

Put Student Name(s) ↓		Put Student IDs ↓	Due Date	Grade Weight
Ishan Aakash Patel		146151238	As Posted	12%
Name	Project1: Build Your own Forensic Workstation			
Main Goal	Set up your functional forensic workstation to conduct forensics investigations using variety of popular tools			
Instructions	<ul style="list-style-type: none"> <li>• It is an Individual assignment. Put your name + Student ID in the empty spaces above.</li> <li>• Submit via the BB relevant link ONLY. NO submission via email please. Be sure to submit the final version file ONLY.</li> <li>• Show me genuine signs of your work is done on your machine. This includes: <ul style="list-style-type: none"> <li>◦ Screenshots that show your desktop background with Date/Time</li> <li>◦ Show me a pop-up bx that shows "your name + IP."</li> <li>◦ Show your logged in account, if applicable</li> <li>◦ Optional: Show your photo.</li> </ul> </li> <li>• Use this same template to include your work in the specified fields below. Submit in PDF.</li> <li>• Submit your report name with the name: CYT215-Project1-Student Name &amp; ID</li> </ul>			
Students Work required for this activity	<ol style="list-style-type: none"> <li>1. You will follow instructions to setup your own forensics workstation on your machine.</li> <li>2. You will check that your forensics workstation is functioning.</li> <li>3. You will use your workstation for memory &amp; malware analysis.</li> <li>4. You will Prepare Your Target System (your own machine). You will Build Your basic Lab.</li> </ol>			
How to start	<ul style="list-style-type: none"> <li>• Read thoroughly &amp; follow the instructions mentioned in this link. The instructions will guide you to a step-by-step of how you complete your work successfully <a href="https://bluecapesecurity.com/build-your-forensic-workstation/">https://bluecapesecurity.com/build-your-forensic-workstation/</a></li> <li>• Take an image of your machine memory. Use this link for guidance: <a href="https://dfirmadness.com/case-001-memory-analysis/">https://dfirmadness.com/case-001-memory-analysis/</a> -- UPDATE INSTEAD ANALYSIS the MEMORY DUMP provided using Volatility.</li> <li>• You can use any installed tool or focus on common tools for memory analysis e.g.: Volatility; Cyber Triage; Rekall; Redline;</li> </ul>			
Important	<ul style="list-style-type: none"> <li>• Your target system should be your own machine</li> </ul>			
Students Reports	<ol style="list-style-type: none"> <li>1. Take screenshots of all your works steps</li> <li>2. Show you have practiced 3 relevant tools to investigate memory.</li> <li>3. For Memory analysis, anaylsis the Memory dump provided and solve the challenges - <a href="https://github.com/stuxnet999/MemLabs/tree/master/Lab%203">https://github.com/stuxnet999/MemLabs/tree/master/Lab%203</a> – Follow this guide: <a href="https://oviche.github.io/2023/10/MemLabs3/">https://oviche.github.io/2023/10/MemLabs3/</a></li> <li>4. Write a detailed report of personal learning experience (free writing).</li> </ol>			
Grading Rubrics	<ul style="list-style-type: none"> <li>• 5 Marks for completing the setup of your forensics workstation successfully.</li> <li>• 5 Marks for using your workstation for memory analysis. You can use Volatility to solve the challenge - <a href="https://github.com/stuxnet999/MemLabs/tree/master/Lab%203">https://github.com/stuxnet999/MemLabs/tree/master/Lab%203</a></li> <li>• 2 Marks for your detailed personal learning experience (free writing)</li> </ul>			
	<ul style="list-style-type: none"> <li>• If you do NOT use this template or delete any part of it or use any other template, you will be degraded.</li> </ul>			

Grading  
Alerts

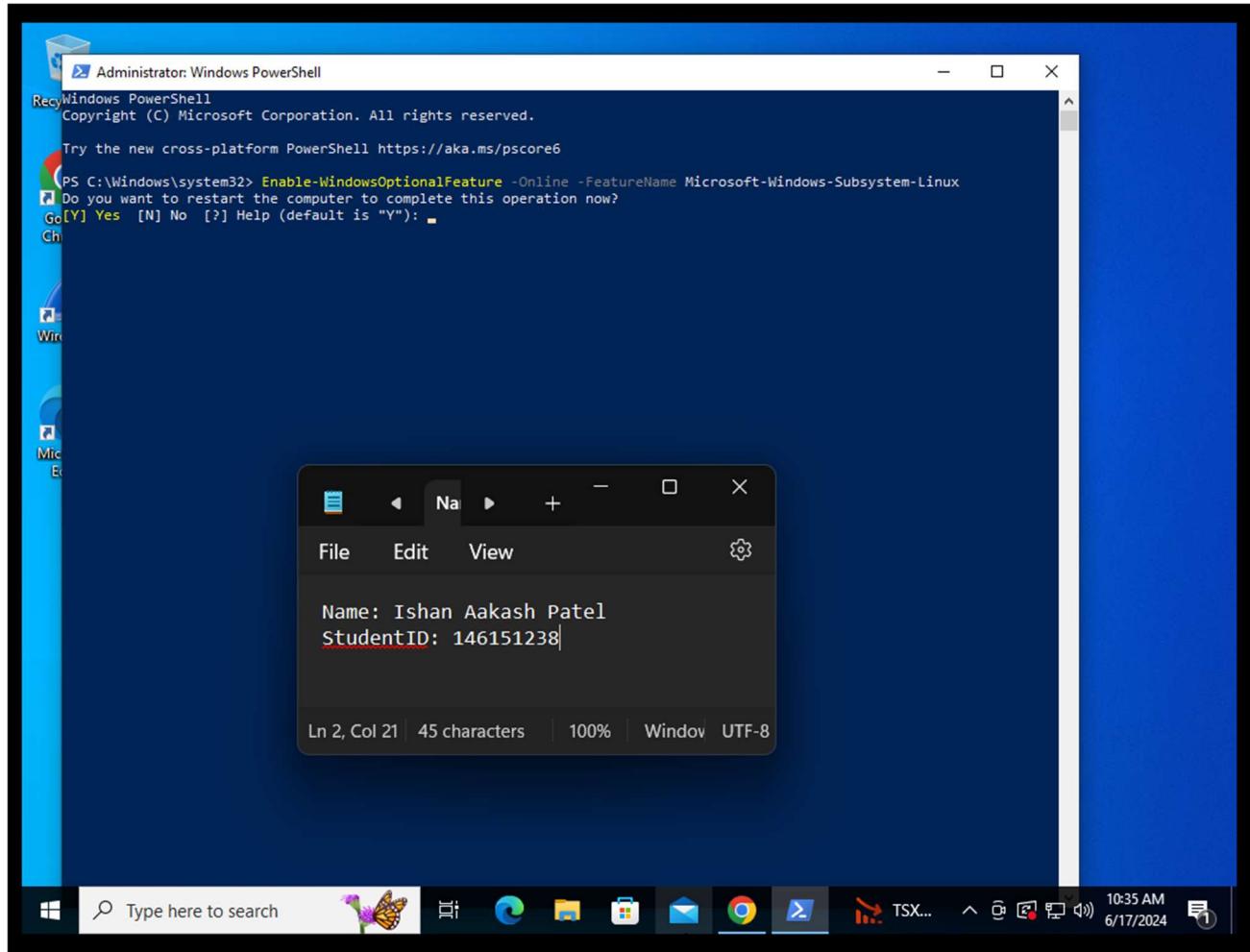
- If you do NOT follow the fie naming convention, you will be degraded.
- If you do NOT submit your file in PDF; you will be degraded.
- If you do NOT show your account real name (when applicable); you will be degraded.
- If you do NOT show your machine desktop background (with date & time) and IP, you will be degraded.
- If you do NOT write (in your own words) your learning experience for the activity practices, you will be degraded.

## Part 1 – Building my own Workstation

I had a Windows 10 already installed in my VMware Workstation Pro, So I had decided to build the forensics lab inside this vm.

First, installing WSL.

### 1) WSL



The screenshot shows a Windows 10 desktop environment. A PowerShell window titled "Administrator: Windows PowerShell" is open, displaying command-line output related to app deployment. A Notepad window is also visible, containing student information. The taskbar at the bottom includes icons for File Explorer, Edge, Mail, and other system tools.

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Deployment operation progress: C:\Users\ishan\Downloads\Ubuntu2204-221101\Ubuntu_2204.1.7.0_x64.appx
    Processing
    [oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo]

Add-AppxPackage : Cannot find path 'C:\Users\ishan\Downloads\Ubuntu2204-221101\Ubuntu_2004.4.2.0_x64.appx' because it
does not exist.
At line:1 char:1
+ Add-AppxPackage .\Ubuntu_2004.4.2.0_x64.appx
+ CategoryInfo          : ObjectNotFound: (C:\Users\ishan\....4.2.0_x64.appx:String) [Add-AppxPackage], ItemNotFou
ndException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.Windows.Appx.PackageManager.Commands.AddAppxPackageCommand

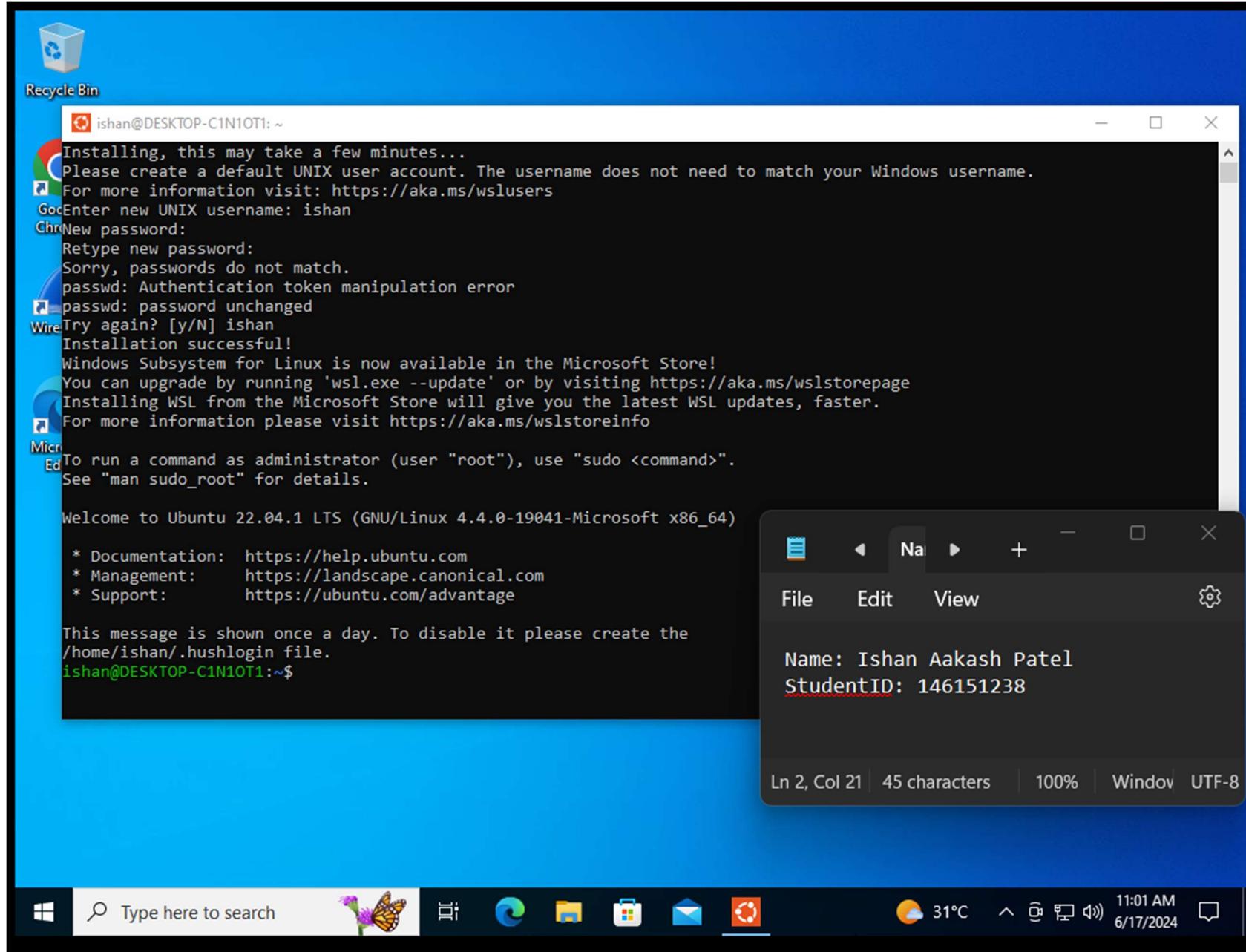
PS C:\Users\ishan\Downloads\Ubuntu2204-221101> dir

