menggunakan *token* Anti-*CSRF*.

Cara yang paling efektif untuk mencegah serangan *CSRF* adalah dengan menambahkan random *token* yang tersembunyi dianggap sebagai solusi paling efisien yang ada untuk mencegah serangan *CSRF* di aplikasi berbasis *web*. Di akhir *server*, *token* ini divalidasi terhadap *token* dalam sesi. Jika kedua *token* cocok, maka dapat dipastikan bahwa permintaan tersebut berasal dari asal yang sama. Dengan memasukkan *token* melalui bidang tersembunyi, akses penyerang ke *token* ditolak (Semastin et al., 2018).



Gambar 2.2 Cara kerja *token* anti *CSRF*

Token Anti-*CSRF* (*Cross Site Request Forgery*) adalah sebuah *token* yang digunakan untuk melindungi aplikasi *web* dari serangan *CSRF*. Cara kerjanya sebagai berikut:

1. *Server* meng*generate token* unik dan menyimpannya di *cookie*.
2. *Token* tersebut dikirimkan ke *client* melalui *header HTTP* atau sebagai parameter dalam *form*.
3. *Client* menyimpan *token* tersebut dan menambahkannya ke setiap *request* yang dikirimkan ke *server*.

4. *Server* memvalidasi *token* dalam setiap *request* untuk memastikan bahwa *request* tersebut dikirimkan oleh *client* yang sah.
5. Jika *token* tidak valid, *server* menolak *request* tersebut dan menampilkan pesan *error*.

Dengan menggunakan *token* Anti-*CSRF*, serangan *CSRF* dapat ditekan karena setiap *request* harus disertai dengan *token* yang valid dan hanya dapat diterima oleh *server* jika *token* tersebut benar.

2.1.3. Algoritma *Rijndael*

Kriptografi merupakan ilmu yang mendalami metode-metode untuk melindungi data dan informasi. Ini dilakukan dengan menggunakan algoritma matematis untuk mengenkripsi (membuat informasi tidak dapat dibaca tanpa kunci) dan mengdeskripsi (membuat informasi dapat dibaca kembali dengan kunci) informasi. Untuk melindungi informasi, Kriptografi menggunakan berbagai metode Untuk melindungi informasi, Kriptografi menggunakan berbagai metode. Saat mengirimkan dan menyimpan data melalui media elektronik, diperlukan tindakan untuk memastikan keselamatan dan integritas dari data yang dikirim.

Secara umum, terdapat dua jenis teknik kriptografi yang ada, yaitu kriptografi simetris dan kriptografi asimetris. Kriptografi simetris memanfaatkan kunci yang identik untuk melaksanakan proses enkripsi serta dekripsi data, dan dikenal juga sebagai *Private Key Cryptography* (Riza et al., 2018).

Menurut (Al-Khowarizmi, 2021) algoritma kriptografi dibagi menjadi 3 bagian berdasarkan kunci yang dipakainya yaitu:

1. Kriptografi Simetris

Hill cipher adalah metode kriptografi yang menggunakan algoritma kunci untuk menjaga kerahasiaan data saat dikirimkan antara pengirim dan penerima. *Hill cipher* menggunakan prinsip dari Aritmatika modular dalam proses enkripsi dan dekripsi datanya.

2. Kriptografi Asimetris

Algoritma asimetris (biasa disebut algoritma kunci publik atau kriptografi kunci publik) menggunakan dua jenis kunci, yaitu kunci yang publik (kunci publik) dan kunci rahasia (kunci rahasia). Kunci publik adalah kunci digunakan untuk mengenkripsi pesan. Pada saat yang sama, kuncinya digunakan untuk mendekripsi pesan. Kunci publiknya adalah publik, yang artinya tidak akan dirahasiakan, jadi siapa pun dapat melihatnya. Pada saat yang sama, kunci rahasia adalah kunci yang menyimpan rahasia, dan hanya orang tertentu saja yang boleh mengetahuinya.

3. Kriptografi *Hybrid*

Kriptografi *hybrid* merupakan metode enkripsi yang menggabungkan prinsip dari kriptografi simetris dan asimetris. Pada metode ini, kedua jenis kriptografi digabungkan. Proses enkripsi dimulai dengan negosiasi kunci rahasia (*private key/session key*) yang digunakan oleh kedua belah pihak menggunakan kriptografi asimetris.

Pada tahun 1997, *National Institute of Standard and Technology* (NIST) mengumumkan bahwa untuk meningkatkan sistem keamanan, saatnya untuk menciptakan standar baru untuk algoritma penyandian yang disebut *Advanced Encryption Standard* (*AES*). Tujuan dari *AES* adalah untuk menggantikan algoritma

DES. Setelah melewati beberapa tahap seleksi, algoritma *Rijndael* dipilih sebagai algoritma kriptografi *AES* pada tahun 2000 (Sianipar et al., 2019).

AES dapat diartikan sebagai algoritma enkripsi simetris blok yang sangat populer. Prosesnya melakukan operasi pada blok 8-*bit*, mengambil *plaintext* dengan ukuran blok 128 *bit*, 192 *bit*, dan 256 *bit*. Kuncinya adalah digambarkan sebagai matriks persegi *byte* (Hidayat, 2019).

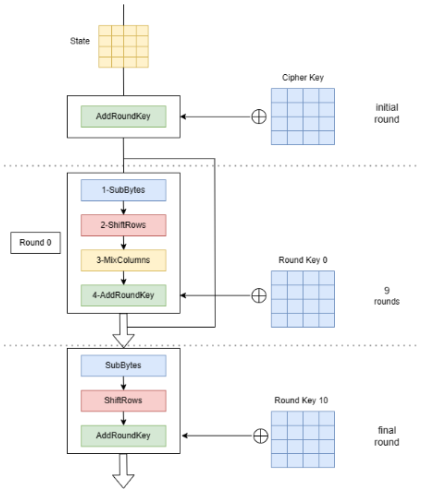
Tabel 2.1 Urutan data algoritma AES

	Panjang Kunci	Panjang Blok	Jumlah Putaran
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Tabel diatas menyajikan informasi tentang variasi tipe dari algoritma *AES*, yang dibedakan berdasarkan ukuran kunci, ukuran blok, dan jumlah putaran yang berbeda-beda.

Dalam ranah kriptografi, *AES* (*Advanced Encryption Standard*) adalah standar enkripsi yang menggunakan kunci simetris. *AES* memiliki keunggulan karena panjang kunci minimal yang digunakan adalah 128 *bit*, sehingga membuatnya sangat terlindungi dari serangan *exhaustive key search* dengan teknologi saat ini. Dengan panjang kunci sebesar 128-bit, terdapat sekitar 2^{128} = 3,4 x 10^{38} kemungkinan kombinasi kunci yang mungkin digunakan. Jika kita memakai

komputer tercepat yang dapat mencoba 1 juta kunci per detik, dibutuhkan waktu sekitar 5,4 x 10^{24} tahun untuk mencoba semua kemungkinan kunci tersebut.. Jika menggunakan komputer tercepat yang dapat mencoba 1 juta kunci per detik, akan diperlukan waktu selama 5,4 x 10^{24} tahun untuk mencoba semua kemungkinan kunci. Namun, jika digunakan komputer tercepat yang dapat mencoba 1 juta kunci per milidetik, waktu yang diperlukan untuk mencoba semua kemungkinan kunci adalah 5,4 x 10^{18} tahun (Tahir et al., 2020).



Sumber: (Prameshwari & Sastra, 2018)

Gambar 2.3 Diagram proses enkripsi Algoritma AES

Di awal proses enkripsi, data masukan yang telah disalin ke dalam keadaan (*state*) akan mengalami tahap pertama yang disebut *AddRoundKey*. Selanjutnya, keadaan akan melewati serangkaian transformasi, yaitu *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey*, yang diulangi sebanyak Nr putaran. Seluruh rangkaian ini dikenal sebagai fungsi putaran (*round function*) dalam algoritma *AES*. Penting dicatat bahwa pada putaran terakhir, keadaan tidak mengalami transformasi *MixColumns*.

2.1.3.1. AddRoundKey

AddRoundKey adalah proses operasi dalam enkripsi *blok cipher* seperti AES (*Advanced Encryption Standard*). Dalam enkripsi *blok cipher*, *AddRoundKey* digunakan untuk menambahkan kunci enkripsi ke setiap *blok plainteks* sebelum proses enkripsi sebenarnya dimulai. Proses ini memastikan bahwa setiap blok plainteks memiliki kunci yang unik sebelum dienkripsi.

AddRoundKey adalah operasi XOR (eksklusif OR) antara setiap *byte* pada matriks *state* (berisi ciphertext) dan setiap *byte* pada *roundkey* yang telah dibangkitkan sebelumnya (Cristy & Riandari, 2021). Dalam enkripsi *blok cipher*, XOR memastikan bahwa hasil enkripsi tidak bergantung pada kunci enkripsi, dan setiap perubahan pada kunci akan mempengaruhi hasil enkripsi.

2.1.3.2. SubBytes

SubBytes adalah proses menggantikan setiap *byte* dalam blok enkripsi dengan *byte* lain yang terkait dengan *byte* asli melalui tabel subsitusi yang disebut *S-Box*. *S-Box* adalah tabel 8-*bit* yang mengandung informasi mengenai bagaimana setiap *byte* dalam blok enkripsi harus digantikan.

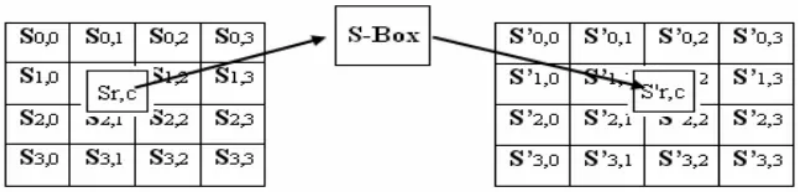
SubBytes adalah langkah transformasi *byte* di mana setiap elemen dalam keadaan (state) akan dihubungkan dengan tabel substitusi yang dikenal sebagai *S-Box*.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Sumber: (Prayudha, 2019)

Gambar 2.4 *S-Box Rijndael*

Untuk masing-masing *byte* dalam rangkaian keadaan (*state*), seperti S[r, c] = xy (dengan xy merupakan representasi heksadesimal dari nilai S[r, c]), penggantian nilai tersebut, yang disebut sebagai S'[r, c], sesuai dengan entri yang ada di tabel substitusi yang terletak di persimpangan baris x dan kolom y. (Prayudha, 2019).



Sumber: (Prayudha, 2019)

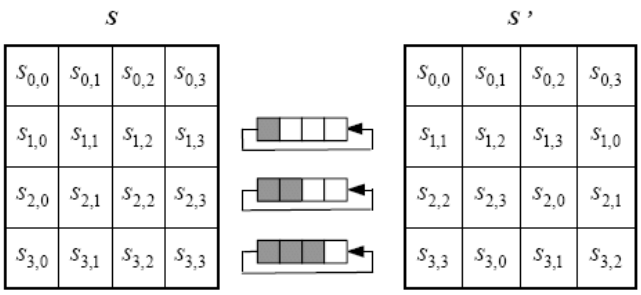
Gambar 2.5 Pengaruh pemetaan pada setiap *Byte* dalam *state*

2.1.3.3. ShiftRows

ShiftRows adalah proses operasi dalam enkripsi *blok cipher* seperti AES (*Advanced Encryption Standard*). Dalam enkripsi *blok cipher*, ShiftRows digunakan untuk menyebarkan informasi plainteks ke seluruh blok enkripsi dan membuat enkripsi lebih sulit diterobos.

Menurut (Marsiani et al., n.d.) *ShiftRows* adalah tahap yang melibatkan pergeseran elemen dalam blok atau tabel, dengan pergeseran yang diterapkan pada setiap baris. Penting dicatat bahwa pada baris pertama, tidak ada pergeseran yang dilakukan. Pada baris ke 2 digeser sebanyak 1 *byte*s lalu setelah itu baris ke 3 digeser sebanyak 2 *byte*s dan yang keempat dilakukan pergeseran 3 *byte*s.

Sedangkan menurut (Putri Harum & Arifianto, 2019) *ShiftRows* adalah transformasi yang melibatkan pergeseran baris dalam matriks *stateplain*. Pada proses ini, baris pertama (r=1) digeser satu byte, baris kedua (r=2) digeser dua byte, dan baris ketiga (r=3) digeser tiga byte. Namun, perlu dicatat bahwa baris keempat (r=0) tidak mengalami pergeseran.



Sumber: (Marsiani et al., n.d.)

Gambar 2.6 Proses ShiftRows

2.1.3.4. MixColumns

MixColumns adalah proses operasi dalam enkripsi blok cipher seperti AES (*Advanced Encryption Standard*), MixColumns adalah proses mengacak kolom dalam blok enkripsi dengan menggunakan operasi matematis. Setiap kolom dalam blok enkripsi diterjemahkan menjadi kolom baru dengan informasi dari kolom lain dalam blok enkripsi.

Menurut (Irsyad & Sitio, 2019) Transformasi *MixColumns* bertujuan untuk mempermutasi data pada setiap kolom dalam *array state*. Dalam proses

MixColumns, setiap kolom dalam array state dikalikan dengan polinomial (x)mod(x4+1).

Transformasi *MixColumns* merupakan operasi perkalian antara matriks *MixColumns Rijndael* dan setiap kolom dalam array *state* (Saputra Djong & Siswanto, 2022). Transformasi *MixColumns* dapat digambarkan dalam perkalian matrik seperti gambar dibawah

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

Sumber: (Saputra Djong & Siswanto, 2022)

Gambar 2.7 Transformasi MixColumns

Perkalian matriks *MixColumns* dalam *AES* akan menghasilkan empat nilai tambahan untuk masing-masing kolom dalam *matriks state*. Persamaan yang digunakan untuk menghitung nilai-nilai baru ini dirumuskan seperti berikut:

$$S'_{0,c} = (02 \bullet S_{0,c}) \text{ XOR } (03 \bullet S_{1,c}) \text{ XOR } S_{2,c} \text{ XOR } S_{3,c}$$

$$S'_{1,c} = S_{0,c} \text{ XOR } (02 \bullet S_{1,c}) \text{ XOR } (03 \bullet S_{2,c}) \text{ XOR } S_{3,c}$$

$$S'_{2,c} = S_{0,c} \text{ XOR } S_{1,c} \text{ XOR } (02 \bullet S_{2,c}) \text{ XOR } (03 \bullet S_{3,c})$$

$$S'_{3,c} = (03 \bullet S_{0,c}) \text{ XOR } S_{1,c} \text{ XOR } S_{2,c} \text{ XOR } (02 \bullet S_{3,c})$$

Sumber: (Saputra Djong & Siswanto, 2022)

2.1.4. *Framework Codeigniter*

Kerangka kerja (*framework*) adalah sekumpulan perintah atau program dasar yang dapat digunakan kembali untuk menyelesaikan masalah yang lebih kompleks dan membantu dalam pembuatan aplikasi baru atau aplikasi kompleks tanpa harus membuat program dari awal. Salah satu *framework PHP* adalah

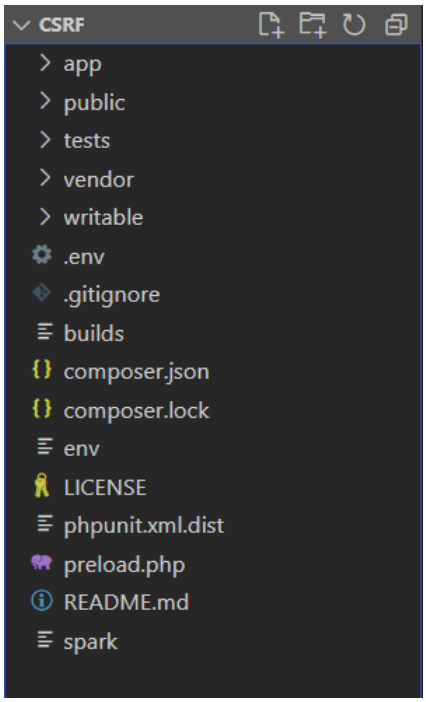
Codeigniter. *Codeigniter* diperkenalkan pertama kali kepada publik pada tanggal 28 Februari 2006 yang dikembangkan oleh *Ellislab* pada awalnya. *Codeigniter* dikembangkan dengan menggunakan banyak libraries, helpers, dan sub-komponen dari basis kode *ExpressionEngine*. Pada tahun 2014, *Codeigniter* diakuisisi oleh *British Columbia Institute of Technology* dan secara resmi diumumkan sebagai proyek yang dikelola oleh komunitas (Priyanto Hidayatullah, 2021).

