ROUTERS: Jurnal Sistem dan Teknologi Informasi
Vol. 3, No. 1, Februari 2025, pp. 20-31
ISSN: 2962-1224, DOI: 10.25181/rt.v3i1.3898

Pembuatan Sistem Deteksi Hardcode Kredensial Pada Repository

Bill Jeferson Nababan[1], Antoni Haikal[2] , Sity Rahmy Maulidya[3]
[1] Program Studi Teknik Informatika, Politeknik Negeri Batam
[2] Program Studi Rekayasa Keamanan Siber, Politeknik Negeri Batam
[3] Pendidikan Matematika, Universitas Pahlawan Tuanku Tambusai

INFORMASI ARTIKEL

Diterima 27 Desember 2024
Direvisi 10 Februari 2025
Diterbitkan 14 Februari 2025

Kata kunci:

Hardcoded Credential;
Trufflehog;
JavaScript; Prototyping;

ABSTRAK

Kredensial yang di-hardcode adalah praktik menyematkan informasi autentikasi, seperti nama pengguna dan kata sandi, secara langsung ke dalam kode sumber perangkat lunak atau aplikasi. Hal ini berarti informasi kredensial tidak disimpan secara terpisah atau dikelola dengan aman, melainkan terintegrasi ke dalam kode program. Praktik ini menimbulkan risiko keamanan yang signifikan, salah satunya adalah kesulitan dalam mengubah kredensial, yang membuat modifikasi kode sumber menjadi tidak praktis dan meningkatkan risiko keamanan. Penelitian ini mengusulkan Sistem Deteksi Kredensial yang di-hardcode berbasis web yang dapat mendeteksi kredensial yang di-hardcode di Repository pada Github. Dengan menerapkan alat keamanan berupa *Trufflehog* ke dalam situs web, sistem ini dapat menampilkan hasil deteksi kredensial yang di-hardcode setelah proses deteksi selesai.Dengan menggunakan metode *Prototyping*, yaitu salah satu pendekatan dalam pengembangan perangkat lunak yang mengikuti serangkaian tahapan yang dilakukan secara berurutan dan diselesaikan satu per satu sebelum masuk ke tahap berikutnya. Teknologi yang digunakan meliputi ReactJs sebagai pustaka untuk membuat *Front-end*, ExpressJs sebagai kerangka kerja untuk membuat Back-end dengan Javascript sebagai bahasa pemrograman, dan MYSQL sebagai basis data. Hasil dari sistem ini dapat membantu menjaga keamanan repository Github dengan menyediakan alat yang dapat mengidentifikasi potensi kebocoran kredensial sensitif. Dengan demikian, pengembang dan tim keamanan dapat mengambil tindakan untuk menghapus atau mengamankan kredensial tersebut secara tepat.

Development Of A System For Detecting Hardcoded Credentials In Repositories

ARTICLE INFO

Received December 27, 2024
Revised February 10, 2025
Published February 14, 2025

Keyword:

Hardcoded Credential;
Trufflehog;
JavaScript; Prototyping;

ABSTRACT

Hardcoded Credential is the practice of embedding authentication information, such as usernames and passwords, directly into the source code of software or applications. This means that the credential information is not stored separately or managed securely, but rather integrated into the program code. This practice poses significant security risks, one of which is the difficulty of changing credentials, making modifications to the source code impractical and increasing security risks. This research proposes a web-based Hardcoded Credential Detection System that can detect Hardcoded Credentials in the Repository on

Github, by implementing security tools in the form of Trufflehog to the website, the system can see the results of Hardcoded Credential detection after the detection process is complete. By using the Prototyping method which is one approach in software development by following a series of stages that are carried out sequentially and completed one by one before entering the next stage. The technologies used include ReactJs as a library for making Front-end, ExpressJs as a Framework for making Back-end with Javascript as a Programming Language, and MYSQL as a database. The results of this system can help in maintaining the security of Github repositories by providing the use of tools that can identify potential leaks of sensitive credentials. Thus, developers and security teams can take action to remove or secure those accidental credentials.

