Analisis Keamanan Aplikasi e-STR Kementerian Kesehatan: Studi Kasus SQL Injection, Stored XSS, dan Nonce Reuse pada Sistem Registrasi Tenaga Kesehatan

Saeful Diyan Pratama[*1], **Aris Tri Jaka Harjanta** [2]
[1]Informatika, Universitas PGRI Semarang, Kota Semarang
[2]Informatika, Universitas PGRI Semarang, Kota Semarang

*Email: 22670115@upgris.ac.id

Abstract.

The security of government web applications is becoming increasingly critical as digital public services continue to grow. This study aims to assess the security posture of the e-STR system developed by the Indonesian Ministry of Health, a digital platform designed for healthcare professional registration. The assessment was conducted using a black-box approach, testing the system from an external perspective without access to its source code. The scope includes both user-facing interfaces and backend API endpoints.The results revealed three critical vulnerabilities. First, SQL Injection was identified, allowing malicious input to manipulate database queries. Second, a Stored Cross-Site Scripting (XSS) vulnerability was found in the admin form, enabling scripts to persist in the system and execute on the client side. Third, nonce reuse was discovered, indicating that authentication tokens could be used multiple times within an extended period, making the system susceptible to replay attacks.These findings suggest that security has not been fully integrated into the system's initial design. Therefore, improvements are needed, such as proper input validation, the use of parameterized queries, and a stricter token management mechanism. This research reinforces the importance of embedding security throughout the software development lifecycle, especially in public-facing digital services

Keywords: web application security; SQL Injection; XSS; nonce reuse, e-STR; black-box testing

Abstrak

Keamanan aplikasi web pemerintah menjadi hal yang semakin penting seiring meningkatnya digitalisasi layanan publik. Penelitian ini dilakukan untuk menguji tingkat keamanan pada sistem e-STR Kemenkes, sebuah platform digital untuk registrasi tenaga kesehatan, yang sedang dalam tahap pengembangan. Pengujian dilakukan melalui pendekatan black-box, di mana sistem diuji dari sisi luar tanpa akses ke kode sumber. Fokus pengujian mencakup antarmuka pengguna dan endpoint API.Hasil pengujian menunjukkan adanya tiga kerentanan kritis. Pertama, SQL Injection memungkinkan penyerang menyisipkan perintah berbahaya ke dalam basis data melalui parameter yang tidak aman. Kedua, *Stored* XSS ditemukan pada form admin, di mana skrip berbahaya yang dimasukkan dapat dieksekusi kembali di sisi pengguna. Ketiga, *reuse* pada *nonce* menunjukkan bahwa token autentikasi masih bisa digunakan ulang dalam jangka waktu yang lama, membuka peluang serangan ulang atau *replay attack*.Temuan ini menandakan bahwa aspek keamanan belum sepenuhnya menjadi bagian dari desain awal sistem. Oleh karena itu, dibutuhkan perbaikan menyeluruh, mulai dari sanitasi input, penggunaan query yang aman, hingga pengelolaan autentikasi yang lebih ketat. Pengujian ini juga menegaskan pentingnya integrasi keamanan sejak awal proses pengembangan aplikasi.

Kata kunci: Keamanan Aplikasi Web; SQL Injection; XSS; Nonce Reuse, e-STR; Black Box Testing

1. Pendahuluan

Digitalisasi layanan publik oleh instansi pemerintah menjadi sangat vital dalam rangka meningkatkan efisiensi dan aksesibilitas masyarakat. Salah satu sistem yang kini tengah dikembangkan oleh Kementerian Kesehatan RI adalah **e-STR** (*Electronic* Surat Tanda Registrasi) di bawah pengelolaan KTKI. Sistem ini merupakan sarana legalisasi praktik tenaga kesehatan dan menyimpan data sensitif pengguna, sehingga aspek keamanan aplikasi adalah prioritas tinggi.

Penelitian di Indonesia menunjukkan kerentanan signifikan pada aplikasi web pemerintah terkait celah ***SQL Injection* (SQLi)** dan ***Cross-Site Scripting* (XSS)**. Sebagai contoh, audit penetrasi terhadap sejumlah situs pemerintah menemukan puluhan celah SQLi dan XSS menggunakan metode *black-box* berdasarkan *framework* OWASP Top 10 [1]. Selain itu, metode mitigasi seperti penggunaan Web *Application Firewall* (WAF), sebagaimana dikembangkan oleh H. Muhammad dkk. (2024), terbukti efektif dalam mencegah serangan SQLi dan XSS [2].

