IJOCIT
ISBN: 2345-3877
www.ijocit.org
Volume 4, Issue 3

Review Paper

Detection of Suspicious PDF Document-Embedded Code

Er. Gurjot Singh [1*]

Received: 13 Jun 2016
Accepted: 19 Aug 2016

Abstract
In this era of Internet and information technology, our data does not remain secure; our computer systems are not fully secured from critical attacks as information is lying in public networks. E-mail has become increasingly popular and virus spreading via email is also increasing. The PDF documents, EXE programs that are attached as e-mail attachment can spread viruses from one system to another computer system. PDF documents are popularly used for reading or sharing purposes because of its portability, easiness, and reliably independently of the environment in which they were created and the attackers are using malicious PDF documents to harm computer security in the present era. Advance antivirus is not proving effective against this kind of threat. In this paper, I opted different techniques to detect the embedded suspicious code in the PDF document and also analyzed the file structure, document structure and objects. I applied peepdf and pdf-parser, the pdf document tools to analyze the viruses and embedded code in malicious pdf documents.

Keywords: Malicious Pdf Document, Virus, Malware Analysis, Pdf, and Pdf-Parser.

Citation: Singh, E. G. (2016). Detection of Suspicious PDF Document- Embedded Code, *Int. J. of Comp. & Info. Tech. (IJOCIT)*, 4(3): 71-78.

1 Department of Computer Science & IT, Malwa College, Samrala, Punjab, India
* Corresponding Author: gs.genconian@gmail.com

1. Introduction

In recent years, the use of Portable Document Format (PDF) file was increasing very fast especially since most of researchers share, exchange and publish their works by using PDF as a medium. These researchers has increased the range of PDF use from research field to finalized document such as meeting minutes, proposals, electronic books and achieve document are in PDF [1, 2]. The reason behind PDF Document's popularity is its reliability and independent of the environment in which they were created. Due to extreme attention and various advantages that PDF document provides, they are now become the main medium for the attacker and malware writers for spreading evil contents. Attackers start to target any programs that become popular enough to dominate, as this would raise the number of potential victims. The dominant programs may include software like PDF viewers, certain web browsers, Adobe Flash player, etc [3].

Form few years ago, the malware authors use PDF files to deliver malware in machines as the executables files are blocked to deliver as e-mail attachments. Not only executable, but also non-executables can be made malicious. Once the vulnerability is found inside a program, attackers try to utilize this vulnerability in order to execute malicious code on the victim's computer [3]. A successful exploitation of the program's vulnerability can result in the host computer getting infected with a malicious program.

Attackers have also become smarter and many countermeasures established by software houses such as Adobe are now bypassed. The malware authors exploit vulnerabilities in Adobe reader software that can execute arbitrary code on particular machines. The PDF language depends on PostScript language [1]. The popular things to do with malicious PDF files are to download the backdoors from command and control servers and then execute them. Some files have the ability to steal critical information from the victim`s machine.

Problem: In today`s era, majority of organizations and other authorized authorities opt PDF documents to exchange their critical information among themselves so relies on it is a great weakness as the attackers change the structure of that PDF files and add malicious contents into them. When the victims open that malicious PDF document then that malicious code is being activated and spoiled the system of that particular victim. So it is a great need to first analyze the PDF documents for malicious code if any present in it or not.

2. Structure of PDF Document

2.1 PDF file's Contents

A PDF file consists of the mainly four parts that are described as follow:

Header

The header of file specifies the PDF version to which the file conforms. It has one line which is mandatory but can also have the second comment lines which include a sequence of non-Printable characters [5, 9].

