Vol. 3 No 1 – April 2025
DOI: 10.59688/bufnets.v3i1.69

XSS *INJECTION* VULNERABILITY PADA OPEN JOURNAL SYSTEMS (OJS)

Ismail Puji Saputra[1], Arif Hidayat[2]

Fakultas Ilmu Komputer, Universitas Muhammadiyah Metro
Jl. Gatot Subroto No.100, Yosodadi Kota Metro Lampung, Indonesia

e-mail: ismailpujisaputra@gmail.com[1], androidarifhidayat@gmail.com[2]

Received : 01 April, 2025	Accepted : 01 April 2025	Published : 21 April 2025

Abstract

The Public Knowledge Project's Open Journal Systems (OJS) has several versions, with version 3.1.1.4 still widely used by institutions to facilitate the publication of scholarly work. Unfortunately, this version contains a security vulnerability—an XSS (Cross-Site Scripting) injection flaw. This XSS vulnerability can be exploited by cybercriminals to hijack the session of a journal manager and carry out illegal activities, such as installing a backdoor into the system. This study focuses on analyzing the attack flow and identifying methods to prevent such attacks. The findings show that the attack is carried out via the discussion panel, which only becomes available when a journal article enters the review process. During this stage, an attacker can add a discussion containing a script designed to steal the journal manager's session. This attack can be mitigated by applying proper input filtering to the discussion field.

Keywords: *Penetration Testing, OJS 3, XSS Attack, XSS Injection*

Abstrak

Open Journal System (OJS) dari PKP memiliki beberapa versi, terdapat versi 3.1.1.4 yang masih banyak digunakan oleh instansi dalam mengakomodir proses publikasi karya ilmiah, sayangnya terdapat sebuah celah keamanan pada OJS tersebut, celah tersebut adalah XSS Injection. Celah XSS Injection dapat dimanfaatkan oleh pelaku kejahatan cyber untuk mencuri sesi dari pengelola jurnal dan melakukan aktivitas illegal (memasang backdoor kedalam sistem). Penelitian ini akan berfokus pada menguji alur serangan serta menemukan cara dalam menangkal serangan tersebut. Hasilnya adalah serangan dilakukan dengan memanfaatkan kolom diskusi, kolom diskusi hanya akan terbuka ketika sebuah artikel *jurnal masuk kedalam proses review, pada saat proses ini Attacker dapat menambahkan diskusi yang mengandung* Skrip *untuk mencuri sesi dari pengelola jurnal, serangan ini dapat dicegah dengan melakukan filtering pada kolom tersebut.*

Kata Kunci: Penetration Testing, OJS 3, XSS Attack, XSS Injection

1. PENDAHULUAN

Open Journal Systems (OJS) adalah sebuah perangkat lunak manajemen dan penerbitan jurnal ilmiah berbasis web yang dikembangkan oleh *Public Knowledge Project (PKP)* [1-3]. OJS digunakan secara luas oleh institusi pendidikan, lembaga penelitian, serta organisasi ilmiah di seluruh dunia karena bersifat *open source*, fleksibel, dan mendukung seluruh proses penerbitan mulai dari pengajuan artikel hingga publikasi akhir [4-5]. Salah satu versi OJS yang masih banyak digunakan adalah versi 3.1.1-4. Versi ini dianggap stabil dan ringan, namun masih mengandung beberapa kerentanan keamanan yang belum sepenuhnya ditangani.

Salah satu celah keamanan yang terdapat pada OJS versi 3.1.1-4 adalah serangan *Cross Site Scripting (XSS) Injection* [6-7]. XSS merupakan jenis serangan terhadap aplikasi web yang memungkinkan pelaku untuk menyisipkan skrip jahat (*malicious* Skrip) ke dalam halaman web, yang kemudian dijalankan di sisi pengguna lain [8-9]. Serangan XSS banyak dimanfaatkan oleh pelaku kejahatan siber untuk mencuri data penting pengguna seperti *cookie*, token autentikasi, maupun sesi pengguna [10-11]. Hal ini sangat berbahaya, terlebih apabila serangan berhasil mencuri sesi dari pengguna dengan hak akses tinggi, seperti pengelola jurnal atau administrator jurnal.

