JIKB
Jurnal Ilmu Komputer dan Bisnis
STMIK DHARMAPALA RIAU

Jurnal Ilmu Komputer dan Bisnis (JIKB)
Mei-2022, Vol. XIII, No.1, hal.94-104
ISSN(P): 2087-3921; ISSN(E): 2598-9715

Peningkatan Sistem Keamanan *Website* Menggunakan Metode *OWASP*

[1]Haeruddin,[2]Hermanto
[1,2]Universitas Internasional Batam

Alamat Surat
Email: [1]Haeruddin@uib.edu, [2]1831144.hermanto@uib.edi

Article History:
Diajukan: 27 Maret 2021; Direvisi: 15 April 2022; Diterima: 25 April 2022

ABSTRAK

Layanan *website* itu telah menjadi sebuah *Cross-Platform* yang paling sering digunakan oleh setiap orang mendapatkan informasi yang ada yang dibutuhkan. *Website* merupakan situs yang di publikasikan di internet dan dapat diakses oleh semua pengguna internet, sehingga sistem keamanan sangat penting untuk menjaga *website* tetap aman terhadap ancaman ancaman serangan peretas. Maka dari itu, perlu melakukan pengujian keamanan layanan *website*. Dalam melakukan pengujian sistem keamanan *website* metode yang digunakan adalah *Open Web Application Security Project* (*OWASP*). *OWASP* adalah sebuah *framework open source* yang berfungsi untuk memberikan informasi celah kemananan dan rekomendasi perbaikan yang dapat dilakukan pada layanan berbasis *web*. Metode ini telah menjadi sebuah petunjuk untuk menganalisa dan melakukan pengecekkan keamanan suatu *website*. Dengan adanya analisa ini, kita bisa mengetahui risiko ancaman yang muncul pada suatu layanan *website* dan mempermudah melakukan tindakan pencegahan berdasarkan rekomendasi yang telah di dapatkan.

Kata kunci: Sistem Keamanan *Web, OWASP, OWASP Zap Zed Attack Proxy*

ABSTRACT

Website service has become a Cross-Platform that is most often used by everyone to get the information they need. Website is a site that is published on the internet and can be accessed by all internet users, so security system is very important to keep the website safe from the threat of hacker attacks. Therefore, it is necessary to test the security of website services. In testing the website security system, the method used is Open Web Application Security Project (OWASP). OWASP is an open source framework that serves to provide information on security vulnerabilities and recommendations for improvements that can be made to web-based services. This method has become a guide for analyzing and checking the security of a website. With this analysis, we can find out the risk of threats that arise on a website service and make it easier to take preventive actions based on the recommendations that have been obtained

Keywords: Web Security System, OWASP, OWASP Zap Zed Attack Proxy

1. PENDAHULUAN

Di era digital saat ini konsen terhadap masalah keamanan dan integritas digital sedang hangat di perbincangkan. Umumnya konsen terhadap masalah keamanan sistem informasi selalu diposisikan kedua atau lebih (Ghozali et al., 2018). Ini membuat banyak serangan *cyber* yang menargetkan terhadap kerentanan suatu sistem informasi yang tidak diperhatikan secara baik oleh pemilik. Salah satu contoh masalah keamanan sistem informasi terjadi pada layanan *web* atau *web services*. *Web*

https://doi.org/10.47927/jikb.v13i1.277

Services memiliki sebuah keunikan, yaitu dapat diakses *cross platform*. Kelebihan tersebut kemudian memicukan masalah keamanan dapat dengan mudah diretas jika sistem keamanan pada layanan *web services* tersebut tidak dikonfigurasi dengan baik (Widhiyanto, 2019). Oleh Karena itu, data-data pengguna akan disalahgunakan oleh pihak yang tidak bertanggung jawab. Keamanan *website* adalah suatu cabang teknologi yang dikenal dengan nama keamanan informasi yang diterapkan pada *website* dengan sasaran perlindungan terhadap informasi/data (Muttaqien, 2019; Wismarini & Prihandono, 2020).

Jika dilihat dari persentase serangan, terdapat beberapa sisi kerentanan pada layanan *web* seperti *Cross Site Scripting* (*XSS*) sebesar 35,57%, *SQL injection* 26,38%, *Information Leakage* sebesar 15,70%, *HTTP Response Splitting* sebesar 9,76%, *Path Traversal* sebesar 1,19% dan serangan lainnya sebesar 4,30% Qian et al., (2018). Salah satu faktor yang menyebabkan kurangnya tingkat keamanan pada aplikasi *website* adalah kesalahan penulisan kode program. Kesalahan pada penulisan kode program pada *website* dapat dijadikan celah yang di manfaatkan oleh penyerang untuk melakukan serangan seperti *SQL Injection, Authentication* dan *XSS* (Ghozali et al., 2018).