Directory: C:\Users\ishan\Downloads\Ubuntu2204-221101

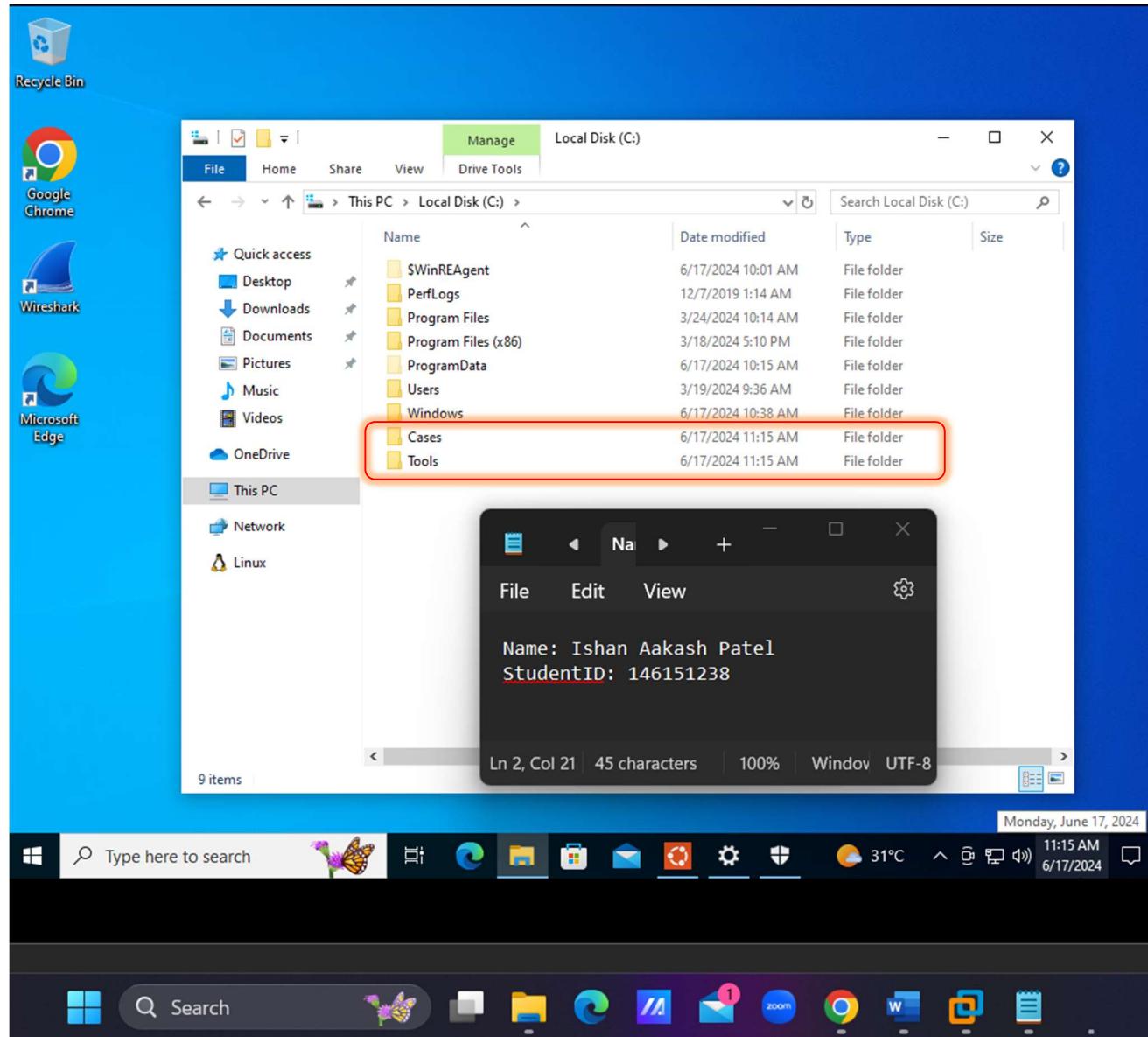
Mode                LastWriteTime         Length Name
----                -              -           -
d---- 6/17/2024 10:53 AM           338 AppxMetadata
-a---- 6/17/2024 10:53 AM           11964 AppxBlockMap.xml
-a---- 6/17/2024 10:53 AM           541855027 Ubuntu_2204.1.7.0_ARM64.appx
-a---- 6/17/2024 10:53 AM           37054 Ubuntu_2204.1.7.0_scale-100.appx
-a---- 6/17/2024 10:53 AM           43413 Ubuntu_2204.1.7.0_scale-125.appx
-a---- 6/17/2024 10:53 AM           48873 Ubuntu_2204.1.7.0_scale-150.appx
-a---- 6/17/2024 10:53 AM           99037 Ubuntu_2204.1.7.0_scale-400.appx
-a---- 6/17/2024 10:53 AM           574740468 Ubuntu_2204.1.7.0_x64.appx
-a---- 6/17/2024 10:53 AM           469 [Content_Types].xml

