Ensuring Password Security: A Comprehensive Guide to Auditing the Hashing Process

Lokesh Kumaran R
20MIS0052
School of Computer Science
Engineering and Information Systems
lokeshkumaran.r2020@vitstudent.ac.in
8072335867

Bavya I
20MIS0060
School of Computer Science
Engineering and Information Systems
bavya.i2020@vitstudent.ac.in
6369151925

Kaushik M
20MIS0306
School of Computer Science
Engineering and Information Systems
kaushik.m2020a@vitstudent.ac.in
6374964793

***Abstract*— Auditing the various hashing schemes used in password storage is a critical aspect of ensuring digital security. This research review covers key aspects of password security, such as the selection of robust hashing algorithms, appropriate work factors, and the strength of the encryption for given password. Password policies that encourage strong, complex passwords, as well as password encryption, are critical components. Additional layers of protection are provided by security audits, monitoring, and compliance with industry standards. Regular evaluations, user education, and an incident response plan all contribute to a robust system that protects against un-authorised access and data breaches while also promoting secure and reliable authentication. This paper along with the formal process of auditing is able to efficiently make the experiment more formal process and we make the process simplified so as to make the understanding of the process easier for educational purpose. This paper provides a review of mostly used password-hashing schemes in todays world and a relevant performance evaluation analysis on a common setting in terms of memory consumption, and execution time and their efficiency.**

Keywords—Auditing, hashcat, bruteforce ,md5,sha, Django framework, bcrypt.

I. INTRODUCTION

In an age dominated by the digital realm, the security of user credentials and sensitive information has become a top priority. Passwords are the first line of defence against unauthorised intrusion. They are alphanumeric keys that grant access to personal accounts, confidential data, and sensitive systems. However, the reliance on these keys has made them prime targets for cybercriminals and hackers, who use increasingly sophisticated methods to breach security.
The cryptographic process of hashing has emerged as a cornerstone in safeguarding sensitive information in order to mitigate the risks associated with password security. Hashing converts user passwords into fixed-length strings of characters known as hash values. These hash values are stored in place of plain-text passwords, making them unreadable to anyone who is not authorized. In essence, hashing strengthens password security by making it exponentially more difficult for adversaries to recover the original passwords.

However, not all hashing algorithms are created equal when it comes to safeguarding user data, and the cybersecurity landscape is changing at a faster rate than before. To ensure the highest level of security, this necessitates a rigorous and ongoing examination of the effectiveness of various hashing schemes. This imperative serves as the foundation for this research paper, which compares different hashing schemes in the context of password security auditing.

As we conduct this comparative analysis, we hope to discover the strengths and weaknesses of various hashing algorithms, as well as their suitability for protecting sensitive user data. This investigation will look into a variety of critical aspects, such as security, performance, salting and "peppering," adaptability to future threats, compliance with industry standards, real-world implementations, and recommendations for optimal usage. By evaluating these aspects, we hope to provide valuable insights that will enable organisations and individuals to make informed decisions about password security.

The selection of a hashing scheme is far from trivial; it is a decision with far-reaching implications for a system's overall security. As a result, the purpose of this research paper is to contribute to the ongoing debate on password security by providing recommendations and guidance that can aid in strengthening the

protection of digital assets and user privacy in an increasingly interconnected and dangerous digital landscape.

The objective of this research paper is to conduct a comprehensive audit of different hashing schemes used in password security, examining their strengths, weaknesses, and suitability for protecting sensitive user data. We will compare and contrast popular hashing algorithms such as algorithms like MD5 ,SHA, Django framework based on sha256 and bcrypt.

II. LITERATURE SURVEY

A. Paper-1:

"A one-way hashing algorithm with variable length output:"

A novel one-way hashing method called HAVAL has been proposed in this study. It may compress a message of any length into a fingerprint consisting of 128, 160, 192, 224, or 256 bits. Moreover, HAVAL offers the freedom to alter the number of passes message blocks are processed in order to accommodate different practical applications. The message digest or fingerprint is represented by the output value. Additionally, they anticipate that the approach will be widely applied in real-world scenarios requiring fingerprints with varying lengths.

B. Paper-2:

"An Improved Scheme Of Single Sign-On Protocol Based On Dynamic Double Password"

In order to combat password attacks, the document suggests a better single sign-on approach for different application systems that uses a dynamic double password and updates data through ticket analysis. The suggested method not only has a higher ability to detect and analyse replay attacks, but it also significantly improves the system's defence against password attacks without adding to the cost or complexity of the user experience. The suggested scheme's architecture and security analysis are presented in the document. The suggested plan is evaluated against the industry-standard Kerberos protocol, and the analysis of the results shows that the new plan is more effective and safe

C. Paper-3:

*"**Password Strength: An Empirical Analysis"***

Emphasising the empirical analysis of real-world passwords from three datasets that differ in user localization and application domain. The study examines the relationship between the search space and the quantity of passwords broken using a variety of guessing strategies, such as Markov chains, probabilistic context-free grammars, dictionary attacks, and brute force. Using three sizable password datasets that differ in terms of application domain and user localization, the authors cross-validate their investigations. The study offers a distinctive and thorough examination of internet applications' password strength. In order to compare the efficacy of various password cracking methods, the authors also assess them using homogeneous measures across various datasets.

