ARTeam
the international bank of knowledge

2009

A Tales of Reversing & Keygenning Two MD5 Registration Schemas

Dead Regeneration developer Cranky Pants Games THQ
© GameWallpapers.com hosted by JTLnet

Author: 2kAD, Nieylana

Graphic Editor: Shub-Nigurrath

ARTeam

February 2009

DISCLAIMER

All code included with this tutorial is free to use and modify; we only ask that you mention where you found it. This tutorial is also free to distribute in its current unaltered form, with all the included supplements.

All the commercial programs used within this tutorial have been used only for the purpose of demonstrating the theories and methods described. No distribution of patched applications has been done under any media or host. The applications used were most of the times already been patched by other fellows, and cracked versions were available since a lot of time. ARTeam or the authors of the papers shouldn't be considered responsible for damages to the companies holding rights on those programs. The scope of this document as well as any other ARTeam tutorial is of sharing knowledge and teaching how to patch applications, how to bypass protections and generally speaking how to improve the RCE art. We are not releasing any cracked application.

VERIFICATION

ARTeam.esfv can be opened in the ARTeamESFVChecker to verify all files have been released by ARTeam and are unaltered. The ARTeamESFVChecker can be obtained in the release section of the ARTeam site: http://releases.accessroot.com

ARTEAM QUALITY 100% GUARANTEED

TABLE OF CONTENTS

1 KEYGENNING MD5 – 2KAD

1.1 FOREWORDS

I have been on the computer scene for a long time… Since 1984 to be exact, and have been reversing or cracking (so it was called back then) since that time. I have seen many protections come and go, and I have unpacked pretty much every protection known to man. However, keygenning is new to me…. Or serious keygenning is. In this tutorial I will show you how I reversed what I will describe as my first serious keygen. The tutorial is not a complete walk through, because it would take months to write. It's written so that reversers with some knowledge should be able to see how I figured out this protection. If you are a complete newbie it's not a tutorial for you yet.

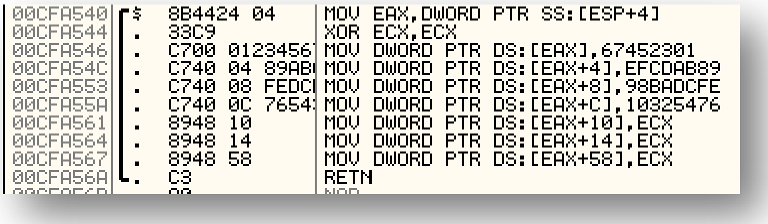
1.1.1 TOOLS USED:

A PC, Olly debugger v1.10, a working brain.

1.2 BEFORE WE START…

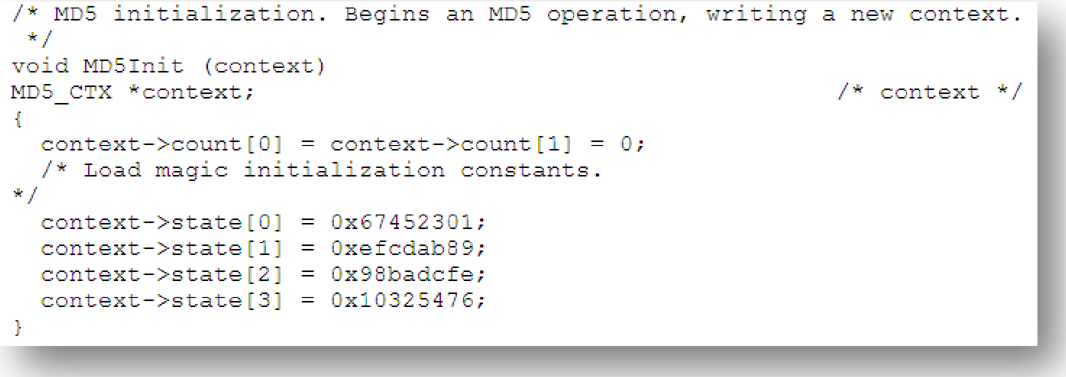
In the following tutorial you will see my comments in the disassembly, that the target uses MD5 hashing. But how do I know ? Well, of course one will have to trace the target and eventually you will stumble upon certain code snippets that for sure will give you the answer.

Here is the first clue:



