Lab 1: Lab Environment & Data Execution Prevention in Windows

This lab has three parts:

Part 1: The purpose of this lab part is to introduce you to set up the lab environment: virtual machine installation, network configuration, and C/C++ development setup.

Part 2: The purpose of this lab assignment is to introduce the Data Execution Presentation (DEP) in the Windows environment and conduct a simple buffer overflow attack with and without the enforcement of DEP.

Part3: The purpose of this lab assignment is to understand the buffer overflows and know how to exploit them in the Windows environment.

Students' Names:

_____

_____

Date:

_____

Part 1: Lab Environment Setup

1.1 Virtual Machine Installation

Please download the following tools at first:

- **VirtualBox**: it is a virtualization software package and it supports additional guest operating systems (e.g. Linux, Mac OS X and Windows) to be installed in your host operating system. It can be downloaded at http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html. Note that the latest versions of VirtualBox may have some compatible issues and the version of **VirtualBox-4.2.24** is suggested in this lab. VirtualBox-4.2.24 can be obtained from https://www.virtualbox.org/wiki/Download_Old_Builds_4_2.
  (As the old VirtualBox version could not be installed on new Mac OS, the latest versions for Mac OS is also fine.)
- **Windows 7 Virtual Box Image**: you can download the official Microsoft Windows 7 for VirtualBox from https://www.modern.ie/en-us/virtualization-tools#downloads. Note that **IE11 - Win7** is suggested.

Please refer to http://www.virtualbox.org/manual/ch02.html to install the VirtualBox in your host operating system. After the VirtualBox is installed, please follow the instructions at http://blog.reybango.com/2013/02/04/making-internet-explorer-testing-easier-with-new-ie-vms/ to install Windows 7 in the VirtualBox. At last, setup the virtual network and make the Windows 7 to be able to access Internet based on the instruction at https://www.virtualbox.org/manual/ch06.html.

Question 1.1: What networking mode you adopt for the Window 7 in the VirtualBox?

_____

1.2 Development Environment Setup

First, please logon to the Windows 7 in the VirtualBox and prepare to download C/C++ development IDE in it.

(If you have a preference on or are familiar with a specific C/C++ development IDE in Windows 7, you can skep the following steps)

In Windows 7, download Netbeans IDE for C/C++ at https://netbeans.org/downloads/. After that, please follow the instructions at https://netbeans.org/community/releases/72/cpp-setup-instructions.html to download all the required components of the Cygwin and setup the C/C++ compiler and debug environment. The Cygwin can be found at https://www.cygwin.com/.

Please create the first C++ application, named "lab1_1", using the following codes to be familiar with the Netbeans development environment:

```
#include <cstdlib>
```

```
#include <iostream>

using namespace std;

int main(int argc, char** argv) {

   std::cout << "Welcome to Developing Secure System!\n";
   if (std::cout.fail()) {
      std::cerr << "Sorry, greeting failed.\n";
      return EXIT_FAILURE;
   } else {
      return EXIT_SUCCESS;
   }
}
```

Question 1.2: What is the output of lab1_1?

_____

Part 2: Data Execution Prevention (DEP) in Windows

2.1 Introduction of DEP

Data Execution Prevention (DEP) is a security feature that can help prevent certain malicious exploits, especially attacks that store executable instructions in a data area via a buffer overflow. Please read the following three links to generally understand its mechanism.

- Wikipedia: http://en.wikipedia.org/wiki/Data_Execution_Prevention
- Microsoft: http://support.microsoft.com/kb/875352
- Microsoft: http://support.microsoft.com/kb/912923

Question 1.3: List the modes of the DEP enforcement?

_____

Open the VirtualBox and Start Windows 7 in your computer. Check the default DEP setting in the Windows 7. Please do not change the default setting of the DEP in your computer.

Question 1.4: What DEP mode(s) does your computer support?