PS C:\Users\ishan\Downloads\Ubuntu2204-221101> Add-AppxPackage .\Ubuntu_2204.1.7.0_x64.appx
```

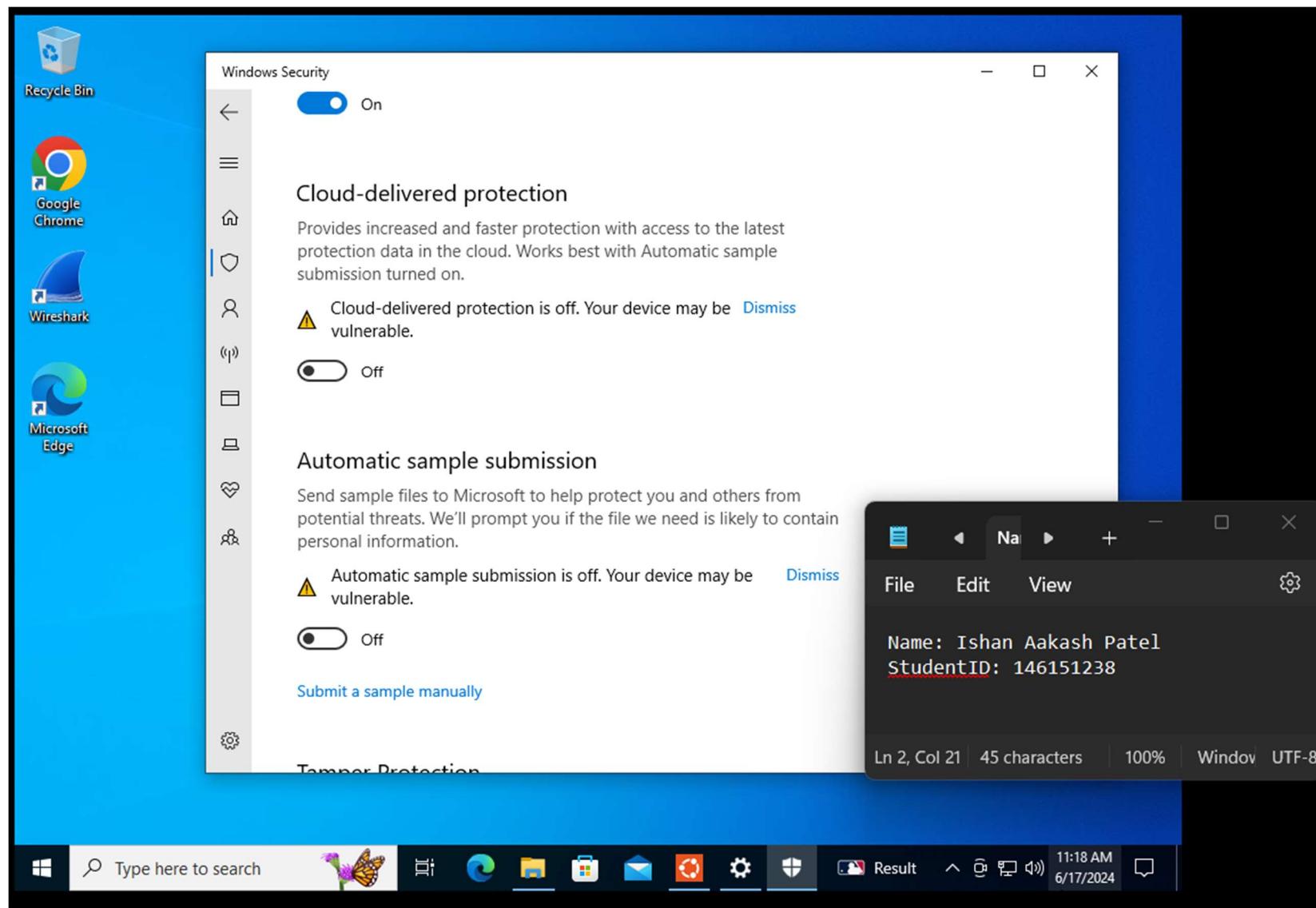
Name: Ishan Aakash Patel  
StudentID: 146151238



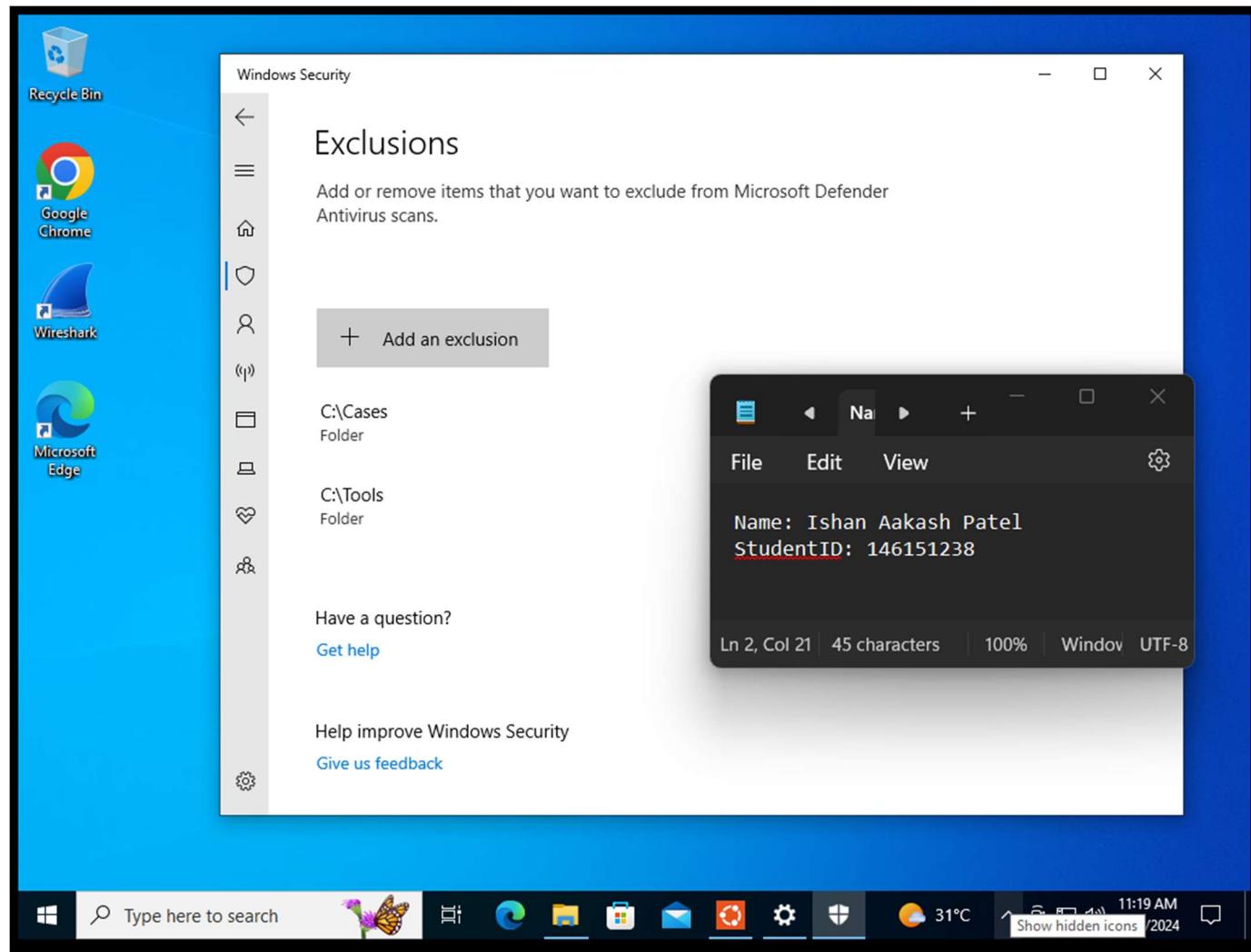
## 2) Create Cases & Tools in C Drive



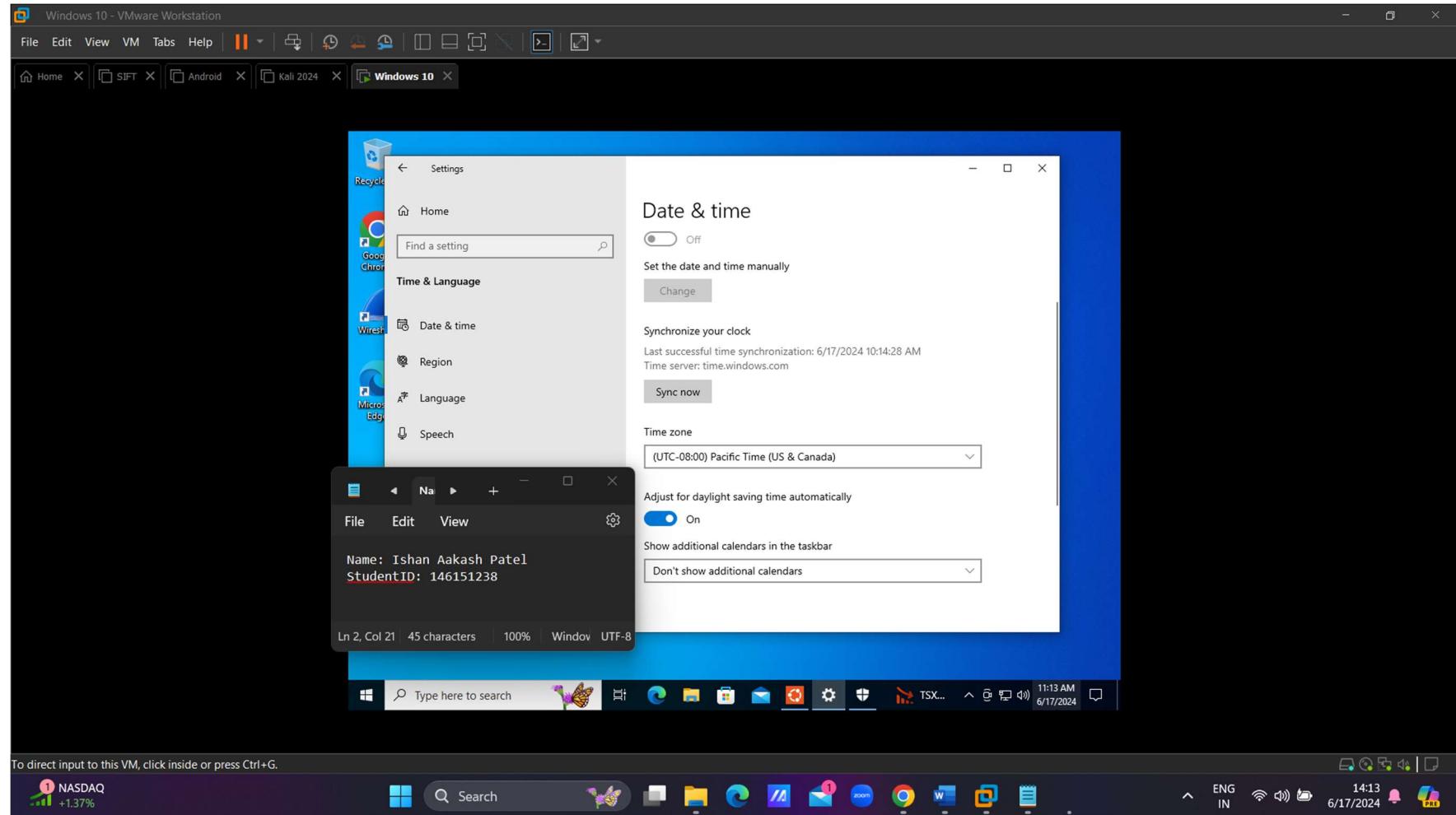
### 3) Configure Microsoft Defender



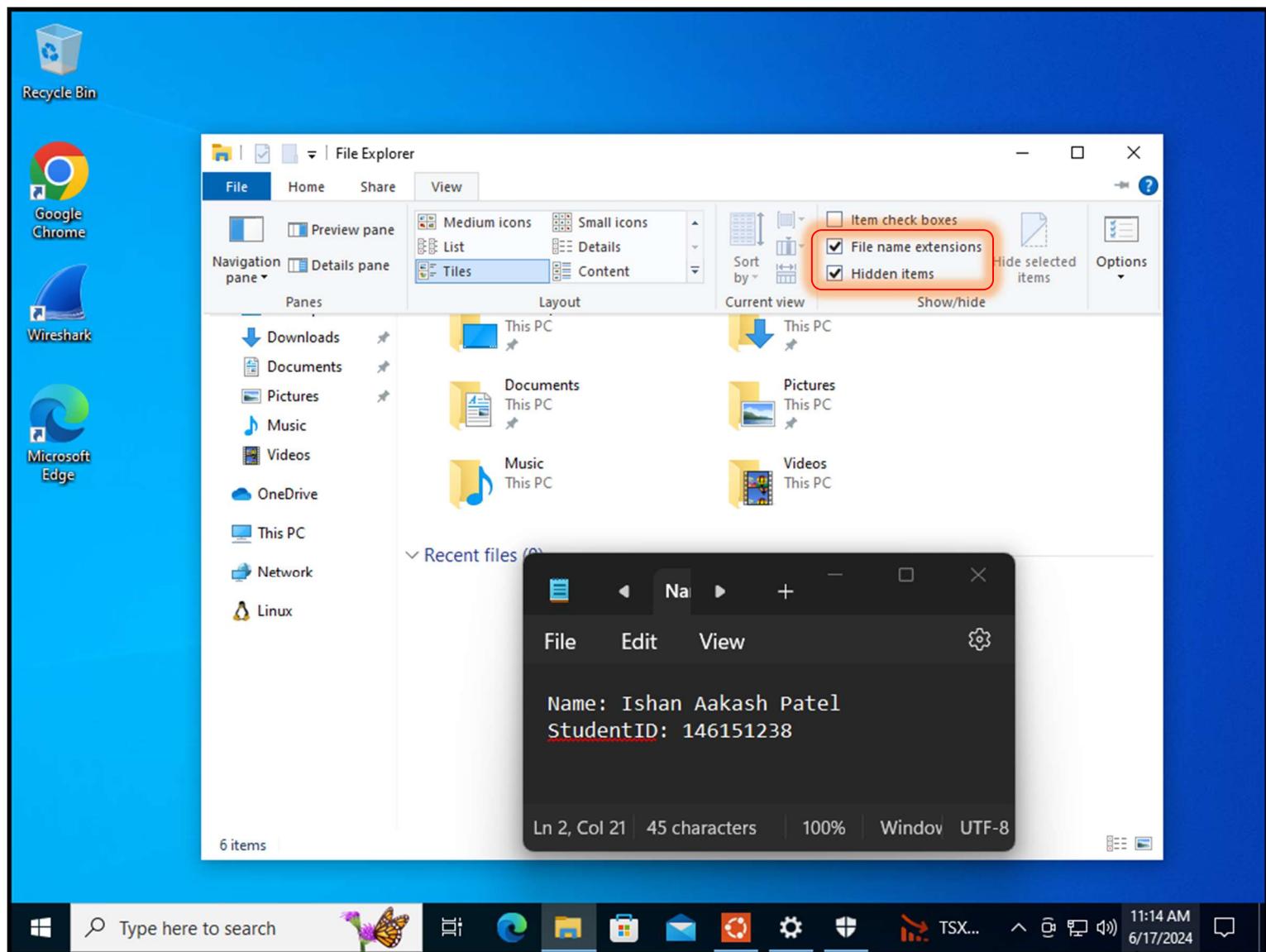
#### 4) Exclude the Working Directories (Cases and Tools)



## 5) Set timezone to UTC



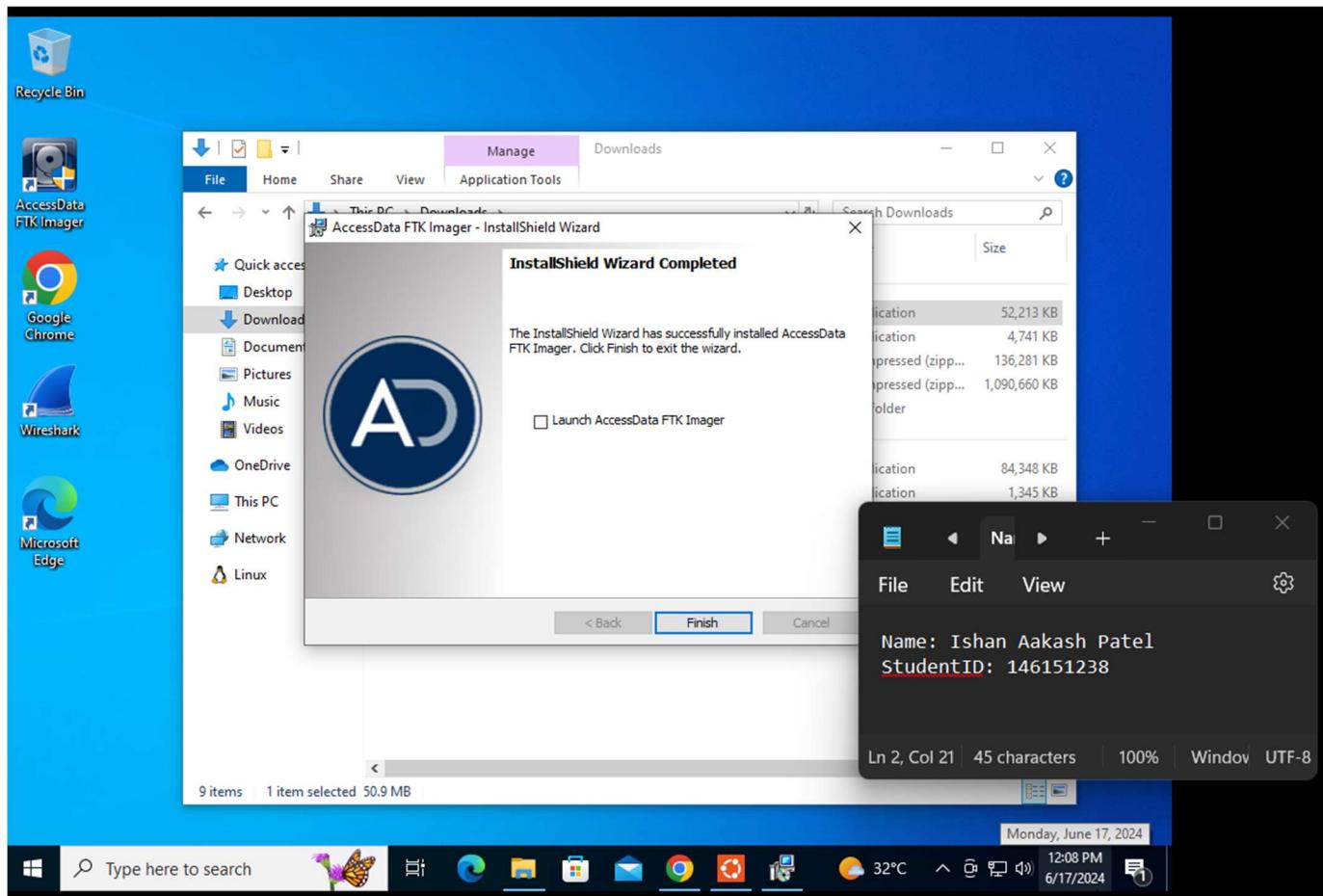
## 6) Show Hidden Files



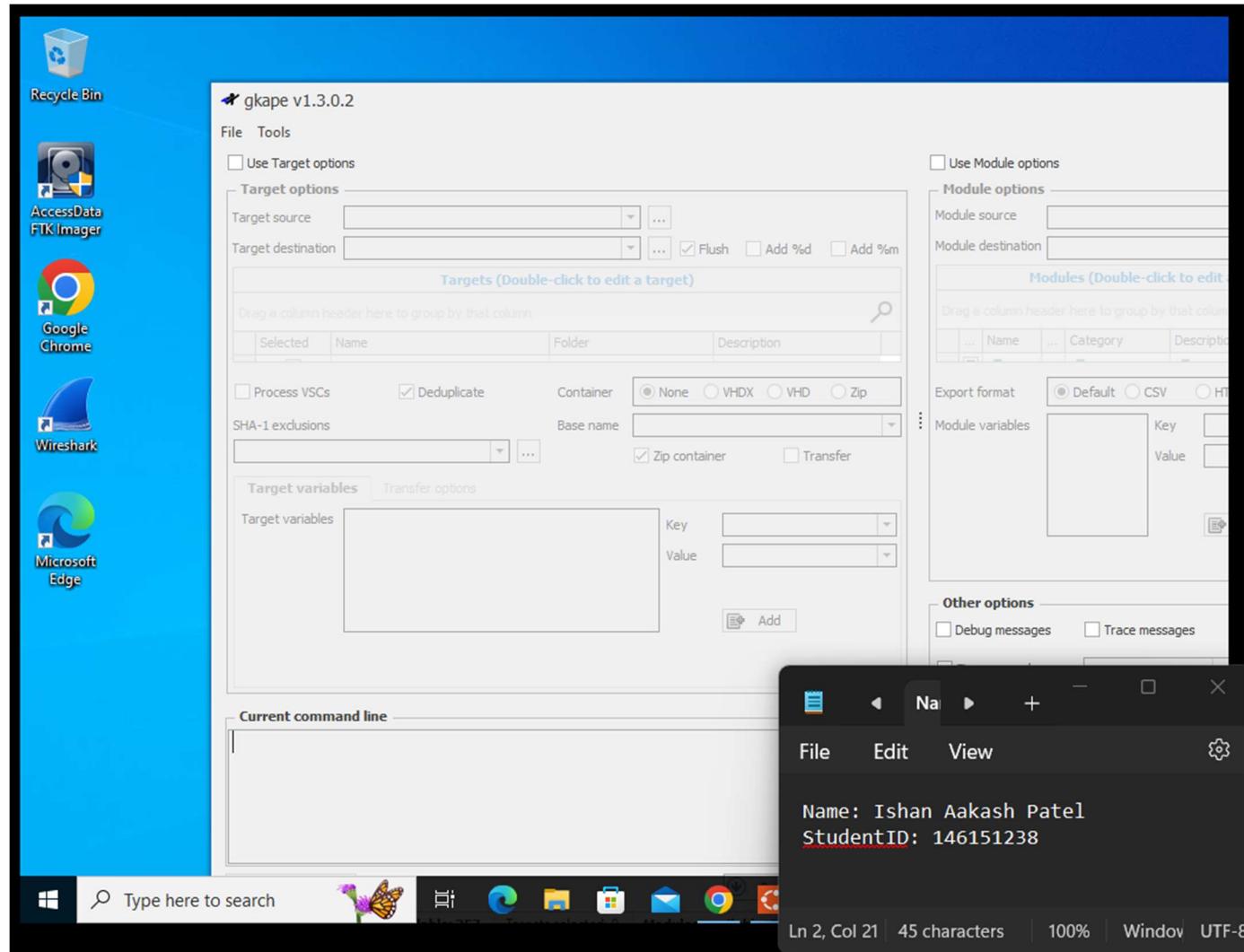
Now we will Install forensics tools.

## Tools

**1) FTK Imager :** FTK Imager, developed by AccessData, is a forensic imaging tool used to acquire and preview forensic images of computer systems and digital devices. It allows users to create exact bit-for-bit copies of hard drives, USB drives, and other storage media. Key features include the ability to view the contents of forensic images, mount images as drive letters, and export files and folders from images. FTK Imager also supports the generation of hash values for data integrity verification and can capture volatile memory. It is widely used in digital forensics for evidence preservation and analysis.



**2) KAPE** : KAPE (Kroll Artifact Parser and Extractor) is a digital forensics tool designed to quickly and efficiently collect and process forensic artifacts from Windows systems. Developed by Eric Zimmerman, KAPE streamlines the collection of key forensic artifacts, making it particularly useful for incident response and triage.



**3) Log2timeline** : Log2timeline, part of the Plaso project, is an open-source tool used in digital forensics for creating a timeline of events from various log files and forensic artifacts. It simplifies the process of aggregating different types of data sources to produce a single, unified timeline that can be analyzed to understand what happened on a system. First install Plaso tools

The screenshot shows a Windows desktop environment with a terminal window and a file viewer window.

**Terminal Window (Left):**

