Mengoptimalkan Header Keamanan pada Website Server OpenLiteSpeed Menggunakan Hardening Berbasis OWASP

Sugiyatno*[1] , Untung Subagyo[2],
[1] *Informatika, STMIK El Rahma Yogyakarta*
[2] *Sistem Informasi, STMIK El Rahma Yogyakarta*
*e-mail: *[1]enoyat@gmail.com, [2]untungsubagyo@stmikelrahma.ac.id*

Abstrak

Keamanan layanan web merupakan aspek penting dalam menjaga integritas, kerahasiaan, dan ketersediaan data pada era digital. Server web yang tidak dikonfigurasi dengan baik rentan terhadap berbagai ancaman seperti Injection, Cross-Site Scripting (XSS), dan brute force attack. Salah satu server web yang banyak digunakan karena performanya adalah OpenLiteSpeed, namun konfigurasi defaultnya masih berpotensi menimbulkan celah keamanan. Kondisi ini menunjukkan urgensi penerapan mekanisme pengamanan yang terstruktur dan sesuai standar internasional, seperti metode hardening berbasis OWASP. Penelitian ini bertujuan meningkatkan keamanan layanan web pada server OpenLiteSpeed berbasis Ubuntu Server melalui penerapan hardening dengan fokus pada optimasi security header. Kebaruan penelitian terletak pada penerapan custom security header optimization, yaitu pengaturan Content-Security-Policy (CSP), X-Frame-Options, dan HTTP Strict Transport Security (HSTS) yang disesuaikan dengan kebutuhan aplikasi. Data penelitian diperoleh melalui identifikasi kerentanan awal dan pengujian keamanan menggunakan OWASP ZAP sebelum dan sesudah implementasi hardening. Metode yang digunakan adalah pendekatan eksperimental dengan tahapan identifikasi kerentanan, konfigurasi firewall, implementasi SSL/TLS, pengaturan hak akses, optimasi security header, dan analisis komparatif hasil pengujian. Hasil penelitian menunjukkan bahwa penerapan hardening berbasis OWASP mampu menurunkan jumlah serta tingkat keparahan kerentanan secara signifikan, sehingga efektif meningkatkan keamanan layanan web dan dapat dijadikan referensi implementasi keamanan server berbasis Linux.

Kata kunci: *OpenLiteSpeed, OWASP Hardening, Security Header Optimization*

1. PENDAHULUAN

Kemajuan teknologi informasi yang cepat telah mendorong masyarakat untuk lebih banyak menggunakan layanan web sebagai sarana utama dalam menyampaikan informasi dan layanan digital. Berbagai bidang seperti pendidikan, perdagangan, dan pemerintahan sangat tergantung pada adanya layanan web yang handal dan aman. Salah satu masalah utama yang kerap muncul adalah kurangnya pengaturan keamanan di server web, yang membuat sistem terpapar berbagai jenis serangan, seperti Injection, Broken Authentication, Security Misconfiguration, dan Cross-Site Scripting (XSS). Salah satu penyebab utama tingginya risiko keamanan server adalah penggunaan konfigurasi default pada web server tanpa penerapan hardening yang memadai. Banyak administrator server masih melakukan konfigurasi keamanan secara manual tanpa mengacu pada standar keamanan tertentu sehingga berpotensi menimbulkan celah keamanan baru. Kondisi tersebut juga ditemukan pada penggunaan web server OpenLiteSpeed yang dikenal ringan dan memiliki performa tinggi, namun tetap memiliki potensi kerentanan apabila tidak dikonfigurasi secara aman. Selain itu, penerapan sistem operasi Ubuntu sebagai server juga memerlukan penguatan konfigurasi agar mampu memenuhi standar keamanan layanan web modern.Hal ini menunjukkan bahwa banyak sistem yang belum mengimplementasikan standar keamanan secara maksimal. Berdasarkan uraian di atas, maka penting dilakukan optimasi header keamanan pada website, urgensi penelitian ini semakin penting karena hingga saat ini implementasi optimasi security headers seperti Content-Security-Policy (CSP), HTTP Strict Transport Security (HSTS), X-Frame-Options, dan X-Content-Type-Options masih belum banyak diterapkan secara optimal pada server berbasis OpenLiteSpeed. Padahal, security headers

memiliki peran penting dalam mitigasi serangan XSS, clickjacking, MIME sniffing, dan berbagai bentuk eksploitasi berbasis browser. Oleh karena itu, diperlukan penelitian yang mampu menerapkan hardening server berbasis standar OWASP secara terstruktur dan terukur guna meningkatkan keamanan layanan web pada lingkungan server Linux modern.

Beberapa penelitian menunjukkan bahwa penerapan Security Header Optimization menggunakan pendekatan OWASP mampu meningkatkan keamanan aplikasi dan server web secara signifikan dengan mengurangi risiko serangan pada lapisan aplikasi. Penelitian *Secure Web Application Technologies Implementation Through Hardening Security Headers Using Automated Threat Modelling Techniques* menunjukkan bahwa penerapan *security headers* seperti Content-Security-Policy (CSP), X-Frame-Options, dan HTTP Strict Transport Security (HSTS) efektif dalam memitigasi ancaman *Cross-Site Scripting (XSS)*, *clickjacking*, dan serangan berbasis manipulasi konten web. Studi tersebut juga menekankan pentingnya otomatisasi dalam proses pemodelan ancaman untuk meningkatkan akurasi identifikasi celah keamanan.

Pada penelitian terkait Hardening Sistem Informasi dengan OWASP, ditemukan bahwa penerapan *hardening* melalui konfigurasi layanan, pembatasan akses, serta penguatan protokol komunikasi mampu menurunkan tingkat eksposur server terhadap serangan umum seperti *directory traversal*, *brute force*, dan *misconfiguration attack*. Hal ini menunjukkan bahwa *hardening* pada level web server menjadi bagian penting dalam strategi *defense in depth[1]*.