Body

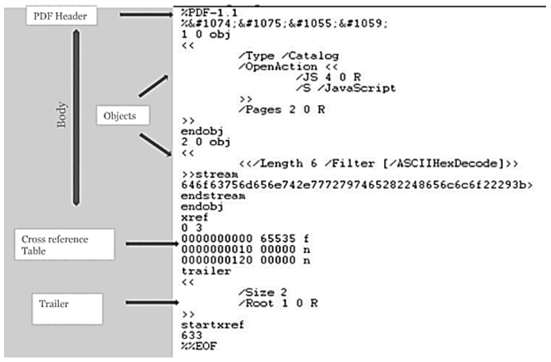
The next part of structure is body which has objects. Objects of pdf file include images, text streams, and multimedia files. The body section contains all the data which is being shown by the user.

Cross-reference table

The third section called Cross-reference table contains the exact address of all the objects in the PDF document i.e. the reference to all the objects in the file as shown in fig 1. The principle of a cross reference table is that it permits random access to objects in the file. The location of object is defined by a byte offset and offset tells the number of bytes between the start of the document and the start of the object description. The objects in the Cross-reference table contain 'in use' or 'free' objects. The 'Free' objects is obsolete and should not be used. But it can be reactivated.

Trailer

Last section is Trailer which placed at the end of the PDF file. This is generally opted to store the location of the cross-reference table and point to the root object. It basically tells how the applications reading the PDF document should find the cross reference table [9].



```
%PDF-1.1
%&#1074;&#1075;&#1055;&#1059;
1 0 obj
<<
        /Type /Catalog
        /OpenAction <<
                /JS 4 0 R
                /S /JavaScript
        >>
        /Pages 2 0 R
>>
endobj
2 0 obj
<<
        <</Length 6 /Filter [/ASCIIHexDecode]>>
>>stream
646f63756d656e742e7772797465282248656c6c6f22293b>
endstream
endobj
xref
0 3
0000000000 65535 f
0000000010 00000 n
0000000120 00000 n
trailer
<<
        /Size 2
        /Root 1 0 R
>>
startxref
633
%%EOF
```

Figure 1. Structure of PDF document

2.2. PDF Objects

Objects, are the basic building block of PDF file, are divided into two categories these are direct and indirect categories. Direct objects are just "inline values". For example: /Filter /Flate Decode. Here the key is Filter with a direct name value of Flate Decode. Indirect objects are referred by other objects and they have an object ID and a generation number. For example: /Contents 2 0 R. The key is Contents and 2 0 R is an indirect reference to a contents stream or a contents array. Basically a PDF document has eight (8) types of objects called "CosObjects" [6].

Boolean objects

Boolean objects are identified by only two values ('*true*' and '*false*'.)

Numbers

Numbers have two categories in PDF these are integer and real. Integers may have signed integers or unsigned integers. Real can be in decimal format.

Strings objects

String objects are stored using literals and are specified using parentheses '(' and ')' or if they include hexadecimal numbers then these are enclosed in angle brackets '<' and '>' as a sequence. For Example: (Testing), <1A3D5C>.

Names objects

A name is a uniquely defined by sequence of characters, which must be preceded by a slash (/). Whitespace and certain delimiter characters cannot be used within names, but these limitations can be bypassed by representing such characters using their corresponding hexadecimal code.

Array objects

An array is a one-dimensional collection of heterogeneous objects arranged sequentially i.e. an array may be made up of any combination of object types. Arrays are always enclosed in square brackets. For example: [0 0 612 792].

Dictionaries objects

A dictionary works as a lookup table in which entries are specified as key and / value pairs. In each entry, first value is defined as key and second is pair. A dictionary is represented by two left angle brackets (<<), followed by a sequence of key–value pairs, followed by two right angle brackets (>>). For example:

```
<< /Type /Example /Key2 12 /Key3 (a string) >>
```

Stream objects

This is a special dictionary object between the stream and endstream. It stores both, the data and dictionary information.

It contains stream data, such as images, script code, text, and compressed it by using the special filters [8].

Null object

Another category of object which is also valid in PDF is NULL Object and is represented by a keyword NULL [17].

2.3. Document Structure

The document structure describes how the objects are arranged within the body of a PDF file and how the several parts of the document are represented by using these objects.

The description of PDF file can be given by the hierarchy of the objects contained in the body. Objects which are mostly used in PDF document are dictionaries. The file usually is dictionaries [5, 18].

