

Writeup til:

## Spilledrengen - Lovens lange ARM

fra:

**NC3CTF2022 / nc3ctf.dk / Nationalt Cyber Crime Center (NC3) / National  
enhed for Særlig Kriminalitet (NSK) / Dansk Politi / politi.dk**

af:

**mr.loo & THeWiZRD**

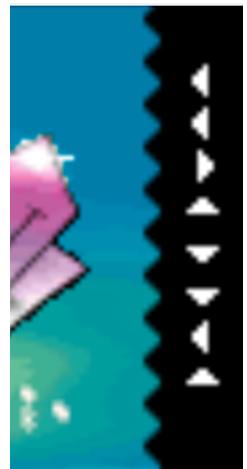


## **Intro:**

En intro forklædt som en CTF challenge, eller omvendt. Til at starte med, kan man jo downloade en emulator til Game Boy Advance, og se det grafiske (og musiske) udfolde sig:



Hvis man nu prøver at trykke på nogle knapper, kan man se at der kommer nogle ikoner ude i siden, alt efter hvilke knapper, der trykkes på.



## Disassembler / Decompiler

På trods af at Ida har været vores ynglingsværktøj i mange år, kan vi ikke komme udenom Ghidra, der med sin gratis pris, er noget af en julegave i disse tider.

Ghidra gennemgår tilmed en ret vild udvikling i disse år, og det kan derfor betale sig at hente den nyeste version (p.t. 10.2.2). Denne nye version kan bl.a. lave "string references" korrekt i denne disassembly. Ja, det bliver næsten for nemt i forhold til tidligere versioner :)

Vælg ARM little-endian og 4t. Load til standard ROM adresse for GBA (hint: 0x8000000).

Vi kan som det første se nogle tekst-strenge, der hinter helt vildt, om

Defined Strings – 19 items			
Location	String Value	String Representa...	Data Type
080000a0	Spilledrenge	"Spilledrenge"	ds
080147b0		" "	ds
080147c0	/      /      /      /      ...	"      /      /... ds	
08014800	/      / / /      /	"       /      /... ds	
080148f0	/      /   /	"       /  ... ds	
08014b78	8      / /   /      /	"8      / /   /... ds	
08015ce7	beeeeeel	"beeeeeel"	ds
08015f5f	HNNNNOOONNNNOOOOOO	"HNNNNOOONN... ds	
0801e33c	Okay nisse, lad os evaluere dit input ...	"Okay nisse, lad o... ds	
0801e36c	For kort	"For kort"	ds
0801e380	OK ... Decrypter:	"OK ... Decrypter:"	ds
0801e6d8	0123456789abcdef	"0123456789abc... ds	
0801e6e9	0123456789ABCDEF	"0123456789ABC... ds	
0801e704	0000000000000000	"0000000000000... ds	
0801e742	r: http://rebrand.ly/vlxxye eller https:/...	"r: http://rebrand... ds	
0801e7d4		" ... ds	
0801eaf0	, Creditz: Kode, grafik & musik: mr1oo   ...	", Creditz: Kode, g... ds	
0801ec34	0000000000000000	NC3{}	"0000000000000... ds
08020570	SZ ER UI TILBAGE MED NCQCTF I...	" SZ ER UI... ds	

Filter:

Gf Decompile: UndefinedFu... × 0101 DAT Defined Strings ×

hvor vi skal hen:

Den sidste tekst, ved 0x08020570, ligner den scroll-tekst, der kører derudaf. Her kommer den forbedrede mulighed for string references virkelig til nytte:

References to s__SZ_ER_UI_TILBAGE_MED_NCQCTF_I_Y_08020570 – 2 locations			
Location	Label	Code Unit	Context
08000144		ldr r1=>s__SZ_ER_UI_TILBAGE_MED_NCQCTF_I_Y_08020...	PARAM
080001d0	PTR_s__SZ_ER_UI_TILBAGE...	addr s__SZ_ER_UI_TILBAGE_MED_NCQCTF_I_Y_08020570	DATA

Vi kan nu finde koden, der bruger teksten. Interessant, men egentlig ikke den kode, vi er ude efter.

Der skal jo være et flag et sted, og mon ikke at de knapper ude i siden af skærmen, skal bruges? Her kunne vi altså begynde at kigge efter kode, der bruger keypad.

