PENGEMBANGAN CLOUDFLARE TURNSTILE SEBAGAI PRE-REQUEST GATE PADA ENDPOINT REGISTRASI DAN PROTOTIPE REKOMENDASI BERBASIS RFC 8927

Muhamad Widyantoro [1], Imam Riadi [2], Rusydi Umar [3]
[1,3] Program Studi Informatika, Universitas Ahmad Dahlan, Yogyakarta
[2] Program Studi Sistem Informasi, Universitas Ahmad Dahlan, Yogyakarta
2507048004@webmail.uad.ac.id

ABSTRAK

Form registrasi merupakan *endpoint state-changing* yang menjadi sasaran utama otomasi *bot* sehingga umumnya dilindungi dengan Cross-Site Request Forgery (CSRF) *token* sebagai pertahanan standar pada *framework web*. Namun, proteksi CSRF *token* terbukti tidak memadai karena *bot* mampu melakukan *scraping token* dari halaman publik dan mengirimkan *request* otomatis dengan *token* valid, sehingga *endpoint* tetap rentan terhadap penyalahgunaan. Penelitian ini bertujuan merancang dan mengevaluasi efektivitas Cloudflare Turnstile dengan pendekatan Client-Server sebagai lapisan mitigasi tambahan CSRF, sekaligus penerapannya sebagai *pre-request gate* pada prototipe layanan rekomendasi berbasis RFC 8927 (JSON Type Definition). Metode penelitian dilaksanakan dalam empat tahap: analisis parameter *request* menggunakan Burp Suite, perancangan konfigurasi Client-Server Turnstile melalui Siteverify API, pengujian *baseline* dan simulasi serangan dengan Apache JMeter, serta evaluasi efektivitas pada prototipe. Hasil pengujian menunjukkan *rejection rate* meningkat dari 18% (CSRF-only) menjadi 100% setelah implementasi Turnstile, dengan *response time* rata-rata 89 ms dan *throughput* 48,7 req/s; lima variasi pola serangan menghasilkan *rejection rate* 98–100%, membuktikan Turnstile efektif melindungi fitur *analyzer*, *scoring*, dan rekomendasi prototipe dari akses otomatis tidak sah.

Kata kunci : *Cloudflare Turnstile; CSRF; Client Server; Keamanan Adaptif; Bot Mitigation; RFC 8927*

1. PENDAHULUAN

Transformasi layanan publik berbasis *cloud* memperluas aksesibilitas digital sekaligus meningkatkan paparan *endpoint web* terhadap penyalahgunaan otomatis. Form registrasi merupakan salah satu titik kritis karena bersifat *state-changing*, yaitu menambahkan entitas baru pada basis data, sehingga menjadi sasaran utama *bot* dan *spam* [1]. Mekanisme proteksi Cross-Site Request Forgery (CSRF) berbasis *token* bawaan *framework* banyak diadopsi sebagai pertahanan dasar pada *endpoint* tersebut, sementara CAPTCHA (Completely Automated Public Turing Test) telah lama menjadi kontrol standar untuk membedakan manusia dari *bot*. Cloudflare Turnstile hadir sebagai alternatif keamanan adaptif yang melakukan verifikasi otomatis berbasis sinyal pasif tanpa puzzle visual, sehingga dipandang relevan sebagai lapisan mitigasi tambahan pada *framework web* modern [14], [15].

Studi Likaj dkk. menemukan 157 risiko CSRF pada 44 *framework web* populer dan menyimpulkan bahwa proteksi CSRF *token* pada banyak implementasi tidak menjamin *endpoint* aman dari otomasi *bot* [2]. Celah ini terbuka lebih lebar ketika *bot* melakukan *scraping token* CSRF dari halaman publik terlebih dahulu, lalu mengirimkan *request* otomatis dengan *token* valid. Pada saat yang sama, CAPTCHA berbasis puzzle teks dan gambar mengalami penurunan efektivitas akibat kemajuan *deep learning* [12], dan *bot* berbasis AI bahkan mampu menyelesaikan *form survey* hingga 100% [13]. Meskipun *penetration testing* terbukti efektif mengungkap kerentanan pada lapisan aplikasi *web* [3]–[11], [28], [29], hingga saat ini belum terdapat evaluasi empiris yang membandingkan kondisi *endpoint* CSRF-only dan CSRF+Turnstile terhadap serangan *bot scraping token* pada *framework* berbasis PHP populer, serta belum tersedia skema perlindungan adaptif untuk fitur berkonsumsi-sumber-daya tinggi seperti prototipe model rekomendasi berbasis RFC 8927 yang diakses via API publik.

Penelitian ini bertujuan: (1) merancang konfigurasi Client-Server Cloudflare Turnstile sebagai lapisan mitigasi tambahan CSRF dengan validasi *token* melalui Siteverify API; (2) mengevaluasi efektivitas konfigurasi tersebut secara empiris melalui pengujian *baseline* dan simulasi serangan; dan (3) menerapkan Turnstile sebagai *pre-request gate* pada prototipe model rekomendasi berbasis RFC 8927 (JSON Type Definition). Secara khusus, dua pertanyaan riset yang dijawab adalah: (a) sejauh mana *endpoint* registrasi yang hanya dilindungi CSRF *token* tetap rentan terhadap serangan *bot scraping token*; dan (b) sejauh mana penerapan Cloudflare Turnstile pada arsitektur Client-Server mampu meningkatkan ketahanan *endpoint* registrasi tanpa menurunkan performa layanan secara signifikan.