OJS memiliki titik-titik input yang rawan terhadap XSS apabila tidak dilengkapi dengan validasi dan filterisasi input yang memadai. Pentingnya pembaruan sistem dan audit kode sumber secara berkala untuk mencegah eksploitasi oleh penyerang. Masih banyak institusi yang menggunakan versi OJS lama karena keterbatasan

teknis atau ketidaktahuan akan risiko keamanan yang terkandung di dalamnya [12-13].

Penelitian ini bertujuan untuk mengidentifikasi dan mengeksploitasi kerentanan XSS pada OJS versi 3.1.1-4, serta memberikan solusi pencegahan serangan berdasarkan hasil pengujian. Fokus utama adalah pada fitur *discussion* dalam proses *review*, di mana kolom diskusi menjadi celah yang memungkinkan penyerang menyisipkan skrip berbahaya. Diskusi ini hanya terbuka saat artikel masuk tahap *review*, sehingga pada saat itu penyerang dapat menambahkan komentar yang mengandung kode XSS. Skrip tersebut dapat digunakan untuk mencuri sesi pengguna lain dan membuka akses tidak sah ke dalam sistem.

Untuk mencapai tujuan tersebut, penelitian ini menggunakan pendekatan metode *Penetration Testing*, yaitu suatu metode sistematis untuk mengidentifikasi, mengeksploitasi, dan mengevaluasi kerentanan dalam sistem informasi. Metode *Penetration testing* melibatkan tahapan perencanaan, pengintaian (*reconnaissance*), pemindaian (*scanning*), eksploitasi, dan pelaporan [14-15]. Dengan pendekatan ini, diharapkan dapat diperoleh pemahaman menyeluruh mengenai skenario serangan serta rekomendasi teknis untuk mitigasi.

Hasil dari penelitian ini diharapkan dapat memberikan kontribusi dalam upaya meningkatkan keamanan sistem OJS, terutama bagi institusi yang masih menggunakan versi lama. Selain itu, penelitian ini juga dapat menjadi referensi bagi pengelola jurnal untuk lebih memperhatikan aspek keamanan dalam pengelolaan platform publikasi ilmiah.

2. METODE PENELITIAN

Metode penelitian kali ini menggunakan penetrasi testing dengan menggunakan metode *whitebox test*, hal ini dikarenakan telah diketahui bahwa OJS versi 3.1.1.4 memiliki celah keamanan XSS *Injection* pada fitur diskuisi, selain itu penulis juga memiliki akses yang sah kedalam server, sehingga proses serangan dan respon server dapat dilihat langsung.

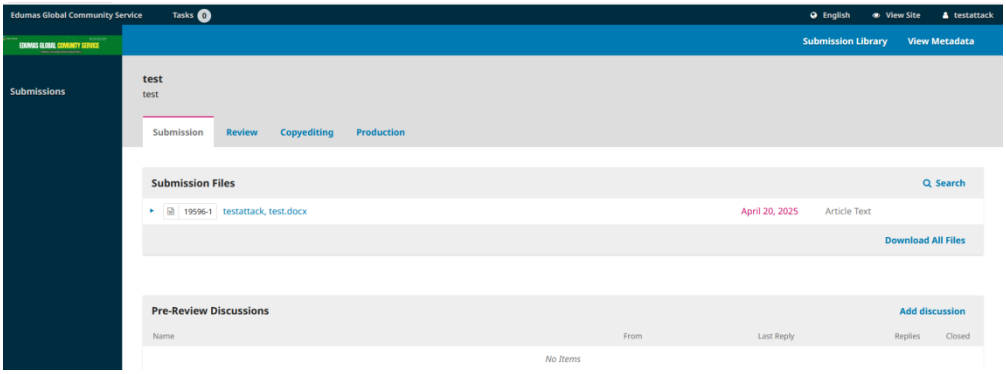
Proses diawali dengan melakukan submission artikel kedalam OJS, artikel akan masuk dalam proses *review*, pada proses ini *Attacker* akan mencoba untuk mengirimkan Skrip berbahaya pada form diskusi, pengelola jurnal akan mengeksekusi link diskusi.