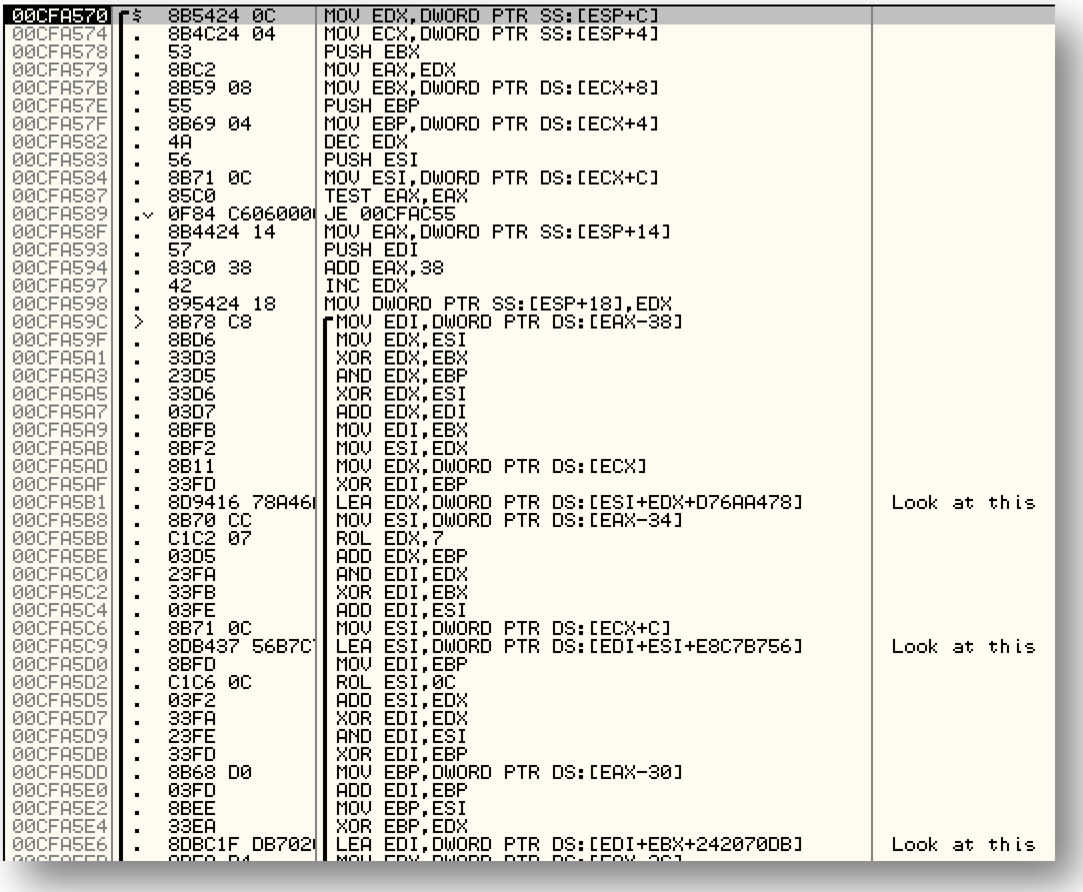
```
00CFA540  $  8B4424 04      MOV EAX,DWORD PTR SS:[ESP+4]
00CFA544  .  33C9           XOR ECX,ECX
00CFA546  .  C700 0123456   MOV DWORD PTR DS:[EAX],67452301
00CFA54C  .  C740 04 89AB   MOV DWORD PTR DS:[EAX+4],EFCDAB89
00CFA553  .  C740 08 FEDC   MOV DWORD PTR DS:[EAX+8],98BADCFE
00CFA55A  .  C740 0C 7654   MOV DWORD PTR DS:[EAX+C],10325476
00CFA561  .  8948 10        MOV DWORD PTR DS:[EAX+10],ECX
00CFA564  .  8948 14        MOV DWORD PTR DS:[EAX+14],ECX
00CFA567  .  8948 58        MOV DWORD PTR DS:[EAX+58],ECX
00CFA56A  .  C3             RETN
```

This taken from RFC 1351 (http://tools.ietf.org/html/rfc1321):



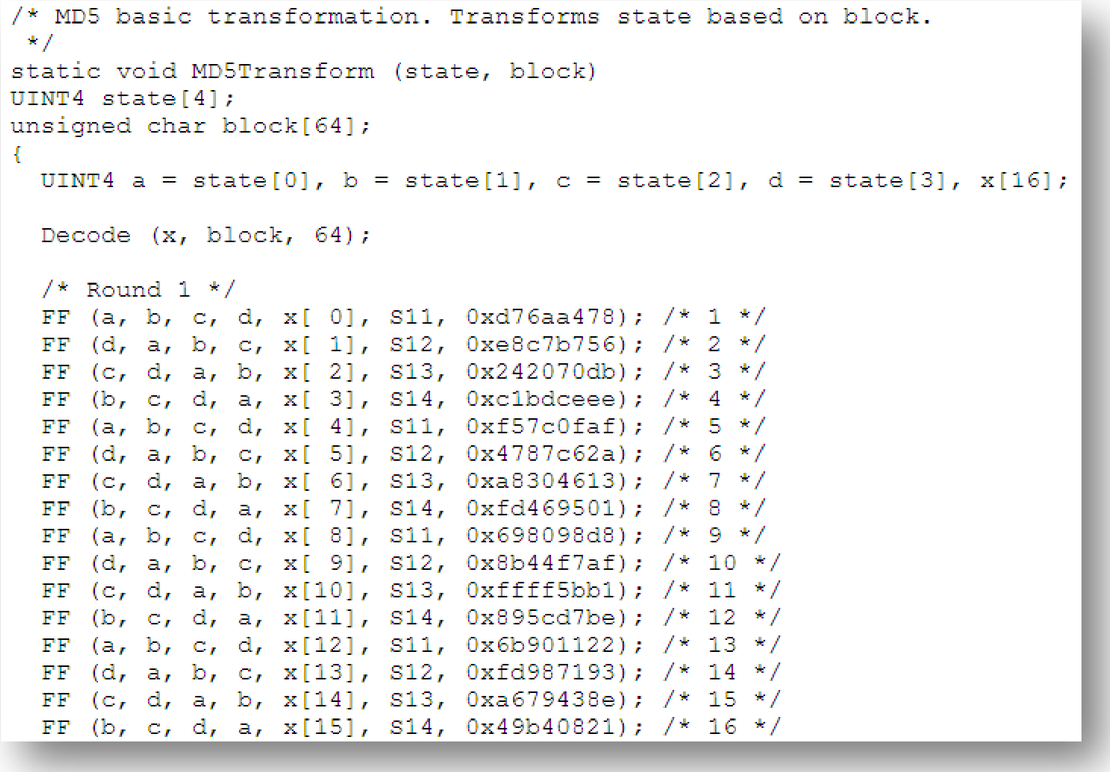
```
/* MD5 initialization. Begins an MD5 operation, writing a new context.
 */
void MD5Init (context)
MD5_CTX *context;                                          /* context */
{
  context->count[0] = context->count[1] = 0;
  /* Load magic initialization constants.
*/
  context->state[0] = 0x67452301;
  context->state[1] = 0xefcdab89;
  context->state[2] = 0x98badcfe;
  context->state[3] = 0x10325476;
}
```