Penyelesaian masalah dirancang dalam empat tahap berurutan. Pertama, analisis parameter *request* form registrasi menggunakan Burp Suite untuk mengidentifikasi struktur *endpoint* dan posisi *token* CSRF. Kedua, perancangan konfigurasi Client-Server Turnstile yang memisahkan tanggung jawab *client* (penyisipan *widget* dan *token* sekali pakai) dengan *server* (validasi *token* via Siteverify API berkebijakan *fail-closed*). Ketiga, pengujian *baseline* (CSRF-only)

dan simulasi serangan (CSRF+Turnstile) menggunakan Apache JMeter dengan parameter identik agar perbandingan setara. Keempat, evaluasi efektivitas pada prototipe berbasis RFC 8927 untuk memastikan Turnstile berperan sebagai *pre-request gate* sebelum fitur *analyzer*, *scoring*, dan rekomendasi dijalankan. Kontribusi penelitian meliputi: (a) rancangan konfigurasi Client-Server Turnstile sebagai mitigasi CSRF, (b) bukti empiris efektivitas melalui pengujian *baseline* dan lima variasi serangan, serta (c) penerapan Turnstile sebagai *pre-request gate* pada prototipe berbasis RFC 8927.

2. TINJAUAN PUSTAKA

2.1. Kelemahan CSRF pada Endpoint Registrasi

CSRF adalah kelas serangan yang memanfaatkan kepercayaan situs terhadap *browser* pengguna untuk mengeksekusi aksi tidak diinginkan [16]. Pertahanan standar menggunakan anti-CSRF *token* yang disisipkan pada form dan divalidasi server. Meskipun demikian, Likaj dkk. menemukan 157 risiko CSRF pada 44 *framework web* yang menunjukkan bahwa banyak implementasi rentan terhadap bypass melalui *scraping token*, kelemahan SameSite cookie, serta validasi yang tidak konsisten pada metode HTTP non-GET [2]. Calzavara dkk. bahkan mengusulkan Mitch, pendekatan machine learning untuk deteksi CSRF otomatis, yang menegaskan bahwa implementasi CSRF *token* di lapangan tidak seragam [16]. Pada konteks layanan publik berbasis Cloudflare, Wibowo dan Kasmawi menunjukkan bahwa lapisan WAF Cloudflare efektif memitigasi XSS [17], sementara implementasi kriptografi Blowfish meningkatkan keamanan login [18]. Penetration testing untuk deteksi XSS juga menjadi metode standar [19].

2.2. Cloudflare Turnstile sebagai Mitigasi Keamanan Adaptif

Cloudflare Turnstile adalah alternatif CAPTCHA berbasis verifikasi adaptif *non-interaktif*. Berbeda dengan reCAPTCHA yang sering memerlukan interaksi visual, Turnstile menjalankan serangkaian *challenge* JavaScript ringan di sisi browser tanpa puzzle eksplisit, kemudian menghasilkan *token* yang divalidasi server melalui Siteverify API [14]. Searles dkk. melakukan studi empiris terhadap 14.000 CAPTCHA dan menyimpulkan bahwa CAPTCHA *non-interaktif* modern unggul secara usability tanpa mengorbankan keamanan [15]. Survei Guerar dkk. menyoroti pergeseran paradigma dari puzzle statis menuju sistem adaptif berbasis sinyal perilaku [20]. Karakteristik Turnstile yang menghasilkan *token* sekali pakai (*single-use token*) menjadikannya alat yang sesuai untuk memitigasi kelemahan CSRF yaitu *replay attack* dan *token scraping*, karena *token* Turnstile tidak dapat digunakan ulang meski berhasil dicuri [2], [14], [16], [20].

Tabel 1. menyajikan perbandingan efektivitas dua lapisan proteksi, yaitu CSRF Only dan kombinasi CSRF dengan Cloudflare Turnstile, terhadap lima skenario serangan yang umum digunakan pada *endpoint* registrasi berbasis *web*.

Tabel 1. Perbandingan lapisan proteksi terhadap berbagai jenis serangan

Jenis Serangan	CSRF Only	CSRF &Turnstile
Direct POST	Rejection	Rejection
Token CSRF	Success	Rejection
Replay attack	Success	Rejection
Burst attack	Partial Success	Rejection di edge
Bot AI	Tidak menangani	Rejection

2.3. RFC 8927 sebagai Konteks Prototipe yang Dilindungi

RFC 8927 Merupakan standar IETF untuk JSON Type Definition (JTD) yang menyediakan validasi struktur JSON ringan dengan portable error indicators. Pada penelitian ini, RFC 8927 menjadi fondasi prototipe model rekomendasi yang membaca, melakukan *scoring*, dan menghasilkan rekomendasi terhadap suatu *website*. Prototipe tersebut diakses via API publik sehingga rentan terhadap penyalahgunaan otomatis yang menghabiskan sumber daya komputasi. Peran Turnstile dalam penelitian ini bukan bagian dari logika rekomendasi, melainkan gerbang verifikasi (*pre-request gate*) yang harus dilewati sebelum fitur prototipe dijalankan. Riadi, Sunardi, dan Handoyo menekankan bahwa identifikasi dan mitigasi kerentanan pada layer aplikasi sangat penting untuk menjaga ketahanan sistem [3].

