JISKa, Vol. 5, No. 3, NOVEMBER, 2020, Pp. 146 – 152
ISSN: 2527 – 5836 (print) | 2528 – 0074 (online)

Analisis Kerentanan Serangan *Cross Site Scripting* (XSS) pada Aplikasi Smart Payment Menggunakan Framework OWASP

Imam Riadi [(1)], Rusydi Umar [(2)], Tri Lestari [(3)*]
[1] Sistem Informasi, Universitas Ahmad Dahlan, Yogyakarta
[2,3] Teknik Informatika Universitas Ahmad Dahlan, Yogyakarta
e-mail : imam.riadi@is.uad.ac.id, rusydi@mti.uad.ac.id, tri1907048008@webmail.uad.ac.id.
* Penulis korespondensi.
Artikel ini diajukan 15 April 2020, direvisi 20 Mei 2020, diterima 31 Mei 2020, dan dipublikasikan 9 November 2020.

Abstract

E-commerce that is growing so rapidly can provide space for unauthorized parties in carrying out cybercrime, security anticipation is needed so that e-commerce applications can be protected from harassment or hacking attacks such as cross-site scripting (XSS), malware, exploits, and database injection. This research was conducted to determine the vulnerability of the Smart Payment application by self-test using the ZAP tool. This test is carried out to secure applications that serve as recommendations for follow-up in securing the Smart Payment application. The results of this study found vulnerabilities in the Smart Payment application. Vulnerabilities found were Information Disclosure-Suspicious Comments, X-Frame-Options Header not Set, X-Content-Type-Options Header Missing, Timestamp Disclosure-Unix, XSS Protection Not Enabled Web Browsers, and Directory Browsing. In addition to obtaining vulnerabilities from the Smart Payment application, solutions are also provided to overcome vulnerabilities in the Smart Payment application.

Keywords: E-commerce, ZAP, Smart Payment

Abstrak

E-commerce yang berkembang begitu pesat dapat memberikan ruang bagi pihak yang tidak berwenang dalam melakukan tindakan kejahatan dunia maya, perlu dilakukan antisipasi keamanan agar aplikasi *e-commerce* dapat terhindar dari gangguan atau serangan peretas seperti *cross site scripting* (XSS), *malware*, eksploitasi, dan injeksi *database*. Penelitian ini dilakukan untuk mengetahui kerentanan aplikasi Smart Payment dengan cara *self test* menggunakan *tool* ZAP. Pengujian ini dilakukan untuk mengamankan aplikasi yang dijadikan sebagai rekomendasi tindak lanjut dalam pengamanan aplikasi Smart Payment. Hasil dari penelitian ini ditemukan kerentanan pada aplikasi Smart Payment. Kerentanan yang ditemukan berupa Information Disclosure-Suspicious Comments, X-Frame-Options Header not Set, X-Content-Type-Options Header Missing, Timestamp Disclosure-Unix, Web Browser XSS Protection Not Enabled, dan Directory Browsing. Selain diperoleh kerentanan dari aplikasi Smart Payment, diberikan juga solusi-solusi untuk mengatasi kerentanan pada aplikasi Smart Payment tersebut.

Kata Kunci: *E-commerce*, ZAP, Smart Payment

1. PENDAHULUAN

Era digital merupakan masa di mana manusia telah memahami teknologi dan semuanya serba terkoneksi seperti saat ini (Alia & Irwansyah, 2018). Era digital yang juga merupakan salah satu tanda kemajuan internet menyebabkan semuanya menjadi mudah dan cepat, sebagai contoh yaitu perkembangan perdagangan elektronik atau disebut dengan *e-commerce* (Pradana, 2016). *E-commerce* adalah upaya penjualan, pembelian, pemasaran, penyebaran barang dan jasa melalui sistem elektronik seperti televisi, jaringan komputer, atau internet, *e-commerce* juga melibatkan sistem inventori otomatis, sistem pengumpulan data otomatis dan transfer dana elektronik (Mumtahana et al., 2017).

