Laman web jurnal: https://ejournal.unama.ac.id/index.php/processor

Processor: Jurnal Ilmiah Sistem Informasi, Teknologi Informasi dan Sistem Komputer

P-ISSN: 1907-6738 | E-ISSN: 2528-0082

Uji Keamanan Back end Aplikasi Berbasis Website Menggunakan Metode Black Box Testing

Abd. Wahab Syahroni[1], Nindian Puspa Dewi [2], Nilam Ramadhani[3] , Ubaidi[4], Badar Said[5]

[1,2,3,4,5]Program Studi Informatika, Fakultas Teknik, Universitas Madura, Jl. Raya Panglegur No.Km 3,5 Kec. Tlanakan, Kab. Pamekasan, Jawa Timur 69371, Indonesia

Abstrak- Keamanan aplikasi sering terabaikan saat pengembangan aplikasi bahkan setelah aplikasi digunakan. Padahal, tanpa pengamanan yang baik, sehebat apapun teknologi yang digunakan dapat menyebabkan kerugian seperti pengaksesan data oleh pengguna yang tidak sah serta kemungkinan kehilangan data karena tindakan penghapusan data. Pengembangan aplikasi menggunakan arsitektur REST API memungkinkan pengguna mengakses endpoint Back end dari luar aplikasi sehingga tidak hanya autentikasi yang diperhatikan tapi juga masalah autorisasi. Pengujian dilakukan menggunakan metode Black Box Testing dengan bantuan alat Postman, yang menunjukkan bahwa terdapat kelemahan dalam otorisasi. Berdasarkan hasil pengujian pada aplikasi SILAB dengan metode Black Box Testing dapat dinyatakan bahwa aplikasi SILAB masih perlu perbaikan pada tingkat keamanan Back end terutama dalam hal autorisasi. Hal tersebut menunjukan bahwa masih terdapat celah dan ancaman yang dapat berpotensi merusak data pada aplikasi SILAB.

Kata Kunci: Keamanan Aplikasi; REST API; Autentikasi; Autorisasi; Black Box Testing.

Abstract- Application security is often overlooked during the development phase and even after the application is deployed. However, without proper security measures, even the most advanced technologies can lead to significant losses, such as unauthorized data access and potential data loss due to deletion actions. Developing applications using the REST API architecture allows users to access backend endpoints from outside the application, so attention must be given not only to authentication but also to authorization issues. Testing was conducted using the Black Box Testing method with the help of the Postman tool, which showed that there were weaknesses in authorization. Based on the results of testing the SILAB application using the Black Box Testing method, it can be concluded that the SILAB application still needs improvements in backend security, particularly in terms of authorization. This indicates that there are still vulnerabilities and threats that could potentially compromise the data in the SILAB application.

Keywords: Application security; REST API; Authentication; Authorization; Black Box Testing .

1. PENDAHULUAN

Tahun 2024 ini, Universitas Madura telah meresmikan gedung baru dengan nama Gedung Laboratorium Bersama (GLB). Laboratorium PRODI Informatika yang bertempat di GLB bersama dengan laboratorium dari PRODI lain telah resmi menggunakan Sistem Informasi laboratorium (SILAB). SILAB merupakan aplikasi untuk manajemen laboratorium yang dapat membantu pengelolaan laboratorium secara efisien. SILAB merupakan aplikasi berbasis web dan terintegrasi dengan Sistem Informasi Manajemen Akademik Terpadu (SIMAT) Universitas Madura[1].

Untuk dapat mengakses SILAB, User dapat login menggunakan akun SIMAT. Adapun hak akses utama pada aplikasi SILAB adalah Mahasiswa yang ikut praktikum dan Dosen Praktikum. User Dosen Praktikum dapat melakukan manajemen data berita acara praktikum seperti menambah, melihat dan menghapus, User Dosen Praktikum juga dapat melakukan absensi kehadiran Mahasiswa yang sedang mengikuti praktikum. Adapun User Mahasiswa dapat memvalidasi berita acara yang telah di inputkan oleh Dosen Praktikum. Dan masih ada beberapa fitur lainnya untuk kedua user tersebut.

Aplikasi berbasis website yang baik biasanya dilengkapi dengan beberapa komponen penting seperti database, framework, back end, front end, server, Aplication Programming Interface (API), dan Secure Socket layer (SSL)[2]. Salah satu hal yang penting untuk dijadikan perhatian dalam suatu pengamanan aplikasi berbasis front end dan back end adalah API[3]. Aplikasi SILAB memanfaatkan arsitektur REST (Representational State Transfer) API yaitu jenis arsitektur API yang menyediakan standar komunikasi antara berbagai sistem atau perangkat lunak melalui protokol HTTP[4]. REST adalah antarmuka sederhana untuk mentransfer informasi

Submitted: 01 Juni 2024; Reviewed Round#1: 27 Juni 2024; Reviewed Round#2:03 Juli 2024 ; Reviewed Round #3: 29 September 2024; Accepted; 08 Oktober 2024; Published: 31 Oktober 2024

yang tidak menggunakan lapisan perangkat lunak pihak ketiga. Artinya, saat mengirim data, tidak ada tahap konversi, informasi disampaikan dalam bentuk aslinya, REST merupakan gaya arsitektur API untuk membuat layanan web. Gaya ini diperkenalkan pada tahun 2000 dalam disertasi doktoral oleh salah satu pencipta protokol HTTP, Roy Fielding[5]. Pada arsitektur REST API, setiap URL unik adalah sebuah objek, untuk dapat memperoleh objek konten dapat menggunakan HTTP method GET, sedangkan untuk memodifikasi dapat menggunakan method POST, PUT, maupun DELETE[3].

Keamanan pada aplikasi berbasis website yang memanfaatkan arsitektur REST API merupakan hal yang sangat penting karena dapat melindungi aplikasi website dari serangan jahat, baik pihak yang tidak punya akses untuk login (authentication), maupun pihak dalam yang telah memiliki hak akses untuk login (authorization). Untuk melakukan pengujian keamanan pada sebuah aplikasi berbasis web, seseorang menggunakan SQL Injection pada form login agar bisa masuk ke dalam aplikasi tersebut[6]. Namun, mengingat SILAB menggunakan akun login dari SIMAT, maka tidak perlu melakukan SQL Injection hanya untuk masuk ke dalam SILAB. Permasalahan dalam penelitian ini adalah ketika User sudah bisa masuk ke dalam SILAB menggunakan akun SIMAT, User dapat melihat sub-bagian yang secara khusus menampilkan semua permintaan jaringan yang dilakukan menggunakan Fetch API atau XMLHttpRequest (XHR), dimana kedua teknik ini sering digunakan untuk membuat permintaan HTTP asinkron dari halaman web ke server, dan memungkinkan pembaruan halaman tanpa harus melakukan reload penuh, yang dapat dilihat melalui inspect element halaman browser google chrome, pilih tab network, pilih pada bagian Fetch/XHR.

