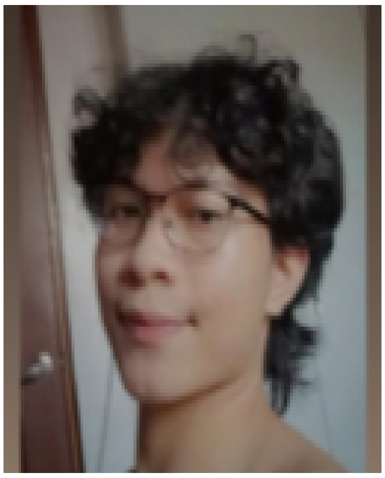


NAME : EJERCITO, MARLON JASON A.

SUBJECT : CPE 243-CPE32S6 – Enterprise Security

PROFESSOR : Engr. KRISELYN B. CABADING

DATE/TIME/RM : 03/23/2024 - 1:30 PM to 6:30 PM - Q5204

ACTIVITY: Hands-on Activity 8.1: Playbook for Extracting an Executable from PCAP

SEAT NO.: 10 SCORE: ___________________________

Lab - Playbook for Extracting an Executable from PCAP

Objectives

Part 1: Create a Manage node and Control node (Choose Ubuntu or CentOS)

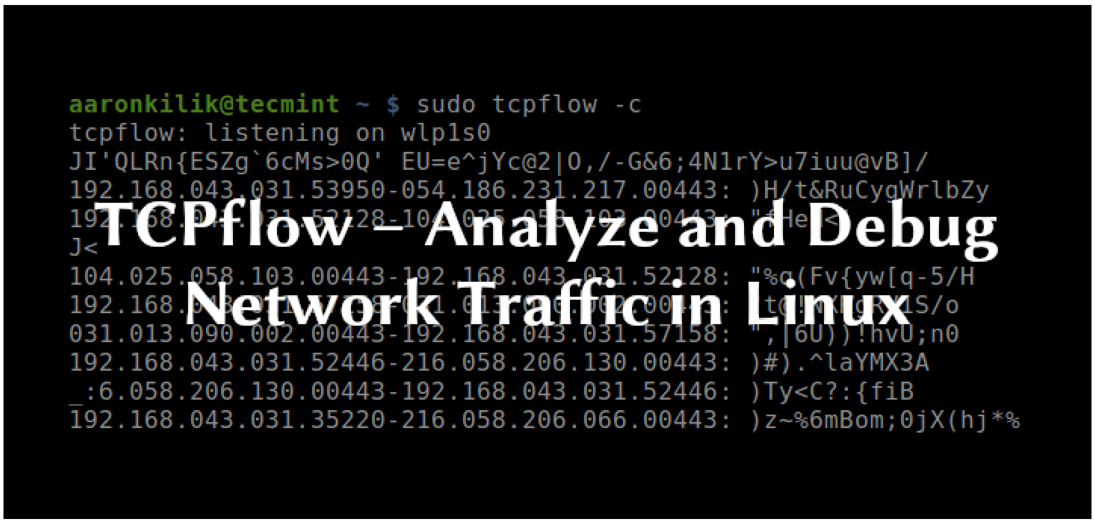
Part 2: Implement network using SSH-key-based authentication

Part 3: Create a playbook that allows the Manage node to extract an executable from PCAP in the Control Node

Part 4: Show input (codes), process (successful run), and output (evidence that executables are collected)

Background / Scenario

Ansible is an open-source automation platform renowned for its agentless architecture, enabling seamless orchestration of IT tasks across diverse environments. Leveraging human-readable YAML syntax, Ansible allows users to define automation tasks in playbooks, facilitating the automation of software deployment, configuration management, and infrastructure provisioning. Its idempotent nature ensures reliability, while its modular design promotes easy integration with other tools and platforms. With Ansible, organizations can streamline operations, increase efficiency, and focus on strategic initiatives, all while minimizing deployment complexity and maximizing scalability.



"tcpflow" is a powerful network analysis tool used to capture and record TCP connections passing through a network interface. It intercepts data packets and reconstructs TCP streams, providing a detailed view of network traffic. This tool can be particularly useful for network forensics, debugging network applications, and monitoring network activity. With its ability to extract and analyze data at the packet level, tcpflow facilitates the identification of anomalies, security threats, and performance issues within network communication. Its straightforward command-line interface and flexibility make it a valuable asset for network administrators, security analysts, and developers alike.

In this lab, you will install Ansible and use Ansible to allow the Manage node to extract an executable from PCAP in the Control Node

