# DETEKSI SERANGAN *VULNERABILITY* PADA *OPEN JURNAL SYSTEM* MENGGUNAKAN METODE *BLACK-BOX*

**Yunanri.W[1], Doddy Teguh Yuwono[2], Rodianto[3] , Yuliadi[4]**

[134]Program Studi Teknik Informatika, Universitas Teknologi Sumbawa. [2]Program Studi Ilmu Komputer, Universitas Muhammadiyah Palangkaraya

Jln. Raya Olat Maras , Batu Alang, Moyo Hulu, Pernek Moyohulu, Kabupaten Sumbawa, Nusa Tenggara Barat, 84371
[1]yunanri.w@uts.ac.id, [2]doddy.zhal09@gmail.com, [3] rodianto@uts.ac.id, [4]yuliadi@uts.ac.id

## Abstract

*Penetration testing is a series of activities carried out to identify and exploit security vulnerabilities. Penetration testing is a test on systems that have critical elements that endanger the Open Journal System (OJS) application running on the internet. The methodology uses the Blackbox method where the testing process is carried out to determine the critical level of error in the software, including three phases: test preparation, test, and test analysis. The pilot phase involves the following steps: information gathering, vulnerability analysis, and vulnerability exploits. Penetration testing. Tests that have been carried out have identified 1 high-risk vulnerability, 7 medium risk vulnerability, 90 vulnerability to low risk in OJS. The total vulnerability in testing amounted to 98 file system vulnerabilities with additional 1043 file system information to be followed up for repairs.*

**Keywords** : *Vulnerability, OJS, Pentration Testing, Blackbox.*

## Abstrak

Pengujian *penetration testing* adalah serangkaian kegiatan yang dilakukan untuk mengidentifikasi dan mengeksploitasi kerentanan keamanan. *Penetration testing* merupakan pengujian pada sistem yang memiliki elemen yang bersifat kritis membahayakan apalikasi *Open Journal System* (*OJS*) yang berjalan pada internet. Metodologi menggunakan metdoe *Blackbox* dimana, proses pengujian dilakukan untuk mengetahui tingkat kesalahan yang bersifat kritis pada perangkat lunak, mencakup tiga fase: persiapan pengujian, tes dan analisis tes. Tahap uji coba melibatkan langkah-langkah berikut: pengumpulan informasi, analisis kerentanan, dan kerentanan mengeksploitasi. Pengujian Penetration testing. Pengujian yang telah dilakukan mengidetifikasi 1 kerentanan *high risk*, 7 kerentanan *medium risk*, 90 pada kerentanan *low risk* pada OJS. Total *vulnerability* pada pengujian berjumlah 98 *vulnerability file* sistem dengan tambahan informasi 1043 *file* sistem untuk ditindaklanjuti dalam perbaikan.

**Kata kunci** : *Vulnerability, OJS, Pentration Testing, Blackbox.*

## 1. PENDAHULUAN

Keamanan sistem informasi adalah salah satu isu utama dalam perkembangan teknologi informasi dan komunikasi. Selain itu, bisnis penting untuk melindungi aset informasi organisasi dengan mengikuti pendekatan yang konprehensif dan terstruktur untuk memberikan perlindungan dari resiko organisasi yang mungkin di hadapi. Dalam upaya memecahkan masalah keamanan dibutuhkan penerapan metode yang dapat menjamin keamanan data, transaksi, dan komunikasi [1].

Dengan pesatnya perkembangan dunia internet saat ini mengakibatkan ada usaha maksimal dari suatu organisasi dan individua untuk membuat sebuah keamanan dalam sistem dan jaringan karena sangat memungkinkan

datangnya sebuah serangan. *Hacker* merupakan seseorang yang memiliki kemampuan dalam pemrograman serta jaringan komputer [2]. Seiring pesatnya perkembangan teknologi maka para *hacker* juga semakin pintar dalam menjalankan pola kegiatan ilegal. Dengan kata lain semakin banyak para *hacker* yang memanfaatkan kelemahan pada sebuah *Webserver* untuk mendapatkan keuntungan pribadi maupun organisasi yang dijalankannya. Melihat kasus yang sering terjadi, seharusnya kita dapat mengambil langkah cepat untuk mengamankan *Webserver* dan apabila diabaikan maka *webserver* yang di miliki oleh suatu badan institusi baik milik perintah, swasta, maupun perseorangan dapat mengalami kerugian yang diakibatkan oleh para *hacker* [2].


