MICT1

EXERCISE WEEK 2

Joeri van Grimbergen (1244825) | Nursize Bilen (1260235) | Rick van Gorp (1328417)
ZUYD HOGESCHOOL | HEERLEN | GROEP 4

1. What we found

De bedoeling van deze weekopdracht was om het antwoord op een vraag te vinden die was verborgen in een mediabestand. Voor dit onderzoek zijn twee mediabestanden beschikbaar gesteld, waarbij het eerste bestand een verborgen hint bevatte die gebruikt zou kunnen worden voor het onderzoek naar het tweede bestand.

De mediabestanden zijn twee afbeeldingstypen (PNG en MNG):

- File1.png
- File2.mng

Om de informatie verborgen in het eerste bestand te vinden wordt de applicatie op de website http://regex.info/exif.cgi gebruikt. Deze tool geeft onder andere het volgende resultaat weer:

- Het bericht in de andere file moet via ROT13 worden gedecodeerd.

In hoofdstuk 3 wordt beschreven hoe deze resultaten zijn gevonden. Aan de hand van bovenstaande informatie is bekend geworden dat de vraag die in het andere bestand gesteld wordt met behulp van het ROT13 algoritme moet worden gedecodeerd.

Vervolgens is het verborgen bericht uit het tweede bestand vastgesteld. Hiervoor is door groep 4 een Python script ontworpen. Voordat het script in gebruik is genomen zijn aan de hand van de documentatie van MNG en PNG de chunks en enkele bijbehorende waarden vastgesteld. Deze resultaten worden ook verder in hoofdstuk 3 uitgelegd.

Om de verborgen informatie efficiënter en eenvoudiger te vinden wordt gebruik gemaakt van het Python script. In het Python script zijn de definities van de chunks uit de documentatie van MNG en PNG gebruikt. Per chunk wordt er een entry voor een array aangemaakt met de gegevens: Chunk Type, Lengte van Chunk en de Offset. Dit script wordt gebruikt om de meest voorkomende chunk lengtes per chunk type te kunnen bepalen en om te kunnen bepalen of er van de standaard afwijkende waarden voorkomen in het bestand. Uit deze meest voorkomende waarden wordt vervolgens bepaald welke chunks met hetzelfde chunk type afwijken van de meest voorkomende waarde.

Met behulp van het Python script is de offset (*0x9ff7d9*) gevonden. Aan de hand van deze offset wordt de beginpositie van de chunk bepaald en kan de chunk data uitgelezen worden. Zoals eerder vermeld, wordt de gevonden string met behulp van ROT13 gedecodeerd:

String/Bericht:	*Hvg jryx wnne vf qr svyz jnne va qr fprar hvg qrmr navzngvr ibbexbzg?*
Vraag:	Uit welk jaar is de film waar in scene uit deze animatie voorkomt?
Antwoord:	2015 - Star Wars: The Force Awakens.

2. Where we found it

In dit hoofdstuk worden de resultaten weergegeven waar de in hoofdstuk 1 beschreven informatie gevonden is in de vorm van screenshots.

De eerste output van de EXIF-viewer van het eerste bestand file1.PNG. Onder MICT1 Hint staat het betreffende bericht.

PNG

Animation	no
Bit Depth	8
Color Type	RGB
Compression	Deflate/Inflate
Filter	Adaptive
Image Size	1,137 × 758
Interlace	Noninterlaced
Profile Name	ICC profile
Pixels Per Unit X	2,835
Pixels Per Unit Y	2,835
Pixel Units	meters
Modify Date	**2016:02:20** 20:06:43 2 days, 16 hours, 34 minutes, 33 seconds ago
MICT1 Hint	Het bericht in de andere file moet via ROT13 worden gedecodeerd. Succes!

