# BAB II

# TINJAUAN PUSTAKA

## 2.1 Penelitian Terdahulu

 Terdapat sebuah penelitian yang telah dilakukan sebagai berikut :

###### 1. *Securing Web Applications from malware attacks using hybrid feature extraction* (Subramaniyaswamy V.\*, Gopireddy Venkata Kalyani, Naladala Likhita, 2018)

Penelitian ini dilakukan karena di era teknologi ini banyak aplikasi yang memanfaatkan layanan internet untuk memenuhi kebutuhan penggunanya. Dengan meningkatnya jumlah pengguna internet, terjadi substantial dalam serangan internet. Karena kenaikan ini, layanan web menimbulkan ancaman keamanan baru.Salah satu kerentanan yang perlu diwaspadai adalah kerentanan *Cross-Site Scripting* (XSS). Lebih dari tiga perempat serangan berbahaya disumbangkan oleh *Cross-Site Scripting* (XSS). Disini akan fokus pada deteksi dan eksploitasi kerentanan *Cross-Site Scripting* (XSS). Selain itu telah dijelaskan sebuah kerangka kerja yang direncanakan untuk kontradiksi kerentanan *Cross-Site Scripting* (XSS).

8

###### 2. *Cross Site Scripting: Detection Approaches in Web Application* (Wasef dan

Zaaba, 2016)

Penelitian ini dilakukan untuk melakukan sebuah analisa tentang masalahmasalah kerentanan *Cross-Site Scripting (XSS)* dalam sebuah aplikasi website dan arah masa depan penelitian ini juga dilihat. Dalam penelitian ini juga terdapat beberapa pendekatan yang disimpulkan. Pendekatan hybrid tidak efisien sebagai pendekatan yang sepenuhnya statis atau dinamis. Algoritma genetika cukup berhasil dalam mendeteksi semua kerentanan XSS dalam aplikasi web tanpa kesalahan namun, ketika menerapkan di PHP, hasilnya masih mengandung banyak kesalahan. Kesimpulan dari penelitian ini yaitu algoritma GA( Genetika Algoritma) telah terbukti efektif dalam mendeteksi kerentanan XSS.

###### 3. *Automated Detecting and Repair of Cross-Site Scripting Vulnerabilities through Unit Testing* (Mohammadi, Chu, Lipford, 2017)

Penelitian ini dilakukan untuk mendeteksi kerentanan *Cross-Site Scripting (XSS)* yang disebabkan oleh pengkodean data yang salah menggunakan pendekatan unit-test untuk mendeteksi kerentanan. Kesimpulan dari penelitian ini yaitu pendekatan yang dilakukan berhasil menemukan kerentanan *CrossSite Scripting (XSS)* dalam skala besar menggunakan metode fuzzing. Sistem encode yang salah berpengaruh besar dan akan menyebabkan kerentanan.

## 2.2 *Fuzzing*

 *Fuzzing* merupakan sebuah metode yang dilakukan untuk mendeteksi kerentanan pada sebuah website secara otomatis sehingga dapat membuat sistem mengalami sebuah crash dan penyerang memanfaatkan celah tersebut untuk kepentingannya. (Jinho,David,Daniel,Kyu dan Taesoo, 2019).



##### Gambar 2.1 Cara Kerja *Fuzzing*

sumber: http://budi.rahardjo.id/files/courses/2016/EL5215-2016-23215097-Report.pdf Proses tersebut bisa dijelaskan sebagai berikut :

1. *Web form crawler*

*Web form crawler* berfungsi untuk mencari form input dari web yang akan menjadi target fuzzing.

1. *Attack script*

*Attack script* yaitu sebuah script yang digunakan untuk melakukan input fuzzing

1. *Fuzz program*

Fuzz program berfungsi untuk generate *attack script* untuk diubah menjadi Bahasa *HTTP* yang akan digunakan untuk melakukan input terhadap aplikasi web.

1. *Web Application (HTTP Client)*

*HTTP client* merupakan sasaran yang tepat untuk menginputkan nilai-nilai acak yang digenerate sebelumnya lalu mengirimkan input ke host dan respon dari host akan ditampilkan *HTTP client* untuk selanjutnya akan disimpan dan dicatat pada *log*.

1. *Logs*

*Logs* merupakan file yang disimpan dari hasil respon *host* terhadap nilai input yang diberikan seperti jenis *error*. Nilai inilah yang akan di Analisis untuk mengetahui jenis kerentanan pada aplikasi web yang diuji.

