PETIR: Jurnal Pengkajian dan Penerapan Teknik Informatika
Vol. 17, No. 2, September 2024, P-ISSN 1978-9262, E-ISSN 2655-5018
DOI: https://doi.org/10.33322/petir.v17i2.2493

Uji Mutasi pada Penerapan Token Mitigasi Kerentanan Cross Site Request Forgery

Iqbal Najmul[1]; Riva Sundara[1)]; Abdurrasyid[2]; Gusti Ayu Putri Saptawati[1]*

1. Institut Teknologi Bandung, Jl. Ganesa No.10, Lb. Siliwangi, Kecamatan Coblong, Kota Bandung, Jawa Barat 40132 Indonesia
2. Institut Teknologi PLN, Menara PLN, Jl. Lingkar Luar Barat, Duri Kosambi, Cengkareng, Jakarta Barat, DKI Jakarta 11750 Indonesia

[*)]Email: riva.hakim@gmail.com

Received: 02 Juni 2024 | Accepted: 13 Desember 2024 | Published: 10 Januari 2025

ABSTRACT

Web application security is a critical concern due to the many vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). These vulnerabilities are exploited by attackers to gain unauthorized access and corrupt web applications. Our research focuses on Cross-Site Request Forgery (CSRF) vulnerability analysis using a mutation testing approach, which implements 5 mutation operators that mutate input tokens on input forms. We introduced automated tools to identify and address CSRF vulnerabilities using secret token patterns. This tool enhances the security of PHP-based web applications without sacrificing their functionality. When a vulnerability is detected, the application notifies the user to fix it immediately, we use the mutation score indicator as a measurement tool to the extent to which mutation testing is successful, the results of all 1022 mutations produced can be stopped with a 100% presentation showing that the mutation operators used can work well to detect the resulting mutant.

Keywords: *CSRF, Broken Access Control, Mutation Testing.*

ABSTRAK

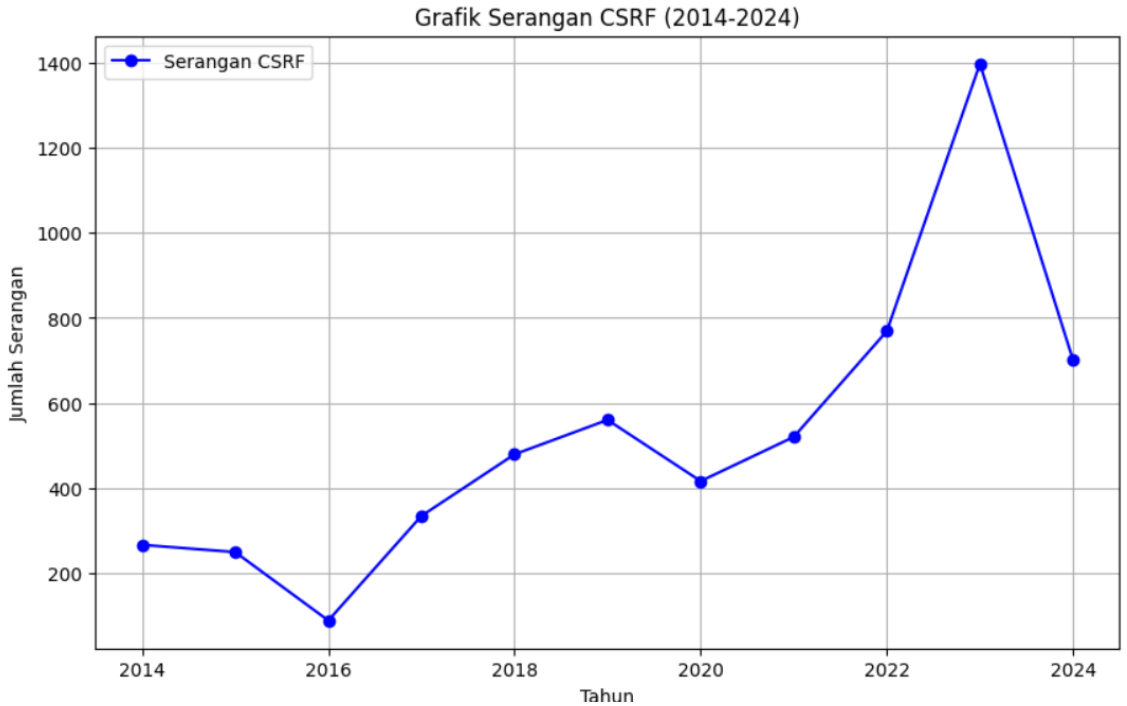
Keamanan aplikasi web merupakan perhatian kritis karena banyaknya kerentanan seperti SQL Injection (SQLi), Cross-Site Scripting (XSS), dan Cross-Site Request Forgery (CSRF). Kerentanan-kerentanan ini dieksploitasi oleh penyerang untuk mendapatkan akses yang tidak sah dan merusak aplikasi web. Penelitian kami berfokus pada analisis kerentanan Cross-Site Request Forgery (CSRF) dengan menggunakan pendekatan pengujian mutasi, yang menerapkan 5 operator mutasi yang memutasi token input pada form input. Kami memperkenalkan alat otomatis untuk mengidentifikasi dan mengatasi kerentanan CSRF menggunakan pola token rahasia. Alat ini meningkatkan keamanan aplikasi web berbasis PHP tanpa mengorbankan fungsionalitasnya. Ketika kerentanan berhasil dideteksi, aplikasi akan memberi tahu pengguna agar segera dapat diperbaiki, indikator skor mutasi kami gunakan sebagai alat pengukuran sejauh mana pengujian mutasi berhasil dilakukan, hasilnya dari 1022 mutasi yang dihasilkan seluruhnya dapat dihentikan dengan presentasi 100% menunjukan bahwa operator mutasi yang digunakan dapat bekerja dengan baik untuk mendeteksi mutan yang dihasilkan.