Required Resources

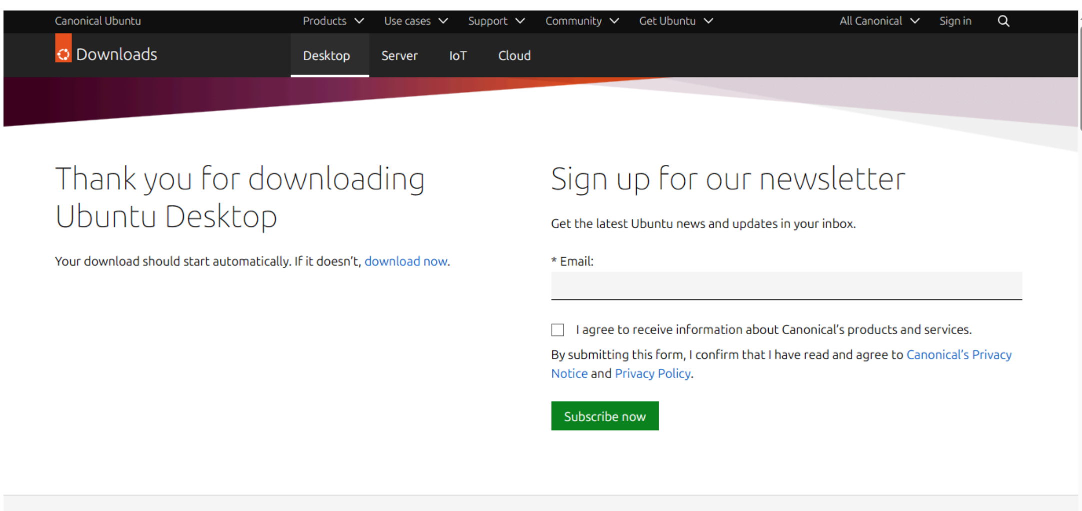
- 2 Virtual Machine with Internet Access (Manage Node and Control Node)
- Ansible
- Oracle VM VirtualBox

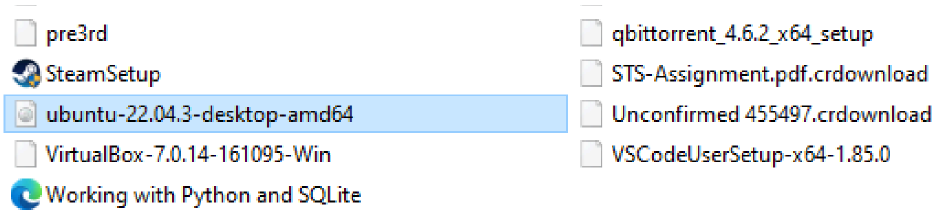
Instructions

Part 1: Create a Manage node and Control node (Choose Ubuntu or CentOS)

Step 1: Download and Install the Linux Distribution, Ubuntu

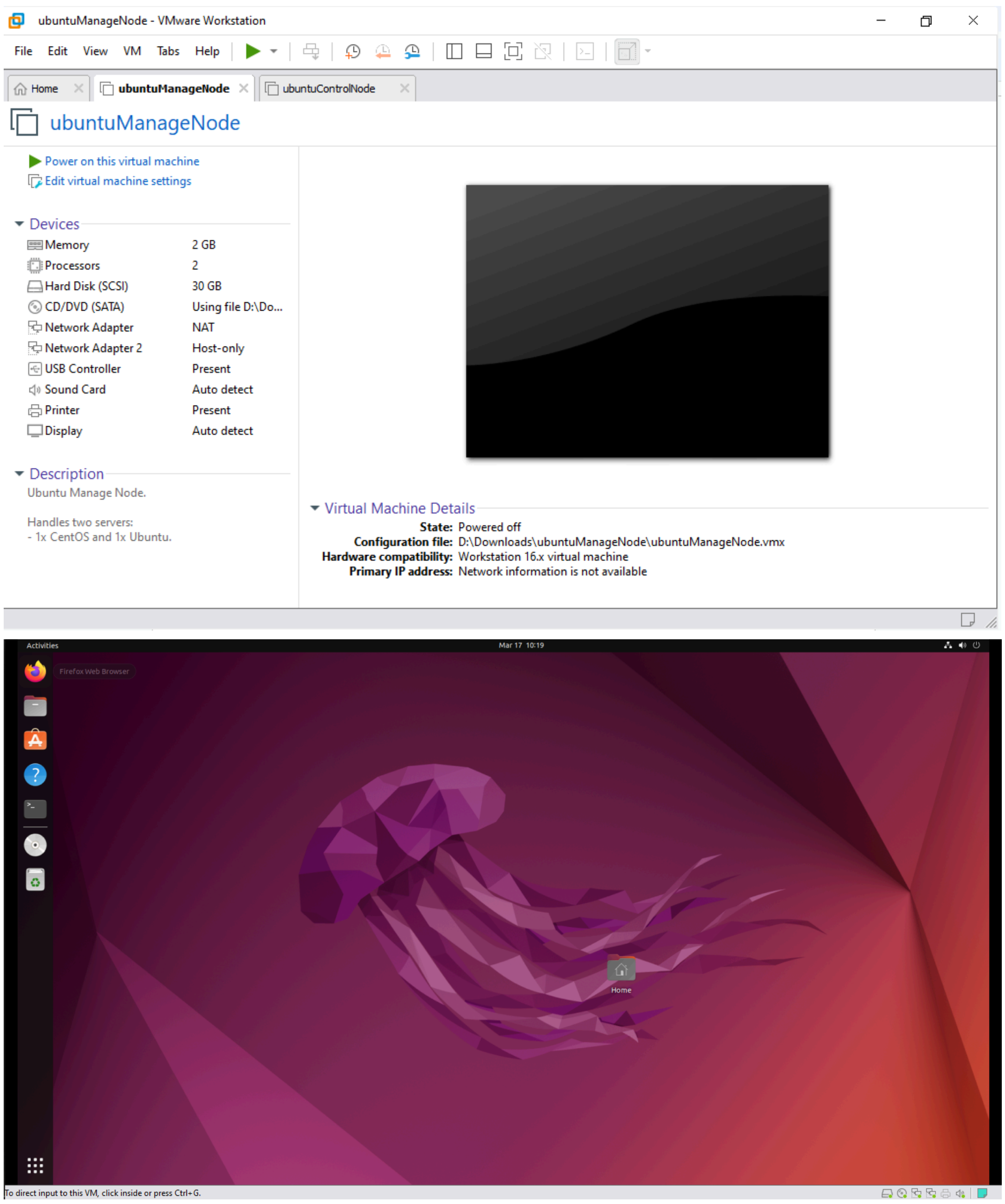
a. Download the latest stable version of Ubuntu from www.ubuntu.com/download . Choose the software version you need based on your PC's architecture and operating system.