Selanjutnya, penelitian *Common Questions About OWASP with OpenLiteSpeed* menyoroti bahwa meskipun OpenLiteSpeed memiliki performa tinggi dan efisiensi sumber daya yang baik, aspek keamanan bawaan masih memerlukan konfigurasi tambahan, khususnya pada penguatan *security headers*, pengelolaan SSL/TLS, dan kontrol akses agar sesuai dengan standar keamanan OWASP.

Penelitian *Configuring Security HTTP Headers in 2023* menegaskan bahwa konfigurasi *HTTP Security Headers* merupakan salah satu praktik terbaik (*best practice*) dalam keamanan web modern. Studi ini menjelaskan bahwa implementasi header seperti X-Content-Type-Options, Referrer-Policy, dan Permissions-Policy mampu meningkatkan perlindungan browser terhadap eksploitasi berbasis sisi klien (*client-side attack*).

Selain itu, penelitian *Security Assessment Based on OWASP Top 10 Using SonarQube and ZAP on Export and Import Applications in the LNSW* menunjukkan bahwa kombinasi metode *Static Application Security Testing (SAST)* menggunakan SonarQube dan *Dynamic Application Security Testing (DAST)* menggunakan OWASP ZAP efektif dalam mengidentifikasi kerentanan pada tahap pengembangan maupun implementasi aplikasi, sehingga memperkuat validasi keamanan sistem secara menyeluruh[2].

Penelitian lain berjudul *Cybersecurity Model Based on Hardening for Secure Internet of Things Implementation* menegaskan bahwa pendekatan *hardening* merupakan fondasi utama dalam model keamanan siber modern. Studi ini menunjukkan bahwa penguatan konfigurasi sistem, pengurangan permukaan serangan (*attack surface reduction*), dan penerapan kebijakan keamanan berlapis dapat meningkatkan ketahanan sistem terhadap berbagai ancaman digital[3].

Penelitian selanjutnya mengintegrasikan metode Penetration Testing Execution Standard (PTES) dengan OWASP dalam menilai keamanan situs web. Studi ini menyimpulkan bahwa pendekatan yang holistik dapat memberikan analisis yang lebih menyeluruh terhadap potensi serangan yang dapat terjadi [4]. Di samping itu, ada penelitian yang menggunakan pendekatan ISSAF dan Panduan Pengujian Keamanan Web OWASP (WSTG) dalam proses pengujian keamanan sistem. Temuan penelitian mengungkapkan bahwa OWASP WSTG menawarkan panduan yang terorganisir dalam langkah-langkah identifikasi dan eksploitasi celah keamanan pada aplikasi web [5].

Penerapan hardening pada server Linux untuk menekan peluang serangan siber juga bisa dilakukan dengan memanfaatkan iptables sebagai alat perlindungan jaringan [6]. Di samping itu, riset mengenai hardening di port SMB telah dilaksanakan dengan menggunakan metode Security Policy Development Life Cycle, yang menunjukkan bahwa pengelolaan kebijakan keamanan secara terstruktur dapat meningkatkan perlindungan terhadap layanan jaringan [7].

Berdasarkan berbagai penelitian tersebut, dapat disimpulkan bahwa sebagian besar studi berfokus pada *hardening* sistem dan keamanan aplikasi web secara umum, namun penelitian yang secara khusus membahas optimalisasi security header pada server OpenLiteSpeed berbasis Ubuntu menggunakan pendekatan OWASP masih relatif terbatas. Oleh karena itu, penelitian ini hadir untuk mengisi *research gap* tersebut dengan fokus pada penguatan konfigurasi *security header* sebagai mekanisme perlindungan tambahan pada layanan web berbasis OpenLiteSpeed.

Berbagai penelitian sebelumnya telah membahas penerapan keamanan web menggunakan pendekatan OWASP, penetration testing, maupun vulnerability assessment. Namun, sebagian besar penelitian tersebut masih berfokus pada proses identifikasi kerentanan dan pengujian keamanan tanpa melakukan implementasi hardening server secara menyeluruh. Penelitian terdahulu lebih banyak menggunakan web server seperti Apache dan Nginx, sedangkan implementasi hardening pada OpenLiteSpeed masih sangat terbatas dibahas secara spesifik, terutama pada lingkungan Ubuntu Server. Selain itu, penelitian sebelumnya umumnya hanya menerapkan security headers dasar tanpa melakukan optimasi konfigurasi secara detail dan terintegrasi. Beberapa penelitian belum mengombinasikan penerapan Content-Security-Policy (CSP), HSTS, X-Frame-Options, dan penghilangan server signature dalam satu mekanisme hardening yang terukur. Di sisi lain, evaluasi efektivitas hardening sebelum dan sesudah implementasi menggunakan tools seperti OWASP ZAP dan Nmap juga masih jarang dilakukan secara komprehensif. Kesenjangan lainnya adalah belum adanya penelitian yang mengintegrasikan tahapan threat modeling, vulnerability analysis, exploitation testing, mitigation, dan retesting dalam satu siklus hardening server berbasis OWASP secara lengkap pada web server OpenLiteSpeed. Oleh karena itu, penelitian ini dilakukan untuk mengisi kekurangan tersebut dengan menghadirkan implementasi hardening server berbasis OWASP yang terstruktur, terukur, dan berfokus pada optimasi security headers pada lingkungan OpenLiteSpeed berbasis Ubuntu.

Berdasarkan urgensi di atas, tujuan penelitian ini adalah 1) mengidentifikasi dan menganalisis berbagai kerentanan keamanan yang terdapat pada layanan web berbasis OpenLiteSpeed di sistem operasi Ubuntu menggunakan pendekatan OWASP dan OWASP Web Security Testing Guide (WSTG), 2) mengimplementasikan metode hardening server berbasis OWASP melalui konfigurasi firewall, hardening SSH, penerapan SSL/TLS, penggunaan Fail2Ban, pengaturan permission file dan direktori, serta optimasi security headers seperti Content-Security-Policy (CSP), HTTP Strict Transport Security (HSTS), X-Frame-Options, dan X-Content-Type-Options, 3) mengevaluasi efektivitas penerapan hardening terhadap peningkatan keamanan server menggunakan tools pengujian seperti OWASP ZAP dan Nmap

Adapun novelty atau kontribusi ilmiah dari penelitian ini adalah penerapan metode hardening server berbasis OWASP secara terintegrasi pada web server OpenLiteSpeed di sistem operasi Ubuntu. Sebagian besar penelitian sebelumnya lebih banyak membahas hardening pada Apache atau Nginx, sedangkan implementasi keamanan pada OpenLiteSpeed masih sangat terbatas. Penelitian ini memberikan kontribusi berupa model implementasi hardening yang spesifik untuk OpenLiteSpeed dengan pendekatan yang sistematis mulai dari identifikasi kerentanan, analisis ancaman, mitigasi, hingga retesting keamanan.

