Jurnal Teknik Informatika (JUTIF)
Vol. 4, No. 6, December 2023, pp. 1479-1486
DOI: https://doi.org/10.52436/1.jutif.2023.4.6.778
p-ISSN: 2723-3863
e-ISSN: 2723-3871

EFFECTIVENESS OF SECURITY THROUGH OBSCURITY METHODS TO AVOID WEB APPLICATION VULNERABILITY SCANNERS

Azis Kurniawan*[1], Kalamullah Ramli[2]

[1,2]Faculty of Engineering, Universitas Indonesia
Email: [1]azis.kurniawan@ui.ac.id, [2]kalamullah.ramli@ui.ac.id

(Article received: December 21, 2022; Revision: January 2, 2023; published: December 23, 2023)

Abstract

The concept of security through obscurity is not recommended by the National Institute of Standards and Technology (NIST) as a form of system security. Basically this concept hides assets as difficult as possible so that it is not easy for attackers to find them, so that it can be used to avoid vulnerability scanner applications that are widely used by attackers to find out web system weaknesses. This research was conducted by modifying the web application firewall (WAF) and testing using the SQLMap and OWASP Zed Attack Proxy (ZAP) vulnerability scanner applications. The results of the study show that SQLMap takes up to 1238 times longer to complete a scan on a modified web application firewall than without modification, while OWASP ZAP cannot complete a scan on the same treatment. Thus the concept of security through obscurity can be applied to web security to extend vulnerability scanning time.

Keywords: *security through obscurity, web application firewall, web security.*

EFEKTIVITAS METODE *SECURITY THROUGH OBSCURITY* UNTUK MENGHINDARI APLIKASI PEMINDAI KERENTANAN

Abstrak

Konsep *security through obscurity* tidak direkomendasikan oleh National Institute of Standards and Technology (NIST) sebagai suatu bentuk pengamanan sistem. Pada dasarnya konsep ini menyembunyikan aset sesulit mungkin agar penyerang tidak mudah untuk menemukannya, sehingga dapat digunakan dalam menghindari aplikasi pemindai kerentanan yang banyak digunakan penyerang untuk mengetahui kelemahan sistem web. Penelitian ini dilakukan dengan memodifikasi *web application firewall* (WAF) serta dilakukan pengujian menggunakan aplikasi pemindai kerentanan SQLMap dan OWASP Zed Attack Proxy (ZAP). Hasil dari penelitian memperlihatkan SQLMap membutuhkan waktu hingga 1238 kali lebih lama menyelesaikan pemindaian pada modifikasi *web application firewall* dibandingkan dengan tanpa modifikasi, sedangkan OWASP ZAP tidak dapat menyelesaikan pemindaian pada perlakuan yang sama. Dengan demikian konsep *security through obscurity* dapat diterapkan pada keamanan web untuk memperlama waktu pemindaian kerentanan.

Kata kunci: *firewall aplikasi web, keamanan web, security through obscurity*.

1. PENDAHULUAN

Kerentanan web semakin bertambah setiap waktunya. Sampai saat ini, informasi yang tersedia saat kerentanan baru diterbitkan dinilai secara manual oleh para ahli menggunakan vektor dan skor *Common Vulnerability Scoring System* (CVSS) [1]. Tercatat pada 17 Desember 2022 terdapat 191314 kerentanan yang dikumpulkan oleh CVSS pada seluruh area [2]. Ukuran baku kategorisasi kerentanan juga dibentuk dengan tujian menjadi ukuran standar menganalisis dan mengukur keparahan kerentanan [3]. CVSS adalah standar *de-facto* untuk mengukur keparahan kerentanan. Dari berbagai standar penilaian kerentanan yang ada tersebut, seluruhnya berdasarkan pada kerangka risiko dari kerentanan yang ditemukan.

Dengan jumlah kerentanan yang sedemikian banyaknya akan sangat sulit untuk memeriksa berapa banyak kerentanan yang ada pada sistem, maka dalam rangka memudahkan mencari kerentanan, penyerang website akan memindai seluruh rentang nomor *port* yang ada sebelum menemukan kerentanan, mengurangi *attack surface* yang tersedia, dengan tujuan meminimalkan risiko [4]. Meningkatnya jumlah dan ukuran jaringan dan sistem komputer, kerentanannya juga meningkat. Melindungi aplikasi web menjadi semakin relevan

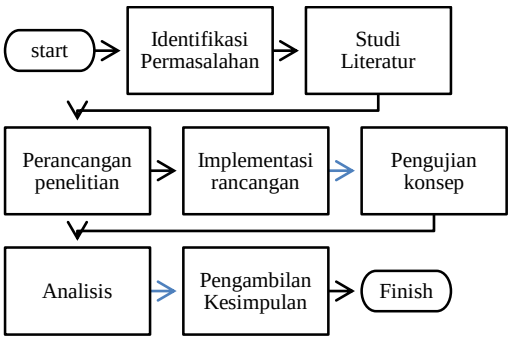
karena sebagian besar transaksi dilakukan melalui internet. Perangkat keamanan tradisional mengontrol serangan di tingkat jaringan, tetapi serangan web modern terjadi melalui protokol HTTP di tingkat aplikasi [5].