E-commerce yang berkembang pesat dapat memberikan ruang bagi pihak yang tidak berwenang dalam melakukan tindak kejahatan di dunia maya (Mumtahana et al., 2017). Perlu dilakukan

antisipasi agar keamanan dari sebuah *e-commerce* dapat terhindar dari gangguan atau serangan seperti *malware*, *cross site scripting* (XSS), injeksi *database,* eksploitasi dan lain sebagainya (Umar et al., 2018) (Muhammad et al., 2017). Kesadaran dan pemahaman yang kurang terhadap isu keamanan aplikasi atau sistem selalu mengancam setiap saat khususnya bagi para pengembang (W et al., 2016). Perusakan atau kebocoran data bisa mengancam setiap waktu seiring dengan meningkatnya sumber daya manusia (Iqbaludin et al., 2018).

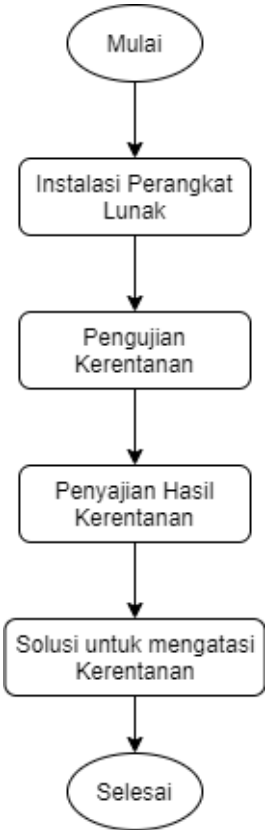
Smart Payment merupakan salah satu dari sekian banyaknya aplikasi *e-commerce* yang sedang dikembangkan. Aplikasi ini digunakan untuk pembayaran uang sekolah secara digital dibuat untuk memudahkan proses transaksi orangtua siswa dan pihak sekolah. Perusakan atau kebocoran data dapat mengancam aplikasi ini setiap saat, seiring dengan meningkatnya sumber daya manusia (Dewanto, 2018). Kesadaran dan pemahaman yang kurang terhadap isu keamanan sistem juga mengancam setiap saat khususnya bagi para pengembang (Syarifudin, 2018). Pengamanan aplikasi Smart Payment perlu dilakukan agar terhindar dari serangan atau gangguan peretas. Pengamanan tersebut dapat dilakukan dengan cara pengujian yang dapat dilakukan dengan beberapa metode salah satunya ialah *self test* menggunakan *framework Open Web Application Security Project* (OWASP) Top 10 (Sunardi et al., 2019).

OWASP Top 10 merupakan sebuah panduan bagi para *developers* dan *security team* tentang kelemahan-kelemahan pada web *apps* yang mudah diserang dan harus segera disiasati (Ghozali et al., 2019). OWASP merekomendasikan perusahaan-perusahaan untuk memperhatikan sepuluh ancaman keamanan aplikasi web yaitu *broken access control, security misconfiguration, insecure deserialization, injectionn, sensitive data exposure, XML external entities, broken authentication, cross site scripting, using components with known vulnerabilities, and insufficient logging and monitoring* (Yunus, 2019). OWASP mengembangkan *tool* yang digunakan untuk mengamankan aplikasi web, salah satunya ialah *Zad Attack Proxy* (ZAP). ZAP merupakan aplikasi untuk menemukan kerentanan dalam suatu aplikasi web dengan cara menyediakan *scanner* otomatis (Syarifudin, 2018). Kelebihan dari ZAP ini di antaranya bersifat mudah diinstal, *community based*, *open source*, *intercepting proxy*, *traditional & ajax spider*, *active scanner*, *growing add ons*, *forced browsing*, *fuzzer*, *dynamic*, *smart card support*, SSL cerificates, *integrared*, dan *web socket support* (Sunardi et al., 2019).

