JURNAL ILMIAH FLASH
Vol. 8 No. 1, Halaman: 36 - 42
Juni 2022

ANALISIS *CROSS-SITE SCRIPTING* (XSS) *INJECTION – REFLECTED* XSS *AND STORED* XSS MENGGGUNAKAN *FRAMEWORK* OWASP 10

Indah O. Laleb
Politeknik Negeri Kupang/Jurusan Teknik Elektro/Teknik Komputer dan Jaringan
* E-mail: indah.laleb@pnk.ac.id

Abstrak

Saat ini, banyak situs memanfaatkan sepenuhnya konten *client-server* (sebagian besar ditulis dalam JavaScript) untuk menembah pengalaman klien. Sebaliknya, tren ini juga telah memperluas keberadaan dan frekuensi *serangan cross-site scripting* (XSS). Keadaan seperti itu dapat muncul, misalnya, karena kurangnya kesadaran akan masalah keamanan oleh pengembang, atau karena kesalahan pemrograman yang disebabkan oleh kendala keuangan dan waktu. Penelitian ini mencakup dua pendekatan umum untuk menginjeksi kode berbahaya (*malicious code*) ke halaman web yang ditampilkan kepada pengguna yaitu Reflected XSS dan Stored XSS. Penelitian ini bertujuan untuk mengetahui cara kerja jenis serangan *XSS Reflected* dan *Stored XSS* serta menyediakan pencegahan atau tindakan preventif terhadap jenis serangan ini.

Kata kunci: *Cross-Site Scripting (XSS), XSS Injection, Reflected XSS, Stored XSS, OWASP*.

PENDAHULUAN

Saat ini, banyak situs memanfaatkan sepenuhnya konten *client-server* (sebagian besar ditulis dalam JavaScript) untuk menembah pengalaman klien. Sebaliknya, tren ini juga telah memperluas keberadaan dan frekuensi serangan cross-site scripting (XSS). Ketika kerentanan *cross-site scripting* tersedia dalam aplikasi web, musuh dapat menginjeksi scripting code ke halaman yang dihasilkan oleh aplikasi web (Fonseca, 2017). Keadaan seperti itu dapat muncul, misalnya, karena kurangnya kesadaran akan masalah keamanan oleh pengembang, atau karena kesalahan pemrograman yang disebabkan oleh kendala keuangan dan waktu.

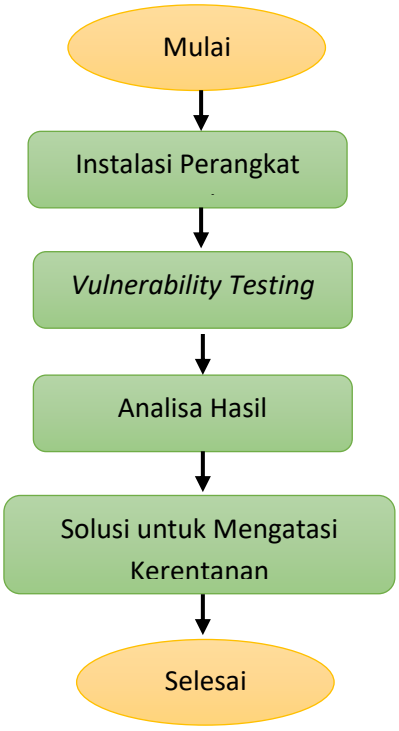
Penelitian ini mencakup mencakup dua pendekatan umum untuk menginjeksi kode berbahaya (*malicious code*) ke halaman web yang ditampilkan kepada pengguna. Pada teknik pertama, yaitu stored XSS, kode berbahaya akan terus disimpan oleh penyerang dalam sumber daya yang dikelola oleh aplikasi web, misalnya, database. Metode lain disebut *reflected XSS*. Metode ini hanya secara langsung mencerminkan scrip kepada pengguna tanpa menyimpannya didatabase . Salah satu tool yang digunakan dalam penelitian ini adalah OWASP Top 10 yang merupakan sebuah *vurnerable platform* yang digunakan oleh *developers* dan *security analyst* sebagai panduan untuk mencari kelemahan-kelemahan pada web *apps* yang mudah diserang dan harus segera disiasati (Ghozali et al., 2019).

