Steganography & Amazon Cloud Drive

The recent announcement by Amazon about their new unlimited tiers for their Cloud Drive storage made me think. They are now offering unlimited storage in 2 tiers, one for unlimited photos, and another one for unlimited files of any type (Office documents, PDFs, videos, etc.); these cost USD11.99 & USD59.99 respectively per year (both very reasonable prices).

With the use of steganography I wondered, would it be possible to upload any file masked as a graphic file in the unlimited photo tier? Let's find out shall we...

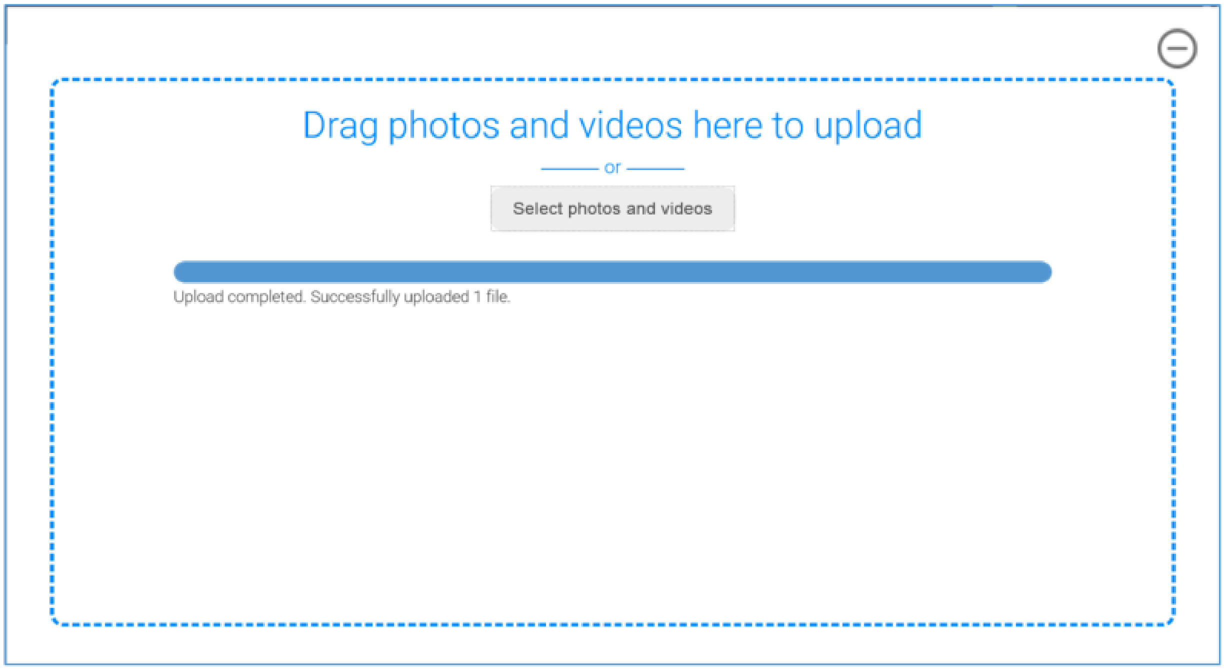
The purpose of this test is not to trick Amazon out of any lost revenue, but to find out how Amazon determines that the "graphic" file that you upload is legitimate.

- OpenPuff - http://embeddedsw.net/OpenPuff_Stega...

OpenPuff is an open source steganographic tool that supports many file types as the "carrier", including .PNG graphic files. It also offers various ways to protect your hidden file; the choice of 16 different cryptographic algorithms, as well as multi-layered data obfuscation (up to 3 passwords). As stated on their website; "Data, before carrier injection, is encrypted, scrambled, whitened and encoded".

