Journal Homepage: -www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI:10.21474/IJAR01/13299
DOI URL: http://dx.doi.org/10.21474/IJAR01/13299

RESEARCH ARTICLE

ANALYZING PASSWORD DECRYPTION TECHNIQUES USING DICTIONARY ATTACK

Pranav Kapoor, Pratham Agrawal and Aju D.
Vellore Institute of Technology, Vellore.

Manuscript Info

Manuscript History
Received: 15 June 2021
Final Accepted: 19 July 2021
Published: August 2021

Abstract

To guard ourselves against a word attack or a breach, it is always important to have an awareness of the unremarkably used sorts of attacks. The most common type of attack is password guessing. Hackers can guess the passwords locally or remotely using either manually or through an automated approach. One such attack is Dictionary Attack. A dictionary attack tries to make an authentication mechanism fail by sequentially entering each word in a dictionary as a password or trying to find the decryption key of an encrypted message or document. In this paper, an empirical research on how dictionary attack works are performed. In addition to that, different techniques and approaches to the existing dictionary attacks are implemented to make the system more robust. Furthermore, a comparison of methods is performed to find which approach is better to protect the system.

Introduction:-

Databases are of the utmost importance, thus their security and protection is also the foremost agenda for the businesses managing and maintaining them. Once a system gets attacked and vulnerable, it can prove to be disastrous for the whole organization. What threat actors do once they need access to an account depends on their supposed goal and the way abundant access that account will give, however, might embody stealing personal information, payment information, belongings, or conducting any attacks on a company. Dictionary attacks work because many computer users and businesses insist on using ordinary words as passwords. These attacks are usually unsuccessful against systems using multiple-word passwords and are also often unsuccessful against passwords made up of uppercase and lowercase letters and numbers in random combinations.

In systems with advanced countersign needs, the brute-force technique of attack, during which each probable combination of characters, numbers, and letters is tested up to a precise most length, will typically be effective. However, a brute-force attack will take a protracted time to supply results.Strong, randomized passwords cannot be predicted easily, and they are extremely unlikely to be included in the predetermined password library.

Because a dictionary attack's guess attempts are confined to a preselected list, it is essentially impossible to crack non-predictable passwords. Hence, in this paper, our main objective is to analyze the different approaches used by attackers to decrypt hashed passwords and compare the time taken for each approach. Thus, allowing users to create a strong password accordingly.

Corresponding Author:- Pranav Kapoor
Address:- Vellore Institute of Technology, Vellore.

Literature Review:-

MD5 is an algorithm that is utilized to confirm information respectability through the production of a 128-bit message digest [1] from information input which might be a message of any length that is professed to be as novel to that particular information as a unique finger impression is to the particular person. It is an augmentation of the MD4 message-digest algorithm. MD5 is somewhat slower than MD4, however is more "traditionalist" in a plan. MD5 represents Message-Digest algorithm 5, it is a broadly utilized cryptographic hash work that was designed by Ronald Rivest in 1991. The thought behind this algorithm is to take up arbitrary information advice regarding text or binary as info and produce fixed-size hash esteem as the yield. The information can be of any size or length; however, the yield hash esteem size is constantly fixed. All the hash values share the accompanying properties: Hash length which is the length of the hash esteem is dictated by the sort of the pre-owned algorithm, and its length doesn't rely upon the size of the document.

Storing a password in plain text or is undermined through simple encryption strategy [2] then there are potential outcomes of decrypting of password and taken. It very well might be bringing about counterfeit login and deficiency of privacy. MD5, SHA1 (Secure Hash Algorithm), and RIPEMD calculation are considered as broken calculations and we ought not to utilize our new application code from cryptography. To get information and password SHA256, SHA512, RipeMD, and WHIRLPOOL are cryptographic hash capacities that can be utilized. Hashing a password is a better technique than encryption of a password because hashing is a single direction work – plain text value from its hash implies the plain password that develops hash can't be recovered from its hash value. So, it is feasible to break a hash password by utilizing a predetermined hash value or utilizing a hash word reference. Hashing calculations are exceptionally deterministic as they produce the same hash value for the same input text. Crude hashes are likewise vulnerable to rainbow tables, a strategy for adjusting a requirement for pre-calculation of hashes, and the enormous capacity is important to keep a whole word reference of hashes.

