Implementasi Dashboard Monitoring untuk Pengujian Kerentanan SQL Injection pada Environment GitLab

Muhammad Fahmi Al Azhar[1], Ruki Harwahyu[2]
[1,2]Departemen Teknik Elektro, Universitas Indonesia
E-mail: [1]**muhammad.fahmi14@ui.ac.id**, [2]ruki.h@ui.ac.id

Abstrak

SQL Injection masih menjadi salah satu jenis kerentanan yang paling sering ditemukan pada aplikasi berbasis web. Pengujian terhadap aplikasi sebelum dirilis ke production harus dilakukan semaksimal mungkin agar kerentanan ini tidak muncul saat aplikasi tersebut rilis ke production. Salah satu jenis pengujian yang harus dilakukan adalah Static Application Security Testing (SAST). SAST bekerja dengan cara memindai dan menganalisis seluruh source code di dalam project untuk diperiksa apakah terdapat kesalahan logika dan jenis kerentanan tertentu. Dengan menggunakan platform GitLab, pengujian dapat dilakukan secara otomatis. Namun, hasil dari pengujian SAST tersebut tidak dapat dilihat secara langsung melalui platform GitLab. Berdasarkan kondisi tersebut, maka dibutuhkan aplikasi dashboard monitoring yang dapat diakses oleh tim pengembang dan tim operasional TI. Dengan menggunakan dashboard ini, maka programmer dapat mengetahui bagian source code mana yang mengandung kerentanan SQL Injection. Dashboard ini dibuat dengan menggunakan framework PHP CodeIgniter 4 dan Database MySQL.

Kata Kunci*—dashboard, sql injection, gitlab, static analysis*

1. PENDAHULUAN

Berdasarkan data dari Open Web Application Security Project (OWASP) tahun 2021, SQL Injection masih menjadi top 10 serangan yang terjadi pada aplikasi berbasis web [1]. SQL Injection adalah teknik eksploitasi dengan cara memodifikasi perintah SQL pada form input yang terdapat pada aplikasi sehingga memungkinkan penyerang untuk mengirimkan syntax SQL ke database. SQL Injection juga dapat terjadi dengan memanfaatkan parameter GET di URL pada halaman tertentu dari sebuah aplikasi web. Serangan ini dapat terjadi karena minimnya validasi yang dilakukan dari fom input pada aplikasi. SQL Injection memiliki dampak yang berbahaya karena dapat menyebabkan kebocoran data sehingga menghilangkan kepercayaan publik.

Cara programmer menulis source code menjadi faktor yang menentukan apakah source code tersebut menjadi rentan atau tidak terhadap serangan SQL Injection. Kebanyakan framework PHP telah menyediakan aturan penulisan source code untuk menghindari SQL Injection. Tetapi seringkali karena deadline yang dekat dan kurangnya pengalaman dari programmer kesalahan penulisan source code masih sering terjadi.

Tahap pengujian adalah tahap yang penting dilakukan ketika membuat aplikasi. Seiring dengan perkembangan teknologi aplikasi, saat ini pengujian dapat dilakukan secara otomatis dan terintegrasi dengan platform DevOps. Pengujian yang terintegrasi dan otomatis memudahkan bagi tim pengembang dan tim operasional untuk melakukan tugasnya.

GitLab [2] adalah salah satu platform DevOps yang bersifat gratis dan menyediakan fitur yang lengkap. GitLab menyediakan fitur untuk melakukan automasi di berbagai tahap pada siklus pengembangan software [3]. Salah satu fitur GitLab yang sering digunakan adalah repositori git dan pipeline. Repositori digunakan untuk menyimpan source code dan melakukan versioning code serta untuk berkolaborasi di dalam tim pengembang. Pipeline digunakan untuk mempercepat proses rilis aplikasi tetapi tetap menjamin kualitas dari aplikasi tersebut. Di dalam pipeline

