WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Password Protecting Application for Prevention of Password from Unauthorized User

[1]Manoj K. Vairalkar, [2]SurajKowale, [3]Priyanka Samarth, [4]Akshay Pothare, [5]Raj Choure

[1]Assistant Professor,[2,3,4,5]UG Students
[1,2,3,4,5]Department of Computer Science & Engineering
[1,2,3,4,5]Govindrao Wanjari College of Engineering and Technology Nagpur, India

ABSTRACT

To protect user passwords, password managers utilise a master password or security key. When Master Password users utilise a weak Master Password, their security is put at risk. Security keys provide limited benefits because users must always use a token to log in. Using a user's server and secret sharing, the technology we show in this article offers high security and usability while being completely innovative. Ten years after its creation, password protection systems are still open to multiple attacks. SECRYPT is a comprehensive encrypted password manager that protects your privacy and security. safeguards the user's cloud service password in a password file. The SaaS functionality of the programme on the cloud improves its usefulness and dependability. Each and every password kept in our software is encrypted and kept in a cloud database. For enhanced security, numerous databases house your passwords and encrypted keys. Therefore, there is a lot of thievery happening all around the world. People worry about keeping anything important in their homes. As a result, a lot of people choose to keep money in banks. However, even banks aren't sufficiently secure in this unstable society to suit people's needs. If there is performance in safety, it is not unusual for a guy to feel that his treasures are safe. As a result, these studies can provide excellent security at a low cost.

Keywords: Android, Windows, Java, security, SQL server, Protection

1. INTRODUCTION

Password management is a set of principles and best practices to be followed by users while storing and managing passwords in an efficient manner to secure passwords as much as they can to prevent unauthorized access. Passwords are a set of strings provided by users at the authentication prompts of web accounts. Although passwords still remain as one of the most secure methods of authentication available to date, they are subjected to a number of security threats when mishandled. The role of password management comes in handy there. Password management is a set of principles and best practices to be followed by users while storing and managing passwords in an efficient manner to secure passwords as much as they can to prevent unauthorized access. Usernames and passwords arestill essential elements of strongcomputer authentication. Mobile work and the internet use tons of reuse every day to maintain and improve the current activityshown in their medications. However, the behavior of negative letters leadsto many ambiguous numbers. The work is believedtobethe result of claims fromsellers and trust from legal researchers, such as in lawsuitsfiledby LinkedIn Therefore, this information is prepared in a way that is clear and easy foraddicts to remember. This fact creates lowentropy secrets and causes online prediction attacks or offline cracks. It is software that enables pharmaceutical companies to create, store and manage credentials for startups and online services. Simplifies the creation, replication of strong certificates, and storingthem in translation files or calculations for installablepackages. Understanding what a manager is, why it's useful, and how it works can help addicts gettheirdevice up andrunning as quickly as possible. People arestoring more and more personal information online (emails, phone calls, calendars, documents, photos, etc.). Protecting the privacy of personal dataontheInternet is very important. It's also difficult because the average internet user doesn't understand the consequences of using an unreliableservice and cares more about simplicity. Thewebsitessharecustomergenerated information with partners, there is a vulnerability that could lead to information disclosure, but users areawareof this risk and use their services to send sensitive information to unsecuredandtrustedwebsites. Our goal is to create protection.Modernencryption relies on the use of computers because it is very difficult forhumans to crack codes with computers. There are currently two main types of encryption: symmetric key encryption and public key encryption. Symmetric key encryption occurs when each computer has a secret key that it can use to encrypt data before sending it over the network to another system. Todothis, you need to know which computers need to beconnectedtogether so that each computerhasaswitch.Both computers must know the same passwordtodecryptthefile. This rule is the key to decrypting the message. A simple example is getting the ASCII value of each character.