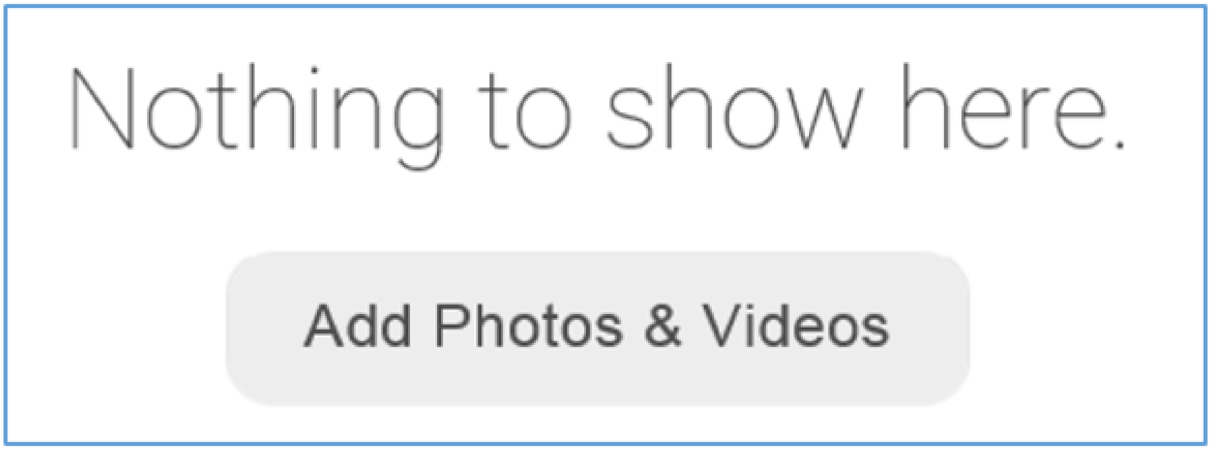
Before getting to the steganography test I wanted to first check other ways users could potentially upload other files as a graphic file to Amazon Cloud Drive.

As a first pass I renamed a Microsoft Office .DOCX file as a .PNG and tried to upload it.

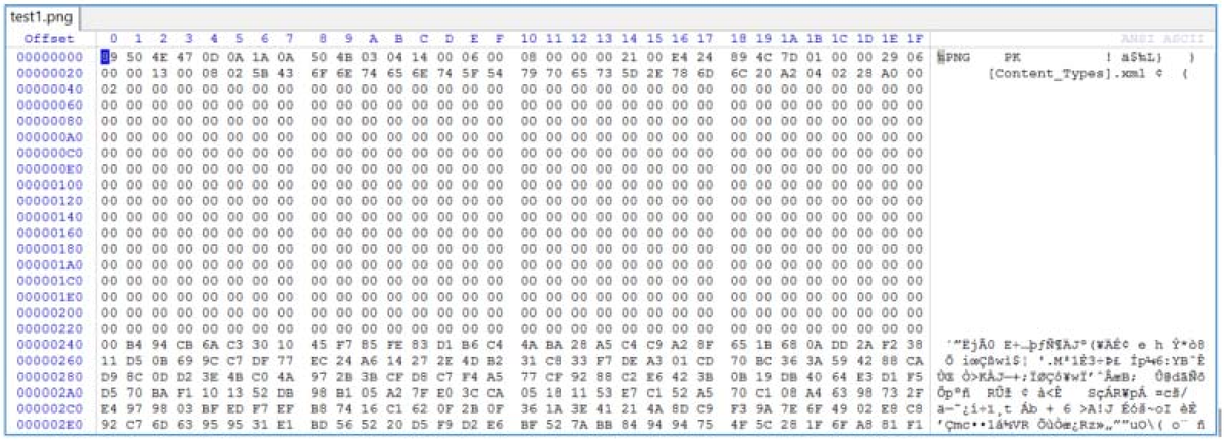


Upload completed. Successfully uploaded 1 file.

The file uploaded correctly, but it was not visible in my "Photos" folder, instead Amazon Cloud Drive had stored it in the root folder as it had not been detected as a graphic file, while my "Photos" folder informed me that it was empty.