Output	



b. Follow the on-screen instructions to install Ubuntu on VirtualBox. Change the settings to the specifications and recommended of the VirtualBox.

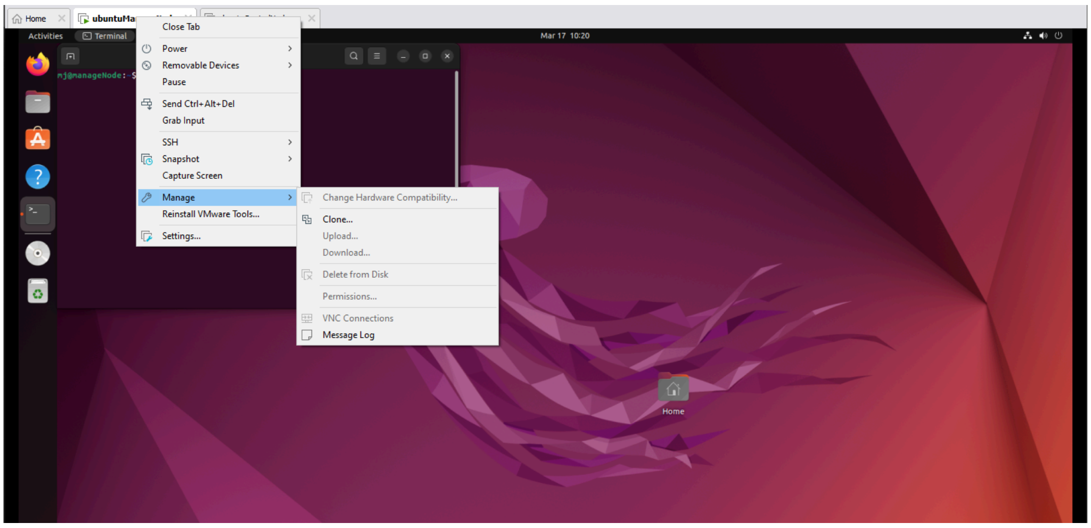
Output



Step 2: Clone the manageNode and name it "controlNode"

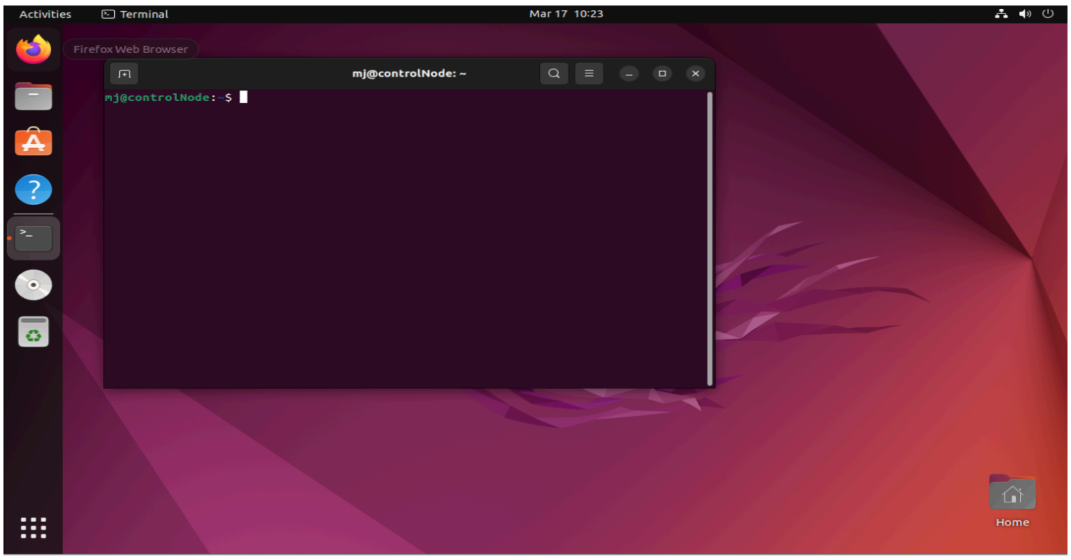
a. After installing and configuring the managed node, clone it to make the control node.