Universitas XYZ merupakan salah satu kampus yang berada di kota Batam, dimana semua proses bisnisnya telah menggunakan sistem informasi berbasis website (Haeruddin, 2019). Terdapat 37 *website* dengan sub domain yang berbeda yang digunakan untuk melayani kebutuhan civitas akademik. Pada peneltian (Haeruddin, 2019) telah melakan pemetaan aset sistem informasi dalam penerapan manajemen risiko menggunakan metode *OCATVE Allegro*, namun belum melakukan pengujian sistem informasi berbasis *web* secara khusus. Sehingga perlu dilakukan pengujian untuk mengetahui apakah terdapat kerentanan pada semua sistem tersebut. Pengujian suatu sistem sangatlah diperlukan untuk mengetahui celah keamana yang ada. Salah satu metode yang dapat digunakan adalah *Open Web Application Security Project* (*OWASP*)(Ghozali et al., 2018). OWASP adalah sebuah *framework open source* yang berfungsi untuk memberikan informasi celah kemananan serta rekomendasi perbaikan yang dapat dilakukan pada layanan berbasis *web*.

2. METODE

Pada penelitian ini, untuk mendeteksi sistem keamanan *website* 43.255.184.109, penulis menggunakan metode penilaian risiko OWASP. OWASP juga dikenal sebagai organisasi nonprofit amal di Amerika Serikat berdiri pada tahun 2004 dan dilengkapi *standart Guide* untuk mempermudah *pentration testing*. Metodologi penilaian risiko OWASP adalah pendekatan sederhana untuk menghitung dan menilai risiko yang terkait dengan aplikasi (Dewanto, 2018; Eka Pratama & Wiradarma, 2019; Kurniawan, 2019; Li, 2020); dimana dengan metode tersebut dapat diputuskan apa saja yang harus dilakukan terhadap resiko-resiko tersebut. Dengan mengetahui resiko yang akan terjadi maka banyak manfaat yang akan diperoleh diantaranya, menghemat waktu dan mengurangi terjadinya resiko yang lebih serius. Pada tahap ini juga terdapat beberapa tahapan OWASP yaitu:

1. *Information Gathering*
   Tahap ini penulis mulai menemukan versi dan jenis *server web* yang berjalan untuk menentukan mengetahui kerentanan dan eksploitasi yang sesuai untuk digunakan selama pengujian.
2. *Session Management Testing*
   Ditahap ini penguji harus mengetahui semua *cookie* yang disetel, dan bahwa konfigurasi keamanan yang tepat sedang diterapkan.
3. *Data Validation Testing*
   Tahap ini penulis akan melakukan validasi input untuk memastikan hanya data yang terbentuk dengan benar yang memasuki alur kerja dalam sistem informasi, mencegah data yang cacat tetap ada dalam *database* dan memicu kerusakan. Perlu di ketahui juga bahwa validasi *Input* tidak boleh digunakan sebagai metode utama untuk mencegah *XSS, SQL Injection,* dan serangan lain yang tercakup dalam lembar contekan masing-masing, tetapi dapat berkontribusi secara signifikan untuk mengurangi dampaknya jika diterapkan dengan benar.

4. *Webservices Testing*
   Pada tahap penguji mulai pengujian dengan eksekusi kode pada sisi klien, biasanya secara native dalam browser *web* atau *browser plugin*. Eksekusi kode di sisi klien berbeda dari mengeksekusi di *server* dan mengembalikan konten berikutnya.

3. HASIL DAN PEMBAHASAN

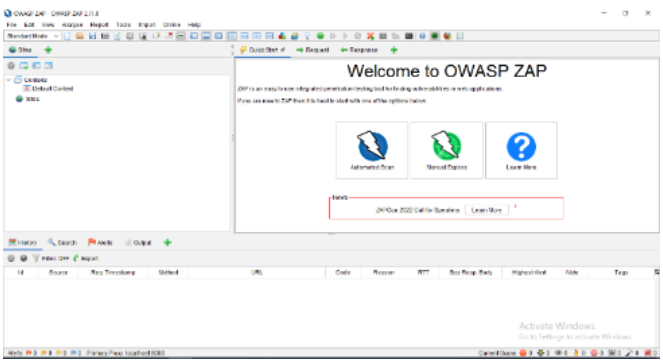
3.1 *Information Gathering*

Dikarenakan penulis sudah mengetahui *website* yang mau coba keamanannya maka penulis langsung melaksanakan testing pada *website* yang mau di coba yaitu 43.255.184.109.