2. LITERATUREREVIEW

S. Prakash, A. Gupta, and R. Garg, "Password Cracking and Detection Techniques: A Comparative Study" (2015):

This study provides an overview of password cracking techniques and analyzes various password detection methods. The authors discuss the importance of incorporating password complexity rules, brute-force attack prevention, and secure storage techniques within password detection applications.

L. T. Nguyen, T. H. Tran, and D. T. Nguyen, "A Password Strength Evaluation System Using Artificial Intelligence" (2017):

This research proposes a password strength evaluation system implemented in Java using artificial intelligence techniques. The authors utilize machine learning algorithms to analyze the strength of passwords and provide feedback to users. The study emphasizes the importance of incorporating AI-based approaches to enhance password detection and security.

H. L. Chen, C. L. Wu, and H. C. Chao, "A Hybrid Password Strength Meter" (2018):

This paper presents a hybrid password strength meter that combines rule-based and probabilistic approaches. The authors propose a system that considers both password complexity rules and probabilistic password guessing techniques to evaluate the strength of passwords. The study showcases the effectiveness of combining different techniques for password detection.

M. J. Farash and R. B. Fisher, "Using Long-Term Memory to Improve Password Strength Meters" (2019):

This research explores the utilization of long-term memory techniques to improve password strength meters. The authors implement a Java-based password detection system that leverages artificial neural networks and machine learning algorithms to predict and detect weak passwords. The study demonstrates the benefits of utilizing memory-based techniques for password detection.

P. N. Dang and N. H. Nguyen, "A Lightweight Password Manager for Enhanced Security" (2020):

This study proposes a lightweight password manager implemented using Java, focusing on enhancing security while maintaining user convenience. The authors discuss techniques for secure password storage, encryption, and password generation. The research emphasizes the importance of proper password management to mitigate security risks.