Output



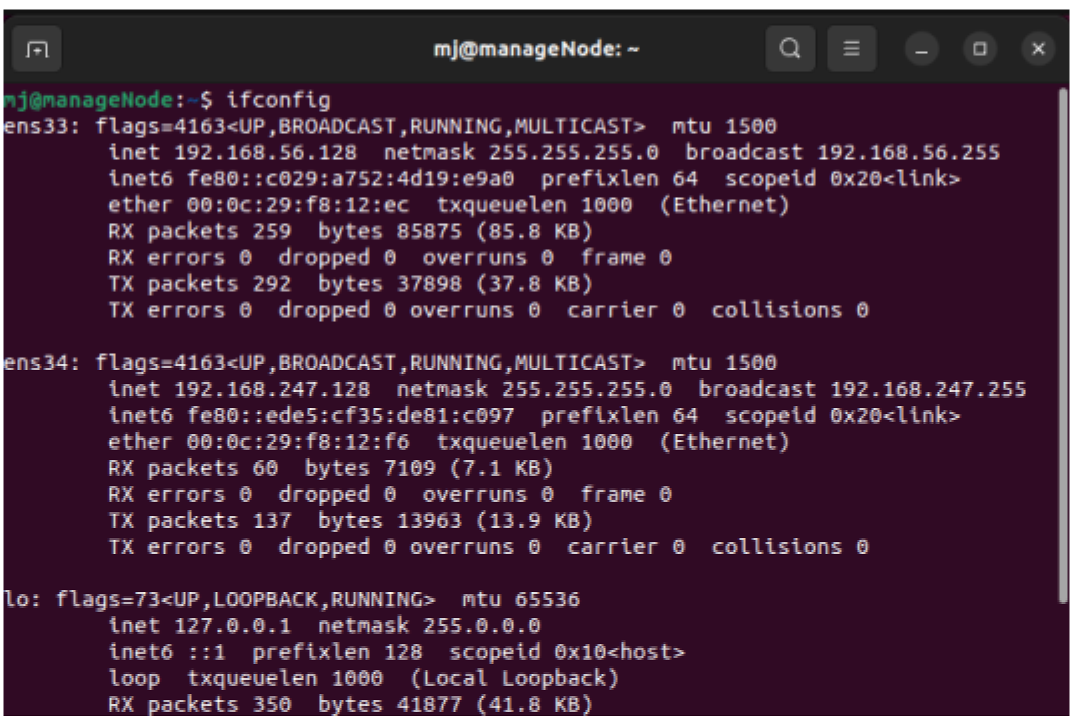
b. After cloning, configure the control node. Make sure that both servers have different IP addresses

Output



c. Issue the command, “ifconfig” to check if both servers have different IP addresses. If they have the same, configure them statically.

Output



```
j@controlNode:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.129  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::643f:b5a1:456b:f4f5  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:90:af:a5  txqueuelen 1000  (Ethernet)
        RX packets 58  bytes 13356 (13.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 135  bytes 14925 (14.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens34: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.247.129  netmask 255.255.255.0  broadcast 192.168.247.255
        inet6 fe80::503c:8910:d58b:35c5  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:90:af:af  txqueuelen 1000  (Ethernet)
        RX packets 5  bytes 582 (582.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 108  bytes 11716 (11.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Part 2: Implement network using SSH-key-based authentication

a. In the managed node, create an SSH Key Pair for User Authentication. The simplest way to generate a key pair is to run ssh-keygen without arguments. In this case, it will prompt for the file in which to store keys. First, the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory. However, in enterprise environments, the location is often different. The default key filename depends on the algorithm, in this case id_rsa when using the default RSA algorithm. It could also be, for example, id_dsa or id_ecdsa

Output

```
mj@manageNode:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mj/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mj/.ssh/id_rsa
Your public key has been saved in /home/mj/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:jLYHlguQKjy/qZw3umgHci3Pd5Lu0OqNy1H1G1E42/8 mj@manageNode
The key's randomart image is:
+---[RSA 3072]----+
|            ..   |
|   .       o.    |
|  o      . .+    |
|.. .   = ....    |
|oo .. B S o  .   |
|o.= .* +   o  .  |
|...=o +.. .    . |
|o..==*+..       E|
|o**+B=+o         |
+----[SHA256]-----+
mj@manageNode:~$
```

b. Issue the command ssh-keygen -t rsa -b 4096. The algorithm is selected using the -t option and key size using the -b option

Output

```
mj@manageNode:~$ ssh-keygen -t rsa -b 4096
```

c. When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the private key file. The passphrase should be cryptographically strong.

Output

```
mj@manageNode:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mj/.ssh/id_rsa):
/home/mj/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mj/.ssh/id_rsa
Your public key has been saved in /home/mj/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:vLEFUy0feF82Gp2gIGdvwnRW/AzbhtknFAqlwCkMTrY mj@manageNode
The key's randomart image is:
+---[RSA 4096]----+
|     +o o.*o*+o+..|
|    + .o OoBo==.+o|
|     E  .oo.B.+%o.|
|         . oo .=o*.|
|          S .   ...|
|           =       |
|          o        |
|                   |
|                   |
+----[SHA256]-----+
```

d. To use public key authentication, the public key must be copied to a server and installed in an authorized_keys file. This can be conveniently done using the ssh-copy-id tool.

Output

```
j@manageNode:~$ ssh-copy-id
sage: /usr/bin/ssh-copy-id [-h|-?|-f|-n|-s] [-i [identity_file]] [-p port]
lternative ssh_config file] [[-o <ssh -o options>] ...] [user@]hostname
        -f: force mode -- copy keys without trying to check if they are alre
nstalled
        -n: dry run    -- no keys are actually copied
        -s: use sftp   -- use sftp instead of executing remote-commands. Can
seful if the remote only allows sftp
        -h|-?: print this help
```

e. Issue the command similar to this: ssh-copy-id -i ~/.ssh/id_rsa user@host.

Output