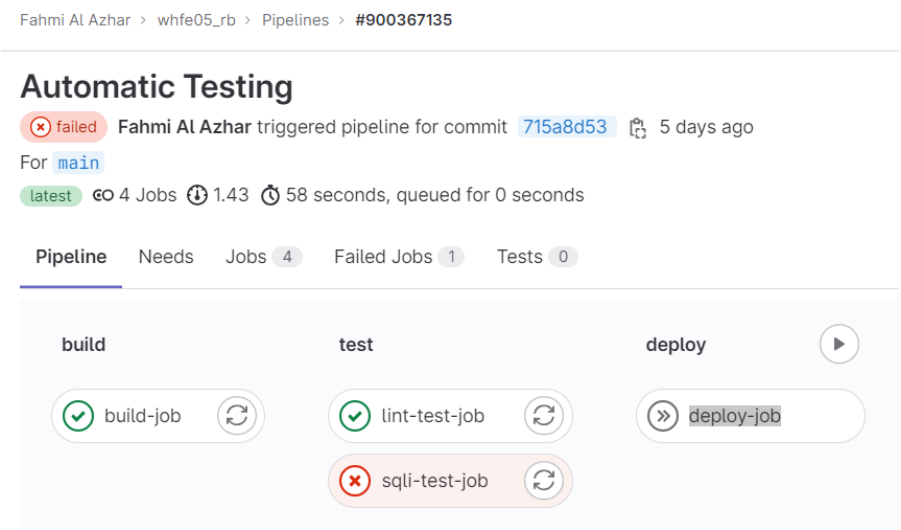
P-ISSN: 2089-676X
E-ISSN: 2549-0796

tersebut dapat didefinisikan beberapa job seperti build source code, unit testing, security testing, bahkan sampai dengan deployment dapat dilakukan secara otomatis. Penelitian ini mengembangkan tools SQL Injection Static Analyzer yang telah dikembangkan penulis pada penelitian sebelumnya. Tools tersebut dapat mendeteksi adanya kerentanan SQL Injection pada framework CodeIgniter 3 menggunakan metode static analysis. Tools ini dibuat khusus untuk diintegrasikan dengan pipeline pada GitLab. Cara kerja dari tools ini adalah dengan mengakses endpoint berikut:

```
/scan/rest?username=[USERNAME GITLAB]&access_token=[ACCESS TOKEN]&url=[URL REPO]
```

Terdapat tiga paramater dalam endpoint tersebut, yaitu username, access_token, dan url. Parameter username adalah username yang ada pada GitLab, sedangkan parameter kedua, yaitu access_token adalah Access Token yang digenerate pada konfigurasi project di GitLab. Parameter ketiga, url, adalah URL repository git dari project yang akan dilakukan pengujian SQL Injection. Endpoint tersebut dapat diakses dengan menggunakan CURL menggunakan request dengan tipe GET. Ketika endpoint itu dipanggil, maka tools tersebut secara otomatis akan melakukan cloning ke server dan melakukan pengujian terhadap kerentanan SQL Injection. Hasil yang didapatkan dari pengujian adalah berupa status PASSED atau FAILED.

Gambar 1 dibawah ini menunjukkan output dari pengujian yang dilakukan secara otomatis di GitLab. Pengujian yang dimaksud pada penelitian ini adalah pada job sqli-test-job, dimana hasil pengujian tersebut adalah Failed.



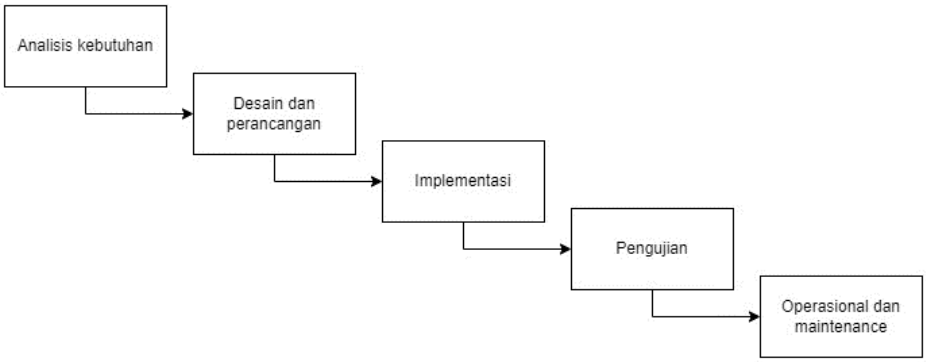
Gambar 1 Hasil pengujian secara otomatis di GitLab

Pengujian SQL Injection secara otomatis telah dapat dilakukan melalui GitLab, tetapi belum ada cara untuk mengetahui bagian source code mana perlu diperbaiki oleh programmer. Hal ini tentu menyulitkan bagi programmer untuk memperbaiki status FAILED dari hasil pengujian tersebut. Melihat kondisi tersebut, maka diperlukanlah sebuah dashboard monitoring yang dapat digunakan untuk memantau setiap riwayat pengujian yang dilakukan di GitLab. Penelitian ini berfokus pada bagaimana cara membuat dashboard yang dapat dengan mudah dimonitor oleh tim pengembang dan operasional IT.

2. METODE PENELITIAN

Metode yang digunakan untuk membuat dashboard monitoring ini adalah dengan menggunakan model Waterfall. Model waterfall melakukan pendekatan pengembangan software secara linear dan sekuensial yang dibagi menjadi beberapa tahapan aktivitas, yaitu *requirement*, *analysis*, *design*, *coding*, dan *testing* [4]. Pressman [5] mengidentifikasi aktivitas yang dilakukan dalam model waterfall terdiri dari: *communication*, *planning*, *modeling*, *construction*, dan *deployment*. Sedangkan Pfleeger and Atlee [6] menjelaskan bahwa waterfall terdiri dari *requirement analysis, system design, program design, coding, unit and integration testing, system testing, acceptance testing* dan *operation and maintenance*.

Dari beberapa penelitian terkait model Waterfall, dapat dirangkum bahwa model Waterfall adalah sebagai berikut:



Gambar 2. Metode Penelitian

Model waterfall adalah model yang berasumsi bahwa dalam proses pembuatan software semua tahapan dapat dilakukan dengan maksimal dan dalam kondisi ideal. Dengan metode ini, akan sulit jika di tengah-tengah proses pengembangan terjadi perubahan kebutuhan. Peneliti memilih metode ini karena penulis berasumsi bahwa terjadinya perubahan kebutuhan selama proses pengembangan software sangat kecil.