SQL injection attack (SQLIA) [3] is among the most well-known security dangers to web-put together services that are sent to the cloud. By misusing web software weaknesses, SQL injection attackers can run discretionary malevolent code on track databases to gain or bargain touchy data. Even though web application firewalls (WAFs) are offered by most cloud service suppliers, inhabitants are hesitant to pay for them, since there are not many methodologies that can report precise SQLIA insights for their conveyed services. Traditional lWAFs center around hindering suspicious SQL requests. Few of them can precisely choose whether an attack is truly hurtful furthermore, rapidly answer how serious the attack is. To raise the occupants' familiarity with the earnestness of SQLIAs, in this paper, we present a novel traffic-based SQLIA recognition and weakness examination structure named DIAVA, which can proactively send admonitions to occupants instantly.

Most websites use passwords for validating client character and for permitting admittance to website assets [4] that may contain delicate data. An enormous number of individuals use word reference words for making passwords. These client passwords are exposed to single direction hash functions and are put away inside the information base as comparing hash values rather than plaintext. A potential programmer can utilize savage power, rainbow table, or word reference attacks to get the input password from the hash values and the most announced genuine hacks were finished by breaking password hashes utilizing word reference attack. This paper proposes a novel strategy for guaranteeing security for passwords against such word reference attacks. This strategy checks the strength of the client passwords utilizing a word reference which is put away as a character tree. This framework assists with making solid password hashes that are impervious to word reference attacks. This methodology consequently offers progressed and prevalent assurance for passwords from breaking endeavors.

Bcrypt calculation is a hashing capacity made from the Blowfish Algorithm by two PC security scientists [5], Niels Provos and David Mazieres. This hashing capacity has a few benefits, utilizing the original arbitrary salt (the salt is the request wherein it is added to the password to make it harder to brute force. Irregular salts likewise forestall query table creation. On this premise, the creators attempt to do a Brute Force probe plaintext that has been encoded by the Bcrypt Algorithm dependent on 3 characters, to be specific alphabetic characters, numeric characters, and mixed characters to see the security consequences of the Bcrypt Algorithm. From the consequences of tests led, the alphabetic character with an aggregate of 4 characters can be gotten back to the original plaintext inside 4 days while if the quantity of 5 characters can't be tracked down the original plaintext. At that point, the numeric characters with an aggregate of 7 characters can be found in the original plaintext within 10 hours. In the interim, for mixed characters with a sum of 7 characters, the original plaintext can't be found inside 5 days. The consequences of this examination demonstrate that the secure execution of the Bcrypt Algorithm is generally excellent in warding off Brute Force attacks for mixed characters while the numeric and alphabetic characters are not adequate.

An improvised methodology for plain text passwordencryption [6] in the worker's information base is presented. One of the significant parts of password insurance issue is to get it through encryption measures. In cryptanalysis, word reference attacks or animal power attacks are the most well-known methods of breaking passwords. Another methodology for ad-libbing the plan of password encryption is utilizing the way toward Jumbling-Salting (JS).

To enlarge the security angle concerning passwords, we are conceiving JS calculation which forestalls word reference and savage power attacks by expanding the length of code text in an impressive cut-off. In this calculation, the jumbling cycle chooses characters from the pre-characterized character set and adding them into the plain password utilizing numerical modulus (%) work; salting contains adding a random string into a jumbled password. Eventually, AES block is carried out which acquires a fixed-length password which is put away in the worker's information base.

The default hashing plans of famous CMS and web application frameworks are evaluated. To begin with, we plan the expense season of password speculating attacks, and next, we investigate the default hashing plans of mainstream CMS and web applications frameworks. We at that point apply our structure to play out a similar examination of the expense season of password speculating attacks between the different CMS and web application frameworks. At long last, taking into account that serious hash functions devour computational assets, we dissect hashing plans from an alternate point of view. That is, we investigate if it is possible and under what conditions to perform moderate rate denial of service attacks from simultaneous login endeavors. Through our investigation, we have determined a bunch of basic perceptions. Strikingly, the famous WordPress utilizes MD5 with a low number of hash iterations. Generally speaking, we accept that the security status of the hashing plans of CMS and web application frameworks calls for changes to the default settings from a select into a quit security policy. Greater security reviews and official library executions are additionally needed to speed up the appropriation of memory-hard functions both by policymakers and the business.

Cryptographic hash functions [8] are used to keep data by giving three fundamental wellbeing attributes: pre-picture opposition, second pre-picture obstruction, and crush the opposition. The foundation of cryptographic security lies in the arrangement of pre-picture obstruction, which makes it hard and time-devouring for an attacker to track down an original message given the particular hash value. This security is given by the idea of one-way functions, which is a critical part of SHA. SHA-256 is quicker on 32-bit processors. SHA-512 is quicker on 64-bit processors. SHA-512 has 25% a greater number of rounds than SHA-256. SHA-256 performs 64 rounds of its compression work throughout 512 bits all at once.