Catalog

The Catalog is the first parent node in the PDF document i.e. root node, and it control the whole PDF files. "The Catalog is a dictionary that is the root node of specific document" [5]. This section is used for referencing other objects (the tree of pages contained in the document, objects representing the document's outline, the document's article threads, and the list of named destinations) which describes the document's contents. It also has information regarding how the document will be displayed on the screen.

Pages

The pages of PDF document are only accessible through a tree called page tree which defines all the pages in document and also the ordering of pages [5]. A document containing thousands of pages can be easily opened by using page tree. This tree contains nodes which represent pages of document, pages are of two types: intermediate and leaf nodes. Intermediate nodes are also called page tree nodes, whereas the leaf nodes are called page objects. The simplest page tree structure can consist of a single page tree node that references the entire page object directly (so the entire page objects are leafs). Each page of the tree has two properties of its own, like the Image able content, Thumbnail and Annotation.

Thumbnails

PDF documents include thumbnails of its pages which is not required. It uses thumb values as the page objects and it does not has type, subtype and name key.[9] *Annotations:* Annotations are other objector objects which are connected with a page but are separate from the page description itself. *Annotations* have various types like: Text notes, Hypertext links moves and sounds [9].

Outline tree

The Outline tree is similar to the design of the document. It holds the relationship between the structure of this document and the parent's nodes and the child nodes. Outline allows a user to access views of a document by name. Outline provides a new view based on the destination description with activation of an outline entry, a link annotation, also called a bookmark. It is accessed from the Outlines key in the Catalog object [5, 9].

Article threads

A PDF document may include one or more article threads. Each thread has its name and elements User can select any pages of which pages he/she want to read, instead of from one page to next page.

- Named Destinations

A destination can be specified in a document when we use an annotation or outline entry. The destinations include a page, location of page display on window. It can be described explicitly by arrays or implicitly. A destination may be represented explicitly as an array or implicitly by using a name or a string .Both of string and a name are used as named destinations [7]. These are effectively useful when the destination is in another file [7, 9].

3. Security Analysis

There are numerous ways for embedding data within a PDF document. Although the content stream may refer to them through *annotations,* embedded objects are not part of the document's content stream. It is possible to embed the links, movies, sounds or file attachments through the annotation. It is also possible to embed JavaScript code [9].

3.1. Malicious techniques used in PDF file

Embedding application or files

The PDF format allows embedding of files in to documents, such as flash application, font application or JavaScript; they also are accessible from the PDF. This feature also used by malware operators, for example disguise malicious file and additional actions. It means that when user opening the PDF file, the *Adobe Reader* can also show the flash application directly or any type of file can be embedded; it is also possible to embed viruses, worms and other malicious code.

3.2. Exploitation of Vulnerabilities

This means that attackers execute shell code with privileges of reader's process. PDF exploits contain two parts:

1. JavaScript-based
2. non-JavaScript-based also called Flash-based.

JavaScript-based method is more familiar and popular, because JavaScript-based PDF malware is usually text only, it is easier to pass the security control.

Flash-based is embedding the flash files in the PDF file. PDF provides several ways for inclusion of JavaScript code. These mechanisms are important for the realization of interactive features, such as forms, dynamic content or 3D rendering [6].

There are basically two types of JavaScript code that can be used: one is that, along with the code to exploit the vulnerability, includes the payload used for the attack, and other one relies on other objects in the file or external malicious [4].

4. Suspicious PDF Document Analysis tools

As the academic tools specifically related to PDF security are not much more. Most of the available tools detect many types of malware at the same time, and some include PDF as well.

4.1. CWSandbox

This tool has represented an important improvement in malware analysis. CWSandbox[11] sophisticated platform is able to extract the dynamic behaviour of a computer system once a certain (e.g. suspicious) file is opened and executed. Files are executed in a controlled (virtual) environment and a detailed report is generated for raised operating system events. This is unable to give a conclusion that a file is malicious or not, but the report which it generates can be used for generating a set of features for automatic classification manual analysis or manual analysis.

