Strategi Pengamanan Kode Dalam Meningkatkan Kualitas Perangkat Lunak Berbasis Web

Dian Utami[1], Muhammad Asep Subandri[2]

[1,2] *Program Studi Rekayasa Perangkat Lunak - Politeknik Negeri Bengkalis*

Jl. Bathin Alam Sei Alam – Kec. Bengkalis, Kabupaten Bengkalis, Riau 2871

dianutami988@gmail.com, subandri@polbeng.ac.id

***Abstrak*—** Perangkat lunak berbasis web rawan mengalami berbagai serangan siber seperti XSS, SQL Injection, Path Traversal, CSRF, dan File *Upload Vulnerability* yang disebabkan oleh kelemahan pada implementasi kode program. Penelitian sebelumnya umumnya hanya mengidentifikasi kerentanan dan memberikan rekomendasi mitigasi menggunakan owasp zap, namun implementasi strategi pengamanan kode yang terintegrasi dan dievaluasi secara sistematis sebelum sistem dipublikasikan masih terbatas. sehingga, sebagian besar penelitian masih berfokus pada proses deteksi dan belum mengintegrasikan strategi pengamanan kode dengan kerangka pengujian keamanan yang sistematis. Penelitian ini bertujuan menerapkan strategi pengamanan kode perangkat lunak berbasis web dengan studi kasus pada sistem informasi akademik yaitu Akalink untuk mengurangi kerentanan keamanan sebelum sistem dipublikasikan. Dalam studi ini, pendekatan yang diterapkan bersifat kualitatif dengan pendekatan deskriptif, dibantu kerangka kerja *Software Testing Life Cycle* (STLC) untuk proses pengujian yang sistematis. Pengujian dilakukan dengan bantuan perangkat OWASP ZAP untuk mengidentifikasi kerentanan sebelum dan sesudah implementasi strategi pengamanan. Strategi yang diterapkan meliputi validasi input, implementasi token CSRF, validasi dan pengamanan unggah berkas, *controlled file storage, blade escaping*, enkripsi data sensitif, serta *rate limiting login*. Hasil penelitian menunjukkan bahwa penerapan strategi pengamanan mampu menurunkan jumlah temuan kerentanan dari 233 menjadi 60 temuan, sedangkan jenis kerentanan berkurang dari 14 menjadi 12 jenis. Hasil tersebut menunjukkan bahwa strategi pengamanan yang diterapkan terbukti efektif menurunkan jumalh kerentanan umum pada aplikasi web tanpa mengganggu fungsi utama sistem, sementara pendekatan STLC memberikan kerangka kerja yang sistematis dan menyeluruh untuk pengujian dan evaluasi keamanan perangkat lunak berbasis web.

***Kata Kunci*—** Keamanan Web, *Software Testing Life Cycle* (STLC), Strategi Pengamanan Kode, Perangkat Lunak

***Abstrak*—** Web-based software is vulnerable to various cyberattacks, such as Cross-Site Scripting (XSS), SQL Injection, Path Traversal, Cross-Site Request Forgery (CSRF), and File Upload Vulnerabilities, which are caused by weaknesses in code implementation. Previous studies have generally focused on identifying vulnerabilities and providing mitigation recommendations using OWASP ZAP; however, the implementation of integrated code security strategies that are systematically evaluated before software deployment remains limited. Consequently, most existing studies emphasize the detection process without integrating secure coding strategies with a systematic security testing framework. This study aims to implement secure coding strategies for web-based software using the Akalink academic information system as a case study to reduce security vulnerabilities before the system is deployed. A qualitative descriptive approach was employed, supported by the Software Testing Life Cycle (STLC) framework to ensure a systematic testing process. Security testing was conducted using OWASP ZAP to identify vulnerabilities before and after the implementation of the security strategies. The implemented strategies include input validation, CSRF token implementation, file upload validation and protection, controlled file storage, Blade escaping, sensitive data encryption, and login rate limiting. The results show that the implementation of these security strategies reduced the number of vulnerability findings from 233 to 60, while the types of vulnerabilities decreased from 14 to 12. These findings demonstrate that the proposed security strategies effectively reduce common web application vulnerabilities without affecting the system's core functionality, while the STLC approach provides a systematic and comprehensive framework for security testing and evaluation of web-based software.

***Keywords*—** Web Security, Software Testing Life Cycle (STLC), Code Security Strategies, Software

I. PENDAHULUAN

Seiring dengan berkembangnya teknologi informasi, pesatnya penggunaan aplikasi berbasis website sebagai sarana berbagi informasi, komunikasi, pengelolaan data berabagai bidang, terutama dalam lingkungan pendidikan.

Salah satu hal yang berkembang cukup pesat adalah aplikasi berbasis web. Aplikasi web memiliki keunggulan karena dapat dijalankan di berbagai platform dan mudah diakses oleh pengguna melalui jaringan internet menggunakan browser[1]. Website ialah sebuah halaman data berbasis web yang menyediakan berbagai informasi, dokumen ataupun tautan yang menghubungkan halaman data satu dengan halaman data lainnya yang dapat kita akses saat terkoneksi dengan internet kapanpun dan dimanapun melalui browser[2].

Meningkatnya penggunaan aplikasi web juga diikuti oleh meningkatnya ancaman terhadap keamanan sistem. Serangan seperti *SQL Injection*, *Cross-Site Scripting* (XSS), dan *Cross-Site Request Forgery* (CSRF) dan sebagainya dapat menyebabkan kebocoran data, manipulasi informasi, maupun akses tidak sah terhadap sistem[3]. Berbagai penelitian telah dilakukan untuk meningkatkan keamanan aplikasi web. Analisis kerentanan sistem informasi menggunakan standar OWASP untuk mengidentifikasi kelemahan keamanan aplikasi web[2]. Kajian lain membahas strategi pengamanan *front-end* dalam pengembangan website untuk meningkatkan perlindungan antarmuka pengguna [4]. Penelitian lain membahas mengimplementasikan fitur keamanan pada framework Laravel [5].

Berdasarkan hasil penelitian terdahulu, implementasi keamanan aplikasi web masih banyak difokuskan pada pendekatan tertentu, seperti analisis kerentanan menggunakan owasp zap, penerapan secure coding, serta perlindungan

terhadap jenis serangan tertentu, seperti *SQL Injection, Cross-Site Scripting (XSS), dan Cross-Site Request Forgery (CSRF*).