In this period of innovation, all the web-based work [9] is being performed byPCs. From talking with companions on interpersonal interaction websites to making on the web installments through Net Banking, everything is being done online through PCs. Since these offices are proficient and make our work simple, we use them in without a doubt. This way to utilize these online services we are storing all our own and touchy data in the databases of these websites and applications, which undoubtedly make this data inclined to different security dangers. So, insurance of this significant client data is one of the significant needs, to evade any abuse of data.

Another significant method of securing this data is by scrambling the data being saved in the databases of these websites. In this paper, we will examine the different database encryption plans proposed by various creators, and study the benefits and negative marks of these plans.

An iterative salted hash encryption instrument named hostile to consistent impact salted hash encryption (ACCSHE) [10] that utilizes cryptographically secure pseudo arbitrary number generator (CSPRNG) and hashed with password information by got hashing calculation (SHA-256). The result shows the adaptability of changing the outcomes by controlling a few key highlights, for example, salt utilized, the scope of the hashing cycle iterations, or the base number of iterations without influencing the framework's convenience.

This exploration built-up head and tail (HT) method that additional intricacy on carrying out MD5 and SHA-1 calculation for hashing passwords utilizing discontinuity, what's more, connection measures. The Head and Tail strategy utilize numerous hashing on secret phrase plain content and spot control on joined secret word condensation to deliver a 512- bit hash. Recurrence (Monobit) Test gives a normally registered p-estimation of 0.72 when HT is applied to MD5 and 0.73 to SHA-1 while Frequency Test inside the square came about to a normal p-estimation of 0.854 and 0.806 for MD5 with HT and SHA-1 with HT individually. Then again, in the run test, a normal p-estimation of 0.40 is accomplished for MD5 with HT and a normal value of 0.54 for SHA-1 with HT. The HT method has significantly improved the haphazardness of the secret phrase digest. It is one of the fundamental prerequisites to secure data. It is the cycle of decidedly checking a client's personality, gadget, or other substance in a PC framework. It is essential to permitting admittance to assets in the programming framework.

Description of Modules/Programs

Storing user password keys in plain text naturally ends up in a rapid compromise of all passwords if the password file is compromised.

To reduce this danger, Windows applies a cryptanalytic hash perform, that transforms every identification into a hash and stores this hash. Hash Suite, is similar to other hashed-based converters and does not decrypt a hash directly which is impossible in reality.

It follows the same procedure used by authentication: it generates different candidate passwords (keys), hashes them, and compares the computed hashes with the stored hashes. This approach works because users generally select passwords that are easy to remember, and as a side-effect, these passwords are typically easy to crack.

Another probable reason why this method is very effective is that Windows uses password hash functions that are very fast to compute, especially in an attack (for each given candidate password). The proposed system also supports most of the commonly used hashing algorithms to store passwords. No dependencies are needed, just python is required.

Algorithms Supported

SHA1 algorithm

There are two achievable line formats: the primary one contains a saltless password whereas the second contains a salt-cured password at the side of the salt.

The passwords are hashed using SHA-1. When salt is utilized, it's just concatenated at the facet of the words as follows: salt || watchword. The attack just reads the dictionary line by line and computes utterly totally different come-at-able hashed passwords for the word contained in each line.

These six generated hashes are compared to every one of the passwords contained within the password.txt file for a match. If there's a match, we get the desired password and result. Else, we tend to merely keep reading the dictionary line by line.

MD5 algorithm

The MD5 hashing algorithmic rule is a unidirectional science perform that accepts a message of any length as input and returns as output a fixed-length digest price to be used for authenticating the initial message.

The MD5 message-digest hashing algorithmic rule processes knowledge in 512-bit blocks weakened into sixteen words composed of thirty-two bits. The output from MD5 could be a 128-bit message digest price. Computation of the MD5 digest price is performed in separate stages that method every 512-bit block of knowledge at the side of the worth computed within the preceding stage. the primary stage begins with the message digest values initialized mistreatment consecutive hex numerical values. every stage includes four message-digest passes that manipulate values within the current knowledge block and values processed from the previous block.The final hash computed from the last block becomes the MD5 digest for that block.

SHA-2 Algorithm

SHA-2 is a family of hashing rules to exchange the SHA-1 algorithmic rule. SHA-2 choices a future level of security than its precursor.
The SHA-2 family consists of six hash functions with digests (hash values) that square measure 224, 256, 384, or 512 bits
SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256.