4.2. Wepawet

A tool which is specifically planned to detect PDF files is called Wepawet[12,15,16] .It is a an online malware detection system which searches for malicious PDF files and URL. It extends CWSandbox approach by adding a features extraction and classification system. It is a tool based on machine learning which concentrates on JavaScript attacks: it extracts, classifies Javascript code, deobfuscates within PDF files. It tool analyses specific commands which are associated to malicious files, also the order in which those commands are executed. It employs a Bayesian classifier, which is advantageous for the purposes of the analysis.

4.3. Nozzle

It is a tool which is developed to detect another serious attack called Heap Spraying attacks [10]. Its basic purpose is not to detect malicious PDF files, but it can be a very helpful resource, because many PDF files implement HS attacks.

4.4. MDScan

It is a most recent and advanced tool which was purposely designed to identify malicious Javascript code within a PDF file [13]. It implements a hybrid approach: in order to retrieve Javascript code it scans the PDF document i.e. it finds the objects related to Javascript routines i.e. static part, and after that the searched code is executed the code using a Javascript interpreter i.e.dynamic part. There is difference between Wepawet and MDScan in classifying the malware files. MDScan examines the part of the memory in which the Javascript routines are written, and heuristics are adopted to conclude whether or not the code is malicious.

4.5. PJScan

This tool is developed by Laskov and Srndi´c, and it extracts the features which are helpful for the classification from the Javascript code embedded in the PDF file, using a static N-gram analysis. After extracting features it uses a one-class SVM to classify the files. It only examines files that haveembedded Javascript code [14, 20].

4.6. PDF Scrutinizer

PDF Scrutinizer is a PDF analyzer in order to classify PDF documents by using static and dynamic detecting. PDF Scrutinizer focuses on JavaScript-based attacks, and also suitable to the non-JavaScript-based documents. It not only displays the resulting classification, but also furnishes further information on the reasons of the classification.

The main function of PDF Scrutinizer is dividing PDF documents into three parts, parsing, and extraction of actions and execution of actions.

1. **Parsing:** In this part loading and parsing the documents performed. Objects of PDF are analyzed and stored in the PDFbox to pursue steps and access them. Here parsing in PDF Scruitnizer try to extract PDF objects at all cost which is not be limited to the PDF specification
2. **Extraction of actions:** Like common PDF reader PDF Scrutinizer looks for the JavaScript action. If the /OpenAction command is used, then the document catalog the dictionary, store a reference to the /Name array and scanned for include JavaScript action. When all the action has been collected, then save it for later analysis and processed.
3. **Execution of actions:** The code extracted in above step is executed in a modified JavaScript engine, where the parts of the acrobat for JavaScript API are emulated. Because in this way they are able to return the correct values, this is the malicious functionality. Execution the code insures the action is good or bad. Then classifications malicious, suspicious and benign documents [9].

4.7. PDF Tools

It is a toolkit which helps in our understanding of the PDF analysis process. This toolkit consists of pdfid.py and pdf-parser.py. Pdfid.py is used for quickly scanning PDF for malicious objects and pdf-parser.py is used for examine their contents.

4.8. PDF Stream Dumper

PDF stream Dumper is robust Windows program that merges a number of PDF analysis tools under a unified GUI. With this Dumper Programs it becomes possible to explore PDF contents, decode object contents, deobfuscate JavaScript, examine shellcode, etc.

4.9. Jsunpack-n

It’s a command-line tool that analyzes malicious websites by emulating browser. It supports numerous other features, but one special tool includes the pdf.py script for extracting JavaScript embedded in PDF files.