Gambar 1. Tab Network pada Inspect Elemen Browser

Dari gambar 1, User yang sudah login ke SILAB, dapat melihat komunikasi antara halaman web (Front End) dengan server (Back End), termasuk Request URL dan Request Method nya.

Beberapa contoh spesifik mengenai ancaman keamanan yang sering dihadapi oleh aplikasi berbasis web yang menggunakan arsitektur REST API antara lain

a. Broken Object Level Authorization (BOLA)
Ancaman ini terjadi ketika API mengizinkan akses langsung ke objek berdasarkan input pengguna tanpa melakukan pemeriksaan otorisasi yang memadai. Misal endpoint '/api/pesan/100' mengembalikan detail pesanan dengan ID 100. Pengguna dapat mengubah ID menjadi pesanan orang lain, seperti '/api/pesanan/150', dan mengakses detail pesanan tersebut tanpa izin yang tepat. Ancaman ini juga dikenal dengan nama IDOR (Insecure Direct Object References) [7].

b. Broken Function Level Authorization (BFLA)
Ancaman ini terjadi ketika API gagal menerapkan pembatasan akses pada fungsi atau metode tertentu berdasarkan hak akses pengguna. Misal endpoint '/api/admin/hapusUser' hanya diakses oleh administrator, namun karena kurangnya pemeriksaan otorisasi, pengguna biasa dapat mengakses dan menggunakan endpoint tersebut untuk menghapus pengguna lain.

c. Insufficient Authorization Checks
Ini terjadi ketika API tidak melakukan pemeriksaan otorisasi yang cukup sebelum proses permintaan. Misal endpoint '/api/user/updateUser' memperbarui profil pengguna berdasarkan ID Pengguna yang

https://doi.org/10.33998/processor.2024.19.2.1752
Submitted: 01 Juni 2024; Reviewed Round#1: 27 Juni 2024; Reviewed Roung#2:03 Juli 2024 ; Reviewed Roung #3: 29 September 2024; Accepted; 08 Oktober 2024; Published: 31 Oktober 2024

disertakan dalam body permintaan. Jika API tidak melakukan verifikasi bahwa pengguna yang membuat permintaan memiliki izin untuk memperbarui data dengan ID yang disediakan, pengguna dapat memperbarui data pengguna lain.

d. Privilege Escalation
Terjadi ketika pengguna mendapatkan hak akses lebih tinggi dari yang seharusnya karena adanya kelemahan dalam autorisasi. Misal pengguna dengan peran 'user' menemukan bahwa mereka dapat mengakses endpoint '/api/admin/getAllUsers' dan mengambil daftar semua pengguna, meskipun seharusnya hanya administrator yang bisa mengakses endpoint tersebut.

Penelitian ini bertujuan untuk mengidentifikasi dan mengatasi celah keamanan pada aplikasi SILAB, khususnya pada aspek otorisasi pengguna dengan melakukan uji keamanan pada Request URL yang dapat dilihat oleh User dalam tab network Google Chrome, baik yang berkaitan dengan Authentication maupun Authorization sehingga nantinya dapat memberikan informasi yang bermanfaat terkait pengamanan REST API pada aplikasi berbasis web secara umum.

2. TINJAUAN PUSTAKA

2.1. Authentication dan Authorization

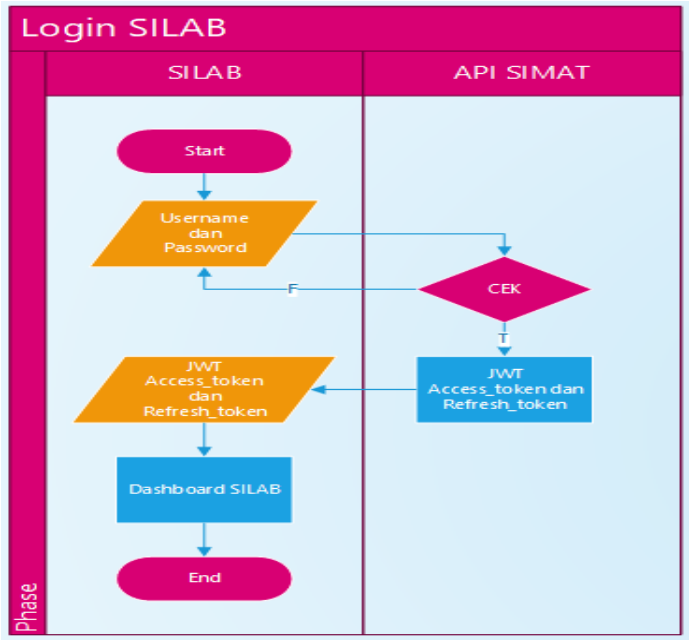
Authentication dan Authorization merupakan dua konsep keamanan yang berbedan tetapi sangat sering digunakan dalam sistem keamanan aplikasi website. Authentication (Autentikasi) merupakan proses verifikasi identitas entitas atau user, pada umumnya user menggunakan username dan password untuk masuk ke dalam sebuah aplikasi [8]. User yang telah login (terautentikasi) kini telah mendapatkan string kode unik berbentuk Json Web Token (JWT) yang tersimpan dalam storage browser sehingga nantinya digunakan untuk dapat mengakses data layanan pada server (Authorization/Otorisasi). API akan melakukan pemeriksaan pada JWT yang dibawa User setiap kali melakukan request ke server[8]. Sehingga dapat dikatakan bahwa Authentication merupakan proses verifikasi identitas, sedangkan Authorization merupakan proses verifikasi hak akses, pada Authorization biasanya menggunakan data peran, izin atau kebijakan akses.

Sangat penting sekali untuk melakukan otorisasi pada aplikasi berbasis web yang menggunakan arsitektur REST API karena sudah terdapat kasus nyata seperti pada twitter yang pernah mengalami kebocoran data yang melibatkan jutaan akun pengguna, kebocoran data ini disebabkan oleh kerentanan Broken Object Level Authorization (BOLA) yang memungkinkan penyerang untuk mengakses data profil pengguna, termasuk alamat email dan nomor telepon. Serta Kerentanan pada API Uber yang memungkinkan akses tidak sah ke data pengguna lain. Ada juga serangan kepada aplikasi web populer, serangan ini disebabkan oleh kerentanan BOLA yang memungkinkan penyerang untuk mengubah ID objek dan mengakses data yang tidak diizinkan. Juga serangan kepada sebuah sistem ERP populer yang memungkinkan penyerang untuk mengakses data keuangan perusahaan. Serangan ini disebabkan oleh kerentanan BOLA yang memungkinkan penyerang untuk mengubah ID objek dan mengakses data yang tidak diizinkan.