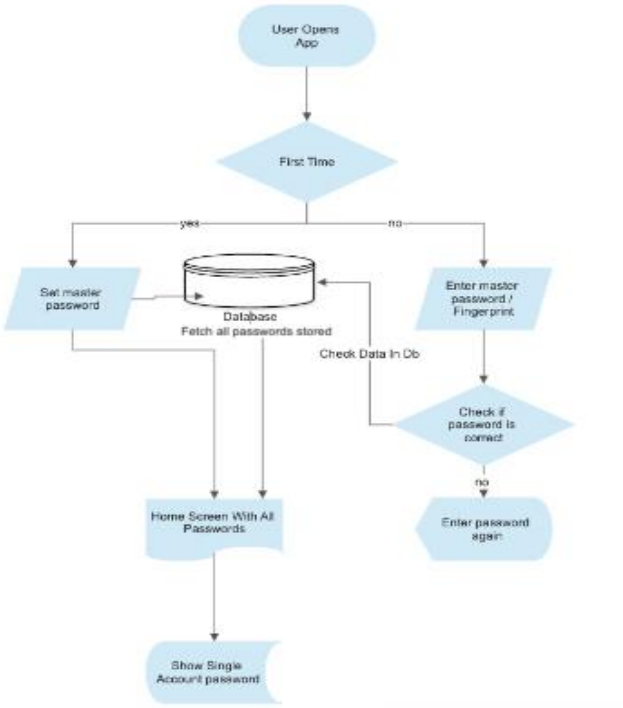
3. FEATURES

- Password Strength Evaluation: Analyze the strength of passwords based on criteria such as length, complexity, and the presence of special characters, uppercase and lowercase letters, and numbers. Provide feedback to users on the strength of their passwords and suggest improvements if necessary.
- Password Hashing and Salting: Securely store passwords by applying cryptographic techniques such as hashing and salting. Hash the passwords using a secure hash function (e.g., SHA-256) and add a unique salt to each password before hashing. This makes it harder for attackers to reverse-engineer passwords even if the hashed values are compromised.
- Brute-Force Protection: Implement measures to prevent brute-force attacks. Enforce account lockouts or temporary delays after a certain number of failed login attempts to discourage attackers from guessing passwords systematically. Provide appropriate error messages and notifications to users when their accounts are locked or under attack.
- Password Policies: Allow administrators or users to configure password policies. Enforce rules such as minimum password length, character requirements, password expiration, and disallowance of common or easily guessable passwords. Educate users about the importance of creating strong passwords and complying with the policies.
- Password Reset and Recovery: Implement a secure password reset and recovery mechanism. Provide options for users to reset their passwords through email verification, security questions, or other validated methods. Ensure the reset process is well-protected to prevent unauthorized access.
- Two-Factor Authentication (2FA): Offer support for additional layers of security by integrating two-factor authentication. Allow users to link their accounts with authentication methods like SMS codes, email verification, authenticator apps, or hardware tokens. This adds an extra layer of verification to the login process.
- Logging and Auditing: Record important security-related events, such as login attempts, password changes, and account activities, in logs. Enable administrators to review and analyze these logs for monitoring and auditing purposes. Ensure the logs are appropriately secured to prevent unauthorized access.
- Error Handling and Validation: Implement robust error handling and input validation mechanisms to handle potential security vulnerabilities. Validate user inputs to prevent common attacks like SQL injection, cross-site scripting (XSS), or command injection. Provide clear error messages to guide users and prevent information leakage.
- User Account Management: Provide functionality for users to manage their accounts. Allow them to update passwords, change security settings, and review account activity. Implement appropriate authorization and access control mechanisms to ensure that users can only perform actions related to their own accounts.

- Security Notifications: Notify users about important security events, such as suspicious login attempts or changes to their account settings. Send email notifications or in-app alerts to keep users informed about any potential security risks and encourage them to take necessary actions.

4. SYSTEM ARCHITECTURE

Design the system architecture and create the necessary design documents such as use case diagrams, class diagrams, and UI mockups. Define the data models, classes, and interfaces required for password detection, storage, and validation. Plan the overall structure of the application, including the division of modules, packages, and dependencies.



Set up the development environment with Java and any required frameworks or libraries.Implement the user interface components based on the design mockups, ensuring a user-friendly and intuitive experience. Implement the password strength evaluation logic, incorporating algorithms or heuristics to analyze password complexity and provide feedback to users. Develop the password storage mechanism, utilizing secure hashing algorithms and salting techniques to store passwords in a secure manner. Implement prevention mechanisms for brute-force attacks, such as setting up login attempt limits, account lockouts, and CAPTCHA integration. Incorporate security measures, including secure session management, SSL/TLS encryption, and protection against common security vulnerabilities. Handle error cases by implementing robust error handling mechanisms and providing appropriate error messages to users. Conduct thorough testing at each stage, including unit testing, integration testing, and security testing, to ensure the correctness and reliability of Application.

5. METHODOLOGY

The software design of a password detection application using Java project involves defining the structure, modules, and interactions of the system components. Here is a suggested software design for such a project:

Use Case Diagram: Create a use case diagram to identify the different user roles and their interactions with the application. This helps in understanding the functionalities and requirements of the system.

Class Diagram: Design a class diagram to represent the classes, relationships, and attributes of the application's components. This diagram captures the overall structure and organization of the software.

Module Design: Identify the major modules or components of the application. This can include modules for user authentication, password validation, password storage, password strength evaluation, and user interface. Each module should have well-defined responsibilities and interfaces.

Package Structure: Organize the classes and modules into logical packages to achieve modularity and maintainability. This helps in grouping related classes and providing a clear structure to the project.

User Interface Design: Design the user interface (UI) components, screens, and forms required for user interactions. Consider the usability and aesthetics of the UI to create an intuitive and user-friendly experience.

Password Validation and Strength Evaluation: Implement the logic to validate user passwords based on predefined rules and criteria. This can include checks for length, character diversity, presence of special characters, and adherence to password policies. Evaluate the strength of passwords using algorithms or heuristics to provide feedback to users.

