MINT Capstone Project

Analysis of Linux Distributions as a Portable Security Solutions Tool

Submitted by: Zainab Ijaz

University of Alberta

Table of Contents

1. Introduction 1
What is a Linux system? 1
Security in Linux 2
User and Group Separation 2
File System Security 3
AppArmor 4
Network Security 4
Cryptography 5
Pluggable Authentication Modules (PAM) 5
Audit Trial 7
Integrity Management 7
2. Linux Distributions 9
Tiny Core Linux 9
Installation 9
Boot Codes 9
USB and other External Storage Devices 9
Dependency Checking and Downloading 10
Modes of Operation 10
Porteus Linux 11
Features 12
Slitaz 15
Silver Blue 15
What is ostree? 15
Why Silverblue? 16
Fedora as an immutable OS 16
Management of application packages in Silverblue 16
3. Implementation 17
Slitaz 17
NMAP 17
WIRESHARK 17
SQLMAP 18

METASPOLIT FRAMEWORK 22
SET 24
Tiny Core 38
Nmap 39
Sqlmap 39
Wireshark 40
Metasploit Framework 41
SET 43
Porteus 50
SET 51
NMAP 59
SQLMAP 59
METASPLOIT FRAMEWORK 60
WIRESHARK 62
Silver Blue 72
NMAP 72
WIRESHARK 73
SET 74
METASPLOIT FRAMEWORK 76
SQLMAP 77
4. Analysis 79
Tools Testing 79
Tools Efficiency/Time comparison 81
Size Comparison 81
5. Conclusion: Determining Best Distribution 84
6. Appendix A 85
Installation Process of Kali Linux 85
Installation Process of Slitaz 105
Installation Process of Tiny Core Linux 122
Installation Process of Porteus Linux 137
Installation Process of SilverBlue Linux 145

Table of Figures

Figure 1- Installing NMAP in Slitaz 17
Figure 2- Installing Wireshark on Slitaz 17
Figure 3- Installing python3 on Slitaz 18
Figure 4- Installing git on Slitaz 19
Figure 5- git cloning on Slitaz 20
Figure 6- Setting ssl verify as false. 20
Figure 7- git cloning on Slitaz 21
Figure 8- Installing python2.7 on Slitaz 21
Figure 9- Starting sqlmap on Slitaz 22
Figure 10- Installing postgresql on Slitaz 23
Figure 11- Installing curl and downloading Metasploit on Slitaz 23
Figure 12- Changing the mode of Metasploit installer 24
Figure 13- Launching Metasploit installer 24
Figure 14- git cloning on Slitaz 25
Figure 15- Installing python3 on Slitaz 25
Figure 16- Installing pip3 in Slitaz 26
Figure 17- Checking the TLS version 27
Figure 18- Installing python3 on Slitaz 28
Figure 19- Checking pip version on Slitaz 28
Figure 20- git cloning on Slitaz 29
Figure 21- Turning of SSL verification for git. 29
Figure 22- git cloning on Slitaz 30
Figure 23- pip version error 30
Figure 24- Compiler error on Slitaz 31
Figure 25- Installing Slitaz toolchain on Slitaz 31
Figure 26- Installing Python3 development tools on Slitaz 32
Figure 27- Installing py3k-cython on Slitaz 32
Figure 28- Installing py3k-setuptools-scm on Slitaz 32
Figure 29- Installing gmp-dev on Slitaz 33
Figure 30- Compiler error on Slitaz 33
Figure 31- Installing openssl-dev on Slitaz 34
Figure 32- Compiler error on Slitaz 34
Figure 33- Installing cryptography=3.2.1 on Slitaz 35
Figure 34- Installing pyopenssl on Slitaz 35
Figure 35- Installing pycrypto on Slitaz 35
Figure 36- Successful build of pip 36
Figure 37- Directory Error on Slitaz 36
Figure 38- Making directory for setoolkit 37
Figure 39- Successful run of setoolkit 37
Figure 40- Successful start of SET tool in Slitaz 38
Figure 41- Installing VMware tools on Tinycore 38

Figure 42- Installing nmap on Tinycore 39
Figure 43- git cloning on Tinycore 39
Figure 44- Installing python on Tinycore 40
Figure 45- Starting sqmap on Tinycore 40
Figure 46- Installing wireshark on Tinycore 41
Figure 47- Downloading Metasploit on Tinycore 41
Figure 48- Changing the installer mode on Tinycore 42
Figure 49- Launching the Metasploit installer on Tinycore 42
Figure 50- Starting Metasploit on Tinycore 42
Figure 51- git cloning on Tinycore 43
Figure 52- Installing requirements.txt python package on Tinycore 43
Figure 53- - Environmental and pip version error on Tinycore 44
Figure 54- pip version upgrade in Tinycore 44
Figure 55- Compiler error on Tinycore 45
Figure 56- Installing compiletc on Tinycore 45
Figure 57- Python error on Tinycore 46
Figure 58- gmp error on Tinycore 46
Figure 59- Successful build of SET tool in Tinycore 47
Figure 60- Decoding Error in Tinycore 47
Figure 61- Locale in Tinycore 48
Figure 62- Making directory in Tinycore 48
Figure 63- Checking locale in Tinycore 49
Figure 64- Changing locale to UTF-8 in Tinycore 49
Figure 65- Locale in Tinycore 50
Figure 66- Starting SET tool in Tinycore 50
Figure 67- Installing git on Porteus 51
Figure 68- Installing python on Porteus 52
Figure 69- Installing pip on Porteus 52
Figure 70- Installing python on Porteus 52
Figure 71- Installing glibc in Porteus 53
Figure 72- Installing pip3 in Porteus 53
Figure 73- git cloning in Porteus 53
Figure 74- C compiler error in Porteus 54
Figure 75- Installing gcc in Porteus 54
Figure 76- C compiler error in Porteus 55
Figure 77- Test C program for dependencies 55
Figure 78- libisl missing package error in Porteus 55
Figure 79- Installing libisl in Porteus 56
Figure 80- Missing binutils library error in Porteus 56
Figure 81- Installing binutils in Porteus 56
Figure 82- C test program in Porteus 57
Figure 83- C compiler error in Porteus 57
Figure 84- Installing gcc-g++ in Porteus 57

Figure 85- Installing libffi in Porteus 58
Figure 86- Installing kernel-headers in Porteus 58
Figure 87- Successful build of SET tool in Porteus 59
Figure 88- Installing nmap in Porteus 59
Figure 89- git cloning in Porteus 60
Figure 90- Launching sqlmap in Porteus 60
Figure 91- Downloading Metasploit framework in Porteus 61
Figure 92- Changing the installer mode of Metasploit in Porteus 61
Figure 93- Starting Metasplot framework in Porteus 61
Figure 94- Running script for installing Wireshark in Porteus 62
Figure 95- Missing library make in Porteuss 62
Figure 96- Installing make package in Porteus 63
Figure 97- Missing libguile package in Porteus 63
Figure 98- Installing guile package in Porteus 63
Figure 99- Missing libgc package in Porteus 64
Figure 100- Installing gc package in Porteus 64
Figure 101- Missing glibc package in Porteus 65
Figure 102- Installing glibc package in Porteus 65
Figure 103- Missing cmake package in Porteus 66
Figure 104- Installing cmake package in Porteus 66
Figure 105- C compiler error in Porteus 67
Figure 106- Missing C-ares package in Porteus 68
Figure 107- Missing package libicui18n in Porteus 68
Figure 108- Missing package gpg in Porteus 69
Figure 109- Missing package zlib in Porteus 69
Figure 110- Missing package m4 in Porteus 70
Figure 111- Missing package gL in Porteus 70
Figure 112- Successful build of Wireshark in Porteus 71
Figure 113- Missing package libxcb in Porteus 71
Figure 114- Wireshark starting in Porteus 72
Figure 115- Installing nmap in SilverBlue 73
Figure 116- Installing Wireshark in Silverblue 73
Figure 117- Wireshark starting in Silverblue 74
Figure 118- git cloning in Silverblue 74
Figure 119- Missing package gmp and mpir in Silverblue 75
Figure 120- Missing package gmp and mpir in Silverblue 75
Figure 121- Missing python dev tools in Silverblue 75
Figure 122- SET tool starting in Silverblue 76
Figure 123- Downloading Metasploit in Silverblue 76
Figure 124- Metasploit starting in SilverBlue 77
Figure 125- Metasploit starting in SilverBlue 77
Figure 126- git cloning in Silverblue 78
Figure 127- Sqlmap starting in Silverblue 78

Figure 128- Tools complexity graph..... 80
Figure 129- Size comparison graph of Linux distros..... 82
Figure 130- Kali Linux download..... 85
Figure 131- Create a new virtual machine..... 86
Figure 132- Selecting the type of configuration. 86
Figure 133- Virtual Machine hardware compatibility 87
Figure 134- Selecting where to install from 87
Figure 135- Selecting the guest operating system. 88
Figure 136- Specifying virtual machine name and location 88
Figure 137- Selecting the number of cores and processors. 89
Figure 138- Specifying the memory of the virtual machine 90
Figure 139- Selecting the network type 90
Figure 140- Selecting the I/O controller type 91
Figure 141- Selecting the virtual disk type 91
Figure 142- Selecting the disk to be used 92
Figure 143- Specifying the disk capacity..... 93
Figure 144- Virtual Machine specifications 94
Figure 145- Inserting the Kali Linux iso file 95
Figure 146- Selecting graphic install 96
Figure 147- Selecting language as English 96
Figure 148- Selecting location as Canada..... 97
Figure 149- Configuring the keyboard 97
Figure 150- Installing Kali Linux in Virtual Machine..... 98
Figure 151- Setting the host name in Kali Linux..... 98
Figure 152- Setting the name of the user account..... 99
Figure 153- Setting a username 99
Figure 154- Setting up the password 100
Figure 155- Configuring the clock..... 100
Figure 156- Selecting the disk partition..... 101
Figure 157- Selecting "all files in one partition." 101
Figure 158- Finish partition of disk and write to file..... 102
Figure 159- Write partition changes to disk 102
Figure 160- Installing the base system of Kali Linux..... 103
Figure 161- Selecting the desktop environment 103
Figure 162- Installing the GRUB boot loader..... 104
Figure 163- Selecting where to intsall GRUB 104
Figure 164- Installation of Kali Linux complete 105
Figure 165- Slitaz official website download page..... 106
Figure 166- Sitaz downloads page 106
Figure 167- Slitaz downloads diectory 107
Figure 168- Slitaz downloads directory 107
Figure 169- Create a new virtual machine 108
Figure 170- Selecting the configuration type..... 108

Figure 171- Specifying the virtual machine hardware compatibility. 109
Figure 172- Selecting location to install the system from 110
Figure 173- Selecting the guest operating system 111
Figure 174- Specifying the name and location of the Virtual Machine..... 112
Figure 175- specifying the number of cores and procerssors 113
Figure 176- Specifying the size of the Virtual Machine..... 114
Figure 177- Select network type 115
Figure 178- Select I/O controller type 116
Figure 179- Select disk type 117
Figure 180- Selecting the disk to be used 118
Figure 181- Specifying the disk capacity..... 119
Figure 1820 Selecting the location to store file 120
Figure 183- Virtual Machine specifications 121
Figure 184- Adding the iso file for Slitaz 121
Figure 185- Slitaz Desktop page..... 122
Figure 186- Tinycore Linux official site download page 122
Figure 187- Tinycore Linux download releases 123
Figure 188- Create a new virtual machine..... 123
Figure 189- Selecting the Configuration type..... 124
Figure 190- Virtual machine hardware compatibility..... 125
Figure 191- Selecting where to install the system from 126
Figure 192- Selecting the guest operating system 127
Figure 193- Specifying the virtual machine name and location 128
Figure 194- specifying the number of processors and cores..... 129
Figure 195- Specifying the memory of virtual machine..... 130
Figure 196- Selecting the network type 131
Figure 197- Selecting I/O controller type 132
Figure 198- Selecting the disk type 133
Figure 199- specifying which disk to use 134
Figure 200- Specifying disk capacity 134
Figure 201- Specifying the disk file..... 135
Figure 202- Virtual machine settings..... 136
Figure 203- Adding the Tinycore downloaded iso file..... 136
Figure 204- Porteus Linux download page..... 137
Figure 205- Porteus Linux download releases..... 138
Figure 206- Create a new virtual machine..... 138
Figure 207- Selecting the configuration type..... 139
Figure 208- Selecting where to install the system from 140
Figure 209- Selecting the guest operating system 141
Figure 210- Specifying the name of the virtual machine and its location 142
Figure 211- specifying the disk capacity 143
Figure 212- Virtual machine settings..... 144
Figure 213- Specifying the memory 144

Figure 214- Specifying the number of processors and cores 145
Figure 215- Adding the iso file of the Porteus Linux 145
Figure 216- create a new virtual machine 146
Figure 217- Specifying the configuration type 146
Figure 218- Selecting the configuration type 146
Figure 219- Selecting the guest operating system 146
Figure 220- Specifying the virtual machine name and location 146
Figure 221- specifying the disk capacity 146
Figure 222- Virtual machine settings 146
Figure 223- Adding the Silverblue Linux downloaded iso file 146

1. Introduction

What is a Linux system?

A Linux is an operating system, just like, iOS, and Windows. An operating system in a computer system manages all the hardware resources related to the computer. In other words, operating system manages the communication between the software and hardware of a computer system and the software cannot function in the absence of an operating system. Along with operating system, Linux is also a kernel. When the term Linux is used, it usually refers to the Linux operating system as a whole, however, it can also refer only to the Linux kernel. The kernel is the heart of the Linux operating system.

Linux was created by Linus Torvalds when he was studying computer science at the university of Helsinki. To continue his studies, he purchased an IBM compatible PC which had the MS DOS operating system. He was not satisfied with the MS DOS operating system and wanted to use the UNIX like operating system which he used at the university. When he decided to get a copy for UNIX operating for his personal computer, he realized that it is very expensive. Driven by the preference to run a UNIX like operating system on his PC, he intended to create Linux. He worked with over 100 developers and as a result the first version of Linux was released in the market in 1994.

Linux has been in the market since 1990's and has reached a user-base that spreads the world i.e., Linux is everywhere. One of the most in demand platforms around the globe, Android, is powered by the Linux operating system. In addition to this, the world's most powerful super computers and stock exchange is run by Linux. It can also be used as a server application to host websites, run database software's and even act as a file server.

https://www.linux.com/what-is-linux/

A Linux system comprises of several different components which are the bootloader, kernel, init system, daemons, graphical server, desktop environment and applications. In describing the components, the bootloader is a software that manages the boot process of the system. It is a splash screen that comes up and goes away after some time in to boot the operating system.

The next vital piece of the Operating System (OS) is the Kernel, which is the heart of the OS. It manages the CPU, memory and the peripheral devices associated with the system. The init system,

it is a subsystem that starts up the user space. It runs the boot process of the system once the initial booting is consigned from the bootloader.

Daemons are the background services like printing and scheduling that start when a user logs in to the system. A graphical sever is a part of the system that manages the display of graphics on the screen, while the desktop environment is a component through which the users interact with the computer and it comes with a lot of built-in applications.

The desktop environment does not come with all the applications that are required. Just like Windows, a user can download and install different kinds of applications from a through an application management system which retrieves applications from centralized repositories.

Linux is open source which means that it is accessible to everyone on the internet. The users of internet around the globe can study and learn how the Linux system works, make copies of it, redistribute it, and run the program freely for any purpose. Furthermore, Linux has a lot of versions to meet the requirements of different kinds of users. These versions are called Linux distributions. The most common distributions used are Ubuntu, Debian, Kali Linux, etc.

Security in Linux

Linux is less prone to viruses and malwares as compared to other operating systems like windows and MacOS. It is because Linux has a lot of security features which are discussed below.

User and Group Separation

In a Linux system, user accounts are used to verify the identity of the user trying to login to the computer system. When the user enters its users name and the password, the system decides whether the user is permitted to log in to the system and which resources is the user allowed to access.

Groups are coherent constructs that are used to group specific user accounts for a purpose. For instance, if an organization has a group of system administrators, they can all be put in a separate group so that they have the required permissions and resources to operate effectively. In this way, the company can manage the access to restricted resources and control which users need them, and which should be denied.

A user can only login to the system if the account exits. The permission settings of each user help in ensuring the protection of sensitive information and damage by other users. There are three permission settings determined by the Linux system. They are:

```
R   Determines that the user can read the file
W   Determines that the user can write the file
X   Determines that the user can execute the file
-- Determines that no access is permitted to the user
```

There are also categories of the user in the Linux OS. They are:

```
Owner     The one who owns the application or file
Group     The group who owns the application or file
Everyone  All users that have access to the system
```

https://www.tenouk.com/linuxunixsecurityfeatures.html

File System Security

On a Linux system, everything is a file, and if it is not, it's a process. The files contain data like executable files, input and output to a program or text files. Although everything that is encountered on a Linux system is a file or process, there are some exceptions listed below.

- **Links**: a mechanism for making a file or directory recognizable in various parts of the file tree of the system
- **Named pipes**: works close to like sockets and set up a system for processes to communicate with each other, in the absence of network sockets
- **Special files**: the technique operated for the purpose of input and output. Majority of the special files are in /dev for instance CD-ROM and USB.
- **(Domain) sockets**: a particular file type which is identical to TCP/IP sockets, contributing to inter-process networking secured by the file system's access control.
- **Directories**: the files that are lists of other separate files.

On a Linux system, each file is possessed by a user and a group of users. In addition to this, there is another category of users who are not the owner and are not part of the group owner. For every category of user, the read, write and executable permissions can be allowed or denied.

Files can be listed in Linux using the `ls -l` command. It also displays the permissions to the files. The first nine characters indicate the permission to the different category of users. The first three are for the owner, the next three for the group users and the last three for all the users. For example,

`-rw-rw-r--`, here the owner of the file and the users of that group can read and write the file, but they cannot execute it. The other users can read the file only; they cannot write or execute it. For changing the permission of a file to a specific user, the `chmod` command is used.

This scheme is implemented very rigorously, and it allows a high level of security even in the absence of a network security. It controls the access of files to users and protects the sensitive data such as configuration files and home directories.

AppArmor

Application Armor is a kernel-level MAC scheme that limits the applications to access the various classes of systems resources. For instance, App Armor can let the word processor to read and write files to the local hard drive but restrict it to access the internet for sending messages. This feature has default settings for a standard computer which helps in disallowing many of the malware attempts.

https://www.comparitech.com/blog/information-security/linux-security-guide/

When the bad guys make a malware, their target is to allow people to deploy it in their computer systems. They do this by sticking the malware in an entirely different kind of file like a subtitle file. In his case, App Armor can be configured for denying these types of files any internet capabilities because a subtitle file would have no requirement for these kinds of functions.

SELinux is another MAC scheme that came before the AppArmor. It is complicated to configure as well as it requires a specific file system with labels whereas AppArmor does not care about the file system.

Network Security

Linux comprises of a diverse and efficient networking stack which supports many attributes and protocols. It can be used as an endpoint on a computer network as well as a router which passes network traffic between two interfaces according to the networking procedures.

https://www.linux.com/training-tutorials/overview-linux-kernel-security-features/

Netfilter is a framework provided by the Linux kernel which attaches packets that pass into, through and from the system. The Linux kernel modules might hook into this in to examine the packets and make security decisions about them. One of the Linux kernel modules is the iptables which is used to implement an IPV4 firewall scheme. The rules for the ipv4 packets are specified

inside the kernel and each packet should pass these rules to pass through the entire networking stack. Other modules include NAT and stateful packet inspection.

Ebtables is another feature which provides filtering of the packets at the link layer and is used to perform access control for the Linux bridges. In the same way, arptables provides the filtering of ARP packets. In addition to this, the networking stack also has IPsec, which ensures the confidentiality, validity and uprightness of the IP networking. VPN can be implemented using it and it also provides the point-to-point security.

Cryptography

The kernel subsystem provides a cryptographic API for use. This API provides support for a variety of cryptographic algorithms and operating modes, which includes the hash functions and deployed ciphers. It provides restricted support to the asymmetric cryptography. There are concurrent and non-concurrent interfaces. The non-concurrent is used for supporting cryptographic hardware that unloads processing from the CPUs.

To manage the cryptographic keys within the kernel, a key management subsystem is provided. The kernel users of cryptographic API comprise of the kernel module signature verification, disk encryption schemes consisting of **`ecryptfs`** and **`dm-crypt`**, as well as IPsec code.

Pluggable Authentication Modules (PAM)

PAM is used to execute different types of tasks including authentication, modification and authorization. It gives the system administrator the authorization to separate the particulars of authentication procedures from the programs themselves. In this way, it lets the policy to not only be comprehensive, in other words it interprets that the applications do not require to be altered for the purpose of updating the policy.

For example, PAM can be used to control login attempts to a GUI or shell interface so that only successful authentication and approved situations are permitted. PAM is a functional system for administrators due to several reasons:

- It provides notable pliability and authority over authentication for the application developers and system administrators.
- It provides a usual authentication strategy which can be used with a vast variety of applications and programs.

- It provides only one library which is fully documented, and it allows the application developers to code programs without having the need to make their individual authentication strategies.

https://medium.com/information-and-technology/wtf-is-pam-99a16c80ac57

The PAM configuration files are in the `/etc/pam.d/` directory for each application that is PAM-aware. Every PAM-aware service has a file in the `/etc/pam.d/` directory. Each file in this directory has the similar name as the application to which it commands access. The PAM-aware application is accountable for interpreting its service name and installing its PAM configuration file in the `/etc/pam.d/` directory. For instance, the login program interprets its service name as login and then installs the `/etc/pam.d/`login PAM configuration file.

Linux-PAM splits the functions of authentication into four individual management modules:

- session modules interpret activities that are carried out at the start and end of each session. It assigns the resources that the users might require, like mounting their home directory or allowing usage limit. A session starts when the user is authenticated by the system.
- password modules are accountable for updating user passwords and are usually integrated to other modules enlisted in the authentication. They are also used to impose strong passwords to users.
- authentication modules are used to validate the identity of the user, for instance by asking for and examining a password or any other credential. Also, they can pass the authentication particulars on to other systems such as a keyring.
- account modules inspect that the stated account is a proper authentication pick out considering the ongoing situations. It comprises of situations such as account expiration, and the account user has approach to requested service.

There are some control arguments which are a significant part of creating the PAM secure policy.

- include — it brings in each one of the lines in configuration file that are equivalent to the given parameters and adds them as arguments to module.
- requisite — if it succeeds PAM goes on to further rules and if it fails then PAM comes to a stop and generates a failure message.
- optional — the outcome is disregarded, and it only becomes mandatory for successful authentication as soon as no other modules reference the operation.
- required — if it works out then move on to the next line and if it fails then moves on to further rules however it will consistently return an unsuccessful authentication anyhow of result. It is practical because it will not specify what lead to the failure, therefore a hacker will not know if they came by some part of the authentication correct.

- sufficient — if it passes on then authentication is successful, and PAM does not require to interpret further rules and if it fails then it keeps on going.

Audit Trial

The Linux audit system is a feature in Linux that records the system activities to assist incident analysis. This system is capable of monitoring three things.

- System calls: It observes which system calls were called including the contextual details such as arguments passed to it, user data, and more.
- File access: It is another method to observe file access activity instead of explicitly monitoring open system call and related calls.
- Choose pre-configured auditable events in the kernel.

The primary value proposition of the Linux Auditing System is to encourage investigations in response to an incident, particularly historical investigations. It is possible to inspect all files and device calls, including failed logins, authentications, failed syscalls, abnormal terminations, programs executed, and more. Most routine device operation falls within that reach, so a quantity of data would not be too limited for your problem.

https://capsule8.com/blog/auditd-what-is-the-linux-auditing-system/

Integrity Management

To protect the integrity of files on the device, the integrity management subsystem of the kernel may be used. The Integrity Measurement Architecture (IMA) component uses cryptographic hashes to conduct runtime integrity measurements of files, comparing them with a list of valid hashes. By means of an aggregate hash stored in the TPM, the list itself can be checked. Measurements carried out by IMA may be reported through the audit subsystem, and may also be used for remote attestation, where their accuracy is checked by an external device.

IMA can also be used by the Assessment extension for local integrity enforcement. The valid calculated file hashes are stored with the files as extended attributes and then checked when accessed. These extended attributes are defended by the Extended Verification Module (EVM) module against offline attack, preferably in conjunction with the TPM. If a file has been changed, the IMA can be configured to refuse access to the file through a policy. By verifying RSA-signed measurement hashes, the Digital Signature extension enables IMA to verify the validity of files in addition to integrity.

The dm-verity module is a simplified approach to integrity management. It is intended to be used as part of a validated boot process, where a system is brought online by an appropriately approved caller, i.e., a trusted partition containing kernel modules to be later loaded. Block by block, the integrity of such modules can be checked transparently when they are read from the disk.

2. Linux Distributions

Tiny Core Linux

The Tiny Core loads itself from storage into RAM, it then mounts applications on storage or installs applications to the RAM from the storage. Loading or installing an extension is said to be independent of the technique used (mount vs. copy to RAM). Tiny Core varies because it does not allow users to perform a conventional, hard-drive operating system installation.

Installation of a hard drive is possible; however Tiny Core is designed to run from a RAM copy created at boot time. This, in addition to being fast, protects files in the system from modifications and maintains a pristine system on every single reset. Renewability and stability, easy, and simple are principal priorities of Tiny Core. If this sounds equivalent to what many live CDs do, the approaches are similar and shared indeed.

Installation

The standard installation method for Tiny Core is frugal. It is not a typical installation of a hard drive, which we call "scattering" mode since all of the system's files are scattered around the disk. You basically have the system in two directories, with frugal, Vmlinuz and core.gz, whose position is specified by the boot loader.

Boot Codes

Depending on how you install the Tiny Core (GRUB, LILO, CD, USB stick...), users have the option of using boot codes on each restart (CD, etc) or save those codes in a boot configuration file (LILO, GRUB, etc).

Boot codes (boot arguments) impact the operation of Tiny Core by characterizing options at boot-time. There are plenty of boot codes. The boot code lists can be used to browse all the available options by pressing F2, F3 or F4 at the CD boot prompt.

The base of the boot code is significant. To simulate the default mode and bypass the program extension installing, the base can be used. It is a valuable tool for troubleshooting, creating extensions, updating, and just finding out how fast Tiny Core can boot on the hardware.

USB and other External Storage Devices

You can instruct Tiny Core to scan for data on external devices on boot time, such as, A USB pen drive, compact flash, or other portable media. This does not need to be the boot media; it is

common, for instance, to store user data on a hard disk when booting from a cd or USB file. Hardware often does not wake up quickly enough for Tiny Core's boot sequence. If the hardware does not wake up in time, Tiny Core moves on and finishes booting without that data.

If you store information on external/slow media, it might be mandatory to use the waitusb=5 boot code or similar. This pauses the boot method for 5 seconds, allowing the slow devices to register with system bus.

Dependency Checking and Downloading

Tiny Core makes it as easy as possible to get applications. The Applications Tool offers application information from individual .info files. it is the important reading content when choosing applications. Always Read .info files, then read them again before updating to capture changes and complaints. Dependencies are the sections (other programs, libraries) that are necessary for an application. In short, the Tiny Core Applications tool will take care of downloading and reviewing dependencies for the user.

Modes of Operation

The modes of operation mix up the way Tiny Core loads, mounts, and installs at boot time. Tiny Core's got three main modes.

• Default mode: cloud/internet mode

• Mounting mode: TCZ/Install

• Copy mode: TCZ/install and copy2fs.flg/lst

The Default Mode: Cloud/Internet

By default, Tiny Core Linux runs as a cloud/internet client. In the default mode Tiny Core is fully booting into RAM. Users can run the Applications Tool to browse the repository and import applications. Application Extensions (downloaded software) last only for existing session. Tiny Core uses as much RAM as possible.

Since Cloud/Internet Mode is running out of RAM, it's running fast. Cloud/Internet Mode is a nomadic and fast-start mode. Application extensions are lost on reboot, but only the system files must be stored again. If you like applications to be stored locally and set up on every reboot, consider the modes of mounting and copying.

Mount Mode

This is the most used and suggested form of using tiny core.

Applications are stored locally in a directory called tce on Persistent store – for example supported disk partition (ext2, ext3, ext4). Applications are optionally installed on reboot. Mounting Apps saves RAM for other purposes.

Unless the `tce=xdyz` boot code is defined, Tiny Core checks for all drives on the computer and use the first/tce directory that it finds for storing and loading extensions. Tiny Core uses the Applications tool to place application extensions to the tce/directory and flag it as "OnBoot" (mount at boot) Or "On Demand"

Copy Mode

Copy is a modification of the mount mode. Selected extensions of the application are copied to the RAM instead of mounting. Applications can be loaded with RAM in bulk (copy2fs.flg), selectively loaded or installed to the RAM (copy2fs.lst). The Applications program tracks installation and loading options (bulk copy, selective) Copy, etc. Boot times are longer, as copying to the RAM takes longer time than mounting, but the pace of runtime, particularly the first start, is a lot quicker.

Copy mode instantly extends the time of boot to obtain some of the RAM- run default mode speed and the persistence of a pure mount mode. It is vital to know, in copy mode, that extensions can be either mounted or copied to the RAM. The Applications software makes this capability possible by keeping track of user selections. It should be noted that the use of a bulk selection, that is, the loading of all extensions to the RAM, allows the storage to be unmounted, and system to prevent power loss corruption.

Porteus Linux

Porteus is a complete Linux operating system intended for USB flash drive, CD, hard drive, and other bootable storage media. It is extremely fast and tiny that enables the user to start and get connected when most other operating systems are still spitting dust. Porteus comes in both 32 and 64 bits and strives to remain on the cutting edge. It also supports a variety of different languages, and the user forum has language parts.

The persistent memory aspect of Porteus Linux, not like other standard live distro sessions, enables the user to bring a full fledge Linux desktop with all the files and specific system settings in the pocket to run on any machine.

Porteus can boot from a USB drive or other bootable external media in less than 15 seconds. The boot time will take longer if you install too many software modules. If you boot from a DVD, the Linux components loading time will take a minute longer than the USB drive. Of course, boot times will eventually depend on your hardware.

No matter how new or old your machine is, Porteus is lightning fast, since it is designed for speed and keeps up with what it needs in the system's RAM to minimize reading time.

Porteus loads in guest mode as an added security measure. The user can bypass this with a cheat code to function as an admin. Also, user can also enter an admin password when accessing privileged areas such as setup and PPM settings. Only think "root user" and type "toor" when prompted—and yes, user can permanently alter it to something else by editing the config file.

Features

IT IS FAST

The ability to load Porteus into RAM results in a ridiculously fast machine which has all the features of a full functioning Slackware installation and double the speed. When Porteus is loaded from flash device or from the hard drive locally, it is still fast—you are ready to go in just 25 seconds after the power button is pressed.

IT IS PORTABLE

Porteus is stored in a compressed format consisting of XZM files that are decompressed very easily. While stored, it weighs below 300Mb, making it a lightweight competitor with a cat's pace. This is achieved by stripping down the entire Slackware installation to the minimum level, which is quite a task. Linux live and boot scripts have been revamped by Fanthom to speed up shutdown time and boot time.

IT IS MODULAR

Another good feature of Porteus is its modular architecture. Unlike other Linux distributions where one has a package manager that links to the internet and installs a package (program), Porteus uses modules. These are pre-compiled packages that you activate and deactivate.

The typical 'setup' of a program is now obsolete as a simple double click on a module, which allows it to be installed and injected into a ready-to-use file system. This occurs in a fraction of a second, and the program is ready to use. Double click again and the module will be disabled and removed from the directory structure. This means that you only use the software when you need it, and the system does not get weighed down with thousands of files that are rarely used. Modules can be downloaded and stored locally for activation as needed.

DIFFERENT FAMILY LINE

Porteus is based on Linux Slackware. It has got a sordid lineage. It started with KDE 3 as the default desktop; it was a community remix of Slax, a now abandoned live CD based on Slackware. Later, the lighter-weight desktops came along as alternative settings.

STANDARD FARE

Porteus has a smart collection of applications designed to satisfy typical users. Some of the applications are different in the other desktop choices, but a mixture of KDE and XFCE/LXDE applications is available and works fine in all desktop environments.

For e.g., Porteus comes with the Firefox Web browser plus about half a dozen regular Linux Internet and networking resources. Graphics applications include the Image Viewer, Paint, Okular, Gwenview, and Screen Capture software. Porteus also has an impressive variety of multimedia applications, including Audio CD Ripper, ISO File Master, Audio Player, Sound Mixer, GNOME and SM Media Players, Pburn, and a webcam software.

POWER PPM

The Porteus Package Manager is a heavy-duty installer that functions as a cross-distro package converter. It has components for accessing repositories for its own applications along with Slackware, Salix, SlackBuild, Alien and Debian.

The PPM resolves dependencies through repositories. It produces a kit module instead of a traditional installation. You can also use PPM to enable only the modules you need for a specific

session and to deactivate them if you do not need them. To uninstall a software package, the module can be deleted by using the file manager.

STARTING IT

The first screen you see when Porteus boots is a list of choices. You choose the graphics mode, which is either KDE, XFCE or LXDE. The third choice to load is the Always Fresh mode. This choice will start Porteus with its original values and will not hold any modifications you make for the next session.

Conversely, you can drag down to the Copy to RAM mode. This method places Porteus entirely in RAM, provided you have at least 768 MB of RAM. It takes a little longer to get to the desktop with this option, but Porteus is running on steroids this way.

The text only mode is another loading option. This just runs Porteus from the command prompt. Porteus can also be loaded with a PXE server feature. This helps you to boot Porteus over a network on other computers.

BOOTING CHEATS

If the Porteus DVD or bootable USB is left on the device, the user can bypass the Porteus boot process without removing it. The user can only pick the boot option from the first hard drive. This will boot the resident OS of the machine installed on the hard drive.

It is quick to do modifications on the fly when Porteus begins by entering cheat codes from the splash screen. After selecting the boot option mentioned above, tap the Tab key instead of the Enter key to position the cursor on the command line.

Then user can enter the desired cheat code to set the parameters for the session. Press the Enter key to complete the loading process when done. The user can enter as many cheat codes as desired. Only separate each of them with a gap. Cheat codes bypass the default settings in the config file. This file has scripts that match the cheat codes that the user enters.

HAVE IT YOUR WAY

Knoppix and Puppy Linux distributions use a similar cheat code scheme to load configuration options while booting. However, the Porteus cheat codes are more like regular phrases, so they are easier to use.

The user can boot into the Porteus sessions from a USB drive which holds not only the Porteus OS but all the software modules that the user have installed and all the OS settings. In this way, the user has a consistent user experience on every device used to run Porteus Linux.

https://linuxinsider.com/story/with-porteus-in-your-pocket-youre-good-to-go-78550.html

Slitaz

Slitaz is an open-source Linux operating system which runs entirely on a RAM from removeable media such as USB or CD-ROM. It is light in size, good in speed and can also be fully installed on a hard drive. Slitaz is available in the form of a LiveCD which can be burned on a CD-ROM and then boot from there. While using the system, the user can eject the LiveCD and use the CD drive for other purposes. The Slitaz LiveCD provides a full featured and a graphical distribution where the user can keep data. The Slitaz system is provided with security updates.

Silver Blue

Fedora Silverblue is known to be an immutable desktop operating system. It is intended to be extremely consistent and reliable. It also aims to be an excellent platform for developers and users of container-focused workflows.

https://docs.fedoraproject.org/en-US/fedora-silverblue/

Silverblue is known as the code name for the new generation of the OS, formerly referred to the “Atomic Workstation”. The fedora silverblue OS is delivered in images that are generated using the rpm-ostree project. The main advantages of the system are security, speed, atomic updates and immutability.

What is ostree?

OSTree also known as libostree is a project which combines a “git-like” model to commit and download bootable filesystem trees, along with a layer to deploy and manage the bootloader configuration. OSTree is used by rpm-ostree, a hybrid package/image-based framework used by Silverblue. It atomically recreates a base operating system and enables the user to “layer” the conventional RPM on top of base operating system if required.

https://fedoramagazine.org/what-is-silverblue/

Why Silverblue?

Silverblue provides a work environment where it allows the user to pay attention on the work instead of the operating system. It is a robust system because its updates are atomic. The user only requires restarting into a new image. If the user finds anything wrong with the current running image, there is an option of rebooting or rolling back into the former image if it is available. If it is not available, the user can download any other image using the ostree command, that was generated in the past.

Using fedora, the users can switch between the fedora releases. The user can easily try the old branches and then return to the current release of fedora.

Fedora as an immutable OS

In fedora, the root file system is read-only by default that increases the security against malicious attacks as well as the accidental damage to the system. The root file system can be upgraded or modified by using the rpm ostree command.

Robustness is another benefit. It is almost impossible for a normal user to bring the OS to a state when it does not boot or when it does not function properly after mistakenly or unintentionally deleting any device library.

Management of application packages in Silverblue

Flatpak is recommended for graphical user interface systems where the program is accessible as a flatpak. The users can pick from Flatpaks also from Fedora and built from Fedora-owned infrastructure and Fedora packages which currently has a wider offering. Users can easily install it via GNOME Apps, that already supports Fedora Silverblue.

3. Implementation

To understand the efficiency of security tools in the Linux distributions, the first step is to install those distributions. The installation process will be documented in this paper (Appendix A). I will also install Kali Linux so that I am able to test those tools in comparison with the Kali Linux. In this way, the best distributions can be selected.

Slitaz

Slitaz does not use apt-get install, instead it uses the tazpkg get-install.

http://doc.slitaz.org/en:handbook:packages

NMAP

Nmap is already available on Slitaz and it can be installed from the tazpkg package manager.

```
root@slitaz:~# tazpkg get-install nmap

Recharging repository "Main"
==============================================================================
Checking...                                                          [ Done ]
Database timestamp: 01/12/2021 03:12
Creating backup of the last packages list...                         [ Done ]
Getting "bundle.tar.lzma"...                                         [ Done ]
Getting "files-list.lzma"...                                         [ Done ]
==============================================================================
Last database is ready to use.

Note that next time you recharge the list, a list of differences will be
displayed to show new and upgradeable packages.

nmap-7.80.tazpkg     100% |*********************************| 1650k  0:00:00 ETA
Tracking dependencies for package "nmap"
==============================================================================
Missing package "libpcap"
==============================================================================
1 missing package to install.
```

Figure 1- Installing NMAP in Slitaz

WIRESHARK

Wireshark is available on Slitaz and it can be installed from the tazpkg package manager.

```
root@slitaz:~# tazpkg get-install wireshark
wireshark-2.6.1.tazp  46% |**************                  | 5577k  0:00:25 ETA
```

Figure 2- Installing Wireshark on Slitaz

SQLMAP

Sqlmap is not available on Slitaz Linux, so to install it the first step is to install python3.

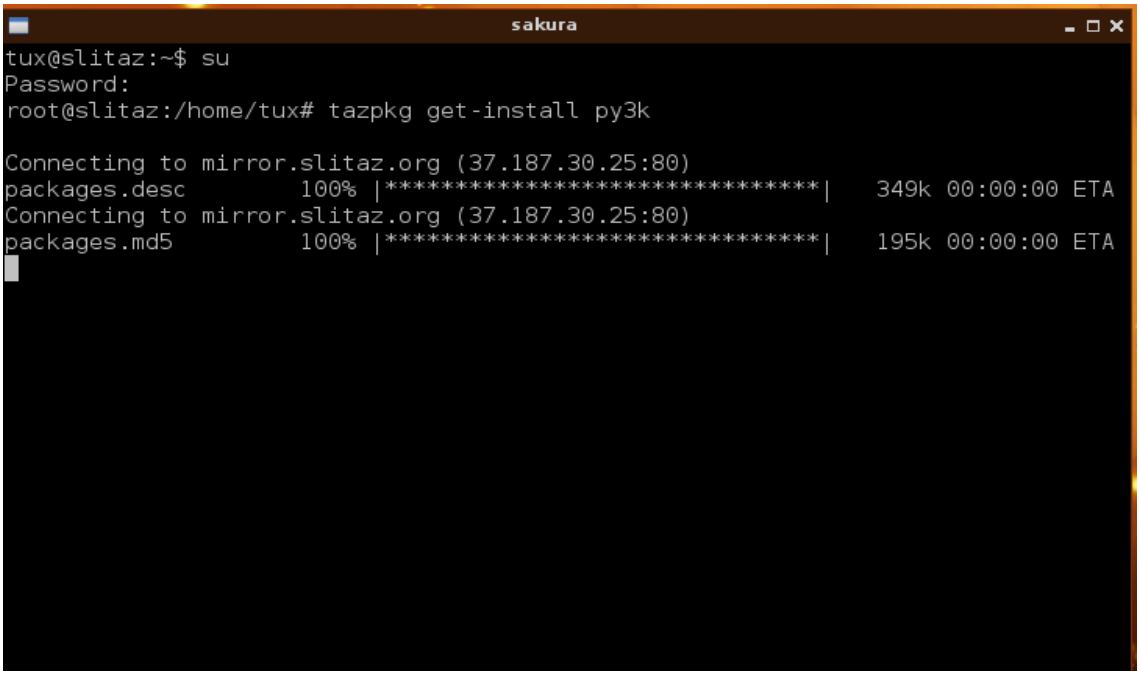


Figure 3- Installing python3 on Slitaz

After installing python3, the next step is to install git.

