Steganography in Microsoft Office documents and ASP pages

Magdalena Pejas
The Warsaw University of Technology
Warsaw, Poland

ABSTRACT

In this paper I describe methods, how to use steganographic techniques to hide data in MS Office and Internet files. In the first chapter I present the idea of data hiding. Basic terms concerning steganography like steganographic data, function, key and capacity are explained here. In the second chapter I explain the object oriented construction of documents, which are created with MS Office software and with scripts included in dynamic web pages. I put the emphasis on the observation, how easy is to access wide range of files and their contents. The third chapter is dedicated to scripting programming. I draw attention to Visual Basic scripting language and its features. I include here the description of example objects and functions used to access basic data, which can be potentially used to hide secret information. Next chapter contains my conception how to estimate steganographic data capacity. Finally, in conclusion I underline the fact, that we do not really know, what documents and web pages really contain.

Keywords: Steganography, Scripts, Programming Language, Object Model, Method, Attribute and Data Capacity.

1. INTRODUCTION

The goal of the author of this paper is to show how popular electronic documents and Internet pages can be used to hide secret information. Thanks to their object oriented and very intuitive structure they can be accessed, modified and even generated with the help of various scripting languages. One of the most popular scripting language is Visual Basic script to construct MS Office documents. For web pages one can choose from various scripting languages, for example Java Script and Visual Basic, PHP or Perl. Scripts run under Microsoft Environment are very easy to write and to learn syntax. Java and VB Scripts need no special compilers or interpreters. This makes occasion to use these small programs to hide and extract hidden information in various documents, data bases, e-mails and dynamic web pages. It is also possible to send hidden data continuous stream via the Internet. Every document has very individual ability to contain additional data. For example documents full of graphics can hide more hidden content than text files.

2. STEGANOGRAPHY

Steganography means in Greek covered writing. It has been used since ancient times to hide secret signals. For example wooden bars with engraved writing and covered with wax layer used to serve as steganographic secret channel. Nowadays it can be easily adapted to informatics. Especially in the age of information and communication we can imagine, how it can be easy to put additional invisible data into tremendous data flow around the world.

The art of steganography can be compared to cryptography. While in cryptography information or data is only encrypted and the result of encryption is visible, in steganography also the fact of data hiding is invisible. This means, no one can be sure, if a given sample data does contain any secret content or not. The idea of steganographic system is shown in Fig. 1.

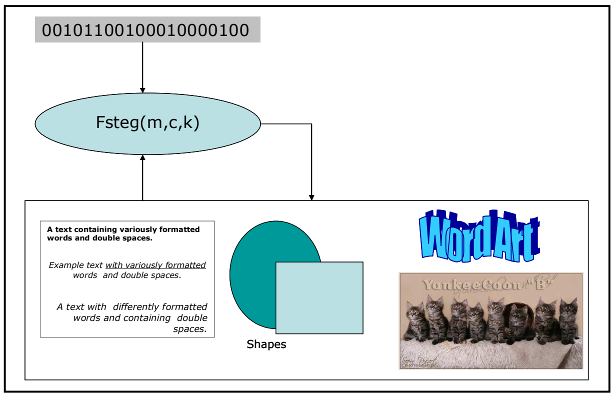
0010110010001000010
Fsteg(m,c,k)
A text containing variously formatted words and double spaces.
Example text with variously formatted words and double spaces.
A text with differently formatted words and containing double spaces.
Shapes
WordArt
YankeeCoon "B"

Fig. 1. The idea of data hiding

Cover data is such data, in which we hide secret data, message or information. In Fig. 1 it is denoted as *C*. Message or information to hide is represented by symbol *m*. Cover data with hidden message is in this case called steganographic data, denoted as *S*. Function F_{steg} to perform data hiding process is called steganographic function. This function may use steganographic key *K*, which is analogical to cryptographic key. Steganographic capacity is the relative amount of secret data which can be hidden in cover data of a given size in such a way, that the difference between the cover data before and after data hiding process is invisible. For example one can hide 100 bytes of text in 800 bytes of an image. This is in the case, when we hide one bit of information in every color component in each pixel of this example image.

3. THE STRUCTURE OF MS OFFICE DOCUMENTS AND WEB PAGES

The structure of MS Office documents is based on hierarchy of objects. Every MS Office document can be easily opened, read, modified and closed by small programs called scripts. Then the all elements of the document can be iterated as elements of an array. Now each element with its parameters is ready to be accessed and modified. For example a text frame contains plain text and parameters such as: width, height, vertical and horizontal position, background color, font color and border color. The textual content of the frame can be formatted in many ways. It depends on the parameters such as font name, style, size, font color and background color.

