Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 5, No. 9, September 2021, hlm. 3745-3753 http://j-ptiik.ub.ac.id

Implementasi Cross Site Scripting Vulnerability Assessment Tools berdasarkan OWASP Code Review

Muhammad Isfa Hany[1], Adhitya Bhawiyuga[2], Ari Kusyanti[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]hanyisfa@student.ub.ac.id, [2]bhawiyuga@ub.ac.id, [3]ari@ub.ac.id

Abstrak

Serangan cross site scripting (XSS) merupakan salah satu kerentanan yang paling sering ditemukan dalam aplikasi web. Sayangnya, tidak semua pengembang aplikasi dan tim security fasih terhadap kerentanan web secara menyeluruh (Khan et al., 2017). OWASP Code Review merupakan dokumen tertulis yang menjelaskan mengenai kaidah, aturan, dan standar yang baik dalam analisis kode aplikasi web. Selain itu, proses vulnerability assessment juga dapat membantu penemuan kerentanan dengan waktu yang lebih efisien. Penelitian ini akan membangun suatu sistem aplikasi yang dapat melakukan proses vulnerability assessment berdasarkan kaidah OWASP Code Review. Dalam perancangannya, didapatkan tujuh pola ekspresi regular yang dapat membantu mengidentifikasi jenis pelanggaran yang dilakukan dalam potongan kode program dan dua pola ekspresi regular utama untuk menemukan kerentanan. Selain itu, dirancang pula lima algoritme pendukung guna memahami bagaimana sistem akan diimplementasikan. Sistem ini diimplementasikan dalam kerangka kerja Django dan telah diuji berdasarkan validitas temuan, penggunaan cpu, dan response time. Berdasarkan hasil pengujian yang dilakukan, sistem yang dibangun terbukti dua kali lebih baik dalam penemuan kerentanan cross site scripting

Kata kunci: OWASP, ekspresi regular, *cross site scripting*, *vulnerability assessment*

Abstract

Cross site scripting (XSS) attacks is one of the most discovered vulnerabilities in the web application. Unfortunately, not all software engineer and security engineering team fluent aganist all of the web vulnerabilities (Khan et al., 2017). OWASP Code review is a written document explaining about principles, rules, and standards about web application source code analysis. Furthermore, vulnerability assessment process can also aid in more efficient web application vulnerability discoveries. This research will also build a system that can perform vulnerability assessment according to OWASP Code Review. In the system design phase, there are seven regular expression patterns that can help to identify security violation from the chunk of source code and two main regular expressions patterns to find vulnerabilities. Moreover, there are five algorithm design in order to understands how the system will be implemented. The system is implemented with Django Framework and have been tested based on validity, cpu usage, and response time. According to the test result, the system built is better than discovering cross site scripting

Keywords: *OWASP, regular expression, cross site scripting, vulnerability assessment*

PENDAHULUAN

Berdasarkan data yang dikumpulkan platform *bug bounty* hackerone pada 2019, XSS berada diperingkat pertama sebagai *bug security* paling berpengaruh dan juga paling banyak diberikan bayaran pada 2019 (Hackerone, 2019). Hal ini senada dengan OWASP *Top Ten Web Application Security Risk* yang terus memposisikan XSS di sepuluh kerentanan paling berbahaya setiap riset dilakukan (Saad et al., 2017).

Vulnerability assessment adalah proses menentukan celah keamanan dari sistem yang berkemungkinan untuk diretas oleh hacker baik dari dalam maupun luar organisasi dengan tujuan mendapatkan data penting, keuntungan finansial, atau tindakan perlawanan lainnya (Umrao, Kaur, dan Gupta 2012). Masalah utama

dari vulnerability assessment adalah kurangnya pengetahuan tim security dari ancaman terbaru maupun ancaman lama (Khan et al., 2017).