2.2. Json Web Token (JWT)

Json Web Token (JWT) adalah sebuah token terenkripsi berbentuk string JSON yang digunakan untuk autentikasi sistem serta pertukaran informasi[9]. JWT biasanya dikirim melalui Header HTTP. Isi dari token JWT pada umumnya berupa informasi data User yang dibutuhkan, sehingga tidak perlu melakukan query berulang ke dalam database. Struktur JWT terdiri atas tiga bagian yaitu header, payload dan signature dan dipisah dengan tanda titik[10]. Dibawah ini gambar alur login SILAB sehingga dibuatkan JWT oleh API SIMAT.

https://doi.org/10.33998/processor.2024.19.2.1752
Submitted: 01 Juni 2024; Reviewed Round#1: 27 Juni 2024; Reviewed Roung#2:03 Juli 2024 ; Reviewed Roung #3: 29 September 2024; Accepted; 08 Oktober 2024; Published: 31 Oktober 2024



Gambar 2. Alur Login SILAB

Saat user menginputkan username dan password di SILAB, username dan password akan di cek ke API SIMAT. Jika username dan password benar maka akan dibuatkan JWT yang berupa access_token dan refresh_token sehingga user bisa masuk ke dashboard aplikasi SILAB dan mengakses menu/API yang ada pada SILAB.

2.3. Postman

Postman merupakan sebuah tools yang digunakan untuk melakukan testing program REST API. Fungsi utama Postman adalah sebagai Testing API, GUI API Caller, serta Monitoring API[11].

2.4. Front end dan Back End

Dalam pembuatan aplikasi berbasis web dikenal dua istilah yaitu Back end dan Front end. Front end menyediakan GUI website yang menarik dan interaktif guna memudahkan user dalam berinteraksi dengan aplikasi berbasis web sekaligus meningkatkan pengalaman pengguna, sedangkan Back End berkaitan dengan suplai data, berkaitan dengan internal sistem, parsing data, pemrosesan data, validasi data maupun penanganan request [12].

2.5. Penelitian Sebelumnya

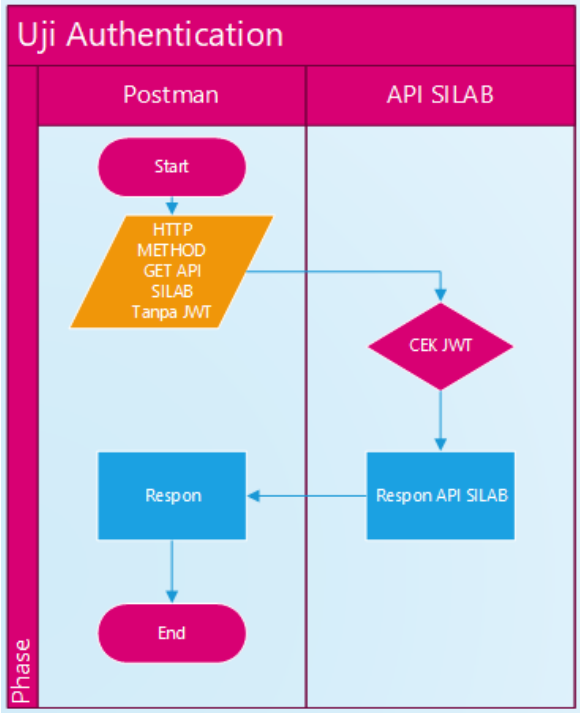
Penelitian yang dilakukan oleh [3] menyampaikan bahwa dengan adanya token, layanan web tidak bisa diakses oleh sembarang User. User memiliki autentikasi untuk mengakses layanan yang ada dengan memakai Token di JWT dan disertakan enkripsi menggunakan metode AES agar penyusup tidak dapat melihat data secara langsung. Penelitian [4] menerapkan sebuah pengamanan terhadap aplikasi menggunakan JWT algoritma HMAC-SHA 512 untuk pengamanan 3 komponen header, payload dan signature sehingga data yang dikirim ke server menjadi terenkripsi. Penelitian [8] menciptakan sebuah rancangan metode autentikasi dan otorisasi sederhana untuk memastikan keaslian dari kode rahasia yang dimiliki client, yang dapat digunakan sebagai alternatif kemanan pada arsitektur microservice. Penelitian [10] menjelaskan bahwa penerapan algoritma HMAC SHA-512 pada JWT dalam web service dan pada arsitektur 64-bit menghasilkan kinerja yang lebih baik. SHA-512 lebih cepat 1% dibandingkan dengan SHA-256 dan tools yang digunakan adalah Postman. Penelitian [11] menginformasikan bahwa penggunaan autentifikasi json web token pada REST API membuat aplikasi menjadi lebih aman karena aplikasi tidak dapat diakses jika tidak menggunakan token. Penelitian [13] menunjukkan bahwa pengujian sistem informasi perpustakaan dengan metode Black Box Testing dapat menemukan sebuah error pada form yang hasil outputnya tidak sesuai dengan yang diharapkan.

https://doi.org/10.33998/processor.2024.19.2.1752
Submitted: 01 Juni 2024; Reviewed Round#1: 27 Juni 2024; Reviewed Roung#2:03 Juli 2024 ; Reviewed Roung #3: 29 September 2024; Accepted; 08 Oktober 2024; Published: 31 Oktober 2024

3. METODOLOGI PENELITIAN

Metode Black Box Testing adalah metode pengujian perangkat lunak di mana penguji mengevaluasi fungsionalitas aplikasi tanpa mengetahui struktur internal, desain, atau kode dari perangkat lunak tersebut. Penguji berfokus pada input yang diberikan ke perangkat lunak dan output yang dihasilkan, serta memastikan bahwa aplikasi berfungsi sesuai dengan spesifikasi yang telah ditentukan. Dengan menggunakan metode ini, Penguji tidak perlu mengetahui bagaimana perangkat lunak diimplementasikan atau bagaimana kode di dalamnya bekerja, penguji memeriksa apakah perangkat lunak melakukan apa yang seharusnya dilakukan berdasarkan fungsionalitas yang diharapkan, Penguji juga dapat memberikan berbagai input ke aplikasi dan memeriksa output yang dihasilkan untuk memastikan kesesuaiannya dengan yang diharapkan[13].
Pada penelitian ini akan dilakukan uji keamanan pada Request URL Back End Aplikasi, baik yang berkaitan dengan Authenticatioan maupun Authorization menggunakan metode Black Box Testing dengan bantuan tools Postman, beberapa penelitian yang pernah menggunakan Postman dalam pengujian keamanan API[14][15]. Request URL Back End Aplikasi akan kita uji apakah data dari Back End Aplikasi melakukan apa yang seharusnya dilakukan berdasarkan fungsionalitas yang diharapkan seperti