```
mj@manageNode:~$ ssh-copy-id -i ~/.ssh/id_rsa mj@controlNode1
```

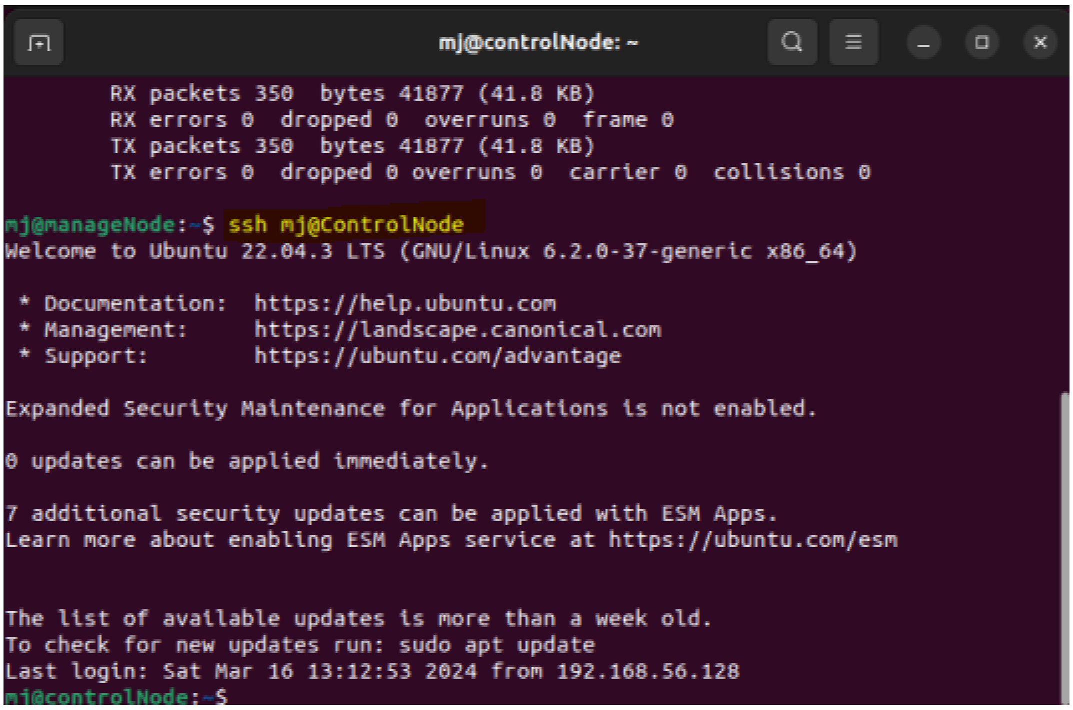
f. Once the public key has been configured on the server, the server will allow any connecting user that has the private key to log in. During the login process ,the client proves possession of the private key by digitally signing the key exchange.

Output

```
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/mj/.ssh/id_rsa.pub"

/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already in
stalled

/usr/bin/ssh-copy-id: ERROR: ssh: connect to host controlnode1 port 22: No route to host

mj@manageNode:~$
mj@manageNode:~$ ssh-copy-id -i ~/.ssh/id_rsa mj@controlNode1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/mj/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already in
stalled
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the
 new keys
mj@controlnode1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'mj@controlNode1'"
and check to make sure that only the key(s) you wanted were added.
```

g. On the local machine, verify that you can SSH with Server 1 and Server 2.

Output

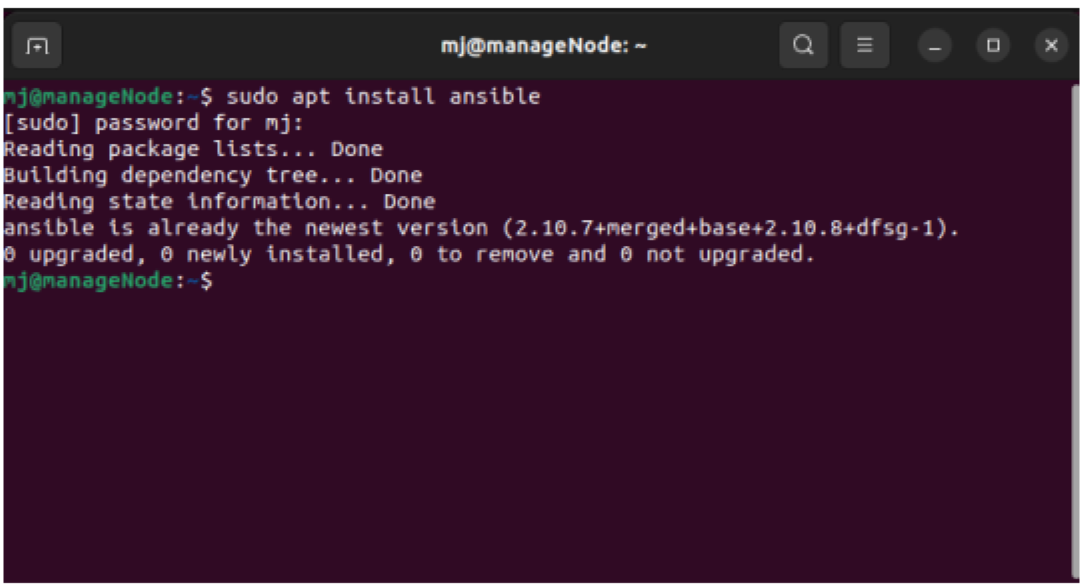


```
mj@controlNode: ~
        RX packets 350  bytes 41877 (41.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 350  bytes 41877 (41.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mj@manageNode:~$ ssh mj@ControlNode
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-37-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

7 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Sat Mar 16 13:12:53 2024 from 192.168.56.128
mj@controlNode:~$
```

Part 3: Create a playbook that allows the Manage node to extract an executable from PCAP in the Control Node

a. Install Ansible in the Manage Node.

Output