Oleh karenanya, untuk menemukan celah kerentanan dibuatlah aplikasi pemindai kerentanan yang tujuannya membantu mendeteksi kerentanan secara otomatis dalam aplikasi dan jaringan berbasis web [6]. Salah satu contoh penggunaan aplikasi pemindai kerentanan adalah penelitian yang dilakukan oleh Izumi dan Widiasari yang melakukan pemindaian kerentanan pada sistem web [7]. Aplikasi ini selain digunakan oleh pemilik situs web untuk mencari celah kerentanan, juga digunakan oleh penyerang untuk menemukan celah kerentanan yang ada. Sehingga dengan banyaknya aplikasi ini justru akan meningkatkan risiko pada situs web.

Untuk menurunkan risiko pemindaian, dibutuhkan suatu cara agar aplikasi pemindai kerentanan ini tidak dapat bekerja dengan semestinya. Penulis mengusulkan metode *security through obscurity* sebagai salah satu teknik yang akan menyulitkan aplikasi pemindai kerentanan melakukan tugas dengan semestinya. Teknik ini merupakan suatu konsep menyembunyikan aset sesulit mungkin sehingga penyerang tidak mudah untuk menemukannya. Prinsip ini pada awalnya diusulkan untuk mengurangi risiko ancaman dan dimaksudkan untuk mendapatkan waktu tambahan sementara dilakukan langkah-langkah perbaikan guna menghilangkan ancaman [8]. Oleh karena itu, penelitian ini dilakukan agar sistem web dapat terlindungi dari pemindaian yang dilakukan aplikasi pemindai kerentanan.

2. METODE PENELITIAN

Tahapan yang dilakukan dalam penelitian adalah sebagai berikut seperti ditunjukkan pada Gambar 1. Alur Penelitian:

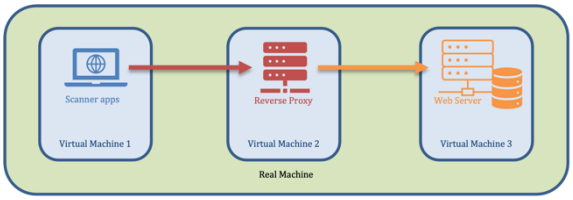


Gambar 1. Alur Penelitian

1. Identifikasi permasalahan mengenai aplikasi pemindai kerentanan yang meningkatkan risiko keamanan web.
2. Studi literatur mengenai permasalahan yang diidentifikasi.
3. Perancangan penelitian dengan modifikasi teknik respon pada *web application firewall* agar mempersulit aplikasi pemindai kerentanan untuk melakukan pekerjaannya.
4. Implementasi rancangan pada lingkungan *virtual* untuk mendapatkan hasil yang tidak terpengaruh oleh sistem dari luar ruang lingkup pengujian.
5. Pengujian konsep dengan melakukan pemindaian menggunakan aplikasi SQLMap dan OWASP ZAP dan mengukur waktu yang dibutuhkan aplikasi dalam menyelesaikan pemindaian.
6. Analisis dilakukan pada hasil penelitian dengan membandingkan 3 (tiga) kondisi sebagai berikut:
   a. Percobaan serangan pada target tanpa menggunakan *web application firewall*,
   b. Percobaan serangan pada target yang dilindungi *web application firewall* tanpa modifikasi,
   c. Percobaan serangan pada target yang dilindungi dengan *web application firewall* yang dimodifikasi.
7. Penarikan simpulan mengenai efektivitas modifikasi *web application firewall* dalam menghindari pemindaian yang dilakukan oleh aplikasi pemindai kerentanan.

