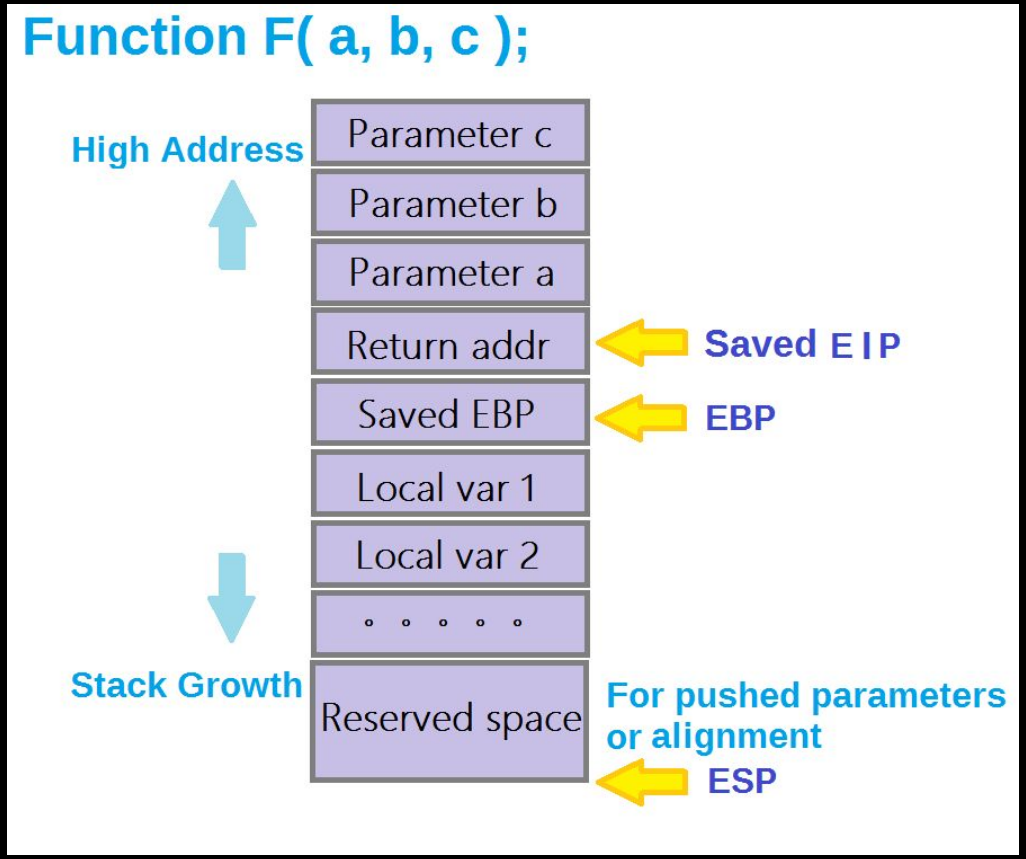
Buffer overflow

Bamboofox & NCTUCSC

Outline

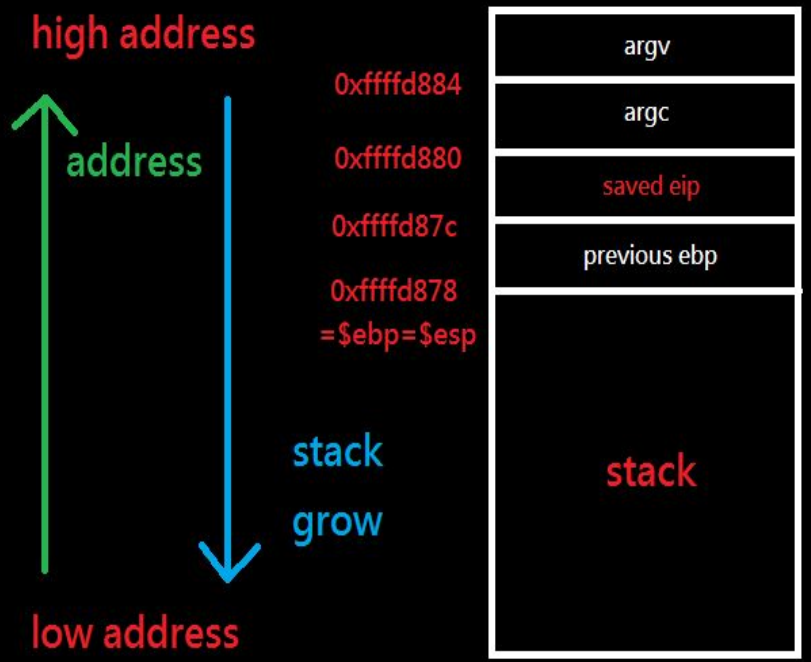
1. Stack frame
2. Buffer overflow
3. Dangerous function
4. Strategy
5. Stack canary

