1. Pendahuluan

Latar Belakang

Secara delapan tahun berturut-turut *javascript* merupakan bahasa pemrograman yang paling populer dengan menempati peringkat 1 [1]. Tidak dapat dipungkiri penggunaan *javascript* yang *dapat* digunakan dimana saja menjadi salah satu faktornya. Sebelumnya *javascript* hanya digunakan sebagai *client-side*. Saat ini *javascript dapat* ditempatkan disisi server, namun tidak lepas dari *javascript* yang digunakan untuk mengatur *behaviour* suatu *website*, *javascript* masih menjadi pilihan pertama untuk urusan client-side. Karena adanya kelalaian *developer* pada kode program tidak melepas kemungkinan adanya celah keamanan di aplikasi *website*.

XSS (Cross Site Scripting) adalah salah satu bentuk serangan berupa injeksi yang dimasukkan kedalam suatu *website* yang digunakan untuk kode berbahaya kepada pengguna [2]. *XSS* biasa terjadi akibat dari kelalaian *developer* di kode program *javascript*. Pada september 2020, instagram memberikan bayaran USD 25.000 kepada seorang *bug hunter* yang telah menemukan celah keamanan *XSS* pada instagram [3]. Celah keamanan *XSS* dapat menjadi pencurian data pribadi pengguna seperti token dan *cookies* yang dapat berdampak pada *account takeover*. Berdasarkan total bayaran yang diberikan, *XSS* merupakan salah satu celah keamanan yang cukup krusial karena dari celah keamanan *XSS dapat* dikembangkan menjadi ancaman yang lebih serius seperti *RCE (Remote Code Execution)* yang dapat mengirimkan perintah kepada perangkat *end-user*. Hal ini pernah terjadi pada Cisco Jabber yang ditemukan pada Juni 2020, yang memungkinkan penyerang untuk memberikan perintah jarak jauh terhadap target dapat mengidentifikasi adanya kemungkinan celah keamanan pada aplikasi *website*.

Karena *framework* dan *library* yang digunakan untuk membuat aplikasi *website* saat ini cukup beragam, maka diperlukan sebuah metode yang dapat melakukan analisis secara global tanpa melihat *framework* dan *library* yang digunakan sebuah aplikasi *website*. Pendekatan dengan menggunakan metode *static code analysis* dan *fuzzing attack simulation* dinilai dapat melakukan deteksi *XSS* tanpa harus melihat *framework* dan *library* yang digunakan oleh aplikasi *website*.

Topik dan Batasannya

Berdasarkan latar belakang masalah yang telah dipaparkan, maka rumusan masalah yang diangkat oleh penulis pada penelitian ini yaitu cara mendeteksi celah keamanan XSS pada aplikasi website tanpa melihat pada *library, framework,* dan bahasa pemrograman yang digunakan aplikasi website, pengukuran akurasi dari penerapan *static code analysis* untuk deteksi XSS, dan perhitungan total *vulnerability* pada pengujian otomatis.

Pada proses deteksi XSS, dilakukan dengan cara membangun sistem fuzzer dengan menggunakan *static code analysis*. Metode *static code anlaysis* digunakan untuk menemukan adanya kemungkinan celah keamanan XSS yang selanjutnya konfirmasi dengan fuzzing sesuai dengan payload yang sudah diinput. Pengukuran akurasi dilakukan dengan cara membandingkan pengujian otomatis dan pengujian manual pada aplikasi web testing DVWA. Perhitungan total *vulnerability* didapatkan dari menghitung jumlah total payload yang valid pada pengujian manual dan pengujian otomatis.

Pengujian dilakukan dengan menggunakan aplikasi web testing DVWA. Serangan XSS akan dinyatakan valid apabila payload yang dikirimkan men*trigger function alert* yang disertai dalam payload.

Tujuan

Tugas akhir ini bertujuan untuk membangun sistem fuzzer yang dapat mendeteksi XSS dengan menggunakan *static code analysis*. Adapun parameter keberhasilan pengujian diukur dari akurasi dan total *vulnerability* yang didapatkan. Keberhasilan payload dapat dibuktikan dengan melakukan pengujian manual pada aplikasi web testing DVWA.

Organisasi Tulisan

Penelitian ini disusun berdasarkan organisasi tulisan sebagai berikut: pada bagian awal menjelaskan pendahuluan, pada bagian kedua menjelaskan studi terkait, pada bagian ketiga menjelaskan sistem yang dibangun, pada bagian keempat menjelaskan evaluasi, dan pada bagian kelima menjelaskan kesimpulan.