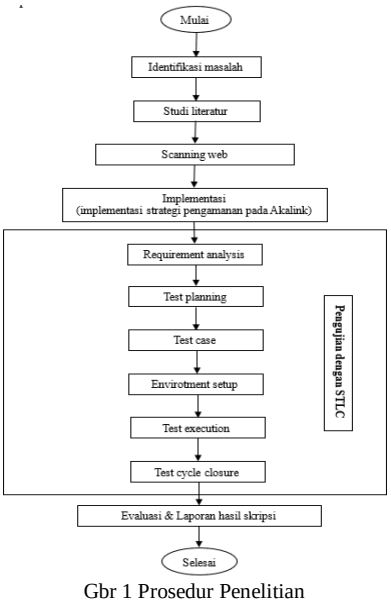
Sebagian besar penelitian tersebut masih berfokus pada mekanisme atau aspek keamanan tertentu sehingga belum mengintegrasikan berbagai strategi pengamanan kode dan proses pengujian keamanan dalam satu implementasi yang komprehensif. Selain itu, penerapan *Software Testing Life Cycle* (STLC) sebagai metode pengujian yang sistematis masih belum banyak dipadukan dengan implementasi mekanisme pengamanan dan pengujian kerentanan aplikasi web. Oleh karena itu, diperlukan penelitian yang mengintegrasikan berbagai mekanisme pengamanan kode dan proses pengujian keamanan untuk meningkatkan keamanan aplikasi web secara lebih menyeluruh.

II. METODE PENELITIAN

Pada studi ini, Penulis menerapkan studi kualitatif. Penelitian kualitatif adalah teknik yang yang memberikan kesempatan bagi peneliti untuk menguraikan secara mendalam fenomena yang diteliti sesuai dengan kondisi nyata pada objek penelitian. Proses pengujian akan dilakukan menggunakan siklus *Software Testing Life Cycle* (STLC) yang dijadikan acuan[6].

A. Prosedur Penelitian

Penelitian ini menggunakan metode deskriptif kualitatif, yaitu metode yang bertujuan untuk menguraikan secara mendalam fenomena yang sedang diteliti berdasarkan dengan realitas di lapangan [4].Alur kegiatan penelitian ini dapat dilihat pada flowchart berikut:

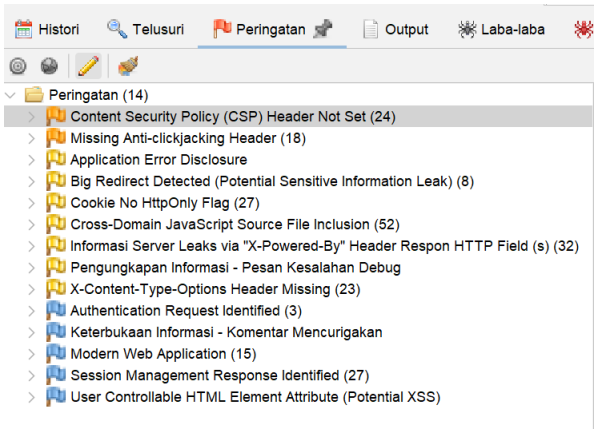


Gbr 1 Prosedur Penelitian

1) Identifikasi Masalah
   Pada tahap ini penulis mengidentifikasi masalah untuk menemukan permasalahan yang didapat pada pengamanan web. Hasil observasi terhadap website akalink menunjukkan masih terdapat adanya kelemahan, seperti halaman guru dapat diakses siapa saja, upload file berpotensi berbahaya, tidak ada pembeda peran (role), rentan CSRF.
2) Studi Literatur
   Dengan melakukan studi literatur, penulis memperoleh pemahaman mengenai metode pengujian perangkat lunak yang mengacu pada *Software Testing Life Cycle* (STLC) serta strategi pengamanan yang pernah diterapkan oleh penelitian sebelumnya. STLC digunakan sebagai acuan dalam merancang dan melaksanakan pengujian sistem secara sistematis melalui beberapa tahapan, yaitu *Test Planning*, *Test Case Development*, *Environment Setup*, *Test Execution*, dan *Test Cycle Closure*[7]. Pada tahap Test Planning, input yang digunakan meliputi dokumen kebutuhan sistem, ruang lingkup aplikasi, serta tujuan pengujian. Dari tahap ini dihasilkan output berupa dokumen rencana pengujian yang berisi strategi pengujian, dan cakupan fitur. Pada tahap Test Case Development, input berasal dari dokumen test plan dan requirement sistem. Tahap ini menghasilkan output berupa skenario uji, data uji, serta kriteria keberhasilan yang akan digunakan untuk memvalidasi fungsi sistem. Pada tahap Environment Setup, input berupa kebutuhan teknis sistem seperti spesifikasi perangkat keras, perangkat lunak, dan konfigurasi aplikasi. Output dari tahap ini adalah lingkungan pengujian yang siap digunakan untuk proses pengujian sesuai dengan kondisi yang menyerupai sistem sebenarnya. Pada tahap Test Execution, input berupa test case dan test environment yang telah disiapkan sebelumnya. Tahap ini menghasilkan output berupa hasil eksekusi pengujian, termasuk catatan error, atau kesesuaian fungsi sistem terhadap kebutuhan. Pada tahap Test Cycle Closure, input berasal dari hasil pengujian dan laporan bug yang telah diperoleh pada tahap sebelumnya. Output dari tahap ini berupa laporan akhir pengujian yang berisi evaluasi hasil pengujian, tingkat keberhasilan sistem.
3) Scanning Web
   Pada tahapan ini, peneliti melakukan analisis dengan scanning kerentanan pada web akalink menggunakan *tools owasp zap*. Dalam konteks ini, OWASP ZAP berperan sebagai alat bantu pengumpulan data, bukan sebagai metode penelitian utama. OWASP ZAP digunakan untuk mendeskripsikan hasil temuan kerentanan pada sistem web tanpa melakukan manipulasi variabel, melainkan melalui proses observasi, identifikasi terhadap hasil pemindaian keamanan sehingga menghasilkan data berupa daftar kerentanan yang kemudian dianalisis secara deskriptif kualitatif berdasarkan kategori risiko, dan tingkat keparahan[8]. Adapun langkah-langkah penggunaan OWASP ZAP dalam penelitian ini adalah sebagai berikut:
   1. Konfigurasi *Proxy Browser*: Peneliti mengatur browser agar terhubung dengan OWASP ZAP melalui proxy lokal. Output tahap ini: koneksi antara

browser dan OWASP ZAP untuk memantau seluruh traffic aplikasi web.