Here is the final proof:



```
00CFA570  $  8B5424 0C     MOV EDX,DWORD PTR SS:[ESP+C]
00CFA574  .  8B4C24 04     MOV ECX,DWORD PTR SS:[ESP+4]
00CFA578  .  53            PUSH EBX
00CFA579  .  8BC2          MOV EAX,EDX
00CFA57B  .  8B59 08       MOV EBX,DWORD PTR DS:[ECX+8]
00CFA57E  .  55            PUSH EBP
00CFA57F  .  8B69 04       MOV EBP,DWORD PTR DS:[ECX+4]
00CFA582  .  4A            DEC EDX
00CFA583  .  56            PUSH ESI
00CFA584  .  8B71 0C       MOV ESI,DWORD PTR DS:[ECX+C]
00CFA587  .  85C0          TEST EAX,EAX
00CFA589  .v 0F84 C606000  JE 00CFAC55
00CFA58F  .  8B4424 14     MOV EAX,DWORD PTR SS:[ESP+14]
00CFA593  .  57            PUSH EDI
00CFA594  .  83C0 38       ADD EAX,38
00CFA597  .  42            INC EDX
00CFA598  .  895424 18     MOV DWORD PTR SS:[ESP+18],EDX
00CFA59C  >  8B78 C8       MOV EDI,DWORD PTR DS:[EAX-38]
00CFA59F  .  8BD6           MOV EDX,ESI
00CFA5A1  .  33D3           XOR EDX,EBX
00CFA5A3  .  23D5           AND EDX,EBP
00CFA5A5  .  33D6           XOR EDX,ESI
00CFA5A7  .  03D7           ADD EDX,EDI
00CFA5A9  .  8BFB           MOV EDI,EBX
00CFA5AB  .  8BF2           MOV ESI,EDX
00CFA5AD  .  8B11           MOV EDX,DWORD PTR DS:[ECX]
00CFA5AF  .  33FD           XOR EDI,EBP
00CFA5B1  .  8D9416 78A46   LEA EDX,DWORD PTR DS:[ESI+EDX+D76AA478]   Look at this
00CFA5B8  .  8B70 CC        MOV ESI,DWORD PTR DS:[EAX-34]
00CFA5BB  .  C1C2 07        ROL EDX,7
00CFA5BE  .  03D5           ADD EDX,EBP
00CFA5C0  .  23FA           AND EDI,EDX
00CFA5C2  .  33FB           XOR EDI,EBX
00CFA5C4  .  03FE           ADD EDI,ESI
00CFA5C6  .  8B71 0C        MOV ESI,DWORD PTR DS:[ECX+C]
00CFA5C9  .  8DB437 56B7C   LEA ESI,DWORD PTR DS:[EDI+ESI+E8C7B756]   Look at this
00CFA5D0  .  8BFD           MOV EDI,EBP
00CFA5D2  .  C1C6 0C        ROL ESI,0C
00CFA5D5  .  03F2           ADD ESI,EDX
00CFA5D7  .  33FA           XOR EDI,EDX
00CFA5D9  .  23FE           AND EDI,ESI
00CFA5DB  .  33FD           XOR EDI,EBP
00CFA5DD  .  8B68 D0        MOV EBP,DWORD PTR DS:[EAX-30]
00CFA5E0  .  03FD           ADD EDI,EBP
00CFA5E2  .  8BEE           MOV EBP,ESI
00CFA5E4  .  33EA           XOR EBP,EDX
00CFA5E6  .  8DBC1F DB702   LEA EDI,DWORD PTR DS:[EDI+EBX+242070DB]   Look at this
```

Again this taken from RFC 1351:

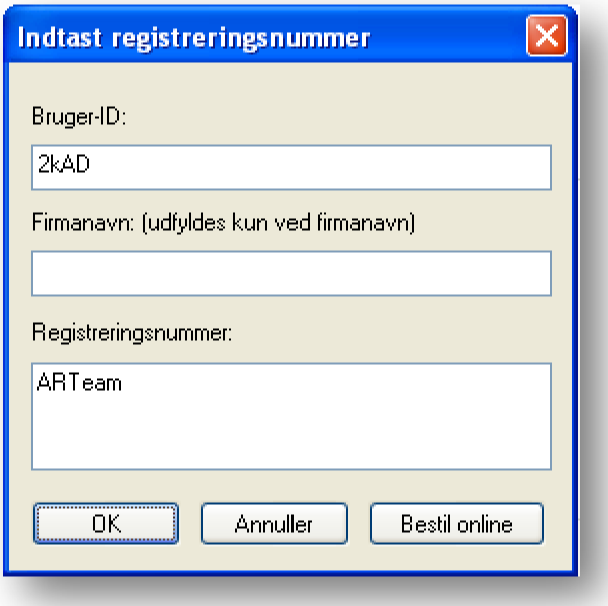


```
/* MD5 basic transformation. Transforms state based on block.
 */
static void MD5Transform (state, block)
UINT4 state[4];
unsigned char block[64];
{
  UINT4 a = state[0], b = state[1], c = state[2], d = state[3], x[16];

  Decode (x, block, 64);

  /* Round 1 */
  FF (a, b, c, d, x[ 0], S11, 0xd76aa478); /* 1 */
  FF (d, a, b, c, x[ 1], S12, 0xe8c7b756); /* 2 */
  FF (c, d, a, b, x[ 2], S13, 0x242070db); /* 3 */
  FF (b, c, d, a, x[ 3], S14, 0xc1bdceee); /* 4 */
  FF (a, b, c, d, x[ 4], S11, 0xf57c0faf); /* 5 */
  FF (d, a, b, c, x[ 5], S12, 0x4787c62a); /* 6 */
  FF (c, d, a, b, x[ 6], S13, 0xa8304613); /* 7 */
  FF (b, c, d, a, x[ 7], S14, 0xfd469501); /* 8 */
  FF (a, b, c, d, x[ 8], S11, 0x698098d8); /* 9 */
  FF (d, a, b, c, x[ 9], S12, 0x8b44f7af); /* 10 */
  FF (c, d, a, b, x[10], S13, 0xffff5bb1); /* 11 */
  FF (b, c, d, a, x[11], S14, 0x895cd7be); /* 12 */
  FF (a, b, c, d, x[12], S11, 0x6b901122); /* 13 */
  FF (d, a, b, c, x[13], S12, 0xfd987193); /* 14 */
  FF (c, d, a, b, x[14], S13, 0xa679438e); /* 15 */
  FF (b, c, d, a, x[15], S14, 0x49b40821); /* 16 */
```

It's evident that we're in the first round of MD5. Also you can use IDA findcrypt plugin. This will identify it for you. There are other tools out there that will do the same job, but being able to identify MD5 yourself will save you time.

1.3 STARTING UP AND GETTING READY

Execute “drag2pdf.exe” and go to the registration dialog. Below you will see what I typed in. Don’t mind the strange language displayed in the picture, it’s just my native language.