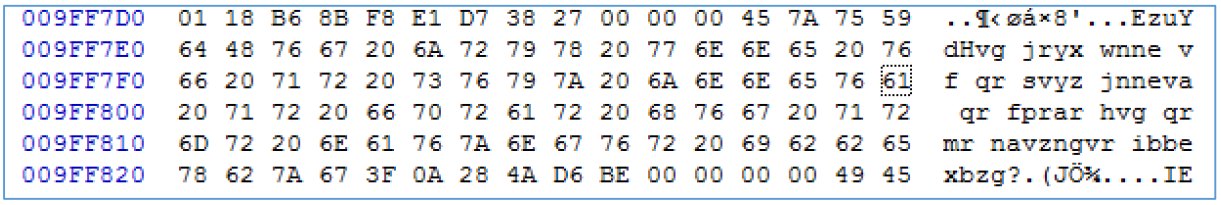
Daarna wordt het tweede bestand geanalyseerd met behulp van het Python script. Onderstaand enkele voorbeelden van chunk types die niet gelijk zijn aan meest voorkomende chunk lengte:

```
Found anomaly in lenghts of: IDAT
Explanation: It differs from the most used chunk length 8192
Length found:4029

Found anomaly in lenghts of: IDAT
Explanation: It differs from the most used chunk length 8192
Length found:6915

Found anomaly in lenghts of: IDAT
Explanation: It differs from the most used chunk length 8192
Length found:1270

Found anomaly in lenghts of: IDAT
Explanation: It differs from the most used chunk length 8192
Length found:1900

Found anomaly in lenghts of: IDAT
Explanation: It differs from the most used chunk length 8192
Length found:1521

Found anomaly in lenghts of: IDAT
Explanation: It differs from the most used chunk length 8192
Length found:657

Found anomaly in lenghts of: IDAT
Explanation: It differs from the most used chunk length 8192
Length found:5381

Found anomaly in lenghts of: IDAT
Explanation: It differs from the most used chunk length 8192
Length found:4416
```

Het python script geeft bij de analyse van afwijkende chunk types een opvallende uitkomst weer. In deze uitkomst staat de chunk offset die wordt gebruikt om de positie van de chunk in een hex editor te bepalen.

```
Found anomaly for Chunk Type used in MNG-file:

Chunk Type:     Chunk Length    Chunk Offset
zuYd            69              0x9ff7d9
```

Met behulp van bovenstaande Chunk Offset wordt de locatie van de ROT13 string bepaald. Deze string is onderstaand weergegeven:



3. How we found it

Om tot de hierboven beschreven resultaten te komen is er een Python script ontworpen door groep 4. In dit gedeelte zal beschreven worden hoe dit Python script tot stand is gekomen en het zal een korte handleiding bevatten om het Python script te bedienen.

3.1 Handelingen voor resultaten

In dit gedeelte worden de uitgevoerde handelingen beschreven om tot de hierboven beschreven resultaten te komen.

3.1.1 Hint in eerste bestand

In de opdracht werd gedefinieerd dat er informatie verborgen stond in file1.png die nuttig kan zijn voor het onderzoek naar het tweede bestand. Vaak is er veel informatie af te leiden van een afbeelding door het weer te geven in een EXIF-viewer. Om de informatie voor deze afbeelding weer te geven is gebruik gemaakt van http://regex.info/exif.cgi.

PNG

Animation	no
Bit Depth	8
Color Type	RGB
Compression	Deflate/Inflate
Filter	Adaptive
Image Size	1,137 × 758
Interlace	Noninterlaced
Profile Name	ICC profile
Pixels Per Unit X	2,835
Pixels Per Unit Y	2,835
Pixel Units	meters
Modify Date	**2016:02:20** 20:06:43 2 days, 16 hours, 34 minutes, 33 seconds ago
MICT1 Hint	Het bericht in de andere file moet via ROT13 worden gedecodeerd. Succes!

In bovenstaande afbeelding staat een deel van de resultaten weergegeven. Hierin wordt beschreven dat het bericht in de andere file met behulp van het ROT13 algoritme moet worden gedecodeerd.

3.1.2 Chunkdefinities

In dit gedeelte wordt beschreven hoe tot de resultaten die betrekking hebben op de chunkdefinities is gekomen:
De eerste 8 bytes voor MNG Header:
8A 4D 4E 47 0D 0A 1A 0A

Chunk Layout		
Length	4bytes	Data field only
Chunk type	4 bytes	Upper and lovercase ascii letters of getallen
Chunk data	0 bytes	
CRS	4 bytes	