## 2.3 *Cross-Site Scripting* (XSS)

 *Cross-Site Scripting (XSS)* merupakan sebuah kerentanan yang secara umum sering terjadi. *Cross-Site Scripting (XSS)* terjadi ketika penyerang mengambil sebuah data tanpa melakukan sebuah validasi terlebih dahulu. Penyerang akan mengeksekusi sebuah browser target dengan cara mengambil session, cookie, deface atau redirect pengguna ke situs lain. (Raharjo, 2016). *Cross-Site Scripting(XSS)* merupakan sebuah kerentanan yang perlu diwaspadai pada sebuah website. Penyerang bisa dengan mudah mengeksploitasi pertukaran *cookies* antara web server dan browser.

*Cross-Site Scripting(XSS)* ini menyerang pada halaman yang mempunyai parameter pada sebuah website. Berikut adalah data kerentanan yang dilansir dari imperva.com.



##### Gambar 2.2 Data Kasus Kerentanan

sumber : https://www.imperva.com/blog/the-state-of-web-application-vulnerabilities-in-2018/

#### 2.3.1 Macam-Macam *Cross-Site Scripting* (XSS)

 Serangan *Cross-Site Scripting* dapat diklasifikasikan dalam tiga kategori dan setiap kategori mempunyai hasil berbeda-beda ketika diimplementasikan. Hasilnya sebagai berikut:

*1. Reflected*

*Reflected Cross-Site Scripting* (XSS) merupakan sebuah kerentanan yang terjadi ketika kode injeksi yang diinputkan dapat tercermin pada halaman tanpa tersimpan pada database. Simulasinya sebagai berikut :



##### Gambar 2. 3 *Reflected* XSS

sumber: jurnal Securing Web Applications from malware attacks using hybrid feature extraction

 Keterangan

 Penyerang akan menginputkan kode injeksi terhadap sebuah halaman website.

Lalu jika berhasil browser akan mengirim sesi *cookie* terhadap penyerang. Kemudian ketika penyerang sudah memiliki *cookie* tersebut maka penyeraang akan memodifikasi halaman website tersebut lalu akan mengirimkan halaman website yang telah dimodifikasi kepada korban dan ketika korban klik link tersebut maka sesi korban akan diambil oleh penyerang.

2. *Stored Cross-Site Scripting* (XSS)

*Stored Cross-Site Scripting* (XSS) merupakan sebuah kerentanan yang terjadi ketika kode injeksi yang diinputkan dapat tersimpan dalam database yang terdapat pada website tersebut. Simulasinya sebagai berikut :



##### Gambar 2. 4 *Stored* XSS

 sumber: jurnal Securing Web Applications from malware attacks using hybrid feature extraction

 Keterangan

 Penyerang akan mencari sebuah website yang akan diinputkan kode injeksi. Setelah website sudah ditemukan maka penyerang akan menginputkan kode injeksi pada website tersebut. Lalu kode injeksi yang diinputkan berhasil masuk dalam database pada website tersebut kemudian penyerang

memodifikasi link website yang terdapat kerentanan *stored* lalu mengirimkan

kepada korban dan ketika link tersebut diklik oleh korban maka sesi *cookie* korban akan dikirimkan ke penyerang.

3. *DOM-Based Cross-Site Scripting* (XSS)

*Document Object Model* (DOM) merupakan kerentanan yang terjadi ketika kode injeksi yang diinputkan dapat memanipulasi *Document Object Model* (DOM) yang ada pada website tersebut. Simulasinya sebagai berikut :



###### Gambar 2. 5 DOM-*based*

sumber: https://medium.com/iocscan/dom-based-cross-site-scripting-dom-xss-3396453364fd

Keterangan

Kerentanan DOM-*based* terjadi hanya pada akun sendiri. Jadi ketika website tersebut dinyatakan terdapat kerentanan DOM *Cross-Site Scripting* (XSS) ketika halaman website tersebut dimodifikasi lalu dikirimkan terhadap korban maka tidak akan terjadi apa-apa pada korban dikarenakan kerentanan tersebut hanya ada pada akun sendiri dan tidak akan menyebabkan kerentanan pada akun orang lain.

