Teknika 19 (1): 25-35

JURNAL TEKNIKA
ISSN: 0854-3143
e-ISSN: 2622-3481
Journal homepage: http://jurnal.polsri.ac.id/index.php/teknika
Journal Email: teknika@polsri.ac.id

Desain Kontrol Keamanan Pada *Content Management System Wordpress* Berdasar Aspek Aplikasi Dengan Panduan OWASP

Ivan Setiawan *[1], Adityas Widjajarto [2], Avon Budiyono [3]

[*1,2,3] Prodi Sistem Informasi, Universitas Telkom, Bandung, Indonesia
*Email Penulis Korespondensi: ivansetiawan@student.telkomuniversity.ac.id

Abstrak

Layanan CMS WordPress sangat populer di seluruh dunia, sehingga keamanan platform ini menjadi sangat penting. Penelitian ini bertujuan merancang desain kontrol keamanan aplikasi pada CMS WordPress berdasarkan eksploitasi kerentanan pada plugin dan non-plugin. Kerentanan yang dieksploitasi mencakup plugin MStore-API, Modern Event Calendar Lite, WPS-Hide-Login, Elementor, Catch Themes Demo Import, serta kerentanan XXE dan serangan Brute Force. Hasil analisis menghasilkan desain kontrol keamanan aplikasi WordPress berdasarkan ancaman dan kerentanan, mencakup ancaman terhadap data dan standar OWASP Top 10. Dari tujuh kerentanan yang diidentifikasi, lima masuk kategori disclosure dan dua dalam kategori alteration, dengan empat kategori OWASP Top 10. Setiap kerentanan diberikan CVE ID dan dinilai menggunakan sistem CVSS. Misalnya, CVE-2023-2732 pada Plugin MStore-API memiliki skor tertinggi (9.8, Critical), sedangkan CVE-2021-29447 (XXE) memiliki skor terendah (6.5, Medium). Desain kontrol keamanan berdasarkan OWASP Top 10 membantu menentukan prioritas. Contohnya pada Identification and Authentication Failures (A07:2021), MStore-API diklasifikasikan sebagai Critical dengan risiko disclosure, menekankan pentingnya penerapan mekanisme keamanan segera. Meskipun Brute Force termasuk dalam kategori OWASP yang sama dan diklasifikasikan sebagai Medium, fokus utama tetap pada kerentanan Critical terlebih dahulu.

Kata kunci— *Wordpress, desain kontrol keamanan, eksploitasi, kerentanan, plugin*

Abstract

The WordPress CMS service is widely popular, making its security paramount. This research aims to design an application security control framework for WordPress CMS by addressing vulnerabilities in both plugins and non-plugins. Exploited vulnerabilities include the MStore-API, Modern Event Calendar Lite, WPS-Hide-Login, Elementor, Catch Themes Demo Import plugins, as well as XXE vulnerabilities and Brute Force attacks. The analysis results in a WordPress security control design based on threats and vulnerabilities, focusing on data threats and OWASP Top 10 standards. Of the seven identified vulnerabilities, five fall into the disclosure category and two into the alteration category, covering four OWASP Top 10 categories. Each vulnerability is assigned a CVE ID and evaluated using the CVSS system. For example, CVE-2023-2732 in the MStore-API plugin has the highest score (9.8, Critical), while CVE-2021-29447

Online Available: September 13, 2024

(XXE) has the lowest score (6.5, Medium). The security control design based on OWASP Top 10 helps in prioritizing responses. In the Identification and Authentication Failures (A07:2021) category, MStore-API is classified as Critical with a disclosure risk, underscoring the need for immediate security measures. Although Brute Force is also categorized in the same OWASP category, it has a Medium vulnerability level, so the main focus remains on Critical vulnerabilities first.

Keywords— *Wordpress, design security control, exploitation, vulnerabilities, plugins*

1. PENDAHULUAN

Pengelolaan dan penyediaan konten digital menjadi kunci dalam berbagai aspek kehidupan, termasuk bisnis, pendidikan, hiburan, dan komunikasi. Mengelola konten digital bukan hal yang sederhana, inilah saatnya *Content Management System* (CMS) berperan penting dalam hal ini. CMS sebagai sistem yang dapat membuat, mengelola, dan menerbitkan konten *web* tanpa perlu *skill* pemrograman [1], [2]. Contoh *CMS* seperti *WordPress* menawarkan *user interface* yang mudah dipahami, membantu efisiensi dalam penciptaan dan pengelolaan halaman *web*.

WordPress adalah salah satu *Content Management System* yang sangat terkenal dan sering digunakan untuk membuat berbagai jenis situs *web*. *WordPress* merupakan *CMS* yang serbaguna, yang mampu digunakan untuk menciptakan beragam jenis situs *web*, seperti *blog*, situs *web* perusahaan, dan *platform e-commerce* [3], [4]. Namun, kepopuleran *WordPress* juga berarti bahwa *platform* ini menjadi sasaran potensial bagi para *hacker* yang mencari celah keamanan untuk melancarkan peretasan pada situs *web*.

