# Required Tools

## Introduction

Before beginning the malware development journey, it is necessary to prepare the development workspace by installing malware development and reverse engineering tools. These tools will aid one in the development and analysis of the malware and will be used throughout the modules.

## Reverse Engineering Tools

Several of the tools mentioned focus more on reverse engineering rather than development. It is essential to reverse engineer the malware built to fully understand its internal workings and have an understanding of what the malware analysts will see upon inspecting the malware.

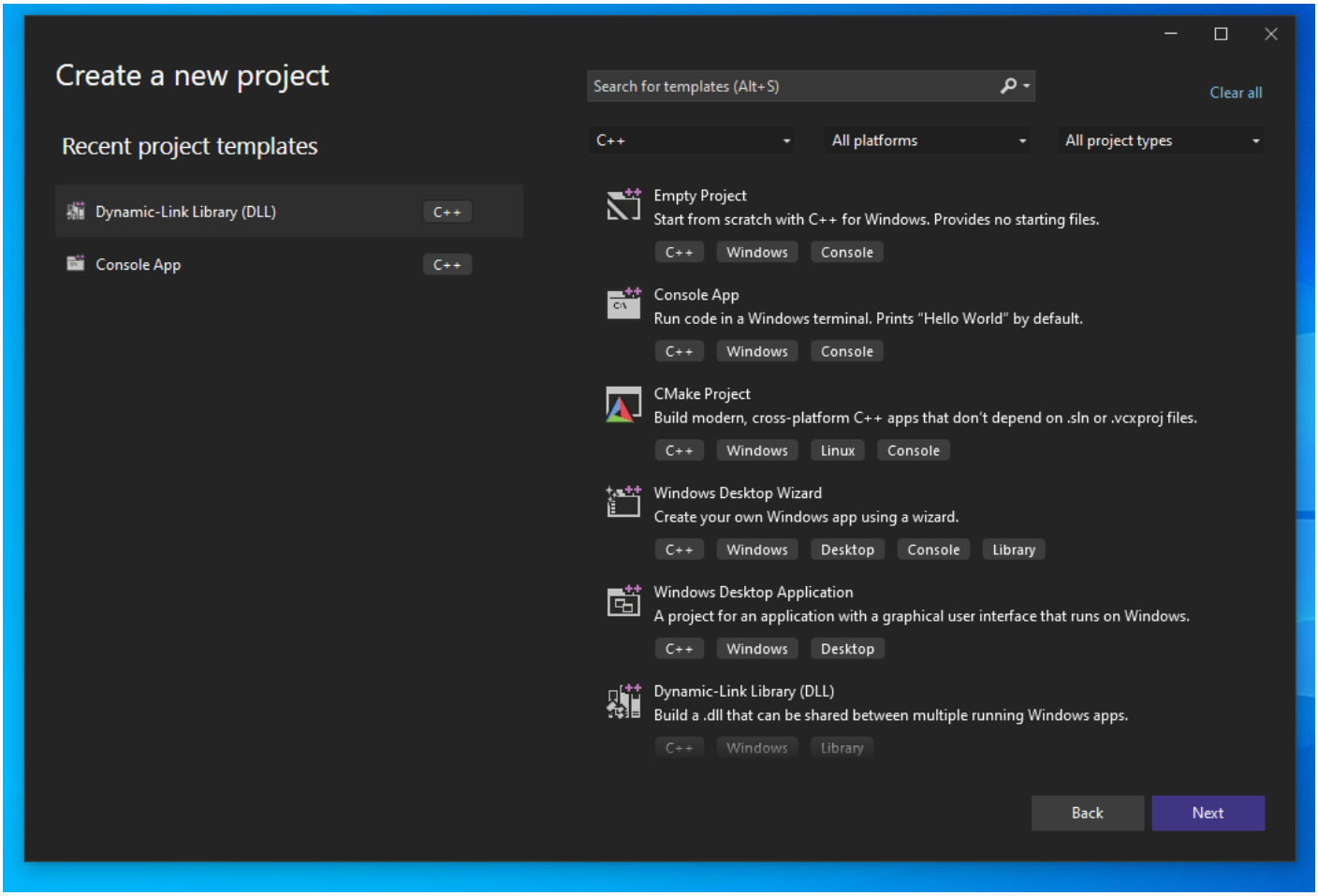
## Tools To Install

Install the following tools:

- Visual Studio - This is the development environment where the coding & compiling process will occur. Install the C/C++ Runtime.
- x64dbg - x64dbg is a debugger that will be used throughout the modules to get an internal understanding of the developed malware.
- PE-Bear - PE-bear is a multiplatform reversing tool for PE files. It will also be used to assess the developed malware and look for suspicious indicators.
- Process Hacker 2 - Process Hacker is a powerful, multi-purpose tool that helps monitor system resources, debug software and detect malware.
- Msfvenom - Msfvenom is a command line interface tool that is used to create, manipulate, and output payloads.

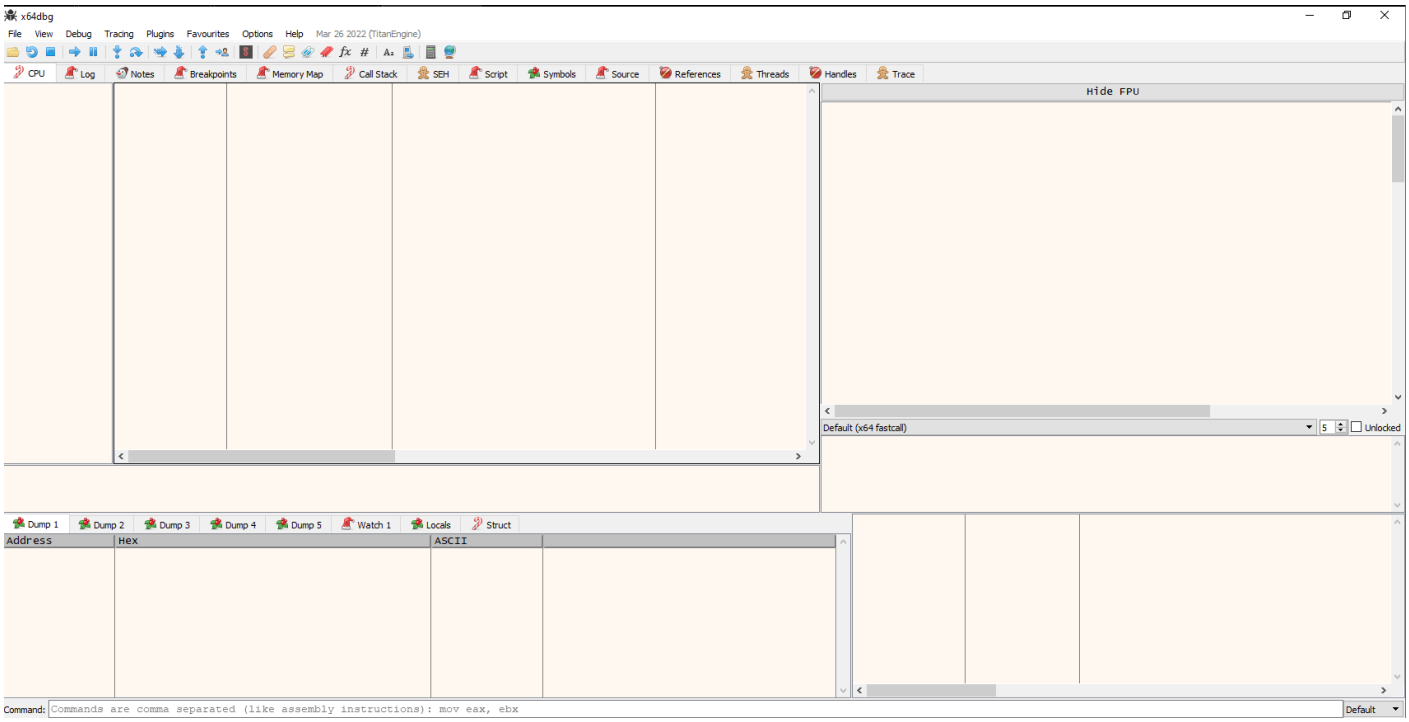
## Visual Studio

Visual Studio is an integrated development environment (IDE) developed by Microsoft. It is used to develop a wide array of software such as web applications, web services and computer programs. It also comes with development and debugging tools for building and testing applications. Visual Studio will be the main IDE used for development in this course.