D. Paper-4:

*"**Simple Authentication for the Web"***

The material offered is a research article that addresses the creation and application of the SAW protocol for website logins. The paper is broken up into multiple sections, such as a thorough analysis of SAW's objectives and architecture, a prototype implementation of SAW, a thorough assessment of the risks to this system, a few of SAW's more sophisticated features, and how SAW gets over the challenges that email-based authentication presents. The paper includes conclusions, a discussion of future work, and a discussion of related work. The National Science Foundation provided money for this investigation, which the authors acknowledge. They also appreciate a number of people for their contributions. A list of references is also included in the article, and it contains materials on public key infrastructure, access control, and email-based authentication.

E. Paper-5:

*"**Online gaming cheating and security issue"***

A passage taken from ITCC 05, the International Conference on Information Technology: Coding and Computing. It talks about different measures that sellers can take to improve the security of virtual currency trade and online gaming. A list of references that offer more information on the subject, such as publications and websites, is also included in this section. Beth Noveck's "Virtual World Governance and Democracy," Celeste Biever's "Sales in virtual goods top $100 million," and Valve Corp.'s Counter Strike Online are a few of the references.

F. *Paper-6:*

"E-Tax Invoice System Using Web Services Technology: A Case Study Of The Revenue Department Of Thailand"

A review of previous research on the "e-Tax Invoice System" that uses Web Services technology is included in the text of 5.pdf. The poll covers the present tax invoice process as well as the potential of the proposed solution to address issues with the current procedure. Additionally, it emphasises how information technology is improving national tax systems and is always creating new information systems to make tax collection easier, especially with regard to Value Added Tax (VAT). Examples of web service technology in tax departments are also given in the survey, including the Dutch Taxonomy Project (DTP) and the Canada tax Agency (CRA). References to similar publications are also included in the survey, such as Chuang-Cheng Chiu and Chieh-Yuan Tsai's "A Web Services-Based Collaborative Scheme for Credit Card Fraud Detection".

G. *Paper-7:*

"Hashing Algorithms for Large-Scale Learning"

In this paper, they propose a b-bit min-wise hashed compact representation of sparse, binary data sets. This representation can be easily integrated with linear learning algorithms, like logistic regression and linear SVM, resulting in significant reductions in training time and/or resource requirements.Additionally, they contrast b-bit minwise hashing with the Vowpal-Wabbit (VW) and Count-Min (CM) sketch algorithms, all of which have variances that are (basically) equal to random projections based on our research [24]. Our evaluations, both theoretical and empirical, show that b-bit minwise hashing produces far better results for binary data (at the same storage).

H. *Paper-8:*

*"**Providing Password Security By Salted Password Hashing Using Bcrypt Algorithm"***

The goal of this project is to use the salted password hashing technique to secure user data. Online shopping can be extremely risky because customer data is stored in a database in plain text. Using hashing, this situation is avoided. The goal of this project is to save user data in an encrypted format in a database as opposed to storing it in plain text. The Bcrypt algorithm is used to increase the security of user data. Up to 512 bits of data can be encrypted using the Bcrypt technique, giving users a longer encryption key and a hashed version of their data. In hash tables, hash functions are mostly used to find data records rapidly.

I. *Paper-9:*

"A novel secure hash algorithm for public key digital signature schemes"

This essay suggests Of all the cryptographic primitives, hash functions are the most widely utilised; they are present in many cryptographic systems as well as security protocols. The output length of SHA-192 is 192 in its basic architecture.The SHA-192 has been engineered to meet various levels of heightened security requirements and fend off sophisticated SHA assaults. When compared to the previous NIST security study, the SHA-192 security analysis provides greater security. Numerous applications, including public key cryptosystems, digital sign encryption, message authentication codes, random generators, and security architecture for soon-to-be-released wireless devices like software-defined radios, can make use of SHA-192.

J. *Paper-10:*

"FPGA Implementation Of MD5 Hash Algorithm"

This project explains message authentication, a crucial method for confirming that communications you receive are real and unaltered from their purported source. The processing of the input message, which can be of any length, involves manipulating 128-bit blocks in 64 stages to process it in 512-bit blocks. Building cryptographic accelerators using hardware HMAC implementations based on hash algorithms like MD5 makes sense. Field programmable gate arrays (FPGAs) have been used to build two different topologies of MD5, namely iterative and full loop unrolling. It is discussed how well these implementations perform.