2.1. Analisis kebutuhan

Pada tahap pertama dari model Waterfall ini terjadi aktivitas pengumpulan kebutuhan dari sistem yang akan dibangun. Analisis kebutuhan diperlukan untuk mengetahui dengan detail aplikasi seperti apa yang dibutuhkan oleh pengguna. Berdasarkan analisis kebutuhan yang telah dilakukan, didapatkan informasi bahwa dibutuhkan sebuah dashboard monitoring untuk dapat memantau hasil pengujian yang telah dilakukan.

2.2. Desain dan perancangan

Setelah analisis kebutuhan selesai dilakukan, maka tahap selanjutnya adalah tahap desain dan perancangan. Tahap ini bertujuan untuk mengubah data analisis kebutuhan yang dilakukan pada tahap sebelumnya menjadi rancangan-rancangan dalam bentuk diagram alur, rancangan database, rancangan interface serta diagram lain yang diperlukan untuk tahap implementasi nantinya.

2.3. Implementasi

Tahap ini adalah tahapan dimana dilakukannya proses *coding* untuk menerjemahkan rancangan sistem menjadi *software*. Pada tahap ini tidak dilakukan lagi analisis kebutuhan, proses *coding* dilakukan berdasarkan pada rancangan yang telah dibuat sebelumnya.

2.4. Pengujian

Tahap pengujian adalah tahap yang dilakukan setelah software selesai dibuat. Pengujian dilakukan untuk memastikan bahwa software telah berjalan dengan semestinya. Pengujian juga dilakukan untuk mendeteksi adanya kesalahan atau bug yang mungkin terjadi.

2.5. Operasional dan Maintenance

Tahap ini adalah tahap terakhir dari rangkaian pembuatan software menggunakan model Waterfall. Setelah tahap software lulus dari tahap pengujian, maka software tersebut akan di rilis sehingga dapat digunakan secara langsung oleh pengguna. Saat software ini telah rilis, tidak menutup kemungkinan bahwa akan ditemukan kesalahan baru yang tidak terdeteksi pada tahap pengujian. Apabila ternyata ditemukan suatu kesalahan, maka metode ini mengharuskan untuk mengulangi dari awal yaitu dengan melakukan analisis kebutuhan terlebih dahulu.

3. HASIL DAN PEMBAHASAN

3.1. Kebutuhan Hardware dan Software

Pembuatan aplikasi dashboard monitoring SQL Injection Static Analyzer ini memerlukan hardware dan software sebagai berikut:

1. Kebutuhan software:

Untuk membangun aplikasi dashboard ini dibutuhkan software pendukung sebagai berikut:

a. Sistem Operasi Windows 11
b. Visual Studio Code versi terbaru
c. PHP versi 7.2 ke atas
d. Database MySQL versi terbaru
e. TablePlus versi terbaru
f. Browser Google Chrome versi terbaru

2. Kebutuhan hardware

Sedangkan kebutuhan hardware untuk membangun aplikasi dashboard ini adalah laptop dengan spesifikasi sebagai berikut:

a. Processor Intel Core i7 8665U
b. SSD 256GB
c. RAM 16GB
d. VGA Intel UHD Graphics 620

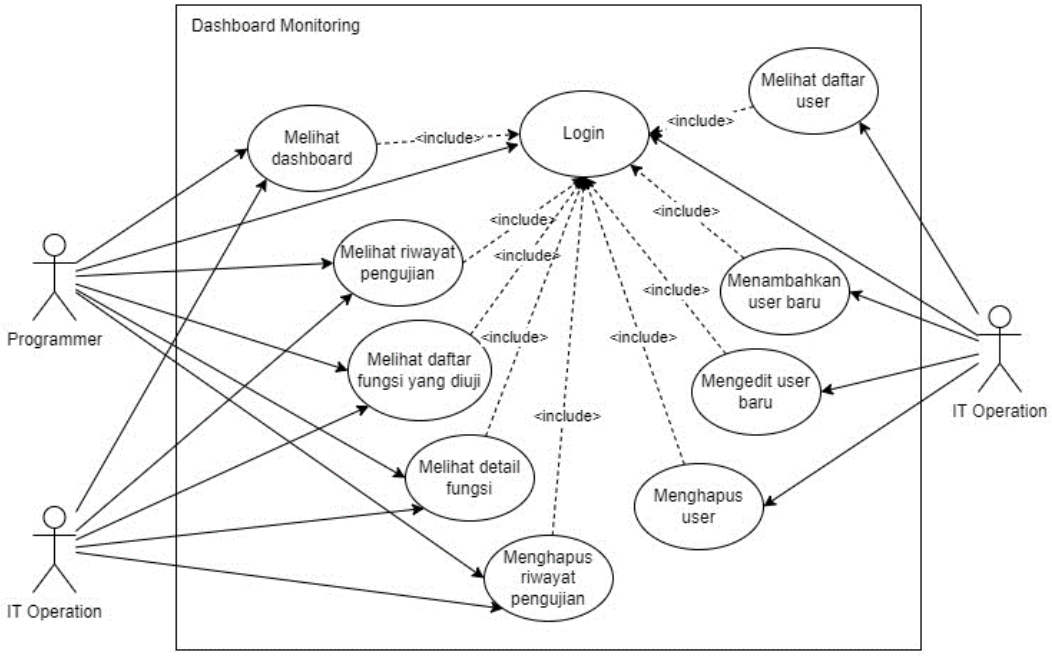
3.2. Desain Sistem

Desain dan pengembangan sebuah dashboard tidak dapat dianggap remeh, karena ada banyak hal yang harus diperhatikan mulai dari data yang ditampilkan sampai dengan pengguna yang akan mengakses dashboard tersebut [7]. Pada bagian ini akan dijelaskan mengenai perancangan sistem yang telah dilakukan. Untuk memudahkan proses implementasi *coding*, maka

digunakan *Unified Modeling Language* (UML). UML merupakan bahasa standar untuk pendokumentasian suatu sistem dan merupakan *blueprint* dari sebuah *software*.