Corresponding Author:

Antoni Haikal, Rekayasa Keamanan Siber, Politeknik Negeri Batam
Email: antoni@polibatam.ac.id

1. PENDAHULUAN

Keamanan perangkat lunak telah menjadi fokus terpenting dalam dunia teknologi informasi. Dengan meningkatnya kompleksitas dan ukuran perangkat lunak, risiko dan kemungkinan kebocoran kredensial semakin menjadi ancaman yang serius. Salah satu bentuk risiko ini adalah penggunaan *hardcoded credentials*, di mana informasi seperti *login*, *username*, *password*, *aws-key*, API, atau kredensial lainnya secara langsung ada di dalam kode sumber. [1]

Kredensial yang tertanam disebut *hardcoded credentials*, yang merupakan teks biasa yang bersifat rahasia tertanam didalam kode sumber. Kredensial mengacu pada praktik menyematkan kredensial teks biasa(tidak terenkripsi), kunci SSH, nama pengguna, *E-mail*, kata sandi, API, *key*, *aws-key* dan lainnya. Dalam beberapa kasus, pelaku ancaman dapat memasukkan kredensial yang dikodekan untuk membuat *backdoor* yang memungkinkan pelaku mengakses perangkat, aplikasi, atau sistem terus-menerus [2]. Pentingnya deteksi dini *hardcoded credentials* sangat krusial, terutama dalam konteks pengembangan perangkat lunak kolaboratif menggunakan sistem kontrol versi seperti Git. Repositori perangkat lunak khususnya github yang mengandung *hardcoded credentials* dapat menjadi sumber potensi kerentanan keamanan yang serius, menyediakan akses langsung ke sumber daya yang dapat dimanfaatkan oleh pihak yang tidak sah [3]. Namun, mengidentifikasi *hardcoded credentials* secara manual dalam repositori yang besar atau kompleks dapat menjadi tugas yang sangat sulit dan rentan terhadap kesalahan manusia. *Hardcoded credentials* dapat dijadikan solusi sementara, walaupun terkadang pengembang tidak menyadari bahwa git sebenarnya melacak secret dalam kredensial [4]. Oleh karena itu, perlu adanya sistem otomatis untuk mendeteksi *hardcoded credentials* dalam repositori pada github. Sistem ini tidak hanya akan membantu pengembang dalam menerapkan praktik keamanan terbaik tetapi juga dapat mengurangi potensi risiko keamanan yang dapat timbul dari kelalaian dan penyebab lainnya.

Mengangkat masalah kebocoran data dari *Cyber Blitz* Direktorat Keamanan Siber, terdapat kebocoran data berupa secret kredensial di *source code* pada Neopets sebesar 69 juta data pribadi, dari 69 juta tersebut didalam *source code* terdapat data pribadi berupa nama pengguna, alamat *E-mail*, jenis kelamin, kata sandi, dan hal pribadi lainnya. Hal serupa juga terjadi pada aplikasi Kesehatan Mental Fellyou yang mengalami kebocoran data 76.000 email pengguna [5].

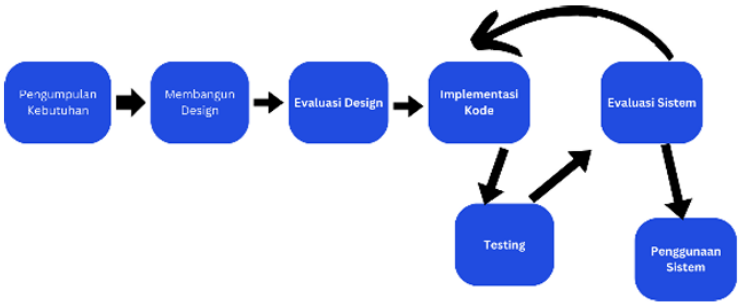
Maka dari uraian masalah diatas, Pembuatan sistem deteksi kredensial yang tertanam pada repository akan menjadi solusi positif dan bisa direalisasikan untuk menghadapi tantangan keamanan ini. Sistem ini diharapkan dapat mengintegrasikan teknik-teknik deteksi kredensial pada repository github dengan *trufflehog* dan analisis sintaksis, untuk secara efisien dan akurat

mengidentifikasi hardcoded kredensial. *TruffleHog* nantinya akan digunakan untuk mendeteksi API, nama pengguna, kata sandi, secret yang dikommit kedalam git [6]. Setelah proses deteksi kredensial yang tertanam selesai, akan dilakukan validasi untuk mengetahui valid atau tidak valid nya kredensial dari hasil validasi sebelumnya, yang dimana jika valid *user* dapat melakukan *export* hasil validasi ke format Excel.

Dengan implementasi sistem ini, pengembang akan mendapatkan keuntungan dari lingkungan pengembangan yang lebih aman, sementara organisasi akan dapat menjaga reputasi dan kepercayaan pengguna mereka. Oleh karena itu, tugas akhir ini bertujuan untuk mengembangkan sistem deteksi hardcoded kredensial dengan bantuan *tools Trufflehog* yang mudah di integrasikan, dan efisien, yang dapat diterapkan dalam berbagai konteks pengembangan perangkat lunak. Implementasi sistem ini akan memberikan kontribusi signifikan terhadap praktik keamanan perangkat lunak dan membantu melindungi informasi kredensial seperti kata sandi, nama pengguna dari risiko kebocoran dan penyalahgunaan [7].

