

# Bezp. syst. i usług inform. 2

## Modyfikacja pliku binarnego

Adam Zimny 209787

27 listopada 2016

## 1 Cel projektu

Celem laboratorium było zmodyfikowanie pliku binarnego programu w taki sposób, aby każde podane hasło akceptowane było jako poprawne.

## 2 Realizacja

W celu przystąpienia do analizy pliku należy poddać go deasemblacji przy pomocy komendy

```
objdump -d patch_me_bin > code
```

Wywołanie tej komendy tworzy plik tekstowy o nazwie code, którego zawartością jest kod pliku patch\_me\_bin w języku assembler.

Czytając kod programu odnaleźć można wywołania kolejnych funkcji. Pozwala to wstępnie odtworzyć szkielet kodu programu:

Listing 1: Szkielet programu

```
winner(){
    print("winner");
}

looser(){
    print("looser");
}

cheater(){
    print("out of time");
}

main(){
    init();
    if(timeguard() != 0){
        cheater();
        return 2;
    }

    kod = scanf();
    kod = obliczenia(kod);
    eax = coś;

    call tab[coś];    // winner lub loser;
}

/**
    jeśli między kolejnymi wywołaniami minie więcej niż 2 sekundy to cheater();
*/
int timeguard(){
    static int czas21 = 2016-11-21 21:00;
    teraz = getTime();
    r = teraz - czas21;
    if(r > 2)
```

```

        return -1;
    czas21 = getTime();
    return 0;
}

void init(){
    for(int i = 0; i < 42; i++){
        edx = &looser;
        tab[i] = edx;
    }

    code = generateCode();
    edx = &winner
    tab[code] = edx;
}

generate_code(){
}

int obliczenia(kod){
}

```

Funkcja `init()` tworzy tablicę o rozmiarze 42 pól z adresami do funkcji `looser`, a następnie zastępuje jeden z nich na nieznanej pozycji adresem funkcji `winner`. Po uruchomieniu programu i podaniu kodu, wykonywane są na nim pewne obliczenia, po czym wykorzystywany jest on do indeksowania tablicy i wywołania funkcji spod odpowiedniego adresu. Poprzez analizę kodu funkcji `generateCode` oraz `obliczenia` możliwe byłoby znalezienie poprawnego hasła, jednak jego znajomość nie jest wymagana do wykonania wymaganych modyfikacji.

## 2.1 Sposób 1. Modyfikacja tabeli z adresami funkcji

Pierwszym sposobem ominięcia zabezpieczeń wykonanym w ramach realizacji laboratorium jest podmiana wartości przypisywanych w pętli inicjującej tabelę tak, aby wszystkie jej pozycje zajmował adres funkcji `winner`. Po wykonaniu tej modyfikacji niezależnie od podanego hasła wywołana zostanie poprawna funkcja.

Inicjalizacją tablicy zajmuje się funkcja `init`, której kod pokazano na poniższym listingu. W uwidocznionej środkowej części kodu będącej pętlą znaleźć można instrukcję `mov $0x8048552,%edx`, która odpowiada przypisaniu adresu funkcji `looser` do rejestru `edx`. Po zakończeniu pętli wykonywana jest kolejna instrukcja przypisania `mov $0x8048566,%edx`, tym razem przenosząca adres `winner`. Poprzez podmianę przeniesienia wewnątrz pętli na to wykonywane po jej zakończeniu, tablica wypełniona zostanie adresami `winner`.

```

0804857a <init>:
804857a: 55                push    %ebp
804857b: 89 e5             mov     %esp,%ebp
804857d: 83 ec 18          sub     $0x18,%esp
8048580: c7 45 f4 00 00 00 movl    $0x0,-0xc(%ebp)
8048587: eb 13             jmp     804859c <init+0x22>

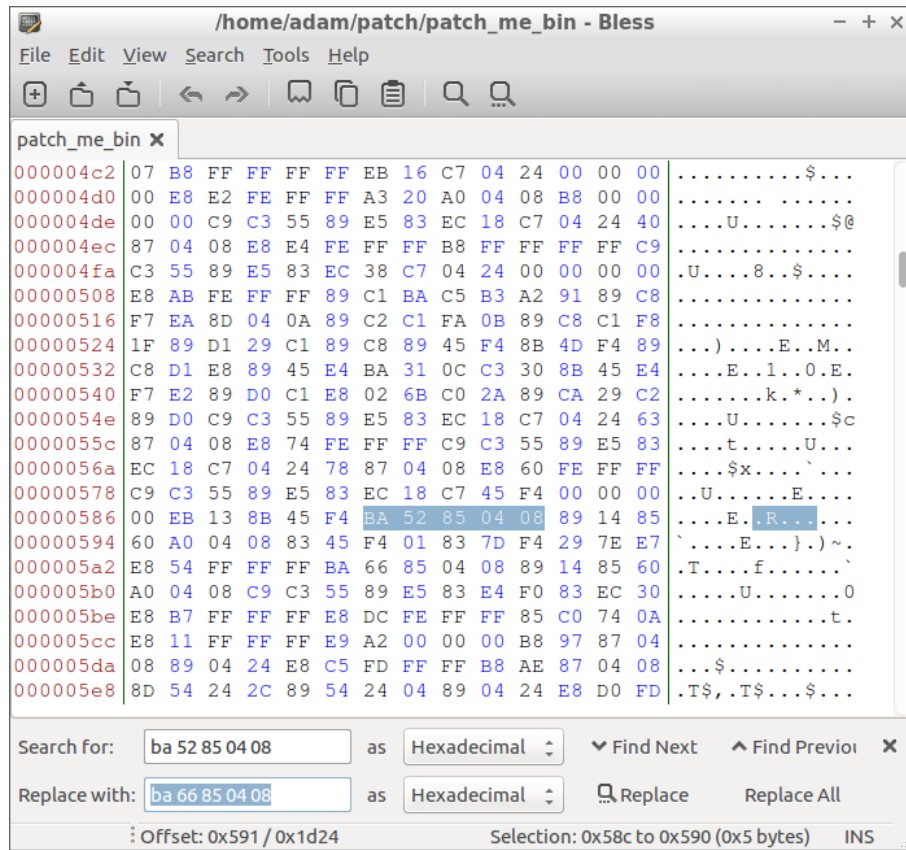
8048589: 8b 45 f4           mov     -0xc(%ebp),%eax
804858c: ba 52 85 04 08    mov     $0x8048552,%edx
8048591: 89 14 85 60 a0 08 mov     %edx,0x804a060(,%eax,4)
8048598: 83 45 f4 01        addl    $0x1,-0xc(%ebp)
804859c: 83 7d f4 29        cmpl    $0x29,-0xc(%ebp)
80485a0: 7e e7             jle     8048589 <init+0xf>

80485a2: e8 54 ff ff ff    call    80484fb <generate_code>
80485a7: ba 66 85 04 08    mov     $0x8048566,%edx
80485ac: 89 14 85 60 a0 08 mov     %edx,0x804a060(,%eax,4)
80485b3: c9                leave   %eax
80485b4: c3                ret

