JOIN (Jurnal Online Informatika)
p-ISSN:2528-1682, e-ISSN:2527-9165
Jilid 8 Nomor 1 | Juni 2023: 1-3
DOI: 10.15575/join.xxxx.xx

Program Java untuk Membuat dan Mengukur Kekuatan Kata Sandi dengan Algoritma

Ramadan Nurwahid
Jurusan Teknik Informatika, UIN Sunan Gunung Djati Bandung, Indonesia

Info Artikel

Kata kunci:

Kata sandi,
Kekuatan,
Algoritma,
Java,
Ekspresi reguler.

Penulis yang sesuai:

Ramadan Nurwahid,
Jurusan Teknik Informatika Fakultas Sains & Teknologi UIN Sunan Gunung Djati Bandung
Jl. AH Nasution No.105, Cibiru, Bandung, Indonesia. 40614
Email:
rnurwahid.ac@gmail.com

ABSTRAK

Kata sandi adalah salah satu cara untuk melindungi informasi pribadi dan rahasia dari akses yang tidak sah. Namun, tidak semua kata sandi memiliki tingkat keamanan yang sama. Kata sandi yang lemah mudah ditebak atau diretas oleh pihak yang tidak berwenang. Oleh karena itu, penting untuk membuat dan mengukur kekuatan kata sandi dengan algoritma yang tepat. Tujuan dari penelitian ini adalah untuk mengembangkan program Java yang dapat membuat dan mengukur kekuatan kata sandi dengan algoritma yang berdasarkan pada kriteria panjang, huruf besar, huruf kecil, angka, dan karakter khusus. Program ini menggunakan kelas Scanner untuk membaca masukan dari pengguna, metode matches untuk memeriksa kesesuaian dengan ekspresi reguler, dan metode bantuan untuk menghitung skor dan menentukan kekuatan kata sandi. Hasil dari penelitian ini menunjukkan bahwa program Java yang dikembangkan dapat membuat dan mengukur kekuatan kata sandi dengan algoritma yang efektif dan akurat. Program ini dapat memberikan umpan balik kepada pengguna tentang kekuatan kata sandi yang dimasukkan, apakah kuat atau lemah, berdasarkan skor yang diperoleh. Program ini juga dapat digunakan sebagai alat pembelajaran bagi pemrogram Java pemula yang ingin mempelajari konsep dasar seperti kelas, metode, percabangan, dan ekspresi reguler.

1. PENDAHULUAN

Perkembangan teknologi informasi saat ini telah mengubah banyak aspek kehidupan manusia. Salah satunya adalah semakin maraknya penyimpanan data dan informasi penting dalam format digital. Data pribadi, data perusahaan, nomor rekening bank, hingga rahasia negara kini banyak yang tersimpan dalam perangkat digital seperti komputer, laptop, smartphone, dan cloud [1]. Di sisi lain, risiko kebocoran dan pencurian data digital ini juga semakin meningkat tajam.

Menurut laporan dari Risk Based Security, jumlah records data yang terkompromi akibat insiden kebocoran data meningkat sebesar 424% di tahun 2019 dibandingkan tahun sebelumnya [2]. Data-data sensitif yang bocor ini berisiko disalahgunakan untuk kejahatan identitas, penipuan bank, hingga mengancam keamanan nasional suatu negara. Oleh karena itu, keamanan data digital saat ini menjadi isu yang sangat krusial.

Salah satu teknik utama dalam keamanan data adalah autentikasi pengguna dan otorisasi akses menggunakan kata sandi [3]. Sayangnya, banyak individu maupun organisasi masih menggunakan kata sandi yang lemah dan mudah ditebak. Survei tahun 2019 menunjukkan bahwa kata sandi seperti

"123456", "password", dan "qwerty" masih mendominasi daftar 200 kata sandi paling umum yang digunakan [4]. Kata sandi lemah semacam ini sangat rentan terhadap serangan brute force attack. Bahkan kata sandi yang dianggap kuat beberapa tahun lalu, seperti kombinasi kata-kata acak, kini bisa dengan mudah dikompromikan menggunakan teknik cracking seperti dictionary attack dan probabilistic context-free grammars [5].

Oleh karena itu, dibutuhkan panduan yang jelas bagi pengguna untuk membuat kata sandi yang benar-benar kuat. Beberapa rekomendasi umum untuk membuat kata sandi yang kuat adalah dengan panjang minimal 8 karakter, mengandung campuran huruf besar, huruf kecil, angka, dan karakter simbol [6]. Akan tetapi, hanya mengandalkan panjang karakter saja tidak cukup. Diperlukan kerumitan kata sandi yang tinggi dengan kombinasi karakter acak agar sulit ditebak.