SHA3 Algorithm

SHA-3 (Secure Hash algorithmic rule 3) is a set of science hash functions outlined in FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions.

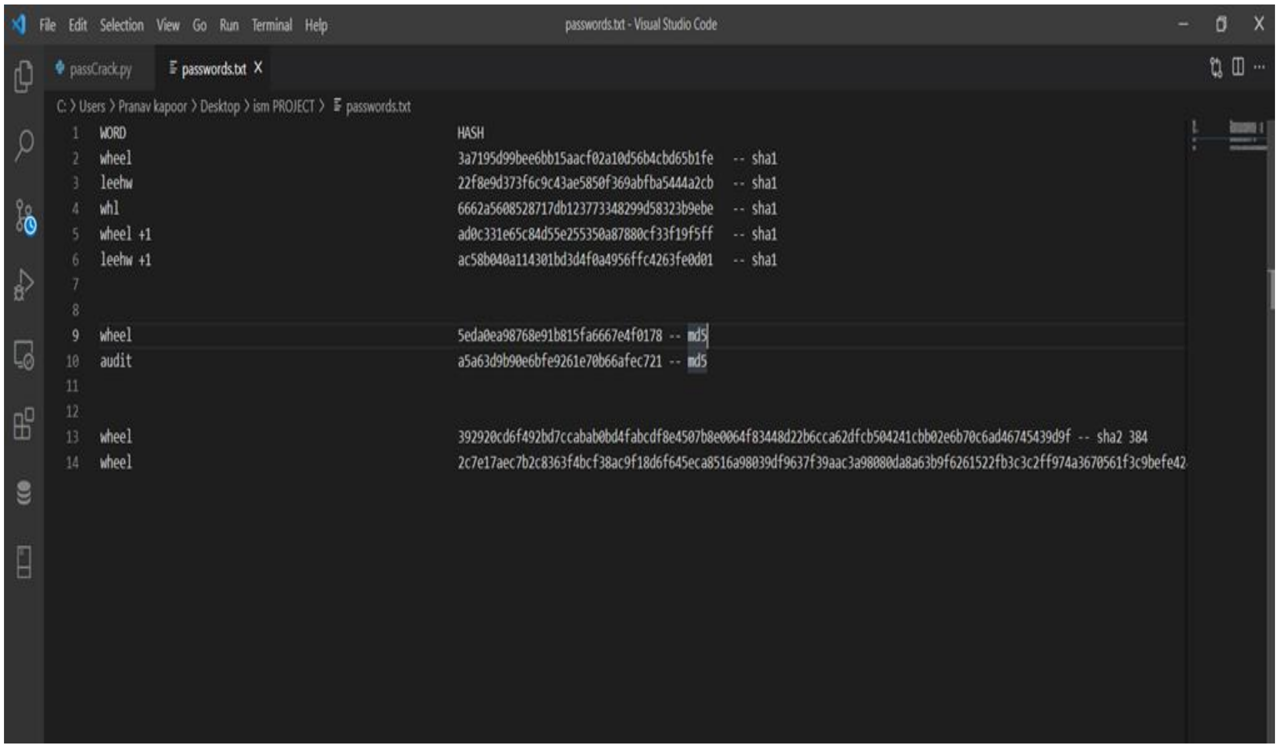
The SHA-3 group includes 6 hash functions with digests (hash values) that square measure 128, 224, 256, 384, or 512 bits: SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256.

Proposed approach

A simple dictionary attack is limited and confined to exact matches, but is still surprisingly successful, as users generally tend to choose simple and predictable passwords. Used words include most common and banal names and surnames from several languages such as favorite food, color, or names of family members who may be vulnerable to attack.
These specific expressions and words are appended to a large wordlist containing the most common and over-used passwords that users generally use, and are obtained in several well-known leaks, forming a 14,457,264-word dictionary for a dictionary attack.

A list of passwords is taken for the attack in a text file in which each line contains a password followed by its hash using the chosen algorithm.

These passwords are tested using all the algorithms used: MD5, SHA1, SHA2, SHA3. A dictionary is used to implement the dictionary attack where each word in the dictionary is hashed and tested against the password used. To make a password more robust, a salt can be added. A salt is simply concatenated together with the passwords which are used by user as follows: salt || password.

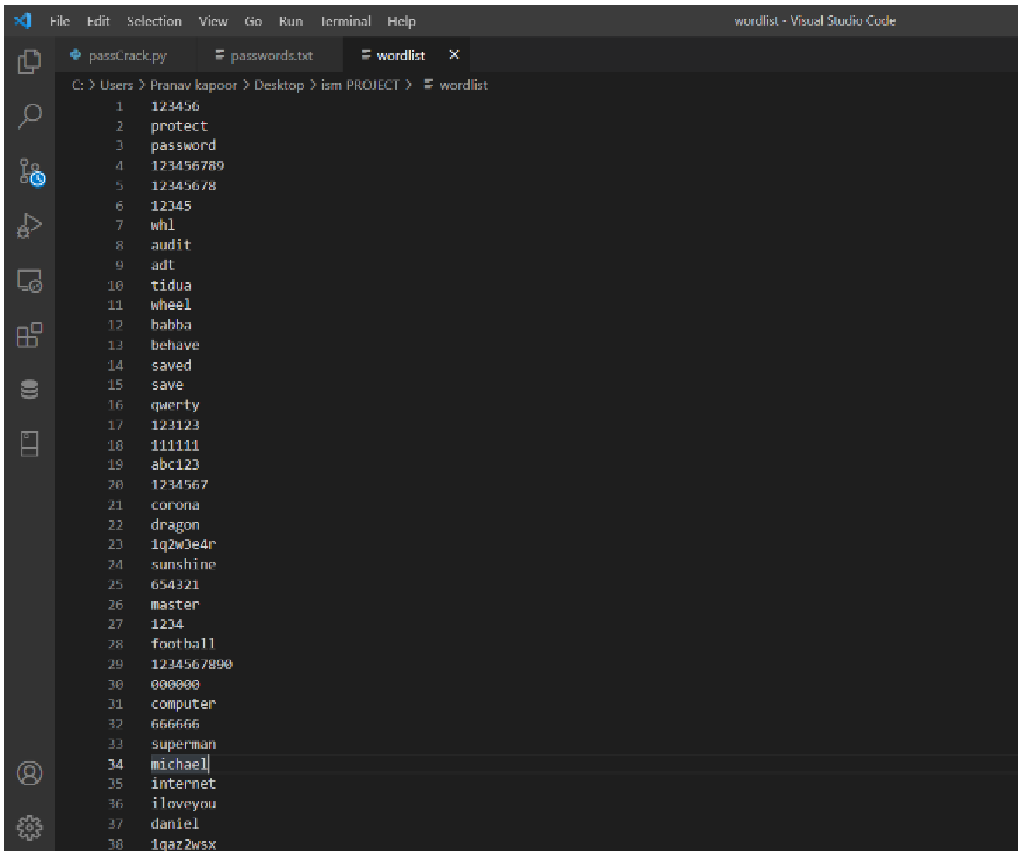

File Edit Selection View Go Run Terminal Help
passwords.txt - Visual Studio Code
passCrack.py
passwords.txt
C: > Users > Pranav kapoor > Desktop > ism PROJECT > passwords.txt

```
WORD                                    HASH
wheel                                   3a7195d99bee6bb15aacf02a10d56b4cbd65b1fe   -- sha1
leehw                                   22f8e9d373f6c9c43ae5850f369abfba5444a2cb   -- sha1
whl                                     6662a5608528717db123773348299d58323b9ebe   -- sha1
wheel +1                                ad0c331e65c84d55e255350a87880cf33f19f5ff   -- sha1
leehw +1                                ac58b040a114301bd3d4f0a4956ffc4263fe0d01   -- sha1


wheel                                   5eda0ea98768e91b815fa6667e4f0178 -- md5
audit                                   a5a63d9b90e6bfe9261e70b66afec721 -- md5


wheel                                   392920cd6f492bd7ccabab0bd4fabcdf8e4507b8e0064f83448d22b6cca62dfcb504241cbb02e6b70c6ad46745439d9f -- sha2 384
wheel                                   2c7e17aec7b2c8363f4bcf38ac9f18d6f645eca8516a98039df9637f39aac3a98080da8a63b9f6261522fb3c3c2ff974a3670561f3c9befe42
```

Figure 1:- A list of passwords taken along with the hashes using different algorithms.


File Edit Selection View Go Run Terminal Help
wordlist - Visual Studio Code
passCrack.py
passwords.txt
wordlist
C: > Users > Pranav kapoor > Desktop > ism PROJECT > wordlist

```
123456
protect
password
123456789
12345678
12345
whl
audit
adt
tidua
wheel
babba
behave
saved
save
qwerty
123123
111111
abc123
1234567
corona
dragon
1q2w3e4r
sunshine
654321
master
1234
football
1234567890
000000
computer
666666
superman
michael
internet
iloveyou
daniel
1qaz2wsx
```

Figure 2:-A list of words taken in a dictionary for testing.