IHDR Image Header Chunk		
Length	00 00 00 1C	28
Chunk type	4D 48 44 52	MHDR
Chunk data	-	
Width	00 00 03 20	800
Height	00 00 01 4B	331
Ticks_per_second	00 00 03 E8	1000
Nominal_layer_count	00 00 00 01	1
Nominal_frame_count	00 00 00 30	48
Nominal_play_time	00 00 00 64	100
Simplicity_profile	00 00 01 FF	0001 1111 1111
Profile Validity	0 (unspecified absence of features)	
Simple MNG Features	0 (absent)	
Complex MNG Features	0 (absent)	
Internal transparency	1 (may be present)	
JNG	1 (JNG or JDAA may be present)	
Delta-PNG	1 (May be present)	
Validity Flag	1 (Absence or Presence in bit 7, 8, 9 defined)	
Background Transparency	1	
Semi- transparency	1	
Stored object buffers	1	

CRC	C9 EF 2F 96
CRC	A1 BE 7F 8F

IHDR Chunk		
Length	00 00 00 0D	13 bytes
Chunk Type	49 48 44 52	IHDR
Chunk Data:		
Width	800	
Height	331	
Bit depth	08	8
Color type	02	2
Compression method	00	0
Filter Method	00	0
Interlace Method	00	0

Next Chunk (Actual Image)		
Length	00 00 20 00	8192
Chunk_Type	49 44 41 54	IDAT

Next Chunk		
Length	00 00 20 00	8192
Chunk_Type	49 44 41 54	IDAT

Next Chunk		
Length	00 00 20 00	8192
Chunk_Type	49 44 41 54	IDAT

Next Chunk		
Length	00 00 20 00	8192
Chunk_Type	49 44 41 54	IDAT

Next Chunk		
Length	00 00 1B 03	6915
A54A0	0A	

Next Chunk		
Length	00 00 0F BD	4029
5E2F0	01	

IEND chunk		
Length	0	
Chunk type	49 45 4E 44	IEND
Chunk_data	None	
CRC	AE 42 60 82	

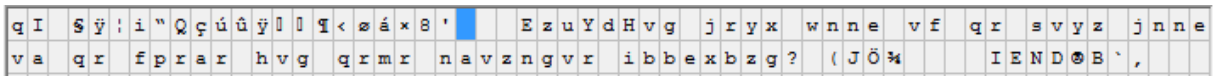
Deze uitkomsten zijn gebaseerd op de documentatie op: http://www.libpng.org/pub/mng/spec/ en http://www.libpng.org/pub/png/spec/1.2/PNG-Chunks.html

3.1.3 Python script

Om de resultaten in het Python script weer te geven is het Python script uitgevoerd. Het bestand file2.mng moet in dezelfde map staan als het Python script. In 3.3 Uitleg Python script wordt beschreven hoe het Python script is opgebouwd. In 3.2 Controle uitvoer Python script wordt de controle van de uitvoer van het Python script beschreven.

3.1.4 Conversie van tekst met ROT13-algoritme

Met behulp van de offset die het Python script als uitvoer heeft gegenereerd is de volgende chunk gevonden:



In de chunk data staat de string: *Hvg jryx wnne vf qr svyz jnne va qr fprar hvg qrmr navzngvr ibbexbzg?*

Door deze string te converteren met een online tool op http://www.rot13.com/ is het resultaat:

Hvg jryx wnne vf qr svyz jnne va qr fprar hvg qrmr navzngvr ibbexbzg?

ROT13 ▾

Uit welk jaar is de film waar in de scene uit deze animatie voorkomt?

3.2 Controle uitvoer Python script

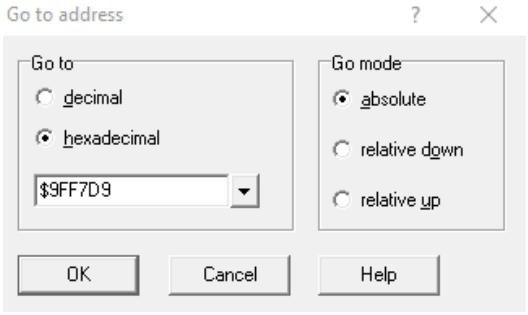
De uitvoer van het Python script wordt gecontroleerd met een hex-editor (XVI32). In de hex-editor is het bestand file.mng geopend.