Gambar 1. Grafik serangan *cybercrime* di Indonesia.

Dalam studi kasus keamanan *webserver*, bertujuan untuk mencari kerentanan atau kelemahan dari sebuah *Webserver*, karena banyak sekali kasus penyerangan yang dilakukan oleh para peretas atau *hacker*. Solusi yang dilakuakn dalam upaya meminimalisir kerentanan atau *vulnerability* pada *webserver* setiap institusi milik pemerintah, institusi swata, maupun perseorangan[3].

Berdasarkan Gambar 1. Diatas mendapatkan informasi bahawa begitu penting nya melakukan audit terhadap *webserver*, yang dimiliki oleh setiap negara untuk meminimalisir celah pada sistem aplikasi *webserver*[4].

Indonesia tercatat sebagai Negara peringkat 5 yang paling banyak terinfeksi *ransoware* di Asia Tenggara dengan jumlah rata-rata 14 kasus terjadi setiap hari, menurut riset yang dilakukan perusahaan peranti lunak antivirus *Symantec* [5]. Program jahat yang masuk dalam kategori *ransomware*, dimana system kerja dari para peretas atau *hacker* akan mengunci data yang tersimpan dikomputer nya sendiri, lalu lewat notifikasi para peretas atau *hacker* akan meminta bayaran berupa *bitcoin* jika pengguna hendak mendapatkan akses kembali datanya, jika tidak dibayar maka data akan terkunci atau korban harus menunggu sampai ada pihak seperti perusahaan Antivirus yang membasmi *malware* dari *Ransomware* tersebut[6].

## 2. TINJAUAN PUSTAKA DAN TEORI

### 2.1. Tinjauan Pustaka

a. Penelitian yang di lakukan Ade Kurniawan, Imam Riadi, Ahmad lutfi "Analisis Forensik dan Penanganan Cross Site, Melakukan Serangan Pada Target Menggunakan Keamanan Web Framework OWASP". Journal of Theoretical & Applied Information Technology (JATIT) Vol.95, ISSU 6. Universitas Ahmad Dahlan Yogyakarta. 2017.

Analisis Forensik dan Cegah Salib *Scripting* Situs Menggunakan Aplikasi *Web* Terbuka Kerangka Proyek *OWASP* mencakup tiga tahapan penting, yaitu: Tahap penyerangan,Analisis, dan *Patching*. Tahapan Menyerang lakukan kegiatan dengan metode *Single*-Korban menggunakan OWASP *Xenotix XSS Attack Exploit Framework* v6.2 untuk memasukkan serangan Informasi Berkumpul, *Keylogger*, Unduh *spoofer* dan *Webcam* langsung *screenshot* ke korban melalui Mozilla Peramban Firefox. Tahapan Analisis dilakukan menggunakan Live Forensic oleh *Wireshark*, HTTP langsung *Header* dan *Tcpdump*. Penggunaan forensik hidup metode diaktifkan untuk menangkap semua jenis kegiatan terjadi permintaan seperti itu, *payload,* dan skrip. Hasil tahap analisis dan bagaimana naskahnya mengajukan. beberapa file dalam nilai hash pengujian dengan aplikasi untuk menggunakannya untuk membandingkan nilai dari integritas file yang telah diunduh oleh file tempat korban yang disimpan di *server*. Mencegah tahap terakhir adalah proses dengan membuat menambal sisi pengguna dengan memasang ekstensi *add-on di browser Mozilla Firefox* dengan nama ekstensi *XSSFilter*

Ada. Garis besar *XSSFilter* menyediakan lebih awal peringatan, menonaktifkan *plugin*, membatasi, mengizinkan *payload*.

b. Penelitian yang di lakukan Rina Elizabeth, Lopez de Jimenez, "Pentesting on web

Aplication Using Ethical Hacking". Internasional Conference for Internet Technology and Secured Transactions (ICITST), Santa Telca, EL Savador. ISSU 503. 2016.

