

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

```

public main
main proc near
mov     ebp, esp

loc_80484CB:
sub     esp, 4
and     esp, 0FFFFFF0h
add     esp, 4
push   offset format ; "Enter password:\n"
call   _printf
mov     esp, ebp
mov     ebp, esp
sub     esp, 0Ch
and     esp, 0FFFFFF0h
add     esp, 0Ch
push   0 ; nbytes
push   offset byte_8048465 ; buf
push   0 ; fd
call   _read
mov     esp, ebp
mov     edx, eax
dec     edx
mov     byte ptr [eax+8048464h], 0
mov     esi, offset loc_8048523
mov     edi, esi
mov     ecx, 12h

loc_8048511:
lodsb
xor     al, 0Ah
stosb
loop   loc_8048511

mov     esi, offset byte_8048465
mov     edi, (offset loc_804852C+1)
mov     ecx, edx

loc_8048523:
push   es
push   esp
bound  eax, [edi+ebx*8]
lea   ecx, [eax+52h]
inc   ecx

loc_804852C:
mov     ds:0EECC090Ch, al
loop   short loc_80484CB
main endp ; sp-analysis failed

```

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:

```

0x000484c9 <+0>:   mov     %esp,%ebp
0x000484cb <+2>:   sub     $0x4,%esp
0x000484ce <+5>:   and     $0xffffffff0,%esp
0x000484d1 <+8>:   add     $0x4,%esp
0x000484d4 <+11>:  push   $0x0048454
0x000484d9 <+16>:  call   0x0048360 <printf@plt>
0x000484de <+21>:  mov     %ebp,%esp
0x000484e0 <+23>:  mov     %esp,%ebp
0x000484e2 <+25>:  sub     $0xc,%esp
0x000484e5 <+28>:  and     $0xffffffff0,%esp
0x000484e8 <+31>:  add     $0xc,%esp
0x000484eb <+34>:  push   $0x9
0x000484ed <+36>:  push   $0x0048465
0x000484f2 <+41>:  push   $0x0
0x000484f4 <+43>:  call   0x0048350 <read@plt>
0x000484f9 <+48>:  mov     %ebp,%esp
0x000484fb <+50>:  mov     %eax,%edx
0x000484fd <+52>:  dec     %edx
0x000484fe <+53>:  movb   $0x0,0x0048464(%eax)
0x00048505 <+60>:  mov     $0x0048523,%esi
0x0004850a <+65>:  mov     %esi,%edi
0x0004850c <+67>:  mov     $0x12,%ecx
0x00048511 <+72>:  lods   %ds:(%esi),%al
0x00048512 <+73>:  xor    $0xaa,%al
0x00048514 <+75>:  stos   %al,%es:(%edi)
0x00048515 <+76>:  loop   0x0048511 <main+72>
0x00048517 <+78>:  mov     $0x0048465,%esi
0x0004851c <+83>:  mov     $0x004852d,%edi
0x00048521 <+88>:  mov     %edx,%ecx
0x00048523 <+90>:  push   %es
0x00048524 <+91>:  push   %esp
0x00048525 <+92>:  bound %eax,(%edi,%ebx,8)
0x00048528 <+95>:  lea   0x52(%eax),%ecx
0x0004852b <+98>:  lnc   %ecx
0x0004852c <+99>:  mov   %al,0xcecc9d9c
0x00048531 <+104>: jmp   0x00484cb <main+2>
0x00048533 <+106>: out   %al,(%dx)
0x00048534 <+107>: stos   %al,%es:(%edi)
0x00048535 <+108>: mov   %esp,%ebp
0x00048537 <+110>: sub   $0x4,%esp
0x0004853a <+113>: and   $0xffffffff0,%esp
0x0004853d <+116>: add   $0x4,%esp
0x00048540 <+119>: push   $0x004856b
0x00048545 <+124>: call  0x0048370 <system@plt>
0x0004854a <+129>: mov   %ebp,%esp
0x0004854c <+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:
0x00404c9 <+8>:   mov     %esp,%ebp
0x00404cb <+2>:   sub     $0x4,%esp
0x00404ce <+5>:   and     $0xffffffff,%esp
0x00404d1 <+8>:   add     $0x4,%esp
0x00404d4 <+11>:  push   $0x0040454
0x00404d9 <+16>:  call   0x0040360 <printf@plt>
0x00404de <+21>:  mov     %ebp,%esp
0x00404e0 <+23>:  mov     %esp,%ebp
0x00404e2 <+25>:  sub     $0xc,%esp
0x00404e5 <+28>:  and     $0xffffffff,%esp
0x00404e8 <+31>:  add     $0xc,%esp
0x00404eb <+34>:  push   $0x9
0x00404ed <+36>:  push   $0x0040465
0x00404f2 <+41>:  push   $0x0
0x00404f4 <+43>:  call   0x0040350 <read@plt>
0x00404f9 <+48>:  mov     %ebp,%esp
0x00404fb <+50>:  mov     %eax,%edx
0x00404fd <+52>:  dec     %edx
0x00404fe <+53>:  movb   $0x0,0x0040464(%eax)
0x0040505 <+60>:  mov     $0x0040523,%esi
0x004050a <+65>:  mov     %esi,%edi
0x004050c <+67>:  mov     $0x12,%ecx
0x0040511 <+72>:  lods   %ds:(%esi),%al
0x0040512 <+73>:  xor     $0xa,%al
0x0040514 <+75>:  stos   %al,%es:(%edi)
0x0040515 <+76>:  loop   0x0040511 <main+72>
0x0040517 <+78>:  mov     $0x0040465,%esi
0x004051c <+83>:  mov     $0x0040524,%edi
0x0040521 <+88>:  mov     %edx,%ecx
0x0040523 <+90>:  lods   %ds:(%esi),%al
0x0040524 <+91>:  dec     %al
0x0040526 <+93>:  scasd  %es:(%edi),%al
0x0040527 <+94>:  jne    0x0040550 <main+135>
0x0040529 <+96>:  loop   0x0040523 <main+90>
0x004052b <+98>:  jmp    0x0040535 <main+108>
0x004052d <+100>: ss
0x004052e <+101>: aaa
0x004052f <+102>: data16
0x0040530 <+103>: fs
0x0040531 <+104>: inc     %ecx
0x0040532 <+105>: xor     -0x77(%eax,%eax,1),%al
0x0040536 <+109>: in     $0x83,%eax
0x0040538 <+111>: in     (%dx),%al
0x0040539 <+112>: add     $0x83,%al
0x004053b <+114>: in     $0xf0,%al
0x004053d <+116>: add     $0x4,%esp
0x0040540 <+119>: push   $0x004050b
0x0040545 <+124>: call   0x0040370 <system@plt>
0x004054a <+129>: mov     %ebp,%esp
0x004054c <+131>: xor     %eax,%eax
0x004054e <+133>: jmp    0x004056a <main+161>
0x0040550 <+135>: mov     %esp,%ebp