Main objects in MS Office documents can be classified as follows:

- MS Word - document
    - frames
    - paragraphs
    - tables
    - shapes: lines, rectangles, textboxes
- MS PowerPoint - presentation
    - slides
    - shapes: ovals, rectangles, lines
    - images, cliparts
- MS Excel - workbook
    - worksheets, cells
- MS Access
    - record sets
- MS Outlook
    - email content
    - file attachments

By analogy main objects in ASP pages can be classified as follows:

- tags in html: buttons, textboxes
- choice lists, check boxes
- tables
- graphic objects

These elements can be easily generated by scripts in dynamic web pages, which are based on text files. Scripting languages have access to such files, by opening, reading and modifying their content. What is more, dynamic web pages can be refreshed in a fixed time and thus can have their content changed with desired frequency.

Thanks to the art of steganography combined with scripting we can imagine secret data flow as shown in Fig. 2.

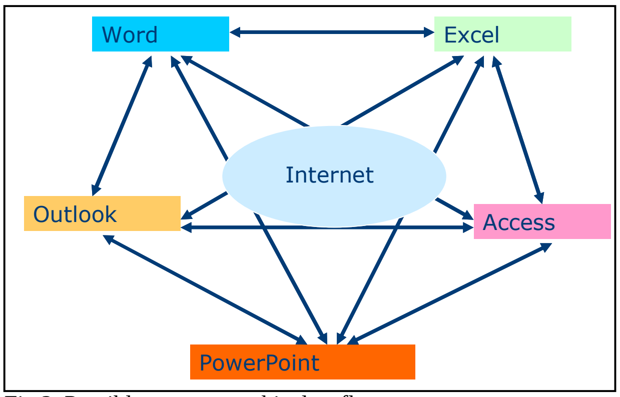

Fig 2. Possible steganographic data flow

Data can be freely converted to another data format and sent via the Internet. Of course data of one type can be hidden in data of another format, especially in web pages full of graphics.

4. THE POWER OF SCRIPTING

There are several scripting languages such as Java Script, Visual Basic Script or Perl.

Programs written in these languages give us many advantages. The most important ones are listed as follows:

- they are easy to implement and run,
- enable learning with the help of macro recording,
- give access to the tree of document objects,
- can generate document content.

First of all Visual Basic and Java scripts are very useful because of no need to install any additional editor, compiler or interpreter under Microsoft environment. What is more, in MS Office documents we can record and play simple example VB scripts called macros. If we don't know the name of an object and its attributes or syntax of its method, we can simply perform a simulation of object construction and of running its methods. For example any user of Word application can switch on a macro and then draw a text frame in the document, change its size and position, then format its content. Then when the user switches off the macro, he can open its code in the Visual Basic editor always available in MS Office Packet.

With the help of scripts we can perform general operations on directories and files. One can search and filter a list of files and then read their attributes, for example extension, size or date of creation. After recognizing the extension of a given file script can read its contents respectively to the file type. Text files are opened in another manner than MS Office ones.

Basic functions used to search and open documents of various types and to access their basic elements are presented in the listings below.

```
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFolder = objFSO.GetFolder("D:\workingdir")
Set colFiles = objFolder.files
For Each objfile in colFiles
            MsgBox(objFile.name)
Next
```

Listing 1. Iterating a list of files

Listing 1. presents a method to open a list of files from a given directory and to display their names. We have here three objects: *ObjFso* representing the file system object, *objFolder* representing a chosen folder and *objFile* is for the currently processed file. The name *colFiles* represents the set of all files visible in the chosen directory.

Another method shows how to open objects in MS Office applications. It is very interesting, that all application objects are opened in the same manner. Listing 2. shows functions opening MS Word, Excel, PowerPoint, Access and Outlook programs. Symbol *<AppName>* represents one of the application names. Every instance of a given application is represented by an object. With the methods of such object we can open and close program documents or set option if opened program is visible for the user or not. This last functionality is very convenient for background batch scripts.

```
Set objApp = CreateObject("<AppName>.Application")
objApp.Visible = False
objApp.Quit

Set objWord = CreateObject("Word.Application")
objWord.Visible = False
objWord.Quit
```