3.2 *Session Management Testing*

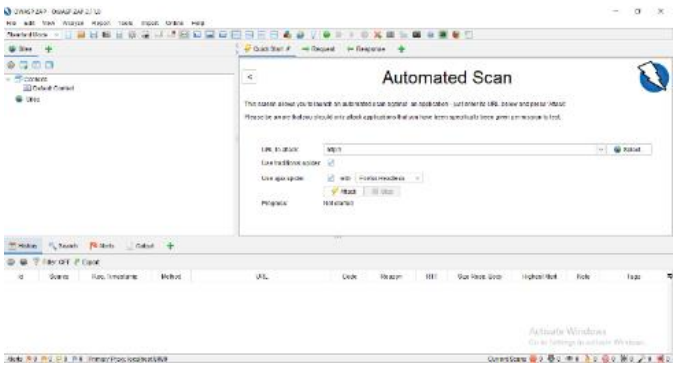
Cara Pengujian

1. Buka Program *OWASP* nya, maka akan muncul tampilan seperti yang ditunjukkan pada gambar 1



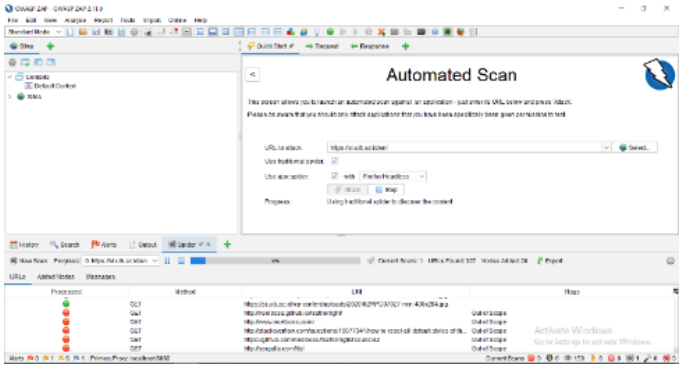
Gambar 1. Tampilan *OWASP ZAP*

2. Setelah itu kita bisa liat dari gambar 1 kemudian klik *automate scan* yang ada pada kolom *welcome to OWASP Zap* setelah diklik maka akan muncul tampulan seperti yang di tunjukkan pada gambar 2



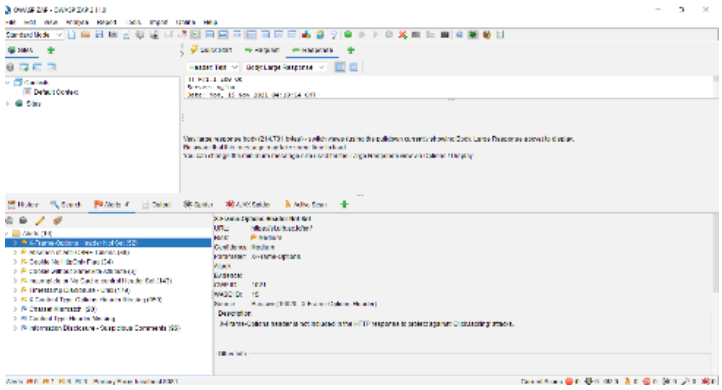
Gambar 2. Tampilan Halaman Utama *OWASP ZAP*

3. Setelah itu pada kolom *URL to Attack* itu kita masukkin *url* yang mau kita *attack* contoh nya *website* 43.255.184.109, klik centang pada *use traditional spider* dan *use ajax spider* untuk pembantuan scan dan klik *start* pada tahap selanjutnya.
4. Maka *OWASP* akan memulai *scan* seperti yang di tunjukkan pada gambar 3



Gambar 3. Tampilan *Scanning* Pada *OWASP ZAP*

5. Setelah siap scan maka kita ada dapat alert nya dari website yang kita attack itu seperti yang ditunjukkan pada gambar 4



Gambar 4. Tampilan Hasil *Scanning* Pada *OWASP ZAP*

6. Setelah siap *scan* kita ambil data nya dan mengulangnya sampai *website* yang mau kita scan semuanya udah siap.

Dari hasil pengujian diatas itu kita bisa tahu bahwa pada *website* 43.255.184.109 telah ditemukan 2 macam *cookies* padanya yaitu *Cookie without Samesite Attribute* dan *Cookie No HttpOnly Flag.*

3.3 *Data Validation Testing*

Cara Pengujian

1. Buka Program OWASP nya, maka akan muncul tampilan seperti yang ditunjukkan pada gambar 1.
2. Setelah itu kita bisa liat dari gambar 1 kemudian klik *automate scan* yang ada pada kolum *welcome to OWASP Zap* setelah diklik maka akan muncul tampulan seperti yang di tunjukkan pada gambar 2.
3. Setelah itu pada kolom *URL to Attack* itu kita masukkin url yang mau kita *attack* contoh nya *website* 43.255.184.109, klik centang pada *use traditional spider* dan *use ajax spider* untuk pembantuan *scan* dan klik *start* pada tahap selanjutnya.
4. Maka *OWASP* akan memulai *scan* seperti yang di tunjukkan pada gambar 3
5. Setelah siap scan maka kita ada dapat *alert* nya dari *website* yang kita *attack* itu seperti yang ditunjukkan pada gambar 4.
6. Dari hasil pengujian diatas itu kita bisa tahu bahwa pada *website* 43.255.184.109 telah ditemukan masalah *Cross Site Scripting (Reflected).*

3.4 *Webservices Testing*

Cara Pengujian

1. Buka Program OWASP nya, maka akan muncul tampilan seperti yang ditunjukkan pada gambar 1.
2. Setelah itu kita bisa liat dari gambar 1 kemudian klik *automate scan* yang ada pada kolum *welcome to OWASP Zap* setelah diklik maka akan muncul tampulan seperti yang di tunjukkan pada gambar 2.
3. Setelah itu pada kolom *URL to Attack* itu kita masukkin *url* yang mau kita *attack* contoh nya *website* 43.255.184.109, klik centang pada *use traditional spider* dan *use ajax spider* untuk pembantuan *scan* dan klik *start* pada tahap selanjutnya.
4. Maka *OWASP* akan memulai *scan* seperti yang di tunjukkan pada gambar 3 .
5. Setelah siap *scan* maka kita ada dapat *alert* nya dari *website* yang kita *attack* itu seperti yang ditunjukkan pada gambar 4.

Dari hasil pengujian diatas itu kita bisa tahu bahwa pada *website* 43.255.184.109 telah ditemukan masalah seperti *SQL Injection, .htaccess Information Leak, Directory Browsing, Vulnerable JS Library, Incomplete or No Cache-control Header Set, Timestamp Disclosure – Unix, Information Disclosure - Sensitive Information in URL, Information Disclosure - Suspicious Comments, X-Frame-Options Header Not Set, Absence of Anti-CSRF Tokens, X-Content-Type-Options Header Missing, Cookie No HttpOnly Flag dan Cookie without SameSite Attribute.*

3.5 Hasil Penelitian

Dari semua testing yang dilakukan kita mendapatkan pada gambar 4.6 akan menunjukan celah keamanan dan total ancaman yang ada di*website* 43.255.184.109.

Tabel 1. Celah Keamanan yang Muncul

Subdomain	Masalah / Jenis Ancaman	Total	Risk
43.255.184.109	Cross Site Scripting (Reflected) (1)	1	High
	SQL Injection (4)	4	High
	.htaccess Information Leak (93)	93	Medium
	Directory Browsing (7)	7	Medium
	Vulnerable JS Library (2)	2	Medium
	X-Frame-Options Header Not Set (177)	177	Medium
	Absence of Anti-CSRF Tokens (360)	360	Low
	Cookie No HttpOnly Flag (3)	3	Low
	Cookie without SameSite Attribute (3)	3	Low
	Incomplete or No Cache-control Header Set (187)	187	Low
	Timestamp Disclosure - Unix (2902)	2902	Low
	X-Content-Type-Options Header Missing (565)	565	Low
	Information Disclosure - Sensitive Information in URL (1)	1	Informational
	Information Disclosure - Suspicious Comments (555)	555	Informational

Berdasarkan dari hasil testing untuk menentukan *OWASP Risk Rating* dari ancaman tersebut ada beberapa tahapan dapat menentukan berapa besarnya resiko yang akan muncul, tahapan tersebut merupakan *Threat Agent Factors, Vulnerability Factors, Technical Impact, Business Impact.*

Tabel 2. Kriteria Pengukuran *Threat Agent*

No	*Threat Agent Factor*	Kriteris Penilaian
1	Tingkat Keterampilan	Seberapa terampil secara teknis kelompok threat agent? tidak ada keterampilan teknis (1) beberapa keterampilan teknis (3) pengguna komputer tingkat lanjut (5) keterampilan jaringan dan pemrograman (6) keterampilan penetrasi keamanan (9).
2	Motif	Seberapa termotivasikan kelompok threat agent untuk menemukan dan mengeksploitasi kerentanan? tidak ada hadiah (1) kemungkinan hadiah (4) hadiah tinggi (9)
3	Peluang	Sumber daya dan peluang apa yang dibutuhkan kelompok threat agent untuk menemukan dan mengeksploitasi kerentanan ini? akses penuh atau sumber daya yang mahal (0) akses atau sumber daya yang khusus (4) beberapa akses atau sumber daya yang yang diperlukan (7)
4	Ukuran	Seberapa besar kelompok *threat agent*? Pengembang (2) administrator sistem (2) pengguna intranet (4) mitra (5) pengguna yang terotentikasi (6) pengguna internet anonim (9)

$$\text{Threat Agent} = \frac{\text{Tingkat Keterampilan} + \text{Motif} + \text{Peluang} + \text{Ukuran}}{4}$$

