Penetration Testing Web XYZ Berdasarkan OWASP Risk Rating

Dimas Febriyan Priambodo [1*], Asep Dadan Rifansyah [2], Muhammad Hasbi [3]

[1,2] Program Studi Rekayasa Keamanan Siber, Politeknik Siber dan Sandi Negara, Depok, Jawa Barat
[3] Program Studi Informatika, STMIK Sinar Nusantara, Surakarta, Jawa Tengah
Email: [1*] dimas.febriyan@poltekssn.ac.id, [2] asep.dadan@bssn.go.id, [3] m.hasbi@sinus.ac.id

(Naskah masuk: 7 Nov 2022, direvisi: 16 Feb 2023, diterima: 20 Feb 2023)

Abstrak

Website "XYZ" merupakan aplikasi yang mempunyai fungsi dalam layanan pembuatan dokumen kependudukan, layanan pendaftaran akses masuk, dan fitur *login*. Penilaian kerawanan secara berkala diperlukan untuk menjamin kehandalan dari aplikasi. Penilaian kerawanan dengan menggunakan *tool* uji saja sekarang tidak dirasa cukup sehingga memerlukan validasi. Salah satu validasi tersebut adalah menggunakan *penetration testing*. Uji penetrasi pada *Website* XYZ Kabupaten XYZ dilaksanakan dengan mengacu kepada *Open Web Application Security Project* (OWASP) *Top* 10-2021. *Penetration testing* dilaksanakan dengan metode *black box* untuk mendapatkan hasil pengukuran tingkat kerentanan pada aplikasi. Keseluruhan penilaian kerentanan dilakukan dalam empat tahap yaitu *planning*, *information gathering*, *vulnerability scanning* menggunakan 2 *tool*s otomatis yaitu Vega dan OWASP ZAP sebagai upaya untuk mendapatkan cakupan yang lebih luas terkait kerentanan yang ditemukan dikuti dengan validasi dilanjutkan tahap *analysis and reporting*. Hasil tahap *vulnerability scanning* menghasilkan 9 jenis kerentanan dengan sebaran 2 *high*, 1 *medium*, dan 6 *low*. Pengujian penetrasi untuk validasi mengacu pada dokumen panduan *Web Security Testing Guide* (WSTG) versi 4.2. Hasil proses akhir berupa rekomendasi dapat digunakan sebagai referensi pengembang aplikasi *web* untuk menangani kerentanan khususnya hilangnya ketersediaan layanan dan kebocoran data.

Kata Kunci: OWASP *Top 10*, Penilaian Kerentanan, Uji Penetrasi, *Web-app*, WSTG.

XYZ Web Penetration Testing Based on OWASP Risk Rating

Abstract

The "XYZ" Website is an application that has functions in the service of making population documents, registration services for entry access, and login features. Periodic vulnerability assessments are required to ensure the reliability of the application. Vulnerability assessment by using the test tool alone is not enough now, so it requires validation. One of these validations is using penetration testing. The penetration test on the XYZ Website in XYZ Regency is carried out with reference to the Open Web Application Security Project (OWASP) Top 10-2021. The penetration testing strategy uses the black box testing method to get the results of measuring the level of vulnerability in the application with four stages, namely planning, information gathering, vulnerability scanning using 2 automated tools, namely Vega and OWASP ZAP as an effort to get a wider coverage of the vulnerabilities found followed by validation followed by analysis and reporting stages. The results of the vulnerability scanning stage resulted in 9 types of vulnerabilities with a distribution of 2 high, 1 medium, and 6 low. Penetration testing for validation refers to the Web Security Testing Guide (WSTG) version 4.2 of the guidance document. The results of the final process in the form of recommendations can be used as a reference for web application developers to deal with vulnerabilities, especially loss of service availability and data leaks.

Keywords: *OWASP Top 10, Penetration testing, Vulnerability Assesment, Web-apps, WSTG.*

I. PENDAHULUAN

Keamanan sistem informasi menjadi suatu hal yang sangat penting. Menurut data laporan Honeynet BSSN tahun 2019, terdapat 98 juta serangan siber ke Indonesia menggunakan berbagai macam teknik serangan siber, oleh karena itu serangan siber saat ini menjadi suatu serangan yang sangat masif di Indonesia [1].

Menurut laporan Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) menunjukan bahwa jumlah pengguna internet di Indonesia tahun 2018 mencapai 171,2 juta orang atau 64,8% total populasi penduduk Indonesia. Jumlah ini meningkat 10,12% dari tahun 2017 [2]. Berdasarkan data tersebut, APJII memprediksi bahwa pengguna internet di Indonesia akan terus meningkat setiap tahun, dengan semakin bertambahnya pengguna layanan internet maka semakin banyak informasi termasuk melalui *website* yang dapat diperoleh dari internet [2].

Dinas Kependudukan dan Pencatatan Sipil Kabupaten XYZ merupakan salah satu Organisasi Perangkat Daerah di Kabupaten yang memiliki kewenangan sebagai Penyelenggara Administrasi Kependudukan, meliputi kegiatan pelayanan pendaftaran penduduk dan pencatatan sipil [3]. Dalam melaksanakan kegiatan administrasi kependudukan, Dinas Kependudukan dan Pencatatan Sipil Kabupaten XYZ telah memanfaatkan perkembangan teknologi informasi dan komunikasi (TIK) untuk mempermudah segala pekerjaan manusia khususnya dalam bidang informasi dan komunikasi, seperti penyebaran informasi melalui media internet yang telah membentuk suatu ruang siber [4].

Dengan meningkatnya pemanfaatan internet tersebut, juga meningkatkan ancaman dalam bentuk pencurian informasi dan data yang bersifat rahasia ditujukan untuk menyerang individu, instansi pemerintah dan militer yang dapat mengancam pertahanan suatu negara [5]. Berdasarkan laporan tahunan Pusopskamsinas BSSN tahun 2019, terdapat total serangan siber sebanyak 290,3 juta serangan siber yang masuk ke Indonesia yang tercatat jumlah serangan siber ke *web server* sebesar 12,6 juta yang meliputi *phishing*, infeksi *malware*, dan *web defacement* [6]. Insiden siber yang tercatat pada laporan tahunan Pusopskamsinas BSSN tahun 2019 masih banyak terjadi karena instansi pemerintah banyak yang tidak mengimplementasikan *Web Application Firewall* (WAF), tidak ada pembatasan akses pada *login* administrator, penggunaan *password* yang relatif lemah, kurangnya pengawasan terhadap akun *website* dan *access log* serta menggunakan aplikasi atau *framework* yang *out of update* [7]. Salah satu bentuk upaya dalaam hal tersebut yakni dengan melakukan penilaian kerentanan (*vulnerability assessment*) [8] terhadap situs atau aplikasi yang menyimpan informasi atau data yang bersifat sensitif atau rahasia.

Pengujian berkala secara terstruktur diperlukan untuk validitas pengukuran dan penilaian. OWASP *Top* 10 telah digunakan beberapa pengujian dan terbukti dapat mempercepat dalam melakukan kategorisasi [9], [10]. Validasi dalam penentuan kerentanan juga diperlukan sehingga digunakanlah *penetration testing* sehingga mengurangi nilai *false* positif dari hasil *vulnerability scanning* sebelumnya.

II. TINJAUAN PUSTAKA

A. *Web* XYZ

Merupakan aplikasi berbasis *website* yang disediakan untuk layanan kependudukan dan catatan sipil secara elektronik di Kabupaten XYZ [11]. Layanan atau fitur yang disediakan oleh *web* "XYZ" terbagi menjadi 3, yakni layanan pembuatan dokumen kependudukan, layanan pendaftaran akses masuk dan fitur *login*. Untuk layanan pembuatan dokumen kependudukan secara *online* di ataranya [11] :

1. Akta Kelahiran bagi WNI meliputi pembuatan akta kelahiran bagi WNI baru, perbaikan data akta kelahiran bagi WNI, dan kerusakan/kehilangan dokumen akta kelahiran bagi WNI.
2. KTP Elektronik meliputi pembuatan KTP elektronik baru, perbaikan data KTP elektronik, dan kerusakan/kehilangan dokumen KTP elektronik.
3. Kartu Keluarga meliputi pembuatan kartu keluarga baru, perbaikan data kartu keluarga, dan kerusakan/kehilangan dokumen kartu keluarga.
4. Pindah Keluar dari Kab. XYZ meliputi pembuatan pindah keluar dari Kab. XYZ baru, perbaikan data pindah keluar dari Kab. XYZ, dan kerusakan/kehilangan dokumen pindah keluar dari Kab. XYZ.
5. Kedatangan dari Luar Kab. XYZ meliputi pembuatan kedatangan dari luar Kab. XYZ baru, perbaikan data kedatangan dari luar Kab. XYZ, dan kerusakan/kehilangan dokumen kedatangan dari luar Kab. XYZ.
6. Balai Konsul yang digunakan untuk bantuan layanan konsolidasi data kependudukan yang belum dapat diakses oleh lembaga pengguna (seperti Bank, BPJS dan sebagainya).

B. Penilaian Kerentanan

Penilaian kerentanan dapat dilakukan dengan cara melakukan pemindaian sistem dengan tujuan untuk mengetahui di dalam sistem tersebut terdapat celah keamanan [12]. Penilaian kerentanan dilakukan dengan mengacu kepada daftar kerentanan yang tersedia di dalam OWASP *Top* 10 sehingga, penilaian kerentanan ini akan membantu praktisi TI terkait masalah keamanan pada aplikasi *web* karena hasil penilaian akan menggambarkan celah keamanan terkini [13]. Dalam melakukan penilaian kerentanan, digunakan beberapa *tool*s untuk melakukan penemuan, pengujian, analisis, dan pelaporan sistem serta kerentanan [14]. Tahapan penilaian kerentanan dikelompokkan menjadi 4 tahapan dijelaskan pada Gambar 1.

Gambar 1. Metodologi *Vulnerability Assessment* [8]

C. *Open Web Application Security Project* (OWASP) *Top* 10-2021.

Open Web Application Security Project (OWASP) *Top*-10 adalah dokumen standar yang berisi daftar kerentanan yang berfokus untuk mengidentifikasi resiko keamanan aplikasi *web* yang paling berbahaya dan yang paling banyak terjadi di organisasi [15].

1. *Broken Access Control* [A01:2021]
   Terjadi ketika pembatasan yang berkaitan dengan perizinan atau *privileges* yang dapat dilakukan oleh pengguna terautentikasi, tidak dilakukan dengan baik. Serangan ini kemudian dapat dimanfaatkan oleh penyerang untuk mendapatkan akses yang tidak sah, termasuk mengakses dan mengubah akun pengguna, data *sensitive*, data pengguna, dan sebagainya.
2. *Cryptographic Failures* [A02:2021]
   Terjadi ketika aplikasi *web* dan API tidak berhasil melindungi data sensitif seperti data pelayanan, data pribadi, informasi keuangan, dan data krusial lainnya. Data dengan tingkat keamanan rendah dapat digunakan oleh penyerang untuk melakukan pencurian identitas, penipuan, dan tindak kejahatan lainnya. Upaya melindungi data sensitif memerlukan keamanan tambahan seperti menggunakan SSL/TLS yang terverifikasi.
3. *Injection* [A03:2021]
   Terjadi apabila suatu data yang tidak terpercaya dikirimkan ke interpreter dan langsung diterjemahkan sebagai permintaan, hal ini memicu penyerang mengirimkan permintaan dengan tujuan dapat mengakses data tanpa melalui otorisasi yang sesuai dengan ketentuan.
4. *Insecure Design* [A03:2021]
   Serangan ini disebabkan oleh salah satu faktor berupa kurangnya profil risiko bisnis pada sistem yang sedang dikembangkan menyebabkan kegagalan untuk menentukan tingkat desain keamanan yang dibutuhkan.
5. *Security Misconfiguration* [A05:2021]
   Serangan ini dapat terjadi sebagai akibat dari tidak melakukan konfigurasi atau hanya sekedar melakukan konfigurasi secara default, tidak melakukan peningkatan sistem, dependensi, kerangka kerja dan komponen secara berkala.
6. *Vulnerable and Outdated Components* [A06:2021]
   Serangan ini dapat terjadi sebagai akibat karena masih menggunakan komponen yang sudah diketahui rentan dieksploitasi, kerentanan ini dimanfaatkan oleh penyerang untuk mengambil data sensitif bahkan pengambilan *server*.
7. *Identification and Authentication Failures* [A07:2021]
   Kerentanan dapat terjadi ketika fungsi dari sistem aplikasi bagian manajemen sesi tidak bekerja sebagaimana mestinya, hal ini menyebabkan penyerang melakukan manipulasi pada kunci atau *token session* bahkan manipulasi pada *password* sehingga digunakan oleh penyerang untuk mendapatkan akses ke dalam sistem.
8. *Software and Data Integrity Failures* [A08:2021]
   Kerentanan ini terjadi dikarenakan ada kode dan infrastruktur yang tidak melindungi dari integritas, seperti aplikasi bergantung pada *plugin*, *libraries*, atau modul dari sumber tidak terpercaya, dari *Control Deliver Network* (CDN).
9. *Security Logging and Monitoring Failures* [A09:2021]
   Serangan akan dilakukan oleh penyerang apabila integrasi untuk respon insiden tidak berjalan baik, pemantauan dan pencatatan yang tidak memadai menjadikan penyerang untuk menyerang sistem lebih lanjut, merusak, mengekstrak, atau menghancurkan data.
10. *Server Side Request Forgery* [A10:2021]
    Serangan ini terjadi saat aplikasi *web* menggunakan *remote resource* tanpa melakukan validasi URL yang disediakan. Serangan ini memicu penyerang menggunakan aplikasi untuk mengirimkan permintaan yang dibuat ke tujuan lain dengan dilindungi oleh *firewall*, *Virtual Private Network* (VPN) atau jenis lain dari daftar *Access Control List* (ACL).

D. OWASP *Web Security Testing Guide Version* 4.2

Sebuah dokumen panduan dalam melakukan pengujian keamanan pada aplikasi *web* yang menghasilkan sumber daya pengujian keamanan siber untuk pengembang aplikasi *web* [15]. Pengujian dilakukan dengan pendekatan *Black box* testing atau kondisi dimana penguji memiliki beberapa informasi penting dan sebagian akses ke dalam sumber daya terhadap target pengujian. OWASP *Web Security Testing Guide* (WSTG) membagi pengujian menjadi dua kategori, yaitu pengujian secara pasif dan pengujian aktif. Pengujian pasif dilakukan dengan mencoba memahami logika aplikasi dan mempelajari aplikasi seolah-olah penguji adalah pengguna. Teknik pengujian aktif dalam OWASP *Web Security Testing Guide* (WSTG) versi 4.2 dapat dilihat pada Tabel 1 [15].

Tabel 1. OWASP *Web Security Testing Guide 4.2*

1	*Information Gathering*
	a. Conduct Search Engine Discovery Reconnaissance for InformationLeakage (WSTG-INFO-01)
	b. Fingerprint Web Server (WSTG-INFO-02)
	c. Review Webserver Metafiles Information Leakage (WSTG-INFO-03)
	d. Enumerate Applications on Webserver (WSTG-INFO-04)
	e. Review Webpage Content for Information Leakage (WSTG-INFO-05)

	f. Identify Application Entry Points (WSTG-INFO-06)
	g. Map Execution Paths Through Application (WSTG-INFO-07)
	h. Fingerprint Web Application Framework (WSTG-INFO-08)
	i. Fingerprint Web Application (WSTG-INFO-09)
	j. *Map Application Architecture (WSTG-INFO-010)*
2	*Configuration and Deployment Management Testing*
	a. Test Network Infrastructure Configuration (WSTG-CONF-01)
	b. Test Application Platform Configuration (WSTG-CONF-02)
	c. Test File Extensions Handling Sensitive Information (WSTG-CONF-03)
	d. Review Old Backup and Unreferenced Files for Sensitive Information(WSTG
	e. CONF-04)
	f. Enumerate Infrastructure and Application Admin Interfaces (WSTG-CONF-05)
	g. Test HTTP Methods (WSTG-CONF-06)
	h. Test HTTP Strict Transport Security (WSTG-CONF-07)
	i. Test RIA Cross Domain Policy (WSTG-CONF-08)
	j. Test File Permission (WSTG-CONF-09)
	k. Test for Subdomain Takeover (WSTG-CONF-010)
	l. *Test Cloud Storage (WSTG-CONF-011)*
3	*Identity Management Testing*
	a. Test Role Definitions (WSTG-IDNT-01)
	b. Test User Registration Process (WSTG-IDNT-02)
	c. Test Account Provisioning Process (WSTG-IDNT-03)
	d. Testing for Account Enumeration and Guessable User Account (WSTG-IDNT-04)
	e. Testing for Weak or Unenforced Username Policy (WSTG-IDNT-05)
4	*Authentication Testing*
	a. Testing for Credentials Transported over an Encrypted Channel (WSTG-ATHN-01)
	b. Testing for Default Credentials (WSTG-ATHN-01)
	c. Testing for Weak Lock Out Mechanism (WSTG-ATHN-01)
	d. Testing for Bypassing Authentication Schema (WSTG-ATHN-01)
	e. Testing for Credentials Transported over an Encrypted Channel (WSTG-ATHN-01)
	f. Testing for Default Credentials (WSTG-ATHN-01)
	g. Testing for Weak Lock Out Mechanism (WSTG-ATHN-01)
	h. Testing for Bypassing Authentication Schema (WSTG-ATHN-01)
	i. Testing for Vulnerable Remember Password (WSTG-ATHN-01)
	j. Testing for Browser Cache Weaknesses (WSTG-ATHN-01)
	k. Testing for Weak Password Policy (WSTG-ATHN-01)
	l. Testing for Weak Security Question Answer (WSTG-ATHN-01)
	m. Testing for Weak Password Change or Reset Functionalities (WSTG-ATHN-01)
	n. Testing for Weaker Authentication in Alternative Channel (WSTG-ATHN-01)
5	*Authorization Testing*
	a. Testing Directory Traversal File Include (WSTG-ATHZ-01)
	b. Testing for Bypassing Authorization Schema Include (WSTG-ATHZ-02)
	c. Testing for Privilege Escalation Include (WSTG-ATHZ-03)
	d. Testing Insecure Direct Object References Include (WSTG-ATHZ-04)
6	*Session Management Testing*
	a. Testing for Session Management Schema (WSTG-SESS-01)
	b. Testing for Cookies Attributes (WSTG-SESS-02)
	c. Testing for Session Fixation (WSTG-SESS-03)
	d. Testing for Exposed Session Variables (WSTG-SESS-04)
	e. Testing for Cross Site Request Forgery (WSTG-SESS-05)
	f. Testing for Logout Functionality (WSTG-SESS-06)
	g. Testing Session Timeout (WSTG-SESS-07)
	h. Testing for Session Puzzling (WSTG-SESS-08)
	i. Testing for Session Hijacking (WSTG-SESS-09)
7	*Input Validation Testing*
	a. Testing for Reflected Cross Site Scripting (WSTG-INPV-01)
	b. Testing for Stored Cross Site Scripting (WSTG-INPV-02)
	c. Testing for HTTP Verb Tampering (WSTG-INPV-03)
	d. Testing for HTTP Parameter Pollution (WSTG-INPV-04)
	e. Testing for SQL Injection (WSTG-INPV-05)
8	*Error Handling*
	a. Testing for Improper Error Handling (WSTG-ERRH-01)
	b. Testing for Stack Traces (WSTG-ERRH-02)
9	*Cryptography*
	a. Testing for Weak Transport Layer Security (WSTG-CRYP-01)
	b. Testing for Padding Oracle (WSTG-CRYP-02)
	c. Testing for Sensitive Information Sent via Unencrypted Channels (WSTG-CRYP-03)
	d. Testing for Weak Encryption (WSTG-CRYP-04)
10	*Business Logic Testing*
	a. Test Business Logic Data Validation (WSTG-BUSL-01)
	b. Test Ability to Forge Requests (WSTG-BUSL-02)
	c. Test Integrity Checks (WSTG-BUSL-03)
	d. Test for Process Timing (WSTG-BUSL-04)
	e. Test Number of Times a Function Used Limits (WSTG-BUSL-05)
	f. Testing for the Circumvention of Work Flows (WSTG-BUSL-06)
	g. Test Defenses Against Application Misuse (WSTG-BUSL-07)
	h. Test Upload of Unexpected File Types (WSTG-BUSL-08)
	i. *Test Upload of Malicious Files (WSTG-BUSL-09)*
11	*Client-side Testing*
	a. Testing for DOM-Based Cross Site Scripting (WSTG-CLNT-01)
	b. Testing for JavaScript Execution (WSTG-CLNT-02)
	c. Testing for HTML Injection (WSTG-CLNT-03)
	d. Testing for Client-side URL Redirect (WSTG-CLNT-04)
	e. Testing for CSS Injection (WSTG-CLNT-05)
	f. Testing for Client-side Resource Manipulation (WSTG-CLNT-06)
	g. Testing Cross Origin Resource Sharing (WSTG-CLNT-07)
	h. Testing for Cross Site Flashing (WSTG-CLNT-08)
	i. Testing for Clickjacking (WSTG-CLNT-09)
	j. Testing Web Sockets (WSTG-CLNT-10)
	k. Testing Web Messaging (WSTG-CLNT-11)
	l. Testing Browser Storage (WSTG-CLNT-12)
	m. Testing for Cross Site Script Inclusion (WSTG-CLNT-13)
12	*API Testing*
	a. *Testing GraphQL* (WSTG-APIT-01)