```
ishan@DESKTOP-C1N1OT1:~$ sudo add-apt-repository ppa:gift/stable
Repository: 'deb https://ppa.launchpadcontent.net/gift/stable/ubuntu/ jammy main'
Description:
  Periodic releases, contains periodic releases intended for non-development use
  More info: https://launchpad.net/~gift/+archive/ubuntu/stable
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/gift-ubuntu-stable-jammy.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/gift-ubuntu-stable-jammy.list
Adding key to /etc/apt/trusted.gpg.d/gift-ubuntu-stable.gpg with fingerprint 3ED1EAEC81894B17
Hit:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:5 https://ppa.launchpadcontent.net/gift/stable/ubuntu jammy InRelease [18.0 kB]
Get:6 https://ppa.launchpadcontent.net/gift/stable/ubuntu jammy/main amd64 Packages [58.9 kB]
Get:7 https://ppa.launchpadcontent.net/gift/stable/ubuntu jammy/main Translation-en [14.3 kB]
Fetched 91.2 kB in 4s (21.0 kB/s)
Reading package lists... Done
ishan@DESKTOP-C1N1OT1:~$ sudo apt-get install plaso-tools
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  artifacts-data libbbe libbbe-python3 libcaes libcaes-python3 libcreg libcreg-python3 libesedb libesedb-python3
  libevtx libevtx-python3 libevtx-python3 libewf libewf-python3 libfcrypto libfcrypto-python3 libfsapfs
  libfsapfs-python3 libfsextr libfsextr-python3 libfsfat libfsfat-python3 libfshfs libfshfs-python3 libfsntfs
  libfsntfs-python3 libfsxfs libfsxfs-python3 libfvde libfvde-python3 libfwnt libfwnt-python3 libfwsi libfwsi-python3
```

**File Viewer Window (Right):**

File	Edit	View
Name: Ishan Aakash Patel		
StudentID: 146151238		

Ln 2, Col 21 | 45 characters | 100% | Windows | UTF-8

• Download and install the Windows-based tools as listed in the table below.

A screenshot of a Windows desktop environment. In the center is a terminal window titled 'ishan@DESKTOP-C1N1OT1: ~'. The terminal displays the following output:

```
Setting up python3-plaso (20240308-1ppa1~jammy) ...
Setting up plaso-tools (20240308-1ppa1~jammy) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
ishan@DESKTOP-C1N1OT1:~/log2timeline.py -h
usage: log2timeline.py [-h] [--troubles] [-V] [--artifact_definitions PATH] [--custom_artifact_definitions PATH]
                      [--data PATH] [--archives TYPES] [--artifact_filters ARTIFACT_FILTERS]
                      [--artifact_filters_file PATH] [--extract_winreg_binary] [--preferred_year YEAR]
                      [--skip_compressed_streams] [-f FILE_FILTER] [--hasher_file_size_limit SIZE]
                      [--hashers HASHER_LIST] [--parsers PARSER_FILTER_EXPRESSION] [--yara_rules PATH]
                      [--partitions PARTITIONS] [--volumes VOLUMES] [--codepage CODEPAGE] [--language LANGUAGE_TAG]
                      [--no_extract_vinevt_resources] [-z TIME_ZONE] [--no_vss] [--vss_only]
                      [--vss_stores VSS_STORES] [--credential TYPE:DATA] [-d] [-q] [-u] [--info] [--use_markdown]
                      [--no_dependencies_check] [--logfile FILENAME] [--status_view TYPE] [--status_view_file PATH]
                      [--status_view_interval SECONDS] [--buffer_size BUFFER_SIZE] [--queue_size QUEUE_SIZE]
                      [--single_process] [--process_memory_limit SIZE] [--temporary_directory DIRECTORY]
                      [--vfs_back_end TYPE] [--worker_memory_limit SIZE] [--worker_timeout MINUTES]
                      [--workers WORKERS] [--sigsegv_handler] [--profilers PROFILERS_LIST]
                      [--profiling_directory DIRECTORY] [--profiling_sample_rate SAMPLE_RATE] [--storage_file PATH]
                      [--storage_format FORMAT] [--task_storage_format FORMAT]
                      [SOURCE]

log2timeline is a command line tool to extract events from individual
files, recursing a directory (e.g. mount point) or storage media
image or device.

More information can be gathered from here:
  https://plaso.readthedocs.io/en/latest/sources/user/Using-log2timeline.html

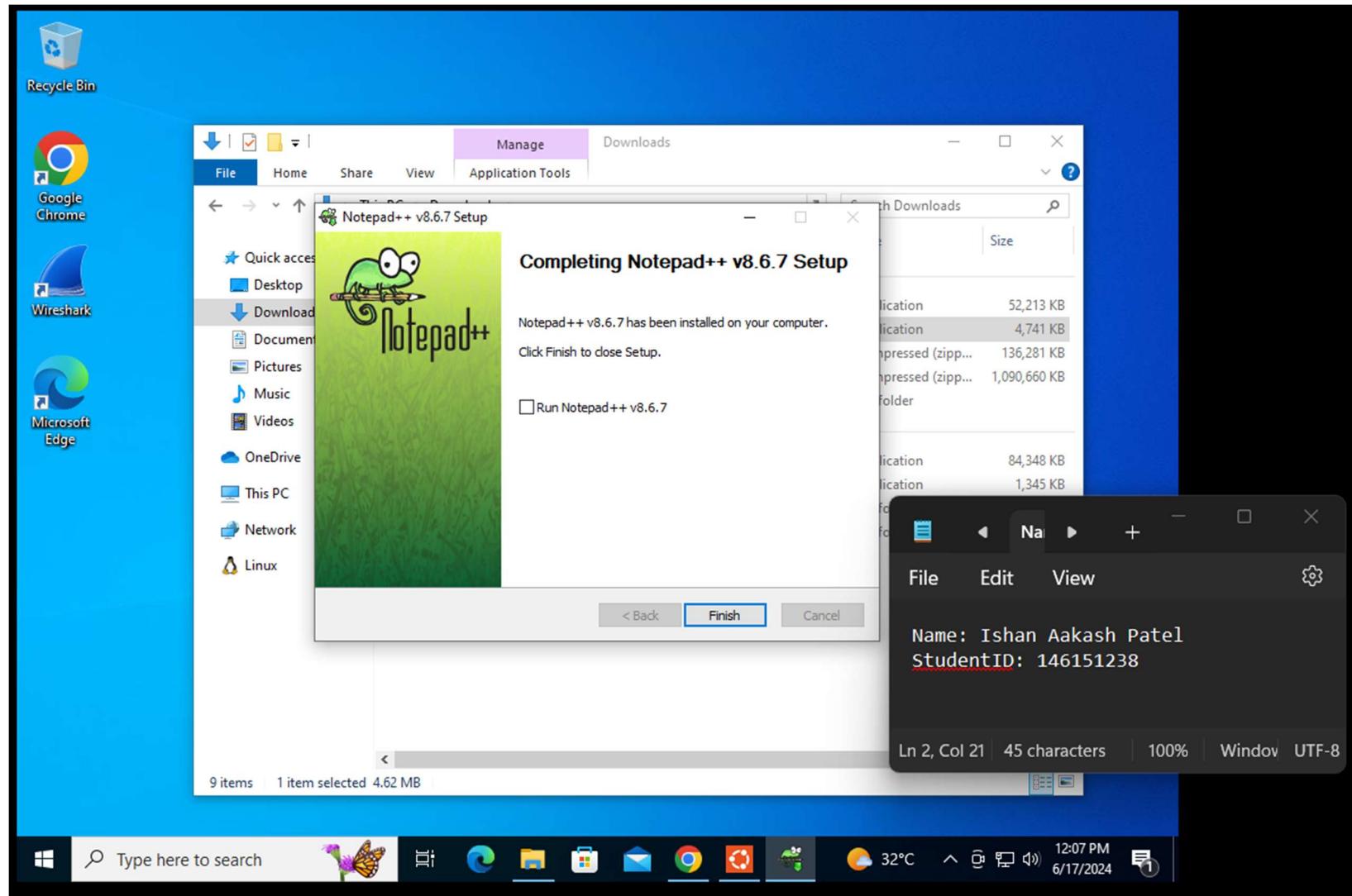
positional arguments:
```

The terminal window has a dark background and white text. Below the terminal is a file viewer window titled 'Name'. It contains the following information:

Name
Ishan Aakash Patel
StudentID: 146151238

The file viewer window has a dark background and white text. At the bottom of the screen, there is a taskbar with various icons and a search bar.

**4) Notepad++** : Notepad++ is a free, open-source text and source code editor for Windows. It supports multiple programming languages and features syntax highlighting, code folding, and a tabbed interface. Notepad++ is known for its speed, lightweight design, and extensive plugin ecosystem, making it a popular choice for programmers and text editors alike.



**5) Ole tools** : OLE forensics involves analyzing Object Linking and Embedding (OLE) objects within digital documents to uncover hidden data, metadata, and potential malicious content. This type of analysis is crucial in digital forensics, especially when examining documents such as Microsoft Office files, which often use OLE for embedding various objects.

The screenshot shows a Windows desktop environment. In the center, there is a terminal window titled "ishan@DESKTOP-C1N1OT1: ~". The terminal displays several lines of command-line output related to the installation of Python packages, specifically "oletools" and other dependencies. The output includes progress bars for package downloads. To the right of the terminal is a Notepad window with the following content:

```
Name: Ishan Aakash Patel  
StudentID: 146151238
```

At the bottom of the screen, the taskbar is visible with icons for File Explorer, Mail, and a browser. The system tray shows the date and time as "6/17/2024 12:00 PM".

```
ishan@DESKTOP-C1N1OT1: ~
Go Instead of answering questions, indicate some of the options on the
Chr command line (including data from particular VSS stores).
    log2timeline.py --vss_stores 1,2 /cases/plaso_vss.plaso image.E01

And that is how you build a timeline using log2timeline...
ishan@DESKTOP-C1N1OT1:~$ pip3 install oletools
WireDefaulting to user installation because normal site-packages is not writeable
Collecting oletools
  Downloading oletools-0.60.1-py2.py3-none-any.whl (977 kB)
    977.2/977.2 kB 1.2 MB/s eta 0:00:00
Collecting colorclass
  Downloading colorclass-2.2.2-py2.py3-none-any.whl (18 kB)
Collecting easygui
  Downloading easygui-0.98.3-py2.py3-none-any.whl (92 kB)
    92.7/92.7 kB 9.5 MB/s eta 0:00:00
Collecting olefile>=0.46
  Downloading olefile-0.47-py2.py3-none-any.whl (114 kB)
    114.6/114.6 kB 7.1 MB/s eta 0:00:00
Collecting pcodedmp>=1.2.5
  Downloading pcodedmp-1.2.6-py2.py3-none-any.whl (30 kB)
Collecting pyparsing<3,>=2.1.0
  Downloading pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
    67.8/67.8 kB 6.3 MB/s eta 0:00:00
Collecting mssofcrypto-tool
  Downloading mssofcrypto_tool-5.4.1-py3-none-any.whl (48 kB)
    48.1/48.1 kB 5.3 MB/s eta 0:00:00
Collecting cryptography>=35.0
  Downloading cryptography-42.0.8-cp39-abi3-manylinux_2_28_x86_64.whl (3.9 MB)
    3.9/3.9 kB 10.6 MB/s eta 0:00:00
Requirement already satisfied: cffi>=1.12 in /usr/lib/python3/dist-packages (from cryptography>=35.0->mssofcrypto-tool->v
```

A screenshot of a Windows desktop environment. At the top, there's a blue taskbar with various icons: Start, Search, File Explorer, Edge, Task View, Mail, Photos, and a system tray showing weather (32°C), battery (12:01 PM, 6/17/2024).

The main window is a terminal session titled "ishan@DESKTOP-C1N1OT1: ~". It shows the output of a command to install packages from a repository. The output includes several warning messages about script locations and PATH settings. The command "ole" was run, but it was not found, so the terminal suggests alternatives like "ols", "le", "mle", "gle", "ode", and "ale". Finally, it shows the command "sudo apt install <deb name>".

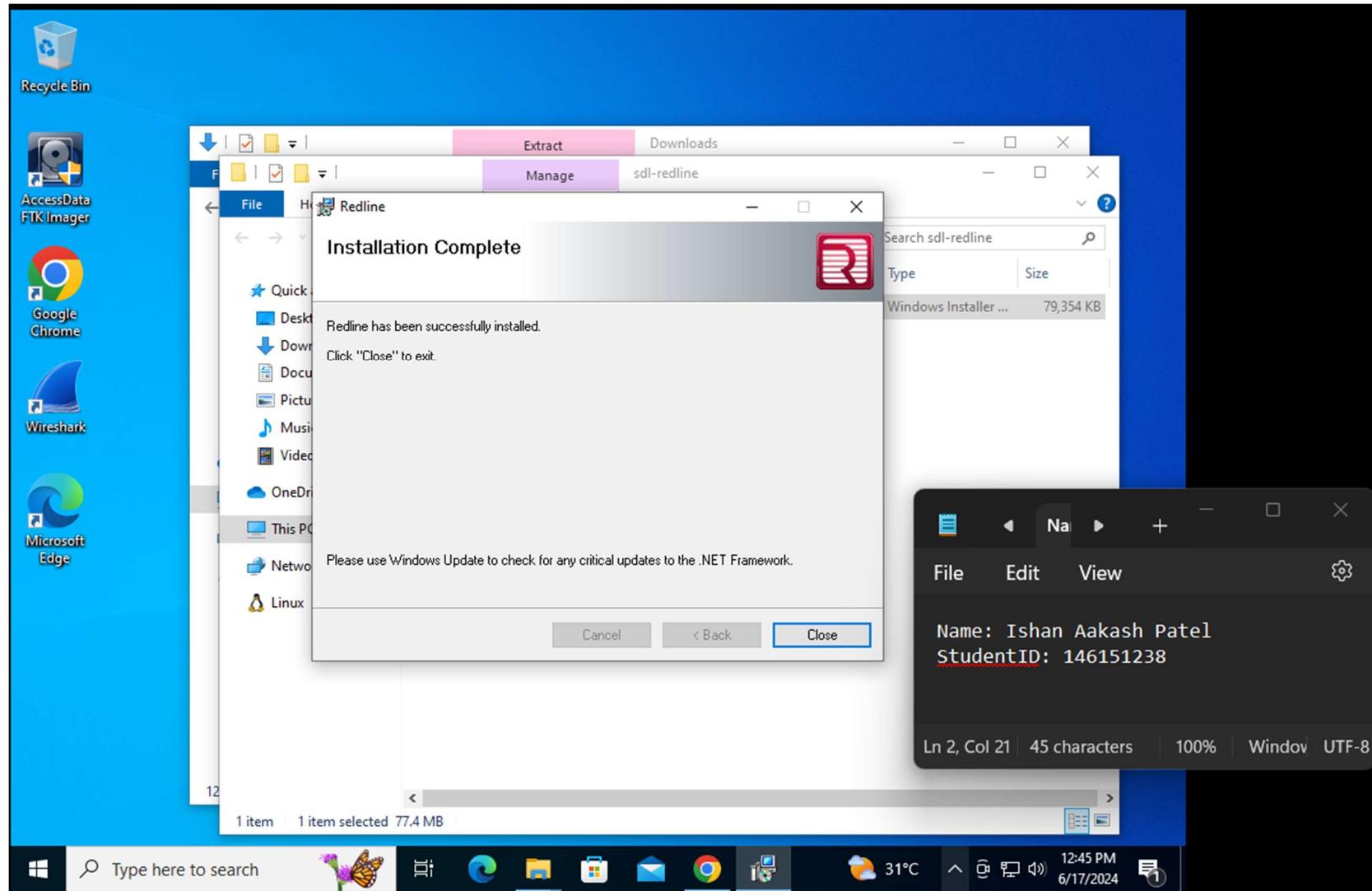
```
ishan@DESKTOP-C1N1OT1: ~$ ole
EdCommand 'ole' not found, did you mean:
  command 'ols' from deb speech-tools (1:2.5.0-12)
  command 'le' from deb le (1.16.8-0.1)
  command 'mle' from deb mle (1.4.3-2)
  command 'gle' from deb gle-graphics (4.2.5-9)
  command 'ode' from deb plotutils (2.6-11)
  command 'ale' from deb ale (0.9.0.3-5)
Try: sudo apt install <deb name>
ishan@DESKTOP-C1N1OT1: ~$ ole
ole32.dll      oleacc.dll      oleacchooks.dll  oleaccrc.dll    oleaut32.dll    oledlg.dll    oleprn.dll
```

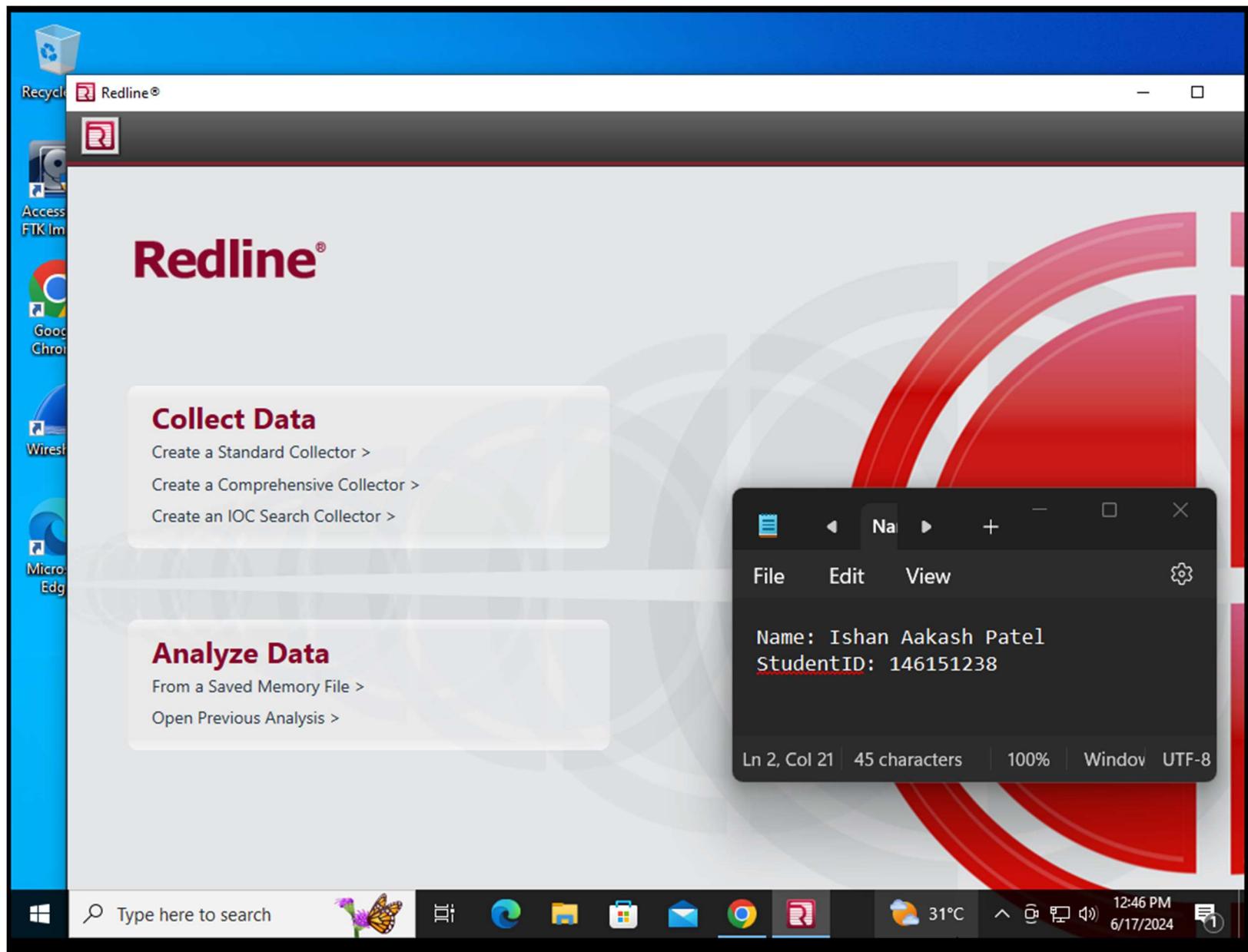
A secondary window titled "Name" is open, displaying personal information:

Name:	Ishan Aakash Patel
StudentID:	146151238

At the bottom right of the terminal window, status information is displayed: Ln 2, Col 21 | 45 characters | 100% | Window | UTF-8.

**6) Redline** : Redline, developed by FireEye, is a free endpoint security tool designed for memory and file analysis on Windows systems. It assists in identifying and analyzing potential security incidents by providing comprehensive information about the system's state.





## 7) Volatility

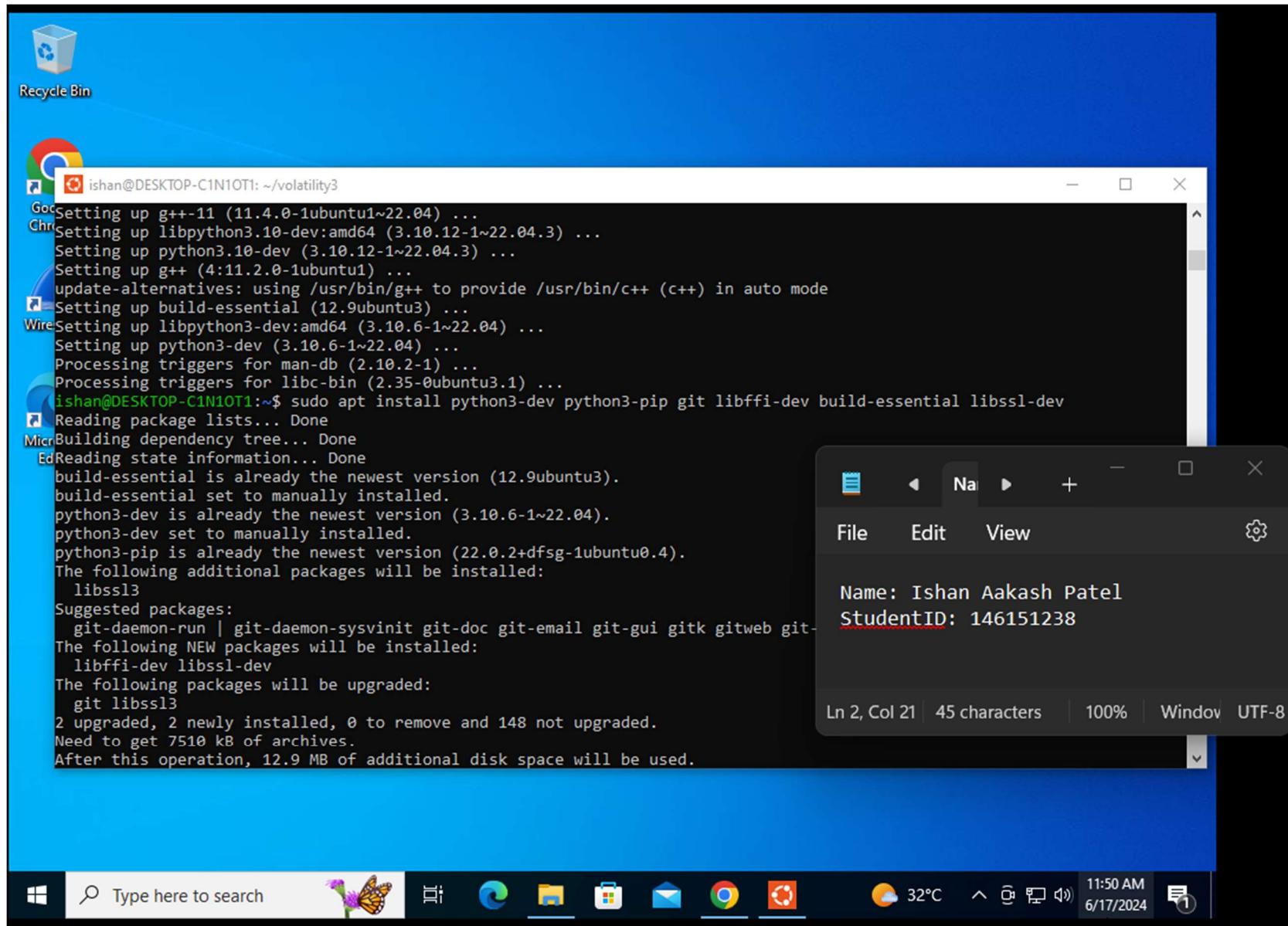
ishan@DESKTOP-C1N1OT1: ~/volatility3