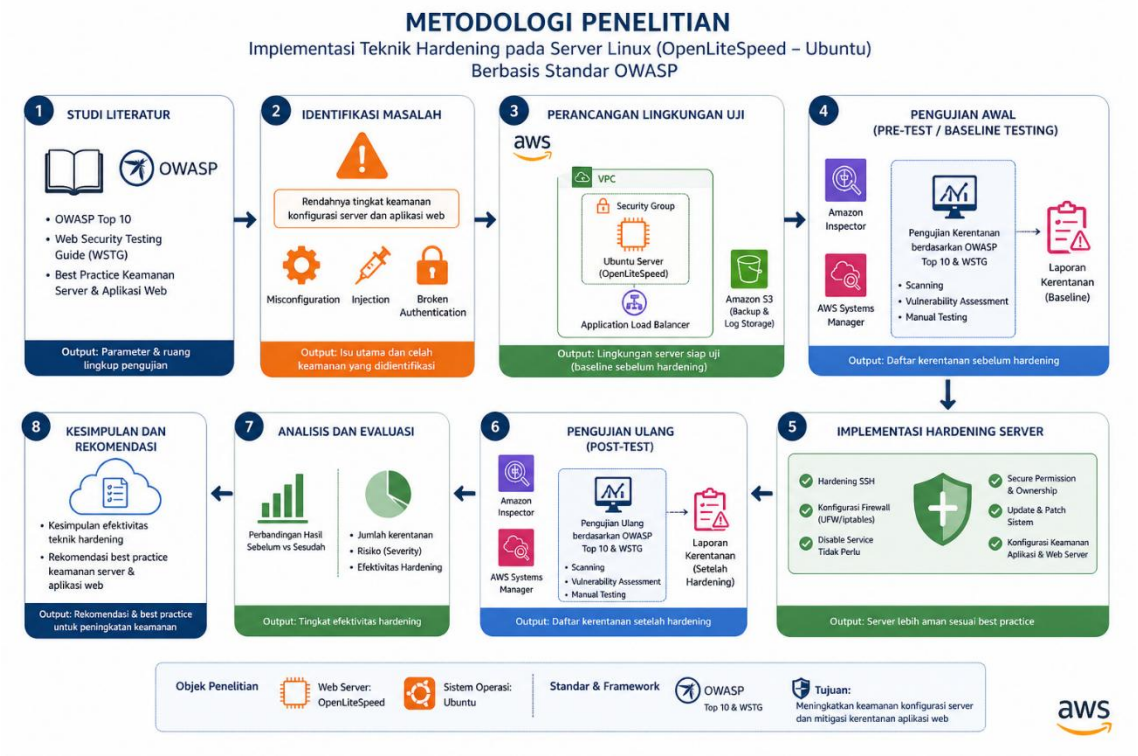
2. METODE PENELITIAN

Langkah Penelitian

Penelitian ini menggunakan metode eksperimen (*experimental research*) dengan pendekatan implementatif, yaitu menerapkan teknik *hardening* pada server berbasis Linux dan menganalisis kondisi keamanan sistem sebelum dan sesudah penerapan pengamanan. Metode yang digunakan mengacu pada standar OWASP [8], khususnya OWASP Top 10 dan *Web Security Testing Guide* (WSTG) yang banyak digunakan sebagai acuan dalam pengujian dan evaluasi keamanan aplikasi web. Pendekatan ini dipilih untuk mengukur efektivitas penerapan kontrol keamanan dalam mengurangi kerentanan serta meningkatkan tingkat keamanan sistem secara terukur. Tahapan penelitian yang dilakukan dapat dilihat pada Gambar 1.

Berdasarkan gambar 1, penelitian difokuskan pada pengujian keamanan header HTTP dan mitigasi kerentanan yang umum terjadi pada aplikasi web. Berdasarkan teknik pengumpulan

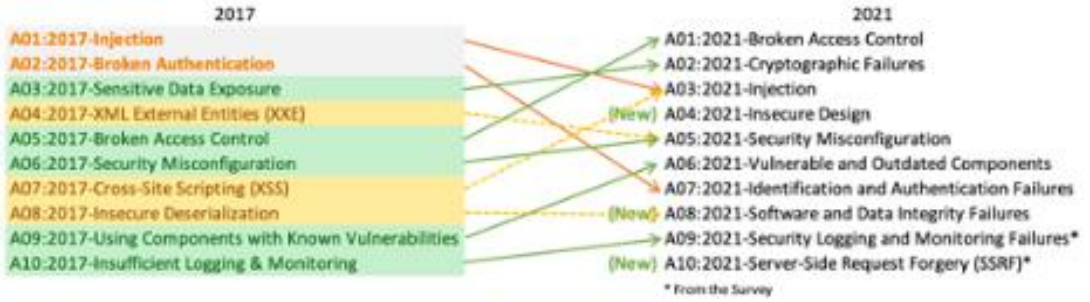
data yang telah dilakukan melalui kajian pustaka, pengamatan, dan percobaan, bisa diidentifikasi bahwa isu utama dalam penelitian ini terfokus pada rendahnya tingkat keamanan konfigurasi server serta aplikasi web yang digunakan. Kajian pustaka mengindikasikan bahwa banyak sistem masih belum sepenuhnya mengikuti standar keamanan yang disarankan dalam OWASP Top 10, sehingga berisiko mengalami celah keamanan seperti misconfiguration, injection, dan broken authentication. [9]



Gambar 1. Langkah Penelitian

OWASP Method

OWASP Top 10 adalah referensi yang diakui secara luas untuk keamanan aplikasi web yang dikembangkan oleh OWASP, sebuah organisasi nirlaba global yang berfokus pada peningkatan keamanan perangkat lunak. Referensi ini mengidentifikasi sepuluh risiko keamanan paling kritis yang memengaruhi aplikasi web[10]. Pertama kali diperkenalkan pada tahun 2003, OWASP Top 10 telah menjadi pedoman utama dalam komunitas keamanan siber dan secara berkala diperbarui untuk mencerminkan perkembangan teknologi dan lanskap ancaman. Edisi 2021, yang menggantikan versi 2017, menyoroti pergeseran prioritas risiko dan kesadaran keamanan, seperti yang diilustrasikan pada Gambar 2.