Pengujian penetrasi, pada Kali Linux, merupakan aplikasi *framework* gratis, yang mampu menganalisa serangan yang berdampak kerentanan pada *website*. Analisa yang di lakukan, memberikan sebuah informasi tentang aplikasi web yang benar-benar aman dan gratis dari serangan, tetapi dengan menggunakan teknik atau uji intrusi (*Pentesting*) sebagai alat *Ethical Hacking,* semua kerentanan tersebut dapat mengatasi, menghindari serangan yang merusak integritas dan kehandalan data yang mereka tangani.

c. Penelitian yang di lakukan Tiago Vieira, Carlos serrao, "Web Security in the Finance Sector". Internatonal Conference for Internet Technology and Secured Transaction (ICITST). Institut University de Lisboa. Portugal, Vol 11. 2016

Menganalisis hasil dalam konteks *web*, bahkan dengan keamanan pertimbangan dalam pengembangan mereka, kerentanan kritis ditemukan. Dengan waktu dan motivasi, bahkan mungkin lebih kerentanan kritis atau dengan konsekuensi kritis bisa ditemukan[11].

Perbandingan dangkal dapat dilakukan di sektor fimance di mana layanan aplikasi web berdasarkan teknologi *.Net* dikembangkan. Meskipun. *Net Framework* memiliki pertahanan mekanisme seperti pertahanan injeksi, kerentanan lain mungkin eksploitasi mereka dapat menjadi ancaman bagi pihak-pihak yang terlibat[12].

d. Penelitian yang di lakukan Yaroslav Stefinko, Andrian Piskozub, Roman Banakh, "Manual and Automated Penetration Testing. Benefits and Drawbacks. Modrn Tendency".TCSET- IEEE. Ukraina, 2016

Pengujian *penetration testing* (pentest) memungkinkan organisasi untuk menilai kerentanan secara proaktif, menggunakan *eksploitasi* kata-nyata, memungkinkan mereka untuk mengevaluasi potensi sistem mereka untuk ditumbangkan melalui peretasan dan skema *malware* dengan cara yang sama seperti yang digunakan oleh para penyerang. tes pentest manual masih lebih populer dan berguna, karena kerentanan yang berbeda di bagian keamanan, misalnya dalam faktor manusia. *Etihcal Hacking* etika berpengalaman digunakan untuk menulis *script* sendiri atau bahkan mengotomatiskan salah satu tahapan, agar dapat melanjutkan dengan cepat dan menemukan lebih banyak kebocoran keamanan dalam sistem target. proses otomatisasi ini dapat ditingkatkan dengan menggunakan bahasa *scripting*.

e. Penelitian yang di lakukan Muhammad Nur Faiz, Rusdy Umar, Anton Yudhana, "Implemantasi Forensik Langsung untuk Perbandingan *browser* pada keamanan *Email*." Jurnal Informatika Sunan Kalijaga (JISKa). Universitas Ahmad Dahlan Yogyakarta, Vol 1. ISSU 3. No. 108-114. 2017

Keamanan pada *browser* merupakan suatu tantangan tersendiri untuk mengembangkan fitur kemanan dan kemudahan dalam menggunakan *browser.* Microsoft Edge merupakan *browser default* dari Windows 10 dengan berbagai fitur yang lebih baik dari *Internet Explorer* namun ternyata untuk segi keamanan lebih lemah jika dibandingkan dengan *browser Mozilla Firefox,* sedangkan *Google Chrome* lebih kuat pada *password* nya.

### 2.2. Teori terkait.

*Black Box Testing* merupakan pengujian yang didasarkan pada detail aplikasi seperti tampilan aplikasi, fungsi-fungsi yang ada pada aplikasi, dan kesesuaian alur fungsi dengan bisnis proses yang diinginkan oleh *customer*. Pengujian ini tidak melihat dan menguji *souce code* program.

Kegiatan pengujian pada target:

- membuat test case untuk menguji fungsi-fungsi yang ada pada aplikasi.
- membuat *test case* untuk menguji kesesuaian alur kerja suatu fungsi di aplikasi dengan *requirement* yang dibutuhkan *customer* untuk fungsi tersebut.
- mencari *bugs*/*error* dari tampilan (*interface*) aplikasi.