Indtast registreringsnummer

Bruger-ID: 2kAD

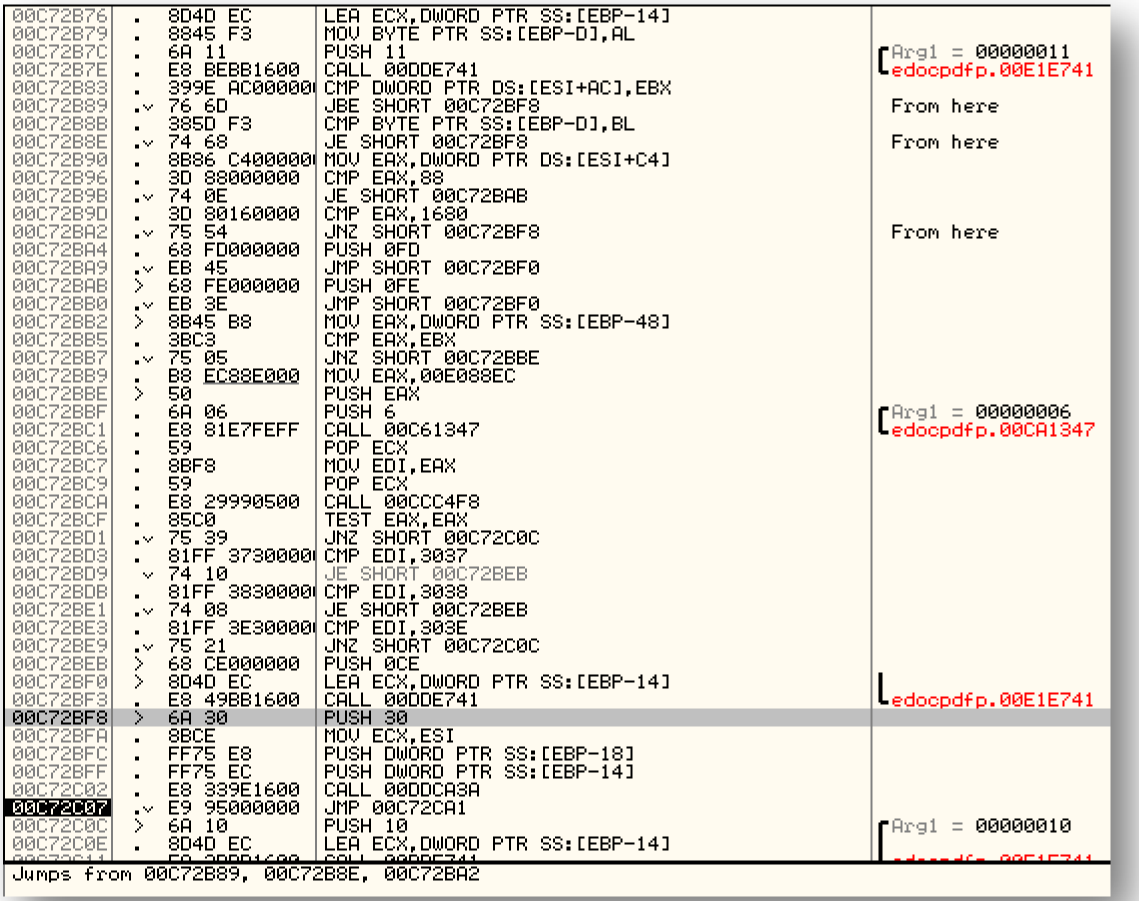
Firmanavn: (udfyldes kun ved firmanavn)

Registreringsnummer: ARTeam

Once you typed in a name and serial, press OK.

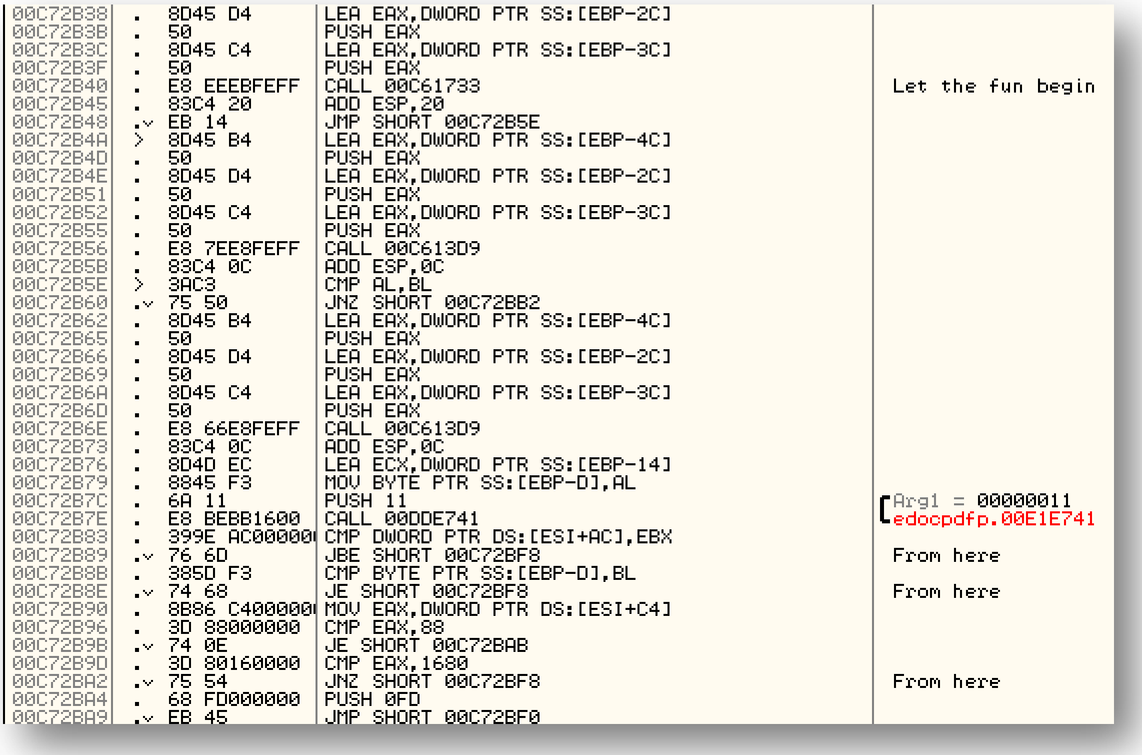
You will get a messagebox saying the that serial is not valid. So I decided to set breakpoints on ***MessageBoxA***, ***MessageBoxExA***, ***MessageboxExW*** and run it again.