K. *Paper-11:*

*"**Security Analysis of MD5 algorithm in Password Storage"***

They covered a variety of strategies in their study to counter these attacks: The following techniques are used to slow down the MD5 computation: (1) using salts; (2) using strong passwords to reduce the likelihood that they will appear in dictionaries; (3) key stretching and iteration hashing; (4) chaining method, which uses a different initialization vector for each password and uses the output of one iteration as the input of the next iteration; (5) encrypting the password before hashing; and (6) XOR cypher to prevent the final hash value from being found in any rainbow table. The suggested method is implemented with Microsoft SQL

Server as the database and C# as the programming language. The attacker will now need to use the suggested method to hack into the database, the server contain-ing the application code as well as the external file.

L. *Paper-12:*

"Evaluation of Password Hashing Schemes in Open Source Web Platforms"

This study assessed the hashing algorithms' security for well-known content management systems and web application frameworks. A straightforward approach was suggested by them to measure the time and expense involved in breaking passwords. The suggested framework examines both brute force and dictionary attacks, taking into account the key elements of password guessing assaults. Salt, terations, hashrate, length of password, number of hashes, number of guesses, and efficacy are some of these parameters. They came to the conclusion that most web application and CMS frameworks lack secure de-fault options.further looked at the possibility of hashing techniques being abused to cause denial-of-service attacks. Lastly, they offered a list of recommended practises and substitutes to improve the security of password storage.

M. *Paper-13:*

"Polypassword hasher Improving Password Storage Security"

This project is about hashed passwords from password databases being stolen and then cracked. It is necessary to update the strategies that have made cracking challenging because systems have become quicker and attackers have obtained access to clusters or specialised gear used for cracking. They have developed a technique called PolyPass-wordHasher that adds an extra encryption stage by using shared keys. To exploit it, an attacker must simultaneously crack several keys. According to their projection, PolyPasswordHasher extends the time required to crack passwords, even those that are short, beyond what is thought to be the present universe's age.

N. *Paper-14:*

"Highly Secured Blockchain Based Electronic Voting System Using SHA3 and Merkle Root"

This work has created an efficient methodology based on block-chain block creation and sealing for producing a block for each vote. The votes for each candidate they have recommended can be counted using the sealed blocks. In order to address the drawback of using an electronic voting machine (EVM) to cast votes, the system has brought attention to the fact that there is no method to ensure that votes are not cast fraudulently and that only the voter has access to their personal information and the votes they have cast.All the data regarding the electronic voting system is highly safe in the blockchain thanks to the use of SHA3, RSA, and the Merkel root algorithm. Even though blockchains are very safe when it comes to protecting the data that is stored on them.

O. *Paper-15:*

"Hashing Algorithms for Large-Scale Learning"

The authors of this paper suggest a compact representation of sparse, binary data sets using b-bit minwise hashing. This representation can be seamlessly integrated with linear learning algorithms, like logistic regression and linear SVM, resulting in significant reductions in training time and/or resource requirements.Additionally, they contrast b-bit minwise hashing with the VowpalWabbit (VW) and Count-Min (CM) sketches, which, based on our study, all have variances that are essentially the same as random projections [24]. For binary data, b-bit minwise hashing is noticeably more accurate (at the same storage), according to our theoretical and actual assessments.

P. *Paper-16:*

"A simple password authentication scheme based on geometric hashing function"

Password authentication is crucial to prevent unauthorized access to resources. Many schemes use smart cards, but they are not suitable for many internet applications due to their high cost and inconvenience. Existing password authentication schemes, such as hash functions, are vulnerable to replay, man-in-the-middle, and verifier stolen attacks. This paper focuses on designing a simple, secure, and high-efficiency scheme based on geometric hash function without a smart card. The scheme is simple, efficient, and can withstand various attacks.

Q. *Paper-17:*

"Password-Hashing Status."

Computers are used for everyday activities, with one-factor authentication being the common choice. However, poor password management practices can be exploited by attackers, exposing user credentials. Password-hashing techniques, such as PBKDF2, Bcrypt, and Scrypt, are used to fortify user-related information. The Password Hashing Competition (PHC) aims to identify more secure password-hashing schemes for widespread adoption. This paper reviews password-hashing schemes and analyzes their

performance in terms of code size, memory consumption, and execution time.

R. *Paper-18:*

"Integrated Password-based Algorithms with Auditing Capability for Database Applications."

This research aims to protect database confidentiality and integrity by developing a security application. The first layer is a password-based system, authenticating user IDs. Encryption using MD5 hashing and salt hashing algorithms prevents passwords from being exposed and increases the difficulty of offline dictionary attacks. Salt hashing prevents duplicate passwords and generates a different password for adversaries. An auditing mechanism is embedded to monitor all transactions and operations within the database, ensuring data integrity.

S. *Paper-19:*

*"**Analysis of Secure Hash Algorithm (SHA) 512 for Encryption Process on Web Based Application."***

The MD5 hash function used in web-based applications' login mechanism has weaknesses in preventing collision attacks, potentially compromising data security. To improve reliability and system security, the SHA 512 method is used. Data was collected through literature, internet, and observation. The research method includes needs analysis, system vulnerability analysis, and improvement analysis. The new hash function is implemented, and testing results shows a clear view of the system.