2.1. Rancangan Sistem

Untuk melakukan evaluasi dalam penelitian ini terkait dengan arsitektur yang digunakan untuk melakukan percobaan ditampilkan pada Gambar 2 sebagai berikut:



Gambar 2. Arsitektur Infrastruktur Penelitian

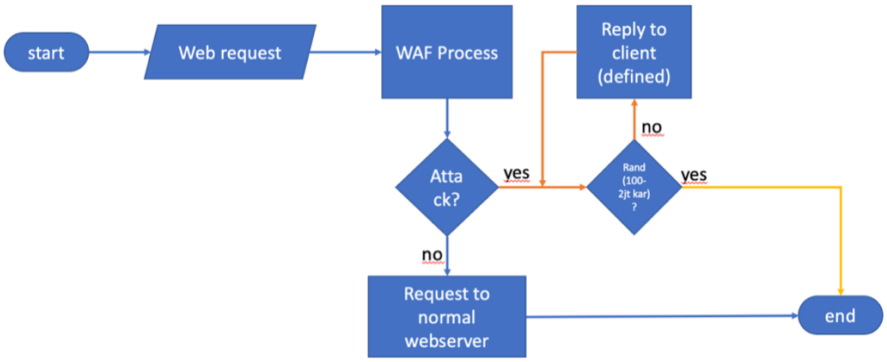
Terdapat 3 (tiga) buah *virtual machine* yang akan bertindak sebagai perangkat independen dan terhubung pada jaringan dengan peruntukan berikut:

a. VM1 : Mesin yang bertindak sebagai penyerang, aplikasi yang digunakan adalah SQLMap 1.6 dan OWASP ZAP 2.11 yang berjalan pada sistem operasi Kali Linux..
b. VM2 : Mesin yang bertindak sebagai WAF, aplikasi yang digunakan adalah NGINX (diinstall sebagai *reverse proxy* dengan ModSecurity), PHP 7.4, MariaDB 10.3.34-MariaDB-0ubuntu0.20.04.1, yang berjalan pada sistem operasi Ubuntu Server 20.04 LTS.
c. VM3 : Mesin target yang diserang, aplikasi yang digunakan adalah NGINX (web server), PHP 7.4, MariaDB 10.3.34-MariaDB-

0ubuntu0.20.04.1, yang berjalan pada sistem operasi Ubuntu Server 20.04 LTS. Selain itu juga ditambahkan dengan aplikasi *login* sederhana yang memiliki kerentanan berupa *SQL Injection*.

2.2. Modifikasi *Web Application Firewall*

Web Application Firewall (*WAF*) yang diimplementasikan pada proses penelitian ini menggunakan NGINX dengan ModSecurity. Modul *signature WAF* yang diimplementasikan adalah *OWASP ModSecurity Core Rule Set*, yaitu *signature* standar dari ModSecurity. Desain dari model modifikasi yang dilakukan dalam penelitian ini ditunjukkan pada Gambar 3.



Gambar 3. Modifikasi Respon Pada *Web Application Firewall*

1. *Request* akses kepada website diterima oleh *WAF*.
2. *WAF* akan melakukan proses pemeriksaan terhadap *signature* yang sudah didefinisikan.
3. Apabila *WAF* menyatakan bukan merupakan serangan, maka *request* akan diteruskan kepada *web server* dan setelah *client* menerima jawaban dari *web sever* maka koneksi akan dinyatakan selesai.
4. Apabila *WAF* menyatakan merupakan serangan, maka akan diberikan respon berupa karakter acak sebanyak 100 hingga 2.000.000 karakter (secara acak) yang juga ditambahkan dengan *string* respon web yang rentan, serta dilakukan perulangan juga secara acak.
5. Apabila *client* sudah menerima respon tersebut, maka koneksi dinyatakan selesai.

3. HASIL DAN PEMBAHASAN

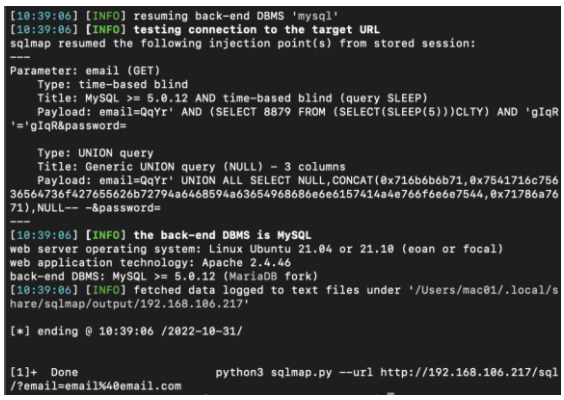
Hasil yang didapatkan pada penelitian ini adalah sebagai berikut.

3.1. Tanpa *Web Application Firewall*

Percobaan yang dilakukan pada perlakuan pertama yaitu dengan melakukan pemindaian aplikasi web dengan menonaktifkan *WAF*. Percobaan dilakukan dengan menggunakan 2 (dua) Aplikasi pemindai kerentanan yaitu SQLMap (khusus memindai kerentanan SQL, terutama *SQL injection*), dan OWASP ZAP.