Men vi har været endnu mere flinke, og kommet med ekstreme hints i andre strings:

0801e33c	Okay nisse, lad os evaluere dit input ...	ds
0801e36c	For kort	ds
0801e380	OK ... Decrypter:	ds

Hurtigt kan vi derfor finde den relevante funktion:

```
08020260 c4 42 9f e5      ldr      r4,[DAT_0802052c]          = 080009D9h
08020264 c4 02 9f e5      ldr      r0=>s_Okay_nisse,_lad_os_evaluere_dit_i_0801e3  = "Okay nisse,
lad os evaluere d
                                                =
0801e33c
```

Et alternativ er at se efter 4-byte ARM kode, da den almindelige GBA kode kører 2-byte THUMB. Den relevante kode er nemlig compilet som ARM, netop for at hinte om hvilken kode, der er relevant at analysere.

Den decompilede funktion viser nært koden:

```
void UndefinedFunction_08020254(void)
{
    byte bVar1;
    byte *pbVar2;
    code *pcVar3;
    undefined *puVar4;
    byte *pbVar5;
    uint uVar6;
    char *pcVar7;
    uint uVar8;
    char *pcVar9;
    char *pcVar10;
    uint uVar11;
    uint uVar12;
    undefined auStack_30 [16];

    pcVar3 = DAT_0802052c;
    pbVar2 = DAT_08020528;
    (*DAT_0802052c)(PTR_s_Okay_nisse,_lad_os_evaluere_dit_i_08020530);
    (*DAT_08020534)(auStack_30,PTR_DAT_08020538,*pbVar2);
    (*pcVar3)(auStack_30);
    puVar4 = PTR_DAT_08020544;
    if (*pbVar2 < 0x11) {
        (*pcVar3)(PTR_s_For_kort_0802056c);
        return;
    }
    pcVar10 = DAT_0802053c + 0x11;
    pcVar7 = DAT_0802053c;
    pcVar9 = PTR_LAB_0801e392+1_08020540;
    do {
        pcVar7 = pcVar7 + 1;
        pcVar9 = pcVar9 + 1;
        if (*pcVar7 != *pcVar9) {
            (*pcVar3)(PTR_DAT_08020548);
            return;
        }
        (*pcVar3)(puVar4);
    } while (pcVar7 != pcVar10);
    (*pcVar3)(PTR_s_OK_..._Decrypter:_0802054c);
    pbVar5 = DAT_08020554;
    pbVar2 = DAT_08020550;
    if (((uint)DAT_08020554 | (uint)DAT_08020550) & 3) == 0) {
        uVar11 = DAT_08020558 ^
            CONCAT13(DAT_08020550[3],
            CONCAT12(DAT_08020550[2],CONCAT11(DAT_08020550[1],*DAT_08020550)));
        uVar8 = DAT_0802055c ^ *(uint*)(DAT_08020550 + 4);
        uVar6 = DAT_08020560 ^ *(uint*)(DAT_08020550 + 0xc);
        uVar12 = DAT_08020564 ^ *(uint*)(DAT_08020550 + 8);
        DAT_08020554[1] = (byte)(uVar11 >> 8);
        *pbVar5 = (byte)uVar11;
        pbVar5[3] = (byte)(uVar11 >> 0x18);
        pbVar5[4] = (byte)uVar8;
        pbVar5[6] = (byte)(uVar8 >> 0x10);
        pbVar5[7] = (byte)(uVar8 >> 0x18);
        pbVar5[8] = (byte)uVar12;
        pbVar5[2] = (byte)(uVar11 >> 0x10);
        pbVar5[5] = (byte)(uVar8 >> 8);
        pbVar5[9] = (byte)(uVar12 >> 8);
        pbVar5[10] = (byte)(uVar12 >> 0x10);
        pbVar5[0xb] = (byte)(uVar12 >> 0x18);
        pbVar5[0xc] = (byte)uVar6;
        pbVar5[0xd] = (byte)(uVar6 >> 8);
        pbVar5[0xe] = (byte)(uVar6 >> 0x10);
        pbVar5[0xf] = (byte)(uVar6 >> 0x18);
        bVar1 = pbVar2[0x10];
    }
    else {
        bVar1 = DAT_08020550[1];
        *DAT_08020554 = *DAT_08020550 ^ 0x4f;
        pbVar5[1] = bVar1 ^ 0x41;
        bVar1 = pbVar2[3];
        pbVar5[2] = pbVar2[2] ^ 0x52;
        pbVar5[3] = bVar1 ^ 0x52;
        bVar1 = pbVar2[5];
        pbVar5[4] = pbVar2[4] ^ 0x4b;
        pbVar5[5] = bVar1 ^ 0x57;
    }
}
```

```

bVar1 = pbVar2[7];
pbVar5[6] = pbVar2[6] ^ 0x4f;
pbVar5[7] = bVar1 ^ 0x41;
bVar1 = pbVar2[9];
pbVar5[8] = pbVar2[8] ^ 0x52;
pbVar5[9] = bVar1 ^ 0x4f;
bVar1 = pbVar2[0xb];
pbVar5[10] = pbVar2[10] ^ 0x47;
pbVar5[0xb] = bVar1 ^ 0x5a;
bVar1 = pbVar2[0xd];
pbVar5[0xc] = pbVar2[0xc] ^ 0x4d;
pbVar5[0xd] = bVar1 ^ 0x5c;
bVar1 = pbVar2[0xf];
pbVar5[0xe] = pbVar2[0xe] ^ 0x59;
pbVar5[0xf] = bVar1 ^ 0x51;
bVar1 = pbVar2[0x10];
}
pbVar5[0x10] = bVar1 ^ 0x53;
pbVar5[0x11] = 0;
(*pcVar3)(DAT_08020554);
(*DAT_08020568)();
return;
}