Namun, sebagian besar studi tersebut belum mengeksplorasi secara praktis kerentanan pada layanan kritikal berbasis API, seperti sistem e-STR. Beberapa penelitian terdahulu banyak berfokus pada metode umum pengamanan aplikasi web tanpa mengungkap kelemahan autentikasi dan penggunaan *nonce* yang mungkin dapat dieksploitasi.

Dalam kesempatan magang industri di PT ABISEKA, dilakukan pengujian penetrasi berbasis *black-box* terhadap aplikasi web dan API e-STR. Hasilnya mengungkap kerentanan serius berupa **SQL *Injection* berbasis waktu**, ***Stored* XSS**, serta ***Nonce Reuse Vulnerability***, yang memungkinkan *bypass* autentikasi, manipulasi data, dan eksekusi skrip berbahaya. Temuan ini menunjukkan bahwa meskipun sistem telah menerapkan sanitasi dan parameterisasi, celah teknis praktis masih dapat dimanfaatkan oleh penyerang. **Kontribusi artikel ini** meliputi:

1. Dokumentasi hasil pengujian penetrasi praktis terhadap sistem e-STR,
2. Analisis mendalam terhadap kerentanan yang ditemukan,
3. Rekomendasi mitigasi berdasarkan praktik terbaik industri untuk meningkatkan keamanan aplikasi pemerintah di sektor kesehatan.

Pengujian menggunakan pendekatan ***black-box* testing** pada *endpoint* web dan API mensimulasikan skenario serangan dari pihak eksternal tanpa pengetahuan internal sistem.

2. Metode

Pengujian dilakukan dengan pendekatan ***black-box***, di mana penguji tidak memiliki akses terhadap kode sumber aplikasi dan hanya mengevaluasi sistem dari sisi luar sebagaimana pengguna umum. Metode ini sesuai dengan pendekatan yang umum digunakan dalam pengujian keamanan web berbasis OWASP *Top Ten* [3] dan telah dibahas secara luas dalam pengujian keamanan sistem web secara praktik [4].

A. Lingkungan Uji (*Testing Environment*)

Pengujian dilakukan terhadap versi pengembangan (*development environment*) aplikasi **ktki.kemkes.go.id/dev**, yang diberikan aksesnya oleh pihak pengelola sistem. Semua pengujian dijalankan menggunakan perangkat laptop dengan spesifikasi minimum:

- OS: Kali Linux 2023.3
- RAM: 4 GB
- Browser: Firefox + Developer Tools
- Tools tambahan: Burp Suite Profesional, OWASP ZAP, SQLMap, Postman, dan curl Untuk pengujian terhadap API, digunakan *request replay* dan manipulasi payload menggunakan Burp Repeater dan Postman untuk mendeteksi kelemahan validasi token dan *nonce*.

B. Tahapan Pengujian

1) Enumerasi dan Identifikasi Endpoint

Langkah awal dilakukan dengan memetakan seluruh *endpoint* yang tersedia pada aplikasi dan API. Enumerasi dilakukan menggunakan teknik *passive reconnaissance* (melalui tampilan aplikasi) dan *active enumeration* menggunakan Burp Suite untuk menganalisis struktur URL dan parameter.

2) Uji SQL *Injection*

Pengujian terhadap kerentanan **SQL *Injection*** dilakukan pada *endpoint* /adminvalidasipemohon/getregbaru. Uji awal dilakukan dengan menyisipkan *payload* sederhana seperti ' OR '1'='1 dan ';SELECT PG_SLEEP(5)-- pada parameter *start*, untuk mengamati *delay* pada waktu respon.

- Tools: **SQLMap** digunakan untuk mengeksekusi *time-based blind SQLi*.
- Tujuan: Mengetahui kemungkinan injeksi perintah SQL yang dapat dimanfaatkan untuk manipulasi atau pencurian data.