## 3. METODOLOGI PENELITIAN

### A. Metodologi

*Metodologi* ini mengacu pada *penetration testing*, metode *blackbox* yang diasumsikan pada pengujian yang tidak mengetahui sama sekali infrastruktur sistem bangun yang dimiliki pada terget. Dengan demikian seorang *security audit* melakukan serangan pada *webserver* yang bertujuan mengumpulkan informasi yang dibutuhkan.

### 3.1. Sekenario penyerangan

*Sekenario* penyerangan mengacu pada vega *framework* yang powerful seperti struts atau spring. Ditemukan cacat pada *XSS*, akan ada list pada target yang diperbaiki agar mudah untuk diperbaiki dari hasil exploitasi oleh *tool vega scanner vulnerability* [7].

### 3.2. Pengujian pada *Admin* dan *Password*

Untuk penyerangan pada *page admin* dan *password* tidak akan dirubah pada saat *penetration testing guide* dalam mencari *Error code* [8].

### 3.3. *Listing Code*

Listing *direktori class java* mengarah pada memperoleh kode aplikasi yang cacat.

### 3.4. Analisis Potensial ada nya *vulnerability*

Konfigurasi *stack trace* ke *user* bertujuan mencari celah yang memiliki potensial cacat pada *app server*[9].

### B. Sekenario Penyerangan (*Red Tim* )

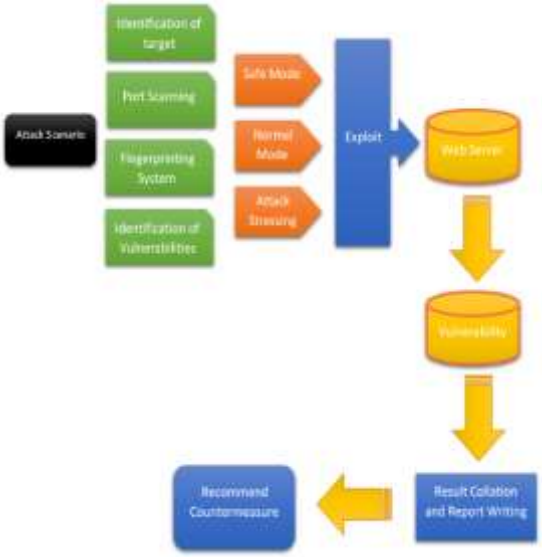
*Berdasarkan* gambar 3. Skema serangan pada *webserver* diatas dapat disimpulkan bahwa:

a. Menunjukan bahwa proses proses *input* masuk pada kategori "yes" merupakan tindakan eksekusi untuk mencari informasi baik *informasi gathering* , identifikasi *port,* proses *Scanning* sistem dan jaringan.
b. Kategori "*no*" semua aktifitas penetrasi testing di batalkan.
c. Kategori "output" aktifitas proses *pentration testing* telah selesai, dimana data yang berkaitan kerentanan atau *vulnerbility* dapat ditampilakan baik berupa data file, gambar kodingan, diagram dan lain-lain [9]. Sebagai mana yang ditampilkan pada Gambar 2 di bawah ini:


Gambar 2. Fase Proses *Attacker* Sistem

C. Diagram Proses Kerja Tool Open Web Aplication Security Project (OWASP).

*Penetration testing* pada *webserver* dengan melibatkan *tool Open Web Application Security Project* (OWASP) standart mode.




Gambar 3. Fase Proses *Attacker* Sistem

Berdasarkan Gambar 3. Menunjukan skema cara kerja dari *tool vega scanner vlnerability.*

Merupakan pola cara kerja dari *Vega Scanner framework* dalam melakukan serangan :

- *Safe Mode.*
- Standar atau Normal *Mode*.
- *Attack Stressing Mode*.

- Untuk melakukan eksploitasi pada *webserver* dalam upaya mencari kerentanan sistem[10].

- Hasil dari ujicoba pada *webserver* yang memiliki kerentanan atau *vulnerability* akan ditampilkan setelah 100% *scanning[*11].

- Solusi akan diberikan sebagai upaya memperbaiki sistem yang memiliki kerentanan atau *vulnerability[*12].