```

The code has changed !

(gdb) disas main

Dump of assembler code for function main:

```

0x000484c9 <+0>: mov    %esp,%ebp
0x000484cb <+2>: sub    $0x4,%esp
0x000484ce <+5>: and    $0xffffffff0,%esp
0x000484d1 <+8>: add    $0x4,%esp
0x000484d4 <+11>: push  $0x0048454
0x000484d9 <+16>: call   0x0048360 <printf@plt>
0x000484de <+21>: mov    %ebp,%esp
0x000484e0 <+23>: mov    %esp,%ebp
0x000484e2 <+25>: sub    $0xc,%esp
0x000484e5 <+28>: and    $0xffffffff0,%esp
0x000484e8 <+31>: add    $0xc,%esp
0x000484eb <+34>: push  $0x9
0x000484ed <+36>: push  $0x0048465
0x000484f2 <+41>: push  $0x0
0x000484f4 <+43>: call   0x0048350 <read@plt>
0x000484f9 <+48>: mov    %ebp,%esp
0x000484fb <+50>: mov    %eax,%edx
0x000484fd <+52>: dec    %edx
0x000484fe <+53>: movb   $0x0,0x0048464(%eax)
0x00048505 <+60>: mov    $0x0048523,%esi
0x0004850a <+65>: mov    %esi,%edi
0x0004850c <+67>: mov    $0x12,%ecx
0x00048511 <+72>: lods   %ds:(%esi),%al
0x00048512 <+73>: xor    $0xaa,%al
0x00048514 <+75>: stos   %al,%es:(%edi)
0x00048515 <+76>: loop  0x0048511 <main+72>
0x00048517 <+78>: mov    $0x0048465,%esi
0x0004851c <+83>: mov    $0x004852d,%edi
0x00048521 <+88>: mov    %edx,%ecx
0x00048523 <+90>: push  %es
0x00048524 <+91>: push  %esp
0x00048525 <+92>: bound %eax,(%edi,%ebx,8)
0x00048528 <+95>: lea   0x52(%eax),%ecx
0x0004852b <+98>: inc   %ecx
0x0004852c <+99>: mov   %al,0xccc9d9c
0x00048531 <+104>: jmp   0x00484cb <main+2>
0x00048533 <+106>: out   %al,(%dx)
0x00048534 <+107>: stos  %al,%es:(%edi)
0x00048535 <+108>: mov   %esp,%ebp
0x00048537 <+110>: sub   $0x4,%esp
0x0004853a <+113>: and   $0xffffffff0,%esp
0x0004853d <+116>: add   $0x4,%esp
0x00048540 <+119>: push $0x004856b
0x00048545 <+124>: call  0x0048370 <system@plt>
0x0004854a <+129>: mov   %ebp,%esp
0x0004854c <+131>: xor   %eax,%eax