3.2.1. Use Case Diagram

Use Case Diagram adalah salah satu jenis diagram dalam perancangan *software* yang digunakan untuk menggambarkan interaksi antara aktor (pengguna eksternal) dan sistem yang akan dikembangkan. Diagram ini membantu dalam pemahaman yang jelas tentang fitur-fitur utama sistem, interaksi antara pengguna dan sistem, serta tujuan utama yang ingin dicapai. Gambar 3 menunjukkan use case diagram dari aplikasi dashboard monitoring.

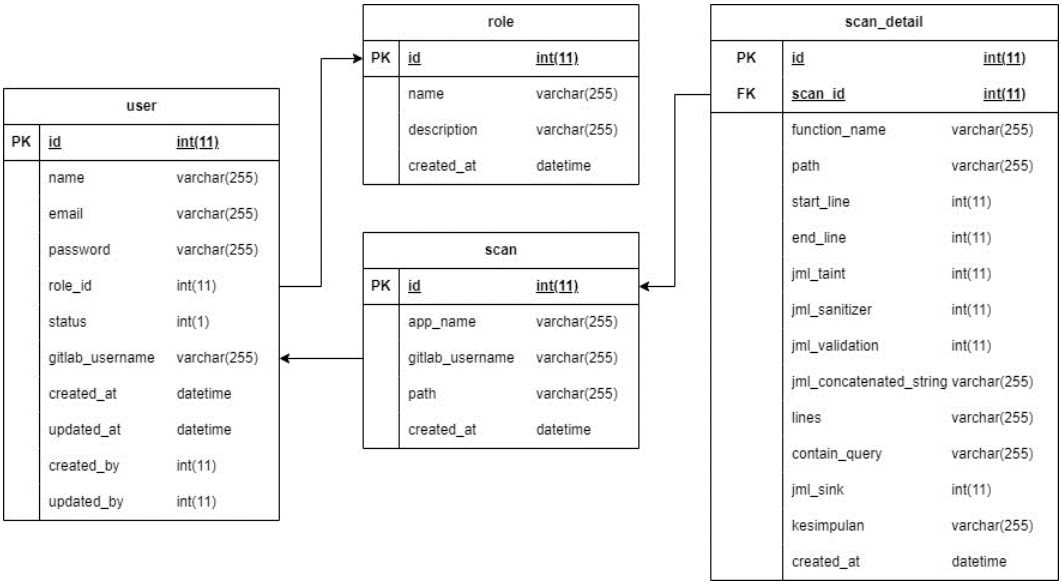


Gambar 3 Use Case Diagram

Dari gambar di atas dapat diketahui bahwa terdapat dua aktor di dalam sistem yang akan dibangun. Aktor pertama adalah pengguna biasa atau programmer yang hanya bertugas untuk memonitor dan mengelola riwayat penguijan miliknya sendiri. Programmer tidak dapat melihat data pengujian milik programmer lain. Aktor kedua adalah IT Operation atau yang bertugas sebagai admin di dalam aplikasi dashboard ini. Admin memiliki wewenang untuk mengelola seluruh data tanpa terkecuali.

3.2.2. Entity Relationship Diagram (ERD)

ERD adalah suatu alat visual yang digunakan untuk menggambarkan hubungan antara entitas dalam suatu sistem atau database. B. Loonam [8]menyebutkan bahwa ERD digunakan untuk memodelkan kebutuhan data dari suatu organisasi, biasanya oleh System Analys dalam tahap analisis sebagai persyaratan dalam pengembangan sistem. Diagram ERD membantu dalam pemodelan struktur data, mengidentifikasi entitas utama, atribut, serta hubungan antar entitas tersebut. Gambar 4 menunjukkan ERD dari aplikasi dashboard monitoring.