Penelitian ini bertujuan untuk mengetahui cara kerja jenis serangan XSS Reflected dan Stored XSS serta menyediakan pencegahan atau tindakan preventif terhadap jenis serangan ini.

METODE PENELITIAN

Penelitian ini dilakukan dengan empat tahap yang meliputi instalasi perangkat lunak. vulnerability testing, analisa hasil dan memberikan solusi dari kerentanan sistem.

1. Instalasi perangkat lunak merupakan proses dimana akan dilakukan install beberapa tools yang akan digunakan dalam penelitian. Adapaun tools yang dipersiapkan adalah system operasi kali Linux dan metasploitable. Metasploitable digunakan untuk mengakses OWASP 10. Sedangkan Kali linux adalah salah satu system operasi yang digunakan untuk *penetration testing.*
2. Vurnerability testing merupakan proses untuk mengetahui kerentanan yang ada pada system.
3. Analisa hasil merupakan tahap dimana menampilkan hasil kerentanan system
4. Tahap terakhir adalah memberikan solusi dari kerentanan yang terjadi.



Gambar 1 Tahap Pengujian Kerentanan

HASIL DAN PEMBAHASAN

Chosen Vulnerability

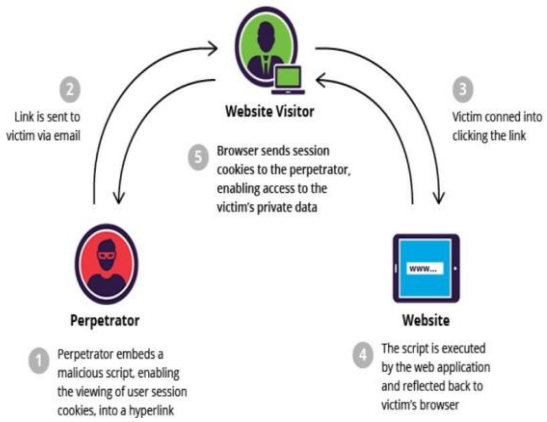
Penelitian ini menjelaskan dan memeriksa kerentanan situs web populer yang disebut Cross-site Scripting (XSS). Cross-website scripting (XSS) adalah salah satu kerentanan dalam keamanan komputer yang ditemukan dalam aplikasi web. XSS memberdayakan penyerang untuk menyuntikkan *script* pada sisi klien ke halaman situs web yang dilihat oleh pengguna. Kerentanan cross-site scripting dapat digunakan oleh musuh untuk melewati kontrol akses, misalnya, *same-origin policy.* Cross-site scripting berkontribusi sekitar 84% dari semua kerentanan keamanan yang diarsipkan oleh Symantec mulai tahun 2007. Organisasi bug bounty HackerOne pada tahun 2017 merinci bahwa XSS masih menjadi ancaman bagi aplikasi web.

Dampak XSS berfluktuasi dari gangguan yang tidak signifikan ke risiko keamanan berbahaya, tergantung pada pengaruh informasi yang diurus oleh *open website* dan gagasan tentang bantuan keamanan yang diaktualisasikan oleh pemilik situs web (Wikipedia, 2018).

Penjelasan *Vulnerability*

Dua jenis kerentanan XSS akan dibahas dalam penelitian ini yaitu *reflected XSS* dan *stored xss*.

1. *Reflected XSS* menurut Kals.et.al (2006), menegaskan bahwa *reflected XSS* dapat terjadi kapan saja pengguna mengirimkan input ke aplikasi yang kemudian dipantulkan kembali ke pengguna yang sama.
   Script diinjeksikan melalui koneksi, yang mengirimkan permintaan ke situs web rentan yang dapat mengeksekusi skrip berbahaya.
   Biasanya, kerentanan ini dihasilkan dari permintaan yang tidak cukup disanitasi oleh sistem yang memiliki probabilitas untuk memanipulasi fungsi web dan mengeksekusi skrip berbahaya. Proses reflected XSS digambarkan pada gambar 2.