4.10. Peepdf

Another interactive command-line tool called Peepdf which allows users to explore and analyze contents of PDF files. This is used for analyzing object contents, examining the file's structure as well as decoding embedded JavaScript and shellcode. Peepdf is a Python tool to explore PDF files in order to find out if the file can be harmful or not. The aim of this tool is to provide all the necessary components that a security researcher could need in a PDF analysis without using 3 or 4 tools to make all the tasks. With peepdf it is possible to see all the objects in the document showing the suspicious elements, it supports all the most commonly used filters and encodings; it can parse different versions of a file, object streams and encrypted files. With the installation of PyV8 and Pylibemu it provides Javascript and shell code analysis wrappers too. Apart of this it is able to create new PDF files and obfuscate existing ones. The main functionalities of peepdf are:

Analyze the different aspects like:

Decoding: hexadecimal, octal, name objects, references in objects and where an object is referenced, strings search (including streams), their physical structure (offsets) and logical tree structure, metadata, modifications between versions (changelog), compressed objects (object streams), analysis and modification of Javascript (PyV8):

Unescape, replace, join and shell-code analysis (Libemu python wrapper, pylibemu), variables (set command). It also Extract old versions of the document, objects, Javascript code, shell-codes (>, >>, $>, $>>) and checking hashes on VirusTotal**.**

4.11. Origami

It is a ruby framework for creating PDF files, parsing and analyzing. it carries a the pdfscan.rb script to scan the PDF for malicious objects in addition to providing programmers with the strength to automate PDF interactions, and for extracting Javascript embedded in a file extractjs.rb is used .

4.12. MalObjClass

Another Framework which builds an object Called JSON object which represents PDF files components. It permits programmers to examine, decode and easily parse PDF Objects .It also encompasses a feature which is used to scan file with VirusTotal.

5. Analysis and Discussion

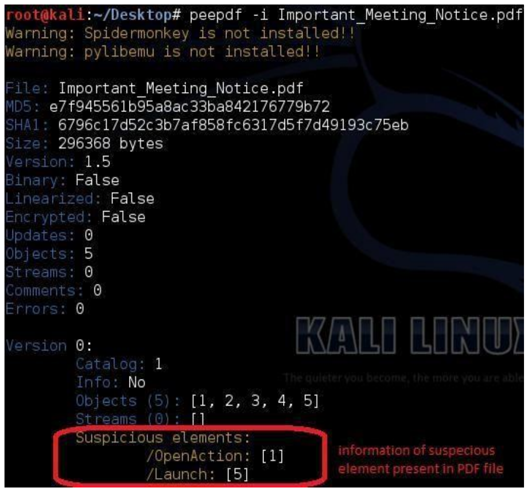
For experimentation on malicious PDF document, I first use Adobe PDF Escape Exe social engineering module in metasploit framework on linux operating system to create the suspicious PDF file and then execute the malicious Pdf attack**.** Almost every user has Adobe Acrobat application in their systems. When the victim accesses a particular suspicious PDF document that the attacker transmits to him using e-mail attachments or some other methods, then all of its credentials (victim credentials) are directed to the attackers system using reverse Tcp handler module that connects the victim`s system to the attackers system. After that, I applied forensic analysis tools i.e. peepdf and pdf-parser to analyze that malicious code in suspicious PDF document:

5.1. Peepdf:

Attackers continue to use malicious PDF files as part of targeted attacks and mass-scale client-side exploitation. Peepdf is an excellent addition to the PDF analysis toolkit for examining and decoding suspicious PDFs. The peepdf tool is written in Python. It especially operates in its interactive mode that leads to the tool's shell that allows you to navigate the PDF file's structure and explore its contents. For experiment with this, open peepdf tool.

Examining a PDF File for Suspicious Characteristics

You can scan the PDF file by using the "peepdf *filename.pdf*" command to obtain critical information regarding the suspicious PDF file.



```
root@kali:~/Desktop# peepdf -i Important_Meeting_Notice.pdf
Warning: Spidermonkey is not installed!!
Warning: pylibemu is not installed!!

File: Important_Meeting_Notice.pdf
MD5: e7f945561b95a8ac33ba842176779b72
SHA1: 6796c17d52c3b7af858fc6317d5f7d49193c75eb
Size: 296368 bytes
Version: 1.5
Binary: False
Linearized: False
Encrypted: False
Updates: 0
Objects: 5
Streams: 0
Comments: 0
Errors: 0

Version 0:
        Catalog: 1
        Info: No
        Objects (5): [1, 2, 3, 4, 5]
        Streams (0): []
        Suspicious elements:
                /OpenAction: [1]
                /Launch: [5]
```

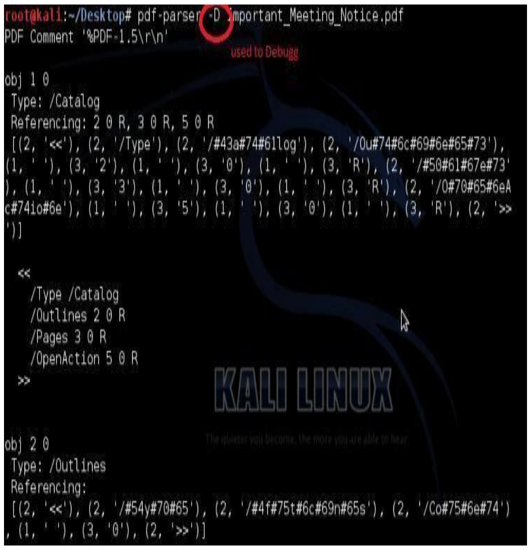
information of suspecious element present in PDF file
KALI LINUX
The quieter you become, the more you are able

Figure 2. Analyze malicious PDF file with Peepdf

The above figure 2, describes the information about the malicious PDF file i.e. the malicious elements and the actions that are perform by the suspicious PDF file, hashing function that is used, the objects present in that particular file and any kind of error that are present in the suspicious PDF file. The (– I) mode present in the command describes the interactive mode.

Pdf- Parser

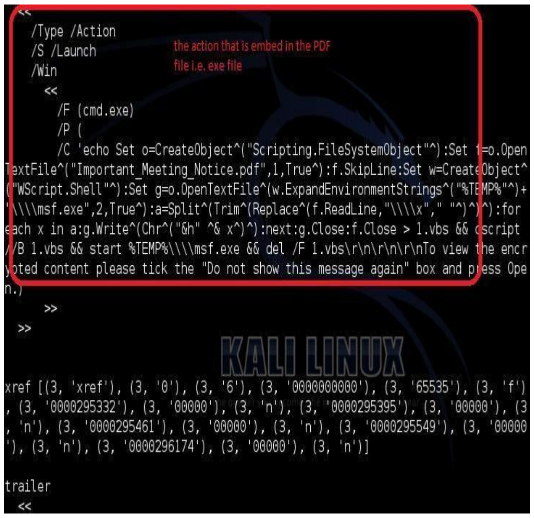
This tool will parse a PDF document to identify the fundamental elements used in the analyzed file.



```
root@kali:~/Desktop# pdf-parser -D Important_Meeting_Notice.pdf
PDF Comment '%PDF-1.5\r\n'

obj 1 0
 Type: /Catalog
 Referencing: 2 0 R, 3 0 R, 5 0 R
 [(2, '<<'), (2, '/Type'), (2, '/#43a#74#61log'), (2, '/Ou#74#6c#69#6e#65#73'),
(1, ' '), (3, '2'), (1, ' '), (3, '0'), (1, ' '), (3, 'R'), (2, '/#50#61#67e#73'
), (1, ' '), (3, '3'), (1, ' '), (3, '0'), (1, ' '), (3, 'R'), (2, '/O#70#65#6eA
c#74io#6e'), (1, ' '), (3, '5'), (1, ' '), (3, '0'), (1, ' '), (3, 'R'), (2, '>>
')]

  <<
    /Type /Catalog
    /Outlines 2 0 R
    /Pages 3 0 R
    /OpenAction 5 0 R
  >>


obj 2 0
 Type: /Outlines
 Referencing:
 [(2, '<<'), (2, '/#54y#70#65'), (2, '/#4f#75t#6c#69n#65s'), (2, '/Co#75#6e#74')
, (1, ' '), (3, '0'), (2, '>>')]
```

