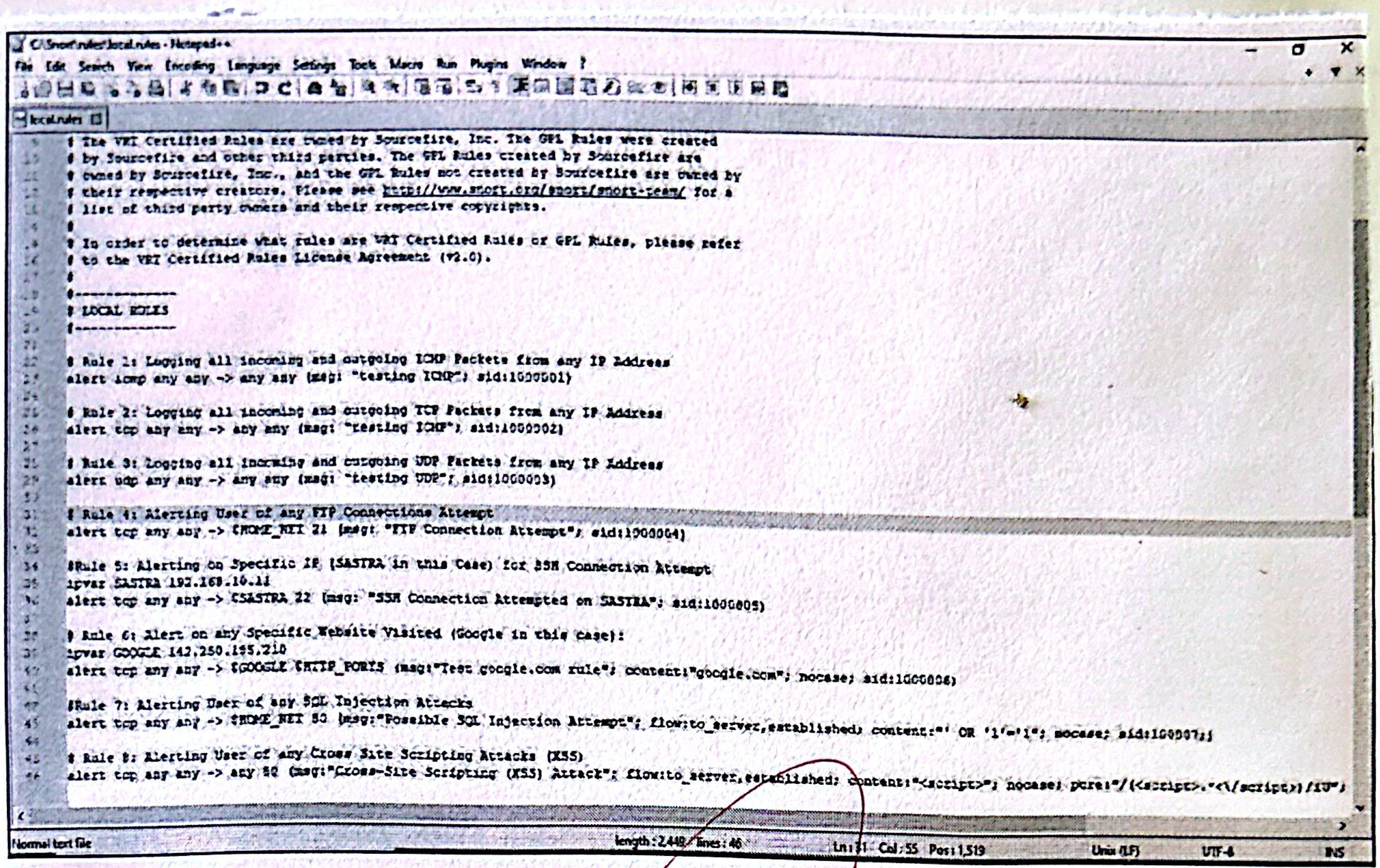


Fig 4a.a - Customized rules

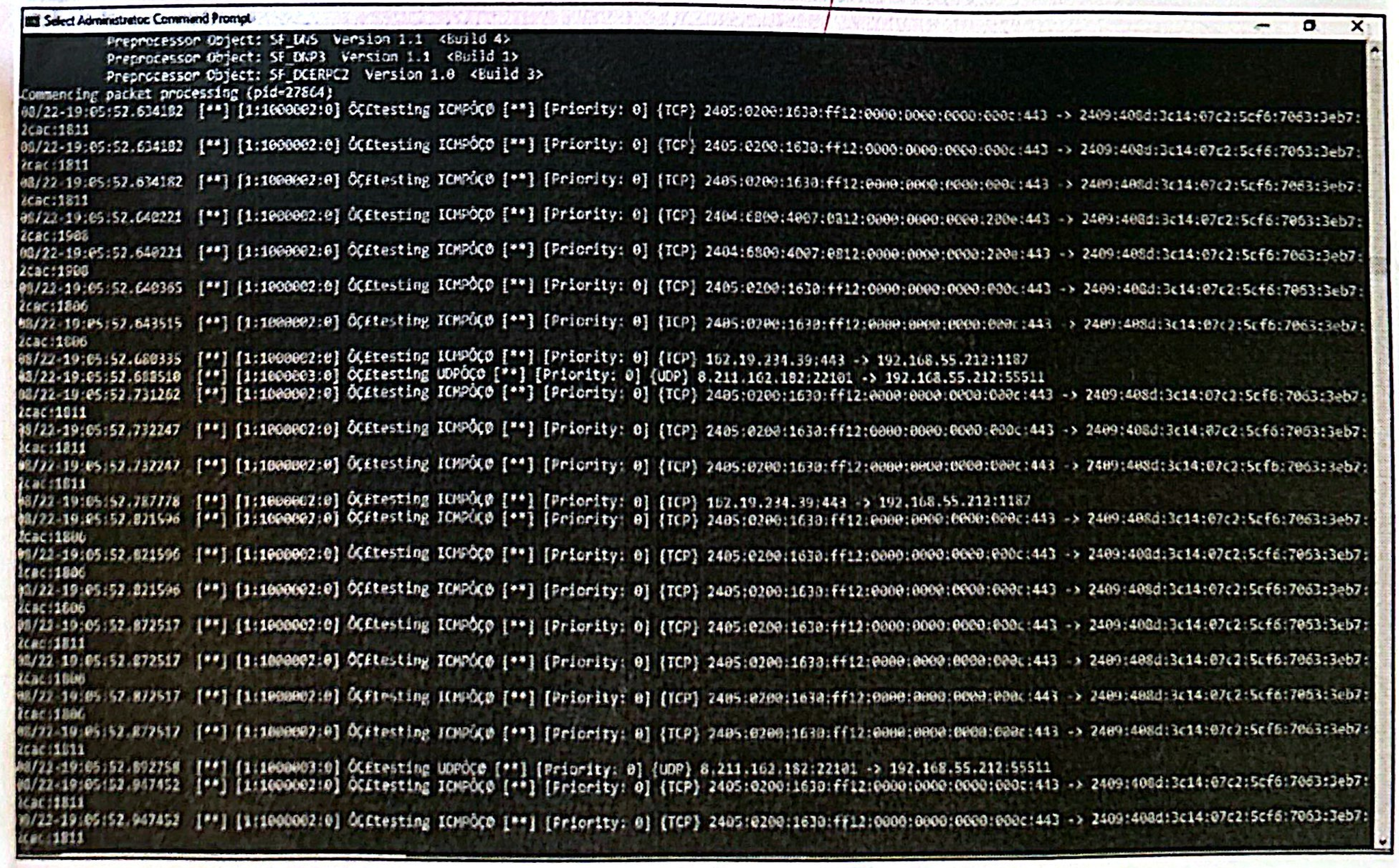


Fig 4a.b - Alerts raised for the rules in command prompt

Date : 21-8-23

E4- Making Customized Rules in Snort

Aim: To apply customized rules and detecting the appropriate alerts through the snort Intrusion Detection Tool

Procedure:

I) Create customized rules in local.rules in "rules folder"

II) Configure snort for the following Ruleactions:

1) ALERT: Generates an alert when suspicious packet is detected.
II) BLOCK: Blocks the suspicious & all subsequent packets coming from that network flow of any protocol specified.
III) DROP: Drops the packet as soon as alert is generated.
IV) LOG: LOGS the details of packet as soon as alert is generated.
V) PASS: Ignores the suspicious packet & marks it as "Passed"

III) Once customized rules according to the above rule patterns are written, save the file and execute the command

Snort -i <Interface Index> -C<Conf file path> -A Console.