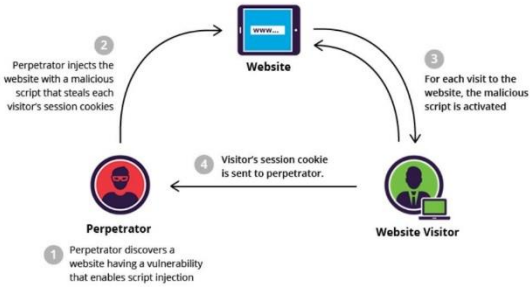


Gambar 2 Reflected XSS Attack
Source: https://www.incapsula.com/web-application-security/reflected-XSS-attacks.html

   Untuk memasukan tautan berbahaya, skrip berbahaya disematkan ke situs web pihak ketiga oleh pelaku (misalnya, email, bagian komentar di situs web atau media sosial).
   Tautan dimasukkan ke dalam anchor text atau anchor link yang digunakan untuk memprovokasi pengguna untuk mengklik tautan. Setelah itu, permintaan XSS akan dikirim ke situs web yang dieksploitasi, dan itu akan mencerminkan kembali serangan terhadap pengguna.
   Musuh akan berhasil menyerang ketika pengguna mengklik tautan yang disediakan oleh penyerang (Kals.et.al 2006).

2. *Stored XSS*
   Kals.et.al (2006) menyatakan bahwa store attack adalah salah satu serangan XSS dengan menyuntikkan kode berbahaya pada server target secara permanen, misalnya: database. Di bagian pesan dan kolom komentar, serangan terjadi ketika korban meminta informasi

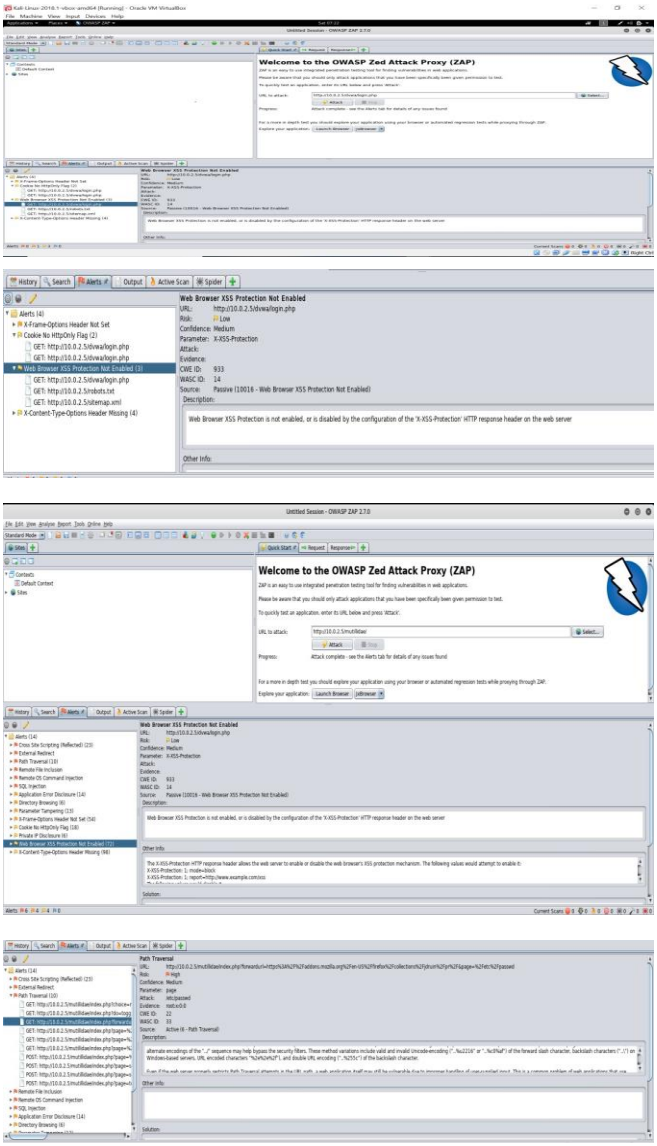
yang disimpan dari server kemudian dari server, mereka menerima skrip berbahaya.



Gambar 3 Stored XSS Attack
Source: https://www.incapsula.com/web-application-security/cross-site-scripting-XSS- attacks.html

Ada beberapa strategi untuk menemukan kerentanan dalam aplikasi web, salah satu alat ini disebut OWASP Zap. OWASP zap adalah alat untuk menemukan kerentanan secara otomatis di situs web.