E. OWASP *Risk Rating*

Penentuan tingkat risiko kerentanan dapat dilakukan dengan metodologi OWASP *Risk Rating* [16]. Tingkat risiko atau dampak yang disebabkan oleh suatu kerentanan ditentukan berdasarkan 3 faktor yakni *exploitability* (vektor serangan), *weakness prevalence* (keberadaan kelemahan), *weakness detectability* (deteksi kelemahan), *technical impacts* (dampak teknikal), dan *business impacts* (dampak bisnis) seperti tergambar pada Tabel 2. Agen Ancaman atau *Threat Agents* hanya diketahui oleh lingkup penyerang dan dampak bisnis atau *business impact* hanya pihak pemilik aplikasi yang dapat mengetahui secara khusus dari dampak yang ditimbulkan.

Tabel 2. OWASP *Web Security Testing Guide 4.2*

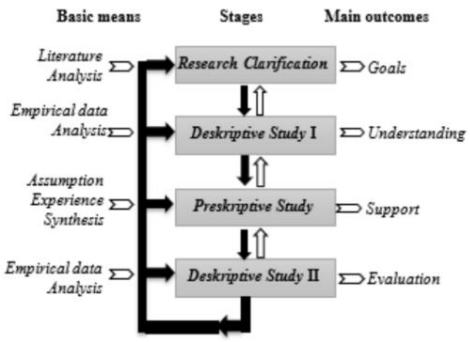
	Exploitability	Weakness prevalence	Weakness detectability	Technical impacts	Business impact
?	Mudah (3)	Tersebar Luas (3)	Mudah (3)	Parah (3)	(?))
	Sedang (2)	Biasa (2)	Sedang (2)	Cukup (2)	
	Sulit (1)	Tidak Biasa (1)	Sulit (1)	Rendah (1)	

F. Penelitian Terkait

1. *Vulnerability Assessment and Penetration testing to Enhance the Security of Web Application*.
   Penelitian dari Arvind dkk [17] melakukan penilaian kerentanan pada berbagai aplikasi Lembaga keuangan dengan menggunakan Nessus dan OWASP ZAP dengan tahapan *Planning*, *Discovery* (*information gathering*, *vulnerability scanning*), *Attack* (*vulnerability exploitation*), *Report Analysis*.
2. *Vulnerability Assessment and Penetration testing of Web Application*.
   Penelitian dari Prof. Sangeeta dkk [18] menggabungkan uji manual dan otomatis menggunakan Burpsuit, ZAP dan Accunetix untuk aplikasi *E-Commerce* dan *Cloud*.
3. *Security Assessment of Libyan Government Websites*
   Hasil karya Abdullah ahmed dkk [19] melakukan *security assessment* dengan uji penetrasi dengan standar dari Weidman menggunakan Acunetix dan Netsparker pada *Website* pemerintah Libya.
4. *Survey of Websites and Web Application Security Threats Using Vulnerability Assessment*
   Vincent Appiah dkk [8] melakukan *vulnerability assessment* dengan tahapan *Planning*, *Information Gathering*, *Vulnerability Scanning* dan *Reporting* menggunakan gabungan *tool*s antara lain Nmap, Nikto dan Nessus menggunakan mesin virtual untuk *web* 5 institusi berbeda di Ghana.

III. METODOLOGI PENELITIAN

Analisis keamanan akan dilakukan pada aplikasi berbasis *web* "XYZ" dibagi ke dalam 4 tahapan yaitu *planning*, *information gathering*, *vulnerability scanning*, *analysis and reporting*. Pengelompokan tersebut merupakan pendekatan penelitian yang dilakukan oleh Vincent Appiah et al. [8]. Pada akan dilakukan perencanaan dalam menentukan ruang lingkup pengujian yang akan dilakukan. Tahap kedua, melakukan *information gathering* dengan mengumpulkan semua informasi dan data mengenai nilai aset yang ada pada target uji. Tahap ketiga *vulnerability scanning* adalah pemindaian kerentanan dengan menggunakan *tool*s terotomasi yaitu OWASP ZAP Proxy dan Vega dan mengategorikan hasil kerentanan yang ditemukan berdasarkan OWASP *Top*-10. Tahap keempat adalah proses *exploitation* untuk melakukan validasi kerentanan yang terindentifikasi. Tahap kelima yaitu *analysis and reporting* untuk mengetahui keterkaitan dampak risiko yang ditimbulkan terhadap bisnis organisasi. Pembuatan rekomendasi terhadap kerentanan yang telah divalidasi sebagai bentuk tahapan akhir dalam penyusunan laporan. Semua tahapan tersebut diaplikasikan dalam metode penelitian *Design Research Method* (DRM) agar lebih efektif dan efisien. Gambar 2 menunjukkan hubungan antara tahap-tahap ini. Panah tebal di antara tahapan menggambarkan proses utama.



Gambar 2. DRM *Framework*

A. *Research Clarification* (RC)

Proses menemukan beberapa bukti atau setidak-tidaknya indikasi yang mendukung asumsi dalam rangka merumuskan suatu dan tujuan penelitian yang berharga. Pencarian bukti utama dilakukan dengan mencari literatur yang dapat mempengaruhi keberhasilan seperti penelitian L. T. M. Blessing and A. Chakrabarti [20].

B. *Descriptive Study I* (DS-I)

Deskripsi rinci untuk menentukan faktor mana yang harus ditangani untuk meningkatkan tugas klarifikasi seefektif dan seefisien mungkin. Kriteria yang didapat adalah:

a. Kriteria sukses
   Kondisi yang harus dipenuhi dan ditetapkan untuk tujuan. Dalam penelitian ini, kriteria sukses adalah skor kerentanan aplikasi dan rekomendasi untuk kerentanan tersebut.
b. Faktor kunci
   Faktor yang mempengaruhi faktor lain dan merupakan faktor utama untuk mencapai tujuan kriteria sukses atau proses penilaian kerentanan itu sendiri.

C. *Prescriptive Study* (PS)

Menggunakan peningkatan pemahaman tentang situasi yang ada untuk memperbaiki dan menguraikan deskripsi awal

situasi yang diinginkan. Deskripsi ini mewakili visi tentang bagaimana mengatasi satu atau lebih faktor dalam situasi yang ada akan mengarah pada realisasi dari situasi yang diinginkan dan ditingkatkan. Dikembangkan dengan berbagai kemungkinan skenario dengan memvariasikan faktor yang ditargetkan.

D. *Descriptive Study II* (DS-II)

Proses eksploitasi dilakukan pada aplikasi dilakukan penilaian kerentanan melalui proses pengujian penetrasi sebagai tahap validasi kerentanan dari hasil pemindaian kerentanan sekaligus berfungsi sebagai referensi untuk memberikan rekomendasi.

IV. PENGUJIAN DAN PENILAIAN KERENTANAN

A. *Research Clarification* (RC)

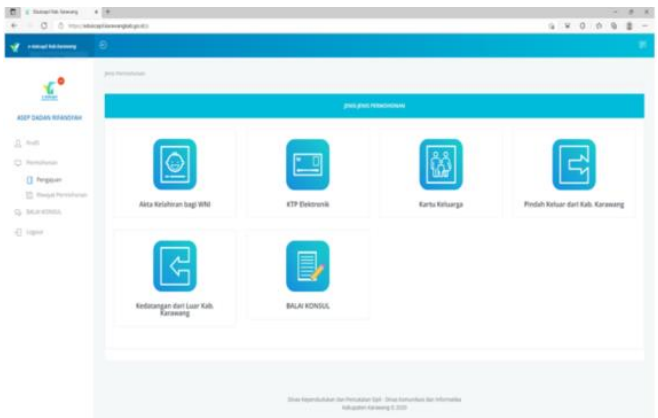
Alat dan lingkungan ditetapkan berdasarkan analisis literature dan didapatkan daftar yang tercantum dalam Tabel 3.

Tabel 3. Kebutuhan Alat dan Aplikasi

Alat / Aplikasi	Fungsi
Sistem Operasi Windows 10 Pro	Lingkungan pengujian kerentanan
Vega	Pemindai kerentanan
OWASPZAP versi 2.11.0	Pemindai kerentanan
Burp Suite Community v2022.3.7	Pemindai kerentanan
Chrome dan Firefox	Perantara target pengujian kerentanan
Clickjacker.io dan Vulnerable.live	Aplikasi pengujian kerentanan

B. *Research Clarification* (RC)

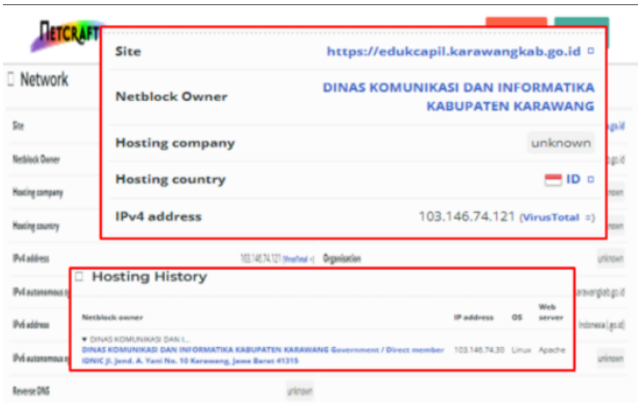
Information gathering dilakukan dalam *Research Clarification* melalui proses pengumpulan informasi aplikasi *web* "https://edukcapil.xxxxkab.go.id/". *Web* "XYZ" merupakan aplikasi berbasis *website* yang disediakan untuk layanan kependudukan dan catatan sipil secara elektronik di Kabupaten XYZ. Halaman utama *web* XYZ dapat dilihat dalam Gambar 3.



Gambar 3. Tampilan Halaman Utama *Web* XYZ

1. *Netcraft*

Netcraft sebagai *tool*s otomatisasi yang banyak dan umum digunakan berdasarkan pedoman subkategori pengujian *Fingerprint Web Server* (WSTG-INFO-02). Hasil penggunaan *Netcraft* untuk mencari informasi *Website* secara otomatis ditunjukkan pada Gambar 4.

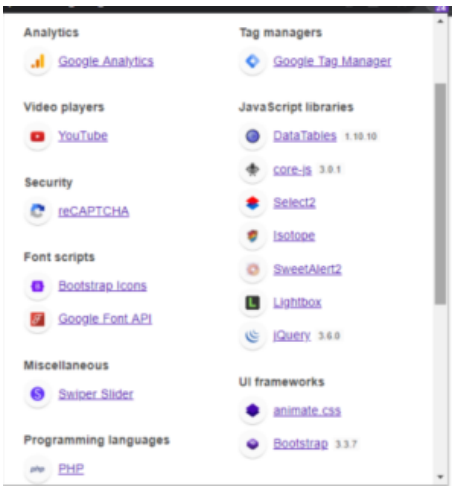


Gambar 4. Tampilan Informasi dari *Netcraft*

2. *Wappalyzer*

*Tool*s otomatisasi yang banyak dan umum digunakan berdasarkan pedoman subkategori pengujian: *Fingerprint Web Application Framework* (WSTG-INFO-08). *Wappalyzer* sebelumnya telah diinstal pada ekstensi *browser Google Chrome* sehingga ketika membuka halaman aplikasi *web* XYZ

Maka akan menampilkan beberapa informasi berupa teknologi yang digunakan oleh situs *web*, seperti yang ditunjukkan pada Gambar 5.

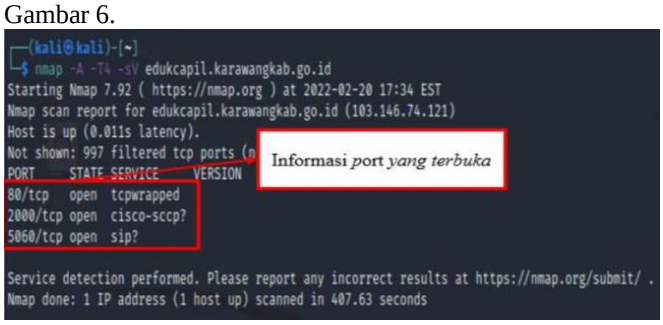


Gambar 5. Informasi Teknologi Pada Aplikasi *Web* XYZ

3. *Nmap*

*Tool*s otomatisasi yang banyak dan umum digunakan berdasarkan pedoman subkategori pengujian: *Fingerprint Web Server* (WSTG-INFO-02) dan *Enumerate Application on Webserver* (WSTG-INFO-04) dan bersifat *open source* [21]. *Nmap* dengan versi *command-line* yang dioperasikan melalui terminal sistem operasi *Kali Linux* tanpa menggunakan *Graphic User Interface* (GUI) atau dikenal dengan Zenmap [8]

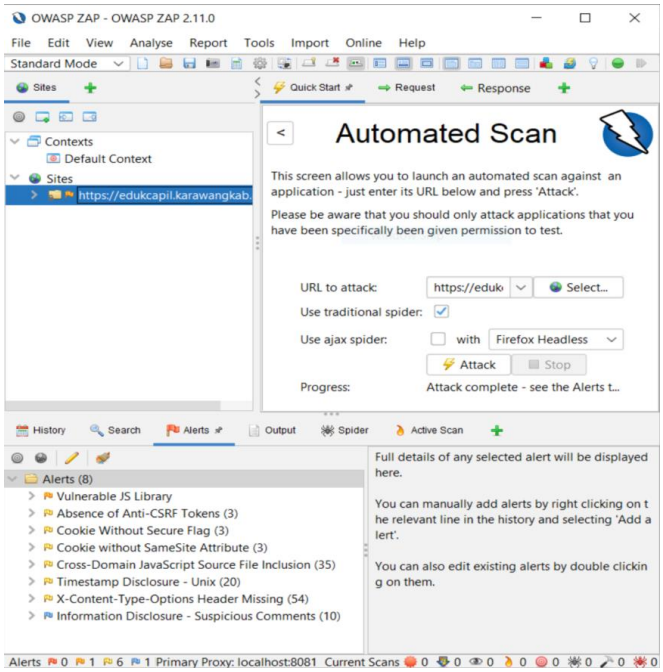
untuk memindai *port* yang tersedia seperti ditunjukkan dalam Gambar 6.



Gambar 6. *Scanning* Menggunakan Aplikasi *Nmap*

4. OWASP ZAP

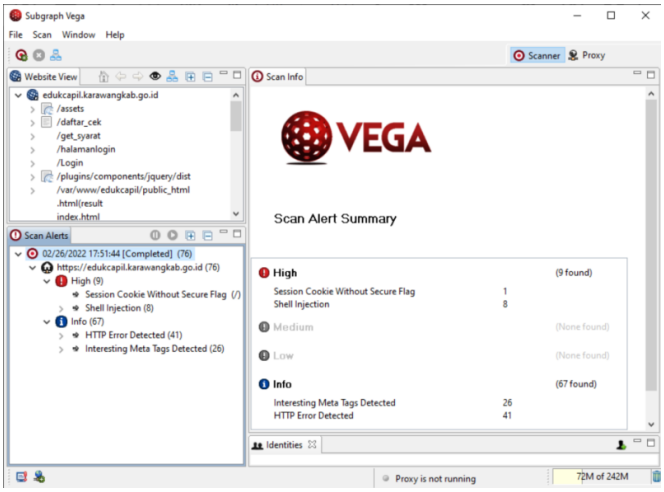
Setelah melakukan *information gathering* dengan ketiga aplikasi tersesbut maka dilakukan *vulnerability scanning* menggunakan *tool*s otomatis yaitu *Vega* dan OWASP ZAP Prox. Hasil dari pemindaian menggunakan OWASP ZAP teridentifikasi 8 jenis kerentanan yang ditemukan dengan total keseluruhan 118 kerentanan yang terdiri dari tingkat kerentanan 1 *medium*, 6 *low* dan 1 *informational*. Hal tersebut ditunjukkan pada Gambar 7.



Gambar 7. Tampilan Aplikasi OWASP ZAP

5. *Vega*

Aplikasi pemindaian kerentanan secara otomatis kedua yang digunakan peneliti yaitu *Vega*. Hasil dari pemindaian menggunakan *Vega* teridentifikasi 4 jenis kerentanan yang ditemukan dengan total keseluruhan 76 kerentanan yang terdiri dari tingkat kerentanan 2 *High* dan 2 *informational*. Hal tersebut ditunjukkan pada Gambar 8.



Gambar 8. Tampilan Aplikasi Pemindaian Kerentanan *Vega*

C. *Prescriptive Study* (PS)

Hasil pemindaian kerentanan pada aplikasi *web* XYZ yang diperoleh menggunakan 2 alat pemindaian kerentanan otomatis yaitu Vega ditunjukkan terkompilasi pada Tabel 4 dan pengklasifikasian menurut OWASP *Top* 10 pada Tabel 5.

Tabel 4. Hasil Perbandingan Pemindaian Kerentanan

Vega			OWASP ZAP	
Tingkat	**Jumlah**	**Kerentanan**	**Jumlah**	**Kerentanan**
High	2	*Session Cookie Without Secure Flag*	-	Tidak Ditemukan
Medium	-	Tidak Ditemukan	1	*Vulnerable JS Library*
Low	-	Tidak Ditemukan	6	*Cross-Domain JavaScript Source File Inclusion*
				X-Content-Type-Options Header Missing
				Absence of Anti-CSRF Tokens
				Cookie Without Secure Flag
				Cookie Without SameSite Attribute

Tabel 5. Pengkategorian kerentanan terhadap OWASP

No	Kerentanan	Kategori OWASP	Jumlah
1	• *Absence of Anti-CSRF Token* • *Cookie Without SameSite Attribute* • *Timestamp Disclousure*	*Broken Access Control [A01:2021]*	3
2	• *Shell Injection*	*Injection [A03:2021]*	1
3	• *Session Cookie Without Secure Flag* • *Cookie Without Secure Flag*	*Security Misconfiguration [A05:2021]*	3

	• *X-Content-Type-Options Header Missing*		
4	• *Vulnerable JS Library*	*Vulnerable and Outdated Components [A06:2021]*	1
5	• *Cross-Domain JavaScript Source File Inclusion*	*Software and Data Integrity Failures [A08:2021]*	1

D. *Descriptive Study II* (DS-II)

1. *Session Cookie Without Secure Flag*

Tabel 6. *Risk rating Session Cookie Without Secure Flag*

Threat Agents	*Attack Vector*	*Weakness prevalence*	*Weakness detectability*	*Technical impacts*	*Business impacts*
(?)	Mudah (3)	Tersebar Luas (3)	Mudah (3)	Cukup (2)	(?)

Tabel 6 menunjukkan letak OWASP *Risk Rating* dan pengisiannya yang selanjutnya divalidasi Berdasarkan subkategori pengujian *Testing for Cookies Attributes* (WSTG-v42-SESS-02) menggunakan *Burp Suite* untuk melihat *cookie* yang dikirimkan pada *browser proxy* yang digunakan dan meneruskan ke fitur *repeater* untuk melihat *response header* permintaan yang dikirimkan, seperti pada Gambar 9. Hasil validasi yang telah dilakukan untuk aplikasi *web* XYZ tidak terbukti rentan terhadap *Session Cookie Without Secure Flag* dikarenakan sudah menerapkan *secure flag*, dengan demikian maka hasil pengujian ***False Positive***.

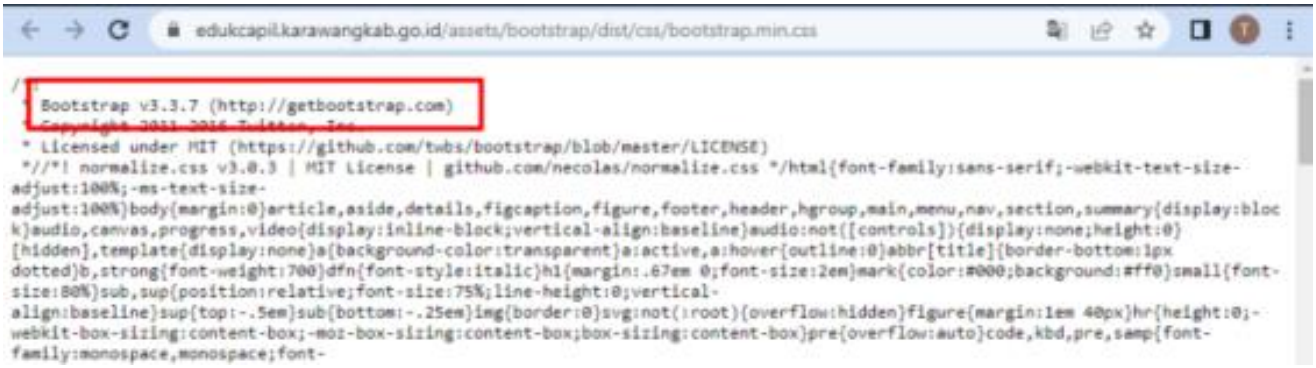
Gambar 9. *Response Header Cookie(s) Without Secure Flag*

2. *Vulnerable JS Library*

Tabel 7. OWASP *Risk Rating* Untuk *Vulnerable JS Library*

Threat Agents	*Attack Vector*	*Weakness prevalence*	*Weakness detectability*	*Technical impacts*	*Business impacts*
(?)	Mudah (3)	Tersebar Luas (3)	Mudah (3)	Cukup (2)	(?)

Tabel 7 menunjukkan kerentanan *vulnerable* JS library dalam OWASP *Risk Rating* sehingga menghasilkan nilai *medium*. Gambar 10 menunjukkan proses validasi versi dari Bootstrap dan muncul juga dalam *scaning* menggunakan OWASP ZAP yaitu *Bootstrap* versi 3.3.7. Berdasarkan *Vulnerability Details*: CVE-2019-8331 [22] untuk Skor CVSS: 4.3, Tidak ada dampak terhadap kerahasiaan sistem, modifikasi beberapa file sistem atau informasi dimungkinkan tetapi penyerang tidak memiliki kendali atas apa yang dapat dimodifikasi atau cakupan dari apa yang dapat dipengaruhi penyerang terbatas, meskipun tidak ada dampak namun jenis kerentanan *Cross-Site Scripting* (XSS), dan untuk CWE ID 79 terbukti sehingga kerentanan *Vulnerable JS Library* tergolong kedalam ***True Positive***.



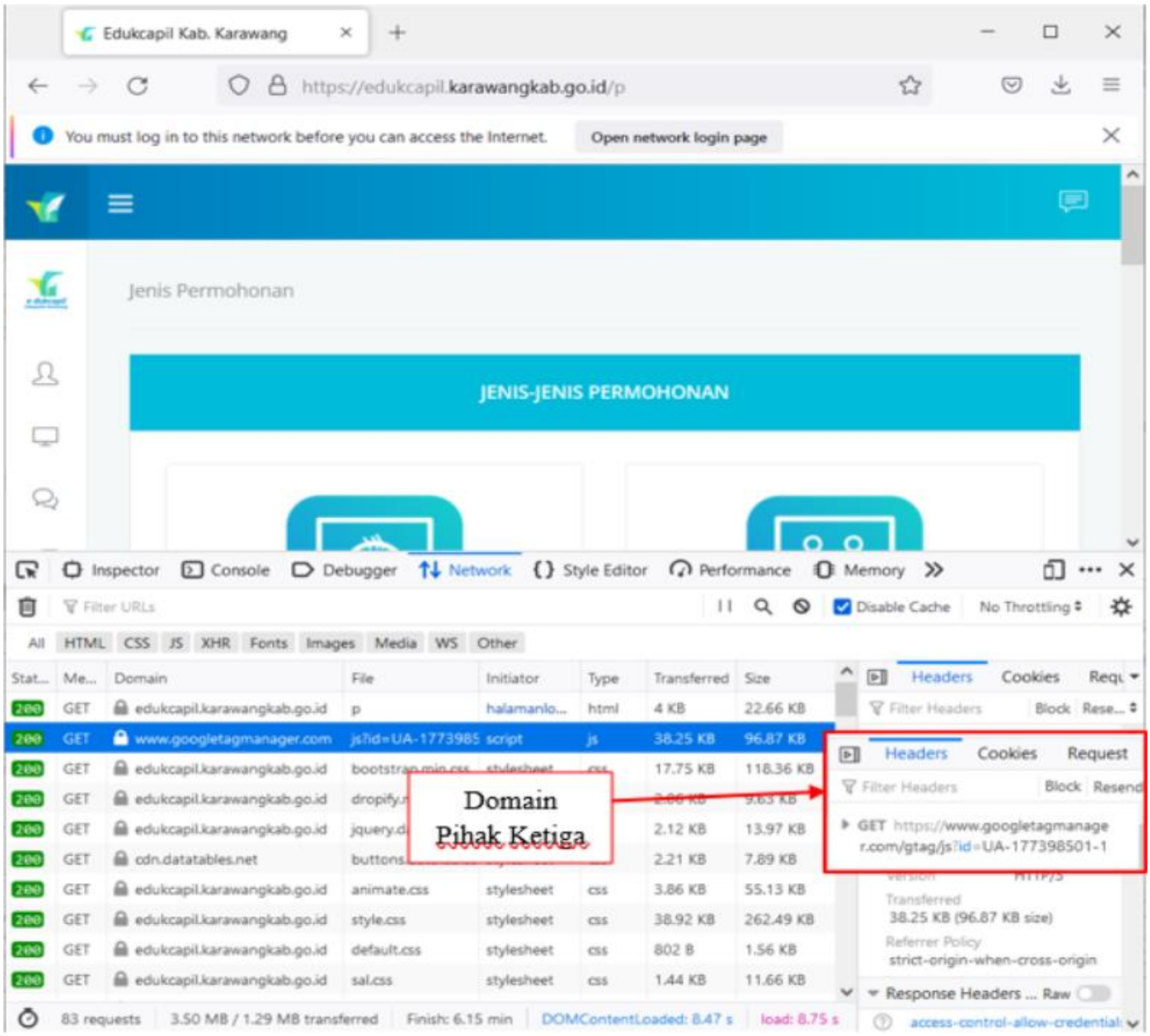
Gambar 10. Tangkapan Layar *Javascript* Halaman *Login*

3. *Cross-Domain JavaScript Source File Inclusion*

Tabel 8. OWASP *Risk Rating* untuk *Cross-Domain JavaScript Source File Inclusion*

Threat Agents	*Attack Vector*	*Weakness prevalence*	*Weakness detectability*	*Technical impacts*	*Business impacts*
(?)	Mudah (3)	Tersebar Luas (3)	Mudah (3)	Rendah (1)	(?)

Tabel 8 menunjukkan OWASP *Risk Rating* untuk *Cross-Domain JavaScript Source File Inclusion*. Validasi kerentanan merupakan tahapan yang dilakukan untuk melihat *response header* dari sumber *JavaScript* yang disediakan oleh domain pihak ketiga yang ditunjukkan pada Gambar 11 dengan fitur *inspect element*.



