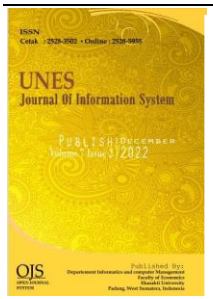


UNES Journal of Information System

Volume 8, Issue 2, December 2023

P-ISSN 2528-3502

E-ISSN 2528-5955

Open Access at: https://fe.ekasakti.org/index.php/UJIS

PERANCANGAN APLIKASI OTOMATISASI DETEKSI XSS BERBASIS CLI DENGAN BAHASA PEMROGRAMAN GO

CLI-BASED XSS DETECTION AUTOMATION APPLICATION DESIGN WITH GO PROGRAMMING LANGUAGE

Rika Rosnelly[1)], Satrya Wira Yudha[2)]

[1,2] Informatika, Teknik dan Ilmu Komputer, Universitas Potensi Utama.

E-mail: rikarosnelly@gmail.com, satryawira23@gmail.com

INFO ARTIKEL

Kata kunci

Fuzzing, Web, XSS, CLI, GO

ABSTRAK

Kemajuan dunia digital, keamanan aplikasi web menjadi perhatian kritis. Salah satu celah keamanan yang umum ditemui adalah XSS, di mana penyerang memanfaatkan celah pada aplikasi web untuk menyisipkan skrip berbahaya ke dalam halaman web yang kemudian dapat dieksekusi oleh pengguna lain. Serangan XSS dapat digunakan untuk mengeksekusi kode berbahaya di browser korban, yang dapat menyebabkan pencurian data, penyebaran malware, dan gangguan layanan. Ada berbagai teknik yang dapat digunakan untuk mendeteksi kerentanan XSS. Salah satu teknik yang paling umum adalah metode Fuzzing. Fuzzing merupakan suatu metode mencari kesalahan piranti lunak dengan menyediakan input yang tidak diduga lalu memonitoring hasilnya. Fuzzing pada umumnya merupakan proses otomatisasi atau semi automatisasi yang melibatkan manipulasi dan menyediakan data secara berulang-ulang untuk selanjutnya akan diproses oleh target piranti lunak. Penelitian ini akan membuat sebuah rancangan alat otomatisasi yang dapat digunakan untuk melakukan pengujian terhadap deteksi kerentanan xss pada web.

Copyright © 2023 UJIS. All rights reserved.

ARTICLE INFO

Keywords:
Fuzzing, Web, XSS, CLI, GO

ABSTRACT

With the advancement of the digital world, web application security has become an important concern. One commonly encountered security flaw is XSS, where an attacker exploits a loophole in a web application to inject malicious script into a web page that can then be executed by other users. XSS attacks can be used to execute malicious code in a victim's browser, which can lead to data theft, malware spread, and service disruption. There are various techniques that can be used to detect XSS vulnerabilities. One of the most common techniques is the Fuzzing method. Fuzzing is a method of finding software errors by providing unexpected input and then monitoring the results. Fuzzing is generally an automation or semi-automation process that involves repeatedly manipulating and providing data for further processing by the target software. This research will design an automation tool that can be used to test XSS vulnerability detection on the web.

Copyright © 2023 UJIS. All rights reserved.

PENDAHULUAN

Dalam era digital yang semakin berkembang, keamanan aplikasi web menjadi perhatian kritis bagi organisasi dan pengembang. Salah satu celah keamanan yang umum ditemui adalah XSS Cross Site Scripting (XSS) pada dasarnya adalah serangan code injection dimana script dimasukkan sebagai kode berbahaya[1]. Cross-site scripting (XSS) adalah salah satu kerentanan keamanan web yang paling umum dan berbahaya. Serangan XSS dapat digunakan untuk mengeksekusi kode berbahaya di browser korban, yang dapat menyebabkan pencurian data, penyebaran malware, dan gangguan layanan. Untuk mengatasi ancaman XSS ini, dibutuhkan upaya yang kuat dalam mengidentifikasi dan menghilangkan celah keamanan potensial.