Gambar 4 menjelaskan deskripsi kerentanan, OWASP ZAP menunjukkan jenis kerentanan dan jenis risiko (tinggi, sedang atau rendah). Aplikasi ini juga memberikan informasi tentang tingkat kepercayaan keberadaan kerentanan dan tempat di situs web yang telah disuntikkan. Dari Gambar 3, informasi tentang target juga tersedia di aplikasi ini (serangan mencoba untuk mendapatkan /etc/passwd pengguna).



Gambar 4 Scan Vulnerabilities

Vulnerabilities Analysis

Untuk menemukan kerentanan apa pun, beberapa teknik dapat digunakan: telusuri target dan temukan kode javascript ke dalam halaman. Metode lain adalah dengan mencoba menyuntikkan pada textbox, parameter, atau URL apa pun yang memiliki tampilan serupa seperti **http://attack.com/page.php?something=something (Incapsula, 2018) .**

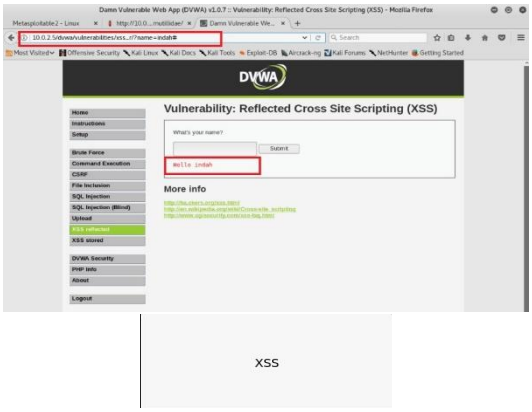
Pada Reflected XSS, metode ini adalah jenis non-persisten dan tidak disimpan dan hanya berfungsi jika target mengunjungi URL yang dibuat. Contoh sampel XSS yang dipantulkan adalah **http://attack.com/page.php?something=<Script>alert("XSS")</script>.**

Pada jenis serangan ini, ketika target menjalankan kode, kode berbahaya akan mengeksekusi pada

mesin mereka. Pelaku memasukkan dan mengeksekusi pada permintaan pencarian dengan sepotong kode ini:

<script type='text/javascript'>alert('XSS');</script>

yang menyebabkan: Akan ada pesan peringatan seperti "XSS" pada kueri (Gambar 5.).



Gambar 5 Langkah untuk Menemukan Kerentanan pada Web

Kode<script type='text/javascript'>alert('XSS');</script> tidak ditemukan akan ditampilkan di halaman website.

URL halaman http://ecommerce.com?q=<script type="text/javascript">alert('XSS'); </script>.

Jika pesan-pesan ini muncul di situs web, itu berarti situs web itu rentan. Setelah itu, penyerang membuat tautan dan menyuntikkan kode berbahaya ke dalam tautan, **http://forum.com?q=news<\script%20src="http://hackersite.com/authstealer.js** dan secara diam-diam mengirimkannya ke email.

Menggunakan Reflected XSS, penyerang dapat mencuri informasi penting seperti cookie dan membajak forum (Incapsula, 2018).

Sedangkan *stored XSS*, injeksi dapat dilakukan dengan memasukkan kode <script>alert("XSS")</script> pada fields atau textbox.

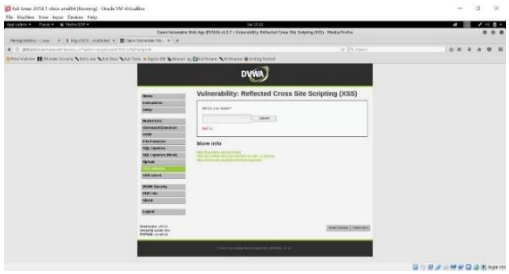
Saat menjalankan kode itu di situs web atau database yang rentan, komputer korban secara otomatis telah diinjeksi dan mencuri cookie sesi pengguna yang membuka akses ke penyerang untuk mencuri informasi sensitif seperti kartu kredit dan informasi pribadi. Suntikan semacam ini sangat berbahaya bagi system dan penguna.

Vulnerability Exploitation for Reflected XSS

Pada tulisan ini, website yang digunakan **http://10.0.2.5/dvwa/login.php** dari metasploitable2 dengan tingkat keamanan yang rendah dan sedang.