Penelitian ini bertujuan untuk mengetahui kerentanan pada aplikasi Smart Payment dengan cara *self test* menggunakan *tool* ZAP, kerentanan yang diperoleh dapat dijadikan sebagai rekomendasi tindak lanjut dalam pengamanan aplikasi Smart Payment. Penelitian serupa telah dilakukan oleh peneliti sebelumnya diantaranya deteksi kerentanan yang dilakukan dengan membandingkan dua *tool* yaitu ZAP dan Archni yang berhasil mendapatkan bukti dalam bentuk deskripsi, URL, metode, parameter, informasi, dan bukti (Sunardi et al., 2019). Peneliti lain juga melakukan analisis deteksi *vulnerability* pada web *server* menggunakan OWASP *scanner* yang berhasil menemukan kerentanan yang dapat menyebabkan file lokal dapat dimanipulasi dengan menggunakan serangan *croos site scripting* (XSS), penelitian ini juga dilengkapi dengan solusi penanganan kerentanannya (W et al., 2016). Pengujian kerentanan terhadap web *server* SIMAK dengan melakukan *penetration testing* menemukan dua kelemahan yaitu Apache Server ETag Header Information Disclosure dengan status medium atau *middle risk* dan Unix Operating System Unsupported Version Detection yang berstatus *critical* atau *high risk* (Wahyudi, 2019).

2. METODE PENELITIAN

Penelitian ini dilakukan dengan 4 tahap, tahap pertama yang dilakukan ialah instalasi perangkat lunak*,* kemudian dilanjut dengan pengujian kerentanan, setelah itu penyajian hasil dari pengujian kerentanan dan terakhir ialah memberikan solusi dari kerentanan aplikasi. Skema metode pada penelitian ini dapat dilihat seperti pada Gambar 1.



Gambar 1. Tahap pengujian kerentanan aplikasi Smart Payment.

Gambar 1 merupakan *flowchart* yang menggambarkan langkah-langkah atau alur yang dilakukan pada penelitian ini. Instalasi *software* merupakan proses yang dilakukan untuk mempersiapkan *tools* yang digunakan dalam penelitian. Pengujian kerentanan ialah proses yang dilakukan untuk mengetahui kerentanan pada aplikasi Smart Payment, setelah diketahui kerentanan dari aplikasi Smart Payment maka langkah selanjutnya ialah menampilkan hasil dari pengujian kerentanan, dengan kata lain menyajikan kerentanan apa saja yang ada pada aplikasi Smart Payment. Langkah terakhir yaitu memberikan solusi penangan dari kerentanan yang terjadi pada aplikasi Smart Payment.

3. HASIL DAN PEMBAHASAN

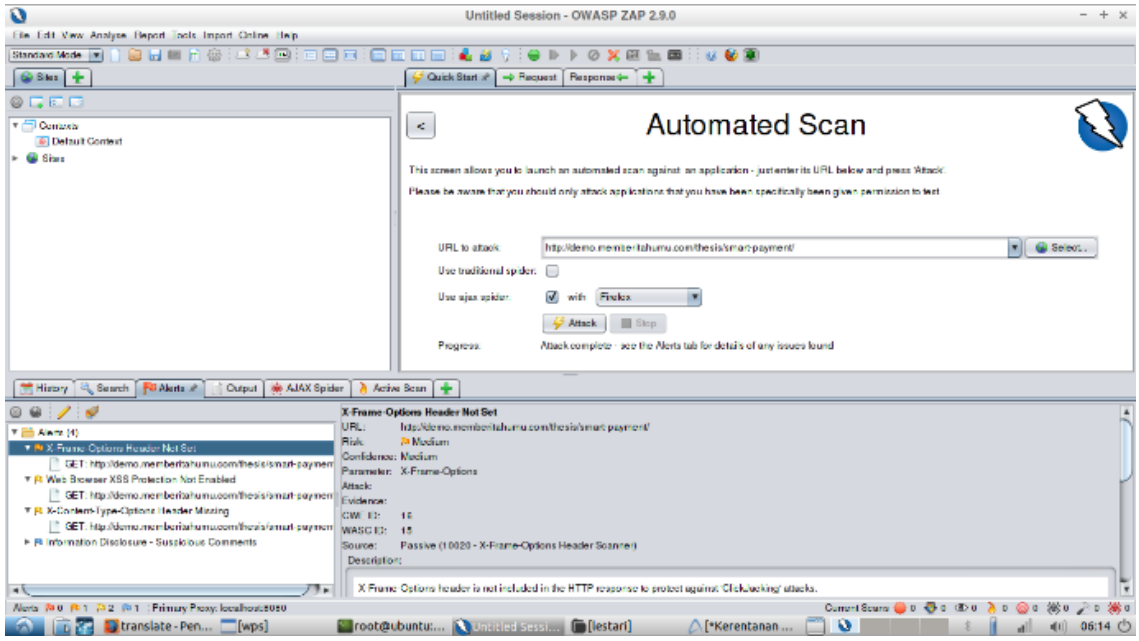
Aplikasi Smart Payment memiliki enam kerentanan keamanan, di mana kerentanan tersebut diperoleh menggunakan *tool* ZAP. Langkah-langkah yang dilakukan untuk memperoleh kerentanan pada aplikasi Smart Payment di antaranya adalah sebagai berikut,