T. *Paper-20:*

"A Comparative study. International Journal of Computing Sciences Research, 5(1), 407–425."

This paper concentrates on the comparison of security auditing tools specifying password cracking tools based on different matrices. Passwords are the most popular and dominant means of access control in every authentication process. Every password is vulnerable in the virtual world; all we can do is to delay it for one to break into us. Password cracking used in two opposite intentions; either it can be used for an administrator to protect from unauthorized access and for users to recover forgotten passwords or for an intruder to break into a secure system

III. CHALLENGES:

- Evolving Threat Landscape: Researchers must keep up to date and modify their work to address new threats because cybercriminals are always evolving in their strategies and capabilities.

- Usability vs. Security Trade-off: A constant challenge is finding the ideal balance between robust security and user-friendly experiences, since overly complicated security measures may turn off users.

- Regulatory Compliance: Researchers have to navigate ever-changing and complex data protection regulations to make sure their findings comply with industry standards and legal requirements.

- Interdisciplinary Collaboration: Although it can be difficult because of different viewpoints and terminologies, combining knowledge from the domains of cybersecurity, user experience, and cryptography is essential for thorough research.

- Quantum Computing Threat: The development of post-quantum secure alternatives is a major challenge in preparing for the possible impact of quantum computing on cryptographic systems.

- Scalability: New solutions are needed to match password security protocols to the needs of distributed, modern systems and cloud environments.

- User Education and Behavior: It is a constant struggle to get people to adopt secure password practices and change their behaviour, particularly when dealing with ingrained habits and password fatigue.

- Ethical Considerations: When conducting experiments related to password security, researchers must navigate ethical considerations, particularly when such experiments involve real-world data or vulnerabilities.

- Cross-Platform Compatibility: Due to varying implementations and user habits, it can be difficult to ensure consistent password security practises across different platforms and services.

- Privacy Concerns: Even when collecting and storing user data for research purposes, researchers must ensure participant anonymity while addressing privacy concerns.

IV. Definitions of algorithm used.

A. Md-5:

An input (a message or any other type of data) is fed into MD5, which generates a fixed-length 128-bit (16-byte) hash value that is commonly expressed as a 32-character hexadecimal integer. The main function of MD5 is to generate a fixed-size hash from an arbitrary quantity of data. This hash is frequently utilised in digital signatures, data integrity verification, and password storing applications.

B. Sha-1:

Digital signatures and certificate authority were among the many uses for which SHA-1 was initially created to ensure data integrity and security.SHA-1 generates a fixed-length 160-bit (20-byte) hash value from an input message or set of data. Its main objective, similar to that of other cryptographic hash functions, is to take an arbitrary amount of data and generate a fixed-size hash, which is commonly expressed as a 40-character hexadecimal integer.

C. Sha256

The cryptographic hash function known as "Secure Hash Algorithm 256-bit," or SHA-256, is a member of the SHA-2 family of hash functions. The National Institute of Standards and Technology (NIST) publicised it after the National Security Agency (NSA) created it. Numerous security and cryptography applications make extensive use of SHA-256.SHA-256 generates a fixed-length 256-bit (32-byte) hash value from an input message or set of data. Its main objective, similar to that of other cryptographic hash functions, is to take an arbitrary amount of data and generate a fixed-size hash, which is commonly expressed as a 64-character hexadecimal integer.

D. Sha512

The cryptographic hash function known as "Secure Hash Algorithm 512-bit," or SHA-512, is a member of the SHA-2 family of hash functions. It is SHA-256 upgraded to a higher level of security. The National Institute of Standards and Technology (NIST) published SHA-512, which was created by the National Security Agency (NSA).
SHA-512 generates a fixed-length 512-bit (64-byte) hash value from an input message or set of data. Its main objective, like that of other cryptographic hash functions, is to take an arbitrary amount of data and generate a fixed-size hash, which is commonly expressed as a 128-character hexadecimal integer.

E .Django framework:

A password hashing technique called PBKDF2-SHA256 (Password-Based Key Derivation Function 2 with SHA-256) is utilised by Django to safely store user passwords in a web application's database. A prominent Python web framework called Django offers built-in support for safe password hashing, with one of the supported methods being PBKDF2-SHA256. It offers a significant degree of protection. Because of its high computational cost, dictionary or brute force password cracking is challenging for attackers. When distinct salts are used for every user, it guarantees that even if two users have the same password, the salts will cause their hashes to vary.

F. Bcrypt:

Bcrypt is presumably used by web apps and other systems to protect user credentials. In addition to ensuring user credentials remain secure in the event of a data breach, it also lessens the likelihood of numerous common password-related issues.The well-known and incredibly secure password hashing method bcrypt is designed to protect user passwords by making them difficult to crack using dictionary or brute-force attacks. In software systems and online applications, it is often used to securely store and verify user credentials.

V. Auditing Process:

1. *Collect Passwords:*

   Gather a representative sample of user passwords from your system. We for our experiment we consider the password: “pass@123”.