Ada beberapa langkah untuk mengeksploitasi reflected XSS.

1. Tingkat keamanan rendah / low-security level
   Jalankan script code --- <script>peringatan("XSS")</script> pada textbox. Setelah mengklik tombol kirim, perhatikan bahwa pesan "Halo" di bawah textbox dan nama tidak ditampilkan di website.



Gambar 6 Website Injection

Pada URL, penetrasi akan mengirim URL tersebut kepada siapa pun **(http://10.0.2.5/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E#)**, kode ini akan dieksekusi pada mesin mereka dan mendapatkan pesan "XSS". Pada Gambar 7, "Hello" diikuti oleh script code yang telah disuntikkan pada textbox.

```
<pre>
  Hello
  <script>alert("XSS")</script>
</pre>
```

Gambar 7 Website Code

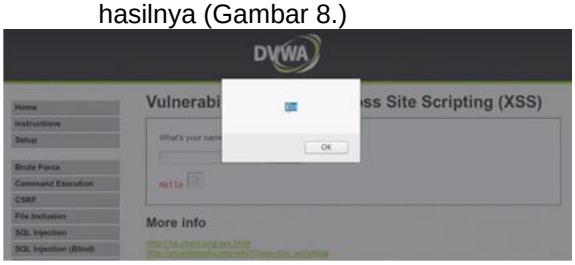
2. Medium security level
   Ketika tingkat keamanan diatur ke medium, maka <script>alert(“XSS")</script> tidak bekerja dengan baik. Kode ini tidak difilter dan tidak dieksekusi di situs web (tidak ditempatkan di dalam tag skrip seperti pada Gambar 7).

Menurut Owasp (2018), ada beberapa eksploitasi untuk muatan atau *payloads* yang dapat digunakan untuk melewati filter dan menyuntikkan kode berbahaya pada jenis situs web ini.

- Inject as a tag → xss link
- Inject as a javascript event
  → <IMG SRC=# onmouseover=”alert(‘XSS’)”>give an image and wrong source code
- Inject using image and give a wrong source code and get the script to run when there is an error <IMG SRC=/ onerror=”alert(‘XSS’)”></img>. Fro this experiment, this code has been used to inject the website, and the XSS message will appear as a result.

Dari percobaan ini, kode ini telah

digunakan untuk menyuntikkan situs web, dan XSS pesan akan muncul sebagai hasilnya (Gambar 8.)

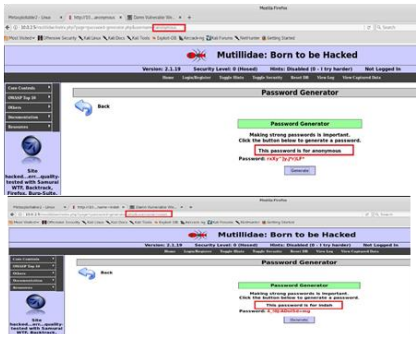


Gambar 8 Medium Reflected XSS Result

3. Advanced Security Level

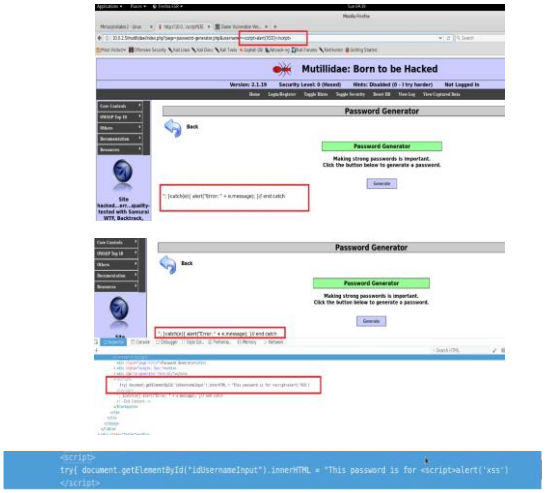
Dari jenis serangan ini, kode berbahaya akan disuntikkan ke dalam sistem menggunakan <script>alert('XSS')</script>.