Kata kunci: *CSRF, Broken Access Control, Mutation Testing.*

PETIR: Jurnal Pengkajian dan Penerapan Teknik Informatika
Vol. 17, No. 2, September 2024, P-ISSN 1978-9262, E-ISSN 2655-5018
DOI: https://doi.org/10.33322/petir.v17i2.2493

1. PENDAHULUAN

Keamanan aplikasi web menjadi kebutuhan yang semakin krusial seiring dengan meningkatnya ancaman dan kerentanan yang ditemukan pada aplikasi web. Di era modern yang semakin marak bentuk transaksi jual-beli, perbankan dan lainnya. Kerentanan seperti SQL *Injection* (SQLi*), Cross-Site Scripting* (XSS), dan *Cross-Site Request Forgery* (CSRF)[1] sering kali menjadi metode yang banyak digunakan peretas untuk mendapatkan akses yang tidak sah untuk melakukan tindakan yang merugikan pengguna[2]. Setiap tahun serangan relatif terus meningkat, puncaknya pada tahun 2023 tercatat terjadi 1.400 serangan. Berikut ini adalah tren serangan CSRF dari tahun 2014-2024[3]:



Gambar 1. Grafik serangan CSRF tahun 2014-2024

CSRF merupakab bagian dari *Broken Access Control* dan masuk peringkat pertama dari 10 TOP Vulnerabiliti menurut OWASP pada tahun 2021[4]. CSRF dapat dilakukan dengan berbagai cara. Serangan tersebut bisa dilakukan menggunakan tag HTML IMG atau URL yang dibuat khusus yang di-attach ke dalam aplikasi target. Hal ini bekerja dengan baik karena korban sudah memiliki sesi ke dalam *website* tersebut. Cara lain adalah dengan membuat *website* yang memungkinkan pengguna membuat link atau gambar yang ketika diklik akan mengeksploitasi data yang dimiliki pengguna, kemudian kredensial si pengguna dimanfaatkan oleh *attacker*[5]. CSRF adalah jenis serangan yang meretas hak akses pengguna dalam melakukan aksi yang tidak diinginkan. Jenis kerentanan ini terjadi ketika server tidak memeriksa dengan benar apakah *request* dilakukan oleh pengguna yang sah atau bukan[6]. Permintaan dapat dipalsukan dan dikirim ke pengguna untuk membuat mereka melakukan tindakan yang tidak diinginkan seperti mengubah email, mengubah kata sandi, atau melakukan sebuah transaksi. Dengan demikian, sebuah web aplikasi sangat penting untuk memiliki mekanisme pencegahan yang efektif untuk melindungi integritas dan keamanan data dan hak akses pengguna.

Untuk melakukan mitigasi terhadap serangan CSRF dapat dilakukan dengan metode SAST. Metode ini dikenal juga sebagai "*white box testing*"[7] sangat ideal untuk pengujian integrasi kode

Vol. 17, No. 2, September 2024, P-ISSN 1978-9262, E-ISSN 2655-5018
DOI: https://doi.org/10.33322/petir.v17i2.2493

dalam proses pengembangan aplikasi. Metode ini memungkinkan pengembang untuk memeriksa struktur internal dan logika kode tanpa menjalankan program, sehingga dapat mendeteksi potensi kerentanan dan kesalahan lebih awal dalam siklus pengembangan. Dengan menggunakan *white box testing*, tim pengembang dapat memastikan bahwa setiap komponen kode memenuhi standar keamanan dan kualitas sebelum diintegrasikan ke dalam aplikasi. Pendekatan ini sangat penting dalam membangun *Secure Software Development Life Cycle* (SDLC) yang mengutamakan keamanan sejak tahap awal pengembangan. Selain itu, penerapan *white box testing* dalam *Continuous Integration* (CI) membantu memastikan bahwa setiap perubahan kode yang diintegrasikan secara otomatis diuji dan diverifikasi, mengurangi risiko cacat dan meningkatkan keandalan aplikasi secara keseluruhan.

Dalam menangani serangan CSRF dapat dilakukan beberapa cara diantaranya adalah menerapkan pola token rahasia yang dapat menjamin kredensial dari pengguna[8]. Penggunaan token lebih aman dan efektif secara kriptografi karena menghasilkan token yang unik setiap sesi. Token akan diverifikasi oleh server untuk memastikan sesuai dengan id pemilik.

