BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION COMPUTER SCIENCE IN GERMAN

Cloud based malicious PDF Detection using Machine Learning

– Diploma thesis –

Author
Viorel GURDIS

2020

Abstract

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Context

The informational technology progress, that is in continous growth, brings a lot of benefits along with new responsabilities. There are plenty of applications that we use everyday across the entire World Wide Web. We don't even realize how much of our personal information is transfered to the virtual environment. That being said, it's important to be prepared for cybersecurity threats that can misuse our sensitive data. The problem is that the cyber attackers develop a lot of *hacking*[1] techniques, which are becoming increasingly difficult to detect. The popular software applications are the best target to inject malicious behavior. This happened with the Adobe PDF format. PDF documents are well known, trustworthy files and they are a global solution for sharing information. There are a lot of PDF readers and even browsers and email applications have support to open these files for viewing. It became so convenient to work with this format, that users interact with PDF documents without noticing any possible danger. However PDF is a often used attack vector. The large number of discovered PDF vulnerabilities and also the support of embedding Javascript code into documents are just some of the most exploited methods.

1.2 Motivation

Under the guise of seeming harmless, PDF documents are used on a daily basis across numerous public institutions, private companies and for personal purposes. Most of the people don't consider that these files could be dangerous and just copy the documents to their computers and access them. It is enough for one PDF to be malicious and the entire network affiliated to an institution becomes compromised. The cyber attackers succeed in achieving their goals, but

[1] attempt to gain unauthorized access to data in a system

the victim institution requires huge resources, both financially and time wise, to restore the integrity and security of their infrastructure. A measure to combat the described situation is having a real time protection installed on the computer. The most common solution, antivirus applications, work by signature matching, which is effective for detecting previously identified malware[2]. This means that all the antivirus applications require permanent updates in order to keep their malware databases up-to-date. Many users opt out of security solutions because of the multitude of hardware requirments they need. In this thesis we will go through a new approach that implies transfering complex detection algorithms from user's computers to a remote service, whose only task is analyzing suspicios uploaded PDF documents. Consequently the computers that will use this solution get rid of additional workload while still maintaining the system secure.

1.3 Paper structure and original contributions

The main contribution of this thesis is the research, design and implementation of alternative security solution oriented on multiplatform use and performance efficiency. For achieving this, the work was splitted into three important parts:

1. **Real time monitoring system on Windows**, whose goal is to notify the user when a specific file type, in our case PDF, is downloaded. The main focus when designing this software was to keep its functionality basic, so that it would require minimal computer's resources. As soon as it detects a new such file, it submits it to the Cloud for further analysis.

2. **Development of intelligent algorithm for PDF classification** implies studying the PDF format skeleton, researching popular attack vectors using PDFs, extracting the most relevant features from a document and feeding them to the most fit Machine Learning classification algorithm. A part of this effort was also spent for searching the dataset, based on which the classification model was trained to correctly detect malicious PDF files.

3. **Cloud based application for remote scanning**, which is a generic application that can integrate models such as the developed one for PDF classification and can use them for analyzing the uploaded files. It offers an user friendly dashboard[3] for visualizing the scanned files and also provides the possibility of submitting an URL containing a PDF file. The application will care about downloading and analyzing the file, showing the user

[2]any software intentionally designed to cause damage to a computer, server, client, or computer network
[3]type of graphical user interface which provides relevant informations

the scanning results. This is a perfect option for users that requires the Clouds features from a smartphone.

The remaining parts of this thesis are structured in the following manner: Chapter 2 contains the research made in the past years on malicious PDFs and Cloud Computing as an antimalware solution. Chapter 3 introduces the reader into fundamental mechanisms used for implementing the final application, as well as a brief history of malware in PDF. In Chapter 4 we present a Proof of Concept for developed classification model based on recreating a pseudo attack using malicious PDF. The 5th Chapter is putting each component together and presents the overall design of the application. Chapter 6 is meant for our final conclusions and we also show some of the future ideas for improving the application.