Pada tahun 2023, sekitar 4,3% dari situs web *WordPress* mengalami peretasan, artinya hampir 1 dari 25 situs terpengaruh. Ini setara dengan sekitar 13.000 situs yang diserang setiap hari. Dalam setahun, diperkirakan sekitar 4,7 juta situs *WordPress* mengalami serangan peretasan. Sebagian besar kerentanan keamanan pada *WordPress* disebabkan oleh plugin dan tema. Pada tahun 2021, 99,42% dari semua kerentanan keamanan ditemukan pada tema dan *plugin*, dengan rincian 92,81% disebabkan oleh *plugin* dan 6,61% oleh tema [5].

Control adalah proses yang dirancang untuk untuk meningkatkan keamanan aplikasi dengan mengurangi potensi kerentanan [6]. Dalam konteks ini, *control* melibatkan identifikasi resiko, penilaian dampaknya, dan penerapan langkah-langkah untuk mengurangi kemungkinan terjadinya insiden keamanan atau dampaknya jika insiden tersebut terjadi. salah satu panduan yang dapat digunakan adalah *Open Web Application Security Project* (OWASP) yaitu merupakan komunitas global yang fokus pada penanganan resiko dan pemahaman keamanan aplikasi *web* [7].

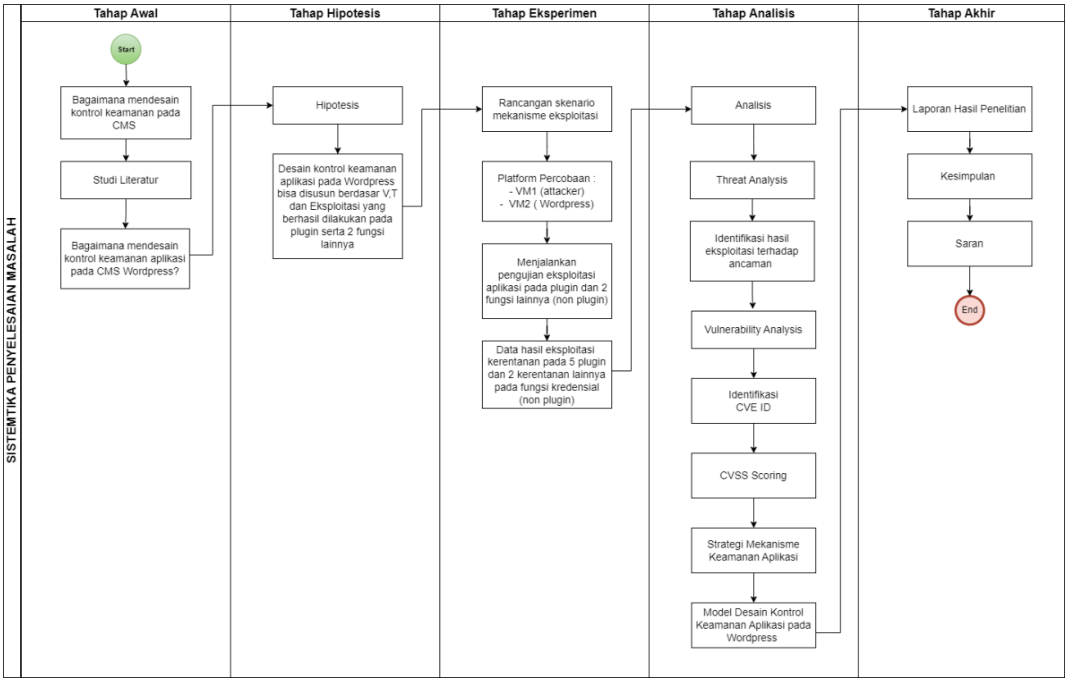
Keamanan situs *web* menjadi semakin penting. Desain kontrol keamanan pada *WordPress* memiliki dampak signifikan dalam meningkatkan aspek keamanan. Dengan pendekatan ini, pengguna *WordPress* dapat mengidentifikasi kebutuhan keamanan yang sesuai dengan tujuan bisnis, mengelola resiko yang terkait dengan situs *web*, dan menyesuaikan kontrol keamanan yang tepat. Melalui pengaturan kontrol yang relevan, seperti manajemen akses dan perlindungan terhadap eksploitasi, dapat menjaga dan meningkatkan tingkat keamanan situs *WordPress* sesuai dengan kebutuhan dan resiko yang dihadapi.

Dengan merancang desain kontrol keamanan yang baik, pengguna *WordPress* dapat memperkuat keamanan situs web melalui peningkatan lapisan keamanan yang dapat mengurangi kerentanan dan meminimalkan resiko peretasan. Penelitian ini menggunakan pendekatan OWASP, karena OWASP menyediakan standar komprehensif dan diakui secara internasional, seperti OWASP Top 10, yang fokus pada resiko aplikasi *web* yang paling kritis [8], [9]. Melalui pendekatan ini, situs *web* menjadi lebih tangguh dan dapat memberikan perlindungan yang lebih efektif terhadap serangan siber.