Stack frame


Function F(a, b, c);
High Address
Parameter c
Parameter b
Parameter a
Return addr
Saved EIP
Saved EBP
EBP
Local var 1
Local var 2
. . . . .
Stack Growth
Reserved space
For pushed parameters or alignment
ESP

Buffer overflow

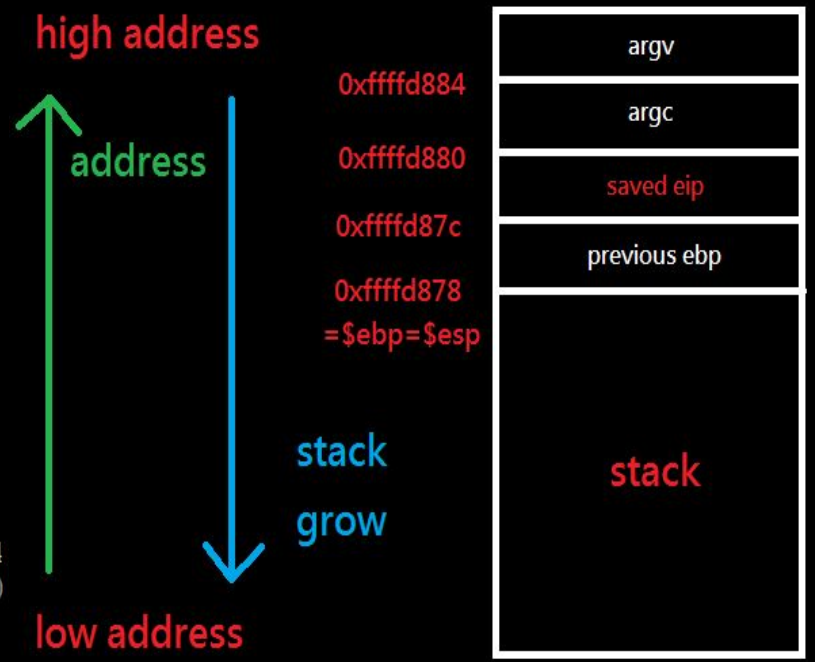
```
1 #include<stdio.h>
2 int main(void)
3 {
4       char s[10];
5       scanf("%s",s);
6       return 0;
7 }
8
```



Buffer overflow

```
Reading symbols from bof...(no debugging symbols found)...done.
(gdb) b main
Breakpoint 1 at 0x804843f
(gdb) r
Starting program: /net/cs/101/0116320/bamboofox/bof/bof

Breakpoint 1, 0x0804843f in main ()
(gdb) info frame
Stack level 0, frame at 0xffffd880:
 eip = 0x804843f in main; saved eip = 0xf7e12637
 Arglist at 0xffffd878, args:
 Locals at 0xffffd878, Previous frame's sp is 0xffffd880
 Saved registers:
  ebp at 0xffffd878, eip at 0xffffd87c
(gdb) x/10x $ebp previous ebp        eip          argc         argv
0xffffd878:     0x00000000      0xf7e12637      0x00000001      0xffffd914
0xffffd888:     0xffffd91c      0x00000000      0x00000000      0x00000000
0xffffd898:     0xf7fad000      0x08048224
(gdb)
```

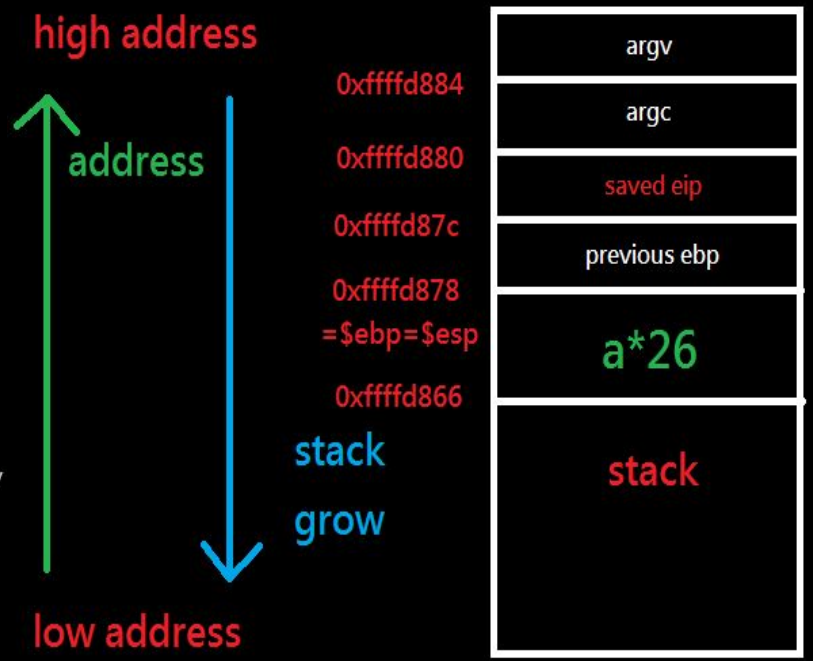


Buffer overflow

```
Reading symbols from bof...(no debugging symbols found)...done.
(gdb) b __isoc99_scanf
Breakpoint 1 at 0x8048340
(gdb) r
Starting program: /net/cs/101/0116320/bamboofox/bof/bof

Breakpoint 1, 0xf7e58af6 in __isoc99_scanf () from /usr/lib32/libc.so.6
(gdb) info frame
Stack level 0, frame at 0xffffd850:
 eip = 0xf7e58af6 in __isoc99_scanf; saved eip = 0x8048459
 called by frame at 0xffffd880
 Arglist at 0xffffd848, args:
 Locals at 0xffffd848, Previous frame's sp is 0xffffd850
 Saved registers:
  ebx at 0xffffd83c, ebp at 0xffffd848, esi at 0xffffd840, edi at 0xffffd844,
  eip at 0xffffd84c
(gdb) x/10x $ebp
0xffffd848:     0xffffd878      0x08048459      0x080484f0      0xffffd866
0xffffd858:     0xffffd91c      0xf7e28b7b      0xf7fad3dc      0x080481f4
0xffffd868:     0x0804847b      0x00000000
(gdb)
```

previous ebp　eip　"%s" string address　buf address

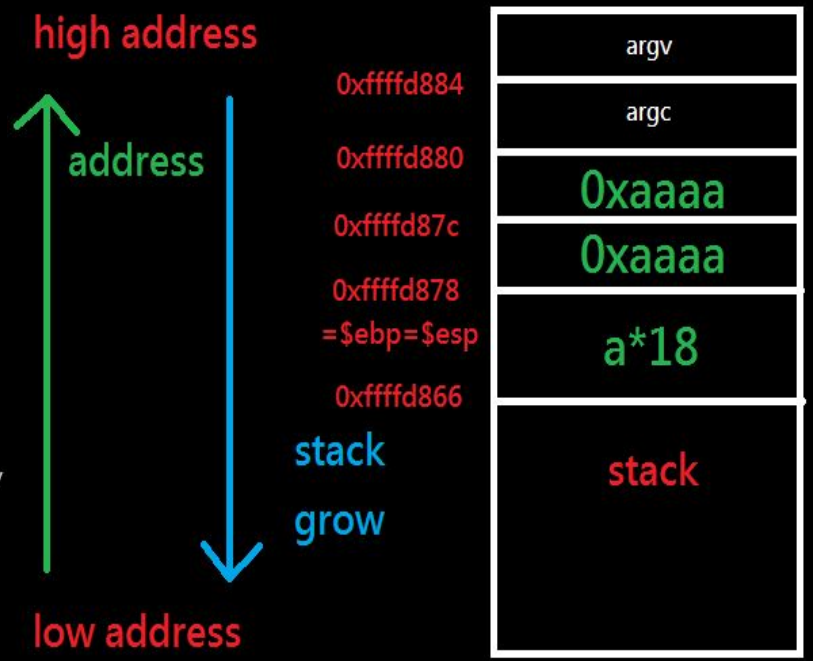


Buffer overflow

```
Reading symbols from bof...(no debugging symbols found)...done.
(gdb) b __isoc99_scanf
Breakpoint 1 at 0x8048340
(gdb) r
Starting program: /net/cs/101/0116320/bamboofox/bof/bof

Breakpoint 1, 0xf7e58af6 in __isoc99_scanf () from /usr/lib32/libc.so.6
(gdb) info frame
Stack level 0, frame at 0xffffd850:
 eip = 0xf7e58af6 in __isoc99_scanf; saved eip = 0x8048459
 called by frame at 0xffffd880
 Arglist at 0xffffd848, args:
 Locals at 0xffffd848, Previous frame's sp is 0xffffd850
 Saved registers:
  ebx at 0xffffd83c, ebp at 0xffffd848, esi at 0xffffd840, edi at 0xffffd844,
  eip at 0xffffd84c
(gdb) x/10x $ebp
0xffffd848:     0xffffd878      0x08048459      0x080484f0      0xffffd866
0xffffd858:     0xffffd91c      0xf7e28b7b      0xf7fad3dc      0x080481f4
0xffffd868:     0x0804847b      0x00000000
(gdb)
```

previous ebp　eip　"%s" string address　buf address



Buffer overflow

```
Reading symbols from bof...(no debugging symbols found)...done.
(gdb) r
Starting program: /net/cs/101/0116320/bamboofox/bof/bof
aaaaaaaaaaaaaaaaaaaaaaaaaa a*26

Program received signal SIGSEGV, Segmentation fault.
0x61616161 in ?? ()
(gdb)
```