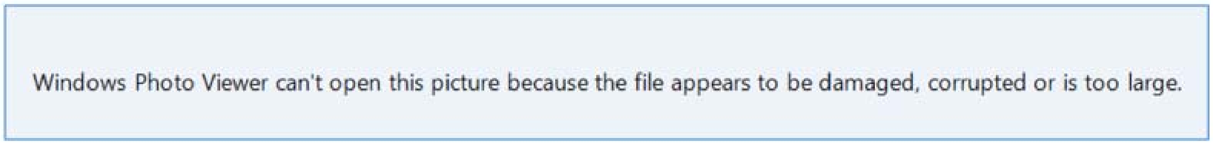

Nothing to show here.

I then took a legitimate .PNG file, opened it up in a hex editor and pasted the .DOCX file just after the .PNG file header - `89 50 4E 47 0D 0A 1A 0A` and left the .PNG footer at the end of the file (after the end of the .DOCX file) - `49 45 4E 44 AE 42 60 82`. I then saved this file as a .PNG.



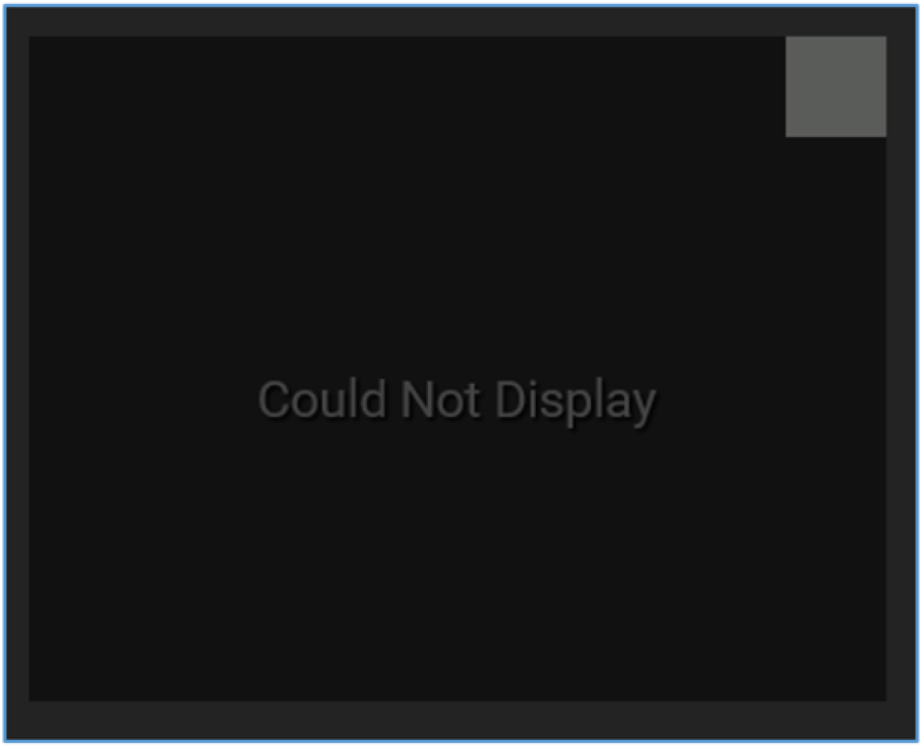
```
test1.png
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F  10 11 12 13 14 15 16 17  18 19 1A 1B 1C 1D 1E 1F  ANSI ASCII
00000000  89 50 4E 47 0D 0A 1A 0A  50 4B 03 04 14 00 06 00  08 00 00 00 21 00 E4 24  89 4C 7D 01 00 00 29 06  ‰PNG    PK          ! ä$‰L}   )
00000020  00 00 13 00 08 02 5B 43  6F 6E 74 65 6E 74 5F 54  79 70 65 73 5D 2E 78 6D  6C 20 A2 04 02 28 A0 00        [Content_Types].xml ¢  (
00000040  02 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000000A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000000C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000000E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000120  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000200  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000220  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000240  00 B4 94 CB 6A C3 30 10  45 F7 85 FE 83 D1 B6 C4  4A BA 28 A5 C4 C9 A2 8F  65 1B 68 0A DD 2A F2 38   ´”ËjÃ0 E÷…þƒÑ¶ÄJº(¥ÄÉ¢ e h Ý*ò8
00000260  11 D5 0B 69 9C C7 DF 77  EC 24 A6 14 27 2E 4D B2  31 C8 33 F7 DE A3 01 CD  70 BC 36 3A 59 42 88 CA   Õ iœÇßwì$¦ '.M²1È3÷Þ£ Íp¼6:YBˆÊ
00000280  D9 8C 0D D2 3E 4B C0 4A  97 2B 3B CF D8 C7 F4 A5  77 CF 92 88 C2 E6 42 3B  0B 19 DB 40 64 E3 D1 F5  ÙŒ Ò>KÀJ—+;ÏØÇô¥wÏ’ˆÂæB;  Û@dãÑõ
000002A0  D5 70 BA F1 10 13 52 DB  98 B1 05 A2 7F E0 3C CA  05 18 11 53 E7 C1 52 A5  70 C1 08 A4 63 98 73 2F  Õpºñ  RÛ˜± ¢ à<Ê   SçÁR¥pÁ ¤c˜s/
000002C0  E4 97 98 03 BF ED F7 EF  B8 74 16 C1 62 0F 2B 0F  36 1A 3E 41 21 4A 8D C9  F3 9A 7E 6F 49 02 E8 C8  ä—˜ ¿í÷ï¸t Áb + 6 >A!J Éóš~oI èÈ
000002E0  92 C7 6D 63 95 95 31 E1  BD 56 52 20 D5 F9 D2 E6  BF 52 7A BB 84 94 94 75  4F 5C 28 1F 6F A8 81 F1  ’Çmc••1á½VR ÕùÒæ¿Rz»„””uO\( o¨ ñ
```