2. METODE PENELITIAN

2.1 Sistematika Penyelesaian Masalah

Dalam penelitian ini, diperlukan suatu metode untuk menyelesaikan masalah yang terdiri dari beberapa tahapan, dimulai dari tahapan awal hingga akhir. Proses penyelesaian masalah penelitian ini terdiri dari lima tahapan utama: Tahap Awal, Hipotesis, Tahap Eksperimen, Tahap Analisis, dan Tahap Akhir. Tahapan-tahapan ini dijelaskan secara rinci dan dapat dilihat pada Gambar 1.



Gambar 1 Sistematika Penyelesaian Masalah

2.1.1 Tahap Awal

Pada tahap awal penelitian, langkah pertama adalah metode untuk mendesain kontrol keamanan pada *CMS*. Ini dilanjutkan dengan melakukan riset melalui studi literatur terkait untuk memastikan relevansi masalah yang diteliti dan mendalami teori-teori terkait proses penrancangan desain kontrol keamanan. Selanjutnya, merumuskan perancangan desain kontrol keamanan aplikasi pada CMS *WordPress*.

2.1.2 Tahap Hipotesis

Setelah tahap awal selesai, penelitian dilanjutkan ke tahap berikutnya, yaitu perumusan hipotesis. Pada tahap ini, dilakukan penyusunan hipotesis yang menghasilkan asumsi bahwa desain kontrol keamanan aplikasi pada CMS *WordPress* dapat disusun berdasarkan *vulnerability, threat*, dan eksploitasi yang berhasil dilakukan pada *plugin* serta dua fungsi lainnya (*non-plugin*)

2.1.3 Tahap Eksperimen

Tahap berikutnya adalah tahap eksperimen. Pada tahap ini, langkah pertama adalah merancang skenario mekanisme eksploitasi, kemudian menyiapkan *platform* yang berperan sebagai *server* WordPress dan *attacker* pada *virtual mechine*. Setelah itu, dilakukan eksploitasi terhadap tujuh kerentanan pada *WordPress*, yang terdiri dari lima kerentanan pada *plugin* dan dua kerentanan pada fungsi lainnya (*non-plugin*). Setelah pengujian eksploitasi selesai, akan diperoleh data hasil eksperimen yang terperinci mengenai dampak dari setiap kerentanan yang diuji. Data hasil eksperimen ini nantinya akan digunakan untuk analisis lebih lanjut

2.1.4 Tahap Analisis

Pada tahap ini, dilakukan analisis terhadap hasil eksperimen. Data yang diperoleh dari pengujian akan dianalisis dengan fokus pada parameter-parameter seperti *vulnerbility*, *threat*, keberhasilan eksploitasi untuk mengidentifikasi ancaman keamanan, serta identifikasi CVE ID, CVSS *scoring*, strategi mitigasi, dan desain kontrol keamanan aplikasi. Pada *base metric* dalam CVSS *scoring*, dirancang untuk mencerminkan sifat mendasar dari kerentanan itu sendiri tanpa mempertimbangkan faktor eksternal seperti lingkungan di mana kerentanan berada atau bagaimana kerentanan dieksploitasi.

2.1.5 Tahap Akhir

Pada tahap akhir, output nya berupa sebuah laporan yang menggambarkan secara detail proses pengujian dan analisis sebelumnya. Laporan ini menjadi dokumen yang mencatat seluruh tahap uji coba serta evaluasi yang dilakukan. Selain itu, penarikan kesimpulan terkait dengan hasil pengujian serta pemberian saran akan dituliskan pada laporan hasil penelitian.

2.2 CMS Wordpress

Content Management System (CMS) adalah perangkat lunak yang memungkinkan pengguna membuat dan mengelola situs web tanpa memerlukan kemampuan teknis khusus seperti pemrograman [8]. *WordPress* adalah *platform* pembuatan situs *web open source* yang dapat dimodifikasi oleh siapa saja dan menawarkan *plugin* untuk memperluas fitur dengan mudah. Sejak dirilis pada tahun 2003 oleh Matt Mullenweg dan Mike Little, *WordPress* telah menjadi *CMS* paling populer, digunakan untuk 30% situs *web* di seluruh dunia karena kemudahan penggunaan, fitur lengkap, dan gratis. Penciptaan *WordPress* diawali oleh penutupan *software blogging b2/cafeblog*, yang mendorong Mike dan Matt untuk membuat platform serupa, sehingga pada tahun 2004 *WordPress* 1.0 dirilis [9].