Penelitian ini menggunakan pengujian mutasi yang digunakan untuk mengevaluasi kasus uji yang digunakan untuk memastikan kerentanan dalam suatu aplikasi web terutama kerentanan CSRF, Pengujian mutasi telah banyak dilakukan oleh para peneliti, namun untuk domain *software security* belum terlalu banyak dilakukan, Huang mengusulkan 13 operator mutasi untuk *SQL Injection* [9], mengulang apa yang sudah dilakukan Appelt pada tahun 2014 [10], sedangkan Loise mengusulkan 15 operator mutasi untuk jenis kerentanan *SQL Injection* dan XSS pada aplikasi berbasis java, Siavashi melakukan penilaian kerentanan *user* and session authentication pada web service yang merupakan no tujuh pada Top 10 OWASP dengan mengenalkan 8 operator mutasi [11] hal ini menunjukkan bahwa pengujian mutasi dapat membantu tester dalam memastikan kualitas kasus uji yang baik untuk menjamin kerentanan keamanan perangkat lunak.

Penelitian lain yang dilakukan oleh[2], penggunaan uji mutasi tradisional untuk memastikan validitas aplikasi web PHP ditemukan dapat menjadi metode yang efektif. Awalnya peneliti menyoroti pentingnya pengujian aplikasi web mengingat layanan yang disediakan oleh aplikasi web kepada miliaran pengguna di seluruh dunia. Peneliti juga membahas kurangnya teknik dan metode pengujian aplikasi web serta pentingnya mempertimbangkan sifat dinamis dari aplikasi web. Metode yang digunakan dalam penelitian ini adalah uji mutasi tradisional. Peneliti mendefinisikan 54 operator mutasi yang diklasifikasikan ke dalam enam kategori yang berbeda untuk menguji aplikasi web yang dibangun menggunakan bahasa pemrograman PHP. Peneliti juga mengimplementasikan sebuah alat prototipe bernama μWebPHP untuk secara otomatis menghasilkan mutan untuk aplikasi web PHP berdasarkan operator mutasi yang diidentifikasi. Hasil awal penelitian menunjukkan bahwa uji mutasi memungkinkan untuk aplikasi web PHP. Temuan ini menunjukkan bahwa uji mutasi dapat menjadi metode yang efektif dalam memastikan validitas aplikasi web PHP.

Kemudian [12] mengembangkan solusi baru untuk masalah pengujian integrasi aplikasi web menggunakan analisis mutasi. Operator mutasi baru didefinisikan, sebuah alat bernama webMuJava yang mengimplementasikan operator-operator ini disajikan, dan hasil dari studi kasus menerapkan alat ini untuk menguji aplikasi web kecil. Metode yang digunakan dalam penelitian ini adalah analisis mutasi. Analisis mutasi adalah teknik pengujian perangkat lunak yang menghasilkan mutan dari program asli dengan menerapkan operator mutasi. Mutan-mutan ini kemudian diuji dengan menggunakan rangkaian tes untuk melihat apakah tes tersebut dapat mendeteksi perubahan perilaku yang disebabkan oleh mutasi. Hasil penelitian menunjukkan bahwa penggunaan analisis mutasi dalam pengujian aplikasi web dapat membantu menemukan kesalahan, serta menunjukkan beberapa arah untuk perbaikan. Namun, penelitian juga menyoroti beberapa arah untuk perbaikan, termasuk

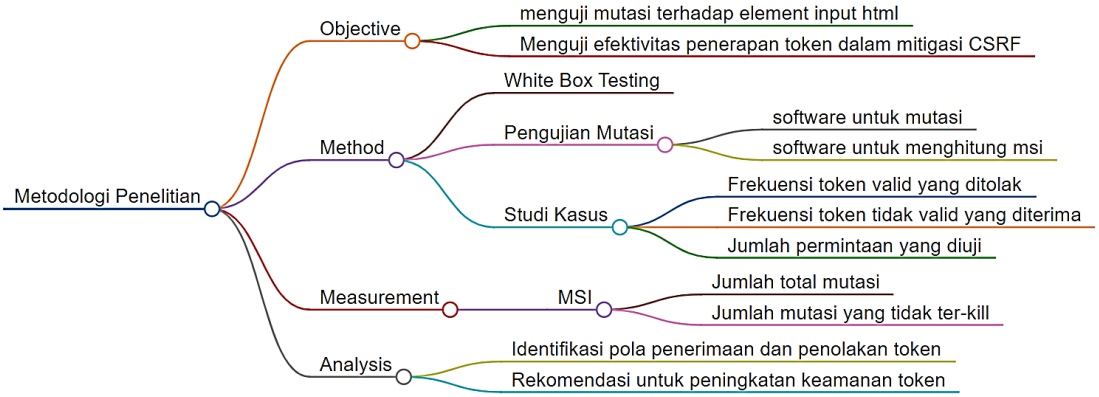
Vol. 17, No. 2, September 2024, P-ISSN 1978-9262, E-ISSN 2655-5018
DOI: https://doi.org/10.33322/petir.v17i2.2493

kebutuhan akan operator mutasi tambahan dan rencana untuk mengimplementasikan operator mutasi SQL. Selain itu, penelitian berencana untuk melakukan eksperimen yang lebih terkontrol menggunakan aplikasi web yang lebih besar, kompleks, dan industri untuk lebih mengevaluasi pendekatan ini.