Penelitian akan mengamati bagaimana proses sesi dicuri dan adakah respon dari *firewall* ketika serangan sedang terjadi, selanjutnya adalah mitigasi serangan dengan berbagai cara, sehingga penelitian akan mengetahui celah dan bagaimana cara menutup celah tersebut.

3. HASIL DAN PEMBAHASAN

3.1 Registrasi Akun dan *Submit* Artikel

Untuk dapat *submit* kedalam OJS maka hal yang harus dilakukan adalah melakukan registrasi sebagai *author* pada sebuah OJS, untuk melakukan penetrasi testing dan menguji apakah pengelola jurnal waspada pada setiap serangan, maka sebaiknya proses *submit* artikel dibuat seolah-olah sebagai jurnal asli, hal ini dilakukan agar artikel dapat masuk ke proses *review*, jika artikel yang di*submit* tidak sesuai *scope* dan terkesan asal, maka pengelola jurnal akan curiga dan melakukan penolakan pada artikel, sehingga serangan akan gagal. Berikut ini contoh proses *submit* artikel:



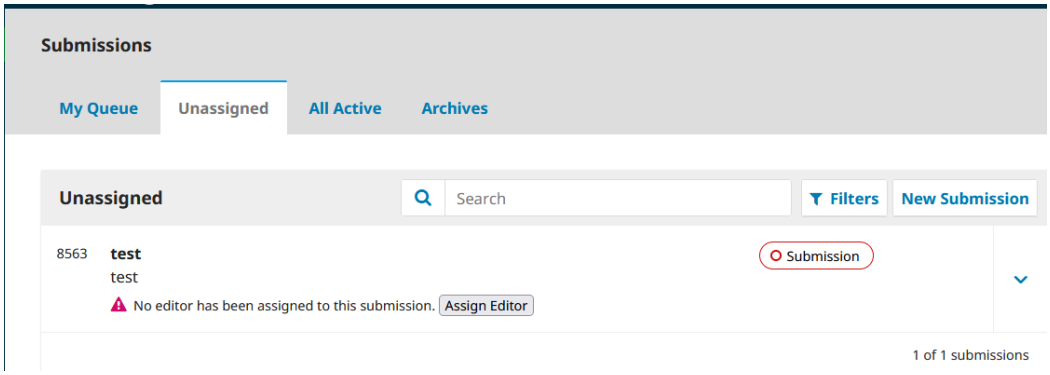
Gambar 1. Proses *Submit* Artikel
[Sumber: Penulis, 2025]

Setelah proses *submit*, maka pada halaman *dashboard* pengelola jurnal akan tampil bahwa terdapat seseorang yang *submit* artikel kedalam OJS, maka hal yang dilakukan pengelola jurnal adalah memeriksa *submit* tersebut. Jika artikel yang di *submit* sesuai dengan *scope* jurnal dan susunan penulisan artikel sesuai dengan *template* jurnal, maka biasanya pengelola jurnal akan memasukan artikel ke dalam proses *review*. Maka keberhasilan serangan ini tergantung dari artikel yang kita

submit, sebisa mungkin artikel menyesuaikan *template* sehingga dapat lanjut ke proses serangan selanjutnya, namun apabila pengelola jurnal menolak artikel yang di *submit*, maka serangan ini akan gagal.

3.2 Pengelola Jurnal Memproses Artikel

Jika artikel berhasil mengelabuhi pengelola jurnal, hal yang dilakukan pengelola adalah memilih *editor* untuk memproses artikel. Berikut ini adalah tampilan dari sisi pengelola jurnal:

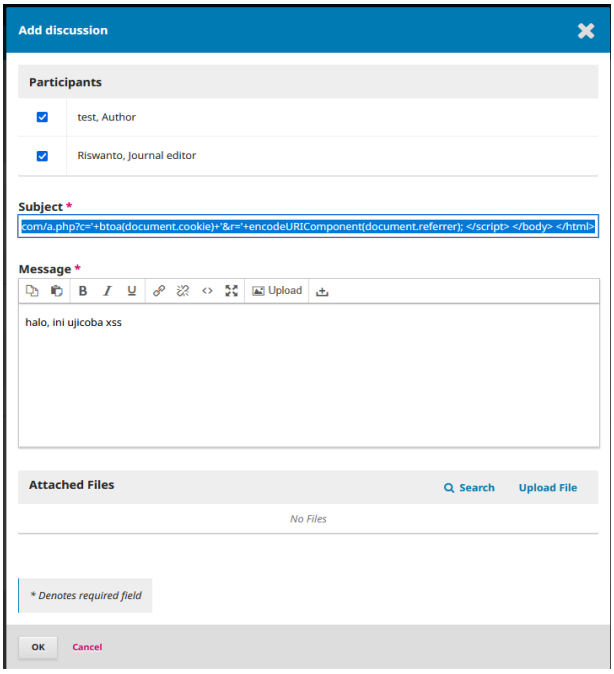


Gambar 2. Tampilan *Dashboard* Pengelola Jurnal
[Sumber: Penulis, 2025]

Pada gambar 2 terdapat tombol *Assign Editor*, tombol tersebut adalah aksi yang digunakan oleh pengelola jurnal untuk memilih *editor* untuk memproses artikel yang di*submit*, jika tombol tersebut di klik maka artikel akan diproses. Jika artikel telah diproses oleh pengelola dan mendapatkan *editor* untuk memproses artikel tersebut, maka proses serangan dapat dilanjutkan dengan menambahkan diskusi.

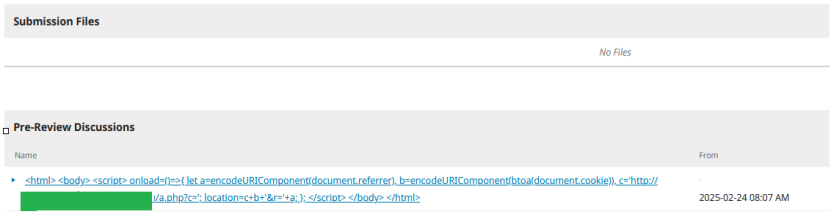
3.3 *Attacker* melakukan XSS *Injection*

Attacker hanya dapat membuat diskusi apabila artikel telah memiliki *editor*, *Attacker* akan menambahkan *editor* sebagai partisipan dalam sebuah diskusi. Berikut ini adalah tampilan dari sisi *Attacker* ketika membuat diskusi dengan judul yang menyisipkan serangan XSS *Injection*:



Gambar 3. Proses input Skrip XSS *Injection*
[Sumber: Penulis, 2025]

Setelah *Attacker* menginjeksi XSS ke diskusi maka tampilan diskusi akan menjadi seperti pada gambar dibawah ini:



Gambar 4. XSS *Injection* berhasil
[Sumber: Penulis, 2025]

Jika Skrip tersebut dilihat dan diklik oleh pengelola jurnal maka secara otomatis sesi pengelola jurnal akan dikirimkan ke server *Attacker*. Sesi sendiri dapat digunakan untuk masuk kedalam OJS tanpa login, sesi di input kedalam browser menggunakan plugin tertentu misalnya *cookie editor*. Skrip dapat dimanipulasi untuk mencegah deteksi *firewall*, teknik ini disebut dengan *evasion* yaitu menghindari *firewall*, caranya adalah dengan menggunakan *obfuscation* yaitu melakukan enkripsi pada Skrip XSS *Injection* [16-17].