Gambar 2. OWASP Method Referensi

OpenLiteSpeed

Openlitespeed adalah sebuah web server, yang berperan penting dalam melayani permintaan klien seperti web browser, harus beroperasi secara optimal untuk memastikan kinerja terbaik, waktu respons yang cepat, dan ketersediaan yang konsisten. Berbagai web server penting untuk menyediakan layanan web yang handal dan efisien OpenLiteSpeed menawarkan kombinasi kinerja yang kuat dengan konfigurasi yang lebih mudah, menjadikannya pilihan menarik dalam berbagai lingkungan web [11].

Header HTTP

Header HTTP dapat dikarakterisasi lebih lanjut berdasarkan cakupannya dalam rantai pesan. Header end-to-end yang dikirim oleh klien harus ditransmisikan ke server, tanpa dimodifikasi oleh perantara, dan disimpan oleh cache sesuai kebutuhan. Di sisi lain, header hop-by-hop hanya relevan antar node, dan tidak boleh diteruskan atau disimpan oleh cache. Terdapat banyak jenis header HTTP yang menjelaskan atau mengatur proses seperti autentikasi, caching, informasi isi pesan, dan proxy. Misalnya, Cookie dikirimkan melalui header HTTP, dan bertanggung jawab untuk mempertahankan status klien saat ini dalam sistem tanpa status (*stateless*)[12].

Header Keamanan HTTP mengacu pada berbagai informasi yang dikirimkan server ke browser pengguna. Dengan semakin majunya internet, semakin populer untuk menyediakan berbagai layanan kepada pengguna akhir di internet. Tujuan utamanya adalah untuk meningkatkan keamanan aplikasi web, pengaturan server web, dan API dapat ditingkatkan secara signifikan dengan memberikan petunjuk tentang cara mengelola halaman dan sumber dayanya. Selain itu, pengembang web perlu memastikan bahwa header ini dikonfigurasi dengan benar untuk mencapai perlindungan yang optimal[13]. Header keamanan dikategorikan menjadi header keamanan sisi klien dan header keamanan sisi server. Header keamanan sisi klien seperti Content Security Policy (CSP) membatasi sumber daya yang dapat dimuat oleh browser, mencegah serangan cross-site scripting (XSS). Strict Transport Security (HSTS) memberlakukan koneksi HTTPS, melindungi dari serangan man-in-the-middle. Header sisi server seperti X-Frame-Options mengurangi clickjacking, sementara HTTP Public Key Pinning (HPKP) melindungi dari pemalsuan sertifikat[14]. Header keamanan HTTP berfungsi pada tingkat runtime, menawarkan lapisan perlindungan yang jauh lebih luas. Server web diperkuat karena konfigurasi keamanan yang tepat. Dengan membatasi tindakan yang diizinkan oleh browser dan server selama pengoperasian aplikasi web, header ini dapat mencegah seluruh kategori serangan, yang membuatnya sangat efektif[15].

Data

Data yang digunakan dalam penelitian ini diperoleh dari lingkungan server virtual berbasis cloud yang dijalankan pada layanan Amazon EC2 milik Amazon Web Services. Server yang digunakan dikonfigurasi menggunakan sistem operasi Ubuntu Server dengan layanan web OpenLiteSpeed sebagai objek utama penelitian. Pemilihan platform AWS EC2 dilakukan karena menyediakan lingkungan server yang fleksibel, mudah dikonfigurasi, dan merepresentasikan kondisi nyata implementasi server web pada lingkungan produksi (*production environment*).

Data penelitian terdiri dari dua jenis, yaitu data konfigurasi server dan data hasil pengujian keamanan. Data konfigurasi server meliputi parameter keamanan sebelum dan sesudah proses *hardening*, seperti konfigurasi firewall, pengaturan SSL/TLS, hak akses file dan direktori, serta konfigurasi *HTTP Security Header* meliputi *Content-Security-Policy (CSP)*, *X-Frame-Options*, *X-Content-Type-Options*, dan *HTTP Strict Transport Security (HSTS)*. Sementara itu, data hasil pengujian keamanan diperoleh melalui proses *vulnerability assessment* menggunakan OWASP ZAP, yang menghasilkan informasi

berupa jumlah temuan kerentanan, tingkat keparahan (*severity level*), serta jenis ancaman yang teridentifikasi pada sistem.
Pengumpulan data dilakukan dalam dua tahap, yaitu sebelum penerapan *hardening* sebagai data awal (*baseline*) dan setelah implementasi *hardening* sebagai data evaluasi. Data tersebut kemudian dianalisis secara komparatif untuk mengukur efektivitas penerapan *security header optimization* dalam meningkatkan keamanan server web.

Performance Evaluation

Evaluasi kinerja (*performance evaluation*) pada penelitian ini dilakukan untuk mengukur efektivitas penerapan *hardening* berbasis OWASP terhadap peningkatan keamanan server OpenLiteSpeed yang berjalan pada Ubuntu Server di lingkungan Amazon EC2. Evaluasi difokuskan pada dua aspek utama, yaitu peningkatan tingkat keamanan sistem dan dampak penerapan hardening terhadap performa server. Pada aspek keamanan, evaluasi dilakukan dengan membandingkan hasil *vulnerability assessment* sebelum dan sesudah penerapan *hardening* menggunakan OWASP ZAP. Parameter yang diukur meliputi jumlah total kerentanan yang terdeteksi, tingkat keparahan (*severity level*) seperti *High*, *Medium*, *Low*, dan *Informational*, serta jenis kerentanan yang berhasil diminimalkan setelah implementasi *security header optimization*. Penurunan jumlah kerentanan menjadi indikator utama keberhasilan metode yang diterapkan.