Ada berbagai teknik yang dapat digunakan untuk mendeteksi kerentanan XSS. Salah satu teknik yang paling umum adalah metode Fuzzing. Fuzzing adalah suatu metode dengan cara memasukan nilai tak terduga ke sistem target dan memonitor respon abnormal untuk menemukan kerentanan aplikasi web [2].

Jurnal ini membahas tentang perancangan alat otomatisasi deteksi XSS berbasis CLI dengan bahasa pemrograman Go. Command Line Interface (CLI) adalah antarmuka pengguna yang paling populer dan paling dasar untuk berinteraksi dengan komputer[3]. Command line interface (CLI) adalah antarmuka pengguna (UI) berbasis teks yang digunakan untuk berinteraksi dengan komputer. Pemrograman Go adalah bahasa pemrograman yang dikembangkan oleh perusahaan Google[4]. Go adalah bahasa pemrograman yang multi-paradigma, yang berarti dapat digunakan untuk berbagai gaya pemrograman, termasuk pemrograman prosedural, pemrograman berorientasi objek, dan pemrograman fungsional.

Jurnal penelitian sebelumnya KameleonFuzz: evolutionary fuzzing for black-box XSS detection [5]. XSS PEEKER: Dissecting the XSS Exploitation Techniques and Fuzzing Mechanisms of Blackbox Web Application Scanners [6]. Black-box adversarial attacks

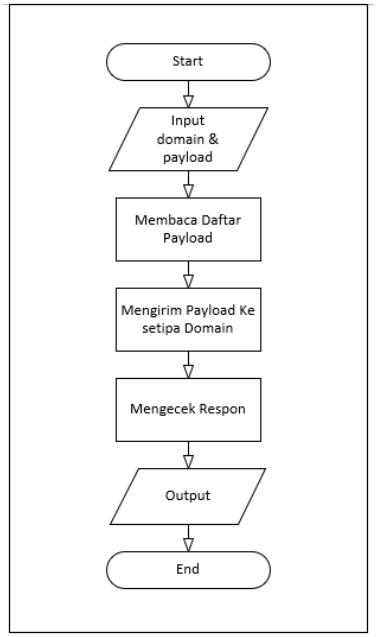
on XSS attack detection model [7]. Adaptive Random Testing for XSS Vulnerability [8]. Peneliti sebelumnya membuat aplikasi deteksi xss yang tidak dapat dikombinasikan dengan aplikasi penetration lain.

Pada penelitian ini penulis membuat tools otomatisasi yang dapat di kombinasikan dengan tools penetrasi lain menggunakan teknik fuzzing untuk memeriksa respons aplikasi dan mencari tanda-tanda eksekusi kode berbahaya. Alat otomatisasi ini diharapkan dapat membantu para pentester dalam mendeteksi kerentanan XSS secara lebih cepat dan efisien.

METODE PENELITIAN

Beberapa masalah yang dihadapi dalam pengujian keamanan untuk mendeteksi xss adalah keragaman jenis payload. Untuk mengatasi masalah ini maka menggunakan metode fuzzing untuk menguji kerentanan xss pada situs web. Pada penelitian ini metode fuzzing dijalankan untuk mengirimkan payload ke setiap domain. Payload adalah kode yang dapat digunakan untuk mengeksploitasi kerentanan XSS. Setelah payload dikirimkan metode fuzzing akan memeriksa respons untuk melihat apakah ada tanda-tanda kerentanan XSS. Alur dari rancangan dapat dilihat pada gambar 1.

Gambar 1 flowchart aplikasi Xss



Tahapan dalam fuzzing secara umum dapat dibedakan menjadi 6 tahapan yakni identifikasi target, identifikasi input, Membuat data fuzzing, input data fuzzing, monitoring respon, dan evaluasi. Berikut adalah tahapan-tahapan yang dilakukan:

1. Membaca daftar domain dari input user

Pada tahap ini, program membaca daftar domain dari input user. Daftar domain ini digunakan sebagai sasaran fuzzing.

```
// Membaca daftar domain dari input user
sc := bufio.NewScanner(os.Stdin)

// Ambil domain dari input user
for sc.Scan(){
        domain := sc.Text()
        jobs <- domain
}
```