3.4 Mitigasi Serangan

Beberapa mitigasi serangan dapat dilakukan, beberapa penelitian sebelumnya menyarankan untuk melakukan upgrade pada versi OJS yang lebih baru, namun proses upgrade OJS terkadang sangat kompleks dan dapat menambah masalah baru, maka penelitian ini menyarankan untuk melakukan mitigasi inputan pengguna. Langkah yang harus dilakukan adalah melakukan *filtering* pada form *subject discussion*. File sistem pada OJS yang perlu diperbaiki terletak pada *lib/pkp/controllers/grid/queries/form/QueryForm.inc.php* isi dari file tersebut dapat dilihat pada gambar dibawah ini:

```
327     function execute() {
328         $request = Application::getRequest();
329         $queryDao = DAORegistry::getDAO('QueryDAO');
330         $query = $this->getQuery();
331
332         $headNote = $query->getHeadNote();
333         $headNote->setTitle($this->getData('subject'));
334         $headNote->setContents($this->getData('comment'));
335
```

Gambar 5. Potongan Skrip QueryForm.inc.php
[Sumber: Penulis, 2025]

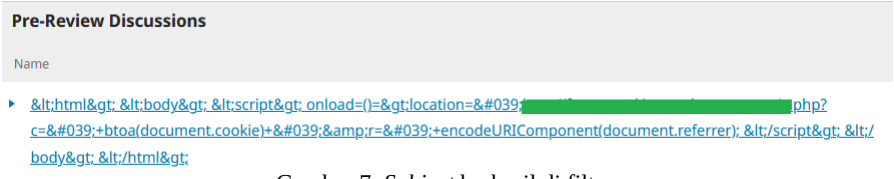
Pada variable *subject* yang ditunjukan pada baris 333 *subject* tidak memfilter HTML *special* karakter, sedangkan di baris 334 yang juga tidak memfilter HTML *special* karakter masih tetap aman, hal ini dikarenakan *filtering* telah dilakukan pada *text editor*.

Maka untuk melakukan mitigasi serangan tersebut adalah dengan menambahkan perintah *filtering* pada baris 333. Berikut ini adalah Skrip QueryForm.inc.php yang telah diperbaiki:

```
327     function execute() {
328         $request = Application::getRequest();
329         $queryDao = DAORegistry::getDAO('QueryDAO');
330         $query = $this->getQuery();
331
332         $headNote = $query->getHeadNote();
333         $headNote->setTitle(htmlspecialchars($this->getData('subject'), ENT_QUOTES, 'UTF-8'));
334         $headNote->setContents($this->getData('comment'));
```

Gambar 6. Potongan Skrip QueryForm.inc.php yang telah diperbaiki
[Sumber: Penulis, 2025]

Setelah proses perbaikan tersebut, maka ketika terdapat serangan serupa, judul akan menjadi *text* biasa, dan tidak dapat dieksekusi untuk mencuri sesi. Berikut ini adalah tampilan jika *filtering* telah berhasil:



Gambar 7. *Subject* berhasil di filter
[Sumber: Penulis, 2025]

4. KESIMPULAN

Penelitian ini membuktikan bahwa OJS versi 3.1.1-4 memiliki celah keamanan XSS *Injection* pada fitur diskusi saat proses *review*. Celah ini bisa dimanfaatkan penyerang untuk mencuri sesi pengguna, termasuk akun pengelola jurnal. Dengan metode penetratin testing, serangan berhasil disimulasikan dan menunjukkan risiko nyata terhadap sistem. Solusi yang diusulkan adalah menambahkan filter pada input diskusi untuk mencegah eksekusi Skrip. Langkah ini efektif mencegah serangan tanpa harus upgrade versi OJS. Penelitian ini diharapkan bisa membantu pengelola jurnal lebih waspada terhadap celah keamanan dan segera melakukan mitigasi.

DAFTAR PUSTAKA

[1] Wahyudi, E. (2024). Implementasi E-Journal berbasis Open Journal System (OJS) untuk Meningkatkan Jumlah Publikasi Penelitian Dosen IPDN Kampus NTB. Explore, 14(1), 35-41.

[2] Layaman, L., Wartoyo, W., & Ghoni, A. (2024). Pengelolaan Jurnal Ilmiah Berkala: Upaya Menuju Jurnal Terakreditasi Nasional dan Internasional. BERNAS: Jurnal Pengabdian Kepada Masyarakat, 5(2), 1670-1677.