Namun, karena keamanan situs web cukup tinggi, maka proses menemukan kerentanan harus dianalisis tidak hanya dengan menggunakan URL tetapi juga masuk lebih dalam untuk "memeriksa elemen" pada javascript. Pada bagian ini, injeksi akan dieksekusi pada Mutillidae OWASP top 10 Injection JavaScript password generator. Halaman ini akan menghasilkan kata sandi secara acak untuk pengguna anonim. Untuk menemukan kerentanan, kode disuntikkan pada URL, dan jika hasilnya di layar seperti pada Gambar 9, maka itu menunjukkan bahwa ada kerentanan di situs web ini.



Gambar 9 Contoh Kerentanan pada Mutilidae

Ketika penyerang mencoba untuk menempatkan <script>alert ('XSS')</script> pada URL, sistem memberikan pesan kesalahan yang berarti bahwa sistem ini cukup kuat untuk jenis injeksi (Gambar 10.).



Gambar 10 Pesan Error pada Advanced Reflected XSS

Mengenai eksploitasi menggunakan Reflected XSS di situs web ini, penyerang menyisipkan kode " ; alert('XSS'); di situs web. ";" berarti akan menutup "Kata sandi ini untuk" (Gambar 9.) dan menempatkan peringatan ('XSS') di baris berikutnya. Tambahan tambahan dari // adalah untuk mengomentari semua yang datang setelah " ; alert('XSS');(similar dengan injeksi SQL). Pada jenis injeksi ini, penyerang tidak menempatkan <script></script> lagi pada URL atau dalam skrip karena kode itu sudah disediakan pada sistem. Oleh karena itu, penyerang hanya menempatkan kode itu di URL secara langsung atau meletakkannya di skrip. Setelah menerima pesan XSS seperti yang ditampilkan pada Gambar 10, maka penyerang dapat membagikan URL yang disuntikkan ke situs web korban yang menjalankan URL tersebut.

Dari injeksi lanjutan ini, hal yang perlu dipertimbangkan adalah penyerang harus mengeksploitasi website yang rentan dengan melihat detail pada URL dan skrip.

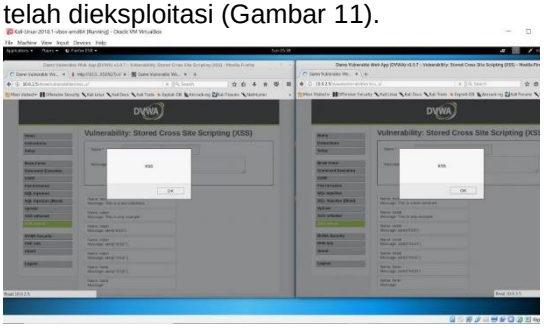
Vulnerability Exploitation for Stored XSS

Stored XSS adalah injeksi persisten dan biasanya disimpan di halaman atau DB. Perbedaan antara stored XSS dan reflected XSS adalah pada reflected XSS, penyerang berinteraksi dengan korban dengan mengirim URL, tetapi dalam stored XSS, kode injeksi dijalankan setiap kali halaman dimuat. Gambar 11 menunjukkan bahwa informasi apa pun yang dimasukkan ke dalam kolom komentar, halaman pengguna juga diperbarui karena penyerang tidak berinteraksi langsung dengan korban.

a) Low-security level

Penyerang dapat menyuntikkan database dengan menambahkan kode berbahaya<script>alert("XSS")</script>

pada field pesan, dan pesan XSS yang dihasilkan akan muncul di situs web. Setelah korban membuka halaman dan pesan XSS ditampilkan, itu berarti halaman telah dieksploitasi (Gambar 11).



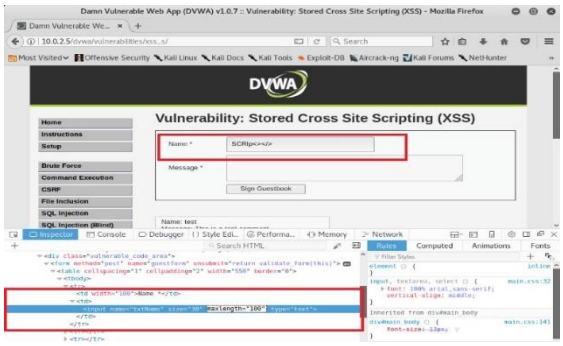
Gambar 11 The Execution of Stored XSS Injection on the attacker page (right) and victim page (left).

b) Medium security level

Pada tingkat ini, kode berbahaya akan disuntikkan pada name field bukan dengan menempatkan kutipan pada payload tetapi dengan memecahkan kode char dan berjalan sebagai String. Misalnya, dengan menambahkan kode <SCRIpT>(String.fromCharCode(120, 115, 115, 50))</ScRIPT> (Owasp, 2018).