2. *Spider/Crawling* Aplikasi: OWASP ZAP melakukan proses spidering untuk menjelajahi seluruh halaman dan endpoint pada web Akalink. Output: daftar URL, endpoint, dan struktur navigasi aplikasi yang akan dianalisis lebih lanjut.
3. *Passive Scanning*: Pada tahap ini, OWASP ZAP secara otomatis menganalisis request dan response tanpa melakukan serangan aktif. Output: indikasi awal kerentanan seperti header security yang lemah, cookie tidak aman, atau informasi sensitif yang terekspos.
4. *Active Scanning:* OWASP ZAP melakukan pengujian aktif dengan mengirimkan payload ke aplikasi untuk mengidentifikasi kerentanan seperti SQL Injection atau XSS. Output: temuan kerentanan beserta tingkat risiko (*Low, Medium, High, Critical).*



Gbr 2 Scanning web owasp zap sebelum implementasi

4) Implementasi

Pada tahap implementasi, peneliti melakukan proses pengkodean dengan menerapkan strategi pengamanan perangkat lunak berbasis web untuk meminimalkan potensi kerentanan keamanan aplikasi. Mekanisme pengamanan yang diterapkan meliputi validasi input, perlindungan terhadap serangan *SQL Injection, Cross-Site Scripting (XSS), dan Cross-Site Request Forgery (CSRF)*, dan sebagainya[5]. Hasil implementasi strategi pengamanan tersebut dievaluasi melalui pengujian keamanan menggunakan *OWASP ZAP*. Tingkat keberhasilan penerapan mekanisme keamanan diukur berdasarkan jumlah dan tingkat risiko kerentanan yang terdeteksi, yang diklasifikasikan ke dalam kategori *High, Medium, Low, dan Informational*[9].

5) Pengujian STLC

Pendekatan ini dipilih karena memberikan struktur yang sistematis dalam memastikan kualitas perangkat lunak sebelum perangkat lunak itu dirilis[10]. tahapan yang akan dilakukan adalah Reqruitmen Analysis yaitu menganalisis pada komponen-komponen yang berpotensi menjadi titik masuk serangan. Test planning menyusun rencana uji yang mencakup strategi pengujian keamanan, metode yang diterapkan, serta alat bantu (tools) untuk mendukung proses identifikasi kerentanan. Test Case Development menyusun skenario pengujian yang relevan dengan ancaman keamanan web. Envirotment Setup penyiapan lingkungan pengujian yang sesuai dengan kondisi implementasi sistem. Lingkungan ini mencakup konfigurasi server, basis data, serta framework yang digunakan dalam pengembangan aplikasi. Test Execution adalah tahap pelaksanaan pengujian sesuai dengan kasus uji yang telah dirancang. Test Cycle Closure melakukan evaluasi menyeluruh terhadap hasil pengujian yang telah dilakukan.

6) Penyusunan Laporan

Menyusun hasil penelitian, dokumentasi dan seluruh proses yang telah dijalankan, termasuk analisis, implementasi, hasil pengujian ke dalam laporan.

III. HASIL DAN PEMBAHASAN

A. *Strategi Pengamanan Kode*

1) CSRF Token

Csrf token merupakan langkah pengamanan pada aplikasi web yang bertujuan untuk mencegah situs berbahaya memaksa pengguna yang sudah terverifikasi untuk melakukan tindakan tanpa disadari di situs terpercaya, proteksi ini umumnya mengunakan token unik yang tidak dapat diprediksi, memastikan setiap permintaan yang dikirim ke aplikasi web berasal dari sumber sah oleh pengguna[5]. Terdapat Middleware berfungsi sebagai lapisan perantara yang memfilter request masuk ataupun keluar sebelum mencapai aplikasi[11]. @csrf digunakan untuk membuat token csrf di setiap semua form blade yang melakukan request POST, PUT, PATCH, atau DELETE.

Mengaktifkan *VerifyCsrfToken middleware* pada level global/web dan menyematkan direktif @csrf pada setiap formulir blade yang menggunakan metode HTTP POST, PUT, PATCH, atau DELETE. Berdasarkan hasil pengujian pada skenario TCSRF_001 hingga TCSRF_007, strategi ini sangat efektif. Saat token dihapus via inspect element atau dimanipulasi melalui tools luar seperti Postman/Burp Suite, Laravel langsung memblokir permintaan tersebut dan mengembalikan *Error 419 Page Expired*. Dampaknya, risiko serangan pembajakan sesi formulir berhasil dieliminasi sepenuhnya. Tanpa token CSRF, sistem Akalink rentan terhadap eksploitasi di mana penyerang dapat menjebak pengguna yang terautentikasi (misalnya Guru atau Admin) untuk mengeksekusi aksi destruktif tanpa disadari (seperti mengubah password atau memanipulasi data akademik) melalui situs pihak ketiga. Dengan mengaktifkan middleware ini, setiap permintaan wajib menyertakan token kriptografis rahasia yang unik untuk setiap sesi.

TABEL SOURCE CODE 1 AKTIFKAN CSRF MIDDLEWARE

```
<?php
use Illuminate\Foundation\Application;
use
   Illuminate\Foundation\Configuration\Exceptions;
use
   Illuminate\Foundation\Configuration\Middleware;
use
   Illuminate\Foundation\Http\Middleware\VerifyCsr
   fToken;
return Application::configure(basePath:
   dirname(__DIR__))
    ->withRouting(
        web: __DIR__ . '/../routes/web.php',
        api: __DIR__ . '/../routes/api.php',
        commands: __DIR__ .
   '/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware(function (Middleware
   $middleware) {
        //
        $middleware->web(append: [
            VerifyCsrfToken::class,
        ]);
    })
```

TABEL SOURCE CODE 2 TAMBAHKAN @CSRF TIAP FORM BLADE

```
…….
        <h1>Login Guru</h1>
        <p>SIM - Akademik</p>
<form action="{{
   route('loginguru.submit') }}"
   method="POST" class="form-
   container">
            @csrf

            @if ($errors->any())
            <div style="color:red;
   margin-bottom:10px;">
                {{ $errors->first()
   }}
            </div>
            @endif
…….
```

2) Validasi Unggahan File Upload Mimes dan Type Mime

Aturan mimes digunakan untuk memvalidasi file berdasarkan ekstensi atau label yang diizinkan. Ketika sistem menetapkan mimes:pdf,docx,pptx, maka hanya file dengan ekstensi tersebut yang akan diterima. Sementara itu, mime type pemeriksaan yang lebih eksplisit dan memastikan bahwa tipe konten benar-benar sesuai dengan yang diizinkan oleh sistem. Mime type memberitahu browser atau server jenis file apa yang sedang dikirim (misalnya: apakah itu gambar JPG, dokumen PDF, atau file audio MP3), bukan hanya sekadar memeriksa ekstensi (.jpg, .pdf)[4].

