Analisis Kerentanan Keamanan pada Management Information System USAID SEA-PROJECT Menggunakan Metode OWASP

Yuswandi

Universitas Gunadarma
Jl. Margonda Raya, No. 100, 16424, Depok, Indonesia
yuswandi@outlook.com

Abstrak

Kerentanan keamanan pada sebuah aplikasi berbasis web dapat diuji dengan cara melakukan penetrasi terhadap aplikasi tersebut. Teknik pengujian yang diterapkan dalam penelitian ini mengambil petunjuk dari panduan yang dibuat oleh organisasi bernama *Open Web Application Security* (OWASP), yaitu sebuah organisasi nirlaba dengan misi untuk membuat keamanan perangkat lunak dapat dilihat oleh Individual dan Organisasi untuk membantu mereka mengambil keputusan berdasarkan informasi tentang risiko keamanan perangkat lunak mereka. Objek yang menjadi target dalam pengujian ini adalah aplikasi web *Management Information System* (MIS) yang dibuat oleh organisasi USAID SEA-PROJECT (2016-2021) dengan alamat www.sea-mis.org. Penelitian ini bertujuan untuk menemukan celah kerentanan keamanan yang ada dalam aplikasi web tersebut. Pengujian yang dijalankan dalam penelitian ini bersifat otomatis dengan memakai perangkat lunak dari OWASP yang bernama *Zed Attack Proxy* (ZAP) untuk analisis dinamis dan perangkat lunak RIPS untuk analisis statis. Perangkat lunak tersebut melakukan pemindaian terhadap elemen-elemen yang ada dalam aplikasi web tersebut dan kemudian mengumpulkan informasi tentang celah kerentanan keamanan atau akses. Kerentanan yang ditemukan dibagi menjadi empat tingkatan, mulai dari yang paling parah yaitu *High*, *Medium*, *Low*, dan *Informational*. Hasil penelitian ini adalah perbaikan untuk menutup celah kerentanan keamanan yang berhasil ditemukan tersebut terutama untuk tingkat *High* dan *Medium* guna peningkatan keamanan untuk kemudian diterapkan oleh para pengelola aplikasi web tersebut.

Kata Kunci: kerentanan, keamanan, aplikasi, web, OWASP

Pendahuluan

Proyek USAID *Sustainable Ecosystem Advanced* (USAID SEA-PROJECT) merupakan proyek lima tahun (2016-2021) yang mendukung Pemerintah Indonesia dalam menguatkan tata kelola sumber daya perikanan dan kelautan, serta konservasi keanekaragaman hayati. Proyek yang diimplementasikan oleh Tetra Tech dan konsorsium mitra ini bekerja pada tingkat nasional, provinsial, serta lokal di Papua Barat, Maluku, dan Maluku Utara yang termasuk di dalam Wilayah Pengelolaan Perikanan (WPP) 715 Indonesia [1].

Dalam rangka menyatukan, mengelola, dan memastikan bahwa setiap bidang ataupun divisi dalam proyek tersebut berjalan baik dan bersinergi dalam proses pencapaian tujuan tadi, maka sudah seharusnya dibangun sebuah sistem informasi manajemen yang mampu memberikan kemudahan, bisa diakses dari mana saja, menghasilkan *output* atau informasi yang cepat, tepat waktu, dan akurat dalam proses pengambilan keputusan. Maka oleh karena itu dibangunlah sebuah aplikasi web sistem informasi manajemen yang saat ini ditempatkan dan dijalankan pada sebuah server lokal dan domain yang dinamakan sea-mis.org.

Namun dalam perjalanannya, seiring dengan segala kelengkapan dan kemudahan yang

DOI : http://dx.doi.org/10.32409/jikstik.19.4.355

didapatkan dari aplikasi sistem informasi manajemen tersebut, timbul pula aspek negatifnya, yaitu kerentanan keamanan terhadap data dan informasi proyek yang dimasukkan dan tersimpan dalam sistem informasi manajemen tersebut. Akses terhadap aplikasi www.sea-mis.org yang bisa dilakukan kapan saja dan dari mana saja, selain memudahkan tetapi juga membuka peluang untuk terjadinya serangan atau penyusupan ke dalam sistem dari pihak-pihak yang tidak bertanggung jawab hingga bisa berakibat terjadinya kebocoran data atau informasi maupun kerusakan pada sistem secara keseluruhan.

Dengan berkembangnya teknologi web, popularitas aplikasi berbasis web telah sangat berkembang. Hari ini, aplikasi berbasis *web* digunakan dalam lingkungan keamanan yang kritis seperti kementerian, departemen dan agensi. Demikian pula halnya dengan pengguna aplikasi web kerentanan meningkat dalam organisasi kita, ini akan tanpa keraguan mengekspos lebih banyak pengguna aplikasi *web* terhadap serangan berbahaya [2].

Secara umum bentuk serangan pada aplikasi web yang sering terjadi seperti misalnya *Malware*, *Penetration Testing*, $SQL - Injection$, dan lain-lain. Penyebab semua kerentanan yang teridentifikasi dalam aplikasi web, masalahnya disebabkan oleh *input* yang tidak diperiksa yang diakui sebagai yang paling umum [3].

Namun dalam konteks aplikasi web www.sea-mis.org ini permasalahannya bisa lebih daripada itu, yaitu pencurian dan penggunaan identitas oleh pihak yang tidak berhak hingga berakibat pada bocornya data dan informasi yang tersimpan dalam aplikasi *web* www.sea-mis.org tersebut. Berdasarkan UU ITE Pasal 15, penyelenggara sistem elektronik harus mengatur sistem elektronik secara andal dan aman dan bertanggung jawab atas pengoperasian sistem elektronik yang sesuai. Karenanya, identifikasi harus dilakukan untuk melihat kelemahan keamanan dalam aplikasi, komputer, dan jaringan. Evaluasi keamanan jaringan memungkinkan waktu untuk memperbaiki sistem sebelum menjadi buruk [4].

Oleh karena itu organisasi perlu melakukan penilaian (*assessment*) pada aplikasi berbasis web agar organisasi mampu mendeteksi kerentanan dan memahami risiko yang dihadapi [5]. Peningkatan keamanan di situs web dapat dilakukan dengan menguji kerentanan situs web, salah satunya adalah metode pengujian penetrasi [6]. Selain itu juga serangan berbahaya di aplikasi web adalah SQL *Injection*, semua aplikasi web rentan terhadap serangan dan yang paling banyak digunakan untuk menyerang adalah SQL *Injection* [7].