```

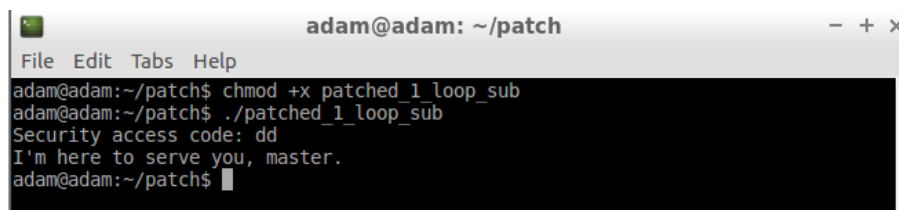
Do wykonania modyfikacji wykorzystano program Bless. Po otwarciu pliku binarnego `patch_me.bin` wyświetlony zostaje jego kod w postaci szesnastkowej. Postać szesnastkową instrukcji odnaleźć można

w drugiej kolumnie pliku wygenerowanego komendą `objdump`. Korzystając z narzędzia Znajdź i zamień wykonywana jest modyfikacja.



Rysunek 1: Modyfikacja pliku binarnego z użyciem programu Bless

Po zapisaniu zmodyfikowanego pliku pod nazwą `patched_1_loop_sub` i nadaniu uprawnień wykonywania utworzonego pliku możliwe jest sprawdzenie jego działania:



Rysunek 2: Efekt działania programu po wykonaniu modyfikacji

## 2.2 Sposób 2. Modyfikacja funkcji loser

Drugim sposobem jest modyfikacja funkcji `loser` tak, aby wykonała te same operacje co funkcja `winner`. Porównując kod obu funkcji zauważyć można, że jedyną różnicą w ich implementacji jest adres, spod którego pobierany jest ciąg znaków wyświetlany na ekranie przy pomocy funkcji `puts`.

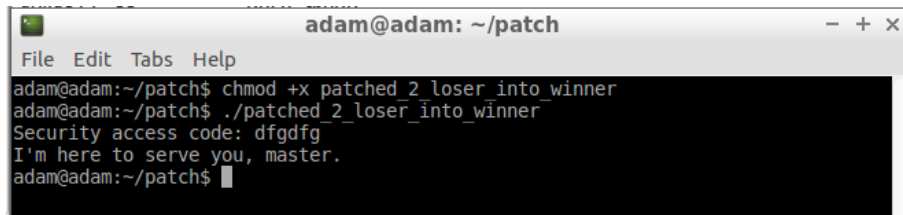
Listing 2: Funkcja loser

```
08048552 <loser>:
push    %ebp
mov     %esp,%ebp
sub     $0x18,%esp
movl    $0x8048763,(%esp)
call    80483d8 <puts@plt>
leave
ret
```

Listing 3: Funkcja winner

```
08048566 <winner>:
push    %ebp
mov     %esp,%ebp
sub     $0x18,%esp
movl    $0x8048778,(%esp)
call    80483d8 <puts@plt>
leave
ret
```

Po zmianie w funkcji **looser** adresu tesktu na ten z funkcji **winner**, ich działanie będzie jednakowe. Do modyfikacji pliku ponownie wykorzystano program Bless. Po modyfikacji wynik działania programu jest następujący:



```
adam@adam: ~/patch
File Edit Tabs Help
adam@adam:~/patch$ chmod +x patched_2_loser_into_winner
adam@adam:~/patch$ ./patched_2_loser_into_winner
Security access code: dfgdfg
I'm here to serve you, master.
adam@adam:~/patch$
```

Rysunek 3: Efekt działania programu po wykonaniu modyfikacji 2

### 3 Podsumowanie

Praca podczas laboratorium pokazała, jak poprzez edycję pliku binarnego możliwa jest zmiana działania programu. W ramach realizacji wykonane zostały dwie niezależne modyfikacje pozwalające na dostęp do zabezpieczonych funkcji programu bez znajomości poprawnego hasła. Do edycji pliku binarnego wykorzystane zostało narzędzie Bless.