Given this was a raw hack of a .PNG file trying to open it resulted in an error message.



Windows Photo Viewer can't open this picture because the file appears to be damaged, corrupted or is too large.

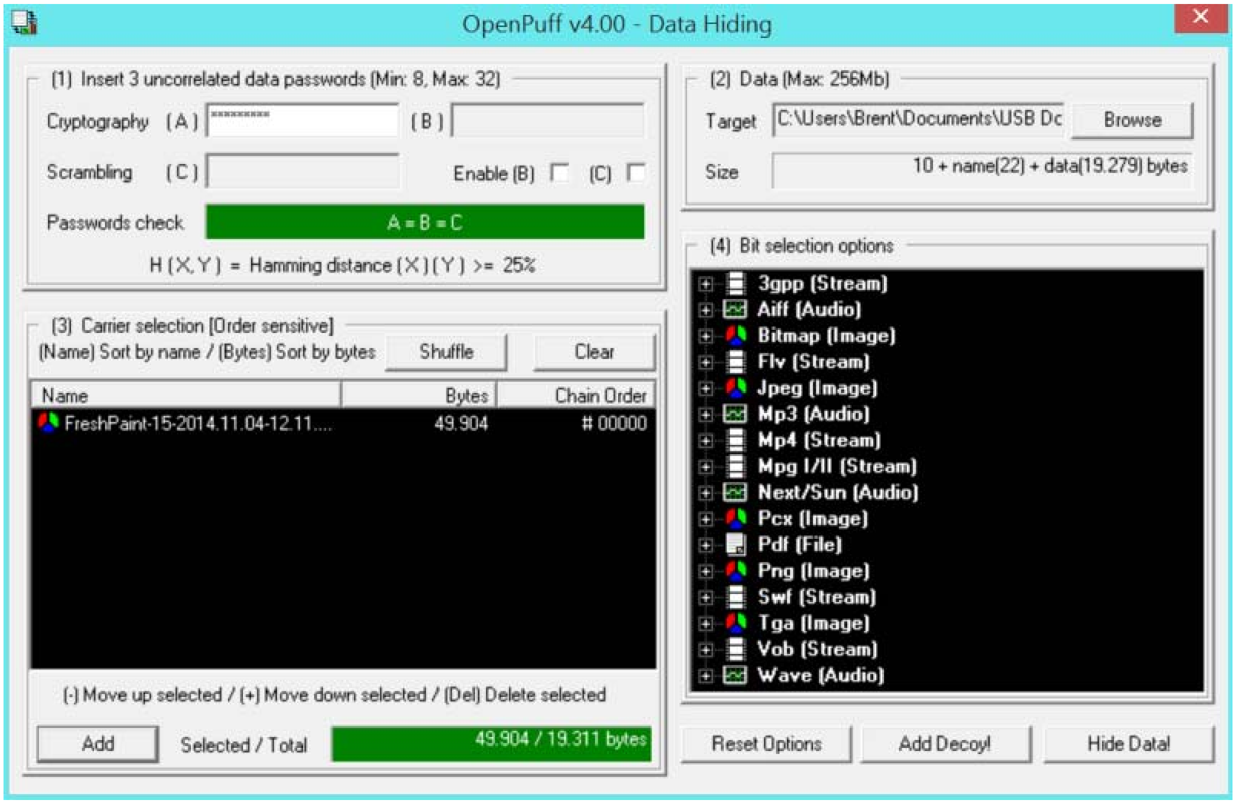
Either way I wanted to see how Amazon Cloud Drive would handle a corrupted .PNG file hiding a real .DOCX file inside it, so I tried to upload it. Again I received an uploaded completed correctly message, and this time the .PNG file was stored in the correct "Photos" folder on Amazon Cloud Drive. As to be expected though, the file could not be displayed correctly and Amazon Cloud Drive told me as much. So far so good, Amazon Cloud Drive thinks that my malformed .PNG file is legitimate.