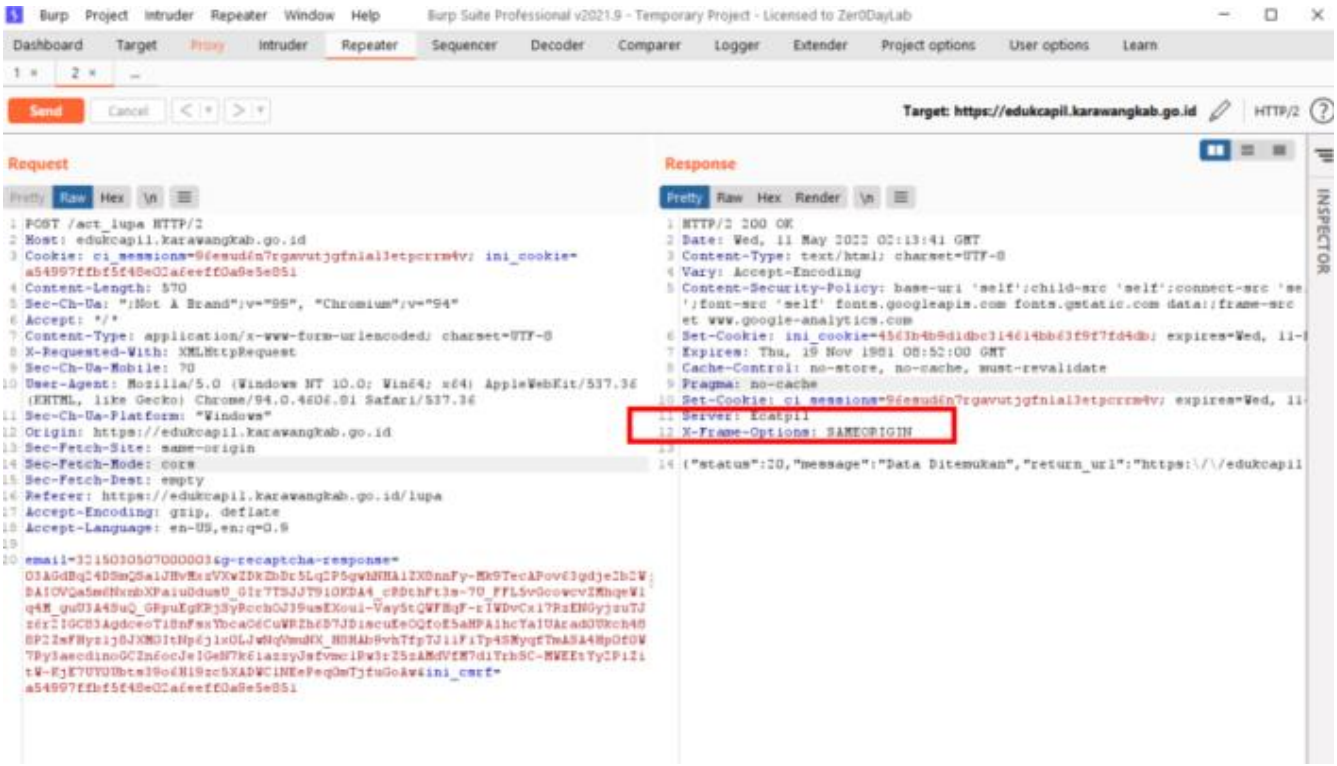
Gambar 11. Informasi *JavaScript* Dari Domain Pihak Ketiga

4. *X-Content-Type-Options Header Missing*

Tabel 9. OWASP *Risk Rating* untuk *X-Content-Type-Options Header Missing*

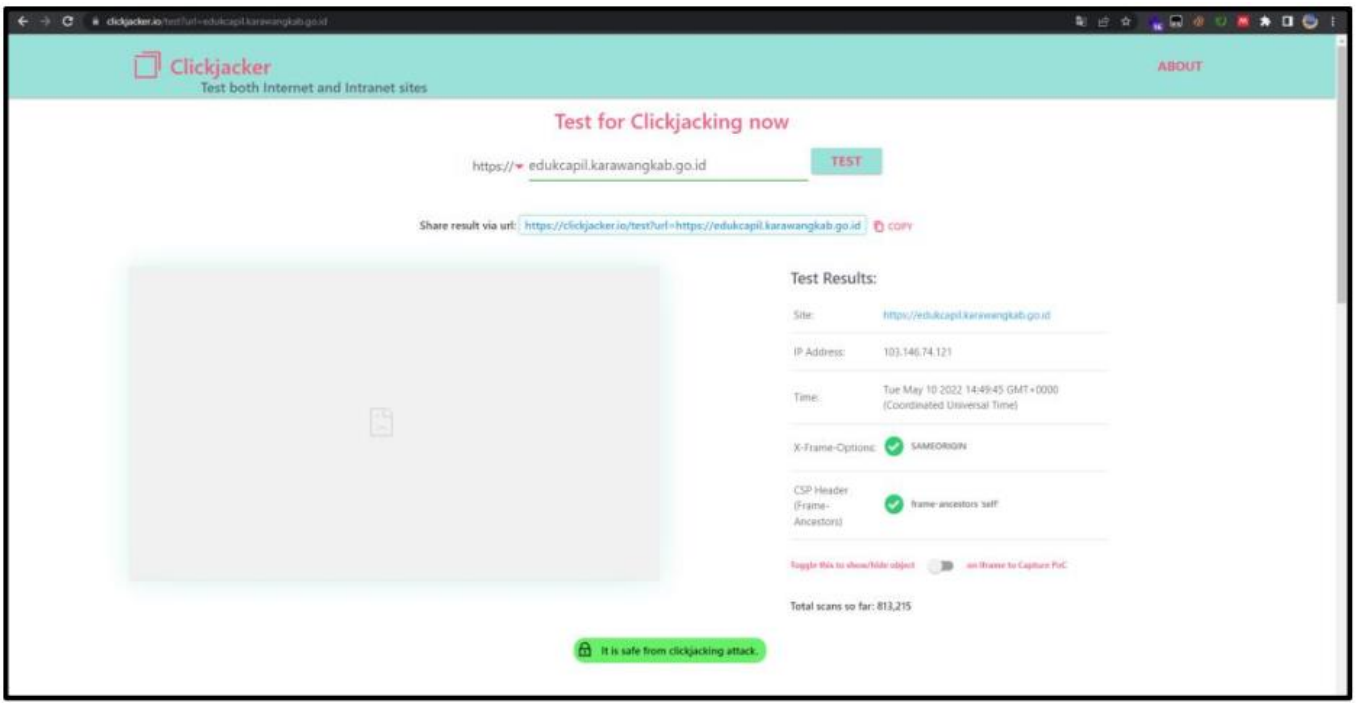
Threat Agents	*Attack Vector*	*Weakness prevalence*	*Weakness detectability*	*Technical impacts*	*Business impacts*
(?)	Sedang (2)	Tersebar Luas (3)	Mudah (3)	Parah (3)	(?)

Hasil pemindaian aplikasi OWASP ZAP kerentanan ini berada pada level *Low* dengan OWASP *Risk Rating* terlampir dalam Tabel 9. *Kerentanan X-Content-Type-Options Header Missing* adalah kerentanan dimana *server* tidak mengembalikan *X-Frame-Options*, sehingga dapat mengakibatkan aplikasi berisiko terkena serangan *clickjacking* [23]. *Header response* HTTP *XFrame-Options* berfungsi untuk menunjukkan apakah *browser* diizinkan untuk me-*render* halaman di dalam bingkai atau *frame*. *Tool Burpsuite* digunakan untuk melihat *response* dan *X-Frame-Options* yang digunakan dengan cara melakukan *GET Request* menggunakan fitur *repeater*. Hasil dari pengujian menggunakan *tool Burpsuite* ditunjukkan pada Gambar 12 ditemukan informasi mengenai *response* sudah menggunakan *X-Frame-Options* SAMEORIGIN.

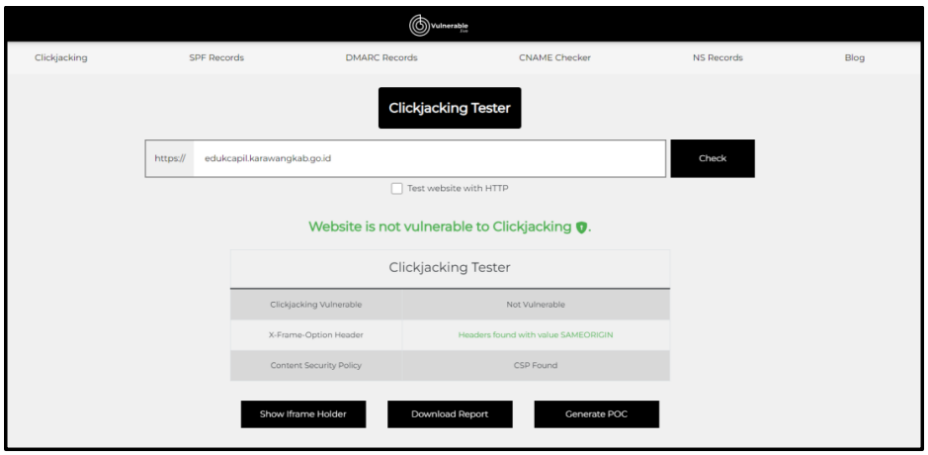


Gambar 12. *Request* dan *response X-Content-Type-Options Header Missing*

Selain itu dilakukan validasi kerentanan aplikasi *web edukcapil* terhadap *clickjacking* menggunakan aplikasi *clickjacker.io* dan *vulnerable.live*. *Clickjacker.io* merupakan aplikasi pengujian kerentanan *clickjacking* dengan analisis *X-Frame-Options* dan *Frame-Ancestors* yang dapat diakses melalui internet sedangkan vulnerable.live adalah *platform* yang memungkinkan untuk menguji situs *web* terhadap *clickjacking*, DMARC *Records*, SPF *Records*, NS *Records* dan CNAME *Records*. Hasil dari pengujian pada aplikasi *web* XYZ terhadap *clickjacking* menggunakan *clickjacker.io* tertera pada Gambar 13 dan *vulnerable.live* tertera pada Gambar 14.