1. Authentication pada API SILAB yaitu apakah API pada SILAB harus login terlebih dahulu agar dapat di akses.
   Dari gambar 1 diatas, dapat dilihat bahwa setelah user berhasil login pada aplikasi SILAB, user dapat melihat endpoint API SILAB. Dari endpoint yang sudah didapat, akan dilakukan uji coba dengan mengakses endpoint tersebut melalui POSTMAN dengan skenario sebagai berikut

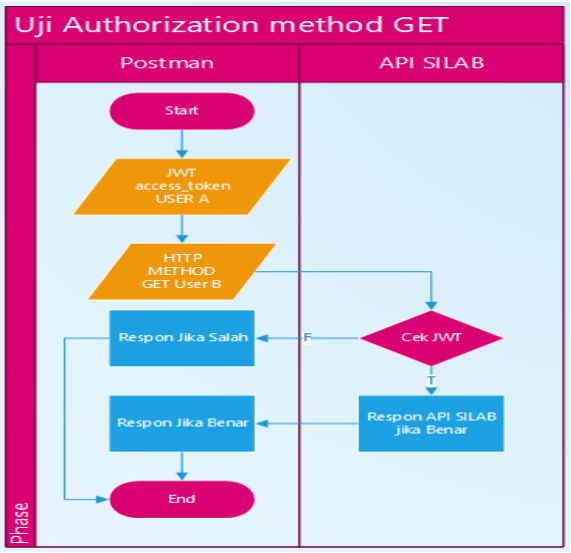


Gambar 3. Skenario uji authentication

API SILAB akan langsung diakses melalui Postman tanpa menyertakan JWT dan melihat respon seperti apa yang akan diberikan oleh API SILAB.

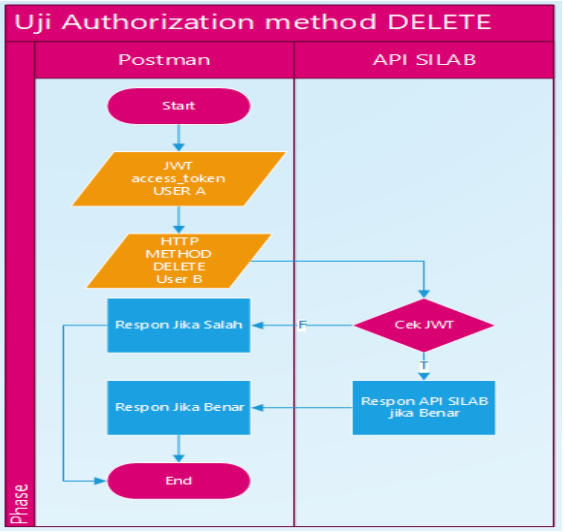
2. Authorization pada HTTP Method GET yaitu apakah API pada SILAB milik user A dapat dilihat oleh user B
   Pada pengujian ini, JWT dari user A yang sudah diperoleh setelah berhasil login, akan kita ambil dan diletakkan pada Postman. Kemudian akan dilakukan uji coba dengan mengubah id yang berbentuk integer pada API SILAB.

https://doi.org/10.33998/processor.2024.19.2.1752
Submitted: 01 Juni 2024; Reviewed Round#1: 27 Juni 2024; Reviewed Roung#2:03 Juli 2024 ; Reviewed Roung #3: 29 September 2024; Accepted; 08 Oktober 2024; Published: 31 Oktober 2024



Gambar 4. Skenario uji authorization method GET

3. Authorization pada HTTP Method Delete yaitu apakah user A dapat menghapus data milik user B
Pada pengujian ini, JWT dari user A yang sudah diperoleh setelah berhasil login, akan kita ambil dan diletakkan pada Postman. Kemudian akan dilakukan uji coba hapus data dengan mengubah id yang berbentuk integer pada API SILAB.



Gambar 5. Skenario uji authorization method DELETE

4. HASIL DAN PEMBAHASAN

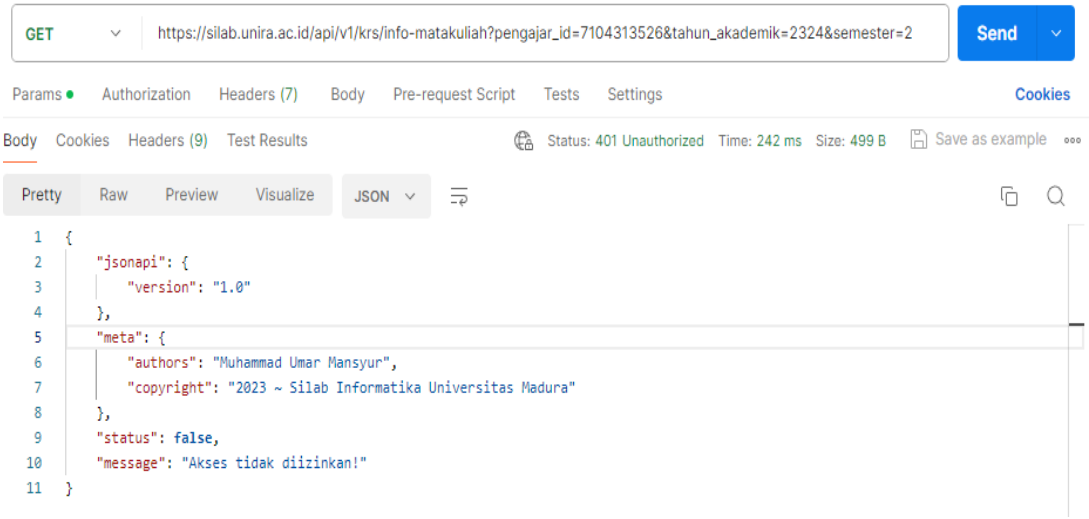
4.1. Hasil dan Analisis

Pengujian telah dilakukan dengan skenario menggunakan akun salah satu dosen praktikum, kita misalkan user A. User A akan login ke akun SILAB, setelah berhasil login kemudian akan mencari letak JWT di storage browser dan ditemukan pada bagian Session Storage dengan nama access_token dan refresh_token, kita ambil nilai dari access_token. Kemudian kita buka aplikasi Postman.