Figure 3. Debugg the malicious PDF file

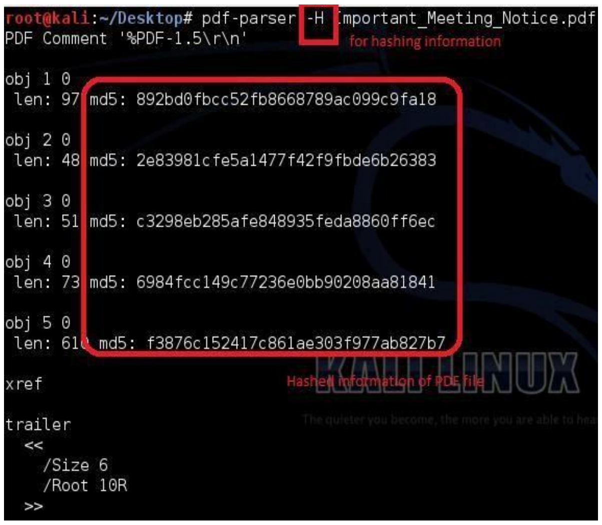
The above figure 3 shows the full description of malicious PDF file i.e. number of objects (obj), catalog, pages, open actions present in the suspicious pdf file. The (-D) mode present in the command debugs the malicious PDF file.



```
  <<
    /Type /Action
    /S /Launch
    /Win
      <<
        /F (cmd.exe)
        /P (
        /C 'echo Set o=CreateObject^("Scripting.FileSystemObject"^):Set f=o.Open
TextFile^("Important_Meeting_Notice.pdf",1,True^):f.SkipLine:Set w=CreateObject^
("WScript.Shell"^):Set g=o.OpenTextFile^(w.ExpandEnvironmentStrings^("%TEMP%"^)+
'\\\msf.exe",2,True^):a=Split^(Trim^(Replace^(f.ReadLine,"\\\\x"," "^)^)^):for
each x in a:g.Write^(Chr^("&h" ^& x^)^):next:g.Close:f.Close > 1.vbs && cscript
//B 1.vbs && start %TEMP%\\\msf.exe && del /F 1.vbs\r\n\r\n\r\nTo view the encr
ypted content please tick the "Do not show this message again" box and press Ope
n.)
      >>
  >>


xref [(3, 'xref'), (3, '0'), (3, '6'), (3, '0000000000'), (3, '65535'), (3, 'f')
, (3, '0000295332'), (3, '00000'), (3, 'n'), (3, '0000295395'), (3, '00000'), (3
, 'n'), (3, '0000295461'), (3, '00000'), (3, 'n'), (3, '0000295549'), (3, '00000
'), (3, 'n'), (3, '0000296174'), (3, '00000'), (3, 'n')]

trailer
  <<
```

Figure 4. Embedded code in malicious PDF

The above figure 4 describes the malicious embedded .exe in the PDF document. The action (cmd.exe) that is execute when the malicious PDF file is opened by the clients. The description of cmd.exe file is shown in fig. 4.



```
root@kali:~/Desktop# pdf-parser -H Important_Meeting_Notice.pdf
PDF Comment '%PDF-1.5\r\n'

obj 1 0
 len: 97 md5: 892bd0fbcc52fb8668789ac099c9fa18

obj 2 0
 len: 48 md5: 2e83981cfe5a1477f42f9fbde6b26383

obj 3 0
 len: 51 md5: c3298eb285afe848935feda8860ff6ec

obj 4 0
 len: 73 md5: 6984fcc149c77236e0bb90208aa81841

obj 5 0
 len: 610 md5: f3876c152417c861ae303f977ab827b7

xref

trailer
  <<
    /Size 6
    /Root 10R
  >>
```

for hashing information
Hashed information of PDF file
KALI LINUX
The quieter you become, the more you are able to hear

Figure 5. Hash information of PDF file

The above figure 5 shows the hashing information about the malicious PDF file. The (-H) mode present in the command fetches the hash information of the particular file. There are other modes also that are used for other purposes.