3.1. Instalasi *Software*

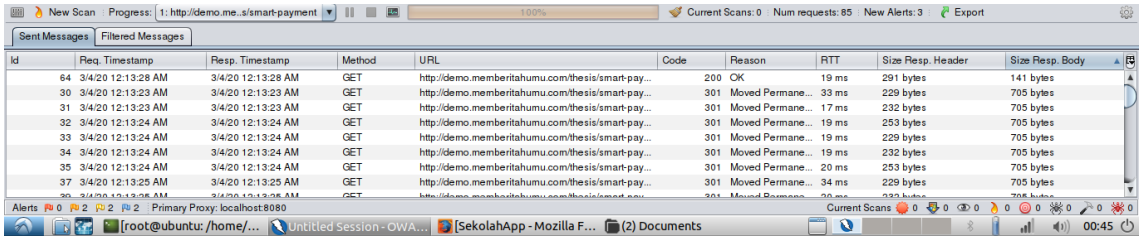
Beberapa *software* yang digunakan untuk melakukan pengujian kerentanan aplikasi Smart Payment pada penelitian ialah menggunakan sistem operasi Linux yang dilengkapi xampp/lampp versi 7.3.14-0 dan ZAP versi 2.9.0 yang digunakan sebagai *tool* penguji kerentanan.

3.2. Proses Pengujian Kerentanan

Proses pengujian kerentanan atau proses mendeteksi kerentanan pada aplikasi Smart Payment dalam penelitian ini dilakukan dengan menggunakan *tool* ZAP seperti yang telah dijelaskan sebelumnya ZAP ini merupakan salah satu proyek OWASP yang paling aktif dan diberi status unggulan. Gambar 2 dan Gambar 3 merupakan proses memindai kerentanan pada aplikasi Smart Payment.



Gambar 2. Kerentanan aplikasi Smart Payment



Gambar 3. Proses pemindaian aplikasi Smart Payment

Gambar 2 merupakan proses awal yang dilakukan untuk memindai aplikasi Smart Payment dan Gambar 3 merupakan proses ketika pemindaian aplikasi Smart Payment sedang berlangsung. Proses pemindaian kerentanan pada aplikasi Smart Payment menggunakan *tool* ZAP membutuhkan waktu sekitar 10 menit, setelah pemindaian selesai maka secara otomatis diperoleh laporan hasil kerentanan pada aplikasi Smart Payment. Kerentanan yang diperoleh dari aplikasi Smart Payment dapat dilihat pada Tabel 1.

Tabel 1 berisi hasil pemindaian Aplikasi Smart Payment. Identifikasi untuk menentukan tingkat resiko pada penelitian ini menggunakan metode pemodelan ancaman (*threat modeling*). Metode ini dapat digunakan untuk memperkirakan tingkat keparahan semua risiko terhadap aplikasi web dan membuat keputusan berdasarkan informasi tentang apa yang harus dilakukan terhadap risiko tersebut. Risiko diukur dengan skala 0 hingga 9 dan dibagi menjadi tiga bagian seperti pada Tabel 2.