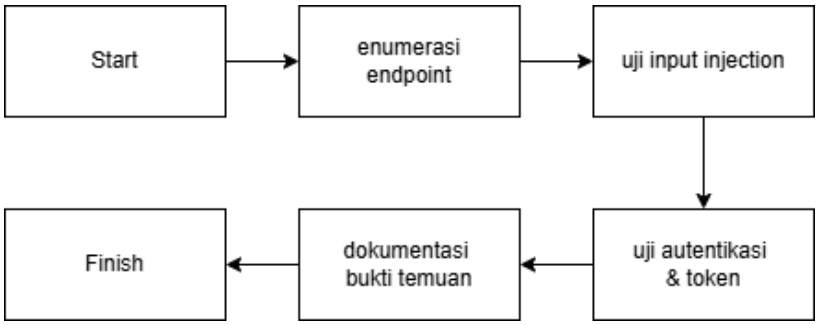
3) Uji *Nonce Reuse Vulnerability*

Pengujian dilakukan pada endpoint /encryption/encode dan /encryption/decode. Peneliti mengamati bahwa *hash* hasil *encode* masih dapat digunakan untuk proses *decode* hingga lebih dari 24 jam kemudian. Hal ini menandakan adanya celah *replay attack* pada mekanisme otentikasi.

- Proses: *Hash* disimpan selama beberapa jam, lalu dikirim ulang ke *endpoint / decode*.
- Hasil: Sistem tetap menerima hash tersebut sebagai valid, menandakan bahwa *nonce* tidak diatur sebagai sekali pakai (*one-time use*).

4) Struktur Alur Pengujian

Berikut ini adalah alur pengujian yang dilakukan



5) Dokumentasi dan validasi temuan

Setiap temuan diuji minimal dua kali untuk memastikan konsistensi hasil. Semua bukti berupa *screenshot*, *response logs*, dan payload dicatat dan dilaporkan. Validasi dilakukan juga bersama tim pengembang internal dari pihak Kemenkes untuk memastikan kerentanan benar-benar dapat direproduksi dan tidak berasal dari false positive.

3. Hasil dan Pembahasan

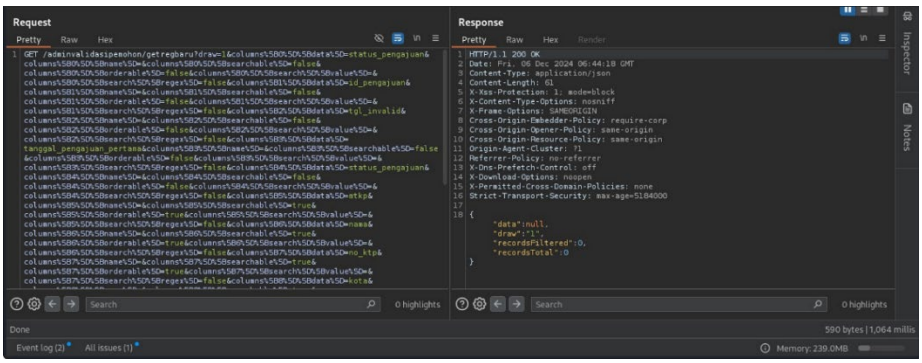
3.1. Penyajian Hasil

Pengujian keamanan terhadap sistem e-STR Kemenkes dilakukan pada lingkungan pengembangan (*development*) dengan pendekatan *black-box*. Berdasarkan tahapan metode yang dilakukan, ditemukan tiga kerentanan utama yang dinilai kritis berdasarkan potensi dampaknya terhadap keamanan data dan sistem.

a) Hasil Pengujian SQL Injection

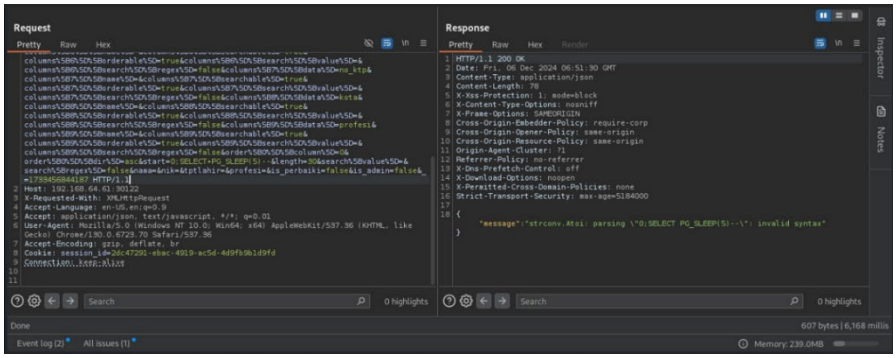
Pengujian terhadap *endpoint* /adminvalidasipemohon/getregbaru berhasil mengonfirmasi adanya kerentanan SQL *Injection* berbasis waktu (*time-based blind SQLi*). Saat parameter *start* diubah menggunakan *payload* ';SELECT PG_SLEEP(5)--, sistem memberikan respons yang tertunda selama lima detik, yang menunjukkan

bahwa input pengguna dimasukkan langsung ke dalam *query* SQL tanpa validasi atau parameterisasi yang memadai. Di bawah ini merupakan waktu respons yang normal yaitu 1,064 *millis* atau sekitar 1 detik