2. *Assess Password Policies:*

   understand the minimum and maximum password length, complexity requirements (e.g., uppercase, lowercase, digits, special characters), and expiration rules. In our case we use symbol, number and lowercase variables.

3. *Generate Wordlist:*

   Create a dictionary or wordlist of commonly used passwords, dictionary words, and patterns that attackers might try when attempting a brute-force attack.

4. *Select Brute-Force Tools:*

   Choose one or more password-cracking tools that are commonly used for brute-force and dictionary attacks. These tools will help simulate

potential attacks on your passwords. We for experiment we use hashcat.

5. *Crack Passwords:*

   Use the selected tools to attempt to crack the collected passwords using the generated word-list and different attack methods.
   In our case we tend to use optimizer like zero-byte.
   early-skip, not-Salted, not-iterated, single-hash, single-salt, brute-force and raw-hash.

6. *Measure Success Rate:*

   Record and analyze the success rate of the password cracking attempts. Identify which passwords were cracked and how long it took to crack them.

7. *Analyse Password Complexity:*

- Lowercase:4.
- Symbol:1
- Number:3

8. *Recommend Improvements:*

   Provide recommendations for improving password security based on the weaknesses identified. Password length can be increased with increasing complexity and different hashing techniques with hybrid can be used.

9. *Educate Users:*

   Conduct user awareness and training programs to help users understand the importance of strong passwords and teach them how to create and manage secure passwords.

10. *Monitor and Respond to Anomalies:*

    Implement monitoring for unusual login attempts, failed logins, or suspicious activity and have response procedures in place to react to potential security breaches.

VI. EXPERIMENTS:

Mostly the passwords that we enter into a website are not stored directly in the database. The database on other hand has the hashed value of the entered password so that the password in the database remains a bit protected and the hashed value is decrypted then processed when the registered user enters a website. Thus, for the purpose of auditing the password security and strength, we just store a chosen password and hash them with mostly used hashing algorithm and attack them with an optimized attack by using a penetration tool. The result that we get are to be compared with each other and thus the explain their performance for providing a clear picture of the strength of different hashing algorithm

Requirements:

1) ***Find the parameter by which we are going to compare to find the efficient hashing parameter:*** For our project we able to find the time taken to retrieve the password,for this purpose we try to perform penetration testing attack that is optimized by:

- Zero-byte.
- Early-skip.
- Not-Salted.
- Not-Iterated.
- Single-hash.
- Single-salt.
- Brute-Force.
- Raw-hash.

2) ***Technical details:***

a) Selected tool: hashcat-6.2.6

b) System specification:

- System specs:
- Processor: 10th Gen Intel Core i5
- RAM: 8 GB
- Storage: 256GB M.2 PCIe SSD / 128GB M.2 PCIe SSD +1TB SSD hybrid / 2TB SATA HDD
- Operating System: Windows 11

c) Storage necessary criteria:

Intel(R) UHD Graphics,1536/3187 MB (796 mb alloctable),32MCU.

d) Identify the algorithm:

The hashing methods we are taking for the projects are given below:

- Md5.
- Sha 1.
- Sha 256.
- Sha 512.
- Django(PBKDF2-SHA256).
- Bcrypt.

3) ***Procedure:***

1) Install hashcat for the system(windows 11).For our system,the version of hashcat is 6.2.6.

2) After installation, extract the files and you will be able to see the hashcat-6.2.6 folder.

3) Take a text and encrypt a text using hashing methods like Sha, md5, Django based method.

4) Save the encrypted text in the hashcat-6.2.6 folder and enter your terminal and change the course of directory to hashcat folder.

5) Consider the we create a demo file in which the text “pass” is encrypted in md5 format.

6) The command that can be used is as follows:

“C:\<User>\hashcat-6.2.6>hashcat.exe -m 0 demo.txt -a 3 -1 ?u?l?d ?1?1?1?1”.

7) Sample
Example:”pass”
Model:md5.

Encrypted text:
1a1dc91c907325c69271ddf0c944bc72

```
1a1dc91c907325c69271ddf0c944bc72:pass

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 1a1dc91c907325c69271ddf0c944bc72
Time.Started.....: Sun Nov 05 19:17:57 2023 (0 secs)
Time.Estimated...: Sun Nov 05 19:17:57 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?1?1?1 [4]
Guess.Charset....: -1 ?l, -2 Undefined, -3 Undefined, -4 Undefined
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  6417.9 kH/s (0.15ms) @ Accel:128 Loops:26 Thr:32 Vec:1
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 26624/456976 (5.83%)
Rejected.........: 0/26624 (0.00%)
Restore.Point....: 0/17576 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-26 Iteration:0-26
Candidate.Engine.: Device Generator
Candidates.#1....: sari -> xmgg

Started: Sun Nov 05 19:17:53 2023
Stopped: Sun Nov 05 19:17:59 2023
```