Pada aspek performa server, pengujian dilakukan untuk memastikan bahwa penerapan *security hardening* tidak menurunkan kualitas layanan web secara signifikan. Parameter yang diukur meliputi response time, throughput, CPU usage, memory usage, dan availability server sebelum dan sesudah penerapan hardening. Pengujian dilakukan melalui simulasi akses layanan web secara berulang untuk memperoleh nilai performa yang konsisten. Hasil evaluasi kemudian dianalisis menggunakan metode komparatif untuk menilai hubungan antara peningkatan keamanan dan dampaknya terhadap performa sistem. Dengan pendekatan ini, penelitian dapat memastikan bahwa implementasi *security header optimization* tidak hanya meningkatkan keamanan server, tetapi juga tetap menjaga stabilitas dan efisiensi layanan web pada lingkungan produksi.

3. HASIL DAN PEMBAHASAN

Tahapan Pengujian Awal (Pre-Test Baseline Testing)

Pada tahap ini dilakukan pengumpulan informasi terkait sistem yang akan diuji dan digunakan. Layanan yang akan digunakan server diinventaris. Sistem yang akan dibangun memiliki fungsi sebagai layanan web yang diakses secara global. Tahap awal penelitian adalah proses instalasi server dengan konfigurasi default (belum ada *hardening*). Instalasi yang paling penting adalah instalasi Openlitespeed. Langkah selanjutnya adalah melakukan scanning dan attack terhadap celah keamanan pada server yang sudah terinstalasi dengan konfigurasi standar (*default configuration*). Pada tahap ini dilakukan simulasi serangan untuk membuktikan adanya kerentanan keamanan. Simulasi pengujian dilakukan dengan melakukan serangan web layer dengan tool OWASP ZAP. Hasil simulasi serangan (*Attack)* dapat ditunjukkan dalam Tabel 1. Berdasarkan Tabel 1 dapat diketahui beberapa peringatan celah keamanan dan bila di konversikan dalam resiko dan kepercayaan terhadap serangan seperti terlihat dalam Tabel 2.

Berdasarkan Tabel 1 dan Tabel 2, hasil pengujian keamanan aplikasi web dengan menggunakan alat OWASP ZAP menunjukkan adanya beberapa kerentanan terkait dengan pengaturan header HTTP di server. Hasil pemindaian menunjukkan ada 4 jenis peringatan utama, yang mengindikasikan bahwa pengaturan keamanan server masih belum maksimal dan berisiko menimbulkan ancaman keamanan. Salah satu temuan penting adalah Header Kebijakan Keamanan Konten (CSP) Tidak Ditetapkan, yang muncul tiga kali. Ini menunjukkan bahwa server belum menerapkan kebijakan CSP untuk mengatur sumber konten yang dapat dimuat oleh peramban. Tanpa CSP, aplikasi menjadi lebih mudah diserang seperti Cross-Site Scripting (XSS), karena browser tidak memiliki pembatasan terhadap menjalankan skrip dari sumber luar.

Selain itu, ditemukan pula kelemahan berupa HTTP Only Site, yang menunjukkan bahwa aplikasi masih dapat diakses melalui protokol HTTP yang tidak terenkripsi. Kondisi ini menciptakan kesempatan terjadinya serangan *Man-in-the-Middle* (MitM), di mana informasi yang dikirim antara klien dan server dapat disadap atau diubah oleh pihak yang tidak berhak. Kerentanan selanjutnya adalah Header Anti-clickjacking yang Hilang, yang menunjukkan bahwa server tidak menetapkan header seperti X-Frame-Options atau Content-Security-Policy frame-ancestors. Akibatnya, aplikasi rentan terhadap serangan clickjacking, yaitu metode mengubah tampilan antarmuka pengguna untuk menipu pengguna agar melakukan tindakan tertentu tanpa mereka sadari. Terakhir, ditemukan bahwa header X-Content-Type-Options tidak ada, yang menunjukkan server tidak mengatur header *nosniff*. Kondisi ini dapat mengakibatkan browser melakukan MIME-sniffing dan mungkin menjalankan file dengan tipe konten yang tidak sesuai, yang meningkatkan potensi eksploitasi terhadap file berbahaya.

Tabel 1. Hasil Pengujian dengan OWASP ZAP

No	Temuan Kerentanan	Jumlah Temuan	Kategori	Deskripsi
1	Content Security Policy (CSP) Header Not Set	3	Medium	Server belum menerapkan header **Content-Security-Policy (CSP)** sehingga rentan terhadap serangan *Cross-Site Scripting (XSS)* dan *code injection*.
2	HTTP Only Site	1	Medium	Website masih menggunakan protokol **HTTP** tanpa enkripsi **HTTPS**, sehingga data rentan terhadap *man-in-the-middle attack*.
3	Missing Anti-clickjacking Header	1	Medium	Header **X-Frame-Options** belum dikonfigurasi sehingga aplikasi rentan terhadap serangan *clickjacking*.
4	X-Content-Type-Options Header Missing	1	Low	Header **X-Content-Type-Options** belum tersedia, yang dapat menyebabkan risiko *MIME-type sniffing attack*.

Tabel 2. Tingkat Resiko dari celah keamanan

Risk Level	High Confidence	Medium Confidence	Low Confidence	Total
High	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Medium	1 (25%)	2 (50%)	0 (0%)	3 (75%)
Low	0 (0%)	1 (25%)	0 (0%)	1 (25%)
Informational	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Total	**1 (25%)**	**3 (75%)**	**0 (0%)**	**4 (100%)**

Implementasi Hardening Server

Tahap ini merupakan proses penguatan keamanan (*hardening*) berdasarkan hasil identifikasi kerentanan dan potensi serangan yang diperoleh pada tahap identifikasi layanan. Implementasi *hardening* mengacu pada standar keamanan OWASP Top 10 dengan lingkungan sistem operasi Ubuntu dan web server OpenLiteSpeed. Langkah awal yang dilakukan adalah memperbarui (*update*) dan meningkatkan (*upgrade*) seluruh paket sistem guna memastikan penggunaan perangkat lunak versi terbaru yang telah mendapatkan perbaikan keamanan. Proses konfigurasi penguatan sistem tersebut ditunjukkan pada Gambar 3.