Gambar 1. respons normal

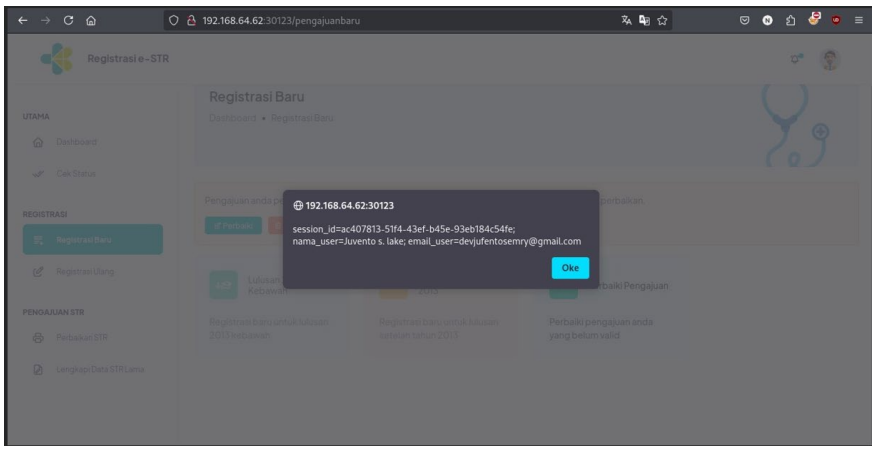
Pada gambar di bawah ini merupakan hasil setelah dilakukan sql *injection* menggunakan *payload* SELECT PG_SLEEP(5) yang menghasilkan waktu 6,168 **millis** atau sekitar 6 detik



Gambar 2. respons setelah dilakukan sql injection

b) Hasil Pengujian *Stored* XSS

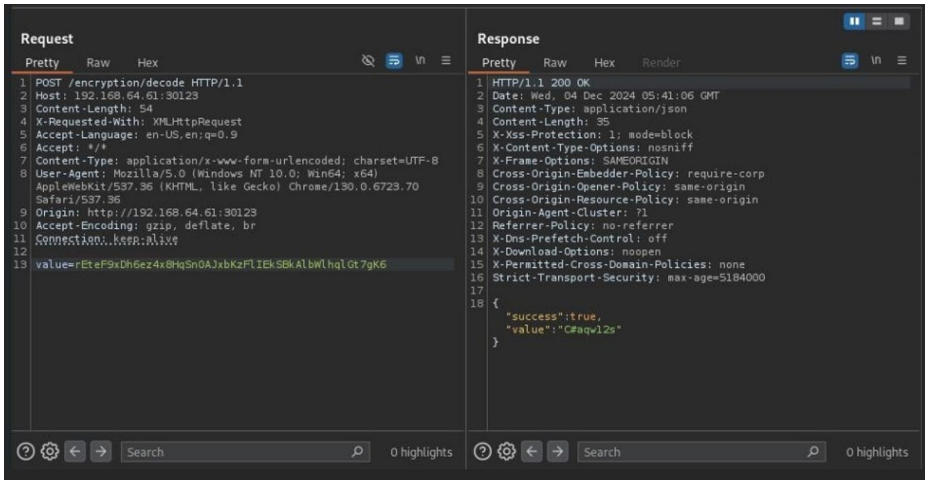
Kerentanan Stored XSS ditemukan pada fitur form “alasan perbaikan” yang diakses melalui endpoint admin /adminvalidasipemohon dan /adminvalfixing. *Payload* <script>alert('XSS')</script> yang disisipkan pada kolom tersebut berhasil disimpan dalam basis data dan kemudian ditampilkan kembali tanpa sanitasi pada *endpoint* /pengajuanbaru milik pengguna. Hal ini menyebabkan skrip dieksekusi di sisi klien, membuka potensi pencurian *cookie* atau pengambilalihan sesi.