OWASP — Open Web Application Security Project — sebagai organisasi keamanan web internasional mendokumentasikan proses pengujian dan code review yang dituliskan dalam buku elektronik bernama OWASP Web Security Testing Guide dan semenjak tahun 2006 proses code review dituliskan pada OWASP Code Review. Pada OWASP Code Review bagian pencegahan serangan Cross Site Scripting — Prevention Guide — terdapat delapan aturan mengenai pencegahan kerentanan Cross Site Scripting berjenis reflective dan stored serta ada satu bab khusus yang membahas mengenai DOM based XSS.

Sebelumnya, telah ada berbagai aplikasi yang melakukan proses vulnerability assessment kerentanan web cross site scripting dengan pendekatan analisis statis diantaranya Vulny Code Static Analysis dan insideVuln. Namun, pada pembuatannya tidak dilibatkan suatu standar ataupun metode tertentu sebagai dasar pengujiannya. Dengan demikian, diperlukan suatu penelitian yang melakukan pendekatan berbeda dengan menerapkan standar dan metode yang telah ada. Oleh sebab itu, peneliti akan membuat suatu tools vulnerability assessment guna menemukan kerentanan cross site scripting yang berdasarkan dengan kaidah kaidah OWASP Code Review.

PENELITIAN TERKAIT

Penelitian lainnya yang telah menggunakan OWASP Testing Guide dilakukan oleh Yumnun dan kawan kawan. Pada penelitian tersebut, dilakukan pengembangan dari OWASP Mobile Application Testing Guide. Penelitian ini lebih berfokus pada keamanan data yang terdapat pada file .apk dan juga berbasis android Penelitian ini mengimplementasikan proses decompiling serta melakukan analisa statis terhadap file apk yang telah diupload. Analisa statis adalah kegiatan pencarian celah tanpa melakukan proses debugging (Yumnun, et al., 2019). Analisis statis yang dilakukan diawal proses system development life cycle (SDLC) akan membantu untuk mengidentifikasi masalah dengan biaya yang lebih minim dan lebih efisien (Novikov, et al., 2017).

Penelitian mengenai XSS lainnya berjudul "Detecting Cross Site Scripting Vulnerabilities Introduced by HTML5" yang dilakukan oleh Dong membahas mengenai Cross Site Scripting kerentanan XSS pada sistem surat elektronik dan dikembangkan menggunakan java. Penelitian ini diimplementasikan dengan mengimpor Java Mail Development Kit, Mengumpulkan attack vector yang ada dan menyimpannya ke dalam repositori. Kunci utama dari penelitian ini adalah pendekatan untuk menginjeksi attack vector ke checkpoint yang telah ditentukan. Checkpoint yang diuji coba yaitu bagian judul, badan surat, judul lampiran, nama lampiran, dan nama pengirim.

Pada penelitian yang dilakukan oleh Sivanesan dan kawan kawan, telah dikembangkan suatu ekstensi browser dengan kurang lebih 10 sampel attack vector. Penelitian ini juga telah menganalisa pattern serangan dengan bentuk direct attack, multi format dan juga DOM based attack pattern (Sivanesan et al., 2018).

Penelitian yang dilakukan oleh Zhang berjudul "Cross-site scripting attack in social network — APIs" telah dilakukan studi sistematik dan analisis mengenai kerentanan XSS pada API sosial media dan dapat digunakan untuk mendeteksi berbagai isu mengenai respon API, inkonsistensi penanganan masukan pengguna dan kesalahan respon API. Ia merancang sebuah tools untuk menganalisa kerentanan desain dan implementasi Restful API pada jaringan sosial (Zhang et al., 2018)

DASAR TEORI

3.1. Cross Site Scripting

Cross Site Scripting (XSS) pada dasarnya adalah serangan code injection dimana script dimasukkan sebagai kode berbahaya. script yang telah diinjeksi ini bisa dibuat dengan nyaris semua client side script seperti javascript, vbscript, dan actionscript. Kerentanan ini memungkinkan berbagai ancaman keamanan. Hal ini termasuk pencurian identitas, akses informasi sensitif dan terbatas, mengubah fungsionalitas web, dan juga ancaman denial of service. Kerentanan ini hadir diakibatkan oleh kurangnya sanitasi masukan yang disiapkan sistem sehingga tempat input tersebut menjadi tempat dimasukkannya script. Hal ini menjadi lebih berbahaya saat munculnya teknologi AJAX dan web 2.0 (malviya, 2013). XSS dibagi menjadi tiga yaitu Reflected XSS, Stored XSS,

dan DOM XSS.