2. Membuat channel untuk mengirim domain ke goroutine

Pada tahap ini, program membuat channel untuk mengirim domain ke goroutine. Channel ini digunakan untuk mendistribusikan domain ke goroutine secara paralel.

```
// Buat channel untuk mengirim domain ke goroutine
jobs := make(chan string)
```

3. Memulai 20 goroutine untuk melakukan fuzzing

Pada tahap ini, program memulai 20 goroutine untuk melakukan fuzzing. Setiap goroutine akan mengambil domain dari channel dan mengirimkan permintaan HTTP GET ke domain tersebut.

```
// Mulai 20 goroutine untuk melakukan fuzzing
for i:= 0; i < 20; i++{
        wg.Add(1)
        go func(){
                defer wg.Done()
                for domain := range jobs {
                        // Kirim permintaan HTTP GET ke domain
                        ...
                }
        }()
}
```

4. Mengirim permintaan HTTP GET ke domain

Pada tahap ini, goroutine mengirimkan permintaan HTTP GET ke domain. Permintaan HTTP GET adalah permintaan yang paling sederhana dan tidak memerlukan data apa pun.

```
// Kirim permintaan HTTP GET ke domain
resp, err := http.Get(domain)
if err != nil{
        continue
}
```

5. Memeriksa respons untuk melihat apakah ada tanda-tanda kerentanan XSS

Setelah permintaan HTTP GET dikirim, goroutine akan memeriksa respons untuk melihat apakah ada tanda-tanda kerentanan XSS. Tanda-tanda kerentanan XSS dapat berupa kode yang tidak diharapkan muncul di respons domain.

```
// Periksa respons untuk melihat apakah ada tanda-tanda kerentanan XSS
for _, payload := range payloads {
        if strings.Contains(sb, payload) {
                vulnerable = true
                break
        }
}
```

6. Mencetak domain jika rentan terhadap XSS

Jika ada tanda-tanda kerentanan XSS, goroutine akan mencetak domain tersebut.

```
// Cetak domain jika rentan terhadap XSS
if vulnerable {
        fmt.Println(string(colorRed), "Possible Xss:", domain, string(colorReset))
} else {
        fmt.Println(string(colorGreen), "Not Vulnerable:", domain, string(colorReset))
}
```

HASIL DAN PEMBAHASAN

Alat ini dapat digunakan untuk mengirim banyak permintaan HTTP dapatkan dengan cepat dan mendeteksi kerentanan xss. Alat ini juga dapat digunakan bersamaan dengan Alat otomatisasi seperti qsreplace dan sejenisnya. Proses alat ini bekerja dapat dijelaskan sebagai berikut:

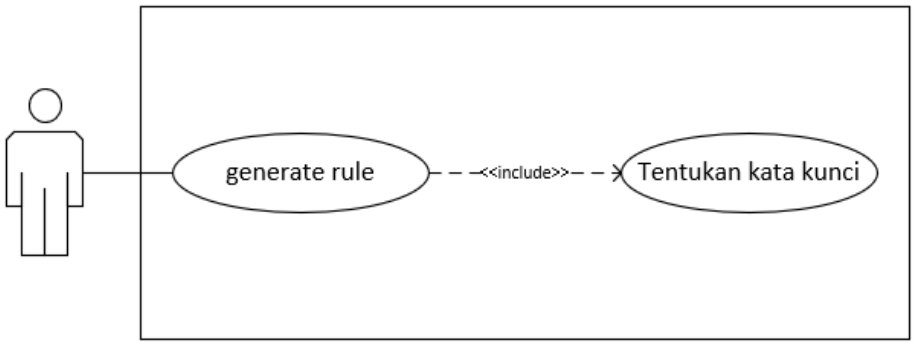
Perancangan UML

UML adalah salah satu tool/model untuk merancang pengembangan software yang berbasis object-oriented [9]. UML digunakan untuk menggambarkan visualisasi dari berbagai aspek sistem, termasuk struktur, fungsi, dan interaksi antara komponen-komponennya.