2. METODE

Penelitian ini dibuat dengan tujuan untuk mendeteksi kebocoran hardcoded kredensial pada repository di github. Oleh karena itu beberapa tahapan yang digunakan untuk mengembangkan sistem deteksi ini digambarkan pada Gambar 1 di bawah ini:



Gambar 1. Ilustrasi *Protyping* Pada Sistem

Metode Prototyping ini bertujuan mengumpulkan informasi dari user atau client sehingga client dapat berinteraksi dan memakai situs web dengan model *prototype* yang akan dibangun secara keseluruhan [8]. Terlihat pada Gambar 1 bahwa terdapat 7 tahapan *Prototyping* pada sistem, berikut penjelasannya:

1. Tahapan Pengumpulan Kebutuhan, teknik pengumpulan data yang akan digunakan untuk kebutuhan sistem adalah mengidentifikasikan semua kebutuhan dan garis besar sistem yang akan dibuat. Salah satunya adalah pengumpulan repositori pada github, yang nantinya penulis akan mengumpulkan banyak link repository dari github yang akan dideteksi kredensialnya melalui trufflehog pada website.
2. Membangun Desain, membangun Desain yang bertujuan memberikan gambaran lengkap tentang bagian apa saja yang harus dikerjakan dan bagaimana tampilan dari sebuah sistem, sehingga membantu peneliti dalam mendefinisikan arsitektur sistem yang akan dibuat secara keseluruhan. Tahapan Desain meliputi desain UI (*User Interface* menggunakan aplikasi Figma, dan perancangan database menggunakan aplikasi Mysql Workbench.
3. Evaluasi Desain, Evaluasi desain dilakukan oleh client apakah desian atau prototyping yang sudah dibuat sesuai dengan keinginan klien.
4. Implementasikan Kode, rancangan desain yang sudah dirancang tadi akan diimplementasikan kedalam sebuah Bahasa pemrograman. Bahasa pemrograman yang akan penulis gunakan nantinya adalah untuk bagian Front-End nanti menggunakan Bahasa pemrograman javascript

dan menggunakan Framework Vite dan untuk di *Back- End* sendiri akan menggunakan *Framework Express-js,* Node JS dan database berupa Mysql.

5. Testing, setelah pengembang selesai membangun situs web sampai ke tahap dimana *truffleHog* juga sudah bisa digunakan di dalam situs web tersebut, akan dilakukan pengujian pada sistem bersama dengan klien untuk mencari kesalahan atau kekurangan yang ada dalam situs web yang akan diperbaiki untuk hasil yang lebih baik.
6. Evaluasi Sistem, di tahapan ini akan membutuhkan bantuan dari client untuk membantu dalam mengevaluasi sistem. Apakah sistem atau website ini sudah sesuai kebutuhan client dan dapat digunakan dengan baik. Dan jika tidak terpenuhi akan dilakukan perbaikan oleh pengembang.
7. Penggunaan Sistem, Sistem yang sudah di uji dan disetujui oleh klien sudah bisa digunakan.

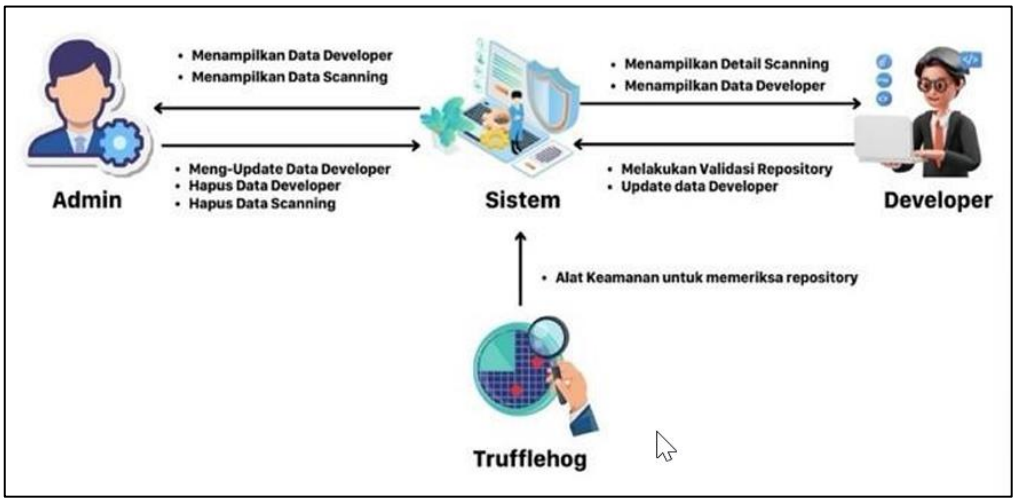
3. HASIL DAN PEMBAHASAN

3.1. Tahapan *Requirement*

Sistem Deteksi kredensial yang tertanam pada repository ini dibuat dengan menggunakan arsitektur web yang dapat diakses melalui web penjelajah. Analisis dilakukan berdasarkan dokumentasi yang ada serta pemahaman mendalam terhadap kebutuhan sistem berupa:

- Developer dapat melakukan login ke dalam situs web.
- Developer melakukan pemeriksaan repository melalui web dengan bantuan *Truffalehog*.
- Hasil scanning akan ditampilkan dan diperiksa secara manual oleh Developer.

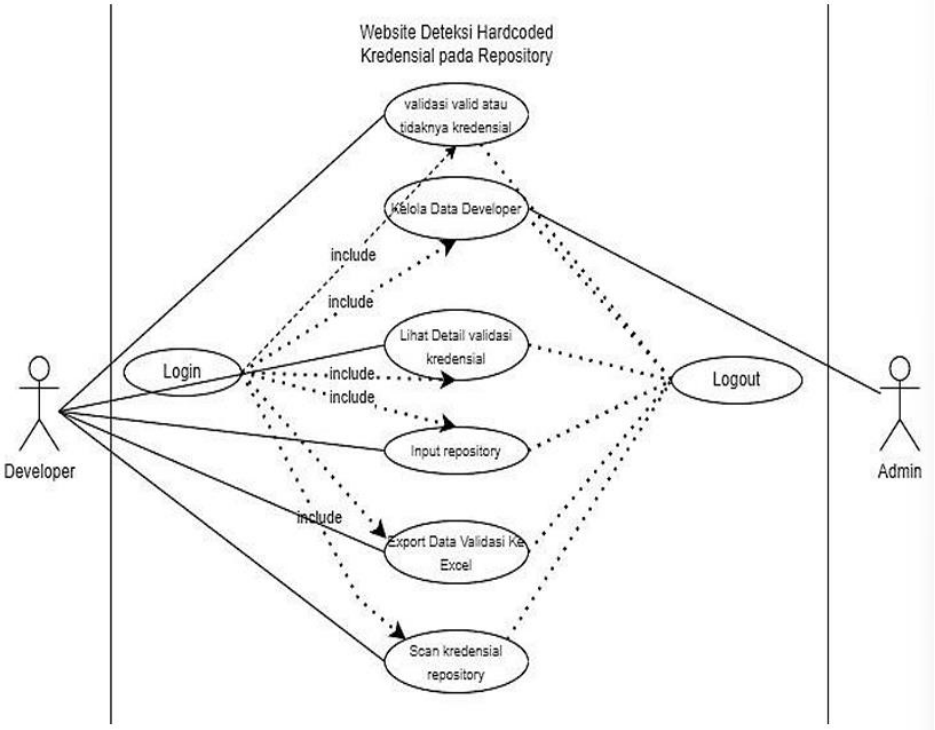
Pada sistem ini memiliki 2 roles yaitu ada Admin dan Developer dan tools yang kita sebut sebagai *trufflehog*. Untuk roles admin nantinya akan digunakan oleh tim keamanan perusahaan, dan developer akan digunakan oleh pengguna *platform* github secara umum. Untuk lebih jelasnya perhatikan Gambar Umum Sistem yang terlampir pada Gambar 2 berikut:



Gambar 2. Gambaran Umum Sistem

1. Usecase Diagram

Diagram usecase menunjukkan interaksi antara dua aktor utama, yaitu Admin dan Developer dengan sistem deteksi kredensial hardcoded pada repository. Melalui Gambar 3 berikut terlihat lebih detail mengenai alur dalam usecase diagram.



Gambar 3. Gambar Diagram *Use Case*

Terlihat pada Gambar 3, Diagram use case di atas menunjukkan interaksi antara dua aktor utama, yaitu Admin dan Developer, dengan sistem deteksi kredensial hardcoded pada repository. Berikut adalah penjelasan mengenai alur dalam diagram tersebut:

a) Admin dan *Developer* harus melakukan login terlebih dahulu untuk mengakses fitur-fitur yang tersedia dalam sistem. Setelah *login, Developer* dapat melakukan *scanning* terhadap repository yang ada di GitHub menggunakan alat yang disebut *Trufflehog*. Trufflehog adalah alat keamanan yang digunakan untuk memeriksa repository dan mencari kredensial seperti *email, password, username*, dan informasi rahasia lainnya yang mungkin terdapat dalam kode sumber.

