International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2020/193270 Volume 7, Issue 1, January – February (2020)

RESEARCH ARTICLE

Vulnerability Exploitations Using Steganography in PDF Files

Istteffanny Isloure Araujo
Intelligent Systems Research Center, London Metropolitan University, United Kingdom
i.araujo@londonmet.ac.uk

Hassan Kazemian
Intelligent Systems Research Center, London Metropolitan University, United Kingdom
h.kazemian@londonmet.ac.uk

Published online: 20 February 2020

Abstract – This article analyses the ways malicious executable files hides with Steganography on the most used files of our daily basis such as PDF, Word, Text, and Image. It demonstrates how data is hidden and gathers innovative ways of identifying potential attacks to prevent them by engaging the safety and exploitation of files distributed online. It is concerned with infected files that can have malicious executable applications embedded, executing itself upon the opening of the original file. Several experiments are detailed exploiting gaps in PDF, email and image files in order to draw awareness to security professionals and Ethical hackers' trainees.

Index Terms – Digital Attacks, Email Security, Ethical Hacking, PDF Security, Steganography.

1. INTRODUCTION

Steganography derives from Ancient Greek, merely meaning "information hiding", and used to secret camouflage messages during conflicts [1]. Today, Steganography applies to media database systems, digital content access control, data alteration protection, confidential communication and secret storage [2] within the text, video, audio and image files. This current research will involve identifying Steganography used to incorporate malicious executable files to the most used file extension, PDF (Portable Document Format); hence, we will study secret storage. We will learn more about PDF once we have the foundations for Steganography in the next chapter. There are different types of Steganography algorithms described below.

1.1. Pure Steganography

It is independent of a Stego Key, i.e., once the algorithm, is known, the data can also be extracted, carrying underlying security [3].

1.2. Private Key Steganography

It depends on a Symmetric Key, i.e., the same "password" hides and uncovers the secrecy, carrying moderate security [4].

1.3. Public Key Steganography

It uses distinct Stego Keys at each end. It has higher leverage of Security [5].

Computer criminals use steganography in order to secure illegal activities such as paedophilia, intellectual property breaches, viruses, harassment, pornography, hacking, fraud, gambling and criminal communications [6]. The process of identifying Steganography on documents and applications is called Steganalysis [7].

Between other Steganography and Cryptography differences, Steganography works on hiding the data, so it is not visible to a third party while Cryptography does not necessarily worry about hiding the data but about making it unreadable.

Steganography consists of hiding some content inside another, and Steganalysis is the process of identifying Steganography itself. It is important to note that Steganalysis does not reveal the hidden message identified [8]. To uncover the hidden text, we apply Cryptography which includes the studies of how to break mathematical algorithms within secret communications.

At first, we will be using PDF files as a case study to identify and stop possible bugs on PDF documents to protect data and avoid breaches of privacy, concentrating mainly on the three points below.

- Analysis of how malicious data is embedded in PDF.
- Identification of sections that could contain Steganography applications.
- Simple tools used to hide information on the background of a file.

Chapter 2 will detail more Steganography Principals, PDF files and the current methodology adopted in Chapter 3. Chapter 4 describe the main initial Steganography experiments accomplished for the research, while Chapter 5 summaries our findings, followed by all references used.

2. RELATED WORK

Steganography can improve security if used for protecting confidential data, but can also be used to breach security when the content is hidden is illegal and used to hide executable application that can steal somebody's password [9, 10] for this reason we will analyse how and where criminals can exploit this file format to draw attention and reinforce security on this specific areas.

2.1. PDF Files

The Portable Document Format used to present documents independent of applications since 1993 from which it evolved to version 9 by Adobe where permissions (Digital Rights Management), passwords and encryption (MD5, RC4: 40-bit and 128 -bit) have adopted and improved further on documents especially to comply with security weaknesses [11].

PDF documents contain some scripts programmed in JavaScript and XML to structure the document as well as PostScript for generating layout and graphics and its data compression technology [11]. Two different layouts are in place today for PDF, the linear and non-linear which consumes less space [12], and it can also contain audio for disabled people.

2.1.1. Basic Security on PDF

The security in place today on Adobe PDFs also explained by Application Level using SSl& Cookies, Network Level using Firewalls &SSL and Physical Level using Permissions & Monitoring. On the 25th March 2015, Adobe has made available a new security update especially to help to cope with the execution of code. The most common form of security on PDFs is encryption via public-key cryptography and digital signature for authentication. The passwords (owner or user) will restrict whether the document can print, copy, delete, import, export and modify [13], however depending on the strength of the password and encryption, hacking may occur [14].

2.1.2. Most Common PDF Vulnerability

The most discussed vulnerability of PDFs nowadays is the way it attaches on emails and webpages configured to start even if secure browsers and antivirus are in place and also the RC4 itself substituted with the modern AES-256 algorithm has also started to be used from PDF 1.7 version to avoid the weaknesses of password checking algorithm exploited by brute force attack previously [15]. On the 25th of this month, Adobe has made available a new security update especially to help to cope with the execution of code. Another known unexpected attack by PDF is when the application suddenly crashes, and the attacker takes control of the computer [16]. Guidelines to make users more precautious and one of the fixes is disabling JavaScript and not opening PDF from unknown sources which it is unfeasible nowadays.