Untuk mengukur tingkat kerumitan dan kekuatan sebuah kata sandi, diperlukan komputasi algoritma tertentu. Beberapa peneliti telah mengusulkan berbagai algoritma untuk mengestimasi dan memberi skor kekuatan kata sandi, di antaranya Shannon entropy, NIST password scoring system, password guessability, dan nomogram password cracker [7][8][9][10]. Sayangnya implementasi algoritma-algoritma tersebut dalam bahasa pemrograman populer seperti Java masih sangat terbatas.

Padahal, dengan adanya program komputer berbasis Java yang dapat menganalisis dan memberi skor kekuatan kata sandi, hal ini akan sangat bermanfaat bagi pengguna awam dalam menghasilkan kata sandi yang lebih kuat. Selain itu, program komputer juga memungkinkan pengintegrasian algoritma canggih untuk analisis kata sandi ke dalam berbagai aplikasi keamanan data.

Oleh karena itu, penelitian ini bertujuan untuk mengimplementasikan algoritma pengukuran kekuatan kata sandi dalam bahasa pemrograman Java. Program Java yang dibuat akan menerima masukan kata sandi dari pengguna, kemudian secara otomatis menganalisisnya berdasarkan panjang dan kerumitan menggunakan algoritma tertentu. Selanjutnya program akan memberikan skor kekuatan kata sandi berdasarkan hasil analisis algoritma.

Beberapa algoritma yang akan diimplementasikan dan dibandingkan dalam program Java ini di antaranya Shannon entropy, NIST password scoring system, dan Zxcvbn. Shannon entropy adalah algoritma klasik yang mengukur kerumitan kata sandi berdasarkan informasi entropy, semakin acak kombinasi karakternya maka semakin tinggi nilai entropy-nya [7]. NIST password scoring system menilai kata sandi berdasarkan panjang dan kerumitan dengan memeriksa ada tidaknya campuran uppercase, lowercase, angka dan simbol [8]. Sedangkan Zxcvbn merupakan library open source yang baru-baru ini populer untuk mengestimasi kekuatan kata sandi dengan memanfaatkan common password dictionaries dan pattern masking [11].

Dengan mengimplementasikan dan membandingkan beberapa algoritma analisis kata sandi ini, diharapkan program Java yang dihasilkan dapat memberikan skor kekuatan kata sandi yang akurat dan objektif. Pengguna awam maupun administrator sistem keamanan data dapat memanfaatkan program ini untuk mengevaluasi dan meningkatkan kekuatan kata sandi yang digunakan. Lebih lanjut, penelitian ini diharapkan dapat berkontribusi dalam pengembangan ilmu pengetahuan di bidang keamanan informasi khususnya terkait keamanan kata sandi.

2. METODE

Penelitian ini bertujuan untuk membangun sebuah program Java yang dapat menganalisis dan memberikan penilaian terhadap kekuatan sebuah kata sandi yang dimasukkan oleh pengguna. Program ini akan menerapkan sebuah algoritma penilaian kekuatan kata sandi berdasarkan panjang, kerumitan, dan pemenuhan kriteria penggunaan huruf besar, huruf kecil, angka, dan karakter khusus.

Desain penelitian mengacu pada pendekatan prototyping, di mana program dibangun melalui proses iteratif dengan menambahkan fitur secara bertahap dan melakukan pengujian berulang kali hingga

prototipe yang diinginkan tercapai [14]. Pertama, program dasar Java dibuat dengan kemampuan menerima input kata sandi dari pengguna dan menampilkan output penilaian kekuatannya. Kemudian secara bertahap algoritma penilaian disempurnakan dengan menambahkan aturan pengecekan panjang kata sandi, pola karakter, dan perhitungan skor kekuatan. Setiap iterasi diakhiri dengan pengujian program untuk memastikan hasil penilaian sudah sesuai.

Algoritma penilaian kekuatan kata sandi pada program ini terdiri dari 2 fungsi utama, yaitu calculatePasswordStrength( ) dan determineStrength( ) (Pseudocode pada Gambar 1) [17]. Fungsi calculatePasswordStrength( ) menerima input kata sandi dari pengguna, kemudian mengeceknya terhadap pola ekspresi regular yang sudah ditentukan. Jika kata sandi memenuhi semua kriteria (panjang min 8 karakter, huruf besar, huruf kecil, angka, dan karakter khusus), maka akan diberi skor maksimal 8. Jika tidak, skor 0.

Selanjutnya fungsi determineStrength( ) akan menentukan rating kekuatan kata sandi berdasarkan skor dari calculatePasswordStrength( ). Jika skor 8, maka kekuatan 'Kuat'. Jika skor 0, maka 'Lemah' dan akan menampilkan alasan mengapa kata sandi tersebut lemah. Pengecekan alasannya dilakukan dengan memeriksa apakah kata sandi memenuhi panjang minimum, huruf besar, angka, dan karakter khusus [12], [16].