Maka sebagai antisipasi terhadap segala permasalahan tersebut di atas, maka penelitian ini akan mencoba menelaah lebih dalam tentang pokok aspek keamanan dari sistem manajemen informasi berbasis web www.sea-mis.org tersebut dengan menekankan titik fokus pada penggunaan metode OWASP sebagai dasar pengujian dan penilaiannya. Adapun OWASP atau singkatan dari *Open Web Application Security Project* yaitu sebuah organisasi nirlaba dengan misi untuk membuat keamanan perangkat lunak dapat dilihat oleh Individual dan Organisasi untuk membantu mereka mengambil keputusan berdasarkan informasi tentang risiko keamanan perangkat lunak mereka. OWASP TOP 10 adalah proyek keamanan yang disponsori oleh OWASP. Proyek tersebut mempublikasi sebuah daftar dari apa yang dianggap sebagai 10 teratas ancaman keamanan aplikasi web di seluruh dunia. Daftar tersebut menjelaskan kerentanan dengan contoh yang relevan dan cara menghindarinya [8].

Penelitian ini dilakukan guna menemukan celah kerentanan keamanan yang ada dalam aplikasi web yang menjadi target tersebut untuk kemudian diberikan solusi untuk menanggulanginya dan dapat diterapkan sebagai perbaikan terhadap masalah tersebut. Beberapa kegunaan yang bisa didapatkan sebagai hasil dari penelitian ini diantaranya yang pertama adalah sebagai masukan dan solusi bagi pengelola aplikasi web www.sea-mis.org untuk meningkatkan keamanan pada sistemnya.

Kedua, sebagai bantuan atau alternatif bagi organisasi lainnya yang ingin melakukan pengujian keamanan aplikasi berbasis web serupa yang sudah berjalan. Ketiga, dapat dipakai sebagai tolok ukur proses pengujian ketahanan atau tingkat keamanan aplikasi berbasis web. Kemudian yang keempat yaitu dapat membantu para pengembang aplikasi berbasis web menemukan celah kerentanan keamanan dan menerapkan solusi yang diberikan pada aplikasi yang dibuat tersebut.

Tinjauan Pustaka

Sistem Informasi Manajemen

Sistem Informasi Manajemen (SIM) menurut para ahli pada dasarnya berkaitan dengan proses pengumpulan, pemrosesan, penyimpanan, dan transmisi informasi yang relevan untuk mendukung operasi manajemen di organisasi mana pun [8]. Jenis sistem informasi komputer organisasi, yang mengambil informasi internal dari sistem pemrosesan operasi dan merangkumnya menjadi bentuk yang bermakna dan berguna sebagai laporan manajemen untuk digunakan dalam melakukan tugas manajemen [9].

Aplikasi Web

Web (*World Wide W*eb, atau WWW) adalah ruang informasi di mana item-item yang menarik, disebut sebagai sumber daya, diidentifikasi oleh pengidentifikasi global yang disebut *Uniform Resource Identifiers* (URI). Dengan demikian, web adalah ruang informasi. Tiga spesifikasi pertama untuk teknologi web mendefinisikan URL, HTTP, dan HTML [10]. Sementara itu, aplikasi web adalah sebuah perangkat lunak yang berjalan pada sebuah server web, ia menjadi semakin popular dan bagian penting dalam kehidupan sehari-hari daripada sebelumnya [3].

Keamanan Informasi

Secara umum, keamanan adalah "suatu kualitas atau keadaan yang terjamin – bebas dari bahaya." Dengan kata lain, perlindungan terhadap musuh – dari mereka yang akan berbuat jahat, dengan sengaja atau tidak – adalah tujuannya [11]. Sedangkan Informasi sebagaimana yang telah disebutkan di atas yaitu data yang telah dikonversi menjadi konteks yang bermakna dan berguna bagi pengguna akhir tertentu [12].

Ancaman dan Kerentanan

ISO/IEC 27000: 2012 mendefinisikan ancaman sebagai "penyebab potensial dari insiden yang tidak diinginkan, yang dapat mengakibatkan kerusakan pada sistem atau organisasi." Ancaman adalah segala tindakan atau aktor yang dapat menyebabkan konsekuensi yang tidak diinginkan, seperti pelanggaran kerahasiaan atau kehilangan layanan. Sedangkan Kerentanan didefinisikan dalam ISO/IEC 27000: 2012 sebagai kelemahan suatu aset atau kontrol yang dapat dieksploitasi oleh satu atau lebih ancaman. Definisi kerentanan lainnya yaitu merupakan kesalahan perangkat lunak yang dieksploitasi sebagai akibat dari serangan keamanan. [13].

Penilaian Kerentanan

Penilaian kerentanan adalah sub set dari audit dan difokuskan pada menemukan kelemahan atau kerentanan dalam aplikasi web [14]. Definisi lainnya bahwa penilaian kerentanan adalah proses pemindaian sistem atau perangkat lunak atau jaringan untuk mengetahui kelemahan dan celah di dalamnya. Celah ini dapat memberikan pintu belakang bagi penyerang untuk menyerang korban. Suatu sistem mungkin memiliki kerentanan kontrol akses, kerentanan kondisi batas, kerentanan validasi input, kerentanan otentikasi (*authentication*), kerentanan kelemahan konfigurasi, dan kerentanan penanganan pengecualian [15].

Pengujian Penetrasi

Pengujian penetrasi adalah langkah berikutnya setelah penilaian kerentanan. Pengujian penetrasi adalah untuk mencoba mengeksploitasi sistem dengan cara yang sah untuk mengetahui kemungkinan eksploitasi dalam sistem. Dalam pengujian penetrasi, tester memiliki wewenang untuk melakukan pengujian penetrasi dan dia dengan sungguh-sungguh mengeksploitasi sistem dan mencari tahu kemungkinan eksploitasi [15].

Teknik Penilaian Kerentanan

1. **AnalisisStatis** (***Static Analysis***)

   Dalam teknik ini Tester tidak menjalankan uji kasus atau eksploitasi. Tester menganalisis struktur kode dan konten sistem. Dengan teknik ini kita bisa mengetahui tentang semua jenis kerentanan. Dalam teknik ini kita tidak mengeksploitasi sistem, jadi tidak akan ada efek buruk pengujian ini pada sistem. Salah satu kelemahan besar dari teknik ini adalah sangat lambat dan membutuhkan banyak waktu untuk melakukan.

2. **UjicobaManual** (***Manual Testing***)

Dalam teknik ini, tester tidak memerlukan alat atau perangkat lunak apa pun untuk mengetahui kerentanan. Dalam tester ini gunakan pengetahuan dan pengalamannya sendiri untuk mengetahui kerentanan dalam sistem. Pengujian ini dapat dilakukan dengan rencana pengujian yang disiapkan (pengujian manual sistematis) atau tanpa rencana pengujian apa pun (pengujian manual eksplorasi). Teknik ini biayanya lebih murah dibandingkan dengan teknik lain, karena kita tidak perlu membeli alat penilaian kerentanan apa pun untuk teknik ini.