Gambar 4 Entity Relationship Diagram

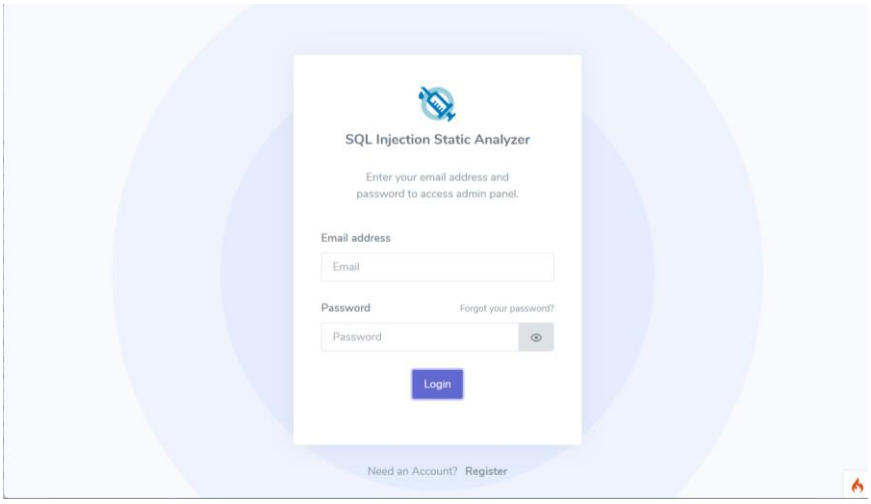
3.3. Implementasi Sistem

Pada bagian ini dijelaskan mengenai tahapan implementasi yang telah dilakukan dalam pembuatan aplikasi dashboard monitoring untuk kerentanan SQL Injection ini. Sistem ini diimplementasikan dengan menggunakan bahasa pemrograman PHP dengan menggunakan framework CodeIgniter 4 [9]. Framework ini dipilih karena dapat mempercepat proses pengembangan aplikasi dengan tetap memperhatikan standar penulisan sehingga akan lebih mudah jika diperlukan *maintenance* ke depannya [10].

3.3.1. Halaman Login

Halaman login adalah halaman yang pertama kali diakses ketika membuka aplikasi dashboard monitoring ini. Pengguna yang ingin mengakses harus merupakan pengguna yang terdaftar di dalam aplikasi. Untuk melakukan login, pengguna harus memasukkan email dan password.

P-ISSN: 2089-676X
E-ISSN: 2549-0796



Gambar 5 Halaman login dashboard

3.3.2. Halaman Beranda/Dashboard

Halaman beranda adalah halaman yang pertama kali diakses ketika pengguna telah berhasil login. Pada halaman beranda ini terdapat informasi berupa:

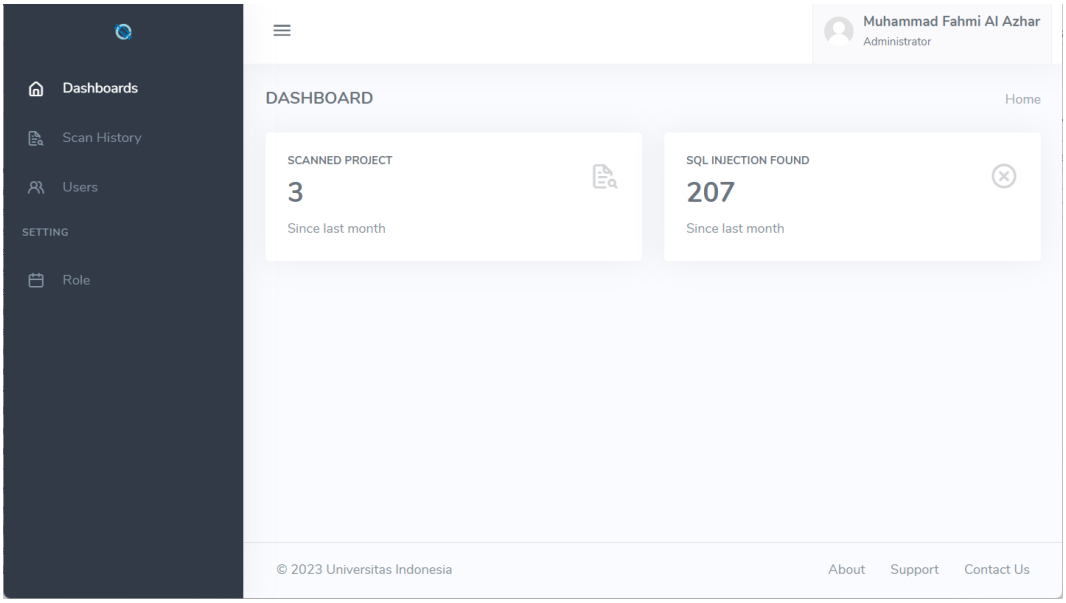
1. Jumlah project yang telah dilakukan pengujian.

   Merupakan jumlah project yang telah dilakukan pengujian. Jumlah yang dihitung pada dashboard ini adalah jumlah project pada bulan yang sedang berjalan.

2. Jumlah SQL Injection yang ditemukan.

   Merupakan jumlah SQL Injection yang ditemukan dari seluruh project yang dilakukan pengujian.

P-ISSN: 2089-676X
E-ISSN: 2549-0796



Gambar 6 Halaman Beranda

3.3.3. Halaman Scan History

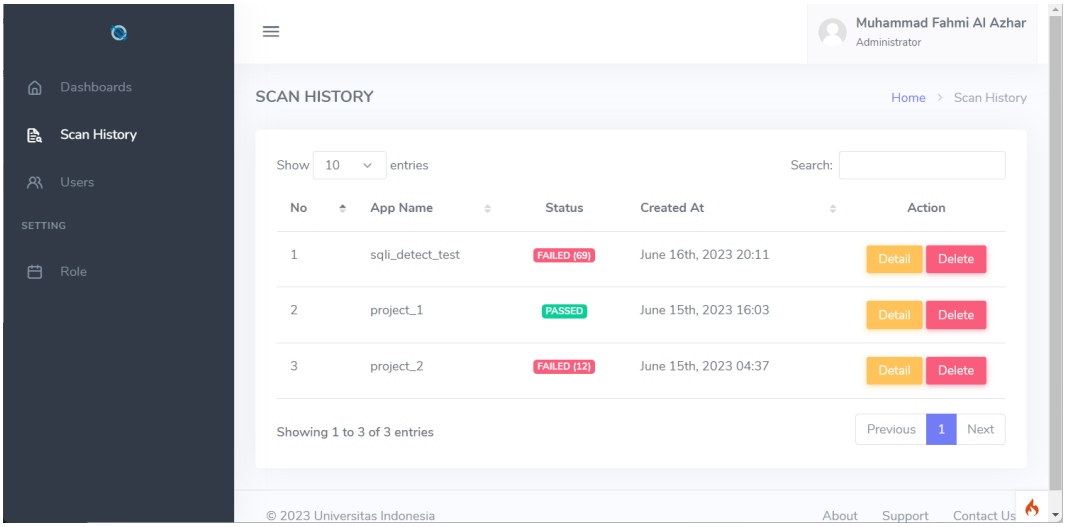
Halaman Scan History adalah halaman yang memuat daftar riwyat pengujian yang pernah dilakukan oleh pengguna terkait. Daftar riwayat yang muncul ditampilkan dalam bentuk tabel yang bisa diurutkan menurut kolom tertentu, dan dilakukan pencarian data dengan menggunakan field filter yang tersedia di sebelah kanan atas tabel. Data yang tampil di halaman ini sudah diurutkan berdasarkan tanggal terbaru dilakukannya pengujian.

Di dalam halaman scan history terdapat beberapa informasi sebagai berikut:

1. App Name, adalah nama aplikasi yang dilakukan pengujian. Nama aplikasi ini diambil secara otomatis dari nama project yang ada di GitLab.
2. Status, adalah status dari pengujian tersebut. Status akan tertulis PASSED jika tidak ditemukan kerentanan SQL Injection di dalam source code yang diuji. Sebaliknya, jika source code mengandung kerentanan SQL Injection, maka status akan tertulis FAILED.
3. Created At, adalah tanggal dilakukannya pengujian. Tanggal ini dihitung dari ketika pipeline di dalam GitLab mulai dijalankan.
4. Aksi, terdiri dari dua tombol yaitu Detail dan Delete. Sesuai dengan Namanya, Detail digunakan untuk melihat detail dari hasil pengujian tersebut, sedangkan Delete digunakan untuk menghapus riwayat pengujian.

Gambar 7 di bawah ini menunjukkan halaman Scan History.