Pada penelitian ini menerapkan 5 operator mutasi yang diadaptasi dari penelitian yang telah dilakukan oleh abdurrasyid yang mengenalkan 8 operator mutasi untuk memastikan kelayakan kasus uji untuk menjamin keamanan perangkat lunak dari kerentanan CSRF, adapun untuk pengukuran digunakan persamaan indikator skor mutasi [13].

2. METODE/PERANCANGAN PENELITIAN

2.1. Kerangka Pemikiran



Gambar 2. Kerangka Pemikiran

Penelitian ini menggunakan metode pengujian mutasi. Aplikasi yang dibangun bernama cyberPHP berkemampuan untuk menerapkan operator-operator mutasi yang dikembangkan. Uji mutasi dilakukan guna mengidentifikasi kesalahan syntax[14]. Berdasarkan penelitian yang telah dilakukan abdurrasyid pada tahun 2024 dengan merekomendasikan 8 operator mutasi yang dapat digunakan dalam pengujian mutasi untuk mendeteksi kerentanan CSRF untuk dilakukan pengujian mutasi, adapun pada penelitian ini mencoba untuk menggunakan 5 operator mutasi yang digunakan pada pembuatan tag input biasa digunakan untuk menyertakan token kedalam form sebuah aplikasi [13].

2.2. Alur Kerja Aplikasi

Aplikasi yang kami kembangkan untuk mempermudah dalam melakukan pengujian mutasi dibuat menggunakan bahasa python. Aplikasi ini dapat men-*genarate testcase* untuk kemudian dilakukan pengujian kepada *source* yang sudah dimutasi. Library yang digunakan untuk melakukan pengujian adalah phpunit versi 11, serta menggunakan composer sebagai paket manajer libray php yang sudah bisa digunakan oleh programmer php hampir diseluruh dunia. Rancangan alur dari penggunaan aplikasi untuk mutasi dapat dilihat pada gambar 3 sebagai berikut:

Vol. 17, No. 2, September 2024, P-ISSN 1978-9262, E-ISSN 2655-5018
DOI: https://doi.org/10.33322/petir.v17i2.2493

Gambar 3. Rancangan Alur Kerja Aplikasi

Gambar 3 di atas menjelaskan alur kerja dari aplikasi yang dibuat, dimana dalam prosesnya pengujian yang dilakukan merupakan pengujian white box yang meminta *user* untuk memasukan source code aplikasi yang hendak diuji, lalu mencari fungsi yang digunakan dalam aplikasi, diambil atribut dari *source code* yang berkenaan dengan pembangkitan input tag kedalam form, kemudian dihilangkan atribut yang tidak termasuk kedalam pengujian. Aplikasi akan mengambil kode sumber tersebut sebagai bahan mutasi untuk men-*generate* file tes. Untuk mutasinya digunakan 5 operator mutasi, terakhir melakukan kalkulasi perhitungan dengan rumus indikator skor mutasi.

Proses pembacaan kode sumber dilakukan dengan mengkategorikan string yang membangun tag HTML menjadi tag atribut dan nilai-atribut, menghilangkan perbedaan pola tag buka dan tag tutup, dan mengubah string tag kedalam huruf kecil. Berikut adalah operator mutasi yang akan diterapkan:

Tabel 1. Operator Mutasi

No	Operator Mutasi	Nama Operator	Variasi Mutan
1	HTML Element <input>	Operator HTML Element	<label>, <select>, <button>, <textarea>, <fieldset>
2	HTML Attribute Type	Operator HTML Attribute Type	text, password, checkbox, radio, file, submit, reset, button, number, date, email, url
3	HTML Attribute Name	Operator HTML Attribute Name	Kind of csrf name: csrf-token, token, xsrf-token
4	HTML Attribute Value	Operator HTML Value	$_COOKIE[“PHPSESSID”] Approximately 1000 random value
5	Existence HTML Input Element	Operator ExHTML Input Element	Exist/Unexist

Tabel 1 di atas menunjukan operator mutasi yang digunakan dalam penelitian ini, terdiri dari 5 yaitu *Operator HTML Element, Operator HTML Attribute Type, Operator HTML Attribute Name, Operator HTML Value, dan Operator ExHTML HTML Element*, adapun variasi mutasi yang dihasilkan dapat dilihat pada kolom variasi mutan.

