ETHICAL HACKING V2
LAB SERIES

Lab 13: Understanding Buffer Overflows

Document Version: 2020-08-24

Material in this Lab Aligns to the Following	
Books/Certifications	**Chapters/Modules/Objectives**
All-In-One CEH Chapters ISBN-13: 978-1260454550	6: Web-Based Hacking: Servers and Applications
EC-Council CEH v10 Domain Modules	13: Hacking Webservers 14: Hacking Web Applications 15: SQL Injection
CompTIA Pentest+ Objectives	4.4: Given a scenario, analyze a basic script (limited to Bash, Python, Ruby, and PowerShell)
CompTIA All-In-One PenTest+ Chapters ISBN-13: 978-1260135947	6: Social Engineering

Contents

Introduction

Buffer overflows are programming errors, whether intentional or accidental. This lab shows how to create a vulnerable program and demonstrate the vulnerability.

Objective

In this lab, you will be conducting ethical hacking practices using various tools. You will be performing the following tasks:

1. Writing a Buffer Overflow Program
2. Run Code to Demonstrate Buffer Overflow
3. Analyzing and Modifying Overflow Code

Pod Topology

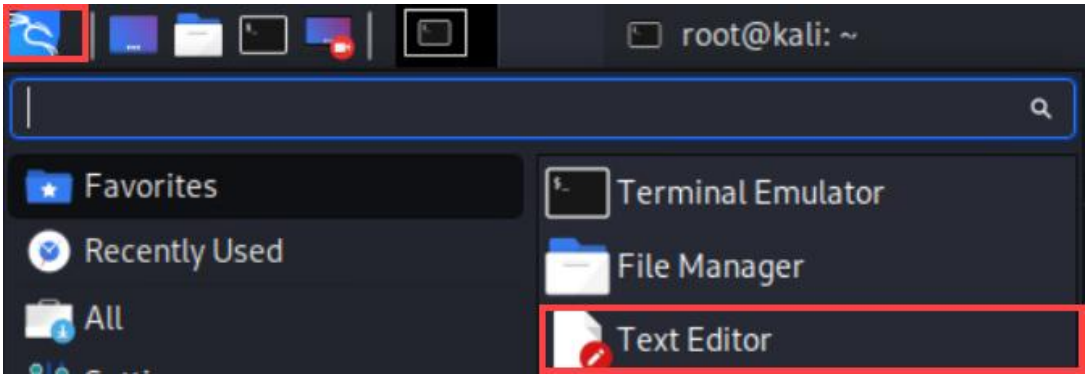
Lab Settings

The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

Virtual Machine	**IP Address**	**Account** (if needed)	**Password** (if needed)
Kali Linux	192.168.9.2 192.168.0.2	root	toor

1 Writing a Buffer Overflow Program

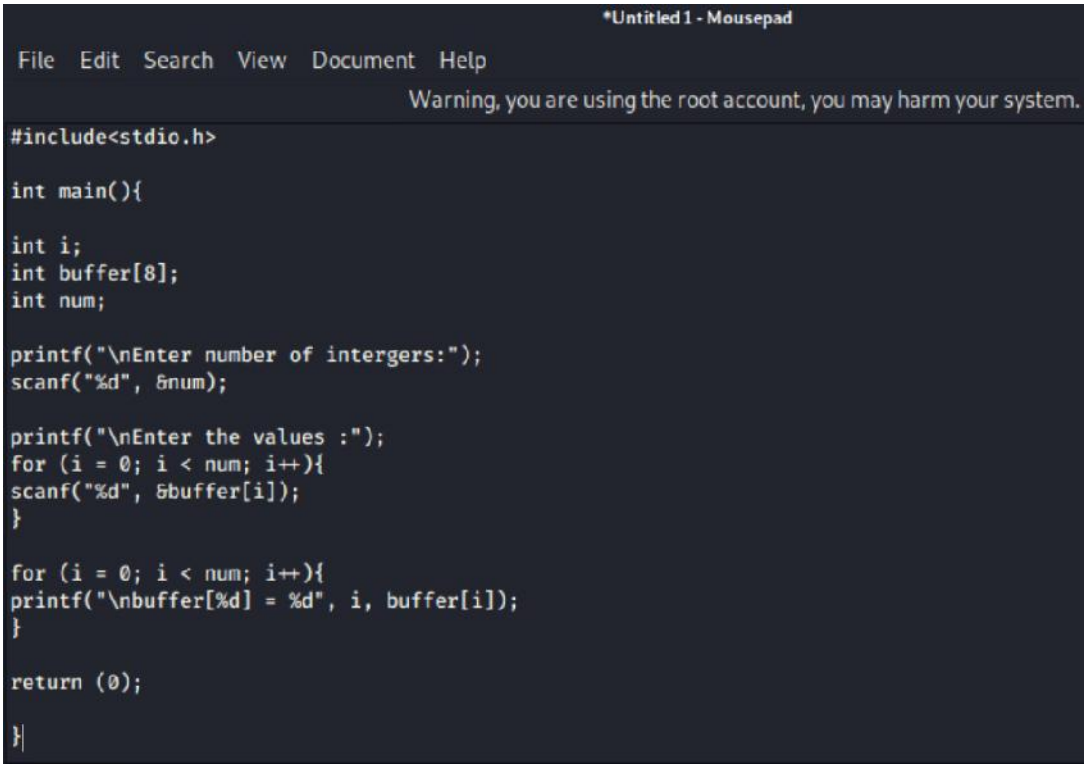
1. Click on the **Kali** tab.
2. Click within the console window and press **Enter** to display the login prompt.
3. Enter **`root`** as the *username*. Press **Tab**.
4. Enter **`toor`** as the *password*. Click **Log In**.
5. Launch the *Mousepad* editor by selecting the **Application Menu > Text Editor**.



6. Using the *Mousepad* editor, type the script below. The text shown in the red font are comments. These comments only explain what each respective line processes and do not need to be included in the script.

```
#include<stdio.h>

int main(){

int i;              //integer value for variable "i"
int buffer[8];      //an array of integers called "buffer" of up to 8 chars
int num;            //integer value for variable "num"

printf("\nEnter number of integers:");  //get user's input on the # of ints
scanf("%d", &num);                      //place values into the "num" variable

//Read into array section
printf("\nEnter the values :");  //get the integers from the user
for (i = 0; i < num; i++){       //loop to read the integers into the "buffer"
scanf("%d", &buffer[i]);
}

//Print the array of integers
for (i = 0; i < num; i++){              //loop thru buffer and print integers
printf("\nbuffer[%d] = %d", i, buffer[i]);
}

return (0);         //end the program

}
```


Text should read as shown below:



7. Click the **File** menu option from the top pane and select **Save As**.
8. Type **`vuln1.c`** into the *Name* text field.
9. Click **Save**.
10. Close the **Mousepad** application.

2 Run Code to Demonstrate Buffer Overflow

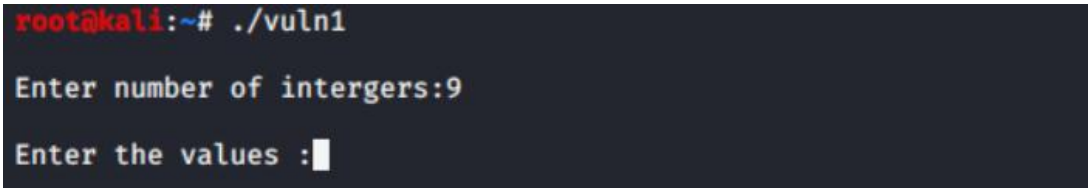
1. Open the *Terminal* by clicking on the **Terminal** icon located at the top of the page if not already open.
2. Compile the *C* code by typing the command below into the *Terminal,* followed by pressing the **Enter** key.

```
gcc vuln1.c –o vuln1
```

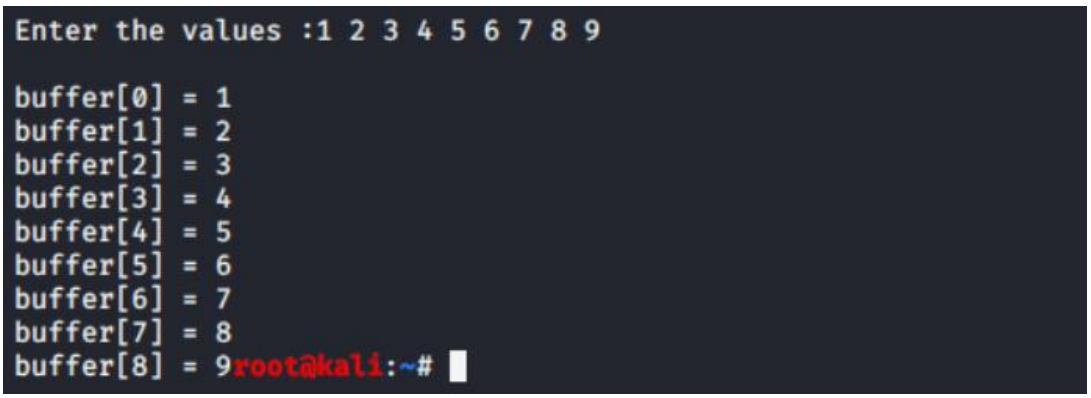
3. Run the program by entering the command below.

```
./vuln1
```

