

# strlen v1

```
1 size_t strlen_a(const char * str) {  
2     size_t length = 0 ;  
3     while (*str++)  
4         ++length;  
5     return length;  
6 }
```

# strlen v2

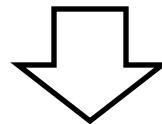
```
1 size_t strlen_b(const char * str) {  
2     const char *cp = str;  
3     while (*cp++)  
4         ;  
5     return (cp - str - 1);  
6 }
```

# strlen in c standard library

```
9     for (char_ptr = str; ((ulong)char_ptr
10          & (sizeof(ulong) - 1)) != 0 ;
11          ++char_ptr) {
12     if (*char_ptr == '\0')
13         return char_ptr - str;
14 }
15
16 longword_ptr = (ulong*)char_ptr;
```

# The Trick

```
18     magic_bits = 0x7effeffL;
```



b3	b2	b1	b0
31----->0			
magic_bits: 01111110 11111110 11111110 11111111			

The Purpose: test 0 byte

b3	b2	b1	b0
31----->0			
longword: XXXXXXXX XXXXXXXX 00000000 XXXXXXXX			
+ magic_bits: 01111110 11111110 11111110 11111111			

# The Trick

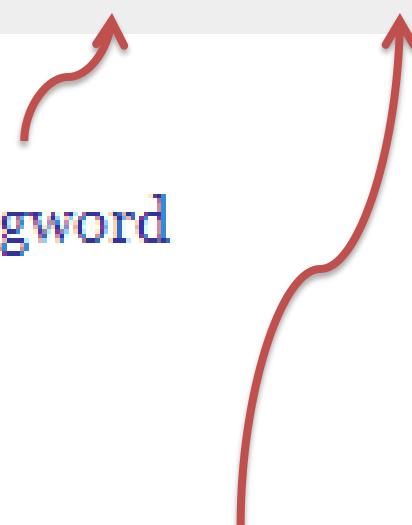
We find if there exists all 0 byte by

```
23
24     if (((longword + magic_bits) ^ ~longword) & ~magic_bits) != 0 ) {
```

Get unchanged bits in longword

$\wedge \sim \text{longword}$

Originally longword=01  
+magic make it 11,  
 $11 \wedge \sim 01 = 11 \wedge 10 = 01$



$\wedge \sim \text{magic\_bits}$  Get unchanged “hole”

Results=0 if all unchanged  
Otherwise, !=0

# The Trick

	b3	b2	b1	b0			
	31	-----	-----	-----	>0		
	longword:	00001001	00011000	00000000	00001100		
+	magic_bits:	01111110	11111110	11111110	11111111		
		sum:	10001000	00010110	11111111	00001011	
^	~longword:	11110110	11100111	11111111	11110011		
		a:	01111110	11110001	00000000	11111000	
&	~magic_bits:	10000001	00000001	00000001	00000000		
		result:	00000000	00000001	00000000	00000000	

# The Trick

If we find it, compute the offset

```
24     if (((longword + magic_bits) ^ ~longword) & ~magic_bits) != 0 ) {  
25  
26         const char *cp = (const char*)(longword_ptr - 1);  
27  
28         if (cp[0] == 0)  
29             return cp - str;  
30         if (cp[1] == 0)  
31             return cp - str + 1;  
32         if (cp[2] == 0)  
33             return cp - str + 2;  
34         if (cp[3] == 0)  
35             return cp - str + 3;  
36     }
```

# The Trick

If we don't find it, continue the while loop

```
20     while (1) {  
21  
22         longword = *longword_ptr++;    
23  
24         if (((longword + magic_bits) ^ ~longword) & ~magic_bits) != 0 ) {  
25  
26             const char *cp = (const char*)(longword_ptr - 1);   Why -1?  
27  
28             if (cp[0] == 0)  
29                 return cp - str;  
30             if (cp[1] == 0)  
31                 return cp - str + 1;  
32             if (cp[2] == 0)  
33                 return cp - str + 2;  
34             if (cp[3] == 0)  
35                 return cp - str + 3;  
36         }  
37     }
```