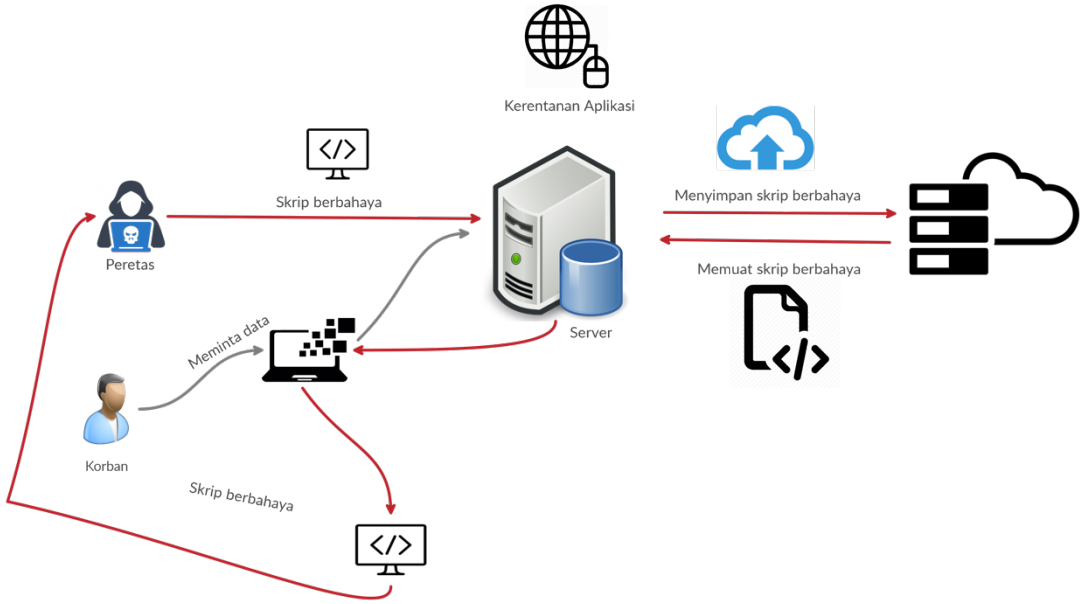
Tabel 1. Kerentanan aplikasi Smart Payment.

Jenis	Url	Risk	Confidence
Directory Browsing tidak di nonaktifkan	http://demo.memberitahumu.com/thesis/smart-payment/asset/	Medium	Medium
X-Frame-Options Header tidak diaktifkan	http://demo.memberitahumu.com/thesis/smart-payment/asset/	Medium	Medium
Web Browser XSS Protection tidak diaktifkan	http://demo.memberitahumu.com/thesis/smart-payment/asset/	Low	Medium
X-Content-Type-Options Header hilang	http://demo.memberitahumu.com/thesis/smart-payment/asset/	Low	Medium
Information Disclosure-Suspicious Comments	http://demo.memberitahumu.com/thesis/smart-payment/asset/	Informational	Medium
Timestamp Disclosure-Unix	http://demo.memberitahumu.com/thesis/smart-payment/asset/Login%20-%20EGREF_files/materialize.css	Informational	Low

Tabel 2. Level Risiko

Tingkat kemungkinan dan dampak	
0 sampai <3	Low (rendah)
3 sampai <6	Medium (menengah)
6 sampai 9	High (tinggi)

Tabel 2 merupakan level risiko untuk menentukan seberapa parah resiko yang mungkin terjadi pada aplikasi Smart Payment dengan adanya kerentanan yang dimiliki. Penelitian ini menemukan enam kerentanan dimana informasi kerentanan yang diperoleh sebagai berikut, dengan tingkat resiko menengah (2 kasus), rendah (2 kasus) dan dua kasus diantarnya hanya bersifat *informational*. Selanjutnya, berikut ini merupakan simulasi serangan XSS pada aplikasi Smart Payment dapat dilihat seperti pada Gambar 4.