2.3. Skor Mutan

Pengujian mutasi bertujuan untuk menghasilkan serangkaian kasus uji yang cukup untuk menemukan semua kemungkinan kesalahan dalam sebuah program. Proses ini melibatkan pengujian kasus uji terhadap berbagai variasi mutasi dari kode program tersebut. Kasus uji yang mampu membunuh semua mutan yang tidak setara dianggap memadai dengan skor yang tinggi. Skor mutan dihitung dengan melihat indikator skor mutasi berdasarkan hasil bagi jumlah total mutan yang berhasil dibunuh dengan jumlah mutan yang tidak setara. Skor mutasi adalah nilai nyata dari 0,0 hingga 1,0 yang dianggap sebagai skor tertinggi[15]. Untuk mencapai skor mutasi terbaik, himpunan

Vol. 17, No. 2, September 2024, P-ISSN 1978-9262, E-ISSN 2655-5018
DOI: https://doi.org/10.33322/petir.v17i2.2493

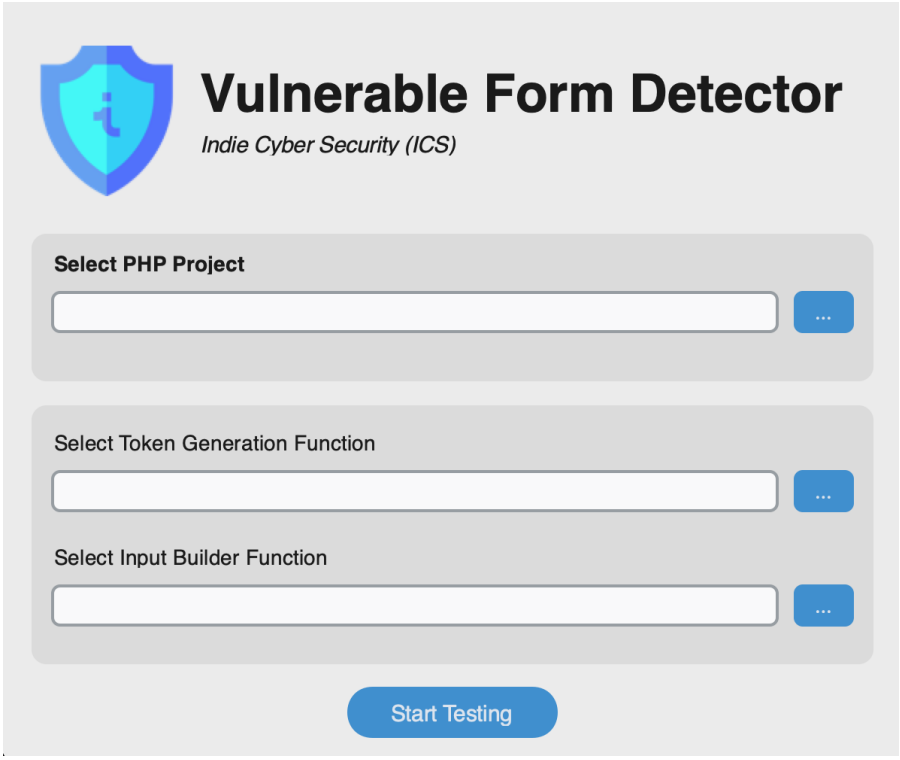
uji harus berhasil meng-*kill* semua mutan. Namun, beberapa mutan yang dihasilkan mungkin tidak berhasil di-kill dengan kode asli dan tidak mempengaruhi skor mutasi secara signifikan. Berikut adalah persamaan yang digunakan untuk menghitung skor mutan.

$$MSI = (TotalDefeatedMutants / TotalMutantsCount) * 100; \quad (1)$$

3. HASIL DAN PEMBAHASAN

3.1. Tampilan Aplikasi

Dari segi tampilan aplikasi dibuat sesederhana mungkin agar memudahkan *user* dalam memahami alurnya. Diawal *user* harus memasukan lokasi direktori aplikasi yang akan di test, kemudian *user* memilihkan yang berisi kelas dan fungsi untuk membangkitkan token dan kelas dan fungsi pembuat elemen input HTML, berikut gambar tampilannya.