Framework Codeigniter merupakan salah satu *framework PHP* yang popular pada saat ini, karena memiliki kelebihan dibanding *framework* lain. Berikut adalah kelebihan *Codeigniter* dibanding *framework* lain:

1. Paling mudah untuk dipelajari, terutama untuk pemula.
2. *Framework* yang berukuran kecil. *Codeigniter* 4 berukuran 1.2 MB dan 6 MB untuk *user guide*.
3. Menggunakan *design pattern Model View Controller* (*MVC*). Dengan menggunakan *MVC* membuat kode program akan menjadi lebih terstruktur.
4. Kemanan data terjamin. *Codeigniter* memiliki fitur kemanan yang bisa digunakan untuk melindungi dari serangan *CSRF* dan *XSS*.
5. Dokumentasi lengkap. *Codeigniter* menyediakan dokumentasi *offline* yang didapatkan ketika kita mendownload *Codeigniter* dan dokumentasi *online* pada *website Codeigniter*.
6. Konfigurasi awal hampir tidak ada. Sebagian besar konfigurasi *Codeigniter* dilakukan dengan konvensi, misalnya meletakkan *model* dalam *folder* "*model*". Namun untuk konfigurasi lebih lanjut, tersedia beberapa konfigurasi yang terdapat pada *folder* config.
7. *Codeigniter* terus di-*update* secara berkelanjutan.

Sumber: (Priyanto Hidayatullah, 2021)

Codeigniter mempunyai struktur *folder* yang disusun secara rapi dan sistematis, secara default struktur *folder Codeigniter* 4 adalah sebagai berikut:



Gambar 2.8 Struktur *folder Codeigniter* 4

Berikut adalah penjelasan dari struktur *folder* pada *Codeigniter* 4:

"*app*": *Folder* ini berisi aplikasi seperti *controller*, *model*, *view*, dan lainnya.

"*public*": *Folder* ini berisi *file* publik seperti gambar, CSS, dan JavaScript.

"*tests*": *Folder* ini berisi *file* tes untuk memastikan bahwa aplikasi bekerja dengan baik.

"*vendor*": *Folder* ini berisi library pihak ketiga yang diinstal melalui *Composer*.

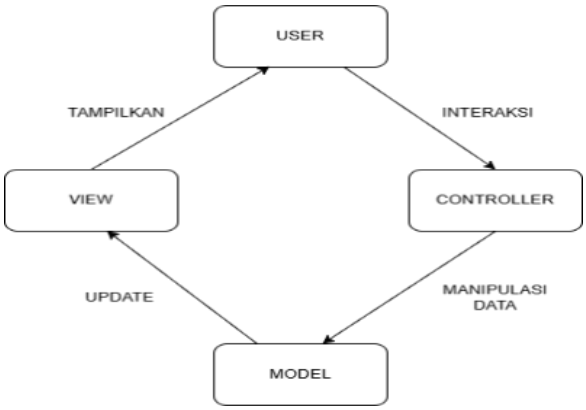
"*writable*": *Folder* ini digunakan sebagai tempat penyimpanan berkas yang dapat ditulis oleh aplikasi, seperti sesi, *cache*, dan log.

"*.env*": *File* ini berisi pengaturan lingkungan aplikasi Anda, seperti kredensial *database*, URL basis, dan lainnya.

"*env*": *Folder* ini berisi *file* lingkungan aplikasi.

"*composer.json*": *File* ini berisi daftar paket library yang diperlukan aplikasi Anda. **"*composer.lock*"**: *File* ini berisi versi terikat paket library yang digunakan oleh aplikasi.

Codeigniter merupakan sebuah kerangka kerja (*framework*) yang digunakan dalam pengembangan aplikasi *web*, dengan prinsip-prinsip dasar *Model*, *View*, dan *Controller* (*MVC*). *Framework PHP* ini dapat membantu seorang pengembang *web* dalam membangun situs dengan lebih efisien karena menyediakan beragam sumber daya yang komprehensif. (Sulistiani & Hendra Saputra, 2020)



Sumber: (Priyanto Hidayatullah, 2021)

Gambar 2.9 Konsep MVC *Codeigniter*

MVC (*Model-View-Controller*) adalah arsitektur desain yang digunakan dalam pengembangan aplikasi *web*. *Codeigniter* adalah *framework PHP* yang menggunakan arsitektur *MVC*. Berikut adalah cara kerja *MVC* di *Codeigniter*:

1. *Model*: *Model* menangani logika bisnis dan akses data. *Model* menerima permintaan dari *Controller* dan mengambil data dari basis data atau sumber data lainnya.
2. *View*: *View* menangani tampilan data ke pengguna. *View* menerima data dari *Model* dan menampilkan data tersebut ke pengguna melalui template HTML atau format lainnya.

3. *Controller*: *Controller* menjembatani antara *Model* dan *View*. *Controller* menerima permintaan dari pengguna melalui *URL* dan menentukan bagaimana menangani permintaan tersebut. *Controller* meminta data dari *Model* dan memberikan data tersebut ke *View* untuk ditampilkan.

Di *Codeigniter*, setiap *controller* harus memiliki *method* yang sesuai dengan *URL* yang diterima, dan method tersebut akan menentukan bagaimana menangani permintaan tersebut, misalnya memanggil *Model* dan *View* yang sesuai. Setelah proses selesai, *Controller* akan mengirimkan respons ke pengguna.

Dengan menggunakan arsitektur *MVC*, *Codeigniter* membantu memisahkan logika bisnis, akses data, dan tampilan, sehingga mempermudah proses pemeliharaan dan pengembangan aplikasi *web*.

2.1.5. *Unified Modelling Language* (*UML*)

UML (*Unified Modeling Language*) adalah sebuah bahasa *modeling* yang digunakan untuk menggambarkan dan me*model*kan sistem dan perangkat lunak. *UML* digunakan untuk menggambarkan aktivitas dan perilaku dalam sebuah sistem, termasuk interaksi antara objek, proses, dan tipe data. *UML* membantu dalam memahami dan mengevaluasi sistem dan memudahkan proses pengembangan *software*.

Menurut (Wira et al., 2019) *UML* (*Unified Modeling Language*) adalah sebuah bahasa pemodelan yang terintegrasi dan memiliki penggunaan yang luas di dunia industri. Bahasa ini digunakan untuk menentukan kebutuhan, melakukan analisis dan perancangan, serta menggambarkan arsitektur dalam konteks pemrograman berorientasi objek.

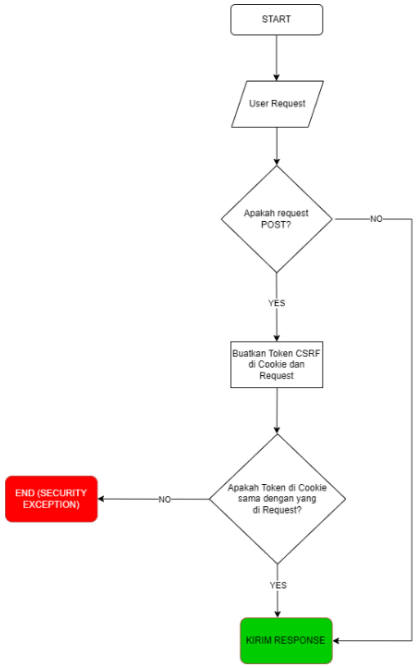
UML menyediakan cara visual untuk menjelaskan bagian-bagian sistem dalam perangkat lunak, mempermudah spesifikasi, konstruksi, dan dokumentasi. Ini membantu programmer/developer membangun perangkat lunak dengan lebih mudah (Sumiati et al., 2021).

2.1.6. Flowchart

Flowchart adalah sebuah diagram yang menggambarkan aliran proses atau aktivitas secara sistematis dan visual. *Flowchart* menggunakan simbol-simbol standar untuk menunjukkan aliran data atau proses, seperti operasi, alur logika, kondisi dan aktivitas. *Flowchart* membantu memvisualisasikan dan memahami proses yang kompleks, sehingga mempermudah proses troubleshooting dan pemecahan masalah.

Menurut (Syamsiah, 2019) *Flowchart* adalah diagram yang menunjukkan alir logika dalam prosedur atau program sistem. *Flowchart* bertujuan untuk memberikan representasi visual dari alur prosedur atau program tersebut. *Flowchart* adalah suatu teknik yang digunakan untuk visualisasi dan mengilustrasikan langkah-langkah yang diperlukan dalam menyelesaikan suatu masalah, dengan memanfaatkan simbol-simbol yang sederhana dan mudah dipahami. *Flowchart* mempermudah dalam pemahaman dan visualisasi tahapan-tahapan dalam proses pemecahan masalah.

Flowchart adalah representasi *visual* dari alur kerja suatu proses dalam sistem yang telah dirancang, dirancang untuk memudahkan pemahaman dan penjelasan. Alat ini menggunakan simbol-simbol khusus untuk menggambarkan dengan detail urutan langkah-langkah proses serta interaksi antara instruksi-instruksi dengan proses lain dalam program tersebut (Unang Achlison, 2020).



Gambar 2.10 Flowchart Perlindungan *CSRF*

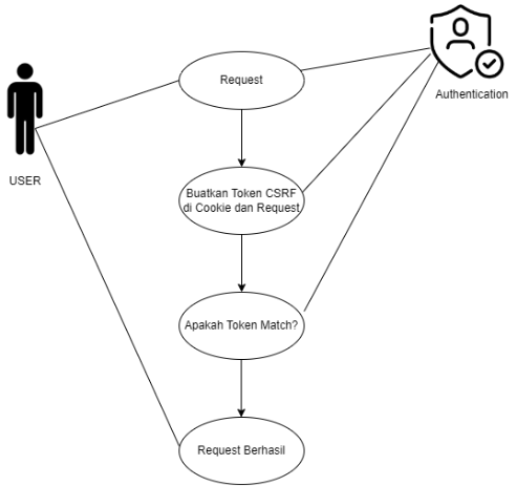
Dari *flowchart* diatas dapat dinarasikan bahwa saat *user* membuat *request*, aplikasi juga akan menambahkan *token* unik sebagai bagian dari form. Saat *request* tersebut dikirimkan, *token* juga akan dikirimkan bersamaan. *Server* akan memverifikasi *token* yang diterima dengan *token* yang disimpan di *server*. Jika *token* yang diterima cocok dengan *token* yang disimpan, maka permintaan diterima dan diproses. Jika *token* tidak cocok, maka permintaan ditolak dan diproses sebagai serangan *CSRF*.

2.1.7. Use case

Use Case adalah penjelasan mengenai bagaimana suatu sistem berinteraksi dengan pengguna atau sistem lainnya.. *Use Case* menggambarkan bagaimana sistem akan memenuhi kebutuhan pengguna atau mengatasi masalah yang ada. *Use Case* memfokuskan pada fungsi-fungsi yang diterima sistem dan bagaimana sistem akan bereaksi terhadap berbagai situasi yang mungkin terjadi.

Diagram Use Case mengilustrasikan cara di mana interaksi terjadi antara sistem informasi yang akan dibangun dengan satu atau lebih aktor. (Darwis et al., 2020). Use case membantu untuk menyajikan visualisasi tentang bagaimana interaksi terjadi antara pengguna sistem dan sistem itu sendiri.

Menurut (Amazon et al., 2021) *Use case* adalah antarmuka sistem yang merespons perintah dari seorang aktor dalam bentuk peristiwa (*event*). Ini berhubungan dengan pelaksanaan dalam rangkaian pengiriman pesan antara objek-objek yang terlibat.



Gambar 2.11 Use Case Diagram

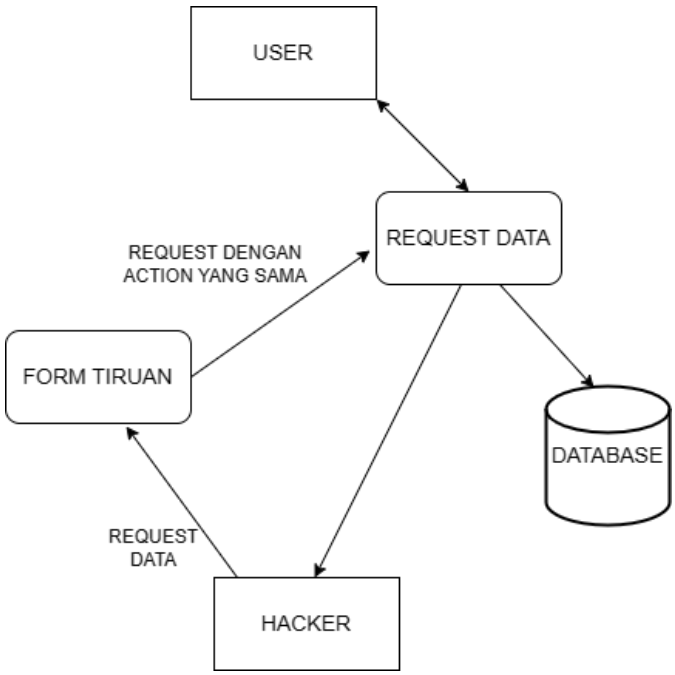
Beradasarkan *use case* diatas dapat dijelaskan saat *user* membuat *request*, sistem akan membuat autentikasi dengan memvalidasi *token* yang ada di *cookie* dan di *request*. Kemudian jika *token* yang ada *cookie* dan *request* cocok maka sistem akan menjalankan *request* tersebut.

2.1.8. Data Flow Diagram (DFD)

Data Flow Diagram (DFD) adalah suatu metode untuk mengilustrasikan sebuah sistem sebagai kumpulan proses fungsional yang saling terhubung oleh arus data. Metode ini bisa digunakan oleh para ahli sistem untuk membuat model sistem

secara manual atau dengan bantuan komputer (Handrianus Pranatawijaya et al., 2019).

Menurut (Muliadi et al., 2020) *Data Flow Diagram* (DFD) adalah diagram yang menyajikan visualisasi dari arus data dalam suatu organisasi. Ini menggunakan simbol-simbol tertentu untuk menjelaskan bagaimana data berpindah dalam sistem.



Sumber: (Mahdi Maulana Lubis et al., 2022)

Gambar 2.12 Data Flow Diagram *CSRF* Attack

Berdasarkan *data flow* pada gambar di atas diketahui serangan *CSRF* dilakukan dengan hacker membuat form tiruan dengan method dan action yang sama dengan *website* yang ingin diserang setelah itu hacker melakukan *request* ke *website* asli , *website* asli akan mengirimkan data ke *hacker*.

2.1.9. *Black Box Testing*

Black Box Testing adalah teknik pengujian yang sederhana karena hanya memerlukan informasi mengenai rentang nilai yang diharapkan dari data yang akan digunakan. Penaksiran jumlah data uji dapat dihitung berdasarkan jumlah *field* data yang akan diuji, aturan entri yang harus dipatuhi, dan kasus-kasus batas atas serta batas bawah yang relevan. Dengan metode ini, kita dapat mengevaluasi apakah sistem masih mampu mengatasi masukan data yang tidak sesuai harapan, yang dapat berpotensi menghasilkan data yang tidak valid. (Fahrozi et al., 2023).

BAB III

METODOLOGI PENELITIAN

3.1. Jenis Penelitian

Pendeketan pada penelitian ini dapat dilihat dari tujuan untuk menemukan solusi dalam suatu masalah yang secara langsung pada keamanan *website*. Dalam penelitian ini, peneliti memanfaatkan pendekatan penelitian kuantitatif. Metode kuantitatif berupa analisis keamanan *website* dari serangan *Cross Site Request Forgery* (*CSRF*) dengan *token randomize* menggunakan algoritma *AES* dan diimplementasikan di setiap *form request*.