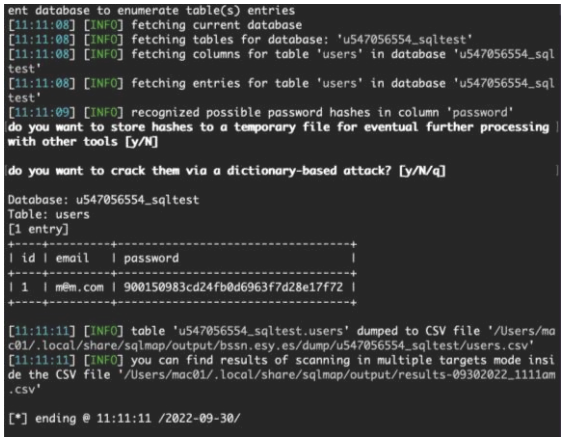
1. SQLMap

Pada percobaan dengan melakukan serangan pada *web server* menggunakan SQLMap didapatkan 2 buah kerentanan yang langsung dapat terdeteksi dalam rentang waktu rata-rata 12 detik. Adapun percobaan dilakukan sebanyak 1.300 kali dengan metode perulangan dan mendapatkan hasil yang sama. Hasil pada proses pemindaian dapat dilihat pada Gambar 4.



Gambar 4. Kerentanan Yang Ditemukan SQLMap Pada Percobaan Tanpa *WAF*

Terkonfirmasi bahwa aplikasi web memang memiliki kerentanan SQL injection dan dapat dieksploitasi lebih lanjut seperti *dump database* seperti yang ditunjukkan pada Gambar 5.



Gambar 5. *Dump Database*

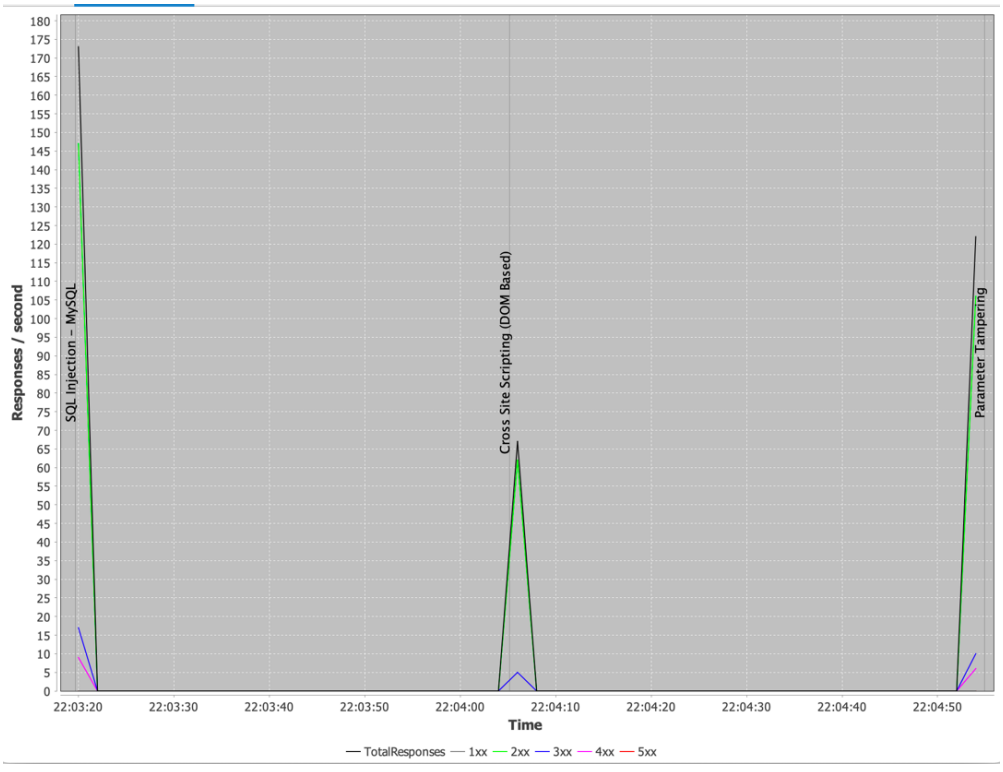
2. OWASP Zed Attack Proxy (ZAP)

Proses penyerangan aplikasi web dengan menggunakan OWASP ZAP dilakukan sebanyak 10 kali dengan beberapa hasil yang berbeda. Adapun hasil yang didapatkan dari percobaan yang dilakukan adalah sebagai Tabel 1 berikut:

Tabel 1. Pemindaian Web Tanpa *WAF* Dengan OWASP ZAP

Percobaan	Waktu	Jumlah *Request*	Jumlah *Alerts*
1	02:06.3	780	14
2	02:03.3	525	14
3	01:36.8	527	14
4	01:47.7	526	14
5	01:43.3	527	14
6	01:39.5	528	14
7	01:41.0	526	14
8	01:24.6	527	14
9	01:24.6	527	14
10	01:23.3	527	14
Rata-rata	**01:41.0**	**552**	**14**

Percobaan dengan jelas mendapatkan beberapa kerentanan yang timbul sesuai dengan kerentanan valid pada web server, selain itu juga muncul beberapa kerentanan yang *false positive* seperti ditunjukan pada Gambar 6.