Pada jenis kode ini, kode skrip mengetik dengan campuran antara huruf kapital dan huruf yang lebih rendah. Dengan melakukan ini, ia dapat melewati filter apa pun saat filter memeriksa skrip pada sistem.

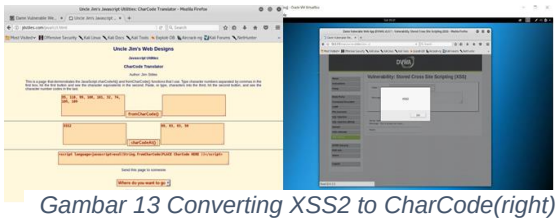
Karena kode berbahaya akan disuntikkan pada name field, panjang field juga harus ditingkatkan (berubah menjadi 100). Gambar 12. menggambarkan metode untuk mengubah panjang bidang dengan memeriksa kode skrip.



Gambar 12 Mengubah Length Name Field

Cara lain adalah dengan mengkonversi XSS ke CharCode seperti pada perintah dibawah ini:

<SCRIpT>alert(String.fromCharCode(88 , 83, 83, 50))</ScRIPT>



Gambar 13 Converting XSS2 to CharCode(right) and The Result of Stored XSS Injection(left)

Penelitian ini telah membahas injeksi XSS yang mencakup *reflected XSS* dan *stored XSS*. Beberapa teknik telah dibahas dalam tulisan ini untuk menemukan kerentanan termasuk menggunakan Owasp Zap dan secara manual menemukan dan menyuntikkan skrip. Pada reflected XSS, injeksi telah digunakan dalam tiga tingkat keamanan yang rendah, sedang dan tinggi. Sementara di XSS yang disimpan, tingkat keamanan rendah dan menengah sudah cukup untuk menunjukkan teknik untuk mengeksploitasi situs web.

Pada reflected XSS, penyerang berinteraksi dengan korban dengan mengirimkan URL kepada mereka. Setelah korban mengeksekusi URL, maka halaman web korban akan tereksploitasi. Sebaliknya, stored XSS tidak menghubungi pengguna karena penyerang dapat mengeksekusi kode dan halaman web korban dapat disuntikkan secara otomatis.

Reflected XSS dan stored XSS dapat berdampak buruk pada commerce dan hubungan antara klien.

Oleh karena itu, untuk mencegah dua serangan ini, pengembang harus membuat beberapa validasi ke field dengan menempatkan hanya huruf atau huruf kombinasi dan angka pada textbox atau field dan meminimalkan input pengguna pada halaman / HTML. Solusi lain untuk melawan XSS yang tersimpan adalah dengan menggunakan firewall aplikasi web pada sistem.

DAFTAR PUSTAKA

[1]. Fonseca, J., Vieira, M., & Madeira, H. (2007, December). Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks. In *Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on* (pp.
a. 365-372). IEEE.

[2]. Ghozali, B., Kusrini, & Sudarmawan. (2019). *Mendeteksi Kerentanan Keamanan Aplikasi Website Menggunakan Metode Owasp (Open Web Application Security Project) untuk Penilaian Risk Rating*. *January*. https://doi.org/10.24076/citec.2017v4i4.119

[3]. Incapsula, 2018. Retrieved from https://www.incapsula.com/web-application-

security/reflected-xss-attacks.html
[4]. Kals, S., Kirda, E., Kruegel, C., & Jovanovic, N. (2006, May). Secubat: a web vulnerability scanner. In *Proceedings of the 15th international conference on World Wide Web* (pp. 247-256). ACM.
[5]. OWASAP. (2018). Retrieved from https://www.owasp.org/index.php/OWASP_Top_Ten_Cheat_Sheet
[6]. Wikipedia. (2018). Retrieved from https://en.wikipedia.org/wiki/Cross-site_scripting