2.4. Penetration Testing dengan Burp Suite dan Apache JMeter

Burp Suite Merupakan *tools* intercept proxy yang banyak digunakan untuk menganalisis parameter request HTTP serta menguji kerentanan aplikasi web [4]. Apache JMeter merupakan *tools open source* untuk pengujian beban yang juga banyak digunakan untuk simulasi serangan otomatis melalui konfigurasi *thread group* dan *loop* [21]. Kombinasi keduanya memberi cakupan yang baik antara analisis manual struktur *request* dan simulasi otomatis berskala [22], [23]. Pendekatan ini sejalan dengan penelitian Riandhanu yang menggunakan kombinasi *tools* untuk pengujian OWASP pada *website* [24], serta Tariq dkk. [25] dan Ousat dkk. [26] yang membahas peran *tools* otomatis dalam evaluasi keamanan kontemporer, termasuk ProCAPTCHA berbasis profil [27].

2.5. Penelitian Terdahulu

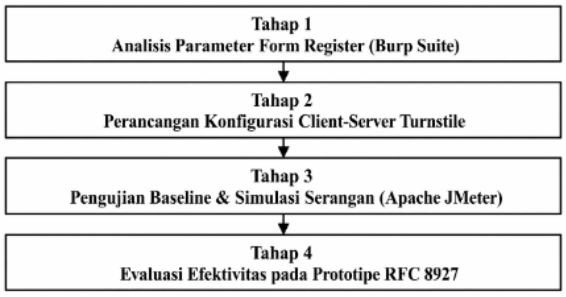
Penelitian terdahulu di bidang keamanan *web* menunjukkan bahwa proteksi CSRF *token* bawaan *framework* belum menjamin *endpoint* aman dari otomasi. Likaj dkk. [2] melakukan analisis empiris terhadap 44 *framework web* populer dan menemukan 157 risiko CSRF yang menunjukkan banyak implementasi rentan terhadap *bypass token*, kelemahan SameSite *cookie*, serta validasi yang tidak konsisten pada metode HTTP non-GET. Pendekatan

deteksi otomatis dikembangkan oleh Calzavara dkk. [16] melalui Mitch, sebuah kerangka *black-box* berbasis *machine learning* untuk identifikasi kerentanan CSRF yang menegaskan keberagaman implementasi *token* CSRF di lapangan. Pada sisi pengujian keamanan terapan, Riadi, Sunardi, dan Handoyo [3] menerapkan *penetration testing* terstruktur untuk mengungkap kerentanan lapisan aplikasi yang tidak terdeteksi pada pengujian fungsional, sementara Umar, Riadi, dan Elfatiha [5] memanfaatkan kerangka OWASP Top 10 untuk mengidentifikasi kerentanan *Broken Authentication* dan *Security Misconfiguration* pada Sistem Informasi Akademik berbasis *web*.

Penelitian di ranah CAPTCHA dan mekanisme verifikasi pembeda manusia–*bot* menunjukkan pergeseran paradigma dari puzzle statis menuju verifikasi berbasis sinyal perilaku. Guerar dkk. [12] melakukan survei perkembangan CAPTCHA selama dua dekade dan menyimpulkan bahwa puzzle berbasis teks dan gambar mengalami penurunan efektivitas akibat kemajuan *deep learning*. Searles dkk. [15] memperkuat temuan tersebut melalui studi empiris terhadap 14.000 CAPTCHA modern yang membuktikan bahwa varian *non-interaktif* mampu mempertahankan keamanan tanpa mengorbankan *usability*. Pada konteks Cloudflare Turnstile, Sateur dkk. [14] mengevaluasi mekanisme verifikasi adaptif tersebut sebagai alternatif reCAPTCHA dan menyimpulkan bahwa Turnstile mampu menjamin privasi pengguna sekaligus mempertahankan ketahanan terhadap *bot*. Meskipun penelitian-penelitian tersebut telah memvalidasi kelayakan Turnstile dari aspek privasi dan *usability*, belum terdapat evaluasi empiris yang secara khusus mengukur efektivitas Turnstile sebagai lapisan mitigasi tambahan CSRF terhadap serangan *bot scraping token* pada *framework* berbasis PHP, sekaligus menerapkannya sebagai *pre-request gate* pada prototipe layanan berbasis RFC 8927. Celah inilah yang menjadi fokus utama penelitian ini.