3.2. Vulnerability Assessment

Vulnerability assessment adalah proses menentukan celah keamanan dari sistem yang berkemungkinan untuk diretas oleh hacker baik dari dalam maupun luar organisasi dengan tujuan mendapatkan data penting, keuntungan finansial, atau tindakan perlawanan lainnya (Umrao, Kaur, dan Gupta 2012). Masalah utama dari vulnerability assessment adalah kurangnya pengetahuan tim security dari ancaman terbaru maupun lama (Khan et al., 2017).

3.3. Analisis Statis

Istilah analisis statis mengacu pada setiap proses penilaian *source code* tanpa melakukan eksekusi (Delev et al., 2017)(Yumnun et al. 2019). analisis statis juga atau yang biasa dikenal juga sebagai analisis sumber kode biasanya dilakukan sebagai bagian dari *code review* dan dilakukan pada tahap implementasi *System Development Life Cycle* (SLDC). Analisis Statis biasanya mengacu pada proses menjalankan alat yang mendeteksi kerentanan yang berjenis statis dari sumber kode dengan menggunakan teknik seperti *Taint Analysis* atau *Data Flow Analysis.*

3.4. OWASP Code Review

OWASP Code Review merupakan dokumen pengulasan kode aplikasi web yang dibuat oleh lembaga independen mengenai keamanan web OWASP. OWASP Code Review pada awalnya merupakan bagian dari OWASP Web Security Testing Guide tetapi semenjak 2006 mulai dibuat proyek pembuatan dokumen mandiri mengenai pengulasan kode ini karena dianggap cakupannya yang luas dan perlu difokuskan pada satu dokumen tersendiri.

3.5. Ekspresi Regular (RegEx)

Ekspresi Regular adalah suatu pola yang menjelaskan serangkaian huruf, digit, titik, garis bawah, tanda persen, dan tanda hubung (Tania et al., 2017). Regular expression telah dimanfaatkan secara luas di berbagai kebutuhan pemrograman seperti pencarian berbagai macam teks (Larson et al., 2016).

PERANCANGAN

3.1. Perancangan Antarmuka

Gambar 1. Antarmuka menu utama (8pt, ditengah)

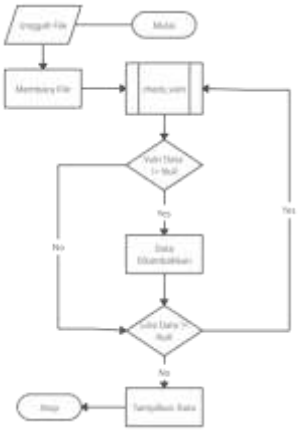
Perancangan antarmuka adalah tahap pembuatan rancangan tampilan awal web yang akan berinteraksi dengan pengguna. Ada dua tampilan web yang dibuat pada penelitian ini, yaitu tampilan menu utama dan tampilan menu laporan. Gambar 4.1 merupakan rancangan antarmuka untuk menu utama dan Gambar 4.2 merupakan rancangan antarmuka untuk menu laporan.