De relevante uitkomst die gecontroleerd wordt is onderstaand weergegeven:

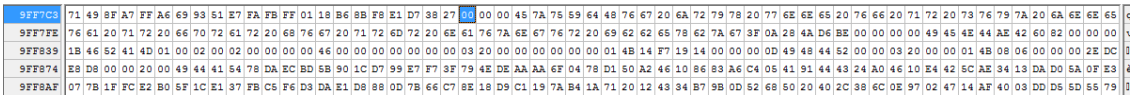
```
Found anomaly for Chunk Type used in MNG-file:

Chunk Type:      Chunk Length     Chunk Offset
zuYd             69               0x9ff7d9
...
```

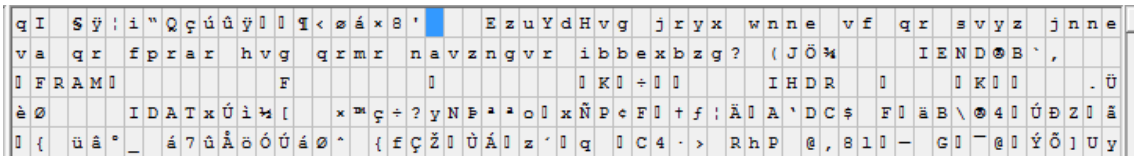
In de hex-editor is gezocht naar de betreffende offset die het script als uitvoer weergeeft door de volgende handelingen uit te voeren in het menu: *Address -> Goto…*

Met onderstaande instellingen en waardes is gezocht:



Vervolgens werden in de hex-editor de volgende waarden weergegeven:





De gemarkeerde waarde is het deel van de chunk dat de lengte van de volledige chunk weergeeft. Deze is 4 bytes lang. Na deze 4 bytes wordt de Chunk Type weergegeven. Deze waarde is gelijk aan zuYd, zoals in de uitvoer van het script wordt weergegeven. Deze set bytes wordt gevolgd door de data binnen de Chunk en kunnen worden geconverteerd met het ROT-13 algoritme naar plaintext (leesbare teskt).

3.3 Uitleg Python script

Het Python script is te vinden in de bijbehorende Github-repository van groep 4. Onderstaand worden delen van het script kort uitgelegd.

```
from struct import *
```

Bovenstaand wordt één library weergegeven waar het Python script gebruik van maakt. *Struct* wordt gebruikt om binaire data te vertalen naar bijvoorbeeld *unsigned integers* en *unsigned chars*. *Signed* is ook mogelijk.

```
#MNG Header definitie
intDefinition = 8 #Length of MNG identifier = 8 bytes

#MNG Chunk definitie
intLength = 4        #Length of MNG chunk length identifier = 4 bytes
intChunkType = 4     #Length of MNG chunk type = 4 bytes
intChunkData = 0     #Depends on intLength
intCRC = 4           #Length of CRC = 4 bytes
```

Vervolgens is volgens de definitie die te vinden is op: http://www.libpng.org/pub/mng/spec/ en http://www.libpng.org/pub/png/spec/1.2/PNG-Chunks.html vastgesteld hoe een chunk in een *mng*-bestand is opgebouwd. De waardes die in bovenstaande afbeelding zijn weergegeven omvatten het aantal bytes waaruit de onderdelen van de chunk bestaan. De lengte van de chunkdata hangt af van de definitie van de lengte van de chunkdata. Deze moet eerst opgehaald worden uit het bestand alvorens deze gebruikt kan worden.

```
#Function to read file by pre-defined chunks
def ReadChunk(f):
    offset = hex(f.tell()) #Get current location in file
    intLchunk = f.read(intLength)
    intLchunk = unpack('>I', intLchunk)[0] #Read as Big Endian -> Unsigned Integer
    strType = f.read(intChunkType)
    strType = str(strType)[2:len(str(strType))-1]
    strChunkData = f.read(intLchunk)
    intChecksum = f.read(intCRC)
    intChecksum = unpack('>I', intChecksum)[0] #Read as Big Endian -> Unsigned Integer
    return [strType, intLchunk, offset]
```