3. **UjicobaOtomatis (*Automated Testing*)**

   Dalam teknik pengujian otomatis tester menggunakan alat pengujian kerentanan otomatis untuk mengetahui kerentanan dalam sistem. Alat-alat ini mengeksekusi semua kasus uji untuk mengetahui kerentanan. Ini mengurangi jam kerja dan waktu tester wajib melakukan pengujian. Karena pengujian alat berulang juga dapat dilakukan dengan sangat mudah. Pengujian otomatis memberikan akurasi yang lebih baik daripada yang diberikan teknik lain. Dibutuhkan waktu yang sangat sedikit dan uji kasus yang sama dapat digunakan untuk operasi di masa depan. Tetapi alat meningkatkan biaya pengujian. Satu alat tidak mampu menemukan semua jenis kerentanan. Jadi ini meningkatkan total biaya untuk melakukan penilaian kerentanan.

4. **UjicobaFuzz (*Fuzz Testing*)**

   Ia juga dikenal sebagai *fuzzing*. Dalam hal ini tester memasukkan data yang tidak valid atau sembarang data ke dalam sistem dan kemudian mencari crash dan kegagalan. Ini seperti pengujian ketahanan. Teknik ini dapat diterapkan dengan interaksi manusia yang sangat sedikit. Teknik ini dapat digunakan untuk mengetahui kerentanan *zero day*.

Teknik Pengujian Penetrasi

1. **Black BoxTesting**

   Dalam teknik ini, tester tidak memiliki pengetahuan sebelumnya tentang arsitektur jaringan atau sistem jaringan pengujian. Biasanya pengujian kotak hitam dilakukan dari jaringan eksternal ke jaringan internal. Penguji harus menggunakan keahlian dan keterampilannya untuk melakukan pengujian ini.

2. **GreyBox Testing**

   Dalam teknik ini, tester memiliki pengetahuan parsial tentang jaringan pengujian. Penguji tidak memiliki pengetahuan tentang arsitektur jaringan yang lengkap, tetapi dia tahu beberapa informasi dasar pengujian jaringan dan konfigurasi sistem. Sebenarnya pengujian kotak abu-abu adalah kombinasi dari kedua teknik lainnya. Ini dapat dilakukan dari internal atau jaringan eksternal.

3. **WhiteBox Testing**

   Tester memiliki pengetahuan lengkap tentang konfigurasi jaringan dari jaringan pengujian dan sistem konfigurasi jaringan/sistem pengujian. Biasanya pengujian ini dilakukan dari jaringan internal. kotak putih pengujian membutuhkan pemahaman yang mendalam tentang jaringan atau sistem pengujian dan memberikan hasil yang lebih baik.

Open Web Application Security Project (OWASP)

OWASP atau singkatan dari *Open Web Application Security Project* yaitu sebuah organisasi nirlaba dengan misi untuk membuat keamanan perangkat lunak dapat dilihat oleh Individual dan Organisasi untuk membantu mereka mengambil keputusan berdasarkan informasi tentang risiko keamanan perangkat lunak mereka. OWASP TOP 10 adalah proyek keamanan yang disponsori oleh OWASP. Proyek tersebut mempublikasi sebuah daftar dari apa yang dianggap sebagai 10 teratas ancaman keamanan aplikasi web di seluruh dunia. Daftar tersebut menjelaskan kerentanan dengan contoh yang relevan dan cara menghindarinya [16].

Berdasarkan data dari OWASP Top 10 *Web Application Security Risks*, ada 10 kerentanan keamanan utama dalam sebuah aplikasi web, yaitu sebagai berikut [17]:

1. *Injection*

Kelemahan injeksi, seperti SQL, NoSQL, OS dan injeksi LDAP, terjadi ketika data yang tidak valid dikirim ke interpreter sebagai bagian dari perintah atau permintaan. Data dari penyerang dapat mengecoh interpreter untuk mengeksekusi perintah yang tidak diinginkan atau mengakses data tanpa otorisasi yang tepat.

2. *Broken Authentication*

   Fungsi aplikasi yang terkait dengan otentikasi dan manajemen sesi sering diterapkan secara tidak benar, memungkinkan penyerang untuk mengompromikan kata sandi, kunci, atau token sesi, atau untuk mengeksploitasi kelemahan implementasi lainnya untuk mengasumsikan identitas pengguna lain sementara atau secara permanen.

3. *Sensitive Data Exposure*

   Banyak aplikasi web dan API tidak melindungi data-data sensitif seperti keuangan, jaminan kesehatan, dll. Penyerang bisa mencuri atau mengubah data-data tersebut untuk melakukan penipuan kartu kredit, pencurian identitas, atau kejahatan lainnya. Data-data sensitif bisa dibongkar tanpa perlindungan ekstra, seperti enkripsi saat tidak dipakai atau saat dalam proses pengiriman, dan memerlukan tindakan pencegahan khusus ketika diakses oleh *browser*.

4. *XMLExternal Entities* (XXE)

   Banyak prosesor XML lama atau tidak dikonfigurasi dengan baik mengevaluasi referensi entitas eksternal dalam dokumen XML. Entitas eksternal dapat digunakan untuk mengungkapkan file internal menggunakan file URI handler, file share internal, pemindaian port internal, eksekusi kode jarak jauh, dan serangan *Denial of Service* (DoS).

5. *BrokenAccess Control*

   Pembatasan tentang apa yang diizinkan dilakukan oleh pengguna yang terotentikasi sering kali tidak dijalankan dengan benar. Penyerang dapat mengeksploitasi kelemahan ini untuk mengakses fungsionalitas dan/atau data yang tidak sah, seperti mengakses akun pengguna lain, melihat file sensitif, memodifikasi data pengguna lain, mengubah hak akses, dll.

6. *SecurityMisconfiguration*

   Kesalahan konfigurasi keamanan adalah masalah yang paling umum terjadi. Ini umumnya merupakan hasil dari konfigurasi default yang tidak aman, konfigurasi tidak lengkap, penyimpanan cloud terbuka, header HTTP yang salah konfigurasi, dan pesan kesalahan verbose yang berisi informasi sensitif. Semua sistem operasi, kerangka kerja, perpustakaan, dan aplikasi tidak hanya harus dikonfigurasikan dengan aman, tetapi juga harus ditambal/diperbaharui tepat waktu.