2.1.3. PDF and Steganography Attacks

Steganography hides the virus on PDF by adding an image and Digital Watermarking to the file which carries and hides the virus [17]. We shall investigate some of the algorithms used to generate PDF and how the virus hides on this type of file in order to produce an application that can identify and avoid such practices [18]. The way the PDF document is structure has space for hiding code [19].

2.1.3.1. Known PDF Exploitation Steganographic Techniques and Identification Methods

2.1.3.1.1. Analysing the TJ Spaces

Apart from using the end of the objectives on the File Structure of PDF as per Figure 1, it uses the space between the characters to hide text. An example of this approach described in the following algorithm which uses the StegoPDf software to embed data, although very efficient as there is a limited amount of empty spaces we could use, and capacity is an issue.

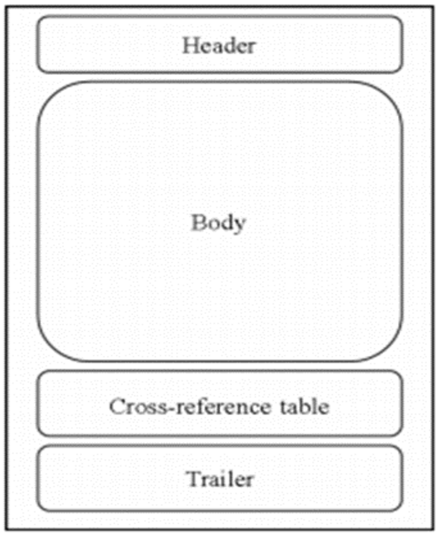


Figure 1. Structure of PDF [20]

2.1.3.1.2. Predicting & Extracting JavaScript (Extraction Algorithm)

Gathering JavaScript Unsecure Code to hide malicious programs. The PDF file has gaps within the code data, and malicious code can be embedded [21]. The figure below states how a simple PDF file formed. For every end of the object as in the figure above and in between objects, we can utilise the space the PDF language does not read to hide data [22]. Further demonstrated in Session 2.2.3.4, where we provide a sample PDF code with data hidden in between the objects with a JavaScript code embedded. The identification technique predicts and extracts JavaScript from the PDF code.

2.1.3.1.3. Statistical Analysis (Anomaly Based Detection)

Statistical Analysis is achieved using Mathematical Algorithms to encode data. A common way to detect Steganography is Steganalysis which investigates weather Steganography is present on a document. As an example to Statistical Analysis to embed and detect hidden data, we use Jackson [23] research which started to build what he has called a CIS (Computation Immune System) to detect hidden data and improve security on JPG files. Furthermore, this technique mathematically determines the number of occurrence of substitution encoded using pattern recognition, neural networks and generic algorithms mainly [23].

2.1.3.1.4. Dynamic Analysis (detectable pattern):

Dynamically analysing patterns of insecure data. This approach mainly uses intelligent software to analyse and detect malicious behaviour of a file, applying machine learning algorithms, including analysis of Digital Signatures, insecure loading, integration and component usage.

2.1.3.1.5. Analyse of Metadata

It investigates Metadata to discover hidden data. Metadata is the header of a file itself — it holders crucial information as to the location of the clusters of the file, logging and security settings. Steganography is made possible here by hiding information inside specific clusters [24].

2.1.3.1.6. Structural Analysis

It looks top to bottom to find hidden data. This technique uses of Forensics Tools such the EnCase to perform Steganalysis on the file. Freeware like Stegdetect will work just fine with JPEG file formats [25]. EnCase lets the examiner search through the files simultaneously and give the capability of reconstructing broken links and extracting hidden text found [26, 27].

2.1.3.1.7. Substitution of bytes using xor

It uses the xor algorithm to embed data. The xor algorithm swap a bit unit to another to embed a secret message. The advantage of this is that the memory used will remain the same as it is a simple swap, while the disadvantages relate to its simplicity, making it easier for computers to undo the algorithm and find the data hidden. The following illustration demonstrates the XOR substitution algorithm.

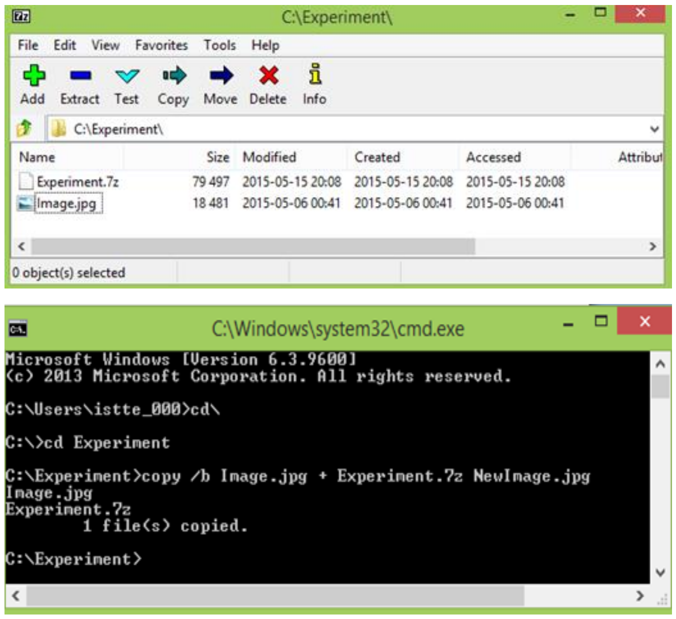
$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x+y \\ y \end{pmatrix}.$$