In bovenstaande afbeelding wordt de functie *ReadChunk* weergegeven. Deze functie gebruikt als invoer een bestand en definieert deze als *f.* Vervolgens wordt de huidige offset bepaald en wordt één chunk uit het bestand volledig gelezen en opgeslagen in variabelen. In dit gedeelte worden de byte-lengtes die in de MNG Chunk definitie zijn gedefinieerd gebruikt. Sommige waardes maken gebruik van de functie *struct.unpack*. Deze waardes worden gelezen als Big Endian en Unsigned Integer. Dit resulteert in een integer die opgeslagen wordt in een Python variabele. De waardes die zijn vastgesteld voor een chunk: *Chunk Type, Lengte Chunk en Offset* worden return gestuurd.

```
#Count most common values to determine general size used per chunk data type
def FindMostCommon(chunkType):
    counter = {}
    for i in range(0,len(arrValues)):
        if arrValues[i][0] == chunkType:
            counter[arrValues[i][1]] = counter.get(arrValues[i][1], 0) + 1
    counts = [(j,i) for i,j in counter.items()]
    if counts:
        intMostCommon = max(counts)[1]
        return(intMostCommon)
```

In bovenstaande afbeelding wordt de functie *FindMostCommon* weergegeven. Deze functie gebruikt als invoer een string die gelijk is aan *ChunkType*. Deze functie wordt gebruikt om per Chunk Type te bepalen wat de meestgebruikte lengte is van de chunk. De waarde die het meest voorkomt wordt als *intMostCommon* return gestuurd.

```
#Print anomalies in length per chunk type
def PrintAnomalyCount(chunkType):
    intMostCommon = FindMostCommon(chunkType)
    for i in range(0,len(arrValues)):
        if arrValues[i][0] == chunkType:
            if arrValues[i][1] != intMostCommon and arrValues[i][1] != 0:
                print("Found anomaly in lenghts of: " + arrValues[i][0])
                print("Explanation: It differs from the most used chunk length " + str(intMostCommon))
                print("Length found:" + str(arrValues[i][1]) +"\n")
```

In bovenstaande afbeelding wordt de functie *PrintAnomalyCount* weergegeven. Deze functie gebruikt als invoer een string die gelijk is aan *chunkType*. Deze functie wordt gebruikt om per Chunk Type afwijkende waarden als uitvoer weer te geven.

```
file = open('file2.mng', 'rb')

magic_val = file.read(8)
arrDefaultMNG = ["IDAT", "IEND", "IHDR", "MHDR", "MEND", "TERM", "tEXt","tIME", "FRAM", "LOOP"]
```

Dit is code dat buiten de functies valt. In dit gedeelte wordt het *mng-bestand* geopend in binaire leesmodus en worden de eerste 8 bytes gelezen, omdat dit nog geen onderdeel is van een chunk. Vervolgens worden de meestvoorkomende *MNG-chunktypes* in een array gedefinieerd.

```
#Read complete file until end
blnEOF = False
arrValues = []

while blnEOF == False:
    try:
        arrValues.append(ReadChunk(file))
    except:
        blnEOF = True
```

Bovenstaande loop wordt gebruikt om tot het einde van het bestand de gegevens per chunk op te halen. De waardes die als return worden gestuurd door de functie *ReadChunk* worden toegevoegd aan het array *arrValues*.

```
for i in range(0,len(arrDefaultMNG)):
    PrintAnomalyCount(arrDefaultMNG[i])
```

In bovenstaande *for-loop* wordt per chunk type de functie *PrintAnomalyCount* aangeroepen. De waarde van het array dat zich op index *i* bevindt wordt als variabele meegestuurd.

```
#Check if anomaly in Chunk Type exists
for i in range(0,len(arrValues)):
    if ((arrValues[i][0]) not in arrDefaultMNG):
        print("Found anomaly for Chunk Type used in MNG-file:\n")
        print("Chunk Type:\tChunk Length\tChunk Offset")
        print(str(arrValues[i][0]) + "\t\t" + str(arrValues[i][1]) + "\t\t" + str(arrValues[i][2]))
```

In bovenstaande *for-loop* wordt bepaald of het bestand waardes chunk types bevat die niet overeenkomen met de specificaties van *MNG* of *PNG.* Deze chunk types zijn te vinden op: http://www.libpng.org/pub/mng/spec/ en http://www.libpng.org/pub/png/spec/1.2/PNG-Chunks.html. Als er een afwijking gevonden is wordt deze weergegeven met de lengte van een chunk en de bijbehorende offset.