International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 08 | Aug-2016 www.irjet.net p-ISSN: 2395-0072

Malicious PDF – A Review

Gurjeet Singh[1]

[1]*Student, Dept of CSE, Chandigarh University, Punjab, India*

Abstract - *Nowadays one of the serious threats to system security is malicious PDF files. Attacks via malicious PDF files usually occur through email communications. Various social engineering techniques are being used by the attackers to make users open malicious files. In this paper overview of PDF file structure is provided and basic attacks that occur via PDF files are discussed.*

***Key Words*: PDF structure, Malicious PDF, Java script attack, Code Obfuscation.**

1. INTRODUCTION

Information security is often divided into the field of computer security and network security. By computer security, we mean measures taken to safeguard the individual machine from attacks. When we discuss network security, we mean measures taken to safeguard the complete network, as well as connected systems and devices from attacks. In a similar manner, computer systems attacks are also conducted internally by the host, or remotely over the network. We frequently talk over with the term latter as network attacks. Usually, network attacks target at the services running on the computers and servers and therefore referred to as server-side attacks. These attacks deem the target running services on the open port and exploit vulnerabilities in these services. Server-side attacks are generally conducted in five phases, described by Skoudis et al. [4].

- Reconnaissance
- Scanning
- Exploitation
- Keeping Access
- Removing tracks

Even though we safeguard machine by default firewall set up to block the new incoming traffic, attackers still have their alternative tactics to evade the security. The firewall generally blocks new in-bound connection attempt but permit users behind the firewall to out-bound the connection. Thereby allowing both parties with established connections to communicate over the channel in both directions [5]. This reality is exploited by attackers which we call client-side attacks. Client-side attacks exploit susceptibilities in user software, like e-mail applications, web browsers, runtime environments, media players and last however not least document viewers. The exploitation of PDF document viewers has been vital for the last number of years and appears to still be on a rise. PDF document viewers are widely targeted for several reasons. First of all, we have all got one, and PDF is the common standard for the document exchange. Hence, we are all ready to open a PDF document and expect to receive PDF documents from every kind of sources. Secondly, PDF is an old format and at the same time extremely versatile. This allows the attackers to use the versatility to exploit a piece of code in a way that could never be imagined. Further sections provide a deeper look at the PDF format and how it can be exploited.

1.1 Portable Document Format (PDF)

The Portable Document Format (PDF) [6] is basically a file format produced by Adobe Systems in 1993 and used to exchange and present documents reliably, independent of hardware, software, or operating system. In 2008, the PDF format was officially declared as an open standard by ISO and since the release, PDF is considered as the industry standard for the file exchange. PDF files, instead of containing text and images, the format also offers the embedding of JavaScript and Flash, and has the flexibility to open external resources from the native machine or the internet [7]. These features are there how utilized by attackers to exploit vulnerabilities in the PDF document viewers.

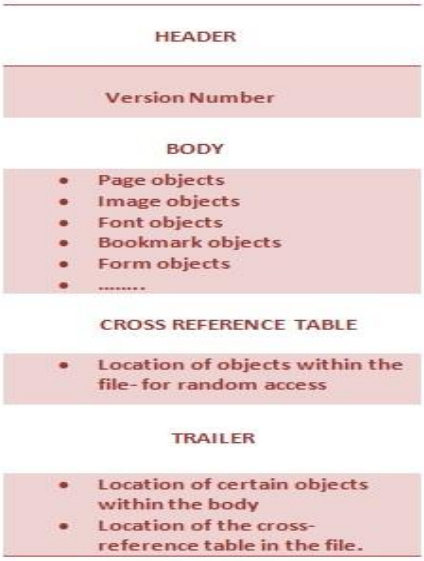

Fig -1: PDF Structure

1.2 Malicious Use of PDF Documents

As discussed earlier, PDF is the popular standard for document exchange. Nearly everyone has a PDF reader installed, or the owner can merely have a tough time to

participate in society. Covered with the actual fact that PDF may be an extremely powerful and flexible format, it's virtually a dream come true for an assaulter. In the following, the foremost common strategies of malicious PDF distribution are going to be mentioned. Following this can be a superficial inspects some ways in which the exploits are literally implemented.

1. Distribution of Malicious PDF

According to [9], there are main three channels for distributing malicious PDF documents. These channels are mass mailing, targeted attacks, and drive-by downloads. These are the client-side attack methods. Mass mailing is well known for malicious PDF distribution since the general public is accustomed to receiving PDF attachments in e-mails. In a mass mailing theme, massive spam campaigns are set up to deliver e-mails containing malicious PDFs to a large range of users. Social engineering tricks are used to encourage the receiver to open up the attached document. Typically, the content of these e-mail shows an ingress style paragraph to a recent news event, with a promise of the complete story and exciting details within the hooked up PDF document. The popular subject includes:

- E-mail from the organization or a government department.
- Politics.
- A recent incident (Accidents, disasters, war).
- Controversial/sexual subjects.

In mass mailing campaigns malicious PDFs are sent and these can often contain embedded executable pay-loads, which is extracted and also executed when the PDF is opened in a susceptible environment. Due to the entrenched feature of routinely opening PDF documents in most browsers, drive- by downloads is additionally a well-liked channel for malicious PDF distribution. A user might not even notice that a PDF has been opened on their machine once falling victim to a drive-by download. As hostile PDFs sent through mass mailing, a web-hosted PDF can typically be tiny and not contain any embedded executables. Instead, they contain a tiny piece of code that, when successful exploitation, can transfer and execute malicious executables from the internet. Such a theme provides the assaulter good flexibility, as they are going to be able to update the malicious code that's downloaded at any time. A targeted malicious PDF is targeting a personal or a corporation and is specially crafted to achieve success against this target. The possibility of success is boosted by fastidiously researching the target and designing the attack. By gathering data on the target the social engineering content of the attack may be created in such how that the target goes to own high trust within the received PDF document. Also, the exploit may be chosen in such a way that it has a high probability of being successful on the target system. With targeted attacks, there's typically an extreme motivational and capable organization responsible. Such threats might have access to zero-day exploits which can greatly increase the likelihood of their success. The value of targeted attacks is comparatively low and attributable to its sophistication and concealment several are most likely never reported because the victim is unaware of the compromise.

2. Exploit Implementation

New vulnerabilities in PDF readers and associated plug-ins and libraries emerge all the time, and with the newest vulnerabilities follow an exploit. Totally different sorts of exploits may be utilized in a malicious PDF, and one single PDF might contain many exploits classified along. In [9] PDF exploits are classified into two distinct classes; JavaScript and non-JavaScript based exploits. JavaScript-based exploits are created through the JavaScript support within the PDF description. Attackers know the ability a scripting language like JavaScript brings to the format. JavaScript is to exploit vulnerabilities within the PDF JavaScript API and to fill the PDF reader's memory with malicious code, employing a technique referred to as heap spray. A whole exploit typically consists of code that initial heap sprays the reader's memory with shell code, and then perform a vulnerable function. This might lead to the shell code being executed.
According to [9] malicious PDFs now a day use JavaScript in one way or another. Non-JavaScript based exploits are much rarer than the JavaScript-based exploits. An alternative to JavaScript is to use PDF's ability to enter Flash content. Such content is also accustomed exploit vulnerabilities within the Flash engine, or to place shell code within the heap of the reader. There also exists a vulnerability within the approach TIFF-images can handle, which may result in code execution even while not a heap spray. In addition to such specific vulnerabilities, the Portable Document Format has several nice options which may facilitate the assaulter in building a malicious PDF while not the utilization of any vulnerability intrinsically. At Black Hat Europe 2008 Eric Filiol et al. given a paper [10] on this subject. The options include practicality to open different documents, open hyperlinks, amendment document hierarchy placement, access resources outside the active document, execute applications, open files, print documents, access remote resources, submit resources to remote and import information from the user. A deeper verify a number of these options and their attainable misuse follow:

- *OpenAction* - The OpenAction function let the creator of a PDF document outline actions to trigger once the document is opened. The perform doesn't do a lot itself, however, once functions like Launch or ActionClass space given as input, things will get ugly. The function is maliciously accustomed run exploits as shortly because the PDF document is opened, giving the victim no likelihood to prevent it.
- *AA* - The Additional Action perform works in a similar approach as OpenAction task. However, rather than triggering on the document being opened, it triggers on specific actions set by the PDF creator. Such actions

enclosed triggering once a precise page is opened or closed, once clicking certain area, when the mouse is over a specific area, when printing and so on.

- *Action category* - The Action group comprises numerous functions that may be placed in an OpenAction or AddidtionalAction function. Action category functions are used for executing files, activating hyperlinks, sending kind information and much more. These all functions might aide the assaulter in making an operating malicious PDF. These days' threats are alleviated by most PDF readers by showing the victim a confirmation box whenever an action is triggered. However, through social engineering and also the restricted awareness of the general public, this threat continues to be one to reckon with.
- *The Launch* is from the Action category function. It permits execution of any file on the target system, with no obligatory arguments. Before confirmation boxes were enforced into PDF readers this was the only most crucial vulnerability in PDF.
- *SubmitForm* is generally used for electronic forms. It permits the creator of the PDF to send information from the form element to a specified address. This provides a wonderful information channel for an assaulter to retrieve information from the target host.

1.3 Obfuscation of PDF Documents

Attackers use many ways to cover, or alter, the malicious contents of a PDF document. Such ways are possible because of the powerful and versatile nature of the PDF format. Especially the JavaScript typically found in an exceedingly malicious PDF document has large amounts of obfuscation ways. Obfuscation is supposed to throw off the analyst attempting to research the malicious content, and additionally to evade detection by signature anti-virus solutions or IDS. The techniques may fit on their own, however typically a combination is employed to produce an effective step against each detection mechanisms and human analysts. In [11], Leif Arne Sand presents some common techniques which are briefly mentioned below.

1. Separating Malicious Code over Multiple Objects

As in a coding language and in a malicious code of a PDF document can be spread among many objects. The code is then created in such a way that during execution congregates itself into an entire code performing the required actions [9]. By taking advantage of the flexibility to seek advice from indirect objects spreading ways in an exceedingly JavaScript embedded within the document. Objects can build the work for analyst a lot of more durable and throw of signature based IDS and antivirus.

2. Applying Filters

By applying filters the author is in a position to encode and compress the streams of PDF document. Attackers will use this feature so as to evade detection by security code. If the code does not support the filters used it may never even, see the malicious code. Applying filters would not evade a knowledgeable human analyst, however, it will most definitely build his job longer overwhelming.

3. White area randomisation

Since JavaScript ignores whitespace at run-time, it attainable to insert as arbitrary amounts of whitespace characters within the code [12]. Whereas this cannot fool the human analyst, the signature primarily based detection mechanism could simply be thrown off. Any detection mechanism counting on the hash sum of the JavaScript also will be fooled, as this will modify once whitespace characters are inserted.

4. Block randomization

Block randomization involves ever-changing the structure of the JavaScript in such a manner that it functions in the same way, however, includes a different syntax. The instance below shows three other ways of writing a loop that performs precisely the same function [12].

5. Comment randomization

Comments also are unheeded by the JavaScript parser at run-time. This implies that attacker could insert or edit comments within the source code to alter its hash sum. This can however only effect detection mechanisms counting on the hash sum. The strategy has no impact on the human analyst.

6. Variable Name Randomization

Since one will provide variables virtually any name one would need, it's attainable for the wrongdoer to alter variable names. This could fool signature primarily based detection mechanisms searching for specific variable names, however, will have very little impact on the human analyst.

7. String Obfuscation

The goal of string obfuscation is to alter strings in order to appear insignificant and unreadable to the human analyst. This may be achieved in many ways. The wrongdoer could split the string into many substrings that are concatenated at run-time. Additionally, the strings could also be encoded using schemes like Unicode, hexadecimal, base64, and so on. Finally, the attacker could alter the string using arbitrary function over it, like XOR or substitutions. A de-obfuscation performs would then be executed at run-time, revealing actuality string simply before it is used. This technique includes a large impact on the human analyst, which can have to be compelled to pay plenty of their time revealing actuality content of the strings. The strategy is additionally effective for concealing, for instance, shell code from signature based IDS.

8. Function Name Obfuscation

This technique is applied to cover the use of standard functions, like the usually used unescape() and eval().

Creating pointers for such functions using arbitrary names, can build a human analyst job more durable and can bypass signature-based detection mechanisms searching for specific functions.

9. Whole number Obfuscation

Integer obfuscation aims at representing numbers in an exceedingly set of various ways. For example, if the malicious code uses a suspicious memory address e.g. 0x08000000, detection mechanisms could check for this address within the code. Using the integer obfuscation, the wrongdoer could instead represent 0x08000000 as 16777216*8.

2. RELATED WORK

Tzermias et al. [15] introduced an approach called MDScan which was both static as well as dynamic file scanner used to identify the malicious PDF files. MDScan worked by extracting all objects and embedded java scripts in objects from the PDF file body. Then embedded JavaScript code was examined by SpiderMonkey which is JavaScript engine. During execution period the string variables are analyzed and if address space of interpreter is found with a shellcode, the file is classified as malicious.

Vatamanu et al. [16] introduced two clustering methods of PDF files namely hash table and hierarchal bottom up clustering on the basis of embedded JavaScript tokenization. The basic approach was to detect obfuscated java scripts using clustering methods and fingerprints are created for every pdf file which is inspected. The dataset comprised 1333420 benign files and 997615 malicious files. According to their results. The empirical results showed that java scripts were found in 90 % of malicious files and about 5 % in benign files also hash table clustering proved better than bottom up clustering.

Lu et al. [22] presented a tool MPScan that integrated dynamic java script obfuscation and static malicious PDF detection. It was capable of dealing with both attacks- java script based and non java script based. This tool is composed of two modules namely multi level malware detection unit and embedded code extraction unit. The proposed tool is very effective against unknown obfuscation methods, heap spraying and detection of shell codes.

Schmit et al. [18] presented tool, PDF scrutinizer which detects malicious PDFs by using dynamic and static methods of analyzing. The main focus of this tool is attacks based on java scripts. The tool consisted of three modules namely parser, action executor and action executor where parser imitates the adobe readers' way of parsing a document, action executor Extracts JavaScript activities statistically and action executor performs execution in JSengine on extracted JavaScript code. The evaluation dataset consists 11,278 malicious and 6054 benign PDF files which were gathered from honeypots, emails and websites. Empirical results illustrate a detection rate of 90%. The authors also contrasted their proposed tool with former existing PDF analysis tools such as MDScan, Wepawet and PJScan.

3. CONCLUSIONS

The paper presents an overview of malicious PDF files, its structure and common attacks methods. In order to detect attacks via pdf files it is required to study PDF structure in depth. Mostly javascripts and obfuscation techniques are used as attack vectors in PDF files. Attackers either perform mass mailing or target unique users to make them open malicious PDF file via social engineering techniques. Attachments as PDFs or word document are rarely filtered at email gateway causing huge risks to systems. Thus it is important to detect malicious PDFs and prevent them from causing any harm to user systems.

REFERENCES

[1]http://netsecurity.about.com/cs/hackertools/a/aa030504.htm

[2]https://www.paloaltonetworks.com/documentation/glossary/what-is-an-intrusion-prevention-system-ips

[3]http://www.symantec.com/connect/articles/social-engineering-fundamentals-part-i-hacker-tactics

[4] Skoudis, Ed, and Tom Liston. Counter hack reloaded: a step-by-step guide to computer attacks and effective defenses. Prentice Hall Press, 2005.

[5] Harper, Allen, Shon Harris, Jonathan Ness, Chris Eagle, Gideon Lenkey, and Terron Williams. Gray Hat Hacking The Ethical Hackers Handbook. McGraw-Hill Osborne Media, 2011.

[6]http://www.adobe.com/devnet/pdf/pdf_reference.html /

[7] Tzermias, Zacharias, Giorgos Sykiotakis, Michalis Polychronakis, and Evangelos P. Markatos. "Combining static and dynamic analysis for the detection of malicious documents." In Proceedings of the Fourth European Workshop on System Security, p. 4. ACM, 2011.

[8]http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/pdf_reference_1-7.pdf

[9]https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the_rise_of_pdf_malware.pdf

[10] Blonce, Alexandre, Eric Filiol, and Laurent Frayssignes. "Portable document format (pdf) security analysis and malware threats." In Presentations of Europe BlackHat 2008 Conference, Amsterdam. 2008.

[11] Sand, L. A. "The portable document format-a study of common distribution, exploit, evasion, mitigation and detection techniques." Student paper in Information Security at Gjøvik University College (2011).

[12] Feinstein, Ben, Daniel Peck, and I. SecureWorks. "Caffeine monkey: Automated collection, detection and analysis of malicious javascript." Black Hat USA 2007 (2007).

[13] Kononenko, Igor, and Matjaž Kukar. Machine learning and data mining: introduction to principles and algorithms. Horwood Publishing, 2007.

[14] Laskov, Pavel, and Nedim Šrndić. "Static detection of malicious JavaScript-bearing PDF documents." In Proceedings of the 27th Annual Computer Security Applications Conference, pp. 373-382. ACM, 2011.

[15] Tzermias, Zacharias, Giorgos Sykiotakis, Michalis Polychronakis, and Evangelos P. Markatos. "Combining static and dynamic analysis for the detection of malicious documents." In Proceedings of the Fourth European Workshop on System Security, p. 4. ACM, 2011.

[16] Vatamanu, Cristina, Dragoş Gavriluţ, and Răzvan Benchea. "A practical approach on clustering malicious PDF documents." Journal in Computer Virology 8, no. 4 (2012): 151-163.

[17] Maiorca, Davide, Giorgio Giacinto, and Igino Corona. "A pattern recognition system for malicious pdf files detection." In Machine Learning and Data Mining in Pattern Recognition, pp. 510-524. Springer Berlin Heidelberg, 2012.

[18] Schmitt, Florian, Jan Gassen, and Elmar Gerhards-Padilla. "PDF SCRUTINIZER: Detecting JavaScript-based attacks in PDF documents." In Privacy, Security and Trust (PST), 2012 Tenth Annual International Conference on, pp. 104-111. IEEE, 2012.

[19] Smutz, Charles, and Angelos Stavrou. "Malicious PDF detection using metadata and structural features." In Proceedings of the 28th Annual Computer Security Applications Conference, pp. 239-248. ACM, 2012.

[20] Šrndic, Ned, and Pavel Laskov. "Detection of malicious pdf files based on hierarchical document structure." In Proceedings of the 20th Annual Network & Distributed System Security Symposium. 2013.

[21] Pareek, Himanshu, P. Eswari, N. Sarat Chandra Babu, and C. Bangalore. "Entropy and n-gram Analysis of Malicious PDF Documents." In International Journal of Engineering Research and Technology, vol. 2, no. 2 (February-2013). ESRSA Publications, 2013.

[22] Lu, Xun, Jianwei Zhuge, Ruoyu Wang, Yinzhi Cao, and Yan Chen. "De-obfuscation and detection of malicious pdf files with high accuracy." In System Sciences (HICSS), 2013 46th Hawaii International Conference on, pp. 4890-4899. IEEE, 2013.

[23] Maiorca, Davide, Igino Corona, and Giorgio Giacinto. "Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious pdf files detection." In Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, pp. 119-130. ACM, 2013.

[24] Pareek, Himanshu, P. R. L. Eswari, and N. Sarat Chandra Babu. "Malicious PDF document detection based on feature extraction and entropy." In International Journal Journal of Security, Privacy and Trust Management, Volume 2, No 5, 2013.

[25] Maiorca, Davide, Davide Ariu, Igino Corona, and Giorgio Giacinto. "An Evasion Resilient Approach to the Detection of Malicious PDF Files." In Information Systems Security and Privacy, pp. 68-85. Springer International Publishing, 2015.