The sequence of operations is then expressed as:

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

2.1.3.1.8. Compress so more comfortable to hide

It compresses the document and embeds data. Compressing data has proven to make files easier to hide using a compressing software like 7 Zip Portable, which is freely available. Files can hide inside a JPEG image per example, as shown in Figure 2, only by typing one line of code on Command Prompt.


C:\Experiment\
File Edit View Favorites Tools Help
Add Extract Test Copy Move Delete Info
C:\Experiment\
0 object(s) selected
C:\Windows\system32\cmd.exe

Name	Size	Modified	Created	Accessed	Attribut
Experiment.7z	79 497	2015-05-15 20:08	2015-05-15 20:08	2015-05-15 20:08	
Image.jpg	18 481	2015-05-06 00:41	2015-05-06 00:41	2015-05-06 00:41	

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\istte_000>cd\

C:\>cd Experiment

C:\Experiment>copy /b Image.jpg + Experiment.7z NewImage.jpg
Image.jpg
Experiment.7z
        1 file(s) copied.

C:\Experiment>
```

Figure 2. Hiding Files inside an Image Using Compression

2.1.3.2. Technologies used on PDF Files

PDF Files can contain several technologies embedded such as PGP, JavaScript, XML, HTML, SOAP, Compression and Encryption. Pretty Good Privacy is used on PDF to secure data; JavaScript is added to PDF files mainly to create forms and interactive data [28]. Malicious coders can exploit this capability to send a virus to victims' computers to as an example eavesdrop sensitive data.

XML is the extensible mark-up language and provides the PDF document with the linkage with the web. The FDF was an XML adapted language to use within PDF files; however, new updates of PDF use plain XML to combine form data sent to a server. HTML and the improved XHTML has been combined and became XForms used on PDF, which is defined from XML as well [29]. SOAP is also XML derived, and its structure highlighted in Figure 3 is very similar to the PDF one.

Compression reduces the size of the PDF file and adjusts it to previous versions of PDF software, making makes it easier to email documents, especially with multimedia applications embedded [31]. PDF files can be encrypted, MD5 hashes can verify the file. Files can be password protected; however, there is current software that can remove the applied security of PDF files easily. This topic needs further research, and it analysed in more detail as the research progress.

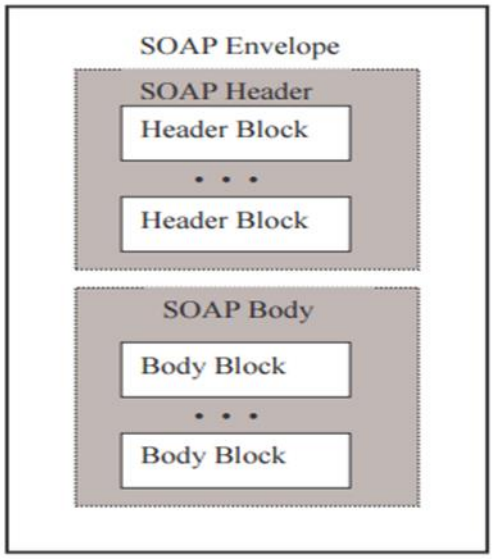


Figure 3. SOAP Structure (Adapted from Walker & Thomas [30]).

2.1.3.3. Identified Software to combine Steganography and PDF Files

Other examples of Steganography software used on specific file formats in files are JPHide, OutGuess, F5, AppendX, Camouflage and Tripwire for servers (detect and alert of changes).

2.1.3.4. Initial PDF Source Code with Hidden Message

The programming code 1 is the first attempt to create a PDF file from scratch with hidden text and code placed strategically at the End of Object and End of File where the document supposed to ignore.