Gambar 2. Antarmuka laporan

3.2. Perancangan Algoritme

Perancangan algoritme yang dilakukan memiliki maksud untuk memberi gambaran mengenai alur proses dari sistem yang dibangun. Algoritme yang akan digunakan terdiri dari satu algoritme sistem dan empat algoritme fungsi. Algoritme sistem dibangun untuk memahami alur proses sistem yang dibangun. Algoritme fungsi yang dibangun yaitu algoritme *check_vuln*, algoritme *vuln*, algoritme *security_check*, dan algoritme *skipping.*

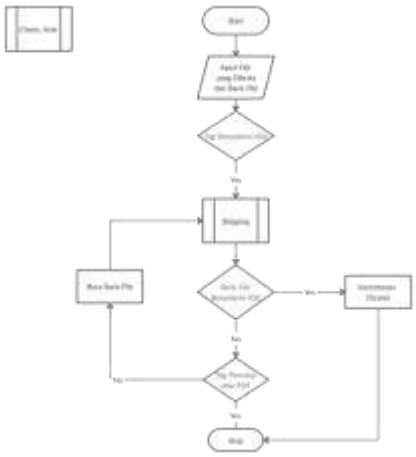
a. Algoritme Sistem



Gambar 3. Flowchart algoritme sistem

Gambar 3 dimulai dengan pengunggahan file yang akan dicek kerentanannya. Setelah itu, sistem akan membuka file tersebut dan membaca baris pertamanya. Sistem akan melakukan pengecekan kerentanan pada baris tersebut dengan memanggil fungsi check_vuln. jika data kerentanan ditemukan, maka data tersebut akan ditambahkan. Jika tidak ditemukan, maka akan dilakukan pengecekan apakah baris tersebut berada diakhir baris. Jika tidak berada diakhir baris, maka sistem akan melakukan pengecekan kerentan kembali pada baris *file* selanjutnya. Jika berada diakhir baris, maka data akan ditampilkan dan sistem akan berhenti berjalan.

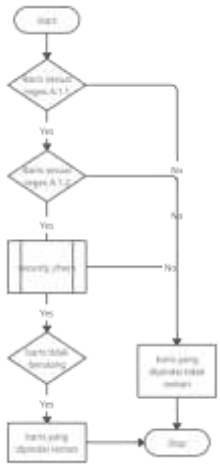
b. Algoritme Check_Vuln



Gambar 4. Flowchart algoritme sistem

Pada diagram alir pada Gambar 4, algoritme ini dimulai dengan masukan file yang telah dibuka sebelumnya dan juga baris file yang dikirimkan. Setelah itu, dilakukan pengecekan kerentanan XSS. Jika ditemukan kerentanan XSS, sistem akan menjalankan fungsi skipping. Fungsi ini secara garis besar berfungsi untuk melewatkan tag yang berada dalam tag atau biasa kita kenal dengan tag bersangkar. Setelah itu, sistem akan mengecek apakah terdapat baris file didalam tag yang berpotensi memiliki kerentanan XSS,

c. Algoritme Vuln



Gambar 5. Flowchart algoritme sistem

Diagram alir pada Gambar 5 merupakan fungsi vuln yang digunakan pada algoritme fungsi check_vuln. Fungsi ini digunakan untuk melakukan pengecekan kerentanan secara lebih spesifik. Fungsi ini digunakan untuk melakukan pengecekan terhadap baris baris pada tag dari file yang berpotensi terkena serangan pada fungsi sebelumnya. Fungsi ini diawali dengan pengecekan baris apakah sesuai dengan pattern ekspresi regular yang telah dibuat. pengecekan pertama mengecek apakah baris mengandung baris yang mengeluarkan keluaran pada web. seleksi kondisi ekspresi regular kedua melakukan pengecekan apakah baris yang dicek menggunakan masukan dari variabel variabel yang mungkin berasal dari pengguna seperti $_GET, $_POST, ataupun $_COOKIES. Jika salah satu dari kondisi tersebut salah, maka akan dianggap tidak rentan. Tetapi jika keduanya rentan, akan dilakukan pengecekan keamanan selanjutnya yang melakukan pengecekan pada

fungsi security_check. Jika kembalian fungsi security_check rentan, maka akan dilanjutkan ke seleksi kondisi berikutnya yang melihat apakah baris tersebut pernah ditemukan sebelumnya atau tidak. Jika tidak pernah ditemukan sebelumnya, maka baris tersebut dianggap rentan.