4. When prompted for the number of integers, type **9** even though the buffer only allocates 8. Press **Enter**.



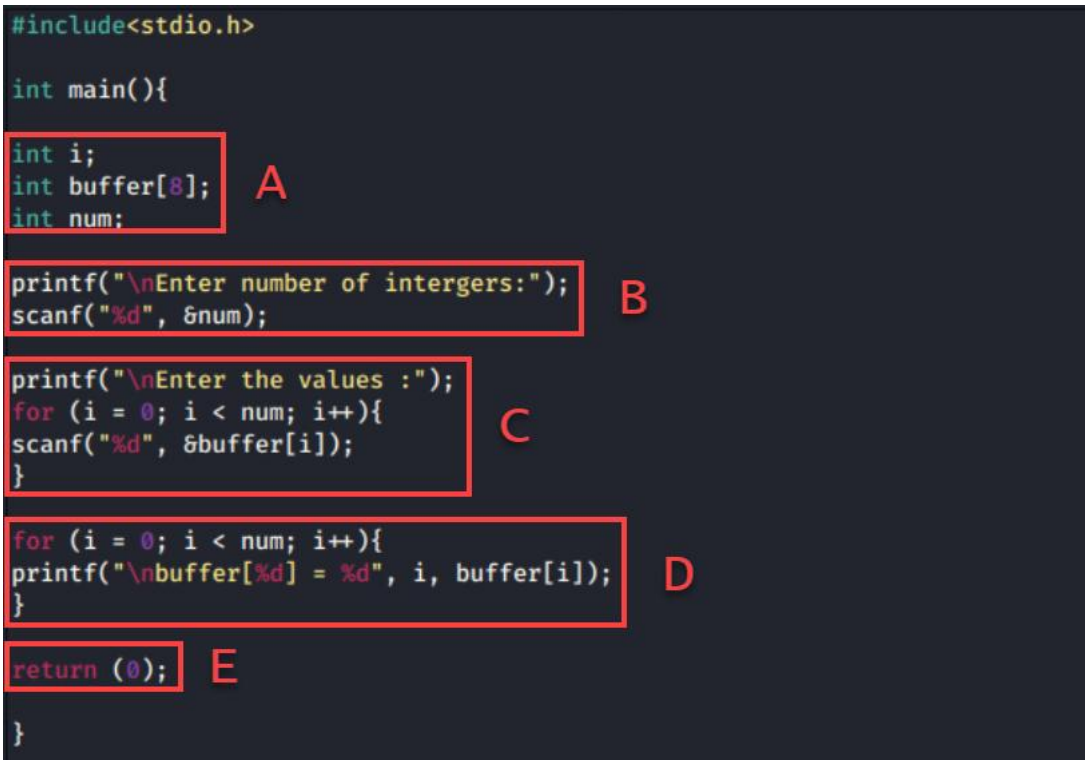
5. When prompted to enter the values, type **1 2 3 4 5 6 7 8 9**. Press **Enter**.



Notice a buffer overrun was caused because 9 elements were chosen as an input into a buffer of 8.

3 Analyzing and Modifying Overflow Code

1. Open the **Mousepad** application once more from **Application > Text Editor.**
2. Click **File** and choose **Open**.
3. Select **vuln1.c** from the middle pane and click **Open**.
4. Observe the contents of the vuln1.c program:



- ***Section A***: This sets up the variables called *i*, *num*, and an array or list of characters *buffer* of up to 8 in length.

- ***Section B***: This section asks the user how many integers to put into the array. Note that the *scanf* function is the problem.

- ***Section C***: Asks the user for the integers to use and then puts them one by one into the array *buffer*.

- ***Section D*:** Prints out the integers that the user inputted previously.

- ***Section E*:** Ends the program.

5. Alter the code to prevent an overflow. Add the following lines of code to *vuln1.c,* so it reads as shown below.

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

int main(){

int i;              //integer value for variable "i"
int buffer[8];      //an array of integers called "buffer" of up to 8 chars
int num;            //integer value for variable "num"

printf("\nEnter number of integers:");  //get user's input on the # of ints
scanf("%d", &num);                      //place values into the "num" variable

if (num > 8){              //check the input is less than 8 numbers
printf("You entered a value greater than allowed!\n");
exit(0);                   //exit the program
}

//Read into array section
printf("\nEnter the values :");  //get the integers from the user
for (i = 0; i < num; i++){       //loop to read the integers into the "buffer"
scanf("%d", &buffer[i]);
}

//Print the array of integers
for (i = 0; i < num; i++){              //loop thru buffer and print integers
printf("\nbuffer[%d] = %d", i, buffer[i]);
}

return (0);         //end the program

}
```


The text should read as below:

6. Click **File** and select **Save As**.
7. Type **`fixed_vuln1.c`** into the *Name* text field.
8. Click **Save**.
9. Close the **Mousepad** application.
10. Change focus to the **Terminal** window and enter the command below to compile the code for the new *fixed_vuln1.c* program.

```
gcc fixed_vuln1.c –o fixed_vuln1
```

11. Run the new code.

```
./fixed_vuln1.c
```

12. When prompted for the number of integers, type **9**. Press **Enter**.

Notice an output is given signaling that a value greater than 8 was entered and is not allowed resulting in the program to exit.

13. You may now end your reservation.

