**Name: Mukul Rai**

**Student ID: 700748568**

**GitHub Link for Project:** [**https://github.com/raimukul/Malware\_Project**](https://github.com/raimukul/Malware_Project)

**Project 2**

You should parse two given applications in project1 for their all IMAGE\_DIRECTORY\_ENTRY\_IMPORT (including INT table, Name/Hint Table), all IMAGE\_DIRECTORY\_ENTRY\_BASERELOC,  and all Section headers. The codes should be written in C or C++.  Each of the structure could have more than one and you should parse all of them.

For example,  =you should print their values similar as below.

e\_magic: 5A4D

e\_cblp: 90

e\_cp: 3

e\_crlc：0

e\_cparhdr: 4

…

All values should be in hexadecimal.

**Code (Using C Programming Language)**

#include <stdio.h>

#include <windows.h>

int main(int argc, char \*argv[])

{

    if (argc != 2)

    {

        printf("Usage: %s <filename>\n", argv[0]);

        return 1;

    }

    HANDLE fileHandle = CreateFile(argv[1], GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

    if (fileHandle == INVALID\_HANDLE\_VALUE)

    {

        printf("Error opening file %s\n", argv[1]);

        return 1;

    }

    HANDLE mappingHandle = CreateFileMapping(fileHandle, NULL, PAGE\_READONLY, 0, 0, NULL);

    if (mappingHandle == NULL)

    {

        printf("Error creating file mapping\n");

        CloseHandle(fileHandle);

        return 1;

    }

    LPVOID mapView = MapViewOfFile(mappingHandle, FILE\_MAP\_READ, 0, 0, 0);

    if (mapView == NULL)

    {

        printf("Error creating file mapping view\n");

        CloseHandle(mappingHandle);

        CloseHandle(fileHandle);

        return 1;

    }

    PIMAGE\_DOS\_HEADER dosHeader = (PIMAGE\_DOS\_HEADER)mapView;

    if (dosHeader->e\_magic != IMAGE\_DOS\_SIGNATURE)

    {

        printf("Invalid DOS signature\n");

        UnmapViewOfFile(mapView);

        CloseHandle(mappingHandle);

        CloseHandle(fileHandle);

        return 1;

    }

    PIMAGE\_NT\_HEADERS ntHeaders = (PIMAGE\_NT\_HEADERS)((LPBYTE)dosHeader + dosHeader->e\_lfanew);

    if (ntHeaders->Signature != IMAGE\_NT\_SIGNATURE)

    {

        printf("Invalid NT signature\n");

        UnmapViewOfFile(mapView);

        CloseHandle(mappingHandle);

        CloseHandle(fileHandle);

        return 1;

    }

    PIMAGE\_DATA\_DIRECTORY importDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_IMPORT];

    printf("Import Directory (virtual address): 0x%08X, size: % 4X\n", importDirectory->VirtualAddress, importDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY exportDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_EXPORT];

    printf("Export Directory (virtual address): 0x%08X, size: % 4X\n", exportDirectory->VirtualAddress, exportDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY resourceDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_RESOURCE];

    printf("Resource Directory (virtual address): 0x%08X, size: % 4X\n", resourceDirectory->VirtualAddress, resourceDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY exceptionDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_EXCEPTION];

    printf("Exception Directory (virtual address): 0x%08X, size: % 4X\n", exceptionDirectory->VirtualAddress, exceptionDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY securityDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_SECURITY];

    printf("Security Directory (virtual address): 0x%08X, size: % 4X\n", securityDirectory->VirtualAddress, securityDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY baserelocDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_BASERELOC];

    printf("Base Relocation Table (virtual address): 0x%08X, size: % 4X\n", baserelocDirectory->VirtualAddress, baserelocDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY debugDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_DEBUG];

    printf("Debug Directory (virtual address): 0x%08X, size: % 4X\n", debugDirectory->VirtualAddress, debugDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY architectureDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_ARCHITECTURE];

    printf("Architecture Specific Data (virtual address): 0x%08X, size: % 4X\n", architectureDirectory->VirtualAddress, architectureDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY globalptrDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_GLOBALPTR];

    printf("RVA of GP (virtual address): 0x%08X, size: % 4X\n", globalptrDirectory->VirtualAddress, globalptrDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY tlsDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_TLS];

    printf("TLS Directory (virtual address): 0x%08X, size: % 4X\n", tlsDirectory->VirtualAddress, tlsDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY loadconFigDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_LOAD\_CONFIG];

    printf("Load Configuration Directory (virtual address): 0x%08X, size: % 4X\n", loadconFigDirectory->VirtualAddress, loadconFigDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY boundImportDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_BOUND\_IMPORT];

    printf("Bound Import Directory in headers (virtual address): 0x%08X, size: % 4X\n", boundImportDirectory->VirtualAddress, boundImportDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY iatDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_IAT];

    printf("Import Address Table (virtual address): 0x%08X, size: % 4X\n", iatDirectory->VirtualAddress, iatDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY delayImportDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_DELAY\_IMPORT];

    printf("Delay Load Import Descriptors (virtual address): 0x%08X, size: % 4X\n", delayImportDirectory->VirtualAddress, delayImportDirectory->Size);

    PIMAGE\_DATA\_DIRECTORY descriptorDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_COM\_DESCRIPTOR];

    printf("COM Runtime descriptor (virtual address): 0x%08X, size: % 4X\n", descriptorDirectory->VirtualAddress, descriptorDirectory->Size);

      PIMAGE\_DATA\_DIRECTORY relocDirectory = &ntHeaders->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_BASERELOC];

    printf("IMAGE\_DIRECTORY\_ENTRY\_BASERELOC (virtual address): 0x%08X, size: % 4X\n", relocDirectory->VirtualAddress, relocDirectory->Size);

    UnmapViewOfFile(mapView);

    CloseHandle(mappingHandle);

    CloseHandle(fileHandle);

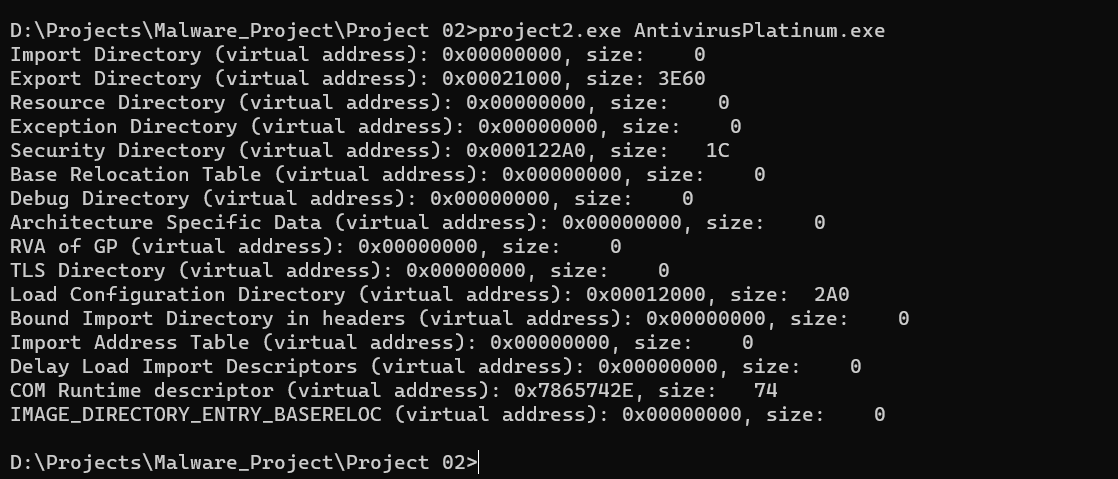
    return 0;

}

lllllll

**Output**

1. Output for AntivirusPlatinum.exe



1. Output For Stardust.exe