Gambar 6. Grafik Hasil Pemindaian Dengan OWASP ZAP

3.2. Dengan *Web Application Firewall* Tanpa Modifikasi

Percobaan dilakukan terhadap *WAF* yang tidak dimodifikasi, dilakukan pengaturan *default* dengan *signature OWASP ModSecurity Core Rule Set* dengan tanpa perubahan apapun, terjadi perbedaan yang cukup signifikan bila dibandingkan dengan percobaan tanpa menggunakan *WAF*.

1. SQLMap

Percobaan dilakukan menggunakan SQLMap dengan target *WAF*. Dari 1300 kali percobaan, hasil waktu pemindaian rata-rata adalah 28 detik. Waktu

pemindaian bertambah hingga lebih dari 2 (dua) kali daripada pemindaian tanpa menggunakan *WAF*. SQLMap tidak dapat menemukan kerentanan *SQL Injection* yang ada, hal ini menandakan bahwa *WAF* dapat bekerja dengan semestinya dan menghalangi serangan bertipe *SQL injection*.

Dari percobaan didapatkan bahwa tools tidak menemukan kerentanan yang sebenarnya dikarenakan telah ditutup oleh *WAF*.

2. OWASP Zed Attack Proxy (ZAP)

Percobaan dengan proses penyerangan aplikasi web dengan menggunakan OWASP ZAP dilakukan sebanyak 10 kali secara manual dengan beberapa hasil yang berbeda. Adapun hasil yang didapatkan dari percobaan yang dilakukan adalah sebagaimana dapat dilihat pada Tabel 2. Percobaan Pemindaian *WAF* Dengan OWASP ZAP.

Tabel 2. Percobaan Pemindaian *WAF* Dengan OWASP ZAP

Percobaan	Waktu	Jumlah *Request*	Jumlah *Alerts*
1	00:24.4	621	9
2	00:24.0	522	21
3	00:24.0	522	21
4	00:21.0	522	12
5	00:20.7	522	9
6	00:21.0	483	9
7	00:21.0	485	9
8	00:22.0	522	9
9	00:21.0	522	9
10	00:20.0	522	12
Rata-rata	**00:21.9**	**524**	**12**

Pada percobaan didapatkan hasil pemindaian bahwa terdapat penurunan jumlah *alerts* yang didapatkan dibandingkan dengan pemindaian pada target tanpa menggunakan *WAF*. Hal ini karena *WAF* telah bekerja dengan semestinya yang akhirnya aplikasi pemindai kerentanan tidak dapat mengetahui kerentanan yang sebenarnya.

3.3. Dengan Modifikasi *Web Application Firewall*

Percobaan dilakukan terhadap *WAF* yang dimodifikasi dengan melakukan pengaturan terhadap *signature OWASP ModSecurity Core Rule Set* pada perubahan parameter *rules* yaitu *REQUEST-949-BLOCKING-EVALUATION.conf*

berikut ini merupakan perbandingan dari hasil pengujian penelitian. Diperlihatkan bahwa pada SQLMap tanpa menggunakan *WAF* menyelesaikan waktu pemindaian paling cepat, kemudian dan *RESPONSE-959-BLOCKING-EVALUATION.conf*. Perubahan *rules* digunakan untuk merubah respon *error* yang didapatkan dari setiap *request* menjadi *valid* agar aplikasi pemindai kerentanan tidak mengetahui bahwa *request* tersebut tidak *valid*.

1. SQLMap

Percobaan dilakukan menggunakan SQLMap dengan target *WAF* yang dimodifikasi. Dilakukan 1 (satu) kali percobaan dengan didapatkan hasil waktu untuk menyelesaikan pemindaian adalah 9 jam 38 menit dan 10 detik. SQLMap tidak dapat menemukan kerentanan *SQL injection*, hal ini menandakan bahwa *WAF* dapat bekerja dengan semestinya dan menghalangi serangan bertipe *SQL injection*, namun menyebabkan SQLMap membutuhkan waktu untuk menyelesaikan pemindaian jauh lebih lama karena setiap *request* yang diberikan menghasilkan nilai *valid* akan tetapi tidak dapat dieksploitasi lebih lanjut.

2. OWASP Zed Attack Proxy (ZAP)

Percobaan dengan proses pemindaian kerentanan web menggunakan OWASP ZAP dilakukan sebanyak 1 (satu) kali secara manual. Adapun hasil yang didapatkan adalah OWASP ZAP tidak dapat menyelesaikan pemindaian dalam waktu 72 jam ditunjukkan pada Tabel 3.