7. *Cross – Site Scripting* (*XSS*)

   Kekurangan XSS terjadi setiap kali suatu aplikasi memasukkan data yang tidak valid dalam halaman web baru tanpa validasi yang benar, atau memperbarui halaman web yang ada dengan data yang disediakan pengguna menggunakan API browser yang dapat membuat HTML atau JavaScript. XSS memungkinkan penyerang untuk mengeksekusi script di browser korban yang dapat membajak sesi pengguna, deface (penyisipan halaman) situs web, atau mengarahkan pengguna ke situs berbahaya.

8. *Insecure Deserialization*

   Deserialisasi yang tidak aman sering menyebabkan eksekusi kode jarak jauh. Bahkan walaupun kerentanan deserialisasi tidak berakibat pada eksekusi kode jarak jauh, maka ia dapat digunakan untuk melakukan serangan, termasuk serangan replay, serangan injeksi, dan serangan eskalasi hak akses (*privilege*).

9. *Using Components with Known Vulnerabilities*

   Komponen, seperti library, frameworks, dan modul perangkat lunak lainnya, dijalankan dengan hak akses yang sama dengan aplikasi. Jika komponen rentan dieksploitasi, serangan seperti itu dapat memfasilitasi kehilangan data yang serius atau pengambilalihan server. Aplikasi dan API yang menggunakan komponen dengan kerentanan yang telah diketahui

dapat merusak pertahanan aplikasi dan mengaktifkan berbagai serangan dan implikasi lainnya.

10. *Insufficient Logging & Monitoring*

Pencatatan dan pemantauan yang tidak memadai, ditambah dengan integrasi yang tidak efektif dengan respons insiden, memungkinkan penyerang untuk menyerang sistem lebih lanjut, merusak, mengekstrak, atau menghancurkan data. Sebagian besar studi tentang peretasan menunjukkan waktu untuk mendeteksi peretasan adalah lebih dari 200 hari, biasanya terdeteksi oleh pihak eksternal daripada proses internal atau pemantauan.

Sebelum proses pengujian kerentanan keamanan pada aplikasi web tersebut dijalankan, peneliti melakukan studi literatur dan kemudian menemukan beberapa penelitian terkait yang serupa. Pandikumar dan Eshetu dalam penelitiannya mengemukakan solusi *Tainted Mode Model* (TMM) yang memungkinkan deteksi kerentanan antar modul dan menyimpulkan bahwa secara efektif menggunakan *extended Tainted Mode Model* yang sebagian besar pendekatan yang ada didasarkan pada model kerentanan *Tainted Mode* yang tidak dapat menangani kerentanan antar modul. Manajemen kerentanan harus dipertimbangkan sebagai prioritas mengingat malware canggih yang menargetkan PC klien di dalam organisasi [3].

Ghozali, Kusrini, dan Sudarmawan dalam penelitiannya menghasilkan dua faktor untuk memperkirakan *Likelihood* dan *Impact*, dari masing-masing faktor terdapat tiga risiko yang ditemukan yaitu *risk severity High*, *risk severity Medium* dan *risk severity Low* [5]. Pratama dan Wiradarma dalam penelitiannya mendapatkan hasil pengujian dan analisis dengan framework OWASP versi 4 dengan modul Testing for Information Gathering menunjukkan bahwa sistem aplikasi web yang digunakan oleh Perusahaan X memiliki kerentanan informasi versi web server yang digunakan, permintaan GET dan POST, sistematika URL, framework website, komponen pembangun situs web, dan garis besar arsitektur situs web [6].

Riadi, Umar, dan Sukarno dalam penelitiannya menyimpulkan bahwa penggunaan Application Programming Interface (API) yang aman dapat menghindari interpreter, antarmuka berparameter, dan waspada terhadap prosedur yang tersimpan meskipun mereka berparameter tetapi masih ada celah. Jika tidak ada API berparameter, berhati-hatilah saat meneruskan karakter khusus penggunaan tertentu untuk meneruskan perintah interpreter. Teknik untuk melepaskan dari yang hal tersebut ada pada OWASP Java Encoder. Validasi positif atau rekomendasi white list memerlukan karakter khusus. ESAPI OWASP memberikan validasi input white list [7].

Pandya dan Patel dalam penelitiannya menemukan bahwa kerentanan kesalahan konfigurasi A5-keamanan adalah kontributor utama risiko keamanan web di situs web pemerintah. Selain itu, jelas terlihat bahwa sebagian besar kerentanan yang ditemukan di situs web pemerintah termasuk dalam kelompok berisiko rendah tetapi masih sedikit kerentanan yang berdampak tinggi yang ada dan perlu ditangani tanpa penundaan [16].

Idris, Majigi, Abdulhamid, Olalere, dan Rambo dalam penelitiannya menemukan bahwa kerentanan A4 - referensi objek langsung tidak aman dengan 49% merupakan kontributor utama risiko keamanan web di situs MDA. Selain itu, jelas terlihat bahwa mayoritas kerentanan yang ditemukan di situs MDA termasuk dalam kelompok risiko informasional dengan 45,82% tetapi masih sedikit kerentanan berdampak tinggi yang ada dan perlu ditangani tanpa penundaan [2].

Lubis dan Tarigan dalam hasil penelitiannya memberikan rekomendasi untuk perbaikan keamanan. Mengenai hasil penetrasi, administrator dapat memperbaiki kerentanan yang ada di situs tersebut. Dari pembahasan tersebut dapat disimpulkan bahwa pengujian penetrasi dilakukan untuk mengetahui kelemahan atau kerentanan pada aplikasi web domain dan subdomain di www.pancabudi.ac.id. menunjukkan bahwa masih ada celah yang memungkinkan untuk dieksploitasi oleh SQL Injection [4].

Metode Penelitian

Pengumpulan Data

Dalam mencari dan mendapatkan semua data-data yang diperlukan untuk penelitian dan pengujian kerentanan aplikasi web ini peneliti menggunakan metode pengumpulan data sebagai berikut:

1. Studi Literatur

Data-data didapatkan dari studi kepustakaan yang dilakukan dengan cara pencarian di situs-situs internet, jurnal-jurnal, perpustakaan kampus, dan juga buku-buku digital yang memiliki tema yang sesuai atau serupa dengan penelitian ini.

2. Observasi (Pengamatan)

Dalam hal ini, peneliti tidak terlibat dalam proses manipulasi (*input/output*) data yang ada dalam aplikasi web www.sea-mis.org yang menjadi target dari penelitian ini ataupun kode program sumbernya (*source code*). Namun sebagai staff IT yang bekerja pada kantor USAID SEA-PROJECT, peneliti bertanggung jawab dalam hal operasional dan perawatan Server tempat dimana aplikasi web www.sea-mis.org tersebut berjalan. Oleh karena itu, peneliti bisa mendapatkan data dengan melihat dan mencatat segala aktivitas jalannya Server tersebut namun terbatas pada data-data yang diperlukan untuk penelitian ini.