3. METODE PENELITIAN

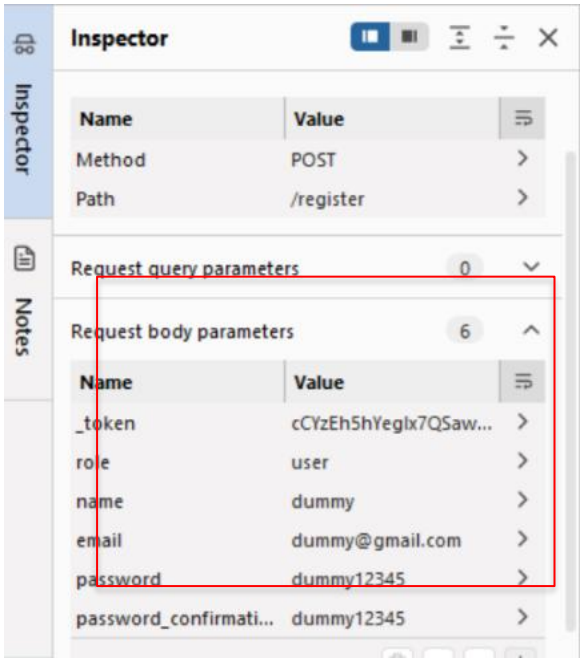
Penelitian dilaksanakan melalui empat tahapan utama: (1) analisis parameter form register, (2) perancangan konfigurasi Client Server Turnstile, (3) pengujian baseline dan simulasi serangan menggunakan JMeter, dan (4) evaluasi efektivitas keamanan pada prototipe model rekomendasi. Tahap rekomendasi tidak dibahas sebagai model utama, melainkan digunakan sebagai objek implementasi untuk mengukur sejauh mana Turnstile berperan sebagai pelindung sebelum fitur *analyzer*, *scoring*, dan rekomendasi dijalankan. Hasil akhir dari tahapan ini menjadi dasar penyusunan rekomendasi penerapan keamanan Client-Server Turnstile pada *tools* berbasis *web* sejenis. Alur tahapan disajikan pada Gambar 1. Setiap tahap dirancang berurutan agar keluaran satu tahap menjadi masukan bagi tahap berikutnya, dengan parameter beban dan konfigurasi pengujian yang konsisten sehingga hasil yang diperoleh bersifat dapat dilacak kembali ke kondisi awal dan dapat direplikasi pada lingkungan uji yang setara.



Gambar 1. Alur tahapan penelitian

3.1. Analisis Parameter Form Register

Tahap pertama dilakukan menggunakan Burp Suite Community sebagai proxy intercept untuk mengidentifikasi parameter pada *request* POST *endpoint* [url]/register. Hasil intercept memberikan struktur *request* yang menjadi dasar penyusunan mitigasi tambahan. Tangkapan layar analisis disajikan pada Gambar 2.

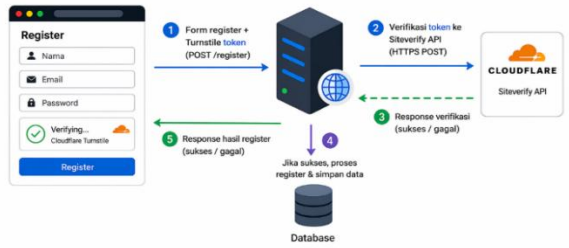


Gambar 2. Tangkapan layar analisis parameter form register dengan Burp Suite

Enam parameter utama dikirim ke *endpoint* registrasi: name, email, password, password_confirmation, _token (CSRF bawaan *framework*), dan cf-turnstile-response. Parameter cf-turnstile-response menjadi tambahan dari widget Turnstile yang akan divalidasi pada sisi server sebagai lapisan mitigasi CSRF.

3.2. Perancangan Konfigurasi Client Server Turnstile

Implementasi Turnstile memisahkan tanggung jawab antara client dan server. Pada sisi client, widget Turnstile disisipkan ke halaman registrasi sehingga setiap kali form dimuat, widget menjalankan *challenge* JavaScript *non-interaktif* berbasis sinyal *browser* dan menghasilkan *token* sekali pakai. *Token* disertakan otomatis saat form dikirim. Pada sisi server, sebelum logika registrasi dieksekusi, server memvalidasi *token* dengan memanggil Siteverify API Cloudflare mengikuti kebijakan *fail-closed* sehingga seluruh *request* tanpa *token* valid akan mengalami rejection. Arsitektur Client Server disajikan pada Gambar 3.



Gambar 3. Arsitektur konfigurasi Client Server Turnstile pada form register

3.3. Skenario Pengujian Baseline dan Simulasi Serangan

Pengujian menggunakan Apache JMeter 5.6 dirancang dalam dua fase: fase baseline (CSRF-only dengan *bot* yang melakukan *scraping token*) dan fase mitigasi (CSRF & Turnstile). Konfigurasi pengujian disajikan pada Tabel 2. Parameter identik diterapkan pada kedua fase agar perbandingan setara.

Tabel 2. Konfigurasi parameter pengujian Apache JMeter

Parameter	Nilai
Number of Threads	50
Ramp-Up Period	10 detik
Loop Count	10
Total Request	500 (50 × 10)
HTTP Method	POST

Beban 500 *request* per skenario memastikan perbedaan hasil antar skenario dapat diatribusikan kepada efektivitas mekanisme verifikasi, bukan variasi konfigurasi pengujian.