Gambar 13. Hasil Pengujian Menggunakan *Clickjacker.io*



Gambar 14. Hasil Pengujian Menggunakan *Vulnerable.Live*

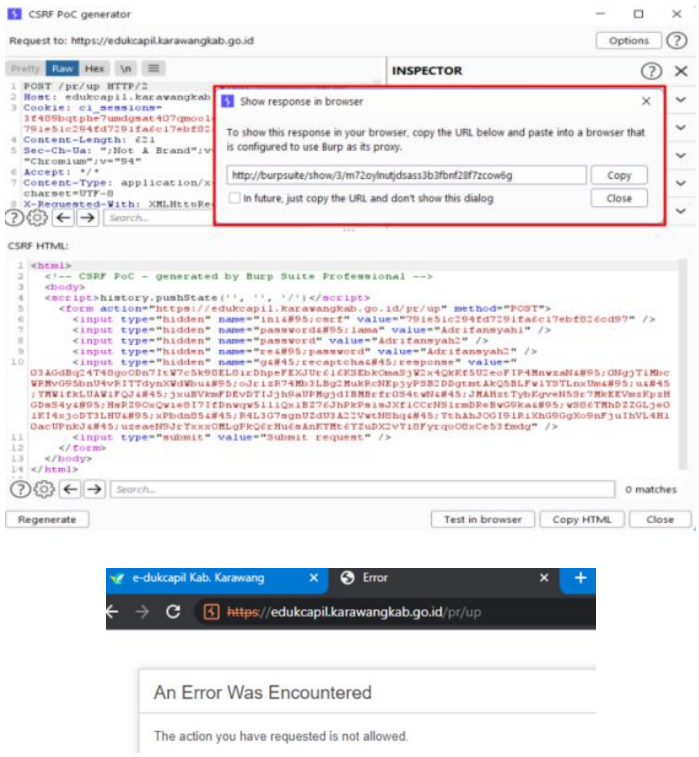
Hasil validasi kerentanan aplikasi *web* XYZ terhadap *clickjacking* menggunakan aplikasi *clickjacker.io* dan *vulnerable.live* sudah menggunakan *X-Frame-Options* SAMEORIGIN dan CSP *Header* (*Frame-Ancestors*) sehingga kerentanan *XContent-Type-Options Header Missing* tergolong kedalam ***False Positive.***

5. *Absence of Anti-CSRF Tokens*

Tabel 10. OWASP *Risk Rating* untuk *Absence of Anti-CSRF Tokens*

Threat Agents	*Attack Vector*	*Weakness prevalence*	*Weakness detectability*	*Technical impacts*	*Business impacts*
(?)	Sedang (2)	Biasa (2)	Mudah (3)	Cukup (2)	(?)

Hasil *scanning* aplikasi OWASP ZAP untuk kerentanan *Absence of Anti-CSRF Tokens* berada pada level *Low* didukung juga pada OWASP *Risk Rating* pada Tabel 10. Aplikasi *Burp Suite* digunakan untuk menunjukkan bahwa serangan dapat dilakukan dengan memanfaatkan *method* GET dalam permintaan paket HTTP pada *web server*, seperti pada Gambar 15.



Gambar 15. Hasil Setelah Dilakukan Ekstraksi CSRF

Apabila serangan *Cross Site Request Forgery* (CSRF atau XSRF) menyerang administrator dari aplikasi *web e-dukcapil* maka akan lebih berbahaya karena sistem keamanan dan keutuhan data yang terdapat dalam aplikasi *web* dapat dikompromikan. Serangan *Cross Site Request Forgery* (CSRF atau XSRF) terdeteksi oleh OWASP ZAP, namun setelah dilakukan uji coba kerentanan tersebut terdeteksi namun terbatasi oleh adanya fitur pihak ketiga dari *web e-dukcapil* yang menggunakan re-CAPTCHA, sehingga kerentanan tersebut menjadi ***False Positive***.

6. *Cookie Without Secure Flag*

Tabel 11. *Risk Rating* untuk *Cookie Without Secure Flag*

Threat Agents	*Attack Vector*	*Weakness prevalence*	*Weakness detectability*	*Technical impacts*	*Business impacts*
(?)	Mudah (3)	Tersebar Luas (3)	Mudah (3)	Cukup (2)	(?)

Hasil pemindaian aplikasi OWASP ZAP didukung OWASP *Risk Rating* pada Tabel 11 berada pada level *Low*. Tahap validasi yang dilakukan untuk membuktikan kerentanan ini berdasarkan subkategori pengujian: *Testing for Cookies Attributes* (WSTG-v42-SESS-02) menggunakan *Burp Suite* untuk melihat *cookie* yang dikirimkan pada *browser proxy* yang digunakan dan meneruskan ke fitur *repeater* untuk melihat *response* header permintaan yang dikirimkan, seperti pada Gambar 16. Hasil validasi yang telah dilakukan tidak terbukti rentan terhadap *Cookie Without Secure Flag* dikarenakan sudah menerapkan *secure flag* maka hasil pengujian ***False Positive***.

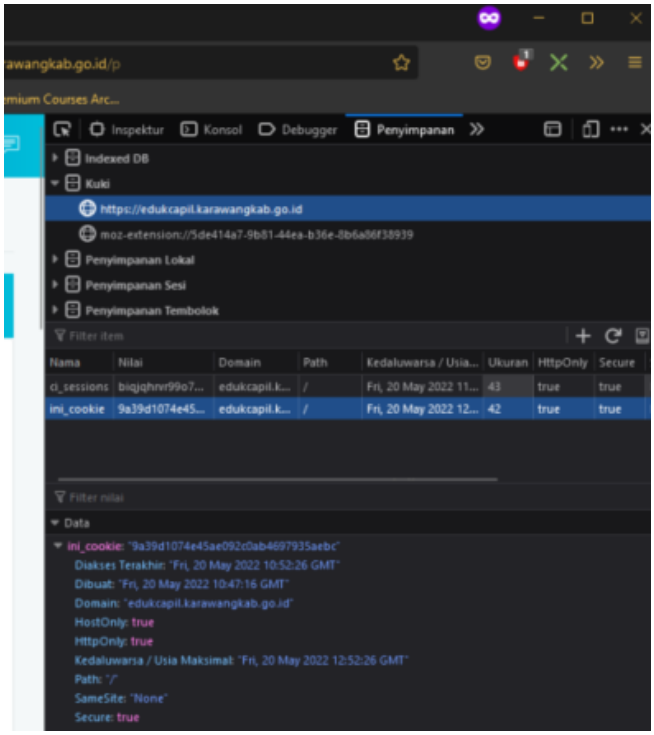
Gambar 16. *Response Header Cookie(s) Without Secure Flag*

7. *Cookie Without SameSite Attribute*

Tabel 12. *Risk Rating Cookie Without SameSite Attribute*

Threat Agents	*Attack Vector*	*Weakness prevalence*	*Weakness detectability*	*Technical impacts*	*Business impacts*
(?)	Mudah (3)	Tersebar Luas (3)	Mudah (3)	Cukup (2)	(?)

Sama dengan pemindaian sebelumnya seperti ditunjukkan dalam Tabel 12 kerentanan ini berada pada level *low*. Tahap validasi yang dilakukan untuk membuktikan kerentanan ini berdasarkan subkategori *Testing for Cookies Attributes* (WSTG-v42-SESS-02) secara manual pada *browser* yang digunakan untuk melihat atribut yang dikirimkan dengan bantuan fitur *inspect element* melalui fitur *storage* tertera pada Gambar 17. Aplikasi *web* XYZ tidak mengimplementasikan atribut *SameSite* pada *cookie* yang berarti *cookie* akan dikirim dalam semua konteks, yaitu sebagai *response*s and *cross-site requests* [24]. Hasil validasi yang telah dilakukan terbukti rentan terhadap *Cookie Without SameSite Attribute* dikarenakan belum menerapkan atribut *SameSite* pada semua *cookie* maka hasil pengujian ***True Positive***.



Gambar 17. Atribut SameSite Pada Cookie Masih None

Tahapan validasi kerentanan yang mendasar pada hasil yang diperoleh dari *vulnerability scanning* menggunakan *tool* terotomatisasi Vega dan OWASP ZAP, telah berhasil dilakukan dengan menguji 7 jenis kerentanan dari 9 kerentanan yang teridentifikasi. Shell Injection dan *Timestamp Disclousure* tidak dapat dilakukan pembuktian pada tahap validasi kerentanan karena dilakukan *maintenance* aplikasi *web* XYZ yang dilaksanakan pada bulan April 2022. Hasil uji penetrasi yang dilakukan terdapat 2 jenis kerentanan bersifat *True Positive* yang menunjukkan bahwa kerentanan tersebut benar benar ada dan terbukti merupakan suatu kerentanan yang memiliki dampak. Selain itu, terdapat 5 jenis kerentanan yang

bersifat *False Positive* yang menunjukkan bahwa kerentanan tersebut teridentifikasi sebagai kerentanan namun tidak memiliki dampak yang berarti aplikasi.

V. REKOMENDASI PERBAIKAN

A. *Session Cookie Without Secure Flag*

Menerapkan *secured c*ookies sesuai standarisasi ASVS 4.0.2 poin V3.4.1 tentang *Cookie-based Session Management* dengan rekomendasi pendukung CWE 614, yang menerapkan *cookie* dengan atribut "*Secure*" dan poin V3.4.3 dengan rekomendasi pendukung CWE 16 yang menerapkan *cookie* dengan atribut "*SameSite=Strict*" untuk membatasi eksposur terhadap serangan pemalsuan permintaan lalu lintas pada situs aplikasi *web* XYZ. Pengembang aplikasi dapat menerapkankonfigurasi *cookie* dengan atribut yang paling aman sebagai berikut: *Set-Cookie*: *Host-SID=<session token>*; *path=/*; *Secure*; *HttpOnly*; *SameSite=Strict*

B. *Vulnerable JS Library Versi 3.6.0*

Melakukan pembaharuan versi Bootstrap yang digunakan ke versi yang paling terbaru yaitu Bootstrap versi 5.2.

C. *Cross-Domain JavaScript Source File Inclusion*

Menerapkan standarisasi ASVS 4.0.2 poin V12.3.6 tentang *File Execution Requirements* dengan rekomendasi pendukung CWE 829 yang melakukan verifikasi bahwa aplikasi tidak menyertakan dan menjalankan fungsionalitas dari sumber tidak terpercaya, sehingga memastikan bahwa file sumber *JavaScript* yang dimuat hanya dari sumber terpercaya.

D. *X-ContentType-Options Header Missing*

Menerapkan standarisasi ASVS 4.0.2 poin V14.4.4 tentang HTTP *Security Headers Requirements* dengan rekomendasi pendukung CWE 116 yang melakukan verifikasi bahwa semua respon berisi *X-Content-Type-Options*: *nosniff header*. Memastikan aplikasi/*server web* menyetel *header Content-Type* dengan tepat, dan menyetel *header X-Content-Type-Options* ke *'nosniff'* untuk semua halaman *web* [25]. Jika memungkinkan, pastikan bahwa pengguna menggunakan *browser web* yang sesuai standar dan *modern* yang tidak melakukan atau diarah melakukan *sniffing* MIME.

E. *Absence of Anti-CSRF Tokens*

Menerapkan standarisasi ASVS 4.0.2 poin V4.2.2 *Operation Level Access Control* dengan rekomendasi pendukung CWE 352 yang melakukan verifikasi bahwa aplikasi atau kerangka kerja menerapkan mekanisme anti-CSRF yang kuat untuk melindungi fungsionalitas yang diautentikasi, dan anti-otomatisasi atau anti-CSRF yang efektif melindungi fungsionalitas yang tidak diautentikasi. CSRF yang menerapkan anti-CSRF dirancang dengan baik melibatkan atribut-atribut sebagai berikut.

a. *Token* anti-CSRF harus unik untuk setiap sesi pengguna
b. Sesi akan berakhir secara otomatis setelah jumlah waktu yang sesuai
c. *Token* anti-CSRF harus berupa nilai acak kriptografis yang panjangnya signifikan
d. Token anti-CSRF harus aman secara kriptografis, yaitu dihasilkan oleh algoritma *Pseudo Random Number Generator* (PRNG) yang kuat
e. Token anti-CSRF ditambahkan sebagai bidang tersembunyi untuk formulir, atau di dalam URL (hanya diperlukan jika permintaan GET menyebabkan perubahan status, yaitu, permintaan GET tidak idempoten), server harus menolak tindakan yang diminta jika token anti-CSRF gagal tervalidasi.

Ketika pengguna mengirimkan beberapa permintaan terotentikasi lainnya yang membutuhkan *Cookie* maka *token* anti-CSRF harus dimasukkan dalam permintaan. Kemudian, aplikasi *web* akan memverifikasi keberadaan dan kebenaran *token* ini sebelum memproses permintaan. Jika *token* hilang atau salah maka permintaan dapat ditolak.

F. *Cookie Without Secure Flag* dan *Cookie Without SameSite Attribute*

Menerapkan *secured cookies* sesuai standarisasi ASVS 4.0.2 poin V3.4.1 tentang *Cookie-based Session Management* dengan rekomendasi pendukung CWE 614, yang menerapkan *cookie* dengan atribut "*Secure*" dan poin V3.4.3 dengan rekomendasi pendukung CWE 16 yang menerapkan *cookie* dengan atribut "*SameSite=Strict*" untuk membatasi eksposur terhadap serangan pemalsuan permintaan lalu lintas pada situs aplikasi *web* SISPERON. Pengembang aplikasi dapat menerapkan konfigurasi *cookie* dengan atribut yang paling aman sebagai berikut: Set-Cookie: Host-SID=<*session token*>; *path=/Secure*; *HttpOnly*; *SameSite=Strict*

VI. KESIMPULAN

Berdasarkan hasil analisis kerentanan yang dilakukan ditemukan 9 jenis kerentanan dengan tingkat *high*, *medium*, dan *low*. Kerentanan tersebut terdiri dari 2 jenis kerentanan dengan tingkat *high*, 1 jenis kerentanan dengan tingkat *medium*, dan 6 jenis kerentanan dengan tingkat *low*. Setelah melakukan uji penetrasi sebagai bentuk validasi kerentanan, dari 9 jenis kerentanan terdapat dua yaitu *Shell Injection* dan *Timestamp Disclousure* tidak dapat dilakukan pengujian karena adanya proses *maintenance* dan *patching* sehingga tidak ditemukan lagi kerentanan tersebut.

REFERENSI

[1] BSSN RI, "Laporan Tahunan Honeynet Project Tahun 2019," 2019.
[2] APJII, "Hasil Survey Tahun 2018." 2018. [*Online*]. Available: http://www.apjii.or.id/
[3] D. KARAWANG, "Gambaran Umum." http://dukcapil.karawangkab.go.id/gambaran-umum (accessed Apr. 09, 2021).
[4] S. Juhani, "Mengembangkan Teologi Siber Di Indonesia," *J. Ledalero*, vol. 18, no. 2, p. 245, 2019, doi: 10.31385/jl.v18i2.189.245-266.

[5] I. Rahmawati, P. M. Pertahanan, and U. P. Indonesia, "The analysis of cyber crime threat risk management to increase cyber defense analisis manajemen risiko ancaman kejahatan siber (cyber crime) dalam peningkatan cyber defense," pp. 55–70.
[6] PUSOPSKAMSINAS, "Indonesia Cyber Security Monitoring Report 2019," *Indones. Secur. Incid. Response Team Internet Infrastruct.*, p. 42, 2020, [*Online*]. Available: https://cloud.bssn.go.id/s/nM3mDzCkgycRx4S/download
[7] BSSN, "Laporan Tahunan Gov-CSIRT," Jakarta, 2019.
[8] V. Appiah, M. Asante, I. K. Nti, and O. Nyarko-Boateng, "Survey of *Website*s and *web* application security threats using vulnerability assessment," *J. Comput. Sci.*, vol. 15, no. 10, pp. 1341–1354, 2019, doi: 10.3844/jcssp.2019.1341.1354.
[9] D. F. Priambodo, M. Hasbi, and M. S. Malacca, "Security Assessment Aplikasi Mobile E-Kinerja dengan Acuan OWASP Top 10 Mobile Risks," *JEPIN (Jurnal Edukasi dan Penelit. Inform.*, vol. 8, no. 3, pp. 560–571, 2022.
[10] D. F. Priambodo, G. S. Ajie, H. A. Rahman, A. C. F. Nugraha, A. Rachmawati, and M. R. Avianti, "Mobile Health Application Security Assesment Based on OWASP Top 10 Mobile Vulnerabilities," in *2022 International Conference on Information Technology Systems and Innovation (ICITSI)*, 2022, pp. 25–29. doi: 10.1109/ICITSI56531.2022.9970949.
[11] E-DUKCAPIL KARAWANG, "Layanan." https://edukcapil.karawangkab.go.id/ (accessed Apr. 09, 2021).
[12] B. V. Tarigan, A. Kusyanti, and W. Yahya, "Analisis Perbandingan *Penetration testing Tool* Untuk Aplikasi *Web*," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 1, no. 3, pp. 206–214, 2017.
[13] V. Appiah, I. Kofi Nti, and O. Nyarko-Boateng, "Investigating *Website*s and *Web* Application Vulnerabilities: *Web*master's Perspective," *Int. J. Appl. Inf. Syst.*, vol. 12, no. 3, pp. 10–15, 2017, doi: 10.5120/ijais2017451673.
[14] ISACA, "ISACA WP Vulnerability Assessment 1117," 2017. [*Online*]. Available: https://cybersecurity.isaca.org/
[15] R. M. Eli Saad, "OWASP *Web Security Testing Guide* v4-2," 2020.
[16] J. William, "OWASP Risk Rating Methodology," 2021. https://owasp.org/wwwcommunity/OWASP_Risk_Rating_Methodology (accessed Apr. 21, 2021).
[17] A. Goutam and V. K. Tiwari, "Vulnerability Assessment and *Penetration testing* to Enhance the Security of *Web* Application," *2019 4th Int. Conf. Inf. Syst. Comput. Networks*, pp. 601–605, 2019.
[18] S. Nagpure and S. Kurkure, "Vulnerability Assessment and *Penetration testing* of *Web* Application," *2017 Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2017*, pp. 1–6, 2018, doi: 10.1109/ICCUBEA.2017.8463920.
[19] A. A. Ali and M. Zamri Murah, "Security Assessment of Libyan Government *Website*s," *Proc. 2018 Cyber Resil. Conf. CRC 2018*, pp. 1–4, 2019, doi: 10.1109/CR.2018.8626862.
[20] L. T. M. Blessing and A. Chakrabarti, *DRM, a Design Research Methodology*. London: Springer London, 2009. doi: 10.1007/978-1-84882-587-1.
[21] T. Jain and N. Jain, "'*Framework* for *Web* Application Vulnerability Discovery and Mitigation by Customizing Rules Through ModSecurity,'" *2019 6th Int. Conf. Signal Process. Integr. Networks, SPIN 2019, pp. 643–648*, 2019.
[22] "CVE-2019-8331 : In Bootstrap before 3.4.1 and 4.3.x before 4.3.1, XSS is possible in the *tool*tip or popover data-template attribute." https://www.cvedetails.com/cve/CVE-2019-8331/ (accessed Nov. 07, 2022).
[23] A. Lavrenovs and F. J. R. Melón, "HTTP security headers analysis of top one million *Website*s," in *2018 10th International Conference on Cyber Conflict (CyCon)*, 2018, pp. 345–370. doi: 10.23919/CYCON.2018.8405025.
[24] developer mozilla, "SameSite cookies," 2021. https://developer.mozilla.org/enUS/docs/*Web*/HTTP/Headers/Set-Cookie/SameSite (accessed Apr. 21, 2021).
[25] "OWASP ZAP – Export Report." https://www.zaproxy.org/docs/desktop/addons/export-report/ (accessed Nov. 07, 2022).