Result: Thus, we analyzed & implemented custom rules for Intrusion Detection Systems using Snort in Windows 11 Operating System.

Teacher's Signature:

Date: 21-8-23

Subtask - Signature Based Rule Permission

Page No.: 11
Expt. No.: 4b

Aim: To apply Signature Based Rule Permission (Allow & disallow coding in Particular app)

Procedure: We are doing coding for "File Permission" for extracting & displaying file permissions and access control for given file(by user)

language: Python

module used: pywin32, os, win32security (① pywin32, ② os, win32security)

1) First, we import the modules, os and win32security mainly for file system information and windows security settings

2) Defining SDDL to be translated as a suitable function
Its a way to represent security systems in windows
fn: ~~Sddl~~ translate_sddl, takes SDDL string as input & translates to human readable format, by examining access control list (ACL).

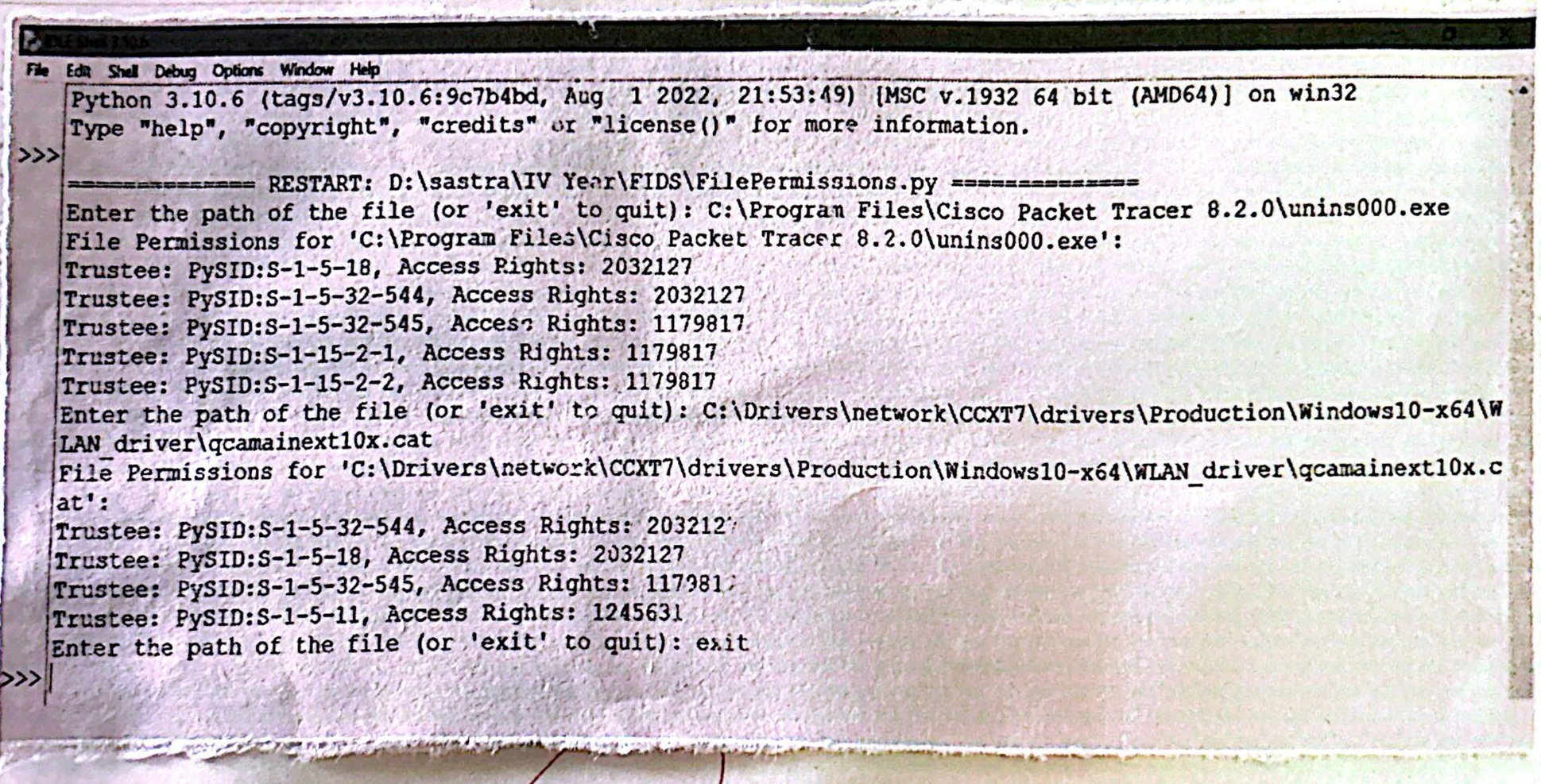
3) Defining main fn for main logic, to get file path as well as exit command input from user.

