Bezpieczeństwo oprogramowania - lab 3

Zadanie 1

Kod błędu:

```
memcpy(bp, pl, payload);
```

Źródło:

What is the Heartbleed bug, how does it work and how was it fixed? | CSO Online

Opis problemu:

Błąd występujący w popularnej bibliotece OpenSSL pozwalał na odczyt danych chronionych przez szyfrowanie SSL i TSL. Usterka polegała na wykorzystaniu wyżej wypisanego polecenia, gdzie `pl` oznaczało miejsce źródłowe kopiowania danych, a `payload` oznaczał długość tych danych, jednak nie została zaimplementowana weryfikacja, czy długość danych w `pl` odpowiada podanej długości `payload`.

Przykładowa poprawka:

```
if (1 + 2 + 16 > s->s3->relent)
return 0;
hbtype = *p++;
n2s(p, payload);
if (1 + 2 + payload + 16 > s->s3->rrec.length)
return 0;
pl = p;
```

Zadanie 2

1. Możliwość zrzutu pamięci - nullowanie zmiennych przechowujących wrażliwe dane w momencie gdy nic już na nie nie wskazuje, nieużywanie niemutowalnych typów dla tych zmiennych
2. Plik wymiany/pamięć wirtualna - nadpisywanie sektorów dysku zerami po zwolnieniu pamięci, szyfrowanie dysku
3. Hibernacja systemu - szyfrowanie dysku, uwierzytelnianie pre-boot
4. Możliwość odczytu zawartości RAM po restarcie systemu - nullowanie zmiennych

przechowujących wrażliwe dane w momencie gdy nic już na nie nie wskazuje; niektóre systemy umożliwiają funkcję czyszczenia pamięci po uruchomieniu sekwencji restartu systemu; używanie zewnętrznego magazynu kluczy

Klucze kryptograficzne w dużych zrzutach pamięci można znaleźć przy użyciu szerokiego wachlarza narzędzi, które potrafią zidentyfikować ciągi znaków AES w pliku, takich jak *winhex*, *TrueCrypt* albo *findaes*.

Zadanie 3

- mlock - blokuje strony w zakresie pamięci zaczynając od `addr` i kontynuując przez `len` bajtów; strony pozostają w RAMie aż do odblokowania,
- munlock - odblokowuje strony w zakresie pamięci zaczynając od `addr` i kontynuując przez `len` bajtów,
- mlockall - blokuje wszystkie strony powiązane z przestrzenią adresów procesu wywołującego metodę,
- munlockall - odblokowuje wszystkie strony powiązane z przestrzenią adresów procesu wywołującego metodę.

JVM API nie wspiera blokowania pamięci. C# także posiada mechanizm garbage collector, stąd blokowanie pamięci jest niedostepne.

Zadanie 4

```
#include <sys/mman.h>
#include <iostream>
#include <cstdlib>

int main() {
    int* preKey = new int[16];
    for (int i = 0; i < 16; i++) {
        preKey[i] = rand() % 2;
    }
    mlockall(MCL_CURRENT);
    munlockall();
    delete[] preKey;
    return 0;
}
```

Zadanie 5

Trwałe usunięcie danych z nośników danych można wykonać za pomocą demagnetyzera, jednak taki nośnik danych po poddaniu procesowi demagnetyzacji nie nadaje się do dalszego użycia. Problemem jest także wysoka cena demagnetyzatora sięgająca nawet kilkadziesiąt tysięcy złotych.

#school