Listing 2. Run & Close applications

Each application object can be used to open documents from files directly, as it is shown in Listing 3. This is analogical for all available MS Office applications. With the object representing a given application one can open, save and close MS Office files of all available types.

```
Set objDocument = objWord.Documents.Open(„file.doc”)
Set objWorkBook = objExcel.Workbooks.Open(„file.xls”)
Set objPresentation = objPPT.Presentations.Open(„file.ppt”)
Set objDataBase =  objAccess.OpenCurrentDatabase(„file.mdb”)
objDocument.SaveAs(strPath1)
objWorkBook.SaveAs(strPath1)
objPresentation.SaveAs(strPath1)
objDocument.Close
objWorkBook.Close
objPresentation.Close
```

Listing 3. Opening & Closing documents

```
Set colParagraphs = objDocument.Paragraphs
iParagraphs=colParagraphs.Count
For iP=0 to iParagraphs-1
            Set objParagraph=colParagraphs(iP)
            MsgBox(objParagraph.name)
Next
```

Listing 4. Iterating document elements

Having document file, for example Word, PowerPoint or Excel open, a script can have access to all its contents, for example frames, paragraphs, shapes or worksheets, respectively. The object *objDocument* presented in Listing 4 contains paragraphs, which are iterated in the collection *colParagraphs* of objects. These objects are now easy to be accessed. Now any application user can extract text and its format. In Listing 5 we have an example code, how to extract some content from an object.

```
objParagraph.Range.Text
objParagraph.Shading.Texture
objFrame.Range.Text

objShape.Left
objShape.Top
objShape.Width
objShape.Height
objShape.Fill.ForeColor.RGB
objShape.Fill.Transparency
objShape.HasTextFrame
objShape.TextFrame
objShape.TextFrame.HasText
objShape.TextFrame.TextRange.Text

objWorkSheet.Cells(iR,iC)
```

Listing 5. Parameters of example document elements

Listing 5 presents a set of example objects, which can be directly used to hide additional bits of information. Some of these parameters can be changed only by one bit but other are less vulnerable to changes, so these changes are not visible by someone reading document.

By analogy the Internet pages can be generated by printing contents to a html file. Listing 6 shows an example how a script can open, read, write and close a text file. Object *objFso* has methods not only to open directories, iterate list of files but also to create or open for reading and writing any text file.

```
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.OpenTextFile(„file.txt”, ForReading)
Do Until objFile.AtEndOfStream
   strLines = strLines & objFile.ReadLine
Loop
objFile.Close

Set objFile = objFSO.OpenTextFile(„file1.txt”, ForWriting)
objFile.Write(strLines)
objFile.Close
```

Listing 6. Accessing a text file

Having access to all lines from a given text file, this is enough to replace the content of any html or asp file to have information hidden. Another method to manipulate parts of any web file is to include a script in the body of this file. An example of simple page content generation is shown in Listing 7.

```
<script type="text/javascript">
function f(form)
{
   if (form.checkbox1.value=="On")
   {
            strHtmlText=„<table>….</table>”;
            Document.write(strHtmlText);
   }
}
</script>
```

Listing 7. Manipulating web page content

The syntax, according to which basic elements of web pages can be generated is presented in Listing 8.

```
<hn align=„a” ></hn>

<font color=„c” size=„s”>
<b>…</b> <i>…</i> <u>…</u>
<blink>...</blink> <em>...</em> <strong>...</strong>

<frame bordercolor="c" scrolling="scr" noresize="noresize"
marginwidth="x" marginheight="y" />,

<img src=„image.gif„ width=„w” height=„h” border=„n”
alt=„at„ align=„a”/>

<table width=„w” height=„h” border=„b” bordercolor=„bc”
bgcolor=„bgc” cellpadding=„cp” align=„a”>
```

Listing 8. The syntax of example elements in html body

Here we have page elements like font, frame, image and table. They all have individual attributes. Apart from that there are additional simple elements like big *b*, italic *i*, underline *u* which also can be set to hide secret bits of information. A very specific element header *hn* can hide a value from one to 24, because *n* can be set from 1 to 6 and align *a* can take one of four values: „left”, „right”, „center”, „justify”. This makes 24 possibilities to obtain different kinds of headers.

If a given document element has more flexible parameters like color or width, than the number of different possibilities to set is much higher and it is equal to the multiplication of all setting possibilities of every attribute. This fact will be important for estimation of relative steganographic capacity of documents and web pages.