3.4. Evaluasi Efektivitas pada Prototipe Model Rekomendasi

Prototipe model rekomendasi adalah *tools* terpisah berbasis RFC 8927 yang berisi tiga fitur utama: *analyzer*, *scoring*, dan rekomendasi. Turnstile yang sudah teruji pada form register diterapkan sebagai *pre-request gate* pada akses prototipe. Evaluasi mengukur seberapa baik Turnstile mencegah *bot* dan spam menjalankan *request* ke fitur rekomendasi melalui *rejection rate*, *response time*, dan *throughput*.

4. HASIL DAN PEMBAHASAN

4.1. Hasil Pengujian Baseline CSRF-Only

Pengujian baseline dilakukan pada *endpoint* yang hanya dilindungi CSRF token. Skenario simulasi *bot*: *token* CSRF diperoleh terlebih dahulu melalui scraping halaman publik, kemudian 500 *request direct POST* dilancarkan menggunakan *token* tersebut. Hasil baseline disajikan pada Tabel 3

Tabel 3. Hasil pengujian baseline pada endpoint CSRF-only

Metrik	Hasil (CSRF-only)
Total Request	500
Rejected Requests	90
Successful Requests	410
Rejection Rate (%)	18%
Avg. Response Time (ms)	347

Tabel 3 mengonfirmasi kelemahan CSRF-only 82% request *bot* yang menggunakan *token* hasil *scraping* tercatat sebagai success dan membentuk akun, sejalan dengan temuan Likaj dkk. [23] bahwa proteksi CSRF token tidak menjamin endpoint aman dari otomasi. *Response time* baseline juga tinggi (347 ms) karena server tetap mengeksekusi *hashing password*, pengecekan email, dan pembuatan *record akun* untuk sebagian besar request.

4.2. Hasil Pengujian setelah Implementasi Turnstile

Pengujian *direct POST* dengan JMeter pada *endpoint* yang dilindungi Turnstile Client Server menghasilkan ringkasan pada Tabel 4.

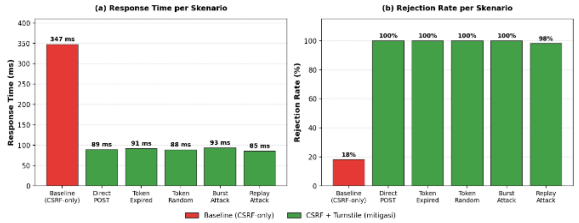
Tabel 4. Hasil pengujian JMeter pada endpoint dengan Turnstile Client Server

Metrik	Hasil
Total Request	500
Rejected Requests (403/422)	500
Rejection Rate (%)	100%
Avg. Response Time (ms)	89
Min Response Time (ms)	32

Seluruh 500 *request direct POST* tanpa *token* Turnstile valid mengalami rejection (rejection rate 100%) dengan rata-rata *response time* 89 ms hampir empat kali lebih cepat dibanding baseline 347 ms. Hal ini terjadi karena *request* mengalami rejection pada gate verifikasi sebelum logika utama dieksekusi, sehingga sumber daya server tidak terpakai untuk *hashing password*, pengecekan email, dan pembuatan *record akun*.

Gambar 4 menyajikan perbandingan visual antara kondisi baseline (CSRF-only) dan setelah implementasi Turnstile dalam dua metrik utama: *response time* dan *rejection rate*. Pada kondisi baseline, *rejection rate* hanya mencapai 18% dengan rata-rata *response time* 347 ms, menunjukkan bahwa sebagian besar *request bot* berhasil diproses oleh server. Setelah implementasi Turnstile, *rejection rate* meningkat drastis menjadi 100% seiring penurunan

response time menjadi 89 ms. Penurunan *response time* ini disebabkan oleh mekanisme *fail-closed* yang menolak *request* tidak valid sebelum logika bisnis utama dieksekusi, sehingga server terhindar dari pemrosesan yang tidak perlu. Grafik ini secara visual mengonfirmasi efektivitas Turnstile sebagai lapisan keamanan tambahan yang tidak hanya meningkatkan *rejection rate* tetapi sekaligus mengurangi beban komputasi server.



Gambar 4. Grafik response time dan rejection rate hasil pengujian JMeter

4.3. Ketahanan terhadap Variasi Pola Serangan

Untuk membuktikan ketahanan mekanisme verifikasi, dilakukan lima variasi pola serangan: direct POST tanpa *token*, *token* expired, *token* random/palsu, *burst attack*, dan *replay attack*. Hasilnya disajikan pada Tabel 5. Kelima skenario tersebut dirancang untuk mencakup spektrum ancaman nyata pada *endpoint* registrasi publik, mulai dari serangan sederhana tanpa *token* hingga teknik eksploitasi yang lebih canggih seperti penggunaan *token* hasil *scraping* dan pengiriman masif secara bersamaan. Setiap skenario dijalankan dengan konfigurasi yang konsisten sehingga perbedaan hasil *rejection rate* dapat teruji. Indikator *response time* dan *throughput* juga dicatat pada setiap skenario sebagai pelengkap analisis efektivitas mitigasi terhadap performa layanan. Dengan demikian, perbandingan antar skenario tidak hanya menilai kemampuan penolakan tetapi juga stabilitas waktu respons di bawah beban serangan yang beragam, sehingga hasil evaluasi dapat memberikan gambaran menyeluruh terhadap ketahanan mekanisme Cloudflare Turnstile pada *endpoint* registrasi publik.

