Name: Archit Benipal

Subject: Software Supply Chain Security (CY 653)

Assignment 6: Windows Executable

Method 1: Using MinGW in Linux environment

What is MinGW

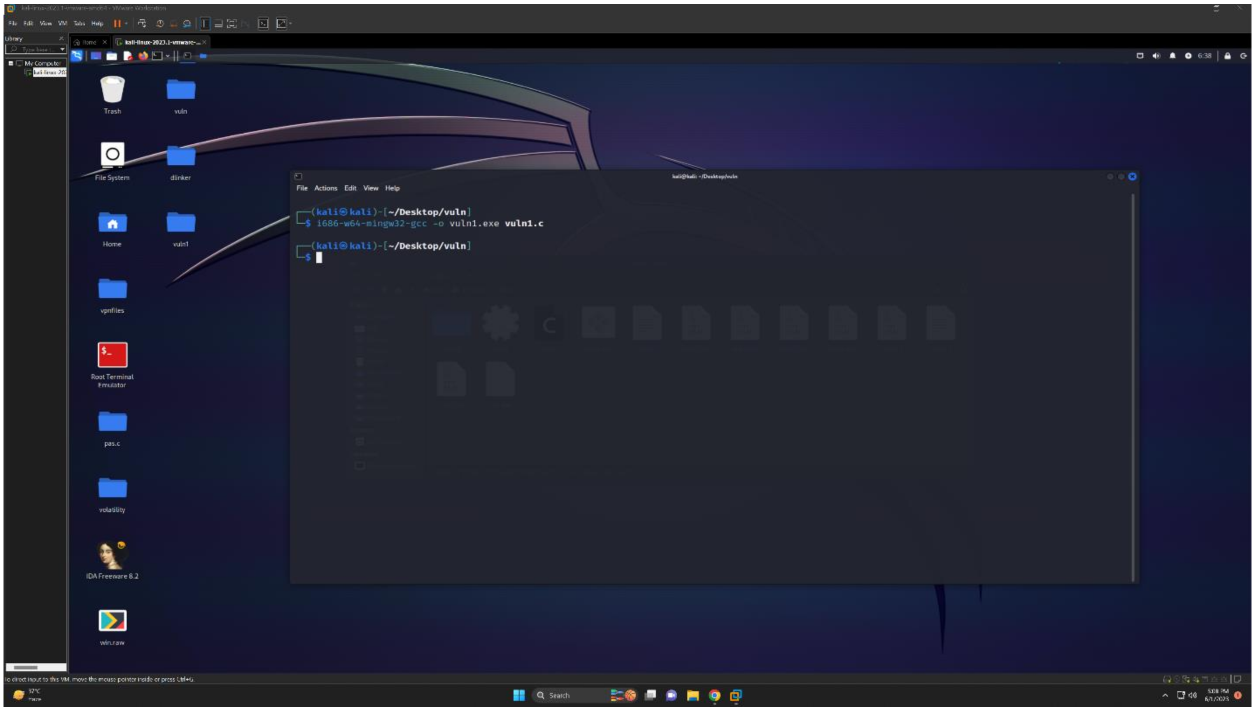
MinGW is a complete GCC toolchain (including half a dozen frontends, such as C, C++, Ada, Go, and whatnot) for the Windows platform which compiles for and links to the Windows OS component C Runtime Library in msvcrt.dll. MinGW provides a port of the GNU Compiler Collection (GCC), including C, C++, ADA, and Fortran compilers, which can generate native Windows executables. It also includes a set of freely available Windows specific header files and static import libraries that supplement the headers and libraries provided with any Win32 system. The key components of MinGW are:

- **GNU Compiler Collection (GCC)**: This is the compiler used to convert source code into executable programs. GCC supports several programming languages, including C, C++, and Fortran.
- **GNU Binutils**: This is a set of tools used to create and manage binary files, object code, and libraries. It includes programs like **ld** (the GNU linker) and **as** (the GNU assembler).
- **MSYS (Minimal SYStem)**: This provides a Unix-like command line interface to the MinGW system. It's a collection of GNU utilities such as **bash**, **make**, **gawk**, and others, which provide functionality similar to what you would find on a Linux system.
- **Windows API libraries**: These libraries provide interfaces to the Windows system. They include many of the standard Windows headers and libraries that your code may link to, like **kernel32**, **user32**, **gdi32**, and more.
- **GDB (GNU Debugger)**: This is the GNU project's debugger, allowing you to see what is going on 'inside' another program while it executes, or what another program was doing at the moment it crashed.

One of the significant advantages of MinGW is that it allows developers to create Windows applications using a similar workflow to what they might use in a Unix-like environment like Linux. The GCC compiler is a standard compiler on Unix-like systems, so many developers are already familiar with it. With MinGW, they can use that familiar environment to build Windows applications.

Breakdown of each steps with demo:

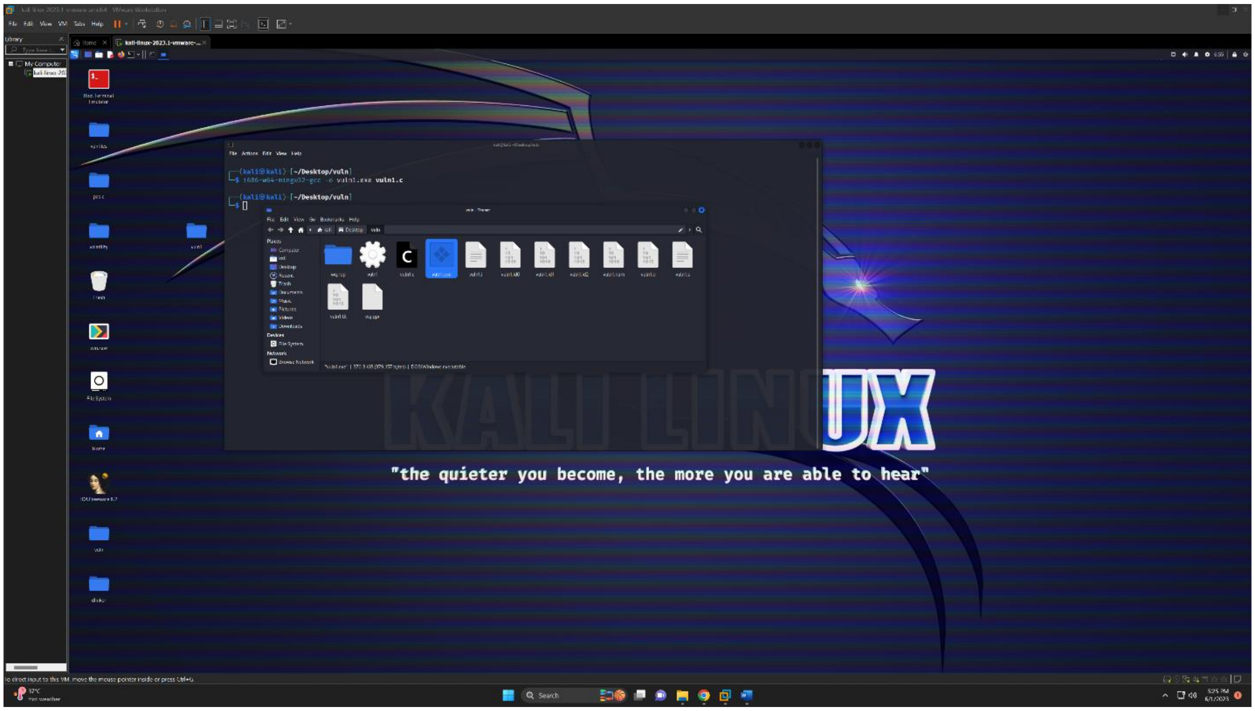
Step 1: Installation of MinGW in linux, command used “sudo apt-get install mingw-w64”



```
┌──(kali㉿kali)-[~/Desktop/vuln]
└─$ i686-w64-mingw32-gcc -o vuln1.exe vuln1.c

┌──(kali㉿kali)-[~/Desktop/vuln]
└─$ 
```

Step 2: Compilation of the file command used “i686-w64-mingw32-gcc -o vuln1.exe vuln1.c”



```
┌──(kali㉿kali)-[~/Desktop/vuln]
└─$ i686-w64-mingw32-gcc -o vuln1.exe vuln1.c

┌──(kali㉿kali)-[~/Desktop/vuln]
└─$ 
```

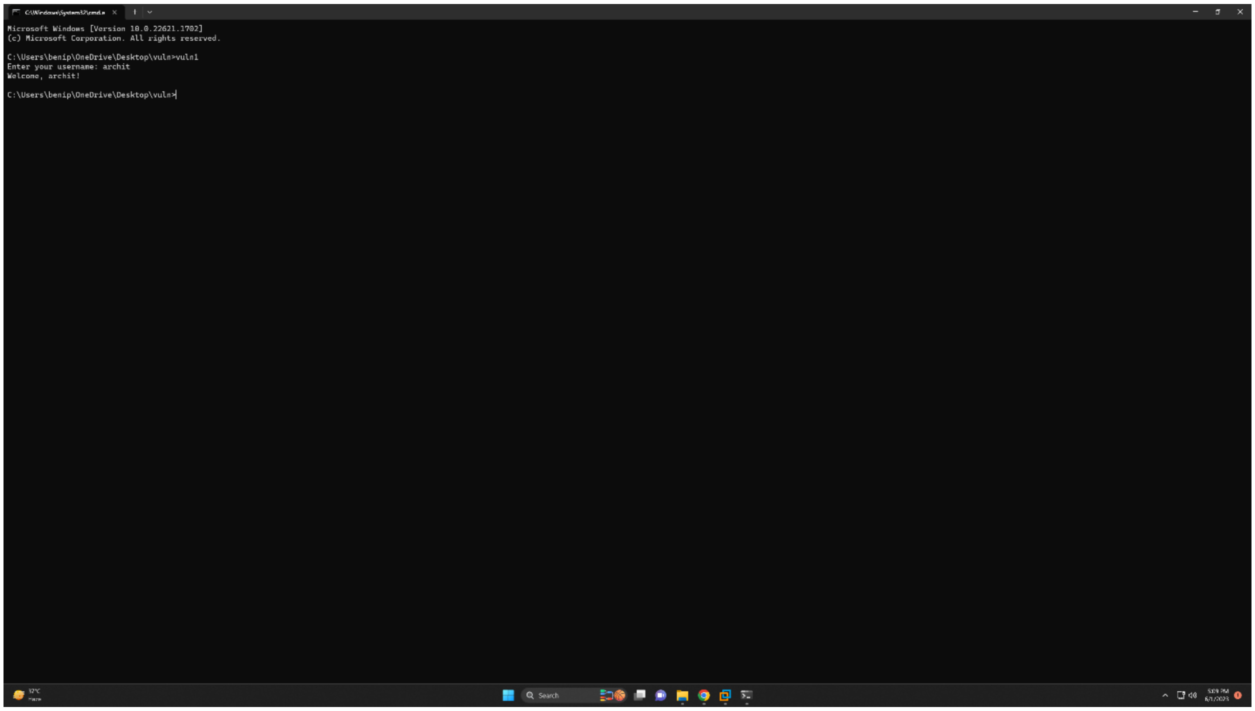
File Actions Edit View Help
File Edit View Go Bookmarks Help
kali
Desktop
vuln
Places
Computer
kali
Desktop
Recent
Trash
Documents
Music
Pictures
KALI LINUX
"the quieter you become, the more you are able to hear"

Here we got the vuln1.exe which we can shift to our windows machine for execution.

Detailed breakdown of the command used:

- **i686-w64-mingw32-gcc**: This is the name of the compiler being used. In this case, it's a version of the GCC compiler that's been set up to cross-compile for 32-bit Windows (i686 architecture) on a different type of system. The "mingw32" in the name indicates that it's using the MinGW software to create Windows-compatible executables.
    - **i686**: This refers to the target architecture, which is a 32-bit x86 CPU architecture.
    - **w64**: This indicates that the toolchain supports Windows 64-bit as well.
    - **mingw32**: This specifies that we are using the MinGW software, which allows us to compile code for Windows.
- **-o vuln1.exe**: This part of the command specifies the output file that the compiler should create. The **-o** option is followed by the desired name of the output file. In this case, the output file is **vuln1.exe**. If you didn't include the **-o vuln1.exe** part, the compiler would create an output file named **a.exe** by default.
- **vuln1.c**: This is the source file that you're compiling. It's a C source file, as indicated by the **.c** extension. The compiler will read this file, translate it into machine code, and create an executable file based on it.

Step 3: Execution in windows system



```
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\benip\OneDrive\Desktop\vuln>vuln1
Enter your username: archit
Welcome, archit!

C:\Users\benip\OneDrive\Desktop\vuln>
```

Here we have executed the .exe file with cmd in windows system

Method 2: Use Dev-C++ (beginner friendly)

What is Dev-C++

Dev-C++ is a free, open-source integrated development environment (IDE) that is used to write, compile, run, and debug C and C++ programs. It's widely used, particularly in educational settings, because of its simplicity and ease of use. Dev-C++ supports the use of MinGW GCC and Cygwin GCC compilers.

Breakdown of the steps:

Step 1: Once installation is complete, we can create a project

Step 2: Add the source code and compile, It contains a C++ compiler which also supports c, named MinGW.

Step 3: Run the vuln1.exe file.

Team Name: All safe

Team Members:

- **Archit Benipal**
- **Sivaprasad Kolli**
- **Venkata Nethaji Yenduri**