1. Menguji Authentication pada API SILAB yaitu apakah API pada SILAB harus login terlebih dahulu agar dapat di akses.

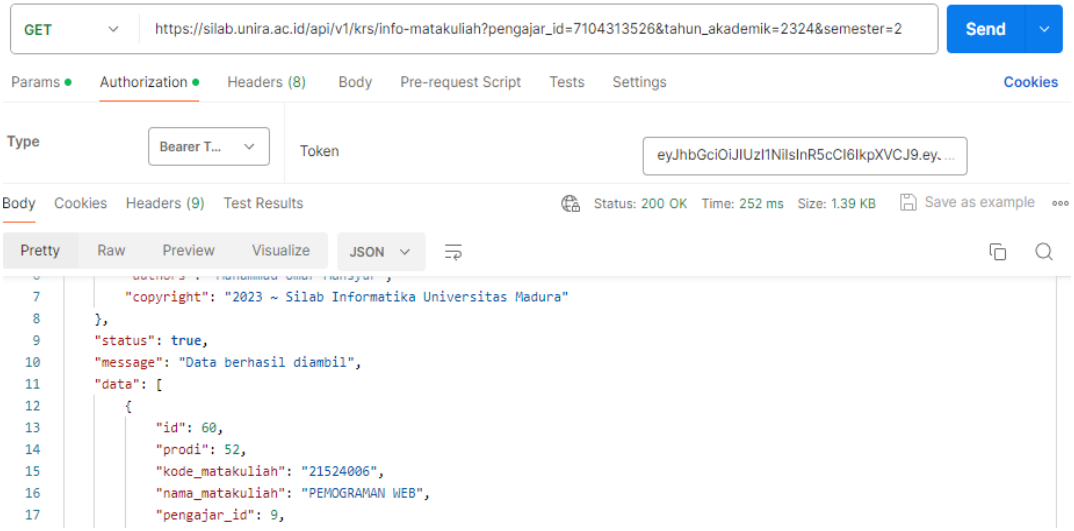
https://doi.org/10.33998/processor.2024.19.2.1752
Submitted: 01 Juni 2024; Reviewed Round#1: 27 Juni 2024; Reviewed Roung#2:03 Juli 2024 ; Reviewed Roung #3: 29 September 2024; Accepted; 08 Oktober 2024; Published: 31 Oktober 2024

Saat mengakses salah satu API SILAB menggunakan Postman tanpa JWT maka data tidak dapat dilihat dengan pesan “akses tidak diizinkan”. Berikut tampilan layar pada aplikasi Postman



Gambar 6. Postman Respon Status 401 Unauthorized

Terlihat jelas respon yang diberikan berupa “status:false” dengan “message: Akses tidak diizinkan”. Ini menunjukkan bahwa untuk mengakses API SILAB harus menyertakan JWT yang valid.
Kemudian, penguji akan mengakses API yang sama, namun dengan menyertakan JWT yang sudah diperoleh dari access_token. Dengan cara memilih tab Authorization pada Postman, pilih type Bearer Token dan isikan nilai dari access_token pada bagian inputan Token. Klik Send. Dan Data berhasil ditampilkan. Berikut hasil tangkapan layar pada aplikasi Postman.

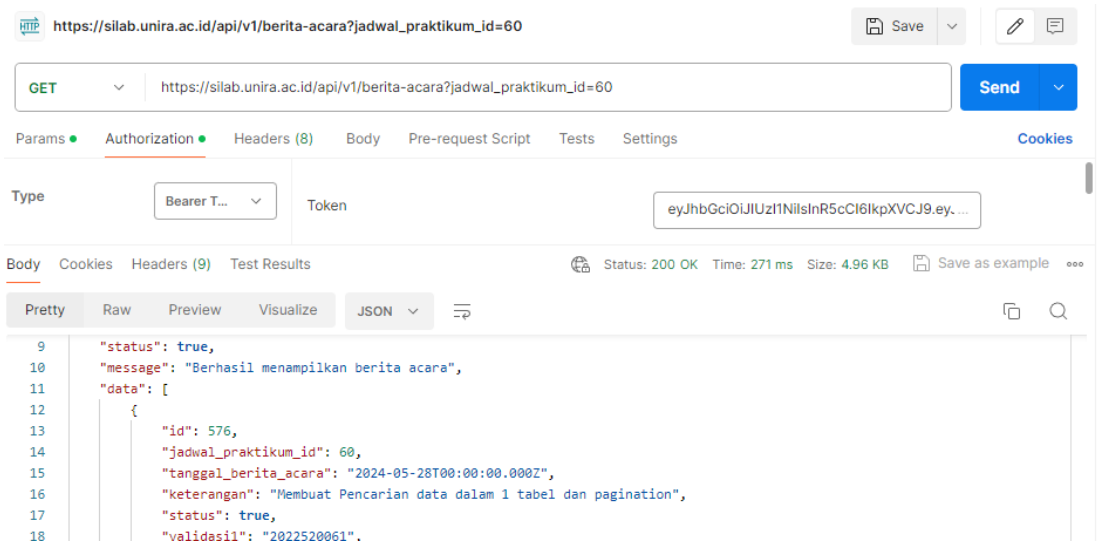


Gambar 7. Postman Respon Status 200 OK

Terlihat jelas respon yang diberikan berupa “status:true” dengan “message:data berhasil diambil” serta mengirimkan “data” yang berupa kumpulan *array of object* yang terdiri dari id, prodi, kode_matakuliah, dan seterusnya.
Ini sangat bagus, karena memang untuk menampilkan data tertentu harus melakukan Authentication terlebih dahulu menggunakan JWT yang sudah dibuat[16].

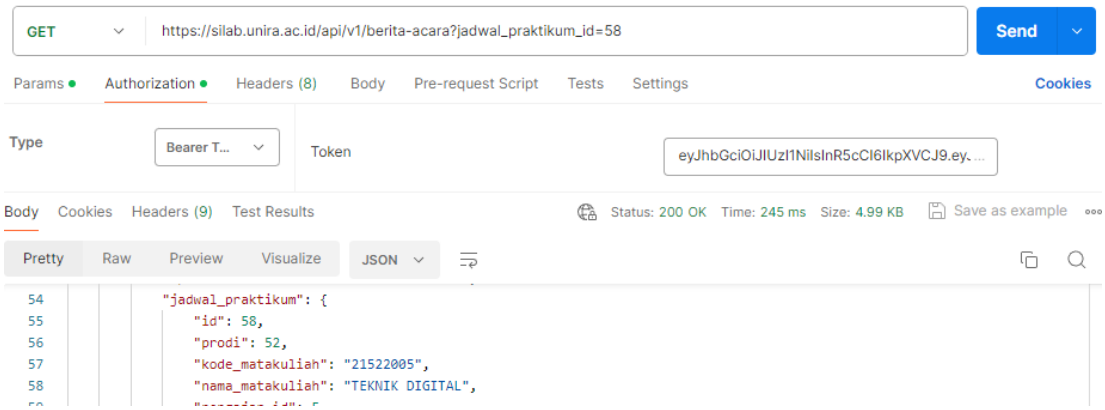
https://doi.org/10.33998/processor.2024.19.2.1752
Submitted: 01 Juni 2024; Reviewed Round#1: 27 Juni 2024; Reviewed Roung#2:03 Juli 2024 ; Reviewed Roung #3: 29 September 2024; Accepted; 08 Oktober 2024; Published: 31 Oktober 2024