4) ***Experimental result:***

1)MD5:

```
Next dictionary / mask in queue selected. Bypassing current one.

Session..........: hashcat
Status...........: Bypass
Hash.Mode........: 0 (MD5)
Hash.Target......: dc06698f0e2e75751545455899adccc3
Time.Started.....: Wed Nov 08 12:40:56 2023 (2 secs)
Time.Estimated...: Mon Nov 20 10:49:37 2023 (11 days, 22 hours)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?1?1?1?1?1?1?1 [8]
Guess.Charset....: -1 ?u?l?d, -2 Undefined, -3 Undefined, -4 Undefined
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   212.0 MH/s (6.45ms) @ Accel:128 Loops:32 Thr:16 Vec:1
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 343932928/218340105584896 (0.00%)
Rejected.........: 0/343932928 (0.00%)
Restore.Point....: 0/916132832 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:5248-5280 Iteration:0-32
Candidate.Engine.: Device Generator
Candidates.#1....: qyaerane -> j20iaane
```

2)Sha1:

```
Next dictionary / mask in queue selected. Bypassing current one.

Session..........: hashcat
Status...........: Bypass
Hash.Mode........: 100 (SHA1)
Hash.Target......: 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
Time.Started.....: Wed Nov 01 11:48:48 2023 (2 secs)
Time.Estimated...: Fri Nov 24 09:03:53 2023 (22 days, 21 hours)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?1?1?1?1?1?1 [8]
Guess.Charset....: -1 ?u?l?d, -2 Undefined, -3 Undefined, -4 Undefined
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   110.4 MH/s (5.97ms) @ Accel:64 Loops:32 Thr:16 Vec:1
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 159383552/218340105584896 (0.00%)
Rejected.........: 0/159383552 (0.00%)
Restore.Point....: 0/916132832 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:4864-4896 Iteration:0-32
Candidate.Engine.: Device Generator
Candidates.#1....: 8taerane -> QLAz5778
```

3)Sha 256:

```
Next dictionary / mask in queue selected. Bypassing current one.

Session..........: hashcat
Status...........: Bypass
Hash.Mode........: 1400 (SHA2-256)
Hash.Target......: 5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a1...1542d8
Time.Started.....: Wed Nov 01 11:24:34 2023 (2 secs)
Time.Estimated...: Tue Dec 19 11:42:59 2023 (48 days, 0 hours)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?1?1?1?1?1?1?1 [8]
Guess.Charset....: -1 ?u?l?d, -2 Undefined, -3 Undefined, -4 Undefined
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 52633.6 kH/s (6.42ms) @ Accel:32 Loops:8 Thr:64 Vec:1
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 106954752/218340105584896 (0.00%)
Rejected.........: 0/106954752 (0.00%)
Restore.Point....: 0/916132832 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:1632-1640 Iteration:0-8
Candidate.Engine.: Device Generator
Candidates.#1....: e23erane -> o23iaane
```

4)Sha 512:

```
Next dictionary / mask in queue selected. Bypassing current one.

Session..........: hashcat
Status...........: Bypass
Hash.Mode........: 1700 (SHA2-512)
Hash.Target......: b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e0...acbc86
Time.Started.....: Wed Nov 01 11:57:38 2023 (3 secs)
Time.Estimated...: Fri Jun 28 02:51:05 2024 (239 days, 14 hours)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?1?1?1?1?1?1?1 [8]
Guess.Charset....: -1 ?u?l?d, -2 Undefined, -3 Undefined, -4 Undefined
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 10546.2 kH/s (8.01ms) @ Accel:32 Loops:8 Thr:16 Vec:1
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 31588352/218340105584896 (0.00%)
Rejected.........: 0/31588352 (0.00%)
Restore.Point....: 0/916132832 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:1928-1936 Iteration:0-8
Candidate.Engine.: Device Generator
Candidates.#1....: 0veerane -> k70mtine

Started: Wed Nov 01 11:56:51 2023
Stopped: Wed Nov 01 11:57:43 2023
```

5)Django framework:

```
Session..........: hashcat
Status...........: Bypass
Hash.Mode........: 10000 (Django (PBKDF2-SHA256))
Hash.Target......: pbkdf2_sha256$390000$E1AQCq34JRR1BKAxz0eFan$u3+pPwv...zbRNg=
Time.Started.....: Wed Nov 01 11:30:37 2023 (2 secs)
Time.Estimated...: Next Big Bang (> 10 years)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?1?1?1?1?1?1?1 [8]
Guess.Charset....: -1 ?u?l?d, -2 Undefined, -3 Undefined, -4 Undefined
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:       65 H/s (6.62ms) @ Accel:32 Loops:8 Thr:32 Vec:1
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 0/218340105584896 (0.00%)
Rejected.........: 0/0 (0.00%)
Restore.Point....: 0/3521614606208 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:1760-1768
Candidate.Engine.: Device Generator
Candidates.#1....: sarierin -> sErcherd
```