Could Not Display

To test if Amazon Cloud Drive only checks a file header I did the same as above but this time I only used the .PNG header, and not the footer, and then uploaded it to Cloud Drive. This test was also successful, resulting in the malformed .PNG file being stored in the "Photos" folder, indicating that Amazon Cloud Drive only really checks a file signature, and not a footer. Now was the time to check how steganography would fare...

Using OpenPuff I hid this .DOCX file inside a legitimate .PNG and again tried to upload it.



OpenPuff v4.00 - Data Hiding

(1) Insert 3 uncorrelated data passwords (Min: 8, Max: 32)

Cryptography (A) ********** (B)

Scrambling (C) Enable (B) □ (C) □

Passwords check A = B = C

H (X,Y) = Hamming distance (X)(Y) >= 25%

(2) Data (Max: 256Mb)

Target C:\Users\Brent\Documents\USB Dc

Size 10 + name(22) + data(19.279) bytes

(3) Carrier selection [Order sensitive]

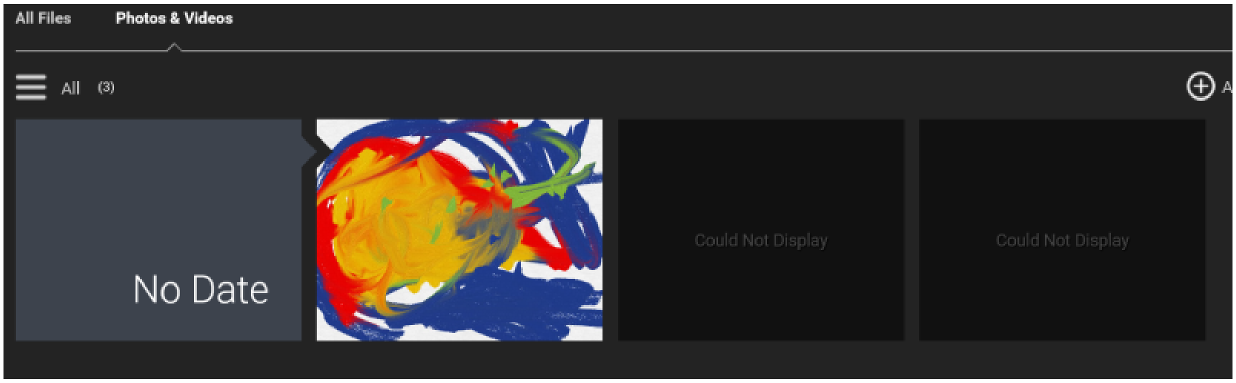
(Name) Sort by name / (Bytes) Sort by bytes