The attack goes through the dictionary line by line and computes 6 different hashed passwords for the word contained in each lineusing 6 approaches. The 6 approaches used to convert each word into its subsequent hash from the dictionary are word, reversed word, a word without vowels, salted word, salted reversed word,and salted word without vowels.

(word): Here each word in a dictionary is hashed with the respective algorithm and compared with the hashed password. If the hash matches, then we get the password used.

(drow) (Reversed word): In this approach, each word in a dictionary is reversed and hashed with the respective algorithm and is compared with the hashed password. If the hash matches, then we get the password used.

(wrd) (Word without vowels): In thisapproach, each word in a dictionary is takenwithout any vowels and is hashed with therespective algorithm and compared with thehashed password. If the hash matches, thenwe get the password used.
(salt|word) (Salted word): In this approach, each word in a dictionary is reversed and hashed with the respective algorithm and is compared with the hashed password. If the hash matches, then we get the password used.

(salt||drow) (Salted reversed word): Here each word in a dictionary is reversed and hashed with the respective algorithm and is compared with the hashed password. If the hash matches, then we get the password used.

(salt||wrd) (A Salted word without vowels)- Here each word in a dictionary is reversed and hashed with the respective algorithm and is compared with the hashed password. If the hash matches, then we get the password used.

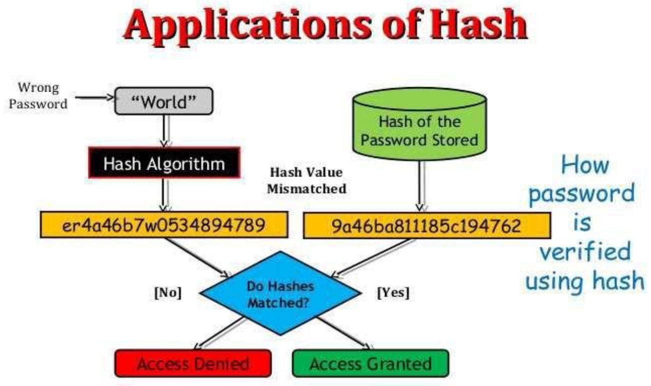
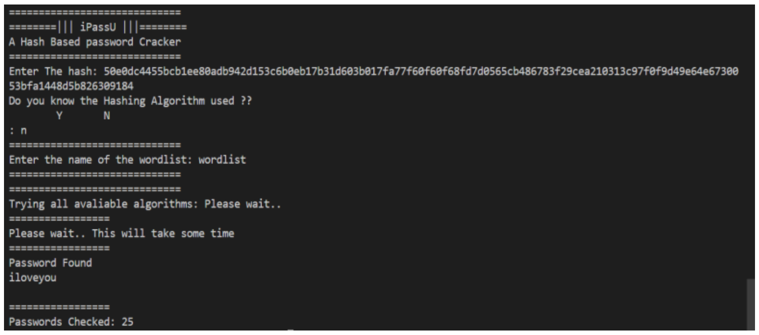


Figure 3:-Flow diagram of hashing and password matching process.

Results:-

Case 1: All algorithms are tested Together



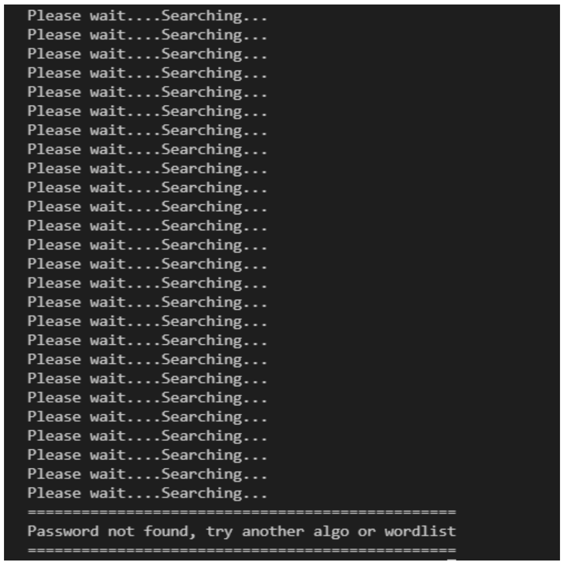
```
=============================
========||| iPassU |||========
A Hash Based password Cracker
=============================
Enter The hash: 50e0dc4455bcb1ee80adb942d153c6b0eb17b31d603b017fa77f60f60f68fd7d0565cb486783f29cea210313c97f0f9d49e64e67300
53bfa1448d5b826309184
Do you know the Hashing Algorithm used ??
        Y       N
: n
=============================
Enter the name of the wordlist: wordlist
=============================
=============================
Trying all avaliable algorithms: Please wait..
=================
Please wait.. This will take some time
=================
Password Found
iloveyou

=================
Passwords Checked: 25
```

Figure 4:-Passwords checked = 25 bytesting using all the algorithms.

Case 2: A specific algorithm is chosen with which the password is hashed

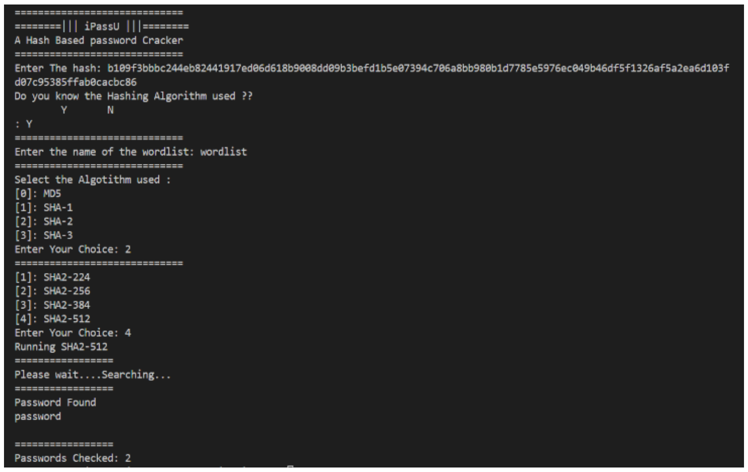
SHA 1 ALGORITHM



```
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 Please wait....Searching...
 ==============================================
 Password not found, try another algo or wordlist
 ==============================================
```

Figure 5:- Password Not found with a salted password which is reversed and hashed using the SHA1 algorithm.

SHA 2 ALGORITHM

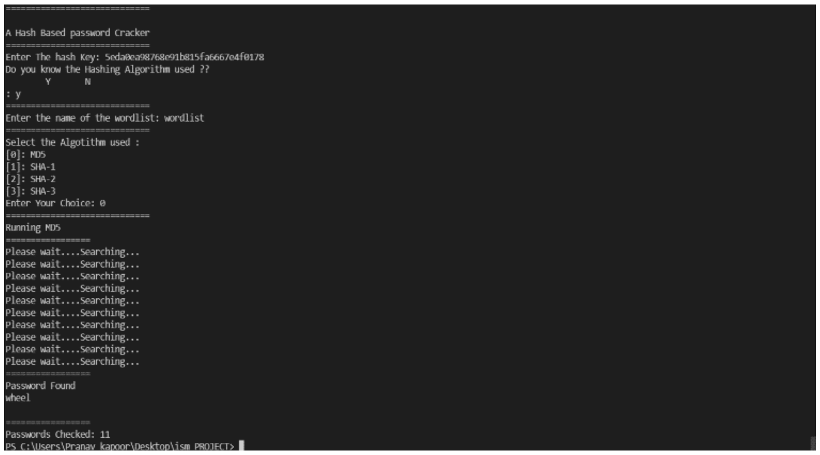


```
==============================
========||| iPassU |||========
A Hash Based password Cracker
==============================
Enter The hash: b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e07394c706a8bb980b1d7785e5976ec049b46df5f1326af5a2ea6d103f
d07c95385ffab0cacbc86
Do you know the Hashing Algorithm used ??
        Y       N
: Y
==============================
Enter the name of the wordlist: wordlist
==============================
Select the Algotithm used :
[0]: MD5
[1]: SHA-1
[2]: SHA-2
[3]: SHA-3
Enter Your Choice: 2
==============================
[1]: SHA2-224
[2]: SHA2-256
[3]: SHA2-384
[4]: SHA2-512
Enter Your Choice: 4
Running SHA2-512
=================
Please wait....Searching...
=================
Password Found
password

=================
Passwords Checked: 2
```

Figure 6:-Passwords checked = 2 using a simple word and hashed using the SHA2 Algorithm.