Celah *File Upload Vulnerability* merupakan salah satu ancaman paling kritis karena memungkinkan penyerang mengunggah berkas berbahaya seperti *web shell* (misal: .php). Jika berkas tersebut disimpan di direktori publik (dapat diakses langsung via URL browser), penyerang dapat mengeksekusi kode tersebut di server untuk mengambil alih kendali sistem kendali penuh (*Remote Code Execution*). Kolaborasi validasi mime dan controlled storage terbukti sangat efektif memitigasi risiko. Hasil pengujian skenario TCfile_001 hingga TCfile_005 menunjukkan sistem berhasil menolak berkas berbahaya, termasuk teknik bypass manipulasi ekstensi ganda seperti shell.php.jpg. Karena file tersimpan di luar folder publik, file tersebut tidak memiliki executable path langsung melalui URL, sehingga menutup peluang eksekusi web shell di server.

TABEL SOURCE CODE 1 VALIDASI UNGGAHAN FILE - MIMES

```
    public function store(Request $request)
     {
// Validasi input
    $request->validate([
    'judul' => 'required|string|max:255',
    'nama' => 'required|string|max:255',
    'deskripsi' => 'required|string',
    'waktu' => 'required|date_format:Y-m-d\TH:i',
    'kelas' => 'required|string|max:100',
    'file' =>
'required|files|mimes:pdf,docx,zip,jpg,png|max:1
0240',
 ]);
```

TABEL SOURCE CODE 4. VALIDASI UNGGAHAN FILE - TYPE MIME

```
public function store(Request $request)
{
 $request->validate([
            'judul_materi' =>
   'required|string|max:255',
            'nama_guru' =>
   'required|string|max:255',
            'mata_pelajaran' =>
   'required|string|max:255',
            'waktu_dibagikan' => 'required|date',
            'kelas' => 'required|string|max:10',
'file_materi' =>
    'required|file|mimes:pdf,docx,pptx|mimetype
    s:application/pdf,application/vnd.openxmlfo
    rmats-
    officedocument.wordprocessingml.document,ap
    plication/vnd.openxmlformats-
    officedocument.presentationml.presentation|
    max:10240', // Validasi file


        ]);
```

3) Validasi Input

Validasi input adalah proses memastikan data yang masuk memenuhi aturan tertentu sebelum diproses. Contoh sederhana bisa berupa pengecekan kolom form yang wajib diisi terdapat dalam permintaan, ataupun kolom form harus diisi dengan pola tertentu sesuai kebutuhan aplikasi[11]. Menerapkan kelas Validator bawaan Laravel untuk membatasi tipe data, panjang karakter maksimal, keunikan data (unique), serta format isian form yang diizinkan. Strategi ini efektif membatasi ruang gerak penyerang dalam melakukan injeksi logika pada formulir pendaftaran dan autentikasi. Hasil uji TCSQL_001 menunjukkan payload ' OR '1'='1 langsung patah pada proses validasi atau dianggap sebagai string literal biasa tanpa merusak struktur kueri SQL basis data.

TABEL SOURCE CODE 2 VALIDASI INPUT

```
……
 public function submitRegistrasi(Request
   $request)
    {
// VALIDASI INPUT
$validator = Validator::make($request-
   >all(), [
'nama' => 'required|string|max:100',
'nip' =>
   'required|string|max:8|unique:gurus,nip'
   ,
'alamat' => 'required|string|max:255',
'jeniskelamin' => 'required|in:L,P',
'username'=>'required|string|max:50|unique:
   gurus,username',
'password' =>
   'required|string|min:6|confirmed',
   //gunakan field password_confirmation di
   form
        ]);
……
```

4) Blade Escaping

Laravel blade secara otomatis membersihkan output berbahaya menggunakan sintaks kurung kurawal ganda {{ }} pada komponen UI Laravel Blade. Fungsi ini mengubah karakter berbahaya seperti < menjadi < dan > menjadi ">"[5]. hasilnya, browser hanya menampilkan teks script tersebut sebagai teks biasa dan tidak mengeksekusinya sebagai program. Berdasarkan hasil pengujian otomatis dan manual (TCXSS_001 - TCXSS_009), strategi ini efektif menampilkan payload berbahaya sebagai teks biasa, kecuali pada satu komponen yaitu fitur Kalender Akademik. Kegagalan pada satu titik ini mengindikasikan bahwa output encoding belum diterapkan secara merata di seluruh komponen aplikasi, yang menjadi catatan krusial untuk perbaikan sistem selanjutnya.

TABEL SOURCE CODE 7 BLADE ESCAPING

```
……
 <tr>
        <td>{{ $g->id }}</td>
        <td>{{ $g->nama }}</td>
        <td>{{ $g->nip }}</td>
        <td>{{ $g->alamat }}</td>
……
```

5) Enskripsi Data Sensitif

Laravel menawarkan dua fitur enkripsi yakni encrypt() dan decrypt(). Fungsi encrypt() melibatkan proses pengamanan informasi sensitif pengguna (contohnya: nomor identitas, alamat, NIK, KTP, dan data pribadi lainnya) yang dapat dibaca dengan mengubah bentuk yang tidak dapat dimengerti saat melakukan simpanan ke database. Sedangkan decrypt() menampilkan data kembali dalam bentuk semula pada tampilan[5]. Strategi ini sangat efektif mengamankan data diam. Meskipun tidak mencegah serangan siber secara langsung, enkripsi berhasil meminimalkan dampak keparahan (seperti sanksi hukum/kebocoran privasi) apabila database berhasil diretas oleh pihak luar.