Name	Bytes	Chain Order
FreshPaint-15-2014.11.04-12.11....	49.904	# 00000

(-) Move up selected / (+) Move down selected / (Del) Delete selected

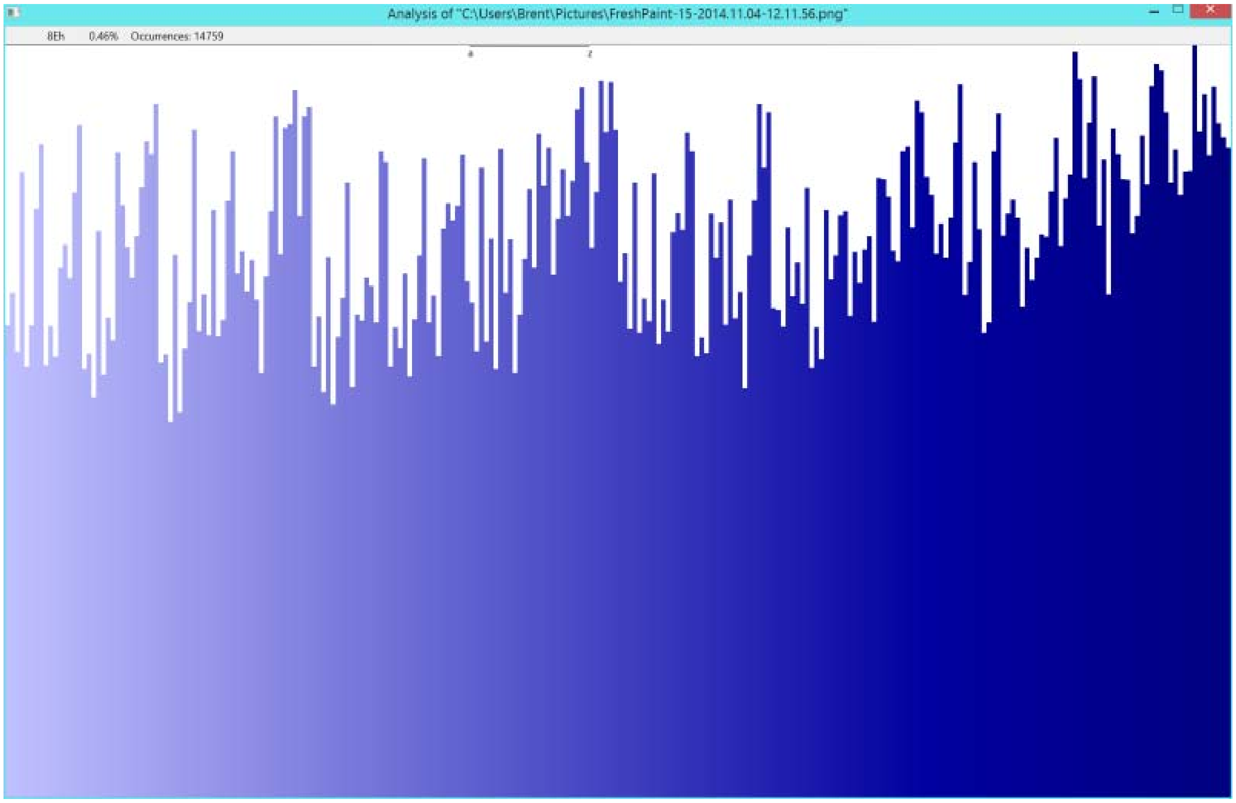
Selected / Total 49.904 / 19.311 bytes

This upload was successful, and because I had used steganography the .PNG file was still legitimate and visible through the Amazon Cloud Drive front-end.



There are many ways to detect steganographic files, such as statistical analysis and entropy testing, but detecting the presence of steganography does not equate to "cracking" the underlying steganographic system. OpenPuff uses encryption in its steganographic processes, and entropy tests will probably point you in the right direction, but you will be no closer to revealing the true contents. I highly doubt that Amazon will be implementing any statistical analysis tools during the Cloud Drive upload process to detect the use of steganography, as it is hard to actually prove/detect. For example a legitimate .PNG file has a high entropy value anyway. Below are the comparisons of the distribution of the hex values between the original .PNG file and the steganographically altered .PNG file (respectively). As you can see any automated entropy test would be unable to determine which file contained the steganographic data.