```
%PDF
obj
  >>
endobj
//Malicious Code --> PHP Code hidden at the End of Object.
   (Supposedly an ordinary PDF) Tj
end stream
endobj
//Keylogger Code -> JavaScript hidden at End of Object.
%%EOF
Again, the secret message  - > Hidden Message after the End of File where compiler ignores it.
```

Programming Code 1 Writing a PDF

3. PROPOSED MODELLING

The chosen methodology used a way to prevent the exploitation of PDF files using Steganography. The experimental studying on Steganography broadens the knowledge on security vulnerabilities of PDF files and email systems used to distribute such files. We provided details of how to develop ways of preventing malicious applications interception after completing a set of experiments, using RUP/USDP methodology for the developing of any application that born from this study. PDF files are commonly available, accepted and distributed in different OSs and mobile devices, therefore, to detect current vulnerabilities to improve security on a day to day file format together with email systems used to carry it is essential. For this reason, understanding PDF vulnerabilities it is the best way to avoid attacks.

The Methodology to develop the “framework” does not necessarily need to be RUP/USDP. Depending on the solution to the problem, we may use another methodology such as DSDM for more dynamic development; however, we shall use Java derived languages such as JavaScript, JSP or Python to build the security application as these are the most commonly used and available languages, so the focus is “How to combat the PDF virus Interception by Criminals Motivated on stealing Personal Data? Can Steganography benefit such an activity?”.

For accomplishing this goal, we have started intense Literature Research focusing on the most recent papers and articles but noting down what has been considered in the past to overcome previous vulnerabilities. The research question is broaden at present, and this as well tends to be more specific once we have gained more knowledge of the subject area in terms of what other researchers studying the same area are exploiting.

Another point we must make here is the fact that we will be dealing with investigation and gathering of sensitive information from which we do not aim to use our own private devices and not experimenting on the general public; therefore, we must make clearer that our method of gathering data will not analyse sensitive data from people not involved with this research nor people indirectly involved and principal readers.

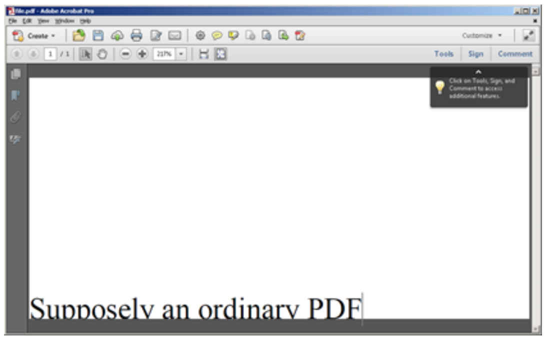
4. RESULTS AND DISCUSSIONS

The initial results accomplished in our research described below. We started by learning how to code a PDF from scratch and where messages and applications hides, then we have identified the most straightforward way to embed files on PDF using free software commonly available. We finalised by accomplishing what the software vendors tagged as not possible, by identifying our first vulnerability on PDF documents and emails used to distribute the file.

4.1. Experiments Explained

4.1.1. Hidden Text on PDF Code directly on the code

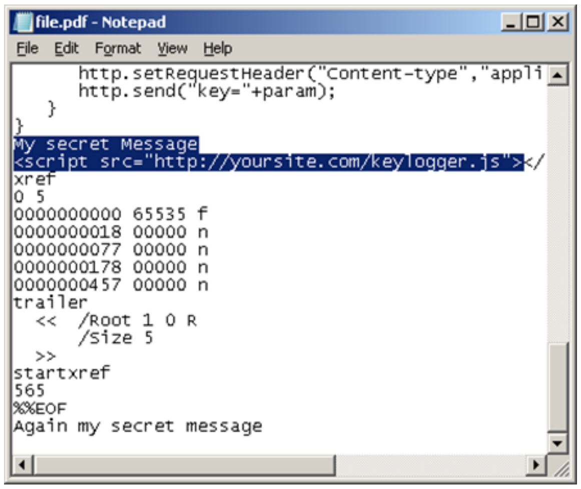
The first experiment we made was how to hide a message on PDF documents. We have used the PDF language to write down the PDF document from scratch, and we utilised the technique of hiding data after the end of files and end of an object.


file.pdf - Adobe Acrobat Pro
Create
Tools
Sign
Comment
Click on Tools, Sign, and Comment to access additional features.

Supposely an ordinary PDF

Figure 4. PDF Embedding Hidden Message

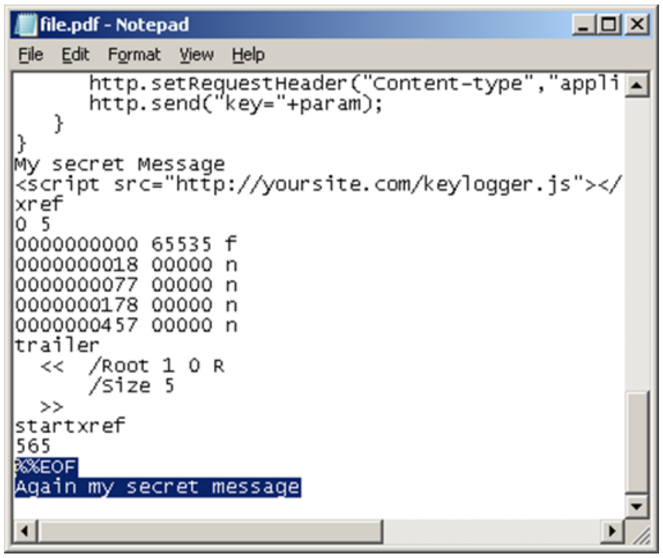
Figure 4 shows us a PDF File with hidden messages on it, create inside the code for the document. Note that there is no attachment for this document as the hidden message in on the code itself. In the PDF code, we can also embed JavaScript programs to execute when the PDF is open, but now current experiments on this did not work for us, as it differs from PDF versions and other resources.



```
      http.setRequestHeader("Content-type","appli
      http.send("key="+param);
   }
}
My secret Message
<script src="http://yoursite.com/keylogger.js"></
xref
0 5
0000000000 65535 f
0000000018 00000 n
0000000077 00000 n
0000000178 00000 n
0000000457 00000 n
trailer
  <<  /Root 1 0 R
      /Size 5
  >>
startxref
565
%%EOF
Again my secret message
```

Figure 5. Hidden Data at the End of an Object

Opening the file with a text editor, we can provide two examples of where we can hide data as per Figures 5 and 6. The message is hidden at the end of a random object on the PDF code while below, we have hidden the message after the End of File (EOF) which are place holders the PDF language ignore and people who are viewing the PDF cannot see the message hidden.



```
      http.setRequestHeader("Content-type","appli
      http.send("key="+param);
   }
}
My secret Message
<script src="http://yoursite.com/keylogger.js"></
xref
0 5
0000000000 65535 f
0000000018 00000 n
0000000077 00000 n
0000000178 00000 n
0000000457 00000 n
trailer
  <<  /Root 1 0 R
      /Size 5
  >>
startxref
565
%%EOF
Again my secret message
```

Figure 6. Hidden Message at the End of the File

4.1.2. Hidden Word and other file on PDF

With specific software, we can easily hide another file on PDF, per example, a word file with .doc extension. We have accomplished this using the software FoxitPhantomPDF presented in Figure 7, which lets us attach files to PDF as well as make it hidden. Plus, using the latest version of FoxitPhantomPDF, we can easily attach a file by clicking on the attachment icon on the left and selecting the desired file from any archives.

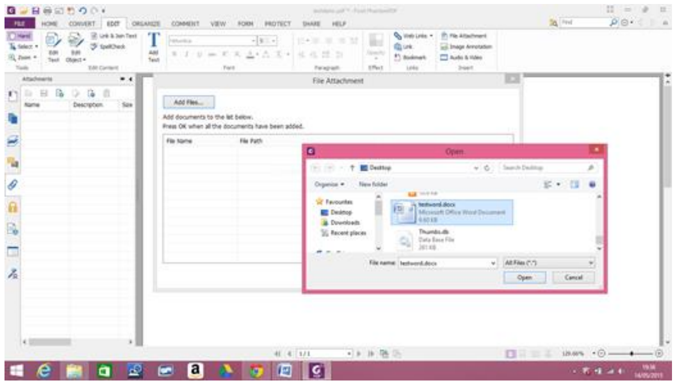

File Attachment
Add Files...
Add documents to the list below.
Press OK when all the documents have been added.
File Name
File Path
Open
Desktop
testword.docx
Open
Cancel

Figure 7. Embedding a Word File inside a PDF

After adding the file, it shows on the attached menu, but it hides the document by clicking on the attached icon, to hide the window. This way at first, the user does not know the PDF has a word document embedded on it. We can also hide an image file on PDF with the same software described above, by attaching the file and making it hidden using Fox PhantomPDF in the same way we did for the word file with the ".doc" extension. Current versions of PDF do not allow us to embed executable files on PDF. The security team are working daily to identify and eradicate any security gaps. If this is attempted, a pop up will alert the user. Adding an application to PDF and making it execute on the victim's computer was a well-known type of virus that current versions of PDFs tried to abolish. For this reason, due to security issues, the feature is disabled hence we have tested if we could potentially embed our "malicious" application to other file types and then add it to the PDF document.

4.1.3. Hidden Executable File on Image, Could not Email Initially. Then used Compression Techniques

In this experiment, we have hidden an executable file inside an image by playing with the properties of the file. In this case, we create a shortcut for the file and change its properties to make it execute an application every time the image is open, pointing the original file to an executable one like in the example below:

C:\WINDOWS\System32\cmd.exe/c filename

For the executable file, we would mask it to look like an image file by changing its shortcut image to:

%SystemRoot%\System32\SHELL32.dll

Making sure the executable file is not visible with the following Programming Code 2:

```
If (this.rawValue ==0){
Xfa resolve Note (“Page4”).
Presence= “hidden”;
} else{
Xfa.resolveNote (“Page4”).presence= “visible”;
}
```

Programming Code 2. Hiding Executable File Script.

With the above experiment, we were unable to send the file via email as the email has identified the executable file even though; we have experimented sending the image inside a ZIP file.

As per Figure 8, we have pointed TestFile.exe to execute when opening anything.jpg and then hidden TestFile.exe and zipped both inside a zip file to see if we could email it. However, using the previous experiment from Session 2.2.3.1 under unit 8 demonstrated with 7 Zip software using compression techniques, we managed to successfully email the "NewImage" file with hidden contents inside illustrated in Figure 9.

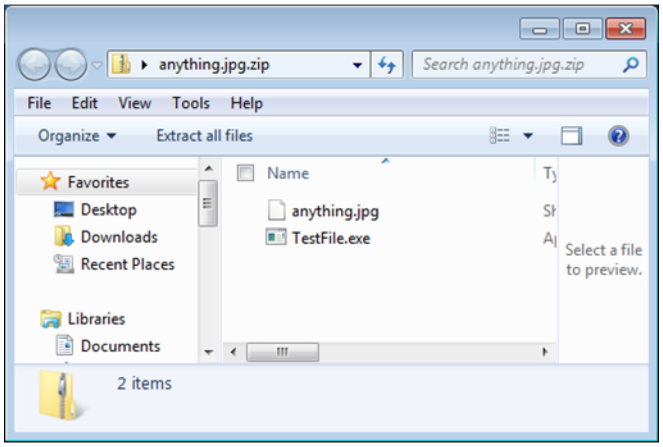
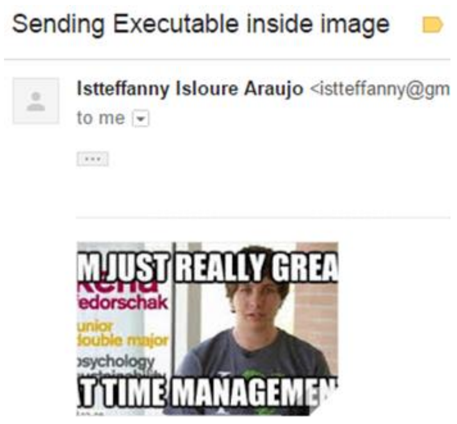


Figure 8. Masking an Executable inside an Image


Sending Executable inside image

Istteffanny Isloure Araujo <istteffany@gm
to me

Figure 9. JPEG Filed Emailed with Executable and Other Files

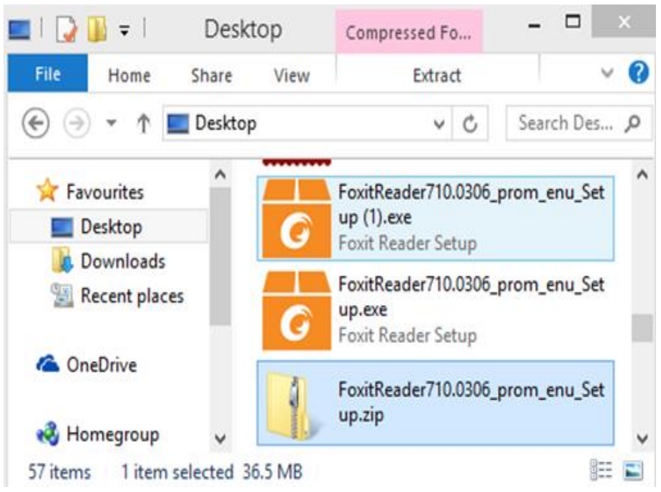
4.1.4. Simulated simple email on PHP with attachments

After trying to email the hidden executable file with emails and it did not allow us, another experiment was to code the email in PHP to see if it would be possible to attach an executable file and send to an existing email using this method. It is easy to code and send an email with text only, but executable attachments included is a step further for criminals and information security professionals to identify and stop.

4.1.5. Hidden Exec on zipping then on PDF then sent by email

The last experiment we accomplished and managed to achieve a good result was using the Foxit PhantomPDF as well, we have found out that if we ZIP an executable file first, we can easily attach to the PDF document and we can email without any problems or change of settings. Since we are emailing a PDF file extension, emails do not recognise that there is an attached ZIP file with an embedded EXE file shown in Figure 10, finding no harms.

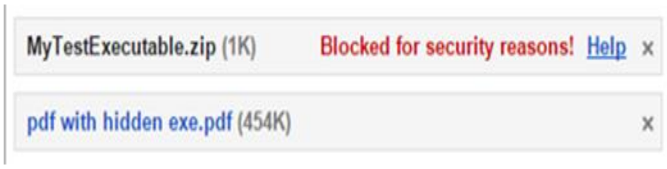
We started by zipping our executable file. We can potentially change the name of the file to make it less suspicious in case of the victim gain visibility of it, but in this example, we have kept it the original zip name. After zipping the executable file, it embeds in the PDF document. At the Appendix of this report, we can note down an email chat we had with the software representatives which reassured that it is not possible to embed an executable file on PDF using the software, but we have proven otherwise.



FoxitReader710.0306_prom_enu_Setup (1).exe
Foxit Reader Setup
FoxitReader710.0306_prom_enu_Setup.exe