d. Algoritme Security Check

Gambar 6. Flowchart algoritme sistem

Pada Gambar 6 terdapat diagram alir fungsi Security_Check yang digunakan pada fungsi vuln. Fungsi ini terdiri dari beberapa seleksi kondisi yang mencakup seleksi kondisi pengenalan aturan keberapa yang mendekati baris yang terindikasi rentan dan juga apa yang harus dilakukan setelahnya. Dapat dilihat, bahwa ada 4 jenis level atau tingkatan yang mengkategorikan metode enkoding seperti apa yang harus dilakukan. Jika masukan yang diberikan untuk level = 0, maka sudah dipastikan bahwa baris itu rentan. Jika masukan yang diberikan adalah level = 1 dan tidak mengandung fungsi htmlentities() ataupun htmlspecialchars(), maka baris tersebut juga rentan. Jika masukan yang diberikan adalah level = 2 dan tidak mengandung json_encode() atau htmlentities dan htmlspecialchars, maka baris tersebut juga rentan. Selain itu, jika masukan yang diterima adalah level = -1 dan tidak mengandung urlencode(), maka baris tersebut dapat dipastikan rentan. Selain itu semua, baris diindikasikan tidak rentan.

e. Algoritme Skipping

Gambar 7 merupakan representasi diagram alir algoritme scanning. Algoritme ini dimulai dengan memasukkan tag buka dan tag tutup. tag tag yang dimasukkan tersebut akan dihapus pada list tag berbentuk ekspresi reguler. Setelah itu, dimasukkan baris yang akan dibaca dan dilakukan iterasi. iterasi yang dilakukan adalah pengecekan baris yang mengandung tag pada list. jika ditemukan, akan dicek kembali apakah baris tersebut bernilai Null. Jika tidak, maka baris akan mengecek tag tutup. jika tidak ada tag tutup, maka akan dilakukan iterasi hingga ditemukan tag tutup pada list. Saat ditemukan tag tutup ataupun baris bernilai null, maka fungsi dihentikan. fungsi akan dihentikan lebih awal jika baris tidak mengandung tag yang ada pada list buka.

Gambar 7. Flowchart algoritme sistem

3.3. Perancangan Ekspresi Regular

Pada perancangan ekspresi regular, Ada 13 ekspresi regular yang diimplementasikan. Ekspresi regular ini dibagi menjadi tujuh jenis dengan sepuluh ekspresi regular untuk identifikasi jenis aturan yang dilanggar dalam OWASP. Selain itu, ada satu jenis ekspresi regular dengan dua ekspresi regular untuk pencarian kerentanan awal. Selain itu, ada ekspresi regular dengan fungsi identifikasi petik pada sumber kode.

Tabel 1. Perancangan Ekspresi Regular

Kegunaan	Ekspresi Regular

Pencarian Kerentanan Utama	^(?=(.*echo)|(.*print)|(.*printf)|(.*vprintf)|(.*trigger_error)|(.*user_error)|(.*odbc_result_all)|(.*if_htmltbl_result)|(.*die)|(.*exit)|(.*var_dump)|(.*nl2br))(.*)$ ^(?=(.*\\$_GET)|(.*\\$_FILES)|(.*\\$_POST)|(.*\\$_REQUEST)|(.*\\$_COOKIES)|(.*\\$_SESSION)|(.*\\$(?!this|e-)[a-zA-Z0-9_]*))(.*)$
Petik	'.*?<\?.*?\?>.*?'
Aturan Nol	<script>|<.*? .*?=\".*?='<\?.*?\?>.*?'\" <[A-Za-z]+.*?.+=[A-Za-z]+.*?/> <!-- <<\?.*/>
Aturan Satu	<[^/|\?][A-Za-z]+>
Aturan Dua	<[A-Za-z]+.*?[A-Za-z]+=.*?<\?
Aturan Tiga	^(?=(.*?='<\?.*?\?>)|(<.*? .*?=\".*?='<\?.*?\?>.*?'\")|(.*?\(.*?'.*?<\?.*?\?>))(.*)$
Aturan Empat	<style>|span style
Aturan Lima	<a href=.+<\?.+\?>.+>
Aturan Tujuh	javascript:.*?<\?

