Lab 7 Exercise 2 - Protecting the Return Address

Name: NOEL LIM XIAN

Student Number: A0211270Y

Email: E0493357@u.nus.edu

Date: 31 OCTOBER 2022

**Exercise 2**

**// EXERCISE 2**

**// Only f is modified from lab7-ex1.c**

**// Two compiled binaries are compiled for Exercise 2 as below. Both are executed with arg `-p` to flip my\_protect bit to 1 and activate the detection mechanism.**

**// 1. using `$ gcc -fno-stack-protector lab7-ex2.c`**

**// 2. using `$ gcc -fno-stack-protector -O2 lab7-ex2.c`**

**int f()**

**{**

**// Define canary size as 8 [characters].**

**#define CANARY\_SIZE 8**

**/\***

**1. Open "/dev/urandom" in read mode and assign the new FILE pointer from fopen() to fp.**

**This should be the first memory allocation instruction in the stack to prevent interjection of the order of variables in this exercise.**

**We get data "/dev/urandom" for the canary value so that we have random values in each execution (harder for attacker to predict runtime, arbitrary value)**

**\*/**

**FILE \*fp = fopen("/dev/urandom", "r");**

**/\***

**2. The variables in the stack should be in contiguous address in order of declaration:**

**+--------+**

**| | canary\_value (8 bytes)**

**+--------+**

**| | buf (8 bytes)**

**+--------+**

**| | canary (8 bytes)**

**+--------+**

**. . c (1 byte)**

**+--------+**

**canary\_value - random value as original "cookie".**

**buf - buf to get data from stdin.**

**canary - canary that is immediate after buf in the addressing space.**

**\*/**

**char canary\_value[CANARY\_SIZE];**

**char buf[8];**

**char canary[CANARY\_SIZE];**

**char c;**

**/\* 3. my\_protect reads 8 bytes from /dev/urandom into canary\_value. canary\_value is then copied into canary.**

**\*/**

**if (**

**my\_protect == 1)**

**{**

**int p = fread(&canary\_value, 1, CANARY\_SIZE, fp);**

**if (p != CANARY\_SIZE)**

**{**

**fprintf(stderr, "size read from /dev/urandom deviates from CANARY\_SIZE \n");**

**exit(1);**

**}**

**while (--p >= 0)**

**canary[p] = canary\_value[p];**

**}**

**fclose(fp);**

**// LINE1 - don't change code from LINE1 to LINE2**

**printf("Enter a string: ");**

**for (i = 0; (c = getchar()) != '\n'; i++)**

**buf[i] = c;**

**buf[i] = '\0';**

**printf("string = [%s]\n", buf);**

**// LINE2**

**/\* 4. If there is buffer overflow from the array buf, canary will change and differ from canary\_value (the original value).**

**We will compare each character of canary\_value and canary.**

**Upon detection of change, cookie\_error is raised.**

**\*/**

**if (my\_protect == 1)**

**{**

**for (int p = 0; p < CANARY\_SIZE; p++)**

**{**

**if (canary[p] != canary\_value[p])**

**{**

**cookie\_error();**

**}**

**}**

**}**

**return 0;**

**}**

**END**