```
Get:38 http://archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]
Get:39 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [27.2 kB]
Get:40 http://archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.3 kB]
Get:41 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [644 B]
Get:42 http://archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]
Fetched 31.8 MB in 14s (2314 kB/s)
Reading package lists... Done
ishan@DESKTOP-C1N1OT1:~$ sudo apt install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  build-essential bzip2 cpp cpp-11 dpkg-dev fakeroot fontconfig-config fonts-dejavu-core g++ g++-11 gcc gcc-11
    gcc-11-base gcc-12-base javascript-common libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl
    libasan6 libatomic1 libc-dev-bin libc-devtools libc6 libc6-dev libcc1-0 libcrypt-dev libdeflate0 libdpkg-perl
    libexpat1 libexpat1-dev libfakeroot libfile-fcntllock-perl libfontconfig1 libfreetype6 libgcc-11-dev libgcc-s1
    libgd3 libgomp1 libisl23 libitm1 libjbig0 libjpeg-turbo8 libjs-jquery libjs-sphinxdoc libjs-underscore
    liblsan0 libmpc3 libnsl-dev libpython3-dev libpython3.10 libpython3.10-dev libpython3.10-minimal
    libpython3.10-stdlib libquadmath0 libstdc++-11-dev libstdc++6 libtiff5 libtirpc-dev libtsan0 libubsan1 libwebp7
    libxpm4 linux-libc-dev lto-disabled-list make manpages-dev python3-dev python3-distutils python3-lib2to3
    python3-pkg-resources python3-setuptools python3-wheel python3.10 python3.10-dev python3.10-minimal rpcsvc-proto
    zlib1g-dev
Suggested packages:
  bzip2-doc cpp-doc gcc-11-locales debian-keyring g++-multilib g++-11-multilib gcc-11-doc gcc-multilib autoconf
  automake libtool flex bison gdb gcc-doc gcc-11-multilib apache2 | lighttpd glibc-doc bzr libgd-tools
  libstdc++-11-doc make-doc python-setuptools-doc python3.10-venv python3.10-doc binfmt-support
Recommended packages:
  libnss-nis libnss-nisplus
The following NEW packages will be installed:
  build-essential bzip2 cpp cpp-11 dpkg-dev fakeroot fontconfig-config fonts-dejavu-core g++ g++-11 gcc gcc-11
```

Name: Ishan Aakash Patel  
StudentID: 146151238

Ln 2, Col 21 | 45 characters | 100% | Window | UTF-8

Windows 11 taskbar icons: Start, Search, File Explorer, Mail, Chrome, Task View, Power, Volume, Network, Weather (32°C), Clock (11:49 AM), Date (6/17/2024).



A screenshot of a Windows desktop environment. At the top, there's a blue header bar with a 'Recycle Bin' icon. Below it is a dark-themed desktop background. In the center, there's a terminal window titled 'ishan@DESKTOP-C1N1OT1: ~/volatility3'. The terminal shows the user cloning the volatility3 repository from GitHub and then running a 'setup.py install --user' command. To the right of the terminal is a dark-themed note-taking application window with a title bar 'File Edit View' and a status bar 'Ln 2, Col 21 | 45 characters | 100% | Window | UTF-8'. The note contains the text 'Name: Ishan Aakash Patel' and 'StudentID: 146151238'. At the bottom, the Windows taskbar is visible with icons for File Explorer, Mail, and a browser. The system tray shows the date and time as '11:50 AM 6/17/2024'.

```
ishan@DESKTOP-C1N1OT1: ~volatility3
Setting up libssl-dev:amd64 (3.0.2-0ubuntu1.15) ...
Setting up git (1:2.34.1-1ubuntu1.11) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for install-info (6.8-4build1) ...
ishan@DESKTOP-C1N1OT1: $ git clone https://github.com/volatilityfoundation/volatility3.git
Cloning into 'volatility3'...
remote: Enumerating objects: 33448, done.
remote: Counting objects: 100% (4375/4375), done.
remote: Compressing objects: 100% (1139/1139), done.
remote: Total 33448 (delta 3821), reused 3503 (delta 3230), pack-reused 29073
Receiving objects: 100% (33448/33448), 6.53 MiB | 6.50 MiB/s, done.
Resolving deltas: 100% (25510/25510), done.
ishan@DESKTOP-C1N1OT1: ~$ cd volatility3/
ishan@DESKTOP-C1N1OT1: ~/volatility3$ ls
API_CHANGES.md MANIFEST.in doc requirements-minimal.txt test volatility3
CITATION.cff README.md mypy.ini requirements.txt vol.py volshell.py
LICENSE.txt development requirements-dev.txt setup.py vol.spec volshell.spec
ishan@DESKTOP-C1N1OT1: ~/volatility3$ python3 setup.py install --user
running install
/usr/lib/python3/dist-packages/setuptools/command/install.py:34: SetuptoolsDeprecationWarning: setup.py install is deprecated. Use build and pip and other standards-based tools.
  warnings.warn(
/usr/lib/python3/dist-packages/setuptools/command/easy_install.py:158: EasyInstallDeprecationWarning: easy_install command is deprecated. Use build and pip and other standards-based tools.
  warnings.warn(
/usr/lib/python3/dist-packages/pkg_resources/_init__.py:116: PkgResourcesDeprecationWarning: 1.1build1 is an invalid version and will not be supported in a future release
  warnings.warn(
running bdist_egg
```

A screenshot of a Windows desktop environment. In the center is a terminal window titled 'ishan@DESKTOP-C1N1OT1: ~/volatility3'. The terminal displays a log of command-line operations for installing Python packages:

```
ishan@DESKTOP-C1N1OT1: ~/volatility3
Creating 'dist/volatility3-2.7.1-py3.10.egg' and adding 'build/bdist.linux-x86_64/egg' to it
Removing 'build/bdist.linux-x86_64/egg' (and everything under it)
Processing volatility3-2.7.1-py3.10.egg
Creating /home/ishan/.local/lib/python3.10/site-packages/volatility3-2.7.1-py3.10.egg
Extracting volatility3-2.7.1-py3.10.egg to /home/ishan/.local/lib/python3.10/site-packages
Adding volatility3 2.7.1 to easy-install.pth file
Installing vol script to /home/ishan/.local/bin
Installing volshell script to /home/ishan/.local/bin

Installed /home/ishan/.local/lib/python3.10/site-packages/volatility3-2.7.1-py3.10.egg
Processing dependencies for volatility3==2.7.1
Searching for pefile>=2023.2.7
Reading https://pypi.org/simple/pefile/
Ed/usr/lib/python3/dist-packages/pkg_resources/_init__.py:116: PkgResourcesDeprecationWarning: is an invalid version and
will not be supported in a future release
    warnings.warn(
Downloading https://files.pythonhosted.org/packages/55/26/d0ad8b448476d0a1e8d3ea5622dc77b916db84c6aa3cb1e1c0965af948fc/pefile-2023.2.7-py3-none-any.whl#sha256=da185cd2af68c08a6cd4481f7325ed600a88f6a813bad9dea07ab3ef73d8d8d6
Best match: pefile 2023.2.7
Processing pefile-2023.2.7-py3-none-any.whl
Installing pefile-2023.2.7-py3-none-any.whl to /home/ishan/.local/lib/python3.10/site-packages
Adding pefile 2023.2.7 to easy-install.pth file

Installed /home/ishan/.local/lib/python3.10/site-packages/pefile-2023.2.7-py3.10.egg
Finished processing dependencies for volatility3==2.7.1
ishan@DESKTOP-C1N1OT1:~/volatility3$
```

To the right of the terminal is a dark-themed Notepad window. It contains the following text:

```
Name: Ishan Aakash Patel
StudentID: 146151238
```

The desktop taskbar at the bottom includes icons for the Start button, search bar, Microsoft Store, File Explorer, Mail, and Edge browser. The system tray shows the date (6/17/2024), time (11:51 AM), and temperature (32°C).

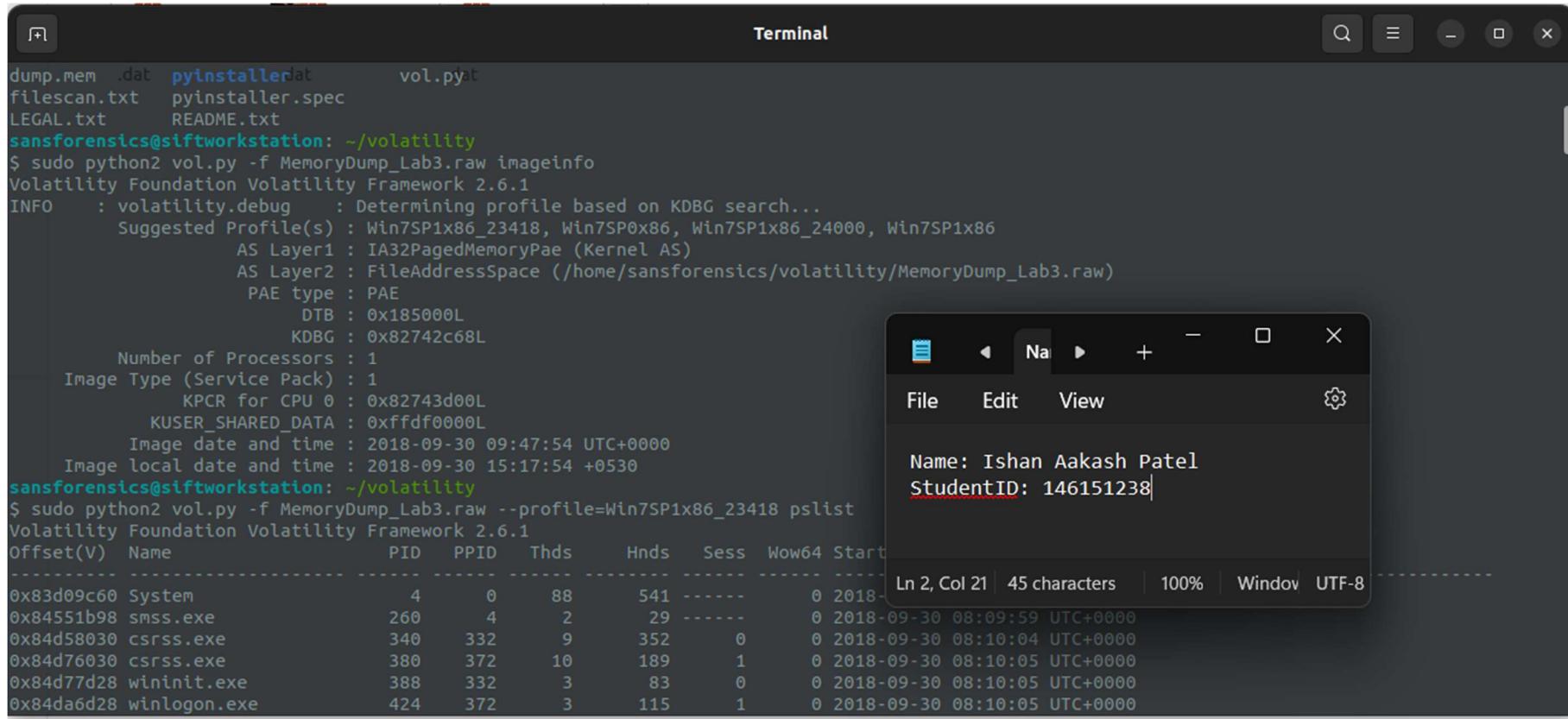
Part -1 is completed here as I no longer have to install the malware and scan my own image.

## **Part – 2 Memlabs Challenge 3**

I have done this challenge in SIFT workstation.

### **Step 1 : Get the image info**

Command : sudo python2 vol.py -f MemoryDump\_Lab3.raw imageinfo



The screenshot shows a terminal window titled "Terminal" and a text editor window titled "Name".

**Terminal Output:**