Calc offset from the buffer to EIP

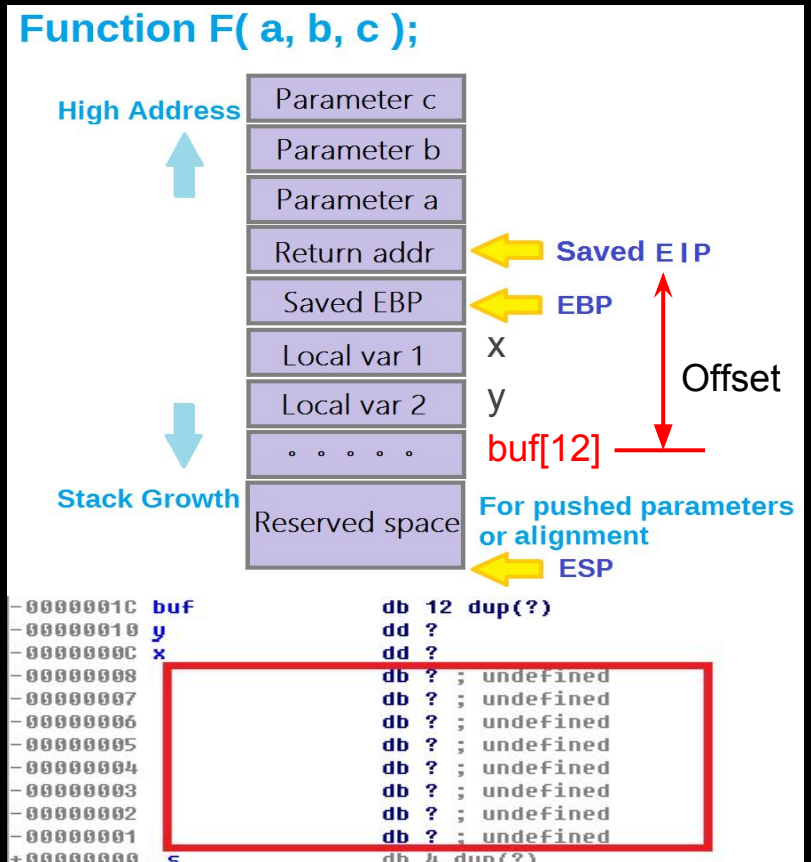
For example,

```
F(a,b,c){
        int x,y;
        char buf[12];
        scanf("%s",buf);
}
```

EIP = EBP + 4

buf[0] = EBP-12(buf)-8(2 int)-8(align)

=> 28 bytes !



Easy practice

Mangic

http://ctf.cs.nctu.edu.tw/problems/1

http://train.cs.nctu.edu.tw/files/magic

nc 140.113.209.24 11000

Overflow

http://ctf.cs.nctu.edu.tw/problems/2

Dangerous function

1. gets → fgets
2. sprintf → snprintf
3. strcpy → strncpy
4. strcat → strncat
5. scanf → never use %s format

Strategy

With this way, we could control eip.

And combine other strategies to get shell.

- Shellcode
- Return to libc
- Return-oriented programming

But there is a way to protect against the buffer overflow, which is called Stack Canary.

Stack canary (stack protector, stack guard)

In default, this protection is disabled.

gcc -fstack-protector

Modify function prologue and epilogue.

After prologue, add 4 bytes random value (called canary)

Before epilogue, xor canary with original value

```
if ( zero /* a.k.a equivalent */ ) return;
else terminated /* send the SIGSEGV signal */;
```

How does Linux generate the value of the stack canary :
https://xorl.wordpress.com/2010/10/14/linux-glibc-stack-canary-values/

Bypass stack canary

Leak stack canary

- Canaries are fixed after load.
- Leak it and overwrite stack with a same value.

Skip stack canary

- Overwrite a buffer pointer which would be wrote and point it to return address, directly write EIP.
- Overwrite a function pointer which would be called.

Reference

Behind the process (Recommand!)
http://bottomupcs.sourceforge.net/csbu/x3564.htm

Secure Programming : Smashing the stack
http://secprog.cs.nctu.edu.tw/files/Secure_Programming_Smashing_the_Stack.pdf

Stack Canary
https://en.wikipedia.org/wiki/Buffer_overflow_protection#GCC_Stack-Smashing_Protector_.28ProPolice.29