Password Storage: Define the mechanism for securely storing passwords. This involves using cryptographic hashing algorithms such as bcrypt or Argon2 to store hashed passwords instead of plaintext. Implement proper salting techniques to enhance security.

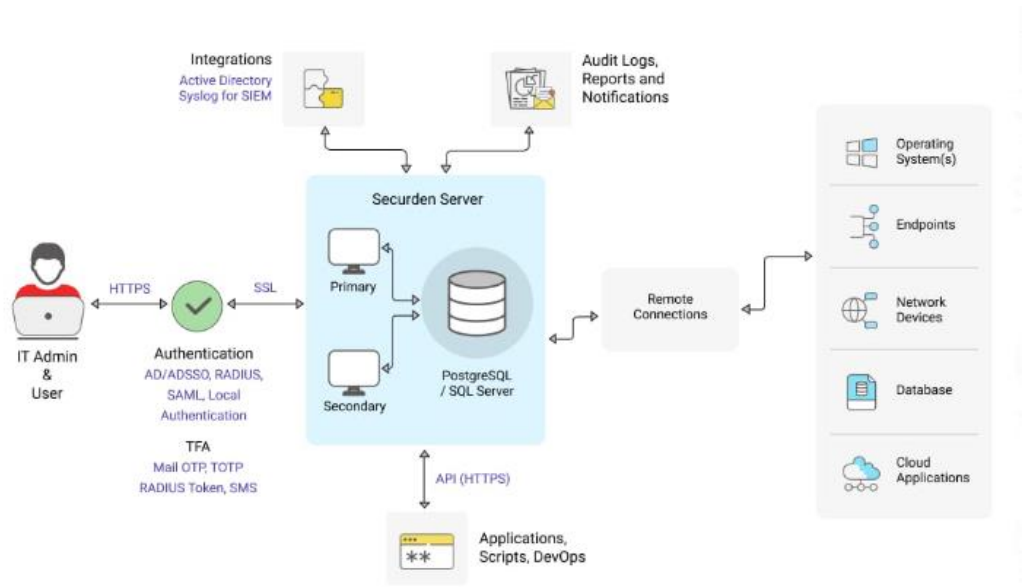
Brute-Force and Dictionary Attack Prevention: Incorporate mechanisms to prevent brute-force and dictionary attacks. This can include setting up login attempt limits, implementing account lockouts, and using CAPTCHA or reCAPTCHA to protect against automated password guessing.

Security Measures: Integrate security measures such as secure session management, SSL/TLS encryption for communication, and protection against cross-site scripting (XSS) and SQL injection attacks.

Error Handling and Logging: Implement robust error handling mechanisms to handle exceptions and provide meaningful error messages to users. Incorporate logging to record important events, errors, and activities for auditing and troubleshooting purposes.

Testing and Quality Assurance: Plan and execute comprehensive testing strategies to ensure the reliability, security, and performance of the application. Use techniques such as unit testing, integration testing, and security testing to identify and fix any issues.

Documentation: Create documentation that includes user manuals, system architecture diagrams, and code documentation to aid in understanding and maintaining the application. The software design of a password detection application using Java involves defining the structure, modules, and interactions of the system components. By following a well-thought-out software design, the project can be developed systematically and ensure the security, usability, and maintainability of the application.



6. SYSTEM REQUIRNMENT

- Operating System: Any operating system that supports Java Development Kit (JDK) can be used. This includes Windows, macOS, and Linux.
- Java Development Kit (JDK): Install the latest version of JDK on your system. The JDK provides the necessary tools and libraries for Java development. You can download it from the Oracle website or use an OpenJDK distribution.
- Integrated Development Environment (IDE): An IDE is recommended to make development easier. Popular choices for Java development include Eclipse, IntelliJ IDEA, and NetBeans. Install your preferred IDE.
- Memory (RAM): The minimum RAM requirement for Java development is usually around 4GB. owever, having more RAM can improve the performance of the IDE and allow you to work with larger projects more smoothly.

- Disk Space: Reserve a sufficient amount of disk space for the development environment, IDE, and project files. A few gigabytes should be enough.
- Processor: A modern processor with multiple cores will help in faster compilation and execution of your Java code.
- Internet Connectivity: Having an internet connection is useful for accessing documentation, libraries, and resources related to Java programming.

7. PROPOSEDSYSTEM

If you create a password-protected website, you should consider protectinguser password information. Even if a person trusts the security of their data, any administrator working on the hostthatowns the data can still access the user's password. Hijackinga password with a simple algorithm increasessecrecy, but not "security". Another method is to encrypt all passwords in the file withanindustry-standardpassword such as Message-Digest Algorithm 5 (MD5). MD5 has its weaknesses.

It is important to understand that password encryption does not protect the website, only the password. Unlessyour website is well secured,encrypting your password will not prevent yourwebsite from being hacked. If the system is compromised, hackers can damage it and gain access to sensitive information, including password databases. However, when this data is stored encrypted, it seems impossible for hackers to use it. Cracking encrypted passwords takes a lot of time and energy, even on modern computers.

These encrypted passwords are then stored. When someone tries to log in, the entered password is re-encrypted and compared to the login in the file where the password is stored. If they match, access is granted. While DES is a two-way encryption algorithm, most erasuresare one-way. If your website isn't well protected, Word's encryption won't protect it from an attack. However, if your system is compromised, hackers can damage it and gain access to non-public information, including sensitive information. However, if this information is saved after translation, hackers cannot use it. Deciphering the definition will take a lot of time and energy,andimmediately save it on the computer. The encryption model is shown in Figure 2. The Tomcat 6 server was used in the study. This server is configured via the URL: http:\\localhost:8080. Tomcat is an open source servlet container developed by the Apache Software Foundation (ASF). Tomcat uses Sun Microsystems' Java Servlets and Java Server Pages (JSP) and provides a "pure Java" HTTP web server environment for running Java code. For registered users, first create a file called start. Jsp on this server.Thenlogin.jspisinstructed to open a login page where the user enters their username and password. This runs the Encrypt Dataservlet,whichoptionallyrunstheEncryptPasswordservlet, which encrypts the user's data, associates it with the stored data, and thencompares the password and username to the database.

8. RESULT ANALYSIS

Define evaluation metrics: Determine the metrics you want to use to measure the performance of your password detection application. Some common metrics include accuracy, precision, recall, and F1 score. Accuracy measures the overall correctness of the predictions, while precision measures the proportion of correctly identified positive instances (true positives) out of all instances predicted as positive. Recall measures the proportion of true positives identified correctly out of all actual positive instances, and the F1 score is the harmonic mean of precision and recall.

- Collect a dataset: Gather a dataset of passwords that you will use to evaluate the application's performance. This dataset should include a variety of password types, such as weak passwords, strong passwords, and potentially compromised passwords.
- Label the dataset: Annotate or label each password in the dataset based on its characteristics. For example, you could label passwords as weak or strong, or indicate if they have been compromised or are commonly used.

- Run the application on the dataset: Execute your password detection application on the labeled dataset. The application should analyze each password and provide feedback or predictions based on its internal algorithms or rules.
- Compare the results: Compare the predictions made by your application with the ground truth labels in the dataset. Calculate the evaluation metrics defined in step 1 to assess the application's performance. This will give you an understanding of how well the application is able to detect weak passwords, strong passwords, compromised passwords, or any other characteristics you are interested in.

9. CONCLUSION

Passwordprotection is difficult. A security breach in a bad service exposed a lot of sensitive information, makingitdifficult to trust the service provider. Whenusedproperly,OAuth concepts can be secure and causesetbacks.OAuth services can be used by small businesses that don't have a largesecuritybudget.Producers should borrow from government efforts to collect, store and issuemill certificates. Despite these unique benefits, the gesture ofsmoking still dominates. Good IT strategies must address the mysteries of low entropy or predictable performance and the ever-growing challenge of BYOD. Security awareness programs can improve prevention by educating employees on precautions and protections.Comprehensiveverification can be extended to other parts of the password implementation, such as:B. Wi-Fi passwords (set by a mysterious or abandoned generation) and session keys. Password Managers are among the most amazingprograms available today. People need it more than ever, but theydon't understand whatit means.Thenumberofhacking attempts is increasing day by day and will continueto increase in the future if nothing is done.

10. ACKNOWLEDGEMENT

It is our proud duty and privilege to acknowledge the kind of help and guidance received from several people in preparation of this report. Itwould not have been possible to prepare this project in this form without their valuable help, co-operation and guidance. First and foremost, wethank our project guide and co-ordinator**MrManojvairarkar**Asst. Professor Department of Computer Science and Engineering, GovindraoWanjari College of Engineering and Technology for their valuable guidance and all the encouragement that lead towards completion of ourproject. We would like to thank **Mr. VivekanandThakare,** HOD, Department of Computer Science and Engineering, GovindraoWanjariCollege of Engineering and Technology for his valuable suggestions and guidance throughout the period of this project. We also wish to recordour sincere gratitude of **Dr. Salim Chavan**, Principal, GovindraoWanjari College of Engineering and Technology for his constant support andencouragementin preparationofthisreportandforprovidingLibraryandlaboratoryfacilitiesneededtoprepareourprojectreport.

Lastbutnotleast,wewouldliketothankouradvisor,friends,parents,teachingandnon-teachingstaffofGWCET.

11. REFERENCES

[1] Y Kumarakalva, P H Sridhar, K KShreekara, et al (2016). Password manager - Automatic password change using headless browsers: A survey, Journal of Emerging Technologies and Innovative Research, 3(12), 66-70, Available at: https://www.jetir.org/papers/JETIR1612008.pdf.

[2] P P. Churi, V Ghate and K Ghag (2015). Jumbling-Salting: An improvised approach for password encryption. 2015 International Conference on Science and Technology (TICST). IEEE, Available at: https://doi.org/10.1109/TICST.2015.7369364.

[3] L Zheng and A C. Myers (2008). Securing nonintrusive web encryption through information flow. Proceedings of the third ACM SIGPLAN workshop on Programming languages and analysis for security, (pp.125-134). ACM, Available at: https://doi.org/10.1145/1375696.1375712.

[4] R Shay, S Komanduri, P Gage Kelley, et al (2010). Encountering stronger password requirements: user attitudes and behaviors. Proceedings of the Sixth Symposium on Usable Privacy and Security, (pp. 1-20). ACM, Available at: https://doi.org/10.1145/1837110.1837113.

[5] S N Patil, R M Vani and P.V.Hunagund (2015). Data security using advanced encryption standard (AES) in reconfigurable hardware for SDR based wireless systems, International Journal of Computer Engineering and Technology, 6(1), 95-100, Available at: https://sdbindex.com/Documents/index/00000005/00000-65359.

[6] R Shay, S Komanduri, A L. Durity, et al (2016). Designing password policies for strength and usability, ACM Transactions on Information and System Security, 18(4), 1-34, Available at: https://doi.org/10.1145/2891411.

[7] R M Balajee, M V Mohan and M K JayanthiKannan (2021). Performance analysis of bag of password authentication using Python, Java and PHP implementation. 2021 6th International Conference on Communication and Electronics Systems (ICCES). IEEE, Available at: https://doi.org/10.1109/ICCES51350.2021.9489233.

[8] N Meghanathan (2013). Identification and removal of software security vulnerabilities using source code analysis: A case study on a java file writer program with password validation features, Journal of Software, 8(10), 2412-2424, Available at: http://www.jsoftware.us/vol8/jsw0810-04.pdf.