```
dump.mem .dat pyinstaller.dat      vol.pyat
filescan.txt  pyinstaller.spec
LEGAL.txt     README.txt
sansforensics@siftworkstation: ~/volatility
$ sudo python2 vol.py -f MemoryDump_Lab3.raw imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug      : Determining profile based on KDBG search...
Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86_24000, Win7SP1x86
                  AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                  AS Layer2 : FileAddressSpace (/home/sansforensics/volatility/MemoryDump_Lab3.raw)
                  PAE type : PAE
                  DTB : 0x185000L
                  KDBG : 0x82742c68L
Number of Processors : 1
Image Type (Service Pack) : 1
  KPCR for CPU 0 : 0x82743d00L
  KUSER_SHARED_DATA : 0xffffdf0000L
Image date and time : 2018-09-30 09:47:54 UTC+0000
Image local date and time : 2018-09-30 15:17:54 +0530
sansforensics@siftworkstation: ~/volatility
$ sudo python2 vol.py -f MemoryDump_Lab3.raw --profile=Win7SP1x86_23418 pslist
Volatility Foundation Volatility Framework 2.6.1
Offset(V) Name          PID  PPID  Thds  Hnds  Sess  Wow64 Start
-----  -----
0x83d09c60 System        4     0     88     541  -----  0 2018-09-30 08:09:59 UTC+0000
0x84551b98 smss.exe     260    4     2     29  -----  0 2018-09-30 08:10:04 UTC+0000
0x84d58030 csrss.exe    340    332    9     352     0  0 2018-09-30 08:10:05 UTC+0000
0x84d76030 csrss.exe    380    372   10     189     1  0 2018-09-30 08:10:05 UTC+0000
0x84d77d28 wininit.exe   388    332    3     83     0  0 2018-09-30 08:10:05 UTC+0000
0x84da6d28 winlogon.exe  424    372    3    115     1  0 2018-09-30 08:10:05 UTC+0000
```

**Text Editor Content:**

```
Name: Ishan Aakash Patel
StudentID: 146151238
```

**Bottom Status Bar:**

Ln 2, Col 21 | 45 characters | 100% | Window | UTF-8

First profile is Win7SP1x86\_23418

## Step 2: Using pslist find the malicious script

Command : sudo python2 vol.py -f MemoryDump\_Lab3.raw --Win7SP1x86\_23418 pslist

The screenshot shows a VMware Workstation interface with multiple windows open. The terminal window in the foreground displays the output of the volatility pslist command against a memory dump from a Windows 7 SP1 system. The command was run with the profile 'Win7SP1x86\_23418'. The output lists various processes with their PIDs, PPIDs, handles, sessions, and start times. A small modal window is overlaid on the terminal, displaying the user's name ('Ishan Aakash Patel') and student ID ('146151238').

Offset(V)	Name	.dat	PID	PPID	Thds	Hnds	Sess	Wow64	Start	Exit
0x83d09c60	System		4	0	88	541	-----	0	2018-09-30 08:09:59 UTC+0000	
0x84551b98	smss.exe		260	4	2	29	-----	0	2018-09-30 08:09:59 UTC+0000	
0x84d58030	csrss.exe		340	332	9	352	0	0	2018-09-30 08:10:04 UTC+0000	
0x84d76030	cssrss.exe		380	372	10	189	1	0	2018-09-30 08:10:05 UTC+0000	
0x84d77d28	wininit.exe		388	332	3	83	0	0	2018-09-30 08:10:05 UTC+0000	
0x84da6d28	winlogon.exe		424	372	3	115	1	0	2018-09-30 08:10:05 UTC+0000	
0x84dcdbd0	services.exe		484	388	6	195	0	0	2018-09-30 08:10:07 UTC+0000	
0x84dd0658	lsass.exe		492	388	6	561	0	0	2018-09-30 08:10:08 UTC+0000	
0x84dd4b28	lsm.exe		500	388	10	151	0	0	2018-09-30 08:10:08 UTC+0000	
0x8454e348	svchost.exe		588	484	10	351	0	0	2018-09-30 08:10:12 UTC+0000	
0x84e15d28	VBoxService.exe		648	484	12	115	0	0	2018-09-30 08:10:13 UTC+0000	
0x84e1d030	svchost.exe		712	484	8	268	0	0	2018-09-30 08:10:14 UTC+0000	
0x84e5ad28	svchost.exe		800	484	18	438	0	0	2018-09-30 08:10:14 UTC+0000	
0x84e67d28	svchost.exe		852	484	16	371	0	0	2018-09-30 08:10:15 UTC+0000	
0x84e6b030	svchost.exe		880	484	18	452	0	0	2018-09-30 08:10:15 UTC+0000	
0x84edfa18	svchost.exe		904	484	31	1116	0	0	2018-09-30 08:10:15 UTC+0000	
0x8481bc00	svchost.exe		1236	484	15	478	0	0	2018-09-30 08:10:22 UTC+0000	
0x8484a800	spoolsv.exe		1340	484	12	285	0	0	2018-09-30 08:10:24 UTC+0000	
0x8485b030	svchost.exe		1368	484	18	302	0	0	2018-09-30 08:10:24 UTC+0000	
0x8488e860	svchost.exe		1488	484	11	267	0	0	2018-09-30 08:10:26 UTC+0000	
0x84893030	svchost.exe		1516	484	12	215	0	0	2018-09-30 08:10:26 UTC+0000	
0x85192030	LogonUI.exe		876	388	5	152	0	0	2018-09-30 08:10:40 UTC+0000	
0x8514cae0	sppsvc.exe		292	484	6	153	0	0	2018-09-30 08:12:31 UTC+0000	
0x8514bbf0	svchost.exe		440	484	13	342	0	0	2018-09-30 08:12:32 UTC+0000	
0x84d69d00	SearchIndexer.exe		1184	484	15	724	0	0	2018-09-30 08:12:33 UTC+0000	
0x8441d7e0	taskhost.exe		4816	484	8	196	1	0	2018-09-30 09:28:32 UTC+0000	
0xa0b21170	dwm.exe		3028	852	3	186	1	0	2018-09-30 09:28:36 UTC+0000	
0x8449d890	explorer.exe		5300	5128	30	871	1	0	2018-09-30 09:28:36 UTC+0000	
0x851cdd28	VBoxTray.exe		3064	5300	14	154	1	0	2018-09-30 09:28:44 UTC+0000	
0x84d77868	wuauctl.exe		5644	904	3	86	1	0	2018-09-30 09:28:49 UTC+0000	
0x9c627d28	msiexec.exe		1016	484	7	345	0	0	2018-09-30 09:39:03 UTC+0000	
0xbc2d08a8	msiexec.exe		5652	1016	0	-----	1	0	2018-09-30 09:39:13 UTC+0000	2018-09-30 09:41:17 UTC+0000
0xbc21b9f0	TrustedInstall		4724	484	4	139	0	0	2018-09-30 09:40:24 UTC+0000	
0x84489800	audiocda.exe		5996	800	4	120	0	0	2018-09-30 09:45:22 UTC+0000	

Two instances found of notepad.exe, now lets look at the command-line of this...

### Step 3 : Command-line arguments of notepad processes using cmdline plugin.

Command : sudo python2 vol.py -f MemoryDump\_Lab3.raw cmdline | grep -i notepad.exe

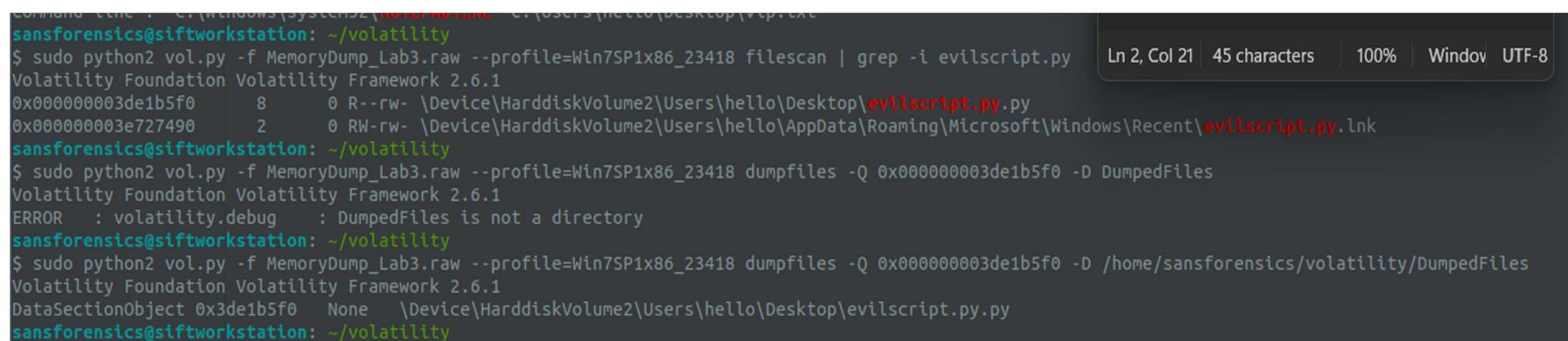


```
sansforensics@siftworkstation: ~/volatility
$ sudo python2 vol.py -f MemoryDump_Lab3.raw cmdline | grep -i notepad.exe
Volatility Foundation Volatility Framework 2.6.1
sansforensics@siftworkstation: ~/volatility
$ sudo python2 vol.py -f MemoryDump_Lab3.raw --profile=Win7SP1x86_23418 cmdline | grep -i notepad.exe
Volatility Foundation Volatility Framework 2.6.1
notepad.exe pid: 3736
Command line : "C:\Windows\system32\NOTEPAD.EXE" C:\Users\hello\Desktop\evilscript.py
notepad.exe pid: 3432
Command line : "C:\Windows\system32\NOTEPAD.EXE" C:\Users\hello\Desktop\vip.txt
sansforensics@siftworkstation: ~/volatility
```

In the above output we found two malicious scripts 1) evilscript.py 2) vip.txt, now lets dump them...

### Step 4 : Dump the malicious scripts

Command : 1) sudo python2 vol.py -f MemoryDump\_Lab3.raw --Win7SP1x86\_23418 filescan | grep -i evilscript.py  
2) sudo python2 vol.py -f MemoryDump\_Lab3.raw --profile=Win7SP1x86\_23418 dumpfiles -Q 0x000000003de1b5f0 -D /home/sansforensics/volatility/DumpedFiles