Perangkat yang Digunakan

Perangkat atau peralatan yang digunakan dalam penelitian ini yaitu berupa perangkat keras (*Hardware*) dan perangkat lunak (*Software*) adalah sebagai berikut:

1. **PerangkatKeras** (***Hardware***)

   Satu unit komputer laptop yang terinstal dua sistem operasi yang difungsikan sebagai penyerang dengan detail sebagai berikut:

   (a) MacBook Air 2014

   (b) Intel® Core™ i7-5650U CPU @ 2.20GHz

   (c) RAM 8.00 GB

   (d) Harddisk SSD 500GB

2. **PerangkatLunak** (***Software***)

   Yaitu semua program atau perangkat lunak yang terpasang dalam komputer laptop tersebut yang dijalankan untuk melakukan pengujian dalam penelitian ini adalah sebagai berikut:

   (a) Sistem Operasi MacOS Catalina dan Windows 10 Pro 64-bit, x64-based processor.

   (b) Software untuk pengujian yaitu OWASP Zed Attack Proxy dan RIPS.

   (c) Web Browser Microsoft Edge (Chromium based), Google Chrome, dan Mozilla Firefox.

   (d) VirtualBox 6.1.10 r138449 (Qt5.6.2)

Metode OWASP

Strategi pengujian kerentanan terhadap aplikasi web www.sea-mis.org menggunakan metode OWASP dalam penelitian ini dibagi ke dalam dua bentuk analisis, yaitu sebagai berikut:

1. **AnalisisDinamis** (***Dynamic Analysis***)

   Merupakan analisis yang dilakukan pada server atau domain yaitu tempat dimana aplikasi web yang menjadi target dari penelitian ini beroperasi. Semua data kerentanan didapatkan melalui proses pemindaian dan penilaian kerentanan pada tahapan ini. Pemindaian dilakukan dengan menggunakan perangkat lunak ZAP berdasar rekomendasi dari OWASP karena merupakan perangkat lunak Open Source, gratis, mudah, populer, dan aktif dikembangkan tim relawan internasional.

2. **AnalisisStatis** (***Static Analysis***)

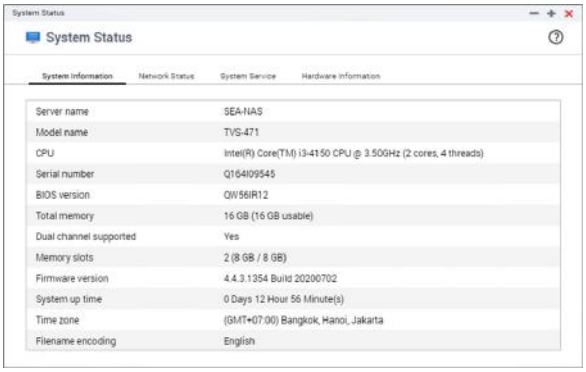
   Merupakan analisis yang dilakukan pada kode sumber (*source code*) dari aplikasi web yang menjadi target dari penelitian ini. Hal ini dimungkinkan karena aplikasi web www.sea- mis.org dibangun dan dimiliki secara internal oleh tempat peneliti bekerja sehari-hari. Pemindaian dilakukan menggunakan perangkat lunak RIPS juga berdasar rekomendasi dari OWASP karena mudah, gratis, dan dikhususkan untuk aplikasi web yang dibangun dengan PHP atau Java.

Penelitian sebelumnya yang juga menggunakan perangkat lunak yang sama yang disebutkan diatas misalnya seperti yang dilakukan oleh Idris, Majigi, Abdulhamid, Olalere, dan Rambo [2].

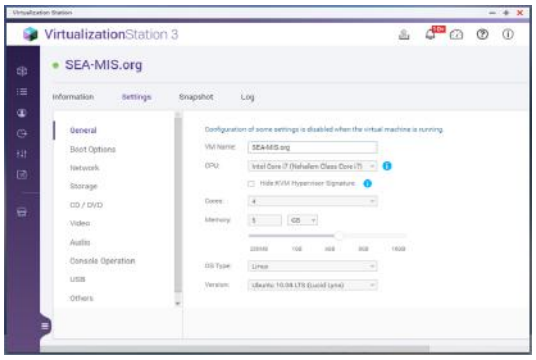
Hasil dan Pembahasan

Pemeriksaan Perangkat Sasaran

Saat tahapan pengujian akan mulai dilakukan, maka sebelumnya tentu saja dibutuhkan data dan informasi detail mengenai perangkat atau infrastruktur yang menjalankan objek yang menjadi target pengujian itu. Data dan informasi tersebut adalah sebagaimana yang tampak pada Gambar 1 dan Gambar 2.



Gambar 1: Spesifikasi Perangkat Sasaran – System Information)

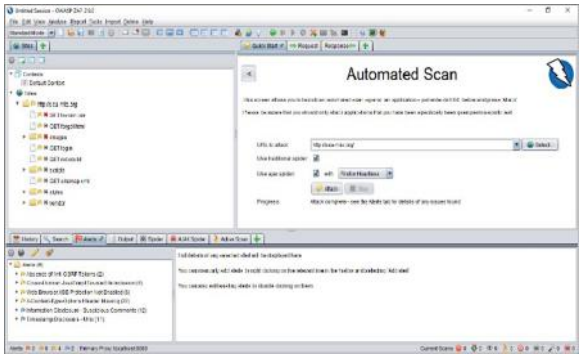


Gambar 2: Spesifikasi Perangkat Sasaran – Virtualization Station

Sebagaimana tampak pada Gambar 1 dan Gambar 2, bahwa server dari aplikasi web yang menjadi target penelitian ini dipasang dan berjalan pada sebuah Mesin Virtual yang beroperasi pada perangkat keras *Network Access Storage* (NAS) model TVS-471. Adapun server dari aplikasi web tersebut menggunakan sistem operasi Linux Ubuntu.

Pemindaian (Scanning)

Tahapan ini adalah saat alat uji coba kerentanan pada aplikasi web www.sea-mis.org mulai dieksekusi. Tahapan ini dilakukan agar didapatkan data dan informasi tentang sampai sejauh mana keamanan dari aplikasi web tersebut, segala risiko yang ada padanya, dan juga untuk menemukan kerentanan atau ancaman yang terdapat dalam aplikasi web tersebut. Tahapan ini telah dieksekusi pertama kali pada tanggal 5 Maret 2020, dan kemudian dilakukan kembali kedua kalinya pada tanggal 22 Juni 2020 yaitu setelah kerentanan yang ditemukan pada eksekusi pertama diperbaiki. Gambar 3 adalah tampilan permulaan pemindaian otomatis (*Automated Scan*) dengan ZAP.