Gambar 3. XSS termuat di dalam tampilan pengguna

c) Hasil Pengujian Nonce Reuse

Endpoint /encryption/encode menghasilkan hash acak sebagai bagian dari proses autentikasi. Namun, *hash* yang telah di-generate tetap dapat digunakan berulang kali dalam waktu lebih dari 24 jam melalui endpoint /encryption/decode, tanpa adanya validasi keunikan atau batas waktu penggunaan (*nonce reuse*). Temuan ini menunjukkan potensi *replay attack*, di mana penyerang dapat merekam *hash* dan menggunakannya kembali untuk melewati proses autentikasi.



Gambar 4. Percobaan pemanggilan ulang hash lama (berusia lebih dari 24 jam) yang tetap diterima oleh endpoint decode, membuktikan adanya reuse

3.2. Pembahasan

Berdasarkan hasil pengujian terhadap aplikasi e-STR, ditemukan tiga jenis kerentanan kritis, yaitu SQL *Injection*, *Stored* XSS, dan *reuse* pada mekanisme *nonce*. Ketiganya merepresentasikan celah keamanan yang cukup umum dijumpai pada aplikasi web modern, namun tetap berisiko tinggi apabila tidak ditangani dengan benar.

Kerentanan SQL Injection yang ditemukan melalui teknik *time-based* menunjukkan bahwa *query* yang dibangun oleh sistem masih belum menggunakan mekanisme parameterisasi yang aman. Respons server yang tertunda ketika payload PG_SLEEP(5) disisipkan menjadi indikasi kuat bahwa data input dari pengguna tidak difilter secara aman sebelum diproses oleh *database*. Temuan ini memperkuat penelitian oleh Fernandes dan Lina (2024), yang menunjukkan bahwa SQL *Injection* tetap menjadi salah satu vektor serangan utama pada aplikasi yang tidak menggunakan *prepared statements* secara konsisten [5].

Sementara itu, kerentanan *Stored* XSS ditemukan melalui *input* pada kolom "alasan perbaikan" dari sisi admin yang berhasil ditampilkan kembali tanpa proses encoding pada sisi pengguna. Hal ini merupakan kegagalan dalam validasi *input* dan penanganan *output*, dua komponen penting dalam model keamanan sisi server. Pada aplikasi admin panel di mana terdapat kerentanan *stored* XSS pada fitur manajemen pengguna. Penyerang dapat menyisipkan *payload* berbahaya ke *field username* saat pembuatan *user*. Payload ini kemudian dieksekusi ketika admin menghapus *user* tersebut, sehingga *script* berjalan di browser admin. Dampaknya meliputi pencurian token sesi admin, pengambilan data sensitif, hingga potensi pengambilalihan aplikasi secara penuh jika hak admin berhasil dieksploitasi [6].

Selain dua kerentanan sebelumnya, ditemukan kelemahan pada mekanisme *nonce* yang digunakan dalam proses autentikasi. Sistem masih menerima *hash* atau token yang sama meskipun telah digunakan lebih dari 24 jam sebelumnya, yang menunjukkan bahwa nonce tidak diperlakukan sebagai nilai unik yang hanya berlaku sekali. Menurut Packetlabs, nonce harus dijamin keunikannya untuk setiap transaksi agar dapat mencegah *replay attack*. Jika nonce dapat digunakan ulang atau tidak memiliki masa berlaku yang singkat, maka

penyerang berpeluang merekam dan mengirim ulang token tersebut untuk mendapatkan akses yang tidak sah. Oleh karena itu, validasi keunikan dan pembatasan masa aktif *nonce* merupakan langkah krusial dalam menjaga keamanan sistem autentikasi dan mencegah eksploitasi *replay attack* [7].

Ketiga temuan tersebut menunjukkan proses pengembangan e-STR belum sepenuhnya menerapkan prinsip keamanan dari awal (*secure-by-design*). Menurut dokumen HHS (2022), menerapkan praktik keamanan di seluruh tahap SDLC dari *requirement* hingga *deployment* dapat mengurangi celah desain dan implementasi [8]. Hal ini didukung oleh konsep *Secure Development Lifecycle* (SSDLC) dari Invicti (2021), yang menyoroti pentingnya integrasi otomatisasi pengujian keamanan seperti DAST/DAST dalam *pipeline* CI/CD [9]

4. Kesimpulan

Berdasarkan hasil pengujian keamanan terhadap sistem e-STR Kemenkes menggunakan pendekatan *black-box*, ditemukan tiga kerentanan utama yang tergolong kritis, yaitu **SQL *Injection***, ***Stored* XSS**, dan ***Nonce Reuse Vulnerability***. Ketiga kerentanan tersebut menunjukkan bahwa sistem belum sepenuhnya aman dari eksploitasi pihak luar, dan masih terdapat celah yang dapat dimanfaatkan untuk *bypass* autentikasi, manipulasi data, serta eksekusi skrip berbahaya.