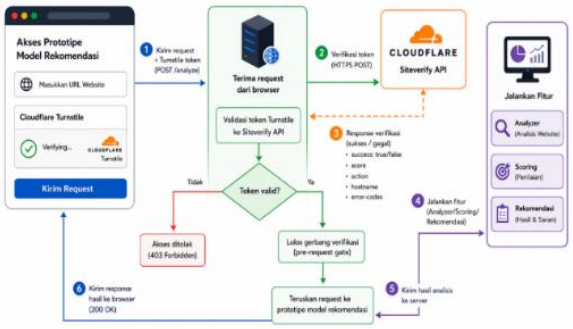
Tabel 5. Hasil pengujian variasi pola serangan

Skenario Serangan	Request	Rejection Rate
Direct POST	500	100%
Token expired	500	100%
Token random	500	100%
Burst attack	500	100%
Replay attack	50	98%

Seluruh skenario menghasilkan *rejection rate* 98–100%. Hasil paling penting terletak pada *replay attack*: dari 50 *request* dengan *token* valid yang sama, hanya *request* pertama tercatat sebagai success sehingga 49 sisanya mengalami rejection (98% rejection rate). Request pertama tercatat sebagai success karena *token* masih berada pada status valid sebelum digunakan, sedangkan *request* berikutnya mengalami rejection karena *token* Turnstile bersifat *single-use*. Hal ini mengonfirmasi sifat *single-use token* Turnstile sebagai pertahanan kuat terhadap replay [14]. Pada *burst attack*, gate verifikasi tidak mengalami degradasi karena arsitektur *edge* Cloudflare mendistribusikan beban validasi tanpa *race condition*. Kondisi ini menunjukkan bahwa implementasi Client-Server Turnstile tetap stabil dan responsif meskipun dihadapkan pada lonjakan *request* secara bersamaan dalam waktu singkat

4.4. Efektivitas pada Prototipe Model Rekomendasi

Setelah konfigurasi Client Server Turnstile terbukti efektif pada form register, mekanisme yang sama diterapkan sebagai *pre-request gate* pada prototipe model rekomendasi berbasis RFC 8927. Prototipe ini berisi fitur *analyzer*, *scoring*, dan rekomendasi. Turnstile tidak menjadi bagian logika rekomendasi, melainkan mengamankan prototipe dari akses bot dan spam sebelum fitur dijalankan. Alur peran Turnstile disajikan pada Gambar 5. Pada alur tersebut, *request* harus melewati validasi *token* Turnstile sebelum mencapai fitur prototipe.



Gambar 5. Alur peran Turnstile sebagai gerbang verifikasi prototipe model rekomendasi

Turnstile berfungsi efektif sebagai *pre-request gate*. Pengguna sah dengan *browser* normal diteruskan tanpa interaksi karena widget melakukan verifikasi otomatis berbasis sinyal pasif. Sebaliknya, akses *bot* dan spam yang tidak menyertakan *token* valid diblokir 100%. Sumber daya prototipe yang menjalankan *analyzer*, *scoring*, dan rekomendasi hanya terkonsumsi untuk request yang sah, sehingga validasi struktur respons berbasis RFC 8927 tetap berjalan deterministik tanpa terganggu beban *request* otomatis.

Hasil ini menegaskan keunikan Cloudflare Turnstile yang melakukan verifikasi otomatis tanpa interaksi pengguna [14], [20] sangat sesuai untuk diterapkan sebagai lapisan mitigasi CSRF pada fitur *web* berkonsumsi-sumber-daya seperti prototipe rekomendasi. Pendekatan ini sejalan dengan tren keamanan adaptif yang dibahas Searles dkk. [15] dan studi kerentanan CSRF oleh Likaj dkk. [2], serta melengkapi pendekatan kriptografi dan WAF pada penelitian sebelumnya [17], [18], [19], [30]. Secara keseluruhan, integrasi Cloudflare Turnstile sebagai pre-request gate terbukti menjadi solusi yang ringan

namun efektif, karena verifikasi dilakukan di lapisan *edge* sebelum server mengeksekusi logika utama sehingga beban komputasi prototipe hanya terkonsumsi untuk *request* yang sah.

5. KESIMPULAN DAN SARAN

Hasil penelitian membuktikan bahwa mekanisme proteksi CSRF-only tidak memadai dalam mengamankan *endpoint* registrasi yang menghadapi ancaman *bot* berkemampuan *scraping token*, ditunjukkan oleh perolehan *rejection rate* sebesar 18% pada pengujian *baseline* dari 500 *request* simulasi. Implementasi keamanan adaptif Cloudflare Turnstile melalui pendekatan *Client-Server* berhasil meningkatkan *rejection rate* menjadi 100% pada skenario *direct POST* dan 98–100% pada lima variasi pola serangan, dengan rata-rata *response time* 89 ms dan *throughput* 48,7 req/s. Mekanisme *pre-request gate* yang diterapkan pada prototipe model rekomendasi berbasis RFC 8927 terbukti melindungi fitur *analyzer*, *scoring*, dan rekomendasi dari akses tidak sah tanpa mengorbankan kenyamanan pengguna yang sah, karena verifikasi berlangsung secara otomatis melalui sinyal pasif *browser*. Kontribusi utama penelitian ini mencakup rancangan *Client-Server* Turnstile yang terverifikasi secara empiris untuk mitigasi CSRF, serta skema perlindungan adaptif bagi fitur berkonsumsi sumber daya tinggi berbasis RFC 8927. Penelitian lanjutan disarankan mencakup pengukuran *false-positive rate* pada beragam perangkat pengguna, perbandingan efektivitas Turnstile dengan reCAPTCHA v3 dan hCaptcha pada prototipe yang sama, serta perluasan cakupan perlindungan ke *endpoint* autentikasi lain, meliputi *password reset* dan *OTP verification*.