_____

2.2 Enforcement of the DEP: A Simple Example of a Buffer Overflow Problem

First, you need to get a computer with Windows 7 64bit installed. It should not be a virtual machine and the hardware-enable DEP is active. Then, install Netbeans and Cygwin in that Windows 7. Open the Netbeans and create another C++ application, named lab1_2, using the following codes:

```
#include <stdio.h>
#include <string.h>

using namespace std;

int main(int argc, char** argv) {

    int value = 5;
    char buffer_one[8], buffer_two[8];
    strcpy(buffer_one, "one"); /* Put "one" into buffer_one. */
    strcpy(buffer_two, "two"); /* Put "two" into buffer_two. */
    printf("[BEFORE] buffer_two is at %p and contains \'%s\'\n", buffer_two, buffer_two);
    printf("[BEFORE] buffer_one is at %p and contains \'%s\'\n", buffer_one, buffer_one);
    printf("[BEFORE] value is at %p and is %d (0x%08x)\n", &value, value, value);


    /*Start a buffer overflow instance*/
```

```
    printf("\n[STRCPY] copying %d bytes into buffer_two\n\n", strlen(argv[1]));
    strcpy(buffer_two, argv[1]); /* Copy first argument into buffer_two. */
    printf("[AFTER] buffer_two is at %p and contains \'%s\'\n", buffer_two, buffer_two);
    printf("[AFTER] buffer_one is at %p and contains \'%s\'\n", buffer_one, buffer_one);
    printf("[AFTER] value is at %p and is %d (0x%08x)\n", &value, value, value);
}
```

Run lab1_2 using the argument ***1234567890*** (e.g. lab1_2.exe 1234567890).

Question 1.5: What are the outputs of lab1_2? Is the buffer overflow prevented by the DEP?

Now, you need to turn off the DEP in the Windows 7.

Open the VirtualBox and Start Windows 7. After you logon the system, open the Command Prompt in Windows 7 and use "***bcdedit /set {current} nx AlwaysOff***" to turn off the DEP.

Restart the Windows 7. Run the Command prompt again and type "***wmic OS Get DataExecutionPrevention_Drivers***" to check if all the DEP modes have been turned off.

Then, open the Netbeans installed in the Window 7 and create another C++ application, also named lab1_2, using the same codes used above.

Run lab1_2 using the same argument ***1234567890***

Question 1.6: What are the outputs of lab1_2 now?

Question 1.7: Is the buffer overflow exploited? If it is, briefly explain what happened and the outputs?

Part 3: Buffer Overflow Attacks

3.2 Heap-based Buffer Overflow

In the Windows 7 installed in the Virtual Box, first make sure that DEP is turned off. Then, open the Netbeans and create another C++ application, named lab1_3, using the following codes:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

#define BUFSIZE 16
#define OVERSIZE 8


using namespace std;

typedef unsigned long u_long;
typedef unsigned int u_int;

/*
 *
 */
int main(int argc, char** argv) {

    u_long diff;
    char *buf1 = (char *) malloc(BUFSIZE), *buf2 = (char *) malloc(BUFSIZE);

    diff = (u_long) buf2 - (u_long) buf1;
    printf("buf1 = %p, buf2 = %p, diff = 0x%x bytes\n", buf1, buf2, diff);

    memset(buf2, 'A', BUFSIZE - 1), buf2[BUFSIZE - 1] = '\0';

    printf("before overflow: buf2 = %s\n", buf2);
    memset(buf1, 'B', (u_int) (diff + OVERSIZE));
    printf("after overflow: buf2 = %s\n", buf2);

    return 0;
}
```

Question 1.8: What are the outputs? Where is the buffer overflow in the above codes? Briefly explain it.

_____

_____

_____

_____

_____

3.1 Stack-Based Buffer Overflow Attack

In the Windows 7 installed in the Virtual Box, first make sure that DEP is turned off. Then, open the Netbeans and create another C++ application, named lab1_4, using the following codes:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

using namespace std;

int check_authentication(char *password) {
   int auth_flag = 0;
   char password_buffer[16];
   strcpy(password_buffer, password);
   if (strcmp(password_buffer, "brillig") == 0)
      auth_flag = 1;
   if (strcmp(password_buffer, "outgrabe") == 0)
      auth_flag = 1;
   return auth_flag;
}

/*
 *
 */
int main(int argc, char** argv) {

   if (argc < 2) {
      printf("Usage: %s <password>\n", argv[0]);
      exit(0);
   }
   if (check_authentication(argv[1])) {
      printf("\n-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n");
      printf(" Access Granted.\n");
      printf("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n");
   } else {
      printf("\nAccess Denied.\n");
   }
   return 0;
}
```

Question 1.9: Where is the buffer overflow vulnerability in the above codes?

_____

_____

Question 1.10: What are the required passwords to pass the authentication? Can you try to exploit the buffer overflow in the above codes to pass the authentication without entering the required passwords (HINTS: Enter an enough long string for the argument)? If you can, what is your input (argument) and what is the corresponding output? Also, briefly explain how you exploited the vulnerability and why you can do it?

Question 1.11: Briefly explain the differences between the stack-based buffer overflow and the heap-based buffer overflow?