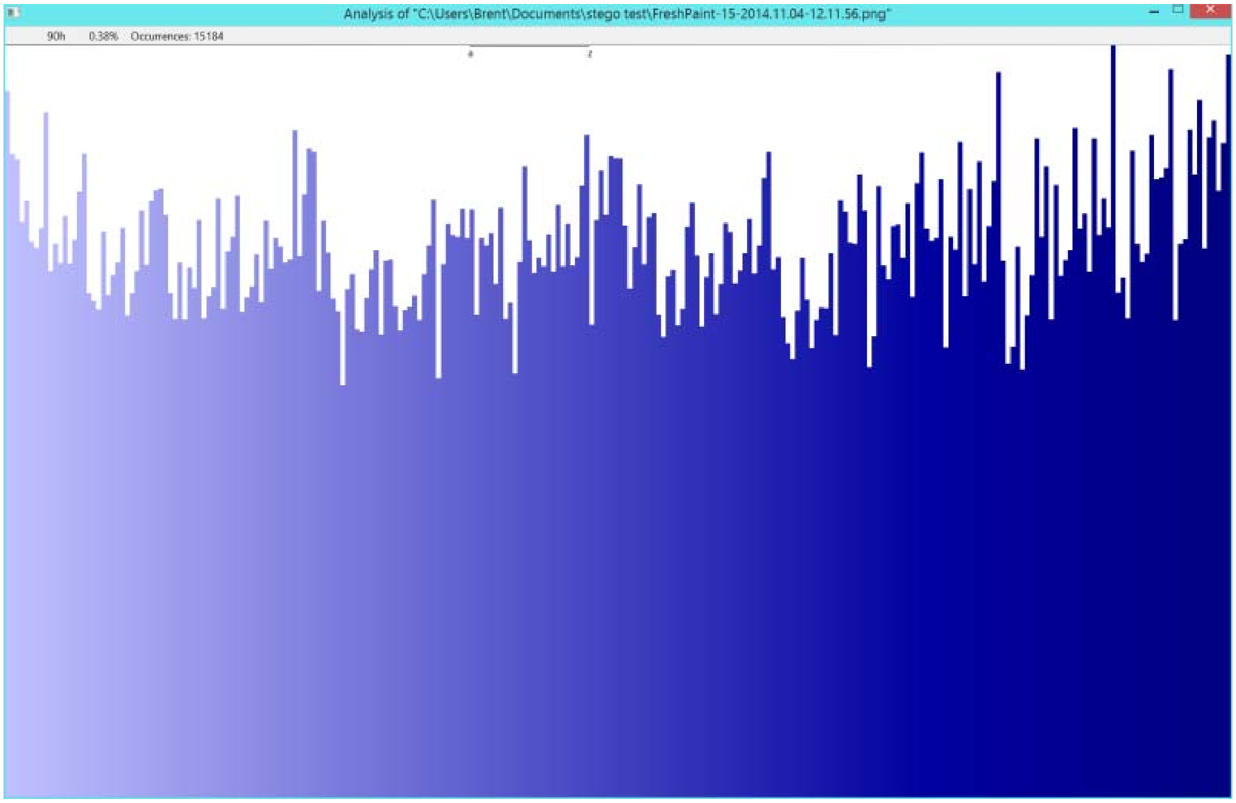
Original .PNG



Analysis of "C:\Users\Brent\Pictures\FreshPaint-15-2014.11.04-12.11.56.png"