2.3 Ancaman Keamanan Data

Ancaman keamanan data, yang dikenal sebagai DAD *triad*, meliputi tiga risiko utama: *Disclosure*, yaitu pengungkapan informasi yang seharusnya dirahasiakan; *Alteration*, yaitu perubahan, perusakan, atau manipulasi data; dan *Denial*, yaitu ketidak mampuan pengguna berwenang untuk mengakses data atau sistem [10]. DAD *triad* sendiri merupakan kebalikan dari CIA *triad*, yang meliputi *Confidentiality* (kerahasiaan data), *Integrity* (keakuratan data), dan Availability (aksesibilitas data) [11], [12]. Menerapkan prinsip CIA *triad* membantu memitigasi risiko dari DAD *triad* dan meningkatkan keamanan informasi.

2.4 CVE & CVSS

Common Vulnerabilities and Exposures (CVE) adalah daftar lengkap kerentanan keamanan pada produk, baik *software* maupun *firmware*, yang mencakup berbagai celah yang dapat menjadi target serangan siber [13], [14]. Sedangkan *Common Vulnerability Scoring System* (CVSS) merupakan sebuah pendekatan untuk memberikan penilaian kualitatif terhadap tingkat keseriusan suatu kerentanan. Perlu dicatat bahwa CVSS bukanlah alat untuk mengukur resiko. Dalam penerapannya, CVSS menggunakan tiga kategori metrik utama: *base*, *temporal*, dan *environmental* [15].

3. HASIL DAN PEMBAHASAN

3.1 Persiapan

Persiapan dan perancangan diperlukan untuk mencapai keberhasilan dalam penelitian eksperimen. Dibutuhkan perancangan sistem yang matang untuk melakukan pengujian. Perancangan ini membutuhkan arsitektur yang terdiri dari *hardware* (perangkat keras) dan *software* (perangkat lunak) yang dirancang khusus untuk membantu dalam pengumpulan informasi dan analisis data dalam penelitian ini.

3.2 Data Hasil Eksploitasi

Dalam pengujian eksploitasi terhadap tujuh kerentanan, yang terdiri dari lima *plugin* dan dua kerentanan *non-plugin*, hasil eksploitasi dapat dianalisis sebagaimana tertera dalam Tabel 1. Analisis ini memberikan gambaran yang jelas mengenai dampak dari kerentanan yang dieksploitasi.

Tabel 1 Data Hasil Eksploitasi

Eksploitasi	*Vulnerability*	*Tools Attack*	*Data Attack*	Hasil Eksploitasi
MStore-API *Plugin*	*Authentication Bypass Vulnerability*	Python	mstore-api.py	Mendapatkan akses ke *WordPress* dan masuk sebagai pengguna yang dipilih oleh penyerang berdasarkan ID pengguna.
Modern Event Calender Lite *Plugin*	*SQL Injection Vulnerability*	Python, Sqlmap	Mecl.py	Memperoleh data informasi MySQL seperti *username*, nama *database*, tabel, kolom, dan melakukan *dumping* data.
WPS-Hide-Login *Plugin*	*Security Bypass*	CMD	*Curl header referer*	*URL login* yang disembunyikan berhasil ditemukan.
XXE	*XXE Vulnerability*	echo command, http server, zlib_decode dan base64_decode	Payload .wav	Informasi MySQL WordPress target teridentifikasi, termasuk nama *database*, *password, dan username.*
Brute Force	*Weak password*	Wpscan	*Brute Force Password*	Mendapatkan akses masuk ke dalam akun pengguna WordPress yang berhasil diretas
Elementor *Plugin*	*Missing Capability Check Vulnerability*	Python, PHP, Netcat	Exploit.py, PHP reverse shell (Elementor-Pro.Zip)	Mendapatkan akses jarak jauh atau Eksekusi Kode Jarak Jauh (*Remote Code Execution*/RCE)
Catch Themes Demo Import *Plugin*	*Arbitrary File Upload Vulnerability*	Python, Webshell	50580.py	Mendapatkan akses jarak jauh atau Eksekusi Kode Jarak Jauh (*Remote Code Execution*/RCE)

Dari data Tabel 1, menggambarkan hasil eksploitasi dari beberapa kerentanan yang ditemukan dalam *plugin* dan *non-plugin*. Untuk MStore-API *plugin*, eksploitasi dilakukan melalui *Authentication Bypass Vulnerability* menggunakan Python dengan skrip "mstore-api.py", yang memungkinkan penyerang untuk mengakses *WordPress* dan masuk sebagai pengguna yang dipilih berdasarkan ID pengguna. Modern *Event Calendar* Lite *plugin* mengalami *SQL Injection Vulnerability*, yang dieksploitasi menggunakan *Python* dan *Sqlmap* dengan skrip "Mecl.py",

