

# Bezpieczeństwo systemów komputerowych

## Sprawdzanie poprawności danych

Marcin Peczarski

Instytut Informatyki Uniwersytetu Warszawskiego

9 października 2011

# Lista zagadnień

- ▶ Forma kanoniczna
- ▶ Ciągi formatujące
- ▶ Przepelnienie arytmetyczne

## Forma kanoniczna, zarys problemu

- ▶ Nie wolno podejmować żadnych decyzji mających wpływ na bezpieczeństwo na podstawie nazwy zasobu.
- ▶ Zasób posiada często więcej niż jedną poprawną nazwę.

# Przykłady

- ▶ W pewnym programie rodzicielskiej kontroli dostępu do sieci można było wprowadzić blokadę dostępu do witryny, np.

[www.stwory.pl](http://www.stwory.pl)

Jednak nie blokowało to dostępu, jeśli na końcu nazwy występowała kropka, co jest kanoniczną nazwą domeny, np.

[www.stwory.pl](http://www.stwory.pl).

- ▶ Sąd nakazał pewnemu serwisowi zaprzestania udostępniania utworów chronionych prawami autorskimi. Działanie blokady oparte było o nazwę utworu, np. zablokowane zostały tytuły:

[Candyman](#), [92 degrees](#), [Deepest Chill](#),

ale nie zablokowano nazw:

[AndymanCay](#), [92 degree\\$](#), [Deepest Ci11](#).

# Nazwa pliku

- Plik może być dostępny za pomocą wielu nazw:

Tajny plik.doc

TAJNYP~1.DOC

c:\Dane\Tajne\Tajny plik.doc

c:\Dane\Tajne\Tajny plik.doc.

C:\DANE\TAJNE\TAJNY PLIK.DOC

c:\Dane\Jawne\..\Tajne\Tajny plik.doc

..\..\Tajne\Tajny plik.doc

Tajny plik.doc::\$DATA

\\?\c:\Dane\Tajne\Tajny plik.doc

# URL

- ▶ W nazwach można stosować szesnastkowe kody znaków:

<http://www.dane.com>

<http://www.%64ane.com>

- ▶ W pewnym systemie blokowała się myszy, gdy pojawił się znacznik

`<IMG SRC=file:///dev/mouse>`

który można też zakodować tak

`<IMG SRC=file:///dev&#x2Fmouse>`

# UTF-8

- ▶ Niektóre analizatory UTF-8 akceptują nielegalne, przedłużone kodowania, np. kodem znaku „?” jest:

3F

a nielegalne kodowania tego znaku wyglądają następująco:

C0 BF

E0 80 BF

F0 80 80 BF

F8 80 80 80 BF

FC 80 80 80 80 BF

- ▶ Należy uważnie obliczać długość napisów.

## Na marginesie, typ `wchar_t`

- ▶ Typ `wchar_t` ma w Linuksie rozmiar 4 bajty, a w Windows 2 bajty.
- ▶ Funkcje operujące na ciągach znaków tego typu jako rozmiar bufora oczekują liczby znaków, a nie liczby bajtów.



## Ciągi formatujące, zarys problemu

- Czym różnią się poniższe wywołania?

```
printf(napis);  
printf("%s", napis);  
printf("%.*s", dlugosc, napis);
```

# Atak typu unieruchomienie usługi (ang. DoS)

- ▶ Rozważmy program:

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

```
int main(int argc, char *args[]) {  
    printf(args[1]);  
    return 0;  
}
```

- ▶ Skompilujmy go:

```
gcc -Wall -O2 -o cform cform.c
```

- ▶ Spróbujmy go uruchomić:

```
$ ./cform "%s"
```

```
$ ./cform "%s%s"
```

```
...
```

```
$ ./cform "%s%s%s%s%s%s%s%s%s%s"
```

Naruszenie ochrony pamięci

# Atak zmieniający działanie programu (1)

- Rozważmy program:

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

```
long i;
```

```
void foo(const char *s, long *j) {  
    *j = 7;  
    printf(s);  
}
```

```
int main(int argc, char *args[]) {  
    foo(args[1], &i);  
    printf("\ni = %ld\n", i);  
    return 0;  
}
```

- Skompilujmy go:

```
gcc -Wall -O2 -fno-inline -o cform cform.c
```

## Atak zmieniający działanie programu (2)

- ▶ Co wypisze ten program?
- ▶ Uruchommy go:

```
$ ./cform
```

```
i = 7
```

```
$ ./cform "abc"
```

```
abc
```

```
i = 7
```

- ▶ Wydaje się, że działa zgodnie z oczekiwaniem.

## Atak zmieniający działanie programu (3)

- ▶ A co będzie, gdy podamy bardziej „wyrafinowany” argument?

```
$ ./cform "a%n"
```

```
a
```

```
i = 1
```

```
$ ./cform "abc%n"
```

```
abc
```

```
i = 3
```

```
$/cform "abcdefghijklmnopqrstuvwxyzn"
```

```
abcdefghijklmnopqrstuvwxyzn
```

```
i = 26
```

- ▶ Mogę zmieniać wartość zmiennej w programie!

## Przepełnienie arytmetyczne, zarys problemu

- Co wypisze poniższy program?

```
#include <stdio.h>

int main() {
    char    c;
    unsigned u;
    signed  s;

    c = 128;
    printf("c = %i\n", c);
    u = 1;
    s = -1;
    if (u > s)
        printf("u > s\n");
    else
        printf("u <= s\n");
    return 0;
}
```

## Przepełnienie arytmetyczne, cd.

- ▶ Na architekturze x86 program wypisuje

```
c = -128
```

```
u <= s
```

- ▶ Znany jest przykład ataku wykorzystującego przepełnienie arytmetyczne, <http://cve.mitre.org/cgi-bin/cvename.cgi?name=2008-2249>.