Pengujian program dilakukan dengan memasukkan berbagai variasi kata sandi, mulai dari yang sangat lemah hingga sangat kuat. Kata sandi lemah misalnya 'password123' dan kata sandi kuat misalnya 'jDh92!8sLdkW' [13]. Respon program terhadap masing-masing kata sandi diamati dan dibandingkan dengan ekspektasi kekuatan seharusnya. Jika terdapat inkonsistensi hasil penilaian, maka algoritma dan aturan pemeriksaan diperbaiki [14], [15]. Data uji yang digunakan merujuk pada daftar 200 kata sandi paling umum tahun 2019 [13].

Hasil pengujian menunjukkan bahwa program Java yang dibangun sudah mampu menilai kekuatan kata sandi dengan akurat sesuai dengan aturan panjang, kerumitan, dan kriteria penggunaan pola karakter yang telah ditentukan. Implementasi algoritma penilaian kekuatan kata sandi pada program Java ini diharapkan dapat membantu pengguna menghasilkan kata sandi kuat, serta bermanfaat bagi pengembangan aplikasi keamanan data.

Gambar 1. Pseudocode algoritma penilaian kekuatan kata sandi [17]

```
import java.util.Scanner;

public class PasswordStrengthTester {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Masukkan panjang kata sandi: ");
        int length = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Kata sandi yang dibuat: ");
        String password = scanner.nextLine();

        int score = calculatePasswordStrength(password);
        String strength = determineStrength(score, password, length);

        System.out.println(strength);
    }

    public static int calculatePasswordStrength(String password) {
        int score = 0;
```

```
    if (password.matches("^(?=.*[A-Z])(?=.*[a-z])(?=.*\\d)(?=.*[!@#$%^&*()_+=\\-\\[\\]{};':\"\\\\|,.<>\\/?]).{8,}$")) {
      score = 8;
    } else {
      score = 0;
    }

    return score;
  }

  public static String determineStrength(int score, String password, int length) {
    if (score >= 8) {
      return "Kekuatan kata sandi: Kuat";
    } else {
      StringBuilder reason = new StringBuilder("Kekuatan kata sandi: Lemah\nAlasan: ");

      if (password.length() < length) {
        reason.append("Kata sandi memiliki panjang kurang dari ").append(length).append(" karakter. ");
      }

      if (!password.matches(".*[A-Z].*")) {
        reason.append("Kata sandi tidak memiliki huruf besar. ");
      }

      if (!password.matches(".*[!@#$%^&*()_+=\\-\\[\\]{};':\"\\\\|,.<>\\/?].*")) {
        reason.append("Kata sandi tidak memiliki karakter khusus. ");
      }

      if (!password.matches(".*\\d.*")) {
        reason.append("Kata sandi tidak memiliki angka. ");
      }

      return reason.toString();
    }
  }
}
```

3. HASIL DAN PEMBAHASAN (10 PT)

Penelitian ini telah berhasil membangun program Java yang dapat menganalisis dan memberikan penilaian kekuatan kata sandi yang dimasukkan pengguna. Berikut ini adalah hasil implementasi program beserta pembahasan mengenai proses dan hasil pengujiannya.

3.1. Analisis Algoritma

Algoritma yang dikembangkan berfokus pada pengukuran kekuatan sebuah kata sandi. Algoritma ini memanfaatkan ekspresi reguler untuk memeriksa berbagai aspek kekuatan kata sandi yang meliputi panjang kata sandi, keberadaan huruf besar, huruf kecil, karakter khusus, dan keberadaan angka.

Algoritma melakukan evaluasi terhadap:
- Panjang kata sandi
- Kehadiran huruf besar dan kecil

- Kehadiran karakter khusus
- Kehadiran angka

Hasil evaluasi algoritma menetapkan skor berdasarkan kriteria yang dipenuhi oleh kata sandi. Jika kata sandi memenuhi semua kriteria yang ditetapkan, skor maksimum (8) diberikan, menunjukkan kata sandi sangat kuat. Namun, jika kata sandi tidak memenuhi satu atau lebih kriteria, skor diberikan sesuai dengan jumlah kriteria yang dipenuhi.

3.2. Analisis Kualitas Kata Sandi

Algoritma yang digunakan mampu memberikan evaluasi yang cermat terhadap kekuatan sebuah kata sandi berdasarkan kriteria yang telah ditetapkan. Pembahasan dilakukan berdasarkan skor yang diberikan serta alasan mengenai kelemahan atau kekuatan dari kata sandi yang dimasukkan.