TABEL 1. POLA SERANGAN OLEH *TOOL* TERHADAP SISTEM APLIKASI *OPEN JOURNA SISTEM* (OJS)

| No | Teridetifiksi jenis serangan | keterangan |
|---|---|---|
| 1. | a. *Identification of target*<br>b. *Port scanning*<br>c. *System fingerprinting*<br>d. *Identification of vulnerability* | a. Mengidentifikasi sasaran.<br>b. Melakukan scanning pada setiap *port-port* terbuka yang bertujuan untuk mencari kerentanan pada *port webserver*.<br>c. Sistem parameter yang digunakan untuk menyimpulkan sistem operasi pada target.<br>d. Mengidentifikasi kerentanan yang dimiliki oleh sistem aplikasi *webserver*. |
| 2. | a. *Safe mode*<br>b. *Normal attack*<br>c. *Attack stressing* | Tingkat pola serangan atau exploitasi pada sistem yang ditujukan pada target. |
| 3. | *Webserver* | Kerentanan pada sistem dan jaringan |
| 4. | *Result collation and report writing* | Hasil *scanning* 100 % pada sistem jaringan. |
| 5. | *Recommend countermeasure* | Memberikan rekomendasi perbaikan sistem pada *webserver.* |

## 6. HASIL DAN PEMBAHASAN

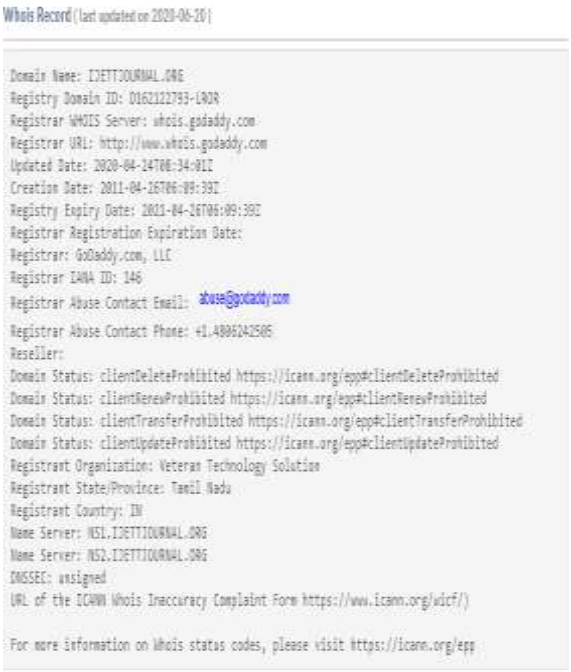
Simulasi yang dilakukan pada percobaan ini mengacu pada aplikasi *open journal system* (OJS) yang *real-time* menggunakan *tool Vega framework scanner* [13].

### A. Pengujian pada aplikasi Open Journal System (OJS).

**a. Information Gathering.**

Informasi berupa data-data baik berupa :

- Jaringan
- Domain
- Email
- Nama perusahaan
- Kemajuan suatu perusahaan atau lembaga, badan , isntitusi dan lain-lain




Gambar 4. Simulasi mencari informasi sebanyak-banyak mungkin mengenai target OJS.

Hasil scanning secara live scanning *menngunakan tool online* yang dpat disajikan pada bagian 4 dapat berupa

1) Nama *domain* dari *Journal* yang menjadi Target.
2) *Email* dari *server* apalikasi OJS.
3) *Server* (N1dan N2)

**b. Scanning Ip Address.**

Gambar 5. Simulasi mendapatkan *IP address* target.

Gambar 5. Merupakan hasil *scanning* pada IP *address OJS* menggunakan *whois domain tool.*

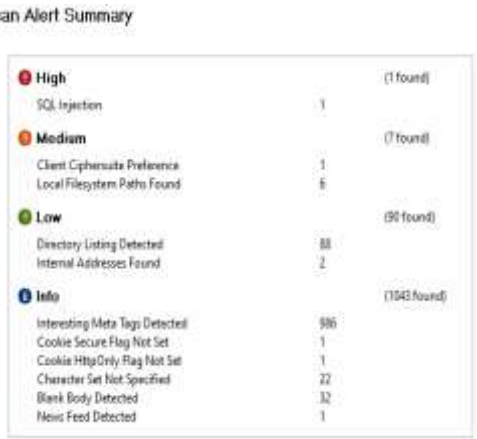
c. ***Scanning port (port discover)***

Gambar 6. Simulasi mendapatkan *Open Port discover*.