```

Before loop

Dump of assembler code for function main:

```

0x000484c9 <+0>: mov    %esp,%ebp
0x000484cb <+2>: sub    $0x4,%esp
0x000484ce <+5>: and    $0xffffffff0,%esp
0x000484d1 <+8>: add    $0x4,%esp
0x000484d4 <+11>: push  $0x0048454
0x000484d9 <+16>: call   0x0048360 <printf@plt>
0x000484de <+21>: mov    %ebp,%esp
0x000484e0 <+23>: mov    %esp,%ebp
0x000484e2 <+25>: sub    $0xc,%esp
0x000484e5 <+28>: and    $0xffffffff0,%esp
0x000484e8 <+31>: add    $0xc,%esp
0x000484eb <+34>: push  $0x9
0x000484ed <+36>: push  $0x0048465
0x000484f2 <+41>: push  $0x0
0x000484f4 <+43>: call   0x0048350 <read@plt>
0x000484f9 <+48>: mov    %ebp,%esp
0x000484fb <+50>: mov    %eax,%edx
0x000484fd <+52>: dec    %edx
0x000484fe <+53>: movb   $0x0,0x0048464(%eax)
0x00048505 <+60>: mov    $0x0048523,%esi
0x0004850a <+65>: mov    %esi,%edi
0x0004850c <+67>: mov    $0x12,%ecx
0x00048511 <+72>: lods   %ds:(%esi),%al
0x00048512 <+73>: xor    $0xaa,%al
0x00048514 <+75>: stos   %al,%es:(%edi)
0x00048515 <+76>: loop  0x0048511 <main+72>
0x00048517 <+78>: mov    $0x0048465,%esi
0x0004851c <+83>: mov    $0x004852d,%edi
0x00048521 <+88>: mov    %edx,%ecx
0x00048523 <+90>: lods  %ds:(%esi),%al
0x00048524 <+91>: dec   %al
0x00048526 <+93>: scas  %es:(%edi),%al
0x00048527 <+94>: jne   0x0048550 <main+135>
0x00048529 <+96>: loop  0x0048523 <main+98>
0x0004852b <+98>: jmp   0x0048535 <main+108>
0x0004852d <+100>: ss
0x0004852e <+101>: aaa
0x0004852f <+102>: data16
0x00048530 <+103>: fs
0x00048531 <+104>: inc   %ecx
0x00048532 <+105>: xor   -0x77(%eax,%eax,1),%al
0x00048536 <+109>: in   $0x83,%eax
0x00048538 <+111>: in   (%dx),%al
0x00048539 <+112>: add  $0x83,%al
0x0004853b <+114>: in   $0xf0,%al
0x0004853d <+116>: add  $0x4,%esp

```

After loop

=>

A new loop appears !

The code has changed !

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,...

```

=> 0x08048517 <+78>:  mov     $0x8048465,%esi
0x0804851c <+83>:  mov     $0x804852d,%edi
0x08048521 <+88>:  mov     %edi,%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>
  
```

Put one byte of ESI in AL
Sub AL
Compare AL with one byte of EDI

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

```

1  (gdb) b *main+88
2  Breakpoint 1 at 0x8048521
3  (gdb) run
4  Enter password:
5  mypass
6  Breakpoint 1, 0x08048521 in main (
7  (gdb) x/s $esi
8  0x8048465 <frame_dummy+53>: ‘‘mypass’’
9  (gdb) x/s $edi
10 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

```

1  $ ./crackme
2  Enter password:
3  78geB3E
4  $

```

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>
0x08048517 <+78>:      0x08048515 %esi

```

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

**Xor byte to byte ESI
with 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 performed 12 times, because of the value of ECX. ...)