TABEL SOURCE CODE 3 ENSKRIPSI

```
……
  // INSERT DATA (lebih aman menggunakan
   Eloquent atau Query Builder dengan
   binding)
      DB::table('gurus')->insert([
       'nama' => htmlspecialchars($request-
   >nama), // escape output
      'nip' => Crypt::encryptString
   ($request->nip), //dienkripsi
……
```

TABEL SOURCE CODE 4 DESKRIPSI

```
……
public function index()
    {
         // Ambil semua guru, 10 per halaman
        $gurus = Guru::orderBy('id', 'asc')-
   >paginate(10);

        // Dekripsi NIP untuk tampilan full di UI
        foreach ($gurus as $guru) {
            try {
                $guru->nip =
   Crypt::decryptString($guru->nip); //dekripsi
            } catch (DecryptException $e) {
                /// NIP lama belum terenkripsi,
   tampil apa adanya
                $guru->nip = $guru->nip;


……
```

6) Rate Limiting Login

Rate limiting adalah strategi keamanan yang membatasi jumlah percobaan login atau permintaan yang dilakukan oleh pengguna di suatu situs web. Mekanisme ini berfungsi dengan menghitung jumlah permintaan yang dikirim oleh suatu alamat IP, akun, atau pengguna dalam periode waktu tertentu, misalnya maksimal 5 kali percobaan dalam 1 menit. Apabila jumlah permintaan tersebut melebihi batas yang telah ditentukan, sistem akan secara otomatis membatasi atau menolak permintaan selanjutnya untuk sementara waktu[12]. Tanpa adanya batasan, penyerang dapat melakukan serangan *Brute Force* secara masif menggunakan bot untuk menebak kombinasi username dan password pengguna.

TABEL SOURCE CODE 5 RATE LIMITING LOGIN

```
…
Route::post('/login',
   [LoginController::class,
   'login']) -
   >middleware(('throttle:5,1'));
….
Route::post('/loginguru',
   [LoginController::class, 'loginGuru'])-
   >middleware('throttle:5,1')-
   >name('loginguru.submit');
……
```

B. *Pengujian STLC*

a. *Requirement Analysis*

Pada tahap ini dilakukan identifikasi kebutuhan pengujian keamanan sistem. Lima jenis kerentanan yang akan diuji telah ditentukan, yaitu:

1) SQL Injection
2) Cross Site Scripting (XSS)
3) Cross Site Request Forgery (CSRF)
4) File Upload Vulnerability

fase ini dirancang untuk menentukan cakupan lingkup pengujian yang akan dilakukan.

b. Test Planning

Fase ini menciptakan rencana pengujian yang didasarkan pada analisis kebutuhan. Aktivitas yang dilakukan adalah menentukan metode pengujian melalui black-box testing, menyusun skenario pengujian yang sesuai dengan standar keamanan, serta menggunakan tools yang digunakan adalah browser untuk mengakses web akalink.

c. Test Case Development

Dalam fase ini, dilakukan penyusunan dan pengembangan test case yang digunakan sebagai acuan dalam proses pengujian sistem. Tujuan dari proses ini adalah untuk memastikan bahwa setiap fitur dan mekanisme keamanan pada aplikasi dapat diuji secara sistematis dan terukur.

TABEL I
TEST CASE DEVELOPMENT

No	Test skenario ID	Nm_Test	Test kasus ID	Test case	Langkah uji	Hasil yang diharapkan
1	TS_001	SQL Injection	TCSQL_001	Melakukan percobaan login guru. Masukkan payload ' OR '1'='1 di kolom username atau password, klik login	Username = kosong Password = ' OR '1'='1	Sistem menolak login
2			TCSQL_002	Login admin masukkan payload ' -- di kolom username	username : admin' -- password : kosong/bebas	Sistem menolak login
3			TCSQL_003	Login admin masukkan payload ' OR 1=1 --	Username = valid Pw = ' OR 1=1 --	Sistem menolak login
4			TCSQL_004	Langsung akses http://test/admin/guru/edit/1 tanpa melakuka	Masukkan http://test/admin/guru/edit/1	Sistem menolak akses atau tidak berpindah halaman
				n login admin		
5			TCSQL_005	Login admin>data guru>klik tombol atau akses edit salah satu data guru (misal:Yuli sari)> lalu akses langsung melalui url http://test/admin/guru/edit/9	Akses url http://test/admin/guru/edit/9 ketika sedang akses edit data guru lain	Sistem menolak akses
6			TCSQL_006	Manipulasi parameter URL edit guru. Ubah parameter ID di URL /admin/guru/edit/1' OR '1'='1	Masukkan http://test/admin/guru/edit/1' OR '1'='1	Sistem menolak parameter tidak
7			TCSQL_007	Admin-fitur data guru. Input payload ' OR 1=1 --pada form tambah guru lalu klik simpan	Isi form Nama guru: ' OR 1=1 -- Untuk form lainya isi normal	Sistem menolak input atau menyimpannya sebagai string biasa tanpa error SQL
8			TCSQL_008	Admin-melakukan tambah kalender akademik dengan mengisi payload berikut '; DROP TABLE kalender_akademik; --	Masukkan pada judul: '; DROP TABLE kalender_akademik; -- lalu simpan	Sistem menolak input berbahaya atau menyimpannsebagai string biasa. Tidak terjadi error SQL. Tabel kalender akademik tidak terhapus. Data lain tidak terpengaruh
9			TCSQL_009	Guru – melakukan upload materi dengan memasukkan	Input judul materi: 1' OR 1=1 -- Form lain isi bebas	Sistem menolak input

No	Test skenario ID	Nm_ Test	Test kasus ID	Test case	Langkah uji	Hasil yang diharapkan
				payload berikut Materi 1' OR 1=1 --		
10	TS_002	XSS	TCXSS_001	Guru-tambah laporan. Memasukkan payload <script>alert('XSS')</script> ke judul di fitur tambah laporan	Isi judul: <script>alert('XSS')</script>	Script tidak dieksekusi, hanya tampil sebagai teks biasa.
11			TCXSS_002	Menguji apakah kolom pencarian dapat menjalankan script pada admin-data guru	Masukkan payload <svg onload=alert(1)> pada kolom pencarian kemudian klik search	Sistem tidak menjalankan script dan tidak menampilkan popup alert
12			TCXSS_003	Menguji apakah kolom pencarian dapat menjalankan script pada admin-siswa	Masukkan payload <svg onload=alert(1)> pada kolom pencarian kemudian klik search	Sistem tidak menjalankan script dan tidak menampilkan popup alert
13			TCXSS_004	Menguji apakah field alamat rentan XSS	Data guru> tambah data> isi form seperti biasa pada bagian form alamat masukkan payload "<img src=x onerror=alert(1)>" klik simpan	Sistem menampilkan payload sebagai teks biasa
14			TCXSS_005	Uji parameter id di admin-data guru	Masukkan url berikut http://test/admin/guru/edit/1<script>alert	Sistem menolak request/menampilkan error
					(1)</script>	
15			TCXSS_006	Menguji apakah field kalender rentan XSS di admin – fitur kalender	Kalender akademik > tambah kalender > isi form seperti biasa pada bagian form judul masukkan payload <img src=x onerror=alert(1)> klik simpan	Sistem menampilkan payload sebagai teks biasa
16			TCXSS_007	Menguji apakah kolom pencarian dapat menjalankan script pada guru-pengumpulan	Masukkan payload: <svg onload=alert(1)> pada kolom pencarian > klik search	Sistem tidak menjalankan script dan tidak menampilkan popup alert
17			TCXSS_008	Menguji apakah input pada form tambah kalender rentan terhadap XSS	Buka halaman admin-kalender > klik tambah event > masukkan payload <script>alert('XSS')</script> pada kolom judul > klik simpan	Script tidak dieksekusi, hanya ditampilkan sebagai teks biasa
18			TCXSS_009	Menguji XSS pada admin - fitur edit data guru	Login admin > buka data guru > edit salah satu data > masukkan payload <script>alert('XSS')</script> pada kolom nama dan alamat > klik simpan	Sistem menampilkan sebagai teks biasa dan tidak mengeksekusi script