## x64dbg

x64dbg is an open-source debugging utility for x64 and x86 Windows binaries. It is used to analyze and debug user-mode applications and kernel-mode drivers. It provides a graphical user interface that allows users to inspect and analyze the state of their programs and view memory contents, assembly instructions, and register values. With x64dbg, users can set breakpoints, view stack and heap data, step through code, and read and write memory values.



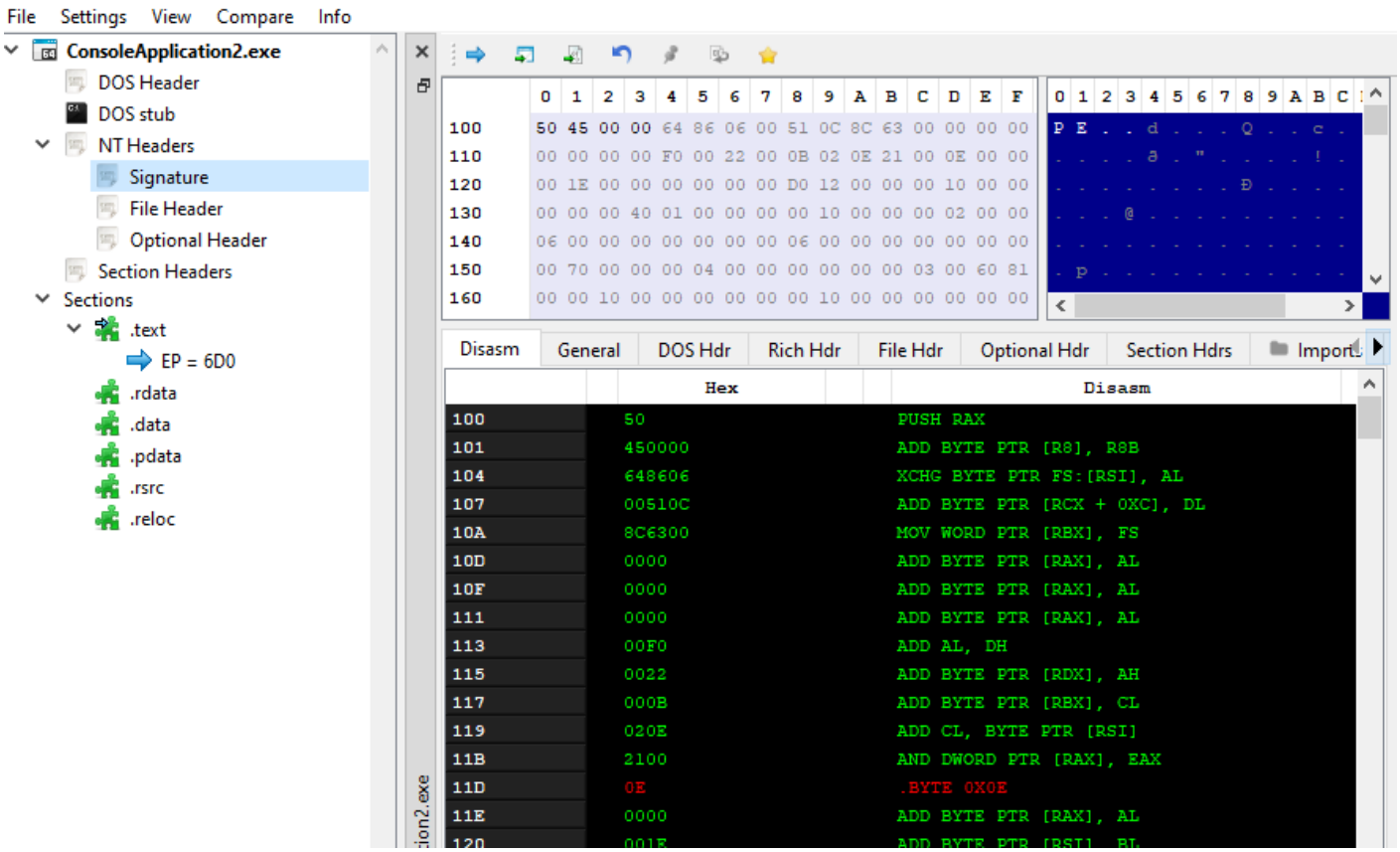

The main "CPU" tab has 4 screens:

1. Disassembly (Top-left): This window displays the assembly instructions being executed by the application.
2. Dump (Bottom-left): This window displays the memory contents of the application being debugged.
3. Registers (Top-right): This window displays the values of the CPU registers.
4. Stack (Bottom-right): This window displays the contents of the stack.

The remaining tabs also provide useful information but they will be discussed in the modules when they are used.

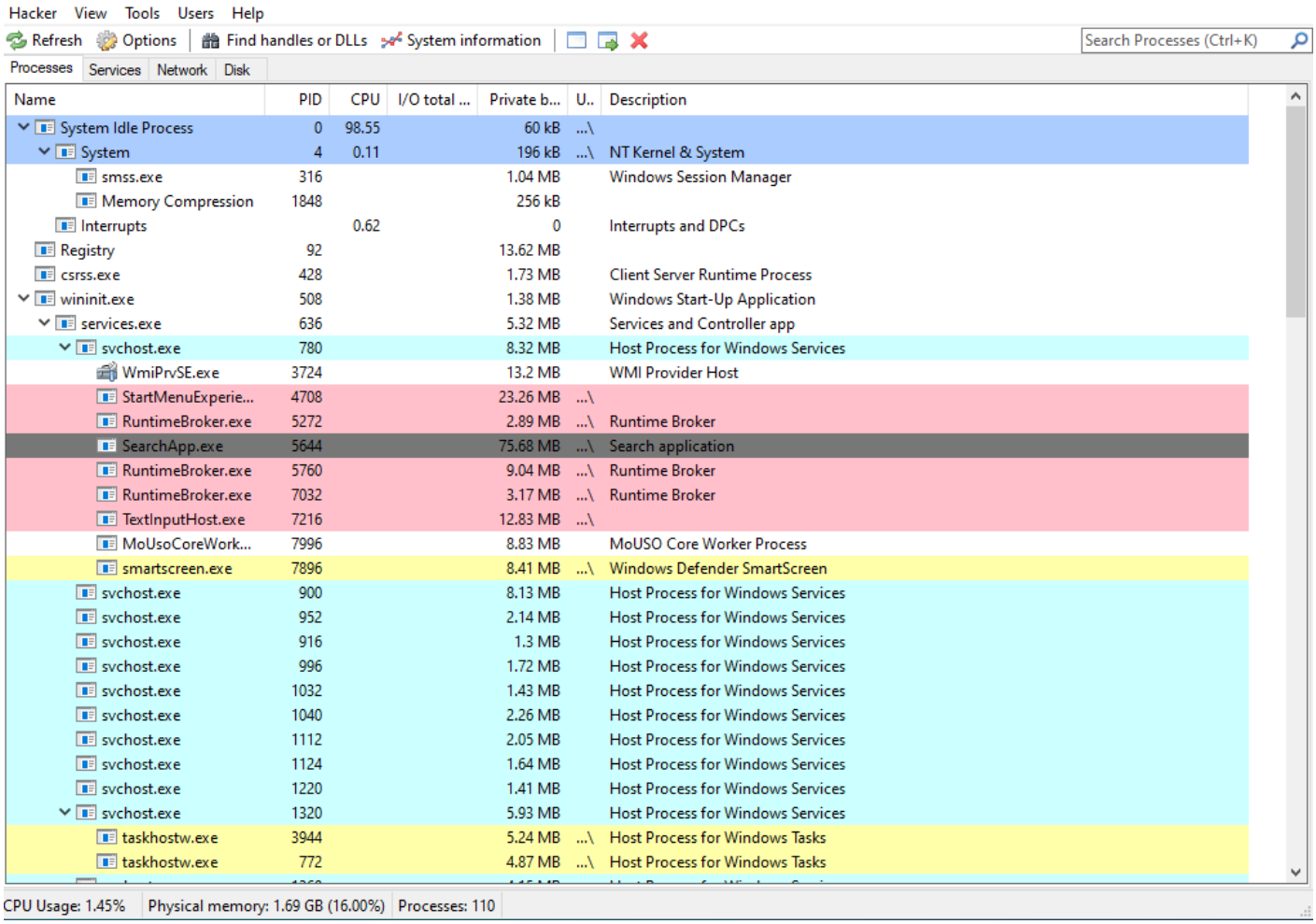
## PE-Bear

PE-Bear is a free, open-source tool designed to help malware analysts and reverse engineers quickly and easily analyze Windows Portable Executable (PE) files. It helps to analyze and visualize the structure of the PE file, view the imports and exports of each module, and perform static analysis to detect anomalies and possible malicious code. PE-bear also includes features such as PE header and section validation, as well as a hex editor.




## Process Hacker

Process Hacker is an open-source tool for viewing and manipulating processes and services on Windows. It is similar to Task Manager but provides more information and advanced features. It can be used to terminate processes and services, view detailed process information and statistics, set process priorities and more. Process Hacker will be useful when analyzing running processes to view items such as loaded DLLs and memory regions.




## Msfvenom

Msfvenom is a Metasploit framework standalone payload generator that allows users to generate various types of payloads. These payloads will be used by the malware created in this course.

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -h
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.exe

Options:
    -l, --list            <type>     List all modules for [type]. Types are: payloads, encoders, nops, platforms, archs, encrypt, f
ormats, all
    -p, --payload         <payload>  Payload to use (--list payloads to list, --list-options for arguments). Specify '-' or STDIN f
or custom
        --list-options               List --payload <value>'s standard, advanced and evasion options
    -f, --format          <format>   Output format (use --list formats to list)
    -e, --encoder         <encoder>  The encoder to use (use --list encoders to list)
        --service-name    <value>    The service name to use when generating a service binary