```

DAT_0801e394			XREF[1]:
080202cc(R)	0801e394 01	undefined1 01h	
			XREF[1]:
080202cc(R)	0801e395 02	undefined1 02h	
	0801e396 03	?? 03h	
	0801e397 04	?? 04h	
	0801e398 01	?? 01h	
	0801e399 02	?? 02h	
	0801e39a 03	?? 03h	
	0801e39b 04	?? 04h	
	0801e39c 01	?? 01h	
	0801e39d 01	?? 01h	
	0801e39e 02	?? 02h	
	0801e39f 02	?? 02h	
	0801e3a0 04	?? 04h	
	0801e3a1 04	?? 04h	
	0801e3a2 03	?? 03h	
	0801e3a3 03	?? 03h	
	0801e3a4 04	?? 04h	

Her ses så den faktiske kildekode:

```
void OnStartPressed()
{
    nocash_puts("Okay nisse, lad os evaluere dit input ...");

    char temp[16];
    sprintf(temp, "%i", g_currentUserInputIndex);
    nocash_puts(temp);

    if (g_currentUserInputIndex < C_INPUT_KEYS_MAX)
    {
        nocash_puts("For kort");
        return;
    }

    static const u8 correct_keys[] = {1, 2, 3, 4, 1, 2, 3, 4,     1, 1, 2, 2,
4, 4, 3, 3, 4};
    for (u8 i = 0; i < C_INPUT_KEYS_MAX; i++)
    {
        if (g_userInputKeys[i] == correct_keys[i])
        {
            nocash_puts("1");
        }
        else
        {
            nocash_puts("0");
            return;
        }
    }

    nocash_puts("OK ... Decrypter:");

    static const u8 g_encodedFlag[] = {0x4f, 0x41, 0x52, 0x52, 0x4b, 0x57,
0x4f, 0x41, 0x52, 0x4f, 0x47, 0x5a, 0x4d, 0x5c, 0x59, 0x51, 0x53, };

    for (u8 i = 0; i < C_INPUT_KEYS_MAX; i++)
    {
        u8 c1 = g_encodedFlag[i];
        u8 c2 = g_userInputKeys[i];           // Important not to use
correct_keys[i] here as that would allow one to simply skip to this code
directly
        u8 c = c1 ^ c2;

        g_scrollMessage[i] = c;
    }

    g_scrollMessage[C_INPUT_KEYS_MAX] = 0;
    nocash_puts(g_scrollMessage);

    InitScrollText();
}
```

I variablen "correct\_keys" ses den rigtige rækkefølge af tastetryk. Men hvilke taster svarer 1, 2, 3 og 4 til?

Når man ser koden her, bliver det klart at g\_userInputKeys kommer fra:

PTR\_DAT\_0802053c

Og denne bliver skrevet til fra:

References to DAT_030018cc - 8 locations			
Location	Label	Code Unit	Context
0800061c		ldr r1=>DAT_030018cc, [DAT_08000928]	PARAM
08000626		str r4, [r1,#0x0]=>DAT_030018cc	WRITE
08000630		str r1=>DAT_030018cc, [sp,#local_30]	DATA
0800083e		str r2, [r0,#0x0]=>DAT_030018cc	WRITE
0801ed32		strb r5, [r2,r3]=>DAT_030018cc	WRITE
0801ed84	PTR_DAT_0801ed84	addr DAT_030018cc	DATA
080202c8	LAB_080202c8	ldrb r2, [r5,#0x1]!=>DAT_030018cc	READ
0802030c		ldrb r0, [r3,#0x0]=>DAT_030018cc	READ

Dvs.:

```
*(ushort *) (puVar3 + iVar5) = *(ushort *) (puVar3 + iVar5) & (ushort) DAT_0801ed80;  
PTR_DAT_0801ed84[uVar6] = uVar7;
```

Og længere oppe:

```
if (param_1 == 0x40) {  
    uVar4 = 10;  
    uVar7 = 3;  
}  
else if (param_1 < 0x41) {  
    if (param_1 == 0x10) {  
...  
        uVar4 = 8;  
        uVar7 = 2;  
    }  
    else {  
        if (param_1 != 0x20) {  
            return in_lr;  
        }  
        uVar4 = 8;  
        uVar7 = 1;  
    }  
}  
else {  
    if (param_1 != 0x80) {  
        return in_lr;  
    }  
...  
    uVar4 = 10;  
    uVar7 = 4;  
}
```

Når man ser konstanter, er det altid interessant. Og da vi har en god fornemmelse om at vi er ude efter taster ("keys") kan vi søge efter dette i en søgemaskine:

```
#define KEY_SELECT      0x0004  
#define KEY_START       0x0008  
#define KEY_RIGHT        0x0010  
#define KEY_LEFT         0x0020  
#define KEY_UP           0x0040  
#define KEY_DOWN         0x0080
```

fra:

<https://www.coranac.com/tonc/text/keys.htm>

Så:

```
1 = KEY_LEFT  
2 = KEY_RIGHT  
3 = KEY_UP  
4 = KEY_DOWN
```

Den rigtige rækkefølge bliver derfor:

```
Venstre, højre, op, ned  
Venstre, højre, op, ned  
Venstre, venstre, højre, højre  
ned, ned, op, op  
ned
```

Hvis dette indtastes, ses flaget i scroll teksten:



= nc3{julesne\_i\_år}

## Statisk!?:

Hvad hvis vi prøver at løse opgaven uden brug af en emulator/rigtig GBA?

Dette vil blive ret svært, da der er lavet om i charsettet. Dvs. at ASCII 'V' vil blive vist på skærmen som '{', og 'Z' vil blive vist som 'Å', osv. Dette kan også konstateres hvis man ser på "strings" i filen, og ser at scroll teksten ses med underlige karakterer midt inde i nogle af ordene:

```
SZ ER UI TILBAGE MED NCQCTF I YOYY     [[[ EN SIDSTE GANG  
FOR OS Y ^ MR[HUNDREDE OG THE_I`RD]
```

Dette hinter jo kraftigt til, at der bruges et andet charset, som man skal tage højde for, hvis man vil kun vil bruge statiske metoder.

Dette blev gjort for at vække folks interesse for den måde, de her ældre maskiner fungerer på: Med tilesets og tilemaps. Herunder ses f.eks. debuggeren i VisualBoyAdvance, der kan vise "Tile Viewer":



Herunder er det script, der lavede flaget:

```
# -*- coding: utf-8 -*-
#flag = 'nc3{julesne_i_år}'
#abcdefghijklmnopqrstuvwxyz{|}_2å.|||WZ
#ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
flag = 'NCQVJULESNEXIXZRW'

correct_keys = [1, 2, 3, 4, 1, 2, 3, 4,      1, 1, 2, 2, 4, 4, 3, 3,      4]
r = ''

for i in range(0, len(flag)) :
    c = ord(flag[i])
    c = c ^ correct_keys[i]
    r += hex(c) + ", "

# 0x4f, 0x41, 0x52, 0x52, 0x4b, 0x57, 0x4f, 0x41, 0x52, 0x4f, 0x47, 0x5a, 0x4d, 0x5c, 0x59, 0x51,
# 0x53,
print(r)

r = ''
encodedFlag = [0x4f, 0x41, 0x52, 0x52, 0x4b, 0x57, 0x4f, 0x41, 0x52, 0x4f, 0x47, 0x5a, 0x4d, 0x5c, 0x59, 0x51, 0x53, ]
for i in range(0, len(encodedFlag)) :
    c = encodedFlag[i]
    c = c ^ correct_keys[i]
    r += chr(c) + " "

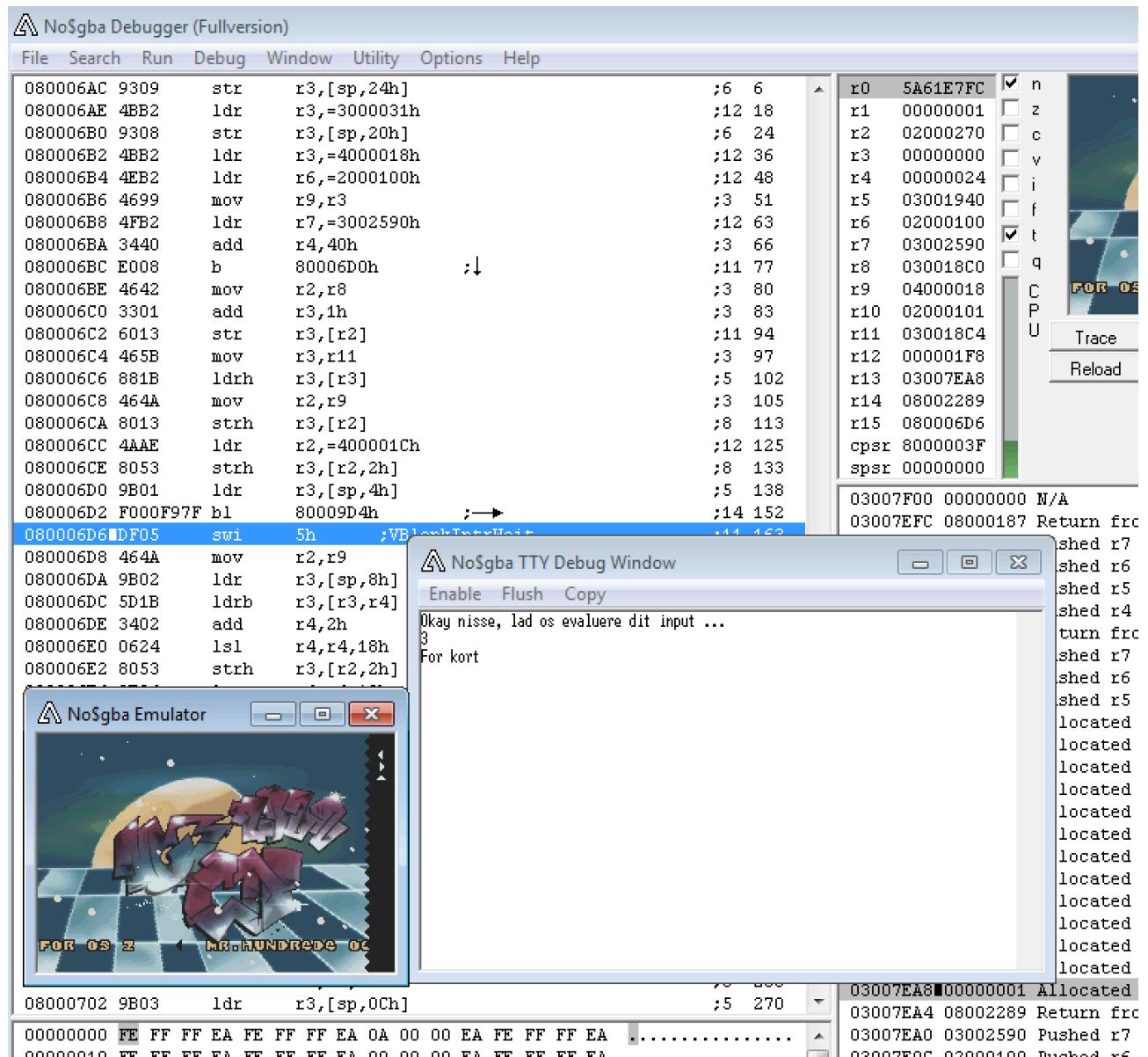
print(r)
```