4) Retrieving file permission: when user provides file path, code attempts to retrieve & disp permission for that file using "win32security.GetFileSecurity" funtion including discretionary access control list which houses the permission.

5) Displaying file permission: Script extracts info from DACL including trustee and access rights, displays it to us as output

Teacher's Signature:

OUTPUT SNIP :-


File Edit Shell Debug Options Window Help
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug 1 2022, 21:53:49) [MSC v.1932 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\sastra\IV Year\FIDS\FilePermissions.py ==========
Enter the path of the file (or 'exit' to quit): C:\Program Files\Cisco Packet Tracer 8.2.0\unins000.exe
File Permissions for 'C:\Program Files\Cisco Packet Tracer 8.2.0\unins000.exe':
Trustee: PySID:S-1-5-18, Access Rights: 2032127
Trustee: PySID:S-1-5-32-544, Access Rights: 2032127
Trustee: PySID:S-1-5-32-545, Access Rights: 1179817
Trustee: PySID:S-1-15-2-1, Access Rights: 1179817
Trustee: PySID:S-1-15-2-2, Access Rights: 1179817
Enter the path of the file (or 'exit' to quit): C:\Drivers\network\CCXT7\drivers\Production\Windows10-x64\WLAN_driver\qcamainext10x.cat
File Permissions for 'C:\Drivers\network\CCXT7\drivers\Production\Windows10-x64\WLAN_driver\qcamainext10x.cat':
Trustee: PySID:S-1-5-32-544, Access Rights: 203212
Trustee: PySID:S-1-5-18, Access Rights: 2032127
Trustee: PySID:S-1-5-32-545, Access Rights: 117981
Trustee: PySID:S-1-5-11, Access Rights: 1245631
Enter the path of the file (or 'exit' to quit): exit
>>>

Date :

Page No. : 12
Expt. No. : 4b

6) If any errors are detected in the process, such as lack of permission, file inexistant and soon, error msg will be shown in form of access right code.

7) Enter file path or exit after getting results from the program.

Program:

```
import os
import win32 security
def translate_sddl(sddl):
    try:
        Sd = win32 security.ACL()
        win32security.SetSecurityDescriptorDacl(sd, True, win32 security.ACL())
        win 32 security.SetSecurityDescriptorSddlDacl(Sd, sddl, False)

        dacl=sd.GetSecurityDescriptorDacl()
        permission: []
        for i in range(dacl.GetAce(i)
        trustee, access_rights_=ace
                                                ace=dacl.Get.Ace(i)

        permissions.append((trustee, access_rights))
        rebum permissions
    except Exception as e:
    rebum str(e)


def main():
    while True:
    file_Path = input("Enter path of file (or 'exit' to qu:t):")
        if file_path.lower() == 'exit':
        break

    try:
```

Teacher's Signature:

Date :

Page No. : 13
Expt. No. : 4b

```
Sd = win32Security.GetFileSecurity(file_path, win32security.OWNER_SECURITY
- INFORMATION | win32security.DACL_SECURITY_INFORMATION)
dacl = sd.GetSecurityDescriptorDacl()

permissions []
for i in range(dacl.GetAceCount()):
ace = dacl.GetACE(i)
  trustee = ace[2]
  access_rights = ace[1]
  permissions.append((trustee, access_rights))
print(f"File Permission for '{file_Path}':")
  for trustees, access_rights in permissions:
      print(f"Trustee: {trustee}, Access Rights: {access_rights}")
      except Exception as e:
          print(f"Error: {str(e)}")

if __name__ == "__main__":
main()
```

Result: Finally, Successfully applied Signature based rule permission

Teacher's Signature: