[kEYGENING Zack's Crackme4Hard]
OR
[Good things turn into bad things at the end…]

by bLaCk-eye [Kanal23]

INTRO:

This target has been into the crypto keygenme's target for some time now and still it hadn't no solutions, so I decided to make a keygen for it. This is the story of a very disappointed cracker (me) when cracking a target that at first looks very original and interesting but in the end is way bellow the initial expectations. You'll see

TUTORIAL:

Get the target from www.cryptokg.cjb.net.

You see it's not packed and made with MSVC 6.

Dissasemble it with IDA and look though the string references and you'll see:

```
.data:0040C9D4 aMiraclNotIniti db 'MIRACL not initialised- no call to
mirsys()',0Ah,0
```

So it's clear it uses the miracle bignum library.

Now to identify the miracle calls we use a signature file for ida. If you have my last on tkm! Trial crackme you'll have there a signature file for miracle which works perfectly with this target, if not I provided the signature file in this package tutorial.

Apply the signature file (if you don't know how to do that read the tkm! Tutorial) and see that we have 60 idenfied functions.Now let's find the protection:

```
.text:00401250                 push    edi             ; lParam
.text:00401251                 push    12Bh            ; wParam
.text:00401256                 push    WM_GETTEXT      ; Msg
.text:00401258                 push    edx             ; hWnd
.text:00401259                 mov     [esp+108h+var_9C], ebp
.text:0040125D                 call    ds:SendMessageA
.text:00401263                 cmp     eax, 5
.text:00401266                 jge     short loc_40126F
.text:00401268                 mov     flag, 1
.text:0040126F
.text:0040126F loc_40126F:                             ; CODE XREF:
DialogFunc+246 j
.text:0040126F                 mov     eax, dword_40D474
.text:00401274                 push    ebp             ; lParam
.text:00401275                 push    12Bh            ; wParam
.text:0040127A                 push    WM_GETTEXT      ; Msg
.text:0040127C                 push    eax             ; hWnd
.text:0040127D                 call    ds:SendMessageA
.text:00401283                 cmp     eax, 0C8h
.text:00401288                 jge     short loc_401291
.text:0040128A                 mov     flag, 1
```

So the first call to sendmessage it's to get our name and the second one of course for the serial. The name must be >=5 chars and the serial >=0c8h (200). That's quite a long serial, we have something interesting maybe?

Now let's go on:

```
.text:00401291 loc_401291:                             ;
.text:00401291                 mov     al, [edi]
.text:00401293                 test    al, al
.text:00401295                 jz      short loc_4012A9
.text:00401297                 mov     ecx, edi
.text:00401299
.text:00401299 loc_401299:                             ;
.text:00401299                 movsx   edx, al
.text:0040129C                 mov     al, [ecx+1]
.text:0040129F                 xor     edx, 14h
.text:004012A2                 add     esi, edx
.text:004012A4                 inc     ecx
.text:004012A5                 test    al, al
.text:004012A7                 jnz     short loc_401299
.text:004012A9
.text:004012A9 loc_4012A9:                             ;
.text:004012A9                 add     esi, 783Eh
.text:004012AF                 push    esi
.text:004012B0                 push    offset aI       ; LPCSTR
.text:004012B5                 push    edi             ; LPSTR
.text:004012B6                 call    ds:wsprintfA
```

So the crackme makes a small sum from the name chars and converts it to decimal string.

```
.text:004012BC                 mov     ecx, ebx
.text:004012BE                 mov     esi, offset a54269873541268 ;
"54269873541268974123698753215985265487 2"...
.text:004012C3                 lea     edi, [esp+104h+var_84]
.text:004012CA                 push    0Ah
.text:004012CC                 rep movsd
.text:004012CE                 push    400h
.text:004012D3                 movsb
.text:004012D4                 call    _mirsys
.text:004012D9                 push    0
.text:004012DB                 mov     dword ptr [eax+238h], 0Ah
.text:004012E5                 mov     dword ptr [eax+240h], 1
.text:004012EF                 call    _mirvar
.text:004012F4                 push    0
.text:004012F6                 call    _mirvar
.text:004012FB                 push    0
.text:004012FD                 mov     [esp+118h+big_1], eax
.text:00401304                 call    _mirvar
.text:00401309                 push    0
.text:0040130B                 mov     [esp+11Ch+big_2], eax
.text:00401312                 call    _mirvar
.text:00401317                 push    0
.text:00401319                 mov     esi, eax
.text:0040131B                 call    _mirvar
.text:00401320                 push    0
.text:00401322                 mov     [esp+124h+big_3], eax
```

```
.text:00401329                call    _mirvar
.text:0040132E                push    0
.text:00401330                mov     edi, eax
.text:00401332                call    _mirvar
.text:00401337                push    0
.text:00401339                mov     ebx, eax
.text:0040133B                call    _mirvar
.text:00401340                push    0
.text:00401342                mov     ebp, eax
.text:00401344                call    _mirvar
```

Now we have a bignum copied to a buffer and allot of bignum initialized (9 to be precise).

```
.text:00401349                mov     ecx, [esp+130h+var_94]
.text:00401350                mov     [esp+130h+var_8C], eax
.text:00401357                push    ecx
.text:00401358                push    eax
.text:00401359                call    _cinstr
.text:0040135E                add     esp, 40h
.text:00401361                lea     edx, [esp+0F8h+var_84]
.text:00401365                push    edx
.text:00401366                push    ebp
.text:00401367                call    _cinstr
.text:0040136C                push    edi
.text:0040136D                push    0Ah
.text:0040136F                push    80h
.text:00401374                call    _bigdig
.text:00401379                push    edi
.text:0040137A                push    edi
.text:0040137B                call    _nxprime
.text:00401380                push    ebx
.text:00401381                push    0Ah
.text:00401383                push    80h
.text:00401388                call    _bigdig
.text:0040138D                push    ebx
.text:0040138E                push    ebx
.text:0040138F                call    _nxprime
.text:00401394                push    ebp
.text:00401395                push    ebp
.text:00401396                call    _nxprime
.text:0040139B                add     esp, 38h
.text:0040139E                push    esi
.text:0040139F                push    1
.text:004013A1                call    _mirvar
.text:004013A6                add     esp, 4
.text:004013A9                push    eax
.text:004013AA                push    edi
.text:004013AB                call    _subtract
.text:004013B0                mov     eax, [esp+104h+var_98]
.text:004013B4                add     esp, 0Ch
.text:004013B7                push    eax
.text:004013B8                push    1
.text:004013BA                call    _mirvar
.text:004013BF                add     esp, 4
.text:004013C2                push    eax
```

```
.text:004013C3                push    ebx
.text:004013C4                call    _subtract
.text:004013C9                mov     ecx, [esp+104h+var_98]
.text:004013CD                push    esi
.text:004013CE                push    ecx
.text:004013CF                push    esi
.text:004013D0                call    _multiply
.text:004013D5                mov     edx, [esp+110h+var_98]
.text:004013D9                mov     eax, [esp+110h+big_2]
.text:004013E0                push    edx
.text:004013E1                push    eax
.text:004013E2                push    eax
.text:004013E3                push    esi
.text:004013E4                push    ebp
.text:004013E5                call    _xgcd
.text:004013EA                push    esi
.text:004013EB                push    ebx
.text:004013EC                push    edi
.text:004013ED                call    _multiply
.text:004013F2                mov     edi, [esp+130h+big_3]
.text:004013F9                mov     eax, [esp+130h+var_8C]
.text:00401400                push    edi
.text:00401401                push    esi
.text:00401402                push    ebp
.text:00401403                push    eax
.text:00401404                call    _powmod
.text:00401409                mov     esi, [esp+140h+var_94]
.text:00401410                add     esp, 48h
.text:00401413                push    esi
.text:00401414                push    edi
.text:00401415                call    _cotstr
.text:0040141A                mov     edx, [esp+100h+var_9C]
.text:0040141E                add     esp, 8
.text:00401421                xor     ebp, ebp
.text:00401423                mov     ebx, esi
.text:00401425                sub     edx, esi
.text:00401427
.text:00401427 loc_401427:                            ; CODE XREF:
DialogFunc+429 j
.text:00401427                mov     cl, [ebx]
.text:00401429                mov     al, [edx+ebx]
.text:0040142C                cmp     cl, al
.text:0040142E                jz      short loc_401437
.text:00401430                mov     flag, 1
.text:00401437
.text:00401437 loc_401437:                            ; CODE XREF:
DialogFunc+40E j
.text:00401437                mov     edi, [esp+0F8h+var_9C]
.text:0040143B                inc     ebp
.text:0040143C                or      ecx, 0FFFFFFFFh
.text:0040143F                xor     eax, eax
.text:00401441                inc     ebx
.text:00401442                repne scasb
.text:00401444                not     ecx
.text:00401446                dec     ecx
.text:00401447                cmp     ebp, ecx
.text:00401449                jbe     short loc_401427
```

we have allot of interesting calls here :like nxprime,bigdig,xgcd (if you don t know read the miracle manual), but let s take a step at a time.

The sum obtained from the name is inputed as a bignum(let s call it big_sum), and so is the ascii bignum we talked about earlier (let s call it e).Then the crackme generates two bignums (let s call them p,q), finds out the next prime numbers after them and next finds out the next prime number after the ascii bignum.My first idea was that we are handling some runtime-RSA parameter generation(lolz ✍).Let

Then the program calculates p*q and (p-1)(q-1).I was very happy that my initial idea of the runtime rsa was true, but the fat lady didn t sing yet. After this it does an Extended Euclidian Algorithm to find some numbers that it doesn t use at all ✍ so the xgcd can be very well omitted.Next it calculates big_sum^e mod (p*q) and it outputs the result into base 10 and it compares it with our entered serial number. HUH????????

Yeap my exactly reaction:all this trouble to generate yourself the good serial?No rsa?No interesting stuff? My happy world collapsed (not really).So this is it. Now to make a keygen.

We need to:

1.Calculate the sum from the name (just rip the algo from crackme)

2.Find p and q (read down)

3.calculate big_sum^e mod (p*q)

The first problem I had was to find the factors of the modulus,p and q.But I remembered that the bigdig function generates random bignums from a seed that must be initialized by a call to irand, but the crackme doesn t call it.If you read the documentation of the miracle lib you ll see that a call to mirsys calls irand with the random seed as a 0.So if you have the same seed you have the same bignum and these are:

P=80FAA96126C399C7E83E89CA1B40CCB992A1BBA4D6C7CC2AA0B8B765630E241388E65E3FB7227E3098CB6D9E390B75E15C40193B0B
Q=13286317F7711D35FAEBEEF5BDC1308755996ADBBE50205343EDA5B71276B12E3C028E5307519CAB52AB00581D2E2856DEF9F53B1BF

And the encryption exponent is:

E=140AFBB597E2943B38CBBC5D80E26A81C152DA4BE47C1E163E1C633874D0CD2C3FC67457AF83AEC05956D45A378FE63591F3B8B635B

That s all you need to do.

I told you bad things are reserved for the end ✍

Cya into my next tutorial.

Regards bLaCk-eye@2004

Comments and suggestions:

mycherynos@yahoo.com

Greetz:
KANAL23 members
TKM! Members
#Reversing4Elites dudes
#Compression dudes (especially Jibz)
And you for reading this ✍