Bezpieczeństwo usług sieciowych
— laboratorium 4 —
crackme

Adrian Frydmański

16 stycznia 2018

1 Zadanie do wykonania

Zadaniem była zmiana kodu binarki na poziomie bitów, by program wykonał się tak, jak chce tego autor – wykonał funkcję `winner`.

2 Kroki prowadzące do rozwiązania

Deasemblacja wymagała wpisania `objdump -D crackme`. Pierwszym – i całkiem trafnym – pomysłem była podmiana adresu wywoływanej funkcji. Oto adresy, na które można było zwrócić uwagę:

- sekcja `<cheater>` – 80484e2
- sekcja `<winner>` – 8048566
- sekcja `<looser>` – 8048552
- sekcja `<init>` – 804857a
- wywołanie `<init>` w `<main>` – 80485be
- wywołanie `<cheater>` w `<main>` – 80485cc

2.1 Sposób 1

Proponowaną zmianą jest podmiana adresu `<init>` na `<winner>` w wywołaniu `<init>` w `<main>`:

```
08048552 <looser>:
 8048552: 55  push %ebp
 8048553: 89 e5  mov %esp,%ebp
 8048555: 83 ec 18  sub $0x18,%esp
 8048558: c7 04 24 63 87 04 08  movl $0x8048763,(%esp)
 804855f: e8 74 fe ff ff  call 80483d8 <puts@plt>
 8048564: c9  leave
```

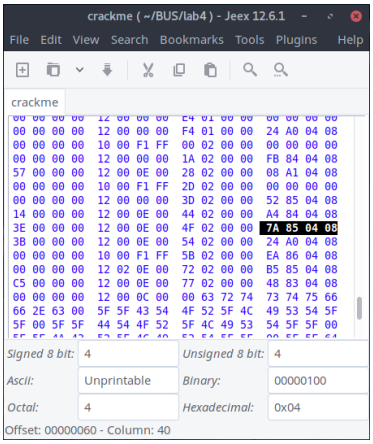
```
 8048565: c3  ret

08048566 <winner>:
 8048566: 55  push %ebp
 8048567: 89 e5  mov %esp,%ebp
 8048569: 83 ec 18  sub $0x18,%esp
 804856c: c7 04 24 78 87 04 08  movl $0x8048778,(%esp)
 8048573: e8 60 fe ff ff  call 80483d8 <puts@plt>
 8048578: c9  leave
 8048579: c3  ret

080485b5 <main>:
 80485b5: 55  push %ebp
 80485b6: 89 e5  mov %esp,%ebp
 80485b8: 83 e4 f0  and $0xfffffff0,%esp
 80485bb: 83 ec 30  sub $0x30,%esp
 80485be: e8 b7 ff ff ff  call 804857a <init>
 80485c3: e8 dc fe ff ff  call 80484a4 <time_guard>
 80485c8: 85 c0  test %eax,%eax
 80485ca: 74 0a  je 80485d6 <main+0x21>
 80485cc: e8 11 ff ff ff  call 80484e2 <cheater>
```

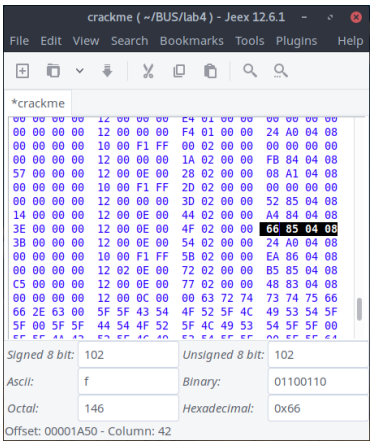
Listing 1: Fragment objdumpa

W celu edycji posłużyłem się programem do edycji plików binarnych Jeex. Wyszukałem adres sekcji (funkcji) w pliku na podstawie danych z objdumpa. Wystepował tylko raz, więc miałem pewność, że to ten.