[3] Fikri, O. M., Rukmana, E. N., & CMS, S. (2022). LAYANAN TATA KELOLA JURNAL DI PUSAT PENGELOLAAN PENGETAHUAN UNIVERSITAS PADJAJARAN. JPUA: Jurnal Perpustakaan Universitas Airlangga: Media Informasi & Komunikasi Kepustakawanan, 12(1).

[4] Silitonga, F., Cahayani, K., & Muthma'innah, M. (2024). Pendampingan Pembuatan dan Manajemen OJS Pendekar Sebagai Wadah Publikasi Hasil Pengabdian Kepada Masyarakat di Universitas Batam. SIGMA: Jurnal Sinergi Mengabdi, 1(2), 82-95.

[5] Darmanto, D., Ilfiani, P. D., Negara, K. M. T., Haryadi, W., & Satriawansyah, T. (2023). Pendampingan Tata Kelola Jurnal Ilmiah Online Berbasis Open Journal System (OJS) 3 Sesuai Standar Akreditasi Jurnal Nasional. Jurnal Pengembangan Masyarakat Lokal, 6(1), 125-131.

[6] Riadi, I., & Yudhana, A. (2020). Analisis Keamanan Website Open Journal System Menggunakan Metode Vulnerability Assessment. Jurnal Teknologi Informasi Dan Ilmu Komputer, 7(4), 853-860.

[7] Guntoro, G., Costaner, L., & Musfawati, M. (2020). Analisis Keamanan Web Server Open Journal System (Ojs) Menggunakan Metode Issaf Dan Owasp (Studi Kasus Ojs Universitas Lancang Kuning). JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika), 5(1), 45-55.

[8] Saputra, I. P. (2024). Mencegah Serangan Session Hijacking pada WEB PHP. SMART: Jurnal Teknologi Informasi dan Komputer, 3(2), 107-115.

[9] Lakshmi, B. S., Kovvuri, D., Bolisetti, H. V., Chikkala, D. S., Karri, S., & Yadlapalli, G. (2024, June). A proactive approach for detecting SQL and XSS injection attacks. In 2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC) (pp. 1415-1420). IEEE.

[10] Rodríguez-Galán, G., & Torres, J. (2024). Personal data filtering: a systematic literature review comparing the effectiveness of XSS attacks in web applications vs cookie stealing. Annals of Telecommunications, 79(11), 763-802.

[11] Alsaffar, M., Aljaloud, S., Mohammed, B. A., Al-Mekhlafi, Z. G., Almurayziq, T. S., Alshammari, G., & Alshammari, A. (2022). Detection of Web Cross-Site Scripting (XSS) Attacks. Electronics, 11(14), 2212.

[12] Arromdoni, B. U. H., Kusuma, M., & Sugiantoro, B. (2024). Web application vulnerability analysis using the owasp method (case study: ojs csfd uin sunan kalijaga yogyakarta). Engineering Headway, 6, 211-218.

[13] Mulyanto, Y., Zaen, M. T. A., Yuliadi, Y., & Sihab, S. (2022). Analisis Keamanan Website SMA Negeri 2 Sumbawa Besar Menggunakan Metode Penetration Testing (Pentest). Journal of Information System Research (JOSH), 4(1), 202-9.

[14] Tahir, M., & Ardiansyah, M. R. (2024). Analisis Keamanan Website Dinas Pemerintahan Yogyakarta Dengan Metode PTES (Penetration Testing Execution Standard). Jurnal Teknik Informatika UNIKA Santo Thomas, 118-125.

[15] Temara, S. (2023). Maximizing penetration testing success with effective reconnaissance techniques using chatgpt. arXiv preprint arXiv:2307.06391.

[16] Emmanuel, O. I., Ayodele, A. A., Adebiyi, A. M., & Osang, B. F. (2021, September). Windows firewall bypassing techniques: An overview of HTTP tunneling and Nmap evasion. In International Conference on Computational Science and Its Applications (pp. 546-556). Cham: Springer International Publishing.

[17] Abideen, Z. U., Gokulanathan, S., J. Aljafar, M., & Pagliarini, S. (2024). An overview of FPGA-inspired obfuscation techniques. ACM Computing Surveys, 56(12), 1-35.