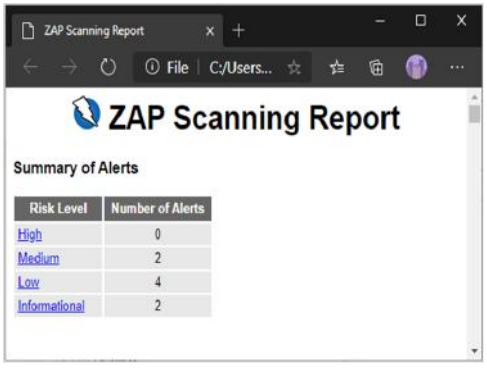


Gambar 3: Proses pemindaian menggunakan perangkat lunak ZAP dari OWASP

Hasil Pengujian

Setelah proses pemindaian aplikasi web www.sea- mis.org pertama kali selesai dilakukan dengan menggunakan perangkat lunak ZAP dari OWASP dan kemudian juga dilanjutkan dengan pemindaian kode sumber dengan menggunakan perangkat lunak RIPS juga dari OWASP.

A. Hasil Analisis Dinamis (Dynamic Analysis)



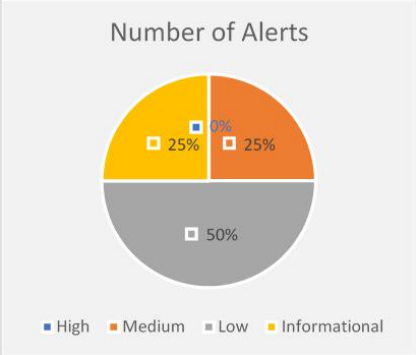
Gambar 4: Laporan Pemindaian ZAP (ZAP Scanning Report)

Tampak pada Gambar 4 adalah laporan hasil pemindaian menggunakan ZAP. Rekapitulasi dan rincian laporan hasil pemindaian dan kerentanan yang ditemukan adalah sebagaimana tampak pada Tabel 1, 2, dan 3.

Tabel 1: Rekapitulasi total jumlah kerentanan yang ditemukan

Tingkat Risiko (Risk Level)	Jumlah Kerentanan (Number of Alerts)
Tinggi (High)	0
Menengah (Medium)	2
Rendah (Low)	4
Informasi (Informational)	2

Berdasarkan data dalam Tabel 1, maka dapat digambarkan dalam bentuk persentase yaitu tingkat risiko kerentanan Tinggi (High) 0% yang berarti tidak ditemukan, Menengah (Medium) 25%, Rendah (Low) 50%, dan Informasi (Informational) 25% seperti Gambar 5.



Gambar 5: Persentase dari total jumlah kerentanan yang ditemukan

Gambar 6: Kerentanan tingkat Menengah (Medium) yang ditemukan saat dieksploitasi

Tabel 2: Rincian kerentanan tingkat Menengah (Medium)

Medium	Directory Browsing
Description	It is possible to view the directory listing. Directory listing may reveal hidden scripts, include files, backup source files, etc. which can be accessed to read sensitive information.
URL	http://sea-mis.org/scripts/jquery-validation/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/vendor/jquery/dist/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/vendor/jquery_appear/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/vendor/bootstrap/dist/css/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/vendor/slimScroll/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/vendor/bootstrap/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/scripts/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/vendor/jquery/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/vendor/fastclick/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/vendor/bootstrap/dist/js/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/styles/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/images/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/vendor/bootstrap/dist/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/vendor/jquery.easing/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/vendor/jquery.easing/
Method	GET
Attack	Parent Directory

URL	http://sea-mis.org/vendor/fastclick/lib/
Method	GET
Attack	Parent Directory
URL	http://sea-mis.org/vendor/
Method	GET
Attack	Parent Directory
Instances	16
Solution	Disable directory browsing. If this is required, make sure the listed files does not induce risks.
Reference	http://httpd.apache.org/docs/mod/core.html#options http://alamo.satlug.org/pipermail/satlug/2002-February/000053.html
CWE Id	548
WASC Id	48
Source ID	1

Gambar 7: Kerentanan tingkat Menengah (Medium) yang ditemukan saat dieksploitasi

Gambar 6 dan Gambar 7 memperlihatkan hasil ekploitasi dari celah kerentanan dimana bisa terlihat isi direktorinya.

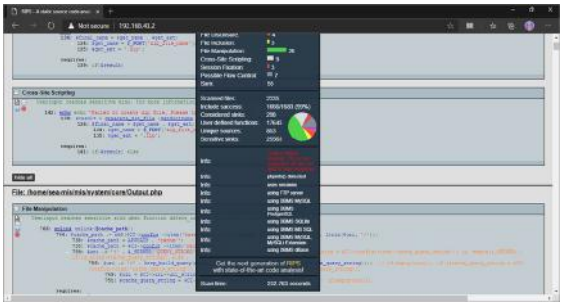
B. Hasil Analisis Statis (Static Analysis)

Hasil dari pemindaian kode sumber (source code) menggunakan perangkat lunak RIPS dari OWASP dapat dilihat pada Gambar 8 dan Gambar 9, tampak total file, jumlah dan jenis kerentanannya.

Tabel 3: Rincian kerentanan tingkat Menengah (Medium)

Medium	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://sea-mis.org/login
Method	GET
Parameter	X-Frame-Options
URL	http://sea-mis.org
Method	GET
Parameter	X-Frame-Options
Instances	2
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.aspx
CWE Id	16
WASC Id	15
Source ID	3

Gambar 8: Hasil pemindaian kode sumber (source code) dengan perangkat lunak RIPS



Gambar 9: Hasil pemindaian kode sumber (source code) dengan perangkat lunak RIPS

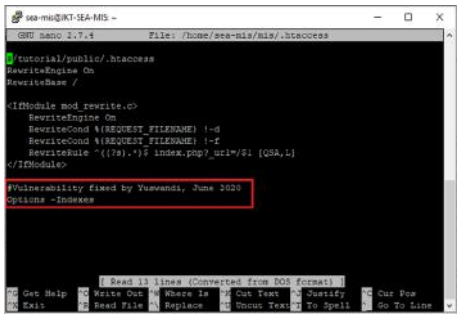
Tindak Lanjut

Solusi yang diberikan sebagai tindak lanjut perbaikan untuk kerentanan yang ditemukan

dalam penelitian ini difokuskan untuk memperbaiki kerentanan tingkat Menengah (*Medium*). Adapun perbaikan yang dilakukan terhadap kerentanan tingkat *Medium* kategori *Directory Browsing* adalah dengan menambahkan baris script berikut ini ke dalam file konfigurasi server yang bernama .htaccess:

`Options – Indexes`

Hasil penerapan script tersebut saat dibuka adalah sebagaimana tampak pada Gambar 10.

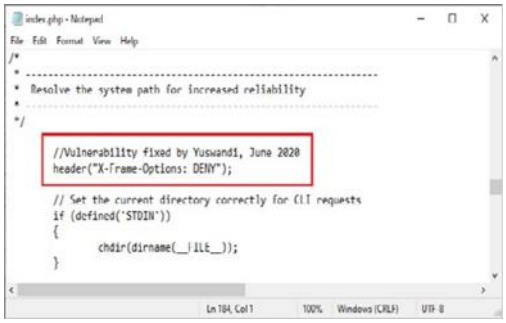


Gambar 10: Tindak lanjut perbaikan kerentanan tingkat Menengah (Medium)

Kemudian untuk perbaikan terhadap kerentanan tingkat Medium kategori X-Frame-Options Header Not Set adalah juga dengan menambahkan baris script berikut ini ke dalam file index.php pada kode sumbernya (source code):

`header("X – Frame – Options : DENY");`

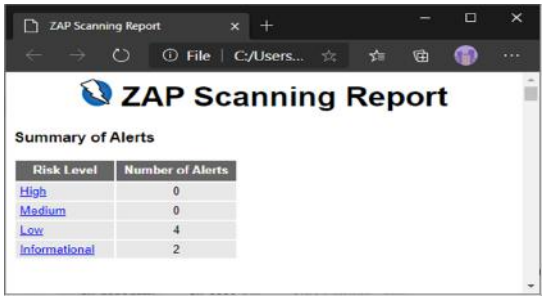
Hasil penerapan script tersebut adalah sebagaimana tampak pada Gambar 11.



Gambar 11: Tindak lanjut perbaikan kerentanan tingkat Medium

Setelah celah kerentanan tersebut berhasil ditutup dan diperbaiki, maka proses pemindaian dijalankan kembali untuk melihat perbedaan saat sebelum dan setelah perbaikan diterapkan. Laporan hasil pemindaian kembali tersebut adalah sebagaimana tampak pada Gambar 12.

Tampak pada Gambar 12 bahwa kerentanan tingkat Tinggi (*High*) dan Menengah (*Medium*) telah menjadi 0 (nol), yang berarti bahwa kerentanan yang paling kritis yaitu tingkat tinggi dan menengah telah berhasil diatasi.



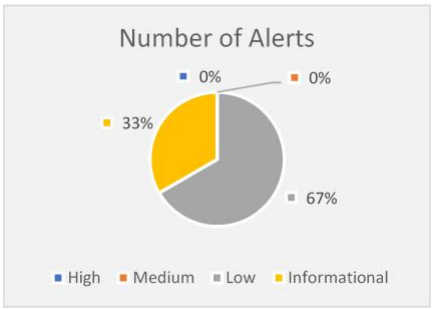
Gambar 12: Laporan Pemindaian ZAP (ZAP Scanning Report) setelah kerentanan tingkat Medium diperbaiki

Rekapitulasi dan rincian laporan hasil pemindaian dan kerentanan yang ditemukan setelah celah kerentanan berhasil ditutup dan diperbaiki adalah sebagaimana tampak pada Tabel 4.

Tabel 4: Rekapitulasi total jumlah kerentanan yang ditemukan setelah celah kerentanan ditutup dan diperbaiki

Tingkat Risiko (Risk Level)	Jumlah Kerentanan (Number of Alerts)
Tinggi (High)	0
Menengah (Medium)	0
Rendah (Low)	4
Informasi (Informational)	2

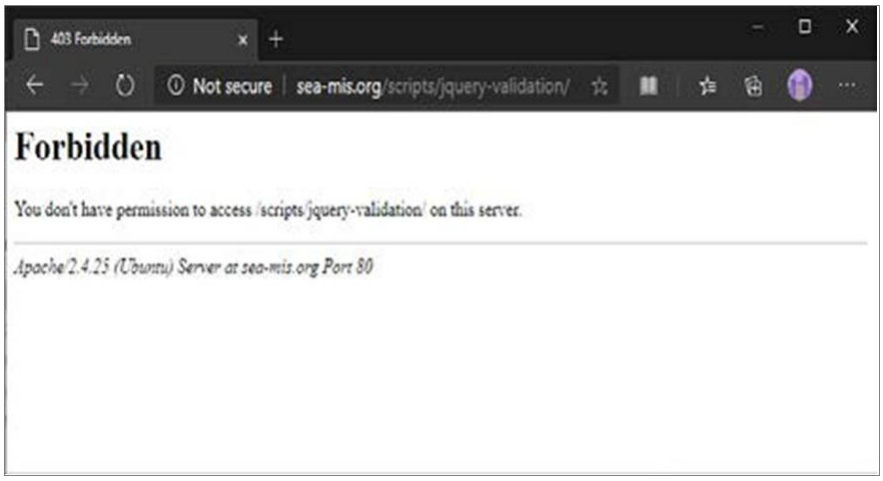
Berdasarkan data dalam Tabel 4, maka dapat digambarkan dalam bentuk persentase yaitu tingkat risiko kerentanan Tinggi (*High*) 0% yang berarti tidak ditemukan, Menengah (*Medium*) 0% yang berarti sudah tidak ditemukan lagi, Rendah (*Low*) 67%, dan Informasi (*Informational*) 33%.



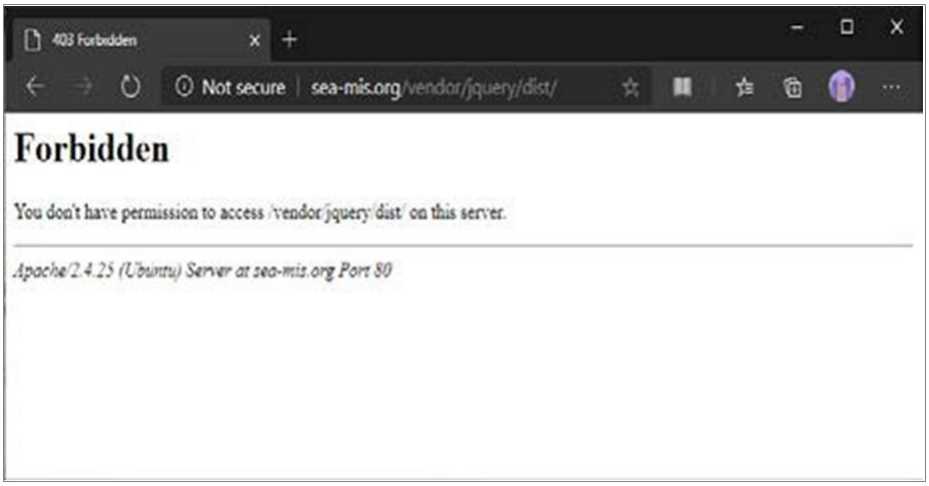
Gambar 13: Persentase dari total jumlah kerentanan yang ditemukan setelah celah kerentanan ditutup dan diperbaiki

Kemudian untuk memastikan bahwa kerentanan tingkat Menengah (*Medium*) yang ditemukan telah benar-benar tertutup maka dilakukan kembali percobaan eksploitasi terhadap celah tersebut, dan hasilnya sebagaimana tampak pada Gambar 14 dan Gambar 15.