2. Menguji Authorization pada HTTP Method GET yaitu apakah API SILAB milik user A dapat dilihat oleh user B
Penguji mencari API SILAB pada menu jadwal praktikum, kemudian mengambil API yang diberikan dan meletakkan pada Postman dengan masih menggunakan JWT yang sama. Dan berikut hasil tangkapan layar pada Postman.



Gambar 8. Berhasil Menampilkan Data Milik User A

API pada jadwal praktikum ini mengirimkan sebuah method GET di URL dengan nama jadwal_praktikum_id dan memiliki nilai yang berupa integer. Kemudian penguji melakukan perubahan pada nilai tersebut, dan berhasil ditampilkan data milik user lain atau data milik dosen lain atau data jadwal praktikum milik matakuliah lain. Seperti pada tangkapan layar Postman dibawah ini.

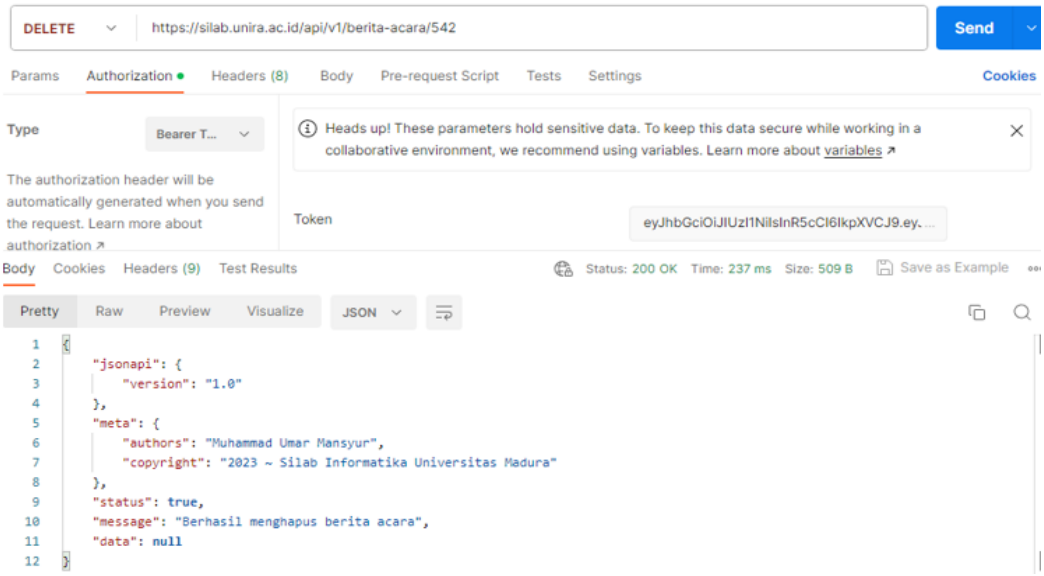


Gambar 9. JWT dari User A Berhasil Menampilkan Data Milik User B

Jika data yang ditampilkan tidak bersifat rahasia, hal seperti ini dapat dimaklumi mengingat API memang dirancang agar dapat berkomunikasi antar sistem dengan mudah. Namun jika data bersifat pribadi, maka hal seperti ini tidak boleh dilakukan oleh sebuah sistem.
Dari sisi pengujian secara Black Box Testing ini, hal ini tetap dikatakan tidak boleh, dikarenakan data yang diinput oleh user B dapat dilihat oleh user A, padahal pada bagian Front end pun hanya menampilkan data jadwal praktikum milik dosen yang sedang login dan tidak menampilkan data jadwal praktikum milik dosen lain.

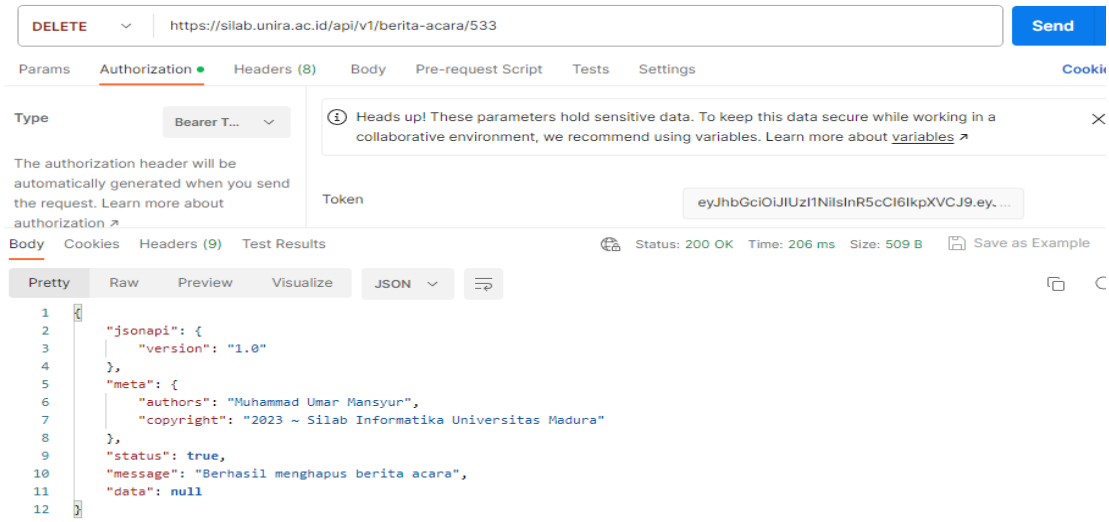
https://doi.org/10.33998/processor.2024.19.2.1752
Submitted: 01 Juni 2024; Reviewed Round#1: 27 Juni 2024; Reviewed Roung#2:03 Juli 2024 ; Reviewed Roung #3: 29 September 2024; Accepted; 08 Oktober 2024; Published: 31 Oktober 2024

3. Menguji Authorization pada HTTP Method Delete yaitu apakah user A dapat menghapus data milik user B Untuk mengimplementasikan pengujian ini, penguji mencari terlebih dahulu fitur delete, dan tersedia pada tombol delete atau hapus berita acara praktikum. Kemudian mengambil Request URL nya, dengan masih menggunakan akun user A dan data berhasil di hapus. Berikut tangkapan layar pada aplikasi Postman.