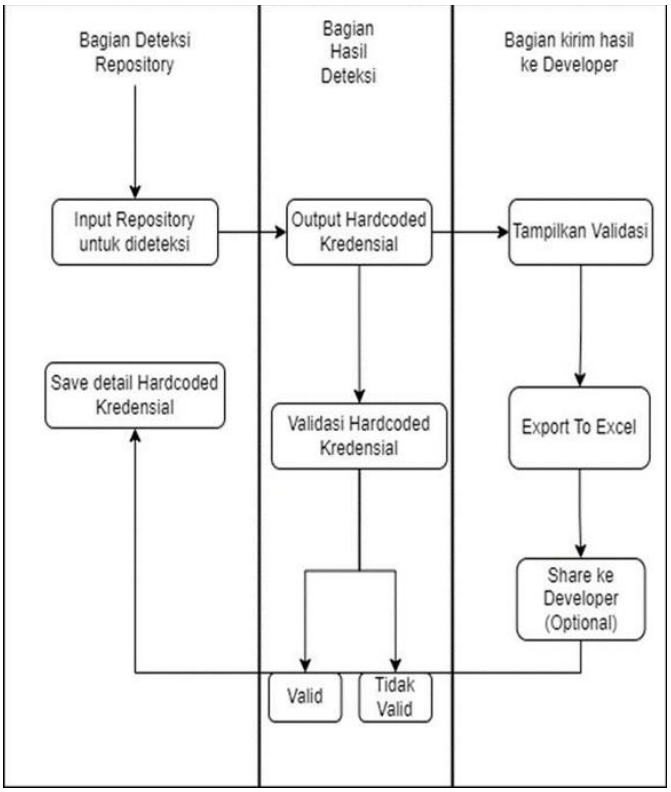
b) Setelah proses scanning dilakukan oleh *Trufflehog*, Developer memeriksa hasil *scanning* tersebut secara manual untuk menentukan apakah kredensial yang ditemukan valid atau tidak valid. Jika valid, Jika valid, maka kredensial tersebut masih aktif dan perlu ditangani dengan hati-hati. Jika tidak valid, berarti kredensial tersebut sudah tidak digunakan atau tidak aktif lagi.

c) Admin memiliki tugas utama untuk mengelola data pengguna dan data repository. Tugas ini mencakup melihat detail validasi kredensial, meng-update, dan menghapus data yang diperlukan. Admin tidak terlibat langsung dalam proses scanning, tetapi memastikan bahwa data yang dikelola tetap akurat dan up-to-date.

d) Setelah semua proses selesai, baik Admin maupun Developer dapat keluar dari sistem

2. Activity Diagram

Diagram aktivitas merupakan rancangan aliran aktivitas yang digunakan pada sebuah sistem yang dijalankan. Terlihat pada Gambar 4 berikut bahwa di dalam alur aktivitas terdapat komponen dengan bentuk tertentu yang dihubungkan melalui tanda panah. Kemudian panah itu mengarah ke urutan aktivitas yang akan dilakukan dari awal sampai akhir. Berikut gambaran activity diagram dan penjelasannya.



Gambar 4. Gambar Diagram Alur Aktivitas

Activity diagram di bawah ini menggambarkan proses deteksi dan validasi kredensial hardcoded dalam sebuah repository, serta penyampaian hasil deteksi tersebut ke developer. Proses ini terdiri dari tiga bagian utama.

A. Deteksi Repository
   a. Input Repository untuk Dideteksi: Langkah pertama adalah memasukkan repository yang ingin diperiksa untuk kredensial yang tertanam.
   b. Save Detail Hardcoded Kredensial: Setelah deteksi dilakukan, detail dari kredensial yang ditemukan disimpan untuk langkah-langkah selanjutnya.

B. Hasil Deteksi
   a. Output Hardcoded Kredensial: Kredensial yang ditemukan kemudian dikeluarkan sebagai *output*.
   b. Validasi Hardcoded Kredensial: Kredensial tersebut kemudian divalidasi untuk memastikan apakah masih berlaku atau sudah tidak diragukan lagi.
   c. Valid: Jika kredensial valid, berarti kredensial tersebut masih aktif dan perlu ditangani dengan hati-hati.

d. Tidak Valid: Jika tidak valid, berarti kredensial tersebut sudah tidak aktif atau tidak lagi digunakan.

C. Kirim Hasil ke Developer

a. Tampilkan Validasi: Hasil dari proses validasi ditampilkan agar bisa dilihat.
b. Export to Excel: Hasil ini kemudian diekspor ke dalam file Excel untuk memudahkan analisis dan dokumentasi.
c. Share ke Developer (Optional): Akhirnya, hasil ini bisa dibagikan kepada developer yang bertanggung jawab untuk memperbaiki atau menghapus kredensial hardcoded dari kode.

3. Kebutuhan Fungsional dan Non Fungsional

a. Functional Requirement (FR)

Functional Requirement adalah kebutuhan yang berisi proses atau layanan yang nantinya akan disediakan oleh sistem. Tabel 1 menunjukkan hasil analisis mengenai Functional Requirement pada Sistem Informasi Dana Amal Polibatam.

Tabel 1. Functional Requirement pada Sistem Informasi Dana Amal Polibatam

NO	Kebutuhan Fungsional
F001	Pengguna (Developer dan Admin) dapat melakukan login
F002	Developer dapat melakukan Register jika belum terdaftar
F003	Admin dapat melakukan delete/update pada data akun Developer
F004	Developer dapat melakukan Update User
F005	Developer dapat melakukan input link repository github ke dalam form trufflehog
F006	Developer dapat melihat detail dari validasi yang berupa valid atau tidak saat di deteksi
F007	Developer hanya dapat melihat hasil validasi dari link yang Developer input, developer lain tidak dapat melihat hasil validasi developer yang lainnya
F008	Developer dapat meng-Export hasil validasi dalam format Excel
F009	Admin dapat menghapus data hasil validasi Repository
F010	Admin dapat menyimpan data akhir dari validasi kredensial kedalam database.

b. Non Functional Requirement (NFR)

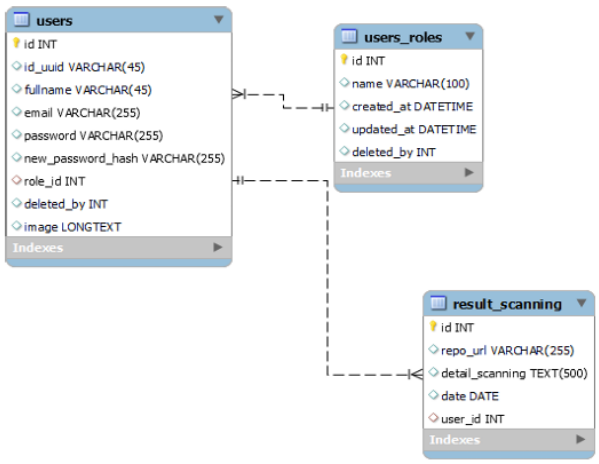
Non Functional Requirement adalah kebutuhan yang menitik beratkan pada property pendukung yang dimiliki oleh sistem. Berikut disajikan Tabel 2 untuk detail penjelasan kriteria dan parameter pendukunh yang dimiliki sistem.

Tabel 2. Tabel Non Functional Requirement

Kriteria	Parameter
Availability	Sistem dapat di akses apabila terhubung ke jaringan internet Sistem mampu berjalan selama 24 jam non-stop, kecuali apabila ada perawatan sistem atau pembaharuan sistem Sistem dapat dijalankan dimana saja dan kapan saja
Ergonomy	Sistem harus dapat digunakan dengan mudah atau user friendly
Bahasa	Menggunakan Bahasa Inggris
Safety	Sistem dapat memastikan bahwa data yang digunakan harus terlindungi dari akses yang tidak berwenang

3.2 Membangun Desain

Desain User Interface (UI) adalah proses merancang elemen-elemen visual dan interaktif dari suatu sistem atau produk, dengan tujuan utama untuk meningkatkan pengalaman pengguna[9]. Entity Relationship Diagram (ERD) adalah alat visual yang digunakan dalam rekayasa perangkat lunak dan desain sistem database. ERD digunakan untuk menggambarkan dan memodelkan hubungan antara table pada database. Gambar 5 menunjukkan struktur basis data untuk sistem deteksi kredensial yang tertanam. Pada gamabr tersebut terlihat bahwa terdapat tiga tabel utama, yaitu *users, users_roles*, dan *result_scanning*.