Rysunek 1: Znaleziony adres funkcji init

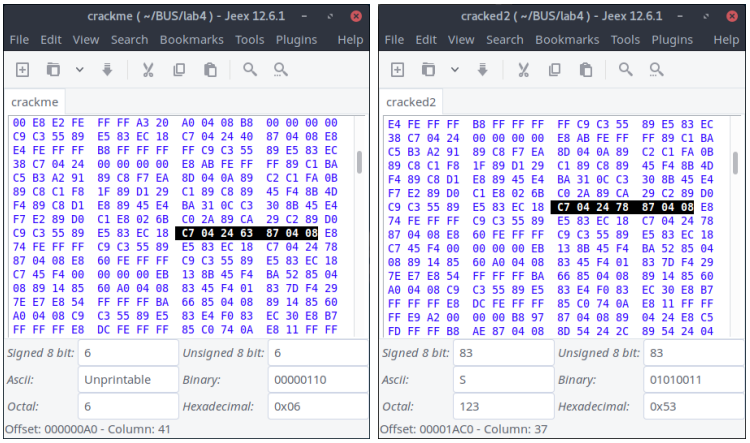
Na co należało zwrócić uwagę, to fakt, iż przy wyszukiwaniu adresu trzeba zmienić kolejność bajtów, gdyż w pliku binarnym są podawane od najmłodszego.



Rysunek 2: Adres funkcji init zmieniony na adres funkcji winner

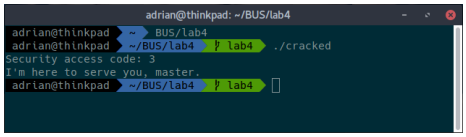
Ostatecznie zmianie uległ najmłodszy bajt adresu – z `0x7a` (`01100011b`) na `0x66` (`01111000b`) – co dało 4 zmienione bity.

Funkcje `<winner>` i `<looser>` są prawie takie same i można w nich zamienić adres tekstku drukowanego na ekran przez wywołanie `<puts@plt>`.



Rysunek 3: Adresy stringów drukowanych na ekran

W efekcie tego działania po wpisaniu dowolnego znaku (lub znaków) wywoływana jest funkcja winner.

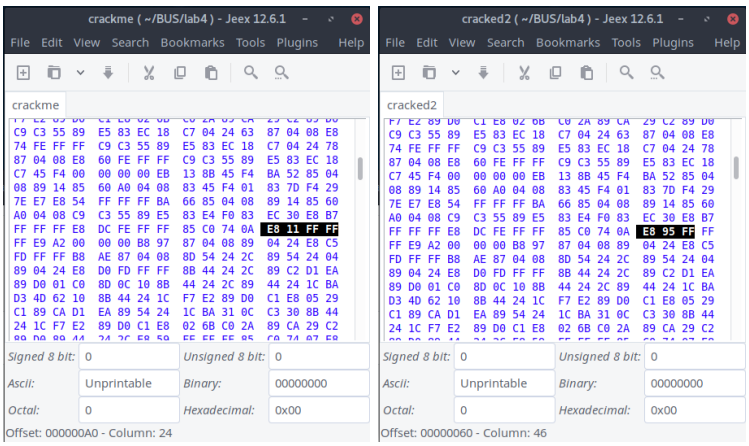


Rysunek 4: Wynik działania ze zmeinionym adresem funkcji

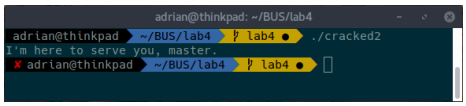
Taki efekt w momencie pisania sprawozdania był możliwy po zmianie czasu systemu przez porównywanie czasu w programie z timestampem z dnia zajęć.

2.2 Sposób 2

Jest też inna możliwość. Można zamienić pierwsze wywołanie `<cheater>` w `<main>` na wywołenie `<winner>`. W callu jest przesunięcie względem aktualnego adresu. Instrukcję `80485cc: e8 11 ff ff ff call 80484e2 <cheater>` należy zamienić na `e8 95 ff ff ff` – przy zmianie `11` (`00010001`) na `95` (`10010101`) uzyskujemy zmianę tylko 2 bitów. Wynika to z różnicy w adresach: $0x080484e2 - 0x08048566 = 0x84$. Tyle właśnie należało dodać do przesunięcia – początkowo oznaczonego jako 0x11 w instrukcji – aby przenieść się o 0x84 dalej, do funkcji `<winner>`. Niestety takie wykonanie programu kończy się tak, jak w przypadku otrzymania komunikatu o błędzie – z inną od zera zwróconą wartością – i było możliwe dopiero po zakończeniu zajęć laboratoryjnych albo po zmianie daty w systemie.



Rysunek 5: Zmiana calla



Rysunek 6: Wynik działania ze zmienionym callem

3 Podsumowanie

Zadanie pokazało kolejne ciekawe aspekty wynikające ze znajomości assemblera i umiejętności operowania na plikach binarnych.

Plik `cracked` zawiera binarkę zmienioną pierwszym sposobem, zaś `cracked2` – drugim.