```
mj@manageNode: ~
mj@manageNode:~$ sudo apt install ansible
[sudo] password for mj:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ansible is already the newest version (2.10.7+merged+base+2.10.8+dfsg-1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
mj@manageNode:~$
```

b. Create a directory called “H8_YourName” and inside that directory, create a new playbook and name it pcap.yml.

Output

```
mj@manageNode: ~/H8_Ejercito
j@manageNode:~$ mkdir H8_Ejercito
j@manageNode:~$ ls
CPE232-Activity_13   Downloads        HOA4.2_SysAd3.pcapng   PyBirthdayWish
CPE232-Activity_14  'et TCP inste>'   key.txt                snap
CPE232-Activity_15   H7_Ejercito      Music                  Templates
Desktop              H8_Ejercito      Pictures               Videos
Documents            HBD              Public
j@manageNode:~$ cd H8_Ejercito/
j@manageNode:~/H8_Ejercito$ touch pcap.yml
j@manageNode:~/H8_Ejercito$ ls
cap.yml
j@manageNode:~/H8_Ejercito$
```

c. Make sure that the ansible configuration and inventory file is also in the directory. The ansible configuration file provides a centralized location to define settings and parameters that influence the behavior of the Ansible command-line tool and playbooks. The inventory file in Ansible serves as a crucial component for managing and organizing the hosts that Ansible will interact with during automation tasks. It acts as a directory of hosts, grouping them into categories such as development, production, or specific roles such as web servers or database servers.

Output

```
mj@manageNode:~/H8_Ejercito$ ls
ansible.cfg  inventory  pcap.yml
mj@manageNode:~/H8_Ejercito$ cat ansible.cfg
[defaults]

inventory = inventory
host_key_checking = False
deprecation_warnings = False
private_key_file = ~/.ssh/id_rsa
mj@manageNode:~/H8_Ejercito$ cat inventory
[controlNode]
192.168.56.129 ansible_user=mj
mj@manageNode:~/H8_Ejercito$
```

d. To verify the connection with the other server using Ansible, issue the command ansible <hostname_or_group> -m ping. Replace <hostname_or_group> with the specific hostname or group defined in your Ansible inventory file. This command utilizes the ping module, which sends a test message to the target host(s) and waits for a response.

Output

```
mj@manageNode: ~/H8_Ejercito
mj@manageNode:~/H8_Ejercito$ ansible controlNode -m ping
192.168.56.129 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
mj@manageNode:~/H8_Ejercito$
```

e. Edit the pcap.yml file so that Manage node can extract an executable from PCAP in the Control Node.

Output