Gambar 5. Gambar Diagram *Entity Relationship*

3.3 Implementasi Kode

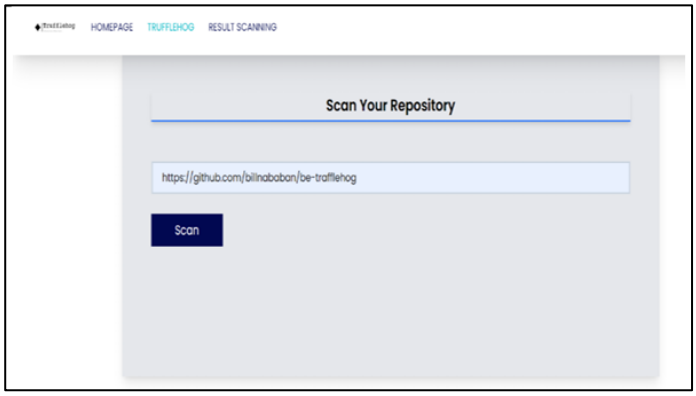
1. Implementasi halaman landing page

Halaman landing page ini yang dapat diakses secara bebas oleh semua orang tanpa harus *login*. Pada halaman ini menampilkan seputar informasi umum tentang kredensial yang tertanam, Pengenalan *Truflehog* dan Keamanan lainnya. Implementasi halaman landing page ini dapat dilihat pada Gambar 6.

Gambar 6. Landing Page

2. Implemtansi Halaman Validasi Repositori

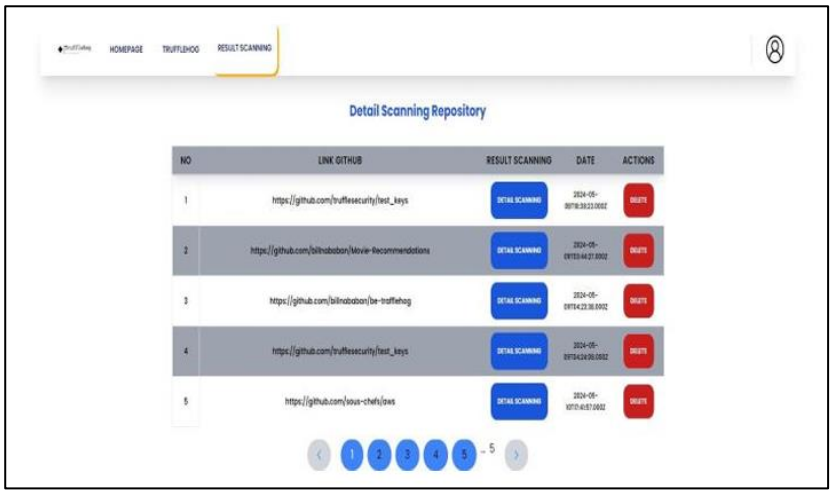
Halaman ini hanya dapat diakses jika developer sudah melakukan login terlebih dahulu. Pada halaman ini, Developer akan menginput link repository github mereka untuk divalidasi oleh *Trufflehog* untuk validasi keamanan repositori. Berikut diberikan Gambar 7 yang menunjukkan input link repository.



Gambar 7. Form Input Link repository

3. Implementasi Halaman History Hasil Validasi

Di halaman History ini, user dapat melihat hasil dari validasi repository yang sudah selesai di deteksi oleh *Trufflehog*, dan untuk history ini juga sudah dipisah untuk data tiap developer, para developer hanya bisa melihat data history dari Link yang mereka input dan Developer tidak bisa melihat history Developer yang lainnya. Berikut disajikan Gambar 8 yang menunjukkan halaman rincian scanning yang telah dilakukan.



Gambar 8. Halaman Detail Scanning

3.4 Testing dan Evaluasi

Testing pada sistem ini bertujuan untuk memverifikasi bahwa sistem yang telah dikembangkan dapat berfungsi secara optimal sesuai dengan kebutuhan fungsionalnya. Proses pengujian sistem ini dilakukan menggunakan pendekatan black box, yang menitik beratkan pada penilaian kualitas perangkat lunak dengan cara mengevaluasi fungsionalitasnya untuk mengidentifikasi ketidakcocokan fungsi, kesalahan antarmuka, dan potensi kesalahan lainnya[10]. Berikut disajikan Tabel 3 yang berisikan hasil dari testing pada sistem ini.

Tabel 3. Hasil testing sistem

No	Fungsional	Skenario Pengujian	Role	Indikator Keberhasilan	Hasil Pengujian
1	Login	Pengguna memasukan username benar dan password benar	Developer dan Admin	Sistem memberikan pesan Selamat dating dan Pengguna berhasil login	Sesuai
		Pengguna memasukan email benar dan password salah		Sistem memberikan pesan Password salah	Sesuai
		Pengguna memasukan email salah dan password benar		Sistem memberikan pesan Email yang anda masukkan salah	Sesuai
		Pengguna memasukan email salah dan password salah		Sistem memberikan pesan Akun anda tidak terdaftar silahkan registrasi	Sesuai
2	Register	Pengguna memasukkan username, email, password, dan konfirmasi	Developer	Sistem akan memberikan pesan register berhasil dan langsung	Sesuai

No	Fungsional	Skenario Pengujian	Role	Indikator Keberhasilan	Hasil Pengujian
		password		diarahkan ke halaman login	
		Pengguna memasukkan password tidak berisi simbol, angka dan huruf kapital		Sistem akan memberikan pesan, silahkan tambahkan simbol, angka dan huruf kapital untuk keamanan data anda	Sesuai
		Pengguna memasukkan password yang berisi simbol, angka dan huruf kapital		Sistem akan memberikan pesan, Password benar silahkan lakukan register	Sesuai
3	Mengubah data user	Admin Memasukkan Email dan Full Name data user yang ingin dirubah	Admin	Sistem akan menampilkan pesan Ubah data user berhasil	Sesuai
		Admin memasukan data user dengan Username yang sudah digunakan		Sistem akan menampilkan pesan Username sudah ada sebelumnya	Sesuai
		Admin memasukan data user dengan Email yang digunakan		Sistem akan menampilkan pesan Email sudah digunakan	Sesuai
		Admin memasukan data user dengan Username dan Email yang sudah digunakan		Sistem akan menampilkan pesan Username dan email sudah digunakan	Sesuai
4	Menghapus user	Menekan tombol hapus pada user yang ingin dihapus	Admin	Sistem akan menampilkan pesan Akun berhasil dihapus	Sesuai

Dari testing dan evaluasi diatas didapat bahwasanya sistem sudah sesuai dengan kebutuhan yang diinginkan.

4. Kesimpulan

Berdasarkan hasil dari perancangan dan implementasi telah dibangun Sistem deteksi *Hardcoded Credential* pada repository *Public* di Github Berbasis Web dengan metode *Prototyping*, dapat disimpulkan bahwa sistem ini memberikan kemudahan bagi developer dalam memeriksa keamanan repositori Github, melihat validasi dari *trufflehog*. Berdasarkan hasil testing dengan metode *Black Box* sistem ini sudah sesuai dan dapat digunakan dengan baik. Namun sistem ini masih belum sepenuhnya sempurna, dikarenakan proses validasi dari *Trufflehog* sedikit lama dikarenakan trufflehog memeriksa lewat commit secara keseluruhan. Oleh karena itu diperlukan pengembangan selanjutnya untuk menambahkan fitur-fitur lain yang masih belum tersedia

DAFTAR PUSTAKA

[1] Ashari, M. (2022). Belajar Dari Kebocoran Data Kredensial: Data Yang Paling Berharga adalah Data Pribadi. Kementerian Keuangan Republik Indonesia, 22 Maret 2022. Available at: https://www.djkn.kemenkeu.go.id/kpknl-kisaran/baca-artikel/14838/Belajar-Dari-Kebocoran-Data-Kredensial-Data-Yang-Paling-Berharga-adalah-Data-Pribadi.html.

[2] Miller, M. (2019). Hardcoded and embedded credentials are an IT security Hazard–Here's what you need to know. *Hardcoded and embedded credentials are an IT security hazard Here's what you need to know*.

[3] Majdinasab, V., Bishop, M. J., Rasheed, S., Moradidakhel, A., Tahir, A., & Khomh, F. (2024, March). Assessing the Security of GitHub Copilot's Generated Code-A Targeted Replication Study. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)* (pp. 435-444). IEEE.

[4] Segura, T. (2023). The nightmare of hard-coded credentials. *Network Security*, *2023*(2).

[5] C. Blitz, "Data Kebocoran-kebocoran data kredensial," pp. 1-4, 2022.

[6] Nehmer, F. (2022). *Entwicklung eines Systems zur automatisierten statischen Sicherheitsüberprüfung von Software Repositories zur Integration in agile Softwareentwickungsprozesse* (Doctoral dissertation, Hochschule für Angewandte Wissenschaften Hamburg).

[7] A. Developers, "Secret Kriptografis yang Di-hardcode," 07 11 2023.

[8] R. A. W. N. Petrus Yoko, "Penerapan Metode Prototypedalam Perancangan AplikasiSIPINJAMBerbasis Websitepada Credit UnionCanaga Antutn," *Jurnal Ilmiah Merpati* , vol. Vol. 7, pp. 1-12, 2019.

[9] D. Ridzky, "User Interface Modelling for SIBI (Sistem Isyarat Bahasa Indonesia/Indonesian Sign Language System) learning applications using the User-Centered Design Method," *Journal of Physics: Conference Series*, 2018.

[10] I. Permatasari, "Pengujian Black Box Menggunakan Metode Analisis Nilai Batas pada Aplikasi DANA," vol. Vol. 3 No. 2, p. 15, Desember 2023.