6)Bcrypt:

```
Next dictionary / mask in queue selected. Bypassing current one.

Session..........: hashcat
Status...........: Bypass
Hash.Mode........: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target......: $2a$04$Pk/zPHTy2GHRw8a5H3rEmennu2raAIiyq/7u741TaVSi...EHoRBq
Time.Started.....: Wed Nov 08 12:38:28 2023 (2 secs)
Time.Estimated...: Next Big Bang (> 10 years)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?1?1?1?1?1?1?1 [8]
Guess.Charset....: -1 ?u?l?d, -2 Undefined, -3 Undefined, -4 Undefined
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     1687 H/s (12.27ms) @ Accel:1 Loops:1 Thr:16 Vec:1
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 2048/218340105584896 (0.00%)
Rejected.........: 0/2048 (0.00%)
Restore.Point....: 0/3521614606208 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:4-5 Iteration:14-15
Candidate.Engine.: Device Generator
Candidates.#1....: barierin -> bmsstane
```

5) ***Observation and inference:***

a) Observation table: We perofrm main experiment, consider the key to be: “pass@123”. The result are shown in table 1.

TABLE - 1:

	method	Hash function	key	Port number	Time to retrieve
1	Md5	dc06698f0e2e757515454559899adccc3	pass@123	0	12days 2hours
2	Sha1	ba97b1cf397425a852d1316d10787b1d97b5bc85	pass@123	1400	21days 22hours
3	Sha256	d97086919b6522e13ba9b46c04902c38372102218a4b3ef2f45ac2a80e9fd240	pass@123	100	47days 21hours
4	Sha512	419b199de5751e4ac1e1ed5e601007331344f75d5939829eb-fef895353e76ec95d14be93ecbc51e59f80d64e949379bbdfb52df7d93dd967676ab389ae173b84	pass@123	1700	230days 13hours
5	Django	pbkdf2_sha256$390000$E1AQCq34JRR1BKAxz0eFan$u3+pPwvXA9hHtL+RPt3nu2MBa-Pqi7t86K8tt32zbRNg=	pass@123	10000	>10 years
6	Bcrypt	$2a$04$Pk/zPHTy2GHRw8a5H3rEmennu2raAIiyq/7u741TaVSi.9uEHoRBq	pass@123	3200	>10 years

b)Inference:

- Asper observing the above experiment, the time difference with respect to different algorithm shows the strength of algorithm, thus we can observe:

  Md5<sha1<sha256<sha512<Django(PBKDF2-SHA256)/Bcrypt

  Thus, the Django framework with PBKDF2-SHA256 is strong as compared to the other chosen methods.

- The next observation is with the demo and md5 copy: Both the demo and md5 has hashed functions with md5, but we observe the difference in time that is needed for brute force. The main difference is that the length and structure of the password changes: For demo the text encrypted was pass this has not symbol and the length is comparatively less complex
  Demo.text:pass
  Md5.text:pass@123
  The md5 involves Symbols(@) and numbers(1,2,3) thus the complexity increases and the attack time differs,also in the command line in demo.text only”?u” means the brute force attack works on with only with lowercase variable and while checking with md5 the attack works with ”?u?l?d” means it involes with all the uppercase, lowercase and symbols.

- Another observation that we can observe is that both bcrypt and Django based framework are greater than 10 years and in the context of password security, both bcrypt and PBKDF2-SHA256 are considered secure options. However, bcrypt is often favored for password hashing due to its specific design for the task and its adaptive nature, which means it can be adjusted to remain secure even as computing power increases. Django, a popular web framework, has historically used PBKDF2 for password hashing, but it now also supports bcrypt and other password hashing algorithms for added security.

- Also the system specification also plays a major role in exploring the brute force attack time, using high performance processor system reduces time taken to attack. Our medium performance ranged system reading are given above in table-1. Thus, we improve the efficiency of penetration attack.

VII.FUTURE WORK:

Continuous research and adaptation are crucial in the ever changing field of cybersecurity. The area of password security may be advanced and made more robust in the face of new threats and shifting technological paradigms with the support of these future research topics. To create comprehensive password security solutions, multidisciplinary collaboration between specialists in cybersecurity, user experience, and cryptography is essential.

There are multiple directions for further study and investigation to improve our knowledge of password security and keep up with changing threats, even though this research article offers a thorough comparative review of various hashing techniques in the context of password security audits. Future research might focus on the following areas:

- Develop precise metrics to quantify the security of hashing schemes, such as their

resistance to attacks and computational requirements for password cracking.

- Investigate hybrid hashing schemes that combine multiple algorithms to improve security and examine their real-world performance.

- Post-Quantum Cryptography: Investigate post-quantum cryptographic methods to mitigate the risks posed by quantum computing.

- Password-less and User-Centric Authentication: Investigate password-less and user-centric authentication methods, as well as usability, security, and adoption considerations.

VIII. CONCLUSION:

The comparative analysis of different hashing schemes in the context of password security underscores the importance of selecting the right cryptographic technique to safeguard user credentials and sensitive data. As the digital landscape continues to evolve, and with the ever-present threat of cyberattacks, the choice of a hashing scheme is a critical decision that has profound implications for overall system security. Our examination of various hashing schemes like legacy algorithms like MD5 and SHA-1, has revealed significant disparities in security, performance, and adaptability to emerging threats.