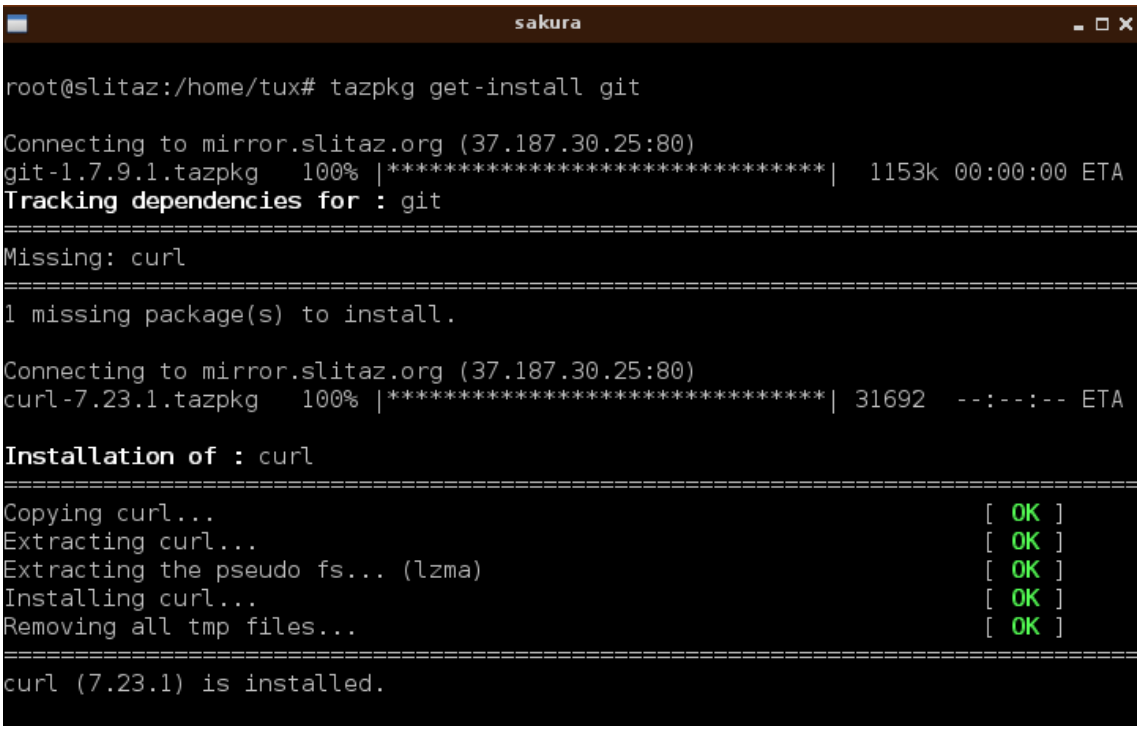


Figure 4- Installing git on Slitaz

After installing git and python3, now it's time to start git cloning to install sqlmap

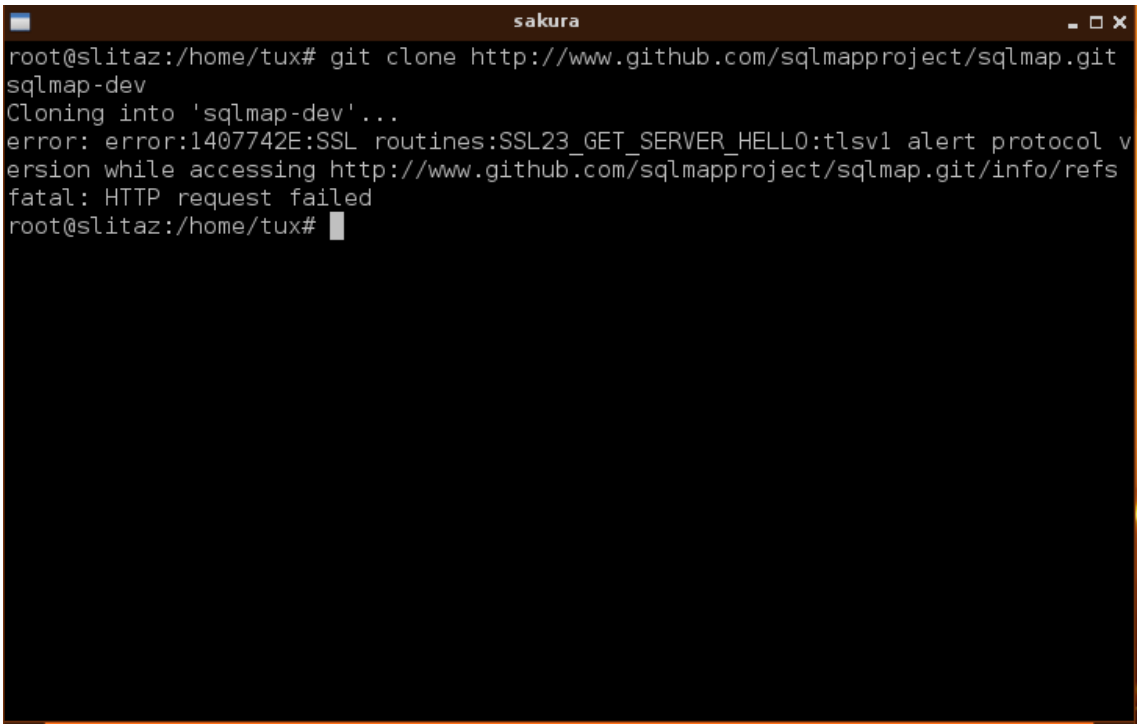


Figure 5- git cloning on Slitaz

Here the git cloning failed due to ssl verification, so I disabled it.

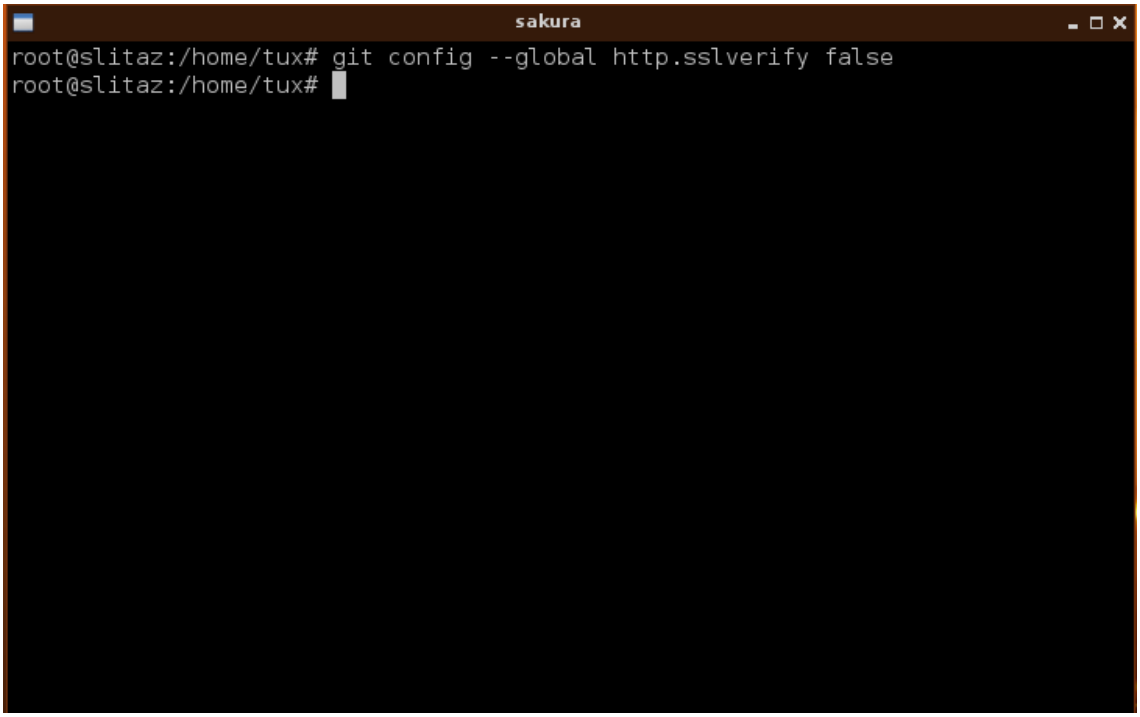


Figure 6- Setting ssl verify as false.

```
root@slitaz:/home/tux# git clone git://www.github.com/sqlmapproject/sqlmap.git s
qlmap-dev
Cloning into 'sqlmap-dev'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 77838 (delta 0), reused 0 (delta 0), pack-reused 77835
Receiving objects: 100% (77838/77838), 73.45 MiB | 13.31 MiB/s, done.
Resolving deltas: 100% (61617/61617), done.
root@slitaz:/home/tux#
```

Figure 7- git cloning on Slitaz

Python3 had some dependencies so I installed python 2.7.

```
root@slitaz:/home/tux# tazpkg get-install python

Connecting to mirror.slitaz.org (37.187.30.25:80)
python-2.7.2.tazpkg  100% |*****************************************************************************

Installation of : python
==============================================================================
Copying python...                                                   [ OK ]
Extracting python...                                                [ OK ]
Extracting the pseudo fs... (lzma)                                  [ OK ]
Installing python...                                                [ OK ]
Removing all tmp files...                                           [ OK ]
==============================================================================
python (2.7.2) is installed.
```

Figure 8- Installing python2.7 on Slitaz

Now, the sql map is installed and ready to run.

```
root@slitaz:/home/tux/sqlmap-dev# python sqlmap.py -hh
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.5.1.28#dev}
|_ -| . [,]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

Usage: python sqlmap.py [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
    At least one of these options has to be provided to define the
    target(s)

    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -d DIRECT           Connection string for direct database connection
    -l LOGFILE          Parse target(s) from Burp or WebScarab proxy log file
    -m BULKFILE         Scan multiple targets given in a textual file
    -r REQUESTFILE      Load HTTP request from a file
    -g GOOGLEDORK       Process Google dork results as target URLs
    -c CONFIGFILE       Load options from a configuration INI file

  Request:
    These options can be used to specify how to connect to the target URL

    -A AGENT, --user..  HTTP User-Agent header value
    -H HEADER, --hea..  Extra header (e.g. "X-Forwarded-For: 127.0.0.1")
    --method=METHOD     Force usage of given HTTP method (e.g. PUT)
    --data=DATA         Data string to be sent through POST (e.g. "id=1")
```

Figure 9- Starting sqlmap on Slitaz

https://www.geeksforgeeks.org/use-sqlmap-test-website-sql-injection-vulnerability/

https://github.com/sqlmapproject/sqlmap/

https://asciinema.org/a/46601

METASPOLIT FRAMEWORK

It depends on the postgresql database so first I installed it on my Slitaz Linux system

```
root@slitaz:/home/tux# tazpkg get-install postgresql

Connecting to mirror.slitaz.org (37.187.30.25:80)
postgresql-9.1.2.taz 100% |*****************************************************************************************|  2043k 00:00:00 ETA
Tracking dependencies for : postgresql
==============================================================================
Missing: postgresql-client
==============================================================================
1 missing package(s) to install.

Connecting to mirror.slitaz.org (37.187.30.25:80)
postgresql-client-9. 100% |*****************************************************************************************|   212k 00:00:00 ETA
Tracking dependencies for : postgresql-client
==============================================================================
Missing: libpostgresqlclient
==============================================================================
1 missing package(s) to install.

Connecting to mirror.slitaz.org (37.187.30.25:80)
libpostgresqlclient- 100% |*****************************************************************************************| 44572  --:--:-- ETA

Installation of : libpostgresqlclient
==============================================================================
Copying libpostgresqlclient...                                      [ OK ]
Extracting libpostgresqlclient...                                   [ OK ]
Extracting the pseudo fs... (lzma)                                  [ OK ]
Installing libpostgresqlclient...                                   [ OK ]
Removing all tmp files...                                           [ OK ]
==============================================================================
libpostgresqlclient (9.1.2) is installed.


Installation of : postgresql-client
```

Figure 10- Installing postgresql on Slitaz

I installed wget and then downloaded metaspolit

```
root@slitaz:/home/tux# tazpkg get-install wget

Connecting to mirror.slitaz.org (37.187.30.25:80)
wget-1.13.4.tazpkg   100% |*****************************************************************************************|   110k 00:00:00 ETA

Installation of : wget
==============================================================================
Copying wget...                                                     [ OK ]
Extracting wget...                                                  [ OK ]
Extracting the pseudo fs... (lzma)                                  [ OK ]
Processing pre-install commands...
Removing all Busybox replaced utils...                              [ OK ]
Installing wget...                                                  [ OK ]
Removing all tmp files...                                           [ OK ]

wget is in conflict with the busybox wget.
If you want to use the busybox wget, do (as root):
 ln -s /bin/busybox /usr/bin/wget

==============================================================================
wget (1.13.4) is installed.

root@slitaz:/home/tux# wget http://downloads.metasploit.com/data/releases/metasploit-latest-linux-installer.run
--2021-01-14 06:09:03--  http://downloads.metasploit.com/data/releases/metasploit-latest-linux-installer.run
Resolving downloads.metasploit.com (downloads.metasploit.com)... 96.6.229.197
Connecting to downloads.metasploit.com (downloads.metasploit.com)|96.6.229.197|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 153356021 (146M) [text/plain]
Saving to: `metasploit-latest-linux-installer.run'

100%[===================================================================================================>] 153,356,021 30.4M/s   in 5.3s

2021-01-14 06:09:10 (27.4 MB/s) - `metasploit-latest-linux-installer.run' saved [153356021/153356021]

root@slitaz:/home/tux#
```

Figure 11- Installing curl and downloading Metasploit on Slitaz

I changed the mode of installer to be executable.

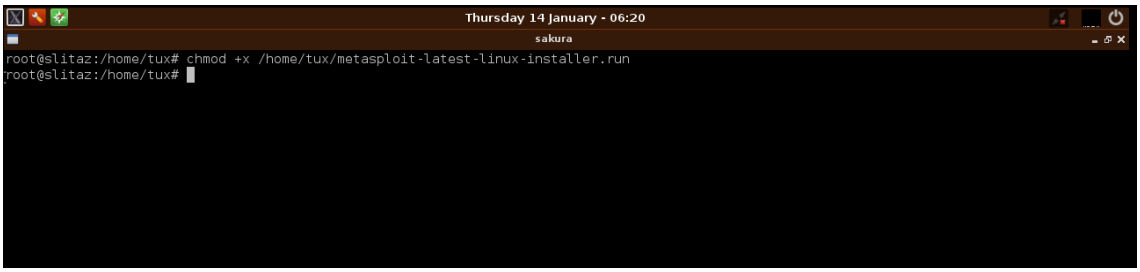


Figure 12- Changing the mode of Metasploit installer

Now I launch the installer.

https://subscription.packtpub.com/book/networking_and_servers/9781788295970/2/ch02lvl1sec21/installing-metasploit-on-linux

https://flicsdb.com/how-to-install-metasploit-on-kali-linux/

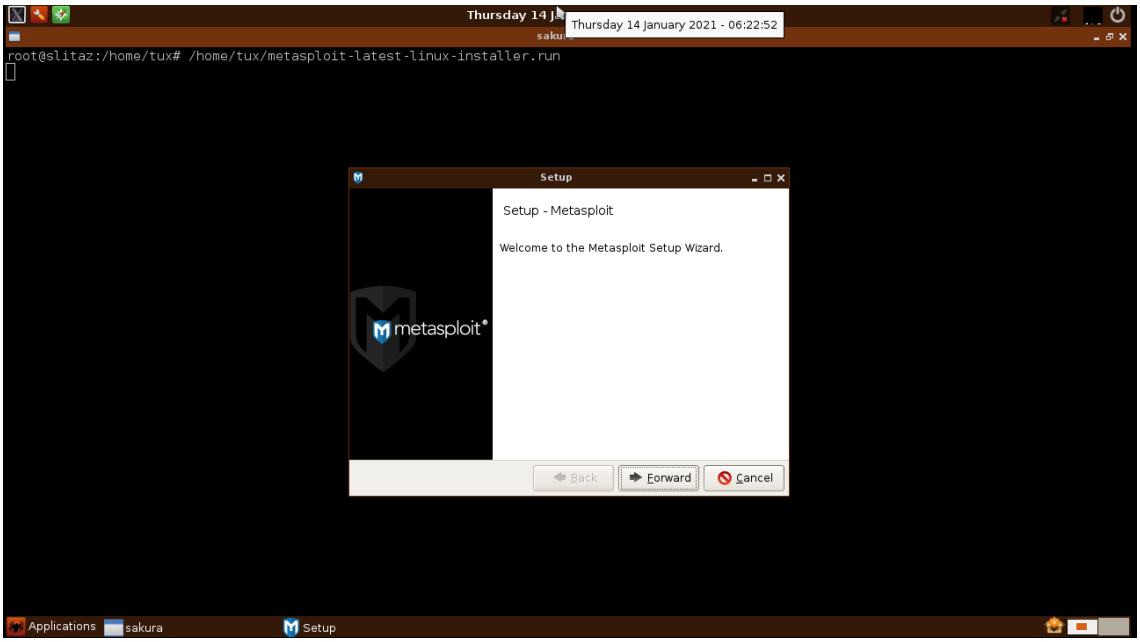


Figure 13- Launching Metasploit installer

SET

SET tool is not available on Slitaz and can be installed using the git cloning method.

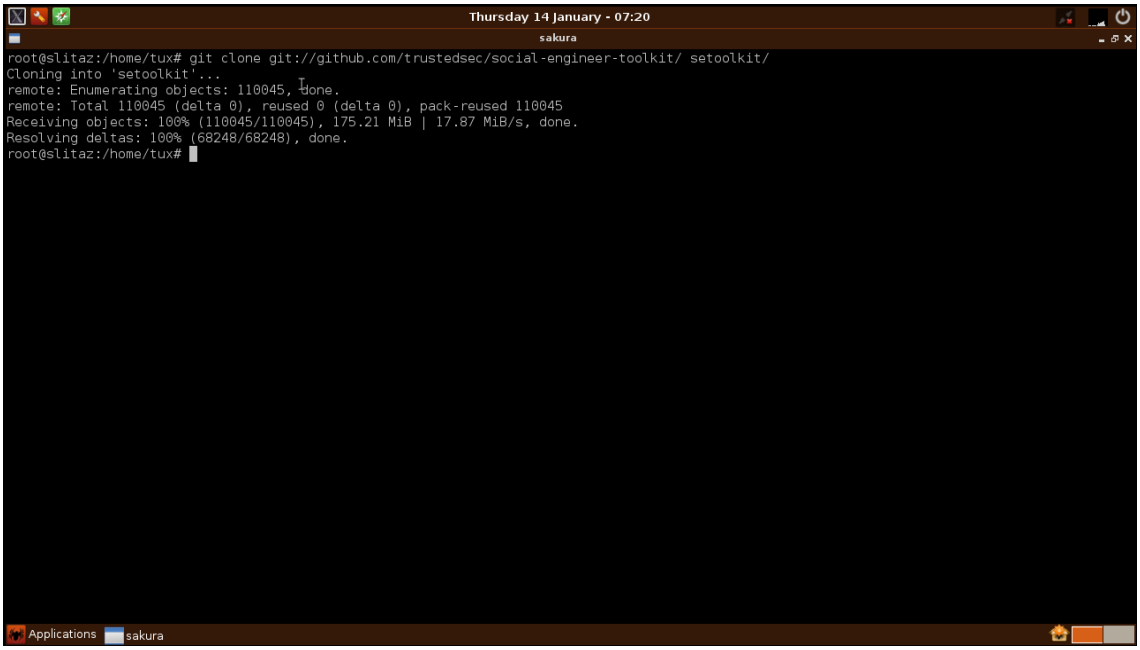


Figure 14- git cloning on Slitaz

https://github.com/trustedsec/social-engineer-toolkit/

As this works with python3 and pip so I had to install python first on the Slitaz system

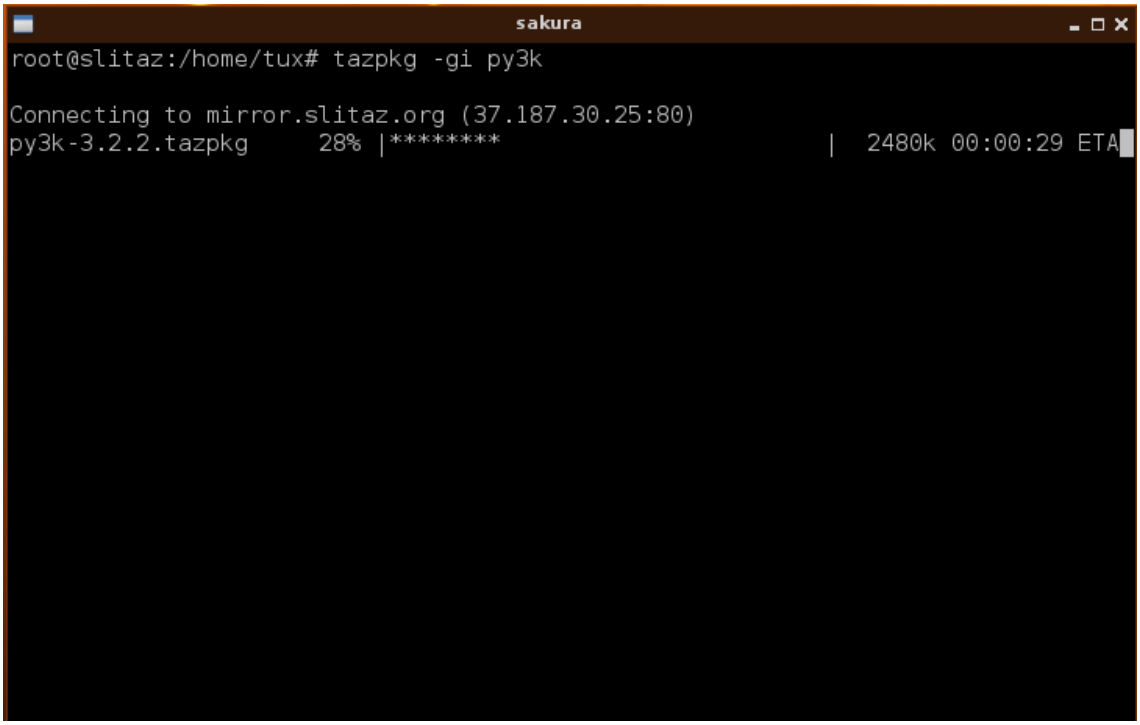


Figure 15- Installing python3 on Slitaz

This shows that pip3 does not come with python, so I must install pip manually.

https://pip.pypa.io/en/stable/installing/

Using **curl,** I try to download pip.

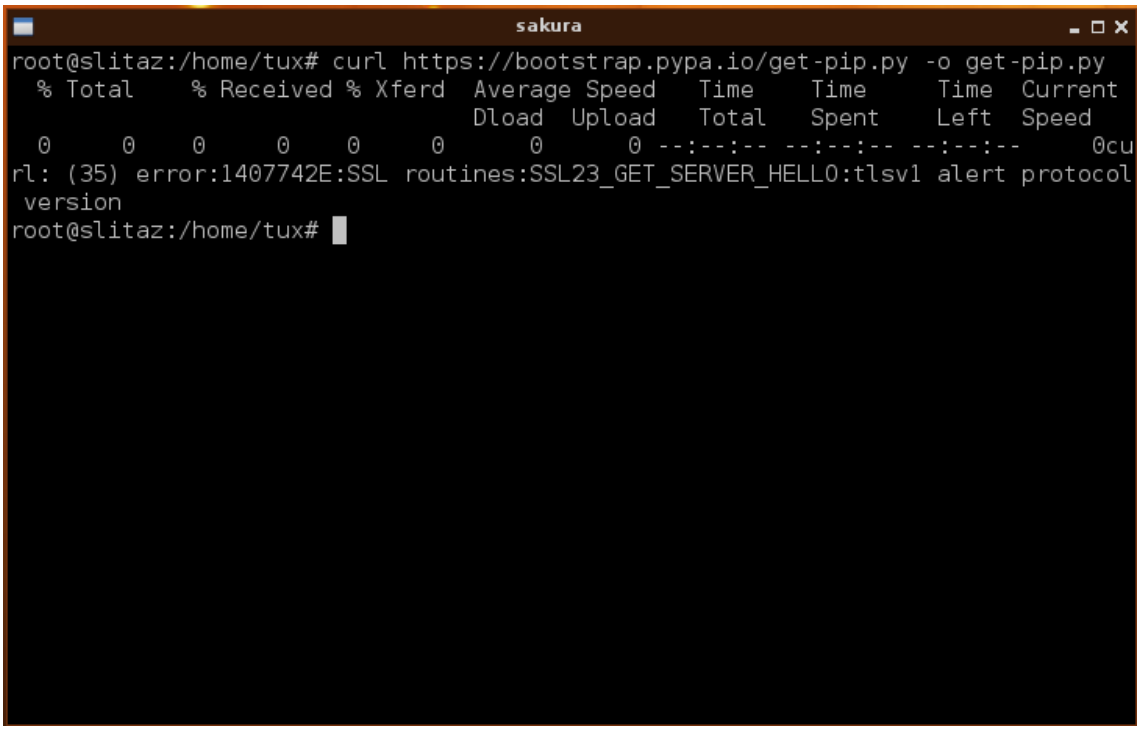


Figure 16- Installing pip3 in Slitaz

Here I get this error of tls v1 alert protocol version error because so many websites have now moved to tls v1.1, tls v1.2, tls v1.3 version because of enhanced security and unfortunately Slitaz version 4 does not support tls v1.1, tls v1.2, tls v1.3 version.

http://forum.slitaz.org/topic/cant-connect-to-https-addresses

I checked the TLS version of the website I tried to download a file using curl. And it shows the following version.

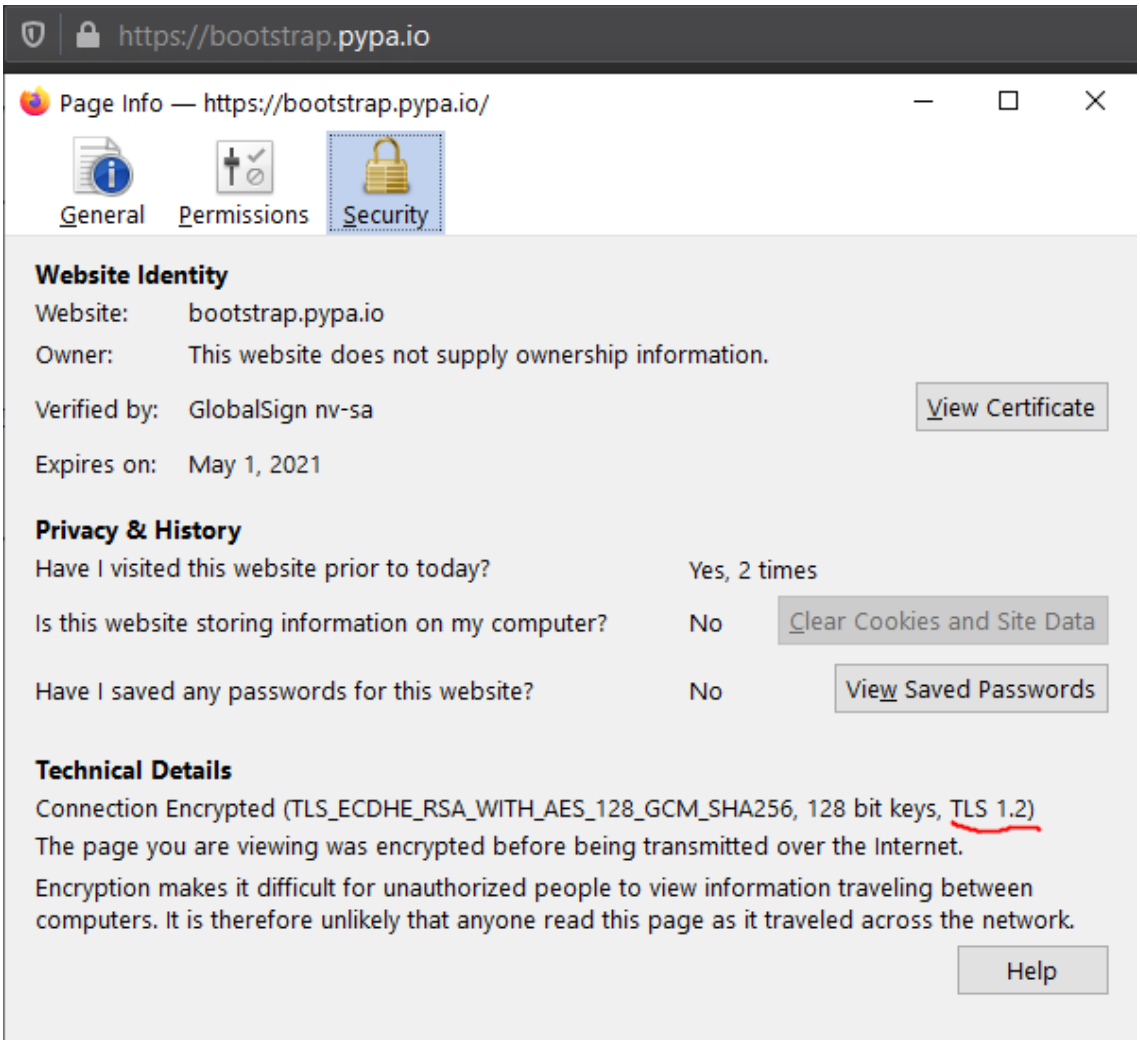


Figure 17- Checking the TLS version

After knowing this, I tried to install the newer version of Slitaz which is the version 5 because curl has a new version on it. Slitaz does not come with a GUI and a stable version for the version 5 so I tried doing it through the rolling version and command line interface.

First, installing python3.

```
root@slitaz:~# tazpkg -gi py3k
Creating folder "/var/cache/tazpkg"...                                  [ Done ]

Recharging repository "Main"
================================================================================
Checking...                                                             [ Done ]
Database timestamp: 01/14/2021 19:03
Creating backup of the last packages list...                            [ Done ]
Getting "bundle.tar.lzma"...                                            [ Done ]
Getting "files-list.lzma"...                                            [ Done ]
================================================================================
Last database is ready to use.

Note that next time you recharge the list, a list of differences will be
displayed to show new and upgradeable packages.

py3k-3.7.0.tazpkg      2% |                                  |  621k  0:03:05 ETA
```

Figure 18- Installing python3 on Slitaz

Now checking if it has pip come with the python 3.

So, the pip does come with it.

```
root@slitaz:~# python3 -m pip --version
pip 10.0.1 from /usr/lib/python3.7/site-packages/pip (python 3.7)
root@slitaz:~#
```

Figure 19- Checking pip version on Slitaz

Now trying to download SET through git cloning.

```
root@slitaz:~# git clone https://github.com/trustedsec/social-engineer-toolkit s
etoolkit/
Cloning into 'setoolkit'...
fatal: unable to access 'https://github.com/trustedsec/social-engineer-toolkit/'
: SSL certificate problem: unable to get local issuer certificate
root@slitaz:~# _
```

Figure 20- git cloning on Slitaz

To resolve this error, I turned off the ssl verification for git

```
root@slitaz:~# git config --global http.sslverify false
root@slitaz:~#
```

Figure 21- Turning of SSL verification for git.

Now trying again

```
root@slitaz:~# git clone https://github.com/trustedsec/social-engineer-toolkit s
etoolkit/
Cloning into 'setoolkit'...
remote: Enumerating objects: 110045, done.
remote: Total 110045 (delta 0), reused 0 (delta 0), pack-reused 110045
Receiving objects: 100% (110045/110045), 175.21 MiB | 26.35 MiB/s, done.
Resolving deltas: 100% (68248/68248), done.
root@slitaz:~#
```

Figure 22- git cloning on Slitaz

The cloning is done now it is time to go for next command.

```
        self.run_command(cmd_name)
      File "/usr/lib/python3.7/distutils/cmd.py", line 313, in run_command
        self.distribution.run_command(command)
      File "/usr/lib/python3.7/distutils/dist.py", line 985, in run_command
        cmd_obj.run()
      File "/tmp/pip-install-fk4ppho0/pycrypto/setup.py", line 251, in run
        self.run_command(cmd_name)
      File "/usr/lib/python3.7/distutils/cmd.py", line 313, in run_command
        self.distribution.run_command(command)
      File "/usr/lib/python3.7/distutils/dist.py", line 985, in run_command
        cmd_obj.run()
      File "/tmp/pip-install-fk4ppho0/pycrypto/setup.py", line 278, in run
        raise RuntimeError("autoconf error")
    RuntimeError: autoconf error

    ----------------------------------------
Command "/usr/bin/python3.7 -u -c "import setuptools, tokenize;__file__='/tmp/pi
p-install-fk4ppho0/pycrypto/setup.py';f=getattr(tokenize, 'open', open)(__file__
);code=f.read().replace('\r\n', '\n');f.close();exec(compile(code, __file__, 'ex
ec'))" install --record /tmp/pip-record-e7k7kl6u/install-record.txt --single-ver
sion-externally-managed --compile" failed with error code 1 in /tmp/pip-install-
fk4ppho0/pycrypto/
You are using pip version 10.0.1, however version 20.3.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
root@slitaz:~/setoolkit# _
```

Figure 23- pip version error

It gives this error, I upgraded pip version and then tried again.

```
  creating build/temp.linux-i686-3.7/build/temp.linux-i686-3.7
  i486-slitaz-linux-gcc -pthread -Wno-unused-result -Wsign-compare -DNDEBUG -g -
fwrapv -O3 -Wall -march=i486 -Os -pipe -fomit-frame-pointer -march=i486 -Os -pip
e -fomit-frame-pointer -fPIC -I/usr/include/python3.7m -c build/temp.linux-i686-
3.7/_openssl.c -o build/temp.linux-i686-3.7/build/temp.linux-i686-3.7/_openssl.o
 -Wconversion -Wno-error=sign-conversion

      ============================DEBUG ASSISTANCE============================
      If you are seeing a compilation error please try the following steps to
      successfully install cryptography:
      1) Upgrade to the latest pip and try again. This will fix errors for most
         users. See: https://pip.pypa.io/en/stable/installing/#upgrading-pip
      2) Read https://cryptography.io/en/latest/installation.html for specific
         instructions for your platform.
      3) Check our frequently asked questions for more information:
         https://cryptography.io/en/latest/faq.html
      ============================DEBUG ASSISTANCE============================

  error: command 'i486-slitaz-linux-gcc' failed with exit status 1
  ----------------------------------------
  ERROR: Failed building wheel for cryptography
Failed to build cryptography
ERROR: Could not build wheels for cryptography which use PEP 517 and cannot be i
nstalled directly
```

Figure 24- Compiler error on Slitaz

I installed the Slitaz toolchain to solve the problems with compiling the code. This tool chain comes with a compiler, a linker and other run time libraries that are required to execute a program. This way the Slitaz will not have any issues in compiling the program.

```
root@slitaz:~/setoolkit# tazpkg -gi slitaz-toolchain
slitaz-toolchain-5.0 100% |*********************************|  4668  0:00:00 ETA
Tracking dependencies for package "slitaz-toolchain"
==============================================================================
Missing package "binutils"
Missing package "linux-api-headers"
Missing package "glibc-dev"
Missing package "gcc"
Missing package "make"
Missing package "elfkickers"
==============================================================================
6 missing packages to install.

binutils-2.33.1.tazp  74% |************************         | 1043k  0:00:01 ETA
```

Figure 25- Installing Slitaz toolchain on Slitaz

Even after installing the Slitaz toolchain I got the same error so as I am trying to use pip 3 and python, I installed the python setup tools and development tools to more easily build and distribute the Python packages, particularly the ones that have dependencies on some other packages. I installed py3k-dev (py3k means python3 in Slitaz), py3k-cython, and py3k-setuptools_scm and gmp-dev.

```
root@slitaz:~/setoolkit# tazpkg -gi py3k-dev
py3k-dev-3.7.0.tazpk 100% |*********************************|  125k  0:00:00 ETA
Installation of package "py3k-dev"
===============================================================================
The Python programming language devel files.
-------------------------------------------------------------------------------
Copying package...                                                    [ Done ]
Extracting package...                                                 [ Done ]
Remember modified packages...                                         [ Done ]
Installing package...                                                 [ Done ]
Removing all tmp files...                                             [ Done ]
===============================================================================
Package "py3k-dev" (3.7.0) is installed.
```

Figure 26- Installing Python3 development tools on Slitaz

```
root@slitaz:~/setoolkit# tazpkg -gi py3k-cython
py3k-cython-0.29.13. 100% |*********************************| 2492k  0:00:00 ETA
Installation of package "py3k-cython"
===============================================================================
Language to write C extensions for Python.
-------------------------------------------------------------------------------
Copying package...                                                    [ Done ]
Extracting package...                                                 [ Done ]
Remember modified packages...                                         [ Done ]
Installing package...                                                 [ Done ]
Removing all tmp files...                                             [ Done ]
===============================================================================
Package "py3k-cython" (0.29.13) is installed.

root@slitaz:~/setoolkit# _
```

Figure 27- Installing py3k-cython on Slitaz

```
root@slitaz:~/setoolkit# tazpkg -gi py3k-setuptools_scm
py3k-setuptools_scm- 100% |*********************************| 34836  0:00:00 ETA
Installation of package "py3k-setuptools_scm"
===============================================================================
The blessed package to manage your versions by scm tags
-------------------------------------------------------------------------------
Copying package...                                                    [ Done ]
Extracting package...                                                 [ Done ]
Remember modified packages...                                         [ Done ]
Installing package...                                                 [ Done ]
Removing all tmp files...                                             [ Done ]
===============================================================================
Package "py3k-setuptools_scm" (3.3.3) is installed.

root@slitaz:~/setoolkit#
```

Figure 28- Installing py3k-setuptools-scm on Slitaz

```
root@slitaz:~/setoolkit# tazpkg -gi gmp-dev
gmp-dev-6.2.0.tazpkg 100% |*********************************|  212k  0:00:00 ETA
Installation of package "gmp-dev"
================================================================================
GNU Multiple Precision Arithmetic - development files.
--------------------------------------------------------------------------------
Copying package...                                                    [ Done ]
Extracting package...                                                 [ Done ]
Remember modified packages...                                         [ Done ]
Installing package...                                                 [ Done ]
Removing all tmp files...                                             [ Done ]
================================================================================
Package "gmp-dev" (6.2.0) is installed.

root@slitaz:~/setoolkit# _
```

Figure 29- Installing gmp-dev on Slitaz

```
e -fomit-frame-pointer -fPIC -I/usr/include/python3.7m -c build/temp.linux-i686-
3.7/_openssl.c -o build/temp.linux-i686-3.7/build/temp.linux-i686-3.7/_openssl.o
 -Wconversion -Wno-error=sign-conversion
  build/temp.linux-i686-3.7/_openssl.c:575:30: fatal error: openssl/opensslv.h:
No such file or directory
  compilation terminated.

      ============================DEBUG ASSISTANCE============================
      If you are seeing a compilation error please try the following steps to
      successfully install cryptography:
      1) Upgrade to the latest pip and try again. This will fix errors for most
         users. See: https://pip.pypa.io/en/stable/installing/#upgrading-pip
      2) Read https://cryptography.io/en/latest/installation.html for specific
         instructions for your platform.
      3) Check our frequently asked questions for more information:
         https://cryptography.io/en/latest/faq.html
      ============================DEBUG ASSISTANCE============================

  error: command 'i486-slitaz-linux-gcc' failed with exit status 1
  ----------------------------------------
  ERROR: Failed building wheel for cryptography
Failed to build cryptography
ERROR: Could not build wheels for cryptography which use PEP 517 and cannot be i
nstalled directly
root@slitaz:~/setoolkit# _
```

Figure 30- Compiler error on Slitaz

As it can be seen from the above image that even after installing the python setup tools and the slitaz toolchain, the compilation was successful, but it terminated due to an openssl error. I searched for openssl packages on slitaz and installed the open-ssl-dev.

```
root@slitaz:~/setoolkit# tazpkg -gi openssl-dev
openssl-dev-1.0.2u.t 100% |*********************************|  375k  0:00:00 ETA
Tracking dependencies for package "openssl-dev"
================================================================================
Missing package "libcrypto-dev"
Missing package "pkg-config"
================================================================================
2 missing packages to install.

libcrypto-dev-1.0.2u   4% |*                                | 34920  0:00:38 ETA
```

Figure 31- Installing openssl-dev on Slitaz

```
  build/temp.linux-i686-3.7/_openssl.c:50656:3: warning: implicit declaration of
 function ■i2d_re_X509_CRL_tbs■ [-Wimplicit-function-declaration]
  build/temp.linux-i686-3.7/_openssl.c: In function ■_cffi_d_i2d_re_X509_REQ_tbs
■:
  build/temp.linux-i686-3.7/_openssl.c:50709:3: warning: implicit declaration of
 function ■i2d_re_X509_REQ_tbs■ [-Wimplicit-function-declaration]

      ============================DEBUG ASSISTANCE============================
      If you are seeing a compilation error please try the following steps to
      successfully install cryptography:
      1) Upgrade to the latest pip and try again. This will fix errors for most
         users. See: https://pip.pypa.io/en/stable/installing/#upgrading-pip
      2) Read https://cryptography.io/en/latest/installation.html for specific
         instructions for your platform.
      3) Check our frequently asked questions for more information:
         https://cryptography.io/en/latest/faq.html
      ============================DEBUG ASSISTANCE============================

  error: command 'i486-slitaz-linux-gcc' failed with exit status 1
  ----------------------------------------
  ERROR: Failed building wheel for cryptography
Failed to build cryptography
ERROR: Could not build wheels for cryptography which use PEP 517 and cannot be i
nstalled directly
root@slitaz:~/setoolkit# clear
```

Figure 32- Compiler error on Slitaz

It still gave error, so I installed the python packages for cryptography and ssl. It only worked when I set the cryptography version to 3.2.1.

```
root@slitaz:~/setoolkit# pip3 install cryptography==3.2.1
Collecting cryptography==3.2.1
  Downloading cryptography-3.2.1.tar.gz (540 kB)
     |[redacted]| 540 kB 2.1 MB/s
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
    Preparing wheel metadata ... done
Collecting cffi!=1.11.3,>=1.8
  Using cached cffi-1.14.4-cp37-cp37m-manylinux1_i686.whl (379 kB)
Collecting six>=1.4.1
  Using cached six-1.15.0-py2.py3-none-any.whl (10 kB)
Collecting pycparser
  Using cached pycparser-2.20-py2.py3-none-any.whl (112 kB)
Building wheels for collected packages: cryptography
  Building wheel for cryptography (PEP 517) ... done
  Created wheel for cryptography: filename=cryptography-3.2.1-cp37-cp37m-linux_i
686.whl size=623437 sha256=63163aadea1088923231f250aee678ace5bc6811ef8531d7fdb62
c11eee23192
  Stored in directory: /root/.cache/pip/wheels/03/a5/70/f834a5c113244ec554aefd6e
b6b778e5ecc719567cb42b8a8f
Successfully built cryptography
Installing collected packages: pycparser, six, cffi, cryptography
Successfully installed cffi-1.14.4 cryptography-3.2.1 pycparser-2.20 six-1.15.0
```

Figure 33- Installing cryptography=3.2.1 on Slitaz

```
root@slitaz:~/setoolkit# pip3 install pyopenssl
Collecting pyopenssl
  Using cached pyOpenSSL-20.0.1-py2.py3-none-any.whl (54 kB)
Requirement already satisfied: cryptography>=3.2 in /usr/lib/python3.7/site-pack
ages (from pyopenssl) (3.2.1)
Requirement already satisfied: six>=1.5.2 in /usr/lib/python3.7/site-packages (f
rom pyopenssl) (1.15.0)
Requirement already satisfied: cffi!=1.11.3,>=1.8 in /usr/lib/python3.7/site-pac
kages (from cryptography>=3.2->pyopenssl) (1.14.4)
Requirement already satisfied: pycparser in /usr/lib/python3.7/site-packages (fr
om cffi!=1.11.3,>=1.8->cryptography>=3.2->pyopenssl) (2.20)
Installing collected packages: pyopenssl
Successfully installed pyopenssl-20.0.1
root@slitaz:~/setoolkit# _
```

Figure 34- Installing pyopenssl on Slitaz

```
root@slitaz:~/setoolkit# pip3 install pycrypto
Collecting pycrypto
  Using cached pycrypto-2.6.1.tar.gz (446 kB)
Using legacy 'setup.py install' for pycrypto, since package 'wheel' is not insta
lled.
Installing collected packages: pycrypto
    Running setup.py install for pycrypto ... done
Successfully installed pycrypto-2.6.1
root@slitaz:~/setoolkit# _
```

Figure 35- Installing pycrypto on Slitaz

As everything was installed successfully, I ran the command “pip3 install –r requirements.txt” and it worked.

```
Collecting dnspython
  Using cached dnspython-2.1.0-py3-none-any.whl (241 kB)
Collecting future
  Using cached future-0.18.2.tar.gz (829 kB)
Collecting pycryptodomex
  Using cached pycryptodomex-3.9.9-cp37-cp37m-manylinux1_i686.whl (13.7 MB)
Using legacy 'setup.py install' for impacket, since package 'wheel' is not insta
lled.
Using legacy 'setup.py install' for pefile, since package 'wheel' is not install
ed.
Using legacy 'setup.py install' for future, since package 'wheel' is not install
ed.
Installing collected packages: pyasn1, MarkupSafe, Werkzeug, ldap3, Jinja2, itsd
angerous, future, dnspython, click, urllib3, pycryptodomex, ptyprocess, ldapdoma
indump, idna, flask, chardet, certifi, requests, qrcode, pymssql, pillow, pexpec
t, pefile, impacket
    Running setup.py install for future ... done
    Running setup.py install for pefile ... done
    Running setup.py install for impacket ... done
Successfully installed Jinja2-2.11.2 MarkupSafe-1.1.1 Werkzeug-1.0.1 certifi-202
0.12.5 chardet-4.0.0 click-7.1.2 dnspython-2.1.0 flask-1.1.2 future-0.18.2 idna-
2.10 impacket-0.9.22 itsdangerous-1.1.0 ldap3-2.8.1 ldapdomaindump-0.9.3 pefile-
2019.4.18 pexpect-4.8.0 pillow-8.1.0 ptyprocess-0.7.0 pyasn1-0.4.8 pycryptodomex
-3.9.9 pymssql-2.1.5 qrcode-6.1 requests-2.25.1 urllib3-1.26.2
root@slitaz:~/setoolkit#
```

Figure 36- Successful build of pip

Then I ran the command “python3 setup.py” but it gave an error that directory is not found

```
Requirement already satisfied: future in /usr/lib/python3.7/site-packages (from
ldapdomaindump>=0.9.0->impacket->-r requirements.txt (line 6)) (0.18.2)
Requirement already satisfied: dnspython in /usr/lib/python3.7/site-packages (fr
om ldapdomaindump>=0.9.0->impacket->-r requirements.txt (line 6)) (2.1.0)
Requirement already satisfied: ptyprocess>=0.5 in /usr/lib/python3.7/site-packag
es (from pexpect->-r requirements.txt (line 1)) (0.7.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/lib/python3.7/site-
packages (from requests->-r requirements.txt (line 3)) (1.26.2)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/lib/python3.7/site-pack
ages (from requests->-r requirements.txt (line 3)) (4.0.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/lib/python3.7/site-pac
kages (from requests->-r requirements.txt (line 3)) (2020.12.5)
Requirement already satisfied: idna<3,>=2.5 in /usr/lib/python3.7/site-packages
(from requests->-r requirements.txt (line 3)) (2.10)
[*] Installing setoolkit to /usr/share/setoolkit..
/root/setoolkit
mkdir: can't create directory '/usr/share/setoolkit/': File exists
mkdir: can't create directory '/etc/setoolkit/': File exists
[*] Creating launcher for setoolkit...
Traceback (most recent call last):
  File "setup.py", line 15, in <module>
    filewrite = open("/usr/local/bin/setoolkit", "w")
FileNotFoundError: [Errno 2] No such file or directory: '/usr/local/bin/setoolki
t'
```

Figure 37- Directory Error on Slitaz

To solve this, I made the directory that was found and ran the command again.

```
ages (from requests->-r requirements.txt (line 3)) (4.0.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/lib/python3.7/site-pac
kages (from requests->-r requirements.txt (line 3)) (2020.12.5)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/lib/python3.7/site-
packages (from requests->-r requirements.txt (line 3)) (1.26.2)
[*] Installing setoolkit to /usr/share/setoolkit..
/root/setoolkit
mkdir: can't create directory '/usr/share/setoolkit/': File exists
mkdir: can't create directory '/etc/setoolkit/': File exists
[*] Creating launcher for setoolkit...
Traceback (most recent call last):
  File "setup.py", line 15, in <module>
    filewrite = open("/usr/local/bin/setoolkit", "w")
FileNotFoundError: [Errno 2] No such file or directory: '/usr/local/bin/setoolki
t'
root@slitaz:~/setoolkit# mkdir /usr/local/bin/setoolkit
mkdir: can't create directory '/usr/local/bin/setoolkit': No such file or direct
ory
root@slitaz:~/setoolkit# mkdir /usr/local/
root@slitaz:~/setoolkit# mkdir /usr/local/bin/setoolkit
mkdir: can't create directory '/usr/local/bin/setoolkit': No such file or direct
ory
root@slitaz:~/setoolkit# mkdir /usr/local/bin/
root@slitaz:~/setoolkit# mkdir /usr/local/bin/setoolkit
root@slitaz:~/setoolkit# _
```

Figure 38- Making directory for setoolkit

After solving the directory issue and running the command again, it finally ran.

```
Requirement already satisfied: MarkupSafe>=0.23 in /usr/lib/python3.7/site-packa
ges (from Jinja2>=2.10.1->flask>=1.0->impacket->-r requirements.txt (line 7)) (1
.1.1)
Requirement already satisfied: dnspython in /usr/lib/python3.7/site-packages (fr
om ldapdomaindump>=0.9.0->impacket->-r requirements.txt (line 7)) (2.1.0)
Requirement already satisfied: future in /usr/lib/python3.7/site-packages (from
ldapdomaindump>=0.9.0->impacket->-r requirements.txt (line 7)) (0.18.2)
Requirement already satisfied: ptyprocess>=0.5 in /usr/lib/python3.7/site-packag
es (from pexpect->-r requirements.txt (line 1)) (0.7.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/lib/python3.7/site-packages
(from requests->-r requirements.txt (line 4)) (2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/lib/python3.7/site-pack
ages (from requests->-r requirements.txt (line 4)) (4.0.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/lib/python3.7/site-pac
kages (from requests->-r requirements.txt (line 4)) (2020.12.5)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/lib/python3.7/site-
packages (from requests->-r requirements.txt (line 4)) (1.26.2)
[*] Installing setoolkit to /usr/share/setoolkit..
/root/setoolkit
mkdir: can't create directory '/usr/share/setoolkit/': File exists
mkdir: can't create directory '/etc/setoolkit/': File exists
[*] Creating launcher for setoolkit...
[*] Done. Chmoding +x....
[*] Finished. Run 'setoolkit' to start the Social Engineer Toolkit.
```

Figure 39- Successful run of setoolkit

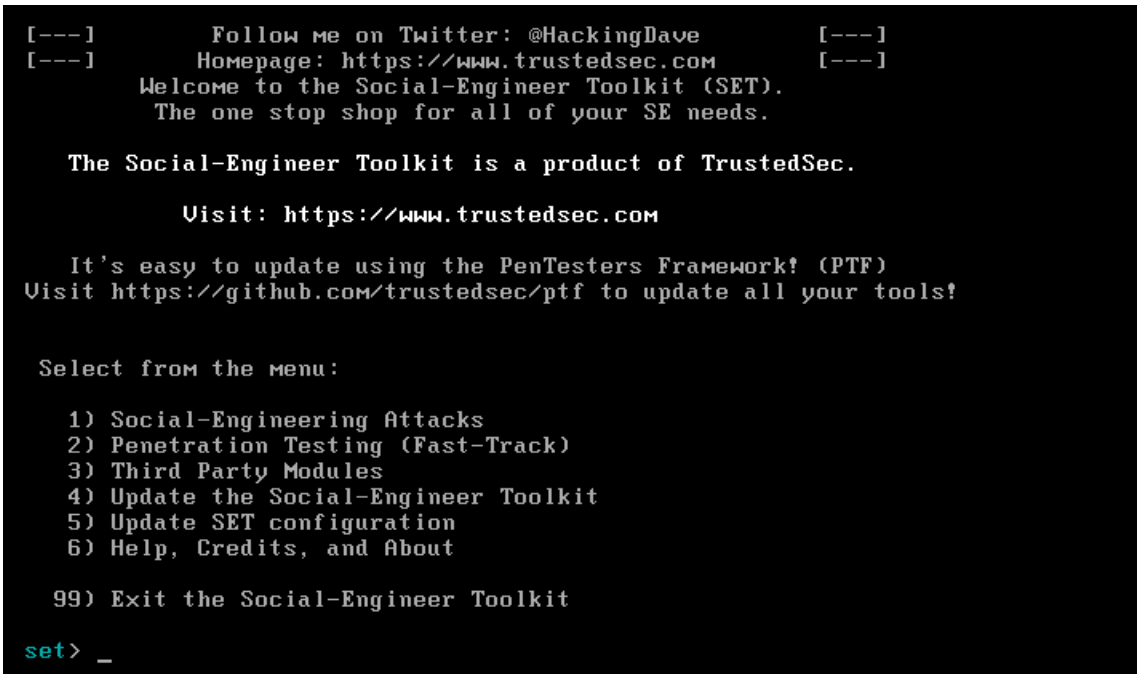


Figure 40- Successful start of SET tool in Slitaz

Installing this SET tool required a lot of steps and it had a lot of dependencies one after the another. First, it was required to install the Linux distribution Slitaz's basic toolchain which is significant for every distribution. Other than that, as I was using python3 so its dev packages, setup tools, and the python pip packages were also required.

Tiny Core

Before installing any security tools, I installed the VMware Tools.

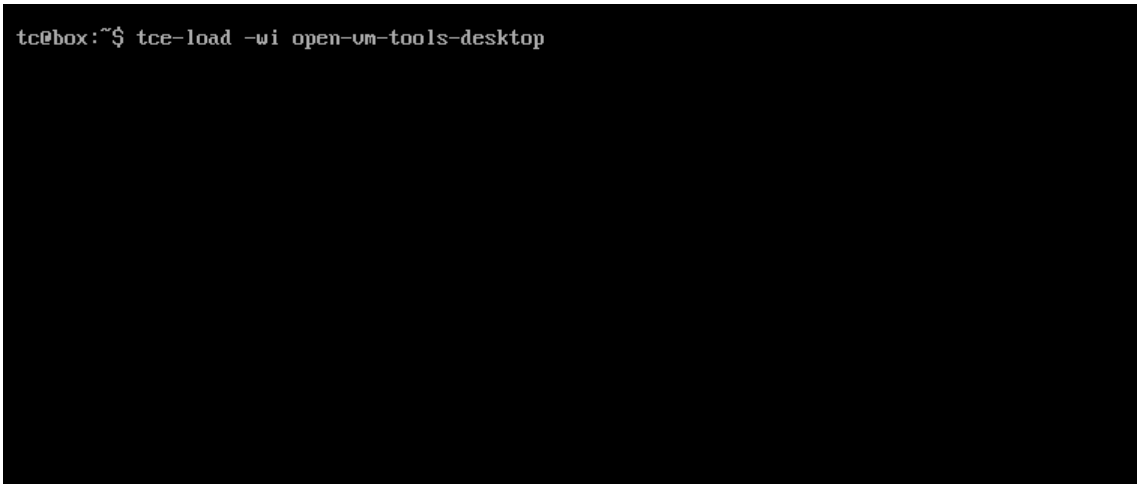


Figure 41- Installing VMware tools on Tinycore

Nmap

Nmap is already available on Tinycore Linux and it can be installed using tce-load package manager.

```
tc@box:~$ tce-load -wi nmap
nmap.tcz.dep OK
libpcap.tcz.dep OK
libusb.tcz.dep OK
libssh2.tcz.dep OK
Downloading: liblinear.tcz
Connecting to repo.tinycorelinux.net (89.22.99.37:80)
saving to 'liblinear.tcz'
liblinear.tcz        100% |*******************************************************************************************************************| 28672  0:00:00 ETA
'liblinear.tcz' saved
liblinear.tcz: OK
Downloading: libssh2.tcz
Connecting to repo.tinycorelinux.net (89.22.99.37:80)
saving to 'libssh2.tcz'
libssh2.tcz          100% |*******************************************************************************************************************|  100k  0:00:00 ETA
'libssh2.tcz' saved
libssh2.tcz: OK
Downloading: lua-lib.tcz
Connecting to repo.tinycorelinux.net (89.22.99.37:80)
saving to 'lua-lib.tcz'
```

Figure 42- Installing nmap on Tinycore

Sqlmap

Sqlmap is not available on Slitaz so I will install it through the git cloning process.

```
tc@box:~$ git clone https://www.github.com/sqlmapproject/sqlmap.git sqlmap-dev
Cloning into 'sqlmap-dev'...
warning: redirecting to https://github.com/sqlmapproject/sqlmap.git/
remote: Enumerating objects: 59, done.
remote: Counting objects: 100% (59/59), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 77894 (delta 30), reused 30 (delta 15), pack-reused 77835
Receiving objects: 100% (77894/77894), 73.62 MiB | 28.32 MiB/s, done.
Resolving deltas: 100% (61647/61647), done.
tc@box:~$ _
```

Figure 43- git cloning on Tinycore

First step is to install python.

```
tc@box:~$ tce-load -wi python
python.tcz.dep OK
Downloading: sqlite3.tcz
Connecting to repo.tinycorelinux.net (89.22.99.37:80)
saving to 'sqlite3.tcz'
sqlite3.tcz          100% |*********************************************************
'sqlite3.tcz' saved
sqlite3.tcz: OK
Downloading: gdbm.tcz
Connecting to repo.tinycorelinux.net (89.22.99.37:80)
saving to 'gdbm.tcz'
gdbm.tcz             100% |*********************************************************
'gdbm.tcz' saved
gdbm.tcz: OK
Downloading: python.tcz
Connecting to repo.tinycorelinux.net (89.22.99.37:80)
saving to 'python.tcz'
python.tcz           100% |*********************************************************
'python.tcz' saved
python.tcz: OK
tc@box:~$
```

Figure 44- Installing python on Tinycore

Starting the sqlmap that was cloned.

```
tc@box:~/sqlmap-dev$ python sqlmap.py -h_
```

Figure 45- Starting sqmap on Tinycore

Wireshark

Wireshark is already available on tinycore Linux and it can be installed by the tce-load package manager.

```
tc@box:~$ tce-load -wi wireshark
wireshark.tcz.dep OK
gnutls3.6.tcz.dep OK
nettle3.tcz.dep OK
p11-kit.tcz.dep OK
libidn2.tcz.dep OK
libgcrypt.tcz.dep OK
Downloading: libnl.tcz
Connecting to repo.tinycorelinux.net (89.22.99.37:80)
saving to 'libnl.tcz'
libnl.tcz            100% |*******************************************************************************
'libnl.tcz' saved
libnl.tcz: OK
Downloading: liblz4.tcz
Connecting to repo.tinycorelinux.net (89.22.99.37:80)
saving to 'liblz4.tcz'
liblz4.tcz           100% |*******************************************************************************
'liblz4.tcz' saved
liblz4.tcz: OK
Downloading: libcares.tcz
Connecting to repo.tinycorelinux.net (89.22.99.37:80)
```

Figure 46- Installing wireshark on Tinycore

Metasploit Framework

It is not available on the Tinycore Linux so to install it, it is required to download a script.

```
tc@box:~$ wget http://downloads.metasploit.com/data/releases/metasploit-latest-l
inux-installer.run
--2021-01-18 23:14:24--  http://downloads.metasploit.com/data/releases/metasploi
t-latest-linux-installer.run
Resolving downloads.metasploit.com... 23.204.60.91
Connecting to downloads.metasploit.com|23.204.60.91|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 153356021 (146M) [text/plain]
Saving to: 'metasploit-latest-linux-installer.run'

metasploit-latest-l 100%[===================>] 146.25M  29.4MB/s    in 5.6s

2021-01-18 23:14:30 (26.3 MB/s) - 'metasploit-latest-linux-installer.run' saved
[153356021/153356021]

tc@box:~$ _
```

Figure 47- Downloading Metasploit on Tinycore

https://docs.rapid7.com/metasploit/installing-metasploit-pro/

Changing the mode of the installer to be executable

```
tc@box:~$ chmod +x /home/tc/metasploit-latest-linux-installer.run
tc@box:~$ _
```

Figure 48- Changing the installer mode on Tinycore

Running the installer

```
root@box:/home/tc# /home/tc/metasploit-latest-linux-installer.run
```

Figure 49- Launching the Metasploit installer on Tinycore

Metasploit framework gets started!

```
metasploit.example.com). A certificate is generated for a specific server name
and web browsers will alert users if the name does not match.

Server Name [localhost]:

Days of validity [3650]:

Should the generated certificate be added to the operating system's trusted
store?

Yes, trust certificate [Y/n]: y


-------------------------------------------------------------------------
Setup is now ready to begin installing Metasploit on your computer.

Do you want to continue? [Y/n]: y

-------------------------------------------------------------------------
Please wait while Setup installs Metasploit on your computer.

 Installing
 0% ______________ 50% ______________ 100%
 ######Killed
```

Figure 50- Starting Metasploit on Tinycore

SET

To install the SET information gathering tool in Tiny core Linux, I first installed git on the system by the command “tce-load –wi git”. After this, following the GitHub repository instructions of the SET tool set I cloned the tool using the below command as shown. It was done successfully.

```
tc@box:~$ git clone https://github.com/trustedsec/social-engineer-toolkit/ setoo
lkit/
Cloning into 'setoolkit'...
remote: Enumerating objects: 110045, done.
remote: Total 110045 (delta 0), reused 0 (delta 0), pack-reused 110045
Receiving objects: 100% (110045/110045), 175.21 MiB | 25.08 MiB/s, done.
Resolving deltas: 100% (68248/68248), done.
tc@box:~$ _
```

Figure 51- git cloning on Tinycore

https://pip.pypa.io/en/stable/installing/

As I had to use python and pip3 to get the required packages, I installed python 3.6 on the system and pip3 cam automatically with it so I did not have to download it separately. After this I ran the required command to install the requirements.txt python package as shown above.

```
tc@box:~/setoolkit$ pip3 install -r requirements.txt
Collecting pexpect (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/39/7b/88dbb785881c28a10261
9d46423cb853b46dbccc70d3ac362d99773a78ce/pexpect-4.8.0-py2.py3-none-any.whl (59k
B)
    100% |################################| 61kB 1.4MB/s
Collecting pycrypto (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/60/db/645aa9af249f059cc3a3
68b118de33889219e0362141e75d4eaf6f80f163/pycrypto-2.6.1.tar.gz (446kB)
    100% |################################| 450kB 809kB/s
Collecting requests (from -r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/29/c1/24814557f1d22c56d502
80771a17307e6bf87b70727d975fd6b2ce6b014a/requests-2.25.1-py2.py3-none-any.whl (6
1kB)
    100% |################################| 61kB 5.7MB/s
Collecting pyopenssl (from -r requirements.txt (line 4))
  Downloading https://files.pythonhosted.org/packages/b2/5e/06351ede29fd4899782a
d335c2e02f1f862a887c20a3541f17c3fa1a3525/pyOpenSSL-20.0.1-py2.py3-none-any.whl (
54kB)
    100% |################################| 61kB 28.1MB/s
Collecting pefile (from -r requirements.txt (line 5))
  Downloading https://files.pythonhosted.org/packages/36/58/acf7f35859d541985f0a
6ea3c34baaefbfaee23642cf11e85fe36453ae77/pefile-2019.4.18.tar.gz (62kB)
    100% |################################| 71kB 14.4MB/s
```

Figure 52- Installing requirements.txt python package on Tinycore

```
6))
  Downloading https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef8
f1da638721e9da21e5bacee597595b318f71d62e/Werkzeug-1.0.1-py2.py3-none-any.whl (29
8kB)
     100% |################################| 307kB 4.7MB/s
Collecting pycparser (from cffi>=1.12->cryptography>=3.2->pyopenssl->-r requirem
ents.txt (line 4))
  Using cached https://files.pythonhosted.org/packages/ae/e7/d9c3a176ca4b02024de
bf82342dab36efadfc5776f9c8db077e8f6e71821/pycparser-2.20-py2.py3-none-any.whl
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10.1->flask>=1.0->impacket->-r requi
rements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/35/25/8560907c79805c1ed2d1
b8297c43ad82f5f23a5376d846bc1a2ace2aee53/MarkupSafe-1.1.1-cp36-cp36m-manylinux1_
i686.whl
Installing collected packages: ptyprocess, pexpect, pycrypto, idna, chardet, cer
tifi, urllib3, requests, six, pycparser, cffi, cryptography, pyopenssl, future,
pefile, pyasn1, pycryptodomex, ldap3, dnspython, ldapdomaindump, MarkupSafe, Jin
ja2, itsdangerous, click, Werkzeug, flask, impacket, qrcode, pillow, pymssql
Could not install packages due to an EnvironmentError: [Errno 13] Permission den
ied: '/usr/local/lib/python3.6/site-packages/ptyprocess-0.7.0.dist-info'
Consider using the '--user' option or check the permissions.

You are using pip version 18.1, however version 20.3.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
tc@box:~/setoolkit$ _
```

Figure 53- - Environmental and pip version error on Tinycore

It gave an Environmental error and asked to upgrade pip to the latest version. For the first error I changed the user to root and then upgraded pip to the latest version as shown below

```
root@box:/home/tc/setoolkit# pip3 install --upgrade pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/54/eb/4a3642e971f404d69d4f
6fa3885559d67562801b99d7592487f1ecc4e017/pip-20.3.3-py2.py3-none-any.whl (1.5MB)
     100% |################################| 1.5MB 978kB/s
Installing collected packages: pip
  Found existing installation: pip 18.1
    Uninstalling pip-18.1:
      Successfully uninstalled pip-18.1
Successfully installed pip-20.3.3
root@box:/home/tc/setoolkit#
```

Figure 54- pip version upgrade in Tinycore

After upgrading pip and changing to root user, it gave some errors related to open ssl and gcc.

```
  creating build/temp.linux-i686-3.6/build
  creating build/temp.linux-i686-3.6/build/temp.linux-i686-3.6
  gcc -flto -fuse-linker-plugin -march=i486 -mtune=i686 -Os -pipe -pthread -Wno-
unused-result -Wsign-compare -DNDEBUG -Wall -fPIC -I/usr/local/include/python3.6
m -c build/temp.linux-i686-3.6/_openssl.c -o build/temp.linux-i686-3.6/build/tem
p.linux-i686-3.6/_openssl.o -Wconversion -Wno-error=sign-conversion

      ============================DEBUG ASSISTANCE============================
      If you are seeing a compilation error please try the following steps to
      successfully install cryptography:
      1) Upgrade to the latest pip and try again. This will fix errors for most
         users. See: https://pip.pypa.io/en/stable/installing/#upgrading-pip
      2) Read https://cryptography.io/en/latest/installation.html for specific
         instructions for your platform.
      3) Check our frequently asked questions for more information:
         https://cryptography.io/en/latest/faq.html
      ============================DEBUG ASSISTANCE============================

  error: command 'gcc' failed with exit status 1
  ----------------------------------------
  ERROR: Failed building wheel for cryptography
Failed to build cryptography
ERROR: Could not build wheels for cryptography which use PEP 517 and cannot be i
nstalled directly
root@box:/home/tc/setoolkit# _
```

Figure 55- Compiler error on Tinycore

First, I installed the compiletc package because it comes with all the tool chain, compilers and libraries to compile the python tiny core related packages.

http://forum.tinycorelinux.net/index.php?topic=8950.0

```
tc@box:~/setoolkit$ tce-load -wi compiletc
compiletc.tcz.dep OK
gawk.tcz.dep OK
mpfr.tcz.dep OK
gcc.tcz.dep OK
gcc_libs-dev.tcz.dep OK
binutils.tcz.dep OK
mpc.tcz.dep OK
patch.tcz.dep OK
Downloading: zlib_base-dev.tcz
Connecting to repo.tinycorelinux.net (89.22.99.37:80)
saving to 'zlib_base-dev.tcz'
zlib_base-dev.tcz    100% |*********************************| 94208  0:00:00 ETA
'zlib_base-dev.tcz' saved
zlib_base-dev.tcz: OK
Downloading: util-linux_base-dev.tcz
Connecting to repo.tinycorelinux.net (89.22.99.37:80)
saving to 'util-linux_base-dev.tcz'
```

Figure 56- Installing compiletc on Tinycore

After this, it still gave some error related to python, so I installed the python3.6-dev package as I am using the python 3.6 and pip 3 for this system. This package comes with all the development tools related to python.

```
p.linux-i686-3.6/_openssl.o -Wconversion -Wno-error=sign-conversion
  build/temp.linux-i686-3.6/_openssl.c:57:10: fatal error: Python.h: No such fil
e or directory
     57 | #include <Python.h>
        |          ^~~~~~~~~~
  compilation terminated.

      ==============================DEBUG ASSISTANCE==============================
      If you are seeing a compilation error please try the following steps to
      successfully install cryptography:
      1) Upgrade to the latest pip and try again. This will fix errors for most
         users. See: https://pip.pypa.io/en/stable/installing/#upgrading-pip
      2) Read https://cryptography.io/en/latest/installation.html for specific
         instructions for your platform.
      3) Check our frequently asked questions for more information:
         https://cryptography.io/en/latest/faq.html
      ==============================DEBUG ASSISTANCE==============================

  error: command 'gcc' failed with exit status 1
  ----------------------------------------
  ERROR: Failed building wheel for cryptography
Failed to build cryptography
ERROR: Could not build wheels for cryptography which use PEP 517 and cannot be i
nstalled directly
tc@box:~/setoolkit$ _
```

Figure 57- Python error on Tinycore

After installing the python development tools and running the command again, it gave an error related to gmp. So, I checked if I have the gmp package or not. I did have it, so I searched for more gmp packages and installed the gmp-dev package which are again development tools.

http://tinycorelinux.net/9.x/armv6/tcz/

```
     1481 | #define _POSIX_C_SOURCE 200809L
          |
    In file included from /usr/include/bits/libc-header-start.h:33,
                     from /usr/include/stdio.h:27,
                     from src/_fastmath.c:29:
    /usr/include/features.h:295: note: this is the location of the previous defi
nition
      295 | # define _POSIX_C_SOURCE 199506L
          |
    src/_fastmath.c:36:11: fatal error: gmp.h: No such file or directory
       36 | # include <gmp.h>
          |           ^~~~~~~
    compilation terminated.
    error: command 'gcc' failed with exit status 1
    ----------------------------------------
ERROR: Command errored out with exit status 1: /usr/local/bin/python3.6 -u -c 'i
mport sys, setuptools, tokenize; sys.argv[0] = '''''/tmp/pip-install-h6y984n1/py
crypto_b0545080bc6a4f459f311d218c2cea12/setup.py'''''; __file__='''''/tmp/pip-in
stall-h6y984n1/pycrypto_b0545080bc6a4f459f311d218c2cea12/setup.py'''''';f=getattr
(tokenize, '''''open'''''', open)(__file__);code=f.read().replace('''''\r\n'''''',
 '''''\n'''''');f.close();exec(compile(code, __file__, '''''exec'''''))' install
--record /tmp/pip-record-5nj0emo5/install-record.txt --single-version-externally
-managed --user --prefix= --compile --install-headers /home/tc/.local/include/py
thon3.6m/pycrypto Check the logs for full command output.
tc@box:~/setoolkit$ _
```

Figure 58- gmp error on Tinycore

After this, it finally successfully ran without any errors.

I ran “python3 setup.py” and the SET tool was finally cloned.

```
686.whl size=361731 sha256=cd235f3b427d2e7da4b47712989101e0c169f9b47226fca1d98b2
0cc4f700df2
  Stored in directory: /root/.cache/pip/wheels/1a/55/d3/e4def74afc4abd59779d4f71
7838681ff06b0c68ed60b8e489
Successfully built cryptography
Installing collected packages: pycparser, six, pyasn1, MarkupSafe, cffi, Werkzeu
g, ldap3, Jinja2, itsdangerous, future, dnspython, cryptography, click, urllib3,
 pyopenssl, pycryptodomex, ldapdomaindump, idna, flask, chardet, certifi, reques
ts, qrcode, pymssql, pycrypto, pillow, pefile, impacket
    Running setup.py install for future ... done
    Running setup.py install for pycrypto ... done
    Running setup.py install for pefile ... done
    Running setup.py install for impacket ... done
Successfully installed Jinja2-2.11.2 MarkupSafe-1.1.1 Werkzeug-1.0.1 certifi-202
0.12.5 cffi-1.14.4 chardet-4.0.0 click-7.1.2 cryptography-3.3.1 dnspython-2.1.0
flask-1.1.2 future-0.18.2 idna-2.10 impacket-0.9.22 itsdangerous-1.1.0 ldap3-2.8
.1 ldapdomaindump-0.9.3 pefile-2019.4.18 pillow-8.1.0 pyasn1-0.4.8 pycparser-2.2
0 pycrypto-2.6.1 pycryptodomex-3.9.9 pymssql-2.1.5 pyopenssl-20.0.1 qrcode-6.1 r
equests-2.25.1 six-1.15.0 urllib3-1.26.2
[*] Installing setoolkit to /usr/share/setoolkit..
/home/tc/setoolkit
[*] Creating launcher for setoolkit...
[*] Done. Chmoding +x....
[*] Finished. Run 'setoolkit' to start the Social Engineer Toolkit.
root@box:/home/tc/setoolkit#
```

Figure 59- Successful build of SET tool in Tinycore

I ran “setoolkit” to start the tool and it gave a decoding error.

```
ts, qrcode, pymssql, pycrypto, pillow, pefile, impacket
    Running setup.py install for future ... done
    Running setup.py install for pycrypto ... done
    Running setup.py install for pefile ... done
    Running setup.py install for impacket ... done
Successfully installed Jinja2-2.11.2 MarkupSafe-1.1.1 Werkzeug-1.0.1 certifi-202
0.12.5 cffi-1.14.4 chardet-4.0.0 click-7.1.2 cryptography-3.3.1 dnspython-2.1.0
flask-1.1.2 future-0.18.2 idna-2.10 impacket-0.9.22 itsdangerous-1.1.0 ldap3-2.8
.1 ldapdomaindump-0.9.3 pefile-2019.4.18 pillow-8.1.0 pyasn1-0.4.8 pycparser-2.2
0 pycrypto-2.6.1 pycryptodomex-3.9.9 pymssql-2.1.5 pyopenssl-20.0.1 qrcode-6.1 r
equests-2.25.1 six-1.15.0 urllib3-1.26.2
[*] Installing setoolkit to /usr/share/setoolkit..
/home/tc/setoolkit
[*] Creating launcher for setoolkit...
[*] Done. Chmoding +x....
[*] Finished. Run 'setoolkit' to start the Social Engineer Toolkit.
root@box:/home/tc/setoolkit# setoolkit
Traceback (most recent call last):
  File "./setoolkit", line 59, in <module>
    data = fileopen.read()
  File "/usr/local/lib/python3.6/encodings/ascii.py", line 26, in decode
    return codecs.ascii_decode(input, self.errors)[0]
UnicodeDecodeError: 'ascii' codec can't decode byte 0xe2 in position 6798: ordin
al not in range(128)
root@box:/home/tc/setoolkit#
```

Figure 60- Decoding Error in Tinycore

I checked the locale and decided to convert it to UTF-8

```
root@box:/home/tc/setoolkit# locale
LANG=C
LC_CTYPE="C"
LC_NUMERIC="C"
LC_TIME="C"
LC_COLLATE="C"
LC_MONETARY="C"
LC_MESSAGES="C"
LC_PAPER="C"
LC_NAME="C"
LC_ADDRESS="C"
LC_TELEPHONE="C"
LC_MEASUREMENT="C"
LC_IDENTIFICATION="C"
LC_ALL=
root@box:/home/tc/setoolkit# _
```

Figure 61- Locale in Tinycore

I made this directory.

https://wiki.linuxonlinehelp.eu/index.php/Tinycore_Linux

```
root@box:/home/tc/setoolkit# mkdir -p /usr/lib/locale_
```

Figure 62- Making directory in Tinycore

I checked the locale, it changed to UTF-8, but it still gave some error.

```
root@box:/home/tc/setoolkit# locale
locale: Cannot set LC_CTYPE to default locale: No such file or directory
locale: Cannot set LC_MESSAGES to default locale: No such file or directory
locale: Cannot set LC_ALL to default locale: No such file or directory
LANG=C
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_COLLATE="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
LC_PAPER="en_US.UTF-8"
LC_NAME="en_US.UTF-8"
LC_ADDRESS="en_US.UTF-8"
LC_TELEPHONE="en_US.UTF-8"
LC_MEASUREMENT="en_US.UTF-8"
LC_IDENTIFICATION="en_US.UTF-8"
LC_ALL=en_US.UTF-8
root@box:/home/tc/setoolkit# _
```

Figure 63- Checking locale in Tinycore

I installed the package “getlocale” and then ran this command.

https://serverfault.com/questions/275403/how-do-i-change-my-locale-to-utf-8-in-centos

```
root@box:/home/tc/setoolkit# localedef -c -f UTF-8 -i en_US en_US.UTF-8
root@box:/home/tc/setoolkit#
```

Figure 64- Changing locale to UTF-8 in Tinycore

Then I checked the locale and it looked all good

```
root@box:/home/tc/setoolkit# locale
LANG=C
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_COLLATE="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
LC_PAPER="en_US.UTF-8"
LC_NAME="en_US.UTF-8"
LC_ADDRESS="en_US.UTF-8"
LC_TELEPHONE="en_US.UTF-8"
LC_MEASUREMENT="en_US.UTF-8"
LC_IDENTIFICATION="en_US.UTF-8"
LC_ALL=en_US.UTF-8
root@box:/home/tc/setoolkit#
```

Figure 65- Locale in Tinycore

Then I started the SET tool and it started without giving any errors.

```
[---]        Follow me on Twitter: @HackingDave        [---]
[---]       Homepage: https://www.trustedsec.com       [---]
        Welcome to the Social-Engineer Toolkit (SET).
         The one stop shop for all of your SE needs.

   The Social-Engineer Toolkit is a product of TrustedSec.

           Visit: https://www.trustedsec.com

   It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!


 Select from the menu:

   1) Social-Engineering Attacks
   2) Penetration Testing (Fast-Track)
   3) Third Party Modules
   4) Update the Social-Engineer Toolkit
   5) Update SET configuration
   6) Help, Credits, and About

  99) Exit the Social-Engineer Toolkit

set> _
```

Figure 66- Starting SET tool in Tinycore

Porteus

The porteus linux system uses the USM (Unified Slackware Package Manager). Using this package manager, the user can download and install packages from various Slackware repositories.

I installed the Porteus linux system on the VMware workstation and after starting it I opened the terminal and typed the command “usm -u all” which will download all the libraries required and

all other dependencies to keep the system up and running. But usm is unstable and it is even removed from the latest version of Porteus (version 5, rc2). It gave an error that it was unable to decompress some file while downloading the libraries.

I decided to install packages directly from the Slackware package manager using the getpkg and install pkg commands.

https://www.linux.com/training-tutorials/intro-slackware-package-management/

Now it is time to install the security tools.

SET

To download SET tool, I need git package for cloning, so I installed it.

```
root@porteus:/home/guest# getpkg git
Checking that mirror is online ...

--2021-01-23 23:37:55--  https://dfw.mirror.rackspace.com/slackware//slackware64
-current/FILELIST.TXT
Resolving dfw.mirror.rackspace.com (dfw.mirror.rackspace.com)... 74.205.112.120
Connecting to dfw.mirror.rackspace.com (dfw.mirror.rackspace.com)|74.205.112.120
|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1385273 (1.3M) [text/plain]
Saving to: '/tmp/getpkg/FILELIST.TXT'

FILELIST.TXT        100%[===================>]   1.32M   799KB/s    in 1.7s

2021-01-23 23:37:58 (799 KB/s) - '/tmp/getpkg/FILELIST.TXT' saved [1385273/13852
73]


[OK] git-2.30.0-x86_64-2.txz

Please enter a directory to download the packages to.
>
```

Figure 67- Installing git on Porteus

After installing git, I installed python as I need it after cloning the SET tool.

```
root@porteus:/home/guest# getpkg python3

[OK] python3-3.9.1-x86_64-2.txz

Please enter a directory to download the packages to.
> Desktop

Downloading: python3-3.9.1-x86_64-2.txz   7%
```

Figure 68- Installing python on Porteus

Usually pip3 comes with python, I checked but it did not come so I installed it manually.

```
root@porteus:/home/guest# curl https://bootstrap.pypa.io/get-pip.py -o get-pip.p
y
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1883k  100 1883k    0     0  3867k      0 --:--:-- --:--:-- --:--:-- 3867k
root@porteus:/home/guest#
```

Figure 69- Installing pip on Porteus

Here, it was unable to find glib, so I searched for this glibc package on the Slackware repository and found it. Then I installed it.

```
root@porteus:/home/guest# python3 get-pip.py
python3: /lib64/libc.so.6: version `GLIBC_2.32' not found (required by /usr/lib6
4/libpython3.9.so.1.0)
root@porteus:/home/guest#
```

Figure 70- Installing python on Porteus

Installation of GLIBC

```
root@porteus:/home/guest# getpkg glibc

[OK] glibc-2.32-x86_64-1.txz

Please enter a directory to download the packages to.
> Desktop

Downloading: glibc-2.32-x86_64-1.txz   DONE
Converting glibc-2.32-x86_64-1.txz ...
Verifying package glibc-2.32-x86_64-1.txz.
```

Figure 71- Installing glibc in Porteus

Pip3 was successfully installed.

```
root@porteus:/home/guest# python3 get-pip.py
Collecting pip
  Downloading pip-21.0-py3-none-any.whl (1.5 MB)
     |                                | 1.5 MB 2.3 MB/s
Collecting setuptools
  Downloading setuptools-52.0.0-py3-none-any.whl (784 kB)
     |                                | 784 kB 2.3 MB/s
Collecting wheel
  Downloading wheel-0.36.2-py2.py3-none-any.whl (35 kB)
Installing collected packages: wheel, setuptools, pip
Successfully installed pip-21.0 setuptools-52.0.0 wheel-0.36.2
```

Figure 72- Installing pip3 in Porteus

After installing pip3, I cloned the link and it worked as shown below.

```
root@porteus:/home/guest# git clone https://github.com/trustedsec/social-enginee
r-toolkit/ setoolkit/
Cloning into 'setoolkit'...
remote: Enumerating objects: 110045, done.
remote: Total 110045 (delta 0), reused 0 (delta 0), pack-reused 110045
Receiving objects: 100% (110045/110045), 175.21 MiB | 27.64 MiB/s, done.
Resolving deltas: 100% (68248/68248), done.
root@porteus:/home/guest#
```

Figure 73- git cloning in Porteus

Then I ran the command “pip3 install –r requirements.txt” but it gave an error about the C compiler.

```
  configure: error: no acceptable C compiler found in $PATH
  See `config.log' for more details
  Traceback (most recent call last):
    File "<string>", line 1, in <module>
    File "/tmp/pip-install-v71uep5k/pycrypto_b80883d92a834f02a1d02d265f0f2d79/se
tup.py", line 456, in <module>
      core.setup(**kw)
    File "/usr/lib64/python3.9/distutils/core.py", line 148, in setup
      dist.run_commands()
    File "/usr/lib64/python3.9/distutils/dist.py", line 966, in run_commands
      self.run_command(cmd)
    File "/usr/lib64/python3.9/distutils/dist.py", line 985, in run_command
      cmd_obj.run()
    File "/usr/lib64/python3.9/site-packages/wheel/bdist_wheel.py", line 299, in
 run
```

Figure 74- C compiler error in Porteus

It gave an error related to the C compiler so installed gcc as it compiles the C code.

```
root@porteus:/home/guest# getpkg gcc

[OK] gcc-10.2.0-x86_64-3.txz

Please enter a directory to download the packages to.
> Desktop

Downloading: gcc-10.2.0-x86_64-3.txz  26%
```

Figure 75- Installing gcc in Porteus

After installing gcc it still gave an error that the compiler is not working.

```
    Skipping optional fixer: ws_comma
    running build_ext
    running build_configure
    checking for gcc... gcc
    checking whether the C compiler works... no
    configure: error: in `/tmp/pip-install-2w2bsafe/pycrypto_bdc26db77a6b4678bc
20e7dc4e62141':
    configure: error: C compiler cannot create executables
    See `config.log' for more details
    Traceback (most recent call last):
      File "<string>", line 1, in <module>
      File "/tmp/pip-install-2w2bsafe/pycrypto_bdc26db77a6b4678bca20e7dc4e62141
setup.py", line 456, in <module>
        core.setup(**kw)
      File "/usr/lib64/python3.9/distutils/core.py", line 148, in setup
        dist.run_commands()
      File "/usr/lib64/python3.9/distutils/dist.py", line 966, in run_commands
        self.run_command(cmd)
      File "/usr/lib64/python3.9/distutils/dist.py", line 985, in run_command
        cmd_obj.run()
      File "/usr/lib64/python3.9/site-packages/setuptools/command/install.py",
ine 61, in run
```

Figure 76- C compiler error in Porteus

To find the problems with the compiler, I created a test file and tried to run it with gcc.

https://linuxize.com/post/how-to-install-gcc-on-ubuntu-20-04/

```
#include <stdio.h>

int main() {
        printf("Hello");
        return 0;
}
~
~
~
```

Figure 77- Test C program for dependencies

It gave an error related to the libisl file.

```
root@porteus:/home/guest# gcc hello.c -o hello
/usr/libexec/gcc/x86_64-slackware-linux/10.2.0/cc1: error while loading shared l
ibraries: libisl.so.23: cannot open shared object file: No such file or director
y
root@porteus:/home/guest#
```

Figure 78- libisl missing package error in Porteus

I searched for the libisl file and found a package and installed it.

```
Please enter a directory to download the packages to.
> Desktop

Downloading: isl-0.23-x86_64-2.txz   DONE
Converting isl-0.23-x86_64-2.txz ...
Verifying package isl-0.23-x86_64-2.txz.
Installing package isl-0.23-x86_64-2.txz:
PACKAGE DESCRIPTION:
# isl (Integer Set Library)
#
# isl is a thread-safe C library for manipulating sets and relations
# of integer points bounded by affine constraints. The descriptions of
# the sets and relations may involve both parameters and existentially
# quantified variables. All computations are performed in exact integer
# arithmetic using GMP.
#
# Homepage: http://isl.gforge.inria.fr
#
Executing install script for isl-0.23-x86_64-2.txz.
Package isl-0.23-x86_64-2.txz installed.
Creating /home/guest/Desktop/isl-0.23-x86_64-2.xzm
```

Figure 79- Installing libisl in Porteus

I tried executing the test file with gcc again and it gave an error.

```
root@porteus:/home/guest# gcc hello.c -o hello
gcc: fatal error: cannot execute 'as': execvp: No such file or directory
compilation terminated.
root@porteus:/home/guest#
```

Figure 80- Missing binutils library error in Porteus

This 'as' error is due to a missing binutils library, so I downloaded and installed it.

https://stackoverflow.com/questions/56801179/g-gcc-9-1-0-fatal-error-cannot-execute-as-execvp-no-such-file-or-directo

```
root@porteus:/home/guest# getpkg binutils

[OK] binutils-2.35.1-x86_64-2.txz

Please enter a directory to download the packages to.
> Desktop

Downloading: binutils-2.35.1-x86_64-2.txz  50%
```

Figure 81- Installing binutils in Porteus

```
root@porteus:/home/guest# gcc hello.c -o hello
root@porteus:/home/guest#
```

Figure 82- C test program in Porteus

The gcc ran successfully on the test file so I tried with the SET tool again. This time it gave an error about /lib/cpp files, so I installed the gcc-g++ which is a compiler for C++.

https://packages.slackware.com/

```
    checking how to run the C preprocessor... /lib/cpp
    configure: error: in `/tmp/pip-install-x6l24gua/pycrypto_1b2e16a0c2e94cfba2b
33acf4698aa84':
    configure: error: C preprocessor "/lib/cpp" fails sanity check
    See `config.log' for more details
    Traceback (most recent call last):
      File "<string>", line 1, in <module>
      File "/tmp/pip-install-x6l24gua/pycrypto_1b2e16a0c2e94cfba2b33acf4698aa84/
setup.py", line 456, in <module>
        core.setup(**kw)
      File "/usr/lib64/python3.9/distutils/core.py", line 148, in setup
        dist.run_commands()
      File "/usr/lib64/python3.9/distutils/dist.py", line 966, in run_commands
        self.run_command(cmd)
      File "/usr/lib64/python3.9/distutils/dist.py", line 985, in run_command
        cmd_obj.run()
      File "/usr/lib64/python3.9/site-packages/setuptools/command/install.py", l
ine 61, in run
        return orig.install.run(self)
      File "/usr/lib64/python3.9/distutils/command/install.py", line 546, in run
        self.run_command('build')
      File "/usr/lib64/python3.9/distutils/cmd.py", line 313, in run_command
        self.distribution.run_command(command)
```

Figure 83- C compiler error in Porteus

Installation of gcc-g++

```
root@porteus:/home/guest# getpkg gcc-g++

[OK] gcc-g++-10.2.0-x86_64-3.txz

Please enter a directory to download the packages to.
> Desktop/

Downloading: gcc-g++-10.2.0-x86_64-3.txz  29%
```

Figure 84- Installing gcc-g++ in Porteus

It still gave errors, so I installed the libffi package and the kernel headers.

```
Downloading: libffi-3.3-x86_64-2.txz   DONE
Converting libffi-3.3-x86_64-2.txz ...
Verifying package libffi-3.3-x86_64-2.txz.
Installing package libffi-3.3-x86_64-2.txz:
PACKAGE DESCRIPTION:
# libffi (A Portable Foreign Function Interface Library)
#
# FFI stands for Foreign Function Interface. A foreign function
# interface is the popular name for the interface that allows code
# written in one language to call code written in another language.
# The libffi library really only provides the lowest, machine dependent
# layer of a fully featured foreign function interface.
#
# Homepage: https://sourceware.org/libffi/
#
Executing install script for libffi-3.3-x86_64-2.txz.
Package libffi-3.3-x86_64-2.txz installed.
Creating /home/guest/Desktop/libffi-3.3-x86_64-2.xzm


 Processing finished.
Your files are in: Desktop/
```

Figure 85- Installing libffi in Porteus

```
root@porteus:/home/guest# getpkg kernel-headers

[OK] kernel-headers-5.10.10-x86-1.txz

Please enter a directory to download the packages to.
> Desktop

Downloading: kernel-headers-5.10.10-x86-1.txz   DONE
Converting kernel-headers-5.10.10-x86-1.txz ...
Verifying package kernel-headers-5.10.10-x86-1.txz.
Installing package kernel-headers-5.10.10-x86-1.txz:
PACKAGE DESCRIPTION:
# kernel-headers (Linux kernel include files)
#
# These are the include files from the Linux kernel.
#
# You'll need these to compile most system software for Linux.
#
```

Figure 86- Installing kernel-headers in Porteus

Finally, command ran successfully and then I ran the setup.py command and opened the SET tool.

```
        Welcome to the Social-Engineer Toolkit (SET).
         The one stop shop for all of your SE needs.

   The Social-Engineer Toolkit is a product of TrustedSec.

           Visit: https://www.trustedsec.com

   It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!


 Select from the menu:

   1) Social-Engineering Attacks
   2) Penetration Testing (Fast-Track)
   3) Third Party Modules
   4) Update the Social-Engineer Toolkit
   5) Update SET configuration
   6) Help, Credits, and About

  99) Exit the Social-Engineer Toolkit

set>
```

Figure 87- Successful build of SET tool in Porteus

NMAP

Nmap is already available on Porteus and it can be installed using the getpkg package manager.

```
root@porteus:/home/guest# getpkg nmap

[OK] nmap-7.91-x86_64-2.txz

Please enter a directory to download the packages to.
> Desktop

Downloading: nmap-7.91-x86_64-2.txz  31%
```

Figure 88- Installing nmap in Porteus

SQLMAP

Sqlmap is not available on Porteus and can be installed through the git cloning process.

```
root@porteus:/home/guest# git clone https://github.com/sqlmapproject/sqlmap.git
sqlmap-dev
Cloning into 'sqlmap-dev'...
remote: Enumerating objects: 106, done.
remote: Counting objects: 100% (106/106), done.
remote: Compressing objects: 100% (80/80), done.
remote: Total 77941 (delta 53), reused 50 (delta 25), pack-reused 77835
Receiving objects: 100% (77941/77941), 73.66 MiB | 24.55 MiB/s, done.
Resolving deltas: 100% (61670/61670), done.
root@porteus:/home/guest#
```

Figure 89- git cloning in Porteus

After git cloning, I started the sqlmap using python and it worked successfully.

```
root@porteus:/home/guest# cd sqlmap-dev/
root@porteus:/home/guest/sqlmap-dev# python sqlmap.py -hh
        ___
       __H__
 ___ ___["]_____ ___ ___  {1.5.1.40#dev}
|_ -| . [)]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

Usage: python sqlmap.py [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
    At least one of these options has to be provided to define the
    target(s)

    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -d DIRECT           Connection string for direct database connection
```

Figure 90- Launching sqlmap in Porteus

METASPLOIT FRAMEWORK

It is not available on the Porteus Linux, so I downloaded it using wget.

```
root@porteus:~# wget https://downloads.metasploit.com/data/releases/metasploit-l
atest-linux-x64-installer.run
--2021-01-24 01:07:11--  https://downloads.metasploit.com/data/releases/metasplo
it-latest-linux-x64-installer.run
Resolving downloads.metasploit.com (downloads.metasploit.com)... 104.92.177.250
Connecting to downloads.metasploit.com (downloads.metasploit.com)|104.92.177.250
|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 204571447 (195M) [text/plain]
Saving to: 'metasploit-latest-linux-x64-installer.run'

metasploit-latest-l 100%[===================>] 195.09M  34.7MB/s    in 6.3s

2021-01-24 01:07:18 (30.8 MB/s) - 'metasploit-latest-linux-x64-installer.run' sa
ved [204571447/204571447]

root@porteus:~#
```

Figure 91- Downloading Metasploit framework in Porteus

Here, I changed the mode of the installer to be executable.

```
root@porteus:~# chmod +x metasploit-latest-linux-x64-installer.run
root@porteus:~#
```

Figure 92- Changing the installer mode of Metasploit in Porteus

Metasploit framework started successfully.

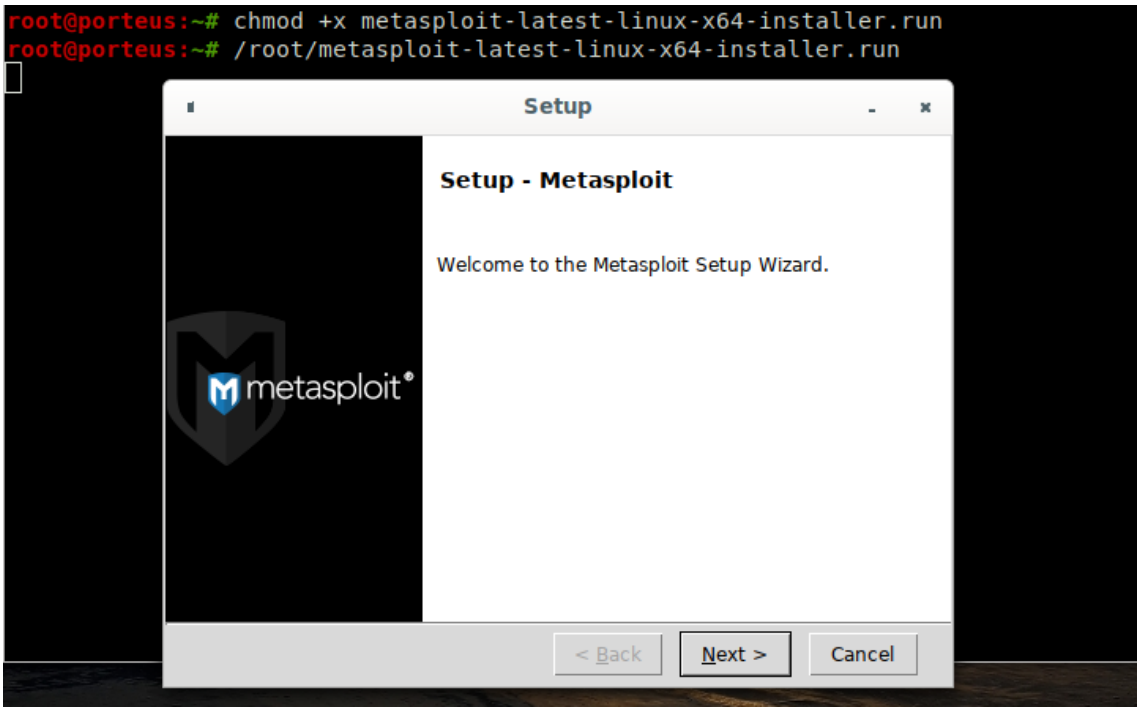


Figure 93- Starting Metasplot framework in Porteus

WIRESHARK

https://gist.github.com/syneart/2d30c075c140624b1e150c8ea318a978

To install/build Wireshark on Porteus, I used this script from GitHub.

```
root@porteus:~/Desktop# wget -O - https://gist.githubusercontent.com/syneart/2d3
0c075c140624b1e150c8ea318a978/raw/build_wireshark.sh | sh
--2021-01-23 21:04:38--  https://gist.githubusercontent.com/syneart/2d30c075c140
624b1e150c8ea318a978/raw/build_wireshark.sh
Resolving gist.githubusercontent.com (gist.githubusercontent.com)... 151.101.128
.133, 151.101.64.133, 151.101.192.133, ...
Connecting to gist.githubusercontent.com (gist.githubusercontent.com)|151.101.12
8.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1586 (1.5K) [text/plain]
Saving to: 'STDOUT'

-                   100%[===================>]   1.55K  --.-KB/s    in 0s

2021-01-23 21:04:39 (22.2 MB/s) - written to stdout [1586/1586]

sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
```

Figure 94- Running script for installing Wireshark in Porteus

The script was unable to run because it could find the package make so I had to install it after finding it from the Slackware repository.

```
+-------------------------+
| NOTICE! MAKE NOT FOUND |
+-------------------------+
The 'make' program is not installed on your system. In order
to compile any software, you will need to download and/or
activate the porteus 'devel' module, which contains make, gcc
and other programs that are necessary to compile source code.

Please visit http://dl.porteus.org/x86_64/Porteus-v5.0/modules to download
the development module for your version and architecture.

Press enter to quit this program.  If you are running a build
script, it may continue and state additional errors.
```

Figure 95- Missing library make in Porteuss

Installation of package “make.”

```
root@porteus:~/Desktop# getpkg make

[OK] make-4.2.1-x86_64-8.txz

Please enter a directory to download the packages to.
> Desktop

Downloading: make-4.2.1-x86_64-8.txz   DONE
Converting make-4.2.1-x86_64-8.txz ...
Verifying package make-4.2.1-x86_64-8.txz.
Installing package make-4.2.1-x86_64-8.txz:
PACKAGE DESCRIPTION:
# make (GNU make utility to maintain groups of programs)
#
# This is the GNU implementation of make, which was written by Richard
# Stallman and Roland McGrath. The purpose of the make utility is to
# determine automatically which pieces of a large program need to be
# recompiled, and issue the commands to recompile them.
#
# This is needed to compile just about any major C program, including
# the Linux kernel.
```

Figure 96- Installing make package in Porteus

Then it showed a dependency of the missing package libguile so I found the package from the Slackware repository and installed it

```
Cloning into '/root/wireshark'...
remote: Enumerating objects: 76, done.
remote: Counting objects: 100% (76/76), done.
remote: Compressing objects: 100% (38/38), done.
fatal: write error: No space left on device55 MiB | 17.02 MiB/s
fatal: index-pack failed
sh: line 48: cd: /root/wireshark: No such file or directory
mkdir: cannot create directory 'build': File exists
sh: line 51: cmake: command not found
make: error while loading shared libraries: libguile-3.0.so.1: cannot open share
d object file: No such file or directory
```

Figure 97- Missing libguile package in Porteus

Installation of package “guile”

```
root@porteus:~# getpkg guile

[OK] guile-3.0.5-x86_64-2.txz

Please enter a directory to download the packages to.
> Desktop

Downloading: guile-3.0.5-x86_64-2.txz  48%
```

Figure 98- Installing guile package in Porteus

Then it showed a dependency of the missing package libgc so I found the package from the Slackware repository and installed it

```
Cloning into '/root/wireshark'...
remote: Enumerating objects: 76, done.
remote: Counting objects: 100% (76/76), done.
remote: Compressing objects: 100% (38/38), done.
fatal: write error: No space left on device89 MiB | 14.88 MiB/s
sh: line 48: cd: /root/wireshark: No such file or directory
sh: line 51: cmake: command not found
make: error while loading shared libraries: libgc.so.1: cannot open shared objec
t file: No such file or directory
```

Figure 99- Missing libgc package in Porteus

Installation of package “gc”

```
root@porteus:~# getpkg gc
3600 seconds reached. Updating FILELIST.TXT
--2021-01-23 21:15:39--  https://dfw.mirror.rackspace.com/slackware//slackware64
-current/FILELIST.TXT
Resolving dfw.mirror.rackspace.com (dfw.mirror.rackspace.com)... 74.205.112.120
Connecting to dfw.mirror.rackspace.com (dfw.mirror.rackspace.com)|74.205.112.120
|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1385049 (1.3M) [text/plain]
Saving to: ‘/tmp/getpkg/FILELIST.TXT’

FILELIST.TXT        100%[===================>]   1.32M   757KB/s    in 1.8s

2021-01-23 21:15:41 (757 KB/s) - ‘/tmp/getpkg/FILELIST.TXT’ saved [1385049/13850
49]


[OK] gc-8.0.4-x86_64-2.txz

Please enter a directory to download the packages to.
> Desktop/

Downloading: gc-8.0.4-x86_64-2.txz   DONE
```

Figure 100- Installing gc package in Porteus

Then it showed a dependency of the missing package GLIBC, so I found the package from the Slackware repository and installed it.

```
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
git: /lib64/libc.so.6: version `GLIBC_2.32' not found (required by git)
sh: line 48: cd: /root/wireshark: No such file or directory
mkdir: cannot create directory 'build': File exists
sh: line 51: cmake: command not found
make: /lib64/libc.so.6: version `GLIBC_2.32' not found (required by /usr/lib64/l
ibguile-3.0.so.1)
make: /lib64/libc.so.6: version `GLIBC_2.32' not found (required by /usr/lib64/l
ibgc.so.1)
```

Figure 101- Missing glibc package in Porteus

Installation of package "glibc"

```
root@porteus:/home/guest# getpkg glibc

[OK] glibc-2.32-x86_64-1.txz

Please enter a directory to download the packages to.
> Desktop/

Downloading: glibc-2.32-x86_64-1.txz  12%
```

Figure 102- Installing glibc package in Porteus

It was unable to find cmake so I installed it.

```
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
sudo: apt-get: command not found
Cloning into '/root/wireshark'...
remote: Enumerating objects: 179, done.
remote: Counting objects: 100% (179/179), done.
remote: Compressing objects: 100% (127/127), done.
remote: Total 623383 (delta 121), reused 110 (delta 52), pack-reused 623204
Receiving objects: 100% (623383/623383), 814.98 MiB | 19.33 MiB/s, done.
Resolving deltas: 100% (515910/515910), done.
Updating files: 100% (6164/6164), done.
sh: line 51: cmake: command not found
make: *** No targets specified and no makefile found.  Stop.
```

Figure 103- Missing cmake package in Porteus

Installation of package “cmake”

```
root@porteus:/home/guest# getpkg cmake

[OK] cmake-3.19.3-x86_64-2.txz

Please enter a directory to download the packages to.
> Desktop/

Downloading: cmake-3.19.3-x86_64-2.txz  36%
```

Figure 104- Installing cmake package in Porteus

After installing cmake, it showed some memory issues, so I increased the RAM to 16GB as porteus uses only the RAM. Then it showed some C compiler issues as shown below.

```
-- The C compiler identification is unknown
-- The CXX compiler identification is unknown
CMake Error at CMakeLists.txt:32 (project):
  No CMAKE_C_COMPILER could be found.

  Tell CMake where to find the compiler by setting either the environment
  variable "CC" or the CMake cache entry CMAKE_C_COMPILER to the full path to
  the compiler, or to the compiler name if it is in the PATH.


CMake Error at CMakeLists.txt:32 (project):
  No CMAKE_CXX_COMPILER could be found.

  Tell CMake where to find the compiler by setting either the environment
  variable "CXX" or the CMake cache entry CMAKE_CXX_COMPILER to the full path
  to the compiler, or to the compiler name if it is in the PATH.


-- Configuring incomplete, errors occurred!
See also "/root/wireshark/build/CMakeFiles/CMakeOutput.log".
See also "/root/wireshark/build/CMakeFiles/CMakeError.log".
make: *** No targets specified and no makefile found.  Stop.
```

Figure 105- C compiler error in Porteus

I installed all the packages that are required for the C compiler to run properly. They include gcc, gcc-++, binutils, libffi, isl, kernel-headers, glib2, libgcrypt. After installing the required compiler packages, it showed a missing CARES library. I was unable to find CARES from the official Slackware repository, so I took it from an online link, copied it in VMshared, accessed VMshared from the Porteus system and installed it.

https://slackware.pkgs.org/14.1/slackonly-x86_64/c-ares-1.10.0-x86_64-1_slack.txz.html

```
-- Could NOT find PkgConfig (missing: PKG_CONFIG_EXECUTABLE)
-- Checking for one of the modules 'glib-2.0'
-- Found GLIB2: /usr/lib64/libglib-2.0.so (found suitable version "2.66.4", mini
mum required is "2.36.0")
-- Found GMODULE2: /usr/lib64/libgmodule-2.0.so
-- Found GTHREAD2: /usr/lib64/libgthread-2.0.so
-- Found GCRYPT: /usr/lib64/libgcrypt.so (found suitable version "1.9.0", minimu
m required is "1.5.0")
CMake Error at /usr/share/cmake-3.19/Modules/FindPackageHandleStandardArgs.cmake
:218 (message):
  Could NOT find CARES (missing: CARES_LIBRARY CARES_INCLUDE_DIR) (Required
  is at least version "1.5.0")
Call Stack (most recent call first):
  /usr/share/cmake-3.19/Modules/FindPackageHandleStandardArgs.cmake:582 (_FPHSA_
FAILURE_MESSAGE)
  cmake/modules/FindCARES.cmake:36 (FIND_PACKAGE_HANDLE_STANDARD_ARGS)
  CMakeLists.txt:1050 (find_package)
```

Figure 106- Missing C-ares package in Porteus

After installing C-ares, it showed other missing packages and I found them from the Slackware repository and installed them. The packages that I installed include libcap, libpcap, qt5, Qt5-webkit and libssh. Then it showed a missing libcui18n package and I installed it. It is named as “icu4c” in the Slackware repository.

https://packages.slackware.com/

```
[  0%] Generating wireshark_zh_CN.qm
/usr/lib64/qt5/bin/lrelease: error while loading shared libraries: libicui18n.so
.68: cannot open shared object file: No such file or directory
make[2]: *** [ui/qt/CMakeFiles/qtui_autogen.dir/build.make:131: ui/qt/wireshark_
zh_CN.qm] Error 127
make[1]: *** [CMakeFiles/Makefile2:11488: ui/qt/CMakeFiles/qtui_autogen.dir/all]
 Error 2
make[1]: *** Waiting for unfinished jobs....
[  0%] Building C object CMakeFiles/capture_opts.dir/capture_opts.c.o
[  1%] Building C object CMakeFiles/shark_common.dir/cfile.c.o
[  1%] Building C object CMakeFiles/shark_common.dir/extcap.c.o
We are not tagged.
version.h unchanged.
[  1%] Built target version
[  1%] Building C object CMakeFiles/shark_common.dir/file_packet_provider.c.o
[  1%] Building C object CMakeFiles/shark_common.dir/extcap_parser.c.o
[  1%] Building C object CMakeFiles/shark_common.dir/frame_tvbuff.c.o
```

Figure 107- Missing package libicui18n in Porteus

For the error shown below I installed libgpg-error from the Slackware repository.

```
[  1%] Building C object wsutil/CMakeFiles/wsutil.dir/crc5.c.o
[  1%] Building C object CMakeFiles/shark_common.dir/sync_pipe_write.c.o
[  1%] Building C object wsutil/CMakeFiles/wsutil.dir/crc6.c.o
[  1%] Building C object wsutil/CMakeFiles/wsutil.dir/crc7.c.o
[  1%] Built target shark_common
[  1%] Building C object wsutil/CMakeFiles/wsutil.dir/crc8.c.o
[  1%] Building C object wsutil/CMakeFiles/wsutil.dir/crc11.c.o
[  1%] Building C object wsutil/CMakeFiles/wsutil.dir/curve25519.c.o
[  1%] Building C object wsutil/CMakeFiles/wsutil.dir/dot11decrypt_wep.c.o
In file included from /root/wireshark/wsutil/wsgcrypt.h:24,
                 from /root/wireshark/wsutil/curve25519.h:21,
                 from /root/wireshark/wsutil/curve25519.c:13:
/usr/include/gcrypt.h:30:10: fatal error: gpg-error.h: No such file or director
   30 | #include <gpg-error.h>
      |          ^~~~~~~~~~~~~
compilation terminated.
make[2]: *** [wsutil/CMakeFiles/wsutil.dir/build.make:290: wsutil/CMakeFiles/ws
til.dir/curve25519.c.o] Error 1
make[2]: *** Waiting for unfinished jobs....
make[1]: *** [CMakeFiles/Makefile2:11380: wsutil/CMakeFiles/wsutil.dir/all] Err
r 2
[  1%] Built target qtui_autogen
make: *** [Makefile:160: all] Error 2
```

Figure 108- Missing package gpg in Porteus

For the error shown below I installed zlib from the Slackware repository.

```
[ 48%] Building C object epan/dissectors/CMakeFiles/dissectors.dir/packet-rtp-mi
di.c.o
[ 48%] Building C object epan/dissectors/CMakeFiles/dissectors.dir/packet-rtp.c.
o
[ 48%] Building C object epan/dissectors/CMakeFiles/dissectors.dir/packet-rtp-ed
137.c.o
[ 48%] Building C object epan/dissectors/CMakeFiles/dissectors.dir/packet-rtppro
xy.c.o
[ 48%] Building C object epan/dissectors/CMakeFiles/dissectors.dir/packet-rtps.c
.o
/root/wireshark/epan/dissectors/packet-rtps.c:52:10: fatal error: zlib.h: No suc
h file or directory
   52 | #include "zlib.h"
      |          ^~~~~~~~
compilation terminated.
make[2]: *** [epan/dissectors/CMakeFiles/dissectors.dir/build.make:15254: epan/d
issectors/CMakeFiles/dissectors.dir/packet-rtps.c.o] Error 1
make[2]: *** Waiting for unfinished jobs....
make[1]: *** [CMakeFiles/Makefile2:5178: epan/dissectors/CMakeFiles/dissectors.d
ir/all] Error 2
make: *** [Makefile:160: all] Error 2
```

Figure 109- Missing package zlib in Porteus

For the error shown below I installed Ffex, doxygen, m4, autoconf from the Slackware repository.

```
flex: fatal internal error, exec of /usr/bin/m4 failed
flex: fatal internal error, exec of /usr/bin/m4 failed
make[2]: *** [wiretap/CMakeFiles/wiretap.dir/build.make:102: wiretap/k12text.c]
Error 1
make[2]: *** Deleting file 'wiretap/k12text.c'
make[1]: *** [CMakeFiles/Makefile2:11298: wiretap/CMakeFiles/wiretap.dir/all] Er
ror 2
make[1]: *** Waiting for unfinished jobs....
[  5%] Building C object epan/dissectors/CMakeFiles/dissectors.dir/packet-9p.c.o
[  5%] Building C object epan/wmem/CMakeFiles/wmem.dir/wmem_allocator_block_fast
.c.o
[  5%] Building C object epan/wmem/CMakeFiles/wmem.dir/wmem_allocator_simple.c.o
[  5%] Building C object epan/wmem/CMakeFiles/wmem.dir/wmem_allocator_strict.c.o
```

Figure 110- Missing package m4 in Porteus

For the error shown below I installed libglvnd from the Slackware repository.

```
                 from /usr/include/qt5/QtPrintSupport/QtPrintSupportDepends:4,
                 from /usr/include/qt5/QtPrintSupport/QtPrintSupport:3,
                 from /root/wireshark/ui/qt/widgets/qcustomplot.h:81,
                 from /root/wireshark/build/ui/qt/qtui_autogen/EWIEGA46WW/../../
../../../ui/qt/lte_rlc_graph_dialog.h:16,
                 from /root/wireshark/build/ui/qt/qtui_autogen/EWIEGA46WW/moc_lt
e_rlc_graph_dialog.cpp:10,
                 from /root/wireshark/build/ui/qt/qtui_autogen/mocs_compilation.
cpp:62:
/usr/include/qt5/QtGui/qopengl.h:141:13: fatal error: GL/gl.h: No such file or d
irectory
  141 | #   include <GL/gl.h>
      |             ^~~~~~~~~
compilation terminated.
make[2]: *** [ui/qt/CMakeFiles/qtui.dir/build.make:374: ui/qt/CMakeFiles/qtui.di
r/qtui_autogen/mocs_compilation.cpp.o] Error 1
make[1]: *** [CMakeFiles/Makefile2:11518: ui/qt/CMakeFiles/qtui.dir/all] Error 2
make[1]: *** Waiting for unfinished jobs....
[ 19%] Building C object epan/dissectors/CMakeFiles/dissectors.dir/packet-bthci_
```

Figure 111- Missing package gL in Porteus

The script successfully ran.

```
[100%] Building C object plugins/epan/wimax/CMakeFiles/wimax.dir/msg_res_cmd.c.o
[100%] Building C object plugins/epan/wimax/CMakeFiles/wimax.dir/msg_rep.c.o
[100%] Building C object plugins/epan/wimax/CMakeFiles/wimax.dir/msg_clk_cmp.c.o
[100%] Building C object plugins/epan/wimax/CMakeFiles/wimax.dir/msg_dsx_rvd.c.o
[100%] Building C object plugins/epan/wimax/CMakeFiles/wimax.dir/wimax_harq_map_
decoder.c.o
[100%] Building C object plugins/epan/wimax/CMakeFiles/wimax.dir/wimax_compact_d
lmap_ie_decoder.c.o
[100%] Building C object plugins/epan/wimax/CMakeFiles/wimax.dir/wimax_compact_u
lmap_ie_decoder.c.o
[100%] Building C object plugins/epan/wimax/CMakeFiles/wimax.dir/wimax_utils.c.o
[100%] Building C object plugins/epan/wimax/CMakeFiles/wimax.dir/crc.c.o
[100%] Building C object plugins/epan/wimax/CMakeFiles/wimax.dir/crc_data.c.o
[100%] Building C object plugins/epan/wimax/CMakeFiles/wimax.dir/wimax_tlv.c.o
[100%] Linking C shared module ../../../run/plugins/3.5/epan/wimax.so
[100%] Built target wimax
\nBuild Success!
You can execute the Wireshark by command "sudo ./wireshark"
at "/root/wireshark/build/run"
```

Figure 112- Successful build of Wireshark in Porteus

I tried to run Wireshark, but it gave an error as shown below, so I set “Export QT_DEBUG_PLUGINS=1” in the root to see the missing library. It showed “libxcb-xccm” which is “xcb-util-wm” in the Slackware.

```
00:47:01.392     Main Warn Got keys from plugin meta data ("xcb")
00:47:01.392     Main Warn QFactoryLoader::QFactoryLoader() checking directory p
ath "/root/wireshark/build/run/platforms" ...
00:47:01.392     Main Warn Cannot load library /usr/lib64/qt5/plugins/platforms/
libqxcb.so: (libxcb-icccm.so.4: cannot open shared object file: No such file or
directory)
00:47:01.393     Main Warn QLibraryPrivate::loadPlugin failed on "/usr/lib64/qt5
/plugins/platforms/libqxcb.so" : "Cannot load library /usr/lib64/qt5/plugins/pla
tforms/libqxcb.so: (libxcb-icccm.so.4: cannot open shared object file: No such f
ile or directory)"
00:47:01.393     Main Info Could not load the Qt platform plugin "xcb" in "" eve
n though it was found.
Aborted
```

Figure 113- Missing package libxcb in Porteus

Wireshark successfully started!

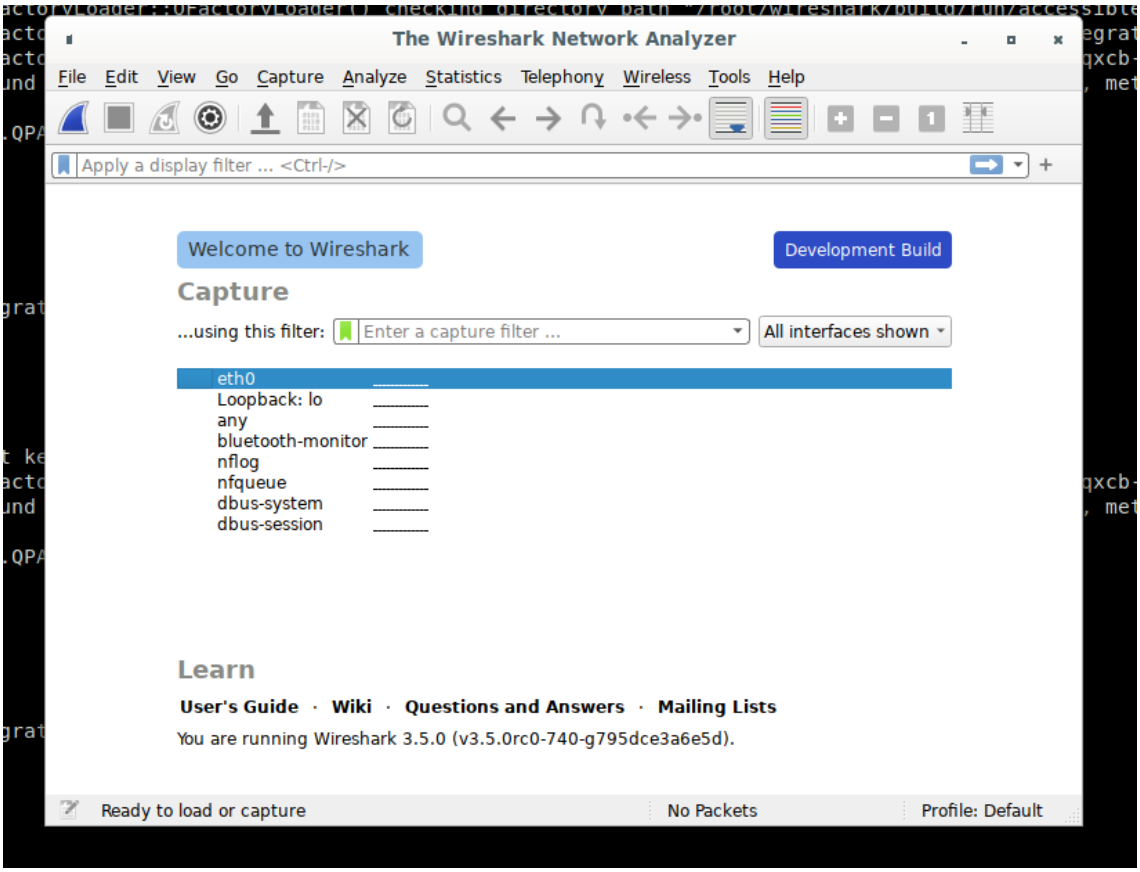


Figure 114- Wireshark starting in Porteus

Silver Blue

After installing fedora silver blue on the VMware workstation, the next step is to start installing the security tools. Packages can be installed in Silver blue using rpm-ostree or dnf. Dnf does not work in some versions but rpm-ostree always works.

NMAP

It is already available on Silverblue and can be installed using the rpm-ostree.

```
[zainabi@localhost ~]$ rpm-ostree install nmap
Inactive requests:
  toolbox (already provided by toolbox-0.0.96-1.fc33.x86_64)
Checking out tree 7e1aaac... done
Enabled rpm-md repositories: fedora-cisco-openh264 updates fedora
rpm-md repo 'fedora-cisco-openh264' (cached); generated: 2020-08-25T19:10:34Z
rpm-md repo 'updates' (cached); generated: 2021-01-26T01:49:56Z
rpm-md repo 'fedora' (cached); generated: 2020-10-19T23:27:19Z
Importing rpm-md... done
Resolving dependencies... done
Will download: 3 packages (6.0 MB)
Downloading from 'fedora'... done
Importing packages... done
```

Figure 115- Installing nmap in SilverBlue

WIRESHARK

It is already available on Silverblue and can be installed using the rpm-ostree.

```
[root@localhost ~]# rpm-ostree install wireshark
Inactive requests:
  toolbox (already provided by toolbox-0.0.96-1.fc33.x86_64)
Checking out tree 7e1aaac... done
Enabled rpm-md repositories: fedora-cisco-openh264 updates fedora
rpm-md repo 'fedora-cisco-openh264' (cached); generated: 2020-08-25T19:10:34Z
rpm-md repo 'updates' (cached); generated: 2021-01-26T01:49:56Z
rpm-md repo 'fedora' (cached); generated: 2020-10-19T23:27:19Z
Importing rpm-md... done
Resolving dependencies... done
Will download: 6 packages (29.3 MB)
Downloading from 'updates'... done
Downloading from 'fedora'... done
```

Figure 116- Installing Wireshark in Silverblue

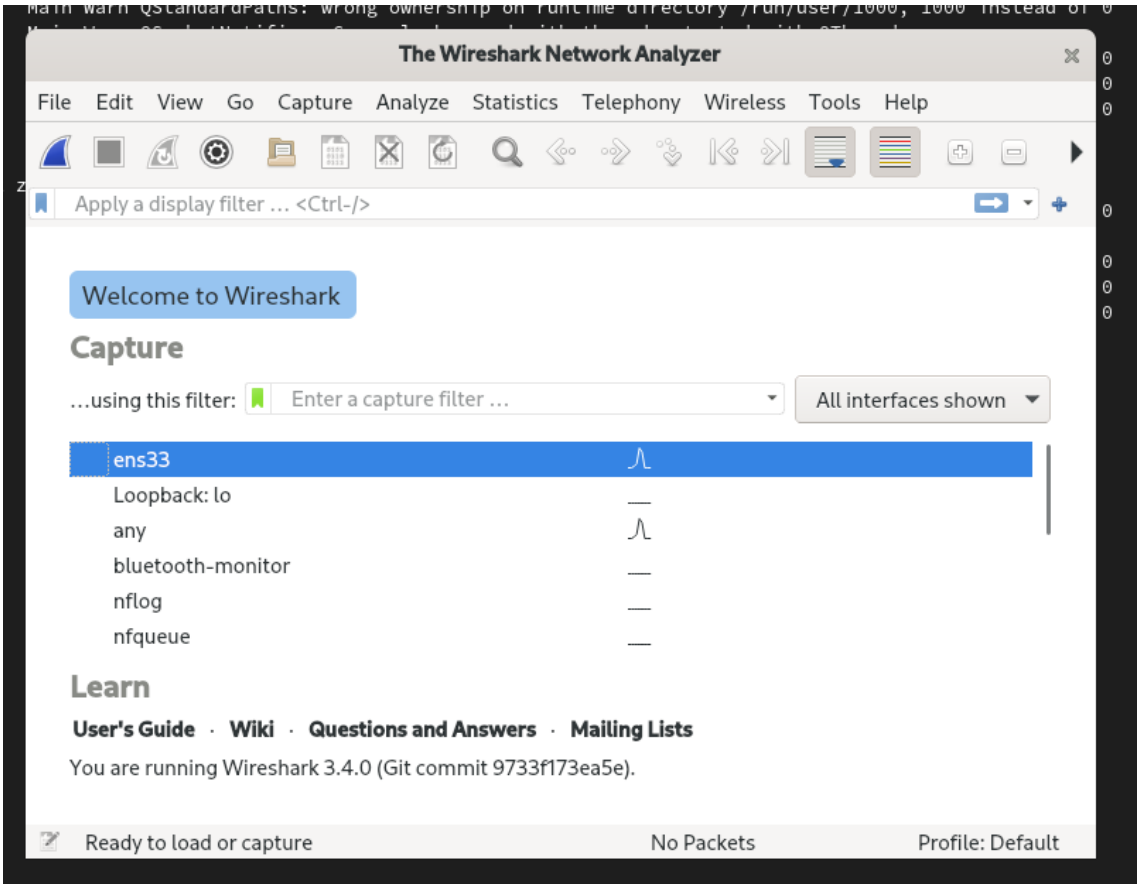


Figure 117- Wireshark starting in Silverblue

SET

The SET security tool is not available on Fedora Silverblue and can be installed using the git cloning method.

```
[root@localhost zainabi]# git clone https://github.com/trustedsec/social-engineer-toolkit/ setoolkit/
Cloning into 'setoolkit'...
remote: Enumerating objects: 110045, done.
remote: Total 110045 (delta 0), reused 0 (delta 0), pack-reused 110045
Receiving objects: 100% (110045/110045), 175.21 MiB | 18.81 MiB/s, done.
Resolving deltas: 100% (68248/68248), done.
```

Figure 118- git cloning in Silverblue

Here, it showed that it has no C compiler, so I installed gcc, gcc-g++, kernel-headers.

```
checking for gcc... no
checking for cc... no
checking for cl.exe... no
configure: error: in `/tmp/pip-install-jissbyl1/pycrypto':
configure: error: no acceptable C compiler found in $PATH
See `config.log' for more details
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/tmp/pip-install-jissbyl1/pycrypto/setup.py", line 456, in <module>
    core.setup(**kw)
  File "/usr/lib64/python3.9/distutils/core.py", line 148, in setup
    dist.run_commands()
  File "/usr/lib64/python3.9/distutils/dist.py", line 966, in run_commands
    self.run_command(cmd)
  File "/usr/lib64/python3.9/distutils/dist.py", line 985, in run_command
    cmd_obj.run()
  File "/usr/lib/python3.9/site-packages/setuptools/command/install.py", line 61, in run
    return orig.install.run(self)
  File "/usr/lib64/python3.9/distutils/command/install.py", line 557, in run
```

Figure 119- Missing package gmp and mpir in Silverblue

Here, it showed missing gmp so I installed gmp and mpir using the rpm-ostree install

```
    warning: GMP or MPIR library not found; Not building Crypto.PublicKey._fastm
ath.
    building 'Crypto.Hash._MD2' extension
    creating build/temp.linux-x86_64-3.9
    creating build/temp.linux-x86_64-3.9/src
    gcc -pthread -Wno-unused-result -Wsign-compare -DDYNAMIC_ANNOTATIONS_ENABLED
=1 -fexceptions -grecord-gcc-switches -pipe -Wall -Werror=format-security -Wp,-D
_FORTIFY_SOURCE=2 -Wp,-D_GLIBCXX_ASSERTIONS -fstack-protector-strong -m64 -mtune
=generic -fasynchronous-unwind-tables -fstack-clash-protection -D_GNU_SOURCE -fP
```

Figure 120- Missing package gmp and mpir in Silverblue

Here, it was unable to compile, because of missing python development tools, so I installed python3-devel.

```
D_GNU_SOURCE -fPIC -fwrapv -fPIC -std=c99 -O3 -fomit-frame-pointer -Isrc/ -I/usr
/include/python3.9 -c src/MD2.c -o build/temp.linux-x86_64-3.9/src/MD2.o
    src/MD2.c:31:10: fatal error: Python.h: No such file or directory
       31 | #include "Python.h"
          |          ^~~~~~~~~~
    compilation terminated.
    error: command '/usr/bin/gcc' failed with exit code 1
    ----------------------------------------
ERROR: Command errored out with exit status 1: /usr/bin/python3 -u -c 'import sy
s, setuptools, tokenize; sys.argv[0] = '"'"'/tmp/pip-install-qgu9z674/pycrypto/s
```

Figure 121- Missing python dev tools in Silverblue

After this, it successfully ran without any errors.

```
[---]       Homepage: https://www.trustedsec.com       [---]
        Welcome to the Social-Engineer Toolkit (SET).
         The one stop shop for all of your SE needs.

   The Social-Engineer Toolkit is a product of TrustedSec.

           Visit: https://www.trustedsec.com

   It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!


 Select from the menu:

   1) Social-Engineering Attacks
   2) Penetration Testing (Fast-Track)
   3) Third Party Modules
   4) Update the Social-Engineer Toolkit
   5) Update SET configuration
   6) Help, Credits, and About

  99) Exit the Social-Engineer Toolkit

set>
```

Figure 122- SET tool starting in Silverblue

METASPLOIT FRAMEWORK

The Metasploit framework is not available on Silverblue and can be downloaded using wget.

```
[root@localhost ~]# wget https://downloads.metasploit.com/data/releases/metasplo
it-latest-linux-x64-installer.run
--2021-01-26 21:19:34--  https://downloads.metasploit.com/data/releases/metasplo
it-latest-linux-x64-installer.run
Resolving downloads.metasploit.com (downloads.metasploit.com)... 23.204.60.91
Connecting to downloads.metasploit.com (downloads.metasploit.com)|23.204.60.91|:
443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 204571447 (195M) [text/plain]
Saving to: 'metasploit-latest-linux-x64-installer.run'

metasploit-latest-l 100%[===================>] 195.09M  30.7MB/s    in 6.8s

2021-01-26 21:19:41 (28.7 MB/s) - 'metasploit-latest-linux-x64-installer.run' sa
ved [204571447/204571447]

[root@localhost ~]#
```

Figure 123- Downloading Metasploit in Silverblue

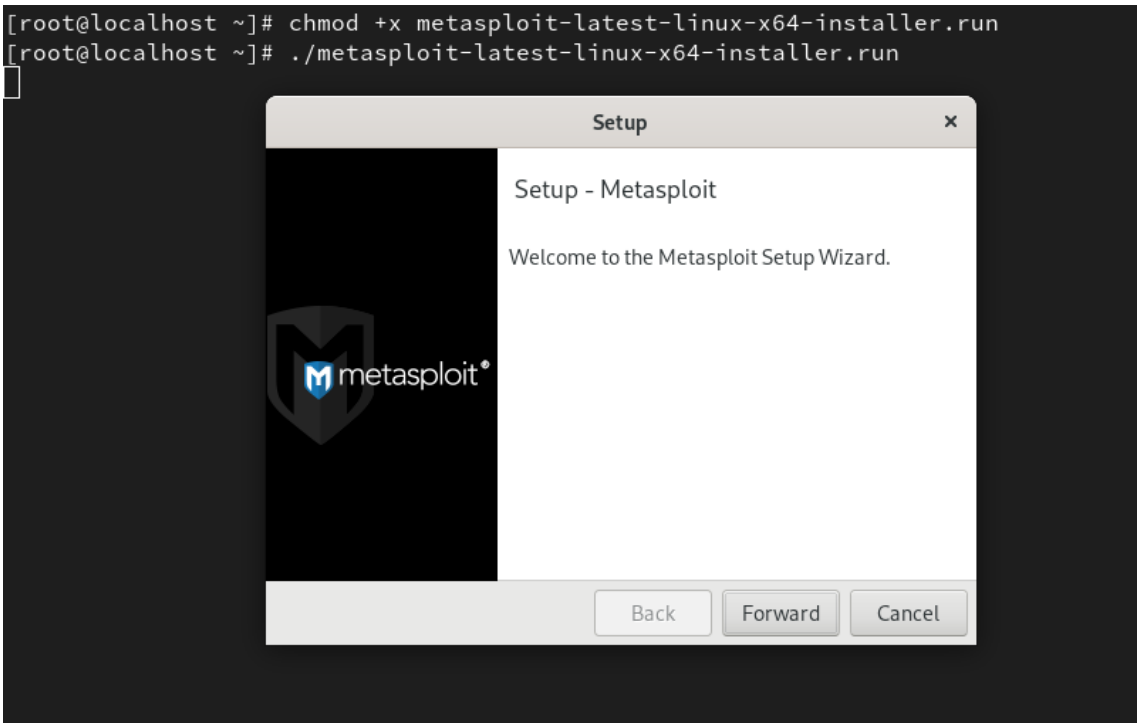


Figure 124- Metasploit starting in SilverBlue

Metasploit framework started!

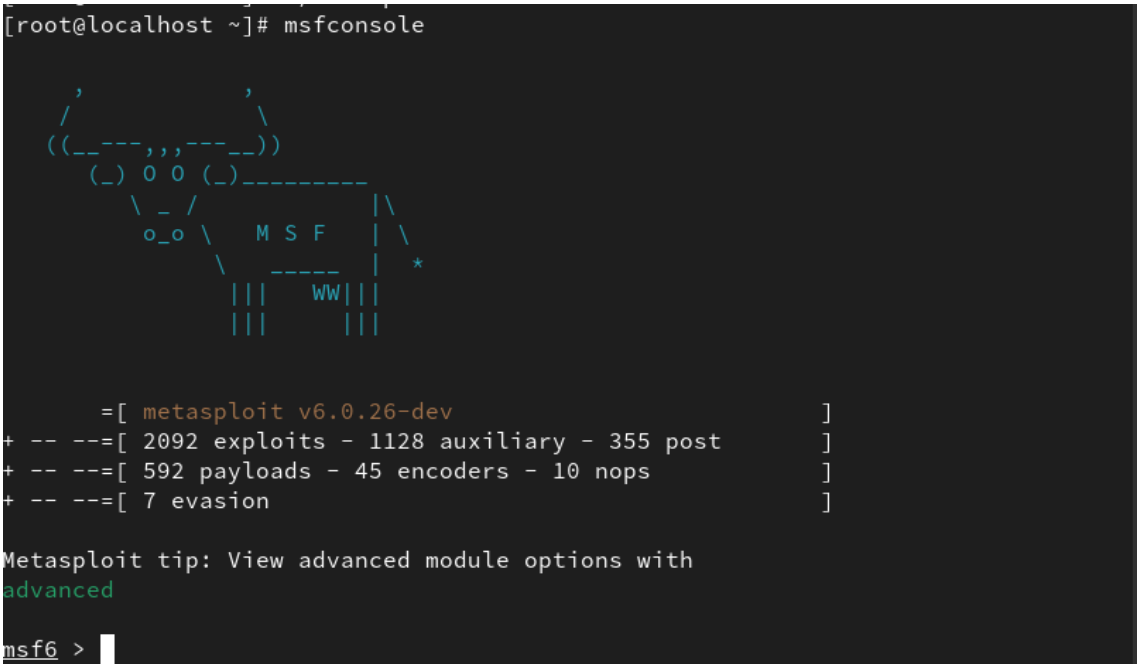


Figure 125- Metasploit starting in SilverBlue

SQLMAP

It is not available on Silverblue Linux and can be installed using the git cloning method.

```
[root@localhost ~]# git clone https://github.com/sqlmapproject/sqlmap.git sqlmap
-dev
Cloning into 'sqlmap-dev'...
remote: Enumerating objects: 106, done.
remote: Counting objects: 100% (106/106), done.
remote: Compressing objects: 100% (80/80), done.
remote: Total 77941 (delta 53), reused 50 (delta 25), pack-reused 77835
Receiving objects: 100% (77941/77941), 73.66 MiB | 12.00 MiB/s, done.
Resolving deltas: 100% (61670/61670), done.
[root@localhost ~]#
```

Figure 126- git cloning in Silverblue

Sqlmap started successfully!

```
[root@localhost ~]# cd sqlmap-dev/
[root@localhost sqlmap-dev]# python sqlmap.py -hh
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.5.1.40#dev}
|_ -| . ['] |  .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

Usage: python sqlmap.py [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
    At least one of these options has to be provided to define the
    target(s)

    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -d DIRECT           Connection string for direct database connection
    -l LOGFILE          Parse target(s) from Burp or WebScarab proxy log file
```

Figure 127- Sqlmap starting in Silverblue

4. Analysis

Tools Testing

To determine the best Linux distribution which can act as a portable security solution tool, the best way is to check if the Kali security tools are installable on those distributions or not. To do this, I selected the five Kali security tools to test their installation on four different Linux distributions. The tools that I selected are Nmap, Sqlmap, Wireshark, Metasploit framework and SET.

Kali Tools	**Slitaz**		**Silverblue**		**Tiny Core**		**Porteus**	
	Available	Installable	Available	Installable	Available	Installable	Available	Installable
Nmap	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
sqlmap	No	Yes	No	Yes	No	Yes	No	Yes
Wireshark	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Metasploit Framework	No	Yes	No	Yes	No	Yes	No	Yes
SET	No	Yes	No	Yes	No	Yes	No	Yes

Nmap is the only tool that is already available on all the Linux distributions. Other than that, sqlmap is not available on any distribution but it was installable on all of them. I installed it using the git cloning method in all the distributions. For Slitaz Linux, I first installed git using the tazpkg package manager because it did not have it already. Then I installed python3, but it had some dependencies and did not work so I then installed python 2.7.2. Then I used the git cloning method to install sqlmap and it worked without any error. For the other three distributions, i.e., Silverblue, Tinycore and Porteus, I did it the same way by installing python and then cloning the link address.

Wireshark is available on all the Linux distributions except the Porteus. To install Wireshark on Porteus, I used a script and tried to run it with wget. It showed a lot of dependencies and missing libraries one by one. It did not have basic compilers, cmake and tools to install Wireshark. Every time it showed different missing libraries and I installed all of them using the Slackware repository. At the end, Wireshark started without any errors.

Metasploit framework is not available on any distribution but it is installable on all of them. To install it, I used wget on all the distributions, changed the mode of the link address to executable and ran it. It successfully started on all the distributions.

SET tool is not available on any distribution but it is installable on all of them. It was the only tool that was the most complex to install on all the distributions. It showed a lot of dependencies and compiler errors. It required a lot of libraries one after the another. All the distributions lacked the basic compilers and development tools required to run a python program. After installing all the dependencies, the SET tool worked on all the distributions.

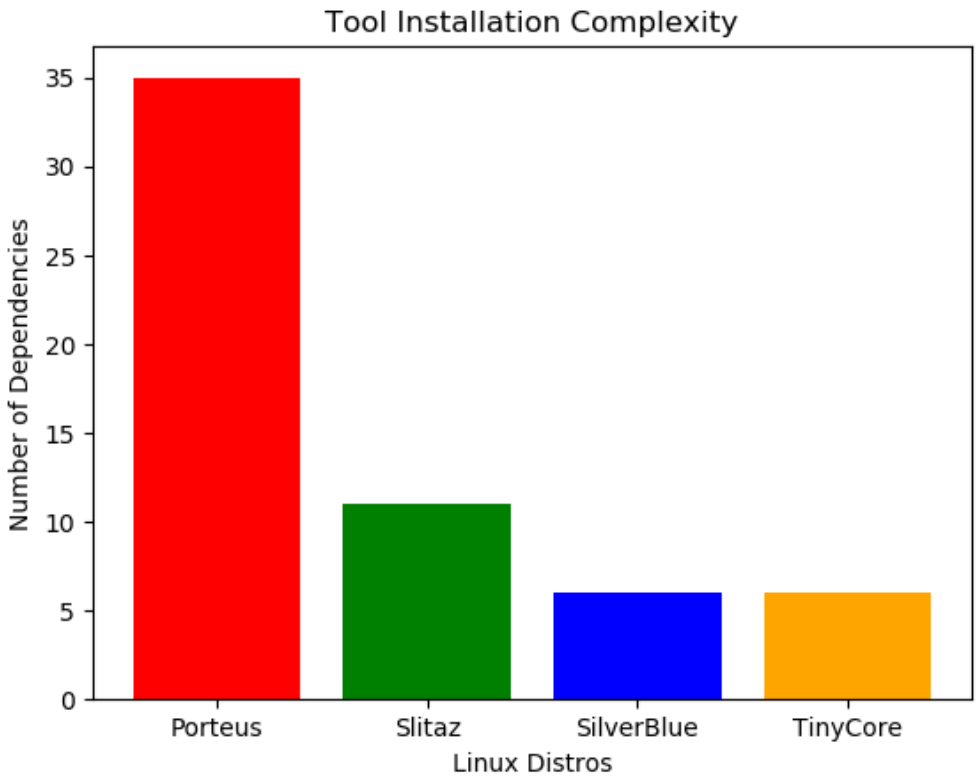


Figure 128- Tools complexity graph

The above graph shows the complexity of installing the security tools in the four distributions i.e., Silverblue, Tinycore, Slitaz and Porteus. According to this graph, installing the security tools on Porteus was the most complex because it involved a lot of dependencies and missing libraries. Slitaz also had dependencies and had issues with the version as it showed a lot of incompatibility issues. Apart from Porteus and Slitaz, Tiny core and Silver Blue showed very less dependencies which were mostly related to the compiler and development tools.

For selecting the best distribution, which is a portable security solution tool, Tiny Core and SilverBlue stand out. Here if I had to choose one, I would choose Tiny Core because its much smaller in size and does not save the state as compared to the Fedora Silverblue.

Tools Efficiency/Time comparison

The other factor that matters in selecting the best security solution tool is the speed of the Linux system or the time is takes to fetch a result from a security tool command. To test this, I tested same command of each tool on all the distributions and noted the time out of it.

Kali Tools	**Kali**	**Slitaz**	**Silver Blue**	**Tiny Core**	**Porteus**
Nmap (seconds)	5.01	9.96	15.07	10.23	10.28
Sqlmap(seconds)	36.30	40.06	30.75	39.84	30.14
Wireshark	380 packets in 1 min	508 packets in 1 min	1687 packets in 1 min	456 packets in 1 min	311 packets in 1 min
Metasploit framework (seconds)	4.92	10.56	14.10	11.50	11.00
SET	N/A	N/A	N/A	N/A	N/A

Regarding the time efficiency of the distributions, Porteus gave the best result and Tiny core took the longest time.

Size Comparison

To select the best Linux distribution, which is portable as a security solution tool, the first thing is to consider the size of the distribution.

	Kali	**Slitaz**	**Silverblue**	**Tinycore**	**Porteus**
Size	4.1 GB	34.7 MB	2.67 GB	14.07 MB	347 MB

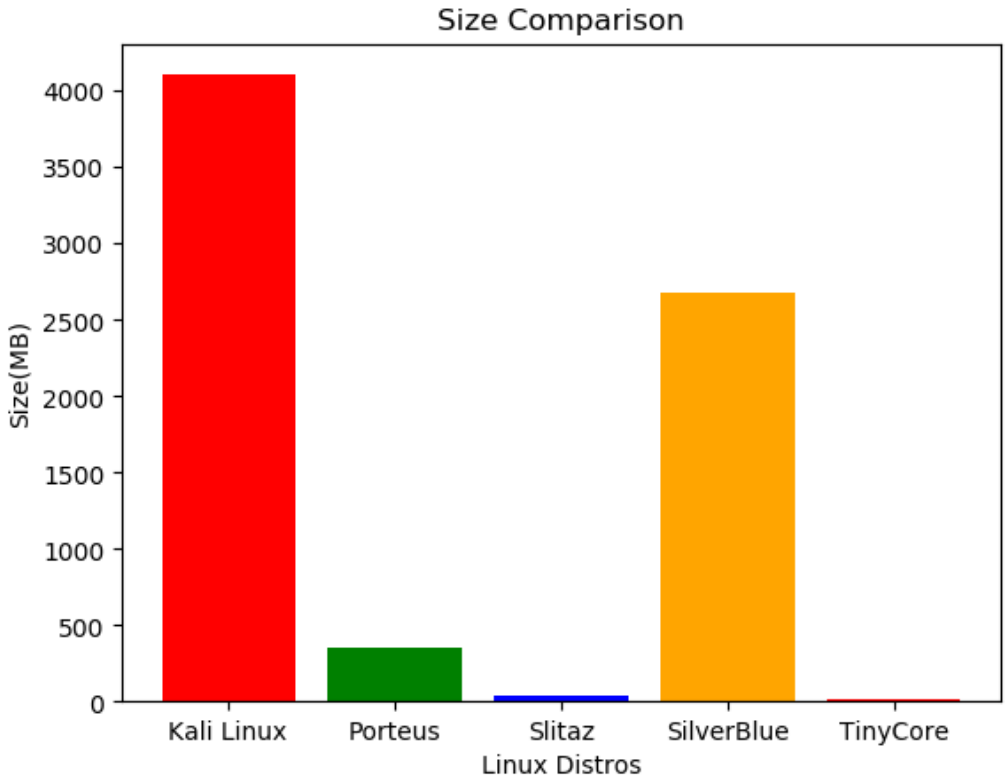


Figure 129- Size comparison graph of Linux distros

Kali Linux is known as the best security-based distribution of Linux, but it is too large to act as a portable security solution tool. For a tool to be portable, the size should be enough to fit in a small USB so that the user can plug in the USB anywhere, do the task and plug it out.

As Kali Linux is large, it takes a lot of RAM to run the system because it must load the resources, the GUI interface and prepare the hardware for use. Kali Linux can be installed with as less space as 500MB and RAM with 128 MB but only as a SSH with no desktop. With small distributions, they can run on system which do not have a lot of RAM. Also, with small size and less RAM usage, it is easy for the user to plug the USB, use the system and then plug it out. These distributions acts as LiveCD and as soon as the user is done and shuts the system, nothing is saved. This way, there are other distributions to consider which can act as a portable security solution tool.

Considering the size of a Linux distribution, Tiny Core is the best because it very small in size and it can run on any computer whether it is old or new as it requires very minimum requirements. It uses very minimum resources on a system which helps the PC to make the resources available for

other things like running the server and apps the user wants to. It does not come with a lot of security tools like Kali Linux, but the user can install any tool on it depending on the requirement. In this way, the user gets a lightweight, fast, reliable, and secure Linux distribution which can act as a portable security solution tool.

5. Conclusion: Determining Best Distribution

Focusing on the basics of security of all the Linux distributions, a lot of Linux distributions can act as a portable security solution tool but not be the best solution. Talking about the world of security, every company, organization or individual wants a security solution which is the best in every aspect.

Linux is known for its security but selecting the best distribution considering the basics of security is not an easy task. For this, it is required to look at the compatibility, GUI, installation of security tools, size comparison and efficiency. When focused on these factors and according to the findings described in this research paper, Tiny core Linux meets the criteria of the best portable security solution tool.

Why is Tiny Core the best solution as compared to the other distributions? For this question, I compared it with Kali Linux as it is the best-known security Linux distribution in the market. But because of its size it is not portable, and the purpose of this research paper was to select a Linux Distribution which is as good as Kali Linux but less in size in order to be portable.

Tiny core can install the Kali Linux security tools with much less dependencies as compared to other distributions. In addition to this, it is small in size so it can run on old computers that do not require a lot of RAM. It does not save anything from a session and as soon as the users logs out or unplugs the USB, nothing is saved. It is very helpful for security because, if an individual is working on the system which is not secure and the lights go out or the system crashes, nothing is saved.

Regarding the efficiency of Tiny Core, it took a lot of time in running the commands of the security tools but considering the other aspects of a portable security solution tool it is the best among Porteus, Slitaz and Silver Blue. If Tiny Core Linux is more optimized and its RAM is the same as Kali Linux, it will be almost equal to Kali Linux. In conclusion, Tiny Core is the best portable security solution tool.

6. Appendix A

Installation Process of Kali Linux

To install Kali Linux on VMware workstation,

1. The first step is to download the iso file of Kali Linux from the official website.

https://www.kali.org/

2. After going into downloads in the official website.

Kali Linux Downloads

Download Kali Linux Images

We generate fresh Kali Linux image files every few months, which we make available for download. This page provides the links to download Kali Linux in its latest official release. For a release history, check our Kali Linux Releases page. Please note: You can find unofficial, untested weekly releases at http://cdimage.kali.org/kali-weekly/. Downloads are **rate limited to 5 concurrent connections**.

Image Name	Torrent	Version	Size	SHA256Sum
Kali Linux 64-Bit (Installer)	Torrent	2020.4	4.1G	50492d761e400c2b5e22c8f253dd6f75c27e4bc84e33c2eff272476a0588fb02
Kali Linux 64-Bit (Live)	Torrent	2020.4	3.3G	4d764a2ba67f41495c17247184d24b7f9ac9a7c57415bbbed663402aec78952b
Kali Linux 64-Bit (NetInstaller)	Torrent	2020.4	471M	fbbb3b86567892f91b8298be7c03e9be8c78c6f048e4c6fff539948743465d79

Figure 130- Kali Linux download

From here, I selected the first download which is "**Kali Linux 64-bit (Installer)**".

Note: The size of Kali Linux is "4.1GB"

3. After that I go to the VMware workstation and click on "Create a New Virtual Machine."

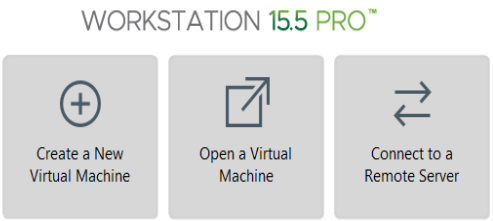


Figure 131- Create a new virtual machine.

4. Choose the “Custom (Advanced)” option

Figure 132- Selecting the type of configuration.

5. Click “Next” from here.

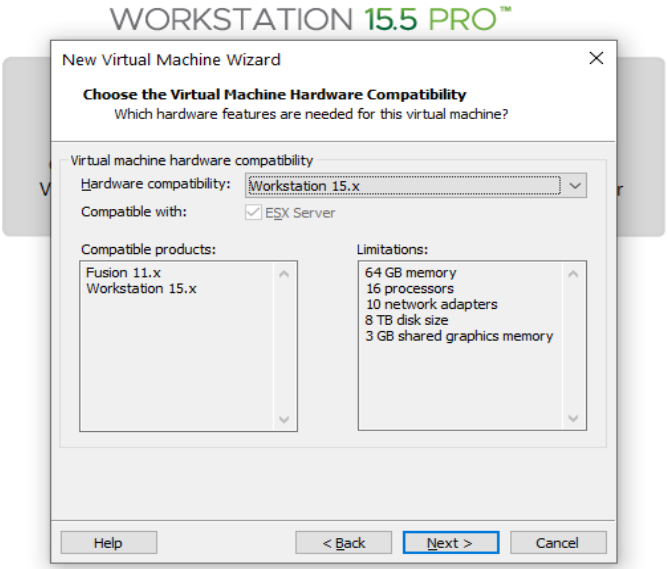


Figure 133- Virtual Machine hardware compatibility

6. Select “I will install the operating system later.”

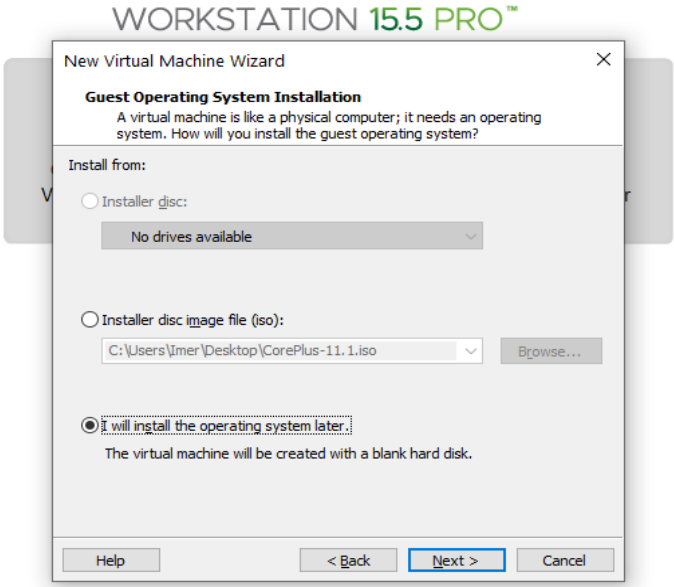


Figure 134- Selecting where to install from

7. Select “Linux” as the Guest Operating System and “Debian 10 x 64-bit" as the version.

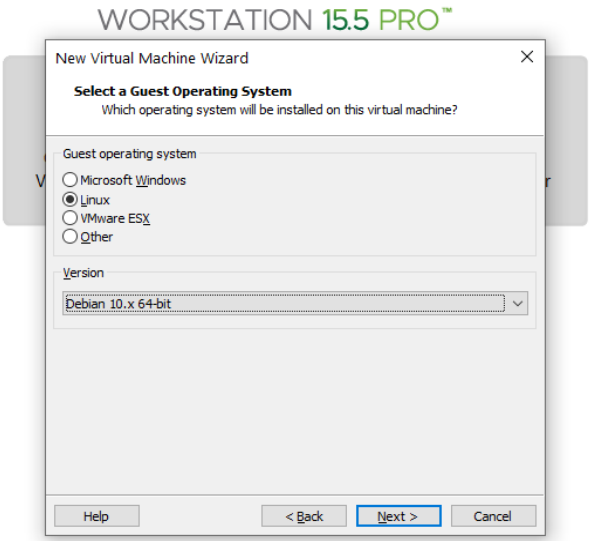


Figure 135- Selecting the guest operating system.

8. Here we specify a name for the Linux system and specify its location as well.

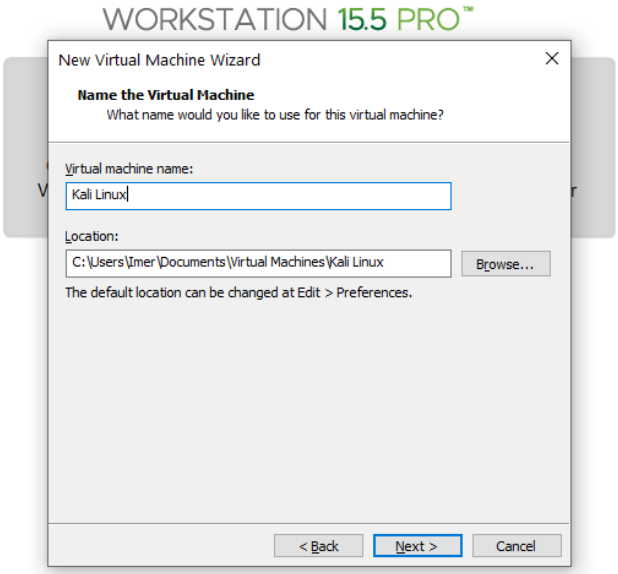


Figure 136- Specifying virtual machine name and location

9. Then specify the number of processors, number of cores per processor and the total processor core.

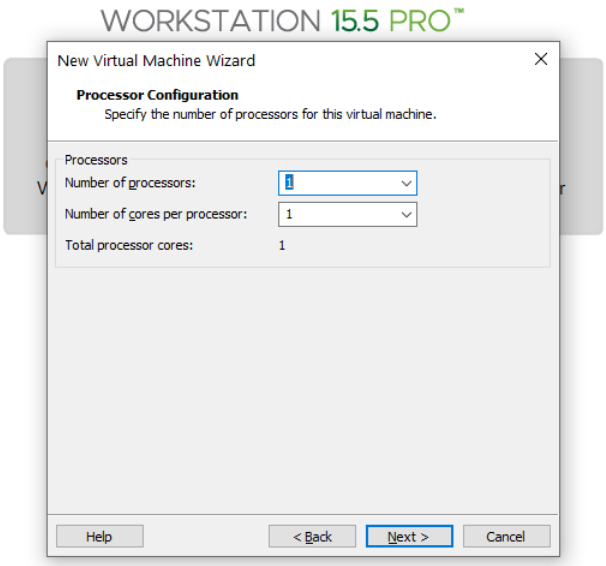


Figure 137- Selecting the number of cores and processors.

10. In the next step, determine the memory of the virtual machine.

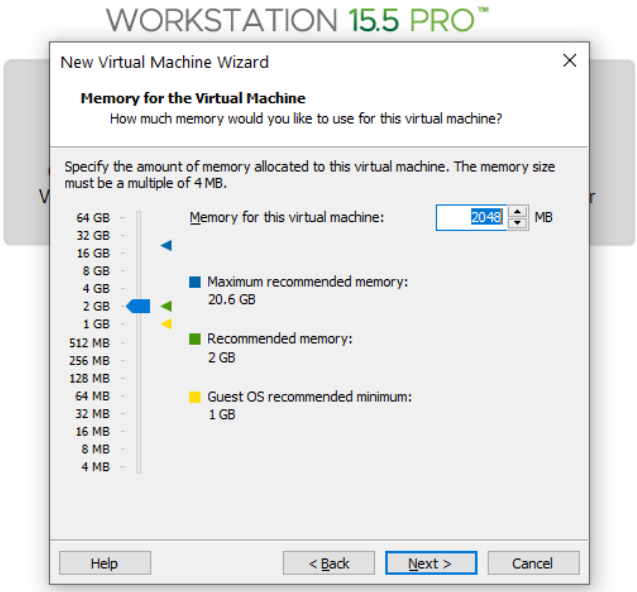


Figure 138- Specifying the memory of the virtual machine

11. Select “NAT” as the network type.

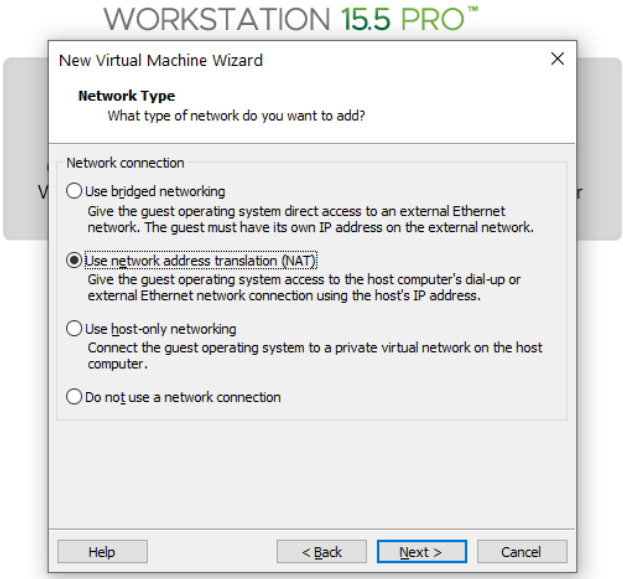


Figure 139- Selecting the network type

12. Select “LSI Logic” as the I/O controller type.

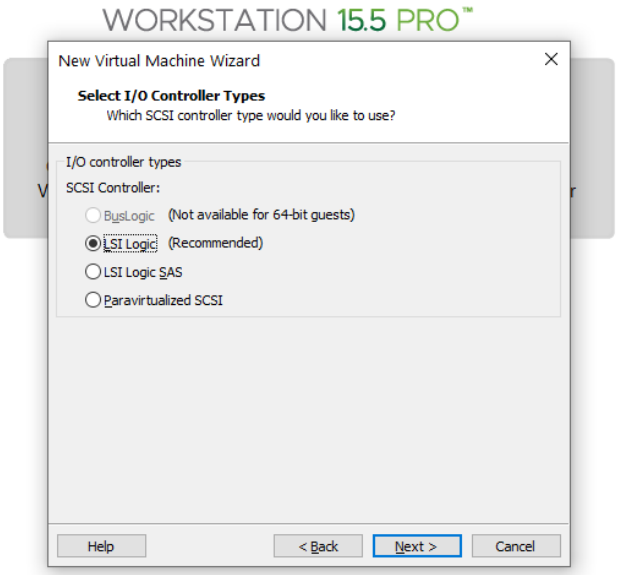


Figure 140- Selecting the I/O controller type

13. Select “SCSI” as the virtual disk type

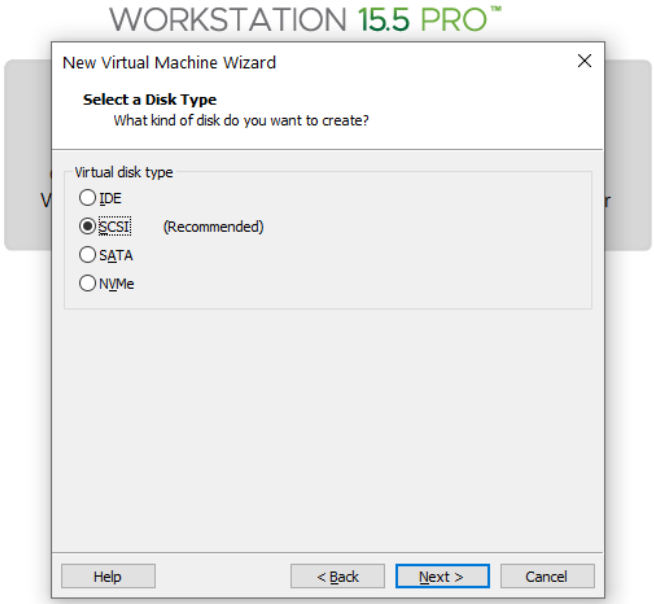


Figure 141- Selecting the virtual disk type

14. Here, select a new virtual disk.

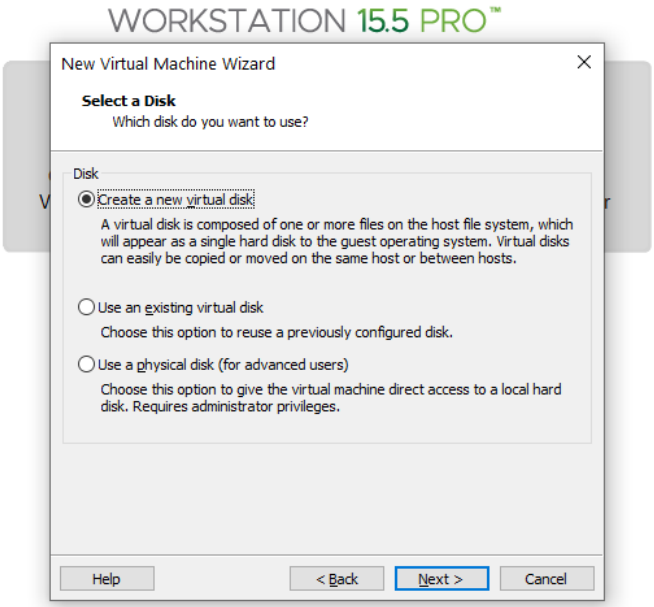


Figure 142- Selecting the disk to be used

15. Then, select the disk capacity and “store virtual disk as a single file.”

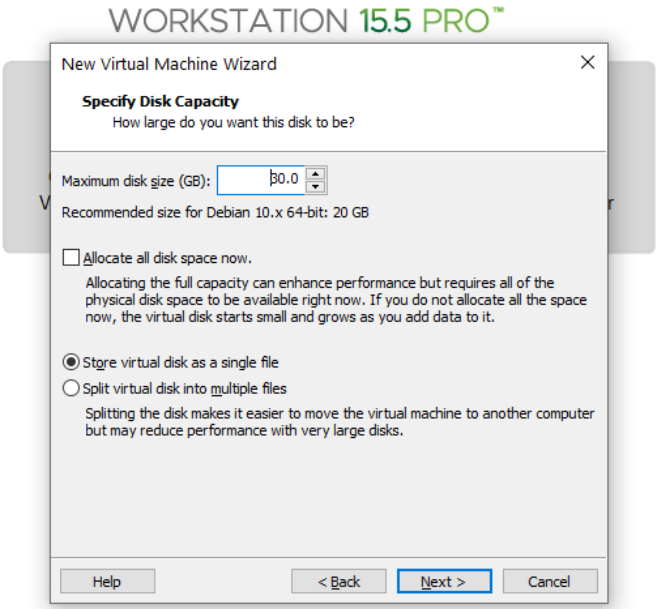


Figure 143- Specifying the disk capacity

16. Here, the screen will show all the specifications and the next step is to go to “Customize Hardware.”

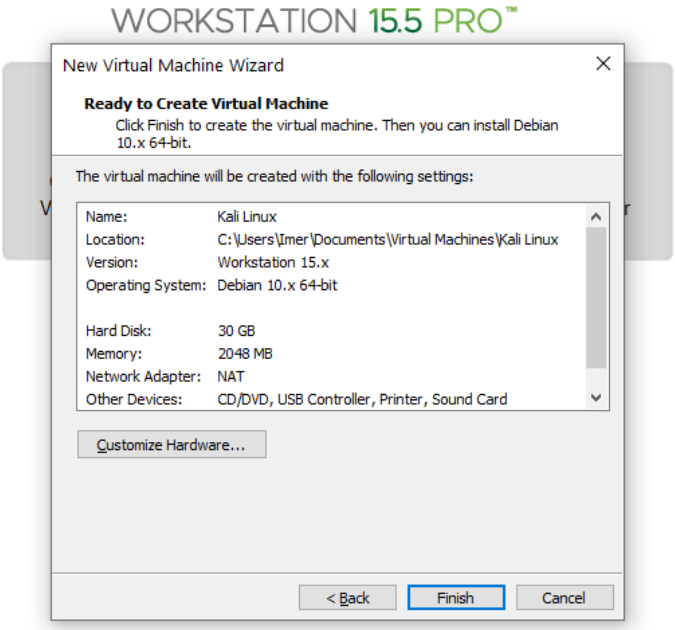


Figure 144- Virtual Machine specifications

17. Here, insert the downloaded iso file of Kali Linux, and remove the USB controller, Sound Card and printer.

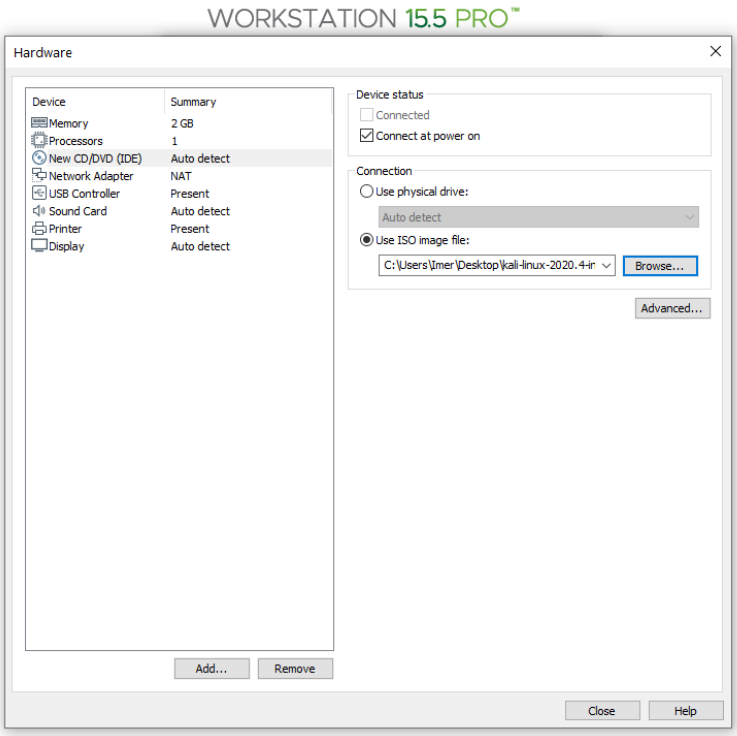


Figure 145- Inserting the Kali Linux iso file

8. After powering on the machine, select graphic install from the menu

Figure 146- Selecting graphic install

19. Select language as English

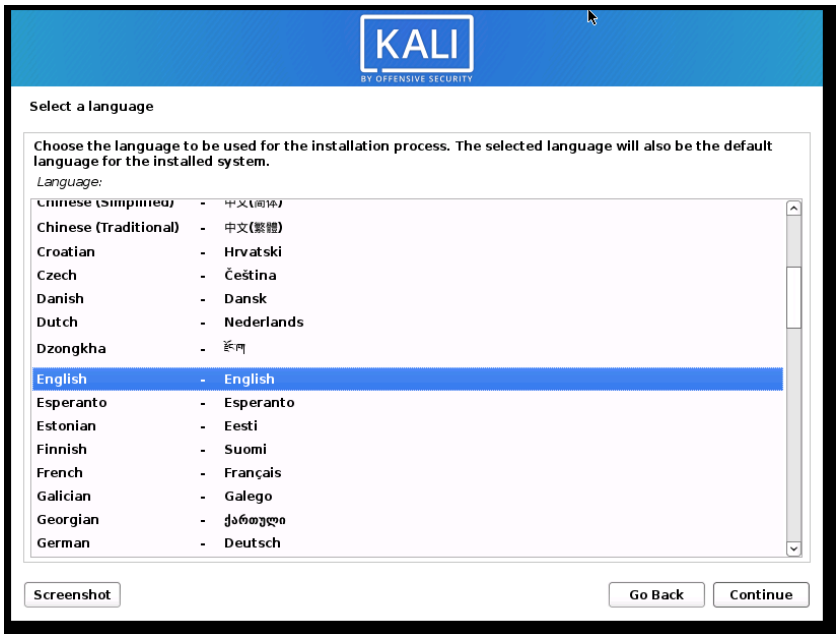


Figure 147- Selecting language as English

20. Select location as Canada

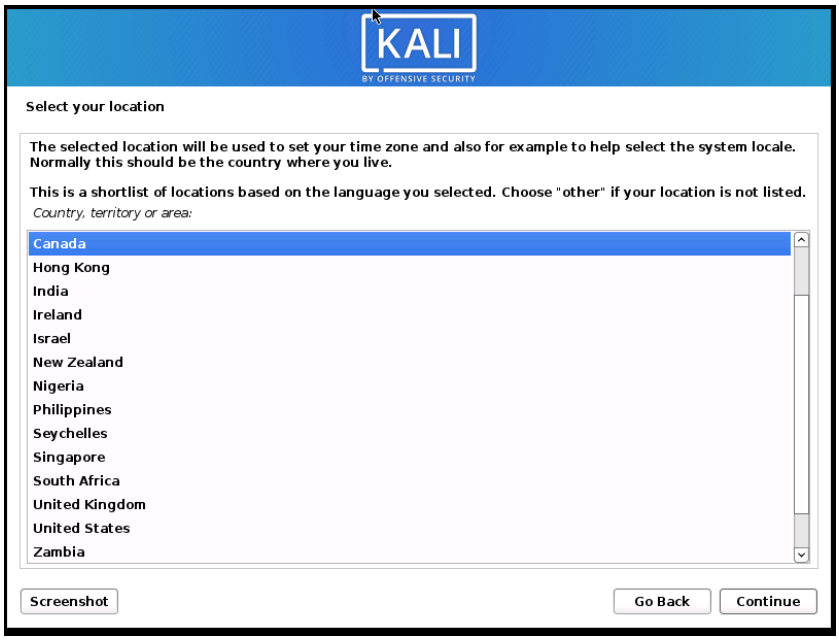


Figure 148- Selecting location as Canada

21. Specify keymap

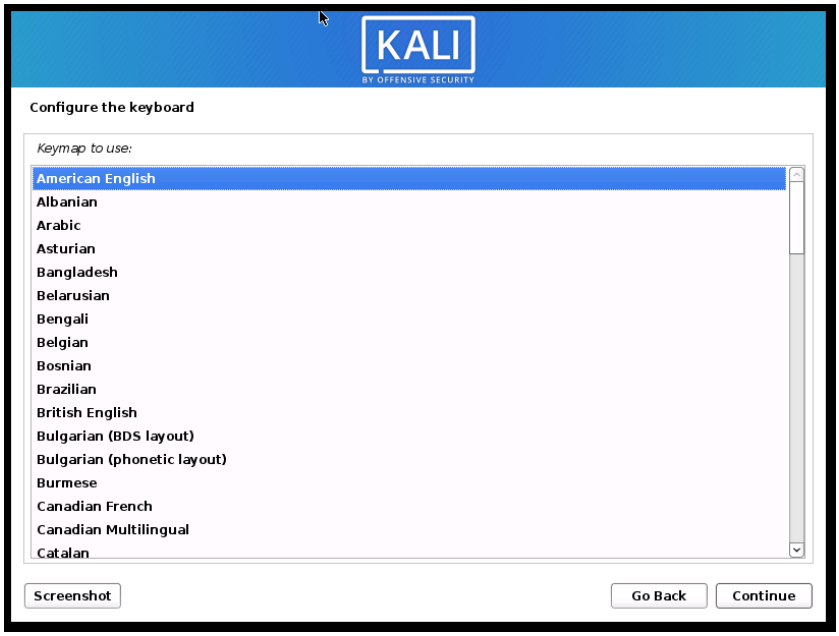


Figure 149- Configuring the keyboard

22. It will start installing.

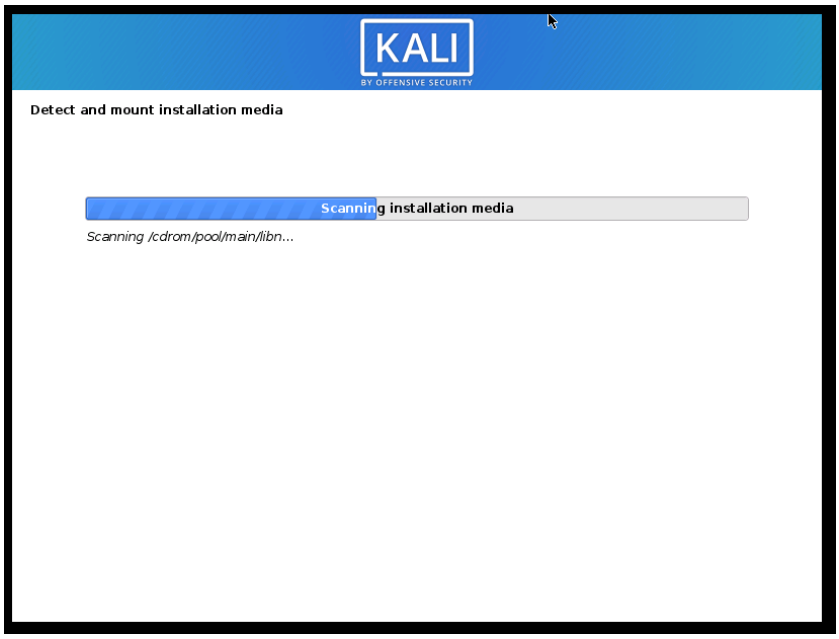


Figure 150- Installing Kali Linux in Virtual Machine

23. Setup the host name

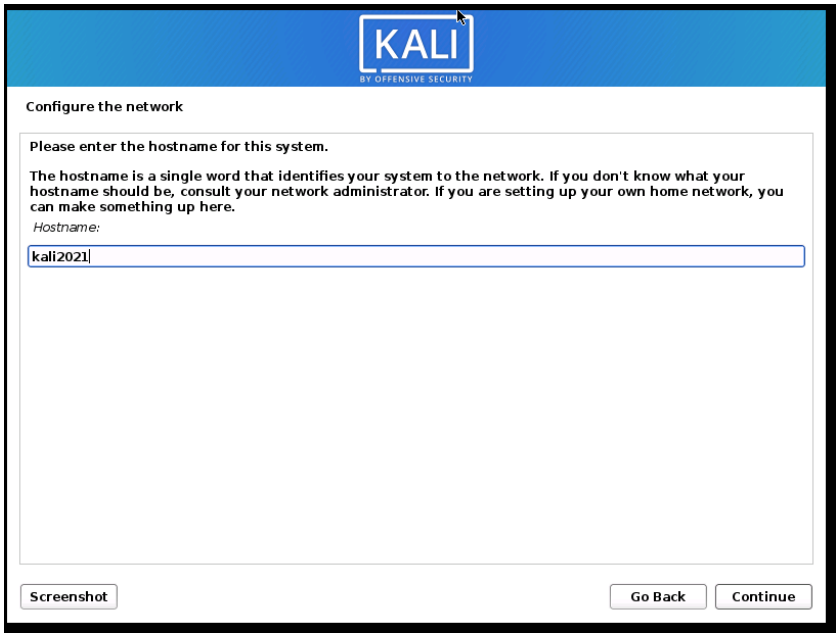


Figure 151- Setting the host name in Kali Linux

24. Setup full name

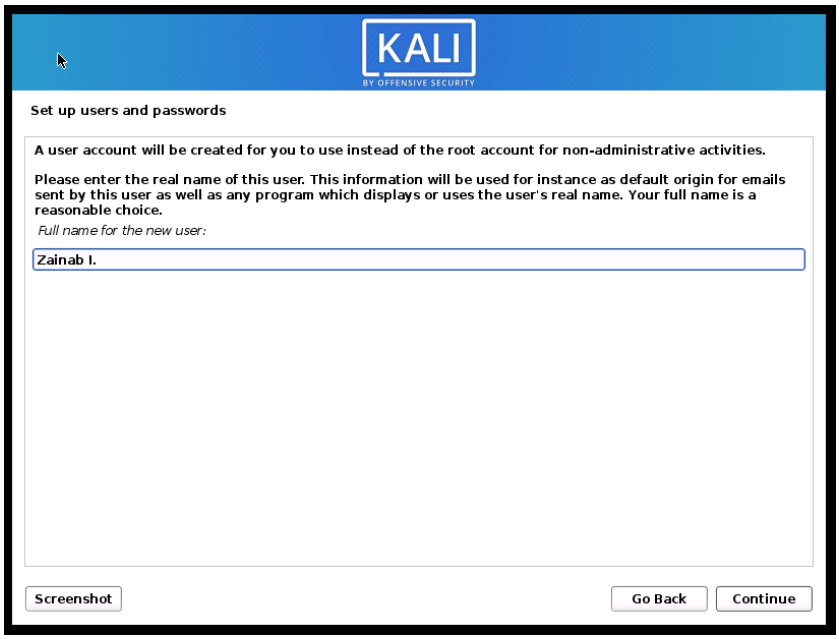


Figure 152- Setting the name of the user account

25. Set username

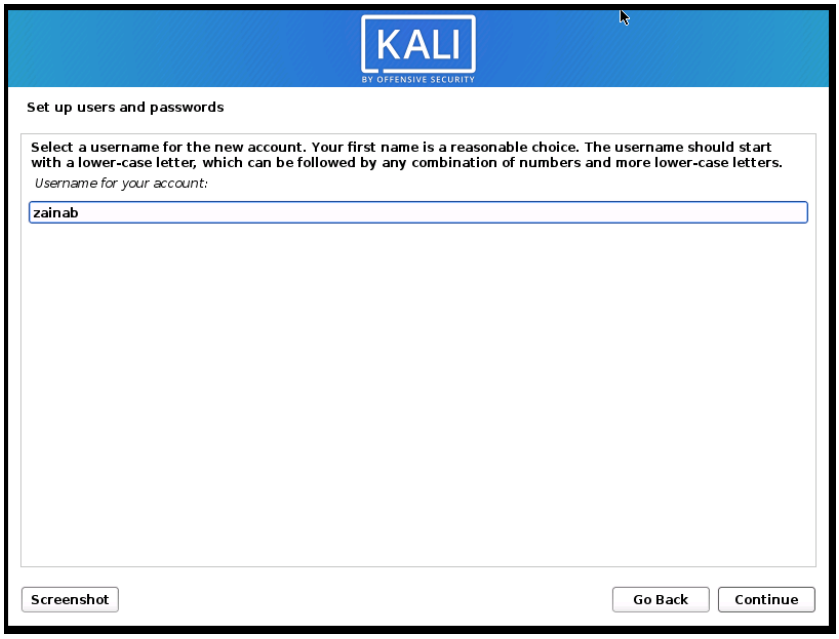


Figure 153- Setting a username

26. Set password

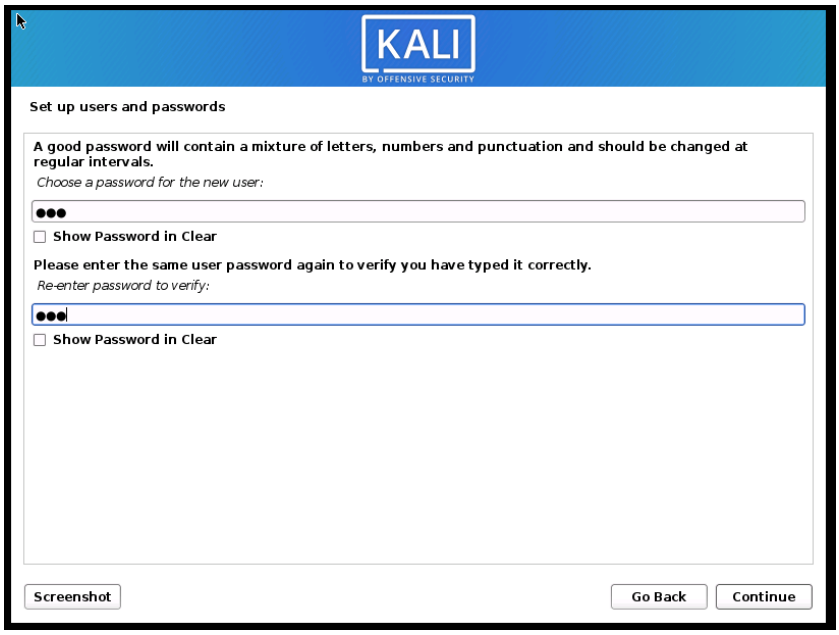


Figure 154- Setting up the password

27. Setup the clock

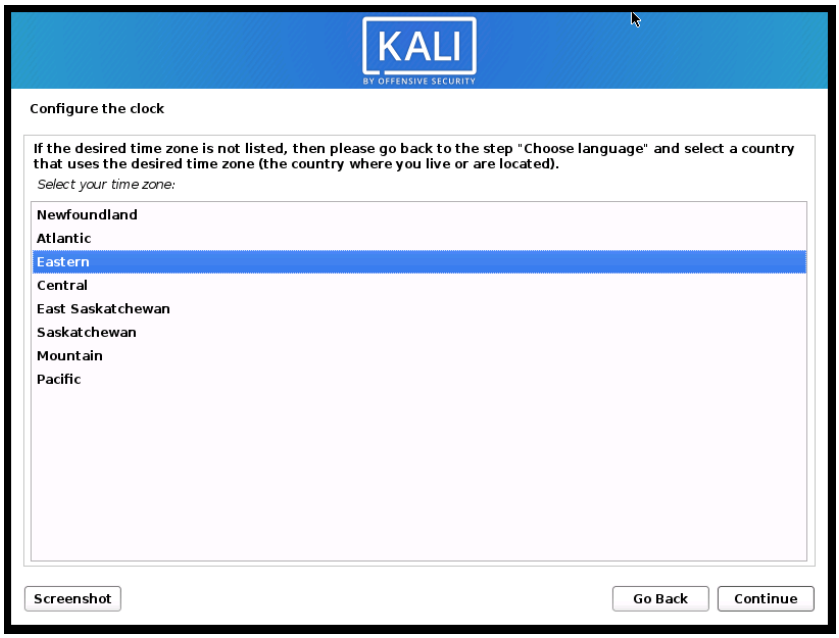


Figure 155- Configuring the clock

28. Select Disk Partitioning

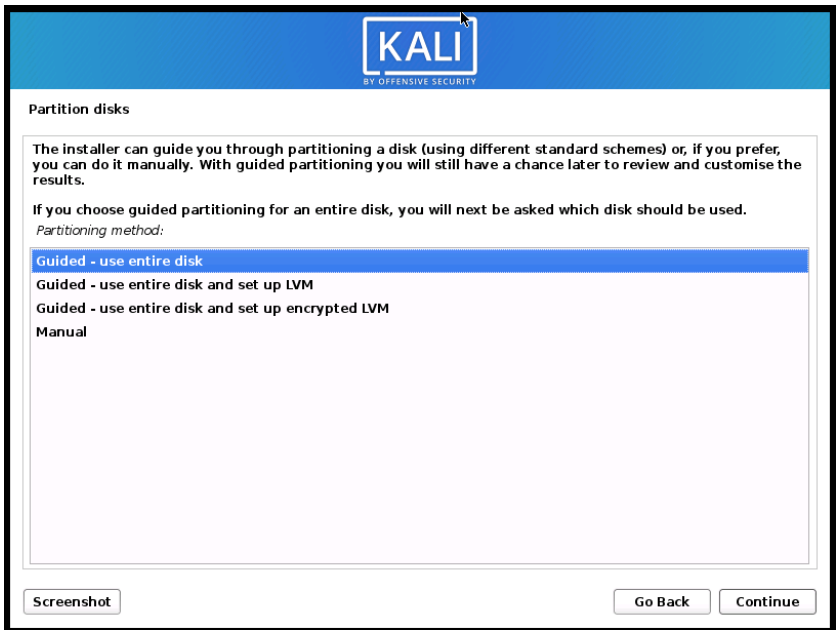


Figure 156- Selecting the disk partition

29.

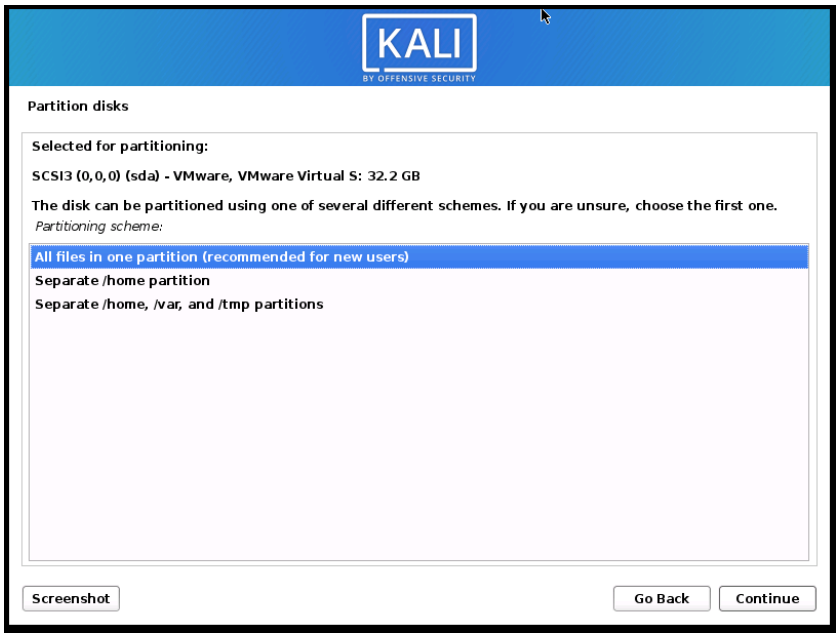


Figure 157- Selecting "all files in one partition."

30.

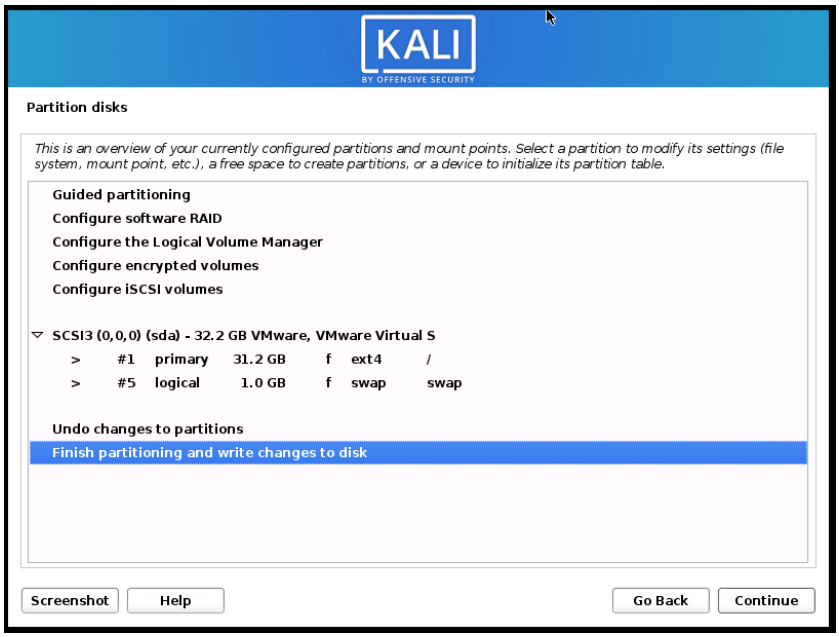


Figure 158- Finish partition of disk and write to file.

31.

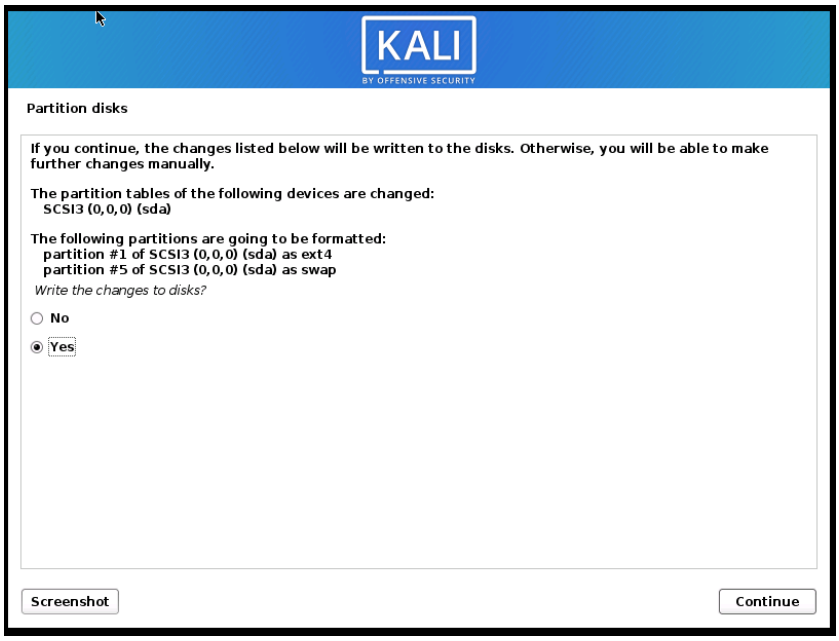


Figure 159- Write partition changes to disk

32.

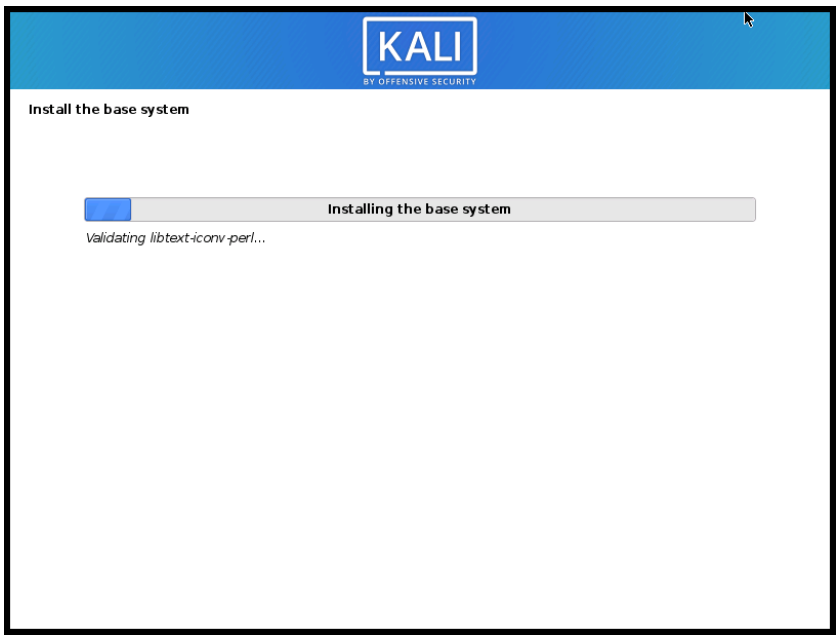


Figure 160- Installing the base system of Kali Linux

33.

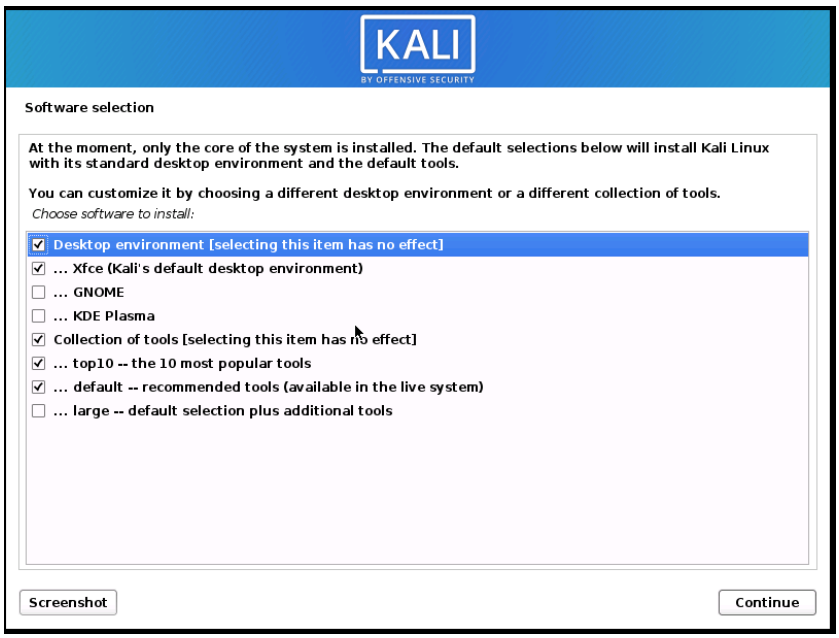


Figure 161- Selecting the desktop environment

34.

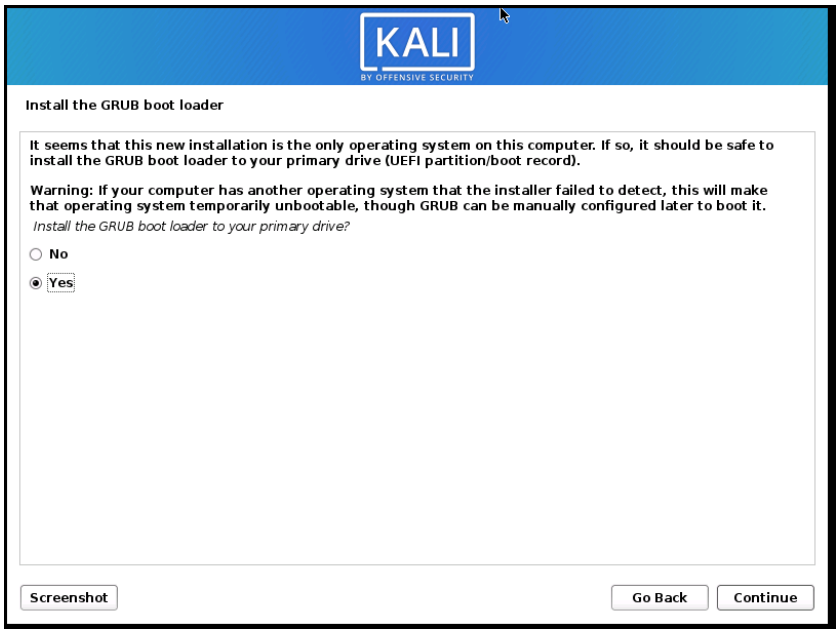


Figure 162- Installing the GRUB boot loader

35.

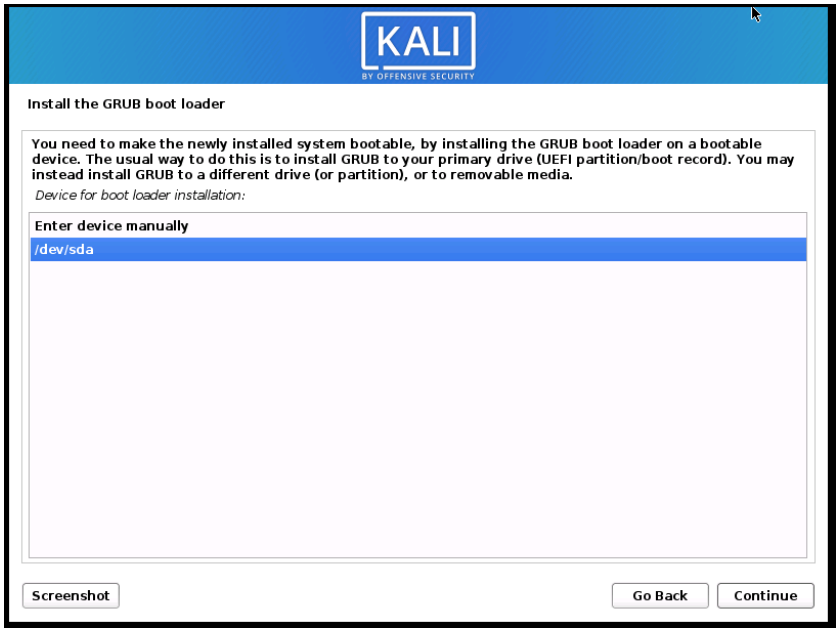


Figure 163- Selecting where to intsall GRUB

36.

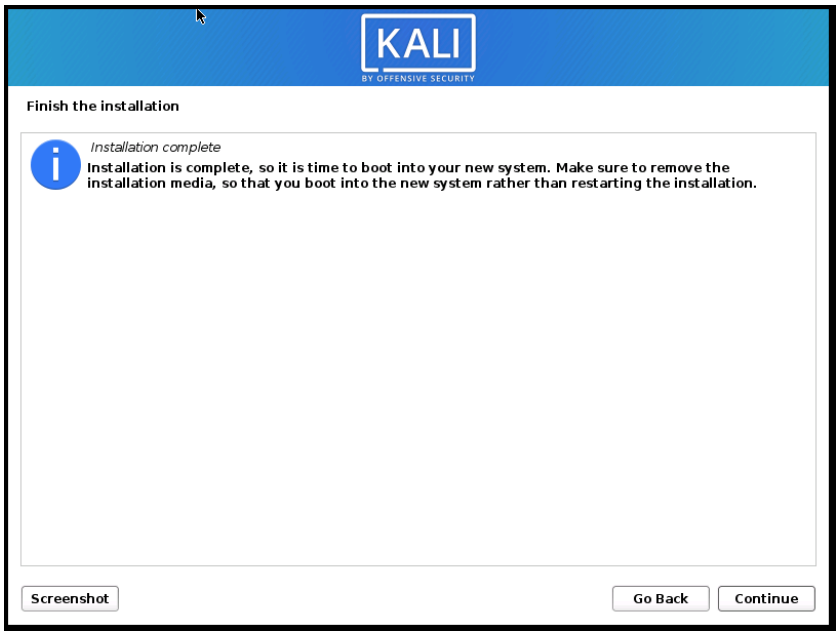


Figure 164- Installation of Kali Linux complete

Conclusion (Installation)

The installation process of Kali Linux is very smooth and not lagging at all. Even for a beginner, it is very easy to figure out the installation process. It comes with a set of security tools already and it also asks for a selection of desktop environment as well.

Installation Process of Slitaz

To install the Slitaz system on the VMware workstation

1.The first step is to download the iso file of Slitaz from its official website.

http://www.slitaz.org/en/

2. The next step is to go into downloads.

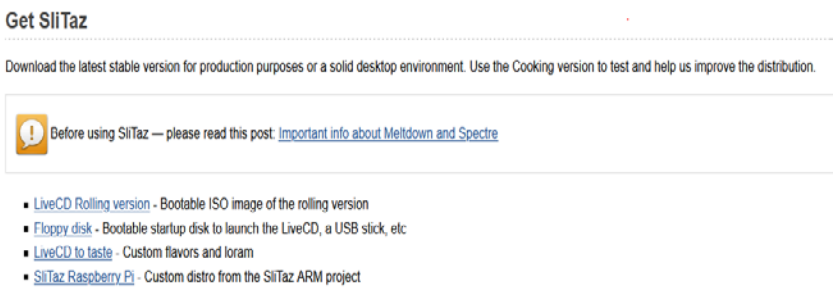
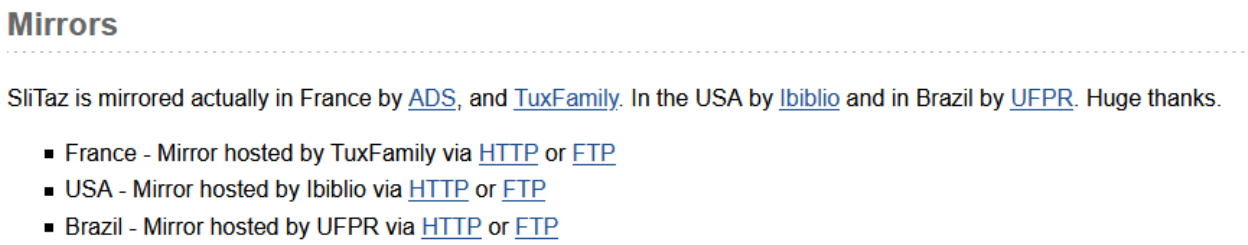
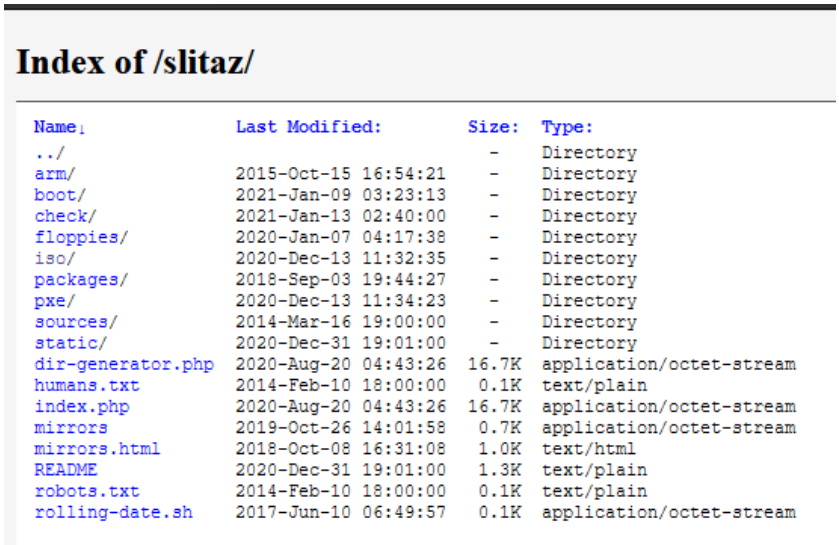


Figure 165- Slitaz official website download page.

After going into LiveCD rolling version, I selected the HTTP by USA



Then I went to iso



Index of /slitaz/

Name	Last Modified:	Size:	Type:
../		-	Directory
arm/	2015-Oct-15 16:54:21	-	Directory
boot/	2021-Jan-09 03:23:13	-	Directory
check/	2021-Jan-13 02:40:00	-	Directory
floppies/	2020-Jan-07 04:17:38	-	Directory
iso/	2020-Dec-13 11:32:35	-	Directory
packages/	2018-Sep-03 19:44:27	-	Directory
pxe/	2020-Dec-13 11:34:23	-	Directory
sources/	2014-Mar-16 19:00:00	-	Directory
static/	2020-Dec-31 19:01:00	-	Directory
dir-generator.php	2020-Aug-20 04:43:26	16.7K	application/octet-stream
humans.txt	2014-Feb-10 18:00:00	0.1K	text/plain
index.php	2020-Aug-20 04:43:26	16.7K	application/octet-stream
mirrors	2019-Oct-26 14:01:58	0.7K	application/octet-stream
mirrors.html	2018-Oct-08 16:31:08	1.0K	text/html
README	2020-Dec-31 19:01:00	1.3K	text/plain
robots.txt	2014-Feb-10 18:00:00	0.1K	text/plain
rolling-date.sh	2017-Jun-10 06:49:57	0.1K	application/octet-stream

Figure 166- Sitaz downloads page

I select stable from here

Index of /slitaz/iso/

Name	Last Modified	Size	Type
Parent Directory/		-	Directory
1.0/	2008-Mar-22 18:52:52	-	Directory
2.0/	2009-Apr-16 22:35:40	-	Directory
3.0/	2010-Mar-28 21:04:45	-	Directory
4.0/	2012-Apr-10 22:49:44	-	Directory
5.0-rc/	2016-Jan-04 22:17:51	-	Directory
cooking/	2021-Jan-11 12:21:23	-	Directory
latest/	2021-Jan-10 04:32:12	-	Directory
next/	2018-Jul-18 01:29:46	-	Directory
rolling/	2021-Jan-10 04:32:12	-	Directory
stable/	2012-Apr-10 22:49:44	-	Directory
tank/	2014-Sep-25 23:35:40	-	Directory
vintage/	2007-Dec-06 00:00:00	-	Directory
.filelist	2020-Dec-13 17:55:19	0.1K	application/octet-stream
.folderlist	2020-Dec-13 17:55:19	1.8K	application/octet-stream

Figure 167- Slitaz downloads diectory

Then I download the “slitaz-4.0.iso” file

Index of /slitaz/iso/stable/

Name	Last Modified	Size	Type
Parent Directory/		-	Directory
flavors/	2014-Dec-12 15:34:04	-	Directory
.filelist	2017-May-29 11:53:49	1.8K	application/octet-stream
.folderlist	2017-May-29 11:53:49	0.1K	application/octet-stream
slitaz-4.0-base.iso	2012-Apr-10 22:49:44	8.0M	application/x-iso9660-image
slitaz-4.0-base.log	2012-Apr-10 22:49:44	3.2K	application/octet-stream
slitaz-4.0-base.md5	2012-Apr-10 22:49:44	0.1K	application/octet-stream
slitaz-4.0-isohybrid.iso	2012-Apr-10 22:49:44	35.0M	application/x-iso9660-image
slitaz-4.0-isohybrid.md5	2012-Apr-10 22:49:44	0.1K	application/octet-stream
slitaz-4.0.iso	2012-Apr-10 22:49:44	34.7M	application/x-iso9660-image
slitaz-4.0.md5	2012-Apr-10 22:49:44	0.1K	application/octet-stream
slitaz-4.0.zip	2015-Mar-31 10:20:17	51.0M	application/zip

Figure 168- Slitaz downloads directory

Note: The size of Slitaz Linux is “34.7MB”

3. Now, I go to the VMware workstation and “Create a new virtual machine”

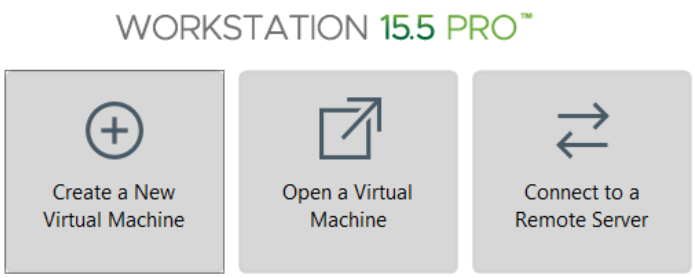


Figure 169- Create a new virtual machine

4. Select the Custom option

Figure 170- Selecting the configuration type

5. Click Next from here

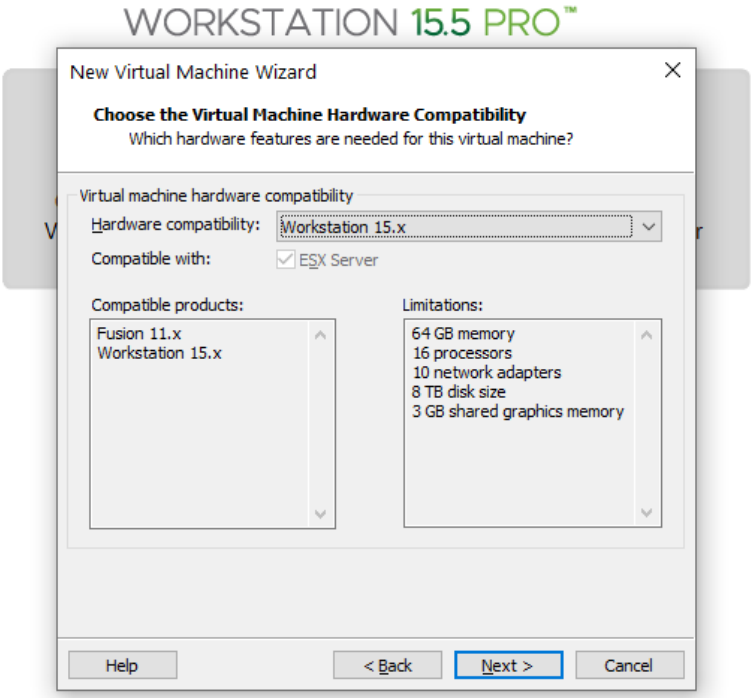


Figure 171- Specifying the virtual machine hardware compatibility.

6. Select last option.

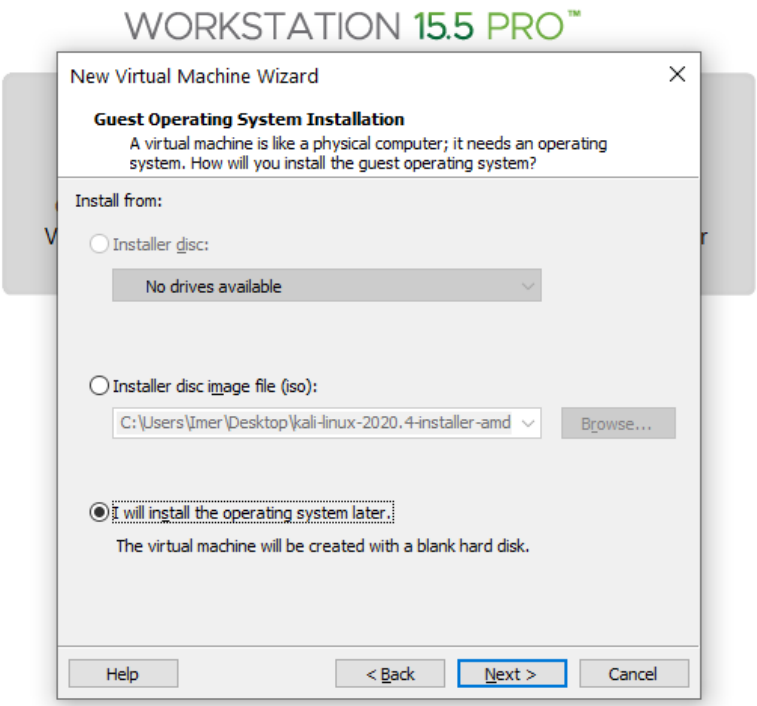


Figure 172- Selecting location to install the system from

7. The specifications

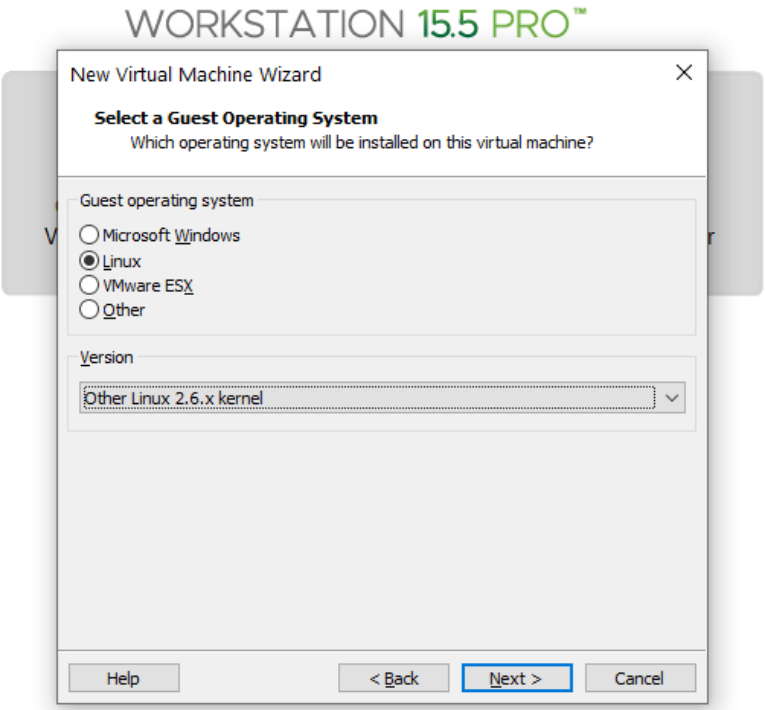


Figure 173- Selecting the guest operating system

8. The name and location

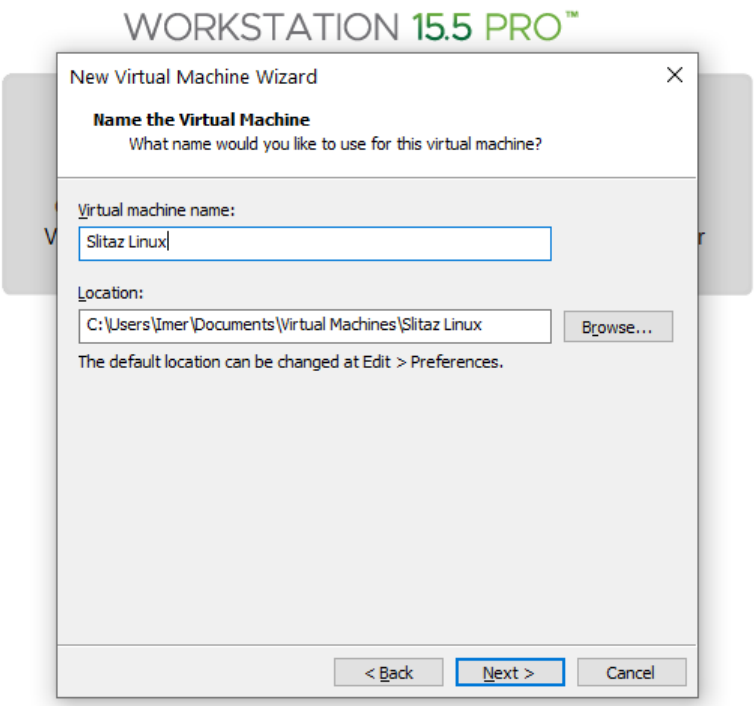


Figure 174- Specifying the name and location of the Virtual Machine.

9. The number of processors

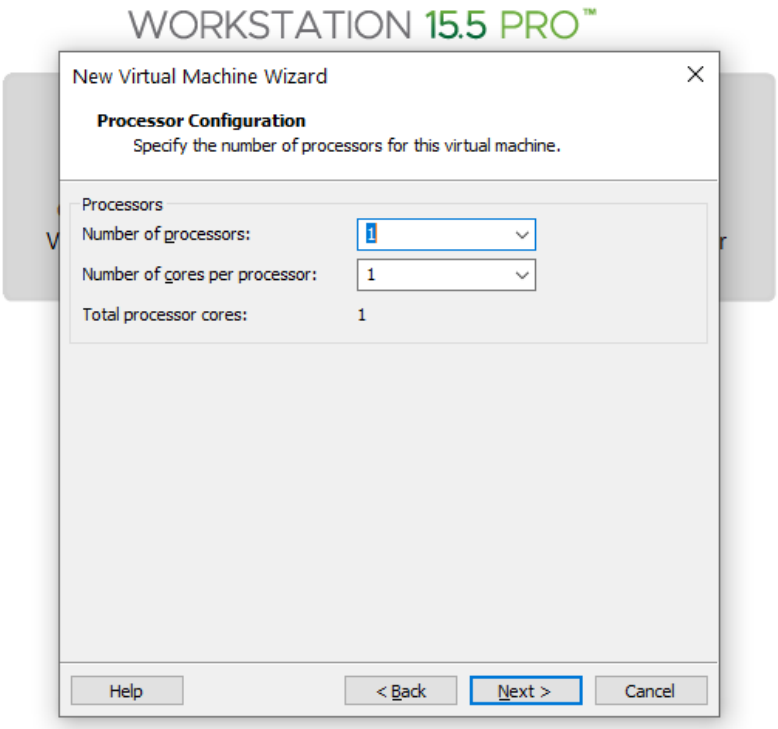


Figure 175- specifying the number of cores and procerssors

10. The size

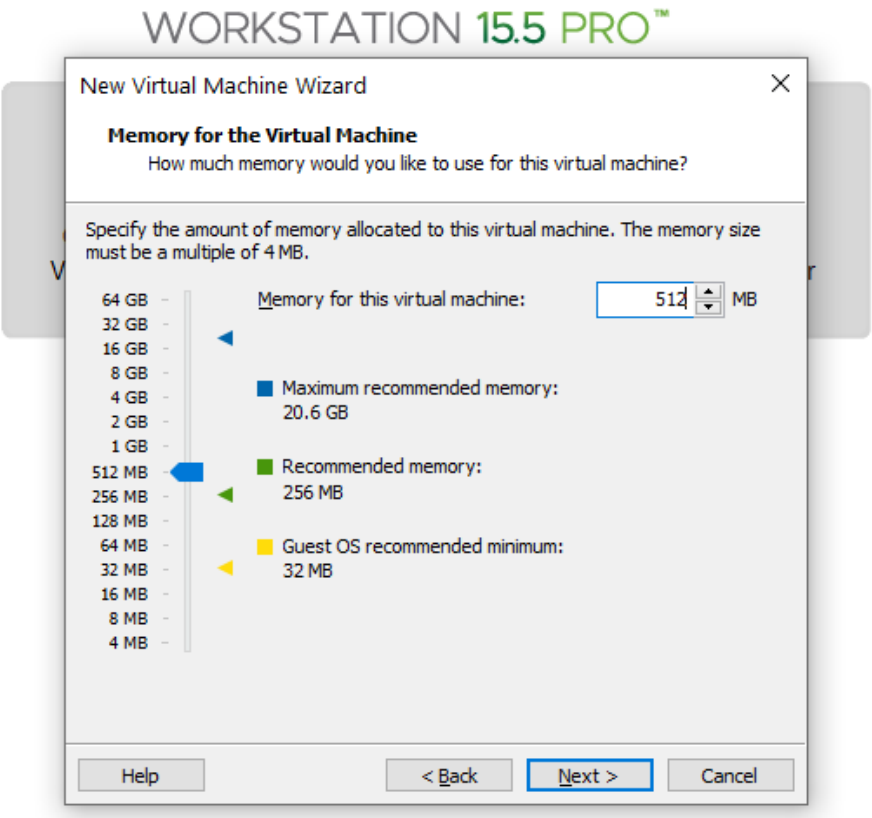


Figure 176- Specifying the size of the Virtual Machine

11. Network Type

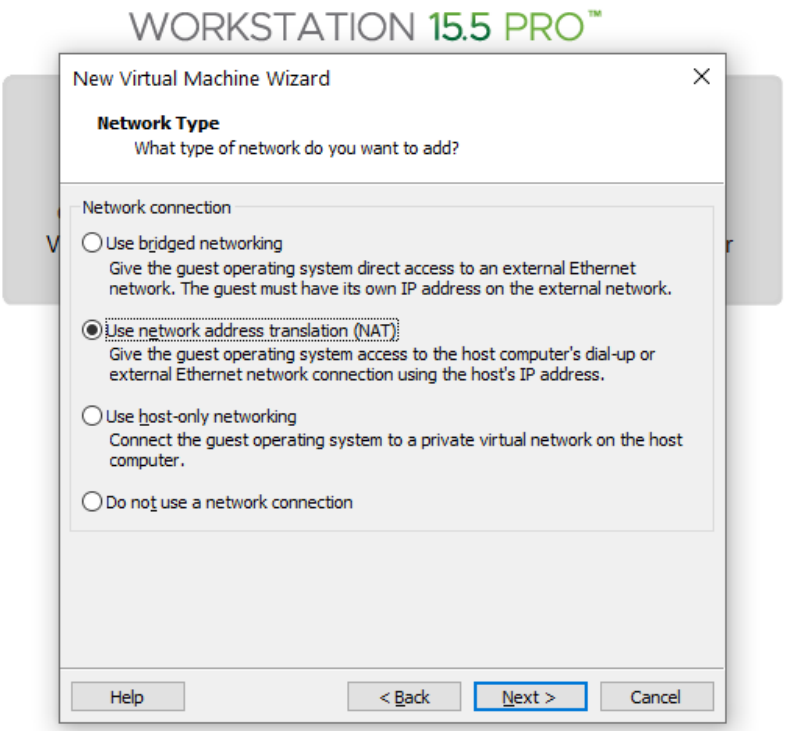


Figure 177- Select network type

12. I/O controller types

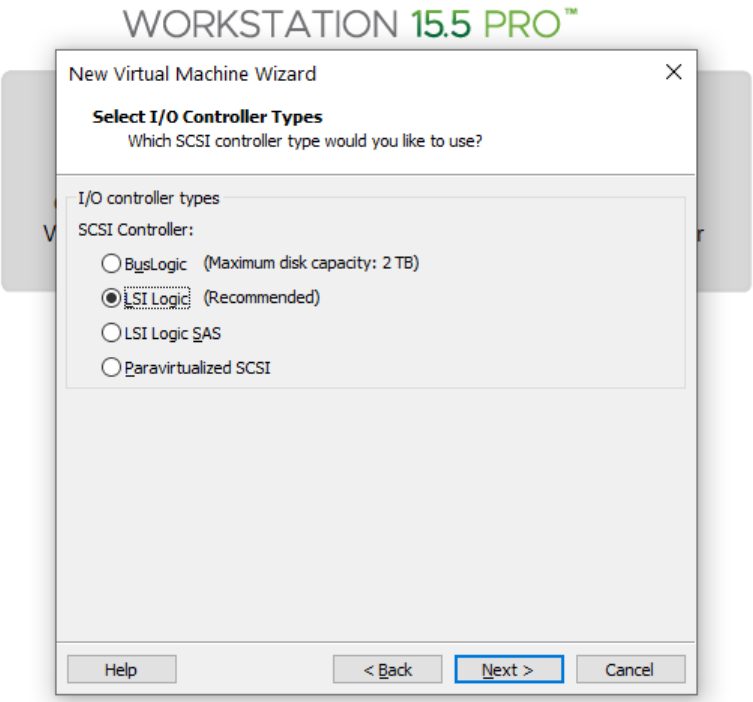


Figure 178- Select I/O controller type

13. Virtual Disk Type

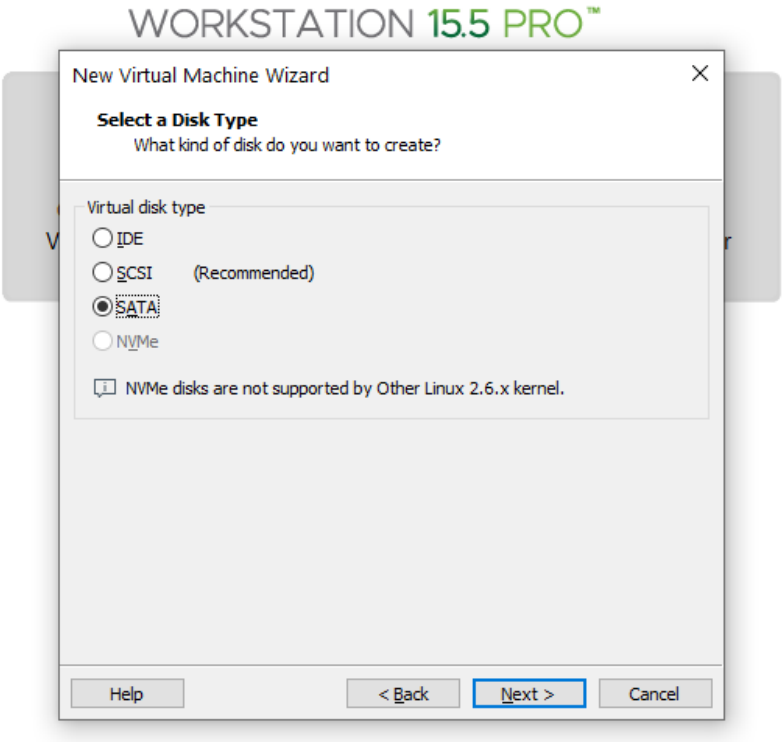


Figure 179- Select disk type

14. Select a disk

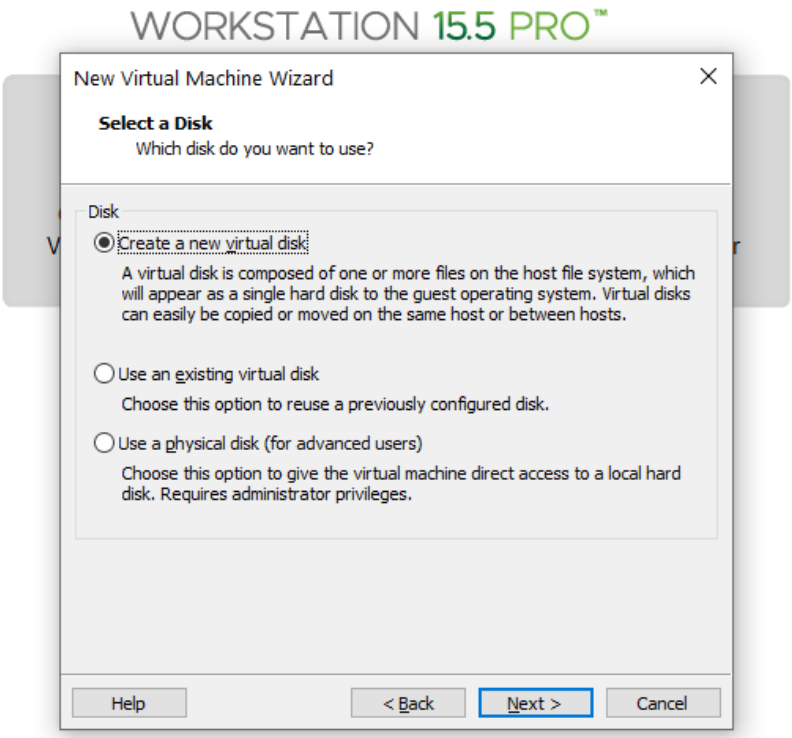


Figure 180- Selecting the disk to be used

15. Specify disk capacity

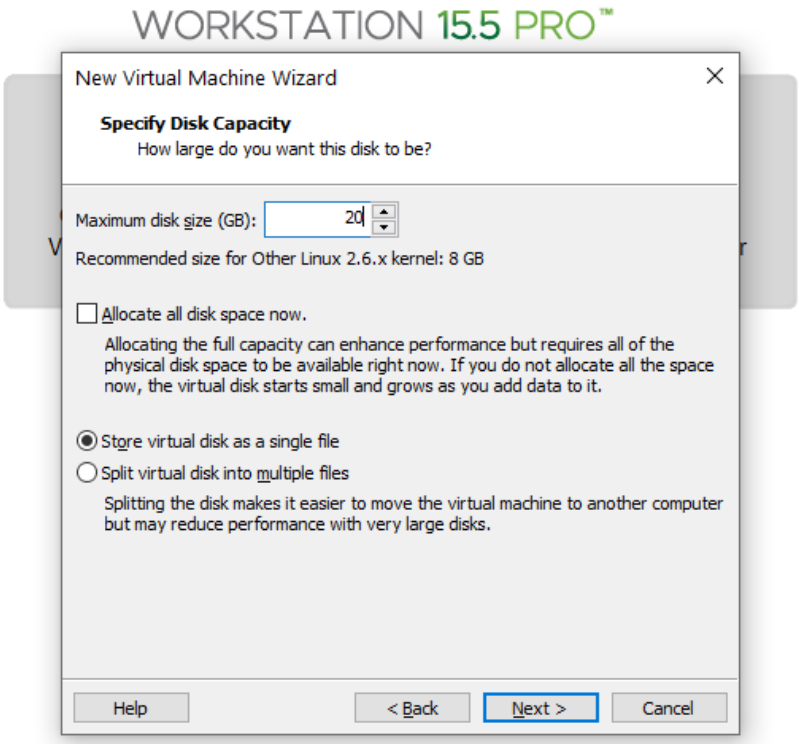


Figure 181- Specifying the disk capacity

16.

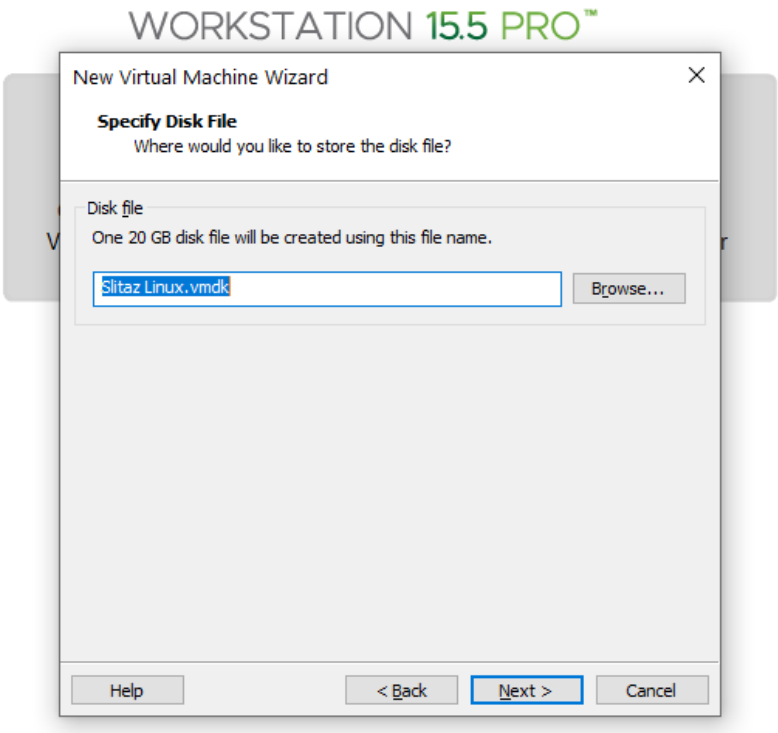


Figure 1820 Selecting the location to store file

17.

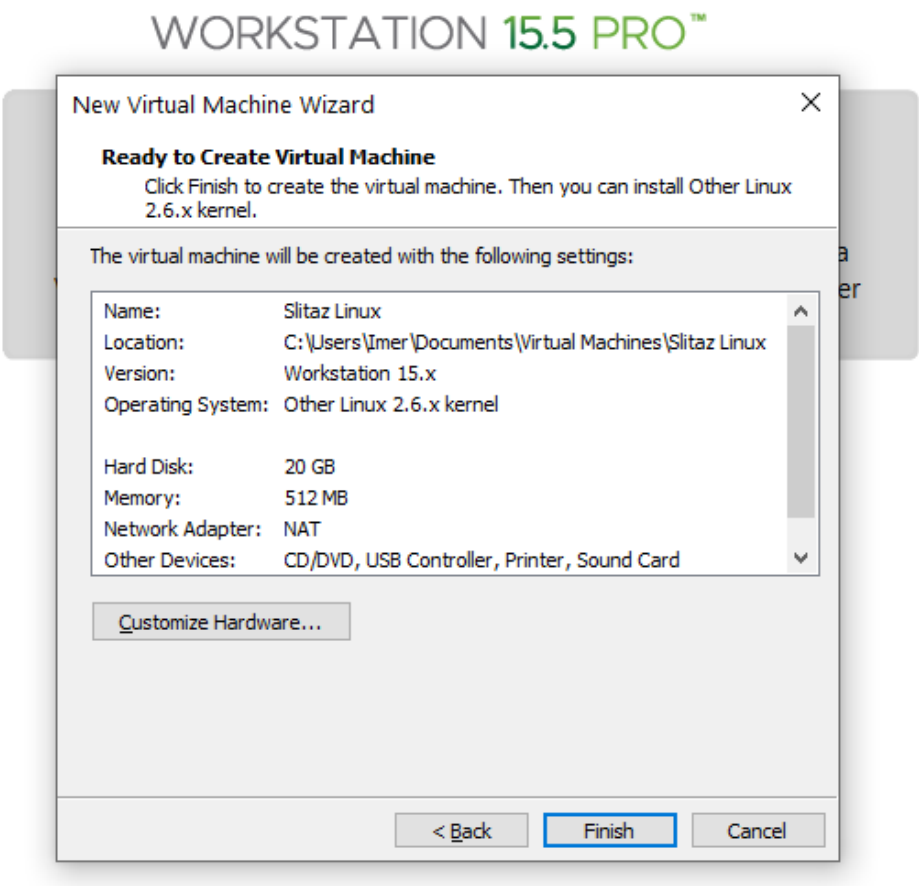


Figure 183- Virtual Machine specifications

18. Adding the file

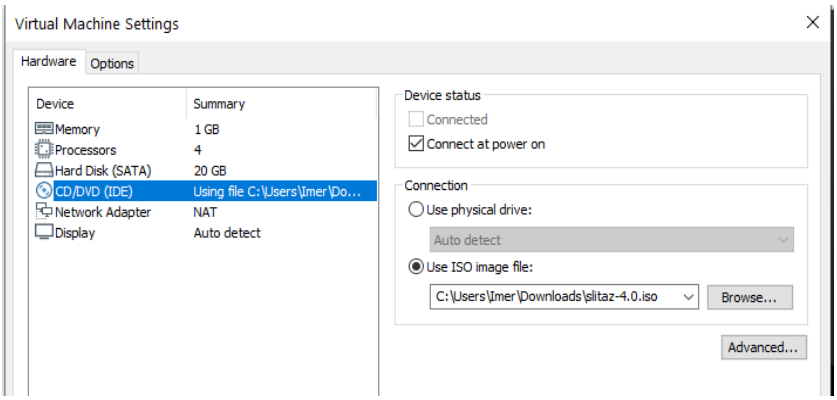


Figure 184- Adding the iso file for Slitaz

19.

Figure 185- Slitaz Desktop page

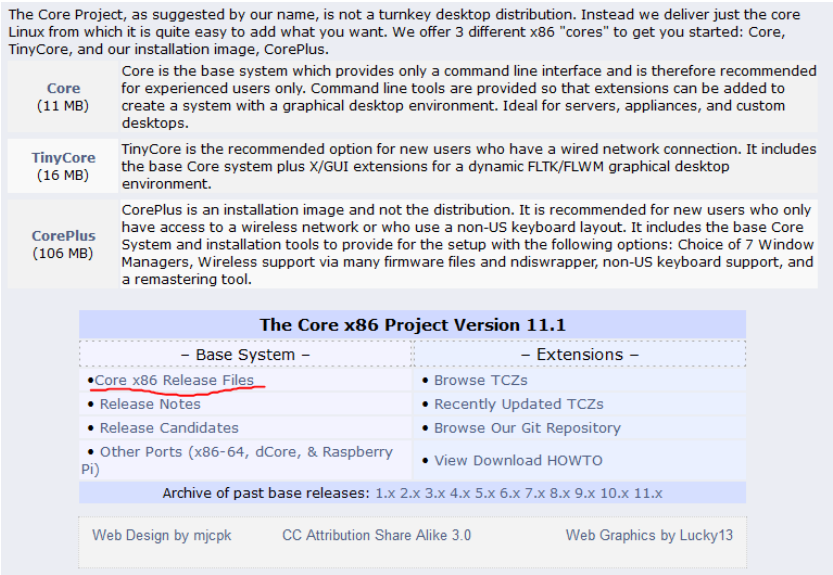
Installation Process of Tiny Core Linux

To install the tiny core linux in the VMware workstation,

1.The first step is to go to the website of tiny core linux

http://tinycorelinux.net/

2. Now I go to the downloads section of the website an select the “Core x86 Release Files”



The Core Project, as suggested by our name, is not a turnkey desktop distribution. Instead we deliver just the core Linux from which it is quite easy to add what you want. We offer 3 different x86 "cores" to get you started: Core, TinyCore, and our installation image, CorePlus.

Core (11 MB)	Core is the base system which provides only a command line interface and is therefore recommended for experienced users only. Command line tools are provided so that extensions can be added to create a system with a graphical desktop environment. Ideal for servers, appliances, and custom desktops.
TinyCore (16 MB)	TinyCore is the recommended option for new users who have a wired network connection. It includes the base Core system plus X/GUI extensions for a dynamic FLTK/FLWM graphical desktop environment.
CorePlus (106 MB)	CorePlus is an installation image and not the distribution. It is recommended for new users who only have access to a wireless network or who use a non-US keyboard layout. It includes the base Core System and installation tools to provide for the setup with the following options: Choice of 7 Window Managers, Wireless support via many firmware files and ndiswrapper, non-US keyboard support, and a remastering tool.

The Core x86 Project Version 11.1

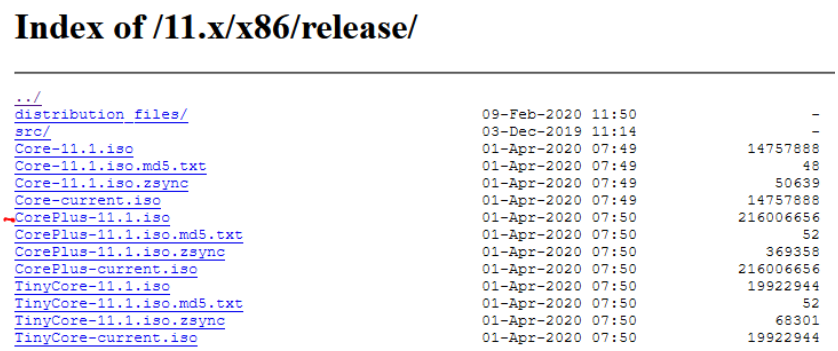
– Base System –	– Extensions –
•Core x86 Release Files	• Browse TCZs
• Release Notes	• Recently Updated TCZs
• Release Candidates	• Browse Our Git Repository
• Other Ports (x86-64, dCore, & Raspberry Pi)	• View Download HOWTO

Archive of past base releases: 1.x 2.x 3.x 4.x 5.x 6.x 7.x 8.x 9.x 10.x 11.x

Web Design by mjcpk CC Attribution Share Alike 3.0 Web Graphics by Lucky13

Figure 186- Tinycore Linux official site download page

3. Now I Download the CorePlus-11.1.iso



Index of /11.x/x86/release/

../		
distribution_files/	09-Feb-2020 11:50	-
src/	03-Dec-2019 11:14	-
Core-11.1.iso	01-Apr-2020 07:49	14757888
Core-11.1.iso.md5.txt	01-Apr-2020 07:49	48
Core-11.1.iso.zsync	01-Apr-2020 07:49	50639
Core-current.iso	01-Apr-2020 07:49	14757888
CorePlus-11.1.iso	01-Apr-2020 07:50	216006656
CorePlus-11.1.iso.md5.txt	01-Apr-2020 07:50	52
CorePlus-11.1.iso.zsync	01-Apr-2020 07:50	369358
CorePlus-current.iso	01-Apr-2020 07:50	216006656
TinyCore-11.1.iso	01-Apr-2020 07:50	19922944
TinyCore-11.1.iso.md5.txt	01-Apr-2020 07:50	52
TinyCore-11.1.iso.zsync	01-Apr-2020 07:50	68301
TinyCore-current.iso	01-Apr-2020 07:50	19922944

Figure 187- Tinycore Linux download releases

4. Next step is to go to virtual machine and “Create a new Virtual Machine”

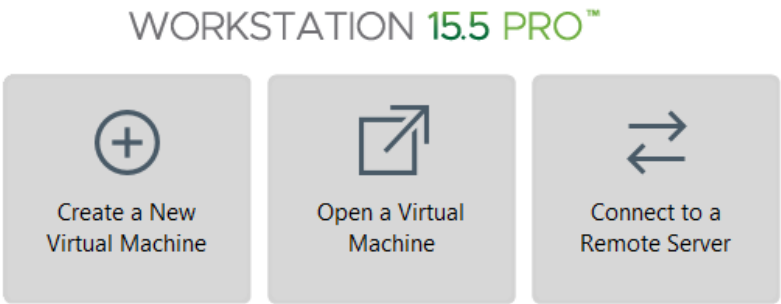


Figure 188- Create a new virtual machine

5. Select the “Custom” Option

Figure 189- Selecting the Configuration type

6. Click Next

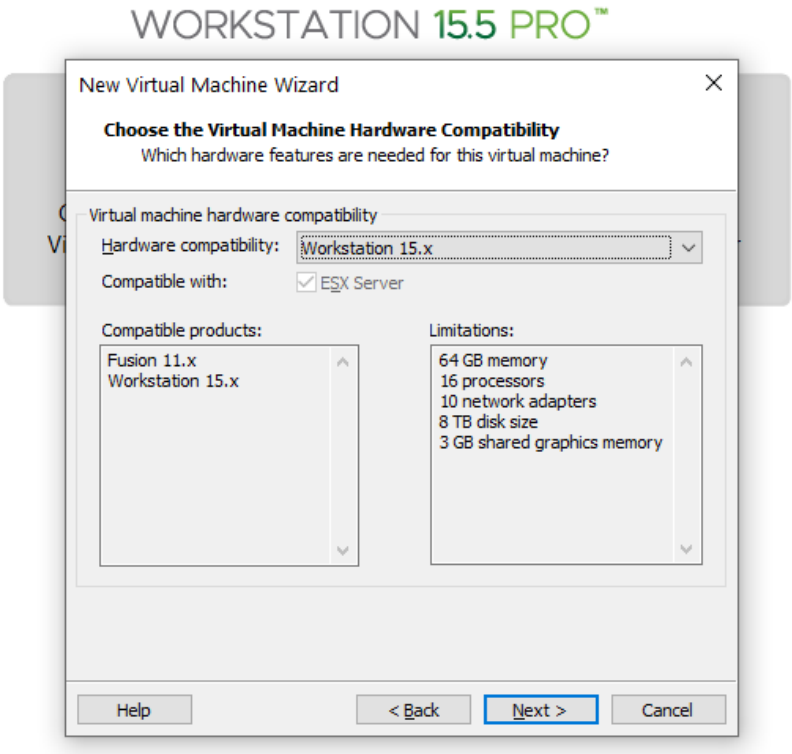


Figure 190- Virtual machine hardware compatibility

7. Select the last option

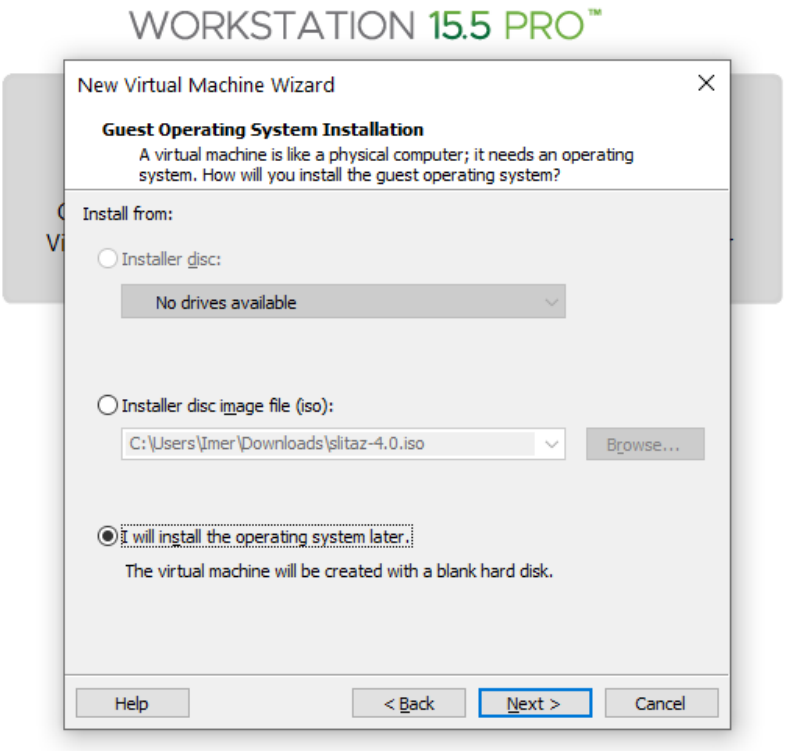


Figure 191- Selecting where to install the system from

8. Select the system and its type

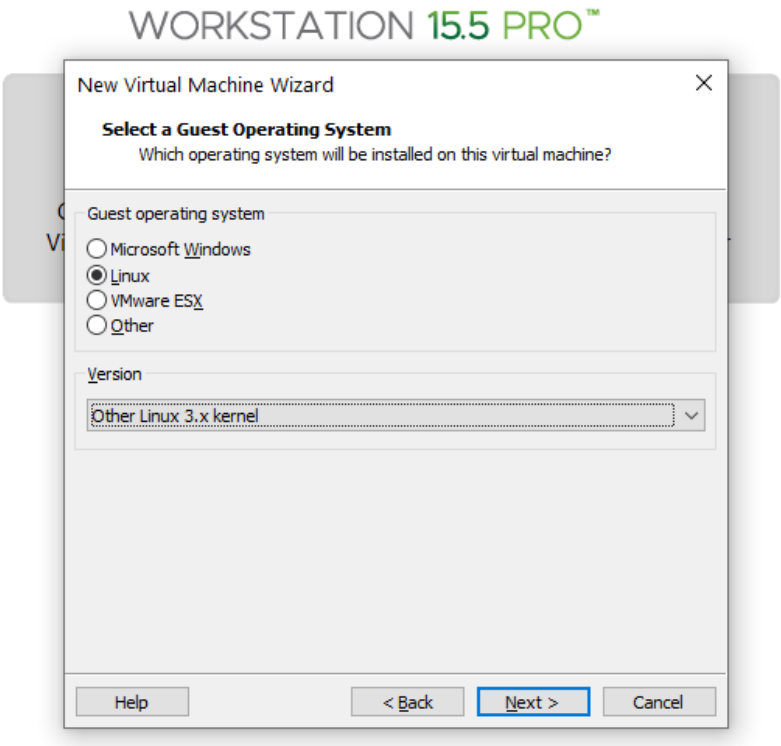


Figure 192- Selecting the guest operating system

9. Specify the name

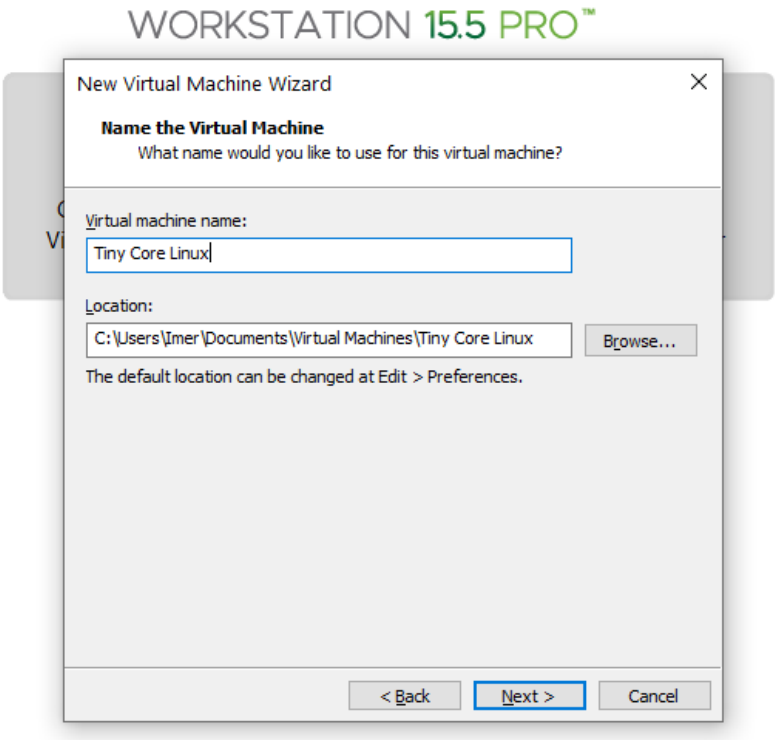


Figure 193- Specifying the virtual machine name and location

10. Select number of processors

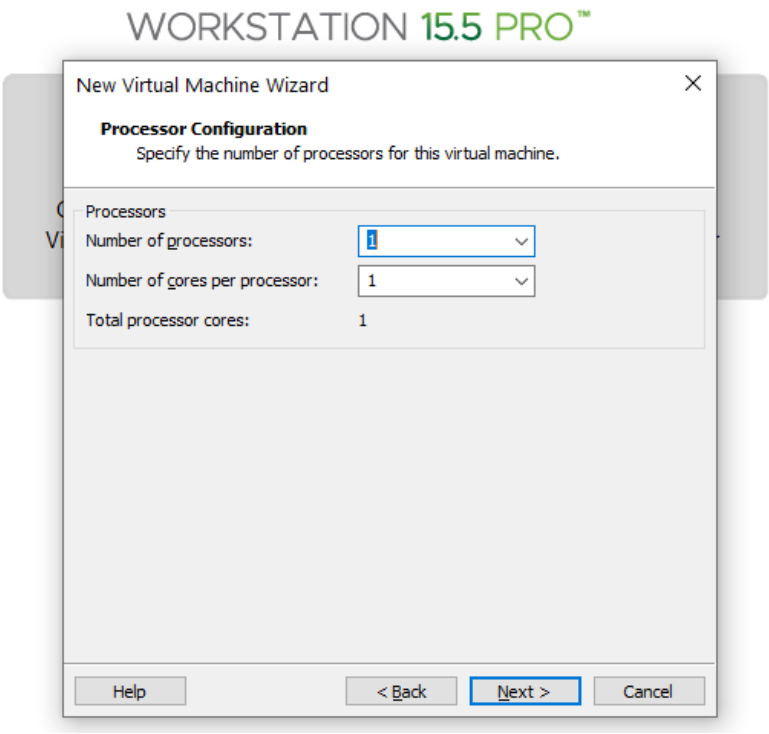


Figure 194- specifying the number of processors and cores

11. Memory of virtual machine

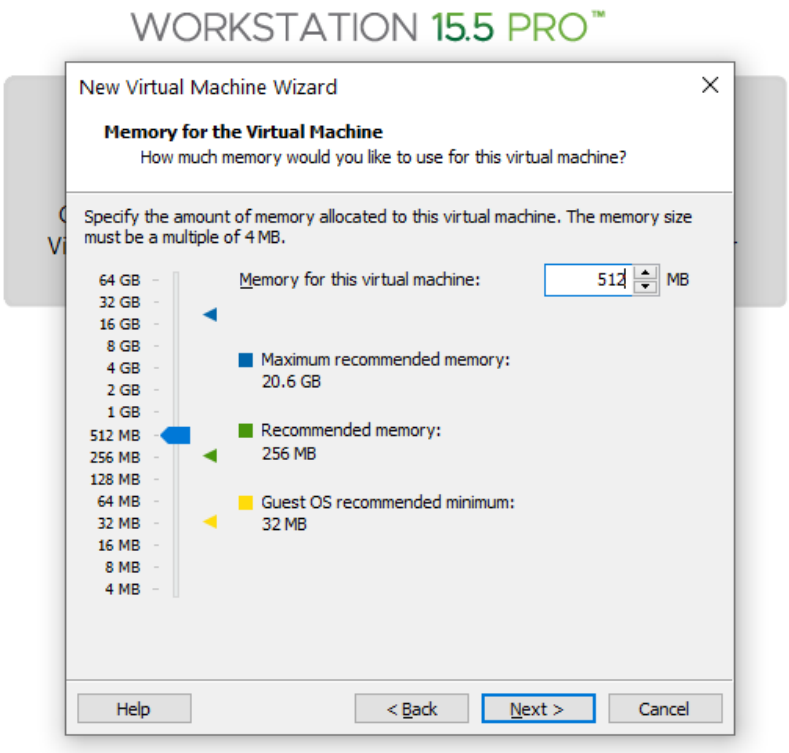


Figure 195- Specifying the memory of virtual machine

12. Select NAT

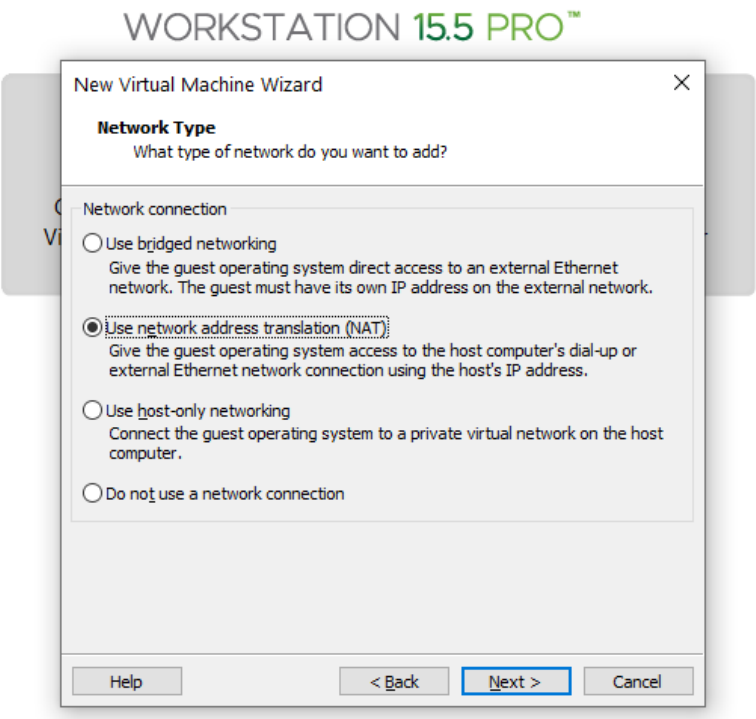


Figure 196- Selecting the network type

13. I/O Controller type

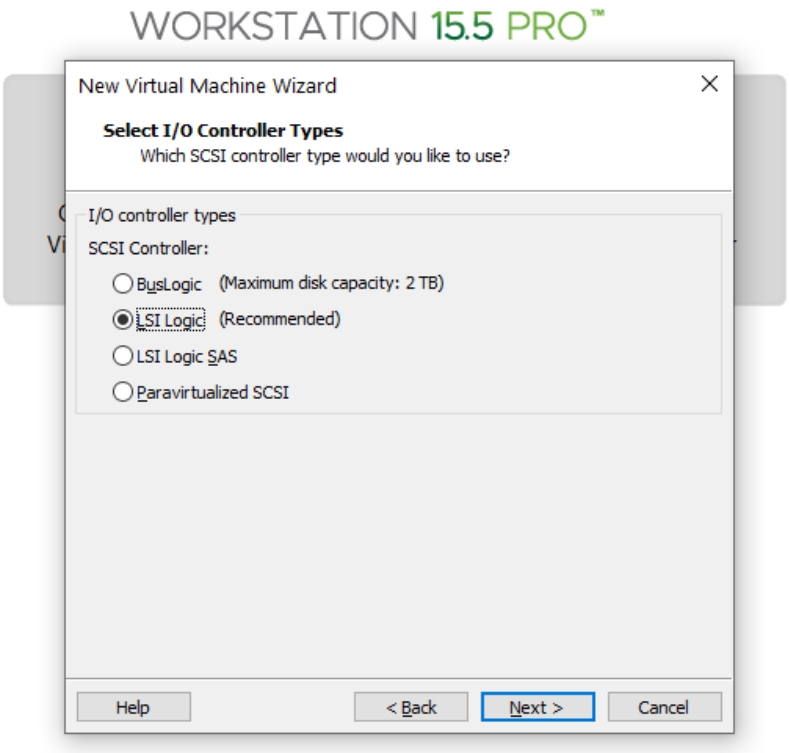


Figure 197- Selecting I/O controller type

14. Select Disk Type

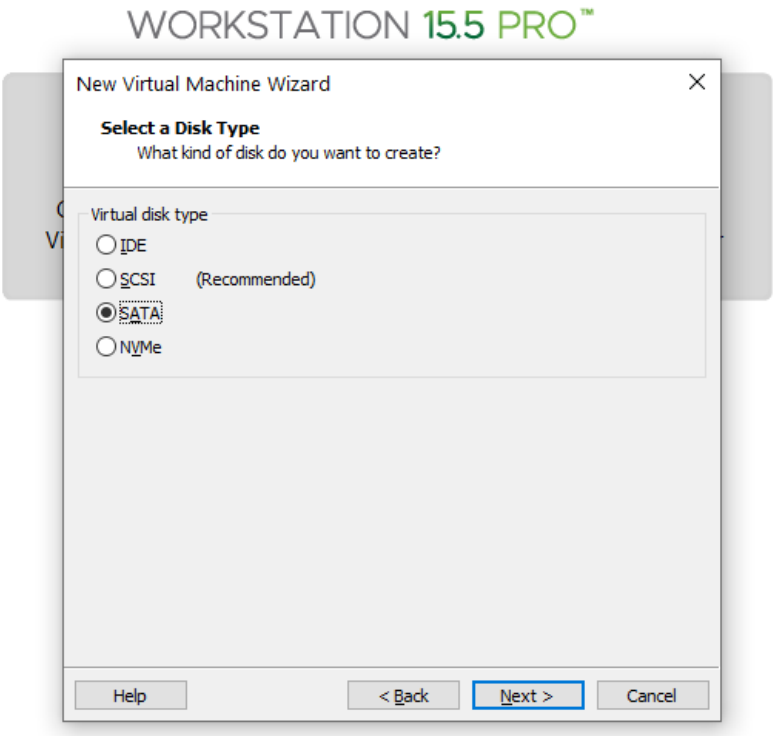


Figure 198- Selecting the disk type

15. Select Disk

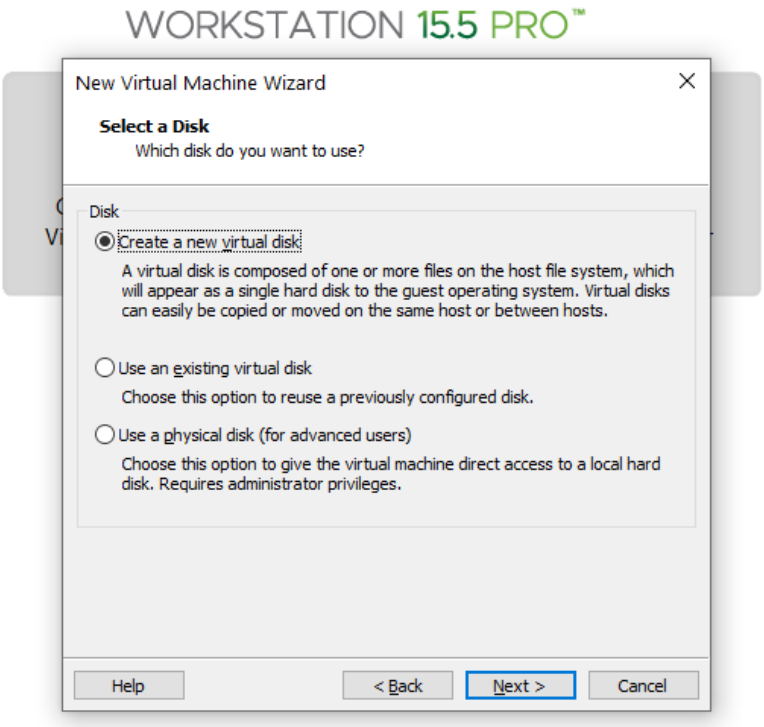


Figure 199- specifying which disk to use

16. Disk Capacity

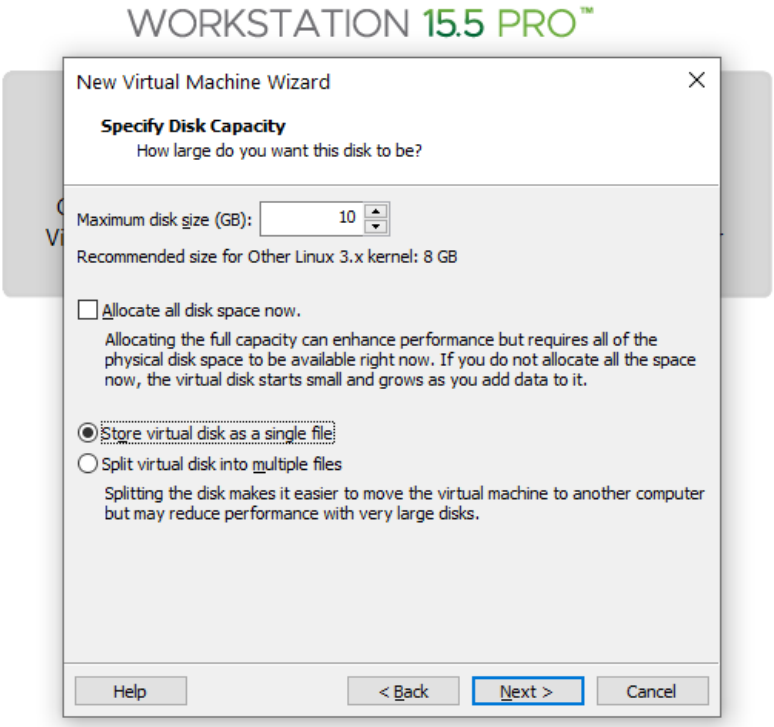


Figure 200- Specifying disk capacity

17. Next

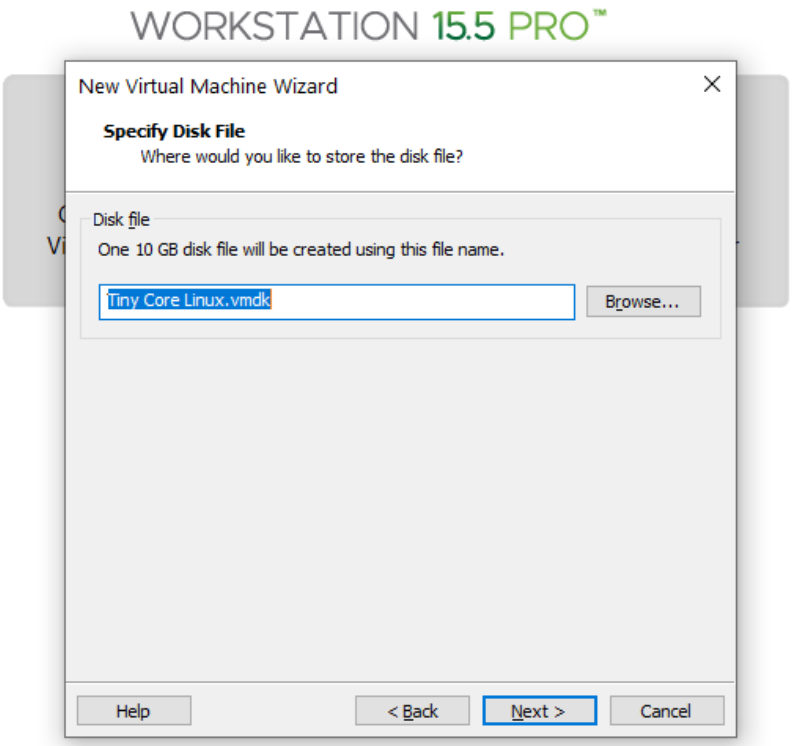


Figure 201- Specifying the disk file

18. Customize Hardware

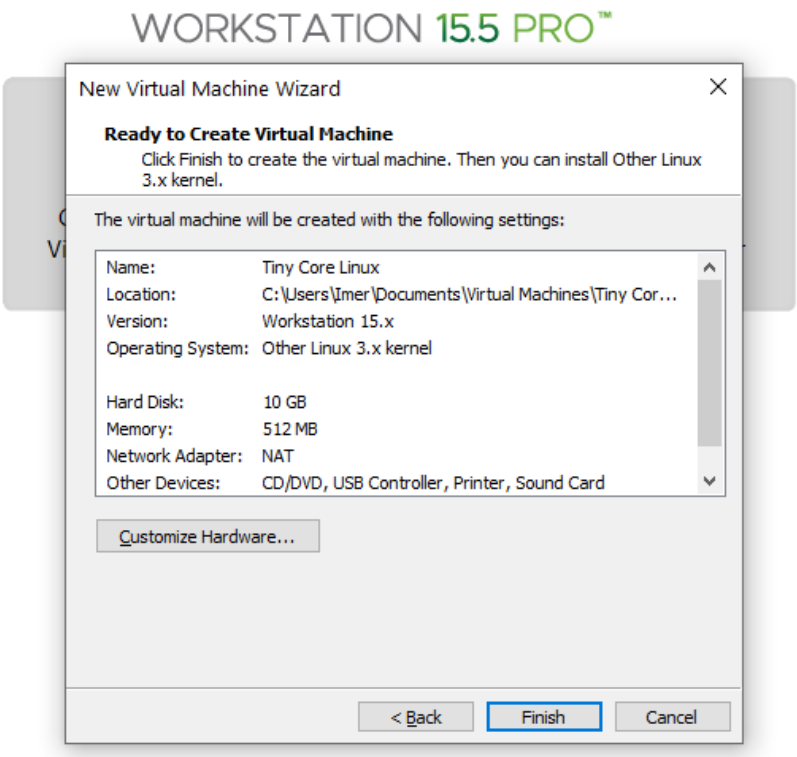


Figure 202- Virtual machine settings

19. Use the iso file downloaded earlier

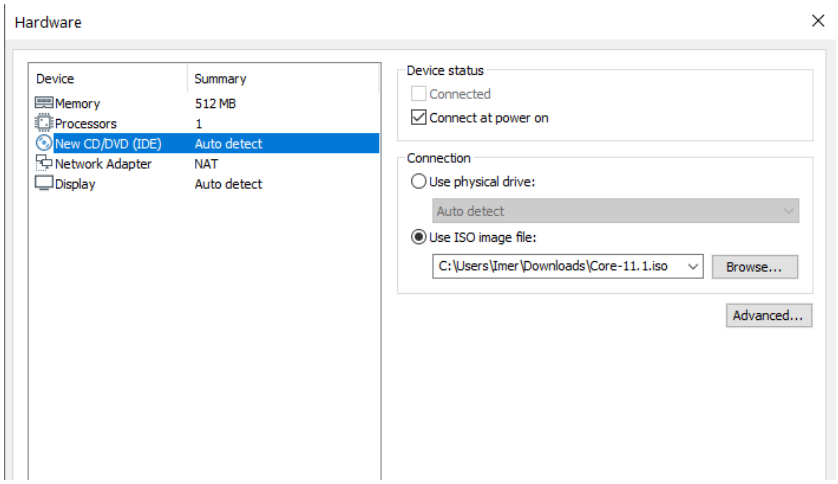


Figure 203- Adding the Tinycore downloaded iso file

Now I power on the virtual machine and it loads into a command prompt. This version does not have a GUI.

Installation Process of Porteus Linux

To install the porteus Linux system on the VMware work station,

1. Go to the official website of porteus

http://www.porteus.org/

2. In the main page

https://forum.porteus.org/viewtopic.php?t=9179

Figure 204- Porteus Linux download page

3. Following that button, I selected one mirror, went to the x86/64 section, version 5 rc2 and the first iso file

The size of the file is 347 MB

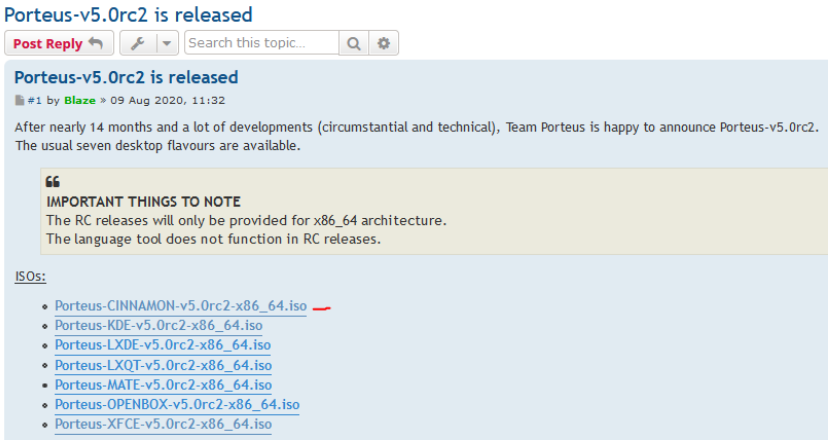


Figure 205- Porteus Linux download releases

4. I go to the VMware workstation and “Create a new virtual machine”

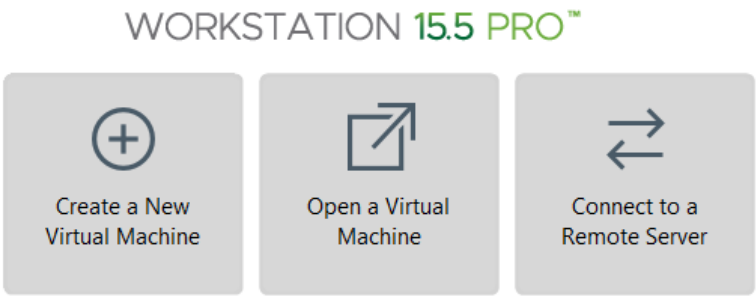


Figure 206- Create a new virtual machine

5. I select the “typical” option

Figure 207- Selecting the configuration type

6. Selecting the last option

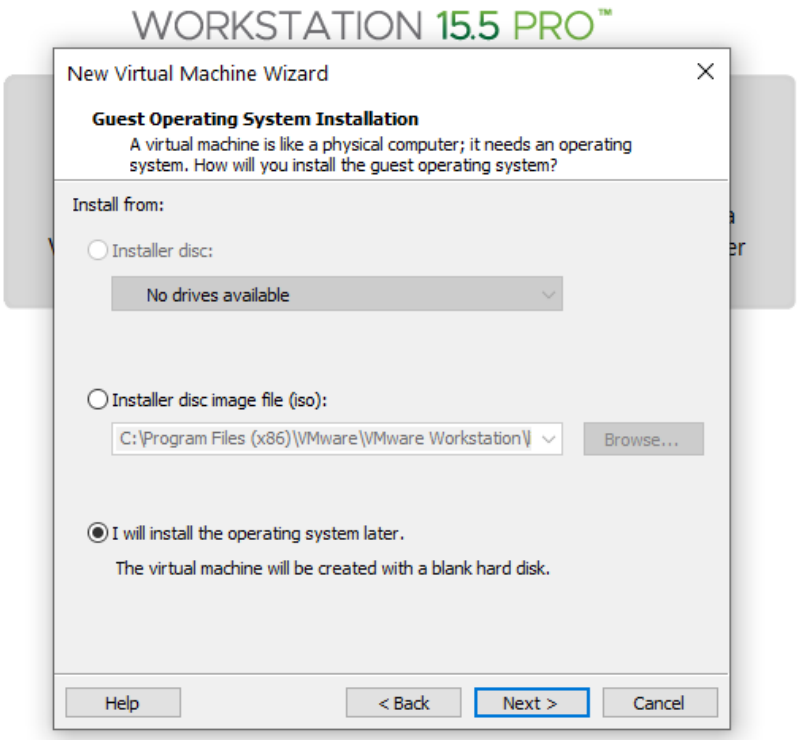


Figure 208- Selecting where to install the system from

7. specifying the guest operating system type and the version

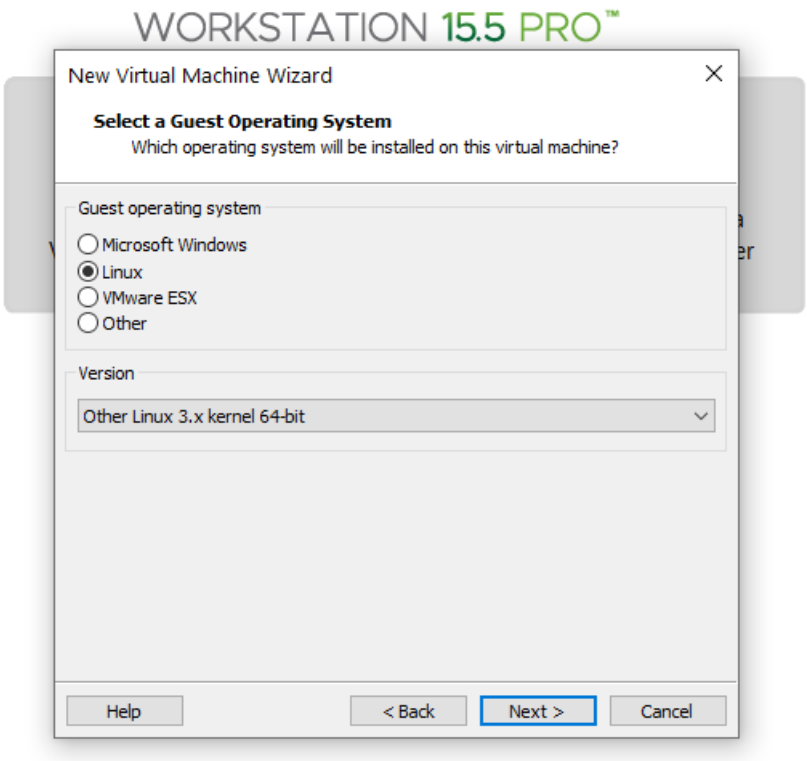


Figure 209- Selecting the guest operating system

8. Name of the machine and its location

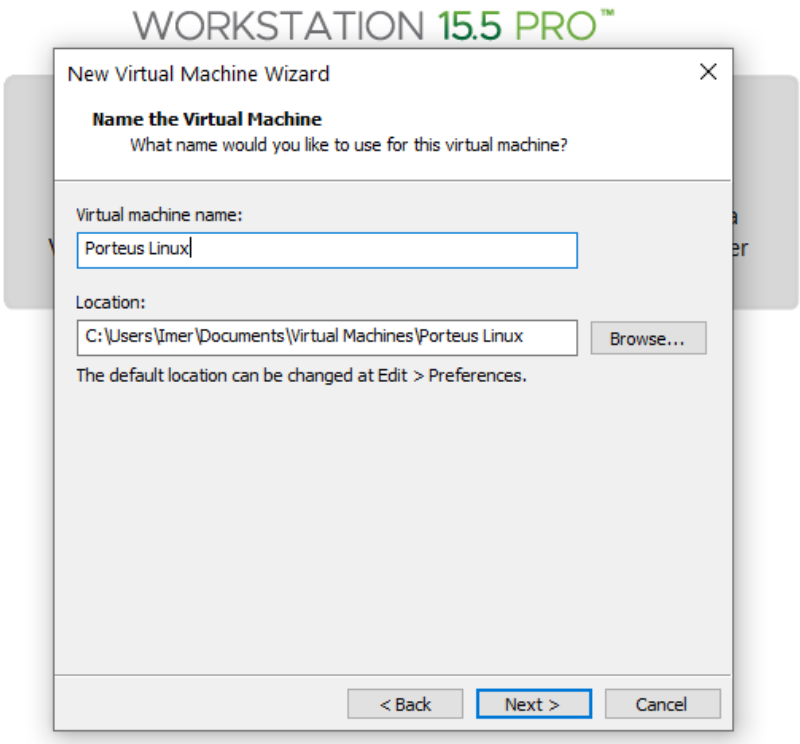


Figure 210- Specifying the name of the virtual machine and its location

9. Specifying the capacity of the disk and storing it as a single or multiple file

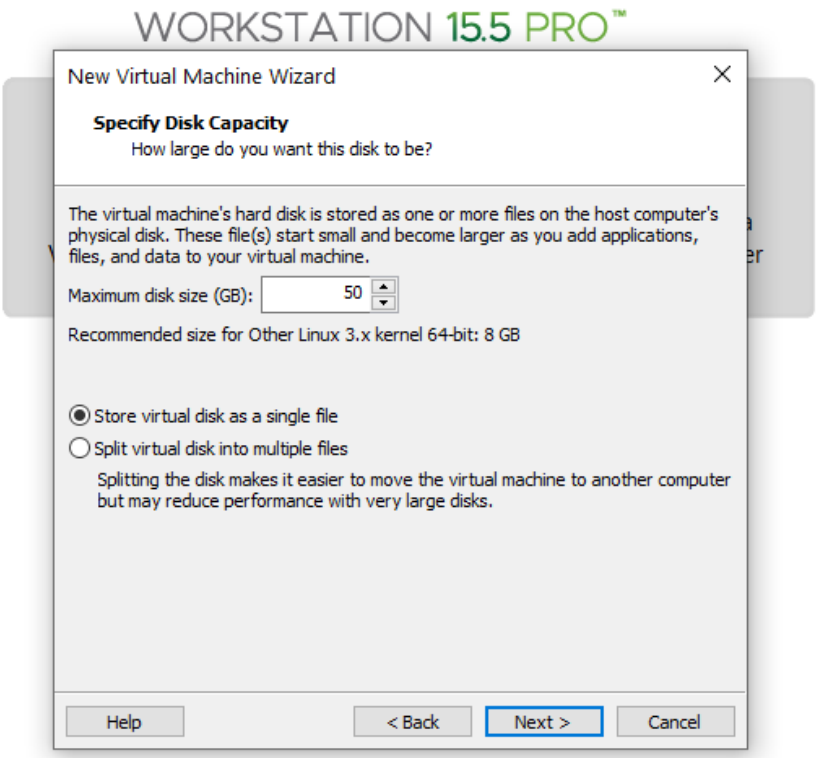


Figure 211- specifying the disk capacity

10. Customize the hardware

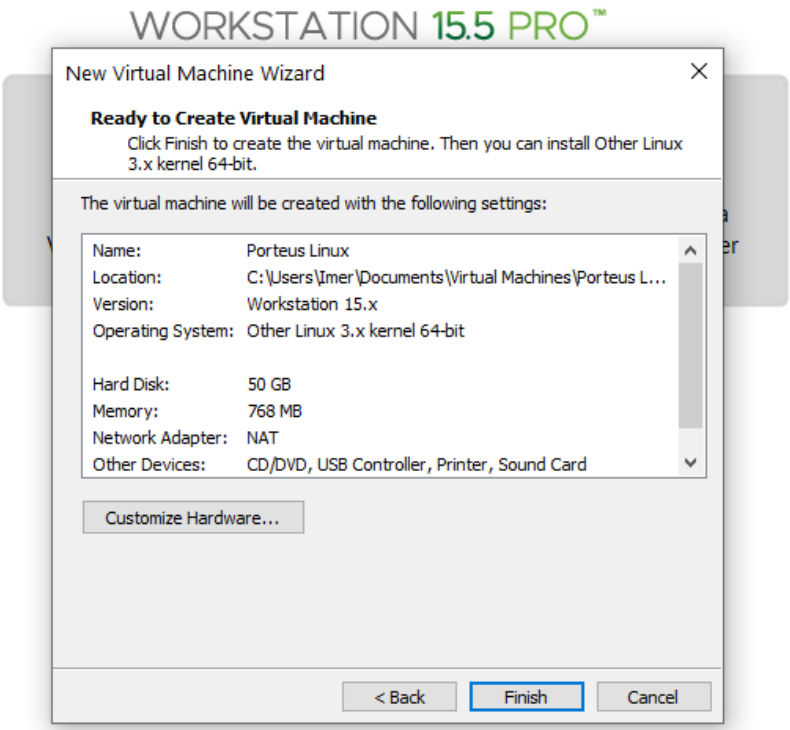


Figure 212- Virtual machine settings

11. Memory as 2 GB

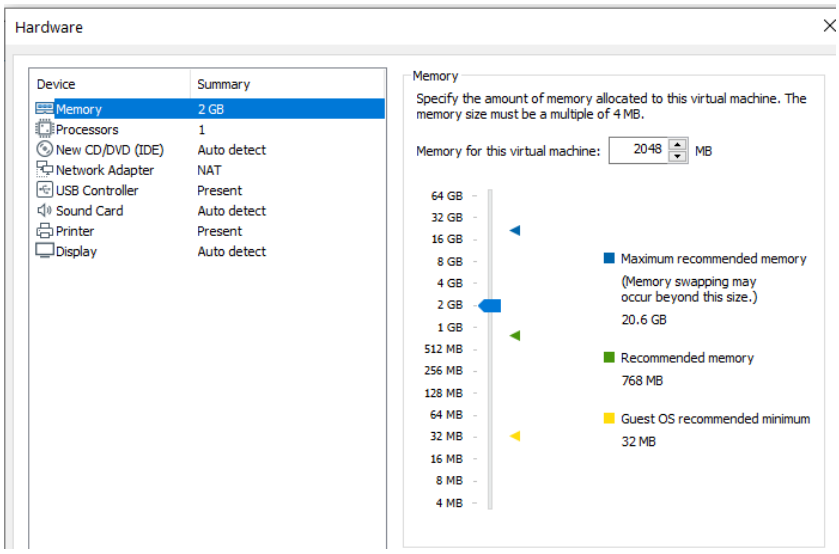


Figure 213- Specifying the memory

12. Number of processors

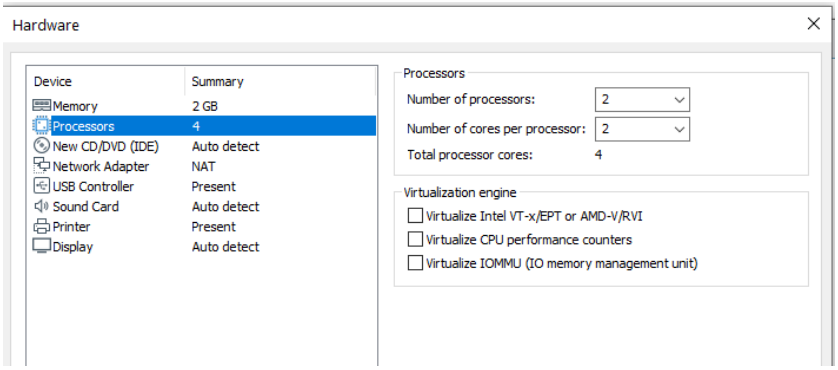


Figure 214- Specifying the number of processors and cores

13. Uploading the iso file

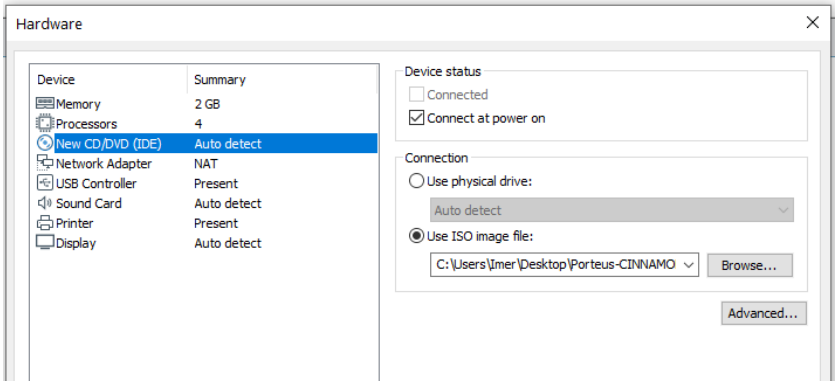


Figure 215- Adding the iso file of the Porteus Linux

Installation Process of SilverBlue Linux

To install the Silver blue Linux on the VMware workstation,

1.Go to the main website of fedora Silver blue and download the iso file of the latest version

Size: 2.67GB

https://silverblue.fedoraproject.org/download

2. Go to the VMware workstation and “Create a New Virtual Machine”

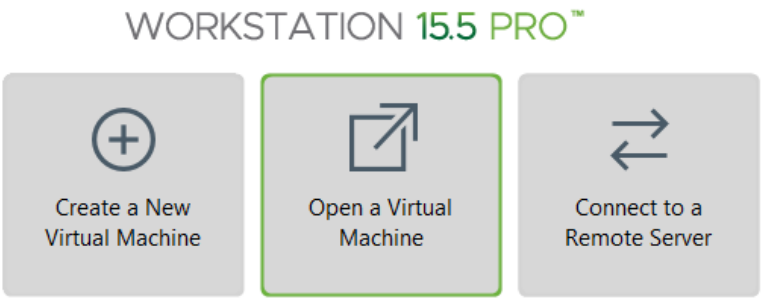


Figure 216- create a new virtual machine

3. Typical machine

Figure 217- Specifying the configuration type

4. Last option

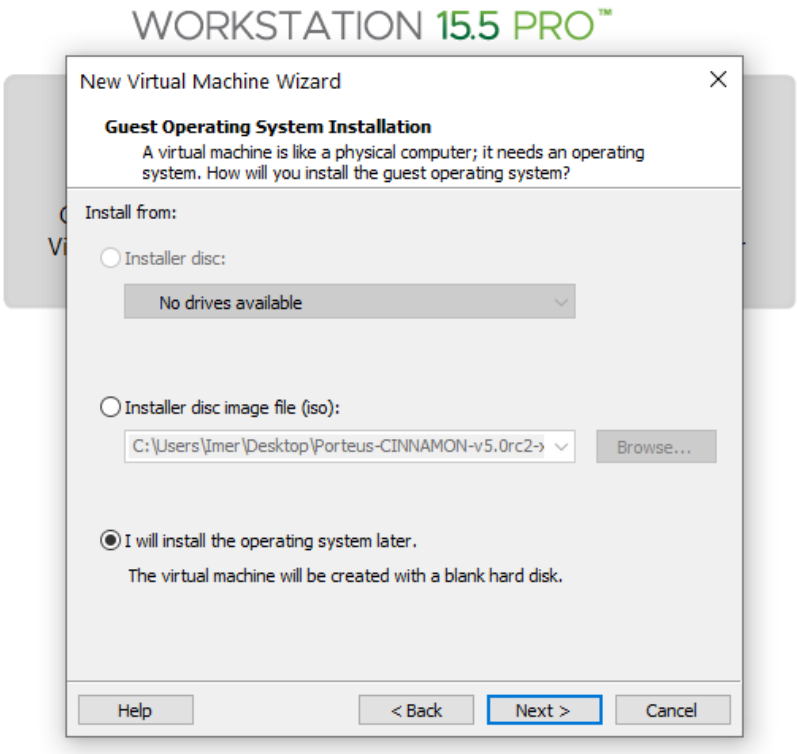


Figure 218- Selecting the configuration type

5. Guest OS and version

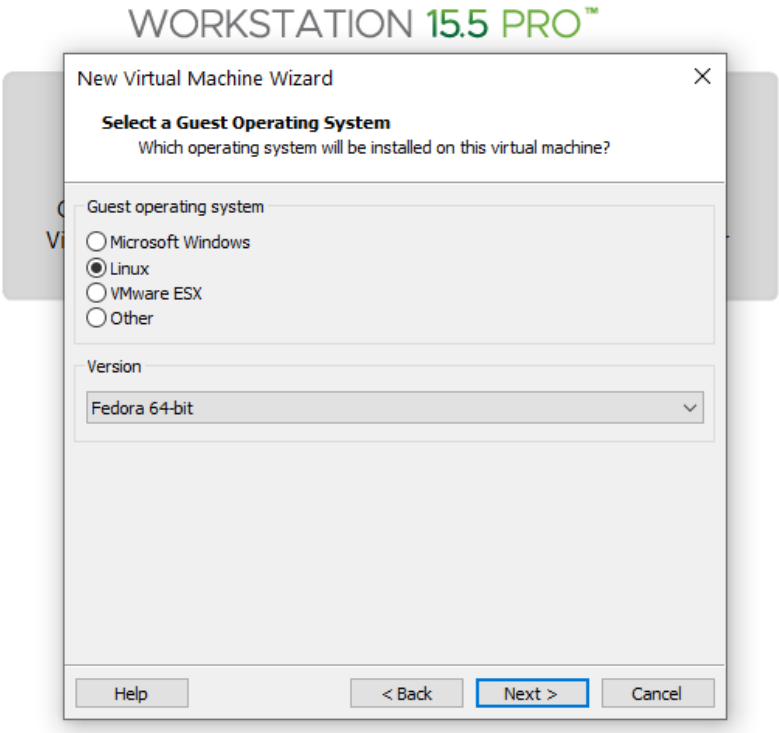


Figure 219- Selecting the guest operating system

6. Virtual machine name

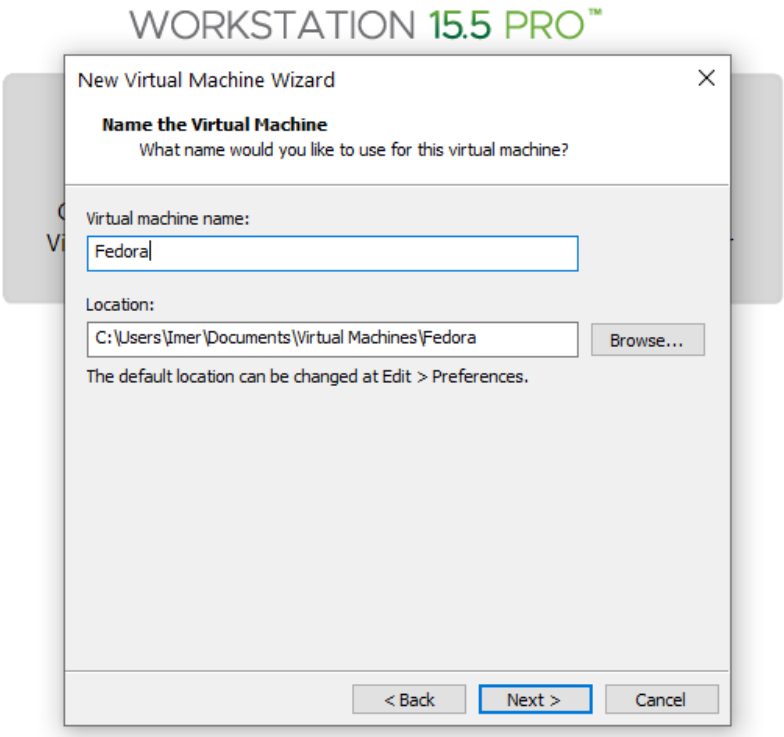


Figure 220- Specifying the virtual machine name and location

7. Disk Capacity

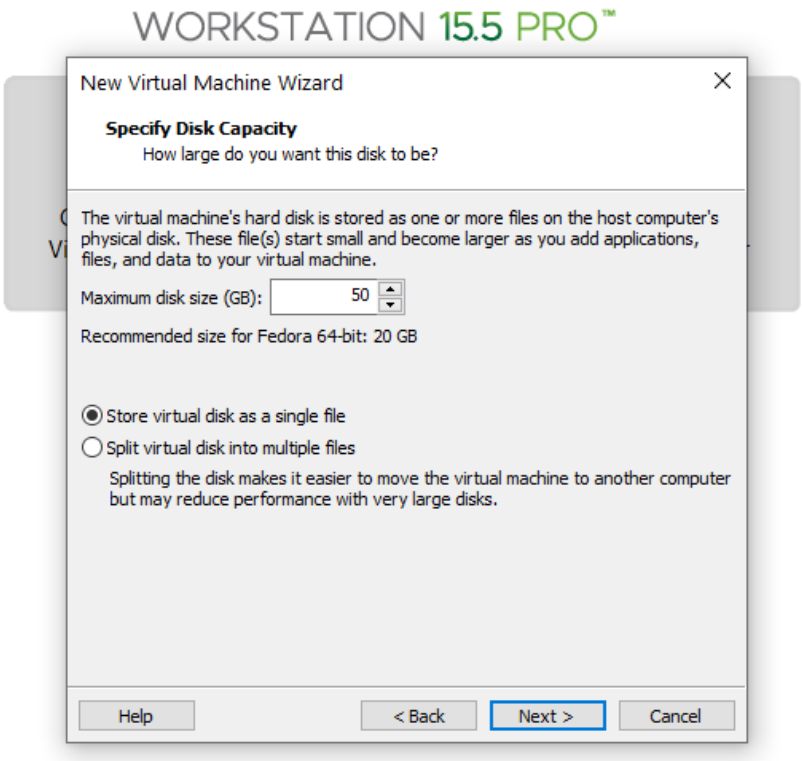


Figure 221- specifying the disk capacity

8. Customize Hardware.

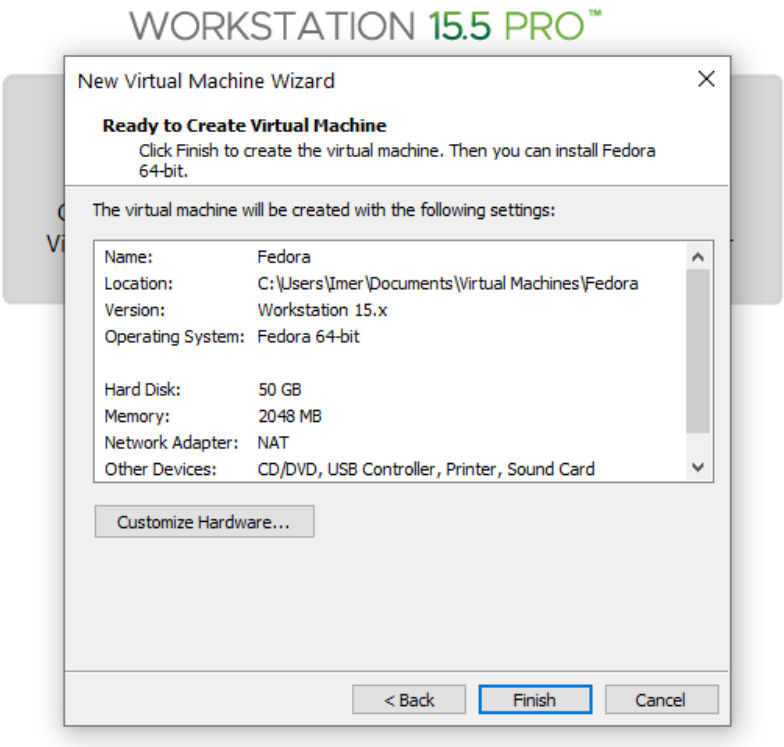


Figure 222- Virtual machine settings

9. The iso file

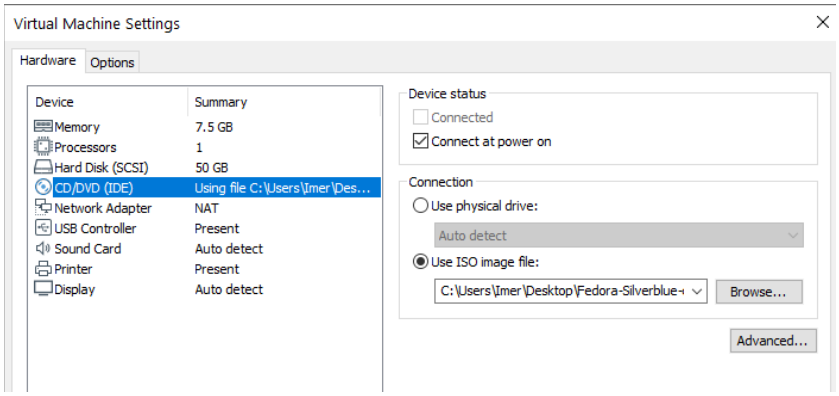


Figure 223- Adding the Silverblue Linux downloaded iso file