MD5 ALGORITHM



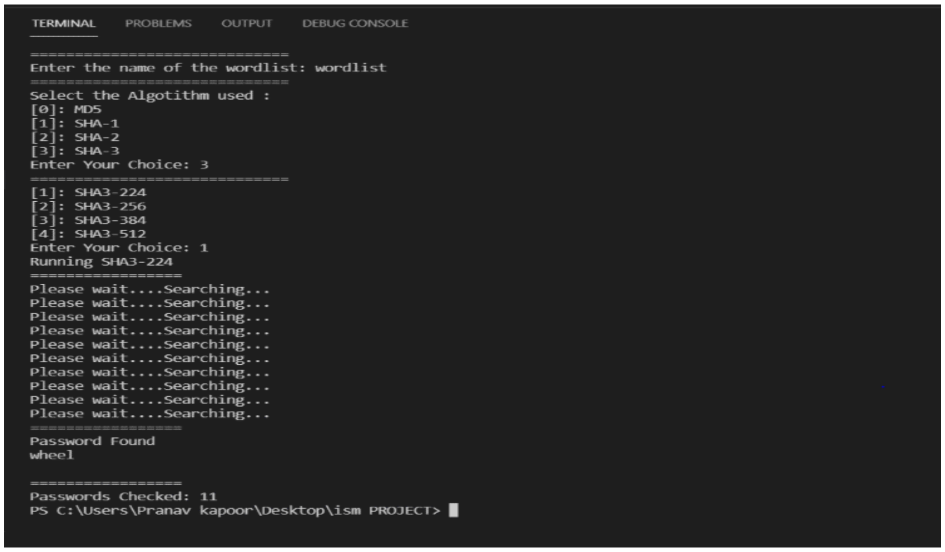
```
==============================

A Hash Based password Cracker
==============================
Enter The hash Key: 5eda0ea98768e91b815fa6667e4f0178
Do you know the Hashing Algorithm used ??
        Y       N
: y
==============================
Enter the name of the wordlist: wordlist
==============================
Select the Algotithm used :
[0]: MD5
[1]: SHA-1
[2]: SHA-2
[3]: SHA-3
Enter Your Choice: 0
==============================
Running MD5
=================
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
=================
Password Found
wheel

=================
Passwords Checked: 11
PS C:\Users\Pranav kapoor\Desktop\ism PROJECT> █
```

Figure 7:-Passwords checked = 11 using a reversed word in MD5 Algorithm.

SHA 3 ALGORITHM



```
==============================
Enter the name of the wordlist: wordlist
==============================
Select the Algotithm used :
[0]: MD5
[1]: SHA-1
[2]: SHA-2
[3]: SHA-3
Enter Your Choice: 3
==============================
[1]: SHA3-224
[2]: SHA3-256
[3]: SHA3-384
[4]: SHA3-512
Enter Your Choice: 1
Running SHA3-224
=================
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
Please wait....Searching...
=================
Password Found
wheel

=================
Passwords Checked: 11
PS C:\Users\Pranav kapoor\Desktop\ism PROJECT> █
```

Figure 8:- Passwords checked=11 using words without vowels approach in SHA3 algorithm.

Conclusion:-

After running all the approaches and algorithms, we find that dictionary attack is successful in most of the cases where common words are used whereas in cases such as passwords with a salt, the attack takes a lot of time and has to go through a huge number of passwords present in the list.

Hence the key to protecting passwords from a dictionary attack is to apply salting consisting of an amalgam of letters, numbers, and symbols to protect the password from getting decrypted by attackers. In a large database consisting of lakhs of passwords, a salted password will take a lot of time to be detected, and thus can help the organization to protect themselves in this time.

References:-

1. Vulnerability of data security using MD5 function in php database design. Jacob Neyole.
2. Providing Password Security By Salted Password Hashing Using Bcrypt Algorithm P. Sriramya and R. A. Karthika
3. DIAVA: A Traffic-Based Framework for Detection of SQL Injection Attacks and Vulnerability Analysis of Leaked Data Haifeng Gu, Jianning Zhang, Tian Liu, Ming Hu, Junlong Zhou , Tongquan Wei and Mingsong Chen
4. Securing Passwords from Dictionary Attack with Character-Tree Jacob Jose, Tibin T. Tomy, Vibin Karunakaran Anjali Krishna V, Anoop Varkey, Nisha C.A.
5. Analysis Performance BCRYPT Algorithm to Improve Password Security from Brute Force To cite this article: Toras Pangidoan Batubara
6. Jumbling- Salting: An Improvised Approach for Password Encryption Prathamesh P. Churi Vaishali Ghate, Kranti Ghag
7. Evaluation of password hashing schemes in open source web platforms Christoforos Ntantogian , Stefanos Malliaros , Christos Xenakis
8. A Study of SHA Algorithm in Cryptography Soe Moe Myint1, Moe Moe Myint2, Aye Aye Cho
9. Database Security Using Encryption Prabhsimran Singh Dr. Kuljit Kaur
10. Anti-continuous collisions user-based unpredictable iterative password salted hash encryption Munqath H. Alattar.