Penelitian kuantitatif merupakan penelitian yang bersifat obyektif, menguji teori, bersifat generalisasi dan menguji hipotesis dengan cara statistik. Penelitian kuantitatif melibatkan penggunaan data dalam bentuk angka untuk menganalisis informasi tentang topik tertentu dan menemukan pengetahuan baru. Dalam penelitian ini, data disusun dan dianalisis dengan memakai teknik-teknik statistik untuk menghasilkan informasi yang berguna tentang topik yang sedang diteliti (Indra Prasetia, 2022).

3.2. Teknik Pengambilan Sampel

Dalam penelitian sebuah sampel harus dipilih dengan tepat agar dapat merepresentasikan populasi penelitian. Sampel yang dipilih harus mencerminkan karakteristik populasi yang ingin diteliti, sehingga informasi yang diperoleh dari sampel dapat dianggap mewakili informasi dari populasi secara keseluruhan. Oleh karena itu, dibutuhkan metode pemilihan sampel yang tepat untuk mendapatkan sampel yang mewakili karakteristik populasi dengan baik. Dengan demikian,

informasi yang diperoleh dari sampel yang baik dapat merepresentasikan informasi dari populasi secara keseluruhan (Indra Prasetia, 2022).

Teknik pengambilan sampel (*sampling*) pada serangan *CSRF* menggunakan *penetration testing*. *Penetration testing* adalah metode pengujian keamanan yang bertujuan untuk mengevaluasi sistem keamanan suatu aplikasi. Teknik pengambilan sampel pada penetration testing bertujuan untuk mengevaluasi seberapa efektif sistem keamanan *website* dalam melindungi *website* dari serangan tersebut.

Pada penelitian ini penulis akan melakukan pengujian *penetration testing* sebanyak sepuluh kali untuk mengambil sampel kepada *website* yang sudah terlindungi dari serangan *CSRF*.

3.3. Teknik Pengumpulan Data

Pengumpulan data adalah suatu prosedur atau pendekatan yang diterapkan oleh peneliti untuk menghimpun informasi yang diperlukan guna mencapai sasaran dalam penelitian (Benu L Fred & Benu S Agus, 2019). Hal yang harus dijelaskan adalah mengenai langkah-langkah analisis dan proses untuk menyimpulkan hasil. Pada penelitian ini menggunakan data primer, yaitu Data yang diperoleh secara internal melalui observasi langsung atau metode sejenis.

Adapun Teknik pengumpulan data pada penelitian ini adalah:

1. Observasi: melakukan pengamatan dan analisis keamanan *website* menggunakan *AES* dan *token* anti *CSRF* terhadap serangan *Cross Site Request Forgery*.

2. Studi Pustaka: Melakukan pencarian informasi yang terkait dengan variabel yang sedang diamati melalui sumber-sumber seperti buku, jurnal, artikel, dan sumber lainnya.

3.4. Teknik Analisis Data

Teknik analisis data adalah metode yang digunakan untuk mengolah dan menganalisis data dalam konteks penelitian, termasuk penerapan alat-alat statistik yang sesuai (Indra Prasetia, 2022). Proses analisis data yang dilakukan melibatkan perhitungan persentase serangan yang berhasil diluncurkan terhadap situs web yang telah diamanankan dari serangan *CSRF*. Kemudian setelah beberapa serangan di implementasikan penulis akan melakukan pengamatan dan analisis seberapa efektif penambahan *token* anti *CSRF* terhadap serangan *CSRF*.

Berikut adalah tahapan dalam menganalisis dan menarik kesimpulan dengan metode deskriptif kuantitatif yaitu:

1. Koleksi data, pada tahapan ini bertujuan untuk mengumpulkan data yang berasal dari beberapa kali serangan *CSRF* dilakukan.
2. Transkripsi data, menulis ulang data yang terkumpul dalam bentuk terstruktur yang dapat dianalisis.
3. Analisis data, tahapan ini bertujuan menganalisis hasil data serangan yang dilakukan.
4. Verifikasi data, setelah melewati seluruh tahapan, data kembali diperiksa dan validasi semua data yang ada.
5. Menyimpulkan hasil analisis dan mempresentasikan secara jelas dan terperinci.

3.5. Literatur Review

Studi literatur melibatkan eksplorasi mendalam terhadap variabel penelitian, melakukan klasifikasi elemen-elemen yang relevan dan tidak relevan, melakukan sintesis informasi, memunculkan sudut pandang baru, serta mengidentifikasi hubungan dan korelasi antara variabel tersebut (Ridwan et al., 2021).

Dalam review literatur ini, Penulis akan memberikan ikhtisar mengenai penelitian-penelitian sebelumnya yang relevan dengan permasalahan dan tujuan penelitian tentang *CSRF*. Penulis akan memaparkan serta menganalisis temuan-temuan penting yang telah ditemukan oleh peneliti lain yang berkaitan dengan topik yang serupa dengan penelitian yang sedang dilakukan.

Tabel 3.1 Literatur Review

No	Nama Peneliti	Judul Penelitian	Hasil
1	M.Mahdi Maulana Lubis, Tommy, Divi Handoko, Nur Wulan	Analisis Implementasi *Laravel* 9 Pada *Website E-Book* Dalam Mengatasi N+1 *Problem* Serta Penyerangan *Csrf* dan *Xss*	Evaluasi terhadap serangan *CSRF* dan *XSS* pada implementasi *website e-book* dengan menggunakan *Laravel* 9 terbukti sangat efisien. *Laravel* 9 memberikan perlindungan yang solid terhadap serangan-serangan ini dengan mengatasi masalah *XSS* melalui penggunaan *blade template* yang secara otomatis mengkonversi semua *output* menjadi teks biasa melalui fungsi *htmlspecialchars*, sehingga semua skrip atau kode diubah menjadi teks biasa.
2	Asri Prameshwari, Nyoman Putra Sastra	Implementasi Algoritma Advanced Encryption Standard (AES) 128 Untuk Enkripsi dan Dekripsi File Dokumen	Algoritma *AES*-128 bisa digunakan sebagai salah satu pilihan untuk menjaga keamanan data, khususnya dalam proses enkripsi dan dekripsi dokumen. Keamanan hasil enkripsi ini dapat dipastikan selama kunci simetris (*symmetry key*) tidak

			jatuh ke tangan yang tidak berwenang.
3	Yendi Putra,Yuhandri Yunus, Sumijan	Meningkatkan Keamanan *Web* Menggunakan Algoritma *Advanced Encryption Standard* (*AES*) terhadap Seragan *Cross Site Scripting*	Dari hasil penelitian yang melibatkan implementasi Algoritma *AES* pada *token* untuk meningkatkan keamanan *website* dari serangan *XSS*, terbukti bahwa algoritma *AES* relatif mudah diimplementasikan, memiliki tingkat keamanan yang tinggi, dan memerlukan sedikit memori dalam pengoperasiannya, sehingga tidak memengaruhi proses dan ukuran file secara signifikan.
4	Ilham Firman Ashari, Vina Oktariana,Ringgo Galih Sadewo, Salman Damanhuri	Analysis of Cross Site Request Forgery (CSRF) Attacks on West Lampung Regency Websites Using OWASP ZAP Tools	Berdasarkan hasil pengamatan dan analisis penelitian yang telah dilaksanakan, dapat disimpulkan bahwa kerentanan keamanan pada *website* dapat menjadi target dari berbagai teknik serangan. Dari hasil pengamatan dan analisis tersebut, terungkap bahwa terdapat 53 *URL* halaman yang rentan terhadap serangan *CSRF*. Selain itu, berdasarkan pengamatan yang telah dilakukan, jenis risiko serangan yang ditemukan pada *website* ini tergolong dalam tingkat rendah.
5	Rusdiana , Banta Cut , Sanusi	Analisa Keamanan Website Terhadap Serangan Cross-Site Request Forgery (CSRF)	Melalui evaluasi keamanan dan analisis serangan terhadap *website* pemerintah Aceh Timur, telah dibuat sebuah *token* anti *CSRF* dengan tujuan untuk mengatasi beragam bentuk serangan yang menggunakan teknik *CSRF*. Selain itu, untuk mencegah serangan *DOM XSS*, langkah-langkah seperti modifikasi skrip dan filterisasi karakter dari input pengguna harus diterapkan.

3.6. Spesifikasi Minimum

Adapun spesifikasi yang dipakai untuk mengembangkan program ini adalah:

- Processor : Intel I5 – 12400
- VGA : Nvidia GT – 720
- Ram : 16gb
- Codeigniter v4.3.1
- PHP 7.4
- VS Code

3.7. Perancangan

Perancangan melibatkan proses menggambarkan, merencanakan, dan mengatur beberapa elemen yang berdiri sendiri menjadi satu kesatuan fungsional yang utuh (Fariyanto & Ulum, 2021). Hasil yang ingin dicapai pada tahap ini adalah berhasilnya pelaksanaan sistem yang telah direncanakan. Pada tahap ini, akan dipaparkan mengenai sistem yang telah direncanakan dan cara operasionalnya.

3.7.1. Halaman Input

Halaman input adalah area tempat pengguna dapat memasukkan data atau informasi ke dalam suatu sistem atau aplikasi. Tampilan *input* dibawah ini sangat sederhana hanya terdapat input *email* dan *password*. Karena tujuan input dibawah ini hanya untuk memastikan data masuk kedalam *database*.

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Anti CSRF</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD" crossorigin="anonymous">
</head>

<body>
    <div class="container mt-2">
        <h1>Anti CSRF</h1>


        <?php
        if (session()->getFlashdata('pesan')) : ?>
            <div class="alert alert-success" role="alert" mt-2>
                <?= session()->getFlashdata('pesan'); ?>
            </div>
        <?php endif; ?>

        <form action="/home/simpan" method="POST">
            <div class="mb-3">
                <label for="email" class="form-label">Email</label>
                <input type="email" class="form-control" id="email" aria-describedby="emailHelp" name="email">
            </div>
            <div class="mb-3">
                <label for="password" class="form-label">Password</label>
                <input type="password" class="form-control" id="password" name="password">
            </div>

            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXvN" crossorigin="anonymous"></script>
</body>

</html>
```

Gambar 3.1 *Source Code Website* Asli

Program ditersebut adalah *website* asli yang sudah terlindungi oleh serangan *croos site request forgery* (*CSRF*), dengan melihat *form action* yang mengarah ke *Controller home* dan *Function* simpan. Kemudian hasil program tersebut menjadi tampilan dibawah ini.

Gambar 3.2 Tampilan *Website* Asli

3.7.2. Halaman Serangan *CSRF*

Halaman dibawah ini adalah area tempat untuk menyerang *website* yang sudah terlindungi oleh serangan *CSRF*. Dengan input yang sama, seperti menginput

email dan *password* dan yang berbeda adalah *action* dari *input* ini mengarah ke *website* yang terlindungi dari serangan *CSRF*.

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>CSRF ATTACK!</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
GLhlTQ8iRABdZLl6OJoVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD" crossorigin="anonymous">
</head>

<body>
    <div class="container mt-2">
        <h1>CSRF ATTACK!</h1>
        <form action="../../csrf/public/home/simpan" method="POST">
            <div class="mb-3">
                <label for="email" class="form-label">Email</label>
                <input type="email" class="form-control" id="email" aria-describedby="emailHelp" name="email">
            </div>
            <div class="mb-3">
                <label for="password" class="form-label">Password</label>
                <input type="password" class="form-control" id="password" name="password">
            </div>

            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
w76AqPfDkMBDo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXvN" crossorigin="anonymous"></script>
</body>

</html>
```

Gambar 3.3 Source Code *Website* Tiruan atau Palsu

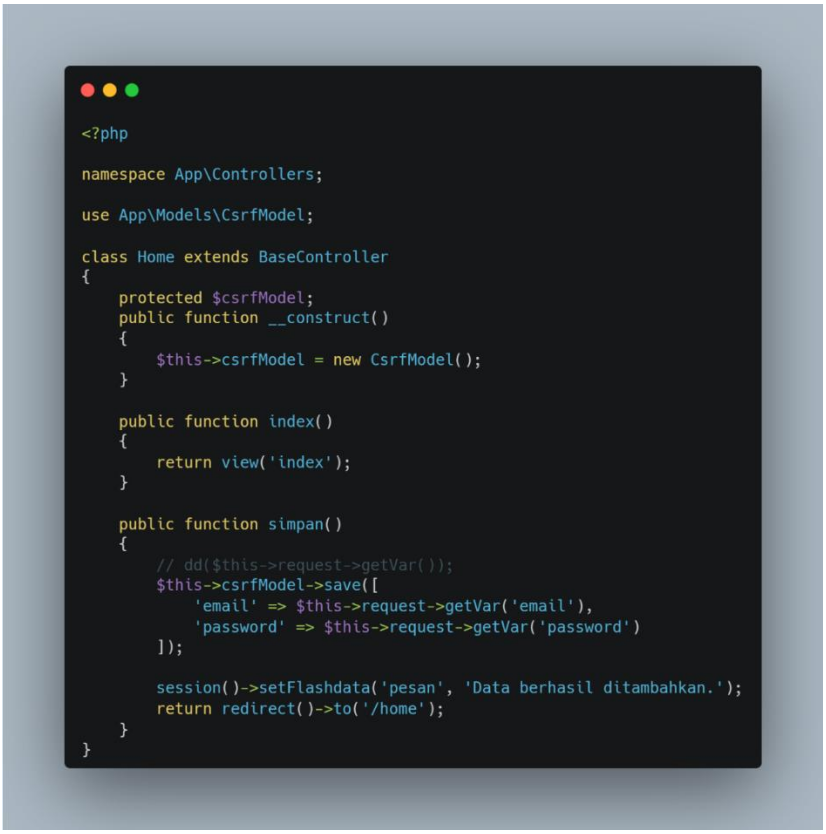
Untuk tampilan tidak ada perubahan dengan *website* yang asli, yang membedakannya hanya *form action* yang ditandai dengan warna merah. *Form action website* palsu ini mengarah ke folder *website* yang asli kemudian menirukan *controller* dan *functionnya*.

Gambar 3.4 Tampilan *Website* Tiruan atau Palsu

3.7.3. *Controller* dan *Function* Input Data

Controller di *Codeigniter* adalah komponen yang bertanggung jawab untuk mengontrol alur logika aplikasi berbasis *web*. *Controller* ini berfungsi sebagai perantara antara model dan *view* dalam pola desain *Model-View-Controller* (MVC) yang digunakan oleh *Codeigniter*.

Function adalah metode yang ada di dalam *controller* yang digunakan untuk mengelola logika aplikasi berdasarkan permintaan yang diterima dari pengguna. Setiap fungsi dalam *controller* dihubungkan dengan sebuah *URL* dan dipanggil ketika *URL* tersebut diakses. Berikut ini terdapat *controller* bernama *home* dan memiliki *function* simpan, dimana *function* ini berfungsi untuk menangkap data inputan dari *form*.



```php
<?php

namespace App\Controllers;

use App\Models\CsrfModel;

class Home extends BaseController
{
    protected $csrfModel;
    public function __construct()
    {
        $this->csrfModel = new CsrfModel();
    }

    public function index()
    {
        return view('index');
    }

    public function simpan()
    {
        // dd($this->request->getVar());
        $this->csrfModel->save([
            'email' => $this->request->getVar('email'),
            'password' => $this->request->getVar('password')
        ]);

        session()->setFlashdata('pesan', 'Data berhasil ditambahkan.');
        return redirect()->to('/home');
    }
}
```

Gambar 3.5 *Controller* dan *Function* Input *Data*

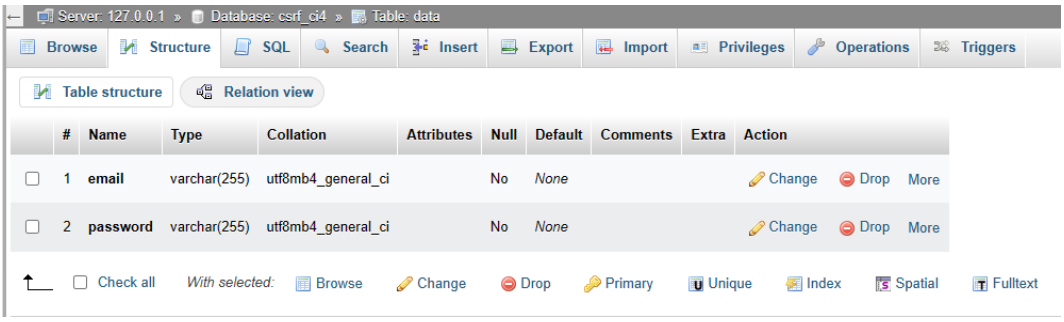
3.7.4. *Models*

Models di *Codeigniter* merupakan komponen yang bertanggung jawab untuk mengelola akses ke basis data atau sumber data lainnya. Berikut terdapat *models* yang bernama *CSRFModel* kemudian terdapat *protected table* yang berfungsi untuk menentukan tabel database dan juga ada *allowedfields* yang berguna untuk menentukan kolom-kolom mana yang diizinkan untuk dimasukkan data.