Perancangan UML yang digunakan dalam mengerjakan Penulisan ini menggunakan UML, yaitu Use Case Diagram

Use Case Diagram

Alur dari aplikasi yang dibuat dalam Use Case Diagram dapat dilihat, seperti pada Gambar 2



Gambar 2 use case diagram aplikasi xss

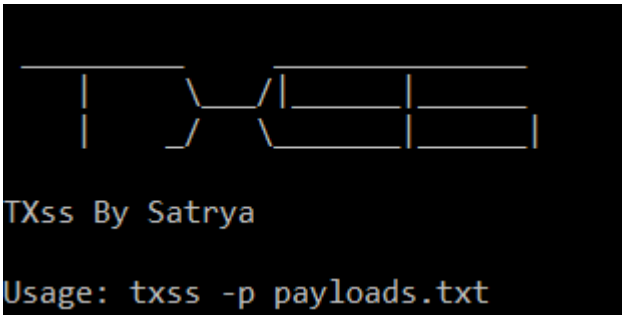
Gambar 2 menunjukkan use case diagram dari alat automatic generate rule. Alat ini dijalankan user dengan mengetikkan perintah pada command prompt. Pada case ini untuk membuat suatu rule aplikasi membutuhkan input berupa kata kunci pembuatan rule, nama payload txt untuk output rule dan jumlah sumber rule. Alat ini bekerja dengan terlebih dahulu melakukan listing terhadap semua kemungkinan rule yang bisa dibentuk berdasarkan kata kunci yang diinput. Lalu jumlah rule yang akan disimpan pada file payload text akan dibatasi oleh jumlah rule yang diinput user

Perancangan dan Implementasi Alat Deteksi Xss

Alat dibuat sesuai dengan perancangan yang dibuat menggunakan bahasa pemrograman go. berikut tampilan alat otomatisasi xss.

Tampilan Alat

Alat ini berbasis command line interface. Untuk menampilkan tampilan atau melihat perintah apa saja yang dapat dijalankan. Ketikan perintah main go run main.go pada terminal. Maka tampilan pada gambar 3 akan muncul