Tampak pada Gambar 14 dan Gambar 15 bahwa celah kerentanan keamanan tingkat Menengah (*Medium*) yang ditemukan telah tertutup dan tidak dapat dieksploitasi lagi.



Gambar 14: Percobaan ekploitasi kerentanan tingkat Menengah (Medium) setelah celah kerentanan ditutup dan diperbaiki



Gambar 15: Percobaan ekploitasi kerentanan tingkat Menengah (Medium) setelah celah kerentanan ditutup dan diperbaiki

Penutup

Dari analisis kerentanan keamanan pada aplikasi berbasis web www.sea-mis.org berdasarkan panduan dari OWASP, maka dapat disimpulkan bahwa metode ini telah sukses menemukan adanya kelemahan atau celah kea-

manan dalam aplikasi tersebut. Hal ini juga membuktikan bahwa masih ada celah yang memungkinkan untuk dieksploitasi terutama dalam hal akses *directory* untuk mendapatkan file atau data tanpa melalui proses login.

Solusi perbaikan yang diberikan dalam penelitian ini telah terbukti berhasil dan dapat diterapkan langsung oleh pengelola aplikasi web www.sea-mis.org untuk menutup celah keamanan demi perbaikan dan peningkatan keamanan aplikasi tersebut.

Daftar Pustaka

[1] Tetra Tech ARD, "USAID SEA-PROJECT", tersedia pada URL: https://www.sea-indonesia.org/id/, waktu akses: 15 Juli 2020

[2] Ismaila Idris., Mohammad Umar Majigi., Shafii Abdulhamid., Morufu Olalere, Saidu Isah Rambo, "Vulnerability Assessment of Some Key Nigeria Government Websites", International Journal of Digital Information and Wireless Communications (IJDIWC) 7(2): 143-152, ISSN: 2225-658X (Online); ISSN 2412-6551 (Print), The Society of Digital Information and Wireless Communications, 2017.

[3] T. Pandikumar, Tseday Eshetu, "Detecting Web Application Vulnerability using Dynamic Analysis with Penetration Testing", International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, p-ISSN: 2395-0072, Volume: 03 Issue: 10 | Oct-2016, www.irjet.net, 2016.

[4] Akhyar Lubis, Avinanta Tarigan, "Security Assessment of Web Application Through Penetration System Techniques", International Journal of Recent Trends in Engineering & Research (IJRTER), Volume 03, Issue 01, January 2017, ISSN (Online): 2455-1457, @IJRTER, 2017.

[5] Bahrun Ghozali, Kusrini, Sudarmawan, "Mendeteksi Kerentanan Keamanan Aplikasi Website Menggunakan Metode Owasp (Open Web Application Security Project) untuk Penilaian Risk Rating", Citec Journal, Vol. 4, No. 4, Agustus 2017 - Oktober 2017, ISSN: 2460- 4259, Universitas AMIKOM Yogyakarta, Dikirim: 09 Februari 2018; Direvisi: 12 Februari 2018, 19 Februari 2018; Diterima: 21 Februari 2018.

[6] I Putu Agus Eka Pratama, Anak Agung, "Open Source Intelligence Testing Using the OWASP Version 4 Framework at the Information Gathering Stage (Case Study: X Company)". MECS (http://www.mecs-press.org/), Vol. 7, No. 8-12, I. J. Computer Network and Information Security, 2019.

[7] Imam Riadi, Rusydi Umar, Wasito Sukarno, "Vulnerability of injection attacks against the application security of framework-based websites open web access security project (OWASP)", JURNAL INFORMATIKA, ISSN: 1978-0524, Vol. 12, No. 2, Department of Information System, Universitas Ahmad Dahlan, Yogyakarta 55164, Indonesia, 2018.

[8] I. A. Ajayi, Omirin, Fadekemi F, "The Use of Management Information Systems (MIS) In Decision Making In The South-West Nigerian Universities", Educational Research and Review, Vol. 2, No. 5, pp. 109-116, 2007.

[9] A. Heidarkhani, A.A. Khomami, Q. Jahanbazi and H. Alipoor, "The Role of Management Information Systems (MIS) in Decision-Making and Problems of its Implementation", Universal Journal of Management and Social Sciences, Vol. 3, No. 3, pp. 78-89, 2013.

[10] Anonymous, "W3C's,"Architecture of the World Wide Web Volume One", diakses daring pada URL: https://www.w3.org/Help/#webinternet, waktu akses: 9 Juni 2020.

[11] Michael E. Whitman, Herbert J. Mattord, "Principles of Information Security", Course Technology, Cengage Learning, ISBN-13: 978-1- 111-13821-9. ISBN-10: 1-111-13821-4, 2011.

[12] J.A.. O'Brien and G.M. Marakas, "Management information systems-10th ed", McGraw- Hill/Irwin, a business unit of The McGraw-Hill Companies, 2007.

[13] Hiroyuki Okamura, Masataka Tokuzane, Tadashi Dohi, "Optimal Security Patch Release Timing under Non-homogeneous Vulnerability- Discovery Processes", Researchgate, 120-128. 10.1109/ISSRE.2009.19, 2009.

[14] Ron Lepofsky, "The Manager's Guide to Web Application Security: A Concise Guide to the Weaker Side of the Web", Springer Science + Business Media New York, ISBN-13 (pbk): 978- 1-4842-0149-7. ISBN-13 (electronic): 978-1- 4842-0148-0, 2014.

[15] Jai Narayan Goel, BM Mehtre, "Vulnerability Assessment & Penetration Testing as a Cyber Defence Technology",Elsevier B.V., Science Direct, Procedia Computer Science, 57 710-715, 2015.

[16] Deven Pandya, Dr. N. J. Patel, "OWASP Top 10 Vulnerability Analyses in Government Websites", International Journal of Enterprise Computing and Business Systems, ISSN (Online): 2230- 8849, Vol. 6, Council of Scientific and Industrial Research, Ministry of Science and Technology, Govt. of India, 2016.

[17] OWASP, "Top 10 Web Application Security Risks", diakses daring pada URL: https://owasp.org/www-project-top-ten/, waktu akses: 22 Februari 2020.