Another example ASP script presents how to access image pixels. It is shown in Listing 9.

```
<body>
<%@ Page Language="VB" %>
<%@ Import Namespace="System.Drawing" %>
<script runat="server">
   strPath=Server.MapPath("images/imageflip.jpg")
   myBitmap = _
  New Bitmap(System.Drawing.Image.FromFile(strPath))
  For Y = 0 To myBitmap.Height - 1
    For X = 0 To myBitmap.Width – 1
            objColor=myBitmap.GetPixel(X, Y)
            R=objColor.R
            G=objColor.G
            B=objColor.B
            objColor1=Color.FromArgb(R1, G1, B1)
            myBitmap.SetPixel(X, Y, objColor1)
    Next
  Next
  myBitmap.Save(Response.OutputStream,  _
 Imaging.ImageFormat.Jpeg)
 Response.Flush()
</script>
</body>Response.ContentType = "image/jpeg„
```

Listing 9. Direct access to the colors of a bitmap

This script is written also in Visual Basic language and included as ASP code into a web page. It uses system objects to open bitmaps from files. Object *myBitmap* is in fact a matrix of objects representing pixel colors. Thanks to methods *GetPixel* and *SetxPixel* a code writer can access every color component of each pixel of any bitmap.

Many kinds of scripting languages provide also useful basic functions, such as:

- read and write any file,
- split a string into substrings,
- convert a string to the array of characters,
- convert a character to integer and vice versa,
- get and set bit mask.

These operations are basic for steganography. When we have simple data which are not objects but values of a given type like string or integer, they can be directly used to hide additional bits of information.

```
iVal=Asc(cVal)
cVal=Chr(iVal)
iVal And bVal
iVal mod iMod
arrVal=Split(strVal,strSeparator)
strVal1=Mid(strVal,Starti,iOffset)
strVal1=Left(strVal,iLength)
strVal1=Right(strVal,iLength)
```

Listing 10. Operations on the string and integer values

Listing 10 contains basic scripting functions which can be run on basic data with types like string *strVal*, character *cVal* or integer value *iVal*. With these functions we can split text into words, or select a given part from a text or a word. Thanks to function *Asc* and *Chr* one can calculate the ASCI code of a given single character and vice versa. Apart from that it is possible to get or set bit masks of single integer values.

5. STEGANOGRAPHIC CAPACITY OF DOCUMENTS AND WEB PAGES

To estimate document steganographic capacity one has to find all document basic elements and investigate their data type. For a given data type we have specific relative steganographic capacity. Then the total capacity of a given document is the sum of unitary capacities of all basic document or page elements.

$$\Sigma_{iP=1}^{Prsnts} \Sigma_{is=1}^{Slides} \Sigma_{ish=1}^{Shapes} \Sigma_{iA=1}^{Attributes} (C(iA)*S\ (iA)) \quad (1)$$

$$\Sigma_{iD=1}^{Docs} (SP + ST + SF)$$

$$SP=\Sigma_{iP=1}^{Paragraphs} (C_{txt}*S\ (iP))$$

$$ST=\Sigma_{iT=1}^{Tables} \Sigma_{iC=1}^{Columns} \Sigma_{iR=1}^{Rows} (C(iC)*S\ (iC,iR))$$

$$SF=\Sigma_{iF=1}^{Frames} (C_{txt}*S\ (iF)) \quad (2)$$

$$\Sigma_{iW=1}^{Wrbks} \Sigma_{iWs=1}^{Worksheets} \Sigma_{iC=1}^{Columns} \Sigma_{iR=1}^{Rows} C(iC)*S(iC,iR) \quad (3)$$

Eq. (1) represents the calculation of the total steganographic capacity of a PowerPoint presentation. The presentation consists of slides, which contain shapes. As shown in the second chapter of this paper in Listing 5, each Shape object has several attributes. Each of them has individual capacity depending on the binary size of its value. Value *C* is the relative steganographic capacity of a given basic attribute *Ai* and *S* is for the size of this attribute value. For example for type color (0..255) typical estimation is 1/8, what means one bit per one byte.

Eq. (2) presents analogical calculation for Word documents and the last one, Eq. (3) represents calculation for Excel workbooks.

By analogy one can calculate steganographic capacity of a web page. The total capacity is equal to the sum of individual capacities of page elements. Eq. (4) describes the summarization of capacities of particular forms containing elements like buttons, text boxes or tables and images.

$$SItem=\Sigma_{iF=1}^{Forms} \Sigma_{iI=1}^{Items} \Sigma_{iA=1}^{Attributes} (C(iA)*S\ (iA))$$

$$STag=\Sigma_{iT=1}^{Tags} \Sigma_{iA=1}^{Attributes} (C(iA)*S\ (iA))$$

$$STab= \Sigma_{iTa=1}^{Tables} \Sigma_{iC=1}^{Columns} \Sigma_{iR=1}^{Rows} (C(iC)*S\ (iC,iR))$$

$$SImg= \Sigma_{iIm=1}^{Images} \Sigma_{iY=1}^{Height} \Sigma_{iX=1}^{Width} (1*/8) \quad (4)$$

$$Csteg \sim (SItem+STag+STab+SImg)/T_{refresh} \quad (5)$$

Eq. 5 represents the relation between total steganographic capacity of a web page and componential capacities of its elements and the frequency of page refreshing.