Gambar 3.6 *Models Database SQL*

3.7.5. Database

Database yang penulis gunakan sangatlah sederhana, hanya ada 1 tabel dengan nama data yang berisi 2 *field* yaitu *email* dan *password*. Karena penelitian ini membahas hak akses dari serangan *CSRF* tidak berfokus ke *database*.



Gambar 3.7 *Field Database*

BAB IV

HASIL DAN PEMBAHASAN

4.1. Implementasi Keamanan *Website*

4.1.1. Enkripsi Algoritma *AES* 128 *Bit*

AES adalah sebuah algoritma *block cipher*, yang berarti bahwa data masukan akan dipecah menjadi blok-blok terlebih dahulu, kemudian proses enkripsi akan dilaksanakan secara terpisah pada setiap blok data. Setiap blok *AES* terdiri dari matriks 4x4 *byte*, sehingga pesan akan dibagi menjadi blok-blok dengan ukuran 128 *bit* per blok. Proses enkripsi dalam algoritma AES dilakukan dalam beberapa tahapan. Tahap pertama, yang disebut Proses AddRoundKey, melibatkan operasi XOR antara kunci dan teks asli (byte). Tahap berikutnya adalah Proses SubBytes, yang melibatkan substitusi byte menggunakan S-Box. Kemudian, ada Proses ShiftRow, di mana byte dalam matriks state bergeser. Tahap selanjutnya adalah Proses MixColumns, yang mengubah kolom dalam matriks state. Setelah itu, terjadi lagi Proses AddRoundKey, di mana operasi XOR dilakukan antara state dan roundkey. Semua langkah di atas, mulai dari a sampai e, diulangi sebanyak 9 kali. Pada akhirnya, setelah iterasi selesai, langkah a sampai d dilakukan pada state tanpa langkah MixColumn.

Plaintext : 54 68 61 74 | 73 20 6D 79 | 20 4B 75 6E | 67 20 46 75

Key : 54 77 6F 20 | 4F 6E 65 20 | 4E 69 6E 65 | 20 54 77 6F

Tabel 4.1 RCON

Rcon (1) = 01000000	Rcon (2) = 02000000
Rcon (3) = 04000000	Rcon (4) = 08000000
Rcon(5) = 10000000	Rcon (6) = 20000000
Rcon (7) = 40000000	Rcon (8) = 80000000
Rcon (9) = 1B000000	Rcon (10) = 36000000

Tabel 4.2 Tabel S-Box

S-Box Values																
S(xy)		**y**														
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
x	**0**	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Langkah pertama yang harus dilakukan adalah menghitung *key schedule*. Hasil perhitungan ini akan menjadi bagian integral dari langkah selanjutnya dalam proses enkripsi.

Key schedule

Chiper Key : 54 68 61 74 | 73 20 6D 79 | 20 4B 75 6E | 67 20 46 75

54	73	20	67
68	20	4B	20
61	6D	75	46
74	79	6E	75

RotWord : 20 46 75 67

S-Box : B7 5A 9D 85

54	73	20	67
68	20	4B	20
61	6D	75	46
74	79	6E	75

54
68
61
74

⊕

B7
5A
9D
85

⊕

01
00
00
00

=

E2	91	B1	D6
32	12	59	79
FC	91	E4	A2
F1	88	E6	93

54 ⊕ B7 ⊕ 01 = 01010100 ⊕ 10110111 ⊕ 00000001 = 11100010 = **E2**

E2	91	B1	D6
32	12	59	79
FC	91	E4	A2
F1	88	E6	93

Round Key 1

56	C7	76	A0
08	1A	43	3A
20	B1	55	F7
07	8F	69	FA

Round Key 2

D2	15	63	C3
60	7A	69	03
0D	BC	E9	IE
E7	68	01	FB

Round Key 3

A1	B4	D7	14
12	68	51	52
02	BE	57	49
C9	A1	A0	5B

Round Key 4

B1	05	D2	C6
29	41	10	42
3B	85	D2	9B
33	92	32	69

Round Key 5

BD	B8	6A	AC
3D	7C	6C	2E
C2	47	95	0E
B7	15	27	4E

Round Key 6

CC	74	1E	B2
96	EA	86	A8
ED	AA	3F	31
16	03	24	6A

Round Key 7

8E	FA	E4	56
51	BB	ED	95
EF	45	7A	4B
21	22	06	6C

Round Key 8

BF	45	A1	F7
E2	59	64	F1
BF	FA	80	CB
90	B2	B4	D8

Round Key 9

28	6D	CC	3B
FD	A4	C0	31
DE	24	A4	6F
F8	4A	FE	26

Round Key 10

a. Initial Round

Tahap awal melibatkan langkah inisialisasi yang mencakup operasi XOR antara State dan Key.

54	73	20	67		54	4F	4E	20		00	3C	6E	47
68	20	4B	20		77	6E	69	54		1F	4E	22	74
61	6D	75	46	⊕	6F	65	6E	77	=	0E	08	1B	31
74	79	6E	75		20	20	65	6F		54	59	0B	1A
Initial State					*Chiper Key*					*After xor*			

b. Round 1

00	3C	6E	47
1F	4E	22	74
0E	08	1B	31
54	59	0B	1A

Setelah tahap inisialisasi selesai, langkah awal dalam setiap iterasi adalah *SubBytes*. *SubBytes* melibatkan penggantian nilai-nilai dalam *State* menggunakan S-Box seperti yang terlihat pada tabel 4.2, cara mencarinya sebagai berikut:

State Round 1 memiliki matriks { 00 1F 0E 54 | 3C 4E 08 59 | 6E 22 1B 0B | 47 74 31 1A }, jika diketahui *byte* pertama 00 y = 0 , x = 0, maka

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0	**63**	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76

Setelah mendapat perpotongan sumbu y dan x maka *SubBytes* yang ditemukan adalah **63**. Proses berlangsung seperti itu hingga seluruh matriks telah mengalami substitusi dengan tabel S-Box.

63	EB	9F	A0
C0	2F	93	92
AB	30	AF	C7
20	CB	2B	A2

After SubBytes

Tahap selanjutnya adalah *ShiftRows*. *ShiftRows* adalah tahap yang melibatkan pergeseran elemen dalam blok atau tabel, dengan pergeseran yang diterapkan pada setiap baris. Harap dicatat bahwa pada baris pertama, tidak ada pergeseran yang diperlukan. Pada baris ke 2 digeser sebanyak 1 *byte*s lalu setelah itu baris ke 3 digeser sebanyak 2 *byte*s dan yang keempat dilakukan pergeseran 3 *byte*s.

Cara pergeserannya adalah sebagai berikut:

63	EB	9F	A0		63	EB	9F	A0		63	EB	9F	A0
C0	2F	93	92	→	2F	93	92	C0	→	2F	93	92	C0
AB	30	AF	C7		**AB**	**30**	AF	C7		AF	C7	AB	30
20	CB	2B	A2		20	CB	2B	A2		**20**	**CB**	**2B**	A2

→	63	EB	9F	A0
	2F	93	92	C0
	AF	C7	AB	30
	A2	20	CB	2B
	After ShiftRows			

Langkah selanjutnya adalah tahap *MixColumns*, di mana dilakukan perkalian matriks antara hasil dari *ShiftRows* dan matriks Galois field yang telah ditetapkan sebelumnya

Matriks

02	03	01	01		63	EB	9F	A0		**BA**	84	E8	1B
01	02	03	01	x	2F	93	92	C0	=	75	A4	8D	40
01	01	02	03		AF	C7	AB	30		F4	8D	06	7D
03	01	01	02		A2	20	CB	2B		7A	32	0E	5D

Hasil **BA** merupakan hasil dari (02 x 63) ⊕ (03 x 2F) ⊕ (01 x AF) ⊕ (01 x A2)

- (02 x 63) = x+($x^6 + x^5 + x + 1$)

  = $x^7 + x^6 + x^2 + x$

  = 11000110

- (03 x 2F) = x+1(x^5 +x^3 + x^2 + x +1)

  = x^6 + x^4 + x^3 + x^2 + x + x^5 +x^3 + x^2 + x +1

  = x^6 + x^5 + x^4 + 1

  = 01110001

- (01 x AF) = 1(x^7 + x^5 + x^3 + x^2 + x + 1)

  = x^7 + x^5 + x^3 + x^2 + x + 1

  = 10101111

- (01 x A2) = 1(x^7 + x^5 + x)

  = x^7 + x^5 + x

  = 10100010

11000110

01110001

10101111

10100010
_____⊕
10111010 = **BA**

BA	84	E8	1B	⊕	E2	91	B1	D6	=	58	15	59	CD
75	A4	8D	40		32	12	59	79		47	B6	D4	39
F4	8D	06	7D		FC	91	E4	A2		08	1C	E2	DF
7A	32	0E	5D		F1	88	E6	93		8B	BA	E8	CE
After MixColumns					*AddRoundkey 1*					*After AddRoundkey*			

c. *Round* 2

6A	59	CB	BD
A0	4E	48	12
30	9C	98	9E
3D	F4	9B	8B

After SubBytes

6A	59	CB	BD
4E	48	12	A0
98	9E	30	9C
8B	3D	F4	9B

After ShiftRows

15	C9	7F	9D
CE	4D	4B	C2
89	71	BE	88
65	47	97	CD

After MixColumns

43	0E	09	3D
C6	57	08	F8
A9	C0	EB	7F
62	C8	FE	37

After AddRoundKey

d. *Round* 3

1A	AB	01	27
B4	5B	30	41
D3	BA	E9	D2
AA	E8	BB	9A

After SubBytes

1A	AB	01	27
5B	30	41	B4
E9	D2	D3	BA
9A	AA	E8	BB

After ShiftRows

AA	65	FA	88
16	0C	05	3A
3D	C1	DE	2A
B3	4B	5A	0A

After MixColumns

78	70	99	4B
76	76	3C	39
30	7D	37	34
54	23	5B	F1

After AddRoundKey

e. *Round* 4

BC	51	EE	B3
38	38	EB	12
04	FF	9A	18
20	26	39	A1

After SubBytes

BC	51	EE	B3
38	EB	12	38
9A	18	04	FF
A1	20	26	39

After ShiftRows

10	BC	D3	F3
D8	94	E0	E0
53	EA	9E	25
24	40	73	7B

After MixColumns

B1	08	04	E7
CA	FC	B1	B2
51	54	C9	6C
ED	E1	D3	20

After AddRoundKey

f. *Round* 5

C8	30	F2	94
74	B0	C8	37
D1	20	DD	50
55	F8	66	B7

After SubBytes

C8	30	F2	94
B0	C8	37	74
DD	50	D1	20
B7	55	F8	66

After ShiftRows

2A	26	8F	E9
78	1E	0C	7A
1B	A7	6F	0A
5B	62	00	3F

After MixColumns

9B	23	5D	2F
51	5F	IC	38
20	22	BD	91
68	F0	32	56

After AddRoundKey

g. *Round* 6

14	26	4C	15
D1	CF	9C	07
B7	93	7A	81
45	8C	23	B1

After SubBytes

14	26	4C	15
CF	9C	07	D1
7A	81	B7	93
B1	45	8C	23

After ShiftRows

A9	37	AA	F2
AE	D8	0C	21
E7	6C	B1	9C
F0	FD	67	3B
After MixColumns			

14	8F	C0	5E
93	A4	60	0F
25	2B	24	92
77	E8	40	75
After AddRoundKey			

h. *Round* 7

FA	73	BA	58
DC	49	D0	67
3F	F1	36	4F
F5	9B	09	9D
After SubBytes			

FA	73	BA	58
49	D0	67	DC
36	4F	3F	F1
9D	F5	9B	09
After ShiftRows			

9F	37	51	37
AF	EC	8C	FA
63	39	04	66
4B	FB	B1	D7
After MixColumns			

53	43	4F	85
39	06	0A	52
8E	93	3B	57
5D	F8	95	BD
After AddRoundKey			

i. *Round* 8

ED	1A	84	97
12	6F	67	00
19	DC	E2	5B
4C	41	2A	7A

After SubBytes

ED	1A	84	97
6F	67	00	12
E2	5B	19	DC
7A	4C	41	2A

After ShiftRows

E8	8A	4B	F5
74	75	EE	E6
D3	1F	75	58
55	8A	0C	38

After MixColumns

66	70	AF	A3
25	CE	D3	73
3C	5A	0F	13
74	A8	0A	54

After AddRoundKey

j. *Round* 9

33	51	79	0A
3F	8B	66	8F
EB	BE	76	7D
92	C2	67	20

After SubBytes

33	51	79	0A
8B	66	8F	3F
76	7D	EB	BE
20	92	C2	67

After ShiftRows

B6	E7	51	8C
84	88	98	CA
34	60	66	FB
E8	D7	70	51

After MixColumns

09	A2	F0	7B
66	D1	FC	3B
8B	9A	E6	30
78	65	C4	89

After AddRoundKey

k. *Final Round*

01	3A	8C	21
33	3E	B0	E2
3D	B8	8E	04
BC	4D	1C	A7

After SubBytes

01	3A	8C	21
3E	B0	E2	33
8E	04	3D	B8
A7	BC	4D	1C

After ShiftRows

29	57	40	1A
C3	14	22	02
50	20	99	D7
5F	F6	B3	3A

After AddRoundKey

Pada putaran terakhir dalam algoritma *AES* (*Advanced Encryption Standard*), tidak ada proses *MixColumns* yang dilakukan. Setelah melalukan proses panjang diatas maka sudah didapatkan *ChiperText* dari enkripsi algoritma *AES* yaitu: 29 C3 50 5F 57 14 20 F6 40 22 99 B3 1A 02 D7 3A

29	57	40	1A
C3	14	22	02
50	20	99	D7
5F	F6	B3	3A

4.1.2. Implementasi *ChiperText* Menjadi *Token*

Langkah pertama yang harus dilakukan adalah memasukkan *ChiperKey* yang telah terenkripsi ke dalam *Codeigniter*. *ChiperKey* ini akan di hash kembali oleh program agar *ChiperKey* dapat menjadi *token* anti *CSRF* yang akan di sembunyikan melalui *cookie browser*.