sehingga penyerang dapat memperoleh informasi MySQL seperti *username*, nama *database*, tabel, kolom, dan melakukan *dumping* data. WPS-Hide-Login *plugin* mengalami *Security Bypass*, di mana menggunakan *Curl header referer* melalui CMD, *URL login* yang disembunyikan berhasil ditemukan. XXE *Vulnerability* memanfaatkan file "payload.wav" dengan teknik *echo command*, *http server*, *zlib_decode*, dan *base64_decode*, yang mengungkapkan informasi *MySQL WordPress* target termasuk nama *database*, *password*, dan *username*. Metode *Brute Force* pada *password* yang lemah menggunakan *Wpscan* berhasil memperoleh akses ke akun pengguna *WordPress* yang diretas. Elementor *plugin* mengalami *Missing Capability Check Vulnerability*, yang dieksploitasi menggunakan *Python*, *PHP*, dan *Netcat* dengan skrip "Exploit.py" dan PHP *reverse shell* (*Elementor-Pro.Zip*), memungkinkan akses jarak jauh atau *Remote Code Execution* (RCE). Terakhir, *Catch Themes Demo Import plugin* dengan *Arbitrary File Upload Vulnerability* dieksploitasi menggunakan *Python* dan *Webshell* dengan skrip "50580.py", yang juga menghasilkan akses jarak jauh atau *Remote Code Execution* (RCE).

3.3 Analisis Ancaman Kemanan Data

Dari sudut pandang penyerang, ada tiga hal yang ingin dilakukan terhadap data, yaitu *Disclosure*, *Destruction*, dan *Denial*. Pada Tabel 2 adalah tabel yang mengkategorikan ancaman terhadap keamanan data berdasarkan eksploitasi yang telah dilakukan.

Tabel 2 Analisis Ancaman Data

Eksploitasi	Kategori	Alasan
MStore-API *Plugin*	*Disclosure*	Mengakses akun *user* termasuk admin yang seharusnya dirahasiakan.
Modern Event Calender Lite *Plugin*	*Disclosure*	Melihat data seperti *username* MySQL, nama *database*, tabel, kolom, dan melakukan *dump* data, ini mengungkapkan informasi yang seharusnya dirahasiakan.
WPS-Hide-Login *Plugin*	*Disclosure*	Menemukan *URL login* yang disembunyikan, sehingga informasi yang seharusnya dirahasiakan menjadi terungkap.
XXE	*Disclosure*	Akses tidak sah ke file internal *server* melalui unggahan file yang dimanipulasi, yang seharusnya dirahasiakan.
Brute Force	*Disclosure*	Mendapatkan *password* pengguna, sehingga informasi yang seharusnya dirahasiakan menjadi terungkap.
Elementor *Plugin*	*Alteration*	Dengan *remote code execution*, dapat merusak atau menghapus data situs atau bahkan mengambil alih kontrol.
Catch Themes Demo Import *Plugin*	*Alteration*	Dengan *remote code execution*, dapat merusak atau menghapus data situs atau bahkan mengambil alih kontrol.

3.4 Analisis Ancaman Menggunakan Standar OWASP

Dalam mengidentifikasi dan mengatasi ancaman terhadap keamanan, menggunakan standar yang telah diakui seperti OWASP Top 10 sangat penting. OWASP Top 10 menyediakan panduan yang lengkap untuk memahami risiko keamanan yang paling kritis yang dapat memengaruhi aplikasi *web*, yang dapat dilihat pada Tabel 3.

Tabel 3 Analisis Ancaman OWASP TOP 10

Eksploitasi	OWASP *Categories*	Alasan
MStore-API *Plugin*	*Identification and Authentication Failures* (A07:2021)	Kerentanan *bypass* otentikasi memungkinkan penyerang yang tidak terautentikasi mengakses akun pengguna.

Eksploitasi	OWASP *Categories*	Alasan
Modern Event Calender Lite *Plugin*	*Injection* (A03:2021)	Masalah keamanan akibat kurangnya validasi parameter pada pernyataan SQL memungkinkan serangan SQL *injection.*
Elementor *Plugin*	*Broken Access Control* (A01:2021)	Ketidakmampuan memeriksa izin pengguna pada beberapa tindakan AJAX memungkinkan perubahan data dan eksekusi kode berbahaya.
WPS-Hide-Login *Plugin*	*Broken Access Control* (A01:2021)	Kerentanan memungkinkan pengguna yang tidak terautentikasi mendapatkan akses ke halaman *login* yang seharusnya tersembunyi.
Catch Themes Demo Import *Plugin*	*Security Misconfiguration* (A05:2021)	Kurangnya validasi tipe file memungkinkan unggahan file berbahaya, berpotensi menyebabkan RCE.
Brute Force	*Identification and Authentication Failures* (A07:2021)	Serangan *brute force* disebabkan oleh kebijakan *password* yang lemah dan kurangnya mekanisme perlindungan.
XXE	*Security Misconfiguration* (A05:2021)	Konfigurasi buruk pada penerjemah XML memungkinkan serangan XXE yang dapat mengekspos data sensitif.