$T_{refresh}$ represents the time between two sequential refreshing moments. This frequency can be easily set by Java Script, but the time given for successful refresh depends on the size of page contents and on the network and server efficiency.

If we imagine, that during every refreshing we change page contents by adding bits of secret information then we have current secret data flow.

All types of single attributes of objects can be classified into the following families of similar types:

1) discrete set with law number of values,
2) size, average from 10x10 to 600x800 pixels,
3) color, average values range from 0 to 255,
4) text containing words and blank characters,
5) shapes with solid color,
6) typical images in RGB format.

Class of data types 1

The first type includes attributes such as text or header alignment. In such a value one can code little information. If there are four possible text alignments, we can code here two bits of information. Header or text size can have more possible values. Although font size can be set from 8 even to 72, typical range is from 8 to 16, so the text wouldn't look strange. If we assume, that every paragraph can have different font size than in such text we can hide number of paragraphs multiplied by 3 bits of information.

Class of data types 2

Parameters such as position or size are more flexible and less vulnerable to noticing in documents full of frames, shapes and images. If we assume that we can change position and size in every dimension by +/- 2 points than we obtain 4x4 values per each shape what gives 4 bits multiplied by the amount of all shape objects of hidden information size.

Class of data types 3

Any font or background color can be modified by +/-1 per 255 possible shades of grey, so the changes would be hardly to notice. It makes 1 bit per any attribute associated with sold color.

Class of data types 4

Objects containing text are very interesting for estimation of steganographic capacity. When we hide information in double spaces, then we have steganographic relative capacity equal to the number of words divided by the number of characters in the whole text. In this paper it is about 3200 words / 15700 characters with spaces what gives approximate capacity equal to 22%.

Class of data types 5

Solid shapes can have steganographic capacity comparable to parameters describing colors of fonts, background or characters. This is 1 bit per whole shape, what gives total capacity equal to the number of shapes multiplied by one bit or in other words number of shapes divided by 8 in bytes.

It is interesting, that in web pages we have not only object analogical to the elements of word documents, but also such pages can contain long streams of blank characters which are not visible on opened page. If we use tabulator, space and return caret it makes 3 combinations per each character of the invisible part of a html file. If we combine them into pairs it makes 9 values to encode in two sequential blank characters.

Class of data types 6

Finally, about the most interesting object for steganography; pictures and images. The simplest way to hide secret information in a bitmap is to change every pixel value in every color content by +/-1. This makes changes almost invisible and the relative capacity of image in this case is 1/8. If an image is full of contrasts, has very little smoothness, than we can dare to hide even 3 bits of information in every color ingredient, what gives the relative image capacity equal to 3/8. As human eye is the least sensitive to blue color changes, one can hide more bits in blue component than in red one.

Of course there are more sophisticated methods of hiding data in images, for example in Velvet transform or Fourier transform but this is beyond the scope of this paper.

6. CONCLUSIONS

In this paper I showed a new point of view on steganography. I presented a concept how to apply script programming and knowledge about the structure of various electronic documents to hide or detect hidden content. I drew attention to the observation, that script languages especially these running in Microsoft environment are easy to develop and can freely spread in every office and through networks.

I also showed a concept how easily we can calculate potential steganographic capacity of documents of various types. However it doesn't mean, that we can predict steganographic content, or that we will not make any mistake due to false alarm. This paper doesn't contain all possible methods of data hiding in files but this I leave for the recipients' imagination.

I must emphasis once again, that it is almost impossible to determine if a given document, full of blinking changing colors and advertisements, like most web pages or Power Point presentations, does contain any hidden additional bits of secret information or not.

7. REFERENCES

[1] http://www.microsoft.com/poland/technet/scriptcenter/.

[2] http://www.w3schools.com/VBscript/.

[3] http://www.tizag.com/vbscriptTutorial/.

[4] P. Wayner, **Disappearing Cryptography: Being and Nothingness on the Net**, 1996.

[5] M. Owens, **"**A Discussion of Covert Channels and Steganography**"**, **Sans InfoSec Reading Room**, 2002.