Aturan aturan ini dimaksudkan untuk mencari potensi masukan pengguna baik pada html, css, ataupun javascript yang menyebabkan munculnya kerentanan XSS.

IMPLEMENTASI DAN PENGUJIAN

Sistem yang dibangun diimplementasikan dan diuji pada sistem operasi linux 64 bit dengan processor Core i7 7th gen serta RAM 8 GB. Selain itu, ada N perangkat lunak yang digunakan seperti yang dapat dilihat pada Tabel 2.

Tabel 2. Perangkat Lunak

No	Perangkat Lunak	Deskripsi
1	Linux	Sistem Operasi
2	Django	Framework Backend
3	wsgi	Web Server
4	dropzone.js	Library
5	bootstrap	Framework Frontend
6	Visual Studio Code	Text Editor
7	Mozilla Firefox	Browser

Selain itu, Algoritme diimplementasikan sesuai dengan hasil perancangan yang telah dibuat. Algoritme dituliskan dengan bahasa pemrograman python dan kerangka kerja Django.

Sistem yang dibangun akan melalui tahapan tahapan pengujian yang telah disiapkan oleh penulis yaitu pengujian validitas, response time, dan juga penggunaan CPU yang dapat dilihat pada Tabel 3

Tabel 3. Jenis Pengujian

No	Jenis Pengujian	Tujuan Pengujian
1	Pengujian Validitas	Mengetahui validitas dari temuan sistem yang dibangun jika dibandingkan dengan aplikasi sumber terbuka lainnya
2	Pengujian Response Time	Mendapatkan rata rata kecepatan aplikasi yang diuji saat dijalankan dengan berbagai kondisi
3	Pengujian CPU Usage	Mengetahui nilai dari penggunaan CPU.

PEMBAHASAN

Sistem yang dibangun diimplementasikan sesuai dengan perancangan yang telah dituliskan sebelumnya. Sistem mengimplementasikan lima algoritme dengan tujuh jenis ekspresi regular untuk pencarian pola pada sumber kode. Selain itu, pengujian yang dilakukan terdiri dari pengujian pengujian yang ada pada Tabel 3.

Pengujian pertama adalah pengujian validitas. Pada pengujian ini, sistem yang dibangun dibandingkan hasil pengujiannya dengan dua aplikasi analisis statis sumber terbuka lainnya yaitu InsideVuln dan juga

Vulny. Pengujian ini juga penggunaan dummy code dikarenakan sulitnya menemukan sumber kode yang mencakup semua jenis kerentanan yang ada pad OWASP *Code Review*.

Setelah dilakukan pengujian validitas, ditemukan bahwa sistem yang dibangun dapat mendeteksi kerentanan dua kali lebih variatif sehingga hasil yang didapatkan juga lebih banyak. Hal ini terjadi karena sistem yang dibangun tidak mempercayai seutuhnya pada enkoding bawaan bahasa PHP Hypertext Preprocessor yaitu htmlencode() dan htmlspecialchars(). Sistem yang dibangun juga mengacu pada jsonencode() untuk enkripsi tingkat lanjutnya dan juga ada bagian bagian yang dianggap selalu rentan meski sudah menggunakan enkoding.