3.5 Analisis Kerentanan

Langkah pertama dalam analisis kerentanan adalah mengidentifikasi kerentanan dan penentuan nomor *Common Vulnerabilities and Exposures* (CVE) yang sesuai dilakukan dengan merujuk pada database CVE yang dikelola oleh MITRE, di mana setiap CVE memiliki deskripsi unik tentang kerentanan dan dampaknya. Setelah itu, tingkat keparahan kerentanan dinilai menggunakan *Common Vulnerability Scoring System* (CVSS), yang memberikan skor penting untuk menentukan prioritas penanganan. Skor ini biasanya diperoleh melalui *National Vulnerability Database* (NVD) atau CVSS *calculator*. Pada Tabel 4, terdapat dua kerentanan dengan tingkat keparahan *critical*, tiga dengan tingkat keparahan *high*, dan dua dengan tingkat keparahan *medium*.

Tabel 4 *Vulnerability Assessment*

Eksploitasi	*Vulnerability*	CVE	*Score*	*Level*
MStore-API *Plugin*	*Authentication Bypass Vulnerability*	CVE-2023-2732	9.8	*Critical*
Modern Event Calender Lite *Plugin*	*SQL Injection Vulnerability*	CVE-2021-24946	9.8	*Critical*
Elementor *Plugin*	*Missing Capability Check Vulnerability*	CVE-2022-1329	8.8	*High*
WPS-Hide-Login *Plugin*	*Security Bypass*	CVE-2021-24917	7.5	*High*
Catch Themes Demo Import *Plugin*	*Arbitrary File Upload Vulnerability*	CVE-2021-39352	7.2	*High*
Brute Force	*Weak Password*	CVE-2022-0828	6.9	*Medium*
XXE	*XXE Vulnerability*	CVE-2021-29447	6.5	*Medium*

3.6 Strategi Mekanisme Keamanan

Strategi mekanisme keamanan sangat penting dilakukan untuk mencegah kerentanan agar tidak dieksploitasi kembali, yang dapat dilihat pada Tabel 5.

Tabel 5 Streategi Mekanisme Keamanan

Eksploitasi	*Level*	*Security Mechanism*
MStore-API *Plugin*	*Critical*	Memasang Wordfence *Plugin* untuk dapat memantau dan membatasi akses REST API sehingga memblokir pengambilan *username*.
Modern Event Calender Lite *Plugin*	*Critical*	Menggunakan *Web Application Firewall* (WAF) - untuk melindungi database.
Elementor *Plugin*	*High*	Menginstall Wordfence *plugin* dan mengaktifkan *firewall*
WPS-Hide-Login *Plugin*	*High*	• Membatasi akses ke file options.php dengan menggunakan file .htaccess. • Menggunakan *plugin* lain seperti *iThemes Security*
Catch Themes Demo Import *Plugin*	*High*	Menginstall Wordfence *plugin* dan mengaktifkan *firewall*.
Brute Force	*Medium*	Mengganti *URL login* Batasi akses ke halaman *login* dengan htaccess Mengaktifkan autentikasi 2 faktor (2fa).
XXE	*Medium*	Tambahkan kode filter MIME untuk memeriksa dan memvalidasi jenis file yang diunggah. Untuk melindungi dari serangan XXE

3.6 Desain Kontrol Keamanan

Untuk memastikan keamanan *WordPress* secara menyeluruh, perlu dilakukan desain kontrol keamanan yang sistematis dan terstruktur. Desain kontrol ini bertujuan untuk mengidentifikasi dan mengatasi kerentanan yang ada dengan memperhatikan berbagai jenis ancaman yang mungkin mengarah pada eksploitasi. Melalui pendekatan ini dapat merancang dan menerapkan mekanisme keamanan yang efektif, sesuai dengan kategori *OWASP* yang relevan, untuk melindungi sistem dari potensi ancaman.

Tabel 6 Desain Kontrol Keamanan

OWASP Top 10 *Categories*	Eksploitasi	*Level*	Jenis Ancaman	Mekanisme Keamanan
Broken Access Control (A01:2021)	Elementor *Plugin*	*High*	*Alteration*	Menginstall Wordfence *plugin* dan mengaktifkan *firewall*
	WPS-Hide-Login *Plugin*	*High*	*Disclosure*	• Membatasi akses ke file options.php dengan menggunakan file .htaccess. • Menggunakan *plugin* lain seperti iThemes Security • Diatur secara manual
Injection (A03:2021)	Modern Event Calender Lite *Plugin*	*Critical*	*Disclosure*	Menggunakan *Web Application Firewall* (WAF) - untuk melindungi database

OWASP Top 10 *Categories*	Eksploitasi	*Level*	Jenis Ancaman	Mekanisme Keamanan
Security Misconfiguration (A05:2021)	Catch Themes Demo Import *Plugin*	*High*	*Alteration*	Menginstall Wordfence *plugin* dan mengaktifkan *firewall*
	XXE	*Medium*	*Disclosure*	Tambahkan kode filter MIME untuk memeriksa dan memvalidasi jenis file yang diunggah. Untuk melindungi dari serangan XXE
Identification and Authentication Failures (A07:2021)	MStore-API *Plugin*	*Critical*	*Disclosure*	Memasang Wordfence *Plugin* agar dapat memantau dan membatasi akses REST API sehingga memblokir pengambilan *username*
	Brute Force	*Medium*	*Disclosure*	Mengganti *URL login*, membatasi akses ke halaman *login* dengan htaccess, mengaktifkan autentikasi 2 faktor (2fa)