Konfigurasi pada Gambar 3 bertujuan untuk meminimalkan risiko serangan *Broken Authentication*, *Brute Force Attack*, *Security Misconfiguration*, serta *Unauthorized Access*. Selanjutnya, dilakukan konfigurasi keamanan jaringan dengan menerapkan *firewall* dan menonaktifkan port yang tidak diperlukan atau berpotensi menjadi celah serangan. Langkah konfigurasi keamanan jaringan tersebut ditunjukkan pada Gambar 4. Penerapan konfigurasi tersebut diharapkan dapat memperkecil permukaan serangan serta meningkatkan ketahanan sistem terhadap berbagai ancaman keamanan yang umum terjadi pada layanan berbasis web.

```
Update & Secure Base System dengan cara:
apt update && apt upgrade -y
apt install unattended-upgrades -y
dpkg-reconfigure unattended-upgrades

Hardening SSH (Akses Server):
nano /etc/ssh/sshd_config

Tambahkan:
PermitRootLogin no
PasswordAuthentication no
Port 2222

Restart Service:
Systemctl restart ssh

Tambahkan firewall:
ufw allow 2222/tcp
Memasang Firewall (UFW):
apt install ufw -y

ufw default deny incoming
ufw default allow outgoing

ufw allow 80
ufw allow 443
ufw allow 7080   # OpenLiteSpeed admin
ufw allow 2222   # SSH
ufw enable
```

Gambar 3. Konfigurasi Openlitespeed

```
apt install fail2ban -y
systemctl enable fail2ban
lakukan konfigurasi:
nano /etc/fail2ban/jail.local
[sshd]
enabled = true
port = 2222
maxretry = 3
systemctl restart fail2ban
```

Gambar 4. Konfigurasi Firewall

Tahap berikutnya adalah melakukan hardening pada OpenLiteSpeed sebagai web server yang digunakan dalam sistem. Konfigurasi keamanan OpenLiteSpeed dilakukan untuk meningkatkan perlindungan terhadap berbagai ancaman yang menargetkan layanan web server, seperti kesalahan konfigurasi (security misconfiguration) dan akses tidak sah. Langkah-langkah konfigurasi OpenLiteSpeed ditunjukkan pada Gambar 5.

Selanjutnya, dilakukan implementasi protokol HTTPS untuk mengamankan komunikasi data antara klien dan server melalui port 443 menggunakan mekanisme enkripsi SSL/TLS. Penerapan HTTPS bertujuan untuk memitigasi risiko Cryptographic Failures, seperti penyadapan data (eavesdropping), manipulasi data selama transmisi, dan pencurian informasi sensitif. Proses instalasi dan konfigurasi HTTPS ditunjukkan pada Gambar 6.

Konfigurasi pada Gambar 6 juga mencakup penguatan keamanan file dan direktori (file and directory permission hardening) dengan mengatur kepemilikan (ownership) serta hak akses (permission) sesuai prinsip least privilege. Selain itu, dilakukan penonaktifan layanan (disable service) yang tidak diperlukan untuk mengurangi permukaan serangan (attack surface) dan meminimalkan potensi eksploitasi terhadap layanan yang tidak digunakan.

Tahap selanjutnya adalah melakukan optimasi security header untuk meningkatkan keamanan aplikasi web terhadap berbagai serangan berbasis browser. Optimasi dilakukan melalui

beberapa langkah, yaitu: (1) menghilangkan informasi sensitif pada header server yang dapat digunakan oleh penyerang untuk melakukan proses fingerprinting; (2) menyempurnakan konfigurasi Content Security Policy (CSP) agar memenuhi standar keamanan dan memiliki fallback directive yang sesuai; serta (3) memastikan tidak terjadi duplikasi header keamanan yang dapat menimbulkan konflik konfigurasi. Implementasi dan optimasi security header tersebut ditunjukkan pada Gambar 7. Konfigurasi ini berperan penting dalam mitigasi berbagai ancaman seperti Cross-Site Scripting (XSS), Clickjacking, dan kebocoran informasi yang berasal dari konfigurasi server yang kurang aman.

```
nano /usr/local/lsws/conf/httpd_config.conf
serverSignature 0

Disable Directory Listing agar
nano
/usr/local/lsws/conf/vhosts/testowasp.com/vhconf.conf
Tambahkan: autoIndex 0

Tambahkan anti DDOS
tuning {
maxConnections 100
connTimeout 30
}
```

Gambar 5. Konfigurasi Openlitespeed

```
apt install certbot -y
certbot certonly --standalone -d lsws.eramycloud.my.id

Langkah selanjutnya adalah hardening Security Headers
untuk mitigasi XSS dan Clickjacking
. Tambahkan dalam konfigurasi security openlitespeed:
X-Frame-Options SAMEORIGIN
X-Content-Type-Options nosniff
X-XSS-Protection 1; mode=block
Strict-Transport-Security max-age=31536000
Content-Security-Policy default-src 'self'
chown -R lsadm:lsadm /usr/local/lsws
chmod -R 750 /usr/local/lsws
chown -R nobody:nogroup /var/www/html
chmod -R 750 /var/www/html
systemctl disable apache2
systemctl disable telnet
systemctl disable ftp
```