Gambar 10. JWT dari User A Berhasil Menghapus Data Miliknya Sendiri

API pada hapus berita acara praktikum ini mengirimkan sebuah identifier di URL berupa nilai bertipe integer. Kemudian penguji melakukan perubahan pada nilai tersebut, dan berhasil menghapus data milik user lain atau data milik dosen lain atau data berita acara praktikum milik matakuliah lain, seperti yang terlihat pada tangkapan layar Postman di bawah ini.



Gambar 11. JWT dari User A Berhasil Menghapus Data Milik User B

Penghapusan data milik user lain dapat dilakukan dengan mudah hanya dengan mengubah nilai dari parameter identifier yang digunakan untuk menghapus data karena nilainya bertipe integer auto increment,

https://doi.org/10.33998/processor.2024.19.2.1752
Submitted: 01 Juni 2024; Reviewed Round#1: 27 Juni 2024; Reviewed Roung#2:03 Juli 2024 ; Reviewed Roung #3: 29 September 2024; Accepted; 08 Oktober 2024; Published: 31 Oktober 2024

oleh sebab itu disarankan menggunakan identifier yang terenkripsi[17] atau menggunakan id unik panjang yang susah ditebak.
Dari sisi pengujian secara Black Box Testing, hal seperti ini sangat tidak boleh, dimana user A dapat menghapus data milik user B atau user lain.

Dari hasil pengujian dapat dikatakan bahwa untuk authentication API SILAB, sudah bekerja sangat baik, dimana user harus login terlebih dahulu untuk dapat mengakses API SILAB. Namun untuk Authorization, masih ditemukan ketidaksesuain, dimana masih terdapat data milik user A dapat dilihat oleh user B atau user yang lain. Dan yang paling fatal adalah ketika user A dapat menghapus data milik user B atau user yang lain melalui Postman. Permasalahan seperti ini bisa juga terjadi kepada aplikasi berbasis website lainnya, sehingga penulis memberi rekomendasi antara lain:

1) Keamanan sebuah aplikasi berbasis website, tidak hanya masalah seseorang dapat mencuri username dan pasword sehingga orang tersebut dapat masuk ke dalam aplikasi berbasis website, namun seseorang yang sudah memiliki username dan password pun juga dapat melakukan tindakan yang tidak diinginkan, sehingga sebuah sistem harus benar-benar dibangun dengan keamanan yang baik. Apalagi jika arsitektur yang digunakan menggunakan REST API. Dimana ketika JWT sudah dibuat dan belum *expired*, maka JWT tersebut dapat digunakan untuk mengakses endpoint atau request URL ke server (Back end) dari luar aplikasi (luar Front end) dengan bantuan tools seperti Postman.
2) Membuat enkripsi atau generator id yang unik yang tidak mudah ditebak pada data yang dikirim melalui method GET, jangan menampilkan data apa adanya seperti data unik yang bersifat integer auto increment, karena user dapat dengan mudah mengubah nilai yang diberikan.
3) Jika data tersebut bersifat rahasia, maka pastikan hanya user yang memiliki data tersebut yang dapat melihat.
4) Jangan menampilkan sebuah id unik pada saat HTTP Method GET seperti id berita acara praktikum pada kasus ini, yang juga digunakan sebagai identifier untuk menghapus data, sehingga user lain akan dengan mudah mengambil data itu dan menggunakan data itu pada HTTP Method Delete.
5) Pastikan, hanya admin atau user pembuat data yang dapat menghapus data tersebut.
6) Gunakan CSRF Token supaya tidak dapat melakukan penghapusan data dari luar aplikasi (Front end), namun hal ini lebih cocok jika arsitektur yang digunakan adalah monolitik, bukan REST API.
7) Gunakan API Key sebagai metode sederhana untuk mengidentifikasi aplikasi yang mengakses API. Setiap klien diberi kunci unik yang harus disertakan dalam setiap permintaan API.
8) Atur kebijakan Cross-Origin Resource Sharing (CORS) untuk mengontrol dari mana permintaan API dapat berasal. Hal ini penting dilakukan agar dapat mencegah permintaan yang tidak sah dari domain lain.
9) Pantau dan catat semua aktivitas API untuk mendeteksi dan merespons aktivitas mencurigakan atau tidak sah dengan menggunakan alat pemantauan keamanan API untuk analisis secara real-time.
10) Jika diperlukan, implementasikan protokol keamanan yang lebih canggih seperti Mutual TLS (mTLS), yang membutuhkan verifikasi identitas dua arah antara klien dan server.

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Pengujian keamanan pada Back end Aplikasi berbasis website menggunakan metode Black Box Testing dengan studi kasus aplikasi SILAB telah selesai dilakukan. Keamanan sebuah aplikasi berbasis website perlu dibuat tidak hanya kepada user yang belum melakukan autentikasi, tapi juga kepada user yang sudah memiliki autorisasi, terutama aplikasi dengan arsitektur REST API karena ketika JWT sudah dibuat dan belum expired, maka JWT tersebut dapat digunakan untuk mengakses endpoint atau request URL ke server (Back end) dari luar aplikasi (luar Front end) dengan bantuan tools seperti Postman. Melindungi API dalam arsitektur REST API memerlukan kombinasi beberapa teknik keamanan. Mulai dari autentikasi yang kuat, lanjutkan dengan autorisasi yang ketat, gunakan id unik yang sulit ditebak dan pastikan komunikasi dienkripsi. Selain itu, dapat juga digunakan API key, CORS dan pemantauan untuk meningkatkan keamanan API. Dengan menerapkan langkah-langkah ini, Aplikasi yang dibangun dapat terlindungi dari akses yang tidak sah dan potensi ancaman keamanan

https://doi.org/10.33998/processor.2024.19.2.1752
Submitted: 01 Juni 2024; Reviewed Round#1: 27 Juni 2024; Reviewed Roung#2:03 Juli 2024 ; Reviewed Roung #3: 29 September 2024; Accepted; 08 Oktober 2024; Published: 31 Oktober 2024

lainnya. Pengujian dapat dilakukan dengan metode Black Box Testing, lakukan pengujian sebelum aplikasi digunakan agar tidak menyulitkan saat melakukan perbaikan.