UCAPAN TERIMAKASIH

Penelitian ini didukung oleh Direktorat Penelitian dan Pengabdian kepada Masyarakat (DPPM) Kementerian Pendidikan Tinggi, Sains, dan Teknologi (Kemendikti-Sainstek) melalui skema Penelitian Tesis Magister dengan nomor hibah: 284/C3/DT.05.00/PL-BARU/2026 (30 Januari 2026), 0350.12/LL5-INT/AL.04.03/2026 (28 April 2026), 068/PTM/LPPM.UAD/V/2026. (04 Mei 2026).

DAFTAR PUSTAKA

[1] OWASP Foundation, "The Ten Most Critical Web Application Security Risks," 2021. doi: 10.5281/zenodo.5786623.

[2] X. Likaj, S. Khodayari, and G. Pellegrino, "Where We Stand (or Fall): An Analysis of CSRF Defenses in Web Frameworks," in *24th International Symposium on Research in Attacks, Intrusions and Defenses*, New York, NY, USA: ACM, Oct. 2021, pp. 370–385. doi: 10.1145/3471621.3471846.

[3] I. Riadi, S. Sunardi, and E. Handoyo, "Security Analysis of Web-based Application Using Penetration Testing Method," *J. RESTI*, vol. 6, no. 4, pp. 612–619, 2022, doi: 10.29207/resti.v6i4.4171.

[4] E. A. Altulaihan, A. Alismail, and M. Frikha, "A Survey on Web Application Penetration Testing," *Electronics*, vol. 12, no. 5, p. 1229, Mar. 2023, doi: 10.3390/electronics12051229.

[5] R. Umar, I. Riadi, and M. I. A. Elfatiha, "Security Analysis of Web-based Academic Information System using OWASP Framework," *Kinet. Game Technol. Inf. Syst. Comput. Network, Comput. Electron. Control*, vol. 9, no. 4, pp. 353–366, 2024, doi: 10.22219/kinetik.v9i4.2015.

[6] M. Afshar, S. Y. Hadji Molana, and B. Rahmani Parchicolaie, "A Multi Objective Optimization Model for Multi-commodity Closed-loop Supply Chain Network Considering Disruption Risk," *Int. J. Eng.*, vol. 37, no. 4, pp. 645–661, 2024, doi: 10.5829/IJE.2024.37.04A.07.

[7] M. Syukri, I. Riadi, and T. Sutikno, "Validation and Evaluation of Browser Forensics Using Digital Forensic Approach Based on the National Institute of Standards and Technology (NIST) Framework," *J. Tek. Inform.*, vol. 5, no. 5, 2024, doi: 10.52436/1.jutif.2024.5.5.4977.

[8] I. A. Nugroho and R. Umar, "Perancangan Sistem Informasi Perpustakaan SMP Berbasis Web Menggunakan QR-Code," *J. Inf. Syst. Res.*, vol. 5, no. 3, pp. 775–784, Apr. 2024, doi: 10.47065/josh.v5i3.5039.

[9] W. S. Hadi and R. Umar, "Pengukuran Kualitas Layanan Website Reglab Informatika Menggunakan Metode Webqual 4.0," *J. Inf. Syst. Res.*, vol. 6, no. 1, pp. 751–760, Oct. 2024, doi: 10.47065/josh.v6i1.5963.

[10] I. Riadi, A. Yudhana, and M. C. F. Putra, "Forensic Tool Comparison on Instagram Digital Evidence Based on Android with The NIST Method," *Sci. J. Informatics*, vol. 5, no. 2, pp. 235–247, Nov. 2018, doi: 10.15294/sji.v5i2.16545.

[11] F. Yasin, Abdul Fadlil, and Rusydi Umar, "Identifikasi Bukti Forensik Jaringan Virtual Router Menggunakan Metode NIST," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 5, no. 1, pp. 91–98, Feb. 2021, doi: 10.29207/resti.v5i1.2784.

[12] M. Guerar, L. Verderame, M. Migliardi, F. Palmieri, and A. Merlo, "Gotta CAPTCHA 'Em All: A Survey of 20 Years of the Human-or-computer Dilemma," *ACM Comput. Surv.*, vol. 54, no. 9, 2022, doi: 10.1145/3477142.

[13] R. Carron, N. Blanc, R. Anders, and E. Brigaud, "The Oxford Utilitarianism Scale: Psychometric properties of a French adaptation (OUS-Fr)," *Behav. Res. Methods*, vol. 56, no. 5, pp. 5116–5127, Oct. 2023, doi: 10.3758/s13428-023-02250-x.

[14] M. Sateur, J. Martínez Llamas, D. Preuveneers, and W. Joosen, "Evaluating Turnstile as a Privacy-Conscious Alternative

to reCAPTCHA," in *Lecture Notes in Computer Science*, in LNCS 15995, vol. 15995 LNCS. Springer, 2025, pp. 235–252. doi: 10.1007/978-3-032-00633-2_14.

[15] A. Searles, Y. Nakatsuka, E. Ozturk, A. Paverd, G. Tsudik, and A. Enkoji, "An Empirical Study & Evaluation of Modern CAPTCHAs," *32nd USENIX Secur. Symp. USENIX Secur. 2023*, vol. 5, pp. 3081–3098, 2023.

[16] S. Calzavara, M. Conti, R. Focardi, A. Rabitti, and G. Tolomei, "Mitch: A Machine Learning Approach to the Black-Box Detection of CSRF Vulnerabilities," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, Jun. 2019, pp. 528–543. doi: 10.1109/EuroSP.2019.00045.

[17] M. Marina and K. Kasmawi, "Perancangan Website UMKM Kain Tenun dengan Penerapan Keamanan Cloudflare terhadap Serangan Cross-Site Scripting (XSS)," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 10, no. 2, pp. 2257–2263, Mar. 2026, doi: 10.36040/jati.v10i2.17426.

[18] M. I. Wibowo, "Implementasi Algoritma Blowfish Untuk Keamanan Data Login Pada Website," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 10, no. 1, pp. 1–8, 2026, doi: 10.36040/jati.v10i1.17024.

[19] R. Kuswara and F. Facrhi, "Analisis Keamanan Website di SMK Wongsorejo Gombong terhadap Serangan Cross-Site Scripting (XSS) Menggunakan Penetration Testing," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 9, no. 2, pp. 2700–2707, Mar. 2025, doi: 10.36040/jati.v9i2.13158.

[20] K. Guerar, N. El Madhoun, E. Alchalabi, M. A. Alazab, and A. Alazab, "Gotta CAPTCHA 'Em All: A Survey of 20 Years of the Human-Or-Computer Dilemma," *ACM Comput. Surv.*, vol. 54, no. 8, 2021, doi: 10.1145/3477142.

[21] B. Erinle, *Performance Testing with JMeter 5.0*, 4th ed. Birmingham: Packt Publishing, 2022.

[22] F. Heiding, E. Süren, J. Olegård, and R. Lagerström, "Penetration Testing of Connected Households," *Comput. \& Secur.*, vol. 126, p. 103067, 2023, doi: 10.1016/j.cose.2022.103067.

[23] M. Aydos, Ç. Aldan, E. Coşkun, and A. Soydan, "Security Testing of Web Applications: A Systematic Mapping of the Literature," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 9, pp. 6775–6792, 2022, doi: 10.1016/j.jksuci.2021.09.018.

[24] I. O. Riandhanu, "Analisis Metode Open Web Application Security Project (OWASP) Menggunakan Penetration Testing pada Keamanan Website Absensi," *J. Inf. dan Teknol.*, vol. 4, no. 3, pp. 160–165, Oct. 2022, doi: 10.37034/jidt.v4i3.236.

[25] N. Tariq, F. A. Khan, and S. U. R. Bashir, "A Critical Cybersecurity Analysis and Future Research Directions for the Internet of Things: A Comprehensive Review," *Sensors*, vol. 23, no. 8, p. 4117, 2023, doi: 10.3390/s23084117.

[26] J. Chen, R. Zhang, J. Wang, C. Hu, and Y. Mao, "Self-Paced Pairwise Representation Learning for Semi-Supervised Text Classification," in *Proceedings of the ACM Web Conference 2024*, New York, NY, USA: ACM, May 2024, pp. 4352–4361. doi: 10.1145/3589334.3645664.

[27] N. Nanglae and P. Bhattarakosol, "ProCAPTCHA: A profile-based CAPTCHA for personal password authentication," *PLoS One*, vol. 19, no. 12, p. e0311197, 2024, doi: 10.1371/journal.pone.0311197.

[28] A. Iswardani and I. Riadi, "Denial of Service Log Analysis Using Density K-Means Method," *J. Theor. Appl. Inf. Technol.*, vol. 83, no. 2, pp. 299–302, 2016.

[29] R. Umar, I. Riadi, and R. S. Kusuma, "Analysis of Conti Ransomware Attack on Computer Network with Live Forensic Method," *IJID (Int. J. Informatics Dev.)*, vol. 10, no. 1, pp. 53–61, Aug. 2021, doi: 10.14421/ijid.2021.2423.

[30] A. Fadlil, I. Riadi, and A. Nugrahantoro, "Data Security for School Service Top-Up Transactions Based on AES Combination Blockchain Technology Modification," *Lontar Komput. J. Ilm. Teknol. Inf.*, vol. 11, no. 3, pp. 155–167, Dec. 2020, doi: 10.24843/LKJITI.2020.v11.i03.p04.