8Eh 0.46% Occurrences: 14759

Steganographic .PNG

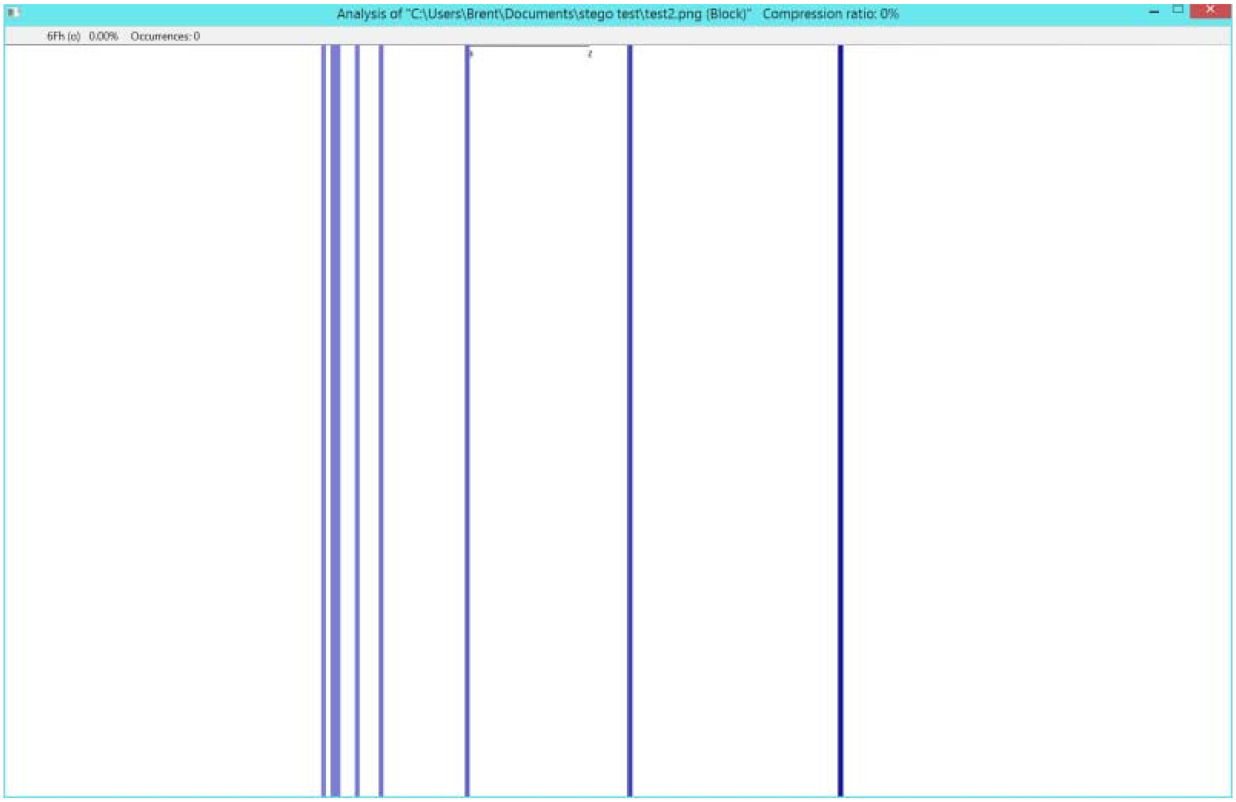


Analysis of "C:\Users\Brent\Documents\stego test\FreshPaint-15-2014.11.04-12.11.56.png"

90h 0.38% Occurences: 15184

To demonstrate the differences, the below graphic is for the malformed .PNG file I hacked together in a hex editor to use in the earlier test.

Malformed .PNG (.DOCX embedded after .PNG header)



Analysis of "C:\Users\Brent\Documents\stego test\test2.png (Block)" Compression ratio: 0%

6Fh (o) 0.00% Occurences: 0

Amazon Cloud Drive has a 2GB file size limit but using a steganographic tool such as OpenPuff you could (in theory) split a file into an unlimited number of segments and still upload these files to the Cloud Drive service on the unlimited "Photo" tier. Given that cloud storage is not 100% secure (what is) it is always a good idea to encrypt files before uploading them to cloud storage providers anyway, and since the Amazon model checks for legitimate graphic files (via their file signatures), using steganography is probably going to be your best choice.

The above method of using steganography with cloud storage providers is not an attempt to fool Amazon, merely an attempt to push the boundaries of what is possible.

Remember "Knowledge is power" and "with great power comes great responsibility", so use this knowledge wisely...