Tabel 3. Percobaan Pemindaian Modifikasi WAF Dengan OWASP ZAP

Percobaan	Waktu	Jumlah *Request*	Jumlah *Alerts*
1	> 3 hari	256	3

OWASP ZAP tidak mendapatkan kerentanan sebanyak yang didapatkan ketika hanya dengan menggunakan *WAF* tanpa modifikasi. Selain itu, diperlukan waktu yang sangat lama bahkan tidak selesai untuk menyelesaikan proses pemindaian.

3.4. Perbandingan Model *Web Application Firewal* Tanpa Modifikasi Dengan Modifikasi

Pada

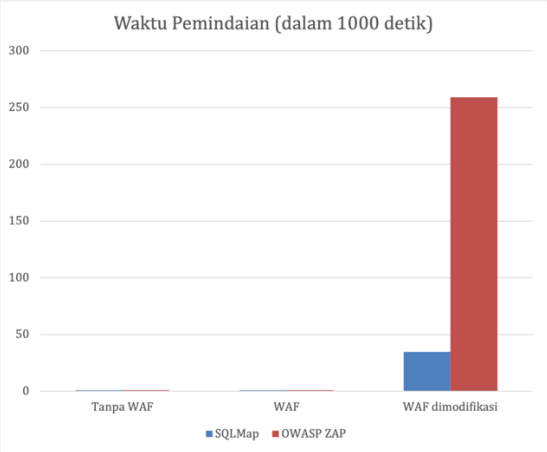
Tabel 4

membutuhkan waktu lebih lama pada saat menggunakan *WAF* tanpa modifikasi, dan secara signifikan membutuhkan waktu yang jauh lebih lama pada saat dilakukan modifikasi pada *WAF*.

Tabel 4. Perbandingan Hasil Proses Percobaan

Aplikasi Pemindai	Pencatatan	Tanpa *WAF*	*WAF* (tanpa modifikasi)	Modifikasi *WAF*
OWASP ZAP	Waktu	01:41.0	00:21.9	tidak selesai (3 hari)
	Jumlah *request* (rata-rata)	552	524	191
	Kerentanan	14	21	10
	Total percobaan	10	10	1
SQLmap	Waktu	0:12	0:28	9:38:10
	Jumlah *request* (rata-rata)	81	(tidak ada informasi)	(tidak ada informasi)
	Kerentanan	2	0	0
	Total percobaan	1300	1300	1

OWASP ZAP menyelesaikan pemindaian dengan waktu paling cepat adalah dengan pada *WAF* tanpa modifikasi, kemudian membutuhkan waktu lebih lama pada pemindaian aplikasi web tanpa *WAF* dan tidak dapat menyelesaikan pemindaian pada aplikasi web dengan perlindungan *WAF* yang dimodifikasi.



Gambar 7. Grafik Perbandingan Waktu Pemindaian

Gambar 7 menunjukkan bahwa dengan melakukan modifikasi pada *Web Application Firewall* (WAF) dengan menggunakan konsep *security through obscurity* secara signifikan memperlama waktu pemindaian pada Perangkat Lunak Pemindaian Kerentanan SQLMap dan OWASP ZAP.

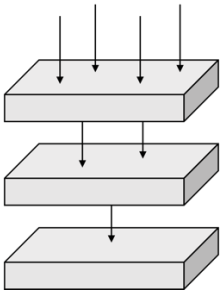
4. DISKUSI

Hasil percobaan yang dilakukan didapatkan hasil bahwa metode yang diusulkan efektif untuk menghindari pemindaian oleh aplikasi pemindai.

4.1. *Security Through Obscurity*

Prinsip *security through obscurity* digunakan sebagai langkah awal untuk mengurangi risiko ancaman dan dimaksudkan untuk memastikan hasil sementara saat mengambil langkah-langkah untuk menghilangkan ancaman [9] yang ditunjukan pada Gambar 8. Biasanya mitigasi ancaman dengan

bantuan prinsip ini yaitu perubahan *port transport layer* dalam sebuah jaringan, misalnya *port* SSH adalah 22 yang diganti menjadi 24 dengan maksud untuk menipu peretas yang datang. Prinsip ini digunakan pada *web application firewall* untuk memperlama waktu pemindaian dengan mengirimkan respon dalam jumlah besar dan random. Aplikasi pemindai akan menunggu respon tersebut hingga selesai dan akhirnya pemindaian membutuhkan waktu yang lebih lama bahkan tidak selesai.