We trap it at ***MessageboxExW***. Back trace until you end up in a code-section.



```
00C72B76  .   8D4D EC        LEA ECX,DWORD PTR SS:[EBP-14]
00C72B79  .   8845 F3        MOV BYTE PTR SS:[EBP-D],AL
00C72B7C  .   6A 11          PUSH 11                                 [Arg1 = 00000011
00C72B7E  .   E8 BEBB1600    CALL 00DDE741                           [edocpdfp.00E1E741
00C72B83  .   399E AC00000   CMP DWORD PTR DS:[ESI+AC],EBX
00C72B89  .v  76 6D          JBE SHORT 00C72BF8                       From here
00C72B8B  .   385D F3        CMP BYTE PTR SS:[EBP-D],BL
00C72B8E  .v  74 68          JE SHORT 00C72BF8                        From here
00C72B90  .   8B86 C400000   MOV EAX,DWORD PTR DS:[ESI+C4]
00C72B96  .   3D 88000000    CMP EAX,88
00C72B9B  .v  74 0E          JE SHORT 00C72BAB
00C72B9D  .   3D 80160000    CMP EAX,1680
00C72BA2  .v  75 54          JNZ SHORT 00C72BF8                       From here
00C72BA4  .   68 FD000000    PUSH 0FD
00C72BA9  .v  EB 45          JMP SHORT 00C72BF0
00C72BAB  >   68 FE000000    PUSH 0FE
00C72BB0  .v  EB 3E          JMP SHORT 00C72BF0
00C72BB2  >   8B45 B8        MOV EAX,DWORD PTR SS:[EBP-48]
00C72BB5  .   3BC3           CMP EAX,EBX
00C72BB7  .v  75 05          JNZ SHORT 00C72BBE
00C72BB9  .   B8 EC88E000    MOV EAX,00E088EC
00C72BBE  >   50             PUSH EAX
00C72BBF  .   6A 06          PUSH 6                                  [Arg1 = 00000006
00C72BC1  .   E8 81E7FEFF    CALL 00C61347                           [edocpdfp.00CA1347
00C72BC6  .   59             POP ECX
00C72BC7  .   8BF8           MOV EDI,EAX
00C72BC9  .   59             POP ECX
00C72BCA  .   E8 29990500    CALL 00CCC4F8
00C72BCF  .   85C0           TEST EAX,EAX
00C72BD1  .v  75 39          JNZ SHORT 00C72C0C
00C72BD3  .   81FF 3730000   CMP EDI,3037
00C72BD9   v  74 10          JE SHORT 00C72BEB
00C72BDB  .   81FF 3830000   CMP EDI,3038
00C72BE1  .v  74 08          JE SHORT 00C72BEB
00C72BE3  .   81FF 3E30000   CMP EDI,303E
00C72BE9  .v  75 21          JNZ SHORT 00C72C0C
00C72BEB  >   68 CE000000    PUSH 0CE
00C72BF0  >   8D4D EC        LEA ECX,DWORD PTR SS:[EBP-14]           [
00C72BF3  .   E8 49BB1600    CALL 00DDE741                           [edocpdfp.00E1E741
00C72BF8  >   6A 30          PUSH 30
00C72BFA  .   8BCE           MOV ECX,ESI
00C72BFC  .   FF75 E8        PUSH DWORD PTR SS:[EBP-18]
00C72BFF  .   FF75 EC        PUSH DWORD PTR SS:[EBP-14]
00C72C02  .   E8 339E1600    CALL 00DDCA3A
00C72C07  .v  E9 95000000    JMP 00C72CA1
00C72C0C  >   6A 10          PUSH 10                                 [Arg1 = 00000010
00C72C0E  .   8D4D EC        LEA ECX,DWORD PTR SS:[EBP-14]           [
Jumps from 00C72B89, 00C72B8E, 00C72BA2
```

Notice that the BAD BOY call originates from either 3 positions. So lets back trace a little until we hit gold.

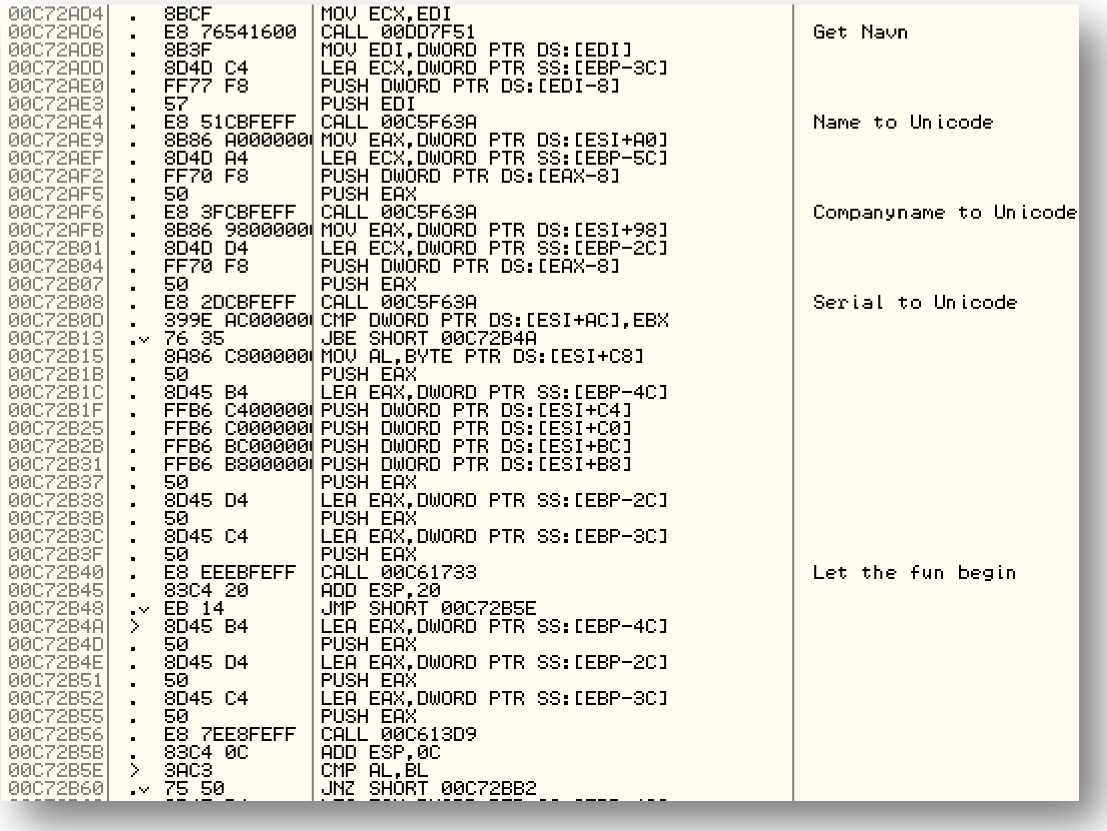


```
00C72B38  .   8D45 D4       LEA EAX,DWORD PTR SS:[EBP-2C]
00C72B3B  .   50            PUSH EAX
00C72B3C  .   8D45 C4       LEA EAX,DWORD PTR SS:[EBP-3C]
00C72B3F  .   50            PUSH EAX
00C72B40  .   E8 EEEBFEFF   CALL 00C61733                          Let the fun begin
00C72B45  .   83C4 20       ADD ESP,20
00C72B48  .v  EB 14         JMP SHORT 00C72B5E
00C72B4A  >   8D45 B4       LEA EAX,DWORD PTR SS:[EBP-4C]
00C72B4D  .   50            PUSH EAX
00C72B4E  .   8D45 D4       LEA EAX,DWORD PTR SS:[EBP-2C]
00C72B51  .   50            PUSH EAX
00C72B52  .   8D45 C4       LEA EAX,DWORD PTR SS:[EBP-3C]
00C72B55  .   50            PUSH EAX
00C72B56  .   E8 7EE8FEFF   CALL 00C613D9
00C72B5B  .   83C4 0C       ADD ESP,0C
00C72B5E  >   3AC3          CMP AL,BL
00C72B60  .v  75 50         JNZ SHORT 00C72BB2
00C72B62  .   8D45 B4       LEA EAX,DWORD PTR SS:[EBP-4C]
00C72B65  .   50            PUSH EAX
00C72B66  .   8D45 D4       LEA EAX,DWORD PTR SS:[EBP-2C]
00C72B69  .   50            PUSH EAX
00C72B6A  .   8D45 C4       LEA EAX,DWORD PTR SS:[EBP-3C]
00C72B6D  .   50            PUSH EAX
00C72B6E  .   E8 66E8FEFF   CALL 00C613D9
00C72B73  .   83C4 0C       ADD ESP,0C
00C72B76  .   8D4D EC       LEA ECX,DWORD PTR SS:[EBP-14]
00C72B79  .   8845 F3       MOV BYTE PTR SS:[EBP-D],AL
00C72B7C  .   6A 11         PUSH 11                                [Arg1 = 00000011
00C72B7E  .   E8 BEBB1600   CALL 00DDE741                          [edocpdfp.00E1E741
00C72B83  .   399E AC00000  CMP DWORD PTR DS:[ESI+AC],EBX
00C72B89  .v  76 6D         JBE SHORT 00C72BF8                     From here
00C72B8B  .   385D F3       CMP BYTE PTR SS:[EBP-D],BL
00C72B8E  .v  74 68         JE SHORT 00C72BF8                      From here
00C72B90  .   8B86 C400000  MOV EAX,DWORD PTR DS:[ESI+C4]
00C72B96  .   3D 88000000   CMP EAX,88
00C72B9B  .v  74 0E         JE SHORT 00C72BAB
00C72B9D  .   3D 80160000   CMP EAX,1680
00C72BA2  .v  75 54         JNZ SHORT 00C72BF8                     From here
00C72BA4  .   68 FD000000   PUSH 0FD
00C72BA9  .v  EB 45         JMP SHORT 00C72BF0
```

The call at 0C72B40 is our wonder call. So let go to the beginning of this procedure and find out what happens. Do not trace into the call at 0C72B40 yet.