Gambar 3 tampilan aplikasi

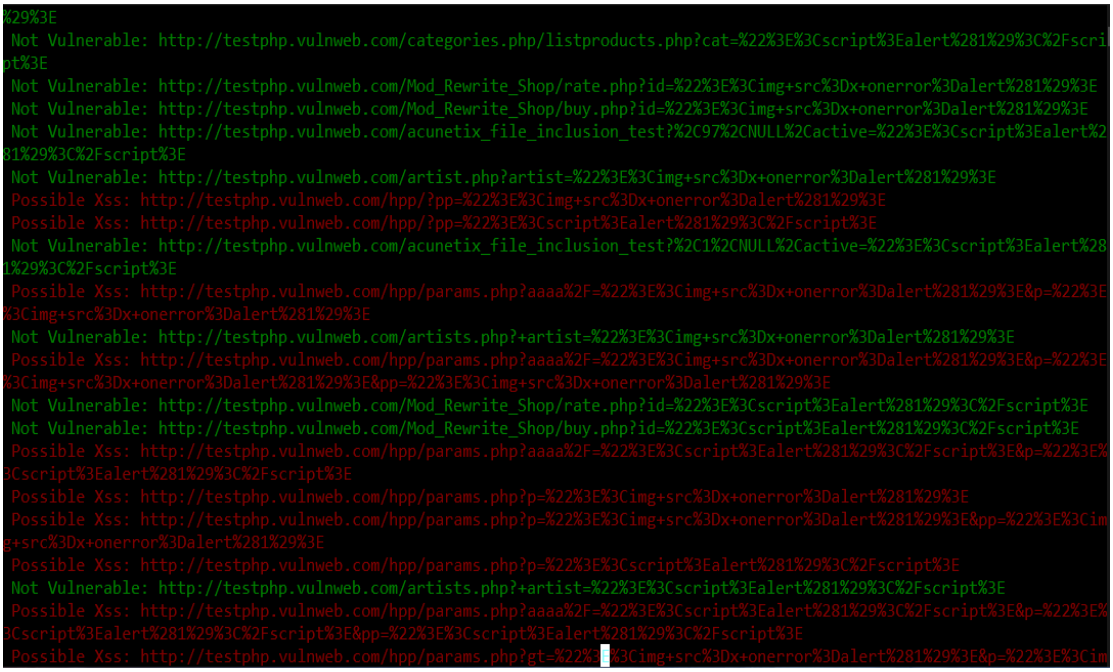
Tampilan Saat Dijalankan

Alat ini dapat dijalankan dengan mengkombinasikan dengan alat penetration yang lain seperti waybackurls, qsreplace dan lain-lain. Untuk menjalankan alat ada beberapa tahapan sebagai berikut:

1. Mengumpulkan seluruh url dari website dengan menggunakan Alat seperti waybacurls, gau dan sejenisnya
2. Memasukan payload xss pada setiap parameter dari url yang dikumpulkan menggunakan qsreplace dan alat sejenisnya.
3. Kemudian jalankan Txss ini untuk mengecek apakah ada kode berbahaya pada respone domain dari yang sudah di suntikkan payload xss berdasarkan payload yang dipilih.

Jika tahapan diatas sudah dilakukan maka tampilan akan seperti gambar 4

Gambar 4 tampilan saat di jalankan

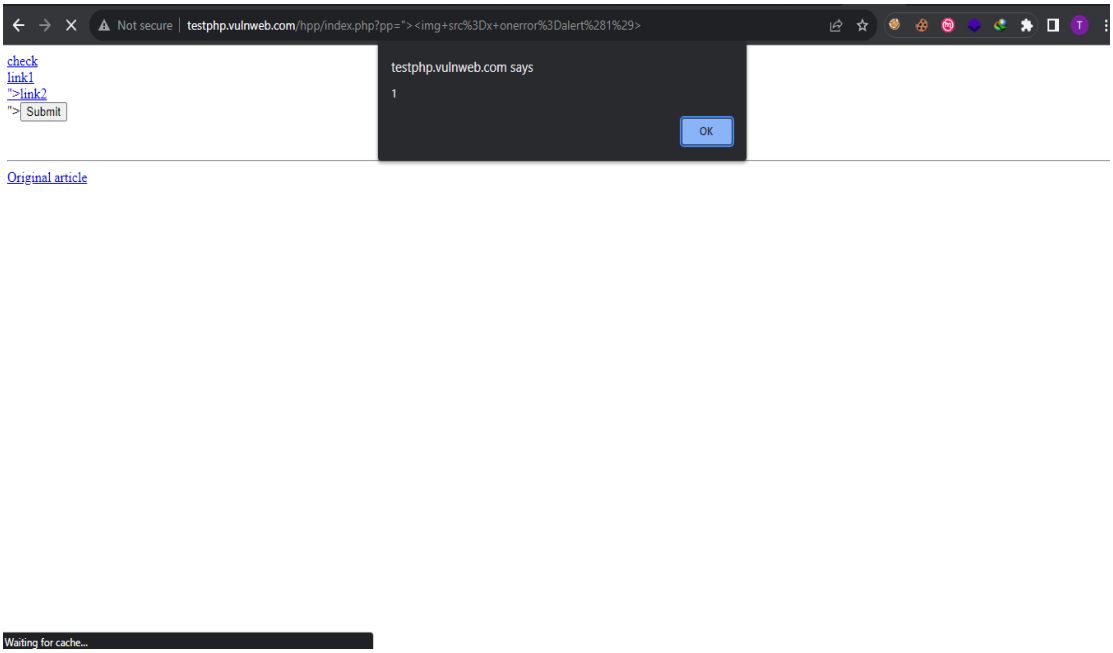


Metode Pengujian

Metode yang digunakan untuk pengujian adalah Vulnerability scanning. Vulnerability scanning adalah proses memperoleh informasi vulnerability dengan memanfaatkan berbagai tools network scanning dan vulnerability scanner, seperti port yang terbuka, bugs aplikasi server dan lain-lain[10] . Adapun alat yang digunakan pada penelitian ini yaitu Waybackurls, qsreplace, dan Txss. Bug yang ingin dicari pada penelitian ini adalah kerentanan xss.