Gambar 8. Prinsip *Security Through Obscurity*

4.2. Arsitektur Infrastruktur

Performa dari *WAF* juga bergantung pada jumlah koneksi yang datang. Oleh karenanya penelitian dibuat dalam lingkungan tertutup yang tidak terhubung dengan internet publik untuk menghindari adanya koneksi dari entitas lain diluar ruang lingkup penelitian. Penggunaan *virtual*

machine dalam entitasi fisik akan semakin memastikan koneksi terhubung secara langsung tanpa adanya halangan dari perimeter keamanan lain yang dapat menghalangi pemindaian.

4.3. *Injection*

Salah satu kategori serangan yang umum dan merusak adalah *injection*. *Injection* adalah serangan di mana penyerang menyuntikkan suatu *input* berbahaya ke aplikasi web. Biasanya karena kurangnya keterampilan pemrograman dalam kueri aplikasi menginterpretasikan *input* sebagai bagian dari perintah atau kueri, yang dapat mengakibatkan kerusakan parah [10]. Dalam penelitian ini, pengujian dilakukan pada serangan *SQL injection*. Serangan ini dilaporkan sebagai 10 kerentanan teratas [11] dan telah banyak menarik minat untuk meneliti dan mempublikasikan secara eksplisit [12] [13] [14] [15] [16] [17] [18].

Meskipun demikian, para peneliti dan praktisi tidak menyebutkan fitur *SQL injection* yang bisa digunakan untuk mendeteksi serangan SQL [19]. Walaupun mereka telah mengusulkan berbagai metode untuk mengatasi masalah injeksi SQL, pendekatan saat ini gagal untuk mengatasi cakupan atau batasan penuh dari serangan *SQL injection*. Pada dasarnya, serangan *SQL injection* sangat bergantung pada hasil respon yang diberikan oleh website. Dengan memodifikasi jenis respon yang diberikan oleh website, penelitian ini menawarkan metode untuk menghindari serangan tersebut dengan mengaburkan respon. Sehingga penyerang akan semakin jauh dari informasi kerentanan sebenarnya.

4.4. Modifikasi *Web Application Firewall*

Pendekatan metode aplikasi pemindai kerentanan diklasifikasikan ke dalam analisis statis, analisis dinamis, *white box*, *black box*, *taint analysis*, pengujian penetrasi, pengujian *fuzz*, pengujian *concolic*, pengujian eksekusi simbolik, dan pemeriksaan model, sebagian besar diklasifikasikan pada pendekatan ke dalam tiga kategori, yaitu, statis, dinamis, dan *hybrid* [16]. Namun secara keseluruhan, pendekatan tersebut tidak lepas dari artefak (misalnya, *Tag* HTML, fungsi logika, variabel dalam kode sumber, permintaan HTTP, parameter GET/POST, dan nilai *session* atau *cookie*). Hal ini menyebabkan aplikasi pemindai kerentanan sangat bergantung pada nilai balikan (respon) dari web server, oleh karenanya penelitian ini berfokus untuk melakukan modifikasi respon.

Kontrol yang digunakan pada penelitian ini dengan menggunakan *web application firewall* (*WAF*) untuk menyediakan kondisi tertentu sesuai dengan tujuan penelitian. *WAF* biasa digunakan di industri untuk melindungi aplikasi web dari eksploitasi kerentanan dan serangan keamanan lainnya, serta dapat bekerja dengan cara *heuristic* dan menggunakan prosedur atau pengaturan yang seringkali menggunakan pendekatan daftar blok [20].

Modifikasi *WAF* dilakukan dalam menghalangi aplikasi pemindai untuk mendapatkan kerentanan dengan cepat. Dilakukan modifikasi respon yang panjang dan jumlah respon acak serta memberikan kode *signature* yang seolah memiliki kerentanan *SQL injection* sehingga aplikasi pemindai kesulitan menyelesaikan pekerjaannya.

5. KESIMPULAN

Berdasarkan hasil uji yang dilakukan bahwa pemindaian kerentanan menggunakan SQLMap terhadap web application firewall yang dimodifikasi membutuhkan waktu lebih lama hingga 9 (sembilan) jam dibandingkan dengan tidak dilakukan modifikasi. Hal yang serupa juga terjadi pada penggunaan OWASP ZAP yang tidak dapat menyelesaikan pemindaian lebih dari 72 jam. Percobaan menunjukkan bahwa modifikasi pada *web application firewall* dengan konsep yang diajukan dapat menghindari pemindaian kerentanan. Hal ini membuktikan bahwa konsep *security through obscurity* dapat diterapkan pada keamanan web dalam menghindari aplikasi pemindai kerentanan yang akan memperlama waktu pemindaian kerentanan.

DAFTAR PUSTAKA

[1] P. Kühn, D. N. Relke dan C. Reuter, "Common Vulnerability Scoring System Prediction based on Open Source Intelligence Information Sources," *ArXiv.Org*, 5 10 2022.