P-ISSN: 2089-676X
E-ISSN: 2549-0796



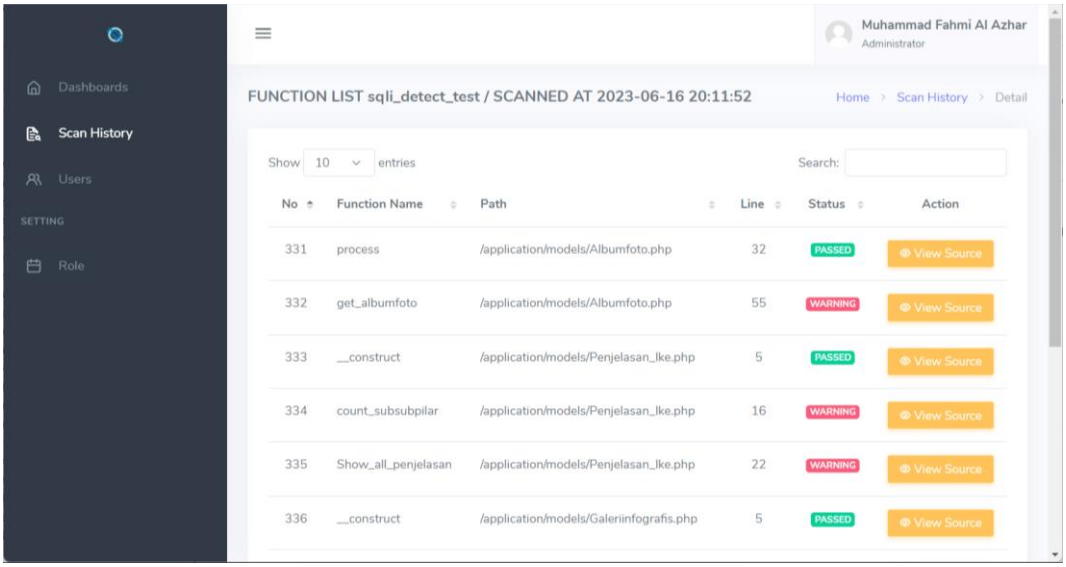
Gambar 7 Halaman Scan History

3.3.4. Halaman Detail Scan History

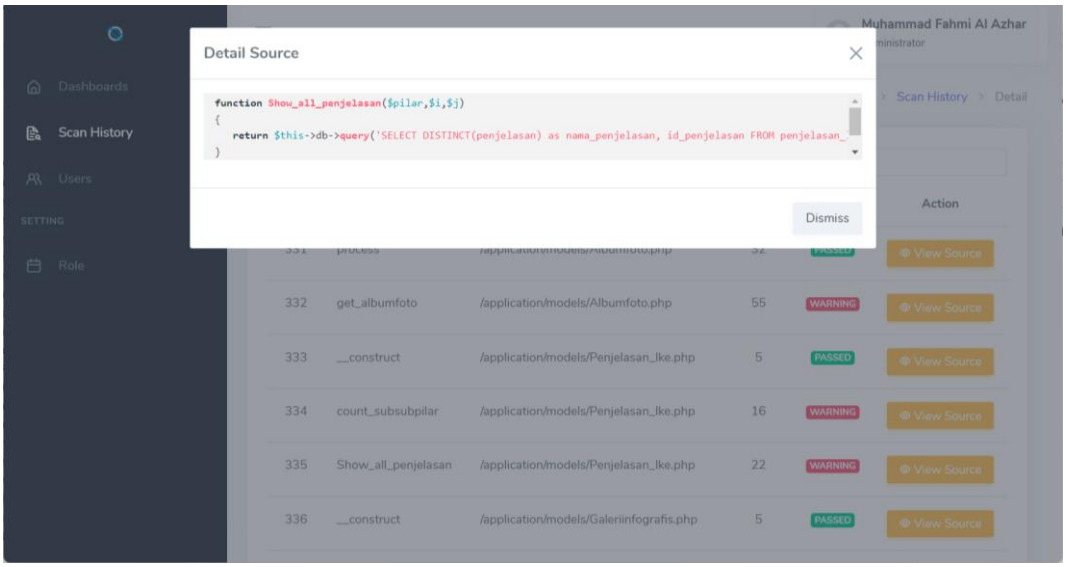
Halaman Detail Scan History adalah halaman yang diakses dengan melakukan klik pada tombol detail di halaman Scan History. Pada halaman ini ditampilkan daftar fungsi yang ada di dalam source code yang telah dilakukan pengujian. Adapun informasi yang ada di halaman ini adalah sebagai berikut:

1. Function Name, adalah nama fungsi di dalam suatu file PHP di dalam source code tersebut. Nama fungsi ini diambil secara otomatis oleh tools deteksi SQL Injection yang digunakan oleh penulis.
2. Path, adalah lokasi dimana function tersebut berada.
3. Line, adalah posisi nomor baris dimana function tersebut berada.
4. Status, terdiri dari dua nilai, yaitu PASSED, dan WARNING. PASSED diberikan ketika fungsi tersebut tidak ditemukan kerentanan SQL Injection, sebaliknya WARNING diberikan apabila fungsi tersebut terindikasi terdapat kerentanan SQL Injection.
5. Aksi, terdapat sebuah tombol View Source, berfungsi untuk melihat potongan source code yaitu fungsi yang ada di dalam baris data tersebut.

Gambar 8 menunjukkan halaman Detail Scan History.



Gambar 8 Detail Scan History



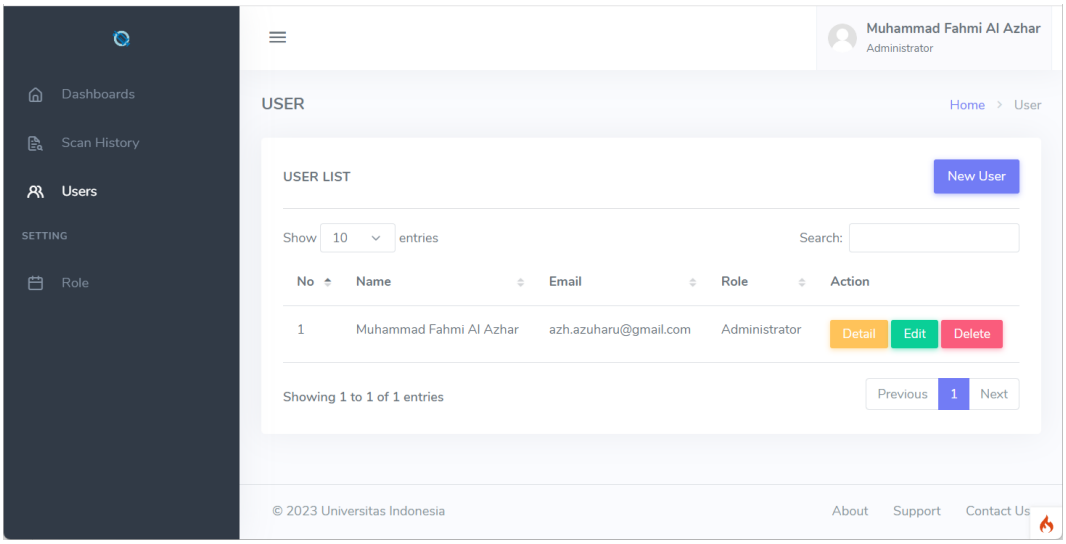
Gambar 9 View Source dari Function Terkait

3.3.5. Halaman Users

Halaman user digunakan untuk mengelola pengguna dari aplikasi dashboard monitoring. Pada halaman ini ditampilkan daftar pengguna yang dapat mengakses ke dalam dashboard. Fitur di dalam halaman user ini adalah:

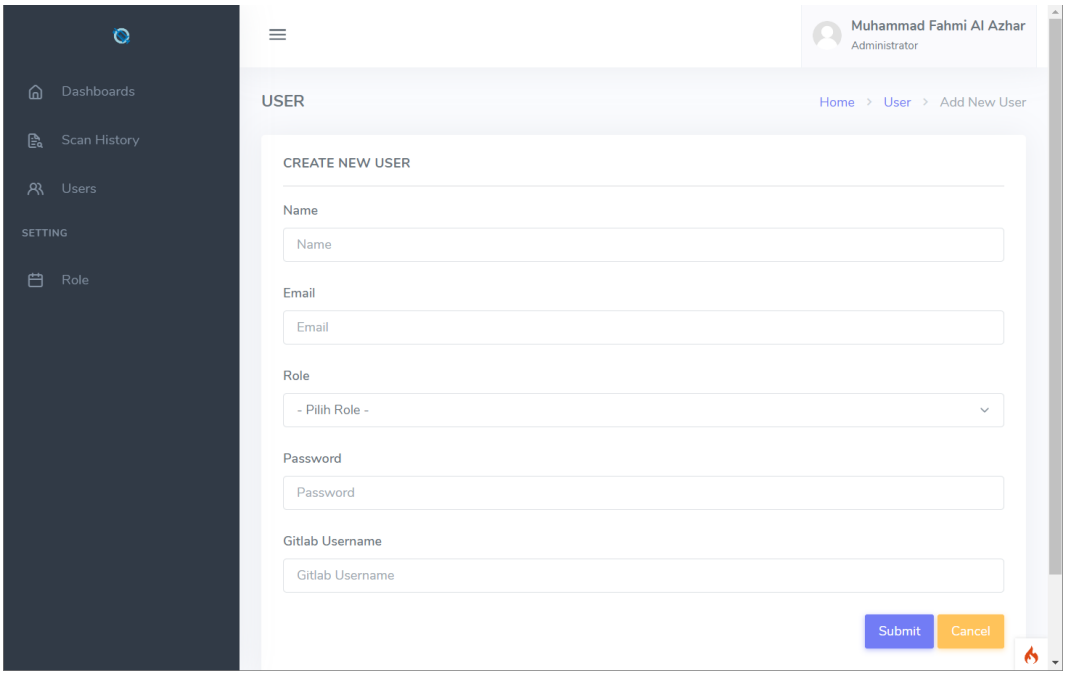
1. Menambah, edit, dan menghapus pengguna.
2. Melakukan filter data pengguna

P-ISSN: 2089-676X
E-ISSN: 2549-0796



Gambar 10 Halaman Users

Gambar 11 menunjukkan tampilan halaman untuk membuat user baru. Terdapat beberapa field yang harus diisi oleh admin ketika menambahkan pengguna baru, yaitu: Nama, Email, Role, terdiri dari Admin dan Member, Password, dan GitLab Username. Username GitLab dibutuhkan sebagai mapping untuk dapat menamplkan riwayat sesuai dengan pengguna di GitLab.



Gambar 11 Form Input User

4. KESIMPULAN

Penelitian yang dilakukan oleh penulis menghasilkan sebuah aplikasi dashboard monitoring sebagai tempat untuk memantau hasil pengecekan yang dilakukan secara otomatis pada platform GitLab. Dengan menggunakan dashboard monitoring ini programmer dapat melihat secara detail hasil pengujian pada source code yang di *push* pada repository git di platform GitLab. Selain itu dashboard ini dapat dimanfaatkan oleh tim operasional, dan tim keamanan IT untuk memantau kerentanan yang ada pada project yang sedang berjalan sejak awal masa pengembangan aplikasi.

UCAPAN TERIMA KASIH

Penelitian ini disponsori oleh Kementerian Komunikasi dan Informatika Republik Indonesia dalam program Beasiswa S2 Dalam dan Luar Negeri Tahun 2021. Peneliti mengucapkan terima kasih kepada Kementerian Komunikasi dan Informasi yang telah memberikan dukungan untuk penelitian ini.

DAFTAR PUSTAKA

[1] OWASP, "https://owasp.org/Top10/," Jun. 2023.
[2] https://about.gitlab.com/, "The DevSecOps Platform | Gitlab," Jun. 2023.
[3] P. Choudhury, K. Crowston, L. Dahlander, M. S. Minervini, and S. Raghuram, "GitLab: work where you want, when you want," *Journal of Organization Design*, vol. 9, no. 1, Dec. 2020, doi: 10.1186/s41469-020-00087-8.
[4] M. Fowler, *Fowler, M. (2004), UML Distilled a Brief Guide to the Standard Object Modelling Language*. Boston: Pearson Education, Inc, 2004.
[5] R. S. Pressman, *Software Engineering: A Practitioners Approach, 6th Edition.* Singapore: McGraw-Hill, 2005.
[6] S. L. Pfleeeger and J. M. Atlee, *Software Engineering: Theory and Practice, 3rd Edition*. US: Prentice Hall, 2006.
[7] A. Vazquez-Ingelmo, F. J. Garcia-Penalvo, and R. Theron, "Information Dashboards and Tailoring Capabilities-A Systematic Literature Review," *IEEE Access*, vol. 7. Institute of Electrical and Electronics Engineers Inc., pp. 109673–109688, 2019. doi: 10.1109/ACCESS.2019.2933472.
[8] B. Loonam and E. Relationship, *Pengertian Entity Relationship Diagram (ERD) Simbolsimbol untuk membuat diagram ERD*. 2010.
[9] https://codeigniter.com/, "Welcome to CodeIgniter," Jun. 2023.
[10] M. Laaziri, K. Benmoussa, S. Khoulji, and M. L. Kerkeb, "A Comparative study of PHP frameworks performance," in *Procedia Manufacturing*, Elsevier B.V., 2019, pp. 864–871. doi: 10.1016/j.promfg.2019.02.295.