Pengujian lainnya yang dilakukan adalah pengujian Response Time, Pengujian ini dilakukan sepuluh aplikasi yang didesain khusus dengan kerentanan bahasa pemrograman PHP Hypertext Preprocessor antara lain DVWA, xxed, XSS-sploits, dan php sploits. Dari hasil pengujian ini, didapatkan bahwa aplikasi yang dibangun lebih cepat pada 8 dari 10 pengujian. Hasil pengujian dapat dilihat pada Tabel 4.

Tabel 4. Pengujian *Response Time*

No	*Response Time(ms)*		
	Aplikasi Riset	***Inside Vuln***	***Vulny***
1	38.8165	347.498	105.3623
2	4.3198	50.7142	49.7674
3	6.5905	53.0865	28.3457
4	135.7298	142.4098	90.9213
5	20838.454	166297.148	66770.537
6	3.8467	40.3683	47.0793
7	8.6826-	83.6625	50.3948
8	34.1466	76.1861	52.934
9	38.3157	69.5757	67.0144
10	64831.2327	21433.7339	1812.388

Pengujian terakhir yang dilakukan adalah pengujian CPU *Usage*. Sistem yang dibangun pada penelitian lebih besar penggunaan CPU nya hingga 25% dibanding dengan sistem aplikasi *vulnerability assessment* jenis analisis statis lainnya. Hal ini memiliki sisi baik dan buruknya secara bersamaan kerena dengan tingginya penggunaan CPU, sistem yang dibangun berarti tidak membuang sumber daya yang ada dan menukarnya dengan kecepatan pendeteksian. Namun, memiliki kekurangan yaitu berimbas pada penggunaan CPU secara menyeluruh.

KESIMPULAN

Berdasarkan implementasi, pengujian, dan analisis Cross Site Scripting Vulnerability Assessement Tools berdasarkan OWASP Code Review maka peneliti mengambil beberapa kesimpulan yaitu:

1. OWASP Code Review Cross Site Scripting dapat diimplementasikan dengan pendekatan analisis statis berjenis Taint Analysis yang bertujuan untuk mencari masukan end user dan juga variabel yang tidak tersanitasi.
2. Aplikasi yang dibangun dapat mendeteksi kerentanan dua kali lebih banyak dan variatif dibandingkan dengan aplikasi analisis statis sumber terbuka lainnya yang dijadikan sebagai pembanding.
3. Berdasarkan cpu usage dan juga response time dari hasil pengujian cross site scripting vulnerability assessment tools berdasarkan OWASP Code Review, diketahui bahwa aplikasi hasil riset lebih cepat dalam pengujian pada 8 dari 10 pengujian. Hal ini dipengaruhi pula dengan konsumsi penggunaan cpu oleh aplikasi hasil riset yang terus mengoptimalkan penggunaan cpu hingga 100%.

DAFTAR PUSTAKA

Conkilin, L., Robinson, G., Code Review Guide 2.1 ed.:OWASP.

Di Biase, M., Bruntink, M., & Bacchelli, A. (2016). A Security Perspective on Code Review: The Case of Chromium. 2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM). doi:10.1109/scam.2016.30

E. Larson and A. Kirk, “Generating evil test strings for regular expressions,” in Software Testing, Verification and Validation (ICST), 2016 IEEE

International Conference on. IEEE, 2016, pp. 309–319.

Farah, T., Shojol, M., Hassan, M., & Alam, D. (2016). Assessment of vulnerabilities of web applications of Bangladesh: A case study of XSS & CSRF. 2016 Sixth International Conference on Digital Information and Communication Technology and Its Applications (DICTAP). doi:10.1109/dictap.2016.7544004

Guowei Dong, Yan Zhang, Xin Wang, Peng Wang, & Liangkun Liu. (2014). Detecting cross site scripting vulnerabilities introduced by HTML5. 2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE). doi:10.1109/jcsse.2014.6841888

Hackerone, 2019 – Top 10 Most Impactful and Rewarded Vulnerability Types. (Tersedia di: https://www.hackerone.com/blog/hackerone-top-10-most-impactful-and-rewarded-vulnerability-types)[Diakses: 22 August 2019]

He, G., Zhang, Y., & Wu, X. (2013). Information Extraction of Forum Based on Regular Expression. 2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics. doi:10.1109/ihmsc.2013.175

Hu Junwei, Qin Yiqin, Zhang Wei, "Regular expression and its applications to web information extraction", Journal of Beijing Information Science and Technology University,Vol.26 No.6,Dec. 2011.