## 2.4 *Pentester*

 *Pentester (penetration tester)* merupakan seseorang yang melakukan kegiatan *pentesting (penetration testing)* atau bisa disebut dengan kegiatan mencari kerentanan pada sebuah sistem jaringan sesuai dengan 10 standar resmi dari *Open Web*

*Application Security Project* (OWASP) sebagai acuan dalam melakukan pentesting. OWASP merupakan sebuah Lembaga atau organisasi terbuka yang bertujuan untuk menampung semua jenis laporan kerentanan pada sebuah website. Ada 10 jenis kerentanan berdasarkan standard OWASP dengan urutan sebagai berikut :



Gambar 2. 6 10 Standar OWASP

sumber : https://owasp.org/www-pdf-archive/OWASP\_Top\_10-2017\_%28en%29.pdf.pdf

Keterangan

Terdapat perbedaan pada tahun 2013 dan 2017 dikarenakan telah ditemukan sebuah kerentanan baru pada tahun 2017 yang sangat berdampak pada sebuah wesbite. Urutan kerentanan tersebut merupakan kerentanan yang banyak dilaporkan dan akan mengalami perubahan pada setiap tahunnya.

## 2.4 *Linux*

 *Linux* merupakan sebuah sistem operasi yang gratis. Kode sumber *linux* boleh dimodifikasi dan didistribusikan kembali secara bebas oleh siapa saja. Salah satu modifikasi dari linux yaitu Debian dan dimodifikasi lagi menjadi ubuntu. Keuntungan menggunakan linux untuk pembuatan *tools* yaitu mudah digunakan, mengalami keamanan yang unggul, kinerja yang ringan dan tidak perlu *install* *driver*.

## 2.5 *Command-Line Interface* (CLI)

 *Command-Line Interface* (CLI) atau bisa disebut dengan antarmuka baris perintah adalah sebuah baris perintah yang digunakan untuk mengoperasikan sebuah perangkat lunak dan sistem operasi sehingga memungkinkan pengguna untuk menanggapi permintaan visual dengan mengetikkan sebuah perintah tunggal ke dalam antarmuka baris perintah dan menerima balasan dengan cara yang sama. Keuntungan menggunakan CLI yaitu bisa me*recovery* jika terjadi *crash* dan mudah digunakan.

## 2.5 *Python*

 *Python* merupakan sebuah bahasa pemrograman yang interpretatif multiguna. Perancangan python dengan menggunakan filosofi yang berfokus pada tingkat keterbacaan kode. Python merupakan bahasa yang menggabungkan kemampuan, kapabilitas dengan sintak yang sangat jelas serta dilengkapi dengan libraries yang besar dan komprehensif. Python termasuk bahasa pemrograman yang dinamis dan dilengkapi dengan manajemen memori secara otomatis. Python bisa digunakan untuk pengembang perangkat lunak dan bisa berjalan di platform sistem operasi. Python masih merupakan bahasa yang dominan di dunia keamanan informasi. Alat berbasis python merupakan alat yang sangat exploit sekali. (Black Hat Python, Justin Seitz)

#### 2.5.1 *Python 3*

 Berikut merupakan sebuah tabel perbandingan python 2 dan python 3 menurut data dari python.org

Tabel 2. 1 Perbandingan *Python*

|  |
| --- |
| TABEL PERBANDINGAN *PYTHON*  |
| *Python* 2  |   | *Python* 3  |
| ASCII  | +  | Unicode  |
| 5/2=2  | !=  | 5/2=2.5  |
| print”hello”  | !=  | print(“hello”)  |

Keterangan :

* *Python* dua mempunyai ASCII dan python 3 mempunyai ASCII dan *Unicode*
* Cara pembagian *python* 2 dan *python* 3 tidak sama
* Mempunyai fungsi print *code* yang berbeda

#### 2.5.2 *Libraries Python*

 Pembuatan tools ini membutuhkan sebuah libraries yang sudah dianalisa.

libraries yang dibutuhkan kurang lebih seperti berikut .

**Tabel 2. 2 *Lybrary python***

|  |  |  |
| --- | --- | --- |
| No  | Nama *Library*  | Fungsi *Library*  |
| 1  | Import OS  | Modul python untuk berinteraksi dengan sistem operasi  |
| 2  | Import copy  | Modul python untuk mendemonstrasikan operasi  |