Pada penelitian ini penulis melakukan pengujian terhadap website testphp.vulnweb.com . Hasil vulnerability scanning dengan Txss dapat dilihat pada gambar 5 dan 6.

Gambar 5 jumlah yang terdeteksi



Gambar 6 hasil

Dari hasil gambar 5 diatas terlihat bahwa alat Txss dapat mendeteksi xss pada web. jumlah yang berhasil di deteksi dapat dilihat pada tabel 1.

Tabel 1 Jumlah kerentanan yang terdeteksi

Vulnerability Scanning	Alert	Total Alert
TXss	Reflected Xss	26

SIMPULAN

Setelah melakukan pengujian alat ini berjalan dengan baik. Tidak kesalahan dalam source code dan Alat ini mampu mendeteksi kerentanan xss pada sebuah website serta alat ini dapat dikombinasikan dengan alat otomatisasi lainnya seperti waybackurls dan qsreplace. Dapat disimpulkan, sebagai berikut:

1. Alat ini tidak ada kesalahan dalam tahap pengkodean
2. Kerentanan yang terdapat pada parameter GET mampu di deteksi dengan payload yang sudah dikumpulkan
3. Dan alat ini dapat dikombinasikan dengan alat otomatisasi lain
4. Alat ini Hanya mendeteksi xss pada parameter GET. Tidak bisa mendeteksi xss pada parameter POST

DAFTAR PUSTAKA

Hany, M. I., Bhawiyuga, A., & Kusyanti, A. (2021). Implementasi Cross Site Scripting Vulnerability Assessment Tools berdasarkan OWASP Code Review. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *5*(9), 3745-3753.

Kristiawan, Y. Fuzzing untuk Menemukan Kerentanan Aplikasi Web.

Verma, P. (2013). Gracoli: a graphical command line user interface. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems* (pp. 3143-3146).

Arisha, l. G. (2018). *Penggunaan bahasa go untuk menghitung energi keadaan dasar atom-atom sederhana berbasis density functional theory (dft)* (doctoral dissertation, universitas airlangga).

Mallisza, D., Adri, J., & Ismanto, H. (2022). The Improving Efort Of Technical Drawing With Giving An Assignment Methode (Recitation) Students Grade X Tkr 1 SMK State 2 Painan. International Conference On Global Education, 434-438. Retrieved from http://114.5.194.187/index.php/ICGE/article/view/124

Duchene, F., Rawat, S., Richier, J. L., & Groz, R. (2014, March). KameleonFuzz: evolutionary fuzzing for black-box XSS detection. In *Proceedings of the 4th ACM conference on Data and application security and privacy* (pp. 37-48).

Bazzoli, E., Criscione, C., Maggi, F., & Zanero, S. (2016). XSS PEEKER: Dissecting the XSS exploitation techniques and fuzzing mechanisms of blackbox web application scanners. In *ICT Systems Security and Privacy Protection: 31st IFIP TC 11 International Conference, SEC 2016, Ghent, Belgium, May 30-June 1, 2016, Proceedings 31* (pp. 243-258). Springer International Publishing.

Wang, Q., Yang, H., Wu, G., Choo, K. K. R., Zhang, Z., Miao, G., & Ren, Y. (2022). Black-box adversarial attacks on XSS attack detection model. *Computers & Security*, *113*, 102554.

Lv, C., Zhang, L., Zeng, F., & Zhang, J. (2019, December). Adaptive random testing for XSS vulnerability. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 63-69). IEEE.

Sonata, F. (2019). Pemanfaatan UML (Unified Modeling Language) dalam perancangan sistem informasi e-commerce jenis customer-to-customer. *Jurnal Komunika: Jurnal Komunikasi, Media Dan Informatika*, *8*(1), 22-31.

Khulaimi, M., & Taqiudin, M. (2023). Management Konfigurasi Hotspot Local Area Network (LAN) SMK Darussholihin NW Kalijaga Menggunakan Metode Vulnerability Scanning. *Digital Transformation Technology*, *3*(2), 418-425.