## Ekstra hjælp:

Udover en del strenge til at lokalisere koden, efterlod vi også disse strenge i en "printf" til debuggeren NoCash fra:

<https://problemkaputt.de/gba.htm>

Dvs. at hvis man kører introen i denne emulator, og vælger "Debug Window", kan man nemt se hvad programmet printer ud. Dette kunne også bruges til at printe egne værdier når man evt. skulle udrede hvad de enkelte integers indeholdt:



## Mere hjælp?

Nååårh ja. Strings viser jo lidt mere:

Creditz: Kode, grafik & musik: mrloo | Effekter: THeWiZRD

og:

1:E6C0h	0D 0D 0E 0E	0E 0F 0F 10	20 20 20 00	20 20 20 00	..... . .
1:E6D0h	20 20 20 00	20 20 20 00	30 31 32 33	34 35 36 37	. . . 01234567
1:E6E0h	38 39 61 62	63 64 65 66	00 30 31 32	33 34 35 36	89abcdef.0123456
1:E6F0h	37 38 39 41	42 43 44 45	46 00 28 20	20 20 20 29	789ABCDEF.( )
1:E700h	00 30 00 FF	30 30 30 30	30 30 30 30	30 30 30 30	.0.ÿ00000000000000
1:E710h	30 30 30 30	20 20 20 20	20 20 20 20	20 20 20 20	0000
1:E720h	20 20 20 20	5F 5F 5F 5F	5F 5F 5F 5F	00 00 00 00	----- . . .
1:E730h	20 20 20 00	20 48 6A E6	6C 70 65 6E	20 65 72 20	. Hjælpen er
1:E740h	6E E6 72 3A	20 68 74 74	70 3A 2F 2F	72 65 62 72	nær: <a href="http://rebrand.ly/vlxxye">http://rebrand.ly/vlxxye</a>
1:E750h	61 6E 64 2E	6C 79 2F 76	6C 78 78 79	65 20 20 20	eller <a href="https://rebrand.ly/ahq589">https://rebrand.ly/ahq589</a>
1:E760h	65 6C 6C 65	72 20 20 68	74 74 70 73	3A 2F 2F 72	.ÿÿ
1:E770h	65 62 72 61	6E 64 2E 6C	79 2F 61 68	71 35 38 39	
1:E780h	20 20 20 20	20 20 20 20	20 20 20 20	00 FF FF FF	
1:E790h	20 20 20 20	20 20 20 20	20 20 00 00	20 20 20 20	..
1:E7A0h	20 20 20 20	00 00 00 00	20 20 20 20	20 20 20 20	....
1:E7B0h	20 20 20 00	20 20 20 20	20 20 20 20	20 20 00 00	. . ..
1:E7C0h	20 20 20 20	20 20 20 00	20 20 20 20	20 20 20 20	. .
1:E7D0h	20 20 20 00	20 20 20 20	20 20 20 20	20 20 20 20	.
1:E7E0h	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 20	
1:E7F0h	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 20	
1:E800h	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 20	.ÿÿ....
1:E810h	20 20 20 20	20 20 20 20	20 00 FF FF	05 00 00 00	

Hjælpen er nær: <http://rebrand.ly/vlxxye> eller <https://rebrand.ly/ahq589>

Nu er en joke jo aldrig rigtig sjov, hvis den skal forklares. Men der er tale om vores danske pendant til den klassiske Rick Roll. Nemlig den nu klassiske Guld Jul.

## Outro

Det var det. Den sidste challenge fra os 2 til NC3s CTF. Det har været en fornøjelse.

#PeaceOut