```
    - name: Extract executable from PCAP file
      command: "tcpflow -r /home/mj/CPE243_HOA8/collectedPCAP/SkypeIRC.pcap -o /home/mj/CPE243_HOA8/extractedPCAP/"
      args:
        creates: /home/mj/CPE243_HOA8/extractedPCAP/executable.exe
```

Part 4: Show input (codes), process (successful run) and output (evidence that logs are collected)

a. This Ansible playbook consists of two plays designed to automate the extraction of an executable file from a PCAP (Packet Capture) file and its subsequent analysis. The first play targets all hosts to ensure the operating system is up to date and fetches the PCAP file from a remote server to the local directory. The second play, executed on the localhost, utilizes the tcpflow command to extract the executable from the fetched PCAP file. Subsequently, it reads and registers the content of the extracted executable file for debugging purposes. This playbook streamlines the process of analyzing network traffic data and extracting potentially harmful executables for further investigation.

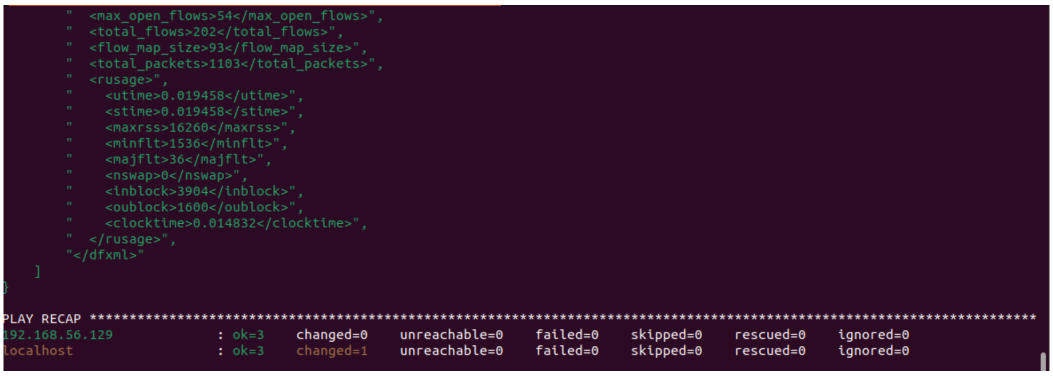
Input

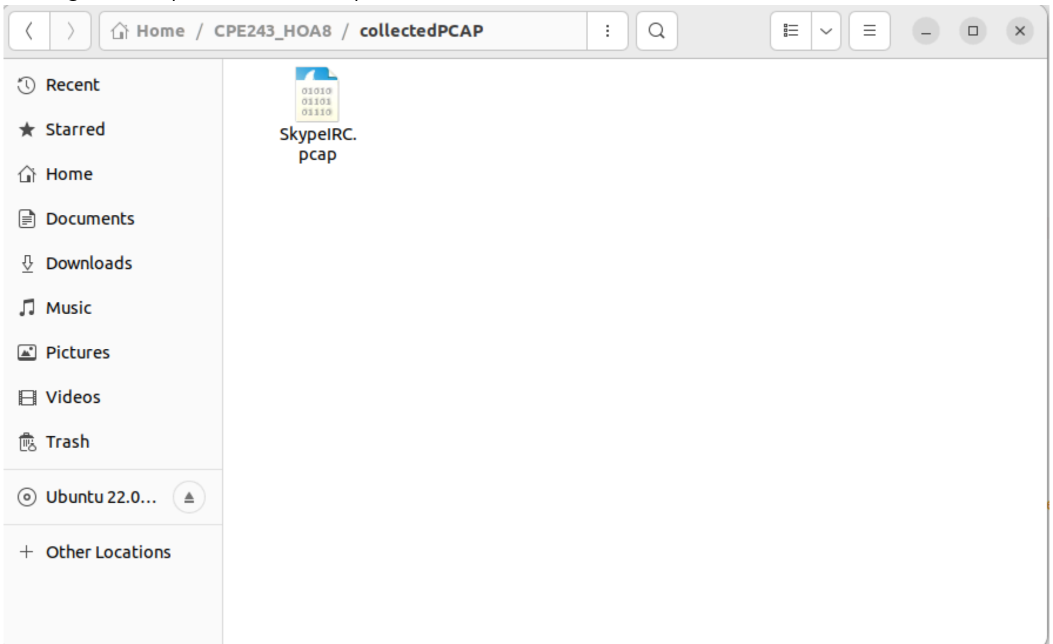
pcap.yml | inventory

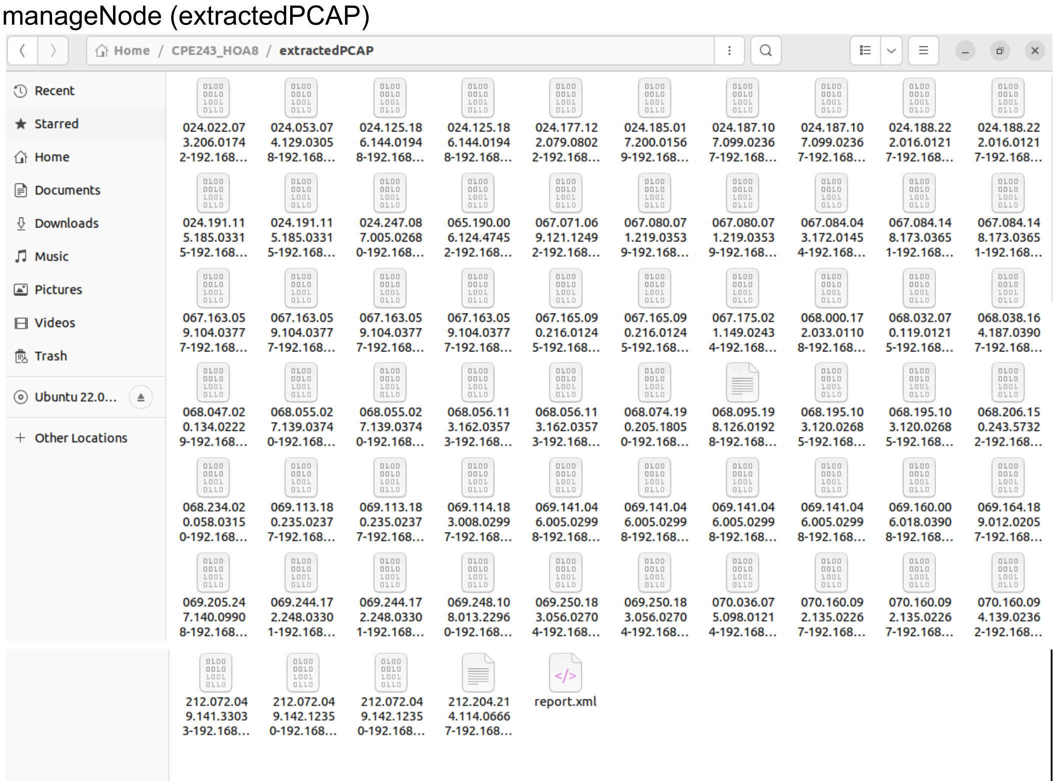
```
---
- hosts: all
  become: true
  become_user: root
  tasks:

    - name: Updating and upgrading the operating system
      apt:
        name: "*"
        state: latest
        update_cache: true

    - name: Collect PCAP file from server
      fetch:
        src: /home/mj/pcaps/SkypeIRC.pcap
        dest: /home/mj/CPE243_HOA8/collectedPCAP/
        flat: yes

- hosts: localhost
  connection: local
  gather_facts: false
  tasks:

    - name: Extract executable from PCAP file
      command: "tcpflow -r /home/mj/CPE243_HOA8/collectedPCAP/SkypeIRC.pcap -o /home/mj/CPE243_HOA8/extractedPCAP/"
      args:
        creates: /home/mj/CPE243_HOA8/extractedPCAP/executable.exe

    - name: Debug .exe file
      command: "cat /home/mj/CPE243_HOA8/extractedPCAP/report.xml"
      register: cat_output
      changed_when: false