Gambar 4.1 Input *Chiperkey* ke *Codeigniter*

Selanjutnnya adalah *generate Chiperkey* menjadi *token* anti *CSRF*, tujuan dari *generate Chiperkey* ini adalah agar *Chiperkey* dapat menjadi *token* yang akan dikirimkan ke *cookie browser*, dan juga akan menambah keamanan yang berganda dari serangan *CSRF*.

```
@return string

protected function randomize(string $hash): string
    {
        $keyBinary  = random_bytes(static::CSRF_HASH_BYTES);
        $hashBinary = hex2bin($hash);

        if ($hashBinary === false) {
            throw new LogicException('$hash is invalid: ' . $hash);
        }

        return bin2hex(($hashBinary ^ $keyBinary) . $keyBinary);
    }

@return string

@throws InvalidArgumentException
protected function derandomize(string $token): string
    {
        $key   = substr($token, -static::CSRF_HASH_BYTES * 2);
        $value = substr($token, 0, static::CSRF_HASH_BYTES * 2);

        try {
            return bin2hex(hex2bin($value) ^ hex2bin($key));
        } catch (ErrorException $e) {
            throw new InvalidArgumentException($e->getMessage());
        }
    }
```

Gambar 4.2 *Generate Chiperkey* Menjadi *Token*

Kemudian membuat verifikasi *token*, program ini membuat validasi apakah *token* yang diberikan cocok dengan nilai *hash*. Jika *token* dan *hash* tidak di set atau nilai *hash* tidak cocok dengan *token* maka akan diarahkan ke *security exception* menggunakan metode *forDisallowedAction*().

```php
public function verify(RequestInterface $request)
    {
        // Protects POST, PUT, DELETE, PATCH
        $method           = strtoupper($request->getMethod());
        $methodsToProtect = ['POST', 'PUT', 'DELETE', 'PATCH'];
        if (!in_array($method, $methodsToProtect, true)) {
            return $this;
        }

        $postedToken = $this->getPostedToken($request);

        try {
            $token = ($postedToken !== null && $this->tokenRandomize)
                ? $this->derandomize($postedToken) : $postedToken;
        } catch (InvalidArgumentException $e) {
            $token = null;
        }

        // Do the tokens match?
        if (!isset($token, $this->hash) || !hash_equals($this->hash, $token)) {
            throw SecurityException::forDisallowedAction();
        }

        $this->removeTokenInRequest($request);

        if ($this->regenerate) {
            $this->generateHash();
        }

        log_message('info', 'CSRF token verified.');

        return $this;
    }
```

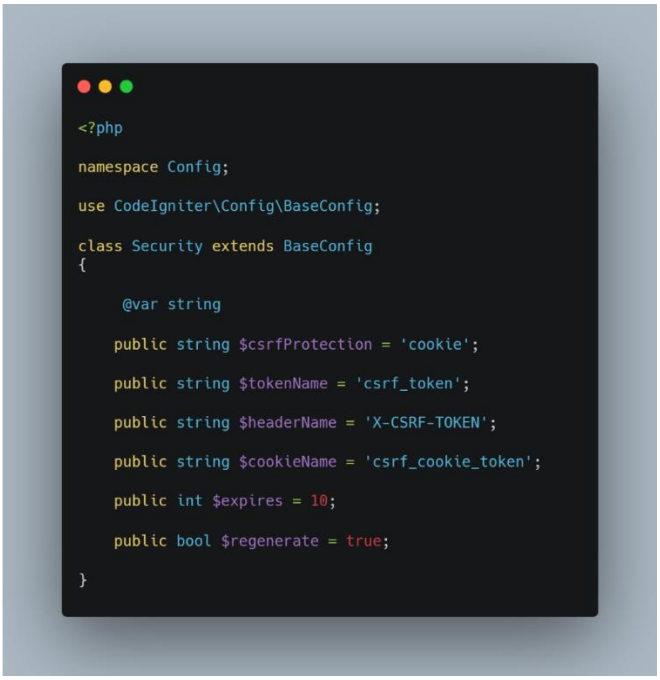
Gambar 4.3 Verifikasi *Token*

Setelah *generate Chiperkey* menjadi *token* dan membuat verifikasi *token* selanjutnya adalah membuat *function* untuk mendapatkan *token, header name* dan *cookie name.* Serta membuat perlindungan *CSRF* menggunakan *cookie* dan mengatur waktu berapa lama *token* tersebut akan *expires*.

```php
public function getTokenName(): string
    {
        return $this->tokenName;
    }

public function getHeaderName(): string
    {
        return $this->headerName;
    }

public function getCookieName(): string
    {
        return $this->cookieName;
    }

@deprecated
public function isExpired(): bool
    {
        return $this->cookie->isExpired();
    }
```

Gambar 4.4 Function Untuk Mendapatkan *Token*

Setelah mendeklarasi *function* maka langkah selanjutnya adalah memasukkan fungsi tersebut seperti berikut:



```php
<?php

namespace Config;

use CodeIgniter\Config\BaseConfig;

class Security extends BaseConfig
{
     @var string

    public string $csrfProtection = 'cookie';

    public string $tokenName = 'csrf_token';

    public string $headerName = 'X-CSRF-TOKEN';

    public string $cookieName = 'csrf_cookie_token';

    public int $expires = 10;

    public bool $regenerate = true;

}
```

Gambar 4.5 Input Function

Selanjutnya *token* sudah bisa digunakan untuk melindungi *website* dari serangan *CSRF*. Dengan langkah-langkah yang sudah diterapkan memungkin *website* sudah sangat aman dari serangan *CSRF*. Mulai dari menggunakan algoritma *AES* 128 *bit*, kemudian *ChiperText* di *hash* kembali guna memperkuat *token* agar tidak mudah di tebak.

4.1.3. Implementasi *Token* ke Dalam *Website*

Sebelumya penulis sudah melakukan enkripsi *Plain Text* menggunakan algoritma *AES*, kemudian *Chipertext* dari enkripsi di *generate* menjadi sebuah *token* yang akan digunakan untuk melindungi *website* dari serangan *CSRF*.

Selanjutnya adalah penulis akan mengimplementasikan *token* ke dalam *form input website*, untuk implementasinya adalah sebagai berikut.

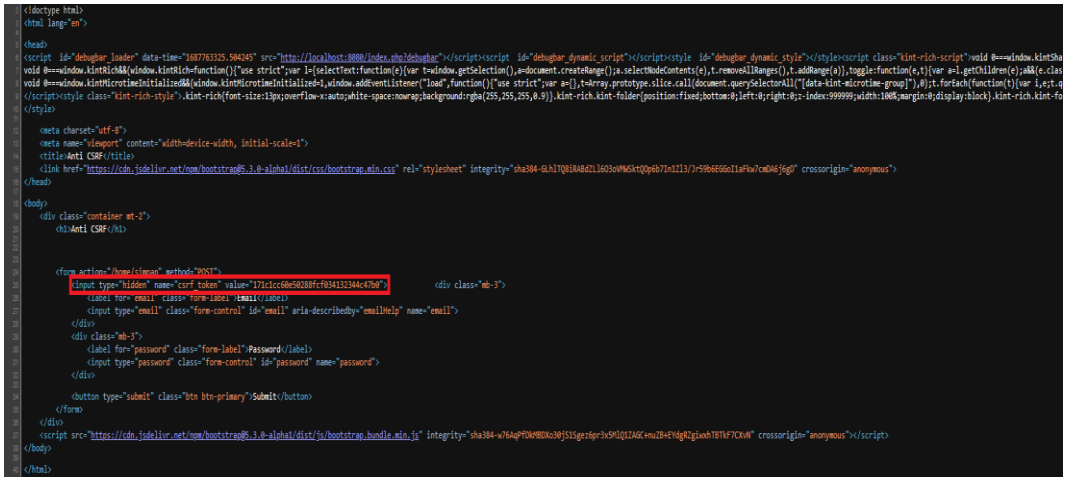
Cara yang harus dilakukan adalah dengan memanggil sebuah *method*, yang dimana *method* ini digunakan untuk menghasilkan input tersembunyi (*hidden input*) yang berisi *token CSRF*. Dimana *token* ini berasal dari *Chiperkey* yang sudah dienkripsi melalui algoritma *AES* yang sudah di*hash* oleh program agar bisa digunakan menjadi *token*.

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Anti CSRF</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD" crossorigin="anonymous">
</head>

<body>
    <div class="container mt-2">
        <h1>Anti CSRF</h1>

        <?php
        if (session()->getFlashdata('pesan')) : ?>
            <div class="alert alert-success" role="alert" mt-2>
                <?= session()->getFlashdata('pesan'); ?>
            </div>
        <?php endif; ?>

        <form action="/home/simpan" method="POST">
            <?= csrf_field(); ?>
            <div class="mb-3">
                <label for="email" class="form-label">Email</label>
                <input type="email" class="form-control" id="email" aria-describedby="emailHelp" name="email">
            </div>
            <div class="mb-3">
                <label for="password" class="form-label">Password</label>
                <input type="password" class="form-control" id="password" name="password">
            </div>

            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXvN" crossorigin="anonymous"></script>
</body>

</html>
```

Gambar 4.6 Implementasi *Token* ke Dalam *Website*

Pada gambar 4.13 terlihat sebuah *method* yang penulis *highlight* dengan warna merah. Fungsi method tersebut adalah untuk menghasilkan *token CSRF* yang sebelumnya sudah dibuat dengan algoritma *AES*.

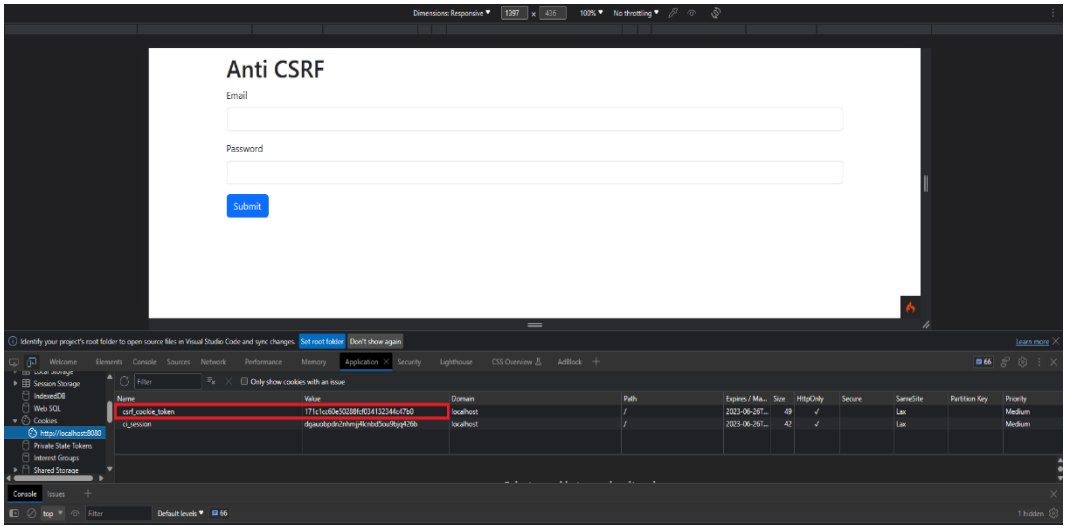
Method tersebut diletakkan dibawah *form action* yang menghasilkan elemen input tersembunyi dengan atribut '*type*' yang bernilai "*hidden*" kemudian '*name*' yang bernilai "*tokenName*" dan '*value*' yang bernilai "*token CSRF* yang valid".



Gambar 4.7 Value *Token CSRF* Random

Pada gambar diatas dapat terlihat bahwa *token CSRF* sudah terinput ke dalam *form website*. Pada *value* terdapat *token CSRF random* yang akan sulit diretas jika ada yang ingin menyerang *website* tersebut.

Ketika formulir tersebut dikirimkan melalui metode *POST*, *token CSRF* akan terkirim dengan permintaan tersebut. Pada sisi *server*, *Codeigniter* akan memeriksa kecocokan *token CSRF* yang dikirimkan dengan *token* yang tersimpan dalam *cookie browser*. Jika *token* tidak cocok, *Codeigniter* akan menganggap permintaan tersebut tidak valid kemudian menolaknya dengan menampilkan pesan kesalahan.



Gambar 4.8 *Token* di *Cookie Browser*

Pada gambar tersebut terlihat bahwa *token CSRF* yang ada di *cookie browser* sama dengan *token* yang ada di *form input* pada gambar. Yang artinya jika dijalankan input pada *website* tersebut maka input tersebut akan dipercaya dari *website* yang sah, karena menyertakan kedua *token CSRF* yang sama.

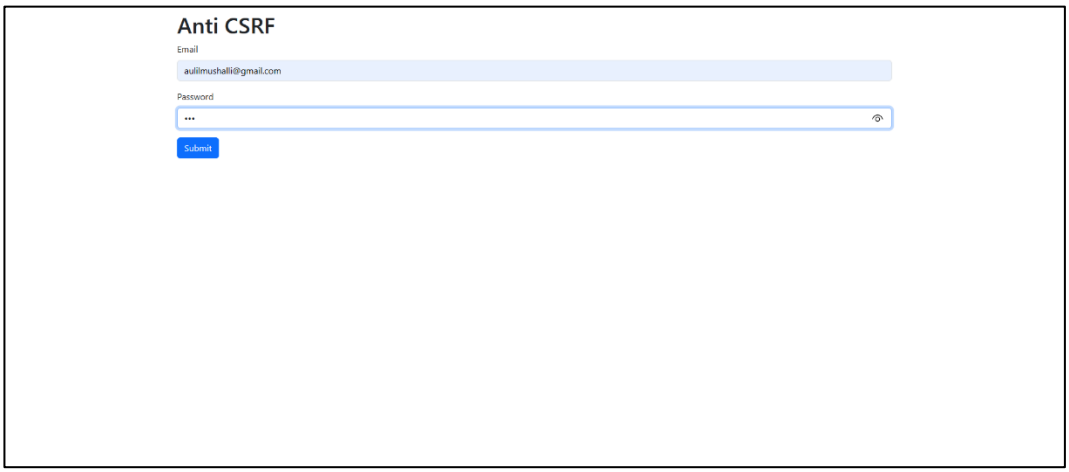
4.2. Pengujian Keamanan *Website*

Untuk pengujian keamanan *website* disini penulis akan menggunakan teknik *penetration testing*. Tujuan dari penggunaan *penetration testing* ini adalah untuk menguji kemanan *website* yang sudah terlindungi dari serangan *CSRF*.

Penulis akan melakukan menyerang *website* asli yang sudah memiliki *token* anti *CSRF* menggunakan *website* tiruan sebanyak sepuluh kali. Kemudian pada serangan terakhir penulis akan menghilangkan *token* anti *CSRF* pada *website* asli untuk mengetahui apakah dengan tidak adanya *token* anti *CSRF* pada *website* asli, *website* tiruan bisa menginput data kedalam *website* yang asli.

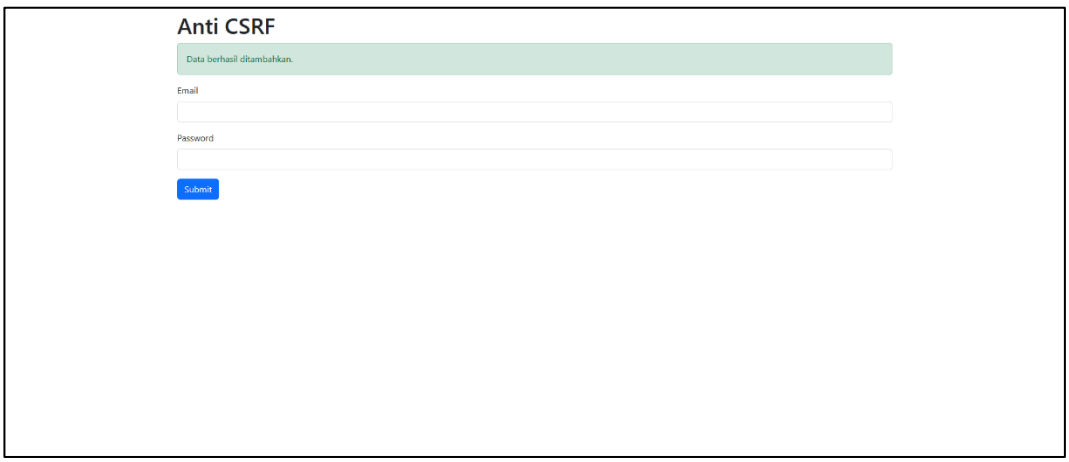
4.2.1. Input Data Melalui *Website* Asli

Pertama penulis melakukan input data menggunakan *website* asli dengan inputan *email* = aulilmushalli@gmail.com dan *password* = asd



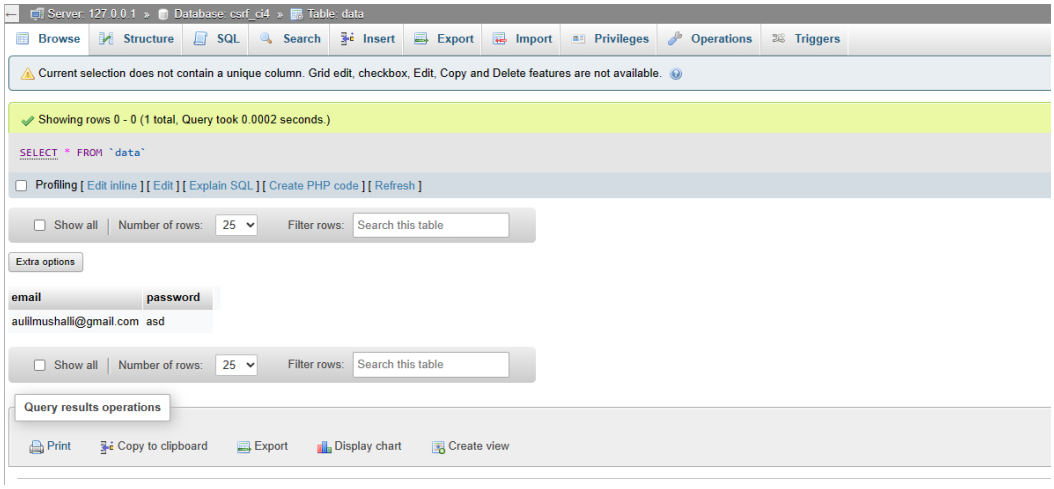
Gambar 4.9 Input Data Melalui *Website* Asli

Seteleh klik tombol *submit* maka data akan masuk kedalam *database*, dan akan ada *alert* bahwa data berhasil di input.



Gambar 4.10 Data berhasil di Input Pada *Website* Asli

Untuk memastikan data tersebut masuk atau tidak kedalam *database*, maka disini penulis menyertakan lampiran *field* yang terisi dari inputan tersebut.



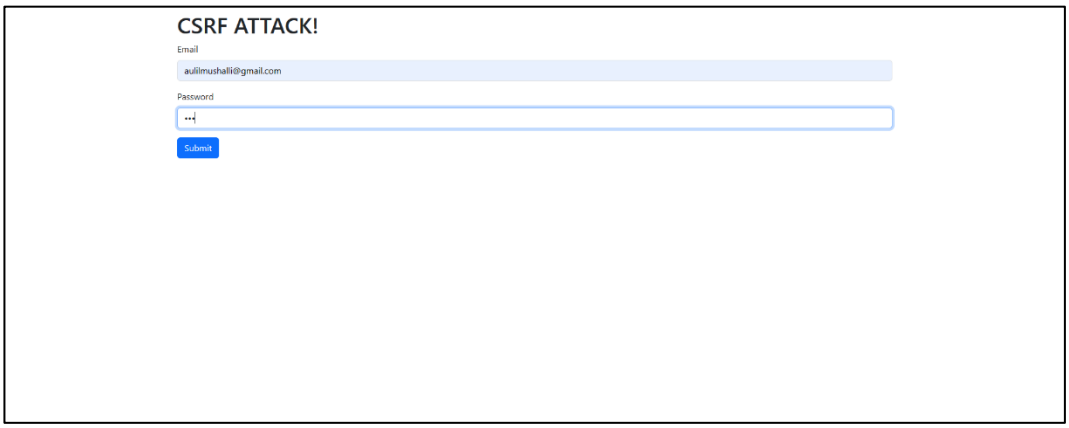
Gambar 4.11 Data Inputan Masuk ke *Database*

4.2.2. Input Data Melalui *Website* Tiruan Atau Palsu

Disini penulis akan melakukan input menggunakan *website* tiruan atau palsu yang dimana *form action* dari *website* ini mengarah ke *website* yang asli, bisa dilihat pada gambar 4.3 yang di *highlight* berwarna merah.

a. Penyerangan pertama

Percobaan pertama dilakukan dengan inputan yang sama dengan *website* asli yaitu dengan inputan email = aulilmushalli@gmail.com dan password = asd



Gambar 4.12 Input Data Melalui *Website* Tiruan Atau Palsu

Setelah klik tombol *submit* seketika ada tampilan *error* seperti gambar berikut

Gambar 4.13 Pesan Error *Token* Tidak Match

System akan menolak menerima *request* input data tersebut karena inputan tersebut tidak menyertakan *token* anti *CSRF*. Keterangan dari *error* diatas adalah jika *token* salah satu ada keduanya belum diatur maka kondisinya akan bernilai *true*. Maka *system* akan mendeteksi bawah ini adalah inputan dari *website* yang tidak sah.

b. Pengujian kedua

Percobaan kedua penulis melakukan lagi dengan inputan berbeda *email* = gtrskyline@gmail.com dan *password* = 12345

Gambar 4.14 Pengujian Penyerangan ke 2

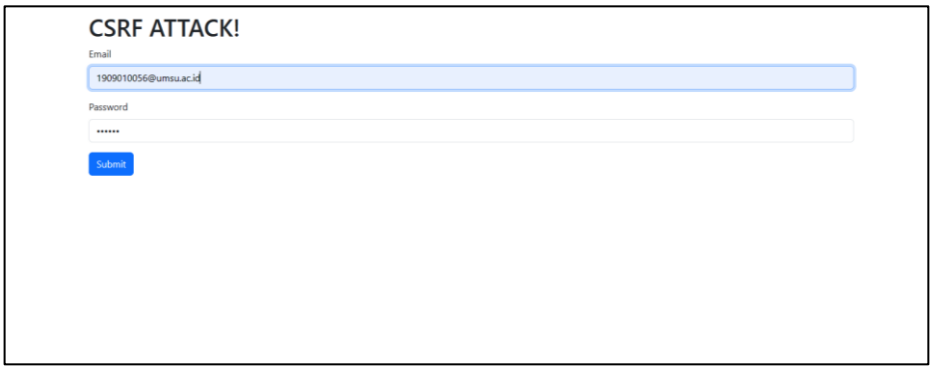
Setelah klik tombol *submit* maka tampilan *error* akan muncul kembali karena *request* ini tidak menyertakan *token* anti *CSRF*.

Gambar 4.15 Pesan Error *Token* Tidak Match ke 2

Kembali lagi *website* tiruan tidak bisa membuat *request* ke *website* yang asli, karena *website* tiruan tidak bisa mengirimkan *token* anti *CSRF* yang seolah-oleh *request* yang dibuat berasal dari *website* yang sah.

c. Pengujian ketiga

Percobaan ketiga penulis akan melakukan penyerangan kembali menggunakan *website* tiruan atau palsu dengan inputan email = 1909010056@umsu.ac.id dan password = qwerty. Tetapi pada percobaan kali ini penulis melakukan serangan kepada website yang sudah di*hosting* ke *server*.



Gambar 4.16 Pengujian Penyerangan ke 3

Setelah klik tombol *submit* maka tampilan akan kembali seperti semula karena *request* yang dikirim ditolak oleh *website* yang sudah di*hosting*, tentunya *website* yang di*hosting* sudah terlindungi dari serangan *CSRF*.

Gambar 4.17 *Website* yang di*hosting* menolak serangan

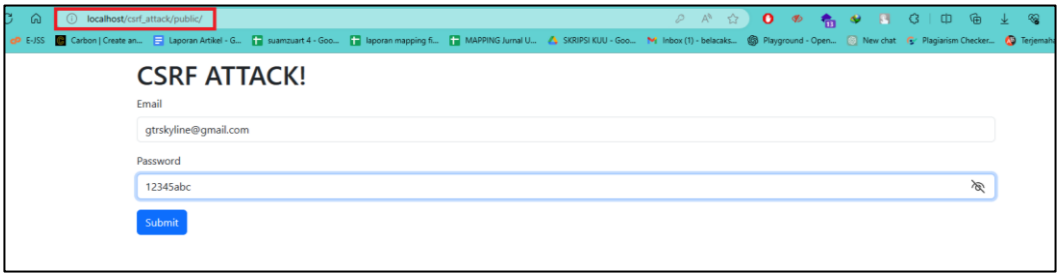
Balik lagi *website* tiruan tidak bisa membuat *request* ke *website* yang asli, karena *website* tiruan tidak bisa mengirimkan *token* anti *CSRF* yang seolah-oleh *request* yang dibuat berasal dari *website* yang sah. Ini sudah percobaan serangan yang ke tiga, terlihat betapa sulit untuk mencari celah keamanan dari penggunaan *token* anti *CSRF* tersebut.

Setelah penulis melakukan serangan ke *website* asli sebanyak sembilan kali, hasilnya tetap sama saja seperti percobaan pertama kedua dan ketiga. *Website* yang terlindungi dari serangan *CSRF* dengan menggunakan *token* sangat sulit untuk dicari celah keamanannya.

Untuk itu pada serangan terakhir penulis mencoba untuk tidak memakai *token* anti *CSRF* pada *website* yang asli, tujuan dari menghilangkan *token* ini adalah untuk melihat apakah dengan tidak adanya *token* pada *website* yang asli apakah *website* tiruan atau palsu bisa membuat *request* pada serangan terakhir ini.

Gambar 4.18 Action Dari *Website* Palsu Mengarah ke *Website* Asli

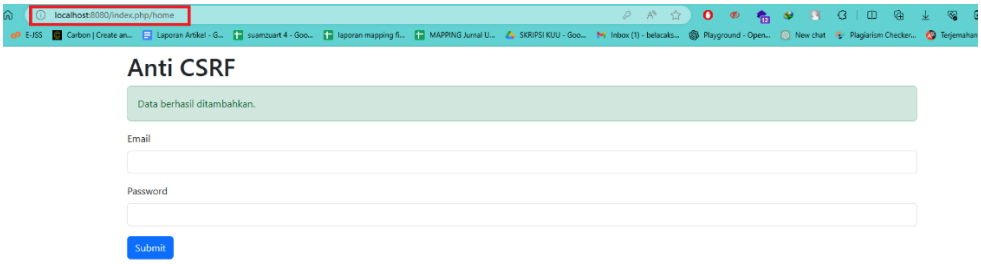
Highlight berwarna merah diatas menunjukkan bawah *action* dari *form website* palsu ini mengarah langsung ke folder *website* asli berserta *controller* dan juga *method*nya.



Gambar 4.19 Penyerangan ke *Website* Asli Tanpa *Token* Anti *CSRF*

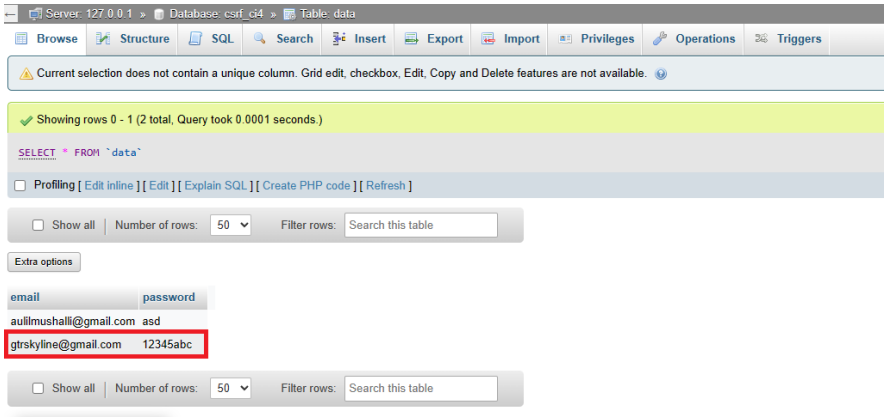
Pada gambar diatas warna yang dihighlight berwarna merah adalah menandakan kita berada pada halaman *website* tiruan atau palsu. Percobaan terakhir penulis akan melakukan penyerangan kembali dengan input *email* = gtrskyline@gmail.com dan *password* = 12345abc

Setelah klik tombol *submit* seketika inputan berhasil dilakukan, sedangkan inputannya menggunakan *website* tiruan atau palsu. *Action* tersebut mengarah langsung ke *website* yang asli, sehingga inputan tersebut masuk ke *database*.



Gambar 4.20 Request Dari *Website* Palsu Berhasil

Untuk memastikan inputan dari *website* palsu masuk atau tidak ke *database*, penulis menyertakan gambar *field* di *database*.



Gambar 4.21 Inputan Data Dari *Website* Palsu Berhasil Masuk

Seperti yang terlihat pada gambar 4.26 bahwa inputan dari *website* palsu sama dengan yang ada di dalam *database*. Ini menandakan bahwa inputan tersebut berhasil masuk, penyerang bisa melakukan request ke *website* yang asli tanpa harus menggunakan *website* tersebut.

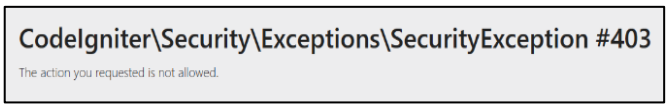
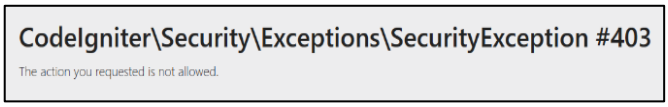
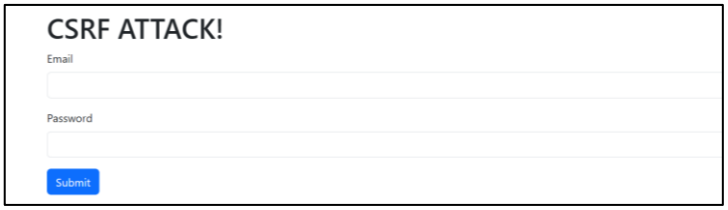
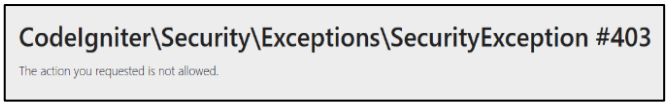
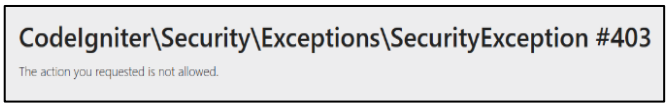
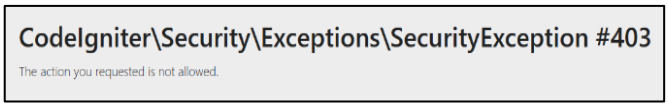
4.3. Hasil Pengujian (testing)

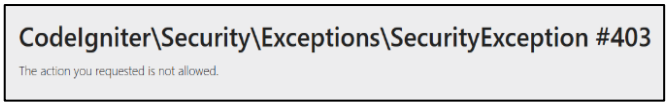
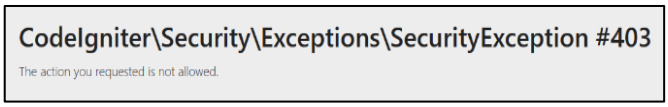
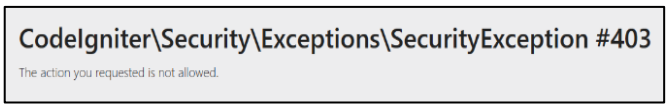
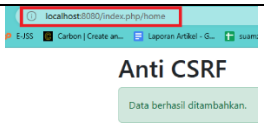
Penulis melakukan pengujian system dalam penggunaan metode blackbox, pengujian terfokus pada evaluasi fungsi sistem yang telah didefinisikan dan diimplementasikan, dengan tujuan menguji apakah program berjalan dengan baik atau tidak.

4.3.1. Tahap Pengujian

Pada tahap pengujian, penulis menggunakan metode *blackbox* untuk menguji seluruh serangan yang telah dilakukan terhadap situs *web* yang telah diamanankan dari serangan *CSRF*. Pengujian bertujuan untuk memeriksa kesesuaian antara hasil pengujian dan analisis serta rancangan yang telah disusun sebelumnya. Hasil pengujian dipresentasikan dalam tabel berikut:

Tabel 4.3 Pengujian Pentration Testing

No	Penetration Testing	Parameter	Hasil	Keterangan
1	Penyerangan ke - 1	*Token* anti *CSRF* diterapkan	*CSRF* ditolak	 *Website* asli menolak serangan
2	Penyerangan ke - 2	*Token* anti *CSRF* diterapkan	*CSRF* ditolak	 *Website* asli menolak serangan
3	Penyerangan ke - 3 (*website* yang di*hosting*)	*Token* anti *CSRF* diterapkan	*CSRF* ditolak	 *Website* asli menolak serangan
4	Penyerangan ke - 4	*Token* anti *CSRF* diterapkan	*CSRF* ditolak	 *Website* asli menolak serangan
5	Penyerangan ke - 5	*Token* anti *CSRF* diterapkan	*CSRF* ditolak	 *Website* asli menolak serangan
6	Penyerangan ke - 6	*Token* anti *CSRF* diterapkan	*CSRF* ditolak	 *Website* asli menolak serangan

7	Penyerangan ke - 7	*Token* anti *CSRF* diterapkan	*CSRF* ditolak	 *Website* asli menolak serangan
8	Penyerangan ke - 8	*Token* anti *CSRF* diterapkan	*CSRF* ditolak	 *Website* asli menolak serangan
9	Penyerangan ke – 9	*Token* anti *CSRF* diterapkan	*CSRF* ditolak	 *Website* asli menolak serangan
10	Penyerangan ke – 10	*Token* anti *CSRF* tidak diterapkan	*CSRF* berhasil	 *Website* tiruan bisa melakukan *request* (*token* anti *CSRF* dihilangkan dari *website* asli)

Hasil pengujian menggunakan metode *blackbox* diatas adalah *website* asli mampu menahan serangan *CSRF* dengan menggunakan *token* anti *CSRF*. Dengan pengujian sebanyak sepuluh kali yang dimana pada serangan terakhir penulis tidak menerapkan *token* anti *CSRF* pada *website* yang asli. Sehingga pada serangan terakhir *website* palsu bisa melakukan *request input* data pada *website* asli.

BAB V
PENUTUP

5.1. Kesimpulan

Berdasarkan hasil dan pembahasan penelitian implementasi keamanan *website* dari serangan *cross site request forgery* menggunakan algoritma *Rijndael* pada *framework Codeigniter* dapat disimpulkan sebagai berikut:

1. Penerapan algoritma *Rijndael* untuk mengenkripsi *token* anti *CSRF*, tingkat keamanan *website* meningkat secara signifikan dalam melindungi *website* dari serangan *CSRF*. *Rijndael* merupakan algoritma enkripsi yang kuat dan dapat membantu melindungi *website* dari serangan *CSRF* yang berbahaya.
2. Implementasi algoritma *Rijndael* memungkinkan pembuatan dan penggunaan *token* keamanan yang unik pada setiap permintaan yang dibuat oleh pengguna. Ini membantu memastikan bahwa setiap permintaan berasal dari sumber yang sah dan mencegah penyerangan *CSRF*.
3. Setelah melakukan pengujian *penetration testing* dengan metode pengujian *black box website* asli mampu menahan serangan *CSRF* dengan menggunakan *token* anti *CSRF*.
4. Tanpa adanya *token* untuk melindungi dari serangan *CSRF* tingkat keamanan *form* pada *website* menjadi sangat rendah. *CSRF* memanfaatkan akses yang sudah diautentikasi pengguna untuk mengeksekusi perintah tanpa sepengetahuan atau izin dari pengguna itu sendiri. Dengan menerapkan *token* anti *CSRF* tingkat keamanan *form website* meningkat menjadi 98% dari serangan *CSRF* setelah melakukan penerapan *token* anti *CSRF*.

5.2. Saran

Berdasarkan hasil penelitian yang telah diperoleh yang ada dalam penelitian ini, saran yang diberikan peneliti adalah sebagai berikut:

1. Penting untuk diingat bahwa *CSRF* bukan satu-satunya ancaman keamanan yang perlu diperhatikan dalam pengembangan *website*, tetapi merupakan salah satu yang signifikan. Sebagai pengguna maupun *developer* yang membangun sistem harus mempunyai kesadaran atas kejahatan *cyber*, contohnya seperti serangan *CSRF* ini. Serangan ini akan berakibat sangat fatal pada sistem *website* jika tidak memberikan keamanan lebih awal pada *website* dari serangan *CSRF*.
2. Penelitian ini bisa dijadikan bahan perbandingan untuk penelitian yang lebih lanjut dan dapat dikembangkan lagi. Dengan membuat *token* menggunakan algoritma kriptografi yang lain, dan juga dapat menerapkan sistem keamanan *website* yang lain dari serangan *CSRF* seperti *SameSite Cookies*, *double submit cookies*, *captcha* dan lainnya.
3. Dengan diketahuinya bahwa serangan *CSRF* ini dapat membahayakan sistem keamanan *website* maka dari itu sangat penting untuk menambahkan *token* anti *CSRF* ini di setiap *request form* dengan action *post* yang akan di*submit* agar *website* terlindungi dari serangan *CSRF*.

DAFTAR PUSTAKA

Al, M., Rizki, K., & Op, A. F. (2021). RANCANG BANGUN APLIKASI E-CUTI PEGAWAI BERBASIS WEBSITE (STUDI KASUS : PENGADILAN TATA USAHA NEGARA). *Jurnal Teknologi Dan Sistem Informasi (JTSI)*, *2*(3), 1–13. http://jim.teknokrat.ac.id/index.php/JTSI

Al-Khowarizmi. (2021). *Pengantar Teknologi Informasi (Dalam Perkembangan Data Science)*. UMSUPRESS.

Amazon, F., Handrianus Pranatawijaya, V., Hendrik Timang, J., Palangka Raya, K., & Tengah, K. (2021). Rancang Bangun Sistem Informasi Akademik Fakultas Matematika Dan Ilmu Pengetahuan Alam Berbasis Website. In *JOINTECOMS (Journal of Information Technology and Computer Science) p-ISSN: xxxx-xxxx* (Vol. 1, Issue 1).

Ashari, I. F., Oktarina, V., Sadewo, R. G., & Damanhuri, S. (2022). Analysis of Cross Site Request Forgery (CSRF) Attacks on West Lampung Regency Websites Using OWASP ZAP Tools. *Jurnal Sisfokom (Sistem Informasi Dan Komputer)*, *11*(2), 276–281. https://doi.org/10.32736/sisfokom.v11i2.1393

Benu L Fred, & Benu S Agus. (2019). *Metode Penelitian Kuantitatif*. Prenadamedia Group.

Budiman, A., Ahdan, S., & Aziz, M. (2021). ANALISIS CELAH KEAMANAN APLIKASI WEB E-LEARNING UNIVERSITAS ABC DENGAN VULNERABILITY ASSESMENT. In *Jurnal Komputasi* (Vol. 9, Issue 2).

Cristy, N., & Riandari, F. (2021). Niolinda Cristy 1 , Fristi Riandari 2 [Implementasi Metode Advanced Encryption Standard (AES 128 Bit) Untuk Mengamankan Data Keuangan. *JIKOMSI [Jurnal Ilmu Komputer Dan Sistem Informasi]*, *4*(2), 75.

Darwis, D., Ferico Octaviansyah, A., Sulistiani, H., & Putra, R. (2020). APLIKASI SISTEM INFORMASI GEOGRAFIS PENCARIAN PUSKESMAS DI KABUPATEN LAMPUNG TIMUR. *Jurnal Komputer Dan Informatika*, *15*, 159–170.

Fahrozi, I., Fadly, A., Pratama, H., Nuraeni, Y., & Pratama Juniar, R. (2023). PENGUJIAN BLACK BOX TESTING PADA APLIKASI ACTION &

STRATEGY BERBASIS ANDROID DENGAN TEKNOLOGI PHONEGAP. *OKTAL : Jurnal Ilmu Komputer Dan Science*, *2*(5). https://journal.mediapublikasi.id/index.php/oktal

Fariyanto, F., & Ulum, F. (2021). PERANCANGAN APLIKASI PEMILIHAN KEPALA DESA DENGAN METODE UX DESIGN THINKING (STUDI KASUS: KAMPUNG KURIPAN). *Jurnal Teknologi Dan Sistem Informasi (JTSI)*, *2*(2), 52–60. http://jim.teknokrat.ac.id/index.php/JTSI

Handrianus Pranatawijaya, V., Noor Kamala Sari, N., Bagus Adidyana Anugrah Putra, P., Teknik Informatika UPR, J. F., Timang Tunjung Nyaho Jurusan Teknik Informatika FT UPR, J. H., & Timang Tunjung Nyaho, J. H. (2019). SISTEM INFORMASI GEOGRAFIS MENCARI RUTE LOKASI TRAVEL DI KOTA PALANGKA RAYA BERBASIS WEBSITE. *Jurnal Teknologi Informasi*, *13*(1).

Hidayat, T. (2019). *ENCRYPTION SECURITY SHARING DATA CLOUD COMPUTING BY USING AES ALGORITHM: A SYSTEMATIC REVIEW*. *2*(2).

Indra Prasetia. (2022). *Metodologi Penelitian Pendekatan Teori dan Praktik* (Akrim & E. Sulasmi, Eds.). UMSU PRESS.

Irsyad, S., & Sitio, A. S. (2019). I N F O R M A T I K A PENERAPAN KONSEP MVC PADA SISTEM PENJUALAN ONLINE DENGAN SISTEM KEAMANAN MENGGUNAKAN ALGORITMA RIJNDEAL. *Jurnal Informatika, Manajemen Dan Komputer*, *11*(2).

Kour MTech Student, P. (2020). A Study on Cross-Site Request Forgery Attack and its Prevention Measures. *Int. J. Advanced Networking and Applications*, 4561–4566. http://127.0.0.1/hello.php

Lalia, S., & Moustafa, K. (2019). Implementation of web browser extension for mitigating CSRF attack. *Advances in Intelligent Systems and Computing*, *931*, 867–880. https://doi.org/10.1007/978-3-030-16184-2_82

Mahdi Maulana Lubis, M., Handoko, D., & Wulan, N. (2022). Analisis Implementasi Laravel 9 Pada Website E-Book Dalam Mengatasi N+1 Problem Serta Penyerangan Csrf dan Xss. *Jurnal Ilmu Komputer Dan Sistem Informasi*

(JIRSI), *2023*(2), 173–187. https://jurnal.unity-academy.sch.id/index.php/jirsi/index

Marsiani, E. S., Setiadi, I., Cahyo, A., Raya, J., No, T., Gedong, K., Rebo, P., & Timur, J. (n.d.). IMPLEMENTASI SISTEM KEAMANAN AES 256-BIT GCM GUNA MENGAMANKAN DATA PRIBADI. In *Jurnal Rekayasa Komputasi Terapan)* (Vol. 01).

Muliadi, M., Andriani, M., & Irawan, H. (2020). PERANCANGAN SISTEM INFORMASI PEMESANAN KAMAR HOTEL BERBASIS WEBSITE (WEB) MENGGUNAKAN DATA FLOW DIAGRAM (DFD). *JISI: Jurnal Integrasi Sistem Industri*, *7*(2), 111. https://doi.org/10.24853/jisi.7.2.111-122

Patil, A., Yadav, A., Krishnan, H., & Prasad, R. (2019). ANALYSIS OF CROSS SITE REQUEST FORGERY ATTACK ON WEBKIT. *International Journal of Research and Analytical Reviews*. www.ijrar.org

Prameshwari, A., & Sastra, N. P. (2018). Implementasi Algoritma Advanced Encryption Standard (AES) 128 Untuk Enkripsi dan Dekripsi File Dokumen. *Eksplora Informatika*, *8*(1), 52. https://doi.org/10.30864/eksplora.v8i1.139

Prayudha, J. (2019). Implementasi Keamanan Data Gaji Karyawan Pada PT. Capella Medan Menggunakan Metode Advanced Encryption Standard (AES). *SAINTIKOM*, *18*(SAINTIKOM), 119–129.

Priyanto Hidayatullah. (2021). *Pemrograman Web*. Informatika Bandung.

Putri Harum, D., & Arifianto, S. (2019). Improvisasi Algoritma Advanced Encryption Standard (AES) Dengan Melakukan Pemetaan S-Box Pada Modifikasi Mixcolumns. *Jurnal Repositor*, *1*(2), 95–104.

Rankothge, W. H., & Randeniya, S. M. N. (2020). Identification and Mitigation Tool for Cross-Site Request Forgery (CSRF). *IEEE Region 10 Humanitarian Technology Conference*, *R10-HTC*, *2020-December*. https://doi.org/10.1109/R10-HTC49770.2020.9357029

Ridwan, M., Ulum, B., Muhammad, F., Indragiri, I., & Sulthan Thaha Saifuddin Jambi, U. (2021). Pentingnya Penerapan Literature Review pada Penelitian Ilmiah (The Importance Of Application Of Literature Review In Scientific Research). *Jurnal Masohi*, *2*(1), 42–51. http://journal.fdi.or.id/index.php/jmas/article/view/356

Riza, F., Sridewi, N., Husein, A. M., & Harahap, M. K. (2018). Analisa Frekuensi Hasil Enkripsi Algoritma Blowfish Terhadap Keamanan Informasi. *JUTIKOMP (Jurnal Teknik Informatika Komputer*, *1*(1).

Rohi Abdulloh. (2018). *7 IN 1 PEMROGRAMAN WEB UNTUK PEMULA*. PT Elex Media Komputindo.

Ronaldo, M., & Pasha, D. (2021). *SISTEM INFORMASI PENGELOLAAN DATA SANTRI PONDOK PESANTREN AN-AHL BERBASIS WEBSITE* (Vol. 2, Issue 1).

Saputra Djong, H., & Siswanto, S. (2022). IMPLEMENTASI KRIPTOGRAFI DENGAN MENGGUNAKAN METODE RC4 DAN AES-256 UNTUK MENGAMANKAN FILE DOKUMEN PADA PT VARNION TECHNOLOGY SEMESTA. In *Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI) Jakarta-Indonesia*.

Semastin, E., Azam, S., Shanmugam, B., Kannoorpatti, K., Jonokman, M., Samy, G. N., & Perumal, S. (2018). Preventive measures for cross site request forgery attacks on Web-based Applications. *International Journal of Engineering and Technology(UAE)*, *7*(4), 130–134. https://doi.org/10.14419/ijet.v7i4.15.21434

Sianipar, K. D. R., Siahaan, S. W., Siregar, M., & Gunawan, I. (2019). PENGAMANAN FILE SUARA MENGGUNAKAN KRIPTOGRAFI ALGORITMA RIJNDAEL DENGAN PROSES ENKRIPSI DAN DEKRIPSI. *TECHSI - Jurnal Teknik Informatika*, *11*(3), 431. https://doi.org/10.29103/techsi.v11i3.1967

Sulistiani, H., & Hendra Saputra, V. (2020). Penerapan Codeigniter Dalam Pengembangan Sistem Pembelajaran Dalam Jaringan Di SMK 7 Bandar Lampung. *Jurnal CoreIT*, *6*(2).

Sumiati, M., Abdillah, R., & Cahyo, A. (2021). Pemodelan UML untuk Sistem Informasi Persewaan Alat Pesta. *JURNAL FASILKOM*, *11*(2).

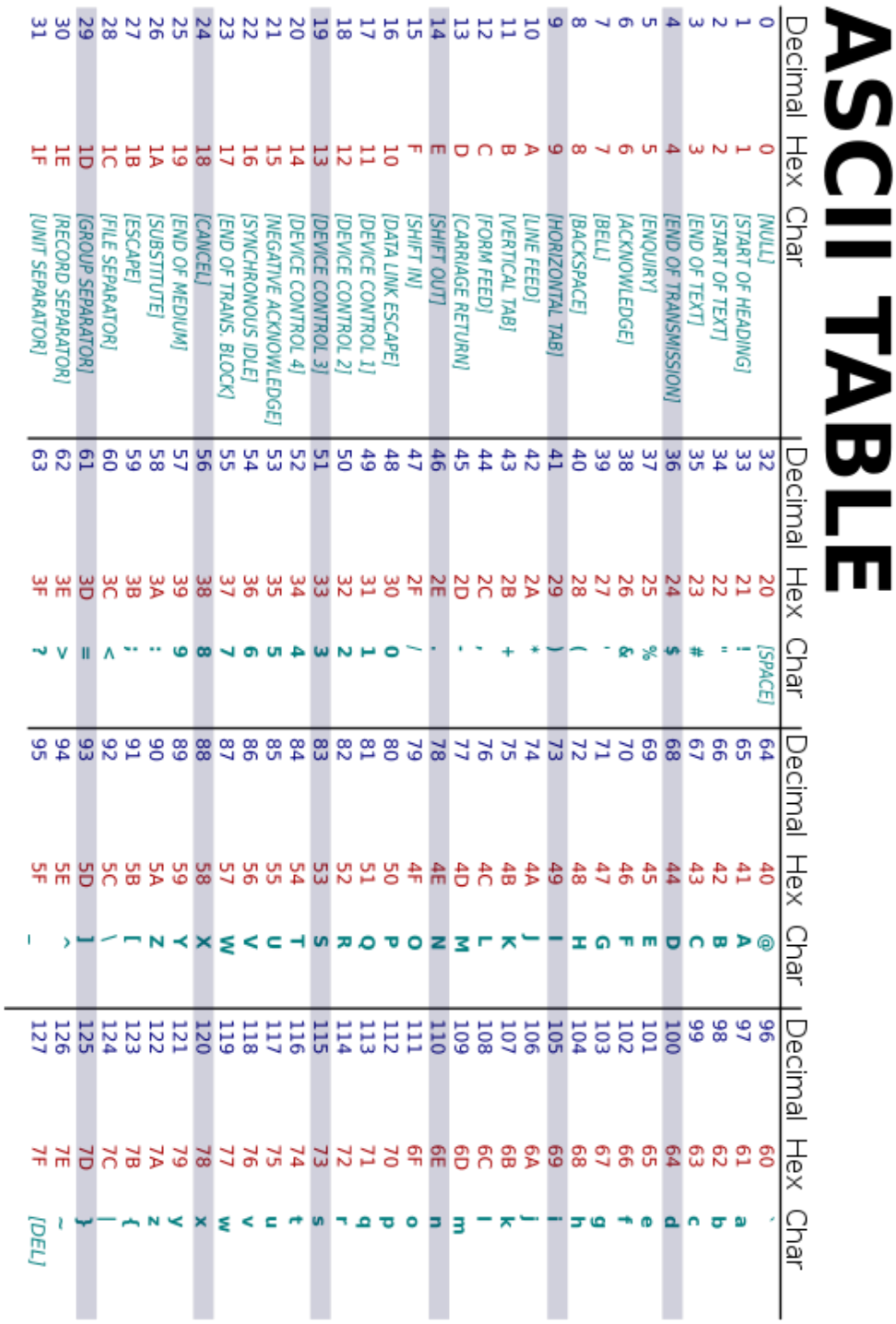
Syamsiah. (2019). PERANCANGAN FLOWCHART DAN PSEUDOCODE PEMBELAJARAN MENGENAL ANGKA DENGAN ANIMASI UNTUK ANAK PAUD RAMBUTAN. *STRING (Satuan Tulisan Riset Dan Inovasi Teknologi)* , *4*(1), 86–93.

Tahir, T. bin, Hadi Sirad, M. A., & Rais, M. (2020). Sistem Informasi Encrypt Dan Decrypt Dengan Algoritma AES Menggunakan Framework Laravel. *Patria Artha Technological Journal*, *4*(1). https://doi.org/10.33857/patj.v4i1.326

Unang Achlison. (2020). Analisis Implementasi Pengukuran Suhu Tubuh Manusia dalam Pandemi Covid-19 di Indonesia. *JURNAL ILMIAH KOMPUTER GRAFIS*, *13*(2), 102–106. http://journal.stekom.ac.id/index.php/pixel□page102

Wira, D., Putra, T., & Andriani, R. (2019). Unified Modelling Language (UML) dalam Perancangan Sistem Informasi Permohonan Pembayaran Restitusi SPPD. *Jurnal Teknoif Teknik Informatika*, *7*(1).

LAMPIRAN



ASCII TABLE

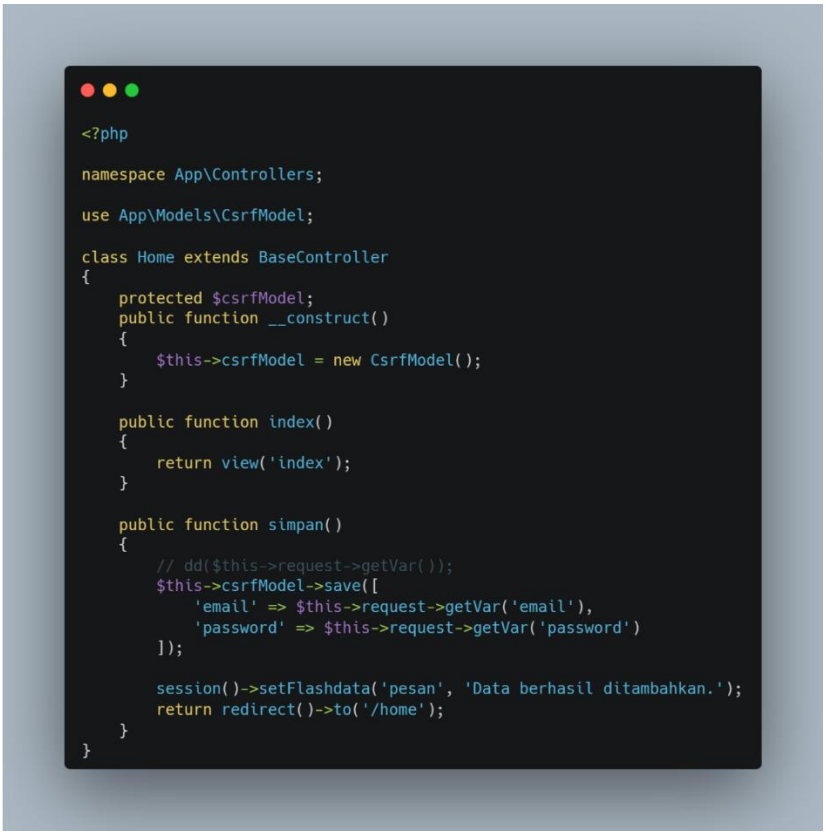
Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Anti CSRF</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD" crossorigin="anonymous">
</head>

<body>
    <div class="container mt-2">
        <h1>Anti CSRF</h1>


        <?php
        if (session()->getFlashdata('pesan')) : ?>
            <div class="alert alert-success" role="alert" mt-2>
                <?= session()->getFlashdata('pesan'); ?>
            </div>
        <?php endif; ?>

        <form action="/home/simpan" method="POST">
            <div class="mb-3">
                <label for="email" class="form-label">Email</label>
                <input type="email" class="form-control" id="email" aria-describedby="emailHelp" name="email">
            </div>
            <div class="mb-3">
                <label for="password" class="form-label">Password</label>
                <input type="password" class="form-control" id="password" name="password">
            </div>

            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXvN" crossorigin="anonymous"></script>
</body>

</html>
```