Pada Tabel 6, dijelaskan mengenai desain kontrol keamanan yang diterapkan untuk mengatasi kerentanan berdasarkan kategori OWASP dan jenis ancaman yang ada. Tabel ini mencakup berbagai eksploitasi dengan tingkat keparahan yang berbeda, serta mekanisme keamanan yang diimplementasikan untuk mitigasi. Kategori OWASP Top 10 dipilih karena merupakan daftar risiko keamanan paling signifikan yang dihadapi aplikasi *web* saat ini. Penilaian tingkat keparahan kerentanan dalam setiap kategori OWASP menunjukkan prioritas mekanisme keamanan yang harus diterapkan terlebih dahulu untuk menangani ancaman dengan tingkat keparahan yang lebih tinggi.

Contohnya, pada kategori *Security Misconfiguration* (A05:2021), *Catch Themes Demo Import Plugin* dan XXE menunjukkan bahwa kontrol seperti menginstal *Wordfence Plugin* dan menambahkan kode filter MIME harus diterapkan sesuai dengan tingkat keparahan kerentanannya. Catch Themes Demo Import *Plugin* yang memiliki *level High* perlu diprioritaskan lebih dahulu dibandingkan dengan XXE yang memiliki *level Medium*.

Demikian juga, pada kategori *Identification and Authentication Failures* (A07:2021), MStore-API *Plugin* diklasifikasikan sebagai *Critical* dengan ancaman *disclosure*. Ini menunjukkan bahwa mekanisme keamanan, seperti pemasangan *Wordfence Plugin* untuk memantau dan membatasi akses *REST API*, harus menjadi prioritas utama. Sementara itu, *Brute Force* dalam kategori yang sama dengan tingkat keparahan *Medium*, juga termasuk dalam kategori *Identification and Authentication Failures*. Meskipun penting, mekanisme keamanan seperti mengganti URL login, membatasi akses dengan htaccess, dan mengaktifkan autentikasi dua faktor (2FA) dianggap sebagai langkah-langkah mitigasi yang perlu dilakukan setelah menangani kerentanan yang lebih *Critical*.

Desain kontrol yang terstruktur ini memastikan bahwa setiap kerentanan ditangani sesuai dengan tingkat keparahannya, dengan fokus pada penerapan mekanisme keamanan yang efektif untuk melindungi sistem dari potensi ancaman.

4. KESIMPULAN

Berdasarkan analisis yang telah dilakukan terhadap kerentanan yang dieksploitasi pada bagian sebelumnya, penelitian ini menghasilkan kesimpulan sebagai berikut:

1. Pengujian eksploitasi kerentanan dilakukan pada tujuh kerentanan, Tipe eksploitasi yang diuji meliputi *Authentication Bypass* (MStore-API), *SQL Injection* (*Modern Event Calendar*

Lite), *Security Bypass* (WPS-Hide-Login), *Remote Code Execution* (*Elementor dan Catch Themes Demo Import*), serta kerentanan XXE dan serangan *Brute Force*.

2. Penyusunan desain kontrol keamanan disusun berdasarkan analisis ancaman dan kerentanan. Analisis ancaman mencakup ancaman data dan standar OWASP, dengan lima kerentanan dalam kategori *disclosure* dan dua dalam kategori *alteration*. Dari tujuh kerentanan, kerentanan tersebut termasuk empat kategori OWASP Top 10: *Broken Access Control* (A01:2021*), Injection* (A03:2021), *Security Misconfiguration* (A05:2021) dan *Identification and Authentication Failures* (A07:2021). Dalam analisis kerentanan, setiap kerentanan ditentukan CVE ID-nya dan dinilai menggunakan CVSS. CVE-2023-2732 pada *Plugin* MStore-API memiliki skor tertinggi yaitu 9.8 (*Critical*), sedangkan CVE-2021-29447 (XXE) memiliki skor terendah yaitu dengan skor CVSS 6.5 (*Medium*).
3. Desain kontrol keamanan berdasarkan kategori OWASP Top 10 membantu menentukan prioritas implementasi mekanisme keamanan. Seperti contoh pada kategori *Security Misconfiguration* (A05:2021), *Catch Themes Demo Import Plugin* dan XXE menunjukkan bahwa kontrol seperti menginstal *Wordfence Plugin* dan menambahkan kode filter MIME harus diterapkan sesuai dengan tingkat keparahan kerentanannya. *Catch Themes Demo Import Plugin* yang memiliki *level High* perlu diprioritaskan lebih dahulu dibandingkan dengan XXE yang memiliki *level Medium*.