Gambar 6. Merupakan hasil *scanning* pada pada *port server* yang terbuka yang menggunakan *Nmap tool*. Mendeteksi adanya 12 *port* yang terbuka pada *server* dan 5 *port* yang terfilter.

**d. Teknik Scanning menggunakan *framework vega scanner.***

Hasil *scanner* oleh aplikasi *Vega canner* mendeteksi 4 jenis kerentanan pada aplikasi *Open Journal System* (OJS). Yang di tampilkan pada gambar di bawah sebagai berikut:




Gambar 7. Hasil scanning oleh vega scnner

Dari gambar 7. Dapat kita lihat ada nya 3 jenis kerentanan atau *vulnerability* pada OJS antara lain :

1. High risk
2. Medium risk
3. Low risk
4. Informasi tambahan

Keterangan gambar 6[15]. Dimana *high risk* mendeteksi 1 *file SQL Injection*, *medium risk* mendeteksi 7 *local filesystem paths found* dan *client cipersuit preference*, *low risk* mendeteksi 90 *file diantaranya directory listing detected* dan *internal addresses found.*

**B. Penjabaran hasil *scanning vulnerability***

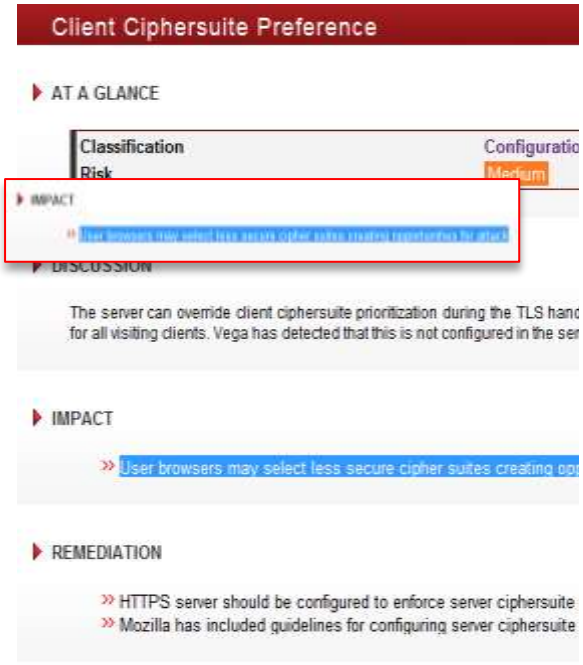
1. **High risk** ( risiko tinggi atau berbahaya) dengan risiko terkena *SQL _Injection* .

*Sql injection* merupkan serangkaian serang yang berbentuk *script* idak beraturan baik berbentuk huruf, angka, tanda, lambang dan lain sebagainya, yang akan mengganggu atau menyerang pada froam kolom *user name* dan *password.* Pada suatu *website* atau *webserver.*


Gambar 8. Dampak dari serangan *SQL_Injection*

Serangan *sql injection* memiliki risiko tinggi pada sistem *website* atau *webserver* sasaran utama yang menjadi target penting yaitu pada kolom *user name* dan *password* yang ditampilkan pada gambar 8 [16].

**2. Medium risk ( tingkat resiko menengah)**




Gambar 9. Dampak dari serangan *client ciphersuite preference.*

Serangan *client ciphersuite preference* dimana tingkat risiko menengah atau *medium* pada dampak bagi sistem *websiste* atau *webserver* sasaran utama yang menjadi target penting yaitu *setting* pada konfigurasi *server* berpotensi membahayakan bagi *user* lama, dimana ditampilkan pada gambar 9[17].

TABEL 2. HASIL SCANNING VEGA SCANNER

| *Vulnerability* atau kerentanan *file* sistem | Tingkat bahaya risiko pada *Open Journal Sistem* (OJS) | | |
|---|---|---|---|
| | *High Risk* | *Medium Risk* | *Low Risk* |
| SQL Injection | 1 | | |
| *Client Cipersuite preference* | | 1 | |
| *Local file system paths found* | | 6 | |
| *Directory Listing Detecting* | | | 88 |

| | | | |
|---|---|---|---|
| *Internal Address Found* | | | 2 |
| *Interesting Meta Tags Detected* | | | 986 |
| *Cookie secure flag not set* | | | 1 |
| *Cookie http only flag not set* | | | 1 |
| *Caracter set Not spesified* | | | 22 |
| *Blank body Detected* | | | 32 |
| *News feed detected* | | | 1 |

*Vega Framework Scanner* berhasil melakukan Audit pada *Open journal System* (OJS) .mendeteksi *beberapa* kerentanan terdapat pada *OJS* tersebut secara *realtime*.