Jeffrey E.F. Friedl,Mastering Regular Expressions,Beijing. Publishing house of electronics industry.2011. pp. 80–145.

Kamongi, P.; Kotikela, S.; Kavi, K.; Gomathisankaran, M.; and Singhal, A. 2013. Vulcan: Vulnerability assessment framework for cloud computing. In Software Security and Reliability (SERE), 2013 IEEE 7th International Conference on, 218–226. IEEE.

Khan, Saad and Parkinson, Simon (2017) Towards Automated Vulnerability Assessment. In: 11th Scheduling and Planning Applications Workshop (SPARK), 19th June 2017, Carnegie Mellon University, Pittsburgh, USA.

Khera, Y., Kumar, D., Sujay, & Garg, N. (2019). Analysis and Impact of Vulnerability Assessment and Penetration Testing. 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). doi:10.1109/comitcon.2019.8862224

Malviya, V. K., Saurav, S., & Gupta, A. (2013). On Security Issues in Web Applications through Cross Site Scripting (XSS). 2013 20th Asia-Pacific Software Engineering Conference (APSEC). doi:10.1109/apsec.2013.85

Nelson, S., & Schumann, J. (2004). What makes a code review trustworthy? 37th Annual Hawaii International Conference on System Sciences, 2004. doi:10.1109/hicss.2004.1265711

Novikov, A. S., Ivutin, A. N., Troshina, A. G. & Vasiliev, S. N., 2017. The Approach to Finding Errors in Program Code Based on Static Analysis Methodology. Bar, IEEE.

OWASP (2017, March 16). About The Open Web Application Security Project. 29 December, 2016. <https://www.owasp.org/index.php/About_OWASP

Rigby, P. B. Cleary, F. Painchaud, M.A. Storey, and D. German. Open source peer review–lessons and recommendations for closed source. IEEE Software, 2012.

Saad, E., Meucci, M., Mitchell, R. 2020. Web Security Testing Guide(WSTG). 4.1 ed.:OWASP.

Sivanesan, A. P., Mathur, A., & Javaid, A. Y. (2018). A Google Chromium Browser Extension for Detecting XSS Attack in HTML5 Based Websites. 2018 IEEE International Conference on Electro/Information Technology (EIT). doi:10.1109/eit.2018.8500284

Stock, A. V. D., et al. 2017. OWASP Top Ten 2017: OWASP

Tania, D., K., Tama B., A., 2017. Implementation of Regular Expression (Regex) on Knowledge Management System. 2017 International Conference on

Data and Software Engineering (ICoDSE). doi:10.1109/icodse.2017.8285877

Umrao, S., Kaur, M., and Gupta, G., K., 2012. Vulnerability assessment and penetration testing. International Journal of Computer & Communication Technology, 3(6–8), pp. 71–74.

Yumnun, L. H., Kusyanti, A., Kartikasari D., P., 2019. Implementasi OWASP Mobile Security Testing Guide Untuk Pengujian Keamanan Pada Aplikasi Android. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 3(11), pp. 10579-10585

Zhang Jing, Zhang Yan, "Regular Expression and Application in Information Extraction",Computer Knowledge and Technology, 2009,5(15) : 3867— 3868.

Zhang, Y., Liu, Q., Luo, Q., & Wang, X. (2014). XAS: Cross-API scripting attacks in social ecosystems. Science China Information Sciences, 58(1), 1–14. doi:10.1007/s11432-014-5145-1