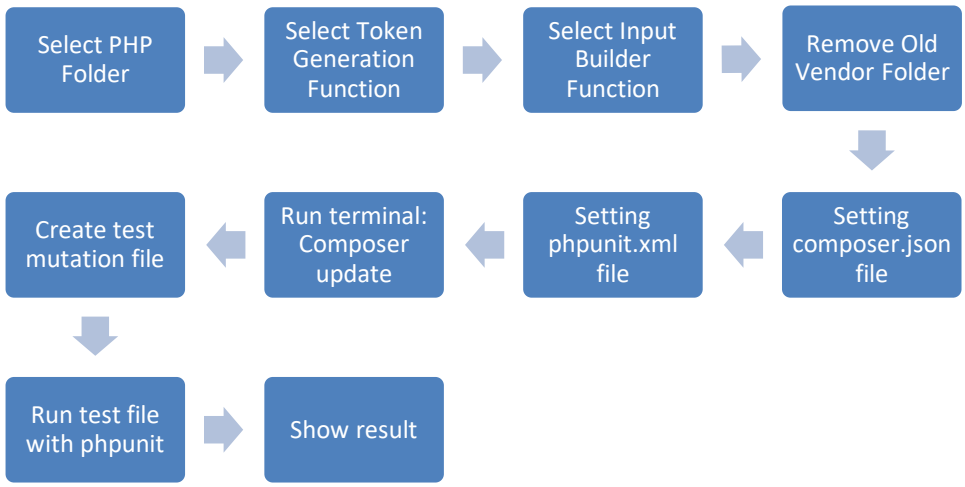
Gambar 4. Tampilan depan Aplikasi

Gambar 4 di atas menunjukan tampilan dari aplikasi yang dibuat dengan menerapkan pengujian mutasi berdasarkan operator mutasi yang ditentukan pada tabel 1 di atas, dengan menekan tombol "*Start Testing*" serangkaian proses akan dilakukan dari persiapan library phpunit, pembuatan file uji yang dibutuhkan, sampai mengeksekusi file uji yang telah dibuat serta menampilkan skor hasil pengujian nya. Dalam rancangan penelitian aplikasi yang dibangun akan segera menampilkan skor hasil uji setelah tombol ditekan, namun pada penerapannya diperlukan serangkaian proses yang kompleks dari awal persiapan sampai hasil akhir ditampilkan. Dimulai dari mengecek apakah ada folder penampung library yang biasa diberi nama "vendor". Apabila ada maka folder tersebut harus dihapus. Selanjutnya menyiapkan file "composer.json" sebagai file utama yang akan dibaca oleh paket manajer composer untuk menentukan library apa saja yang akan diunduk dan dimasukan kedalam folder yang baru. Bisa saja aplikasi dari *user* sudah memiliki daftar library yang dibutuhkan

Vol. 17, No. 2, September 2024, P-ISSN 1978-9262, E-ISSN 2655-5018
DOI: https://doi.org/10.33322/petir.v17i2.2493

dalam menjalankan aplikasinya, sehingga dalam tahap ini tidak mungkin menghapus daftar library tersebut. Maka yang dilakukan adalah menambah library phpunit kedalam daftar tersebut. File selanjutnya yang perlu disiapkan adalah *phpunit.xml*, file ini akan dibaca oleh library phpunit saat dijalankan. File ini mengandung sejumlah pengaturan dasar sehingga phpunit mengetahui seperti apa target pengerjaan tes yang diinginkan.

Di antara rangkaian persiapan yang juga harus dilakukan aplikasi adalah, menjalankan perintah pada terminal fungsi "*composer update*". Fungsi ini akan meminta composer sebagai paket manajer library php untuk mengunduh semua library yang telah terdaftar didalam file "composer.json". Dengan demikian library phpunit yang akan kita gunakan akan terunduh dan siap untuk digunakan. Berikut gambar diagram yang menjelaskan alur lengkap kinerja aplikasi.



Gambar 5. Alur Kinerja Aplikasi

Gambar 5 di atas menunjukan langkah pertama yang harus dilakukan *user d*alam menggunakan aplikasi ini adalah memiliki *folder source code* yang akan dianalisa kemudian *user* memilih *function* apa yang melakukan *token generation* dan memilih mana yang akan melakukan input. Selanjutnya sistem akan melakukan penyesuaian kepada sistem php unit dan melakukan *running* php unit secara otomatis tanpa harus mengakses dari terminal. Setelah semua mutasi dilakukan, maka akan menghasilkan skor MSI.

++Original

```
$hidden = "<!--\n--><input type=\"hidden\"" . " name=\"" . //next code
```

--Mutant

```
1
$hidden = "<!--\n--><label type=\"hidden\"" . " name=\"" . //next code
2
$hidden = "<!--\n--><select type=\"hidden\"" . " name=\"" . //next code
3
$hidden = "<!--\n--><button type=\"hidden\"" . " name=\"" . //next code
4
$hidden = "<!--\n--><textarea type=\"hidden\"" . " name=\"" . //next code
5
$hidden = "<!--\n--><fieldset type=\"hidden\"" . " name=\"" . //next code
```

Gambar 6. Contoh mutasi pada operator mutasi HTML Element

PETIR: Jurnal Pengkajian dan Penerapan Teknik Informatika
Vol. 17, No. 2, September 2024, P-ISSN 1978-9262, E-ISSN 2655-5018
DOI: https://doi.org/10.33322/petir.v17i2.2493

Gambar 6 di atas merupakan contoh mutasi Operator HTML Element. Terdapat 5 kemungkinan mutasi yang terjadi karena pada form peluang perubahan tag input dapat diubah ke dalam <label>, <select>, <button>, <textarea>, <fieldset>.