```
Command line : "C:\Windows\system32\NOTEPAD.EXE" C:\Users\hello\Desktop\vip.txt
sansforensics@siftworkstation: ~/volatility
$ sudo python2 vol.py -f MemoryDump_Lab3.raw --profile=Win7SP1x86_23418 filescan | grep -i evilscript.py
Volatility Foundation Volatility Framework 2.6.1
0x000000003de1b5f0      8      0 R--rw- \Device\HarddiskVolume2\Users\hello\Desktop\evilscript.py.py
0x000000003e727490      2      0 RW-rw- \Device\HarddiskVolume2\Users\hello\AppData\Roaming\Microsoft\Windows\Recent\evilscript.py.lnk
sansforensics@siftworkstation: ~/volatility
$ sudo python2 vol.py -f MemoryDump_Lab3.raw --profile=Win7SP1x86_23418 dumpfiles -Q 0x000000003de1b5f0 -D DumpedFiles
Volatility Foundation Volatility Framework 2.6.1
ERROR   : volatility.debug   : DumpedFiles is not a directory
sansforensics@siftworkstation: ~/volatility
$ sudo python2 vol.py -f MemoryDump_Lab3.raw --profile=Win7SP1x86_23418 dumpfiles -Q 0x000000003de1b5f0 -D /home/sansforensics/volatility/DumpedFiles
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x3de1b5f0  None  \Device\HarddiskVolume2\Users\hello\Desktop\evilscript.py.py
sansforensics@siftworkstation: ~/volatility
```

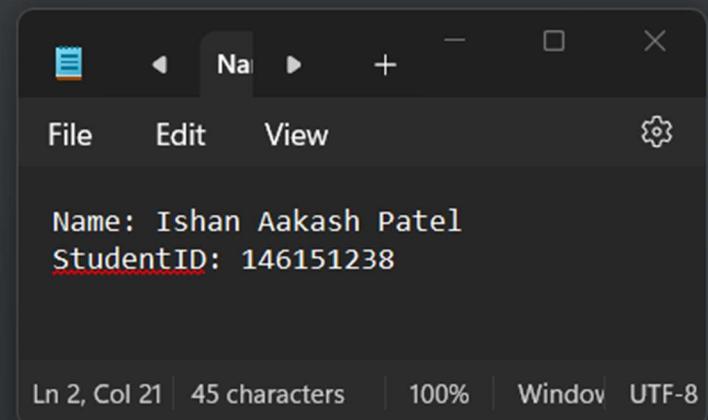
## Step 5 : Open the script (evilscript.py)

```
sansforensics@siftworkstation: ~/volatility
$ cd DumpedFiles/
sansforensics@siftworkstation: ~/volatility/DumpedFiles
$ ls Downloads
file.None.0xbc2b6af0.dat
sansforensics@siftworkstation: ~/volatility/DumpedFiles
$ cat file.None.0xbc2b6af0.dat
import sys
import string

def xor(s):
    a = ''.join(chr(ord(i)^3) for i in s)
    return a
+ Other Locations

def encoder(x):
    return x.encode("base64")

if __name__ == "__main__":
    f = open("C:\\\\Users\\\\hello\\\\Desktop\\\\vip.txt", "w")
    arr = sys.argv[1]
    arr = encoder(xor(arr))
    f.write(arr)
    f.close()
sansforensics@siftworkstation: ~/volatility/DumpedFiles
```

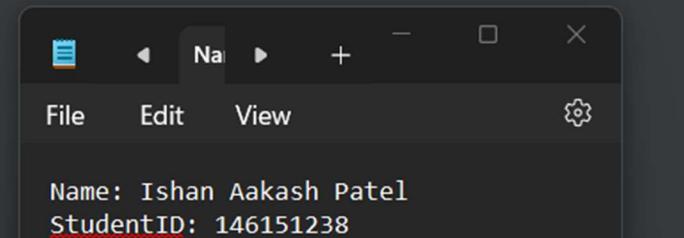


The script receives a one-string argument as input and then xor its characters with value 0x3. Then, apply base64 encoding on the result and save it to the text file “vip.txt”.

### Step 6 : Dump vip.txt

Command : 1) sudo python2 vol.py -f MemoryDump\_Lab3.raw --Win7SP1x86\_23418 filescan | grep -i vip.txt  
2) sudo python2 vol.py -f MemoryDump\_Lab3.raw --profile=Win7SP1x86\_23418 dumpfiles -Q 0x000000003e727e50  
-D /home/sansforensics/volatility/DumpedFiles

```
sansforensics@siftworkstation:~/volatility$ 0xfcfc38.      0xbc2b6af0
$ sudo python2 vol.py -f MemoryDump_Lab3.raw --profile=Win7SP1x86_23418 filescan | grep -i vip.txt
Volatility Foundation Volatility Framework 2.6.1
0x000000003e727e50      8      0 -W-rw- \Device\HarddiskVolume2\Users\hello\Desktop\vip.txt
sansforensics@siftworkstation: ~/volatility
$ cd DumpedFiles/
sansforensics@siftworkstation: ~/volatility/DumpedFiles
$ sudo python2 vol.py -f MemoryDump_Lab3.raw --profile=Win7SP1x86_23418 dumpfiles -Q 0x000000003e727e50 -D /home/sansforensics/volatility/DumpedFiles
python2: can't open file 'vol.py': [Errno 2] No such file or directory
sansforensics@siftworkstation: ~/volatility/DumpedFiles
$ cd ..
sansforensics@siftworkstation: ~/volatility
$ sudo python2 vol.py -f MemoryDump_Lab3.raw --profile=Win7SP1x86_23418 dumpfiles -Q 0x000000003e727e50 -D /home/sansforensics/volatility/DumpedFiles
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x3e727e50  None  \Device\HarddiskVolume2\Users\hello\Desktop\vip.txt
sansforensics@siftworkstation: ~/volatility
$ cd DumpedFiles/
sansforensics@siftworkstation: ~/volatility/DumpedFiles
$ ls
file.None.0x83e52420.dat  file.None.0xbc2b6af0.dat
sansforensics@siftworkstation: ~/volatility/DumpedFiles
$ cat file.None.0x83e52420.dat
am1gd2V4M20wXGs3b2U=
```



Now we can convert this string to flag through cyberchef.

## Step 7 : Cyberchef

The screenshot shows the CyberChef interface running in a Firefox browser window on a Kali Linux VM. The left sidebar lists various operations like XOR, XOR Brute Force, and Magic. The main area shows a 'From Base64' recipe being used on the input 'jam1gd2V4M20wXGs3b2U=' with a key of '03'. The output is 'inctf{0n3\_h4lf}'.

Operations

- XOR
- XOR
- XOR Brute Force
- XKCD Random Number
- Hex to Object Identifier
- Unicode Text Format
- Text Encoding Brute Force
- Lorenz
- Magic
- Favourites
- Data format
- Encryption / Encoding
- Public Key
- Arithmetic / Logic
- Networking

Recipe

From Base64

Input: jam1gd2V4M20wXGs3b2U=

XOR

Key: 03 Scheme: Standard Null preserving

Output: inctf{0n3\_h4lf}

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

Firefox toolbar: Back, Forward, Stop, Refresh, Home, Address bar (https://gchq.github.io/CyberChef/#recipe=From\_Base64('A-Za-z0-9%2B%3D',true,false)XOR({'option':'Hex','string':'03'},'Standard',false)&input=YW0xZ2QyVjRNMjB3WEdzM2lyVT0), Search, Zoom, File, Edit, View, VM, Help.

Bottom status bar: Weather (86°F, Mostly cloudy), Date (7/9/2024), Time (17:11), ENG IN, Battery, Signal strength.

Half flag : inctf{0n3\_h4lf}

**Step 8 : Recovering the hidden data:** The second clue suggests using the steghide tool, which is designed for hiding data within images or audio files. According to the documentation, steghide supports JPEG, BMP, WAV, and AU file formats. Using these extensions, I searched for the file that conceals the data. I got lucky with a JPEG file, as shown in the following command.

Command : sudo python2 vol.py -f MemoryDump\_Lab3.raw --profile=Win7SP1x86\_23418 filescan | grep -i jpeg

The screenshot shows a terminal window with the following command and output:

```
sansforensics@siftworkstation: ~/volatility
$ sudo python2 vol.py -f MemoryDump_Lab3.raw --profile=Win7SP1x86_23418 filescan | grep -i jpeg
Volatility Foundation Volatility Framework 2.6.1
0x00000000004f34148      2      0 RW---- \Device\HarddiskVolume2\Users\hello\Desktop\suspision1.jpeg
sansforensics@siftworkstation: ~/volatility
$ sudo python2 vol.py -f MemoryDump_Lab3.raw --profile=Win7SP1x86_23418 dumpfiles -Q 0x00000000004f34148 -D /home/sansforensics/volatility/DumpedFiles
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x04f34148  None  \Device\HarddiskVolume2\Users\hello\Desktop\suspision1.jpeg
sansforensics@siftworkstation: ~/volatility
$ cd DumpedFiles/
sansforensics@siftworkstation: ~/volatility/DumpedFiles
$ ls
```

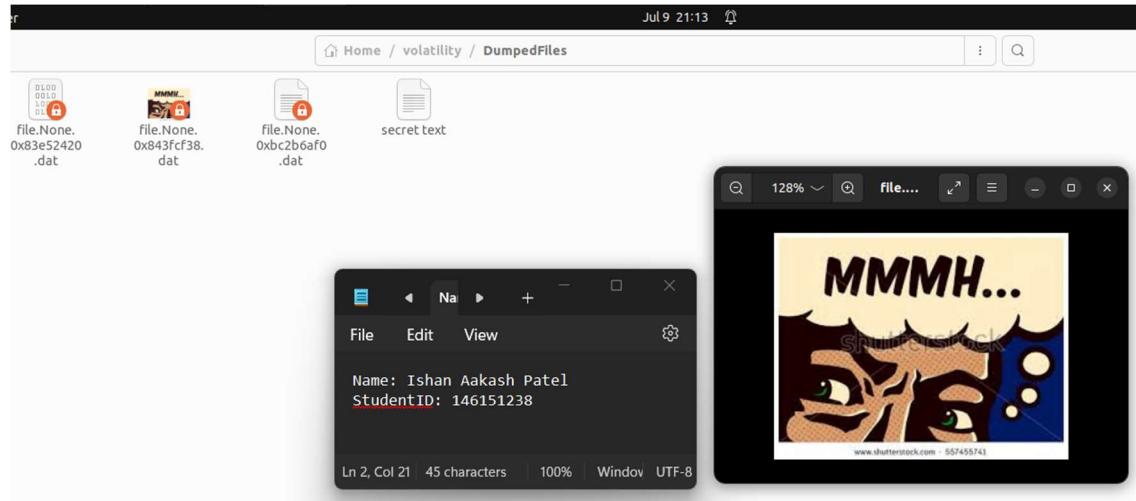
The terminal window has a status bar at the top right showing "Ln 2, Col 21 | 45 characters | 100% | Window UTF-8".

Below the terminal is a docked application bar with icons for various tools like a terminal, browser, file manager, and others. A weather widget on the left shows "86°F Mostly cloudy".

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

After dumping the file here is the output.

## Step 9 : Output

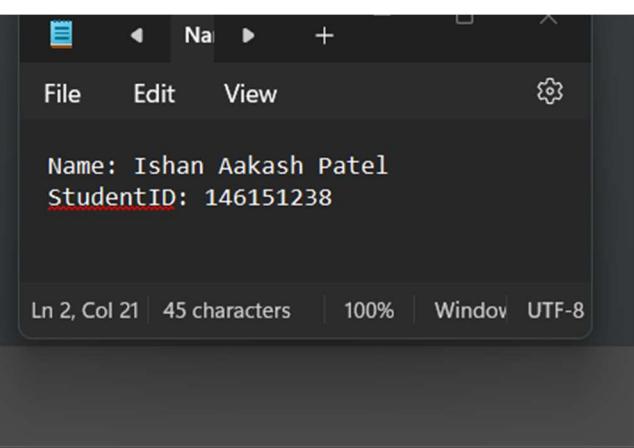


## Step 10 : Now use steghide tool to extract the hidden data.

Password was the first half of the flag

Second flag : \_1s\_n0t\_3n0ugh{

```
sansforensics@siftworkstation: ~/volatility/DumpedFiles
$ steghide extract -sf file.None.0x843fcf38.dat
Enter passphrase:
wrote extracted data to "secret text".
sansforensics@siftworkstation: ~/volatility/DumpedFiles
$ ls
file.None.0x83e52420.dat  file.None.0x843fcf38.dat  file.None.0xbc2b6af0.dat  'secret text'
sansforensics@siftworkstation: ~/volatility/DumpedFiles
$ cat secret\ text
_1s_n0t_3n0ugh}
sansforensics@siftworkstation: ~/volatility/DumpedFiles
$
```



## **Learning Experience**

To complete my forensics workstation setup and memory analysis project, I followed a detailed guide provided by Blue Cape Security. Setting up the workstation involved installing essential tools like Volatility and Cyber Triage, ensuring they were functional for memory and malware analysis. Throughout the process, I documented each step with screenshots, ensuring to capture my desktop with the date/time visible, and showing my logged-in account details.

Using Volatility, I analyzed memory dumps from the MemLabs challenges, applying techniques learned from online resources. This practical exercise not only honed my technical skills but also deepened my understanding of forensic investigation methodologies. Reflecting on this experience, I gained insights into the complexities of digital forensics, particularly in analyzing memory artifacts to uncover potential security breaches or malicious activities. This project underscored the importance of meticulous documentation and methodical analysis in cybersecurity investigations, skills that are crucial in today's digital landscape.