        --sec-name        <value>    The new section name to use when generating large Windows binaries. Default: random 4-characte
r alpha string
        --smallest                   Generate the smallest possible payload using all available encoders
        --encrypt         <value>    The type of encryption or encoding to apply to the shellcode (use --list encrypt to list)
        --encrypt-key     <value>    A key to be used for --encrypt
        --encrypt-iv      <value>    An initialization vector for --encrypt
    -a, --arch            <arch>     The architecture to use for --payload and --encoders (use --list archs to list)
        --platform        <platform> The platform for --payload (use --list platforms to list)
    -o, --out             <path>     Save the payload to a file
    -b, --bad-chars       <list>     Characters to avoid example: '\x00\xff'
    -n, --nopsled         <length>   Prepend a nopsled of [length] size on to the payload
        --pad-nops                   Use nopsled size specified by -n <length> as the total payload size, auto-prepending a nopsled
 of quantity (nops minus payload length)
    -s, --space           <length>   The maximum size of the resulting payload
        --encoder-space   <length>   The maximum size of the encoded payload (defaults to the -s value)
    -i, --iterations      <count>    The number of times to encode the payload
    -c, --add-code        <path>     Specify an additional win32 shellcode file to include
    -x, --template        <path>     Specify a custom executable file to use as a template
```

## Pre-Built Virtual Machines

There are pre-built virtual machines available for use. These VMs have all the required tools and course code pre-installed. Note that Microsoft Defender has a pre-configured exclusion for the `C:\Users\MALDEV01\Desktop\Maldev-code` folder.

- VMware - Download VM
    - SHA256: 01ac8ef7078e0c263af10123d54a065b0d74f6d6bf4ea41c0a31c90105a40ba6

- VirtualBox - Download VM
    - SHA256: 95774225f6e9265ac147e1a6af8a5ae488d833e6dd75debd13f3a38fa4f2de24

### 3D Accelerated Graphics

Ensure the virtual machine settings has 3D accelerated graphics disabled as that can result in the virtual machine to freeze or lag.

### Changelog

Some of the code has been updated after the VMs were deployed. To ensure you have the latest code samples, reinstall the code snippets from the modules below: