НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

«Безпечне програмування»

*Звiт з лабораторної роботи №2*

*Тема: «Робота зі строками. Руйнування стеку»*

Виконав:

ст. гр. 16Б

Башкатов В.Є.

Харків – 2019

***Мета:*** Дослідити руйнування стеку використовуючи відлагодник, показати призначення і принцип роботи використання BufferSecurityCheck аргументів компіляторів.

***Завдання***

Використовуючи компілятори vc від MS Visual Studio та GCC, створити програму, що демонструє наявність проблеми, де переповнення змінної-масиву на рівні стека в функції може привести до руйнування стеку - при спробі доступа до недійного адресу в пам'яті. Довести, що дана недійсна адреса в пам'яті пов'язана з даними, що записуються до масиву.

Використовуючи відлагодник, показати призначення і принцип роблоти (на рівні асемблерних команд) використання BufferSecurityCheck аргументів компіляторів (/GS для MSVS, та -fno-stack-protector для GCC).

Рекомендується використовувати відлагодник від IDA

**Текст програми**

#include <stdio.h>

void show\_array(int arrlen, char array[])

{

char buffer[32];

int i;

for (i = 0; i < arrlen; i++) buffer[i] = array[i];

printf(buffer);

}

int main()

{

char mystr[] = "1223123234324234234234234234242342342342342342342343";

show\_array(101, mystr);

return 0;

}

1. На рис.1 використовуючи компілятор Visual Studio 12 демонструється наявність проблеми, де переповнення змінної-масиву на рівні стека в функції може привести до руйнування стеку – при спробі доступа до недійного адресу в пам'яті.

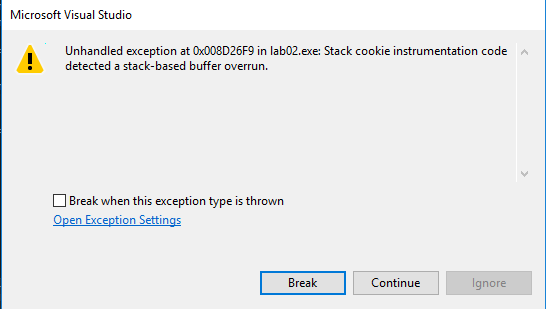
****

Рисунок 1 – переповнення змінної-масиву на рівні стека в функції(руйнування стеку)

1. На рис.2 використовуючи компілятор GCC демонструється наявність проблеми, де переповнення змінної-масиву на рівні стека в функції може привести до руйнування стеку – при спробі доступа до недійного адресу в пам'яті.

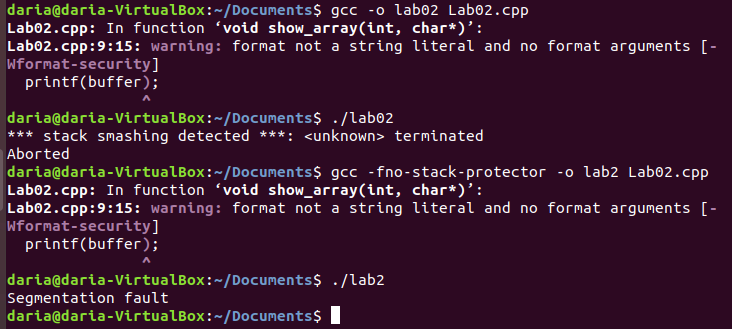


Рисунок 2 – переповнення змінної-масиву на рівні стека в функції(руйнування стеку)

1. На рис.3-4 використовуючи відлагодник показано призначення використання BufferSecurityCheck аргументів компіляторів (/GS для MSVS, та -fno-stack-protector для GCC).

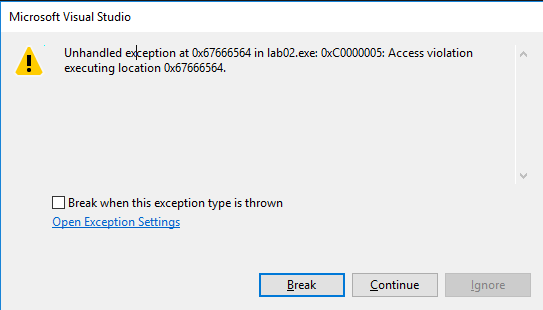


Рисунок 3 – з вимкненим BufferSecurityCheck для *MSVS*



Рисунок 4 – з вимкненим BufferSecurityCheck для *GCC*

**Висновки:**

В результаті виконання лабораторної роботи було з’ясовано, що основна причина вразливості – використання деяких функцій мови С, які працюють з рядками і не перевіряючих розміри своїх аргументів (наприклад strcpy, strcat, gets або sprintf). Тому переважна більшість (якщо не всі) уразливих програм написані на С.