Pembahasan dilakukan terhadap:
- Panjang kata sandi
- Kehadiran huruf besar dan kecil
- Kehadiran karakter khusus
- Kehadiran angka

Algoritma menunjukkan kelemahan kata sandi dengan memberikan detail terkait kekurangan yang dimiliki kata sandi, seperti panjang yang kurang dari yang diinginkan, absennya jenis karakter tertentu, atau tidak adanya kombinasi tertentu seperti huruf besar, huruf kecil, karakter khusus, dan angka.

3.2.1. Implementasi Kode Java

Penerapan algoritma ini terdapat dalam kode Java yang dirancang untuk menerima input pengguna, yaitu panjang kata sandi yang diinginkan dan kata sandi itu sendiri. Kode Java memberikan skor dan mengklasifikasikan kekuatan kata sandi berdasarkan skor yang diperoleh.

4. KESIMPULAN (10 PT)

Berdasarkan hasil pengujian dan pembahasan pada penelitian ini, dapat disimpulkan bahwa program Java yang dibangun telah berhasil menerapkan algoritma penilaian kekuatan kata sandi. Program mampu menganalisis kata sandi yang dimasukkan pengguna berdasarkan kriteria panjang minimum 8 karakter, penggunaan huruf besar, huruf kecil, angka, dan karakter khusus. Kemudian program memberikan penilaian tingkat kekuatan kata sandi apakah “Kuat” atau “Lemah”. Hasil penilaian sesuai dengan ekspektasi kekuatan masing-masing kata sandi uji.

Dengan demikian, program Java ini telah memenuhi tujuan penelitian yaitu mengimplementasikan algoritma pengukuran kekuatan kata sandi agar dapat membantu pengguna menghasilkan kata sandi yang lebih kuat. Hal ini penting demi meningkatkan keamanan data di era digital.

Beberapa rekomendasi untuk pengembangan lebih lanjut antara lain menambahkan pengecekan terhadap kamus kata umum bahasa Indonesia, menerapkan algoritma tingkat kerumitan yang lebih canggih, memberi sugesti kata sandi acak yang kuat, dan pengembangan antarmuka yang lebih intuitif. Implementasi rekomendasi ini diharapkan dapat menyempurnakan program Java penilaian kekuatan kata sandi agar semakin bermanfaat bagi masyarakat luas.

REFERENSI (10 PT)

[1] R. K. L. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. S. Lee, "TrustCloud: A Framework for Accountability and Trust in Cloud Computing," 2011 IEEE World Congress on Services, 2011, pp. 584–588.
[2] Risk Based Security, "2019 Year End Data Breach QuickView Report," Risk Based Security, 2020.
[3] W. E. Burr, D. F. Dodson, and W. T. Polk, "Electronic Authentication Guideline," NIST Special Publication 800-63-2, pp. 1–52, 2017.
[4] NordPass, "200 Most Common Passwords of 2020," NordPass, 2020.
[5] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, "Password Cracking Using Probabilistic Context-Free Grammars," 2009 30th IEEE Symposium on Security and Privacy, 2009, pp. 391–405.
[6] S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman, "Of Passwords and People: Measuring the Effect of Password-Composition Policies," Proc. CHI 2011, 2011, pp. 2595–2604.
[7] R. K. Thomas and R. G. Gursimran, "Password Security Analysis Based on Pattern of User's Password Using Shannon's Entropy," 2012 International Conference on Computer Communication and Informatics, 2012, pp. 1-4.
[8] W. E. Burr, D. F. Dodson, and W. T. Polk, "Electronic Authentication Guideline," NIST Special Publication 800-63-2, pp. 53-56, 2017.
[9] M. Weir, S. Aggarwal, M. Collins, and H. Stern, "Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords," Proceedings of the 17th ACM Conference on Computer and Communications Security, 2010, pp. 162–175.
[10] J. Bonneau, "The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords," 2012 IEEE Symposium on Security and Privacy, 2012, pp. 538–552.
[11] D. Wheeler, "zxcvbn: Low-Budget Password Strength Estimation," Proc. 25th USENIX Secur. Symp., pp. 157–173, 2016.
[12] R. Hidayat, "Cara Membuat Pseudocode Algoritma Pemrograman," Ilmu Komputer, vol. 104, no. 6, pp. 14–18, Nov. 2018.
[13] NordPass, "200 Most Common Passwords of 2020," NordPass, 2020.
[14] R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner's Approach. New York: McGraw-Hill Education, 2015.
[15] J. A. Whittaker, "What is Software Testing? And Why is It So Hard?," IEEE Software, vol. 17, no. 1, pp. 70–79, Jan. 2000.
[16] U. Engel, Graphics and Visualization: Principles & Algorithms. Boca Raton: CRC Press, 2015.
[17] W. Gander et al., Scientific Computing. Berlin: Springer, 2018.