Gambar 5. Rumus Hasil Threat Agents

Berikut hasil dari faktor *threat agent* yang ditunjukkan pada tabel 3

Tabel 3. Skor Threat Agent Factors

Jenis Ancaman	Tingkat Keterampilan	Motif	Peluang	Ukuran
Cross Site Scripting (Reflected)	9	9	4	9
SQL Injection	9	9	4	2
.htaccess Information Leak	6	4	4	6
Directory Browsing	6	4	4	6
Vulnerable JS Library	6	4	4	9
X-Frame-Options Header Not Set	6	4	7	4
Absence of Anti-CSRF Tokens	9	4	7	9
Cookie No HttpOnly Flag	6	4	4	6
Cookie without SameSite Attribute	6	4	4	6
Incomplete or No Cache-control Header Set	6	4	7	4
Timestamp Disclosure - Unix	6	4	9	9
X-Content-Type-Options Header Missing	6	4	7	4
Information Disclosure - Sensitive Information in URL	5	4	9	9
Information Disclosure - Suspicious Comments	6	4	9	2

Tabel 4.Kriteria Pengukuran Vulnerability Factors

No	*Vulnerability Factors*	Kriteris Penilaian
1	Kemudahan Penemuan	Seberapa mudah kelompok threat agent untuk menemukan kerentanan ini? cara praktis tidak mungkin (1) sulit (3) mudah (7) tersedia alat otomatis (9)
2	Kemudahan Eksploitasi	Seberapa mudah kelompok threat agent untuk benar-benar mengeksploitasi kerentanan ini? teoritis (1) sulit (3) mudah (5) tersedia alat otomatis (9)
3	Kesadaran	Seberapa terkenal kerentanan ini terhadap kelompok threat agent? tidak diketahui (1) tersembunyi (4) jelas (6)
4	Deteksi Intrusi	Seberapa besar kemungkinan eksploitasi terdeteksi? Deteksi aktif dalam aplikasi (1) dicatat dan ditinjau (3) dicatat tanpa ditinjau (8) tidak dicatat (9)

$$\text{Vulnerability} = \frac{\text{Kemudahan Penemuan} + \text{Kemudahan Eksploitasi} + \text{Kesadaran} + \text{Deteksi Intrusi}}{4}$$

Gambar 6. Rumus Hasil *Vulnerability Factors*

Berikut hasil dari *vulnerability factors* yang ditunjukkan pada tabel 5

Tabel 5. Skor Vulnerability Factors

Jenis Ancaman	Kemudahan Penemuan	Kemudahan Eksploitasi	Kesadaran	Deteksi Intrusi
Cross Site Scripting (Reflected)	9	5	9	1
SQL Injection	9	5	9	1
.htaccess Information Leak	9	3	4	1
Directory Browsing	9	3	4	1
Vulnerable JS Library	9	3	4	1
X-Frame-Options Header Not Set	9	3	9	1
Absence of Anti-CSRF Tokens	9	3	9	1
Cookie No HttpOnly Flag	9	3	9	1
Cookie without SameSite Attribute	9	3	9	1
Incomplete or No Cache-control Header Set	9	3	9	1
Timestamp Disclosure - Unix	9	5	9	1
X-Content-Type-Options Header Missing	9	3	9	1
Information Disclosure - Sensitive Information in URL	9	3	4	1
Information Disclosure - Suspicious Comments	9	3	4	1

Tabel 6. Kriteria Pengukuran *Vulnerability Factors*

No	*Technical Impact*	Kriteris Penilaian
1	Kehilangan Kerahasiaan	Berapa banyak data yang dapat diungkapkan dan seberapa sensitif? Data yang diungkapkan minimum dan tidak sensitif (2) minimal data penting yang diungkapkan (6) data tidak sensitif ekstensif yang diungkapkan (6) data penting dan ekstensif diungkapkan (7) semua data yang diungkapkan (9)
2	Kehilangan Integritas	Berapa data yang rusak dan seberapa rusaknya? Data yang korup minimal sedikit (1) data korup yang tidak serius (3) data yang agak korup(7) semua data benar-benar korup (9)
3	Kehilangan Ketersediaan	Berapa banyak layanan yang bisa hilang dan seberapa pentingnya layanan tersebut? Layanan sekunder minimal terputus (1) layanan primer minimal terputus (5) layanan sekunder yang ekstensif terputus (5) layanan primer yang ekstensif terputus (7) semua layanan benar-benar hilang total (9)
4	Kehilangan Akuntabilitas	Apakah tindakan threat agent bisa dilacak pada individu? dapat dilacak sepenuhnya (1) kemungkinan dapat dilacak (7) sepenuhnya anonim (9)

$$\text{Technical Impact} = \frac{\text{Kehilangan Kerahasiaan} + \text{Kehilangan Integritas} + \text{Kehilangan Ketersediaan} + \text{Kehilangan Akuntabilitas}}{4}$$

Gambar 7. Rumus Hasil *Techincal Impact*

Berikut hasil dari *Technical Impact* yang ditunjukkan pada tabel 7

Tabel 7. Skor *Technical Impact*

Jenis Ancaman	Kehilangan Kerahasiaan	Kehilangan Integritas	Kehilangan Ketersediaan	Kehilangan Akuntabilitas
Cross Site Scripting (Reflected)	2	1	1	9
SQL Injection	2	1	1	9
.htaccess Information Leak	6	3	1	7
Directory Browsing	2	1	1	1
Vulnerable JS Library	2	1	1	1
X-Frame-Options Header Not Set	6	1	1	7
Absence of Anti-CSRF Tokens	6	3	1	9
Cookie No HttpOnly Flag	2	3	1	7
Cookie without SameSite Attribute	2	3	1	7
Incomplete or No Cache-control Header Set	6	1	1	7
Timestamp Disclosure - Unix	2	1	1	7
X-Content-Type-Options Header Missing	6	1	1	7
Information Disclosure - Sensitive Information in URL	2	1	1	7
Information Disclosure - Suspicious Comments	2	3	1	7

Tabel 8. Kriteria Pengukuran *Business Impact*

No	*Business Impact*	Kriteris Penilaian
1	Kerusakan Finansial	Berapa banyak kerusakan finansial yang diakibatkan dari eksploitasi? kurang dari biaya untuk memperbaiki kerentanan (1) pengaruh kecil terhadap laba tahunan (3) berpengaruh signifikan terhadap laba tahunan (7) kebangkrutan (9).
2	Kerusakan Reputasi	Apakah hasil eksploitasi akan mengakibatkan merusak reputasi yang akan merugikan bisnis? kerusakan minimal (1) kehilangan akun utama (4) kehilangan niat baik (5) kerusakan merek (9)
3	Ketidakpatuhan	Berapa banyak paparan yang ditimbulkan oleh ketidakpatuhan? pelanggaran kecil (2) pelanggaran jelas (5) pelanggaran profil tinggi (7)
4	Kehilangan Akuntabilitas	Berapa banyak informasi identitas pribadi yang bisa diungkapkan? Satu orang (3) ratusan orang (5) ribuan orang (7)

$$\text{Business Impact} = \frac{\text{Kerusakan Finansial} + \text{Kerusakan Reputasi} + \text{Ketidakpatuhan} + \text{Pelanggaran Privasi}}{4}$$

Gambar 8. Rumus Hasil Techincal Impact

Berikut hasil dari *Technical Impact* yang ditunjukkan pada tabel 9

Tabel 9. Skor *Technical Impact*

Jenis Ancaman	Kerusakan Finansial	Kerusakan Reputasi	Ketidakpatuhan	Pelanggaran Privasi
Cross Site Scripting (Reflected)	1	1	2	3
SQL Injection	1	1	2	3
.htaccess Information Leak	1	1	2	3
Directory Browsing	1	1	2	3
Vulnerable JS Library	1	1	2	3
X-Frame-Options Header Not Set	1	1	2	3
Absence of Anti-CSRF Tokens	1	1	2	3
Cookie No HttpOnly Flag	3	1	2	3
Cookie without SameSite Attribute	3	1	2	3
Incomplete or No Cache-control Header Set	1	1	2	3
Timestamp Disclosure - Unix	1	1	2	3
X-Content-Type-Options Header Missing	1	1	2	3
Information Disclosure - Sensitive Information in URL	3	4	2	3
Information Disclosure - Suspicious Comments	3	4	2	3

Website yang telah di*scan* itu akan secara automatis terdapat beberapa kerentanan ynag muncul. Dari kerentanan tersebut dapat dihasilkan tingkat resiko keamanan yang akan dihitung dengan rumus dibawah ini:

$$\text{Likelihood} = \frac{\textit{Threat Agent Factor} + \textit{Vulnerability Factor}}{2}$$

Gambar 9. Rumus Perhitungan *Likelihood*

$$\text{Impact} = \frac{\textit{Technical Impact} + \textit{Business Impact}}{2}$$

Gambar 10. Rumus Peritungan *Impact*

Skor secara keseluruhan *Likelihood* dan *Impact* dari hasil pegukuran yang telah dilaksanakan diatas adalah 5.5 dan 2.6 pada website 43.255.184.109/, berikut skor *Likelihood* dan *Impact*, seperti pada tabel berikut

Tabel 10. Hasil Total Threat Agent Factors

https://elearning.uib.ac.id/					
OWASP Risk Rating	Tingkat Keterampilan	Motif	Peluang	Ukuran	*Total*
Threat Agent Factor	6.6	4.7	5.9	6	5.8

Tabel 11. Hasil Total Vulnerability Factors

https://elearning.uib.ac.id/					
OWASP Risk Rating	Kemudahan Penemuan	Kemudahan Eksploitasi	Kesadaran	Deteksi Intrusi	*Total*
Vulnerability Factor	9	3.4	7.2	1	5.1

Tabel 12. Hasil Likelihood

https://elearning.uib.ac.id/			
OWASP Risk Rating	*Threat Agent Factor*	*Vulnerability Factor*	*Total*
Likelihood	5.8	5.1	5.45

Tabel 13. Hasil Total Technical Impact

https://elearning.uib.ac.id/					
OWASP Risk Rating	Kehilangan Kerahasiaan	Kehilangan Integritas	Kehilangan Ketersediaan	Kehilangan Akuntabilitas	*Total*
Technical Impact	3.4	1.7	1	6.6	3.2

Tabel 14. Hasil *Total Business Impact*

https://elearning.uib.ac.id/					
OWASP Risk Rating	Kerusakan Finansial	Kerusakan Reputasi	Ketidakpatuhan	Pelanggaran Privasi	*Total*
Business Impact	1.6	1.4	2	3	2

Tabel 15. Hasil *Impact*

https://elearning.uib.ac.id/			
OWASP Risk Rating	*Technical Impact*	*Business Impact*	*Total*
Impact	3.2	2	2.6

Untuk bisa mengetahui *website* yang telah kita analisa tersebut termasuk *severity risk* yang mana maka kita memerlukan tabel16 untuk lebih mudah mengetahui keamanan dari *website* tersebut.

Tabel 16. *Likelihood and Impact Levels*

Likelihood and Impact Levels	
0 to <3	Low
3 to <6	Medium
6 to 9	High

Bisa kita liat dari *likelihood* itu bisa kita ketahui bahwa domain *website* ini termasuk dalam *Low severity risk* sedangkan dari *impact* itu bisa kita ketahui bahwa domain *website* ini termasuk dalam *Medium severity risk*. Sehingga dari itu kita memberikan tiap tiap jenis ancaman tersebut solusi masing-masingnya yang bisa di liat pada tabel 17.

Tabel 17. *Likelihood and Impact Levels*

https://elearning.uib.ac.id/	
Jenis Ancaman	Solusi
Cross Site Scripting (Reflected)	Gunakan perpustakaan atau kerangka kerja yang diperiksa yang tidak memungkinkan kelemahan ini terjadi atau menyediakan konstruksi yang membuat kelemahan ini lebih mudah untuk dihindari. Contoh pustaka dan kerangka kerja yang memudahkan untuk menghasilkan keluaran yang disandikan dengan benar termasuk pustaka Anti-XSS Microsoft, modul Pengodean ESAPI OWASP, dan Apache Wicket.
SQL Injection	Jangan percaya input sisi klien, bahkan jika ada validasi sisi klien. Secara umum, ketik centang semua data di sisi server. Jika aplikasi menggunakan JDBC, gunakan PreparedStatement atau CallableStatement, dengan parameter yang diteruskan dengan "?' Jika aplikasi menggunakan ASP, gunakan Objek Perintah ADO dengan pemeriksaan tipe yang kuat dan kueri berparameter. Jika Prosedur Tersimpan database dapat digunakan, gunakanlah. Jangan *jangan* gabungkan string ke dalam kueri dalam prosedur tersimpan, atau gunakan 'exec', 'exec segera', atau fungsionalitas yang setara! Jangan membuat kueri SQL dinamis menggunakan rangkaian string sederhana. Melarikan diri dari semua data yang diterima dari klien. Terapkan 'daftar yang diizinkan' untuk karakter yang diizinkan, atau 'daftar yang ditolak' untuk karakter yang tidak diizinkan dalam input pengguna. Terapkan prinsip hak istimewa paling rendah dengan menggunakan pengguna basis data yang paling tidak memiliki hak istimewa. Secara khusus, hindari menggunakan pengguna database 'sa' atau 'db-owner'. Ini tidak menghilangkan injeksi SQL, tetapi meminimalkan dampaknya. Berikan akses basis data minimum yang diperlukan untuk aplikasi.
.htaccess Information Leak	Pastikan file .htaccess tidak dapat diakses.
Directory Browsing	Nonaktifkan penjelajahan direktori. Jika ini diperlukan, pastikan file yang terdaftar tidak menimbulkan risiko.
Vulnerable JS Library	Harap tingkatkan ke versi jquery terbaru.
X-Frame-Options Header Not Set	Sebagian besar browser Web modern mendukung header HTTP X-Frame-Options. Pastikan itu disetel di semua halaman web yang dikembalikan oleh situs Anda (jika Anda mengharapkan halaman dibingkai hanya oleh halaman di server Anda (misalnya itu bagian dari FRAMESET) maka Anda akan ingin menggunakan SAMAORIGIN, jika tidak, jika Anda tidak pernah mengharapkan halaman untuk dibingkai, Anda harus menggunakan DENY. Atau pertimbangkan untuk menerapkan arahan "frame-ancestors" Kebijakan Keamanan Konten.
Absence of Anti-CSRF Tokens	Gunakan perpustakaan atau kerangka kerja yang diperiksa yang tidak memungkinkan kelemahan ini terjadi atau menyediakan konstruksi yang membuat kelemahan ini lebih mudah untuk dihindari. Misalnya, gunakan paket anti CSRF seperti OWASP CSRFGuard.
Cookie No HttpOnly Flag	Pastikan bahwa flag HttpOnly disetel untuk semua cookie.
Cookie without SameSite Attribute	Pastikan atribut SameSite diatur ke 'lax' atau idealnya 'strict' untuk semua cookie.
Incomplete or No Cache-control Header Set	Jika memungkinkan, pastikan header HTTP kontrol-cache disetel dengan no-cache, no-store, must-revalidate.
Timestamp Disclosure - Unix	Konfirmasikan secara manual bahwa data stempel waktu tidak sensitif, dan bahwa data tidak dapat digabungkan untuk mengungkapkan pola yang dapat dieksploitasi.
X-Content-Type-Options Header Missing	Pastikan bahwa aplikasi/server web menyetel header Content-Type dengan tepat, dan menyetel header X-Content-Type-Options ke 'nosniff' untuk semua halaman web. Jika memungkinkan, pastikan bahwa pengguna akhir menggunakan browser web yang sesuai standar dan modern yang tidak melakukan sniffing MIME sama sekali, atau yang dapat diarahkan oleh aplikasi web/server web untuk tidak melakukan sniffing MIME.
Information Disclosure - Sensitive Information in URL	Jangan berikan informasi sensitif dalam URI.
Information Disclosure - Suspicious Comments	Hapus semua komentar yang mengembalikan informasi yang dapat membantu penyerang dan perbaiki masalah mendasar yang mereka rujuk.

4. SIMPULAN DAN SARAN

Dalam melakukan pengujian sistem keamanan *web* menggunakan metode OWASP pada 43.255.184.109 berdasarkan dari seluruh kegiatan yang dilakukan diatas maka penulis memberikan kesimpulan bahwa dari hasil penelitiannya kita bisa mengetahui bahwa *website* UIB memiliki tingkat risiko yang rendah namun demikian, tetap harus melakukan perbaikan terhadap ancaman yang muncul untuk mencegah datangnya serangan serangan yang lebih membahayakan *website*.

Saran yang dapat diberikan dari penelitian adalah untuk perlunya dilakukan pengujian pada seluruh sistem yang memiliki domain uib.ac.id agar segera dapat ditanggulangi jika terdapat celah keamanan, melakukan konfigurasi kembali pada server uib.ac.id agar hanya sebagian orang tertentu saja yang dapat melakukan *request* terhadap data yang *sensitive* dan perlu dilakukan *update* secara berkala terhadap beberapa *plugin* yang terdapat dalam *web* yang dikelola.

5. DAFTAR PUSTAKA

Dewanto, A. P. (2018). *Penetration Testing padaDomain uii.ac.id Menggunakan OWASP 10*.

Eka Pratama, I. P. A., & Wiradarma, A. A. B. A. (2019). Open Source Intelligence Testing Using the OWASP Version 4 Framework at the Information Gathering Stage (Case Study: X Company). *International Journal of Computer Network and Information Security*, *11*(7), 8–12. https://doi.org/10.5815/ijcnis.2019.07.02

Ghozali, B., Kusrini, & Sudarmawan. (2018). *Mendeteksi Kerentanan Keamanan Aplikasi Website Menggunakan Metode Owasp (Open Web Application Security Project) untuk Penilaian Risk Rating Detect Web Application Security Flaws Using the Owasp (Open Web Application Security Project) Method for Risk Asse*. *4*, 12.

Haeruddin. (2019). Mapping Information Asset Profile In The Implementation Of Risk Management Information System Using Octave Allergo. *Journal of Information Technology Education: Research*, *3*(1), 67–75. https://doi.org/10.31289/JITE.V3I1.2601

Kurniawan, A. (2019). Penerapan Framework OWASP dan Network Forensics untuk Analisis, Deteksi, dan Pencegahan Serangan Injeksi di Sisi Host-Based. *Jurnal Telematika*, *14*(1).

Li, J. (2020). Vulnerabilities mapping based on OWASP-SANS: A survey for static application security testing (SAST). *Annals of Emerging Technologies in Computing*, *4*(3), 1–8. https://doi.org/10.33166/AETiC.2020.03.001

Muttaqien, R. (2019). Rancang Bangun Aplikasi Mobile Untuk Peminjaman Barang Menggunakan Layanan Web (Studi Kasus: Kantor Bpn Kota Langsa). *Jurnal Karya Ilmiah Teknik Elektro*, *4*(4), 1–9.

Qian, K., Parizi, R. M., & Lo, D. (2018). *OWASP Risk Analysis Driven Security Requirements Specification for Secure Android Mobile Software Development*.

Widhiyanto, A. C. (2019). *Rancang Bangun Web Server Berbasis Jaringan Cisco Catalyst Series 2960 Di Pt . Telekomunikasi Indonesia Divre V Jatim*. 1–41.

Wismarini, T. D., & Prihandono, A. (2020). Rancang Bangun Aplikasi Android Terintegrasi Web Service Dengan Volley Untuk Layanan Publik. *Dinamik*, *25*(1), 10–19. https://doi.org/10.35315/dinamik.v25i1.7515