[2] MITRE Corporation, "CVE Details," MITRE Corporation, [Online]. Available: https://www.cvedetails.com/. [Diakses 17 Desember 2022].

[3] Amankwah, Richard, J. Chen, Kudjo, Patrick Kwaku, Agyemang, Beatrice Korkor dan Amponsah, Alfred Adutwum, "An automated framework for evaluating open-source web scanner vulnerability severity," *Service Oriented Computing and Applications*, vol. 14, no. 4, p. 297–307, 2020.

[4] S. Bisson, "Add Security to Azure Applications with Azure WAF.," *InfoWorld.com; San Mateo*, 2022.

[5] N. M. Thang, "Improving Efficiency of Web Application Firewall to Detect Code Injection Attacks with Random Forest Method and Analysis Attributes HTTP Request," *Program Comput Soft*, vol. 46, p. 351–361, 2020.

[6] Bhatt, N., Kaur, J., Anand, A. dan Alhazmi, O. H., "Selecting best software vulnerability scanner using intuitionistic fuzzy set TOPSIS," *Computers, Materials, & Continua*, vol. 72, no. 2, pp. 3613-3629, 2022.

[7] A. C. Izumi dan I. R. Widiasari, ""SIASAT" UKSW (UNIVERSITAS KRISTEN SATYA WACANA) WEBSITE SECURITY ANALYSIS USING OWASP (OPEN WEB APPLICATION SECURITY PROJECT)," *Jurnal Teknik Informatika (Jutif)*, vol. 3, no. 3, pp. 763-770, 2022.

[8] W. Guo, Q. Wang, K. Z. A. G. Ororbia, S. Huang, X. Liu, C. L. Giles, L. Lin dan X. Xing, "Defending Against Adversarial Samples Without Security through Obscurity," *2018 IEEE International Conference on Data Mining (ICDM)*, vol. 10.1109/ICDM.2018.00029., pp. 137-146, 2018.

[9] A. D. Dakhnovich, D. A. Moskvin dan D. P. Zegzhda, Using Security-through-Obscurity Principle in an Industrial Internet of Things, Russia: ISSN, 2021.

[10] M. Amouei, M. Rezvani dan M. Fateh, "RAT: Reinforcement-Learning-Driven and Adaptive Testing for Vulnerability Discovery in Web Application Firewalls," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, pp. 3371-3386, 2022.

[11] A. van der Stock, B. Glas, N. Smithline dan T. Gigler, "OWASP Top Ten," OWASP, 2021. [Online]. Available: https://owasp.org/www-project-top-ten/. [Diakses 17 12 2022].

[12] Zhuo, Z., Cai, T., Zhang, X. dan Lv, F., "Long short-term memory on abstract syntax tree for SQL injection detection," *IET Software*, vol. 15, no. 2, pp. 188-197, 2021.

[13] M. N. Halgamuge, "Estimation of the success probability of a malicious attacker on blockchain-based edge network," *Computer Networks*, vol. 219, 2022.

[14] Z. Jiang, W. Zhenhua, J. Wang, L. Hua dan Z. Ming, "State Intellectual Property Office of China Receives Shenzhen Open Source Internet Security Tech Limited and Shenzhen Suzhou Andomain Science and Tech's Patent Application for SQL (Structured Query Language) Injection Vulnerability Detection Method and De". CN Paten CN114297662, 9 12 2022.

[15] E. Computer, "Prevention against Bot-Driven SQL Injection Attacks," *Express Computer*, 6 12 2022.

[16] B. Zhang, J. Li, J. Ren dan G. , "Efficiency and Effectiveness of Web Application Vulnerability Detection Approaches: A Review," *ACM Computing Surveys*, vol. 54, no. 9, pp. 1-35, 2022.

[17] S. Ruikun, L. Sen, W. Qiwei, Y. Zhijie dan C. , "Shanghai Pudong Development Bank Seeks Patent for Injection Attack Detection Method and Device, Computer Equipment and Readable Storage Medium". CN Paten CN114244558, 26 11 2022.

[18] H. Zhaoyang, R. Yukun, . X. Xin, H. Xiaogang, X. Chaoxu dan W. Junqiang, "Nanjing Ink Net Yunrei Science and Tech Submits Patent Application for SQL Injection Attack Identification Method Based on Deep Learning". CN Paten CN114169431 (A), 9 11 2022.

[19] S. H. N. Harip, I. R. A. Hamid, N. Murli dan N. Hassan, "Classification of SQL injection attack using K-Means clustering algorithm," dalam *AIP Conference Proceedings 2644, 040004 (2022)*, 2022.

[20] M. D. Junior a dan N. F. Ebecken, "A new WAF architecture with machine learning for resource-efficient use," *Computers & Security*, vol. 106, 2021.