Lab 6 - Overflow Vulnerabilities

EECS 4481 4.0 Computer Security Lab, Winter 2015

Due: March 8th, 2015, 11:59pm.

Format: Individual

Learning Objective: To gain first-hand experience on different types of overflow vulnerabilities. Such vulnerabilities can be utilized by a malicious user to alter the flow control of the program, even execute arbitrary pieces of code. For instance, buffer overflow vulnerability arises due to the mixing of the storage for data (e.g. buffers) and the storage for controls (e.g. return addresses): an overflow in the data part can affect the flow of the program, because an overflow can change the return address and as a result enable execution of malicious code.

In this lab, you will be given various programs. Your task is to check whether the programs have vulnerabilities and fix them. In addition, you will attack Linux and Windows OSs using the metasploit framework.

Task 1

The Metasploit framework is an efficient tool for exploiting known security bugs. For this task, you must use it to try to exploit overflows of the Linux or Windows OSs in the Attack Lab, as well as any web browsers in these systems.

Report: Describe the bugs you found, the steps you followed for your attacks.

Task 2

Find security vulnerabilities in the following programs.

Report: For every buffer overflow bug you find, demonstrate evidence of altering the program counter register.

Program A

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

int bof(char *str)
{
    char buffer[12];
    strcpy(buffer, str);
    return 1;
}
int main(int argc, char **argv)
{
    char str[517];
    FILE *file;
    file = fopen(argv[1], "r");
    fread(str, sizeof(char), 517, file);
    bof(str);
    printf("Done!\n");
    return 1;
}
```

Program B

```
#include <stdio.h>
void read_input()
{
    char input[512];
    int c = 0;
    while (read(0, input + c++,1) == 1);
}
int main ()
{
   read_input();
   printf("Done !\n");
   return 0;
}
```

Program C

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 50

 int main (int argc, char **argv) {
      char buf[MAX_SIZE], *pbuf = buf;
      int count = atoi(argv[1]);
      if (count > MAX_SIZE) count = MAX_SIZE -1;
      while (count --) *pbuf++ = getchar();
      printf("Done !");
      return 0;
 }
```

Program D

```
/* based on gera@core-sdi.com's code */
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 50

unsigned int convertToNum(char * str) {
     unsigned int res = 0;
     for (; *str&&isdigit(*str); res*=10, res+= *str++ - '0');
     return res;
}


int main (int argc, char **argv) {
     char buf[MAX_SIZE], *pbuf = buf;
     int count = convertToNum(argv[1]);
     if (count > MAX_SIZE) count = MAX_SIZE -1;
     while (count --) *pbuf++ = getchar();
     printf("Done !");
     return 0;
 }
```