Programowanie strukturalne - wykład 5

ASLR

ASLR (Address Space Layout Randomization) tłumaczony jest jako mechanizm losowego generowania lokalizacji alokacji pamięci wirtualnej.

Wykonywanie czynności przedstawionych na dalszych slajdach związanych z ASLR może oznaczać narażenie komputera na niebezpieczeństwo i podatność na ataki. Wykonanie tych działań nie jest zalecane i robione tylko na własną odpowiedzialność.

Exploit Protection

Funkcja Exploit Protection jest wbudowana w system Windows 10 w celu zabezpieczenia urządzenia przed atakami. Twoje urządzenie jest od razu skonfigurowane za pomocą ustawień ochrony, które są odpowiednie dla większości użytkowników.

Ustawienia funkcji Exploit Protection

Dowiedz się więcej

Wymuś losowe generowanie obrazów (obowiązkowa funkcja ASLR)
Wymuś relokację obrazów, które nie zostały skompilowane z użyciem przełącznika /DYNAMICBASE

Włączone domyślnie

Generuj losowo alokacje pamięci (funkcja ASLR „od dołu do góry”)
Generuj losowo lokalizacje alokacji pamięci wirtualnej.

Włączone domyślnie

Funkcja ASLR o wysokiej entropii
Zwiększ zmienność podczas używania ustawienia Generuj losowo alokacje pamięci (funkcja ASLR „od dołu do góry”).

Włączone domyślnie

Funkcja `malloc`

```
void *malloc(size_t size);
```

Funkcja służy do dynamicznego rezerwowania miejsca w pamięci. Gdy funkcja zostanie wywołana, w przypadku sukcesu zwróci wskaźnik do nowo zarezerwowanego miejsca w pamięci; w przypadku błędu zwraca wartość NULL.

Funkcja `free`

```
void free(void *ptr);
```

Funkcja free zwalnia blok pamięci wskazywany przez ptr wcześniej przydzielony przez `malloc`. Jeżeli ptr ma wartość NULL funkcja nie robi nic.

Wskaźniki na funkcję

```
typ_zwracany (*nazwa_wsk)(typ1 arg1, typ2 arg2);
```

```c
#include <stdio.h>

int suma (int lhs, int rhs)
{
  return lhs+rhs;
}

int main ()
{
  int (*wsk_suma)(int a, int b);
  wsk_suma = suma;
  printf("4+5=%d\n", wsk_suma(4,5));
  return 0;
}
```

Jaka różnica?

```
int * wsk1();
int (*wsk2)();
int *(*wsk3)();
```

1. Funkcja zwracająca wskaźnik.
2. Wskaźnik na funkcję.
3. Wskaźnik na funkcję, zwracającą wskaźnik.

Inne typy liczbowe?

Pełna analogia.

Duży błąd merytoryczny - 2 na egzaminie(!)

Dereferencja niezainicjalizowanych wskaźników:

```c
#include <stdio.h>
int main(void)
{
    int *wsk; // niezainicjalizowany wskaznik
    *wsk = 5;
    return 0;
}
```

Tablice jednowymiarowe

Definicja

Tablica - ciąg elementów jednego typu.

Deklaracja:

```
int tab[4];
char tab2[3];
float tabf[5];
```

Inicjalizacja

```
int tab[4] = {1,-3,4,5};
```

Stałe tablice

```
const int tab[4] = {1,-3,4,5};
```

Tworzenie przez stałą DEFINE

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 3

int main()
{
    int tab[ROZMIAR] = {3,-4,3};
    return 0;
}
```

Brak inicjalizacji?

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 3

int main()
{
    int tab[ROZMIAR];
    for(int i=0;i<ROZMIAR;i++)
    {
        printf("%d\n",tab[i]);
    }
    return 0;
}
```

Za mało wartości?

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 3

int main()
{
    int tab[ROZMIAR] ={2,3};
    for(int i=0;i<ROZMIAR;i++)
    {
        printf("%d\n",tab[i]);
    }
    return 0;
}
```

Za dużo wartości?

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 2

int main()
{
    int tab[ROZMIAR] ={2,3,-2,0};
    for(int i=0;i<ROZMIAR;i++)
    {
        printf("%d\n",tab[i]);
    }
    return 0;
}
```

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

int main()
{
    int tab[] ={2,3,-2,0};
    for(int i=0;i<sizeof tab/sizeof(int);i++)
    {
        printf("%d\n",tab[i]);
    }
    return 0;
}
```

Oznaczona inicjalizacja - od C99

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 5

int main()
{
    int tab[ROZMIAR] = {[2] = 34};
    for(int i=0;i<ROZMIAR;i++)
    {
        printf("%d\n",tab[i]);
    }
    return 0;
}
```

Przypisanie wartości

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 5

int main()
{
    int tab[ROZMIAR] = {[2] = 34};
    tab[0]=2;
    for(int i=0;i<ROZMIAR;i++)
    {
        printf("%d\n",tab[i]);
    }
    return 0;
}
```

Zakres tablic

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 5

int main()
{
    int tab[ROZMIAR] = {1,-2,3};
    printf("%d",tab[-1]);
    return 0;
}
```

Jak określić rozmiar inaczej niż przez stałą DEFINE?

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

int main()
{
    int n = 5;
    int m = 8;
    int tabb1[5]; // ok
    int tabb2[3*2+1]; // ok
    int tabb3[sizeof(int)+1]; // ok
    //int tabb4[-2]; // nie,  bo <0
    int tabb5[0]; // ???
    //int tabb6[6.5]; // nie, bo nie calk.
    //int tabb7[(int)2.5]; // ???
    int tabb8[n]; // brak dla C90
    int tabb9[m]; // Brak dla C90
    return 0;
}
```

Bibliografia

- Richard Reese, Wskaźniki w języku C, Wydawnictwo Helion 2014.
- https://pl.wikibooks.org/wiki/C/Wska%C5%BAniki, dostęp online 15.03.2020.
- http://wazniak.mimuw.edu.pl/index.php?title=Wst%C4%99p_do_programowania_w_j%C4%99zyku_C/Wska%C5%BAniki, dostęp online 15.03.2020.
- https://pl.wikibooks.org/wiki/C/Wska%C5%BAniki_-_wi%C4%99cej, dostęp online 15.03.2020.
- Stephen Prata, Język C. Szkoła programowania. Wydanie VI, Wyd. Helion, 2016.
- https://pl.wikibooks.org/wiki/C/Tablice, dostęp online 20.03.2020.
- https://pl.wikibooks.org/wiki/C/Tablice_-_wi%C4%99cej, dostęp online 20.03.2020.