5.2. Saran

Pengujian menggunakan metode Black Box Testing bukanlah satu-satunya cara untuk menguji keamanan Back end aplikasi berbasis website, masih banyak cara lain untuk melakukan pengujian. Segera lakukan pengujian keamanan aplikasi saat pengembangan aplikasi (*development*), jangan menunggu aplikasi digunakan terlebih dahulu (*production*) karena terlalu fokus dengan fitur-fitur yang menarik. Langkah-langkah yang dapat diambil untuk memperbaiki kelemahan keamanan yang ditemukan pada penelitian ini antara lain gunakan teknik enkripsi yang lebih kuat untuk identifier, seperti UUID, dan pastikan setiap request API dilengkapi dengan CSRF token, serta pastikan hanya user dengan otorisasi yang bisa mengakses maupun menghapus data. Adapun untuk pengamanan aplikasi berbasis web dengan arsitektur REST API antara lain dengan memastikan setiap permintaan diverifikasi dan divalidasi untuk memastikan pengguna memiliki izin yang tepat, terapkan pemeriksaan otorisasi yang ketat untuk setiap endpoint dan fungsi, gunakan kontrol akses berbasis peran untuk membatasi tindakan yang dapat dilakukan oleh setiap peran, lakukan logging dan audit terhadap semua tindakan yang melibatkan data sensitif dan otorisasi, serta secara rutin lakukan pengujian penetrasi dan review kode untuk mengidentifikasi dan memperbaiki kelemahan otorisasi.

DAFTAR PUSTAKA

[1] A. W. Syahroni, M. T. Supratman, N. Ramadhani, B. Said, and N. P. Dewi, "Analisis Sentimen Komentar Mahasiswa Terhadap Dosen Pengampu Matakuliah pada Aplikasi SIMAT," *J. Process.*, vol. 18, no. 2, pp. 209–217, 2023.

[2] S. Steven, W. Rifaldi, and U. Nugraha, "Strategi Pengamanan Front-end dalam Pengembangan Website," *JUSTINFO | J. Sist. Inf. dan Teknol. Inf.*, vol. 1, no. 1, pp. 42–53, 2023.

[3] R. C. Saputra and Y. D. Setianto, "Web Service Security System Analysis with Rest Architecture Using The Aes Method with JWT," *Praxis (Bern. 1994).*, vol. 3, no. 1, p. 76, 2020.

[4] J. S. Utama and A. D. Indriyanti, "Pengamanan Restful API Web Service Menggunakan Json Web Token (Studi Kasus: Aplikasi Siakadu Mobile Unesa)," *J. Emerg. Inf. ...*, vol. 04, no. 01, pp. 8–17, 2023.

[5] D. V. Kornienko, S. V. Mishina, and M. O. Melnikov, "The Single Page Application architecture when developing secure Web services," *J. Phys. Conf. Ser.*, vol. 2091, no. 1, pp. 0–12, 2021.

[6] A. Bastian, H. Sujadi, and L. Abror, "Analisis Keamanan Aplikasi Data Pokok Pendidikan (Dapodik) Menggunakan Penetration Testing Dan Sql Injection," *INFOTECH J.*, vol. 6, no. 2, pp. 65–70, 2020.

[7] R. Haddad and R. E. Malki, "OpenAPI Specification Extended Security Scheme: A method to reduce the prevalence of Broken Object Level Authorization," *arXiv Prepr. arXiv2212.06606*, no. Idl, pp. 1–10, 2022.

[8] I. G. N. Ady Kusuma, "Perancangan Simple Stateless Autentikasi Dan Otorisasi Layanan Rest-Api Berbasis Protokol Http," *J. Manaj. Inform. dan Sist. Inf.*, vol. 4, no. 1, p. 78, 2021.

[9] P. Painem and H. Soetanto, "Sistem Presensi Pegawai Berbasis Web Service Menggunakan Metode Restfull Dengan Keamanan JWT Dan Algoritma Haversine," *Fountain Informatics J.*, vol. 5, no. 3, p. 6, 2020.

[10] A. Rahmatulloh, H. Sulastri, and R. Nugroho, "Keamanan RESTful Web Service Menggunakan JSON Web Token (JWT) HMAC SHA-512," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 7, no. 2, 2018.

[11] E. Edy, F. Ferdiansyah, W. Pramusinto, and S. Waluyo, "Pengamanan Restful API menggunakan JWT untuk Aplikasi Sales Order," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 3, no. 2, pp. 106–112, 2019.

[12] I. P. A. Eka Pratama, "Pengujian Performansi Lima Back-End JavaScript Framework Menggunakan Metode GET dan POST," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 6, 2020.

[13] Uminingsih, M. Nur Ichsanudin, M. Yusuf, and S. Suraya, "Pengujian Fungsional Perangkat Lunak Sistem Informasi Perpustakaan Dengan Metode Black Box Testing Bagi Pemula," *STORAGE J. Ilm. Tek. dan Ilmu Komput.*, vol. 1, no. 2, pp. 1–8, 2022.

[14] T. A. Nugroho *et al.*, "Keamanan Berbasis Service Oriented Architecture Menggunakan Oauth 2.0 dan Json Web Token," *IJESPG J.*, vol. 1, no. 3, pp. 229–236, 2023.

[15] H. Prasetyo and U. Nugraha, "Optimasi Keamanan dalam Pengembangan Aplikasi Menggunakan Metode Agile Scrum dan JSON Web Token," *JUSTINFO | J. Sist. Inf. dan Teknol. Inf.*, vol. 1, no. 1, pp.

https://doi.org/10.33998/processor.2024.19.2.1752
Submitted: 01 Juni 2024; Reviewed Round#1: 27 Juni 2024; Reviewed Roung#2:03 Juli 2024 ; Reviewed Roung #3: 29 September 2024; Accepted; 08 Oktober 2024; Published: 31 Oktober 2024

34–41, 2023.
[16] A. Umarjati and A. Wibowo, “Implementasi JWT pada Aplikasi Presensi dengan Validasi Fingerprint ,” *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 1, no. 10, pp. 1085–1091, 2021.
[17] E. Gunadhi and A. P. Nugraha, “Penerapan Kriptografi Base64 Untuk Keamanan URL (Uniform Resource Locator) Website Dari Serangan SQL Injection,” *J. Algoritm.*, vol. 13, no. 2, pp. 391–398, 2017.

https://doi.org/10.33998/processor.2024.19.2.1752
Submitted: 01 Juni 2024; Reviewed Round#1: 27 Juni 2024; Reviewed Roung#2:03 Juli 2024 ; Reviewed Roung #3: 29 September 2024; Accepted; 08 Oktober 2024; Published: 31 Oktober 2024