Gambar 4. Simulasi Serangan XSS Aplikasi Smart Payment.

Gambar 4 menunjukkan tahapan simulasi serangan XSS pada aplikasi Smart Payment. Seorang peretas menyisipkan naskah berbahaya ke dalam server melalui salah satu kerentanan pada aplikasi sehingga peretas dapat mengendalikan sesi pengguna, mencuri data dan menjalankan kode jahat. Berikut ini merupakan penjelasan dari seluruh kerentanan yang ada pada aplikasi Smart Payment :

1) Directory Browsing yang tidak dinonaktifkan memungkinkan peretas untuk melihat daftar direktori pada aplikasi Smart Payment, daftar direktori biasanya dapat mengungkap skrip tersembunyi termasuk *file* yang dapat diakses untuk membaca informasi sensitif.
2) X-Frame-Options Header yang tidak diatur memungkinkan peretas untuk melakukan serangan ClickJacking, peretas dapat menanamkan sebuah *script* atau kode dimana *script* tersebut bisa mencuri data atau mengendalikan komputer pengguna yang mengklik tautan atau tombol.
3) Web Browser XSS Protection yang tidak diaktifkan dapat menyebabkan peretas melakukan serangan XSS *(cross site scripting)*. Salah satu serangan yang dapat dilakukan peretas ialah pencurian *cookie* (kunci untuk membajak *session*).
4) X-Content-Type-Options Header tidak distel ke 'nosniff yaitu penanda yang digunakan oleh server untuk menunjukan tipe MIME yang diiklankan dalam *header* tidak boleh diubah atau diikuti tidak ada, sehingga dapat menyebabkan tidak adanya perlindungan Cross-Origin Read Blocking (CORB) untuk file HTML, TXT, JSON dan XML (tidak termasuk gambar SVG / svg + xml).
5) Information Disclosure-Suspicious Comments merupakan respon yang berisi komentar mencurigakan yang dapat membantu peretas untuk melakukan serangan.
6) Timestamp Disclosure-Unix merupakan cara untuk membantu aplikasi atau server web-Unix melacak dan memilih informasi sesuai dengan tanggal mereka saat menggunakan internet. Jika *timestamp disclosure-unix* tidak ada maka informasi yang sesuai tanggal penggunaan aplikasi atau server *web-unix* tidak dapat dilacak.

3.3. Rekomendasi Penangan Kerentanan

Berdasarkan OWASP Top 10 untuk menangani kerentanan yang terdapat pada aplikasi Smart Payment yang telah dijelaskan sebelumnya, maka diberikan rekomendasi sebagai berikut :

1) Directory Browsing harus dinonaktifkan karena jika tidak dinonaktifkan dapat digunakan oleh peretas untuk melihat *file*, menyalin gambar, mencari tahu struktur direktori dan informasi lainnya. Cara menonaktifkan Directory Browsing ialah dengan menambahkan kode "options-indexes" pada *file* .htaccess.
2) X-Frame-Options Header diatur hanya diperbolehkan kepada alamat web atau IP tertentu.
3) Solusi untuk mengatasi serangan *cross site scripting* (XSS) ialah dengan memastikan filter XSS web browser diaktifkan, dengan cara mengatur *header respons* HTTP X-XSS-Protection ke'1 '.
4) Solusi dari X-Content-Type-Options Header yang hilang ialah memastikan aplikasi Smart Payment menetapkan X-Content-Type-Options Header dengan tepat. X-Content-Type-Options Header diatur ke 'nosniff' dan diatur untuk semua halaman web.
5) Solusi untuk mengatasi Information Disclosure-Suspicious Comments ialah dengan cara menghapus semua komentar yang mengembalikan informasi sehingga dapat membantu penyerang dan memperbaiki masalah yang tidak sesuai yang mereka rujuk.

