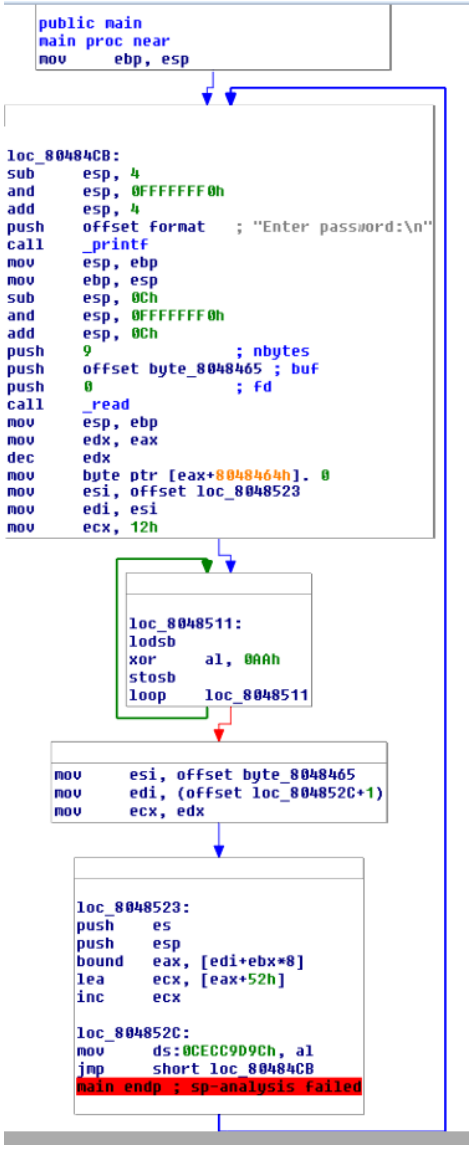
Crackme

- A crackme example: Easy but a good one to start !
- Author : Emmanuel Fleury (University of Bordeaux)

launch it !

```
1  $ ./crackme
2  Enter password:
3  mypass
4  Wrong password
```

Open with IDA



Static view

- Strange loop ?
- IDA seems to not be able to understand the code
- Strange loop + IDA lost = usually self modifying code

Let's play with gdb

```
(gdb) disas main
Dump of assembler code for function main:
   0x080484c9 <+0>:     mov    %esp,%ebp
   0x080484cb <+2>:     sub    $0x4,%esp
   0x080484ce <+5>:     and    $0xfffffff0,%esp
   0x080484d1 <+8>:     add    $0x4,%esp
   0x080484d4 <+11>:    push   $0x8048454
   0x080484d9 <+16>:    call   0x8048360 <printf@plt>
   0x080484de <+21>:    mov    %ebp,%esp
   0x080484e0 <+23>:    mov    %esp,%ebp
   0x080484e2 <+25>:    sub    $0xc,%esp
   0x080484e5 <+28>:    and    $0xfffffff0,%esp
   0x080484e8 <+31>:    add    $0xc,%esp
   0x080484eb <+34>:    push   $0x9
   0x080484ed <+36>:    push   $0x8048465
   0x080484f2 <+41>:    push   $0x0
   0x080484f4 <+43>:    call   0x8048350 <read@plt>
   0x080484f9 <+48>:    mov    %ebp,%esp
   0x080484fb <+50>:    mov    %eax,%edx
   0x080484fd <+52>:    dec    %edx
   0x080484fe <+53>:    movb   $0x0,0x8048464(%eax)
   0x08048505 <+60>:    mov    $0x8048523,%esi
   0x0804850a <+65>:    mov    %esi,%edi
   0x0804850c <+67>:    mov    $0x12,%ecx
   0x08048511 <+72>:    lods   %ds:(%esi),%al
   0x08048512 <+73>:    xor    $0xaa,%al
   0x08048514 <+75>:    stos   %al,%es:(%edi)
   0x08048515 <+76>:    loop   0x8048511 <main+72>
   0x08048517 <+78>:    mov    $0x8048465,%esi
   0x0804851c <+83>:    mov    $0x804852d,%edi
   0x08048521 <+88>:    mov    %edx,%ecx
   0x08048523 <+90>:    push   %es
   0x08048524 <+91>:    push   %esp
   0x08048525 <+92>:    bound  %eax,(%edi,%ebx,8)
   0x08048528 <+95>:    lea    0x52(%eax),%ecx
   0x0804852b <+98>:    inc    %ecx
   0x0804852c <+99>:    mov    %al,0xcecc9d9c
   0x08048531 <+104>:   jmp    0x80484cb <main+2>
   0x08048533 <+106>:   out    %al,(%dx)
   0x08048534 <+107>:   stos   %al,%es:(%edi)
   0x08048535 <+108>:   mov    %esp,%ebp
   0x08048537 <+110>:   sub    $0x4,%esp
   0x0804853a <+113>:   and    $0xfffffff0,%esp
   0x0804853d <+116>:   add    $0x4,%esp
   0x08048540 <+119>:   push   $0x804856b
   0x08048545 <+124>:   call   0x8048370 <system@plt>
   0x0804854a <+129>:   mov    %ebp,%esp
   0x0804854c <+131>:   xor    %eax,%eax
```