++Original

```
$hidden = "<!--\n--><input type=\"hidden\"" . " name=\"" . //next code
```

--Mutant

```
1
$hidden = "<!--\n--><input type=\"text\"" . " name=\"" . //next code
2
$hidden = "<!--\n--><input type=\"password\"" . " name=\"" . //next code
3
$hidden = "<!--\n--><input type=\"checkbox\"" . " name=\"" . //next code
4
```

Gambar 7. Contoh mutasi pada operator mutasi HTML Attribute

Gambar 7 di atas merupakan contoh mutasi Operator HTML Attribute, terdapat kemungkinan 12 mutasi yang terjadi karena pada attribute type peluang perubahan type hidden dapat diubah ke dalam *text, password, checkbox, radio, file, submit, reset, button, number, date, email,* url. Salah satu aplikasi yang kami uji adalah sebuah sistem parkir yang di dalamnya terdapat formulir input yang mengandung token sebagai pengaman dari CSRF. Berikut perbandingan kode sumber sebelum dimutasi dengan file tes yang telah di-*generate*:

```php
public function insertHiddenToken()
{
    $csrfToken = $this->getCSRFToken();
    $hidden = "<!--\n--><input type=\"hidden\"" . " name=\"" . $this->xssafe($this->formTokenLabel) .
        "\"" . " value=\"" . $this->xssafe($csrfToken) . "\"" . " />";
    return $hidden;
}

public function getCSRFToken()
{   $this->unsetToken();
    if (empty($this->session[$this->sessionTokenLabel])) {
        $this->session[$this->sessionTokenLabel] = bin2hex(random_bytes($this->tokenLen));
    }
    if ($this->hmac_ip !== false) {
        $token = $this->hMacWithIp($this->session[$this->sessionTokenLabel]);
    } else {
        $token = $this->session[$this->sessionTokenLabel];
    }
    return $token;
}
```

Gambar 8. Kode sumber sebelum dimutasi

Vol. 17, No. 2, September 2024, P-ISSN 1978-9262, E-ISSN 2655-5018
DOI: https://doi.org/10.33322/petir.v17i2.2493

```php
public function test_existance() {
    $this->construct();
    $this->assertSame(count($this->data) , 0);
}
public function test_val_sessid() {
    $this->construct();
    $token = $this->class1->getCSRFToken(); // getCSRFToken();
    $this->assertSame($token, $_COOKIE["PHPSESSID"]);
}
public function test_tag_0() { // func_name(use attribute), index
    $this->construct();
    $validator = 'label'; // value
    $tester = isset($this->data["tag"]) ? $this->data["tag"] : $validator;
    $this->assertSame($tester, $validator);
}
public function test_tag_1() { // func_name(use attribute), index
    $this->construct();
    $validator = 'select'; // value
    $tester = isset($this->data["tag"]) ? $this->data["tag"] : $validator;
    $this->assertSame($tester, $validator);
}
public function test_tag_2() { // func_name(use attribute), index
    $this->construct();
    $validator = 'button'; // value
    $tester = isset($this->data["tag"]) ? $this->data["tag"] : $validator;
    $this->assertSame($tester, $validator);
}
public function test_tag_3() { // func_name(use attribute), index
    $this->construct();
    $validator = 'textarea'; // value
    $tester = isset($this->data["tag"]) ? $this->data["tag"] : $validator;
    $this->assertSame($tester, $validator);
}
```

Gambar 9. Kode yang ada pada file tes setelah di-*genrate*

Gambar 8 di atas menunjukan menunjukkan implementasi metode untuk mengelola token CSRF (*Cross-Site Request Forgery*). Fungsi insertHiddenToken() bertugas untuk menghasilkan elemen HTML input tersembunyi yang menyisipkan token CSRF ke dalam formulir, menggunakan nilai token yang diperoleh dari metode getCSRFToken(). Sementara itu, fungsi getCSRFToken() bertugas menghasilkan token CSRF baru jika tidak ada token yang tersimpan dalam sesi, dengan memanfaatkan bin2hex() dan random_bytes() untuk membuat token unik dan aman. Fungsi ini juga mendukung HMAC (*Hash-based Message Authentication Code*). Kombinasi kedua metode ini bertujuan untuk melindungi aplikasi web dari serangan CSRF dengan menyematkan token unik di setiap permintaan formulir. Adapun pada gambar 9 meunujukan hasil mutasi yang dibangkitkan oleh aplikasi.

Hasil dari pengujiannya bernilai sempurna, dari total 1.022 mutan yang dihasilkan tidak ada satupun mutan yang tidak terhentikan atau tidak terbunuh sehingga skor akhirnya bernilai 1. Ini disebabkan karena *programmer* aplikasi sistem parkir tersebut sudah membuat tag input dengan benar serta menggunakan algoritma pembangkit token yang baik sehingga dalam 1.000 kali token tersebut dibangkitkan tidak ada satupun yang sama. Berikut gambar tampilan nilai skor pengujiannya.

Vol. 17, No. 2, September 2024, P-ISSN 1978-9262, E-ISSN 2655-5018
DOI: https://doi.org/10.33322/petir.v17i2.2493

Gambar 10. Hasil pengujian pada sistem parkir

Gambar 10 di atas menunjukkan hasil dari pengujian yang dilakukan, menunjukkan aplikasi yang dibuat dapat mendeteksi mutant yang dihasilkan. Berikut gambar di bawah ini menunjukkan contoh mutasi yang dihasilkan.

4. KESIMPULAN DAN SARAN

Serangkaian metode yang yang dilakukan oleh cyberPHP telah membantu pengguna dalam mendeteksi kerentanan web aplikasi yang dibuat. Pengklasifikasian string elemen input HTML yang dilakukan membantu meningkatkan kualitas mutan menjadi lebih baik sehingga pendeteksian kerentanan menjadi lebih akurat. Meski demikian variasi mutan perlu ditambah lagi guna meningkatkan kualitas uji, hasil skor 100 yang didapat pun karena variasi mutasi yang masih terbilang sedikit. Variasi waktu pembangkitan token pun menjadi salah satu tolak ukur kualitas uji yang dilakukan, semakin banyak variasi waktu yang digunakan semakin baik pula kualitas uji.

Selanjutnya, penelitian yang dapat diusulkan adalah menambahkan variasi waktu pembangkitan token dan variasi mutan untuk meningkatkan kualitas uji, serta menyediakan web aplikasi dengan minimal 3 kategori, yaitu kerentanan rendah, kerentanan sedang dan kerentanan tinggi.

UCAPAN TERIMAKASIH

Penulis mengucapkan terima kasih kepada STEI ITB yang telah memberi dukungan yang membantu pelaksanaan penelitian dan atau penulisan artikel.

DAFTAR PUSTAKA

[1] K. Al-Talak and O. Abbass, “Detecting Server-Side Request Forgery (SSRF) Attack by using Deep Learning Techniques.” [Online]. Available: www.ijacsa.thesai.org

[2] A. A. Saifan and M. B. Ata, “Mutation Testing for Evaluating PHP Web Applications,” International Journal of Software Innovation, vol. 7, no. 4, pp. 25–50, Oct. 2019, doi: 10.4018/IJSI.2019100102.

[3] cvedetails.com, “CVE Security Vulnerability Database.” Accessed: Jun. 02, 2024. [Online]. Available: https://www.cvedetails.com/vulnerabilities-by-types.php

[4] “Securing Cross-Site Request Forgery Vulnerabilities in Web Applications Using Mutation Analysis.”

Vol. 17, No. 2, September 2024, P-ISSN 1978-9262, E-ISSN 2655-5018
DOI: https://doi.org/10.33322/petir.v17i2.2493

[5] E. B. I, "An Enhanced Mechanism for Protecting Web Applications from Cross Site Request Forgery (CSRF)," British Journal of Computer, Networking and Information Technology, vol. 7, no. 1, pp. 1–17, 2024, doi: 10.52589/BJCNIT_R5YYKXKA.

[6] S. K. #1 and Prasath, "A conceptual study of Cross Site Request Forgery with comprehensive scrutiny", [Online]. Available: http://127.0.0.1/klog.js

[7] "DerScanner.WhitePaper".

[8] W. H. Rankothge and S. M. N. Randeniya, "Identification and Mitigation Tool for Cross-Site Request Forgery (CSRF)," in IEEE Region 10 Humanitarian Technology Conference, R10-HTC, Institute of Electrical and Electronics Engineers Inc., Dec. 2020. doi: 10.1109/R10-HTC49770.2020.9357029.

[9] Y. Huang et al., "A mutation approach of detecting SQL injection vulnerabilities," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10603 LNCS, pp. 175–188, 2017, doi: 10.1007/978-3-319-68542-7_15.

[10] D. Appelt, C. D. Nguyen, L. C. Briand, and N. Alshahwan, "Automated Testing for SQL Injection Vulnerabilities : An Input Mutation Approach," in International Symposium on Software Testing and Analysis, 2014, pp. 259–269.

[11] F. Siavashi, D. Truscan, and J. Vain, "Vulnerability assessment of web services with model-based mutation testing," Proceedings - 2018 IEEE 18th International Conference on Software Quality, Reliability, and Security, QRS 2018, pp. 301–312, 2018, doi: 10.1109/QRS.2018.00043.

[12] U. Praphamontripong and J. Offutt, "Applying mutation testing to web applications," in ICSTW 2010 - 3rd International Conference on Software Testing, Verification, and Validation Workshops, 2010, pp. 132–141. doi: 10.1109/ICSTW.2010.38.

[13] Abdurrasyid, B. Sitohang, Y. D. W. Asnar, and G. A. P. Saptawati, "Securing Cross-Site Request Forgery Vulnerabilities in Web Applications Using Mutation Analysis," in 2024 2nd International Conference on Software Engineering and Information Technology, ICoSEIT 2024, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 227–232. doi: 10.1109/ICoSEIT60086.2024.10497499.

[14] A. B. Sanchez, J. A. Parejo, S. Segura, A. Duran, and M. Papadakis, "Mutation Testing in Practice: Insights from Open-Source Software Developers," IEEE Transactions on Software Engineering, May 2024, doi: 10.1109/TSE.2024.3377378.

[15] S. Hamimoune and B. Falah, "Mutation testing techniques: A comparative study," in Proceedings - 2016 International Conference on Engineering and MIS, ICEMIS 2016, Institute of Electrical and Electronics Engineers Inc., Nov. 2016. doi: 10.1109/ICEMIS.2016.7745368.