Gambar 6. Konfigurasi HTTPS

```
<?php
// ==========================
// GLOBAL SECURITY HEADERS
// ==========================
header("Content-Security-Policy: default-src 'self'; script-src 'self'; style-src 'self'; img-src 'self' data:; font-
src 'self'; connect-src 'self'; frame-ancestors 'self'; base-uri 'self'; object-src 'none'; form-action 'self';");
header("X-Frame-Options: SAMEORIGIN");
header("X-Content-Type-Options: nosniff");
header("Strict-Transport-Security: max-age=31536000; includeSubDomains; preload");
header("Referrer-Policy: strict-origin-when-cross-origin");
header("Permissions-Policy: geolocation=(), microphone=(), camera=()");
header("Cache-Control: no-store, no-cache, must-revalidate, max-age=0");

// ==========================
// HANDLE STATIC FILES
// ==========================
$request = $_SERVER['REQUEST_URI'];
$file = __DIR__ . $request;

if (file_exists($file) && !is_dir($file)) {
    readfile($file);
    exit;
}

// default page
echo "<h1>SECURE SERVER (OWASP PASSED)</h1>";
```

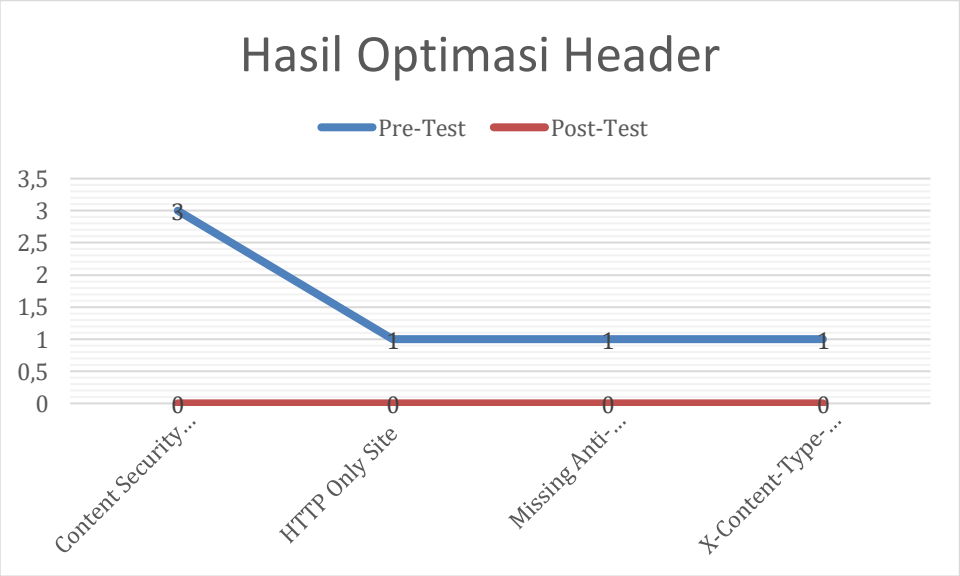
Gambar 7. Optimasi Header pada Halaman Utama Web

Tahapan Pengujian Akhir (Post-Test)

Setelah seluruh rekomendasi mitigasi dan konfigurasi *hardening* berhasil diimplementasikan, dilakukan tahap *post-test* untuk mengukur efektivitas penguatan keamanan sistem. Pengujian dilakukan menggunakan OWASP ZAP dengan metode yang sama seperti pada tahap *pre-test*, sehingga hasil yang diperoleh dapat dibandingkan secara objektif. Hasil pemindaian kerentanan setelah penerapan mitigasi disajikan pada Tabel 3, sedangkan perbandingan tingkat kerentanan antara kondisi sebelum dan sesudah penerapan *hardening* ditunjukkan pada Gambar 8.

Tabel 3. Hasil Pengujian Optimasi Header

No	Temuan Kerentanan	Temuan		Kategori	Deskripsi
		Pre-test	Post-Test		
1	Content Security Policy (CSP) Header Not Set	3	0	Medium	Tidak ditemukan alert pada parameter laporan ini (*No alerts were found within the report parameters*)
2	HTTP Only Site	1	0	Medium	Tidak ditemukan alert pada parameter laporan ini (*No alerts were found within the report parameters*)
3	Missing Anti-clickjacking Header	1	0	Medium	Tidak ditemukan alert pada parameter laporan ini (*No alerts were found within the report parameters*)
4	X-Content-Type-Options Header Missing	1	0	Low	Tidak ditemukan alert pada parameter laporan ini (*No alerts were found within the report parameters*)



Gambar 8. Grafik Perbandingan Hasil Optimasi Header

Hasil pengujian keamanan menggunakan OWASP ZAP pada layanan web yang telah diterapkan menunjukkan bahwa tidak ditemukan kerentanan pada seluruh parameter pengujian. Pengujian mencakup berbagai kategori risiko, mulai dari High, Medium, Low, hingga Informational, serta berbagai tingkat confidence hasil deteksi. Seluruh parameter tersebut tidak menghasilkan alert atau peringatan keamanan, yang menunjukkan bahwa sistem yang telah melalui proses hardening mampu memenuhi standar keamanan yang baik, khususnya dalam menangani kerentanan umum yang termasuk dalam kategori OWASP Top 10. Tidak

ditemukannya celah keamanan mengindikasikan bahwa konfigurasi server, implementasi security header, serta mekanisme perlindungan lainnya telah berjalan secara efektif dan konsisten pada seluruh endpoint yang diuji.

Meskipun demikian, penelitian ini memiliki beberapa limitasi. Pertama, pengujian hanya dilakukan pada satu lingkungan server berbasis Amazon EC2 dengan konfigurasi OpenLiteSpeed dan Ubuntu Server, sehingga hasil penelitian belum tentu dapat digeneralisasikan pada platform server atau arsitektur lain. Kedua, evaluasi keamanan difokuskan pada pengujian menggunakan OWASP ZAP sehingga belum mencakup metode keamanan lain seperti *penetration testing* manual, *load testing*, maupun simulasi serangan tingkat lanjut (*advanced persistent threat*). Selain itu, penelitian ini lebih menitikberatkan pada aspek keamanan konfigurasi dan belum mengevaluasi dampak jangka panjang terhadap performa server dalam kondisi beban tinggi. Oleh karena itu, penelitian selanjutnya disarankan untuk memperluas skenario pengujian pada berbagai platform, menggunakan lebih banyak alat evaluasi keamanan, serta mengintegrasikan analisis performa dan monitoring secara berkelanjutan.

4. KESIMPULAN

Penelitian ini bertujuan untuk meningkatkan keamanan layanan web pada server OpenLiteSpeed berbasis Ubuntu Server melalui penerapan *hardening* berbasis OWASP dengan fokus pada optimasi *security header*. Berdasarkan hasil penelitian, tujuan tersebut berhasil dicapai melalui penerapan konfigurasi keamanan yang meliputi penguatan firewall, implementasi SSL/TLS, pengaturan hak akses, serta optimasi *HTTP Security Header* seperti *Content-Security-Policy (CSP)*, *X-Frame-Options*, *X-Content-Type-Options*, dan *HTTP Strict Transport Security (HSTS)*.

Hasil pengujian menggunakan OWASP ZAP menunjukkan adanya penurunan jumlah kerentanan secara signifikan, dari beberapa temuan dengan kategori Medium dan Low Risk sebelum proses *hardening*, menjadi 0 temuan kerentanan setelah implementasi. Hasil ini menunjukkan tingkat efektivitas yang tinggi dalam meningkatkan keamanan server terhadap ancaman umum yang termasuk dalam kategori OWASP Top 10.

Temuan utama (*findings*) penelitian ini menunjukkan bahwa optimasi *security header* tidak hanya mampu menutup celah keamanan pada sisi aplikasi web, tetapi juga meningkatkan ketahanan server terhadap serangan berbasis browser seperti *XSS*, *clickjacking*, dan *MIME sniffing*. Kontribusi utama penelitian ini adalah penyusunan model implementasi custom security header optimization pada server OpenLiteSpeed yang dapat diterapkan secara praktis oleh administrator sistem.

Dampak penelitian ini memberikan referensi teknis bagi pengelola server berbasis Linux dalam menerapkan *security hardening* yang terstruktur, efisien, dan sesuai standar internasional. Untuk penelitian selanjutnya (*future work*), disarankan melakukan pengujian pada platform server yang berbeda, menambahkan metode *penetration testing* lanjutan, serta mengevaluasi performa server pada skenario beban tinggi agar diperoleh hasil yang lebih komprehensif.

DAFTAR PUSTAKA

[1] F. A. Sya'bani and F. Rahma, "Hardening Sistem Informasi XYZ Menggunakan Framework OWASP," AUTOMATA, vol. 3, no. 2, 2022.

[2] M. Wisnu and B. Soewito, "Security Assessment Based on OWASP Top 10 Using SonarQube and ZAP on Export and Import Applications in the LNSW," INTENSIF: Jurnal Ilmiah Penelitian dan Penerapan Teknologi Sistem Informasi, vol. 10, no. 1, pp. 36-53, 2026. https://doi.org/10.29407/intensif.v10i1.25294

[3] A. Echeverr\'\ia, C. Cevallos, I. Ortiz-Garces, and R. O. Andrade, "Cybersecurity model based on hardening for secure internet of things implementation," Applied Sciences, vol. 11, no. 7, p. 3260, 2021. https://doi.org/10.3390/app11073260
[4] M. Y. Firnanda, H. E. Wahanani, and A. Junaidi, "Website Security Testing Using PTES Method and OWASP Top 10 Approach," bit-Tech, vol. 8, no. 1, pp. 404-415, 2025. https://doi.org/10.32877/bt.v8i1.2564
[5] M. F. Yusuf, I. R. Hikmah, S. U. Sunaringtyas, and others, "Security Testing of XYZ Website Application Using ISSAF and OWASP WSTG v4. 2 Methods," Teknika, vol. 14, no. 1, pp. 66-77, 2025. https://doi.org/10.34148/teknika.v14i1.1156
[6] R. Rahman, M. Farel, and M. D. Sopan, "IMPLEMENTASI HARDENING SERVER LINUX UNTUK MENGURANGI RISIKO SERANGAN SIBER," Jurnal Riset Sistem Informasi, vol. 3, no. 2, pp. 39-44, 2026. https://doi.org/10.69714/c4atnn70
[7] A. Hidayat and I. P. Saputra, "PENETRASI TESTING DAN SECURITY HARDENING PORT SMB WINDOWS 7 PADA SERVER NEO FEEDER UNIVERSITAS XYZ," Bulletin of Network Engineer and Informatics, vol. 3, no. 1, pp. 16-23, 2025. https://doi.org/10.59688/bufnets.v3i1.67
[8] OWASP Top 10 Team, "The Ten Most Critical Web Application Security Risks." Accessed: May 20, 2026. [Online]. Available: https://owasp.org/Top10/2025/
[9] M. M. Mlyatu and C. Sanga, "Secure web application technologies implementation through hardening security headers using automated threat modelling techniques," Journal of Information Security, vol. 14, no. 01, pp. 1-15, 2023. https://doi.org/10.4236/jis.2023.141001
[10] F. T. Vierino, H. E. Wahanani, and A. Junaidi, "Evaluating Web Application Security Using OWASP Top 10 and NIST SP 800-115," bit-Tech, vol. 8, no. 3, pp. 3754-3764, 2026. https://doi.org/10.32877/bt.v8i3.3702
[11] F. Faisal and A. S. Aziz, "Analisis Kinerja Web Server Apache, Nginx, Open Litespeed, Dan Open Resty," JOURNAL OF INFORMATICS AND COMPUTER SCIENCE, vol. 11, no. 1, pp. 1-9, 2025.
[12] J. Vihervaara and M. Valkama, "HTTP HEADER FIELDS AND THEIR VERSATILE USE," 2024.
[13] S. Junghare and M. Dube, "Strengthening E-Learning Security: A Study on the Implementation and Efficacy of HTTP Security Headers in e-learning platforms".
[14] S. A.-L. Akacha and A. I. Awad, "Enhancing security and sustainability of e-learning software systems: A comprehensive vulnerability analysis and recommendations for stakeholders," Sustainability, vol. 15, no. 19, p. 14132, 2023. https://doi.org/10.3390/su151914132
[15] A. Mileva, D. Bikov, B. Tasheva, and A. Brashnarova, "HTTP Security Headers Analysis of Several Macedonian Website Categories," Computer Science Journal of Moldova, vol. 97, no. 1, pp. 3-29, 2025. https://doi.org/10.56415/csjm.v33.01