1.4 ANALYZING THE ALGORITHM

Once we find out where we are directed we end up here:

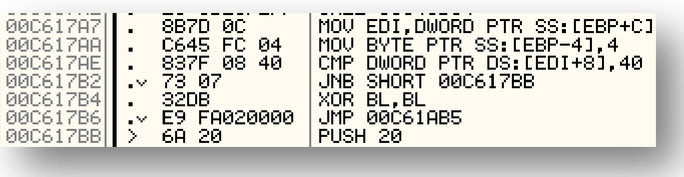


```
00C72AD4  .   8BCF           MOV ECX,EDI
00C72AD6  .   E8 76541600    CALL 00DD7F51                          Get Navn
00C72ADB  .   8B3F           MOV EDI,DWORD PTR DS:[EDI]
00C72ADD  .   8D4D C4        LEA ECX,DWORD PTR SS:[EBP-3C]
00C72AE0  .   FF77 F8        PUSH DWORD PTR DS:[EDI-8]
00C72AE3  .   57             PUSH EDI
00C72AE4  .   E8 51CBFEFF    CALL 00C5F63A                          Name to Unicode
00C72AE9  .   8B86 A000000   MOV EAX,DWORD PTR DS:[ESI+A0]
00C72AEF  .   8D4D A4        LEA ECX,DWORD PTR SS:[EBP-5C]
00C72AF2  .   FF70 F8        PUSH DWORD PTR DS:[EAX-8]
00C72AF5  .   50             PUSH EAX
00C72AF6  .   E8 3FCBFEFF    CALL 00C5F63A                          Companyname to Unicode
00C72AFB  .   8B86 9800000   MOV EAX,DWORD PTR DS:[ESI+98]
00C72B01  .   8D4D D4        LEA ECX,DWORD PTR SS:[EBP-2C]
00C72B04  .   FF70 F8        PUSH DWORD PTR DS:[EAX-8]
00C72B07  .   50             PUSH EAX
00C72B08  .   E8 2DCBFEFF    CALL 00C5F63A                          Serial to Unicode
00C72B0D  .   399E AC00000   CMP DWORD PTR DS:[ESI+AC],EBX
00C72B13  .v  76 35          JBE SHORT 00C72B4A
00C72B15  .   8A86 C800000   MOV AL,BYTE PTR DS:[ESI+C8]
00C72B1B  .   50             PUSH EAX
00C72B1C  .   8D45 B4        LEA EAX,DWORD PTR SS:[EBP-4C]
00C72B1F  .   FFB6 C400000   PUSH DWORD PTR DS:[ESI+C4]
00C72B25  .   FFB6 C000000   PUSH DWORD PTR DS:[ESI+C0]
00C72B2B  .   FFB6 BC00000   PUSH DWORD PTR DS:[ESI+BC]
00C72B31  .   FFB6 B800000   PUSH DWORD PTR DS:[ESI+B8]
00C72B37  .   50             PUSH EAX
00C72B38  .   8D45 D4        LEA EAX,DWORD PTR SS:[EBP-2C]
00C72B3B  .   50             PUSH EAX
00C72B3C  .   8D45 C4        LEA EAX,DWORD PTR SS:[EBP-3C]
00C72B3F  .   50             PUSH EAX
00C72B40  .   E8 EEEBFEFF    CALL 00C61733                          Let the fun begin
00C72B45  .   83C4 20        ADD ESP,20
00C72B48  .v  EB 14          JMP SHORT 00C72B5E
00C72B4A  >   8D45 B4        LEA EAX,DWORD PTR SS:[EBP-4C]
00C72B4D  .   50             PUSH EAX
00C72B4E  .   8D45 D4        LEA EAX,DWORD PTR SS:[EBP-2C]
00C72B51  .   50             PUSH EAX
00C72B52  .   8D45 C4        LEA EAX,DWORD PTR SS:[EBP-3C]
00C72B55  .   50             PUSH EAX
00C72B56  .   E8 7EE8FEFF    CALL 00C613D9
00C72B5B  .   83C4 0C        ADD ESP,0C
00C72B5E  >   3AC3           CMP AL,BL
00C72B60  .v  75 50          JNZ SHORT 00C72BB2
```

Some of the comments are just my own comments added while I debugged the target. I will not go into details. You can trace them yourself.

The interesting part is the CALL at 0C72B40 and the CMP at 0C72B5E. If you trace over the call at 0C72B40 you will see that the call returns FALSE. This returned BOOL is then checked at 0C72B5E. So…. The call at 0C72B40 has to return TRUE in order to on.

Tracing into the call at 0C72B40 we first end up here:



```
00C617A7  .   8B7D 0C        MOV EDI,DWORD PTR SS:[EBP+C]
00C617AA  .   C645 FC 04     MOV BYTE PTR SS:[EBP-4],4
00C617AE  .   837F 08 40     CMP DWORD PTR DS:[EDI+8],40
00C617B2  .v  73 07          JNB SHORT 00C617BB
00C617B4  .   32DB           XOR BL,BL
00C617B6  .v  E9 FA020000    JMP 00C61AB5
00C617BB  >   6A 20          PUSH 20
```

Take a look at 0C617AE. This compares the length of our name-string with 40h (64 bytes). So we can conclude that our name-string has to be at least 64 bytes long. Set a breakpoint at 0C617AE and press F9. Reenter a 64 byte long name-string. I recommend you type in