6. Conclusion

At present, the malicious PDF files are serious threat that requires effective and robust detection strategies. The JavaScript is basically used to create the malicious scripts as attackers have deep knowledge about the JavaScript language. The effective tools that analyze malicious PDF files are quite modest and affected by the application of simple obfuscation techniques. These tools give information about the malicious contents that are present in the suspicious PDF files. In this paper, I implied peepdf and pdf-parser tools that effectively analyze the malicious PDF document i.e. the malicious action and their description and also the objects that are present in that particular malicious document.

References

[1] Azuan Ahmad, Bharanidharan Shanmugam, Norbik Bashah Idris, Ganthan Nayarana Samy, and Sameer Hasan AlBakri, “Forensic Analysis Tool for Malicious Pdf Files And Shellcode Analysis”, International Conference on Emerging Trends in Engineering and Technology, Dec. 7-8, 2013, Patong Beach, Phuket (Thailand).

[2] Stevens D., “Malicious PDF documents explained. Security & Privacy”, IEEE, 2011. 9(1): p. 80-82.

[3] Ahmad Bazzi1 and Yoshikuni Onozato, “Automatic Detection of Malicious PDF Files Using Dynamic Analysis”, http://www.jsst.jp/e/JSST2013/extended_abstract/pdf/Paper%2056.pdf.

[4] C. Smutz and A. Stavrou, “Malicious pdf detection using metadata and structural features”, In Proceedings of the 28th Annual Computer Security Applications Conference, ACSAC ’12, 2012.

[5] Tim Bienz, Richard Cohn, and James R.meehan. “Portable Document Format Reference Manual” Adobe Systems Incorporated. November 12, 1996.

[6] Pavel Laskov and Nedim Šrndic. “Static detection of malicious JavaScript-bearing PDF documents”. ACSAC’ 11 Proceedings of the 27th Annual Computer Security Applications Conference.

[7] Internet Security Threat Reports. 2011 Trends. Symantec, April 2012.

[8] Davide Maiorca, Igino Corona, Giorgio Giacinto. “Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious PDF files detection”, Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, Pages 119-130.

[9] Dai Haobing, “Malicious PDF Document Analysis”, Faculty of Engineering and Sustainable Development University of Gävle S-801 76 Gävle, Sweden.

[10] Ratanaworabhan, P., Livshits, B. and Zorn, B.: NOZZLE: A Defense Against Heapspraying Code Injection Attacks. In: SSYM 2009 Proceedings of the 18th Conference on USENIX Security Symposium (2009)

[11] Willems, C., Holz, T. and Freiling, F., “Toward Automated Dynamic Malware Analysis Using CWSandbox”, Journal IEEE Security and Privacy Archive 5(2) (2007)

[12] Cova, M., Kruegel, C. and Vigna, G., “Detection and Analysis of Drive-by-Downloads Attacks and Malicious Javascript Code”, In: Proceedings of International World Wide Web Conference, (2010)

[13] Tzermias, Z., Sykiotakis, G., Polychronakis, M. and Markatos, E.P.: Combining Static and Dynamic Analysis for the Detection of Malicious Documents. In: EUROSEC 2011 Proceedings of the Fourth European Workshop on System Security (2011)

[14] Laskov, P. and ˇSrndi´c, N.: Static Detection of Malicious JavaScript-Bearing PDF Documents. In: Annual Computer Security Applications Conference (2011).

[15] Wepawet, http://wepawet.iseclab.org/.

[16] Davide Maiorca, Giorgio Giacinto, and Igino Corona, “A Pattern Recognition System for Malicious PDF Files Detection”, pp. 510–524, 2012, Springer-Verlag Berlin Heidelberg 2012.

[17] http://labs.appligent.com/pdfblog/pdf-object-types/

[18] http://resources.infosecinstitute.com/pdf-file-format-basic-structure/.