Kerentanan SQL *Injection* ditemukan pada *endpoint* validasi pemohon yang memproses *input* tanpa *parameterisasi query*. *Stored* XSS terjadi karena lemahnya validasi *input* pada kolom "alasan perbaikan", yang menyebabkan skrip jahat tersimpan dan dieksekusi kembali di sisi pengguna. Sedangkan pada *Nonce Reuse*, sistem masih menerima *hash* yang seharusnya hanya digunakan satu kali, bahkan setelah lebih dari 24 jam, sehingga membuka kemungkinan serangan *replay*.

Temuan-temuan ini mempertegas pentingnya penerapan prinsip ***secure-by-design*** dalam proses pengembangan aplikasi, serta integrasi pendekatan ***Secure Software Development Lifecycle* (SSDLC)** untuk menjamin keamanan sistem sejak tahap perancangan hingga implementasi. Selain itu, pengujian penetrasi harus menjadi bagian rutin dalam siklus pengembangan, tidak hanya dilakukan di akhir proses atau menjelang peluncuran sistem.

Untuk meningkatkan ketahanan keamanan aplikasi e-STR, tim pengembang disarankan menerapkan langkah mitigasi berikut:

- Menggunakan *parameterized queries* untuk seluruh interaksi dengan basis data.
- Menerapkan validasi *input* dan *encoding output* pada semua *endpoint*, termasuk yang bersifat internal seperti antarmuka admin.
- Memastikan *nonce* atau token bersifat unik, sekali pakai, dan memiliki waktu kedaluwarsa yang jelas.

Melalui integrasi pendekatan keamanan sejak dini, diharapkan sistem e-STR tidak hanya mampu menjalankan fungsinya secara administratif, tetapi juga dapat memberikan jaminan perlindungan data bagi seluruh penggunanya.

5. Referensi

[1] N. A. Prasetiyo, R. B. Huwae, And A. H. Jatmika, “Audit Dan Analisis Website Pemerintah Menggunakan Pengujian Penetrasi Sql Injection Dan Cross Site Scripting (Xss) (Audit And Analysis Of Government Websites Using Sql Injection And Cross-Site Scripting (Xss) Penetration Testing).” [Online]. Available: Http://Jtika.If.Unram.Ac.Id/Index.Php/Jtika/

[2] H. Haikal Muhammad, A. Id Hadiana, And H. Ashaury, “Pengamanan Aplikasi Web Dari Serangan Sql Injection Dan Cross Site Scripting Menggunakan Web Application Firewall,” *Jati (Jurnal Mahasiswa Teknik Informatika)*, Vol. 7, No. 5, 2024, Doi: 10.36040/Jati.V7i5.7320.

[3] Owasp Foundation, “Owasp Top Ten Web Application Security Risks 2021,” Https://Owasp.Org/Top10. Accessed: Jun. 06, 2025. [Online]. Available: Https://Owasp.Org/Top10
[4] R. R. Yusuf And T. N. Suharsono, “Prosiding Seminar Sosial Politik, Bisnis, Akuntansi Dan Teknik (Sobat) Ke-5 Bandung, 28 Oktober 2023 Pengujian Keamanan Dengan Metode Owasp Top 10 Pada Website Eform Helpdesk”.
[5] C. Dougherty, “Practical Identification Of Sql Injection Vulnerabilities Background And Motivation,” 2012. [Online]. Available: Http://Cwe.Mitre.Org/Top25/Archive/2011/2011_Cwe_Sans_Top25.Html
[6] Yogeshojha, “Stored Xss On Admin Panel When Deleting A User,” Https://Github.Com/Yogeshojha/Rengine/Security/Advisories/Ghsa-23wx-5q5w-334w.
[7] Packetlabs, “A Guide To Replay Attacks And How To Defend Against Them,” Https://Www.Packetlabs.Net/Posts/A-Guide-To-Replay-Attacks-And-How-To-Defend-Against-Them/.
[8] “Web Application Attacks In Healthcare,” 2022.
[9] Zbigniew Banach, “Building A Secure Sdlc For Web Applications,” Https://Www.Invicti.Com/Blog/Web-Security/Secure-Software-Development-Lifecycle-Ssdlc-Web-Applications.