Call to printf (enter password:)

Call to read (get input from user)

The strange loop

Let's put a breakpoint just after the loop, and see what happened ?

```
1   (gdb) b *main+78
```

```
Dump of assembler code for function main:
   0x080484c9 <+0>:     mov    %esp,%ebp
   0x080484cb <+2>:     sub    $0x4,%esp
   0x080484ce <+5>:     and    $0xfffffff0,%esp
   0x080484d1 <+8>:     add    $0x4,%esp
   0x080484d4 <+11>:    push   $0x8048454
   0x080484d9 <+16>:    call   0x8048360 <printf@plt>
   0x080484de <+21>:    mov    %ebp,%esp
   0x080484e0 <+23>:    mov    %esp,%ebp
   0x080484e2 <+25>:    sub    $0xc,%esp
   0x080484e5 <+28>:    and    $0xfffffff0,%esp
   0x080484e8 <+31>:    add    $0xc,%esp
   0x080484eb <+34>:    push   $0x9
   0x080484ed <+36>:    push   $0x8048465
   0x080484f2 <+41>:    push   $0x0
   0x080484f4 <+43>:    call   0x8048350 <read@plt>
   0x080484f9 <+48>:    mov    %ebp,%esp
   0x080484fb <+50>:    mov    %eax,%edx
   0x080484fd <+52>:    dec    %edx
   0x080484fe <+53>:    movb   $0x0,0x8048464(%eax)
   0x08048505 <+60>:    mov    $0x8048523,%esi
   0x0804850a <+65>:    mov    %esi,%edi
   0x0804850c <+67>:    mov    $0x12,%ecx
   0x08048511 <+72>:    lods   %ds:(%esi),%al
   0x08048512 <+73>:    xor    $0xaa,%al
   0x08048514 <+75>:    stos   %al,%es:(%edi)
   0x08048515 <+76>:    loop   0x8048511 <main+72>
=> 0x08048517 <+78>:    mov    $0x8048465,%esi
   0x0804851c <+83>:    mov    $0x804852d,%edi
   0x08048521 <+88>:    mov    %edx,%ecx
   0x08048523 <+90>:    lods   %ds:(%esi),%al
   0x08048524 <+91>:    dec    %al
   0x08048526 <+93>:    scas   %es:(%edi),%al
   0x08048527 <+94>:    jne    0x8048550 <main+135>
   0x08048529 <+96>:    loop   0x8048523 <main+90>
   0x0804852b <+98>:    jmp    0x8048535 <main+108>
   0x0804852d <+100>:   ss
   0x0804852e <+101>:   aaa
   0x0804852f <+102>:   data16
   0x08048530 <+103>:   fs
   0x08048531 <+104>:   inc    %ecx
   0x08048532 <+105>:   xor    -0x77(%eax,%eax,1),%al
   0x08048536 <+109>:   in     $0x83,%eax
   0x08048538 <+111>:   in     (%dx),%al
   0x08048539 <+112>:   add    $0x83,%al
   0x0804853b <+114>:   in     $0xf0,%al
   0x0804853d <+116>:   add    $0x4,%esp
   0x08048540 <+119>:   push   $0x804856b
   0x08048545 <+124>:   call   0x8048370 <system@plt>
   0x0804854a <+129>:   mov    %ebp,%esp
   0x0804854c <+131>:   xor    %eax,%eax
   0x0804854e <+133>:   jmp    0x804856a <main+161>
   0x08048550 <+135>:   mov    %esp,%ebp
```

The code has changed !

```
(gdb) disas main
Dump of assembler code for function main:
   0x080484c9 <+0>:     mov    %esp,%ebp
   0x080484cb <+2>:     sub    $0x4,%esp
   0x080484ce <+5>:     and    $0xfffffff0,%esp
   0x080484d1 <+8>:     add    $0x4,%esp
   0x080484d4 <+11>:    push   $0x8048454
   0x080484d9 <+16>:    call   0x8048360 <printf@plt>
   0x080484de <+21>:    mov    %ebp,%esp
   0x080484e0 <+23>:    mov    %esp,%ebp
   0x080484e2 <+25>:    sub    $0xc,%esp
   0x080484e5 <+28>:    and    $0xfffffff0,%esp
   0x080484e8 <+31>:    add    $0xc,%esp
   0x080484eb <+34>:    push   $0x9
   0x080484ed <+36>:    push   $0x8048465
   0x080484f2 <+41>:    push   $0x0
   0x080484f4 <+43>:    call   0x8048350 <read@plt>
   0x080484f9 <+48>:    mov    %ebp,%esp
   0x080484fb <+50>:    mov    %eax,%edx
   0x080484fd <+52>:    dec    %edx
   0x080484fe <+53>:    movb   $0x0,0x8048464(%eax)
   0x08048505 <+60>:    mov    $0x8048523,%esi
   0x0804850a <+65>:    mov    %esi,%edi
   0x0804850c <+67>:    mov    $0x12,%ecx
   0x08048511 <+72>:    lods   %ds:(%esi),%al
   0x08048512 <+73>:    xor    $0xaa,%al
   0x08048514 <+75>:    stos   %al,%es:(%edi)
   0x08048515 <+76>:    loop   0x8048511 <main+72>
   0x08048517 <+78>:    mov    $0x8048465,%esi
   0x0804851c <+83>:    mov    $0x804852d,%edi
   0x08048521 <+88>:    mov    %edx,%ecx
   0x08048523 <+90>:    push   %es
   0x08048524 <+91>:    push   %esp
   0x08048525 <+92>:    bound  %eax,(%edi,%ebx,8)
   0x08048528 <+95>:    lea    0x52(%eax),%ecx
   0x0804852b <+98>:    inc    %ecx
   0x0804852c <+99>:    mov    %al,0xcecc9d9c
   0x08048531 <+104>:   jmp    0x80484cb <main+2>
   0x08048533 <+106>:   out    %al,(%dx)
   0x08048534 <+107>:   stos   %al,%es:(%edi)
   0x08048535 <+108>:   mov    %esp,%ebp
   0x08048537 <+110>:   sub    $0x4,%esp
   0x0804853a <+113>:   and    $0xfffffff0,%esp
   0x0804853d <+116>:   add    $0x4,%esp
   0x08048540 <+119>:   push   $0x804856b
   0x08048545 <+124>:   call   0x8048370 <system@plt>
   0x0804854a <+129>:   mov    %ebp,%esp
   0x0804854c <+131>:   xor    %eax,%eax
```

Before loop

The code has changed !

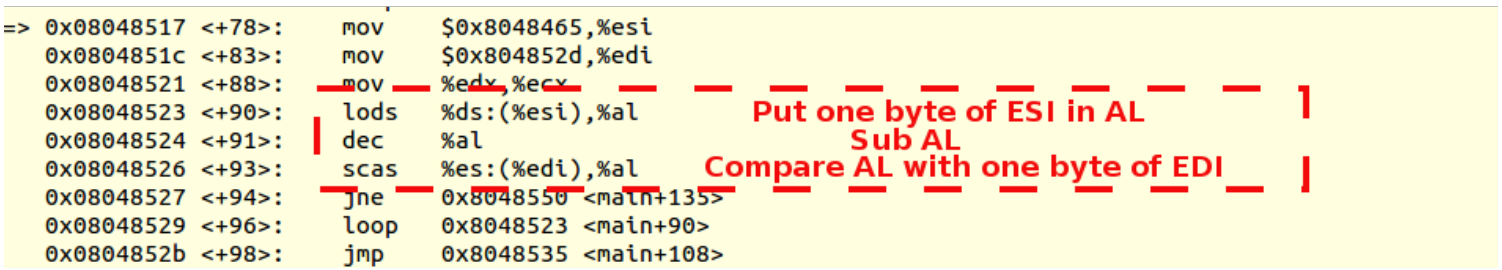
```
Dump of assembler code for function main:
   0x080484c9 <+0>:     mov    %esp,%ebp
   0x080484cb <+2>:     sub    $0x4,%esp
   0x080484ce <+5>:     and    $0xfffffff0,%esp
   0x080484d1 <+8>:     add    $0x4,%esp
   0x080484d4 <+11>:    push   $0x8048454
   0x080484d9 <+16>:    call   0x8048360 <printf@plt>
   0x080484de <+21>:    mov    %ebp,%esp
   0x080484e0 <+23>:    mov    %esp,%ebp
   0x080484e2 <+25>:    sub    $0xc,%esp
   0x080484e5 <+28>:    and    $0xfffffff0,%esp
   0x080484e8 <+31>:    add    $0xc,%esp
   0x080484eb <+34>:    push   $0x9
   0x080484ed <+36>:    push   $0x8048465
   0x080484f2 <+41>:    push   $0x0
   0x080484f4 <+43>:    call   0x8048350 <read@plt>
   0x080484f9 <+48>:    mov    %ebp,%esp
   0x080484fb <+50>:    mov    %eax,%edx
   0x080484fd <+52>:    dec    %edx
   0x080484fe <+53>:    movb   $0x0,0x8048464(%eax)
   0x08048505 <+60>:    mov    $0x8048523,%esi
   0x0804850a <+65>:    mov    %esi,%edi
   0x0804850c <+67>:    mov    $0x12,%ecx
   0x08048511 <+72>:    lods   %ds:(%esi),%al
   0x08048512 <+73>:    xor    $0xaa,%al
   0x08048514 <+75>:    stos   %al,%es:(%edi)
   0x08048515 <+76>:    loop   0x8048511 <main+72>
=> 0x08048517 <+78>:    mov    $0x8048465,%esi
   0x0804851c <+83>:    mov    $0x804852d,%edi
   0x08048521 <+88>:    mov    %edx,%ecx
   0x08048523 <+90>:    lods   %ds:(%esi),%al
   0x08048524 <+91>:    dec    %al
   0x08048526 <+93>:    scas   %es:(%edi),%al
   0x08048527 <+94>:    jne    0x8048550 <main+135>
   0x08048529 <+96>:    loop   0x8048523 <main+90>
   0x0804852b <+98>:    jmp    0x8048535 <main+108>
   0x0804852d <+100>:   ss
   0x0804852e <+101>:   aaa
   0x0804852f <+102>:   data16
   0x08048530 <+103>:   fs
   0x08048531 <+104>:   inc    %ecx
   0x08048532 <+105>:   xor    -0x77(%eax,%eax,1),%al
   0x08048536 <+109>:   in     $0x83,%eax
   0x08048538 <+111>:   in     (%dx),%al
   0x08048539 <+112>:   add    $0x83,%al
   0x0804853b <+114>:   in     $0xf0,%al
   0x0804853d <+116>:   add    $0x4,%esp
```

After loop

A new loop appears !

Go deeper

A new loop appears !
In ESI : 0x8048465
In EDI : 0x804852d
The loop takes each byte of ESI, subtracts by one, and compares with one byte of EDI (and shift ESI/EDI).
The first byte of ESI -1 is compared to the first byte of EDI,
The second byte of ESI -1 is compared to the second byte of EDI,...



Let's have a look to what is inside ESI and EDI !

```
(gdb) b *main+88
Breakpoint 1 at 0x8048521
(gdb) run
Enter password:
mypass
Breakpoint 1, 0x08048521 in main (
(gdb) x/s $esi
0x8048465 <frame_dummy+53>: ``mypass''
(gdb) x/s $edi
0x804852d <main+100>: ``67fdA2D'' )<++>
```

(x/s -> print value as a string)

So the loop takes each character of "mypass" minus one, and compares to "67fdA2D".
Let's take each character of "67fdA2D" plus one as password : 78geB3E

```
$ ./crackme
Enter password:
78geB3E
$
```

We have a shell ! Crackme pwned !

Bonus

```
0x08048505 <+60>:    mov    $0x8048523,%esi
0x0804850a <+65>:    mov    %esi,%edi
0x0804850c <+67>:    mov    $0x12,%ecx
0x08048511 <+72>:    lods   %ds:(%esi),%al
0x08048512 <+73>:    xor    $0xaa,%al
0x08048514 <+75>:    stos   %al,%es:(%edi)
0x08048515 <+76>:    loop   0x8048511 <main+72>
```

Put @ main +90 in ESI
EDI = ESI (= @main+90)

Xor byte to byte ESI
wuth 0xAA and put in EDI

Few words about the strange loop

ESI = @main+88

EDI = ESI

Then the loop takes each bytes of ESI, xor with 0xAA, and saved the results in EDI

So this loop xor the code from @main+88 (remark : and the loop is perfomed 12 times, because of the value of ECX...)