**C. Countermeasure atau Rekomnedasi yang harus dilaksanakan.**

TABEL 3. COUNTERMEASURE

| No. | Nama file sistem | Tingkat resiko kerentanan | Jumlah file | Rekomendasi |
|---|---|---|---|---|
| 1 | *SQL Injection* | High risk / risiko tinggi (berbahaya) | 1 | Kesalahan kustom pada sumber halaman mengacu pada mekanisme untuk memberikan refrensi atau pengenal kesalahan Unix pada *server* dan tidak dapatnya di akses oeh user atau pengguna. |
| **2** | *Client Cipersuite preference* | Medium /sedang | 1 | Halaman yang berbentuk SSl atau *TLS.* Tidak bisa di kirim melalui *HTTP* yang tidak di enkripsi. Termasuk pada konten pihak ke tiga. |
| **3.** | Local file system paths found | Medium / sedang | 6 | *Web* moderen yang menggunakan *HTTP, X-frame options* terlebih dahulu di upayakan penyetingan pada halaman *web* yang terdapat pada (framest) atau ( *Deny. Allow* From)yang bertujuan sebagai pendukung pada *website*. |
| 4 | Directory Listing Detecting | Low / rendah | 88 | *Cookie* yang berisi informasi *Flag* yang sensitif sebaiknya di dienkriipsi terlebih dahulu |
| 5 | Internal Address Found | Low / rendah | 2 | *File javascript* harus mengacu pada |

| | | | | |
|---|---|---|---|---|
| | | | | sumber yang tepat agar tidak dapat di *Remote* oleh pihak yang tidak bertanggung jawab. |
| 6 | Interesting Meta Tags Detected | Low / rendah | 986 | Pastikan pada *header HTTP Control-cache* dan *header* bawah *HTTP* Pragma di seting tanpa *cache.* |
| 7 | Cookie secure flag not set | Mediu m / sedang | 1 | Matikan *atribut Autocmplete* dalam bentuk atau elemen yang berisi kata sandi atau *password*. |
| 8 | Cookie http only flag not set | Low / rendah | 1 | Halaman yang berbentuk *SSI* atau *TLS.* Tidak bisa dikirim melalui *HTTP* yang tidak di enkripsi termasuk pada konten pihak ke tiga |
| 9 | Caracter set Not spesified | Low / rendah | 22 | Proteksi *XSS browser web* harus diaktifkan pada *header HTTP x-xss protection* di webserver. |
| 10 | Blank body Detected | Low / rendah | 32 | Pastikan aplikasi *webserver* yang mengguna kan konten yang tepat. *X-content type-option* dirubah menjadi *nosniff* untuk semua halaman *web.* Pastikan penggunaa n *web* standar dan tidak melakukan *MIME-Sniffing.* |

1. Tabel 3. *Countermesure* merupakan solusi atau penanganan pada *OJS* yang telah diaudit menggunakan *Vega Scanner framework*. Serta mendeteksi kerentanan *OJS* atara lain:
   - *High risk*.
   - *Medium risk*.
   - *Low risk.*

2. Adapun penanganan nya berbeda-beda di setiap sub-sub file sistemnya.
   - *SQL Injection.*
   - *Client Cipersuite preference.*
   - *Local file system paths found.*
   - *Directory Listing Detecting.*
   - *Internal Address Found.*
   - *Interesting Meta Tags Detected.*
   - *Cookie secure flag not set.*
   - *Cookie http only flag not set.*
   - *Caracter set Not spesified.*
   - *Blank body Detected.*