No	Test skenario ID	Nm_ Test	Test kasus ID	Test case	Langkah uji	Hasil yang diharapkan
19	TS_003	CSRF	TCSRF_001	Memastikan sistem menolak request POST tanpa token CSRF.	Login admin >buka halaman tambah guru> buka inspect element> hapus <input type="hidden" name="_token" value=…..> Klik tambah	Sistem menampilkan pesan eror
20			TCSRF_002	Memastikan sistem menolak token yang dimodifikasi	Login guru > buka upload materi > isi form > buka inspect > ubah <input type="hidden" name="_token" value="123456abcdef" … > lalu klik unggah	Sistem menampilkan error 419.
21			TCSRF_003	Menguji request Login Guru dari luar aplikasi tanpa token CSRF	Kirim request POST menggunakan tools (Postman /Burp Suite) pada halaman Login Guru tanpa menyertakan token CSRF	Sistem menolak request dan menampilkan error (419 Page Expired)
22			TCSRF_004	Menguji reuse token CSRF pada Login Guru (token lama digunakan kembali)	Ambil token CSRF lama Login Guru> gunakan kembali setelah logout/login ulang> kirim request	Sistem menolak request karena token tidak valid
23			TCSRF_005	Menguji CSRF Login Admin-Data guru pada fitur tombol hapus data	Login admin> buka data guru> hapus data tanpa token CSRF (hapus via inspect atau tools)	Sistem menolak request penghapusan dan menampilkan error
24			TCSRF_006	Menguji request Login Admin dari luar aplikasi tanpa token CSRF	Kirim request POST menggunakan tools (Postman /Burp Suite) pada halaman Login Guru tanpa token	Sistem menolak request dan menampilkan error (419 Page Expired)
25			TCSRF_007	Menguji reuse token CSRF pada Login Admin (token lama digunakan kembali)	Ambil token CSRF lama Login Admin> gunakan kembali setelah logout/login ulang> kirim request	Sistem menolak request karena token tidak valid
26	TS_004	File Upload vulnerability	TCfile_001	Menguji apakah sistem menolak file bereksten si .php pada fitur guru-upload tugas	Isi form seperti biasa, pada upload file pilih file shell.php. lalu klik upload	Sistem menolak file. File tidak tersimpan.
27			TCfile_002	Menguji apakah sistem menolak file bereksten si .exe pada fitur guru-upload tugas	Isi form seperti biasa, pada upload file pilih file bereksten si .exe lalu klik upload	Sistem menolak file. File tidak tersimpan

No	Test skenario ID	Nm_ Test	Test kasus ID	Test case	Langkah uji	Hasil yang diharapkan
28			TCfile_003	Menguji apakah sistem menolak file bereksten si .php pada fitur guru-upload materi	Isi form seperti biasa, pada upload file pilih file shell.php klik upload	Sistem menolak file dan file tidak tersimpan
29			TCfile_004	Menguji apakah sistem menolak file bereksten si .exe pada upload materi	Isi form seperti biasa, pada upload file pilih file .exe klik upload	Sistem menolak file dan file tidak tersimpan
30			TCfile_005	Menguji upload file dengan double extension (bypass) pada fitur guru-upload tugas maupaun fitur guru-upload materi	Isi form seperti biasa lalu upload file dengan nama: shell.php. jpg	Sistem mendeteksi file berbahaya dan menolak upload

d. Test Case Execution

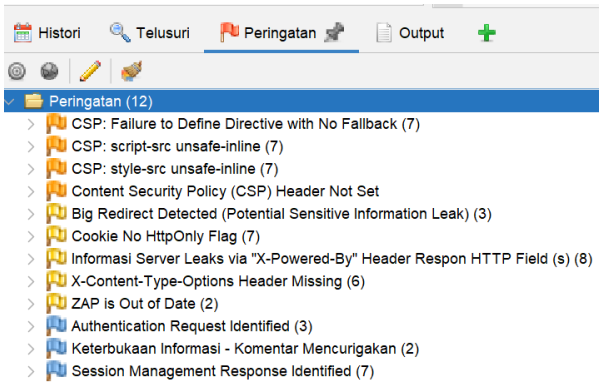
Aktivitas pengujian dilaksanakan sesuai dengan prosedur yang telah ditentukan untuk memastikan sistem berjalan sesuai harapan sebagaimana mestinya serta untuk mengidentifikasi potensi kesalahan maupun kerentanan yang mungkin masih terdapat dalam aplikasi. Setiap test case dari tabel development dijalankan dengan hasil sebagai berikut:

TABEL II
TEST EXECUTION

No	Test skenario ID	Nama Tes	Tes kasus ID	Hasil	Analisis atau catatan temuan
1	TS_001	SQL Injection	TCSQL_001	Lulus	Sistem berhasil menolak dengan respon “username atau password salah”
2			TCSQL_002	Lulus	Sistem berhasil menolak dengan respon “username atau password salah”
3			TCSQL_003	lulus	Sistem berhasil menolak dengan respon “username atau password salah”
4			TCSQL_004	Lulus	Sistem berhasil tidak berpindah halaman
5			TCSQL_005	Lulus	Sistem berhasill menolak akses dan menampilkan pesan eror 403 forbidden
6			TCSQL_006	lulus	Sistem Sistem berhasill menolak akses payload dan menampilkan pesan eror 403 forbidden
7			TCSQL_007	lulus	Script tidak dieksekusi, payload disimpan sebagai teks biasa
8			TCSQL_008	lulus	Script tidak dieksekusi, payload disimpan sebagai teks biasa
9			TCSQL_009	lulus	Sistem berhasil menolak input payload
10	TS_002	XSS	TCXSS_001	lulus	Payload tidak dieksekusi, menampilkan sebagai teks biasa
11			TCXSS_002	lulus	Payload tidak dieksekusi
12			TCXSS_003	lulus	Payload tidak dieksekusi
13			TCXSS_004	lulus	Payload tidak dieksekusi, menampilkan sebagai teks biasa
14			TCXSS_005	lulus	Script tidak dieksekusi, Sistem menampilkan pesan eror 404 not found
15			TCXSS_006	gagal	Sistem masih rentan terhadap serangan XSS pada fitur kalender akademik. Payload yang dimasukkan berhasil dieksekusi dan menampilkan popup alert “127.0.0.1:8000 says 1”. Hal ini menunjukkan bahwa input pengguna belum difilter atau di-escape dengan baik pada bagian tersebut
16			TCXSS_007	lulus	Payload tidak dieksekusi

No	Test skenario ID	Nama Tes	Tes kasus ID	Hasil	Analisis atau catatan temuan
17			TCXSS_008	lulus	Payload tidak dieksekusi, menampilkan sebagai teks biasa
18			TCXSS_009	lulus	Payload tidak dieksekusi, menampilkan sebagai teks biasa
19	TS_003	CSRF	TCSRF_001	lulus	Sistem menampilkan eror 419 page expired
20			TCSRF_002	lulus	Sistem menampilkan eror 419 page expired
21			TCSRF_003	lulus	Sistem menampilkan eror 419 page expired
22			TCSRF_004	lulus	Sistem menampilkan eror 419 page expired
23			TCSRF_005	lulus	Sistem menampilkan eror 419 page expired dan ketika refresh/kembali halaman sebelumnya data tersebut tidak hilang
24			TCSRF_006	Lulus	Sistem menampilkan eror 419 page expired
25			TCSRF_007	Lulus	Sistem menampilkan eror 419 page expired
26	TS_004	File Upload vulnerability	TCfile_001	lulus	Sistem menolak file yang tidak diminta seperti berekstensi shell.php dan terdapat pesan alert jenis file yang boleh diunggah
27			TCfile_002	lulus	Sistem menolak file yang tidak diminta seperti berekstensi .exe dan terdapat pesan alert jenis file yang boleh diunggah
28			TCfile_003	lulus	Sistem menolak file yang tidak diminta seperti berekstensi .php dan terdapat pesan alert jenis file yang boleh diunggah
29			TCfile_004	lulus	Sistem menolak file yang tidak diminta seperti berekstensi .exe dan terdapat pesan alert jenis file yang boleh diunggah
30			TCfile_005	Lulus	Sistem menolak file yang tidak diminta seperti berekstensi shell.php.jpg dan terdapat pesan alert jenis file yang boleh diunggah

Setelah dilakukan implementasi strategi pengamanan pada kode program, penulis melakukan pengecekan kembali dengan alat pengujian otomatis untuk mengevaluasi efektivitas perbaikan yang telah diterapkan. Pengujian ini ditujukan untuk memastikan bahwa kerentanan yang sebelumnya teridentifikasi telah diminimalkan.



Gambar 3 Scanning owasp zap setelah implementasi

TABEL III KETERANGAN LEVEL

Nama level resiko	Keterangan
Merah/*High*	Kerentanan sangat berbahaya dan kritis, harus segera diperbaiki
Oren/*Medium*	Cukup berbahaya jika dibiarkan bisa dimanfaatkan penyerang
Kuning/*Low*	Resiko kecil tidak darurat tetapi tetap perlu diperbaiki
Biru/*Information*	Bukan celah langsung hanya berupa informasi tambahan. Hanya memberitahu adanya fitur tertentu yang terdeteksi

TABEL IV PERBANDINGAN SCANNING OWASP ZAP

No	Level Resiko	Jumlah Temuan Sebelumnya	Jumlah Temuan Sesudah Penerapan
1	Merah/*High*	-	-
2	Oren/*Medium*	42	22
3	Kuning/*Low*	144	26
4	Biru/*Information*	47	12
TOTAL		233	60

Berdasarkan hasil pengujian ulang menggunakan OWASP ZAP menunjukkan adanya penurunan jumlah temuan kerentanan yaitu berjumlah 60 temuan, serta penurunan jenis

kerentanan dengan jumlah 12 kerentanan. Dibandingkan sebelum implementasi pengamanan terdapat 233 temuan dengan 14 jenis kerentanan. Beberapa kerentanan yang sebelumnya terdeteksi berhasil diminimalkan.

IV. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa implementasi strategi pengamanan pada Akalink seperti, validasi input, penggunaan token CSRF, perlindungan file upload, terbukti dapat mencegah serangan umum pada aplikasi web, di antaranya *SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF)*, serta File Upload Vulnerability. Penerapan mekanisme keamanan tersebut menunjukkan bahwa penguatan pada sisi kode dan konfigurasi sistem berperan penting dalam meningkatkan ketahanan aplikasi terhadap potensi eksploitasi. Selain itu, penerapan *Software Testing Life Cycle* (STLC) memberikan struktur yang sistematis dalam proses pengujian keamanan web, sehingga identifikasi kerentanan dilakukan secara lebih terarah, terukur, dan terdokumentasi dengan baik. Dengan pendekatan ini, setiap hasil pengujian dapat ditelusuri kembali berdasarkan tahapan yang jelas.

Hasil pengujian menggunakan OWASP ZAP sebelum implementasi strategi pengamanan menunjukkan terdapat 14 jenis kerentanan dengan total 233 temuan. Setelah dilakukan penerapan strategi pengamanan kode, jumlah kerentanan menurun menjadi 12 jenis dengan total 60 temuan. Penurunan ini menunjukkan bahwa strategi pengamanan yang diterapkan cukup efektif dalam menutup sebagian celah keamanan, khususnya pada aspek validasi input, perlindungan CSRF, serta pengamanan unggah berkas. Namun demikian, beberapa kerentanan masih ditemukan yang umumnya berkaitan dengan optimalisasi konfigurasi Content Security Policy (CSP) serta penguatan pengaturan keamanan tambahan lainnya, sehingga tingkat risiko sistem setelah implementasi berada dalam kondisi yang lebih terkontrol dibandingkan sebelumnya. Selanjutnya, implementasi strategi pengamanan yang diterapkan tidak mengganggu fungsi utama sistem.