5. SARAN

Berdasarkan hasil analisia penelitian, beberapa saran yang dapat dijadikan peluang lanjutan dari penelitian ini sebagai berikut:

1. Penelitian lebih lanjut dapat fokus pada teknik analisis *white-box* yang mengeksplorasi internal *source code* dari plugin dan aplikasi CMS WordPress.
2. Penelitian berikutnya dapat mengeksplorasi kategori serangan lain, seperti serangan CSRF, SSRF, dan jenis serangan lainnya.
3. Selain menggunakan CVSS *scoring* untuk perangkingan kerentanan, analisis *attack tree* juga dapat dilakukan dengan mempertimbangkan metrik seperti *time* dan *cost.*

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tim Redaksi Jurnal Teknika Politeknik Negeri Sriwijaya yang telah memberi memberi kesempatan, sehingga artikel ilmiah ini dapat diterbitkan.

DAFTAR PUSTAKA

[1] Techjury, “What Percentage of Websites are WordPress in 2024?” Accessed: Aug. 20, 2024. [Online]. Available: https://techjury.net/blog/percentage-of-wordpress-websites/

[2] R. Subariah *et al.*, “PELATIHAN PEMBUATAN WEBSITE MENGGUNAKAN CMS (CONTENT MANAGEMENT SYSTEM) JOOMLA PADA SMK BINA PUTRA MANDIRI,” 2021. [Online]. Available: https://dmi-journals.org/jai/

[3] Md. Abu Naim Heera, “WordPress As A CMS,” *CENTRIA UNIVERSITY OF APPLIED SCIENCES*, May 2019.

[4] A. Price, “WORDPRESS A MARKETING MACHINE,” 2016.

[5] MoonThemes, “53 WordPress Security & Hacking Statistics in 2023,” MoonThemes. Accessed: Aug. 06, 2024. [Online]. Available: https://moonthemes.com/53-wordpress-security-hacking-statistics-in-2023/

[6] A. Mishra, Y. I. Alzoubi, A. Q. Gill, and M. J. Anwar, “Cybersecurity Enterprises Policies: A Comparative Study,” *Sensors*, vol. 22, no. 2, Jan. 2022, doi: 10.3390/s22020538.

[7] K. A. Sedek, N. Osman, M. N. Osman, and Hj. K. Jusoff, “Developing a Secure Web Application Using OWASP Guidelines,” *Computer and Information Science*, vol. 2, no. 4, Oct. 2009, doi: 10.5539/cis.v2n4p137.

[8] K. Dhiatama Ayunda, A. Widjajarto, and A. Budiyono, "IMPLEMENTASI DAN ANALISIS OPEN SOURCE MODSECURITY WAF PADA APLIKASI BERBASIS WEB DENGAN STANDAR OWASP IMPLEMENTATION AND ANALYSIS OF OPEN SOURCE MODSECURITY WAF IN WEB-BASED APPLICATION WITH OWASP STANDARDS," 2021.

[9] A. A. A. W. Wafiuddin Akbar, "ANALISIS KERENTANAN WEBSITE XYZ PADA SITUS LAYANAN PENGADAAN SECARA ELEKTRONIK MENGGUNAKAN STANDAR OWASP TOP 10," 2021.

[10] R. S. Prabowo and A. Budi, "PEMBUATAN MEDIA PEMBELAJARAN BERBASIS BROWSER TRAINING DENGAN MENGGUNAKAN SOFTWARE CONTENT MANAGEMENT SYSTEM JOOMLA PADA MATA DIKLAT PEMELIHARAAN/SERVIS TRANSMISI MANUAL DAN KOMPONEN," 2009.

[11] A. Diana, D. Retno Utari, J. Raya, P. Utara, and K. Jakarta Selatan, "Implementasi Website E-Commerce Berbasis Content Management System Wordpress Pada Toko Pesona Tanaman," 2021.

[12] Jason. Andress, *Foundations of information security : a straightforward introduction*. No Starch Press, 2019.

[13] M. M. Alhassan, A. Adjei-Quaye, and M. Mahfouz Alhassan, "Information Security in an Organization," *International Journal of Computer*, 2017, [Online]. Available: https://www.researchgate.net/publication/314086143

[14] R. Vansuri *et al.*, "Peran CIA (Confidentiality, Integrity, Availability) Terhadap Manajemen Keamanan Informasi," vol. 2, no. 1, doi: 10.38035/jim.v2i1.

[15] K. Kanakogi *et al.*, "Tracing CVE vulnerability information to capec attack patterns using natural language processing techniques," *Information (Switzerland)*, vol. 12, no. 8, Aug. 2021, doi: 10.3390/info12080298.

[16] Olli Huuhtanen, "THE USE OF CVE-RELATED DATABASES IN IMPROVING THE CYBERSECURITY OF EMBEDDED SYSTEMS UNIVERSITY OF JYVÄSKYLÄ FACULTY OF INFORMATION TECHNOLOGY," 2021.

[17] R. Rizqllah, P. Saputra, Y. Purwanto, and M. F. Ruriawan, "UJI KERENTANAN PADA SISTEM PROCTORING UJIANBERBASIS LEARNING MANAGEMENT SYSTEM," 2023.