    - debug:
        var: cat_output.stdout_lines
```

Process

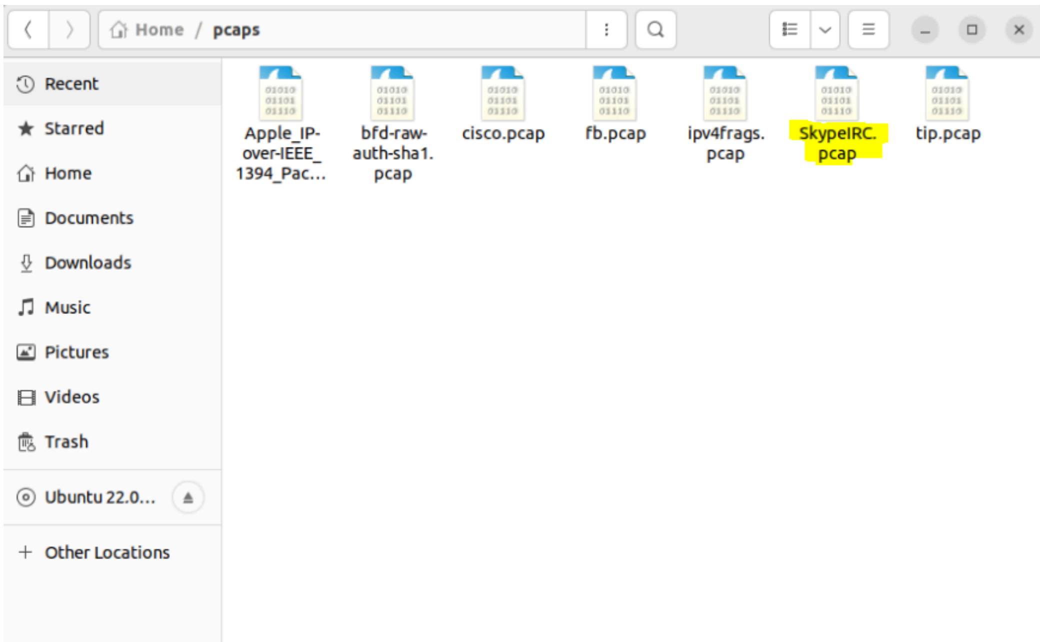
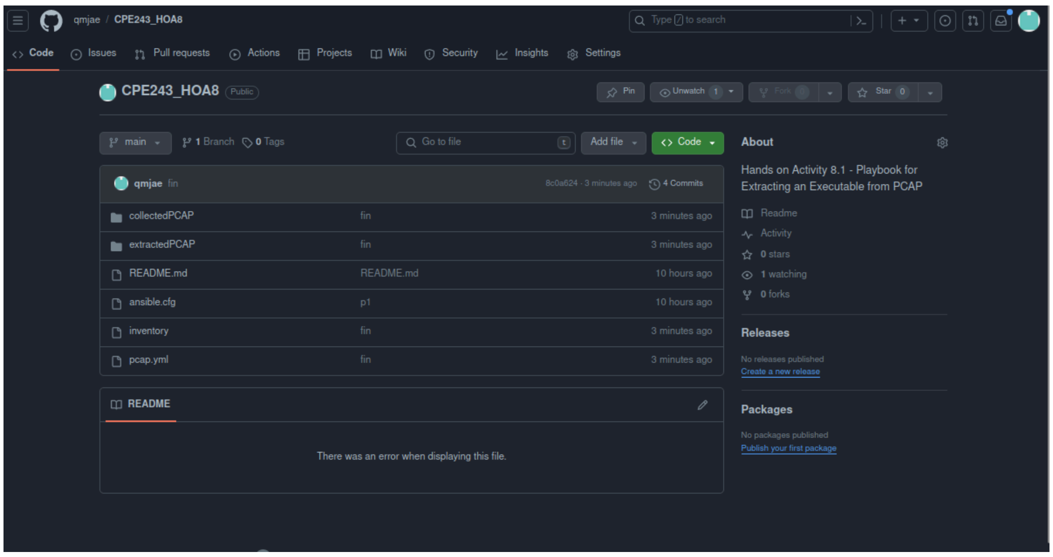
```
mj@manageNode:~/CPE243_HOA8$ ansible-playbook --ask-become-pass pcap.yml
BECOME password:

PLAY [all] ***************************************************************************************

TASK [Gathering Facts] ***************************************************************************
ok: [192.168.56.129]

TASK [Updating and upgrading the operating system] ***********************************************
ok: [192.168.56.129]

TASK [Collect PCAP file from server] *************************************************************
ok: [192.168.56.129]

PLAY [localhost] *********************************************************************************

TASK [Extract executable from PCAP file] *********************************************************
changed: [localhost]

TASK [Debug .exe file] ***************************************************************************
ok: [localhost]

TASK [debug] *************************************************************************************
ok: [localhost] => {
```

<table>
<tr><td></td><td>


</td></tr>
<tr><td>Output</td><td>
manageNode (collectedPCAP)


manageNode (extractedPCAP)


controlNode (PCAP)
</td></tr>
</table>

<table>
<tr><td></td><td>
</td></tr>
<tr><td>GitHub</td><td>
</td></tr>
</table>

Reflection and Conclusion

In this activity, I was able to create a managed node and clone it and named it control node, specifically Ubuntu as the distribution of Linux. As well as, I was able to implement a network using SSH-key-based authentication, and I was able to configure and access the control node via manage node remotely. In addition, I was able to create a playbook that allows the Manage node to extract an executable from PCAP in the Control Node. Lastly, I was able to document the process of this activity and I was able to show the input, process, and intended outcomes for this activity.