Foxit Reader Setup
FoxitReader710.0306_prom_enu_Setup.zip
57 items 1 item selected 36.5 MB

Figure 10. Zipped Executable File

The file can be seen in the attachment tab, proving we can still embed an executable file to a PDF regardless of its security updates. Again, we can hide the embedded files by clicking on the attachment icon. Testing if we can email such file using Gmail email system, we can be disappointed to know that if the file is too big, we will not be able to send it, example, if it exceeds 25 megabytes. Therefore, we came to one disadvantage that the size of the file can make it suspicious and to send it by email, we will need to either zip the PDF with the Zip file containing the "Malicious" application or make sure the attached application is not too big as the previous attached application was 36Mb.

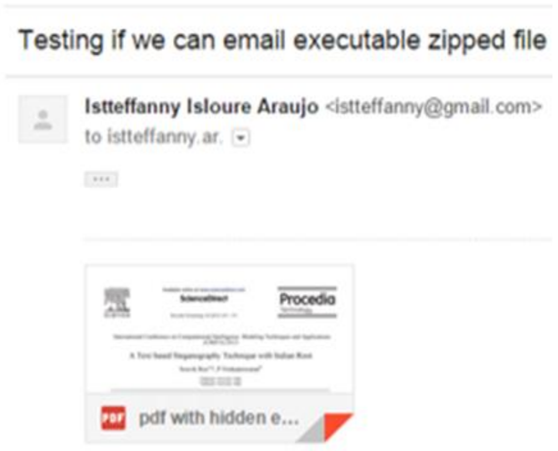


MyTestExecutable.zip (1K) Blocked for security reasons! Help ×
pdf with hidden exe.pdf (454K) ×

Figure 11. Attached PDF file with Hidden Executable to Email

Another point to make here is that the PDF itself which we have coded from scratch had one word on the body of the document, so it was not significant. Hence, we have simulated with an executable file containing the Python Key logger code. Our executable file is now called MyTestExecutable.exe, and it is relatively small and easy attached inside a zip file to the PDF. We have tested with an ordinary PDF now which would not raise any suspicion. The zip file with the executable application inside cannot be emailed by itself as the email system blocks it as seen in Figure 11. However, we will be using the most acceptable and straightforward document file to carry this executable application and be able to email it.

We were able to bypass the PDF and Gmail security by finally sending the PDF with executable code as per Figure 12.



Testing if we can email executable zipped file
Istteffanny Isloure Araujo <istteffanny@gmail.com>
to istteffany.ar.
pdf with hidden e...

Figure 12. "Malicious" Application Sent by Email inside PDF Document After Latest Security Updates to PDFs and Emails

Using our current technique and software, anyone can accomplish the same results. We have not yet read about someone using the same technique to distribute the virus with the latest security updates on PDF files and email systems. The next step is to research on how to make the executable file to execute by extracting the application from the zip file inside the PDF file. We plan to make this by coding using JavaScript as it is possible to add JavaScript to PDF files using Foxit PhantomPDF quickly. It works in conjunction with the JavaScript to open a text file and record keystrokes from which the following code referenced on WebPages:

```
<script src="path to code.js">
```

5. CONCLUSION

This paper demonstrated how important it is for research professionals to work on prevention of digital crimes which are challenging to identify without any background measures implemented by IT Security professionals and Ethical Hackers. We shall prevent such criminal activity successfully, protecting our data and sensitive information once we know what the gaps in the current system are. In order to achieve the initial experiments successfully, we have noted down fundamentals of Steganography and concentrated on PDF and its capabilities as our first case study. For this reason, we started to analyse different ways data hidden in commonly used files, such as PDF and JPG. We have already managed to hide text on PDF, as well as to attach documents to it, including a zipped executable file. As per PDF security updates, we shall not be able to attach executable files into it for security reasons; however, our research has proven that simple tricks can make this possible as per our initial experiments. We can see that there are gaps in the code where code is added without displaying the contents inside the PDF document. These gaps are present at the end of the object from the PDF code and the end of the file. Sample relevant codes analysed here and the current adopted methodology was also present to reinforce how we experimented with the data and achieved results. Further work includes continuing to analyse and research different ways of how to prevent the executable file hidden on the PDF/JPG document to execute automatically and how to stop eavesdropping information, such as passwords so the security techniques can continuously mitigate any such attack from victim's computer.

REFERENCES

[1] Kessler, G. And Hosmer, C. (2011). Chapter 2 – An Overview of Steganography. Available: http://www.sciencedirect.com/science/article/pii/B9780123855107000023. Last accessed 16th Feb 2020.
[2] Kawaguchi, E. (2015). Applications of Steganography. Available: http://datahide.org/BPCSe/applications-e.html. Last accessed 16th Feb 2020.
[3] Kwon, T. (2011). Detecting and Analyzing Insecure Component Integration. Computer Science. 1 (1), p1-146.
[4] Ahn, L. And Hopper, N. (2012). Public-Key Steganography.Computer Science Dep. 1 (1), p1-18.
[5] Al-Ani, Z et al. (2010). Overview: Main Fundamentals for Steganography. Available: http://arxiv.org/ftp/arxiv/papers/1003/1003.4086.pdf. Last accessed 16th Feb 2020.
[6] Judge, J. (2001). Steganography: Past, Present, Future. SANS Institute InfoSec Reading Room. 2001 (1.2f), 20.
[7] Indika. (2011). Difference Between Cryptography and Steganography.Available: http://www.differencebetween.com/difference-between-cryptography-and-vs-steganography/. Last accessed 16th Feb 2020.
[8] Wayner, P (2009). Disappearing Cryptography. Information Hiding and Watermarking. 3rd ed. Burlington: Elsevier. p337-353.
[9] Camilleri, K. (2011). A Steganographic Framework: Information hiding in the Spatial Domain using Digital Images. Available: http://thesis.klauscamilleri.com/. Last accessed 16th Feb 2020.
[10] Zaidoon, K. et al. (2010). Main Fundamentals for Steganography.Journal of Computing. 3 (3), p158-163.
[11] Adobe (2006). PDF Reference, sixth edition. Available: http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/pdf_reference_1-7.pdf. Last accessed 16th Feb 2020.
[12] Roebuck, K. (2012). Electronic Documents: High-impact Strategies - What to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors. London: Emereo Publishing. p60-76.
[13] Fletcher, A. (2009). PHP: Sending Email (Text?HTML/Attachments). Available: http://webcheatsheet.com/php/send_email_texthtml_attachment.php. Last accessed 15th May 2015.
[14] Adobe (2008). Document management – Portable document format-Part 1: PDF 1.7. California: Adobe Systems Incorporated. P45-62.
[15] Stevens, D. (2011). Malicious PDF Documents Explained. Security & Privacy, IEEE. 9 (1), p80-82.
[16] Adobe. (2014). PDF Reference and Adobe Extensions to the PDF Specification. Available: http://www.adobe.com/devnet/pdf/pdf_reference.html. Last accessed 16th Feb 2020.
[17] Borders, K. (2013). Steganography in PDF Files. Available: http://stackoverflow.com/questions/16111471/steganography-in-pdf-files. Last accessed 16th Feb 2020.
[18] Stolfo, S. et al. (2013). Research in Attacks, Intrusions, and Defenses. New York: Portland State University. p204-223.
[19] Srndic, N. And Laskov, P. (2013). Detection of Malicious PDF Files Based on Hierarchical Document Structure. Internet Society. 1 (1), p1-18.
[20] Lai, Yin. And Tsai, Wen. (2013). Covert Communication Via Pdf Files By New Data Hiding Techniques. National Chiao Tung University Journal. 1 (1), p1-6.
[21] FreeMyPDF. (2015). Removing Passwords and Restrictions from PDF. Available: http://freemypdf.com/. Last accessed 16th Feb 2020.
[22] W3C. (2012). PDF Techniques for WCAG 2.0. Available: http://www.w3.org/TR/WCAG20-TECHS/pdf.html. Last accessed 16th Feb 2020.
[23] Jackson, J. et al. (2003). Blind Steganography Detection Using a Computational Immune System: A Work in Progress. International Journal of Digital Evidence. 4 (1), p1-19.
[24] PDFlib. (2012). Extensible Metadata Platform (XMP). Available: http://www.pdflib.com/knowledge-base/xmp-metadata/. Last accessed 16th Feb 2020.
[25] Richer, P. (2003). Steganalysis: Detecting hidden information with computer forensic analysis. SANS Institute InfoSec Reading Room. 1 (1.4b), p1-13.
[26] Partington, T. (2007). Computer Forensics: Final Report. Software Engineering. 1 (1), p1-46
[27] Wee, C. (2014). Analysis of hidden data in NTFS file system . Edith Cowan University Journal. 1 (1), p1-21.
[28] Wikipedea. (2015). Steganography tools. Available: http://en.wikipedia.org/wiki/Steganography_tools. Last accessed 16th Feb 2020.
[29] Rosenthol, L. (2001). Using XML and PDF Together, why do not necessarily have to choose. Available: http://www.planetpdf.com/planetpdf/pdfs/pdf2k/01W/rosenthol_xmlpdf.pdf. Last accessed 16th Feb 2020.
[30] Walker, F. And Thoma, G. (2007). A SOAP-Based Tool for User Feedback and Analysis. National Library of Medicine. 1 (1), p1-10
[31] Zhong, S. et al. (2007). Data Hiding in a Kind of PDF Texts for Secret Communication. International Journal of Network Security. 4 (1), p17-23.

Authors

Istteffanny Isloure Araujo received the B.S.c and M.S.c degrees in Computer Science and Computer Forensics and IT Security from London Metropolitan University in 2013 and 2015, respectively. Now, she is in the Intelligent Systems and Research Centre of the School of Computing and Digital Media. Her subject area is Big Data, Digital Security, Copyright and Privacy using Steganography. She also is an Associate Lecturer in subjects like Databases, Fundamentals of Computing, Programming, Network and Cloud Security, Networks and Operating Systems, Game Development, Cybersecurity Fundamentals and Ethical Hacking. Istteffanny has published two papers entitled "Protecting against eavesdropping on Mobile Phones to snip data with Information Security Awareness and Steganography principles " and "Enhancement of Capacity, Detectability and Distortion of BMP, GIF and JPEG images with Distributed Steganography" and has other ones reviewing.

RESEARCH ARTICLE

Dr Hassan Kazemian received a B.Sc. in Engineering from Oxford Brookes University, UK in 1985. He received an M.Sc. in Control Systems Engineering from the University of East London, UK in 1987. He followed with a PhD in Learning Fuzzy Controllers from Queen Mary University of London, UK, in 1998. He is currently a professor at London Metropolitan University. He worked for Ravensbourne College University Sector, UK, as a senior lecturer for eight years. Previous lecturing experience includes the University of East London, UK, University of Northampton, UK, and Newham College, UK. Research interests include AI and ML applications to cybersecurity. Prof. Kazemian is a Fellow of the Institution of Engineering and Technology FIET (formerly IEE) UK, Chartered Engineer (C.Eng.) UK, and Fellow of British Computing Society (BCS) UK.