***Source Code Website* Asli**

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>CSRF ATTACK!</tittle>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD" crossorigin="anonymous">
</head>

<body>
    <div class="container mt-2">
        <h1>CSRF ATTACK!</h1>
        <form action="../../csrf/public/home/simpan" method="POST">
            <div class="mb-3">
                <label for="email" class="form-label">Email</label>
                <input type="email" class="form-control" id="email" aria-describedby="emailHelp" name="email">
            </div>
            <div class="mb-3">
                <label for="password" class="form-label">Password</label>
                <input type="password" class="form-control" id="password" name="password">
            </div>

            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
w76AqPfDkMBDo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXvN" crossorigin="anonymous"></script>
</body>

</html>
```

***Source Code Website* Tiruan atau Palsu**



```php
<?php

namespace App\Controllers;

use App\Models\CsrfModel;

class Home extends BaseController
{
    protected $csrfModel;
    public function __construct()
    {
        $this->csrfModel = new CsrfModel();
    }

    public function index()
    {
        return view('index');
    }

    public function simpan()
    {
        // dd($this->request->getVar());
        $this->csrfModel->save([
            'email' => $this->request->getVar('email'),
            'password' => $this->request->getVar('password')
        ]);

        session()->setFlashdata('pesan', 'Data berhasil ditambahkan.');
        return redirect()->to('/home');
    }
}
```

Controller* dan *Function Input Data

```php
<?php

namespace App\Models;

use CodeIgniter\Model;

class CsrfModel extends Model
{
    protected $table = 'data';
    protected $allowedFields = ['email', 'password'];
}
```

Models Database SQL

```php
<?php

namespace Config;

use CodeIgniter\Config\BaseConfig

class Encryption extends BaseConfig
    {
        public string $key = '29C3505F571420F6402299B31A02D73A';

        public string $driver = 'OpenSSL';

        public string $digest = 'SHA512';
    }
```

Input Chiperkey* ke *Codeigniter

```php
@return string

protected function randomize(string $hash): string
    {
        $keyBinary  = random_bytes(static::CSRF_HASH_BYTES);
        $hashBinary = hex2bin($hash);

        if ($hashBinary === false) {
            throw new LogicException('$hash is invalid: ' . $hash);
        }

        return bin2hex(($hashBinary ^ $keyBinary) . $keyBinary);
    }

@return string

@throws InvalidArgumentException
protected function derandomize(string $token): string
    {
        $key   = substr($token, -static::CSRF_HASH_BYTES * 2);
        $value = substr($token, 0, static::CSRF_HASH_BYTES * 2);

        try {
            return bin2hex(hex2bin($value) ^ hex2bin($key));
        } catch (ErrorException $e) {
            throw new InvalidArgumentException($e->getMessage());
        }
    }
```

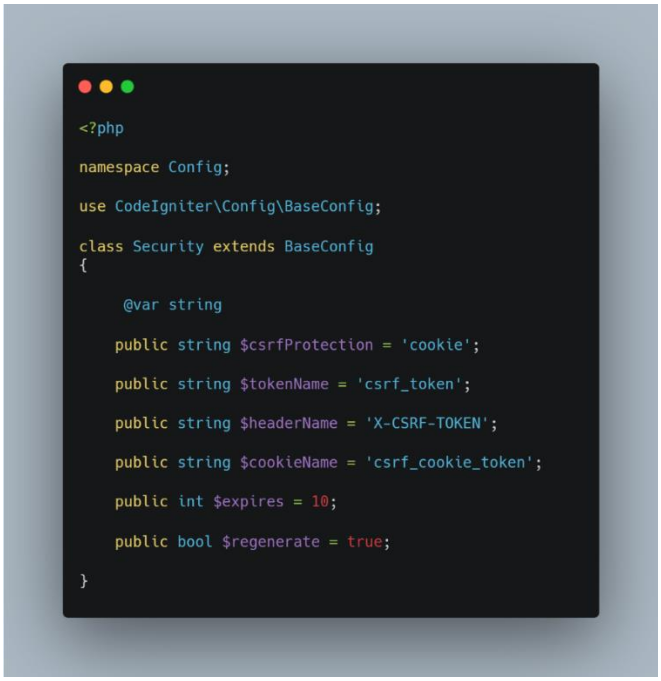
Generate Chiperkey* Menjadi *Token

```php
public function verify(RequestInterface $request)
    {
        // Protects POST, PUT, DELETE, PATCH
        $method           = strtoupper($request->getMethod());
        $methodsToProtect = ['POST', 'PUT', 'DELETE', 'PATCH'];
        if (!in_array($method, $methodsToProtect, true)) {
            return $this;
        }

        $postedToken = $this->getPostedToken($request);

        try {
            $token = ($postedToken !== null && $this->tokenRandomize)
                ? $this->derandomize($postedToken) : $postedToken;
        } catch (InvalidArgumentException $e) {
            $token = null;
        }

        // Do the tokens match?
        if (!isset($token, $this->hash) || !hash_equals($this->hash, $token)) {
            throw SecurityException::forDisallowedAction();
        }

        $this->removeTokenInRequest($request);

        if ($this->regenerate) {
            $this->generateHash();
        }

        log_message('info', 'CSRF token verified.');

        return $this;
    }
```

Verifikasi *Token*

```php
public function getTokenName(): string
    {
        return $this->tokenName;
    }

public function getHeaderName(): string
    {
        return $this->headerName;
    }

public function getCookieName(): string
    {
        return $this->cookieName;
    }

@deprecated
public function isExpired(): bool
    {
        return $this->cookie->isExpired();
    }
```

Function* Untuk Mendapatkan *Token



```php
<?php

namespace Config;

use CodeIgniter\Config\BaseConfig;

class Security extends BaseConfig
{

     @var string

    public string $csrfProtection = 'cookie';

    public string $tokenName = 'csrf_token';

    public string $headerName = 'X-CSRF-TOKEN';

    public string $cookieName = 'csrf_cookie_token';

    public int $expires = 10;

    public bool $regenerate = true;

}
```

Input Function* Untuk Mendapatkan *Token

MAJELIS PENDIDIKAN TINGGI PENELITIAN & PENGEMBANGAN PIMPINAN PUSAT MUHAMMADIYAH
UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UMSU Terakreditasi A Berdasarkan Keputusan Badan Akreditasi Nasional Perguruan Tinggi No. 89/SK/BAN-PT/Akred/PT/III/2019
Pusat Administrasi: Jalan Mukhtar Basri No. 3 Medan 20238 Telp. (061) 6622400 - 66224567 Fax. (061) 6625474 - 6631003
UMSU Unggul | Cerdas | Terpercaya
https://fikti.umsu.ac.id fikti@umsu.ac.id umsumedan umsumedan umsumedan umsumedan

Bila menjawab surat ini agar disebutkan nomor dan tanggalnya

Berita Acara Pembimbingan Proposal

Nama Mahasiswa : AULIL MUSHALLI
NPM : 1909010056
Nama Dosen Pembimbing : FERDY RIZA, ST., M.Kom
Program Studi : SISTEM INFORMASI
Konsentrasi : KRIPTOGRAFI
Judul Penelitian : IMPLEMENTASI KEAMANAN WEBSITE DARI SERANGAN CROSS SITE REQUEST FORGERY MENGGUNAKAN ALGORITMA RIJNDAEL PADA FRAMEWORK CODEIGNITER

Tanggal Bimbingan	Hasil Evaluasi	Paraf Dosen
03-01-2023 06-01-2023	-REVISI Judul dan Perbaikan Latar Belakang - Perbaikan isian BAB I dan tambahkan narasi -Tambahkan daftar Pustaka	
11-01-2023	- Perbaiki daftar pustaka -Perbaiki Latar belakang -Fokuskan algoritma pada latarbelakang	
1-02-2023	- Perbaikan Bab II - tata cara penulisan - isi -	
07-02-2023	- Perbaikan isi BAB II - Perbaikan situasi - Sumber Gambar	
13-02-2023	- lanjut Bab III	
27-02-2023	ACC SEMPRO	

Medan, 27-02-2023

Diketahui oleh :
Ketua Program Studi
Sistem Informasi

(MARTHIANO, S.Kom, M.Kom)

Disetujui oleh :
Dosen Pembimbing

(Ferdy Riza)

MAJELIS PENDIDIKAN TINGGI PENELITIAN & PENGEMBANGAN PIMPINAN PUSAT MUHAMMADIYAH
UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UMSU Terakreditasi A Berdasarkan Keputusan Badan Akreditasi Nasional Perguruan Tinggi No. 89/SK/BAN-PT/Akred/PT/III/2019
Pusat Administrasi: Jalan Mukhtar Basri No. 3 Medan 20238 Telp. (061) 6622400 - 66224567 Fax. (061) 6625474 - 6631003
https://fikti.umsu.ac.id fikti@umsu.ac.id umsumedan umsumedan umsumedan umsumedan

Berita Acara Pembimbingan Skripsi

Nama Mahasiswa : AULIL MUSHALLI
NPM : 1909010056
Nama Dosen Pembimbing : FERDY RIZA, S.T., M.KOM
Program Studi : SISTEM INFORMASI
Konsentrasi : KRIPTOGRAFI & SECURITY SYSTEM
Judul Penelitian : IMPLEMENTASI KEAMANAN WEBSITE DARI SERANGAN CROSS SITE REQUEST FORGERY MENGGUNAKAN ALGORITMA PADA FRAMEWORK CODEIGNITER

Tanggal Bimbingan	Hasil Evaluasi	Paraf Dosen
07-07-2023	- Perbaikan Hitungan AES	
18-07-2023	- Perbaikan pengujian BAB 4 dan Penulisan pada bab 4 - Lanjut Bab 5	
24-07-2023	- Perbaikan Bab 5	
27-07-2023	- Perbaikan pengujian dan bab 5 bab 4	
10-08-2023	ACC SIDANG	

Medan, 10-08-2023

Diketahui oleh :
Ketua Program Studi
Sistem Informasi

(.....)
MAFTIANO, S.Pd., SKom., M.Kom

Disetujui oleh :
Dosen Pembimbing

(.....)
FERDY RIZA, S.T., M.KOM