Seluruh fitur utama seperti login, unggah materi, unggah tugas, serta pengelolaan data tetap dapat berjalan dengan baik setelah penerapan strategi pengamanan kode. Berdasarkan pengujian skenario TCXSS_006, ditemukan kerentanan Cross-Site Scripting (XSS) pada fitur Kalender Akademik, khususnya pada field Judul Kalender yang digunakan untuk menyimpan dan menampilkan informasi kegiatan akademik. Kerentanan terjadi karena tidak adanya mekanisme validasi, sanitasi, atau output encoding yang memadai sehingga payload yang dimasukkan berhasil dieksekusi dan menghasilkan popup alert "127.0.0.1:8000 says 1" oleh browser.

Dampak dari kerentanan ini cukup serius karena penyerang dapat menyisipkan skrip berbahaya yang akan dijalankan oleh browser pengguna lain yang mengakses halaman kalender akademik. Jika dieksploitasi lebih lanjut, penyerang dapat mencuri cookie sesi pengguna, mengambil informasi autentikasi, melakukan pengalihan ke situs phishing, memodifikasi tampilan halaman, atau menjalankan aksi tertentu atas nama pengguna yang sedang login. Untuk mengatasi kerentanan tersebut, pengembang disarankan menerapkan validasi dan sanitasi input pada seluruh field yang menerima data dari pengguna, termasuk field Judul Kalender. Implementasi Content Security Policy (CSP) juga direkomendasikan untuk membatasi eksekusi skrip yang tidak berasal dari sumber terpercaya. Setelah perbaikan dilakukan, pengujian ulang perlu dilaksanakan untuk memastikan payload XSS tidak lagi dapat dieksekusi dan hanya ditampilkan sebagai teks biasa sesuai dengan hasil yang diharapkan.

V. DAFTAR PUSTAKA

[1] H. Herdianto, H. Prasetyo Utomo, dan Y. Supendi, "Implementasi Secure Coding Pada Aplikasi Pelayanan Online Lab El Shaddai," *Jurnal InfoSecure (JISEC)*, vol. 3, no. 2, 2022.

[2] T. Ariyadi, T. L. Widodo, N. Apriyanti, dan F. S. Kirana, "Analisis Kerentanan Keamanan Sistem Informasi Akademik Universitas Bina Darma Menggunakan OWASP," *Techno.Com*, vol. 22, no. 2, hlm. 418–429, 2023, doi: 10.33633/tc.v22i2.7562.

[3] N. Saragih dan T. Zebua, "Analisis Keamanan dan Implementasi Secure Code Pada Pengembangan Keamanan Websitefikom-methodist.com Menggunakan Penetration Testing dan CVSS," *Jurnal Informatika Kaputama (JIK)*, vol. 7, no. 2, hlm. 242–253, 2023, doi: 10.59697/jik.v7i2.233.

[4] S. Steven, W. Rifaldi, dan U. Nugraha, "Strategi Pengamanan Front-end dalam Pengembangan Website," *JUSTINFO | Jurnal Sistem Informasi dan Teknologi Informasi*, vol. 1, no. 1, hlm. 42–53, 2023, doi: 10.33197/justinfo.vol1.iss1.2023.1200.

[5] S. M. Husain, L. Azhari, M. L. Aksani, dan S. A. Saputra, "Analisis Dan Implementasi Fitur Keamanan Aplikasi Pada Framework Laravel," *JIKA (Jurnal Informatika)*, vol. 8, no. 3, hlm. 281, 2024, doi: 10.31000/jika.v8i3.11198.

[6] F. A. Hassanah, E. Ryansyah, F. M. Setiawan, R. Alamsyah, dan A. S. Y. Irawan, "Analisis Kerentanan Keamanan Menggunakan OWASP ZAP dan Pengujian Manual pada Tampilan Antarmuka Laman PDDikti," *JELIKU (Jurnal Elektronik Ilmu Komputer Udayana)*, vol. 13, no. 3, hlm. 671–678, 2025, doi: 10.24843/JLK.2025.v13.i03.p14.

[7] A. Arfan, "Penerapan STLC dalam Pengujian Automation Aplikasi Mobile (Studi kasus: LMS Amikom Center PT.GIT Solution)."

[8] G. Kusuma, "Implementasi Owasp Zap Untuk Pengujian Keamanan Sistem Informasi Akademik," *Jurnal Teknologi Informasi: Jurnal Keilmuan dan Aplikasi Bidang Teknik Informatika*, vol. 16, no. 2, hlm. 178–186, 2022, doi: 10.47111/jti.v16i2.3995.

[9] Shalsabila Titanie Harianto, Zahrah Aliyah Rachman, Aliyyah Nabiilah Farahdita, Fahryan Putra Ramadi, Agung Brastama, dan Siti Mukaromah, "Analisis Keamanan Website E-learning ILMU2 Menggunakan Metode Open Web Application Security Project ZAP (OWASP ZAP)," *Neptunus: Jurnal Ilmu Komputer*

Dan Teknologi Informasi, vol. 3, no. 3, hlm. 21–36, Jun 2025, doi: 10.61132/neptunus.v3i3.887.

[10] Ruliansyah, Tukino, Baenil Huda, dan April Lia Hananto, "Penerapan Software Testing Life Cycle Pada Pengujian Otomatisasi Platform Dzikra," *CSRID (Computer Science Research and Its Development Journal)*, vol. 15, no. 1, hlm. 01–11, Mei 2023, doi: 10.22303/csrid.15.1.2023.01-11.

[11] V. Laurensius *dkk.*, "Analisis Dan Perancangan Keamanan Frontend Dalam Aplikasi Web: XSS dan CSRF," *Prosiding SISFOTEK*, vol. 8, no. 1, hlm. 12–18, 2024, [Daring]. Tersedia pada: https://seminar.iaii.or.id/index.php/SISFOTEK/article/view/438

[12] M. K. Fahram dan V. V. Kusuma, "Pengembangan Aplikasi Pendaftaran Akun Email Microsoft 365 Dan Google Workspace Berbasis Laravel Dan React Di Politeknik Piksi Input Serang". Vol. 7 No. 2 (2025): Vol 7 No 2 (Agustus 2025): Journal Of Innovation And Future Technology (Iftech).