This experiment focuses on the mostly used algorithm in today's world and this study is done only as a penetration test not as hacking attack, and this study can be improved by new tools, new performance processor and new algorithm to improve efficiency of protection or detailed studied on the cyber-attack with their solutions.

IX. REFERENCES

[1] Zheng, Y., Pieprzyk, J., & Seberry, J. (1992). HAVAL - A One-Way Hashing Algorithm with Variable Length of Output. *Theory and Application of Cryptographic Techniques*, 83–104. https://dblp.uni-trier.de/db/conf/asiacrypt/asiacrypt92.html#ZhengPS92

[2] Y. Jian, "An Improved Scheme of Single Sign-On Protocol Based on Dynamic Double Password," 2009 International Conference on Environmental Science and Information Application Technology, Wuhan, China, 2009, pp. 572-574, doi: 10.1109/ESIAT.2009.508.

[3] M. Dell'Amico, P. Michiardi and Y. Roudier, "Password Strength: An Empirical Analysis," 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 2010, pp. 1-9, doi: 10.1109/INFCOM.2010.5461951.

[4] T. W. van der Horst and K. E. Seamons, "Simple Authentication for the Web," 2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm 2007, Nice, France, 2007, pp. 473-482, doi: 10.1109/SECCOM.2007.4550369.

[5] Ying-Chieh Chen, Jing-Jang Hwang, Ronggong Song, G. Yee and L. Korba, "Online gaming cheating and security issue," International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II, Las Vegas, NV, USA, 2005, pp. 518-523 Vol. 1, doi: 10.1109/ITCC.2005.215.

[6] S. Suwisuthikasem and S. Tangsripairoj, "E-Tax Invoice System Using Web Services Technology: A Case Study of the Revenue Department of Thailand," 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Phuket, Thailand, 2008, pp. 937-942, doi: 10.1109/SNPD.2008.83

[7] Li, P., Shrivastava, A., Moore, J., & König, A. (2011). Hashing algorithms for large-scale learning. *Advances in neural information processing systems*, *24*.

[8] Sriramya, P., & Karthika, R. A. (2015). Providing password security by salted password hashing using bcrypt algorithm. *ARPN journal of engineering and applied sciences*, *10*(13), 5551-5556.

[9] Lakshmanan, T., & Madheswaran, M. (2012). A novel secure hash algorithm for public key digital signature schemes. Int. Arab J. Inf. Technol., 9(3), 262-267

[10] Deepakumara, J., Heys, H. M., & Venkatesan, R. (2001, May). FPGA implementation of MD5 hash algorithm. In *Canadian Conference on Electrical and Computer Engineering 2001. Conference Proceedings (Cat. No. 01TH8555)* (Vol. 2, pp. 919-924). IEEE.

[11] Ah Kioon, M. C., Wang, Z. S., & Deb Das, S. (2013). Security analysis of MD5 algorithm in password storage. *Applied Mechanics and Materials*, *347*, 2706-2711.

[12] Ntantogian, C., Malliaros, S., & Xenakis, C. (2019). Evaluation of password hashing schemes in open source web platforms. *Computers & Security*, *84*, 206-224.

[13] Aruna, S., Maheswari, M., & Saranya, A. (2020, December). Highly secured blockchain based electronic voting system using sha3 and merkle root. In *IOP Conference Series: Materials Science and Engineering* (Vol. 993, No. 1, p. 012103). IOP Publishing.

[14] Chand, K. R., Farida, S. K., Jyotshna, G. V., Karthik, M., & Venkat, S. M. A Secure Password Authentication Framework Based on Encrypted Negative Password.

[15] Li, P., Shrivastava, A., Moore, J. L., & König, A. C. (2011). Hashing Algorithms for Large-Scale Learning. *Neural Information Processing Systems*, *24*, 2672–2680. http://fodava.gatech.edu/files/reports/FODAVA-11-05.pdf

[16] Zhuang, X., Chang, C., Wang, Z., & Zhu, Y. (2014). A simple password authentication scheme based on

geometric hashing function. International Journal of Network Security, 16, 271–277.

[17] Hatzivasilis, G. (2017). Password-Hashing Status. Cryptography, 1(2), 10. doi:10.3390/cryptography1020010

[18] Mohamed Mostafa, A., & Ayied Almu-tairi, F. (2017). Integrated Pass-word-based Algorithms with Auditing Capability for Database Applications. Research Journal of Applied Sciences, Engineering and Technology, 14(5), 203–208. doi:10.19026/rjaset.14.4290

[19] Sumagita, Meiliana & Riadi, Imam. (2018). Analysis of Secure Hash Algo-rithm (SHA) 512 for Encryption Process on Web Based Application. 7. 373-381.

[20] Islam, S. (2021). Security Auditing Tools: A Comparative study. Interna-tional Journal of Computing Sciences Research, 5(1), 407–425. https://doi.org/10.25147/ijcsr.2017.001.1.49Y.