3. Setiap 1 sub file sistem memiliki rekomendasi dapat dilihat pada Tabel 3.

## 7. Kesimpulan dan Saran

Ujicoba pada *Open Journal System* (OJS) menggunakan standar atau *normal mode*, dimana hasil ujicoba yang dilakukan, ditemukan kerentanan yang mencakup: *high risk, medium risk* sampai *low risk*. Semua teknik *penetration testing* ini mengacu pada teknik dasar penetrasi testing metode *BlackBox* yang melibatkan *tool vega framework scanner*, dalam menemukan *vulnerability* pada *webserver*. Adapun dampak yang ditimbulakan pada setiap *penetration testing* memiliki fariasi yang ditunjukan pada refrensi *vega framework scanner*. Hasil pengujian pada *OJS* memperoleh 1 kerentanan bersifat *high risk*, 7 *file* sistem *medium risk,* 90 *file* sistem *low risk* dengan informasi tambahan agar ditindak lanjuti dalam perbaikan berjulah 1043. Penelitian ini diharapkan, dapat dikembangkan lebih spesifik dengan objek-objeck yang berbeda.

**Daftar Pustaka:**

[1] Z. Yang, "A NEW METHOD FOR VULNERABILITY ANALYSIS AND APPLICATION IN RURAL DWELLINGS," 2019 Symp. Piezoelectrcity,Acoustic Waves Device Appl., no. 1, pp. 1–4, 2019.
[2] Kate Siccho " Hacking Choreography Dance and Live Coding". University of Lincoln. 2014.
[3] A. Mendoza and G. Gu, "Mobile Application Web API Reconnaissance : Web-to-Mobile Inconsistencies & Vulnerabilities," 2018 IEEE Symp. Secur. Priv., pp. 756–769, 2018.
[4] J. Hu et al., "A Memory-Related Vulnerability Detection Approach Based on Vulnerability Features," vol. 25, no. 5, pp. 604–613, 2020.
[5] N. Naik, P. Jenkins, N. Savage, and L. Yang, "Cyberthreat Hunting - Part 2 : Tracking Ransomware Threat Actors using Fuzzy Hashing and Fuzzy C-Means Clustering," 2019 IEEE Int. Conf. Fuzzy Syst., pp. 1–6, 2019.
[6] R. Madhusudhan, "Cross Channel Scripting ( XCS ) Attacks in Web Applications : Detection and Mitigation Approaches," 2018 2nd Cyber Secur. Netw. Conf., pp. 1–3, 2018.
[7] L. Pang, M. Yu, W. Yi, G. Jiang, W. Liu, and Z. Jiang, "Relativity Analysis-Based Error Concealment Algorithm for Entire Frame Loss of Stereo Video," 2006.
[8] S. Q. R. Codes, M. Yuan, S. Member, K. Liu, and S. Singamaneni, "Self-Powered Forward Error-Correcting Biosensor Based on Integration of Paper-Based Microfluidics," pp. 1–9, 2016.
[9] A. Schr and N. Bettenburg, "Do Stack Traces Help Developers Fix Bugs?". pp. 118–121, 2010.
[10] E. Crifasi, S. Pike, and Z. Stuedemann, "Cloud-Based Source Code Security and Vulnerabilities Analysis Tool for C / C ++ Software Systems," 2018 IEEE Int. Conf. Electro/Information Technol., pp. 651–654, 2018.
[11] M. Almousa, N. C. A, and T. State, "Predictive Analytics," 2019 17th Int. Conf. Privacy, Secur. Trust, pp. 1–3, 2019.
[12] A. Alzahrani, A. Alqazzaz, H. Fu, and N. Almashfi, "Web Application Security Tools Analysis," 2017.
[13] R. A. Khan, "Evaluating Performance of Web Application Security Through a Fuzzy Based Hybrid Multi-Criteria Decision-Making Approach: Design Tactics Perspective," vol. 8, 2020.
[14] S. Tyagi, "Evaluation of Static Web Vulnerability Analysis Tools," 2018 Fifth Int. Conf. Parallel, Distrib. Grid Comput., pp. 1–6, 2018.
[15] A. Shukla, B. Katt, and L. O. Nweke, "Vulnerability Discovery Modelling With Vulnerability Severity," 2019.
[16] L. K. Shar, D. Bianculli, L. Briand, and J. Thom, "An Integrated Approach for Effective Injection Vulnerability Analysis of Web Applications through Security Slicing and Hybrid Constraint Solving," vol. 5589, no. c, pp. 1–33, 2018.
[17] C. Ping, "A second-order SQL injection detection method," pp. 1792–1796, 2017.