Solusi dari Timestamp Disclosure-Unix ialah dengan mengkonfirmasi secara manual bahwa data stampel waktu tidak sensitif dan data tidak dapat dikumpulkan untuk mengungkap pola yang dapat dieksploitasi.

4. KESIMPULAN

Hasil yang diperoleh dari penelitian ini terdapat beberapa kerentanan pada aplikasi Smart Payment yang diperoleh dengan menggunakan tools ZAP. Kerentanan yang ditemukan berupa Information Disclosure-Suspicious Comments, X-Frame-Options Header not Set, X-Content-Type-Options Header Missing, Timestamp Disclosure-Unix, Web Browser XSS Protection Not Enabled, dan Directory Browsing. Selain diperoleh kerentanan dari aplikasi Smart Payment, diberikan juga solusi-solusi untuk mengatasi kerentanan pada aplikasi Smart Payment tersebut.

Berdasarkan pengujian yang sudah dilakukan dapat disimpulkan bahwa aplikasi Smart Payment memiliki beberapa kerentanan yang diperoleh menggunakan *tool* ZAP.

DAFTAR PUSTAKA

Alia, T., & Irwansyah, I. (2018). Pendampingan Orang Tua pada Anak Usia Dini dalam Penggunaan Teknologi Digital. *A Journal of Language, Literature, Culture and Education*, *14*(1), 65. https://doi.org/10.19166/pji.v14i1.639

Dewanto, A. P. (2018). *Penetration Testing padaDomain uii.ac.id Menggunakan OWASP 10*.

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

Iqbaludin, Ferdiansyah, D., & Kurniawan, I. (2018). *Pengujian Celah Keamanan pada Website Captive Portal dengan Menerapkan Penetration Testing (Studi Kasus: Teknik Informatika Universitas Pasundan)*. Universitas Pasundan.

Muhammad, A. W., Riadi, I., & Sunardi, S. (2017). Deteksi Serangan DDoS Menggunakan Neural Network dengan Fungsi Fixed Moving Average Window. *JISKA (Jurnal Informatika Sunan Kalijaga)*, *1*(3), 115. https://doi.org/10.14421/jiska.2017.13-03

Mumtahana, H. A., Nita, S., & Tito, A. W. (2017). Pemanfaatan Web E-Commerce untuk Meningkatkan Strategi Pemasaran. *Khazanah Informatika: Jurnal Ilmu Komputer Dan Informatika*, *3*(1), 6. https://doi.org/10.23917/khif.v3i1.3309

Pradana, M. (2016). Klasifikasi Bisnis E-Commerce Di Indonesia. *Modus*, *27*(2), 163. https://doi.org/10.24002/modus.v27i2.554

Sunardi, Riadi, I., & Raharja, P. A. (2019). Vulnerability analysis of E-voting application using open web application security project (OWASP) framework. *International Journal of Advanced Computer Science and Applications*, *10*(11), 135–143. https://doi.org/10.14569/IJACSA.2019.0101118

Syarifudin, I. (2018). *Pentesting dan Analisis Keamanan Web Paud Dikmas*. *April*.

Umar, R., Riadi, I., & Zamroni, G. M. (2018). Mobile Forensic Tools Evaluation for Digital Crime Investigation. *International Journal on Advanced Science, Engineering and Information Technology*, *June*. https://doi.org/10.18517/ijaseit.8.3.3591

W, Y., Riadi, I., & Yudhana, A. (2016). Analisis Keamanan Webserver Menggunakan Metode Penetrasi Testing (PENTEST). *Annual Research Seminar*, *2*(1), 300–304.

Wahyudi. (2019). Analisa Pengujian Kerentanan Terhadap Web Server SIMAK (Studi Kasus : STMIK Kharisma Karawang). *Jurnal Teknologi Informasi*, *5*(1).

Yunus, M. (2019). *Analisis Kerentanan Aplikasi Berbasis WEB Menggunakan Kombinasi Security Tools Project Berdasarkan Framework OWASP Versi 4*.