|  |  |  |
| --- | --- | --- |
|  |  | penyalinan  |
| 3  | Import random  | Modul python untuk menghasilkan angka yang acak  |
| 4  | Import randint  | Modul python turunan dari random yang berfungsi mengembalikan integer  |
| 5  | Import sleep  | Modul python untuk menunda eksekusi selama beberapa detik  |
| 6  | Import time  | Modul python untuk mengembalikan jumlah detik yang berlalu  |
| 7  | Import fuzzywuzzy  | Modul python untuk melakukan pencocokan string  |
| 10  | Import re  | Modul python untuk regular ekspression  |
| 11  | Import base64  | Modul python untuk melakukan pengkodean base64  |
| 12  | Import logging  | Modul python untuk mencatat pesan yang ingin dilihat  |
| 13  | Import warnings  | Modul python untuk menampilkan pesan warning  |
| 14  | Import requests  | Modul python untuk membuat permintaan ke halaman web dan melihat respon dari web tersebut  |
| 15  | Import ProtocolError  | Modul python untuk mendeteksi sebuah eror dalam berlangsungnya proses pemindaian  |
| 16  | Import json  | Modul python untuk mengubah kamus python menjadi file json  |
| 17  | Import hashlib  | Modul python untuk mengambil Panjang variable byte dan mengubahnya menjadi urutan Panjang tetap  |
| 18  | Import urlparse  | Modul python untuk memecah string URL (Uniform Resource Locator) dalam komponen dan menggabungkan Kembali komponen ke dalam string URL (Uniform Resource Locator) .  |
| 19  | Import sys  | Modul python untuk menyediakan akses ke beberapa variable yang digunakan dan memberi nama semua modul ked ala interpreter python  |
| 20  | Import platform  | Modul python untuk mengakses data platform seperti perangkat keras, sistem operasi  |

## 2.5 *Exploit*

 *Exploit* merupakan sebuah cara yang digunakan oleh penyerang maupun pentester. Pada metode ini penyerang atau pentester dapat mengambil keuntungan dari kecacatan sebuah sistem. Sistem akan diserang dengan cara yang bisa memberikan hasil yang diinginkan.

## 2.6 *Payload*

 *Payload* merupakan sebuah kode injeksi dimana penyerang menginginkan aplikasi web mengeksekusinya sehingga terjadi sebuah kesalahan. Contohnya penyerang dapat terkoneksi dengan server melalui sebuah payload yang dikirimkan pada sebuah aplikasi web

## 2.7 *Web Application Firewall* (WAF)

 Web Application Firewall (WAF) merupakan sebuah aplikasi firewall yang memiliki fungsi untuk memantau dan memblokir lalu lintas yang mencurigakan terhadap website. Gambaran dari WAF adalah sebagai berikut :



 sumber : https://avinetworks.com/what-is-a-web-application-firewall/

###### Gambar 2. 7 *Web Application Firewall* (WAF)

Keterangan :

Ketika user ingin mengisi buku tamu yang ada pada website maka, waf akan melakukan penyaringan terhadap isi dari buku tamu yang diinputkan oleh user. Jika isi buku tamu yang diinputkan tersebut bukan string yang berbahaya maka waf akan meloloskan inputan tersebut dan menyimpannya di database. Tapi jika user tersebut menginputkan string berbahaya seperti *<script>alert(‘xss’)</script>* maka waf juga akan melakukan penyaringan terhadap string berbaya tersebut dan jika waf pada website tersebut memiliki keamanan yang sangat baik maka, string berbahaya yang diinputkan tidak akan berhasil memasuki database website dan string berbahaya yang diinputkan akan diblokir oleh waf. Jika waf pada website tersebut tidak cukup aman maka string berbahaya akan diloloskan dan akan menginjeksi database pada website tersebut.

## 2.8 *Crawling*

 *Crawling* merupakan sebuah metode untuk melakukan sebuah pemindaian terhadap semua halaman website secara bersamaan.

#### 2.8.1 Cara Kerja *Crawling*

 Berikut ini adalah gambaran sederhana cara kerja crawling



Gambar 2. 8 Cara Kerja *Crawling*

Keterangan :

Gambar diatas merupakan cara kerja *crawling*. Pertama-tama *tools* akan melakukan pencarian file *JQuery*, *Bootstrap* pada alamat website yang diinputkan lalu ketika pada halaman *http//:web.com/index.php* terdapat sebuah link website *http://aaa.com* maka secara otomatis *tools* akan mencari kerentanan di website tersebut. Jika pada website tersebut terdapat tautan ke website lain maka *tools* akan melakukan perayapan terhadap website yang tertaut tersebut dan seterusnya sampai benar-benar tidak ada alamat website yang bisa dirayapi.