Chapter 2

Related work

Beginning with the first reported occurences of malware, the security researchers have made a huge effort to prevent the harmful behavior. Over the years malware has evolved in different forms and of course the antivirus industry has developed more complex detection solutions. Since PDF documents became a good target for cybercriminals, more and more specialists pay attention to the analysis of dangerous PDFs. Integration of so many analysis tools in a single antivirus software has a negative impact on computers performance. For this reason, the cybersecurity field attaches importance to Cloud Computing. In the following sections we will analyze other academic and industrial approaches for transfering antimalware engines to the cloud and some documented detection techniques of malicious PDFs.

2.1 Cloud based malware detection

In the field of Cloud solutions for malware detection, there is a sparse amount of shared academic works. As compensation, in the antivirus industry there is a fast growing interest for Cloud Computing, which has higher computational power and could therefore run more advanced and complex detection algorithms.

An important example is **VirusTotal** [11], a popular Cloud application developed by Hispasec Sistemas, that aggregates more than 50 antivirus engines and makes them publicly available for scanning uploaded files. From the user perspective, this is an excellent application, that can extract metadata of the submitted file and can identify any dubious signal. The provided result represents a comparation between analysis verdicts of cybersecurity market leaders. Of course this is an advantage in terms of the scanning result precision and respectively a gain regarding spent computational time. On average it takes aprox. 55 seconds to upload and analyze a 400KB file. VirusTotal is also helping to maintain the global cybersecurity at a high level, by sharing all the submitted suspicious files with the security researchers. Thereby the antivirus engines will be permanently improved.

Sandbox Analyzer is another antimalware cloud solution developed by Bitdefender [2]. It's approach is a bit different, because it ensures the security on a private network, where the Bitdefender product is installed, thus not being available for public access. It's operating principle is also specific, by preventing the execution of threats on an endpoint and automatically sending of harmful files to the Cloud. After extra analysis in the Cloud, the Sandbox Analyzer can take remediation action based on the verdict. In that way, malicious files get disinfected, quarantined or deleted. One of the benefits for this solution is in-depth analysis of malicious files in an isolated environment, rather than on user's machine. Thereby the risk for performance implication, as well as the risk of accidental run of malware on an endpoint machine are eliminated. At the core of the Sandbox Analyzer there are Machine Learning algorithms and dynamic behavior analysis techniques that are constantly improved to detect fresh threats.

2.2 Detection of malicious PDF

PJScan, presented in the paper of Laskov and Srndic [6], is a Machine Learning approach that trains One-Class Support Vector Machine (*OCSVM*) to classify PDF files. This approach is focused on static analysis of embedded Javascript code, as it is known for the ability of integrating malicious behavior in PDF documents. The authors used *n-gram* analysis to extract lexical features, such as Javascript operators and other tokens. The obtained sequence of features served later as input for the machine learning algorithm. The trained model can correctly classify malicious samples containing Javascript code. However PJScan has a lower accuracy, because it is not able to detect obfuscated parts and there are also some samples containing other types of malicious payload, such as SWF[1].

Another example of static analysis of PDF Structure is **PDF Tools** by Didier Stevens [12]. It represents a suite of tools for scanning, parsing and dumping PDF files. This approach focuses on identifying the fundamental elements of the format, such as Streams, Cross Reference Tables etc. The advantage of PDF Tools is their ability of name obfuscation handling and their simplicity. Because of their high speed performance, PDF Tools are largely used in cybersecurity research.

A completely new approach mentioned in the research paper of Fettaya et al. [4] describes an algorithm that uses a Convolutional Neural Network (*CNN*) to detect malicious PDF files. The trained model, based on a single convolutional layer with a global max pool and a linear layer doesn't require any data preoprocessing. Instead of this, the model is trained using as input the binary representations of the files. It is worth noting that the described algorithm could be efficiently used for distinguishing various families of malware. The rate of correct detections achieves 94%, the algorithm being also capable to classify aprox. 80% of the malware into

[1] Adobe Flash file format used for multimedia

different categories.

Chapter 3

Scientific Problem

3.1 Problem definition

The purpose of the research in this thesis consists in approaching some alternative methods to replace classic file scanning techniques. Particularly, our target is to demonstrate that Machine Learning algorithms can be used as an efficient mechanism for malicious PDF files detection. Additionally, we want to present a framework that can remotely analyze suspicious PDF files by applying earlier mentioned performant algorithms. This framework has several advantages:

- Takes on the complicated task of applying classification model for uploaded files and giving the analysis result to the users.
- It is designed to be deployed as a Cloud Application on a powerfull server that can handle multiple requests at the same time, lacking users care about having high specifications for their computers.
- Guarantees the privacy of the scanned documents. The algorithms doesn't require the documents content in order to decide whether they are malicious or benign. Also, after analysis process, none of the documents are stored on the Cloud.
- Keeps the detection algorithms up-to-date. It won't be the users responsability anymore to regularly update the software antivirus installed on their computers.
- Provides a generic implementation for analyzing uploaded files. The support for a new file extension to be scanned, can be effortlessly added by adding a classifier Machine Learning model for the wanted file format.

Integrating both Machine Learning and Cloud Computing into Cybersecurity should provide a significant progress to this field.

In order to fully automate the process of detection malicious PDF files, our application provides a tool (currently has support only for Microsoft Windows) that independently uploads detected files to the Cloud Service, waits for analysis results and takes appropiate protective measures. In the following we will introduce some operating system concepts that helped us to achieve the expected behavior for our application.

3.2 Background processes in Microsoft Windows

In computing, a background process is a process[1] that runs independently and doesn't require any user involvement. Usually it has the task of system monitoring and sending any types of notifications to the user. Each operating system has its own principle of implementation for background processes. We will have an in-depth look at background processes in Microsoft Windows, which are called **Windows Services**. According [14] these services are started by a central utility - *Services Control Manager* at the boot-time[2] of the computer and they don't have any graphical interface. For security and safety reasons, specifically to prevent user applications from accessing Windows Services that could run with high privileges, Windows OS creates a new session for each logged in user, reserving *Session 0* for non-interactive services. The mentioned features make Windows Services suitable for long-running functionality, whose operation doesn't interesect with other users work on the same computer. The support for creating a service is provided by **.NET Framework**, its features being possible to access by using one of Microsoft's high-level programming language C#.

3.3 File System monitoring

In the field of operating systems, a file system is aimed to store data in form of files in a long-term storage. Before being stored on physical devices, data that is created and saved by applications in User-Mode, goes through a chain of drivers from Kernel-Mode. This principle is adopted in many operating systems, inclusively in Microsoft Windows (see Figure 3.1). In order to track every change on the computer's file system, we need a monitoring mechanism. Throughout the development of Windows, serveral technologies have been implemented for integration by software engineers, including: File System Filter Drivers, Update Sequence Number Journal (*USN Journal*), Event Tracing for Windows (*ETW*), *FileSystemWatcher* Class from .NET. In this section we are focusing on two of the most popular Windows file system monitoring mechanisms: Minifilter drivers managed by *Filter Manager* and *FileSystemWatcher*.

[1] a container for a set of resources used when executing the instance of the program
[2] the period when a computer is starting

Windows Filter Manager is a Kernel-Mode driver that intercepts every file system I/O operation and can extend the functionality of the requested operation through each minifilter driver that is registered to it (see Figure 3.1). The order of attachement of each minifilter is identified by an altitude[3], thus avoiding interleaving the functionality of callback routines of each minifilter. Windows Driver Kit (*WDK*) provides the necessary SDK[4] for developing minifilter drivers, which are most suitable for Backup agents, Encrpytion managers and Antivirus filters. For more details consult [9].

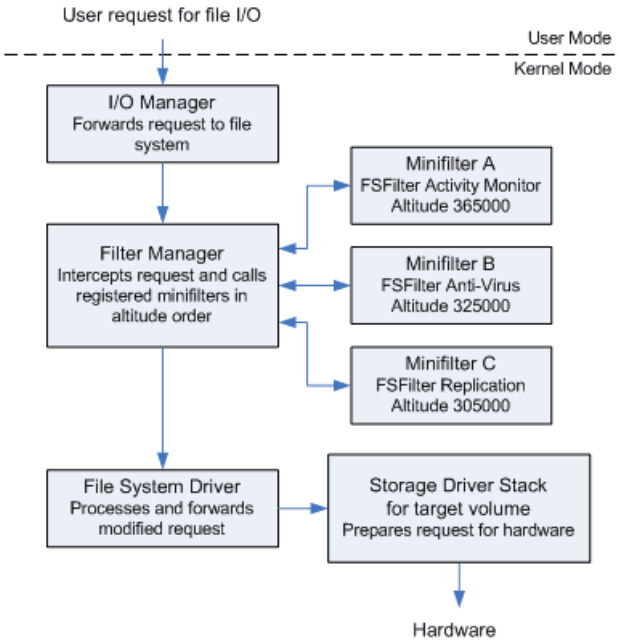

Figure 3.1: Simplified I/O stack from Windows [9]

Compared to the previous described technology, which is complex for development and mantaining, **FileSystemWatcher** is a class from .NET that makes file system monitoring straightforward to implement. According to [1], this class represents a wrapper over Win32 SDK *ReadDirectoryChangesW* function, which is also used by Windows Explorer to monitor folder changes. By being provided by .NET, it is clear, that FileSystemWatcher can be used in User-Mode for development (see Figure 3.2).

[3]unique identifier assigned by Microsoft that determines when a minifilter is loaded
[4]collection of software development tools

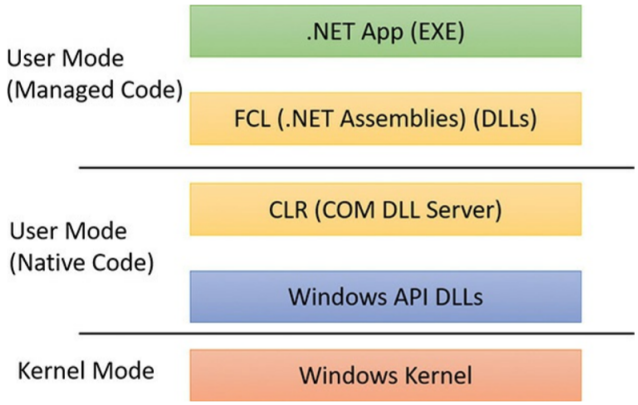

Figure 3.2: Relationship between .NET and Windows OS [14]

To configure a working instance of this class, we need to specify a couple of properties, such as *Path* (for indicating the folder to monitor), *NotifyFilters* (for specifieng what kind of file changes to monitor) and *EnableRaisingEvents* (to start/stop monitoring process). Also we need to attach the corresponding events that we want to receive (see Table 3.1).

Table 3.1: Events supported by FileSystemWatcher

Event	**Occuring time**
Created	When a new file is created or is moved into monitored folder
Changed	When a file has its content or file attributes modified
Deleted	When a file is removed or is moved out of monitored folder
Renamed	When the name of the file get changed

Earlier we mentioned that minifilter drivers live in Kernel-Mode, while FileSystemWatcher provided by .NET lives in User-Mode. One more argument in favor of FileSystemWatcher, besides the high level object-oriented programming language C#that we have at our disposal over unmanaged[5] C language, is that it's safer to run applications in User-Mode. Each application runs in isolation in User-Mode and in case of a crash, it won't affect other applications or even worse the OS itself, what can happen with a Kernel-Mode driver. Even the smallest error that occurs in a driver can cause BSOD[6].

[5]code executed directly by OS; does not provide any security to the application
[6]"Blue Screen of Death", error screen displayed by Windows in case of fatal system crash

3.4 Analyzing PDF File Structure

In this section we are going to understand PDF structure which is a foundation stone of identifieng backdoors[7] in this file format.

PDF was first developed by Adobe in the 1990's with many version been released afterwards. Its initial purpose was to allow to present various types of data, including: text, images, webpage links etc. regardless of the environments it was opened in. As explained in [5], PDF files consist of objects, which are of eight types: Boolean values, Numbers (integer and real), Strings, Names, Arrays, Dictionaries, Streams and the Null object. Each PDF document should begin with a header that identifies it as a *PDF* and includes the version number `%PDF-1.7`. The document should also end in a certain way - with the signature `%%EOF` (see Figure 3.3). After header the objects are declared, that can contain necessary information for rendering the document. One of the most important objects is the stream, that can store an unlimited size sequence of bytes. Usually streams are used for storing text, but they can also store executable code, i.e., checkbox for agreement of *Terms and Conditions*. Streams can contain the optional entry `/Filter`, which is aimed for compression and encryption of data inside streams. Next comes the Cross-Reference Table (*Xref*), which determines the location of existing objects in PDF. The last section of the file is *Trailer*, that contains the metadata of the file, e.g., size of the file, number of objects, the id of the root object, etc. Hence it becomes clear, that PDF documents are parsed from the end to the start.

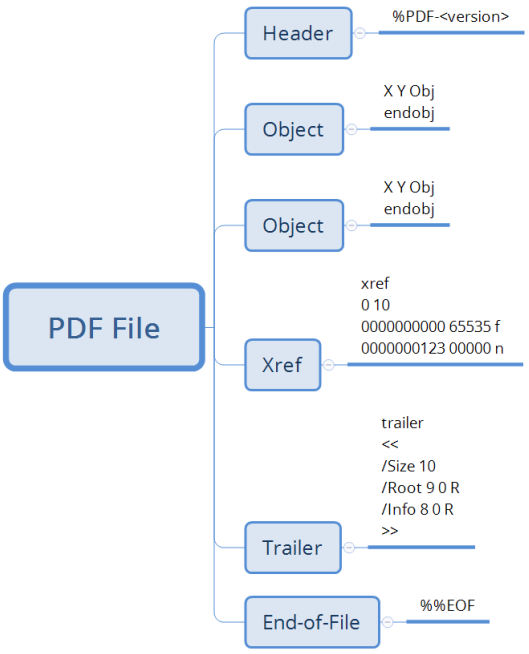

Figure 3.3: Structure of PDF Format [7]

[7] a malicious method by which a system bypass is hidden

3.5 Malware in PDF

Over the years PDF format has evolved and now it has support not just for text and images, but also for editable forms, animations and even 3D graphics. All of these features made PDF a great solution for information sharing. Meanwhile it also became a good target for cybercriminals, malware being easily embedded into PDF documents and widespread through different types of communication, e.g., HTTP, emails, etc. (see Figure 3.4)

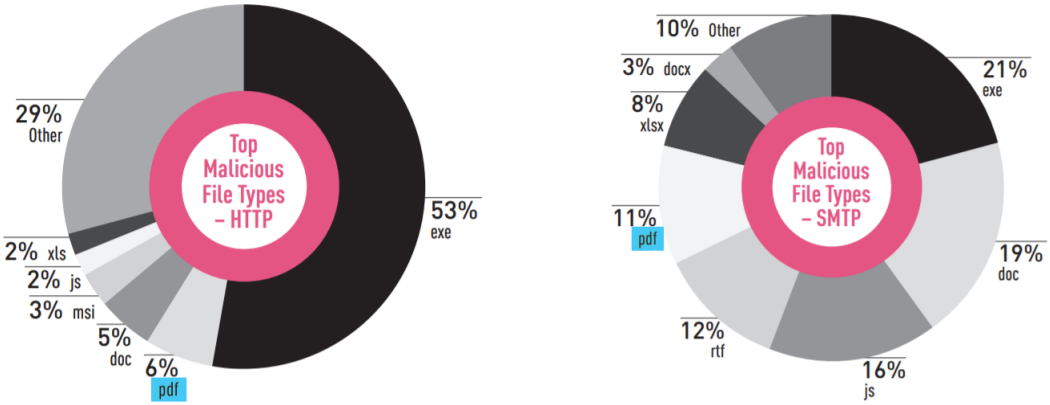

Figure 3.4: Top malicious file types - H1 2019 [10]

Malicious PDF can be classified into three categories: *reader exploitation*, *feature abuse*, *phishing attacks*. Every year numerous vulnerabilities related to most used PDF readers, e.g., Adobe Reader, Acrobat Reader, Chrome, are reported. Before these weaknesses are patched[8], attackers create exploits to execute arbitrary code, when PDF is opened with one of the vulnerable application. Phishing attacks are more difficult to detect, because in the context of PDF documents, they aren't infected, instead the social engineering is used in order to psychologically enforce someone to click on malicious links written in the documents. Another frequently used attack vector involves misusing the PDF extra-functionalities, such as running malicious code when file is opened, stealing sensitive information and sending to remote addresses, downloading infected executables and so on. Pursuant to [8], it's usually the fault of embedded Javascript code, whose intentions could be wrong, but as seen in Table 3.2 there are also several standard PDF entries, which seem to be interesting in terms of malware hunting. However, when searching for them we should not forget about the possibility of obfuscation using hex code, in which, for example, `/Launch` can turn into `/L0x61unc0x68`. Encrypting the streams and strings is another often applied "trick" to complicate the analysis work for both antiviruses and researchers. A PDF document from which it is impossible to copy text or which can not be printed, is such a obfuscated sample.

[8] applying changes to fix bugs or errors in a software program

Table 3.2: Standard PDF Entries whose functionality can be abused

Entry	**Functionality**
`/JavaScript`	Sets JavaScript code to be executed
`/OpenAction`, `/Names`, `/AcroForm`, `/Action`, `/AA`	Defines a script or an action to be automatically run
`/Launch`	Runs a program or opens a document
`/URI`	Accesses a resource by its URL
`/SubmitForm`, `/GoToR`	Can send data to indicated URL
`/ObjStm`	Hides objects inside Streams

3.6 Machine Learning for Malware Detection

Chapter 4

Proposed approach

4.1 Dataset

4.2 Proof of Concept

4.2.1 Feature Selection

4.2.2 Classification Techniques

4.2.3 Performance Evaluation

4.2.4 Experiment

4.3 Used technologies

4.3.1 Microsoft .NET Framework

4.3.2 PDF Tools and Metasploit Framework

[15]

4.3.3 Python Flask

4.3.4 Scikit-learn Machine Learning Library

[3] [13]

4.3.5 ReactJS Framework

Chapter 5

Application

5.1 Design

5.2 Windows Service

5.3 Cloud API

5.4 Dashboard Interface

Chapter 6

Conclusion and future work

Bibliography

[1] Tom Archer and Nishant Sivakumar. *Extending MFC Applications with the .NET Framework*. Addison-Wesley, 2003.

[2] Bitdefender. Sandbox analyzer. link. [Online; Accessed 1-April-2020].

[3] Clarance Chio and David Freeman. *Machine Learning and Security*. O'Reilly Media, Inc., 2018.

[4] R. Fettaya and Y. Mansour. Detecting malicious pdf using cnn. *International Conference on Learning Representations*, 2020.

[5] Adobe Systems Incorporated. *PDF Reference, sixth edition: Adobe Portable Document Format Version 1.7*. Adobe, 2006.

[6] P. Laskov and N. Srndic. Static detection of malicious javascript-bearing pdf documents. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 373–382. ACM, 2011.

[7] LogRhythm. Detecting potentially malicious javascript embedded within a pdf file using logrhythm netmon. link. [Online; Accessed 11-April-2020].

[8] Xakep Magazine. Looking for exploits in pdf-documents on our own. link, 2014.

[9] Microsoft. Microsoft docs. link. [Online; Accessed 1-April-2020].

[10] Check Point Research. Cyber attack trends: 2019 mid-year report. link. [Online; Accessed 11-April-2020].

[11] Hispasec Sistemas. Virustotal. link. [Online; Accessed 1-April-2020].

[12] Didier Stevens. Pdf tools. link, 2008.

[13] Emmanuel Tsukerman. *Machine Learning for Cybersecurity Cookbook*. Packt Publishing, 2019.

[14] Pavel Yosifovich, Alex Ionescu, Mark E. Russinovich, and David A. Solomon. *Windows Internals*. Microsoft Press, 2017.

[15] Lenny Zeltser. link. [Online; Accessed 1-April-2020].