## 2.9 *Regular Expression* (REGEX)

 *Regex* merupakan sebuah karakter atau sekumpulan notasi yang digunakan untuk mendeskripsikan suatu pola pada sebuah pencarian yang berbasis huruf. *Regex* bisa mengenali suatu string atau yang mempunyai karakter tertentu seperti email atau tanggal lahir (Opie, 2014). Contohnya dengan notasi d+by maka akan menyaring semua kata kecuali kata d dan by seperti *deby, diby, doby*.

## 2.10 *Encode Payload*

 *Encode* merupakan sebuah filter penyandian yang digunakan untuk mengubah bentuk asal menjadi bentuk yang lebih optimal. Tujuan *encode* yaitu untuk mengubah sebuah data menjadi sebuah data yang bisa digunakan oleh berbagai macam sistem.

## 2.11 Base64

 Base64 merupakan string yang dikonversi yang merupakan konversi dari karakter non ASCII menjadi ASCII. *Encode* payload melalui base64 merupakan suatu cara yang bisa menghindari *payload* terdeteksi oleh WAF.

## 2.12 *Parsing*

 *Parsing* merupakan suatu cara untuk memecah-mecah sebuah rangkaian masukan yang nantinya menghasilkan suatu uraian yang akan digunakan pada tahap berikutnya.

## 2.13 JSON (*JavaScript Object Notation*)

 JSON merupakan sebuah format yang ringkas untuk melakukan pertukaran data komputer. Formatnya sangat mudah dibaca oleh manusia karena berbasis teks serta dapat digunakan merepresentasikan larik asosiatif dan struktur data sederhana.

## 2.14 Website Testing

 Website sangat dibutuhkan untuk menunjang berlangsungnya pengujian *tools* kerentanan *Cross-Site Scripting* (XSS) berbasis *Command-Line Interface* (CLI) yang telah dibuat

## 2.15 *Flowchart*

 *Flowchart* merupakan sebuah bagan yang terdapat symbol-simbol tertentu, simbol-simbol tersebut berfungsi untuk menggambarkan sebuah proses secara detail dan menghubungkan antara satu proses dengan proses lainnya dalam suatu program

Berikut merupakan symbol-simbol flowchart:

##### Tabel 2. 3 Simbol *flowchart*

|  |  |  |  |
| --- | --- | --- | --- |
| NO  | SIMBOL  | NAMA  | FUNGSI  |
| 1  |   | Terminator  | Permulaan atau akhir program  |
| 2  |   | Garis alir  | Arah aliran program  |
| 3  |   | Preparation  | Proses inisiasi atau pemberian harga awal  |
| 4  |   | Proses  | Proses perhitungan atau proses pengolahan data  |
| 5  |   | Input atau output data  | Proses input atau output data  |
| 6  |   |  |  |  |  | Predefined proses (sub program)  | Permulaan sub program atau proses menjalankan sub program  |
|  |
| 7  |   | *Decision*  | Penyeleksian data menuju kelangkah selanjutnyA  |
| 8  |   | *On Page* *Connector*  | Menghubungkan bagian-bagian flowchart yang ada pada satu halaman  |
| 9  |   | *Off page connector*  | Penghubung  | bagian-bagian  |
|  |  |  | flowchart yang halaman berbeda  | berada pada  |

## 2.16 *Activity Diagram*

 *Activity Diagram* berfungsi untuk menggambarkan suatu proses atau aktivitas dari sebuah system pada perangkat lunak, bukan Tindakan actor pada system tersebut

(Rosa & Shalahuddin, 2013). Berikut adalah symbol-simbol activity diagram

###### Tabel 2. 4 Simbol *Activity Diagram*

|  |  |  |  |
| --- | --- | --- | --- |
| NO  | SIMBOL  | NAMA  | KETERANGAN  |
|   |   | *Activity*  | Memperlihatkan bagian masing-masing kelas antarmuka saling berinteraksi satu sama lain  |
|   |   |  *Action*  | State dari system yang mencerminkan eksekusi dari suatu aksi  |
|   |   | *Initial Node*  | Bagaimana objek dibentuk atau diawali  |
|   |   | *Activity Final* *Node*  | Bagaimana objek dibentuk dan diakhiri  |
|   |   | *Decision*  | Digunakan untuk menggambarkan suatu keputusan atau Tindakan yang harus  |
|  |  |  | diambil pada kondisi tertentu  |
|   |   | *Line Connector*  | Digunakan untuk menghubungkan satu symbol dengan symbol lainnya  |